

FPGA アクセラレータによる Android アプリケーションの高速化手法

小 池 恵 介[†] 太 田 淳[†] 大 島 浩 太[†]
藤 波 香 織[†] 郡 信 幸^{††}
竹 本 正 志^{††} 中 條 拓 伯[†]

Android 端末における Java 実行の高速化をはかるために、FPGA によるアクセラレータを実装し、ハードウェア実行可能な Java のソース部分を FPGA 上で実行することにより全体の性能向上を実現する。Dalvik VM を実行する Intel Atom プロセッサと FPGA との間において、PCI Express インタフェースの実装を完了し、DMA 転送により 1.25 Gbps の通信性能を確認し、Atom + FPGA による実験システムの構築を進めている。ここでは、FPGA アクセラレータによる Android の高速化手法について述べ、実験環境および、プロセッサと FPGA 間の通信性能について報告し、アクセラレーションによる性能見積と今後の発展について述べる。

Java Acceleration with an FPGA Accelerator in an Android Mobile Terminal

KEISUKE KOIKE,[†] ATSUSHI OHTA,[†] KOHTA OHSHIMA,[†]
KAORI FUJINAMI,[†] NOBUYUKI KOHRI,^{††} MASASHI TAKEMOTO^{††}
and HIRONORI NAKAJO[†]

We have implemented an FPGA accelerator, which realizes higher performance by executing a part of a Java source code executable in hardware, to accelerate Java execution in an Android mobile terminal. Between an Intel Atom processor which executes a Dalvik virtual machine and an FPGA accelerator, we have implemented PCI Express interface which performs high speed communication of 1.25 Gbps with DMA transfer in our experimental environment. In this paper, acceleration of Android with an FPGA accelerator is described. Communication performance between a processor and an FPGA has been measured and the future performance with the acceleration is estimated.

1. はじめに

1.1 背景：携帯機器と FPGA

現在、携帯機器の高機能化が進められ、Apple 社の iPhone の登場で拍車がかかり、NTT docomo、SoftBank、au などの携帯キャリアや国内メーカなどがスマートフォンの開発、販売を積極的に進めており、それを利用するユーザの要望もますます高くなっている。現在のところ、携帯機器の限られた機器容積の中に搭載できるハードウェア量も限られており、ユーザの用途についてもやや閉塞感はあるが、今後性能に関するブレークスルーとともに、新たな用途、可能性は広がっていくものと考えられる。今後、このような携帯機器を開発プラットフォームとして利用して行く上に

において、機器によっては、限られた環境でしか、ネットワークを利用した実験を行うことができないものもある。とりわけ、広域ネットワーク環境において実験を進めていくには NDA を結ぶ必要があったりなど、研究開発の足かせとなっているのも事実である。

これに対し、Google 社によって開発された Android を搭載した端末が急速に普及しつつある。Android は、OS、ミドルウェア、そしていくつかの基本的なアプリケーションからなる、携帯情報端末向けのプラットフォームだけでなく、最近では組み込み Android として、組み込み機器への利用が広まりつつある。Android は、オープン・ソースで提供されており、無償で利用・改変できるため、開発コストを抑えたい携帯機器メーカ各社が積極的に採用を進めている。実際、HTC、Sony Ericsson などが相次いで Android 端末をリリースしている。

Android には、独自の Java の仮想マシン (VM) を有し、多くの Java プログラムがその上で実行されると

[†] 東京農工大学

Tokyo University of Agriculture and Technology

^{††} 株式会社ビート・クラフト

BeatCraft, Inc.

いう特徴がある。Android では、Java を Dalvik バイトコードと呼ばれる中間言語に変換し、それを Dalvik VM と呼ばれるレジスタ・ベースのプロセッサ上で実行する。そして、アプリケーションだけでなく、それらのフレームワークも Java で記述される。VM を介して実行することから、Dalvik バイトコードの実行時のオーバーヘッドは大きい。また、独自の VM を使用しているため、既存の Java VM の高速化技術をそのまま適用することが困難である。その結果、Dalvik VM 上での Java の実行時間は JavaVM 上でのそれよりも劣ってしまう。

従来から、書き換え可能 LSI である FPGA (Field Programmable Gate Array) を活用して、ターゲットとなる LSI をエミュレーションにより動作確認する方が採られることが多い。さらに、その FPGA も、大容量化し、高速化が進められ、従来の LSI エミュレーションだけでなく、ネットワーク機器やメディア処理機器など、さまざまなプロトコルに対応したり、内部仕様が頻繁に更新されるような機器などにおいて利用されるようになってきた。

今後の携帯機器においては、さまざまなネットワーク環境への対応が要求され、それぞれに応じたプロトコルに対応する必要がある。ユーザや利用環境に応じた機器を実現する上で、ソフトウェアのみで対処するには、現状のプロセッサパワーでは対応できず、かといって、環境、ユーザに応じた携帯機器シリーズを多種準備することは、メーカー側にとってコストに見合わない。そういった中で、組込み機能をハードウェア的に再構成可能となるような携帯機器が望まれている。

2010 年 11 月に、Intel 社は、モバイル系のプロセッサと FPGA を同一のパッケージに搭載した新たなプロセッサを発表した¹⁾。現状このプロセッサと FPGA は PCI Express x2 で接続されており、アプリケーションによっては通信遅延が問題になると考えられる。しかし、今後高速化が求められ、また LSI 実装技術が進めば、プロセッサと FPGA はより密に接続されるようになり、この二つの間の通信遅延は短くなっていくものと期待される。

1.2 今後の携帯機器の方向性

以上から、今後の携帯機器、スマートフォンに FPGA の搭載という動きは一つの方向性をなすものと予測される。現状で問題となっているのは、FPGA の消費電力であり、電池駆動で利用するには、長時間の利用に耐えられないということが挙げられる。しかしながら、低消費電力の FPGA の開発は進められており、一般的な FPGA では、その内部論理を SRAM に保持す

るのに対して Flash ROM に記録することで大幅な電力削減を行った FPGA が ACTEL 社によっても開発されており、今後の携帯機器への応用も進められている。しかしながら、現状の製品動向を見てみても、実際に FPGA を搭載した携帯電話、スマートフォンなどは製品化にはいたっていない。

前述のように、Android 機器では、Java をもとにした Dalvik コードを実行することで、さまざまな携帯アプリケーションに対応している。しかしながら、Dalvik VM のオーバーヘッドから、一部のアプリケーションについては、実行速度が問題となっている。単純な方向性としては、プロセッサの周波数を上げることであるが、それには消費電力の増大を招くこととなる。また、複数のコアを用いて、内部において並列処理を行う方向も研究が進められているが、一気にコア数を増加させるようなことは、現状においてハードウェア的にも、ソフトウェア上においても、多くのハードルがあり、劇的な速度向上は期待できない。

以上の背景から、我々は、今後のスマートフォンの一つの形態として、Reconfigurable Android を提唱し、FPGA 搭載の Android 機器を開発することで、画像、動画、音声などのマルチメディア処理の高速化とともに、Android 携帯機器の高機能化を目指す。さらに、種々のセンサを搭載し、さまざまなネットワーク環境に対応できるように、再構成可能な Android プラットフォームの一形態を作り上げていく。

本稿では、現在推進している Reconfigurable Android の概念について述べる。そのプラットフォームとして、FPGA を搭載して再構成可能とする機構を組み込んだ Android システムを紹介し、その高速化手法について説明する。現在の実験環境および、プロセッサと FPGA 間の通信性能について報告し、画像処理を例として、アクセラレーションによる高速化の性能評価結果を示し、今後の発展について述べる。

2. Reconfigurable Android

現在、東京農工大と、Android のオープンソース・ガジェットを開発している株式会社ビート・クラフトにおいて、次世代の Android 機器の可能性を探るべく、Reconfigurable Android プロジェクトを推進している。本プロジェクトにおける全体構想イメージを図 1 に示す。

Reconfigurable Android とは、ARM や Atom などの汎用プロセッサと FPGA を組み合わせ、Android 機器にさまざまな機能を組み込み、高速化とともに柔軟性、発展性を伴う新たな Android の形態を模索し、

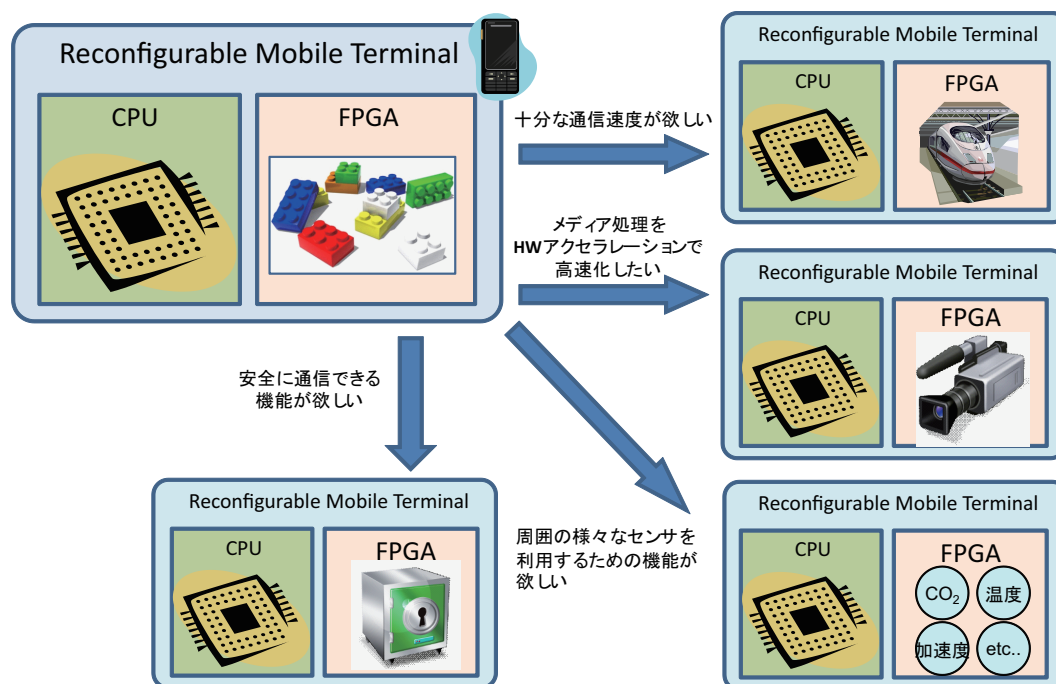


図 1 リコンフィギャラブル Android 携帯機器プロジェクトの全体イメージ

携帯機器のみならず，今後の組み込み Android においても，その可変機構を有効に発揮するシステムの構築を目指すものである．

2.1 ネットワークにおける構造可変の必要性

Reconfigurable Android は，携帯機器単体の性能向上とともに，ネットワーク環境を含め，今後のユビキタス環境における応用を示していくことも目的としている．そのためには，ネットワーク上のさまざまな問題を解決して行く必要がある．具体的な対応事項を以下に示す．

- (a) 急速に進歩・変化する情報通信技術への即時対応
- (b) サービスやアプリケーションがネットワークに求める広帯域・応答性・安全性などの機能を柔軟に提供
- (c) 分散管理されセキュリティが保障されないインターネットの安全利用
- (d) 周囲に設置されたセンサ情報から情報を収集する際に必要なプロトコルへの自動対応
- (e) 収集した膨大なセンサ情報の有効利用方式

携帯端末における無線通信は，Wi-Fi や 3G 以外にも IrDA, Bluetooth, ZigBee などの近距離通信から，RFID や TransferJet などの近接通信まで様々なものが利用でき，また新しい通信方式は日々開発が進められている．これらの通信方式を利用するサービスは，人気度や利用率が日々変化し，さらに多くの新しいサー

ビスが日々開発・提供されている．この変化は，同時にサービスがネットワークに求める要求機能も変化することを意味している．あるサービスが人気の時はネットワークに広帯域性が求められ，他のサービスが人気の時は安全性が重視されるなどである．このような流動的な変化に対して柔軟に対応することが求められる (a),(b),(c)．また，センサ技術と無線通信技術の発展に伴い，無線通信機能を備えた様々なセンサが設置され，社会生活に浸透すると考えられる．このような状況下では，どのような通信方式およびプロトコルでセンサから情報を取得するかが課題となる．環境情報を網羅的に取得するには膨大な数のセンサを様々な環境に設置することになり，必然的にセンサは安価で低性能なものにならざるを得ない．そのため，携帯端末は，センサが利用可能な通信方式やプロトコルに自動的かつ適応的に対応できる必要がある (d)．さらに，センサから収集した膨大な量のデータは，ユーザが理解できる形に変換して提示することで，初めてユーザにとって価値のある情報となる．そのため，ユーザにとって利便性の高いセンサ情報の活用法が求められる (e)．

これまでの情報通信分野では，標準化により相互接続性を保とうとする動きにあるが，標準は新しい技術開発の足かせになる場合も多く，また標準がそのサービスやシステムに対して常に有効であるとは限らない．

そのため、多様な通信方式やプロトコルに柔軟に対応するというアプローチにより、独創的な研究開発を促進しつつ、真に有益なサービス、アプリケーションを即座に広く提供することができると考えられる。

対して、現在の情報通信分野は、各国が競い合って新しいインターネットを作ろうという動きの中にある。そのため、これまでは既存網との相互接続性が最重視され研究開発の足かせになっていたが、相互接続性を考えない独創的な多くの新しい通信方式やプロトコルが提案されつつある。その中でも、利用者数が爆発的に増加している携帯端末を用いた無線通信技術に世界的な関心が高まっている。しかし、研究者が携帯端末上で容易に無線通信技術を開発できる有効なテストベッドが無い状態であり、研究開発が進んでいるとは言い難い。FPGA を搭載したりコンフィギュラブル携帯端末があれば、まずネットワークの階層構造のうち低位に位置する物理層について研究開発を促進することができる。また、Reconfigurable Android では無償で利用・改変できる Android を用いるため、物理層だけでなく、OS の機能としてソフトウェアで実現される上位層までを一貫して開発することが可能である。さらに、従来はソフトウェアで処理していた機能を FPGA 上に実装することによるネットワークの高速化、省電力化、安全化も期待できる。前述した今後のユビキタス環境におけるネットワーク上の様々な問題は、Reconfigurable Android 携帯端末により解決することが可能である。

さらに、今後の低炭素時代を目指すためには、機器やデバイス自体の省電力化は必要不可欠ではあるが、さらに、携帯機器の活用方法として、個人に対して、電力、水道使用量、さまざまな環境状態をリアルタイムでユーザに働きかける仕組みが必要となると考えられる。携帯機器は通常、ポケットやかばんなどに携行する場合が多く、またネットワークに接続されているために、環境に関するさまざまな情報の収集や伝達が可能となる。また、今後のホームエレクトロニクス、ホームネットワークと連動することで、家庭内における情報を瞬時に得ることができるだけでなく、現在自分が置かれている状況に対しても、得られた環境情報を管理サーバに伝達することで、きめ細かい環境分析といったことも可能となる。

こういった機能を実現するためには、携帯機器を高機能化し、さまざまなセンサを組み込むような機構が必要となる。しかしながら、あらゆるセンサを搭載することはオーバスペックとなる危険性があり、また機器自体が大掛かりなものとなり、携行に困難を伴って

しまい、一般に普及しなくなる恐れがある。そこで、それぞれの使い方にしたがって、センサを接続できるようなインタフェースが要求され、そのインタフェースにも FPGA が活用できるものと考えられる。

2.2 Reconfigurable Android における高速化

Dalvik バイトコードのソフトウェアによる高速化手法は古くから考案されている。その多くは VM によるバイトコードの逐次翻訳に替わり、バイトコードから実行するプロセッサに対応する機械語を生成し、それを実行するといった手法である。ネイティブコードの生成タイミングに応じて、大きく分けて JIT (Just in Time) コンパイルと AOT (Ahead of Time) コンパイルに分類される。

これに対し、Dalvik バイトコードをハードウェアによって解釈することで、高速化を行う手法について挙げる。ここではこの手法を Java アクセラレータと呼ぶ。Java アクセラレータは VM の中で最も負荷が高いバイトコード実行を、ハードウェアで行うことで高速化を図る。Java アクセラレータの実装方法は、いくつかあるが、ここでは水野らが分類に用いた 3 つの実装方法²⁾を基に、ハードウェアによる方式を挙げ、各々の特徴を示す。図 2 に、各実装方式におけるプロセッサとアクセラレータの接続関係を示す。

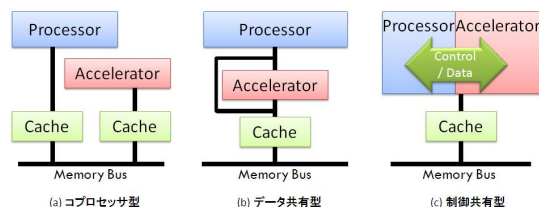


図 2 Java アクセラレータの実装方式

- データ分離型：コプロセッサのように、主となるプロセッサとは独立したプロセッサ機能を有する。コプロセッサバスやメモリマップド I/O を介して、アクセラレータを制御する。
- データ共有型：データバスを共有する方式であり、データバスに介在し、バイトコードをネイティブ・コードに変換してプロセッサへ入力する。接続方式の制約上、バイトコードを読み出してからネイティブ・コードを CPU へ出力するまでに数サイクルかかる。
- 制御共有型：アクセラレータはプロセッサに組み込まれ、プロセッサの制御も行う。単純に変換した命令列を送り出すだけでなく、プロセッサの動作を読み出し、その内容からより緻密なアクセラ

レーションを可能とする．

我々は、これまで上記の制御共有型の Jazelle 方式に基づいたハードウェアアクセラレーションについて研究を進めてきた^{3)~5)}．Jazelle DBX は ARM アーキテクチャ向けの Java アクセラレータであり、実際の ARM プロセッサに実装されている．しかしながら、制御共有型はプロセッサ内部に組み込まれるため、アクセラレーション機構を組み込むためには、既存のプロセッサに追加することは困難であり、新たなプロセッサを設計・実装することとなり、今後もバージョンアップされていく Android の Dalvik コードに対しては、開発コストが問題となる．

そこで、既存のプロセッサには手を加えない、コプロセッサ型やデータ共有型のアクセラレーションに着目し、そのアクセラレータの実装に FPGA を活用する方式を、前述の Jazelle 方式と並行して研究を進めることとした．

2.3 Reconfigurable Android を実現する bc-R

これまで、Android 携帯の開発環境として、ARM プロセッサをベースとした TI 社製の OMAP 3530 を搭載したオープンソース・ガジェットの bc9 および bc10 を利用してきた⁶⁾．そこで、Reconfigurable Android を実現するプラットフォームとして、FPGA を搭載した、bc-R の開発を現在進めている．

bc-R の設計指針として、以下が挙げられる．

- 汎用プロセッサをハードコアで、もしくは IP コアとして内蔵する FPGA を搭載すること．
- 信号処理用に乗算器もしくは DSP を多数利用可能な FPGA を搭載すること．
- センサを接続するための USB ポートを複数搭載すること．
- データログ用に大容量 SD カードインタフェースを搭載すること．
- bc10 のソフトウェア資産を、デバイスドライバを含めて、できる限り活用できること．
- 胸ポケットに入る程度のコンパクト化をはかること．

この指針をもとに、現状における内部ブロック図を図 3 に示す．

3. Reconfigurable Android 開発環境と高速化方式

3.1 開発環境

Reconfigurable Android を実現する bc-R の開発環境として、図 4 に示すようなシステムの設計、構築

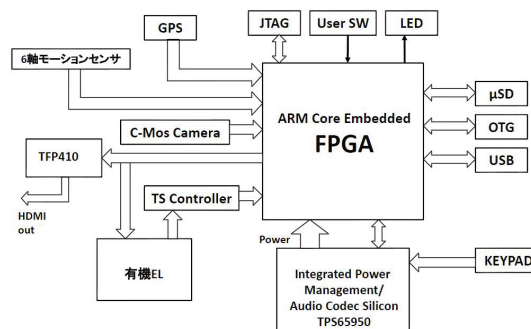


図 3 bc-R の内部ブロック図

を行なった．本システムの目的は、Android 上における種々のアプリケーションの実行によるオーバーヘッドの検証とアクセラレーション方式とその有効性の評価を行うものである．

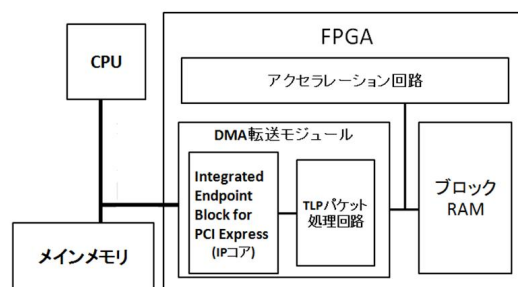


図 4 Reconfigurable Android 開発環境

この環境を実現するために、現在、我々は図 5 に示す、東京エレクトロニクス社の Application Reference Platform for Image Processing (ARPIP) を利用したテストベッドシステムの構築を行なった．ARPIP には、インテル Atom プロセッサ Z530 と Xilinx Spartan-6(XC6SLX45T) を搭載した画像処理機器開発のためのシステムである．

ARPIP は、元々は高精度カメラを接続して、画像処理を行うためのシステムとして開発されたものであり、開発環境や通信インタフェースはすべて Windows をベースにしたものしか提供されていない．しかしながら、Android を実装するためには Linux をベースにした環境が必要不可欠であり、Atom プロセッサ - FPGA 間の PCI Express インタフェースにおいても、プロトコル、デバイスドライバを含めて、Linux の環境下での開発が必要であり、その実装を行なった．

3.2 FPGA アクセラレーションを用いた実行方式

ARPIP 上において、FPGA を用いて、Android ア



図 5 Application Reference Platform for Image Processing

アプリケーションの高速化を試みる。FPGA の制御は Java ソースコード内において記述する。この方式は、Java において提案されている方式であるが⁷⁾、Android において実現された例はない。実際に FPGA の制御を行うのはデバイスドライバであるが、デバイスドライバの関数呼び出しはネイティブコードライブラリを用いて行う。従って、本システムでは、FPGA の制御を行うために、以下の 3 種類のソースファイルを用いる。

- Android アプリケーションの処理について記述した Java ソースファイル
- open や read, write, ioctl などのファイル入出力システムコールを行う C ソースファイル：コンパイルされた後 so ファイルとなり、ネイティブコードライブラリとして Java の実行時にリンクされる。
- デバイスドライバとなる C ソースファイル

Java ソースファイルには FPGA との通信を行うメソッドがいくつか定義されており、デバイスオープンやデータの送受信などの機能を持つ。これら通信メソッドの具体的な処理は、Java ソースファイルではなく別の C ソースファイルに関数として定義されている。C ソースファイルの処理はネイティブコードライブラリとして、Java アプリケーションから JNI を通じて呼び出される。C ソースファイルでは、open や write, read, ioctl システムコールを行い、対応するデバイスドライバの関数を呼び出す。デバイスドライバはメモリマップド I/O を通じて FPGA を制御し、Java アプ

リケーションと FPGA との通信を行う。

以上、Android における OS とアプリケーションの関係について以下に示す。

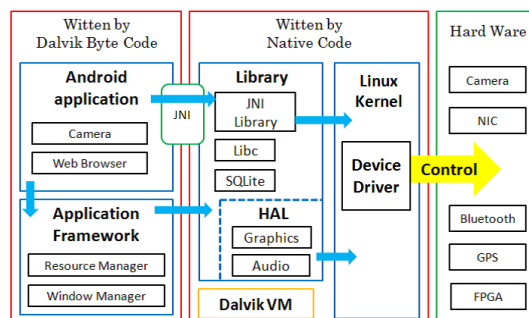


図 6 Android における OS とアプリケーションの関係

まず、Java アプリケーションは FPGA が行う処理に必要なデータを全て FPGA へと転送する。その後、FPGA のアクセラレーション回路に処理開始命令を出し、FPGA と並列に処理を進める。FPGA は処理が完了すると CPU に割り込みをかけ、デバイスドライバがこれを処理する。デバイスドライバには、FPGA の処理を管理する変数が用意されており、Java アプリケーションは適当なタイミングでこの変数を読み取り、処理が完了していれば結果を受け取る。

3.3 FPGA 内部モジュール構成

FPGA 内部のモジュールは以下の 3 つのモジュールから構成される。

- (1) PCI Express による DMA 転送を行うモジュール
- (2) アクセラレーション回路
- (3) ブロック RAM

(1) には内部に Xilinx 社の IP コアが含まれている。この IP コアはデータリンク層の処理を行い、TLP パケットの入出力インタフェースを持つ。DMA 転送はメインメモリとブロック RAM 間で行われ、その転送には、DMA バッファのアドレス、ブロック RAM のアドレス、転送するバイト数の 3 つの情報を必要とする。デバイスドライバがメモリマップド I/O にこれらの情報を書き込むと、FPGA にパケットが送信される。FPGA はパケット内に示されたアドレスによって、続くデータの種別を判別し、適切なレジスタに格納する。メインメモリからブロック RAM へ転送を行う際は、このインタフェースにリードリクエストを送信し、そのコンプリーションとしてメインメモリのデータを受信する。ブロック RAM からメインメモリへ転送を行う際は、インタフェースにライトリクエスト

トを送信する。転送が完了すると CPU に対し割り込みをかける。アクセラレーション回路は、Java アプリケーション内の特定の処理を CPU の代わりに行う。

デバイスドライバは、init_module, delete_module, open, close, read, write, ioctl の 7 種類のシステムコール, および割り込みハンドラに対応するエントリポイントを準備する。それぞれのシステムコールとその機能を表 1 に記す。

表 1 システムコールと機能

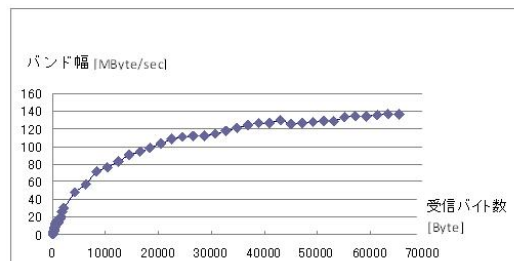
システムコール	機能
init_module	・キャラクタデバイスの登録 ・PCI デバイスの有効化 ・割り込みハンドラの登録
delete_module	・キャラクタデバイスの削除 ・割り込みハンドラ登録解除
open	・DMA バッファの確保
close	・DMA バッファの解放
read	・DMA バッファからユーザ空間へコピー
write	・ユーザ空間から DMA バッファへコピー
ioctl	・FPGA へ DMA 転送するバイト数を DMA 転送モジュールに登録 ・送信元メインメモリアドレスを DMA 転送モジュールに登録 ・送信先ブロック RAM アドレスを DMA 転送モジュールに登録 ・メインメモリから FPGA への DMA 転送を開始 ・メインメモリへ DMA 転送するバイト数を DMA 転送モジュールに登録 ・受信元ブロック RAM アドレスを DMA 転送モジュールに登録 ・受信先メインメモリアドレスを DMA 転送モジュールに登録 ・FPGA からメインメモリへの DMA 転送を開始

4. Reconfigurable Android 開発システムの性能評価

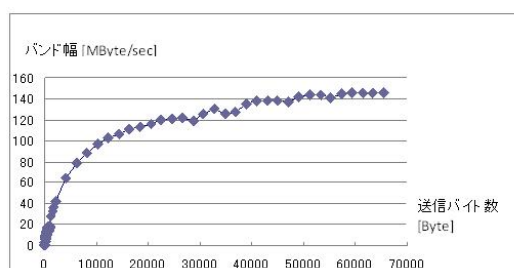
4.1 プロセッサ - FPGA 間の通信性能

Reconfigurable Android 開発システムにおいて, PCI Express モジュールを FPGA 内に実装し, Linux 上においてデバイスドライバを開発した。図 7 には, プロセッサ - FPGA 間の転送バンド幅を測定した結果を示す。(a) は Atom プロセッサカード上のメインメモリから FPGA 内の Block RAM への転送を行なった場合であり, (b) はその逆方向の場合である。この結果から, 現状において 40KB の転送バイト数あたりから 140MB/S のバンド幅の性能を発揮している。PCI Express 1.1 規格では, 最大 250MB/S の転送速度を保証しているが, 現状の性能は Spartan-6 に搭載

された Rocket I/O の性能により縛られたものとなっている。



(a) メインメモリ → Block RAM



(b) Block RAM → メインメモリ

図 7 転送バイト数とバンド幅の関係

図 8 には, プロセッサ - FPGA 間における転送時の通信レイテンシを測定した結果を示す。レイテンシは FPGA が DMA コントローラに対し, リードリクエストを発行してからデータが届くまでの時間を計測した。この結果から, レイテンシは 6μ 秒から 11μ 秒の間にあることがわかる。

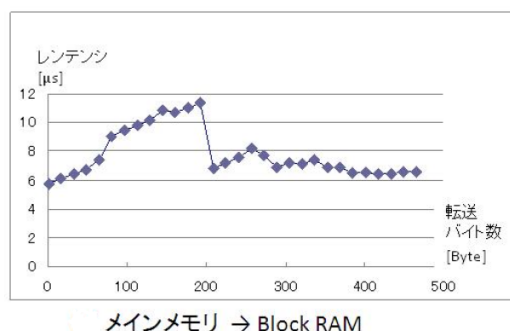


図 8 転送バイト数とレイテンシの関係

4.2 画像処理による性能評価

Reconfigurable Android 開発システムにおいて, ラプラシアンフィルタを適用したエッジ検出画像処理を

行う Java プログラムを、FPGA アクセラレーションにより高速化した。Java ソースコードにおけるエッジ検出関数部分を Verilog-HDL で記述し、論理合成・配置配線して作成した Configuration ファイルを予め用意しておき、3.2 節で示した手順により、Java プログラムから FPGA アクセラレータを起動して高速実行を行う。なお、使用した Android のバージョンは Android 2.2 を x86 向けにポーティングしたものであり、JIT コンパイルは実装されていない。動作周波数は 62.5MHz であり回路規模は表 2 のとおりである。

表 2 回路規模

Logic Resources	Used	Utilization Rate
Slice	3530	51%
FF	13224	24%
6 in LUT	8466	31%

256 × 256 の画像に対して実行した結果、ソフトウェアによる実行では 1785ms 要したのに対し、FPGA アクセラレーションによる実行では、624ms であり、高速化の効果を実証できた。FPGA アクセラレーションによる実行の内訳は以下となっていた。

- FPGA に転送するための画素データの準備: 316ms
- FPGA へのデータ送信に要した時間: 455 μ s
- FPGA からのデータ受信に要した時間: 478 μ s
- エッジ検出のハードウェア処理時間: 310 μ s
- FPGA から受信したデータを画像として表示するための処理: 306ms

これより、現状の開発システムは PCI Express の 1 レーンを使用しているため、多少のデータの転送時間を要しているが、処理の大部分は画像データを FPGA で処理するための前処理と処理後に表示するための処理時間が大半を占めている。しかしながら、実際の処理時間のみを比較すると、数百倍から千倍を超える高速化が期待でき、FPGA によるハードウェアアクセラレーションの可能性は Android においても示すことができると考えられる。

また、FPGA とプロセッサ間の通信時間については、今後 Intel 社の Atom E600C のようなプロセッサと FPGA が一体となった形態においても、LSI 実装技術の向上などにより短縮されていくものと期待される。

5. 現状と今後について

現在、Reconfigurable Android 開発システムについては、Java コードによるベンチマークプログラムの中の LU 分解や FFT などの数値計算を中心とした

アプリケーションをハードウェア化して評価を進めている。

bc-R については、Atom と FPGA が一体となったプロセッサを利用する以外に、Xilinx 社がすでに ARM コアのシリーズを発表しており、ARM プロセッサをベースとした形態も考えられる。

上記設計・実装を進めるとともに、Reconfigurable Android 開発システム上において、さまざまな通信機構についても、最適な形で再構成を行えるような機構を考案し、bc-R が未来の携帯機器の標準プラットフォームとなるインテリジェントな携帯機器としての地位を築くべく、研究を推進していく。

謝辞 本研究の一部は、独立行政法人日本学術振興会とフィンランドアカデミーとの二国間交流事業（共同研究）による支援を得た。

参 考 文 献

- 1) <http://www.intel.com/jp/intel/pr/press2010/101124.htm>
- 2) H. Mizuno, N. Irie, K. Uchiyama, Y. Yanagisawa, S. Yoshioka, I. Kawasaki, and T. Hattori. SH-Mobile3: Application Processor for 3G Cellular Phones on a Low-Power SoC Design Platform. Hot Chips 16 (2004).
- 3) 太田 淳, 茂手木 貴彦, 三輪 忍, 中條 拓伯: Dalvik アクセラレータのための MIPS シミュレータを用いた評価環境, 先進的計算基盤システムシンポジウム (SACIS2010) ポスター・セッション, Vol.2010, No.5, pp.113-114 (2010)
- 4) 太田 淳, 三輪 忍, 中條 拓伯: Dalvik アクセラレータ: Android 端末における Java アプリケーションの高速実行機構, 組込みシステムシンポジウム (ESS2010), pp.13-22 (2010)
- 5) 太田淳, 三輪 忍, 中條 拓伯: Android 端末におけるハードウェアによる Java の高速化手法の提案, 情報処理学会論文誌 コンピューティングシステム, Vol.4, No.3, pp.115-132 (2011)
- 6) <http://labs.beatcraft.com/ja/index.php?Open%20Source%20Gadgets>
- 7) E. Lattanzi, A. Gayasen, M. Kandemir, N. Vijaykrishnan, L. Benini, A. Bogliolo: Improving Java performance using dynamic method migration on FPGAs, Int. Journal of Embedded Systems Vol.1, No.3-4, pp.228-236 (2005)