第4章 オリジナル・ボードをSDSoC対応にする…ステップ2

I/O対応版プラットフォームの作成

横山 雅一 Masakazu Yokoyama

ここでは，第3章で作成したZynqの内部だけで完結する最小構成のプラットフォームにI/Oの機能を追加して，実際のボードで使用できるプラットフォームに発展させます．外部とつながるデバイスを制御する機能ブロック（IPコア）をあらかじめ組み込んでおくことで，ソフトウェアからはレジスタ操作だけで使えるようにしています．

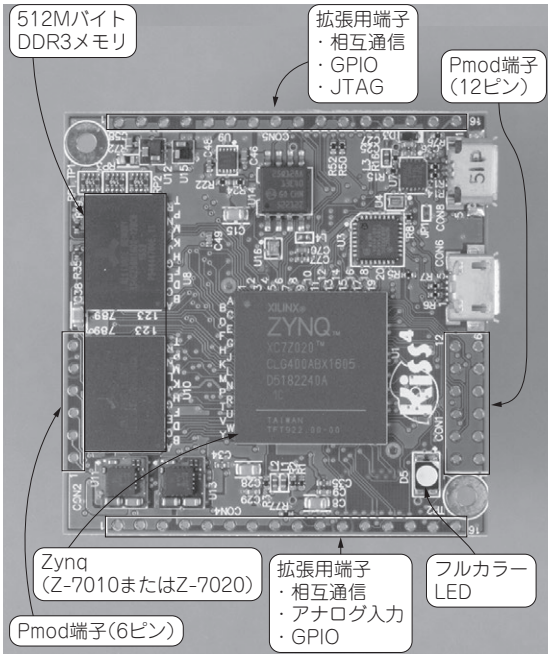

写真1　ターゲットのZynqボード

1．I/Oデバイス制御用のIPコアを組み込む

第2章Appendixで説明したボード（**写真1**）は，FPGAから簡単にアクセス可能なI/Oとして4個の青色LEDと1個のフルカラー高輝度LED，Pmodコネクタを搭載しています．これらのようなボードに特有のI/OもSDSoCから使えるようになると，ユーザにとっては利便性が上がります．

ここではSDSoCで採用されているLinuxのUIOを使ったプラットフォームの作成例を紹介します．

なお，Standaloneではレジスタ群を直接扱えるためUIOを必要としません．

● オリジナル標準プラットフォーム

写真1のボードで採用したSDSoC用の標準プラットフォームを**図1**に示します．

本標準プラットフォームでは，以下のような構成を採ります．

- LEDに対して今回の記事のために設計したPWM（Pulse Width Modulation）コアを接続
- 12ピンのPmodコネクタにはOLED制御用のIPコア（Diglent製）を接続
- 6ピンのPmodコネクタにはI^2C通信用のIPコア（Xilinx製）を接続
- 拡張コネクタにはUART通信用のIPコアを接続

またZynq内蔵のA-Dコンバータが使えるように，XADC用のIPコア（Xilinx製）も追加してあります

● UIOに対応したハードウェア記述ファイル

プラットフォームの作成方法は，第3章で説明したシンプル版プラットフォームと大きく変わりません．Vivadoのハードウェア・プロジェクトには，プラットフォーム提供者が付け加えたIPコアがあるので，それらをSDSoCのハードウェア記述ファイルを作成する際に追加します（**リスト1**）．

変更点は以下の通りです．

- 使用するクロックを1系統に簡略化
- `M_AXI_GP0`を`pfm_axi_port`では指定しないようにした
- `sdsoc::pfm_iodev`によりIPコアと`uio`の対応をした

`pfm_iodev`では`uio`でアクセス可能なIPコアを指定します．ここで`uio`を指定する順番は，後述のデバイス・ツリーで指定する順番と合わせておきます．

● デバイス・ツリーにUIOの記述を追加

UIOを使用したことをLinuxのカーネルに教えるため，デバイス・ツリーを書き換えます．SDSoCに付属の`devicetree.dtb`をいったん`dts`（編集可能なソース）に変換し，**リスト2**のように編集後，再度