第2章

レッツ・エンジョイ・“Cベース設計”！

—CRT制御，ブロック崩しゲームの製作事例

井倉将実，桑野雅彦

C/C++を拡張した言語を利用してハードウェア（LSI）を設計する手法が脚光を浴びている．ここでは，こうした言語の一つである「Handel-C」を使ってFPGAを設計し，LEDダイナミック点灯やシリアル通信，CRT制御，ブロック崩しゲーム，3次元ワイヤフレーム表示を実現する．新人エンジニアやソフトウェア技術者など，ハードウェア設計の専門家でなくても，プログラミング感覚で気軽にFPGAを設計できる．なお，本記事で使用した設計ツール（評価版）やサンプル記述，3次元ワイヤ・フレーム表示の製作記事は，本誌付属のCD-ROMに収録されている．（編集部）

新人エンジニアのみなさんは，「LSI設計」と聞いて，どのような作業を想像されるのでしょうか．新人エンジニアを受け入れる側は，VHDLやVerilog HDLといったハードウェア記述言語（HDL：hardware description language）について，大学の授業や研究室などである程度の手ほどきを受けてきているものと期待します．ところが現実には，C/C++やJava，Perl，FORTRAN（！）といったソフトウェア言語は知っていても，ハードウェア記述言語は扱ったことがないという人が多いようです．中には，習得しなければならない課題の多さに，不安を感じている新人エンジニアの方がいらっしゃるかもしれません．

1．Cベース設計なら，新人のほうが有利!?

ですが，みなさん，朗報があります．最近では，HDLではなく，もっとポピュラなソフトウェア言語，すなわちC/C++をベースとする言語でLSIを開発する手法の検討が進んでいます．

LSI設計という限られた領域ではなく，システム全体（電子機器，ソフトウェア，アルゴリズムなど）の開発業務において，もっともよく使われている言語と言えば，やはりC/C++でしょう．Windows上で動作する使いやすい開発ツールが出回っていますし，生産性やコードの可読性，各種プラットホームへの移植性にも優れています．C/C++は，昨今の工業高校や工学系の大学では必須の履修科目となっています．書店に行って工学書の棚を見れば，そこには膨大な数の参考書が並んでいます．

● LSI設計の敷居を下げるCベース設計

C/C++によるLSI設計の目的はいくつかありますが，そのうちの一つは，「多くの人が慣れ親しんでいるソフトウェア言語でハードウェアを設計できるようにしよう」というものです．もっと端的に言えば，「LSI設計の敷居を下げて，システム設計者やソフトウェア開発者にもLSIを設計してもらおう」ということです．

そういう意味で，現在，学校を卒業して企業に入られたみなさんは，現役のハードウェア技術者よりも恵まれた状況にあると言えるのかもしれません．最新の設計技術を活用するときに，すでに会得しているソフトウェア言語の基礎知識がそのまま役に立つからです．

従来から，C言語を操るプログラマがLSIの設計を行うことは不可能ではありませんでした．というのも，標準言語の一つであるVerilog HDLの構文が比較的C言語に似ているためです（VHDLはPascalスタイル）．しかし，結局のところ，Verilog HDLは「C言語と似て非なるもの」であり，記述のかなりの部分でハードウェア（詳細な回路構成）を意識しなければなりませんでした．とにかくLSIを小さくする場合，高性能にする場合，コストを重視する場合，複数の異なるクロックで動作する機能ブロックを接続する場合

などは，どうしてもLSIを構成する回路の動作をきちんと理解したうえで設計する必要があります．これが，LSI設計の敷居を高くしていたいちばんの理由だと思います．

ところが，昨今のC/C++（および，C/C++をベースとした言語）によるLSI設計環境では，回路の構造をほとんど意識しなくても，実現したい機能（アプリケーション）を実現できます．本稿の後半では，実際にCベースの言語による設計事例を紹介しています．一部，LSIの細部にかかわる部分もあるのですが，コードの本体はまさにC言語そのものです．見慣れない記述や関数は，外部ライブラリを使用しているものと考えれば，理解が早いかと思います．

● 大規模回路になると，記述量削減を実感できる

HDLによるハードウェア設計に長年携わっている人は，C/C++によるサンプル記述を見ても，「従来のHDLと比べて，記述量は変わらないのでは？」と感じることが多いようです．確かに，後述のLEDを1個点滅させる程度の回路では記述量はほとんど変わりません．しかし，例えば3次元グラフィックス処理回路（30万ゲート以上）などになると，その違いが如実に出てきます．C言語の再帰処理（リカーシブ・コール）を使えば，繰り返し処理や回路ブロックの再利用などを数行で記述できます．Verilog HDLやVHDLと異なり，ちまちまとライブラリ・コール関数を書き続ける必要はありません．

また，C/C++ベースの設計では検証作業が楽になります．Windows上で機能検証を行う場合は，米国Microsoft社のVisual C++フレームワークの環境を利用できます．Windows APIを駆使して，アプリケーション・ソフトウェアと同じようにデバッグを行うことも可能です．このように，既存のC/C++の開発環境を利用することによって，LSI開発において大きな工数を費やしている機能検証の期間を短縮できます．C/C++のデバッグ環境をそのまま利用できるという点だけをとっても，HDL設計よりC/C++ベース設計のほうが優位にあると筆者は考えています．

リスト1　Hello world！

「Hello world！」という文字列を表示させるC言語のソース・コード．printf()関数は外部サブルーチン・コールとして使用される．

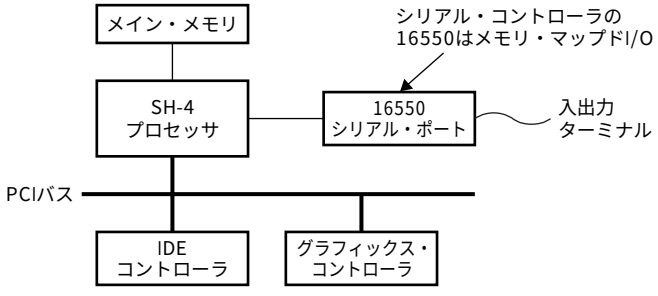
```
/*
  Hello world ! サンプル・コード
*/

#include <stdio.h>

//-------------------------------------------------

int main() {

  printf("Hello world !¥n");

  while(1) {};

  return 0;     // int main()しているのでリターンが必須
                // 付けないとワーニングを出されるだろう
}
```

2. Cベース設計の概要をつかむ

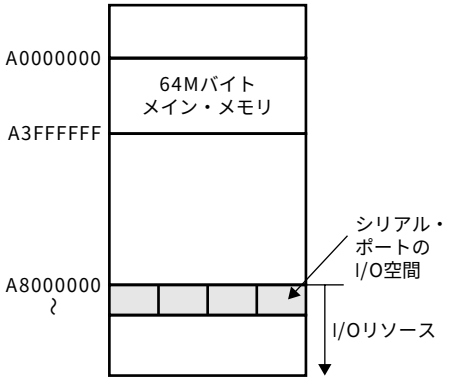
Cベース設計の詳細について説明する前に，**リスト1**のような画面表示のソース・コードに対するハードウェアの動きを考えてみましょう．ターゲットとなるシステムは，SH-4プロセッサにメモリと入出力用のシリアル・ポートがついたワンボード・マイコンとします（**図1**）．

● まずはコードとハードウェアの関係を理解

C言語で書かれたソース・コードをCコンパイラに入力



（**a**）回路構成



（**b**）メモリ・マップ

図1　ターゲットとなる回路

SH-4プロセッサの周辺に，PCIバス接続でハード・ディスクやグラフィックス・コントローラを実装した例．メイン・メモリやシリアル・ポートはSH-4プロセッサに直結されている．