4J-2

OpenCLを用いたメニーコア・アクセラレータの仮想化手法と評価環境の構築

坂本 龍一[†]　佐藤 未来子[†]　天野 英晴[‡]　中村 宏[††]　近藤 正章[‡‡]　並木 美太郎[†]
[†]東京農工大　[‡]慶應義塾大学　[††]東京大学　[‡‡]電気通信大学

1　はじめに

高性能な組込み向け SoC は高性能化に伴い回路規模が増大し消費電力が増加している．バッテリーにて動作する組込みシステムでは大きな問題である．そこで CREST 省電力プロジェクトにおいて消費電力の問題を解決するために，電力効率が高い省エネルギー大規模リコンフィギャラブルシステムの Cool Mega-Array（CMA）[1] と汎用のプロセッサからなる組込みシステムの研究開発が進められている．プロジェクトでは三つの CMA と一つの MIPS プロセッサを 3 次元集積した Cube-1[2] の研究開発を行っている．しかし Cube-1 をソフトウェアからどのように高速かつ省電力に実行する手段は示されていない．Cube-1 を高い稼働率で動作させるためには転送，実行を非同期に行う必要があり，これらをアセンブラにて記述するのは難しい．そこで本研究では CMA 用 OpenCL API を提供し C/C++から非同期にアクセラレータを利用できる環境を目指す．本 OpenCL API を用いることで CMA を高効率に動作可能なプログラミングが可能とする．NVIDIA，Intel，AMD による OpenCL 実装は存在するが，省エネルギー大規模リコンフィギャラブルデバイス向けに OS とともに OpenCL 実装を行う点が本研究の特徴である．本論文では OpenCL API と CMA の動作の対応を検討し，CMA が OpenCL API を用いて簡潔な記述より制御可能であることを示す．また，検証用に Cube-1 を複数 FPGA にて構築する．

2　メニーコアアクセラレータ Cube-1

Cube-1 は汎用プロセッサ一枚と三枚の CMA を積層し構成されている．図 1 に Cube-1 の概要と CMA の概要を示す．CMA はストリーミング処理を想定しており二つのバンクのメモリを持つ．BSEL の値により CMA の 2 つのメモリの接続先が変更される．一方のメモリが PE と接続されているときは，もう一方が外部のルータと接続される．BSEL は CPU 側から制御できる．また CMA には CMA 間 DMA に対応した DMAC を備えている．

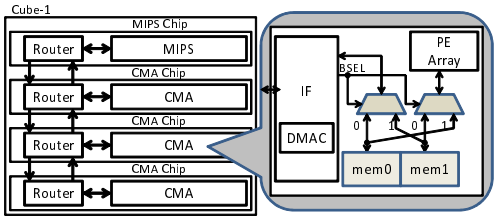

図 1: Cube-1 の概要と CMA の概要

3　Cube-1 用 OpenCL API

本章では Cube-1 用 OpenCL API について示す．初めに設計方針を示し，次に CMA と OpenCL デバイスとの対応関係を示す．最後に CMA 用に新設する関数について述べながら動作を示す．

3.1　設計方針

本研究では Cube-1 を対象に OpenCL による抽象化を行う．複数の CMA を制御することに特化した OpenCL API を提案しアセンブリコードで記述しなくとも詳細に CMA を制御できるようにする．したがって必要に応じて CMA 専用の OpenCL API を追加することとする．本研究における OpenCL API ではカーネルの実行，メモリコピー等を行う部分に限定する．すなわち，CMA 用プログラムはホストに事前に別途生成しておき，アクセラレータのプログラムを本 OpenCL API を用いて実行制御する環境を目指す．

3.2　プラットフォームモデルとの対応

OpenCL にて定義されているホスト，デバイス，メモリモデル，ワークアイテム，及びワークグループと CMA の対応を示す．またホスト，演算デバイス，及びメモリモデルと Cube-1 の対応関係を図 2 示す．データ，カーネルのコピーを行うホストは汎用プロセッサに対応させる．三つの CMA を一つの演算デバイスに対応させる．CMA 内部のメモリをローカルメモリとする．ただし，ホストから読み書き可能なものとする．CMA 間ではメモリ共有ができないためワークアイテム，ワークグループは最大数を一とする．

3.3　動作モデルと追加 API

一つの CMA 上で一つのカーネルを実行する．CMA の二つのメモリ制御を OpenCL の枠組みで行うが，従来の OpenCL の仕様では本メモリのバンク制御を

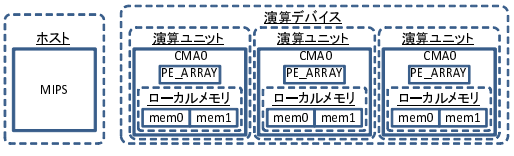

図 2: CMA とプラットフォームモデルの対応

Virtualization techniques using OpenCL and construct evaluation environment for Many-core accelerator
Ryuichi Sakamoto[†], Mikiko Sato[†], Hideharu Amano[‡], Hiroshi Nakamura[††], Masaaki Kondo[‡‡] and Mitaro Namiki[†]
Tokyo University of Agriculture and Technology([†])
Keio University([‡])
The University of Tokyo([††])
The University of Electro-Communications([‡‡])

```
//q は command_queue    e は event
memA=cl_CMA_CreateBuffer(context,
    CL_MEM_READ_WRITE,mem_size,NULL,&err);      (1)
memB=cl_CMA_CreateBuffer(context,
    CL_MEM_READ_WRITE,mem_size,NULL,&err);      (1)
clSetKernelArg(kernelA,0,sizeof(int),memA);     (2)
clSetKernelArg(kernelB,0,sizeof(int),memB);     (2)
clEnqueueWriteBuffer(q,memA.mem0,CL_TRUE,
    offset,data_size,in_data,0,NULL,e1);        (3)
cl_CMA_EnqueueChangeBSEL(q,memA,mem1,1,e1,e2); (4)
clEnqueueTask(q,kernelA,1,e2,e3);               (5)
cl_CMA_EnqueueChangeBSEL(q,memA,mem0,1,e3,e4); (4)
clEnqueueCopyBuffer(q,memA.mem0,memB.mem0,
    offset,offset,data_size,1,e4,e5);           (6)
cl_CMA_EnqueueChangeBSEL(q,memB,mem1,1,e5,e6); (4)
clEnqueueTask(q,kernelA,1,e6,e7);               (5)
cl_CMA_EnqueueChangeBSEL(q,memB,mem0,1,e7,e8); (4)
clEnqueueReadBuffer(q,memB.mem0,CL_TRUE,
    offset,data_size,out_data,1,e8,NULL);       (7)
```

図 3: OpenCL 疑似コード

記述できない．そこで OpenCL 側に CMA 用メモリ確保関数 cl_CMA_CreateBuffer と BSEL 切換え関数 cl_CMA_EnqueueChangeBSEL を追加する．

3.4 OpenCL API を用いた実行手順

手順 (1) から (7) を用いる簡単なプログラム例を図 3 に示す．また，手順 (1) から (7) の動作を図 4 に示す．in_data のデータに対して処理 A を行い，その後 B の処理を行い out_data に出力するものとする．手順 (3)，(6)，(7) において指定する CMA 側のメモリは，手順 (1) で確保した cl_CMA_mem メンバの値 (memA.mem0 等) を指定する．関数名に Enqueue と付くものはエンキューされ実行される．そのためイベントオブジェクト生成し，実行の状態を監視することが可能となり，イベントにて同期や実行順序を制御することができる．

(1) メモリの確保

cl_CMA_CreateBuffer にて CMA の二つのメモリを確保する．戻り値は新設する構造体の cl_CMA_mem 型へのポインタとし，cl_mem 型のメンバ mem0，mem1 を返す．

(2) カーネルの割り当て

clSetKernelArg にてカーネルを割り当てる．メモリ確保で確保したメモリオブジェクトを引数としカーネルと CMA をバインドする．

(3) ホストのデータを CMA へ転送

clEnqueueWriteBuffer にてホストのデータを CMA へ転送する．

(4) BSEL の設定

新設する cl_CMA_EnqueueChangeBSEL にて BSEL を設定する．この操作はエンキューし実行する．第三引数が BSEL の値とし，“ mem0 ”又は“ mem1 ”を指定する．指定した値のメモリが CPU 側から読み書きできるようになり，指定していないもう一方が PE 側と接続される．

(5) 実行

clEnqueueTask にてタスクを実行する．

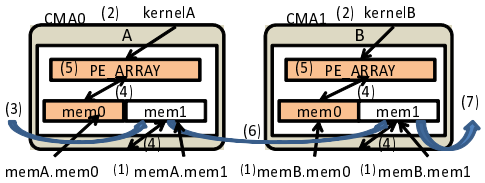

図 4: CMA 用 OpenCL を用いた CMA 動作

(6) CMA 間 DMA

clEnqueueCopyBuffer にて転送元と転送先を指定し CMA 間 DMA を用いデータのコピーを行う．

(7) CMA のデータをホストへ転送

clEnqueueReadBuffer にて CMA のデータをホストへ転送する．

プログラム例に示したように，CMA 用に拡張した OpenCL API を用いることで CMA へのデータの引き渡し，CMA 間 DMA，BSEL の切換え等を一行で記述でき，簡潔に CMA 制御を記述できるようになる．また，イベントオブジェクトを用いることで非同期な転送や実行を指示できる．

4 FPGA による評価環境

システムソフトウェアの検証用に Cube-1 環境を FPGA 上で構築している．構築した評価環境を図 5 に示す．Xilinx 社の ML605 を四枚用いて，一枚のチップに対し一台の ML605 を使用し，ボード間を RocketIO にて接続している．

5 終わりに

Cube-1 向けの OpenCL ランタイム API 実装の検討を行った．メモリ管理用の二つの拡張 API を追加することによって Cube-1 を OpenCL の枠組みで制御可能であることを示した．今後は FPGA を用いた評価環境上で CAM 用デバイスドライバの実装を行い CMA 用 OpenCL ランタイム API の実装を進める．

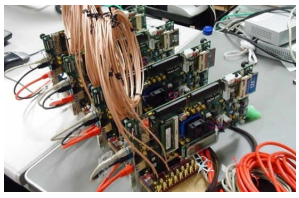

図 5: FPGA 評価環境

謝辞

本研究は JST CREST「革新的電源制御による次世代超低電力高性能システム LSI の研究」によるものである．

参考文献

[1] Ozaki Nobuaki 他:Cool Mega-Arrays: Ultralow-Power Reconfigurable Accelerator Chips, Micro, IEEE, Nov.-Dec. 2011, Volume: 31

[2] 佐々木瑛一 他:チップ間ワイヤレス接続を利用した三次元積層アーキテクチャの研究，信学技報 CPSY2011-10,Vol.111,No.163 ,pp.7-12,Jul 2011.