2K-7

組込み CMA アクセラレータ用 OpenCL ライブラリと OS の設計

坂本 龍一† 佐藤 未来子† 小泉 佑介‡ 近藤 正章‡‡ 天野 英晴‡
中村 宏‡‡ 並木 美太郎†
東京農工大学† 慶應義塾大学‡ 東京大学‡‡

1 はじめに

CMA アクセラレータ[1]は一つの汎用 MIPS プロセッサチップと複数の CMA チップ，チップ間ワイヤレス伝送路から構成され，パイプライン処理を得意とするマルチコアプロセッサである(図 1)．MIPS プロセッサ上では Linux 等の OS が動作し，CMA の制御を担う．CMA は演算器アレイをベースとした超省電力再構成可能アクセラレータであり，それぞれの CMA には，ダブルバッファで構成されたローカルメモリと，CMA 間データ転送用の DMAC が搭載されている．CMA にはそれぞれ異なるカーネルを割り当て，演算データをチップ間で効率良く受け渡すことで，積層した複数の CMA チップを有効に並列動作させることができる．一方で，複数のカーネルを並列実行させる時の CMA 特有の制御が複雑化することが課題となる．

本研究では，CMA アクセラレータの実行制御に OpenCL API を用い，OpenCL の枠組みの中で CMA を提供する．CMA 間の同期やデータ転送などを OpenCL API で指示することで，低レベルの煩雑な CMA カーネル制御をプログラマから隠ぺいするシステムを目指す．本論文では， CMA アクセラレータ向けの OpenCL による CMA 制御方式について述べる．

2 CMA アクセラレータ向け OpenCL 環境

2.1 CMA 用 OpenCL の概要

本研究では，CMA アクセラレータを OpenCL のタスク並列モデルで抽象化し，OpenCL API プログラミングにより，CMA 間同期，ダブルバッファ制御，DMAC を用いたデータ転送などの CMA 特有の煩雑な制御をプログラマから隠蔽する．OpenCL のモデルにおいて，タスク制御やメモリ管理を行うホストに対応する部分を MIPS プロセッサに対応させ，OpenCL の演算デバイスを各 CMA に対応させる．CMA プログラマは各 OpenCL デバイスへカーネルをマッピングし，カーネル間の同期に関しては OpenCL のイベントオブジェクトを定義して依存関係を記述する．

Design of OpenCL library and OS for CMA embedded accelerator
Ryuichi Sakamoto†, Mikiko Sato†, Yusuke Koizumi‡, Masaaki Kondo‡‡, Hideharu Amano‡, Hiroshi Nakamura‡‡, Mitaro Namiki†
Tokyo University of Agriculture and Technology†
Keio University‡
The University of Tokyo‡‡

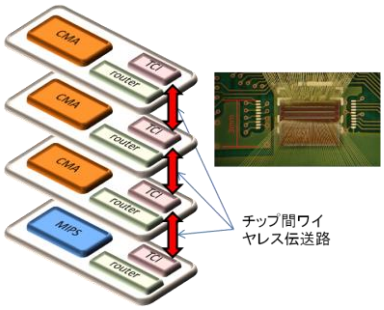

図 1 CMA アクセラレータ

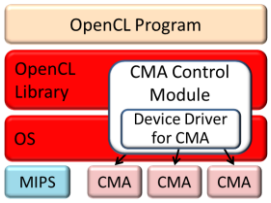

図 2 OpenCL プログラム実行基盤

2.2 OpenCL プログラム実行基盤の概要

CMA 用 OpenCL プログラムの実行基盤を図 2 に示す．MIPS プロセッサ上で稼働する OS では OpenCL プログラムを実行させ，OpenCL ライブラリと CMA 制御部で CMA を管理・制御する．

OpenCL ライブラリでは，CMA アクセラレータの抽象化モデルを実際の CMA アクセラレータに対応付けるために，OpenCL API で記述されたカーネルオブジェクト，イベントオブジェクト，メモリオブジェクト，コマンドキューを CMA に対応付けて管理する．また，メモリオブジェクトに対する Read/Write やカーネル実行など，OpenCL の各種イベントを CMA ごとに実行するために，各 CMA 制御に必要となる内部的な CMA 制御コマンドを生成し，これを CMA 制御部で逐次処理する設計としている．CMA 制御コマンドというインタフェースを内部で設けることにより，実際に CMA に対する制御を司る CMA の実装に依存する部分と，OpenCL デバイスを仮想的に管理するライブラリ部分とを分けて設計することができるようにしている．例えば，図 2 では CMA 制御部を CMA 制御モジュールとデバイスドライバといったソフトウェアで実現する例を示しているが，CMA 制御コマンドを直接解釈実行するハードウェアの導入も可能になるといった利点がある．

表 1　CMA 用 OpenCL API の機能

CMA用OpenCL API	機能
clCreateBuffer	OpenCLで抽象化されたCMAのローカルメモリを扱うためのメモリオブジェクトを生成
clCreateProgam	CMA上で動作するカーネル用プログラムオブジェクトを生成
clCreateCommandQueue	OpenCLのコマンドキューオブジェクトを生成する．また，コマンドキューオブジェクトとCMAを1対1に対応させて，物理CMAを予約
clCreateKernel	OpenCLのカーネルオブジェクトを生成
clSetKernelArg	カーネルオブジェクトに対してメモリオブジェクトをバインド
clEnqWrite	主記憶からCMAのローカルメモリへデータを書込み
clEnqRead	CMAのローカルメモリから主記憶へデータを読込み
clEnqTask	CMAのカーネルを実行
clWaitForEvent	OpenCLで指定したイベントの終了をホスト側で待機
clFinish	OpenCLのコマンドキューへ入れたすべてのイベントの終了をホスト側で待機
clGetEventInfo	OpenCLのイベントの状態を確認

3 CMA アクセラレータ向けの OpenCL による CMA 制御

本研究で中心的な役割を果たす OpenCL の API を表 1 に示す．その中の clEnqRead，clEnqWrite，clEnqTask，clFinish，clWaitForEvent の API が呼ばれた際に，CMA 用 OpenCL ライブラリが表 2 に示す CMA 制御コマンドの組みを各 CMA に応じて生成し，CMA 制御部がこれを受け取り実行する．どの CMA で CMA 制御コマンドを実行するかは，clCreateCommandQueue で OpenCL のコマンドキューオブジェクトを生成した時に，コマンドキューオブジェクトと CMA を 1 対 1 に対応付けておき，上記 API の引数で指定されるコマンドキューオブジェクトによって，制御する CMA を識別可能にする．

以下，関数ごとに生成される CMA 制御コマンドについて述べる．

clEnqWrite では Write を発行する．また，clEnqRead では Read を発行する．本 API の実行前に同一メモリオブジェクトを利用しているイベントを実行している場合には，Join によりイベントの終了を待ち，CtrlDBuff によりダブルバッファを制御した後に Write/Read を発行する．

clEnqTask では Exe を発行する．本 API 実行時にまだ CMA のカーネルプログラムがロードされていなければ，ProgCMA を生成する．また，本 API の引数で与えられたカーネルオブジェクトから，メモリオブジェクトの利用状況を確認し，ローカルメモリ上のデータが利用可能な状態であれば Exe のみを生成する．他の CMA がローカルメモリのデータを操作している状況もあるため，必要に応じて Join，DMA，CtrlDBuff，Sync_wait，Sync_post を生成する．

clFinish では Join, Sync_host を発行する．本 API の実行までに CMA 制御部へ送った CMA 制御コマンドの終了を待ち，ホストとの同期を行うために Join と Sync_host コマンドを生成する．

clWaitForEvent では Sync_host を発行する．ホストとの同期をイベント単位で行うために Sync_host コマンドを生成する．

表 2　CMA 制御コマンドの仕様

CMA制御コマンド	機能	コマンド引数
ProgCMA	カーネルプログラムの書込み	主記憶上のアドレス
CtrlDBuff	ローカルメモリのバンクを切換え	なし
Write	主記憶のデータからCMAへデータのコピーを実行	主記憶上のアドレス，データサイズ
Read	CMAのLMから主記憶へデータのコピーを実行	主記憶上のアドレス，データサイズ
Exe	非同期にCMAの実行を開始	なし
DMA	非同期にDMAの起動	DMA先のCMA番号，データサイズ
Sync_post	CMA間同期を要求	同期相手のCMA番号
Sync_wait	CMA間同期の要求を待機	なし
Sync_host	ホストと同期するためにホストへ割込み	なし
Join	Join以前に実行されているすべてのコマンドの終了を待機	なし

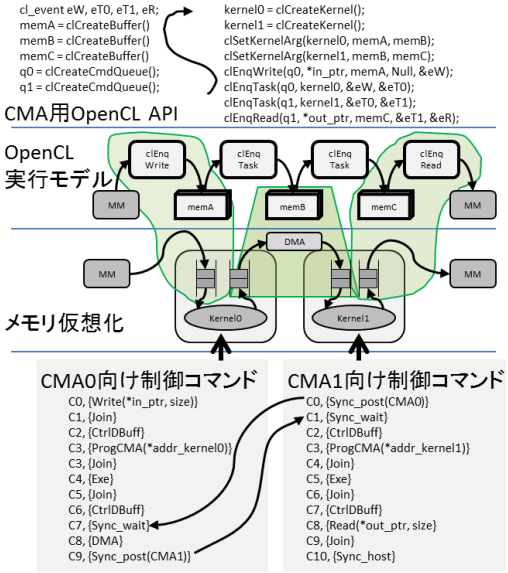

図 3　パイプライン処理時の例

図 3 に 2 個の CMA を使ったパイプライン処理の OpenCL プログラム例を示す．OpenCL 実行モデルへ CMA0，CMA1，ローカルメモリなどを対応づけ，図中に示すような各 CMA 向けに生成した CMA 制御コマンドを順に CMA 制御部で実行する．

4 まとめ

本論文では CMA アクセラレータ向けの OpenCL 環境と CMA 制御方式について述べた．今後は，本提案方式の実現とプログラム実行性能評価を行う．

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] 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, 電子情報通信学会技術研究報告CPSY2011-10, Vol.111, No.163, pp.7-12 (2011).
[2] http://www.khronos.org/opencl/
[3] Francis M David et al.: Context Switch Overheads for Linux on ARM Platforms, Proc. of ExpCS '07, Article No. 3 (2007).