論文

HDL（回路記述言語）の制御システムへの応用

森　久直* 坂巻佳壽美*

An application of HDL(hardware description language) to a control system

Hisanao MORI and Kazumi SAKAMAKI

Abstract　This report describes the method of developing a control system without being conscious of the conventional integrated circuit(IC). In the method, the field programmable gate array(FPGA) having a logic structure which can be configured arbitrarily was used. The FPGA configuration needs HDL, and so the processer and the IC ware described by HDL. A designer can use and select intellectual property(IP), and a designer can change the bit width of the bus. In other words, the configurable and scalable control system was realized.

Keywords　HDL, FPGA, Configurable, Scalable, Processer, IC, IP, Control system

１．はじめに

テキサスインスツルメンツの 74xx シリーズ TTL や，8 ビットマイコンの周辺 LSI である 82xx シリーズ LSI など，かつては店頭での品揃えが豊富だった。今日では，これらの IC を見かけなくなっている。仮に求めていた IC を見つけ出せたとしても，在庫数はあてにならない。それは，半導体メーカーの IC の販売方針が変わったためである。いつ売れるかも分からないような数個からの店頭での販売は行わず，数万個から数十万個の受注販売を行うようになったのである。従って，これらの IC を求める場合には，中小企業は大量の IC を購入しなければならず，さらには数ヶ月の納期を待たなくてはならない。

事実上，中小企業は既存の IC を用いた製品開発や製造が困難になったのである。

その一方では，開発現場で任意に論理を構成できる半導体デバイス（FPGA：Field Programmable Gate Array）が登場した。この FPGA の中には，AND や OR，フリップフロップなどの基本的な論理ゲートが大量にかつ規則的に配列されている。これらの間にある SRAM などの記憶素子を結線することで，目的とする論理回路ができるのである。　そして，FPGA に与えるべき結線データを作成するためには，回路記述言語（HDL：Hardware Description Language）を用いる。FPGA を提供している半導体メーカーは，パソコン上で使用する開発ツールを同時に提供している。この開発ツールを使用することで，HDL で設計した記述から FPGA に書き込む結線データを作成できる。つまり，FPGA で，既存の IC を再現できるということである。それは，既存の IC で構成してきた制御システムのハードウェアを FPGA に置き換えることができるということである。

そこで，HDL と FPGA を用いて中小企業の利用目的に合った制御システムを設計開発し，その有効性を検証した。

２．制御システムの HDL による設計

制御システムの設計開発では，次のツールを用いた。

○論理合成ツール（Synplicity-Synplify）

HDL 記述から中間ファイル（ネットリストファイル）を作成する。

○配置配線ツール（Altera-MAX ＋ plus Ⅱ）

ネットリストファイルから FPGA に書き込む結線データを作成する。

○ FPGA（Altera-FLEX10K100）

結線データを書き込むデバイスである。10 万のゲート数があり，プロセッサをはじめとする多くのゲート数が必要となる回路を書き込める。

2.1 FPGA による制御システムの実現

プロセッサとその周辺回路が，１個の FPGA の中に実現した。プロセッサについては，制御システムに必要な命令を入れて独自に設計開発した [1)]。周辺回路は，メモリ，I/O，シリアル/パラレル変換回路，タイマカウンタ

*情報システム技術グループ

などを HDL で設計開発した[2)]。各々の周辺回路については，予め動作検証を行っている。その後，これらの回路で制御システムを一つ構成し，動作検証を行った。

この結果から，既存の IC が入手できなくても，FPGA と HDL を用いることで制御システムのハードウェアを構成することができ，その有効性を検証した。構成した制御システムを図１に示す。

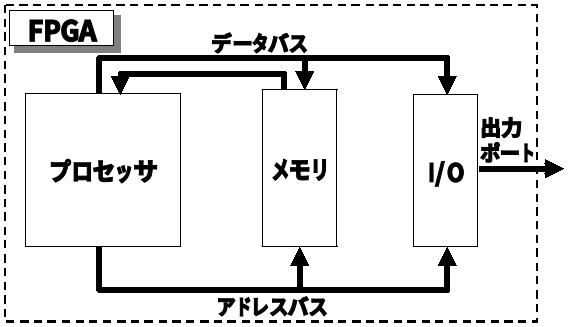

図１　制御システムのブロック図

2.2 制御システムの処理能力を可変にする設計

開発する制御システムによっては，その中で取り扱うデータの処理能力が異なる。そこで，制御システムの処理能力を変更できる（スケーラブル）ようにした。ここでいう処理能力とは，制御システムにおいて取り扱うことができるデータの大きさである。例えば，プロセッサとメモリの間で伝送されるデータのビット幅が 16 ビットと 32 ビットでは，後者の方が高精度な演算を行える。つまり，この場合はデータのビット幅が大きいほど，プロセッサとメモリの間の処理能力が高いということである。

そこで，プロセッサとメモリなどの周辺回路をつなぐデータバスのビット幅を数値で設定可能にすることにより，制御システムの処理能力を変更できるようにした。

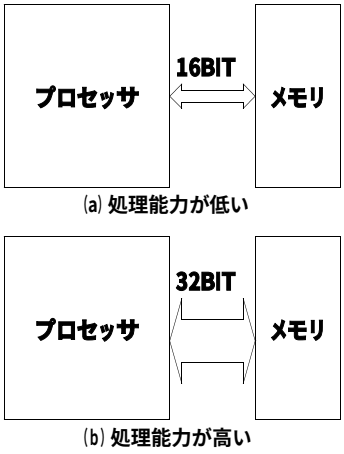

図２　スケーラブル化による処理能力差

プロセッサやメモリなどの周辺回路は HDL で記述したが，その記述の中にデータバスのビット幅の数値を設定する箇所を用意した。この記述箇所でのビット幅の数値設定により，プロセッサ-メモリ間，プロセッサ-I/O 間などのデータバスのビット幅が変更可能となった。変更できるビット幅は８，16，32 ビットである。組み込み機器を手掛けている企業の多くがこのビット幅を採用しているからである。

2.3 制御システムを再構成可能にする設計

プロセッサの周辺回路を記述した HDL ソースリストを設計資産（IP：Intellectual　Property）として，選択利用を可能にした（コンフィギュラブル）。図３に示すように，制御システム毎に初めから新しく設計開発を行うよりも，過去に設計した IP を部品のように選択してプロセッサと組み合わせ，新しく組み込む回路のみを設計開発した方が，短期間で設計ができる。設計期間，コストの削減に結びつくので，IP の再利用は有効である。

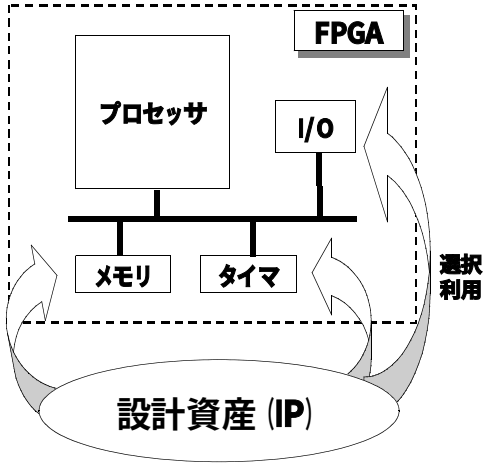

図３　IP を用いた設計の効率化

３．制御システムの動作検証

3.1 プロセッサとその周辺回路の１チップ化

図１のように，設計開発した周辺回路のうち，メモリ（ROM，RAM），I/O を選択し，プロセッサとあわせて制御システムを構成した。開発環境上で波形シミュレーションを行った後，FPGA に書き込み，実機で動作検証を行った。ROM にはプログラムが書き込んであり，この内容にしたがって制御システムが動作する。RAM は，プロセッサで処理したデータを一時的に保存するとき（サブルーチンのコール等）に用いる。アドレスバスとデータバスのビット幅は 16 ビットである。I/O は，LED などの外部機器とプロセッサのデータの入出力を仲介する。

本検証では，出力インターフェイスとして利用した。また，I/O の扱うデータのビット幅は 16 ビットにした。結果を図４に示す。

この図は，プロセッサの汎用レジスタに格納している値を I/O を通じて外部出力する様子である。A はアドレスデータ，D_IN はプロセッサへの入力データ，D はプロセッサからの出力データである。メモリの 001A 番地から B40F という命令コード（汎用レジスタの内容を，指定したアドレスへ書き込む命令）を読み出し，続いて 001B 番地から 8000 という I/O アドレスを読み出している。そして，8000 番地の I/O アドレスに対して 0055 という汎用レジスタに格納していた値を書き込み，I/O を通じて 0055 という値を FPGA 外部に出力している。これらから，制御回路の結線データを FPGA に書き込んだ後も正しく動作したことが分かった。

図４　制御システムの動作検証結果

3.2 スケーラブル化

RAM をスケーラブルにした設計例を図５，図６に示す。データバス（din，dout）のビット幅（K）を指定する記述は，RAM の HDL ソースとは別途作成した HDL ソース（パッケージファイル）において行った。これは，RAM の他に I/O，タイマカウンタ，シリアル/パラレル変換回路などのデータバスのビット幅も，そのパッケージファイルにて K のような変数を用意しておき，一括して管理するためである。ビット幅が８ビットの場合の，動作検証結果を図７に示す。

この結果から，0.75 ～ 1.25[μs]で８ビットデータ（00001111）$_2$ を RAM のアドレス（10）$_2$ 番地に書き込み，2.25 ～ 2.75[μs]で RAM のアドレス（10）$_2$ 番地から 8 ビットデータ（00001111)$_2$ を読み出していることが分かる。これらより，RAM のスケーラブル化した回路の動作が正しく行われていることが確かめられた。

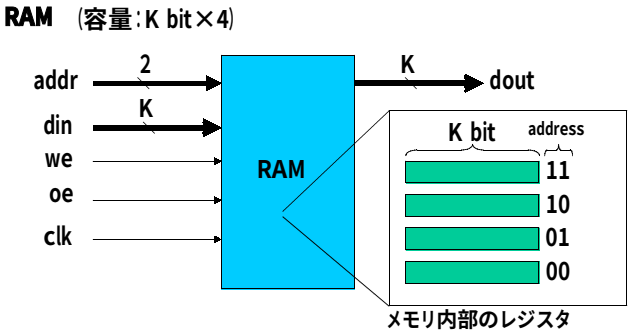

図５　スケーラブル化した RAM

```
library IEEE;
USE IEEE.std_logic_1164.all;

package mypack is
  constant K:integer:=  8;
end mypack;
```

パッケージファイル

RAMのHDLソースファイル

```
ic_1164.all;
USE WORK.mypack.all;
entity RAM is
port(
      ck,we,oe:in  std_logic;
      din:in  std_logic_vector(K-1 downto 0);
      addr:in  std_logic_vector(1 downto 0);
      dout:out std_logic_vector(K-1 downto 0));
end RAM;

architecture RTL of RAM is
signal en:std_logic_vector(3 downto 0);
```

図６　スケーラブル化した RAM の HDL ソースリスト

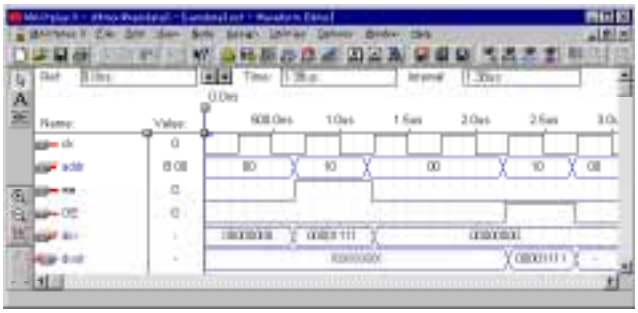

図７　スケーラブル化した RAM の動作検証結果

3.3 コンフィギュラブル化

プロセッサと各種 IP を接続するために，HDL 上の記述方法（コンポーネント・インスタンス文）を用いた。この記述方法は回路の構造化記述であり，基板上の IC のピンとピンを銅線で接続するときと類似している。使用したい IP があるときには，コンポーネント・インスタンス文の中に，必要な IP を入れる記述を行う。

図８のように，プロセッサ，メモリ，I/O をコンポーネント・インスタンス文を用いて記述した。

動作検証を行った結果，データバスのビット幅が８ビットと 16 ビットの場合について，スケーラブルかつコン

```
library ieee;
use ieee.std_logic_1164.all;
use work.mypack.all;

entity MICON is
port(
        CLK : in std_logic;
        Dout : out std_logic_vector(DATA_LENGTH-1 downto 0 );
        R_RESET : in std_logic);
end MICON;

architecture RTL of CNT10 is
signalWE_SIG,OE_SIG,WE_SIG_MEM,OE_SIG_MEM,WE_SIG_IO,OE_SIG_IO:std_logic;
signal  Din_SIG,Dout_SIG,A_SIG:std_logic_vector(DATA_LENGTH-1 downto 0);

component MEM
port(
            ck,we,oe:in  std_logic;
            din     :in  std_logic_vector(DATA_LENGTH-1 downto 0);
            addr    :in  std_logic_vector(ADR_LENGTH-1 downto 0);
            dout    :out std_logic_vector(DATA_LENGTH-1 downto 0));
end component;

component IO
port(
            we,oe   :in  std_logic;
            PA,PB.PC:inout  std_logic_vector(IO_DATA_LENGTH-1 downto 0);
            addr    :in  std_logic_vector(1 downto 0));
end component;
                                    :
compone U0:MPU port map (A=>A_SIG,CLK=>CLK,Din=>Din_SIG,Dout=>Dout_SIG,
  port                   R_RD=>OE_SIG,R_RESET=>R_RESET,R_WR=>WE_SIG);
        U1:MEM port map (ck=>CLK,we=>WE_SIG_MEM,oe=>OE_SIG_MEM,din=>Din_SIG,
                         addr=>A_SIG,dout=>Dout_SIG);
        U2:IO  port map (we=>WE_SIG_IO,oe=>OE_SIG_IO,PA=>Dout,addr=>A_SIG);
                                    :
end RTL;
```

図８　コンポーネント・インスタンス文

フィギュラブルにする前の制御システムと同様の結果が，図４のように得られた。

このことは，周辺回路を IP として蓄積し，後々必要があるときに再利用することの有効性を示している。

3.4 既存マイコンとの比較結果

多くの製品に使用されている組込みマイコンの SH4，PIC などと比較検討した。図９に示すように，大量生産向けのこれらのマイコンは，機能の面で二極分化してしまっているが，マイコン市場においては代表的なものである。中小企業では，現在でも多品種少量生産のため，両極の中央部分に該当する機能をもったマイコンを必要としている。これを踏まえた上で，開発する制御システムをスケーラブルかつコンフィギュラブルにし，その処理能力や機能を任意に構成できるようにした。

表１から，命令数や，プロセッサの周辺回路の種類，動作周波数，電源電圧で，SH4 と PIC の間に位置し，多品種少量生産向けの製品に適したものが開発できたことが分かる。

表１　既存制御システムとの比較

	PIC（PIC16Cxx）	本成果	SH4（SH7750）
ビット数	8	8,16,32	32,64
命令数	35	57	234
内蔵周辺モジュール	メモリ、A/D、シリアルI/O、タイマ	メモリ、I/O、タイマ、シリアル/パラレル変換回路、その他：IP（設計資産）	メモリ、A/D、DMAコントローラ、タイマ、シリアルI/O、パラレルI/O
動作周波数	～20MHz	～20MHz	～200MHz
電源電圧	2.0～6.0V	1.8、2.5、3.3、5.0V	1.8、3.3V

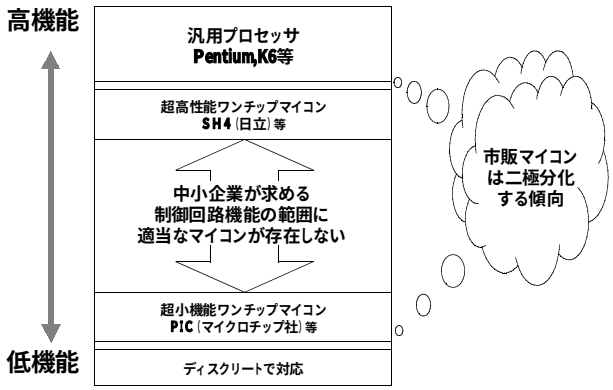

図９　二極分化するマイコン市場

４．まとめ

「HDL の制御システムへの応用」というテーマの中で，プロセッサとその周辺回路から構成され，スケーラブルかつコンフィギュラブルな機能をもった制御システムを開発した。スケーラブルとは制御システム内部のデータバスのビット幅を変更することによって，制御システムの処理能力を決めることであり，コンフィギュラブルとは制御システムを構成する回路を任意に組合わせて制御システムの機能を決めることである。これらは，VHDL の構造化記述や変数を設定する記述を工夫し，更に論理構成を任意に構成できる FPGA の特徴を活かすことで実現できた。

現在，電子機器分野の市場サイクルは早い。したがって，製品開発を短期間で行い，市場へいち早く投入することはこの分野の課題である。既存の IC の入手状況を気にすること無く，様々な製品に合わせて，柔軟に変更ができる本稿の制御システムは，多品種少量生産向けの製品を広く手掛ける中小企業に対して有効である。

参考文献

1）パターソン＆ヘネシー：コンピュータの構成と設計，上巻，下巻，日経 BP 社（1996）.
2）深山正幸ほか：HDL による VLSI 設計，共立出版（1999）.

（原稿受付　平成 13 年 8 月 1 日）