デバイスの記事

第6回（最終回）

FPGAプロトタイピング環境を利用したハードウェア開発（後編）

大庭信之，森岡澄夫，高野光司

前回（本誌2003年2月号，pp.136-141）は，筆者らが設計・開発した2種類のFPGAボードと，それを使ったプロトタイピング環境について紹介しました．今回は，FPGAプロタイピング環境の利用法について，二つの実装例（暗号プロセッサ，PCIバス）を挙げて説明します．（筆者）

開発例1：暗号プロセッサ

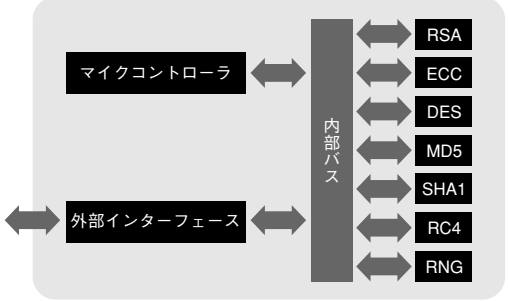
最初の実装例は暗号プロセッサです．1個のASICにいろいろな暗号機能を盛り込んだチップです．**図1**は，暗号プロセッサの内部ブロック図です．中央にある内部バスをはさむようにして，左側にマイクロコントローラと外部インターフェース，右側に暗号演算マクロが並んでいます．RSA，ECC，DES，MD5，SHA1，RC4，RNGは，暗号の基本演算を行う機能モジュールと考えてください．

●暗号プロセッサの特徴

このプロセッサの特徴を以下に示します．

- モジュール化した各種暗号演算回路を搭載．ユーザのリクエストにより，必要な暗号コアだけを組み合わせることができる．
- 各マクロが連携して動作し，いろいろなアプリケーションに対応可能．また，細かい動作は内部のマイクロコントローラのファームウェアを変更することで調整できる．
- 暗号かぎの生成といった機密性の高い演算は，チップの中に閉じ込めることができる．
- 各マクロは1種類ではなく，ユーザの要求によって性能，サイズ，消費電力などをカスタマイズできる．
- PCIや各種マイコンのバスに直結できるように，いくつかの外部インターフェースを用意している．
- 設計はすべてRTLのVHDLで記述されており，複数のプラットホーム（EDAやデバイス・テクノロジ）上で論理合成できる．

将来的にはもっと高性能なマイクロプロセッサを組み込むことも考えられます．しかし，今回は複雑な演算についてはすべて暗号演算器が行うので，マイクロコントローラはそれらの制御だけに専念できます．性能的には8ビット・マイコンで十分でした．



RSA（Rivest Shamir Adleman）	公開かぎ暗号方式の一つ．
ECC（elliptic curve cryptography）	公開かぎ暗号方式の一つ．
DES（data encryption standard）	共通かぎ暗号方式の一つ．8バイトごとに暗号化を行う（ここでは乱数の平滑化に利用）．
MD5（message digest algorithm 5）	電子署名などに利用されるハッシュ関数の一つ．
SHA1（secure Hash algorithm 1）	電子署名などに利用されるハッシュ関数の一つ．
RC4	共通かぎ暗号方式の一つ．1バイトごとに暗号化を行う．
RNG（random number generator）	乱数発生回路

〔図1〕暗号プロセッサ
中央の内部バスをはさむようにして，左側にマイクロコントローラと外部インターフェース，右側に暗号演算マクロがならんでいる．

●開発上難しかった点とその解決法

暗号プロセッサの設計には三つの大きな壁がありました．

- 非常に多くのクロック・サイクルを要する演算があり，シミュレーションにたいへん時間がかかる．
- そのような長い演算をさらに数種類組み合わせて一つの機能を実現しなければならない．
- いろいろな外部バスへインターフェースするため，複数のバス・プロトコルを検証しなければならない．

RSA演算を例にとります．RSAは現在広く使われている公開かぎ暗号の一つです．その演算内容は，簡単に言うと，XのY乗を計算し，それをZで割った余りを求めるというものです．X，Y，Zが小さい数の場合，普通のマイクロプロセッサでも十分高速に計算できますが，実用になるRSAではX，Y，Zが512ビット以上の長さを持ちます．したがって，演算時間はたいへん長くなります．もっとも，すばやく計算できてしまうと暗号の意味がありませんが…．

暗号プロセッサにはRSA演算専用のコアが組み込まれており，コア内部では16ビットまたは32ビットの乗算器を使います．16ビット・コアは32ビット・コアよりも演算時間がかかりますが，サイズは小さくなります．このほか，8ビット・コアや64ビット・コアもありますが，ここでは省略します．どのコアを用いるかは，ユーザの要求（演算速度，消費電力，チップ・サイズ）に合わせて選択します．

表1は，演算ビット数と使うコアの組み合わせ，サイクル数，そして1回の演算をソフトウェア・シミュレーションで行った場合のシミュレーション時間とFPGAを使ったエミュレーション時間を示しています．ソフトウェア・シミュレーションは，短いもので数時間，長いものでは数日間を要します．一方，FPGAエミュレーションは，長いものでも10秒程度であり，その差は圧倒的です[注1]．

X，Y，Zの個々の値はほとんど無制限にとることができるので[注2]，いろいろな組み合わせをなるべく多くシミュレーションしたいところです．しかし，X，Y，Zの一つの組み合わせについて，結果を得るのに数日もかかってしまったのでは，たくさんの組み合わせを試すことができません．

最近，広く使われているSSLなどのインターネット・セキュリティ機能を例にとると，このRSA演算を使ってサーバとクライアントの間で暗号かぎを交換し，さらにそのかぎを使ってDES暗号によるコンテンツを配信するといったことが行われています．すなわち，**表1**に示したRSA計算が単独で使われるということはまれで，ほかの暗号などと組み合わせて使われることが多いのです．そうなると，ソフトウェア・シミュレーションでその一連の動作を検証することは不可能とはいいませんが，たいへん時間のかかる作業となります．今回，そのような時間のかかる演算の検証はFPGAプロトタイピング環境でのみ行いました．

●FPGA容量の問題と解決策

ところが，FPGAへマッピングできる回路規模には限界があります．実際，**図1**に示した暗号プロセッサのすべての論理回路を一つのFPGAへマッピングすることはできませんでした．そこで，暗号マクロを分けてFPGAへマッピングし，別々に検証することになります．

1個のFPGAに入りきらない設計データを検証するうえでひとくふうしてみました．あらかじめ暗号マクロごとに別々のFPGAコンフィグレーション・ファイルを用意しておき，それらを検証実行手順の最中にFPGAへ順次書き換えていくという方法です．筆者らは，これを「動的FPGAリコンフィグレーション」と呼んでいます．

図2を見てください．これは，ホストとなるパソコンのPCIスロットにFPGAプロトタイピング・ボードを挿した状態を示しています．FPGAのコンフィグレーションは，ホストのパソコンのパラレル・ポートに接続された，別のコンフィグレーション・ケーブル（今回は米国Altera社の「ByteBlasterMV」）を通して行います．

〔表1〕RSA演算サイクル数とシミュレーション時間

設計例 RSA演算	サイクル数	ソフトウェア・ シュミレーション	FPGA エミュレーション
1024ビット 32ビット・コア	400万	数時間	約0.3秒
2048ビット 32ビット・コア	3200万	数日	数秒
1024ビット 16ビット・コア	1600万	約1日	数秒
2048ビット 16ビット・コア	1.02億	数日以上	約10秒

注1：ソフトウェア・シミュレーションとFPGAエミュレーションの速度差はどのくらいだろう．FPGA内では，もともと同時並列に動作する論理イベントは，そのとおり同時並列に動く．一方，ソフトウェア・シミュレーションでは同時並列に動作する論理イベントについても順番に実行していく（仮想的に同時並列動作させている）．つまり，もとの設計で同時並列動作する部分が多ければ多いほど，速度差が現れるといってよいだろう．筆者らの経験では，およそ2～5けた，FPGAエミュレーションのほうが速いようだ．

注2：あくまでもXのY乗をZで割った余りを計算するという意味．有効な暗号という点ではX，Y，Zの値には制限がある．