Vol.2012-OS-123 No.7
Vol.2012-EMB-27 No.7
2012/12/5

SMYLE OpenCL における組込み関数の開発と評価

江谷典子[†1]　稗田拓路[†1]　冨山宏之[†2]

組込みシステム向けメニーコアアーキテクチャにおけるヘテロジニアス並列コンピューティング環境を実現するため，高レベル API モデルとして OpenCL 環境の構築に取り組んでいる．そこで，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）』の中で，『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 の評価環境を用いて，OpenCL に準拠した高レベル API の開発を行う．OpenCL バージョン 1.2 に対応した算術関数・整数関数・共通関数・幾何関数・比較関数・同期関数を開発した．また，OS を実装していないコア上で倍精度(double 型)や単精度(float 型)の演算ができるように浮動小数点ソフトウェアエミュレーションプログラムの開発も行った．本稿では，組込みシステム向けメニーコア用 SMYLE OpenCL における組込み関数の設計と実装を述べ，並列ベンチマークテストの評価結果を示す．

Implementation and Evaluation of Built-in Functions in SMYLE OpenCL

NORIKO ETANI[†1]　TAKUJI HIEDA[†1]
HIROYUKI TOMIYAMA[†2]

As a model of high-level API for many-core architecture, an OpenCL environment has been developed in order to build a heterogeneous parallel computing environment. 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". On this environment, high-level API based on OpenCL specification for many-core architecture model in embedded system has been developed. Math functions, integer functions, common functions, geometric functions, relational functions, and barrier function of synchronization functions in built-in functions are developed for OpenCL version 1.2. And the routines for floating point emulation are developed in order to compute the ranges of float and double on the core which OS is not installed in. This paper describes a design and its implementation of built-in functions in SMYLE OpenCL for embedded system with many cores, and shows an evaluation result of the parallel benchmark test.

1. はじめに

組込みシステム向けの高性能かつ低消費電力なメニーコアシステムの実現を目指して，1)組込みシステムを意識した効率的な超並列処理の実現，2)大幅な動作時消費電力の削減，3)ソフトウェアの生産性の向上，を重要な課題として捉えて[1]，ヘテロジニアス並列コンピューティング環境を実現するために，高レベル API モデルとして OpenCL（Open Computing Language の略）環境の構築に取り組んでいる[2].

メニーコアアーキテクチャモデル（図 1）は，共有メモリ，ホストプロセッサ，メニーコアプロセッサアレイから構成されている．想定される動作は，次の通りである．ホストプロセッサでは，OS の実行および制御側プログラムの実行を行う．メニーコアプロセッサアレイでは，各 PE(Processing Element)は，独自のプログラムカウンタを持ち，他の PE に影響されずに単独で演算用プログラムの実行を行う．このため，1 つ 1 つの PE ごとに異なるプログラムを動作させることが可能である．共有メモリを介して，ホストプロセッサとメニーコアプロセッサアレイ間のデータのやり取りを行う[2].

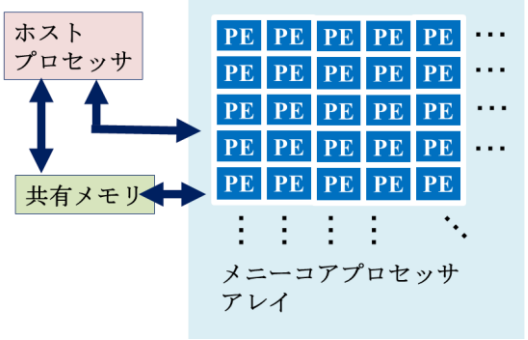

図 1　メニーコアアーキテクチャモデル[2]

†1 立命館大学総合科学技術研究機構
Research Organization of Science and Technology, Ritsumeikan University
†2 立命館大学理工学部
College of Science and Engineering, Ritsumeikan University

本開発では，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)』の中で，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[1]を用いる．SMYLEref のアーキテクチャを図 2 に示す．SMYLEref は，数個（図 2 の例では 8 個）のプロセッサコアをバスで結合したクラスタを，2 次元メッシュの NoC(network on chip)で結合したアーキテクチャを想定している[1]．クラスタには，命令キャッシュ(IL1)とデータキャッシュ(DL1)を実装したスカラコアや分散共有 L2 キャッシュを備えるプロセッサと，チップ外部とのインタフェース(SDRAM コントローラなど)を持つペリフェラルクラスタがある．FPGA を用いた評価環境は，上述の 1 つのクラスタ上で OS やシステムソフトウェアの評価・検証を可能にしている．

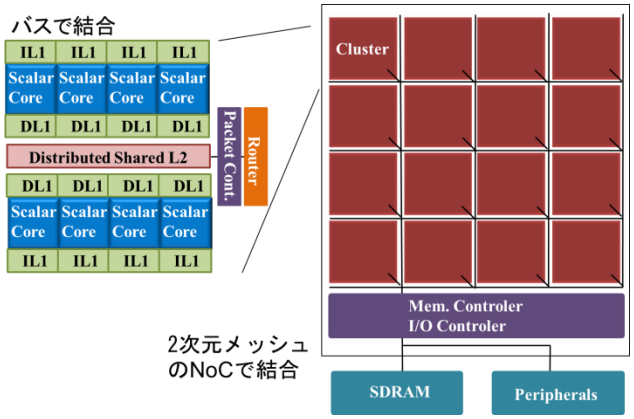

図 2　SMYLEref のアーキテクチャ[1]

また，本プロジェクトにおける平成 24 年度の研究目標[3]には，「実アプリケーションにおいて従来技術と比べて電力当たりの処理性能 2 倍，消費電力量 1/10 が実現可能であることを検証する」と設定されている．この目標設定値は，本開発における評価目安ともなる．

OpenCL とは，CPU・GPU・DSP などが混在するプラットフォームで並列処理プログラムを書くためのフレームワークである[4]．クロノス(Khronos)グループによって標準化が進められて，2008 年 12 月に OpenCL 1.0 が制定された．2010 年 6 月にバージョン 1.1 が，2011 年 11 月にバージョン 1.2 がリリースされている[5]．C99 に基づく C 言語（コンパイラ）と並列計算をサポートする API 群で構成されており，データ並列およびタスク並列にも対応している．この OpenCL プラットフォームを構成する計算要素には，ホストと OpenCL デバイスがある．ホストとは，制御側のソフトウェアが動作する計算環境であり，通常は CPU といったプロセッサおよびそれに付随するメモリを含む．OpenCL デバイスとは，演算用のソフトウェアが動作する計算環境であり，通常は GPU・DSP・Cell/B.E.・CPU などの複数の演算用プロセッサおよびそれらに付随するメモリを含む．デバイス側で動作するプログラムを特にカーネル(kernel)と呼び，OS が動作していない環境を想定している．

本稿では，現在の OpenCL の仕様について，高いリアルタイム性が要求される組込みシステムにも対応できるように，独自の制約と解釈を与えた SMYLE OpenCL を検討し，FPGA を用いたメニーコアアーキテクチャ SMYLEref の評価環境を用いて，その組込み関数の開発と評価を行う．

本論文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 節で本開発におけるシステム構成を説明した後，4 節で設計と実装を行った組込み関数について述べる．その有効性を確認するための並列ベンチマークテストならびに結果について，5 節で示し，最後に 6 節でまとめる．

2. 関連研究

オープンソースとして提供されている OpenCL の概要について説明をおこなう．

SnuCL [6]は，韓国・ソウル大学が開発した OpenCL フレームワークである．OpenCL では，複数の OpenCL 計算デバイスを持った単一異種システムのために書かれた OpenCL アプリケーションを変更しないで利用できるようにしている（再利用性）．単一のホストノードと複数の計算ノードで構成されたターゲット･クラスタは，相互接続ネットワークで接続されている．ホストノードは複数の CPU コアが含まれており，各計算ノードは複数の CPU コアあるいは複数の GPU で構成されている．GPU や CPU コアのセットは，OpenCL の計算デバイスになる．SnuCL のアプリケーションは，ホスト･ノードにあるかのように，計算ノードで計算デバイスを利用することができる（透過性）．2012 年 6 月，SnuCL 1.2beta がリリースされた[7]．

Portable OpenCL(POCL) [8]は，スペイン・レイファンカルロス大学のチャールズ･S ラ･ラマらが中心となって開発を行っている．このプロジェクトの目的のひとつは，ターゲット依存部分を手作業で適合させることを回避して，OpenCL プログラムの可搬性を向上させることである．2012 年 8 月 17 日，POCl 0.6.0 がリリースされた．

組込み関数について，SnuCL はターゲット用コンパイラを利用し，POCL は CLANG を組み込んだ LLVM コンパイラを利用している．本開発で利用しているコンパイラは，32-bit MIPS 用コードを生成するコンパイラ[1]である．よって，ターゲット用コンパイラを利用できる開発環境を提供している SnuCL 1.2beta のソースコードを利用した．また，POCL0.6.0 は，組込み関数の処理速度を計測する並列ベンチマークテストの比較対象として用いた．

Vol.2012-OS-123 No.7
Vol.2012-EMB-27 No.7
2012/12/5

3. システム構成

3.1 開発環境

(1) ハードウェア

SMYLEref 評価環境では，Xilinx 社製 FPGA チップである Virtex-6 を搭載する ML605 評価ボードをプラットフォームとして利用している．表 1 には，ML605 評価ボード，および搭載する Virtex-6 チップの仕様を示す．

表 1　ML605 ボードと Virtex-6 チップの仕様[9][10]

ML605 ボード

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 オシレータ 66MHz ソケットオシレータ

Virtex-6 チップ

CMOS	40nm, 1.0V
Logic Cells	241,152
CLB Slices	37,680
Block RAM	14,976Kbit
ユーザーI/O 数	720
消費電力[11]	Static Power: 3.6W, Total Power: 6.5W

図 3 には開発環境の外観を示す. 表 2 にて示した PC は，ML605 ボードと USB にて接続を行い，Windows 用ターミナルエミュレータ Tera Term Version 4.73 を利用した通信端末としても利用している．

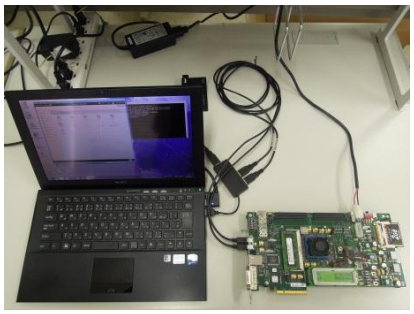

図 3　開発環境の外観

(2) SMYLEref アーキテクチャの評価環境

本評価環境では，ML605 ボードを 1 枚使用し，1 つのボード上には，図 2 で示した SMYLEref アーキテクチャの 1 クラスタをシミュレーションできるように，4 つのソフトコアが準備されている．図 4 では、この 4 つのソフトコアの割り当てを示す．1 つのコアをプロセッサと仮定し，ホストを実行できるコアとして割り当てて，残りの 3 つのコアをデバイスとして用いることを想定している．また，コアへ供給する周波数は，10MHz の設定を用いた[1].

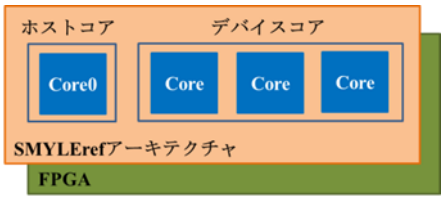

図 4　コアの割り当て

本ハードウェアでは, SIMD(Single Instruction / Multiple Data)は実装されていないので，ベクタ型データや演算は扱わない. また, 浮動小数点演算器は実装されていないので，ホストプログラムでは MPFR（任意精度浮動小数点演算ライブラリ）と GMP（高速多倍長演算ライブラリ）を組み込んだコンパイラ(gcc)のソフトウェアエミュレーション(Soft Float)機能を利用して, 倍精度(double 型)や単精度(float 型)の演算や算術関数の実装を行うことが可能となる. また，カーネルでは，今回開発を行った浮動小数点ソフトウェアエミュレーションプログラムをカーネルへ組み込んで，倍精度や単精度の演算を行う．

(3) ソフトウェア

本開発では，表 2 の開発環境へターゲット OS である mips-geyser-linux[1]用クロスコンパイル環境を構築している. また, ベンチマークテストで比較評価を行う POCL0.6.0 は，この開発環境にインストールされている．

表 2　ソフトウェア開発環境

PC	Sony VAIO
CPU[12]	名称：Intel® Core™ i7-2640M プロセッサ 動作周波数：2.80GHz 消費電力(Max TDP)：35W
OS	Windows 7 Professional Service Pack 1
VM	VMware Player 4.0.3
HOST	32-bit Fedora 16

3.2 ホストプログラムとカーネルの構成

本システムにおけるホストプログラムとカーネルの構成について，図 5 に示す．

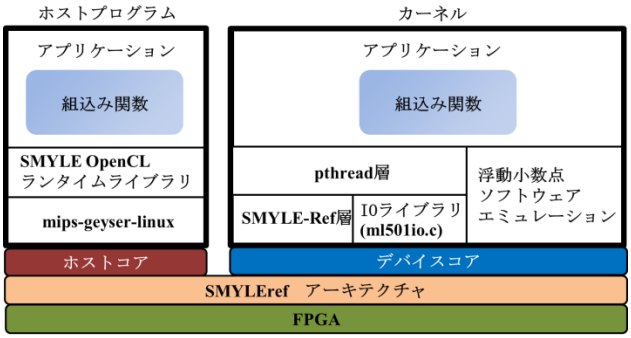

図 5　ホストプログラムとカーネルの構成

ホストプログラムは，アプリケーションプログラム，SMYLE OpenCL ランタイムライブラリ，32-bit MIPS Linux OS である mips-geyser-linux から構成される．本来，この OpenCL ランタイムライブラリを用いて，アプリケーションプログラムは作成されるが，本開発では未実装とした．本 OS 上で動作できるように 32-bit MIPS 用コンパイラにより，アプリケーションプログラムをコンパイル，リンクして，実行ファイルを生成する．次に，この実行ファイルと Linux カーネルを make して，バイナリファイル生成する．バイナリファイルは，SMYLEref アーキテクチャ上へロードされ，実行される．

カーネルは，アプリケーションプログラム，簡易版 pthread ライブラリ，IO ライブラリ，および浮動小数点ソフトウェアエミュレーションから構成される．アプリケーションプログラムは，SMYLEref の基礎評価を目的に開発された簡易版 pthread ライブラリ(並列処理 API)[1]を利用して開発される．この並列処理 API は，低レベルの制御や mutex 操作を実装する SMYLE-Ref 層と既存の IO ライブラリを利用して実装されている．OS を実装していないソフトコア上のアプリケーションプログラムを並列化させて実行させるために用いる．本並列処理 API には，スレッドの生成・終了を制御する pthread_create， othread_join などの他，排他制御用の関数 (pthread_mutex_lock，pthread_mutex_unlock など）の機能がある[1]．ホストと同様の 32-bit MIPS 用コンパイラにより，カーネルを構成するプログラムをコンパイルして，オブジェクトファイルを生成する．次に，リンカ ld により，オブジェクトファイルはリンクされ，バイナリファイルを生成する．バイナリファイルは，SMYLEref アーキテクチャ上へロードされ，実行される．

3.3 ホストとデバイスの動作環境

組込み関数を評価するために，次のような動作環境を用いる．

ホストの動作環境（図 6）は，SMYLEref アーキテクチャの評価環境で想定された通り，1 つのコアをホストコアとして割り当てる．アプリケーションプログラムは，このコアへ実装された OS 上で実行される．

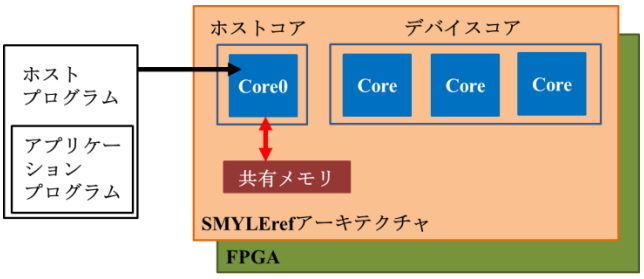

図 6　ホストの動作環境

デバイスの動作環境（図 7）は，SMYLEref アーキテクチャにある 4 つのソフトコアをデバイスとして用いて，アプリケーションプログラムを並列実行させる．

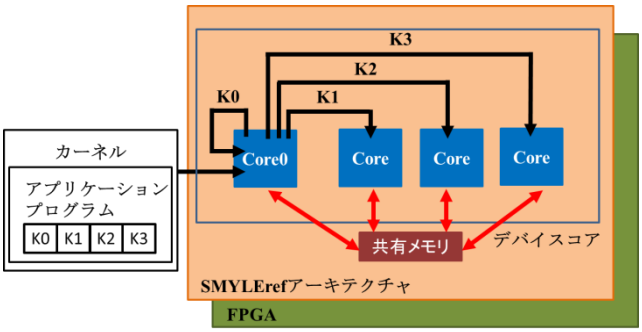

図 7　デバイスの動作環境

よって，組込み関数を利用するアプリケーションプログラムによる評価は，ホストコア上で動作する場合，および 4 つのデバイスコア上で動作する場合を区別して行う．

4. 組込み関数の設計と実装

4.1 スタティックライブラリ

本組込み関数は，OpenCL 1.2 仕様[5]に準拠したオープンソースである SnuCL 1.2beta を利用し，サポートが可能な関数を抽出した．カーネルでは，OS が存在していない環境を想定しているので動的ライブラリでは動作が不可となるため，スタティックライブラリとして生成した．

算術関数 88 項目，整数関数 17 項目，共通関数 9 項目，幾何関数 7 項目，比較関数 16 項目のテストを，ホストの動作環境とデバイスの動作環境にて実施した．また，同期関数 1 項目のテストは，デバイスの動作環境のみで実施した．

本システムにおいて，サポートできる組込み関数を表 3 に示す．○印はサポート可能な関数，空白はサポート不可の関数，※は，標準ライブラリ関数を用いて利用が可能な関数である．

(1) 算術関数

ホストおよびカーネルともにサポート可能な 3 関数以外の関数は，コンパイラに組み込んである MPFR や GMP が必要であるので，ホストで標準ライブラリの算術関数を用いて利用する．

(2) 整数関数

ホストおよびカーネルともにサポート不可の 1 関数は，ホストおよびカーネルで標準ライブラリの算術関数を用いて利用する．

(3) 共通関数

ホストおよびカーネルともにサポート可能な 1 関数以外の関数は，コンパイラに組み込んである MPFR や GMP が必要であるので，ホストでのみ利用することができる．

(4) 幾何関数

コンパイラに組み込んである MPFR や GMP が必要である

ので，ホストでのみ利用することができる．

(5) **比較関数**

カーネルでは，「isequl」「isnotequal」「isordered」「bitselect」「select」の 5 関数以外の関数は，正の値のみを扱うことができる．負の値を扱う場合は，ホストでのみ利用する．

(6) **同期関数**

並列処理 API を用いた開発環境へ組み込めるようなライブラリとして提供する．

表 3　SMYLE OpenCL　組込み関数一覧

算術関数

関数名	ホスト	カーネル	関数名	ホスト	カーネル
acos	※		floor	※	
acospi	※		ldexp	※	
asin	※		log	※	
asinpi	※		log10	※	
atan	※		mad	○	○
atan2	※		nextafter	○	○
atanpi	※		pow	※	
ceil	※		rsqrt	※	
copysign	※		sin	※	
cos	※		sinh	※	
cosh	※		sinpi	※	
cospi	※		sqrt	※	
expm1	※		tan	※	
fabs	※		tanh	※	
fdim	○	○			

整数関数

関数名	ホスト	カーネル	関数名	ホスト	カーネル
abs	※	※	mad_sat	○	○
abs_diff	○	○	max	○	○
add_sat	○	○	min	○	○
clz	○	○	mul24	○	○
hadd	○	○	rotate	○	○
rhadd	○	○	sub_sat	○	○
clamp	○	○	popcount	○	○
mad24	○	○			

共通関数

関数名	ホスト	カーネル	関数名	ホスト	カーネル
degrees	○		radians	○	
mix	○		step	○	○

幾何関数

関数名	ホスト	カーネル	関数名	ホスト	カーネル
dot	○		length	○	
distance	○		normalize	○	

比較関数

関数名	ホスト	カーネル
isequal	○	○
isnotequal	○	○
isgreater	○	○
isgreaterequal	○	○
isless	○	○
islessequal	○	○
islessgreater	○	○
isordered	○	○
bitselect	○	○
select	○	○

同期関数

関数名	ホスト	カーネル
barrier	-	○

4.2 浮動小数点ソフトウェアエミュレーション

カーネルにおいて倍精度(double 型)や単精度(float 型)の演算ができるように，Apple オープンソースの浮動小数点ソフトウェアエミュレーションプログラム(floatlib.c)[13]を本コンパイラへ対応できるように開発した．また，IO ライブラリを用いた本並列処理 API と同様に，カーネルへ組み込んで利用ができる．本エミュレーションの機能は次の通りである．

- 四則演算　「+」「-」「*」「/」
- 比較　「=」「<」「>」
  ただし，「<」「>」は正の値のみ可
- データ型の変換
  「戻り値データ幅の拡張」「戻り値データ幅の縮小」「int 型への変換」
- 符号付きゼロ(-0)の扱いが可

4.3 同期関数（バリア関数）

(1) **仕様**

同期関数は，図 8 で示すように各カーネルを複数並列実行する場合，処理中プログラム間での同期制御を行う関数である．

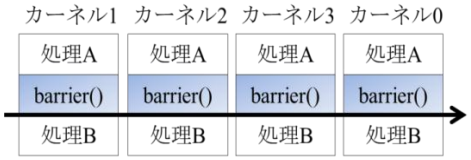

図 8　同期関数の仕様

実行中のカーネル全てが，同期関数をコールするまで，各カーネルは処理 A を実行後，待機する．全てのカーネル

が同期関数をコールすることで，待機中から脱出する．よって，カーネル間で処理 B の開始を揃えることができる．

(2) **設計**

1 つのカーネルでは，同期関数を複数回コールすることが可能である．よって，カーネル間でプログラムの同期制御を行うためには，「カーネル間の同期制御」および「1 つのカーネル内のシーケンス制御」が必要である（図 9）．

図 9　同期制御の方法

同期制御は，カーネル間でのプログラム実行の同期を監視し，全カーネルが同期関数をコールするまで待機する．全カーネルが同期関数をコールすると，全カーネルは同期関数を終了する．

シーケンス制御は，1 つのカーネル内で複数同期関数コールした場合，1 つの同期が完了するまで次の同期関数コールを待機させる．

この２つの制御を行うため，「バリアカウンタ」と「バリア脱出カウンタ」を用いる．両カウンタの制御手順は，次の通りである．

- バリアカウンタ.

① カウンタ値の初期値は 0
② 同期関数をコールするとカウンタは+1
③ カウンタ値が起動デバイスコア数と同値で終了．同値になるまで待機.

- バリア脱出カウンタ

① カウンタ値の初期値は 0
② カウンタ値の初期値が設定されていない場合 同期関数入り口で待機
③ 同期監視ループから脱出した時，カウンタ値は+1
④ 上述③の後，バリア脱出カウンタ値およびバリアカウンタ値がともに起動デバイスコア数と同値であれば，バリアカウンタおよびバリア脱出カウンタを初期化

この２つのカウンタへの操作は critical section となるので，本並列処理 API で用意されている排他制御関数(pthread_mutex_lock，および，pthread_mutex_unlock)を用いて保護を行う．

(3) **実装**

図 10 に同期関数の処理フローを示す．

共有メモリに，バリアカウンタ: global_barrier，バリア脱出カウンタ: global_escape を定義する．

同期関数の先頭では，1 つのカーネル内でプログラムのシーケンス制御を行うために，バリア脱出カウンタの値をチェックし，0 以外の場合は，「他のカーネルが，前回の同期関数コールから脱出していない」と判断して，同期処理へ入らずに待機する．バリア脱出カウンタが 0 になれば，「全カーネルは，同期関数をコールしたので，同期関数コールの待機から脱出した」と判断し, 次のステップへ進む．

同期関数へ入ると，同期関数をコールしたことを記憶するため，排他制御を行い，バリアカウンタの加算(critical section)を行う．その後，他のカーネル全てが，同期関数をコールするまで待機する．

全カーネルが同期関数をコールしたならば，この待機から脱出する．そこで，排他制御を行い，バリア脱出カウンタの加算(critical section)を行う．この時，バリアカウンタとバリア脱出カウンタがともに起動しているデバイスコア数と同値であれば,「今回の同期関数コールは終了した」と判断し，排他制御を行い，両方のカウンタをゼロクリア(critical section)する．ここで，同期関数の処理は終了する．

```
1)  前回の同期関数コール終了待ち
    while(global_escape != 0){
    }
2)  排他制御ロック
3)  バリアカウンタの加算(critical section)
    global_barrier +=1;
4)  排他制御アンロック
5)  同期制御のため待機
    while(global_barrier <  起動デバイスコア数){
    }
6)  排他制御ロック
7)  バリア脱出カウンタ加算(critical section)
    global_escape +=1;
8)  バリアカウンタおよびバリア脱出カウンタが起動デバイスコア数と同じであれば初期化(critical section)
    global_escape = 0;
    global_barrier = 0;
9)  排他制御アンロック
```

図 10　同期関数処理フロー

(4) **動作確認**

OpenCL の仕様は，図 11 で示した「データ並列プログラミングモデル」と「タスク並列プログラミングモデル」を提供している[4].

データ並列プログラミングモデルとは，1 つのカーネル実行タスクが，デバイス内の演算ユニットで，同時に動作するようなモデルである．

タスク並列プログラミングモデルとは，デバイスで動作する複数の異なるカーネル実行タスクが，デバイス内の別々の演算ユニットに割り当てられて，動作するようなモデルである．

本並列処理 API を用いたカーネルでは，上述の並列プログラミングモデルを実現している．

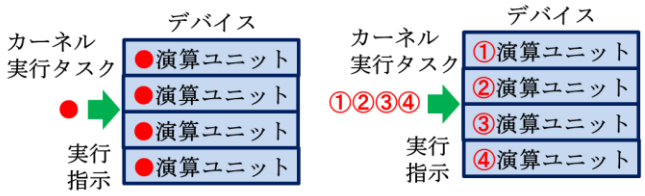

図 11　データ並列プログラミングモデル（左）と
タスク並列プログラミングモデル（右）

そこで，この同期関数を用いたアプリケーションプログラムが，カーネル間で同期をとりながら実行されることを確認するため，次の 2 つのテストを準備した．

テスト 1 は，データ並列プログラミングモデルの検証である．図 12 で示したプログラムを 4 つのカーネルで実行するテストである．

テスト 2 は，タスク並列プログラミングモデルの検証である．図 12 で表示したプログラムをもとに，各デバイスコアで異なるプログラムを実行する．コア 0 では「0」，コア 1 では「1」，コア 2 では「2」，コア 3 では「3」という文字を出力する．

```
int barrier_app(){
        printf("1¥n");
        barrier();
        printf("2¥n");
        barrier();
        printf("3¥n");
        barrier();
        printf("4¥n");
        barrier();
        printf("5¥n");
}
```

図 12　テスト 1 のプログラム

図 13 には，ML605 ボードに接続された通信端末画面に表示されたテスト 1 の実行結果を示す．テスト 1 では，同じプログラムを 4 つのカーネルが実行しているので，同じ文字 4 文字が 5 回「1111」「2222」「3333」「4444」「5555」と出力されている．

図 14 には，テスト 2 の実行結果を示す．テスト 2 では，異なるプログラムを 4 つのカーネルが実行しているので，異なる文字 4 文字が 5 回「1230」「3012」「3012」「3012」「3012」と出力される．

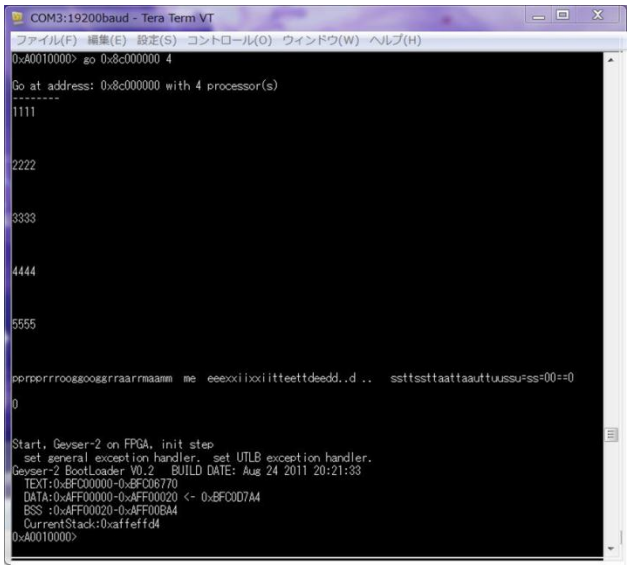

図 13　テスト 1 の実行結果

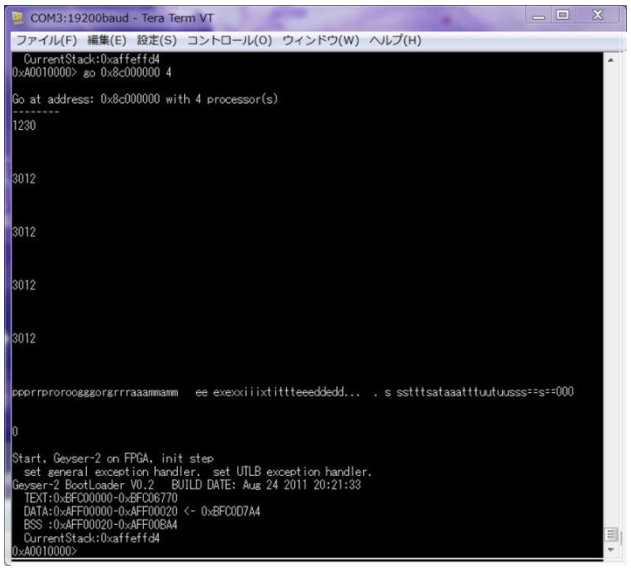

図 14　テスト 2 の実行結果

この出力結果から，アプリケーションプログラムは，同期関数を使うことで，カーネル間で同期をとりながら，並列実行していることを確認できる．

5. 評価

開発した組込み関数の性能を評価するために，カーネルでは，同期関数を用いた並列アプリケーションプログラムの処理速度を計測する．この処理計測には，次の 2 種類のベンチマークを準備した．

- ベンチマーク 1(データ並列プログラミングモデル)
  同じプログラムを 4 つのカーネルが実行する．組込

み関数のうち 12 の整数関数(abs_diff・add_sat・hadd・rhadd・mad24・mad_sat・max・min・mul24・mul24・sub_sat・popcount)を利用する．1 つの整数関数をコールして演算を行う度に，同期関数をコールする．

- ベンチマーク 2(タスク並列プログラミングモデル)
  異なるプログラムを各 4 カーネルが実行する．ベンチマーク 1 のプログラムにおける整数関数をコールする順番を変更して異なるプログラムとする．

また，表 2 のマルチコア CPU を搭載した PC 上で動作する POCL0.6.0 を用いて, SMYLE OpenCL と同じカーネルのプログラムを 4 スレッドで起動させる．その処理速度を計測し，SMYLE OpenCL と比較した．この評価結果を表 4 に示す.

表 4　並列処理ベンチマーク計測結果

単位 msec	ベンチマーク 1	ベンチマーク 2
SMYLE OpenCL	0.0010212296	0.0010212834
POCL	0.001999	-

ベンチマーク 1 およびベンチマーク 2 ともに，処理速度について, SMYLE OpenCL は POCL の約 2 倍の速さであることが分かった．消費電力について，表 1 よりベンチマークテストで利用している FPGA の消費電力は 6.5W, また表 2 よりベンチマークテストで利用した CPU の消費電力(Max TDP)は 35W であると想定すると, SMYLE OpenCL は POCL よりも消費電力が約 1/10 以下となる.

6. まとめ

本稿では，組込みシステム向けメニーコアシステム用 SMYLE OpenCL における組込み関数の設計と実装について述べた．また，開発した同期関数や整数関数を使った並列ベンチマークテストを行い，処理速度を計測し，マルチコア CPU で動作する POCL0.6.0 と比較し，評価した．その結果, SMYLE OpenCL における組込み関数を用いた並列アプリケーションプログラムの処理速度は，マルチコア CPU で動作するプログラムよりも速いことを確認した.

今後は，本開発で利用したプログラムやサンプルプログラムはできる限り公開することで，多くのメニーコアプロセッサのソフトウェア開発基盤整備に役立ち，ソフトウェアの開発や評価へと広く利用してもらうことを考えている.

謝辞　本研究は，独立行政法人新エネルギー・産業技術総合開発機構(NEDO)の委託により実施した.

参考文献

1) 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 .
2) 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.
3) 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）実施方針：平成 24 年度,独立行政法人新エネルギー・産業技術総合開発機構(NEDO) ,
http://www.nedo.go.jp/content/100490844.pdf
4) 株式会社フィックスターズ: 改訂新版 OpenCL 入門 1.2 対応，株式会社インプレスジャパン，ISBN978-4-8443-3172-8, 2012.
5) Khronos OpenCL Working Group: The OpenCL Specification Version 1.2, 2012 ,
http://www.khronos.org/registry/cl/specs/opencl-1.2.pdf
6) Jungwon Kim, Sangmin Seo, Jun Lee, Jeongho Nah, Gangwon Jo, Jaejin Lee: SnuCL: an OpenCL framework for heterogeneous CPU/GPU clusters, ICS '12 Proceedings of the 26th ACM international conference on Supercomputing, pp.341-352, ACM, New York, NYUSA c2012, ISBN: 978-1-4503-1316-2, 2012.
7) SnuCL 1.2 beta,
http://aces.snu.ac.kr/Center_for_Manycore_Programming/SnuCL.html
8) Portable OpenCL Version 0.6.0,
https://launchpad.net/pocl
9) Virtex-6 Family Overview, DS150(V2.4) January 19, 2012,
http://www.xilinx.com/support/documentation/data_sheets/ds150.pdf
10) Xilinx Inc.: Getting Started with the Xilinx Virtex-6 FPGA ML605 Evaluation Kit, UG(v1.4) November 15, 2010.
11) 7 シリーズ FPGA で消費電力を半減,
http://japan.xilinx.com/publications/archives/xcell/issue75-76/p12-19.pdf
12) Intel® Core ™ i7-2640 M Processor Specifications,
http://ark.intel.com/products/53464/Intel-Core-i7-2640M-Processor-4M-Cache-up-to-3_50-GHz
13) Apple Open Source,
http://opensource.apple.com/source/cc/cc-798/cc/floatlib.c