戦略的基盤技術高度化支援事業

「マルチコア環境における組込みソフトウェア設計ツールの開発」

研究開発成果等報告書

平成 24 年 3 月

委託者：九州経済産業局

受託者：財団法人福岡県産業・科学技術振興財団

【目次】

【図目次】

【表目次】

1. 研究開発の概要

1.1. 研究開発の背景

組込みプロセッサとソフトウェアから成る組込みシステムは、日常生活のあらゆる場面で使用されている。携帯電話やデジタル家電に多数の組込みシステムが使用されているのはもちろん、機械製品と考えられている自動車にも３０～３００個程度の組込みプロセッサが搭載されている。

これまで、コンピューターの世界では、高機能化、高速化というニーズに対して、プロセッサの動作周波数を上げるという手法で、これに応えてきた。しかし、この手法は、すでに限界に近づいており、周波数の増大によって、消費電力増大、発熱、ノイズ発生といった問題が生じることとなった。そのため、現在では、プロセッサのコアを複数にすることにより、高機能化、高速化のニーズに応える手法が有望視されている。

パソコンのプロセッサは、既に２コアを経て４コア時代へと突入しており、組込みシステムのプロセッサにおいてもマルチコア化が進みつつある状況となっている。

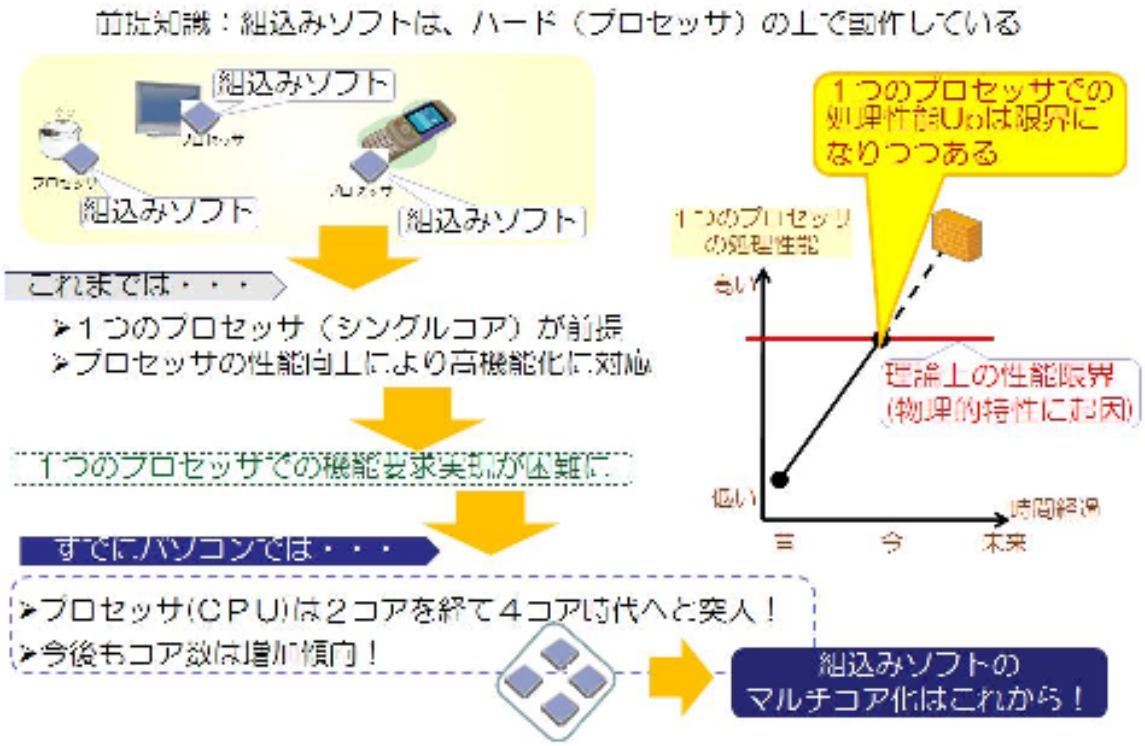


図 1 研究開発の背景(プロセッサのマルチコア化の状況)

他方、組込みシステムの開発においては、「リアルタイム性の保証」や「低消費電力化」など、組込特有の課題を抱えている。

例えば、車載ブレーキシステムである ABS については、遅延・誤作動すると、スリップして大事故になってしまうため、厳密なリアルタイム性が求められる。

また、バッテリー駆動のシステムにおいては、消費電力量に応じてシステムの持続時間が変わる為、可能な限りの低消費電力化が求められている。

しかし、マルチコア環境においては、これらの課題への効果的な対応方法が、未だ確立していない。

さらに、組込みシステムで、「リアルタイム性の保証」や「低消費電力化」等の課題を考慮しながら設計・開発を効率的に行う手段についても、現在は存在していない。

表 1 組込みシステムにおいてマルチコア対応を行う為の課題

番号	課題
課題１	設計工程でのリアルタイム性の保証
課題２	低消費電力の実現
課題３	開発の効率化

1.2. 研究組織および体制

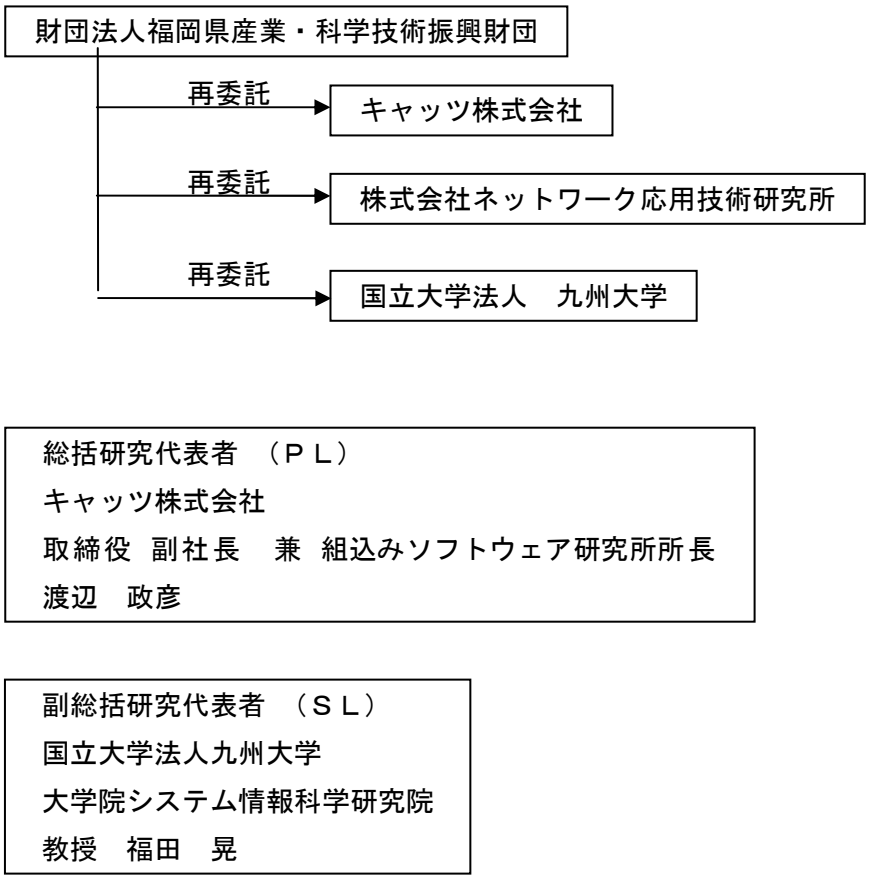


図 2 研究組織(全体)

2. 課題１：リアルタイム性の保証

2.1. 研究目的（１）

組込みシステムは、リアルタイム性が重要であるため、オペレーティングシステム（OS）は、リアルタイム OS が使用されている。

しかし、マルチコアに対応したリアルタイム OS でリアルタイム性を保証するには、動作時に自動的にコア配置が行われるため、全てのコア配置でリアルタイム制の検証を行う必要がある。

そこで本研究開発は、リアルタイム性の保証を実現することを目的として、マルチコア環境において、最適なコア配置を解析・特定し、その最適なコア配置にて動作させることを実現する。

具体的な数値目標としては、リアルタイム OS によるコア配置と比較して、リアルタイム性を従来比 2/3 にする事を目指す。

実際の手法としては、シングルコアで動作したログ情報を入力する事により、タスクが効果的に動作出来るコア配置を解析するツールを作成・実行して、その有効性を評価する。

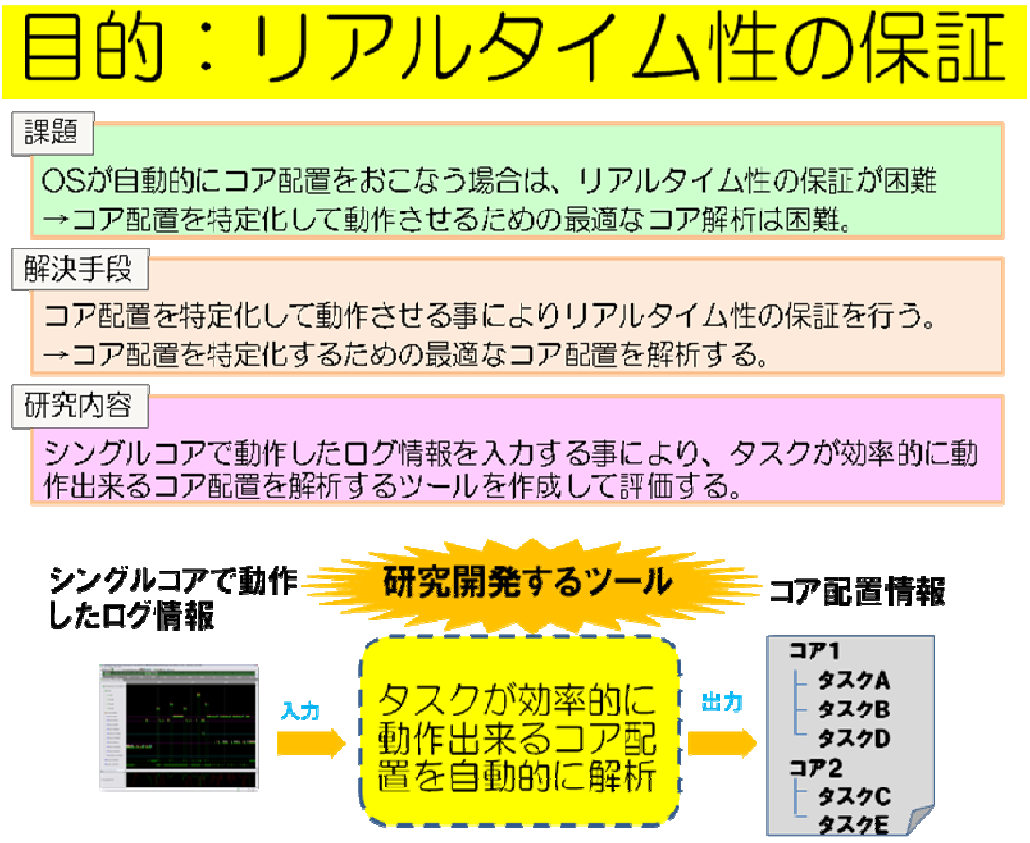


図 3 課題１：リアルタイム性の保証

2.2. 課題詳細

組込みシステムの開発においては、リアルタイム性の保証という特有の課題を抱えている。リアルタイム性とは、決められた時間内に処理を完了させることである。

例えば、車載ブレーキシステムである ABS については、遅延・誤作動すると、スリップして大事故になってしまうため、厳密なリアルタイム性が求められる。

図 4 リアルタイム性の必要性

上述のように、組込みシステムではリアルタイム性が重要であるため、オペレーティングシステム（OS）にも、リアルタイム OS が使用されている。

しかし、マルチコアに対応したリアルタイム OS でリアルタイム性を保証するには、動作時に自動的にコア配置が行われるため、全てのコア配置でリアルタイム制の検証を行う必要がある。

例えば、２コア環境において、タスク３０個のシステムにおけるコア配置は、１０億通り以上のパターンがあるが、この全てのコア配置で、リアルタイム性を保証することは現実的でない。

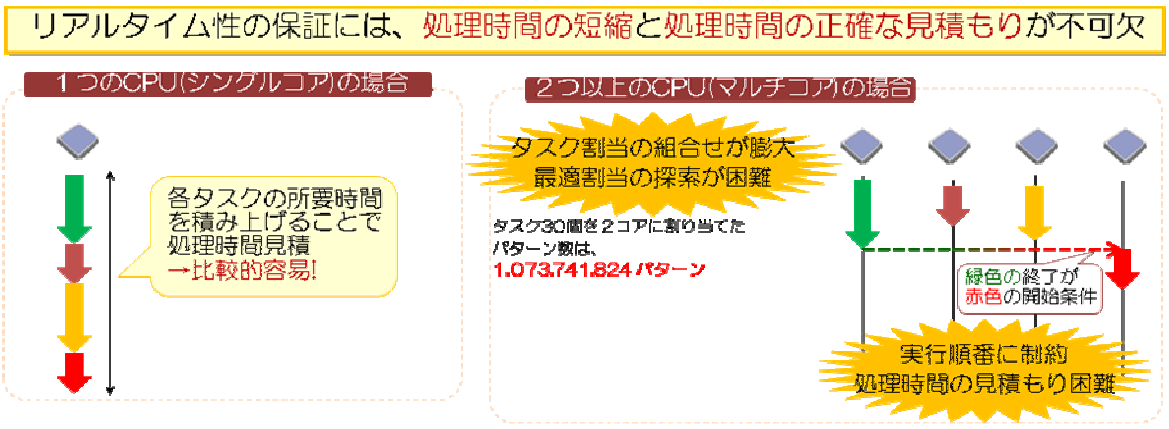


図 4 リアルタイム性の課題

2.3. 解決方法

リアルタイム OS が自動的にコア配置を行う環境は、全てのコア配置においてリアルタイム性の保証をする事は困難である。

しかし、最適なコア配置を解析・特定し、その特定された最適な配置で動作するようにできれば、リアルタイム性の保証が可能である。

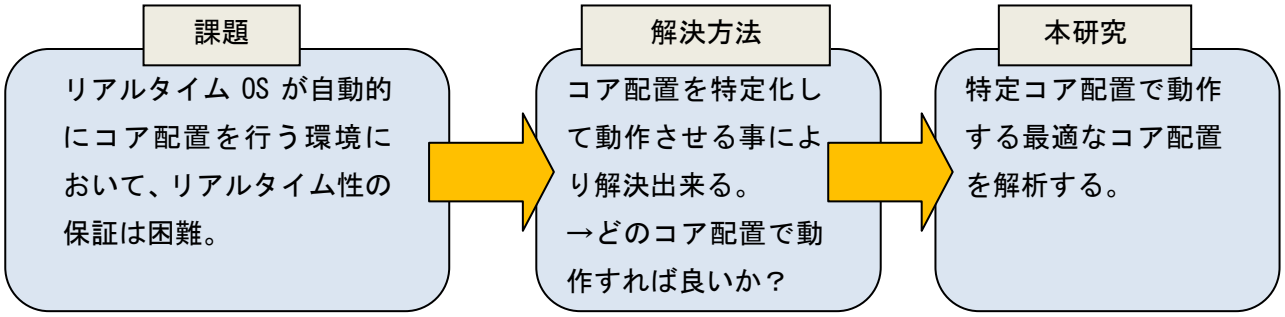


図 5 リアルタイム性の保証における課題解決方法

このため、本研究開発では、まず最適なコア配置を解析するために、シングルコアで動作したログ情報を入力として、「タスクのスループット」、「コアの負荷バランス」、「コア間通信」の３項目を考慮し、タスクの効率的なコア配置を解析するツールを開発する。

ただし、先にも述べた通り、天文学的な数字が存在する全てのコア配置パターンを解析する事は現実的に不可能である。

このため、解析の際は、最適に近い解を短時間で求める手法の一つである SA（シミュレーティド・アニーリング）法（焼きなまし法とも呼ばれる）を使用する。

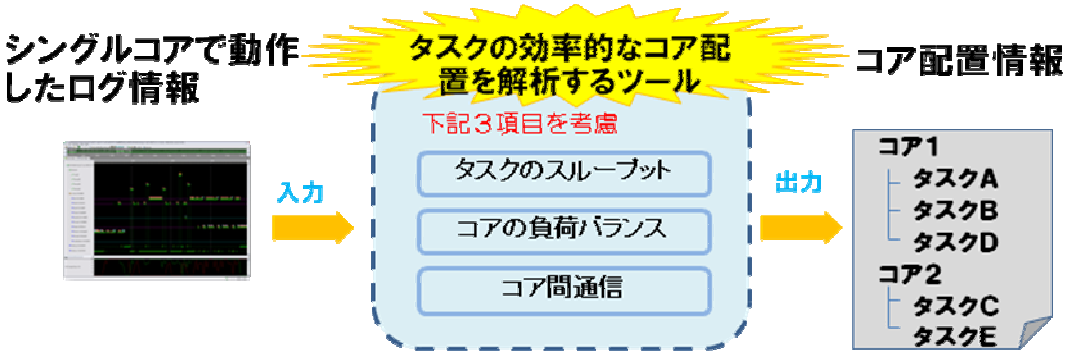


図 6 リアルタイム性の保証における研究手段

2.4. 研究結果と評価

今回の目的は、リアルタイム性が保証する事であるが、そもそもマルチコア化には、性能向上の目的も含まれている。このため、リアルタイム OS と比較して、タスク毎の動作可能(Ready)時間、タスク処理(Running)時間を 2/3 時間にする事を目標値に設定し、研究開発を行った。

その結果、タスク毎の動作可能(Ready)時間については、平均 82%の動作時間に短縮した。また、17 個のタスクの中で、9 個のタスクが高速化を達成しており、6 個のタスクが目標値(2/3)を達成した。

タスク処理 (Running)時間については、平均 85%の動作時間に短縮した。また、17 個の全てのタスクが高速化を達成しており、3 個のタスクが目標値(2/3)を達成した。

図 7 リアルタイム性の保証における評価結果

3. 課題２：低消費電力の実現

3.1. 研究目的（２）

マルチコアプロセッサには、CPU コアが複数含まれるが、CPU コアを複数にすると、消費電力も比例して増加する。マルチコア化には、電圧を下げて低消費電力化が図られるという効果もあるが、一層の消費電力低減が求められている。

そこで、マルチコア環境における低消費電力化の実現を目的として、ソフトウェアの処理が必要とされない時間に、プロセッサを低消費電力モードに切り替える方法を研究開発する。

具体的な数値目標としては、消費電力を従来比 2/3 にする事を目指す。

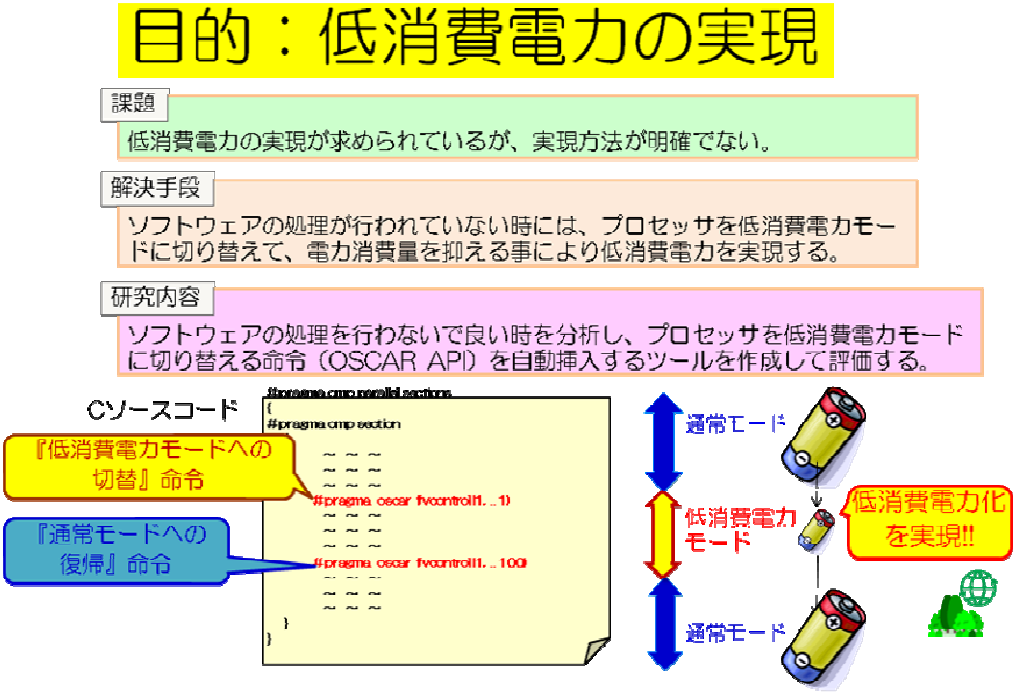


図 8 課題２：低消費電力の実現

3.2. 課題詳細

バッテリー駆動のシステムでは、消費電力量に応じてシステムの持続時間が変わるため、低消費電力化が求められる。

しかも、マルチコアプロセッサでは、コアが複数になるため、その分消費電力量が多くなり、シングルコアのプロセッサよりも一層の低消費電力化が要求される。

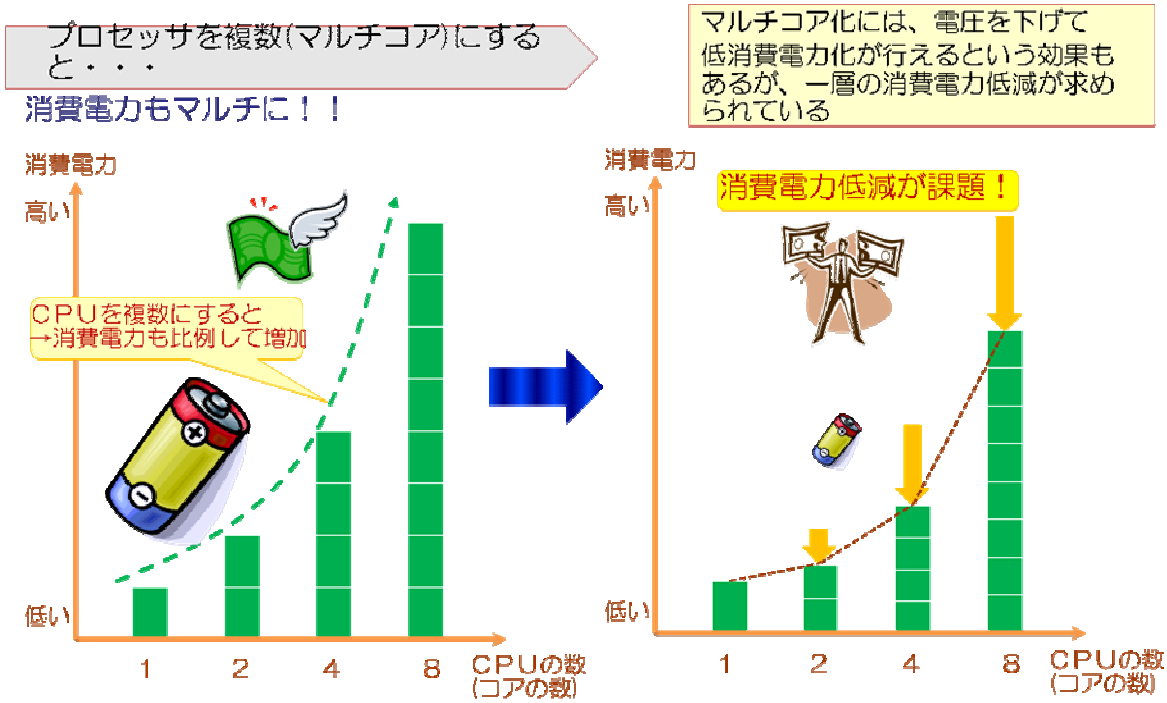


図 9 低消費電力化の実現における課題

3.3. 解決方法

通常のプロセッサには、通常モードと、低消費電力モードが存在している。そこで、本研究開発では、ソフトウェアの処理が行われていない時に、プロセッサを低消費電力モードに切り替えることで電力消費量を抑える。

なお、プロセッサを低消費電力モードに切り替える命令は、デバイス毎で個別に存在しているが、今後標準化された API として「OSCAR API」がリリースされる予定であるため、本機能の開発においては OSCAR API 命令に対応するものとし、評価の際も OSCAR API を活用する。

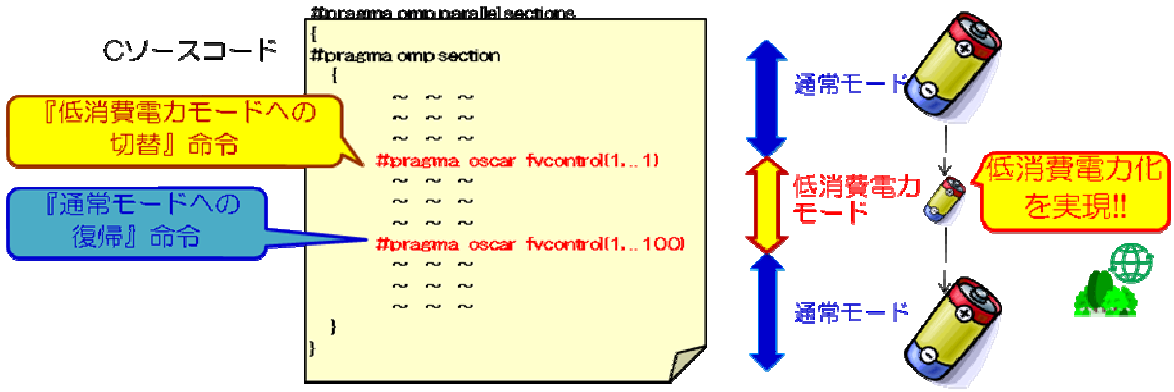


図 10 低消費電力化の実現における課題解決方法

3.4. 研究詳細

状態遷移表を使用する設計ツール（ZIPC）を利用して、マルチコア環境における低消費電力の設計を実現する。

本研究開発では、ZIPC がソースコードを自動生成する際に、ソフトウェアの処理が行われない部分（待ち合わせ箇所等）を解析し、プロセッサを低消費電力モードに切り替える命令（OSCAR API）を自動挿入するツールを開発する。

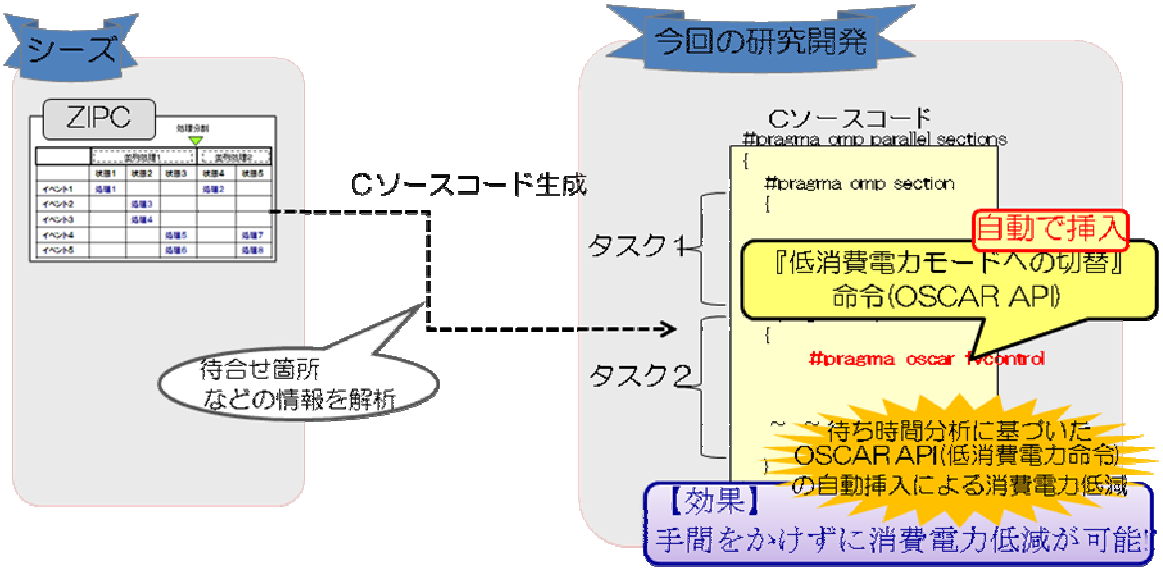


図 11 低消費電力化の実現における研究手段

3.5. 研究結果と評価

プロトタイプ版の開発を完了した。

ただし、OSCAR API の標準化が遅れていることや、半導体メーカー不況の影響から、現時点では OSCAR API に対応したデバイスが存在していない。

よって、現状でこの機能を評価することが不可能であるため、OSCAR API に対応したデバイスがリリースされしだい評価を行うものとする。

4. 課題３：開発の効率化

4.1. 研究目的（３）

現在、マルチコア対応の並列化命令として、OpenMP が標準化され、使用されている。

しかし、マルチコアの開発では、この OpenMP コードを挿入する作業が増え、シングルコアの開発に比べて開発効率が悪くなっている。

そこで、開発効率の向上を目的として、並列化設計部分について、マルチコア対応の並列化命令（OpenMP）を自動挿入する方法を研究開発する。

実際の研究手法としては、、ZIPC 状態遷移モデルの設計書から、C ソースコードを自動生成する際に、マルチコア対応の並列化命令（OpenMP）を自動挿入するツールを作成し、この方法の有効性を評価する。

具体的な数値目標としては、従来比 2 倍の開発効率となる事を目指す。

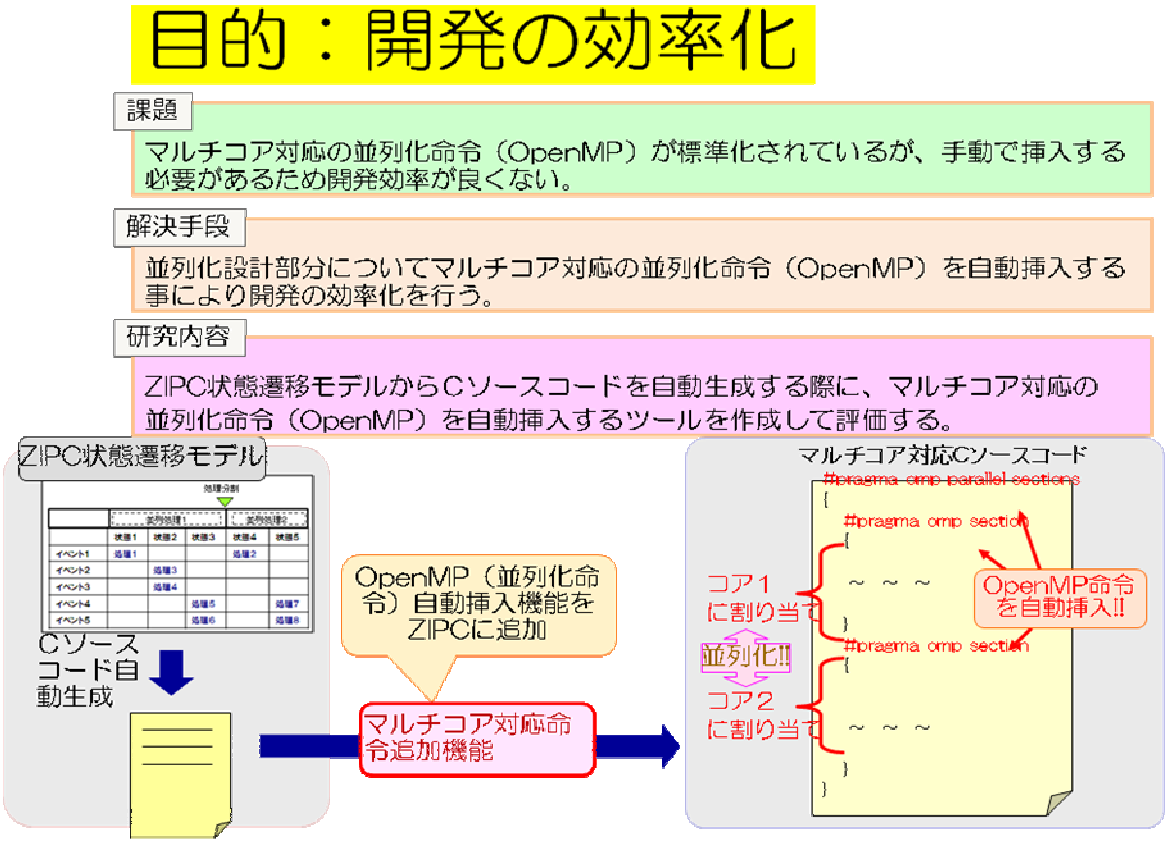


図 12 課題３：開発の効率化

4.2. 課題詳細

組込ソフトウェアにおけるマルチコアの開発現場では、経験と勘を頼りにして、タスク割り当て用命令を記述し、実行／シミュレーションを行う。そして、実行結果を基にして、変更方法を検討するといった事を繰り返し実行している。しかしながら、このような繰り返し作業を手作業で実施するのは、大変非効率なだけれなく、設計漏れの危険性（リスク）も伴う。

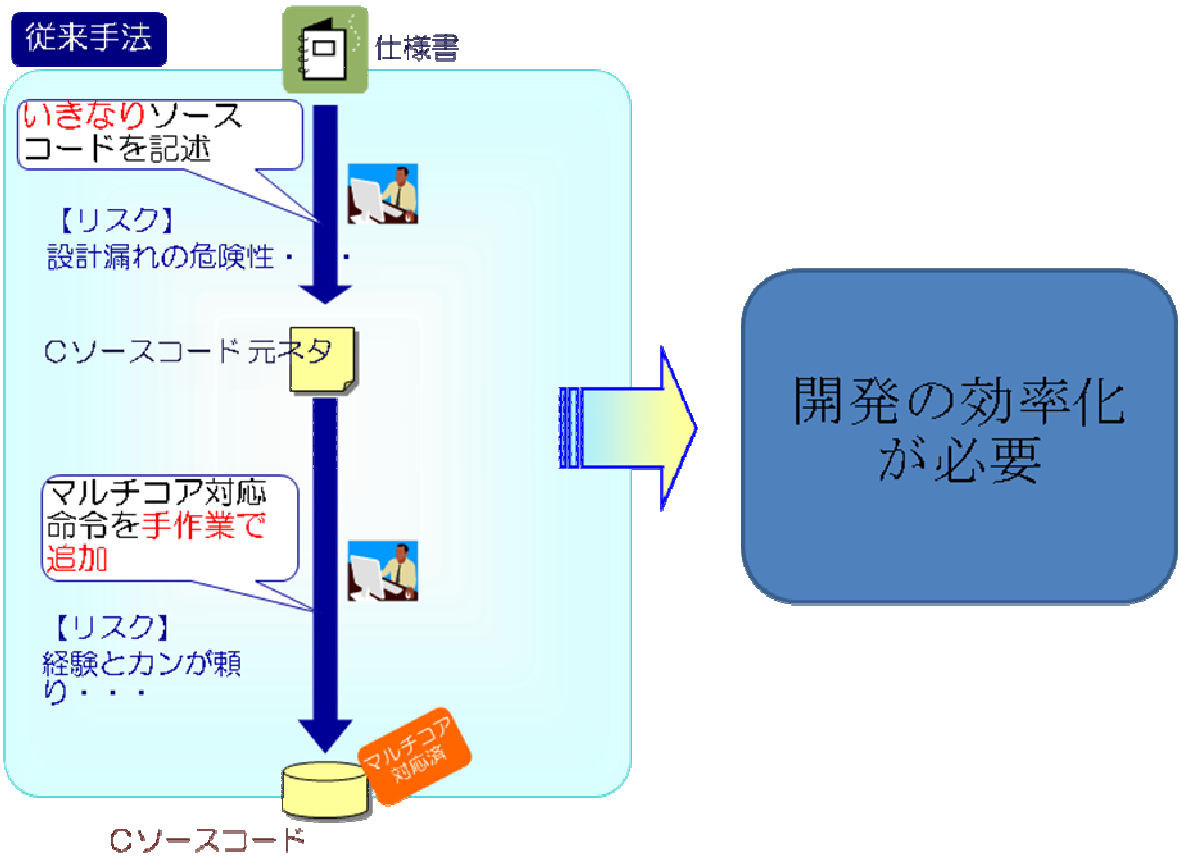


図 13 開発の効率化における課題

4.3. 解決方法

そこで、状態遷移表モデルを使用した設計ツール（ZIPC）に、マルチコア対応命令（OpenMP）の自動挿入機能を追加し、自動的に、マルチコア対応命令（OpenMP）が追加されるようにする。

これにより、マルチコア環境における開発効率が大幅に向上する。

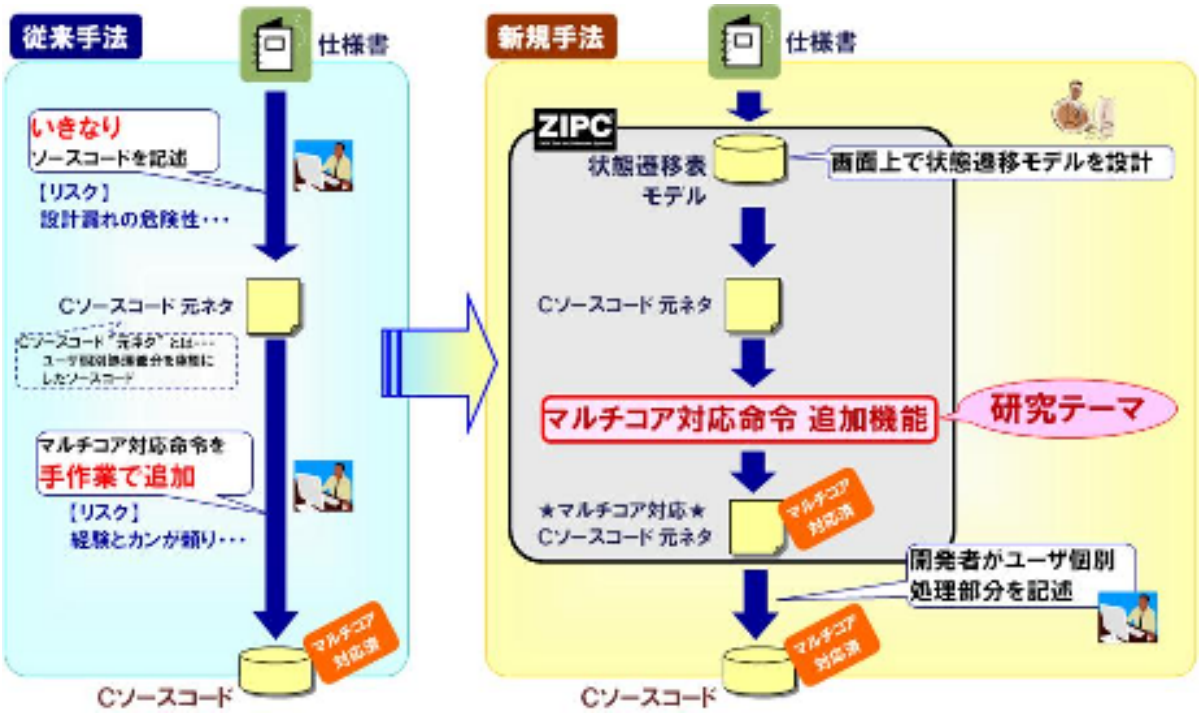


図 14 開発の効率化における課題解決方法

4.4. 研究詳細

ZIPC は、状態遷移表によりモデルベース設計が行えるツールであり、状態遷移表からＣ言語ソースコードを自動生成する事も可能である。

さらに、状態遷移表モデルを使用して設計する際に、並列で処理を行う状態に関しては、「並列状態」として設計する仕様となっている。

しかし、状態遷移表の並列型設計を行った際でも、１つのスレッド（コア）に割り当て、逐次処理されているため、並列型状態標記によって、複数の状態は管理できるが、並列に処理されるわけではない。

そこで、本研究開発では、ZIPC から、マルチコア対応の命令（OpenMP）を自動挿入する機能を構築し、開発効率を向上させる。

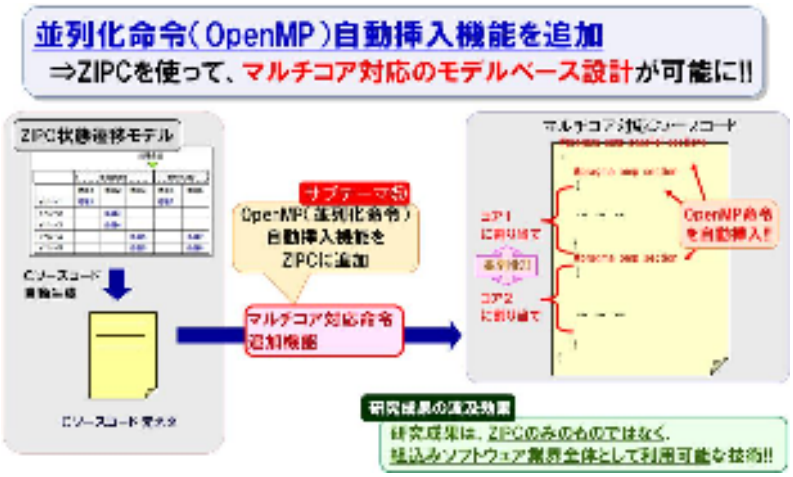


図 15 開発の効率化における研究手段

4.5. 研究結果と評価

ZIPC から並列型設計を行った際に、マルチコア対応の命令（OpenMP）を自動挿入する機能を構築した。これにより、ZIPC から並列型設計を行った際でも、複数スレッドで実行され、並列で処理される事となった。

さらに、以下の観点から評価を行った。

4.5.1. 開発効率の評価

本機能によりマルチコア対応の命令（OpenMP）を自動挿入した場合と、手作業で挿入した場合を比較して、開発効率の評価を行った。

手作業で自動挿入を行った際は、挿入する部分を検討時間と実際に挿入する時間も含めて、4.5 時間が必要であった。ツールを使用して自動生成した際は、約１分でマルチコア対応の命令（OpenMP）を挿入できた。

また、開発効率全体を評価した場合でも、ZIPC 等のモデルベース設計を導入した場合、従来型に比べて設計効率が向上する。シングルコア向け組込みソフトウェア設計における標準的な導入効果として、従来手作業で１９人月であったものが、ZIPC を使用する事により９．５人月となった事例がある。

従って、マルチコアプロセッサ向け組込みソフトウェアの設計をモデルベースで行えるようになった事及び、マルチコア対応の命令（OpenMP）を自動挿入できるようになった事により、本研究開発で目標とした従来比２倍の開発効率向上が実現できた。

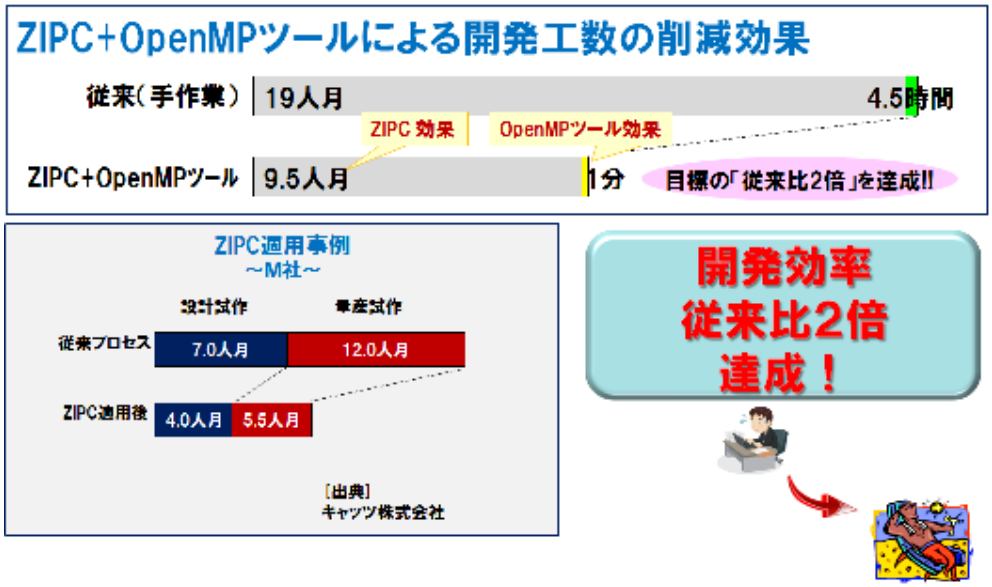


図 16 開発の効率化における評価結果①

4.5.2. 処理性能の評価

今回、組込みシステムにおいて、マルチコア対応命令（OpenMP）を使用して開発した事例が存在していなかったため、Windows 上で、OpenMP（並列化命令）を挿入した際の処理時間測定を行った。

測定ケース１では、従来 2828mSEC の処理が、OpenMP（並列化命令）を挿入した際に、1609ms となり、従来比 1.75 倍の速度向上となった。

測定ケース２では、従来 1257ms の処理が、OpenMP（並列化命令）を挿入した際に、1036mSEC となり、従来比 1.21 倍の速度向上となった。

よって、上記測定結果より１．２倍～１．８倍の処理速度向上効果があることを検証できた。

図 17 開発の効率化における評価結果②

4.6. 数値目標の達成度

【課題１】と【課題３】については、数値目標を達成した。

【課題２】についても、機能の開発は達成したが、OSCAR API 対応デバイスが現時点でリリースされておらず、評価については実施していない。

課題	数値目標	結果
【課題1】 設計工程でのリアルタイム性の保証	リアルタイム性 従来比 2/3 ※リアルタイムOSによる割当との比較	達成 2/3
【課題2】 低消費電力の実現	消費電力 従来比 2/3	OSCAR API対応デバイスが未リリースのため評価出来ず、未達成。
【課題3】 開発の効率化	開発効率 従来比 2倍 ※ソースコードベース設計との比較	達成 2倍

図 18 研究目標と達成度

5. 研究開発成果の事業化

課題１にて研究開発したツールは、「CoreRA®」という登録商標を取得し製品化を行った。

CoreRA®は、シングルコアで動作したログ情報からマルチコア環境におけるタスクの最適なコア配置を解析するツール（パッケージ製品）として 2011 年 3 月にリリース。

図 19 パッケージ製品「CoreRA®」

6. ユーザニーズ対応等

本研究開発の成果である「CoreRA®」のリリース後、ユーザへのヒアリング結果などから、各課題に関連したいくつかの研究テーマがあがってきたため、これをサブテーマと位置付け、対応する事とした。

表 2 サブテーマ毎におけるユーザニーズと課題のヒモ付け

課題対応	サブテーマ	ユーザニーズ概要
【課題１】 設計工程でのリアルタイム性の保証	**【サブテーマＡ】** タスク最適割り当てツール「CoreRA」のユーザニーズ対応	ログ計測装置が接続不可な環境での CoreRA の利用
【課題２】 低消費電力の実現	**【サブテーマＢ】** 消費電力低減のための待ち時間分析	待ち時間分析などタスク内部の詳細分析
【課題２】 低消費電力の実現	**【サブテーマＣ】** 消費電力低減のためのタスク割り当て結果参照	ターゲット環境の構築を必要としない、コア配置の確認

7. サブテーマ A：「CoreRA」のユーザニーズ対応

7.1. 目的と目標

本研究の目的は下記のとおり。

[目的]

ログ計測装置に接続不可な環境に対応して欲しいというニーズに対応する。

[目標]

ユーザニーズ対応の実現。

なお、各サブテーマの位置付けは下記の通りである。

表 3 サブテーマの位置付け

サブテーマ研究概要	目的	目標
【サブテーマA】 「CoreRA」のユーザニーズ対応	・CoreRA の利活用環境拡大による顧客増加	ログ計測装置が接続不可な環境で利用できるよう、CoreRA を機能拡張する。
【サブテーマB】 消費電力低減のための待ち時間分析	・更なる消費電力の低減	待ち時間分析などタスク内部の詳細分析と、CoreRA による低消費電力の効果を促進する
【サブテーマC】 消費電力低減のためのタスク割り当て結果参照	・更なる消費電力の低減 ・CoreRA のユーザニーズ対応による顧客増加	ターゲット環境の構築を必要としない、コア配置の確認方法の構築と、CoreRA による低消費電力の効果を促進する

7.2. 課題詳細

現状、eT-Kernel/eBinder を使用してログを計測するには、ログ計測装置を接続する必要がある。しかし、ログ計測装置は、ターゲットのアドレスバス／データバスの信号線に接続する必要がある。このため、アドレスバス／データバスに接続できないターゲット環境では、CoreRA®が使用出来ない。

ヒアリングを通じて、ユーザから、ログ計測装置が接続不可なターゲット環境でも CoreRA を使用したいとのニーズが出たため、サブテーマＡでは、ログ計測装置が接続不可なターゲット環境の対応を実現する。

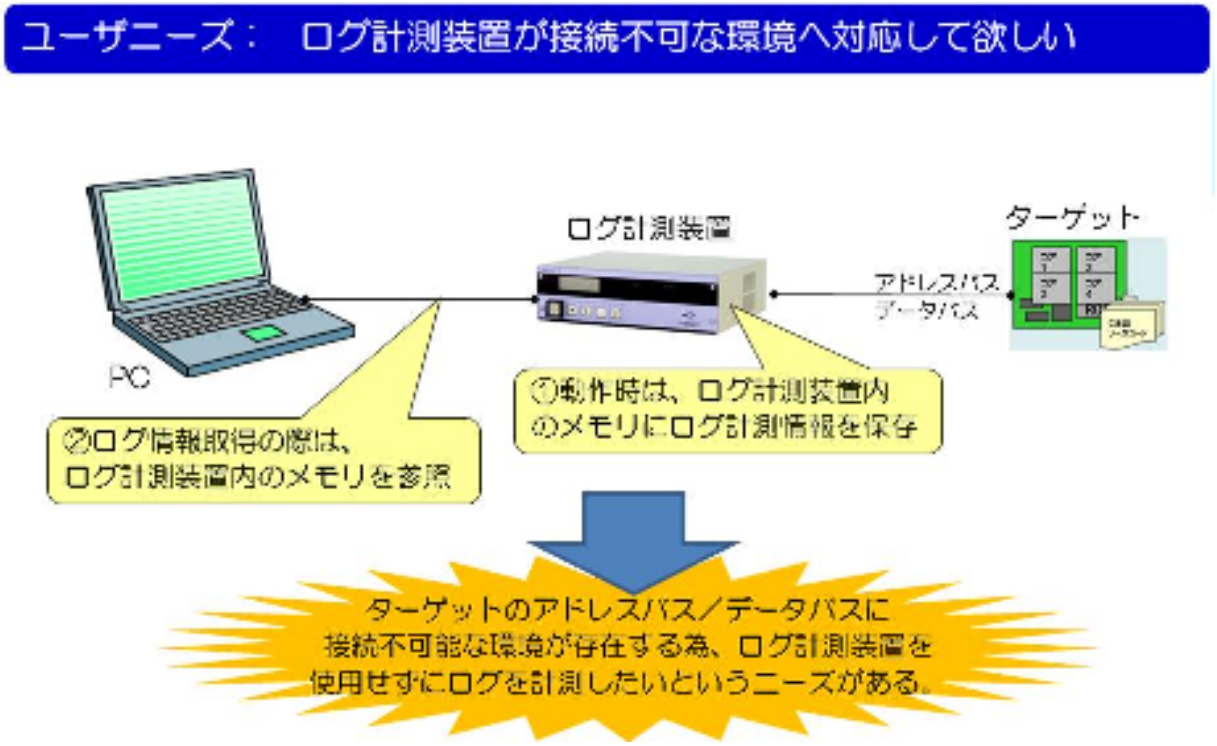


図 20 CoreRA®のユーザニーズ

7.3. 解決方法

ログ計測装置の代わりとして、ターゲットメモリを使用してログ計測を行う。

具体的には、動作時に、ターゲットメモリにログ計測情報を保存しておき、ログ情報取得の際は、Ethernet などの通信信号線などを使用して、ターゲットメモリ内のデータを参照する仕組みで対応を行う。

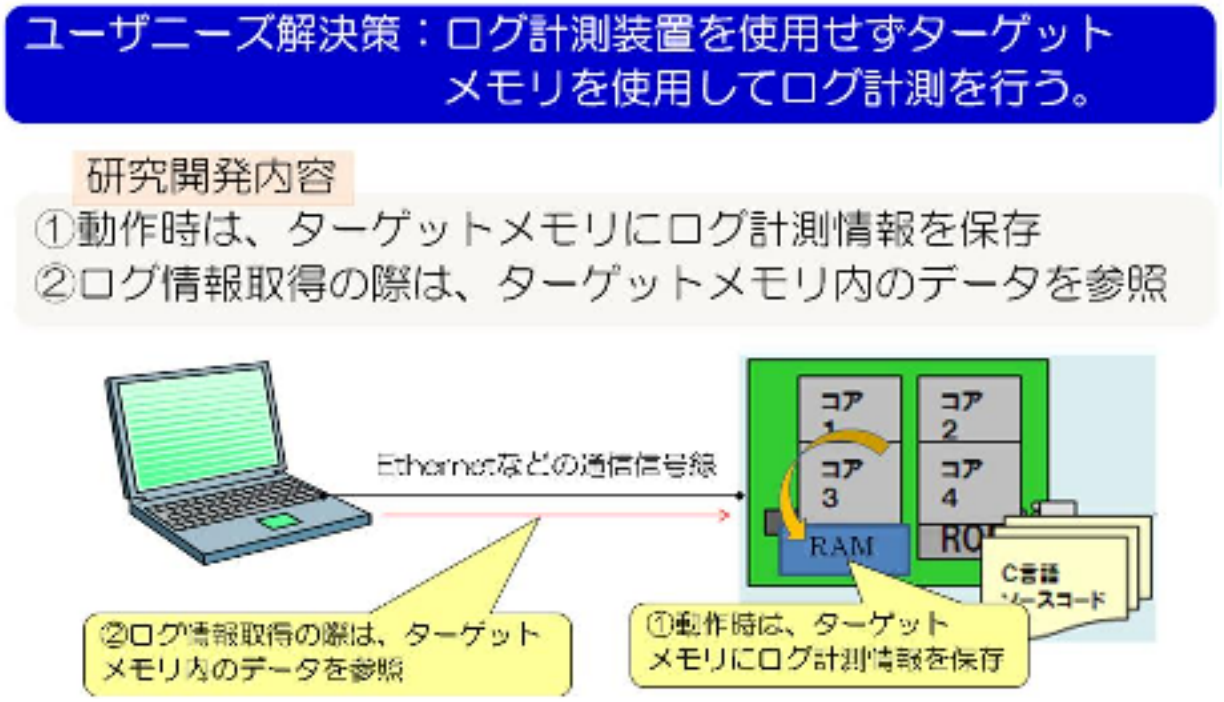


図 21 CoreRA®のユーザニーズ対応

7.4. 研究結果と評価

本研究開発で、ユーザニーズ対応版を開発し、ログ計測装置が接続不可の環境に対して、ターゲットメモリ保存環境を提供する事ができた。

これにより、ログ計測装置を接続した環境と、ターゲットメモリ保存の環境の両方に対応が可能となり、CoreRA®の潜在ユーザ数の増加が見込まれるため、本サブテーマは有効であったと考える。

なお、以下にそれぞれの環境の特徴をまとめる。

【ログ計測装置を接続した環境】

ターゲットのアドレスバス／データバスに接続する必要があるが、アドレスバス／データバスより直接ログ情報を取得出来る為、時間性の良いログ情報取得が可能となる。また、ログ計測装置には、大容量のメモリを搭載している為、大容量のログ情報取得も可能である。

【ログ計測装置を接続不可の環境】

ターゲットの標準ポートに、Ethernet や USB で接続を行い、ターゲットメモリにログを保存することなる。しかし、ターゲット上でログ情報取得プログラムが動作する為、オーバーヘッドとなり、取得に時間（誤差）が生じてしまう。また、取得できるログの容量は、ターゲットメモリの空き領域に依存してしまう。

	ログ計測装置を接続した環境	ターゲットメモリ保存の環境
環境		
接続制限	アドレスバス／データバスの信号線を接続する必要がある	ターゲットの標準ポート(Ethernet/USB)で接続が可能
時間精度	時間精度の良いログ情報取得が可能	ログ情報にオーバーヘッドがあり、取得に時間（誤差）が生じる
ログ情報量	大容量のログ情報取得が可能	ターゲットメモリの空き領域に依存する
対応年度	2009年度	2011年度

図 22 対応環境の比較表

8. サブテーマ B：消費電力低減のための待ち時間分析

8.1. 目的と目標

本研究の目的は下記のとおり。

[目的]

更なる消費電力の低減を目指す。

[目標]

待ち時間分析などタスク内部の分析を行いたいというニーズに対応するとともに、CoreRA による低消費電力化の効果を促進する。

なお、各サブテーマの位置付けは下記の通りである。

表 4 サブテーマの位置付け

サブテーマ研究概要	目的	目標
【サブテーマA】 「CoreRA」のユーザニーズ対応	・CoreRA の利活用環境拡大による顧客増加	ログ計測装置が接続不可な環境で利用できるよう、CoreRA を機能拡張する。
【サブテーマB】 消費電力低減のための待ち時間分析	・更なる消費電力の低減	待ち時間分析などタスク内部の詳細分析と、CoreRA による低消費電力の効果を促進する
【サブテーマC】 消費電力低減のためのタスク割り当て結果参照	・更なる消費電力の低減 ・CoreRA のユーザニーズ対応による顧客増加	ターゲット環境の構築を必要としない、コア配置の確認方法の構築と、CoreRA による低消費電力の効果を促進する

8.2. 課題詳細

CoreRA®は、タスク単位で最適なコア配置を解析するツールである。分割されたタスク単位で最適なコア配置を実現可能となるが、タスク内部の詳細部分の解析は行っていない。このため、さらに最適なコア配置を実現するためには、待ち時間分析などタスク内部の詳細情報を分析し、タスクの分割などを検討する必要がある。

そこで、サブテーマＢでは、待ち時間分析などタスク内部の詳細部分を検討する環境の実現を行う。

また、さらに最適なコア配置の実現により、課題２の低消費電力の効果を促進する。

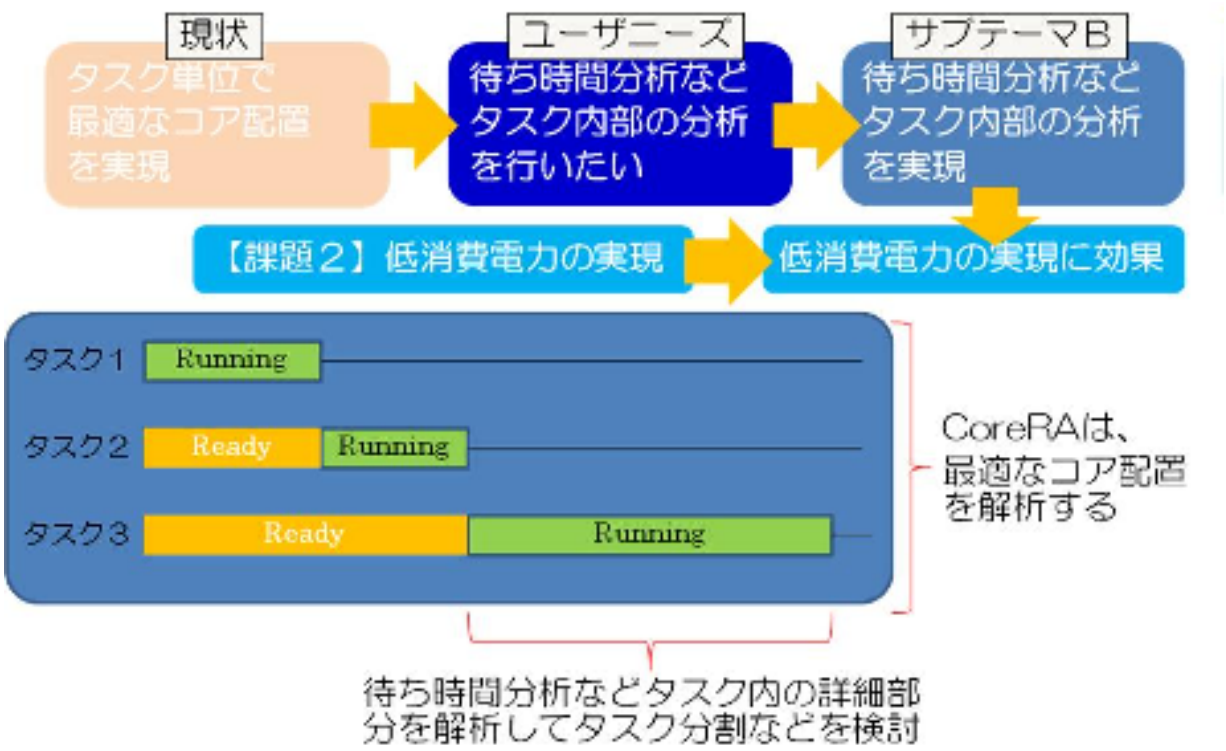


図 23 サブテーマＢ：ユーザニーズ

8.3. 解決方法

待ち時間分析などタスク内の詳細情報を分析して、さらに最適なコア配置を実現するという課題を、タスク内の詳細情報を分析するツールと連携する事により解決する。

また、【課題２】低消費電力の実現に対して、さらに最適なコア配置が行えれば、低周波数クロックの実現が可能になり、低消費電力化が促進される。

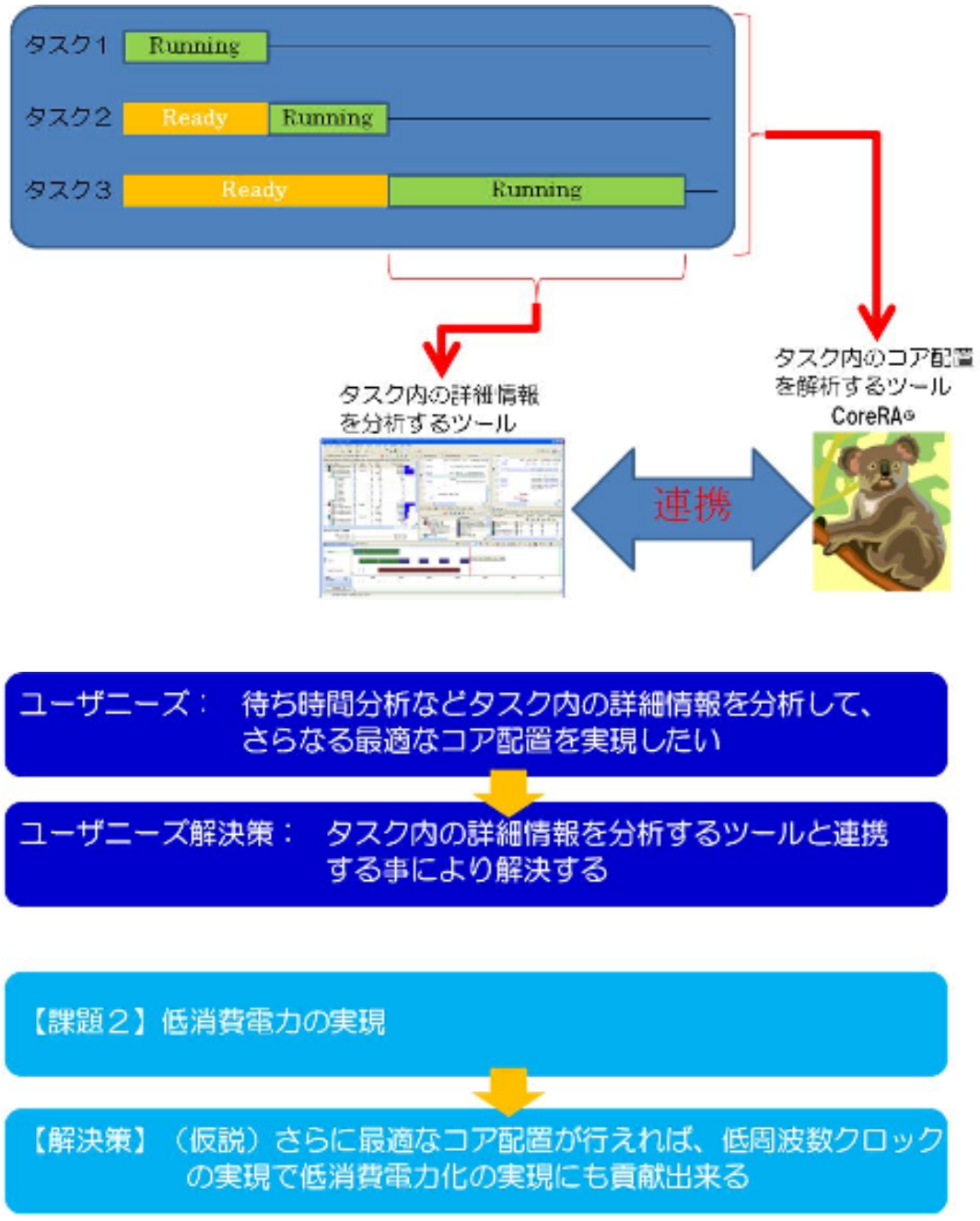


図 24 サブテーマＢ：ユーザニーズ対応

8.4. 開発に使用するタスク分析ツール「Prism」について

Prism（英 CriticalBlue 社）は、変数の依存関係、スレッドの待ち時間、メモリ・アクセスの回数を分析し、マルチコア環境を想定し割り当てを行った際のシミュレーションを実行する機能が存在するなど、タスクの（関数単位での）詳細解析が可能である。

そこで、サブテーマＢでは、Prism と CoreRA®を連携させる事により課題を解決する。

図 25 Prism 機能概要

8.5. 研究成果と評価

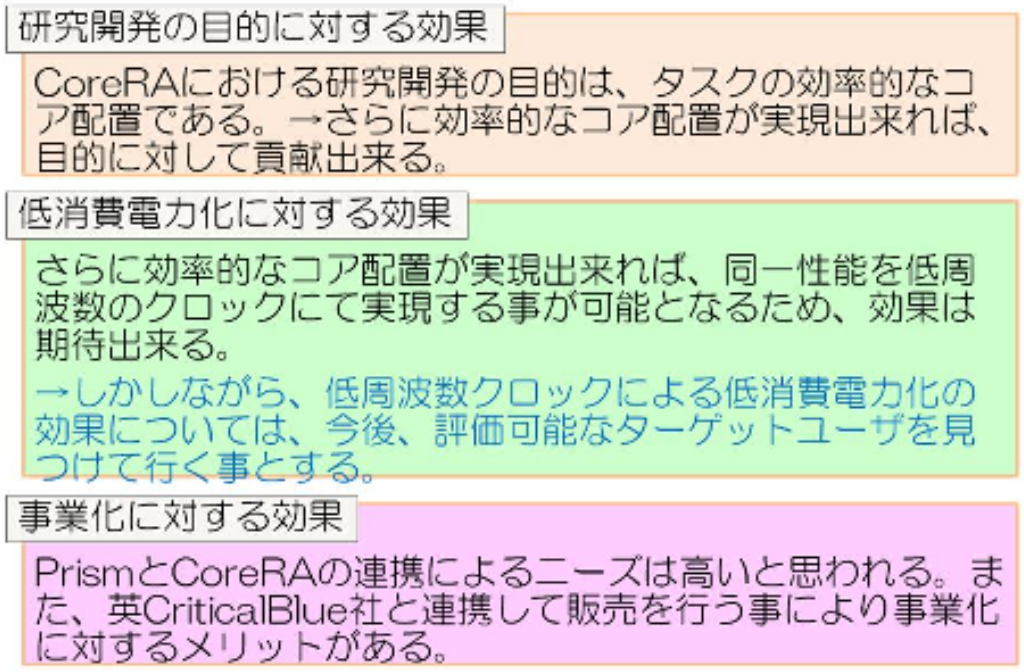
「Prism」と「CoreRA®」との連携機能を開発し、タスク内の詳細情報分析を実施した。

Prism では、関数、変数単位のログ情報を入力として、分割する為に有効な情報を出力し、タスク分割やコア配置は人手で行う事となっているが、CoreRA®と連携してコア配置を自動化する事で、開発効率が上がるというメリットがあった。

表 5　Prism と CoreRA®の比較表

	Prism	CoreRA
入力情報	関数、変数単位のログ情報	タスク単位のログ情報
分割	分割する為に有効な情報を出力し、人手でタスク分割を行う	／
コア配置	人手でコア配置を行う	自動コア配置
コア配置結果参照	配置結果を出力	／

また、研究開発の目的に対する効果、低消費電力化に対する効果、事業化に対する効果は、下記の通りとなった。



9. サブテーマ C：消費電力低減のためのタスク割り当て結果参照

9.1. 目的と目標

本研究の目的は下記のとおり。

[目的]

CoreRA®のユーザニーズ対応による新たな顧客層の獲得を図るとともに、更なる消費電力の低減を目指す。

[目標]

ターゲット環境の構築を必要としない、コア配置の確認方法を構築するとともに、CoreRA による低消費電力の効果を促進する。

なお、各サブテーマの位置付けは下記の通りである。

表 6 サブテーマの位置付け

サブテーマ研究概要	目的	目標
【サブテーマA】 「CoreRA」のユーザニーズ対応	・CoreRA の利活用環境拡大による顧客増加	ログ計測装置が接続不可な環境で利用できるよう、CoreRA を機能拡張する。
【サブテーマB】 消費電力低減のための待ち時間分析	・更なる消費電力の低減	待ち時間分析などタスク内部の詳細分析と、CoreRA による低消費電力の効果を促進する
【サブテーマC】 消費電力低減のためのタスク割り当て結果参照	・更なる消費電力の低減 ・CoreRA のユーザニーズ対応による顧客増加	ターゲット環境の構築を必要としない、コア配置の確認方法の構築と、CoreRA による低消費電力の効果を促進する

9.2. 課題詳細

現状、CoreRA®のコア配置結果については、ターゲット環境を構築し、実際にマルチコア環境において動作させるまで確認できない。

このため、ユーザニーズを調査する中で、効率的に最適なコア配置を分析するために、ターゲット環境の構築を行わずにコア配置の確認を行いたい、というニーズがあった。そこで、これに対応する機能を開発する。

また、これにより、コア配置決定が効率化されるため、課題２の低消費電力の効果も促進される。

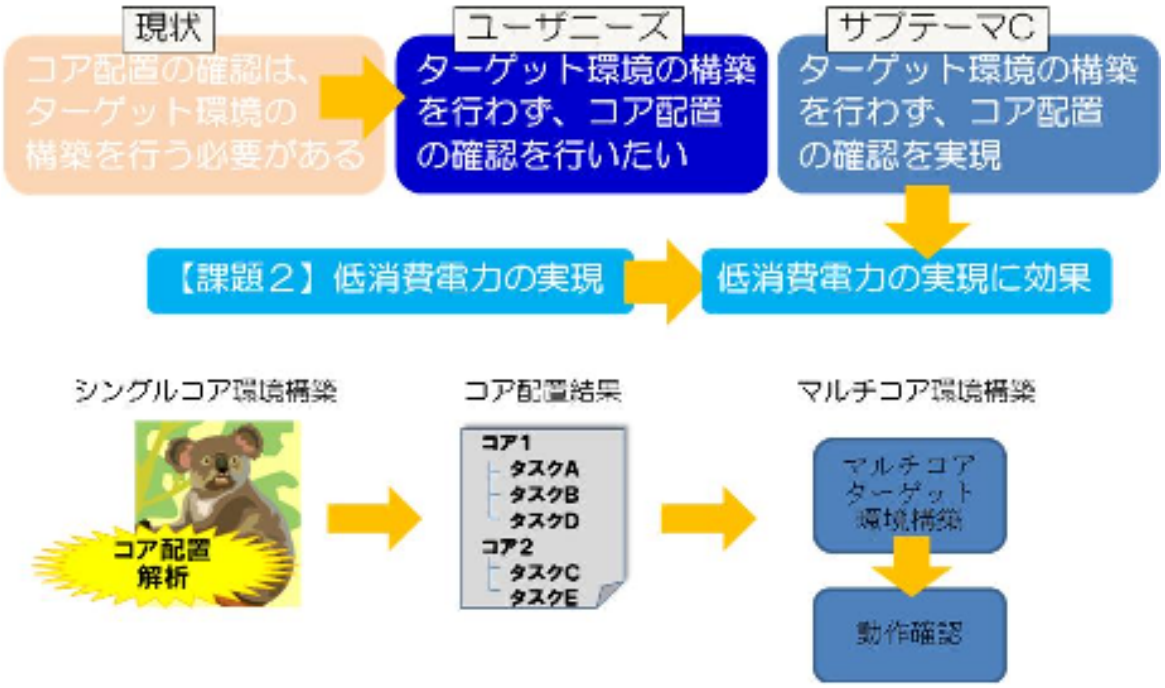


図 26 サブテーマＣ：ユーザニーズ

9.3. 対応方法

ターゲット環境の構築を行わず、コア配置の確認を行う、という課題に対して、タスクの割り当て結果を仮想的に実現する事により解決する。

また、タスクの割り当て結果が即座に参照できるようになり、コア配置決定が効率化された事で、低消費電力化の効果も明確となり、促進される。

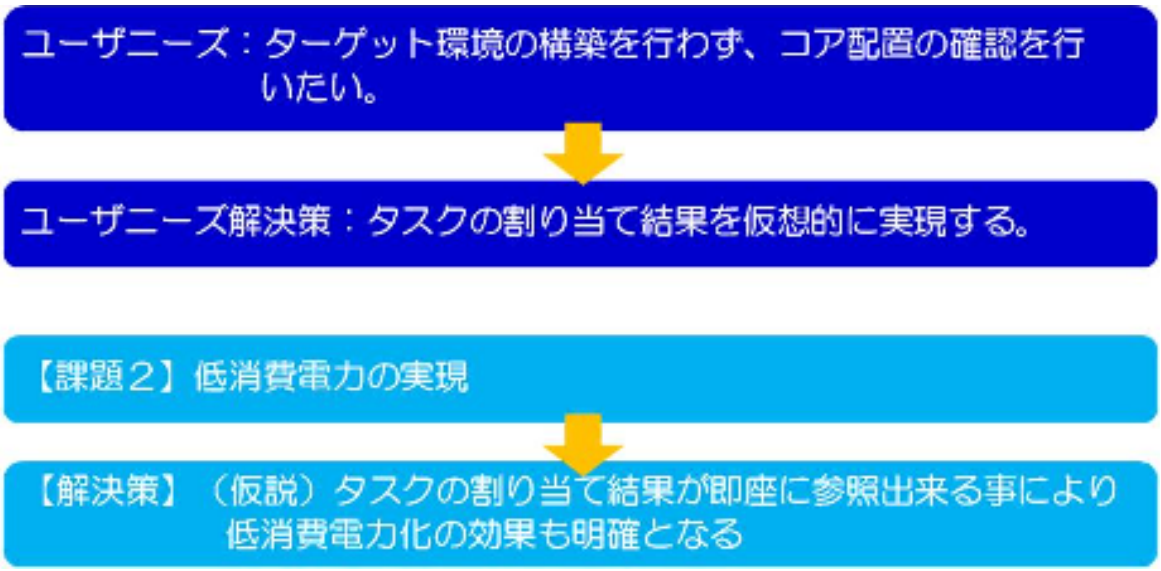


図 27 サブテーマＣ：ユーザニーズ対応

9.4. 研究成果と評価

ターゲット環境の構築を行わず、コア配置の確認を行いたい、というユーザニーズに対して、マルチコアを想定したログに変換するツールを開発した。

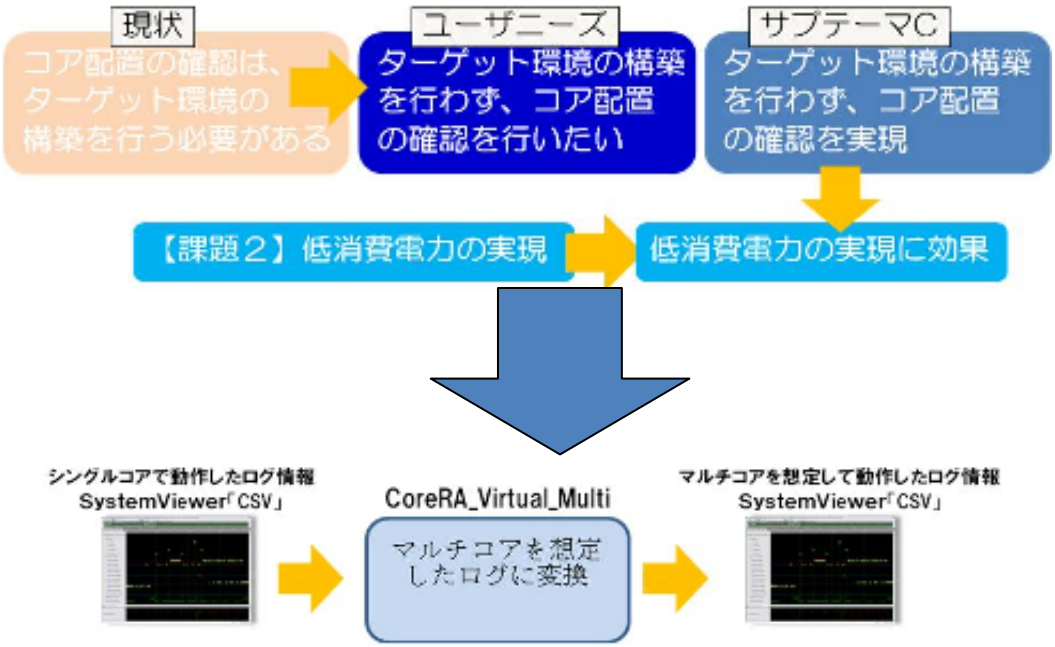


図 28 研究成果１：ユーザニーズ対応

今回開発したログ情報変換ツールでは、シングルコアで動作したログのタスクを、マルチコア環境を想定したコア配置で動作するログに変換する。これによって、各タスクが、どちらのコアで動作するかが明確になった。

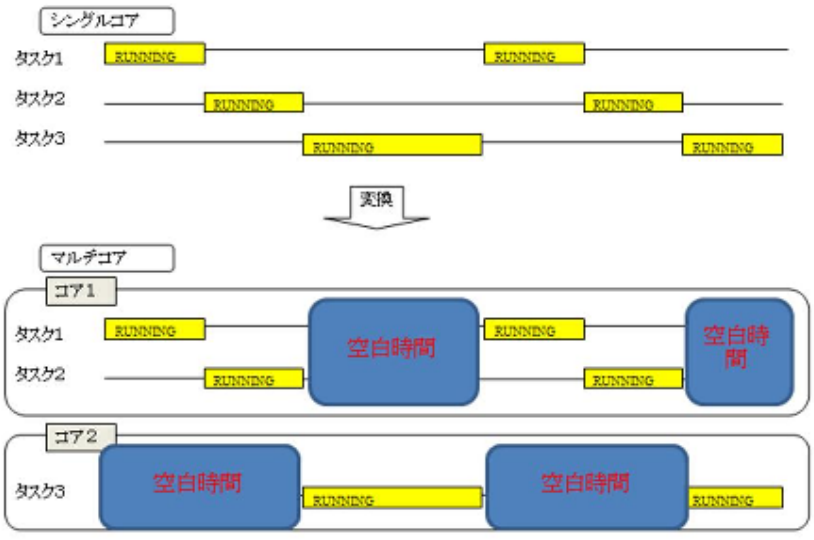


図 29 研究成果１：ユーザニーズ対応版の開発

本研究開発で、マルチコア環境を想定し、各タスクがどちらのコアで動作するかを明確にする事が可能となり、実際のターゲット環境を構築せずに、コア配置を確認できるようになった。

また、実際にマルチコアターゲット環境を構築して確認する場合と比較して、作業効率が良く、多くの事例を検討できるため、より最適なコア配置を決定することができ、低消費電量化が促進される。

10.　研究開発の成果物

10.1.研究開発の成果物

課題１の成果物として、「CoreRA」正式版を開発して、製品化を実施。
課題２の成果物として、「ZIPC OSCAR API 対応」プロトタイプ版を開発した。今後、OSCAR API 標準化の動向に合わせて正式版を開発予定。
課題３の成果物として、「ZIPC OpenMP 対応」正式版を開発した。市場（ニーズ）に合わせて製品化を行う予定。

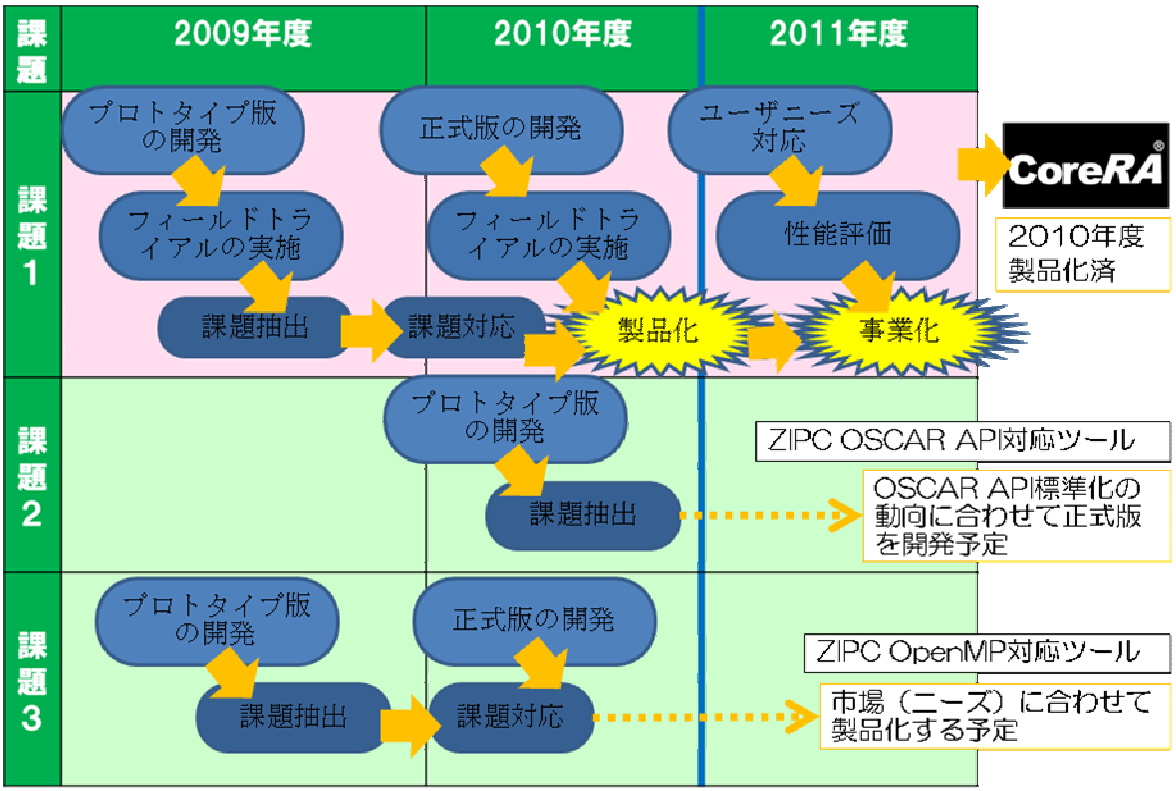


図 30 製品化へのフロー

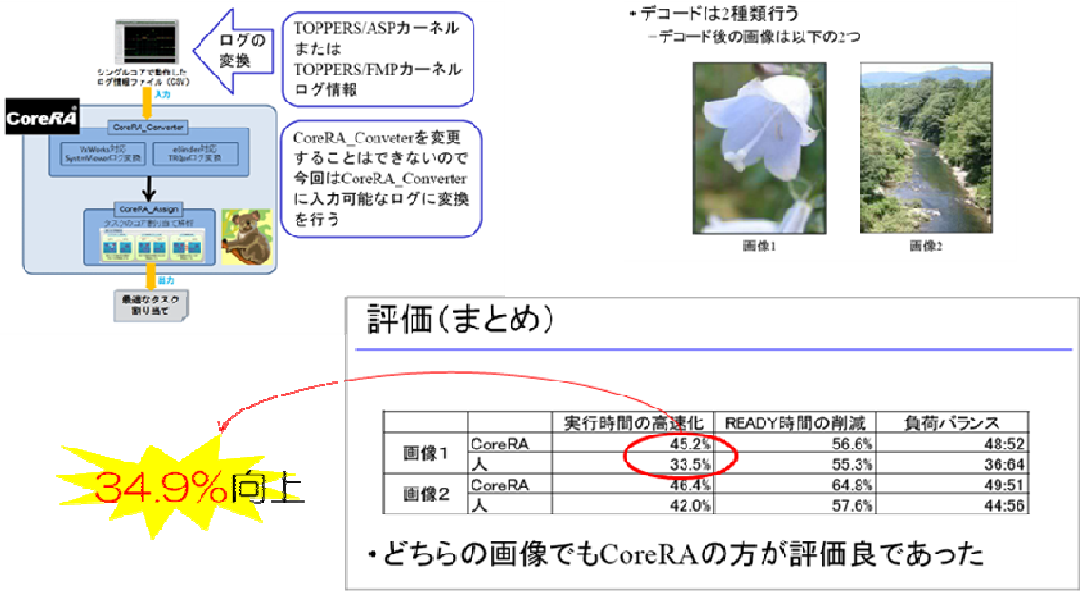
10.2.CoreRA®の製品化

課題１に対応した研究成果物は、「CoreRA®」(コアラ)という登録商標を取得し製品化を実施。CoreRA®は、シングルコアで動作したログ情報からマルチコア環境におけるタスクの最適なコア配置を解析するツール（パッケージソフト）としてリリース。

図 31 パッケージソフト（CoreRA®）の構成

10.3.CoreRA®のユーザ評価事例

名古屋大学にて TOPPERS（リアルタイム OS）による性能評価を実施。CoreRA®は、TOPPERS に未対応のため、名古屋大学が、ログ変換コンバータを自作して対応。評価方法は、人手で割り当てたコア配置と比較しておこなったが、人手で割り当てたコア配置に対して、CoreRA®を使用した場合、34%性能が向上した。



		実行時間の高速化	READY時間の削減	負荷バランス
画像1	CoreRA	45.2%	56.6%	48:52
	人	33.5%	55.3%	36:64
画像2	CoreRA	46.4%	64.8%	49:51
	人	42.0%	57.6%	44:56

図 32 CoreRA のユーザ評価事例

11. 事業化促進

CoreRA®製品化に伴い下記の通り、事業化するための内容を実施

実施年度	実施内容
2009年度	JEITA「組込みマルチコアハンドブック技術・応用編」執筆
2009年度	「シリコンシーベルトサミット福岡2010」出展
2010年度	「ESEC組込みシステム開発技術展2010」出展
2010年度	「ZIPCユーザーズカンファレンス」出展
2010年度	「ET組込み総合技術展」出展
2010年度	商標CoreRA®　キャッツWeb公開
	@IT　MONOist「マルチコア進化論」執筆
2011年度	「ESEC組込みシステム開発技術展2011」出展
2011年度	「ET組込み総合技術展」出展
2011年度	「モノづくりフェア2011」出展

図 33 CoreRA®事業化実施例

CoreRA®の事業化計画に伴いキャッツ Web にホームページを作成
http://www.zipc.com/product/corera/

補足

11.1.専門用語の解説

表 7 専門用語の一覧

専門用語	解説
組込みシステム	組込みシステムとは、家電、FA、自動車、医療機器などに組み込まれているコンピュータ・システムである。ハードである組込みプロセッサと、ソフトである組込みソフトウェアにより構成される。
マルチコア	一つのプロセッサ・パッケージに、並列して動作する複数のコアを搭載すること。パソコンでは一般的となっており、組込みプロセッサにおいてもマルチコア化が進展している。効率的にマルチコア向け組込みソフトウェアを設計する手法は確立していない。
SA 法	タスク割り当ての組み合わせの数は、タスク数の増加に対して、指数関数的に増加する。例えば、２つのコアに３０タスクを割り当てる組み合わせの数は２の３０乗と天文学的な数字となり、最適な組み合わせを求めることは不可能である。SA（シミュレーティド・アニーリング）法（焼きなまし法とも呼ばれる）は、こういったケースにおいて、最適に近い解を短時間で求める手法の一つである。
C 言語	組込みソフトウェアにおいて最も一般的に使用されている開発言語である。
OpenMP	従来の C 言語などでは並列化記述を行う事はできないが、ソースコードに OpenMP のディレクティブを挿入することにより、並列化が実現可能となる。また、OpenMP は、現在標準的な仕様となっている。
OSCAR API	OpenMP は、パソコンなどのプロセッサをターゲットに標準化された仕様であり、組込みシステムに特化したものではない。そこで、組込み向けに低消費電力などの仕様を追加した OSCAR API が 2009 年に公開され、今後標準となる事が予想されている。
タスク	組込みシステムにおいて使用されるリアルタイム OS で動作する処理単位。各タスクは独立しており、マルチコア環境下では、それぞれが並行処理される。
ZIPC	キャッツ㈱の状態遷移表を用いたモデルベース設計ツール。状態遷移表から C 言語のソースコードを自動生成することが可能。国内組込みシステム開発のデファクトスタンダートとなっている。
マルチコア分割検討ツール	キャッツ㈱のマルチコア開発支援ツール。C 言語で記述されたソースコードの関数（処理）間の依存関係（外部変数）を解析する機能を有している。

戦略的基盤技術高度化支援事業

「マルチコア環境における組込みソフトウェア設計ツールの開発」

研究開発成果等報告書

平成 24 年 3 月

委託者：九州経済産業局
〒812-8546　福岡市博多区博多駅東 2-11-1
受託者：財団法人福岡県産業・科学技術振興財団
〒810-0001　福岡市中央区天神 1-1-1
アクロス福岡西オフィス９Ｆ
〒814-0001　福岡市早良区百道浜 3-8-33
福岡システム LSI 総合開発センター
Tel 092-832-7155, Fax 092-832-1700