「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」事後評価第一回分科会
事業原簿（公開）【資料５－２】

「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）」
「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」

事業原簿
（公開）

担当部	独立行政法人新エネルギー・産業技術総合開発機構 電子・材料・ナノテクノロジー部

―目次―

概　要

	最終更新日		平成25年4月25日

プログラム（又は施策）名	ITイノベーションプログラム・ナノテク・エネルギーイノベーションプログラム		
プロジェクト名	「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」	プロジェクト番号	P09003
担当推進部/担当者	電子・材料・ナノテクノロジー部　担当者：田崎　英明　（平成22年12月～平成24年2月） 電子・材料・ナノテクノロジー部　担当者：畠山　敦　（平成24年3月～平成24年12月） 電子・材料・ナノテクノロジー部　担当者：高井　伸之　（平成25年1月～平成25年2月）		

Ⅰ.事業の位置付け・必要性について

0. 事業の概要	情報通信機器や車載機器等の高度化・設置数の急激な増加に伴い、情報処理量とエネルギー消費量の増大が見込まれ、これらの機器に組み込まれるプロセッサの高性能化・高機能化だけでなく低消費電力化が重要な課題となっている。そこで、多数のプロセッサコアをワンチップに搭載したメニーコア・プロセッサ(*)によって、この課題を解決することが有望と考えられているが、平成21年度のメニーコア・プロセッサ先導研究で得られた知見によれば、高性能・高機能かつ低消費電力のメニーコア・プロセッサの実現には、半導体集積回路（LSI）設計技術のみならず、ソフトウェアによる周波数・電圧等のきめ細かい電力制御を行うことが必要である。コア数が増えるに伴い、人間がアプリケーションプログラムの中に電力制御の仕組みを組み込んでいくことは非常に困難が伴うため、API（コンパイラへの指示）等を用いたコンパイラ技術の開発が必須である。 (*)コア数が32～64以上を指す。
Ⅰ．事業の位置付け・必要性について	新・国家エネルギー戦略（2006年5月経済産業省）では、2030年に30%以上のエネルギー消費効率の改善を目標として掲げている。本目標を達成するためには、情報通信機器の 3 次元半導体実装、微細化／高密度化、電力回路設計等ハードウェア技術開発が必要であるが、本分野でも限界が見えだしてきている状況を鑑みて、本プロジェクトでは、効率化、省エネルギー化を実現する1アプローチとして複数プロセス構成で性能を向上させたマルチコア・メニーコアのソフトウェア制御技術を開発する。（ハードウェア技術開発は「極低電力回路・システム技術開発」（研究開発項目①～⑥）をテーマとする別プロジェクトとして取り組んだが、これは本プロジェクト範囲外とする。

Ⅱ．研究開発マネジメントについて

事業の目標	あるべき低消費電力メニーコア用アーキテクチャを提案し、開発するコンパイラ技術を用いて既存技術と比べて電力当たりの処理性能2倍を達成する。かつ、メニーコア・プロセッサ上で組込み向けアプリケーションプログラム実行時の電力消費量を1／10以下にする。

事業の計画内容	主な実施事項	2010	2011	2012
	①　メニーコア基盤技術開発 （メニーコア・アーキテクチャ設計・実装評価）	→	→	→
	②　メニーコア応用技術開発 （ビデオマイニング向けメニーコア開発）	→	→	→
	③　メニーコア応用技術開発 （ソフトウェア開発環境開発）	→	→	→

<table>
<tr><td rowspan="5">開発予算
（会計・勘定別に事業費の実績額を記載）
（単位：百万円）</td><td>会計・勘定</td><td>合計</td><td>2010</td><td>2011</td><td>2012</td></tr>
<tr><td>一般会計</td><td>－</td><td>－</td><td>－</td><td>－</td></tr>
<tr><td>特別会計
（需給）</td><td>370</td><td>20</td><td>150</td><td>200</td></tr>
<tr><td>加速予算</td><td>－</td><td>－</td><td>－</td><td>－</td></tr>
<tr><td>総予算額</td><td>370</td><td>20</td><td>150</td><td>200</td></tr>
<tr><td rowspan="3">開発体制</td><td colspan="2">経産省担当原課</td><td colspan="3">商務情報政策局情報通信機器課</td></tr>
<tr><td colspan="2">プロジェクトリーダー</td><td colspan="3">九州大学　井上　弘士　准教授（PL）
立命館大学　冨山　宏之　教授（サブPL）</td></tr>
<tr><td>委託先／再委託先</td><td colspan="4">企業、研究機関：（国）九州大学、（学）立命館大学、（国）電気通信大学、
（株）トプスシステムズ、（株）フィックスターズ
再委託先：（社）JEITA、キャッツ（株）、イーソル（株）、（国）東京農工大学</td></tr>
<tr><td>情勢変化への対応</td><td colspan="5">CEATEC2011、CEATEC2012 の NEDO ブースにメニーコア成果の展示、メニーコア成果の発表を行い、世界で初めの 128 メニーコアの実証評価結果をアピールした。
また「メニーコア・シンポジウム」を年一回（2011 年 3 月、2012 年 2 月）に開催して、外部有識者、ユーザの意見を聴取してプロジェクト運営管理に反映した。
　具体的には 2011 年3 月のメニーコア・シンポジウムでは参加したトヨタ自動車、デンソー等のユーザ部門の技術者と意見交換を行い、メニーコアの適用用途として、車載向けビデオマイニング用途のアイデアを得て、ビデオマイニングデモシステムの開発を行った。本成果はCEATEC2011、CEATEC2012 にて披露して関係者に情報発信した。</td></tr>
<tr><td rowspan="3">評価に関する事項</td><td colspan="2">事前評価</td><td colspan="3">平成22年度実施</td></tr>
<tr><td colspan="2">中間評価</td><td colspan="3">なし</td></tr>
<tr><td colspan="2">事後評価</td><td colspan="3">平成25年度　事後評価実施</td></tr>
<tr><td rowspan="4">Ⅲ．研究開発成果について</td><td colspan="5">● メニーコア・アーキテクチャ開発
メニーコアにより、多種多様な SoC（Systemon　Chip）に適用可能なプラットフォーム（通称：SMYLEref）を設計開発した。本アーキテクチャに基づき、128 コアを搭載した FPGA プロトタイプを開発してその実現性を実証／評価した。本プロトタイプ上で、並列化された単一アプリケーションを順次実行する従来の方式と比較して、消費エネルギーの 1/10 削減を実証した。
● ビデオマイニング向けメニーコア開発
今後の需要拡大が見込まれるビジュアルコンピューティング（特に動画像認識）向けにメニーコアチップ（通称：SMYLEvideo）を設計した。本チップベースでシミュレーションの結果、従来型の汎用マルチコア・プロセッサと比較して、性能が約 30 倍（700GOPS 相当）、消費電力1/300 以下という大きな優位性があることを実証した。
● メニーコア用ソフトウェア開発環境開発
本メニーコア用ソフトウェア開発環境として3つのサポートツールを開発した。第 1 は、前述した SMYLEref での利用を想定した C-to-OpenCL 半自動ソースコード変換ツール、第 2 はメニーコアシステムにソフトウェアを移植した際の性能を推定するツールを開発した。第 3 は、メニーコアの評価やプログラム最適化知識の共有を目的としたベンチマークを開発した。本ツールにより、メニーコアの移行・普及を促進する環境を形作った。</td></tr>
<tr><td>投稿論文</td><td></td><td colspan="3">39件（内 「査読付き」11件）</td></tr>
<tr><td>特　許</td><td></td><td colspan="3">4件（内「登録済」0件）</td></tr>
<tr><td>その他の外部発表
（プレス発表等）</td><td></td><td colspan="3">28件（内　外部発表　28件、プレス他　0件）</td></tr>
<tr><td>Ⅳ．実用化・事業化に向けた見通しについて</td><td colspan="5">● メニーコア基盤研究で開発した成果を生かして、車載の衝突回避用ビデオマイングシステムの IP 販売事業（略称：SMYLEVideo）を 2013 年より販売を開始した。本 SMYLEvideo はスマートフォン・タブレット、デジタルカメラ、デジタルテレビ、自動車、監視装置、ロボットなど、次世代の様々なスマートな情報機器への応用が期待され現在、実用化開発について、顧客と商談を進めている。（（株）トプスシステムズ）</td></tr>
</table>

	● メニーコア用ソフトウェア開発環境は今後の需要が高く、かつ、高速処理のためのソフトウェアの最適化が強く要求されるであろう4 分野（ライフイノベーション、インターネット・アプリケーショ ン、ファイナンス、ビジュアルコンピューティング）を狙ってOpenCLソースコード、最適化 による性能向上を評価したレポートを無償公開を既に開始した（http://sourceforge.net/projects/bemap/）。これらの成果に関しては、メニーコア向けベンチマークとしての普及を期待することができる。また、フィックスターズ においては、これら公開済みプログラムをサンプルとした有償での組み込みサービスのビジネスを展開しライセンス販売も展開する。（（株）フィックスターズ）	
Ⅴ．基本計画に関する事項	作成時期	平成21年3月　制定訂
	変更履歴	平成22年8月　本プロジェクトの研究開発項目⑦の追加

プロジェクト用語集

No.	用語	意味・説明
1	L1 キャッシュ	マイクロプロセッサ内部に設けられた高速記憶装置。使用頻度の高いデータを記憶しておくことにより、低速なメインメモリへのアクセスを減らし処理を高速化する。階層メモリにおいて、CPU にもっとも近い位置（CPU が最初にデータ参照を試みる位置）のキャッシュを L1（レベル 1）キャッシュと呼ぶ。
2	L2 キャッシュ	階層メモリにおいて、L1 キャッシュの下層に位置するキャッシュメモリである。 通常は L1 キャッシュより大容量かつ低速となる。
3	プロセッサコア	マイクロプロセッサであり、プログラムを実行する機能を有する。複数のマイクロプロセッサが 1 個の LSI に搭載されるマルチコアにおいて、1 個のマイクロプロセッサを「プロセッサコア」と呼ぶ。
4	コア	プロセッサコアの略称。
5	マルチコア	複数のコアを 1 個の LSI に集積したマイクロプロセッサ・チップ。一般的には 4～8 個のコアを搭載する場合が多いが、高性能サーバ向けでは 16 個のコアを搭載する場合もある。
6	メニーコア	多数のコアを 1 個の LSI に集積したマイクロプロセッサ・チップ。一般には 64～128 個以上のコアを搭載する。コア間通信は NoC（Network-on-Chip: LSI チップ上に実現されたネットワーク）上でパケット単位で実現する場合が多い。現在、64 個や 100 個程度のメニーコアのチップ化が報告されている。
7	スレッド	プログラムの実行における処理の分割単位。スレッド間でメモリ空間を共有しており、依存関係が無い場合は並列に実行することができる。タスクは 1 個以上のスレッドで構成されることになる。
8	ヘテロジニアスコンピューティング	種類の異なる複数の実行ユニットを用いたコンピューティング。
9	ホモジニアスコンピューティング	同一種類の複数の実行ユニットを用いたコンピューティング。
10	ベンチマーク・プログラム	コンピュータシステムの性能を測定するためのプログラム。
11	レイテンシ	遅延時間。例えば、メモリ読出しレイテンシと言った場合には、メモリに対してアクセスを開始してから読出しが完了するまで（読出したデータを取得するまで）に要する時間となる。

NO	用語	意味・説明
12	仮想アクセラレータ	本プロジェクトで定義した用語。VAM(Virtual Accelerator on Manycore)と略す場合もある。メニーコア上に実装された 1 個以上のコアで構成され、1 つのプログラムまたはタスクを実行する。実行対象プログラムまたはタスクの特性(例えば、内在する並列性など)に応じて、その構成(例えば、使用するコア数など)を設定できる。
13	BEMAP	本プロジェクトで開発したメニーコア向けベンチマークプログラム集。
14	PEMAP	本プロジェクトで開発した、ソフウェア移植後の性能を見積るためのツール。
15	CLTrump	本プロジェクトで開発した、逐次 Cソースコードから並列化した OpneCL ソースコードを生成するコンパイラ。
16	FPGA	Field Programmable Gate Array の略。書き換え可能な LSI。
17	GPU	Graphics Processing Unit の略。パーソナルコンピュータやワークステーション等に組み込んでグラフィックス処理を高速に行う装置。
18	OpenCL	Open Computing Language の略。ヘテロジニアス・コンピューティングによる並列処理を前提としたプログラム開発を効率的に行うためのプログラミング・モデルと実行フレームワーク。
19	SoC	システム・オン・チップの略。プロセッサ・コア、メモリ、周辺回路、専用回路、ソフトウェア、など、システム構成要素を 1 個の LSI に集積する。
20	SMYLERef	本プロジェクトで開発した、汎用メニーコア SoC を実現するためのメニーコア・アーキテクチャ。
21	SMYLEVideo	本プロジェクトで開発した、ビジュアル・コンピューティング(特に動画像認識)向けメニーコア・アーキテクチャ。

Ⅰ．事業の位置付け・必要性について

1．NEDO の事業としての妥当性

1-1 政策上の位置づけ

新国家エネルギー戦略（2006年5月経済産業省）では、2030年に30%以上のエネルギー消費効率の改善を目標として掲げている。また「第三期科学技術基本計画」においてもＩＴ技術による省エネルギー技術開発の重要性を提唱している。

現在、エネルギー資源の約8割を海外に依存する我が国にとって、エネルギーの効率的利用、即ち「省エネルギー」を図ることは、エネルギー政策上の重要な課題である。新国家エネルギー戦略の元、省エネルギー技術の開発・導入を進め、もって我が国におけるエネルギーの安定供給の確保を図ることを目的として「エネルギーイノベーションプログラム」が策定された。また21世紀の革新的技術として情報、環境、安全/安心、エネルギーなどの広範な分野のＩＴ技術を開発して電力制御することにより、機能・特性の向上や新機能の発現を図る基盤技術を構築し、ＩＴの利活用の深化・拡大を図り、より豊かな国民生活を実現するとともに、我が国の経済活力の向上を図ることを目的として「ＩＴイノベーションプログラム」が策定された。

「エネルギーイノベーションプログラム」での総合エネルギー効率の向上の一環で情報通信機器の省エネルギー化施策として、また「ＩＴイノベーションプログラム」での情報通信機器による消費電力量の大幅な増大に対応し環境調和型ＩＴ社会の構築を図るため、個別のデバイスや機器に加え、ネットワーク全体での革新的な省エネルギー技術の開発の具体化施策として本プロジェクトを発足させた。

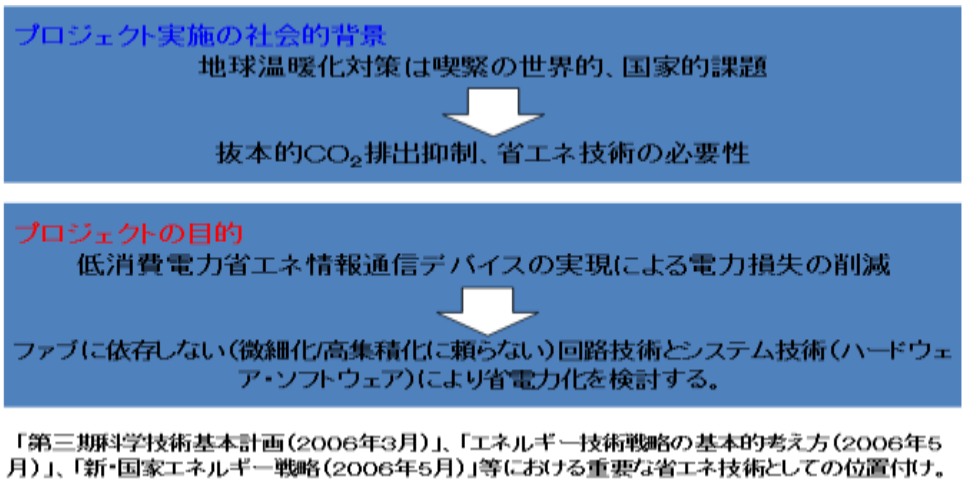


図1-1．プロジェクト実施の社会的背景

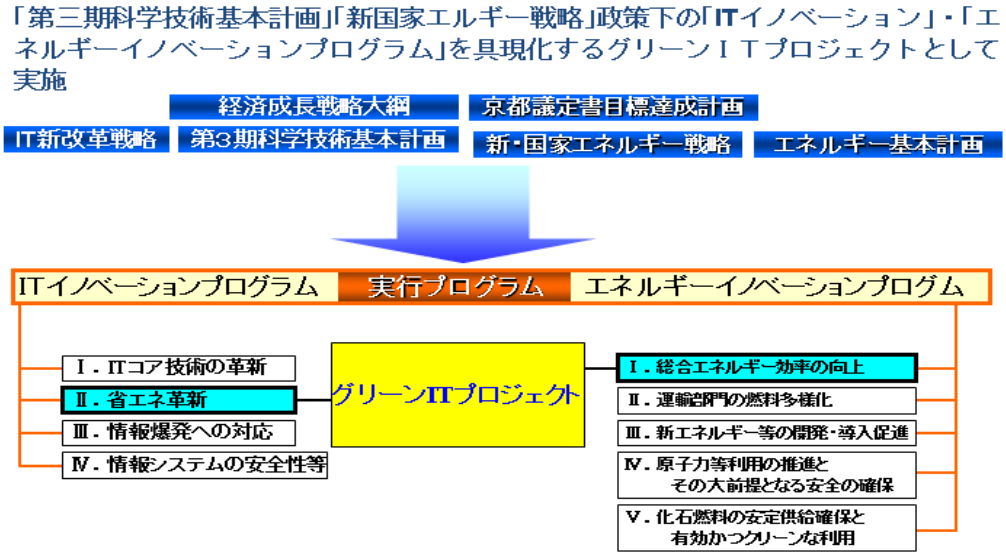


図 1-2．国の政策上の本プロジェクトの位置づけ

現在は情報通信技術の進展により、中核から末端まで情報通信技術を適用した情報通信機器が普及している。特に情報通信機器の高度化・設置台数の急激な増加に加え、ブロードバンド通信の普及、情報サービスの拡大等により多種多様なマルチメディアデータや動画コンテンツなど大容量データが、ネットワーク情報端末を介して流れ社会で扱う情報量は急激に増大しつつある。今後、省エネルギー消費効率目標を達成するためには、社会に普及したこれらの情報通信機器の低消費電力化が求められる。また地球温暖化の観点からもCO_2排出抑制のため、情報通信機器の省エネルギー化が求められる。情報通信機器の低消費電力化の実現に向けては半導体集積回路（ＬＳＩ）の低消費電力化、ロジックやメモリなど構成回路の極低電圧化はもちろん、電源電圧をきめ細かく制御可能な電源、ＬＳＩチップと外部との各種Ｉ／Ｏインタフェースなど、ＬＳＩでの実用化に向けた様々な回路技術、インタフェース設計技術等のハードウェア上の対策が必要である。低消費電力対策としてはこれらのハードウェア対策だけに頼らず、プロセッサコアを中心としたシステムの消費電力削減を図るにはソフトウェアによる省電力化技術による省エネルギー化対策も施策することも重要と考えられる。

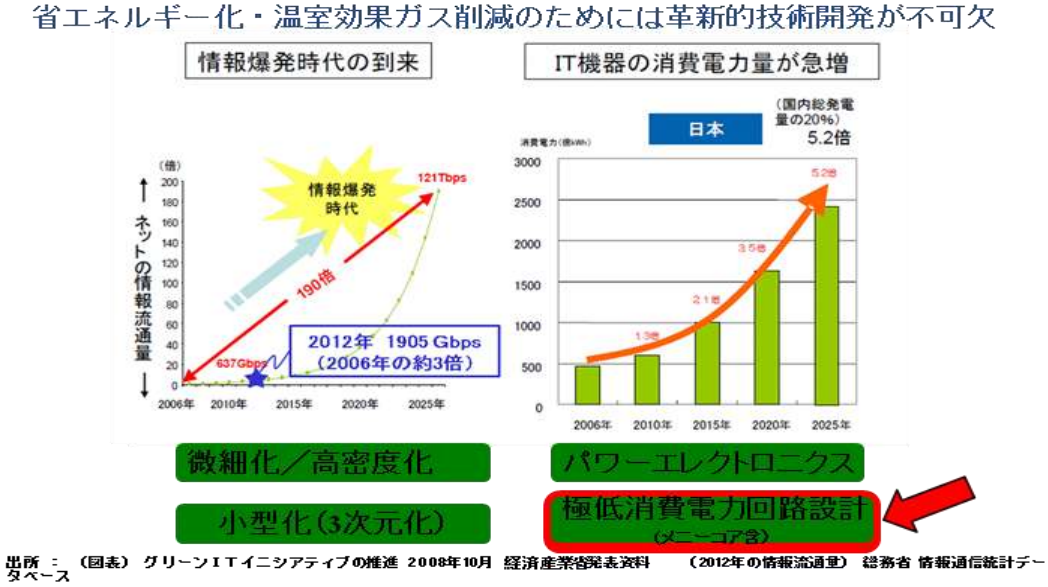


図 1-3．省エネルギー化に向けた確認的技術開発の必要性

1-2　NEDO が取り組むべき意義

世界の至る所に利用される情報通信機器についての省エネルギー化の実現は、低炭素社会の実現に向けて社会的にも重要であり、国家にとって解決すべき課題である。

このような省エネルギー化された情報通信機器を実現する研究開発は難易度が高く、且つ大掛かりな投資規模が必要なため、開発リスクが高い。また市場が不確定な状況での開発リスクの高い大規模な研究投資が要求される。

そのため本課題については国家プロジェクトとして取り組む必要があると判断される。革新的な省電力化技術を効率的に開発するためには、企業単独で取り組むよりは、共通の課題を抱え異なる技術力を持つトップレベルの研究ポテンシャルを有する企業・大学が連携し、産学連携による総合的な取り組みが求められる。このような産学連携による技術開発を行うことによって、その成果を適用した低消費電力で高性能な情報通信機器の実用化及び普及拡大が期待され、国内企業の国際競争力の強化にもつながる。国家プロジェクトとして大学、企業が相互連携して各々の技術を結集して低消費電力情報通信機器システム革新を推進することが効果的且つ効率的と考えられる。

このような情報通信機器の省エネルギー化が実現できれば、デバイス事業においても、システムインテグレーション事業、サービス事業等の関連事業においても企業競争力強化につながる。さらにその応用分野としては自動車産業、医療機器産業等へも拡大が期待できる。

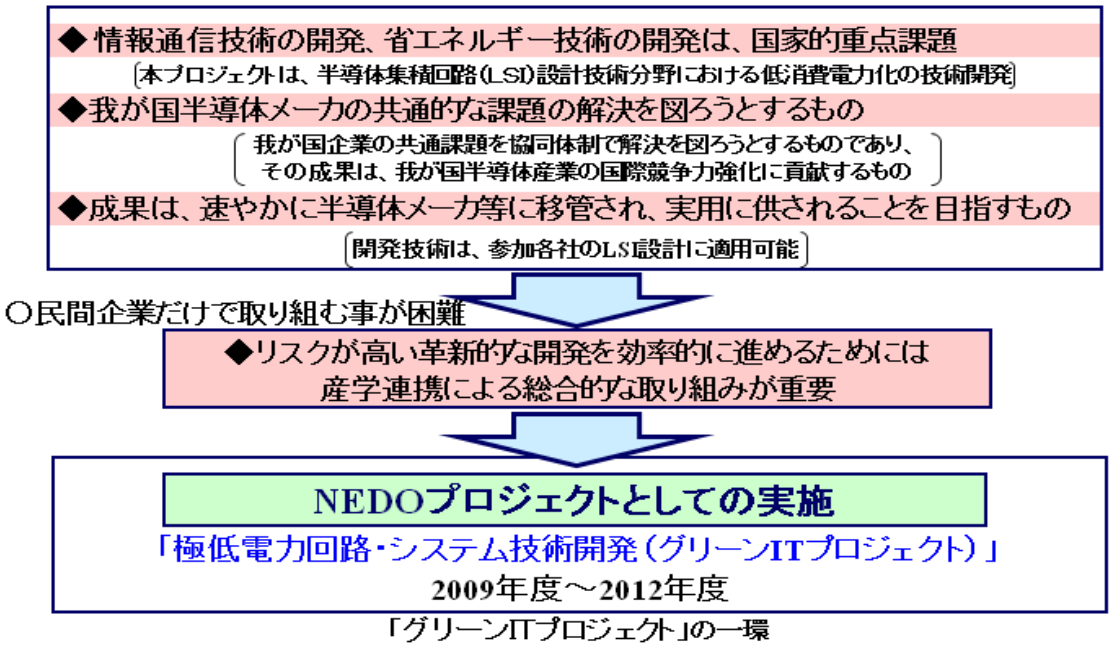


図 1-4. NEDO で取り組むべき必要性

1-3. 事業の背景・目的・位置づけ

前述したとおり情報通信機器の低消費電力化は国家的課題である。それを実現するソリューションとして従来の半導体技術の延長としてチップの高密度化・LSIの低消費電力化等を進めるハードウェアアプローチに頼るだけではなく、ソフトウェアやプロセッサアーキテクチャによるアプローチにより低消費電力化も考えられる。ハードウェアによるアプローチは、別途ＮＥＤＯ「極低電力回路・システム技術開発」の①～⑥の研究項目の技術開発事業にて取り組むが、本プロジェクトでは、「極低電力回路・システム技術開発」研究項目⑦としてソフトウェア技術開発を用いたアプローチによる低消費電力化を実現する基盤技術開発に取り組む。

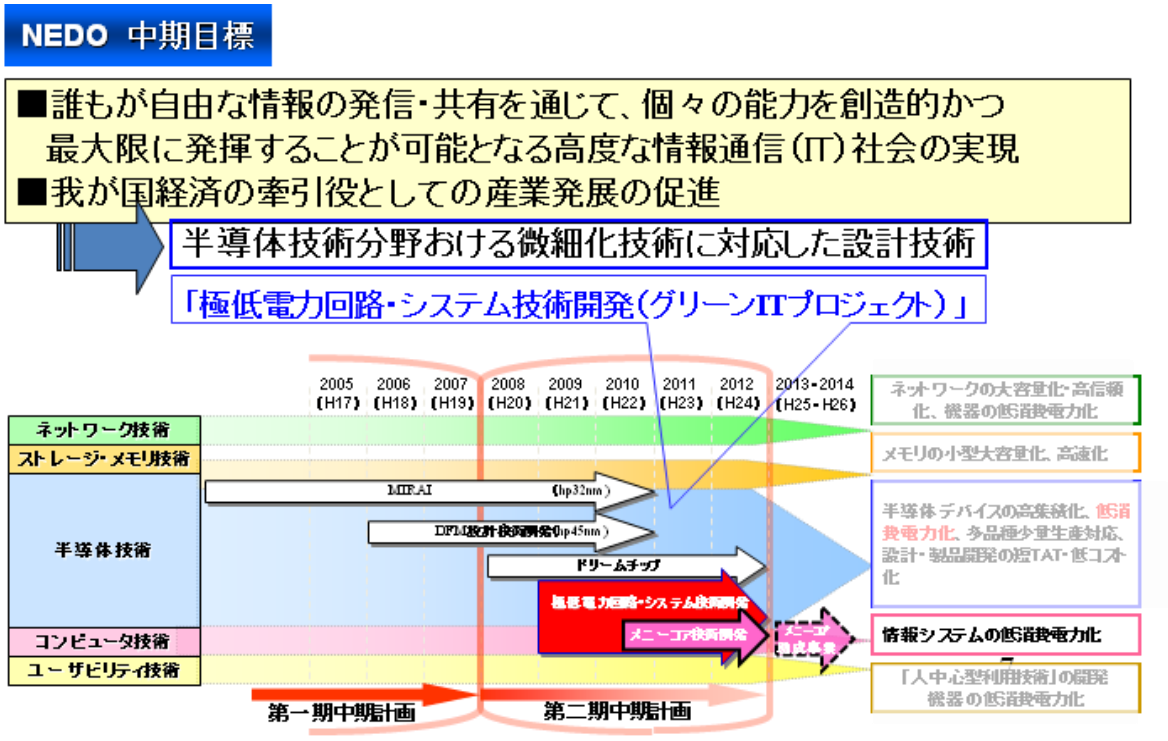


図 1-5. NEDO 中期目標での位置づけ

本技術開発の成果を社会に浸透する情報通信機器に適用することにより、自動車分野、携帯電話分野、、医療分野、基幹通信分野など社会全体の幅広い分野での効果が期待できる。

1-4 プロジェクト実施により期待される効果（費用対効果）

本事業を実施し、従来の技術では実現できない、高効率な情報通信機器や、それらを組み合わせた情報通信システムを構築する基盤技術の確立を行うことにより、日本の半導体産業、情報機器・家電機器企業、関連企業等の競争力強化と産業活性化とともに、国内エネルギー消費量削減に大きく貢献することが期待される。

本事業の実施効果として、例えば車載組込機器、通信機器に本事業の開発成果を適用した場合の効果を以下に示す。

車載ＥＣＵ機器に本事業成果を適用した場合を想定すると、2020 年には、6.5 億 kWh/年の電力消費が想定されるが、本事業の効果により３億ｋWh/年の省ネルギー効果が期待できる。

また通信機器に本事業成果を適用した場合を想定すると、2025 年には、168 億 kWh/年の電力消費が想定されるが、本事業の効果により、52 億ｋWh/年の省ネルギー効果が期待できる。（概ね原発 1.5 基分(*4)。

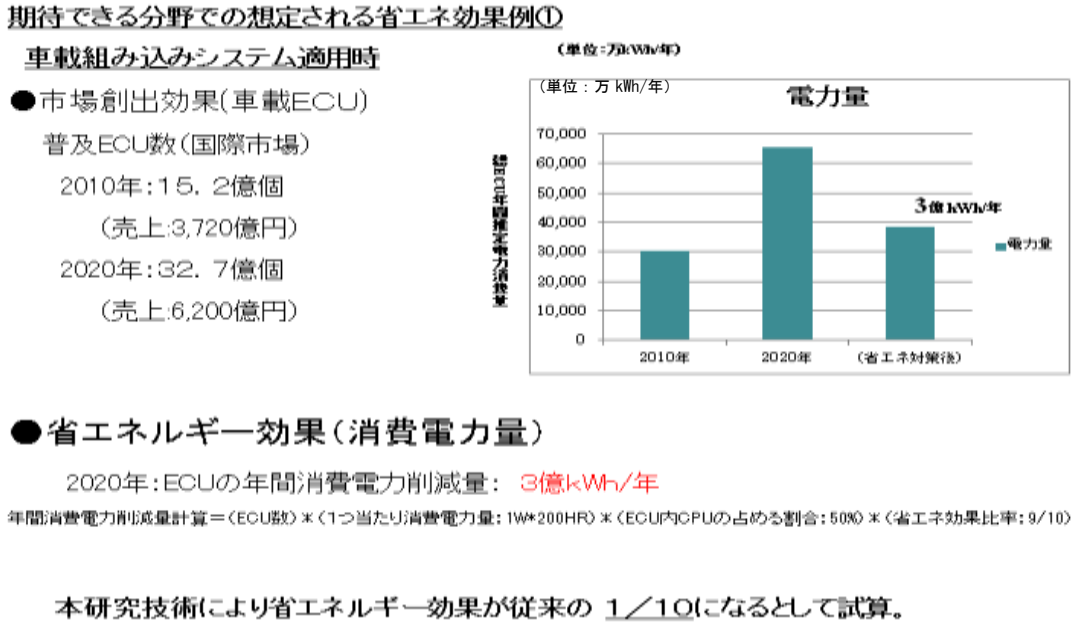


図 1-6.　プロジェクト実施により期待される効果例①：車載組み込みシステム応用

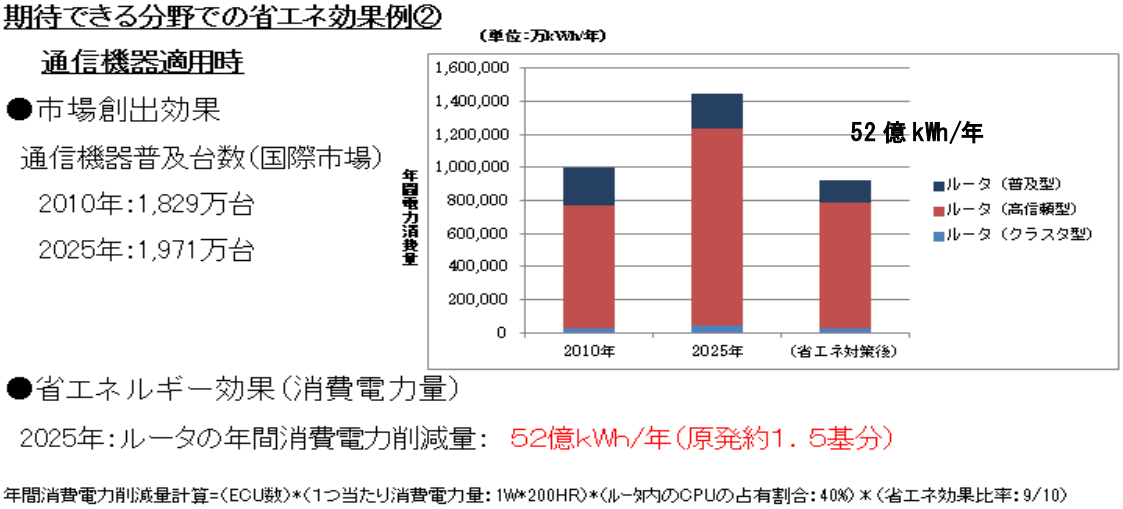


図 1-7.　プロジェクト実施により期待される効果例②：通信機器応用

(*1)算出根拠：：
省エネルギー効果は、①普及台数*②一台当たりの消費電力量*③システム内ﾌﾟﾛｾｯｻ電力消費比率*④省エネルギー効果　にて計算する。
上記中、④については、本紙Ⅱ.1事業の目標より、90%（省エネ効果 1/10以下）　を適用。

上記を前提にして以下のように試算する。
●車載チップについては、国際市場で
2010年：①自動車ECU台数：1,517,300千台　(*6)
2020年：　自動車ECU台数：3,273,600千台（*6)
②　各自動車の年間走行時間：200時間と想定。
ECUの平均消費電力を 1Wと想定。(*1)（*7)
ECUの中で③CPUの電力消費の占める割合：50%　と想定。(*3)

想定 ECU電力消費量は（①*②）：により
2010年：1,517,300千台*200Wh　=　30,346万kWh/年
2020年：3,273,600千台*200Wh　=　65,472万kWh/年

そのうち、2020年においてのﾏﾆｰｺｱ適用による省エネ効果（①*②*③*④）
＝65,472万k Wh/年*50%%*0.9　=29,462万k Wh

●通信機器については国際市場で
2010年：　ｸﾗｽﾀ型ﾙｰﾀ台数：5千台　高信頼型ﾙｰﾀ台数：330千台、普及型ﾙｰﾀ：1,494千台　(*2)
2025年：①ｸﾗｽﾀ型ﾙｰﾀ台数：5千台　高信頼型ﾙｰﾀ台数：303千台、普及型ﾙｰﾀ：1,663千台　(*2)
ﾙｰﾀ中で③CPUの電力消費の占める割合：40%　と想定。(*3)

2025年の年間ﾙｰﾀﾀｲﾌﾟ別消費電力量：
2010年：　ｸﾗｽﾀ型ﾙｰﾀ：52,560ｋW　高信頼型ﾙｰﾀ：22,601Kwh、普及型ﾙｰﾀ：1,511Kwh　(*2)
2025年：②ｸﾗｽﾀ型ﾙｰﾀ：91,104ｋW　高信頼型ﾙｰﾀ：39,175Kwh、普及型ﾙｰﾀ：1,261Kwh　(*2)
2010年の通信機器の年間のルータ消費電力量：
（ｸﾗｽﾀﾀｲﾌﾟﾙｰﾀ：5千台*52,560kWh/年）＋（高信頼ﾙｰﾀ：330千台*22,2,601kWh/年）
＋（普及ﾀｲﾌﾟﾙｰﾀ：1,494千台*1,511kWh/年）
＝　100億 kWh/年。

2025年の通信機器の年間のルータ消費電力量（①*②）：
（ｸﾗｽﾀﾀｲﾌﾟﾙｰﾀ：5千台*91,104kWh/年）＋（高信頼ﾙｰﾀ：303千台*22,39,175kWh/年）
＋（普及ﾀｲﾌﾟﾙｰﾀ：1,663千台*1,261kWh/年）
＝　144億 kWh/年。

その内、本事業の省エネ効果：(①*②*③*④)　　144億 kwh/年　0.40*　9/10　=　52億　kwh/年。

(*1)「大原雄介　カーWathc「ECUとは」(2011年9月)
http://car.watch.impress.co.jp/docs/series/cew/20110907_475672.html
(*2)「平成24年度　我が国情報経済社会における基板整備（情報通信機器のエネルギー消費量に関わる調査」：(株) NTTデータ研究所p51、p61より引用。
(*3)　車載 ECU内 CPUは 50%、ルータ内 CPU比率は 40%と想定して計算。
(*4)「2009年度(平成21年度)福島第一原子力発電所」(1号機データ」(37億 kWh/年) 東京電力　より引用。
(*5) ※原油換算値は 2.54E-4kL/kWh、CO2トン換算値は 0.555kg-CO2/kWh換算。
(*6)「車載 ECU ｱﾅﾗｲｼﾞﾝｸﾞ&ﾏｰｹｯﾄﾚﾎﾟｰﾄ2010」富士キメラより引用
(*7)「富士通テン技報」 Vol.23　富士通　より引用

Ⅱ．研究開発マネジメントについて

１．事業の目標

本メニーコアプロジェクトは、「極低電力回路・システム技術開発」プロジェクトによる省電力化研究開発の研究項目の７番目の研究項目である。「極低電力回路・システム技術開発」プロジェクトの６つの研究項目（注）では、ロジック回路技術開発、メモリ回路技術開発などハードウェア技術開発アプローチにより低消費電力化をテーマに取り組んでいる。ただし 2009 年度に行った先導研究で得られた知見によりフトウェア技術による低消費電力化施策として高性能・高機能かつ低消費電力の施策としてメニーコアが有効であることがわかったため、７番目の研究項目としてソフトウェアという異なる観点から目標達成を目指して本「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」開発に取り組んだ。設定目標としては、「極低電力回路・システム技術開発」プロジェクトの他のハードウェア的アプローチ（注）の研究項目にて設定された目標と整合させて、異なるソフトウェア的アプローチで以下の３つの目標達成を目指した。

－低消費電力メニーコアアーキテクチャを提案すること
－既存技術と比べて電力当たりの処理性能２倍以上を達成すること
－組み込み向けアプリケーション実行時電力消費量１／１０以下とすること

（注）プロジェクトとしてその他にハードウェア的アプローチによる省電力化技術開発（基本計画の研究開発項目①～⑥（ロジック回路技術開発、メモリ回路技術開発、アナログ回路技術開発、電源回路技術等）は技術的要素が異なるため、今回の事後評価の対象外とする。（研究開発項目①～⑥の事後評価分科会は別途 11/19（火）（開催済）。

研究開発項目	研究開発目標	目標設定の根拠
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	●あるべき低消費電力メニーコア用アーキテクチャを提案する ●既存技術と比べて電力当たりの処理性能２倍以上を達成する ●組み込み向けアプリケーション実行時電力消費量を１／１０以下にする	「極低電力回路・システム技術開発」プロジェクト中の極低LSIﾁｯﾌﾟ統合最適化技術の目標（電力消費量１／10以下）と同等の省エネルギー効果をメニーコアアプローチで実現を図る 同時に処理性能についてもユーザビリティ要求を考慮して電力当たり2倍以上の目標を設定（注）

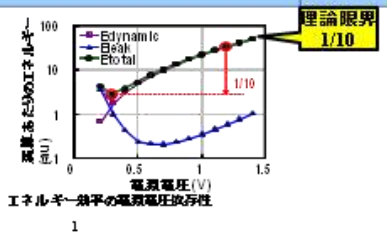


図 2-1. 目標設定の根拠

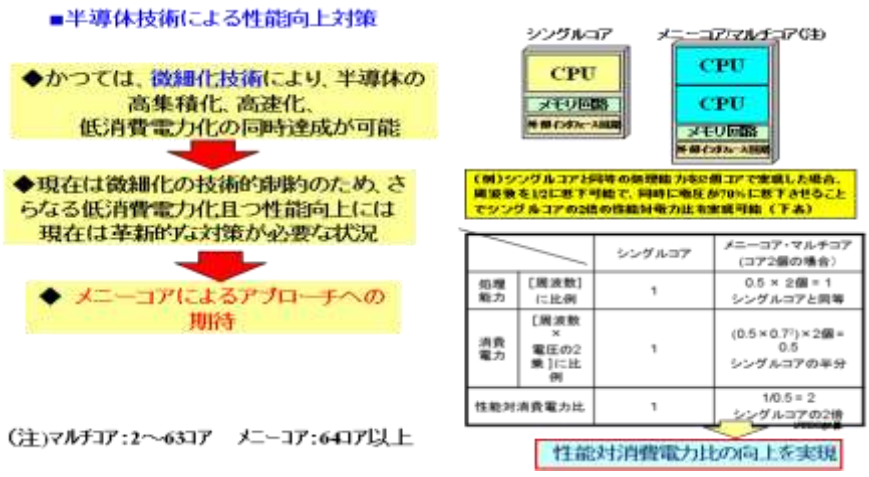


図 2-2. メニーコア化に取り組む意義と効果

メニーコアにより電力量に対する省エネルギー化に取り組むとともに、メニーコアにより期待される電力当たりの性能向上も目標として設定した。

なお、メニーコアを適用する理由を以下に述べる。かつては半導体の微細化技術により、半導体の高集積化、低消費電力化が飛躍的に向上してきた。しかし昨今の微細化技術の制約、限界があるために、さらなる性能向上を図るためには、ソフトウェア技術等革新的な技術の観点からの可能性も考慮した対策が望まれる。メニーコア方式による性能向上を実現するためには、ソフトウェアによる周波数・電圧等のきめ細かい電力制御を行うことが必要であることがわかっている。コア数が増えるに伴い、手動でアプリケーションプログラムの中に電力制御の仕組みを組み込んでいくことには非常な困難が伴うため、API（コンパイラへの指示）等を用いたコンパイラ技術の開発も合わせて行う。このようなアプローチにより、幅広い用途により省エネルギー化が期待されるメニーコアシステムの実用化事業化実現に貢献可能となる。

２．事業の計画内容

2-1　研究開発の内容

低消費電力メニーコア用アーキテクチャを検討し、組込み用のメニーコア・プロセッサ及び各種サーバ上でコンパイラの動作を可能とするAPI（コンパイラへの指示）を策定する。そのAPIを用いたコンパイラを開発し、組込み向けアプリケーションプログラムで評価する。評価結果をもとにアーキテクチャとAPIへのフィードバックを行うこととした。

併せて提案する技術を適用するアプリケーションの拡大に向けた検討に取り組む計画とした。

2-2　研究開発の実施計画

本事業は適宜体制の最適に変更しつつ、約２年３ヶ月に渡り、メニーコアの基盤技術、メニーコア応用技術に分けながら連携しつつ、研究開発する計画とした。

（注１）基盤技術開発：将来的な実用化を目指す革新的な基礎技術開発。
応用技術開発：基盤技術を活用しつつ2-3年以内の実用化を目指した技術開発
（注２）RTL：Register Transfer Level

図 2-3. プロジェクト開発工程と予算計画

なお、情勢の変化や市場調査・技術動向調査結果等に基づき、適宜課題対策・解決を行うこととした。

2-3 研究開発の実施体制

本研究開発は、独立行政法人新エネルギー・産業技術総合開発機構（以下、「ＮＥＤＯ技術開発機構」という。）が本邦の企業、大学委等の研究機関から公募によって研究開発実施者を選定し実施した。

研究開発に参加する各研究開発グループの有する研究開発ポテンシャルの最大限の活用により効率的な研究開発の推進を図る観点から、研究体にはＮＥＤＯ技術開発機構が委託先決定後に指名する研究開発責任者（プロジェクトリーダー。以下、PLと略す）を置いた。PLに関しては委託先で中心的な役割を果たし本分野で専門技術と知見を有する九州大学　井上准教授に委託した。

委託先としては研究者を可能な限り結集して、効果的な研究開発体制を採った。PL下には、基盤研究開発を主体として進める3大学と実用化に向けた応用研究開発を主体として進める２企業が参画する産学連携体制とした。

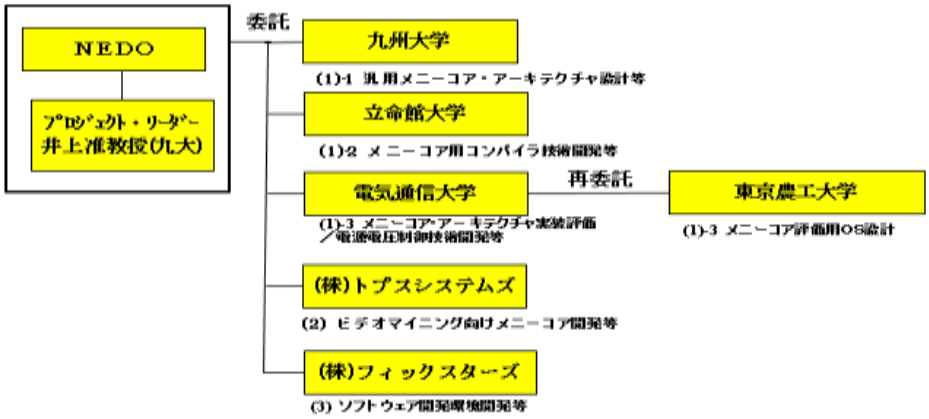


図2-4. 実施体制（全体）

2-4　研究開発成果の実用化・事業化に向けたマネジメント

（１）大学の技術開発から企業主体の技術開発に向けた技術移管の促進

本事業は、大きく分けて基盤技術開発と応用技術開発に渡る３つの課題に取り組んだ。メニーコア技術の基盤技術開発の課題については、適任と考えられるリーダとして九州大学が、応用技術開発の課題についてはリーダとして適任として考えられる２企業が就任し推進した。但し、大学の基盤技術開発成果を生かして速やかに実用化事業化できるように、大学が取り組む基盤技術開発の各課題に対しては大学のみならず、各企業も連携参画して取り組むように研究開発を進めた。また各企業が取り組む応用技術開発の各課題に対しても、企業が主体として進めるが、それを支援する形で大学が連携参画する進め方とした。このような連携を行うことにより、大学から企業への速やかな基盤技術の技術移管の促進と、実用化事業化に向けた応用技術開発の加速を図った。

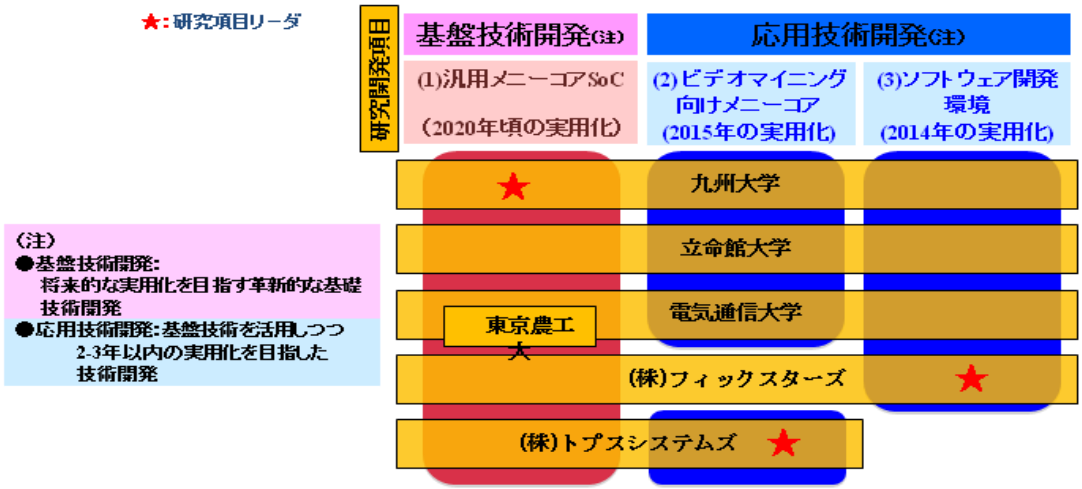


図2-5. 研究項目別実施体制

（２）実施者間の知財ルールに基づく知財管理

最も効率的且つ迅速に各企業が各社の事業戦略に沿って特許を利用できるように、日本版バイドール法により、特許はＮＥＤＯより各実施者に譲渡されるようにした。 なお特許出願後の状況の変更の都度、実施者よりＮＥＤＯに通知報告するルールとした。
また実施者間（３大学、２企業）は知財に関して共同研究契約書を締結した。本契約書に従い、単独機関の発明は単独所有、複数機関の発明は持分に応じた共用所有とすることにした。
出願した特許については、単独、共同を問わず出願時には出願の旨を他実施者に通知し情報共有する仕組みとした。本知財計画により各実施者が本プロジェクトを通じて、知財の情報共有を図り、積極的戦略的に実施者間で知財活用を促進できるようにした。（例：九州大学のメニーコアバリア同期の特許を（株）トプスシステムズが利用してビデオマイニング製品に適用する計画となった。）

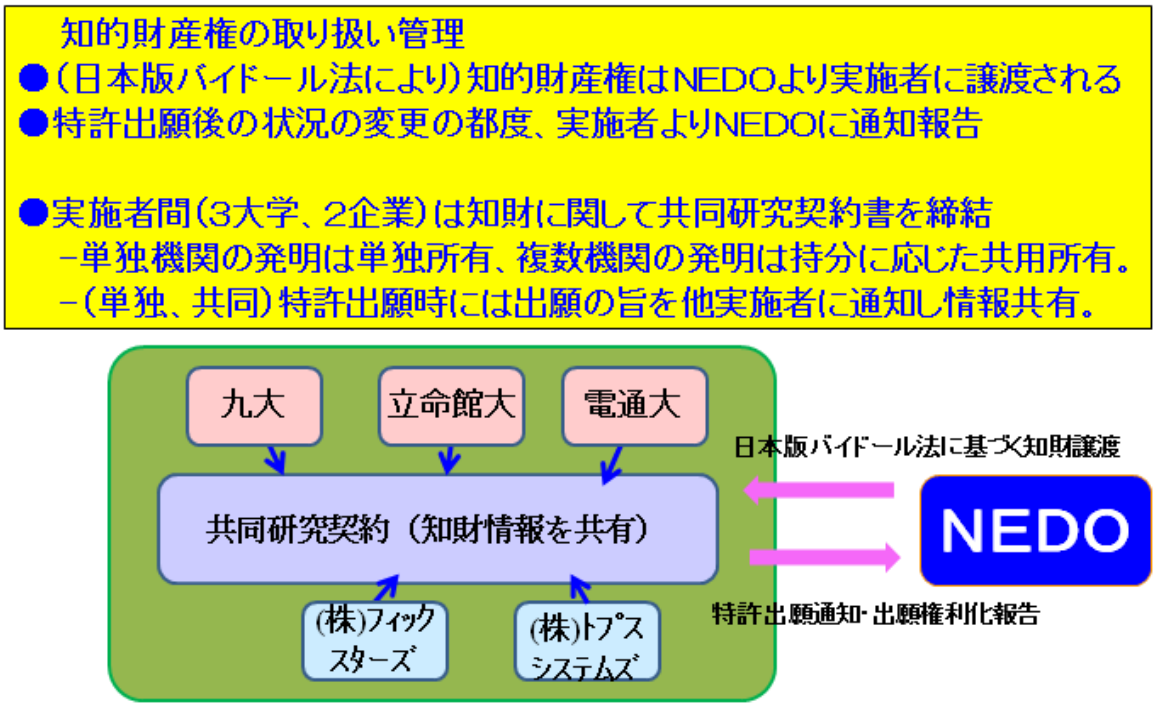


図2-6. 知的財産権の効率的管理方式

（３）研究成果の社外へのアピールによる企業連携・事業化促進

本事業の成果に基づき、プロジェクト外の国内関連企業についても成果の活用を促し、メニーコア実用化事業化を加速化するために、ＮＥＤＯ主催でメニーコアシンポジウムを2011年3月、2012年3月の２回、開催し外部に広く成果をアピールした。さらにＣＥＡＴＥＣ（2011～2012年の2回）のＮＥＤＯブース等にても展示してプロジェクト研究開発成果の成果を公開した。またメニーコアの研究開発成果について、展示やプレゼンテーションを行い、積極的に関連企業への情報発信アピールすることにより、関連企業との連携機会の創出、事業化の促進を図った。（ＣＥＡＴＥＣ2013の事例：ブース来場者数：約15,000名。うち、約60名とプロジェクト成果に関わる具体的な意見交換を実施した。）

図2-7. メニーコア シンポジウム2012、CEATEC2012

2-5　情勢変化への対応等

（1）タイムリなプロジェクト管理

タイムリなプロジェクト管理を行うために以下の4つの対策を実施した。

① PJ進捗会議によるフォローアップ

PL主催による実施者リーダを招集した進捗会議を定期的に開催し、適宜進捗状況をチェックした。また工程遅延や技術課題の明確化を行い、適宜、研究要員支援などの対策を実施した。開催に当たってはTV会議システムを最大限に活用してできる限り全員参加で開催した。ＮＥＤＯは進捗把握、課題の対策施策のため本会議に出席して適宜対策施策支援を行った。（1回／2ヶ月程度。但し緊急課題がある場合には臨時に開催。21回/2年開催）

② 事業化に向けたヒアリング

本事業の成果が出始めた初年度末から2回（2011年5月、2012年5月）、成果を生かした事業化ヒアリングをＮＥＤＯ主催で実施した。本ヒアリングレビューにより、本メニーコアを生かした分野として、ビデオマイニングに絞って、本技術利用候補の企業（自動車企業、半導体企業）とのコンタクトの開始・加速を促した。

③ 市場調査・技術動向調査

本プロジェクト期間内でも、再委託先を活用して、本メニーコア技術の用途、適用分野、市場につき調査を行った。（社）JEITAではメニーコア利用分野と市場領域の調査、eSOL（株）ではメニーコア向けOS調査、CATS（株）ではメニーコア開発環境調査の分担を振り分け、市場性と技術動向の両面で調査を行った。

プロジェクト終了後の有望な用途、市場ニーズ調査の結果、メニーコアの付加価値を生かした汎用ＳｏＣ用途、特定用途としてビデオマイニング用途を有望市場であることがわかった。ビデオマイニングを適用した市場としては、車載用途、医療用途、インターネット検索用途等が想定される。車載分野では、例えば車載カメラから得られた映像から障害物をオンラインで検知して衝突回避するシステムへの適用、医療分野では、内視鏡撮影映像から病巣をオンラインで検出するシステムへの適用、情報分野では、検索エンジンに組み込んでビデオデータからユーザ指定の映像を高速検索するシステムへの適用など多くの市場が期待できる。またメニーコア開発環境についてもメニーコア普及には不十分であることがわかり開発環境の充実化に向けた開発も行う計画とした。

④ プロジェクト終了後の実用化支援

プロジェクト終了後、研究成果を生かして各企業が実用化事業化の技術開発を継続して速やかに実用化事業化することが重要である。事業化の課題としてメニーコア事業化への連携企業の参画を誘導・促進するためにメニーコアモデル国際標準化とそれに基づく技術開発を行うことを目的として、プロジェクト後、ＮＥＤＯ（所管：省エネルギー部）による実用化加速に向けた『多様なマルチ・メニーコアの高度な活用を可能にする標準プラットフォーム開発とエコシステム構築による省エネルギー技術の実用化』ＮＥＤＯ助成事業プログラム（企業が１／２負担。2013年度－2014年度の2年間）プロジェクトに本プロジェクト実施者の（株）トプスシステムズと、車載メニーコアシステムを指向するルネサス（株）が参画して技術開発を推進できるようにした。これによりメニーコア向けソフトウェア開発環境・ツール提供企業が事業参画しやすい環境づくりと、本プロジェクトの成果を元にした国際標準化（標準モデル）推進、事業化実用化のプラットフォーム拡充を図り、省エネ促進メニーコアの情報通信機器の早期普及が可能となる。

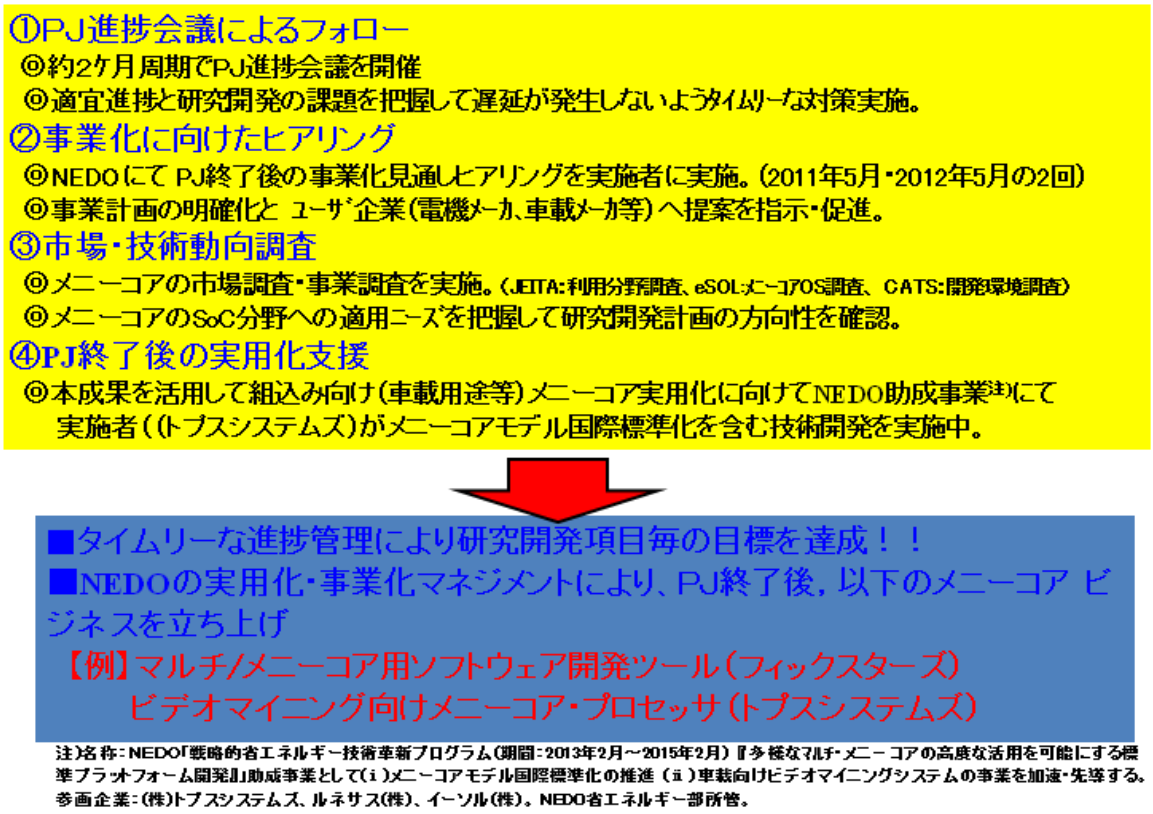


図2-8.　情勢変化への対応

Ⅲ　研究開発成果について

1　研究開発成果のまとめ

1.1　目標の達成と成果の意義

基本計画にて定められた目標と成果の達成度を表 3-1 に示す。また、本事業では成果のオープン化や情報発信にも積極的に取り組んでおり、その内容を表 3-2 ならびに表 3-3 にまとめる。なお、全ての成果と分担は表 3-4 の通りである。

表 3-1　基本計画に対する研究開発成果

目　　標	研究開発成果	達成度
あるべき低消費電力メニーコア用アーキテクチャを提案する。	● SMYLEref:本事業終了後 5～8 年をターゲットにした汎用 SoC 向けメニーコア。並列化された複数アプリケーションの同時実行。HW バリアやコア割り当て等の新技術を開発。FPGA プロトタイプにより 128 コア動作を確認。OpenCL を採用し、各種 API の策定と実装、ランタイムライブラリの開発、等により実行環境も合わせて構築。 ● SMYLEvideo：本事業終了後 1～2 年をターゲットにしたビデオマイニング向けアーキテクチャ。超高効率演算/通信機構を搭載し、複数タスクを同時実行。低レイテンシ・コア間通信方式等の新技術を開発。FPGA ボードで 2 クラスタ（容量の制約より縮小版）の動作を確認。	達成◎
既存技術（平成 20 年度末の最新技術）と比べて電力当たりの処理性能 2 倍を達成する。（ただし、チップ試作は含まれないためシミュレーションなどで確認）	● SMYLEref：既存方式と比較して電力当たり 4 倍の性能向上を FPGA プロトタイプを用いて確認（同世代ハードウェアでの比較）。これに加え、既存サーバ上で動作する動的コア割り当て技術を開発し、64 コア（16 コア×4 ソケット）の実機システムで最大 4.3 倍の性能向上を達成。 ● SMYLEvideo：平成 25 年の専用 HW アクセラレータを具備した 4 コア程度の画像認識プロセッサに対して約 3.2 倍の電力当たり性能の向上を達成。	達成◎
既存技術（平成 20 年度末の最新技術）と比べて、組込み向けアプリケーションプログラム実行時の電力消費量を 1/10 以下にする。（ただし、チップ試作は含まれないためシミュレーションなどで確認）	● SMYLEref：FPGA プロトタイプと見積りにより同一性能を前提とした場合には消費電力量 1/10 が達成可能であることを確認。実機サーバでの評価では、ファン等の周辺機器も含めた電力消費量 1/4 を達成。 ● SMYLEvideo：平成 25 年の画像認識プロセッサに対して消費電力 1/2（電力消費量 1/4）、回路規模 1/2 相当を確認。	達成○

表 3-2　研究開発成果の公開

目　　標	研究開発成果	達成度
ソフトウェア開発支援ツールを開発し公開する	● 並列化支援するツール CLtrump を開発。逐次 C プログラムを入力し、並列化した OpenCL プログラムを出力。ユーザは本ツールを用いてインタラクティブな並列化作業を効率良く実施。ソースコードレベルで公開中 http://ctrump.sourceforge.net/cltrump.html ● メニーコアへのソフトウェア移植の際に使用できる性能見積り支援ツール PEMAP を開発。ターゲット・プリケーションを入力とし、性能推定用ダミー・ソースコードを自動生成。2013 年 12 月を目処に希望するお客様に対してウェブベースの実行サービスを公開予定。 ● 高速ソフトウェア・ライブラリ BEMAP を開発。将来の主要 4 分野（ライフイノベーション、インターネット・アプリケーション、ファイナンス、ビジュアルコンピューティング）から 8 種類のコードを選択し、C のリファレンス・ソースコードならびに各種最適化を施した OpenCL ソースコード、性能評価レポートを無償公開。ベンチマーク・プログラムとしても利用可能。 http://sourceforge.jp/projects/sfnet_bemap/	達成◎
メニーコア評価環境を構築し公開する	● 複数の市販 FPGA ボードを接続したメニーコア FPGA プロトタイプ環境を開発（本プロジェクト終了時には 16 枚の FPGA ボードを用いた 128 コアの実行をサポート(画像処理アプリケーションのデモシステムも開発)。全ての HDL コードならびに関連するソフトウェア（並列ライブラリなど）を公開可能。 HDL レベルで公開可能な 128 コア規模の評価環境の開発は世界初である。今後、マニュアルの整備、ウェブでの公開、などが必要。 ● OpenCL プログラムを実行可能な機能シミュレータを開発。一般的な LinuxPC 上で動作可能であり、ソースコードレベルで公開可能。今後、マニュアルの整備、ウェブでの公開、などが必要。	達成○

表 3-3 特許ならびに情報発信に関する成果

出願特許数（うち海外出願）	4 件（1 件）
研究発表・講演数（うち海外発表）	53 件（22 件）
論文（うち海外論文）	2 件（0 件）

表 3-4　研究開発内容、分担、成果の一覧

汎用メニーコアSoC SMYLErefの開発		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	
九州大学 立命館大学 電気通信大学 東京農工大学	**・クラスタ型ホモジニアス・アーキテクチャ開発** - 基本アーキテクチャ開発 - ハードウェアバリア機構の開発（特許申請） - メニーコア向けDVFS方式の開発 - 動的コア割当て方式の開発 - 低電圧キャッシュ制御方式の開発 - 動的キャッシュ置換制御方式の開発 **・組込みメニーコア向けOpenCL実行環境の構築** - 静的コア割り当て技術の開発 - 応答時間短縮のための実行フレームワーク開発 **・評価検証環境の構築と評価** - 128コア搭載FPGAプロトタイプ・システムの開発(各種APIやLinuxベースOS含む) - 組込みOpenCL用高速ソフトウェア・エミュレータの開発 - 従来の全コア使用による複数プログラム順次実行方式と比較して約4倍の性能向上を確認（動作周波数を1/4にし、低電圧化を進めることで、消費エネルギー1/10の実現可能性を確認） **・実用化への取り組み** - トプスしてテムズへの技術移転（実用化を検討）	トプス システムズ	**・動画像認識を対象としたアプリケーション分析** - SIFTのソフトウェア処理におけるボトルネック分析、並列性解析、分散並列処理化 **・ビデオマイニング向けヘアーキテクチャの開発** - 2クラスタ（10コア）で15fpsでSIFT処理を実現 - スケーラブルなクラスタ型ヘテロジニアス・メニーコア・アーキテクチャの開発 - タスク並列処理＋データ並列処理方式の開発 - ゼロオーバヘッド・コアFIFO通信同期方式の開発（国内及び米国特許申請） - 画像認識系処理向け拡張複合命令の開発 **・消費電力評価環境の開発** - コアのパイプライン粒度の電力モデルの開発 - アプリ動作時の消費電力評価環境の開発 **・ソフトウェア開発環境の開発** -命令レベル・シミュレータの開発 **・動画像認識用ソフトウェア開発** - 分散並列処理型のSIFTソフトウェアの開発 **・実用化への取り組み** - 大規模FPGA(Xilinx V7 FPGA)による評価検証システムの開発 - ハードウェア・エミュレータによる検証環境の構築 - 大手ユーザとの商談開始
メニーコア向けソフトウェア開発環境の構築		**市場ならびに要求仕様の調査**	
フィックスターズ	**・ユーザ/コンパイラ・インタラクティブコード生成ツールCLtrumpの開発** - ソースコードやプロファイル情報に基づく並列化技術の開発 - C-to-OpenCLソースコード生成技術の開発 - オープンソースとして公開中 **・性能推定ツールPEMAPの開発** - 異機種を対象とする性能推定方式の開発 - 実行可能ダミーコード生成技術の開発 **・ベンチマークプログラムセットBEMAPの開発** - 主要4分野から8種類のアプリケーションを選定 - C言語のリファレンスコードとOpenCLによる最適化済みコードの開発（最適化に関するドキュメントも含めてオープンソースとして公開中） **・実用化への取り組み** - 社内にてPEMAPを用いた性能推定を実施 - BEMAPをサンプルとした有償での組込みサービスのビジネスを展開 - フィックスターズブランドM3に組込みライセンス販売を展開。多数の引き合いあり。契約処理中の案件あり	トプス システムズ フィック スターズ	**・ユーザー企業へのヒアリング調査** － 関連企業にヒアリングを実施 － メニーコアのニーズが高い具体的な処理の調査 － 10年後も使用可能な画像認識系コンピューティング・プラットフォームの要件を定義 **・学会や展示会等におけるブース展示ならびにアンケート調査の実施**
		JEITA キャッツ eSOL	**・メニーコア市場調査** - 調査会社による一次調査結果の分析 - セミナー開催とその際のアンケート調査、学会調査 - 日本の強み分野である制御系や制御系と情報系の融合分野への注力、普及促進策の提言 **・ソフトウェア開発環境ならびに企業における課題の調査** - 関連企業のヒアリング等を実施 - 並列化のみならず検証環境の重要性が明らかになった **・メニーコア向けオペレーティングシステムに関する調査** - 学会や文献調査、関連企業のヒアリング等を実施 - リアルタイム性を考慮したメニーコア向けオペレーティングシステムの開発の重要性、などの知見を得た

1.2　知的財産権等の取得

知的財産権の取得に関しては、以下のように国内外で実施した。

1. 中村孝史、河合夏輝、田村陽介、“実行時間算出装置、実行時間算出方法、及びプログラム” 出願番号 2012-251560、 出願日 2012 年 11 月 15 日。本特許はフィックスター

ズ社の事業戦略に位置付けられる「M3 プラットフォーム」にて本成果を活用するために必要となる。

2. 曽我武史、 佐々木広、 井上弘士、“同期処理回路及び同期処理方法”出願番号 特願 2012-207071、出願日 2012 年 9 月 20 日。本特許は汎用性のある内容であり、多コア化が進むにつれ重要性を増す技術である。トプスシステムズにおけるメニーコアの実用化において利用を検討しており、技術移転は完了している。
3. 松本祐教、内田裕之、“プロセッサコア、およびマルチコア・プロセッサ・システム”出願番号 特願 2012-15988、出願日 2012 年 1 月 27 日。本特許はトプスシステムズ社が実用化を目指す SMYLEvideo で用いられる要素技術である。
4. Yukoh Matsumoto and Hiroyuki Uchida、“PROCESSOR CORE AND MULTI-CORE PROCESSOR SYSTEM”出願番号 13718796、出願日 2012 年 12 月 28 日(米国)。上記 3 と同様であり、海外戦略として米国特許の取得は重要である。

1.3 成果の普及と情報発信

学会活動に関しては、論文誌 2 件、国際会議 8 件（最終成果報告書提出後に 1 件追加)、国際会議招待論文 5 件、国際会議招待講演 7 件（最終成果報告書提出後に 1 件追加)、国際会議査読付きポスター発表 2 件、国内会議 1 件、国内会議ポスター発表 1 件、国内研究会発表 15 件、国内招待講演（パネル討論も含む）14 件、にて研究成果を報告した。特に、並列処理関係では世界トップクラスの会議である IEEE/ACM PACT（International Conference on Parallel Architecture and Compilation Technique）での 2 年連続の発表（論文採択率は 20%未満)、EDA ならびに組込み分野で世界トップクラスの会議である DATE（Design, Automation and Test in Europe）での招待講演発表、ASPDAC（Asia and South Pacific Design Automation Conference）でのスペシャルセッション開催、著名人による招待講演のみで構成される MPSoC（International Forum on Embedded MPSoC and Multicore）での講演など、積極的に海外への情報発信を行った。なお、研究成果が認められ、Workshop on Advances in Networking and Computing での Best Paper Award、先進的計算基盤システムシンポジウムでの優秀ポスター賞、情報処理学会山下記念賞、を受賞した。

産業界に対する情報発信としては、メニーコア研究・開発・応用の普及を目的とし、2012 年 3 月と 2013 年 1 月の 2 回にわたり NEDO 主催メニーコア・シンポジウムを開催した。第 1 回は 165 名の参加者（内、企業参加者 122 名）であり、本プロジェクトを 1 年経過した段階での中間報告に対して多くのフィードバックを頂いた。第 2 回は 150 名（内、企業参加者 115 名）であり、本プロジェクトの最終成果を報告すると共に今後のメニーコア応用に関する議論を行った。特にメニーコアの活用を検討しているユーザ視点からの意見が多くあり、情報交換の場としても意義あるシンポジウムであった。

2　研究開発成果の詳細：汎用メニーコア SoC SMYLEref の開発

2.1　SMYLEref の狙い

現在、我が国の SoC（System on a Chip）ビジネスは成功しているとは言いがたい状況である。これは、SoC 開発コストの増大、ならびに、市場シェアの縮小が大きな原因と考える。一般に、SoC では応用に特化して他社との差別化を計るため、多品種少量生産とならざるを得ない。ある特定分野において圧倒的なシェアを有する場合（例えば通信分野における Qualcomm 社の SoC）には良いが、出荷数が小さい場合にはチップ当たりの開発コストが増大し競争力を失ってしまう。したがって、様々な応用に利用できる「汎用 SoC プラットフォーム」の実現が急務の課題となる。

この問題を解決する 1 つのアプローチとして FPGA（Field Programmable Gate Array）の採用が挙げられる。書換え可能なハードウェアを共通部品として使用することで出荷数を増やすことが可能となる。しかしながら、本質的に HW（ハードウェア）設計が必要であり、より大規模化／複雑化する昨今の SoC の置き換えを考えた場合には依然として開発コスト増大の深刻化は避けられない。そこで本事業では、2020 年頃の実用化を目指し、汎用 SoC を実現すべく、以下を目標とする研究開発を実施した。

- 完全 SW（ソフトウェア）処理：現在の SoC にはアプリケーションに応じた専用 HW マクロが多数搭載されている。これらを全て SW 処理に置き換えることで汎用性 SoC の実現を目指す。
- 高スループット&低レイテンシ実行：リアルタイム処理が要求される組込みシステムでは、応答時間の削減が極めて重要になる。そこで、高スループット化と低レイテンシ化の両立を目指す。

これらの目標を達成するために、ハードウェア・プラットフォームとしてはメニーコアを活用する。メニーコアの本質は、「低性能ではあるが小面積かつ低消費電力なコアを大多数搭載し、極めて高い並列処理性能を得る」といった点にある。そして、性能を効率良く電力消費量削減に変換（具体的には動作周波数と電源電圧の低下）し極めて高い電力効率を実現する。すなわち、「空間的または時間的に無駄な処理（急がなくて良い処理）を検出し、これに関するコアの稼働率を低くする」という従来の低消費電力化アプローチではなく、メニーコアにおいては「大多数の小規模コアをフル活用し並列実行効率を最大化する」ことがポイントとなる。また、組込みシステムは様々な製品に搭載されており、今後も多様化の一途を辿ると考えられる。例えば、動画像圧縮/伸張といったストリーム処理、ロボットや自動車などのマルチタスク処理、インターネット・アプリケーションに代表されるインタラクティブ処理など、その処理形態は応用によって様々である。したがって、この

ような多種多様な状況においても常に高い並列化効率を達成できるメニーコア実行フレームワークを構築しなければならない。そこで、本事業では、以下 2 つの設計思想に基づき、汎用 SoC を実現するメニーコア・アーキテクチャ SMYLEref と各種ソフトウェア実行／開発環境の構築を行った。

- アクセラレーション・プラットフォームとしてのメニーコア：多種多様な組込みアプリケーションを効率良く実行するためには、IO アクセスや各種制御などの複雑な処理を必要とする実行部分と、ある程度の並列化が期待できるカーネル部分を分離し、それぞれの特性に適したハードウェアで処理すべきである。そこで、メニーコアは、カーネル部分を加速実行するためのアクセラレーション・プラットフォームとする。
- マルチスレッド・マルチプログラム実行：コア数に対するスケーラビリティを維持するためには、データ／スレッドレベルの並列性だけでは不十分である。また、組込みシステムでは多数のタスクが同時に処理されなければならない場合が多い。そこで、メニーコアでは、並列化された複数アプリケーションの同時実行を前提とした設計最適化を行う。

2.2　SMYLEref アーキテクチャ

SMYLEref アーキテクチャの全体像を図 2.2-1 に示す。SMYLEref では「仮想アクセラレータ（VAM: Virtual Accelerator on Manycore)」の概念を導入している。コンパイラは、アプリケーション特性に応じて自らが仮想アクセラレータの構成（アーキテクチャ）を決定し、それに基づき実行コードを生成する。このように、コンパイラに対してハードウェア・アーキテクチャの決定権を与えることで、自動並列化戦略の選択肢を拡大することができる。また、小規模コアに対して複数の仮想アクセラレータを同時にマッピング可能とすることで「コア数にスケール可能な性能向上」を実現する。以下、SMYLEref アーキテクチャの 3 つの特徴を述べる。

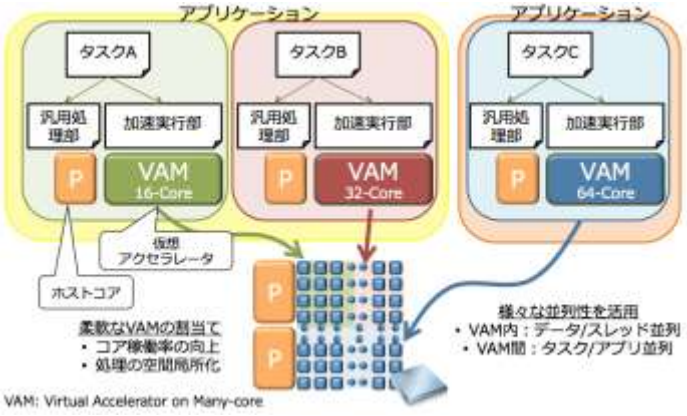
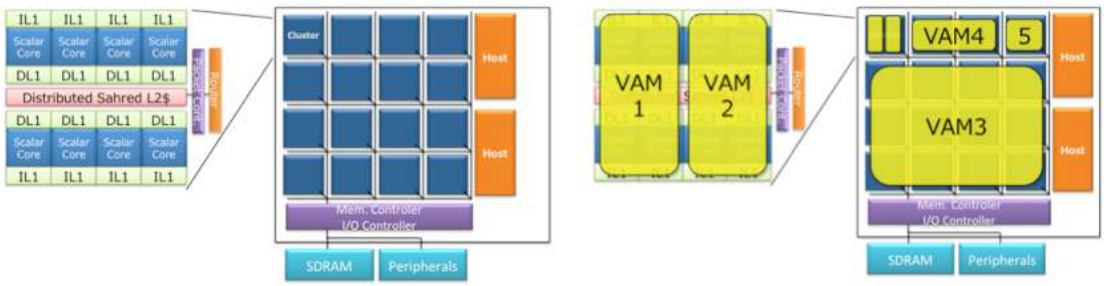


図 3-1 メニーコア上で動作する仮想アクセラレータ VAM の概念



(A) 内部構成　　(B) VAMマッピングの例

図 3-2 クラスタ構成と VAM マッピングの例

- クラスタ構造（付録：最終成果報告書第 2.1.2～2.1.3 節）：図 3-2（A）で示すように、SMYLEref はシンプルな数個のプロセッサコアをバスで結合したクラスタ構造となっており、クラスタ間は 2 次元メッシュのオンチップネットワークで結合される。プロセッサコアは MIPS ベースの命令セットアーキテクチャであり、8KB の L1 命令キャッシュ、および、L1 データキャッシュを有する（本事業最終年度において公開時のライセンス問題を回避するため OpenRISC コアに変更）。図 3-2（B）のように、各 VAM は 1 個以上のクラスタにマッピングされる。本開発では、多くの場合において 8 並列程度までは十分なスケーラビリティが存在することを想定し、単一クラスタ内のコア数を 8 とした。
- メモリアーキテクチャ（付録：最終成果報告書第 2.1.2 節）：各クラスタには分散共有 L2 キャッシュの一部（L2 キャッシュ・スライスと呼ぶ）が搭載される。各 L2 キャッシュ・スライスにはアドレスインターリーブ方式でメモリ空間が割り当てられている。したがって、物理的には分散した配置となっているが、論理的には共有した 1 個の L2 キャッシュとなる。L2 キャッシュは共有方式を採るため、コヒーレンス制御は不要となる。また、L1 キャッシュにおいてもハードウェアでのコヒーレンス制御は保証しない。これは、SMYLEref では OpenCL のプログラミングモデルを採用しており、基本的に OpenCL ではキャッシュの一貫性はプログラマが保証する前提となっているためである。
- VAM 向けハードウェア／ソフトウェア・サポート（付録：最終成果報告書第 2.1.4～2.1.5 節）：各クラスタに配置された L2 キャッシュ・スライスでは、各 VAM に割り当てられたクラスタに属する L2 キャッシュ・スライス群で LLC を構成する。VAM 間での L2 キャッシュ競合を回避するための特別なアドレス変換機構により、分散共有キャッシュ・アクセスにおける空間局所性を高める。また、グループ・ハードウェア・バリアは、それぞれの VAM に割り当てられたコアのみを対象とする高速同期機構であり、バタフライ方式のソフトウェア実装と比較して 60 倍以上の高速化を実現している。さらに、DCFS（Dynamic Core and Frequency Scaling）は複数コアで単一並列アプリケーションを実行する際に電源電圧と動作周波数を動的に最適化する。

2.3 プログラミング・フレームワーク SMYLE OpenCL

SMYLEref 用のプログラミングモデルとして、仕様がオープンでありかつロイヤルティフリーである OpenCL を採用した。これまでに、GPU や汎用マルチコア/マルチプロセッサを対象とした OpenCL 環境がいくつか開発されているが、組込みシステムへの適用を考えた場合にはいくつかの問題点がある。そこで、SMYLE OpenCL では、表 3-5 に示すような拡張を行った。

表 3-5 既存 OpenCL 環境と SMYLE OpenCL の比較

	既存 GPU 向け OpenCL	SMYLE OpenCL
タスク並列実行	多くの場合は逐次実行モデルとして実装	真の空間並列実行モデルとして実装 →タスクレベル並列性の積極活用
タスク実行単位	HW アーキテクチャ上のクラスタ単位（粗粒度）	コア単位（細粒度） →コア利用率の向上
アプリケーション実行	単一プログラム	複数プログラム →プログラムレベル並列性の積極活用
タスク割当て等の実行制御	全て実行時	可能な限りコンパイル時＋実行時 →実行オーバヘッドの削減

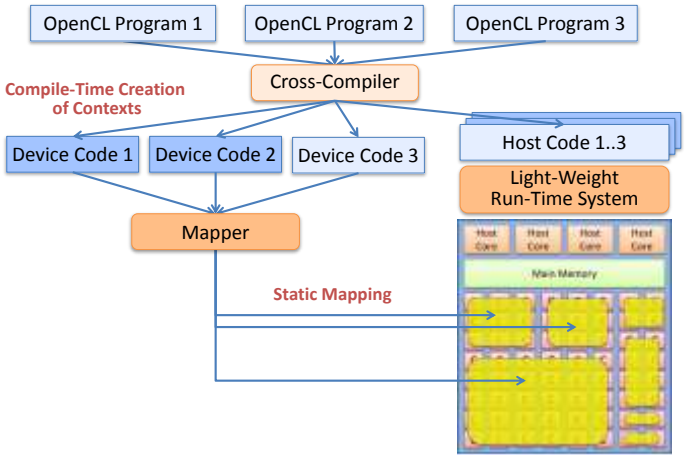


図 3-3　SMYLE OpenCL の全体像

図 3-3 に、SMYLE OpenCL 環境の全体像を示す。SoC 設計者は、クロスコンパイラを用いて実行すべき各プログラムをコンパイルする。これにより、メニーコア上に実装される VAM にて加速実行されるデバイスコード、ならびに、制御を司るホストコードが生成される。その後、全てのデバイスコードに対し、システムに搭載された総コア数を制約条件として VAM が構成され（実行に使用するコア数が決定され）、具体的なマッピングが決まる。なお、現在の SMYLEref 向け SMYLE OpenCL 環境においては、クロスコンパイラは GNU C コンパイラをそのまま使用している。そのため、SMYLE OpenCL 環境は、OpenCL 言語独自のデータ型

や構文はサポートしていない。以下、SMYLE OpenCL でのプログラムの典型的な実行の流れを示す。ここで、二重線で消去している 1〜4 ならびに 8 は、SMYLE OpenCL では SoC 設計時に行われる（既存の OpenCL では動的に実行される）処理の内容である。つまり、コンテキストやコマンドキューなどのオブジェクトは設計時に作成しておき、製品の出荷時にメモリに書き込んでおく。また、各タスク（カーネルやアプリケーション）を実行するコアを決定するタスクマッピング処理も設計時に行う。

~~1. デバイス情報を取得、デバイスを確保~~
~~2. コンテキストを生成~~
~~3. コマンドキューやメモリバッファを生成し、コンテキストに格納~~
~~4. カーネルをビルドし、コンテキストに格納~~
5. 入力データをメモリバッファに格納
6. カーネルを実行
7. 実行結果をメモリバッファから読み出し
~~8. コマンドキュー、メモリバッファ、コンテキストなどを解放~~

このような組込みシステム向け SMYLE OpenCL 環境を構築するため、主に以下を新規開発した。

- ランタイムライブラリ（付録：最終成果報告書第 2.2.2 節）ならびに低レベル API（付録：最終成果報告書第 2.1.7 節）：ホスト用ならびにデバイス用のランタイムライブラリ。OpenCL の仕様が定義している関数の中で、特に重要な物としてホスト用 28 種、デバイス用 36 種の仕様を策定し実装した。これに加え、SMYLEref アーキテクチャにおいてホストコアから VAM を制御するための API を設計し実装を行った。
- タスクマッピング（付録：最終成果報告書第 2.2.3 節）：各デバイス用コードについて、そのデバイス用コードを実行する際に使用するコアを決定する処理（VAM を構成する処理）である。具体的には、シングルコンテキスト静的タスクマッピング手法、ならびに、マルチコンテキスト静的タスクマッピング手法を開発した。シングルコンテキスト静的タスクマッピングでは、タスクが排他的にコアを使用する。タスクは複数のコアを使用することはできるが、コアは複数のタスクを実行することができない。実行時にコンテキストスイッチ（タスクの切替え）が発生しないため、リアルタイム性に優れている。一方、マルチコンテキスト静的タスクマッピングでは、複数タスクが同一コアを共有することを許す方式であり、コア利用率を高めることができる。これらに関して線形計画問題として定式化し、マッピングを決定する技術を確立した。
- 機能シミュレータ開発（付録：最終成果報告書第 2.2.4 節）：開発したランタイム・ライブラリを用いた OpenCL プログラム実行を可能とする機能シミュレータを開発した。

2.4 128 メニーコア FPGA プロトタイプ・システム

メニーコアのアーキテクチャやソフトウェアを検討・開発していくにあたり、その評価環境の構築は重要な課題である。しかしながら、世界的に見ても、OS までを含めた大規模メニーコアの評価技術は未だ確立されていない。そこで、SMYLEref の開発にあたっては、FPGA を利用して評価環境を構築した。なお、本環境は HDL レベルで公開可能であり、本事業に留まらず、メニーコア研究開発を遂行する企業や大学に公開することを前提としている(pthread 相当の並列ライブラリも実装)。本プロトタイピングでは、Xilinx 社製の FPGA チップである Virtex-6 を搭載する ML605 評価ボードを利用した。回路設計には VerilogHDL を用い、 論理合成、マッピング、配置配線には Xilinx 社の ISE を用いた。16 枚の FPGA ボードを用いた 128 コア SMYLEref のプロトタイプ外観を図 3-4 に示す。本プロトタイプの内容は以下の通りである（付録：最終成果報告書第 2.3 節)。

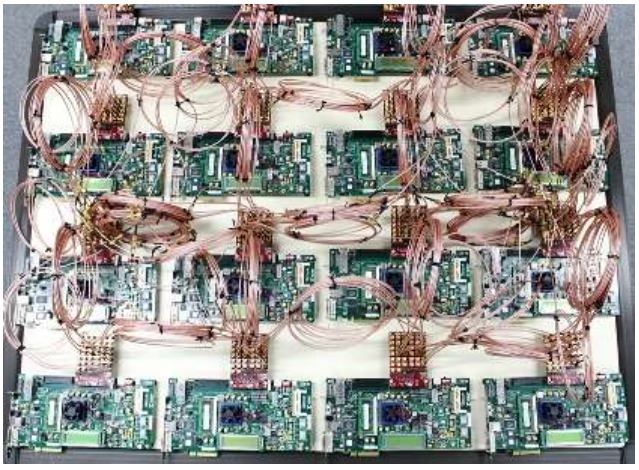


図 3-4　評価環境の外観

- 各クラスタは、MIPS ベースの RISC コアを 8 個、NoC のためのルータ、分散共有 L2 キャッシュ・スライスを搭載（図 3-2)。
- ペリフェラル・クラスタは、SDRAM や UART、SysACE 等のコントローラを備えており、それらは Xilinx 社が提供する PLB (Processor Local Bus) バスに結合される。ペリフェラル・クラスタ上のルータも PLB と接続されており、これによりオンチップネットワークを介して、各コアとペリフェラルが接続されることになる。
- ボード間の通信には高速シリアル通信インタフェースである rocket I/O を使用。ルータモジュール内にボードを跨がって通信を行う際に使用する通信モジュールを開発した。各 FPGA ボードには 1 クラスタが搭載されており、これらが 16 ボード接続して 128 コアとなる。
- プロセッサコアのクロック周波数として数 10MHz 程度、rocket I/O によるルータ間は 5GHz でのシリアルデータの通信。

2.5　SMYLEref の評価

128 コアを搭載したメニーコア FPGA プロトタイプに SMYLE OpenCL 環境を実装して評価を行った。ベンチマークとしては、本事業で開発したメニーコア向けベンチマーク BEMAP（付録：最終成果報告書第 4.3 節）に含まれる 6 種類の OpenCL プログラムを用いた。

SMYLEref 最大の特長は、並列化された複数アプリケーションを同時実行する点にある。そこで、FPGA プロトタイプでの評価に基づき、複数プログラム同時実行時の性能を見積った（付録：最終成果報告書第 2.4.2 節)。評価対象モデルは以下の通りである。

(A) 逐次実行：各プログラムは、最も実行時間が短くなるコア数に対してのみタスクを割り当て、データ並列処理により実行する。あるプログラムの実行が終了すると、次のプログラムが最大 128 コアを使用してデータ並列実行を行う。

(B) アプリケーション並列実行：6 つのプログラムを空間的に並列に実行する。SMYLE OpenCL で開発したシングルコンテキスト静的タスクマッピング手法（整数線形計画法）に基づき、128 コアを 6 つのプログラムに割り当てる。

評価の結果、従来実行方式と比較して、SMYLEref は、全てのコアが使用／未使用関係なく稼働時と同じ電力を消費すると仮定した場合に 4.03 倍の電力当たり性能向上を実現できることを確認した。したがって、性能一定の条件下では、動作周波数を 1/4 とし、それに伴い電源電圧を 40%程度削減することが可能となる（AMD 社のプロセッサ Opteron6136 では動作周波数を 1/3 にすることで約 30%の電源電圧低下が可能であり、動作周波数 1/4 を許容し電源電圧を 40%低下させることは十分可能と考える)。これにより、動的消費エネルギー1/10（CMOS 回路の動的消費電力は動作周波数と電源電圧の 2 乗に比例するため、0.25×0.6×0.6＝0.9 倍）を達成できることが分かった。また、未使用コアは電力を全く消費しないアイドル状態をサポートする場合には、電力当たりの性能向上比 2.5 倍、消費エネルギー1/7 であった。

SMYLE OpenCL のもう 1 つの特長は、実行の前処理／後処理を設計時に静的に行うため性能オーバーヘッドが小さくなることである。そこで、以下の 3 つの OpenCL 処理系の比較を行った（付録：最終成果報告書第 2.4.3 節)。

- FOXC/Intel：Intel Core i7（2.8GHz、4 論理コア）上で、フィックスターズ社が開発した OpenCL 環境 FOXC を使用した場合
- SMYLE OpenCL/FPGA:FPGA 上に実装された SMYLEref アーキテクチャ(10MHz、4 コア）上で、SMYLE OpenCL 環境を使用した場合
- SMYLE OpenCL/Intel：Intel Core i7（2.8GHz、4 論理コア）上で、SMYLE OpenCL の機能シミュレータを使用した場合

評価の結果、FOXC/Intel と比較して、SMYLE OpenCL/FPGA のコアの性能は 1000 分の 1 程度であるにも関わらず実行時間は 45%程度短縮した。また、SMYLE OpenCL/Intel の実行時間は FOXC/Intel の 20 分の 1 以下となり、SMYLE OpenCL の有効性が示された。

2.6　メニーコア向け動的コア数／周波数／電源電圧制御技術

本事業では組込みシステムへのメニーコア応用を前提としていたが、メニーコアの普及を考えた場合にはサーバシステム応用も極めて大きな市場を有する。そこで、SMYLEref の基本思想である「並列化された複数プログラムの同時実行」の効果をより明らかにするため、図 3-5（C）に示すタスクスケジューラを開発し Linux に実装した（付録：最終成果報告書第 2.1.6 節)。本スケジューラは、プログラム実行中にハードウェア・カウンタの値に基づき各プログラムのスケーラビリティを予測する。そして、各プログラムに割り当てるべきコア数ならびに動作周波数と電源電圧を動的に最適化する。比較対象は、全てのコアを使用して各並列プログラムを順番に実行する従来方式とした。64 コア（16 コア×4 ソケット）を搭載した実機サーバシステムを用いた評価を行った結果を図 3-6 に示す（横軸は 4 つのベンチマークの組であり PARSEC ベンチマークセットより選択した)。この結果より、SMYLEref 実行方式では最大で 4.3 倍の性能を達成し、その時の消費電力量は 1/4 以下であることが分かる。本評価では電力メータを用いてサーバシステム全体の消費電力（静的消費電力も含む）を実測している。このように、プロセッサチップのみならず、システムレベルで消費電力量を大幅に削減しており、提案方式の有効性が明らかになった。

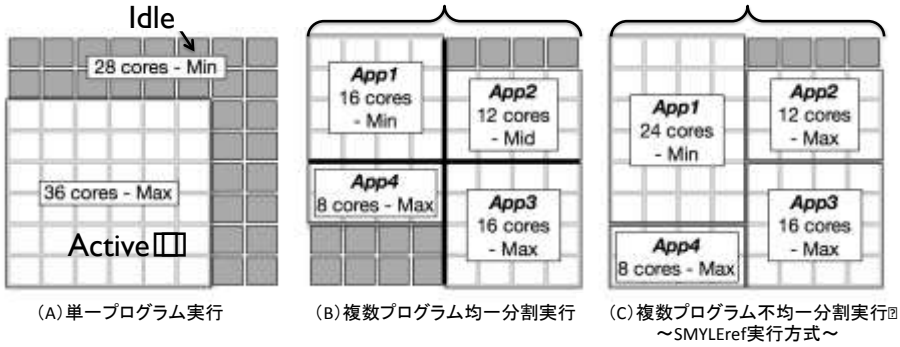


図 3-5：サーバシステムにおける動的コア割り当て方式（概念図）

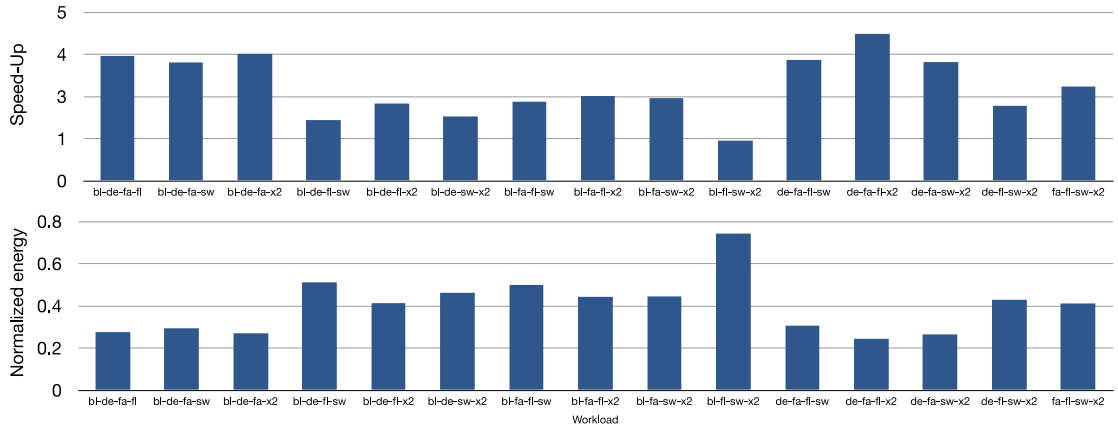


図 3-6：従来実行方式と比較した際の性能向上比（上）と消費エネルギー（下）

3　研究開発成果の詳細：ビジュアル・コンピューティング・メニーコア SMYLEvideo の開発

3.1　SMYLEvideo の狙い

ビジュアル・コンピューティングは今後更なる市場拡大が期待できる極めて有望な分野の１つである。その中でも、ビデオ・マイニング（膨大な動画像データの中から特定の動画像の認識や追跡、検索、ダイジェストの抽出など）は多くのアプリケーションで必要とされる処理である。これまでにも、画像認識等を目的とした専用回路（つまり ASIC）が開発された。しかしながら、特に画像認識に関しては年々新しいアルゴリズムが出現しており、設計コストならびに開発期間の観点から専用回路での実装は限界に近づきつつある。そこで本事業では、2015 年頃の実用化を目指し、以下を目標として研究開発を行った。

- ビデオ・マイニングに特化したメニーコア・アーキテクチャの開発：現在の画像認識用 SoC にはアプリケーションに応じた専用 HW マクロが多数搭載されている。これらを全て SW 処理に置き換えることで、新しいアルゴリズムの出現にも柔軟に対応できる動画像認識向けメニーコアを開発する。特に、豊富なハードウェア資源を効率良く活用することで、最新の動画像認識 LSI を凌駕する（もしくは遜色ない）高い性能を実現し、その上で低消費電力化と小面積化を目指す。
- 特徴量抽出アプリケーションの開発：多くの動画像認識アプリケーションで必要となる特徴量抽出プログラムを対象とし、メニーコア・アーキテクチャの潜在能力を最大限に活用するソフトウェアを開発する。
- デモンストレーション・システムの開発：本事業終了後、速やかに事業化フェーズへと移行するため、FPGA 評価ボードを用いたプロトタイプによるデモンストレーション・システムを開発する。

これらの目標を達成するためには、ビデオ・マイニングを対象としたハードウェアとソフトウェアの協調設計（コデザイン）が重要となる。そこで、本事業では、以下 2 つの設計思想に基づき、ビデオ・マイニング向けメニーコア・アーキテクチャ SMYLEvideo　と各種ソフトウェアを開発した。

- ソフトウェア実装を意識したハードウェア構成：並列処理モデルとして Kahn Process Networks (KPN)を採用し、ハードウェアでのストリーム処理を効率良く行うための各種ハードウェア・サポートを搭載する。並列処理モデルに適した HW プラットフォームを構築することで、無駄のない効率的な実行が可能となる。

- 様々なレベルの並列性の活用：タスクレベル並列性とデータレベル並列性をバランス良く活用することで、負荷の平均化を行う。これにより、豊富なハードウェア資源を活用した効率的な超並列実行が可能となる。
- アプリケーション特性に基づく徹底したハードウェア最適化：プロセッサ・コアの設計において、SIMD 幅の決定／選択や複合命令の導入など、徹底してビデオ・マイニング・アプリケーションに特化した最適化を実施する。これにより、極めて高い電力効率を実現できる。

3.2 SMYLEvideo アーキテクチャ

画像認識のアルゴリズムには潤沢な並列性を内含しており、マルチコア、メニーコアによって、効率的に処理速度の高速化が可能である。しかしながら、処理負荷自体は非常に高く、リアルタイムの画像認識のためには、500GOPS(Giga Operation Per Second)を超える程度の性能を実現する必要がある。このような性能を低消費電力で実現するには、10 個以上の演算コアを有機的に結合して、効率よく動作させるとともに、特定資源へのアクセスの集中が生じないよう注意深くアーキテクチャ設計を行う必要がある。そこで、SMYLEvideo ではトプスシステムズ社が有するヘテロジニアス・マルチコア TOPSTREAM™ アーキテクチャに、並列度を効率的に高めるべくコア数を増やせるクラスタ構造を融合して、所望の性能を実現するアプローチを採った。SMYLEvideo のブロックズを図 3-7 に示す。以下、主な特徴を示す（付録：最終成果報告書第 3.2.1 節)。

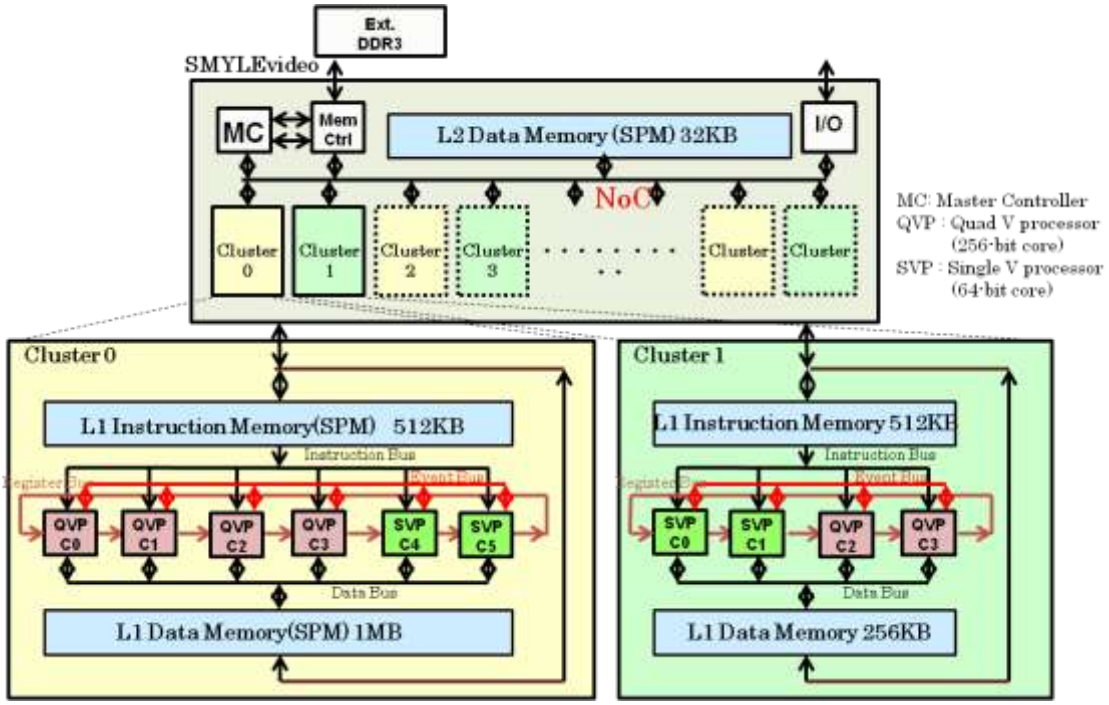


図 3-7　SMYLEvideo のブロック図

- 2 種類のクラスタを多数搭載したヘテロジニアス構成：クラスタ 0 は、主にフィル

タ処理と特徴点抽出処理（全画像を探索）を担当する。一方、クラスタ 1 では、クラスタ 0 で抽出した特徴点の解析を行う。これらのクラスタ構成は、データのローカリティやコア間交信の効率化を考慮して決定した。本クラスタペアを更に追加することにより、スケーラブルに画像認識能力（一秒当たりのフレーム処理枚数）を高めることが可能である。

- ビット幅の異なる 2 種類のコア：クラスタにはビット幅の異なる 2 種類のコアを搭載する。256 ビットデータレジスタバンクならびに SIMD 演算器を備える QVP(Quad Vision Processor)と、 64 ビットデータレジスタバンクならびに SIMD 演算器を備える SVP(Single Vision Processor)である。これらは、処理フェーズによって使い分けることができる。QVP/SVP では、演算処理のバックグランドでストリーム入出力を行うことができ、入出力のための処理時間を隠蔽可能としている。これらコアでは、例えばガウシアン・フィルタ処理といった極めて高いレベルの拡張複合命令をサポートしている。
- 最適化したコア間／クラスタ間通信：各コアに割り当てたプロセス間での通信を効率的に行うため、図 3-8 に示すレジスタファイル共有技術を導入した。また、図 3-9 で示すように命令レベルでコア間交信を明示的に示すことができ、プロセス間同期オーバヘッドを大幅に削減できる。一方、各クラスタは、L2 メモリや処理全体を司る MC (Master Controller)、周辺ペリフェラルインタフェースと同様にスケーラブルな NoC (Network on Chip) にて接続される。この NoC は必要な部分にデータコネクションを張った部分クロスバ構造であり、BUS 型及び直結型の SBUS インタフェースに対応している。

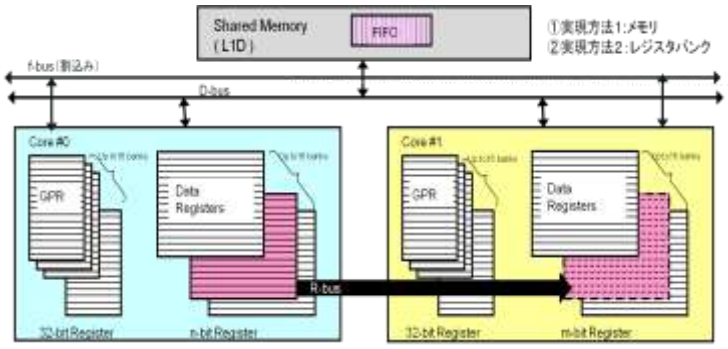


図 3-8　コア間のレジスタファイル共有（FIFO 処理効率化）

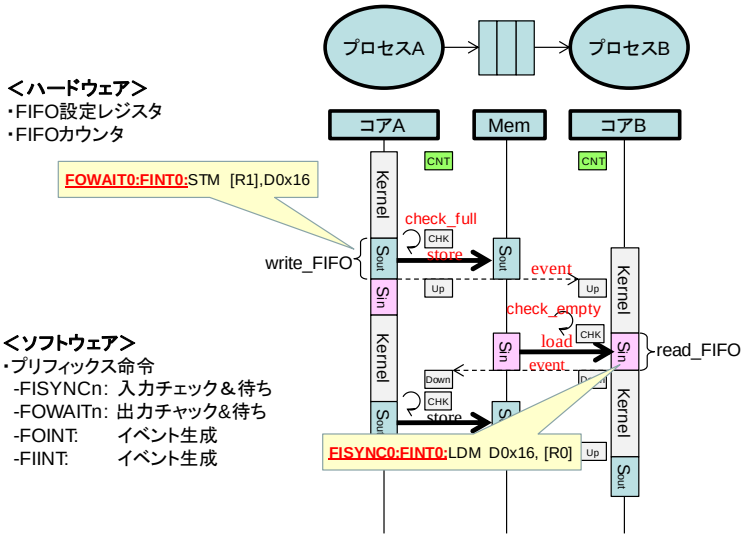


図 3-9　prefix 命令による同期オーバヘッド削減(ゼロオーバヘッド交信)

3.3 動画認識アルゴリズム SIFT の実装

今回メニーコア上への実装を行った SIFT とは Scale-invariant feature transform の略である画像認識アルゴリズムである。このアルゴリズムは画像の回転、サイズの変化、照明の変化などに対して安定した認識が行えるアルゴリズムとして知られており、例えば自動車から撮影した動画像ストリームをこのアルゴリズムで解析することで、交通標識などを自動認識させるような応用が期待されている。今回実装の参考としたソースコードは、オープンソースプロジェクトである OpenCV プロジェクトに含まれているものである。物理的な動画入力装置としては、1920x1080 ピクセルサイズのフルカラーHD ビデオカメラを用いて動画像を入力し、これをハードウェアで 3 対 1 のダウンザイジングし 640x360 ピクセルサイズ、モノクロ 256 階調グレースケール画像としたものをソフトウェアへの入力データとしている。本実装において設定した性能目標は以下の通りである。

- SMYLEvideo 標準構成で 15fps の実時間スループットを実現する
- 複数クラスタ動作させることで 30fps／60fps へのスケーラビリティを確保する

以下、本実装で実施した内容をまとめる（付録：最終成果報告書第 3.2.2 節)。

- 固定小数点化による負荷の軽減：オリジナルの SIFT アルゴリズムは、倍精度浮動小数点演算を多用し、主記憶メモリ上に展開した画像データを大量に読み書きする。例えば、VGA サイズの 1 画面の処理には、PC での一般的な命令セットにおいて 3G

から 9G サイクルもの演算処理が必要とされる。そこで、詳細な数値実験を行い、オリジナルアルゴリズムとほぼ等価な結果を保証する範囲内で浮動小数点数を固定小数点数へと変換した。

- 機能分割：メニーコアへの実装を意識した機能分割を実施した。まず、オリジナルの OpenCV プログラムに対してプロファイリングを行い詳細な解析を行った。そして、オリジナルのソースコードを解読し、細粒度処理（意味のある一つのデータを出力するために必要な入力とその処理の単位）にまで分解した後、次のような機能に再構成（グルーピング）した。①ガウシアンフィルタおよび差分画像の生成機能、②特徴点候補の発見機能、③特徴点候補の絞り込みによる特徴点確定機能、④特徴点まわりの勾配強度、画素方向の算出機能、⑤勾配強度、画素方向にもとづく特徴点主方向の決定機能、⑥特徴ベクトルの算出機能、⑦特徴ベクトルとテンプレート（辞書）ベクトルの照合機能。
- メニーコアへのマッピング：上記①～⑦の各機能の特性を考慮し、QVP/SVP コアへの割り当てを行った。
- 細粒度 KPN 方式の実装：メモリアクセスの削減と電力効率の改善を目的とし、KPN 方式でソフトウェアを実装した。上述した①～⑦の各機能を KPN の 1 プロセスとし、その間を図 3-10 のように接続した。

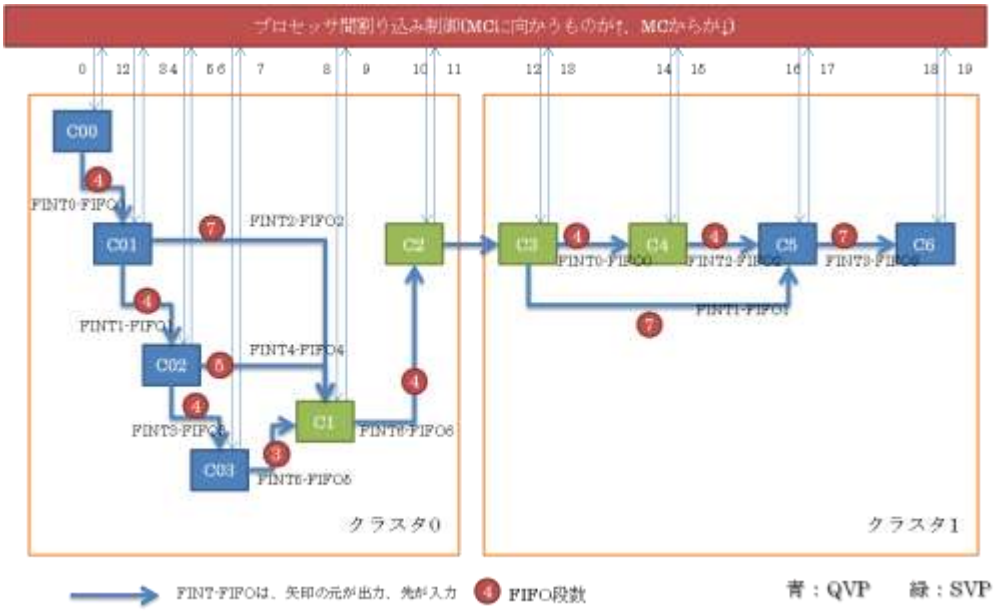


図 3-10　各機能をつなぐ FIFO 構成

3.4 評価環境／デモシステムの開発

SMYLEvideo の有効性を評価し、かつ、その実現可能性を実証するため、以下に示す評価／デモ環境の構築を行った。

- 命令レベル・シミュレータの開発（付録：最終成果報告書第 3.4 節）：メニーコア向けに開発されたソフトウェア（バイナリ）の実行を模擬するシミュレータを開発した。最大で 11 個のコアによる KPN 方式での実行をシミュレーションできる。また、開発したゼロオーバヘッドプロセッサ間通信機構などのハードウェア・サポートも反映している。シミュレーションの様子を図 3-11 に示す。なお、本シミュレータはプログラム開発に使用できるようデバッグ機能を有する。
- 消費電力モデルの開発（付録：最終成果報告書第 3.3 節）：過去に試作した実チップ（TSMC180nm プロセス）を用いて、プロセッサ構成要素を活性化した際の消費電力を測定した。そして、各種ハードウェアユニット稼働率（命令レベル・シミュレータにより取得可能）に基づく消費電力モデルを構築した。
- FPGA 評価／デモボードの開発（付録：最終成果報告書第 3.2.1 節）：SMYLEvideo のアーキテクチャを実証／評価するために VHDL を用いた RTL 開発を行った。また、図 3-12 に示す東京エレクトロンデバイス社の ASIC 開発評価プラットフォーム TB-7V-2000T-LSI　FPGA ボードを用いて動作確認を行った。

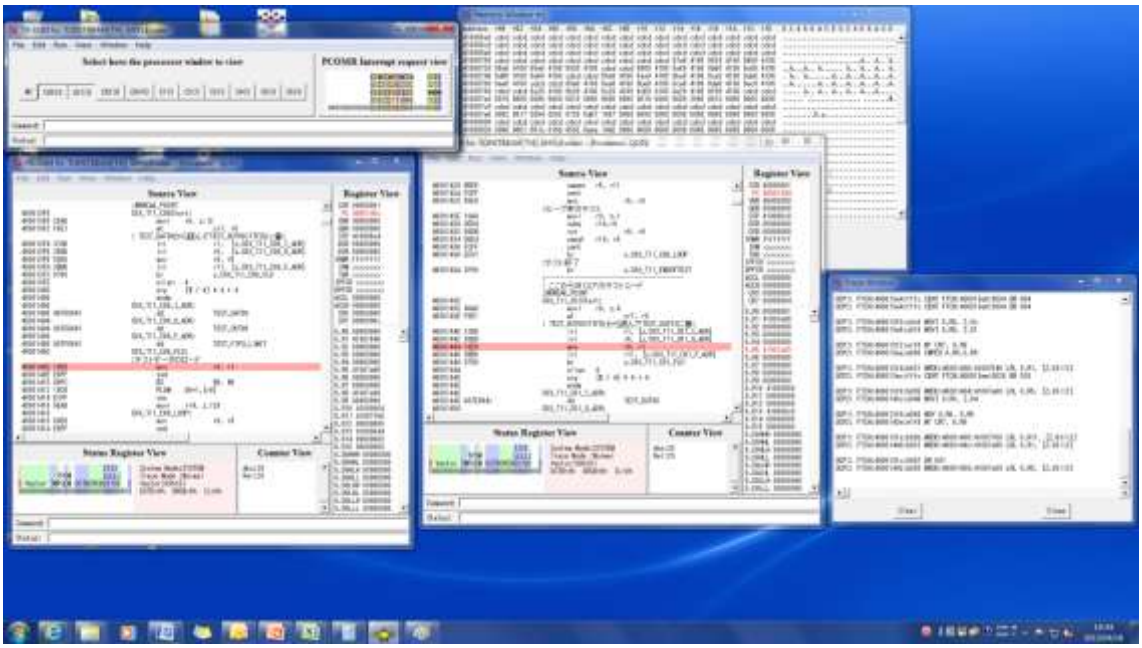


図 3-11　インストラクションシミュレータ（インタラクティブモード）

図 3-12　SMYLEvideo の実装検証ボード
（東京エレクトロンデバイス ASIC 開発評価プラットフォーム TB-7V-2000T-LSI）

3.5 総合評価

SMYLEvideo の有効性を明らかにするため、SIFT 実行に関する処理性能、消費電力、ハードウェア規模、の 3 つの視点から評価を行った（付録：最終成果報告書第 2.5 節）。

まず、処理性能に関して報告する。SMYLEvideo の最小構成は 2 クラスタ（合計 10 個の演算コア（QVP/SVP）が集積される。表 3-6 にそれぞれの演算性能を示す（MC は制御用の小規模コア）。

表 3-6　各 DPE の演算処理性能(100MHz 動作)

	GOPS 値 (16b Intel 換算、ピーク)	個数 2 クラスタ	合計 GOPS
QVP	96	6	576
SVP	24	4	96
MC	0.1	1	0.1
合計			672.1

QVP ならびに SVP は、Intel 換算で約 60 ステップを 1 サイクルで実行するガウシアンフィルタ関係の複合命令を具備しており、演算のピーク値はそれに従って求めた。これによると、2 クラスタ構成の SMYLEvideo は 672GOPS の理論ピーク性能、4 クラスタ構成では 1 TOPS を超える性能を達成可能であることが分かった。また、命令セットシミュレータを用いた性能評価を行った結果、2 クラスタにそれぞれ 6 コア、4 コア、クラスタ外におかれた全体制御用の 1 コアの計 11 コアからなる最小構成メニーコアシステム上に、SIFT アルゴリズムを

搭載し、100MHz のコア動作速度にて 640*360 サイズのモノクロ動画像に対して 15fps の性能を達成できることを確認した。

次に、消費電力評価の結果について報告する。本評価においては、開発した命令レベルシミュレータ、ならびに、消費電力モデルを用いた。具体的には、SIFT 処理において高い負荷となるガウシャンフィルタに着目し、その消費電力を推定した。その結果、180nm プロセスを想定した場合には 35mW の消費電力であることが分かった。この処理は特徴点数が 2000 の場合、SIFT 処理全体のおよそ 5%となる。したがって、処理量比例と考えると SIFT 処理全体の消費電力はおおよそ 710[mW]となる。今回の実装では、これに対して MC および周辺回路の消費電力が加わるため、合計およそ 850[mW]と予測される。しかしながら、この値は 30MHz180nm 1.8V TSMC プロセスの値であるため、本事業でターゲットとする周波数 100MHz、40nm 1.1V TSMC プロセスに変換する必要がある。この場合、負荷容量は 1/3 になると仮定すると、2 クラスタ構成の SMYLvideo では、 SIFT 実行時の消費電力は約 350[mW]と予測される。

次に、回路規模に関する評価結果を報告する。RTL 設計に基づき FPGA 用の合成を行った結果を参考にし、ハードウェアの規模を算出した。その結果を表 3-7 に示す。この結果より、2 クラスタ構成の SMYLEvideo のチップサイズは、1.4mm□に約 2MB のオンチップメモリを加えた面積となる。これを加味するとおおよそ 1.7mm□の面積となると予想される。

表 3-7 ハードウェア規模(製造プロセスは 40nm を仮定)

	単体		2クラスタ	合計	
	面積[um2]	ゲート数	集積数	面積[um^2]	ゲート数
SVP	96,192	146,396	4	384,768	585,584
QVP	253,158	385,286	6	1,518,950	2,311,713
MC	19,544	29,745	1	19,544	29,745
IM(除くメモリ)	6,818	10,376	1	6,818	10,376
DM(除くメモリ)	7,967	12,125	1	7,967	12,125
MEMC	7,022	10,687	2	14,043	21,373
I/O	4,185	6,369	1	4,185	6,369
合計				1,956,275	2,977,285

以下、既存の関連製品と比較結果をまとめる。

- 平成 24 年　汎用マルチコア・プロセッサ(4 コア 約 2.9GHz≒24GOPS,130W)と比較して、性能が約 30 倍（700GOPS 相当)、消費電力 1/300 以下（350mW 程度)。
- 平成 25 年　商用の専用ハードウェアアクセラレータを具備した 4 コア程度の画像認識プロセッサに対して、電力当たり性能 3.2 倍、消費電力 1/2 以下、回路規模 1/2 以下。

4　事業の成果詳細：メニーコア向けソフトウェア開発環境の構築

4.1　狙い

メニーコア時代においては、並列化効率の良いソフトウェアを如何に低コストで開発するかが極めて重要となる。特に組込みシステムにおいては、スーパーコンピュータ等を対象とした高性能計算アプリケーション開発とは異なり、逐次処理を前提として開発されたソフトウェア資産の再利用が必須である。また、並列アプリケーション開発に関する知識や経験が十分に蓄積されているとは言い難く、真の意味でメニーコア向けソフトウェアの開発を加速するためには、一企業におけるノウハウの蓄積に留まらずオープンな形での知識の共有が必要不可欠となる。そこで、本事業においては、2014 年以降のメニーコア本格普及を見据え、以下を目標として研究開発を行った。

- プログラムの「移植」に重きを置いたソフトウェア開発環境の構築と公開：ハードウェア・プラットフォームがすでに決まっており、かつ、アプリケーションをスクラッチから開発する場合には、実行環境に応じた並列化をプログラム開発スタート時から考慮すれば良い。しかしながら、既存のソフトウエア資産を活用する場合には、様々な「改良」が必要となる。特に、極めて高い並列化効率を実現するためには、既存のソースコードを起点とした並列化だけでは不十分であり、並列アルゴリズムの再検討やデータ構造の再構築など、より高いレベルでの最適化が求められる。そこで、メニーコア・プラットフォームに対する逐次コードの移植をサポートするソフトウェア開発環境を構築する。そして、これらを可能な限りソースコードレベルで公開する。
- メニーコア向け高速ソフトウェアの開発とその公開による知識の共有：メニーコアがある種のアクセラレータとして活用されることを鑑み、将来の主要アプリケーションで核となるプログラムカーネルを選定すると供に、メニーコア向け最適化を実施して高速ソフトウェア・ライブラリとする。そして、最適化手法やその効果など、詳細を無償で公開することで、組込みシステムにおけるメニーコア・アプリケーション開発に関する知識の共有を目指す。

上記の成果を広く普及させるためには、ハードウェア・プラットフォームへの依存性を可能な限り抑制する必要がある。そこで、具体的には、本事業で開発した SMYLEref アーキテクチャでの利用に限定せず、現在巨大な市場を獲得している Intel 社の汎用プロセッサや NVIDIA 社の汎用 GPU での利用も可能な環境／ツールとして以下を開発した。

- 半自動並列化コンパイラ CLtrump（付録：最終成果報告書第 4.1 節）:「直列プログ

ラム」を「並列プログラム」に書き換えるという移植作業を支援するツール。アルゴリズム選択などプログラマによるマクロレベルの手動最適化と、ループの変形などコンパイラによるミクロレベルの自動最適化の両方が必要であるとの考えに基づき、プログラマとコンパイラのインタラクティブなアプリケーション開発を支援する。逐次 C 言語プログラムを入力とし、並列化された OpenCL プログラムが出力される。

- 移植後性能推定ツール PEMAP（付録：最終成果報告書第 4.2 節）：既存のソフトウェアをメニーコアに移植する前に移植後の性能予測を可能とするダミーコード自動生成ツール。特に産業界では、最終的な性能がある程度見込まれてから予算が確保できるケースが多く、短時間かつ低コストでの性能見積もり作業の実現に利用できる。
- 高速ソフトウェア・ライブラリ BEMAP（付録：最終成果報告書第 4.3 節）：将来のメニーコア応用が期待される主要分野から選択肢たアプリケーション・カーネル。並列化前の C 言語プログラム、並列化や各種最適化を施した OpenCL プログラム、ならびに、各最適化効果に関する評価結果を公開。メニーコアの評価におけるベンチマーク・プログラムとしての利用も可能。

4.2　半自動並列化コンパイラ CLtrump

メニーコアを対象としたプログラム開発を考えた場合、主に、①手動で時間をかけて最適化する、②自動最適化や pragma を使って手軽に最適化する、の 2 通りが考えられる。前者においては、プログラマの能力を最大限発揮して高い性能が見込まれる反面、レジスタ数の確認やループアンローリング、命令スケジューリングの変更といった時間のかかる作業が多く、作業効率が悪くなる。一方、後者の場合には、OpenMP や HMPP といった代表的なフレームワークが普及しているが、処理が複雑で適応できない場合があり、細かな最適化に対応することが難しい。このように、手動最適化では自動最適化の恩恵である作業効率の向上が受けられず、自動最適化では細部の最適化ができない、といった問題が生じる。

メニーコア向けプログラムの開発作業を見渡した場合、ループ変形などの自動化可能部分と、アルゴリズム選定やデータ構造の決定といったプログラマの介入を要する部分が混在している。これら全てを完全自動化することは非現実的であり、人手による最適化と自動最適化の両方を適材適所にバランス良く利用できる環境が望まれる。そこで、CLTrump では、自動化可能な部分はツールで自動化し、手動化が必要な部分はユーザが自ら開発するようなインタラクティブな半自動並列プログラム開発環境の実現を設計方針とした。図 3-13 に従来の開発方法と CLtrump での開発方法の比較を、また、図 3-14 に CLtrump を使ったソフトウェア開発フローを示す。

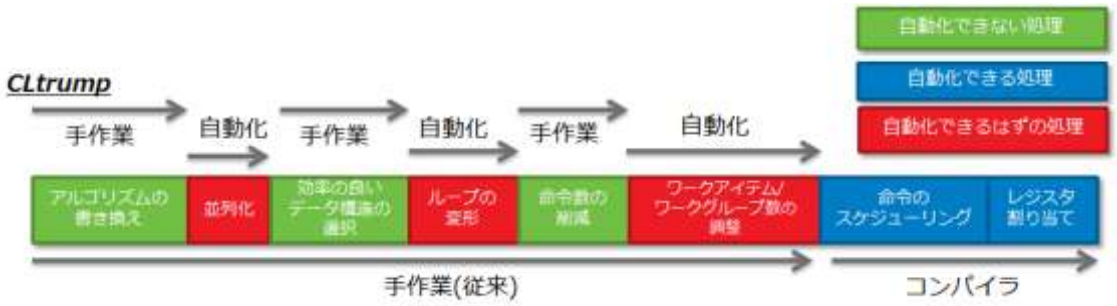


図 3-13 従来の開発方法と CLtrump での開発方法の比較

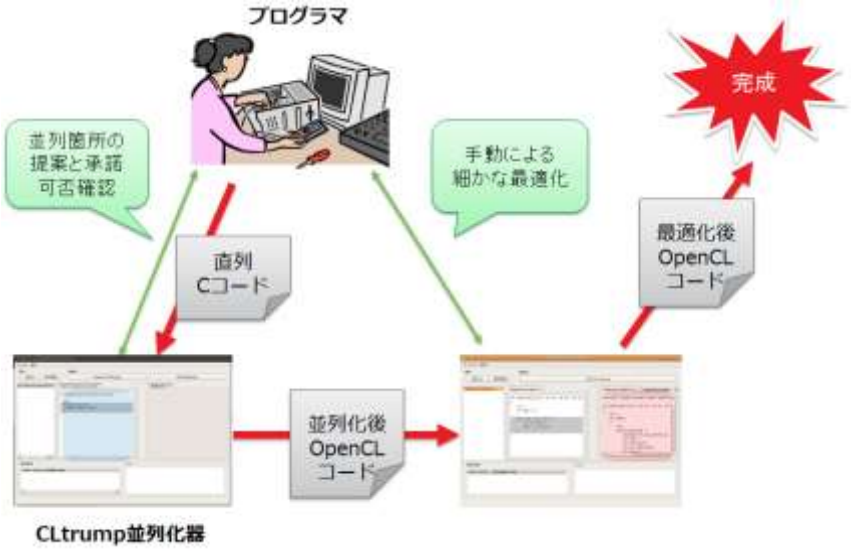


図 3-14 CLTrump を使った開発フローのイメージ

開発工程において自動化部分と手作業の部分が混在しているため、インタラクティブに並列コードを作成することが特徴的である。プログラマは、標準 C 言語で記載された直列コードを入力し、OpenCL 仕様の並列コードを得ることができる。変換時には、並列箇所の提案と承諾可否の確認といったインタラクティブな処理形態を採る。CLtrump 自体はソースからソースに変換するソースジェネレータであり、変換後のコードもソースコードとしてプログラマが解読でき、さらなる最適化を促すことができる。図 3-15 に CLtrump の画面の様子を示す。画面右に入力ファイルが表示されており、これが OpenCL 化されたソースコードへと変換される。画面左下には入力された C 言語プログラムの解析結果が示されており、ユーザはどの部分を並列化するか指定することができる。

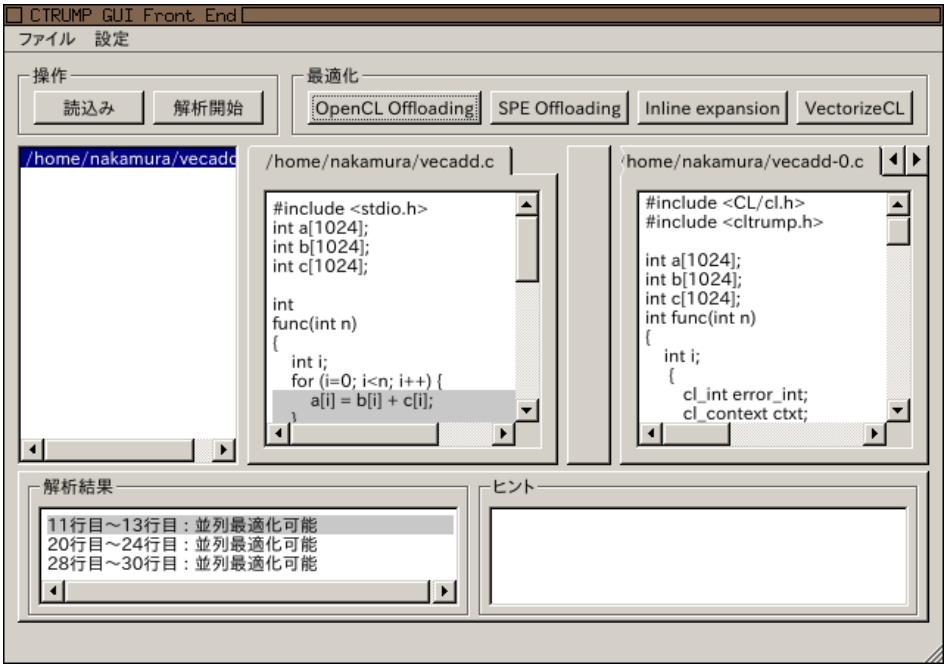


図 3-15　OpenCL 並列コードの出力

4.3　性能推定ツール PEMAP

アーキテクチャの異なるプロセッサ間の移植において、移植前と移植後の命令セットの頻度分布やメモリアクセスパターンに何らかの関連性があることに着目し、移植前のプログラムから、その特性を維持した移植後のダミーコードを自動生成する。ダミーコードは計算結果の合致を犠牲にする変わりにプログラム特性を維持する。移植後の対象プロセッサでダミーコードを実行することで性能の目安となる見積もりが可能となる。PEMAP ワークフローの全体概要を図 3-16 に示す。ユーザは、移植の対象となる C/C++ソースコードにおいて、並列化対象ループの前後に PEMAP を呼び出すためのアノテーションを追加する。そして、当該ソースコードを x86 命令向けにコンパイルして実行する。この際、実行に使用するハードウェア・プラットフォームは移植対象となるプラットフォームである必要はない。実行時における PEMAP の動作は以下の通りである。

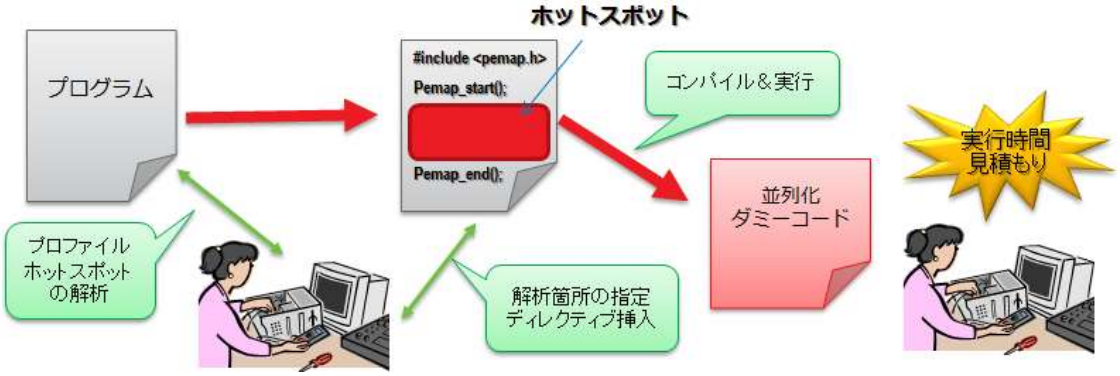


図 3-16　PEMAP のワークフローの全体概要

1. 実行において呼び出された PEMAP 関数は、予測対象ループを逆アセンブルした上で解析し、ループレベルのコントロールフローグラフ（LCFG: Loop-level Control Flow Graph）を構築する。そして、LCFG の各シーケンスを解析し、読み出し／書き換えを行うメモリアドレスやレジスタを明確にする。
2. LCFG の解析後、性能推定の対象となる関数に含まれる機械語命令から、PEMAP が内部で用いる中間表現（Intermediate Representation、 IR）を生成する。そして、生成した IR を解析して各種変換や最適化を行う。
3. 性能推定の対象となる関数に対して動的情報収集機能を埋め込みながら別領域にコピーする。収集する情報としては、①条件付き分岐命令による分岐結果、②実行におけるメモリ読み出し／書き込みアクセス（以降、動的メモリアクセスと呼ぶ）でのアドレス（ただし、実行前の静的解析が不可能なアクセスを対象）、ならびに、③性能推定の対象となる関数に含まれるループ回転数である。
4. IR から性能推定専用の並列化されたダミープログラムを自動生成する。

本技術のポイントは、並列化や各種最適化を施した移植後の性能を正しく推定できるか、すなわち、如何に移植後の最適化を想定した IR 変換（上記作業の「2」）を実現するか、にある。これを実現すべく、条件付き分岐命令の検出やメモリアクセスを行う際のアドレスがランダムなものであるかどうかを判定する解析技術、移植後の最適化を再現するための技術、およびこれらの解析・最適化技術に好適な IR を開発した。

BEMAP に含まれる Gray Scale ベンチマークを選択して PEMAP の評価を行った。Gray Scale ベンチマークは、入力された RGB 画像全体のグレースケールの画像を生成するベンチマークであるが、PEMAP のメモリアクセスパターンの評価のため、入力された RGB 画像の一部のみを処理対象とするよう修正を加えた。また、BEMAP のベンチマークは OpenCL で記述されているため、これを CUDA に修正した。実験結果を図 3-17 に示す。この結果から分かるように、様々なハードウェア・プラットフォームにおいて正しく実行時間を予測できていることが分かる。

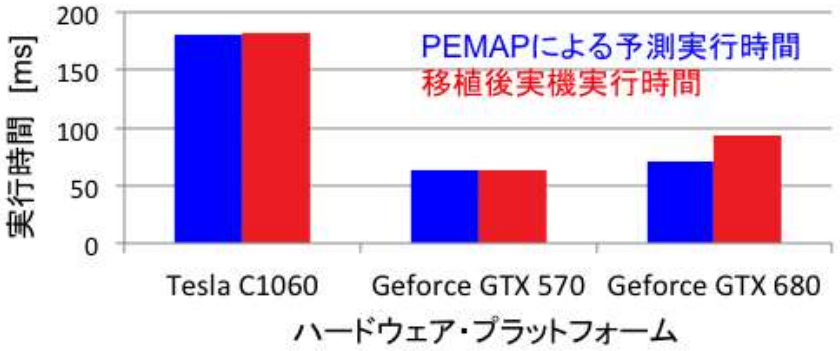


図 3-17　PEMAP での移植後実行時間予測結果

表 3-8　BEMAP の性能表

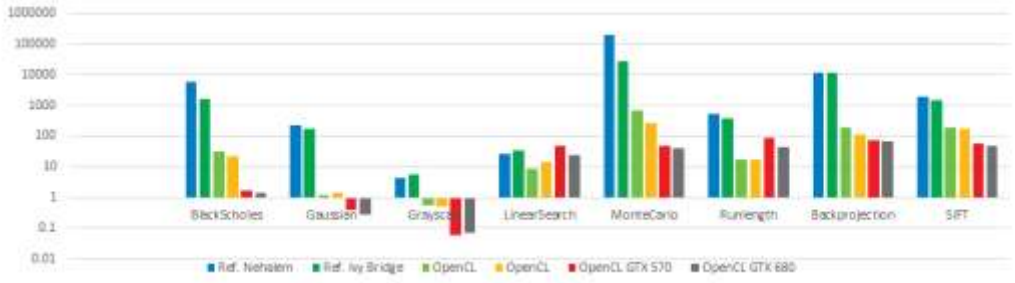
Workload	Ref. Nehalem	Ref. Ivy Bridge	OpenCL Nehalem	OpenCL Ivy Bridge	OpenCL GTX 570	OpenCL GTX 680
BlackScholes	6109.07	1587.80	32.05	21.41	1.70	1.40
Gaussian	227.42	179.30	1.24	1.40	0.39	0.27
Grayscale	4.32	5.47	0.53	0.52	0.06	0.07
LinearSearch	26.25	33.33	8.42	14.24	47.14	24.08
MonteCarlo	193817.40	27423.70	696.53	264.48	49.68	41.50
Runlength	518.28	370.50	17.07	16.77	90.29	43.36
Backprojection	11297.43	11872.04	195.03	116.08	75.83	69.70
SIFT	1860.00	1452.00	194.11	170.51	55.54	48.02

Nehalem
Intel i7-X990 @ 3.47GHz (Nehalem)
8GB memory, 6 cores, 12 threads (HT)

Ivy Bridge
Intel Core i7-3770K @ 3.50GHz (Ivy Bridge)
8GB memory, 4 cores, 8 threads (HT)

GTX 570:
NVIDIA GTX 570 @ 1.46GHz (Fermi GF-110)
480 CUDA cores, 1.2GB GDDR5 memory

GTX 680:
NVIDIA GTX 680 @ 1.06 GHz (Kepler GK-104)
1536 CUDA cores, 2GB GDDR5 memory



4.4　高速ソフトウェア・ライブラリ BEMAP

これまでフィックスターズが蓄積してきた最適化技術のノウハウを共有、さらには、メニーコアソフトウェアの開発経験が少ないユーザを対象にライブラリとしてもすぐに活用できるよう、高速処理が要求される分野の代表的なカーネルを開発しオープンソースとして公開した。2013 年 4 月現在 BEMAP では次に示す 8 種類の処理コード(ガウシアンフィルタ、ブラック・ショールズ、リニアサーチ、グレースケール化、SIFT 処理、ランレングス法、バックプロジェクション、モンテカルロ法)が含まれている。各処理コードには、シングルスレッドで動作するリファレンスコードと手動で並列化したコードがセットとなっている。手動で並列化したコードは最適化手法を指定することができるため自動並列コンパイラの性能を最適化手法ごとに手動の最適化と比較評価することが可能となる。BEMAP で公開している、すべての処理に対する最適化の技術的な説明や評価結果はレポートとして Web で公開している（http://sourceforge.jp/projects/sfnet_bemap/）。表 3-8 に BEMAP で公開しているコードの性能結果をまとめる。OpenCL のコードを測定したプロセッサは、現在市販されているプロセッサで OpenCL のコンパイラが存在しているものを選択している。Intel 社の汎用マルチコアプロセッサとして、6 コアの Nehalem と 4 コアの Ivy Bridge を選定し、GPU として NVIDIA 社の GTX570 と GTX680 で評価した。

5　事業の成果詳細：メニーコア市場調査

マルチ・メニーコアの有望な適用分野や機器市場を明らかにすることを目的とし、以下に示す市場調査を実施した。

- 全分野（付録：最終成果報告書第 5.1 節）：JEITA、キャッツ(株)、イーソル(株)による調査。市場調査に際しては、シングルコアからマルチコアを経てメニーコアへとシフトしていく市場動向を把握するため、対象をメニーコアに限定せず、マルチ・メニーコアを対象とした。マルチ・メニーコア技術のニーズは幅広く、組込み機器全般に及ぶと考えられるが、有望な適用分野、機器市場として、市場規模、成長性、日本の強み分野の観点から、日本の半導体産業が力を入れるべき分野にフォーカスして調査した。海外企業の後追いよりは組込み系、中でも日本の強み分野である制御系や制御系と情報系の融合分野においてビーイクルを選定し、開発/動作環境やAPI、レガシーソフト有効活用といった課題を解決するようなプロジェクトを起こすと共に、マルチ･メニーコア技術者の裾野を広げるような、手ごろなマルチ･メニーコアツールキットの開発・頒布が、マルチ・メニーコアの普及と日本の半導体産業の競争力強化に繋がるといえる。また、ヘテロジニアスマルチ・メニーコアが高性能化を実現する有力候補となっていることから、コア数やコア種、メモリアーキテクチャなどのハードウェアの多様性を許容して、これをソフトウェアが統合的に扱えるようにする必要がある。
- 画像処理分野（付録：最終成果報告書第 5.2 節）：トプスシステムズ社による調査。優れたメニーコア・アーキテクチャが必要とされている画像認識系・超高精度描画等の次世代画像処理である「ビジュアル・コンピューティング分野（ビデオマイニングを含む)」について、特に積極的に新たな「市場を創る」ことを目指して、展示会への出展、技術講演、アンケート調査、ユーザ候補企業訪問によるヒアリング等により市場調査を実施した。その結果、『画像認識系のシステムには、10 年後も使用可能なコンピューティング･プラットフォームが求められている』ということが、明確になった。10 年後も使用可能なコンピューティング・プラットフォームの主な要件としては、①超高性能：1TOPS ~ 1POPS (Peta Operations Per Second)、②スケーラビリティ：コア数やクラスタ数の増加に伴う演算処理能力の向上、③プログラミング・モデル：スケーラビリティを活かせる並列処理モデルの確立、④ソフトウェア開発環境：並列化支援ツールの拡充、⑤低消費電力：～1.5W 程度（組込みシステムの場合)、⑥事業継続性：大手半導体ベンダーによるサポート・製造、が重要であることが分かった。
- ライフイノベーション、インターネットアプリケーション、ファイナンス分野（付録：最終成果報告書第 5.3 節)：本分野に関する調査を行い、第 4.4 節で紹介した高速ソフトウェア・ライブラリ BEMAP の開発へと繋げた。

IV. 実用化・事業化に向けての見通し及び取り組みについて

本事業の成果の実用化／事業化に関して以下にまとめる。トプスシステムズ、フィックスターズ、いずれの企業においても、本事業原簿作成時の 2013 年 9 月の段階において顧客を獲得しており、更なる事業化拡大に向け進行中である。

	トプスシステムズ		フィックスターズ
	汎用 SoC 事業 （SMYLEref）	ビデオマイニング SoC 事業 （SMYLEvideo）	ソフトウェア開発支援事業 （CLtrump/PEMAP/BEMAP）
成果の実用化、事業化の見通し	FPMA(Field Programmable Many-core Array)というコンセプトに基づく事業化へと発展させる。既存の ASIC の置換えが期待されるデバイスとして FPGA がある。しかしながら、最大の課題はハードウェア設計を伴うことに起因する低い生産である。FPMA は、SMYLEref のメニーコア技術を核とすることでソフトウェア処理の高い生産性を維持する。そして、FPGA と遜色ない高い性能と安価で柔軟性を有する SoC プラットフォームとして提供する。	画像認識系のプラットフォームとして、自動車、医療機器、デジタルカメラ、スマートフォン／タブレット、デジタルテレビ、監視装置、ロボットなどへの応用を目指す。特に、本事業で開発した SMYLEvideo 技術をベースとし、ユーザが有する各種環境(コアの種類やソフトウェアなど)に合わせて作り込む。このように、ユーザの要求に対するカスタマイズをサポートすることで多くのユーザ獲得を狙う。2013 年 3 月よりユーザを獲得し、すでに事業として進行している。また、現在国内外の複数メーカと商談を進めている。	ソフトウェア開発支援ツール CLtrump ならびに PEMAP を活用して開発・検証されたメニーコア向け高速ソフトウェア・ライブラリ BEMAP を自社のソフトウェア・プラットフォームのブランドである M3 と統合した。本事業でのポリシと同様にオープンソースとしており、M3 搭載ハードウェアの販売、M3 のユーザ向けカスタマイズ、保守やサポート、といった事業化を進めている。2013 年 9 月現在において、すでに 2 社にて導入が決定しおり、その他にも複数の引き合いを獲得している。
事業化に向けた具体的な取り組み	本事業で開発した SMYLEref プロトタイプに関して設計データを含む全ての情報を大学から移転した。また、本事業に関して社内でプロジェクトを立ち上げ、半導体メーカと実用化開発に関して商談を進めている。	2012 年 1 月にソフトウェア開発を専門とする子会社（株式会社 Cool Soft）を設立し、2013 年 9 月までに国内外のソフトウェア開発会社 5 社と提携した。また、2013 年 7 月に米 Multicore Association にて SHIM (Software Hardware for Multi-Many Core) WG を立上げメニーコア標準モデルの策定を開始した。ハードウェアに関しては、FPGA へ SMYLEvideo を搭載する新事業を拡大するために、Xilinx 社とのパートナーシップを確立し、更に、チップの安定供給のために国内外の半導体ベンダーとの戦略的提携を打診している。	M3 と統合するためにテスト関係ツールの開発や複数ハードウェア・プラットフォーム対応などいくつかの追加開発を行った。チップベンダが新規チップを発表するに伴いソフトウェアを更新し継続的事業へと結びつける。

添付資料１

イノベーションプログラム基本計画

（別　添）

平成２０・０３・２７産局第1号
平 成 ２ ０ 年 ４ 月 １ 日

ＩＴイノベーションプログラム基本計画

１．目的

我が国が目指す高度情報通信ネットワーク社会の構築に向け、経済成長戦略大綱、ＩＴ新改革戦略、科学技術基本計画及び技術戦略マップ等に基づき、情報化の進展に伴うエネルギー消費量の増大等の課題にも考慮しつつ、その基盤となる情報通信機器・デバイス等の情報通信技術を開発し、実社会への利用を促進する。また、情報システム・ソフトウェアについて品質、信頼性及び生産性の向上を推進し、組込みソフトウェア産業強化、オープンソースソフトウェアを安心して活用するための環境整備、独創的な人材の発掘等、我が国産業競争力強化のための必要な基盤整備を実施することによって、ＩＴの利活用の深化・拡大を図り、より豊かな国民生活を実現するとともに、我が国の経済活力の向上を図ることを目的とする。

２．政策的位置付け

○「経済成長戦略大綱」（２００６年７月財政・経済一体改革会議。２００７年６月改訂、経済財政諮問会議報告）

ＩＴ革新による競争力強化、ＩＴ革新を支える産業・基盤の強化に必要な研究開発の推進に対応

○「第３期科学技術基本計画」（２００６年３月閣議決定）

国家的・社会的課題に対応した研究開発の重点推進４分野である情報通信分野、分野別推進戦略（２００６年３月総合科学技術会議）における重点分野である情報通信分野に位置づけられるもの。

○「ＩＴ新改革戦略」（２００６年１月高度情報通信ネットワーク社会推進戦略本部）

次世代のＩＴ社会の基礎となる研究開発の推進等に対応。

３．達成目標

（１） 情報経済社会を形成する上で必要不可欠な基盤技術である情報通信機器・デバイス等に関しては、「革新的な技術の確立」と「その開発成果の普及促進」を図る。

【目標】

・情報通信機器・デバイス産業の付加価値額を、２０２０年度において、２００７年度比で、約５０％増加させる。
・半導体の微細化に係る革新的基盤技術の開発（テクノロジーノード４５ｎｍ以細）
・情報家電の音声認識のタスク率（９５％以上の達成）
・革新的な大型ディスプレイ技術の開発（消費電力を現状機器と比較して約５０％以下）
・革新的なネットワーク機器技術の開発（消費電力を現状機器と比較して６０％以下）

（２） 経済社会システムの信頼性確保に大きく寄与する情報システム・ソフトウェアに関しては、品質、信頼性及び生産性の向上や産学官の開発リソースの連携強化により、「人材育成」と「ソフトウェア工学の開発」等を積極的に推進する。

【目標】

・情報サービス・ソフトウェア産業の付加価値額を、２０１５年度において、２００４年度比で、約２５％増加させる。
・組込みシステム等の不具合発生率（２０１１年度までに２００６年度比５０％減）

４．研究開発内容
［プロジェクト］

－　中　略　－

Ⅱ．省エネ革新
［ⅰ］情報ネットワークシステムの徹底的省エネの実現
（１）グリーンＩＴプロジェクト（運営費交付金）（再掲）
①概要
エネルギー需給構造の高度化を図る観点から行うものであり、ＩＴ化の進展によりネットワークを流れるデータ量が大幅に増加する中で、ＩＴ機器による消費電力量の大幅な増大に対応し、環境調和型ＩＴ社会の構築を図るため、個別のデバイスや機器に加え、ネットワーク全体での革新的な省エネルギー技術の開発を行う。
②技術目標及び達成時期
２０１２年度までに、ＩＴ機器・システムのエネルギー消費効率を２倍に向上させる基盤技術を開発する。
③研究開発期間
２００８年度～２０１２年度

（２）次世代高効率ネットワークデバイス技術開発（運営費交付金）（再掲）
①概要
エネルギー需給構造の高度化を図る観点から行うものであり、ネットワークで伝送されるデータ量の爆発的増加に伴い、関連機器の消費エネルギーが増大している中で、ネットワーク全体の消費電力量を抑制することが喫緊の課題であり、消費エネルギーの低減に大きく貢献するルータ・スイッチの高速化のための研究開発を実施するとともに、機器そのものの消費エネルギーを低減するための研究開発を実施する。
②技術的目標及び達成時期
２０１１年度までに、１チャンネルあたり４０Ｇｂｐｓ超の通信速度に対応するトラフィック計測・分析・管理技術や４０Ｇｂｐｓのインターフェース、さらなる通信速度向上（１００Ｇｂｐｓ超）を実現するハードウェア技術、ＳＦＱ（単一磁束量子）スイッチに関する基盤技術を開発する。
③研究開発期間
２００７年度～２０１１年度

（３）ＩＴＳの規格化事業（第２フェーズ）
①概要
我が国ＩＴＳ産業の振興と国際競争力強化に貢献するため、これまでの個別システム等の規格化から共通基盤の構築のための規格化に重点を移し、ＩＴＳ情報通信基盤の規格化、情報収集・活用基盤の規格化、システム社会導入条件の整備等ＩＳＯ／ＴＣ２０４に対応したＩＴＳの国際規格化等を実施。
②技術的目標及び達成時期
平成２２年度までにＩＴＳに係る標準化案を作成しＩＳＯに対して提案又は国際規格として制定する。また自動車の電子化技術に関して、次世代では日本が主導をとるべく戦略を策定。
③研究開発期間
２００６年度～２０１０年度

［ⅱ］情報機器の徹底的省エネの実現
（１）次世代大型低消費電力ディスプレイ基盤技術開発（運営費交付金）（再掲）
①概要
エネルギー需給構造の高度化を図る観点から行うものであり、次世代の大型液晶及び大型プラズマディスプレイに関する低消費電力ディスプレイを実現するための研究開発を行う。

②技術的目標及び達成時期

２０１１年度までに、液晶に関しては、高効率バックライト、革新的なＴＦＴアレイプロセス技術・製造装置及び低消費電力型の画像処理エンジン等に係る技術を確立する。また、プラズマディスプレイに関しては、超低電圧駆動等に係る技術を確立する。

③研究開発期間

２００７年度～２０１１年度

［iii］省エネを支えるプロセス基盤技術

（１）パワーエレクトロニクスインバータ基盤技術開発（運営費交付金）（再掲）

①概要

エネルギー需給構造の高度化を図る観点から行うものであり、省エネルギーを進めるために、シリコンよりも材料特性に優れたワイドギャップ半導体デバイスを用いた高効率インバータ等の実用パワーエレクトロニクス機器システムの基盤技術の開発を行う。

②技術目標及び達成時期

２００８年度までに、ワイドギャップ半導体デバイスを用いた高効率インバータ等の実用パワーエレクトロニクス技術を開発する。

③研究開発期間

２００６年度～２００８年度

（２）ナノエレクトロニクス半導体新材料・新構造技術開発－うち窒化物系化合物半導体基盤・エピタキシャル成長技術の開発（運営費交付金）（再掲）

①概要

窒化物系化合物半導体は日本が強みを有し、パワーデバイス、高周波デバイス、発光デバイス等、今後のIT社会を支えとなることを期待されている分野である。しかし、既存のバルク単結晶基板成長技術やエピタキシャル成長技術では、従来の半導体では実現できない領域で動作可能なハイパワー、超高効率デバイス性能を十分に引き出すには至っていない。

これを突破するため、大学あるいは研究所を拠点に材料メーカー、デバイスメーカー、装置メーカー等が相互連携して、窒化物半導体の結晶欠陥低減技術やナノ構造作製技術等の革新を図り、これらデバイスの飛躍的な性能向上と消費電力削減の実現を図る。

②技術目標及び達成時期

２０１１年度までに、次世代窒化物系半導体デバイスを実現する以下結晶作製技術を開発する。

●基板技術（ＧａＮ、ＡｌＮバルク結晶作製技術）

・　口径２～４インチで高品質エピ成膜を可能とする低コストの単結晶基板作製技術の確立。

●エピ技術（エピタキシャル成膜及び計測評価技術）

・　低欠陥高品質エピ層を実現する成膜技術及び膜成長過程を計測評価する技術の確立。
・　高出力かつ高安定動作可能なエピ層の実現
・　高耐圧超高速な新しいデバイス構造の開発

③研究開発期間

２００７年度～２０１１年度

－　中　略　－

５．政策目標の実現に向けた環境整備

【法律】

・　情報処理の振興を目的に、昭和４５年に情報処理の促進に関する法律が制定。
・　半導体集積回路の回路配置の適正な利用の確保を目的に、昭和６３年に半導体集積回路の回路配置に関する法律が制定。

【税制】

・　情報セキュリティ強化を確保しつつ生産性の向上を図るためのＩＴ投資に対し、３５％特別償却又は７％税額控除（情報基盤強化税制）。
・　ソフトウェアを含む機械装置等に対し、３０％特別償却又は７％税額控除（中小企業投資促進税

制)。

【国際標準化】
　各プロジェクトで得られた成果のうち、標準化すべきものについては、適切な標準化活動（国際規格（ＩＳＯ／ＩＥＣ))、日本工業規格（ＪＩＳ)、その他国際的に認知された標準の提案等）を実施する。特に、産学連携ソフトウェア工学の実践における組込みソフトウェア開発については、国際標準の動向を踏まえた開発を促進することにより、プロジェクトの成果の幅広い普及を促進する。

【関係機関との連携】
　各プロジェクトのうち、研究開発を効率的・効果的に推進する観点から関係機関との連携が必要なものについては、これを積極的に行う。
　但し、関係機関が行う研究開発等の独自性を妨げるものではない。

【導入普及促進〕
　成果の普及を図るため、これまでの終了プロジェクトの成果の全部または、一部についてはオープンソースソフトウェアとして公開する。

【プロジェクト等の間の連携について】
　高信頼な組込みソフトウェアの開発では、ソフトウェアエンジニアリングセンター（ＳＥＣ）において提供される各種エンジニアリング手法を開発現場に適用し、当該技術の効果を明らかにしながら開発を進める。

【その他】
・グラント事業
　ＮＥＤＯの産業技術研究助成事業を活用し、萌芽的・革新的な情報通信関係の技術シーズの発掘を行う。また、ソフトウェア分野の独創的な技術やビジネスシーズを有した人材を発掘する。
・事業終了後の連携
　産学官連携の研究体制を通して活動を行い、これらの事業の終了後も各分野の研究者・技術者が有機的に連携し、更に新たな研究を作り出す環境を構築する。
・人材育成
　ハードウェア分野においては、出来る限り大学との連携を重視し、各種フェローシップ制度を活用しつつ、最先端の情報通信基盤研究現場への学生等の参画を推進することにより次世代の研究開発人材の育成を図る。また、ソフトウェア分野における独創的な人材を発掘し、育成するとともに、優秀な人材が集うコミュニティを構築するなど、発掘された人材の才能をさらに伸ばすための取組を進める。
・広報／啓発
　毎年１０月を「情報化月間」としている。

６．研究開発の実施に当たっての留意事項
　事業の全部又は一部について独立行政法人の運営費交付金により実施されるもの（事業名に（運営費交付金）と記載したもの）は、中期目標、中期計画等に基づき、運営費交付金の総額の範囲内で、当該独立行政法人の裁量によって実施されるものである。

７．改訂履歴
（１）　平成１２年１２月２８日付け、情報通信基盤高度化プログラム基本計画を制定。
（２）　平成１４年2月２８日付け、情報通信基盤高度化プログラム基本計画及び次世代半導体デバイスプロセス等基盤技術プログラム基本計画を制定。情報通信基盤高度化プログラム基本計画（平成１２・１２・２７工総第１２号）は廃止。
（３）　平成１５年1月３１日付け、情報通信基盤高度化プログラム基本計画及び次世代半導体デバイスプロセス等基盤技術プログラム基本計画を制定。情報通信基盤高度化プログラム基本計画（平成１４・０２・２５産局第１７号）及び次世代半導体デバイスプロセス等基盤技術プログラム基

本計画（平成１４・０２・２５産局第１８号）は、廃止。

（４）　平成１５年3月１０日付け、情報通信基盤高度化プログラム基本計画、次世代半導体デバイスプロセス等基盤技術プログラム基本計画、次世代ディスプレイ技術開発プログラム基本計画及び情報通信基盤ソフトウェア開発推進プログラム基本計画を制定。情報通信基盤高度化プログラム基本計画（平成１５・０１・２９産局第１号）及び次世代半導体デバイスプロセス等基盤技術プログラム基本計画（平成１５・０１・２９産局第２号）は、廃止。

なお、情報通信機器高度化プログラム基本計画（平成１５・０１・２９産局第１号）及び次世代半導体デバイスプロセス等基盤技術プログラム基本計画（平成１５・０１・２９産局第２号）の一部は、次世代ディスプレイ技術開発プログラム基本計画及び情報通信基盤ソフトウェア開発推進プログラム基本計画へ移行。

（５）　平成１６年２月３日付け、高度情報通信機器・デバイス基盤プログラム基本計画及び情報通信基盤ソフトウェア開発推進プログラム基本計画を制定。情報通信機器高度化プログラム基本計画（平成１５・０３・０７産局第１４号）、次世代半導体デバイスプロセス等基盤技術プログラム基本計画（平成１５・０３・０７産局第７号）、次世代ディスプレイ技術開発プログラム基本計画（平成１５・０３・０７産局第４号）は、高度情報通信機器・デバイス基盤プログラム基本計画に統合することとし、廃止。また、情報通信基盤ソフトウェア開発推進プログラム基本計画（平成１５・０３・０７産局第１４号）は、廃止。

（６）　平成１７年3月２５日付け、高度情報通信機器・デバイス基盤プログラム基本計画を制定。高度情報通信機器・デバイス基盤プログラム基本計画（平成１６・０２・０３産局第１号）は廃止。また、平成１７年3月３１日付け、情報通信基盤ソフトウェア開発推進プログラム基本計画を制定。情報通信基盤ソフトウェア開発推進プログラム基本計画（平成１６・０２・０３産局第２号）は廃止。

（７）　平成１８年3月３１日付け、高度情報通信機器・デバイス基盤プログラム基本計画及び情報通信基盤ソフトウェア開発推進プログラム基本計画を制定。高度情報通信機器・デバイス基盤プログラム基本計画（平成１７・０３・２５産局第７号）及び情報通信基盤ソフトウェア開発推進プログラム基本計画（平成１７・０３・２５産局第６号）は廃止。

（８）　平成１９年４月２日付け、高度情報通信機器・デバイス基盤プログラム基本計画及び情報通信基盤ソフトウェア開発推進プログラム基本計画を制定。高度情報通信機器・デバイス基盤プログラム基本計画（平成１８・０３・３１産局第４号）及び情報通信基盤ソフトウェア開発推進プログラム基本計画（平成１８・０３・３１産局第５号）は廃止。

（９）　平成２０年４月１日付け、ＩＴイノベーションプログラム基本計画を制定。情報通信機器高度化・デバイス基盤プログラム基本計画（平成１９・０３・１２産局第７号）及び情報通信基盤ソフトウェア開発推進プログラム基本計画（平成１９・０３・１２産局第８号）は、本プログラム基本計画に統合することとし、廃止。

平成 20･03･24 産局第１号
平 成 ２ ０ 年 ４ 月 １ 日

ＩＴイノベーションプログラム基本計画

１．目的

このプログラムは、情報通信、ライフサイエンス、環境、エネルギーなど、あらゆる分野に対して高度化あるいは不連続な革新（ジャンプアップ）をもたらすナノテクノロジー及び革新的部材技術を確立するとともに、その実用化や市場化を促進することで、我が国産業の国際競争力の維持・強化や解決困難な社会的課題の克服等を可能とすることを目的とする。

２．政策的位置付け

○第３期科学技術基本計画（２００６年３月閣議決定）

- 「ナノテクノロジー・材料分野」は、特に重点的に研究開発を推進すべき分野（重点推進４分野）の一つに指定されていて、優先的に資源配分することとされている。
- 我が国の材料技術は、基礎研究から応用研究、素材、部材の実用化に至るまでの全ての段階において世界のトップレベルを堅持しており、我が国製造業の国際競争力の源泉となっている。

○「イノベーション２５」（２００７年６月閣議決定）

- 「ナノテクノロジー・材料分野」は、中長期的に取り組むべき課題として、「１．生涯健康な社会形成」、「２．安全・安心な社会形成」、「４．世界的課題解決に貢献する社会形成」、及び「５．世界に開かれた社会形成」の分野に位置付けられている。
- 所要の措置を講じていくことが必要である事項として以下の点が指摘されている。
  - 学際領域・融合領域における教育等人材育成、拠点形成
  - 社会受容を促すための積極的な取り組み
  - 知的財産確保のための戦略的な取り組み

○「経済成長戦略大綱」（２００６年７月財政・経済一体改革会議）

- 「我が国の国際競争力の強化」の取り組みとして、高度な部品・材料産業やモノ作り中小企業の強化が掲げられている。
- 「技術戦略マップ」の活用等により、ユーザー企業との垂直連携による研究開発を推進することを通して、我が国経済発展の基盤である高品質、高性能な部品・材料産業の強化を図ることが今後の取組として記載されている。

○「新産業創造戦略２００５」（２００５年６月経済産業省）

- 部材分野は、新産業群の創出を支える共通基盤技術として位置づけられている。
- 「高度部材・基盤産業」の集積を形成していることが、「ものづくり」に不可欠な基盤技術のネットワーク化を通じた現場レベルでの迅速かつ高度な摺り合わせを可能としており、我が国「ものづくり」の強みの源泉となっていると記載されている。

３．達成目標

- 世界に先駆けて、ナノテクノロジーを活用した非連続な技術革新を実現する。
- 我が国部材産業の強みを更に強化することで、他国の追随を許さない競争優位を確保するとともに部材産業の付加価値の増大を図る。
- ナノテクノロジーや高機能部材の革新を先導することで、これら部材を活用した情報通信、ライフサイエンス、環境、エネルギーなどの幅広い産業の付加価値の増大を図る。
- 希少金属などの資源制約の打破、圧倒的な省エネルギー社会の実現など、解決困難な社会的課題の克服を目指す。

４．研究開発内容

［プロジェクト］

Ⅰ．ナノテクノロジーの加速化領域

ナノテクノロジーを活用した不連続な技術革新を加速・促進する。

（１）異分野異業種融合ナノテクチャレンジ（運営費交付金）

①概要

革新的なナノテクノロジーを活用し、川上と川下の連携、異業種異分野の連携で行う部材開発に

対して支援を行い、燃料電池、ロボット、情報家電、健康・福祉・機器・サービス、環境・エネルギー・機器・サービスの５分野に資するキーデバイスの実現を目指す。

②技術目標及び達成時期

マテリアル・プロセス研究、加工・計測技術研究、昨今の環境意識向上に対応した研究、社会課題を解決するための基盤技術研究に加え、異分野等の融合研究を推進することにより、２０１１年度までにナノテクノロジーの産業化のための基盤的技術を確立し、実用化を図る。

③研究開発期間

２００７年度～２０１１年度

（２）ナノテク・先端部材実用化研究開発（運営費交付金）

①概要

新産業創造戦略の趣旨に則り、革新的なナノテクノロジーを活用し、川上と川下の連携、異業種・異分野の連携で行うデバイス化開発の支援を行うため、

○ナノテクノロジー活用による材料・部材の高度化を図る先導的研究開発（ステージⅠ）

○ナノテクノロジー研究成果の部材等への課題設定型実用化により目指した開発支援（ステージⅡ）

について提案公募を実施する。

②技術目標及び達成時期

２０１０年頃に想定される半導体微細加工の限界を克服するため、分子・原子を１つずつ制御し部品部材に組み上げる「ボトムアップ型」のナノテクノロジーなど革新的なナノテクノロジー等の活用により、情報家電・ロボット、燃料電池等新規産業５分野等において、従来の性能・効率を大幅に改善するナノテク・先端部材技術を開発し、我が国が優位にあるナノテクノロジーを基盤とした国際的な産業競争力を強化することを目標とする。

③研究開発期間

２００５年度～２０１１年度

Ⅱ．情報通信領域

ナノテクノロジーや革新的部材開発技術を駆使して既存技術の微細化の壁を突破し、電子デバイス・光デバイスで世界をリードするとともに、高度化された製造技術の開発を行う。

（１）ナノエレクトロニクス半導体新材料・新構造技術開発－うち新材料・新構造ナノ電子デバイス

①概要

ナノエレクトロニクスは、ナノテクノロジーの最大の応用領域の一つであり、デジタル・デバイスのCMOS構造というアーキテクチャは、優れた工学概念である。また、これまでの半導体技術の微細化に基づく高集積化・高速化・低消費電力化の追求は、シリコン材料をベースとするプレーナ構造を基本とした微細加工プロセスの高度化にあった。

しかし、さらなる微細化によるデバイスのパフォーマンス向上は物理的限界に直面しつつあり、問題は、ＦＥＴを、シリコン材料をベースとして作製することにより現出していると考えられる。

そのため、次世代の電子デバイスのために「シリコンで培った微細化技術やデバイス原理をこれまで同様に活用しながら、シリコンという材料の物理的限界を突破するための“新材料”や“新（デバイス）構造”を実現すること」、すなわち、「Ｎｅｗ　Ｎａｎｏ　Ｍａｔｅｒｉａｌｓ／Ｓｔｒｕｃｔｕｒｅ　ｏｎ　Ｓｉｌｉｃｏｎ　ｆｏｒ“Ｍｏｒｅ　Ｍｏｏｒｅ”」の半導体技術を、ナノテクノロジーを最大限に活用することによって研究開発を行い、将来の産業応用への目を見出していく取りかかりとする。

②技術目標及び達成時期

２０１１年度までに、産業界が１０年後を見据えた将来の電子デバイスを開発する際に、産業技術として活用できるかどうかの実現可能性を見極め、また技術シーズを確立する。

③研究開発期間

２００７年度～２０１１年度

（２）ナノエレクトロニクス半導体新材料・新構造技術開発－うち窒化物系化合物半導体基板・エピタキシャル成長技術の開発（運営費交付金）（再掲）

①概要

窒化物系化合物半導体は日本が強みを有し、パワーデバイス、高周波デバイス、発光デバイス等、今後のＩＴ社会を支えとなることを期待されている分野である。しかし、既存のバルク単結晶基板

成長技術やエピタキシャル成長技術では、従来の半導体では実現できない領域で動作可能なハイパワー、超高効率デバイス性能を十分に引き出すには至っていない。

これを突破するため、大学あるいは研究所を拠点に材料メーカー、デバイスメーカー、装置メーカー等が相互連携して、窒化物半導体の結晶欠陥低減技術やナノ構造作製技術等の革新を図り、これらデバイスの飛躍的な性能向上と消費電力削減の実現を図る。

②技術目標及び達成時期

２０１１年度までに、次世代窒化物系半導体デバイスを実現する以下結晶作製技術を開発する。

１）基板技術（ＧａＮ、ＡｌＮバルク結晶作製技術）

・口径２～４インチで高品質エピ成膜を可能とする低コストの単結晶基板作製技術の確立。

２）エピ技術（エピタキシャル成膜及び計測評価技術）

・低欠陥高品質エピ層を実現する成膜技術及び膜成長過程を計測評価する技術の確立。

・高出力かつ高安定動作可能なエピ層の実現

・高耐圧超高速な新しいデバイス構造の開発

③研究開発期間

２００７年度～２０１１年度

－　以下　省略　－

平成２０・０３・２５産局第５号
平 成 ２ ０ 年 ４ 月 １ 日

エネルギーイノベーションプログラム基本計画

１．目的

資源に乏しい我が国が、将来にわたり持続的発展を達成するためには、革新的なエネルギー技術の開発、導入・普及によって、各国に先んじて次世代型のエネルギー利用社会の構築に取り組んでいくことが不可欠である。他方、エネルギー技術開発は、長期間を要するとともに大規模投資を伴う一方で将来の不確実性が大きいことから、民間企業が持続的な取組を行うことは必ずしも容易ではない。このため、政府が長期を見据えた将来の技術進展の方向性を示し、官民双方がこの方向性を共有することで、将来の不確実性に対する懸念が緩和され、官民において長期にわたり軸のぶれない取組の実施が可能となる。以下に５つの政策の柱毎に目的を示す。

１－Ⅰ．総合エネルギー効率の向上

１９７０年代以来、官民をあげて省エネルギーに取り組み、産業構造の転換や新たな製造技術の導入、民生機器の効率改善等により世界最高水準の省エネルギーを達成している。今後、「新・国家エネルギー戦略」に掲げる、２０３０年までにＧＤＰあたりのエネルギー利用効率を約３０％向上を実現していくためには、産業部門はもとより、全部門において、総合エネルギー効率の向上に資する技術開発とその成果の導入を促進する。

１－Ⅱ．運輸部門の燃料多様化

ほぼ１００％を石油に依存する運輸部門は、我が国エネルギー需給構造上、最も脆弱性が高く、その需給構造の次世代化は、将来に向けた早急な対策が不可欠な課題となっている。

「新・国家エネルギー戦略」に掲げる目標（２０３０年に向け、運輸部門の石油依存度が８０％程度となることを目指す）の実現のためにも、官民が中長期的な展望・方向性を共有しつつ、技術開発と関連施策を推進する。

１－Ⅲ．新エネルギー等の開発・導入促進

太陽光、風力、バイオマスなどの新エネルギーは、エネルギー源の多様化や地球温暖化対策の観点から重要である。しかし、現時点では経済性や出力安定性といった普及へ向けての課題が存在する。

そのため、これらの課題解決に向けた技術開発の推進及び新エネルギーの導入促進のための関連施策の実施により、更なる新エネルギーの普及を推進する。

１－Ⅳ．原子力等利用の推進とその大前提となる安全の確保

原子力発電は供給安定性に優れ、運用時にＣＯ２を排出しないクリーンなエネルギー源である。安全確保を大前提に核燃料サイクルを含む原子力発電を着実に推進する。

１－Ⅴ．化石燃料の安定供給確保と有効かつクリーンな利用

化石燃料資源の大宗を輸入に依存する我が国にとって、その安定供給の確保は国家安全保障に直結する課題である。このため、石油・天然ガス等の安定供給確保を目指し、我が国企業による資源国における資源開発等に対する支援等の施策を進めるとともに、その有効かつクリーンな利用を図る。

２．政策的位置付け

○ エネルギー基本計画（２００７年３月閣議決定）

重点的に研究開発のための施策を講ずべきエネルギーに関する技術及びその施策として、

１．総合エネルギー効率の向上に資する技術
２．原子力利用の推進とその大前提となる安全の確保に資する技術
３．運輸部門のエネルギー多様化に資する技術
４．新エネルギーに関する技術
５．化石燃料の安定供給確保と有効かつクリーンな利用に資する技術

以上が位置づけられている。

○ 新・国家エネルギー戦略（２００６年５月）

世界最先端のエネルギー需給構造の実現を図るため

１．省エネルギーフロントランナー計画
２．運輸エネルギーの次世代化計画
３．新エネルギーイノベーション計画
４．原子力立国計画

以上の計画が位置づけられている。また、資源外交、エネルギー環境協力の総合的な強化を図るため、「総合資源確保戦略」が位置づけられている。

○ 第３期科学技術基本計画（２００６年３月閣議決定）

国の存立にとって基盤的であり国として取り組むことが不可欠な研究開発課題を重視して研究開発を推進する「推進４分野」であるエネルギー分野、分野別推進戦略（２００６年３月総合科学技術会議）における「推進４分野」であるエネルギー分野に位置付けられている。

○ 経済成長戦略大綱（２００６年７月財政・経済一体改革会議）

資源・エネルギー政策の戦略的展開として

１．省エネルギーフロントランナー計画
２．次世代自動車・燃料イニシアティブ等による運輸エネルギー次世代化
３．新エネルギーイノベーション計画
４．原子力立国計画
５．資源外交、環境・エネルギー協力等の総合的な強化

以上が位置づけられている。

○ 京都議定書目標達成計画（２００５年４月閣議決定）

「京都議定書の約束を達成するとともに、更に「脱温暖化社会」に向けて長期的・継続的な排出削減を進めるには、究極的には化石燃料への依存を減らすことが必要である。環境と経済の両立を図りつつ、これらの目標を達成するため、省エネルギー、未利用エネルギーの利用等の技術革新を加速し、効率的な機器や先進的なシステムの普及を図り、世界をリードする環境立国を目指す。」とされている。

３．達成目標

３－Ⅰ．総合エネルギー効率の向上

転換部門における「エネルギー転換効率向上」、産業部門における「製造プロセス向上」、民生・運輸部門における「省エネルギー」などにより、エネルギー消費効率を２０３０年度までに少なくても３０％改善することを目指す。

３－Ⅱ．運輸部門の燃料多様化

バイオマス由来燃料、ＧＴＬ、ＢＴＬ、ＣＴＬなどの新燃料、電気自動車や燃料電池自動車などの導入により、現在ほぼ１００％の運輸部門の石油依存度を２０３０年までに８０％程度とすることを目指す。

３－Ⅲ．新エネルギー等の開発・導入促進

太陽光、風力、バイオマスなどの新エネルギーの技術開発や燃料電池など革新的なエネルギー高度利用を促進することにより、新エネルギー等の自立的な普及を目指すことで、エネルギー源の多様化及び地球温暖化対策に貢献する。

３－Ⅳ．原子力等利用の推進とその大前提となる安全の確保

２０３０年以降においても、発電電力量に占める比率を３０～４０％程度以上とすることを目指すため、高速増殖炉サイクルの早期実用化、既設軽水炉代替へ対応する次世代軽水炉の開発、軽水炉技術を前提とした核燃料サイクルの確立、放射性廃棄物対策などの技術開発を推進する。

３－Ⅴ．化石燃料の安定供給確保と有効かつクリーンな利用

石油・天然ガスの化石燃料の安定供給確保を目指し、資源獲得能力の強化に資する先端的な技術開発を推進するとともに、環境負荷低減のために化石燃料の効率的かつクリーンな利用を促進するための技術開発・導入を目指す。

４．研究開発内容

４－Ⅰ．総合エネルギー効率の向上

－　中　略　－

４－Ⅰ－vi．次世代省エネデバイス技術

（１）パワーエレクトロニクスインバータ基盤技術開発（運営費交付金）

①概要

エネルギー需給構造の高度化を図る観点から行うものであり、省エネルギーを進めるために、シリコンよりも材料特性に優れたワイドギャップ半導体デバイスを用いた高効率インバータ等の実用パワーエレクトロニクス機器システムの基盤技術の開発を行う。

②技術目標及び達成時期

２００８年度までに、ワイドギャップ半導体デバイスを用いた高効率インバータ等の実用パワーエレクトロニクス技術を開発する。

③研究開発期間

２００６年度～２００８年度

（２）ナノエレクトロニクス半導体新材料・新構造技術開発　－うち窒化物系化合物半導体基板・エピタキシャル成長技術の開発（運営費交付金）

①概要

窒化物系化合物半導体は日本が強みを有し、パワーデバイス、高周波デバイス、発光デバイス等、今後のＩＴ社会を支えとなることを期待されている分野である。しかし、既存のバルク単結晶基板成長技術やエピタキシャル成長技術では、従来の半導体では実現できない領域で動作可能なハイパワー、超高効率デバイス性能を十分に引き出すには至っていない。

これを突破するため、大学あるいは研究所を拠点に材料メーカー、デバイスメーカー、装置メーカー等が相互連携して、窒化物半導体の結晶欠陥低減技術やナノ構造作製技術等の革新を図り、これらデバイスの飛躍的な性能向上と消費電力削減の実現を図る。

②技術目標及び達成時期

２０１１年度までに、次世代窒化物系半導体デバイスを実現する以下結晶作製技術を開発する。

- 基板技術（ＧａＮ、ＡｌＮバルク結晶作製技術）
  - 口径２～４インチで高品質エピ成膜を可能とする低コストの単結晶基板作製技術の確立。
- エピ技術（エピタキシャル成膜及び計測評価技術）
  - 低欠陥高品質エピ層を実現する成膜技術及び膜成長過程を計測評価する技術の確立。
  - 高出力かつ高安定動作可能なエピ層の実現
  - 高耐圧超高速な新しいデバイス構造の開発

③研究開発期間

２００７年度～２０１１年度

（３）次世代低消費電力半導体基盤技術開発（ＭＩＲＡＩ）（運営費交付金）

①概要

エネルギー需給構造の高度化を図る観点から行うものであり、ＩＴ化の進展、ＩＴ利活用の高度化を支え、あらゆる機器に組み込まれている半導体の低消費電力化を図るため、テクノロジーノード（微細化レベル）４５ｎｍ以細の次世代低消費電力半導体を実現するため、微細加工の基盤技術やマスク（半導体素子製造過程で用いる原板）の低コスト化・製造時間短縮に必要な基盤技術の開発等を行う。

②技術目標及び達成時期

２０１０年度までに、マスク設計・描画・検査の各工程に共通的なマスクデータ処理技術、繰り返しパターンやパターン重要度を利用した描画・検査高速化技術等の基本的な開発及びＥＵＶＬマスク基盤技術として、許容欠陥の指標明確化、ブランクスの位相欠陥検査技術の確立等を完了する。

③研究開発期間

２００１年度～２０１０年度

（４）半導体アプリケーションチッププロジェクト（運営費交付金）

①概要

エネルギー需給構造の高度化を図る観点から行うものであり、ＩＴ化の進展、ＩＴ利活用の高度化を支え、あらゆる機器に組み込まれている半導体の低消費電力化を図るため、情報通信機器、特に、情報家電の低消費電力化を実現できる半導体アプリケーションチップ技術の開発を行う。

②技術目標及び達成時期

２００９年度までに、情報家電の低消費電力化を実現できるアプリケーションチップ技術を開発する。

③研究開発期間

２００３年度～２００９年度

（５）次世代高度部材開発評価基盤の開発（ＣＡＳＭＡＴ２）（運営費交付金）

①概要

エネルギー需給構造の高度化を図る観点から行うものである。半導体産業分野で、集積回路の消費電力低減に必要な配線形成用各種材料等の開発のネックとなっているナノレベルでの材料間の相互影響を評価可能な統合部材開発支援ツールを開発する。これにより、集積回路の種類やデザインルールに応じて、配線形成用各種材料とプロセスの最適な組み合わせの提案技術（統合的材料ソリューション提案技術）を確立する。

②技術的目標及び達成時期

２００８年度までに、半導体材料開発に貢献する材料評価基盤を構築するとともに、上記の統合的材料ソリューション提案技術を確立する。また、本プロジェクトを通して得られた基礎データ等については、プロジェクト実施期間中にデータを体系的に整理し、幅広く社会に提供を図る。

③研究開発期間

２００６年度～２００８年度

（６）次世代プロセスフレンドリー設計技術開発（運営費交付金）

①概要

エネルギー需給構造の高度化を図る観点から行うものであり、ＩＴ化の進展、ＩＴ利活用の高度化を支え、あらゆる機器に組み込まれている半導体の低消費電力化を図るため、テクノロジーノード４５ｎｍ以降の半導体に対応するＳｏＣ（Ｓｙｓｔｅｍ ｏｎ Ｃｈｉｐ）設計技術を開発する。具体的には、テクノロジーノード４５ｎｍ以細の半導体の共通設計基盤技術開発として、ＤＦＭ（Ｄｅｓｉｇｎ Ｆｏｒ Ｍａｎｕｆａｃｔｕｒｉｎｇ）基盤技術を中核とした設計及び製造の全体最適を確保する全く新しいＳｏＣ製造フローを開発する。

②技術目標及び達成時期

テクノロジーノード４５ｎｍ以細のＳｏＣ開発において製造性を考慮した共通設計基盤技術を確立し、システムＬＳＩデバイスの省エネルギーを実現するとともに、設計生産性を従来予想に比べ2倍にすることを目標とする。

③研究開発期間

２００６年度～２０１０年度

－ 中 略 －

５．政策目標の実現に向けた環境整備（成果の実用化、導入普及に向けた取組）

５－Ⅰ．総合エネルギー効率の向上

- 事業者単位の規制体系の導入
- 住宅・建築物に係る省エネルギー対策の強化
- セクター別ベンチマークアプローチの導入と初期需要創出（高効率機器の導入補助等）
- トップランナー基準の対象機器の拡充等
- アジアにおける省エネルギー対策の推進を通じた我が国の国際競争力の向上

- 国民の省エネルギー意識の高まりに向けた取組

５－Ⅱ．運輸部門の燃料多様化

- 公共的車両への積極的導入
- 燃費基準の策定・改定
- アジアにおける新エネルギー協力
- 国際標準化による国際競争力向上

５－Ⅲ．新エネルギー等の開発・導入促進

- 事業者支援補助金等による初期需要創出
- 新エネルギーベンチャービジネスに対する支援の拡大
- 新エネルギー産業構造の形成
- 電気事業制度・ガス事業制度の在り方の検討

５－Ⅳ．原子力利用の推進とその大前提となる安全の確保

- 電力自由化環境下での原子力発電の新・増設の実現
- 資源確保戦略の展開
- 次世代を支える人材育成
- 中小型炉の海外市場への展開、我が国原子力産業の国際展開支援
- 原子力発電拡大と核不拡散の両立に向けた国際的枠組み作りへの積極的関与
- 国と地域の信頼強化

５－Ⅴ．化石燃料の安定供給確保と有効かつクリーンな利用

- 資源国等との総合的な関係強化（研究開発の推進・協力、人材育成・技術移転、経済関係強化など）
- 化石燃料のクリーンな利用の開拓

６．研究開発の実施に当たっての留意事項

事業の全部又は一部について独立行政法人の運営費交付金による実施されるもの（事業名に（運営費交付金）と記載したもの）は、中期目標、中期計画等に基づき、運営費交付金の総額の範囲内で当該独立行政法人の裁量によって実施されるものである。

また、事業名に（採択テーマ）と記載された事業は、提案公募事業により採択されたテーマを記載したものであり、その採択や評価等は、提案公募事業の実施機関の責任の下、実施されるものである。

添付資料２

P09003

（ＩＴイノベーションプログラム・エネルギーイノベーションプログラム）
「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）」基本計画

電子・材料・ナノテクノロジー部

１．研究開発の目的・目標・内容

（１）研究開発の目的

地球温暖化問題は、世界全体で早急に取り組むべき最重要課題であり、経済・社会活動と地球環境の調和を実現するためには、画期的な技術革新が求められている。

ＩＴ機器の高度化・設置台数の急激な増加に加え、ブロードバンド通信の普及等により社会で扱う情報量は急激に増大しつつある。今後、高精細な動画コンテンツなど大容量データが、ネットワーク情報端末を介して流れ、本格的なユビキタス時代を迎える２０１５年ごろには、膨大な数の末端のセンサノードまで情報が行き交う。それに伴い、ネットワークシステムを構成するＩＴ機器が消費する電力も増大し、その省エネルギー化が重要な課題となっている。将来のネットワークシステム等の省電力化のためには、基幹系・アクセス系ネットワーク、ネットワーク端末だけでなく、末端のセンサノードに至る全ての電子機器の低消費電力化が求められる。電子機器の低消費電力化にとって重要となる半導体集積回路（ＬＳＩ）の低消費電力化には、電源電圧の低電圧化が最も効果的である。しかし、低電圧の条件下ではＣＭＯＳ回路の動作が不安定になり、ＬＳＩの製造ばらつきやノイズなどに影響され、動作マージン減少、誤動作などの障害が、現在に比較して極めて増大する。ＬＳＩとして安定動作させるには、ロジックやメモリなど構成回路の極低電圧化はもちろん、電源電圧をきめ細かく制御する電源システム、ＬＳＩチップと外部との各種Ｉ／Ｏインタフェースなど、ＬＳＩでの実用化に向けた様々な回路・システム技術、設計技術が必要である。また、プロセッサコアを中心としたシステムの消費電力削減を図るにはソフトウェアによる電力制御技術も必要である。これらの効率的な開発のためには、デバイスメーカ各社単独で取り組むよりは、共通の課題を抱える企業が協同し、トップレベルの研究ポテンシャルを有する大学と産学連携による総合的な取り組みが必要である。

本プロジェクトは、将来のネットワークシステム等に使われるＬＳＩの低消費電力化に貢献する極低電力回路・システム技術を開発し、我が国の半導体関連産業の国際競争力強化に資すると同時に、地球環境の温暖化抑制に貢献することを目的として、ＩＴイノベーションプログラム・エネルギーイノベーションプログラムの一環として実施する。

当該研究開発事業は、産業界も資金等の負担を行うことにより、市場化に向けた産業界の具体的な取り組みが示されていることを条件として実施する。

（２）研究開発の目標

極低電圧要素回路技術と極低電力無線／チップ間ワイヤレス通信技術を開発し、これら要素技術の主要部分を統合最適化する技術で、ＬＳＩチップの低消費電力化を図る。具体的には、同じ処理を行うための消費エネルギーを従来技術(*1)に比べ１／１０以下に削減することを目標とする(*2)。また、ソフトウェアによる電力制御技術によりプロセッサコアを中心としたシステムの低消費電力化を図る。具体的には同一アプリケーションプログラム実行時の電力消費を１／１０以下に削減することを目標とする。なお、目標の詳細については、別紙の研究開発計画を参照のこと。

(*1) 従来技術とは、以下　平成２０年度末時点での産業界技術レベルを示す。

(*2) 本研究開発の対象は回路、システム、設計技術による消費電力削減であり、新材料、新プロセス、新デバイス構造による消費電力削減効果は含めない。

（３）研究開発の内容

上記目標を達成するために、以下の研究開発項目について、別紙の研究開発計画に基づき研究開発を実施する。

Ⅰ.極低電圧要素回路技術［委託事業］（研究開発期間　平成２１年度から平成２３年度）

低電圧で安定動作するロジック、メモリなどの回路設計技術を確立する。

（研究開発項目①）ロジック回路技術開発:極低電圧ロジック回路の開発

（研究開発項目②）メモリ回路技術開発：オンチップ極低電圧メモリ回路の開発

（研究開発項目③）アナログ回路技術開発：０．５Ｖ動作新方式 PLL 回路等の開発

（研究開発項目④）電源回路技術開発：０．５Ｖで安定動作する新規電源回路の開発

Ⅱ.極低電力ＬＳＩチップ統合最適化技術（研究開発項目⑤）［委託事業］

（研究開発期間　平成２３年度から平成２４年度）

上記Ⅰの要素回路技術の主要部分を統合し、省エネ制御と統合電源システムを組み合わせた極低電力ＬＳＩチップ設計手法を開発する。

本研究開発項目は、実用化まで長期間を要するハイリスクな「基盤的技術」に対して、産学官の複数事業者が互いのノウハウ等を持ちより協調して実施する事業であり、委託事業として実施する。

Ⅲ.低電力無線／チップ間ワイヤレス技術（研究開発項目⑥）［委託事業］

（研究開発期間　平成２１年度から平成２４年度）

従来技術より１桁低電力の低電力無線／チップ間ワイヤレス技術を開発する。

Ⅳ.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（研究開発項目⑦）

［委託事業、共同研究（ＮＥＤＯ負担率：２／３）］

（研究開発期間　平成２２年度から平成２４年度）

ソフトウェアによる周波数・電圧制御等の電力制御でシステムの低消費電力化を図

るためのメニーコア用アーキテクチャの検討、及びコンパイラ技術の研究開発を行う。

本研究開発項目は、（１）実用化まで長期間を要するハイリスクな「基盤的技術」に対して、産学官の複数事業者が互いのノウハウ等を持ちより協調して実施する事業、又は（２）試験・評価方法、基準・プラットフォームの提案等、国民経済的には大きな便益がありながらも、民間企業の研究開発投資に見合うものが見込めない「公共財の研究開発」事業であり、原則、委託事業として実施する。ただし、（１）については上記以外のもの（※1）は、共同研究事業（ＮＥＤＯ負担率：２／３）として実施する。

※１ 民間企業単独、民間企業のみでの連携、大学等の単独等、産学官連携とならないもの。

２．研究開発の実施方式

（１）研究開発の実施体制

本研究開発は、独立行政法人新エネルギー・産業技術総合開発機構（以下、「ＮＥＤＯ」という。）が、単独ないし複数の原則本邦の企業、大学等の研究機関（原則、本邦の企業等で日本国内に研究開発拠点を有していること。なお、国外の企業等（大学、研究機関を含む）の特別の研究開発能力、研究施設等の活用または国際標準獲得の観点から国外企業等との連携が必要な部分を、国外企業等との連携により実施することができる。）から公募によって研究開発実施者を選定後、共同研究契約等を締結する研究体を構築し、委託して実施する。

共同研究開発に参加する各研究開発グループの有する研究開発ポテンシャルの最大限の活用により効率的な研究開発の推進を図る観点から、必要に応じて研究体にはＮＥＤＯが委託先決定後に委嘱する研究開発責任者（プロジェクトリーダ）を置き、その下に研究者を可能な限り結集して効果的な研究開発を実施する。

（２）研究開発の運営管理

研究開発全体の管理・執行に責任と決定権を有するＮＥＤＯは、経済産業省及びプロジェクトリーダと密接な関係を維持しつつ、プログラムの目的及び目標、並びに本研究開発の目的及び目標に照らして適切な運営管理を実施する。また、必要に応じて、外部有識者の意見を運営管理に反映させる。

３．研究開発の実施期間

本研究開発の期間は，平成２１年度から平成２４年度までの４年間とする。

４．評価に関する事項

ＮＥＤＯは、技術的および政策的観点から見た技術開発の意義、目的達成度、成果の技術的意義並びに将来の産業への波及効果等について、外部有識者による研究開発の事後評

価を平成２５年度に実施する。なお、評価の時期については，当該研究開発に係る技術動向、政策動向や当該研究開発の進捗状況などに応じて、前倒しする等、適宜見直すものとする。

５．その他の重要事項

（１）研究開発成果の取り扱い

①成果の普及

得られた研究開発成果のうち、共通基盤技術に係るものについては、プロジェクト内で速やかに共有した後、ＮＥＤＯおよび実施者が協力して普及に努めるものとする。

②産業財産権の帰属

委託研究開発の成果に関わる産業財産権については、「独立行政法人新エネルギー・産業技術総合開発機構　新エネルギー・産業技術業務方法書」第２５条の規定等に基づき、原則として、すべて委託先に帰属させることとする。

（２）基本計画の変更

ＮＥＤＯは、基本計画の内容の妥当性を確保するために、社会・経済的状況、内外の技術開発動向、政策動向、プログラム基本計画の変更、評価結果、研究開発費の確保状況、当該研究開発の進捗状況等を総合的に勘案し、達成目標、実施期間、研究開発体制等、基本計画の見直しを弾力的に行う。

（３）根拠法

本プロジェクトは，独立行政法人新エネルギー・産業技術総合開発機構法第１５条第１項第１号ハに基づき実施する。

６．基本計画の改訂履歴

（１）平成２１年３月、制定。

（２）平成２２年８月、改訂（研究開発項目の追加）

（３）平成２３年３月、改訂（研究開発項目⑤の目標設定）

（別紙） 研究開発計画

研究開発項目① ロジック回路技術（研究開発期間　平成２１年度から平成２３年度）

１．研究開発の必要性

ロジック回路部の消費電力は電源電圧に大きく依存し、電源電圧を極限まで低減した「極低電圧(０．５Ｖ以下)動作のロジック回路技術」が求められている。すでに微細化に伴う素子構造の縮小化によりプロセス起因、或いは使用している物質・材料固有のばらつきが顕在化しているが、極低電圧ではこれらの影響が更に増加する。チップ間やブロック間のシステマティックばらつきの影響を低減するには電源電圧やしきい値電圧の適応制御が有効と知られているが、これまでロジック技術領域ではそれほど問題視されていなかったランダムばらつきへの対策が必要になる。このためには、極低電圧でのロジック回路動作課題を的確に抽出する評価技術と、それを解決する回路技術の開発が強く求められている。

２．研究開発の具体的内容

ロジック技術領域における低電力ＣＭＯＳ　ＬＳＩに適した、極低電圧ＣＭＯＳ回路技術の実現に求められる極低電圧での回路動作課題を抽出する評価技術と、それを解決する回路技術を開発する。

（１）極低電圧での回路動作課題を解決する回路技術の開発

①ベースとなる電源電圧やしきい値電圧の適応制御技術

②ばらつき耐性を実現する細粒度適応制御技術、例えば、ランダムばらつきの影響を低減するパイプラインタイミングの局所適応制御技術や構成要素の冗長化技術等

③極低電圧動作回路技術、耐ノイズ回路技術

（２）極低電圧での回路動作課題を抽出する評価技術の開発

ＴＥＧ試作を通じた、ロジック基本回路の低電圧特性やばらつき耐性に関する評価・解析基盤技術開発

（３）実用製品に適用できるレベルの実証チップの試作

（１）により最先端ＣＭＯＳ技術で実証チップを試作、評価し低電力性能を実証する。

３．達成目標

最終目標として、平成２３年度末までに以下の目標を達成する。

最先端ＣＭＯＳ技術を用いた、ロジック技術領域の極低電力システムＬＳＩを実現する極低電圧ＣＭＯＳ回路の要素技術を開発し、これにより　１６ｂｉｔ加算器あるいは同等以上の機能と規模を持つ回路ＩＰを試作し、エラーレート10^{-10}以下を満たしつつ、本テーマの開発成果を用いていない従来技術との比較で、処理性能を揃えた条件で消費電力が１／１０以下に低減できることを示す。

（別紙） 研究開発計画

研究開発項目②　メモリ回路技術（研究開発期間　平成２１年度から平成２３年度）

１．研究開発の必要性

ロジック技術領域と同様に、メモリ技術領域でも、一層の低電力化のために革新的新技術の導入が必須である。具体的には、不揮発性メモリ技術や３次元実装技術の利用に加えて、極低電力のオンチップメモリが求められている。また、微細化に伴う素子構造の縮小化により、トランジスタのリーク電流が増大して、動作時電力のみでなくオンチップメモリの待機時の電力の低減も重要な課題となっている。オンチップメモリとして重要なＳＲＡＭの技術領域では、メモリ動作がトランジスタのばらつきに敏感なため、低電力、低電圧動作にかかわるブレークスルーが強く求められている。

２．研究開発の具体的内容

低電力メモリの基盤技術として、ロジック回路と同じプロセス、材料で製造でき、コスト面で有利なバルクＣＭＯＳ技術を使った低電力ＳＲＡＭ技術を開発する。動作電源電圧はＳＲＡＭとして、最小動作電力となる電源電圧を本プロジェクトの中で見極めていく。

（１）メモリ周辺回路の制御によりメモリセルの動作マージンを高める技術の開発

①メモリ周辺回路の電圧制御、タイミング制御技術により、メモリセルの動作マージンを向上させつつ低電圧化を図り、低電力化を実現する。

②周辺回路による電圧制御のための高効率オンチップＤＣ－ＤＣコンバータの開発

（２）低電圧動作メモリセルの開発

（１）の検討により、必要に応じて、現在、一般的な６トランジスタセル以外の可能性を検討する。新規メモリセルの開発は、産業界の実用化の可能性を睨みながら判断する。

①低電圧新規メモリセルの開発

②上記メモリセルに対応する最適な周辺回路技術の構築

（３）ＴＥＧ（Test Element Group）試作による実証

（１）、（２）で開発した技術を総合的に組み合わせた上で、ＳＲＡＭを作製し低消費電力性能実証を行う。

３．達成目標

最終目標として、平成２３年度末までに以下の目標を達成する。

新しい回路技術、メモリセル技術を用いた、極小電力を実現する低電圧動作ＳＲＡＭを開発し、これにより　低電力ＳＲＡＭを試作し、本テーマの開発成果を用いていない従来技術との比較で、１Ｍｂｉｔ当たりの消費電力が１／１０以下に低減できることを示す。

(別紙)　研究開発計画

研究開発項目③　アナログ回路技術（研究開発期間　平成２１年度から平成２３年度）

１．研究開発の必要性

システムを集積するＬＳＩ（ＳｏＣ等）において、ロジック、メモリ、アナログ回路の搭載は、必要不可欠なものである。このためロジックとの同電圧で動作するアナログ回路、或いはロジック回路と接続性・親和性を持ったアナログ回路が要求される。

デジタル回路においてはとくに、ロジックと同電位で安定動作するクロック位相調整回路ＰＬＬ（Phase Lock Loop)は不可欠であり、従来技術の延長線上には抜本的な解決策が見あたらない。従来技術とは異なる発想に基づく研究開発に取り組み、実用化開発に向けての指針を提示することが求められている。

２．研究開発の具体的内容

（１）０．５Ｖ動作新方式ＰＬＬ技術の開発

ロジック回路の安定動作を行うためには、クロック信号の安定性能が要求される。クロックのジッタ(*3)特性の劣化はシステム全体にダメージを与え、動作不具合の要因となり得る。またロジックと異電位でのクロック源では、レベルシフタの挿入などによるジッタの悪化が懸念される。これらの問題を解決するために、０．５Ｖロジック回路と共存できるアナログ回路である０．５Ｖ動作低ジッタ性能ＰＬＬを開発する。

①ロジックと同電位０．５Ｖ動作ＰＬＬ回路を開発する。

②低電力で広帯域（１ＭＨｚ～）に適用可能なＰＬＬアーキテクチャーを開発する

（２）その他のアナログ回路（A/Dコンバータ、比較器、増幅器等）

（３）ＴＥＧチップ試作による実証

(*3)ジッタ：時間軸上における信号のゆらぎのこと

３．達成目標

最終目標として、平成23年度末までに以下の目標を達成する。

０．５Ｖ動作新方式ＰＬＬ技術を開発し、これにより、ＴＥＧチップによる低ジッタＰＬＬの実証とロジックも含めた動作実証を行う。

（別紙） 研究開発計画

研究開発項目④　電源回路技術（研究開発期間　平成２１年度から平成２３年度）

１．研究開発の必要性

０．５Ｖ動作でのロジック回路においてもロジックへの電源電圧に誤差が無く、急峻負荷変動においても電圧降下が無い電源が必須である。ロジック或いはシステムＬＳＩの０．５Ｖ安定動作を確保するために、従来のシステム設計を見直し、新たな発想によるロジック回路への電源供給システムを開発する。

２．研究開発の具体的内容

デジタル回路およびメモリ回路に０．５Ｖを安定に供給するには、既存の開発電源回路技術では、低消費電力性能が期待できない。したがってロジック側の動作状態、情報から電力管理を行う協調制御した電源構成等の技術により、安定かつマージンのある低電力電源システムを開発する。

（１）極低電圧ロジック回路を安定動作させる電源システムの開発

①ロジックの消費電力情報をフィードバックする電源システムを開発する。

②最適電圧に可変可能な電源システム開発する。

（２）供給電力として０．５Ｖを想定した低消費電力昇降圧ＤＣ／ＤＣコンバータの開発

①供給電圧０．５Ｖから電源回路内基準電圧発生回路を開発する。

②供給電力から見た最適電力効率となる構成を検討し、高効率な電源システムを開発する。

（３）実証

（１）・（２）で開発した要素回路技術を用いた電源システムの試作により、低消費電力化性能の実証を行う。

３．達成目標

最終目標として、平成23年度末までに以下の目標を達成する。

低電圧システムに適した電源回路、電源システム技術を開発し、これにより、ロジック回路と組み合わせたＬＳＩチップを試作し、高い動作マージンを得られることを示す。

（別紙） 研究開発計画

研究開発項目⑤　極低電力ＬＳＩチップ統合最適化技術

（研究開発期間　平成２３年度から平成２４年度）

１．研究開発の必要性

本プロジェクトで開発される極低電圧要素回路技術を適用し、ＬＳＩチップ全体として統合化するための最適化技術が求められる。

２．研究開発の具体的内容

極低電圧要素回路技術では、低電力化のために極低電圧で動作する要素回路技術を開発する。開発された極低電圧要素回路技術の主要部分を統合し、ＬＳＩチップとして最適な性能を発揮させるための、省エネ制御と統合電源システムを組み合わせた極低電力ＬＳＩチップ統合最適化技術（極低電力ＬＳＩチップ設計手法）を開発する。極低電力ＬＳＩチップ統合最適化技術を適用した、極低電力ＬＳＩチップを試作し、低消費電力化効果を実証する。

（１） 極低電力ＬＳＩチップ統合最適化技術

電源回路やロジック回路、メモリ回路等の要素技術開発との連携を取り、ＬＳＩチップとして最適な性能を導き出すチップアーキテクチャを決定する。

（２） 極低電力ＬＳＩチップを試作し、その性能評価を行う。

（３） 上記（２）に比較してより大規模（１００万トランジスタ程度）で複雑なデータ処理LSIチップを設定し、 必要な設計環境を用いて、その消費電力のシミュレーション又はチップ試作による実証を行う。

３．達成目標

最終目標として、平成２４年度末までに以下の目標を達成する。

（１）（２）により　極低電力ＬＳＩチップの低電力効果(同じ処理を行うための消費電力が従来技術に比べ1／10以下への低減）を実証する。

（３）により　大規模で複雑なデータ処理ＬＳＩチップの大幅な低電力効果を実証し、さらに、大規模化に伴う低消費電力化克服への技術提案を行う。

（別紙） 研究開発計画

研究開発項目⑥　低電力無線回路/チップ間ワイヤレス技術

（研究開発期間　平成２１年度から平成２４年度）

１．研究開発の必要性

将来のネットワーク社会においては、①ＬＳＩチップ間の非接触データ転送技術によるデータ伝送、②センサネット等のユビキタスネットワーク、③アクセスポイント間の高速伝送、高解像動画信号等の無線による高速データ転送の必要性が急速に進むことが予想される。したがって、電力削減のためにはＬＳＩの演算処理部分と同じく、無線技術領域でも革新的低電力技術の導入が必須である。ユビキタス社会の基盤技術として、低電力の無線回路／チップ間ワイヤレス技術の開発が強く求められている。

２．研究開発の具体的内容

低電力インタフェースの基盤技術として、無線距離と伝送速度を等しくして比較した場合、従来技術より一桁省電力の低電力無線／チップ間ワイヤレス技術を開発する。開発した要素回路技術は試作による低消費電力化性能の実証を行う。

用いる技術は汎用性あるいは値段を考えバルクＣＭＯＳ技術を前提とする。電源は０．５Ｖ以下とするが、低電力化のために昇降圧技術を用いることもある。

３．達成目標

最終目標として、平成２４年度末までに以下の目標を達成する。

低電圧ＲＦ　ＣＭＯＳ回路技術を用いた、低電力無線／チップ間ワイヤレス技術を開発し、これにより、ＴＥＧを試作し、５０ｐＪ／ｂｉｔ以下の低消費電力通信技術が実用レベルであることを示す。

（別紙） 研究開発計画

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

（研究開発期間　平成２２年度から平成２４年度）

１．研究開発の必要性

情報通信機器や車載機器等の高度化・設置数の急激な増加に伴い、情報処理量とエネルギー消費量の増大が見込まれ、これらの機器に組み込まれるプロセッサの高性能化・高機能化だけでなく低消費電力化が重要な課題となっている。そこで、多数のプロセッサコアをワンチップに搭載したメニーコア・プロセッサ(*)によって、この課題を解決することが有望と考えられているが、平成21年度のメニーコア・プロセッサ先導研究で得られた知見によれば、高性能・高機能かつ低消費電力のメニーコア・プロセッサの実現には、ソフトウェアによる周波数・電圧等のきめ細かい電力制御を行うことが必要である。コア数が増えるに伴い、人間がアプリケーションプログラムの中に電力制御の仕組みを組み込んでいくことには非常な困難が伴うため、API（コンパイラへの指示）等を用いたコンパイラ技術の開発が必須である。

(*) コア数が32～64以上を指す。

２．研究開発の具体的内容

低消費電力メニーコア用アーキテクチャを検討し、組込み用のメニーコア・プロセッサ及び各種サーバ上でコンパイラの動作を可能とするAPI（コンパイラへの指示）を策定する。そのAPIを用いたコンパイラを開発し、組込み向けアプリケーションプログラムで評価する。評価結果をもとにアーキテクチャとAPIへのフィードバックを行う。

併せて提案する技術を適用するアプリケーションの拡大に向けた検討を行う。

３．達成目標

最終目標として、平成２４年度末までに以下の目標を達成する。

あるべき低消費電力メニーコア用アーキテクチャを提案し、開発するコンパイラ技術を用いて既存技術と比べて電力当たりの処理性能２倍を達成する。かつ、メニーコア・プロセッサ上で組込み向けアプリケーションプログラム実行時の電力消費量を１／１０以下にする。

添付資料３

事前評価関連資料

事前評価書

作成日	平成２２年７月１４日

１．事業名称 （コード番号）	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） （P09003）
２．推進部署名	電子・材料・ナノテクノロジー部
３．事業概要	(1) 概要：半導体集積回路（ＬＳＩ）のさらなる高集積化、高機能化に向けて、材料・プロセス技術とともに半導体技術の車の両輪として重要な設計技術分野における低消費電力化の技術開発が求められている。本プロジェクトは、ＬＳＩにおける消費電力の 1/10 以下への削減を目標とした極低電圧要素回路と統合最適化技術、低電力無線技術の開発により、無線ネットワーク端末やセンサノードなど、将来の「極低電力回路・システム技術」を可能とする。 今回の拡充ではソフトウェアによる電力制御技術によりシステムの低消費電力化を図るためのメニーコア用アーキテクチャの検討およびコンパイラ技術の研究開発を行う。 (2) 事業規模：総事業費　42 億円（未定） 内、拡充分 3.6 億円 (3) 事業期間：平成 21 年度～24 年度（4 年間） 拡充分 平成 22 年度～24 年度（3 年間）

４．評価の検討状況※
（※平成 22 年度に公募を予定しているプロジェクト拡充部分に関する事項を中心に記載）

（１）事業の位置付け・必要性

地球温暖化対策は世界全体で早急に取り組むべき最重要課題であり、経済・社会活動と地球環境との調和を実現することが求められている。このためには、経済・社会活動を支えるあらゆる分野で省エネルギー化を図る画期的な技術革新が必要である。情報通信技術分野では、情報通信機器等に組み込まれるプロセッサの高性能化・高機能化と低消費電力化が重要な課題となっており、この解決には多数のプロセッサコアをワンチップに搭載したメニーコア・プロセッサが有望と考えられている。しかし、高性能・高機能かつ低消費電力のメニーコア・プロセッサを実現するためには、ハードウェアの特性を十分に引き出せるソフトウェアが重要であり、本研究開発による共通基盤技術の必要性は高い。

（２）研究開発目標の妥当性

＜目標＞

・あるべき低消費電力メニーコア用アーキテクチャを検討・提案したうえで、
・既存技術と比べて電力当たりの処理性能 2 倍以上を達成し、かつ
・メニーコア上で組込みアプリケーションプログラム実行時の電力消費を 1/10 以下にする。

＜妥当性＞

設定された技術開発項目は、低消費電力化を実現する上で解決すべき基盤技術で、電力消費を1/10以下にする目標は大きな削減効果を目指す挑戦的な設定値であり、2010年6月11日メニーコア基本計画検討委員会での有識者との議論を踏まえたものであるため、妥当と判断する。

なお、達成目標の設定値については、研究開発実施にあたっての必須の目標値のみを基本計画に設定することで、委託先公募において広く提案を収集し、優れた提案を採択する。したがって、提案者が技術の優位性を示したい場合には、達成目標等を適時追加または改訂することによって対応できるものとする。

また、これら目標設定については今後も委員会ならびに有識者ヒアリング等で聴取した意見を適切に反映させる。

（3）研究開発マネジメント

公募を通じて高い技術を有する民間企業、大学、公的研究機関等による実施体制を構築して産官学連携による委託で実施する。必要に応じて、外部有識者の意見を求め、その結果を踏まえて事業全体の予算配分や計画について見直しを行い、適切な管理運営に努める。さらに別途定められた技術評価に係る指針、および技術評価実施要領に基づき、技術的、および産業技術政策的観点から、研究開発の意義、目標達成度、成果の技術的意義、将来の産業への波及効果等について、外部有識者による事後評価を実施する。

（4）研究開発成果

低消費電力メニーコア・プロセッサの特性を引き出すための共通基盤技術として、理想的なアーキテクチャを提案した上でコンパイラ技術を開発することにより、競争領域となる各社ハードウェアの市場浸透と適用拡大が期待できる。

（5）実用化・事業化の見通し

低消費電力メニーコア・プロセッサは、世界的に注目される技術であり、市場の形成を目前にしている。本研究開発項目は、メニーコア・プロセッサを着実に発展・普及させるための共通基盤技術であり、省エネルギー化に貢献すると共に半導体デバイス産業の発展を支援するものと考えられる。

（6）その他特記事項

特になし。

5．総合評価

本研究開発項目は、メニーコア・プロセッサの特性を引き出すための共通基盤技術としてアーキテクチャの提案とコンパイラ技術の開発を行うものであり、当該技術分野の優れた研究者を核にして専門家が協力・集中して開発する体制が望ましく、NEDO が実施する事業として適切であると判断する。

添付資料４

平　22　　成 24 年度成果報告書

「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 4 月

独立行政法人新エネルギー・産業技術総合開発機構

（委託先名）

国立大学法人九州大学

学校法人立命館 立命館大学

国立大学法人電気通信大学

株式会社フィックスターズ

株式会社トプスシステムズ

内容

要約書

件名：平成２２年度～平成２４年度　成果報告書　「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」

メニーコアの利点は、チップ上に搭載された多数のコアを活用した超並列処理の実現により、高性能化と低消費電力化を両立できる点にある。これを実現するためには、(1)実行対象となるアプリケーションから十分なレベルの並列性を抽出し、かつ、(2)コア数の増加に伴い高い性能を実現できるスケーラビリティを確保することが極めて重要となる。このような課題を解決するため、我々は SMYLE(Scalable ManY-core for Low Energy computing)プロジェクトを遂行した。主な成果は以下の通りである。

汎用メニーコア SoC の開発（SMYLEref、SMYLE OpenCL Compiler、ランタイムライブラリ）：各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可能な汎用メニーコア SoC アーキテクチャ SMYLEref を開発した。また、そのソフトウェア開発／実行環境として、VAM の合成機能を有する SMYLE OpenCL Compiler やランタイムライブラリを開発した。これらに加え、複数の FPGA ボードを用いた 128 コア SMYLEref プロトタイプを開発した。評価を行った結果、全コアを利用し並列プログラムを順次実行する従来方式と比較して約 4 倍の性能向上を実現できることが分かった。これは、動作周波数を 1/4 とし、それに伴い電源電圧を 40%程度削減できれば、動的消費エネルギーを約 1/10 に削減できることを意味する。

ドメイン特化型メニーコア SoC の開発（SMYLEvideo）：ビジュアルコンピューティング（特に動画像認識）向けメニーコア・アーキテクチャを開発した。アーキテクチャとアルゴリズムの協調設計を行い、異なる種類のコアをクラスタ化したヘテロジニアス構成を採用するに至った。また、プロセス間通信に関わる処理時間のオーバーヘッドを隠蔽するゼロオーバヘッド通信方式を考案した。さらに、画像のフレーム単位ではなくブロック単位で処理することにより、外部メモリの使用量を 1/7 に、また、必要となるメモリ帯域を 1/5 程度に削減することに成功した。シミュレーションならびに見積もりに基づく評価を実施した結果、特徴点の検出・特徴量の抽出を行うアルゴリズムである SIFT の実行時に 15 フレーム/秒の性能を実現できることが分かった。また、商用の専用ハードウェアアクセラレータを具備した４コア程度の画像認識プロセッサに対して、最小構成時に性能 1.7 倍、消費電力 1/2 以下、回路規模 1/2 以下という大きな優位性を得た。

ソフトウエア開発環境の構築（CLtrump、PEMAP、BEMAP）： 3 つのソフトウェア開発サポートツールを開発した。CLtrump は逐次 C プログラムから並列化された OpenCL ソースコードを生成する。ここでは、プログラマとのインタラクティブな協調作業により効率の良い並列化を実現する。PEMAP は、アプリケーションをメニーコアへと移植する前に達成可能な性能を推定するツールである。本ツールは、プログラム実行の結果は保証しないが、ターゲットとなるメニーコア環境での実行時間を可能な限り再現するためのコードを自動生成する。評価を行った結果、0.44%のエラーで性能を推定できることが分かった。これらに加え、メニーコア評価用ベンチマーク・プログラム BEMAP を開発し、オープンソースとして公開した。

Title: Ultra Low Power Circuits and System Technology Development (Green IT Project), Research Number 7, Architecture and Compiler for Low Power Many-Core Microprocessors (FY2010-FY2012) Final Report

In designing many-core processors, the most important challenges are (1) ensuring a sufficient level of parallelism and (2) maintaining performance scalability against the increased number of cores. Against this background, we have started a research and development project called SMYLE (Scalable ManY-core for Low Energy computing). The main contributions of this research are as follows.

Development of a Many-core SoC (SMYLEref, SMYLE OpenCL, and Run-time Library) : We have designed a many-core architecture that can be used in general purpose SoCs. The architecture supports efficient execution of VAMs, Virtual Accelerator on Many-core. VAMs are generated through an architecture synthesis technique, and are statically mapped onto the many-core domains. We also have proposed an OpenCL-based programming model and provided a relevant compilation and run-time execution environment. Another contribution of this research is a multi-board FPGA emulation system, which has been developed as an execution platform of 128-core SMYLEref. In our evaluation, it has been observed that the proposed parallel execution methodology can produces 4X speedup compared with a conventional approach. We can achieve 1/10 of energy consumption by means of lowering the clock frequency if we assume the same performance with the conventional execution.

Development of a Domain Specific Many-core SoC (SMYLEvideo) : This research has developed a domain specific many-core SoC. Through an architecture-algorithm co-design for video mining applications we designed a scalable many-core processor, which consists of clustered heterogeneous cores with stream processing capabilities and FIFO-based zero-overhead inter-process communications. For achieving high-performance and low-power consumption, each application is partitioned to exploit both task and data parallelism, and programmed as a distributed stream processing with relatively large local register-file based on Kahn Process Network model. Our evaluation shows that SMYLEvideo can achieve 1.7X performance improvement, less than or equal to 50% of power consumption, and half of the hardware size, compared to a state-of-the-art four-core system integrated with a hardware accelerator.

Support for SW development (CLtrump, PEMAP, and BEMAP): This research focuses on

developing several tools to improve software productivity. A C to OpenCL translator called CLtrump supports user-interactive optimization and parallelization. Another tool called PEMAP has been developed for estimating the sustained many-core performance before porting legacy applications. The tool can be used for early-stage performance estimation. Our experiments on PEMAP show we can estimate performance of hand-parallelized programs in an error of 0.44% of sequential program's performance on average. We have also developed a benchmark set called BEMAP for performance evaluation, which is released under an open source license.

1　研究概要

1.1　研究背景と解決すべき課題

現在、64～128 個以上のプロセッサコア（以降、コアと略す）を搭載したメニーコアが有望視されている。メニーコアの本質は、「低性能ではあるが小面積かつ低消費電力なコアを大多数搭載し、極めて高い並列処理性能を得る」ことにある。しかしながら、依然としてメニーコアは「マルチコアの延長」と捉えられる場合が多く、この本質に基づくメニーコアならではの研究開発の実施が必要である。図 1-1 は、高性能コア 1 個に対するメニーコアの消費電力量（以降、消費エネルギーと記す）の削減率を表す。ここで、メニーコアを構成するコア（以降、小規模コアと呼ぶ）の性能、消費電力、面積は、それぞれ、高性能コアの 1/2、1/4、1/4 と仮定する。また、図中の「F」はプログラム実行において並列化可能な部分の割合（0≤F≤1）を表す。このような条件において、大幅な消費エネルギーの削減を可能にするためには、以下に示す 2 つの課題を克服しなければならない。

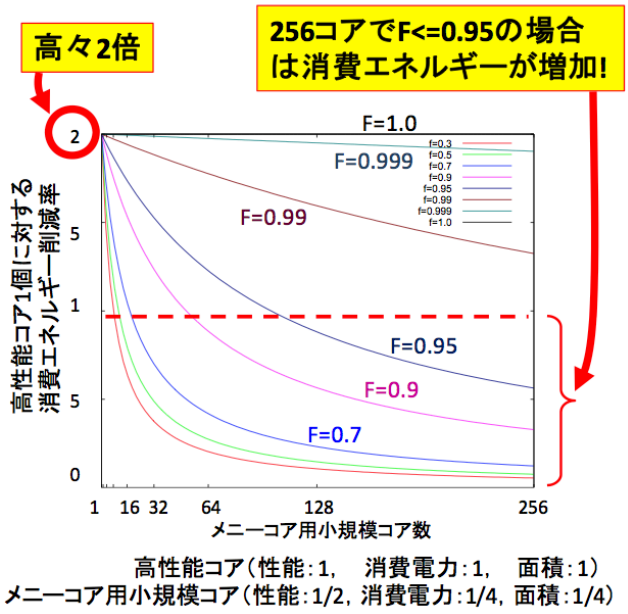


図 1-1 メニーコアの消費エネルギー削減率
（消費電力は面積に比例し、性能は面積のルートに比例すると仮定）

【課題 1】組込みシステム応用を意識した高効率な超並列処理の実現：図 1-1 より、並列化可能な部分の割合 F の低下に伴い、消費エネルギー削減率が急激に悪化することが分かる。これは、小規模コア数の増加に伴う消費電力の増大に対し、性能改善効果が極端に低くなるためである（消費エネルギーは消費電力と実行時間の積）。組込みシステムは様々な製品

に搭載されており、今後も多様化の一途を辿ると考えられる。例えば、動画像圧縮/伸張といったストリーム処理、ロボットや自動車などのマルチタスク処理、インターネット・アプリケーションに代表されるインタラクティブ処理など、その処理形態は応用によって様々である。したがって、このような多種多様な状況においても常に高い並列化効率を達成できるメニーコア実行フレームワークを構築しなければならない。この課題に対する取り組みとして、様々な並列化に関する多くの研究開発が行われてきた。しかしながら、それらはスパコン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である。このような従来のアプローチでは組込みシステムの多様性に対応することは難しく、結果としてメニーコアの応用範囲を限定的にする事態を招きかねない。したがって、並列化された複数タスクの同時実行が極めて重要となり、そのための各種要素技術開発が必要となる。

【課題 2】大幅な動作時消費電力の削減：一般に、メニーコアの消費電力は搭載する小規模コア数に比例する。そのため、図 1-1 において 256 個の小規模コアを搭載した場合、高性能コア 1 個に対して消費電力は 64 倍となる（高性能コアに対する小規模コアの消費電力比は 1/4 のため)。一方、全ての実行を完全に並列化できる理想的な場合、つまりＦが 1.0 の場合には、256 個の小規模コアを用いることで性能は 128 倍向上する（高性能コアに対する小規模コアの性能比は 1/2 のため)。その結果、消費電力と実行時間の積で表される消費エネルギーの削減率は最大でも 2 倍に留まる。この値は、高性能コアに対する小規模コアの性能と消費電力の比のみに支配される。したがって、低消費電力メニーコアの実現には、この消費エネルギー削減率の最大値（つまり、ポテンシャルを十分に引き上げること）が必要不可欠であり、そのためには小規模コア動作時の消費電力を大幅に削減しなければならない。マルチコアやメニーコアの低消費電力化技術として、使用しないコアへの電源供給の停止や、低負荷コアの動作周波数の低減といったアプローチが古くから研究開発されてきた。これは、並列処理において生じる「空間的または時間的に無駄な処理（急がなくて良い処理)」を検出し、小規模コアの稼働率を低くするものである。しかしながら、メニーコアの本質は「大多数の小規模コアをフル活用して高い性能を得る」ことにある。したがって、「使用しない小規模コア」に着眼点を置いた従来アプローチの効果は限定的にならざるを得ず、メニーコアでは「使用する小規模コアの動作時消費電力削減」という抜本的な対策が必要となる。そのためには、低電源電圧化を推し進める必要があるが、その際には性能が大きく低下するといった課題が生じる。したがって、（上記課題 1 と関連するが）電源電圧の低下に耐えうる高い性能を実現することが重要であり、対象アプリケーションがある程度限定できる場合には、実行対象プログラムに特化した最適化を施す等（例えば、ヘテロジニアス構成の採用など）の技術開発が必要となる。

また、組込みシステム開発においては、低消費電力な実行プラットフォームを提供するだけではなく、それと同時に以下の課題を解決しなければならない。

【課題 3】ソフトウェア生産性の向上：現在の組込みシステムではソフトウェア開発コストが爆発的に増大しており、製品開発における最も深刻な問題となっている。ソフトウェア開発者は、機能要件を満足し、かつ、様々な利用シーンにおいても性能や消費電力といった厳しい非機能要件を満たすことが求められる。そのため、ハードウェアの細部を意識したソフトウェア開発を行っているのが現状である。その一方、メニーコアのように大量の計算資源を有するハードウェアを用いる場合には、アルゴリズム・レベルの最適化が性能/消費電力へ極めて大きな影響を与える。したがって、メニーコア向けソフトウェアの開発効率を高めるためには、プログラマはコンパイラと連携した大局的な最適化を実施でき、実効効率を最大化する新しいソフトウェア開発フレームワークが必要となる。これに加え、プログラム移植前の段階において、メニーコアでの実行を想定した場合の性能推定手法の確立とそのツール化も重要課題となる。一般に、既存プログラムにおいて、メニーコア等のアクセラレータへの移植を完全自動で実施することは極めて困難である（または、実現できたとしても高い実行効率を達成することは難しい)。そのため、ソフトウェア移植を開始する前に、メニーコア上での実効性能を精度よく推定することが望まれる。また、ソフトウェア開発コストを削減するためには、設計資産の再利用性の向上や、コミュニティ全体での設計ノウハウの共有が極めて重要となる。これらを実現するためには、主要アプリケーション分野をターゲットとした各種ベンチマーク・プログラムの開発、ならびに、それらに対する最適化技法の適用事例を広く公開し、ある種のエコシステムを構築する必要がある。

1.2　研究内容と成果の概要

本研究開発は、図 1-2 に示す体制で実施した。具体的には、国立大学法人九州大学、学校法人立命館大学、国立大学法人電気通信大学による大学研究者と、ベンチャー企業である株式会社フィックスターズならびに株式会社トプスシステムズの 5 機関が中心となり研究開発を進めた。これに加え、マルチ・メニーコア技術を適用するアプリケーションの拡大に向けた調査、ならびに、ソフトウェア開発環境および動作環境の最新の技術動向の調査を目的として、一般社団法人 電子情報技術産業協会（JEITA)、キャッツ株式会社、イーソル株式会社が九州大学の再委託事業として参加した。また、メニーコア評価環境の構築を加速するため、電気通信大学からの再委託事業として国立大学法人東京農工大学も参加した。

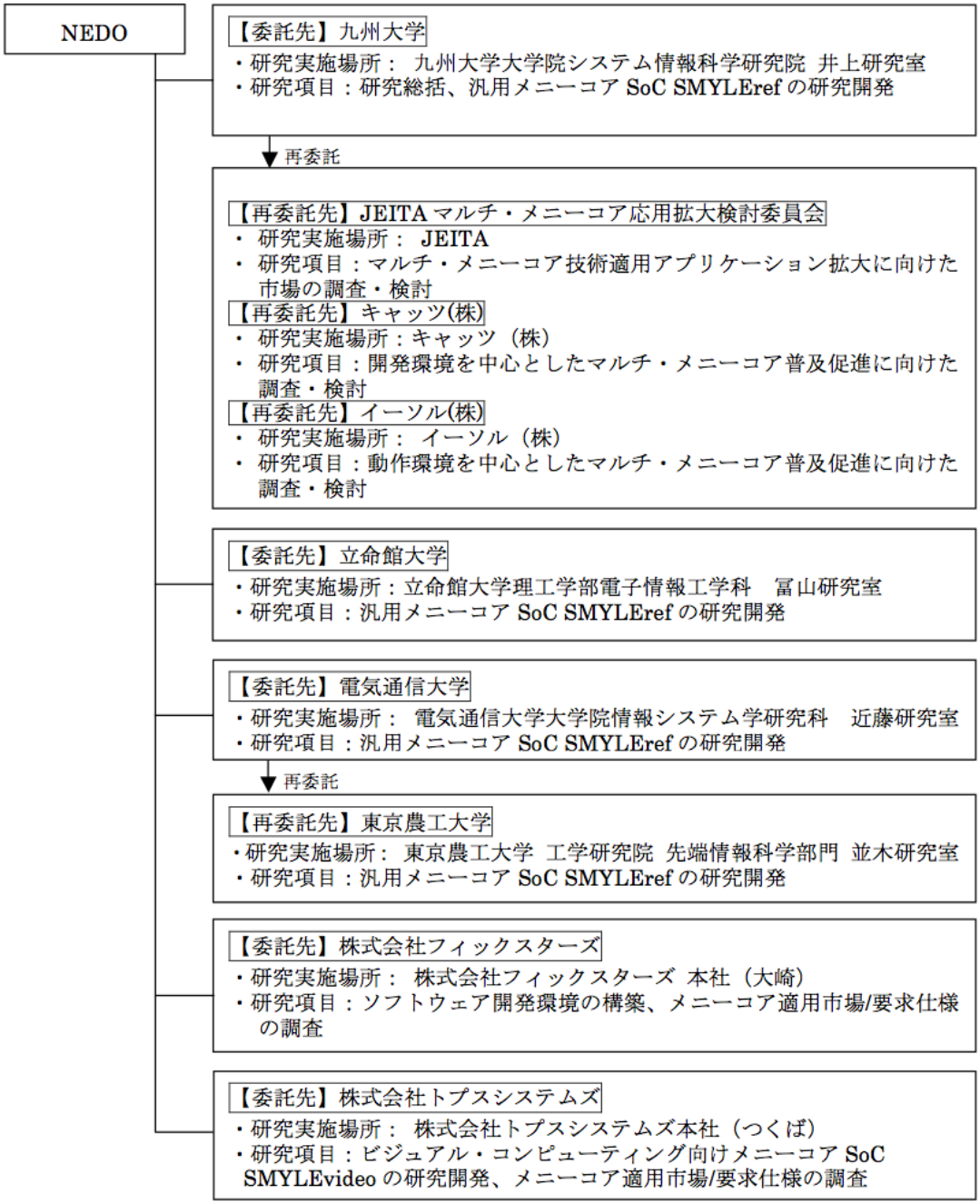


図 1-2 研究体制

本プロジェクトにおいて実施した主な研究開発内容と分担、ならびに、成果の概要を図 1-3 に示す。本研究開発の内容は、主に、1)汎用メニーコア SoC SMYLEref の開発、2)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、3)メニーコア向けソフトウェア開発環境の構築、4)メニーコア市場／技術調査、の 4 つに大別される。各テーマと主な成果の内容は以下の通りである（詳細は本報告書の対応する章を参照）。

<table>
<tr><th colspan="2">汎用メニーコアSoC SMYLErefの開発</th><th colspan="2">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</th></tr>
<tr><td>九州大学

立命館大学

電気通信大学
（＋東京農工大学）</td><td>・クラスタ型ホモジニアス・アーキテクチャ開発
–基本アーキテクチャ開発
–ハードウェアバリア機構の開発（特許申請）
–メニーコア向けDVFS方式の開発
–動的コア割当て方式の開発
–低電圧キャッシュ制御方式の開発
–動的キャッシュ置換制御方式の開発
・組込みメニーコア向けOpenCL実行環境の構築
–静的コア割り当て技術の開発
–応答時間短縮のための実行フレームワーク開発
・評価検証環境の構築と評価
–128コア搭載FPGAプロトタイプ・システムの開発(各種APIやLinuxベースOS含む)
–組込みOpenCL用高速ソフトウェア・エミュレータの開発
–従来の全コア使用による複数プログラム順次実行方式と比較して約4倍の性能向上を確認（動作周波数を1/4にし、低電圧化を進めることで、消費エネルギー1/10の実現可能性を確認）
・実用化への取り組み
–トプスしてテムズへの技術移転（実用化を検討）</td><td>トプス
システムズ</td><td>・動画像認識を対象としたアプリケーション分析
–SIFTのソフトウェア処理におけるボトルネック分析、並列性解析、分散並列処理化
・ビデオマイニング向けヘアーキテクチャの開発
–2クラスタ（10コア）で15fpsでSIFT処理を実現
–スケーラブルなクラスタ型ヘテロジニアス・メニーコア・アーキテクチャの開発
–タスク並列処理＋データ並列処理方式の開発
–ゼロオーバヘッド・コアFIFO通信同期方式の開発（国内及び米国特許申請）
–画像認識系処理向け拡張複合命令の開発
・消費電力評価環境の開発
–コアのパイプライン粒度の電力モデルの開発
–アプリ動作時の消費電力評価環境の開発
・ソフトウェア開発環境の開発
–命令レベル・シミュレータの開発
・動画像認識用ソフトウェア開発
–分散並列処理型のSIFTソフトウェアの開発
・実用化への取り組み
–大規模FPGA(Xilinx V7 FPGA)による評価検証システムの開発
–ハードウェア・エミュレータによる検証環境の構築
–大手ユーザとの商談開始</td></tr>
<tr><th colspan="2">メニーコア向けソフトウェア開発環境の構築</th><th colspan="2">市場ならびに要求仕様の調査</th></tr>
<tr><td rowspan="2">フィックスターズ</td><td rowspan="2">・ユーザ/コンパイラ・インタラクティブコード生成ツールCLtrumpの開発
–ソースコードやプロファイル情報に基づく並列化技術の開発
–C-to-OpenCLソースコード生成技術の開発
–オープンソースとして公開中
・性能推定ツールPEMAPの開発
–異機種を対象とする性能推定方式の開発
–実行可能ダミーコード生成技術の開発
・ベンチマークプログラムセットBEMAPの開発
–主要4分野から8種類のアプリケーションを選定
–C言語のリファレンスコードとOpenCLによる最適化済みコードの開発（最適化に関するドキュメントも含めてオープンソースとして公開中）
・実用化への取り組み
–社内にてPEMAPを用いた性能推定を実施
–BEMAPをサンプルとした有償での組込みサービスのビジネスを展開
–フィックスターズブランドM3に組込みライセンス販売を展開。多数の引き合いあり。契約処理中の案件あり</td><td>トプス
システムズ

フィック
スターズ</td><td>・ユーザー企業へのヒアリング調査
– 関連企業にヒアリングを実施
– メニーコアのニーズが高い具体的な処理の調査
– 10年後も使用可能な画像認識系コンピューティング・プラットフォームの要件を定義
・学会や展示会等におけるブース展示ならびにアンケート調査の実施</td></tr>
<tr><td>JEITA

キャッツ

eSOL</td><td>・メニーコア市場調査
–調査会社による一次調査結果の分析
–セミナー開催とその際のアンケート調査、学会調査
–日本の強み分野である制御系や制御系と情報系の融合分野への注力、普及促進策の提言
・ソフトウェア開発環境ならびに企業における課題の調査
–関連企業のヒアリング等を実施
–並列化のみならず検証環境の重要性が明らかになった
・メニーコア向けオペレーティングシステムに関する調査
–学会や文献調査、関連企業のヒアリング等を実施
–リアルタイム性を考慮したメニーコア向けオペレーティングシステムの開発の重要性、などの知見を得た</td></tr>
</table>

図 1-3 研究内容、分担、ならびに、成果の概要

- **汎用メニーコア SoC SMYLEref の開発（本報告書第 2 章）：**制御を司る少数のコアと多数の同種コアを搭載した SoC プラットフォーム SMYLEref を開発した。メニーコア部分はホモジニアス構成となっており、複数コアで 1 つのクラスタを形成する。各クラスタは 2 次元メッシュの NoC （Network on Chip）で接続される。並列化された複数のアプリケーション（またはタスク）を同時に実行することを意識したアーキテクチャとなっており、その上に実装すべきソフトウェアの開発環境、ならびに、SoC としての実行環境の研究開発も実施した。最終的には、128 コアを搭載した FPGA プロトタイプを

開発した（プロジェクト終了時では単一プログラム実行の動作を確認)。FPGA プロトタイプを用いた基礎評価に基づき複数プログラムの同時実行を想定した性能見積りを行った結果、並列化された単一アプリケーションを順次実行する従来の方式と比較して、同一コア数において約 4 倍程度の性能向上を実現できることが分かった。これは、動作周波数を 1/4 とし、それに伴い電源電圧を 40%程度削減できれば、動的消費エネルギーを約 1/10 に削減できることを意味する。実際、AMD 社のプロセッサチップ Opteron6136 では動作周波数を 1/3 にすることで約 30%の電源電圧低下を可能としており、より低電圧動作を考慮した最適化を施すことで 40%の電源電圧削減は十分に可能であると結論付けられる。なお、本プロジェクトで開発した 128 コア・プロトタイプシステム（OS や各種ドライバといったシステムソフトウェアも含む）や、通常の PC 等で動作する機能検証用シミュレータは公開を前提としており、一部、ラインセンス等の問題が生じる場合を除き無償で利用できる。また、本成果の実用化に関しては、技術移転を実施しトプスシステムにて事業化を開始した。市場成長率と利益率が極めて高い半導体製品である FPGA に対して、従来の FPGA よりも使い易い汎用のアクセラレータとして、マルチ・メニーコア型のプロセッサ（SMYLE）を提供する。FPGA のようにハードウェア設計することなく、所望の機能をソフトウェアのまま搭載して動作させることが可能なため、多くのソフトウェア技術者が使用可能なアクセラレータとして製品化を目指す。

- **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（本報告書第 3 章）:**
  今後の需要拡大が見込まれるビジュアルコンピューティング（特に動画像認識）向けに 1TOPS の処理性能を有するメニーコア・アーキテクチャ SMYLEvideo を開発した。 ビデオ分析、人物検索、ビデオ編集などのアプリケーションを対象に、内在する並列性の詳細分析を行い、アーキテクチャ＆アルゴリズム協調設計により、ビジュアルコンピューティングに適した異なる種類のコアをクラスタ化し集積するヘテロジニアス・メニーコア構成とした。また、プロセス間通信に関わる処理時間のオーバーヘッドを隠蔽するゼロオーバヘッド通信方式を考案した。さらに、画像のフレーム単位ではなくブロック単位で処理することにより、外部メモリの使用量を 1/7、必要となるメモリ帯域を 1/5 程度に削減することに成功した。シミュレーションならびに見積もりに基づく性能評価を実施した結果、特徴点の検出・特徴量の記述を行うアルゴリズムである SIFT の実行時に 15 フレーム/秒の性能を実現できることが分かった。従来型の汎用マルチコア・プロセッサ(4 コア(8 スレッド)約 2.9GHz≒24GOPS,130W)と比較して、性能が約 30 倍（700GOPS 相当)、消費電力 1/300 以下（350mW 程度)、商用の専用ハードウェアアクセラレータを具備した 4 コア程度の画像認識プロセッサに対して、性能 1.7 倍、消費電力 1/2 以下、回路規模 1/2 以下、という大きな優位性がある。SMYLEvideo は、人間の目と大脳の一部（視覚系）に対応する物体認識や意味理解機能を、応用に

合わせて柔軟にかつ低消費電力で実現可能にするメニーコア・プロセッサであり、スマートフォン・タブレット、デジタルカメラ、デジタルテレビ、自動車、監視装置、ロボットなど、次世代の様々なスマートな情報機器への応用が期待される。現在、事業化に向けての実用化開発について、顧客と商談を進めている。

- **ソフトウェア開発環境の構築（本報告書第4章）：**本研究テーマでは、主に3つのソフトウェア開発サポートツールを開発した。まず第1は、前述したSMYLErefでの利用を想定したC-to-OpenCL半自動ソースコード変換ツールCLtrumpの開発である。ソースコードを変換する際、プログラマとコンパイラがインタラクティブに情報をやりとりしながら並列化を進めることが特徴である。本ツールに関してはソースコード・レベルで公開可能である。このように、プログラム開発者から最適化のヒントを得ることで効率のよい並列化を行える。第2は性能推定ツールPEMAPの開発である。先の課題3で言及したように、メニーコアへソフトウェアを移植する前の段階で、移植後の実効性能を見積もる必要がある。本ツールは、プログラム実行の結果は保証しないが、ターゲットとなるメニーコア環境での実行時間を可能な限り再現するためのコードを自動生成する。本ツールはWebを介した実行がサポートされており、移植を試みる設計者は自由に利用できる。また、フィックスターズ社内部においても本ツールを用いた事業展開を実施中である。第3は、メニーコアの評価やプログラム最適化知識の共有を目的としたベンチマーク集（BEMAP）の開発である。今後の需要が高く、かつ、高速処理のためのソフトウェアの最適化が強く要求されるであろう4分野（ライフイノベーション、インターネット・アプリケーション、ファイナンス、ビジュアルコンピューティング）から8種類のコードを選択し、Cのリファレンス・ソースコードならびに各種最適化を施したOpenCLソースコード、最適化による性能向上を評価したレポートを無償公開している（http://sourceforge.net/projects/bemap/）。これらの成果に関しては、メニーコア向けベンチマークとしての普及を期待することができる。また、フィックスターズにおいては、これら公開済みプログラムBEMAPをサンプルとした有償での組込みサービスのビジネスを展開している。さらに、フィックスターズブランドM3に組込み、ライセンス販売も展開中である。

- **マルチ・メニーコア市場ならびに要求仕様の調査（本報告書第5章）：**各種アンケートの実施や展示会での技術説明等により、マルチコアやメニーコアに対する期待、要求、解決すべき課題、市場、などを調査した。これらの結果を本プロジェクトにフィードバックすると同時に、将来に渡って解決すべき課題を探った。また、JEITAよりマルチコア・ハンドブックを出版し、多くの研究開発者への情報提供を行った。さらに、2012年3月と2013年1月の2回にわたり、NEDO主催メニーコア・シンポジウムを開催した。いずれも150名近い参加者であり、プロセッサ開発から応用までの幅広いレンジの技

術者が議論ならびに情報交換する場となり、一つの大きな流れを形成することができた。

2　組込みシステム向け汎用メニーコア SMYLEref の開発

2.1　アーキテクチャ

2.1.1 全体構成

メニーコアにおける最大の課題は、如何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、例えば 128 といった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ない。また、組込みシステムでは、様々な特性（内在する並列性や要求されるリアルタム性）を有する複数アプリケーションを効率良く実行しなければならない。この要求を満足すべく、図 2.1-1 に示すように「仮想アクセラレータ（VAM: Virtual Accelerator on Manycore)」の概念を導入し、メニーコア実行プラットフォームとプログラム実行／開発環境の構築を行った。VAM 導入により、多数の小規模コアを仮想アクセラレータ実現のためのハードウェアプラットフォームとして活用し、一方でコンパイラや並列プログラムはアプリケーション特性に応じて自らが VAM の構成を決定しつつ、それに基づきコード生成や実行を行うことができる。このように、ソフトウェア に対してハードウェアアーキテクチャの決定権を与えることで、高い実行効率を実現する。また、メニーコア上に複数の独立した VAM をマッピングすることで、アプリケーション（またはタスク）レベルの並列性を効果的に活用する。

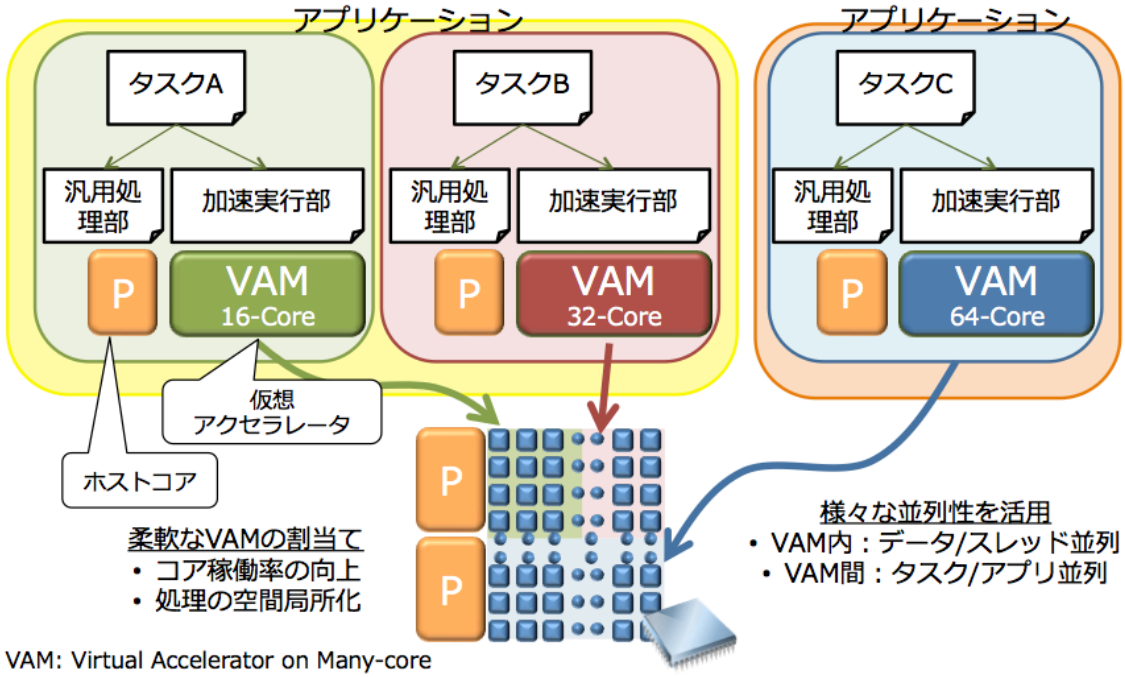


図 2.1-1 メニーコア上で動作する仮想アクセラレータ VAM の概念

2.1.2 コア／クラスタ

本研究では､SMYLEref の各コアとして科学技術振興機構 CREST の研究領域「情報システムの超低消費電力化を目指した技術革新と統合化技術」のプロジェクトである「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」で開発されたプロセッサコア Geyser2 を用いた[関 2010]。Geyser は 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り、実 LSI チップへの実装実績もある。さらに、Geyser を FPGA 上に実装しての Linux 動作も確認されたコアであることから、SMYLEref におけるベースのプロセッサコアとして採用した。なお､Geyser は最粒度パワーゲーティングの研究用途に開発されたプロセッサコアであるが、本研究では最粒度パワーゲーティングの機能は用いない。Geyser コアは 8KB（2 ウェイ・セットアソシアティブ、64B ラインサイズ）の L1 命令キャッシュ、および、L1 データキャッシュを有する。また、命令により制御可能な 16 エントリの TLB を搭載している。なお、データキャッシュはライトバック方式を採用している。

図 2.1-2（A）で示すように、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。そして、複数クラスタがオンチップネットワーク(NoC)で結 合 したアーキテクチャを採る。図 2.1-2（B）のように、各 VAM は 1 個以上のクラスタにマッピングされる。多くのアプリケーションにおいて 8 並列程度までは十分なスケーラビリティが存在することを想定し、単一クラスタ内のコア数は 8 とした。

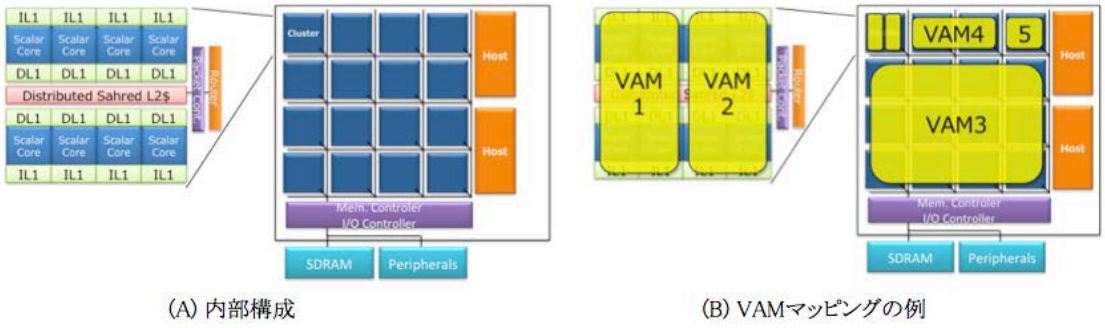


図 2.1-2 クラスタ構成と VAM マッピングの例

各クラスタには分散共有 L2 キャッシュの一部（L2 キャッシュ・スライスと呼ぶ）が搭載される。各 L2 キャッシュ・スライスにはアドレスインターリーブ方式でメモリ空間が割り当てられている。したがって、物理的には分散した配置となっているが、論理的には共有した 1 個の L2 キャッシュとなる。したがって、あるコアが L2 キャッシュにアクセスする際、そのアドレスによって（つまり、クラスタ内とクラスタ外のいずれの L2 キャッシュ・スライスにアクセスするかによって）L2 キャッシュ・アクセスレイテンシが異なる。

L2 キャッシュは共有方式を採るため、コヒーレンシ制御は不要となる。また、L1 キャッ

シュにおいてもハードウェアでのコヒーレンス制御は保証しない。これは、SMYLEref では OpenCL のプログラミングモデルを採用しており、基本的に OpenCL ではキャッシュの一貫性はプログラマが保証する前提となっているためである。

2.1.3 ネットワークオンチップ

クラスタには、コアや L2 キャッシュを備えるコア・クラスタと、チップ外部とのインタフェースの役割を持つペリフェラル・クラスタの 2 種類がある。コア・クラスタ間、およびコア・クラスタ／ペリフェラル・クラスタ間はルータを通してネットワークオンチップ(NoC)により接続され、パケットスイッチング方式でデータ転送が行われる。複数コアでクラスタを構築し、さらにクラスタ間をネットワークで接続するという階層的な構造をとることにより、通信を自クラスタのキャッシュにできる限り閉じ込めることで、通信路の混雑による性能低下の回避を狙う。クラスタ間を接続するオンチップネットワークトポロジは、スケーラビリティの確保と拡張性を考慮して 2 次元メッシュを採用している。この際、デッドロックを防止するために XY ルーティングプロトコルを用い、次元順ワームホールルーティングを行う。SMYLEref で採用するルータアーキテクチャの概要を図 2.1-3 に示す。隣接ノードと Cluster バスとの通信のために 5 本の入出力ポート(N、S、E, W, L)を持つ。なお、データのビット幅は 32 ビットとした。

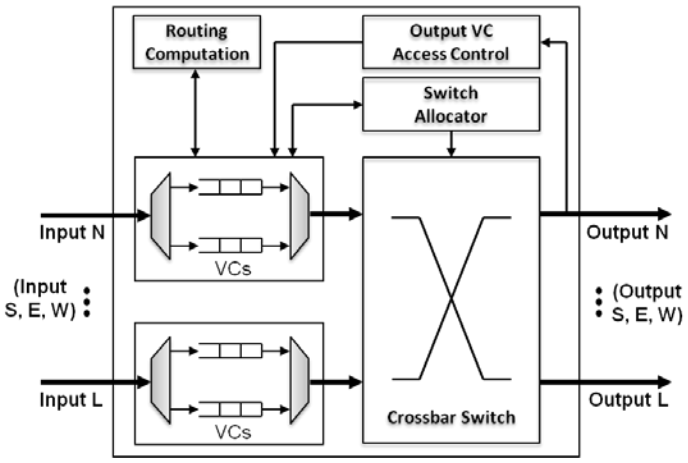


図 2.1-3 ルータアーキテクチャの概要

本ルータは仮想チャネル(VC)ルータアーキテクチャをベースとし、低コスト・高性能なデータ転送を行うことができる On-the-fly ルータ[Nguyen2011]を利用する。通常、ルータは RC (Routing Computation)、VA (Virtual Channel Allocation)、SA (Switch Allocation)、ST (Switch Traversal)の 4 段階の処理を行い、パケットを次のホップ先に転送する。On-the-fly ルータは、SA の確定をしながら VA を行うことで、ST ステージをクリティカルパスより除去し、さらにルックアヘッド・ルーティング技術を利用して RC ステージと SA

ステージを並列に行う。これにより、全ての処理を１サイクルで行うことが可能となり、各ルータで通信パケットを単一サイクルで転送することが可能になる。なお、本ルータは各ポートに 2 個の VC を持つ。

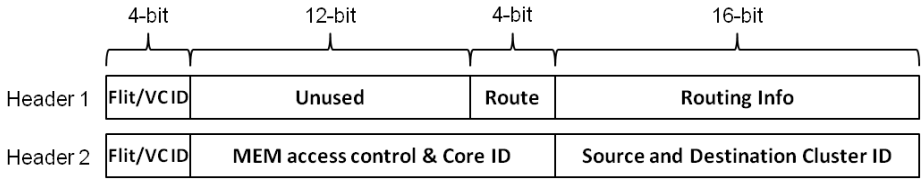


図 2.1-4 ヘッダーフリットのフォーマット

1 つの通信パケットは複数のフリットにより構成される。フリットは head flit、body flit、tail flit の 3 タイプに分類される。head flit はパケットの先頭を示し、body flit はパケットの中間を、tail flit はパケットの最終であることを示す。各パケットは 2 フリットからなるヘッダーフリットを持つ。図 2.1-4 はヘッダーフリットのフォーマットである。Flit/VCID フィールドはフリットの種類(head, body, tail)と仮想チャネル ID を識別するものである。また、Route フィールドはルックアヘッド・ルーティングにより 1 ホップ前のルータで計算されたルーティング情報が格納される。その他、ヘッダーには X 次元と Y 次元のルーティング情報(Routing Info)、メモリアクセスの属性やバス制御情報(MEM access control & Core ID)、発信元クラスタ ID と宛先クラスタ ID(Source/Destination Cluster ID)の情報を持つ。

2.1.4 ハードウェアバリア技術

2.1.4.1 概要

近年のプロセス技術やハードウェア技術、ソフトウェア技術等の進歩により、チップ上に複数のプロセッサコアを搭載したマルチコアプロセッサが広く普及するに至った。また、より多くのプロセッサコアを搭載した、メニーコアプロセッサについても研究が及んでおり、例えば 100 個のプロセッサコアを搭載した商用化チップも存在する[Ramey2011]。メニーコアプロセッサにおいて高い実効性能を実現するためには、搭載したコアの数に見合っただけの並列性を活用できなければならない。例えば、画像処理や科学技術計算のように、極めて高い並列性が内在する場合においてはコアの稼働率を高めることができる。しかしながら、十分な並列性が抽出できない場合が多く存在するのも事実である。したがって、メニーコアプロセッサの応用範囲を拡大しその普及を図るためには、プログラム内並列性のみならず、プログラム間並列性をも積極的に活用しなければならない。すなわち、メニーコアプロセッサにおいては、効率的なマルチスレッド・マルチプログラム実行環境の実

現が極めて重要となる。具体的には、OS、メール、ゲーム、コンパイラなどの複数ソフトウェアが同時に動作する組込みシステムや、複数ユーザが異なるジョブを投入するデータセンター・サーバ応用が考えられる。

並列化された複数のプログラムを同時に実行する場合の課題として、プロセッサコア間での同期処理の高効率化が挙げられる。同期はプロセッサ間の歩調を合わせるために必要であり、例えば、複数プロセッサコアがメモリやバスなどの共通資源を使用する際にアクセス順序を決めるときや、複数プロセッサコアが同時に動作を開始したい場合などに実行される。ここで、ある同期に参加するプロセッサコアの集合をコアグループと呼ぶ。一般に、並列化プログラムの実行に使用するプロセッサコアの数は、当該プログラムに内在する並列性の度合いに大きく依存する。これに加え、複数のプログラムが同時に実行される状況においては、複数のコアグループそれぞれが同期処理を発行する場合もある。したがって、メニーコアプロセッサにおいては、複数の様々なコアグループに対して独立かつ並列に実行可能な同期機構の実現が必要となる。

同期手法の代表的なものの一つとしてバリア同期がある。これは、同期の対象となる各プロセッサコアが実行するプログラム中に、対象コア全てが許可を出すまで停止する指示を埋め込むことにより同期を実現する手法であり、実際に MPI、OpenMP、pthread 等の多くの並列コンピューティング環境に実装されている。一般に、バリア同期は専用ハードウェアの追加なしに実現可能ではあるが、数多くのプロセッサコアが参加する場合には、(1) バリア同期に要する時間が大きくなり頻繁に使用するとソフトウェア実行性能の低下要因となる、(2) 各プロセッサコアのバリア処理完了時刻に大きくばらつきが発生して、事前のチューニングが期待通りに反映されなくなる、などの問題が発生する。そのため、HPC 分野で使用されるような大規模システムにおいてはハードウェアバリア機構が実装されていることが多い。大規模システムでのハードウェアバリア機構は、筐体間やチップ間を繋ぐ長距離ネットワークを制御する多機能なネットワークコントローラの存在を前提として設計されているため回路規模が大きく、そのままメニーコアプロセッサのハードウェアバリアとして使用することは難しい。また、大規模システムでは配線コスト低減を目的としてハードウェアバリアで使用するネットワークを他の通信機能と共有しているため、常に最短レイテンシでのバリア同期実行は期待できない。

そこで、本研究では、メニーコアプロセッサでの利用を前提とし、柔軟性を有する新しいハードウェアバリア方式を開発した。具体的には、(1) 低レイテンシでばらつきが小さく、(2) バリアネットワークの分割を可能にし、かつ、(3) 小規模な回路量で実現できるハードウェアバリア機構である。また、この小規模なバリアネットワークを複数用意することで、多様なプロセスが動作するであろうメニーコアプロセッサにおいても対応可能な柔軟なハードウェアバリア機構を実現する。

2.1.4.2　　柔軟性を有するハードウェアバリア機構

本節では、ある固定的なコアグループを対象とし、提案するハードウェアバリア機構（以下、バリア機構と略す）の基本構造と動作を説明する。なお、様々なプロセッサコア（以下、コアと略す）で構成されるコアグループへの対応、ならびに、複数コアグループでの並列同期処理の実現に関しては次節以降で詳細を述べる。本バリア機構の実現には以下のハードウェア要素が必要となる。

- バリアネットワーク：バリア専用の双方向二分木ネットワーク。4 コアでコアグループを構成した場合の例を図 2.1-5 に示す。
- 葉ノード：各コアに追加されるハードウェア・モジュール。以下に示す 2 種類の 1 ビット・レジスタを有する。
  - 同期設定レジス：当該コアがバリア同期の完了待ち状態にあるか否かを指示するレジスタ。当該コアのプログラム実行が同期ポイントに到達したら'1'にセットされる。また、次に説明する同期状態表示レジスタがリセットされると'0'にリセットされる。
  - 同期状態表示レジスタ：根ノードが管理するバリア機構状態レジスタの複製を保持するレジスタ。
- 根ノード：コアグループにおける同期処理実行状態を管理するハードウェア・モジュール。子ノードとなる全ての葉ノードが管理対象コアグループとなる（ただし、次節で説明するマスク機能を用いた場合を除く）。図 2.1-5 の例では、根ノードが管理するコアグループは葉ノード 0〜3 である。管理下にある葉ノードから同期設定レジスタ値を集計し、全てのコアが同期ポイントに到達したか否かを判定する。内部には 1 ビットの同期状態レジスタを有する。提案するバリア機構には以下に示す 2 つの状態があり、同期状態レジスタにより現状態を表す。状態遷移を図 2.1-6 に示す。
  - バリア完了待ち／受付可能状態：コアグループに属するコアがある同期処理を実行中、もしくは、次のバリア同期処理を実行可能な状態（図 2.1-6 の W_SET）。このとき、同期状態レジスタは'0'にリセットされる。
  - バリア完了通知待ち状態：バリア同期が完了し、全葉ノードに対して完了通知をブロードキャストしている状態（図 2.1-6 の W_RST）。このとき、同期状態レジスタは'1'にセットされる。
- 内部ノード：根ノードと葉ノードの中間に位置するハードウェア・モジュール。内部構造は根ノードと同じであり、バリアネットワークを分割して複数コアグループの同期処理を実現する場合には根ノードとして動作する。

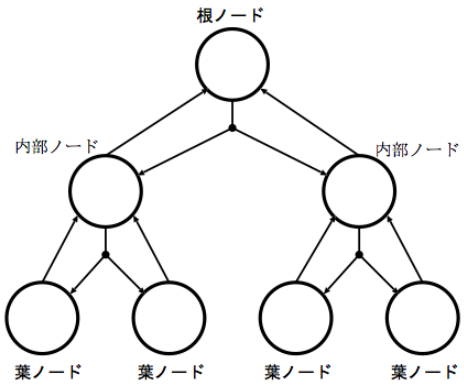


図 2.1-5 4 コアにおけるハードウェアバリア・ネットワーク

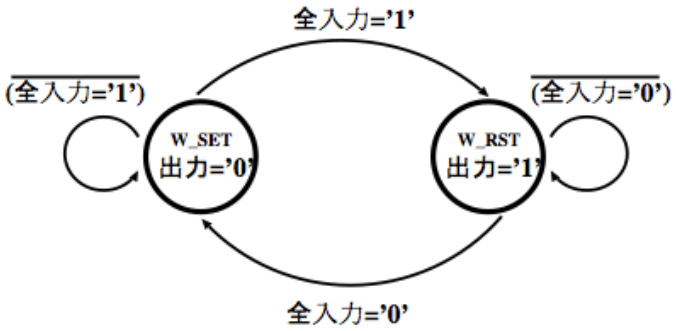


図 2.1-6 バリア同期機構の状態遷移図

以下、図 2.1-7 を用いてバリア機構の動作を説明する。なお、バリア機構の状態は「バリア完了待ち／受付可能状態」であり、全ての葉ノードの同期設定レジスタの初期値は'0'とする（つまり、図 2.1-7（a）に示すように次のバリアを実行可能な状態とする）。

1. 同期処理において、コア 2（葉ノード 2）が同期ポイントに到達する。この時、当該コアの同期設定レジスタが'1'に設定される。
2. コア 2 の同期設定レジスタの値が内部ノード 1 に送信される。この時、他方の子ノードの同期設定レジスタの値がセットされていなければ、同期待ち状態を継続する（図 2.1-7（b））。
3. コア 3 が同期ポイントに到達し、同期設定レジスタが'1'に設定される。これにより、内部ノード 1 は親ノード（ここでは根ノード）に対して同期設定レジスタの値を伝搬させる（図 2.1-7（c））。
4. コア 0 ならびにコア 1 が同期ポイントに到達すると、根ノードは全てのコアが同期ポイントに到達したと判定する。これにより、各葉ノードに対して同期処理完了通知の信号を送り、バリア完了通知待ち状態に遷移する（図 2.1-7（d））。

5. 各葉ノードは、根ノードからバリア完了通知を受け取った後、同期設定レジスタを’0’にリセットし、これが根ノードへと伝搬する（図 2.1-7（e））。
6. 根ノードは、全ての葉ノードの同期設定レジスタの値がリセットされた事を判定し、バリア機構の状態をバリア完了待ち／受付可能状態にする。これにより、次の新たなバリア同期を受け付けることが可能となる（図 2.1-7（a））。

内部ノードは、バリア完了待ち／受付可能状態の場合には、2 つの子ノードから入力した同期設定レジスタの値の論理積をとる。一方、バリア完了通知待ち状態の場合には、これらの値の論理和を親ノードに回送する。これらの値が根に向かって伝搬することにより、根ノードは全ての葉ノードの状況を把握することができる（図 2.1-8）。

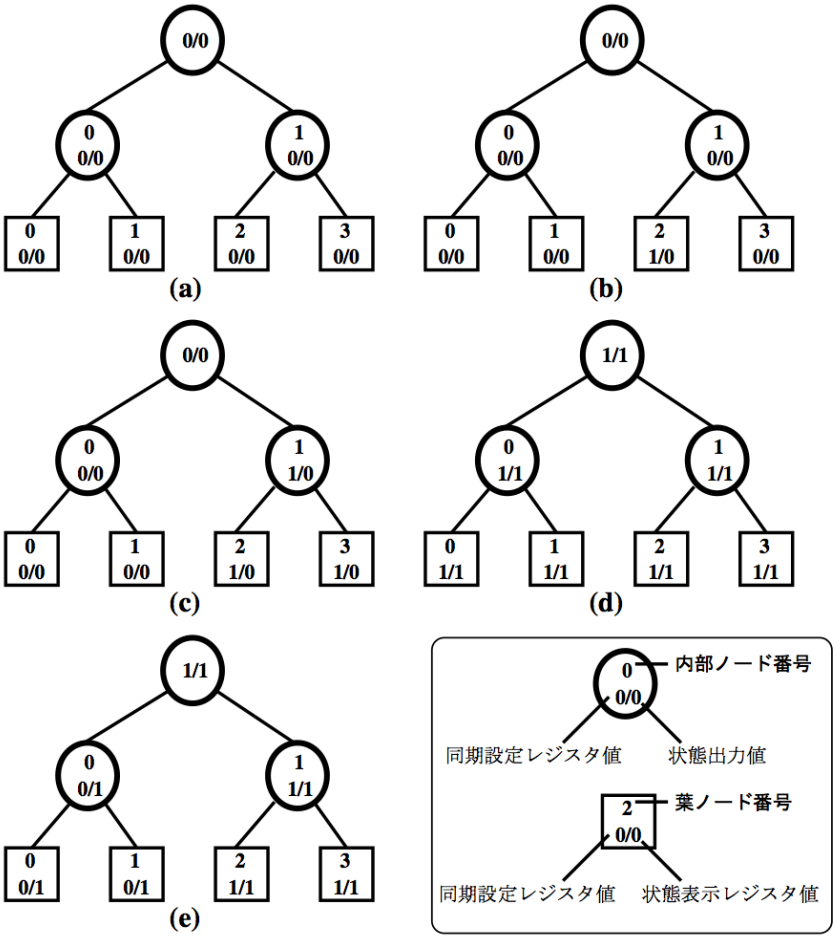


図 2.1-7 バリア同期機構の動作

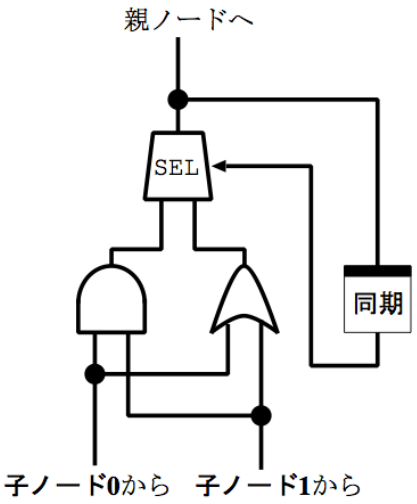


図 2.1-8 内部ノードの同期処理

2.1.4.3　柔軟性を有するハードウェアバリア処理の実現

前節では、根ノードに対応する全ての葉ノードがコアグループに含まれる場合を想定していた。しかしながら、マルチプログラム実行環境下においては、複数の様々なコアグループに対して独立かつ並列に実行可能な同期処理をサポートしなければならない。そこで提案方式においては、接続設定情報の変更によりバリアネットワークを論理的に分割し、一つのバリアネットワーク内で複数のバリア同期を実行できるように拡張する。具体的には、以下に示す2つの機能により、柔軟なハードウェアバリアが可能となる。

- 内部ノードにおける折返し機能のサポート：内部ノードは根ノードと同じハードウェア構成を採る。したがって、内部ノードは根ノードと同一の機能を有することが可能である。そこで、ある内部ノードを根ノードとして動作させることでバリアネットワークを複数の部分木に分割する。図2.1-9は、8個の葉ノードを有するバリアネットワークを4分割した例である。例えば、葉ノード{2, 3}はコアグループBを構成している。この場合、親ノードとなる内部ノードに対して折返し機能を適用することで根ノードとして動作させる。内部ノードに対して折返し機能が適用されると、図2.1-10に示すように、子ノードから回送された信号（同期設定レジスタの値）の論理演算結果を親ノードには回送せず、同期処理完了通知信号として子ノードに転送する。例えば、両方の子ノードが同期ポイントに到達した場合には対応する同期設定レジスタが‘1’にセットされ、これらの論理積がとられる。この結果が、当該子ノードに対して同期処理完了通知信号としてフィードバックされる。また、親ノードに対しては親ノードから回送されてきた同期処理完了通知信号の反転値を転送することで、上位（根ノードに近い方）のノードに対する影響を排除する。このように部分木を用いて小規模なバリアネットワークを複数構成することにより、各コアグループは独立かつ並列にバリア同期処理を行うことができるようになる。
- 葉ノードの切り離し（マスク）機能のサポート：葉ノードに対して適用される設定であり、葉ノードに対応するコアがバリアに参加するか否かを指定する。バリア機構内部では、マスクが有効であれば葉ノードの同期設定レジスタの値が親ノードに送信される。一方、無効な場合には、状態表示レジスタの反転値を親ノードに対して出力することで、上位のノードに対する影響を排除する（図2.1-11）。

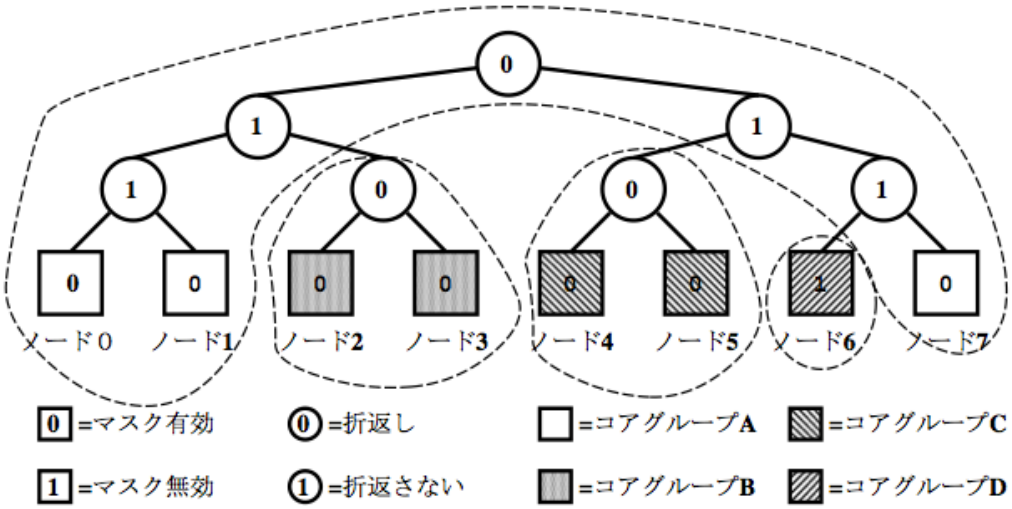


図 2.1-9 折り返し/マスクを用いたグループ分割

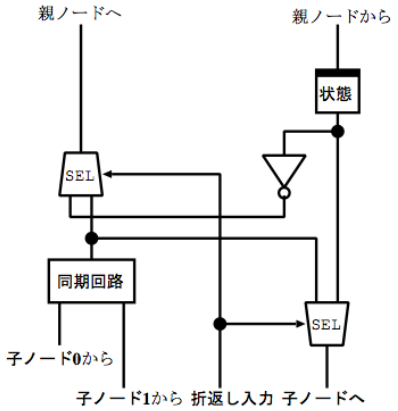


図 2.1-10 折り返し機能

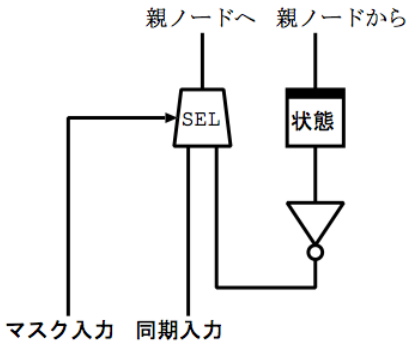


図 2.1-11 マスク機能

2.1.4.4　　多重化による柔軟性の拡大

マスク・折返しによる設定では部分木単位のグループ構成以外は行えないため、それ以外のグループ構成には、バリアネットワークを複数用意すること（多重化）により対応する。図 2.1-12 は、8 ノードからなるバリアネットワークを 2 つを使用して、ノード番号{0, 2, 4, 6}、{1, 3, 5, 7}で組み分けされた、2 つのグループに分割した例となっている。実際には、前節で説明した折返し機能、マスク機能、ならびに、多重化を組み合わせることにより、本バリア機構は少レイテンシでばらつきのないハードウェアバリアを、様々なノードの組み合わせで実行することを可能にしている。

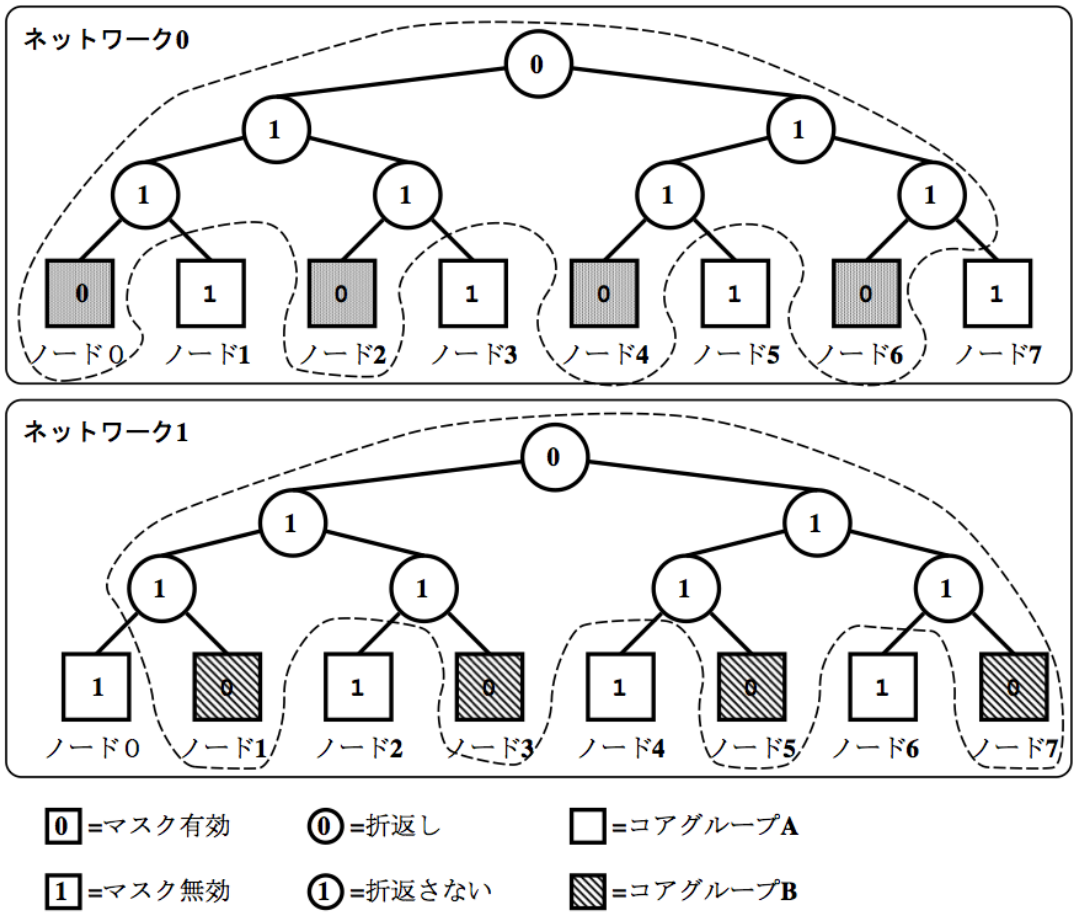


図 2.1-12 多重化

2.1.4.5　　評価

本節では、バリア機構の評価をバリア同期時間の測定および回路規模の計数によって行う。

本評価では、メニーコアアーキテクチャ評価環境である SMYLEref の RTL 設計を使用した。SMYLEref は、MIPS R3000 アーキテクチャをベースとした複数のプロセッサコアをバス結合でクラスタ構成し、多数のクラスタを 2 次元メッシュのオンチップネットワークで結合したアーキテクチャとなっている。本実験では 1 クラスタ、16 コアのモデルを作成し、16 個のコアに対して 8 つのバリアネットワークを実装した。また、バリア制御のために各コア内にレジスタを実装し、各コアがレジスタを読み書きすることによって、バリア同期を実施可能にした。レジスタは、自コアが使用する葉ノード及び、15 ある内部ノードと根ノードの内一つの構成および同期を制御する（図 2.1-13)。

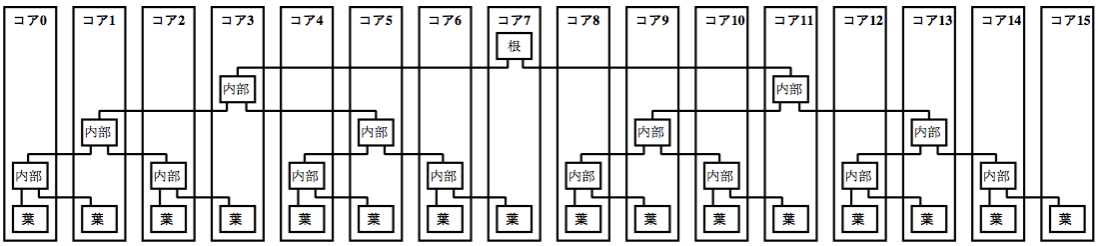


図 2.1-13 ノード構成の変更を行うコアの割り当て

バリア同期時間を測定するため、図 2.1-14 に示す評価用プログラムを用いた。本評価用プログラムは、最初に各コアが担当する、葉ノード、内部ノードのマスク情報および、折返し情報をレジスタに書込む（barrier_config 関数)。次にバリア同期の実行（barrier 関数）を 2 回繰り返す。バリア関数を 2 回実行するのは、関数のプログラムをキャッシュに載せることと、全てのコアが同時に計測対象のバリア関数実行を開始するためである。

```
void main(void)
{
 barrier_config();
 barrier();
 barrier();
 exit;
}

void barrier(void)
{
 check_barrier_reset();
 barrier_sync_set();
 check_barrier_set();
 barrier_sync_reset();
 return;
}
```

図 2.1-14 評価用プログラム

barrier 関数内では以下の処理を行う。

1. check_barrier_reset:状態表示が'0'になるまでレジスタの読込を繰り返す。
2. barrier_sync_set:同期設定='1'をレジスタに書込む。
3. check_barrier_set:状態表示が'1'になるまでレジスタの読込を繰り返す。
4. barrier_sync_reset:同期設定='0'をレジスタに書込む。

バリア同期時間としては、2回目のbarrier関数実行に要したクロックサイクル数計測した。計測は、xilinx 社製 RTL シミュレータ ISIM (0.40d 版) を使用して行った。また、比較ためにハードウェアバリアを使用しないテストプログラム 2 種に対しても計測を行った。これらのテストでは、2 回目のバリア関数実行前にハードウェアバリア関数を 2 回実行して、全コアが同時にバリア関数を実行できるようにしている。2 種のプログラムはそれぞれ、バリア同期の手法[Wilkinson1999]が異なっており、一方は小ノード数向けの master-slave 方式、もう一方は中以上のノード数向けの butterfly 方式を用いている。16 コア全てがバリア同期に参加した時の計測結果を表 2.1-1 に示す。

表 2.1-1 バリア同期時間測定結果

種別	開始最速	開始最遅	完了最速	完了最遅	完了-完了
HW	0	8	55	63	8
MS	0	8	4,837	5,471	634
BUT	0	8	3,891	4,131	240

単位:クロックサイクル

表中の'種別'は測定対象の種別で、'HW'はハードウェアバリア、'MS'はMaster-Slave 方式、'BUT'は butterfly 方式を示す。'開始最速'、'開始最遅'、'完了最速'、'完了最遅'はそれぞれ開始最速を起点とした相対経過クロックサイクル数を示す (図 2.1-15)。'開始最速'は barrier 関数の実行開始が最も早かったコアの相対経過クロックサイクル数であり、表中ではここを起点とするため、全ての種別において値は 0 となる。'開始最遅'は、barrier関数の実行開始が最も遅かったコアの相対経過クロック数を示し、'完了最速'は、barrier関数の完了が最も速かったコア、完了最遅は最も遅かったコアの相対経過クロック数を表す。また、'完了-完了'は完了最速と完了最遅のクロック差を示す。'完了最遅'の値を関数実行レイテンシ、'完了-完了'をコア間のばらつきと考えると、HW は、MS に対してレイテンシを 1/87 に、ばらつきを 1/79 に、BUT に対してはレイテンシを 1/66 に、ばらつきを 1/30 に改善していることが分かる。

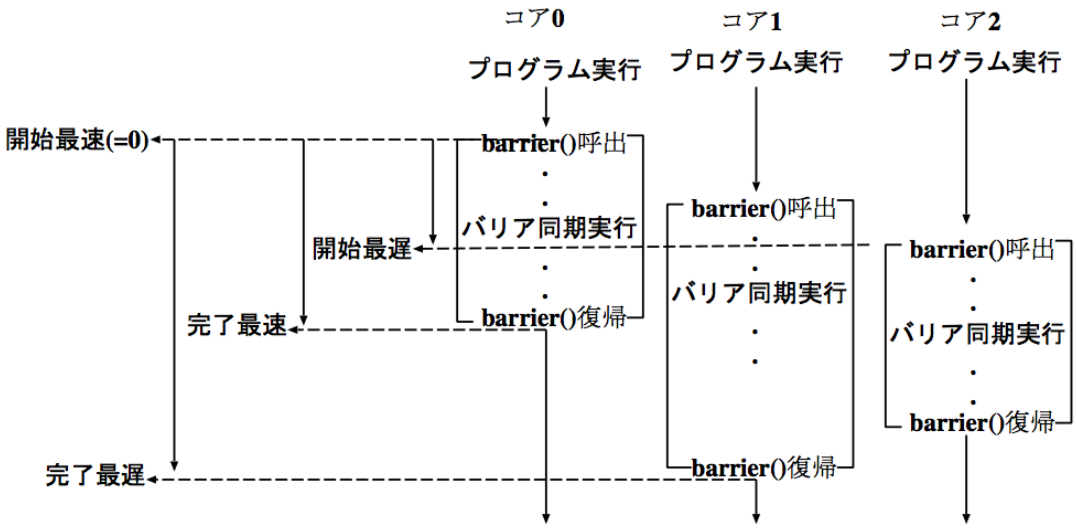


図 2.1-15 測定対象

表 2.1-2 回路規模

Type	BAR	SMYLE	BAR/SMYLE(%)
LUT	2537	192877	1.32
Reg	2361	130289	1.81
RAM	0	117	0

回路規模に関しては、Xilinx 社製の論理合成ツール xst（ISE 13.1-0.40 版）を使用して求めた。合成対象 FPGA には Vertex-6（ML605 評価ボード）を指定している。表 2.1-2 にバリア同期時間測定に使用した SMYLEref モデルの合成結果、及び、そこからハードウェアバリア部分のみを取り出した合成結果を示す。表中の'Type'は計数された対象で、'LUT'は LUT、'Reg'はレジスタ、'RAM'は 36Kbit RAM(ブロック RAM)を示す。また、'BAR'はバリア同期部分、'SMYLE'は SMYLEref の合成結果を表す。'BAR/SMYLE'は LUT、Reg 及び RAM それぞれについて SMYLE に対する BAR の百分率である。BAR/SMYLE はコアの増加によっては変わらないが、クラスタが増加すれば減少する。コア当たり 32KByte 程度のメモリに必要な面積のことも鑑みると、SMYREref メニーコアプロセッサに対してバリアのハードウェア回路は十分に小規模であると考えられる。

2.1.5 動作周波数／コア数スケーリング技術

2.1.5.1 概要

シングルコア・プロセッサの性能向上は、動作周波数の上昇や複雑なアウトオブオーダ実行型のスーパースカラ・アーキテクチャを実装することで達成してきた。しかしながら、消費電力の問題によりその性能向上は限界を迎えている。そのため近年のプロセッサでは、1 つのチップに複数のプロセッサ・コア（以下、コアと略称）を搭載したマルチコア・プロセッサ（以下、マルチコアと略称）が主流である。また、数十のコアを搭載したメニーコア・プロセッサ（以下、メニーコアと呼称）が登場しており、微細化技術の発展により 1 つのチップに搭載されるコアの数は今後更に増加すると予想される[Ramey2011][Seiler2008][Howard2010]。マルチコアの高性能化と低消費電力化を達成するには効率的な並列処理が重要であり、メニーコアにおいてはその重要性がさらに増す。大量のコンピュータを扱う大規模なデータセンタやスーパーコンピュータ等の高性能計算機では消費電力削減に対する要求が高まっており、厳しい消費電力制約下において如何にプロセッサの性能を最大化するかが課題となる。これに加え、今後のメニーコアでは様々な特性や要件を持つアプリケーションを扱うことが予想されるため、それらに応じてエネルギー効率の良い実行を実現できる環境が求められる。

消費電力制約下においてエネルギー効率の良いアプリケーション実行を実現するための既存手法として、Dynamic Voltage and Frequency Scaling (DVFS) が挙げられる[Semeraro2002]。DVFS によりプロセッサの電力効率を向上できるため、マルチコアを対象とした様々な DVFS 適用技術に関する研究がなされている[Herbert2007]。メニーコアにおいても、消費電力バジェットに応じて適したコアの動作周波数と供給電圧を選択することで電力効率の向上が期待できる。しかしながら、そのような場合には、性能を決定する要因として動作周波数だけでなく並列性も同時に考慮しなければならない。なぜなら、チップ上の全コアを用いた並列処理が最高性能やエネルギー効率の良い実行を実現できるとは限らないためである。

本節では、消費電力制約下においてメニーコアにおける並列プログラム実行の性能を最大化する Dynamic Core-count and Frequency Scaling (DCFS) 手法を説明する。本手法では、並列プログラムを実行する際にコア数と動作周波数を動的に制御する。具体的には、限られた消費電力バジェットをプログラムの特性に応じてコア数の増加と動作周波数の上昇に適切に配分することで性能向上を狙う。コア数の増加や動作周波数上昇に対するスケーラビリティはプログラムの種類やその実行箇所に応じて異なるため、プログラムの実行中にそれらを動的に変更する必要がある。従来の DVFS では動作周波数と供給電圧のみを動的に変更するのに対し、DCFS では稼働させるコア数も制御することで消費電力バジェットをより効率的に利用する。一定の消費電力制約下において提案手法を評価した結果、従来

の全コア実行に比べ最大で 35%の性能向上を達成した。

2.1.5.2 コア数と周波数に関するアプリケーション性能分析

本節では、消費電力制約下においてコア数および動作周波数に対するプロセッサの性能特性がプログラムの種類やその実行箇所に応じて異なることを示す。なお、本実験には PARSEC 2.1 [PARSEC2008]から選択したベンチマーク・プログラムと実機の AMD Opteron を用いた。本実験に用いたシステムの構成は表 2.1-3 の通りである。本システムは 4 プロセッサによる SMP (Symmetric Multi-Processor) 構成であり、各プロセッサが 8 コアを搭載するため合計で 32 コアの構成となっている. ベンチマークは *blackscholes*、*dedup*、*x264*、の 3 つを選択し、入力サイズはすべて *native* を用いた。

本節では、プロセッサの消費電力がそれを超えないよう最大動作周波数がコア数によって決定されると仮定する。消費電力制約としては、式(2.1-1)に示すように全コア(32 コア)が最低動作周波数で稼働する際の動的消費電力とする。ここで、α はスイッチング確率、$N_{allcores}$ はチップ上の全コア数、C は 1 コア当たりの負荷容量、f_{min} は最低動作周波数、V_{min} は最低供給電圧を表す。なお、プロセッサの負荷容量はコア数に比例すると仮定する。

$$P_{constraint} = \alpha \times N_{allcores} \times C \times f_{min} \times v^2_{min} \qquad (2.1\text{-}1)$$

表 2.1-3 実機での評価環境

プロセッサ	AMD Opteron 6136
プロセッサ数	4
1 プロセッサあたりの搭載コア数	8
利用可能な全コア数	32 (4 × 8)
L1 I/D キャッシュ	128 KB
L2 キャッシュ	512 KB
共有 L3 キャッシュ	12 MB
主記憶	16 GB (DDR3-1333)
バススピード	6.4 GT/s
テクノロジーサイズ	45 nm

表 2.1-4 消費電力制約下における動作周波数と電源電圧

コア数	動作周波数 [GHz]	供給電圧 [V]	制約に対する消費電力比 (最悪の場合)
1 – 5	2.4	1.3	0.88
6 – 8	1.9	1.2	0.97
9 – 12	1.5	1.13	0.99
13 – 19	1.1	1.04	0.97
20 – 32	0.8	0.95	1

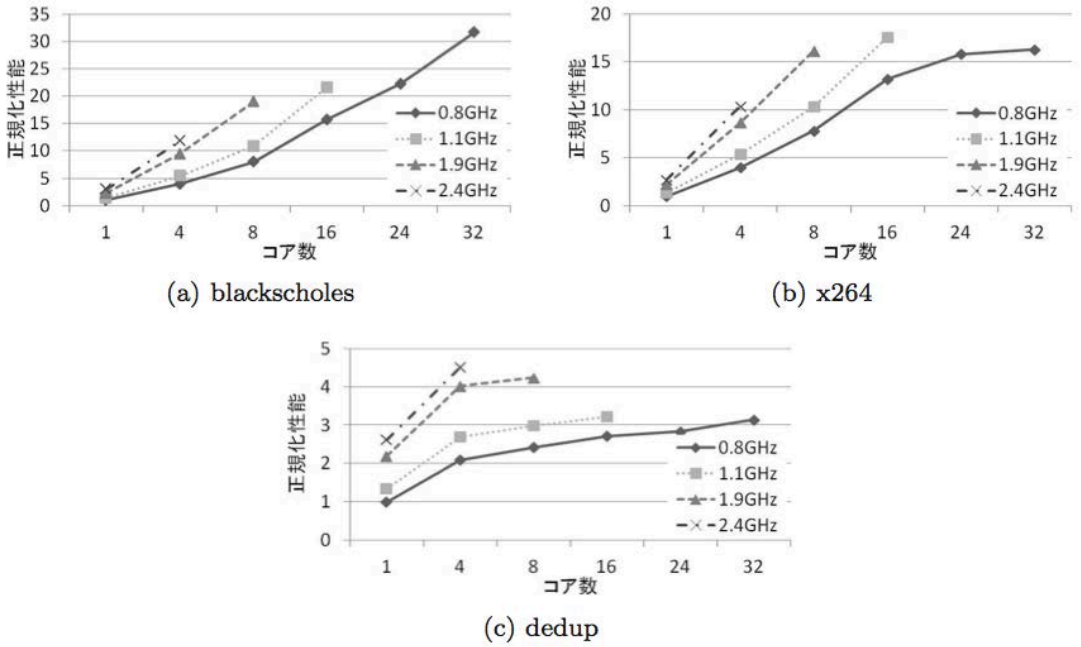


図 2.1-16 ベンチマークプログラム実行におけるコア数と性能の関係

$$P_{Ncores} = \alpha \times N_{cores} \times C \times f \times V^2 \quad (2.1\text{-}2)$$

$$\frac{P_{Ncores}}{Pconstraint} = \frac{N_{cores} \times f \times V^2}{N_{allcores} \times f_{min} \times V_{min}^2} < 1 \quad (2.1\text{-}3)$$

3 つのプログラムにおいて、コア数および動作周波数を変更した場合の性能を図 2.1-16 に示す。グラフの横軸はそれぞれのプログラムに割り当てられるコア数、縦軸は各プログラムの性能（実行時間の逆数）を表している。全ての値は、最低動作周波数（0.8GHz）で 1 コアにより実行した場合を 1 として正規化したものである。なお、本実験では、評価環境における全コア数と等しい 32 スレッドを生成してそれらを横軸が示す数のコアにバインドして（スレッドパッキング[Cochran2011]）実行しており、以降の実験においても同様である。また、4 種類の線はそれぞれ異なる動作周波数（0.8、1.1、1.9、2.4GHz）で実行した場合を表しており、各動作周波数で実行可能な最大コア数は消費電力制約により決定される。

図 2.1-16 の *blackscholes* では、コア数と動作周波数にほぼ比例した性能向上が得られている。このようなプログラムでは、動作周波数の上昇よりもコア数の増加のほうが性能を効率的に、すなわち少ない消費電力の増加で向上できる。これはプロセッサの消費電力が供給電圧の 2 乗と動作周波数に比例するためである。例えば、性能を 2 倍にするためにコア数もしくは動作周波数を 2 倍にする場合をそれぞれ考える。コア数を 2 倍に増加させると負荷容量の増加により消費電力は 2 倍になる。これに対し、動作周波数を 2 倍に上昇

すると供給電圧の上昇を伴うため、消費電力は 2 倍より大きくなる。よって、消費電力制約下において高い並列性を持つプログラムを実行する場合には、低い動作周波数ではあるものの、使用コア数を可能な限り多くすることが得策となる。一方、コア数増加に伴い性能向上が頭打ちになるようなプログラム(図 2.1-16 の *x264* や *dedup*) を実行する際には、コア数を制限し消費電力バジェットを動作周波数の上昇に用いることで性能を最大化できる。例えば、*x264* の場合、16 コアを用いた 1.1GHz での実行により最大性能を達成できる。*dedup* の場合、コア数増加に比べて動作周波数上昇による性能向上が大きいため、最大動作周波数である 2.4GHz で 4 コアにおいて実行した際に性能が最大となる。

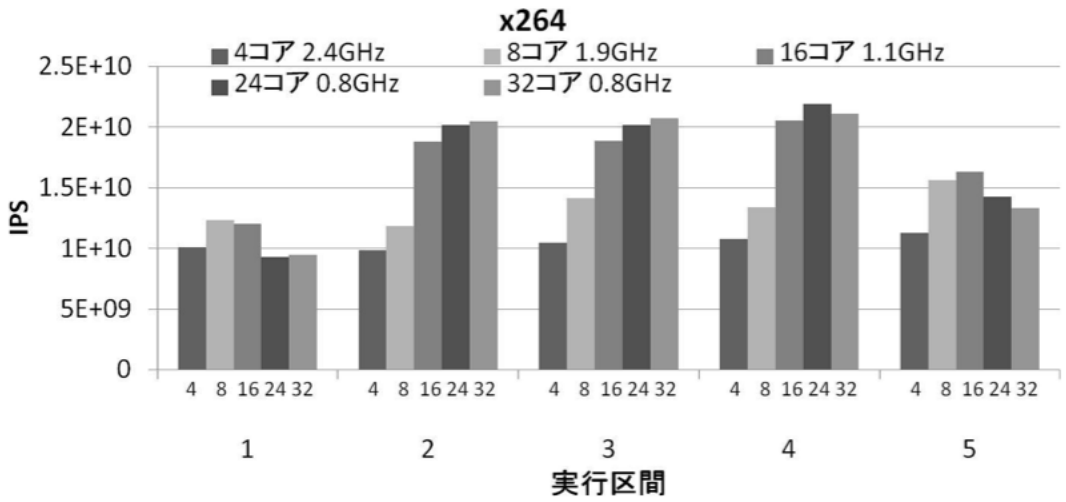


図 2.1-17 *X264* の実行におけるコア数と性能の関係

x264 の実行において一定区間ごとの性能特性を図 2.1-17 に示す。横軸は実行中の連続した 5 つの区間、縦軸は Instructions Per Second(IPS)を表している。5 種類のバーは、それぞれ異なるコア数（4、8、16、24、32）とその時の最大動作周波数の組み合わせで実行した場合の性能である。スレッドパッキングによる実行では、バインドするコア数に関わらず命令の総数がほぼ一定であるため、IPS が性能を示すのに適した指標となる。区間 1 では 1.9GHz の 8 コアによる実行が 5 種類の組み合わせの中で最大の IPS を達成している。これに対し、2 番目と 3 番目の区間では、最高性能を達成する組み合わせが 0.8GHz の 32 コアによる実行となる。また、4 番目と 5 番目の区間では、0.8GHz の 24 コアと 1.1GHz の 16 コアによる実行がそれぞれ性能を最大化している。この結果から、それぞれの区間によって適したコア数と動作周波数が異なることが分かる。

図 2.1-16 ならびに図 2.1-17 の結果から、消費電力制約下において性能を最大化するためには、最適なコア数と動作周波数を動的に制御する手法が必要であると言える。そこで本研究では、アプリケーションの特性（コア数と動作周波数に対する性能特性）を実行中に検知し、適切なコア数と動作周波数の組み合わせを選択する手法を提案する。

2.1.5.3 動的なコア数と動作周波数／電源電圧の最適化

提案する DCFS 手法は、トレーニングフェイズと実行フェイズと呼ばれる 2 種類のフェイズから成り立つ。トレーニングフェイズでは、ある短い時間毎に構成（コア数と動作周波数の組み合わせ）を変更しつつプログラムを実行し、特性を調べるための指標として IPS を測定・記録する。実行フェイズでは、測定した IPS から性能を最大化する構成を推測し、その構成によりプログラムが実行される。また、一定時間ごとに IPS を再計測することにより、プログラムの特性の変化を検知する。このトレーニングフェイズと実行フェイズはプログラムの実行終了まで繰り返される。本方式では、最適な構成を探索するアルゴリズムとして全探索法とヒルクライム法の 2 種類を実装した。なお、この手法は全て動的なものであり、プログラムの静的な解析やプログラム自体の修正は一切必要としない。本手法の概要を図 2.1-18 に示す。

トレーニングフェイズ：トレーニングフェイズでは、最適な構成を選択するために構成を動的に変更しつつ IPS を測定する。全探索法では、コア数を全コア数から 1 まで変化させ、全通りの性能を計測する。なお、各コア数において、そのコア数に応じた最大動作周波数のみで計測を行う。また、ヒルクライム法では、利用可能な動作周波数ごとにコア数を全コア数から減少させつつ IPS を計測し、IPS が最大になるコア数を探索する。以下、各アルゴリズムの詳細な説明を行う。

- 全探索法:全コア数とその時の最大動作周波数（図 2.1-16 の例では 32 コアと 0.8GHz の組み合わせであり、以降の説明においても図 2.1-16 の例を用いる）によりプログラムを実行し、ある一定時間（「1 構成トレーニング時間」と呼称）IPS を計測・記録する。次に、コア数を 1 減少させ、動作周波数をその時の最大動作周波数に上昇させる。そして、1 構成トレーニング時間中に IPS を再び計測・記録する。これをコア数が 1 になるまで繰り返す（コア数が 1 の時の動作周波数は 2.4GHz）。
- ヒルクライム法：全探索法と同様、まずは全コア数とその時の最大動作周波数（32 コアと 0.8GHz）によりプログラムを実行し、1 構成トレーニング時間中に IPS を計測・記録する。次に、動作周波数は一定のまま、使用するコア数を減少させ（24 コアと 0.8GHz）て IPS を得る。そして、IPS が低下するまでコア数を減少させ、現在の動作周波数（0.8GHz）において最大性能を達成する構成を探索する。なお、この手法は IPS をコア数の関数とした場合に、この関数が単峰性関数となっていることを仮定しており、IPS が低下する直前のコア数をその周波数で最大の性能を達成するコア数とみなす。続いて動作周波数を上昇させ（1.1GHz）、その時に最大限利用できるコア数から IPS が減少するまでコア数を減少させつつ IPS を得る。これを利用可能なすべての動作周波数（0.8、 1.1、 1.9、 2.4GHz）に関して繰り返す。

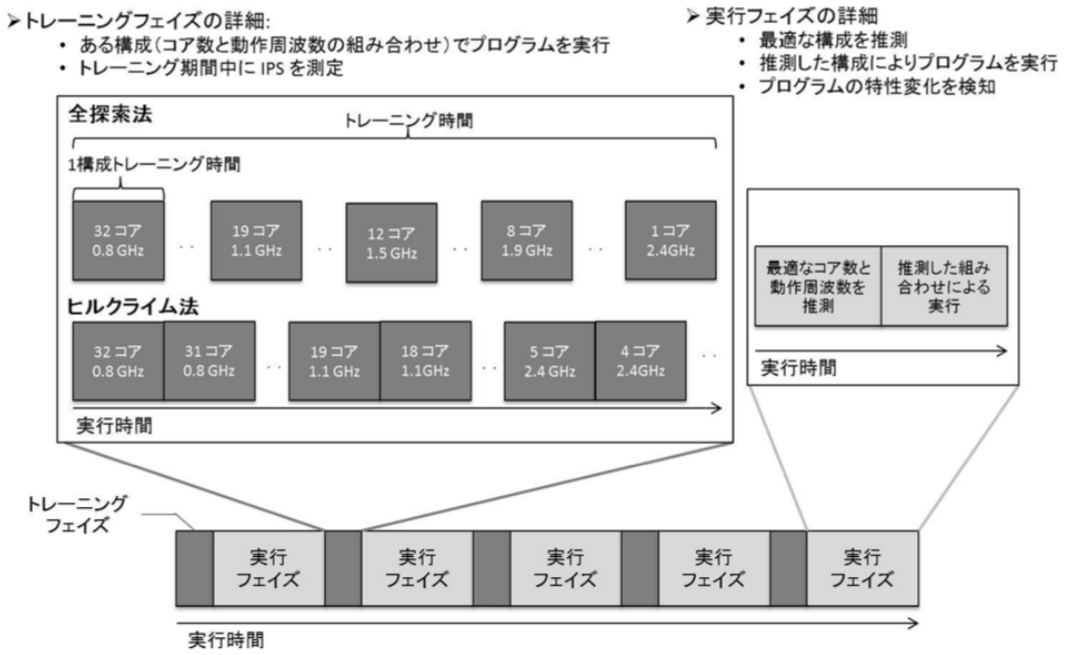


図 2.1-18 DCFS 動作の概要

実行フェイズでは、トレーニングフェイズにて記録したそれぞれの IPS を比較し、最大の IPS を達成した構成を現時点の最適な構成として選択し、この構成でプログラムを実行する。また、プログラムの性質が図 2.1-17 で見たようにプログラムの実行中に変化することが考えられる。このため、一定時間ごと（今回は 1 秒ごと）に IPS を計測し、現在の IPS が前回の IPS と比較して一定範囲以上変化すれば、プログラムの特性が変化したと見なし再びトレーニングフェイズに移行する。

トレーニングフェイズでは、最適でない構成においてもプログラムを実行するため、探索時にオーバーヘッドが発生する。よって、探索する構成の数が多いほど一度のトレーニングに要する時間（「トレーニング時間」と呼称）が長くなりオーバーヘッドが大きくなる。提案手法では、最適な構成を選択する精度を保ちつつ、トレーニングフェイズにおいて探索する構成の数を減らすことでトレーニング時間を短縮し、さらなる性能向上を得ることができる。

全探索法では全コアから 1 コアまでの構成において性能を計測するため、今回の実験環境では、一度のトレーニングフェイズにおいて探索する構成の数は 32 である。一方、ヒルクライム法では利用できる動作周波数の数が 5 通りであり、各動作周波数において 2 から 4 構成を探索するため、一度のトレーニングフェイズにおいて探索する構成数は 10 から 20 程度である。そのため、ヒルクライム法の適用により全探索法に比べオーバーヘッドを削減できる。将来のメニーコアでは、選択可能な動作周波数の数の増加に対してチップ上のコア数が大幅に増加することが予想される。そのため、今後ますますヒルクライム法の有

用性が高くなる。一方、精度に関しては、コア数を全通り探索する全探索法が最も優れているがヒルクライム法でも全探索法と同等の精度が得られる。なぜならヒルクライム法では、IPS をコア数の関数とした場合に単峰性関数となることを仮定しており、図 2.1-16 から分かるように実際のベンチマークにおいてもその仮定が正しいためである。

本手法を適用する際、プログラムの振舞いが変化した後にその振舞いが 1 度のトレーニングフェイズに要する時間に比べ十分長い時間継続する場合にトレーニングフェイズで予測した最適な構成での実行が可能となる。しかしながら、プログラムの振舞いが短い時間間隔で変化することもあり得る。このような場合、トレーニングフェイズの回数が多くなり、最適でない構成での実行により性能が低下する。また、トレーニングフェイズに続く実行フェイズにおいてプログラムの振舞いが変化する場合、最適な構成で実行フェイズを実行することができなくなるため、性能が低下するといった問題が発生する。

2.1.5.4 複数ソケットを用いたメニーコア環境への実装

実装を簡単化するために、本手法をユーザレベルのランタイムシステムとして実装した。具体的には、ハードウェアカウンタの値を読み出しプロファイリングを行うための Linux の標準ソフトウェアである *perf tool* を改良し、タイマにより定期的に実行命令数を計測し、またコア数と動作周波数を制御するハンドラに制御を移す機能を実装した。コア数を指定するためには、生成したスレッドを特定のコアに割り当てるための Linux 標準 API である *sched_setaffinity(2)* を用いた。また、動作周波数の変更に関しては、*/sys/devices/system/cpu/cpuX/cpufreq/scaling_setspeed*（X は CPUID）への動作周波数の値の書き込みで実装した。なお、本実装では、プロセッサ・アフィニティやコアの動作周波数を変更するためのシステムコールによりオーバーヘッドが発生する。本手法をカーネルのスケジューラに実装することで、これらのオーバーヘッドを削減できると考えられる。

提案手法では、1 構成トレーニング時間が性能を決定する重要なパラメタとなる。なぜなら、この時間が長すぎる場合には最適でない構成での実行が性能へ与える影響が大きくなり、短すぎる場合には動作周波数とコア数を変更した後にプロセッサの挙動が不安定になり正確な IPS を測定できないことがあるためである。動作周波数とコア数の変更後、プロセッサの挙動が安定するまで IPS の測定を待機しなければならない。ここでのプロセッサの挙動が不安定になるとは動作周波数とコア数を変更した直後に IPS の値が変動することであり、プロセッサの挙動が安定するとは IPS の値がある一定の値（構成の変更が完了したことを意味する値）に収束することである。動作周波数を変更する際には供給電圧の変更が行われるため、供給電圧が安定するまでの間 IPS が変動する可能性がある。しかしながら、供給電圧の変更は数十マイクロ秒で完了するため、ミリ秒単位で計測する IPS にはほとんど影響しない。これに対し、コア数の変更時に行われるスレッドのマイグレーショ

ンには数十ミリ秒ほどの時間を要するため、計測する IPS に対し影響を及ぼすと考えられる。本実験では、4 プロセッサ（1 プロセッサあたり 8 コアを搭載）から構成されるプラットフォームを用いている。共有 L3 キャッシュはプロセッサごとに搭載されているため、使用するコア数を 8 から 32 に変更する場合にスレッドのスケジューリングやキャッシュミス率の増加といった理由からスレッドのマイグレーションに最も長い時間を要すると予想できる。そこで、この場合において動作周波数・コア数変更後に IPS の値がある一定の値に収束する時間を計測したところ、最悪の場合で 30ms であった。そのため、コア数と動作周波数を変更した後の 30ms 間はデータを収集しないこととした。ただし、スレッドのマイグレーション後のキャッシュミス率増加が計測する IPS に与える影響はプログラムによって異なるため、この 30ms という値は今回用いたベンチマーク一式特有の値である。他のアプリケーションを対象とする場合には、IPS が収束する時間を同様に計測する必要がある。

また、探索時の性能低下を最小化するために、IPS の値が安定した後に現在の構成でプログラムを実行しつつ IPS を計測する時間も可能な限り短くしなければならない。本研究では提案手法をユーザレベルのソフトウェアで実装しており、定期的な IPS の計測や構成変更を実現するために *nanosleep(2)*システムコールを使用している。そこで、本実験で用いたプラットフォームにおいて nanosleep を用いて妥当な IPS を計測できる最短の時間を計測した結果、30ms となった。そのため、構成変更後に IPS の値が安定するまで 30ms 待機し、その後の 30ms において現在の構成での IPS を計測することとした。つまり、1 構成トレーニング時間は 60ms である。実行フェイズでは、現在の IPS が前回の IPS と比較し、10%以上増減すればプログラムの特性が変化したと見なし、トレーニングフェイズに移行する。

2.1.5.5 性能消費電力評価

本節では、実機による提案手法（DCFS）の評価結果について述べる。評価に用いたシステムの構成ならびに消費電力制約とコア数に応じた動作周波数の関係は表 2.1-4 で示した通りである。プログラムは PARSEC 2.1 [PARSEC2008]から *facesim*, *fluidanimate*, *raytrace* を除いた 10 個を用いる。なお、*facesim* と *fluidanimate* を実行する際、コア数が 2 のべき乗である場合しか正しく動作しない。本評価に用いるプラットフォームでは全コア数が 32 であり提案手法適用時の選択可能なコア数が 6 通りに限定されるため、これら 2 つのプログラムは除外した。また、*raytrace* はコンパイルできなかったため評価に用いていない。なお、すべての評価において *native* の入力サイズを用いた。

これらのプログラムを並列性に応じて分類するために、各プログラムを評価プラットフォームにおいて 32 コアと 1 コアで実行し、1 コア実行に対する 32 コア実行時の性能比を算出した。また、動作周波数が 0.8GHz と 2.4GHz のそれぞれの場合に各プログラムを 1 コアで実行し、0.8GHz での実行に対する 2.4GHz での実行の性能比を求めた。その結果を表 2.1-5 に示す。*Blackscholes*、*swaptions*、ならびに、*vips* の並列性が高いプログラムにおいては

従来型の実行方法である全コアでの実行によって高い性能が予想されるため、提案手法では性能を改善する余地が少ないと考えられる。一方で、並列性が中または低いプログラムにおいては、最適な構成が全コアでの実行ではない場合があると予想されるため、提案手法による性能改善が期待される。また、提案手法ではコア数増加に加えて動作周波数上昇により性能向上を狙うため、*canneal* や *streamcluster* のような（動作周波数上昇による性能向上が小さい）プログラムでは提案手法による性能向上が小さいことが考えられる。これに対し、それ以外のプログラムでは動作周波数上昇に比例した性能向上が得られると予想できる。

表 2.1-5 ベンチマーク・プログラムの特性

並列性	ベンチマーク	32 コア実行時の 1 コア実行時に対する性能比	2.4 GHz 実行時の 0.8 GHz 実行時に対する性能比
高	blackscholes	31.6x	2.98x
	swaptions	31.6x	2.96x
	vips	29.7x	2.94x
中	ferret	21.4x	2.98x
	freqmine	18.4x	2.96x
	x264	16.3x	2.78x
	canneal	13.0x	1.61x
	bodytrack	12.4x	2.99x
低	dedup	3.1x	2.86x
	streamcluster	2.9x	1.95x

提案手法を従来の全コア実行（動作周波数は最低動作周波数の 0.8GHz）と比較して評価を行う。提案手法については、全探索法とヒルクライム法(それぞれ、DCFS-EXH と DCFS-HILL) 2 種類の手法の性能を評価する。評価結果を図 2.1-19 に示す。横軸はベンチマーク、縦軸は従来の全コア実行時（0.8GHz の 32 コア）の性能（実行時間の逆数）で正規化した性能を表している。また、最も右側のバーは全コアを最高周波数で実行した場合（2.4GHz の 32 コア）の正規化性能を示しており、バーの上の数字は 1.4 倍以上の性能の値である。ただし、この構成で実行する際の消費電力は本研究で仮定した消費電力制約を超えるため、あくまでも参考値となる。

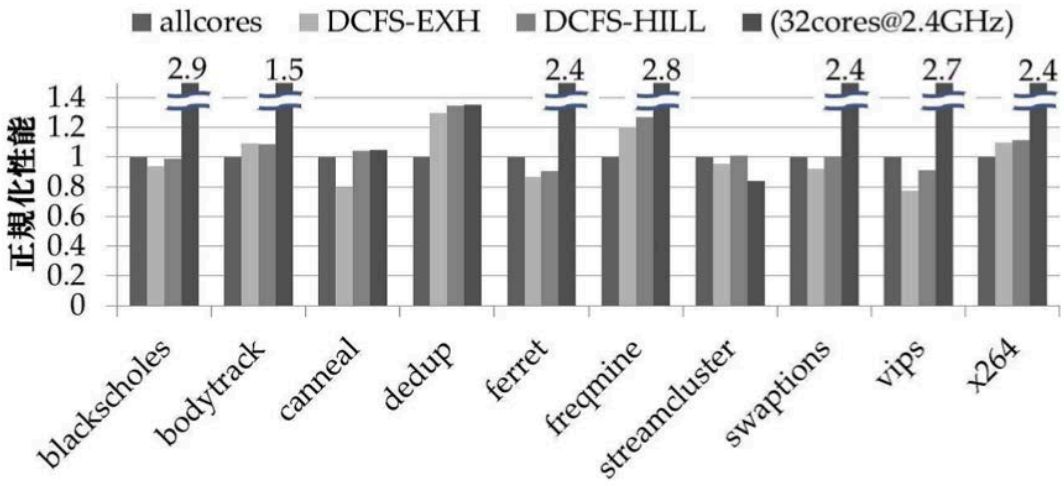


図 2.1-19 評価結果

高い並列性を持つプログラムである *blackscholes*、*swaptions*、*vips* については、提案手法のどちらの手法でも性能向上が得られていないか、もしくは性能が悪化している。また、動作周波数上昇による性能向上が小さいプログラムである *canneal* と *streamcluster* についても、性能向上はほとんど得られていない。一方、*bodytrack*、*dedup*、*freqmine*、*x264* では、提案手法により性能が向上した。表 2.1-5 から分かるように、これらは低・中程度の並列性を持つプログラムである。特に並列性の低い *dedup* では、最大で 35%の性能向上が得られた。評価に用いた全 10 個のプログラムにおいて、全探索法では平均 2%性能が悪化し、ヒルクライム法では平均 6%性能が向上した。また、性能向上が得られた 4 つのプログラムでの性能向上比の幾何平均は、全探索法で 17%、ヒルクライム法で 20%である。

bodytrack を除く全プログラムにおいて、全探索法に比べヒルクライム法のほうがより大きな性能向上を達成している。ヒルクライム法は全探索法に比べ一度のトレーニングフェイズにおいて探索する構成数が少なく、オーバーヘッドが小さいためである。表 2.1-6 にそれぞれの手法においてトレーニングに要した合計時間の実行時間に対する割合を示す。全探索法に比べヒルクライム法のほうがその割合を大幅に削減できていることが分かる。しかしながら、*ferret* や *x264* では依然として実行時間の約 15%近くをトレーニングに費やしている。より高い性能を達成するためには、さらにトレーニング時間を短縮できる探索アルゴリズムを考案する必要がある。また、図 2.1-19 の結果から、ヒルクライム法では最適なコア数と動作周波数決定の精度を維持できていることが分かる。

表 2.1-6 ベンチマーク・プログラム実行時間の内訳

ベンチマーク	トレーニングに要した合計時間の実行時間に対する割合（%）	
	DCFS-EXH	DCFS-HILL
blackscholes	11.5	4.0
bodytrack	10.2	2.2
canneal	9.4	6.9
dedup	15.9	6.9
ferret	33.3	14.8
freqmine	13.4	6.0
streamcluster	8.1	5.4
swaptions	6.7	1.3
vips	8.0	2.8
x264	38.4	18.8

提案手法の特性を理解するために、ferret、canneal、streamcluster について解析を行う。ferret の並列性は中程度であるにも関わらず、提案手法による性能向上は得られなかった。コア数と動作周波数に応じた ferret 実行時の性能を図 2.1-20 に示す。この結果から、コア数増加に伴い性能向上が得られるため、全コア実行により性能が最大となることが分かる。このようなプログラムの実行に提案手法を適用すると、トレーニングフェイズのオーバーヘッドにより性能が悪化する。よって、提案手法のトレーニングフェイズに要する時間をさらに短縮することで性能悪化を緩和できると言える。本研究における実装では、ランタイムシステムはユーザランドで動作するプロセスであり、プロセッサ・アフィニティやコアの動作周波数を変更するためのシステムコールによりオーバーヘッドが発生する。今後の課題としては、このシステムを OS のカーネルに実装することでオーバーヘッドを短縮することが挙げられる。

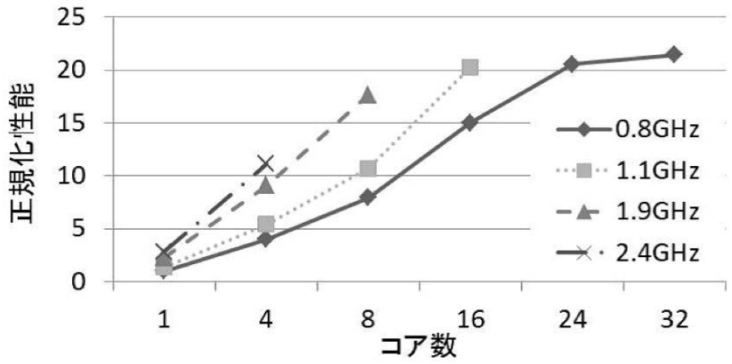


図 2.1-20 *ferret* 実行における性能

canneal と *streamcluster* も低い並列性を持つプログラムであるにも関わらず、提案手法

による性能向上は得られていない。*canneal* と *streamcluster* は評価に用いたプログラムの中で最もメモリバウンドな 2 つのプログラムであることが分かっている[Bienia2008]。DCFS は、コア数と動作周波数の制御により性能向上を狙う手法である。しかしながら、メモリバウンドなプログラムでは、動作周波数の変化による性能への影響が CPU バウンドなプログラムと比較して小さいため、動作周波数上昇による性能向上は期待できない。これは表2.1-5の0.8GHz での実行に対する2.4GHzの実行の性能比からも分かる。図2.1-21は、*canneal* と *streamcluster* の正規化性能を示したものである。この結果から，図 2.1-16 に示したプログラムに比べ、動作周波数上昇による性能向上が小さいことが分かる。このような場合、命令数だけでなくラストレベルキャッシュのミス数を監視することで、メモリバウンドなプログラムの特性を検知できる。その情報を利用すれば、コア数を動的に制御することで消費電力の削減が可能となる。

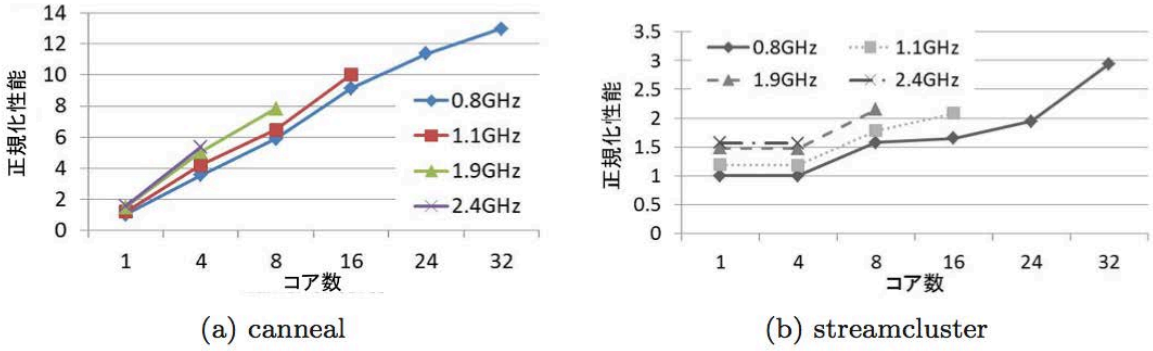


図 2.1-21　*canneal* と *streamcluster* の実行における性能

DCFS により最大の性能向上が得られた *dedup* において、DCFS の効果を詳細に分析する。図 2.1-22 は、0.8GHz の全コア実行（32 コア@0.8GHz）とコア数のみをヒルクライム法で制御する手法（DCS-HILL@0.8GHz）、提案手法（DCFS-HILL）の 3 種類の実行方式で *dedup* を実行した際の実行時間ごとの IPS を示している。また、それぞれのバーの上の数字はコア数を表す。DCS-HILL@0.8GHz の実行時間はトレーニングフェイズのオーバーヘッドにより全コア実行と比べわずかに延長しているが、5〜20 秒の区間においてコア数を制限することで IPS の向上を達成している。提案手法である DCFS-HILL では、使用しないコアの消費電力バジェットを動作周波数上昇に再割り当てすることで IPS をさらに向上できている。この結果から、DCFS の適用によりトレーニングフェイズのオーバーヘッドを補うのに十分な性能向上が得られると言える。

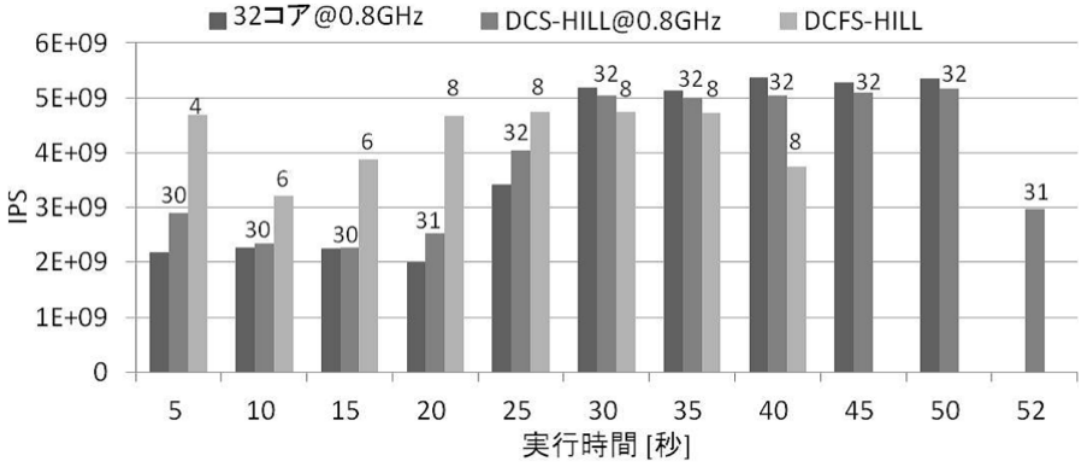


図 2.1-22　dedup の実行における性能

2.1.6 複数プログラム同時スケジューリング技術

2.1.6.1　　スケーラビリティ解析

本研究では、並列化された複数のアプリケーション・プログラムを同時実行することで、メニーコアにおける最大の課題である並列性の確保を実現する。一般に、並列化による性能向上はアプリケーションの特性によって様々である。図 2.1-23 は、PARSEC ベンチマーク[PARSEC2008]の実行におけるスケーラビリティ（すなわち、コア数を増加した際の性能向上率）を表す。ここで、「GREEN」にグルーピングされるアプリケーションは高いスケーラビリティを有しており、「RED」に属するアプリケーションはスケーラビリティが低いプログラムである。また、「YELLOW」はこれらの中間の位置づけとなる。横軸は各プログラムの実行に使用したコアの数を表す。また、縦軸は性能であり、コア数を 1 から 48 まで変化させた際に得られる最も高い性能で正規化している。例えば、GREEN に属する *blackscholes* では 48 コアを用いた実行が最も高い性能となる。一方、RED に属する *streamcluster* では 6 コアでの実行が最も良い結果となっている。各グラフの上部に記載した数字は、1 コアでの実行に対する性能向上値（スピードアップ）である。この結果より、以下に示すことが分かる。

- GREEN：高いスケーラビリティを有するプログラムにおていは、可能な限り多くのコアを用いて実行すべきである。
- YELLOW：基本的には、コア数の増加に伴い性能を向上できる。しかしながら、24～30 コアを超えた辺りから性能向上効果を得ることができていない（各グラフにおいて折れ線が横ばいになっている領域）。これは、コアを余剰に割り当てていることを意味しており、その結果としてハードウェア資源（つまりコア）の利用効率が低下する。
- RED：canneal に関しては YELLOW と同様の傾向であるが、性能向上効果が頭打ちになる点が 18 コアと比較的コア数が少ない。また、streamcluster においては 6 コア実行に

より劇的に性能が向上するものの、その後はコア数の増加に伴い性能が低下している。これは、同期処理やメモリ競合など、並列処理に伴う性能オーバヘッドが顕著に表れたためと予想される。このような場合には、コア数の増加を制限すべきである。

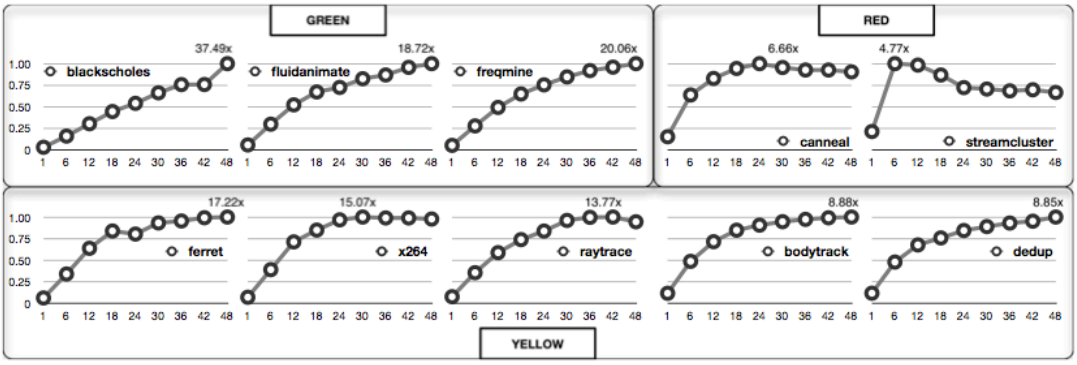


図 2.1-23　PARSEC プログラムのスケーラビリティ

2.1.6.2　スケーラビリティベース・スケジューラ SBMP の開発

第 2.1.6.1 節で説明したように､各プログラムが有するスケーラビリティは様々である。しかしながら、例えば Linux などの OS によるプロセス・スケジューラはこのようなスケーラビリティの違いを考慮していない。その結果、無駄にハードウエア資源（すなわちコア）を消費する状況が発生し、高い実効性能を実現できないといった問題がある。

そこで本研究では、実行中に各プログラムのスケーラビリティを推定し、その結果に基づき各プログラムに割り当てるべきコア数を動的に変更するスケジューラ SBMP（Scalability-based Manycore Partitioning）を開発した[Sasaki2012]。SBMP は、1)最適なコア割り当てが変わったタイミングを検出し、2)パフォーマンスカウンタの値を参照して、ターゲットとなるプログラムのスケーラビリティを推定する。そして、3)性能が最大化されるよう再スケジューリングを行う。本研究では、性能のメトリックとして ANTT（Average Normalized Turnaround Time）を用いる。なお、他のメトリックを用いた最適化を実施したい場合には、SBMP に実装された評価関数を変更するだけでよく、SBMP のスケジューリング方式そのものの一般性は失われない。

具体的には、複数プログラムの実行において、いずれかのプログラムにおいてスケーラビリティの変化（つまり、フェーズの変化）が発生した場合に、SBMP は再スケジューリングを行う。そして、SBMP 内部に搭載した性能モデルに基づき、それぞれのプログラムに対して適切な数のコアを割り当てる（本方式の詳細は文献[Sasaki2012]を参照)。本スケジューラはプロセッサダイの粒度（本実験環境の場合は 6 コア）でコア割り当てを行うことを基本とする。しかしながら、これだけではコアの利用率を最大化することができない。表

2.1-7 は、PARSEC ベンチマークを実行した際の CPU 利用率を表す。実行環境は第 2.1.6.3 節で説明する。一般に、実行に使用するコア数を増加すると CPU 利用率が低下する。これは、並列実行に伴う各種オーバヘッド等により、スケーラビリティはコア数に対して線形には向上しないためである。その結果、CPU 使用率が低下し、引いては、実効性能が低下するといった問題が生じる。この課題を克服するため、本研究ではコア・ドネーション技術を開発した。本技術では、6 コアといった粗粒度でのコア割り当てを行った後、CPU 利用率の低いプログラムが使用しているコアを、CPU 使用率が高いプログラムの実行に提供するものである。このように、粗粒度でのコア割り当てを行った後、コア・ドネーション技術により細流度のコア割り当てを実施することで CPU 利用率を更に高めることが可能となる。

表 2.1-7 PARSEC 実行における CPU 利用率

# of cores	bl	fl	fr	fe	x2	ra	bo	de	ca	st
6	98.6	96.7	96.4	99.9	99.5	89.0	73.7	73.9	82.1	92.6
12	93.6	90.4	86.8	92.7	97.2	78.4	55.5	59.0	71.1	80.7
18	89.8	83.4	65.4	95.4	83.9	71.9	46.4	51.1	68.5	67.3
24	85.3	74.8	68.6	87.9	72.2	62.9	39.0	44.8	69.8	56.7
30	84.4	68.0	61.3	74.7	59.3	60.2	32.5	38.8	67.3	50.6
36	80.5	62.0	43.1	82.2	46.1	50.5	29.9	34.9	66.2	45.7
42	69.6	58.8	34.9	72.6	38.7	43.8	26.8	31.4	70.4	42.3
48	85.7	54.5	44.5	72.1	33.3	37.6	24.1	28.7	65.9	37.7

2.1.6.3 性能評価

現状においては、実際に製造されたメニーコアプロセッサを入手することは困難である。そこで、本評価では 48 コアを搭載した SMP プロセッサシステムを用いた性能評価を実施した。実験環境の詳細を表 2.1-8 に示す。

表 2.1-8 実機を用いたメニーコア評価環境

Processor:	4 × AMD Opteron 6172
# of dies per processor	2
# of cores per die:	6
Total # of cores:	48
L3 cache size:	12 MB per socket
Main memory:	96 GB DDR3 PC3-10600

本評価においては、複数のプログラムが同時に実行される場合を想定しており、提案方式の効果は、同時実行されるプログラムの組み合わせに依存する。そこで、複数の異なるプログラムの組み合わせを想定して実験を行う。まず最初に、各プログラムの実行においては、スケーラビリティがさほど変化しない（プログラム間ではスケーラビリティが異なる）プログラムの組み合わせを用いる。表 2.1-9 に 4 つのプログラムから構成されるワークロード・ミックスを示す。ID はワークロード番号、Benchmarks は各プログラムの先頭 2 文字

のみを記している。また、Type は図 2.1-23 で示したスケーラビリティに基づくグループ（GREEN, YELLOW, RED）である。評価結果を図 2.1-24 に示す。横軸はワークロード ID、縦軸は ANTT である。ここで、SBMP は、プログラム実行中のフェーズ検出、ならびに、コア・ドネーションは行わない、最もシンプルな実装である。この結果より、Linux デフォルトスケジューラと比較して平均で 20%程度の ANTT 削減を達成している。

表 2.1-9 フェーズ変化を伴わないプログラムのワークロード・ミックス

ID	Benchmarks	Type
1	bo-ca-de-st	YYRR
2	bo-ca-fe-st	YYRR
3	bo-ca-ra-st	YYRR
4	ca-de-fe-st	YYRR
5	ca-de-ra-st	YYRR
6	ca-fe-ra-st	YYRR

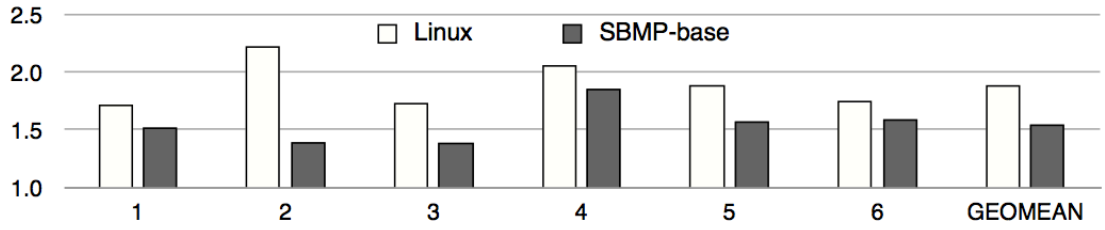


図 2.1-24　Linux デフォルトスケジューラとの性能比較(フェーズ変化なし)

次に、単一プログラム実行において複数のフェーズを有する（つまり、スケーラビリティが変化する）プログラムを含むワークロード・ミックスでの評価を行う。ワークロード・ミックスの詳細を表 2.1-10 に、評価結果を図 2.1-25 に示す。横軸はワークロード ID、縦軸は ANTT である。ここで、SBMP-PP は、SBMP-base において動的なフェーズ検出をサポートしたモデルである。評価結果より、動的フェーズ検出を行う（それにより適切なタイミングでコアの再割り当てを行う）SBMP-PP は、Linux や SBMP-base より大幅な性能向上を実現していることが分かる。特に、ワークロード 10 や 11 などでは顕著である。図 2.1-24 では SBMP-base は Linux より高い性能を実現したにも関わらず、本ワークロードでは性能が低下している。これは、SBMP-base はプログラム実行開始時に 1 度のみコア割り当てを行う方式であり、その結果として動的なスケーラビリティの変化に追従できないためである。これに対し、SBMP-PP はフェーズ検出によりスケーラビリテの変化に追従でき、高い性能を実現している。

表 2.1-10 フェーズ変化を伴うプログラムを含むワークロード・ミックス

ID	Benchmarks	Type	ID	Benchmarks	Type
7	bo-ca-fe-fr	GYYR	16	de-fr-st-x2	GYYR
8	bo-ca-fr-x2	GYYR	17	fe-fr-ra-st	GYYR
9	bo-fe-fr-st	GYYR	18	fr-ra-st-x2	GYYR
10	bo-fr-st-x2	GYYR	19	bo-ca-fe-x2	YYYR
11	ca-de-fe-fr	GYYR	20	bo-fe-st-x2	YYYR
12	ca-de-fr-x2	GYYR	21	ca-de-fe-x2	YYYR
13	ca-fe-fr-ra	GYYR	22	ca-fe-ra-x2	YYYR
14	ca-fr-ra-x2	GYYR	23	de-fe-st-x2	YYYR
15	de-fe-fr-st	GYYR	24	fe-ra-st-x2	YYYR

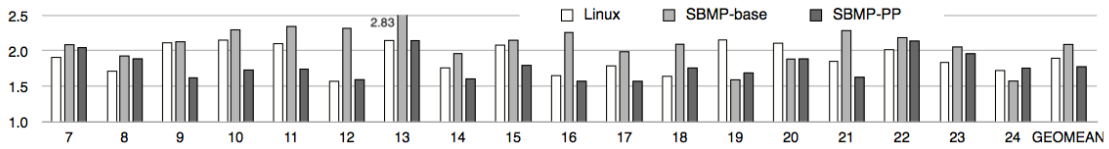


図 2.1-25 Linux デフォルトスケジューラとの性能比較(フェーズ変化あり)

表 2.1-11 低い CPU 利用率をとなるプログラムを含むワークロード・ミックス

ID	Benchmarks	Type	ID	Benchmarks	Type
25	bo-de-fe-fr	GYYY	32	bo-de-ra-x2	YYYY
26	bo-de-fe-x2	GYYY	33	bo-ca-de-fe	YYYR
27	bo-de-fr-ra	GYYY	34	bo-ca-de-ra	YYYR
28	bo-de-fr-x2	GYYY	35	bo-ca-de-x2	YYYR
29	bo-ca-de-fr	GYYR	36	bo-de-fe-st	YYYR
30	bo-de-fr-st	GYYR	37	bo-de-ra-st	YYYR
31	bo-de-fe-ra	YYYY	38	bo-de-st-x2	YYYR

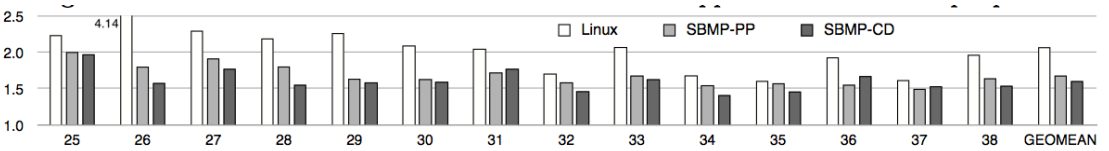


図 2.1-26 Linux デフォルトスケジューラとの性能比較(低 CPU 利用率あり)

最後に、低い CPU 利用率となるプログラム（*bodytrack* と *dedup*）を含むワークロード・ミックスでの評価を行う。ワークロード・ミックスの詳細を表 2.1-11 に、評価結果を図 2.1-26 に示す。横軸はワークロード ID、縦軸は ANTT である。ここで、SBMP-CD は、動的フェーズ検出とコア・ドネーションの両方をサポートしたモデルである。評価の結果から、SBMP-CD は大幅な性能向上を実現していることが分かる。特に、ワークロード 26 においては、Linux スケジューラと比較して ANTT を半分以下に抑えている。これは、スケーラビリ

ティを考慮した祖粒度ならびに細流度なコア割り当てが極めて効果的に動作したためである。すべてのワークロード・ミックスにおける評価結果を図 2.1-27 に示す。横軸はワークロード ID であるが、各評価対象モデルにおいて性能に基づきソートしている。また、縦軸は、これまでの図と同様に ANTT である。この図より、ほとんどすべてのワークロードにおいて、SBMP-CD は Linux の性能を上回っていることが分かる。これにより、提案方式は様々な状況においても効果的に動作することが確認された。

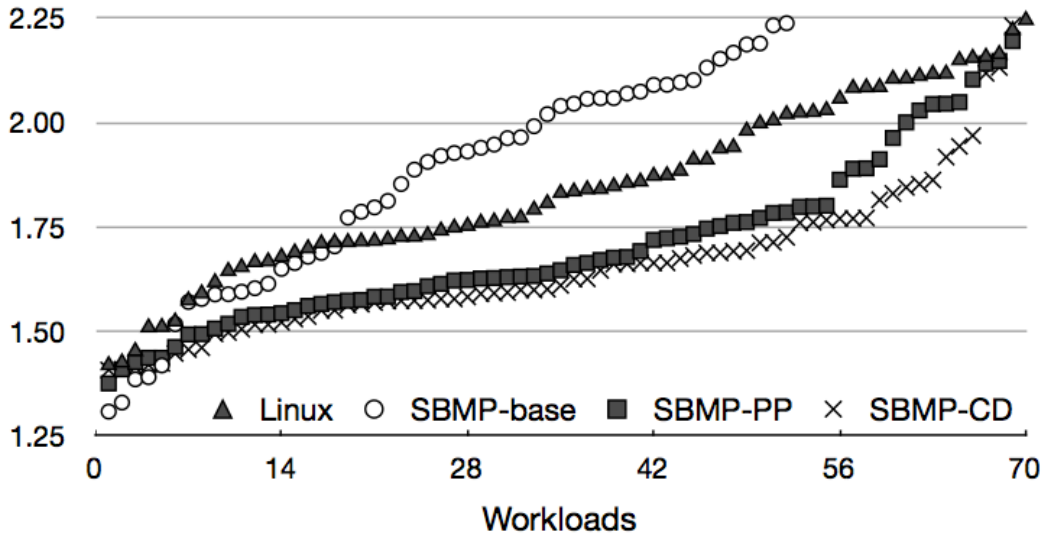


図 2.1-27　Linux デフォルトスケジューラとの性能比較(全ワークロード・ミックス)

2.1.6.4　スケジューラの拡張と消費エネルギー評価

第 2.1.6.3 節までにおいては、性能向上のみを目的としたスケジューラの開発について説明した。本研究プロジェクトでは、SBMP を改良し、消費電力制約下において性能を最大化する新たなスケジューラを開発した。現段階では、内容は未公開であるため詳細は省略するが、AMD Opteron 64 コアの実機システムを用いて評価を行った結果、並列化された各プログラムにおいて 64 コアを用いた並列実行を順次行う従来の実行モデルと比較して、複数プログラムの同時実行を想定することで約 70%の低消費エネルギー化を実現することができた。なお、本評価で測定した消費電力は、電力メータを用いたシステム全体の電力値である。

2.1.7 低レベル API

本節では、SMYLEref アーキテクチャにおいてホストコアからアクセラレータコアを制御するための API の外部仕様ならびに内部仕様を説明する。SMYLEref 低レベル API は、SMYLEref 向け OpenCL フレームワークによって使用されるものであり、ハードウェア（メモリやプロセッサの制御レジスタ等）を直接操作する際のインタフェースを規定したものである。以下、低レベル API の外部仕様を示す。

初期化／終了

srl_init	
書式	srl_result srl_init(void)
機能	低レベル API の初期化処理。
戻り値	成功時：SRL_SUCCEEDED 失敗時：SRL_FAILED
引数	なし
解説	他の API 呼び出しに先立って、一度だけ本 API を呼び出さなければならない。

srl_close	
書式	srl_result srl_close(void)
機能	低レベル API の終了処理。
戻り値	成功時：SRL_SUCCEEDED 失敗時：SRL_FAILED
引数	なし
解説	他の API 呼び出しの完了後、一度だけ本 API を呼び出さなければならない。

実行関連

srl_run	
書式	srl_result srl_run(srl_core_id_t core_id, void* addr)
機能	アクセラレータコアへの実行指示
戻り値	成功時：SRL_SUCCEEDED 失敗時：SRL_FAILED
引数	core_id: 指示対象のアクセラレータコアの ID addr: アクセラレータコアが実行するプログラムのエントリポイント（物理アドレス）
解説	コア ID=core_id のコアに対して、addr 番地に処理を移すように指示する。

srl_suspend	
書式	srl_result srl_suspend(srl_core_id_t core_id, void** pc)
機能	アクセラレータコアへの実行の一時中断指示
戻り値	成功時：SRL_SUCCEEDED 失敗時：SRL_FAILED
引数	core_id: 指示対象のアクセラレータコアの ID pc: 実行を中断した番地の格納先。pc の領域は呼び出し側で確保すること。
解説	コア ID=core_id のコアに対して、実行を一時中断するよう指示する。 中断されたアドレスは pc が指す先に格納される。

srl_restart	
書式	srl_result srl_restart(srl_core_id_t core_id)
機能	アクセラレータコアへの実行再開指示
戻り値	成功時：SRL_SUCCEEDED 失敗時：SRL_FAILED
引数	core_id: 指示対象のアクセラレータコアの ID
解説	コア ID=core_id のコアに対して、中断した実行を再開するよう指示する。

srl_abort	
書式	srl_result srl_abort(srl_core_id_t core_id)
機能	アクセラレータコアへの強制終了指示
戻り値	成功時：SRL_SUCCEEDED 失敗時：SRL_FAILED
引数	core_id: 指示対象のアクセラレータコアの ID
解説	コア ID=core_id のコアに対して、実行中の処理を強制終了するよう指示する。

問い合わせ

srl_isActive	
書式	int srl_isActive(srl_core_id_t core_id)
機能	アクセラレータコアがプログラムを実行中あるか否かを調べる
戻り値	アクセラレータコアがプログラムを実行中の場合は 1 を、さもなくば 0 を返す。
引数	core_id: 指示対象のアクセラレータコアの ID
解説	コア ID=core_id のコアが Running 状態か否かを返す。

srl_isSuspended	
書式	int srl_ isSuspended (srl_core_id_t core_id)
機能	アクセラレータコアがプログラムの実行を一時中断しているか否かを調べる
戻り値	アクセラレータコアがプログラムの実行を一時中断している場合は 1 を、さもなくば 0 を返す。
引数	core_id: 指示対象のアクセラレータコアの ID
解説	コア ID=core_id のコアが Suspended 状態か否かを返す。

srl_isStopped	
書式	int srl_ isStopped (srl_core_id_t core_id)
機能	アクセラレータコアが停止中であるか否かを調べる
戻り値	アクセラレータコアが停止中である場合は 1 を、さもなくば 0 を返す。
引数	core_id: 指示対象のアクセラレータコアの ID
解説	コア ID=core_id のコアが Suspended 状態か否かを返す。

srl_isBusy	
書式	int srl_ isBusy (srl_core_id_t core_id)
機能	アクセラレータコアが Busy 状態であるか否かを調べる
戻り値	アクセラレータコアが Busy 状態である場合は 1 を、さもなくば 0 を返す。
引数	core_id: 指示対象のアクセラレータコアの ID
解説	コア ID=core_id のコアが Busy 状態か否かを返す。

SPM アクセス（予定）

srl_read_spm	
書式	srl_result srl_read_spm(srl_core_id_t core_id, void* dst_addr, void* spm_offset, int size)
機能	指定されたアクセラレータコアの SPM からホストのユーザアドレス空間へデータをコピーする
戻り値	成功時：SRL_SUCCEEDED 失敗時：SRL_FAILED
引数	core_id: コピー元のアクセラレータコアの ID dst_addr: コピー先のアドレス（Linux の仮想アドレス）

	spm_offset: コピー元のアドレス。SPM の先頭アドレスからのオフセット値を指定する。 size: コピーするバイト数
解説	コア ID=core_id のコアの SPM のオフセット spm_offset の位置から size バイトのデータを、dst_addr が指す先にコピーする。

srl_write_spm	
書式	srl_result srl_read_spm(srl_core_id_t core_id, void* spm_offset, void* src_addr, int size)
機能	ホストのユーザアドレス空間から指定されたアクセラレータコアの SPM へデータをコピーする
戻り値	成功時：SRL_SUCCEEDED 失敗時：SRL_FAILED
引数	core_id: コピー先のアクセラレータコアの ID spm_offset: コピー先のアドレス。SPM の先頭アドレスからのオフセット値を指定する。 src_addr: コピー元のアドレス（Linux の仮想アドレス） size: コピーするバイト数
解説	src_addr 番地から size バイトのデータを、コア ID=core_id のコアの SPM のオフセット spm_offset にコピーする。

L1 キャッシュ設定（予定）

srl_set_l1_config	
書式	srl_result srl_set_l1_config(srl_core_id_t core_id, srl_l1_conig_mode mode)
機能	L1 キャッシュ・SPM の設定を行う
戻り値	成功時：SRL_SUCCEEDED 失敗時：SRL_FAILED
引数	core_id: コピー先のアクセラレータコアの ID mode: L1 キャッシュ／SPM をどのように設定するかを指定する。
解説	SMYLEref の L1 キャッシュは一部を SPM として利用するように設定可能である。本 API を用いて設定可能なキャッシュ・SPM の構成は次のとおりである。

	キャッシュのみ 16KB キャッシュのみ 8KB キャッシュのみ 4KB SPM のみ 16KB SPM のみ 8KB SPM のみ 4KB キャッシュ 12KB＋SPM 4KB キャッシュ 8KB＋SPM 8KB キャッシュ 4KB＋SPM 12KB キャッシュ、SPM ともに不使用

srl_get_l1_config	
書式	srl_result srl_get_l1_config(srl_core_id_t core_id, srl_l1_conig_mode* mode)
機能	L1 キャッシュ、SPM の設定を読み出す
戻り値	成功時：SRL_SUCCEEDED 失敗時：SRL_FAILED
引数	core_id: コピー先のアクセラレータコアの ID mode: キャッシュ設定の読み出し先アドレス。呼び出し側で領域を確保すること。
解説	L1 キャッシュ・SPM の設定を読み出し、mode に格納する。

型一覧

srl_core_id_t	
定義位置	sr_lowapi/sr_lowapi.h
説明	コアの ID を格納する型。 ホストコアの ID は０。 アクセラレータコアの ID は 1 から MAXIMUM_CORE_ID まで。

srl_result	
定義位置	sr_lowapi/sr_lowapi.h
説明	API 関数の実行結果の成否を示す型。

<table>
<tr><td colspan="2">srl_l1_config_mode（予定）</td></tr>
<tr><td>定義位置</td><td>sr_lowapi/sr_lowapi.h</td></tr>
<tr><td>説明</td><td>L1 キャッシュ／SPM の構成を示す列挙型。取りうる構成と列挙子の関係は次の通り。
<table>
<tr><td>列挙子</td><td>構成</td></tr>
<tr><td>C16KB4W</td><td>キャッシュのみ 16KB</td></tr>
<tr><td>C8KB2W</td><td>キャッシュのみ 8KB</td></tr>
<tr><td>C4KB1W</td><td>キャッシュのみ 4KB</td></tr>
<tr><td>S16KB</td><td>SPM のみ 16KB</td></tr>
<tr><td>S8KB</td><td>SPM のみ 8KB</td></tr>
<tr><td>S4KB</td><td>SPM のみ 4KB</td></tr>
<tr><td>C12KB3W_S4KB</td><td>キャッシュ 12KB＋SPM 4KB</td></tr>
<tr><td>C8KB2W_S8KB</td><td>キャッシュ 8KB＋SPM 8KB</td></tr>
<tr><td>C4KB1W_S12KB</td><td>キャッシュ 4KB＋SPM 12KB</td></tr>
<tr><td>L1Disable</td><td>キャッシュ、SPM ともに不使用</td></tr>
</table>
</td></tr>
</table>

アクセラレータコアの状態遷移と API の戻り値：アクセラレータコアは、ホストから与えられるプログラムの実行状態などによって次の４つの状態のいずれかの状態を取る。

状態名	説明
Stopped	プログラムを実行していない、待機状態。
Running	プログラム実行状態。
Suspended	プログラム実行を一時中断している状態。
Busy	API の内部処理を実行している状態。

アクセラレータコアの状態は実行関連 API（srl_run など）、SPM アクセス API、L1 キャッシュ設定 API の呼び出しによって遷移する。以下、状態遷移表を示す。表の各セルの内容は「次の状態／API の戻り値」である。

表 2-1-12 アクセラレータコアの状態遷移表

		現在のコアの状態			
		Running(**)	Stopped	Suspended	Busy(***)
API	srl_run()	Running(*)/ failed	Running/ succeeded	Suspended/ failed	Busy(*)/ failed
	srl_suspend()	Suspended/ succeeded	Stopped/ failed	Suspended/ failed	Busy(*)/ failed
	srl_restart()	Running/ failed	Stopped/ failed	Running/ succeeded	Busy(*)/ failed
	srl_abort	Stopped/ succeeded	Stopped/ failed	Stopped/ succeeded	Busy(*)/ failed
	srl_write_spm() srl_read_spm()	Running(*)/ failed	Busy/ succeeded	Busy/ succeeded	Busy(*)/ failed
	srl_set_l1_config() srl_get_l1_config()	Running(*)/ failed	Busy/ succeeded	Suspended/ failed	Busy(*)/ failed

(*) API によるコアへの指示を無視し、実行中のカーネル／API 処理を継続する。
(**) カーネルの実行が完了したら、Stopped に遷移する。
(***) API の内部処理が完了したら、アクセス開始前の状態に遷移する。

SMYLEref アーキテクチャは、MIPS ベースのプロセッサコアを多数使用したメニーコア構成となっている。主記憶はすべての領域が共有されている。全てのコアは他の任意のコアに対してコア間割り込みをかけることができる。以下、低レベル API の内部仕様を示す。

モジュール階層：低レベル API は次の 5 つのモジュールからなる。モジュール階層を図 2.1-28 に示す。

sr_base
コア間割り込み関数や CP0 レジスタアクセス用マクロなど、ホスト側、アクセラレータ側双方にリンクされるコードの集合。

sr_comm
ホスト～アクセラレータ間の通信プロトコル定義部。

sr_dd
Linux にリンクされるデバイスドライバ。ホスト上で動作する。

sr_lowapi
低レベル API のホスト側インタフェース。ユーザ空間での実装もこのモジュールに含む。

sr_accl
アクセラレータ側で動作するモニタプログラム。通信プロトコルの実装部分にあたる。

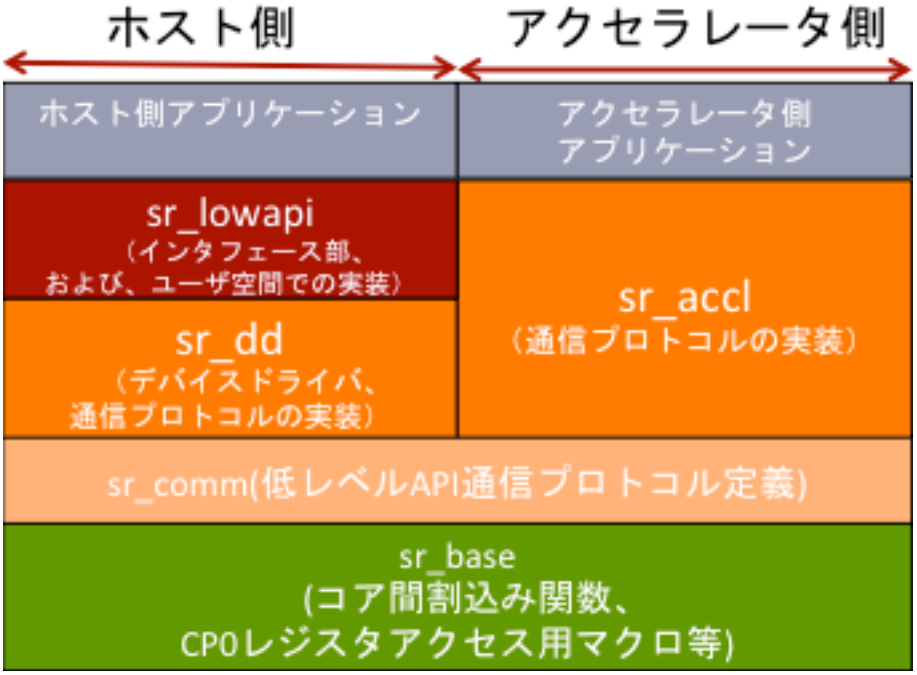


図 2.1-28 モジュール階層

ホスト～アクセラレータ間の通信用共有変数：ホストからアクセラレータへの指示の内容や、アクセラレータからホストへのレスポンスは、全て共有変数を介して通信される。この共有変数は 4 ワード×コア数のサイズを持つ 2 次元配列である。先頭アドレスはホスト側では sr_lowapi/sr_lowapi.c 内で定義される”get_comm_mem()”という内部関数によって取得できる。アクセラレータ側では sr_accl/sr_accl.c 内で定義される”comm_mem”という変数で参照される。共有変数の使い方は、API 関数によって異なる。以下に API 関数毎の共有領域の使い方を記す。

srl_run()		
ワードオフセット	用途	通信方向
[0]	コマンド(run)	ホスト→アクセラレータ
[1]	レスポンス	ホスト←アクセラレータ
[2]	アクセラレータコアの状態	ホスト←アクセラレータ
[3]	実行開始アドレス （物理アドレス）	ホスト→アクセラレータ

srl_suspend()		
ワードオフセット	用途	通信方向
[0]	コマンド(suspend)	ホスト→アクセラレータ
[1]	レスポンス	ホスト←アクセラレータ
[2]	アクセラレータコアの状態	ホスト←アクセラレータ
[3]	実行を中断したアドレス	ホスト←アクセラレータ

srl_restart()		
ワードオフセット	用途	通信方向
[0]	コマンド(restart)	ホスト→アクセラレータ
[1]	レスポンス	ホスト←アクセラレータ
[2]	アクセラレータコアの状態	ホスト←アクセラレータ
[3]	未使用	

srl_abort()		
ワードオフセット	用途	通信方向
[0]	コマンド(abort)	ホスト→アクセラレータ
[1]	レスポンス	ホスト←アクセラレータ
[2]	アクセラレータコアの状態	ホスト←アクセラレータ
[3]	未使用	

以下は実装が予定されている API 関数についての記述である。これらの API を実装する際には、共有変数のサイズを 6 ワード×コア数に拡張する必要がある。

srl_read_spm()		
ワードオフセット	用途	通信方向
[0]	コマンド(read_spm)	ホスト→アクセラレータ
[1]	レスポンス	ホスト←アクセラレータ
[2]	アクセラレータコアの状態	ホスト←アクセラレータ
[3]	コピー元の SPM 先頭からのオフセットアドレス	ホスト→アクセラレータ
[4]	コピー先の VAM 領域の物理アドレス	ホスト→アクセラレータ
[5]	コピーサイズ	ホスト→アクセラレータ

srl_write_spm()		
ワードオフセット	用途	通信方向
[0]	コマンド(read_spm)	ホスト→アクセラレータ
[1]	レスポンス	ホスト←アクセラレータ
[2]	アクセラレータコアの状態	ホスト←アクセラレータ
[3]	コピー先の SPM 先頭からのオフセットアドレス	ホスト→アクセラレータ
[4]	コピー元の VAM 領域の物理アドレス	ホスト→アクセラレータ
[5]	コピーサイズ	ホスト→アクセラレータ

srl_set_l1_config()		
ワードオフセット	用途	通信方向
[0]	コマンド(read_spm)	ホスト→アクセラレータ
[1]	レスポンス	ホスト←アクセラレータ
[2]	アクセラレータコアの状態	ホスト←アクセラレータ
[3]	設定するモード	ホスト→アクセラレータ
[4]	reserved	
[5]	reserved	

srl_get_l1_config()		
ワードオフセット	用途	通信方向
[0]	コマンド(read_spm)	ホスト→アクセラレータ
[1]	レスポンス	ホスト←アクセラレータ
[2]	アクセラレータコアの状態	ホスト←アクセラレータ
[3]	設定されたモード	ホスト←アクセラレータ
[4]	reserved	
[5]	reserved	

主要な定数：低レベル API の実装に用いられる主要な定数を以下に記す。

アクセラレータコアの状態

sr_comm/sr_comm.h

```
typedef enum {
    SR_COMM_RUNNING,
    SR_COMM_STOPPED,
    SR_COMM_SUSPENDED,
    SR_COMM_BUSY,
    SR_COMM_NUM_OF_STATES
} sr_comm_accl_state;
```

コマンド

sr_comm/sr_comm.h

```
typedef enum {
    SR_COMM_RUN,
    SR_COMM_SUSPEND,
    SR_COMM_RESTART,
    SR_COMM_ABORT,
    SR_COMM_NUM_OF_COMMANDS,
 } sr_comm_command;
```

通信用共有変数のワードオフセット

sr_comm/sr_comm.h

```
typedef enum {
    SR_COMM_COMMAND      = 0,
    SR_COMM_RESULT       = 1,
    SR_COMM_STATE        = 2,
    SR_COMM_STARTADDR    = 3,
    SR_COMM_PC           = 3,
    SR_COMM_NUM_OF_COMM_WORDS
 } sr_comm_commmem_offset;
```

成功／失敗、真偽

sr_comm/sr_comm.h

```
typedef enum {
        SR_COMM_FAILED          = 0,
        SR_COMM_SUCCEEDED       = 1,
 } sr_comm_result;

 typedef enum {
        SR_COMM_FALSE           = 0,
        SR_COMM_TRUE            = 1,
 } sr_comm_bool;
```

API 関数の実行フロー:API 関数はその実行フローによって 3 つのグループに分類される。

- 処理がホストのみで完結する関数。
- アクセラレータコアでの処理を必要とするもののうち、アクセラレータコアでの処理の完了を待たずに API から処理が戻るもの（ノンブロッキング関数)。
- アクセラレータコアでの処理を必要とするもののうち、アクセラレータコアでの処理の完了を待って API から処理が戻るもの（ブロッキング関数)。

本節では上記 3 つのグループのそれぞれについて実行フローを示す。

処理がホストのみで完結する関数：このグループに分類される API 関数は処理がホストのみで完結し、アクセラレータコアの関与が不要な API 関数である。

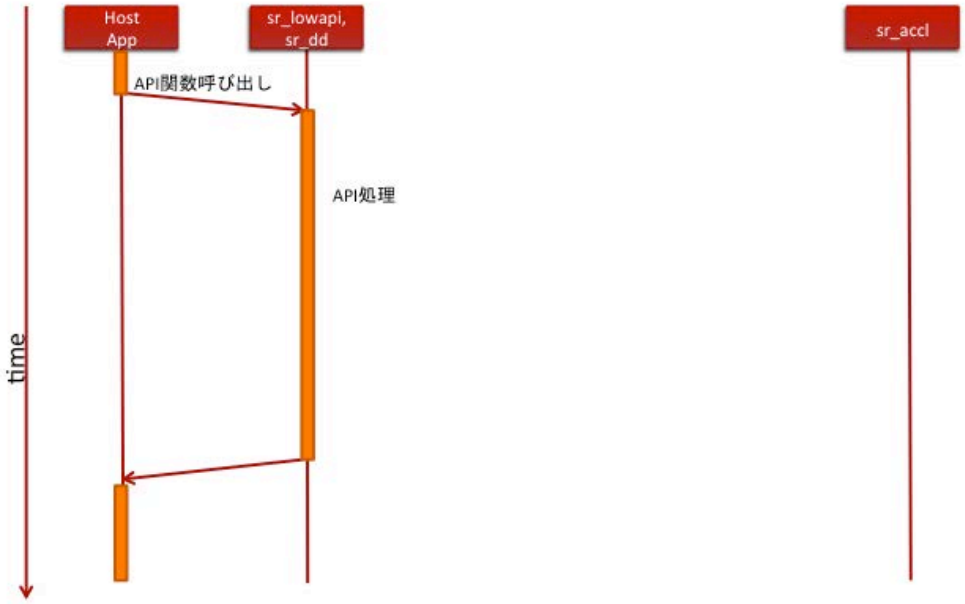


図 2.1-29 処理がホストのみで完結する関数の実行フロー

ノンブロッキング関数：このグループに分類される API 関数は、アクセラレータコアに対して処理を依頼し、その応答を必要としない API 関数である。アクセラレータコア側の処理の完了をホスト側が待たない（ブロックしない）ため、ノンブロッキング関数と呼ぶ。アクセラレータコアへの処理の指示は、通信用共有変数上の適切な位置に指示内容を書き込み、アクセラレータコアに対してコア間割り込みをかけることによって行う。割り込みをかけた後、ホストコアはアクセラレータコアからの割り込みを待つ。割り込みを受けたアクセラレータコアは通信用共有変数を読み出す。自身の状態と指示内容に基づいて指示を受け入れるか拒否するかを決定し、その結果を通信用共有変数に書き込んだ後、ホストコアに対して割り込みをかける。その後、指示を受け入れた場合には自身の状態を適切に変化させ、その内容に沿った処理を実行する。アクセラレータコアからの割り込みを受けたホストは自身の待ちを解除し、API 関数を終了する。

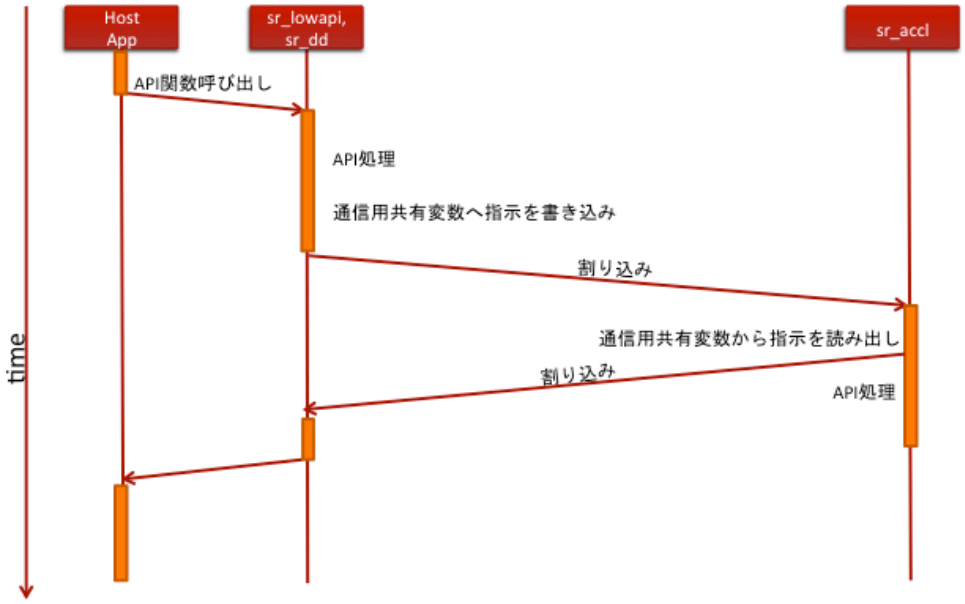


図 2.1-30 ノンブロッキング関数の実行フロー

ブロッキング関数：このグループに分類される API 関数は、アクセラレータコアに対して処理を依頼し、その応答を必要とする API 関数である。アクセラレータコア側の処理の完了をホスト側が待つ（ブロックする）ため、ブロッキング関数と呼ぶ。ホスト側の処理フローはノンブロッキング関数と同様である。アクセラレータコアへの処理の指示を通信用共有変数に書き込んでアクセラレータコアに割り込みをかけた後、割り込み待ちに入る。割り込みを受けたアクセラレータコアは通信用共有変数を読み出す。自身の状態と指示内容に基づいて指示を受け入れるか拒否するかを決定し、その結果を通信用共有変数に書き込んだ後、支持された処理を実行する。処理が完了するとホストコアに対して割り込みをかける。アクセラレータコアからの割り込みを受けたホストは自身の待ちを解除し、アクセラレータコアからの応答をうけて適切な処理を実施し、API 関数を終了する。

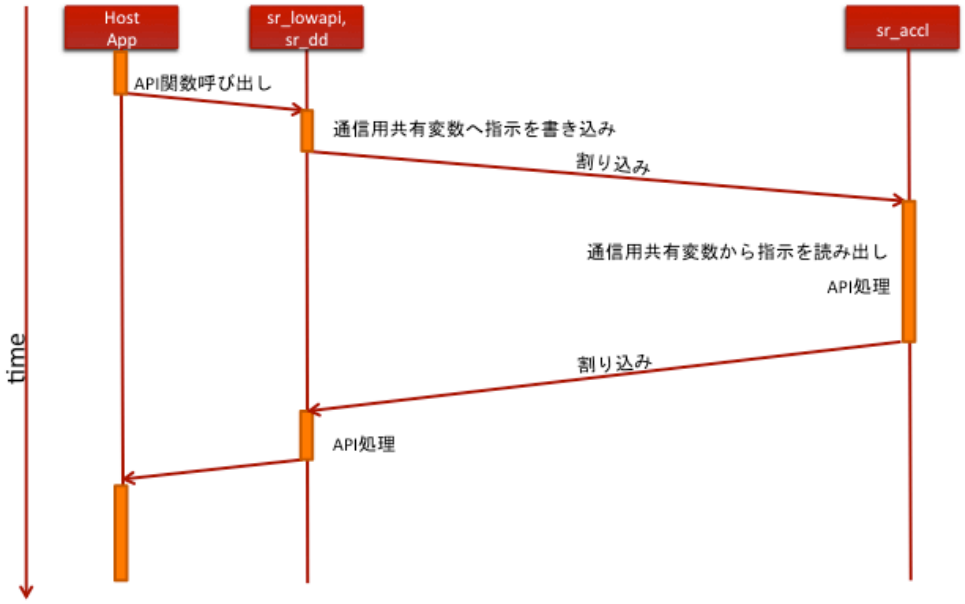


図 2.1-31 ブロッキング関数の実行フロー

API 関数の個別の処理：

初期化／終了

srl_init

処理がホストのみで完結する関数である。初期化 API ではアクセラレータコアのファイルディスクリプタのオープン、VAM 領域の mmap を行う。

srl_close

処理がホストのみで完結する関数である。終了 API では VAM 領域のアンマップとアクセラレータコアのファイルディスクリプタのクローズを行う。

実行関連

srl_run

ノンブロッキング関数である。アクセラレータコアはホストから指定された物理アドレスへと処理を移す。

srl_suspend

ブロッキング関数である。srl_suspend は一時停止した時のプログラムカウンタの値をホスト側に返す必要があるため、ブロッキング関数として実装する必要がある。アクセラレータコアは一時停止した時のプログラムカウンタの値（CP0 の EPC レジスタの値）を記憶すると同時に通信用共有変数に書き込む。一時停止用のビジーループへと処理を移す。

srl_restart
ノンブロッキング関数である。アクセラレータコアは srl_suspend の際に記憶したプログラムカウンタの値に処理を移す。

srl_abort
ノンブロッキング関数である。アクセラレータコアはモニタプログラムのエントリポイントへと処理を移す。

問い合わせ
srl_isActive
処理がホストのみで完結する関数である。指定されたコアとの通信用共有変数からアクセラレータコアの状態を読み出し、Running であるか否かをチェックする。

srl_isSuspended
処理がホストのみで完結する関数である。指定されたコアとの通信用共有変数からアクセラレータコアの状態を読み出し、Suspended であるか否かをチェックする。

srl_isStopped
処理がホストのみで完結する関数である。指定されたコアとの通信用共有変数からアクセラレータコアの状態を読み出し、Stopped であるか否かをチェックする。

srl_isBusy
処理がホストのみで完結する関数である。指定されたコアとの通信用共有変数からアクセラレータコアの状態を読み出し、Busy であるか否かをチェックする。

SPM アクセス（予定）
srl_read_spm
ブロッキング関数である。アクセラレータコアは指定された SPM オフセットアドレスからデータを読み出し、一旦 VAM 領域にコピーする。コピーが完了すると、ホストに対して割り込みをかける。割り込みを受けたホスト側は、VAM 領域からユーザ領域へとデータをコピーする。

srl_write_spm
ノンブロッキング関数である。まず、ユーザ領域のデータを VAM 領域へとデータをコピーしたのち、アクセラレータコアに対して VAM 領域上のデータを SPM にコピーするよう指示を出し、割り込み待ちに入る。
指示を受けたアクセラレータコアは、ホストに割り込みをかけた後、VAM 上のデータを SPM へとコピーする。

L1 キャッシュ設定（予定）
srl_set_l1_config
ブロッキング関数である。アクセラレータコアではホストによって指定された設定を L1 キャッシュに施す。

srl_get_l1_config
ブロッキング関数である。アクセラレータコアは L1 キャッシュの設定内容を通信用共有変数に書き込む。

VAM 領域のアドレスマップ
VAM 領域のアドレスマップは図 2.1-32 のとおりである。

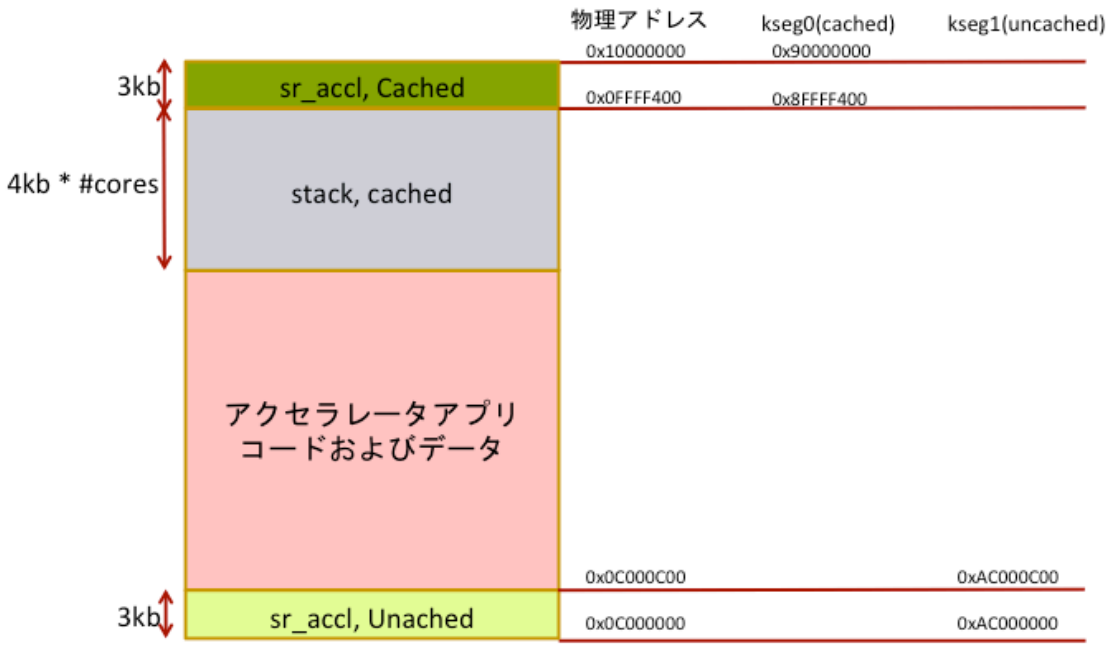


図 2.1-32 VAM 領域のアドレスマップ

sr_accl の主要なラベル、アドレス

sr_accl モジュール（モニタプログラム）の主要なラベル、アドレスを一覧する。

ラベル	アドレス	用途
_entry	0x8FFF6000 (kseg0)	エントリポイント
intr_hndl	0x8FFF6040 (kseg0)	割り込みハンドラ
comm_mem	0xAC000000 (kseg1)	通信用共有変数

2.2 プログラミング・フレームワーク SMYLE OpenCL

2.2.1 プログラミングモデル

SMYLEref 用のプログラミングモデルとして、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。

- 仕様がオープンかつロイヤルティフリーである
- 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
- 既に標準化されたプログラミングモデル/言語を利用する方が、全く新しいモデル/言語を開発するよりも、本研究成果の普及が期待される

図 2.2-1 に、OpenCL が想定するアーキテクチャモデルを示す。

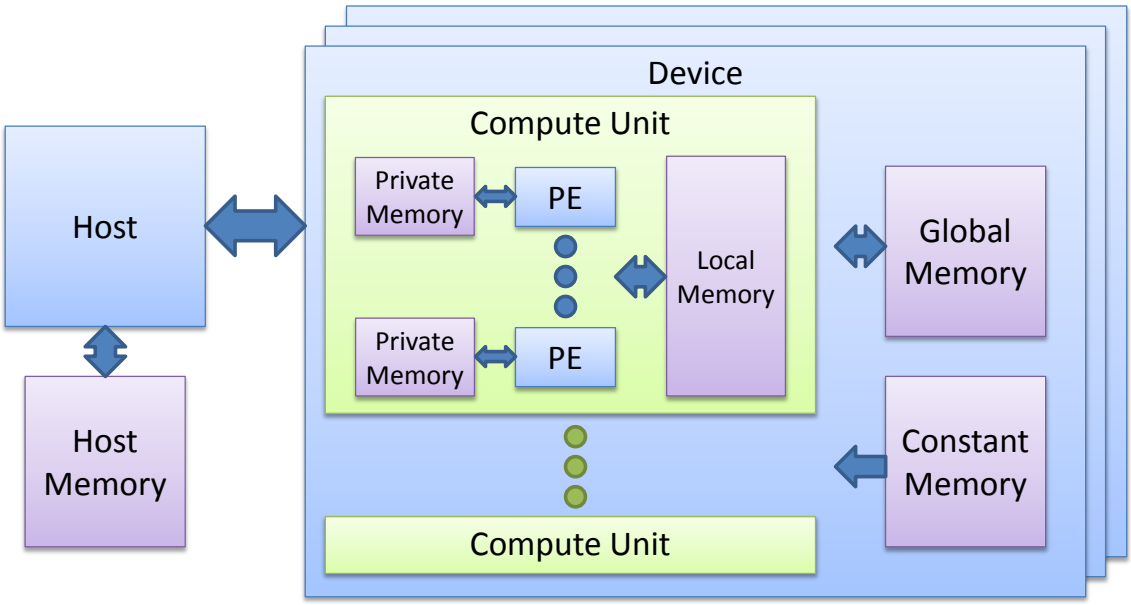


図 2.2-1　OpenCL のアーキテクチャモデル

OpenCL のアーキテクチャモデルは、全体の制御を行うホストと、高速に並列計算を行うデバイスから構成される。例えば、CPU と GPU を搭載したパソコン上で OpenCL を実行する場合、CPU がホスト、GPU がデバイスに対応する。デバイスは 1 個以上の Compute Unit から構成され、Compute Unit は 1 個以上の Processing Unit (PE)から構成される。GPU の場合、個々の GPU コアが PE に対応し、GPU コアのクラスタ (Nvidia 社 GPU における Streaming Multiprocessor, SM) が Compute Unit に対応する。

SMYLEref アーキテクチャの場合、コアが、OpenCL の PE に対応する。SMYLEref は、物理

的にはコアがクラスタ化されている。しかしながら、SMYREref のクラスタが、OpenCL の Compute Unit に対応する訳ではない。SMYLEref のクラスタの境界に関わりになく、任意の数のコアを Compute Unit として使用することができる。

SMYLEref アーキテクチャは多くのコアを有しているが、図 2.2-2 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。

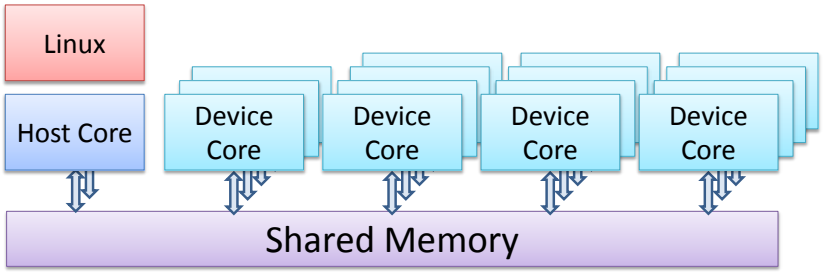


図 2.2-2　SMYLE OpenCL における SMYLEref のコアの使い方

図 2.2-1 に示した通り、OpenCL では複数の異なるメモリが定義されており、それぞれアクセス可能な範囲（ホスト、デバイス、Compute Unit、PE）が異なっている。

先述のように、OpenCL は、データ並列実行とタスク並列実行の両方をサポートしている。図 2.2-3 に、OpenCL におけるデータ並列実行を示す。

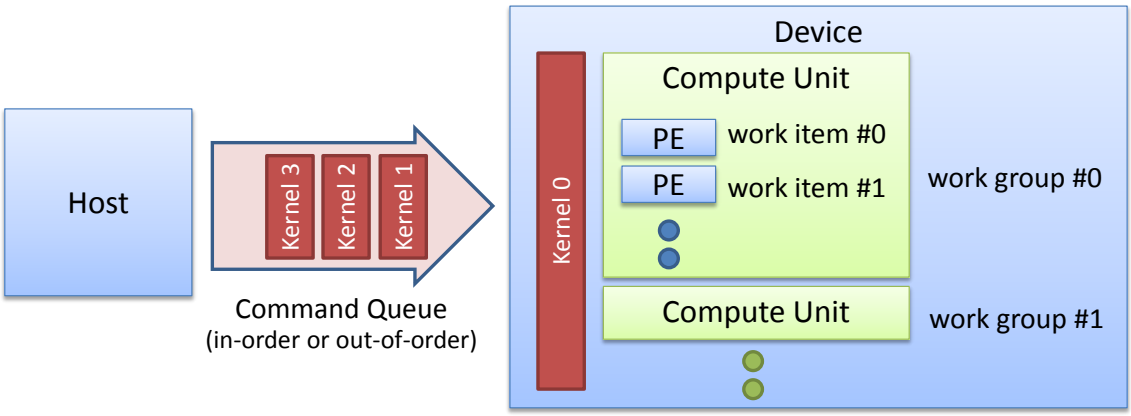


図 2.2-3　OpenCL におけるデータ並列実行

OpenCL では、デバイスで実行されるプログラム（関数）をカーネルと呼ぶ。カーネルは、コマンドキューを通じてホストからデバイスに送られ、デバイスで実行される。データ並列実行では、同一のカーネルが、デバイスの複数の Compute Unit 上で実行される。このとき、処理されるデータ（配列）は複数のワークグループに分割され、各 Compute Unit がひ

とつのワークグループを処理する。Compute Unit 内で、ワークグループは複数のワークアイテムに分割され、各 PE がひとつのワークアイテムを処理する。

図 2.2-4 に、OpenCL におけるタスク並列実行を示す。

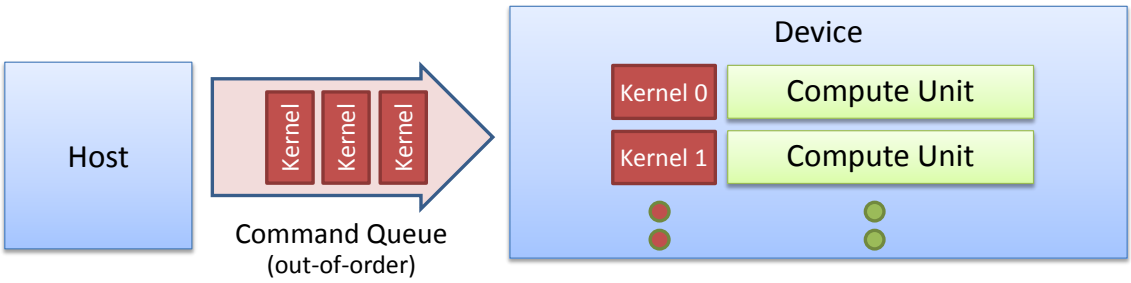


図 2.2-4　OpenCL におけるタスク並列実行

ホストが、複数の異なるカーネルを、アウトオブオーダー方式でコマンドキューに投入することにより、タスク並列実行が行われる。タスク並列実行では、異なるカーネルが異なる Compute Unit 上で実行される。同一 Compute Unit 内の PE は、異なるカーネルを実行することはできない。タスク並列実行は Compute Unit 単位で行われる。

現在までに、GPU や汎用マルチコア/マルチプロセッサを対象とした OpenCL 環境はいくつも開発されている。例えば、NVIDIA 社や AMD 社は、自社の GPU を対象とした OpenCL 処理系を無償で公開している。また、Intel 社やフィックスターズ社は、Intel 社製の汎用マルチコア/マルチプロセッサを対象とした OpenCL 処理系を公開している。しかし、これらの既存の OpenCL 環境を組込みシステムに適用しようとする場合、以下の問題点がある。

- GPU 向けの OpenCL 環境の多くは、真のタスク並列実行をサポートしていない。プログラミングモデルとしてはタスク並列実行をサポートしているが、実際には、タスクを逐次的に実行している。
- GPU 向けの OpenCL 環境は、真のタスク並列実行をサポートしている場合でも、タスク並列実行の単位がハードウェアアーキテクチャのクラスタ数に依存しており、システム上のコアを有効活用できない。例えば、1 クラスタあたりのコア数が 16 コアのハードウェアアーキテクチャにおいて、あるタスクが 1 コアしか使用しない場合、そのタスクが割り当てられるクラスタ内の残りの 15 コアは使用されない。
- GPU 向けの OpenCL 環境は、アプリケーション内のタスク並列実行をサポートしている場合でも、複数のアプリケーションの並列実行をサポートしていない。同時に実行するアプリケーションは唯一であり、実行中のアプリケーションがシステム上のすべての PE を占有する。
- GPU 向けおよび汎用マルチコア/マルチプロセッサ向けの OpenCL 環境は、実行時の性能オーバーヘッドが大きい。

一方、本プロジェクトで開発した SMYLE OpenCL は、上記の問題を解決している。

- 真のタスク並列実行をサポートしている。複数の異なるカーネルを、空間的に並列に実行する。
- タスク並列実行の場合、各タスクに割り当てられるコア数が、ハードウェアアーキテクチャに依存しない。ハードウェアアーキテクチャのクラスタ構造に関わらず、各タスクは、システム上の総コア数を超えない限り、任意の数のコアを使用することができる。
- アプリケーション内のデータ並列実行とタスク並列実行だけでなく、複数のアプリケーションの並列実行をサポートしている。
- 既存の OpenCL 環境では実行時に行っている処理の多くを設計時に行うことで、実行時の性能オーバーヘッドを最小化している。

SMYLE OpenCL では、図 2.2-5 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。

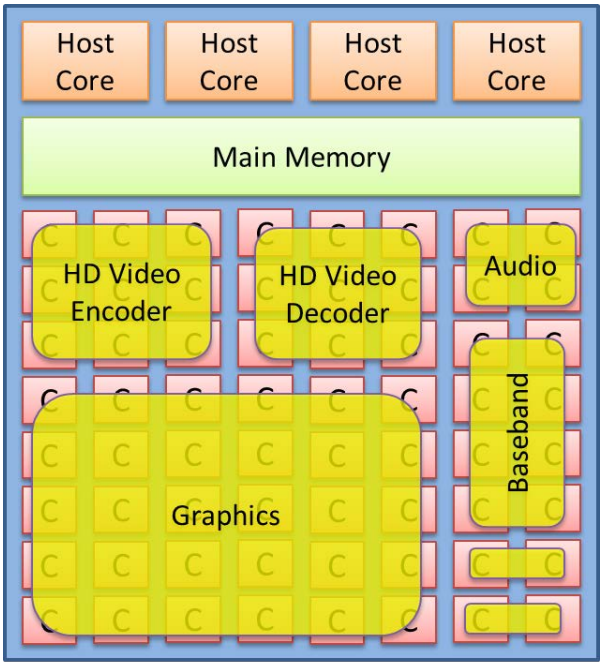


図 2.2-5　SMYLE OpenCL における複数アプリケーションの並列実行

OpenCL プログラムの典型な流れは以下の通りである。

1. デバイス情報を取得、デバイスを確保

2. コンテキストを生成
3. コマンドキューやメモリバッファを生成し、コンテキストに格納
4. カーネルをビルドし、コンテキストに格納
5. 入力データをメモリバッファに格納
6. カーネルを実行
7. 実行結果をメモリバッファから読み出し
8. コマンドキュー、メモリバッファ、コンテキストなどを解放

GPU や汎用マルチコア/マルチプロセッサを対象とした既存の OpenCL 環境では、上記の処理をすべて実行時に行う。コンテキストや各種オブジェクトを動的に生成するイメージ図を図 2.2-6 に示す。

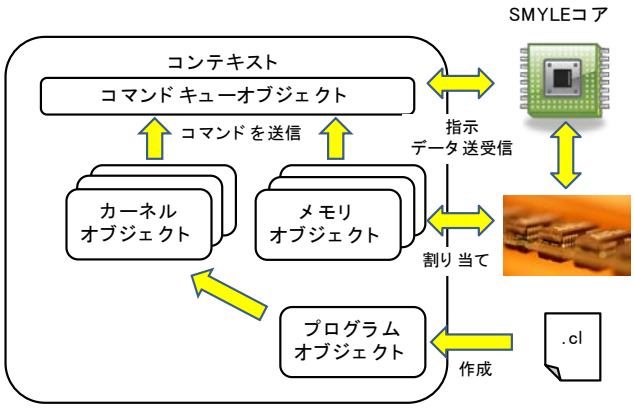


図 2.2-6　コンテキストや各種オブジェクトの動的生成

SMYLE OpenCL では、下記のように、5、6、および、7 のみを実行時に行う。

~~1. デバイス情報を取得、デバイスを確保~~
~~2. コンテキストを生成~~
~~3. コマンドキューやメモリバッファを生成し、コンテキストに格納~~
~~4. カーネルをビルドし、コンテキストに格納~~
5. 入力データをメモリバッファに格納
6. カーネルを実行
7. 実行結果をメモリバッファから読み出し
~~8. コマンドキュー、メモリバッファ、コンテキストなどを解放~~

SMYLE OpenCL では、1～4、および、8 は設計時に行う。つまり、コンテキストやコマンドキューなどのオブジェクトは設計時に作成しておき、製品の出荷時にメモリに書き込んでおく（図 2.2-7 参照）。また、各タスク（カーネルやアプリケーション）を実行するコアを決定するタスクマッピング処理も設計時に行う。

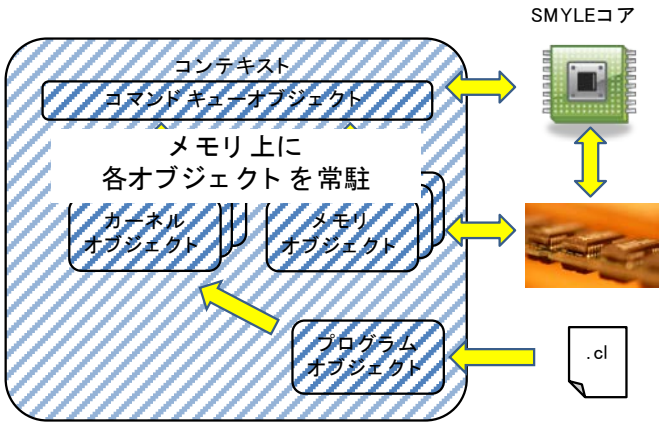


図 2.2-7　SMYLE OpenCL におけるコンテキストや各種オブジェクトの静的生成

図 2.2-8 に、SMYLE OpenCL 環境の全体像を示す。

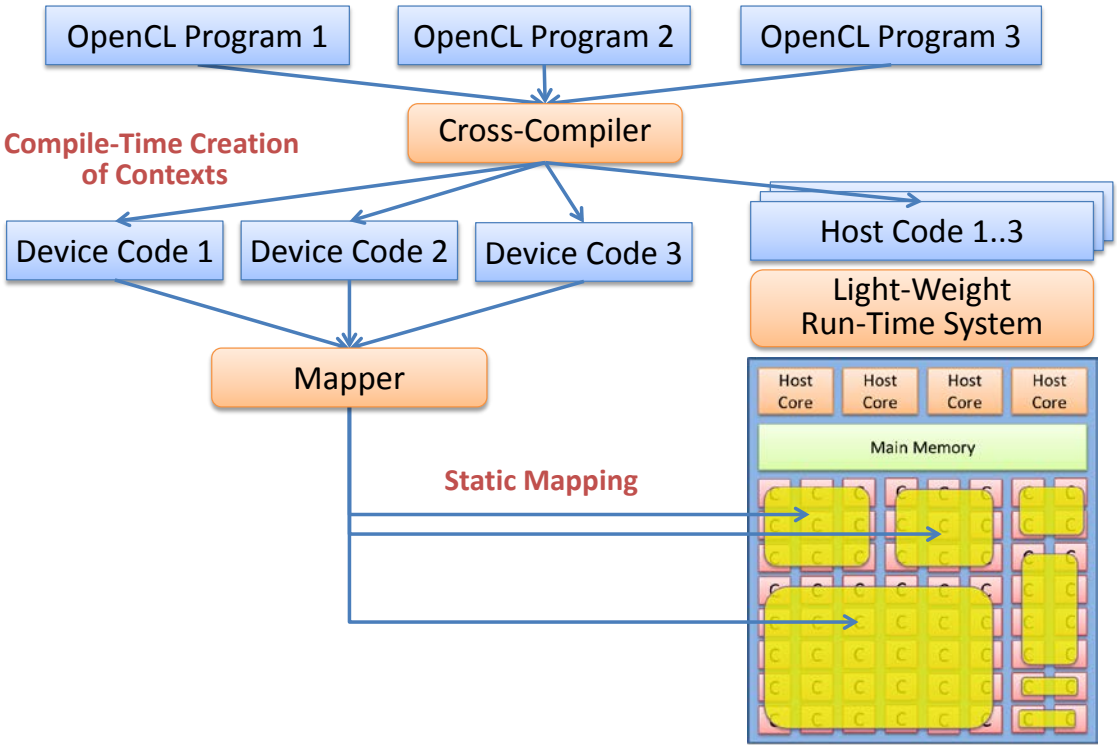


図 2.2-8　SMYLE OpenCL の全体像

現在の SMYLEref 向け SMYLE OpenCL 環境においては、クロスコンパイラは、GNU C コンパイラをそのまま使用している。そのため、SMYLE OpenCL 環境は、OpenCL 言語独自のデータ型や構文はサポートしていない。例えば、SMYLE OpenCL 環境においては、プログラマは、

OpenCL 言語におけるベクタ型を使用することができない。

ランタイムライブラリとマッパについては、2.2.2 節と 2.2.3 節でそれぞれ説明する。

2.2.2 ランタイムライブラリ

SMYLE OpenCL のランタイムライブラリを新規に開発した。ランタイムライブラリは、

- ホスト用のランタイムライブラリ（関数群）
- デバイス用のランタイムライブラリ（関数群）

に大別される。SMYLE OpenCL では、OpenCL の仕様が定義している関数の中で、特に重要な物のみを実装している。現時点において SMYLE OpenCL が実装しているホスト側の関数の一覧を、表 2.2-1 と表 2.2-2 に示す。

表 2.2-1 SMYLE OpenCL が実装しているホスト側の関数一覧

関数名	機能
clCreateContext	デバイスを指定してコンテキストを作成する
clRetainContext	コンテキストのリファレンスカウントを増やす
clReleaseContext	コンテキストのリファレンスカウントを減らす
clCreateCommandQueue	コマンドキューを作成する
clRetainCommandQueue	コマンドキューのリファレンスカウントを増やす
clReleaseCommandQueue	コマンドキューのリファレンスカウントを減らす
clCreateBuffer	バッファメモリオブジェクトを作成する
clRetainMemObject	メモリオブジェクトのリファレンスカウントを増やす
clReleaseMemObject	メモリオブジェクトのリファレンスカウントを減らす
clCreateProgramWithBinary	カーネルバイナリからプログラムオブジェクトを作成する
clRetainProgram	プログラムオブジェクトのリファレンスカウントを増やす
clReleaseProgram	プログラムオブジェクトのリファレンスカウントを減らす
clBuildProgram	プログラムオブジェクトをビルドする
clCreateKernel	カーネルオブジェクトを作成する

表 2.2-2　SMYLE OpenCL が実装しているホスト側の関数一覧（続き）

関数名	機能
clRetainKernel	カーネルオブジェクトのリファレンスカウントを増やす
clReleaseKernel	カーネルオブジェクトのリファレンスカウントを減らす
clSetKernelArg	カーネルの引数を設定する
clWaitForEvents	イベントに対応付けられたコマンドが終了するのを待つ
clRetainEvent	イベントオブジェクトのリファレンスカウントを増やす
clReleaseEvent	イベントオブジェクトのリファレンスカウントを減らす
clFlush	コマンドキュー内の全コマンドをデバイスに渡す
clFinish	コマンドキュー内の全コマンドをデバイスに渡し、実行の完了を待つ
clEnqueueReadBuffer	バッファメモリオブジェクトからデータを読み込むためのコマンドをキューに入れる
clEnqueueWriteBuffer	バッファメモリオブジェクトにデータを書き込むためのコマンドをキューに入れる
clEnqueueNDRangeKernel	カーネルを実行するためのコマンドをキューに入れる
clEnqueueTask	カーネルを実行するためのコマンドをキューに入れる
clEnqueueWaitForEvents	指定したイベントに対応付けられたコマンドの終了を待つためのコマンドをキューに入れる
clEnqueueBarrier	バリアコマンドをキューに入れる

現時点において SMYLE OpenCL が実装しているデバイス側の関数の一覧を、表 2.2-3 と表 2.2-4 に示す。なお、算術演算関連の関数は、オープンソースである SnuCL 1.2beta を利用している。

表 2.2-3　SMYLE OpenCL が実装しているデバイス側の関数一覧（制御関連）

関数名	機能
get_work_dim	ワークグループの次元を取得する
get_global_size	指定された次元のグローバルワークアイテム数を取得する
get_global_id	指定された次元のグローバルＩＤを取得する
get_local_size	指定された次元のローカルワークアイテム数を取得する
get_local_id	指定された次元のローカルＩＤを取得する
get_num_groups	指定された次元のワークグループ数を取得する
get_group_id	指定された次元のワークグループＩＤを取得する
barrier	ワークグループ内のカーネル間で同期する

表 2.2-4　SMYLE OpenCL が実装しているデバイス側の関数一覧（算術演算関連）

fdim	mad
nextafter	abs_diff
add_sat	clz
hadd	rhadd
clamp	mad24
mad_sat	max
min	mul24
rotate	sub_sat
popcount	step
isequal	isnotequal
isgreater	isgreaterequal
isless	islessequal
islessgreater	isordered
bitselect	select

2.2.3 タスクマッピング

マッピングとは、各デバイス用コードについて、そのデバイス用コードを実行する際に使用するコアを決定する処理である。マッピングは、アプリケーションを最小単位として行うことも可能であるし、カーネルを最小単位として行うことも可能である。本節では、マッピングの最小単位のことを「タスク」と呼ぶ。つまり、タスクは、アプリケーション

を指すかもしれないし、カーネルを指すかもしれない。

以下の 2 種類のタスクマッピング手法を開発した。

- シングルコンテキスト静的タスクマッピング手法
- マルチコンテキスト静的タスクマッピング手法

両手法とも、タスクマッピングは静的に行う。各タスクが実行されるコアは設計時に決定される。動的な負荷分散は行わない。

シングルコンテキスト静的タスクマッピングでは、タスクが排他的にコアを使用する。タスクは複数のコアを使用することはできるが、コアは複数のタスクを実行することができない。実行時にコンテキストスイッチ（タスクの切替え）が発生しないため、リアルタイム性に優れている。しかし、コアの CPU 利用率が低下するため、システム全体のスループット性能が低くなる恐れがある。シングルコンテキスト静的タスクマッピングの例を図 2.2-9 に示す。

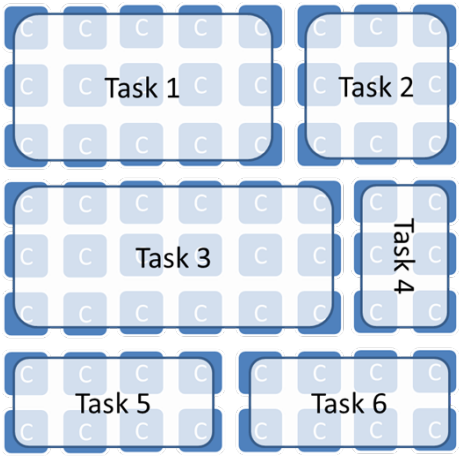


図 2.2-9　シングルコンテキスト静的タスクマッピングの例

タスクマッピングを行う際は、タスク間のタスク並列性だけでなく、各タスク内部のデータ並列性も考慮する必要がある。つまり、データ並列性の高いタスクには多くのコアを割り当てるべきであるし、データ並列性が低いタスクに多くのコアを割り当てるのは得策でない。タスク内部の並列性（コア数の増加に対する性能向上率）は、タスクによって大きく異なる。

図 2.2-10 に、SPLASH-2 ベンチマークの中の 8 つのプログラムについて、コア数を 1 コアから 256 コアまで変化させた場合の性能を示す。コア数が 1 の場合の性能で正規化している。また、本結果は、MIT が開発したオープンソースのサイクル精度のシミュレータである Graphite を使用して測定した。

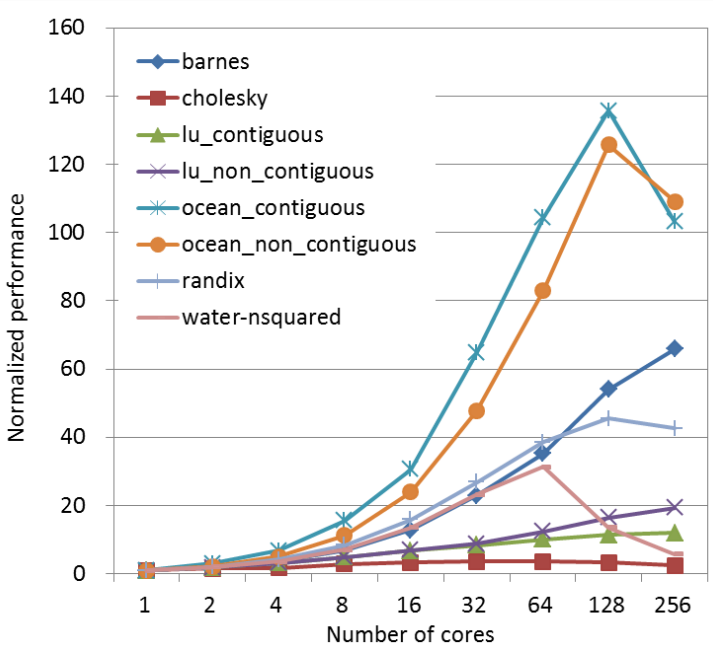


図 2.2-10　SPLASH-2 ベンチマークのスケーラビリティ

図 2.2-10 が示すように、プログラムによって、コア数を増やした場合の性能向上率は大きく異なる。例えば、ocean_contiguous と ocean_non_contiguous は高いスケーラビリティを示しているが、cholesky はほとんどスケーラビリティを有していない。また、water-nsquared など、いくつかのプログラムは、コア数をある程度まで増やすと、性能が低下している。タスクマッピングを行う際は、このようなタスク毎に異なるデータ並列性を活用する必要がある。

シングルコンテキスト静的タスクマッピングを行う手法として、以下の、2 種類の方法を開発した。

- タスクマッピング問題を整数線形計画問題（Integer Linear Programming, ILP）に帰着させ、汎用の数理最適化ソフトウェアを用いて最適解を求める方法
- グリーディアルゴリズムによる方法

まず、整数線形計画問題に基づく手法を説明する。はじめに変数を定義する。map_{ij}は決定変数であり、タスクiがj個のコアにマッピングされる場合に 1 となり、それ以外の時は 0 になる。$gain_{ij}$はタスクiがj個のコアにマッピングされた場合の利益を表す。ここで利益とは、実行効率や電力効率などを総合的に表す指標である。$Ncore$はシステム上のコア数である。

タスクマッピング問題は、以下の制約式(2.2-2)(2.2-3)を満たし、目的関数(2.2-1)を最大化する整数線形計画問題として定式化される。

$$\text{maximize:} \quad \sum_i \sum_j map_{ij} \times gain_{ij} \tag{2.2-1}$$

$$\text{subject to:} \quad \forall i, \quad \sum_j map_{ij} = 1 \tag{2.2-2}$$

$$\sum_i \sum_j j \times map_{ij} \leq Ncore \tag{2.2-3}$$

式(2.2-1)はマッピングされたタスクの利益の総和を表しており、最大化する目的関数である。式(2.2-2)はすべてのタスクが必ずいずれかのコアにマッピングされるという制約を表している。式(2.2-3)は、すべてのタスクが使用するコアの総数は、システム上のコア数を超えることがないという制約を表している。この整数線形計画問題を、汎用の数理最適化ソフトウェアを用いて解くことで、最適なタスクマッピングが得られる。

上記の整数線形計画法に基づく手法によるタスクマッピングは、コア数に対して指数的に計算時間が増加するため、SoC に搭載されているコア数が増えると、マッピングを行うために膨大な計算時間を必要とする。そこで、高速にマッピングを行うグリーディアルゴリズムも開発した。開発したグリーディアルゴリズムを図 2.2-11 に示す。はじめにすべてのタスクの使用コア数を 1 とする。使用コア数を 1 段階大きくしたとき（1 個増やす、2 倍に増やすなど）に得られる利益が最も大きなタスクを選び、そのタスクのコア数を 1 段階増やす。この操作を、タスクが割り当てられていないコアがなくなるまで続ける。

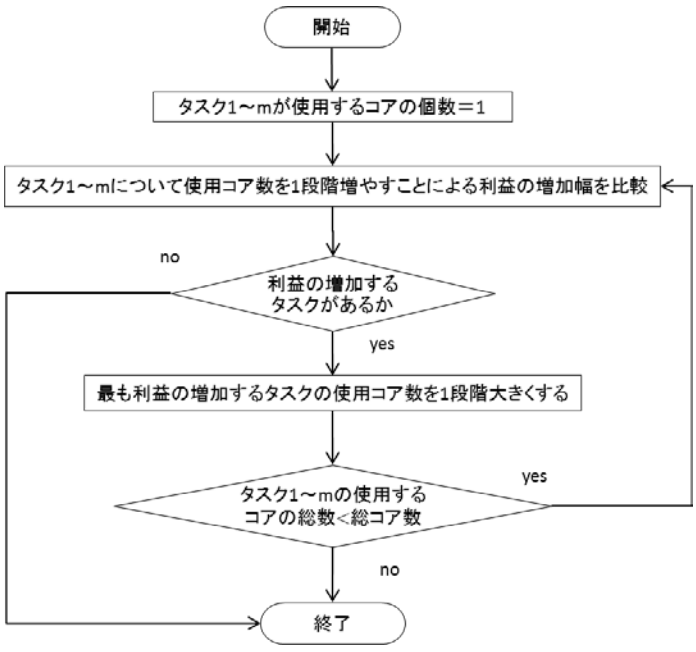


図 2.2-11　グリーディ・タスクマッピング・アルゴリズム

タスク数をm、コア数をnとすると、開発したグリーディアルゴリズムの計算量は$O(nm)$である。整数線形計画問題に基づく手法では最適解を求めるのに対して、グリーディアルゴリズムに基づく手法では最適解の保証はないが、非常に少ない計算量でマッピングを行うことができる。

2 種類のシングルコンテキスト静的タスクマッピング手法について、予備評価を行った。以下の 3 種類のタスクセットを用いた。いずれも、SPLASH-2 ベンチマークに含まれるプログラムである。

- タスクセット 1（4 タスク）：nsquared、ocean_non_contiguous、ocean_contiguous、lu_non_contiguous
- タスクセット 2（4 タスク）：randix、lu_contiguous、cholesky、barnes
- タスクセット 3（8 タスク）：nsquared、ocean_non_contiguous、ocean_contiguous、lu_non_contiguous、randix、lu_contiguous、cholesky、barnes

実験はサイクル精度シミュレータ Graphite 上で行い、各タスクの利益として、図 2.2-10 に示した性能向上率を用いた。

タスクセット 1、2、3 に対する結果を、図 2.2-12、図 2.2-13、図 2.2-14 にそれぞれ示す。

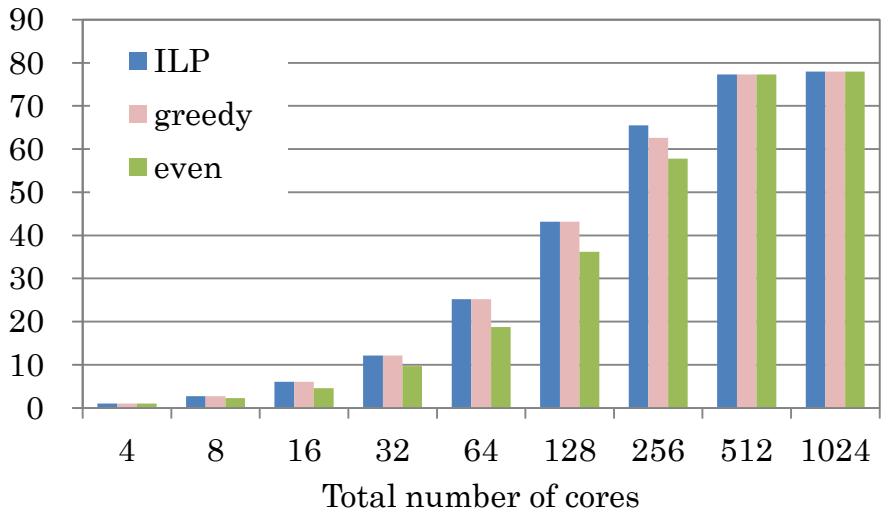


図 2.2-12　タスクセット 1 に対するシングルコンテキスト・マッピング結果

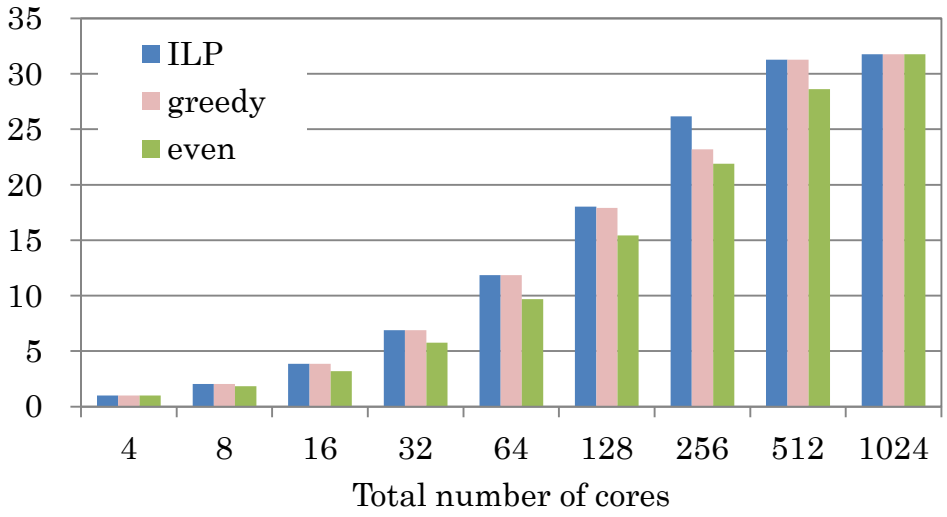


図 2.2-13　タスクセット 2 に対するシングルコンテキスト・マッピング結果

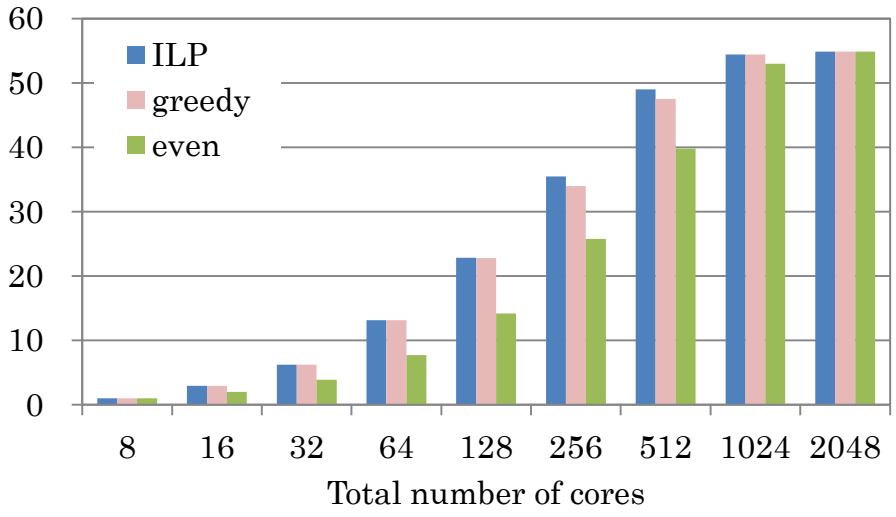


図 2.2-14　タスクセット 3 に対するシングルコンテキスト・マッピング結果

グラフにおいて、横軸はコア数、縦軸は利益の総和（コア数最小の場合で正規化）を表している。ILP が整数線形計画問題に基づく手法（数理最適化ソフトウェアとして、IBM 社 ILOG CPLEX 12.2 を使用）、greedy がグリーディアルゴリズムの結果を表している。even は、タスク毎のデータ並列性の違いを考慮せず、すべてのタスクに均等にコアを割り当てた場合を示している。例えば、コア数が 32、タスク数が 4 の場合、各タスクに 8 コアを割り当てる。even と比較して、開発した 2 つの手法は非常に良い結果（最大 1.7 倍）を示している。また、多くの場合について、グリーディアルゴリズムは、整数線形計画問題に基づく

手法と同等の結果が得られている。

シングルコンテキスト静的タスクマッピング手法は、実行時にコンテキストスイッチが不要なため、高いリアルタイム応答性を実現することができるが、コアの CPU 利用率が低下し、システム全体の平均スループットが低下する恐れがある。そこで、この問題を解決するため、マルチコンテキスト静的タスクマッピング手法を開発した。

図 2.2-15 に、シングルコンテキスト・タスクマッピングとマルチコンテキスト・タスクマッピングの例を示す。

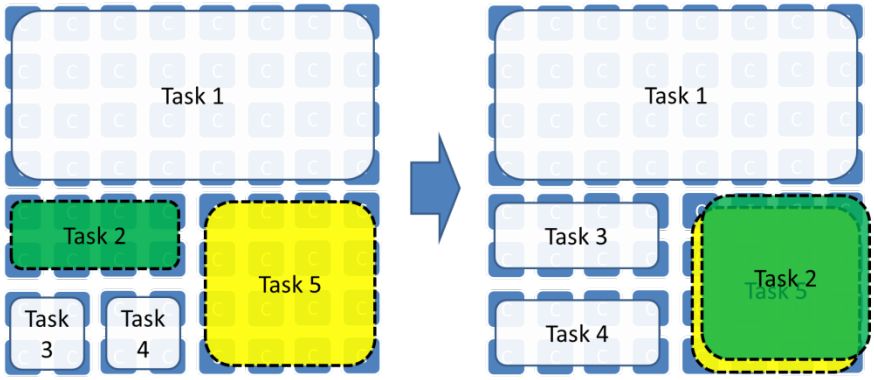


図 2.2-15　シングルコンテキスト・タスクマッピング（左図）とマルチコンテキスト・タスクマッピング（右図）

図 2.2-15 において、タスク 2 とタスク 5 は、決して同時に実行されることがないと仮定する。その場合でも、シングルコンテキスト・タスクマッピングでは、タスク 2 とタスク 5 に異なるコアを割り当てる。一方、マルチコンテキスト・タスクマッピングでは、タスク 2 とタスク 5 に同じコアを割り当てる。これにより、タスク 2 が使用するコア数が増加し、性能が向上する。さらに、副次的な効果として、タスク 3 とタスク 4 にもより多くのコアを割り当てることができるようになり、さらに性能が向上する。

開発したマルチコンテキスト静的タスクマッピング手法は、整数線形計画問題に基づいている。

同時に実行される可能性があるタスクは、そのマッピングが重複してはならない。例えば、図 2.2-16 のようにコア 1～2 に Task1 がマッピングされている時に。Task2 をコア 1～4 にマッピングすることはできない。

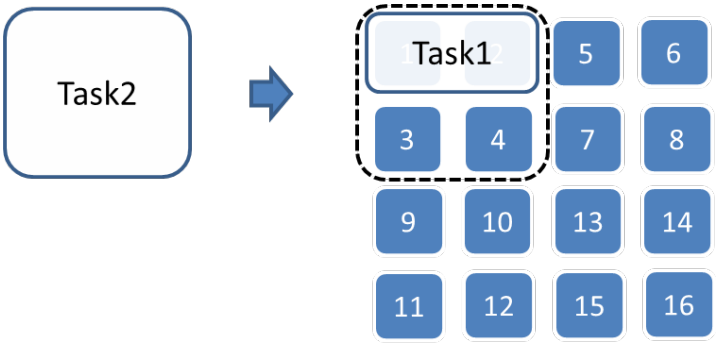


図 2.2-16　マッピングの重複の例

また、単化のため、タスクは必ず隣り合う2^n個のコアにマッピングされるものとする。

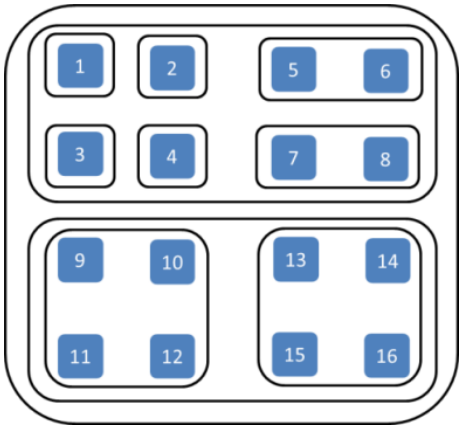


図 2.2-17　マッピングの制限

図 2.2-17 のような 16 個のコアを持つアーキテクチャを考える場合、1 つのタスクがマッピングされる方法は、1 つのコアを使用する方法が 16 通り、2 つのコアを使用する方法が 8 通り、4 つのコアを使用する方法が 4 通り、8 つのコアを使用する方法が 2 通り、16 個すべてのコアを使用する方法が 1 通りで合計 31 通りある。2^n個のコアを持つメニーコア SoC に 1 つのタスクがマッピングされる方法は$2^{n+1}-1$通りである。

本章で扱うタスクマッピング問題は、SoC に搭載されているコア数、各タスクが2^n個のコアを使用して実行された場合の利益、タスク衝突の情報、マッピング重複の情報を入力とし、タスクの利益を最大化するマッピングを求める。

まず、決定変数map_{ij}を再定義する。先述の通り、2^n個のコアを持つメニーコア SoC に 1 つのタスクがマッピングされる方法は$2^{n+1}-1$通りである。この$2^{n+1}-1$通りの個々のマッピング方法をjで識別し、タスクiがj番目の方法でマッピングされる場合に$m_{ij}=1$とし、それ以外の時は 0 とする。

マルチコンテキスト静的タスクマッピング問題は、以下の制約式(2.2-5)(2.2-6)を満たし、目的関数(2.2-4)を最大化する整数線形計画問題として定式化される。

$$\text{maximize:} \quad \sum_i \sum_j map_{ij} \times gain_{ij} \tag{2.2-4}$$

$$\text{subject to:} \quad \forall i, \sum_j map_{ij} = 1 \tag{2.2-5}$$

$$\forall i, interference_{i1i2} = 1 \rightarrow map_{i1j1} \times overlap_{j1j2} = 0 \text{ or } map_{i2j2} \times overlap_{j1j2} = 0 \tag{2.2-6}$$

$overlap_{j1j2}$はマッピングの重複を表す関数であり、マッピング$j1$とマッピング$j2$が重複する時に 1 となる。表 2.2-5 に 4 コアの場合の$overlap$関数を示す。ここで、jは以下のマッピングを意味している。

- $j = 1$：コア 1 を使用
- $j = 2$：コア 2 を使用
- $j = 3$：コア 3 を使用
- $j = 4$：コア 4 を使用
- $j = 5$：コア 1 と 2 を使用
- $j = 6$：コア 3 と 4 を使用
- $j = 7$：コア 1～4 を使用

表 2.2-5　4 コアの$overlap$関数

	j2=1	2	3	4	5	6	7
j1=1	1				1		1
2		1			1		1
3			1			1	1
4				1		1	1
5	1	1			1		1
6			1	1		1	1
7	1	1	1	1	1	1	1

表の要素が 1 のとき、マッピングの重複があることを意味しており、同時に実行されるタスク同士は重複するマッピングを行うことはできない。例えば、表中の灰色で示した要素はコア 1~4 を使用するマッピングとコア 1~2 を使用するマッピングが重複することを表しており、あるタスクがコア 1~4 にマッピングされている時、他のタスクをコア 1~2 にマ

ッピングすることはできない。

$interference_{i1i2}$はタスクの衝突を表す関数であり、タスク$i1$とタスク$i2$が同時に実行される可能性がある場合に 1 となる。表 2.2-6 に$interference$関数の例を示す。

表 2.2-6　タスクの衝突を表す$interference$関数の例

	Task 1	Task 2	Task 3	Task 4
Task 1	0	1	0	0
Task 2	1	0	0	1
Task 3	0	0	0	1
Task 4	0	1	1	0

$interference$関数の値は、タスクが同時に実行される可能性がある場合に 1 となる。例えば、表 2.2-6 に示した$interference$関数において Task 2 に注目すると、Task 1 と Task 4 が同時に実行される可能性があることを表している。同時に実行されるタスク同士のマッピングは重複してはならない。

式(2.2-6)はタスク$i1$、$i2$が同時に実行される可能性がある場合に2つのタスクのマッピング$j1$、$j2$が重複してはならないという制約を表している。$interference_{i1i2}$が 1 かつ$overlap_{j1j2}$が 1 の時に決定変数map_{i1j1}あるいはmap_{i2j2}は 0 にならなければならない。

開発したマルチコンテキスト静的タスクマッピング手法の予備評価を行った。使用したベンチマークは先述のタスクセット 3 であり、利益も同じものを用いた。タスク間の衝突を表す$interference$関数の値はランダムに決定した。その際、関数が 1 となる確率dを、d =0%、25%、50%、75%、100%と変化させた。d =0%の場合は、同時に実行されるタスクは唯一であることを意味している。この場合、各タスクがすべてのコアを占有することができる。d =100%の場合は、すべてのタスクが同時に実行される可能性があることを意味している。つまり、先述のシングルコンテキスト静的タスクマッピングと同じである。コア数を 8、16、32、64、128、256 と変化させ、マッピングを行った。マッピングを行った結果を図 2.2-18 に示す。

搭載コア数に関わらず、並列に実行しなければならないタスクの組み合わせが多い（つまりdの値が大きい）ほど、全体的な性能が低下している。このことは、タスク間の依存関係を解析し、同時に実行されることのないタスクの組み合わせを求めることで、システムの全体的な性能を向上させることが可能であることを示唆している。

また、256 コアへのマッピングにおいて、d =50%以上の場合、マシンのメモリ不足により求解することができなかった。

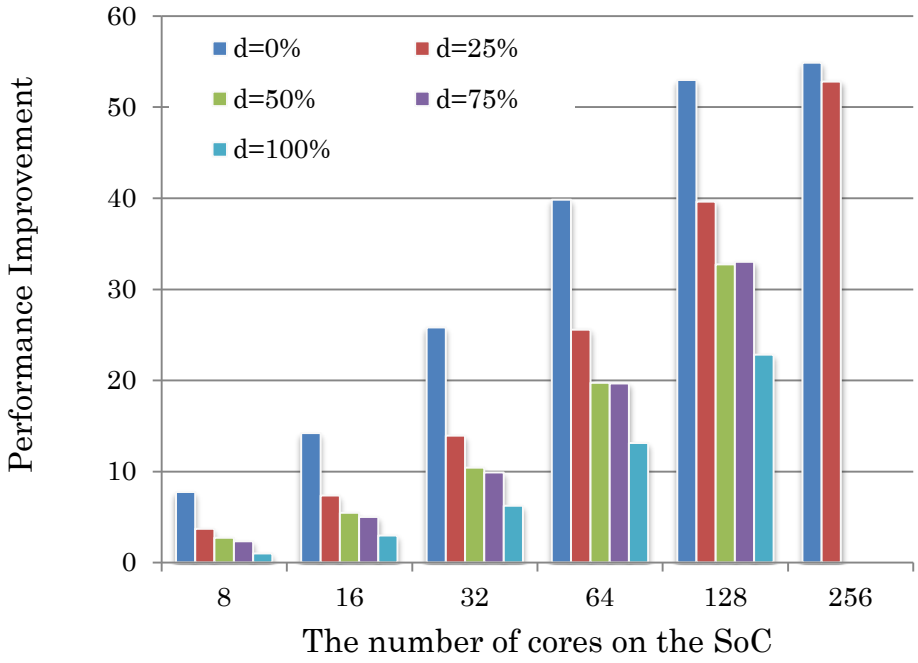


図 2.2-18　マルチコンテキスト・タスクマッピングの結果

2.2.4 機能シミュレータ

開発した SMYLE OpenCL 環境は SMYLEref アーキテクチャをターゲットとしており、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。しかし、FPGA プロトタイプ・システムは高価であり、また、デバッグ容易性（可観測性、可制御性）が低い。そこで、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 機能シミュレータを開発した。本機能シミュレータは Linux が稼働するホスト計算機上で動作する。本機能シミュレータは、SMYLEref アーキテクチャ上で SMYLE OpenCL プログラムを実行した場合の機能的な振る舞いのみをシミュレーションする。SMYLEref アーキテクチャ上でプログラムを実行した場合の実行時間や消費電力を見積もることはできない。

本機能シミュレータの構成を図 2.2-19 に示す。

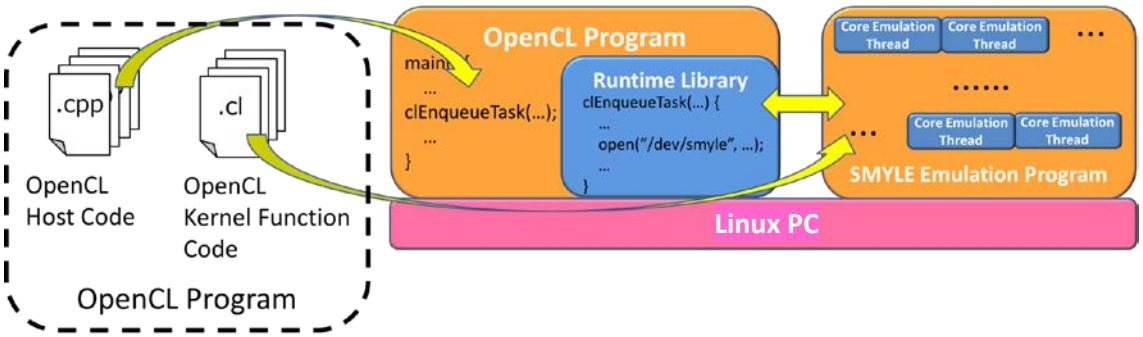


図 2.2-19　SMYLE OpenCL/SMYLEref 機能シミュレータの構成

本機能シミュレータでは、個々のデバイスコアを pthread でシミュレーションする。つまり、デバイスコアが 64 コア存在する場合、機能シミュレータ内部で 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実行の両方をサポートしている。

本機能シミュレータを用いて、Grayscale と Gaussian の 2 つの OpenCL カーネルをタスク並列実行したときの実行時間を図 2.2-20 に示す。

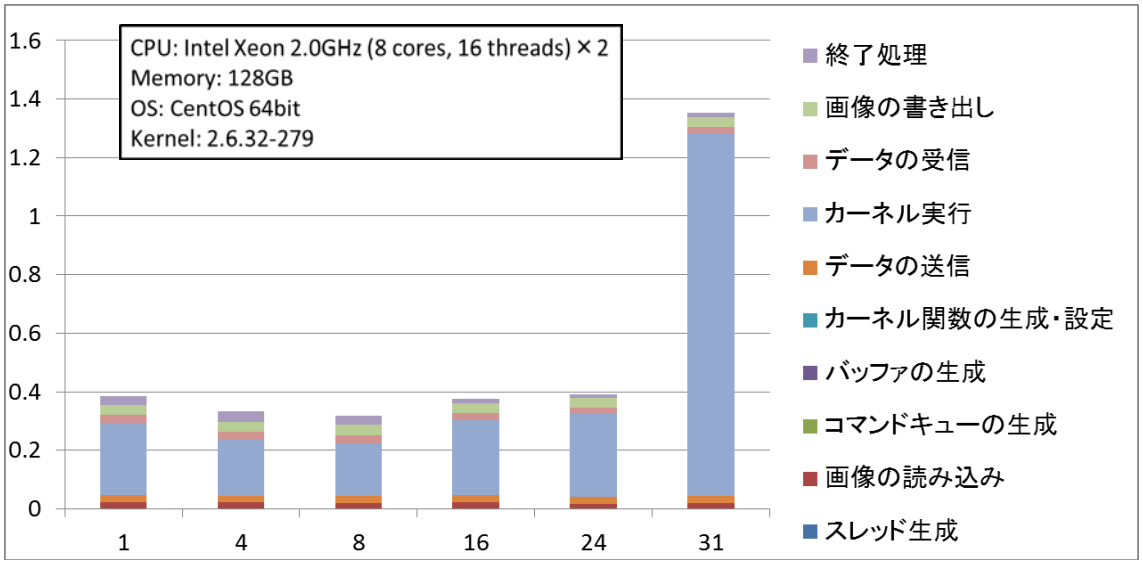


図 2.2-20　機能シミュレータにおけるタスク並列実行

縦軸は実行時間（単位は秒）であるが、これは、ホスト計算機（Intel Xeon、32 論理コア、2GHz）上で実行したときの実行時間であり、SMYLEref アーキテクチャ上で実行した場合の実行時間ではない。本実験では、デバイスコアが 32 コアであることを想定している（つまり、内部で 32 スレッドを生成している）。横軸の数字は、32 コアのうち、Grayscale カーネルが割り当てられたコア数を示している。つまり、横軸の数字が 1 のときは、32 コア

のうちの 1 コアを Glayscale カーネルが使用し、残りの 31 コアを Gaussian カーネルが使用していることを意味している。Gaussian カーネルの方が、Grayscale カーネルよりも計算量が多いため、Gaussian に多くのコアを割り当てた方が全体の実行時間が短縮される傾向にあることが分かる。

実行時間は、あくまでホスト計算機上の実行時間であり、SMYLEref アーキテクチャ上で実行した場合の実行時間ではないが、ソフトウェア開発者が OpenCL プログラムを開発、デバッグ、解析するうえでは、非常に有用である。

2.3 メニーコア FPGA プロトタイプ・システム

2.3.1 概要

メニーコアのアーキテクチャやソフトウェアを検討・開発していくにあたり、その評価環境の構築は重要な課題である。従来、多くのアーキテクチャ研究においては機能レベル、あるいはサイクルレベルのソフトウェアシミュレータが評価環境として使用されてきた。しかし、一般的に多くのソフトウェアレベルシミュレータは、複数プロセッサ上で並列・並行プログラムをシミュレーションする場合でも、逐次的にシミュレーションをすることが多く、コア数が数 10 や数 100 にもなるメニーコアプロセッサを評価する場合、プログラムの一部を評価するだけでも長時間を要する。さらに、OS などのシステムソフトウェアの動作も評価を行うとなると、より長いシミュレーション時間がかかってしまう。

また、他の選択肢として実際に LSI を設計・試作して評価を行うことも考えられる。これにより、並列プログラムなどの評価を実時間で行うことが可能になり、また評価精度も高いという利点がある。その一方、設計・試作のコストが高く、また後から部分的にアーキテクチャ設計を変更したい場合でも簡単には変更や修正ができず、評価を行うことは難しいというように、その柔軟性に問題がある。

その他の選択しとして、近年では FPGA を用いたプロトタイピングが評価環境として有用性を増している。実際に多数のコアを FPGA 上に実装し、それらを並列に動作させることで、評価に要する時間の短縮が期待できる他、OS やシステムソフトウェアの実行もそのまま評価・検証を行うことができる。また、実際に HDL により設計することで、正確かつ詳細な評価・検証を行うことも可能になる。そのため、ソフトウェアレベルのシミュレータでは動作や実行が抽象化され、その結果隠蔽されてしまうような性能上のボトルネック(例えばコア間の同期など)も正確に評価を行うことができ、特にメニーコアプロセッサの評価環境として有用であると考えられる。そこで、SMYLEref の開発にあたっては、FPGA を利用して評価環境を構築することにした。表 2.3-1 に評価環境としての、ソフトウェアシミュレータ、LSI 実装、FPGA によるプロトタイピングの比較をまとめる。

表 2.3-1 評価環境の比較

	Scalability	Accuracy	Flexibility	Development Cost	Evaluation Speed
Software Simulator	Low	Medium	Very High	Low	Low
LSI Implementation	Medium	Very High	Low	Very High	Very High
FPGA Prototyping	Very High	Very High	High	High	High

今回のプロトタイピングでは、Xilinx 社製の FPGA チップである Virtex-6 を搭載する ML605 評価ボードを評価環境のプラットフォームとして利用することにした。ML605 評価ボード、および当該ボードに搭載されている Virtex-6 チップの仕様を以下に示す。

- ML605 評価ボード
  - FPGA デバイス: Virtex-6 XC6VLX240T
  - SDRAM: DDR3 SO-DIMM
  - 搭載 I/O ポート: UART, USB, DVI 出力、CF、SMA 等
  - クロック入力: 200MHz & 66 MHz オシレータ
- Virtex-6 XC6VLX240T
  - 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

回路設計には VerilogHDL を用い、 論理合成、マッピング、配置配線には Xilinx 社の ISE を利用した。図 2.3-1 は本評価環境の外観である。以下、実装の詳細と本評価環境を用いた初期評価実験の結果について述べる。

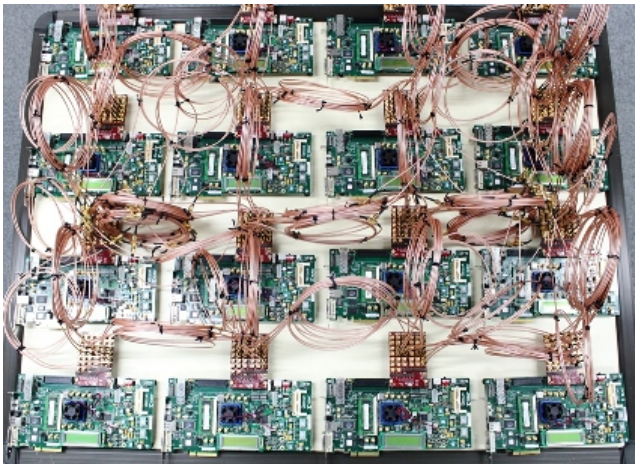


図 2.3-1 評価環境の外観

2.3.2 FPGA 上への実装

クラスタとして、コア・クラスタとペリフェラル・クラスタの 2 種類を構築する必要がある。各コア・クラスタには最大で 8 つの Geyser コアを実装し、2.1.3 節で述べたルータを搭載する。ペリフェラル・クラスタは、SDRAM や UART、SysACE 等のコントローラを備えており、それらは Xilinx 社が提供する PLB (Processor Local Bus) バスに結合される。ペリフェラル・クラスタ上のルータも PLB と接続されており、これによりオンチップネットワークを介して、各コアとペリフェラルが接続されることになる。

評価環境として利用する FPGA (Virtex-6 XC6VLX240T)には Geyser コアを 8 個程度しか搭載することができず、1 つの ML605 ボードだけでは数 10、あるいは数 100 のコアを持つようなメニーコアプロセッサの評価を行うことができない。そこで、複数のボードを結合し、ルータ間の通信をボード間に跨がって行うことでメニーコアの評価環境を実現することにした。

ボード間の通信には高速シリアル通信インタフェースである rocket I/O を使用する。このために、ルータモジュール内にボードを跨がって通信を行う際に使用する通信モジュールを開発した。本モジュールは、ISE の IP コアとして提供されている aurora と呼ばれる高速シリアル通信プロトコルを利用している。なお、物理的なボード間通信インタフェースとして SMA 規格の通信インタフェースを利用する。ML605 評価ボード上は 3 リンク分の SMA コネクタしかないが、FMC の拡張ボードを利用することにより、1 ボードあたり追加で 8 リンク分の SMA コネクタを追加することが可能である。2 次元メッシュネットワークであれば、各方向あたり 2 リンク(双方向で通信のため)の合計 8 リンクがあれば一つのクラスタを構築し、隣接クラスタと結合しつつネットワークを形成することが可能である。したがって、上記の通信インタフェースを用いることで、理論的には無限個のコアを持つメニーコアの評価環境を構築することが可能である。

本評価環境では、プロセッサコアのクロック周波数として数 10MHz 程度を想定している。rocket I/O によるルータ間は 5GHz でシリアルデータの通信が可能であるため、制御情報を含めた 1 パケット(現在は 38bit)を転送するバンド幅は、コアクロックに対して十分なバンド幅を確保できる。一方で、aurora モジュール中でシリアル/ パラレルデータ変換をする際に、クロックの同期を行う必要性から通信レイテンシはルータ同士をボード内で直接接続する場合に比べて約 180ns 程度余分にかかる。そのため、プロセッサコアから見ると通信レイテンシは数サイクル増加してしまう。ただし、例えば I/O アクセスはさらに長いレイテンシがかかるため、この数サイクルのレイテンシ増加は評価結果に大きな影響は与えないと考えられる。

コア・クラスタと周辺機器との接続は PLBv46 Master Burst 通信プロトコルで制御する。PLBv46 Master Burst 通信プロトコルは、Xilinx 社の IP Core の一つであり[PLB2010]、32 ビットの PLB v4.6 バスとユーザーIP コアとの双方向インタフェースを提供する。本プロト

コルは、シングルデータとビート方式の読込み、および書込みスキームを備えている。シングルデータスキームは、一つのアドレスワード(32 ビット)の後一つのデータワードのアクセスが要求され、ビートスキームでは、1 つのアドレスワードに続いて 16 個のデータワードのアクセスが要求される。

2.3.3 本評価環境を用いた基本評価

構築した SMYLEref の FPGA 評価環境を用いて、いくつかの並列プログラムに関して性能評価を行うことで、本評価環境の機能検証をするとともに、メニーコア評価環境としてのスケーラビリティについても検証する。本評価には ML605 ボードを 16 枚使用し、1 つのボード上に 8 個の Geyser コアを持つコア・クラスタを実装した。すなわち、合計 128 コアの評価環境になる。また、ペリフェラル・クラスタは 16 ボードのうちの 1 つに実装した。各機能ブロックへ供給する周波数は以下のように設定した。

- 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

なお、Geyser コアについては本評価環境上で最高 60MHz 程度の周波数で安定動作することを確認している。

SMYLEref を評価するにあたり、まずは通常の並列アプリケーションベンチマークを用いて評価を行った。評価には共有メモリ用のベンチマークである SPLASH2 [Woo1995]ベンチマークの中から FFT、LU プログラムを用いた。SPLASH2 ベンチマークは並列処理のためのプリミティブが m4 マクロにより記述されており、処理系に合わせてマクロを展開することで、実際の並列処理 API などを含むプログラムコードを生成する。本評価では、SPLASH2 ベンチマークに付属されているバリア同期を含む pthread コードを生成するマクロを利用した。

pthread コードを評価するために、SMYLEref 評価環境向けの最低限の機能を持つ簡易版 pthread ライブラリを実装した。本簡易版 pthread ライブラリを用いることで、pthread を用いて並列化された SPLASH2 のコードをほぼそのままコンパイル・実行することが可能である。実装した pthread 関数はスレッド生成・終了を制御する pthread_create、pthread_join などの他、他排他制御用の関数(pthread_mutex_lock、pthread_mutex_unlock など)やバリア同期用の関数(pthread_barrier_wait など)である。

コンパイラは MIPS 用のコードを生成するよう構築した gcc 4.4.6 を用いる。今回評価に用いた FT と LU は浮動小数点演算を含むプログラムである。Geyser に浮動小数点演算器を搭載すると 1 ボードに 8 コアを搭載できない。そこで、浮動小数点演算器なし(1 ボード 8 コアで合計 128 コアまで評価可能)、浮動小数点演算器あり(1 ボード 4 コアで合計 64 コアまで評価可能)の 2 つの構成で評価を行う。浮動小数点演算器なしの場合は gcc の浮動小数点演算のソフトウェアエミュレーション(Soft Float)機能を利用して実行する。

SMYLEref はキャッシュコヒーレンスを前提としたアーキテクチャではなく、当評価環境

でも全コアでのキャッシュコヒーレンス制御は実装していない。一方、SPLASH2 はキャッシュコヒーレントな共有メモリ型の並列計算機が前提であるため、正しい実行結果を得るためにはいくつかの工夫が必要となる。本評価では、共有データ領域(ヒープ領域)をキャッシュ経由でアクセスしない uncachable 領域とし、またバリア同期、および排他制御の度に毎回各コアで L1 データキャッシュをフラッシュすることにした。これにより、プログラム自体を変更することなく、キャッシュされたデータの一貫性を保証することができる。なお、本評価環境ではコア・クラスタ内のコア間のキャッシュコヒーレンスは実装しており、比較評価に用いることができる。

2.3.4 評価結果

まず、実 LSI チップへ実装した場合のある程度の参考になるため、表 2.3-2 に FPGA 上に実装した際の Geyser コア、ルータ等の各モジュールの回路規模を示す。表より、FPU を除くと、コアの面積が他のモジュールに比べて大きいことがわかる。8 コアで 1 つのコア・クラスタを形成することを想定すると、ルータなどの他の機構の面積はほとんどチップ面積には影響しない。このことは、数コアを持つコア・クラスタを構成し、それを NoC で結合するという階層構造がルータ面積オーバーヘッドを削減することに大きく貢献できることを意味している。

表 2.3-2 FPGA 上での回路規模の比較

	Slices (%)	Flip-Flops (%)	LUTs (%)
Core cluster			
Geyser core	3,301 (8.76%)	7,089 (2.35%)	10,942 (7.26%)
L2 cache	2,853 (7.57%)	6,232 (2.07%)	8,813 (5.85%)
Router	1,170 (3.11%)	838 (0.28%)	3,400 (2.26%)
MEM access controller	201 (0.58%)	338 (0.11%)	533 (0.35%)
Board comm. controller	1,257 (3.33%)	3,059 (1.01%)	2,775 (1.84%)
FPU	3,703 (8.80%)	15,968 (5.21%)	15,526 (10.30%)
Peripheral cluster			
I/O controller	1,596 (4.24%)	6,007 (1.99%)	2,577 (1.71%)
Router	1,170 (3.11%)	838 (0.28%)	3,400 (2.26%)
Packet controller	147 (0.39%)	168 (0.05%)	240 (0.16%)

図 2.3-2 に、各プログラムを 1 コアのみを用いて実行した場合に対する複数コアで実行した場合のスピードアップの結果を示す。図中 FFT、LU は浮動小数点演算器を用いない場合を各プログラムの結果を示し、FFT_fpu、LU_fpu は浮動小数点演算器を用いた場合の結果を示している。なお、各プログラムで浮動小数点演算器を用いない場合の 1 コアを基準にしている。ヒープデータ領域が uncacheable であり、容量の大きなキャッシュは必要ないため、本評価では L2 キャッシュは用いない。

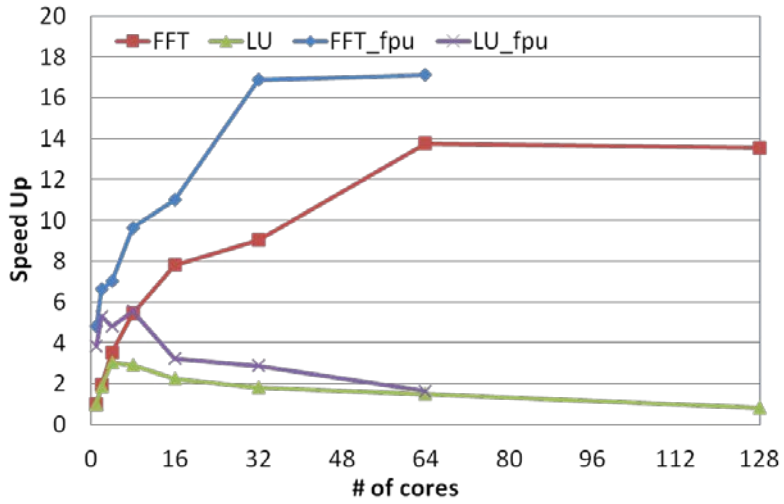


図 2.3-2 スピードアップ

図より、FFT ではコア数を増やすことで性能が向上していることがわかる。ただし、FFT では 64 コアで、FFT_fpu では 32 コア程度で性能向上が飽和している。また、使用したコア数に対してスピードアップが小さい。これは、メモリアクセスがボトルネックとなっているためである。キャッシュの一貫性を保つために、ヒープ領域のデータに対してはキャッシュを利用せず、ほとんどのデータアクセスが主記憶に対するアクセスとなるため、主記憶や NoC の帯域が飽和していると考えられる。LU プログラムではこの主記憶アクセスボトルネックの問題が顕著であり、コア数を増やしても極わずかな性能向上か、あるいは性能が悪化してしまっている。また、今回の評価環境の実装では、同期や排他制御の度にキャッシュをフラッシュしてしまっていること、またバリア同期には排他制御を利用したソフトウェア実装を用いており、同期にかかる時間が長いことも理由としてあげられる。キャッシュコヒーレンスの実装やバリア同期のハードウェアサポートを行うことで、並列処理効率は改善すると考えられる。

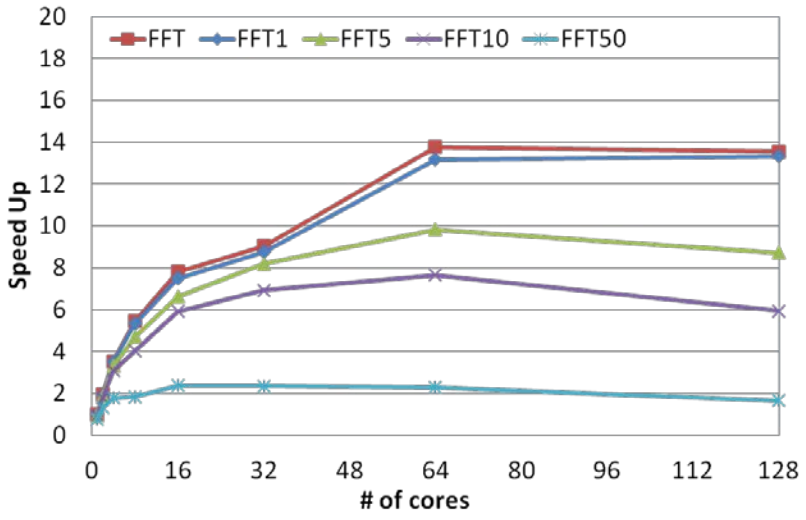


図 2.3-3 メモリアクセス時間の性能への影響

一般的に、主記憶アクセスはプロセッサの処理スピードに比べると遅い。本評価環境はプロセッサコアのクロック周波数が 10MHz、主記憶 DRAM の周波数が 100MHz と、実際のシステムに比べて主記憶アクセスの速度が相対的に低速である。そこで、実際のシステムに近い条件でも評価を行えるように、本評価環境に DRAM アクセスのレイテンシを追加する機能を実装した。図 2.3-3 にメモリアクセスレイテンシを変化させた際の FFT のスピードアップの評価結果を示す。図中、FFT の後に続く数字が追加した主記憶アクセスレイテンシ(プロセッサコアのクロックサイクル単位)である。当然、主記憶アクセスレイテンシが増加すると性能が低下する。そのため、並列化効率も低下してしまう。このように、本評価環境では様々な主記憶レイテンシのもとで評価が行えるが、これは FPGA 評価ボードを用いて柔軟な評価が可能であるという本評価環境の利点である。

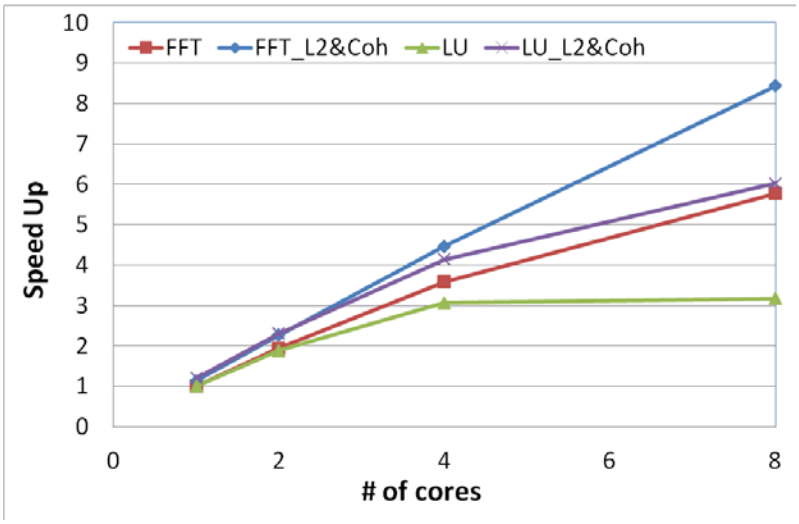


図 2.3-4 コヒーレンス制御と L2 キャッシュを用いた場合のスピードアップ

図 2.3-4 にキャッシュコヒーレンス制御と L2 キャッシュを用いた場合の各プログラムの

スピードアップの結果を示す。本評価では、ヒープデータ領域もキャッシュ経由でのアクセスとしている。現状では、キャッシュコヒーレンス制御はコア・クラスタ内のみが対象であるため、1 クラスタのみを用いて評価を行った。図より、コヒーレンス制御と L2 キャッシュを用いることで性能が大きく向上している。特に FFT では、ほぼコア数分の性能向上が得られている。このことから、メニーコアプロセッサでは、キャッシュを有効活用することが非常に重要であることがわかる。

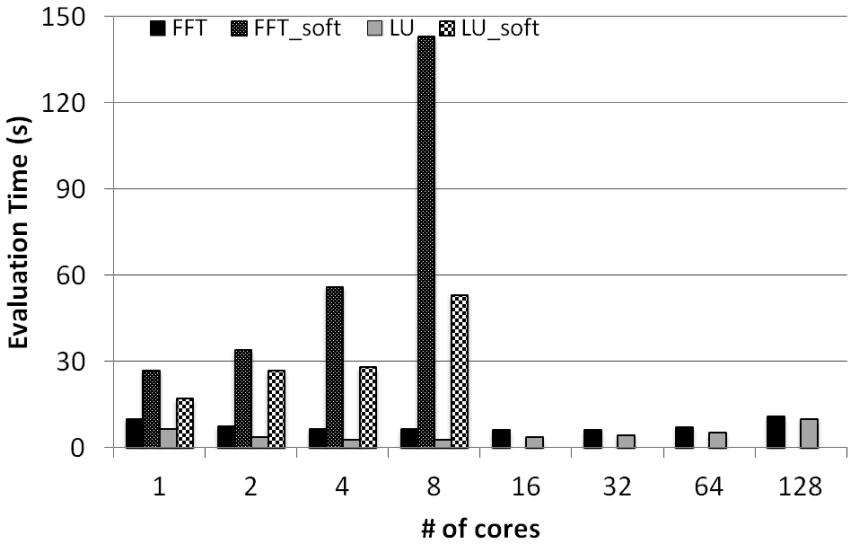


図 2.3-5 本評価環境とシミュレーションとの評価時間の比較

前述のように、FPGA を用いた評価環境の利点はその評価速度とスケーラビリティである。図 2.3-5 は本評価環境(FFT および LU)とソフトウェアシミュレータ(FFT_soft および LU_soft)を用いた場合の評価にかかる時間を比較したものである。ソフトウェアシミュレータには、サイクルレベルのプロセッサシミュレータである MARSS-x86 シミュレータを用いた。MARSS-x86 はシミュレーション速度が他のシミュレータに比べても高速であることが特徴である。シミュレーションを実行するホスト計算機の仕様を以下に示す。

- CPU: Intel core-i7 X980, 3.33GHz
- LLC (L3 キャッシュ): 12MB
- Main Memory: DDR3-1066, 6GB

図より、1 コアの場合では、ソフトウェアシミュレータの評価時間は FPGA 評価環境に比べてわずかに遅い程度である。しかしコア数が増加するにつれて、ソフトウェアシミュレータの評価時間が増加し、FPGA 評価環境の評価時間との差が大きくなっていく。8 コアの場合では、20 倍程度も評価時間に差があることがわかる。なお、MARSS-x86 シミュレータの制限により、ソフトウェアシミュレータでは 8 コアまでしか評価ができなかった。以上のことより、FPGA を用いたプロトタイピングによる評価は高速性と良いスケーラビリティ

を持ち、メニーコアプロセッサの評価環境として非常に有用であると考えられる。

2.4　性能評価

2.4.1 FPGA プロトタイプ・システム上でのデータ並列実行の評価

128 コアを搭載したメニーコア FPGA プロトタイプ・システム上に、SMYLE OpenCL 環境を実装した。この SMYLE Open 環境の上で、以下の 6 種類の OpenCL プログラムを実行した。

- Backprojection
- Gaussian
- Grayscale
- Blacksholes
- Linearsearch
- Runlength

プログラムを実行する際は、128 コアのうちの 1 コアをホストとして使用し、デバイスコアの数を 1 コアから最大 127 コアまで変化させた。コア数を増やす際は、データ並列で実行し、タスク並列実行は行っていない。

プログラムの実行時間を図 2.4-1～図 2.4-6 に示す。

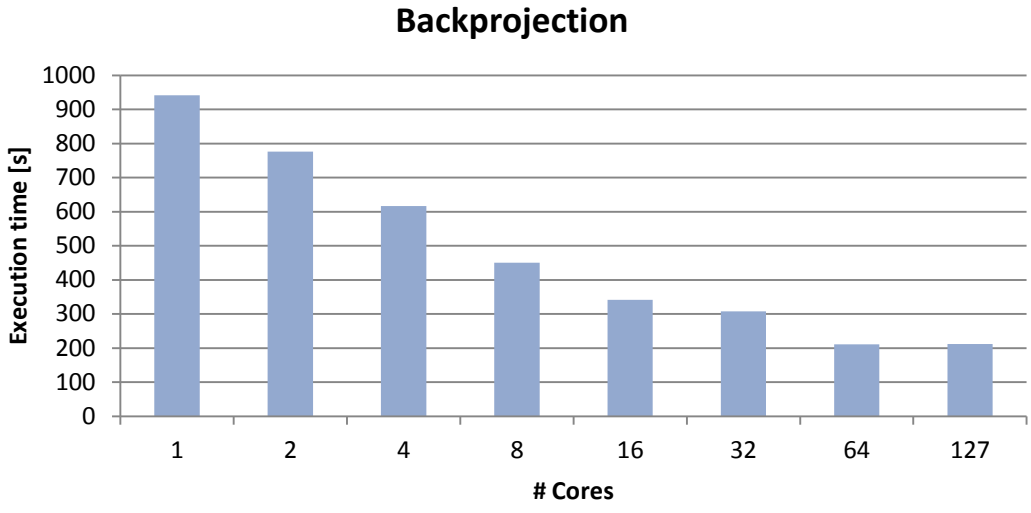


図 2.4-1　Backprojection の実行時間

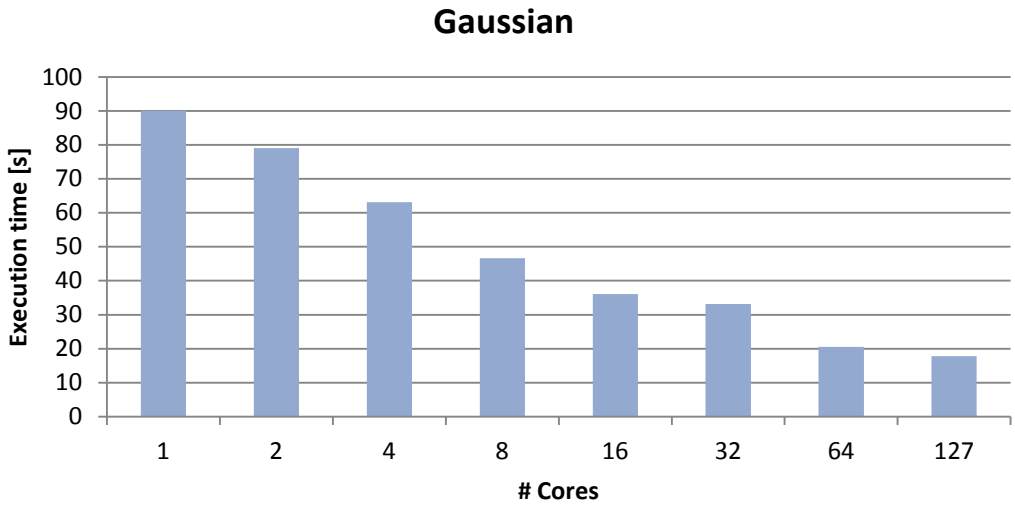


図 2.4-2　Gaussian の実行時間

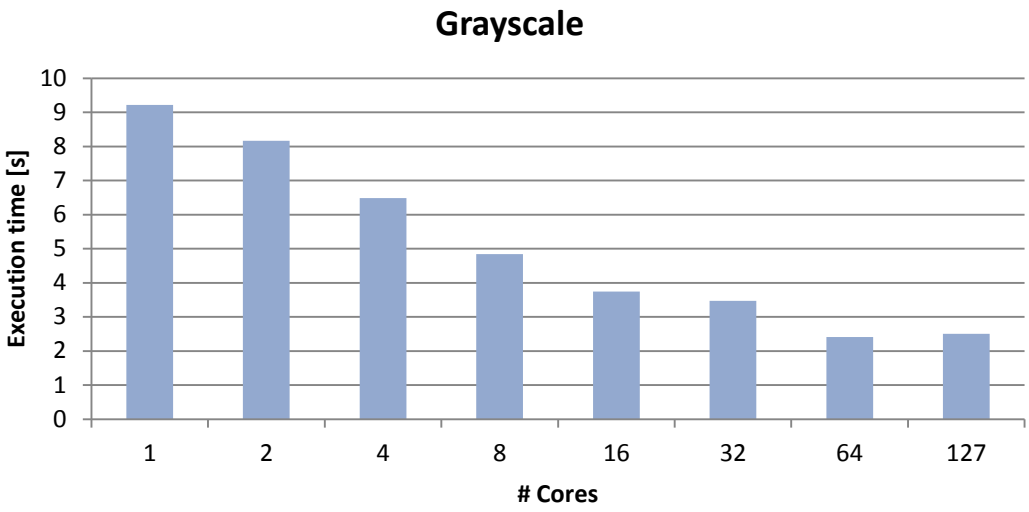


図 2.4-3　Grayscale の実行時間

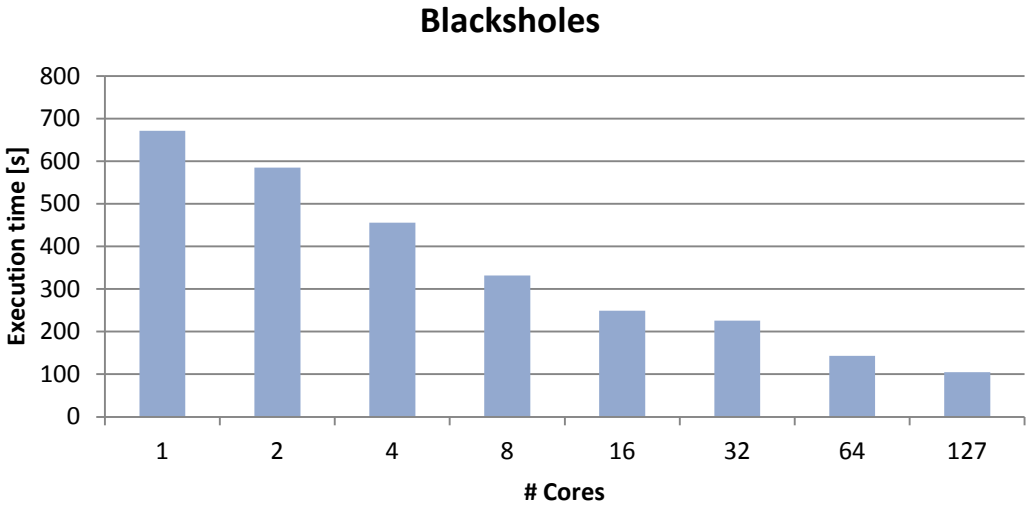


図 2.4-4　Blacksholes の実行時間

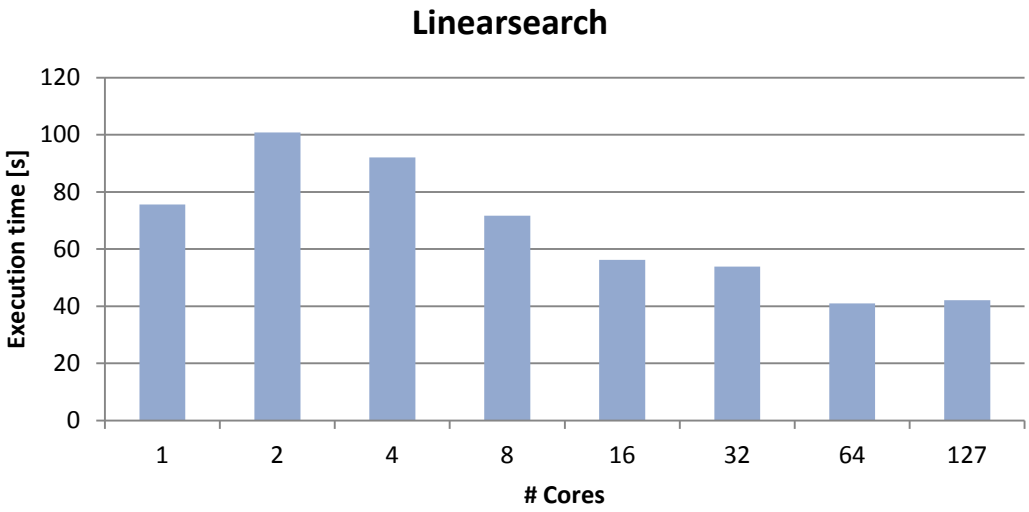


図 2.4-5　Linearsearch の実行時間

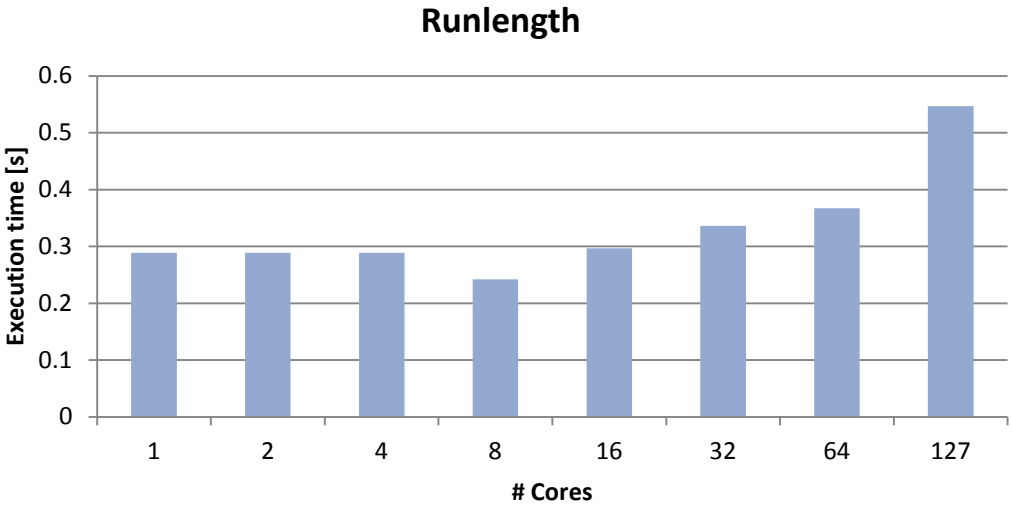


図 2.4-6　Runlength の実行時間

Backprojection、Gaussian、Grayscale、 Blacksholes については、コア数を増やすことで実行時間がほぼ単調に減少しており、並列実行の効果が表れている。Linearsearch については、コア数を 1 コアから 2 コアに増やした際、同期と通信のオーバーヘッドにより、実行時間が長くなっている。しかし、2 コア以上に増やした場合には、実行時間が減少している。

Runlength は、コア数を増やすと、実行時間が長くなる傾向が見られた。この理由は、入力として与えたデータが小さく、データ並列性が低かったことである。そのため、コア数を増やすと同期・通信のオーバーヘッドが大きくなり、実行時間が長くなった。

2.4.2 アプリケーション並列実行の有効性の評価

SMYLE OpenCL の特長の一つは、アプリケーション内のタスク並列実行が可能であるだけでなく、複数のアプリケーションを並列に実行することができることである。複数のアプリケーションの並列実行は、メモリ空間などの制約から、128 コアを搭載したメニーコア FPGA プロトタイプ・システム上では実装されていない。そこで、実測ではなく、机上の計算により、アプリケーション並列実行の有効性を評価した。

2.4.1 節で用いた 6 種類の OpenCL プログラムを、以下の 2 つの方法で実行した場合のスループットを計算した。

(A) 逐次実行：各プログラムは、最も実行時間が短くなるコア数を用いて、データ並列実行を行う。例えば、Gaussian を実行する際は 128 コアすべてを使用するが、Runlength を実行する際は 8 コアのみを使用する。あるプログラムの実行が終了すると、次のプ

ログラムが最大 128 コアを使用してデータ並列実行を行う。このように、6 つのプログラムを順次実行する。最後のプログラムの実行が終了すると、また最初から、6 つのプログラムを実行する。

(B) アプリケーション並列実行：6 つのプログラムを空間的に並列に実行する。128 コアを 6 つの領域に分割し、各領域をアプリケーションに排他的に割り当てる。各アプリケーションは、実行が終了すると、また最初から実行を開始する。各アプリケーションに割り当てるコアの数は、2.2.3 節で説明したシングルコンテキスト静的タスクマッピング手法（整数線形計画法）により決定する。

方法(A)逐次実行の場合、図 2.4-7 に示すように、Backprojection は 64 コア、Gaussian は 127 コア、Grayscale は 64 コア、Blacksholes は 127 コア、Linearsearch は 64 コア、Runlength は 8 コアを使用する。複数のアプリケーションは空間的に並列に実行されない。よって、例えば Runlength を実行中は、残りの 119 コアは停止することになる。

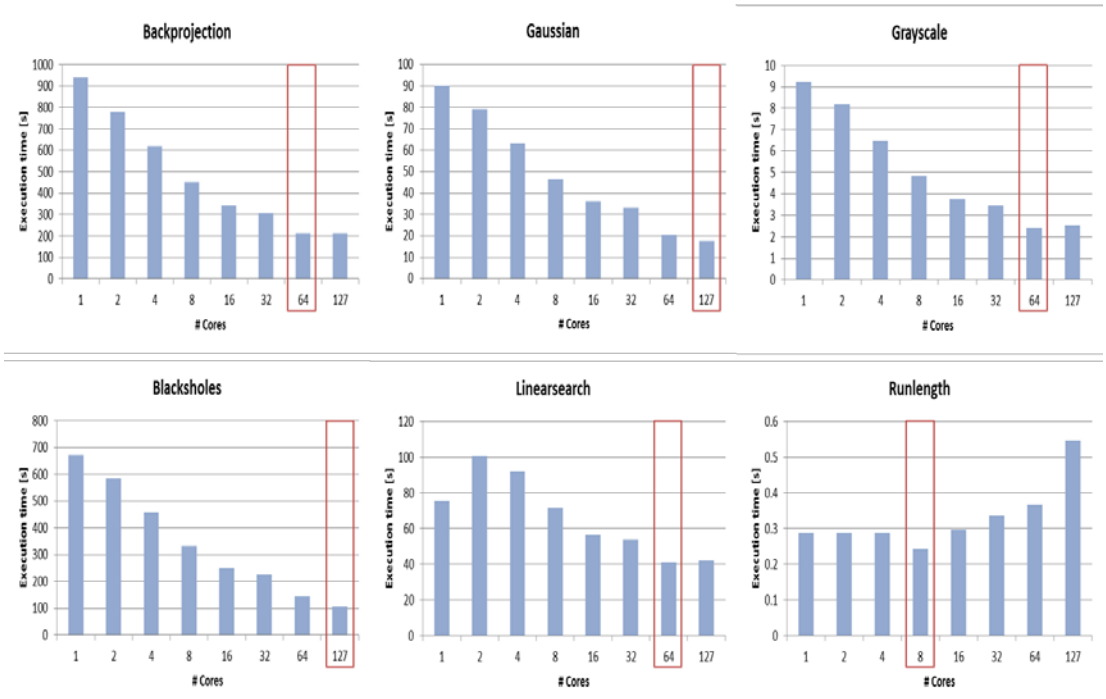


図 2.4-7　逐次実行の場合に各プログラムが使用するコア数

方法(B)アプリケーション並列実行の場合、各プログラムが使用するコア数を図 2.4-8 に示す。

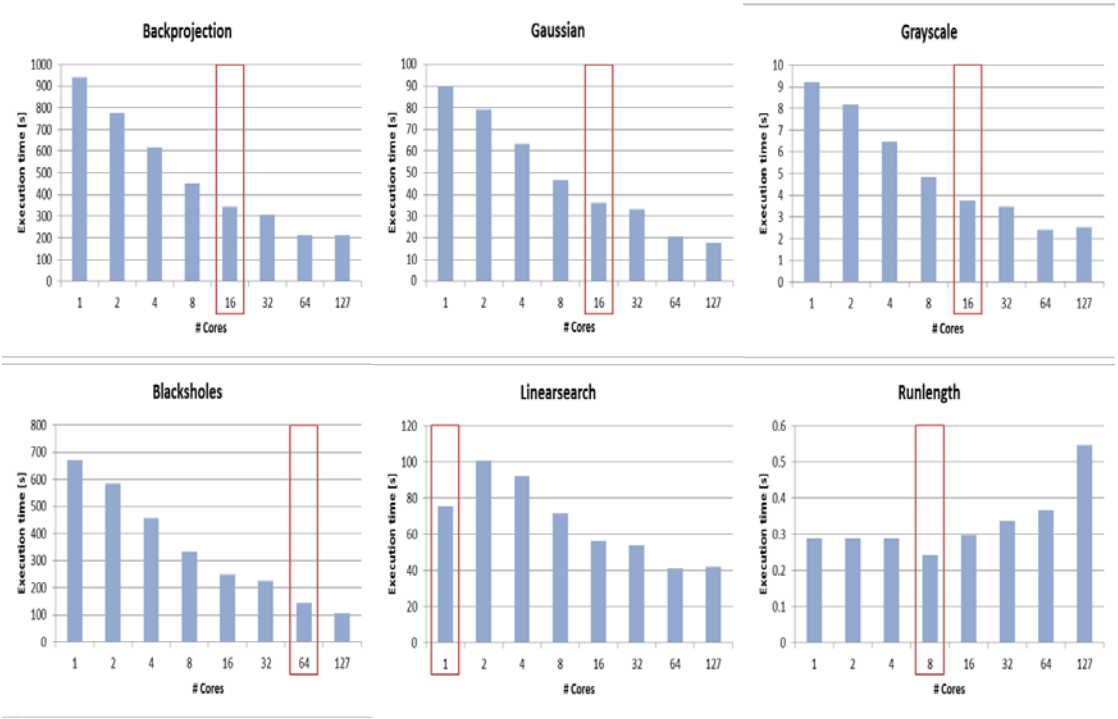


図 2.4-8　並列実行の場合に各プログラムが使用するコア数

Backprojection、Gaussian、Grayscale は各 16 コア、Blacksholes は 64 コア、Linearsearch は 1 コア、Runlength は 8 コア、合計 121 コアを使用する。残りの 6 コアは使用しない。

(A)逐次実行と(B)アプリケーション並列実行の 2 つの方法で 6 つのプログラムを実行した場合のスループットを表 2-4-1 に示す。表において、(A)逐次実行の場合のスループットを 1 として正規化している。

表 2-4-1　相対スループットの比較

(A)逐次実行	(B)アプリケーション並列実行
1	4.03

表に示した通り、逐次実行と比較して、アプリケーション並列実行の方が、スループット性能が 4.03 倍高いという結果が得られた。これは、動作周波数を 1/4 とし、それに伴い電源電圧を 40%程度削減できれば、動的消費エネルギーを約 1/10 に削減できることを意味する(つまり、性能を一定に保ちつつ、動的消費電力を 1/10 程度に削減)。実際、AMD 社のプロセッサチップ Opteron6136 では動作周波数を 1/3 にすることで約 30%の電源電圧低下を可能としており、より低電圧動作を考慮した最適化を施すことで、動作周波数 1/4 を許容することで 40%の電源電圧削減が十分に可能であると考える。

2.4.3 実行オーバーヘッドの評価

SMYLE OpenCL の特長のひとつは、実行の前処理、および、実行の後処理に要する性能のオーバーヘッドが低いことである。これは、タスクのコアへの割り当てや、コンテキストの生成、各種オブジェクトの確保などを、設計時に静的に行っているためである。

SMYLE OpenCL の実行オーバーヘッドを評価するため、既存の OpenCL 処理系との比較を行った。具体的には、以下の 3 つの実行環境の比較を行った。

- FOXC/Intel：Intel Core i7（2.8GHz、4 論理コア）上で、フィックスターズ社が開発した OpenCL 環境 FOXC を使用した場合
- SMYLE OpenCL/FPGA：FPGA 上に実装された SMYLEref アーキテクチャ（10MHz、4 コア）上で、SMYLE OpenCL 環境を使用した場合
- SMYLE OpenCL/Intel：Intel Core i7（2.8GHz、4 論理コア）上で、SMYLE OpenCL の機能シミュレータを使用した場合

ベンチマークプログラムとして Runlength を用いた。3 つの環境で Runlength プログラムを実行した場合の実行時間とその内訳を図 2.4-9 に示す。

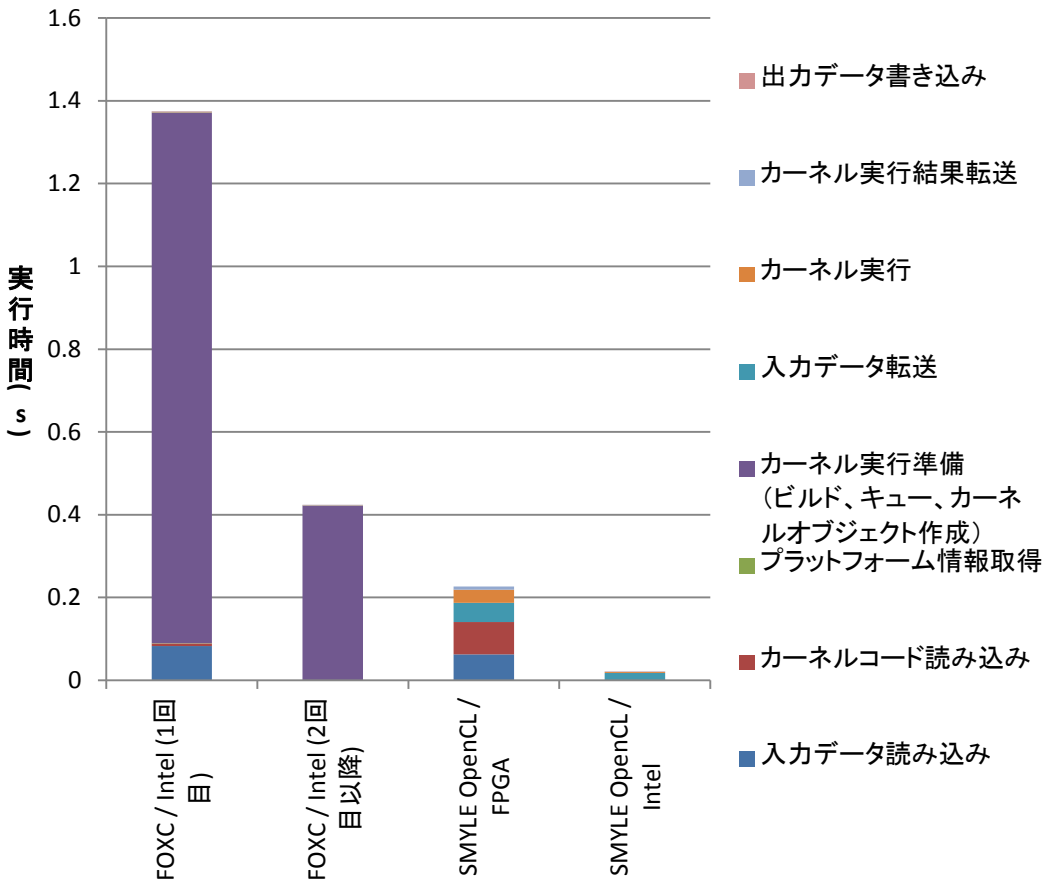


図 2.4-9　実行オーバーヘッドの評価

FOXC/Intel の場合、プログラムを 1 回目に実行した場合と、複数回連続して実行した場合の 2 回目以降とでは、各メモリ階層のキャッシュの効果により、実行時間が大きく異なった。そのため、1 回目と 2 回目以降とでは別々に結果を示している。

FOXC/Intel（2 回目以降）では、実行時間が約 0.42 秒であり、実行時間のほぼすべてをカーネル実行準備（カーネルのビルド、各種オブジェクトの生成）に費やしている。Runlength プログラムに与えた入力データは非常に小さかったため、カーネル実行時間は無視できるほど小さい。

一方、SMYLE OpenCL/FPGA の場合、実行時間が約 0.23 秒であった。FOXC/Intel と比較して、コアの性能は 1000 分の 1 程度であるにも関わらず、実行時間は 45%程度短くなっている。これは、SMYLE OpenCL では、プログラムの起動に関わる実行オーバーヘッドが大幅に削減できていることを示している。

SMYLE OpenCL/Intel の実行時間は約 0.02 秒であり、FOXC/Intel の 20 分の 1 以下となった。このことからも、SMYLE OpenCL のリアルタイム性能の高さが実証された。

次に、SMYLE OpenCL/FPGA 環境において、デバイスコア数を 1 コアから 127 コアまで増やした場合の実行オーバーヘッドを評価した。結果を図 2.4-10 に示す。

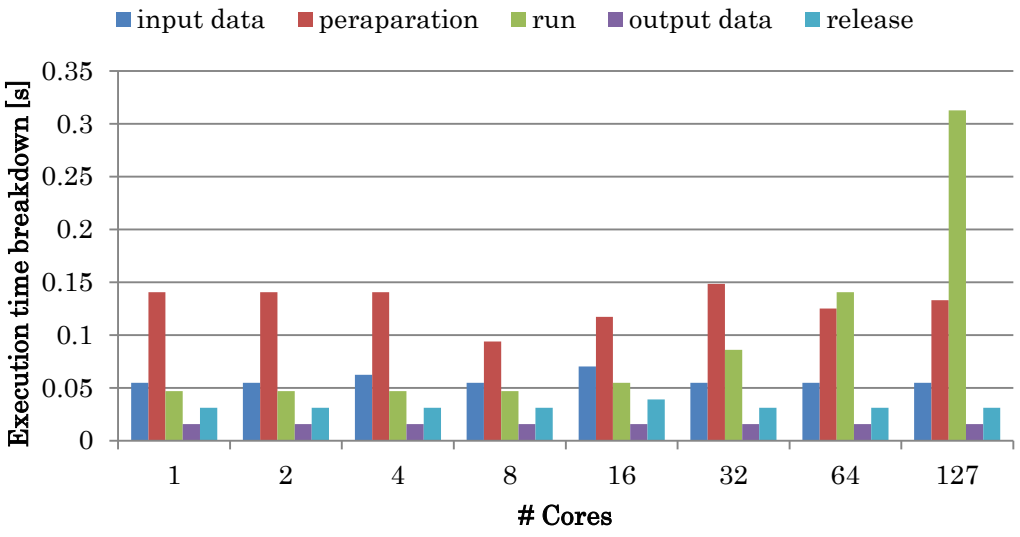


図 2.4-10　コア数を変えた場合の実行オーバーヘッドの評価

グラフは、Runlength プログラムを実行した場合の、実行時間の内訳を表している。各棒グラフの意味は以下の通りである。

- input data：入力データの読込み
- preparation：カーネル実行準備
- run：カーネル実行
- output data：出力データ書込み

- release：カーネル実行後処理（オブジェクトの解放）

Runlength プログラムを実行する際は非常に小さな入力データを与えたため、カーネル実行に要する時間は短い。また、コア数を増やすと、同期・通信のため、カーネル実行時間が悪化している。しかし、ここで注目すべきは、カーネル実行準備とカーネル実行後処理に要する時間（実行オーバーヘッド）である。コア数を 127 コアまで増やしても、カーネル実行の準備と後処理に要するオーバーヘッドは低く抑えられている。このことも、SMYLE OpenCL のリアルタイム応答性の高さを示している。

参考文献

[Seiler2008]Seiler, Larry and Carmean, Doug and Sprangle, Eric and Forsyth, Tom and Abrash, Michael and Dubey, Pradeep and Junkins, Stephen and Lake, Adam and Sugerman, Jeremy and Cavin, Robert and Espasa, Roger and Grochowski, Ed and Juan, Toni and Hanrahan, Pat, "Larrabee: a many-core x86 architecture for visual computing," SIGGRAPH '08, Aug. 2008.

[Ramey2011] Ramey, C., "TILE-Gx100 ManyCore Processor: Acceleration Interfaces and Architecture," Hot Chips 23, 2011.

[Howard2010] Howard, J. and Dighe, S. and Hoskote, Y. and Vangal, S. and Finan, D. and Ruhl, G. and Jenkins, D. and Wilson, H. and Borkar, N. and Schrom, G. and others,"A 48-core IA-32 message-passing processor with DVFS in 45nm CMOS," Prof. of the Solid-State Circuits Conference Digest of Technical Papers (ISSCC), pp.108-109, 2010.

[Semeraro2002] Semeraro, G. and Magklis, G. and Balasubramonian, R. and Albonesi, D.H. and Dwarkadas, S. and Scott, M.L.,"Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling," International Symposium on High-Performance Computer Architecture, pp.29-40, 2002.

[Herbert2007] Herbert, S. and Marculescu, D.,"Analysis of dynamic voltage/frequency scaling in chip-multiprocessors," International Symposium on Low Power Electronics and Design, pp.38-43, 2007.

[PARSEC2008] Bienia, C. and Kumar, S. and Singh, J.P. and Li, K.,"The PARSEC benchmark suite: Characterization and architectural implications," Proc. of the 17th

international conference on Parallel architectures and compilation techniques, pp.72-81, 2008.

[Cochran2011] Cochran, R. and Hankendi, C. and Coskun, A.K. and Reda, S.,"Pack & Cap: Adaptive DVFS and Thread Packing Under Power Caps," Proc. of the 2011 44th Annual IEEE/ACM International Symposium on Microarchitecture, pp.175-185, 2011.

[Bienia2008] Bienia, C. and Kumar, S. and Li, K.,"PARSEC vs. SPLASH-2: A quantitative comparison of two multithreaded benchmark suites on chip-multiprocessors," International Symposium on Workload Characterization, pp.47-56, 2008.

[Wilkinson1999] Barry Wilkinson and Michael Allen,"Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers," Prentice Hall, 1999.

[PLB2012] Xilinx, "PLBV46 Master Burst", http://www.xilinx.com/support/documentation/ip_documentation/plbv46_master_burst.pdf, 2010.

[Woo1995] S.C. Woo, M. Ohara, E. Torrie, J.P. Singh, and A. Gupta, "The SPLASH-2 Programs: Characterization and Methodological Considerations", In Proc. the 22nd International Symposium on Computer Architecture (ISCA-95), pp.24-36, 1995.

[Sasaki2012] Hiroshi Sasaki, Teruo Tanimoto, Koji Inoue, and Hiroshi Nakamura, "Scalability-Based Manycore Partitioning," International Conference on Parallel Architectures and Compilation Techniques (PACT'12), Sep. 2012.

[関 2010] 関直臣ほか,"MIPS R3000 プロセッサにおける細粒度動的スリープ制御の実装と評価," 電子情報通信学会論文誌 D, Vol. J93-D, No. 6, pp. 920-930, 2010.

3　ビジュアル・コンピューティング・メニーコア SMYLEvideo の開発

3.1　研究成果概略

ビジュアル・コンピューティング分野の１つであるビデオ・マイニング（膨大な動画像データの中から特定の動画像の認識や追跡、検索、ダイジェストの抽出など）をターゲット・アプリケーションとした、メニーコア・アーキテクチャ SMYLEvideo を開発した。SMYLEvideo アーキテクチャ・アルゴリズム協調設計、ストリーム処理かつ画像処理に適したデータ処理エンジン、それらを組み合わせたスケーラブルなメニーコア・アーキテクチャによって、従来の組込みマルチコア方式（4 コア程度）と比較して性能 1.7 倍、消費電力量 1/2、回路規模 1/2 が実現可能であることを示した。

3.2　SMYLEvideo の開発

3.2.1 SMYLEvideo ハードウェア・アーキテクチャ開発

画像認識のアルゴリズムには潤沢な並列性を内含しており、マルチコア、メニーコアによって、効率的に処理速度の高速化が可能である。しかしながら、処理負荷自体は非常に高く、リアルタイムの画像認識のためには、500GOPS(Giga Operation Per Second)を超える程度の性能を実現する必要がある。このような性能を低消費電力で実現するには、10 個以上の演算コアを有機的に結合して、効率よく動作させるとともに、特定資源へのアクセスの集中が生じないよう注意深くアーキテクチャ設計を行う必要がある。そこで、SMYLEvideo ではトプスシステムズ社が有するヘテロジニアス・マルチコア TOPSTREAM™ アーキテクチャに、並列度を効率的に高めるべくコア数を増やせるクラスタ構造を融合して、所望の性能を実現するアプローチをとった。SMYLEvideo のアーキテクチャ的な特徴を以下に記す。

- Kahn Process Network 型の分散処理
  並列実行モデルとして Kahn Process Networks (KPN) [Kahn 1974]を採用し、KPN のタスク間 FIFO 結合を効率的に処理するため複数のハードウェアサポートを追加

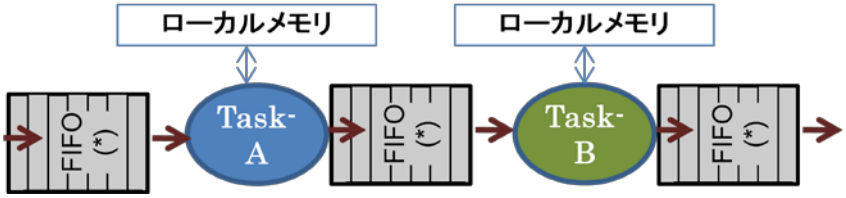


図 3.2-１　Kahn Process Networks

- 2 つの並列処理の組合せ
  KPN に基づくタスク並列処理とデータ処理量に応じたデータ並列処理を組み合わせて、各タスクの負荷を均等化

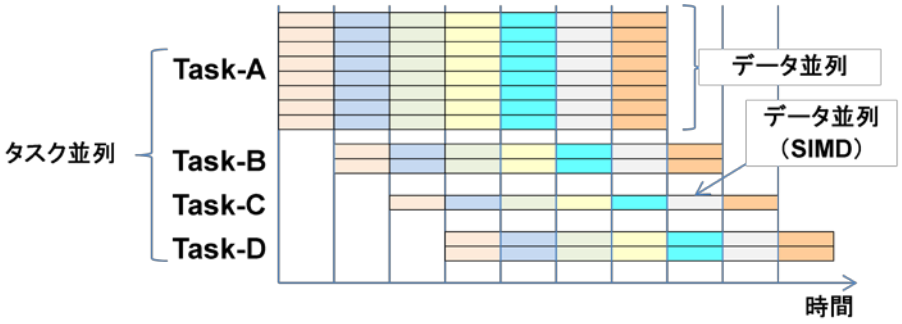


図 3.2- 2 タスク並列とデータ並列の組合せ

- 各データ処理エンジン(Data Processing Engine: DPE) のストリーム処理最適化
  DPE は演算処理のバックグランドで、ストリーム入力、ストリーム出力が動作するように設計して、ストリームの入出力処理を隠ぺい

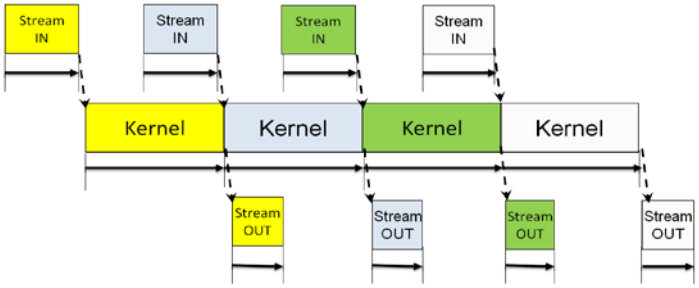


図 3.2- 3　ストリームイン、ストリームアウトと計算処理（Kernel）とのオーバラップ

- DPE コアの画像処理に向けた最適化
  画像処理のアルゴリズムに応じたデータパス、データレジスタ・SIMD の幅選択(64b/256b アーキテクチャ)、複合命令による必要サイクル数の削減、KPN の FIFO 処理を効率化するための専用メモリ、専用 DPE 間結合バスの追加など、必要サイクル数や同期オーバヘッドを削減する仕組みの追加

表 3.2- 1 主な拡張複合命令

Mnemonic	Description
GFHPRAU	Gaussian Filter Horizontal Pre-processing A
GFHPRBU	Gaussian Filter Horizontal Pre-processing B
GFHPRCU	Gaussian Filter Horizontal Pre-processing C
GFHPRDU	Gaussian Filter Horizontal Pre-processing D
GFHPOHU	Gaussian Filter Horizontal Post-processing High

GFHPOLU	Gaussian Filter Horizontal Post-processing Low
SOSOB	Sum of Square of Byte data
SOSOSB	Sum of Square of Subtraction Byte data
PSUBXX	Subtract each element
PATAN	Pseudo ATAN
PNORM	Pseudo norm

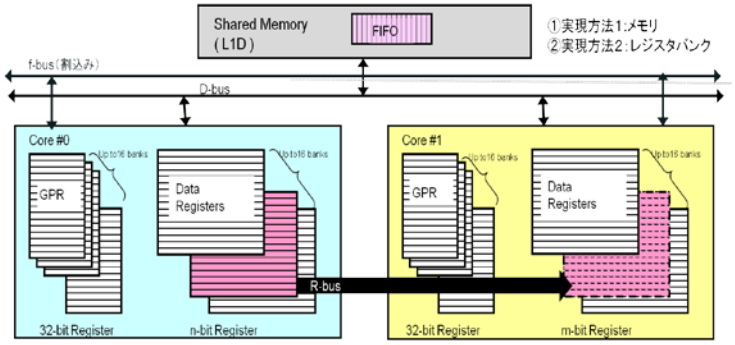


図 3.2- 4　コア間のレジスタファイル共有（FIFO 処理効率化）

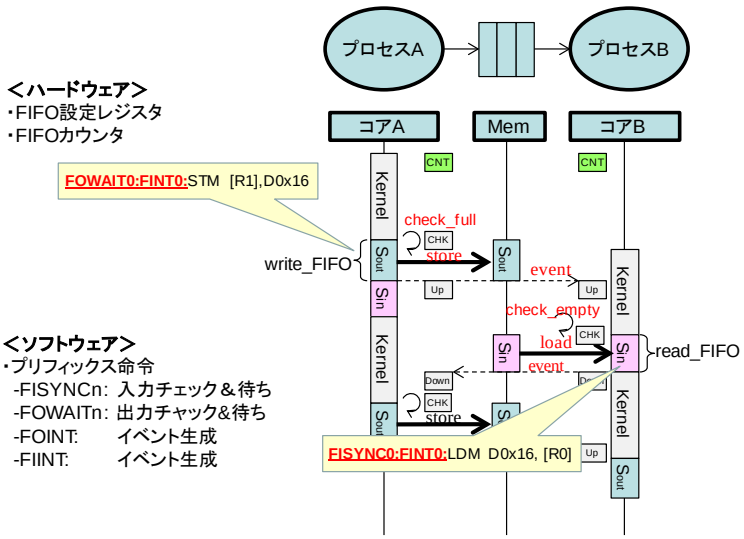


図 3.2- 5　prefix 命令による同期オーバヘッド削減(ゼロオーバヘッド交信)

これらの特徴により、KPN に基づいたストリーム並列処理を効率的に処理することができる。図 3.2- 6 は、SMYLEvideo の全体ブロック図である。

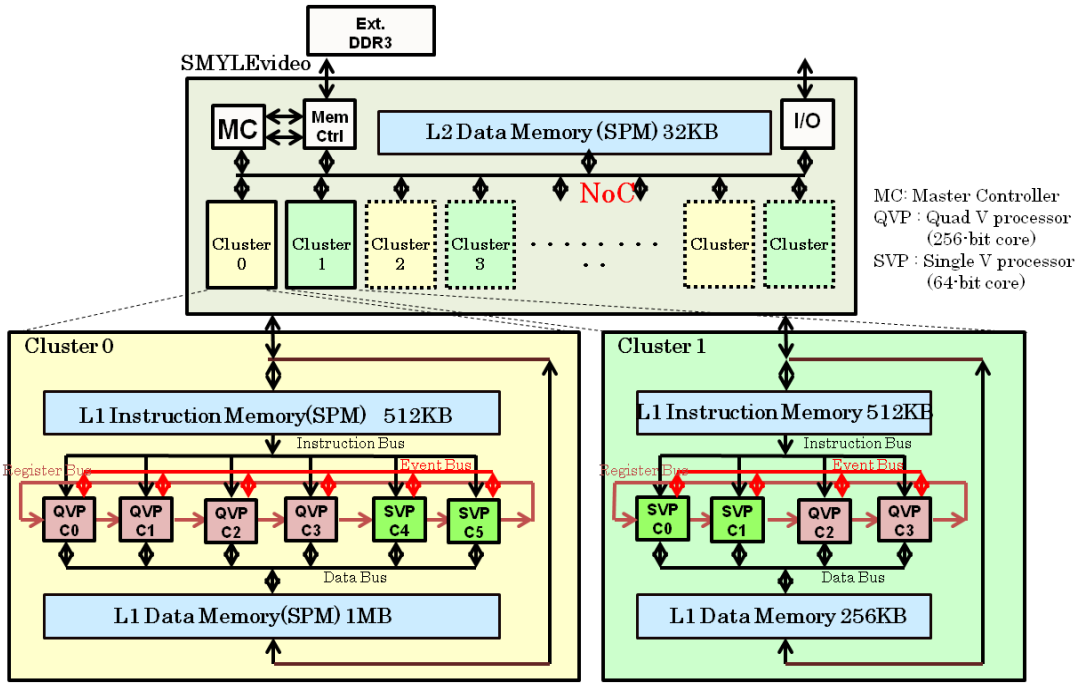


図 3.2- 6　SMYLEvideo のブロック図

前述のように SMYLEvideo は、クラスタ構造を採用している。クラスタ 0 とクラスタ 1 で演算コア DPE（Data Processing Engine）数が異なるが、これは画像認識において、クラスタ 0 が、フィルタ処理を行い、特徴点を抽出するような処理、すなわち全画像を探索する処理に充てられ、クラスタ 1 がクラスタ 0 で抽出した特徴点を解析する処理に充てられることを考慮し、データのローカリティ、DPE 間交信の効率化を踏まえて決定したものである。さらに、DPE は 256 ビットデータレジスタバンクならびに SIMD 演算器を備える QVP(Quad Vision Processor)と、 64 ビットデータレジスタバンクならびに SIMD 演算器を備える SVP(Single Vision Processor)を処理フェーズによって使い分けることから、2 種類用意している。

また、本クラスタペアを追加することによって、スケーラブルに画像認識能力、すなわち一秒当たりのフレーム処理枚数を増加させることが可能である。

これらクラスタや L2 メモリ、処理全体を司さどる MC（Master Controller）、周辺ペリフェラルインタフェース間をスケーラブルな NoC（Network on Chip）にて接続した。

この NoC は以下の特徴を持ち、効率的なクラスタ間データ転送が可能である。

- 必要な部分にデータコネクションを張った、部分クロスバ構造
- BUS 型及び直結型の SBUS インタフェースに対応
- 生成時にクラスタ数を指定すると、それに応じた RTL を生成
- 2 段パイプラインによる通信

- 排他制御機能(ロックによる経路占有)
- busy/wait フロー制御

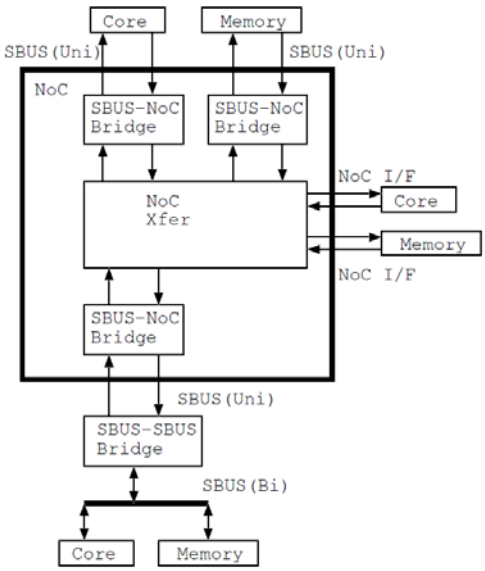


図 3.2- 7　NoC の接続ブロック図

これら SMYLEvideo のアーキテクチャを実証、評価するために、実 RTL 開発を行った。RTL 開発は、VHDL 言語を用い、GUI ベースの設計県境である Visual Elite を用いて行い (図 3.2-8)、東京エレクトロンデバイスの ASIC 開発評価プラットフォーム TB-7V-2000T-LSI　FPGA ボードを用いて、RTL 検証ならびに動作確認を行った。

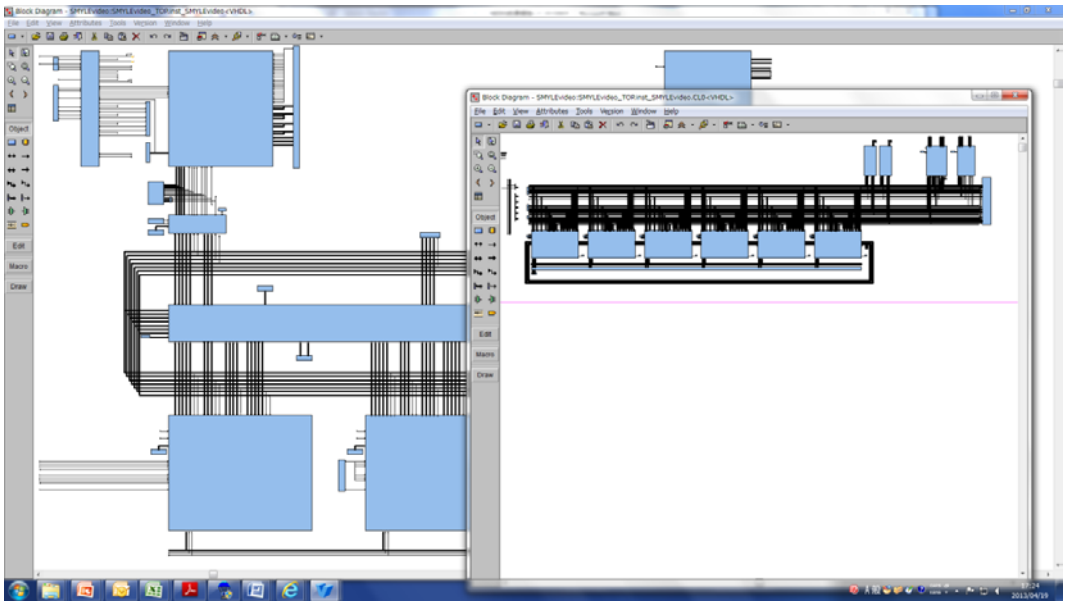


図 3.2- 8　SMYLEvideo の RTL 実装（Visual Elite）

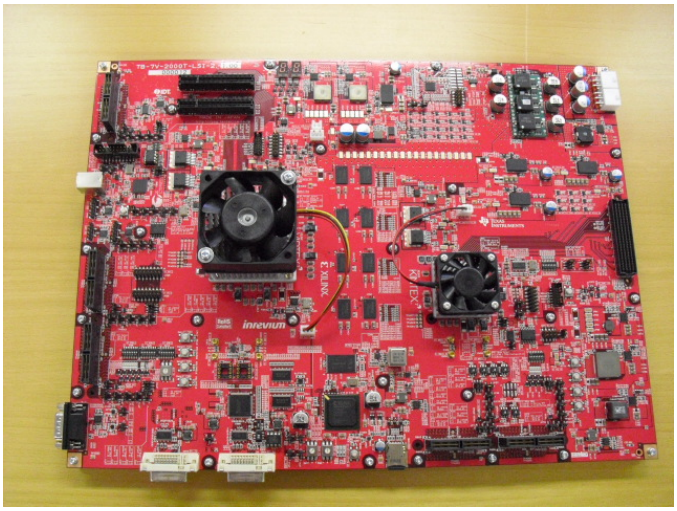


図 3.2- 9　SMYLEvideo の実装検証ボード
（東京エレクトロンデバイス ASIC 開発評価プラットフォーム TB-7V-2000T-LSI）

FPGA 用の合成を行った結果から算出したハードウェアの規模を表 3.2- 2 に示す。

表 3.2- 2　ハードウェア規模(面積は 40nm を仮定)

	単体		2 クラスタ	合計	
	面積[um2]	ゲート数	集積数	面積[um^2]	ゲート数
SVP	96,192	146,396	4	384,768	585,584
QVP	253,158	385,286	6	1,518,950	2,311,713
MC	19,544	29,745	1	19,544	29,745
IM(除くメモリ)	6,818	10,376	1	6,818	10,376
DM(除くメモリ)	7,967	12,125	1	7,967	12,125
MEMC	7,022	10,687	2	14,043	21,373
I/O	4,185	6,369	1	4,185	6,369
合計				1,956,275	2,977,285

このことから、2 クラスタ構成の SMYLEvideo チップサイズは面積から 1.4mm□に約 2MB のオンチップメモリを加えた面積となる。これを加味するとおおよそ 1.7mm□の面積となると予想される。本アーキテクチャの評価については、3.5 節で検討する。

3.2.2 SMYLEvideo への動画認識アルゴリズム SIFT の実装

SIFT アルゴリズムとは

今回メニーコア上への実装を行った SIFT とは Scale-invariant feature transform の略である画像認識アルゴリズムである[David G. Lowe 1999]。開発者は David Lowe 氏である。このアルゴリズムは画像の回転、サイズの変化、照明の変化などに対して安定した認識が行えるアルゴリズムとして知られており、例えば自動車から撮影した動画像ストリームをこのアルゴリズムで解析することで、交通標識などを自動認識させるような応用が期待されている。

今回実装の参考としたソースコードは、オープンソースプロジェクトである OpenCV プロジェクトに含まれているものである。

性能目標

物理的な動画入力装置としては、1920x1080 ピクセルサイズのフルカラーHD ビデオカメラを用いて動画像を入力し、これをハードウェアで３対１のダウンザイジングし 640x360 ピクセルサイズ、モノクロ２５６階調グレースケール画像としたものをソフトウェアへの入力データとしている。

ソフトウェアの上位目標としては標準構成にて 15fps から最大クラスタ構成にて 60fps までスケーラブルに拡張可能な構成とすることであり、今回実装の標準構成では 15fps の実時間スループットを実現することを目標性能とした。またこの構成を複数クラスタ並行動作させることで、30fps、60fps へのスケーラビリティが確保することとした。

SIFT アルゴリズムのメニーコア上での実現方針

オリジナルの SIFT アルゴリズムは、倍精度浮動小数点演算を多用し、主記憶メモリ上に展開した画像データを大量に読み書きするものである。内部で１枚の画像からパラメタの異なる２０枚以上の画像を作り出して処理を行うため、そのメモリアクセス量は、入力画像１枚あたりの容量の数十倍以上に達する。また、VGA サイズの１画面の処理にパーソナルコンピュータ上で一般的な命令セットで 3G から 9G サイクルもの演算処理が必要とされる。このため非常に重いアルゴリズムとして知られており、通常のパーソナルコンピュータなどでは実時間処理が難しい。また、当然、大量のメモリアクセスが本アルゴリズムの消費電力増大の要因となっている。

そこで今回のメニーコア上での実現にあたっては、まず浮動小数点演算を固定小数点化することで負荷を軽減することを考えた。しかし、単純な置き換えはできない。オリジナルアルゴリズムとほぼ等価な結果をえるため、数値実験をおこなって、ソフトウェアの各部における必要な固定小数点ビット数を決定し、これに基づいて実装した。

また、パソコン上と同一のメモリアクセス方式、演算方式では消費電力効率は悪くなる

ため、ソフトウェアの等価性を検証しつつ、メニーコアプロセッサに適した方式に改めることとした。

このため、アルゴリズムの考え方は等価であるが、ソースコードレベルではまったく実装は異なる。

SIFT アルゴリズムの機能分割

オリジナルの SIFT アルゴリズムは 5000 行を超える規模のアルゴリズムであるので、これをメニーコアプロセッサ上で実装するためには、アルゴリズムの機能分割が必須である。機能分割の準備のため、まずパーソナルコンピュータ上で動作させたオリジナルの OpenCV プログラムに対してプロファイリングを行い、使用されている関数毎に、実際に機械語レベルで使用されている演算、データのビット幅、演算回数などの情報を取得した。この情報にもとづき、オリジナルのソースコードを解読、分類し、一端、細粒度処理（意味のある一つのデータを出力するために必要な入力とその処理の単位）にまで分解した後、以下のような機能に再構成（グルーピング）した。

① ガウシアンフィルタおよび差分画像の生成機能
② 特徴点候補の発見機能
③ 特徴点候補の絞り込みによる特徴点確定機能
④ 特徴点まわりの勾配強度、画素方向の算出機能
⑤ 勾配強度、画素方向にもとづく特徴点主方向の決定機能
⑥ 特徴ベクトルの算出機能
⑦ 特徴ベクトルとテンプレート（辞書）ベクトルの照合機能

以下に各機能の概略を示す。

ガウシアンフィルタおよび差分画像の生成機能

入力画像を強度の異なるガウシアンフィルタに次々と通過させガウシアン画像を生成する。ガウシアンフィルタはいわゆるガウス関数（正規分布の釣鐘状の分布を与える）を、画素中心点のまわりに施すもので、強度にもよるが、例えば１画素の結果を得るために 169 点についての乗算とその和の計算が必要とされる。このフィルタにより画像はぼやける。このフィルタ処理の出力をさらにまた次段のフィルタ処理の入力として使って次々と画像を生成するとともに、途中画像のスケーリングも行って、画像サイズを縮小していく。このスケーリングにより距離の大小等に対してロバストな特性を得ることができる。また、生成したガウシアン画像のとなりあう２枚についてその差分をもとめ差分画像として次機能の入力とする。ぼやけたガウシアン画像の差分をとることで、照明や霧などの周囲の効果に影響されにくい特

徴的な部分を抽出することが可能となる。

特徴点候補の発見機能

前機能が出力する差分画像 3 枚を一度に処理し、特徴点の候補となるべき画素を検出する処理である。3 ピクセルｘ3 ピクセルｘ3 ピクセルの立体状の画素空間について、中央の画素が他の画素およびある閾値に対して最大なのか最少なのかを検出し、極値であれば候補として、周囲の画素データとともに次機能に渡す。

特徴点候補の絞り込みによる特徴点確定機能

前機能から渡された候補画素とその周辺の画素データから特徴点として確定してよいか否かを数段階のテストを経て決定する。ただし、決定にあたっては、特徴点位置はピークの位置であって、その位置は、画素と画素の途中にあるものと推定されて計算される。演算の中心はヘッセ行列とその逆行列算出処理である。

特徴点まわりの勾配強度、画素方向の算出機能

前機能から与えられた特徴点の位置を使い、対応するガウシアン画像の周囲の画素数百ピクセルに対して、各画素の勾配強度およびその方向を算出する機能である。ここで求めた勾配強度および方向は次機能および次々機能の入力データとなる。

勾配強度、画素方向にもとづく特徴点主方向の決定機能

前機能から与えられた特徴点の周囲画素の勾配強度について、ガウス関数による重みづけをおこない、画素の方向から決定されるビン番号に対してヒストグラム集計を行う。集計後のヒストグラムを平滑化することで、特徴点の主方向（複数検出されることもありえる）を求める。

特徴ベクトルの算出機能

前機能でもとまった主方向に基づき、ガウス窓を回転し周囲画素の勾配強度に適用する。また、各画素の方向も主方向に基づき回転させる。この処理により、回転に対して強靭な認識を行うことができる。この処理後の各画素データを 4 x 4 領域毎に 8 方向別に分類し、またもヒストグラム集計する。各８分類をもつ合計 16 個のヒストグラムに全 128 要素のデータが集計される。ヒストグラム集計後のデータを 128 要素のひとつのベクトルとみて正規化を行い、これを特徴点に結び付けられた特徴ベクトルとして次機能に渡す。

特徴ベクトルとテンプレート（辞書）ベクトルの照合機能

認識すべき画像情報（たとえば交通標識など）を予め SIFT アルゴリズムに通して得

ておいた比較対象用の特徴ベクトル（テンプレートベクトル、あるいは辞書ベクトル）と、前機能から出力された入力画像中に発見された特徴ベクトルとを照合し、一致を検出する機能である。本実装では、メッシュ投票アルゴリズムが採用されており、ある一定以上の一致確度があるものについて投票を行って、その投票が集中するところをもってテンプレートの発見位置（および角度）としている。

機能分割したソフトウェアのメニーコアへのマッピング

機能分割したソフトウェアを、その要求する処理サイクルをもとに各コアにマッピングをおこなった。本実装では、２種の異なるコア、つまりヘテロジニアス型のメニーコアにマッピングしている。第１のコアタイプは、QVP と呼ばれるタイプであり、256 ビット幅の幅広いデータ演算パスを持つものである。16 ビット/ピクセルのデータ形式の場合、16 ピクセルを一度にＳＩＭＤ型の演算命令により処理することが可能である。第２のタイプは、SVP と呼ばれるタイプであり、64 ビット幅のデータ演算パスを持つタイプである。前述の QVP にくらべると非力ではあるが、やはりＳＩＭＤ型の演算命令を持つ。パス幅が狭い分、ハードウェア量、消費電力ともに低く、幅広の演算パスでは無駄になるようなスカラー処理などに適する。

第１の機能であるガウシアンフィルタ機能はそのデータ並列性を考え、最小構成では 4 個の QVP コアに割り当てた。これにより１画像を複数のコアでデータ並列的に処理することも、また、異なるガウシアン強度の複数画像をパイプライン的に処理することも可能である。標準構成では５個のコアを割り当てる。

第２の機能である特徴点候補の抽出は、比較的軽い処理であるので SVP タイプ１個に割り当てた。また引き続く第３の機能である特徴点候補の絞り込みと確定も SVP タイプ１個である。

第４の機能である勾配強度および方向の算出処理および第５の機能である主方向の決定機能は、いずれも特徴点まわりの局所的な画像を処理するものであり、SVP タイプそれぞれ１個を割り当てている。

第６の特徴点ベクトルの生成および第７の特徴点ベクトルとテンプレート（辞書）ベクトルとの照合機能は、いずれも 128 バイト幅に達する特徴点ベクトルを操作するため、幅広のデータパスを持つ QVP タイプに割り当てた。最小構成にてコアの割り当ては各１個であるが、標準構成では各２個ずつの割り当てとなる。

比較対象としてのフレームパイプライン実装

メニーコア上でのソフトウェア実装にあたっては、まず、従来型のソフトウェアの構成案として、動画像のフレーム時間ごとにパイプライン化する案を考えた。各機能（もしくは複数の機能を結合したもの）に 1 フレーム分の時間を与え、各機能の処理の結果は次のフレームで次機能に処理させる方式である。データを次機能に渡してしまえば、各機能は

次のフレームの処理に入れるから、フレームレートのスループットで処理を進めることができる。ただし、入力画像が入ってから結果が出るまでにはパイプライン化したフレーム処理段数だけのレイテンシーが必要である。各機能を１段としてパイプラインを組んでしまうと実に７フレームの遅延となる。レイテンシー短縮のため、第 2、第 3 の機能を１つのステージ、第 4、第 5 の機能も一つのステージと統合することにより全体を 5 段構成とし、5 フレーム時間の遅延で処理できるようにした。15fps の最小構成では、各段に与えられる処理時間は 100 ミリ秒であり、330 ミリ秒の遅延時間が発生する。

フレームパイプライン実装の問題点

前述のフレームパイプライン実装の一つの問題は､遅延時間であるが､他の問題として、メモリへのアクセス量の多さ、メモリの所要量の多さがあげられる。各フレーム間のインタフェースのために、前段処理は、生成した全データを一端主記憶メモリに書きだし、続いて、次のフレーム時間で後段処理がそれらを読みだす。データによっては、次フレームでなく、次の次のフレームまで保持する必要がある。このため、フレーム間でのデータ競合を防ぐために、最低でもダブルバッファ、データによってはトリプルバッファでの取り扱いが必要となる。

メモリ所要量とメモリアクセスバンド幅削減に関する検討

メモリアクセスのバンド幅といった場合、問題となるのは主記憶へのアクセスのバンド幅である。前述のフレームパイプライン方式の場合、フレーム間の受け渡しのために多数の中間画像を生成する。それらの合計必要量サイズは 22M バイトに達し、30fps 時の主記憶メモリバンド幅は 700M バイト毎秒を超える（後に述べるが処理単位が小さいとこの数倍のバンド幅になる可能性もある)。　このようなメモリ容量をキャッシュに置くことは難しく、また、書き込みからその読み出しまで１フレーム時間（30fps 時でも 33 ミリ秒とプロセッサからしたら長い時間である）あく上に、同じデータを読みだすのは高々２回程度、１回のものもあるのでその性質上もキャッシュには適さない。

その上、メモリアクセスはプロセッサの消費する電流の半分以上を占めるというのが一般的な傾向であり、消費電力の削減の観点からも、必要とされるメモリ量とアクセスバンド幅の削減が必要と考えられた。

メニーコアを生かす細粒度 KPN 方式実装

前述のフレームパイプライン方式の欠点を鑑み、メモリアクセスを削減し、かつ、低消費電力化できる方式として、KPN 方式で、ソフトウェアを実装した。本来理想的な KPN 方式では、各処理を行うプロセス間を１対１単方向、無限長の FIFO（先入れ先だしメモリ）で結び、バケツリレー式のデータ処理を行うことになっている。しかし、実プロセッサ上への実装では無限長の FIFO は不可能であるので、有限長（本実装では最大でも７段である）

の FIFO により代替している。

前述の各機能を KPN の１プロセスとし、その間を以下のような FIFO 構成にて結合した。

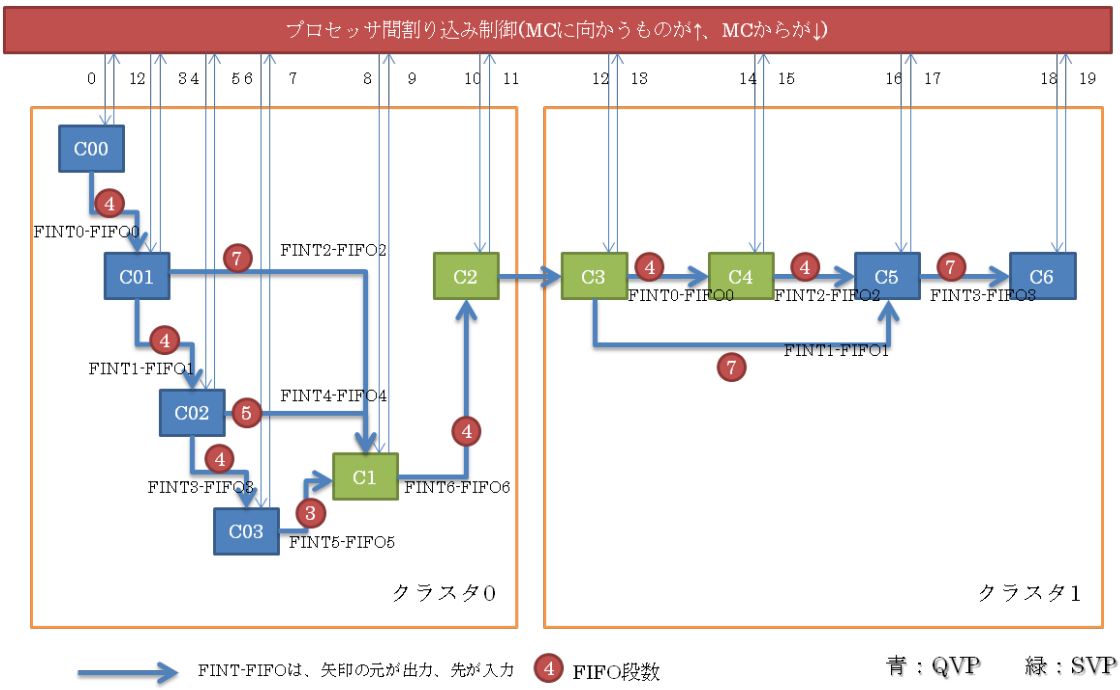


図 3.2- 10　各機能をつなぐ FIFO 構成

このような構成に替えるともに、流れるデータの粒度を細粒度化した。前述のフレームパイプライン方式では、画像フレームを時間単位としたために、各機能間で受け渡しをするデータも画像単位となり、これが１もしくは 2 フレーム期間の間保持されねばならないためにメモリの必要量が増えていた。KPN 方式では、流れるデータの粒度を細粒度とすることで、極端に言えば一ピクセル単位での受け渡しも可能である。処理が終わればメモリは解放できるので、細粒度化することでメモリの所要量を大きく減らすことができる。

しかし、細粒度化で考慮すべきトレードオフがあり、極端な細粒度化も弊害がある。ガウシアンフィルタなど、本アルゴリズムでは注目する画素の周辺範囲を「なめる」操作が繰り返される。ある着目画素からみた他の画素は周辺であるが、逆に他の画素を注目する時には前の着目画素は周辺画素となる。このような関係があるために、処理すべきデータにはオーバラップがある。例えば、周辺縦横１３画素を舐めるガウシアンフィルタでは、１ピクセル単位の操作では毎回 13 x 13=169 ピクセル分ものデータを読み込まなければならず、同じデータを最悪 169 回も読まねばならず膨大なバンド幅を消費する。読むデータと結果データの比をＭｒと置くと Mr=169 となる。ところが２ピクセルを同時に処理することとすれば、高々169+13=182 ピクセルのデータ読み込みで処理は終わる。つまり約 54%のメモリバンド幅、Mr=91 で処理を終わらせることができる。

同時処理数をｍピクセルとおいて、画像端を考えない極限値を前述の 13x13 範囲のフィルタに適用してみると、以下の関係がある。

Ｍｒ＝１＋（１２／ｍ）

これに対して、同時処理数ｍピクセルを達成するために同時にｒピクセルの画素をレジスタ上に持たないとならないと考えると

ｒ＝ｍ＋１２

という関係もある。ｒはプロセッサの持つハードウェアのレジスタ幅である。この関係を画面幅 640 ピクセルに対してグラフ化してみる。

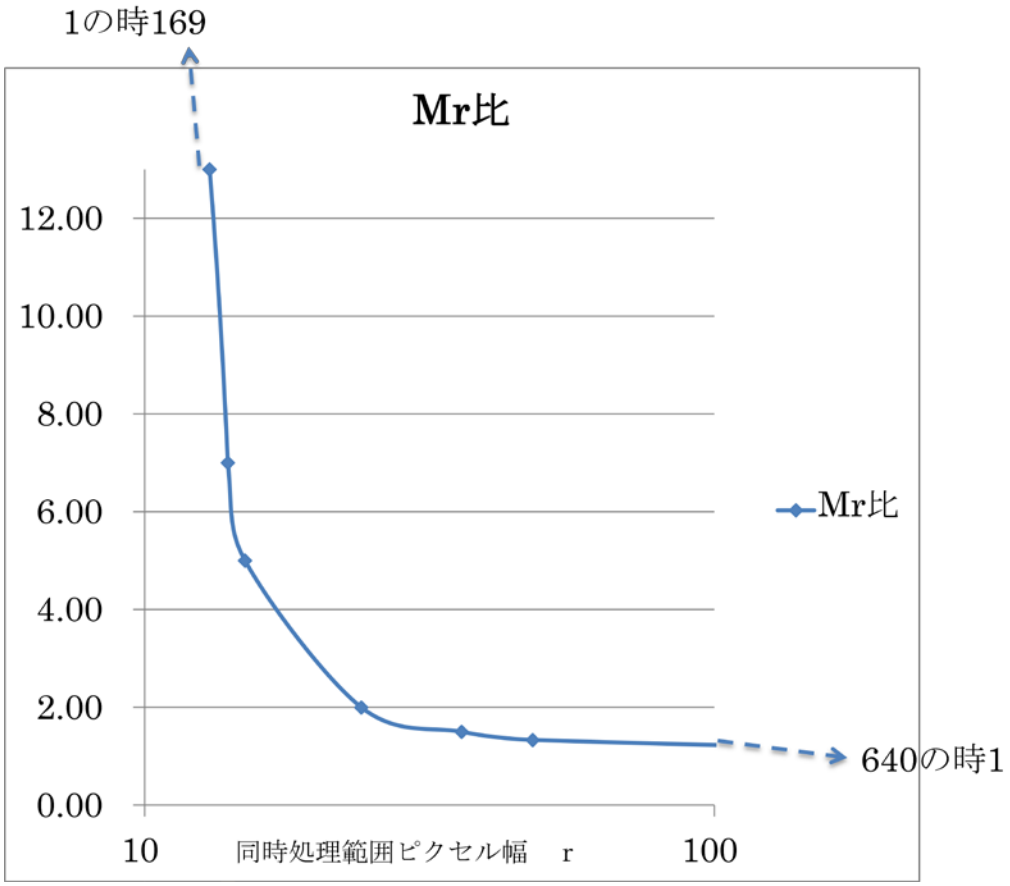


図 3.2- 11　　同時処理幅とメモリアクセスの効率を示す比

1 レジスタに入るピクセル数が増えるに従って効率は 1 に近づく。同時処理幅 12 で比は 2 とクロスするが、それ以降の改善は緩やかとなる。これに対して、プロセッサのレジスタ

の幅はハードウェアの効率上、2 のべき乗とするのが必須である。前述の固定小数点化の検討より、ここでは 1 ピクセルは 16 ビットとしたので、256 ビット幅のレジスタの場合 16 ピクセルを、128 ビット幅のレジスタの場合 8 ピクセルを収容しうる。256 ビットの場合は比 1.75 を達成できるが、128 ビットの場合は 2.5 となる。この比率と処理速度（倍幅ではほぼ倍速の効果が得られる）を検討し、256 ビット幅のレジスタでの処理を決定した。

このような検討を各機能について行い、各コアの処理の単位を決定した。

細粒度 KPN 方式実装の通常型フレームパイプライン実装に対する比較

前述のように KPN 方式により、フレームパイプライン方式よりもメモリバンド幅およびメモリ所要量の点で大幅改善が可能であるが、KPN 方式は遅延時間の改善にも効果がある。

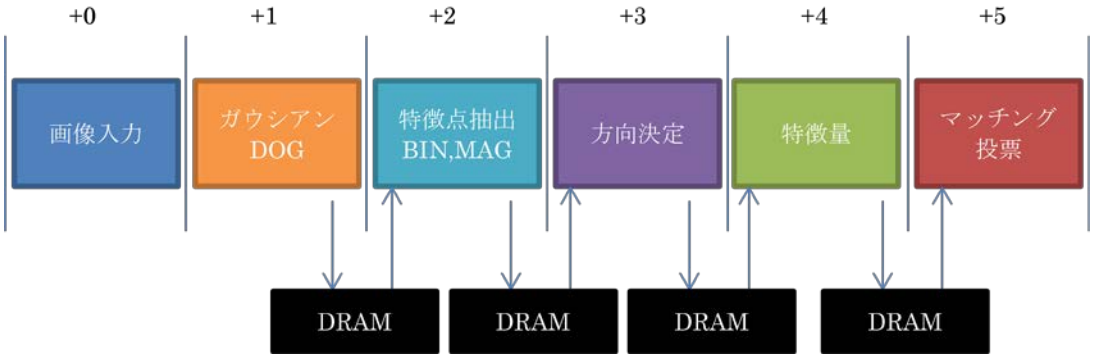


図 3.2- 12　フレームパイプライン方式の模式図

上記図の+0、+1 などは各フレームを表している。フレームパイプライン方式では、各機能はパイプライン化されているが、１つの画像に対しては上記図のように５フレーム時間を要してこのパイプを通過する。これに対して、KPN 方式では、各段処理は再分割されたデータをローカルに受け渡しながら、前の機能ステージとオーバラップして実行されるので、各機能処理が同一の処理時間であるにも関わらず、その遅延時間は小さくなる。この様子を以下の図に示す。

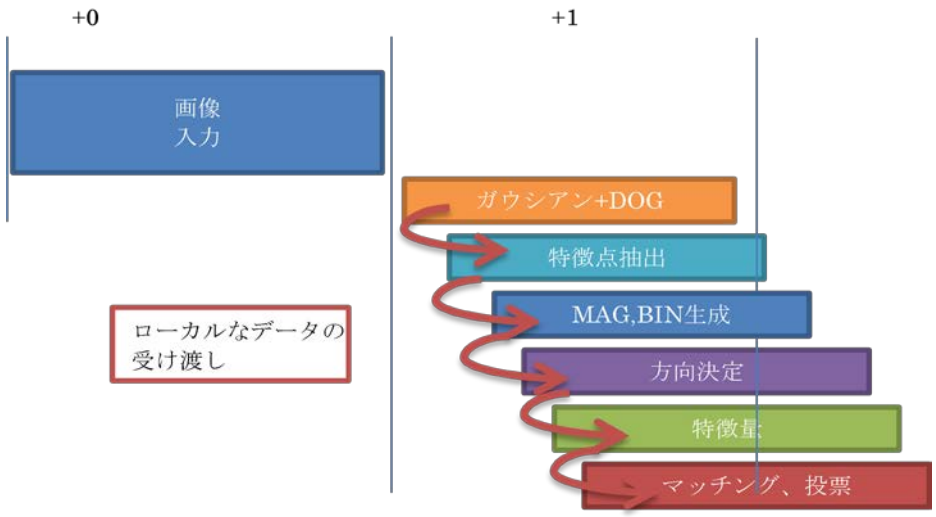


図 3.2- 13　　KPN 方式の模式図

以下に同一構成のメニーコア・マルチクラスタシステム上で、フレームパイプライン方式とＫＰＮ方式のソウトウェア実装の比較表を掲げる。

表 3.2- 3 フレームパイプライン方式と KPN 方式実装の比較

	フレームパイプライン方式	細粒度ＫＰＮ方式
主記憶 DRAM 所要量	22Mbytes	3.1Mbytes
バンド幅（30fps 時）	735MB/Sec	93Mbytes/Sec
バンド幅(15fps 時)	368MB/Sec	46Mbytes/Sec
オンチップメモリ所量	0.15Mbytes	0.8Mbytes
遅延時間(30fps 時)	167mSec	100mSec ＋α
遅延時間(15fps 時)	330mSec	100mSec ＋α

※+αはデータへの依存性がある。

まとめ

2 クラスタにそれぞれ 6 コア、4 コア、クラスタ外におかれた全体制御用の 1 コアの計 11 コアからなる最小構成メニーコアシステム上に、ＳＩＦＴアルゴリズムを搭載し、100MHz

のコア動作速度にて 640*360 サイズのモノクロ動画像に対して 15fps の性能を達成できることを確認した。さらに４クラスタ構成に拡張すれば 30fps への性能向上も可能である。

3.3 消費電力モデルの開発

SMYLEvideo や他のメニーコア・アーキテクチャ及びコンパイラを用いて実現するシステムの性能と消費電力をシミュレーションにより定量的に評価するために、メニーコア・アーキテクチャの消費電力モデルを開発した。消費電力は実行するソフトウェアに大きく依存する。特にマルチコア・プロセッサの消費電力予測は、振る舞いの予測性の低さ、非決定性もあって、難しい。それらの課題を解決するため、実測ベースの電力値や命令実行の統計値を用いた、消費電力予測モデルを開発した。

本モデルは以下の方針で構築した。

- 消費電力はダイナミック電力とリーク成分などのスタティック電力に分けられるが、本モデルではダイナミック電力を対象とする。
- ダイナミック電力は、主としてユニットごとの動作率に比例して増加する。
- 実行するユニットの動作率は命令レベルシミュレーションによる命令実行 MIX によって、算出可能である。
- 特定資源に負荷を集中させるテストプログラムにより、資源の稼働率を RTL シミュレーションから、消費電力は実チップによる動作を実測し、稼働率と消費電力の関係の一次式を作成する。
- 命令レベルシミュレーションによって求めたアプリケーションの命令実行 MIX による、各ユニット稼働率を実測から求めた一次式に適用して、アプリケーションの消費電力を予測する。

命令シミュレーションによる命令 MIX の取得に関しては、3.4 節で述べる。また、消費電力一次式を構築するために、実測を行ったのは、平成２１年度～平成２３年度に NEDO 委託で行われた省エネルギー革新技術開発事業／先導研究／省エネ情報機器のための超並列バスによるヘテロジニアス・マルチチップ積層 Cool System の研究開発で試作された C0/C1 チップを用いた[Matsumoto 2012]。

表 3.3- 1 に示す。

表 3.3- 1　C0/C1 チップ諸元

Name	C0	C1
Process	TSMC 180nm 6M1P Logic Low Power (CL018LP)	
Die Size	7.0 mm x 4.0 mm	10.2 mm x 9.0 mm
Power Supply	core: 1.8 V, I/O: 3.3 V	
Frequency	50 MHz	30 MHz
Gate count (2NAND)	184.5 Kgate	1254.6 Kgate
Hard Macros	RAM, TSV	
Pad count	208	304
Package	LQFP 256	HQFP 304
Power Dissipation	193.796 mW	318.983 mW

本チップに対して、各ユニットの稼働率をばらつかせた 14 本のプログラムを動作させて、演算コア DPE の消費電力を求める一次式を求めた。結果を表 3.3- 2 に示す。

表 3.3- 2　消費電力算出一次式

ユニット	電力依存分 α	固定値成分 C
DPE: IF ステージ	7.31	0.00
DPE: ID レジスタバンク以外	7.26	-0.05
DPE: ID C レジスタバンク	1.50	0.03
DPE: ID D レジスタバンク	7.14	5.44
DPE: EX ステージ ALU	1.92	0.30
DPE: EX ステージ DPU	1.81	0.12
DPE: EX ステージ MAU	7.53	1.55
DPE: MA ステージ	3.55	2.98
L1 命令 SPM	20.79	2.34
L1 データ SPM	21.31	0.35
バスドライバ	0.50	1.11

消費電力予測値はユニットごとの動作率の積算に基づき、以下の式で求められる。

$$Power[mW] = \Sigma \{\alpha \times 動作率 A + C\}$$

本パラメタを用いて逆量子化プログラムを動作させ、計算で求めた消費電力との比較を試

みると、以下のようになった。

実測値　26.3mW
予測値　32.1mW

表 3.3- 3　逆量子化プログラム消費電力予測値

ユニット	稼働率[%]	消費電力[mW]
DPE: IF ステージ	28.3%	2.07
DPE: ID レジスタバンク以外	63.1%	4.53
DPE: ID C レジスタバンク	24.6%	0.40
DPE: ID D レジスタバンク	33.4%	7.83
DPE: EX ステージ ALU	30.4%	0.89
DPE: EX ステージ DPU	2.0%	0.16
DPE: EX ステージ MAU	13.7%	2.58
DPE: MA ステージ	11.9%	3.40
L1 命令 SPM	28.3%	8.23
L1 データ SPM	1.9%	0.75
バスドライバ	28.3%	1.26
合計(180nm TSMC 30MHz)		32.08

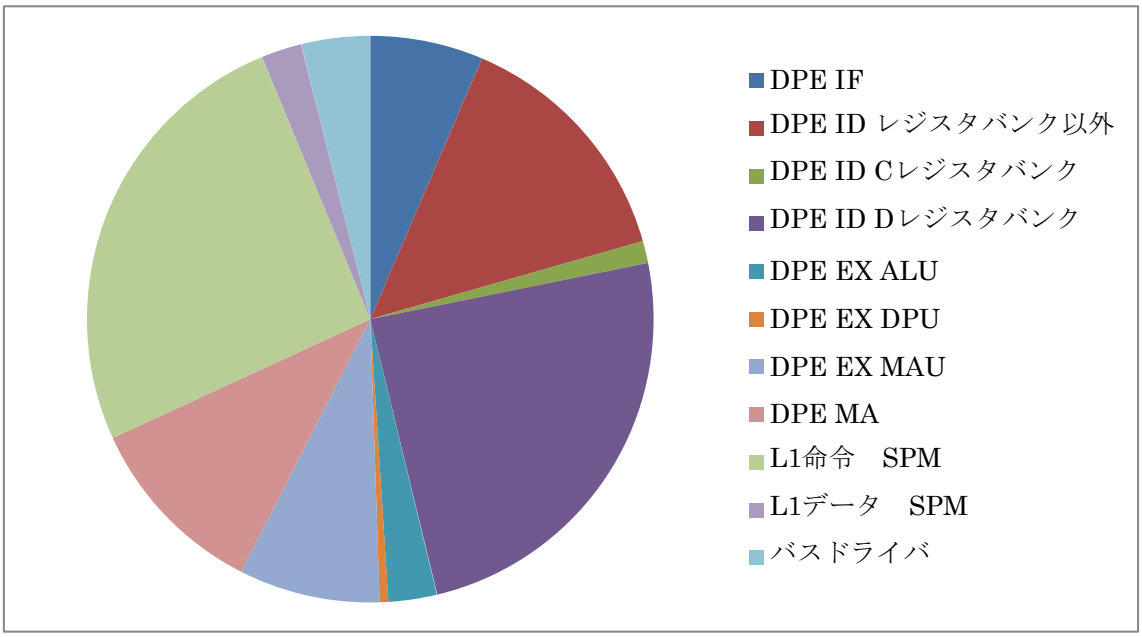


図 3.3- 1　逆量子化プログラム消費電力予測の内訳

消費電力の予測値と実測値には約 19%の誤差が生じているが、目標であった 20%以内に収まっている。ID ステージの D レジスタバンクが大きめに出ているのは、FF で構成された D レジスタのクロックゲーティングがレジスタバンク単位に行われていないためであり、SMYLEvideo では、省電力アーキテクチャの深度化が必要であることが分かった。他のコードに関する様々な検証を行い、予測モデルの精度の向上、確からしさの検証は今後の課題である。さて、上記の式に基づき、SMYLEvideo における SIFT の初段処理であるガウシアンフィルタの消費電力を予測した。結果を表 3.3- 4 と図 3.3- 2 に示す。

表 3.3- 4　ガウシアンフィルタ消費電力予測

ユニット	稼働率[%]	消費電力[mW]
DPE: IF ステージ	17.1%	1.25
DPE: ID レジスタバンク以外	100.0%	7.20
DPE: ID C レジスタバンク	9.4%	0.18
DPE: ID D レジスタバンク	91.9%	12.00
DPE: EX ステージ ALU	59.1%	1.44
DPE: EX ステージ DPU	21.9%	0.52
DPE: EX ステージ MAU	8.6%	2.19
DPE: MA ステージ	1.9%	3.05
L1 命令 SPM	17.1%	5.89
L1 データ SPM	0.9%	0.54
バスドライバ	17.1%	1.20
合計(180nm TSMC 30MHz)		35.46

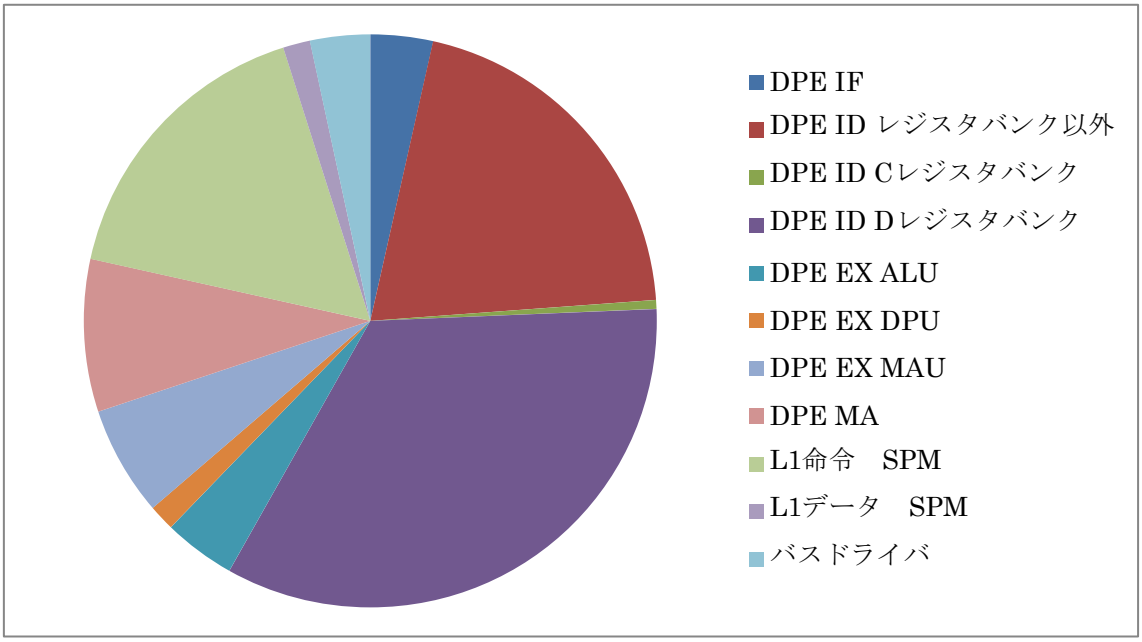


図 3.3- 2　ガウシアンフィルタ消費電力予測の内訳

本試作チップのデータでは、レジスタバンクに対して、バンク毎のクロックゲーティングが行われておらず、フィルタ処理で参照比率が高く、稼働率が 90%を超える D レジスタバンクの消費電力が非常に大きくなっている。それ以外には、命令デコード(ID)、ならびに L1 命令 SPM の電力が目立つ。これは、今回の予測のベースとなったチップも TOPSTREAM アーキテクチャに準じており、本アーキテクチャが一度とってきたデータを有効利用して処理することから、データよりも命令における消費電力が増加する傾向を示すためである。

一方で、SIFT 処理において、もっとも負荷の高いガウシアンフィルタ処理の 1st ステップが 180nm プロセスにおいても、高々35mW で処理できるというのは、TOPSTREAM™ の低消費電力特性を示している。

3.4　メニーコアシミュレータ開発

3.4.1 メニーコア対応インストラクションシミュレータ

インストラクションシミュレータの機能

インストラクションシミュレータは、パーソナルコンピュータ上で、メニーコアプロセッサ向けに書かれたプログラム（機械語命令コード）を模擬的に実行するためのソフトウェアである。主としてコアプロセッサおよびメモリをモデル化したものであり、ごく一部のハードウェアを除いて、入出力デバイスなどは含まれない。動画像の処理に適用する場

合、あるメモリ領域を画像バッファとして想定し、そこに画像データを配置することで処理を行う。結果データもあるメモリ領域に格納する。

メニーコア向けに書かれたプログラムのオブジェクトコードファイルは、インストラクションシミュレータにより読み込まれ、実際のプロセッサと同様に１命令ずつ実行され、その命令の実行によりシミュレータ内部のプロセッサレジスタの内容やメモリの内容が逐次変化していく。

本インストラクションシミュレータの特徴としては、メニーコア対応のため、全１１コアのプロセッサコアを同時並列に動作させることができることがあげられる。

プログラムのデバッグ時には、実際のプロセッサでのソフトウェアのデバッグと同様、ブレークポイントをかけて実行を止めてレジスタやメモリ内容を確認したり、１命令ずつトレースしたりする機能を駆使する。

プログラムの性能評価時は、デバッグ済のプログラムを使って、バッチ処理をおこない、その間のプログラムの挙動がログ出力されるので、これをもとに行う。

以下にデバッグ時に使われる、インタラクティブなデバッガモードでのインストラクションシミュレータの画面を示す。

画面左上のウインドウは全１１個のコアのどのコアを表示させるか制御するための制御パネルである。左中央、および中央の２つのウインドウは、データ処理コアの１つ目と２つ目が実行中の命令とそのときのレジスタやプロセッサの状態を表示するためのウインドウである。中央のウインドウに半分隠れているウインドウはメモリ領域をダンプするウインドウであり、このプログラムの使用するデータが置かれる部分をモニタしている。右下のウインドウは、プロセッサの動作を記録したログを表示するためのものである。

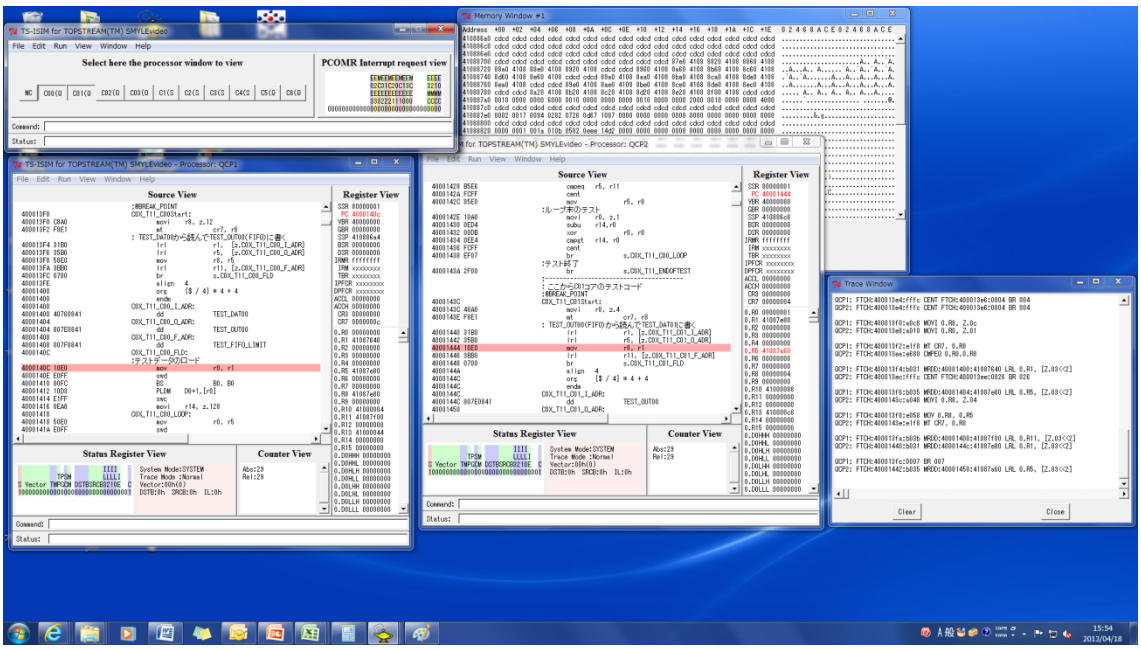


図 3.4- 1　インストラクションシミュレータ（インタラクティブモード）

メニーコア上の KPN 方式ソフトウェアへの対応

多数のプロセッサコアが FIFO により接続され動作するために、本インストラクションシミュレータには、ハードウェアによるゼロオーバヘッドプロセッサ間通信機構を模擬する機能が含まれている。上記のシミュレーション例であると、左側のウインドウのプロセッサが右側のプロセッサに向かって FIFO 通信している。もし右側のプロセッサの処理が遅くて、FIFO に空きが無くなっている状態で、左側のプロセッサが FIFO に書き込もうとする命令を実行しようと試みると、左側のプロセッサにインターロックがかかって、右側のプロセッサが FIFO に空きを作るまで自動的に待たされる。また、左側のプロセッサの処理が遅くて、FIFO が空の状態で、右側のプロセッサが FIFO を読もうとする命令を実行するような場合では、今度は右側のプロセッサが自動的に待たされる。この機構により、KPN 方式で書かれたソフトウェアがメニーコア上で正しく実行されることが担保されている。

インストラクションシミュレータの実装

本インストラクションシミュレータは、大きく以下の３部分に分かれる。

① コアプロセッサの動作をシミュレーションするシミュレータ本体
② シミュレータの動作をインタラクティブに制御する GUI デバッガ
③ シミュレータ動作から性能、消費電力などを読み取りグラフ化するプロファイラ

今回実装ではマイクロソフト社のウインドウズ環境向けにシミュレータを作成したが、Ｌｉｎｕｘ等の環境にも容易に移植可能なように実装されている。

3.4.2 メニーコア対応インストラクションシミュレータによる性能、消費電力評価

インストラクションシミュレータによる性能評価

インストラクションシミュレータのプロファイリング機能を使うことで、メニーコア向けに作成されたプログラムの実行サイクル数などを正確に見積もることができる。実際には、プログラムを作成したのち、プロファイリング機能を使って性能的に問題のある部分を発見し、その部分に対策を施すことで所望の性能を達成する。このような目的にシミュレータのプロファイリング機能を使った例を示す。

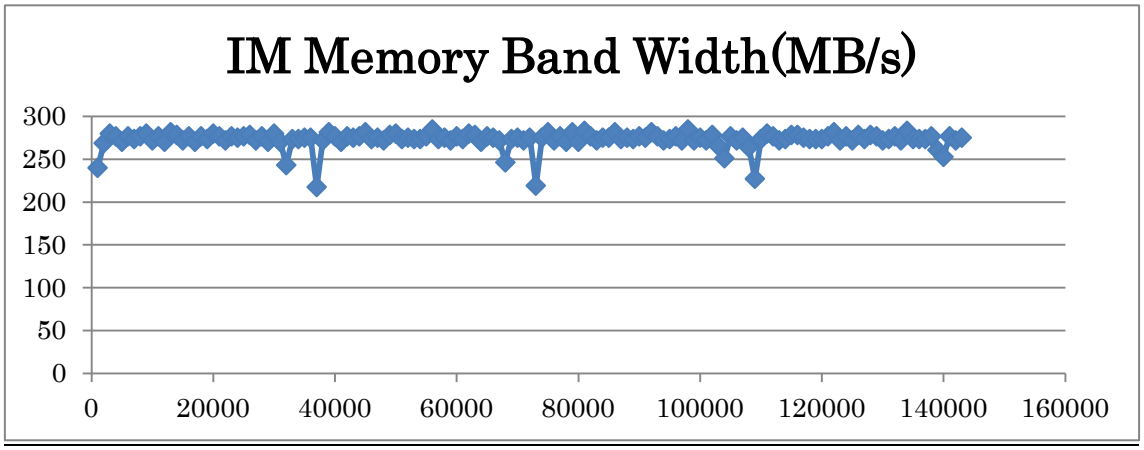


図 3.4- 2　命令メモリへのアクセスに使われているバンド幅

図 3.4- 2 は、x 軸に命令実行数（実行サイクル数とほぼ等しい）、縦軸にあるコア上のプログラムを単体上で走らせたときに命令フェッチに必要であったメモリバンド幅をプロットしたものである。このクラスタの場合、1 個のコアあたりに割り当て可能な命令フェッチ用のメモリバンド幅は約 267M バイト毎秒であり、インストラクションシミュレータのシミュレーション結果の平均では 273M バイト毎秒と約 2%ほどリミットを超過しており、複数コアで協調動作させた場合には、その分の速度低下を折り込んで性能見積もりする必要があることが分かる。

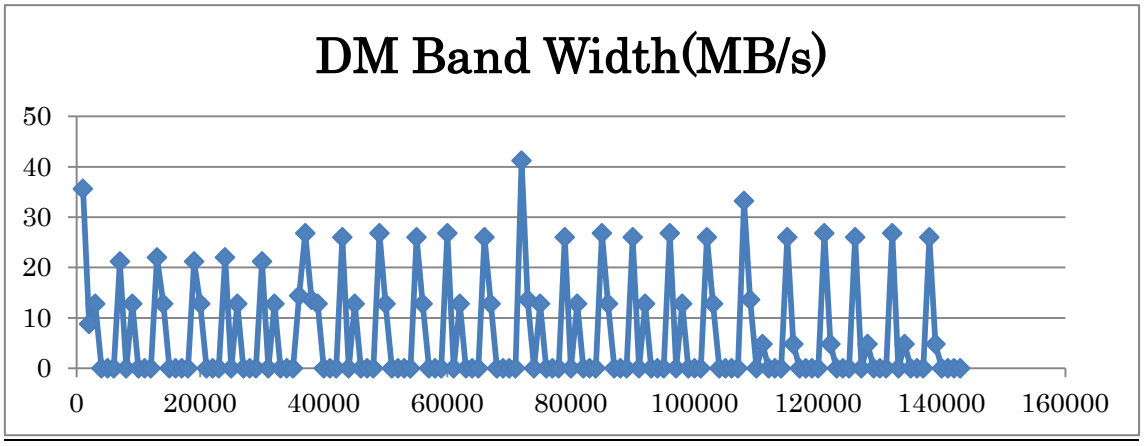


図 3.4- 3　データメモリへのアクセスに使われているバンド幅

図 3.4- 3 は、図 3.4- 2 と同期間のデータメモリへのアクセスに必要なバンド幅をプロットしたもので、こちらはピークでも 40M バイト毎秒程で割り当てリミットの約 267Mバイト毎秒には届かないことが分かる。平均すれば 13M バイト毎秒なので余裕があることが分かる。

インストラクションシミュレータによる消費電力評価

また、プロファイリング機能の中には消費電力評価に使用するための機能も含まれている。プログラムを実行したときに動作するコア内の各ブロックの稼働状況を知らせる機能である。

実際のプログラムにこの機能を適用し、コア内の各ブロックの稼働状況をチェックした例を示す。

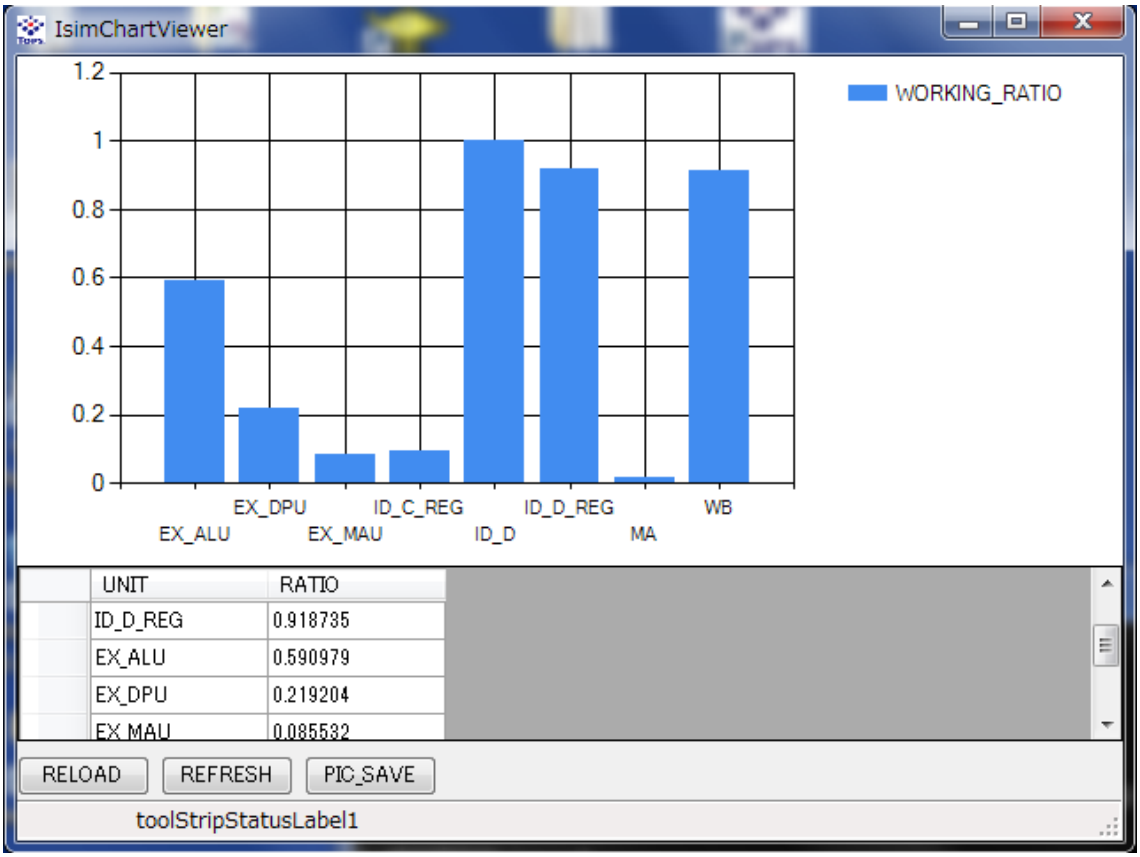


図 3.4- 4　プロセッサ各部の稼働率

図 3.4- 4 はプロセッサコアのハードウェア各部の稼働率を示したものである。演算系は、EX_ALU, EX_DPU, EX_MAU の 3 系統に分けられている。命令デコード系はレジスタアクセスの ID_C_REG, ID_D_REG と実際のデコード論理の ID_D の 3 系統に、メモリアクセスのためのハード MA と、レジスタ書き込みのハード WB に分類されている。ターゲットにしたプログラム例では命令デコードは 100%稼働しており、データレジスタを使った ALU 演算（加算）などが多くを占めることが分かる。これにたいしてメモリアクセス命令は極めてすくない。また、命令については、そのカテゴリ別に集計する機能もある。

図 3.4- 5 は命令別の集計機能の例である。算術論理演算についてはすべて ARITH カテゴ

リにまとめられており、バンク操作や FIFO 待ち合わせのための PFX カテゴリがそれに続いている。MOV カテゴリはデータの転送命令であり、CONT カテゴリは分岐命令などの制御のための命令である。LDST カテゴリのロード、ストア命令もわずかに存在するが、ごく少数であることが分かる。

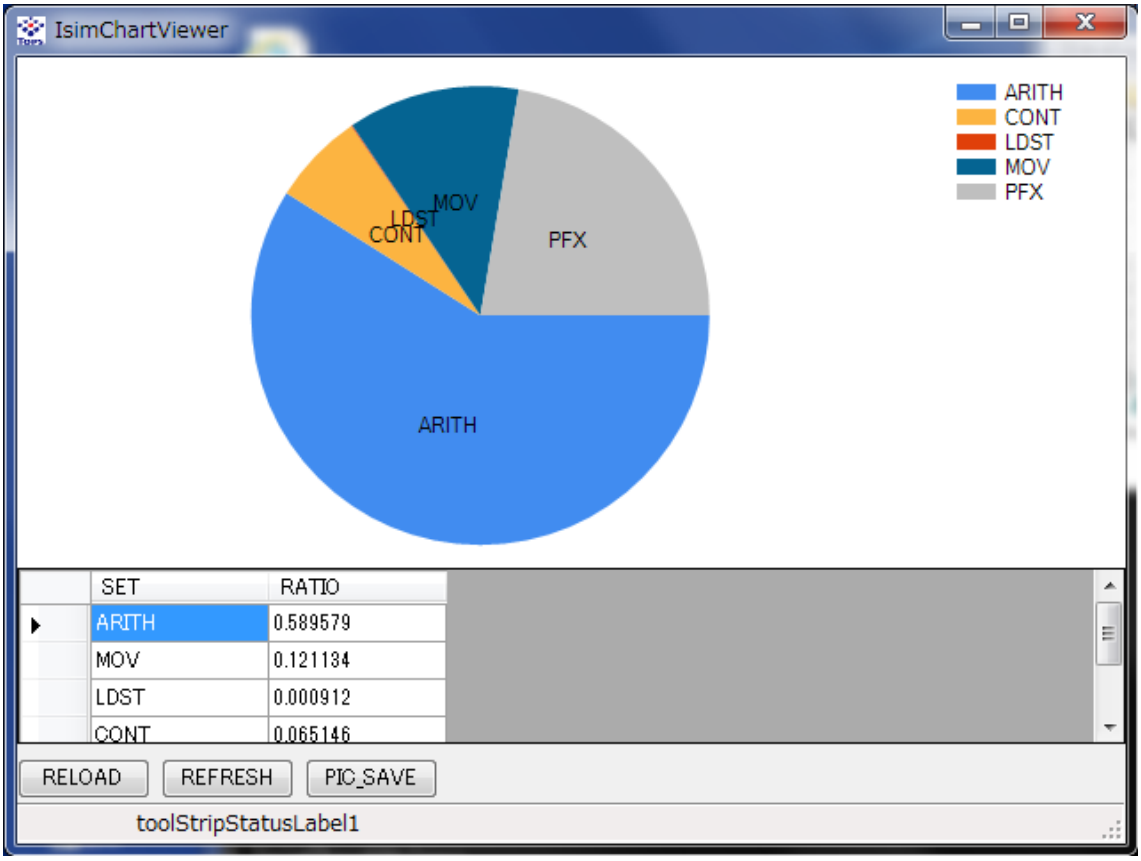


図 3.4-5　あるプログラムの命令 MIX

3.5　SMYLEvideo 総合評価

本節では、SMYLEvideo の評価を行う。SMYLEvideo の評価は、処理性能、消費電力、ハードウェア規模の 3 つの視点から行う。

SMYLEvideo の最小構成は 2 クラスタで、合計 10 個の演算コア DPE が集積される。コアごとに、256 ビットの SIMD 演算器、レジスタバンクを持つ QVP と 64 ビット構成の SVP から構成される。表 3.5-1 に、各 DPE コアの演算性能を示した。

表 3.5-1　各 DPE の演算処理性能(100MHz 動作)

	GOPS 値 (16b　Intel 換算、ピーク)	個数 2 クラスタ	合計 GOPS

QVP	96	6	576
SVP	24	4	96
MC	0.1	1	0.1
合計			672.1

QVP、SVP は、Intel 換算で約 60 ステップを 1 サイクルで実行するガウシアンフィルタ関係の複合命令を具備しており、演算のピーク値はそれに従って求めた。これによると、SMYLEvideo の 2 クラスタ構成では、672GOPS のピーク性能、4 クラスタ構成では、１TOPS を超える性能を達成可能である。

一方、SIFT 実行時の消費電力について検討する。SIFT 処理は、特に後段の特徴点解析フェーズが特徴点数に大きく依存する。ここでは、負荷が重い特徴点が 2000 程度の VGA 画像解析について検討する。表 3.3- 4 に示したガウシアンフィルタによる DOG(Difference Of Gaussian)処理は、処理速度は特徴点数に依存せず、画像全体をスキャンする処理である。この処理は特徴点 2000 の場合、SIFT 処理全体のおよそ 5%となる。したがって、処理量比例と考えると SIFT 処理全体の消費電力は、おおよそ 710[mW]となる。今回の実装では、それに、MC および周辺回路の消費電力が加わるため、先述の実測チップの値から、合計およそ 850[mW]と予測される。しかしながら、この値は 30MHz180nm 1.8V TSMC の値であるため、ターゲット周波数 100MHz、40nm 1.1V TSMC に変換する必要がある。こ の場合、ロードキャパシティは 1/3 となると仮定すると、SMYLvideo2 クラスタ構成 SIFT 実行時の消費電力は約 350[mW]と予測される。また、チップサイズは、3.2.1 で検討したように 1.7mm□である。

これらの値は、従来型の汎用マルチコア・プロセッサ(4 コア(8 スレッド)約 2.9GHz≒24GOPS,130W)と比較して、性能が約 30 倍（700GOPS 相当)、消費電力 1/300 以下（350mW 程度)、商用の専用ハードウェアアクセラレータを具備した４コア程度の画像認識プロセッサに対して、性能：1.7 倍、消費電力：1/2 以下、回路規模：1/2 以下であると言える。

参考文献

[Kahn 1974] Kahn, G. (1974). The semantics of a simple language for parallel programming. In Jack L. Rosenfeld (Ed.): Information Processing 74, Proceedings of IFIP Congress 74, Stockholm, Sweden, August 5-10, 1974. North-Holland, 1974, ISBN 0-7204-2803-3

[David G. Lowe 1999] David G. Lowe, "Object recognition from local scale-invariant features," International Conference on Computer Vision, September 1999, pp. 1150-1157.

[Matsumoto 2012] Yukoh Matsumoto, Tomoyuki Morimoto, Michiya Hagimoto, Hiroyuki Uchida, Nobuyuki Hikichi, Fumito Imura, Hiroshi Nakagawa and Masahiro Aoyagi, “Cool

System Scalable 3-D stacked Heterogeneous Multi-Core / Multi-Chip Architecture for Ultra Low-Power Digital TV Applications, COOL CHIPS XV April/20 2012.

4　メニーコア向けソフトウェア開発環境

本プロジェクトで開発したメニーコアアーキテクチャ SMYLEref の普及を想定し、これらのメニーコアプロセッサの普及が及ぼすソフトウェア開発の課題を検討し、その課題へのアプローチとし次節で説明する 3 種類のソフトウェア開発支援ツールを開発した。

現状、幅広く普及しているメニーコアプロセッサは存在していない。市場(応用分野)にとってはコアの数は特に意識する必要のないことであるため、今後もメニーコアにより新たな市場が創出することは考えにくい。既存プロセッサが適材適所でメニーコアに置き換わることが想定される。既存プロセッサの進化に伴い現状注目されつつあるソフトウェア開発現場での新たな文化、「移植」「性能予測」「最適化」に着目した。

次節でこれらのキーワードの説明と、これらに対するアプローチとして開発したツールを紹介する。また開発したソフトウェアが幅広くメニーコアプロセッサを搭載したシステムでの利用が可能なように、SMYLEref でのみの利用に限定せずに、現在巨大な市場を獲得している Intel 社の汎用プロセッサや NVIDIA 社の汎用 GPU での利用も可能なツールとして開発した。

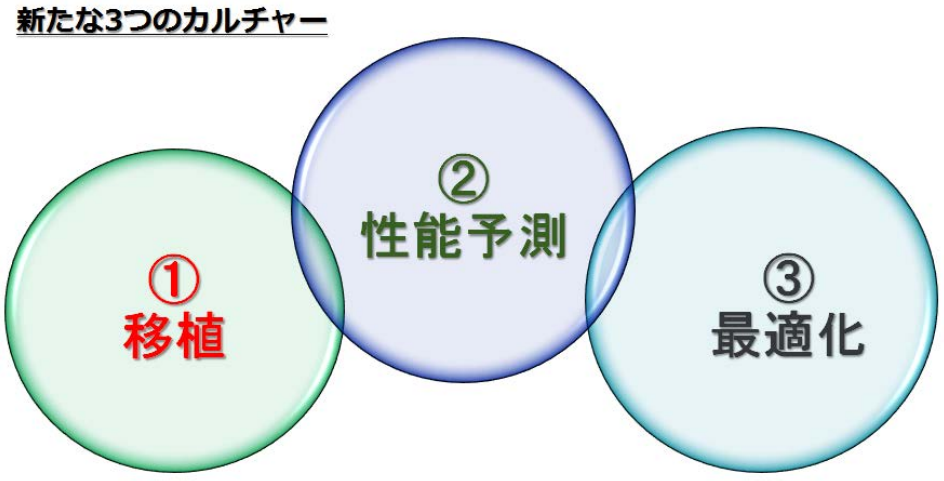


図 4-1　メニーコア時代の新たな 3 つのカルチャー

4.1　半自動並列化コンパイラ CLtrump

4.1.1 背景

CLTrump はメニーコア時代の「移植カルチャー」に対する支援ツールである。これまでのソフトウェア業界におけるソフトウェア開発では仕様書に基づいたソフトウェアの開発が主であった。ソフトウェアベンダは発注元からソフトウェア開発に必要な仕様書を受け取り、それに従ってソフトウェアを作成して納品していた。

一方、メニーコア時代の到来を背景に、近年ソフトウェア業界に根付きはじめている「移植カルチャー」とは、具体的には「直列プログラム」を「並列プログラム」に書き換えるという移植作業である。これは既存のソフトウェア資産の多くはシングルコアプロセッサが主流の時代に作成されたものでありプロセッサの進化に伴いその恩恵を受けられなくなってきているからである。また並列プログラムの作成するためには最初にリファレンスコードとなる直列プログラムを作成してから、それを並列化するというステップが一般的である。すなわちソフトウェアの並列化はメニーコア時代のプログラマにはなくてはならない技術スキルとなる。

4.1.2 CLTrump: C to CL Translation Utilities for Mancore Processor

従来の汎用プロセッサと比較してメニーコア時代のプロセッサは、プログラマの立場からみると、ローカルメモリサイズやワークアイテム数/レジスタ数、メモリレイテンシの隠蔽といったアーキテクチャ依存の最適化要素が多いことが特徴としてあげられる。

この最適化には大きく分けて 2 つのやり方が考えられる。

1. 手動で時間をかけて最適化
2. 自動最適化や pragma を使って手軽に最適化

この 2 通りのやり方にはそれぞれメリットとデメリットがある。1 の場合、プログラマの能力を最大限発揮して高い性能が見込まれる反面、レジスタ数の確認やループアンローリング、命令スケジューリングの変更といった時間のかかる作業が多く、作業効率が悪くなる。2 の場合、OpenMP, HMPP といった代表的なフレームワークがあるが、処理が複雑で適応できない場合があり、細かな最適化ができないといったデメリットがある。

このように既存の最適化手法では手動最適化か自動最適化を排他的に選択しなければな

らないという問題がある。そのため手動最適化では自動最適化の恩恵である作業効率の向上が受けられず、自動最適化では細部の最適化ができない。

一方メニーコア向けプログラムの開発作業においては、自動並列コンパイラといった技術が優位となる自動並列可能な部分とアルゴリズム選定やデータ構造の決定といった人が介入した方が効率的な部分が混在している。本節で提案する CLTrump は自動化可能な部分はツールで自動化し、手動化が必要な部分はユーザが自ら開発するようなインタラクティブな半自動並列プログラム開発環境を提供するツールである。図 4.1-1 に従来の開発方法と CLtrump での開発方法の比較図を示す。

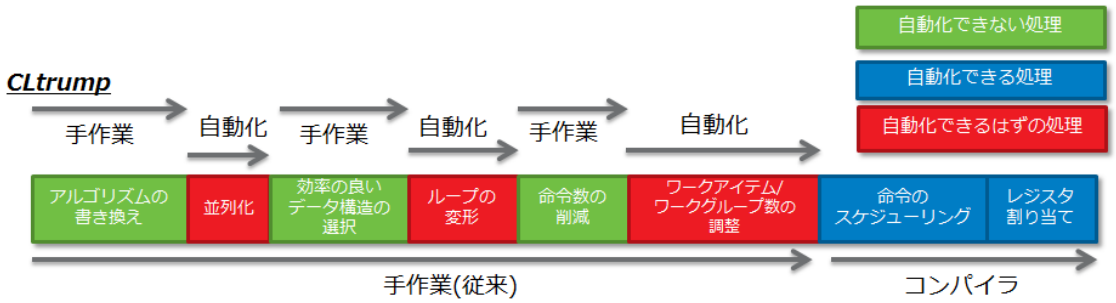


図 4.1-1 従来の開発方法と CLtrump での開発方法の比較

図 4.1-2 に CLtrump を使った開発フローのイメージを示している。開発工程において自動化部分と手作業の部分が混在しているためインタラクティブに並列コードを作成することが特徴的である。プログラマは標準 C 言語で記載された直列コードを CLtrump で OpenCL 仕様[Khronos2012]の並列コードに変換する。変換時には並列箇所の提案と承諾可否の確認が入るため、より高品質な並列コードを作成するために直列コードを編集できる。また CLtrump 自体はソースからソースに変換するソースジェネレータであるため、変換後のコードもソースコードとしてプログラマが解読でき、さらなる最適化を促すことができる。

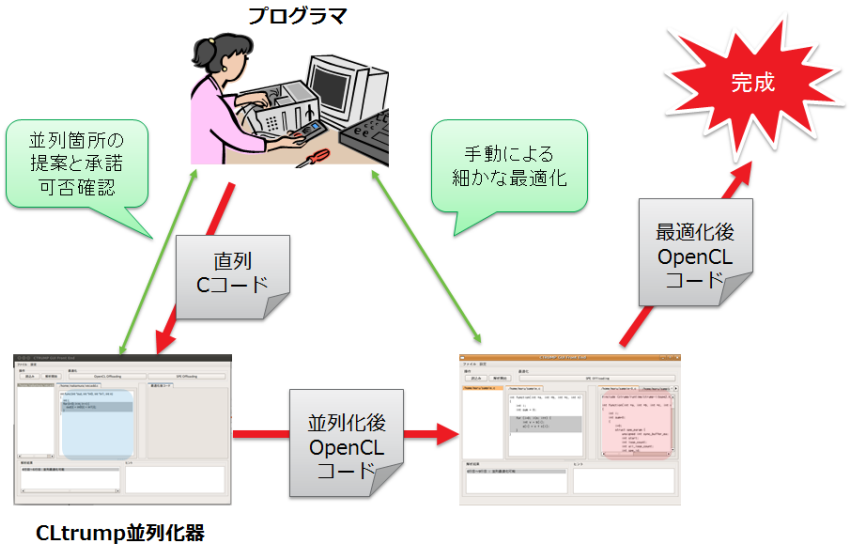


図 4.1-2 CLTrump を使った開発フローのイメージ

Ctrump (C to C TRanslation Utilities for Multicore Processors)

CLtrump は半自動並列化支援ツール Ctrump[産総研 2009](C to C TRanslation Utilities for Multicore Processors をベースとして開発している。

Ctrump は株式会社フィックスターズと産業技術総合研究所、株式会社コーエーおよび株式会社バンダイナムコゲームスが共同で実施したプロジェクトであり PLAYSTATION3 のプロセッサとして有名な Cell Broadband Engine(Cell/B.E.)向けのソフトウェアの作成を支援するツールである。Cell/B.E.は 1 個の汎用プロセッサと 8 個のベクタプロセッサが 1 チップに含まれる 9 コアのプロセッサである。

Ctrump はプログラムの字句・構文解析をするパーサ部分、並列性解析をする並列解析モジュール、解析結果に基づいて Cell/B.E.向けのプログラムを生成するコード生成モジュール、およびそれらをユーザが簡便に利用するためのユーザインタフェースにより構成されている。

CLtrump では Ctrump のパーサ部分や並列解析モジュールをそのまま活用し Cell/B.E.向けの並列プログラムを生成する代わりに OpenCL の並列プログラムを生成している。

4.1.3 CLtrump の利用方法

本節では、CLtrump の動作環境をセットアップする手順を説明する。ここでは例として標準的な構成でインストールされている i386 版の Ubuntu 8.04.2 デスクトップエディションに CLtrump をセットアップする際の説明をする。

インストール

1. CLtrump のソースコードをダウンロードして展開する。

```
$ wget http://sourceforge.net/projects/ctrump/files/cltrump-1.0.tar.gz
$ tar -xf cltrump-1.0.tar.gz
```

2. ワーキングコピーに移動して cmake を実行。ソースディレクトリ以外でのビルドはサポートしていないため cltrump-1.0 以下で実行する。

```
$ cd cltrump-1.0
$ cmake .
```

3. ビルドしてインストールします。python のライブラリをインストールする必要があるため、Python をインストールしたユーザと同じ権限(通常は root 権限)が必要となる。

```
$ make
$ sudo make install
```

対象の Linux 環境をソフトウェア開発に用いたことがない場合、以下に述べるものに加えて cmake, g++, python-dev, sun-java6-jdk などのパッケージをインストールする必要がある。

GUI から使う

PyQt4 が利用可能な環境の場合、GUI から CLtrump を使うことができる。
ctrump_gui を実行すると図 4.1-3 のウインドウが起動する。

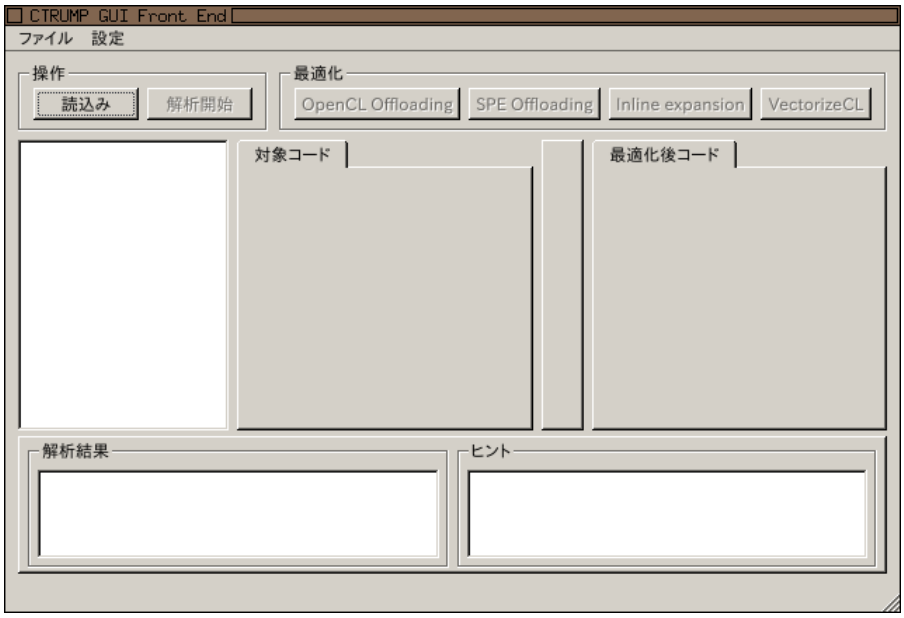


図 4.1-3　CLtrump 起動画面

```
#include <stdio.h>
int a[1024];
int b[1024];
int c[1024];

int
func(int n)
{
   int i;
   for (i=0; i<n; i++) {
      a[i] = b[i] + c[i];
   }
}

int
main()
{
   int i;
   for (i=0; i<10; i++) {
      a[i] = i*2;
      b[i] = i*3;
      c[i] = i*4;
   }

   func(1024);

   for (i=0; i<10; i++) {
      printf("%d %d %d¥n", a[i], b[i], c[i]);
   }

}
```

図 4.1-4 ソースコード

サンプルとして図 4.1-4 に示される逐次コードをどこかに保存する。

図 4.1-5 のウインドウで「読み込み」ボタンを押して、左に表示されたファイル名をダブルクリックしたあと、「解析開始」ボタンを押す。

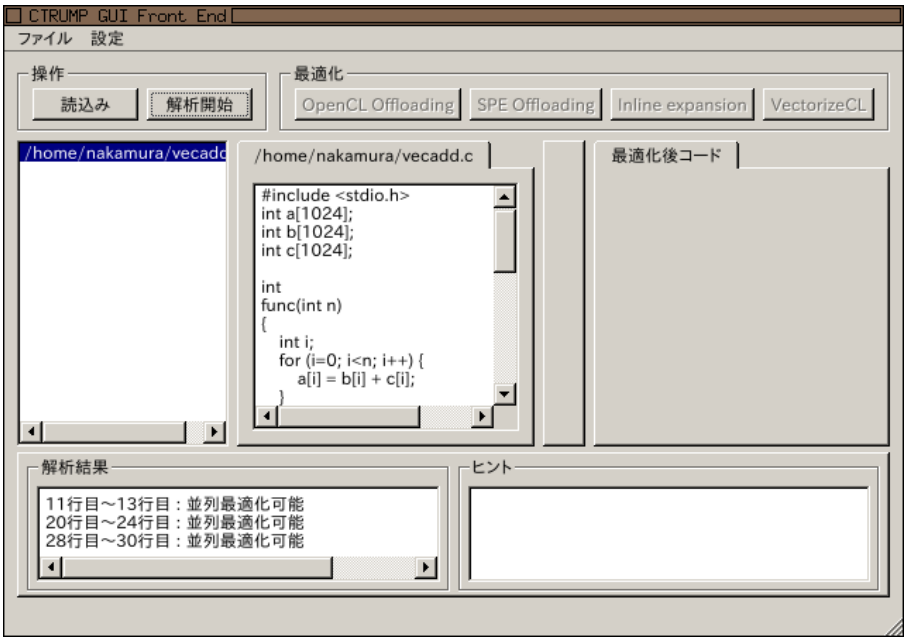


図 4.1-5 逐次コードの読み込みと解析結果表示

左下に行番号を示す情報が表示されるので、これをクリックし最初のループを選んだあと、右上の「OpenCL Offloading」を押し、「文字列として出力する」にチェックを入れ、「OK」を押す。

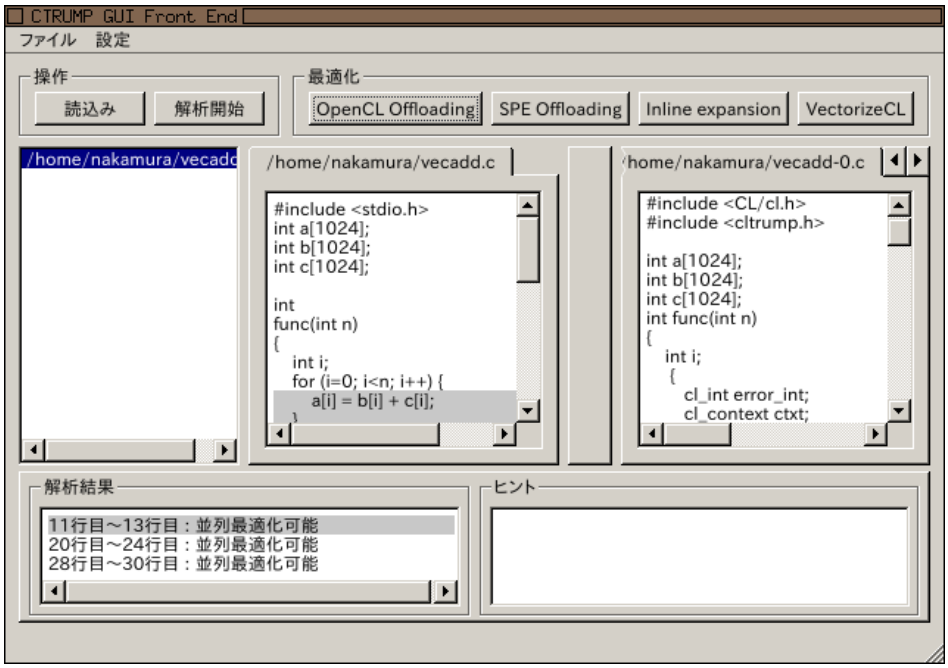


図 4.1-6　OpenCL 並列コードの出力

図 4.1-6 のように右側にファイルが表示されたことが確認できる。これが OpenCL 化されたソースコードになる。メニューから「ファイル」→「最適化ファイルをパスを指定して保存」を選び、ふたつのファイルを保存する。ひとつめのファイルがホストコード、ふたつ目のファイルがデバイスコードとなる。

保存したファイルを以下のようにしてビルドする。

```
$ gcc -O2 vecadd-0.c  vecadd-1.c  -lcltrump-runtime -lOpenCL
```

これで実行可能なファイルが完成する。

現在 CLtrump は後述する並列化機向けベンチマークである BEMAP[FIXSTARS2013]の一部のコードで動作検証が完了している。

4.2　性能推定ツール PEMAP

4.2.1 背景

PEMAP はメニーコア時代の「性能見積もりカルチャー」に対する支援ツールである。メニーコア導入における最大の魅力は性能向上である。そのため、既存のソフトウェアをメニーコアに移植する前にある程度の性能予測をすることが多い。産業界では最終的な性能がある程度見込まれてから予算が確保できるケースが多いため、なるべく低コストでの見積もり作業が求められる。しかし見積もり作業には詳細なコード分析のため時間が必要となり、また体系的な手法も確立されていないため見積もるエンジニアによってもその精度がバラバラという問題がある。

PEMAP では、アーキテクチャの異なるプロセッサ間の移植において、移植前と移植後の命令セットの頻度分布やメモリアクセスパターンに何らかの関連性があることに着目し、移植前のプログラムから、その特性を維持した移植後のダミーコードを自動生成する。ダミーコードは計算結果の合致を犠牲にする変わりにプログラム特性を維持する。移植後の対象プロセッサでダミーコードを実行することで性能の目安となる見積もりが可能となる。

4.2.2 Performance Estimator for MAny-core Processors

本節では PEMAP の動作の概要を述べる。図 4.2-1 は PEMAP のワークフローの全体概要を示した図である。PEMAP のユーザは、予測の対象とする C/C++ソースコード中の、並列化の対象となるループの前後に図 4.2-2 のようなアノテーションを付与し、x86 命令にコンパイルし、実行することで PEMAP を起動する。このアノテーションは、コンパイル時にプリプロセッサにより PEMAP 関数呼び出しと、最適化を回避するコードに展開され、実行時に、予測の対象ループが実行される直前に呼び出される。

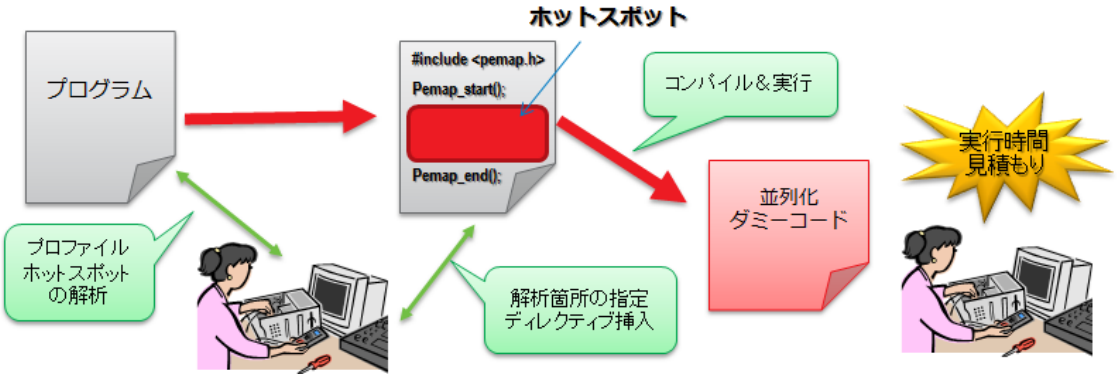


図 4.2-1　PEMAP のワークフローの全体概要

```
1   #include <pemap.h>
2   void func(int *in, int *out)
3   {
4       PEMAP_LOOP_START; // A PEMAP Annotation
5       for (int i = 0; i < N; ++i) {
6           out[i] = in[i] + 1;
7       }
8       PEMAP_LOOP_END; // A PEMAP Annotation
9   }
```

図 4.2-2　PEMAP を利用する時のアノテーションの例

呼び出された PEMAP 関数は、予測の対象ループを逆アセンブルした上で解析し、ループレベルコントロールフローグラフ（LCFG: Loop-level Control Flow Graph）を構築する。通常、コントロールフローグラフ（CFG）を用いる解析では、プログラムは基本ブロックを組み合わせた有向グラフで表現される(図 4.2-3)のに対し、LCFG を用いる解析では、プログラムは基本ブロックとシーケンスを組み合わせた無閉路有向グラフで表現される。そして、このシーケンスも同じ形式のグラフで表現される(図 4.2-4)。

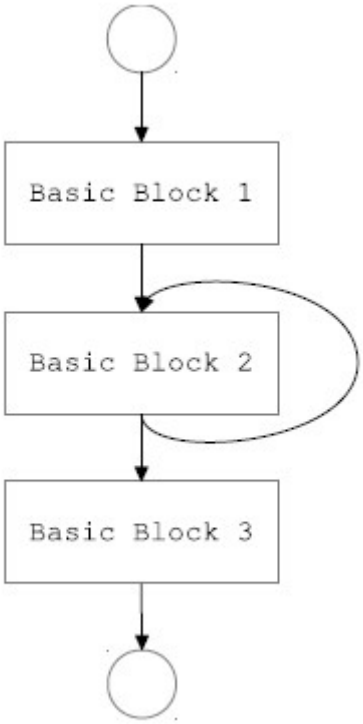


図 4.2-3 CFG の例

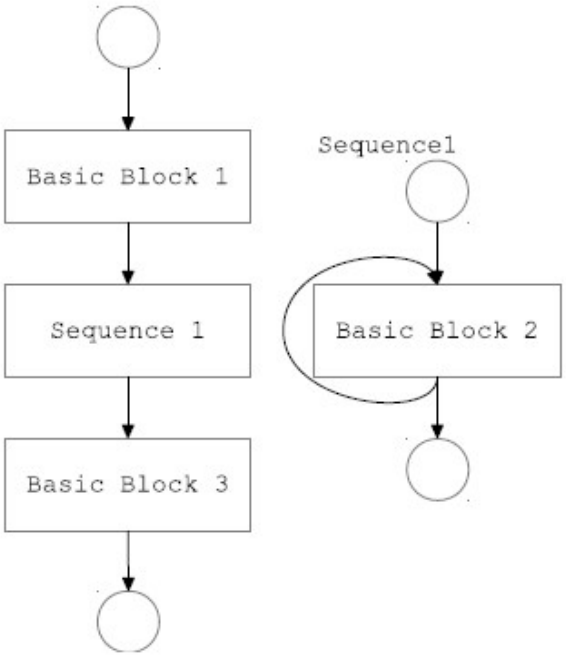


図 4.2-4 LCFG の例

次に LCFG の各シーケンスを解析し、読み出し、書き換えを行うメモリアドレスやレジスタを明確にする。なお、これらの解析手法については広く知られているため本稿では詳細には触れない。

解析後、PEMAP は、対象関数に含まれる機械語命令から、PEMAP が内部で用いる中間表現（Intermediate Representation、 IR）を生成する。さらに、IR を解析し、その変換や最適化を行う。IR や、その解析、最適化手法の詳細は後節にて述べる。

IR への変換後、対象関数を、動的情報収集の機能を埋め込みながら別領域にコピーする。収集する情報は、条件付き分岐命令による分岐が行われたかどうかと、対象関数を実行する前にアドレスを確定できないメモリ読み出し・書き込みアクセス（以降、動的メモリアクセスと呼ぶ）において実際にアクセスしたアドレスと、対象関数に含まれるループが回った回数である。

条件付き分岐命令の結果とループ回数については、対象関数が持つ基本ブロックの先頭に情報収集用関数呼び出しを埋め込み、その実行順序を解析することで取得することができる。また、メモリアクセス時のアドレスについては、メモリアクセスを行う命令の直前に収集用関数呼び出しを埋め込むことで取得することができる。最後に、PEMAP は、IR からダミー並列プログラムを生成して実行することで、プログラムの性能予測を取得する。

PEMAP IR

前節で述べた通り、PEMAP は機械語で書かれた対象関数を一度静的解析に向いた IR に変

換した上で、CUDA C で書かれたダミー並列プログラムを出力する。また、IR の変換と最適化を行うことで、人の手で並列化したプログラムの性能に近づけている。本節では、PEMAP が内部で用いる IR と、その変換と最適化について示す。また、IR からダミー並列プログラムへの変換についても示す。

PEMAP IR の要素

本節では、PEMAP IR を構成する要素を解説する。最適化前の IR は、ある値を別の値に変換する IR 命令、対象関数が実行される前から定まっている値を表す IR 定数、値をメモリからロードする IR ロード、値をメモリに書き込む IR ストア、前のループ実行回に算出した値である IR ループキャリー、2 つ以上の値から 1 つを選択する IR ファイ関数と、IR ループキャリーが所属する IR ループから構成される。

また、IR の変換、最適化により、複数の値の加減算を表す IR ComAssoc、ループが反復される毎に一定数だけ増減する IR インダクション、複数の値から算出された 1 つの値を表す IR リダクションが生成される。

IR 命令は、3 番地コードの形式で表現されたもので、オペランドとして持つ 1 つか 2 つの値から算出される 1 つの値である。IR 命令は、対象関数の処理内容を再現する役割を持つ。

IR 定数は、対象関数内では変更されない値である。対象関数の外で変更される値であっても定数と呼ぶ点に注意が必要である。IR 定数は、自身がどこに由来するものなのか、つまり、即値として与えられたのかレジスタから渡されたのか等の情報を持つ。IR 定数は、IR 命令のオペランドや、メモリアクセス時のアドレスとして定数が使用されたことを明確にする役割を持つ。

IR ロードは、メモリから読み出した値であり、読み出し元のアドレス値を持っている。IR ロードは、メモリ読み出しのコストをダミー並列プログラム上で再現する役割を持つ。

IR ストアは、メモリに書き込むことを表しており、書き込み先のアドレス値と、書き込む値を持っている。IR ストアは、メモリ書き込みのコストを再現する役割を持つ。また、書き込む値を保持することで、ダミー並列プログラム上で計算する必要のある値を明確にする役割を持つ。

IR ファイ関数は、静的単一代入（Static Single Assignment、 SSA）形式において用いられる概念に由来するもので、複数の値から選択された 1 つの値を表す。IR ファイ関数は、選択される複数の値と選択する基準となる IR 命令を持っている。

条件分岐結果によって変わる必要のある値が変わるので、IR ファイ関数はこれを明確にする役割を持つ。

IR キャリーは、ループのある反復回で使用される、前の反復回で計算された値を表しており、初期値と、前の反復回での計算内容を示すバックエッジ値と、所属する IR シーケンスを持っている。

IR キャリーは、ループにおけるデータの流れを IR 上で表現する役割を持つ。IR シーケンスは、 LCFG を IR 上で表現するもので、内部に含んでいる IR シーケンスを持つ。IR シーケンスは、IR ストアや IR キャリーに関連した計算を行っている箇所を記録するために必要となる。IR ComAssoc（交換法則と結合法則を満たす値: Commutative and Associative value）は、複数の値の加減算を表しており、加減算の対象となる値を持っている。IR ComAssoc の役割は IR 命令と同じだが、IR 命令よりも容易に解析を行うことができる。IR インダクションは、ループが反復される毎に一定の値だけ増減する値を表しており、増分となる値を持つ。IR インダクションはメモリアクセスのアドレスとして頻繁に現れ、また CUDA C のスレッドインデックス等を用いて容易に表現できるため高速に算出することができるため、ダミー並列プログラムの性能を向上させ、人の手で並列化したプログラムの性能に近づける役割を果たす。IR リダクションは、複数の値から算出された 1 つの値を表しており、初期値と、算出する元となる値を持つ。IR リダクションに対応する処理の例として、総和や総乗の計算や、最大値や最小値の探索等が挙げられる。IR リダクションの処理は、シーケンシャルプログラムとは異なるアルゴリズムを用いることで、性能を向上させることができる。

この例を示すため、図 4.2-5 のような配列に格納された値の総和計算を取り上げる。シーケンシャルプログラムのループを用いる場合、結果を格納するメモリ領域を確保して、その値を初期化して、順次計算対象となる配列の内容を加算する。これと同等の計算を並列に行う場合、結果を格納するメモリを読みだし、加算を行い、結果を書きだす処理を排他的に行う必要があるため、複数のスレッドがメモリの 1 箇所へアクセスすることによるボトルネックや、排他処理にかかるコストが大きいというパフォーマンス上の問題が生じる。

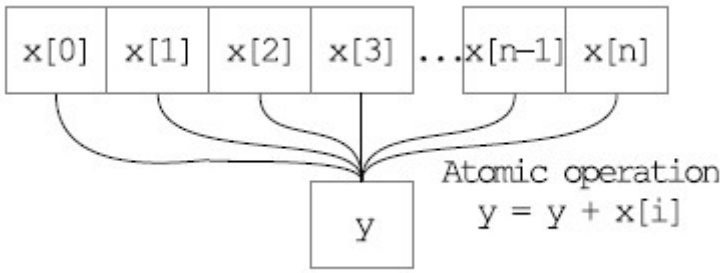


図 4.2-5　単純な総和計算

これに対して、リダクションの処理では図 4.2-6 のように配列の隣り合う値を加算して、長さが半分の新たな配列を作り出す操作を、配列の要素数が 1 になるまで繰り返すことで、並列計算機において、総和を高速に計算することができる。

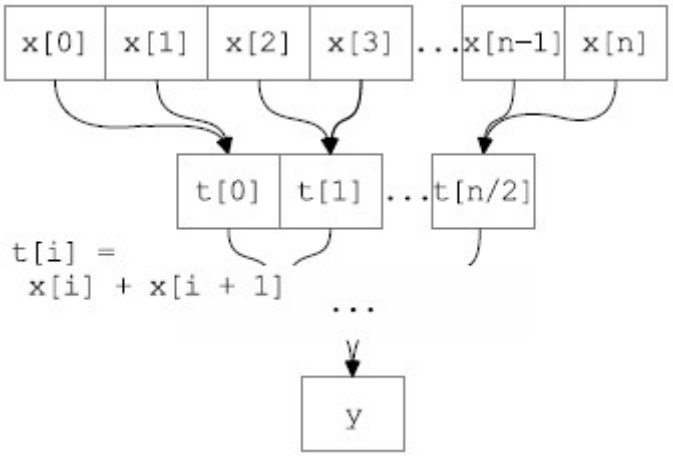


図 4.2-6　リダクションによる総和計算

PEMAP IR の生成

本節では、PEMAP IR を生成する手法について、その詳細を示す。また、生成した IR を変換、最適化して、ダミー並列プログラムの生成に用いることができるようにする手法についても示す。

PEMAP は、対象関数中の各機械語命令の依存関係、つまりある機械語命令が、オペランドとしてどの命令の結果を取っているかを解析した上で、IR 命令に置き換えることで IR を生成する（図 4.2-7）。

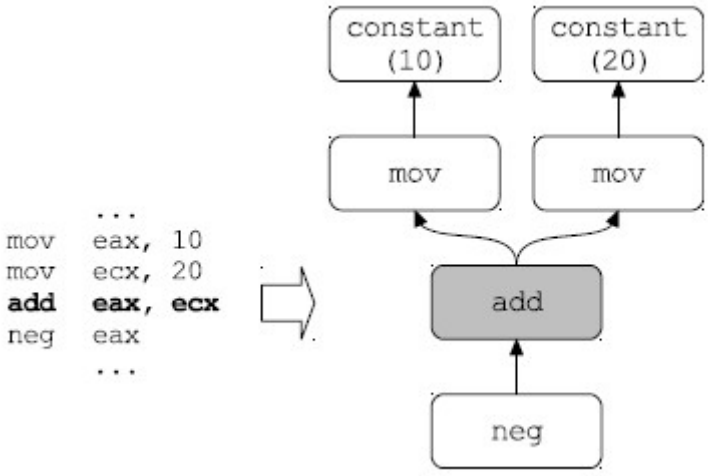


図 4.2-7 IR 命令の生成

対象関数が持つ機械語命令が、他の命令が上書きしていないレジスタの値を参照する場

合、その値から IR の定数を生成する。また、命令が即値をオペランドとして持つ場合も定数を生成する。同様に、他の命令が上書きしていないメモリ上の値を参照する場合、そのアドレスから IR ロードを生成する。これとは逆に、ある機械語命令がメモリに値を書き込んで、その後に上書きされることがなければ、そのアドレスと書き込みを行った機械語命令から IR ストアを生成する。

ループ内で領域に格納した値を次のループ反復回で使用するケースが存在した場合、IR キャリーを生成する。

IR キャリーの初期値には、そのループが実行される前に該当する領域に値を格納した機械語命令に対応する IR 命令や、該当する領域を示す IR 定数等を割り当てる。バックエッジは、ループ内の命令のうち、その領域に最後に値を格納した命令とする。(図 4.2-8)

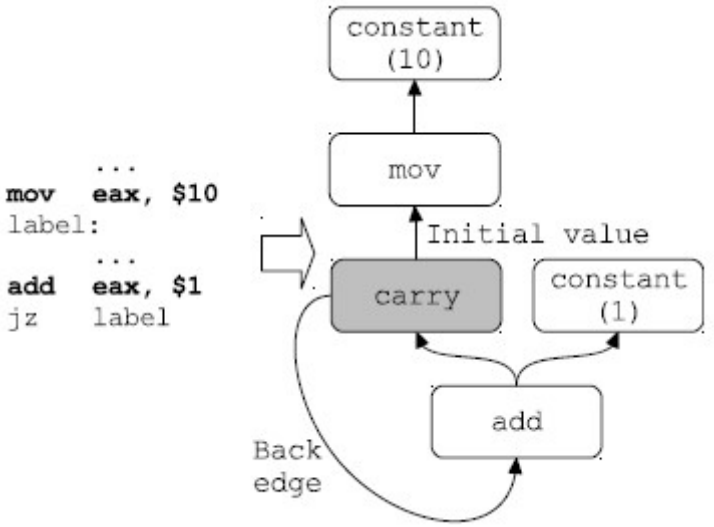


図 4.2-8 IR キャリーの生成

条件付き分岐を含むプログラムでは、ある命令のオペランドが複数の命令の結果のいずれかを取ることがある。このような場合、LCFG から支配木を生成して解析することで、オペランドを複数の値から 1 つを生成するファイ関数を生成する。同時に、ファイ関数に、値を選択する条件として、機械語の条件付き分岐命令が参照するフラグを生成した命令に対応する IR 命令を付与する。

IR の変換・最適化で行う処理は、表 4. 2-1 の通りである。本稿では、これらの手法のうち、性能予測で重要となるインダクション検出とリダクション検出について詳細に述べる。

表 4. 2-1 変換・最適化処理一覧

変換・最適化処理	内容
冗長命令除去	mov 等の変換を行わない命令を除去する
リアソシエート	複数の加減算命令を 1 つの comassoc にまとめる
キャリー展開	展開可能な IR キャリーを展開する
絶対アドレスロード検出	IR メモリロードのうち常に一定のアドレスを持つものを定数扱いとする
ループ不変値検出	値として扱われる IR の要素ついて，どのループから見ると不変なのかを明確にする
インダクション検出	本文にて詳述
配列検出	IR メモリロード，IR メモリストアのうち，同一の配列にアクセスしているものと見なせるものを検出する
共通部分式除去	IR の要素のうち，同一のものとして扱うことのできるものを 1 つにまとめる
リダクション検出	本文にて詳述

インダクション値は、ループの反復回が決まればその値も確定させることができるので、CUDA C におけるスレッドインデックス等を組み合わせて容易に表現できる。

一般的に、インダクション値は配列アクセスのインデックスとして用いられることが多いので、ダミー並列プログラム上でメモリアクセスパターンを再現するために重要である。

前節で述べた通り、キャリー値は、その初期値とバックエッジから構成される。

インダクション検出では、キャリー値のバックエッジが、そのキャリー値と、キャリーが所属するループを実行する前に確定させることのできる値との加減算のみで表現可能である場合に、そのキャリー値をインダクション値と初期値との和に置き換える。

前述の通り、リダクションとは、複数の値から 1 つの値を導出する操作を指す。

そして、並列計算機ではリダクション処理を工夫することで計算時間を短縮することが可能であることが、リダクション検出を行う理由である。

PEMAP の IR 生成部は、IR キャリーのバックエッジをたどり、総和や総乗といったリダクション処理のパターンと合致するかどうかを確認することで、リダクションの検出を行っている。例えば、総和の検出では、IR キャリーのバックエッジが、その IR キャリーと何らかの値の加算命令となっていれば、IR キャリーを IR リダクションに置き換える(図 4. 2-9)。

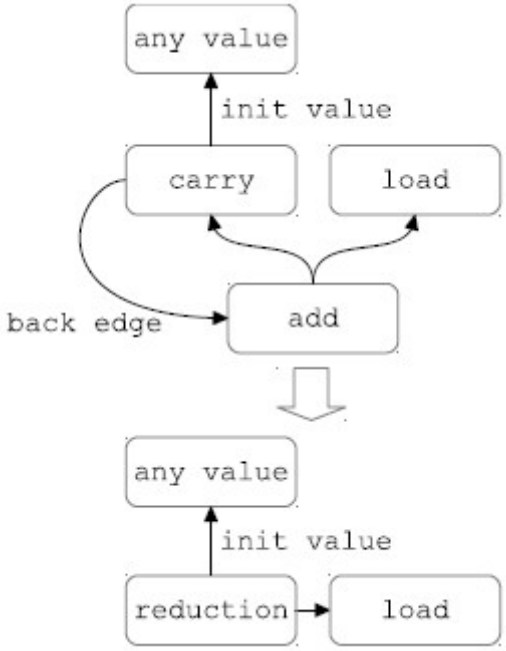


図 4. 2-9 IR リダクションへの置換

最大値や最小値の導出処理の検出では、IR キャリーのバックエッジが、その IR キャリーと何らかの値から選択するファイ関数になっており、選択条件がこれら二つの値の大小比較である場合に、IR キャリーを IR リダクションに置き換える。

PEMAP IR からダミー並列プログラムの生成

本節では、予測対象のシンプルなプログラム（図 4.2-10）から得た IR（図 4.2-11）をダミー並列プログラム（図 4.2-12）に変換する手順を具体的に示すことで、PEMAP IR からダミー並列プログラムを生成する手順を例示し、IR の各要素の役割を明確化する。

```
1   static int in[1024], out[1024];
2   int main()
3   {
4     int i;
5     PEMAP_LOOP_START;
6     for (i = 0; i < 1024; ++i) {
7       out[i] = in[i] + 1;
8     }
9     PEMAP_LOOP_END;
10  }
```

図 4.2-10 予測対象プログラム

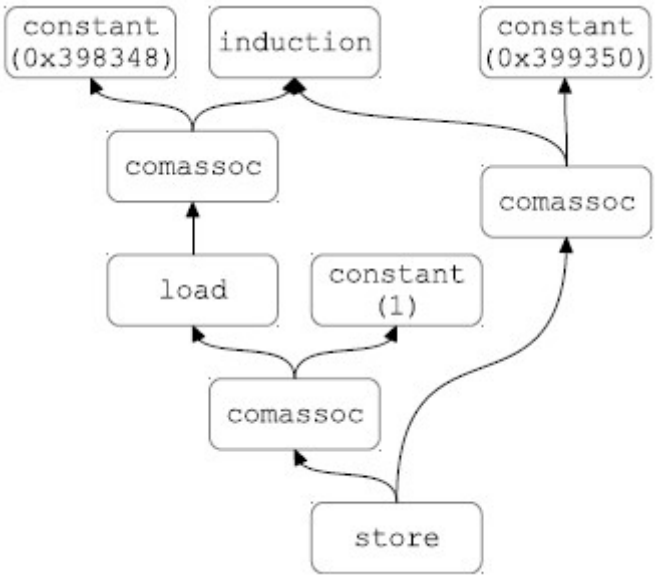


図 4.2-11 予測対象プログラム(IR)

```
1   #define LOOP_TIMES 1024
2   #define SEQ_COUNT_0 1024
3
4   __global__ void dummy_code(
5     unsigned int *array_0,
6     unsigned int *array_1
7   ) {
8     unsigned int t = (offset + blockIdx.x) * blockDim.x + threadIdx.x;
9     if (t < LOOP_TIMES) {
10      unsigned int t0 = t % SEQ_COUNT_0;
11      unsigned int v0 = 4;
12      signed int v1 = t0 * v0;
13      unsigned int v2 = 0;
14      signed int v3 = +v1+v2+0;
15      unsigned int v4 = 0;
16      signed int v5 = +v1+v4+0;
17      unsigned int v6 = array_0[v5 / 4];
18      unsigned int v7 = 1;
19      signed int v8 = +v6+v7+0;
20      array_1[v3 / 4] = v8;
21    }
22  }
```

図 4.2-12 ダミー並列プログラム

ダミー並列プログラムの生成は、最適化した IR に含まれる IR ストアをダミー並列プログラム中の CUDA カーネル（以降、ダミーカーネルと呼ぶ）として出力する。その過程で、

IR をグラフと見なし、IR ストアが持っているアドレスと値を深さ優先探索しながら出力することで、プログラム全体の計算を再現する。

まず、対象関数を実際に動作させて取得したループ回数を出力する。例では、1 行目の LOOP_TIMES マクロがループ回数である。次に、カーネル関数内に、ループ回数分だけ実行されるブロックを生成する。例では、6 行目で変数 t にインデックス値を格納し、7 行目で、実際のループ回数よりも小さいインデックスを持つ場合のみ実行される if 文を提供している。また、このブロック内で、各ループのインデックスを再現する。例では、ループは 1 つしか存在しないため、変数 t0 だけを生成している。

ブロックの出力後、IR 内の全ての IR ストアを、配列への書き込みとして再現する。同時に、IR ストアが持っているアドレス値と書き込み値を同じブロック内に出力する。

例では、20 行目でメモリ書き込みが生成されており、書き込み値として 19 行目で定義された IR ComAssoc が使用されている。さらに、PEMAP はこの IR ComAssoc をたどり、定数（18 行目）とメモリロード（17 行目）を生成し、メモリロードのアドレスも同様に生成（16 行目）している。この時、IR では大きな定数（0x398348）が存在しているが、これは配列が置かれているアドレスであると考えられるので、0 として扱う（15 行目）。

また、アドレスをたどると IR インダクションが現れる。IR インダクションは、インデックス値に定数を乗じたものとみなすこともできるので、12 行目のように出力されている。なお、乗数である整数 4 は、整数のサイズ 4 バイトを表している。さらに、同様の手順で IR ストアのアドレスを生成することで（13、14 行目）、ダミー並列プログラムが完成する。

4.2.3 評価

本節では、自動並列コンパイラの性能測定のためのベンチマークスイートである BEMAP から、Gray Scale ベンチマークを選択して、PEMAP の評価を行った。Gray Scale ベンチマークは、入力された RGB 画像全体のグレースケールの画像を生成するベンチマークであるが、PEMAP のメモリアクセスパターンの評価のため、入力された RGB 画像の一部のみを処理対象とするよう修正を加えた。また、BEMAP のベンチマークは OpenCL で記述されているため、これを CUDA に修正した。Gray Scale ベンチマークに横幅 1152 ピクセル、縦幅 864 ピクセルの画像を入力した際に、PEMAP が生成したダミー並列プログラムのカーネルを図 4.2-13 に示す。

```
1   #define STACK_ARG_0 1152
2   #define STACK_ARG_1 864
3   #define ABS_LOAD_0 0.114478
4   #define ABS_LOAD_1 0.298912
5   #define ABS_LOAD_2 0.586611
6   #define SEQ_COUNT_0 1152
7   #define SEQ_COUNT_1 864
8
9   __global__ void dummy_code(
10    unsigned char *array_0,
11    unsigned char *array_1,
12    unsigned char *array_2,
13    unsigned char *array_3
14  ) {
15    volatile __shared__ int dummy;
16    unsigned int t = (offset + blockIdx.x) * blockDim.x + threadIdx.x;
17    if (t < LOOP_TIMES) {
18      unsigned int t0 = t % SEQ_COUNT_0;
19      unsigned int t1 = (t / SEQ_COUNT_0) % SEQ_COUNT_1;
20      signed int v1 = t0 * 1;
21      signed int v2 = t1 * STACK_ARG_0;
22      signed int v3 = v1 + v2;
23      unsigned char v4 = array_0[v3];
24      signed int v5 = (signed int)v4;
25      float v6 = v5 * ABS_LOAD_0;
26      unsigned char v7 = array_1[v3];
27      signed int v8 = (signed int)v7;
28      float v9 = (float)v8 * ABS_LOAD_1;
29      float v10 = v6 + v9;
30      unsigned char v11 = array_2[v3];
31      signed int v12 = (signed int)v11;
32      float v13 = (float)v12 * ABS_LOAD_2;
33      float v14 = v10 + v13;
34      unsigned int v15 = (unsigned int)v14;
35      array_3[v3] = v15;
36    }
37  }
```

図 4.2-13 GPU のダミーコードのカーネル例

ただし、自動生成されたコードは可読性が低いため、プログラムの意味を壊さない範囲で修正を加えている。

この結果をみると、配列からのロードが、23 行目、26 行目、30 行目にて行われ、配列への書き込みが 35 行目にて行われている。そして、これらのインデックスは全て 20～22 行目にて算出されている。そして、SEQ_COUNT_0 マクロに画像の横幅が格納されていることから、18、19 行目で算出されている t0、t1 は、シーケンシャルプログラムのインデックスに相当するものであることがわかる。さらに、STACK_ARG_0 にも画像の横幅が格納されている

ことを考えると、v1 は画像の横位置を指定するインデックス、v2 は縦位置を指定するインデックスであることが判明する。

これらを踏まえると、このダミー並列プログラムは、画像のサイズだけ 3 つの配列に格納された値を順次読み出し、計算処理を行い、1 つの配列に書き出す動作をすることがわかる。

さらに、Gray Scale ベンチマークは常に入力された画像全体を変換する。これに修正を加え、画像の左上 1/4 だけを対象とするよう修正を加えてダミー並列プログラムを生成した結果を、図 4. 2-14 に示す。なお、カーネル部分については図 4. 2-13 と同じである。この結果をみると、SEQ_COUNT_から始まるマクロの値だけ、修正前の半分になっていることが分かる。これらのマクロは、シーケンシャルプログラムのインデックスに相当する値の範囲を決めていることを踏まえると、修正後のダミー並列プログラムでも画像の左上 1/4 相当の領域へのアクセスを再現できていることがわかる。

```
1  #define STACK_ARG_0 1152
2  #define STACK_ARG_1 864
3  #define ABS_LOAD_0 0.114478
4  #define ABS_LOAD_1 0.298912
5  #define ABS_LOAD_2 0.586611
6  #define SEQ_COUNT_0 576
7  #define SEQ_COUNT_1 432
```

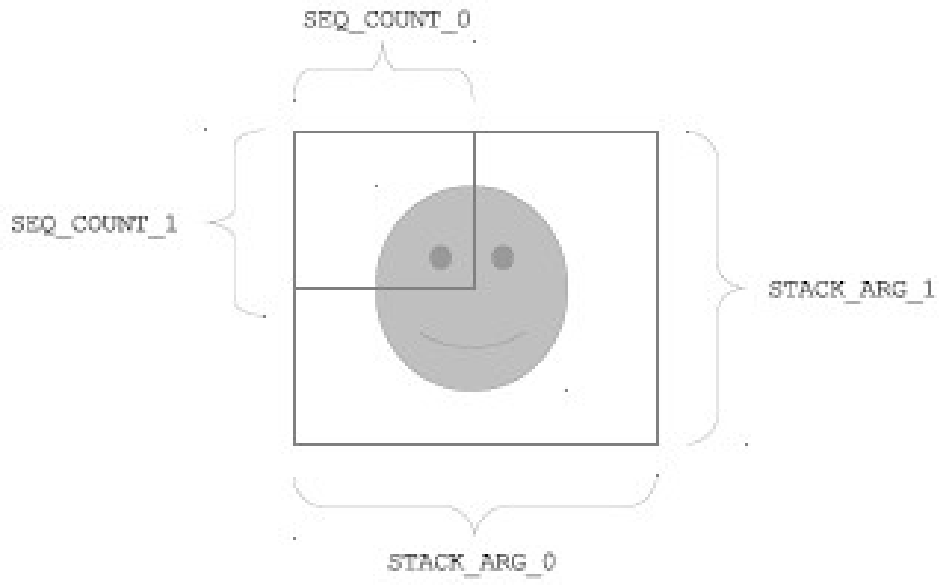


図 4.2-14：ダミー並列プログラムと PEMAP メモリアクセス範囲

4.3　メニーコア向けベンチマークセット BEMAP

4.3.1 背景

BEMAP はメニーコア時代の「最適化カルチャー」に対する支援ツールである。メニーコア化は、いわば各コアのシンプル化を強いることとなりこれまでソフトウェアが恩恵をうけていたマイクロアーキテクチャが極力減らされる方向となる。ハードウェアの挙動やリソースを熟知してプログラムを開発しなければ性能がでないケースが多くなる。

BEMAP では、これまでフィックスターズが蓄積してきた最適化技術のノウハウを共有、さらにはメニーコアソフトウェアの開発経験が少ないユーザを対象にライブラリとしてもすぐに活用できるように、高速処理が要求される分野の代表的な処理をオープンソースで公開した。

4.3.2 BEMAP: BEnchMark for Auto-Parallelizer

BEMAP は OpenCL で記述された高速なソフトウェアライブラリ群と同時に前節で紹介した CLtrump といったコードの自動並列コンパイラのベンチマークという役割も担っている。2013 年 4 月現在 BEMAP では次に示す 8 種類の処理コードが含まれている。各処理コードには、シングルスレッドで動作するリファレンスコードと手動で並列化したコードがセットとなっている。手動で並列化したコードは最適化手法を指定することができるため自動並列コンパイラの性能を最適化手法ごとに手動の最適化と比較評価することが可能となる。

- ガウシアンフィルタ
- ブラック・ショールズ
- リニアサーチ
- グレースケール化
- SIFT 処理
- ランレングス法
- バックプロジェクション
- モンテカルロ法

本報告書では一例として BEMAP に含まれるバックプロジェクションの説明をする。また、BEMAP で公開している、すべての処理に対する最適化の技術的な説明や評価結果はレポートとして Web で公開している[XXX]。

バックプロジェクション

Back Projection（逆射影）とは射影によって次元数を落とされた情報を元の次元数の情報に引き戻すことである。BEMAP に含まれるバックプロジェクションのコードは ACM Programming Contest、 South Pacific Regionals 2000 において出題された”Discrete Digital Tomography”を対象としている。このプログラムでは中身の見えない m 行 n 列に区切られたラックマウント型サーバについて考えている。ラックサーバの区切られた各部屋は空であるかウィジェットが搭載されているかのどちらかの状態となっている。ウィジェットはβ線を吸収するため、ラックに向かってβ線を照射し反対側でそのβ線の強さを観測することで通過したライン上のいくつの部屋にウィジェットが含まれていたかを判断することができる。本プログラムでは図 4.3-1 に示す向きのベータ線を照射し、ライン上に存在するウィジェットの数が与えられる。

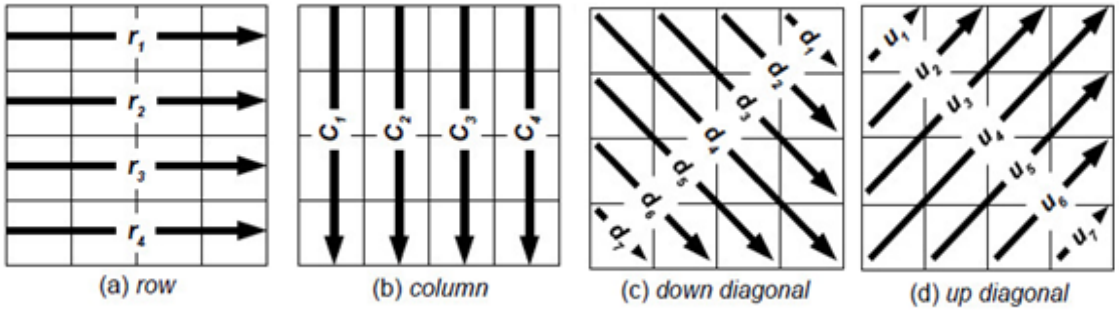


図 4.3-1 β線の向き

このプログラムでは射影前の情報はラック内の各部屋のウィジェットの有無であり、射影により得られている情報は各ライン上のウィジェット数である。そこでまず射影により得られた 4 つの 1 次元空間の情報をそれぞれ 2 次元空間上に引き戻していく。本実装では射影により得られるライン上のウィジェット数 Proj とライン上の部屋の数 Band から、ラ

$$P_l = \frac{Proj_l}{Band_l} \qquad 4.3\text{-}1$$

イン上の各部屋のウィジェットが存在する確率 P を求める。あるライン 1 上の部屋の確率 Pl の値は以下の式で求められる。

次に得られた 4 つの 2 次元情報の重ね合わせを行っていく。重ね合わせのため、各部屋について属する 4 本のラインから得られる確率の積を求め、これを評価 E する。 i 行目 j 列目の部屋はライン ri、 cj、 ui+j、 di-j+n-1 に属するので得られる評価 Ei,j は以下のようになる。

評価 E を部屋にウィジェットが存在する可能性の指標とする。各部屋の評価 E の大きさに基づいて格子中に物体が存在するかを判断していく。

$$E_{i,j} = P_{r_i} \times P_{c_j} \times P_{u_{i+j}} \times P_{d_{i-j+m-1}} \qquad 4.3\text{-}2$$

実際のプログラムは以下の手順からなる。

1. 射影の値に応じて各格子の評価 E を求める
2. 評価 E が最も大きい格子を探す
3. 評価 E の最も大きい格子を推測解に加える
4. 推測解の格子を除外した Proj また Band の値を用いて 1 から 3 の操作を繰り返す
5. 全ての格子における評価が 0 になった時点での推測解を回答とする

バックプロジェクションの実装

プログラム中で解放は 3 つの関数で構成されそれぞれ概ね以下のような役割をもって実装されている。

- backprojection: 各格子の評価 E を導出する
- findmax: 最も評価 E の値が大きい格子を探索する
- decreaseproj: findmax で得られた格子を推測解に加え Proj および Band の値を更新する

これら関数の処理について backprojection 関数において評価 E の導出は格子間に依存関係がないため並列化に適しており、また findmax 関数は一般にリダクションと呼ばれる形で並列性を抽出しやすい。そのため OpenCL での記述は主にこの 2 つの関数の処理の並列化に焦点を当てている。

・並列性の抽出

まず backprojection に関して以下のような方法を試している。

(1) 格子ごとに 1 ワークアイテムを割り当てる
(2) ワークグループごとに正方形型の空間を割り当てる、1 ワークアイテムあたりの計算量はワークグループサイズの数の格子

図 4.3-2 はワークグループあたりのワーク相手無数が 4 のときの様子である。黒色部は 1 ワークアイテムの演算領域、灰色部は 1 ワークグループの演算領域を示す。

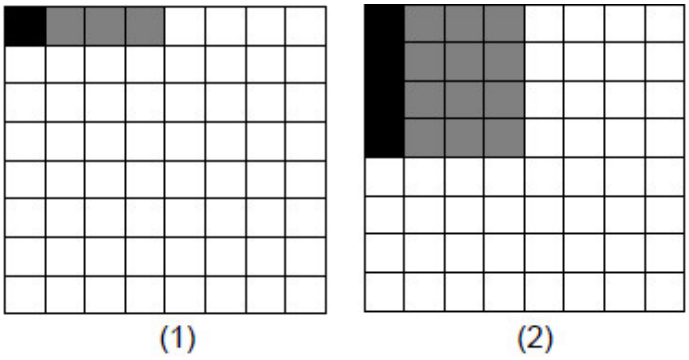


図 4.3-2 ワークアイテムの演算領域

(2)の方法の利点としてはワークグループ内でローカルメモリを介してProj とBandの値を共有することで、グローバルメモリへのアクセス数を削減できる点にある。ワークグループのサイズがnの場合について考える。(1)の手法ではワークアイテムあたり必要なグローバルメモリへの読み込みはProjrにつき1回、Projcにつきn回、Projuにつきn回、Projdにつきn回、またBandへのアクセスも含めるさらに倍のアクセス数となる。対して(2)の手法ではProjrにつきn回、Projcにつきn回、Projuにつき2n-1回、Projdにつき2n-1回となる。双方の1格子あたりのアクセスについて比較すると(1)では2(3n+1)/n、(2)では2(6n-2)/(n*n)でnが十分大きい場合では2/n倍にアクセス数が削減される。

また正方形の空間を割り当てるのはNVIDIA GPUにおけるcoalesced accessについて無駄が生じないためである。このような設計ならばnの値が32(NVIDIA GPU内部において単一 SIMD 命令で制御されるスレッド数)の倍数であればメモリアクセスの資源を十分に活用できる。

ただし(2)の方法を用いることで生成されるワークアイテム数は減るため、問題サイズが小さいような場合では計算資源が十分に使いきれないということが考えられる。

結果として 1024x1024 程度の格子数では(2)の方法は(1)を上回る性能を持っていたため(2)の方法を採用している。

findmax 関数については図 4.3-3 に示すような形で並列実行を行う。各ワークアイテムが 2 要素の比較を行いローカルメモリに値の大きいものを残すといった形になっている。ローカルメモリはワークグループ内でのみ共有されるため、結果はワークグループの数だけ配列内に残される。この処理を最後の 1 要素まで続けることで最大値を特定できる。図 4.3-3 を見てわかるとおりワークグループ内で処理が終わりに近づくにつれ並列性は損なわれ、実際に働くワークアイテムの数が減ることがわかる。性能を発揮させるためにはワークグループあたりに割り当てるタスクの量をある程度大きくすることで資源が使い切る必要がある。

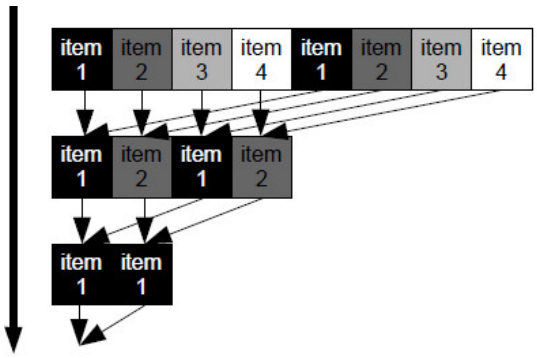


図 4.3-3 *findmax* における並列性の抽出

GPU での実行では多くの計算資源を生かすため以上のような方法でワークアイテムの数を増やしてやることが必要だが、マルチコア CPU ではワークアイテムのスイッチングのオーバーヘッドを考慮するとワークアイテムあたりのタスク量を大きく設定する必要がある。代わりにベクトル型を用いることで SIMD 命令を使いワークアイテム内で並列な処理を行わせることができる。そこで SIMD 命令を利用する backprojection と 1 度めの findmax カーネルは図 4.3-4 のように各ワークアイテムが 1 行分の格子の計算を行う。

Workitem 0
Workitem 1
Workitem 2
Workitem 3
Workitem 4
Workitem 5
Workitem 6
Workitem 7

図 4.3 - 4　SIMD カーネルでのタスク分配

・データ転送回数の削減

ホストーデバイス間のデータ転送はホスト内またはデバイス内のメモリアクセスと比較して大きなレイテンシを伴う。入力データを転送した後の処理を全てデバイスで実行し、イテレーションごとに追加される推測解の座標またはループを脱出する場合は-1 の値のみをホストに送り返す。ホスト側では送り返された推測解を記録することでループを脱出した際に推測解をまとめてホストに送り返すコストも削減している。

・カーネルの呼び出し回数の削減

カーネルの呼び出しにはレイテンシが伴う。問題サイズによってこのレイテンシが全体の実行時間に影響を及ぼす恐れがある。またカーネルを終了する前には計算によって得られた結果をグローバルメモリに書き戻さなくてはならない。複数のカーネルの処理を 1 カーネルにまとめることにより、これらのレイテンシとメモリアクセスを削減することができる。そこで backprojection の処理に findmax の処理を結合し backprojection で計算した領域の中で評価 E の値が最大な値をとる格子についてのみワークグループごとにグローバルメモリに書き戻しを行っている。

・計算量の削減

以下のような方法で計算量の削減を図っている。

1. Pl は同一ライン上の格子の評価 E の導出に複数回利用されるため、Projl/Bandl の計算を各ワークアイテムやイテレーション内でなく、あらかじめ一括して行う

実際にプログラムでは Pl の値は CPU 側で計算し、推測解が加わるたびに推測解を含むラインの Pl の値のみを decraseproj 関数内で更新している。

2. 評価 E の導出において Pr、 Pc、 Pu、 Pd のいずれかが 0 の値をとるとき必ず 0 になる事を利用しメモリアクセス数を減らす

まず計算領域を正方形型に割り当てる Scalar 計算のカーネルではワープが 1 行ずつ計算を行っていくため、最初に Pr にアクセスすると Pr が 0 のとき 1 行分の計算が省ける。SIMD カーネルではワークアイテムに 1 行ずつ格子が割り当てられているため、Pr が 0 のときはそのワークアイテムの計算を全て省き、ワークグループ内の評価 E の最大値を 0 と返すことができる。

3. r、 c、 u、 d 方向それぞれイテレーションにつき 1 ラインの Pl のみ更新される事を利用し、評価 E の値は更新されたラインに含まれる格子のみ更新する

この計算量の削減は CPU 向けのカーネルにのみ実装されている。この方法では backprojection での計算量が O(n^2)のオーダーから O(n)のオーダーとなり大きく削減される。代わりに backprojection カーネルにおいて使用されるワークアイテムの数が findmax カーネルと異なる実装になるため backprojection と findmax の処理の結合ができていない。backprojection 終了時と findmax 開始時に非常に多くのメモリアクセスが発生する欠点が

ある。CPU 実装では結果として高速化されたが、GPU に関しては現時点で速度向上は得られておらず、またバグも存在している。

・メモリ転送の非同期化

問題サイズが小さいときイテレーションごとのデータ転送のレイテンシが全体の実行時間に影響を与える。そのため GPU 実行時のみ複数のコマンドキューを使い、イテレーションごとの推測解の座標の転送を非同期で次のイテレーションの backprojection 関数を並列に行っている。

バックプロジェクションの評価

以下の表は 1024x1024 格子の場合における、各条件化での性能を測定したものである。CPU での最適化前のコードに対して GPU で約 170 倍、CPU で約 83.36 倍の性能向上が得られた。kernel について for GPU は GPU 向けにタスクの分割を細かくしたもの、for CPU は CPU 向けにタスクの分割が粗くワークアイテムが少なくなるものである。また CPU での実行に関して vector を用いて記述したカーネルより scalar によって記述したカーネルのほうが高速での実行となった。

表 4.3 -1 バックプロジェクションの性能結果

Device	kernel	Work Group Size	Execution Time (ms)	Speedup
CPU	without OpenCL		12577.80	1.00
GPU	for GPU (scalar)	32	108.17	116.28
GPU	for GPU (scalar)	64	76.26	164.93
GPU	for GPU (scalar)	128	73.94	170.11
GPU	for GPU (scalar)	256	114.90	109.47
CPU	for GPU (scalar)	32	3889.86	3.23
CPU	for GPU (scalar)	64	3616.79	3.48
CPU	for GPU (scalar)	128	3459.72	3.64
CPU	for GPU (scalar)	256	4309.85	2.92
CPU	for CPU (scalar)		198.49	63.37
CPU	for CPU (vector)		150.89	83.36

4.3.3 評価

表 4.3-2 に BEMAP で公開しているコードの性能結果をまとめている。OpenCL のコードを測定したプロセッサは、現在市販されているプロセッサで OpenCL のコンパイラが存在しているものを選択している。Intel 社の汎用マルチコアプロセッサとして、6 コアの Nehalem と 4 コアの Ivy Bridge を選定し、GPU として NVIDIA 社の GTX570 と GTX680 で評価した。

表 4.3-2　BEMAP の性能表

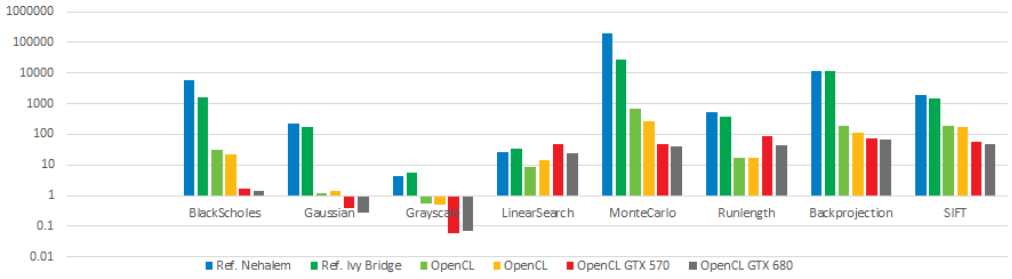
Workload	Ref. Nehalem	Ref. Ivy Bridge	OpenCL Nehalem	OpenCL Ivy Bridge	OpenCL GTX 570	OpenCL GTX 680
BlackScholes	6109.07	1587.80	32.05	21.41	1.70	1.40
Gaussian	227.42	179.30	1.24	1.40	0.39	0.27
Grayscale	4.32	5.47	0.53	0.52	0.06	0.07
LinearSearch	26.25	33.33	8.42	14.24	47.14	24.08
MonteCarlo	193817.40	27423.70	696.53	264.49	49.68	41.50
Runlength	518.28	370.50	17.07	16.77	90.29	43.36
Backprojection	11297.43	11872.04	195.03	116.08	75.83	69.70
SIFT	1860.00	1452.00	194.11	170.51	55.54	48.02

Nehalem
Intel i7-X990 @ 3.47GHz (Nehalem)
8GB memory, 6 cores, 12 threads (HT)

Ivy Bridge
Intel Core i7-3770K @ 3.50GHz (Ivy Bridge)
8GB memory, 4 cores, 8 threads (HT)

GTX 570:
NVIDIA GTX 570 @ 1.46GHz (Fermi GF-110)
480 CUDA cores, 1.2GB GDDR5 memory

GTX 680:
NVIDIA GTX 680 @ 1.06 GHz (Kepler GK-104)
1536 CUDA cores, 2GB GDDR5 memory



参考文献

[産総研2009] 産業技術総合研究所「成果報告書:平成20年度ＩＴ投資効率向上のための共通基盤開発プロジェクト:マルチコアプラットフォーム上でのプログラム開発環境整備」2009年3月31日.

[Khronos2012] Khronos Group: OpenCL - The open standard for par-allel programming of heterogeneous systems(online)、 入手先 http://www.khronos.org/ (2012.12.20).

[FIXSTARS2013] Fixstars Corporation: BEMAP: BEnch-marks for Automatic Parallelizer。入手先 https://sourceforge.net/projects/bemap/ (2013.4.1)

5　メニーコア市場調査と実用化に向けた取り組み

5.1　メニーコア市場調査：全分野

本調査事業の目的は、マルチ・メニーコアの有望な適用分野、機器市場を明らかにし、普及を妨げる要因を抽出し、今後の研究開発の方向性を示すことにより、マルチ・メニーコアの普及と日本の半導体産業の競争力強化に繋がる提言、特に新規プロジェクト提言をまとめることである。市場調査に際しては、シングルコアからマルチコアを経てメニーコアへとシフトしていく市場動向を把握するため、対象をメニーコアに限定せず、マルチ・メニーコアを対象とした。マルチ・メニーコア技術のニーズは幅広く、組込み機器全般に及ぶと考えられるが、有望な適用分野、機器市場として、市場規模、成長性、日本の強み分野の観点から、日本の半導体産業が力を入れるべき分野にフォーカスして調査した。
具体的には、調査会社による一次調査を活用した国内外の最新動向の把握、他の再委託先であるキャッツ(株)、イーソル(株)と共同での調査内容の具体化、調査結果の分析などを行った。一次調査では、フィクスターズ社による調査会社の選定に際して、海外に十分な調査網を持っていること、我々の分析や新規プロジェクト提言に必要なデータを提供できることなどを重視し、iSuppli Japan 社を推薦した。

更に、国内外ユーザ企業のヒアリングを行い、ユーザ企業のニーズ、特に潜在的なニーズを把握に努めた。また、マルチ・メニーコア技術は多くの企業・大学・研究機関が研究開発を進めており、技術革新の速い分野であるため、学会参加等を通じて最新の技術動向把握に努めた。具体的には 2011 年度には、International Symposium on Low Power Electronics and Design (ISLPED)、HOT Chips 23、Micro-44、High Performance and Embedded Architectures and Compilers (HiPEAC)、2012 年度には、HOT Chips 24、Embedded Systems Week (ESWEEK)、International Conference for High Performance Computing, Networking, Storage and Analysis 2012 (SC12)、HiPEAC に参加した。スーパーコンピュータから組込み機器まで、様々な分野で電力制約が性能の上限を決めるようになり、高性能化には電力効率の向上が必須である。この結果、高い電力効率を実現できるヘテロジニアスマルチ・メニーコアが、ソフトウェアの最適化が困難であるにもかかわらず、高性能化を実現する有力候補となっている。

一次調査の分析からは、高性能化・多機能化と電力低減要求がマルチコア化の原動力、車載関連はフォールトトレラントの必要性からもマルチコア化を推進、ハイエンドルータ、基地局、サーバーは 16 個以上のコアを要求、といったトレンドが見えている。また、様々な課題からマルチコア化は進展するものの、International Technology Roadmap for Semiconductors (ITRS) の予測より少ないコア数を想定している会社が多いことが明らかになった。ITRS はベンダ企業が提供可能なコア数を予測しており、調査結果は活用可能なコア数がこれを下回っていることを示している。また、海外企業が得意な基地局、サーバ

ーなどの分野では、2015 年前後を目途にマルチコア化に対する課題を解決する見通しで進んでおり、日系企業の 2-3 年先を歩んでいる。一方、Combined Exhibition of Advanced Technologies (CEATEC)、Embedded Technology (ET) におけるセミナでのアンケート調査では、5 年後に必要と思われるコア数は 8 コア以上が最も多く、一次調査以上にコア数が求められる傾向が見られた。また、普及促進要因としては、開発/動作環境の整備、API 開発、レガシーソフト有効活用、マルチコア技術者の不足解消が挙げられる。特に、コア数増を意識せずに対応できるスケーラビリティがあり、デバッグ・並列化支援も考慮したソフトウェア環境が求められている。

以上の調査結果から、海外企業の後追いよりは組込み系、中でも日本の強み分野である制御系や制御系と情報系の融合分野においてビーイクルを選定し、開発/動作環境や API、レガシーソフト有効活用といった課題を解決するようなプロジェクトを起こすと共に、マルチ･メニーコア技術者の裾野を広げるような、手ごろなマルチ･メニーコアツールキットの開発・頒布が、マルチ・メニーコアの普及と日本の半導体産業の競争力強化に繋がるといえる。また、ヘテロジニアスマルチ・メニーコアが高性能化を実現する有力候補となっていることから、コア数やコア種、メモリアーキテクチャなどのハードウェアの多様性を許容して、これをソフトウェアが統合的に扱えるようにする必要がある。

5.2 メニーコア市場調査：画像処理分野

トプスシステムズ社は、画像処理分野、特に超高性能のコンピューティング・プラットフォームとして、優れたメニーコア・アーキテクチャが必要とされている画像認識系・超高精度描画等の次世代画像処理である「ビジュアル・コンピューティング分野（ビデオマイニングを含む)」について、特に積極的に新たな「市場を創る」ことを目指して、展示会への出展、技術講演、アンケート調査、ユーザ候補企業訪問によるヒアリング等により市場調査を実施した。

その結果、『画像認識系のシステムには、10 年後も使用可能なコンピューティング・プラットフォームが求められている』ということが、明確になった。10 年後も使用可能なコンピューティング・プラットフォームの主な要件を以下に示す。

① 超高性能：1TOPS ~ 1POPS (Peta Operations Per Second)
② スケーラビリティ：コア数やクラスタ数の増加に伴う演算処理能力の向上
③ プログラミング・モデル：スケーラビリティを活かせる並列処理モデルの確立
④ ソフトウェア開発環境；並列化支援ツールの拡充
⑤ 低消費電力：～1.5W 程度（組込みシステムの場合）
⑥ 事業継続性：大手半導体ベンダーによるサポート・製造

これらの要件は、トプスシステムズ社の SMYLEvideo アーキテクチャの研究開発に反映してきた。今後は、特に事業面でユーザから期待されている事業継続性の向上に向けて大手半導体ベンダーとの協業関係の構築に力を入れて、実用化・事業化を進める。

展示会への出展

本プロジェクトの研究開発期間中に国内外の 9 つの展示会に SMYLEvideo を出展し、ビジュアル・コンピューティング分野（ビデオマイニングを含む）のコンピューティング・プラットフォームとしての SMYLEvideo について、①高性能、②低消費電力、③プログラマビリティの高さ、④将来に向けて更なる高性能化が可能、などといった点で高い評価を得た。また、出来上がった際に、評価システムを提供して欲しいという顧客が少なくなかった。

技術講演

本プロジェクトの研究開発期間中に国内外の 4 つの学会等にて技術講演を行った。特に、スマート化が進む次世代組込みシステム（スマートフォン、スマートカー、スマートテレビ、スマートハウスなど）への応用に対する期待が高く、数多くのフィードバックが得られた。更に、米国の大学等から次世代の画像認識向けコンピューティング・プラットフォ

ームの研究開発への参加要請などもあった。

アンケート調査

国内外で出展した展示会や技術講演で合計 200 名以上の方々にアンケート調査にご協力いただいた（表 5.2-1）。特に、SMYLEvideo の展示・説明に対するアンケートの結果から、概ね賛同を得られていることが分かった。具体的な評価結果は以下の通り。

① ターゲットアプリの選定（80%以上の方が賛同）
② システム開発の方向性（80%以上の方が賛同）
③ 並列ソフトウェア開発の方向性（70%以上の方が賛同）
④ メニーコア・アーキテクチャの方向性（80%以上の方が賛同）
⑤ SMYLEvideo IP を活用してみたい（約 15%）
⑥ 次世代製品を開発する上でのマイクロプロセッサの課題
・処理性能（約 35%）
・消費電力（約 50%）
・コスト（約 40%）

尚、アンケート結果は、図 5.2-1 に示す。

表 5.2-1 アンケート実施サマリ

No.	展示会等名	開催日	会場	アンケート種類	回答者数	特筆すべき主な回答
1	ESEC2011	2011/5/11～5/1	東京ビッグサイト	メニーコア、SMYLEプロジェクト全般	9	3次元実装の早期確立、低消費電力とパフォーマンスの両立を期待
2	CEATEC JAPAN 2011	2011/10/5～10/8	幕張メッセ	メニーコア、SMYLEプロジェクト全般	26	並列化の困難なアルゴリズムの高速化へのソリューションが必要
3	ET2011	2011/11/16～11/18	パシフィコ横浜	メニーコア、SMYLEプロジェクト全般	33	高級言語レベルでの並列処理記述、電源機器、クラウドの高速集合演算に期待
4	三菱電機	2011/12/2	三菱電機	メニーコア、SMYLEプロジェクト全般	17	宇宙機、信号処理、産業機器への採用の期待
5	CES2012	2012/1/10～1/13	米国ラスベガス	メニーコア、SMYLEプロジェクト全般	23	CG、ビデオゲーム、放送機器、航空機産業等への導入に期待
6	Bluespec User Group Meeting	2012/1/27	富士ソフトプラザ	メニーコア、SMYLEプロジェクト全般	39	ソフトウェアの自動並列化、処理能力が重要後、開発容易性と消費電力も重要要素
7	ESEC2012	2012/5/9～5/11	東京ビッグサイト	SMYLEvideo	30	
8	CEATEC JAPAN 2012	2012/10/2～10/6	幕張メッセ	SMYLEvideo	30	
9	ET2012	2012/11/14～11/16	パシフィコ横浜	SMYLEvideo	13	
10	CES2013	2013/1/8～1/11	米国ラスベガス	SMYLEvideo	－	
11	いばらき新技術・新工法提案型展示商談会inデンソー	2013/2/5～2/6	株式会社デンソー（メ	SMYLEvideo	－	

Q1. SMYLEvideoの技術説明を聞いて、どのように思いましたか？

①ターゲットアプリの選定

	素晴らしい	良い	普通	やや悪い	悪い	無回答	計
ESEC2011	7	15	3	0	0	5	30
CEATEC2012	8	25	2	0	0	1	36
ET2012	5	6	1	0	0	1	13
計	20	46	6	0	0	7	79

②システム開発の方向性

	素晴らしい	良い	普通	やや悪い	悪い	無回答	計
ESEC2011	13	11	2	0	0	5	31
CEATEC2012	10	23	1	0	0	2	36
ET2012	7	5	1	0	0	0	13
計	30	39	4	0	0	7	80

③並列ソフトウェア開発の方向性

	素晴らしい	良い	普通	やや悪い	悪い	無回答	計
ESEC2011	8	12	3	0	0	7	30
CEATEC2012	8	21	3	0	0	4	36
ET2012	9	3	1	0	0	0	13
計	25	36	7	0	0	11	79

④メニーコア・アーキテクチャの方向性

	素晴らしい	良い	普通	やや悪い	悪い	無回答	計
ESEC2011	9	15	2	0	0	4	30
CEATEC2012	13	18	2	0	0	3	36
ET2012	9	3	1	0	0	0	13
計	31	36	5	0	0	7	79

Q2. SMYLEvideoはIPとして提供する予定です。

① SMYLEvideo IPを活用してみたいですか

	はい	いいえ	わからない	無回答	計
ESEC2011	3	4	21	2	30
CEATEC2012	5	3	26	2	36
ET2012	4	0	6	3	13
計	12	7	53	7	79

Q4. 現在、マルチコア/メニーコアを使用していますか？

	はい	いいえ	無回答	計
ESEC2012	6	18	6	30
CEATEC2012	6	15	15	36
ET2012	6	3	4	13
計	18	36	25	79

Q6. 次世代製品を開発する上で、現在使用しているマイクロプロセッサについて、どう思われていますか？・

■処理性能

	かなり足りない	やや不足	どちらでもない	やや余裕がある	余裕がある	無回答	計
ESEC2011	5	6	5	5	1	8	30
CEATEC2012	0	10	9	2	0	15	36
ET2012	3	4	1	0	0	5	13
計	8	20	15	7	1	28	79

■消費電力

	かなり大きい	やや大きい	どちらでもない	やや低い	かなり低い	無回答	計
ESEC2011	8	7	4	1	2	8	30
CEATEC2012	8	10	4	1	0	13	36
ET2012	3	4	1	0	0	5	13
計	19	21	9	2	2	26	79

■コスト

	かなり高い	やや高い	どちらでもない	やや低い	かなり低い	無回答	計
ESEC2011	8	5	8	1	0	8	
CEATEC2012	2	11	9	0	0	14	
ET2012	2	4	2	0	0	5	
計	12	20	19	1	0	27	79

図 5.2-1 SMYLEvideo アンケート調査結果サマリ

5.3　メニーコア市場調査：ライフイノベーション、インターネットアプリケーション、ファイナンス分野

4.3 節で紹介したベンチマークセット BEMAP はメニーコア市場を意識しており、本調査の結果を反映して開発されたものである。BEMAP は CLTrump と PEMAP の評価に利用することを前提として開発されている。調査では、これまでフィックスターズで実際に「高速化」という依頼で受託した案件の情報からメニーコアプロセッサで利用が想定される処理を複数選択した。

図 5.3-1 に BEMAP と CLTrump の関係を示している。BEMAP では図の緑部分の手動で最適化したコードが含まれており、さらに図の青部分の最適化前のコードが含まれている。青部分のコードを CLTrump で並列化コードに書き換え、最適化したコードと比較が可能である。CLTrump の場合は、性能比較と出力結果の検証が可能である。

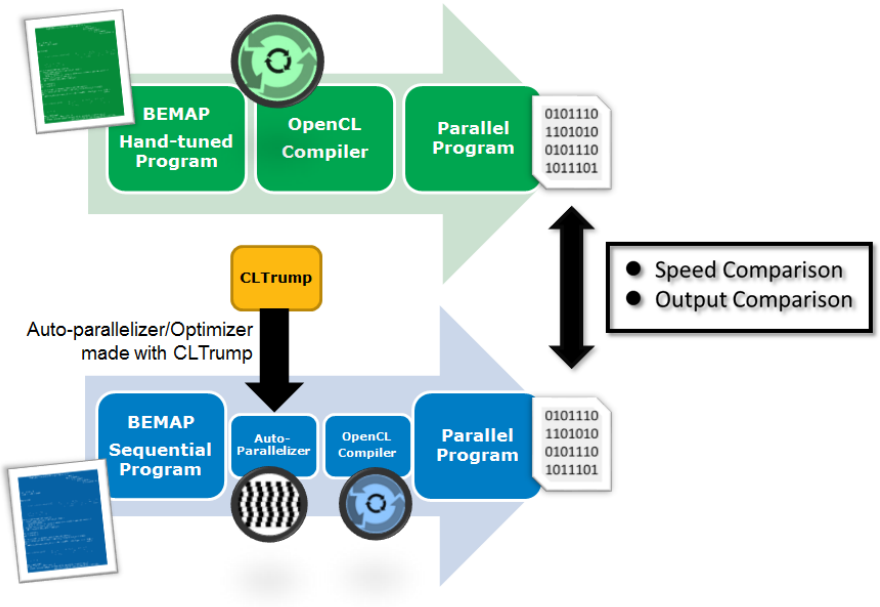


図 5.3-1 BEMAP と CLTrump の関係

図 5.3-2 に BEMAP と PEMAP の関係を示している。CLTrump と同様に、PEMAP でも BEMAP に含まれる青部分の最適化前のコードを処理することで性能見積もりが可能となり、実際の手動で最適化した BEMAP のコードとの性能比較ができる。

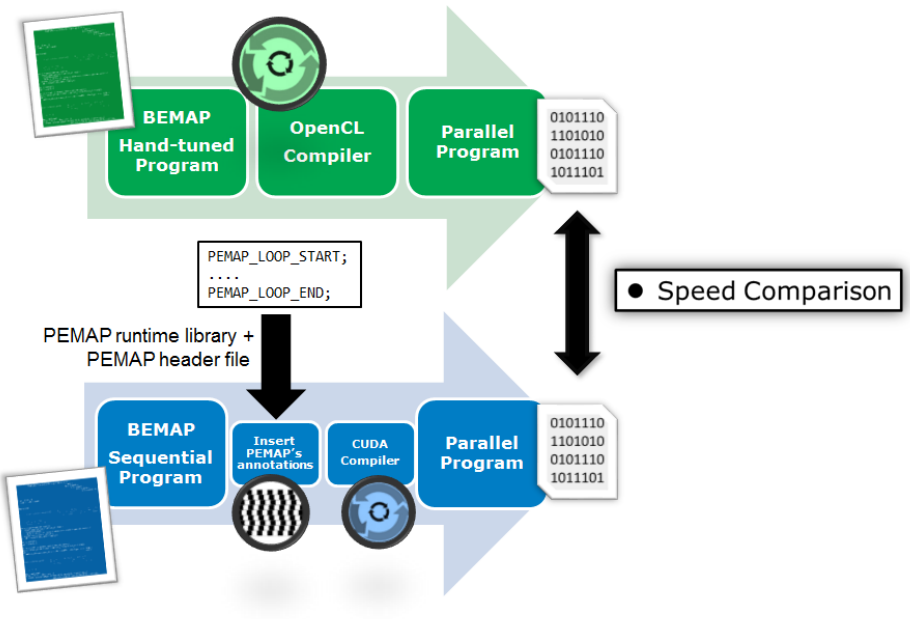


図 5.3-2　BEMAP と PEMAP の関係

5.4　開発環境

マルチ・メニーコアを有効利用するためのソフトウェア開発環境について、最新の技術動向、採用する企業側の課題について調査した。マルチタスク、マルチスレッドにより構成されるような並列処理構造を持つソフトウェアはこれまでも作られて来た。しかし、コアのマルチ・メニーコア化により、これまでの OS などによる仮想的な動作環境での並列から現実の複数コア上での並列動作が実現され、今後は並列処理にまつわる課題（メモリ破壊、デッドロック等）がより顕在化する。その様な問題の解決のために、既存コードからの並列化抽出、または上流から並列設計技術を支援するツールの利用が有効である。また、並列処理ソフトウエアは動作の保証が課題であるため、妥当性、正当性を保証する設計手法、記述方法、検証手法といった技術の確立、標準化が望まれる。それらを実現するためには、モデルベース開発のプロセスに並列処理開発を支援するため Tool Chain（並列化、並列設計、検証など）を構築する事が望まれる。

5.5　オペレーティングシステム

再委託項目である「動作環境を中心としたマルチ・メニーコア普及促進に向けた調査・検討」を実施した。調査方法は、技術動向に関しては学会、論文などを中心に実施し、市場動向に関しては Web と関連企業のヒアリングなどを用いた。具体的には、国際学会では MSRC2011 と HiPEAC2012 に参加した。論文調査に関しては、論文、オープンソース/標準仕様、市販製品ホワイトペーパー等からマルチコア/メニ―コア関連技術を選定し調査を行った。これらの調査の結果として以下の知見を得ることができた。

1. 次世代のマルチ・メニ―コアハードウェアは、キャッシュコヒーレンシを持たない分散メモリ型のアーキテクチャで、同一命令セットでも性能差を持つ複数の汎用プロセッサと、GPU などの応用特化した専用プロセッサのヘテロジニアスな構成となる
2. 次世代のマルチ・メニ―コアハードウェアに適応したメニ―コアオペレーティングシステムが新しいアーキテクチャで開発される必要がある。
3. メニ―コアオペレーティングシステムは、ソフトウェアの再利用性や並列性などさまざまな要求に応え、複数のアプリケーションインタフェースを揃える。
4. メニ―コアの適用分野としては、リアルタイム要求やスループットなど異なる要求を持つ複数アプリケーションを同時実行するシステムが想定される。
5. メニ―コアハードウェアとオペレーティングシステムの性能を引き出すための、並列化コンパイラなどの支援ツールとの連携が望まれる。
6. コア数やメモリ階層などメニ―コアハードウェアの特性を、オペレーティングシステムや並列化支援ツール、並列化コンパイラ、デバッグ/システム解析などのツールが統合的に認識できるような標準的なシステムコンフィギュレーションの仕組みが重要となる。

5.6　メニーコア・シンポジウム

メニーコア開発者、システム開発者、アプリケーション開発者、マーケティングなど、様々な分野を専門とする研究者・開発者・経営者が集まり、メニーコア応用といったニーズ面、ならびに、メニーコア・システム開発といったシーズ面の双方からの議論を行う場として、以下のように、NEDO 主催のメニーコア・シンポジウムを開催した。

第 1 回メニーコア・シンポジウム
日時：2012 年 3 月 30 日 13:00～17:45
場所：学術総合センター中会議場
内容：プロジェクト報告 5 件、招待講演 1 件、パネル討論 1 件
参加者：165 名（企業参加者 122 名、大学関係者 43 名）
ホームページ：http://home.jeita.or.jp/cgi-bin/page/detail.cgi?n=301&ca=1

第 2 回メニーコア・シンポジウム
日時：2013 年 1 月 30 日 13:00～18:30
場所：早稲田大学グリーンコンピューティングシステム研究開発センター
内容：プロジェクト報告 5 件、招待講演 1 件、パネル討論 1 件
参加者：165 名（企業参加者 122 名、大学関係者 43 名）
ホームページ：http://home.jeita.or.jp/cgi-bin/page/detail.cgi?n=486&ca=1

第 1 回シンポジウムにおいては、本プロジェクトを 1 年経過した段階での中間報告を行い、参加者から多くのフィードバックをいただいた。招待講演においては、早稲田大学理工学術院 笠原博徳 教授に「メニーコアプロセッサのための自動並列化・電力制御コンパイラと API」をご講演頂いた。医療応用など、今後のメニーコア適用分野拡大を目指した研究内容のご発表もあり、今後の方向性を示す意義あるご講演内容であった。また、パネル討論においては、パネリストとして、株式会社デンソー 半導体先行開発部 部長 石原秀昭氏、iSuppli ジャパン株式会社 副社長・主席アナリスト 南川明氏、株式会社東芝　セミコンダクター社半導体研究開発センター 主幹 宮森崇氏、早稲田大学理工学術院 笠原博徳教授、東京農工大学工学研究院 並木美太郎 教授をお迎えし、メニーコア市場から車載応用、半導体開発、コンパイラやオペレーティングシステムのシステムソフトウェアといった幅広い観点から、メニーコア実用化に向けた今後の課題等を議論した。特に、リアルタイム処理のための応答時間制約を如何に満たすかといった課題提起や、今後のヘテロジニアス・コンピューティングの重要性などについて様々な意見交換を行った。なお、本シンポジウムの内容に関しては[Ando2012]にてレポートが掲載されている。

第2回シンポジウムでは、本プロジェクトの最終成果を報告するとともに、最先端マルチコア／メニーコア技術に関する招待講演を企画した。1件目は、株式会社東芝セミコンダクター&ストレージ社 半導体研究開発センター　宮森高氏による「組み込み応用向け低消費電力メニ―コアSoCの開発」のご講演である。同社が開発した64コア・プロセッサに関して、世界最先端レベルの高性能かつ低消費電力技術をご紹介いただいた。2件目は、ARM社のJohn Goodacre氏による「ARM next generation 64bit processors for power efficient compute」のご講演である。ARMプロセッサは組込みシステムに広く普及しており、特に、最近ではヘテロジニアス型マルチコアの実用化を行っている。本講演では、ヘテロジニアス・アーキテクチャに加え、メモリにグローバなアドレスを割り当てリモートでのアクセスを可能とする構造などが紹介された。3件目は、Microsoft Research Redmond のAaron Smith氏による「E2: A Scalable Dynamic Multicore Architecture」のご講演である。動的に結合可能な新しいタイプのマルチコア・アーキテクチャとその制御に関する内容であった。これら3件の招待講演に加え、パネル討論では、パネリストとして、タイレラ・コーポレーション　石毛洋一氏、株式会社日立ハイテクノロジーズ 菊地修司氏、早稲田大学理工学術院 木村啓二 教授、イーソル株式会社　権藤正樹氏、株式会社トプスシステムズ鳥居淳氏をお招きし、基礎研究からビジネス展開までを見据えた議論を展開した。本パネルでは、特にメニーコアの活用を検討しているユーザ視点での意見が多くあり、意義ある議論を行うことができた。なお、本シンポジウムに関しては[Ando2013-1] [Ando2013-2] [Ando2013-3]にてレポートが掲載されている。

5.7　実用化に向けた取り組み：トプスシステムズ

トプスシステムズ社は、従来からの TOPSTREAM™ヘテロジニアス・マルチコア IP 事業に加えて、本プロジェクトの成果である 2 つのメニーコア（SMYLEvideo と SMYLEref）の実用化に向けて、次世代組込みシステム向け ASSP で No.1 になる新星企業を創ることを目指した「スマート ASSP 事業構想」を策定し、実用化に向けた活動を開始した。

「スマート ASSP 事業構想」では、国内外の優秀な技術者を大手半導体企業、セットメーカ、国内外の大学、シリコンバレーをはじめとする海外から集約し、システムアーキテクチャ、マルチ・メニーコアのハードウェアとソフトウェア、特に性能のスケーラビリティを追及することにより、利益率の高い ASSP（特定の顧客用にカスタマイズされていないため、顧客を限定せず、複数の顧客に汎用部品として提供することができる特定用途向け LSI）事業展開を目指す。また、その研究成果の実用化と次世代に向けた研究開発のための資金調達も行う計画である（図 5.7-1）。

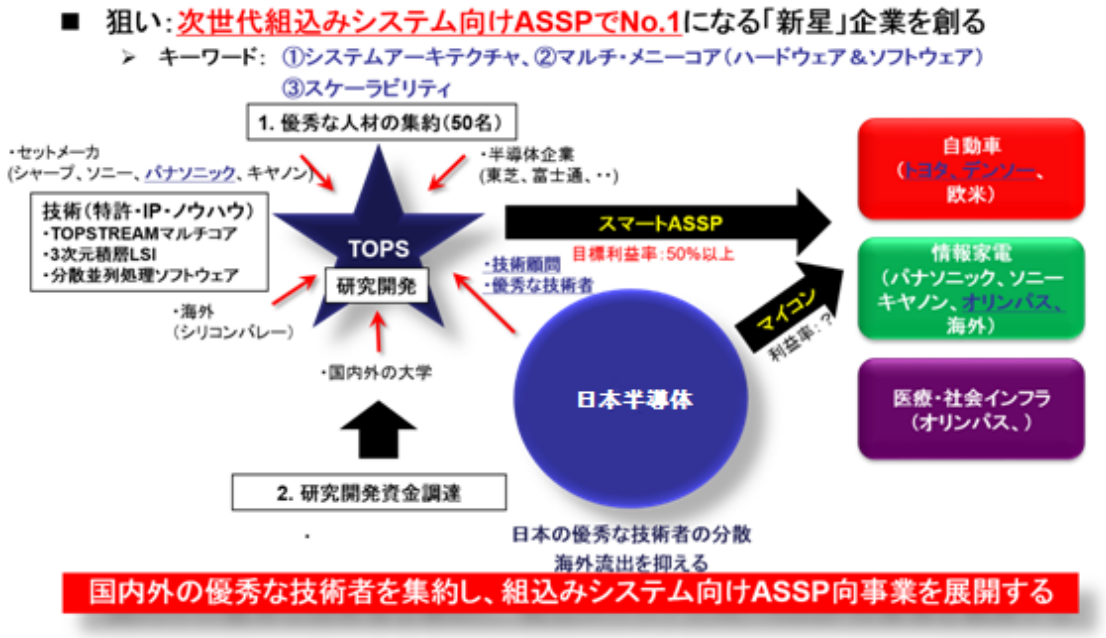


図 5.7-1 スマート ASSP 事業構想

本「スマート ASSP 事業構想」に基づく活動として、本研究開発期間中にも積極的に優秀な人材の集約に尽力してきた。それらを以下（表 5.7-1）に示す。

表 5.7-1 本プロジェクト期間中に集約した人材

役職	名前	所属
技術顧問	荒川文雄	現ルネサスエレクトロニクス
技術顧問	小川 泰	元ルネサスエレクトロニクス
技術アドバイザ	井上弘士	現九州大学准教授
技術アドバイザ	近藤正章	現電気通信大学准教授
マイクロプロセッサ研究開発部長	鳥居 淳	元ルネサスエレクトロニクス
マイクロプロセッサ事業開発部長	泉田正道	元 EPSON
ソフトウェア研究開発部長	中嶋廣二	元パナソニック
ソフトウェア研究開発部マネージャ	ｶﾞﾝﾊﾞ ｼﾞｮﾅ	元サイバネットシステム

「スマート ASSP 事業」では、図 5.7-2 に示すように 3 つのコア技術（TOPSTREAM™ヘテロジニアス・マルチコア（SMYLEvideo を含む）、TOPSTREAM™ 3D（3 次元積層 LSI）、FPMA（SMYLEref に基づくアクセラレータ））に基づく事業を展開する。

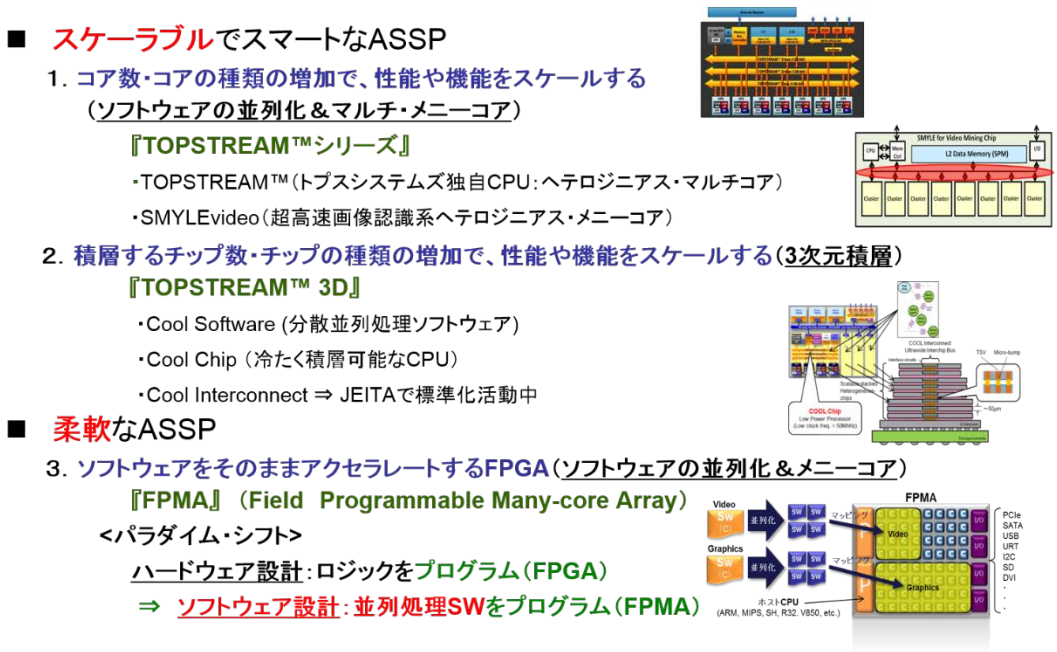


図 5.7-2 スマート ASSP 事業構想

SMYLEvideo メニーコアの実用化

トプスシステムズ社では、SMYLEvideo メニーコアの実用化に向けて、事業化計画を策定し、事業体制を強化してきた。SMYLEvideo は、画像認識系のプラットフォームとして、次世代組込みシステムの要件を満たすメニーコアとして、自動車・スマートフォン/タブレット・DTV(デジタル・テレビ)・監視装置・ロボットなどへの応用を目指す。SMYLEvideo は、トプスシステムズ社がハードウェア IP（特許技術を含む RTL 設計）の使用権を提供し、トプスシステムズ社の子会社である Cool Soft 社がその上で動作するスケーラブルな画像認識系ソフトウェアを提供する。将来的には、人間の 5 感に対応する機能を組込みシステム内でリアルタイムで実現することを目指した、次世代の研究開発と事業化を目指す。

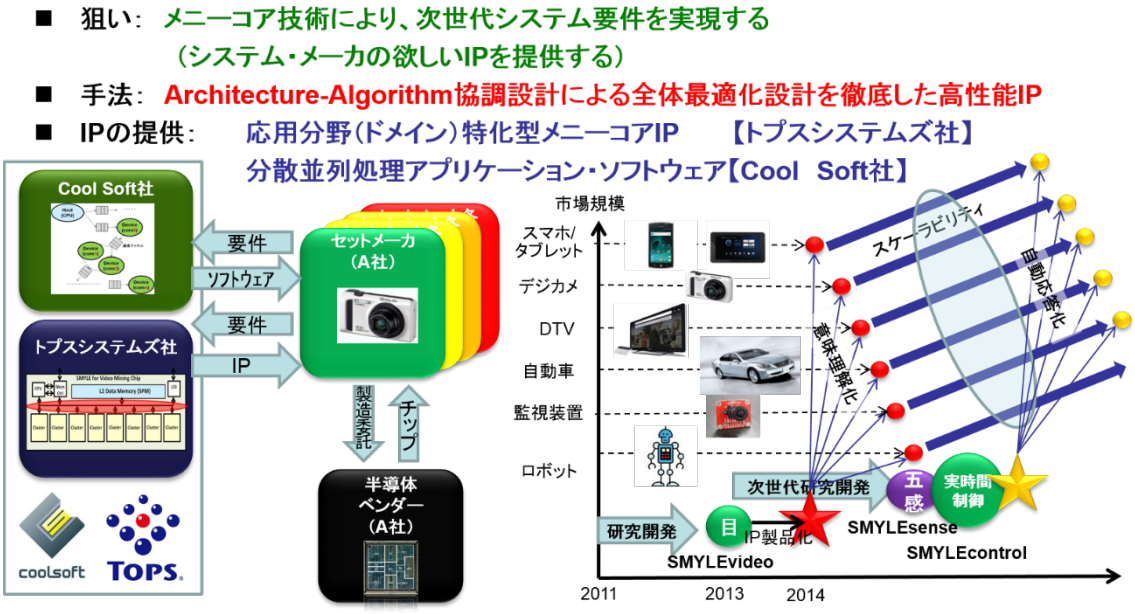


図 5.7-3 SMYLEvideo IP の事業化

また、事業体制の強化として、次の 2 つを進めた

① ソフトウェア開発体制の強化：ソフトウェア開発子会社（Cool Soft 社）の設立

② SoC 事業体制の強化：米 Xilinx 社とのパートナーシップ確立

ソフトウェア開発体制の強化策として、2012 年 1 月にトプスシステムズ社の 100%子会社として、株式会社 Cool Soft を設立し、優秀な技術者を集約しやすいようオフィスを神奈川県高津区（溝の口）のかながわサイエンスパーク（KSP）に開設し、マルチ・メニーコア向けのソフトウェア開発サービス事業を開始した。

また、SMYLEvideo の事業として、比較的単価の高い医療機器や車載システム等への応用展開のために、FPGA 上に SMYLEvideo メニーコアを搭載した製品の提供を目指して、大手

FPGA ベンダーである米 Xilinx 社とパートナー関係を構築した。

また、本プロジェクトを通じて、SMYLEvideo の実用化について、次の課題が明確になった。

１．画像認識系ソフトウェアの分散並列処理支援ツールの拡充
２．分散並列処理型の画像認識・画像処理ソフトウェア・ライブラリの拡充
３．SMYLEvideo メニーコアの更なる最適化と品質向上
　また、事業化に向けては、特に車載に関して、大手半導体ベンダーとのコラボレーションによる安定供給と長期サポートが求められている。

これら 3 つの課題についての解決策は検討済であり、今後、実用化研究を実施することでこれらの課題を解決し、実用化・事業化の確度を高めていく計画である。

SMYLEref メニーコアの実用化

トプスシステムズ社では、本研究成果の１つである SMYLEref メニーコアについて、事業化計画を策定し、九州大学および電気通信大学からの技術移転・実用化に向けての活動を進めていく。

SMYLEref は、ソフトウェア処理高速化（アクセラレーション）用のメニーコアとして、特長づけられる。トプスシステムズ社は、その利点を最大限に活用する事業として、論理回路(ロジック)をプログラムして処理の高速化を実現する既存の FPGA(Field Programmable Gate Arrary) の課題である膨大で複雑なハードウェアのプログラミングを解決するソリューションとして、SMYLEref を応用した FPMA (Field Programmable Many-core Array) というコンセプトに基づく事業化を狙う。FPMA は、逐次処理として記述されたソフトウェアを並列化してメニーコア上にマッピングすることにより、アクセラレーションを容易に行える。FPGA 市場の成長は著しいが、ハードウェア技術者でなければ使いこなすことができない。これに対し、ハードウェア技術者に比べて１桁以上多いソフトウェウエア技術者が使用可能な FPMA は、組込みシステムのアクセラレータとして大きくブレークする可能性を秘めている。

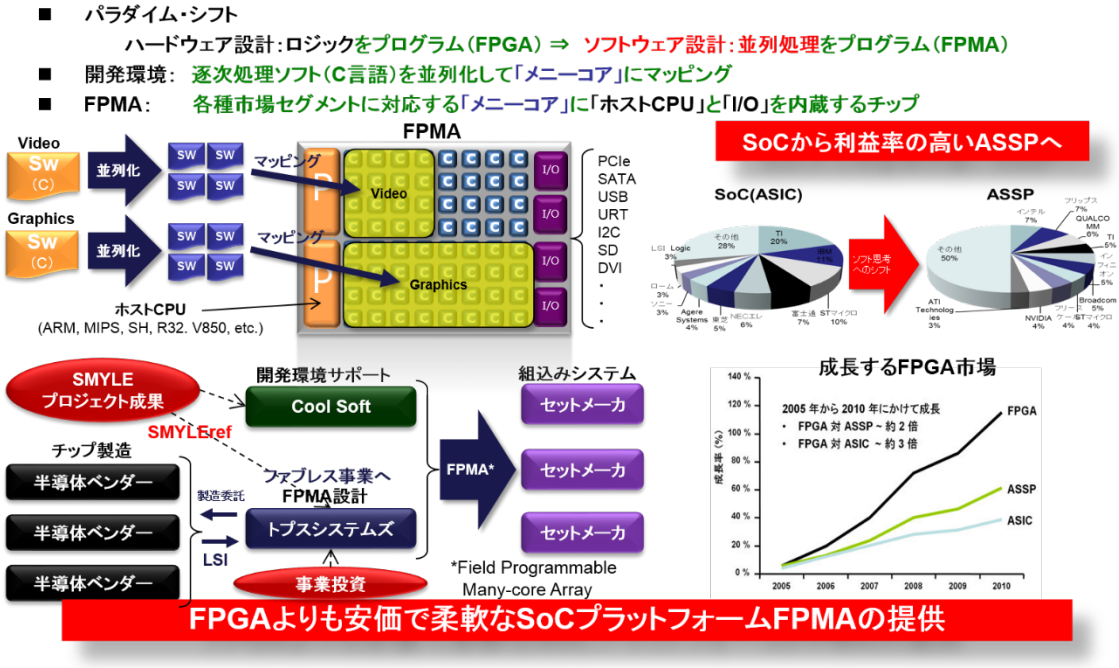


図 5.7-4 SMYLEref メニーコアの事業化

FPMA の事業化に向けた技術移転として、主に以下の 2 つを想定している。

① SMYLEref 設計データの移転 ：電気通信大学からトプスシステムズ社へ

② 特許技術（バリア同期）の移転 ：九州大学からトプスシステムズ社へ

SMYLEref 設計データの移転については、電気通信大学において、他社の権利に抵触しないプロセッサ・コアへの設計変更を行っている。設計変更完了後に、トプスシステムズ社において、演算処理性能・動作速度・回路規模等の評価を行った上で、実用化に向けての計画を策定する。

バリア同期特許技術については、トプスシステムズ社において、技術デューデリジェンスを行っており、SMYLEref のみならず、SMYLEvideo 等の他のマルチ・メニーコアへの適応性を評価・検討している。

5.8　実用化に向けた取り組み：フィックスターズ

実用化への取り組みとして各ツールで次のような活動を行っている。

CLTrump

CLTrump は SourceForge にオープンソースで公開しており、誰でも自由に利用することができる。応用ドメインを絞り込むことで最適化の品質を上げることが可能となる。また現状 OpenCL のコードを出力するが、改良することで他のプロセッサ向けのコードの出力も可能である。窓口はフィックスターズとなり問い合わせに対応している。図 5.8-1 が Web ページのスクリーンショットである。

図 5.8-1 CLTrump のページ（SourceForge）

PEMAP

PEMAP は手軽に性能予測をするという目的を実現するため、Web サービスとして利用できるようにしている。ユーザは PEMAP が稼働している Web サーバにアクセスし、図 5.8-2 のようなフロント画面に見積もりしたいソースコードを貼り付ける。測定したい移植後のプロセッサを指定して「Estimate」ボタンを押下すると、図 5.8-3 のような性能見積もり結果が出力される。図 5.8-4 のように変換されたダミーコードを確認することも可能である。

Web サービスの動作検証は完了しており当社内での利用を推進している。アクセス制御やセキュリティ対策を行った後に他のソフトウェアベンダに対してサービス提供を検討している。

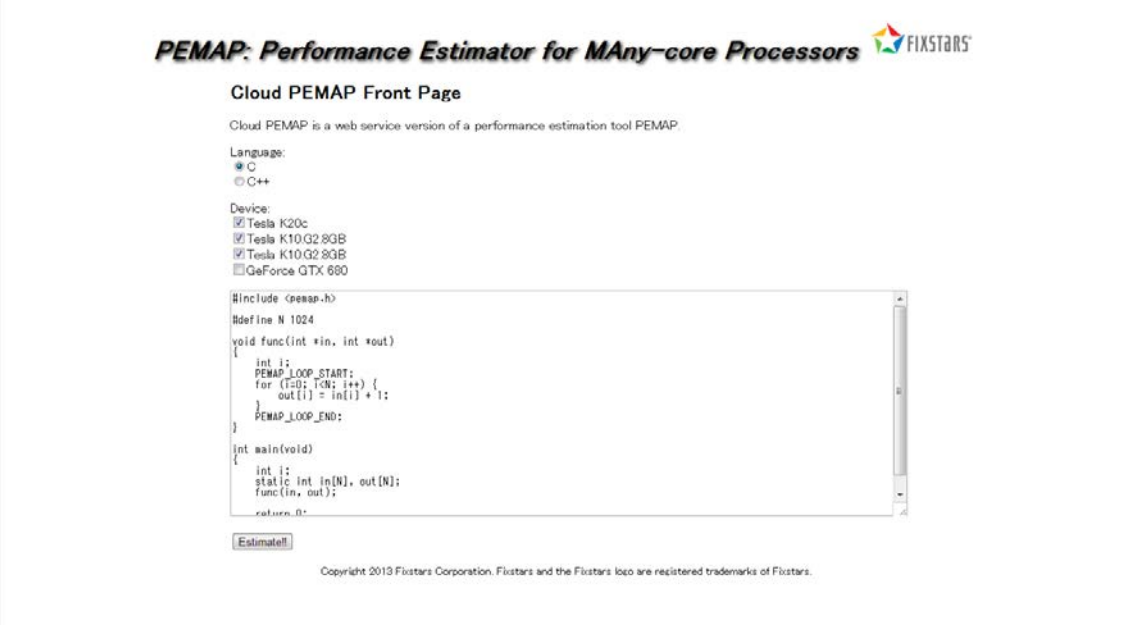


図 5.8-2　PEMAP サーバ（コード入力画面）

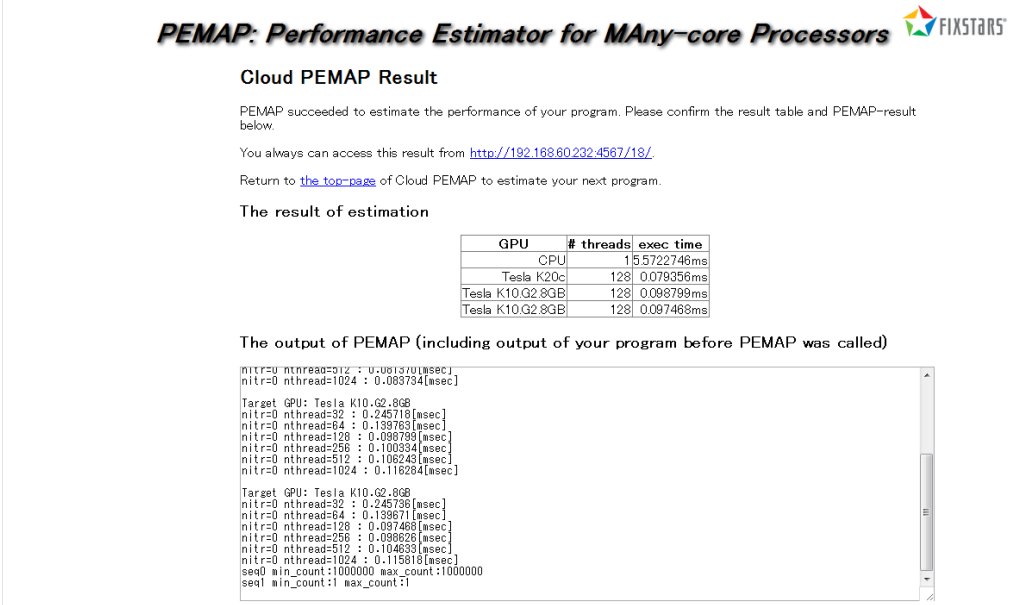


図 5.8-3　PEMAP サーバ（性能予測結果）

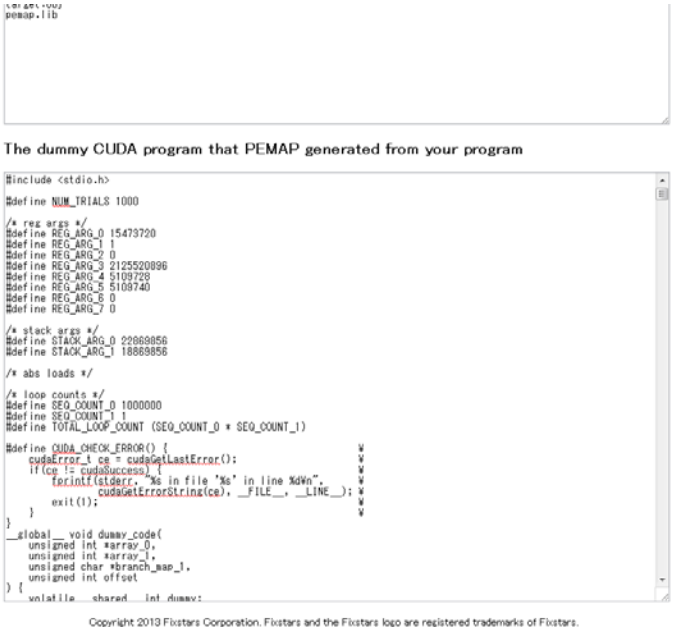


図 5.8-4　PEMAP サーバ（ダミーコード）

BEMAP

BEMAP は図 5.8-5 のようにオープンソースとして Web で公開している。公開版は BSD ライセンスとして無償で利用が可能である。各処理はフィックスターズが有償のマルチプラットフォームライブラリとして販売している M3 プラットフォームに組み込んであり。有償でのサポートも行っている。また一部の処理は OpenCV と組み合わせたアプリケーション一式も公開しており、すぐに応用検討が可能となっている。

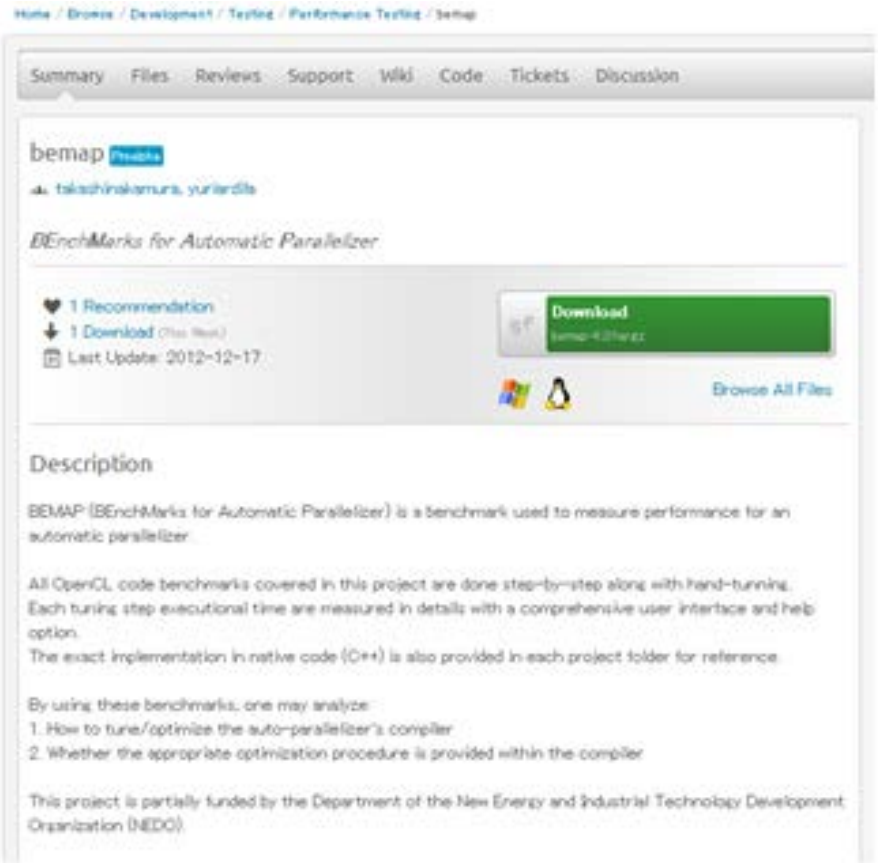


図 5.8-5 BEMAP の公開 Web ページ

図 5.8-6　BEMAP の SIFT を OpenCV から利用したデモ

参考文献

[Ando2012] Hisa Ando、 “従来比2倍の性能を1/10の消費電力で目指す - メニーコアシンポジウムが開催” マイナビニュース
http://news.mynavi.jp/articles/2012/04/05/many_core/index.html　2012年4月5日

[Ando2013-1] Hisa Ando、 “メニーコアシンポジウム - ビデオマイニング用メニーコアアーキテクチャ” マイナビニュース
http://news.mynavi.jp/articles/2013/02/05/manycore_tops/index.html 2013年2月5日

[Ando2013-2] Hisa Ando、 “メニーコアシンポジウム - ARMの次世代64ビットプロセサと将来の方向性” マイナビニュース
http://news.mynavi.jp/articles/2013/02/07/manycore_arm/index.html　2013年2月5日

[Ando2013-3] Hisa Ando、 “メニーコアシンポジウム - 東芝が将来の64コア画像認識SoCを発表” マイナビニュース
http://news.mynavi.jp/articles/2013/02/06/manycore_toshiba/index.html 2013年2月5日

6　総括および結論

本研究プロジェクトにおいては、次世代のメニーコア・アーキテクチャから、プログラム開発環境、評価環境までを一貫して開発した。まず、汎用メニーコア SoC の実現を目指した SMYLEref に関しては、「並列化された複数のタスクやアプリケーション」が同時に実行されるソフトウェアベースの新しい SoC 像を提唱し、メニーコア上にマッピングされる仮想アクセラレータ VAM (Virtual Accelerator on Many-core) の概念を導入した。この考え方に基づき、アーキテクチャ開発からプログラミングモデルの定義、各種 API の策定、プログラム実行のためのランタイムシステムの開発などを行った。これらに加え、動作検証のためのソフトウェア・エミュレータの開発、ならびに、128 コアの動作を可能とする FPGA プロトタイプとそれを用いた評価環境を構築した。プロセッサコアなどの一部に関しては既存設計データを再利用したものの、実質 2 年という短期間においてほぼスクラッチからの開発を行い、将来のメニーコア SoC のあるべき姿（ハードウェア・アーキテクチャからシステムソフトウェア、プログラミングモデルまでを含む）を示すと共に、その実現可能性を実証した。また、複数タスク／アプリケーションを実際に同時実行する FPGA プロトタイプ実行でのデータ取得にまでは至らなかったものの、本プロトタイプでの単一タスク実行での結果に基づき、複数タスク／アプリケーション実行をシミュレーションすることで性能を見積もった。その結果、従来の「並列化された単一タスク／プログラムを順次実行」する方式と比較して約 4 倍の性能向上を達成することができ、同一性能を想定する場合には動的消費電力量を 1/10 程度に削減可能であることを示した。

次に、ビジュアル・コンピューティングに特化した特定用途向けメニーコアである SMYLEvideo の開発においては、ハードウェア／ソフトウェア・コデザインを行い、従来プロセッサに対して大幅な性能向上を実現できることを明らかにした。アーキテクチャはヘテロジニアス・メニーコア構成である。ここでは実行対象を特定アプリケーションに絞り込んでいるため、処理対象に適した異なる種類のコアをクラスタ化している。また、プロセス間通信を隠蔽するための新しい通信方式の開発や、ブロック粒度での処理を導入することによる必要メモリ容量ならびにメモリ帯域の大幅削減を達成した。これらの成果を統合することで、従来型の汎用マルチコア・プロセッサ(4 コア(8 スレッド)約 2.9GHz≒24GOPS,130W)と比較して、性能が約 30 倍（700GOPS 相当)、消費電力 1/300 以下（350mW 程度)、商用の専用ハードウェアアクセラレータを具備した 4 コア程度の画像認識プロセッサに対して、性能 1.7 倍、消費電力 1/2 以下、回路規模 1/2 以下、という大きな優位性を示した。

ソフトウェア開発環境の構築においては、特にメニーコア時代のアプリケーション開発において直面する 3 つの課題を設定し、それらを解決すべく技術開発を行った。まず第 1 は並列化サポートのための環境構築であり、SMYLEref で前提としている OpenCL ベースのソフトウェア開発を想定し、ユーザとコンパイラがインタラクティブに並列化を進めるため

の方法論とツールを開発した。全てをコンパイラに任せるのではなく、プログラマの知識を活用し、かつ、可視化やソースコード解析結果などをプログラマが理解できるようサポートすることで、効率的な並列化が実現可能となる。このようなインタラクティブな並列化方式は、実用レベルでのプログラム並列化を実現するために極めて強力なアプローチである。第 2 は、メニーコアへの既存ソフトウェア資産の移植を検討するための性能推定ツールの開発である。メニーコア向けプログラムの開発には多くの時間的・人的コストを要する場合が多い。そのため、実際に詳細な移植作業に取りかかる前に、どの程度の性能向上効果を期待できるか精度よく見積もる必要がある。本研究プロジェクトではこれをサポートするための性能推定法、ならびに、それを可能にするツールを開発した。今後、メニーコアの普及に伴い、既存ソフトウェア資産を移植するケースが多くなると予想される。このような状況において、本環境は極めて重宝な道具として利用することができる。第 3 は、メニーコア評価のためのベンチマークの開発、ならびに、性能チューニングに関するノウハウの蓄積と一般公開である。メニーコアの普及を実現するためには、チューニングに関する知識の継承と共有が必要不可欠であり、本取り組みはその一端を担うものである。また、最適化技法とその効果に関する評価結果を合わせてソースコード・レベルで公開しているメニーコア向けベンチマーク・プログラムは世界的にも珍しく、その意義は大きい。

本プロジェクトにおいては、上述した研究内容に加え、各種要素技術の開発や市場調査など多くの成果を得た。そして、最終的には、産業面ならびに学術面に対して以下のように貢献した。

- 産業面：開発した SMYLEref ならびに SMYLEvideo はトプスシステムズが、また、ソフトウェア開発環境はフィックスターズが、それぞれのビジネス展開に活用するに至った。また、大学において開発した様々な要素技術を企業へと技術移転した。2 回のメニーコア・シンポジウムを通じ、メニーコアに関連する様々な分野の研究者や技術者による分野横断型の議論を展開した。これにより、解決すべき課題や市場要望などを共有することができた。
- 学術面：世界的にトップクラスの国際会議での発表や、各種招待講演、チュートリアルなど、多くの情報発信を行った。また、研究成果が認められ、国際会議におけるベストペーパー賞、ならびに、国内シンポジウムにおける優秀ポスター賞を受賞したことは特筆すべき成果である。

本プロジェクトでは、研究期間ならびに予算の関係上、チップ試作を行い提案方式の有効性を実チップとして実証するには至っていないが、今後の IP コアビジネス、さらには、ソフトウェア開発ビジネスにおける可能性を明らかにした。また、上記に加え、メニーコア向けベンチマーク・プログラムのオープンソース化や、FPGA ボードを用いたメニーコア評価環境の公開など、本プロジェクトに参画したメンバーのみならず、メニーコア研究開

発ならびに応用を検討する様々な企業／研究機関が本成果を活用できるものとした。今後、これらの成果を活用して、多くの企業や研究機関がメニーコア研究を促進し、その結果として、我が国における半導体産業ならびに組込み産業がより活性化することを望む。

添付資料：研究発表、講演、特許など

論文誌（査読付き）

1. Son Truong Nguyen, Masaaki Kondo, Tomoya Hirao, and Koji Inoue, "A Prototype System for Many-core Architecture SMYLEref with FPGA Evaluation Boards", IEICE Transactions on Information and Systems (採録済み).
2. 今村智史，佐々木広，福本尚人，井上弘士，村上和彰，“コア数と動作周波数の動的変更によるメニーコア・プロセッサ性能向上手法の提案，” 情報処理学会論文誌 コンピューティングシステム（ACS），第 5 巻，第 4 号, pp.24-35, 2012 年.

国際会議（査読付き論文）

3. Keita Nakajima, Takuji Hieda, Ittetsu Taniguchi, Hiroyuki Tomiyama, and Hiroaki Takada, "A Fast Network-on-Chip Simulator with QEMU and SystemC," In Proc. of International Workshop on Advances in Networking and Computing (WANC) in conjunction with International Conference on Networking and Computing (ICNC), pp. 298-301, Naha, Okinawa, Dec. 2012.
4. Mitsuya Uchida, Ittetsu Taniguchi, Hiroyuki Tomiyama, Masahiro Fukui, "Energy-Aware ILP-based Instruction Scheduling for Fine-Grained Power-Gated VLIW Processors," In Proc. of International SoC Design Conference (ISOCC), pp. 139-142, Jeju, Korea, Nov. 2012.
5. Junya Kaida, Takuji Hieda, Ittetsu Taniguchi, Hiroyuki Tomiyama, Yuko Hara-Azumi, and Koji Inoue, "Task Mapping Techniques for Embedded Many-core SoCs," In Proc. of International SoC Design Conference (ISOCC), pp. 204-207, Jeju, Korea, Nov. 2012.
6. Hiroshi Sasaki, Teruo Tanimoto, Koji Inoue, and Hiroshi Nakamura,“Scalability-Based Manycore Partitioning,” International Conference on Parallel Architectures and Compilation Techniques (PACT), pp.107-116, Sep. 2012.
7. Satoshi Imamura, Hiroshi Sasaki, Naoto Fukumoto, Koji Inoue, and Kazuaki Murakami, "Optimizing Power-Performance Trade-off for Parallel Applications through Dynamic Core-count and Frequency Scaling", 2nd Workshop on Runtime Environments/Systems, Layering, and Virtualized Environments (RESoLVE), in conjunction with ASPLOS 2012, Mar. 2012.
8. Kohei Aoki, Ittetsu Taniguchi, Hiroyuki Tomiyama, Masahiro Fukui, "Architecture Optimization based on a Branch-and-Bound Strategy for Low-Energy Embedded VLIW Processors," In Proc. of International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC), pp. 207-210, Gyeongju, Korea, June 2011.
9. Mitsuya Uchida, Ittetsu Taniguchi, Hiroyuki Tomiyama, Masahiro Fukui,

"Energy-Aware ILP-based Instruction Scheduling for Fine-Grained Power-Gated VLIW Processors," In Proc. of International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC), pp. 211-214, Gyeongju, Korea, June 2011.

国際会議（招待論文）

10. Koji Inoue, "SMYLE Project: Toward High-Performance, Low-Power Computing on Manycore-Processor SoCs," In Proc. of Asia and South Pacific Design Automation Conference (ASP-DAC), Invited paper, pp. 558-560, Yokohama, Japan, Jan. 2013.
11. Hiroyuki Tomiyama, Takuji Hieda, Naoki Nishiyama, Noriko Etani, Ittetsu Taniguchi, "SMYLE OpenCL: A Programming Framework for Embedded Many-Core SoCs," In Proc. of Asia and South Pacific Design Automation Conference (ASP-DAC), Invited paper, pp. 565-567, Yokohama, Japan, Jan. 2013.
12. Masaaki Kondo, Son Truong Nguyen, Tomoya Hirao, Takeshi Soga, Hiroshi Sasaki, and Koji Inoue, "SMYLEref: A Reference Architecture for Manycore-Processor SoCs", In Proc. of Asia and South Pacific Design Automation Conference (ASP-DAC), Invited paper, pp. 561-564, Yokohama, Japan, Jan. 2013.
13. Yuri Ardila, Natsuki Kawai, Takashi Nakamura, Yosuke Tamura, "Support Tools for Porting Legacy Applications to Multicore" In Proc. of Asia and South Pacific Design Automation Conference (ASP-DAC), Invited paper, Yokohama, Japan, Jan. 2013.
14. Yukoh Matsumoto, Hiroyuki Uchida, Michiya Hagimoto, Yasumori Hibi, Sunao Torii, Masamichi Izumida, " Manycore Processor for Video Mining Applications," In Proc. of Asia and South Pacific Design Automation Conference (ASP-DAC), Invited paper, pp. 574-575, Yokohama, Japan, Jan. 2013.

国際会議（招待講演、論文なし）

15. Koji Inoue, "Future Low-Power SoC," Design Automation and Test in Europe (DATE), Mar. 2013.
16. Yukoh Matsumoto, " Smart Home to be Enabled by Next Generation Scalable Heterogeneous Multi-Core / Many-Core and 3D LSI stacking," U-home2012, Dalian, Aug. 2012.
17. Hiroyuki Tomiyama, "OpenCL Compiler and Runtime Library for Embedded Manycore SoCs," International Forum on Embedded MPSoC and Multicore (MPSoC), Quebec, Canada, July 2012.
18. Koji Inoue and Masaaki Kondo, "A Manycore Architecture SMYLEref and its

Evaluation Environment", International Forum on Embedded MPSoC and Multicore (MPSoC), Quebec, Canada, July 2012.

19. Yukoh Matsumoto, " Distributed Processing Heterogeneous Multicore / Manycore / 3D Multichip SoC with Zero-Overhead Message Passing and Stream Processing HW/SW mechanism," MPSoC'12, Quebec, July. 2012..
20. Hiroyuki Tomiyama, "Challenges of Programming Embedded Many-Core SoCs with OpenCL," International Forum on Embedded MPSoC and Multicore (MPSoC), Beaune, France, July 2011.

国際会議（査読付きポスター、論文あり）

21. Sunao Torii, Michiya Hagimoto, Hiroyuki Uchida, Masamichi Izumida, Yukoh Matsumoto, " SMYLEvideo: Distributed Processing on Scalable Heterogeneous Manycore Architecture for Video Mining Applications," Poster presentation at COOL Chips XVI, Yokohama, Japan, Apr. 2013（採録済み）.

国際会議（査読付きポスター、論文なし）

22. Takuji Hieda, Naoki Nishiyama, Ittetsu Taniguchi, Hiroyuki Tomiyama and Koji Inoue, "An OpenCL Implementation Supporting Task Parallel Execution on Embedded Many-core Architecture," Poster presentation at Workshop on Designing for Embedded Parallel Computing Platforms: Architectures, Design Tools, and Applications (DEPCP) in conjunction with Design Automation and Test in Europe (DATE), Dresden, Germany, March 2012.

国内学会（査読付き論文）

23. 橋本 崇浩、 近藤 正章、 和田 康孝、 本多 弘樹、 "マルチコア・プロセッサ向けのヘルパースレッドによるキャッシュ制御支援手法の提案"、先進的計算基盤システムシンポジウム SACSIS2013, 2013 年 5 月(発表予定)

国内学会（査読付きポスター、論文あり）

24. グェンチュオンソン、 レイジャオ、 近藤正章、 平尾智也、 曽我武史、 井上弘士、 "FPGA によるメニーコア・プロセッサ SMYLEref の評価環境の構築"、 先進的計算基盤システムシンポジウム SACSIS2012, pp.14-15, 神戸、 2012 年 5 月

国内学会（査読なし論文）

25. 中村駿介、 青木康平、 内田充哉、 谷口一徹、 冨山宏之、 福井正博、 “細粒度電源管理を考慮した基本ブロックレベル消費エネルギー推定手法” 情報処理学会

SLDM/EMB/電子情報通信学会 CPSY/DC 研究会、 対馬、 2013 年 3 月

26. 江谷典子、 稗田拓路、 冨山宏之、 "SMYLE OpenCL デバイスにおける並列プログラミングモデルの実現と評価" 情報処理学会 OS 研究会、 岡山、 2013 年 2 月
27. 稗田拓路、 江谷典子、 西山直樹、 谷口一徹、 冨山宏之、 グェン チュオンソン、 近藤正章、 曽我武史、 平尾智也、 井上弘士、 "SMYLE OpenCL の実装と 128 コア上での評価実験" 情報処理学会 ARC/電子情報通信学会 ICD 研究会、 東京、2013 年 1 月
28. Yuri Ardila, Natsuki Kawai, Takashi Nakamura, Yosuke Tamura, "BEMAP: BEnchMarks for Automatic Parallelizer",、第 195 回計算機アーキテクチャ研究発表会、 2013 年 1 月
29. 河合夏輝、 Yuri Ardila、 中村孝史、 田村陽介、 "PEMAP: PErformance Estimatior for Many-core Processors"、 第 195 回計算機アーキテクチャ研究発表会、 2013 年 1 月
30. 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 EMB/OS 研究会、 東京、 2012 年 12 月
31. 甲斐田純也、 稗田拓路、 谷口一徹、 冨山宏之、 原祐子、 井上弘士、 "組込みメニーコア SoC 向けタスクマッピング手法" DA シンポジウム論文集、 pp. 67-72、 下呂、 2012 年 8 月
32. 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 シンポジウム論文集、 pp. 73-78、 下呂、 2012 年 8 月
33. 今村智史、佐々木広、福本尚人、井上弘士、村上和彰、 "コア数と動作周波数の動的変更によるメニーコア・プロセッサ性能向上手法の提案" 研究報告計算機アーキテクチャ（ARC）, 2012-ARC-200(17), pp.1-9, 2012 年 4 月
34. 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、" 電子情報通信学会 CPSY/DC/情報処理学会 SLDM/EMB 研究会、 松島、 2012 年 3 月
35. 曽我武史、近藤正章、佐々木広、平尾智也、井上弘士、 “メニーコアプロセッサを対象とした柔軟性を有するハードウェアバリア機構の提案“、 情報処理学会研究報告 2012-ARC-199、2012 年 3 月
36. 橋本崇浩、 井上功一、 近藤正章、 平澤将一、 本多弘樹、 "マルチコア・プロセッサ向けのヘルパースレッドによるキャッシュ制御支援手法の検討"、 情報処理学会研究報告 Vol.2012-ARC-199, No.13 長崎、 2012 年 3 月
37. 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、 東京、 2012 年 1 月

38. 青木康平、 谷口一徹、 冨山宏之、 福井正博、 "分枝限定法に基づく組込み向け VLIW 型プロセッサのアーキテクチャ探索手法、" 電子情報通信学会 VLD 研究会、 那覇、 2011 年 3 月
39. 内田充哉、 谷口一徹、 冨山宏之、 福井正博、 "マルチサイクル演算に対応した VLIW 型プロセッサ向け消費電力最小命令スケジューリング手法、" 電子情報通信学会 VLD 研究会、那覇、2011 年 3 月

国内（招待講演、パネルなど、論文なし）

40. 井上弘士、“メニーコアプロセッサの組込みシステム応用”第 16 回 組込みシステム開発技術展、2013 年 5 月
41. 冨山宏之、 “SMYLE プロジェクト” 情報処理学会関西支部ものづくり基盤コンピューティング研究会、 2013 年 3 月
42. 松本 祐教、"スマートな次世代組込みシステムの実現を可能にする メニーコア・プロセッサ" 第 42 回 AIST・筑波大学・TCI ベンチャー技術発表会、 つくば、2013 年 3 月
43. 冨山宏之、 "SMYLE OpenCL: 組込みメニーコア SoC 向け OpenCL 環境" NEDO 主催・第 2 回メニーコアシンポジウム、2013 年 1 月
44. 近藤正章、 "組込みシステム向けメニーコア・アーキテクチャ SMYLEref とその評価環境"、 NEDO 主催・第 2 回メニーコアシンポジウム、 2013 年 1 月
45. 田村陽介、 "メニーコア時代のソフトウェア開発環境" NEDO 主催・第 2 回メニーコアシンポジウム、 2013 年 1 月
46. 松本 祐教、 " SMYLEvideo：ビデオ・マイニング向けメニーコア・アーキテクチャとその性能、" 第 2 回メニーコアシンポジウム、東京、 2013 年 1 月
47. 井上弘士、 石毛洋一、木村啓二、 権藤正樹、 鳥居淳、 "メニーコアは何をもたらすのか？～基礎研究からビジネス展開まで、" パネル討論、 第 2 回メニーコアシンポジウム、 東京、 2013 年 1 月
48. 井上弘士、 冨山宏之、 近藤正章、 "メニーコアプロセッサと組込みシステムへの応用 " チュートリアル、 組込みシステムシンポジウム論文集、 p. 1,東京、 2012 年 10 月
49. 井上弘士、,冨山宏之、 権藤正樹、 安里彰、 石毛洋一、 松本祐教、 "メニーコアプロセッサはメインストリームになり得るか？" パネル討論、 電子情報通信学会ソサイエティ大会、富山、 2012 年 9 月
50. 近藤 正章、 井上 弘士、 冨山 宏之、 “メニーコア・アーキテクチャ SMYLEref とその評価環境” 第 4 回アクセラレーション技術発表検討会、 福井、 2012 年 9 月
51. 井上弘士、“マイクロプロセッサ新時代の幕開け～マルチコアからメニーコアへ～”第 15 回 組込みシステム開発技術展、2012 年 5 月
52. 井上弘士、 冨山宏之、 近藤正章、 "SMYLE プロジェクトの紹介とメニーコア・プロ

セッサ SMYLEref の開発" NEDO 主催・第 1 回メニーコアシンポジウム、2012 年 3 月

53. 田村陽介、 "メニーコア時代のソフトウェア開発環境" NEDO 主催・第 1 回メニーコアシンポジウム、 2012 年 3 月
54. 松本 祐教、 "ビデオ・マイニング向けメニーコアの開発" 第 1 回メニーコアシンポジウム、 東京、 2012 年 3 月
55. 権藤正樹、 藤本洋、“メニーコア向け OS/開発環境調査”第 1 回メニーコアシンポジウム、 東京、 2012 年 3 月
56. 冨山宏之、 “MPSoC 向けシステムレベル設計技術” （株）半導体理工学研究センター、 第 39 回アドバンスト講座・高位設計セミナー～システムレベル設計支援技術の最新動向と活用法～、 2011 年 12 月
57. 松本 祐教、 "超高精度画像合成向けヘテロジニアス・メニーコア型アプリケーション・プロセッサ・アーキテクチャ" 第 152 回 システム LSI 設計技術研究発表会、 仙台、 2011 年 10 月
58. 松本 祐教、 “超高精度画像合成向けヘテロジニアス・メニーコア型アプリケーション・プロセッサ・アーキテクチャ” 第 3 回アクセラレーション技術発表討論会、 神戸、 2011 年 9 月
59. 中村孝史、 薦田登志矢、 圓戸辰郎、 田村陽介"CLtrump：プログラマとツールのインタラクティブな開発を実現する OpenCL 向け半自動並列化フレームワーク、" GTC Workshop Japan 2011 年 7 月

国内展示会

60. 組込み総合技術展（Embedded Technology）、立命館大学、 組込みメニーコア SoC 向け OpenCL 環境, 2012 年 11 月
61. CEATEC Japan 2012、 トプスシステムズ、 動画像をリアルタイムに処理するメニーコア技術, 2012 年 10 月
62. ESEC2012、 トプスシステムズ、SMYLEvideo ビデオマイニング向けメニーコアの開発、 2012 年 5 月 9 日～11 日
63. Embedded Technology 2012、 トプスシステムズ、 SMYLEvideo 次世代組込み向けメニーコア SOC の開発、 2012 年 11 月 14 日～11 月 16 日
64. いばらき新技術・新工法提案型展示商談会 in デンソー、SMYLEvideo ビデオマイニング向けメニーコアの開発、2013 年 2 月 5 日~2 月 6 日

国際展示会

65. 2013 International CES、 トプスシステムズ、 SMYLEvideo ~Scalable Manycore Processor for Video Mining Applications ~、2013 年 1 月 8 日(火)～1 月 11 日(金)

受賞

66. Keita Nakajima, Takuji Hieda, Ittetsu Taniguchi, Hiroyuki Tomiyama, and Hiroaki Takada, Best Paper Award, International Workshop on Advances in Networking and Computing (WANC), Dec., 2012
67. グェンチュオンソン、 レイジャオ、 近藤正章、 平尾智也、 曽我武史、 井上弘、, 優秀ポスター賞、 先進的計算基盤システムシンポジウム SACSIS2012、 神戸、 2012 年 5 月

特許

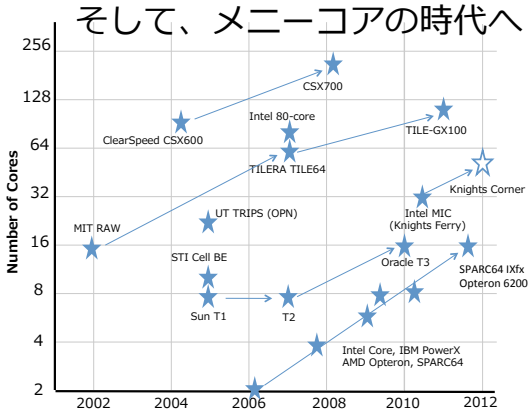
68. 中村孝史、河合夏輝、田村陽介、“実行時間算出装置、実行時間算出方法、及びプログラム” 出願番号 2012-251560, 出願日 2012 年 11 月 15 日
69. 曽我武史、 佐々木広、 井上弘士、 “同期処理回路及び同期処理方法” 出願番号 特願 2012-207071、 出願日 2012 年 9 月 20 日
70. 松本祐教、 内田裕之、 ″プロセッサコア、およびマルチコア・プロセッサ・システム″ 出願番号 特願 2012-15988、 出願日 2012 年 1 月 27 日
71. Yukoh Matsumoto and Hiroyuki Uchida, ″PROCESSOR CORE AND MULTI-CORE PROCESSOR SYSTEM″ 出願番号 13718796、 出願日 2012 年 12 月 28 日(米国)

添付資料：メニーコアシンポジウム講演スライド

第1回メニーコアシンポジウム
2012年3月30日

主催：NEDO　独立行政法人新エネルギー・産業技術総合開発機構

シングルからマルチ、そして、メニーコアの時代へ



SMYLEプロジェクト

～Scalable ManY-core for Low-Energy computing～

- NEDOグリーンITプロジェクト「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」
  - 研究期間は2010年12月～2013年2月
  - 産学連携による研究開発
  - 成果は可能な限り「Open」に!

KYUSHU UNIVERSITY　RITSUMEIKAN　UEC TOKYO　TAT 東京農工大学　eSOL　JEITA　TOPS　FIXSTARS Speed up your Business　Communication Technology Systems

メニーコアの普及と市場拡大に向けて

3つの「?」	本プロジェクトの取り組み
• どのようなメニーコアを作れば良いのか?	• メニーコア向けOS/開発環境/市場調査
• どのようにソフトウェアを開発すれば良いのか?	• メニーコアSoCプラットフォームの開発
• どのようにメニーコア研究開発を進めれば良いのか?	• SW資産の「移植」に重きをおいたプログラム開発環境の構築
	• ドメイン特化型メニーコアの開発
	• 各種評価／開発環境のオープン化

メニーコア・シンポジウムの趣旨

- メニーコア研究開発の活性化
  - ハードウェアからソフトウェア、応用、市場拡大までを見据えた串刺し型の議論を展開
  - メニーコアは「使えるのか?」「売れるのか?」「解決すべき課題は何か?」
- SMYLEプロジェクトへのフィードバック
  - 成果のオープン化に向けたご意見／ご要望
- 我が国のエレクトロニクス産業の復活に貢献

シンポジウムの内容

	本プロジェクトの取り組み
「マルチ・メニーコア技術の応用拡大に向けた市場調査およびプロジェクト提案」 JEITA 荒川文男	
「メニーコア向けOS/開発環境調査」 イーソル 権藤正樹, キャッツ　藤本洋	• メニーコア向けOS/開発環境/市場調査
「SMYLEプロジェクトの紹介とメニーコア・プロセッサSMYLErefの開発」 九大 井上弘士, 立命館 富山宏之, 電通大 近藤正章	• メニーコアSoCプラットフォームの開発
「メニーコア時代のソフトウェア開発環境」 フィックスターズ　田村陽介	• SW資産の「移植」に重きをおいたプログラム開発環境の構築
「ビデオ・マイニング向けメニーコアの開発」 トプスシステムズ　松本祐教	• ドメイン特化型メニーコアの開発
【招待講演】「メニーコアプロセッサのための自動並列化・電力制御コンパイラとAPI」 早稲田　笠原博徳	• 各種評価／実行環境のオープン化
【パネル討論】「メニーコア時代のチャレンジ～普及に向け何が必要か？～」	

「2種類」のアンケートのお願い

- アンケート
  - メニーコアシンポジウムアンケート
  - 組込みマルチコアに関するアンケート
- 提出場所
  - お帰りの際に「受付」でご提出ください。
- プレゼント
  - もれなく「JEITA マルチコアハンドブック応用編（定価6,900円）」を無償でお渡しします

マルチ・メニーコア技術の応用拡大に向けた市場調査およびプロジェクト提案

マイクロプロセッサ専門委員会　委員長
マルチ・メニーコア応用拡大検討委員会　委員長
ルネサス エレクトロニクス株式会社 CPU開発第一部 主管技師
荒川 文男

2012年3月30日

JEITA 一般社団法人 電子情報技術産業協会
URL http://www.jeita.or.jp
マルチ・メニーコア応用拡大検討委員会／マイクロプロセッサ専門委員会

マイクロプロセッサ専門委員会

情報・産業社会システム部会の技術企画・標準委員会の下部組織

委員会の目的

- マイクロプロセッサ、システムLSI関連産業の振興、国際競争力の向上を目的に、新規需要分野の創出と拡大に向けた事業・調査研究を目的とする委員会
- 2006年度より、マルチコア技術が今後の関連産業振興に必須と考え、調査研究を実施

参加企業・大学／委員

ルネサスエレクトロニクス（株）	荒川 文男（委員長）
富士通セミコンダクター（株）	横田 廉（副委員長）
名古屋大学教授	枝廣 正人（客員）
パナソニック（株）	山野 敦雄
日本電気（株）	中村 祐一、久村 孝寛
（株）東芝	林 宏雄
イーソル（株）	権藤 正樹
キャツツ（株）	渡辺 政彦、藤本 洋
（株）トプスシステムズ	松本祐教

※プロセッサ関連のベンダ、ユーザに広く門戸を開放しています

これまでの活動と今後の予定

■'11.04～ NEDO PJ「マルチ・メニーコア技術適用アプリケーション拡大に向けた調査・検討」
■マルチコア懇談会（年2回主催、参加者を有識者20人程度に絞り、活発に議論）
'07.03 第　1回 自由討論、意見の収集
'07.09 第　2回 アプリ側視点での課題の明確化　デンソー／石原氏、富士通研／高橋氏、東芝／吉森氏
'08.03 第　3回 ソフト開発や検証の課題　ドコモ／千葉氏、東工大／吉瀬先生
'08.10 第　4回 ソフト開発の検証と品質確保　早大／中島先生、慶大／天野先生、デンソー／近藤氏
'09.03 第　5回 ソフト開発環境の現状と将来動向 KMC／由良氏、キャツツ／塚田氏、イーソル／権藤氏
'10.03 第　6回 マルチコアの普及・進化による日本の製品競争力強化戦略 京大／松山先生、東芝／境氏
'10.08 第　7回 カーエレクトロニクス進化の展望とマルチコアへの期待　名大／高田先生、デンソー／石原氏
'11.05 第　8回 我が国の競争力強化戦略とマルチコアへの期待 パナソニック／梶本氏、iSuppli／南川氏
'11.08 第　9回 日本の成長戦略とマルチ・メニーコアプロセッサの市場展望 日経BP／浅見氏、東大／小川先生
'12.03 第10回 マルチ・メニーコアプロセッサの動向と戦略 JEITA／荒川、キャツツ／藤本、イーソル／権藤
■CEATECコンファレンス（毎年2セッション主催）
'08.10　早大／木村先生、　名大／本田先生
'09.10　早大／中島先生、NEC、東大／枝廣氏、組込みマルチコアハンドブック基礎編配布・紹介
'10.10 九大、ISIT／村上先生、　デンソー／加古氏、組込みマルチコアハンドブック配布
'11.10　東北大／青木先生、豊田中研／内藤氏、組込みマルチコアハンドブック配布
■その他
'07.06 JEITA情報端末フェスティバル　早大／木村先生（組込み向けマルチコアの技術動向）
'09.09 組込みマルチコアハンドブック基礎編作成・頒布
'10.02 組込みマルチコアハンドブック技術・応用編作成・頒布
'10.02 低消費電力メニーコアプロセッサシステム技術シンポジウム 後援（早大・NEDO共催）
'11.11 ET2011併催JEITAセミナー 名大／枝廣先生、九大／井上先生、日経BP／浅見氏
'12.03 第1回メニーコアシンポジウム（本日）

マルチ・メニーコア技術を適用するアプリケーションの拡大に向けた調査・検討

マルチ・メニーコアが必須かつ活用可能な技術であることを示し、国及び企業の投資を促す

①「マルチ・メニーコア技術適用アプリケーション拡大に向けた市場の調査・検討」 JEITA
- 市場規模、成長性、日本の強み分野の観点から、日本の半導体産業が力を入れるべき有望な適用分野、機器市場にフォーカスして調査・検討
- 調査会社のデータベース／ネットワークによる一次調査で、国内外の最新動向を把握
- 上記の分析／ヒアリングにより、国内外ユーザ企業のニーズ、特に潜在的なニーズを把握
- 日本の半導体産業の競争力強化に繋がる提言

②「開発環境を中心としたマルチ・メニーコア普及促進に向けた調査・検討」 キャツツ（株）
- ヒアリングにより、開発環境への期待と、開発に際しての障害などを具体的に把握
- 技術動向を調査し、現状と将来開発現場で必要になる技術についてまとめる
- 開発環境のあるべき姿についての提言

③「動作環境を中心としたマルチ・メニーコア普及促進に向けた調査・検討」 イーソル（株）
- ヒアリングにより、動作環境への期待や課題などを調査、検討
- 最新の技術動向を調査し、動作環境に求められる特性などを明らかにする
- 動作環境に求められる要件についての提言

マルチ・メニーコアの活用領域と問題点

■ マルチ・メニーコアの活用領域
- 限られた電力バジェットで、今後も性能向上させる領域
- 同等性能でも更に少ない電力バジェットでアプリを実現する領域
- 処理内容が柔軟かつ多様でハードウェアアクセラレータが不向きな領域

■ マルチ・メニーコアの普及阻害要因
- 膨大なシングルコア向けレガシーソフトウェアの存在
- マルチ・メニーコアを活用できる人材・環境の不備
- 使いたくても使えない、使うと大変そう

■ マルチ・メニーコア研究開発の停滞
- 企業は新技術開発の余力がなく、従来の延長を志向
- 新規PJを起こすにはスポンサーに新市場の存在を確信させることが必要
- 大部分のユーザは手ごろな技術があれば使うという立場
- ユーザは1社にキー技術を握られないように行動する
- マルチコアは先行開発から人材・環境整備へ進む動きが鈍い
- 海外が得意な情報系に比べて日本が得意な制御系での動きが鈍い
- 海外と差が広がると挽回困難
- 海外任せだと日本の強みは消されるか消える
- メニーコアは市場の立ち上がりが未だ読めない

IHS has acquired iSuppli

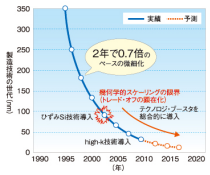
マルチ・メニーコア市場動向と市場拡大に向けた課題

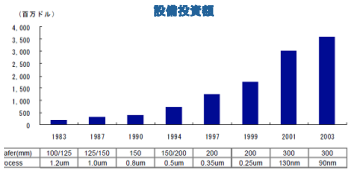
2011年7月25日

Prepared by
Akira Minamikawa
Japan Research
Phone +81 3 3562 1580
aminamikawa@isuppli.com

マルチコアプロセッサ躍進の背景

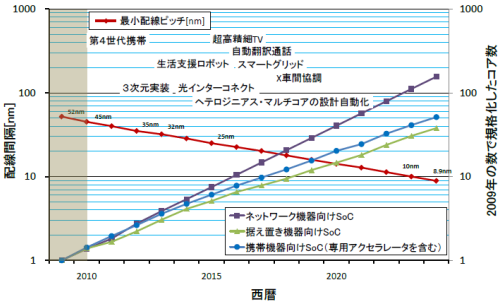
- 微細化の限界と投資コスト増大
- スーパスカラなどの命令レベル並列性の限界
- クロック周波数向上による発熱の問題
- 車載など搭載MCUの個数増加
- センサー増で入力情報増加による処理能力不足
- ビジネスの囲い込み





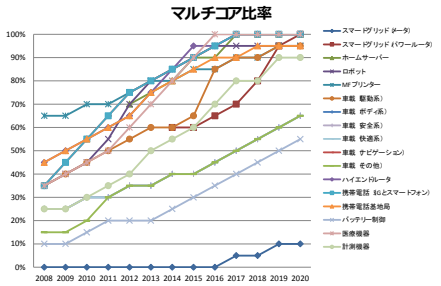
afer(mm)	100/125	125/150	150	150/200	200	200	300	300
ocess	1.2um	1.0um	0.8um	0.5um	0.35um	0.25um	130nm	90nm

ITRSの見方



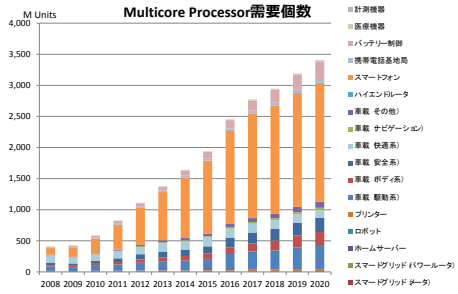
マルチコア比率の見方

- 電子機器のマルチコア化の比率を示している
- MFプリンタ、携帯電話基地局、ルータなどは既にマルチコア化が進んでおり調査対象の機器はスマートメータ以外で2020年までにはマルチコア比率が50%を超える



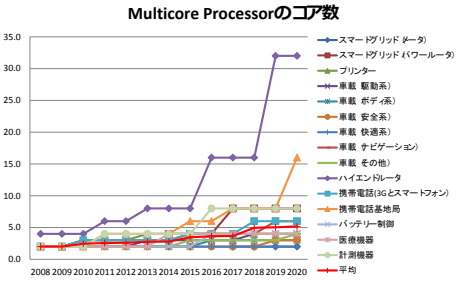
マルチコアプロセッサ需要個数

- マルコアプロセッサの需要個数ではスマートフォンが圧倒的に多く車載（駆動）、バッテリー制御が続く
- 2010年には587百万個が2020年には約6倍の3,404百万個にまで増する



コア数の推移

- ハイエンドルータ、基地局、計測機器、パワールータなどは演算処理増加要求が高くコア数が大きくなる
- 平均のコア数は2010年で2.5個、2020年には5.2個になると見ている



iSuppli Japan 最終調査報告概要

トレンド

- 高機能化を実現するためにシングルコアでは限界が見え始めている。
- 高性能化・多機能化と消費電力低減の要求がマルチコア化の原動力。
- 車載関連はフォールトトレラントの必要性からもマルチコア化を推進。しかし、デザインのリードタイムが長いため2020年までのマルチコア化の進展はゆっくりと進む。
- ハイエンドルータ、基地局、サーバーは16個以上のコアを要求している。
- シングルコアに固執する企業は見られず、価格によるマルチコア化の障害も聞かれない。

課題

- 様々な課題からユーザーはITRS予測より少ないコア数を想定している。
- 開発環境/動作環境の整備、API開発、レガシーソフト有効活用、マルチコア技術者の不足が課題。

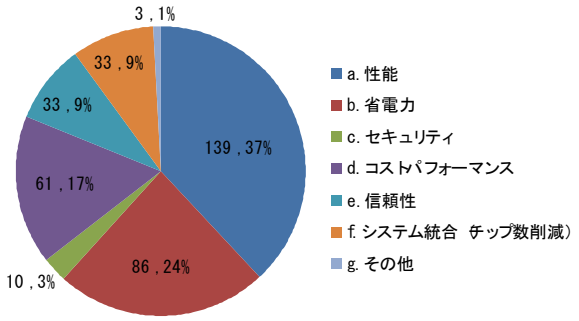
海外動向

- 海外企業が得意な分野（基地局、サーバーなど）では、2015年前後を目途にマルチコア化に対する課題を解決する見通しで進んでいる。
- Intel、AMD、TI、Freescaleなどの海外半導体企業はマルチコア化を推進しており、日系企業の2-3年先を歩んでいる。

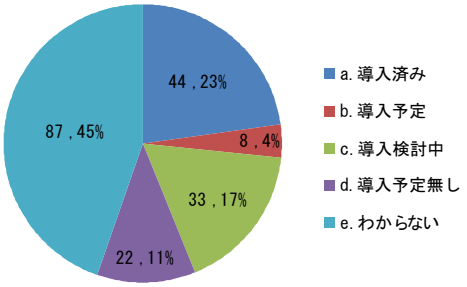
CEATEC2011　ET2011
マルチコア アンケート結果

アンケート実施概要
- 実施日：2011年10月5日、2011年11月18日
- 実施対象参加者：
  CEATEC2011　当委員会企画特別セッション
  ET2011　JEITAセミナーセッション
- アンケート回収：198名（両セッション合計）

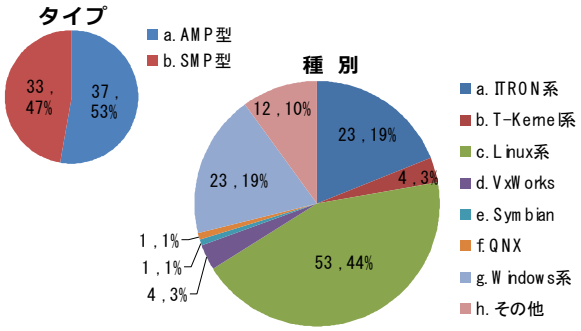
項目3. マルチコアに何を期待しますか（複数回答）



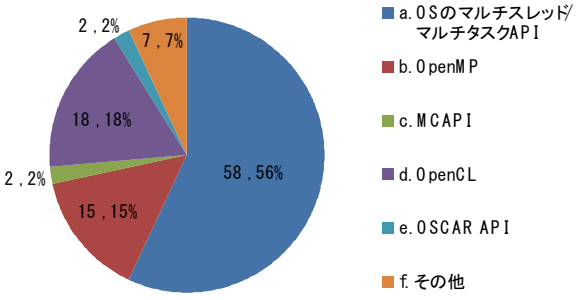
項目4. マルチコアの導入予定はありますか



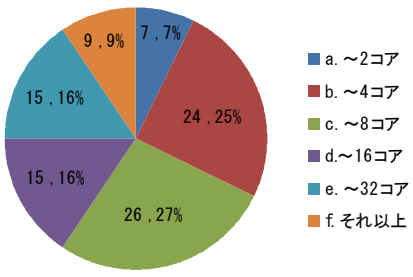
項目5. 使用、または使用予定のOSは何ですか（複数選択可）



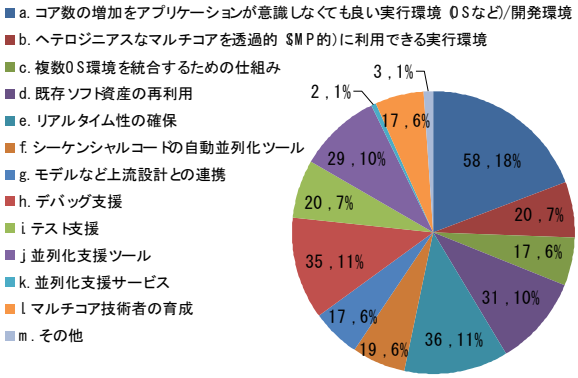
項目6.マルチコアで使用、または使用予定のAPIは何ですか(複数回答)



項目7. 5年後に必要になると思われる最大コア数をお選びください。(複数選択可)



項目8. マルチコアのソフトウェアに関して課題と考えているものは何ですか。(複数選択可)



CEATEC、ET マルチコア アンケート結果総括

- **結果**
  - 期待するものは性能と省電力を合わせると61%、次いでコストパフォーマンスが17%
  - 導入済、予定、検討中で44%、明確に導入予定なしが11%
  - Linuxが最多で45%、次いでTRON系が22%、そしてWindows系が19%
  - AMP:SMP比率はおよそ半々
  - APIは56%がOSのマルチスレッド/タスクAPI、18%がOpenCL、15%がOpenMP
  - 5年後のコア数は8コアが27%と最も多く、ついで4コアが25%、16コアと32コアを併せて32%あり、それ以上も9%
  - ソフトウェア課題に関しては、コア数の増加をアプリが意識しなくてもよい実行/開発環境が18%で最多、次いでリアルタイム性の確保、デバッグが共に11%、そして既存ソフトウェアの再利用と並列化支援がどちらも10%
- **総括**
  - 5年後に必要と思われるコア数は8コア以上が最も多く、iSuppliの調査以上にコア数が求められる傾向が見える。
  - ソフトウェア課題では、コア数増を意識せずに対応できるスケーラビリティがあり、デバッグ・並列化支援も考慮したソフト環境が求められている。

日本の半導体産業の競争力強化に繋がる提言に向けて

- **市場規模、成長性のある有望な適用分野、機器市場**
  - 分野: 医療、農業、交通システム、エネルギー、ロボット
  - アプローチ: サイバーフィジカル、センサーネット、制御と情報処理の融合など
- **海外企業中心の調査結果と考察:日本の強み分野への重点化**
  - 情報系(サーバー、スマートフォン/タブレット、ナビ、基地局など)は2013-15にはマルチコア関連の課題が解決すると考えている
  - 制御系では、課題は完全には解決しないが使っていくという企業も、2013年には課題解決と考えている企業もある
  - 海外企業は情報系を中心に日系企業の2-3年先を行っているので、後追いではなく、組込み分野の、特に日本の強み分野に力を入れるべき
  - この分野で研究開発時の適用対象を選定し、開発/動作環境やAPI、レガシーソフト有効活用といった課題を解決するようなプロジェクトを起こすことを提言する
- **マルチ・メニーコア研究開発を気軽に行うためのツールが手に入らない**
  - マルチコア技術者の裾野を広げるような、手ごろなマルチコアツールキット・教育プログラムの開発・頒布を提言する
  - メニーコア研究を促進するような、手ごろなメニーコアツールキットの開発・頒布を提言する

日本の半導体産業の競争力強化に繋がる提言に向けて

- 開発・動作環境整備、API開発、レガシーソフト活用、マルチコア技術者育成といった課題から、ユーザーはITRS予測より少ないコア数を想定
  - 例えば、コア数増を意識せずに対応できるスケーラビリティがあり、デバッグ・並列化支援も考慮したソフト環境が求められている
  - 制御系は多数の軽い処理の集まりであり、本質的にはマルチ・メニーコアに適していると考えられるが、膨大なレガシーソフトやシングルコアに最適化された設計スタイルやノウハウ、ソフトウェア技術者が移行の障壁
- ヒアリング、アンケートでは、APIはOSのマルチスレッド/タスクAPI、MCAPI、OpenCL、OpenMPなどの活用を想定
  - これらのAPIを活用すると共に、日本の強み分野での活用に向けて、API拡張や新API定義を進め、研究開発・製品化を加速すべき
- 制御系において、情報系の画像認識やモデルシミュレーションを活用する流れがあり、マルチ・メニーコアの活用が望ましい
  - 制御系では信頼性の観点から情報系との分離を志向
  - 情報系と制御系の融合が新たな価値を生む可能性がある

NEDOメニーコア向けOS調査
第1回メニーコアシンポジウム

2012-03-30
権藤正樹
eSOL Co., Ltd.
http://www.esol.co.jp

技術動向：メニーコアチップの傾向

- Many & heterogeneous
  - 性能と消費電力の最適化
  - 同一ISAでも異なる性能（ARMのBig-little）
- Not cache coherent
  - 共有メモリのロックとキャッシュのコヒーレンシー機構によるオーバヘッド
    - x86アーキテクチャのマルチコアとLinuxを用いて計測した結果、物理メモリを獲得するにあたって4コアでは10億CPUサイクル程度だったオーバーヘッドが、16コアになると100億サイクル以上かかることとなり、このオーバーヘッドはコア数が増加するにつれ指数関数的に増えていくことが明らかになっている
- Coherent (but not transparent)
  - 強力なバスでコヒーレンシを維持するものも一部あるが（TILERA）、結局Latencyが異なるため透過的でない
  - ARMのCCI（Cache Coherent Interconnect)も同じ系統
- どの特性もソフトウェアシステムとしての課題を生む
  - 透過的なプログラミング環境の確保
  - スケーラビリティ、ソフトウェアの連続性（再利用性）
  - （ソフトウェアの並列性）、ソフトウェアシステムの機能構成

技術動向：メニーコアOS研究概況

- 大きな二つの流れ
  - スクラッチのアーキテクチャによるOS自身を含めコア分散したメッセージパッシングベースのスケーラブルなメニーコアOS
  - 既存のSMP OSの拡張
- どちらも基本は同じ
  - OS機能の複数コアへの分散
  - 共有メモリは使用しない、またはアクセスを制限
- 純粋なレガシーSMP OS実装の限界
  - 共有メモリの排他制御に使用するセマフォカウンタを分散する技法や、False-Sharingによるキャッシュの非効率を是正する技法で48コアでLinuxを動作させた
  - しかしコア数が増加する度にOSもアプリもチューニングが必要となることから、コア数の増加に追従するスケーラビリティが課題

世界のメニーコアOS研究事例

- Worldwideでは既に複数のメニーコアOSの研究が活発に行われている
- 基本的にはサーバ系が中心、商用はまだ存在しない

名称	研究機関	モデル
Corey	MIT/MS Research	新アーキテクチャ
AKAROS	UC Barkley	Linux拡張
FOS	MIT	新アーキテクチャ
BarrelFish	チューリッヒ工科大学/MS Research	新アーキテクチャ
Helios	MS Research	新アーキテクチャ
Tessellation	UC Barkley	新アーキテクチャ

技術動向：メニーコアOSスケジューリング

- スケジューリング
  - コアローカルでのスケジューリングと、コア間でのスケジューリングの2階層が主流
- 複数スケジューリングポリシーの組合せ
  - スループット重視、リアルタイム性重視、信頼性(時間保護など）重視の3つのタイプが従来あった
  - メニーコア時代には、これら3つの要求を同時に実行するアプリケーションを持つという指摘もあり、複数のスケジューリングポリシーを包含できる仕組みが必要であると提唱されている

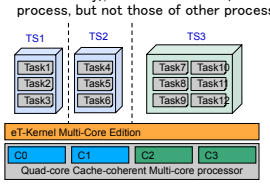
Blended scheduling

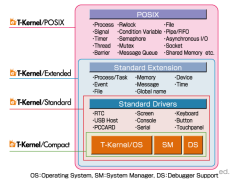
eSOL Announces a Blended Multiprocessing RTOS for the ARM11 MPCore Multiprocessor

New RTOS enables increased levels of real-time control and software flexibility for ARM MPCore multiprocessor designs

TOKYO, JAPAN AND CAMBRIDGE, UK – Sept. 7, 2006 – eSOL Co., Ltd. and ARM (LSE: ARM; Nasdaq: ARMHY), today announced the eT-Kernel Multi-Core Edition real-time operating system (RTOS) coupled with eBinder integrated tools for enhanced multiprocessor solutions, which

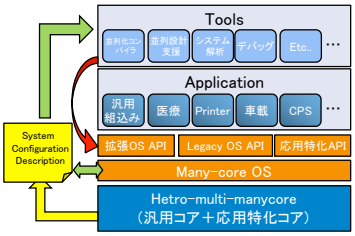
- eSOL eT-Kernel Multi-Core Edition
  - An unique SMP-OS with 4 modes of thread-CPU-affinity scheduling
- Single Processor Mode (SPM)
  - The most CPU-affine scheduling is essentially an AMP mode, with the programmability of SMP transparent API access
  - No thread migration in/out of the core, assuring real-time processing, together with separate kernel data structure and multi-level kernel internal locking mechanism
- The other 3 modes
  - True SMP Mode (normal SMP scheduling), SPM on TSM (normal thread-CPU-affinity), and SRL on TSM (Serialized threads within in a same process, but not those of other processes)





提言：マルチ・メニーコアプロセッサ/ツール/OSフレームワーク

- 海外はまだサーバ系が中心、日本が強い組込みはチャンスがある
- OSを始め、各種開発支援ツール、異なるメニーコアチップを利用できるフレームワークが必要、メニーコアOSはその基盤となる



高いリアルタイム性と、ヘトロなメニーコアを透過的に扱えるメニーコアOS

既存ソフトウェアの再利用性確保

高性能な新規ソフトのための開発環境

安心・安全のための高信頼性

メニーコア向けソフトウェア開発環境調査

第1回メニーコアシンポジウム

キャッツ株式会社
藤本洋

2012年3月30日

メニーコア用ソフトウエア開発の課題

- 動作が複雑で、検証、デバッグが大変
  - ◆データ共有に必要な排他制御機構、デッドロックの回避
  - ◆処理の到達性、安全性、活性、公平性
  - ◆リアルタイム性
  - ◆障害発生時の再現性
- メニーコアアーキテクチャの有効利用
  - ◆十分な並列性抽出とコア割り当て
  - ◆アーキテクチャの理解
  - ◆並列アルゴリズムの利用、開発
- 既存システムの移行
  - ◆既存システム要件を満たした移行
- 利用しやすい開発ツールの提供
  - ◆設計ツール、検証ツールの連携、統合化された環境
- エンジニアの並列設計のKnow-Howの蓄積

技術動向：ソフトウェアの並列構造を作る技術

	並列処理設計（プログラミング）	並列性分析・抽出
有効性	並列処理単位をモジュールとして扱える 時間的に同時に動ける直行する処理をコンパイル単位に分けられる	プログラマは逐次プログラムを書くだけで良い 細粒度の（アーキテクチャ）よりの並列化が可能
問題点	並列設計手法が確立されているとは言えない アーキテクチャに最適化するのは難しい 同期によるデッドロック問題を抱える	自動並列化した「並列構造」が人間にとって理解しやすいとは限らない 各アーキテクチャ毎の並列化コンパイラが無い
支援技術	並列プログラミング言語 Erlang, Go, etc. 並列処理フレームワーク モデルベース設計(MBD)ツール	自動並列化コンパイラ 並列性分析・アドバイスツール OSCAR, CoCo, etc. 参照アーキテクチャ用標準API

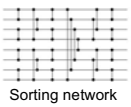
技術動向：安全な動作を保証するための技術

- 設計フェーズでの検証
  - ◆**モデル・シミュレーション**による、設計モデルの妥当性検証
  - ◆**形式手法**による、設計の無矛盾性と網羅的な検証。正当性証明。
- プログラミングモデル
  - ◆**CSP（プロセス代数の一種）**による並行システムの代数的記述。通信チャネルによるプロセス間通信のモデルはいくつかの並列プログラミング言語(Go言語、他)でも採用されている
  - ◆**関数型プログラミング**のもつ特性は並列処理ソフトウェアの開発に向いている。特に変数の**参照透過性**、形式検証との相性の良さがある（言語例：Haskell, Ocaml, Erlang）
- ソフトウェア・トランザクショナルメモリ(STM)
  - ◆**「ロックフリー」**の排他制御機構。ソフトウェアによる実装が既にある。処理性能を上げるため、ハードウェアによるサポートも可能。

技術動向：プログラム設計・再利用技術

- スケルトン並列プログラミング
  - ◆並列処理の典型的なパターンをフレームワーク化し、開発者はブロックを組み立てるように並列処理ソフトウェアを構成できる。（例）Map, Reduce, Scan, Zip, Pipe,Fram

- 並列アルゴリズム
  - ◆メニーコア有効利用：並列アルゴリズムを実装したライブラリを充実させる事でスクラッチから開発する部分を減らせる。アーキテクチャ依存。



Sorting network

技術動向：チップメーカの提供するツール

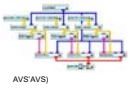
- **チップメーカは自分たちの製品を使ってもらうために使いやすい開発環境を提供している**
  - ◆**Intel Parallel Studio**
  - ◆**NVIDIA CUDA SDK, XMOS Development Tool, etc.**
- **提供されるツール機能（例）**
  - ◆並列化を行うべきコード領域の識別。
  - ◆並列化を実装するためのアドバイス
  - ◆コンパイラ。ライブラリ。デバッガ.
  - ◆パフォーマンス・スケーラビリティ分析
  - ◆ロックと待ち時間の分析
  - ◆メモリーリークとメモリ破壊の検出
  - ◆データ競合とデッドロックの検出
  - ◆シミュレータ

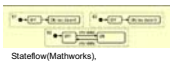
技術動向：MBDツールによる粗粒度並列モデリング

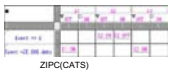
- データフロー型
  - ◆操作と操作の間でのデータの流れを有向グラフとしてプログラムを作成する。
  - ◆データの流れにそった形で並列が自然に表現される
- 状態遷移型
  - ◆複数の状態遷移系による通信ネットワーク
  - ◆階層型の状態遷移モデルの「並列状態」

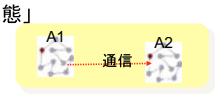

Simulink(Mathworks)


LabView(NI)


AVS'AVS)

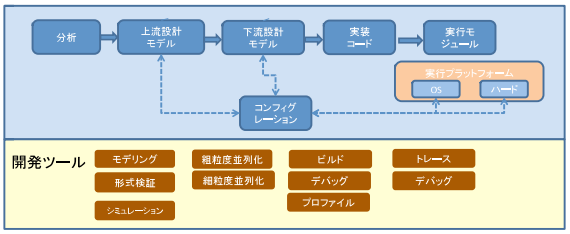

Stateflow(Mathworks),


ZIPC(CATS)



マルチコア対応アプリ設計支援環境の検討

- MBDの開発プロセスへの並列化支援ツールの組み込み。
- 異なるツール間をつなぐツールチェーンの構築、標準的なAPI
- 最適な構成を得るためのコンフィグレーション情報の共有
- 設計フェーズにモデルの検証を入れる事で開発の**フロントローディングを実現**



資料：学会調査

- **FM2011**(International Symposium on Formal Methods)
- IEEEの形式手法の国際学会
- 形式手法では初期の段階から並列・分散システムを研究対象としているため、ここまでに蓄積されてきたケーススタディーが役に立つ
- 研究テーマ動向
  - ◆ 連続系と離散系の統合モデルとその検証
  - ◆ プログラムソースの静的解析への応用
  - ◆ CPS(Cyber Physical System)も対象領域としてホットトピックとして議論されている(マルチ・メニーコアの応用分野としても有力視されている)

- FM2011 rogram
  - ◆ Cyber Physical Systems
  - ◆ Runtime Analysis
  - ◆ Case Studies / Tools
  - ◆ Experience Report
  - ◆ Program Compilation and Transformation
  - ◆ Security
  - ◆ Process Algebra
  - ◆ Education
  - ◆ Concurrency
  - ◆ Dynamic Structures
  - ◆ Model Checking

- **ICPADS2011**(International Conference on Parallel and Distributed System)
- IEEEの並列・分散処理システムに関するカンファレンス
- 研究テーマの動向
  - ◆ マルチコアに関する研究発表としては、エミュレータの高速化、アーキテクチャの効率化、GPGPUの活用に関するものが目立った。
  - ◆ CPS(Cyber-Physical System)はマルチコアの有効適用分野である。リアルタイム・センシング、車載システムとネットワーク、ワイヤレス・ネットワークに関するもの、また、CPSに関わる入力系のワークフローのモデリングなどの発表があった。

- ICPAD2011 Program
  - ◆ Cluster, Grid, Cloud Computing and Services
  - ◆ Parallel Algorithms and Applications
  - ◆ Multicore Computing and Parallel/Distributed Architecture
  - ◆ Mobile Computing
  - ◆ P2P Computing
  - ◆ Security and Trustworthy Computing
  - ◆ Cyber-Physical Systems and Internet of Things

- **POPL2012**(ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages)
- ACMのプログラミング言語に関するカンファレンス
- 研究テーマ動向
  - ◆ 検証(Verification)、証明(Proof)に関するものが目立った。特にSeparation Logicを扱ったもの、また、定理証明系をつかったもの。すなわち、研究のツールとして形式手法を利用している。
  - ◆ 関数型プログラミングの実用的な応用としての「組み込みDSL」
  - ◆ 目立って利用されている言語はHaskell, Ocaml, Scala。
  - ◆ 並列プログラミング言語としては宣言的な言語(関数型、論理型)が採用されている
  - ◆ C, C++のプログラムの意味論に関しても1セッション設けられている。

- POPL 2012 Program
  - ◆ 1.A **Verification** / 1.B Semantics
  - ◆ 2.A **Privacy and Access Control**/ 2.B Decision Procedures
  - ◆ 3.A Security/ 3.B **Complexity for Concurrency**
  - ◆ 4.**Medley** / 4.B Mechanized Proofs
  - ◆ 5.A **Concurrency** / 5.B Type Theory
  - ◆ 6.A Dynamic Analysis / 6.B Names and Binders
  - ◆ 7.A. **Verified Transformations** /B. Functional Programming
  - ◆ 8.A **C/C++ Semantics** / 8.B Type System

資料：企業ヒアリング結果

4社：A(自動車),B(FA),C(FA),D(X)

- マルチ・メニーコア導入の主な理由
  - ◆ 複数のCPUで実現している機能をまとめ、コストを下げたい(A)
  - ◆ 性能を上げたい(A) (B)(D)
  - ◆ 消費電力を抑える必要がある(A)
  - ◆ トータルのコスト削減を考えている(C)
- 対象アプリケーションの特性は
  - ◆ リアルタイム制御(A)(D)
  - ◆ 高速描画処理(D)
  - ◆ 特定のアプリを特定していない(C)
  - ◆ 画像処理：GA、GPの高速化への適応(B)
  - ◆ 画像認識：3次元姿勢推定の高速化(B)
  - ◆ 高速描画処理: 3D表示の高速化(B)
- 設計モデルを作成管理するのに使っているツールは
  - ◆ Simulink (A)
  - ◆ 内製ツール(D)
  - ◆ Visio(お絵かき程度) (B) (C)
  - ◆ 紙と鉛筆(B)
- 設計モデルの記述に利用している表記は
  - ◆ Simulink (A)
  - ◆ フローチャート(A) (B)(D)
  - ◆ 構造化チャート(A) (B)
  - ◆ UML(クラス図、状態遷移図、シーケンス図) (B) (C)
  - ◆ 状態遷移表(B)

- マルチ・メニーコア対応への懸念点
  - ◆ 並列処理設計が難しい(A) (B)
  - ◆ 動作保障、検証が難しい(A)(D)
    - □ 特に、タイミング、再現性
  - ◆ リアルタイム性の保障が難しい(A)
  - ◆ 標準化が遅れている(C)
  - ◆ リアルタイム性の保証が難しい(C)
  - ◆ バス(内部・外部)帯域幅の不足やアクセス競合による性能低下(C)
  - ◆ CPUの利用効率、負荷バランスの悪化(C)
  - ◆ 対応したリアルタイムOSが少ない(C)
  - ◆ リアルタイムOS上での開発をサポートしたツールが少ない(C)
- 開発環境への期待
  - ◆ 並列設計をサポートするツールが欲しい(A) (B)
  - ◆ 検証ツールが欲しい(A) (B)
  - ◆ 分散化された機能を適切にコアに配置する機能をサポートするツール(D)
  - ◆ 既存システムのマルチコア実行環境への移行をサポート(C)
  - ◆ マルチコア化した実際のソフトに対する性能・効率改善のためのアドバイスを提供(C)
  - ◆ ばらばらではなくて統合化されたもの(C)
- その他
  - ◆ 組み込み向けではなく、PC用のプロセッサを利用している(D)

SMYLEプロジェクトの紹介と メニーコア・プロセッサSMYLErefの開発

九州大学　井上こうじ
立命館大学　冨山宏之
電気通信大学　近藤正章

講演内容

- SMYLEプロジェクト
- メニーコアSoCプラットフォーム SMYLEref
- 研究開発状況

講演内容

- SMYLEプロジェクト
- メニーコアSoCプラットフォーム SMYLEref
- 研究開発状況

SMYLEプロジェクト

～Scalable ManY-core for Low-Energy computing～

- NEDOグリーンITプロジェクト「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」
  - 研究期間は2010年12月～2013年2月
  - 産学連携による研究開発
  - 成果は可能な限り「Open」に!

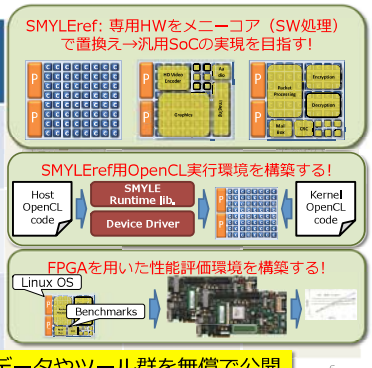
KYUSHU UNIVERSITY　RITSUMEIKAN　UEC TOKYO　TAT 東京農工大学
eSOL　JEITA　TOPS　FIXSTARS　CATS

分担とゴール

	九大＋立命館＋電通大＋農工大	トプスシステムズ	フィックスターズ	JEITA+eSOL+CATS
どのようなメニーコアを作れば良いのか?	メニーコアSoCとその実行環境	ビデオマイニング向けメニーコア		市場調査
どのようにソフトウェアを開発すれば良いのか?		ビデオマイニング用ソフトウェア	プログラム開発環境	動作環境／開発環境調査
どのようにメニーコア研究開発を進めれば良いのか?	FPGA評価環境や機能シミュレータ		アクセラレータ用ベンチマーク	プロジェクト提案

分担とゴール

	九大＋立命館＋電通大＋農工大
どのようなメニーコアを作れば良いのか?	メニーコアSoCとその実行環境
どのようにソフトウェアを開発すれば良いのか?	
どのようにメニーコア研究開発を進めれば良いのか?	FPGA評価環境や機能シミュレータ

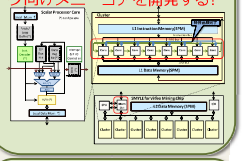


(可能な限り)設計データやツール群を無償で公開

分担とゴール

	九大＋立命館＋電通大＋農工大	トプスシステムズ	フィックスターズ	JEITA+eSOL+CATS
どのようなメニーコアを作れば良いのか?	メニーコアSoCとその実行環境	ビデオマイニング向けメニーコア		市場調査
どのようにソフトウェアを開発すれば良いのか?		ビデオマイニング用ソフトウェア	プログラム開発環境	動作環境／開発環境調査
どのようにメニーコア研究開発を進めれば良いのか?	FPGA評価環境や機能シミュレータ		アクセラレータ用ベンチマーク	プロジェクト提案

SMYLEvideo: ビデオマイニング向けメニーコアを開発する!



評価ボード開発：SHIFTのSWリアルタイム処理を実現する!

評価ボードデモを開発しIPコアビジネスへと展開

分担とゴール

	九大＋立命館＋電通大＋農工大	トプスシステムズ	フィックスターズ	JEITA+eSOL+CATS
どのようなメニーコアを作れば良いのか?	メニーコアSoCとその実行環境	ビデオマイニング向けメニーコア		市場調査
どのようにソフトウェアを開発すれば良いのか?		ビデオマイニング用ソフトウェア	プログラム開発環境	動作環境／開発環境調査
どのようにメニーコア研究開発を進めれば良いのか?	FPGA評価環境や機能シミュレータ		アクセラレータ用ベンチマーク	プロジェクト提案

CLtrump: 並列化コード作成支援ソフトウェアを開発する!

PEMAP: 移植後の性能見積もり支援ソフトウェアを開発する!

FIXSTARS OpenCL Benchmark（4分野8種）を開発

ベンチマークやツール群を無償で公開

分担とゴール

	九大＋立命館＋電通大＋農工	JEITA+eSOL+CATS
どのようなメニーコアを作れば良いのか?	メニーコアSoC	市場調査
どのようにソフトウェアを開発すれば良いのか?		動作環境／開発環境調査
どのようにメニーコア研究開発を進めれば良いのか?	FPGA評価環境	プロジェクト提案

マルチ／メニーコア市場を予測する!

メニーコアに関する動作環境やシステム開発環境を調査する!

日本の半導体産業の競争力強化に繋がる提言をまとめる!

調査結果や検討結果を公開

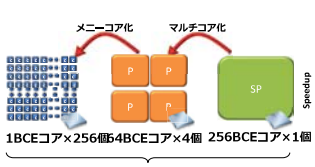
講演内容

- SMYLEプロジェクト
- メニーコアSoCプラットフォーム SMYLEref
- 研究開発状況

メニーコアの本質とは?

低性能，小面積，低消費電力なコアを大量に搭載

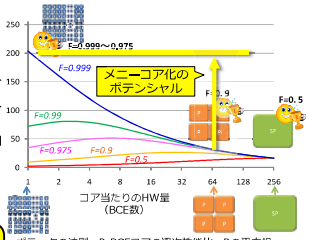
オンチップ並列処理により劇的な性能向上を実現



1BCEコア×256個 64BCEコア×4個 256BCEコア×1個

ハードウェア量は同じ

1BCE（Base Core Equivalent）は最小コア「c」のHW量



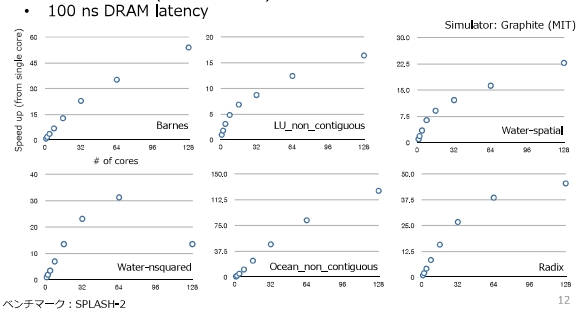
「並列処理」を手段として大量トランジスタを徹底活用!

・ポラックの法則：R-BCEコアの逐次性能比＝Rの平方根
・メモリによる影響等は無視
・アムダールの法則に基づく性能見積り（Hill, HPCA'08）

単一アプリ性能はスケールするのか?

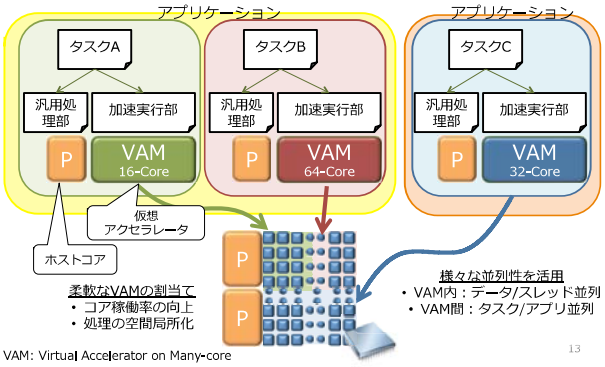
～科学技術計算アプリでさえも・・・～

- In-Order Core@1GHz (1～128コア)　w/ private 32KB L1 & 512KB L2
- 2D Mesh NoC (no contention)
- 100 ns DRAM latency



ベンチマーク：SPLASH-2

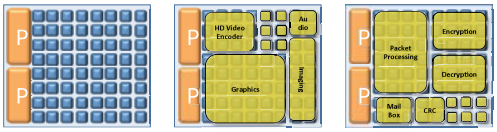
SMYLE基本コンセプト
～柔軟な仮想アクセラレータ実行プラットフォーム～



VAM: Virtual Accelerator on Many-core

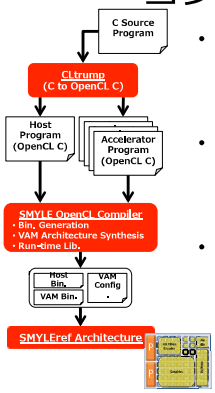
SMYLErefの狙い
～マルチスレッド・マルチタスク(アプリ)実行でスケーラビリティを確保～

- 汎用性重視型メニーコア
  - マルチコア性能<<メニーコア性能<専用エンジン性能
  - 様々なアプリケーションを「比較的」効率良く実行
  - SoCに搭載された複数専用エンジンをメニーコア(SW処理)で置換え



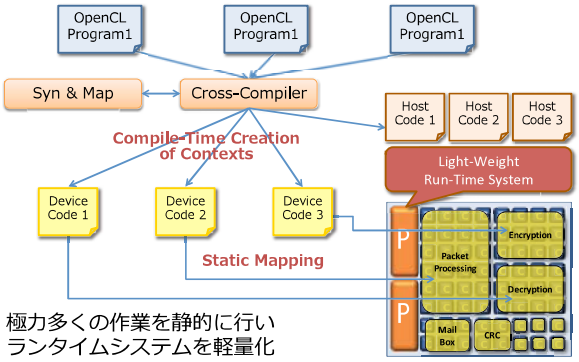
•メニーコア・ドメインはホモジニアス構成
•再構成可能性によりヘテロジニティを実現（例：メモリ構成）
•データ/スレッド並列処理＋タスク並列処理＋アプリ並列処理

コンパイルフロー



- CLtrumpの利用
  - CプログラムをOpenCLに変換
  - ユーザ・インタラクティブな並列化支援
- OpenCLのサポート
  - ヘテロジニアスアーキテクチャ
  - プラットフォーム非依存
- コンパイル時にVAMの構成とマッピングを決定
  - コア数
  - メモリ構成

SoC開発フロー

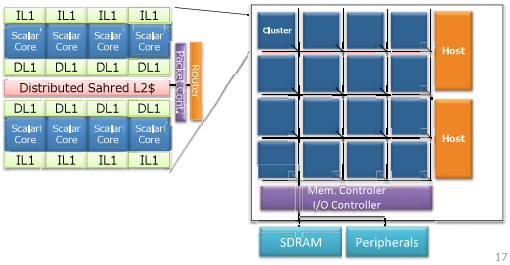


極力多くの作業を静的に行いランタイムシステムを軽量化

複数タスクを空間並列実行

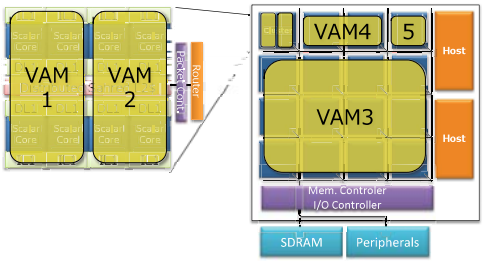
SMYLErefアーキテクチャ

- 共有メモリ・モデル
- シンプルなスカラコアで構成されるクラスタ構造＋NoC接続
- プライベートL1$＋分散共有L2$
  - ハードウェアによるコヒーレンス制御無し



SMYLErefアーキテクチャ

- 共有メモリ・モデル
- シンプルなスカラコアで構成されるクラスタ構造
- プライベートL1$＋分散共有L2$
  - ハードウェアによるコヒーレンス制御無し



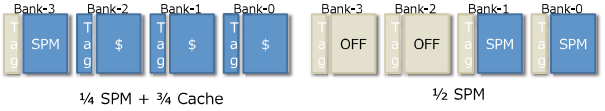
特徴その1：クラスタ構造

- より多くのトランジスタを（通信でなく）演算に使う!

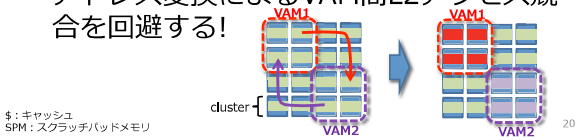
- クラスタ内に閉じた世界での低レイテンシ・コア間通信を実現する!

特徴その2：再構成可能メモリ

- VAMごとのL1$／SPM選択を可能にする!

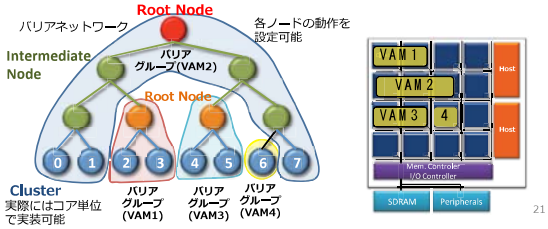


- アドレス変換によるVAM間L2アクセス競合を回避する!

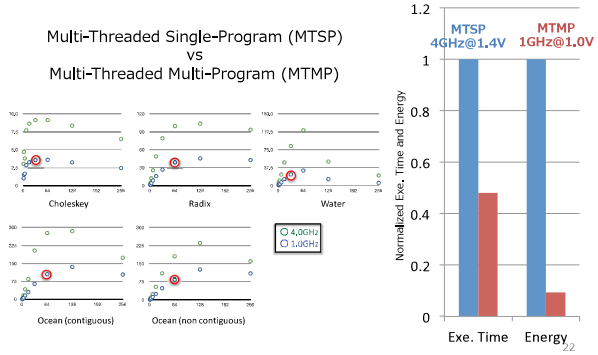


特徴その3：柔軟なHWバリア

- VAMに所属するコアのみを対象とした高速HWバリアを実現する!
- 複数VAMによるHWバリアの並列実行を可能にする!



マルチスレッド・マルチプログラム実行方式の効果見積もり



講演内容

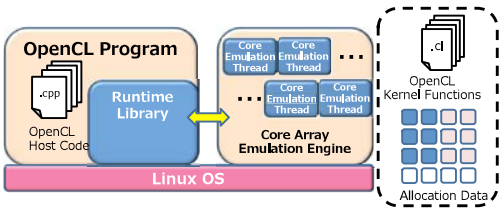
- SMYLEプロジェクト
- メニーコアSoCプラットフォームSMYLEref
- 研究開発状況

機能検証用シミュレータの開発

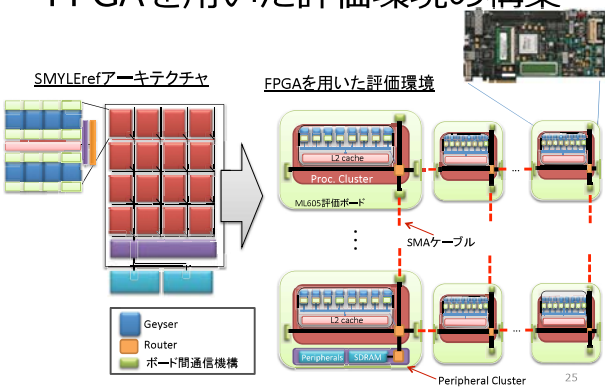
◆ホスト側／メニーコア側ともに同一の実機マシン上で動作

◆１つのコアの動作を1つのスレッドが実行

◆各コアにタスクを静的割り当て



FPGAを用いた評価環境の構築



FPGAを用いた評価環境の構築

なお，本プロトタイプでは，JST CRESTプロジェクト「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（代表：東大 中村宏教授）」の成果を利用しています。

おわりに

- SMYLEプロジェクト
  - 成果は可能な限り「Open」に!
- メニーコアSoCプラットフォーム：SMYLEref
  - アーキテクチャとプロトタイプ
  - その他にも各種の要素技術を開発中
    - VAM（アーキテクチャ）合成&マッピング技術
    - 稼働コア数動的最適化技術
    - 低電圧実行技術
- H24年度の展開
  - 産業界からのフィードバックを反映
  - 実行環境構築と8種のベンチマーク動作確認
  - より詳細な性能／消費エネルギー評価

FIXSTARS
Speed up your Business

メニーコア時代のソフトウェア開発環境

株式会社フィックスターズ
取締役 COO 田村　陽介

株式会社フィックスターズ

事業内容	マルチコアプロセッサ向け ソフトウェア開発事業
設立	2002年8月
資本金	2億1,055万円
社員数	86名（2012/4月 現在）
所在地	東京、カリフォルニア
代表取締役	代表取締役会長 - 長谷川 智彦 代表取締役社長 CEO - 三木 聡
顧問	経営戦略顧問 -マイケル A. クスマノ

FIXSTARS
Speed up your Business

本社：大崎ゲートシティ18F

ソフトウェア開発実績

- 2002年8月 ITベンチャとして設立
- 2004年 東芝様からの依頼でCell/B.E.関連の開発に関わる
  - ✓ ソニー様, 東芝様, IBM様との関係が強まる
  - ✓ 2007年9月 PLAYSTATION 3発売, PS3 Information Site開設
- 2008年頃～ Cell/B.E.を採用するユーザ様とのお取引拡大
  - ✓ 主に金融、計算科学、ヘルスケア、デジタルメディア、産業機器の分野
- 2009年頃～ マルチコアによるソフトウェアの高速化
  - ✓ Cell/B.E. からマルチコアプロセッサ(GPGPU, 組込み系プロセッサや 汎用プロセッサ)向けソフトウェア開発に事業領域を広げる

NEDOプロジェクトでの活動

- NEDOプロジェクト
  「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」
  - ✓ メニーコア向けソフトウェア開発環境を担当
- 本プロジェクトでのアウトプット
  - ✓ 本プロジェクトでの研究開発チップだけを対象にするのではなく、ソフトウェア開発ツールとして汎用性の高いものを開発して公開予定
  - ✓ 特に弊社のこれまでの産業界での経験を活かしメニーコア時代に要求されるであろうソフトウェア開発環境を探求している
- 本日の話の内容
  - ✓ メニーコア時代のソフトウェア開発環境とは何か？
  - ✓ メニーコア時代にエンジニアに新たに要求されることは何か？
    - ■ ソフトウェア開発環境はあくまでエンジニアを支援する環境
  - ✓ 時代背景やソフトウェア業界の状況を含めながら紹介する

時代背景① 並列処理を強いられるソフトウェア

- ご存知の「The Free Lunch Is Over」
  2005年Microsoftの Software ArchitectであるHerb Sutter氏の言葉

Free Lunch is over とは？
これまでCPUはシングルコアの性能を高める方向で進化したためソフトウェアは何もしなくてもCPUの進化の恩恵を受けていた。しかし、CPUベンダがマルチコアに方向転換したため逐次プログラムの性能は伸びずに、並列プログラムの性能が上がっていく。ソフトウェア開発者はCPUの性能を活かすためにソフトウェアを根底から見直さなければならない。

時代背景② 応用指向プロセッサの活用

- 「応用指向プロセッサ」の利用が積極的に検討されはじめた
  - ✓ 特定の処理を想定しているためコアをシンプルに小さくすることができる。そのため、処理によっては飛躍的に性能を向上させることが可能
- 汎用用途に歩み寄るGraphics Processing Unit (GPU)
  - ✓ ソフトウェア開発環境も整備されてきており、これまでのように特殊な知識がなくてもプログラミングが可能となった
- ヘテロジニアスマルチコアの存在
  - ✓ Sony/Toshiba/IBMで開発された Cell/B.E.に続き、AMD社のFusionといった1チップ内に汎用プロセッサと応用指向プロセッサを搭載したプロセッサが製品化

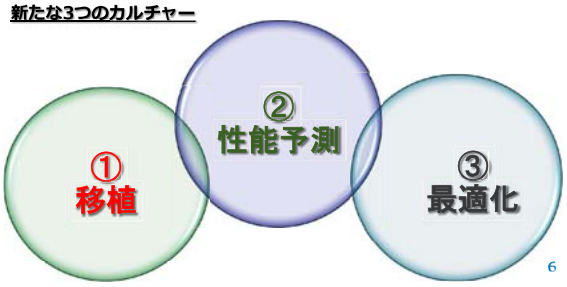
NVIDIA
CUDA

OpenCL

ソフトウェア業界に根付く新たなカルチャー

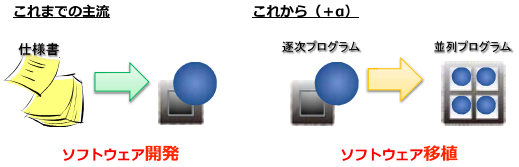
- 時代の変化によりソフトウェア業界に新たなカルチャーが根付いた
- 特に「汎用プロセッサのマルチコア化」と「応用指向プロセッサの活用」はソフトウェアエンジニアに新たな技術の習得を強いることになる
- 今後のソフトウェア環境はこれらの技術を支援することが求められる

新たな3つのカルチャー



①「移植」カルチャー

- シングルコアが性能年々向上していた「Free Lunch 」の時代にはプロセッサの変更に伴うソフトウェアの変更は必要なかったため、ソフトウェア開発といえば**仕様書に基づいた新規開発**が主であった
- マルチコアの時代には、新規開発の他に**既存ソフトウェア資産の移植**という作業が発生する。
- 新規開発においても、リファレンスコードとなる逐次プログラムを最初に開発してから、そのプログラムを**並列プログラムに移植開発**するといった開発方法が一般的となっている。



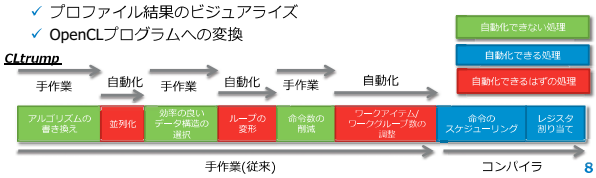
開発ツール①：CLtrump
C to OpenCL Translator Utilities for Many core Processor

概要

- ✓ 並列プログラムの開発作業において、自動化可能な部分と手動化が必要な部分が混在している。CLtrumpでは、自動化可能な部分はツールで自動化し、手動化が必要な部分はユーザが自ら開発するようなインタラクティブな並列プログラム開発環境を提供する。

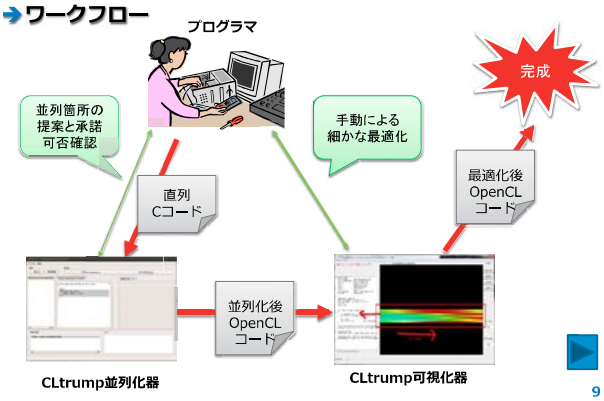
特徴

- ✓ 楽観的な並列性検出器
- ✓ PythonでC99のトランスレータを書くためのインタフェース
- ✓ 実行時プロファイル情報を活用した並列性の抽出
- ✓ 実行時間を考慮した並列化
- ✓ プロファイル結果のビジュアライズ
- ✓ OpenCLプログラムへの変換



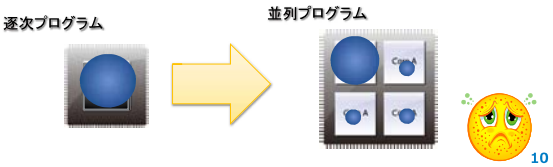
開発ツール①：CLtrump
C to OpenCL Translator Utilities for Many core Processor

ワークフロー



②「性能予測」カルチャー

- ソフトウェア移植の最大の魅力は性能向上である。そのため、既存のソフトウェアを移植する前に**ある程度の性能予測**を要求されることが多い。
- 産業界では最終的な性能がある程度見込まれてから予算が確保できるケースが多いため**なるべく低コストでの見積もり作業**が求められる。
- 見積もり作業には詳細なコード分析が必要となり時間がかかる。体系的な見積もり手法が確立されていないため**見積るエンジニアによって精度もバラバラ**であり信憑性も薄いことが多い。



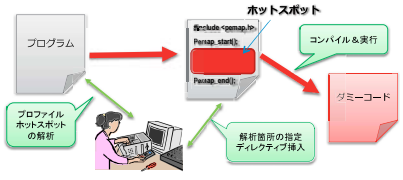
開発ツール②：PEMAP
Performance Estimator for Many core Processors

概要

- ✓ 本開発ツールでは、アーキテクチャの異なるプロセッサ間の移植において、移植前と移植後の命令セットの頻度分布やメモリアクセスパターンに何らかの関連性があることに着目し、移植前のプログラムから、その特性を維持した移植後のダミーコードを自動生成する。ダミーコードは計算結果は異なるが計算量が反映されているため性能の見積もりが可能となる。

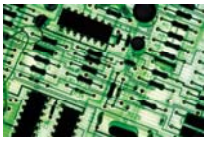
特徴

- ✓ 実行可能なダミーコードを自動生成
- ✓ 実験に基づくデータにより命令セットの特性の違いを反映
- ✓ 統計手法を用いたメモリアクセスパターンの特性抽出と反映



③「最適化」カルチャー

- ハードウェアに最適なプログラムをソフトウェア屋さんが意識する時代
  - ✓ Cell/B.E.の登場や、GPUを使ったプログラムはその典型例
  - ✓ 最近のプロセッサの中にはハードウェアを意識しないでプログラムを書くと性能がほとんど出ないようなものも存在する
- 高品質なソフトウェアを作るためには、プロセッサが提供しているマイクロアーキテクチャの機能をうまく活用することが求められる
  - ✓ ハードウェアだけで実現可能なマイクロアーキテクチャ技術が限界
  - ✓ ソフトウェアとの連携を想定したマイクロアーキテクチャが多く登場
  - ✓ メニーコア化のためにコアのリッチな機能を削減しているプロセッサもある。



ベンチマークの公開③：BMAP
Benchmark for Automatic Parallelization

- 概要
  - ✓ 弊社でこれまで蓄積してきた最適化技術のノウハウを共有する目的でオープンソースとして公開。高速処理のためのソフトウェアの最適化が強く要求される4分野(ライフイノベーション、インターネットアプリケーション、ファイナス、ビジュアルコンピューティング)8種類のコードを公開。
- 公開内容
  1. C言語で記述されたリファレンスプログラム
  2. OpenCLで記述された最適化プログラム
     - ■ 引数の指定により最適化レベルを変更可能
  3. 性能結果や最適化の詳細が記載された技術レポート

公開予定のコード
・ブラック・ショールズ
・モンテカルロシミュレーション
・リニアサーチ
・ラレレングス法
・グレースケール
・カウシアンフィルタ
・バックプロジェクション

SOURCEFORGE

まとめ

- 本プロジェクトではメニーコア時代におけるソフトウェア開発環境の研究開発をしています。
- 弊社のこれまでの経験に基づきソフトウェア開発の現場において起こっている新たな潮流を「移植」「性能見積もり」「最適化」という3カルチャーとして紹介しました。
- 紹介した3カルチャーに対してエンジニアを支援するために本プロジェクトでどのようなアプローチをしているのかを紹介しました。

ビデオマイニング向けメニーコアの開発

TOPS

2012年 3月 30日(金)

株式会社トプスシステムズ

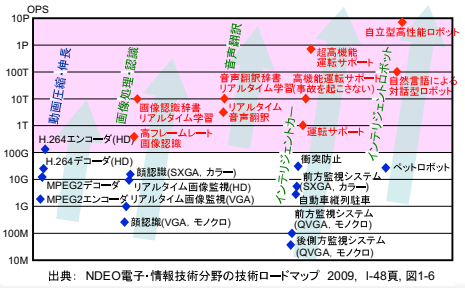
代表取締役
博士(情報科学)　松本　祐教

ビデオ・マイニング向けメニーコアの開発

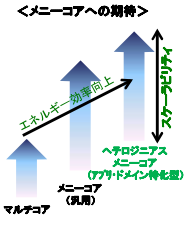
- なぜ、ビデオ・マイニングか
- ビデオ・マイニング処理の要件
- ビデオ・マイニング処理の分析
- ビデオ・マイニングに適した処理方式
- ビデオ・マイニング向けソフトウェア
- SMYLEvideo : ビデオ・マイニング向けメニーコア・アーキテクチャ
  - コア・レベルのアーキテクチャ
  - クラスタ・レベルのアーキテクチャ
  - コア間の通信・同期方式
    - “ゼロ・オーバーヘッド・メッセージ・パッシング”
- ビデオ・マイニング向けメニーコアの開発(まとめ)

次世代アプリケーションとプロセッサ

- **次世代組込みシステム:要求処理能力は、なお止まるところを知らない(数ウ100GOPS~数POPS)**
- **消費電力:　処理に使える電力は増やせない(~数W程度)**



出典:　NDEO電子・情報技術分野の技術ロードマップ　2009,　I-48頁, 図1-6



- 数GOPS/W ⇒ 数100GOPS/W
- 汎用　⇒　ASSP
- 高いスケーラビリティ

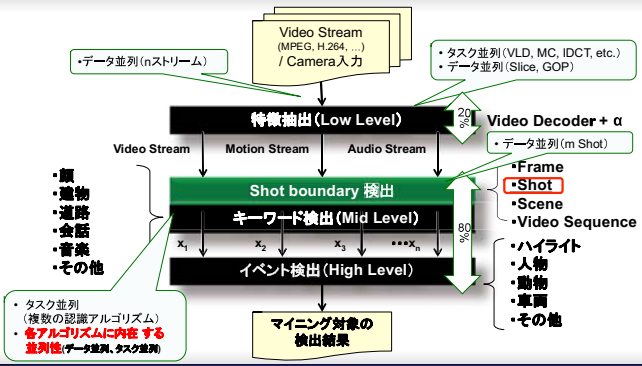
ビデオ・マイニングとは何か？

- ビデオの分析
  - ハイライト・シーンの抽出(スポーツ番組など)
  - 特定のシーンの抽出(山の景色、海の景色、屋内、屋外など)
  - 特定のイベントの検出(ポーズ、ジェスチャー、表情)
- オブジェクト検索 &トラッキング
  - 特定の人物を見つけ、動きを追跡する　「ニュース、サッカー、野球、マラソン、・・・」
  - 動物:犬、猫、馬、牛、ネズミ、・・・
  - 移動体:乗用車、トラック、自転車、・・・
  - 建造物:ビル、家屋、橋、トンネル、・・
- ビデオの編集
  - 分析・検索・トラッキング結果を編集して出力

なぜ、ビデオ・マイニングか？

- 処理全体の99%以上が並列化可能
- 要求処理能力が高く、既存のプロセッサでは実現困難
  - 人間の能力により近い
  - 実世界のモデリング &シミュレーション
- 複数のタイプの異なるアルゴリズムを使い分け:ハード化が困難
  - Video Decoder ; MPEG2, MPEG4, H.264, ・・・
  - SIFT, Optical Flow, Ransac, ハール特徴、モデル・ベース顔認識、・・・
- コア技術の応用展開
  - 次世代組込みシステムのカメラ/ビデオ入力のインテリジェント化による付加価値向上
  - 機能要件・性能要件に合わせてスケーラブルにSMYLE videoメニーコアIPを構成して提供
    - 情報家電:　スマートフォン、タブレット、次世代テレビ、STB、デジカメ、ビデオカメラ
    - 自動車:　車載カメラ(車両周辺認識)、インテリジェント・カー(自動運転)
    - その他:　ロボット、監視装置(ビル/ホーム・セキュリティ、鉄道/道路/その他)

ビデオ・マイニング・システムの概要



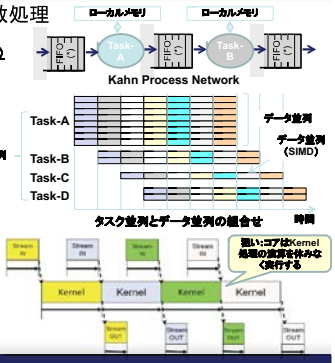
Video Mining処理に内在する並列性

アプリ処理	目的	アルゴリズム	並列性
ビデオ分析	動作の方向と速度の予測	オプティカルフロー	行方向の並列性
	特定の特徴の検出と抽出	SIFT*1	分割データの並列性
	人物の検出・動作の追跡	カスケード型ハール特徴	ピクセル・レベルの並列性
	領域を区別するライン検出	ハフ変換	行方向の並列性
	連続画像を用いて誤検出を排除	Ransac*2	サンプル・データの並列性
人物検索	様々な角度からの顔検出	ベクター型顔検出	画像ブロックの並列性
	顔の特徴を抽出	モデルベース顔認識	タスク・レベルの並列性
	特定の特徴の検出と抽出	SIFT	分割データの並列性
ビデオ編集	セグメントの抽出	グラフベースセグメント分割	グリッド・レベルの並列性
	動きベクトルの検出	ブロック・マッチング	行方向の並列性

*1 SIFT：Scale-Invariant Feature Transform
*2 Ransac：Random Sample Consensus

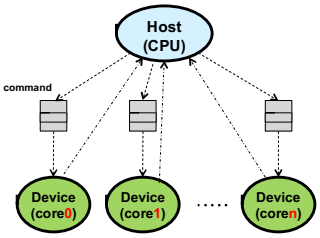
ビデオ・マイニング・アプリに適した処理方式

- Kahn Process Network型の分散処理
  - 複数のメモリ非共有プロセス
  - プロセス間の通信（ゼロオーバーヘッドFIFOアクセス機能）
- 2つの並列処理の組合せ
  - 分散並列処理（タスク、パイプライン並列）
  - データ並列処理（上位レベル、命令レベル）
- ストリーム処理(Core)
  - Kernel(プロセス内の演算処理)
  - Stream-In(メモリ・アクセス)
  - Stream-Out(メモリ・アクセス)
- Coreの最適化
  - ストリーム処理のサポート：Stream-In/Outの隠蔽
  - 複合命令：Kernelのサイクル数削減
  - FIFOのサポート
  - 命令供給＆データ供給エネルギーの削減

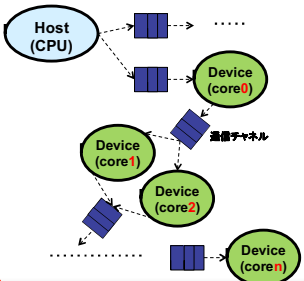


OpenCLと分散処理

- OpenCL（集中制御型）
  - Masterの処理がボトルネックに
  - 分散処理の表現が困難
  - 負荷分散型のキューを構成できない
    （Queueとデバイスは、1対1のみ）

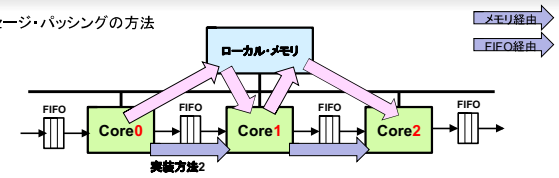


- 分散処理型
  - 高いスケーラビリティ
  - 負荷分散も可能



プロセッサ・コア間FIFOの効果

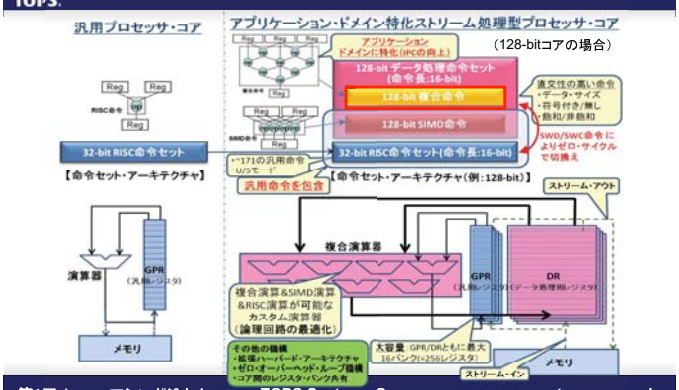
- メッセージ・パッシングの方法



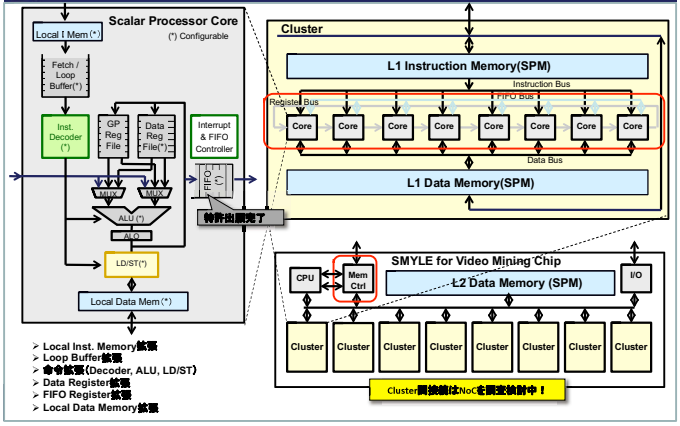
分散並列処理型のアプリケーション実装例	ローカル・メモリへのアクセス量		備考
	メモリ経由	FIFO経由	
JPEGデコーダ	1.15MB	0.69MB(60%)	プロセス数=6, R/W= 10画@QVGA
JPEGエンコーダ	1.38MB	0.92MB(66%)	プロセス数=6, R/W= 10画@QVGA
H.264デコーダ Full-HD @ 30fps	1.44GB/s	1.06GB/s(73%)	プロセス数=10, R/W= 15画@Full-HD(420)
Ray Tracing Full-HD @ 30fps		(約50%)	プロセス数=64

消費電力削減効果：30%程度以上
（メモリの消費電力＝全体の80%、アクセス量削減効果＝60%程度とした場合）

プロセッサ・コアのアーキテクチャ



SMYLE Videoの構成



ビデオ・マイニング向けメニーコアの開発

- ビデオ・マイニング・アプリケーション
  - 99%以上が並列化可能
  - 様々なアルゴリズムの要求するヘテロジニアスなワークロード
  - スマートフォンから自動運転まで幅広い応用範囲
- ビデオ・マイニングに適した処理
  - 分散並列処理をKahn Process Networkでモデル化し、ソフトウェア開発を容易に
  - データ並列とタスク並列を組合せた非同期パイプラインによる分散並列処理
  - 明示的なメモリアクセスのスケジューリングに基づくストリーム処理
- SMYLEvideo：ビデオ・マイニング向けメニーコア・アーキテクチャ
  - 複合命令を有するヘテロジニアスなストリーム・プロセッシング・コア
  - スケーラブルなクラスタ構成
  - 分散並列処理効率を向上するゼロ・オーバーヘッド・メッセージ・パッシング機構

『平成24年度　評価システム(FPGA)上でのデモに向けてSMYLEvideo開発中』

Go Next Generation

TOPS

第1回メニーコアシンポジウム　TOPS Systems Corp.　www.topscom.co.jp

第2回メニーコアシンポジウム
2013年1月30日

主催：NEDO　独立行政法人新エネルギー・産業技術総合開発機構
共催：早稲田大学 グリーンコンピューティングシステム
　　研究開発センター
協賛：電子情報通信学会 集積回路研究専門委員会
　　情報処理学会 計算機アーキテクチャ研究会

SMYLEプロジェクト

～Scalable ManY-core for Low-Energy computing～

- NEDOグリーンITプロジェクト「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」
  - 研究期間は2010年12月～2013年2月
  - 産学連携による研究開発
  - 成果は可能な限り「Open」に!

KYUSHU UNIVERSITY　RITSUMEIKAN　UEC　TAT 東京農工大学
eSOL　JEITA　TOPS　FIXSTARS　CATS

本研究の成果

	九州大学，立命館大学，電気通信大学，農工大，フィックスターズ	トプスシステムズ，九州大学，電気通信大学	フィックスターズ，立命館大学
ハードウェア	***SMYLE ref*** 汎用メニーコアSoCアーキテクチャ	***SMYLEvideo*** ビデオマイニング向けメニーコア	
ソフトウェア	***SMYLE Open CL*** メニーコアSoC向けプログラミング	***SMYLEvideo*** アルゴリズム（HWとのコデザイン）	***CLtrump*** C to OpenCL変換 ***PEMAP*** アプリ移植時の性能推定
実装評価	***SMYLE fpga*** 複数FPGAを用いた実行環境	***FPGA*** デモ用ボード	***BEMAP*** メニーコア評価用OpenCLベンチマーク

3

成果報告ダイジェスト

	九州大学，立命館大学，電気通信大学，農工大，フィックスターズ
ハードウェア	***SMYLE ref*** 汎用メニーコアSoCアーキテクチャ
ソフトウェア	***SMYLE Open CL*** メニーコアSoC向けプログラミング
実装評価	***SMYLE fpga*** 複数FPGAを用いた実行環境

13:10～13:30「SMYLE OpenCL: 組込みメニーコアSoC向けOpenCL環境」
- 組込みメニーコア向けOpenCL環境（公開予定）
- リアルタイム処理のための低レイテンシ実行
- 同一HWで4倍以上の性能向上

13:30～13:50 「組込みシステム向けメニーコア・アーキテクチャSMYLERefとその評価環境」
- 汎用メニーコアSoCアーキテクチャ開発
- 複数FPGAモードを用いた128コア実行プラットフォーム（公開予定）

4

成果報告ダイジェスト

13:50～14:10 「メニーコア時代のソフトウェア開発環境」
- プログラマとのインタラクションを重視したC to OpenCL変換（w/ 並列化）ツール（公開予定）
- アプリ移植後性能推定ツール（Webサービス実施予定）
- 各種最適化を施したOpenCLベンチマークプログラム開発（公開中）

14:10～14:30「SMYLEvideo：ビデオ・マイニング向けメニーコア・アーキテクチャとその性能」
- ビデオマイニングに特化したメニーコアアーキテクチャ
- ハードウェア/ソフトウェア・コデザイン
- 性能2倍弱，消費電力1/2，面積1/2を実現（見積り）

	トプスシステムズ，九州大学，電気通信大学	フィックスターズ，立命館大学
ハードウェア	***SMYLEvideo*** ビデオマイニング向けメニーコア	
ソフトウェア	***SMYLEvideo*** アルゴリズム（HWとのコデザイン）	***CLtrump*** C to OpenCL変換 ***PEMAP*** アプリ移植時の性能推定
実装評価	***FPGA*** デモ用ボード	***BEMAP*** メニーコア評価用OpenCLベンチマーク

5

招待講演＆パネル討論

14:45～15:30「組み込み応用向け低消費電力メニーコアSoCの開発」
株式会社東芝　セミコンダクター＆ストレージ社 半導体研究開発センター
主幹 宮森高
→超低消費電力組込み向け64メニーコア・プロセッサ

15:30～16:15「ARM next generation 64bit processors for power efficient compute」
John Goodacre, Director, Program Management ARM Processor Division
→ARM組込みプロセッサの最新情報

16:15～17:00「E2: A Scalable Dynamic Multicore Architecture」
Aaron Smith, Microsoft Research Redmond
→新しいマルチコア/メニーコア実行方式

17:15～18:20「メニーコアは何をもたらすのか？～基礎研究からビジネス展開まで～」
九大 井上弘士，タイレラ 石毛洋一，日立ハイテク 菊地修司，早稲田 木村啓二，イーソル 権藤正樹，トプスシステムズ 鳥居淳

6

立命館大学

SMYLE OpenCL
～組込みメニーコアSoC向けOpenCL環境～

冨山宏之　稗田拓路　西山直樹
江谷典子　谷口一徹

立命館大学

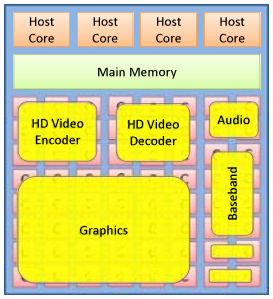
第2回メニーコアシンポジウム
2013年1月30日

SMYLE OpenCLの目的

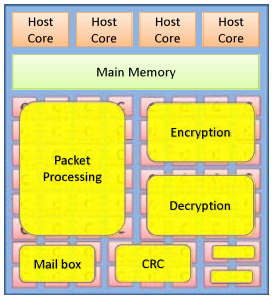
- メニーコアの組込みシステム向けSoCへの応用
  - ASIC型SoCから、ASSP型SoCへ
    - ASIC: Application Specific Integrated Circuit
    - ASSP: Application Specific Standard Product
  - 多品種少量生産から、少品種大量生産へ
- 課題
  - リアルタイム性の保証
  - 高性能（スループット）
  - 低消費電力

ソフトウェアによる専用化

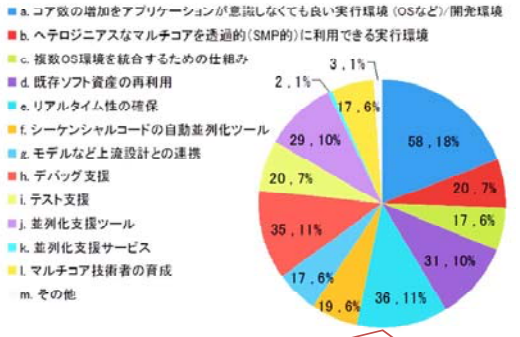
- タブレット用SoC



- ネットワーク機器用SoC



マルチ/メニーコアのソフトウェアの課題

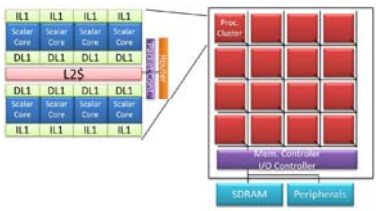


出典：JEITAアンケート（2011）

リアルタイム性の確保

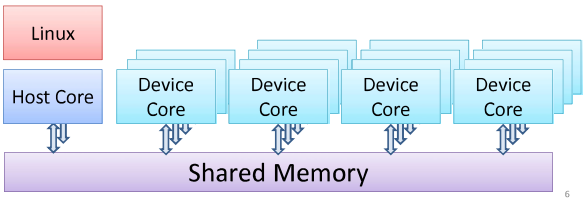
SMYLERefアーキテクチャ

- 九州大学と電気通信大学が開発
- クラスタの2Dメッシュ
  - 8コア/クラスタ
  - 各コアはMIPSベース
- 共有メモリ、プライベートL1キャッシュ、共有L2キャッシュ
- 128コアのFPGAプロトタイプ



SMYLERefのプログラミングモデル

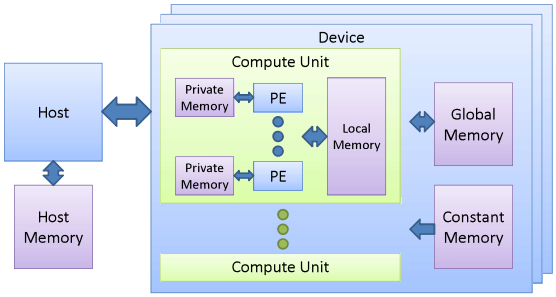
- 1コアをホストとして使用
  - Linux、仮想アドレス
- 残りのコアをデバイス（アクセラレータ）として使用
  - 物理アドレス
- 共有メモリ、ソフトウェア透過なキャッシュ



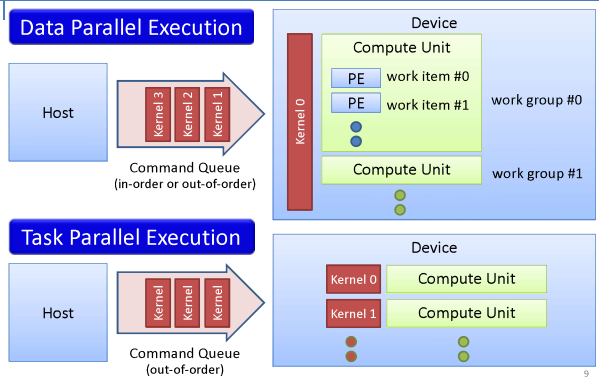
並列プログラミング

- メニーコアを活かすためには、並列プログラミング言語/フレームワークが必要
  - OpenMP, MPI, OpenCL, Intel Threading Building Blocks, Nvidia CUDA, etc
- OpenCLを採用
  - オープンで、ロイヤルティフリーな仕様
    - Specification 1.0 released in 2008
  - C言語ベース
  - ヘテロジニアスなプラットフォームに対応
  - プラットフォーム非依存
    - Intelマルチコア、Nvidia GPU、AMD GPU、Cell B.E.など
  - データ並列実行とタスク並列実行の両方に対応

OpenCLアーキテクチャモデル

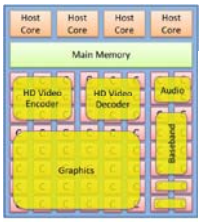


OpenCLプログラミングモデル

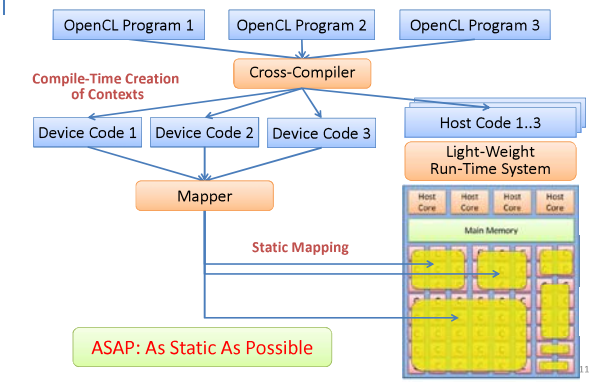


既存のGPU向けOpenCL環境の問題点

- 複数のOpenCLアプリケーションの並列実行が不可
  - 唯一のOpenCLプログラムがデバイス全体を占有
- リアルタイム性の保証が困難
  - 実行時の性能オーバーヘッドが大きい
    - コンテキストの生成、カーネルのディスパッチなど
  - オーバーヘッドの予測性が悪い

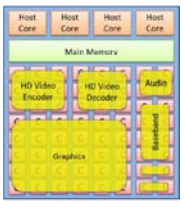


SMYLE OpenCL環境



SMYLE OpenCLの特徴

- 小さな起動時オーバーヘッド
  - 静的なコンテキストやオブジェクトの生成
  - 静的なタスクマッピング
- 様々なレベルにおける並列化
  - 複数のOpenCLアプリの並列実行
  - 個々のOpenCLアプリ内の
    - タスク並列実行
    - データ並列実行



典型的なOpenCLの実行の流れ

1. デバイス情報を取得、デバイスを確保
2. コンテキストを生成
3. コマンドキューやメモリバッファを生成し、コンテキストに格納
4. カーネルをビルドし、コンテキストに格納
5. 入力データをメモリバッファに格納
6. カーネルを実行
7. 実行結果をメモリバッファから読み出し
8. コマンドキュー、メモリバッファ、コンテキストなどを解放

13

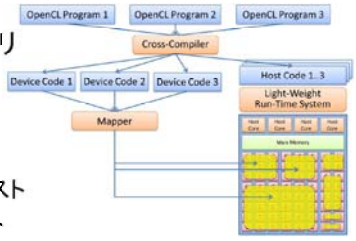
SMYLE OpenCLの実行の流れ

1. ~~デバイス情報を取得、デバイスを確保~~
2. ~~コンテキストを生成~~
3. ~~コマンドキューやメモリバッファを生成し、コンテキストに格納~~
4. ~~カーネルをビルドし、コンテキストに格納~~
5. 入力データをメモリバッファに格納
6. カーネルを実行
7. 実行結果をメモリバッファから読み出し
8. ~~コマンドキュー、メモリバッファ、コンテキストなどを解放~~

14

SMYLE OpenCL Toolkit

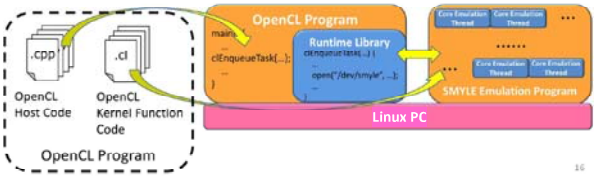
- ◆クロスコンパイラ
  - ◆GCCを使用
- ◆ランタイムライブラリ
  - ◆ホスト側
  - ◆デバイス側
- ◆マッパ
  - ◆シングルコンテキスト
  - ◆マルチコンテキスト
- ◆機能シミュレータ
  - ◆Linuxベースpc上で動作



15

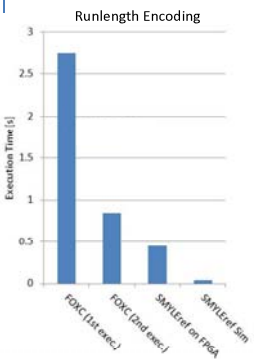
SMYLEref機能シミュレータ

- ◆OpenCLプログラムのデバッグ用
  - ◆各コアをpthreadがシミュレーション
  - ◆性能や消費電力の見積もりは不可能
- ◆通常のOpenCL環境として使用可能



16

実行時オーバーヘッドの評価



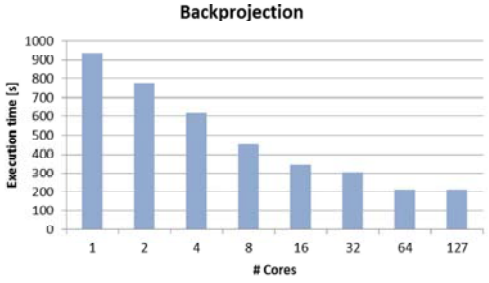
- ◆ FOXC
  - ◆ フィックスターズ社が開発したインテルマルチコア向けOpenCL
  - ◆ Core i7, 2.80GHz, 2 cores (4 threads)
  - ◆ 1回目の起動はキャッシュミス等のため低速
- ◆ SMYLEref on FPGA
  - ◆ 4 single-issue MIPS-based cores
    - ◆ 1 host and 3 device cores
  - ◆ Virtex 6, 10 MHz
- ◆ SMYLEref シミュレータ
  - ◆ Core i7, 2.80GHz, 2 cores (4 threads)

SMYLE OpenCL on 10MHz SMYLEref is faster than 2.8GHz Core i7

17

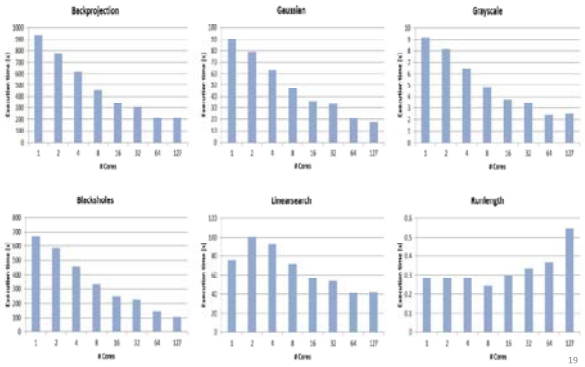
128コアSMYLEref/FPGA上での実行

- ◆Device cores: 1 to 127
- ◆Core clock: 10 MHz



18

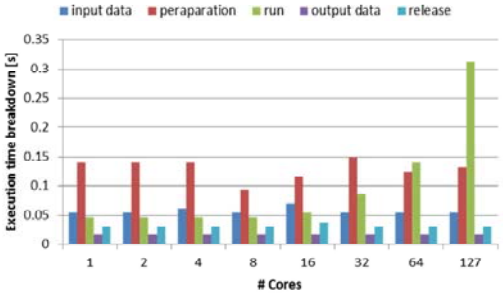
128コアSMYLEref/FPGA上での実行



19

Runlengthの実行時間の内訳

◆起動/終了時オーバーヘッドは（コア数に関わらず）ほぼ一定



20

アプリレベル並列実行の評価

◆2つの実行方法のスループットを比較

A) 各アプリケーションのデータ並列性を最大限活用

- ◆各アプリケーションに最大127コアを割り当て
- ◆アプリケーション間は逐次的に実行

B) 6つのアプリケーションを空間的に並列実行

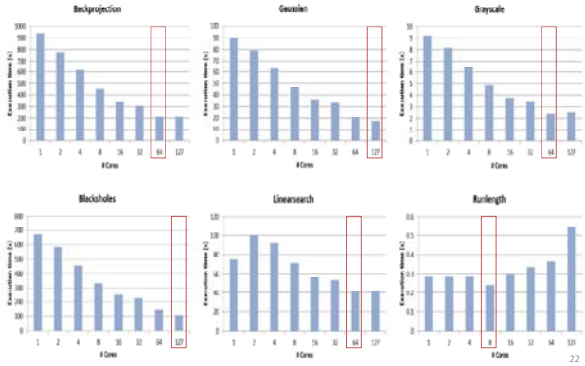
- ◆アプリケーションが使用するコア数の合計が127以下
- ◆各アプリケーションに割り当てるコア数を最適に決定

◆結果

- ◆アプリレベル並列実行（B）が4.03倍高速

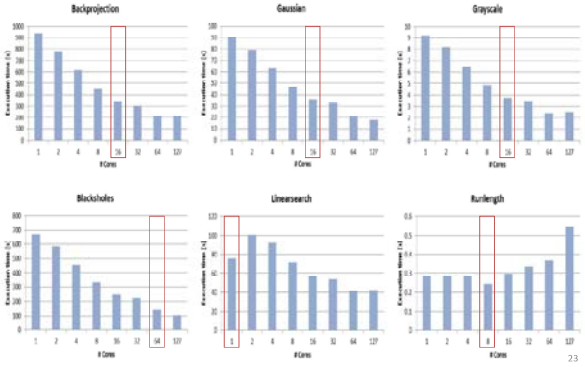
21

アプリ逐次実行の場合のコア数



22

アプリ並列実行の場合のコア数



23

まとめ

◆SMYLE OpenCL環境を開発

- ◆ランタイムライブラリ
- ◆マッパ
- ◆機能シミュレータ

◆特長

- ◆小さな実行時オーバーヘッド
  - ◆静的なコンテキスト生成
  - ◆静的なマッピング
- ◆様々なレベルでの並列実行
  - ◆アプリケーション間の並列実行
  - ◆アプリケーション内のデータ並列/タスク並列実行

◆128コアSMYLErefアーキテクチャ上で実証

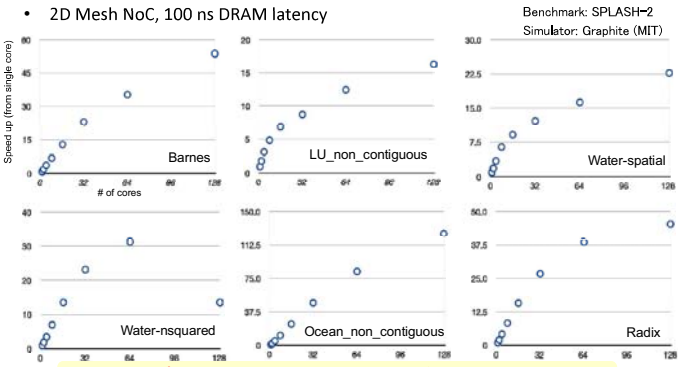
24

組込みシステム向けメニーコア・アーキテクチャSMYLErefとその評価環境

電気通信大学
近藤 正章、グェン チュオン ソン
九州大学
平尾 智也、曽我 武史、佐々木 広 、井上 弘士

単一アプリ性能はスケールするのか?

- In-Order Core@1GHz (up to 128cores) w/ private 32KB L1 & 512KB L2
- 2D Mesh NoC, 100 ns DRAM latency



多くのアプリケーションではスケーラビリティが低い☹

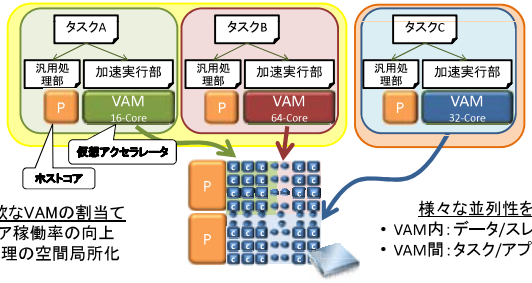
メニーコアプロセッサの課題

- スケーラビリティが低い
  - アプリケーションプログラムが持つ並列性の不足
  - メモリアクセスのボトルネック
  - バリア同期のオーバーヘッド

- 組み込みシステム向けメニーコア・プロセッサの要件
  - 効率的な並列処理の実現による高性能・低消費電力化
  - 様々な並列性(データ / スレッド / タスク / アプリレベル)の活用
  - メモリアクセスボトルネックを緩和するキャッシュメモリ制御
  - 高速バリア同期のサポート

SMYLErefの基本コンセプト

- 柔軟な仮想アクセラレータ実行プラットフォーム:VAM
  - VAM: Virtual Accelerator on Many-core
  - 小規模コアを多数用いてメニーコアを構成

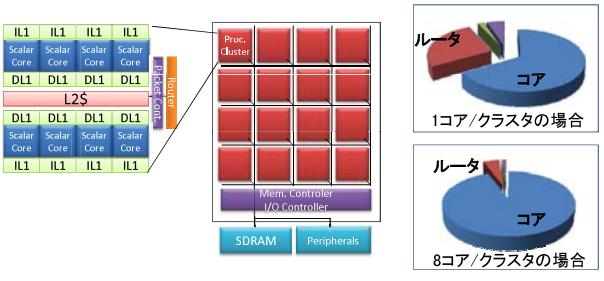


柔軟なVAMの割当て
- コア稼働率の向上
- 処理の空間局所化

様々な並列性を活用
- VAM内:データ/スレッド並列
- VAM間:タスク/アプリ並列

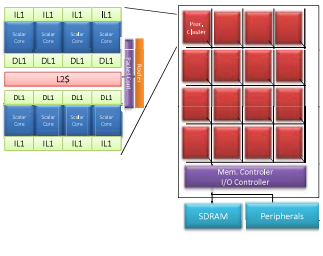
メニーコアアーキテクチャSMYLEref

- SMYLEref: VAM実装のリファレンスアーキテクチャ
  - 数個のプロセッサコアをバスで結合したクラスタ構成
  - クラスタを2次元メッシュのオンチップネットワーク(NoC)で結合
  - → クラスタ化によりルータのコスト減(より多くのトランジスタを演算に利用)



SMYLErefの構成

- プロセッサコア
  - 「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(代表:東大 中村宏教授)」で開発されたGeyserコア
  - MIPS R3000ベース、LSI実装の実績あり
- クラスタの構成
  - Processor Cluster
    - 複数個のコア
    - 分散共有L2$
    - ルータがCluster-Busを通して接続
  - Peripheral Cluster
    - DRAMコントローラ
    - I/Oコントローラ

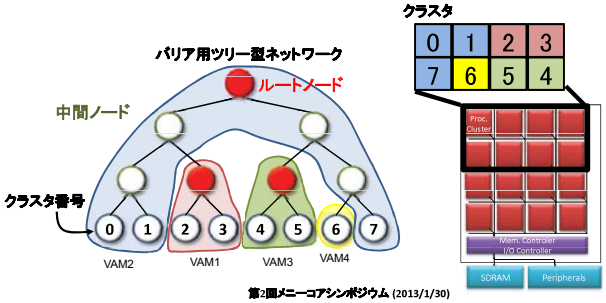


VAM向けの拡張

- 再構成可能L1キャッシュ
  - コア毎にL1キャッシュメモリをキャッシュとして、もしくはSPMとして利用
  - VAMの特性に応じてコンパイラにより決定
- 分散共有L2キャッシュの拡張
  - 基本構成：L2キャッシュは全クラスタで共有
  - 拡張構成：VAM毎にL2キャッシュグループを構成
    - ソフトウェアで設定するアドレス変換機構によりL2キャッシュ位置指定
    - VAM間でのL2キャッシュ競合の回避
- グループ・ハードウェアバリア
  - 各VAMに割り当てられたコアグループを対象にした高速ハードウェアバリアをサポート

グループハードウェアバリア

- バリア同期専用のツリー型ネットワーク
  - VAM内での高速バリア同期の実現
  - 複数VAM間でバリア同期が可能



メニーコア評価環境

- 評価環境の必要条件
  - 大規模メニーコアも評価・検証可能なスケーラビリティ
  - OSを含めた実プログラムの評価・検証が行える環境
  - 種々の構成が評価可能な柔軟性、コスト、・・・
- 評価環境の候補
  - ソフトウェアシミュレータ / LSI実装 / FPGAプロトタイピング

	スケーラビリティ	評価精度	柔軟性	開発コスト	評価スピード
ソフトウェア	×	△	◎	◎	×
LSI実装	△	◎	×	×	◎
FPGA	◎	○	○	○	○

FPGAによるプロトタイピングが有望

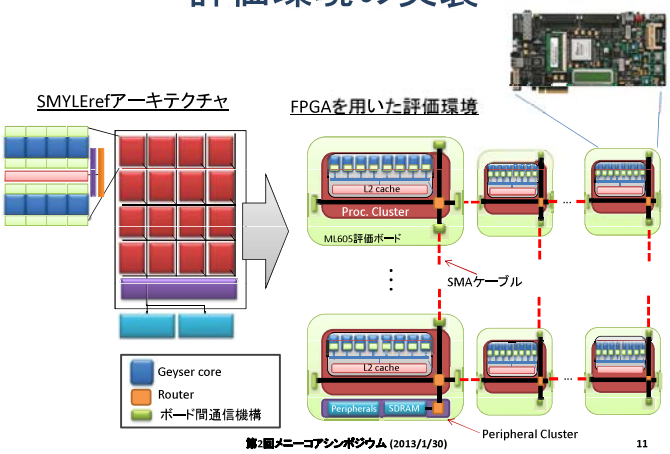
開発環境

- 評価ボード：Xilinx Virtex-6を搭載するML605評価ボード
- 回路設計：Verilog HDL
- 論理合成、マッピング、配置配線：Xilinx ISE

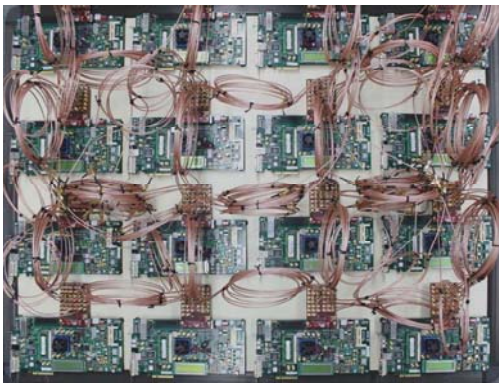
ML605ボード	
FPGAデバイス	Virtex-6 XCVLX240T
SDRAM	DDR3 SO-DIMM
搭載I/Oポート	UART, USB, DVI出力、CF、SMA等
クロック入力	200MHz & 66 MHz オシレータ

Virtex-6（XCVLX240T）	
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

評価環境の実装



評価環境の外観

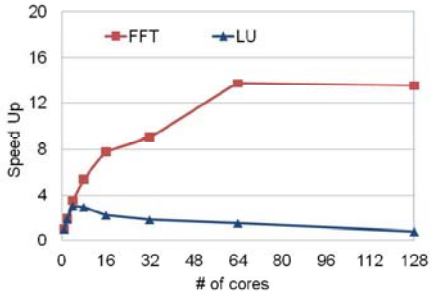


評価の仮定

- ハードウェア
  - 8コア x 8クラスタ + 1ペリフェラルクラスタ (8ボード)
  - コア : 10MHz、内部バス・ルータ : 5MHz、DDR3-SDRAM : 100MHz
  - コヒーレンス制御なし → ヒープ領域はuncacheable
- ソフトウェア
  - ベンチマーク : SPLASH2のFFT, LU
  - コンパイラ: MIPS用のコードを生成するgcc 4.4.6
  - 浮動小数点演算 : ソフトウェアエミュレーション(Soft Float)
- 並列処理API
  - SMYLEref評価環境向け簡易版pthreadライブラリ

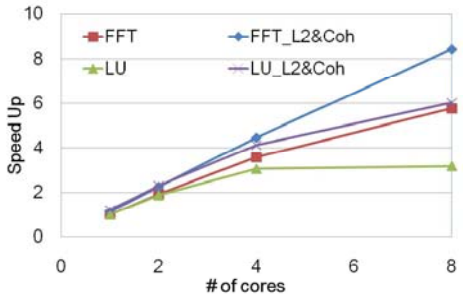
初期評価結果

- 128コアまで評価可能！
- スケーラビリティが得られていない
  - ヒープデータがuncacheableなためメモリアクセスがボトルネックに
  - 同期や排他制御の度にキャッシュをフラッシュするため



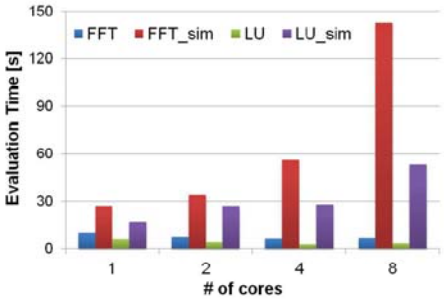
(参考)キャッシュコヒーレンスがある場合

- 1クラスタ内でキャッシュコヒーレンス制御を試作
  - スヌープベースのMSIプロトコルを利用
  - 共有データをキャッシュ経由でアクセス → 性能・スケーラビリティ向上

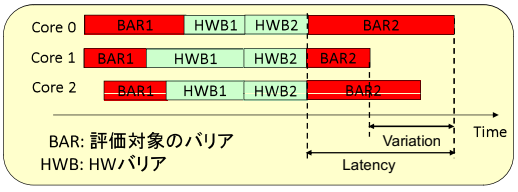


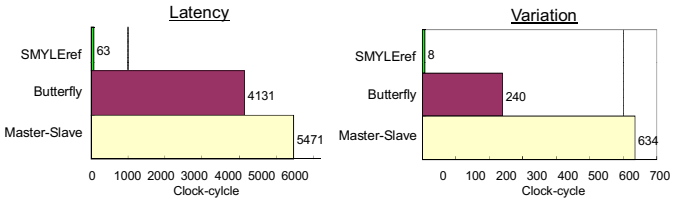
評価環境としてのスケーラビリティ

- 評価時間の比較　SMYLEref v.s. ソフトウェアシミュレータ
  - ソフトウェアシミュレータ: MARSS-x86 simulator
- SMYLEref評価環境はメニーコアを評価する上で有効



ハードウェアバリアの評価結果





MPEGデコードのデモ

- SMYLEref評価環境でのMPEG2デコードのデモ
  - MediaBenchのmpeg2decをpthreadで並列化
  - 8cores x 2-clusters, コア周波数は40MHz

1 core　　16 cores

まとめと今後の課題

- SMYLEref: 組み込みシステム向けメニーコア アーキテクチャ
  - VAMのコンセプトを実現するリファレンスアーキテクチャ
  - クラスタ構成の導入による拡張性、ハードウェアの効率化
- FPGAによる評価環境の構築
  - SPLASH2を用いて128コアまで評価
  - OpenRISC版の評価環境コードはBSDライセンスで公開予定
- 今後の課題
  - SMYLEコンパイル環境を用いたベンチマーク評価
  - VAM向け拡張機能の実装と評価

FIXSTARS
Speed up your Business

メニーコア時代のソフトウェア開発環境

株式会社フィックスターズ
取締役 COO 田村　陽介

フィックスターズについて

➔ フィックスターズは、マルチコアプロセッサにおける
トータルソリューションカンパニーです。

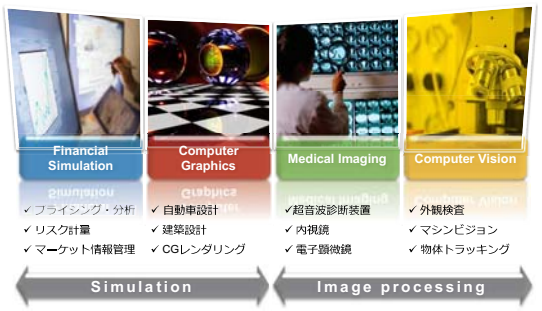
- 設立　2002年8月
- 社員数　87名（2012/12月現在）
- 所在地　東京、カリフォルニア、ビクトリア
- 創業者　代表取締役会長 - 長谷川 智彦
  代表取締役社長 CEO - 三木 聡

フィックスターズの事業

➔ マルチコアプロセッサを活用し、お客様のビジネスを加速するためのサービスや製品を提供しています。

注力分野

➔ 高速計算や大容量データ処理など、高いコンピューティングパワーを必要とする分野にフォーカスしています。



NEDOプロジェクトでの活動

➔ **NEDOプロジェクト**
「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」
✓ メニーコア向けソフトウェア開発環境を担当

➔ **本プロジェクトでのアウトプット**
✓ 本プロジェクトでの研究開発チップだけを対象にするのではなく、ソフトウェア開発ツールとして汎用性の高いものを開発して公開
✓ 特に弊社のこれまでの産業界での経験を活かしメニーコア時代に要求されるであろうソフトウェア開発環境を探求

➔ **本日の話の内容**
✓ メニーコア時代のソフトウェア開発環境とは何か？
✓ メニーコア時代にエンジニアに新たに要求されることは何か？
✓ 時代背景やソフトウェア業界の状況を含めながら紹介する
✓ 最終成果物としてどのようなものが出来上がったのか？

時代背景① 並列処理を強いられるソフトウェア

➔ ご存知の **「The Free Lunch Is Over」**
2005年Microsoftの Software ArchitectであるHerb Sutter氏の言葉

Free Lunch is over とは？
これまでCPUはシングルコアの性能を高める方向で進化したためソフトウェアは何もしなくてもCPUの進化の恩恵を受けていた。しかし、CPUベンダがマルチコアに方向転換したため逐次プログラムの性能は伸びずに、並列プログラムの性能が上がっていく。ソフトウェア開発者はCPUの性能を活かすためにソフトウェアを根底から見直さなければならない。

前回の資料

時代背景② 応用指向プロセッサの活用

- 「応用指向プロセッサ」の利用が積極的に検討されはじめた
  - ✓ 特定の処理を想定しているためコアをシンプルに小さくすることができる。そのため、処理によっては飛躍的に性能を向上させることが可能
- 汎用用途に歩み寄るGraphics Processing Unit (GPU)
  - ✓ ソフトウェア開発環境も整備されてきており、これまでのように特殊な知識がなくてもプログラミングが可能となった
- ヘテロジニアスマルチコアの存在
  - ✓ Sony/Toshiba/IBMで開発された Cell/B.E.に続き、Intel社のIvy Bridge, AMD社のFusionといった1チップ内に汎用プロセッサと応用指向プロセッサを搭載したプロセッサが製品化されている。

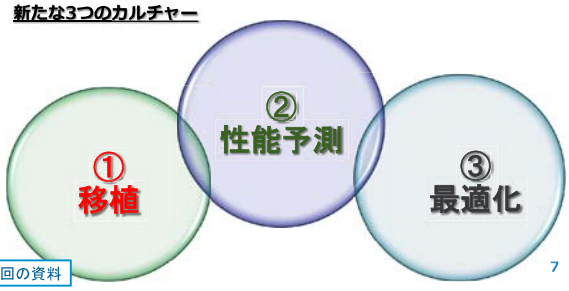
NVIDIA CUDA　OpenCL

前回の資料

ソフトウェア業界に根付く新たなカルチャー

- 時代の変化によりソフトウェア業界に新たなカルチャーが根付いた
- 特に「汎用プロセッサのマルチコア化」と「応用指向プロセッサの活用」はソフトウェアエンジニアに新たな技術の習得を強いることになる
- 今後のソフトウェア環境はこれらの技術を支援することが求められる

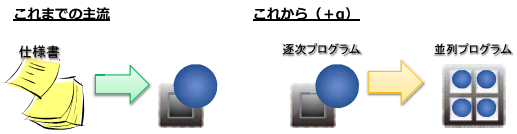
新たな3つのカルチャー



前回の資料

①「移植」カルチャー

- シングルコアが性能年々向上していた「Free Lunch 」の時代にはプロセッサの変更に伴うソフトウェアの変更は必要なかったため、ソフトウェア開発といえば**仕様書に基づいた新規開発**が主であった
- マルチコアの時代には、新規開発の他に**既存ソフトウェア資産の移植**　という作業が発生する。
- 新規開発においても、リファレンスコードとなる逐次プログラムを最初に開発してから、そのプログラムを**並列プログラムに移植開発**するといった開発方法が一般的となっている。



前回の資料

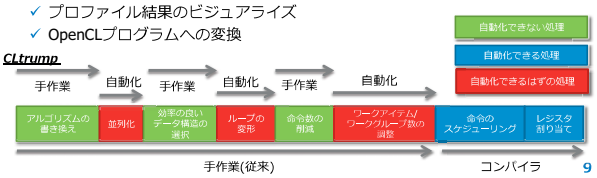
開発ツール①：CLtrump
C to OpenCL Translator Utilities for Many core Processor

概要

- ✓ 並列プログラムの開発作業において、自動化可能な部分と手動化が必要な部分が混在している。CLtrumpでは、自動化可能な部分はツールで自動化し、手動化が必要な部分はユーザが自ら開発するようなインタラクティブな並列プログラム開発環境を提供する。

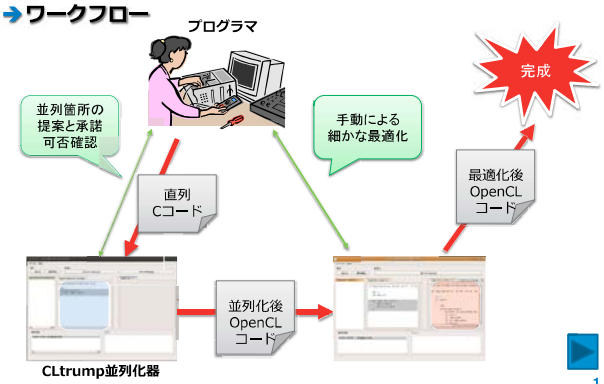
特徴

- ✓ 楽観的な並列性検出器
- ✓ PythonでC99のトランスレータを書くためのインタフェース
- ✓ 実行時プロファイル情報を活用した並列性の抽出
- ✓ 実行時間を考慮した並列化
- ✓ プロファイル結果のビジュアライズ
- ✓ OpenCLプログラムへの変換



開発ツール①：CLtrump
C to OpenCL Translator Utilities for Many core Processor

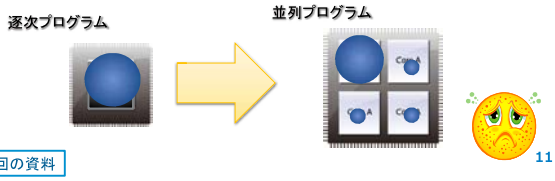
ワークフロー



http://ctrump.sourceforge.net/cltrump.html

②「性能予測」カルチャー

- ソフトウェア移植の最大の魅力は性能向上である。そのため、既存のソフトウェアを移植する前に**ある程度の性能予測**を要求されることが多い。
- 産業界では最終的な性能がある程度見込まれてから予算が確保できるケースが多いため**なるべく低コストでの見積もり作業**が求められる。
- 見積もり作業には詳細なコード分析が必要となり時間がかかる。体系的な見積もり手法が確立されていないため**見積るエンジニアによって精度もバラバラ**であり信憑性も薄いことが多い。



前回の資料

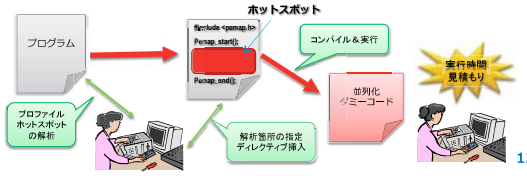
開発ツール②：PEMAP
Performance Estimator for Many core Processors

概要

- ✓ 本開発ツールでは、アーキテクチャの異なるプロセッサ間の移植において、移植前と移植後の命令セットの頻度分布やメモリアクセスパターンに何らかの関連性があることに着目し、移植前のプログラムから、その特性を維持した移植後のダミーコードを自動生成する。ダミーコードを移植後のプロセッサで実行することで性能の見積もりが可能となる。

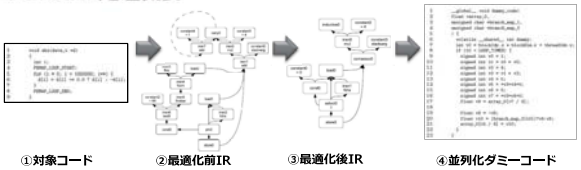
特徴

- ✓ 実行可能なダミーコードを自動生成
- ✓ ダミーコードではオリジナルコードと同じ出力結果を保証しない代わりに、計算内容とメモリアクセスパターンと条件分岐の分岐方向が再現されている。

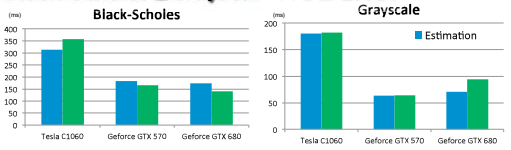


開発ツール②：PEMAP
Performance Estimator for Many core Processors

PEMAP処理概要



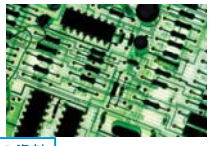
Black-ScholesとGrayscaleの処理を評価



デモ

③「最適化」カルチャー

- ハードウェアに最適なプログラムをソフトウェア屋さんが意識する時代
  - ✓ Cell/B.E.の登場や、GPUを使ったプログラムはその典型例
  - ✓ 最近のプロセッサの中にはハードウェアを意識しないでプログラムを書くと性能がほとんど出ないようなものも存在する
- 高品質なソフトウェアを作るためには、プロセッサが提供しているマイクロアーキテクチャの機能をうまく活用することが求められる
  - ✓ ハードウェアだけで実現可能なマイクロアーキテクチャ技術が限界
  - ✓ ソフトウェアとの連携を想定したマイクロアーキテクチャが多く登場
  - ✓ メニーコア化のためにコアのリッチな機能を削減しているプロセッサもある。



前回の資料

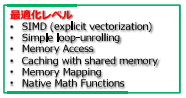
ベンチマークの公開③：BEMAP
Benchmark for Automatic Parallelization

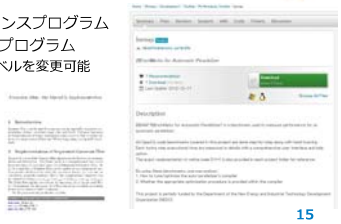
概要

- ✓ 弊社でこれまで蓄積してきた最適化技術のノウハウを共有する目的でオープンソースとして公開。高速処理のためのソフトウェアの最適化が強く要求される4分野(ライフイノベーション、インターネットアプリケーション、ファイナス、ビジュアルコンピューティング)8種類のコードを公開。
- ✓ http://sourceforge.net/projects/bemap/

公開内容

1. C言語で記述されたリファレンスプログラム
2. OpenCLで記述された最適化プログラム
   - 引数の指定により最適化レベルを変更可能
3. 性能結果や最適化の詳細が記載された技術レポート





ベンチマークの公開③：BEMAP
Benchmark for Automatic Parallelization

ms

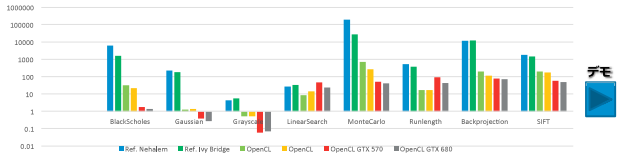
Workload	Ref. Nehalem	Ref. Ivy Bridge	OpenCL Nehalem	OpenCL Ivy Bridge	OpenCL GTX 570	OpenCL GTX 680
BlackScholes	6109.07	1587.80	32.05	21.41	1.70	1.40
Gaussian	227.42	179.30	1.24	1.40	0.39	0.27
Grayscale	4.32	5.47	0.53	0.52	0.06	0.07
LinearSearch	26.25	33.33	8.42	14.24	47.14	24.08
MonteCarlo	193817.40	27423.70	696.53	264.49	49.68	41.50
Runlength	518.28	370.50	17.07	16.77	90.29	43.36
Backprojection	11297.43	11872.04	195.03	116.08	75.83	69.70
SIFT	1860.00	1452.00	194.11	170.51	55.54	48.02

Nehalem
Intel i7-X990 @ 3.47GHz (Nehalem)
8GB memory, 6 cores, 12 threads (HT)

Ivy Bridge
Intel Core i7-3770K @ 3.50GHz (Ivy Bridge)
8GB memory, 4 cores, 8 threads (HT)

GTX 570:
NVIDIA GTX 570 @ 1.46GHz (Fermi GF-110)
480 CUDA cores, 1.2GB GDDR5 memory

GTX 680:
NVIDIA GTX 680 @ 1.06 GHz (Kepler GK-104)
1536 CUDA cores, 2GB GDDR5 memory



デモ

まとめ

- 本プロジェクトはメニーコア時代におけるソフトウェア開発環境の研究開発しました。
- 弊社のこれまでの経験に基づきソフトウェア開発の現場において起こっている新たな潮流を「移植」「性能見積もり」「最適化」という3カルチャーとして紹介しました。
- 紹介した3カルチャーに対してエンジニアを支援するために本プロジェクトでは具体的なツールを開発しました。
- 今後開発される新しいメニーコアチップに対しても同様のアプローチによりツールを拡張し対応させることが可能です。
- 本プロジェクトの成果はできる限り公開し具体的な商用利用に関しても当社で支援する体制を整える予定です。

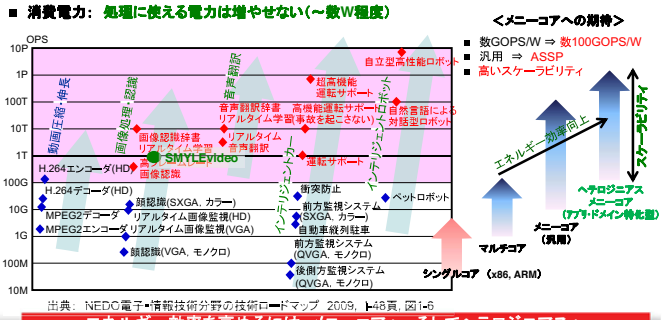
SMYLEvideo：ビデオ・マイニング向け メニーコア・アーキテクチャと性能

2013年 1月30日

株式会社トプスシステムズ
代表取締役　松本 祐教

次世代アプリケーションとプロセッサ

- 次世代組込みシステム：要求処理能力は、なお止まるところを知らない（数100GOPS~数POPS）
- 消費電力：　処理に使える電力は増やせない（~数W程度）



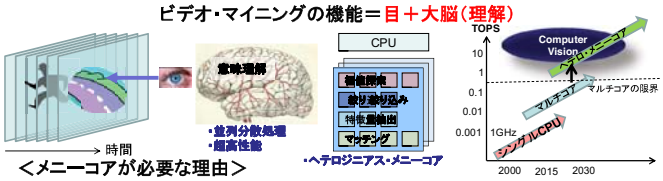
出典：　NEDO電子・情報技術分野の技術ロードマップ　2009，p48頁，図1-6

エネルギー効率を高めるには、メニーコアへ、そしてヘテロジニアスへ

SMYLEvideo: アプリケーション・ドメイン特化型のヘテロジニアス・メニーコアを目指して

『時系列画像入力に対するモデル（学習）ベースの3次元の意味理解』

ビデオ・マイニングの機能＝目＋大脳（理解）

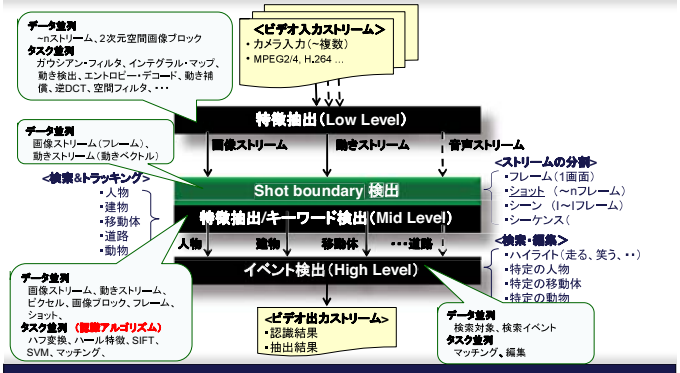


＜メニーコアが必要な理由＞

- アプリに潜在する高い並列性：　全体の99%以上が並列化可能
- 高い要求処理能力（1TOPS以上）：　既存のプロセッサでは実現困難
- 複数のタイプの異なる処理：　ハード化では効率低下（コスト増）
  - ビデオ・デコーダ：　MPEG2, MPEG4, H.264, ・・・
  - 解像度：　VGA, XGA, SGGA, FHD, 2K, 4K, ・・・
  - アルゴリズム：　SIFT, Optical Flow, Ransac, ハール特徴、モデル・ベース顔認識等

重要な用件は、高い性能と柔軟さ

ビデオ・マイニング・システムの概要



ビデオマイニング用アルゴリズムに内在する並列性

アプリ処理	目的	アルゴリズム	並列性	
ビデオ分析	動作の方向と速度の予測	オプティカルフロー	行方向の並列性	
	特定の特徴の検出と抽出	SIFT*1	分割データの並列性	
	人物の検出・動作の追跡	カスケード型ハール特徴	ピクセルレベルの並列性	
	領域を区別するライン検出	ハフ変換	行方向の並列性	
	連続画像を用いて誤検出を排除	Ransac*2	サンプルデータの並列性	
人物検索	様々な角度からの顔検出	ベクター型顔検出	画像ブロックの並列性	
	顔の特徴を抽出	モデルベース顔認識	タスクレベルの並列性	
	特定の特徴の検出と抽出	SIFT	画像ブロック・特徴量の並列性	
ビデオ編集	セグメントの抽出	グラフベースセグメント分割	セグメントレベルの並列性	
	動きベクトルの検出	ブロックマッチング	行方向の並列性	

*1 SIFT：Scale-Invariant Feature Transform
*2 Ransac：Random Sample Consensus

SMYLEvideoメニーコア開発の目標

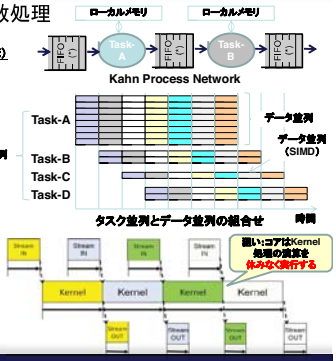
- 処理性能：　1TOPS~（リアルタイムマイニング処理）
- スケーラビリティ：　10fps, 20fps, 30fps
- プログラマビリティ：　ソフトウェアで各種アルゴリズムに対応
- 柔軟性：　OpenCV (Computer Vision)が扱える

  SIFT, Optical Flow, Ransac, Viola & Jones, Model based Recognitions, SVM, etc.
- 消費電力：　~1.5W程度（組込み用）
- 動作クロック:　~100MHz

超高性能、プログラマブル、スケーラブル

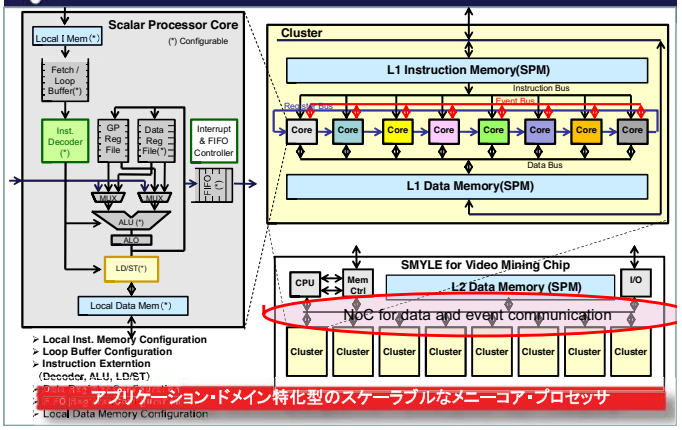
ビデオ・マイニングに適用した処理方式

- Kahn Process Network型の分散処理
  - 複数のメモリ非共有プロセス
  - プロセス間の通信(ゼロオーバーヘッドFIFOアクセス機構)
- 2つの並列処理の組合せ
  - 分散並列処理(タスク、パイプライン並列)
  - データ並列処理(上位レベル、命令レベル)
- ストリーム処理(Core)
  - Kernel(プロセス内の演算処理)
  - Stream-In(メモリ・アクセス)
  - Stream-Out(メモリ・アクセス)
- Coreの最適化
  - ストリーム処理のサポート:Stream-In/Outの隠蔽
  - 複合命令:Kernelのサイクル数削減
  - FIFOのサポート
  - 命令供給＆データ供給エネルギーの削減



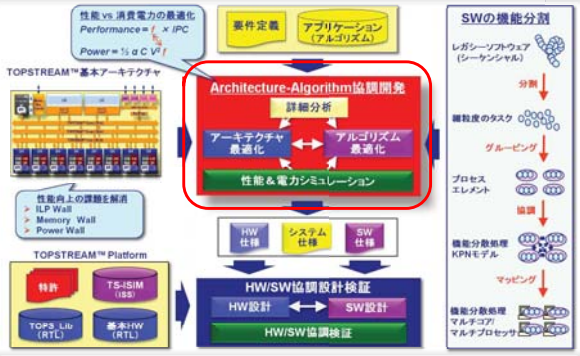
TOPS Systems Corp. www.topscom.co.jp

SMYLEvideo：基本アーキテクチャ



アプリケーション・ドメイン特化型のスケーラブルなメニーコア・プロセッサ

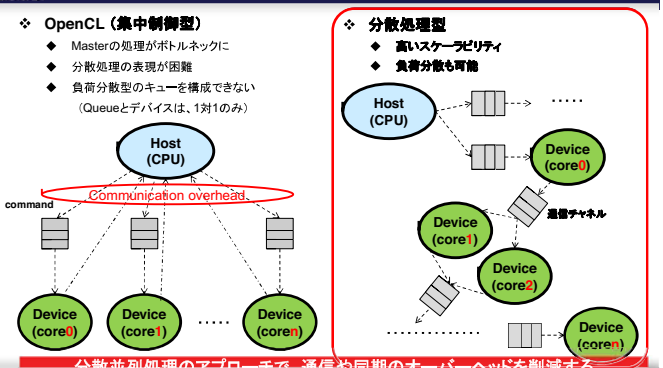
SMYLEvideo用HW/SWの開発フロー



アーキテクチャ＆アルゴリズム協調設計による最適化

OpenCL vs. 分散処理

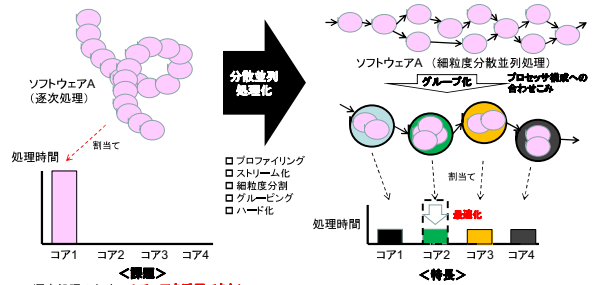
- OpenCL（集中制御型）
  - Masterの処理がボトルネックに
  - 分散処理の表現が困難
  - 負荷分散型のキューを構成できない
    （Queueとデバイスは、1対1のみ）
- 分散処理型
  - 高いスケーラビリティ
  - 負荷分散も可能



分散並列処理のアプローチで、通信や同期のオーバーヘッドを削減する

TOPS Systems Corp. www.topscom.co.jp

ソフトウェアの分散並列処理化



<課題>
- 逐次処理のためマルチコアを活用できない
- 高い処理能力のCPUが必要(画像認識: OpenCV)
- 分散並列処理化が困難(工数が掛かる)

<特長>
- 分散並列処理で各種マルチコアの性能を活用
- ハードウェア化による高速化が可能
- 分散並列処理化が容易

TOPS Systems Corp. www.topscom.co.jp

エネルギー効率向上へのアプローチ

- 目的： 超高性能 & 低消費電力のプログラマブルなアクセラレータ
  (Energy-Efficient, Low Cost, Flexible, Scalable)
- Approach： クロック周波数の低減

◆ 消費電力

$$P_{Total} = P_{Dynmic} + P_{Static}$$

$$= \frac{1}{2} \alpha C V^2 f \alpha + I_{Leak} V$$

C: Load capacity
V: Source voltage
f: Frequency
α: Siwitching rate
I_{Leak}: Leakage current

◆ 処理性能

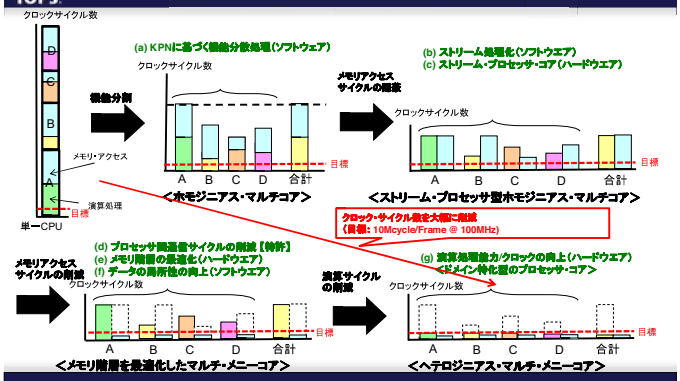
$$Performance = OPC \times f$$

OPC: Operation Per Clock

低い動作クロックで、高性能を追求することで、エネルギー効率を高める

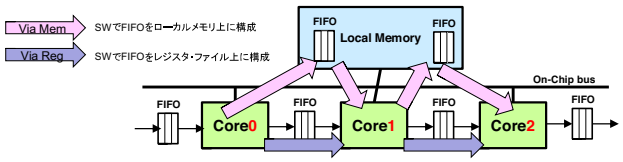
TOPS Systems Corp. www.topscom.co.jp

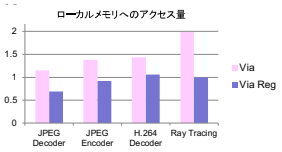
高性能化の基本アプローチ



レジスタ・バンク共有によるコア間通信

■ メッセージ・パッシングの経路

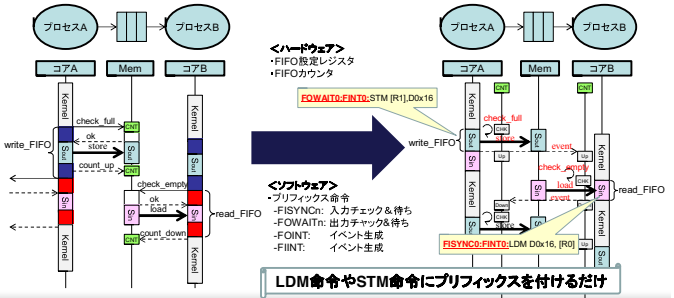




ローカルメモリへのアクセス量： 30%以上削減

ゼロ・オーバーヘッド・メッセージ・パッシング機構（ZOMP）

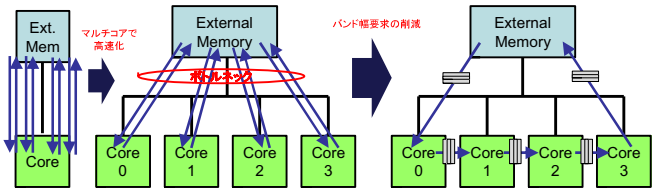
■ 狙い：FIFOアクセスのオーバーヘッド（エントリ数管理）の解消



分散並列処理によるメモリアクセスの削減

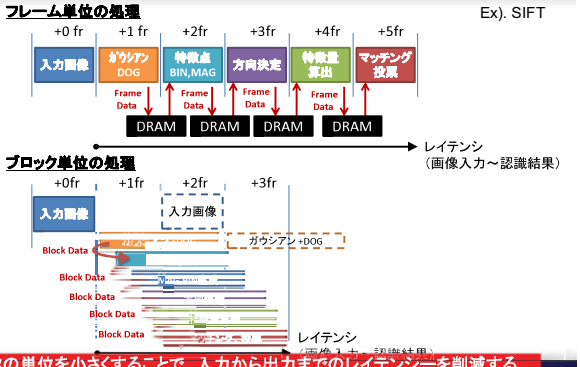
■ メモリ・セントリックな処理
- 各コア上のソフトウェアは外部メモリ上のデータに対する処理
- プロセッサのメモリの集積

■ 分散ストリーム処理
- コアからコアへのストリーム渡し（FIFO）
  - オンチップ・メモリ
  - レジスタ・バンク共有



必要なメモリバンド幅を低減し、高性能化・低消費電力化を追求する

フレーム単位の処理 vs. ブロック単位の処理

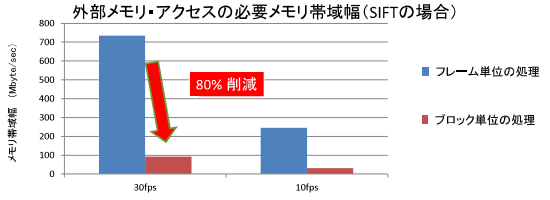


データの単位を小さくすることで、入力から出力までのレイテンシーを削減する

フレーム単位の処理 vs. ブロック単位の処理

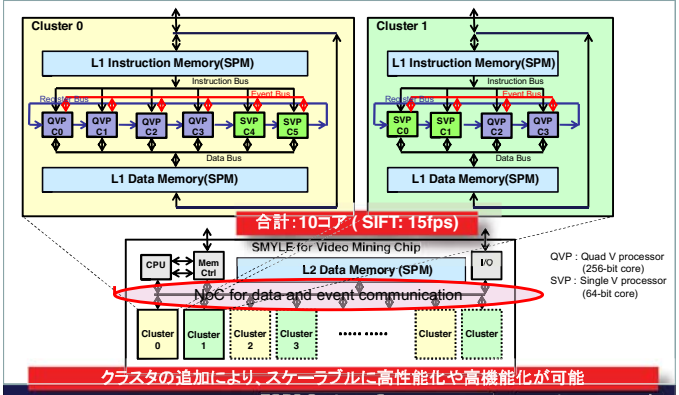
	フレーム単位の処理	ブロック単位の処理
外部メモリの使用量	22Mbytes	3.1Mbytes
ローカルメモリの使用量	0.15Mbytes	0.8Mbytes
処理のレイテンシー	167mSec(5.1frame)	100mSec(3frame)

外部メモリ・アクセスの必要メモリ帯域幅（SIFTの場合）



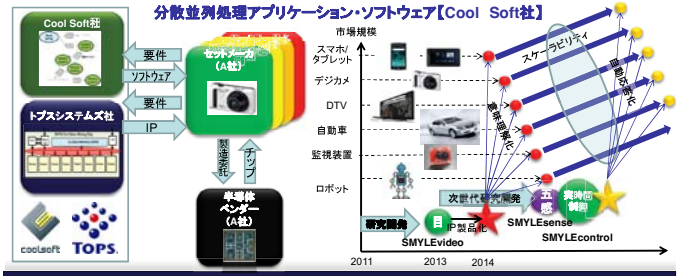
外部メモリの使用量：1/7倍，必要メモリ帯域幅：1/5倍

SMYLEvideoの構成



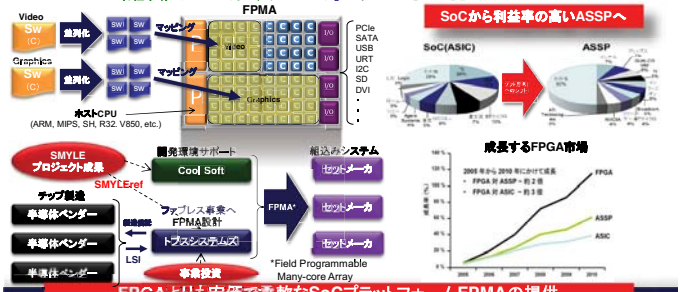
SMYLEvideo IPの事業化

- 狙い：メニーコア技術により、次世代システム要件を実現する
  （システム・メーカの欲しいIPを提供する）
- 手法：Architecture-Algorithm協調設計による全体最適化設計を徹底した高性能IP
- IPの提供：応用分野（ドメイン）特化型メニーコアIP 【トプスシステムズ社】
  分散並列処理アプリケーション・ソフトウェア【Cool Soft社】



SMYLERefメニーコアの事業化

- パラダイム・シフト
  ハードウェア設計：ロジックをプログラム（FPGA） ⇒ ソフトウェア設計：並列処理をプログラム（FPMA）
- 開発環境：逐次処理ソフト（C言語）を並列化して「メニーコア」にマッピング
- FPMA：各種市場セグメントに対応する「メニーコア」に「ホストCPU」と「I/O」を内蔵するチップ



FPGAよりも安価で柔軟なSoCプラットフォームFPMAの提供

まとめ

- メニーコア・プロセッサは、コンピュータ・ビジョン・システムのコア・アプリの1つである『ビデオ・マイニング』に必要とされる超高性能で、柔軟なSoC（ASSP）の実現手段となる。
- 『ゼロ・オーバーヘッド・メッセージ・パッシング（ZOMP）機構』により、メニーコア・プロセッサの性能を効果的、且つスケーラブルに向上することができる。
- 『ブロック単位の分散並列処理』により、メニーコア・プロセッサを活用時のメモリ・アクセス帯域を大幅に削減でき、コア数の増加等によりシステム性能を更に向上できる。
- 『SMYLEvideoメニーコア』は、クラスタの追加により、処理性能や機能をスケーラブルに向上できるアーキテクチャである。

添付資料５

「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」
に係る委託業務実施計画書（抜粋）

法人名： 国立大学法人　九州大学
学校法人立命館 立命館大学
国立大学法人　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

１．実施計画の細目（手法・手段・研究場所等）

（１）事業目的

低消費電力メニーコアの実現においては、大多数の小規模コアの徹底した使用率の向上と、その動作時に消費する電力の大幅な削減が最も重要となる。そして、「コア数にスケール可能な高性能化（コア数を増やせばより性能が高くなる)」と「コア数にスケール可能な低消費電力化（コア数を増やせばより消費電力を削減できる)」といったメニーコアならではの技術開発の実施が急務の課題である。そこで本事業では、組込みシステムにおける低消費電力メニーコアのあるべき姿として「仮想アクセラレータとその実行プラットフォームとしてのメニーコア」を提案し、それを可能にするアーキテクチャの開発、各種 API の策定、ならびに、コンパイラを含めたアプリケーション開発環境の開発を行う。また、シミュレーションならびにプロトタイプにより有効性を明らかにすると共に、提案メニーコアの適応分野に関する調査を実施し実用化に向けた方向性を示す。

（３）事業内容

Ａ－①汎用メニーコア・アーキテクチャの研究開発（主担当：九州大学、副担当：電気通信大学、東京農工大学）

本研究で提案する仮想アクセラレータを効率良くマッピング可能なメニーコア・アーキテクチャを開発する。平成 22 年度においては、メニーコア・ドメインならびにマルチコア・ドメインの基本アーキテクチャを開発する。平成 23 年度には、アーキテクチャ・レベルでの最適化により従来技術と比較して消費電力を 1/2 に削減できることを明らかにする。また、研究テーマＣで開発した低消費電力技術、ならびに、研究テーマＢによるコンパイルを用いることで、仮想アクセラレータの導入により性能 2 倍、消費電力量 1/10 が実現可能であることを簡易シミュレーションにより示す（ただし、ここでは小規模な評価用プログラムを用いる）。平成 24 年度においては、プロセッサコアの拡張やオペレーティングシステムの整備などを行うとともに、テーマ A–③による HDL を用いた設計データなど成果の公開準備を進める。また、総合評価を実施し、実アプリケーションにおいて性能 2 倍、消費電力量 1/10 を達成する。

Ａ－②ビジュアル・コンピューティング・メニーコア・アーキテクチャの研究開発（担当：トプスシステムズ）

ビデオ・マイニングをターゲット・アプリケーションとしメニーコア・アーキテクチャを開発する。平成 22 年度は、ビデオ・マイニング・アプリケーションについて、命令レベルの並列性(ILP)、データレベルの並列性(DLP)、タスクレベルの並列性(TLP)、対象アプリケーションを構成するアルゴリズムの種類、演算処理のタイプ、データ型、適用可能な並列処理手法、データ構造、メモリ・アクセス・パターン等を分析する。平成 23 年度は、提案するアーキテクチャが目標とする性能と消費電力を達成するために提供すべき並列処理能力、演算処理能力、及びメモリ・アクセス能力を定量的に示す。そして、ビジュアル・コンピューティング向けメニーコア・アーキテクチャを開発し、従来の組込みマルチコア方式(4 コア程度)と比較して性能 2 倍、消費電力量 1/10 が実現可能であることを簡易シミュレーションにより示す（ただし、ここでは小規模な評価用プログラムを用いる）。平成 24 年度は、開発したビジュアル・コンピューティング向けメニーコア・アーキテクチャについて総合評価を実施し、実アプリケーションにおいて性能 2 倍、消費電力量 1/10 を達成する。

Ａ－③大規模 FPGA を用いた動作検証（主担当：九州大学、副担当：トプスシステムズ）

テーマＡ－①ならびにＡ－②それぞれに関して、ハードウェア記述言語を用いて提案メニーコア・アーキテクチャの実設計を行い、大規模FPGAを用いたエミュレーションを実施する。これにより、提案するアーキテクチャの評価基盤を構築すると共に、実現可能性を実証する。平成22年度においてはFPGAを用いた動作環境の選定を行う。平成23年度では各構成要素(プロセッサコアやキャッシュ、ルータなど)、ならびに、周辺回路の実装を行い、

平成24年度にこれらを統合した動作検証を実施する。（計画からの変更・追加）なお、本研究テーマの実質的な主担当に関しては開発効率を考慮して電気通信大学として、副担当に東京農工大学も加えた（その他の副担当は九州大学とトプスシステムズ）。

Ｂ－①アクセラレーション部分抽出技術の開発（担当：フィックスターズ）

本研究項目では、ソースコードを入力とし、このアクセラレーション対象部分を決定する技術の確立、ならびに、そのツール化を行う。具体的には、C言語で記述された逐次ソースコードを入力とし並列化可能なアクセラレーション対象部分が出力として明示される。実施手順としてはメニーコアの有望分野でのユースケースを洗い出し典型的なソースコードをいくつか抽出する。これらを検証用コードとして技術開発を行う。実施期間は22年度と23年度とし24年度にはＢ－⑥のソフトウェア開発環境に統合する。年度目標としては、22年度にユースケース調査に基づいた検証用コードをＦ-①、Ｆ－②で挙げた4分野から最低2種類以上定める。23年度には研究開発を行い、本技術の有無により並列ソフトウェアの性能見積もりや開発工数が10%以上削減出来ることを目指す。（計画からの変更・追加）なお、今後のメニーコア向けソフトウェア開発環境として高速ソフトウェア・ライブラリの整備が重要であることが判明したため、この検証コードをライブラリへと発展させ実用化へと発展させた。

Ｂ－②仮想アクセラレータ合成技術の開発（担当：立命館大学）

アクセラレーション対象コードを抽出した後、当該コードを高効率に実行するための仮想アクセラレータ合成技術を開発する。平成22年度と平成23年度では、性能対エネルギー効率の高い仮想アクセラレータを合成する技術を開発する。平成24年度は、実装と評価を行い、評価結果をもとに合成技術を改良する。これに加え、総合評価を行い、実アプリケーションを対象とした場合においても性能2倍を達成可能であることを示す。

Ｂ－③仮想アクセラレータマッピング技術の開発（担当：立命館大学）

合成した仮想アクセラレータをメニーコア・ドメイン上に実装するためのマッピング技術を確立する。それと同時に、メニーコア・ドメインの構成を決定するための構成情報の自動生成を可能にする。平成23年度は、単一のアプリケーションが実行されているという仮定のもとで、仮想アクセラレータをメニーコア・ドメイン上に実装するマッピング技術を開発する。平成24年度は、複数のアプリケーションが同時に実行されるという、より現実的な仮定のもとで、仮想アクセラレータをメニーコア・ドメイン上に実装するマッピング技術を開発する。また、実装と評価を行い、評価結果をもとにマッピング技術を改良する。これに加え、総合評価を行い、実アプリケーションを対象とした場合においても性能2倍を達成可能であることを示す。

Ｂ－④仮想アクセラレータ向け自動並列化技術の開発（主担当：立命館大学、副担当：フィックスターズ）

従来の自動並列化コンパイラ技術が均質なマルチプロセッサを想定しているのに対し、本研究が対象としているメニーコア・ドメインは、仮想アクセラレータの数、仮想アクセラレータ当たりの小規模コア数、などを変更することができる。そのため、様々なレベルで並列性を抽出する仮想アクセラレータ向け自動並列化技術を開発する。これにより、仮想アクセラレータの利点を最大限活用することができ、より広い探索空間での性能対エネルギーの最適化が可能となる。平成23年度は、細粒度並列化技術と粗粒度並列化技術を個別に開発する。平成24年度は、実装と評価を行い、評価結果をもとに並列化技術を改良する。（計画からの変更・追加）なお、本テーマに関しては、テーマＢ－⑤にて、ユーザとのインタラクションを考慮した半自動コンパイル環境が有用である（並列化も自動ではなくユーザとのインタラクションによる半自動化）と判断したこと、また、プログラミング言語としてOpenCLを選定したことから、自動並列化技術の開発は中止し、「組込みメニーコア向けOpenCLランタイムライブラリ開発」とした。

Ｂ－⑤仮想アクセラレータ向けコンパイラの実装（担当：フィックスターズ）

仮想アクセラレータ向けコンパイラを実装する。具体的には、経済産業省 委託研究事業「平成 20 年度 IT 投資効率向上のための共通基盤開発プロジェクト」のテーマ「マルチコアプラットフォーム上での開発環境整備」としてフィックスターズと産総研が共同で開発した自動並列化コンパイラ CTRUMP をベースとし、本事業で確立した各種要素技術を組込む。C 言語で記述された逐次ソースコードと並列可能部分の情報を入力とし OpenCL 言語で記述された並列化コードが出力される。23 年度は変換機能と OpenCL コードのライブラリを作成する。24 年度にはＢ－⑥の成果を組込み、インタフェースも含めてソフトウェア開発ツールとして完成させる。

Ｂ－⑥アプリケーション・チューニング・ツールの開発（担当：フィックスターズ）

メニーコアのエネルギー効率を最大化するためには、プログラマによるアルゴリズムレベルの最適化や対象メニーコアのマイクロアーキテクチャに特化した最適化が必要不可欠となる。そこで本研究項目では、逐次コードを並列化コードに移植した際に実現される性能を大まかに予測するツールを開発し、プログラム開発者がその性能指標に基づいてアプリケーションをチューニングできる開発環境を構築する。具体的には、実行バイナリを入力とし予測性能といったプロファイルを出力する。23年度にはAPI仕様を受けて詳細設計を行う。24年度には機能を実装してツールとして実現する。

Ｃ－①製造ばらつきを考慮した仮想アクセラレータ制御と電源電圧制御技術の開発（主担当：電気通信大学、副担当：九州大学、立命館大学）

製造ばらつき特性情報を用い、各コアの性能差や動作不良の有無を意識した制御、およびきめ細かな電源電圧制御を行うことで低電圧動作を保証する新しいコンパイル技術を開発する。例えば、各コア単位（もしくは複数コアで構成するクラスタ単位）で動作可能な最大周波数情報（あるいは最低電源電圧情報）をチップ製造後に入手し、それに基づき性能・電力効率が最大になるように仮想アクセラレータを制御する。さらにクラスタ単位で小規模コアの動作周波数・電源電圧を最適化し、極低電圧領域での安定したメニーコア実行を可能にする。平成22年度は、極低電圧領域でLSIを動作させた場合の性能と消費電力、およびトランジスタの信頼性（動作不良の発生率など）に関する各種パラメータについて調査を行う。平成23年度は、前年度で得られたパラメータをベースとし、製造ばらつきを前提とした仮想アクセラレータ制御技術、およびきめ細かな電源電圧制御技術のアルゴリズムを開発する。平成24年度は、実アプリケーションを対象とした場合での消費電力量削減を示す。

Ｃ－②複数仮想アクセラレータの多重実行に基づくリーク消費電力削減技術の開発（担当：電気通信大学）

メニーコア・ドメインにて複数の仮想アクセラレータを多重実行可能であるという特徴を利用した瞬時間欠実行制御技術を開発する。複数タスクを性能が担保できる範囲で多重化してマッピングすることで瞬時に実行し、早期に、かつ、長期間システムを待機状態にする。そして、電源供給の停止など、待機状態における徹底した低消費電力化を施すことで、リーク消費電力の大幅な削減を可能にする。平成22年度は、既存のマルチコアプロセッサ環境上でいくつかの並列アプリケーションを動作させ、プログラム実行時のリソース利用率を測定することで、複数タスクの多重化実行の可能性を調査する。平成23年度は、評価環境上で複数並列アプリケーションを多重化して実行しつつ、実際の性能への影響と電力削減効果を調べ、そのデータを基に効率的な瞬時間欠実行制御のアルゴリズムを開発する。平成24年度は、本プロジェクトで開発する評価環境を利用して性能と電力を評価しつつ、アルゴリズムの改良を行う。

Ｄ－①高レベル API の策定（主担当：九州大学、副担当：立命館大学、電気通信大学、フィックスターズ）

プログラムとコンパイラの間で情報をやり取りするための API を策定する。これは、プログラマによる並列化に関する指示や制約条件に関する指示（性能や消費電力など）、アルゴリズム・レベルでの最適化をより効率的にアーキテクチャに反映するための指示、などが考えられる。このような高レベル API を定めることにより、プログラマはハードウェアの詳細を意識することなくプログラム開発に注力できる。平成 22 年度は、高レベル API 策定

に向け、並列プログラム開発者へのヒアリング等に基づく実態調査を実施する。平成 23 年度には、協力企業/大学/研究所を募り、様々な観点から議論を重ね「高レベル API 第 1 版」を策定する。平成 24 年度においては、外部からのフィードバックに基づき改善を重ね、「高レベル API 最終版」を策定する。(計画からの変更・追加) なお、本研究テーマに関しては、平成 22 年度のヒアリング調査等の結果より、独自の API を新たに策定するのではなく、ロイヤルティフリーでオープンな標準である OpenCL をベースとすることに決定した。しかし、現在の OpenCL 仕様は、高いリアルタイム性が要求される組込みシステムには適していないため、独自の制約と解釈を与えることとし、重要 API について実装することとした。

Ｄ－②低レベル API の策定（主担当：九州大学、副担当：立命館大学、電気通信大学、トプスシステムズ）

コンパイラとハードウェアの間で情報をやり取りするためのAPIを策定する。仮想アクセラレータ（VAM）のマッピング制御、ハードウェア構成選択に関する指示などを可能にする。平成22年度は、低レベルAPI策定に向け、回路設計者や半導体開発者、極低電圧回路を専門とする研究者へのヒアリング等に基づく実態調査を実施する。平成23年度には、協力企業/大学/研究所を募り、様々な観点から議論を重ね「低レベルAPI第1版」を策定する。平成24年度においては、外部からのフィードバックに基づき改善を重ね、「低レベルAPI最終版」を策定する。

Ｅ－①低電圧メニーコア実行を想定した消費電力モデルの開発（担当：トプスシステムズ）

提案するアーキテクチャの性能と消費電力をシミュレーションにより定量的に評価するために消費電力モデルを開発する。平成22年度は、提案するメニーコア・アーキテクチャの性能と消費電力の評価に適切な消費電力モデルの要件を策定する。先ず、本提案のメニーコア・アーキテクチャに基づくシステム・シミュレーションの構成と規模、及び各構成要素の消費電力の概算値を算出する。概算値に基づいて、電力パラメータ(実装に使用する半導体テクノロジーに依存する電気的特性)、シミュレーション速度、消費電力見積り精度の要件を定義する。平成23年度は、シミュレーション速度と消費電力精度のトレードオフを考慮しながら、システム構成要素であるソフトウェア、プロセッサ・コア、メモリ、通信ネットワーク等について、計算モデルとしての粒度と抽象度、通信モデルとしての粒度と抽象度（TLM、概算時間レベル、サイクル・アキュレイト）を検討する。また、その結果を消費電力モデルの仕様を纏める。そして、仕様に基づいて消費電力モデルを作成する。平成24年度は、作成した消費電力モデルは、システム構成例を用いてその妥当性を検証する。

Ｅ－②高速メニーコア・シミュレータの開発（担当：トプスシステムズ）

提案する低消費電力メニーコア・アーキテクチャの有効性を容易に評価可能な環境として高速のメニーコア・シミュレータを開発する。平成 23 年度は高速シミュレータの要件を定義する。そして、高速シミュレータの仕様策定と、シミュレータ・カーネルの実装を進め、E-①の消費電力モデルをシミュレータに組込み、基本機能を確認する。平成 24 年度は、シミュレーション速度を評価する。さらに、シミュレータ・カーネルに機能を追加し、各種表示機能を実装することで、性能と消費電力のボトルネック解析を可能にするハードウェア開発支援環境として完成させる。

Ｅ－③提案メニーコア実行方式の総合評価（担当：トプスシステムズ）

提案するメニーコア・アーキテクチャの有効性を定量的なシミュレーションによる分析結果に基づいて各テーマの成果に関して総合的に評価する。平成 23 年度は、提案するメニーコア・アーキテクチャの有効性を示すための、性能/消費電力の評価指標の詳細、評価対象アプリケーションと評価対象とする処理範囲、評価条件（半導体プロセスなど）、及び評価方法を決める。平成 24 年度は、総合評価の条件に従って、各テーマの成果に関する総合性能/消費電力評価を実施する。性能/消費電力評価は、早期に各テーマにフィードバックすることで、技術的な課題とその解決を進め、最終的に、同一プロセス、同一面積制約において、従来の組込みマルチコア方式(4コア程度)と比較して、性能 2 倍以上、消費エネルギー1/10 以下を実現できることを示す。

Ｆ－①メニーコア適用市場と要求仕様に関する調査研究：ライフサイエンス、ネットアプリ、ファイナンスの 3 分野（担当：フィックスターズ）

ライフノベーション分野（医療画像処理、検査装置、ロボットなど）、インターネット・アプリケーション分野（Webブラウジングなど）、ファイナンス分野（金融商品の価格付けやリスク計算など）を対象とし、特に、本事業で開発するメニーコア技術に焦点を絞り、その適用市場ならびに要求仕様を調査する。具体的には、本調査で得られる要求仕様は、本研究開発で実現するソフトウェア開発環境を設計する上での重要なベースとなる。特に各分野での典型的な処理はＢ-①で利用する検証用コードとして実際に評価する。調査研究の実施期間は22年度から24年度とする。22年度は外部機関からの調査報告書に基づき具体的な処理内容まで深く調査する。23年度と24年度は本研究開発で対象としなかったような潜在市場の調査や各市場への普及を含めて調査研究する。

Ｆ－②メニーコア適用市場と要求仕様に関する調査研究：動画像処理分野（担当：トプスシステムズ）

膨大な画像情報を実時間処理することで知的要求や価値創出が期待されるビジュアル・コンピューティング分野は、アルゴリズムに内在する様々な並列性が高く、従来技術に対し消費電力を増やさずに高性能化が可能なメニーコア技術（アーキテクチャやコンパイラ）

が期待されている。そこで、ビジュアル・コンピューティング分野（画像の合成、認識、マイニングの 3 つのモデルに基づくアプリケーション)、特に本事業でターゲット・アプリケーションとしているビデオ・マイニングについて、本事業で開発するメニーコア技術に焦点を絞り、その応用展開の適用市場を調査する。平成 22 年度と平成 23 年度は、本研究開発成果の低消費電力メニーコア IP の事業化と普及を前提に「ビジュアル・コンピューティング分野（ビデオマイニングなど）における」本研究開発の仮想アクセラレータ実行プラットフォームとしての低消費電力メニーコア・アーキテクチャの有望な適用市場(要件、既存のソリューション、技術課題、応用分野の多様性、ユーザ等)の調査分析を行う。調査の方法として、低消費電力メニーコアに対する期待と要求仕様に留まらず、ユーザ候補企業から本研究開発に対するフィードバックを得る。そして、調査結果として得られた要件、技術課題、応用分野の多様性等を「ビジュアル・コンピューティング分野における低消費電力メニーコア適用市場調査結果」として纏め、各研究テーマにフィードバックする。平成 24 年度は、各企業に対する市場調査に加えて、ユーザ候補企業に対する本研究開発成果に基づいて実現可能なビジュアル・コンピューティング（特にビデオ・マイニング）システムを提案し、その結果を「ビジュアル・コンピューティング分野における本研究開発成果の低消費電力メニーコア適用市場調査結果」として纏める。

Ｇ－①マルチ・メニーコア技術適用アプリケーション拡大に向けた市場の調査・検討（担当：JEITA）

マルチ・メニーコア技術を適用するアプリケーションの拡大に向けた市場の調査・検討では、マルチ・メニーコアチップの有望な適用分野、機器市場を調査・分析し、日本の半導体産業の競争力強化に繋がる提言(、特に新規プロジェクト提言)をまとめる。マルチ・メニーコア技術のニーズは幅広く、組込み機器全般に及ぶと考えられるが、市場規模、成長性、日本の強み分野の観点から、日本の半導体産業が力を入れるべき分野にフォーカスして調査する。また、マルチ・メニーコア技術はワールドワイドに多くの企業・大学・研究機関が研究開発を進めており、技術革新の早い分野であるため、適切な提言に向けて最新の技術動向を把握する。

Ｇ－②開発環境を中心としたマルチ・メニーコア普及促進に向けた調査・検討（担当：キャッツ株式会社）

開発環境を中心としたマルチ・メニーコア普及促進に向けた調査・検討では、並列化コンパイラをはじめとするマルチ・メニーコアで動作するシステム開発のために有効な開発環境について調査する。また、マルチ・メニーコアの普及と日本の半導体産業の競争力強化に繋がる提言をまとめる。マルチ・メニーコアを活用することで、製品の低消費電力化が見込まれるが、そこで動作するソフトウェアには同時に複数コアの効果的な利用と、システムとしての性能向上、そして安全性が求められる。これらを満たすための開発環境に

はどの様な要件が必要であるかを明らかにする必要がある。この様な開発環境にフォーカスした専門的な調査を、JEITAによる市場調査を踏まえて行い、検討する。

Ｇ－③動作環境を中心としたマルチ・メニーコア普及促進に向けた調査・検討（イーソル株式会社）

動作環境を中心としたマルチ・メニーコア普及促進に向けた調査・検討では、オペレーティングシステムを中心としたマルチ・メニーコアの動作環境において調査を行う。既存の非マルチ・メニーコア向けソフトウェアの再利用性、コア数の急激な増加が見込まれる上でのスケーラビリティなどをはじめとした要件を調査、検討する。

（4）実施計画

	2010 年度(H22 年度)				2011 年度(H23 年度)				2012 年度(H24 年度)			
	第1四半期	第2四半期	第3四半期	第4四半期	第1四半期	第2四半期	第3四半期	第4四半期	第1四半期	第2四半期	第3四半期	第4四半期
A－①汎用メニーコア･アーキテクチャの研究開発				→	→	→	→	→	→	→	→	
A－②ビジュアル･コンピューティング･メニーコア･アーキテクチャの研究開発				→	→	→	→	→	→	→	→	
A－③大規模 FPGA を用いた動作検証				→	→	→	→	→	→	→	→	→
B－①アクセラレーション部分抽出技術の開発				→	→	→	→	→				
B－②仮想アクセラレータ合成技術の開発				→	→	→	→	→	→	→	→	
B－③仮想アクセラレータマッピング技術の開発					→	→	→	→	→	→	→	
B－④仮想アクセラレータ向け自動並列化技術の開発					→	→	→	→	→	→	→	
B－⑤仮想アクセラレータ向けコンパイラの実装					→	→	→	→	→	→	→	→
B－⑥アプリケーション･チューニング･ツールの開発					→	→	→	→	→	→	→	→
C－①製造ばらつきを考慮した仮想アクセラレータ･マッピングと電源電圧制御技術の開発				→	→	→	→	→	→	→	→	
C－②複数仮想アクセラレータの多重実行に基づくリーク消費電力削減技術の開発				→	→	→	→	→	→	→	→	
D－①高レベル API の策定				→	→	→	→	→	→	→	→	
D－②低レベル API の策定				→	→	→	→	→	→	→	→	
E－①　低電圧メニーコア実行を想定した消費電力モデルの開発				→	→	→	→	→	→	→		
E－②高速メニーコア･シミュレータの開発					→	→	→	→	→	→	→	
E－③提案メニーコア実行方式の総合評価						→	→	→	→	→	→	→
F－①メニーコア適用市場と要求仕様に関する調査研究:ライフサイエンス､ネットアプリ､ファイナンスの 3 分野				→	→	→	→	→	→	→	→	→
F－②メニーコア適用市場と要求仕様に関する調査研究:動画像処理分野				→	→	→	→	→	→	→	→	→

	2010 年度(H22 年度)				2011 年度(H23 年度)				2012 年度(H24 年度)			
	第1四半期	第2四半期	第3四半期	第4四半期	第1四半期	第2四半期	第3四半期	第4四半期	第1四半期	第2四半期	第3四半期	第4四半期
G－①1．マルチ・メニーコア技術を適用するアプリケーションの拡大に向けた調査・検討					→	→	→	→	→	→	→	→
G－②1．開発環境を中心としたマルチ･メニーコア普及促進に向けた調査･検討					→	→	→	→				
G－③1．動作環境を中心としたマルチ･メニーコア普及促進に向けた調査･検討					→	→	→	→				