Cellプログラム実行環境

Runtime Environment for Cell Programs

前田　誠司　　佐藤　記代子　　川上　健
■MAEDA Seiji　■SATO Kiyoko　■KAWAKAMI Ken

Cell Broadband Engine（CBE）の性能を引き出すためには，搭載されたメディア処理用プロセッサコアSPE（Synergistic Processor Element）を効率的に活用することが必要不可欠である。しかし，複雑化しているハードウェアを単にソフトウェア開発者に提供するだけでは，ソフトウェアが複雑化し，並列処理やリアルタイム処理などが困難になると考えられる。

そこで，複数のSPEを用いたソフトウェアの開発をサポートするSPE実行環境を開発した。プログラミングモデルの提案に加えSPE資源予約やSPEオーバレイ機能などを，想定される各種の利用形態に合わせて提供することにより，並列処理や複数処理の同時実行を容易に実現できるため，CBEを様々な製品に適用できると考えられる。

In order to fully exploit the performance of the Cell Broadband Engine (CBE), it is essential to efficiently utilize the Synergistic Processor Elements (SPEs). However, simply supplying complex hardware to software engineers makes software development complicated, and realizing parallel and real-time processing becomes very difficult.

To solve this issue, Toshiba has developed the SPE runtime environment, which supports software development using multiple SPEs. By offering a programming model developed for potential use cases, along with SPE resource reservation and SPE overlay features, the SPE runtime environment enables programmers to easily implement parallel and concurrent processing, making the CBE applicable to various products.

1　まえがき

Cell Broadband Engine（CBE）は，汎用プロセッサコア1基とメディア処理用プロセッサコアSPE（Synergistic Processor Element）8基を混載する非対称マルチコアプロセッサである。CBEの性能を引き出すためには，搭載されたSPEを効率的に活用することが必要不可欠である。しかし，複雑化しているハードウェアを単にソフトウェア開発者に提供するだけでは，システム設計及びソフトウェア構成が複雑化し，開発コストの増大を招くおそれがある。特に，複数のSPEを用いる際の，並列処理性能の向上やリアルタイム性の維持などが困難になると考えられる。

各処理を実現するスレッドを優先度順に実行する制御を行った場合，同時に実行する処理の組合せによって各処理内のスレッドの動作タイミングが変化してしまい，処理性能が変化するという課題がある。例えば，**図1**に示すように，処理dは，処理a及び処理bと同時に実行した場合には，スレッドd-0とd-1が同時に実行されるが，処理a及び処理cと同時に実行した場合には，スレッドd-0とd-1の実行タイミングがずれてしまう。各処理内のスレッド間では，同期処理やデータ通信が頻繁に行われる場合が多いため，スレッドの実行タイミングが変化すると，処理性能が変化し，その結果リアルタイム性が維持できなくなることが想定される。

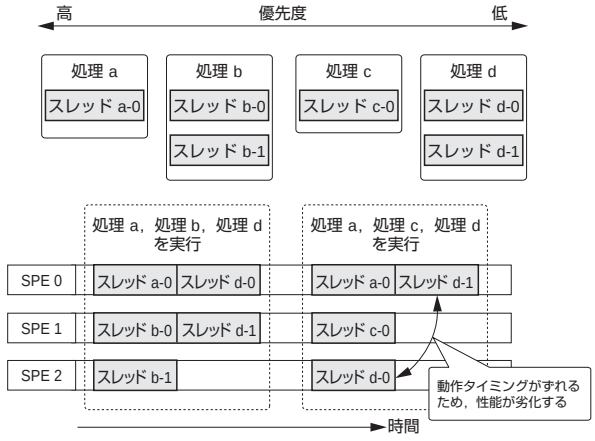

図1．優先度による制御の困難さ ― 優先度による制御では，狙ったタイミングでスレッドを動作させることが難しく，性能劣化を引き起こしやすい。

Difficulty of priority-based control

このような課題を解決するために，Cellリファレンスセットの基本ソフトウェアの一つとして，複数のSPEを用いたソフトウェアの開発をサポートする，SPE実行環境を開発した[1)-4)]。SPE実行環境では，並列処理やリアルタイム処理など，想定される各種の利用形態に合わせた機能を提供することにより，CBEの様々な用途に適用できるように設計している。

ここでは，各種の利用形態に沿ってSPE実行環境の機能を述べる。

2　SPEの利用形態

SPEの利用形態として，**図2**に示すように，単一処理の高速化と複数処理の同時実行の二つが想定されている。

単一処理の高速化とは，1プロセッサでは処理しきれない高負荷のアプリケーションを，複数SPEを用いた並列処理で実現する利用形態である。例えば，HD（High Definition）解像度のH.264/AVC（Advanced Video Coding）のデコード処理及び，ジェスチャ認識や顔画像認識などの高度な画像処理などが想定される。複数SPEを用いる場合には，各並列処理単位にSPEを固定的に割り当て，処理能力を安定的に確保することに加え，SPE間の同期や通信を高速化することが重要と言える。

一方，複数処理の同時実行とは，各処理は1プロセッサで処理できる程度の負荷であるが，高性能なプロセッサの処理能力を生かし，その処理を複数同時に実行する利用形態である。例えば，デジタル放送の複数同時視聴・録画・再生や，複数コンテンツの動画サムネイル同時再生などが想定される。複数処理の同時実行を行う場合には，各処理でSPEを共有しながら，すべての処理が安定的に動作し続けることを保証することが重要と言える。

また，これらの利用形態は，ともに連続データを一定時間内に処理しなければならない，すなわちリアルタイム処理である。そのため，各処理が持つリアルタイム性を保証することが重要であると言える。

更に，SPEプログラムが使用できるローカルストレージ(LS)のサイズは，プログラムとデータを合わせて256 Kバイトと限られている。そのため，LSより大きいプログラムの実行を支援することが重要であると言える。

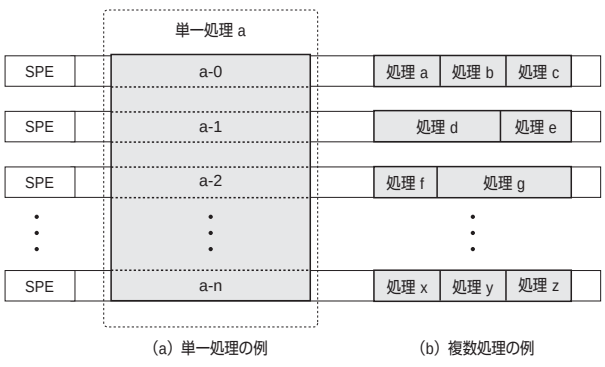

図2．SPEの利用形態 ― 単一処理の高速化と複数処理の同時実行が想定されている。

Possible configurations of SPE running

3　階層型プログラミングモデル

CBE上では，様々な構成のソフトウェアを構築することが可能であるが，各ターゲット個別にソフトウェアを開発していたのでは，開発効率が悪い。そこで，ソフトウェア開発の際に参照するモデルとして，**図3**に示す階層型プログラミングモデルを提案し，アプリケーション本体，ソフトウェアによるメディア処理エンジン（以下，ソフトウェアエンジンと略記），PPE（Power Processor Element）及びSPEモジュールのAPI（Application Programming Interface）仕様と動作モデルを定め，ソフトウェアの再利用性を高めている。

階層型プログラミングモデルは，PPE/SPEモジュール層，SPEスレッド層，SPEオーバレイ層の3層構造で構成されている。PPE/SPEモジュール層では，PPEモジュールとSPEモジュールの2種類のモジュールがあり，これらのモジュールを組み合わせることによりソフトウェアエンジンを構成する。ソフトウェアエンジンを複数同時に実行することにより，複数処理の同時実行を実現することができる。

PPEモジュールは，主にゲストOS（Operating System）が提供するスレッドやタスクを用いて実装する。一方，SPEモジュールは，SPE実行環境が提供するSPEスレッドを用いて実装する。SPEスレッドは，SPEを仮想化した実行単位である。SPEモジュール内で複数のSPEスレッドを用いることに

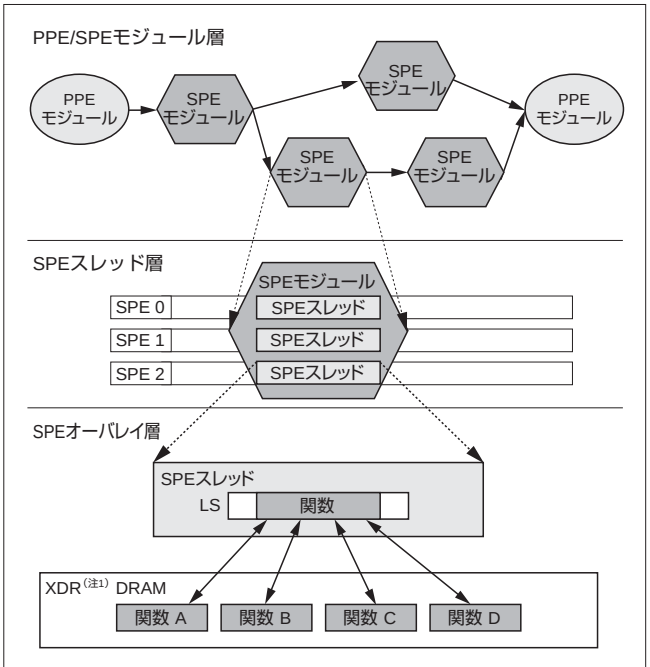

図3．階層型プログラミングモデル ― PPE/SPEモジュール層，SPEスレッド層，及びSPEオーバレイ層の3層構造で構成されている。

Multilayered programming model

（注1）　XDRは，Rambus社の登録商標。

より，並列処理を実現することができる。

またSPEスレッドは，プログラムの分割実行を支援するSPEオーバレイ機能を利用することができる。SPEオーバレイ機能では，プログラムを分割したオーバレイプログラムをメインメモリ上に配置し，処理の進捗（しんちょく）に合わせてオーバレイプログラムをLS上に入れ替えながら実行するための諸機能を提供する。SPEオーバレイ機能を用いることにより，LSより大きいプログラムの実行を実現することができる。

4　SPE実行環境

SPE実行環境では，階層型プログラミングモデルを実現するための諸機能を提供する。またSPEの利用形態に合わせ，単一処理の高速化向けには占有SPE機能を，複数処理の同時実行向けには時分割共有SPE機能を，それぞれ提供する。

4.1　ソフトウェア構成

SPE実行環境のソフトウェア構成を**図4**に示す。

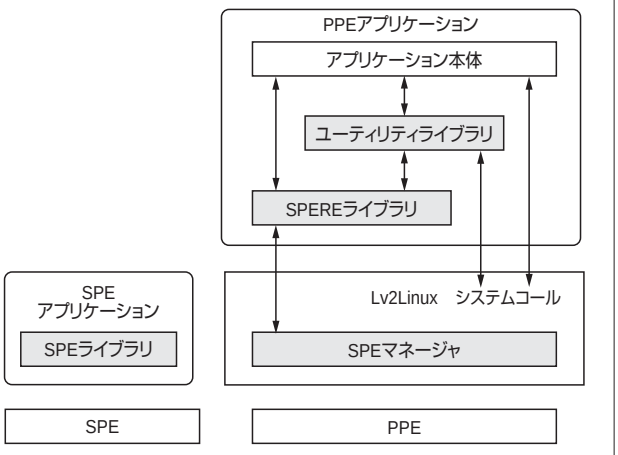

図4．SPE実行環境 ― SPEマネージャに加え，PPE用のSPEREライブラリ，ユーティリティライブラリ，SPE用のSPEライブラリで構成される。
SPE runtime environment

SPE実行環境は，PPE上で動作するSPEマネージャと，SPERE（SPE Runtime Environment）ライブラリ，ユーティリティライブラリ，SPE上で動作するSPEライブラリで構成される。

SPEマネージャは，Cellリファレンスセットでの Linux(注2) の実装であるLevel 2 Linux（Lv2Linux）カーネル内で動作し，SPEスレッド機能など，SPE実行環境の主要機能を実現する。SPEREライブラリは，PPEアプリケーションからSPEマネージャが提供する機能を使用するためのライブラリである。ユーティリティライブラリはLv2Linuxの機能を活用し，SPEマネージャの機能をより使いやすくするためのライブラリである。SPEライブラリは，SPEアプリケーションからSPE実行環境の機能を使用するためのライブラリである。

4.2　リアルタイム性の維持

連続データを一定時間内に処理する場合，そのリアルタイム性の維持には，割込み応答性よりも資源予約と動作環境の準備が重要となる。

資源予約とは，処理に必要な計算機資源を事前に予約することで，安定的に処理を継続するための機能である。SPE実行環境では，資源予約としてSPE予約機能を提供する。SPE予約機能は，SPEの利用形態に合わせて，占有SPE予約機能と時分割共有SPEを用いたスケジューリング予約機能の2種類の予約方式を提供する。

占有SPE予約機能とは，あるSPEを，指定したSPEスレッドで占有して使用する機能である。この占有SPEを複数用いることにより，単一処理の高速化を目指したSPE並列処理を容易に実現することができる。

時分割共有SPEを用いたスケジューリング予約機能とは，あるSPEを，複数のSPEスレッドで時分割共有し，そのスケジューリングタイミングを事前に予約する機能である。このスケジューリング予約機能を用いることにより，複数処理の同時実行を安定して実現することができる。

一方，動作環境の準備とは，処理に必要な動作環境の設定を事前に行うことで，安定的に処理を継続するための機能である。Linuxなどの仮想記憶機構を持つOSでは，メモリ領域を確保しても，アプリケーションがその領域を参照するまでは物理メモリの用意やアドレス変換情報の設定を行わない場合が多い。そのため，初めてメモリ領域を参照した際にページフォールトが発生し，OSによる処理が行われることによりアプリケーションの性能が低下する。この性能低下を抑制するため，SPE実行環境では，メモリ参照の設定を事前に行うアクセスヒント機能を提供する。アクセスヒント機能では，SPEがアクセスするメモリ領域に対し，事前に物理メモリの用意やアドレス変換情報の設定を行う。

4.3　SPE並列処理

SPEを用いた並列処理は，占有SPE予約機能を用いて複数のSPEスレッドを実行することで，容易に実現することができる。更に，SPE実行環境では，CBEの性能を効率的に利用するために，LSマップ機能とSPE制御レジスタ（MMIO（Memory-Mapped Input Output）領域）マップ機能を提供する。

LSマップ機能とMMIO領域マップ機能とは，SPEのLS及びMMIO領域をプログラムのアドレス空間上にマップすることで，他のSPEから直接参照できるようにする機能である。これらの機能を同時に活用することにより，**図5**に示すような

（注2）　Linuxは，Linus Torvalds氏の米国及びその他の国における登録商標。

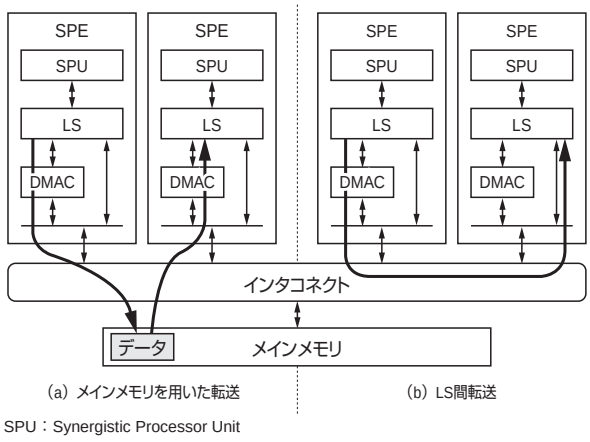

図5．LS間データ転送 ― メインメモリ経由で行う転送に比べ，高速にデータ転送が可能となる。
Data transfer between local storages (LSes)

LS間データ転送を実現することができる。

メインメモリを用いた通常のデータ転送では，送信側のSPEは，DMA（Direct Memory Access）コントローラ（DMAC）を用いてデータをLSからメインメモリに転送する。その後，受信側のSPEはDMACを用いてデータをメインメモリからLSに転送する。このため，メインメモリに対するアクセスが集中しやすく，また，SPE間の同期もメインメモリを用いて行う必要があるため，メインメモリのバンド幅がボトルネックになりやすいという問題がある。

一方，LS間転送では，送信側のSPEはデータを自LSから受信側のLSに直接転送することができる。このため，メインメモリのバンド幅に縛られないだけでなく，転送が1回で終了するため高速にデータ転送を行うことができる。更に，転送終了などの同期を行う際にも，SPEのMMIO領域に用意されたシグナル通知機能を活用できるため，SPE間の同期も高速に行うことができる。

4.4　複数処理同時実行

複数処理の同時実行は，時分割共有SPEを用いたスケジューリング予約機能を用いることで，容易に実現することができる。スケジューリング予約機能とは，指定された予約どおりにSPEスレッドを実行する機能である。

各アプリケーションは，SPEスレッドを含んだソフトウェアエンジンの実行を開始する前に，その実行に必要なSPEの処理能力などを指定したスケジューリング予約情報を生成する。スケジューリング予約情報は，階層型プログラミングモデルのPPE/SPEモジュール層の構成に準じているため，容易に生成することができる。アプリケーションは，生成したスケジューリング予約情報を用いてスケジューリング予約要求を行う。

スケジューリング予約要求は，SPEマネージャに組み込まれたリアルタイム リソース スケジューラによって処理される。

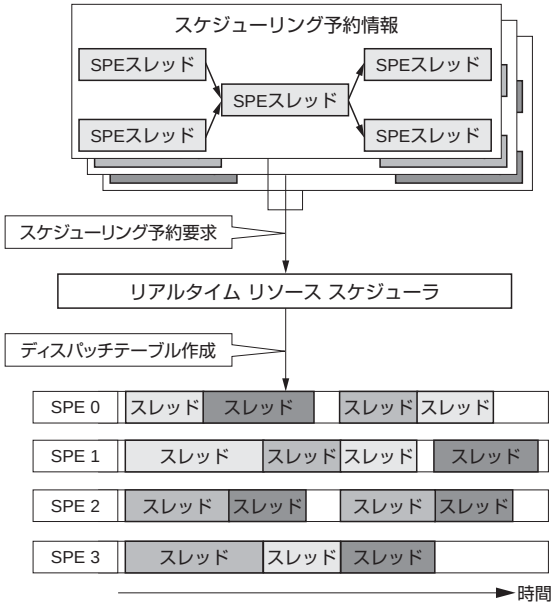

図6．リアルタイム リソース スケジューラ ― 複数のアプリケーションからスケジューリング予約要求を同時に受け付け，すべてのスケジューリング予約を満たすディスパッチテーブルを自動生成する。
Real-time resource scheduler

図6に示すように，リアルタイム リソース スケジューラは複数のアプリケーションからのスケジューリング予約要求を受け付け，各スケジューリング予約に含まれるすべてのSPEスレッドが，リアルタイム性を維持して正しく動作する処理順序を自動計算する。このとき新しい要求を受け付けると，実行中の処理のリアルタイム性を維持できなくなる場合には，新しい要求を却下して既存の処理を保護する。

リアルタイム リソース スケジューラは，スケジューリング結果をディスパッチテーブルとして出力する。ディスパッチテーブルは，高速動作可能なハイパーバイザOSであるBeatに送られ，SPEスレッドの切替えを安定して実行する。

このように，リアルタイム リソース スケジューラが，全アプリケーションからの要求を一括して自動計算し，更に，BeatにてSPEスレッドの切替えを実行することによって，複数処理の同時実行を行った際にも，すべての処理を安定動作させることが可能になる。

4.5　LSより大きいプログラムの実行

SPEプログラムが，256KバイトのLSより大きい場合でも，SPEオーバレイ機能を活用することで，容易に実行することができる。

図7に示すように，SPEオーバレイ機能では，プログラムを主要部分（図7のメイン），ライブラリ，各関数などに分割し，メインメモリ上に配置する。

プログラムの主要部分やライブラリなど恒常的に実行する

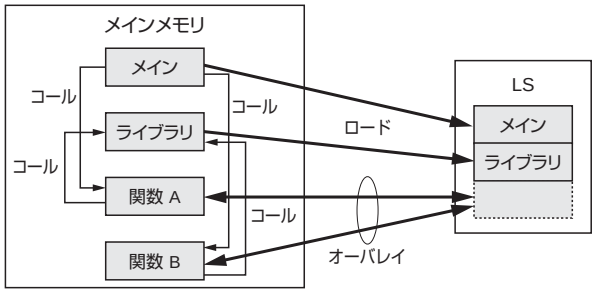

図7．SPEオーバレイ ― 関数Aと関数Bは，プログラム実行の進捗に合わせ，LS上の同じメモリ領域で入れ替わりながら実行する。その際にも，ライブラリの呼出しができる。

SPE overlay

部分は，あらかじめLS上にロードする。一方，関数など一時的に利用する部分は，その実行タイミングでLS上の同一領域に入れ替えながら実行する。入替え処理にはDMACを用いており，入替え完了までSPEは他の処理を継続できる。また，オーバレイされたプログラムからLS上に配置したライブラリを呼び出すことができる。このように，通常の分割コンパイルと同様な機能を備えているため，既存プログラムを容易にSPEオーバレイ対応させることができる。

5　あとがき

SPE実行環境は様々な利用形態を想定し，それぞれに必要となる機能を提供することで，SPE用プログラム開発を多方面からサポートしている。SPE実行環境を利用することによって，単一処理の高速化を目指した並列処理や複数処理の同時実行を容易に実現できると言える。今後は，ユーザーからのフィードバックを考察し，更なる性能改善や機能追加を行っていく。

文　献

(1) Maeda, S., et al. A CELL Software Platform for Digital Media Application. Proc. Cool Chips VIII, IEEE Press, 2005, p.453－464.
(2) Sakai, R., et al. Programming and Performance Evaluation of the Cell Processor. Proc. Hot Chips 17, IEEE Press, 2005, Session I.
(3) Maeda, S., et al. A Real-time Software Platform for the Cell Processor. IEEE micro, IEEE Press, 2005, p.20－29.
(4) 前田誠司，ほか．ヘテロマルチコアプロセッサCell上でのスレッド実行環境．情報処理学会誌“情報処理”．**47**，1，2006，p.34－40．

前田　誠司　MAEDA Seiji

研究開発センター コンピュータ・ネットワークラボラトリー 研究主務。分散リアルタイムシステムの研究・開発に従事。情報処理学会，IEEE会員。
Computer and Network Systems Lab.

佐藤　記代子　SATO Kiyoko

研究開発センター コンピュータ・ネットワークラボラトリー。メモリ管理技術の研究・開発に従事。
Computer and Network Systems Lab.

川上　　健　KAWAKAMI Ken

セミコンダクター社 ブロードバンドシステムLSI事業統括部 ブロードバンドシステムLSI開発センター。ブロードバンドシステムLSIに関するソフトウェア開発・設計に従事。
Broadband System LSI Div.