Vol.2013-ARC-203 No.8
2013/2/1

SMYLE OpenCLの実装と128コア上での評価実験

稗田 拓路[1]　江谷 典子[1]　西山 直樹[1]　谷口 一徹[1]　冨山 宏之[1]　グェン チュオン ソン[2]
近藤 正章[2]　曽我 武史[3]　平尾 智也[3]　井上 弘士[3]

概要：メニーコアアーキテクチャは，数十から数百個の低性能ではあるが小面積かつ低消費電力なコアを並列に駆動させることで，極めて高い並列処理性能を得ることが可能である．このことからメニーコアアーキテクチャは汎用計算機のみならず，組み込みシステムにおいても実用的な選択肢となりつつある．そこで本稿では，独立行政法人新エネルギー・産業技術総合開発機構 (NEDO) のプログラム『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)』において，組込みシステム向けメニーコアアーキテクチャを実現することを目指した『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』プロジェクトの研究により開発された，FPGA を用いたメニーコアアーキテクチャ SMYLEref の評価環境の上に，SMYLE アーキテクチャ向け OpenCL 環境である SMYLE OpenCL の実装を行った．評価実験において，FPGA ボード上に構築された 128 コア搭載の SMYLEref アーキテクチャ上においてベンチマークプログラムを実行し，実装された SMYLE OpenCL の有効性について示す．

SMYLE OpenCL Implementation and Evaluations on 128 Cores

Takuji Hieda[1]　Noriko Etani[1]　Naoki Nishiyama[1]　Ittetsu Taniguchi[1]　Hiroyuki Tomiyama[1]
Nguyen Truong Son[2]　Masahiko Kondo[2]　Takeshi Soga[3]　Tomoya Hirao[3]　Koji Inoue[3]

Abstract: Many-core architecture can achieve highly parallel processing performance, which is derived from tens or hundreds of low performance, small area, and low power cores to work in parallel. This advantage makes many-core a practical choice for embedded systems not only general computing systems. In a program of "Extremely Low-power Circuits and Systems (Green IT Project)" sponsored by New Energy and Industrial Technology Development Organization (NEDO), an environment using FPGA in order to evaluate SMYLEref architecture for many-core processor was developed as result of the research by a project of "many-core architecture for low energy consumption and its compiler technology". This paper describes SMYLE OpenCL, an OpenCL implementation for SMYLEref many-core architecture on the evaluation environment. In experiments, a number of benchmark programs are executed on SMYLEref architecture with 128 cores based on the FPGA evaluation environment to verify effectiveness of SMYLE OpenCL.

1. はじめに

画像処理用に設計された GPU (Graphics Processing Unit) を汎用演算に利用する GPGPU (General-purpose computing on GPUs) に代表されるように，数十から数百個の演算コアを集積したメニーコアアーキテクチャ上で，アプリケーションを並列処理する手法が一般的に普及し始めている．低性能ではあるが小面積かつ低消費電力なコアを多数搭載したメニーコアアーキテクチャは，極めて高い並列処理性能を得ることが可能であること，また処理能力がそれほど必要とされていない場合については，少量のコアのみを演算に使用することで消費電力を削減することが可能であることから，メニーコアアーキテクチャは汎用計算機のみならず，組み込みシステムにおいても実用的な選

[1] 立命館大学理工学部
College of Science and Engineering, Ritsumeikan University
[2] 電気通信大学大学院情報システム学研究科
Graduate School of Information Systems, The University of Electro-Communications
[3] 九州大学大学院システム情報科学研究院
Department of Advanced Information Technology, Kyushu University

択肢となりつつある．メニーコア環境を効果的に使用するためには，状況に応じて実行するタスクへのコア割り当てを最適化することが必要であるため，高性能・低消費電力なメニーコアシステムの実現を目指し，メニーコアアーキテクチャを有効利用するための研究が組み込み分野において盛んに行われている．

本稿では，独立行政法人新エネルギー・産業技術総合開発機構 (NEDO) のプログラム『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）』において，組込みシステム向けメニーコアアーキテクチャを実現することを目指した『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』プロジェクトの研究により開発された，FPGA を用いたメニーコアアーキテクチャ SMYLEref の評価環境の上に，SMYLE アーキテクチャ向け OpenCL 環境である SMYLE OpenCL を実装した．実験では，FPGA ボードを 2 次元メッシュ状に接続して構築された 128 コア並列実行環境において複数のベンチマークプログラムを実行し，提案するメニーコア環境の有効性について示す．

本稿の構成を以下に述べる．まず 2 節で関連研究について述べる．次に 3 節で SMYLE メニーコア環境のシステム構成について説明する．評価実験とその結果について 4 節で示し，5 節で本稿のまとめを述べる．

2. 関連研究

数個から十数個のコアで構成されたマルチコアアーキテクチャと比較して，メニーコアアーキテクチャを搭載したシステムでは，大量のコアを並列駆動させることで高い処理能力を実現することが可能である．メニーコアアーキテクチャの代表例となっている GPU を用いた GPGPU に関しては，既に様々な研究がなされている [1]．また，CUDA に代表されるような GPGPU 向けの並列コンピューティング言語も一般的に用いられるようになっている [2]．しかし，これらの並列コンピューティング言語は GPU ベンダから提供されており，汎用的な言語ではないという欠点がある．

OpenCL は, Khronos グループが策定した並列コンピューティング言語である [3]．OpenCL ではメニーコアアーキテクチャを OpenCL デバイスとしてモデル化している．OpenCL デバイスは複数の演算ユニットで構成され，それぞれの演算ユニットがプロセッシングエレメント (PE) を複数持ち，データ並列ならびにタスク並列プログラミングモデルをサポートしている．このため，組込みシステムから HPC (High Performance Computing) まで幅広い用途に対応することができる．CUDA が nVidia 製 GPU のみを対象としているのに対し，OpenCL は様々なアーキテクチャを想定して設計されている．ただし，CUDA と OpenCL の言語構造には多くの類似点がある．また，仕様が Khronos グループから公開されており，様々なプラットフォームへの移植が可能である．そのため本稿でも，メニーコア環境下で並列アプリケーションを記述するプログラミング言語として OpenCL を採用している．

既存の OpenCL 実装として，nVidia や ATI が GPU 向け OpenCL 環境を，また，Intel が Core プロセッサ向け OpenCL 環境を提供している．しかし，GPU は最小の処理単位が PE ではなく複数の PE をまとめたグループ単位であるため 1 つの PE だけを利用することができない [4]．また Intel Core プロセッサ向け OpenCL では，Core プロセッサに搭載されているコア数がメニーコアと呼ぶには不十分な数であることから，既存の OpenCL 実装ではメニーコア環境としては不完全な面が存在する．今回実装した SMYLE OpenCL は，以前に我々のグループで実装した Linux ホスト環境下で動作するシミュレーション環境に基づいて実装されている [5]．このシミュレーション環境は，メニーコア部のコア 1 つをそれぞれ 1 つのスレッドとして実装し，OpenCL のカーネル関数部の実行をシミュレートする．各スレッドの動作は独立しているため，データ並列実行だけではなくタスク並列実行を行うことが可能である．また，シミュレータ内部のコア数を自由に変更できるため，メニーコアアーキテクチャの設計にも利用可能である．

OpenCL では，OpenCL C 言語で規定されている組込み関数をカーネル関数内部で使用できる．これらの組込み関数は三角関数などの数学関数が中心であり，基本的な四則演算能力を有することが PE に求められる．SMYLE OpenCL におけるカーネル側組み込み関数に関しても現在実装が進められている [6]．

3. システム構成

本節では，SMYLE OpenCL ならびに SMYLEref を用いて FPGA ボード上に構築した 128 コア並列実行環境について説明する．まずはじめに，SMYLE OpenCL の想定するメニーコアアーキテクチャモデルについて述べる．次に，今回使用したメニーコアアーキテクチャである SMYLEref について説明する．その後で SMYLE OpenCL の実装について述べ，最後に FPGA 評価ボード上に構築した 128 コア並列実行環境について説明する．

3.1 メニーコアアーキテクチャモデル

SMYLE OpenCL の想定するアーキテクチャモデルを図 1 に示す．1 つのホストプロセッサとコアアレイ部で構成されており，ホストプロセッサは OS の実行や OpenCL API を用いて記述されたホスト側コードの実行などを担当する．コアアレイは複数の PE で構成されており，ホストプロセッサの指令に従ってカーネル関数の実行を担当する．ホストプロセッサとコアアレイ間でのデータのやり取りは，共有メモリを介して行うことを想定している．各コアは独立して動作するものとし，タスク並列処理とデータ

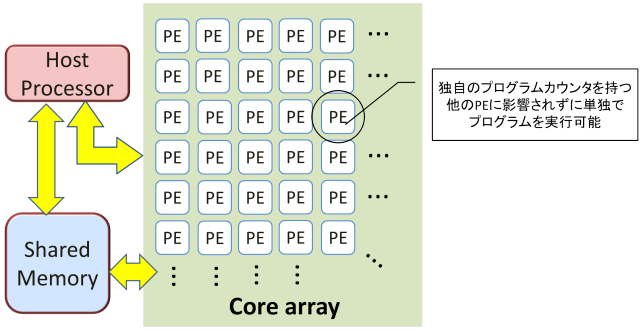

図 1 メニーコアアーキテクチャモデル
Fig. 1 Many-core Architecture Model

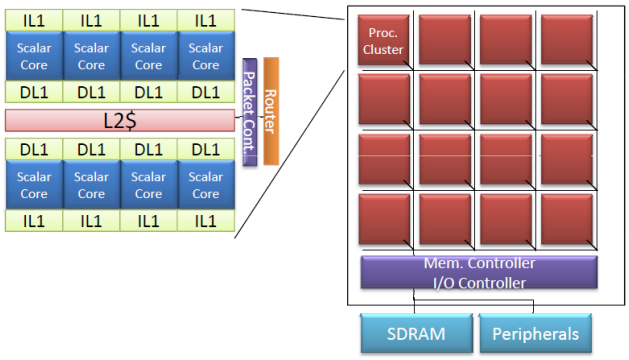

図 2 SMYLEref アーキテクチャ概要
Fig. 2 Overview of the SMYLEref Architecture

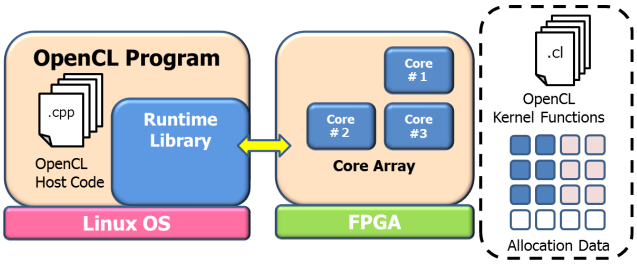

図 3 SMYLE OpenCL 実装概略
Fig. 3 Implementation of SMYLE OpenCL

並列処理を行うことが可能であるとする．

図 1 で示したメニーコアアーキテクチャモデルでは，ホストプロセッサや PE に対する具体的なアーキテクチャは規定していないため，システム設計者が必要なアプリケーションに応じてホストプロセッサや PE の性能を決定することが可能である．ただし，ホストプロセッサ上で OpenCL プログラムを動作させるためには，ホストプロセッサ上で OpenCL ランタイムライブラリが動作することが必要となる．ランタイムライブラリは内部でスレッドライブラリを使用するため，スレッドライブラリが動作する環境であることがホストプロセッサ側に求められる．

3.2 SMYLEref

SMYLEref は，複数のコアクラスタを NoC (Network on Chip) で 2 次元メッシュ状に結合したメニーコアアーキテクチャである．1 つのコアクラスタはシンプルなプロセッサコアを複数個集積し，バスで結合したものである．SMYLEref のアーキテクチャ概要を図 2 に示す．図 2 では，1 つのクラスタ内に存在するコア数を 8 としている．また本稿では，各プロセッサコアのアーキテクチャとして MIPS R3000 アーキテクチャをベースとしたシンプルなプロセッサコアである geyser を採用している [7], [8]．geyser コアはそれぞれ 8KB の L1 命令キャッシュと L1 データキャッシュ，ならびに 16 エントリの TLB を持つ．また，各クラスタごとに共有 L2 キャッシュを持つ [8]．

3.3 SMYLE OpenCL

SMYLE OpenCL は，並列コンピューティング言語の 1 つである OpenCL を SMYLEref アーキテクチャ向けに実装した OpenCL 環境である．SMYLE OpenCL の概要を図 3 に示す．SMYLE OpenCL はホストプロセッサの役割を担う geyser コア上に構築された Linux 環境下で動作し，Linux を動作させていない他のコアを OpenCL デバイスコアとみなして並列実行処理を行う [9]．

SMYLE OpenCL の特徴として，カーネル関数を実行する際のメニーコアへのタスクマッピングをあらかじめ静的に決定しておくことが可能である点が挙げられる．OpenCL で規定されている API では，カーネル関数を何個のタスクに分割するかを指定することが可能であるが，あるカーネル関数を実行する際に何個の PE を割り当てるか，ということは処理系の実装によって定められるため，アプリケーションプログラマが決定することはできない．SMYLE OpenCL 処理系では，あらかじめ実行したいカーネル関数に対して何個のコアを割り当てるかを決めておくことで，それぞれのカーネル関数ごとに必要な分だけの PE を割り当てることで，カーネル関数の並列実行が可能である．また，実行コンテキストの準備や OpenCL デバイス側に処理を指示するためのコマンドキューの作成など，既存の OpenCL ではプログラム実行時に動的に生成する必要がある処理を前もって静的に準備しておくことで，実行時間の削減を可能としている．

3.4 FPGA 評価ボードによる 128 コア並列実行環境の実装

今回使用した 128 コア並列実行環境は，複数の FPGA ボードをクラスタ接続することで，1 つのメニーコア環境として構築されている．今回構築された 128 コア並列実行環境では，Xilinx 社製 FPGA チップである Virtex-6 を搭載する ML605 評価ボードを 16 枚用いて 1 つのメニーコアプラットフォームとして利用している．また，geyser 上に Linux OS が移植されているため，128 コアのうち 1 つの geyser コアをホストプロセッサとして使用する．この

表 1　ML605 評価ボードの主な仕様
Table 1　Specifications of ML605 Evaluation Board

FPGA デバイス	Virtex-6 XC6VLX240T-1FFG1156
SDRAM	DDR3 SODIMM(512MB)
搭載 IO ポート	UART，USB，DVI 出力，CF，SMA 等
クロック入力	200MHz & 66MHz ソケットオシレータ

表 2　Virtex-6 チップの主な仕様
Table 2　Specifications of Virtex-6

テクノロジ	65nm CMOS, 1.0V
Logic Cells	241,152
CLB Slices	37,680
Block RAM	14,975 Kbit
ユーザー I/O 数	720

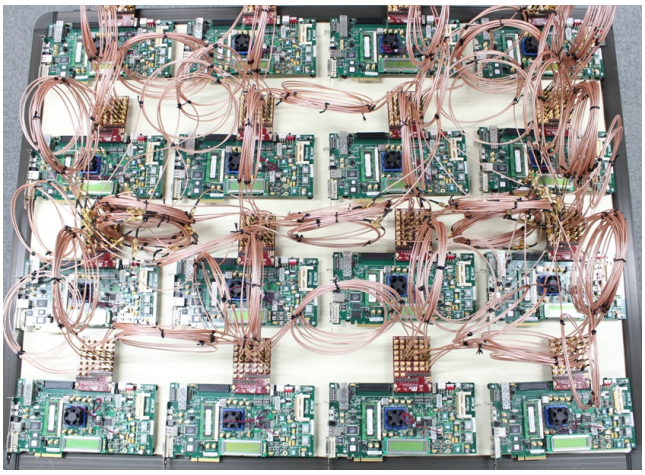

図 4　128 コア並列実行環境の外観
Fig. 4　Appearance of the 128 Cores Environment

ため，実際には OpenCL デバイスコアとして使用可能な最大コア数は 127 コアとなる．ML605 評価ボードならびに Virtex-6 チップの仕様をそれぞれ表 1, 2 に示す．

今回構築した 128 コア並列実行環境では，各 FPGA ボード上の Virtex-6 上に 8 つの geyser コアを含むクラスタを構築し，FPGA ボード 1 枚で 1 クラスタを構成する．16 枚の FPGA ボードは 2 次元メッシュ状にネットワーク接続され，図 4 に 128 コア並列実行環境の写真を示す．16 枚の FPGA ボードがケーブルによってメッシュ状に接続されている様子が写真に見て取れる．

4. 評価実験

実験では SMYLE OpenCL の有効性を確認するために，OpenCL で記述された複数のベンチマークプログラムについて，各タスクの使用するコア数を 1 から 127 コアまで変化させ，それぞれの実行時間について計測を行った．今回の実験にあたって，128 コア並列実行環境を構成する FPGA ボード各部の周波数は以下の設定となっている [8]．

- Geyser コア: 10MHz
- クラスタ内部バス・ルータ・ペリフェラルバス (PLB): 5MHz
- DDR3-SDRAM: 100MHz

実験で使用した OpenCL ベンチマークプログラムは以下の 6 つである．

- `backProjection`：入力画像に対して逆投影法を適用するプログラム
- `blackScholes`：ブラック－ショールズ方程式の計算を行うプログラム
- `gaussian`：入力画像に対してガウシアンフィルタを適用するプログラム
- `grayScale`：入力画像に対してグレイスケール変換を適用するプログラム
- `linearsearch`：配列中のデータを線形検索するプログラム
- `runLength`：入力データに対してランレングス圧縮を行い，結果をファイルに出力するプログラム

`backProjection` は，投影された画像から元の画像を求める逆投影法を適用するプログラムである．入力データは白黒 2 値画像にあたるテキストファイルで，出力結果もテキストファイルとして出力される．`blackScholes` は，ブラック - ショールズ方程式の計算を行うプログラムである．複数の入力に対してカーネル関数を並列実行する．カーネル関数内で数学関数を使用しているため，カーネル側関数が動作するかの確認を行うことができる．`gaussian` と `grayScale` はそれぞれ入力画像に対してガウシアンフィルタならびにグレイスケール変換を適用し，結果をファイルに出力する．入力画像ならびに出力画像は PPM 形式を扱う．`linearsearch` は，1 次元配列に対して線形探索を行うプログラムである．0 クリアされた巨大な配列中である要素の値を 1 にセットし，セットされた場所を探索する．検索する範囲を分割することで，探索処理を並列実行する．あらかじめ指定された回数だけ探索を実行し，その都度セットされる要素の場所を変更する．場所の指定は乱数によっておこなわれる．`runLength` は，入力されたファイルに対してキャラクタ単位でランレングス圧縮を行い，結果をファイルに出力する．任意のファイルを入力としてとることが可能である．

実験では，各ベンチマークプログラム内でいくつかのポイントにおいて `gettimeofday()` システムコールで経過時間を測定し，その差分をとることで各処理部の実行時間測定を行う．実行時間を測定する処理の分割は以下の通りである．

- input data：入力データを取得する部分
- preparation：コンテキストの設定など，OpenCL の実行環境を準備する部分
- run：デバイスコア上でカーネル関数を実行し，結果をホストに返す部分

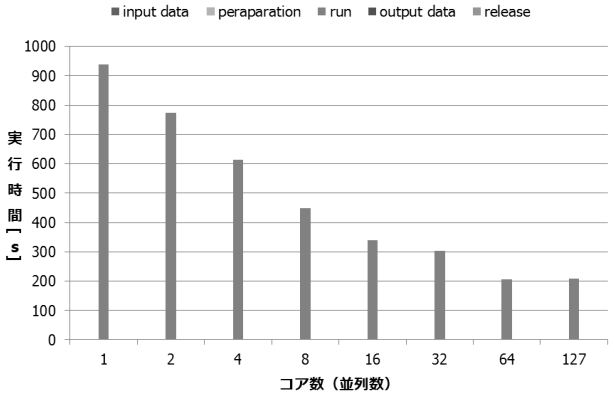

図 5 backProjection の実行時間
Fig. 5 Execution Time of backProjection

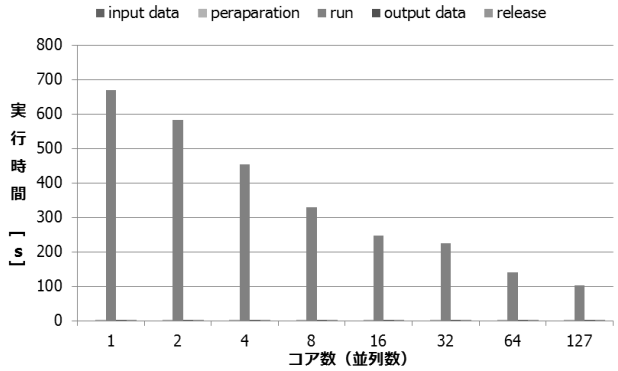

図 6 blackScholes の実行時間
Fig. 6 Execution Time of blackScholes

- output data : 結果を出力する部分
- release : コンテキストやメモリの解放など後処理を行う部分

ただし，`linearsearch` に関しては上記の測定部分に加えて，配列の初期化を行う init 部を測定する．

4.1 実験結果

各プログラムの実行時間を測定したグラフを図 5 から 10 に示す．

それぞれのグラフから分かる通り，`backProjection` や `blackScholes`，`gaussian` ではコア数と実行時間は理想的なトレードオフ関係が結果に表れていることが分かる．また `grayScale` でも，64 コアの時と 127 コアの時とで実行時間の逆転が生じているが，トレードオフ関係が表れている．逆に `runLength` に関しては，コア数が最大の 127 の時に並列演算部の実行時間が最大となってしまっている．これは，今回入力に用いたファイルサイズが 550Byte と小さなものだったため，分割して圧縮作業を行う際に細かく分割され過ぎた結果，それぞれのコアで得られた結果をマージする際に逆に時間がかかったものと考えられる．また，`linearsearch` に関しては，1 コアの時の方が 2 コアや 4 コアの時より早く処理が完了している．これに関しては，メモリを分割して線形探索を行った結果，メモリアクセスの競合が起こり待ち合わせ時間がかかったためと推測される．8 コア以上に関しては，並列処理のメリットがアクセス競合のデメリットを上回ったものと考えられる．

いずれの場合においても，並列実行部以外の処理時間はあらかじめ静的に行われる結果コア数に限らず一定であると言える．これは図 10 の `runLength` の結果を見ると分かりやすい．これにより，SMYLE OpenCL によって組み込

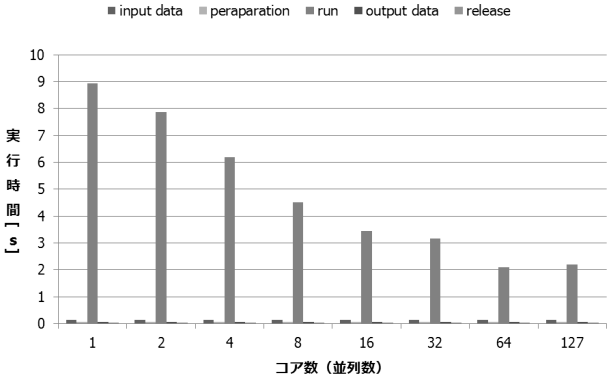

図 7 gaussian の実行時間
Fig. 7 Execution Time of gaussian

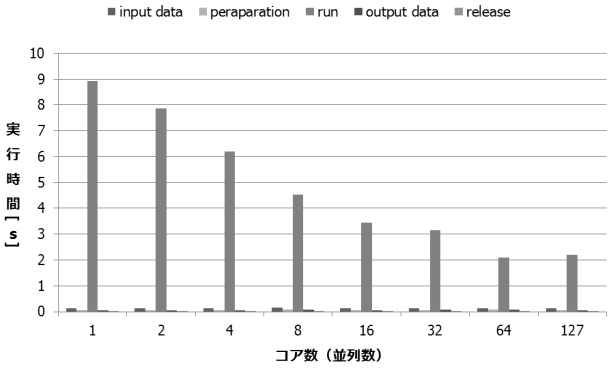

図 8 grayScale の実行時間
Fig. 8 Execution Time of grayScale

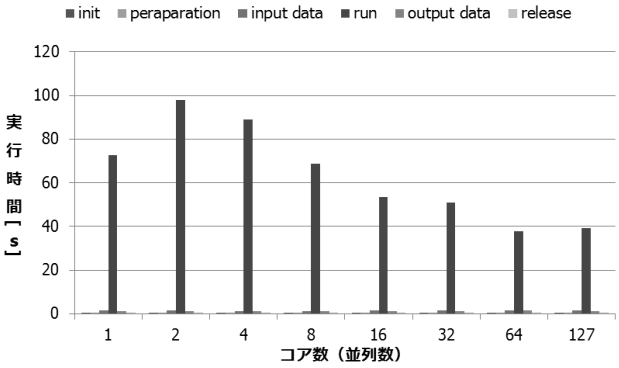

図 9 linearsearch の実行時間
Fig. 9 Execution Time of linearsearch

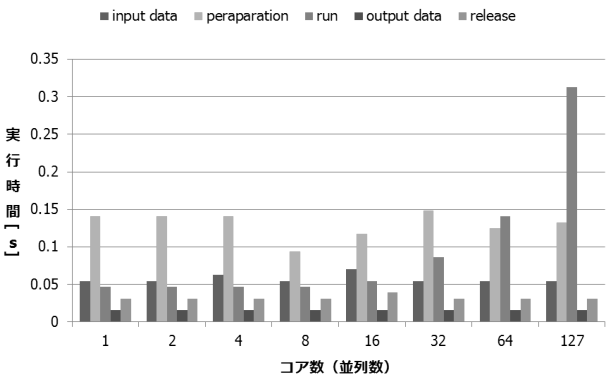

図 10 runLength の実行時間
Fig. 10 Execution Time of runLength

みシステム向けのメニーコアアーキテクチャを有効利用できることが言える．

5. まとめ

本稿では，組込みシステム向けメニーコアアーキテクチャである SMYLEref プロセッサ向けの OpenCL 環境である SMYLEref を実装し，FPGA ボードによる評価環境を用いた 128 コア並列動作環境の上でベンチマークプログラムを動作させて評価実験を行った．実験の結果，各タスクに割り当てるコア数を増やすことで実行時間が短縮されており，SMYLE OpenCL によって今度の課題としては，複数のタスクを同時に SMYLEref コア上で並列動作させ，複数のプログラムを同時並列実行させた環境を整備しシステム全体の実行時間を改善することなどがあげられる．

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，有益なご助言を頂いた（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] Owens, J. D., Luebke, D., Govindaraju, N., Harris, M., Krüger, J., Lefohn, A. and Purcell, T. J.: A Survey of General-Purpose Computation on Graphics Hardware, *Computer Graphics Forum*, Vol. 26, No. 1, pp. 80–113 (2007).
[2] NVIDIA Corporation: *NVIDIA CUDA C Programming Guide, version 4.0*, available from ⟨http://developer.download.nvidia.com/compute/cuda/4_0/toolkit/docs/CUDA_C_Programming_Guide.pdf⟩ (2011).
[3] Khronos OpenCL Working Group: *The OpenCL Specification Version 1.1*, available from ⟨http://www.khronos.org/registry/cl/specs/opencl-1.1.pdf⟩ (2011).
[4] Lindholm, E., Nickolls, J., Oberman, S. and Montrym, J.: NVIDIA Tesla: A Unified Graphics and Computing Architecture, *IEEE Micro*, Vol. 28, pp. 39–55 (2008).
[5] 稗田拓路，西山直樹，谷口一徹，冨山宏之，井上弘士：組込みシステム向けメニーコア用 OpenCL 環境，情報処理学会研究報告，Vol. 2012-SLDM-155, No. 2, pp. 1–6 (2012).
[6] 江谷典子，稗田拓路，冨山宏之：SMYLE OpenCL における組込み関数の開発と評価，情報処理学会研究報告，Vol. 2012-EMB-27, No. 7, pp. 1–8 (2012).
[7] 茂木　勇，木村一樹，砂田徹也，並木美太郎：省電力 MIPS プロセッサ Geyser の FPGA 版評価ボードへの Linux の移植，情報処理学会研究報告，Vol. 2010-ARC-189, No. 9, pp. 1–8 (2010).
[8] 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, pp. 1–7 (2012).
[9] 西山直樹，稗田拓路，谷口一徹，冨山宏之，井上弘士：組込みメニーコア SoC 向け OpenCL 環境の開発と予備評価，DA シンポジウム 2012 予稿集，一般社団法人情報処理学会，pp. 73–78 (2012).