FPGA搭載向けソフトプロセッサコアの開発

大村　　功，長内　　研

Implementation of a Soft Processor Core for Tiny Embedded Systems

Isao OHMURA，Ken OSANAI

抄　録

FPGA(Field Programmable Gate Array)の大容量化により，プロセッサ・コアの利用が広がっている。プロセッサ・コアを利用することで，周辺回路と合わせたSoC(System on a Chip)設計が可能となり，小型化だけでなく，ソフトウェア資産を活用した開発効率の向上などが期待できる。本研究では，比較的小規模な組込みシステムをターゲットとして，低コスト版のFPGAに搭載可能なプロセッサ・コアの設計開発を行った。開発したプロセッサ・コアは，市販マイコンであるＨ８/300H(㈱ルネサステクノロジ)と互換性のある命令セットを採用した一方で，FPGAの特徴を活かしたメモリ構成や命令の拡張を行い，アプリケーションに適したプロセッサの構成を可能とした。

キーワード：FPGA，SoC，プロセッサ・コア，組込みシステム

Abstract

Processor cores implemented on FPGAs (Field Programmable Gate Arrays) are getting to be used in embedded systems as SoC (System on a Chip) devices. Using processor cores on FPGA is effective in downsizing the system, and also reducing time to market for embedded systems. We developed the processor core for tiny embedded systems, that can be implemented to Low Cost FPGAs. The Processor Core has the compatible instruction sets to the microprocessor H８/300H made by Renesas Technology Corp., and we designed the architecture easy to customize for embedded systems.

KEY-WORDS : FPGA，SoC，processor core，embedded system

１．はじめに

組込み機器の開発においては，FPGA(Field Programmable Gate Array)を用いた小型化，高速化が一般的となりつつある。特に，近年のFPGAデバイスの大容量化により，プロセッサ・コアを搭載したSoC(System on a Chip)設計が広がっており，このためのプロセッサ・コアがFPGAメーカーなどから提供されている。

プロセッサ・コアを搭載することで，周辺機能を含めた回路の１チップ化(SoC)が可能となり，小型化だけでなく，ソフトウェア資産を活用した開発効率の向上などが期待できる。また，プロセッサ・コアそのものの機能についても追加や変更が可能であるため，新たな命令や処理機能を追加することができる。特にコストや基板サイズの条件が厳しい組込み機器では，アプリケーションに柔軟に対応できる機能が処理性能や開発効率を向上させる重要なポイントとなる。

しかし，商用的に利用できる多くのプロセッサ・コアでは，ライセンス契約により，ユーザによるコア内部の回路の修正を制限している。そのため，命令の追加や内部機能の拡張が困難であるなど，FPGA搭載向けプロセッサ・コアの特徴を十分に活用できないものとなっている。

本研究では，アプリケーションに応じた回路の修正を可能とするオープンなプロセッサ・コアの提供を目指して，

事業名：一般試験研究
課題名：組込み向け制御機器の設計開発手法に関する研究

FPGAに搭載可能なプロセッサ・コアの開発を行った。開発したプロセッサ・コアは，比較的小規模な組込みシステムをターゲットとしており，市販の16bitマイコンと互換性のある命令セットを採用した。FPGAの特徴を活かした構成にすると同時に，アプリケーションに応じた柔軟なカスタマイズを可能とし，回路規模の小さなFPGAを採用したシステムにおいても利用できる。

2．プロセッサ・コアの開発

2.1　プロセッサ・コアの概要

開発したプロセッサ・コアの仕様を表１に示す。

本プロセッサ・コアは，小規模な組込みシステム向けとして一般に用いられているマイコンH8/300H(㈱ルネサステクノロジ)と互換性のある命令セットを採用した。アドレスの扱いはアドバンスト・モードに対応した24bit幅で処理している。

内部回路は，H8/300Hとの互換性を保つためのレジスタ構成などを除き，独自の設計である。そのため，各命令の実行クロック数やメモリ・アクセス処理，命令の拡張方法など，ベースとしたマイコンとは仕様が異なっている。

設計開発はハードウェア記述言語VHDLにより行い，以下の10個のモジュールに分けて行った。図１にこれらのモジュールの構成を示す。

① プログラム・メモリ制御モジュール(PSM)
　プログラム用メモリの読み出しおよびプログラム・カウンタの更新制御を行う。

② 実行制御モジュール(SMM)
　PSMで読み出された命令コードをデコードし，実行ステージに対応した制御コードを出力する。

③ レジスタ制御モジュール(RFM)
　汎用レジスタの読み書き制御およびレジスタを対象とした演算処理を実行する。

④ データ・メモリ制御モジュール(DSM)
　FPGA内蔵のデータ・メモリの読み書き制御を行う。

⑤ 加減算処理モジュール(AOM)
　32bit加減算処理を行う。

⑥ 論理演算モジュール(LOM)
　32bit論理演算(AND，OR，EXOR処理)を行う。

⑦ 乗算処理モジュール(MOM)
　16bit乗算処理を行う。

⑧ 除算処理モジュール(DOM)
　32bit÷16bitの除算処理を行う。

⑨ データ・アドレス管理モジュール(DAMM)
　データ・アドレス上位をデコードして，データ・メモリ制御モジュール(DSM)および拡張モジュール向けのセレ

表１　プロセッサ・コアの仕様

命令セット	H8/300Hと互換(一部の命令を除く)
動作クロック	30MHz (Altera社Cyclone，EP1C3T144C8使用時)
命令実行サイクル数	頻出命令(2～4byte命令)を1～2クロックで処理
命令の拡張性	拡張命令による追加，削除可能
メモリ空間	プログラム(Max128kbyte)，データ(Max16Mbyte) ベクタ(256エントリ)の各メモリ空間に分離
周辺機能の拡張	データ・メモリ上にマッピング (セレクト信号生成機能あり)
割込機能	最大255本の割込に対応可

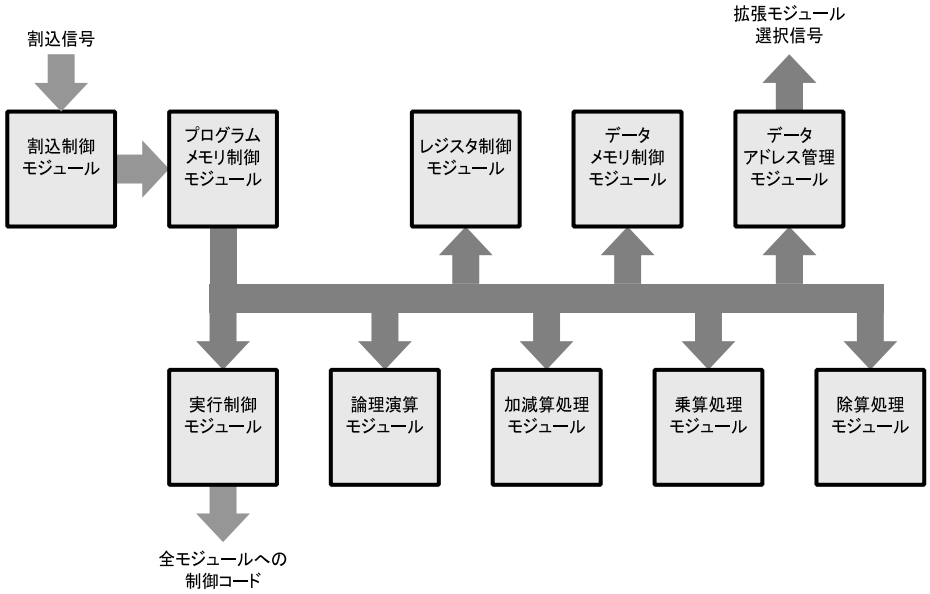

図１　モジュールの構成

クト信号を生成する。

⑩ 割込制御モジュール(IMM)

割込信号の受付処理とベクタ・テーブルの読み出しアドレス生成を行う。

入出力，カウンタ，タイマなど，アプリケーションに応じた周辺機能をプロセッサ・コアに付加するためには，拡張モジュールを追加する。拡張はデータ・メモリ上にマッピングする形で行い，FPGAの容量が許す限り複数の搭載が可能である。

なお，本プロセッサ・コアは，ソフト・プロセッサ・コアとして，ハードウェア記述での提供を想定している。したがって，開発時には，メモリ容量や拡張モジュールのマッピング情報など必要なパラメータを指定した上で，回路の合成を行い，FPGAへの書き込みが必要である。

2.2 メモリ空間

本プロセッサ・コアにおけるメモリ構成を図２に示す。

FPGA内部のリソースを有効に活用して処理性能の向上を図るため，本プロセッサ・コアでは，プログラム・メモリとデータ・メモリを完全に分離したハーバード・アーキテクチャを採用した。これら２つのメモリはプログラム・メモリ制御モジュール(PSM)とデータ・アドレス管理モジュール(DAMM)で独立に管理しており，平行してアクセス処理を行うことができる。

プログラム・メモリはFPGA内部のメモリを割り当てることとし，アドレス幅を16bitに制限することで，回路の簡素化とプログラム読み出しの高速化を図った。また，データ・メモリは，アドレス24bit，データ幅16bitとし，FPGA内部および外部のメモリを利用可能とした。

さらに，通常，メモリ空間に配置されるベクタ・テーブルについても分離し，割込や例外処理，特定の命令からの参照に限定して，応答性の改善を図った。参照可能な命令は，メモリ間接アドレッシングを行う無条件分岐とサブルーチン分岐とし，エントリ数を64から256(各16bit)へ拡張して，高速な分岐テーブルとしても利用可能とした。また，すべてのエントリを割込信号に対応させて，最大255本の割込を可能とした。

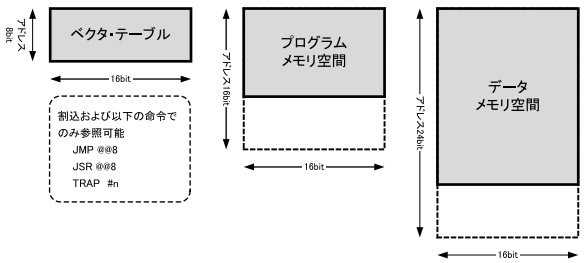

図２ 分離したメモリ空間

表２ H８/300Hとの命令互換性

	実装した命令	実装していない命令
データ転送命令	MOVの全命令	MOVTPE，MOVFPE
算術演算命令	加減算，乗算，除算，比較，符号反転，符号拡張の全命令	10進補正命令(DAA，DAS)
論理演算命令	全命令	なし
シフト命令	全命令	なし
ビット演算命令	BSET，BCLR，BNOT，BTST	Cフラグ処理系の命令
分岐命令	全命令 (メモリ間接アドレッシングは拡張)	なし
システム制御命令	全命令 (SLEEPは停止のみ，TRAPAは拡張)	なし
ブロック転送命令	なし	全命令(EEPMOV)

FPGA内部のRAMを使用した場合，１クロックでの高速なアクセスが実現できる。ただし，内部RAMのサイズは，利用するFPGAの容量に依存するため，アドレス幅(24bit)に対応した容量(最大16Mbyte)まで外部へのデータ・メモリの追加を可能とした。

外部データ・メモリや拡張モジュールなど，データ・メモリにマッピングされたモジュールのアクセス管理は，データ・アドレス管理モジュール(DAMM)で行っている。マッピング情報は，対応するアドレスの上位を予めパラメータ・ファイル中に定数テーブルとして記述しておくことで，アドレス・バス上の上位の値と一致チェックを行う回路が生成される。指定するアドレスのbit数は任意に決めることができるため，管理するメモリ領域の分割の自由度が大きく向上している。

拡張モジュールでは，データ・アドレス管理モジュール(DAMM)で生成されたセレクト信号を受けることで，デコード回路を用意する必要がない。また，処理待ちを制御するウェイト信号の生成機能があり，アクセス・スピードの遅いデバイスをFPGAの外部に拡張することができる。

2.3 命令セット

現在一般に使われているマイコンと互換性をもつことは，ソフトウェア資産が再利用できる点やコンパイラの新規開発を避けることができるという点で大きなメリットがある。また，組込み機器で多用されていることから，組込み向けプロセッサとして充実した基本命令やアドレッシング・モードを実装することができる。

一方，H8/300Hの命令セットはH8/300から拡張された部分も含むため，かなり複雑なものとなっている。命令数はアドレッシングも含め，のべ268あり，命令コード長は２byte～10byteと，多様な命令コードへの対応が要求される。

開発したコアでは，使用頻度が低いと思われる命令(キャリー・フラグを対象としたbit演算，BCD演算など)については実装していない。また，命令コード部分が２byteを超える命令については，命令コードの解析方法を変更し，一部の

			PSM		DSM				RFM							
			PsAddress	PsFetch	DsAddress	DsWData	DsRead	DsWrite	RfRnField	RfWData	RfWrite	RaProc	RaAddShift	RaNotLeft	RaThruAdd	RaLong
			000=Hold 001=RegB 010=PC+D 011=DS out 100=+2 101=DPR 110=Vector 111=Bcc	00=Hold 01=OPR&8 10=OPD16 11=OPD32	000=Hold 001=OPD 010=PS out 011=RegA 100=ADM 101= 110=+2 111=	000=RegB 001=RaProc 010=OPD 011=DS out 100=ADM 101=LOM 110=MULT 111=DIV	00=RegB 01=PC 10=CCR 11=LOM	0=Non 1=Read	0=Non 1=Write	00=2:AB 01=2:BA 10=1 L:A/B 11=ER7	000=RegB 001=RaProc 010=OPD 011=DS out 100=ADM 101=LOM 110=MULT 111=DIV	0=Non 1=Write	00=Shift 01=Add 10=Inv 11=ExtS	000=-1/0in 001=-2/LSB 010=-4/MSB 011= /Carry 100=+1 101=+2 110=+4 111=	0=Thru/Right 1=Inv/Left	0=Thru 1=Add	0=Normal 1=Long
1	4	BRANCH8	111	01	000	00	0	0	00	000	0	00	000	0	0	0
2			100	01	000	00	0	0	00	000	0	00	000	0	0	0
3	4	MOVB_A8_R	000	00	001	00	1	0	10	000	0	00	000	0	0	0
4			100	01	000	00	0	0	10	011	1	00	000	0	0	0
5	4	MOVB_R_A8	100	01	001	00	0	1	10	000	0	00	000	0	0	0
6			100	01	000	00	0	0	00	000	0	00	000	0	0	0
7	4	ADDB_I8_R	100	01	000	00	0	0	10	100	1	00	000	0	0	0
8	4	ADDXB_I8_R	100	01	000	00	0	0	10	100	1	00	000	0	0	0
9	4	CMPB_I8_R	100	01	000	00	0	0	10	000	0	00	000	0	0	0
10	4	SUBXB_I8_R	100	01	000	00	0	0	10	100	1	00	000	0	0	0
11	4	ORB_I8_R	[illegible]		[illegible]	00	0	0	10	101	1	00	000	0	0	0
12								0	10	101	1	00	000	0	0	0
									10	101	1	00	000	0	0	0
										010	1	00	000	0	0	

図 3　マイクロプログラムの内容

コードを分離して10種の拡張命令とした。

追加した拡張命令は，内部のレジスタを制御するほか，デコーダ内の３つの命令拡張フラグを操作するものとし，直後の命令はこの命令拡張フラグとの組み合わせでデコードする方法を採用した。この方法により，互換性を維持しつつ，拡張命令との組み合わせで新たな命令を追加できる命令セットとなった。

これらの変更により，本プロセッサ・コアでは200の命令により，一部を除いてH8/300Hとの命令コードの互換性を実現した。表２にベースとしたマイコンとの命令互換性を示す。

2.4　命令のデコードと制御コードの生成

プログラム・メモリ制御モジュール(PSM)で読み込まれた命令コードは，実行制御モジュール(SMM)でデコードされ，制御コードが生成される。

制御コード生成回路の設計は，水平型のマイクロ・プログラムを記述する方法を採用した。また，実装には制御メモリを使用せず，組み合わせ回路として論理合成した。

水平型マイクロ・プログラムでは，含まれているコードがそのまま各モジュールへの制御コードとなるため，設計開発やデバッグ時における修正が容易となる。また，メモリを使用しないことで，FPGA内部のRAMブロックのリソースを節約できる。

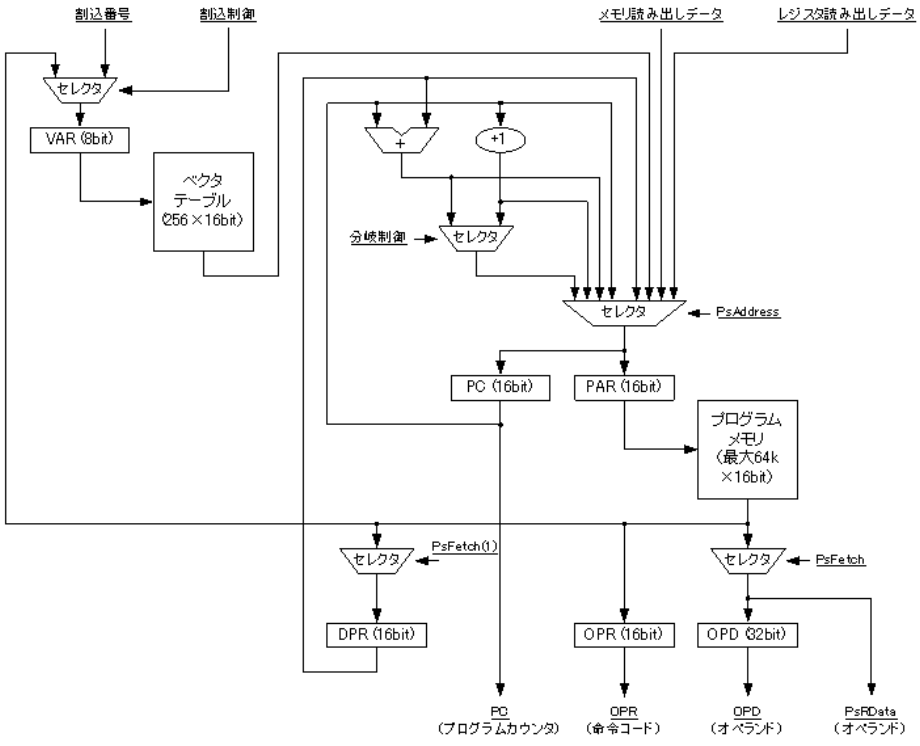

図 4　プログラム・メモリ制御モジュール(PSM)のブロック図

```
-- Decoder (ALL)
GEN_DECODE:
for I in 0 to N_OpcodeAll-1 generate
        Match(I) <= DecodeEx(OpcodeTable(I,0)) and Decode1H(OpcodeTable(I,1))
                        and Decode1L(OpcodeTable(I,2)) and Decode2H(OpcodeTable(I,3))
                        and Nstate(OpcodeTable(I,4));
end generate;

process(Match)
        variable tmp1 : std_logic_vector(SZ_CONTROL-1 downto 0);
        variable tmp2 : std_logic_vector(SZ_CONTROL-1 downto 0);
begin
        tmp2 := (others => '0');
        LOOP64:
        for I in 0 to N_OpcodeAll-1 loop
                if Match(I)='1' then
                        tmp1 := ControlList(conv_integer(OpcodeTable(I, 5)));
                else
                        tmp1 := (others => '0');
                end if;
                tmp2 := tmp2 or tmp1;
        end loop;
        ControlCode <= tmp2;
end process;
```

命令コードの一致チェック回路

制御コードの生成回路

図５　デコーダ生成のVHDL記述

マイクロ・プログラムの内容を表形式で表したものを図３に示す。また，図４にプログラム・メモリ制御モジュール(PSM)内部のブロック図とマイクロ・プログラムによる制御の方法を示す。図３では，最上段がモジュール名を，２段目が制御信号の名称を示している。図３におけるプログラム・メモリ制御モジュール(PSM)への制御信号PsAddressおよびPsFetchは，図４では，プログラムカウンタ(PC)およびオペランド・レジスタ(OPD)のセレクタ制御信号となっていることがわかる。本プロセッサ・コアではこのような制御の記述として，49bitで構成されたマイクロ・プログラムを用いて設計を行った。

マイクロ・プログラムに相当する設計データは，ハードウェア記述とは別に定数テーブルとしてファイル化した。このファイルを差し替えて回路を合成することで，命令セットや実装内容を変更して，プロセッサ・コアを合成することができる。

図５にデコーダ生成のためのVHDL記述を示す。最初の７行は命令コードと一致する命令番号を検索する部分で，ここで得られた一致信号Matchによって，８行目以降，該当するマイクロ・プログラムのコードControlCodeを出力している。マイクロ・プログラムのテーブル記述は300行以上になるが，VHDLの記述自体は20行程度とコンパクトな記述により実現した。

2.5　実行制御

実行は命令フェッチ(IF)，命令実行(EX)，メモリ・アクセス(MA)の３つからなるシングル・パイプラインを基本とし，命令デコードは命令実行(EX)のステージで同時に行う構成とした。パイプラインによる処理の様子を図６に示す。

頻出命令とされる２byte系の命令では，上記パイプラインにより１クロック毎での命令実行を可能とした。一方，４byte以上の命令では，プログラム・メモリのアクセス幅を２byteとしたため，２クロック以上に渡ってオペランドの読み出しが必要となるほか，４byteデータのメモリ・アクセスをともなう場合にはメモリ・アクセス(MA)のステージが連続するなど，実行内容に応じたステージ構成となる。

命令取り込み，実行，メモリ・アクセスを並列実行するパ

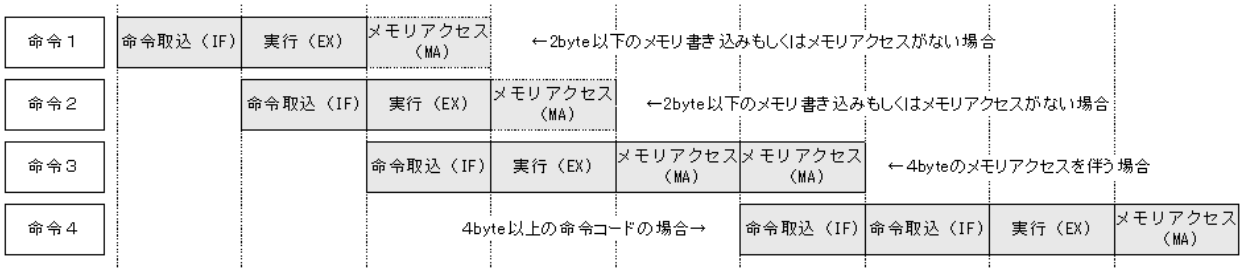

図６　パイプラインによる処理のようす

イプライン形式としたことで，ベースとしたマイコンに比較し，命令あたりの実行クロック数が１/２～１/３程度となり，同じ動作クロック周波数でも2.5倍程度の高速化を図ることができる。

メモリ読み出し命令では，レジスタへの書き込みがメモリ・アクセス(MA)ステージの後となるため，直後の命令で正しいデータを参照できないデータ・ハザードのおそれが生じる。これを回避するには，データのバイパス回路を設けるなどの方法があるが，本プロセッサ・コアでは回路規模を最小限に抑えるため，このような命令では無条件に次命令の実行を抑止する方法を採用した。

2.6　演算処理

演算処理を行うモジュールは，4つのモジュールに分割して設計開発を行った。

論理演算処理モジュール(LOM)，加減算処理モジュール(AOM)はともに32bitのデータを処理する回路とした。８bit，16bitでは，データ幅を識別し，拡張して処理を行っている。なお，プログラム・カウンタを対象としたアドレス計算(相対アドレス計算)については，プログラム・メモリ制御モジュール(PSM)内で演算する方法とした。

乗算処理モジュール(MOM)では，ビット・シフトと加減算を組み合わせた低速で回路規模を抑えた乗算回路(18クロックで実行)と，ハードウェア乗算器を利用した高速な乗算回路(３クロックで実行)を用意した。ハードウェア乗算器は，通常の論理回路用ブロックとは別に，乗算を高速処理するために設けられたもので，一部のFPGAに搭載されている。

除算処理モジュール(DOM)については，ハードウェアでの演算器が用意されていないため，ビット・シフトと加減算を組み合わせた除算回路として実装した。

2.7　カスタマイズ性

組込み向け機器では，小型化，低コスト化，高機能化など様々な要件のもとで，開発を進めなければならない。プロセッサ・コアにおいても，これらの要求に合うようなカスタマイズ機能を持つことで，広範囲に活用できる。

本プロセッサ・コアでは，ソフト・コアとしてハードウェア記述レベルで提供することを想定しており，ハードウェア記述レベルでの変更とパラメータによる変更，さらに拡張モジュールの追加の３つのレベルでのカスタマイズを可能としている。

ハードウェア記述レベルでの変更は，主に演算用モジュールの変更として利用できる。一般に演算回路では，処理速度と回路規模が相反する要素となるため，アプリケーションに応じた処理回路の選択はコスト低減のために重要である。市販のFPGA開発ツールでは，速度や回路規模の組み合わせでいくつかの演算処理IP(Intellectual Property)がライブラリとして利用できる。また，一部のFPGAでは上述したようにハードウェア演算器が搭載されている。これらの機能を利用したモジュールの再設計を行うことで，処理速度や回路規模に応じたモジュール入れ替えが可能となる。

一方，パラメータによる変更としては，命令の追加や変更，削除が可能である。命令の追加は，拡張命令と既存の命令を組み合わせることで，仕様上，数百種の命令を追加できる。これらの追加や変更はマイクロ・プログラムを書き換えることで対応できる。命令の削除についても対応するマイクロ・プログラムを削除することで，回路の削減ができる。

一般に，組込み向けシステムにおいては，設計完了後に新たなユーザ・プログラムの実行を想定する必要がない。したがって，使用するプロセッサ・コアにおいても，実装する命令をプログラム中で使用している命令に限定して回路の生成を行うことができる。このような最適化は，回路規模を削減して低コスト化に対応できるだけでなく，動作クロックの向上も期待できる。

また，拡張モジュールによる機能の追加は，前述したようにデータ・メモリへマッピングすることで可能となっている。特殊な演算処理などは拡張モジュールとして実装することで，命令セットの変更を行わずに追加できる。さらにFPGA上に拡張モジュールを搭載する場合には，プロセッサ・コアと共通のクロックを使用できるため，完全な同期回路として設計できる。これにより，タイミング設計が容易となるほか，インターフェースにおけるデータの受け渡しの高速化を図ることができる。

2.8　回路規模と処理性能

表３に，すべての命令を実装した場合と特定のアプリケーションに合わせて必要最低限の命令を実装した場合の本プロセッサ・コアにおけるFPGAの使用リソース量と動作可能クロック周波数を示す。ターゲットとしたFPGAはアルテラ社EP1C３T144C８で，Cycloneシリーズで最小規模のデバイスである。

アプリケーション例としては，マトリックス・キーからの入力情報から対応するキャラクタを液晶表示器に表示するプログラムとDhrystoneベンチマーク・プログラムの２つを対象とした。

表３から，フルセットの命令を実装した場合でも低価格版と呼ばれるFPGAに容易に搭載できることがわかる。また，実装する命令数を削減することで，回路規模が小さくなり，動作周波数が向上する。このようにアプリケーションに合わせて命令実装を行うことで，SoC設計に適したプロセッサ・コアの生成が可能である。

表３　実装命令数による使用リソース

	実装命令数	使用リソース	最大動作周波数
フルセット命令実装	200	2485 LE	31.8MHz
アプリケーション1（キー入力，液晶表示）	35	1896 LE	33.8MHz
アプリケーション2（Dhrystone）	59	1939 LE	34.7MHz

※FPGAは，アルテラ社Cyclone（EP1C3T144C8）を使用

３．FPGAボードへの実装

3.1　オリジナルFPGAボードの開発

プロセッサ・コアおよび周辺機能を動作させる環境として，２つのオリジナルのFPGAボードを試作開発した。

図７は，プロセッサ・コアなどのIP評価用として開発したFPGAボードで，マトリックス・キーや液晶表示，シリアル通信機能を搭載した。FPGAはアルテラ社のCycloneシリーズを搭載している。仕様を表４に示す。

また，FPGAは，画像や音声，高速通信などのアプリケーションで利用されることが多い。このようなアプリケーション向けとして，アルテラ社のCycloneⅡを搭載したボードの開発を行った。図８はこのボードの外観，表５は仕様である。ハードウェア乗算器を搭載したFPGAを採用することで，乗算を含む処理を高速に実行できる。

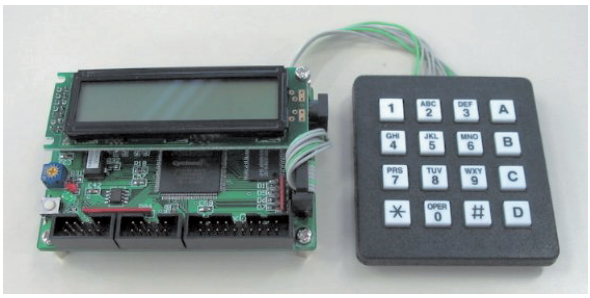

図７　プロセッサ・コア評価向けFPGAボード

表４　プロセッサ・コア評価向けFPGAボードの仕様

使用FPGA	アルテラ社Cyclone（EP1C3T144C8）
通信ポート	RS-232C（3pin），USB（ミニBコネクタ）
入力機能	タクトスイッチ1個（ボード上） 4×4マトリックスキー用コネクタ
出力機能	LED2個（ボード上） キャラクタ液晶表示器用コネクタ
搭載メモリ	SRAM　4Mbit（256x16bit）　アクセス12nsec
拡張コネクタ	16pin（8pin毎に入出力選択）
電源	+5V DC（ボード上で3.3Vおよび1.5V生成）

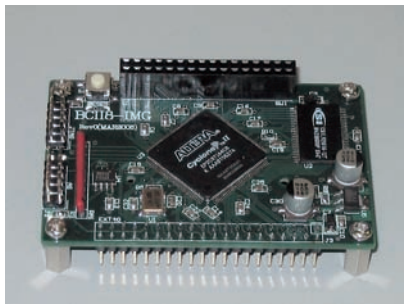

図８　画像処理向けFPGAボード

表５　画像処理向けFPGAボードの仕様

使用FPGA	アルテラ社CycloneII（EP2C8T144C6）
入力機能	CMOSイメージセンサ用コネクタ
搭載メモリ	SRAM　4Mbit（256x16bit）　アクセス12nsec
拡張コネクタ	20pin（FPGA直接接続）
電源	+3.3V DC（ボード上で1.2V生成）

3.2　プロセッサ・コアの性能評価

開発したプロセッサ・コアをFPGAボードに実装して，回路規模や処理速度について評価を行った。

表６はDhrystoneベンチマーク試験をＨ８マイコンのボードと上記ボードで動作させた場合の結果である。Ｈ８マイコン・ボードは㈱秋月電子通商のAKI-H8 マイコンLANボード(クロック25MHzに変更)を使用した。

FPGAに搭載したプロセッサ・コアは，動作クロック数の向上と命令あたりの実行クロック数の短縮により，処理速度が３倍以上に向上していることがわかる。なお，プロセッサ・コアはフル命令実装で，アルテラ社CycloneⅡシリーズのEP2C5T144C6 を用いた場合には，ハードウェア乗算器を利用した。

表６　Dhrystoneベンチマークによる性能比較

	動作クロック	処理速度（Dhrystone/sec）	
ルネサステクノロジ H8/3069F	25MHz	4315	1倍（基準）
プロセッサ・コア（EP1C3T144C8使用）	30MHz	13204	3.1倍
プロセッサ・コア（EP2C5T144C6使用）	36MHz	15901	3.7倍

※EP1C3T144C8はアルテラ社Cycloneシリーズ
EP2C5T144C6はアルテラ社同じくCycloneIIシリーズ

４．今後の課題

本研究では市販マイコンとの互換性を有しながら，機能の拡張やカスタマイズ性の向上を実現するため，メモリ空間の分離や命令の拡張を試みた。

しかし，追加した命令は，コンパイラなど開発環境が対応せず，容易に利用することができないという課題がある。また，メモリ空間を分離したことで，プログラム・メモリには定数テーブルを配置できないなど，ベースとしたマイコンのプログラムの構造を一部変えなければ実行できないという課題もある。

これらの課題の解決には，命令の追加や削除を柔軟にサポートできるコンパイラやリンカなどの開発環境の整備が不可欠であり，今後の開発ポイントとなっている。

５．まとめ

本研究では，市販マイコンと互換性を持ったソフト・プロセッサ・コアの開発をおこなった。処理速度の向上を図る一方で，アプリケーションに応じたカスタマイズを可能としており，組込み向けプロセッサ・コアとして，広い範囲での活用を可能とした。

プロセッサ・コアは，FPGAの利用拡大とともにその利用が広がりつつある。書き換え可能なハードウェアであるFPGAとカスタマイズ可能なソフト・プロセッサ・コアを活用することで，高速化，小型化，機能の向上など，付加価値の高いシステム開発と製品の保守性向上が可能となる。

今後は，開発環境の整備を進るとともに，組込みシステムの実用化開発で活用していく予定である。

引用文献

1) Ｈ８/300Hシリーズ　プログラミングマニュアル，㈱ルネサステクノロジ（1999）
2) CycloneⅡ Device Hand Book, ALTERA（2005）