第5章

HDLによるLSI設計を体験する

■Mentor Graphics社のHDL設計環境「FPGA Advantage」■

森田栄一

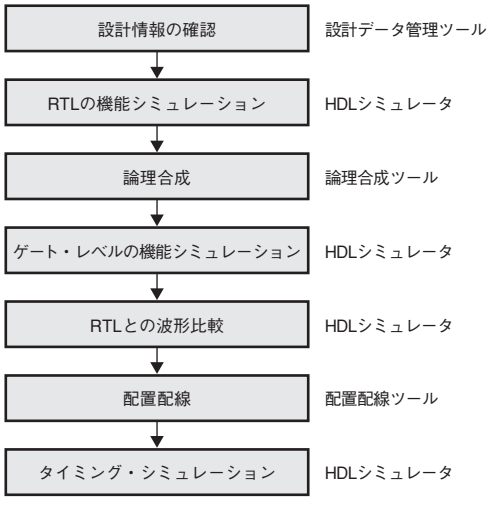
ここでは，キッチン・タイマの回路を例にHDLを利用したFPGAの設計フローについて解説する．使用する設計ツールは，米国Mentor Graphics社のHDL設計環境「FPGA Advantage」である．本ツールは，設計データ管理ツール「HDL Designer Series」，HDLシミュレータ「ModelSim」，論理合成ツール「LeonardoSpectrum」を備えている．また，配置配線には米国Xilinx社の「ISE」を使用する．Mentor社のツールについては，本誌付属のCD-ROMに評価版が収録されている．（編集部）

HDL（hardware description language；ハードウェア記述言語）設計を始めるとき，どこから学び始めますか．たいていは，参考書を読んだり，だれかの書いたソース・コードを変更してシミュレーションを流してみたりといったところから始めるでしょう．ただし，ひと言にHDLといっても，幅広い記述レベルに対応しており，使用目的も単一ではありません．実際の設計でも，その工程によって記述の内容や出力結果が大きく異なってきます．

本チュートリアルは，「設計フローの全体を見通す」という視点からHDL設計の各工程における「使用目的」を理解してもらうことを目標としています．

- 自分の作業の方向性を的確に見つけることができる
- そのために前後の工程の意味を理解する

このような観点で，チュートリアルによる設計フローを体験してもらえればと思います．



〔図1〕チュートリアルの流れ

本チュートリアルは，図のような流れで設計を体験する．右側に示しているのは，各ステップで使用する設計ツールである．

●チュートリアルの概要

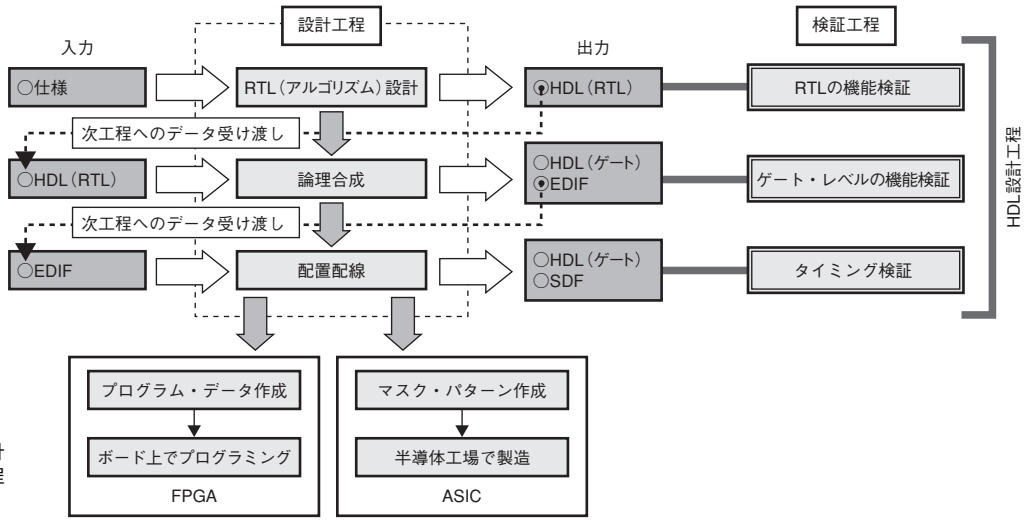
本チュートリアルの流れを図1に示します．フロー全体を通して，下記のツールを使用します．

- 設計データ管理：HDL Designer Series（米国Mentor Graphics社）
- 検証：ModelSim（Mentor社）
- 論理合成：LeonardoSpectrum（Mentor社）
- 配置配線：ISE（米国Xilinx社）

各ソフトウェアのインストール方法とチュートリアル・データの設定方法については，本誌付属のCD-ROMに収録されているドキュメントをご覧ください．

HDL記述としては，すでに用意したものを利用します．Verilog HDLの文法の詳細については，ここでは述べません．しかし，本チュートリアルでは極力簡易な記述に絞っているので，プログラミング言語などに触れたことのある方には十分に理解できる内容となっています．ただし，特にキーとなる動作部分については解説を行います．

また，各ツールの使用方法の説明は最低限にとどめています．使用方法の詳細については，各ツールに添付されているドキュメント類を参照してください．



〔図2〕
HDL設計フロー
もっとも基本的なフローを示す．設計対象によっては，いくつかの検証工程が追加される場合がある．

1 HDLによるLSI設計の流れ

最初にHDL設計で何ができるのかを簡単に説明します．ご存じのように，LSIのようなディジタル回路は論理素子（andやorなど）の組み合わせで構成されています．今，論理素子を組み合わせて，次の動作を実現したいとします．

```
if ((a > 0) && (b == c)) begin
     d = a + b - 1;
     f = a * b + c;
end
```

このような記述から，論理ゲートの組み合わせをすぐに思い浮かべられる方はほとんどいないでしょう．HDLでは，このようなアルゴリズム・レベルの記述を「論理合成ツール」を使うことによって論理素子の組み合わせ（ゲート・レベル記述）に変換することができます．HDLが多く使用されるようになったのは，この「変換」の部分がスムーズに行えるようになったからにほかなりません．

●HDL設計のフローでは「検証」がキーになる

HDLとして使用されているものには，主としてVHDLとVerilog HDLの二つがあります．本チュートリアルではVerilog HDLを用いますが，いずれの言語を使用した場合でも設計の方法やフローに差はありません．

図2にHDL設計フローの概略を示します．最初に，RTL（register transfer level）のHDLを利用して機能記述を作成し，動作の検証を行います．動作に問題がなければ，論理合成，配置配線へと設計工程を進めていきます．工程を進めるごとにHDL記述の内容が，アルゴリズム・レベルから実チップ・レベルへと変わっていきます．この記述の変化に対して，**表1**に示す3種類の検証が必要となります．

〔表1〕三つの検証工程

	RTLの機能検証	ゲート・レベルの機能検証	タイミング検証
HDL記述	RTL	ゲート・レベル	ゲート・レベル
ゲート・ライブラリ	不要	必要	必要
遅延情報	使用しない	微小な遅延*1	実遅延情報
テストベンチ	同　一		
前工程からの変更点	——	HDL記述（機能情報）	遅延情報
検証項目	アルゴリズム	RTLとの等価性	タイミング動作

*1：動作上問題とならない微小な遅延時間を付加する．

「RTLの機能検証」では，アルゴリズム・レベルの記述が期待どおりに動作するかどうかを確認します．「ゲート・レベルの機能検証」では，論理合成によって生成されたゲート・レベル記述が，「RTL」と同様の動作を行うかどうかを確認します．これは，論理合成ツールの「変換」にともなうトラブルを検出することが目的です．最近の論理合成ツールは性能が良くなっているので，論理変換で失敗するということはほとんどありません．しかし，RTLの記述方法が「論理合成向き」でない場合には，結果の不一致が起こる可能性があります．記述が「論理合成向き」であるかどうかを検証するツールもあります．最後の「タイミング検証」では，ゲート・レベルの回路に「実遅延情報」を付加することで，実際のチップになったときの動作遅延による問題を検証します．