研究室紹介

RICE:動的部分再構成可能な協調処理システム

田中 康一郎　九州産業大学情報科学部知能情報学科
TANAKA, Koichiro　Faculty of Information Science, Kyushu Sangyo University
tanaka@is.kyusan-u.ac.jp, http://www.is.kyusan-u.ac.jp/~tanaka/

1. まえがき

現在，開発期間の長期化や製品の短命化を改善するために，高い柔軟性と能率性を兼ね備えた組込みシステムの実現が期待されている．この課題を実現する方法の一つに，ハードウェアの構成を変更できるプログラマブル・ロジック・デバイス (PLD) の採用がある．ところがこのPLDを利用するには，ハードウェア設計を要するためソフトウェアからの移植が困難，組込みプロセッサで実現されていた機能を全て実装するには実装規模が充分ではない，などの問題点がある．

このうち面積制約については，リコンフィギャラブル・デバイス (RD) で解決できると考えられる．なぜならRDでは，PLDで静的に行っていたコンフィギュレーションを動的に行うことで，仮想的に面積制約のないデバイスとして扱うことができるためである．一方移植性については，リコンフィギャラブル・ロジック (RL) と従来システムとの融合方法で緩和できると考えられる．

以下では，リコンフィギャブル・コンピューティング (RC) の研究を行うために開発した協調システムとその評価結果について論ずる．まず **2.** で，我々の提案するリコンフィギャラブル・システム (RS) のための構成を示す．次に **3.** で，そのシステム構成を評価するために開発した協調システムであるRICEの構成を示し，**4.** でRICEのためのハードウェア・ライブラリ (IP) について論ずる．さらに **5.** と **6.** でRICEの性能評価の結果を示し，**7.** でアセンブリ言語からの分割手法について述べる．

2. システム構成

これまで我々は，FPGAとDSPによる協調処理に関する研究を行ってきた．この研究では，DSPのメモリ・インタフェースにFPGAを直結したハードウェア・プラットフォームであるRYUOH[1] とそれらのための高位レベル設計言語による開発環境 [2] を構築した．その成果から，FPGAの実装面積制約，FPGAの完全同期回路制約，ハードウェア／ソフトウェア・トレードオフによる開発コスト，共有メモリモデルによるデータ衝突，などの課題が残った．

上記の課題を克服するために，RSのためのシステム構成を検討した．一般にRSは，ハードウェアエンジン型 [3, 4, 5]，コプロセッサ型 [6]，プロセッサ型 [7] の3種類に大別できる [8]．このうち，我々は従来の組込みプロセッサを基本としたシステムとの親和性を重視できるようにコプロセッサ型を採用した．一般的なコプロセッサ型は，Garp[9] のように専用バスあるいはメモリバスにRLを接続して用いる．我々が提案するモデルはそれらと異なり，RYUOHで採用したプロセッサとメモリ間にRLを挿入する構成である．図1のようにすることで，プロセッサのメモリバンド幅は低下する可能性はあるが，メモリアクセス時にRLでデータを処理することによって性能向上が期待できる．

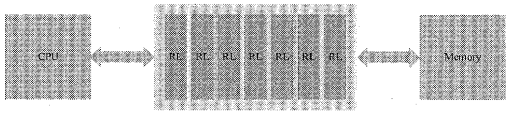


図1 我々が提案するリコンフィギャラブル・アーキテクチャ

上記のアーキテクチャは，RYUOHで問題であった多くの課題を克服できると考えられる．FPGAの実装面積制約については，動的部分再構成で回避できる．完全同期回路制約については，マルチ・クロック・ドメインに対応したIPを開発することで回避できる．トレードオフについては，従来の機能による分割手法に加えて，プロセッサのロード，ストア命令に着目した分割を行うことで，設計自動化に対応できる．共有メモリの衝突に関しては，RLにデュアルポート・メモリで構成されるキャッシュ・メモリを実装することで衝突を最小限にとどめることができる．このように，我々は多くの問題が解決できると考える．

3. RICEの基本構成

RICE[10, 11, 12] は，RYUOHで問題となったハードウェアの実装面積制約を解消できるハードウェア・プラットフォームである．その他のRICEの基本的な設計思想はRYUOHのそれと同等である．図2に示すRICEには，RYUOHと同様にFPGAとDSPが搭載されて

いる．ただしRYUOHではそれぞれ一つずつ搭載していたが，RICEではFPGAの部分再構成機能を活用するために，用途の異なる2つのFPGAを載せた．またRYUOHではPCとのインタフェースとしてPCIインタフェースを用いたが，今回は共同研究であるロボット・システムへの応用 [13] の関係から，USBインタフェース [14] とイーサネット (Ethernet)・インタフェース [15] に変更した．また同様の理由から，CMOSカメラ向けのインタフェース [16] も併せて準備した．メモリについては，チャネル数の増加による性能向上を狙いPC向けのDIMMから通常のSDRのSDRAM[17] に変更した．

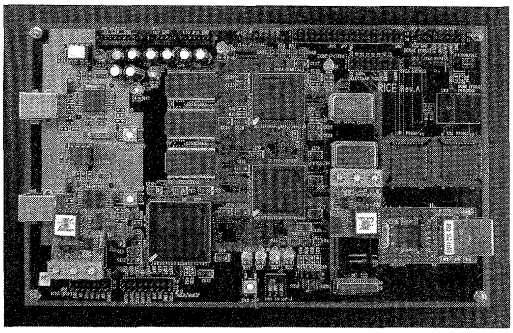


図2 RICE: 動的部分再構成可能なハードウェア

図3に示すように，RICEには2つのFPGAが用意されている．このFPGAは共にXilinx社のVirtex-IIシリーズ [18] である．左のFPGA(以下，**Fixed FPGA**と呼ぶ) はRYUOH上のFPGAと同等の目的で利用される．つまりこのFPGAは各デバイス間を接続するグルーロジック的な用途と専用回路によるハードウェア・アクセラレータ的な用途の双方に用いる．一方右のFPGA(以下，**Reconfig. FPGA**と呼ぶ) は，部分再構成のためのFPGAである．このReconfig. FPGAは，Fixed FPGAのように静的再構成によるハードウェア・アクセラレータ的な使い方も当然できるが，そのような使い方をしてはRYUOHを使用した際に生じるハードウェアの面積的な問題が生じることとなる．そこでこのReconfig. FPGAでは，実行中にFPGA内部の構成を動的変更を繰返すことで，仮想的に無限大の実装規模を実現できるように用いる．このようなハードウェア構成にすることで，RICEではFPGAに対して全て実装できなかったアプリケーションを実装することも可能となると考えられる．

図4に，Fixed FPGAとその他のデバイスとの接続を示す．Fixed FPGAには，DSP, SDRAM, USB, CCDカメラ，Ethernetモジュール，Reconfig. FPGAが接続されている．このFPGAでは入出力端子が8つのバンクに分割されており，それぞれのバンク別に信号レベルを変えることができる [18]．ただし今回の接続では，全てのデバイスの入出力端子レベルがLVTTLで対応できるため，ブロックに跨ってデバイスを接続することが可能である．

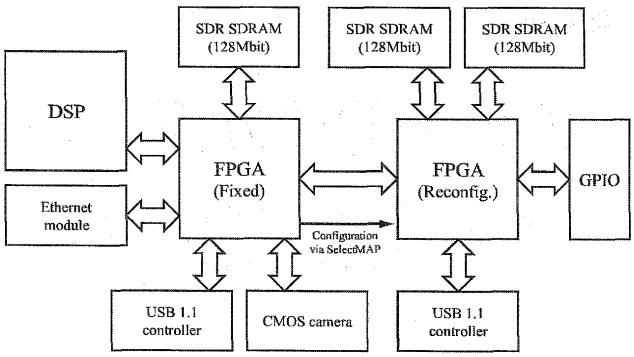


図3 RICEの構成

RICEにおけるDSPとFPGAの接続は，RYUOHと同等である．DSPであるTI社のTMS320C6711C[19] にはメモリインタフェースとしてはEMIFとI/Oインタフェースとしては HPIがあるが，共にFPGAと直接接続されている．なおRICEでは，RYUOHで用意してあったデバッグ向けのSBSRAMについては省略している．

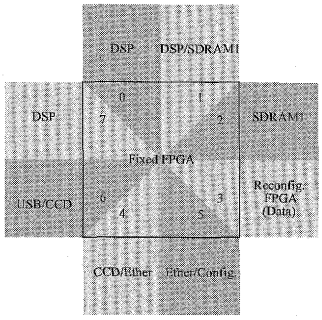
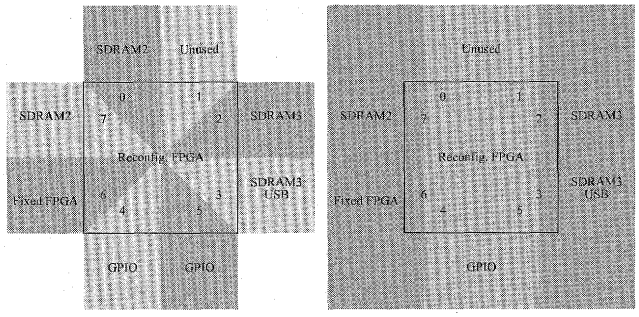


図4 Fixed FPGAの接続

次に，図5にReconfig. FPGAとその他のデバイスとの接続を示す．Reconfig. FPGAには，SDRAM, USB, Fixed FPGAが接続されている．なおSDRAMについては1つではなく2つ接続されている．このFPGAは，前述のFixed FPGAと同一のFPGAであるため，Fixed FPGAと同様に8つのバンクで信号レベルを変更できる．ただしこのFPGAも全てLVTTLで対応できるため，入出力端子レベルにおいてはブロックを考慮する必要はない．

しかしReconfig. FPGAでは，部分再構成に対応するための配慮を行う必要がある．FPGAの再構成は横方向の4つのスライスおよび縦方向全体のスライスを一つの単位として再構成する．その際スライスに付随する入出力端子も再構成される．したがって再構成する領域を最大限に確保するためには，FPGAの左右の入出力端子に全ての他のデバイスを接続できることが望ましい．そこでこのReconfig. FPGAでは図5(b) に示すように左右の最小領域（4スライス）を使ってSDRAM, USB, Fixed.



(a) Input/Output banks (b) Partial reconfiguration

図 5 Reconfig. FPGA の接続

FPGA の全てを接続している．Reconf. FPGA に使用されている FPGA である XC2V1000-4BG456 では横方向に 64 個のスライスが並んでいるが，RICE ではそのうち部分再構成向け領域として最大 56 個のスライスを提供できる．なお部分再構成向けのスライスも固定領域として使うことが可能であり，その領域には汎用の入出力端子 (GPIO) を 36 ピン用意している．このような構成にすることで，FPGA の再構成機能を充分に発揮することができると考えられる．

最後に各 FPGA に対するコンフィギュレーション・モードについて論ずる．Fixed FPGA は，Xilinx 社のダウンロード・ケーブルを介してコンフィギュレーションを行う方法と FLASH ROM (XC18V04[20]) からコンフィギュレーションを行う方法を用意している．前者では，スレーブ・シリアル，SelectMAP，JTAG の各コンフィギュレーションモードに対応しており，後者はマスタ・シリアルに対応している．コンフィギュレーション・モードの切替えは RICE 上のジャンパーピンで行える．他方 Reconfig. FPGA では部分再構成がサポートされる必要がある．これをサポートするコンフィギュレーションモードは JTAG と SelectMAP の 2 つである．RICE は部分再構成を何度も繰返すことで効率的な処理を実現することを目的としているため，コンフィギュレーション時間は非常に重要である．そこで，Fixed FPGA からは高速にコンフィギュレーションを制御できる SelectMAP (slave) を用い，JTAG に関してはダウンロード・ケーブルを介してコンフィギュレーションを行える構成とした．なお Reconfig. FPGA のコンフィギュレーションモードの設定は Fixed FPGA から行うが，JTAG はモード設定が不要であるため，Fixed FPGA をコンフィギュレーションする前に Reconfig. FPGA をコンフィギュレーションすることは可能である．

4. RICE のための IP

RICE を効率的に活用するために，RICE と共に RICE のための IP も併せて開発した．具体的には，ダウンロード・コントローラ，DSP コントローラ，USB コントローラ・インタフェース，SDRAM コントローラ・インタフェースの 4 種類である．これらは，各々異なるクロックで動作するマルチ・クロック・ドメインを想定して設計しているため，IP の内部インタフェースは FIFO メモリで構成されている．これらのうち本稿では，主要な IP であるダウンロード・コントローラと DSP コントローラについて詳説する．

4·1 ダウンロード・コントローラ

§1 ダウンロード・コントローラの動作

ダウンロード・コントローラには，与えられたコンフィギュレーション・データを用いてターゲットとなる FPGA をコンフィギュレーションできることが期待される．このコンフィギュレーション・データは，最初に FPGA の全体をコンフィギュレーションするデータとその後の PDR のためのデータに大別できる．したがって，これらを区別してコンフィギュレーションする必要がある．

ダウンロード・コントローラによるコンフィギュレーションは次のような手順で行われる．まずコンフィギュレーションの種類を認識しているダウンロード・コントローラの外部から，ダウンロード・コントローラのアドレス端子 (ADDR) に対して，調停回路からコンフィギュレーションの種類とビットストリームのサイズが提供される．その後本来コンフィギュレーションに必要なビットストリーム・データがデータ端子 (DATA) へ供給される．ダウンロード・コントローラは，そのビットストリーム・データを用いて FPGA をコンフィギュレーションする．

外部からコンフィギュレーションの種類を与える理由は，コンフィギュレーションの動作が異なるためである．それらの基本動作はほぼ同等であるが，その動作には大きな相違点が 2 点ある．一点目は PDR コンフィギュレーションでは，$\overline{\text{PROG}}$ による初期化をしてはいけないことである．全体のコンフィギュレーションでは，$\overline{\text{PROG}}$ をローにすることで FPGA の初期化を行う．つまり $\overline{\text{PROG}}$ は FPGA のリセット端子であると言える．したがって PDR コンフィギュレーションで同様のことを行ってしまうと，全体が初期化されてしまい PDR コンフィギュレーションができなくなってしまう．したがって，ダウンロード・コントローラでは PDR コンフィギュレーションの際に $\overline{\text{PROG}}$ をローにしていない．もう一点は PDR コンフィギュレーションでは DONE を使用できない点である．全体のコンフィギュレーションでは DONE でコンフィギュレーションの合否を判定することができるが，PDR コンフィギュレーションでは $\overline{\text{INIT}}$ で失敗であるか否かを判定する．ただし，PDR コンフィギュレーションでは，全体のコンフィギュレーションで用いる DONE のようにコンフィギュレーションの成功をコンフィギュレーション端子では認識できない．そのため，コンフィギュレーションの成功を認識するには PDR コンフィギュレーショ

ンで実装される回路に対して工夫をする必要がある．

§2 ダウンロード・コントローラの入出力端子

ダウンロード・コントローラの入出力端子を図6に示す．図中左側の端子はFPGA内部からコンフィギュレーション・データを受信するためのインタフェースである．このダウンロード・コントローラにはコンフィギュレーション・データの一時保管のための2KBのFIFOバッファを用意している．このFIFOバッファに`DIN[7:0]`と`DWR`を使ってデータを書き込むとターゲットであるFPGAに対してコンフィギュレーションを開始する．コンフィギュレーションの正常終了は`INITOUT`により確認できる．また`FULL`によってFIFOバッファの状態は把握できる．一方右側の端子はFPGAのコンフィギュレーションピンに直接接続する端子である．

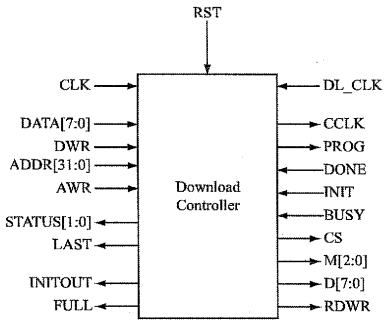


図6 ダウンロード・コントローラの入出力端子

4·2 DSPコントローラ

§1 TI TMS320C67x DSP

本研究のハードウェア・プラットフォームであるRYUOHやRICEで使用した主要なDSPのコアはC67xである．ただしC67xコアはC62xコアの拡張であるためC67xコアに対応しているということは，C62xコアにも対応しているともいえる．RYUOHは，C67xコアのうちTMS320C6701（以下'C6701）と呼ばれるC6000プラットフォームにおける最初の浮動小数点DSPに対応している．一方RICEは'C6701の性能を向上させたTMS320C6711C[19]（以下'C6711）に対応している．表1に'C6701と'C6711Cの比較を示す．'C6711Cは，基本的な性能は大きく変らないが'C6701に比べ最大動作周波数が若干向上していることがわかる．また'C6701は内蔵メモリがキャッシュメモリではなく通常のメモリとして使用するが，'C6711Cではキャッシュメモリまたは通常のメモリとして利用できることがわかる．ただし，レベル1 (L1) キャッシュに関しては，通常のメモリとしては使用することはできない．その他パッケージサイズが異なる以外は大きく違う点はない．最後に参考データとして，表1にコアの異なるTMS320C6201('C6201) の仕様も併せて示す．表1より'C6201と'C6701のデータを比較すると違いがないことが確認できる．

§2 DSPコントローラの設計仕様

本章ではRICEに実装されている'C6711Cを利用するために開発したコントローラの設計仕様を述べる．

'C6711C全体を効率的に制御するためには，2つのインタフェースのためのコントローラを設計する必要がある．一つがHost Port Interface (HPI) であり，もう一つがExternal Memory Interface (EMIF) である．HPIはDSPを制御することができるインタフェースであり，DSP内部の制御レジスタ，内蔵メモリ，外部メモリに対して自由にアクセスすることができる．一方EMIFはメモリインタフェースであり，'C6701Cでは非同期SRAM, SBSRAM, SDRAMのためのインターフェースとして使用することができる．外部メモリは4つの空間に分れており，それぞれ異なるメモリ向けに使うことができる．

DSPを動作させるには，DSP内の制御レジスタを使用する環境向けに設定し，起動する必要がある．'C6711ではHPIから起動する方法とEMIF上のROMからデータを読出すことで設定する方法の2通りがある．なおROMブートの場合は，データ幅を8ビット，16ビット，32ビットの3つから選択することができる．RICEは，スタンドアロンで使うよりもUSBに接続されたPCと連携して使用することが多いと予想されるため，今回はHPIからブートする方法を選択する．

今回HPIのコントローラに必要な機能は次の通りである．

(1) DSPをブートできる．
(2) DSPの制御レジスタを設定できる．
(3) DSPの内部メモリにプログラムを書込める．
(4) DSPの内部メモリにアクセスできる．
(5) DSPの外部メモリにアクセスできる．

今回の設計では上記の仕様を完全に満たすこととした．

次に，EMIFの設計仕様について述べる．EMIFにはデータの読み書きが行えるメモリを接続する必要があるが，RICEではRYUOHのようにEMIFバス上にメモリが用意されておらず，FPGAがEMIFと直接接続されている．したがって，FPGAによってDSPのメモリを構成する必要がある．FPGAによってDSPのメモリを構成するには，FPGA内部のメモリをキャッシュメモリのように用いてFPGA外部のSDRAMを使う方法とFPGAを介して直接SDRAMと接続する方法が考えられる．本研究の目的を考慮すると共に実現できることが望ましいため，どちらとも実現することとする．

§3 HPIコントローラの詳細

HPIコントローラは，HPI経由でDSPの制御を行うことができる．具体的には，DSPの起動，プログラム・コードのダウンロード，実行制御などである．

HPIコントローラのI/O端子の構成を図7に示す．HPIコントローラの左辺はDSPを接続させるための端子である．一方右辺はFPGA内からDSPのI/O空間にアクセスするための端子である．これらは他のIPと同様

表 1 'C6701 DSP と'C6711C DSP の比較

Part Number	Frequency (MHz)	CPU Core	On-Chip SRAM Prog.	On-Chip SRAM Data	EMIF (bit)	External Memory Type Supported	DMA	HPI (bit)	Pachage
C6701	167	C67x	64KB	64KB	32	SRAM SBSRAM SDRAM	4	16	35mm 352BGA
C6711C	200	C67x	4KB (L1)	4KB (L1)	32	SRAM SBSRAM SDRAM	16	16	27mm 272BGA
			64KB (L2)						
C6201	200	C62x	64KB	64KB	32	SRAM SBSRAM SDRAM	4	16	35mm 352BGA

のインタフェースであり，入力バッファへの書込みと出力バッファからの読込みで制御できる．

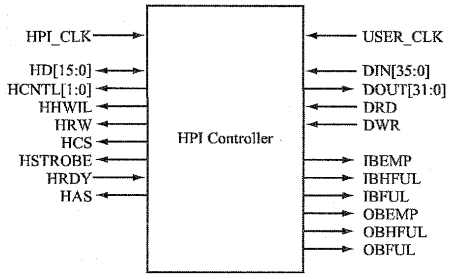


図 7 HPI コントローラの入出力端子

§4 EMIF-SDRAM コントローラの詳細

EMIF-SDRAM コントローラは，DSP の EMIF を経由して DSP と SDRAM とのデータ交換をできる．RICE には SDRAM として Elipda 社の EDS1232AATA を使用している．以下にその概要と各デバイス間の接続について述べる．

i. FPGA による接続

DSP,FPGA,SDRAM の接続を図 8 に示す．単方向の信号線は，DSP の信号を SDRAM に FPGA を介して接続すれば良い．一方データバスのような双方向の信号線は，FPGA の入出力端子の 3 ステートバッファを使用する必要がある．なおデータバスの調停には，$\overline{\text{SDWE}}$ が利用できるが，データのバースト幅に応じてタイミングを調整する必要がある．

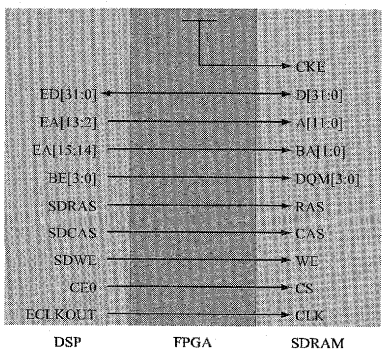


図 8 FPGA を介した EMIF と SDRAM の接続

ii. EMIF レジスタの設定

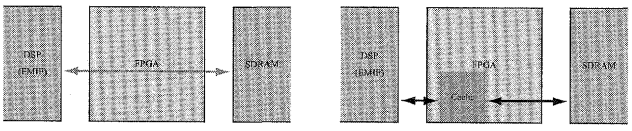
上記の接続を有効にするためには，EMIF レジスタの設定を行う必要がある [19]．本稿における設定を表 2 に示す．

表 2 SDRAM のための EMIF レジスタの設定

Hex Address Range	Acronym	Hex Value
0180 0000	GBLCTL	0000 3078
0180 0008	CECTRL0	FFFF FF33
0180 0018	SDCTL	5748 F000
0180 001C	SDTIM	001E B5DC (default)
0180 0020	SDEXT	001B DF3E

§5 EMIF-CACHE コントローラの詳細

EMIF-CACHE コントローラは DSP の EMIF を経由して DSP と SDRAM との通信を可能にする．この EMIF-CACHE コントローラでは，4•2•4 節で紹介した EMIF-SDRAM コントローラと類似した動作をする．ただし EMIF-SDRAM コントローラのようにコントローラでは EMIF と SDRAM を直接接続するわけではなく，図 9 に示すように FPGA 内にキャッシュメモリを搭載することで EMIF と SDRAM の通信を行う．このような構成にすることで，図 10 に示すように EMIF と SDRAM の間に自由にユーザの意図したロジックを挿入することができるようになる．



a) EMIF-SDRAM　　b) EMIF-CACHE

図 9 EMIF-SDRAM コントローラと EMIF-CACHE コントローラの違い

i. キャッシュメモリの構成

キャッシュメモリの構成方法として，今回の設計では最も単純なダイレクトマップを採用した．図 11 に示すようにダイレクトマップではデータが保管されるデータ・アレイ (Data array) とタグと呼ばれるアドレスの一部が保管されるタグ・アレイ (Tag array) によって構成され

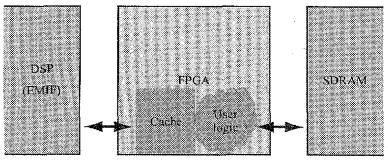


図 10 ユーザ・ロジックを挿入した EMIF-CACHE コントローラ

る．タグ・アレイは，DSP から EMIF 経由でデータを読込みまたは書込みを行う際，データ・アレイに保管されているデータが DSP の期待するデータであるかどうかを確認するために使用される．もしタグ・アレイ 1 のデータとアドレスのデータが同一であればデータ・アレイのデータは有効であるためヒット (Hit) し，同一でなければ無効であるためミス (Miss) する．なおタグ・アレイにはアドレス以外にデータ・アレイのデータ自身の有効を示す有効 (Valid) ビット，データ・アレイ内のデータと SDRAM のデータの相違を示す無効 (Dirty) ビット，メモリ空間 (CEx) の 3 ビットも一緒に保管されている．データメモリのエントリ数は 64 である．ラインサイズは 32 バイトである．このサイズは RICE に搭載されている FPGA の BRAM のサイズと'C6711 の L1 キャッシュのラインサイズ (32 バイト) から決定した．なおタグ・アレイに関しては BRAM を使用すると効率が悪いため，フリップ・フロップによって構成した．この機能以外に，ライトバッファ (Write buffer) を準備した．これにより，本来キャッシングされたくない後処理が直ぐに期待されるデータを効率的に扱うことができる．

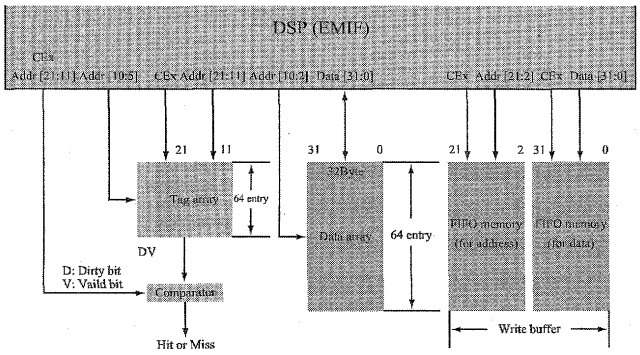


図 11 EMIF-CACHE コントローラの構成

ii. EMIF レジスタの設定

上記の接続を有効にするためには，HPI から EMIF レジスタの設定を行う必要がある [19]．本稿における設定を表 3 に示す．

iii. EMIF-CACHE コントローラの入出力端子

EMIF-CACHE コントローラの入出力端子の構成を図 12 に示す．現在の EMIF-CACHE コントローラは，EMIF-SDRAM コントローラと置換えて利用できるように構成してある．そのため図 8 に示す EMIF-SDRAM

表 3 SRAM のための EMIF レジスタの設定

Hex Address Range	Acronym	Hex Value
0180 0000	GBLCTL	0000 3068
0180 0008	CECTRL0	13D1 8521
0180 0008	CECTRL1	13D1 8521
0180 0018	SDCTL	defalut (0248 F000)
0180 001C	SDTIM	default
0180 0020	SDEXT	default (0017 5f3f)

コントローラとほぼ同様である．大きな違いは，EMIF-CACHE コントローラでは非同期制御を行うため非同期メモリインタフェースのための I/O を用いている点である．また外部 SDRAM のためのクロックは FPGA 内部で生成しているため，それに必要な端子である CLKFB が追加されている．

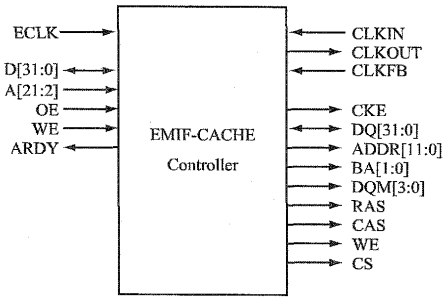


図 12 EMIF-CACHE コントローラの入出力端子

5. ダウンロード・コントローラの性能評価

5·1 PDR のシステム構成

Partial Dynamic Reconfig (PDR) システムは，図 13 に示すように，IP として開発したダウンロード・コントローラ (Download Controller)，SDRAM コントローラ (SDRAM Controller)，USB コントローラ (USB Controller)，とそれらを接続する調停回路 (Arbiter) で構成される．この調停回路には，命令を認識して動作する制御機構 (Controller) と各 IP から IP へのデータを転送する通信機構が実装されている．この通信機構には，データの送受信と同時にデータ幅を変更できる機能やデータをバイト単位でスワップする機能 (swap function) が搭載されている．また各 IP の境界には FIFO バッファ (FIFO Buffer) が用意されており，この FIFO バッファは入力と出力で別系統のクロックを使用できるため，各 IP は調停回路の動作速度に影響を受けることなく最適な動作速度で動作することができる．各 IP の FIFO バッファの深さは異なるが，これは FPGA の BRAM の最小サイズに依存しているためであり，全ての FIFO バッファは約 18kB である．

次に DRP コントローラを用いたときのビットストリーム (Bitstream) の流れを説明する．まず PC 上にある

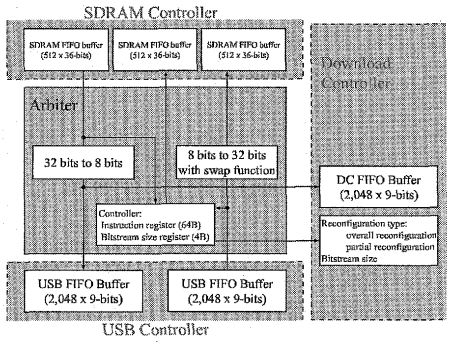


図 13 PDR システムの構成

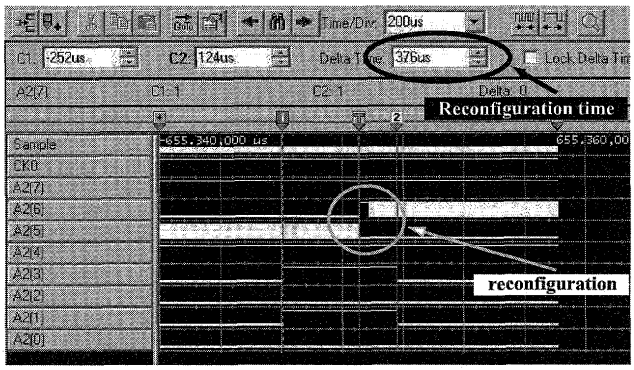


図 14 DRP コントローラの波形 (376μs)

Reconfig. FPGA 用のビットストリームは，USB コントローラと調停回路を経由して SDRAM に格納される．その際ビットストリームのヘッダ部分 (最初の 64 バイト) に対して，調停回路向けの命令を添付しておく．次に調停回路に対してダウンロードするための命令を送信する．すると，その命令からビットストリームデータの最初のメモリアドレスとデータサイズを認識し，その後 SDRAM からそのビットストリームを取得し，そのビットストリームをダウンロード・コントローラに送信する．また，SDRAM に書込んだデータを PC 上で確認するために，SDRAM から PC へデータを転送する機能も有している．なおこの PDR コントローラは，RICE 上の Reconfig. FPGA に対して，全体コンフィギュレーションおよび部分コンフィギュレーションが可能である．

5·2 PDR システムの実装結果

PDR システムの実装結果を表 4 に示す．このビットストリームの生成には，Xilinx 社の論理合成ツールと配置配線ツールである XST 7.1 と ISE 7.1 を用いた．その際，XST のレジスタバランシングオプションを有効にした．各 IP の境界で利用している FIFO バッファの数は，BRAM の数で確認できる．またデジタル・クロック・マネージャ(Digital Clock Manager: DCM) は，定倍，分周，位相シフト機能を備えており，基本クロックの倍クロックで動作する IP (Download, SDRAM) で利用されている．なお今回の PDR システムの設計事例では，調停回路などの特に定った周波数を要求しないものに関しては，ターゲットとする最小周期 (Minimum period) を全て 20ns とした．実装結果から，RDP コントローラは全てのタイミング制約を満たすことを確認できた．なお USB コントローラの最小周期が IP 単体における時間よりも全体 (Overall) における時間の方が短くなっている．この原因は全体では USB コントローラの全ての機能を使っていないためであった．

DRP コントローラの波形を図 14 に示す．この結果から理想的な時間でコンフィギュレーションが行えることが確認できた．

6. DSP コントローラの性能評価

6·1 DSP システムの構成

DSP システムは，図 15 で示すように，UBS コントローラと DSP コントローラ (DSP Controller)，そしてそれらを接続する調停回路 (Arbiter) で構成される．さらにこの DSP コントローラは，DSP のメモリ・インタフェースのための EMIF コントローラ (EMIF Controller) と I/O インタフェースである HPI コントローラ (HPI Controller) に大別される．DSP システムの調停回路も，前述の PDR システムと同様に，制御機構と通信機構が実装されている．なお動作クロックに関しても，PDR システム同様に各 IP や調停回路は FIFO バッファで接続されているため適切な周波数を選択できる．

次に DSP システムの動作について説明する．DSP を利用するには，まず DSP の初期設定を行う必要があるが，PC からはそれらを DSP のメモリとして見なすことができる．したがって，最初に PC から DSP の初期設定を行うべきメモリ領域に対して設定を書込み，次に DSP で実行すべきプログラムを DSP の内部メモリ領域に書込む．最後に DSP の実行させるための領域に “1” を書込むことで動作を開始する．このように USB 経由で DSP の制御やプログラム書込みが行える．また設定の結果やプログラムの実行結果を PC から USB 経由で確認することもできる．

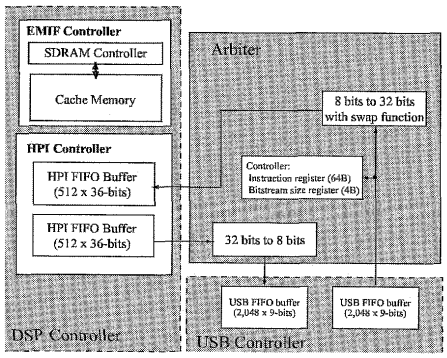


図 15 DSP システムの構成

表4 PDRシステムの実装結果

	Overall	Download	SDRAM	USB	Arbiter	Available
Flip flops	1,049	140	465	136	321	10,240
4-input LUTs	1,889	353	585	124	1,013	10,240
BRAMs	8	1	3	4	0	40
DCMs	3	1	2	0	0	8
Minimum period (ns)	19.804/20 (Core) 9.892/10 (Download) 9.870/10 (SDRAM) 9.508/42 (USB)	11.656/20 9.967/10	10.486/20 9.920/10	12.082/20 11.428/42	10.955/20	—

Using Xilinx ISE 7.1 and XST 7.1.

6·2 DSPシステムの実装結果

DSPシステムの実装結果を表5に示す．このビットストリームの生成にも，Xilinx社のXST 8.2とISE 8.2を用いた．EMIFコントローラには4つのBRAMが使用されているが，そのうち3つがEMIFコントローラに内蔵されているSDRAMコントローラで使用し，残りの一つがキャッシュメモリのデータアレイとして利用されている．なお，このEMIFコントローラで利用されているSDRAMコントローラは，キャッシュのデータメモリのブロックサイズを考慮して読込み書込み共にバースト転送(32バイト）できるように変更している．

6·3 DSPシステムの性能評価

システムの評価として，まず各コントローラを用いたDSPシステムにおける最大動作周波数を調べた．EMIF-CACHEコントローラを用いたDSPシステムは，'C6711Cの最大動作周波数である200MHzで正常に動作した．一方EMIF-SDRAMコントローラを用いたDSPシステムでは，'C6711Cとメモリに間にFPGAが介在しているにも関わらず，図16に示すように124MHzまで正常に動作した．

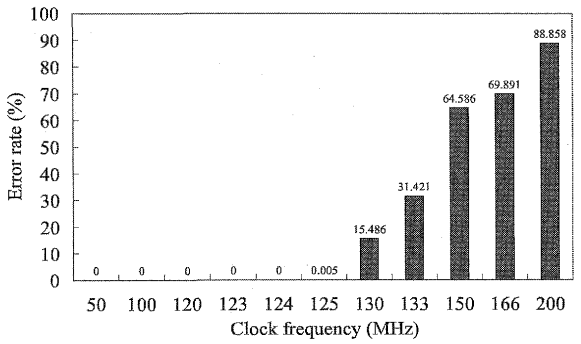
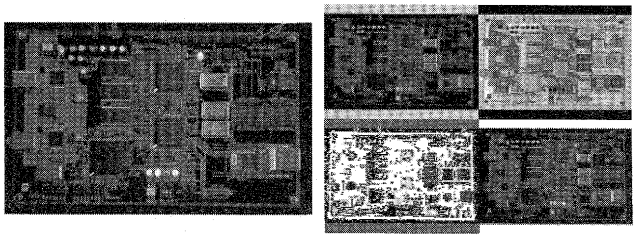


図16 EMIF-SDRAM IPの性能評価

次に，'C6711Cの機能であるChip Supprt Library[21]を用いて各システムのデータ転送バンド幅を測定した．評価のためのアプリケーションには，図17に示すような，色空間変換プログラムを用いた．図18は，200MHz動作のEMIF-CACHEコントローラのデータバンド幅の計測結果である．計測の結果，マルチメディア・アプリケーションのような連続してデータを扱うものには，DMAを活用することが非常に有効であるということが確認できた．また100MHz動作のEMIF-SDRAMコントローラについても，図19に示すように，同様の結果が得られた．



a) Original image (320×240)　　b) Peocessed image

図17 色空間変換の実行例

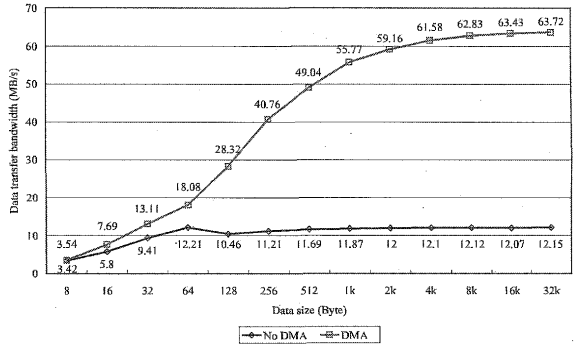


図18 200MHz動作の'C6711C DSP上のEMIF-CACHEコントローラのデータ転送バンド幅

7. アセンブリ・コードからのハードウェア・ソフトウェア分割に関する検討

一般にソフトウェアとハードウェアの分割は，機能に着目する方法とプログラムの実行負荷に着目する方法に分けることができる．前者では，ライブラリのようなものが利用できれば比較だけで良いが，新機能の場合双方を設計して比較する必要があるため開発や検証に多くの時間を必要とする．他方後者では，正確に評価するため

表 5 EMIF-CACHE コントローラを用いた DSP システムの実装結果

	Overall	EMIF	HPI	USB	Arbiter	Available
Flip flops	2,693	2,093	209	161	505	10,240
4-input LUTs	4,091	2,885	168	126	1,376	10,240
BRAMs	12	6	2	4	0	40
DCMs	2	2	0	0	0	8
Minimum period (ns)	19.914/20 (Core) 9,972/10 (EMIF) 24.751/42 (USB)	19.275/20 9.956/10	16.778/20	13.931/20 24.860/42	13.750/20	—

Using Xilinx ISE 8.2 and XST 8.2.

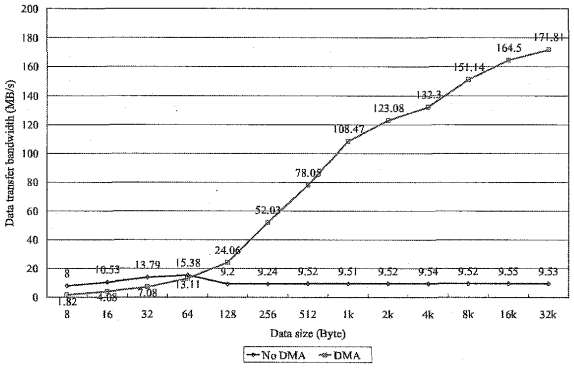


図 19 100MHz 動作の'C6711C DSP 上の EMIF-SDRAM コントローラのデータ転送バンド幅

にプログラムを実環境で動作させる必要があるが，実環境上でトレースさせることは容易ではない．そこで，本章ではアセンブリ・コードを対象としたハードウェア・ソフトウェア分割について検討する．本アーキテクチャでは CPU とメモリとの間に RL があるため，ハードウェアでの処理を CPU の前処理あるいは後処理とみなすことができる．そのため，アセンブリ・コードのメモリアクセスに着目することで分割が可能である．この手法が実現できれば，ハードウェア向けの回路を人手で開発する前に，設計自動化によるハードウェア・ソフトウェア分割が可能となり，従来手法より短時間で処理性能を向上させることができるようになると考える．

アセンブリ・コードに着目した分割を評価するために，評価コードとして JPEG エンコーダにおける色空間変換のためのコードを用いた [1]．そのうち式 (1) に示すような輝度の計算を対象として検討した．なお表 6 に示すように，アセンブリ・コードを正常に生成するために輝度の結果はメモリに格納するように記述している．

$$*Y = (short)((76 * (*r) + 150 * (*g) + 28 * (*b) - 32768) >> 8); \quad (1)$$

図 20 に式 (1) から生成されたアセンブリ・コードを示す．また表 7 にアセンブリ・コードの動作を示す．CPU の前処理では，今回の評価結果から'C6711C ではロード命令 (LDx) でメモリからデータを読込みことがわかるため，その命令を起点にデータの流れを追従することでそれらの処理を RL で前処理できるかどうかを検討することが可能であることが確認できた．またハードウェアを効率的に動作させるには，扱うデータの幅が確実にわかると効果的であるが，この問題に関しては表 8 に示すようにロード命令とストア命令にデータ・タイプとアセンブリ・コードから図 21 に示すような適切な回路設計が行えることも併せて確認できた．なおハードウェアとソフトウェアのトレードオフはハードウェア構成に依存するため，ここでは言及しない．

表 6 変数のデータ型式とメモリ・マップ

Variable	Data Type	Hex Address Range
r	usigned char	0x80000000
g	usigned char	0x80000001
b	usigned char	0x80000002
Y	short	0x80000004

表 7 色空間変換のためのアセンブリ・コードの動作一覧

Line Number	Description
L4-6, L8-9, L13	ロード命令のためのアドレスの生成
L11-12, L15	データのロード
L16-17, L20	定数の設定
L21-22, L24	乗算
L28, L30	加算
L27, L31	32768 を設定
L33, L36	ストア命令のためのアドレスの生成
L37	右シフト
L39	データのストア

8. むすび

本論文では，FPGA の動的部分再構成機能に着目したハードウェア・ソフトウェア協調システムである RICE の詳細について論じた．RICE の基本動作については問題なく動作することから，我々の提案するリコンフィギャブル・アーキテクチャが評価できる環境が整備できた．今後は，このアーキテクチャの評価を行う予定である．

```
1     _main:
2     ;** --------------------------------------------*
3
4                 MVKL    .S2     0x80000002,B4     ; |16|
5     ||          ZERO    .L2     B5                ; |16|
6     ||          MVKL    .S1     0x80000001,A3     ; |16|
7
8                 MVKH    .S1     0x80000001,A3     ; |16|
9     ||          MVKH    .S2     0x80000000,B5     ; |16|
10
11                LDBU    .D1T1   *A3,A3            ; |16|
12    ||          LDBU    .D2T2   *B5,B6            ; |16|
13    ||          MVKH    .S2     0x80000002,B4     ; |16|
14
15                LDBU    .D2T2   *B4,B5            ; |16|
16                MVK     .S2     150,B4            ; |16|
17                MVK     .S1     76,A4             ; |16|
18                NOP             1
19
20                MVK     .S1     28,A4             ; |16|
21    ||          MPYUS   .M1X    B6,A4,A3          ; |16|
22    ||          MPYUS   .M2X    A3,B4,B4          ; |16|
23
24                MPYUS   .M2X    B5,A4,B5          ; |16|
25    ||          RET     .S2     B3                ; |18|
26
27                ZERO    .L1     A3                ; |16|
28    ||          ADD     .S1X    A3,B4,A4          ; |16|
29
30                ADD     .L1X    B5,A4,A4          ; |16|
31    ||          SET     .S1     A3,0xf,0xf,A3     ; |16|
32
33                MVKL    .S2     0x80000004,B4     ; |16|
34    ||          SUB     .L1     A4,A3,A3          ; |16|
35
36                MVKH    .S2     0x80000004,B4     ; |16|
37    ||          SHR     .S1     A3,8,A3           ; |16|
38
39                STH     .D2T1   A3,*B4            ; |16|
```

図 20 色空間変換のためのアセンブリ・コード

表 8 TMS320C6x DSP のロード命令とストア命令の一覧

Mnemonic	Data type	Mnemonic	Data type
LDB	char	STB	char
LDBU	unsigned char	STH	short
LDH	short	STW	int
LDHU	unsigned short		
LDW	int		

(a) Load instructions (b) Store instructions

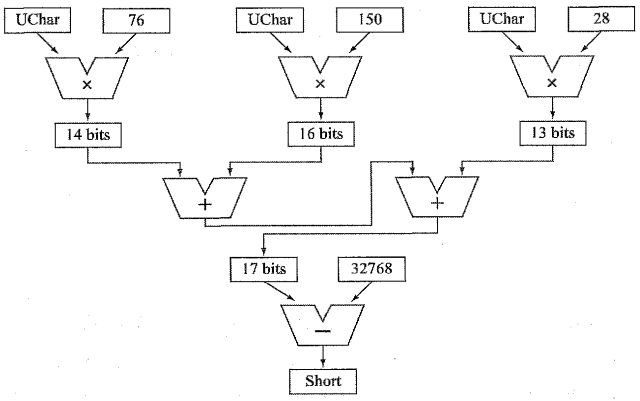


図 21 アセンブリ・コードから生成できた色空間変換回路

◇参 考 文 献◇

[1] K. Tanaka, Y. Iwaya, Y. Hayashi, T. Sato, and I. Arita. Design and Implementation of FPGA/DSP Based PCI Card. In *Proceedings of Fifteenth International Conference on Systems Engineering*, pp. 407 – 413, August 2002.

[2] Y. Hayashi, K. Tanaka, T. Sato, and I. Arita. Design of Hardware/Software Co-Design Environment Using SpecC-Based Tools. In *Proceedings of the 2003 International Technical Conference on Circuits/Systems, Computers and Communications*, Vol. 3, pp. 1599–1602, September 2003.

[3] M. Motomura. A Dynamically Reconfigurable Processor Architecture. In *Microprocessor Forum*, October 2002.

[4] T. Sugiwara, K. Ide, and T. Sato. Dynamically Reconfigurable Processor Implemented with IPFlex's DAPDNA Technology. *IEICE Transactions on Information and Systems*, E87-D(8):1997–2003, August 2004.

[5] K. Tanigawa, T. Hironaka, A. Nojima, and N. Yoshida. PARS Architecture: A Reconfigurable Architecture with Generalized Execution Model — Design and Implementation of Its Prototyoe Processor. *IEICE Transactions on Information and Sysytems*, E86-D(5):830–840, May 2003.

[6] M. Wazlowski, L. Agarwal, T. Lee, A. Smith, E. Lam, P. Athanas, H. Silverman, and S. Ghosh. PRISM-II Compiler and Architecture. In *Proc. IEEE Workshop on FPGAs for Custom Computing Machines*, pp. 9–16, April 1993.

[7] Z. A. Ye, A. Moshovos, S. Hauck, and P. Banerjee. Chimaera: a high-performance architecture with a tightly-coupled reconfigurable functional unit. In *Proceedings of the 27th Annual International Symposium on Computer Architecture*, pp. 225–235, June 2000.

[8] 末吉, 天野 (編) **リコンフィギャラブルシステム** オーム社, August 2005.

[9] T. J. Callahan, J. R. Hauser, and J. Wawrzynek. The Garp Architecture and C Compiler. *Computer*, 33(4):62–69, April 2000.

[10] K. Tanaka. An FPGA Implementation of Partial Dynamic Reconfiguration Controller and Processor Interfaces for a Processor-Based System with Reconfigurable Logics. In *Proceedings of International Technical Conference on Circuits/Systems, Computers and Communications*, Vol. 2, pp.157–160, July 2006. Invited paper.

[11] 川原, 田中, 佐藤. 動的部分再構成可能な FPGA を搭載した FPGA/DSP プラットフォーム RICE のためのコンフィギュレーション・コントローラの実装In *FPGA/PLD Design Conference* **ユーザ・プレゼンテーション論文集**, pp. 1–8, January 2006.

[12] K. Tanaka. Development of a Dynamically Reconfigurable Hardware Platform Using General-Purpose FPGAs. In *Proceesings of the 20th Commemorative International Technical Conference on Circuits/Systems, Computers and Communications*, Vol. 2, pp. 565–566, July 2005.

[13] 田中, 江島. 高度画像処理 LSI とその応用In **第 4 回計測自動制御学会制御部門大会予稿集**, pp. 731 – 734, May 2004.

[14] Cypress Semiconductor, Corp. *EZ-USB Technical Reference Manual*, May 2000.

[15] WIZnet Inc. *IIM7010A*.

[16] 大野, 橋本, 田中, 佐藤. FPGA による CMOS カメラ制御機構の設計. In **第 11 回電子情報通信学会九州支部学生会講演会**, p. 6, September 2003. 電子情報通信学会学生会講演奨励賞.

[17] Elpida Memory, Inc. *128M bits SDRAM EDS1232AATA Data Sheet*, May 2003.

[18] Xilinx, Inc. *Virtex-II Platform FPGAs: Complete Data Sheet*, July 2004.

[19] Texas Instruments, Inc. *TMS320C6711, TMS320C6711B, TMS320C6711C Floating-Point Digital Signal Processors*, July 2004.

[20] Xilinx, Inc. *XC18V00 Serial In-System Programmable Configuration PROMs*, July 2004.

[21] Texas Instruments, Inc. *TMS320C6000 Chip Support Library API User's Guide*, February 2003.