高効率な組込みシステム向けコンピュータアーキテクチャに関する研究

２０２１年　３月

近藤 弘郁

高効率な組込みシステム向けコンピュータアーキテクチャに関する研究

近藤 弘郁

システム情報工学研究群

筑波大学

２０２１年　３月

要旨

現在，家電製品，スマートフォン，産業機器から自動車に至る我々の身の回りにある組込みシステムは，コンピューティングのエッジ環境としてインテリジェント化が進んでいる．この IoT によるエンドポイントをインテリジェント化する流れの中，組込みシステムの多岐にわたる応用分野に対して，電力・性能の最適化，フレキシビリティと高機能化が求められている. 特に近年，車載用組込みシステムの分野では，自動運転の時代に向けて，機能の統合，処理の高度化，安全性の流れが強くなってきた.

組込みシステムは，現実世界の物理量をセンシングし，その信号を処理し，アクチュエータを制御する処理ループで構成される．この処理ループにはリアルタイム制約があるため，信号処理と制御を司る組込みデバイス (LSI) のプロセッサは重要である．組込みデバイスは，大きく言って 2 つのデバイス，MCU（マイクロコントローラ）と SoC（システムオンチップ）に分類されるが，それぞれ求められる要件は異なる．中でもデバイスの構成を決定する最大の制約は，コストである．各分類の制約の下で，電力・性能・機能要件を満たすことが求められる. 自動運転を例とする高度な組込みシステムにおいては，これら MCU と SoC がそれぞれの役割分担の処理を行っているが，今後、インテリジェント化が進む中，それぞれが制約，要件を満たしながら高度化されていく必要がある.

本研究では，このような背景から，高機能な組込みシステムを実現する手法として，デバイスの分類毎にコスト，電力，性能（演算，リアルタイム性）を最適化するコンピュータ（プロセッサ）技術の提案を行った.

処理性能と低消費電力の両立が求められる画像, AI 処理向けの SoC には，小面積/低消費電力のマルチコア CPU と超並列 SIMD(MBMX) 構成を提案し，低コスト，低電力でありながら柔軟な処理を可能にする技術を実現した．Orochi の LSI 実装を行い，実アプリケーションである位相限定相関法による 3 次元計測の場合，LSI（8CPU+2MBMX）は，従来の SMP システム (16CPU) と比較して，消費電力を 45 % 削減した.

SoC 単独では実現できない，より高い処理性能が求められる場合，マルチチップ構成が必要となる．このような場合に対し，組込み向けマルチチップ構成の実現のための高信頼インターコネクト SoC を提案し，PCIe を通信 IF としデュアルブロックシステムによる通常処理とエラー処理の分離による効率化を行った．マルチコア技術に専用コントローラを導入した提案構成による PEACH の実装を行い，4 つの PCI Express IF を用いた PEARL ネットワークにおいて，Infiniband より 51.5 %高い電力効率である 0.04W/Gbps を達成した．通常処理では，インテリジェント・割り込みコントローラのデータ転送開始機能により，CPU 割り込みハンドラを使用する場合と比較して，転送処理時間を 20 %削減した．これにより，組込み向けの高性能なインタコネクトの構築を実証した.

厳しいコスト（面積）制約と低消費電力が求められる IoT 向け MCU では，SIMD などの大規模アクセラレータによる性能向上策を使えない．このカテゴリにおける MCU に対して，命令セットと命令フェッチの手法を最適化した CPU アーキテクチャを提案した．この CPU では，様々な IoT 向け実アプリケーションにおいて，可変長命令セットと高機能命令のミクスチャで，一般的な RISC アーキテクチャ（ARM Thumb2）に対して，最大 46 %コードサイズを効率化した．静的コードサイズの削減は MCU に搭載されるメモリ容量を削減し，厳しいコスト制約をクリアできる．メモリアクセス削減は，メモリ消費電力の削減につながる．効率な命令セットによる動的コードサイズ削減に加え，電力オーバヘッドの少ない FlipFlop 型命令キャッシュの採用で，命令フェッチの効率化と低消費電力化を実現した．既存 RISC(SH2-A) に対して 2.2〜5.7 倍の電力効率と 4.5 CoreMark/MHz の高い処理性能の両立を達成した．

また，コスト/処理性能/低消費電力の要件を満たすとともに，高信頼性が必須である MCU，特に車載向けの MCU に対する，アーキテクチャの提案を行った．ここでは，MCU の統合と高信頼性（ISO26262 ASIL-D）の実現ために，高機能化 (SIMD 命令，仮想化支援機構）マルチコア CPU と高信頼のための専用ハードウェア (BIST) を組み合わせたアーキテクチャを開発した．ハイパーバイザ導入システムを対象に，上記 LSI 実装を行い，仮想マシンのコンテキスト切り替え時間を 72 %短縮し，要件（1us 以内）を達成し，車載向けの高信頼性（ISO26262 ASIL-D）を満たす MCU 機能統合方式を実証した．

高度化されていく組込みシステムを実現するためには，インテリジェントな組込みコンピュータを低コスト化かつ低消費電力化する必要がある．効率の優れた命令セットを選択すること，小面積のマルチコアと小面積のアクセラレータ（SIMD）の効率よい選択をすること，独立性のあるブロックを分離し，高信頼性とリアルタイム性を向上することの３つのバランスをとることにより，高効率な組込みシステムを実現することが可能である．

目次

図目次

表目次

第 1 章

序論

現在，家電製品，スマートフォン，産業機器から自動車に至る我々の身の回りにある組込みシステムは，コンピューティングのエッジ環境としてインテリジェント化が進んでいる．組込みシステムの応用分野は，インフラストラクチャ（ライフライン，交通，構造物，公共システム），スマートホーム（スマートメータ，ホーム エネルギー マネジメント システム，自然エネルギー発電，蓄電，医療，家電製品）・オフィスオートメーション（空調，照明設備の自動管理，ビル集中制御），ファクトリーオートメーション（産業用ロボット，AC ドライブ，インバーター）に広がっている．

本研究は，エッジ環境におけるインテリジェント化が進む組込みシステムに向けて，電力効率と面積効率が求められる組込みデバイスのコンピュータ（プロセッサ）アーキテクチャに関するものである．

本章では，本研究の対象となる組込みシステムの動向を提示し，車載組込みシステムにおける要求，要件と問題点について述べる．次に，組込みデバイスのコンピュータ（プロセッサ）のアーキテクチャ技術がこれらに対しいかに，有効な手段を提供できるかを，最適な技術の提示と有効な技術開発の観点から述べる．

図 1.1: インテリジェント化が進む組込みシステム

1.1　組込みシステム

1.1.1　組込みシステムの動向

組込みシステムは，リアルタイムに世界中のあらゆる場所からネットワークに繋がり，情報を交換し，膨大なセンサー情報に基づく制御を行う．このような高度なインテリジェントシステムの需要が高まり，その需要を満たすために，エンドポイントでのソフトウェアの規模と複雑さが爆発的に増している．近年の動向としては，人工知能（AI）技術のエンドポイントへの適用により，リアルタイムでセンサー情報を判断し，様々な対応を行えるように，クラウド依存から分離する流れが加速している．これは，従来の統計 AI アプリケーションはクラウドで実行し，推論はエンドポイントで行うことで，許容できないクラウド遅延問題を回避するためである．

IoT によるエンドポイントをインテリジェント化する流れの中，組込みシステムの多岐にわたる応用分野に対して，電力・性能の最適化，フレキシビリティと高機能化が求められている [4]. この要求に応えるため，メモリ容量の急速な増加と組込みデバイス (LSI) 機能の進歩により，組込みプロセッサの周波数と処理パフォーマンスが向上してきた．しかしながら，多くの組込みシステムには，依然として厳しいコスト，電力消費，およびスペースの制約がある．

また，大きな変化を迎えている自動車業界の動きは”CASE”という言葉で示される．4 つのキーワードの頭文字から取ったもので，コネクティビティ（接続性）の「C」，オートノマス（自動運転）の「A」，シェアード（共有）の「S」，そしてエレクトリック（電動化）の「E」である．このような潮流の中，自動車の燃費向上/電動化，安全性向上，コネクティビティ/IT 化が進んでいる．

近年，自動運転の時代に向けて，自動車の電気/電子 (E/E) アーキテクチャは ECU の統合と集中化に向けて急速に進化している [5][6]．複数の電子制御ユニット（ECU）を統合することで，車載ネットワークに使用されている ECU 間のワイヤハーネスをなくし [7]，車両の重量が軽減され，その結果，エネルギー消費の低減につながる（図 1.2）．この ECU の統合を実現するためには，多くの機能を 1 つの電子デバイスに詰め込まなくてはならなくなる．

1.1.2　組込みシステムの処理概要

本節では，組込みシステムと組込みデバイスを概観し，本研究で取り上げる基本要素技術について説明する．

組込みシステムの特徴は，機能が特定の応用分野に特化していることにあり，各応用分野に従い，電力・性能の最適化，フレキシビリティと高機能化が求められている． 組込みシステムは，現実世界の物理量をセンシングし，その信号を処理し，アクチュエータを制御する処理ループで構成される．この処理ループには高いリアルタイム性が求められるため，信号処理と制御を司る組込みデバイス (LSI)，特に，信号処理と制御を司るデジタル技術，すなわち，プロセッサ技術は重要である．組込みシステムでは，信号処理のループを，高い演算処理性能でリアルタイム性を満たしながら，少ない電力（低消費電力）かつ低コストで行うことが必要である．

図 1.3 は，組込みシステムの処理を，自動運転の例で示したものである．組込みシステム処理の一つ目

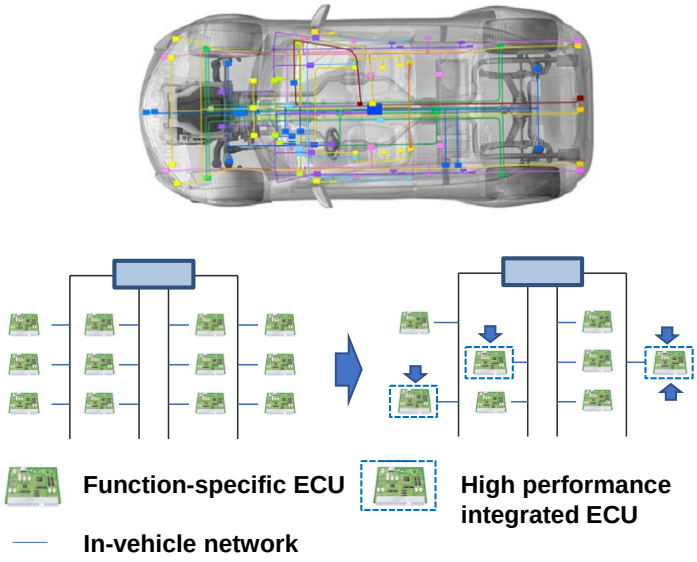


図 1.2: ECU の統合

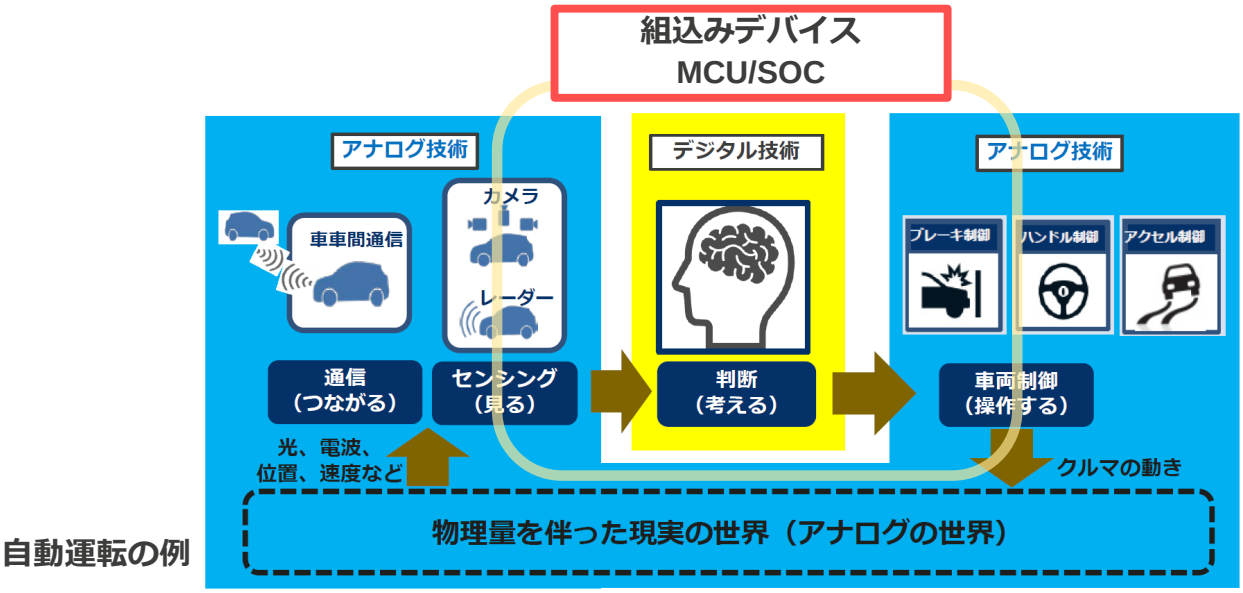


図 1.3: 組込みシステムの処理

の特徴は，信号処理にある．組込みシステムでは，現実世界の物理量（光，電波，位置，速度など）をセンシングし，情報を信号として処理する．信号には，画像，通信のためのインタフェース信号や制御のためのセンサーの測定値などがある．自動運転システムでは，カメラ，レーダなど数多くのセンサー情報を取得し，その情報を処理することで，より正確，広範囲な環境情報を取得する（センサーフュージョン）．信号処理は，各種フィルタなど特定の計算で行われ，かつ，小規模な機能からアプリケーション全体まで幅

広い処理性能要件が存在する．信号処理の結果は，システムは判断を行い，制御対象（ブレーキ制御，ハンドル制御，アクセル制御）に働きかけることで，車両制御の 3 要素（走る，止まる，曲がる）実現する．

組込みシステム処理の二つ目の特徴は，リアルタイム制約にある．組込みシステムでは，特定の計算を特定の時間までに完了する必要がある．特に制御処理には周期性があり，ある時間内に決まった処理を行う必要があり，完了できない場合は，システムに障害の発生にもつながる．

組込みシステム処理の残り二つの特徴は，コスト制約と電力制約である．この二つの制約は，システムの構成要素である組込みデバイスにとっても大きな制約となる．特に，コストの制約は最終的にはすべてに優先される場合が多い．中でもメモリはシステムコストのかなりの部分を占める可能性があり，そのような場合にはメモリサイズを最適化することが重要になる．また，メモリが大きいほど電力も大きくなるため，組込みアプリケーション向けにを最適化することが重要になる．

組込みシステムでは，アプリケーション全体を小容量のオフチップメモリに収める必要があり，プログラムのデータサイズはアプリケーションによって決定されるため，メモリサイズの最適化にはアプリケーションプログラムのコードサイズを最適にすることが重要となる．

電力に対する制約は，組込みシステムに電力を供給する環境に依存しており，特にバッテリー動作するシステムにおいては，より厳しい低消費電力化への要求がある．また，デバイスには物理的な環境温度の限界があり，放熱や冷却により制約をクリアする必要がある．これに対し，組込みデバイスは熱対策に対するコストを最小限に抑えるために，デバイスの消費電力にはさらに制約がかけられる．

上記のように組込みシステムは，人間が入力したデータを取り扱う PC/データセンタなどのシステムとは異なる特徴があるため，多くの場合，組込みプロセッサと汎用プロセッサの情報の処理方法は非常に異なる．

1.1.3 組込みデバイスの要件と分類

組込みシステムに求められる主な要件は，以下の 4 項目である．

- 低コスト実装
- 低消費電力
- 高い演算性能
- 高いリアルタイム性

さらに，車載システムでは，上記 4 項目に加えて

- 高い信頼性

が求められる．

組込みシステムは，上記の要件のバランスを解くことが必要であり，その主たる要素である組込みデバイス (LSI) にも適用される．組込みデバイス (LSI) の構成を決定する最大の制約は，厳しいコスト制約である. その制約の下で，電力/性能/機能の各要件を満たすことが求められる．また，コスト以外の要件は，適用するシステムの性能，機能に依存し様々である．高度化したシステムに対応するには，要件を実現するテクノロジのバランスをとった，組込みデバイス (LSI) を実現する必要がある．

一方，組込みデバイス（LSI) は，その構成により，SoC（システムオンチップ），MCU（マイクロコントローラ）の 2 カテゴリに分類される．

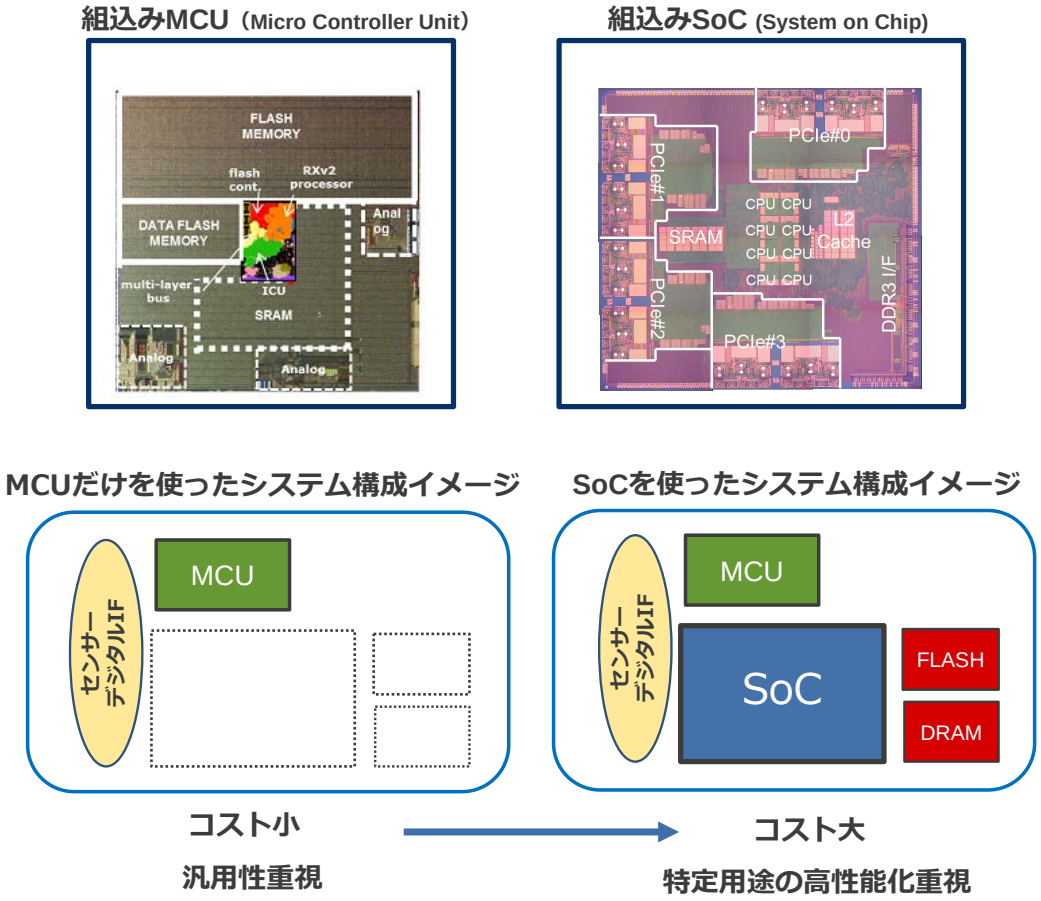


図 1.4: 組込みデバイスの分類とシステムイメージ

図 1.4 に，SoC,MCU の LSI のチップ写真と，これらの LSI が適用される組込みシステム構成の例を示す．SoC は，画像処理等の処理負荷の高いアプリケーション（カメラ，プリンタ）に用いられるデバイスである．SoC は，アプリケーションに応じた特定用途の演算処理を行い，ドメイン・スペシフィックなプロセッシング回路，プロセッサコアとワークメモリとしての SRAM から構成されるのが一般的である．SoC は，アプリケーション固有回路と一緒にプロセッサコアを使用する，特殊用途のマルチプロセッサと言える．また，ロジックプロセスを採用し，動作周波数を高めることができる．システムを構築するには，SoC の主記憶用に外部メモリデバイス (DRAM，FLASH) と，クロック，電源，電源制御などのシステム管理を行うシンプルな MCU(システムコントローラ) と外部センサーとのインタフェースが必要である．SoC は，比較的コスト制約が緩いため，消費電力を抑えながら要求される高い性能を満たすために，回路規模をかけることができる．

MCU は民生・産業機器や車載制御向け ECU に用いられるデバイスである．コスト制約が厳しいアプリケーションに適用され，プロセス進化の恩恵により，MCU の多機能，高機能化が進んでいる．汎用処理を行う CPU コアと主記憶用の大容量 FLASH メモリ，アナログモジュールが，MCU には内蔵され

る．その他，MCU はシステム管理も行うため，それに関わる電源，クロックモジュールまでも内蔵する．FLASH などの NVM(Non Volatile Memory) を内蔵するため，混載プロセスを使用する．このような MCU の進化とドメイン・スペシフィックなプロセッシング回路を組みあわせることで，コストの厳しい組込みシステムが更に複雑なシステムに対応することができるようになる．

以降では，組込みデバイスである SoC，MCU における要件（コスト，消費電力，性能）についてより詳しく述べる．

1.1.4 組込みデバイス（LSI）とコスト

この節では，組込みデバイスのコストが，そのチップ面積に大きく依存することを説明する．組込みシステムでは，システムの価格から組込みデバイスに掛けられる費用の制約が発生する．また，組込みデバイスのカテゴリにより大まかな価格の上限が決まる．この価格の上限が，組込みデバイスのコストの制約となる．デバイスのコストについては，各半導体メーカーの事情に大きく依存するが，一般的傾向/特性は同じであるので，ここでは，Heneesy&Patterson のテキスト [1]（p59-p62）を引用し，デバイスのコストについて説明する．

$$\text{Cost of integrated circuit} = \frac{\text{Cost of die} + \text{Cost of testing die} + \text{Cost of packaging and final test}}{\text{Final test yield}} \tag{1.1}$$

$$\text{Cost of die} = \frac{\text{Cost of wafer}}{\text{Dies per wafer} \times \text{Die yield}} \tag{1.2}$$

$$\text{Dies per wafer} = \frac{\pi \times (\text{Wafer diamter}/2)^2}{\text{Die area}} - \frac{\pi \times \text{Wafer diameter}}{\sqrt{2 \times \text{Die area}}} \tag{1.3}$$

$$\text{Die yield} = \text{Wafer yield} \times (1 + \text{Defects per unit area} \times \text{Die area}/\alpha)^{-\alpha} \tag{1.4}$$

$$\alpha : \text{depends upon the manufacturing process(generally } \alpha = 2.0 \text{ for simple MOS)}$$

デバイスのコストの計算式は，(1.1)-(1.4) の式で与えられる [1]．また，コストの計算式の各パラメータに数値を代入した時の例を図 1.5 と図 1.6 に表とグラフを引用する．ここに使われているデータは，1990 年ごろのものであり，価格の絶対値は現在の値段とズレはあるものの，計算式はほぼ変わらず傾向は同じなので，ここではこの値を使う．

このグラフにより，デバイスのコストの特徴が示されている．図 1.6 は，正方形の一辺の長さを横軸に，コストを縦軸に取ったグラフであるが，面積が $1cm^2$ を越えると Die Cost 影響で急激にコストが増大する．この急激な Die Cost の増加は，歩留まりの影響であり，欠陥密度（Defects per unit rea）と面積（Die area）の積の項によるものである．その他，テストコスト，パッケージコストもデバイスのコストに影響を与えるが，特殊な用途でない組込みデバイスのコストは，ほぼチップ面積で決まるといってもよい．

このように，デバイスの収益性が一辺 $1cm$ を越えるあたりから大きく変わるため，特殊な用途や価格の制約がない（価格には代えがたい機能）場合を除いて，$1cm^2$ 以下に面積を抑える傾向にある．特に，組込み向けのデバイスでは，最終製品の価格や価格競争の観点から面積がこの範囲に収まるように製品の企画を立てることが設計者に求められる．組込みデバイスの中でも，２つのカテゴリ，MCU と SOC とでは，若干状況が違い，面積に対する制約は異なる．組込みシステムとしてアプリケーションの必須

Area (sq. cm)	Side (cm)	Die/ wafer	Die yield/ wafer	Cost of die	Cost to test die	Packaging costs	Total cost after final test
0.06	0.25	2778	79.72%	$0.25	$0.63	$5.25	$6.81
0.25	0.50	656	57.60%	$1.46	$0.87	$5.25	$8.42
0.56	0.75	274	36.86%	$5.45	$1.36	$5.25	$13.40
1.00	1.00	143	22.50%	$17.09	$2.22	$5.25	$27.29
1.56	1.25	84	13.71%	$47.76	$3.65	$52.25	$115.18
2.25	1.50	53	8.52%	$121.80	$5.87	$52.25	$199.91
3.06	1.75	35	5.45%	$288.34	$9.17	$52.25	$388.62
4.00	2.00	23	3.60%	$664.25	$13.89	$52.25	$811.54

図 1.5: 文献 [1] によるデバイスのコストの例

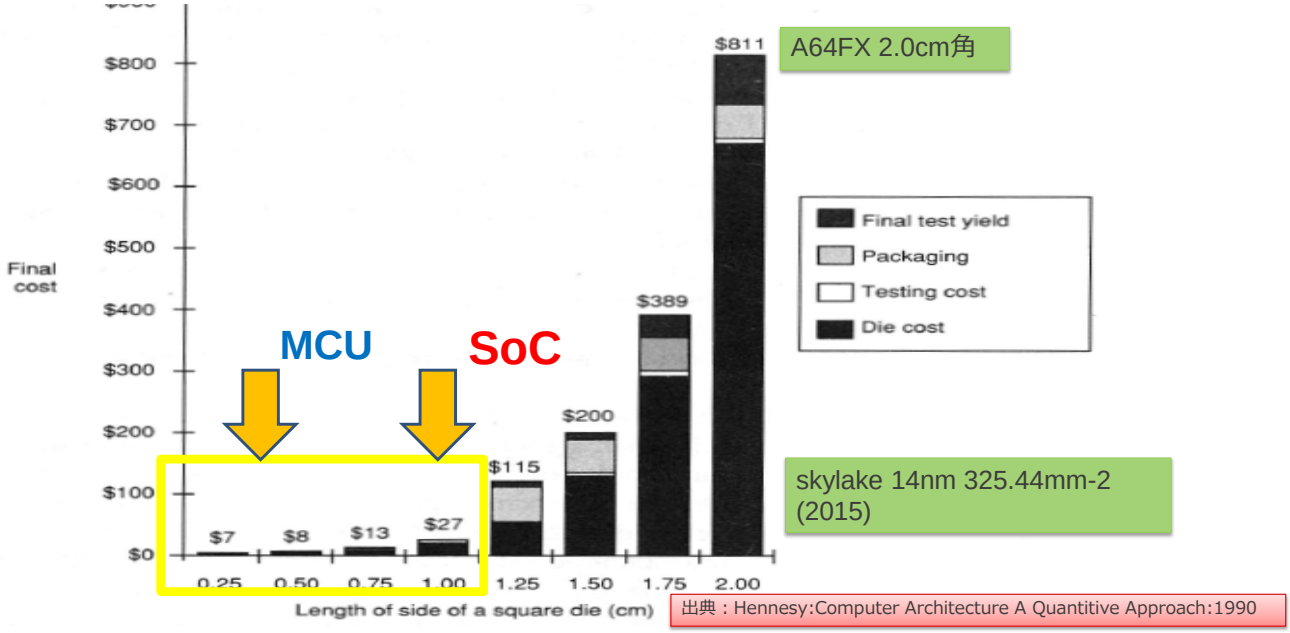


図 1.6: 文献 [1] によるコスト制約

要件から仕様が決まる SoC は，それに対価が払われる傾向にあり，そのため，面積コストの制約は緩く，$0.41cm^2$〜$1.0cm^2$ あたり，一方，汎用で標準的な用途が多い MCU では，価格制約からくる面積コストの制約は SoC より厳しい．そのため，面積制約は，$0.05cm^2$〜$0.5cm^2$ あたりが一般的である．

この面積制約の傾向は，プロセステクノロジが変わっても，あまり変わることはない．よって，この面積制約は現在 (2020 年) にも適用できる．プロセステクノロジが進化し，ゲート密度が上がる一方，ウェハあたりの製造コストが上がることと，製品自体のコストダウンが進むからである．

1.1.5 演算処理性能と電力

電力の制約も，コストと同様にデバイスのカテゴリに依存し，放熱コスト（パッケージ）と電力の環境（バッテリ駆動か，電源につながれているか）により制約される．パッケージの選択により，消費電力の最大許容値 (MAX 値) が変わる．安価な熱抵抗の大きなパッケージでは，より低い電力が求められる．

図 1.7 は，文献 [2] から引用した演算処理性能と電力の関係を示したグラフに情報を追加したグラフである．横軸は性能：DGIPS(Dhrystone Giga Instruction Per Second)，縦軸は電力あたりの性能 (DGIPS/W)，右肩上がりの斜直線は，消費電力値 (W) を示している．横軸は，LSI の周波数レンジとほぼ同等とみなすことができ，グラフの左に進むほど，かけることのできるコストは小さくなる．この図から分かるように，組込みデバイスのカテゴリ毎に，消費電力の制約が異なる．

モバイル PC 向けの LSI(SoC) は，10W（赤）が電力リミットである．MCU は，1W（緑）が電力リミットである．MCU は，さらに，用途別に IoT 向け MCU と車載向け MCU に分類できる. 車載向け MCU は，100mW（緑）〜1W（緑）の電力範囲におさまり，IoT 向け MCU（緑）は 100mW 以下の領域にある．

演算処理性能と電力の関係については，IoT 向け MCU は最もコスト制約が厳しく，性能/電力要件が高い．車載向け MCU は，IoT 向け MCU に較べてややコスト制約，性能/電力に余裕がでるが，プロセステクノロジ観点でいうデバイス自身の信頼性（高温対応等）に加え，機能的な高信頼性についても求められる．このように組込みデバイス全体は，SoC，IoT 向け MCU，車載向け MCU の 3 カテゴリに分類することができる．

赤色の両矢印は SoC の範囲を示し，青色の両矢印は MCU の範囲を示す．プロットは，本論文で示す LSI 実装例である．赤プロットは SoC（低電力・高性能なヘテロマルチコア SoC:Orochi, 低電力・高性能なインターコネクト SoC:PEACH）で，青プロットは，MCU(MCU 向け低電力プロセッサ:RX と車載向け高信頼プロセッサ RH850) である．

1.2 低消費電力化技術

本節では，これまで開発されてきた LSI 設計の低電力化に関する基本要素技術とその分類について簡単に説明する．

1.2.1 低電力技術の分類

CMOS 回路の電力の計算式は以下の通りである．

$$P = (1/2)\sum \alpha C V^2 f + \sum Ileak \times V \tag{1.5}$$

α : $activityratio$（スイッチングの動作率）
C : $capacity$（静電容量）
V : $voltage$　（電源電圧）
f : $operating frequency$（動作周波数）

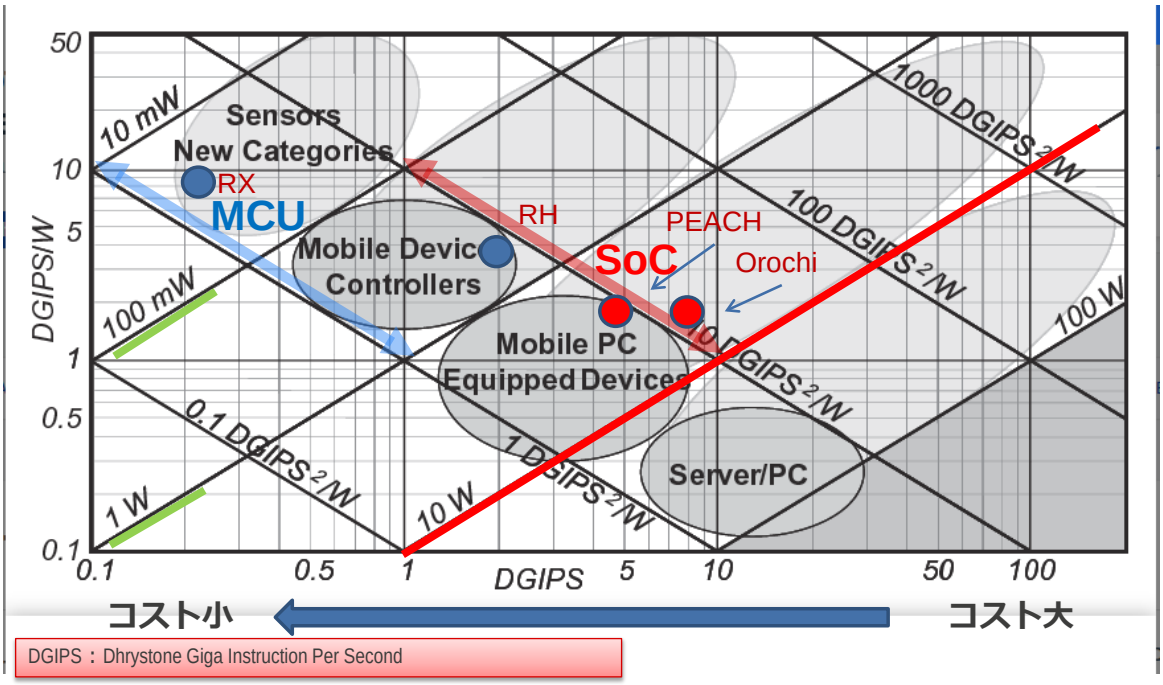


図 1.7: 組込みデバイスのカテゴリと消費電力制約の関係性:文献 [2] より

$Ileak$: $leakcurrentofeachcircuit$（リーク電流）

最初の項は動的電力項であり，2 番目の項は静的（リーク）電力項である．

多くの既存の低電力技術は，低電力条件を管理する方法によって分類される．クロック信号が停止すると，長い待機期間中に動作周波数 f が減少する．クロック周波数制御では，必要な処理性能のために f を減らす．クロックゲーティング技術は，ゲート動作制御の細かいピッチでスイッチングの動作率 α を低減する．パワーゲーティング技術 [8][9]，言い換えれば，チップの全部または一部の電源のオンとオフは，LSI の一部の非動作期間中のリーク電流の低減に関連する．動的電圧周波数スケーリング技術（DVFS）[10] は，動作周波数 f と電源電圧 V を処理負荷に応じて制御する．適応電圧スケーリング技術（AVS）[11] は，動作周波数 f は固定するが LSI の動作条件をモニタし，電源電圧 V を最適化することで電力を減らす．

1.2.2　低電力に向けたハードウエア制御の粒度

前で説明した低電力技術のほとんどには，次のような共通のトレードオフが存在する．低電力技術を LSI ハードウェア中のより小さな要素に個別に適用していくと，一般に，各要素ごとのパワーダウン期間が長くなり，LSI 全体として省電力量は大きくなる．一方，低電力技術をより小さな要素に適用するためには，低電力制御用の追加回路の実装が要素ごとに必要になり，LSI チップの面積オーバーヘッドが大きくなる．低電力制御の粒度と面積オーバーヘッドの間のトレードオフは，低電力 LSI 設計の最も重要な問題の 1 つである．

130nm MOS 標準プロセス技術の後，同じノードのプロセステクノロジの中で，高性能，低リーク，低動作電圧などのいくつかの種類のプロセスオプションが準備されているため，プロセステクノロジの選択はシステム要件を満たすための重要な要素の 1 つである．さらに，マルチ Vt（しきい値電圧）技術 [12] も各プロセスに適用され，速度と電力特性が向上している．今日，マルチ Vt 技術は，自動 CMOS 設計フローに組み込まれており，たとえば，クリティカルパスには低 Vt トランジスタのセルを選択し，それ以外はできるだけ電力消費の少ないセルをマッピングするといった，電力と速度の最適なトレードオフを EDA ツールにより行われている．

次のステップとしては，デバイスと回路の組み合わせによる継続的な電力削減の改善が行われてきた．ここでは，バックゲートバイアス [13] と負電圧バイアス [14] の 2 つの手法を紹介する．バックゲートバイアスはトリプルウェルデバイス構造により，MOS トランジスタのスレッショルド電圧を制御することができ，バックゲートバイアスの動的制御はスタンバイリーク電流の抑制と高速スイッチング回路の両方を実現することができる．負のゲート電圧バイアスは，電源スイッチ [15] と階層型電力線 [14] の回路技術により，サブスレッショルドリーク電流を大幅に抑制する．これらの低消費電力技術は，現在多くの種類のバッテリー駆動のモバイルアプリケーションで使用されている．

長時間にわたる待機時間のクロック信号の停止技術は，一般的に機能モジュールごとに LSI チップに適用される．IP 全体へのクロック停止技術は 1990 年代から実装されている．クロック停止の典型的な例として，ゲート状態がフリップフロップ（FF）値によって決定されるゲートクロックによって制御される動画のコーデック LSI がある．FF 値がゼロに設定されると，LSI のクロック信号は停止する．クロックゲーティングは，IP 全体だけでなく IP の一部にも適用される．

パワーゲーティングに関しては，より大きな粒度がテクノロジに適用される．携帯電話向けコーデック用に設計された SOC[8] は，1 つのチップに 20 のパワードメインを備えている．

アーキテクチャを選択する上で，クロックゲーティングなど動作率 f を下げやすいもの，小面積化することで静電容量 C を抑えるものを選択していくことが，低電力化につながる．

1.3 プロセッサ構成要素の組み合わせ

本節では，組込みプロセッサ設計における構成要素の組み合わせについて述べる．

一連のコスト（面積）と電力の制約が与えられた場合に，最大の性能を推定し，トレードオフを考慮して，デザインの候補を絞り込むことを可能にするのが，組込みプロセッサ技術である．図 1.8 は，組込みプロセッサの構成要素（プロセシング・エレメント）のさまざまなアーキテクチャ候補 (CPU/DSP, MIMD，SIMD，専用ロジック) とそれらの比較を示している．縦軸はフレキシビリティ (アプリケーション適用範囲) を示し，横軸は電力効率を示す [Performance/W] を示している．プロセッサの構成要素は，フレキシビリティと性能により，4 種類に分類される．

1. ソフトウェアとの親和性が高い CPU や DSP は，柔軟性の点でさまざまな用途に採用できるというメリットがあるが，性能/W は限られており，画像処理などのメディアアプリケーションには不十分である．
2. また，特定のアプリケーション向けに設計されたハードワイヤード回路は最高のパフォーマンスと

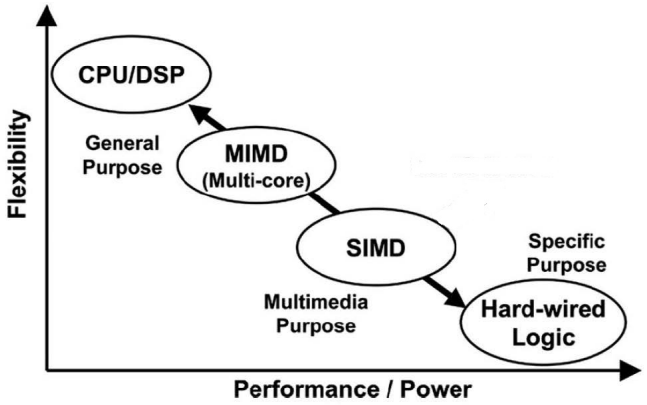


図 1.8: プロセッサ構成要素の組みあわせ

高いエネルギー効率を発揮すると考えられているが，柔軟性はほとんどない．FPGA は限定的な処理に対してパフォーマンスと柔軟性を両立するが，エネルギー効率，面積コストの点で問題があり組込みデバイスに広く採用されるにはいたらない．

3. そのため，並列処理アーキテクチャは，プログラミングコストが高くなるが，パフォーマンスを向上させるための魅力的な手法と見なされている．これは，シリコンチップで使用されるトランジスタの数が着実に増加しているにもかかわらず，今日の最先端半導体プロセスでは，以前のようにクロック周波数が増加しなくなったためである．
4. さらに，マルチメディア，AI 処理向けに用途限定することで，単一命令/複数データストリーム（SIMD）アーキテクチャが採用でき，さらに電力効率（Performance/W）を向上できる．

これら構成要素の組み合わせで，プロセッサを実現する．また，コスト（面積）と柔軟性（flexibility）とのトレードオフも考慮しなくてななら ず，専用ハードウェアか，プログラマブルな汎用ハードウェアか，あるいはその組み合わせかの選択も存在する (図 1.9).

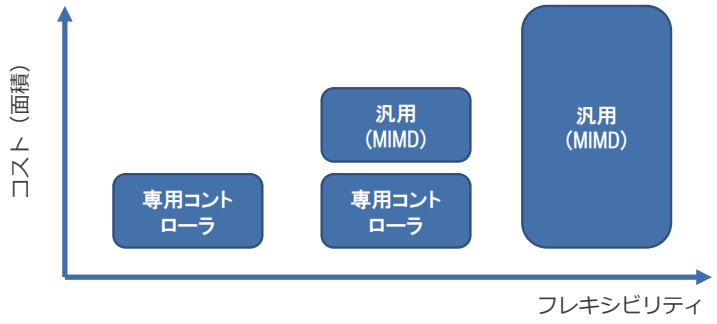


図 1.9: コスト（面積）とフレキシビリティのトレードオフ

1.4 組込みシステムのアーキテクチャ

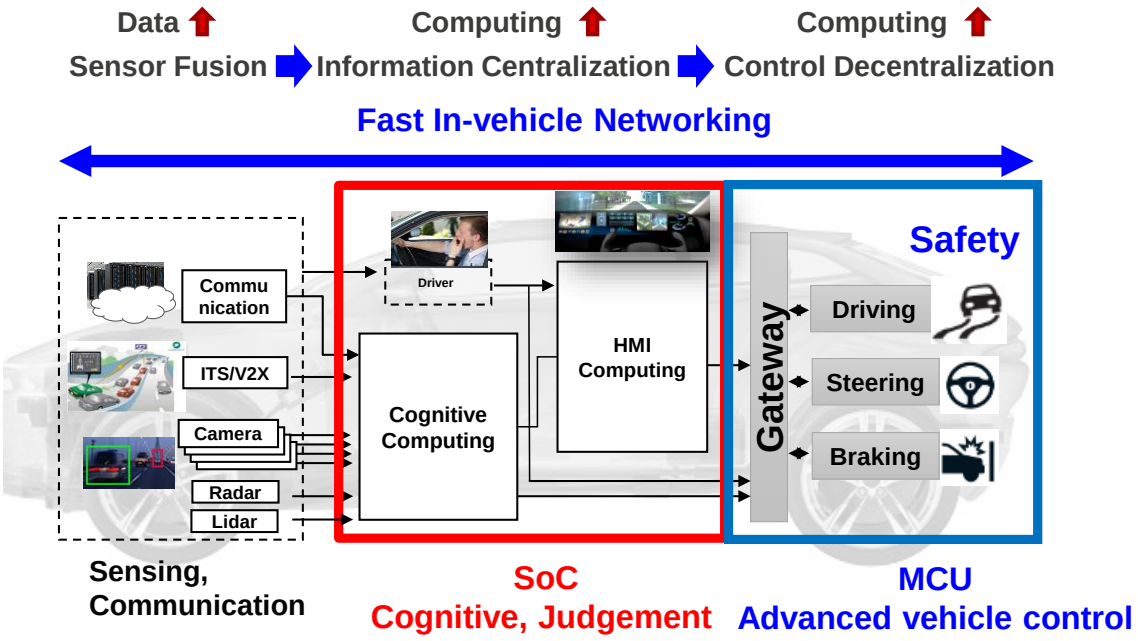


図 1.10: 最新の車載情報/制御アーキテクチャ

新しいプロセッサ・アーキテクチャの開発と評価には，1.1.1 で述べたような高機能化，複雑化する組込みシステムの理解が欠かせない．本節では，組込みシステムの中で，最も高機能化が進んでいる車載電子システムを例にとり，組込みデバイス (LSI) の要件について議論する．

自動運転における電子システムを図 1.10 に示す．システムは，センシングと通信，認識と判断（人/物体認識/衝突検知など），車両制御（走る/止まる/曲がる）の 3 つのシステムから構築される．

自動運転における最終目標は，ドライバーの運転操作からの解放，そして，ヒューマンエラー起因の交通事故リスクの削減にある．自動運転レベルは，レベル 0（ドライバーが全ての運転操作を行う）から，レベル 5（道路や環境条件に関わらず，運転操作を自動化）まで定義されている．現在，実車へのレベル 3〜4 の導入フェーズにあり，このシステムでは，一定の条件下での車両制御を行い，システムの動作が不適切な場合は，非常に短時間で（10 秒以内に）ドライバーへ制御を戻す．また，自動運転車は，道路交通法を遵守し，速度制限を守り，他の車の妨げにならずに追い越しや車線変更を行う．

これらの機能を実現するため，車体周辺には，多くのカメラ，レーダ，LIDAR（レーザー測定），超音波センサーなどが備えられている．これら多数センサーからの道路/環境情報とは，車の情報システム内の自動運転処理向け ECU に提供され，GPS 情報や地図情報とあわせて認識/処理することにより，経路算出が可能になる．高度な車両制御のため，認識処理とヒューマンマシンインターフェース（HMI）コン

ピューティングを緊密に連携させて，外部環境の変化への追随とドライバ状態認識をリアルタイムで行う．さらに，完全自動運転レベルを設計するには，より多くのセンサーを装備する必要がある．このように，自動運転技術は，センサーの融合，情報の集中化，制御の分散化が進むことにより進化していく．

自動運転システムを実現する組込みデバイスは，性能のみならず車載用途で求められる消費電力，安全性を同時に実現することが必要である．これら制約を常に満たしながら，様々な車両設計に対応するために，認識・判断システムは SoC で，車両制御システムは MCU という組込みデバイスを選択する．

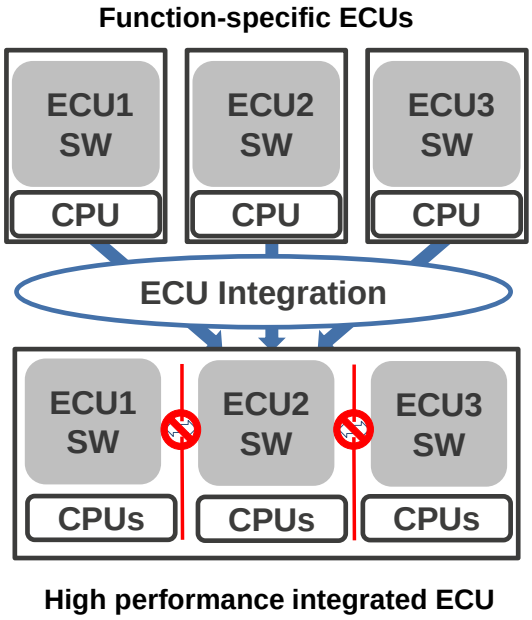


図 1.11: ECU 統合によるソフトウエアコンポーネントの統合

車両制御システムにおいては，多くの機能を 1 つのデバイスに統合するという流れがある（図 1.11）．機能統合については，半導体 Die の集積密度を上げる必要があるが，これは，プロセス技術の向上によって実現することになる．一方，ECU では水冷/空冷やサーマルシートなどの冷却方法を使用できないという理由で，車載 MCU への消費電力に関する要求は，パッケージ熱的制約のもとで数ワットでなければならない．そのため，速度，電力，集積密度の 3 つのポイントに関して従来の 40nm 組込み FLASH プロセスでは不十分であったが，近年，28nm の組込み FLASH プロセスが立ち上がり，集積密度に関しては解決に目途がついてきた（図 1.12）．

また，安全運転のためには高度な車両制御がますます重要になってきており，MCU の統合の流れと併せて，最高の自動車安全度レベル，および演算処理性能の向上が求められてきている．

今後の認識・判断，AI といった処理を行う SoC のアーキテクチャには次の課題がある．

- SoC の高い性能（処理性能，リアルタイム性）の実現

また，今後の車両制御システム向けの MCU のアーキテクチャには次の課題がある．

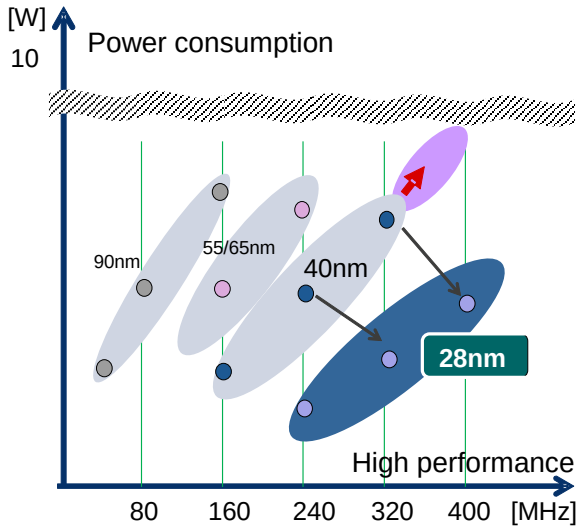


図 1.12: 車載 MCU 向けプロセステクノロジのトレンド

- 制御のインテリジェント化に十分な MCU の演算性能とコストダウン
- 機能統合と最高レベルの高信頼性とリアルタイム性

(1)SoC の処理性能と消費電力の両立

認識・判断システムに採用される車載 SoC は，カメラ・センサーにより取得したデータを対象とした画像認識，AI 処理をはじめとする自動運転向けの広範囲なアリゴリズム処理のために，高い処理性能が求められる．先端プロセスを適用しクロック周波数を高めることでプロセッサの処理性能を向上する一方で，消費電力の増加が課題である．

(2) インテリジェント化に必要な MCU の演算性能

また，高度な車両制御実現のために，先進的な制御や AI 技術を取り込んだ制御方式が考えられている．MCU は，コストと電力の制約のプライオリティが高く，処理性能は抑えられてきたが，それらのシステム要件を満たすためには，これまで以上の処理性能を制約された電力とコストの中で実現することが求められてきている．

(3) 機能統合と最高レベルの高信頼性とリアルタイム性

今日，自動運転システムで使用されている組込みデバイスは，車両システム全体が ISO26262 規格を満たすことをサポートする必要がある [16]．ISO26262 規格では，最も厳しい自動車安全度レベル（ASIL-D）の量的安全要件を規定している．認識・判断システムにおいては，ASIL-B までの対応で留まる一方で，車両制御システムにおいては ASIL-D レベルまでの高い機能安全性能も求められる．車両制御システムでは，ECU の統合により，さまざまな安全度レベルのソフトウェア コンポーネントが 1 つの MCU の中に混在することになり，これらソフトウェアコンポーネント間の干渉を防ぐために，機能安全的観点からリ

ソースの分離は必須となる（図 1.11）.

上記３つの課題をコストと電力の制約の中，解決していく必要がある.

1.5 研究の目的と概要

本研究では，このような背景から，1.4 で述べた高機能な組込みシステム向けの SoC と MCU の２つのカテゴリに対し，消費電力とコストについて高効率なコンピュータ（プロセッサ）技術を提案する．まず，組込みデバイスに対する 4 つの要件（コスト，電力，演算性能，リアルタイム性（応答性能））を満たす組込みプロセッサを実現するには，プロセッシング・エレメントの選択と組みあわせとプロセッシング・エレメントのブラッシュアップを行い，電力効率と面積効率を突き詰めていくことが，急速な応用分野の進化に組み込みデバイスが追従する手段として欠かせない．特に，MCU は SoC と比較して約 1/2 の面積コストが要求されるため，その傾向が顕著である．また，車載システムの高信頼性，高い機能安全性能を満たすプロセッサ技術も車載システム向けの組込みデバイスには必須である．このような組込みシステム向けの SoC および MCU に対して，以下の研究について述べる.

1. 低電力・高性能なヘテロマルチコア SoC の設計：小型のマルチコアと SIMD アクセラレータをコンフィギュラブルに設定できるようにすることで，低コスト，低電力でありながら柔軟な処理を可能にする技術を提案する.
2. 低電力・高性能なインターコネクト SoC の設計：組込み向けマルチチップ構成の実現のためのインタ―コネクト技術として，PCIe を通信 IF とし，インテリジェント化された専用割り込みコントローラと小型マルチコアで構成されるデュアルブロックシステムで通常処理とエラー処理の分離による低電力かつリアルタイム性を満たす方式を提案する.
3. MCU 向け低電力プロセッサの設計：命令セットアーキテクチャと命令フェッチの効率化で，コスト制約の下，いかに電力効率を向上できるか実証する.
4. 車載 MCU 向け高信頼プロセッサの設計: 高機能化 (SIMD 命令，仮想化支援機構）マルチコア CPU と専用ハードウェア (BIST) を組み合わせた，高信頼な MCU 統合方式を実証する.

SoC は，比較的コスト制約が緩いので，消費電力を抑えつつ高い性能要求を満たすために，アプリケーション固有回路とマルチコアで構成される用途別ヘテロマルチ構成を取ることができる．さらに，1 チップで性能要求を満たせず，マルチチップで性能向上を図るための低電力システム実現に向けたインターコネクトの研究開発を行った．一方，IoT 向け MCU には特に厳しいコスト（面積）制約があるため，SoC のように SIMD 型の大規模アクセラレータによる性能向上策を適用できない．そのため，単一 CPU 性能そのものが組込みデバイスの性能を決定することになる．こうしたコスト制約の下での CPU コアのアーキテクチャの改善を行った．車載 MCU では，コスト，処理性能，低消費電力の要件を満たすとともに車載向けの高信頼性（ISO26262 ASIL-D）を満たす方式の研究を行った.

プロセッサ設計の善し悪しは，プロセッシング能力だけで判断されがちだが，実際の組込みデバイスへの適用においては様々な制約によって最適な解が変わる.

1.6　本論文の構成

本論文の構成は次の通りである．第 2 章では，低電力・高性能なヘテロマルチコア SoC Orochi の設計について述べる．ここでは，小型のマルチコアと SIMD アクセラレータをコンフィギュラブルに設定できるようにすることで，低コスト，低電力でありながら柔軟な処理を可能にした．第 3 章では，低電力・高性能なインターコネクト SoC の設計について述べる．組込み向けの高性能なインターコネクトを実現するために，インテリジェント化された専用割り込みコントローラと小型マルチコアで低電力でかつリアルタイム性を満たす方式を提案した．第 4 章では，MCU 向け低電力プロセッサの設計について述べる．命令セットアーキテクチャと命令フェッチの効率化で，コスト制約の下，いかに電力効率を向上できるか実証した．第 5 章では，車載向けの高信頼性 MCU の研究について述べる．最後の第 6 章で結論を述べる．

本論文は，組込みプロセッサの高演算性能化，低消費電力化，低コスト化の実現に伴う諸問題に対して，論理的，回路的，システム的にいくつかの解決方法を提案するとともに，実際の車載向け組込みプロセッサ設計に適用してその有効性を実証する．

第 2 章

低電力・高性能なヘテロマルチコア SoC Orochi の設計

画像認識は近年，家庭用電化製品における主要技術となっており，現在のデジタルスチルカメラへの顔認識機能を搭載はその代表的な例である．画像認識は，従来はパーソナルコンピュータで処理を行うほど，非常に大量な処理を必要とする．高性能と低消費電力を兼ね備える高度な画像認識技術用 SoC の需要は高い．さらに，画像認識ばかりでなく，推論，計測，制御及びセキュリティなどの機能をも使用した大規模で複雑なアプリケーションには，将来，より高い処理能力を持った組込み向け SoC が欠かせなくなる．

画像処理，AI 処理の向けの SoC として，電力消費量を抑えながら処理性能を実現するために，マルチコア CPU とニアメモリ・単一命令/複数データストリーム（SIMD）アーキテクチャを組み合わせた低電力・高性能なヘテロマルチコア SoC Orochi を設計開発した．

本章では，電力を抑えながら処理性能を向上させるための要素技術，1) マルチコアでは面積と電力が n 倍化するため，シンプルで小面積な CPU 採用とコア当たりの電力を下げるための可変レイテンシキャッシュと分割マッチライン方式 TLB，2) SIMD 機構としてニアメモリで高速動作する超並列アクセラレータ MBMX，3) それらをモード毎に LSI の機能構成を変化させる，コンフィギュラブル・ヘテロジニアス・マルチコアシステム，4) 低消費電力化技術として，ディレイ・マネジメント・ネットワークについて述べる．Orchi の実装により，フレキシビリティを保持したまま，電力効率（Performance/W）を上げることができることを実証した．

2.1　ヘテロマルチコアによる高効率化へのアプローチ

複数のプロセッサを 1 つのチップに統合することは，高性能，低消費電力そして短期開発の要求を満たす最善策のひとつと考えられてきた．すでに多くの種類のマルチコアプロセッサ・チップが開発されている [17][18][19]．単一命令，複数データ (SIMD) アプローチは，並列処理で大量のデータ処理を効率よく行うことから，スカラー処理を高速に行うための複雑な構造が単純化されるので，チップ面積や消費電力という点で，マルチコアプロセッサ (MIMD) に比べて有利である．その一方で，SIMD アプローチは，汎用処理や制御処理といった複雑な処理には向かない．

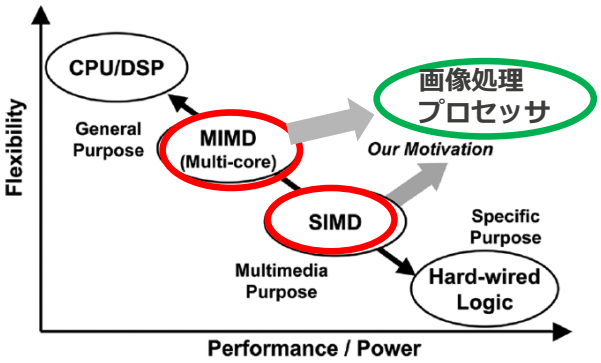


図 2.1: 低電力・高性能なヘテロマルチコア SoC

認識，計測といったマルチメディア・アプリケーションは，2 種類の処理から構成されている．ひとつは，データ処理であり，もうひとつは，制御処理である．これらのアプリケーションを実行する際に，性能を引き出す最も効率的な方法は，ヘテロジニアス・システム上で様々なアプリケーションに適した，異なるハードウエアを利用することである．この方法は，電力効率と面積効率という観点で単一プロセッサシステムを使うよりも有効である．このアプローチのためには，実行するアプリケーションを次の 2 種類の処理に分割することが必須である．ひとつは，SIMD アーキテクチャの並列アルゴリズムに適したデータ処理であり，もうひとつは，単純でない制御処理である．マルチスレッド並列処理により，システム性能を向上すると共に，ヘテロジニアス・システムを用いて，処理負荷のより効果的な割り付けを行い，性能向上を図ることができる（図 2.1）．

チップ内の CPU の数が増加するにしたがって，電力，熱，ノイズ低減を維持するチップ全体の管理ユニットが必要となってくる．今や，多くのチップが内部モニタリング機構を採用している．Orochi には，複数の温度センサーとして機能するデジタル遅延モニタ，クロック遅延アジャスター及び，内部制御用の MCU を搭載し，チップ内部状態をチェックすることで，クロック位相と CPU 処理負荷の調整を行うことができる．

2.2　CPU の低電力化技術

2.2.1　可変レイテンシ・命令キャッシュ

キャッシュ・メモリは CPU コアの構成要素の中で最も電力を消費する部分である．マルチコアでは，複数のキャッシュ・メモリが同時に動作するため，この電力消費を小さく抑えることが重要になる．命令実行のスループットを下げずに消費電力を削減するため，可変レイテンシ方式の命令キャッシュを開発した．図 2.2(a) に示すように，ブランチやジャンプなどの分岐命令後のキャッシュ・メモリアクセスと命令プリフェッチによるキャッシュ・メモリアクセスとで，キャッシュ・メモリの動作を変えている．

分岐命令が実行された場合，パイプライン内に充填されている後続の命令を破棄し，新たに分岐後の命令列でパイプラインを埋める動作が発生する．この分岐のペナルティを最小にすることで，処理のス

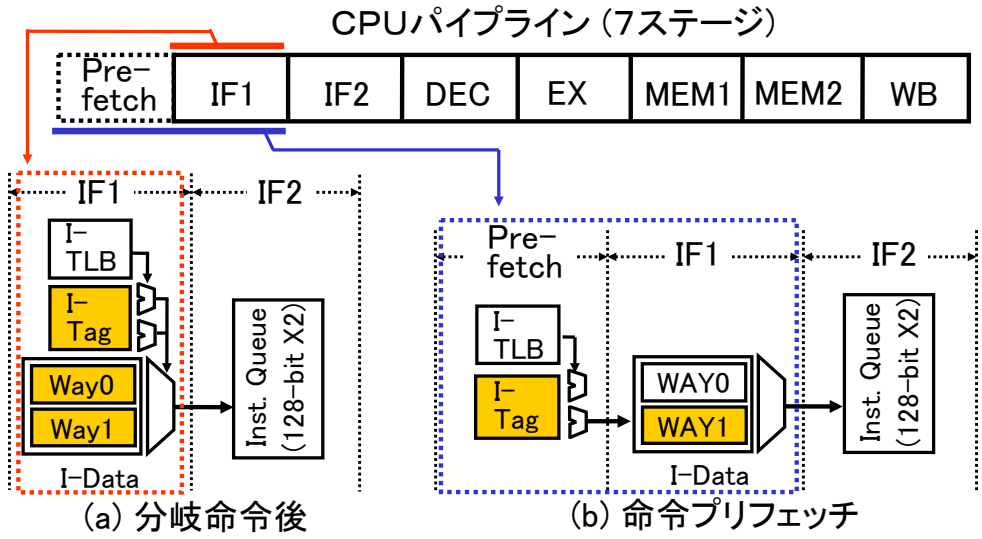


図 2.2: 可変レイテンシキャッシュ

ループットは向上する．そのため，分岐命令後の命令キャッシュ・アクセスは，図 2.2(a) に示すように，TLB，タグメモリ，データメモリの両ウェイを同時に読み出し，キャッシュ・アクセスを 1 サイクルで終了させている．このとき，タグメモリのヒット/ミス判定を待たずにデータメモリの両方のウェイを読み出しているため，比較的大きな電力を消費することになる．

一方，命令プリフェッチの場合は，メモリ上に連続して配置された命令を順次命令キューに取り込んでいくため，次に読み出される命令のアドレスが事前にわかっている．この場合は図 2.2(b) に示すように，まず，1 サイクル目に TLB とタグメモリを引き，命令キャッシュ内に所望の命令が含まれているかどうかをチェックし，含まれている場合には，該当するウェイのメモリ・プレーンのみ活性化させ，読み出しを行う．キャッシュミスの場合は，データメモリへのアクセスそのものを抑止する．

このように，可変レイテンシ制御を行うことにより，分岐のペナルティを抑えることと，平均消費電力の削減を両立している．すべてのキャッシュ・アクセスを 1 サイクルで処理する場合に比べて，キャッシュ・メモリの平均消費電力を 40% 削減できる．

2.2.2 分割マッチライン TLB による低電力化

フルアソシアティブ方式の TLB では, 仮想アドレスの TAG アレイは CAM(Content Addressable Memory) で構成する．この場合マッチしなかったエントリのマッチラインがディスチャージされるが，通常，マッチするのは 1 エントリだけであり，残りのマッチラインはアクセスの都度，充放電を繰り返すだけである．命令フェッチ時に常時アクセスが行われる TLB について，分割マッチライン方式を開発し，TLB の高速化と，電力削減の両立を図った．

CPU コアでは, TLB の仮想アドレスタグを ASID(Address Space Identifier) 部分と, VPN(Virtual Page Number) 部分に分割し，独立した CAM アレイとして構成している．仮想アドレスのうち，ASID の部分は OS によってタスクが切り替えられたときのみ変更される．タスク・スイッチが発生した場合，および新しい

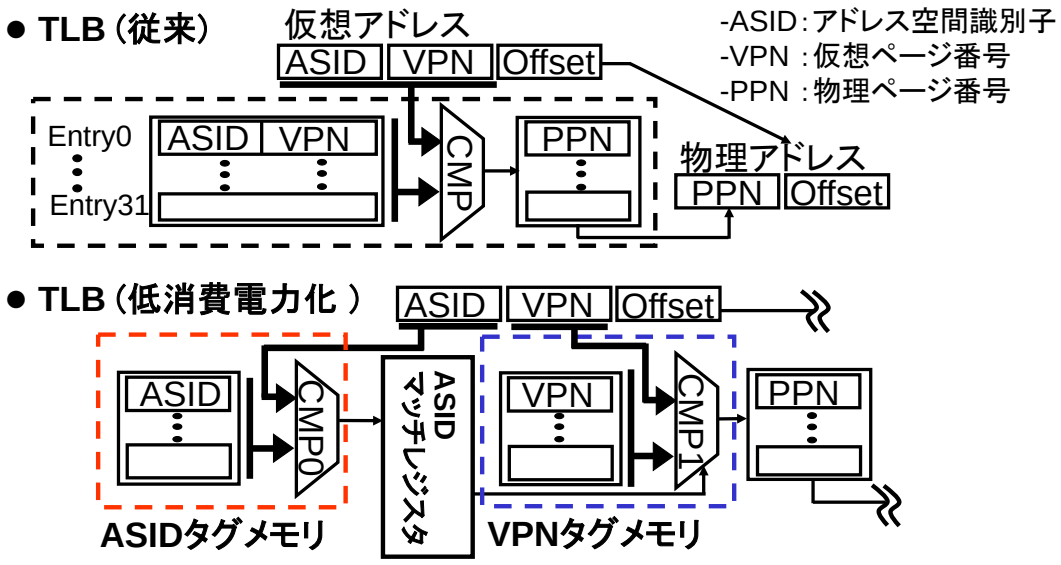


図 2.3: 分割マッチライン TLB

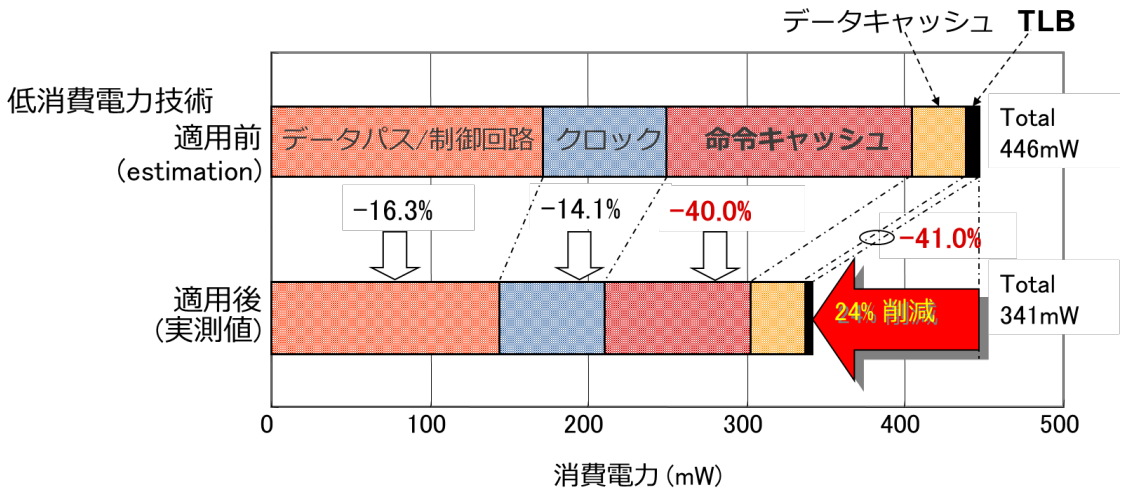


図 2.4: 消費電力の削減効果

TLB エントリが登録された場合にのみ，ASID に対応する CAM を動作させ，各エントリの比較結果を図 2.3 に示すように，ASID マッチレジスタに登録する．通常のメモリアクセスでは，マッチレジスタの内容により，現在の ASID の下で有効なエントリのみ，CAM の比較動作を行わせる．

この方式では，次のような利点がある．(1) 現在の ASID と異なる ID を持つエントリは比較されず，マッチする可能性のないエントリの比較で電力を消費することがない．(2) 通常のアクセスでは VPN のみを比較するため，マッチラインの負荷が軽くなり，高速動作が可能である．従来の一本のマッチラインに ASID，VPN の CAM がすべて接続されている構成に比べて，41 %の電力削減が可能である．

2.2.3 低消費電力設計の効果

マルチコアでは，複数の CPU コアが搭載される．このため，低消費電力化を実現するためには，CPU コア単体の消費電力を削減することが重要になる．図 2.4 に CPU コアの消費電力を示す．測定には，Dhrystone ベンチマークプログラム（キャッシュヒット率 100%）を使用した．150nmCMOS プロセス実装の CPU コアの消費電力は，600MHz，1.5V 動作，室温時に 341mW である．命令キャッシュと命令 TLB の低消費電力設計により，従来設計より 24% の消費電力が削減された．

2.3 ニアメモリで高速動作する超並列アクセラレータ：マルチバンク・MX プロセッサ MBMX

MX プロセッサは大容量の画像データ処理を効率よく行うことを目的に設計した SIMD タイプの超並列プロセッサである．第一世代の MX プロセッサは，小面積実装を目的にハードマクロ設計で開発を行った [20]．この MX プロセッサを SoC 向けに最適化するための検討を行い，合成可能な RTL 設計手法で再設計を行ったのがマルチバンク・MX プロセッサ（MBMX）である．

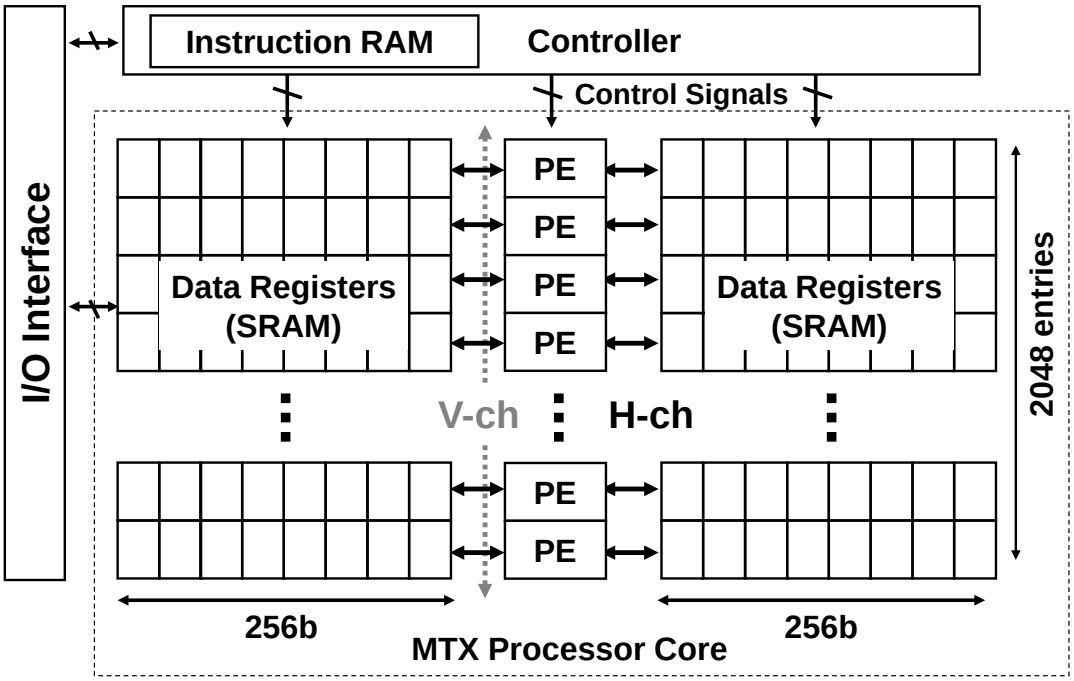


図 2.5: MX プロセッサ構成

2.3.1　MX プロセッサ

図 2.5 に SIMD アクセラレータである MX プロセッサの概要を示す．MX プロセッサは，データの超並列処理を実行する演算アレイ部，並列演算を制御するシーケンスコントローラ部，および，データの入出力を制御するバス・インタフェース部から構成される．演算アレイ部は，細粒度（2 ビット粒度）の 2048 個の PE と 512 ビット x 2048 ワードのデータレジスタから成る．第一世代 MX プロセッサのデータレジスタは，エリア効率を高めるためにシングルポート SRAM セルで構成した．

PE は，各 PE のサイズを最小化するために，メモリ設計手法との親和性の高さも考慮し，2 つの全加算器といくつかの論理回路を含む 2 ビット粒度の PE 構成を採用している．SIMD アーキテクチャを採用する場合，必要なシリコン面積と性能のトレードオフを考慮すると，PE のサイズと並列性には強い相関関係がある [21]．PE を 2 ビット構成にすることにより，SRAM の周辺回路に多数の PE を統合することができる．これにより，単位シリコン面積の並列処理が最大化され，優れたエネルギー効率と面積効率を備えた高性能動作が実現できる．

図 2.5 に示すように，データ処理用に 2 方向のチャネルがある．1 つは，データレジスタの配列行列と PE を接続する水平チャネル（H-ch）であり，もう 1 つは，PE 間の柔軟なデータ通信を実現する垂直チャネル（V-ch）である．MX プロセッサの高い処理性能は，H-ch と V-ch の連携により実現することができる．MX プロセッサの設計コンセプトは SIMD アーキテクチャに基づいており，すべての PE と SRAM データレジスタは，コントローラ部によって生成された制御信号により同じように動作する．コントローラ部は，8KB の命令 RAM に格納されている命令をデコードすることで制御信号を生成する．MX プロセッサのすべての動作は，命令 RAM にロードされたシーケンスプログラムによって制御を行う．

（1）H-ch を使った基本加算演算

H-ch を利用した加算演算の動作例を図 2.6 に示す．H-ch の演算単位（1 つのエントリ）として，データレジスタアレイの 1 列と 1 つの PE を 1 つのグループと考える．各 PE には，データレジスタからの読み出しデータを一時的に格納する 2 ビットの一時レジスタと ALU が装備されている．PE は 2 ビットであるため，データレジスタに格納されている多ビットのオペランドはビットシリアル方式（2 ビットごと）で処理される．

サイクル k では，左サイドのオペランドの 2 ビット（LSB）が読み取られ，一時レジスタに格納される．次のサイクル k+1 で，右サイドのオペランドの 2 ビットがデータレジスタから読み取られ，一時レジスタに格納されたデータと加算される．PE の出力データは，SRAM のリード・モディファイ・ライト動作を利用して，k+1 の同じサイクル内でデータレジスタに書き戻される．

2 オペランドタイプの演算方式を採用しているので，各 SRAM メモリセルはアキュムレータのように動作する．また，図 2.5 に示す 2 バンク構造により，1 サイクルあたり最大 2 ビットの加算まで処理スループットが向上する．上記演算手法で，2048 個の 16 ビット加算を 2048 個のエントリで並列に実行した場合，MX プロセッサはすべてのデータを 10 サイクルで処理できるため（パイプライン操作のオーバーヘッドを含む），1 つのエントリにあたり，約 0.005 サイクル（10 サイクル/2048 エントリ）で 16 ビット加算が処理されたことになる．PE と SRAM メモリの実際の実装は完全に対称的であり，一時レジ

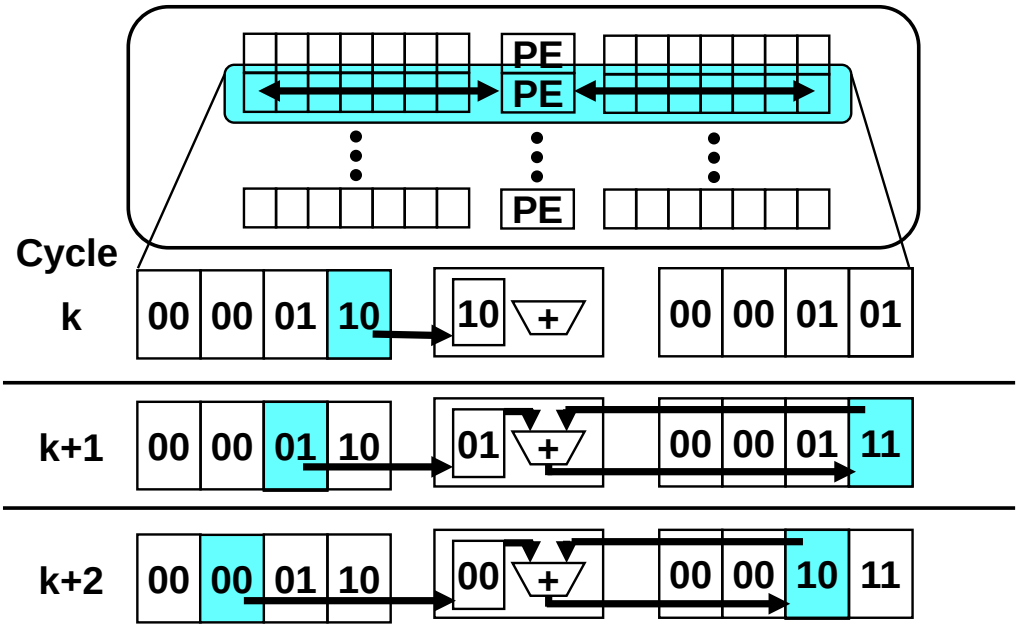


図 2.6: MX プロセッサの基本動作例（8 ビット加算）

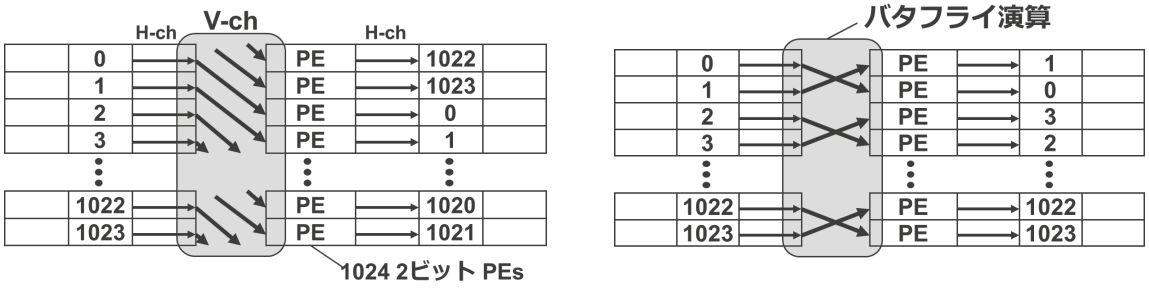


図 2.7: V-ch によるエントリ間データ転送

スタと PE は，2 つのデータレジスタアレイの間で接続されている．これにより，処理スループットを向上させながら面積効率を維持している．

(2) V-ch を使ったエントリ間データ転送

複数のデータエントリと PE を使用して複雑なアルゴリズムを処理するには，PE 間でデータを通信するための効率的な仕組みが SIMD プロセッサにとって必要である．図 2.5 の V-ch はこの目的のために設計されており，V-ch を使用することでエントリ間のデータ転送が可能である．図 2.7 に，特定のデータを隣接するエントリに格納する概要を示す．

超並列プロセッサ用にいくつかの種類の PE ネットワークが報告されているが [22][23][24]，これらの回路はかなりの面積オーバーヘッドがあるか，操作が複雑すぎて単純な SIMD 制御信号で制御できないといった課題がある．MX プロセッサでは単純なシフトレジスタ型ネットワークを採用した．シフトレジ

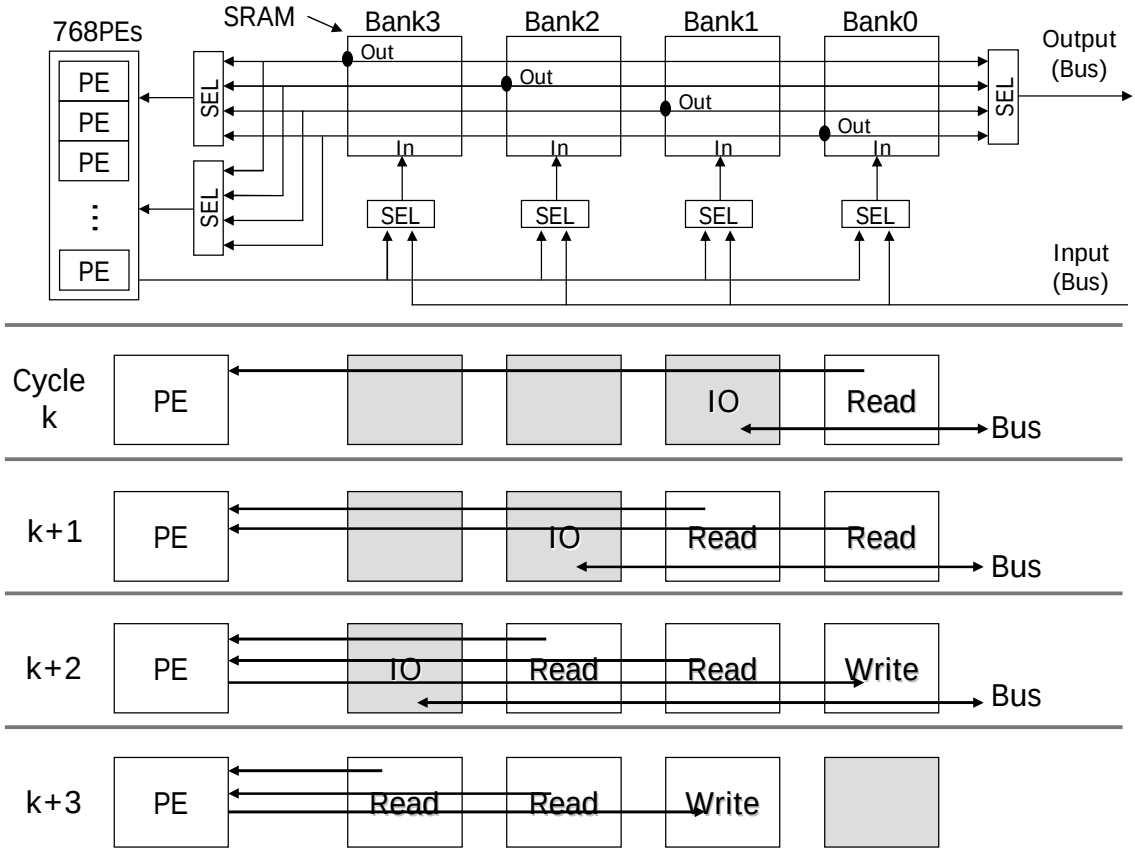


図 2.8: マルチバンク化した MBMX のブロック図

スタ型のネットワークでは，長距離のデータ移動を行うには多くのサイクルが必要となる．MX プロセッサでは，1 回のシフト量を +/- 1/2/4/8/256 などのいくつかの種類をサポートすることで，長距離のデータ移動のサイクルオーバーヘッドを削減している．柔軟な処理を可能とるす V-ch を備えたシンプルな PE ネットワークは，畳み込み，FFT などの多くのアプリケーションに非常に効果的である．

2.3.2　4 バンク構成による効果的なデータアクセス

第一世代 MX プロセッサは，主にポータブルなアプリケーション向けであり，リード・モディファイ・ライト動作を 1 サイクルで実行し，低い周波数で動作することを想定して設計を行った．しかしながら，3 次元計測のようなより複雑化したアプリケーションに必要な処理性能を，第一世代 MX プロセッサでは提供できていなかった．

第一世代 MX プロセッサの性能向上のため，性能劣化要因の解析を行った結果，MX プロセッサにはまず，動作周波数の向上が必要であることと，並列処理の演算を行っている時間よりも入力・出力データの転送に多くの処理時間が使われていることが分かった．さらにこの解析から，動作周波数の向上のネックとなるクリティカルパスは，MX プロセッサ内の SRAM セルへのリード・モディファイ・ライト動作であることが明らかになった．この問題を解決するために，MBMX では，リード・モディファイ・ライ

ト動作をリードオペレーションとライトオペレーションの 2 個に分割し，500MHz で動作可能とした．

一方, 1 サイクルでのリード・モディファイ・ライト動作と同等なパフォーマンスを得るために, MBMX の SRAM は 4 バンク構成とした (図 2.8)．MBMX では処理をパイプライン化することで，リード・モディファイ・ライト動作は 2 サイクルで動作するにも拘わらず，MBMX のスループットは第一世代の MBMX と同等である．SRAM へのアクセスパターンを考慮して，MBMX はライト要求とリード要求を同時に発行できるようになっている．リード要求があるバンクに発行されると，ライト要求は他のバンクに同時に発行される．2 個の連続したリード・モディファイ・ライト要求は，異なる SRAM バンクに割り当てられ，連続アクセスが衝突することはない．これにより，リード・モディファイ・ライト処理のスループットを 1 サイクルにすることが可能となった．

4 バンク SRAM の構成をとることで，MBMX はシングル・ポートの SRAM のみで構成が可能であり，性能要求も満たすことができる．また，MBMX 内の SRAM はコンパイルド・SRAM で構成することも可能である．この構成をとった場合の最大のメリットは，MBMX が合成可能なプロセッサとなり，PE の数を搭載する SoC に合わせて変更が容易になった点である．

さらに，4 バンク構成のうち，演算中は 3 バンクしか利用されない．従って，使用中でない残り 1 個 SRAM バンクをアイドル状態へ遷移させることで，消費電力を低減できる．図 2.8 には MBMX のパイプライン処理フローも図示している．演算時，1 個の 2 リード/1 ライト動作は，3 バンクを使用する．影を付けた部分は，アイドル状態にあるか，データ I/O 動作中であるバンクを示している．

2.3.3 データ転送と演算の並列処理

第一世代 MX プロセッサは，プロセッサ外とのデータ転送（データ I/O 動作）と演算の並列実行をサポートしていなかった．この点を改善するために，MBMX は外部メモリから MBMX へのデータ転送，あるいは MBMX から外部メモリへのデータ転送を，演算と並列に効果的に行う仕組みを搭載した．演算のための 2 リード/1 ライト動作が SRAM の 3 バンクを使用している間に，残り 1 バンクはデータ転送を専用で行うことが可能である (図 2.8)．これにより，第一世代 MX プロセッサのトータル・システム性能は，演算時間とデータ I/O 処理時間の和で決まっていたところが，MBMX では，演算時間でトータル・システム性能が決まることになる．MBMX は，最大 2 倍 MX より高速に処理を行うという結果を得ている．MBMX は大半のマルチメディア・アプリケーションに見合う充分なデータバンド幅を提供できるだけでなく，アプリケーションが，データ転送のために，ある特定のバンクにアクセスする必要があれば，演算と SRAM アクセスを停止する機能をも搭載している．この機能は，画像処理を取り扱う際に，特に，データ I/O 処理時のタイミング制約が厳しい場合に効果的である．

2.3.4 位相限定相関法による画像認識への応用

位相限定相関法（Phase-Only Correlation:POC) は，高精度な画像マッチング手法の一つであり，2 枚の画像間の平行移動量，回転角度，類似度を高精度に推定することができる [25]．POC は画像の位相情報を扱っており，ノイズはカスレなどの劣化に強くロバストなアルゴリズムである．また，解像度の低いセンサを用いた場合でも高いマッチング性能を有する．このため，POC は高精度な画像対応付けが必要と

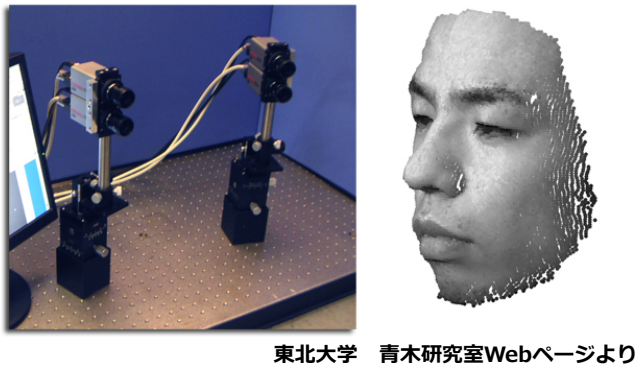


図 2.9: 位相限定相関法による 3 次元計測

なるマシンビジョン，医療用画像のレジストレーションや，高精度受動型 3 次元計測（図 2.9）など，様々な用途に利用されている．POC を用いた平行移動量推定は，2 枚の画像が与えられたとき，それぞれの画像の 2 次元 FFT を計算する．この結果を振幅成分で正規化した上で積をとったものに，2 次元逆 FFT をかけて位相限定相関関数（POC 関数）を計算する．画像が類似している場合，POC 関数は極めて鋭いピークを示す．このことから，この相関ピークの高さは画像の類似の尺度として有用である．また，ピークの座標は，2 画像の相対的なずれ位置に対応している．

128x128 ピクセル，8 ビットグレースケール画像に対する平行移動量推定と画像の類似度を計算を，MX プロセッサに実装した場合の処理時間は 22.8ms であり，リアルタイム処理が充分に実現可能である．また，この値は，汎用組込み RISC プロセッサ (SH-2A) を用いた場合と比較して約 115 倍高速化されている．

2.4　遅延管理ネットワークによる電力マネージメント

2.4.1　温度測定のためのデジタル遅延モニタ

チップ温度を測るために，アナログ・サーモメータはよく使われてきたが，面積が大きいため 1 チップの中にたくさんのアナログ・サーモメータを搭載することは難しい．この問題を解決するために，新しく小型のデジタル遅延モニタを開発し，この Orochi の各プロセッサに 1 個ずつ搭載した．デジタル遅延モニタはチップ全体を管理する場面で，より効果的に使用される．その理由は，温度，プロセス技術や供給電圧に関する調整を全く必要としないからである．つまり，デジタル遅延モニタは，チップ温度を直接計測するのではなく，遅延時間を測定する．また，アナログ回路と違い，デジタル回路で構成されているため標準セルと混在が可能であり，システム上で測定したい対象の近くに自由に配置することができるからである．

この小型のデジタル遅延モニタは，高精度モニタと粗精度モニタの 2 つのモニタで構成されている（図 2.10）．高精度モニタは，High レベルの信号が 1 サイクル間で何個のインバータを通過したかを測定できるように，15 個のフリップ・フロップと 15 段インバータ・チェインで構成されている．粗精度モニタは，

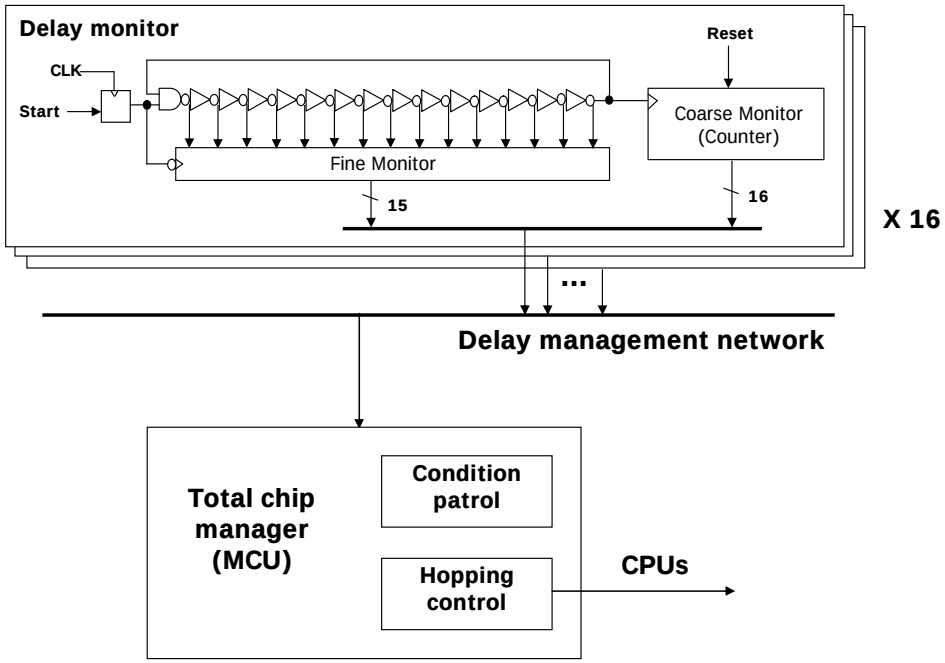


図 2.10: 遅延モニタと遅延管理ネットワーク

16 ビットのカウンタで，高精度モニタのインバータ・チェインのループ回数をカウントする．デジタル遅延モニタの出力は，高精度 15 ビットデータと粗精度データ 16 ビットデータからなる．統合チップ管理ユニットは，デジタル遅延モニタの出力を読み出し，遅延時間に変換する．デジタル遅延モニタの計測精度は，20psec〜50psec である．

2.4.2 遅延管理ネットワークとタスクホッピング機能

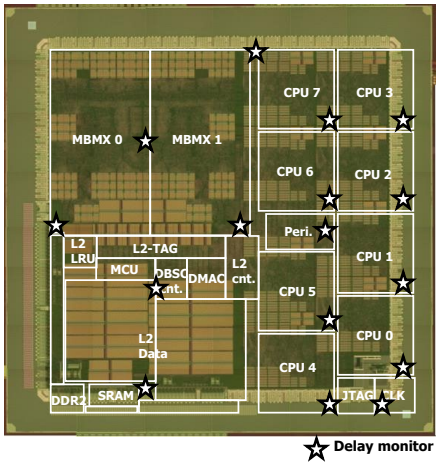


図 2.11: 遅延モニタの配置

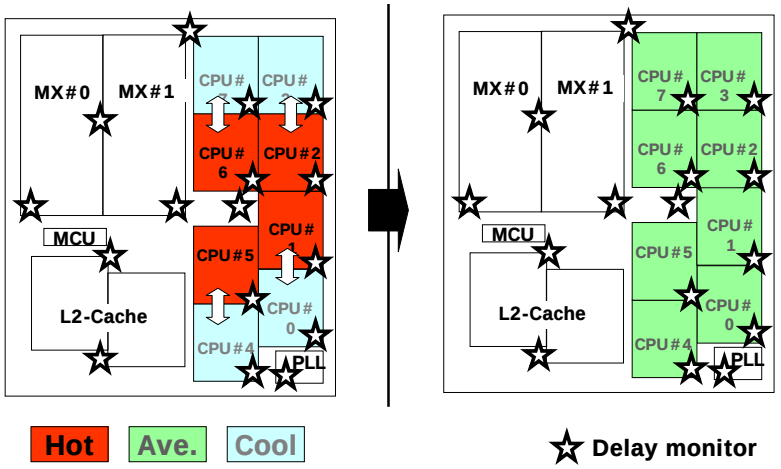


図 2.12: タスクホッピングの例

表 2.1: 電力制御技術の比較

Technique	FV control	Task hopping
Detect time	Several ns	Several ns
Control time	Several μs	1μs (Cache Copy)
Area	0.1mm^2	0.025mm^2
Design methodology	Analog hardmacro	Synthesizable

1 個の SoC 上に多くのプロセッサを搭載すると，仕事量に依存した複数のホットスポット（温度が高い場所）が出現することになる．適切な温度分布を維持するために，ホットスポットの位置を，統合チップ管理ユニットにフィードバックする必要がある．図 2.10 に，遅延管理ネットワークの構成を示す．16 個の遅延モニタ，統合チップ管理ユニットとして管理プログラムが動作する制御用 CPU ユニットで遅延管理ネットワークは構成されている．図 2.11 には，実際に Orochi で実装したデジタル遅延モニタの場所を示す．

この遅延管理ネットワークを用いて，我々が定義した「タスクホッピング」と呼ぶ機能を実現することができる．タスクホッピング機能は，個々の CPU の温度データを監視し，温度の高くなっている CPU の負荷は高いと判断し，その負荷が高くなっている CPU の処理を他の負荷の低い CPU に再分配する機能である．この機能により，高温度になった CPU の偏った熱を分散させ，適度な熱分布を保持することで最高性能を得ることができる．具体的には，性能劣化なしで，より低い VDD を設定することができ，低消費電力化が可能となる．このタスクホッピング機能は，ハードウエアとソフトウエアの両者により実現している．図 2.12 で，タスクホッピング動作の例を示す．統合チップ管理ユニットは，遅延管理ネットワークから計測された遅延時間を受け取り，各位置に対して，条件フラグ（hot,average or cool）を生成し，タスクの温度を管理する．

表 2.1 に，2 種類の電力制御技術の比較を示す．一つは，周波数-電圧 (FV) 制御技術で，周波数と供給電圧をチップの処理量に応じて制御する．温度は，アナログの温度センサで計測されるので，回路面積は

表 2.2: Orochi チップ諸元

Processor	M32R(1GHz) x 8 I$:8kB, D$:8kB, LM:8kB, MMU, FPU M32C(500MHz) x 1 MBMX(38GOPS@500MHz) x 2
Memory	L2-cache:512kB Internal SRAM:32kB
DRAM I/F	DDR2 I/F x 1 SDRAM I/F x 1
Bus Process	Multi-layer bus (4-layer:4GB/s) Pipelined bus (8GB/s for read, 2GB/s for write) Fly-by bus (1GB/s) 90nm Generic CMOS
Chip size	6.35nm x 6.35nm
Package	729FCBGA
Power	5.0W

大きい．もう一方はタスクホッピング技術である．CPU に割り当てられたタスクを置き換えるときには，データをメモリへコピーバックしなくてはならない．タスク置き換えの際発生する，8-KB キャッシュ内のデータのコピーバックには，1μsec かかる．FV 制御技術（給電圧とバックゲート・バイアス制御のために，複雑な制御が必要となる）と比較して，表 2.1 に示すように，本提案技術は，より小さい面積，より速い応答を実現している．この提案技術を使うことにより，供給電圧は，2% 下げることができる．

2.5 Orochi の実装と評価

2.5.1 チップ概要

8 CPU の SMP 構成の汎用マルチコアと MBMX からなるヘテロジニアス・マルチコア SoC Orochi の実装評価として，90nm CMOS プロセス 8 層銅配線で試作し，CPU の 1GHz 動作と 76GOPS の MBMX の 500Mhz 動作を確認した [26][27].

表 2.2 に，チップの主な機能ユニットの仕様を示す．試作した Orochi は，3 種類の合成可能なプロセッサを実装している．複雑な制御/処理を行うための 8 個のマルチコア CPU(M32R) と大容量のマルチメディア・データを処理する 2 個の MBMX とチップ全体制御のための制御用 CPU(M32C) である．図 2.13 に，試作チップのブロック図を示す.

大規模なスレッドレベル並列処理機構 (TLP) を活用しその能力を強化するには，1 チップに搭載する CPU コアの数を増やすことが必要である．このアプローチをより効果的に行うには，シンプルで小規模な CPU コアを単一チップに数多く集積することである．Orochi の汎用プロセッサブロックは，8 個の CPU

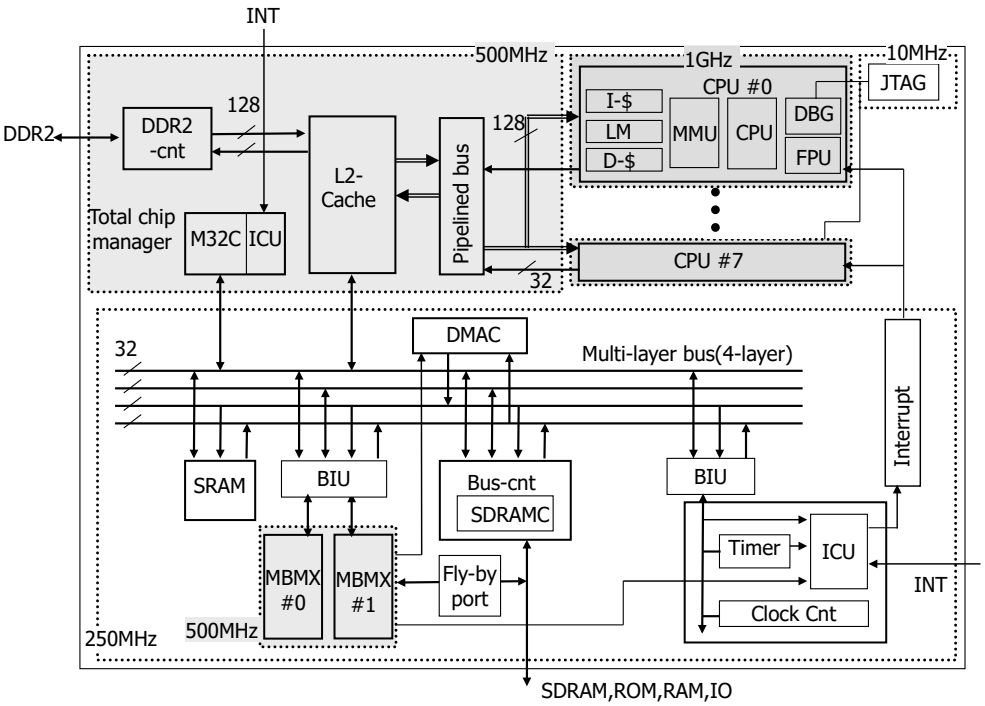


図 2.13: Orochi ブロック図

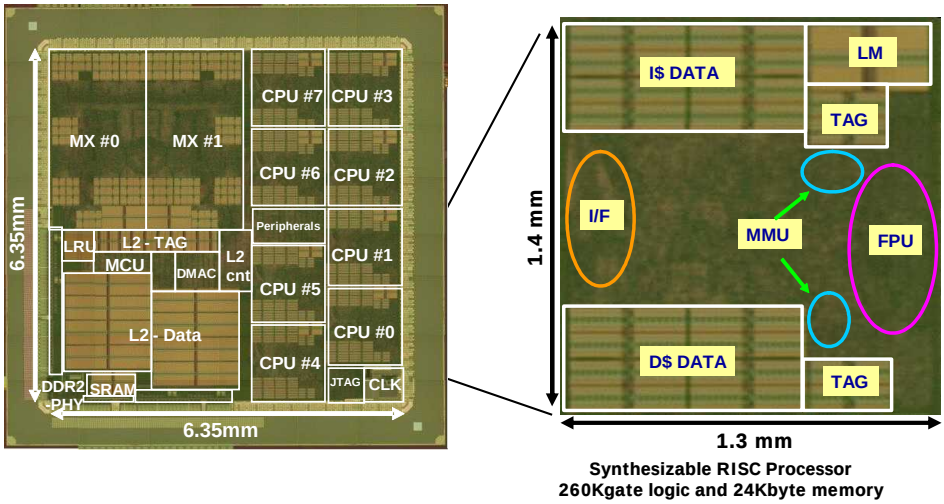


図 2.14: デバイスと CPU のチップ写真

のマルチコア構成とした．各 CPU コアは 32 ビット M32R RISC アーキテクチャ [28] をベースにしたものである．CPU コアは 1-GHz 動作する合成可能なプロセッサで，倍精度浮動小数点ユニット（FPU），メモリ管理ユニット（MMU），デバッグモジュールを備えている．CPU コアは，7 ステージの Dual-issue パイプライン構成で，MMU には 32 エントリのフルアソシアティブ TLB を搭載し，命令とデータ，それぞれ 8K バイトのキャッシュメモリを搭載する．データキャッシュは，MESI プロトコルのコヒーレンシ制御機能を持つ．さらに，8-KB のローカルメモリを各 CPU コアに搭載している．

8 個の CPU コアは，512-KB L2 キャッシュを共有することで，内部バストラフィックを削減している．各 CPU コアをつなぐ共有パイプラインバスのバンド幅はリード動作時 8GB/s，ライト動作時は 2GB/s である．MBMX は，768 個の 2bit プロセッシング・エレメント (PE)，32K バイトの命令メモリ，4 バンク構成の 96K バイトのデータメモリから構成されている．3 種類のプロセッサは，チップ上の広帯域のマルチレイヤ（4 層）・システムバスで相互に接続されている．8 個の CPU コアは L2 キャッシュを経由して，マルチレイヤ・システムバスと接続されている．内蔵共有 SRAM，内部 I/O，SDRAM 及び MBMX は，全て，このマルチレイヤ・システムバスによって接続されている．この 4 層レイヤ・システムバスは，これらの資源に並列にアクセスするのに充分な転送レート (4GB/s) を備えている．

MBMX の性能を引き出す鍵となるのは，データの入出力性能である．MBMX の性能を最大限に利用するために，特別に広帯域データ転送機構のために Fly-by バスを採用し，個々の MX プロセッサと外部メモリ間のデータ転送能力を向上させた．Fly-by バスはシングルアドレス転送技術を使って，1 バスサイクルの間にリードあるいはライトトランザクションを発行する．したがって，Fly-by バスは 1 サイクル当り 1 データパケット (32bit) という高速な転送が可能であり，最大バスバンド幅は 1GB/s である．Fly-by バスと DMAC を使うことで，大量のデータを外部メモリから MBMX へ転送することが可能である．

Orochi は，さらに，16 個の小型デジタル遅延モニタを搭載している．モニタ出力は制御用小型 CPU(M32C) で管理されており，遅延管理ネットワークを構築でき，タスクホッピング機能の利用が可能である．

図 2.14 はチップと CPU の写真である．CPU は，1.3mm x 1.4mm で面積はわずか 1.82mm2 と小さい．CPU はコンパイルド・SRAM から構成される 3 個のメモリ部分があり，その他の部分は論理合成により実装した．

2.5.2 CPU グルーピング機能を持つパイプラインバス

8 個の CPU は，スループットを向上させるためにパイプライン化された共有パイプラインバスで接続されている．パイプラインは，アービトレーションステージ，スヌープステージ，スレーブアクセスステージ，およびデータステージの 4 つのステージで構成されている（図 2.15）．この共有パイプラインバスは，キャッシュコヒーレンシの機能を持つ．キャッシュコントローラーとバスアービターは，8 つの CPU コアのキャッシュコヒーレンシを維持するために，MESI キャッシュコヒーレンシプロトコルをサポートしている．

各 CPU には 2 セットのバスアクセスポートがあり，1 つは命令フェッチ用，もう 1 つはロード/ストア用である．したがって，内部 CPU バスには，複数の要求を起動する 16 個のバスマスターがある．16 バスマスターの要求は，調停段階で調停される．これらのバスマスターの要求の優先順位は，ラウンドロビン方式によって決定される．スヌープステージは，キャッシュの一貫性を維持するために使用される．スレーブアクセスステージとデータステージは，スレーブデバイスにアクセスするために使用される．特定のバス操作に不要なサイクルを排除するために，一部のパイプラインステージはスキップされる．データステージはオペランドストアから削除される．スヌープステージは，命令フェッチと非キャッシュ領域アクセスで削除される（図 2.15）．共有パイプラインバスは，キャッシュからキャッシュへの転送，投機的実行，および分岐予測をサポートする専用ハードウェアは採用していない．これらの追加の複雑な回路に

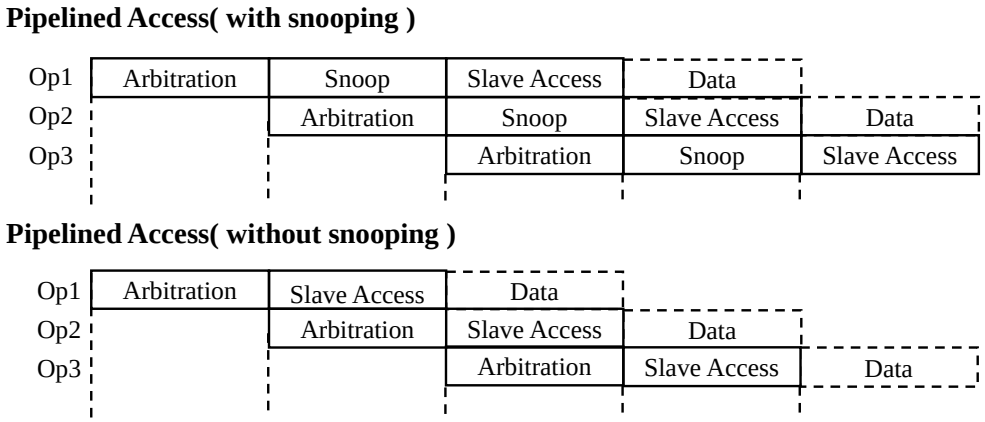


図 2.15: 共有パイプラインバス

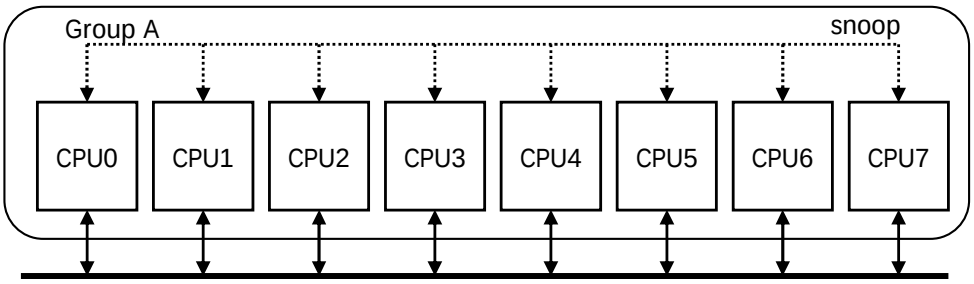
より，複雑さ，コスト，および電力消費が増加するためである．MESI キャッシュコヒーレンシプロトコルをサポートする回路のみを搭載した．128 ビットの内部共有パイプラインバスは，500MHz で動作するように設計されており，バスの帯域幅は 8.0GB/s となる．プロセッサ間同期のために，アトミックメモリ操作を可能にするロック/ロック解除機能もサポートしている．

プロセッサ間割り込み (IPI) は，プロセッサの同期をとるために使われるもので，1 つの CPU が他の任意の CPU に対して割り込みをかけることを許可する．CPU は 4 ビットのプロセッサ間割り込み要因番号を他の CPU に送信する．共有パイプラインバスは L2 キャッシュに接続されていて，L2 キャッシュは DDR2 インタフェースとマルチレイヤ・システム・バスに直接接続されている．その結果，データ転送ネットワークは，アプリケーションがプロセッサの処理能力を利用するのに必要なスピードとバンド幅を提供することができる．

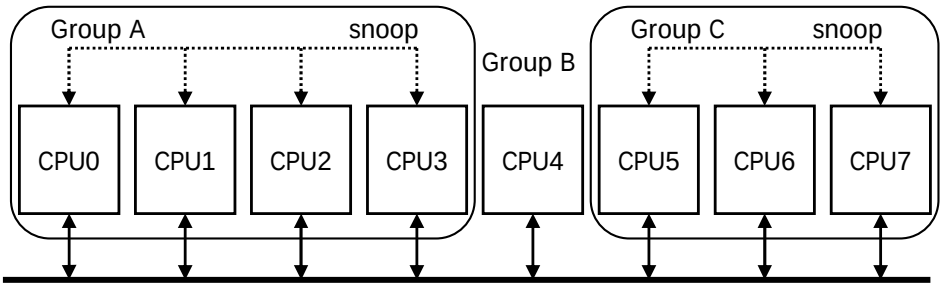
この 8 つの CPU の汎用プロセッサブロックは，コヒーレンスを保つという観点から，8-way の一般的な SMP 構成をとっている．加えて，CPU を異なる複数のグループに分割するという，独自の CPU グルーピング機能をサポートしている．CPU のスヌープ動作は，同じグループにある CPU からの場合に限って行われる．図 2.16 は CPU スヌープのための CPU グループモードを示している．図 2.16(a) は，シングルグループモード，図 2.16(b) はマルチプルグループモードを表している．

シングルグループモード（図 2.16(a)）において，スヌープ動作は，全ての CPU の L1 キャッシュコヒーレンスを保証するように実行される．スヌープ範囲指定機能が有効であるときには，プロセッサはいくつかのプロセッサグループに分割され，その組み合わせは任意である（図 2.16(b)）．各プロセッサグループは個々の特定の処理を実行する．この CPU スヌープグループピングモードは，スヌープオーバーヘッドの削減という利益をもたらし，約 8% の処理性能向上が得られる．

グルーピング機能で最も効果の発揮する場面は，複数の OS が同一ハード内で混在する場合である．一般的な SMP システムにおいては，アプリケーションの動作は，例えば Linux-SMP のような単一の OS が前提となっている．このため，リアルタイム性向上などの対策を行うには OS のサポートなしにアプリケーションのみで性能を向上させなくてはならなかった．そこで，汎用 OS とリアルタイム OS といった



(a) Single group mode for CPU snooping.



(b) Multiple group mode for CPU snooping.

図 2.16: CPU とスヌープのグルーピング機能

複数 OS 間で適切な役割分担を行うことができる，複数 OS プラットフォームを提案した [29] [30].

Orochi の汎用プロセッサブロックでは，CPU をいくつかのグループに分けて編成し，複数の OS がそれぞれのグループで同時に動作する．この OS 拡張性を「マイクロ・クラスタリング・モデル」と名づけることにした．このモデルでは，異なる CPU グループ間のキャッシュコヒーレンスを維持する必要はない．何故なら，各グループで動作する OS はそれぞれ，異なるアドレス空間にアクセスするからである．CPU 間のスヌープ通信量を最小化することは，バスバンド幅をより効果的に利用することになる．各 CPU では，スヌーププロトコルがローカルキャッシュへのアクセスを妨げる要因となることがよく起き，性能劣化の原因となる．スヌープ通信量の最小化は各 CPU をこれらの障害から守り，全体としてキャッシュアクセス性能を改善することになる．

2.5.3 用途に適応したコンフィグレーション変更によるシステム性能の効率化

本節では，Orochi を用いたコンフィギュラブル・ヘテロジニアス・マルチコア・システムを提案する．コンフィギュラブル・ヘテロジニアス・マルチコア・システムは，チップ上のハードウエアリソースを複数のグループに分割する．各々のグループは，あるものは，データ処理を多く行うアプリケーションに，あるものは，演算処理を多く行うアプリケーションに，高度に特化している．図 2.17 に，本コンフィギュラブル・ヘテロジニアス・マルチコア・システムの例を示す．黒矢印はデータの流れを，色付きボックスはハードウエアリソースグループを表している．Orochi を使ったこのシステムは，異なる 4 つのバストランザクションの処理を同時に行う．1 つ目は，スヌープ機構を備えた各 CPU の L1 キャッシュによる，スヌープトランザクションであり，2 つ目は，共有パイプラインバスによるスヌープ・オーバヘッドの隠

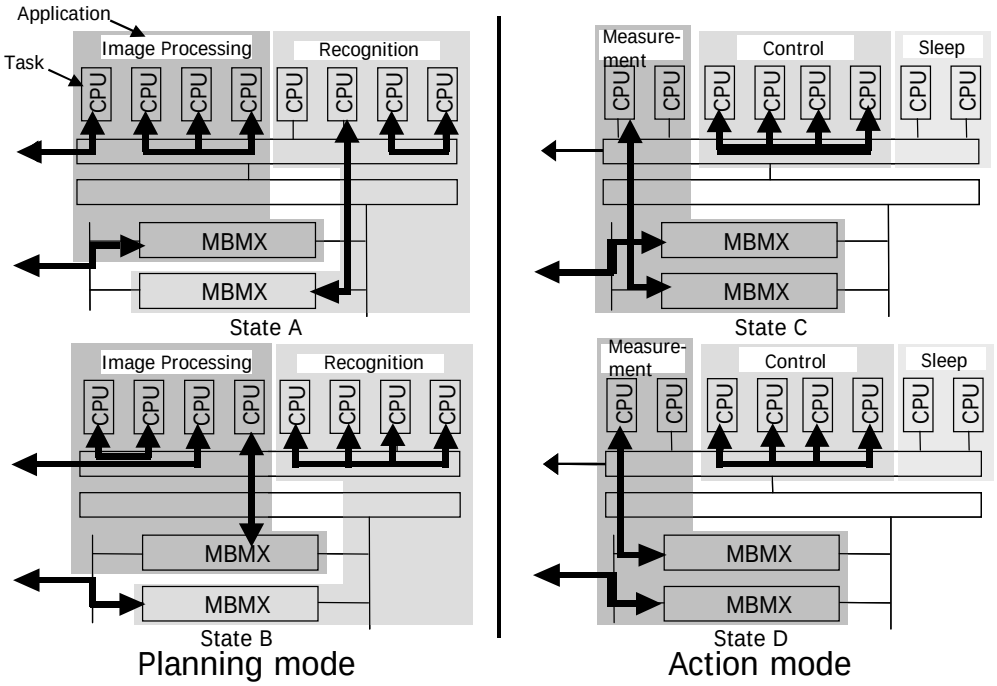


図 2.17: コンフィギュラブルシステムの例

蔽動作である．3 つ目は，図 2.16 にも示したスヌープのグルーピング機能である．4 つ目は，L2 キャッシュによる DDR コントローラとマルチレイヤバスに対する並列アクセスである．さらに，MBMX は，Fly-by バスを経由した外部 SDRAM とのデータ転送と，マルチレイヤバスを介した CPU からのアクセスと，その両方を処理することができる．このように，Orochi のバスシステムはハードウエアの競合による性能の劣化を防いでいる．車載向け高信頼のアプリケーションなどでは，このマルチコアシステム上で，複数のタスクで構成される多重アプリケーションが同時に実行される．各タスクは，画像処理，認識，制御や計測などのアプリケーションによって起動され，ひとつの CPU グループに割り当てられる．

この例として，プランニングモードとアクションモードの 2 種類のモードを使った例を示す．プランニングモードでは，画像処理と認識処理を実行し，4 個の CPU と 1 個の MBMX からなる画像処理グループと，別の 4 個の CPU と 1 個の MBMX からなる認識処理グループで構成されている．状態 A では，画像処理グループが，カメラから取り込んだ画像を Fly-by バスを介して MBMX へ転送し，4 個の CPU で複雑な処理を実行する．画像処理グループによる MBMX へのデータ転送に Fly-by バスが占有されている間，画像認識グループの MBMX が CPU と通信しながら大量データ処理を行う．その後，状態 A は状態 B に遷移し，各アプリケーションにおいて，タスク割り当てを動的に変化させる．画像処理グループの MBMX はデータ処理を開始し，認識グループは CPU が他のタスクを実行している期間を利用して，新しいデータを受け取る．これら二状態を設けることにより，Fly-by バス上のデータ転送競合を回避することができる．以上に述べたように，このハードウエアリソースをグルーピングするメカニズムは，プロセッサ負荷のピークを分散し，ハードウエアリソース競合を低減する．

アクションモードは，3 個の CPU グループで構成される．アクションモードに対応する状態 C と状態 D の図は，プランニングモードと同じ方法での，ダイナミックなタスク割り当て変更を示している．この

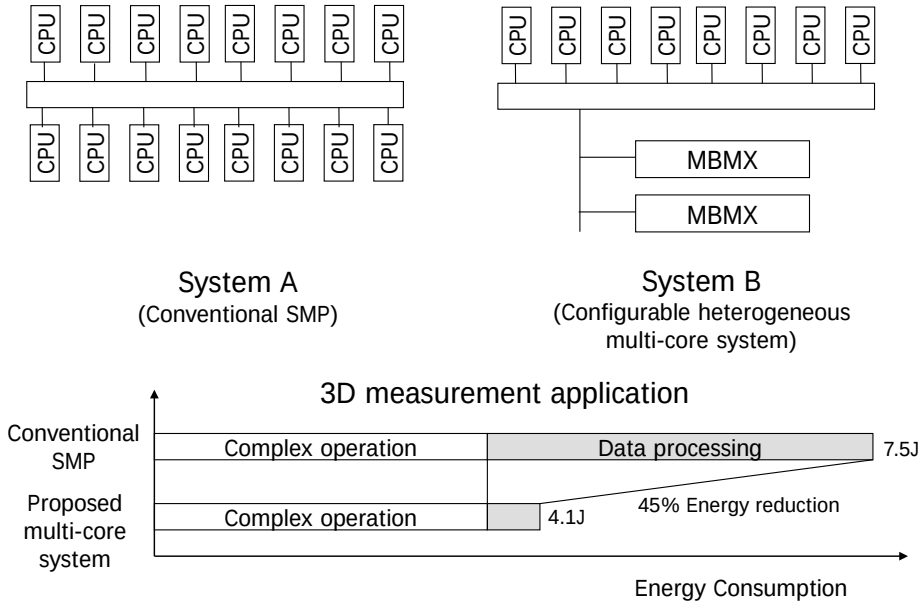


図 2.18: 消費エネルギーの比較

構成では，2 個のハードウエアリソースグループが計測と制御アプリケーションを実行する．1 個のハードウエアリソースグループが低消費電力化のためにスリープ状態に移行する．計測アプリケーション実行には，2 個の CPU と 2 個の MBMX が必要で，制御アプリケーション実行は 4 個の CPU で行われる．

図 2.18 に，上記コンフィギュラブル・ヘテロジニアス・マルチコア・システムによる，消費エネルギーの向上効果の例を示す．この評価では，アプリケーションプログラムとして位相限定相関法を用いて，想定する 2 つのシステムの比較を行った．想定するシステムとしては，従来構成のシステム（システム A）として 16 個の CPU の理想的な SMP 構成をモデル化したものと提案するシステム（システム B）である 8CPU と 2 つの MBMX を用いた構成のシステムである．

位相限定相関法は，画像データに対して 2 次元 FFT 等を行うデータ処理とその結果を基に行う複雑な処理とで構成されている． 1 個の CPU で行った場合，複雑な処理では 25G クロック，データ処理は 28G クロックかかっている．従って，3 次元計測アルゴリズムにかかる総サイクル数は，53G クロックとなる．CPU 理論性能は 1GOPS で，消費電力は 0.14W である．MBMX の理論性能は 38GOPS で，消費電力は 0.75W である．2 システムの消費電力はそれぞれ，システム A の消費エネルギーは，7.5J，システム B の消費エネルギーは，4.1J である．結果として，我々のアプローチであるシステム B は，MBMX が全てのデータ処理を制御することにより，従来 SMP であるシステム A に対して，45% の消費エネルギー削減を行うことができる．MBMX を用いたコンフィギュラブル・ヘテロジニアス・マルチコア・システムは，大規模 2 次元 FFT 等の多数のデータ処理を行うアプリケーションに，よい性能を発揮する．

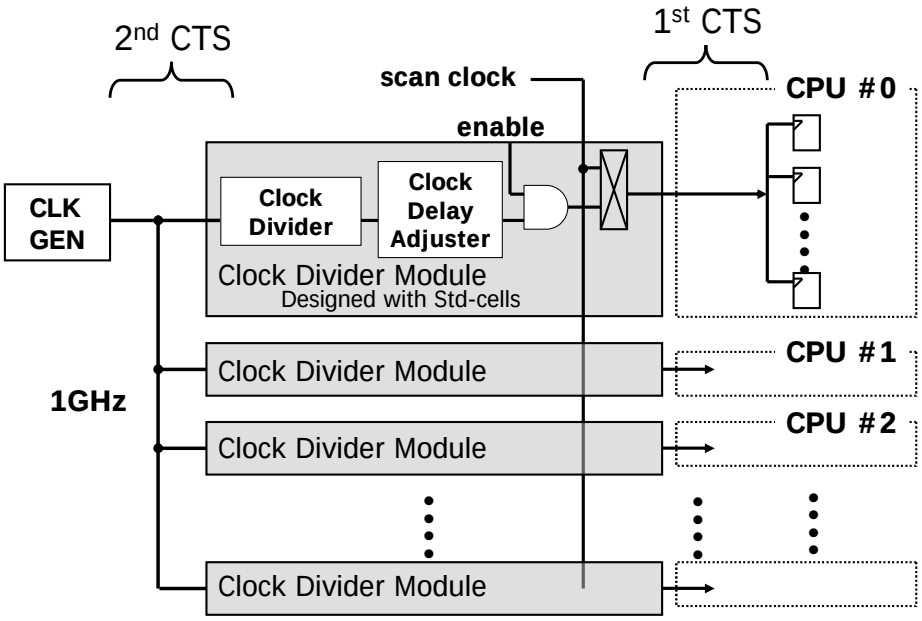


図 2.19: クロックツリーの設計

2.5.4　クロック遅延アジャスターによる低電力設計

クロックツリー設計においては，クロックスキューを適切なやり方で調整する必要がある．Orochi では，クロック遅延調整機能をもったクロック分周器モジュールを，各プロセッサの中心に配置した．図 2.19 はクロックツリー構造を図示している．クロックツリー合成 (CTS) は 2 段階で行われる．第一ステップでは，クロック分周器モジュールから各プロセッサ・モジュール内フリップ・フロップのリーフ全てに至るまでのクロックツリーを合成する．第二ステップでは，クロック生成器から各クロック分周器モジュールまでのクロックツリーを合成する．この手法をとることで，ローカルクロック・スキューが小さいクロックツリーを生成することができる．各 CPU の動作周波数 1GHz を達成するために，ローカルクロック・スキューは 150psec 近辺に最適化されている．

高速動作する SoC のために，クロックスキューを削減することは，一般的に有効とされている [31]．しかしながら，あまりにもクロックスキューが小さいと，電圧降下やピークノイズを増やしかねない．動的なピーク電流を減らすために，各 CPU のクロックパス遅延を調整し，クロックスキューを分散することが必須である．図 2.20 は Orochi に搭載したクロック遅延アジャスター (CDA) を示している．CDA は，遅延要素とセレクタで構成されていて，その分解能は 78psec である．CDA は各プロセッサのクロック位相を微調整することで，動的なピーク電流とノイズを効果的に抑止することができる．

図 2.21 は，3 個のサンプル・チップの CDA 評価結果である．この評価では，7 つの CPU の位相を合わせた上で，1 つの CPU の CDA 設定値を他と違った値に変化させ，最大動作周波数の値を測定した．図 2.21 の X 軸は，CDA 設定値を，Y 軸は評価対象 CPU（スキューありのクロックで動作している）の最大動作周波数を示している．図 2.21 には，2 箇所で，動作周波数が低くなる結果が現れている．1 箇所は，

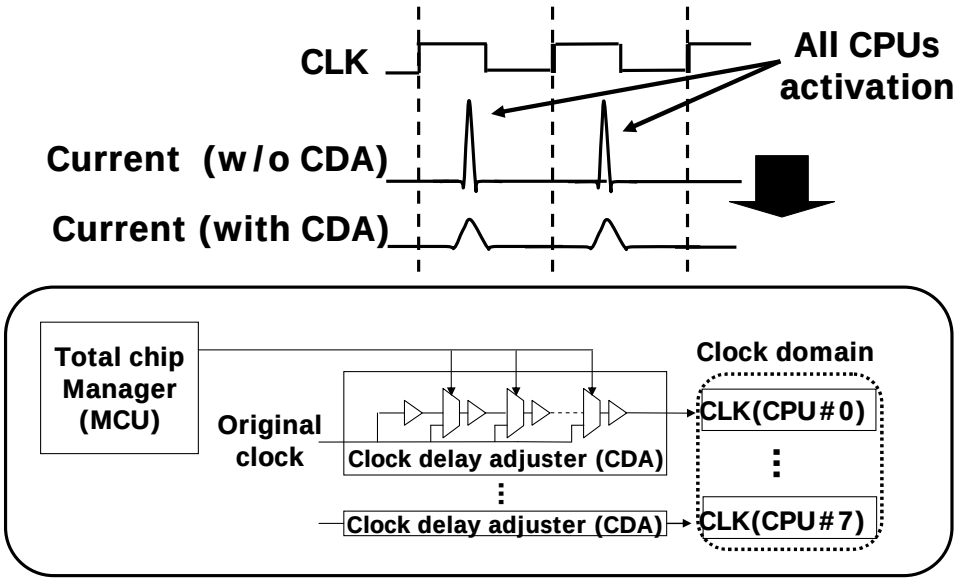


図 2.20: クロック遅延アジャスタ―回路

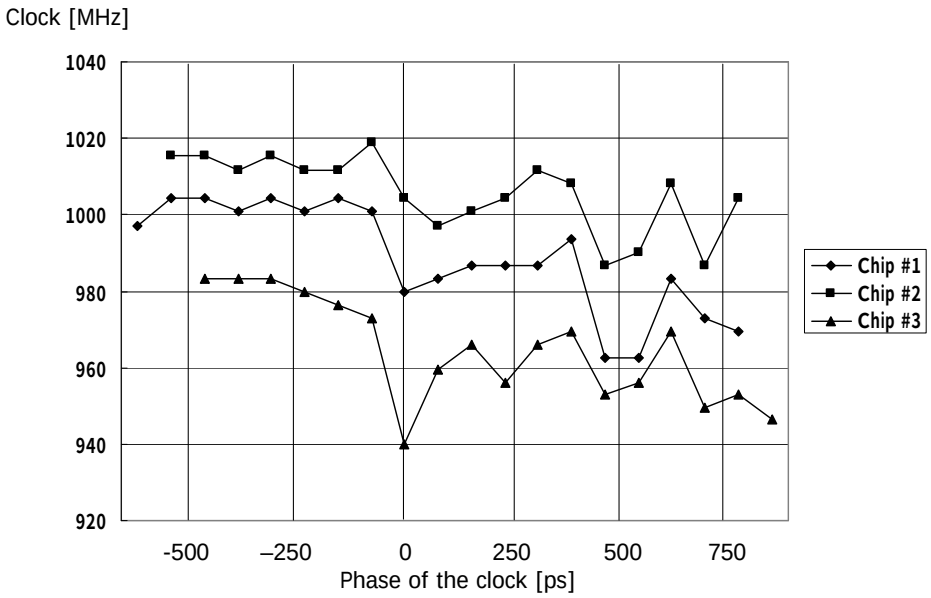


図 2.21: クロックスキュー制御の評価結果

CDA 設定値 0psec 近辺，もう 1 箇所は 500psec 近辺である．

前者では，全ての CPU のクロック位相が揃ってしまったために，評価対象 CPU のピーク電流が，他の 7 個の CPU のピーク電流とオーバラップしてしまったため電圧降下が大きくなり，最大動作周波数が減少する結果となった．また，後者では，規模の大きい電圧降下が起きているが，これは，SoC 内の複数の SRAM がクロックのネガティブ・エッジ近辺で大電流を消費しているためである．このように，これらのピーク電流のオーバラップを分散することで，電圧降下が減少することができれば，対象の CPU の動

作周波数は向上する．実装評価結果では，クロック位相の調整をすることで，約 4%(40MHz) の動作周波数改善が見られた．この CDA を使った調整手法は，動作周波数の向上に有効である．

2.6　関連研究

シングルチップのマルチコアプロセッサは，クロック周波数を上げることなく高性能を実現するための重要なソリューションであり，微細化と電力問題から 2000 年台入って 1 チップのマルチコアが活発に研究開発された [32]．2000 年頃に，スレッドレベルの並列処理（TLP）を備えたチップが組込みシステム用に設計された [33][34][35]．キャッシュコヒーレンシプロトコルを備えた Stanford hydra[36], IBM Power4[37] などのいくつかのシングルチップのマルチコアプロセッサも発表された．スタンフォード大の hydra プロセッサは，無効化のみのコヒーレンスプロトコルであった．IBM Power4 はマルチチップモジュール（MCM）パッケージを採用しており，消費電力が大きいため，組込みシステムでは利用できなかった．これらのマルチプロセッサと比較して，MESI 方式のコヒーレントキャッシュを備え低消費電力と高性能仕様の両方を満たすマルチコアプロセッサである M32R を開発した [26]．その後，ルネサスでは，マルチコアプロセッサラインナップに SH アーキテクチャ [9] を追加した．ARM はマルチコアの取り組について後発であったが NEC Electronics[38] のサポートにより MPcore を開発するに至り，その後の Cortex-A シリーズにつなげた．

2.7　まとめ

ヘテロマルチコア技術により画像や AI 処理向けの SoC の電力を抑えながら処理性能を向上させる研究を行った．この研究では，画像処理や AI 処理向けに認識，推論，計測，制御及びセキュリティなど，高い処理能力と低消費電力が要求される様々なアプリケーションに対応できるマルチコア SoC Orochi を開発した．試作した Orochi は，3 種類の合成可能なプロセッサを実装している．複雑な制御/処理を行うための 8 個のマルチコア CPU(M32R) と大容量のマルチメディア・データを処理する SIMD アクセラレータの MBMX とチップ全体制御のための制御用 CPU(M32C) である．3 つのプロセッサはそれぞれ，CPU は 1GHz，MBMX は 500MHz，制御用 CPU は 500MHz で動作する．試作チップは，90nm CMOS プロセス 8 層銅配線で製造し，729 ピン BGA でパッケージされている．

3 種類のプロセッサはチップ内で，高帯域幅のマルチレイヤシステムバスで相互に接続されている．8 個の CPU はキャッシュコヒーレンス・メカニズム備えた 1 本の共有パイプラインバスで互いに接続されており，さらに，512KB L2 キャッシュを共用することにより，内部バス通信量を削減している．MBMX は，2-read/1-write の演算方式とバックグラウンドの IO 動作を備えることで性能向上を図った．また，CPU コア当たりの電力を下げるため，命令フェッチの電力を効率化する可変レイテンシキャッシュと分割マッチライン方式 TLB を開発し，命令キャッシュの電力と命令 TLB の消費電力を 40 %削減した．

これらの技術により，9 個の CPU と 2 個のマトリックスプロセッサで構成される，ヘテロジニアス・マルチコア・アーキテクチャは，実アプリケーションである位相限定相関法による 3 次元計測の場合，従来の SMP システム (16CPU) と比較して，45 %の消費エネルギーを削減した．Orochi の実装により，フレキシビリティを保持したまま，電力効率（Performance/W）の向上が図れることを実証した．

さらに，あらゆる応用分野やプロセステクノロジにおいても適用できる小型のデジタル遅延モニタを開発した．遅延管理ネットワークを使用しチップ内の温度環境をモニタすることにより，タスクホッピング機能等のソフトウエアによる低消費電力化が可能となった．

第 3 章

低電力・高性能なインターコネクト SoC PEACH の設計

自動運転などの車載組込みシステムには，より高いパフォーマンスと信頼性が重要である．このような組込みシステムを実現するためには，複数のチップでシステムを構成することが有用であり，その場合，高い転送機能を備えた信頼性の高いインターコネクト SoC が必要となる．同時に，組込みシステムとしての低消費電力化も求められる．

本章では，組込みシステムにコンピューティングクラスターを採用する際に，どのように高い信頼性を実現するか，ネットワークトラフィックの高速化と障害処理をどのように管理するか，低消費電力をどのように実現するかの方法について述べる．

複数チップ構成でのシステムにおける通信 SoC に対し，小面積マルチコア CPU と専用割り込みコントローラを組み合わせた，低電力・高性能なインターコネクト SoC を提案する（図 3.1）．通常処理と例外処理の分離により，パケット処理専用コントローラによる通常処理の自動化とマルチコアプロセッサによる複雑，多様なエラー処理の両方が可能となる．また，通常処理時はエラー処理用のブロックを低消費電力モードに移行させることで，低消費電力化が可能となる．

この通信処理のリアルタイム性と複雑かつ多様なエラー処理を両立するため，通常処理・エラー処理を

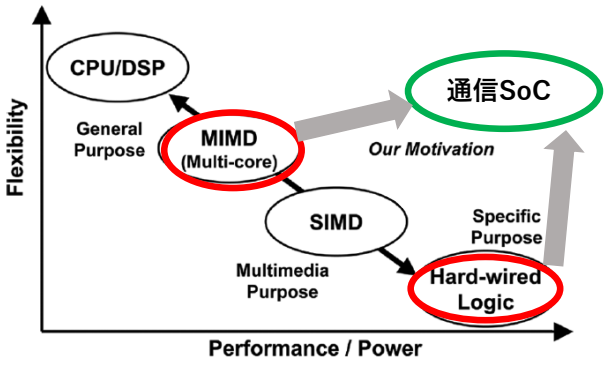


図 3.1: 低電力・高性能なインターコネクト SoC

それぞれ分離し，別々の 2 つのブロック（2 プレーン）で行う方式を提案した．さらに，通常処理の高速化のために，CPU の割込みハンドラを使用せずに通常処理行う，インテリジェント割込みコントローラ（ICU：Interrupt Control Unit）を開発した．

高速ネットワーク通信のインタフェースとしては，PCI Express を採用し，計算ノード間の通信リンクとして用いることで 80Gbps の高い通信性能の PEARL ネットワークを実現した．

3.1　組込み向けクラスター技術:PEARL

過去，コンピューティングクラスターの急速な成長に伴い，一連のネットワークアーキテクチャの開発につながってきた．これらの目標は，多数のコンピューティングノードを低遅延で高帯域幅のネットワークで相互接続することである．これらのコンピューティングクラスターでは，消費電力とシステムコストが増加してきたが，組込みシステムへの適用を考えるとシステム全体を小型化することが重要である．さらに，コンピューティングクラスターの規模が大きくなると障害の確率が高くなるため，信頼性を考慮することも必要である．

3.1.1　PEARL ネットワーク

Infiniband [39] は，ハイパフォーマンスコンピューティングおよびエンタープライズデータセンターで使用されるスイッチファブリック通信リンクである．Infiniband DDR4x の帯域幅は 20Gbps である．Infiniband は，そのネットワークマネージャーであるサブネットマネージャーにより，動的障害回復機能を有するなど高い信頼性を実現する．しかし，フォールトトレランスのためにスイッチの数を増やしていく必要があるが，コントローラチップとスイッチはそれぞれ 1 ポートあたり 3〜5W を消費するため，システム全体の消費電力は大きなものとなるという課題があった．このため，Infiniband は，組込みにおける小規模なコンピューティングクラスターには最適ではない．

汎用のネットワークインタフェース規格であるギガビットイーサネット（GbE）は，コストと電力において Infiniband と同等であるが，転送能力について大きく劣るという課題があるため，これも転送能力が求められるコンピューティングクラスターには適さない．

こうした背景から，高性能，パワーアウェア，信頼性の高いネットワークを実現するために，PEARL（PCI Express Adaptive and Reliable Link）と呼ぶ組込みシステム用の小型コンピューティングクラスター技術を提案した [40]．PEARL では，標準のネットワークデバイスを使用せず，PC 周辺機器間の接続用の高速シリアル I/O インタフェース規格である PCIExpress（PCIe） [41] を周辺機器接続デバイスとしてだけでなく，コンピューティングノード間の通信リンクとして使用することで，高性能と低消費電力の両方を実現できる．

PCIe は，高帯域幅でパケットを転送することができるが，ルートコンプレックス（RC）とエンドポイント（EP）の Point-to-point でのみ接続することができる．コンピューティングノードのすべてのノード CPU が RC であるため，この PCIe の接続規格により，PC 上の PCIe インタフェースを相互に接続できないという問題が生じる．この問題を解決するために，スイッチングデバイスとして機能するインターコネクト SoC である PEACH（PCI Express Adaptive Communication Hub）を開発した [42][43][44]．PCIe

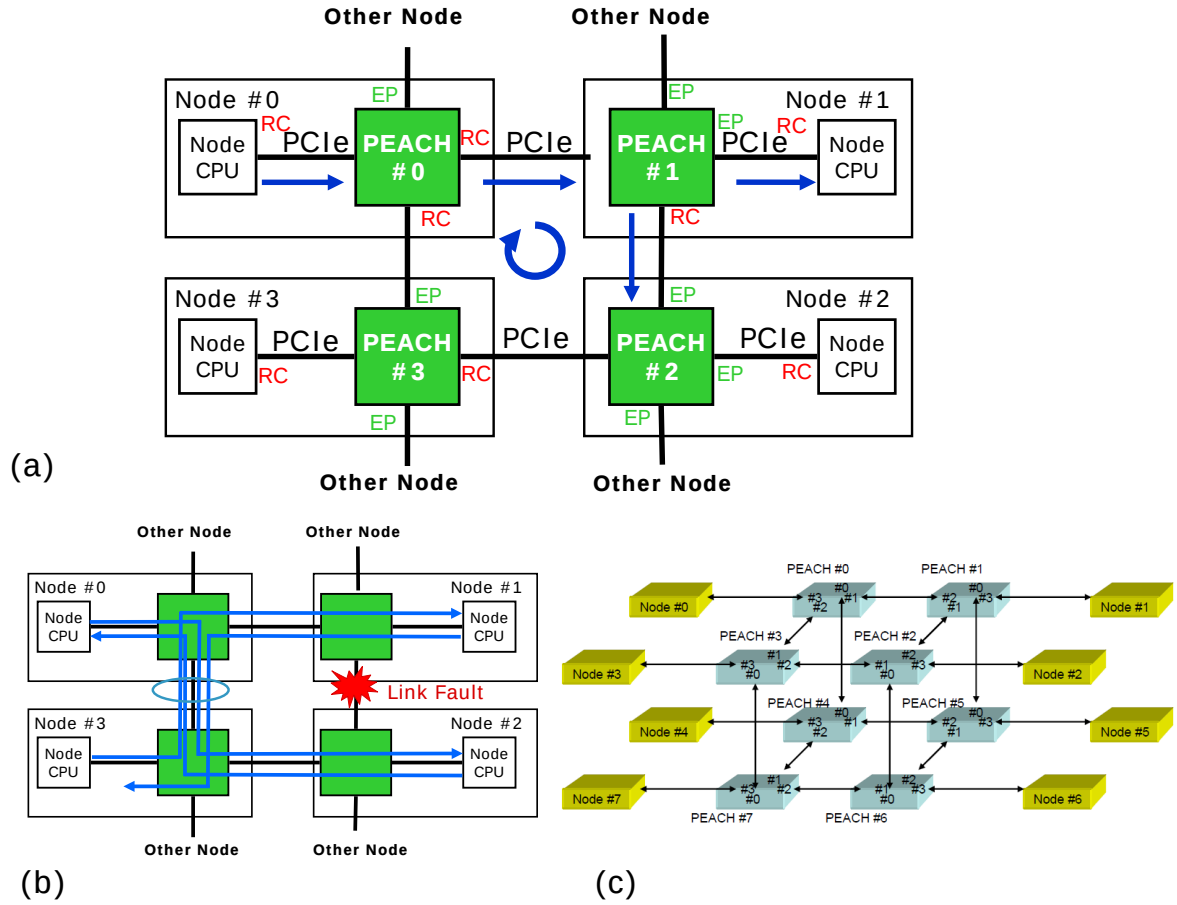


図 3.2: PEARL における隣接ノードとの通信

ケーブルの両端にある RC と EP をペアリングするために，PEACH では RC ポートと EP ポートを切り替えることで，2 つのコンピューティングノードをピアツーピアで接続する．これにより，PEACH は 2 つのコンピューティングノードをピアとしてアドレス指定でき，単一のマスターにのみリンクするという従来の PCIe の制限を回避した．

3.1.2　インターコネクト SoC PEACH の動作

図 3.2 に，4 つの PEACH を使った PEARL システムの例を示す．図 3.2(a) では，PEACH は 4 つの PCIe ポートを介して 3 つのネットワークノードと接続されている．隣接する 1 つのノードはノード CPU で，他のノードは PEACH チップである．PEACH#0 が PCIe ポートを介してノード CPU から要求を受信すると，PEACH#0 はパケットヘッダーを生成し，それを適切な宛先ポートに送信する．PEACH#1 がノードからパケットを受信すると，PEACH#1 はパケットヘッダーを分析し，PEACH#1 はパケットを別のノードに転送するか，ノード CPU に渡す．

ハードウェアは，PCIe 仕様のエラー検出，フロー制御，および再送信制御機能を自動的に処理する．Infiniband はリンクごとのカットオフのみをサポートするが，障害のあるレーンが 1 つあればそれだけで

リンクをダウンさせる必要がある．これとは対照的に PEACH では，自動的に修正できないリンクエラーが発生した場合でも PCIe ポートのレーンの数を減らし，リンクを再初期化して欠陥のあるレーンを削除することができる．この Infiniband よりも強化された機能を活用することにより，ネットワークの信頼性を向上することができる．

迂回ルーティング機能は障害発生時に適用され，障害のあるリンクとノードをバイパスし，ネットワーク機能の回復を可能とする（図 3.2(b)）．PEACH は継続的にシステムを監視し，電力とパフォーマンスの需要を満たすための適応ルーティングと，信頼性の高いネットワークを実現するための迂回ルーティングの両方を動的に実行する．PEARL のノード数は理論的には無制限であるが，設計目標は最大 16 ノードのネットワークとした．図 3.2(c) には，8 ノードのネットワークの例を示す．

3.2　マルチポート，マルチレーンの PCIe インタフェース

3.2.1　PEACH 搭載の PCIe の仕様

表 3.1 にて，PCIe インタフェースの機能を説明する．各 PCIe ポートには，リンクコントローラー，物理層（PHY），ローカル DMA コントローラー（DMAC），およびローカルパケットバッファー RAM がある．PCIe Revision 2.0 標準の転送速度は，1 レーンあたり最大 5.0Gbps で，Revision 1.1 の 2 倍（2.5Gbps）である．Revision 2.0 は，Revision 1.x との互換性があるため，2.5Gbps と 5.0Gbps の両方の転送速度をサポートする．さらに，PCIe の仕様には，2.5Gbps から最大 5.0Gbps への移行手順が規定されている．各ポートには 4 レーンあり，その合計転送速度は 20Gbps であるが，エラー検出の 8 ビット/10 ビットエンコーディングにより，理論上のピーク帯域幅は実際には 2GByte/s となる．4 つの PCIe ポートを備えた PEACH は，最大 80Gbps の高い通信能力を持つ．

表 3.1: PEACH 搭載の PCIe 仕様

	諸元
PCIe Interface	PCI Express standard Revision 2.0 転送速度（1 レーン）：5.0 Gbps, 2.5 Gbps 4 レーン (20 Gbps) x 4 ポート Up-configuration 機能 自動再送機能 ルートコンプレックス/エンドポイント切替機能

3.2.2　PCIe Up-Configuration 機能

通信 SoC の電力全体の 80 %以上が，PCIe PHY の回路によって消費される可能性があるため，PEACH を使ったシステムを構築する上で PCIe の電力をいかに削減するかが重要な課題となる．Infiniband の電力認識制御は，リンクごとの電力遮断のみという制限付きであるが，PCIe には，柔軟な電力コント

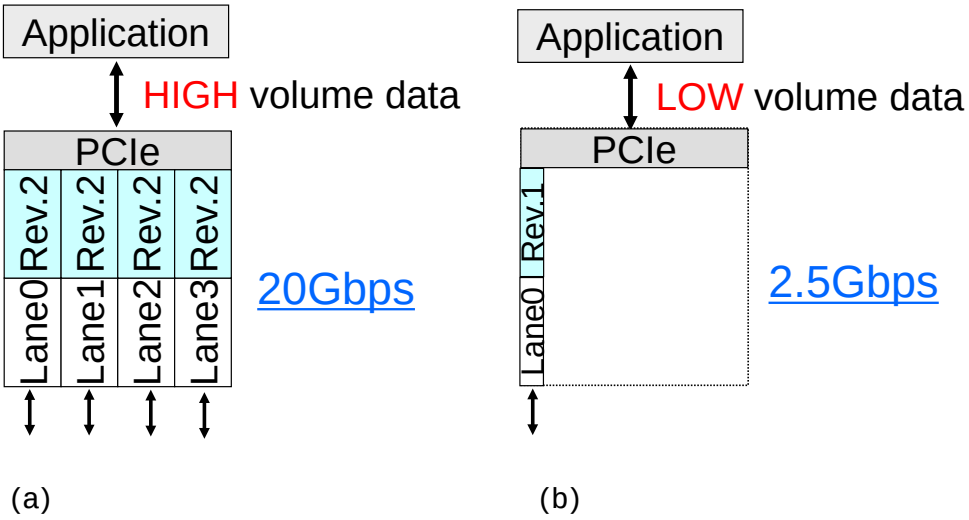


図 3.3: PCIe Up-configuration 機能

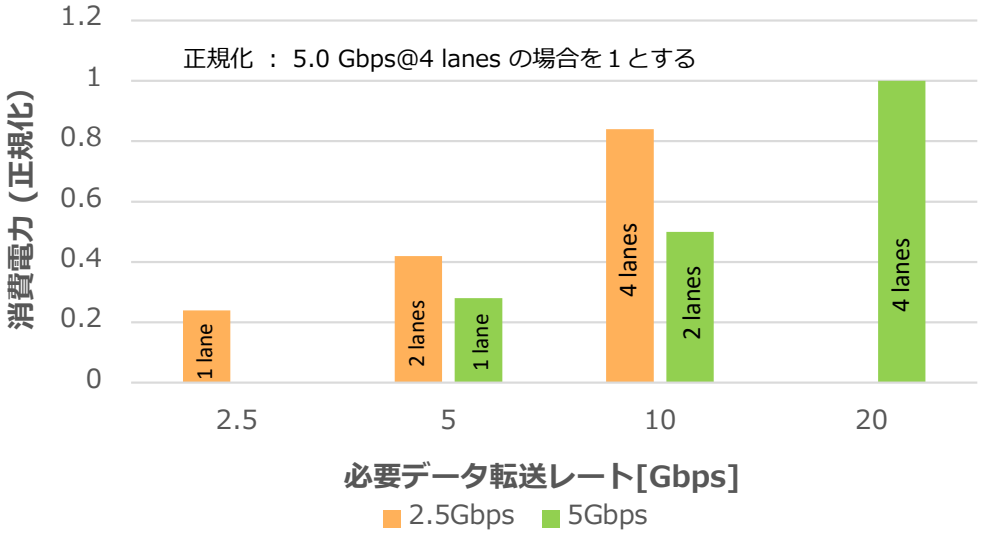


図 3.4: データ転送レートごとの PCIe の消費電力

ロール機能があり，リンクとノード全体でレーン数とレーン速度を動作中にダイナミックに変更できる．PEACH では，使用されるデータ量の変化に応じて転送速度とレーン数を切り替えるために，PCIe の Up-Configuration 機能を使用した（図 3.3）．必要な転送量が多い場合は，PCIe ポートは 20Gbps の最大転送能力で動作するが，転送量が少ない場合，2.5Gbps の 1 つのレーンのみを動作させることで低消費電力化が実現できる．

図 3.4 に，転送量に応じてレーンを切り替えた場合の PCIe 物理層（PHY) の 1 ポートあたりの消費電力の比較を示す．データは，40nmCMOS プロセスで実装された PCIe PHY を用いて消費電力を測定し，

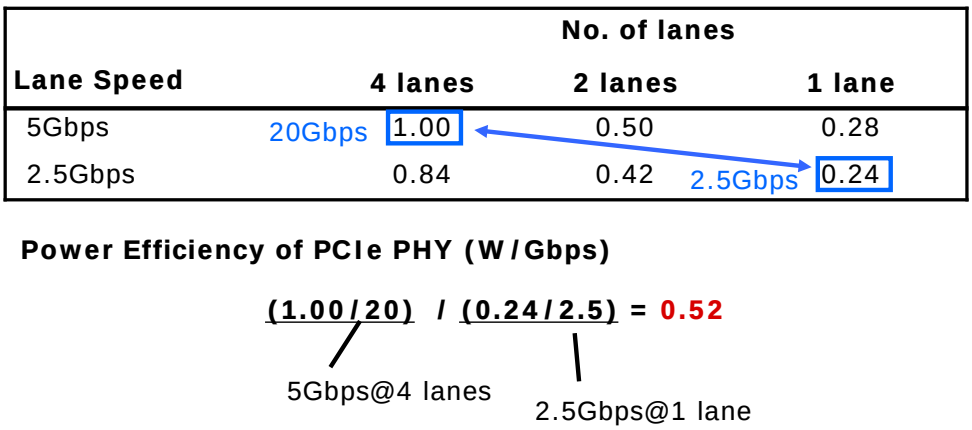


図 3.5: 1G ビット転送あたりの消費電力計算

5.0Gbps の 4 レーンの場合を 1 として正規化した．必要な転送量が 2.5Gbps 未満の場合，2.5Gbps の PCIe ポートを 1 つ使用すると，消費電力が最小になる．必要な転送量が 2.5Gbps より大きい場合は，5.0Gbps の PCIe ポートを使用することで，データ転送レートに応じた最適な消費電力となる．

図 3.5 に図 3.4 の値を基にした 1Gbit データあたりの電力（W/Gbps）の計算例を示す．2.5Gbps の 1 レーンを使用した場合，消費電力は，最大転送速度である 20Gbps の場合の 0.24 倍であるが，データ転送量を加味した電力では，最大転送の場合と比べて，0.52 倍であり，その逆数の電力効率（Gbps/W）は 1.92 倍となる．

3.3　デュアルブロックシステム

3.3.1　デュアルブロックシステムの機能

PEACH では，通信処理のリアルタイム性と複雑かつ多様なエラー処理を両立するため，通常の転送処理とエラー処理をそれぞれ分離し，別々の 2 つのブロック（2 プレーン）で行うデュアルブロックシステムを設計した．

図 3.6 にデュアルブロックシステムを搭載した PEACH のブロック図を示す．このシステムは，主にバスブリッジで接続された制御処理ブロックと転送処理ブロックの 2 つのブロックで構成される．2 つのブロックはバスブリッジで接続されている．

転送処理ブロックには，最大 4 レーンを使用する 4 つの PCIe ポート，通信パケットの一時退避用の 512KB SRAM，および CPU と PCIe I/F と緊密に連携するインテリジェント ICU がある．各 PCIe ポートには，物理層コントローラー（PHY），リンクコントローラー，ローカル DMA コントローラー（DMAC），およびローカルパケットバッファー RAM がある．転送処理ブロックのすべてのメインモジュールは，高速内部システムバスによって接続されている．

インテリジェント ICU は，通常処理の自動転送とエラー時の制御処理ブロックへの通知を行う．また，マルチコアプロセッサからの割り込みサービスをオフロードする高速自動データ転送機能をサポートして

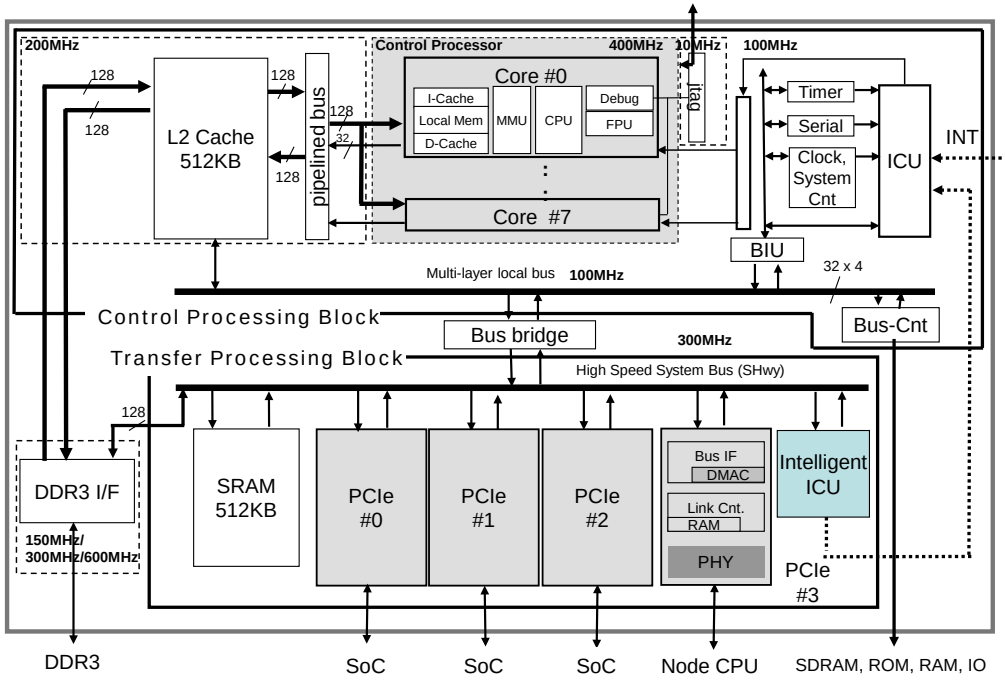


図 3.6: PEACH チップのデュアルブロックシステム

おり，処理時間を大幅に短縮できる．

制御処理ブロックには，128 ビットパイプラインバスで接続されている 8 つの制御用プロセッサがあり [45]，512KB L2 キャッシュを通して DDR3-600 I/F と接続されている．また，L2 キャッシュは，転送処理ブロックと内部マルチレイヤーバスを通して接続されており，DDR3 I/F と内部マルチレイヤーバスの両方に並列にアクセスすることができる．制御処理ブロックでは，適応型ネットワークルーティングとパケットヘッダー分析などのデータ処理とデータフロー制御を行う．

256MB DDR3-600 I/F は，制御処理ブロックと転送処理ブロックの両方から並列にアクセスすることが可能である．256MB の DDR メモリは，制御処理ブロック内の制御用プロセッサの主記憶として使用されるが，通信パケットサイズが内蔵 SRAM の容量である 512KB より大きい場合，大容量のパケットバッファとしても使用される．

制御処理ブロックのマルチコアプロセッサで複数の PCIe ポート処理を効果的に分散することにより，複数の 20Gbps スループットを効率的に活用し，高いトラフィックレートを実現できる．

3.3.2 マルチコアによるネットワーク制御の高速化

マルチコアの利点を最大限に活用するには，それぞれのコアが異なるデータストリームからのパケットを同時に処理できる必要がある．１つのコアを，特定の通信のパケットのみを処理するように割り当てることで，他の通信処理との競合状態による影響を受けることなく，パケットの処理と接続の状態情報の更

新などを行うことができる．これは，パケット処理リソースの分散に関する重要な設計上の考慮事項となる．

特定の PCIe ポートからの割込み信号を指定するコアに割り当てるマッピング機能を PEACH に実装した．Linux での IRQ（割込み要求）アフィニティとこのマッピング機能を使用することで，特定の割込み処理をどの CPU コアで行うかをプログラムから指定することができる．

システムソフトウェアは，割込みサービス中を除いて，制御 CPU コアを着実にアイドル状態にすることができる．前のプロセスからのコンテキストスイッチのオーバーヘッドはなく，割込み応答時間を短縮することができるので，CPU コアは割込み処理をスムーズに開始することができる．

3.4　インテリジェント ICU

インテリジェント割込みコントローラ（ICU：Interrupt Controller Unit）の主な機能は，

- 割込みリレー機能
- チップ間割り込み機能
- 高速自動データ転送機能

の 3 つであり，すべての機能はノード間通信で使用できる．

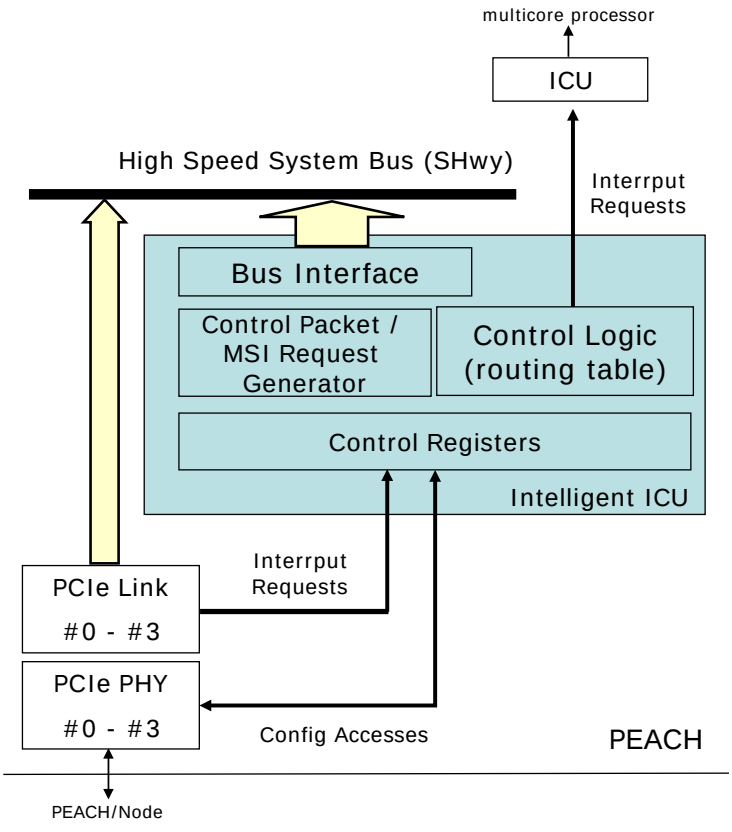


図 3.7: インテリジェント ICU ブロック図

割込みリレー機能は，PCIe リンクアップまたは PCIe の DMA 転送処理の完了通知として生成される PCIe I/F の割込み要求を制御処理ブロック内の ICU（インテリジェント ICU ではない）を介して，転送処理ブロックの PCIe IF モジュールから CPU コアへの割り込み要求を中継する．チップ間割り込み機能は，チップ間のデータ転送の完了通知などの情報を送信する場合に使用する．PCIe を介して PEACH に接続された隣接するチップは，インテリジェント ICU に制御レジスタを書き込んで，コアへの割り込み要求をアサートできる．高速自動データ転送機能は，PEACH の CPU コアを使用せずに転送処理を自動的に処理する機能である．

3.4.1　インテリジェント ICU のデータ転送フロー

図 3.7 に，インテリジェント ICU のブロック図を示す．インテリジェント ICU は，PEACH 内の PCIe ポートとの通信と PEACH 外部の隣接する別の PEACH チップおよびその先の CPU ノードとの通信を行う．どちらの通信にも，高速システムバスと PCIe を介したメッセージパッシングを使用する．インテリジェント ICU は，PCIe を介して隣接ノードにメッセージ信号割り込み（MSI）パケットを送信することもできる．PCIe DMA 転送が完了したこと，または PCIe エラーがあることをインテリジェント ICU に通知する場合，PCIe Link が割り込み要求をインテリジェント ICU に直接送信することができる．

データ送信には，割込みサービスを使用するプロセッサモードとインテリジェント ICU モードの 2 つのフローがある．プロセッサモードでは，エラー処理または起動シーケンスのための柔軟なルーティング

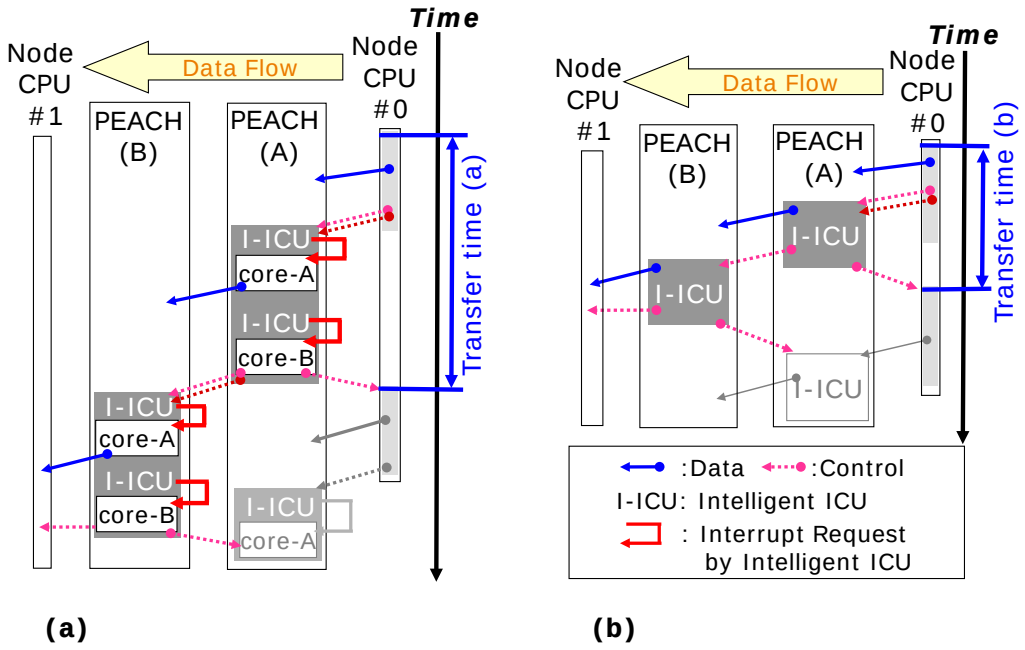


図 3.8: データ転送フロー

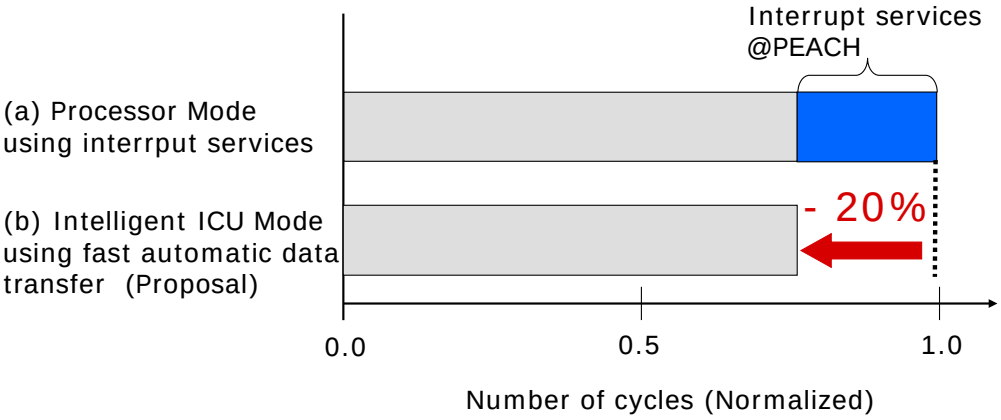


図 3.9: インテリジェント ICU モードの効果

を行うことができる．インテリジェント ICU モードでは，高速自動データ転送機能を用いて高速にデータを転送できるが，ルーティングテーブルは固定されている．

図 3.8 に，割り込みサービスを使用するプロセッサモード（図 3.8(a)）とインテリジェント ICU モード（図 3.8(b)）の 2 つのデータ転送フローを示す．このフローチャートでは，PEACH(A) および PEACH(B) を介したノード CPU#0 からノード CPU#1 へのデータ送信のフローを示している．ノード間のすべての通信パケットは，PCIe を介して送信される．

プロセッサモードでは（図 3.8(a)），通常の CPU コアの割り込み機能を使用してデータ送信を行う．送信フローは以下のとおりである．

Node CPU#0 がデータパケットと制御パケットを送信した後，通信チャネルを確立するために，PEACH(A) のインテリジェント ICU に割り込み要求パケットを送信する．インテリジェント ICU(I-ICU) は，割り込み要求と制御パケットを PEACH(A) の制御処理ブロックにあるマルチコア部に割り込み要求することで転送処理を開始する．割り込みハンドラー（core-A）内で，CPU コアはデータのソース，宛先アドレス，サイズを含むパケットを分析し，パケットヘッダーとパケットを宛先に送信する．core-A は PCIe 内の DMAC を起動し，データを PEACH(B) に転送する．PCIe は，インテリジェント ICU を介して DMA によるデータ転送が完了したことを CPU コアに通知する．core-B 内の割り込みハンドラーでは，core-B は制御パケットと割り込み要求パケットを PEACH(B) に送信し，終了パケットを Node CPU#0 に送信する．PEACH(B) は PEACH(A) と同様に動作し，Node CPU#1 は最終的にデータを受信する．

マルチコアプロセッサで実行されるパケット処理は柔軟だが，割り込み処理のオーバーヘッドは避けられない．インテリジェント ICU はルーティングテーブルを保持しており，ルート計算を自動的に処理する事ができるので，インテリジェント ICU を使うことで，割り込み処理のオーバーヘッドを省き，転送レイテンシを短縮することができる．

3.4.2 インテリジェント ICU によるデータ転送の高速化

図 3.8(b) に，高速自動データ転送機能がマルチコアプロセッサを使用せずに転送処理を行う方法を示す．Node CPU#0 が PEACH(A) にデータ転送要求パケットを送信すると，PEACH(A) のインテリジェント ICU は，PCIe の DMAC を起動し，データを PEACH(B) に転送する．PEACH(B) のインテリジェント ICU も同様に動作し，Node CPU#1 は最終的にデータを受信する．

図 3.9 に高速自動データ転送機能の効果を示す．プロセッサモードとインテリジェント ICU モードのそれぞれについて転送処理時間のサイクル数を示す（プロセッサモードで正規化）．インテリジェント ICU の高速自動データ転送機能により，通常のネットワーク操作での転送処理時間の 20 %を削減できる．

3.5 IRQ アフィニティによるエラー処理の効率化

障害処理とエラー監視/ロギングを，制御処理ブロックのマルチコアプロセッサにより効率的に行うには，適切なロードバランシングとパフォーマンスチューニングが重要であり，割込みサービスが実行される場所の制御が必要である．IRQ アフィニティ制御により，特定の割込み処理を，指定した CPU にバインドすることができる．IRQ アフィニティと CPU スヌープ動作の CPU グルーピング機能は，各 PCIe I/F ポートからの割込みを CPU または CPU グループに 1 対 1 の関係でバインドし，割り込み応答時間を高速化する．インテリジェント ICU とマルチコアの組み合わせにより，転送処理とエラー応答のパフォーマンスを向上することができる．

3.5.1 パケットデータのハンドリング

Linux の IRQ（割り込み要求）アフィニティにより，プログラムは，特定の割り込みを処理するコアを指定できる．PEACH では，IRQ アフィニティは，各 PCIe ポートからの割り込みを 1 対 1 の関係で特定のコアにバインドする．ネットワークパケットは，目的のコアに送信される．この分散処理を使用することにより，PEACH は効率的にパケットを処理することができる．さらに，CPU スヌーピングの CPU グループ機能は，同じグループ内の他の CPU からのスヌーピングのみに制限できるため，スヌーピングのオーバーヘッドを軽減する．不要な内部スヌープトランザクションを排除することで，通信サービスの安定性が向上する．

図 3.10(a) に，PEACH のデータフロー制御を示す．灰色の矢印はデータフローを示す．PCIe#3 がノード CPU からデータを受信すると，SRAM はそのデータを一時的に格納し，その後，データは適切な宛先ポート（PCIe#0）を介して別の PEACH チップに送信される．

点線の矢印は制御フローを示す．PCIe を介して PEACH に接続されているデバイスは，インテリジェント ICU に制御パケットを送信できる．ノード CPU は制御パケットと割り込み要求パケットを送信して通信を確立する．インテリジェント ICU はこの割り込み要求をコアに中継する．割り込みハンドラでは，コアがパケットを分析し，アドレス変換を実行して，DMAC を起動する．

図 3.10(d) に，高速自動データ転送機能による転送処理を示す．インテリジェント ICU は，PCIe でアドレス変換と DMAC の起動を自動的に実行することにより，CPU コアの割り込みサービスを使用せずに

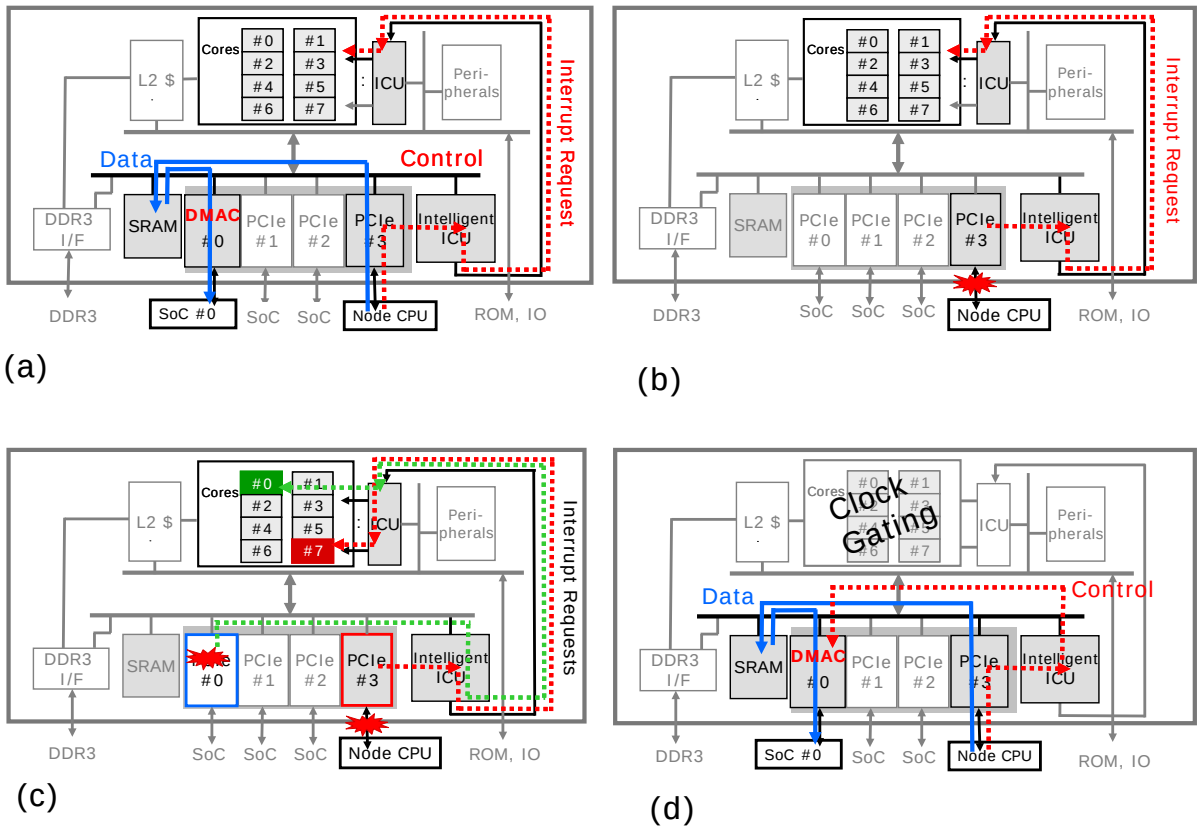


図 3.10: パケットハンドリング

データを転送する．

3.5.2　ソフトウェアによる高信頼性制御

PCIe 仕様のエラー検出，フロー制御，および再送信制御の機能は，ハードウェアによって自動的に処理される．自動的に修正できないリンクエラーが発生した場合，PEACH のネットワークマネージャーは PCIe I/F のレーン数を減らし，リンクを再初期化することで欠陥のあるレーンを削除し通信を再開することができる．対照的に，Infiniband では，リンクごとのカットオフのみしかサポートしていないため，1 つのレーンに障害のあるとリンク全体がダウンする．

図 3.10(b) に，信頼性のための障害処理を示す．リンクまたは隣接ノードの障害が原因で通信が切断されると，PCIe はインテリジェント ICU を介してマルチコア部に割り込み要求が送信される．割り込みを受け付けた CPU コアは，欠陥のあるレーンを削除するか，迂回ルーティングを適用することにより，エラー回復を開始する．PEACH はシステムを継続的に監視し，電力とパフォーマンスの要求に応じた適応ルーティングと迂回ルーティングの両方を動的に実行して，高信頼なネットワークを実現する．

信頼性の観点から，これらの割り込み処理には，エラー応答の迅速さや障害処理の高速化などが不可

欠である．これらの障害処理やエラー監視/ロギングをマルチコア部にて効率的に行うためには，適切なロードバランシングとパフォーマンスチューニングが重要であり，割り込みサービスをどの CPU で実行するかの制御やそのスケジューリングを適切に行う必要がある．IRQ アフィニティは，特定のコアを PCIe ポートに割り当てて，その PCIe ポートによってのみ要求された割り込みサービスタスクを処理させることができる機能である．IRQ アフィニティにより，システムソフトウェアは，割り込みサービス中を除いて，CPU コアを着実にアイドル状態にすることができる．また，実行する CPU コアを固定するため，前のプロセスからのコンテキスト切り替えのオーバーヘッドはなく，CPU コアは割り込み処理にスムーズに移行できる，割り込み応答時間を短縮することができる（図 3.10(c)）．

マルチコアプロセッサでのインテリジェント ICU と IRQ アフィニティの組み合わせにより，割り込み応答時間が改善され，安定性の高い通信サービスが実現できる．

3.6 PEACH の実装と評価

3.6.1 デバイスの実装

実装評価のためにテストチップ PEACH を 45nm CMOS（8 層，トリプル Vth）プロセスで製造し，ダイは，1008 ピンのフリップチップ BGA にパッケージ化した．

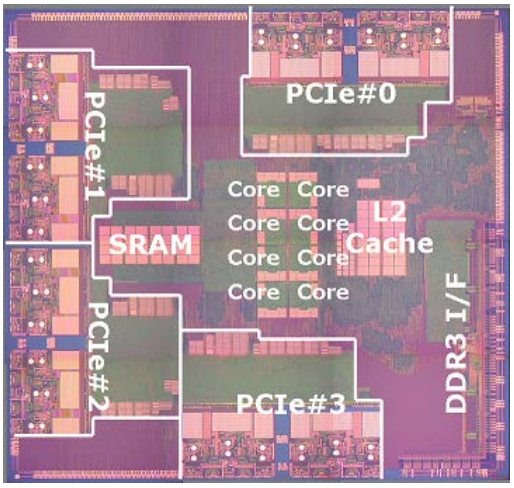


図 3.11: PEACH チップ写真

デュアルブロックシステムを採用した PEACH の主要な機能ブロック図は，図 3.6 に示したとおりである．また，図 3.11 にチップ写真，表 3.2 に PEACH のチップ諸元を示す．

PEACH は，制御処理ブロックと転送処理ブロックの 2 つのブロックから構成され，2 つのブロックはバスブリッジで接続されている．転送処理ブロックには 4 つの PCIe ポートがあり，各ポートは最大 4 レーンを使用してパケットを転送することができる．

制御処理ブロックには，8 つの制御用プロセッサコアがありキャッシュ・コヒーレンス機構を備えたパイプラインバスで接続されている．8 つの CPU を繋ぐパイプラインバスは，リード用，ライト用独立したバスで，そのバス幅は，リード用 128 ビット，ライト用 32 ビットである．バンド幅はそれぞれリード

表 3.2: PEACH チップ諸元

Characteristic	Description
クロック周波数	内部：400MHz max 外部バス：100MHz
プロセッサ	8 コア　マルチコア（SMP）
	L1 キャッシュ: 命令 8KB+ データ 8KB,LRAM 8KB
	MMU,FPU
CPU コア	32bit Processor (400MHz max)
内蔵メモリ	L2 キャッシュ: 512KB
	内蔵 SRAM: 32KB,512KB
内部 BUS	Packet Router
	Multilayer bus (4 layers)
	Piplined bus
DRAM I/F	DDR3-600 x 1, SDRAM x 1
PCIe Interface	PCI Express standard Revision 2.0
	転送速度（1 レーン）：5.0 Gbps, 2.5 Gbps
	4 レーン (20 Gbps) x 4 ポート
	Up-configuration 機能
	自動再送機能
	ルートコンプレックス/エンドポイント切替機能
インテリジェント ICU	転送アドレス，サイズ情報レジスタ x 3
	自動データ転送機能

動作時 8GB/s，ライト動作時は 2GB/s となる．さらに，8 つの CPU で共有する 512KB L2 キャッシュにより内部バストラフィックを削減している．

各 CPU コアは合成可能なプロセッサであり, 浮動小数点ユニット（FPU), メモリ管理ユニット（MMU), 8KB の L1 キャッシュ（I/D），および 8KB のローカルメモリを搭載している．

3.6.2　PEACH の電力評価

図 3.12 に, PEACH の消費電力と電力内訳を示す．コア電圧は 1.2V で CPU クロック周波数は 400MHz で動作し，4 つの PCIeRev.2.0 ポートで 80Gbps の通信時に，消費電力は室温で 3.2W となった．したがって，電力効率は 0.04W/Gbps となり，Infiniband の電力効率 0.083W/Gbps[39] と比較すると，PEACH は，Infiniband よりも 51.5 %高い電力効率である．

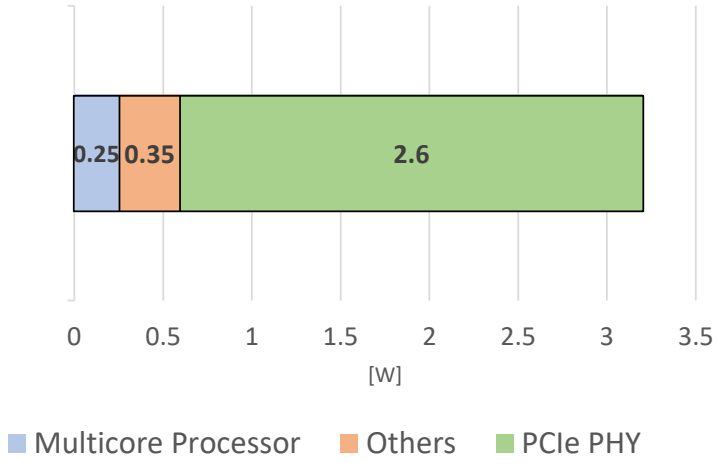
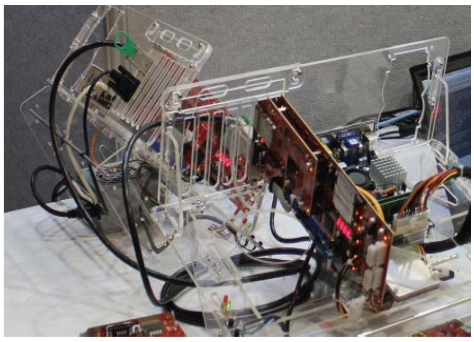


図 3.12: PEACH 消費電力

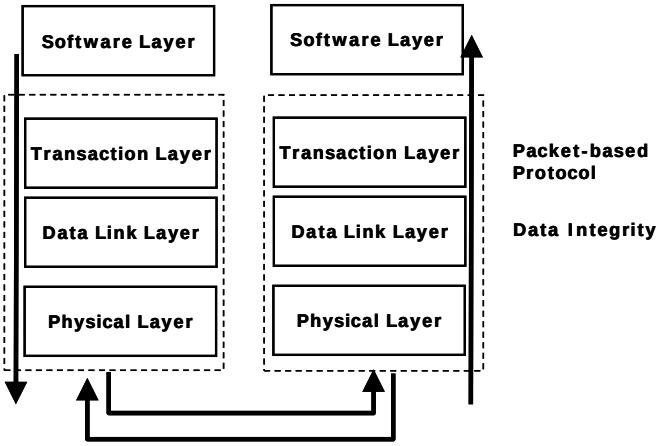
3.7 PEARL システム評価

これまでに説明したコンポーネントの説明に基づいて，PEARL ネットワークシステムの 2 ノードプロトタイプシステムを開発した．図 3.13(c) に，PEACH チップと PCIe 外部ケーブルコネクタを備えた PCIe x4 のホストアダプタボードの写真を示す．ボードには，コンパクトフラッシュカードスロット，4M バイトのフラッシュメモリ，および 2 つの 128M バイト DDR3 メモリが搭載されている．コンパクトフラッシュカードには，Linux カーネル 2.6.35 を含む Linux Ext3 ファイルシステムが格納されており，ログ情報の保存にも使用する．Linux は PEACH のマルチコアプロセッサ上で単独で動作し，コンパクトフラッシュカードに格納された Linux カーネルイメージをロードして起動する．このホストアダプタボードは，コンピューティングノードのマザーボードの PCIe スロットに挿入する（図 3.13(a)）．Linux の CPU ホットプラグにより，システム負荷に応じてコアを動的に一時停止および再開することができ，動作するコア数を負荷に応じてコントロールすることで，システム全体の消費電力をコントロールすることができる．

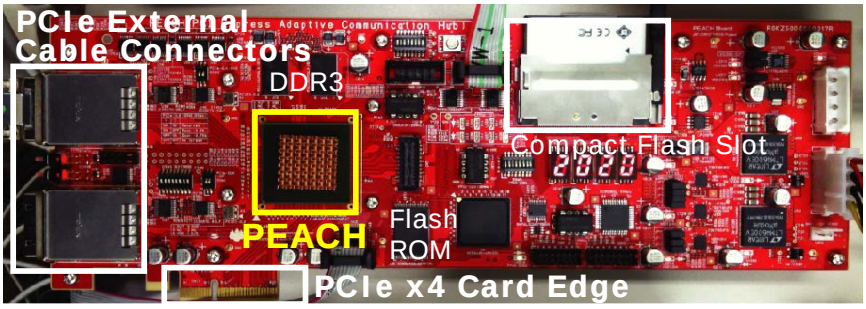
PCIe を使ったネットワークアーキテクチャは，4 つの個別の論理層で構成されてる（図 3.13(b)）．下から上に向かって，物理層，データリンク層，トランザクション層，およびソフトウェア層があり，ソフトウェア層は，トランザクション層によって転送される読み取り要求と書き込み要求を生成する．トランザクション層は，メモリへの読み出しまたはメモリからの書き込み，メッセージの受け渡し，動作環境の設定など，通信のトランザクションを管理する．データリンク層は，パケットのシーケンスやデータの整合性などのリンク管理を処理する．これには，エラーの検出と訂正が含まれる．物理層は，インピーダンス整合および入力バッファを備えたドライバ，パラレルからシリアルおよびシリアルからパラレルへの変換，PLL などすべてハードウェア回路で構成されている．各 PCIe ポートには，ハードウェアモジュールとして PHY，データリンク層コントローラー（MAC），およびトランザクション層コントローラーがある．



(a)



(b)



(c)

図 3.13: PEARL プロトタイプ

3.7.1　PEARL ネットワーク例

図 3.14 に，PEARL ネットワークシステムの 6 ノードのプロトタイプ例を示す．PEARL では，コンピューティングノードの接続を PCIe 外部ケーブルで行う．PEACH を搭載したネットワークカードは，コンピューティングノードのマザーボード上の PCIe スロットに挿入する．PCIe による最小のレイテンシは 1.2μs であり，Infiniband の約 2μs と比較して低遅延である．ネットワークカードの CF カードには，Linux カーネル 2.6.35 を含む Linux Ext3 ファイルシステムが格納されており，CF カードから Linux カーネルイメージをネットワークカードのメモリにロードされ，PEACH のマルチコア上で Linux が起動する．ログ情報の保存には，CF カードが使用される．

3.7.2　PEARL ネットワークマネージャ（PNM）

基本的なネットワークルーティングスキームでは，任意の 2 つのコンピューティングノード間で待ち時間の短いルートを探索するが，より高度なルーティングスキームを実現するために PEARL ネットワーク

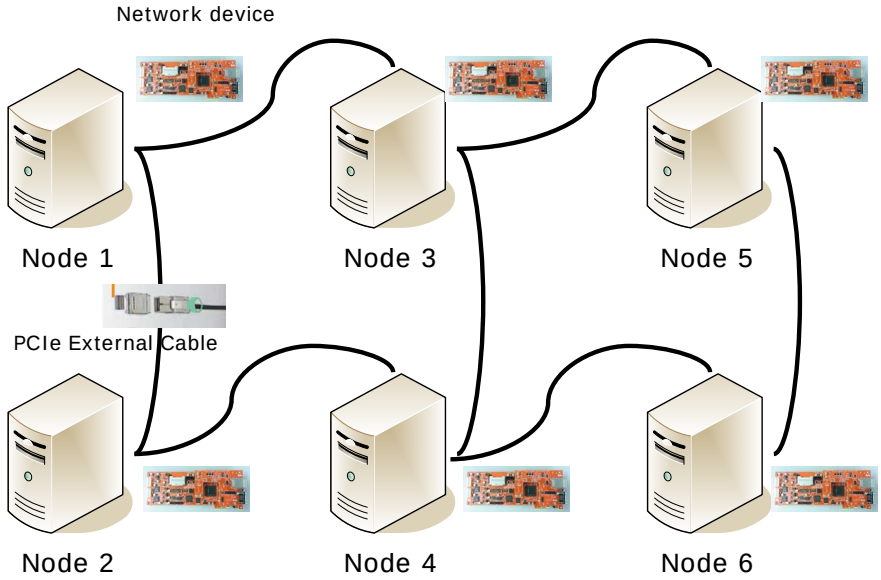


図 3.14: PEARL 6 ノード環境

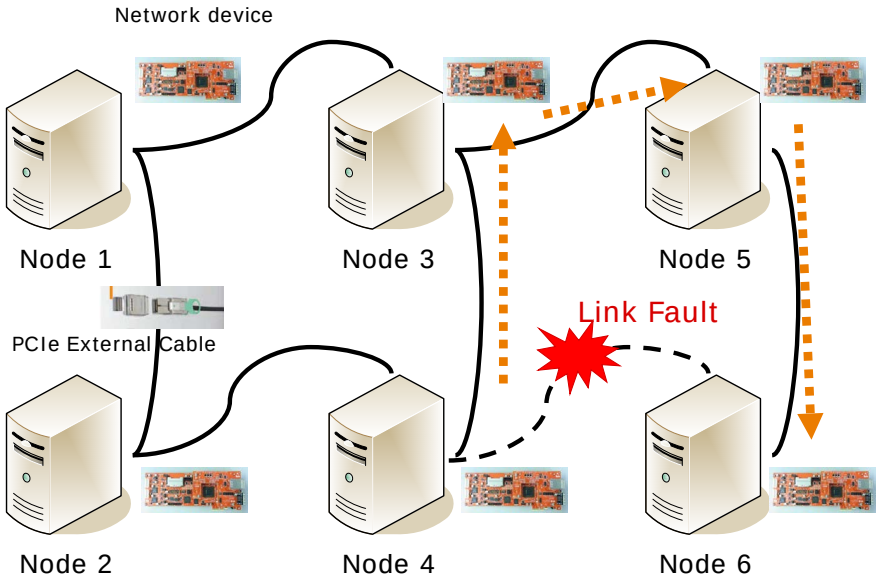


図 3.15: 障害発生時のバイパス処理

マネージャー（PNM）を開発した．PNM は，PEACH 上の各 Linux でユーザーランドプログラムとして実行され，障害状態での自律的な迂回ルーティングや電力とスループットの要求に応じた適応ルーティングを行う．

PEARL ネットワークシステムでは，各 PEACH チップのドライバにルーティングテーブルがある．起動時にマスターノードはルート検索を実行し，インテリジェント ICU モードを想定してルーティングテーブルを作成する．PNM は，デーモンプログラムとしてネットワークステータスを監視し，選択され

たマスターノードに情報を送信する．PCIe は，プロセッサへの割り込み要求によって，Linux 上の sysfs インタフェースを使用してリンクステータスの変更をデーモンに通知する．障害が発生すると，PNM は各 PEACH のルーティングテーブルを上書きしてルートを変更し，障害のあるリンクとノードをバイパスする（図 3.15）

3.7.3　ネットワークシステムの電力マネージメント

アプリケーションの要求に応じて，マスターノードは PCIe に対して動的に電力制御を行う．トラフィックをモニタリングすることで，オンザフライでルーティングおよび再ルーティングを行いトラフィックを最適化できる．PEACH の各ネットワークマネージャーはトラフィックの状況に応じて PCIe のリンク構成を変更するが，この動的電力制御において重要なポイントは，PCIe のスイッチング時間のオーバーヘッドを小さくすることである．

表 3.3: PCIe up-configuration 機能のスイッチング時間

レーン数	スイッチング時間		
To/from	1 レーン	2 レーン	4 レーン
1 レーン	-	4.6μs	4.6μs
2 レーン	9.1μs	-	4.6μs
4 レーン	9.1μs	9.0μs	-

表 3.3 に，PCIe のコンフィギュレーション変更時に必要なスイッチング時間の測定結果を示す．レーン数を増やすのに必要な時間はほぼ 9.1μs だが，減らすのに必要な時間は 4.6μs である．一方，レーン速度を上げるのに必要な時間は 6.5μs であり，速度を下げるのに必要な時間は 3.8μs である．両者に差があるのは，PCIe PHY がレーン速度を上げるには追加の時間が必要なためである．この評価システムで測定された PEACH チップ間の DMA 転送の最小レイテンシは 1.0μs であった．

このように，PEACH のマルチコアプロセッサは，デーモンプログラムを使用してネットワークステータスを監視し，up-configuration 機能とクロックゲーティングなどのプロセッサ電源管理を用いて PCIe の物理層のパフォーマンスを管理する．ここで重要な点は，PEACH ではマルチコアプロセッサを採用することで，PCIe のコンフィギュレーションをきめ細かく操作することで電力を考慮した制御を行うことができることである．

3.8　関連研究

高信頼性が要求される組み込みシステムには，低電力技術と高性能ネットワーク技術が不可欠である．Infiniband は，高性能コンピューティングクラスタの低遅延および高帯域幅ネットワークとして頻繁に使用されるが，電力消費が大きいため，組み込みシステムへの適応には問題がある．GbE は低コストネットワークのもう 1 つの候補であるが，パフォーマンスの点で，要求を満たすことはできない．

RI2N（安価なネットワークとの冗長相互接続）は，GbE [46] のマルチリンクに基づくフォールトトレラ

ントで高性能な相互接続ネットワークであり，GbE 用のコントローラーチップの消費電力は，Infiniband の消費電力よりもはるかに小さくなっている．RI2N は，組込みネットワークに適応するためにスループットの向上が必要であるが，これにより消費電力が大きくなる．さらに，GbE は本来長距離用のネットワーク IF であるため，GbE を使用した通信レイテンシは比較的大きくなる．そのため，従来のネットワーク IF は，低電力および高性能ネットワークには適していなかった．

3.9 まとめ

DEOS（実用的なディペンダブルエンベデッドオペレーティングシステム）プロジェクトによる研究から生まれた PEARL の高速転送機能の通信リンク用キーデバイスとして 80Gbps PCIe I/F を搭載した PEACH を開発した．組込み向けマルチチップ構成の実現のための高信頼インターコネクト SoC に対し，PCIe を通信 IF としデュアルブロックシステムによる通常処理とエラー処理の分離による効率化の提案を行った．

通常処理では，インテリジェント ICU のデータ転送開始機能により，CPU 割り込みハンドラを使用する場合と比較して，転送処理時間を 20 %削減した．エラー処理においては，アフィニティ機能を備えたマルチコアプロセッサが，ネットワークパフォーマンスの向上と障害の検出，回復による信頼性の向上に寄与することを示した．マルチコア技術に専用コントローラを導入した構成による PEACH の実装を行い，PCIeRev2.0 を 4 ポート備えた PEARL ネットワークにおいて，Infiniband より 51.5 %高い電力効率である 0.04W/Gbps を達成した．また，PEARL 評価システムでその処理能力，低消費電力，および高い信頼性を実証した．

PEACH のマルチコアプロセッサは，特定の PCIe ポートから専用コアにネットワークパケット処理を割り当てて，特定のタスクと特定のコアをバインドすることができる．この機能は，高速パケット処理のための効果的な手法を提供する．マルチコアプロセッサで複数の PCIe ポート処理を効果的に分散することにより，複数の 20Gbps スループットを効率的に活用し，高いトラフィックレートを実現できた．PEACH と PEARL で示したように，PCIe をノード間通信に拡張適用することにより，さまざまな通信の新しい可能性を切り開くことができる．

これにより，組込み向けのコストを抑えた高性能で高信頼のインタコネクトが可能となる．

第 4 章

MCU 向け低電力プロセッサ RX の設計

SoC とは異なり，IoT 向け MCU では，特に厳しいコスト（面積）制約があるため，マルチコアや SIMD などの大規模アクセラレータによる性能向上策を使うことができない．そのため，単一 CPU 性能そのものが組込みデバイスの性能を決定する．そこで，IoT 向け MCU 実現のため，最適化された命令セットとメモリアクセス数を削減する新しいプリフェッチ回路により，高いコード密度，低消費電力，および高い処理性能を実現するアーキテクチャである RX プロセッサを設計開発した．

本章では，低コストで高い処理性能と低消費電力を両立するための次の 3 つの要素技術，1) 可変長命令セットアーキテクチャによる低コストと低消費電力化の実現，2) 命令フェッチの効率化による低消費電力化手法（小型キャッシュ），3) FPU/DSP 機能の低コスト実装方式について述べる．

4.1　小型 MCU 向けプロセッサアーキテクチャ

組込みシステムのインテリジェント化は，ビルディングオートメーション，医療，モータ制御，e-メータなどの MCU 応用分野の裾野にまで広がってきている．このような，高度インテリジェントシステムに対応するため，ソフトウエアのスケールや複雑度は増す一方である．また，MCU への大容量メモリ搭載や高機能化への要求は，プロセッサの動作周波数向上，高機能化を必要とする一方で，多くの MCU では，コスト，消費電力，エリアの制約が非常に厳しいことに変わりはない．

RX プロセッサは，大容量 FLASH メモリを搭載した MCU 向けプロセッサアーキテクチャである．FLASH メモリを混載するプロセステクノロジは，その時代の最先端ロジックプロセスから何世代も古いものを使わざるを得ない．また，電力効率を高めるために，動作周波数を高くすることだけで処理性能を稼ぐことはできない．以上のような理由から FLASH メモリを内蔵する MCU は，動作周波数では 30MHz から 200MHz が現在の主流である．これらの動作周波数レンジ，プロセステクノロジを使った場合，どこが律速条件となるかを明確にし最適化を行う必要がある．

最大動作周波数の向上のためにはパイプラインの段数を多くする必要があるが，性能に影響を与える分岐ペナルティの増加を抑えるための分岐予測機構を強化する必要がある．一方，パイプライン段数が少ない場合，そのようなサポート回路が不要となるので小面積でサイクル性能を上げることができる．動作周波数を上げて処理性能の向上を図れない状況では，パイプライン段数は少ない方が有利である．この様な事情から，MCU 向け低消費電力プロセッサとしては，パイプライン段数としては，3 段から 7 段あたり

が適切となる．

以上のとおり，MCU に搭載されるプロセッサは，動作周波数レンジ毎に，最適な命令セットが検討されるべきである．それは，動作周波数とコードサイズの間にトレードオフが存在するためである．RX プロセッサの開発では，レジスタ構成や命令数は RISC 相当としながらも CISC 的な可変長命令セットや高機能命令を動作周波数のレンジを考慮しながら取り込むことで，高いコード効率と電力効率の実現を図った．また，CPU 内の演算器数およびレジスタ数に対して，コスト制約を考慮して命令フォーマットの選択やパイプライン構成の設計も行った．

4.2　可変長命令セットアーキテクチャ

4.2.1　低消費電力化に寄与する静的コードサイズ効率

組込み分野ではメモリ容量削減はコスト観点からも不可避である．静的コードサイズの削減は MCU に搭載されるメモリ容量を削減する．RISC アーキテクチャでは，デコードの負荷軽減のために固定長命令の採用が一般的である．しかも 4 バイトの固定命令長を採用するアーキテクチャが多い．これは，「動作周波数を上げることで性能を稼ぐ」が RISC アーキテクチャの設計思想だからである．しかし，静的コードサイズが大きくなる問題があるため，組込みデバイスとしては，固定長命令は採用されなかった．いくつかの RISC アーキテクチャでは，このコード効率の短所を補うために 16 ビットと 32 ビットの命令を組み合わせた命令セットアーキテクチャ（ISA) を採用してきた．MIPS，ARM などは，当初は 4 バイトの固定長命令であったが，16 ビットと 32 ビットの可変長の ISA を後に仕様追加した [47]．しかし，16 ビット命令は 32 ビット命令のサブセットとして定義されることが多く，このコード効率に関するペナルティ軽減策の効果は限定的であった．

また，RISC アーキテクチャの多くは，動作周波数の向上を目的にデコードや演算を簡略化するため，アドレッシングモードを大幅に簡略化，削減している．この弊害として，例えば，メモリアクセス時のアドレス計算にいくつかの演算が必要となる．このような単純化しすぎた ISA は，コード効率の観点ではコストの高い ISA と言える．

コード効率の問題を解決するために，厳選した 113 命令を持つ ISA を RX 命令セットアーキテクチャ（RX アーキテクチャ）として定義した [3]．命令セットの概要を表 4.1 に示す．その命令数は，RISC ベースのアーキテクチャのもつ命令数と同じレベルである．

高性能，高コード効率，低消費電力を実現するために，RX では 1〜8 バイトの可変長命令方式を採用した．RX アーキテクチャの命令長はバイト単位で可変であり，その命令長は使用するデータサイズとアドレッシングモードに依存する．特に大きなアドレス値または大きなデータ値を指定する必要がある場合，RX アーキテクチャは最大 8 バイトの長さで 1 バイト単位でその命令長を調整できるため，コード効率の点で有利なアーキテクチャとなっている．これにより，命令コード効率が向上し，1 命令あたりのメモリからフェッチされるデータ量が抑えることができる．

CISC アーキテクチャは元来，1 命令あたり実行する処理量の面で有利であり，それはコード効率の向上につながる．RX アーキテクチャでは，文字列処理のようにロード命令等を繰り返すループ処理を行うなどの高機能命令を定義することで，静的コードサイズだけでなく動的コードサイズを削減し低消費電力

表 4.1: RX 命令セットの概要 [3]

Operator Type	RX Instructions
算術・論理演算	ABS, ADC, ADD, AND, CMP, DIV, DIVU, EMUL, EMULU, SUB, MAX, MIN, MUL, NEG, NOP, NOT, OR, RMPA, ROLC, TST, RORC, ROTL, ROTR, SAT, SATR, SBB, SHAR, SHLL, SHLR, XOR
データ転送	MOV, MOVCO, MOVLI, MOVU, POP, PUSH, POPC, POPM, PUSHC, PUSHM, REVL, REVW, SCCnd, STNZ, STZ, XCHG (RXv3) BFMOV, BFMOVZ, RSTR, SAVE
ビット操作	BCLR, BMcnd, BNOT, BSET, BTST
システム操作	BRK, CLRPSW, INT, MVTIPL, MVFC, MVTC, RTE, RTFI, SETPSW, WAIT
分岐	Bcnd, BRA, BSR, JMP, JSR, RTS, RTSD
ストリング操作	SCMPU, SMOVB, SMOVF, SMOVU, SSTR, SUNTIL, SWHILE
DSP 機能	EMACA, EMSBA, EMULA, MACHI, MACLH, MACLO, MSBHI, MSBLH, MSBLO, MULHI, MULLH. MULLO, MVFACGU, MVFACHI, MVFACLO, MVFACMI, MVTACGU, MVTACHI, MVTACLO, RACL, RACW, RDACL, RDACW
単精度 浮動小数点	FADD, FCMP, FDIV, FMUL, FSUB, FSQRT, FTOI, FTOU, ITOF, ROUND, UTOF
倍精度 浮動小数点	DMOV, DPOPM, DPUSHM, MVFDC, MVFDR, MVTDC, DABS, DADD, DCMPcm, DDIV, DMUL, DNEG, DROUND, DSQRT, DSUB, DTOF, DTOI, DTOU, FTOD, ITOD, UTOD

化を図った．ストリング操作命令（SCMPU:ストリング比較），ビット操作命令 (BMcnd:条件ビットセット)，FIR フィルタ用ループ命令 (RMPA:積和演算)，飽和演算 (SATR:32 ビット符号付き飽和処理) などがあげられる．DSP 機能命令，FPU 命令については後述する．

4.2.2 組込みアプリケーションにおける命令実行頻度の分析

命令の出現頻度に関する研究結果は，Hennessy & Patterson[1] や Wilkinson[48] にも掲載されているが，RX アーキテクチャ決定に際しては，32 個の組込みシステムの実アプリケーションを用いて命令頻度解析を行い，最も頻繁に使用される命令とアドレッシングモードの調査，分析を行った（表 4.2）．実アプリケーションは，オフィスオートメーション/PC 周辺機器，民生機器，家電，通信機器，産業機器，自動車の各分野から抽出した．

表 4.2: 命令実行頻度の分析

Rank	RX Instructions	Integer Average % total static codesize
1	Move	31%
2	Conditional Branch	15%
3	Compare	11%
4	Subroutine Branch	8%
5	Add	6%
6	Others	29%
Total	-	100%

調査の結果，MOV 命令は，すべての操作の 30 %以上を占める最も頻繁に使用される命令であることがわかった．次に使用頻度が高かったのが，条件付き分岐命令である．続いて，比較命令，サブルーチン分岐命令，ADD 命令の順になる．

4.2.3　命令コード割り当て

4.2.2 に示した分析により，効果的なアドレッシングモードと，高頻度の命令・オペランドにより短い命令コードを割り当てた．

命令フォーマット内で，アドレス指定に利用するレジスタ相対フォーマットでは，ディスプレースメント値を 8 ビット (dsp:8[R])，16 ビット (dsp:16[R]) をサポートし，更に，高頻度の短いディスプレースメン値に対して，ショートフォーマット (dsp:5[R]) を備え，コードサイズを削減した．また，組込みアプリケーションでは即値は様々なデータ設定や MCU の周辺機能制御のために重視される．MOV 命令は，レジスタだけでなく，メモリへの即値設定ができる．また 4 ビット (imm:4) から 32bit(imm:32) に至る様々なデータサイズを符号付き，符号なしの両方を用意することで，命令セットの効率化を図った

また，使用頻度の高い演算命令のコード効率を最適化するための 3 オペランドの命令フォーマットを定義した．

汎用レジスタの本数は，レジスタフィールドのビット数となり，オペランド数との積で命令コード長に影響を与える．一般的に，8 本，16 本，32 本の中の選択となるが，上記の組込み実アプリケーションの分析から，汎用レジスタは 16 本とし，レジスタフィールドを 4 ビットに抑えた．

頻出命令の短い命令長への割り当て方針を例を挙げて説明する．1 バイト命令にはショートブランチ命令（前方分岐距離が 10 バイトより短い分岐命令）を割り当て，if-then-else のプログラムコードの命令コードサイズを小さくする．2 バイト命令には，転送命令を割当てる．3 バイト命令には，よく使われる演算命令を割当てる．代表的な 1〜3 バイト命令を表 4.3, 表 4.4, 表 4.5 に示す．RX アーキテクチャのコード効率向上における最大の貢献者は 3 バイト命令である．他の RISC アーキテクチャは 2 バイト命令の次は 4 バイト命令を選択せざる負えないからである．特に，基本的な演算命令の 3 オペランドフォー

マットを 3 バイト命令に割り当てることが効果的である．

表 4.3: RX 1 バイト命令

命令	ニーモニック	ディスプレースメント
PC 相対条件分岐	BEQ.S，BNE.S	pcdisp:3
PC 相対絶対分岐	BRA.S	pcdisp:3

表 4.4: RX 2 バイト命令

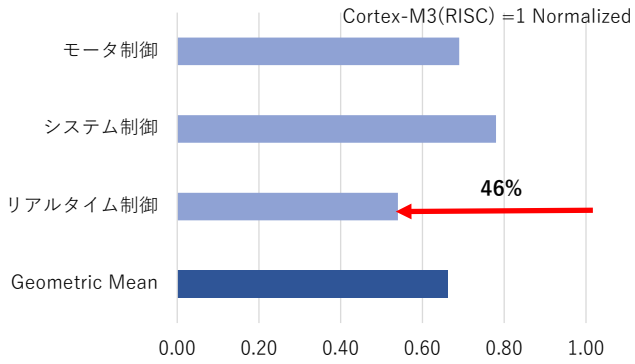
命令	ニーモニック
データ転送	MOV (レジスタ-レジスタ，メモリ-メモリ，ロード，ストア)
比較	CMP (レジスタ-レジスタ，レジスタ-即値)
加算	ADD (レジスタ + レジスタ，レジスタ + 即値)
サブルーチン分岐	BSR
乗算	MUL (レジスタ*レジスタ)

表 4.5: RX 3 バイト命令

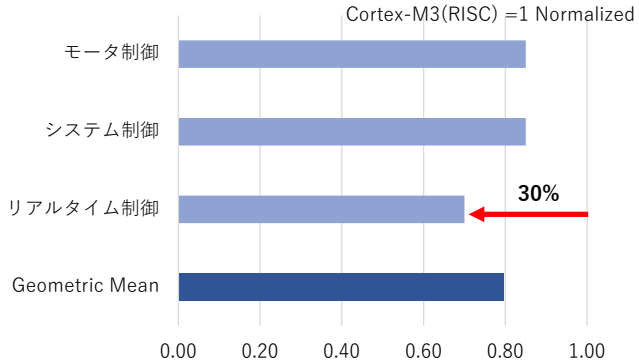
命令	ニーモニック
除算	DIV (レジスタ/レジスタ)
積和 (32 ビット)	EMAC (レジスタ*レジスタ)
浮動小数点加算	FADD (レジスタ + レジスタ)
浮動小数点乗算	FMUL (レジスタ*レジスタ)

4.2.4 コード効率の評価

図 4.1 は，RX プロセッサと RISC アーキテクチャ (ARM Thumb2) とのコードサイズ分析を 3 つの異なるタイプのアプリケーションで示す．比較は，コンパイルされたオブジェクトの静的なコードサイズとアプリケーション動作時に実際に実行される命令の動的なコードサイズの 2 つの面で行った．結果は，RX プロセッサの命令セットは，RISC アーキテクチャ (ARM Thumb2) に対して，静的コードサイズが最大 46 %（幾何平均 34%），動的コードサイズが最大 30 %（幾何平均 20%）削減されている．静的コードサイズが小さいことは，命令を格納する ROM のサイズが約半分で済むことを示しており，このことは面積コストの削減に大きく貢献する．また，動的コードサイズが小さいことは，同じアプリケーションを実行するのに命令フェッチアクセスが 30 %少ないことを示しており，結果，命令フェッチの電力が削減されるので，プロセッサ実行時の低消費電力化に貢献する．



(a)



(b)

図 4.1: 命令コードサイズの分析 (a)static codesize(relative) (b)dynamic codesize(relative)

4.2.5　マイクロコードを用いた高機能命令

CISC アーキテクチャでは高機能命令の実現のため，マイクロコードを用いたマイクロプログラミングが使用されていた．過去の CISC アーキテクチャの実装では，RTL による設計手法も浸透していなかったため，マイクロ ROM とマイクロシーケンサを用いてマイクロコード方式で実現していた．このため，面積コスト，実行速度の面で，単純化された RISC と比べて劣っていた．

現在，RTL による機能記述が一般的となり，EDA による論理最適化が進んだことにより，この状況は変わってきている．少数の高機能命令であれば，簡単なマイクロコードを実行する単純なシーケンサーで実現可能であり，それらを RTL で記述し，すべて論理ゲートで実装することで，面積コスト，実行速度に与える影響を最小に抑えることができる．RX プロセッサでは，シーケンスを持つ高機能命令をストリング操作命令と FIR フィルタ用ループ命令に絞って定義することで，小面積を維持しながら，コード効率の向上を図った．

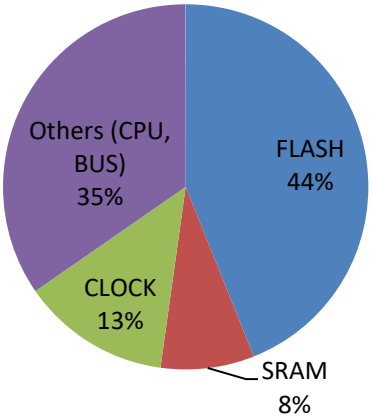


図 4.2: MCU 消費電力

4.3　命令フェッチの効率の改善

4.3.1　命令フェッチの効率化による低消費電力化

MCU の消費電力の大部分は，内蔵 FLASH メモリから CPU への経路での電力消費である（図 4.2）．したがって，内蔵 FLASH メモリへのアクセス数を減らすことは，消費電力を削減する上で重要である．FLASH メモリへのアクセス数削減の施策として，可変長 ISA により動的コードサイズを小さくし，ある処理における命令フェッチ数を削減することが有効である．次のアプローチとしては，CPU と FLASH メモリの間に命令キャッシュを追加することで，内蔵 FLASH メモリへ直接アクセスすることを低減することも有効である．

4.3.2　小型キャッシュによる命令フェッチ効率の向上

SoC などのプロセッサは，メモリ動作速度より大幅に高い動作周波数で動作するため，CPU からメモリへのアクセスにはウエイトが入るようになる．これを緩和するための一般的な方法としてキャッシュを搭載してきた．組込み MCU に搭載される命令用の内蔵 FLASH メモリのパフォーマンスは，MCU の動作速度が比較的低速（200MHz 以下）であり高速な FLASH メモリをウエイトなしでアクセスできるため，100 %キャッシュヒット時のパフォーマンスと同等となる．

RX プロセッサでは，プロセッサとメモリの速度ギャップを緩和するだけではなく，メモリアクセス回数を削減し低消費電力化を図るために，小規模分岐ターゲットキャッシュを備えた Advanced Fetch Unit(AFU) を開発した（図 4.3）．

AFU 構造を決定するにあたり，いくつかのパフォーマンスとコスト間のトレードオフについて検討を行った．プロセッサは，120MHz ノーウエイトでアクセスできる，40nm MONOS FLASH テクノロジー [49] を採用しており，内蔵 FLASH メモリから CPU デコーダに直接フェッチするレイテンシは 1 サイクルである．したがって，RX プロセッサは命令プリフェッチ時のアクセスウエイトによるパフォーマン

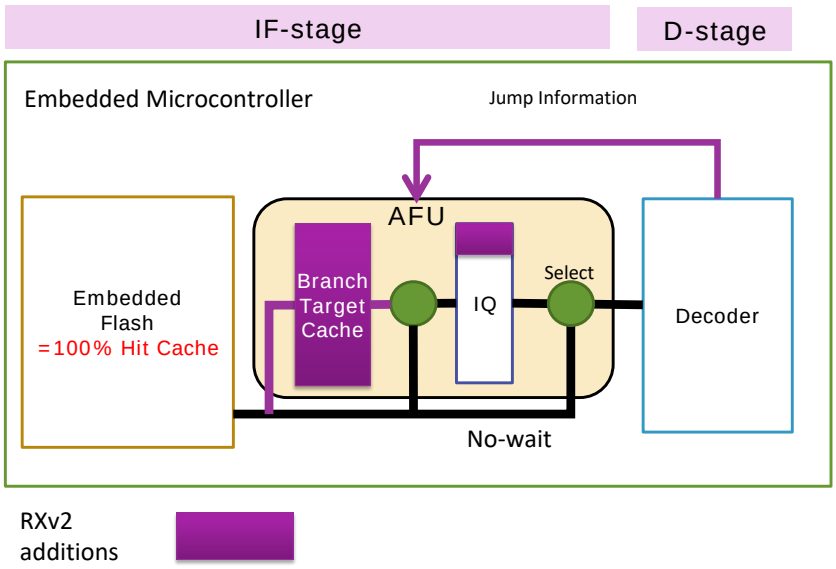


図 4.3: AFU の概要

ス低下は起こらないので，パフォーマンス向上のために，分岐ペナルティの改善に専念している．RX プロセッサは，一般的なキャッシュよりも面積が比較的小さくコストメリットのあるブランチターゲットキャッシュを採用した．AFU は，CPU デコーダと FLASH メモリの間に配置され，命令キュー（IQ）と，LRU 置換アルゴリズムを備えた小さなフルアソシアティブの分岐ターゲットキャッシュで構成されている．AFU は次の機能を備えている．

- 分岐先コードを保存する（分岐ターゲットキャッシュ）
- 無駄なプリフェッチを抑止する（小さなループでの命令キューの再利用，JUMP 命令が検出されたときのプリフェッチの停止など）
- 動的な優先度制御に基づきキャッシュラインを置き換える（8 エントリ LRU，適応ロックなど）

4.3.3　小容量キャッシュ実装方法の選択

AFU では，分岐の高速化だけでなくメモリアクセスを減らすためにキャッシュを導入するが，その目的のためには，キャッシュ電力が FLASH メモリへのアクセス電力より低消費電力でなければならない．一般的にキャッシュ機構を搭載する場合には，面積効率の良い SRAM が選択されるが，AFU では電力オーバヘッドの少ない FlipFlop（FF）でキャッシュを構成した．FF でキャッシュを構成するにあたり考慮する点は，どこまでのサイズであれば SRAM のキャッシュに対して面積コスト的にオーバーヘッドにならないかである．FF は，消費電流についてはメリットがあるが，面積コストが大きいためエントリ数等のサイズ制約があるからである．

図 4.4 に，40nm FLASH 混載プロセスにおける SRAM と FF の面積コストについての比較をスケールの違う 2 つのグラフで示す．セルベースの FF に比べて SRAM のメモリセルは小さいが，センスアンプ

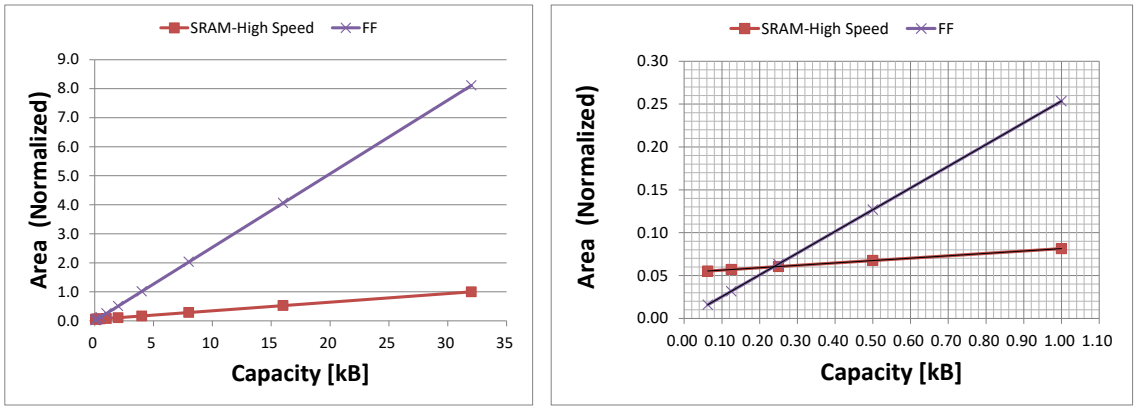


図 4.4: キャッシュの面積比較

等周辺回路が SRAM には必要なので，容量が 200 バイト近辺までは FF の方が SRAM に比べて小さく，面積コストに関して有利である．それ以上の容量になると，SRAM の方がメモリセルの小ささが全体の面積に効いてくるので面積コストに関して有利となる．電力に関しては，FF は，SRAM のセンスアンプ等のオーバヘッドが無いため，読み出し電流はほぼ無視できるレベルである．このように数百バイトまでの小容量であれば，キャッシュを実装する上で面積コスト，消費電力共に FF を使った方がメリットがある．また，レイテンシに関しても SRAM と比較すると FF の方が 1 サイクル早くデータを返せるため，分岐時のアクセスの高速化にも効果的である．このような理由から 100 バイトあたりの小容量のキャッシュを FF で実装した．

4.3.4　低消費電力化のためのコード再利用機能

内蔵 FLASH メモリがノーウエイトでアクセスできるためキャッシュミスによるペナルティが発生しない．このため，AFU とノーウエイトの FLASH メモリの組み合わせにより，消費電力を削減しかつパフォーマンスを向上させることができる．この AFU は，プログラムフローの分析に基づいて，命令のバッファリングをその場で決定する．

(1) ショートループ

AFU は，命令キューに存在する過去にフェッチした命令コードの再利用を行う．分岐ターゲットキャッシュにあるループ検出機構により短いループコードを検出すると，AFU はフェッチされた命令を IQ および分岐ターゲットキャッシュで再利用する．IQ は，分岐によるパイプラインのフラッシュ動作からループ内の命令コードを保護するためにロックする．

この短いループのバッファリングは，両方の分岐ペナルティを削減し，ループコード全体を格納するループキャッシュなどの一般的なアプローチよりも，低コストで FLASH メモリアクセスを削減する．この結果，プロセッサの命令フェッチの消費電力を削減することができ，また，プロセッサの性能を下げる

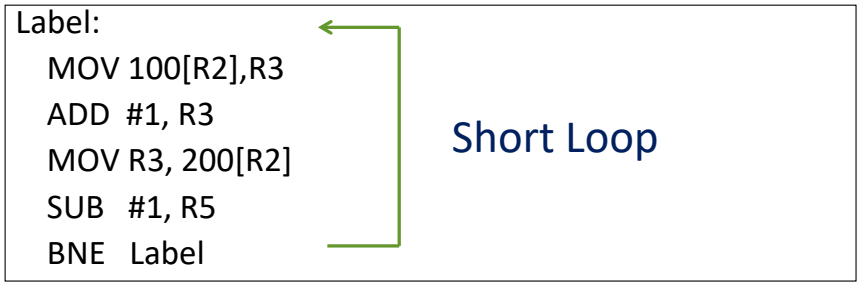


図 4.5: ショートループ

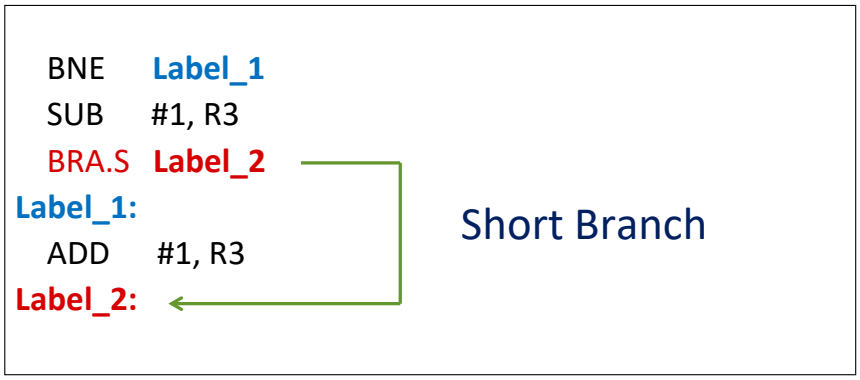


図 4.6: ショートブランチ

ことがないため，性能あたりの消費電力の効率を高めることができる．

(2) 前方ショートブランチ

メモリからフェッチされたデータを効率的に利用するもう 1 つの方法は，短い前方分岐高速で行うことである．CPU コアは分岐先までの距離情報を AFU に伝える．AFU が，IQ 内に分岐先コードを見つけると，CPU はパイプラインフラッシュとメモリアクセスを行わずに IQ から次の命令コードをフェッチする．この手法により，iF-then-else 制御フローのサイクル性能と電力消費を改善する．

4.3.5　AFU の性能評価

図 4.7 に，AFU による小さなループプログラムの電力効率の向上結果を示す．評価では，実際の負荷を使用したゲートレベルの電力シミュレーションで消費電力とパフォーマンスを測定した．図における RXv1 は AFU を搭載しなかった世代であり，RXv2 は AFU を搭載しており，両者の差が AFU の効果を示す．

AFU を搭載した RXv2 は，搭載しない RXv1 と比較して 5.7 倍の電力効率向上を達成し，AFU の有効性を確認した．

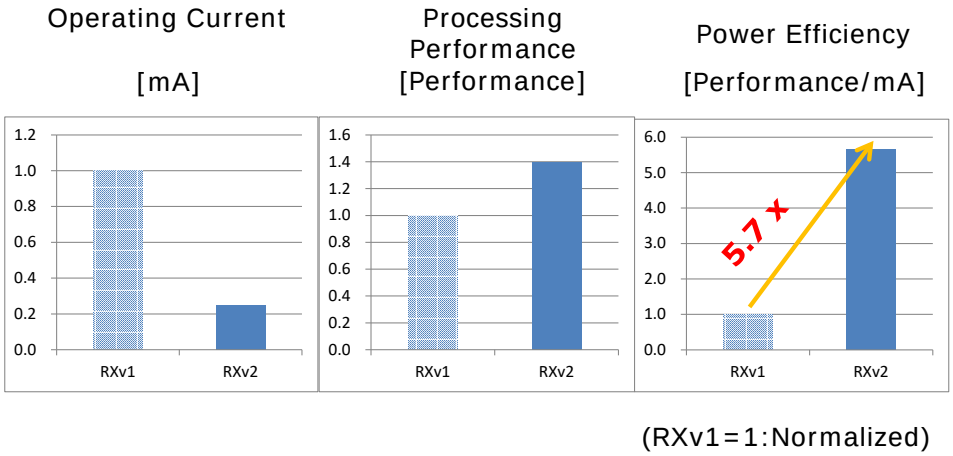


図 4.7: 小さなループプログラムにおける電力性能の向上

4.4　MCU 向け省電力・小型化を可能とするパイプライン構成

4.4.1　パイプライン統合 FPU

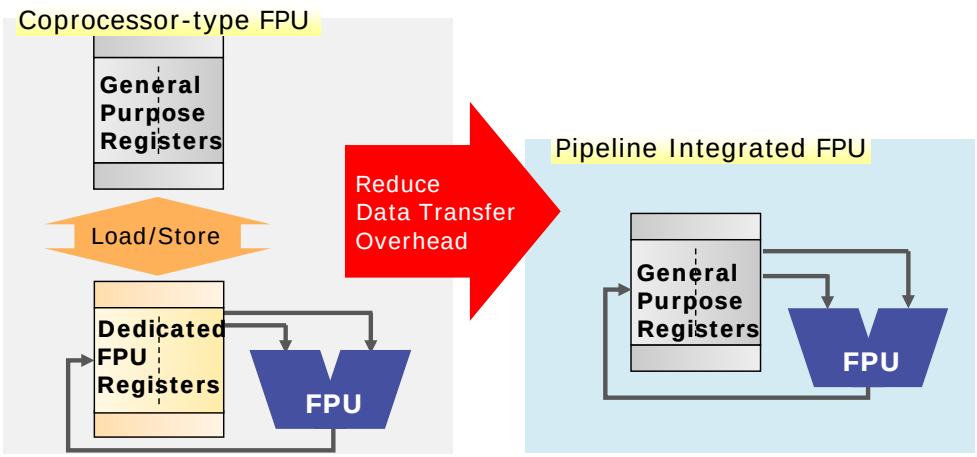


図 4.8: RX FPU アーキテクチャ

近年，モーター制御，ファクトリオートメーション，オフィスオートメーションなどのさまざまなアプリケーションで，高度なリアルタイム制御を実現する上で高い数値安定性と広いダイナミックレンジが重要な要素となってきており，ローエンドの組み込みシステムにおいても浮動小数点演算に対するニーズが高まってきている．そして，浮動小数点演算を効率よく行う上で，ライブラリによる浮動小数点演算ではなく浮動小数点演算器（FPU）が不可欠となってきた．

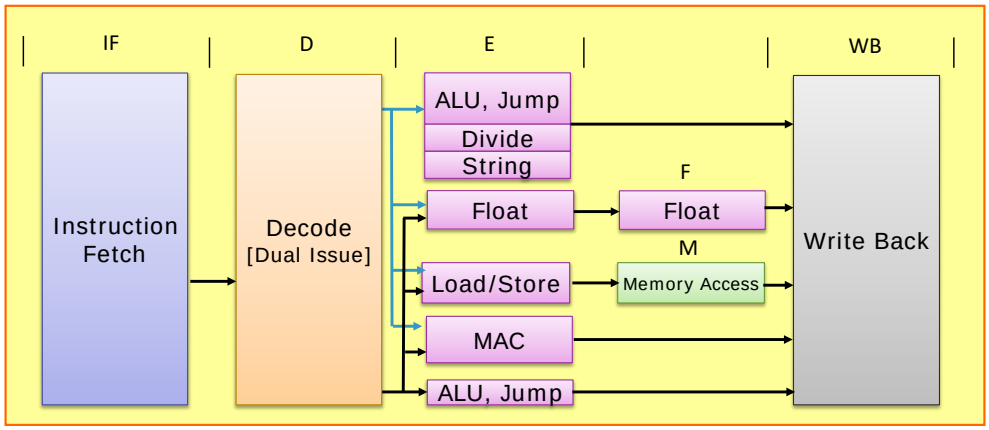


図 4.9: RX パイプライン構造

これまでにも，MCU に対してコプロセッサタイプの FPU の搭載による信号処理性能の向上が行われてきた．しかし，コプロセッサタイプのアプローチでは，個々の演算性能向上が容易に図れるメリットはあるが，非効率な専用レジスタセットが必要で，かつ，データ転送のオーバヘッドや面積が劇的に増大するという問題がある．基本命令セットを定義した後に，追加の形で FPU/DSP の命令拡張を行ったからである．そのため，面積制約の厳しい安価な MCU では，この手法はコスト観点で不利であったため，FPU 搭載がなかなか進まなかった．

そこで，RX プロセッサでは FPU 演算時に整数演算と同じく汎用レジスタを使う命令セットを定義し，FPU を整数演算のパイプラインに統合することでこれらの問題を解決した（図 4.8）．一般的な MCU で採用されるコプロセッサ型ではなく，汎用レジスタを浮動小数点用レジスタと共用し，面積だけでなく，レジスタ間オーバヘッドを削減することで，面積コスト削減と性能向上の両立を図った．

RX プロセッサの FPU 命令は，表 4.1 に示す，算術論理演算の他に，SQRT，浮動小数/整数変換等を含む 11 命令である．さらに，レジスタ割り当ての無駄をさらに減らすために，3 オペランド形式の FPU 命令を持つ．FPU ユニットは，パイプライン処理を採用することでスループットを向上している．RX プロセッサはほとんどの FPU 命令を，スループットを 1 サイクル，レイテンシが 1〜3 サイクル (FADD/FSUB:2 サイクル，FMUL:2 サイクル) で実行する．3 オペランド形式の命令の採用により，積和演算処理が高速化され，高速フーリエ変換（FFT）と無限インパルス応答（IIR）フィルターのパフォーマンスが向上した．

4.4.2　FPU/DSP を統合した RX パイプラインの概要

命令セットアーキテクチャ（ISA）を拡張するだけでは，デジタル信号処理アプリケーションのパフォーマンスを向上させるのに十分ではない．信号処理の性能向上には演算能力向上とあわせて，高いデータ供給能力が欠かせない．RX プロセッサでは FPU/DSP 演算とメモリアクセスの同時実行を可能にすることで，性能の大幅な向上を達成した．

RX プロセッサのパイプライン構成を図 4.9 に示す．RX プロセッサは 5 ステージのパイプライン構造を採用し，分岐，整数演算，浮動小数点演算，DSP 演算およびロード/ストアの 2 命令以上の命令発行に

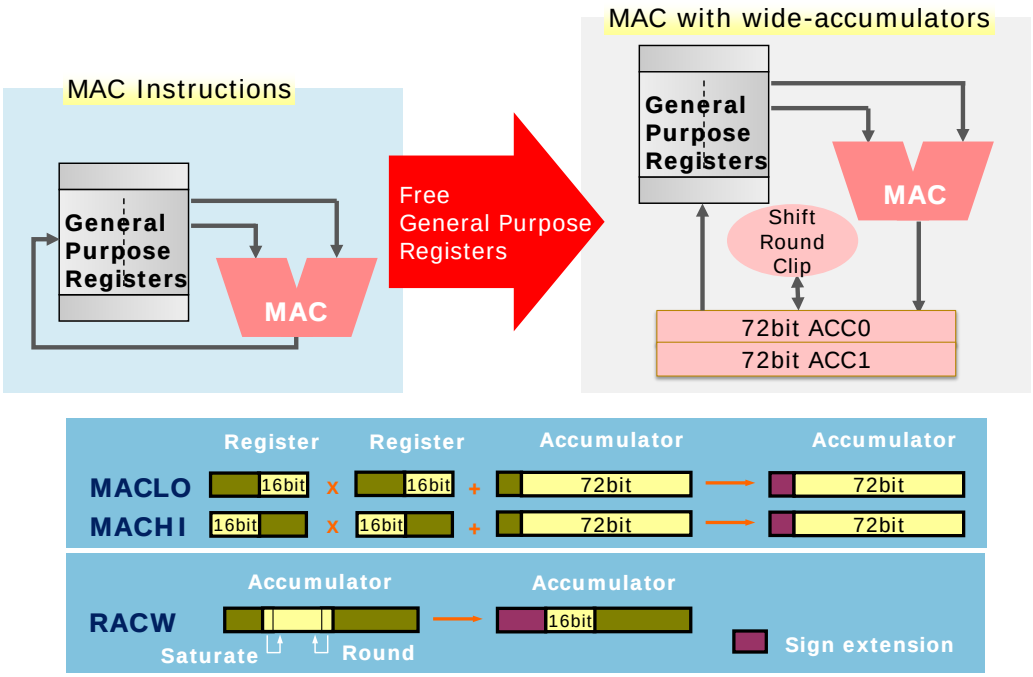


図 4.10: RX DSP 機能

より IPC（サイクルあたりの命令）を向上させている [50][51].

FPU と整数演算との汎用レジスタの共有化に加え，FPU で使用する演算器（ALU，乗算器）を整数演算と共用化し，浮動小数点のフォーマット処理とシーケンス処理を行うハードウエアをパイプラインに追加するのみで，FPU 命令をサポートした. その結果，FPU 命令追加時の面積増加は，コプロセッサタイプの FPU を追加した場合と比較して 1/5 に削減できた.

4.4.3　小型 MCU 向け DSP 機能

FPU のパイプライン統合のアプローチとは対照的に，RX DSP 命令は，計算結果を保持するレジスタとして，72 ビット× 2 本の専用アキュムレータを採用した．また，32 ビット固定小数点の積和演算にも対応できるように設計した.

汎用 DSP に相当するパフォーマンスを要求するユーザーは，固定小数点データフォーマットを用いる．この場合のアプリケーケーションでは信号処理としてフィルタ演算が頻出する．フィルタ演算では，一連のデータロードと積和演算の後に，演算結果が参照される．この場合，汎用レジスタと別に専用にアキュムレータを持った方が，ライブラリのアセンブラチューニングが容易である．アキュムレータを 2 本使用することで，固定小数点 DSP アルゴリズムのパフォーマンスを向上させることができる．たとえば，FIR は 2 つの一連の演算の各計算結果（係数*データ）が各アキュムレータに格納されるという並列性を備えているため，メモリからのデータ転送の数が削減される.

また，一般的な MCU には 16 個の汎用レジスタしかないため，レジスタリソースの不足は深刻な問題であるが，専用アキュムレータはこの問題も解決する．32 ビット固定小数点の積和演算にも対応できるよ

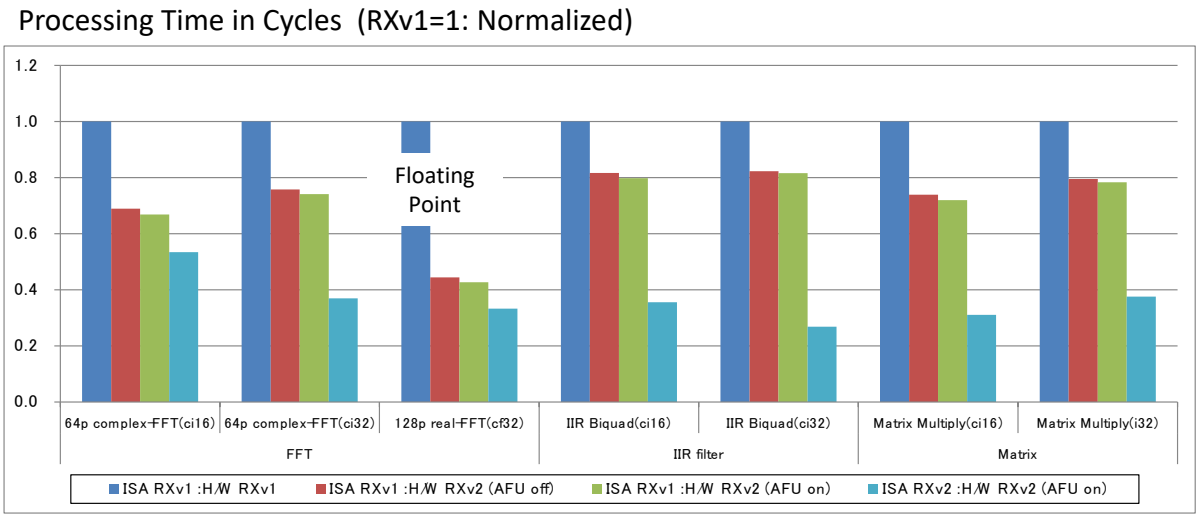


図 4.11: 信号処理ベンチマーク結果 (RXv1 vs RXv2)

うに設計した．たとえば，32 ビットの乗算（32bx32b ⇒ 64b）を実行する場合，2 つのソースレジスタと 2 つの結果格納用ディスティネーションレジスタの合計 4 つの 32 ビットレジスタを使用する必要がある．結果格納のための汎用レジスタを他の計算用に解放するために，RX プロセッサには 64 ビットの結果をアキュムレータにセーブする EMULA，EMACA，EMSBA 命令（32bx32b ⇒ ACC，ACC ± 32bx32b ⇒ ACC）がある．これらの命令は，4 つの汎用レジスタではなく 2 つの汎用レジスタのみを使用する．適切な数のガードビットをアキュムレータに付加することで，オーバーフローが発生する可能性を取り除くことができる．RX DSP アーキテクチャでは，32 ビット積和演算対応のためアキュムレータ幅はワイドレンジの 72 ビットを採用した．

RX DSP 命令は，32 ビットおよび 16 ビットの固定小数点乗算および積和演算を 1 サイクルで処理する．16bit 固定小数点フォーマットでの演算は，次のように行う．MACLO，MACHI 命令は，レジスタの 16 ビットを別のレジスタの 16 ビットで乗算し，その結果をアキュムレータの値に加算する．一連の積和演算の最後に，RACW（アキュムレータワードを丸める）命令によりアキュムレータの値に対し，丸め処理と飽和処理を行い 16 ビットデータとしてレジスタに格納する．

前述のように，2 命令発行のパイプラインは，DSP 演算と並列にメモリアクセスを行うことができるため，DSP 処理に必要な高いデータ供給能力を持っている．VoIP などの一部のアプリケーションでは，更に並列度の高い演算器を搭載した専用の DSP デバイスが必要になるが，多くのセンサー，音声，オーディオといったアプリケーションを RX プロセッサに実装することができる．

4.4.4　信号処理性能の評価

フィルタープログラムなどの数値演算機能を含む DSP プログラムによるベンチマークを行い，DSP 演算能力の評価を行った．図 4.11 と図 4.12 に，16 ビット固定小数点，32 ビット固定小数点および単精度浮

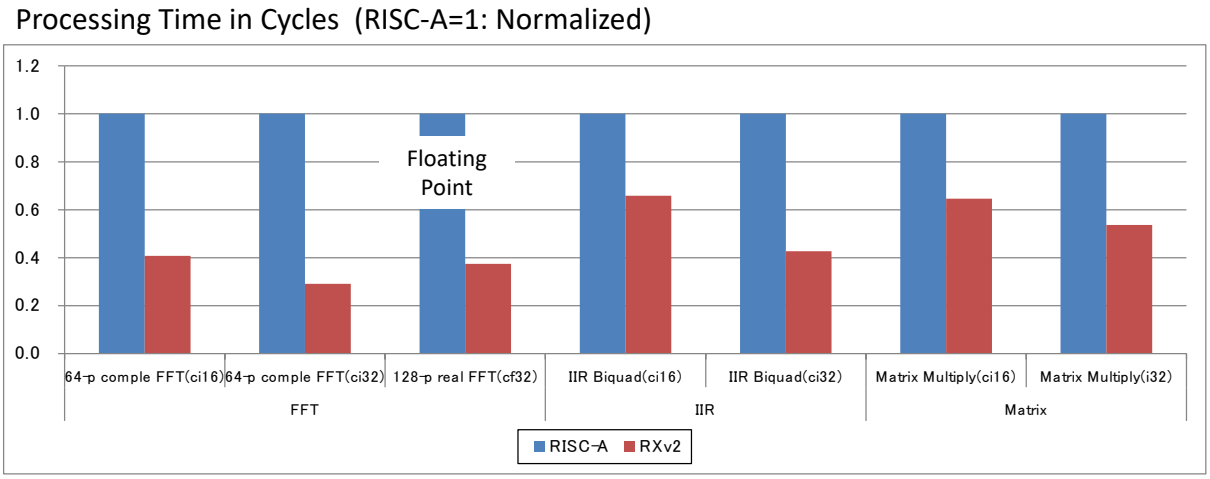


図 4.12: 信号処理ベンチマーク結果 (Cortex-M4 vs RXv2)

動小数点の 3 つの条件下で FFT，IIR，および Matrix を実行した場合の比較を示す．図 4.11 では，AFU を搭載しない第一世代の RX（RXv1）を 1 と正規化し，AFU を搭載した RXv2 の AFU をオンした場合，AFU をオフした場合，命令セットアーキテクチャの差異も含めて比較している．

まず，パイプライン構成を 1 命令発行から 2 命令同時発行にすることで，演算処理性能とデータ供給能力が向上した．さらに，AFU による分岐先コードのキャッシングなどにより命令フェッチ効率が向上した．最後に命令セットを RXv1 から RXv2 にし，アキュムレータを 1 本から 2 本に増やしたことで，固定小数点信号処理性能が向上した．命令セットを RXv1 から RXv2 にした場合の，FPU の性能向上は FPU 演算器のスループット向上によるものである．RXv2 プロセッサは，RXv1 プロセッサの 1.9〜3.7 倍のサイクル性能を実現し，DSP マイコンとしての性能を大幅に向上させている．

また，図 4.12 に，FFT，IIR，および Matrix のベンチマークについて RISC ベースの組込みプロセッサ (ARM Cortex-M4) との比較を示す．ARM Coretex-M4 に対しては 1.5〜3.4 倍のサイクル性能を実現している．逐次実行型パイプラインの Cortex-M4 に対し，RXv2 は FPU/DSP 演算とデータロードを並列に行うことができるからである．

RX アーキテクチャには RXv1，RXv2，RXv3 の 3 世代がある．表 4.6 に RX アーキテクチャ比較をまとめる．最新の RXv3 は，RXv2 から倍精度 FPU 追加や同時実行命令組みあわせ強化などを行っている．

4.5　RX MCU の実装と評価

RX プロセッサを搭載した MCU テストチップを 40nm の FLASH 混載低消費電力 CMOS プロセスで製造した．テストチップのチップ写真を図 4.13 に示す．この MCU テストチップは，4MB の内蔵 FLASH メモリ，内蔵 SRAM と RX プロセッサが搭載されている．また，RX プロセッサを周辺 IO に接続するための内部マルチレイヤバスと，割り込みコントローラユニット（ICU）も搭載されている．

表 4.6: RX アーキテクチャ比較

項目	RXv1	RXv2	RXv3
アーキテクチャ	32bit RX アーキテクチャ		
汎用レジスタ	32bit x 16 本		
命令数	90	109	113
パイプライン	5 段		
	逐次実行	メモリアクセスと演算の並列実行	同時実行命令の組み合わせ強化
DSP 機能命令	シングルサイクル MAC 命令 (16 ビット), アキュムレータ 1 本	シングルサイクル MAC 命令 (16 ビット，32 ビット), アキュムレータ 2 本	
FPU	単精度浮動小数点演算命令		単精度/倍精度浮動小数点演算命令
性能	3.12 CoreMark/MHz	4.55 CoreMark/MHz	5.82 CoreMark/MHz

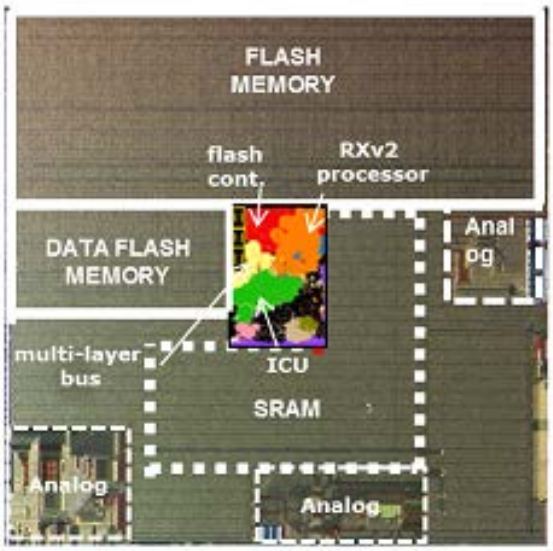


図 4.13: RX MCU テストチップのチップ写真

4.5.1　CoreMark による処理性能と消費電力の評価

EEMBC(EDN Embedded Microprocessor Benchmark Consortium) スイート [52] [53] は，幅広い組み込みアプリケーション領域のマイクロプロセッサやマイクロコントローラの性能を測定するベンチマーク・コレクションである．合成ベンチマークではなく，実際の組込みアプリケーションから直接導き出さ

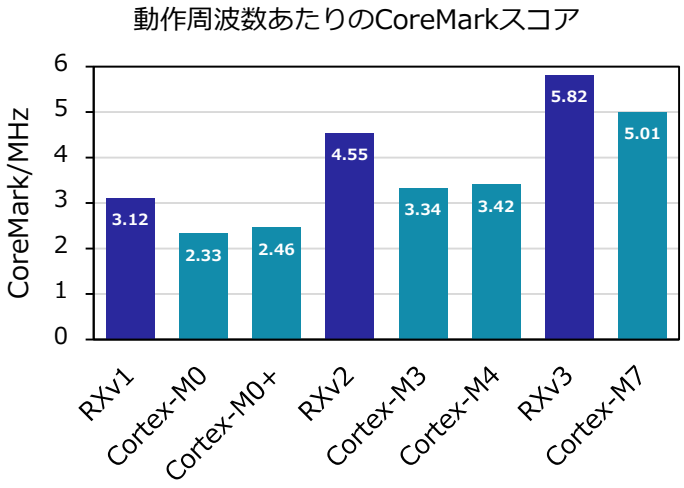


図 4.14: CoreMark による演算性能評価結果

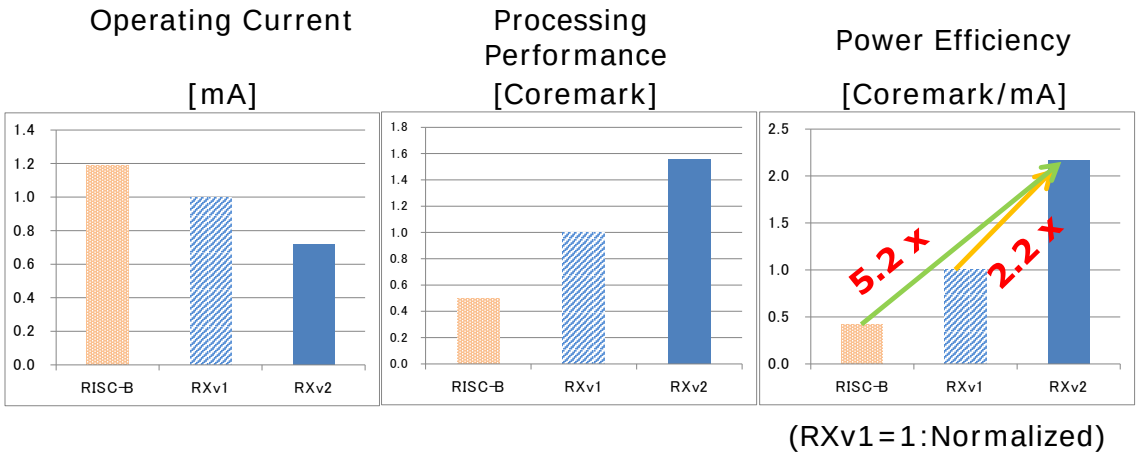


図 4.15: CoreMark による電力性能評価結果

れたものである．すべてのベンチマークは標準 C 言語でコード化され，入力データと出力データの複数の参照セットが含まれる．EEMBC スイートの主な指標は実行スループットで，プロセッサが特定のベンチマークを 1 秒間に繰り返す回数（iterations/sec）である．組込みプロセッサの多様な領域内での公正な比較を可能にするために，EEMBC スイートには，主要な組込みアプリケーション・ドメインごとにベンチマーク・カテゴリを提供している．

EEMBC CoreMark [54] [55] は，組込みシステムで使用される CPU コアの性能を測定する整数演算ベンチマークである．組込み CPU コアの性能指標として 2021 年時点で，古くからある Dhrystone ベンマークの代わりに広く使われている．CoreMark には，リスト処理（検索とソート），行列操作（一般的な行列演算），ステートマシン（入力ストリームに有効な数値が含まれているかどうかの判定），CRC（周期的冗長性チェック）などのアルゴリズム実装が含まれる．

上記 CoreMark ベンチマークで処理性能と消費電力の評価を行った．RX プロセッサの CoreMark 値

は，4.5CoreMark/MHz@120MHz であった．RX プロセッサは，既存の製品と比較して，さまざまなパフォーマンスで 50 %〜150 %の改善を達成した．その結果，競合する RISC マイクロコントローラよりも優れたパフォーマンスが得られる (図 4.14).

図 4.15 は，CoreMark による消費電流と CoreMark 値および電力効率を評価したデータであり，RX プロセッサが RISC ベースの組込みプロセッサ (SH-2A) の 5.2 倍の電力効率を達成することを示している．図 4.14 と図 4.15 に示すベンチマーク評価の結果より，RX プロセッサのパフォーマンスが他の RISC ベースのプロセッサと比較して十分な電力効率であり，効率の優れた命令セットと AFU による FLASH メモリアクセス数の削減が，演算性能の向上と低消費電力化に効果的であると言える．

4.6　関連研究

4.6.1　命令フェッチの削減による低電力化

MCU では，ROM と RAM は従来よりチップに内蔵されており，外部メモリを排除することで，低消費電力を実現しているが，最近では，バッテリー駆動や太陽光による電力で動作するデバイスからは，さらに低消費電力化が求められている．組込み MCU のシステムの場合，命令フェッチはチップ電力の大部分（約 50 %）を占める可能性があり，メモリアクセスを削減するために，フェッチするビット数を減らすためにプログラムを圧縮したり，メインメモリへのアクセスを除外する効率的な命令キャッシュを設計するといったいくつかのアプローチが提案されている．

プログラムサイズの圧縮手法として主に２つの手法がある．１つはコード圧縮で，もう１つは命令コード自体のサイズ削減である．コード圧縮アーキテクチャは，ハードウェアを使用して，最も一般的に実行される命令を圧縮する．この方法は，命令ごとのメモリアクセスを削減する [56]．この方法の欠点は，圧縮されたコードの解凍によって引き起こされるチップ領域とパフォーマンスのオーバーヘッドである．命令セットアーキテクチャによるコードサイズの縮小化は，エネルギーの低減に直接的に寄与し，圧縮のためのハードウェアを必要としない．もう一つのアプローチは，プロファイリングによって組み込みアプリケーションの共通機能を利用することである．これにより，アーキテクチャをカスタマイズして，特定のアプリケーションを最も効率的に実行できる [57][58]．このタイプのアーキテクチャ調整の非常に積極的な形式には，アプリケーション固有の命令セットと呼ばれるカスタマイズされた命令セットの作成が含まれる．キャッシュを含むメモリ設計も，アプリケーション用にカスタマイズされている．

4.6.2　効率的な命令キャッシュ設計

メモリアクセスを削減する別のアプローチは，効率的な命令キャッシュ設計である．いくつかの小さな命令キャッシュアーキテクチャは，低消費電力向けプロセッサの更なる低消費電力化に対して有効である．

フィルターキャッシュは，ダイレクトマップ方式の小さなキャッシュで，CPU と L1 キャッシュの間に配置される．標準のタグ比較を行うキャッシュであり，ミスした場合次に L1 キャッシュにアクセスを行う．フィルターキャッシュは L1 キャッシュよりも面積的に遥かに小さく，アクセス時間も短いため，アクセスあたりの消費電力は低くなる．ただし，ミス率が高く，レイテンシーが余分に増える分，全体的な

パフォーマンスが低下する可能性がある [59]．ループキャッシュは，タグのないプロセッサと緊密に統合されている小さな命令バッファで，ループキャッシュコントローラーは，後方への短距離分岐命令を使った，ネストしない単純ループが検出された時に，ループキャッシュを埋める役割を果たす [60].

分岐ターゲットバッファは，分岐予測パフォーマンスを向上させる一般的な方法である [47]．パイプラインの分岐ペナルティを削減するために，分岐ターゲットバッファは，分岐命令をデコードする前に，命令コードをメモリから投機的にフェッチする．フェッチされた命令の PC（プログラムカウンター）値が予測バッファ内の PC 値と一致する場合，対応する予測された分岐先 PC 値が投機的命令フェッチに使用される．このメカニズムにより，余分なメモリアクセスが発生し，消費電力が増加する．したがって，メモリからの命令コードを再利用する分岐ターゲットキャッシュは，低消費電力のための有効な手法である [61].

4.7 まとめ

厳しいコスト（面積）制約上，SIMD などの大規模アクセラレータによる性能向上策を使えない IoT 向け MCU を念頭に, 命令セットを最適化した CPU アーキテクチャを提案した.

MCU に搭載されるプロセッサは，動作周波数とコードサイズの間にトレードオフが存在するため，動作周波数レンジ毎に，最適な命令セットが検討されるべきである．RX プロセッサの開発では，レジスタ構成や命令数は RISC 相当としながらも CISC 的な可変長命令セットや高機能命令を取り込むことで，高いコード効率と電力効率の実現を図った．また，CPU 内の演算器数およびレジスタ数に対して，コスト制約を考慮して命令フォーマットの選択やパイプライン構成の設計も行った.

IoT 向け実アプリケーションにおいて，可変長命令セットと高機能命令の組みあわせで，一般的な RISC アーキテクチャ（ARM Thumb2）に対して，最大 46 %コードサイズを効率化した．静的コードサイズの削減は MCU に搭載されるメモリ容量を削減し，厳しいコスト制約をクリアできる.

メモリアクセス削減は，メモリの消費電力の削減にもつながる．効率的な命令セットによる動的コードサイズ削減に加え，電力オーバヘッドの少ない FlipFlop 型命令キャッシュの採用で，命令フェッチの効率化と低消費電力化を実現した．4.5CoreMark/MHz の高い処理性能と RISC(SH-2A) に対して 2.2〜5.7 倍の電力効率の両立を達成した.

第 5 章

車載 MCU 向け高信頼プロセッサ RH850 の設計

今日，車載 MCU では E/E アーキテクチャに基づく ECU(Electronic Control Unit) 機能統合の流れがあり，それを実現するために，MCU の機能統合が必須である．ECU の統合により，さまざまな安全度レベルのソフトウェアコンポーネントが 1 つの MCU の中に混在することになる．これらソフトウェアコンポーネント間の干渉を防ぐために，機能安全的観点からはリソースの分離が必須となる．このリソースの分離のために，仮想化技術の導入が考えられるが，仮想化技術のオーバヘッドのためリアルタイム性が求められる組込み用途への導入は難しかった．一方，車両制御にも高度な制御アルゴリズムの適用が求められてきており，それらの処理をアクセラレートする手段も併せて CPU には必要となってきている．

また，自動車システムで使用されている MCU は，車両システム全体が ISO26262 規格 [16] を満たすことをサポートする必要がある．ISO26262 では要求される安全度レベルに対し Automotive Safety Integrity Level(ASIL) が定義されており，ASIL-D が最も高い安全度レベルである．この ASIL-D 対応という高い要求には，高い故障検出率を得るための Field-BIST が必須である．

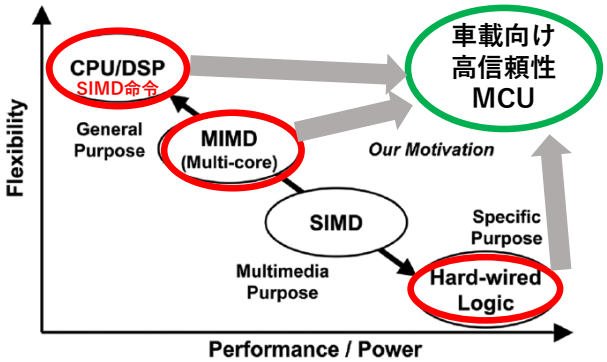


図 5.1: 車載向け高信頼プロセッサ

SoC よりも厳しいコスト（面積）制約を満たしつつ，上記 E/E アーキテクチャ対応のための MCU 統合を実現する車載 MCU を開発した．高信頼性が求められる車載 MCU に対し，高機能化 (仮想化支援機

構，SIMD 命令）マルチコア CPU と専用ハードウェア (BIST) を組み合わせた車載向け高信頼プロセッサ RH850 を提案する（図 5.1）．

本章では，ECU 機能統合に加え，車載特有の高信頼性を担保し，機能安全（ISO26262 ASIL-D）要求対応するために開発した，1) 仮想化のオーバーヘッドを最小限に抑える仮想化支援機構，2) 低消費電力のスリープモード（クロックや CPU は電源を停止）で待機するユースケースにも対応できる Field-BIST（Sleep-Resume BIST），3) ニューラルネットワーク処理をアクセラレートする浮動小数点 SIMD コプロセッサについて述べる．

5.1　仮想化支援機構

5.1.1　仮想化技術

仮想化技術は，もともとメインフレームシステムとサーバーシステムで使用されていた技術である [62]．仮想化技術により，物理ハードウェア上で仮想ハードウェアをエミュレートして，干渉なしに複数の異なるオペレーティングシステム（OS）を実行することができる．ハイパーバイザーは，OS に仮想ハードウェアを提供したり，ハードウェアリソースへのアクセスを制限したりするソフトウェアであるが，マルチコア環境においては，セキュリティのためにリソースを分離し，より安全なシステムを構築するためにマルチコアハイパーバイザーを開発してきた [63][64]．このシステムのハイパーバイザーは，ハードウェアを完全にはエミュレートしない準仮想化モデルに基づいており，パフォーマンスのオーバーヘッドを低く抑えている．各仮想マシン（VM）に，基本となる物理ハードウェアと類似しているが同一ではないハードウェアの抽象化を提示することにより，パフォーマンスを向上している．しかし，応答時間に悪影響を与えるため，自動車制御アプリケーションでの仮想化技術の使用を避けてきた [65][66]．現在のアプリケーションの複雑さは，マルチコアプロセッサによる高いパフォーマンスを必要としており，この問題を解決するために，仮想化のオーバーヘッドを軽減する仮想化支援機構を開発した．

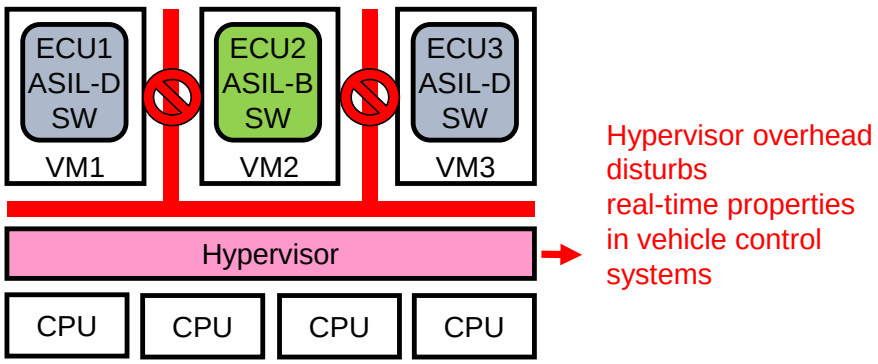


図 5.2: 仮想マシーンによるリソースの分離

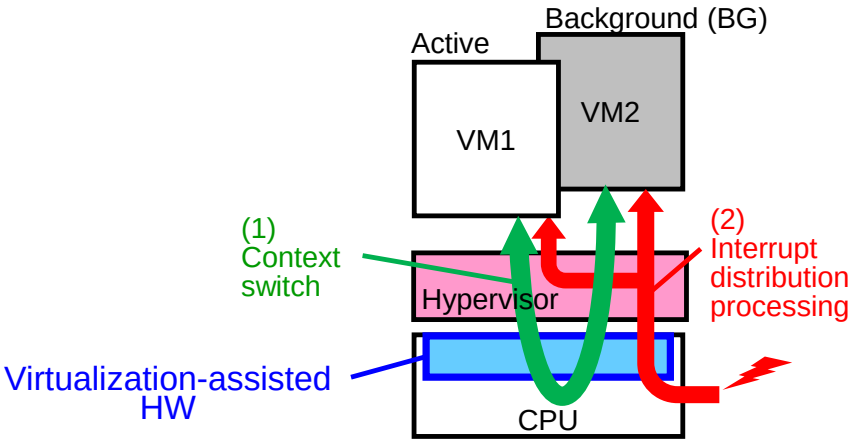


図 5.3: ハイパーバイザのオーバーヘッド

5.1.2 仮想化によるリソースの分離とその課題

安全レベルが異なる車載 ECU を統合するには，別々の ECU で動いていたソフトウェアコンポーネント間の干渉を防ぐ必要があり，そのために仮想化技術を導入する (図 5.2)．仮想化技術を使用することにより，各 ECU 上のソフトウェアは，分離された仮想マシン（VM）で実行され，VM 間のデータと時間の干渉が抑止される．VM をハイパーバイザーを用いて制御するが，ハイパーバイザーのオーバーヘッドは，車両制御システムのリアルタイム応答を阻害する．

図 5.3 に，2 種類のハイパーバイザーによるオーバーヘッドを示す．1 つ目は，VM コンテキストの切り替え時間 (1) である．1 つの CPU に 2 つの VM がある場合，1 つの VM がアクティブになり，もう 1 つはバックグラウンドになる．ハイパーバイザーは，制御期間ごとに VM のコンテキストを切り替えるが．その切り替えのためのリソースの退避および復帰にかなりの時間を要する．2 つ目のオーバーヘッドは，割り込み要求の分配のための時間 (2) である．VM への割り込み要求（IRQ）が発生すると，一旦ハイパーバイザーが IRQ の要因を解析し，IRQ を適切な VM に伝える必要がある．仮想化のパフォーマンスにとって重要な指標は，これら仮想マシンのコンテキスト切り替え時間と割り込み応答時間の 2 つである．コンテキスト切り替え時間は 1μs 未満であることと割り込み応答時間のオーバーヘッドを最小限に抑える必要がある．

5.1.3 仮想化支援のためのプロセッサアーキテクチャの概要

車載向けの RH850 プロセッサは，1 つのプロセッサに複数の VM を生成することができ，そのプロセッサあたりの VM の数に MCU に搭載するプロセッサの数を掛けた数の VM のコンテキストを車載向け MCU は扱うことができる．そして，ハイパーバイザーは，各 VM のコンテキストをメモリ上のスタック領域に退避および復元を行う．この VM コンテキスト切り替えのオーバーヘッドを削減するため，

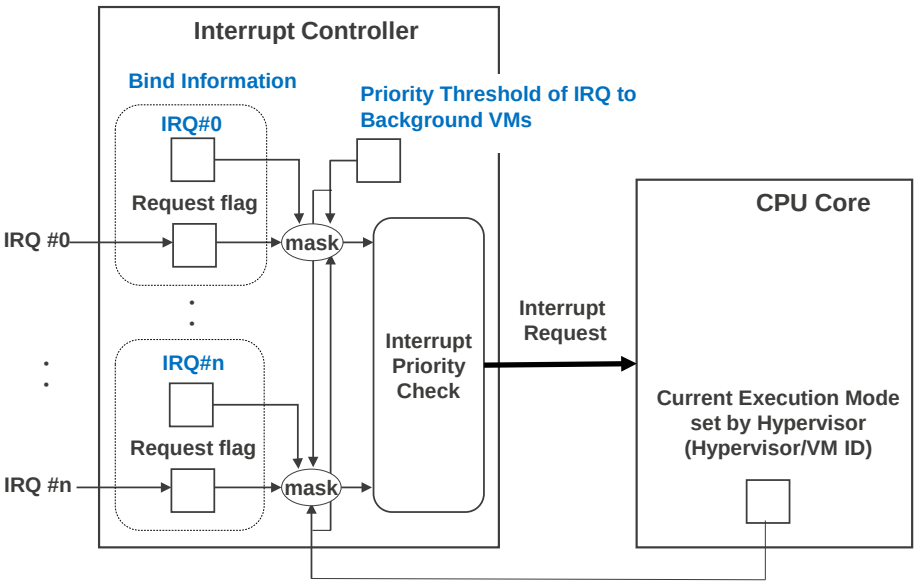


図 5.4: 割り込みコントローラ（ICU）

RH850 プロセッサには仮想化用のシステムレジスタバンクと CPU-RAM 間の広帯域転送を行う保存/復元機構を実装した．

また，割り込み応答時間のオーバーヘッドの低減のために，各割り込み要求に対応する VM を指定することができる機能（分配機能）を持つ割り込みコントローラを開発した（図 5.4）．ハードウェアには割り込み要求（IRQ）アフィニティ制御インターフェイスもあり，特定の割り込みの VM サービスを指定することができる．図 5.5 に，割り込み要求に対する状態遷移のフローチャートを示す．

VM が MCU にある複数の CPU 間で移行できることは，システムの柔軟性の観点から好ましいが．VM のコンテキスト切り替えのオーバーヘッドを最小限に抑えるには，スタックメモリのレイテンシを低くする必要がある．CPU 間で VM を移行させる場合，スタックメモリのアクセスコストを全ての CPU から均等に最小化しなければならず，結果としてハードウェアコストが膨大に増加する．そこで，リアルタイムシステムでは，VM を特定の CPU にバインドするという制約を設けることで，ハードウェアコストの削減を優先させる．スタックメモリは，アクセスレイテンシが低い CPU のローカル RAM（LRAM）に配置する．また，サーバ向けのプロセッサでは，性能低下を防ぐために，分岐予測器のコンテキストやキャッシュなどの CPU のすべてのコンテキストをハードウエアでスワップしていが，ハードウェアコストを考慮して，プログラムフローを維持するための最低限のコンテキストをスワップする仕様とした．

5.1.4　仮想化におけるオーバーヘッドの低減

仮想化のオーバーヘッドは，VM コンテキストの切り替え時間とハイパーバイザーの割り込み分配のための時間である．

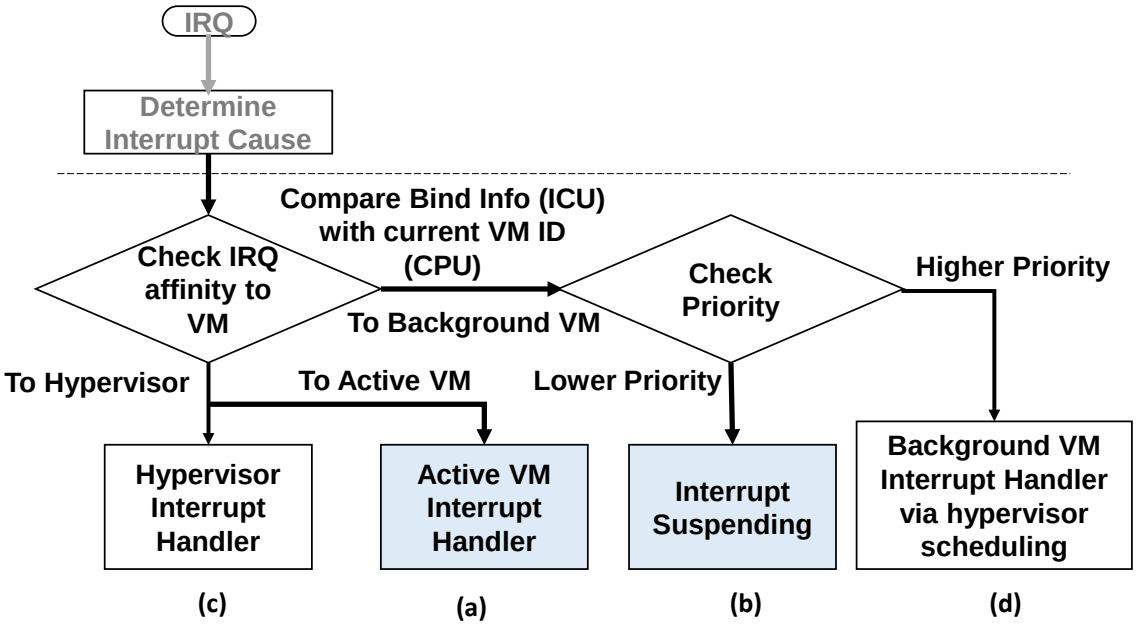


図 5.5: 割り込み要求に対する状態遷移フロー

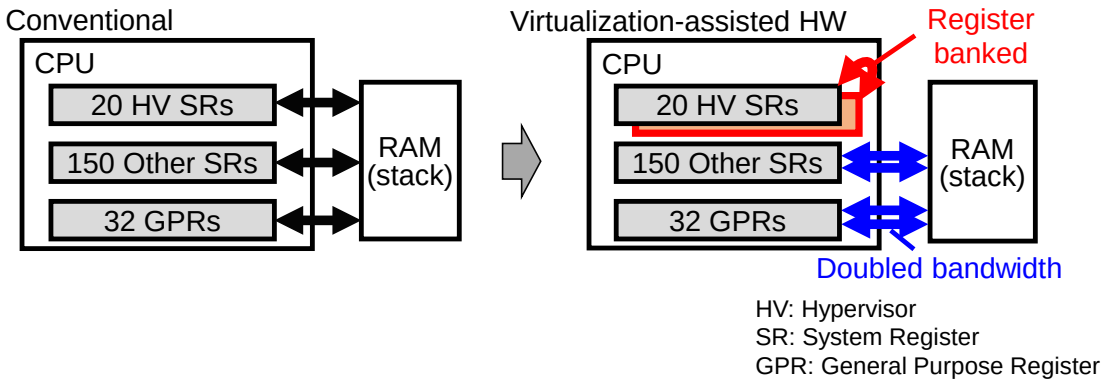
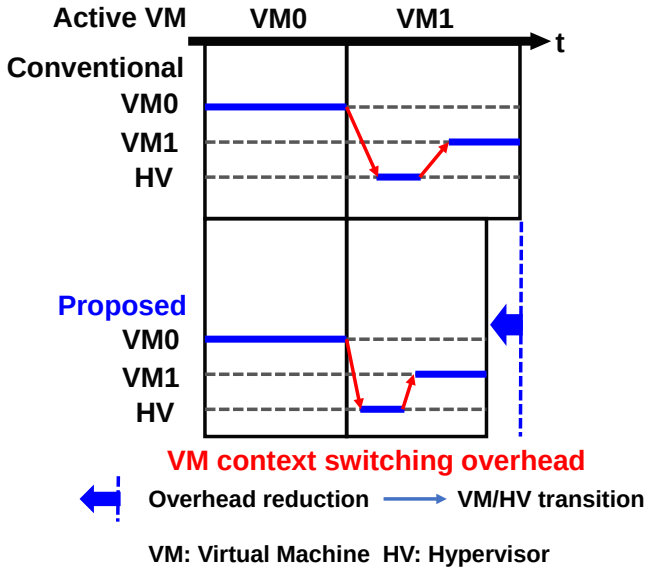


図 5.6: VM コンテキスト操作の高速化

(1) コンテキスト切り替えのオーバーヘッド低減機構

図 5.6 に，VM のコンテキストスイッチの高速化機構を示す．従来の操作では，ハイパーバイザーはスタックメモリに約 200 のレジスタの保存および復元を行う．開発された仮想化支援機構には，ハイパーバイザーによる制御に必要な 20 個のシステムレジスタを 1 サイクルで切り替えるレジスタバンクがある．さらに，CPU のシステムレジスタへのアクセスポートのバス幅を 2 倍にすることで，CPU と RAM 間の帯域幅が 2 倍になり，VM コンテキストスイッチングのオーバーヘッドを削減した．仮想化支援機構により，VM コンテキストスイッチングの実行時間は 72 %（1.56μs $\Rightarrow$ 0.44μs）短縮され，1μs の要件を満たす（図 5.7）．



Requirement: < 1 us

	VM context switching [us]		
	VM0 to HV	HV to VM0	Total
Conventional	0.75	0.81	1.56
Proposed	0.19	0.25	0.44
Reduction Rate	74%	69%	72%

600MHz CPU clock

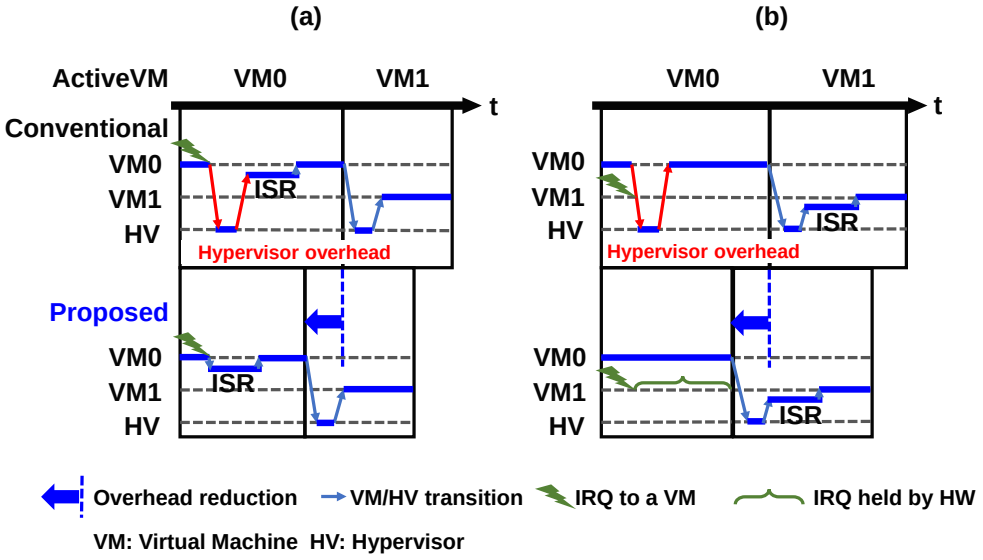
図 5.7: VM スイッチングのオーバーヘッド低減効果

(2) 割り込みの分配機能

割り込みコントローラは，アクティブ VM への IRQ が発生した場合（図 5.5(a)），アクティブ VM 上の割り込みハンドラを，ハイパーバイザのソフトウエア処理を介さずに起動する．この場合，ハイパーバイザが起動することによるオーバーヘッドをなくすことができる（図 5.8(a)）．

次に，バックグラウンド VM に対して優先順位の低い割り込み要求が発生した場合（図 5.5(b)），バックグラウンド VM がアクティブになるまで IRQ を保留することで，アクティブ VM への干渉を無くすことが可能である．この場合，バックグラウンド VM への IRQ がプロテクトされ，要因解析のためのハイパーバイザへの遷移/復帰の時間が削除される（図 5.8(b)）．この割り込み保留機能は，割り込み優先度と VM 優先度に応じた VM の切り替えを制御する．

車両制御システムでは，VM 間の干渉は通常，機能安全の観点から抑止しなければならないが，この割り込み保留機能により，バックグラウンド VM への優先度の低い IRQ は，アクティブな VM に干渉しない．しかし，車両制御に影響を与える障害が検出されると，システムは直ちに安全な状態に移行する必要がある．このような緊急割り込みには高い優先度が割り当てられる．優先度の高い IRQ が発生した場合（図 5.5(d)）は，干渉禁止を解除して，その優先度の高い IRQ が VM に分配される．この場合，システム



	(a) IRQ to the Active VM [us]		
	VM0 to HV	HV to VM0	Total
Conventional	0.07	0.08	0.15
Proposed	0.00	0.00	0.00
Reduction Rate	100%	100%	100%

	(b) Low priority IRQ to a background VM [us]		
	VM0 to HV	HV to VM0	Total
Conventional	0.06	0.08	0.14
Proposed	0.00	0.00	0.00
Reduction Rate	100%	100%	100%

600MHz CPU clock

図 5.8: 割り込みのオーバーヘッド低減効果

は直接通常の状態に戻らないため，割り込み処理の待ち時間は問題にならない．図 5.5(c) の場合，割り込みはハイパーバイザーに分配される．この場合は，VM への割り込みやオーバーヘッドは発生しない．

割り込みコントローラーは，自動車システム構成に基づいてバックグラウンド VM に送信される割り込み優先度レベルのしきい値を設定できる．通常状態での最悪の場合の割り込み処理レイテンシは，図 5.8(b) に示すバックグラウンド VM に送信された優先度の低い IRQ の中断時間である．この VM がバックグラウンドに入ってからアクティブに戻るまでの時間は，システムに依存する．

割り込み分配時に発生するハイパーバイザーによる遷移オーバーヘッド時間は，これら 2 つのケース ((a),(b)) で 0.0μs となる（図 5.8）．追加のハードウェアの面積オーバーヘッドは，CPU の面積の 3 %であり，チップ全体の面積の 1 %未満であり，電力/エネルギーのオーバーヘッドは，チップのオーバーヘッドの 1 %未満である．

5.2　高信頼化のための BIST 技術

5.2.1　フィールドテスト対応の設計技術

ISO26262 規格に対応した自己診断機能である Field-BIST は，製造試験後の実際の動作環境の中で実施されるテストの一つである．この車載 MCU に対するテストにより，期待値と一致しない結果が発生すると，関連するエラーメッセージがホストシステムに通知され，その後，自動車システムは安全性の確認や対策をとることになる．この Field-BIST には，車載 MCU をテストする上で，指定されたメトリックや限られたテストアプリケーション時間などといった，いくつかの満たすべき制約が存在する [67].

ASIL-D での 1 時間あたりのランダムなハードウェア障害は 10^{-8} 未満でなければならないという条件（PMHF:目標故障率）がある．MCU デバイスが ASIL-D [16] を満たすためには，単一箇所における故障検出率 (SPFM) と潜在的な故障検出率（LFM) の条件をクリアする必要がある．SPFM は，回路のある箇所が故障した場合の検出確率で，これを 99 %以上，一方 LFM は，安全機構自身の故障を検出する確率で，これも 90 %以上が求められる．

車両制御アプリケーションは，シャーシ&安全システム，パワートレインシステム，ボディコントロールシステムの 3 つのシステムに分かれており，シャシー&セーフティシステムには，電動パワーステアリング，アンチロックブレーキシステム，安全のためのエアバッグなどのステアリングおよびブレーキ機能が含まれている．ボディコントロールシステムには，パワーウィンドウ，オートエアコン，ヘッドライトの方向制御が含まれる．表 5.1 は，車載アプリケーション間での MCU/SoC のフィールドテスト方法の比較をまとめたものである．

BIST は，必要な LFM を達成するために安全機構（Safty Mechanizum）を診断するためのテスト回路である．パワーオンセルフテスト（POST）時に安全機構の障害を検出するため，BIST が各起動サイクル（機能の動作を開始する前のチップのパワーオン手順中の期間）中に実行される．ハードウェアの面積コスト低減とテストアプリケーションの時間要件の両方を満たすために，動作中に時分割で行うランタイム BIST が，ADAS 向け SoC に採用されている [67]．このランタイム BIST は，動作中に定期的に診断するため，車載アプリケーションの実行時に障害を検出することができる．

表 5.1: フィールドテスト手法の比較

Category	Application	POST	Sleep-Resume	Runtime Test
SoC for driver assistance systems (ASIL-B)	ADAS	BIST	-	BIST w/ time slice
MCU for Vehicle control (ASIL-D)	Chassis & Safety Powertrain	BIST	-	LSDC
	Body Control	BIST	BIST w/time slice	LSDC

5.2.2　ロックステップ デュアルコア（LSDC）

車両制御システムが ASIL-D に対応するには，MCU に ASIL-D の信頼性が求められるが，そのためには BIST に加えて，より高い故障検出システムが必要である．この要件を満たすため，ロックステップデュアルコア（LSDC）方式の CPU コアを搭載している．LSDC 方式は，ロックステップで 2 つのプロセッサを一緒に実行することで，タイミングのオーバーヘッドなしに，セーフティクリティカル機能の要件を非常に簡単に満たすことがでる．LSDC 方式を構成する基本的な要素は，2 つの CPU（メインコアとチェッカーコア）と，CPU の出力を継続的に比較して障害を検出する一連のコンパレーターである [68]. 2 つの CPU を使用すると面積コスト的オーバーヘッドが生じるが，車両制御用の MCU にはリアルタイム機能が必要でありるためこの方式が採られている．MCU/SoC は，メモリにはパリティと ECC が必須である．

5.2.3　Sleep-Resume BIST(SR-BIST)

これまでボディコントロールシステムは，機能安全レベルを ASIL-B で実現してきた．近い将来の ECU の統合と集中化の傾向により，ASIL-D を必須とするシステムと混在することなり，ASIL-D に適合する高度なボディコントロールシステムを考慮する必要がでてきた．車載 MCU もこのトレンドに従う必要がある．

車両が駐車されると，車両内の MCU は，シャーシおよび安全システムとパワートレインシステムでパワーダウンされる．一方，ボディコントロールシステムの MCU は，停車中も電源が切れることはないため，POST を適用する機会はすべて失われる．この問題を解決するために，スリープモード（部分的に電

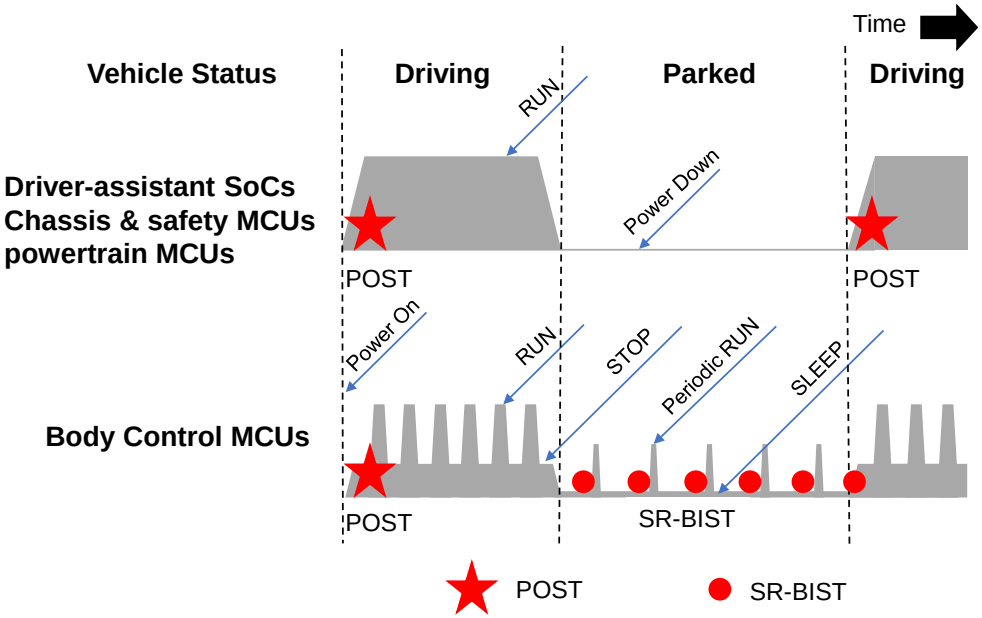


図 5.9: POST と SR-BIST

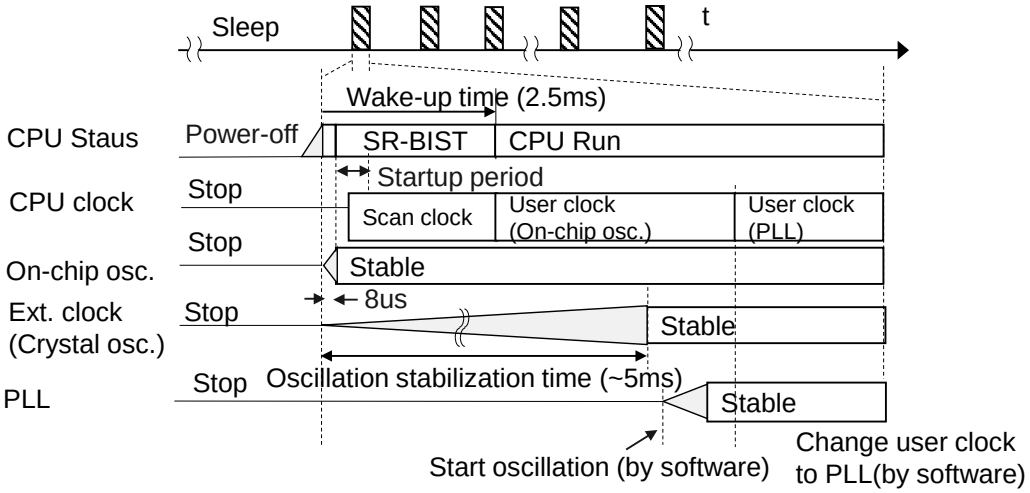


図 5.10: 専用オシレータによる高速ウエイクアップ

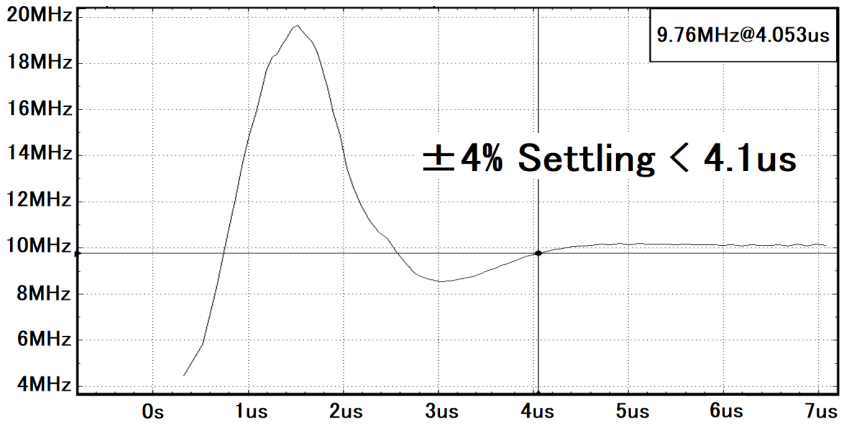


図 5.11: オシレータの周波数遷移

源が遮断された低電力モード）中に定期的に実行する Sleep-Resume BIST（SR-BIST）を提案し，駐車中でも時分割で BIST が可能となるようにした（図 5.9）．SR-BIST により，MCU が定期的にウェイクアップするタイミングで SR-BIST を実行して，キーレスエントリを含む車両のステータスを確認することができる．

SR-BIST は，CPU の処理期間を妨げないように，CPU が処理を開始する前のスリープ再開期間中に実行される．SR-BIST のタイミングチャートを図 5.10 に示す．SR-BIST は，このスリープ再開期間（2.5ms など）という時間要件を満たす必要がある．

一方，スリープ期間中，発振器は電力消費を最小限に抑えるために停止される．水晶発振器の長い発振安定時間（約 5ms）は，時間要件を超えているため，SR-BIST は，水晶発振器の代わりにオンチップ発振器をソースクロックとして使用した．このオンチップリングオシレータの安定化時間はわずか 8μs（5.11）であり，ウェイクアップ時間要件に対して十分に短い [69].

5.2.4 電流変動の抑止機構

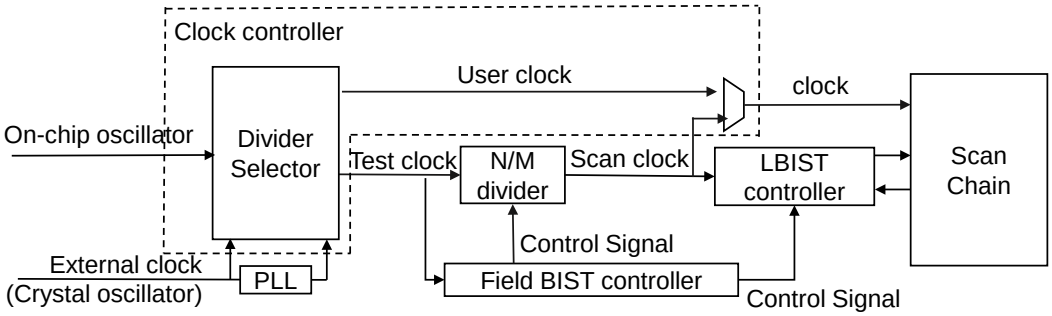


図 5.12: di/dt 抑制回路

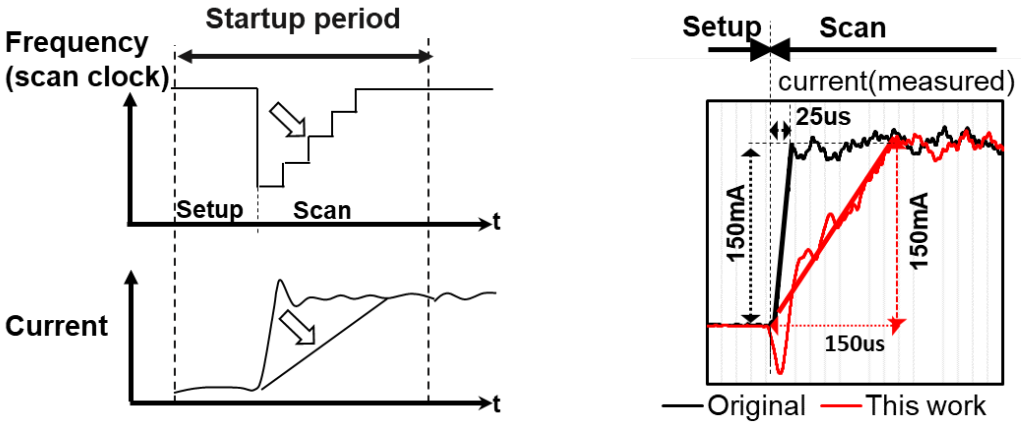


図 5.13: di/dt 抑制回路の評価結果

さらに，ECU の下位互換性により，車両制御 MCU の電流変動（di/dt）は，前世代の MCU 以下であることが求められる．Field-BIST 時には Logic BIST を行うが，スキャンクロック周波数が高い場合は，スキャンチェーンのシフト操作により di/dt ピークが発生する．この問題を解決するため，SR-BIST に di/dt 抑制回路を搭載した（図 5.12）．

SR-BIST ブロックは，クロックコントローラー，N/M 分周器，LBIST コントローラー，およびフィールド BIST コントローラーで構成されている．スリープモードからの再開シーケンスが発生すると，クロックコントローラはオンチップオシレータクロックを 8 分周する．この分周されたクロックは，オンチップオシレータの安定化が完了した後のテストクロックとして使用される．フィールド BIST コントローラーは，LBIST コントローラーを使用してターゲット回路のスキャンテストを開始する．スキャンテストが終了すると，LBIST コントローラーは結果をフィールド BIST コントローラーに送信する．クロックコントローラは，オンチップオシレータクロックを 2 分周する．この分周クロックはユーザークロックとして

使用される．

各 SR-BIST の最初の di/dt は，N/M 分周器（N=1-16，M=16）を使用してスキャンクロックの周波数を段階的に増加させることにより低減される．N/M 分周器はクロックゲーティング回路で構成され，回路内の 16 ビットシフトレジスタの各ビットはクロックイネーブル信号として使用される．

図 5.13 に di/dt 抑止回路の動作概要と測定結果を示す．実験結果は，di/dt が 6mA/μs から 1mA/μs に減少し，要件を満たしていることを確認した．

BIST 動作は，SR-BIST の複数のテスト動作に分割されるが，その故障検出率は 1 つの運転サイクル（数時間など）の POST の故障検出率と同じである．したがって，SR-BIST の POST に対する追加の電力消費は無視できる．SR-BIST は，高度なボディシステムで ASIL-D の基準である LF メトリック 90 %以下の達成に貢献する．

5.3　車両制御 AI 実現のための浮動小数点 SIMD コプロセッサ

5.3.1　浮動小数点 SIMD（FP-SIMD）コプロセッサの概要

車両制御におけるセンサーデータのデジタル信号処理にニューラルネットワーク（NN）が使用されるようになり，NN 処理の高速化が求められるようになってきた．しかし，サーバークラスのプロセッサの GPU とアクセラレータの消費電力は 75W～150W であり [70]，自動車の要件ではこの消費電力は許容できない．消費電力の低減のために，車載 MCU 向け CPU のための浮動小数点 SIMD（FP-SIMD）コプロセッサを開発した（図 5.14）．この，FP-SIMD コプロセッサは，4 並列の浮動小数点算術演算命令とメモリアクセス命令を備えている．また，FP-SIMD コプロセッサに，128 ビット（4 ワード）× 32 エントリのベクトルレジスタ内のデータに対して移動と置換を行う機能を搭載することにより，データを効果的に

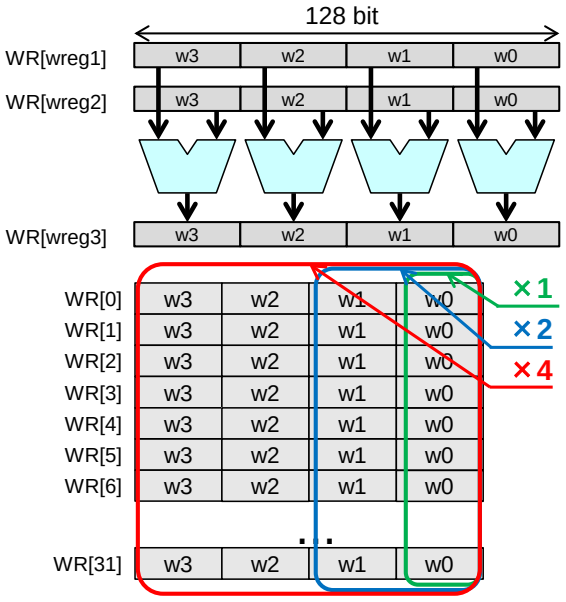


図 5.14: 浮動小数点 SIMD コプロセッサ

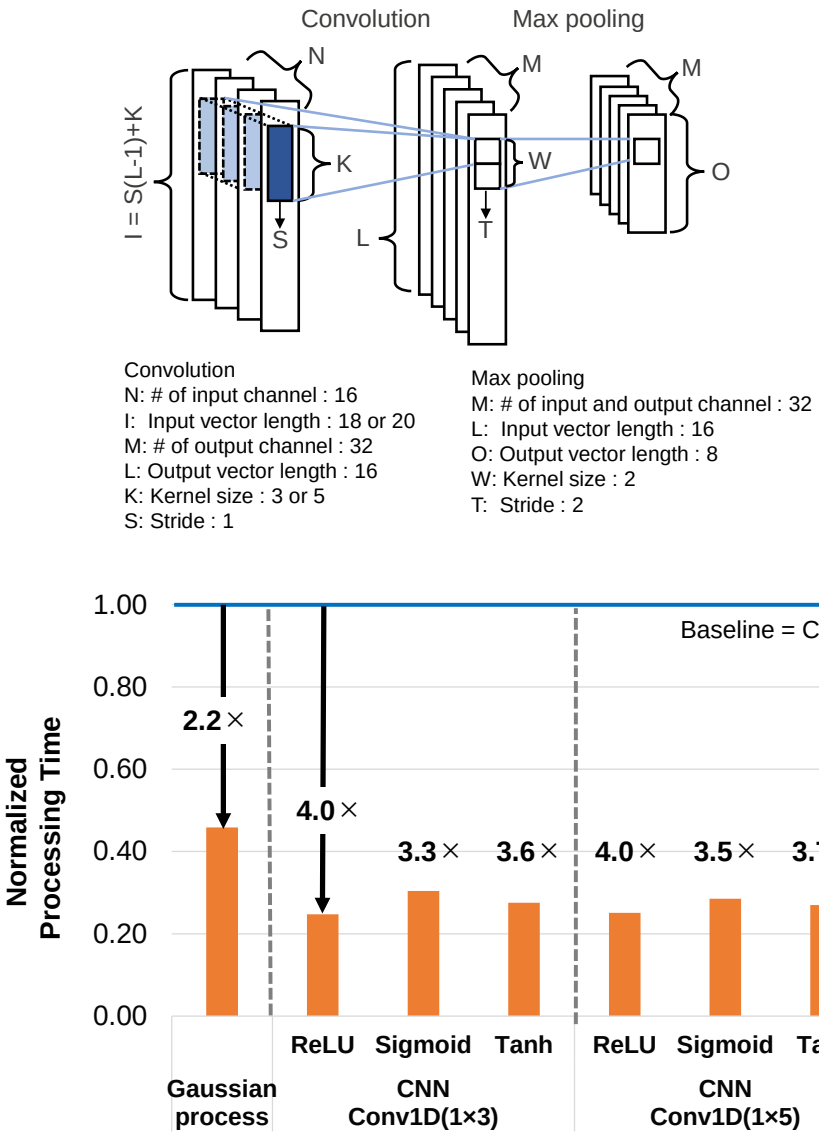
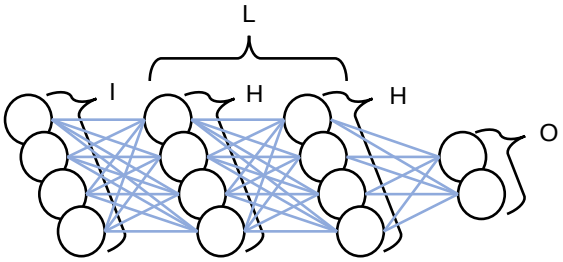


図 5.15: ガウスプロセスと CNN の性能評価

操作することができ，積和演算命令と共にこれらの機能は，NN アプリケーションで効果的である．

浮動小数点データ型の畳み込みニューラルネットワーク（CNN）およびフィードフォワードニューラルネットワーク（FFNN）のベンチマーク結果を，図 5.15 および図 5.16 に示す．図 5.15 は，ガウスプロセスと CNN の処理時間の結果である．CNN の処理時間は，1 つの畳み込み層と 1 つの maxpooling で構成される．CNN の maxpooling の処理時間はごくわずか（1 つの畳み込み層の 1/100 未満）である．したがって，CNN の処理時間は，N 層を実行する場合，（処理時間/層）× N になる．

FP-SIMD コプロセッサを使用することで，CNN の実行処理性能は，CPU で行った場合の 2.2〜4.0 倍となる．一方，FP-SIMD コプロセッサで実行した場合の消費電力は，CPU の消費電力よりも約 10 %高くなる．したがって，FP-SIMD コプロセッサを搭載した CPU のエネルギー効率（1/（消費電力×処理時



I: # of input neuron　: 16
H: # of hidden neuron : 16
O: # of output neuron : 8
L: # of hidden layer　: 3

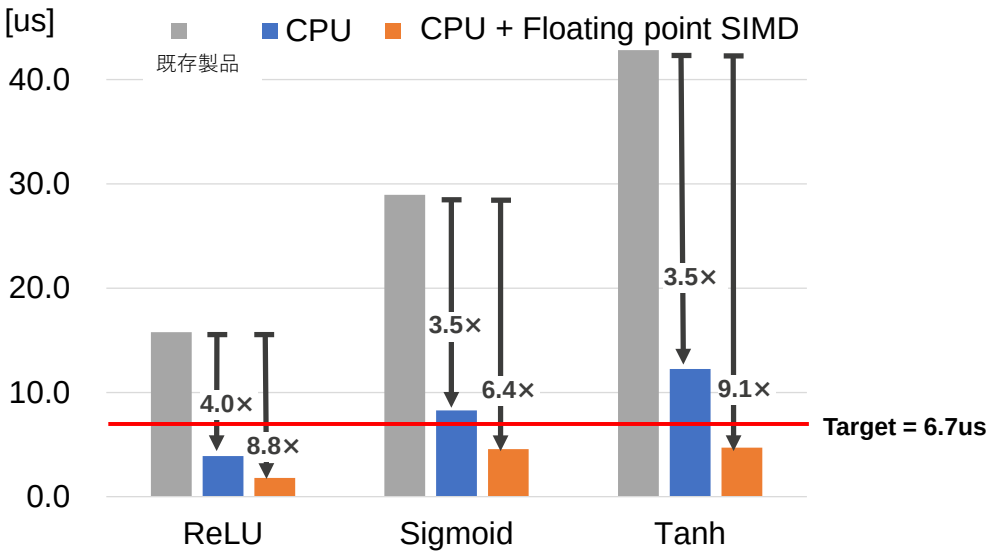


図 5.16: FFNN の性能評価

間)）は，搭載しない CPU の 2.0〜3.6 倍となる．

FFNN の結果は，図 5.16 に示すとおりである．FFNN の実行処理性能は，CPU で行った場合の 1.8〜2.6 倍となる．また，FP-SIMD コプロセッサを備えた CPU が FP-SIMD コプロセッサを搭載していない従来の製品 [71] に対し 6.4〜9.1 倍の処理性能となった．実行時間の要求である 6.7μs のターゲットを FP-SIMD コプロセッサの搭載により達成することができた．

5.4　RH850 車載 MCU の実装と評価

車載向け高信頼プロセッサ RH850 のテストチップ（RH850 車載 MCU）を開発し，28nm SG-MONOS FLASH メモリ混載，high-K メタルゲート CMOS プロセスで製造した [72]．多くの機能を 1 つの MCU に統合するには，半導体ダイの集積密度を上げる必要があるが，これは，プロセス技術の向上によって実現することができる．一方，ECU では水冷/空冷やサーマルシートなどの冷却方法を使用できな

表 5.2: RH850 テストチップ諸元

Technology	28nm HKMG CMOS process w/ SG-MONOS embedded Flash
Clock Frequency	Processor : 600MHz max.
Chip Size	11.5mm x 8.2mm
Supply Voltage	3.0〜5.5 V with embedded DC/DC-ctrl
Operating Temperature	Tj= -40°C〜170°C
Power Consumption	521mW@25°C, Typical
CPU	32bit Processor (Lock-Step) x 4 MPU,FPU Floating point SIMD Virtualization-assisted function
Memory	Code Flash : 16MB, SRAM : 3MB Local RAM : 256KB
CAN I/F	16 x RSCAN-FD
Ethernet	SGMII : 1bit,1.25GHz,Serial
System, Security & Safety	{safety} Stress Monitor, Field-BIST, Over-The-Air,ECC {secutiry} Hardware Security Module {system} Sleep Modes, Clock Monitor, Temperature Sensor

いという理由で，車載 MCU への消費電力に関する要求は，熱的制約のもとで数ワットでなければならない．そのため，速度，電力，集積密度の 3 つのポイントに関して従来の 40nm 組込み FLASH プロセスではこれらの軸では不十分であるために 28nm プロセスを採用した．表 5.2 にチップ仕様を示す．

5.4.1 RH850 アーキテクチャの概要

図 5.17 に，RH850 車載 MCU 向けに開発した車両制御システムの全体的なアーキテクチャを示す．システムは，ファイアウォールによって，2 つのサブシステム（セキュアシステムとアプリケーション（非セキュア）システム）に分割されている．非セキュアシステムでは，複数の車載向けアプリケーションが実行される．

図 5.18 に RH850 車載 MCU のブロック図を示す．RH850 車載 MCU には，16MB の内蔵 FLASH メモリと 3MB の共有 SRAM と 2 つの CPU クラスターがある．各 CPU クラスターには，仮想化支援機能を備えた 2 つのロックステップデュアルコア CPU と，エラー訂正コード（ECC）機能を備えた 256KB ローカル RAM（LRAM）がある．CPU は最大 8 つの VM を生成することができ，2x2 で 4 つの CPU を備え

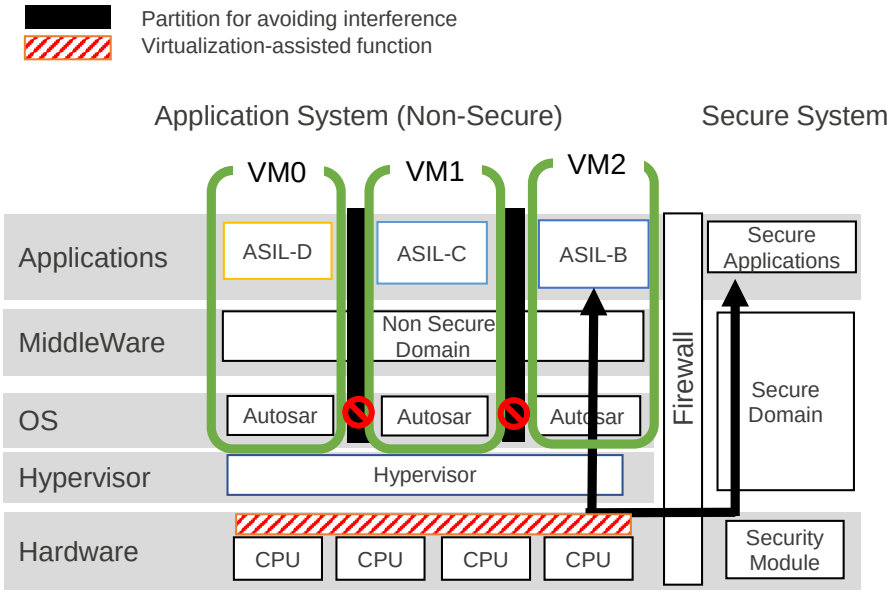


図 5.17: 車両制御のシステムアーキテクチャ

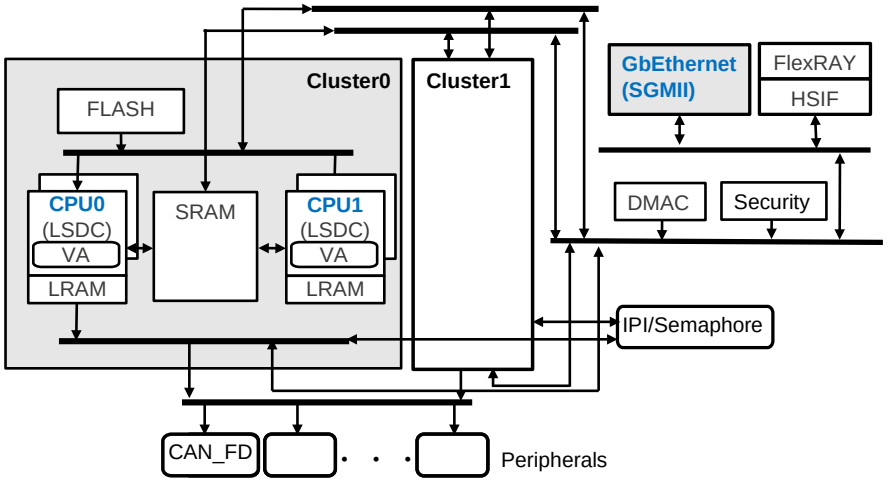


図 5.18: RH850 車載 MCU の機能ブロック図

た RH850 車載 MCU は，ハイパーバイザーと 31 個の VM の最大 32 個のコンテキストを持つことができる．CPU クラスターは，高帯域幅の内部クラスターバスと相互接続されている．また，プロセッサ間同期を実現するためにプロセッサ間同期機構（IPI）とセマフォコントローラーを実装している．RH850 車載 MCU には，SGMII 標準の 1Gbps イーサネット通信インタフェース，FlexRAY インタフェース，CAN-FD インタフェースなど，一連のネットワークインタフェースも装備されている．

CPU は，仮想化支援機能を搭載した 8 ステージのパイプラインを備えた 32bit RISC アーキテクチャの RH850 プロセッサであり，内蔵 FLASH メモリと LRAM の両方と密結合されている．CPU には，単精

表 5.3: 車載向けプロセッサ機能比較

	This Work RH850	既存製品 [71]	TriCore/TC3xx [73]	PowerPC/e200z7 [74]
Technology	28nm	40nm	40nm	-
Frequency	600MHz	240MHz	300MHz	264MHz
CPU	32bit Processor MPU,FPU Out-of-Order I-Cache:64KB	32bit Processor MPU,FPU I-Cache:32KB	32bit Processor MPU,FPU I-Cache:32KB	32bit Processor MMU,MPU,FPU I-Cache:16KB D-Cache:16KB
Virtualization Assisted Function	Yes	No	No	No
Floating-point SIMD	Yes	No	No	No

度/倍精度浮動小数点処理ユニット，メモリ保護ユニット，命令キャッシュ，および分岐予測機構をも内蔵されている．CPU の命令セットアーキテクチャは，2/4/6/8 バイトの可変長命令を採用して，コードサイズを最小化している．

この CPU と他の車載 MCU の CPU との機能比較を表 5.3 に示す．ここに示すように，開発した RH850 車載 MCU は，仮想化支援機能をサポートしない前世代 MCU [71]，TriCore/TC3xx [73]，PowerPC/e200z7 [74] とは異なり，ハードリアルタイム機能を満たす仮想化支援機能を備えた最初の車載 MCU である．さらに，40nm プロセステクノロジーで設計された TriCore/TC3xx，PowerPC/e200z7 と比較して，CPU は 28nm プロセステクノロジーを採用することで，高い周波数である 600MHz で動作することを確認した．

5.4.2 車載向けベンチマークによる性能評価

車載 MCU 向けプロセッサの性能評価のベンチマークとして，EEMBC Automotive & Industrial カテゴリ（Autobench 1.1) [75] [53] と自動車用実アプリケーション（車両制御アプリ）を用いた．Autobench のカテゴリには，エンジンやエアバッグの制御など，産業用制御機器や自動車アプリケーションから派生したタスクが含まれ，16 個のベンチマーク・プログラムから構成される．図 5.19 と図 5.20 にその結果を示す．

Autobench による評価（図 5.19）では，従来の製品 [71] と PowerPC/e200z7 [74] と RH850 プロセッサの 3 つのプロセッサを比較した．数値は， [71] に対する規格化した数値である．RH850 プロセッサは，幾何平均で [71] の 3.5 倍の処理パフォーマンスである．

車両制御アプリでは，周波数向上，CPU 数の増加と Out-of-Order 実行により RH850 プロセッサは，既

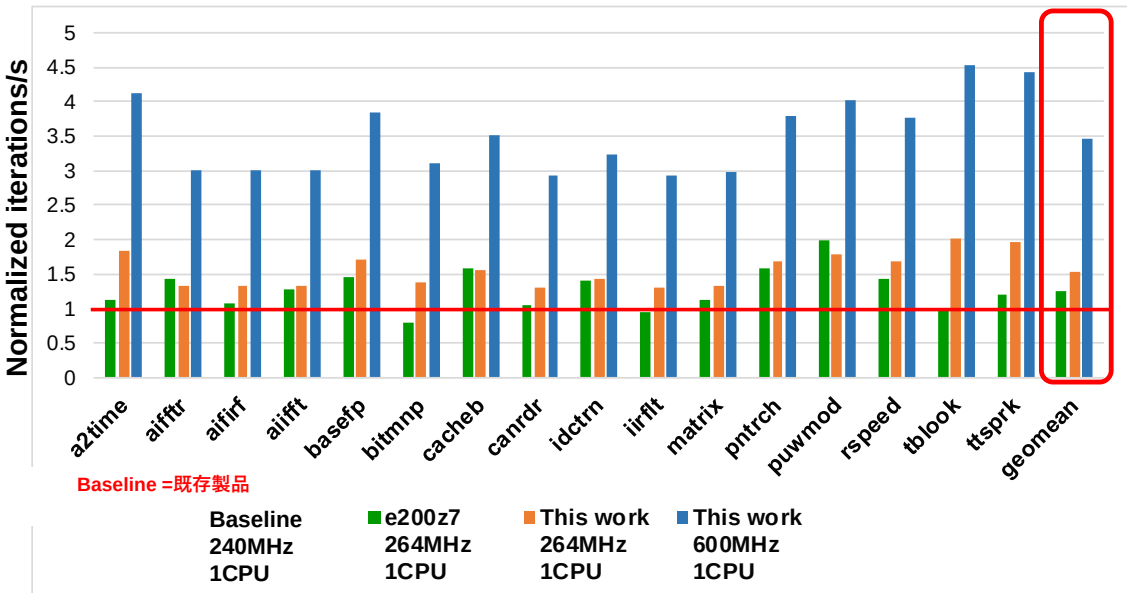


図 5.19: Autobench による CPU 性能の評価

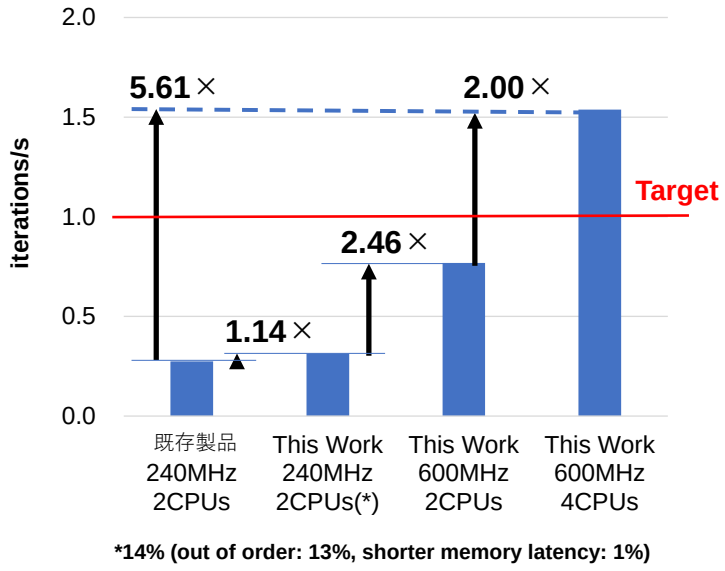


図 5.20: 車両制御アプリケーションによる CPU 性能の評価

存製品 [71] に対し 5.6 倍の処理パフォーマンスである（図 5.20）．車両制御アプリでは，テーブルルックアップが頻繁に発生するが，RH850 プロセッサはメモリのレイテンシが低く，Out-of-Order 実行のパフォーマンスの向上への寄与は 13 %である．一方，Autobench では 40 %向上に寄与している．

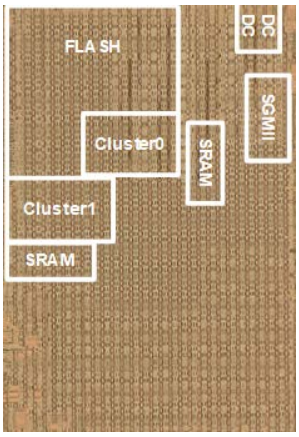


図 5.21: RH850 テストチップのチップ写真

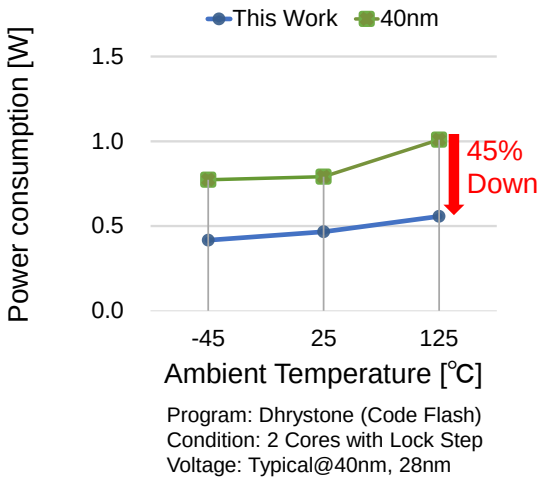


図 5.22: 消費電力評価結果

5.4.3 テストチップ試作結果

28nm SG-MONOS FLASH メモリのセルサイズは，以前に報告された 0.053μm^2 [76] から 15 %以上削減されており [77]，この削減の結果，面積の増加を抑つつ 16MB の大容量 FLASH メモリを混載することが可能となった．図 5.21 にテストチップのチップ写真を掲載する．

この RH850 車載 MCU は，標準の BGA パッケージを考慮して 5W の電力制限内で設計を行った．図 5.22 に，温度条件を振った時の消費電力評価の結果を示す．28nm のテストチップの消費電力は 125°C で 0.56W となり，40nm 車載 MCU より 45 %少ない結果であった．

5.5 関連研究

リアルタイム OS と汎用 OS の両方をサポートするマルチコアプロセッサ上のハイブリッド OS 環境が導入された．この環境にて 2 種類の OS は，マルチコア上で独立して動作し，割り込み応答とアプリケーション処理のパフォーマンスを向上させた [29] [30]．この環境では，2 つの OS をサポートするためにマルチコアハイパーバイザーのプロトタイプが実装され，OS 間の通信 IF も設計されている [78].

仮想化は，もともとメインフレームとサーバーシステムからのものであったが [62]，組込みシステム，特に車載システムにもマルチコアの普及に伴って徐々に普及してきた [79]．仮想化は，複数の異なる OS を実行するために，物理ハードウェア上で仮想ハードウェアをエミュレートすることによって実現される．ハイパーバイザーには，type1[vmwarevsphere，xenserver，hyperv，kvm] と type2[xen] の 2 種類がある．Type1 ハイパーバイザーは，物理ハードウェア上で直接実行され，パフォーマンスが向上し，コードサイズが小さくなる特殊な OS であるため，組込みシステムに適している．マルチコアハイパーバイザーはセキュリティ用途への採用も期待されている [80].

5.6　まとめ

自動運転の時代，自動車のアーキテクチャは ECU の統合と集中化に向けて急速に進化している．安全運転のために高度な車載制御がますます重要になってきているが，それらを実現するには，MCU の統合，最高の自動車安全度レベル (ASIL-D)，およびネットワークパフォーマンスの向上が必要である．これら次世代自動車アーキテクチャ向けた MCU の統合と高信頼性（ISO26262 ASIL-D）の実現ために，必須となる 3 つの技術を開発し，その実現性を実証した．

- 機能安全のための仮想化支援プロセッサ（CPU）
- 機能安全のためのスリープ/レジューム時のフィールドセルフテスト（SR-BIST）
- 車両制御 AI 実現のための低コストかつ高演算性能（浮動小数点 SIMD コプロセッサ）

この実証のため，16MB の FLASH メモリと 600MHz 動作の RH850 プロセッサを内蔵したテストチップを 28nm FLASH 混載プロセスにて試作した．CPU は，ハイパーバイザーが複数の異なる仮想マシンの実行を干渉なしに行うリアルタイム仮想化を実現することをサポートする．この RH850 車載 MCU は，リアルタイム仮想化を使用してリソースの分離を実現する一方，仮想マシンのコンテキスト切り替え時間を 72 %短縮し，割り込み応答時間のオーバーヘッドを最小限に抑え，車両制御における厳しいリアルタイム性（1μs 未満）を達成した．また，SR-BIST により，さまざまな自動車アプリケーションで 10^{-8} 以下の PMHF（目標故障率）を実現し，ASIL-D の規格を満たすことができた．今回開発した RH850 車載 MCU は，0.52W の低消費電力を実現し，ISO2262 ASIL-D と ECU 統合のシステム要件を満たすことを実証した．

第 6 章

結論

6.1 本研究の成果

本論文では，高機能な組込みシステムに適用可能なアーキテクチャに関して，SoC と MCU のそれぞれの制約条件に則して，低コストで電力，性能（演算，リアルタイム性）を最適化するコンピュータ（プロセッサ）技術を提案した．

低電力・高性能なヘテロマルチコア SoC Orochi の設計では，画像処理，AI 処理向けの SoC に対して，小面積/低消費電力のマルチコア CPU と超並列 SIMD 構成を提案し，低コスト，低電力でありながら柔軟な処理を可能にする技術を実現した．Orochi による LSI 実装を行い，実アプリケーションである位相限定相関法による 3 次元計測の場合，LSI（8CPU+2MBMX）は，従来の SMP システム (16CPU) と比較して，45 %の消費エネルギーを削減した．

低電力・高性能なインターコネクト SoC PEACH の設計では，組込み向けマルチチップ構成の実現のための高信頼インターコネクト SoC に対し，PCIe を通信 IF としデュアルブロックシステムによる通常処理とエラー処理の分離による効率化の提案を行った．マルチコア技術に専用コントローラを導入した提案構成による PEACH の実装を行い，PCIeRev2.0 を 4 ポート備えた PEARL ネットワークにおいて，Infiniband より 51.5 %高い電力効率である 0.04W/Gbps を達成した．通常処理では，インテリジェント・割り込みコントローラのデータ転送開始機能により，CPU 割り込みハンドラを使用する場合と比較して，転送処理時間を 20 %削減した．これにより，組込み向けの高性能なインタコネクトが実現可能となった．

MCU 向け低電力プロセッサ RX の設計では，厳しいコスト（面積）制約上，SIMD などの大規模アクセラレータによる性能向上策を使えない IoT 向け MCU を念頭に，命令セットを最適化した CPU アーキテクチャを提案した．この CPU では，高機能命令と可変長命令によるコード効率の向上を図った結果，IoT 向け実アプリケーションによるベンチマークで，一般的な組込み RISC アーキテクチャ（ARM Thumb2）に対し，最大 46 %コードサイズを効率化した．静的コードサイズの削減は MCU に搭載されるメモリ容量を削減し，厳しいコスト制約をクリアできる．一方，メモリアクセス削減はメモリの消費電力の削減につながる．効率な命令セットによる動的コードサイズ削減に加え，電力オーバヘッドの少ない FlipFlop 型命令キャッシュの採用で，命令フェッチの効率化と低消費電力化を実現した．既存組込み向け RISC(SH-2A) に対して 2.2〜5.7 倍の電力効率と 4.5CoreMark/MHz の高い処理性能の両立を達成した．

車載 MCU 向け高信頼プロセッサの設計では，コスト，処理性能，低消費電力の要件を満たすとともに，

高信頼性を備えた車載 MCU 向けプロセッサを構築した．MCU の統合と高信頼性（ISO26262 ASIL-D）の実現ために，高機能化 (SIMD 命令，仮想化支援機構）マルチコア CPU と高信頼のための専用ハードウェア (BIST) を組み合わせたアーキテクチャを提案した．ハイパーバイザ導入システムを対象に，上記 LSI 実装を行い，仮想マシンのコンテキスト切り替え時間を 72 %短縮し，要件（1us 以内）を達成し，車載向けの高信頼性を満たす MCU 統合方式を実証した．

高度化されていく組込みシステムを実現するためには，インテリジェントな組込みコンピュータを低コスト化かつ低消費電力化する必要がある．効率の優れた命令セットを選択すること，小面積のマルチコアと小面積のアクセラレータ（SIMD）の効率よい選択をすること，独立性のあるブロックを分離し，高信頼性とリアルタイム性を向上することの 3 つのバランスをとることにより，高効率な組込みシステムを実現することが可能である．

提案した技術により，高機能な組込みシステムに向けの SoC と MCU を低コストかつ低消費電力で実現できることを示した．

6.2　これからの組込みプロセッサの課題と展望

6.2.1　ヘテロマルチコア

組込みシステムは，決まったアプリケーションを実行するシステムである．それに用いられるデバイスは，アプリケーション特化にするより汎用の製品を開発する方が開発コストの面で有利である．なぜなら，使用されるアプリケーションの普及度（出荷数）が大きければ，そのアプリケーション特化に開発しても開発費の回収は比較的容易であるが，多くの場合，複数のアプリケーションで回収せざるをえなくなり，汎用的要素をコスト増加を抑えて行わなくてはならない．

プロセステクノロジーの微細化が進み集積度が上がるに従い，システムを構成する複数のデバイスを 1 チップにする動きは，自然な流れである．例えば，携帯電話等の普及に従い，ベースバンド用のデバイスとシステムソフトを動作させるアプリケーションデバイスが 1 チップ化されていったケースがその例である．このように，SoC は，用途別の複数の CPU が搭載されることは珍しくなかった．

微細化と電力問題から 2000 年代入って 1 チップのマルチコアが活発に研究開発された [32]．しかし，適用するアプリケーション分野もそれほどなく，またソフトウエア開発環境の問題から，SoC と MCU では，その後の状況は異なった（図 6.1）．SoC では，2010 年以降，Linux 等のマルチコア OS を前提に導入が加速した．携帯電話もスマートフォーンが普及したことも一因である．ニューラルネットによる機械学習が脚光を浴び，画像認識を代表とする AI アプリケーション向けに GPU と共にマルチコアも一般的となっていった．それらを背景に，自動運転など自動車のインテリジェント化が進み，SoC に求められる機能，処理性能もそれに伴い大きくなった．

一方，MCU では，依然としてマルチコアを活用できるアプリケーションが 2010 年を越えても見つからなかった．開発ツールが整備されていない等の原因がまず挙げられるが，ソフト環境だけでなく，リアルタイム認識等を行う SoC にと比べても，MCU が扱うアプリケーションには，単純な制御が多く，以下の点がその理由と考えられる．

- データ・セットが小さく，再利用が多い

図 6.1: アプリケーションの並列性

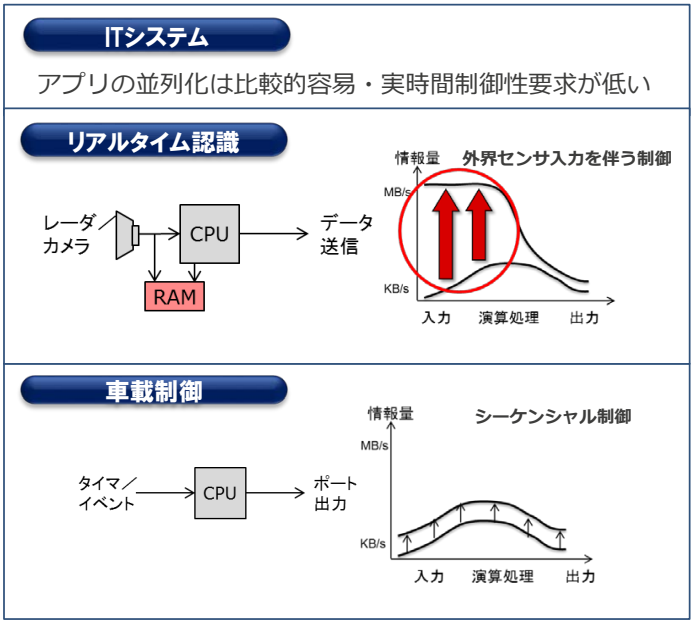


図 6.2: 情報量と並列性

- 処理時間が短く，複数の処理が複雑な起動関係で動作
- リアルタイム性の保証が必要

情報量といった観点でも，SoC と MCU では，事情が違う．図 6.2 に示す通り，SoC の扱うリアルタイム認識などのアプリケーションは，入力データが多く，そこにデータの並列性が存在するが，MCU が扱う車載制御の例では，入力から出力まで一貫して少ない情報量で処理が進み，ここにはデータの並列性も少ない．

このように，SoC は，並列化による性能アップが期待できるため，アプリケーションの特性を見ながらマルチコア化による性能向上を図るアプローチは有効な手段である．一方，MCU ではシングルプロセッサでの性能を面積コスト見合いで高めて行くことが第一の取るべきアプローチである．しかし，近年の車載システムのように，複数のシステムを統合していく場合，仮想化等の技術と共にマルチコア化が進み，

マルチコアの技術が活用できるようになる．今後，インテリジェント化の必要な処理については，開発ツールの進化も必要であるが，MCU も SoC との境界が縮まり，ヘテロマルチコアの MCU が開発されていくだろう．

6.2.2 SIMD 命令と SIMD アクセラレータ

データに並列性が存在する場合には，SIMD 的なアプローチは有効である．コストに余裕があれば，単独の SIMD アクセラレータを搭載する選択もあるが，面積コストの制約が厳しい MCU であっても，SIMD 命令による性能向上は選択の一つである．この場合，プロセッサ内に保持できるデータの数がコストと性能のバランスポイントになるためアプリケーションのデータ並列性を見極めて仕様を策定する必要がある．

CPU とは独立した SIMD アクセラレータは，そのアプリケーションにマッチした仕様であれば，処理性能と消費電力の両面で有利である．しかし，ソフト開発環境がしっかりしていないと，プログラムが増えていかず，ソフトウエアの生産性に課題を残す．汎用性が少なくアプリケーションに特化する程度が高い場合，開発環境の整備とソフト開発には大きなコストがかかるため，特化するアプリケーションのボリュームが大きくなければそれらを維持していくのは難しい．MX は，データの物理的位置を把握しながらアプリケーションを書く必要があり，また，開発サポートのためのツールも基本的なものしか無かったので，ソフト開発が難しく，アプリケーションは増えていかなかった．アプリケーションが増えなければ，ユーザのボリュームも増えず悪循環に陥った．

メモリの近傍でのプロセッシングは，組込み向けに並列性を使って処理性能を高めるためには消費電力効率的にも有利な技術分野である．今後はアナログ演算の PIM 技術も含めて，アプリケーションに適合したものも出てくると思われる．その場合に考慮する点は，ハードウエアの観点では，データの転送等の演算が開始するまでの準備に要するオーバーヘッドをいかに小さく抑えるかという点と前述のソフト開発環境である．

6.2.3 マルチチップシステム

高度な自動運転を実現するためには，より多くの演算能力が必要となるが，現在のマルチコア SoC が 1 チップでは十分ではなく，マルチチップによるシステムの性能向上を図りながら進化していくことになると思われる．いずれは 1 チップの SoC になるかもしれないが，その過程でマルチチップによる選択が行われるからである．車載の中の広域ネットワーク IF としては，イーサネットが採用されていくが，プロセッシングのためのデバイス間の通信には高性能であり汎用的（低コスト）である PCIe が採用されている．PCIe を使った高信頼なネットワークシステム技術は，こうした分野で有効な技術となっていくであろう．

6.2.4 命令セットアーキテクチャ

現在，プロセッサのアーキテクチャとしては RISC が主流となっているが，RX プロセッサの開発では，あえて CISC 的な要素を採用することで最適化を図った．MCU の動作仕様を考慮し，動作周波数の向上

という要件を外し，技術の選択を行ったからである．このように，組込みシステムはその使用される環境がさまざまであり，その環境の制約条件を考慮した上で最適化の選択を行うことが，アーキテクチャの設計にとって重要である．

6.2.5　プロセステクノロジの変化

AI におけるニューラルネットの技術がコンピュータの処理能力の変化で状況が変わったように，テクノロジの進化である時からそれまでの状況と質的に変わる場合もある．同様のことは，プロセステクノロジの進化によっても起こると予想できる．実際，設計の手法等も設計ツールの進化と共に変化してきた．過去，トランジスタを人手で設計していた手法は，集積度の増加に伴い合成による設計スタイルに変化していき，また，合成にあったアーキテクチャが採られるように変わって行った．

現時点では，SoC が使用するロジックのプロセステクノロジーは進化し，それによりゲートの実装密度は増えていく．MCU が使用する NVM 混載プロセスの微細化は SoC と比べて遅く，28nm 以降の NVM 混載のプロセスついては，まだどれが本命かは決定していない状況である．増える続けるゲート量を有効に活用するだけの機能が開発されるかは未知の部分があり，MCU などは PAD ネックになるため微細化に対する動機も SoC に比べて弱い．しかし，時期の差は大きくなるかもしれないが，徐々に進むであろう微細化に，アーキテクチャの選択もその時代のテクノロジに合わせて，過去，不利であった技術も含め定量評価を広く行いながら決定していくことが，効率的な設計につながる．

謝辞

学位審査にあたり，ご多忙な中，審査を快くお引き受けくださいました筑波大学　佐藤三久教授，朴泰祐教授，和田耕一教授，安永守利教授，東京大学　塙敏博教授に，深謝いたします．多くの貴重なアドバイスやご意見，ご指導を賜りました．特に主査を引き受けてくださいました　佐藤三久教授には，お忙しい中，何度も時間をつくり研究テーマに関連する事柄や論文の構成などについてご指導いただき終始暖かい励ましを頂きました．また，佐藤三久教授，朴泰祐教授，塙敏博教授とは PEACH プロジェクトで共同開発できたことはよい経験でした．

長期間にわたる研究活動の中，私の研究の方向性と質に影響を与えてくれた偉大な人々と一緒に仕事をすることができ幸運でした．特に，清水徹教授（現東洋大学）には，三菱電機（株）時代から長きに渡り，私の研究活動と博士学位取得への動機付け等，さまざまな面についてご指導及びサポートしていただいたことに感謝いたします．ルネサスエレクトロニクス株式会社の大谷寿賀子博士，中島雅美博士とのマイクロプロセッサアーキテクチャの開発における数々の議論や作業は，私の研究にとって貴重な機会でした．大変感謝しております．また，本研究を遂行するにあたり、様々なご協力を多くの方に頂きました．ルネサスエレクトロニクス株式会社の現上司の西原達也氏をはじめ，元上司の岡山県立大学の有本和民教授，長年の同僚であるルネサスエレクトロニクス株式会社の奥村直人氏，枡井規雄氏，上村稔氏，服部俊洋博士，島崎靖久氏，金子智氏，石川直氏，大槻典正氏，石田一哉氏に感謝したいと思います．

最後に，様々な形でご協力いただいた歴代の職場の同僚や昔からの友人，家族の協力にも感謝いたします．

本研究に関する発表論文

■２章 低電力・高性能なヘテロマルチコア SoC Orochi の設計

国際学会

- S.Kaneko, K.Sawai, N.Masui, K.Ishimi, T.Itou, M.Satou, H.Kondo, N.Okumura, Y.Takata, H.Takata, M.Sakugawa, T.Higuchi, S.Ohtani, K.Sakamoto, N.Ishikawa, M.Nakajima, S.Iwata, K.Hayase, S.Nakano, S.Nakazawa, O.Tomisawa, T.Shimizu：
  A 600MHz Single-Chip Multiprocessor with 4.8G Bytes/sec Internal Shared Pipelined Bus and 512K Byte Internal Memory,
  2003 IEEE International Solid-State Circuits Conference (ISSCC 2003), San Francisco, USA, February (2003), pp.254-255.
- M.Nakajima, H.Kondo, N.Okumura, N.Masui, Y.Takata, T.Nasu, H.Takata, T.Higuchi, M.Sakugawa, H.Yoneda, H.Fujiwara, K.Ishida, K.Ishimi, S.Kaneko, T.Itoh, M.Sato, O.Yamamoto, K.Arimoto：
  Design of a Multi-Core SoC with Configurable Heterogeneous 9CPUs and 2Matrix Processors,
  2007 IEEE Symposium on VLSI Circuits (VLSI Circuit), June (2007), pp.14-15.

論文誌

- H.Kondo, M.Nakajima, M.Bober, K.Kucharski, O.Yamamoto, T.Shimizu：
  Implementation of Face Recognition Processing Using an Embedded Processor,
  Journal of Robotics and Mechatronics, Vol.17, No.4(2005), pp.428-436.
- S.Kaneko, H.Kondo, N.Masui, K.Ishimi, T.Itou, M.Satou, N.Okumura, Y.Takata, H.Takata, M.Sakugawa, T.Higuchi, S.Ohtani, K.Sakamoto, N.Ishikawa, M.Nakajima, S.Iwata, K.Hayase, S.Nakano, S.Nakazawa, K.Yamada, T.Shimizu：
  A 600-MHz Single-Chip Multiprocessor with 4.8-GB/s Internal Shared Pipelined Bus and 512-KB Internal Memory,
  IEEE Journal of Solid-State Circuits, Vol.39, No.1(2004), pp.184-193.
- H.Kondo, M.Nakajima, N.Masui, S.Otani, N.Okumura, Y.Takata, T.Nasu, H.Takata, T.Higuchi, M.Sakugawa, H.Fujiwara, K.Ishida, K.Ishimi, S.Kaneko, T.Itoh, M.Sato, O.Yamamoto, K.Arimoto：
  Design and Implementation of a Configurable Heterogeneous Multi-Core SoC with 9 CPUs and 2 Matrix Processors,

IEEE Journal of Solid-State Circuits, Vol.43, No.4(2008), pp.892-901.

■３章 低電力・高性能なインターコネクト SoC PEACH の設計

国際学会

- S.Otani, H.Kondo, I.Nonomura, A.Ikeya, M.Uemura, K.Asahina, K.Arimoto, S.Miura, T.Hanawa, T.Boku, M.Sato：
  An 80 Gbps dependable multicore communication SoC with PCI express I/F and intelligent interrupt controller,
  2011 IEEE Cool Chips XIV, Yokohama, Japan, April (2011).
- S.Otani, H.Kondo, I.Nonomura, A.Ikeya, M.Uemura, Y.Hayakawa, T.Oshita, S.Kaneko, K.Asahina, K.Arimoto, S.Miura, T.Hanawa, T.Boku, M.Sato：
  An 80Gb/s dependable communication SoC with PCI express I/F and 8 CPUs,
  2011 IEEE International Solid-State Circuits Conference (ISSCC 2011), San Francisco, USA, February (2011), pp.266-267.

論文誌

- S.Otani, H.Kondo, I.Nonomura, T.Hanawa, S.Miura, T.Boku：
  PEACH: A multicore communication system on chip with PCI express,
  IEEE Micro, Vol.31, Issue 6(2011), pp.39-50.

■４章 MCU 向け低電力プロセッサ RX の設計

国際学会

- S.Otani, N.Ishikawa, H.Kondo：
  RXv2 processor core for low-power microcontrollers,
  2013 IEEE COOL Chips XVI, Yokohama, Japan, April (2013).

論文誌

- S.Otani, H.Kondo：
  RX v2: Renesas’ s New-Generation MCU Processor,
  IEICE Transactions on Electronics, Vol.E98-C, No.7(2015), pp.544-549.

■５章 車載 MCU 向け高信頼プロセッサ RH850 の設計

国際学会

- S.Otani, N.Otsuki, Y.Suzuki, N.Okumura, S.Maeda, T.Yanagita, T.Koike, Y.Shimazaki, M.Ito, M.Uemura, T.Hattori, T.Yamauchi, H.Kondo：
  A 28nm 600MHz Automotive Flash Microcontroller with Virtualization-Assisted Processor for Next-Generation Automotive Architecture Complying with ISO26262 ASIL-D,
  2019 IEEE International Solid-State Circuits Conference (ISSCC 2019), San Francisco, USA, February (2019), pp.54-55.

論文誌

- H.Kondo, S.Otani, N.Otsuki, Y.Suzuki, N.Okumura, S.Maeda, T.Yanagita, T.Koike, K.Yayama , Y.Shimazaki, M.Ito, M.Uemura, T.Hattori, N.Sakamoto：
  A 28-nm Automotive Flash Microcontroller With Virtualization-Assisted Processor Supporting ISO26262 ASIL D,
  IEEE Journal of Solid-State Circuits, Vol.55, No.1(2020), pp.133-144.

参考文献

[1] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. Elsevier, 1990.

[2] F. Arakawa, "Chapter1:Low Power Multicore Processors for Embedded Systems," in *Embedded Systems: Hardware, Design, and Implementationmbedded Systems*, K. Iniewski, Ed. Hoboken, NJ, USA: John Wiley & Sons, Inc., Nov. 2012, pp. 1–60.

[3] "RX ファミリ RXv3 命令セットアーキテクチャユーザーズマニュアル ソフトウェア編," https://www.renesas.com/jp/ja/document/man/906216?language=ja.

[4] T. Kono, Y. Taito, and H. Hidaka, "Essential Roles, Challenges and Development of Embedded MCU Micro-Systems to Innovate Edge Computing for the IoT/AI Age," *IEICE Transactions on Electronics*, vol. E103.C, no. 4, pp. 132–143, 2020.

[5] "The Wild West Of Automotive Semiconductor Engineering [Online]," Apr. 2015.

[6] D. Reinhardt and M. Kucera, "Domain controlled architecture," in *Proc. Third International Conference on Pervasive and Embedded Computing and Communication Systems (PECCS 2013)*, 2013.

[7] "ワイヤーハーネス　自動車機器 [Online]," https://www.yazaki-group.com/wireharness/.

[8] T. Hattori, T. Irita, M. Ito, E. Yamamoto, H. Kato, Go Sado, T. Yamada, K. Nishiyama, H. Yagi, T. Koike, Y. Tsuchihashi, M. Higashida, H. Asano, I. Hayashibara, K. Tatezawa, Y. Shimazaki, N. Morino, K. Hirose, S. Tamaki, S. Yoshioka, R. Tsuchihashi, N. Arai, T. Akiyama, and K. Ohno, "A Power Management Scheme Controlling 20 Power Domains for a Single-Chip Mobile Processor," in *2006 IEEE International Solid State Circuits Conference - Digest of Technical Papers*, Feb. 2006, pp. 2210–2219.

[9] M. Ito, T. Hattori, Y. Yoshida, K. Hayase, T. Hayashi, O. Nishii, Y. Yasu, A. Hasegawa, M. Takada, M. Ito, H. Mizuno, K. Uchiyama, T. Odaka, J. Shirako, M. Mase, K. Kimura, and H. Kasahara, "An 8640 MIPS SoC with Independent Power-Off Control of 8 CPUs and 8 RAMs by An Automatic Parallelizing Compiler," in *2008 IEEE International Solid-State Circuits Conference - Digest of Technical Papers*, Feb. 2008, pp. 90–598.

[10] J. Howard, S. Dighe, Y. Hoskote, S. Vangal, D. Finan, G. Ruhl, D. Jenkins, H. Wilson, N. Borkar, G. Schrom, F. Pailet, S. Jain, T. Jacob, S. Yada, S. Marella, P. Salihundam, V. Erraguntla, M. Konow, M. Riepen, G. Droege, J. Lindemann, M. Gries, T. Apel, K. Henriss, T. Lund-Larsen, S. Steibl, S. Borkar, V. De, R. V. D. Wijngaart, and T. Mattson, "A 48-Core IA-32 message-passing processor with DVFS in 45nm CMOS," in *2010 IEEE International Solid-State Circuits Conference - (ISSCC)*, Feb. 2010, pp. 108–109.

[11] H. Mair, A. Wang, G. Gammie, D. Scott, P. Royannez, S. Gururajarao, M. Chau, R. Lagerquist, L. Ho, M. Basude, N. Culp, A. Sadate, D. Wilson, F. Dahan, J. Song, B. Carlson, and U. Ko, "A 65-nm Mobile Multimedia Applications Processor with an Adaptive Power Management Scheme to Compensate for Variations," in *2007 IEEE Symposium on VLSI Circuits*, Jun. 2007, pp. 224–225.

[12] N. Rohrer, C. Akrout, M. Canada, D. Cawthron, B. Davari, R. Floyd, S. Geissler, R. Goldblatt, R. Houle, P. Kartschoke, D. Kramer, P. McCormick, G. Salem, R. Schulz, L. Su, and L. Whitney, "A 480 MHz RISC microprocessor in a 0.12 /spl mu/m L/sub eff/ CMOS technology with copper interconnects," in *1998 IEEE International Solid-State Circuits Conference. Digest of Technical Papers, ISSCC. First Edition (Cat. No.98CH36156)*, Feb. 1998, pp. 240–241.

[13] H. Mizuno, K. Ishibashi, T. Shimura, T. Hattori, S. Narita, K. Shiozawa, S. Ikeda, and K. Uchiyama, "An 18-/spl mu/A standby current 1.8-V, 200-MHz microprocessor with self-substrate-biased data-retention mode," *IEEE Journal of Solid-State Circuits*, vol. 34, no. 11, pp. 1492–1500, Nov. 1999.

[14] Y. Kanno, H. Mizuno, Y. Yasu, K. Hirose, Y. Shimazaki, T. Hoshi, Y. Miyairi, T. Ishii, T. Yamada, T. Irita, T. Hattori, K. Yanagisawa, and N. Irie, "Hierarchical Power Distribution With Power Tree in Dozens of Power Domains for 90-nm Low-Power Multi-CPU SoCs," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 1, pp. 74–83, Jan. 2007.

[15] Y. Kanno, H. Mizuno, N. Oodaira, Y. Yasu, and K. Yanagisawa, "/spl mu/I/O architecture for 0.13-/spl mu/m wide-voltage-range system-on-a-package (SoP) designs," in *2002 Symposium on VLSI Circuits. Digest of Technical Papers (Cat. No.02CH37302)*, Jun. 2002, pp. 168–169.

[16] International Organization for Standardization, "ISO 26262:Road vehicles - Functional safety [Online]," https://www.iso.org/standard/43464.html.

[17] D. Pham, S. Asano, M. Bolliger, M. Day, H. Hofstee, C. Johns, J. Kahle, A. Kameyama, J. Keaty, Y. Masubuchi, M. Riley, D. Shippy, D. Stasiak, M. Suzuoki, M. Wang, J. Warnock, S. Weitzel, D. Wendel, T. Yamazaki, and K. Yazawa, "The design and implementation of a first-generation CELL processor," in *ISSCC. 2005 IEEE International Digest of Technical Papers. Solid-State Circuits Conference, 2005.*, Feb. 2005, pp. 184–592 Vol. 1.

[18] S. Rusu, S. Tam, H. Muljono, D. Ayers, and J. Chang, "A dual-core multi-threaded Xeon processor with 16MB L3 cache," in *2006 IEEE International Solid State Circuits Conference-Digest of Technical Papers*. IEEE, 2006, pp. 315–324.

[19] A. S. Leon, K. W. Tam, J. L. Shin, D. Weisner, and F. Schumacher, "A Power-Efficient High-Throughput 32-Thread SPARC Processor," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 1, pp. 7–16, Jan. 2007.

[20] H. Noda, M. Nakajima, K. Dosaka, K. Nakata, M. Higashida, O. Yamamoto, K. Mizumoto, T. Tanizaki, T. Gyohten, Y. Okuno, H. Kondo, Y. Shimazu, K. Arimoto, K. Saito, and T. Shimizu, "The Design and Implementation of the Massively Parallel Processor Based on the Matrix Architecture," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 1, pp. 183–192, Jan. 2007.

[21] S. Kyo, T. Koga, S. Okazaki, and I. Kuroda, "A 51.2-GOPS scalable video recognition processor for intelligent cruise control based on a linear array of 128 four-way VLIW processing elements," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 11, pp. 1992–2000, Nov. 2003.

[22] W. D. Hillis, *The Connection Machine*. MIT Press, 1989.

[23] R. J. Swan, A. Bechtolsheim, K.-W. Lai, and J. K. Ousterhout, "The implementation of the CM multimicroprocessor," in *Proceedings of the June 13-16, 1977, National Computer Conference*, ser. AFIPS '77. New York, NY, USA: Association for Computing Machinery, Jun. 1977, pp. 645–655.

[24] 天野英晴, **並列コンピュータ**. 昭光堂, 1996.

[25] N. Uchida, T. Shibahara, T. Aoki, H. Nakajima, and K. Kobayashi, "3D face recognition using passive stereo vision," in *IEEE International Conference on Image Processing 2005*, vol. 2, Sep. 2005, pp. II–950.

[26] S. Kaneko, H. Kondo, N. Masui, K. Ishimi, T. Itou, M. Satou, N. Okumura, Y. Takata, H. Takata, M. Sakugawa, T. Higuchi, S. Ohtani, K. Sakamoto, N. Ishikawa, M. Nakajima, S. Iwata, K. Hayase, S. Nakano, S. Nakazawa, K. Yamada, and T. Shimizu, "A 600-MHz single-chip multiprocessor with 4.8-GB/s internal shared pipelined bus and 512-kB internal memory," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 1, pp. 184–193, Jan. 2004.

[27] H. Kondo, M. Nakajima, M. Bober, K. Kucharski, O. Yamamoto, and T. Shimizu, "Implementation of Face Recognition Processing Using an Embedded Processor," *Journal of Robotics and Mechatronics*, vol. 17, no. 4, pp. 428–436, Aug. 2005.

[28] T. Shimizu, J. Korematu, M. Satou, H. Kondo, S. Iwata, K. Sawai, N. Okumura, K. Ishimi, Y. Nakamoto, M. Kumanoya, K. Dosaka, A. Yamazaki, Y. Ajioka, H. Tsubota, Y. Nunomura, T. Urabe, J. Hinata, and K. Saitoh, "A multimedia 32 b RISC microprocessor with 16 Mb DRAM," in *1996 IEEE International Solid-State Circuits Conference. Digest of TEchnical Papers, ISSCC*, Feb. 1996, pp. 216–217.

[29] 遠藤 幸典, 菅井 尚人, and 近藤 弘郁, "シングルチップマルチプロセッサ上のハイブリッド OS 環境の実現-システムアーキテクチャ-," **第 *66* 回全国大会講演論文集**, vol. 2004, no. 1, pp. 9–10, Mar. 2004.

[30] 菅井 尚人, 近藤弘郁, and 落合 真一, "組込み向けチップマルチプロセッサ上の複数 OS 実行基盤," **情報処理学会研究報告計算機アーキテクチャ（*ARC*）**, vol. 2007, no. 4(2007-ARC-171), pp. 43–48, Jan. 2007.

[31] P. Vuillod, L. Benini, A. Bogliolo, and G. De Micheli, "Clock-skew optimization for peak current reduction," in *Proceedings of 1996 International Symposium on Low Power Electronics and Design*, Aug. 1996, pp. 265–270.

[32] 安藤 秀樹, "新世代マイクロプロセッサアーキテクチャ（前編）:1. アーキテクチャ基盤技術 5. チップ・マルチプロセッサ," **情報処理**, vol. 46, no. 10, pp. 1124–1130, Oct. 2005.

[33] M. T. J. Strik, A. H. Timmer, J. L. van Meerbergen, and G. van Rootselaar, "Heterogeneous multiprocessor for the management of real-time video and graphics streams," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 11, pp. 1722–1731, Nov. 2000.

[34] T. Koyama, K. Inoue, H. Hanaki, M. Yasue, and E. Iwata, "A 250-MHz single-chip multiprocessor for audio and video signal processing," *IEEE Journal of Solid-State Circuits*, vol. 36, no. 11, pp. 1768–1774, Nov. 2001.

[35] N. Nishi, T. Inoue, M. Nomura, S. Matsushita, S. Torii, A. Shibayama, J. Sakai, T. Ohsawa, Y. Nakamura, S. Shimada, Y. Ito, M. Edahiro, M. Mizuno, K. Minami, O. Matsuo, H. Inoue, T. Manabe, T. Ya-

mazaki, Y. Nakazawa, Y. Hirota, Y. Yamada, N. Onoda, H. Kobinata, M. Ikeda, K. Kazama, A. Ono, T. Horiuchi, M. Motomura, M. Yamashina, and M. Fukuma, "A 1 GIPS 1 W single-chip tightly-coupled four-way multiprocessor with architecture support for multiple control flow execution," in *2000 IEEE International Solid-State Circuits Conference. Digest of Technical Papers (Cat. No.00CH37056)*, Feb. 2000, pp. 418–419.

[36] L. Hammond, B. A. Hubbert, M. Siu, M. K. Prabhu, M. Chen, and K. Olukolun, "The Stanford Hydra CMP," *IEEE Micro*, vol. 20, no. 2, pp. 71–84, Mar. 2000.

[37] K. Diefendorff, "Power4 focuses on memory bandwidth," *Microprocessor Report*, vol. 13, no. 13, pp. 1–8, 1999.

[38] K. Hirata and J. Goodacre, "ARM MPCore; The streamlined and scalable ARM11 processor core," in *2007 Asia and South Pacific Design Automation Conference*, Jan. 2007, pp. 747–748.

[39] InfiniBand Trade Association, "The Infiniband architecture specification [Online]," https://www.infinibandta.org/.

[40] T. Hanawa, T. Boku, S. Miura, M. Sato, and K. Arimoto, "PEARL: Power-Aware, Dependable, and High-Performance Communication Link Using PCI Express," in *2010 IEEE/ACM Int'l Conference on Green Computing and Communications Int'l Conference on Cyber, Physical and Social Computing*, Dec. 2010, pp. 284–291.

[41] PCI-SIG, "PCI Express Base Specification [Online]," https://pcisig.com/.

[42] S. Otani, H. Kondo, I. Nonomura, A. Ikeya, M. Uemura, Y. Hayakawa, T. Oshita, S. Kaneko, K. Asahina, K. Arimoto, S. Miura, T. Hanawa, T. Boku, and M. Sato, "An 80Gb/s dependable communication SoC with PCI express I/F and 8 CPUs," in *2011 IEEE International Solid-State Circuits Conference*, Feb. 2011, pp. 266–268.

[43] S. Otani, H. Kondo, I. Nonomura, A. Ikeya, M. Uemura, K. Asahina, K. Arimoto, S. Miura, T. Hanawa, T. Boku, and M. Sato, "An 80 Gbps dependable multicore communication SoC with PCI express I/F and intelligent interrupt controller," in *2011 IEEE Cool Chips XIV*, Apr. 2011, pp. 1–3.

[44] S. Otani, H. Kondo, I. Nonomura, T. Hanawa, S. Miura, and T. Boku, "Peach: A Multicore Communication System on Chip with PCI Express," *IEEE Micro*, vol. 31, no. 6, pp. 39–50, Nov. 2011.

[45] H. Kondo, M. Nakajima, N. Masui, S. Otani, N. Okumura, Y. Takata, T. Nasu, H. Takata, T. Higuchi, M. Sakugawa, H. Fujiwara, K. Ishida, K. Ishimi, S. Kaneko, T. Itoh, M. Sato, O. Yamamoto, and K. Arimoto, "Design and Implementation of a Configurable Heterogeneous Multicore SoC With Nine CPUs and Two Matrix Processors," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 4, pp. 892–901, Apr. 2008.

[46] T. Okamoto, S. Miura, T. Boku, M. Sato, and D. Takahashi, "RI2N/UDP: High bandwidth and fault-tolerant network for a PC-cluster based on multi-link Ethernet," in *2007 IEEE International Parallel and Distributed Processing Symposium*, Mar. 2007, pp. 1–8.

[47] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach 4th Edition*. Elsevier, 2007.

[48] Barry Wilkinson, **計算機設計技法 マルチプロセッサシステム論**. Prentice Hall, 1994.

[49] T. Kono, T. Ito, T. Tsuruda, T. Nishiyama, T. Nagasawa, T. Ogawa, Y. Kawashima, H. Hidaka, and T. Yamauchi, "40-nm Embedded Split-Gate MONOS (SG-MONOS) Flash Macros for Automotive With 160-MHz Random Access for Code and Endurance Over 10 M Cycles for Data at the Junction Temperature of 170 \circ C," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 1, pp. 154–166, Jan. 2014.

[50] S. Gary, C. Dietz, J. Eno, G. Gerosa, Sung Park, and H. Sanchez, "The PowerPC 603 microprocessor: A low-power design for portable applications," in *Proceedings of COMPCON '94*, Feb. 1994, pp. 307–315.

[51] Y. Sugure, "A Very-Low-Latency Superscalar Microcontroller for Automotive, Industrial, and PC-Peripheral Applications," *COOL Chips VII, Apr. 2004*, 2004.

[52] T. R. Halfhill, "EEMBC Releases First Benchmarks," *Microprocessor Report: The Linley Group*, May 2000.

[53] J. A. Poovey, T. M. Conte, M. Levy, and S. Gal-On, "A BENCHMARK CHARACTERIZATION OF THE EEMBC BENCHMARK SUITE," *IEEE MICRO*, p. 12, 2009.

[54] EEMBC, "Coremark an EEMBC Benchmark [Online]," https://www.eembc.org/coremark/.

[55] T. R. Halfhill, "EEMBC's Dhrystone Killer," *Microprocessor Report: The Linley Group*, Jun. 2009.

[56] L. Benini and M. Poncino, "Selective Instruction Compression for Memory Energy Reduction in Embedded Systems," p. 6.

[57] A. Gordon-Ross, S. Cotterell, and F. Vahid, "Exploiting Fixed Programs in Embedded Systems: A Loop Cache Example," *IEEE Computer Architecture Letters*, vol. 1, no. 1, pp. 2–2, Jan. 2002.

[58] S. Cotterell and F. Vahid, "Tuning of Loop Cache Architectures to Programs in Embedded System Design," p. 6.

[59] J. Kin, Munish Gupta, and W. Mangione-Smith, "The filter cache: An energy efficient memory structure," in *Proceedings of 30th Annual International Symposium on Microarchitecture*. Research Triangle Park, NC, USA: IEEE Comput. Soc, 1997, pp. 184–193.

[60] A. Gordon-Ross and F. Vahid, "Dynamic loop caching meets preloaded loop caching-a hybrid approach," in *Proceedings. IEEE International Conference on Computer Design: VLSI in Computers and Processors*. Freiberg, Germany: IEEE Comput. Soc, 2002, pp. 446–449.

[61] Chi-Hung Chi and Jun-Li Yuan, "Load-balancing branch target cache and prefetch buffer," in *Proceedings 1999 IEEE International Conference on Computer Design: VLSI in Computers and Processors (Cat. No.99CB37040)*. Austin, TX, USA: IEEE Comput. Soc, 1999, pp. 436–441.

[62] 大町 一彦, "仮想マシン道しるべ [1] : 仮想マシン草創期," **情報処理**, vol. 48, no. 8, pp. 903–905, Aug. 2007.

[63] H. Kondo, O. Yamamoto, S. Otani, N. Sugai, and T. Shimizu, "Software architecture of a secure multimedia system using a multicore SoC and software virtualization," in *2009 Digest of Technical Papers International Conference on Consumer Electronics*, Jan. 2009, pp. 1–2.

[64] H. Kondo, S. Otani, M. Nakajima, O. Yamamoto, N. Masui, N. Okumura, M. Sakugawa, M. Kitao, K. Ishimi, M. Sato, F. Fukuzawa, S. Imasu, N. Kinoshita, Y. Ota, K. Arimoto, and T. Shimizu, "Heterogeneous Multicore SoC With SiP for Secure Multimedia Applications," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 8, pp. 2251–2259, Aug. 2009.

[65] A. K. Sundar Rajan, A. Feucht, L. Gamer, I. Smaili, and N. D. M., "Hypervisor for consolidating real-time automotive control units: Its procedure, implications and hidden pitfalls," *Journal of Systems Architecture*, vol. 82, pp. 37–48, Jan. 2018.

[66] D. Reinhardt and G. Morgan, "An embedded hypervisor for safety-relevant automotive E/E-systems," in *Proceedings of the 9th IEEE International Symposium on Industrial Embedded Systems (SIES 2014)*, Jun. 2014, pp. 189–198.

[67] S. Shibahara, C. Takahashi, K. Fukuoka, Y. Kitaji, T. Irita, H. Hara, Y. Shimazaki, and J. Matsushima, "A 16 nm FinFET Heterogeneous Nona-Core SoC Supporting ISO26262 ASIL B Standard," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 77–88, Jan. 2017.

[68] A. Floridia and E. Sanchez, "Hybrid On-Line Self-Test Strategy for Dual-Core Lockstep Processors," in *2018 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, Oct. 2018, pp. 1–6.

[69] G. Zhang, K. Yayama, A. Katsushima, and T. Miki, "A 3.2 ppm/°C Second-Order Temperature Compensated CMOS On-Chip Oscillator Using Voltage Ratio Adjusting Technique," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 4, pp. 1184–1191, Apr. 2018.

[70] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P.-l. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, and D. H. Yoon, "In-Datacenter Performance Analysis of a Tensor Processing Unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ser. ISCA '17. Toronto, ON, Canada: Association for Computing Machinery, Jun. 2017, pp. 1–12.

[71] "RH850/P1H-C: High-end Automotive Microcomputers for Chassis Control [Online]," https://www.renesas.com/us/en/products/microcontrollers-microprocessors/rh850-automotive-mcus/rh850p1h-c-high-end-automotive-microcomputers-chassis-control.

[72] S. Otani, N. Otsuki, Y. Suzuki, N. Okumura, S. Maeda, T. Yanagita, T. Koike, Y. Shimazaki, M. Ito, M. Uemura, T. Hattori, T. Yamauchi, and H. Kondo, "2.7 A 28nm 600MHz Automotive Flash Microcontroller with Virtualization-Assisted Processor for Next-Generation Automotive Architecture Complying with ISO26262 ASIL-D," in *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*, Feb. 2019, pp. 54–56.

[73] "32-bit AURIX™ Microcontroller based on TriCore [Online]," https://www.infineon.com/cms/en/product/microcontroller/32-bit-tricore-microcontroller/.

[74] "E200z7 AutoBench Production Silicon Benchmark Scores [Online]," https://www.eembc.org/viewer/?benchmark_seq=1198&certification_type=OUT.

[75] “EEMBC : Embedded Microprocessor Benchmark Consortium [Online],” https://www.eembc.org/autobench/.

[76] Y. Taito, T. Kono, M. Nakano, T. Saito, T. Ito, K. Noguchi, H. Hidaka, and T. Yamauchi, “A 28 nm Embedded Split-Gate MONOS (SG-MONOS) Flash Macro for Automotive Achieving 6.4 GB/s Read Throughput by 200 MHz No-Wait Read Operation and 2.0 MB/s Write Throughput at Tj of 170° C,” *IEEE Journal of Solid-State Circuits*, vol. 51, no. 1, pp. 213–221, Jan. 2016.

[77] A. Kanda, T. Kurafuji, K. Takeda, T. Ogawa, Y. Taito, K. Yoshihara, M. Nakano, T. Ito, H. Kondo, and T. Kono, “A 24-MB Embedded Flash System Based on 28-nm SG-MONOS Featuring 240-MHz Read Operations and Robust Over-the-Air Software Update for Automotive Applications,” *IEEE Solid-State Circuits Letters*, vol. 2, no. 12, pp. 273–276, Dec. 2019.

[78] 菅井 尚人, 遠藤 幸典, and 近藤 弘郁, “シングルチップマルチプロセッサ上のハイブリッド OS 環境の実現-OS 間インタフェースの実装-,” **第 *66* 回全国大会講演論文集**, vol. 2004, no. 1, pp. 11–12, Mar. 2004.

[79] 茂田井 寛隆, 山本 整, 落合 真一, 安達 浩次, 鈴木 均, 城倉 梨香, 福井 昭也, 小川 敏行, 田原 康宏, and 奥村 直人, “組込み向け CPU 仮想化技術対応ハイパーバイザの設計,” **組込みシステムシンポジウム *2012* 論文集**, vol. 2012, pp. 95–100, Oct. 2012.

[80] 大山 恵弘, “仮想マシン道しるべ [4] : 仮想マシン技術の応用,” **情報処理**, vol. 48, no. 11, pp. 1283–1287, Nov. 2007.