Vol.2014-SLDM-165 No.13
Vol.2014-EMB-32 No.13
2014/3/15

組込みメニーコア向け OpenCL プログラムの機能検証環境

西山直樹[†1]　稗田拓路[†2]　谷口一徹[†1]　冨山宏之[†1]
†1 立命館大学　†2 九州大学

近年，数十個～数百個のコアを搭載したメニーコアアーキテクチャが注目されており，組込みシステムにおいてもメニーコアの利用が始まっている．我々は，組込みシステムを対象とした並列プログラミング環境 SMYLE OpenCL を開発している．SMYLE OpenCL は，カーネルや各種オブジェクトを静的に生成するため，リアルタイム性能に優れている．本研究では，SMYLE OpenCL プログラムの機能検証をホスト PC 上で実行するための環境を開発した．本環境は，Linux 上で動作し，OS レベルでの実行並列数の指定が可能であるという特徴がある．評価実験において，本環境で複数の OpenCL アプリケーションを様々な並列数で実行し，機能検証環境としての有用性を示した．

A Functional Verification Environment of OpenCL Programs for Many-core Embedded Systems

Naoki Nishiyama[†1]　Takuji Hieda[†2]　Ittetsu Taniguchi[†1]　Hiroyuki Tomiyama[†1]
†1 Ritsumeikan University　†2 Kyushu University

Many-core architecture, which has tens or hundreds of processor cores, attracts attention in recent days, even in embedded areas. We developed *SMYLE OpenCL*, which is a parallel programming framework for embedded systems. SMYLE OpenCL is superior to conventional OpenCL environment in terms of real-time performance since it creates kernels and other objects statically. This paper describes an environment for functional verification of SMYLE OpenCL programs on a host computer. This environment runs on Linux and users can decide the number of threads for executing OpenCL programs. In our experiments, several OpenCL applications are executed by a various number of threads, and the results show that the implemented environment is effective for functional verification of SMYLE OpenCL.

1. はじめに

半導体プロセスの微細化による動作速度の向上が限界に近づきつつある近年，汎用計算機において複数のプロセッサコアを並列駆動させるマルチコアアーキテクチャが主流となっている．組込みシステムにおいてもマルチコア化が進んでおり，システム全体を１つのチップ上に集積させた MPSoC（Multi-Processor System-on-Chip）が利用されている [1].

近年，組込みシステムは様々な製品に搭載されており，ロボットや自動車における画像処理，インターネット機器におけるインタラクティブ処理など複数の高度な処理を必要とするケースが増えている．しかし，並列化に対する取り組みとして行われているのは，スーパーコンピュータなどのハイパフォーマンス・コンピューティングで培った並列処理技術を応用した，単一のアプリケーションからの並列性抽出に着眼点を置いているものがほとんどである．

これらの背景から，様々なレベルの並列性を活用できる組込み汎用メニーコア SoC アーキテクチャおよびそのプログラミング環境の研究開発を目的として，九州大学，立命館大学，電気通信大学，株式会社フィックスターズ，株式会社トプスシステムズは「*SMYLE*（Scalable ManY-core for Low Energy computing）プロジェクト」を実施した [2].

SMYLE プロジェクトでの研究成果の１つとして，九州大学と電気通信大学が開発した汎用メニーコアアーキテクチャ *SMYLEref* がある．SMYLEref はアプリケーションやタスクを加速実行する仮想的なアクセラレータ VAM（Virtual Accelerator on Many-core）の概念を導入し，複数の VAM をメニーコア領域にマッピングすることで様々なレベルの並列性を活用することが可能である [3]．またそのプログラミング環境として，立命館大学が *SMYLE OpenCL* を開発した．SMYLE OpenCL では，ヘテロジニアス環境を利用した並列プログラミング言語である OpenCL を採用した．従来の OpenCL 環境にない，完全なタスク並列実行やアプリケーション並列実行のサポート，組込み SoC 向けのオーバヘッドの削減などを特徴としている．

SMYLE OpenCL 環境は SMYLEref アーキテクチャの FPGA プロトタイプ・システム上で動作する．この FPGA は高価であり，またデバック容易性（可観測性，可制御性）が低い．そこで本研究では，ソフトウェア開発者が SMYLEref アーキテクチャ向け OpenCL プログラムを開発／デバッグすることを目的として，SMYLE OpenCL／SMYLEref 機能検証環境を開発した．本環境の特徴として，Linux を搭載したホスト PC 上で動作すること，OS レベルでの実行並列数の指定が可能であることが挙げられる．

本稿の構成を以下に述べる．始めに 2 章で SMYLE プロジェクトについて述べる．次に 3 章において，開発した機

Vol.2014-SLDM-165 No.13
Vol.2014-EMB-32 No.13
2014/3/15

能検証環境について述べる．本環境の機能検証環境としての有効性を確認するための評価実験ならびに結果について 4 章で示し，最後に 5 章でまとめる．

2. SMYLE プロジェクト

SMYLE プロジェクトは，独立行政法人新エネルギー・産業技術総合開発機構（NEDO）の委託により，九州大学，立命館大学，電気通信大学，株式会社フィックスターズ，株式会社トプスシステムズが実施した．九州大学と電気通信大学が汎用メニーコア SoC である SMYLEref の開発，立命館大学がそのプログラミングフレームワーク SMYLE OpenCL の開発，株式会社トプスシステムズがビジュアル・コンピューティング向けメニーコア SMYLEvideo の開発，株式会社フィックスターズがメニーコア向けソフトウェア開発環境の構築を行った．本研究では，立命館大学が開発した SMYLE OpenCL の機能検証環境の実装を行った．

本章では，SMYLE プロジェクトで行われた研究の中でも，本研究と関わりの深い SMYLEref と SMYLE OpenCL について記述する．

2.1 SMYLEref

本節では，九州大学と電気通信大学が開発した組込みシステム向け汎用メニーコア SMYLEref について説明する．メニーコアにおける最大の課題は，いかにしてコア数の増加に対する性能スケーラビリティを確保するかである．しかしながら，多数のコアを活用するだけの並列性を単一アプリケーションから引き出すことは容易ではない．そこで SMYLEref では，図 2.1 に示す「仮想アクセラレータ（VAM）」の概念を導入し，メニーコア実行プラットフォームの構築を行った．

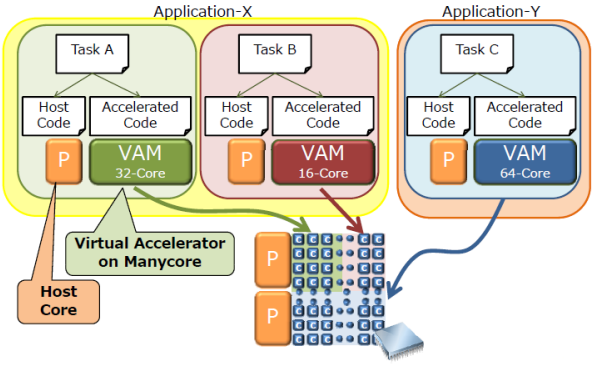

図 2.1　仮想アクセラレータ VAM の概念 [4]

SMYLEref はシンプルな数個のプロセッサをバスで結合したクラスタと呼ぶブロックを，2 次元メッシュのオンチップネットワーク（NoC）で結合したアーキテクチャである． 図 2.2 に示すように，SMYLEref はシンプルな数個のプロセッサコアをバスで結合しクラスタを構成する．各 VAM は 1 個以上のクラスタにマッピングされる．

図 2.2　クラスタ構成と VAM マッピングの例 [4]

2.2 SMYLE OpenCL

本節では，立命館大学で開発された SMYLEref 向け OpenCL プログラミングフレームワーク SMYLE OpenCL について記述する．SMYLEref 用のプログラミンモデルとして，OpenCL を採用した．OpenCL を採用した主な理由は以下のとおりである [4].

- 仕様がオープンかつロイヤルティーフリーである
- 特定のハードウェアプラットフォームに依存していない
- C 言語をベースとしており習得が容易である
- データ並列実行だけでなく，タスク並列実行もサポートしている
- 標準化されたプログラミングモデル／言語であり，本研究の成果を広く普及させることができる

SMYLEref アーキテクチャは多くのコアを有しているが，図 2.3 に示すように SMYLE OpenCL では，1 つのコアをホストとして使用し，そのホスト上で Linux オペレーティングシステムが動作していることを想定している．残りのコアをデバイスとして使用する．

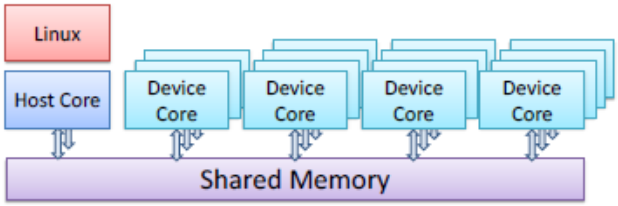

図 2.3　SMYLE OpenCL におけるコアの使用例 [2]

Vol.2014-SLDM-165 No.13
Vol.2014-EMB-32 No.13
2014/3/15

SMYLE OpenCL では，図 2.4 のように複数のアプリケーションを空間的に並列に実行することができる．各アプリケーションの内部では，データ並列実行，タスク並列実行あるいはその両方が行われる．各アプリケーションが使用するコア数は，ハードウェアアーキテクチャのクラスタ構造の境界を越えて，アプリケーションにコアが割り当てられる．

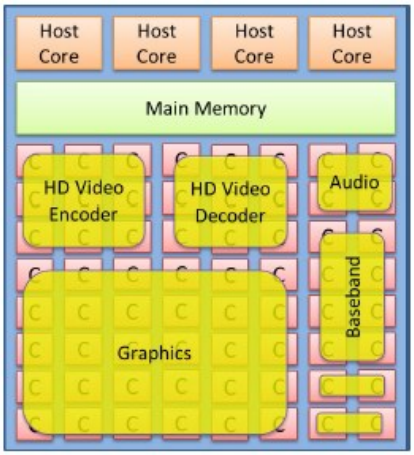

図 2.4　SMYLE OpenCL における並列実行 [2]

3. SMYLE OpenCL の機能検証環境

本章では開発した SMYLE OpenCL の機能検証環境について述べる．本環境について説明するにあたって，3.1 節にて OpenCL のプログラミングモデルについて記述する．その後，本環境の構成とメニーコアシミュレーションについて述べる．

3.1 OpenCL

OpenCL アプリケーションは，ホスト上で動作するホストプログラムと，OpenCL デバイス上で動作するカーネルプログラムで構成される．カーネルプログラムに並列処理部分，ホストプログラムに並列処理の制御部分が記述される．カーネルプログラムを実行する OpenCL デバイスは複数の Processing Element (PE)を持ち，カーネルプログラムを並列実行する．OpenCL では，この並列実行の処理単位をワークアイテムと呼び，このワークアイテムをまとめたものをワークグループと呼ぶ．OpenCL ではこのワークアイテム数を指定することによってアプリケーションレベルでの並列数の指定を行う [5].

3.2 本環境の構成

本環境は，図 3.に示すように，OpenCL のランタイムライブラリと GCC から構成される．実行環境は x86 プロセッサに対応している．また OS は Linux（CentOS，Ubuntu）での実行を確認している．

従来の OpenCL 環境（Intel, NVIDIA など）では，ホストプログラムの実行中にカーネルプログラムのコンパイルを行う（オンラインコンパイル）ことができるが，本環境では，カーネルプログラムをホストプログラム実行前（ホストプログラムコンパイル時）にコンパイルするオフラインコンパイルのみに対応している．これは，本環境は組込み環境をターゲットとしており，実行時のオーバヘッドをできる限り削減するためである．

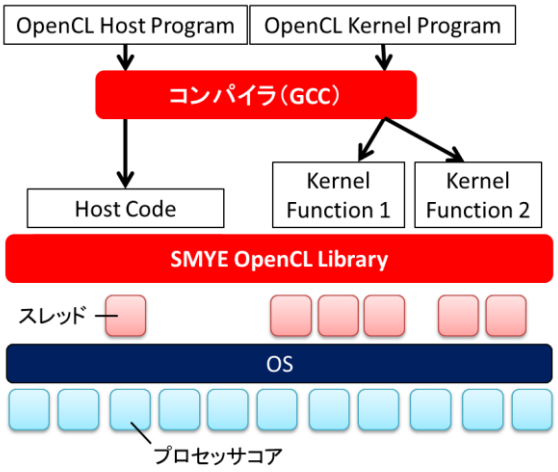

図 3.1　本環境でのプログラム処理系の構成

3.3 メニーコアシミュレーション

本環境は，POSIX スレッド（pthread）によって実装されている．POSIX スレッドは，代表的な非同期処理の仕組みであるスレッドモデルを利用するための API 群である．pthread を用いて本環境を実装した理由は，個々のデバイスコアを pthread でシミュレーションするためである．図 3. に本環境の構成を示す. デバイスコアが 64 コア存在する場合，本環境内部で 64 個のスレッドが生成される．各スレッドは独立に動作し，OpenCL プログラムのデータ並列実行とタスク並列の両方をサポートしている．

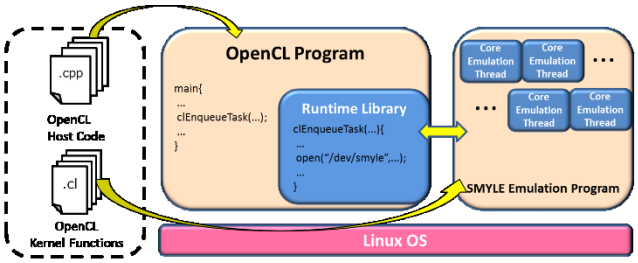

図 3.2　本機能検証環境の構成

先の節で記述した通り OpenCL では，ワークアイテムによってソフトウェアレベルでの実行並列数の指定を行うことが可能である．従来の OpenCL 環境では，このワークアイテムで指定された実行並列数は，各環境の処理系が実行環境に最適化して並列実行を行う [6] [7]．そのためハードウェアレベルでの実行並列数は，ソフトウェア開発者からは分からないようになっている．これに対し本環境では，

ワークアイテム数で指定した並列数でスレッドの生成を行う．これによってソフトウェア開発者が，OS レベルでの実行並列数の指定することができる．

本環境では，簡単化のため，ワークグループという概念を実装していない．そのため，OpenCL ではワークグループ間でのバリア同期が可能であるが，本環境では使用できない．

先の項で説明した通り，本環境ではグローバルワークアイテム数がそのまま生成するスレッド数として扱われる（図 3.1 参照）．したがって後に行う実験のように，実行スレッド数を変化させながら同じプログラムを実行したい場合，プログラムに工夫が必要となる．

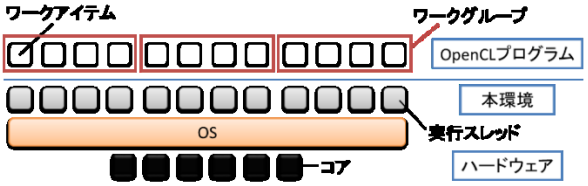

図 3.1　本環境でのワークアイテムの扱い

4. 実験

本章では，開発した OpenCL 環境にて OpenCL アプリケーションが実行可能か，機能検証環境として有用性があるかを評価する実験について述べる．

開発した OpenCL 環境の評価実験を行う環境を表 4.1 に示す．

表 4.1　実験環境

CPU	Intel Xeon 2.0GHz (8 cores, 16 threads) × 2
Memory	DDR3-1333 128GB
OS	CentOS 64bit
Kernel	2.6.32-358

本環境は，x86 アーキテクチャのプロセッサを搭載するホスト PC 上で SMYLE OpenCL で記述された OpenCL プログラムの実行が可能である．それを示すために，下記の 4 つの OpenCL アプリケーションを実行する．また Gaussian（ガウシアンフィルタ）については，入力画像のサイズが 3000×2000 と 8000×6000 の場合の 2 パターンを行った．

- Blackscholes：　ブラック・ショールズ方程式（コール・プットオプションの計算）
- Runlength：　ランレングス圧縮
- Grayscale：　画像のグレースケール化
- Gaussian：　画像のガウシアンフィルタ適用

これらの 4 つのアプリケーションにより評価実験を行う．

4.1 アプリケーション単体実行

環境の特徴の 1 つとして，SMYLE OpenCL で記述した並列数を，OS レベルで指定することが可能である．これを示すため，同じプログラムに対して実行並列数（スレッド数）を 1～16384（$2^{0\sim14}$）に変化させて実行時間を計測する．この結果を見るにあたって，実行並列数によって全体の計算量が変わっていないことに注意が必要である．

比較対象として，従来の OpenCL 環境である FOXC との比較を行った．カーネルプログラムのコンパイルは，本環境と同じオフラインコンパイルで実行時間を計測した．

本環境で，4 つのサンプルプログラムを実行した結果を図 4.1～図 4.5 に示す．

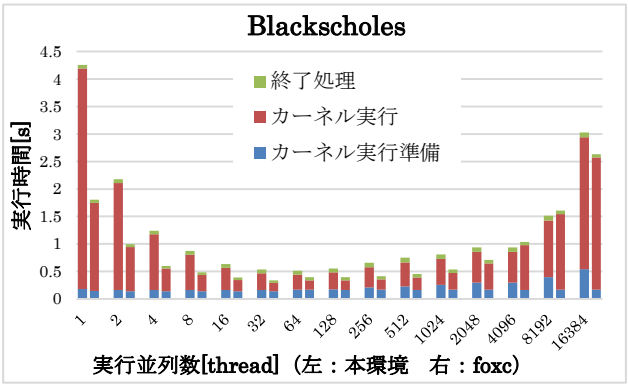

図 4.1　Blackschole の実行結果

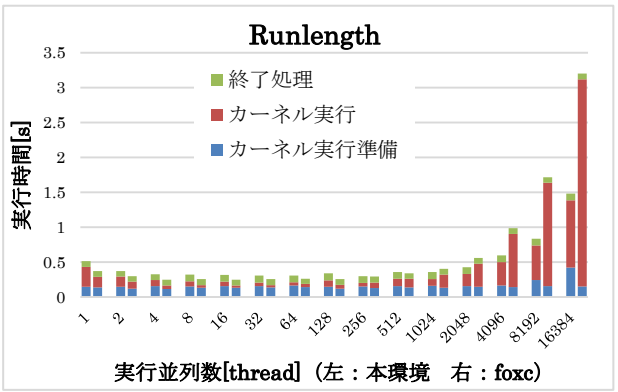

図 4.2　Runlength の実行結果

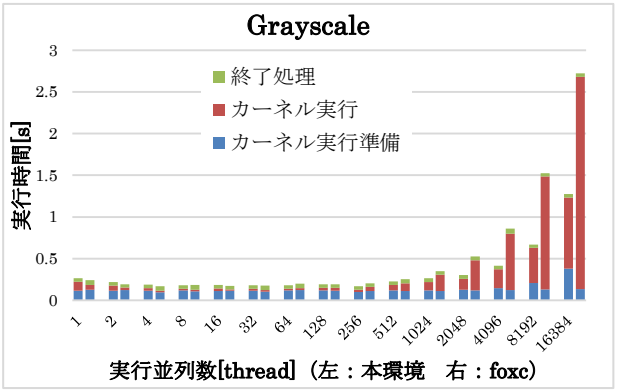

図 4.3　Grayscale の実行結果

Vol.2014-SLDM-165 No.13
Vol.2014-EMB-32 No.13
2014/3/15

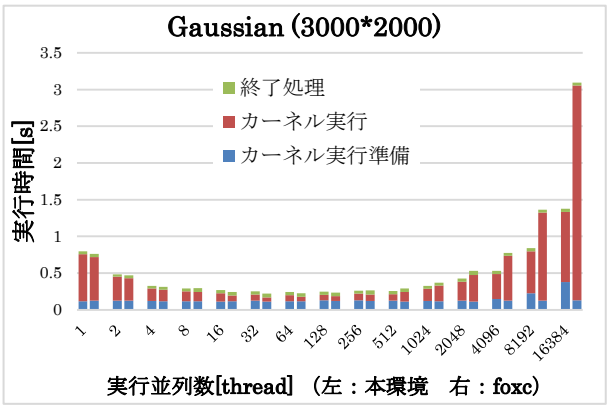

図 4.4　Gaussian（3000*2000）の実行結果

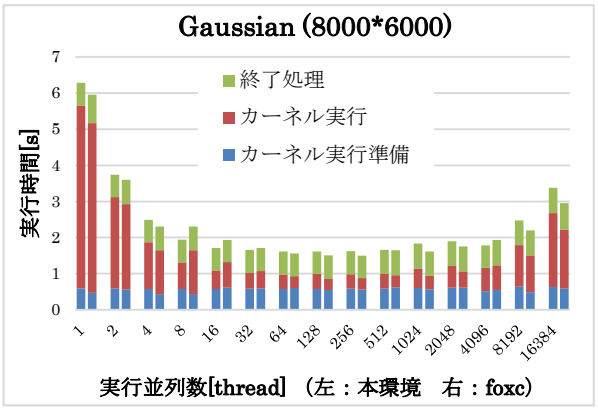

図 4.5　Gaussian（8000*6000）の実行結果

まずそれぞれのアプリケーションについて，従来環境で実行した場合の実行結果と本環境で実行した場合の実行結果が同じであることを確認した．そのことから，本環境が機能検証環境として有用であることを示した．

次に実行並列数と実行時間の関係に着目すると，アプリケーションによって最も実行時間が少ない時の実行並列数が異なることが分かる．これにより，アプリケーションを FPGA で実行する前に，ある程度ではあるがそのアプリケーションにどのくらいプロセッサリソースを割けばよいか予測することができる．例えば，Grayscale では実行並列数が 1～256 の時，ほとんど実行時間に変化がない．それに対し Gaussian では実行並列数が 16～128 の時，他に比べ高速化している．リソースが 32 スレッドで，Grayscale と Gaussian を同時に実行する場合，Grayscale に 1～8 スレッド，Gaussian に 24～31 スレッド割り当てて実行すれば良いという予測ができる．

従来環境との比較では，総合的に見ると本環境での実行はやや遅いが，ほぼ同等といえる水準である．従来環境よりやや遅くなってしまった原因としては，本環境ではコンパイラに GCC を用い，標準のライブラリを用いたのに対し．従来環境の FOXC では Intel プロセッサに対して最適化されたライブラリを使用していることが考えられる．

4.2　タスク／アプリケーション並列実行

SMYLE OpenCL はタスク／アプリケーション並列実行をサポートしている．本環境でも，OS レベルでのタスク／アプリケーション並列実行を行うことができる．これを示すため，Grayscale プログラムと Gaussian プログラムのタスク／アプリケーション並列実行を行った．それぞれの並列実行で使用するスレッド数の合計を 32 に固定して実験を行った．タスク並列実行は OpenCL の機能であるコマンドキューのアウトオブオーダー実行を用いる．アプリケーション並列実行では，OS の並列実行機能を用いて行う．

本環境で，2 つの OepnCL プログラムをタスク／アプリケーション並列実行を行った結果と評価を示す．図 4.6 に Grayscale/Gaussian のタスク並列実行の結果を示す．横軸に示す値は，「1_31」の場合なら Grayscale に 1 スレッド，Gaussian に 31 スレッドを使用し実行を行っていることを示す．

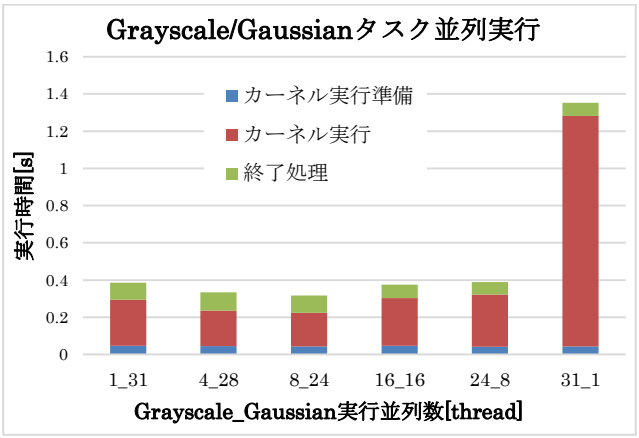

図 4.6　Grayscale/Gaussian のタスク並列実行の結果

図 4.7 に Grayscale/Gaussian のアプリケーション並列実行の結果を示す．

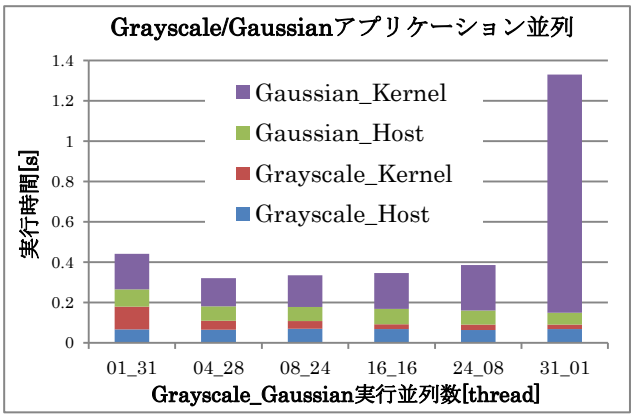

図 4.7　Grayscale/Gaussian のアプリケーション並列実行の結果

タスク並列とアプリケーション並列の結果を比べるとほとんど差はなかった．この 2 つの実行の差は，ホストプログラムの実行にスレッドをいくつ使うか（タスク並列なら 1 スレッド，アプリケーション並列なら 2 スレッド）である. もともと Grayscale/Gaussian のホストプログラムの実行時間はカーネルプログラムの実行時間に比べ，非常に短いため差が生まれなかったと推測できる．

先の節で述べた通り，Grayscale に 4～16 スレッド，Gaussian に 16～28 スレッド使用する時の実行時間が最も短くなった．先で述べたように，それぞれのアプリケーションで最適な並列実行数が違うという理由がある．Grayscale の実行時間は約 0.18～0.2 秒，Gaussian の実行時間は約 0.3～0.35 秒ほどかかる．それぞれのアプリケーションで実行時間にばらつきがあり，どのような実行並列数の配分をすれば，並列実行時に実行時間が最も短くなるかを事前に予測するのは難しい．本環境を利用することにより，ある程度その予測を立てることができ，検証環境としての有用性を示した．

5. まとめ

本稿では，組込みシステムを対象とした並列プログラミング環境 SMYLE OpenCL プログラムの機能検証をホスト PC 上で実行するための環境の開発について述べた．SMYLE OpenCL には，組込みメニーコア環境でプログラムを効率よく実行を行うための実行モデル，メモリモデル，プログラミングモデルがあり，それらをホスト PC 上で正確に検証できるよう実装を行った．また本環境をスレッドプログラミングで実装し，OS レベルでの実行並列数の指定を可能とした．これにより，OpenCL プログラムをメニーコア環境で実行する時，どの程度の並列数で実行すれば高速に実行できるかを予測することが可能である．本環境で，複数のアプリケーションを様々な実行並列数でアプリケーション単体実行，タスク／アプリケーション実行を行うことにより，本環境の機能検証環境としての有用性を示した．

謝辞

本研究は一部，独立行政法人新エネルギー・産業技術総合開発機構（NEDO）の委託により実施した．本研究を実施するに当たり，有益なご助言とデータを頂いた元立命館大学研究員の江谷典子博士，九州大学の井上弘士准教授，平尾智也氏，曽我武史氏，元電気通信大学（現東京大学）の近藤正章准教授，（株）フィックスターズ，（株）トプスシステムズの諸氏に感謝する．

参考文献

[1] N. Gabriela, O. Ian, P. Christian, *Design Teechnology for Heterogeneous Embedded Systems*, Springer, 2012.

[2] 独立行政法人新エネルギー・産業技術総合開発機構, *「極低電力回路・システム技術開発（グリーン IT プロジェクト）」研究開発項目⑦「低消費電力メニーコア用アーキテクチャとコンパイラ技術」*, 平成 25 年 5 月.

[3] グェン チュオン ソン, レイ ジャオ, 近藤正章, 平尾智也 , 井上弘士, “FPGA を用いたメニーコア・アーキテクチャ SMYLEref の評価環境の構築,” *情報処理学会研究報告,* Vol.2012-ARC-198, No.15, 2012 年 1 月.

[4] K. Inoue and M. Kondo, “SMYLE: Scalable Many-core for Low-Energy computing,” *12th International Forum on Embedded MPSoC and Multicore,* July 2012.

[5] A. Munshi, *The OpenCL Specification*, November 2012. Available: http://www.khronos.org/registry/cl/specs/opencl-1.2.pdf.

[6] Intel, *Intel® SDK for OpenCL* Applications 2013 | Intel® Developer Zone*, Available: http://software.intel.com/en-us/vcsource/tools/opencl-sdk., Access: January 2014.

[7] NVIDIA, *OpenCL | NVIDIA Developer Zone*,Available: http://developer.nvidia.com/opencl, Access: January 2014.