社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2003－ＳＬＤＭ－111　(9)
2003／10／23

FPGAによるDSP向け多機能メモリの実現

橋本耕太郎[†]　林　悠平[†]　田中康一郎[††]　佐藤　寿倫[†,††]

† 九州工業大学 情報工学部 知能情報工学科
†† 九州工業大学 マイクロ化総合技術センター
〒820–8502 福岡県飯塚市川津 680–4
E-mail: †{hashi,yuhei,tsato}@mickey.ai.kyutech.ac.jp, ††tanaka@cms.kyutech.ac.jp

あらまし　高速処理が必要なDSPシステムでは，一般的に複数のDSPや専用ICを利用している．しかし，そのようなシステムは単一DSPと比較して，ハードウェア構成だけでなく大幅なプログラムの変更やハードウェア/ソフトウェアトレードオフなども行う必要がある．これに対して，我々はDSPとメモリとの間に再構成デバイスであるFPGAを挿入する構成(多機能メモリ)が有効であると考える．FPGAにより処理の一部をメモリアクセス時に行うことでDSP全体の処理能力向上が期待できる．本稿では，多機能メモリの概要とFPGA/DSPを搭載したRYUOHによる多機能メモリの設計事例ついて報告する．また，ハードウェア設計の経験がないDSPプログラマにも対応するためにシステムレベル設計言語を用いた設計についても言及する．多機能メモリを用いてJPEGエンコーダを実装した結果，従来は適用が困難であったDMA転送が可能となりDSPの処理性能が約33%向上した．
キーワード　FPGA，DSP，ハードウェア/ソフトウェア協調設計，リコンフィギャラブルシステム

Design of Multi-Functional Memory Based on FPGAs for DSPs

Kohtaro HASHIMOTO[†], Yuhei HAYASHI[†], Koichiro TANAKA[††], and Toshinori SATO[†,††]

† Department of Artifical Intelligence, Kyushu Institute of Technology
†† Center for Microelectronic Systems, Kyushu Institute of Technology
Kawazu 680–4, Iizuka, Fukuoka, 820–8502, Japan
E-mail: †{hashi,yuhei,tsato}@mickey.ai.kyutech.ac.jp, ††tanaka@cms.kyutech.ac.jp

Abstract　This paper presents novel DSP systems containing a multi-functional memory that consist of both a reconfigurable device and a general-purpose memory. In general, DSP system architectures, which can support complex processing, are multi-processor based DSP systems and DSP systems with hardware acceleration. The novel DSP systems, which connect FPGA between a DSP and a memory for the DSP, can accelerate the complex processing that can not show performance sufficient in those architectures. The novel DSP system is evaluated on an original hardware platform called RYUOH that consists of an FPGA, a DSP, and a dual-inline memory module. We confirmed that the performance of a JPEG encoder with multi-functional memory improved about 33% rather than it without the memory.
Key words　FPGA, DSP, Hardware/software co-design, Reconfigurable system

1. はじめに

集積回路技術の発展により，DSP (Digital Signal Processor) の性能は飛躍的に向上してきた．その結果，様々な情報機器においてDSPシステムが一般的に使用されている．その中には，画像処理など高い処理能力を要求するシステムが多数存在する．そのようなシステムでは，処理性能の問題から単一DSPによる実現が困難な場合がある．そこで，システム全体の処理を複数のDSPに分散する方法や，ASICや汎用ICなどのハードウェアに処理の一部を負担させる方法により処理能力を向上することで対応してきた．

上記の構成における処理の分割は比較的粗粒度であることから，単一DSPと比較してハードウェア構成だけではなく，プログラムの大幅な変更やハードウェアとソフトウェアのトレードオフなどを行う必要があるため，容易に性能向上を行うことは難しい．

そこで，DSP とメモリとの間に専用のロジックを挿入する構成を検討した [1]．この専用ロジックとメモリを合せて多機能メモリと称する．多機能メモリによりメモリアクセス時に処理を行うことで，DSP 内に新たな命令を追加した場合と同等の効果を得ることが可能となる．また，主要な処理におけるアドレス計算とデータ処理のような細粒度な処理の分割を実現することができるため，効果的な性能向上が期待できる．この際，専用ロジックに再構成デバイスである FPGA（Field Programmable Gate Array）を用いることで，システムの特性を生かした柔軟なシステムが実現できる．また，一般的な DSP プログラマには HDL（Hardware Description Language）による FPGA 設計は難しいと考えられる．そこで，システムレベル設計言語の一つである SpecC 言語を用いた多機能メモリの設計についても検討した．

本稿では，多機能メモリの概要と SpecC 言語を用いた設計について説明し，その性能を確認するための設計事例について紹介する．次章では，一般的な DSP システムの構成について利点と欠点を基に検討し，3 章で本稿にて提案する多機能メモリについて述べる．4 章では SpecC 言語を用いた多機能メモリの設計方法を説明し，5 章で設計事例として，ハードウェアプラットホームである RYUOH を用いた多機能メモリの設計と実装結果について述べる．

2. 一般的な DSP システムの構成

画像処理などの複雑で高い処理能力を要求する DSP システムでは，一般的に図 1 に示すような，(a) 複数の DSP で処理を分散させる構成や，(b) ASIC などのハードウェアにより処理の一部を負担させる構成により対応してきた．

(a) では，DSP の台数に応じた高速化が図れることから，大規模なシステム構成が比較的容易である．しかしながら，メモリなどの共有資源がボトルネックとなるほか，DSP 間の同期が必要となるためデータ通信の回数が増加する．また，DSP のプログラムを行う際に通信のスケジューリングを行う必要があるため設計が複雑になる．一方 (b) では，ハードウェアを用いることにより，DSP を追加した場合と比較して高速処理が可能となる．しかしながら，ソフトウェアとハードウェア双方の設計を行うことから，それぞれ異なる言語と設計ツールを用いた設計検証となる．そのため，トレードオフや協調動作の検証が困難となりソフトウェアだけのシステムと比較して開発期間が長期化する．また，ハードウェアの機能が固定されることでシステム全体の柔軟性が低下する．

これらの構成における処理の分割は比較的粗粒度であることから，単一 DSP と比較してハードウェア構成だけではなく，プログラムの大幅な変更やハードウェアとソフトウェアのトレードオフなどを行う必要があるため，容易な性能向上が難しい．そこで，DSP とメモリとの間に専用ロジックを挿入する (c) の構成が考えられる．メモリと専用ロジックを一つの多機能メモリと扱うことで，DSP からはメモリアクセスにより，専用ロジックによるデータ処理が可能となる．次章で多機能メモリの概要を述べる．

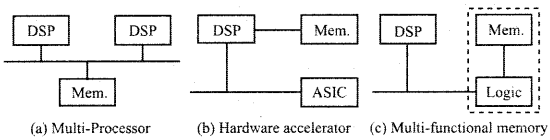

図 1　システム構成

3. 多機能メモリ

3.1 多機能メモリの構成

多機能メモリを専用の IC とメモリにより実現した場合，機能が固定されることによりシステム全体の柔軟性が低下する．そこで，専用ロジックには再構成デバイスである FPGA の利用が効果的であると考えられる．FPGA は集積回路技術の発展によりその性能が飛躍的に向上したことで，メモリなどの高速なデバイスとの直接接続が可能となった．FPGA を利用することで，システムの特性を生かした柔軟なシステムが構築できる．

FPGA の内部ロジックは，図 2 に示すように，DSP との通信を調停するための DSP インタフェース，メモリを制御するためのメモリコントローラ，多機能メモリの機能を実現するための機能ロジックで構成する．また，多機能メモリでは複数の機能を実現し，その機能は DSP プログラマが自由に選択できることが望ましい．そこで，DSP のメモリ領域に対して多機能メモリをマッピングする方法が考えられる．これにより，DSP プログラマは利用したい機能に対応したメモリ領域にアクセスすることで様々な機能を利用することが可能となる．

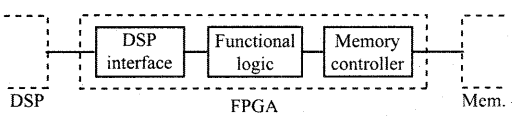

図 2　多機能メモリの構成

3.2 機能ロジックの構成と設計方法

機能ロジックに複数の機能を実現する方法として，図 3 に示すように，(a) 複数の機能モジュールをロジック内に実装する構成と，(b) 動的部分再構成 [2] により機能モジュールを入れ替える構成の 2 通りの構成が考えられる．

(a) では，実装された機能を切り替えるための機構を用意することで，複数の機能を同時に使用することが可能となる．しかし，FPGA には回路規模の制限があるため，大規模な処理や多くの機能を実装することは難しい．一方 (b) では，あらかじめ複数の機能モジュールを作成しておき，システムの動作中に必要となった段階で機能モジュールの再構成を行う．このため，回路規模を抑えることが可能となり大規模なシステムに対しても多機能メモリが利用できる．

複雑な DSP システムでは，多機能メモリを利用することで性能向上が実現できる処理が多数存在すると考えられる．それらの処理全てを FPGA に実装する場合，回路規模の制限が問題となると予想されるため (b) の構成が有効だと考えられる．

一方，多機能メモリを用いた DSP システムは，2 章の (b)

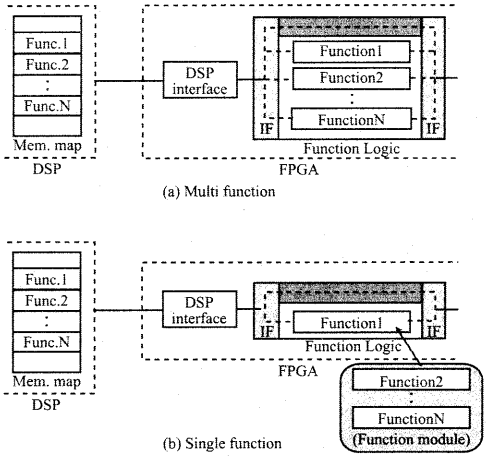

図 3　機能ロジックの構成

と同様にハードウェア/ソフトウェア混在システムであるため設計期間の長期化が予想される．そこで，システムレベル設計言語の一つである SpecC 言語を用いた設計が有効であると考えられる．次章で SpecC 言語を用いた多機能メモリの設計方法について説明する．

4. SpecC 言語を用いた多機能メモリの設計

4.1 システムレベル設計

ハードウェア/ソフトウェア混在システムの設計にはそれぞれ異なる言語と設計ツールが必要なことから，協調動作の検証やハードウェア/ソフトウェアトレードオフが困難となり設計期間が長期化する．この問題を解決する方法として，C/C++言語や HDL を拡張した SpecC [3] や SystemC [4]，SystemVerilog [5] などのシステムレベル設計言語が提案されている．また，信号処理の分野で用いられてきたアルゴリズム開発のためのシミュレーション環境である The MathWorks 社の MATLAB/Simulink を利用した，ブロックライブラリによるシステムレベル設計なども検討されている [6] [7]．システムレベル設計により，同一の環境においてソフトウェアとハードウェアを意識することなく協調設計および検証を行うことが可能となり，設計期間の短期化が図れる．

4.2 SpecC

SpecC は ANSI-C をハードウェア記述が可能となるように拡張した言語である [8] [9]．図 4 に示すように，SpecC プログラムモデルは，ビヘイビア（動作），チャネル，およびインタフェースの 3 種類で構成される．ビヘイビアは仕様に沿った動作を記述する部分であり，階層化が可能である．チャネルは通信をカプセル化しており，この中で通信のプロトコルを定義する．インタフェースはビヘイビアとチャネル間の柔軟なリンクの役目を果たしている．このように 3 つのコンポーネントでシステムを構成することにより，動作と通信を分離した記述が可能となる．

4.3 SpecC 言語を用いた設計方法

我々はこれまでに，SpecC 言語を用いたハードウェア/ソフトウェア混在システムの設計として，VisualSpec と eXCite の 2 つのツールを用いた設計方法を検討してきた [15] [16]．Interdesign Technologies 社の VisualSpec [10] は，SpecC 言語をサポートした統合設計ツールであり，ANSI-C から拡張されている多くの部分をグラフィカルに表現することで，C 言語を用いたプログラムによるシステム設計を可能としている．また，Y Explorations 社の eXCite [11] は C 言語ベースの高位合成ツールであり，ANSI-C から直接ハードウェアを生成することができる．

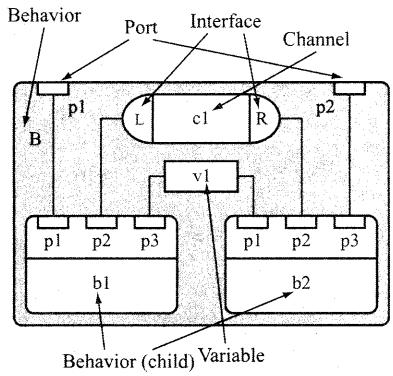

図 4　SpecC モデル

これまでに SpecC 言語を用いた協調設計を検討してきた結果，多機能メモリの設計方法として，DSP と多機能メモリを協調設計するための VisualSpec テンプレートを利用することが効果的であると考えられる．

DSP インタフェースとメモリコントローラはシステムに合せて固定することが可能であるため，これらを VisualSpec 上のチャネルとインタフェースで定義する．また，DSP の動作を記述する DSP ビヘイビアと，多機能メモリの機能を記述する機能ロジック・ビヘイビアを用意し，DSP ビヘイビアからは直接のメモリアクセスと機能ロジックを経由したメモリアクセスを可能とすることで，図 5 に示すような DSP と多機能メモリの協調設計環境が実現可能となる．設計者は，DSP ビヘイビアと機能ロジック・ビヘイビアに動作を記述をするだけで，DSP と多機能メモリの協調設計が可能となる．また，同一言語による設計であるため，DSP ビヘイビアに記述したプログラムの一部を機能ロジック・ビヘイビアに単純に移行することでトレードオフが行える．

5. 設計事例

5.1 実装環境:RYUOH

本稿の実装対象として，図 6 に示す本学で開発した DSP と FPGA および大規模メモリを搭載したハードウェアプラットホームである RYUOH [12] を利用する．RYUOH には図 7 に示すように，Texas Instruments 社の高性能 DSP である TMS320C6701（‘C6701）[13]，Xilinx 社の大規模 FPGA である Virtex-E XCV1000E [14] が実装されており，それらは DSP の EMIF（External Memory Interface）と HPI（Host Port

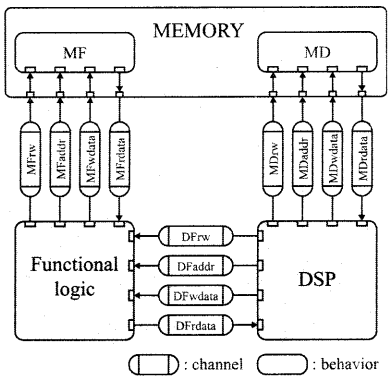

図 5　VisualSpec による多機能メモリ用テンプレート

Interface）で接続されている．

各デバイスの制御やデバイス間の通信を行うためのロジックは，図 8 に示すような構成として FPGA に実装されており，それらはライブラリとして使用できる．なお，全てのライブラリが相互に直接接続されている．ライブラリを用いることで，DSP と FPGA の外部メモリとして FPGA に接続されている DIMM を利用することが可能となる．また，‘C6701 に実装されている EMIF は 4 つの CE（CE0-3）を持ち，各 CE に異なるメモリを接続することが可能である．そのため，DSP 内部のメモリマップには各 CE に対応した領域が用意されており，その領域にアクセスすることで割当てられたメモリにアクセスすることができる．RYUOH ライブラリでは，CE0 に SBSRAM，CE2 に DIMM が接続されている．設計環境として DSP には Texas Instruments 社の Code Composer Studio 2.2，FPGA には Xilinx 社の ISE 5.2i を使用した．

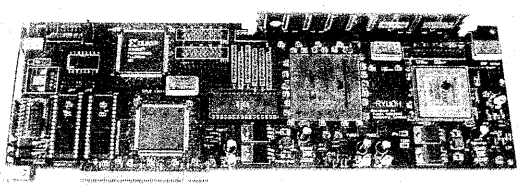

図 6　RYUOH

5.2　アプリケーション

FPGA による多機能メモリの有効性を示すため，JPEG エンコーダ [17] を DSP と FPGA により設計した．JPEG エンコーダには 4 つの処理方式が存在し，今回はその中で最も利用頻度の高い基本 DCT 方式を設計した．その処理は，図 9 に示すように，色空間変換，2 次元 DCT，量子化，ハフマン符号化の 4 つの処理から構成されている．各処理は MCU（Minimum Coding Unit）と呼ばれる 8 × 8 画素のブロックの整数倍を 1 単位として処理する．今回は 320 × 240 の RGB 画像を用い，8 × 8 画素のブロック一つを MCU とした．

一般的に，画像データは図 10 に示すような配置でメモリに

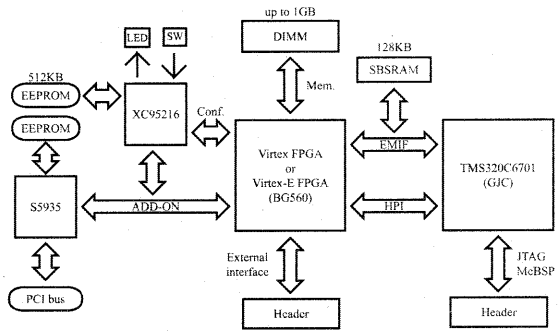

図 7　RYUOH の構成

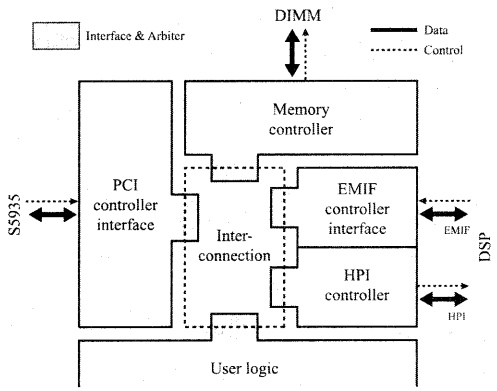

図 8　RYUOH ライブラリ（FPGA）

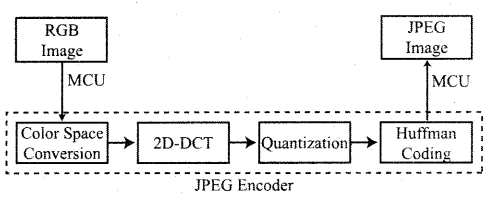

図 9　JPEG エンコーダ（基本 DCT 方式）

格納されている．そのため，JPEG エンコードの主要な処理以外に MCU データを取り出すためのアドレス計算が必要となる．その結果，DSP からは連続したメモリアクセスが行えず，DSP の一般的な機能である DMA（Direct Memory Access）の適用には考慮が必要であった．しかし，多機能メモリでアドレス計算を行うことで，DSP では連続したメモリアクセスにより MCU データの取得が可能となることから，DMA の適用が容易となる．そのため，処理性能の飛躍的な向上が期待できる．そこで，MCU データ取得のためのアドレス計算を多機能メモリで行うこととした．

5.3　設計の詳細

RGB 画像は図 11 に示すように，RYUOH 上の DIMM に 4 バイト毎に RGB それぞれ 8 ビットのデータが，画像の横軸に対して順番に格納されているものとする．

5.1 節で述べたように，RYUOH は DSP と DIMM の間に FPGA が挿入された構成であることから，FPGA と DIMM を

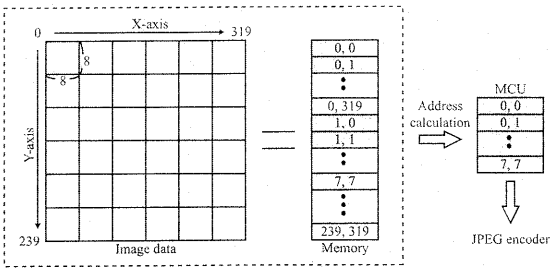

図 10　MCU データの取得

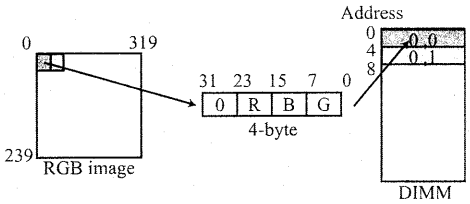

図 11　画像データの配置

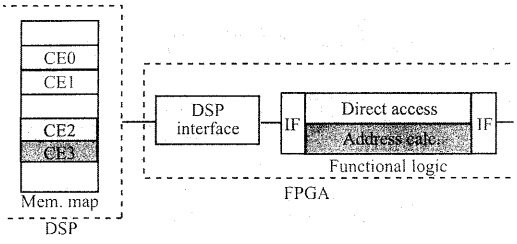

図 12　多機能メモリの設計

```
OFFSET = 0x02000000;  // CE2
for (y=0; y<240; y+=8)      // height
  for (x=0; x<320; x+=8)    // width
    for (mcu=0; mcu<64; mcu++) {  // mcu
      addr=(mcu&7)+((mcu>>3)+y)*320+x;
      intmem[k] = extmem[addr+OFFSET];
      k++;
    }
```

図 13　データ転送プログラム

```
OFFSET = 0x03000000;  // CE3
for (y=0; y<240; y+=8)      // height
  for (x=0; x<320; x+=8)    // width
    for (mcu=0; mcu<64; mcu++) {  // mcu
      intmem[k] = extmem[k+OFFSET];
      k++;
    }
```

図 14　変更されたデータ転送プログラム

```
ADDR = mcu[2:0]+(mcu[7:3]+y)*320+x;
mcu = mcu+1;
if (mcu == 64) begin      // mcu
  mcu = 0; x = x + 8;
  if (x == 320) begin     // width
    x = 0; y = y + 8;
    if (y == 240) begin   // height
      y = 0;
    end
  end
end
```

図 15　アドレス計算のための HDL コード

一つの多機能メモリと見なすことが可能である．また，FPGA にはメモリコントローラと EMIF コントローラ・インタフェースがライブラリとして用意されているため，EMIF コントローラ・インタフェースを DSP インタフェースと見なし，新たに機能ロジックを追加することで多機能メモリが実現できる．

多機能メモリで計算されたアドレスに対して DIMM アクセスを行うため，図 12 に示すように DSP の CE2 と CE3 を DIMM に接続し，CE2 領域へのアクセスは通常の DIMM アクセス，CE3 領域へのアクセスは機能ロジック内で計算されたアドレスに対する DIMM アクセスとなるよう設計した．これにより，DSP からは CE2 と CE3 はそれぞれ独立したメモリとして扱うことが可能となる．

設計手順として，最初に DSP でアドレス計算を含む JPEG エンコーダを作成して動作検証を行った．DIMM から DSP の内部メモリに RGB データを転送する DSP プログラムを図 13 に示す．次に，図 13 のプログラムに対して多機能メモリでアドレス計算を行うように，メモリアクセスの領域を CE2 から CE3 に変更し，アドレス計算部分を機能ロジックに移行した．修正した DSP プログラムを図 14 に示す．また，機能ロジックに実装するために移行したアドレス計算部分を Verilog HDL により設計した（図 15）．同様に，SpecC 言語を用いたアドレス計算プログラムからアドレス計算回路を生成した．

5.4 実装結果

MCU データを取得するためのアドレス計算を行うロジックを多機能メモリに実装し，320 × 240 の RGB 画像に対して JPEG エンコードを行った．

まず，MCU データ一つをエンコードする際のサイクル数をシミュレーションで測定した結果を図 16 に示す．比較のため，DSP でアドレスを計算した場合のサイクル数も記載する．図 16 に示すように，多機能メモリによりアドレス計算を行った場合，MCU データ一つのエンコードに対して必要なサイクルが約 33%減少したことから，細粒度な処理の分割による僅かな DSP プログラムの変更で速度向上が可能なことを確認した．

次に，JPEG エンコードにおいて多機能メモリを用いて DMA を適用した場合の，処理速度の実測結果を図 17 に示す．比較のため，全ての処理を DSP で行った場合と多機能メモリを用いて DMA を適用しない場合の結果も記載する．図 17 に示すように，多機能メモリにより DMA が適用可能となることで，約 33%と著しい速度向上が見られたことから，多機能メモリによる DMA の有効利用が可能なことが確認できた．また，DMA を適用しない場合でも約 10%の速度向上となった．

一方，SpecC 言語を用いてアドレス計算を行う機能ロジック

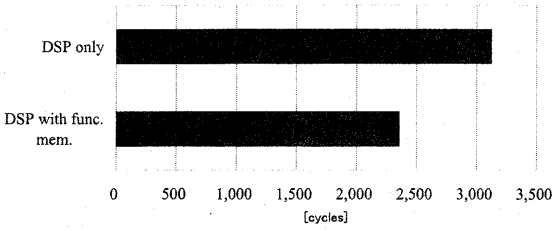

図 16　MCU データ一つをエンコードするために必要なサイクル数

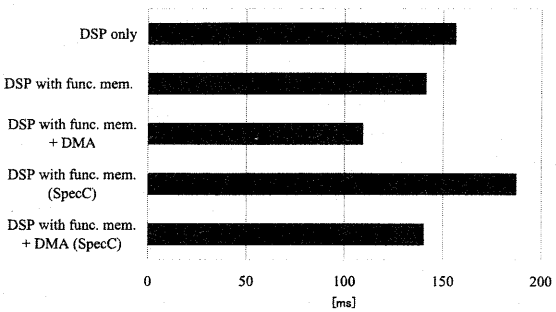

図 17　JPEG エンコードの処理時間（画像サイズ:320 × 240）

を設計した場合，Verilog HDL を用いた設計と比較して処理性能が低下した．この理由として，VisualSpec/eXCite が生成する HDL コードは全てステートマシンを構成することから，レイテンシが増加したことが原因と考えられる．しかし，SpecC 言語による設計は DSP プログラマが多機能メモリの設計を行う上で重要であり，ここでは性能は考慮しないこととする．多機能メモリと DSP を同じコードで記述できることから，トレードオフが容易であることが確認できた．

最後に，表 1 に JPEG エンコーダを実装した際の FPGA の回路規模を示す．多機能メモリを実現する際に必要となる DSP インタフェースとメモリコントローラ，機能ロジックを合計した回路規模は FPGA 全体の 9%であり，そのうち機能ロジックは 1%となった．このことから，アドレス計算だけでなく，大規模な機能を多機能メモリで実現可能であることが確認できた．なお，SpecC 言語で設計した機能ロジックの回路規模は約 3%であった．

表 1　FPGA の回路規模（XCV1000E）

	Slices(%)
All RYUOH library	1,608(13%)
(1)DSP interface	411(3%)
(2)Memory controller	592(4%)
(3)Functional logic	178(1%)
(SpecC)	386(3%)
(1)+(2)+(3)	1,181(9%)

6. おわりに

本稿では，DSP システム向け多機能メモリについて，構成および SpecC 言語を用いた設計を述べた．JPEG エンコード処理において，アドレス計算を多機能メモリで実現した結果，DSP 単体の処理と比較して処理性能が向上した．これにより，多機能メモリを用いた細粒度の処理分割が可能であり，DSP プログラムの僅かな変更で性能向上が可能なことを確認した．また，多機能メモリによるアドレス計算と DMA を合せることで効果的な速度向上となるなど，多機能メモリによる DMA の有効利用が期待できる．

SpecC 言語を用いた設計は協調設計が可能となり，トレードオフも容易であるため，今後は SpecC 言語を用いた効果的な設計について検討を行う必要がある．

文　献

[1] 橋本耕太郎，林悠平，田中康一郎，佐藤寿倫，有田五次郎，“多機能メモリを用いた DSP システムとその開発支援環境，” 第 5 回 DSPS 教育者会議予稿集，pp.85-90，Sep，2003.
[2] 林田幸英，林悠平，田中康一郎，佐藤寿倫，“動的部分再構成型 FPGA によるシステム設計事例，” 第 1 回リコンフィギャラブルシステム研究会論文集，pp.227-234，Sep，2003.
[3] SpecC Technology Open Consortium, *http://www.specc.org/*.
[4] Open SystemC Initiative, SYSTEMC Version 2.0 User's Guide, 2002.
[5] Accellera Organization, Inc., SystemVerilog 3.0 Accellera's Extensions to Verilog, 2002.
[6] 橋本耕太郎，櫻木誠，田中康一郎，佐藤寿倫，有田五次郎，“HW/SW 協調動作に対するブロックダイアグラム環境利用に関する一評価，” 情報科学技術フォーラム一般講演論文集，pp.195-196，Sep. 2002.
[7] 橋本耕太郎，林悠平，田中康一郎，佐藤寿倫，有田五次郎，“Simulink によるハードウェア/ソフトウェア協調設計環境の構築事例，” 情報処理学会 DA シンポジウム 2003 論文集，pp.1599-1602，Jul. 2003.
[8] SpecC Technology Open Consortium, SpecC Language Reference Manual Version 2.0, 2002.
[9] 木下常雄，富山宏之，“SpecC 仕様記述言語と方法論CQ 出版社，2002.
[10] Interdesign Technologies, *http://www.interdesigntech.co.jp/*.
[11] Y Explorations, Inc., *http://www.yxi.com/*.
[12] K. Tanaka, Y. Iwaya, Y. Hayashi, T. Sato, and I. Arita, “Design and Implementation of FPGA/DSP Based PCI Card,” Proc. 15th International Conference on System Engineering, pp.19-24, Aug. 2003.
[13] Texas Instruments, Inc., TMS320C6701, Code Composer Studio 2.2. *http://www.tij.co.jp/*.
[14] Xilinx, Inc., Virtex-E, ISE 5.2i. *http://www.xilinx.co.jp/*.
[15] Y. Hayashi, K. Tanaka, T. Sato, and I. Arita, “Design of Hardware/Software Co-desin Environment Using SpecC-Based Tools,,” Proc. 18th International Thechnical Conference on Circuits/Systems, Computes and Communications, pp.163-168, Jul. 2003.
[16] 田中康一郎，岩谷祐一，林悠平，佐藤寿倫，有田五次郎，“SpecC による協調設計環境の構築，” 情報処理学会 DA シンポジウム 2003 論文集，pp.157-162，Jul. 2003.
[17] 林悠平，岩谷祐一，田中康一郎，佐藤寿倫，有田五次郎，“FPGA/DSP による協調設計環境の構築，” 信学技報 VLD2002-59，pp.19-24，Jun. 2002