第2章 FPGA内蔵メモリ・ブロックを
デバイスの違いを意識しないで使うテクニック

VGA表示回路を例にした My回路のIPコア開発とライブラリ化

石原 ひでみ Hidemi Ishihara

Cyclone V SoCもZynqも，どちらも画面表示系のコントローラを内蔵していません．そこで特集ではFPGA部分にVGA表示回路を実装して，Linux上からフレーム・バッファとして使ってみることにします．その第1段階として．VGA表示回路をIPコアとしてライブラリ化し，Altera社およびXilinx社製FPGA開発ツールに登録します．IPコア・ライブラリ化することで設計資産となり，今後の設計に再利用もしやすくなります．

1．自分の設計資産をIPコアとして活用する

● 独自回路はIPコアにするところから始まる

昔からFPGAを開発してきた人は，それまでの設計資産（プロジェクト・ファイル一式や一部のHDLソース・コード）を流用して，次のFPGA開発を進めていることが多いと思います．この場合，ファイルをコピーして少しカスタマイズし，回路を設計している方も多いのではないでしょうか？

カスタマイズするにはいろいろ理由がありますが，ほとんどの場合，使用するボードや接続する他の回路の仕様の都合に合わせて，てっとり早く修正して仕上げるためでしょうか．筆者も以前までは，それまでに設計した回路を使い回し，I/O周りのつじつまを合わせるために少しだけ手を加えて使用することが多々ありました．

● ビルディング・ブロック開発

FPGAは今後も，ゲート規模の増加やARMコア・プロセッサを内蔵するなど，ますます大規模化/複雑化していくことでしょう．そのような中で，プロジェクト・ファイルを流用するような設計手法を使用していては，開発が立ち行かなくなってくるでしょう．

ビルディング・ブロック開発とは，プロジェクト・ファイルやHDLソースのファイルをコピーしてそれぞれカスタマイズして使っていくのではなく，標準的なライブラリとしてIPコアを登録しておき，必要なときにそれを使用して開発を行う手法です．FPGAの大規模化/複雑化には必須の開発手法です．

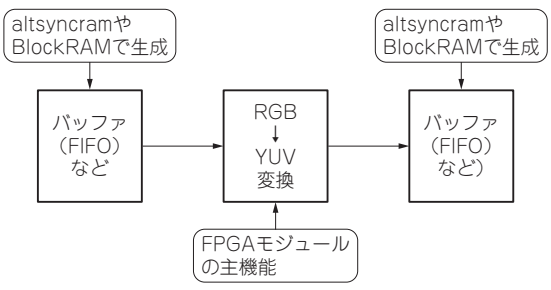

図1 RGBをYUVに変換する回路

● FPGA内蔵メモリを使う例

通常，FPGAの開発は，実装するターゲット・デバイスを意識しながら行うことが多いと思います．例えば何らかのバッファ・メモリが必要になり，FPGA内蔵のメモリを使おうとする場合，ターゲット・デバイスがXilinx社製のFPGAであればBlock RAM（BRAM）を，Altera社製ならaltsyncramを意識しながら設計を行うでしょう．

もう少し具体的な事例を挙げましょう．

図1のように，画像系の色変換回路のRGBをYUVに変換する回路を例にとると，変換回路の入出力部分にバッファ・メモリを置くケースが多々あります．こういう場面でのバッファは，デバイス固有の機能（プリミティブ）を直接使用することも多いのではないでしょうか？

● メモリを論理合成ツールに類推させる方法もあるが

デバイス固有の機能を直接使うのではなく，論理合成ツールに推論させる記述で作成することもあるでしょう．実際，**図1**のように主機能に当たる演算部分はRTLで式を記述し，デバイス固有の機能を使用して回路を明示的に記述することは多くないと思います．

しかし，2ポートRAMとして使う場合など，論理合成ツールに類推させるのが難しいときもあるので，やはり固有機能を直接使いたくなるのです．

● デバイス固有の機能は直接は使わない

固有の機能を直接使った設計は，それをそのまま他社のFPGAに持っていくことはできません．デバイス固有の機能を一切使わない汎用的な記述のみのRTL設計は，IPコア化することにより，開発した回