平成 29 年度 博士論文

センサ向けデバイスコンピューティング アーキテクチャの研究

The study of device computing architecture for sensor applications

指導教員　岩崎 晃 教授

東京大学大学院 工学系研究科 先端学際工学専攻

37-147291

HIHARA, Hiroki

檜原 弘樹

概要

社会インフラシステムの一翼を担う Internet of Things (IoT)アプリケーションでは多数のセンサがフィールドに配置される。これらのセンサからは大量のデータが出力され、既設の通信回線では全てのデータを送信しきれなくなってきている。IoT のセンサノードとして位置付けられ、各種のセンサを搭載する人工衛星の例においても、搭載しているセンサから出力されるデータは人工衛星と地上局間の通信回線の容量を超えるようになってきている。このため、センサから出力するデータを校正して有意なものを抽出する、あるいは圧縮処理などによる伝送容量を削減する機能が不可欠になっている。このような処理を行うに際しては、組込み用マイクロコントローラ（Micro-controller Unit: MCU）や Field Programmable Gate Array（FPGA）を用いたプロセッサが用いられることが多く、これらのプロセッサの小型化、および低消費電力化が課題となってきている。このため近年では半導体のみならず金属の応用も試みられ、金属イオンの架橋を動的に制御する原子スイッチ、ないしはナノブリッジと呼ばれる機構を利用したデバイスも提案されている。また、演算処理を半導体集積回路に実装するソフトウェア技術としてセンサ信号を処理するアルゴリズムを直接回路に落とし込むための動作合成技術や、それらのアルゴリズムを高級言語で記述することを可能とする高位合成技術も実用化されている。これらの新しいデバイス技術やソフトウェア技術を統合し、センシングデバイス自体に演算処理を埋め込むことができれば、多数のセンサを屋内・屋外を問わず配置し、既設の通信回線を活用しつつ、十分な量のデータを収集し、解析することを可能とする IoT システムが構築できるようになる。本研究では、IoT に用いられるセンシングデバイスに信号処理機能を埋め込むにあたり、これらのデバイスやソフトウェア技術を効果的に活用して小型化、低消費電力化、および高信頼性化を達成するアーキテクチャ、および設計手法を提案した。さらに、提案するアーキテクチャと設計手法を実装し、評価した結果、目的としたセンシングデバイスへ埋め込むための小型、低消費電力、および高信頼性を実現できることを確認した。本論文ではその成果を報告すると共に、さらなる応用の可能性として得られた知見についても述べる。

本研究により実現した技術は、具体的には、新しく登場したナノブリッジ技術を活用するために、新しい Processigng Element (PE)のアーキテクチャと、階層化の概念を用いた新しい設計手法を実現するものである。これらを実装し、評価した結果、IoT で多用されるセンサに組込むことのできる小型・低消費電力で、かつ高信頼性を有するプログラマブルデバイスを実現できることを確認した。技術的には現在の技術で実現されている 3 種類のプログラマブルデバイスである MCU, FPGA, および動的再構成プロセッサ（Dynamically Reconfigurable Processor: DRP）の利点を、高位合成技術を活用して統合したものである。

これら 3 種類のプログラマブルデバイスを統合するに当たっては、克服すべき課題を明確化すべく、数理モデルの見直しにまでさかのぼった。IoT のセンサに組込むプロセッサが参照すべき数理モデルを策定するに当たっては、外付けメモリ素子を不要とするプロセッサのアーキテクチャを考案し、計算モデルの見直しを行った。このように新しい数理モデルを定義した結果、克服すべき課題が明らかとなり、これらの課題を克服すべく、PE のアーキテクチャと設計手法を新たに提案した。識別した課題は以下の二点である。

・外付けメモリ素子を要しないプロセッサエレメントのアーキテクチャの創出

・外付けメモリ素子を要しないプロセッサエレメントの設計手法の確立

これらについて具体的には以下の提案と実装評価を行った。

1)　外付けメモリ素子を要しないプロセッサエレメントのアーキテクチャの創出

MCU などのストアードプログラム方式のプロセッサエレメントは演算器と外付けメモリ素子が別々に実装されている。このため、演算器と外付けメモリ素子との間のデータの送受信の効率化が性能向上の鍵であると共に、これがボトルネックとなって実装面積の増大と消費電力の増加を招いている。実装面積の小型化と低消費電力化を目指して演算器と外付けメモリ素子間のデータの送受信の効率を上げる研究は種々行われている。これに対して、外付けメモリ素子を不要とし、演算器と外付けメモリ素子間のデータの転送自体を不要とできれば効果的であることは、専用のシステム LSI などと比較しても明らかである。IoT に用いられるセンシングデバイスに演算器を埋め込むに当たってはこのような小型化と低消費電力化が不可欠である。この外付けメモリ素子を不要とするプロセッサエレメントのアーキテクチャを確立すべく、関数演算器化の概念と整合性のとれた粒度別プロセッサエレメントのアーキテクチャを提案し、信頼性を高めるべく提案した分散化高信頼性アーキテクチャと統合することにより、ジェネリックなプロセッサエレメントのアーキテクチャを提案した。実際にこの Generic Processing Element (GPE)を実現するに当たっては、近年実用的になってきた高位合成技術を活用し、高位合成ツールの機能拡張としての関数演算器化機能を活用すべく、必要な要素と結合方法を検討した。

2)　外付けメモリ素子を要しないプロセッサエレメントの設計手法の確立

MCU などのストアードプログラム方式のプロセッサとシステム LSI は一般に対極的に論じられる。すなわち、最も小型で低消費電力を実現するが、開発コストが大きく開発期間が長い専用のシステム LSI と、開発期間が短く、開発コストも安価であるが、サイズと消費電力はシステム LSI に叶わない MCU などのプロセッサ、という構図である。更に、この中間的な存在として FPGA を位置付けるという説明も広く行われている。ナノブリッジ技術の実用化により、LSI 内の金属配線がプログラマブルになったことを活かすためには、この比較指標自体を見直す必要性を識別した。具体的には、評価指標を見直すために、演算処理機能を再整理して粒度別階層化演算モデルを提起し、演算器の構成方法を見直した。

この結果、粒度別階層化演算モデルを参照してプロセッサエレメントを設計するために上述の Generic Processing Element (GPE)の概念に加え、粗粒度の関数を演算器に対応付ける粗粒度バインディング層の必要性が明らかとなり、これを高位合成ツールの機能と対応させるべく、高位合成／動作合成ツールである CyberWorkBench (CWB)の拡張機能である関数演算器化機能に対応させた。

上述した粒度別プロセッサエレメントの概念、関数演算器化機能、および分散化高信頼性アーキテクチャを統合した Generic Processing Element アーキテクチャと、機能拡張を施した高位合成／動作合成ツールである CWB を用いた粒度別プロセッサエレメントの設計手法を具体化すべく、人工衛星搭載用の赤外線センサをモチーフとした試作評価を行った結果、所期の小型化と低消費電力化が実現できることを確認した。また、Wire rate processing とよばれる高速信号処理が達成できたことも確認でき、センシングデバイスに埋め込むことのできるプロセッサエレメントの要件である小型化・低消費電力化・高信頼性化の実現の目途を得て有効性を実証した。これにより IoT 向けデバイスコンピューティングアーキテクチャを確立できたものと考える。これらの結果について本論文に詳細を述べた。

また、これらの実装評価の結果、他分野への波及効果も得られている。具体的には、この拡張した高位合成／動作合成ツールである CWB を用いて畳み込み演算を対象とした詳細解析を行った結果、ナノブリッジを用いた FPGA のみならず、従来の方式による FPGA でも小型化の効果が顕著に認められたと共に、システム LSI の設計にも活用できる目途が得られた。

以上の成果について、提案する粒度別 Processing Element (PE)アーキテクチャ、および分散化高信頼性アーキテクチャは国内外で特許が認められており、本研究によるオリジナルのアーキテクチャである。本研究では提案する粒度別 PE アーキテクチャに高位合成技術を拡張した CyberWorkBench を組み合わせ、Generic な PE アーキテクチャと粒度別 PE 設計手法として纏めた。このアーキテクチャと設計手法に本研究で提案する分散化高信頼性アーキテクチャを融合し、ナノブリッジ FPGA を小型化・高信頼性化する粒度別高信頼性化設計手法を確立した。更に、DRP の技術をこの粒度別高信頼性化設計手法により改善することで小型化・低消費電力化・高信頼性化（分散化）の目途を得、センシングデバイスに埋め込める小型化・低消費電力化・高信頼性化の実現性の目途を得た。この研究成果の今後の研究開発展望について、システム LSI の技術トレンドである Makimoto’s Wave に基づいて論じると共に、ニューロモルフィック等のアナログ技術を活用した応用拡大などについても論じた。

図一覧

表一覧

目次

第1章
研究の背景

はじめに、本研究の背景となった新しいデバイスとの登場と、ネットワーク社会を背景とした新しい技術要求について述べる。半導体を用いたセンサやアクチュエータは近年広範囲に利用されている。半導体の集積度向上の動機付けとなってきたムーアの法則［1］の限界が予見されるようになり、プロセッサを構成するデバイスとして半導体のみならず金属の応用も試みられるようになってきた。この一つとして金属イオンの架橋を動的に制御する原子スイッチ、ないしはナノブリッジと呼ばれる機構を利用したデバイスが提案され、実用化の域に達している［2, 3, 4］。このデバイスの登場により、半導体センサを用いた機器の消費電力を顕著に削減できることが期待されている。一方、センサを繋いだネットワークは社会インフラシステムの一翼を担うようになり、Internet of Things (IoT)アプリケーションでは多数のセンサがフィールドに配置される［5］。これらのセンサからは大量のデータが出力される一方で、全てのデータを既設の通信回線では送信しきれないという課題が顕在化してきた。IoT のセンサノードとして位置付けられ、各種のセンサを搭載する人工衛星の例［6, 7］においても、搭載しているセンサから出力されるデータは人工衛星と地球地上局間の通信回線の容量を超えているものが少なくない。このため、センサから出力するデータを校正あるいは選択して有意なものを抽出する、あるいは圧縮して伝送容量を削減するといった演算機能をセンサに内蔵する必要性が増している。上述のナノブリッジを用いたデバイスを使いこなすことができれば、低消費電力を活かしつつ演算機能をセンサに埋め込めるようになることが期待される。このようにナノブリッジを用いたデバイスを演算素子として使いこなすことのできる枠組みが求められるようになった。これらについて以下に述べる。

1.1. 新しいデバイス：ナノブリッジ

センサから出力される信号を処理するに当たっては、組込みシステム用途として汎用的に用いられる Micro-Controller Unit (MCU)や、専用の信号処理回路を実装したシステム LSI (Large Scale Integration: 大規模集積回路)等が用いられる。システム LSI として最も広く使われているのは CBIC (Cell Based Integrated Circuit)である。CBIC は標準的な回路のセルで構成され、回路の接続はあらかじめ製造時に行われるため、その後に回路を変更することはできない。CBIC は一旦開発が完了すれば安価に量産ができる反面、開発に際して多額のコストと開発期間を要するため、回路を書き換えられるようにした FPGA (Field Programmable Gate Array)に代表される再構成可能 LSI が使われるようになってきた。再

構成可能 LSI は処理内容をプログラムできるロジックセルと配線、およびプログラムするための多数のスイッチから構成される。このスイッチはメモリとパストランジスタから構成され、このスイッチだけでチップ面積の半分以上を占めることも珍しくない。このスイッチの使用数を減らすために、高機能でサイズが大きなロジックセルが用いられるケースも増えているが、高機能化するにつれて消費電力の増加などの副作用を招く。このため、スイッチ自体を小型化する開発が進められており、その中の一つとして、スイッチを半導体ではなく金属で構成する手法の一つとしてナノブリッジが開発されている [2, 3, 4]。

簡潔な 2 端子のナノブリッジのスイッチング動作を図 1-1 に示す。ナノブリッジは 2 種類の金属原子（Cu および Ru）に挟まれた高分子固体電解質（Polymer Solid Electrolyte: PSE）を有する原子スイッチから構成されている。このスイッチは、PSE 膜の両端にバイアス電圧を印加することによりナノメートルスケールの金属ブリッジが現れるかないしは消滅し、これらに対応し、それぞれスイッチはオンまたはオフになる。図 1-1 に示す Cu 電極に正の電圧を印加すると、Cu 電極から Cu +イオンが PSE に供給される。Cu +イオンは Ru 電極で中和され、析出する。続いて析出した Cu は、2 つの電極の間に導電性ブリッジを形成し、したがって電気抵抗の低い ON 状態に変化させる。逆に、Cu 電極に負電圧を印加することにより、Cu ブリッジがイオン化して消滅し、オフ状態となる。各状態は不揮発性であり一旦オンすると所定の負電圧を掛けるまで安定してオン状態を保ち、一旦オフすれば、所定の正電圧を掛けるまで安定してオフ状態を保つ。2 つの状態の切り替えは繰り返し可能であり、マイクロ秒単位の応答性能を有することが確認されている [2]。ナノブリッジは通常の CMOS（Complementary metal oxide semiconductor: 相補型金属酸化膜半導体）下地の上に形成することができるため、ナノブリッジを使用した FPGA は従来の安定した技術に基づく信頼性の高いプロセスで製造することができる。FPGA に用いられる Look Up Table（LUT）の一連のオン／オフ状態をナノブリッジで実装することにより、高いオフ状態インピーダンスを持つスイッチが実現でき、信頼度の高い FPGA を構成することができる。

この 2 端子ナノブリッジにはいくつかの課題があったが、次のように解決され、実用的なデバイスが開発されるに至っている。まず、オフからオン状態へのスイッチング電圧は 0.2V 程度とロジックの信号電圧(1V 以上)よりも低い。ロジックの信号によってスイッチの抵抗状態が変化しないように、スイッチング電圧はロジック電圧よりも高い必要がある。この問題は、低イオン伝導度を備えた固体電解質を用いることによって解決している。また、スイッチング時の電流が大きいことも課題となる。このために消費電力が大きくなり、素子の熱破壊等が起こる懸念が生じるためである。オン抵抗を保ちながらスイッチング電圧を高めてしまうと、さらにスイッチング時の電流が増大してしまう。これを解決するために相補型原子スイッチ（Complementary Atom Switch: CAS）と呼ぶ 3 端子のナノブリッジが開発された [4]。CAS の構成を図 1-2 に示す。ナノブリッジの基本的な構成は、この CAS である。オン抵抗は 1kΩ程度であり、オフ電流は 10nA 以下である。2 つの Cu 電

極から成る CAS のスイッチング回数は最大 1,000 回まで信頼性が確認されており、各状態は不揮発性である。本信頼性評価結果は実用に供するに十分である。

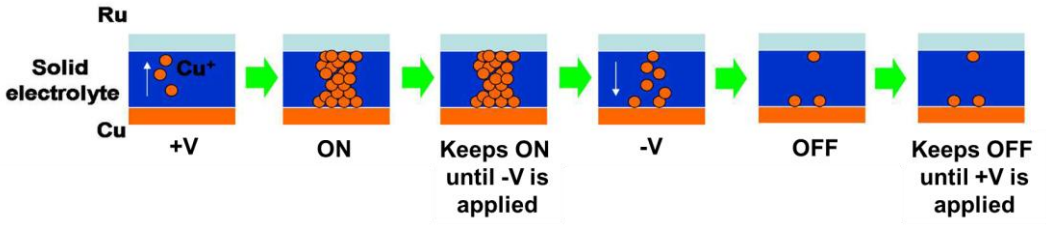


図 1-1　原子スイッチ（ナノブリッジ）のスイッチング動作

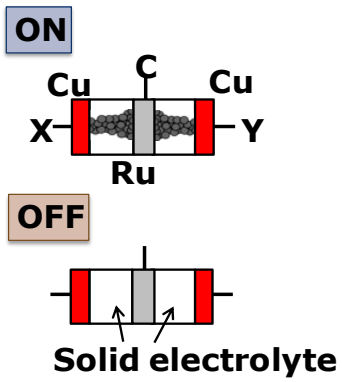


図 1-2　相補型原子スイッチ（Complementary Atom Switch: CAS）

ナノブリッジ内の銅イオンブリッジの形成の様子を図 1-3 に示し、ナノブリッジを使用したプログラマブル FPGA の FIB-SIM (Focused Ion Beam – Scanning Ion Microscope) /TEM (Transparent Electron Microscope) によるクロスセクションイメージ図 1-4 に示す。図 1-5 に示すのは今回評価に使用した、プラスチックパッケージに組み込んだ状態のナノブリッジ FPGA である。

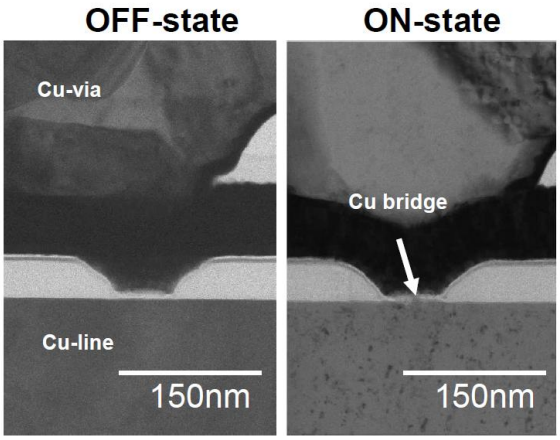


図 1-3　ナノブリッジ内の銅イオンブリッジの形成 [49]

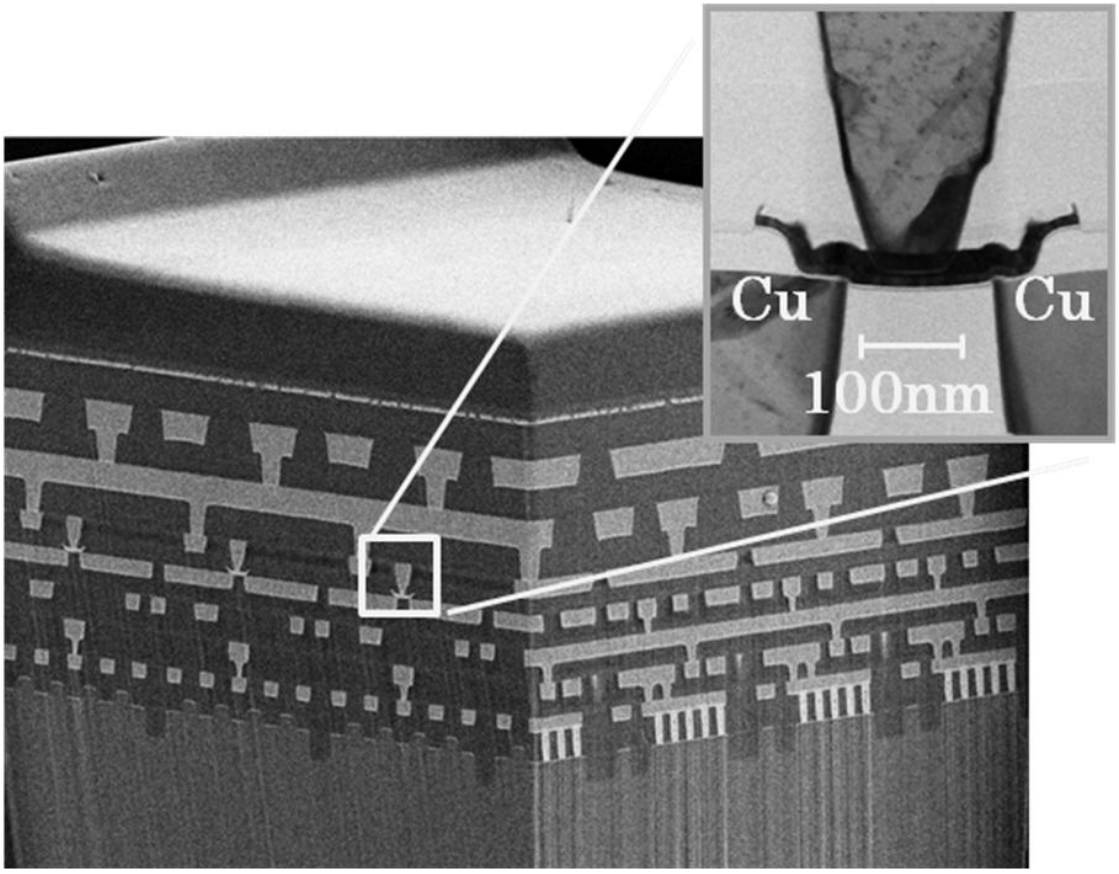


図 1-4　FIB-SIM/TEM によるクロスセクションイメージ [49]

図 1-5　プラスチックパーケージに封入した評価用デバイス [49]

本研究では、ナノブリッジを活用するにあたり、プロセッサにプログラミング言語でプログラムを書くようにセンシングデバイスに信号処理をプログラムするしくみを実現することを目標とした。高級プログラミング言語との親和性を持たせるために演算処理は関数として扱うこととし、ナノブリッジを活用して関数で表現される処理をいかにハード化するかを研究課題とした。センサに演算処理機能を埋め込むことを検討するに当たっては、あらかじめマイクロプロセッサ等を組み込むのではなく、CBIC や FPGA が有するプリミティブな論理演算素子のみの存在を前提とした。

関数型言語の利用を前提とし、演算処理をハード化するしくみの概要は次のとおりである。アプリケーション要求を実装するために適した関数を定義し、この関数を最も効率的に実装できるよう、プリミティブな論理演算素子を回路として凝集させる。次に、アプリケーション要求から導き出されるデータフローグラフに基づいてこれらの関数を接続すると共に、条件判断などの制御構造を表わす制御フローグラフに基づいて組織化を行い、最終的に外界の実世界との入出力を接続する。この様子を図 1-6 に示す。このプロセスを実現するアーキテクチャと設計手法を確立することが求められる。

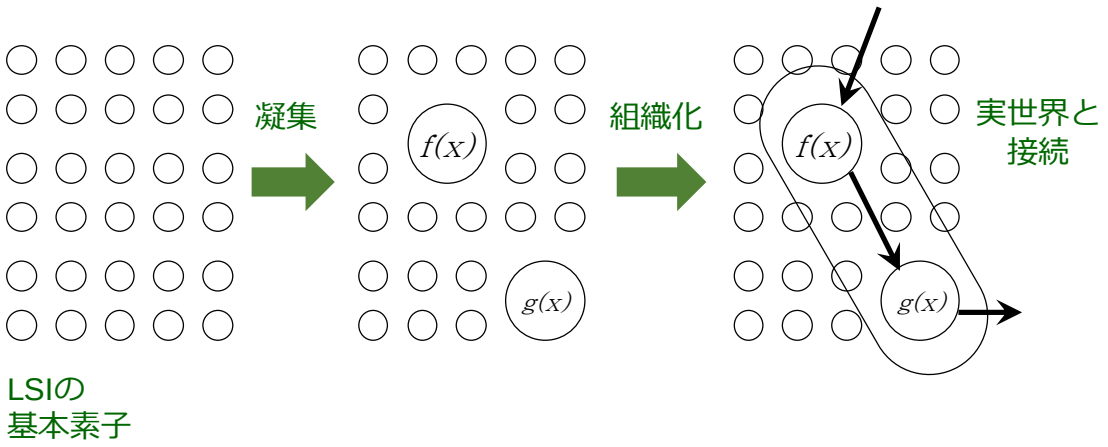


図 1-6　関数表現をハード化するアプローチ

1.2. IoT 用センシングデバイスに対するインテリジェント化要求

あらゆるものがインタネットに接続される Internet of Things (IoT)は、社会情報インフラストラクチャに不可欠なものとなりつつある。利用者の視点でシステムを構築するに当たっては，IoT は 図 1-7 に示すように人々の生活をより快適・安全にし，ビジネスにおいても新たな成長・進化をもたらす巨大な「知恵袋」と定義され，あらゆるモノが Machine to Machine (M2M)ネットワークによりいつでもどこでも繋がり，さまざまな情報をクラウドデータセンターに収集・蓄積することで，社会の知恵やノウハウを集積するシステムが描かれる [5]。この IoT を構成する技術要素としてセンサ，ネットワーク，情報技術 (Information Technology: IT)，ロボティクスが識別され，エネルギー，水道，交通，物流，放送，通信といったライフラインを支えるために，道路，空港，鉄道，発電所，プラントといった社会インフラシステムが構築される。IoT を構成するセンサノード、アクチュエータノード、および中継ノードは、IoT アプリケーションのための基本的で重要な構成要素であり、これらはすべてがいつでもどこでも M2M ネットワークを介して接続される。センサネットワークは、集中的に設置された高性能コンピュータと高性能ネットワーク機器で構成されるクラウドシステムと統合される。

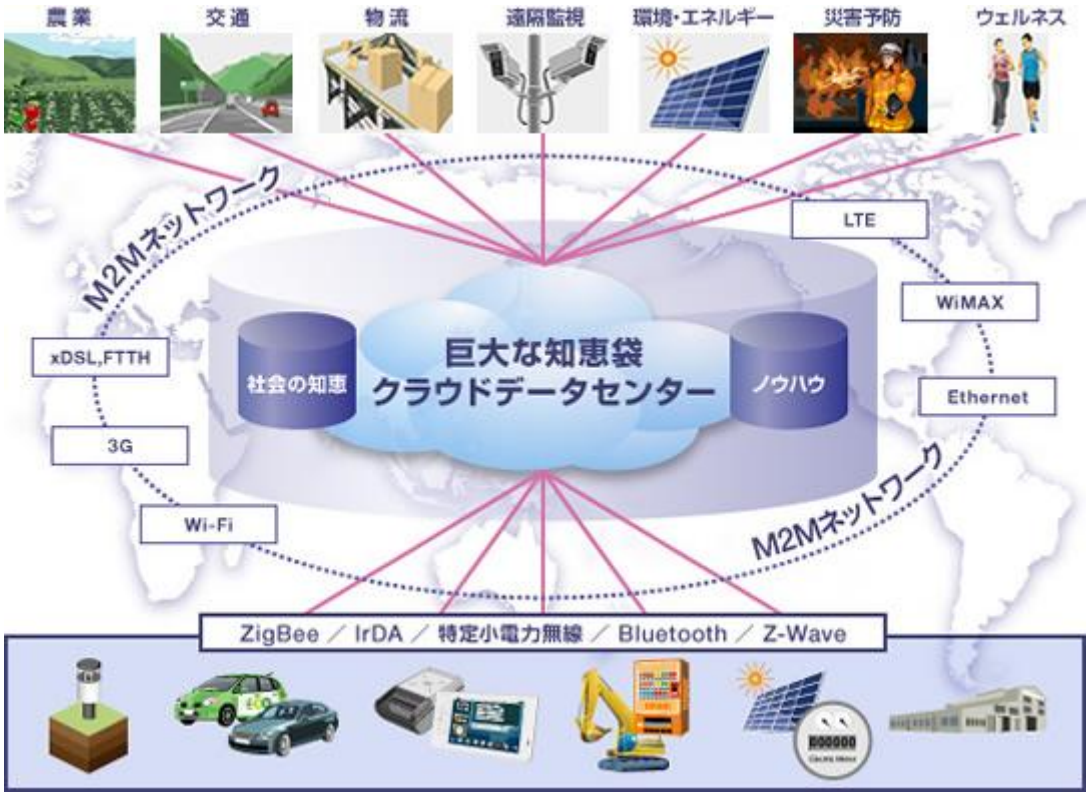


図 1-7　組込みマイコンを活用した IoT [5]

IoT に用いるセンサノードから出力される膨大な量のデータをクラウドシステムに伝送して利用するに際しては、センサノードとクラウドシステムとの間に設置されている通信回線の伝送チャネル容量の制約に配慮する必要がある。既設のネットワーク通信回線のチャネルを介して十分なデータを送信するためには、センサノード内にデバイスコンピューティング機能を実装し、データ量を削減する必要がある。すなわち、大量の生データをクラウドシステムに送信するのではなく、有意なデータを抽出してデータ量を削減したうえで送信することが求められる。このためには、データの校正や、選択、圧縮などのセンサを知能化するインテリジェントな機能が必要である。図 1-8 に示すように、観測センサを搭載した人工衛星は宇宙空間に配置されて利用されるため [6]、システムインテグレーションにあたっては衛星と地上局間の伝送チャネル容量の制約を考慮する必要がある。すなわち、既設の通信ネットワークの伝送容量を有効に活用するためにセンサの中でデバイスコンピューティングによりデータ容量を削減することが求められる。具体的には、リモートセンシング画像で非可逆圧縮を行う場合には 1/6 程度までは画質を落とさずに圧縮できるとされている。センサ出力信号をリアルタイムに逐次処理し、これくらいまでデータ量を圧縮できれば十分地上局に必要なデータを伝送できる。センサ出力信号をリアルタイムに逐次処理し、内容を判断できる処理速度が実現できれば、空間情報を圧縮するか、ないしは周波数領域で圧縮処理を行うかなどの選択も可能となり、Hyperspectral Sensor への応用も期待できる。

データセンターに集められたデータを活用するためには，必要な時、あるいは定期的にデータを収集し、伝送する機能が求められることから、センサの知能化を図るに際してはセンサに埋め込む演算器にも同様な特性が求められる。すなわち、データを収集するセンサモジュール自体にデータの校正や選択，圧縮といったインテリジェントな機能が組み込まれていることが望ましい。このような背景から、本研究はセンサを知能化するインテリジェントな機能の実現をターゲットとし、その中でも特に、IoT に用いられるセンサのアプリケーションに焦点を当てた。

さらに、このようなインテリジェント化が求められるセンサは多数フィールドに配置されることから，屋外の過酷な環境も考慮する必要がある。すなわち、安価で小型，かつ低消費電力であることが求められると共に高信頼性も求められる。センサノードの機能は、広い温度範囲、高い振動レベル、さらには高濃度な自然背景放射線に晒される過酷な環境でも常に利用可能でなければならない。特に、自動車や原子力関連施設，宇宙システムなどの社会インフラシステム向けには高信頼性が求められる。IoT を構成するシステムの設置場所や利用形態は図 1-8 に示すように大気圏外にも想定されており[6]，これらの環境上の制約等も考慮しながらシステムを構築する必要がある。例えば、ノイズや自然背景放射線によって引き起こされる半導体メモリのソフト・エラーは主要な懸念事項であり、特に半導体メモリに見られるデータ反転のようなソフト・エラーは、連続動作のために回避できなければならない。このためには、プロセッサエレメントに用いる半導体メモリを原理的

に減らすことが有効である。更にメモリのみならず周辺のメモリ制御回路も削減できれば効果的である。センサノードでは、実装面積が小さく、消費電力が削減されることが求められており、従来の組込みシステム用 MCU において大きな面積を占めている外付けメモリやチップ上の内蔵メモリ、およびメモリ制御回路を削減することはソフト・エラーを回避するばかりでなく、実装面積と消費電力の削減にも効果が見込める。

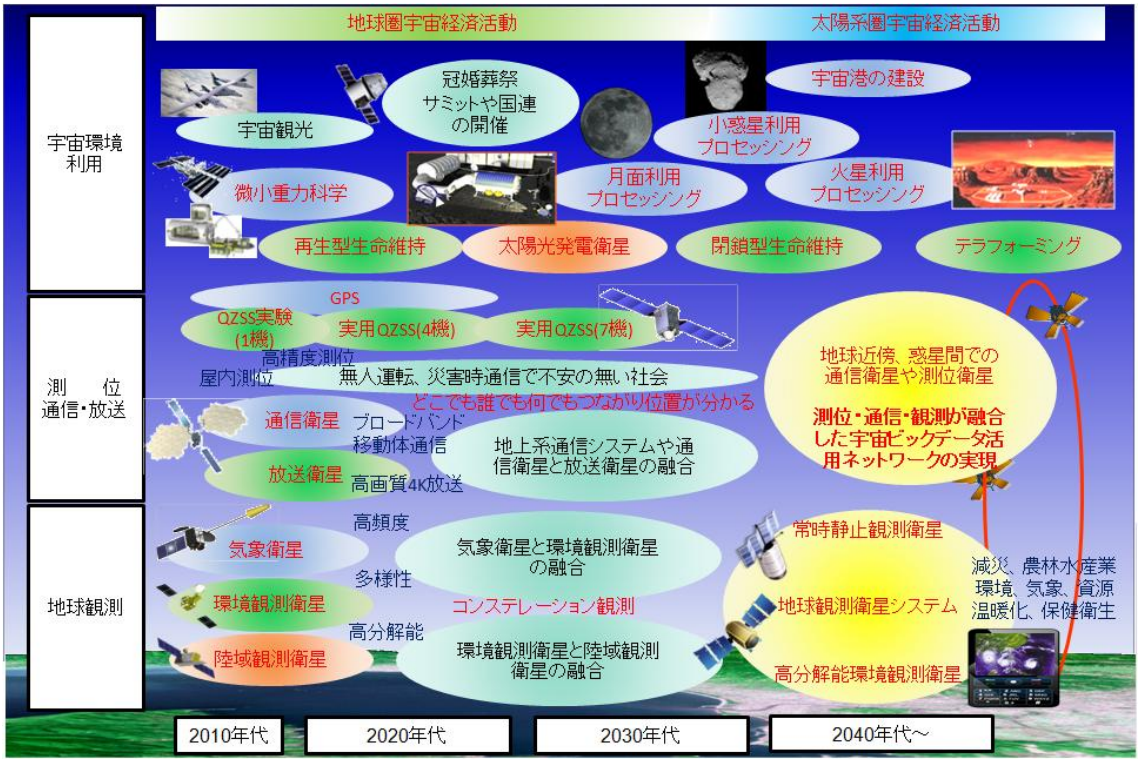


図 1-8　宇宙利用・地球観測系ロードマップ [6]

1.3. 信頼性要求

本研究ではセンサに埋め込むプロセッサのアーキテクチャを検討するに当たり，極力汎用性を持たせるよう留意したが、実験・評価を進めるにあたっては人工衛星搭載用機器を評価のモチーフとした。IoT の中で人工衛星はセンサノード，および通信の中継ノードとして位置付けられ，図 1-9 に示すように地上システムのネットワーク通信機器や高性能コンピュータと合わせてシステム構築がなされる[7]。人工衛星のセンサノードとしてのサイズや複雑さは、環境監視、交通監視、ホームセキュリティなどに用いられるセンサノードと比較しても大きい。信頼性の観点からは、医療や自動車などの他のアプリケーションでも高いレベルの信頼性が要求され、人工衛星と同様に厳しいものと考えられる。しかしながら、低コスト、量産性、カスタマイズの多様性、短納期といった他の観点からの要求の相違から、IoT サブシステムに求められる信頼性要求には差異があり得る。

人工衛星が晒される放射線環境については，従来は宇宙空間特有のものとして扱われていた。しかし，昨今の半導体プロセスの微細化により，デバイスのパッケージから放出される微量の放射線や，地球上の自然放射線によって引き起こされるデータ・エラー（後述するソフト・エラー）が顕在化してきている [67]。このため，宇宙機の設計技術者と民生機器の設計技術者が情報交換するケースも出てきている。民生機器の開発者が半導体プロセスの微細化に伴って直面しているソフト・エラーへの対策は，激しい放射線環境にさらされる宇宙機の設計技術者がこれまでに実施してきたものと非常に近いものとなっている。本研究では、特に、極力多くの応用に適するように、半導体デバイスを用いたメモリの使用量を削減することを信頼度向上の尺度とした。半導体メモリ素子を削減することは、人工衛星では直接的に信頼度の向上に資すると共に、他の産業分野でも共有する課題である。また、後述するようにメモリ素子に依存したアーキテクチャから脱却することにより、プロセッサアーキテクチャとしての演算処理性能も向上が見込める。以下、本研究の背景となる半導体素子に対する放射線の影響について纏める [8]。

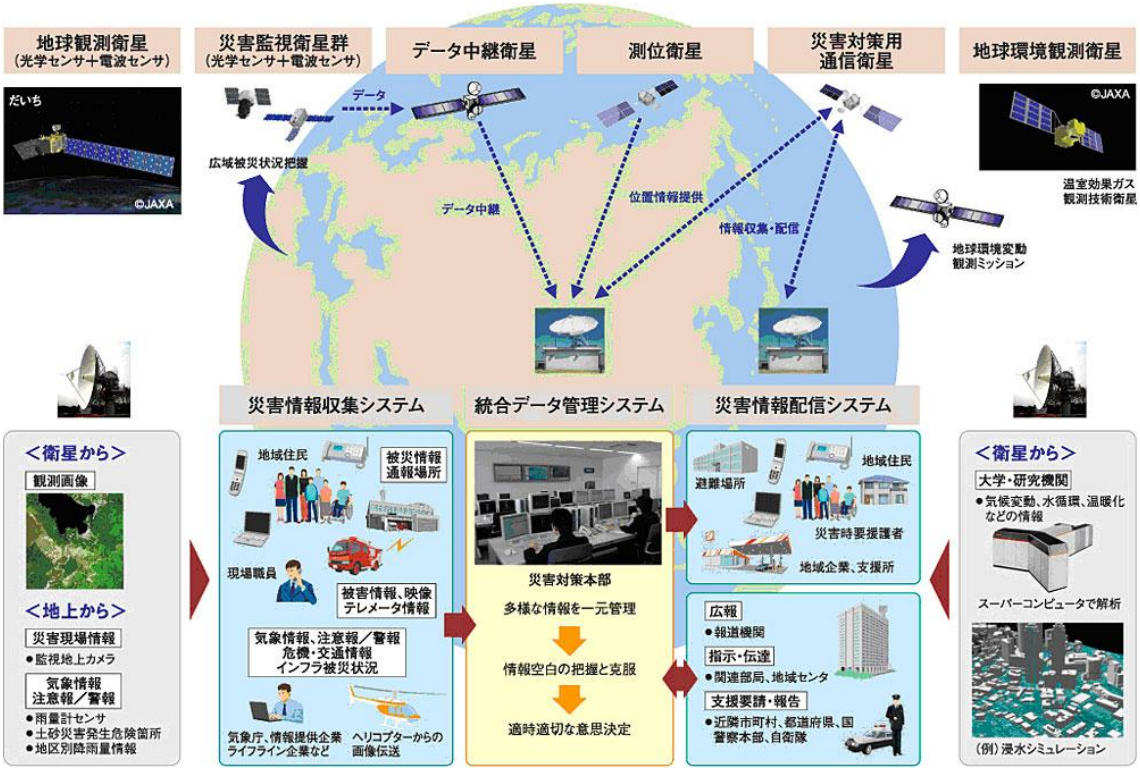


図 1-9　宇宙システムを活用した IoT システムの実装 [7]

宇宙機に使用する電子部品は，ロケットで打ち上げる際に 10G～20G におよぶ振動レベルや-30℃～+60℃に及ぶ温度変化にさらされる。打ち上げに成功し，初期の運用準備期間を経て地球周回軌道などに安定すると，とくに激しい振動にさらされることはなくなる。

その一方で、外部環境としては、宇宙空間特有の温度差や放射線環境を考慮する必要がある。温度差に関しては近年の宇宙機システム開発設計技術の進歩により、運用中の衛星内部の温度を適切な一定の範囲内に維持すべくコントロールされた環境で運用されるようになってきており、温度変化を緩やかにすることが可能となっている。しかしながら、輸送、打ち上げなども含めたライフサイクル全体を考慮した場合、コンポーネント単体として動作を保証する温度幅は依然として 70℃から 90℃にも達する。

放射線環境としては、いったん大気圏から外に出ると宇宙空間特有の高いレベルの放射線にさらされる。この宇宙放射線環境のもとになるのは宇宙空間に存在する高エネルギー粒子であり、惑星の近傍や太陽風などに含まれるγ線などの放射線、および銀河宇宙線などに含まれる重粒子イオンなどである。前者は主として、その総線量によってゲートの閾値の変動やリーク電流の増加などのデバイス特性の劣化を招く。後者に対しては陽子（プロトン）や電子，中性子，重イオン粒子，α粒子などによる影響を考慮する必要がある。なかでも量的に多いのは太陽風（solar wind）が運ぶ放射線である。これらの放射線が地球の磁気圏（magnetosphere）に捉えられている様子を図 1-10 に示す [8]。荷電粒子がデバイスを貫通することによる電荷の発生により、データの反転が生じるシングル・イベント・アップセット、ないしは過度の電荷により CMOS デバイスなどの内部に存在する寄生ダイオードが活性化され、デバイス内部で短絡を起こす現象（シングル・イベント・ラッチアップ）などの過渡フォールトを引き起こす。

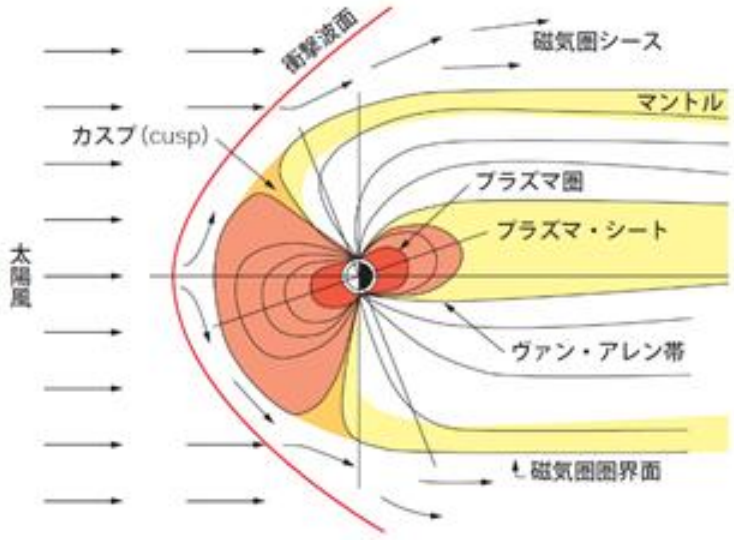


図 1-10　放射線が地球の磁気圏（magnetosphere）に捉えられている様子 [8]

この高エネルギー粒子が半導体デバイスに与える影響として，以下のようなものが挙げられる。

1）トータル・ドーズ効果

トータル・ドーズ効果は，部品がさらされる放射線の総量に応じて，半導体デバイスの性能が劣化する現象である。これは永久変化とみなされ，トランジスタのしきい値が変化することなどにより，性能が劣化していく。この現象は，主として陽子と電子によって引き起こされる。これらの粒子が大量に飛び込んできても，時間がたつと緩和される（アニール効果と呼ばれる）場合もあるが，一般には劣化を抑えるのは難しく，金属などの遮蔽によって影響を軽減させる対策が一般的である。

2）シングル・イベント・アップセット

シングル・イベント・アップセット（Single Event Upset: SEU）は，メモリやフリップフロップなどの LSI 内の回路に太陽や銀河系からの重イオン粒子や陽子などの高エネルギー粒子が当たり，データが反転してしまう一過性の故障である。一般には「ソフト・エラー」と呼ばれている。たとえ正しいソフトウェアやデータがメモリに格納されていても，このような誤りが発生してしまうとデジタル LSI の誤動作を引き起こす。デジタル LSI のメモリセルやフリップフロップを構成するトランジスタの PN 接合部に高エネルギー粒子が衝突し、通過する際の様子を図 1-11 に示す。PN 接合部を重イオンが通過することによりファネリングと呼ばれる現象により電子・正孔対が生成され、メモリセル内の電荷量が変動する。これにより、一時的なデータエラー（ソフト・エラー）が発生するが、メモリセルの場合にはこのまま電荷量が所期の値に回復されないとビットエラーが発生することになる。

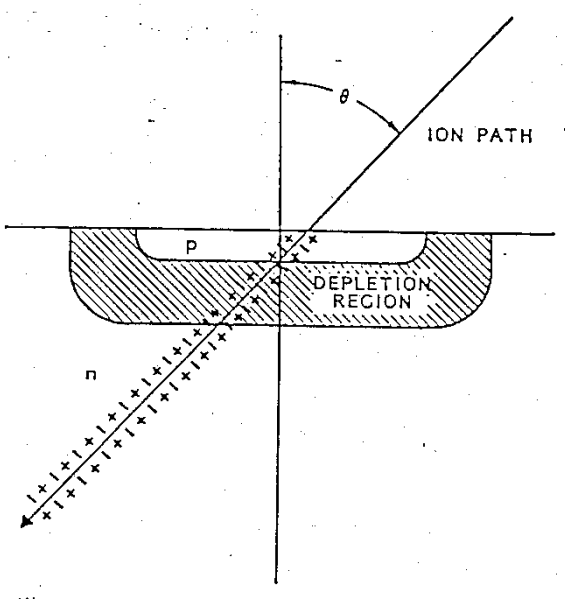


図 1-11　半導体デバイスの PN 接合を重イオンが通過する際の影響

3）シングル・イベント・ラッチアップ

シングル・イベント・ラッチアップ（Single Event Latchup: SEL）は、前述の SEU と同じように高エネルギー粒子の放射線によって引き起こされる現象である。一般に，LSI などの半導体デバイスには回路構成の組み合わせによりサイリスタとよく似た回路が予期せずできていることがある。この半導体基板上に形成されているサイリスタ構造に高エネルギー粒子が当たってしまうと，この部分の回路がオンしてしまい，過大な電源電流が流れる。一過性の故障に留まらず永久故障につながることもあるので「ハード・エラー」とも呼ばれる。

上記の SEU と SEL に加え、Single Event Transient（SET）、Single Event Functional Interrupt（SEFI）など、このほかにもさまざまな放射線による影響がある。これら SEU、SEL、SET、SEFI などを総称して Single Event effect（SEE）と呼ぶ。これらの影響を回避するため，物理的にデータの反転が起こりにくいデジタル LSI を選択したり，正しいデータをつねにリフレッシュしてできるだけ回復を図り、故障が生じてもほかに影響が波及しないような対策を施したりするなど、部品を保護すべくさまざまな対策が施される。特に人工衛星は宇宙空間に設置され、利用されるため、高レベルの放射と広い温度範囲の厳しい環境においても常に動作を継続しなければならず、高度にインテリジェントな処理を厳しい環境の中で実行しなければならない。上述の放射線の影響により宇宙機システムの動作中に半導体デバイスにしばしば見られるソフト・エラーなどの SEE に対処するにあたっては、原因となる粒子のエネルギーの大きさのために遮蔽などの対策は現実的ではない。このためデバイス自体が SEE の影響を回避しやすい特性を有する必要がある。本研究ではこのために提案した従来の多数決方式を上回る高信頼性アーキテクチャを統合した。

1.4. 論文の構成

本論文は次のように構成している。まず、概要に於いて本研究の貢献範囲とオリジナリティを論じた。第 1 章では本研究の背景を述べ、第 2 章では、ナノブリッジを活用したデバイスを IoT に効果的に活用すべく、本研究の目的としたナノブリッジを活かすアーキテクチャ、その対象範囲と新しいデバイスの活用指針を述べた。更に新しいアーキテクチャを実現する設計手法の研究課題を論じた。第 3 章では第 2 章で提起したナノブリッジを活かすアーキテクチャを提案し、その応用範囲を論じた。第 4 章では第 3 章に述べた新しいアーキテクチャを実現するツールとして高位合成／動作合成を活用すべく拡張した概念と実装との対応について述べた。第 5 章ではナノブリッジの特長を活かすことが期待されている高信頼性分野に適用するに際して、従来の多数決方式を凌ぐ、単一故障点を排除した高信頼性化アーキテクチャの提案内容と評価結果について述べた。第 6 章では第 3 章から第 5 章に亘って述べた提案内容をフレームワークとしてまとめると共に、本研究により提案したアーキテクチャと設計手法を用いて試作評価を行った結果について述べ、所期の成果が得られたと共に、他分野への波及効果も確認できたことを報告する。第 7 章では本論文の結論をまとめると共に、今後の展望を示した。図 1-12 に本論文の構成を示す。

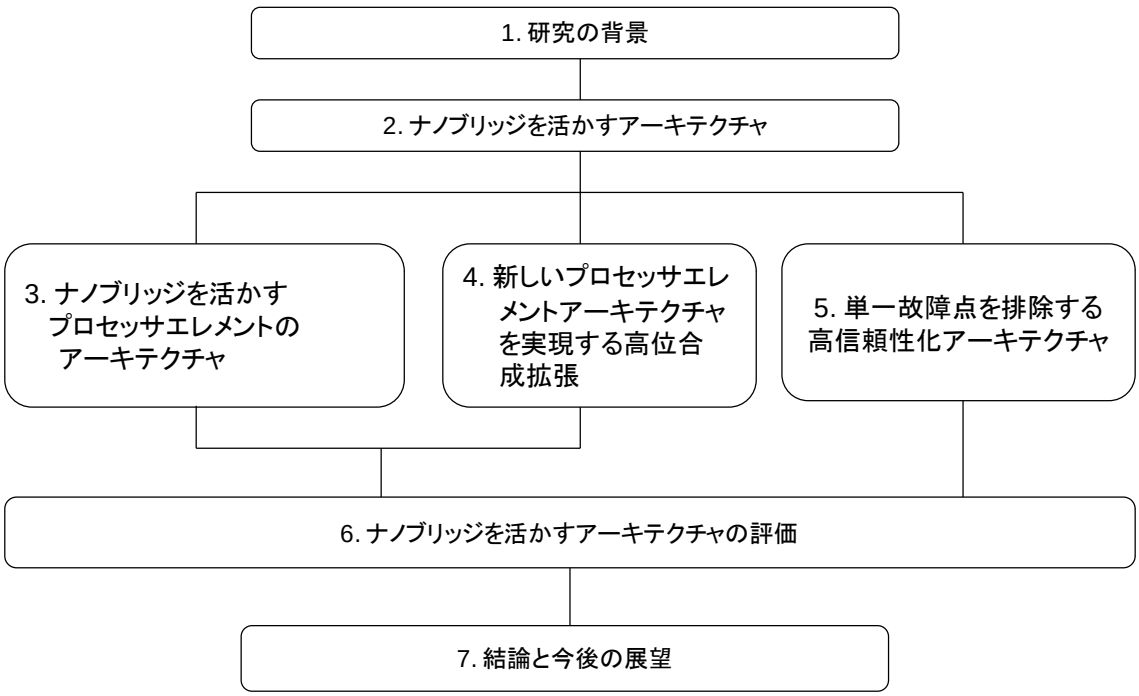


図 1-12　本論文の構成

第2章
ナノブリッジを活かすアーキテクチャ

前節で述べた新しいデバイスの登場と並行して演算処理を直接 LSI の回路として実装する技術も十分実用に耐えるようになってきている。具体的には、センサ信号を処理するアルゴリズムを C 言語等の高級プログラミング言語で記述することを可能とする高位合成技術や、それらのアルゴリズムを直接回路に落とし込むための動作合成技術も実用化されている[9, 10, 11]。これらの新しいデバイス技術やソフトウェア技術を統合し、センシングデバイス自体に演算処理を埋め込むことが実現できれば、IoT のように多数のセンサを配置するアプリケーションを既設の通信回線に容易に接続することができるようになる。本研究は、IoT に用いられるセンシングデバイスに演算処理機能を埋め込むにあたり、これらのデバイスやソフトウェア技術を効果的に活用して小型化、低消費電力化、および高信頼性化を達成するアーキテクチャを開発し、設計手法を確立することを目的として進めた。

2.1. 研究対象範囲

IoT アプリケーションはセンサの出力する信号とその解析結果を単に伝送するシステムに留まるものではない。IoT には様々な機能・性能を有するコンピュータが活用されることから、一連の信号処理システムを構成する全体的な枠組みを考慮しなければならない。IoT アプリケーションに用いられる各種のコンピュータと、それらを接続するネットワークを合わせた 5 階層のアーキテクチャが提唱されている。この 5 階層モデルの概念を図 2-1 に示す [12]。この 5 階層の IoT アーキテクチャモデルを参照し、本研究の対象範囲を明らかにする。センサ出力信号を処理する一連のシステムを柔軟に構成し、運用手順ないし利用状況の変化に対しても堅牢性を持ち、システム全体の性能を維持するしくみは、この 5 層モデルで表すコンセプトの重要な特徴である。階層という概念を導入することで、他のシステムとの接続性が向上し、リンクの実現が容易になり、システムの段階的・持続的な成長をサポートし、更に新しい価値を生み出すことが可能となる。

コンピュータは 3 階層から構成され、実世界と直に接するデバイスコンピューティング層、インタネット等のネットワーク回線との通信制御機能も有するエッジコンピューティング層、および豊かな資源を元に高性能なコンピュータを集積したクラウドコンピューティング層からなる。各コンピューティング層との間はネットワークで接続する。本研究では現時点で最もインテリジェント化が遅れているデバイスコンピューティング層にナノブリッジを活かすことを主眼とした。具体的には、IoT アプリケーションに用いられるセンサをインテリジェント化すべく、センシングデバイス自体に演算処理機能を埋め込むために、

ナノブリッジを活用するデザインフレームワークを確立することを目的として研究を進めた。

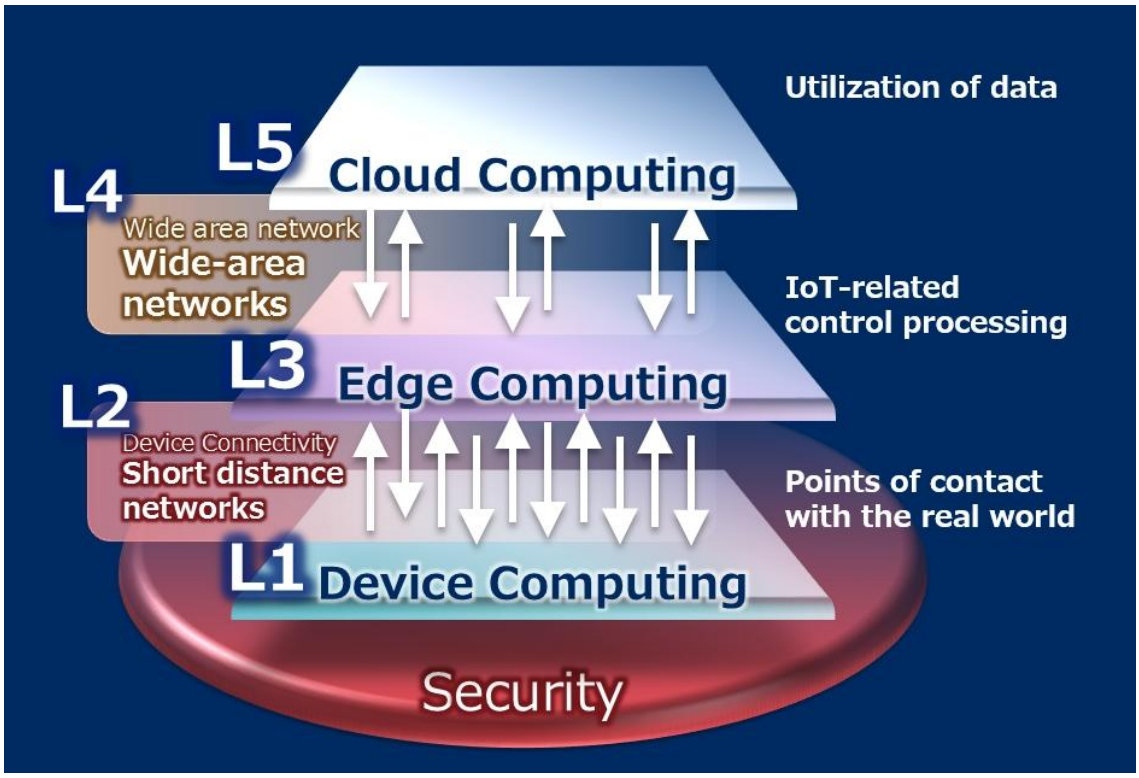


図 2-1　IoT architecture 5-layer model [12]
http://www.nec.com/en/global/about/vision/closeup/01.html

2.2. 新しいデバイス:ナノブリッジの活用指針

ナノブリッジは前章で述べたように、回路接続情報を記憶するために半導体メモリデバイスを用いずに金属架橋により微視的なスイッチを実現したところに優位性がある。この優位性を活かし、演算処理デバイスのコアとして用いる Processing Element (PE)について外付けメモリ素子を用いないアーキテクチャを提案し、そのアーキテクチャを用いてセンサ信号を処理するプロセッサを構成する設計手法を確立することを目的として研究を行った。ここで言う外付けメモリとは物理的に PE の外部に接続するメモリを意味しており、半導体プロセス技術により LSI の 1 チップ上に集積されたメモリも含むものとする。

外付けメモリ素子を用いないアーキテクチャを開発することにより、メモリ素子やメモリ制御回路を削減することができ、自然放射線による半導体メモリのソフト・エラーの影響を受けないようにすることができる。このことから高信頼性を要する応用分野への適用がまず考えられるが、後述するように、本研究で提案するアーキテクチャによりリアルタ

イム演算処理性能も向上した。常時データを出力し続けるセンサに埋め込む PE にとっては、このリアルタイム演算処理性能も重要な利点であり、センサ信号を処理するデバイスとしては相乗効果が見込める。

一方、ナノブリッジは実用化の目途が得られた段階にある新しい技術であり、最新の FPGA や PROM 程集積度は高くなく、書込み回路(配線)の信頼性については、これからフィールド評価を確立する段階にある。したがって、本研究では集積度や書込み回数の向上を前提としないメリットを追求することとした。特に、ナノブリッジの動的な変更、ないし再構成や ON/OFF の反応速度などはこれから改善が進められる見込みであり、これらに依存したメリットの追求は今後の課題とした。

2.3. ナノブリッジを活かすアーキテクチャ

前述のデバイスコンピューティングに割り当てるべき処理を実装するために、センシングデバイスに埋め込む PE のアーキテクチャを検討するに当たっては、以下の 3 種類のプログラマブルデバイスの利点を統合するものとした。

a) 組込みマイコン（Micro-Controller Unit: MCU）
b) Field Programmable Gate Array (FPGA)
c) 動的再構成プロセッサ（Dynamically reconfigurable Processor: DRP）

外付けメモリ素子を用いることなくこれら 3 種類のプログラマブルデバイスの利点を統合するためには、これらのプログラマブルデバイスの課題を解決しつつ、取り込む必要がある。センサの出力信号を処理するためには MCU、あるいは FPGA を用いたプロセッサが用いられることが多く[13, 14, 15, 16]、これらのプロセッサの小型化、および低消費電力化が課題となる。さらに、ソフト・エラーに耐性を持たせるためにはこれらのプロセッサに用いられている 2 種類の外付けメモリ素子を不要としなければならない。一つは MCU の主記憶として用いられるプログラムメモリであり、もう一つは書換え可能な FPGA の回路接続情報を格納するコンフィギュレーションメモリである。

MCU の利点は C 言語等の高級プログラミング言語でソースコードを記述して開発を進めることができる点である。この利点を活かしつつ、従来の MCU に使用されているプログラムメモリを削除するに当たっては、近年実用化の進む高位合成技術を適用することとした [9, 10, 11]。高位合成技術は専用 LSI や FPGA の設計技術として注目されているが、本研究ではストアードプログラムアーキテクチャの代替技術として活用することとした。このためには高位合成技術に用いられる高級プログラミング言語の利点をうまく活用する必要があり、本研究では粒度別階層構造を持つ PE アーキテクチャを提案した。

FPGA の利点はセンサの出力する信号を止めずにそのまま処理できる、所謂 Wire rate processing ができることである。書き換え可能な FPGA では回路接続情報を格納するために SRAM（Static Random Access Memory）などを用いたコンフィギュレーションメモリ

を備える。SRAM は宇宙放射線や地球上の自然背景放射線によりソフト・エラーを生じるため、これらの放射線に耐性を持たせるためには SRAM の代替品を用いる必要がある。EEPROM（Electrically Erasable Programmable Read-Only Memory：電気的に消去可能なプログラマブル読み出し専用メモリ）またはフラッシュメモリなどもコンフィギュレーションメモリとして用いられ、SRAM よりもソフト・エラーに対して耐性は優るが、ソフト・エラーを無くすまでには至らない。コンフィギュレーションメモリの内容を定期的にリフレッシュしてソフト・エラーの蓄積を防ぐためにメモリスクラブ回路と呼ぶ外部回路を付加することがあるが、これは消費電力および実装面積を増やすなどのデメリットが生じる。したがって、ソフト・エラー対策のみならず、消費電力と実装面積を削減するためにも、これらのコンフィギュレーションメモリを削除することが有効である。コンフィギュレーションメモリはナノブリッジを使用することにより直接的に削除できる。

特定の信号処理目的としては、動的再構成可能プロセッサ（Dynamic Reconfigurable Processor: DRP）も用いられている [17, 18, 19, 20, 21, 22]。状態遷移に応じて最適な演算器を接続できるという動的再構成アーキテクチャの利点を活用することにより、MCU や FPGA に比べて DRP は処理速度や実装面積について優位性がある。その一例として、DRP は同等の処理をストアードプログラム方式のプロセッサで実行するのに比べて二桁程度の低消費電力を実現できることが報告されている[23, 24]。その一方で、プログラムメモリとコンフィギュレーションメモリの双方を持つことになり、IoT 用のセンサに埋め込むためには、やはりソフト・エラー耐性を持たせるなどの配慮が必要になる。本研究では動的再構成アーキテクチャを活用しつつ、外付けメモリ素子を要しないプログラマブルデバイスを構成するものとした。本方式は特許[90]として認められており、本研究のオリジナルのものである。

以上のメモリを削除しても、CMOS（Complementary Metal Oxide Semiconductor）などの半導体の下地自体は高信頼性化アーキテクチャによりソフト・エラー耐性を持たせる必要がある。この高信頼性化アーキテクチャは特許が認められており[91, 92]、本研究のオリジナルのものである。一般的によく用いられている多数決方式を上回る信頼度を有する、この高信頼性化アーキテクチャ [25, 26, 27, 28] に本研究で提案する PE のアーキテクチャを組み合わせることにより、更に高い信頼性を実現できることが明らかになった。

2.4. ナノブリッジを活かす設計手法

本研究では、前述した新しいアーキテクチャのみならず、ソフトウェア技術を統合し、センシングデバイス自体に信号処理を埋め込む設計手法を提案した。ナノブリッジは前述のように、従来の書換え可能な FPGA に用いられているコンフィギュレーションメモリを不要とする特長が先行して活用されている。通常、FPGA を用いてプロセッサを構成するに当たっては、回路接続イメージを維持した RTL（Register Transfer Level）のプログラ

ミング言語が用いられることが多い。しかしながら、センシングデバイスの出力信号の処理アルゴリズムは高度化してきており、処理アルゴリズムを通常のマイクロプロセッサに高級プログラミング言語でプログラムを書くようにプログラムすべく、C 言語等の高級プログラミング言語で処理を記述できることが望ましい。演算処理を LSI の回路として実装するソフトウェア技術としては、アルゴリズムを直接回路に落とし込むための動作合成技術に加え、それらのアルゴリズムを高級言語で記述することを可能とする高位合成技術も実用化されている。高位合成技術は、ANSI-C 言語のような高級プログラミング言語でソースコードを記述し、CyberWorkBench（CWB）等の高位合成ツールを使用して VHDL または Verilog 言語によるレジスタ転送レベル（RTL）のソースコードを生成するものである。この設計フローを図 2-2 に示す [9, 10, 11]。ナノブリッジを活かしてセンサ信号を処理するために、従来の FPGA に用いられている動作合成技術に加え、近年実用化の進む高位合成技術の利点をうまく活かす設計手法を本研究では提案した。

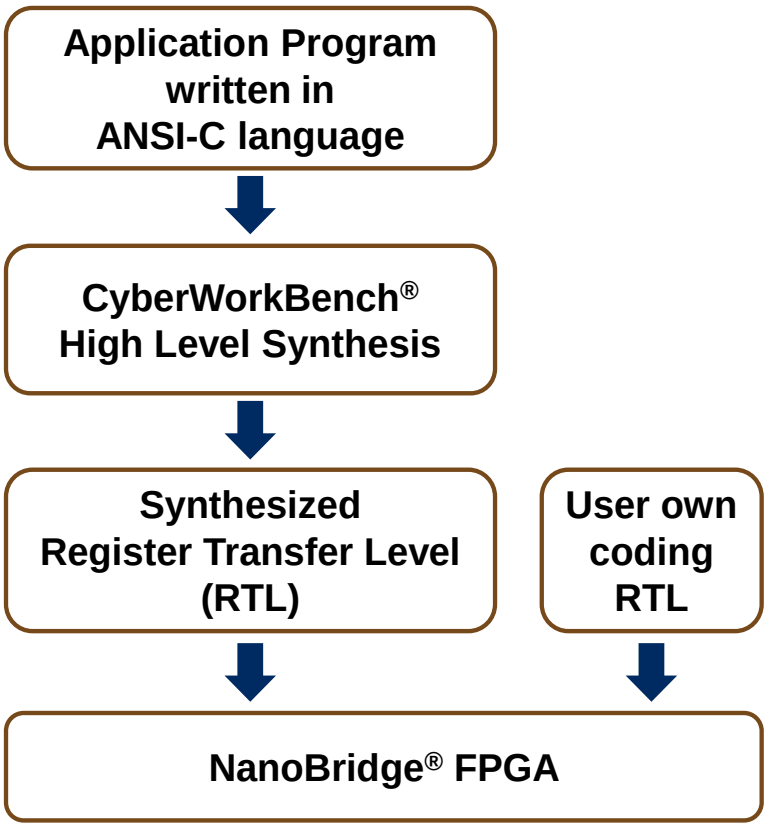


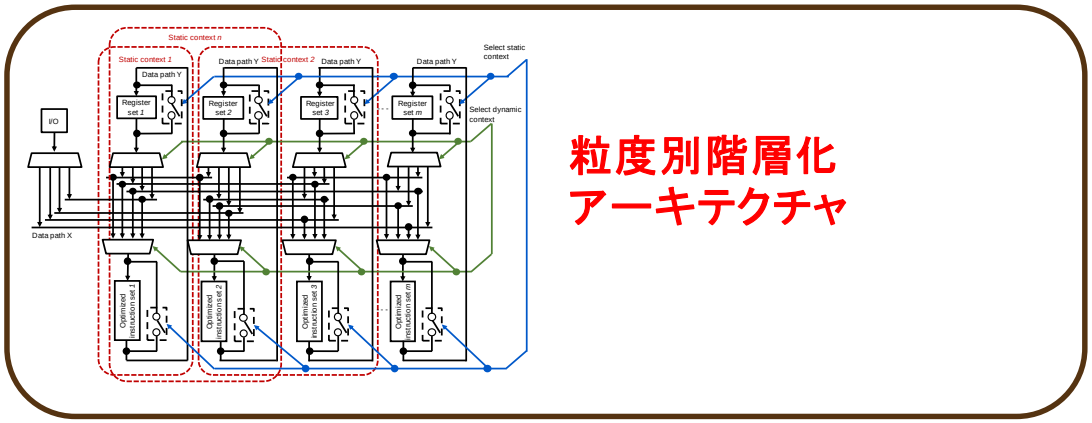
図 2-2　高位合成／動作合成を活用してナノブリッジ FPGA を設計するフロー

ただし、高位合成技術は専用 LSI や FPGA の設計技術として注目されているためか、高級プログラミング言語の活用は生産性向上の手段として考えられていることが多い[9, 10, 11]。本研究では、従来の専用 LSI や FPGA を代替するのみならず、従来の MCU が有する

ストアードプログラムアーキテクチャも代替して外付けメモリ素子を不要とすることを狙ったため、関数定義の手法を活用して回路実装レベルの最適化を図ることを目的とした。このために、本研究では粒度別階層構造を持つ PE アーキテクチャを提案し、この粒度別階層構造を用いて回路実装の最適化を図っている。本アーキテクチャは特許[90]を取得しているオリジナルのアーキテクチャであり、これを活用することにより、ナノブリッジの利点を最大限に活かすことができる。すなわち、本研究では高位合成技術をプログラムメモリを無くす技術として活用し、高級プログラミング言語開発環境を活用して MCU をナノブリッジ FPGA に置き換えることを狙ったとも言える。

3 種類のプログラマブルデバイスの利点を高位合成技術を活用して統合するに当たっては、プログラムの拡張性を可能とするスケーラビリティを考慮しなければならない。なぜなら、スケーラビリティを有さなければ、センサに埋め込める処理は一つの LSI チップの規模に制約されることとなり、近年技術開発の進む三次元実装などの LSI チップの積層技術を活用する上で障害となってしまうためである。このため、本研究で提案するアーキテクチャは、PE 自体を連結して拡張できるスケーラビリティを持たせることを狙った。これらの研究目的を一覧としてまとめたものを図 2-3 に示す。

Processing Element (PE) 間通信を標準化
→ **スケーラビリティ**

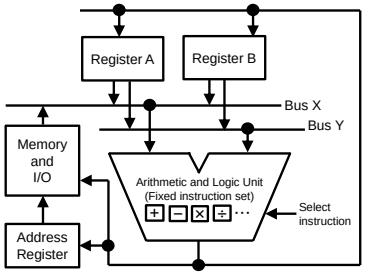


粒度別階層化アーキテクチャ

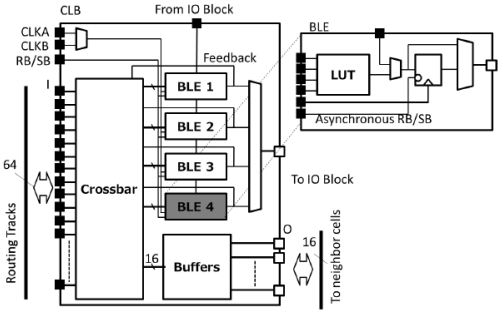
高級言語でプログラムする。

Wire rate processing

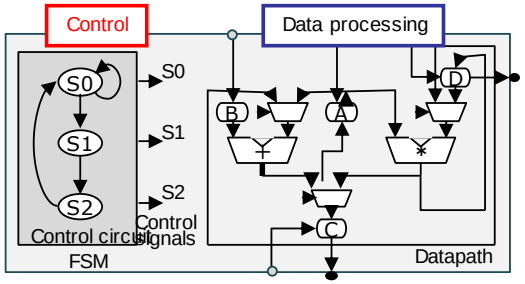
各状態に応じて必要な演算器を接続



Micro-Controller Unit (**MCU**)



Field Programmable Gate Array (**FPGA**)



Dynamically reconfigurable processor（**DRP**）

図 2-3　3 種のプログラマブルデバイス（MCU, FPGA, DRP）の利点の統合

2.5. 関連研究

本研究で提案、および評価を行ったセンサノードに演算処理機能を埋め込むことを目的としたアーキテクチャ、およびその設計手法と高信頼性化設計手法について、従来の関連研究をどのように発展させたかを述べる。IoT アプリケーションではセンサが屋外に設置されることが少なくなく、広い温度範囲などの厳しい環境を考慮しなければならない。自然放射線や宇宙線などの背景放射線による半導体メモリのソフト・エラーも考慮しなければならない。特にソフト・エラーについては半導体メモリに物理的に耐性を持たせることも解ではあるが、半導体メモリを減らすことによりソフト・エラーの発生源自体を削減することができる。

2.5.1. センサノードに演算処理機能を埋め込むアーキテクチャに関する関連研究

IoT に用いられるセンサノードをインテリジェント化するに当たっては、インタネット等のネットワーク回線との通信制御機能も有するエッジコンピューティング層に MCU がよく用いられる。このため、MCU の小型化、および低消費電力化が常に追求されている。MCU では 3.2 節に述べるようにプログラムを格納するメモリ素子がアーキテクチャ上必須であり、半導体プロセス技術の高度化を背景としてこのメモリ素子が演算器と同じ LSI 上に集積されることも多い。これは組込みマイコンとも称されている。この時、メモリ素子のみならずメモリ制御回路も同じ LSI 上に集積される。したがって、外見上は LSI 1 チップに MCU が集積されるが、相対的に MCU 上の広いエリアをメモリ素子とメモリ制御回路が占めており、ソフト・エラーの発生源が減っているわけではない。むしろ、近年の半導体プロセス技術の高度化を背景としてメモリ容量は増加傾向にあり、ソフト・エラーの発生確率は増えている。

近年の FPGA の高集積化、および開発環境の充実を背景として MCU を FPGA で置き換えることによりインテリジェントな機能をセンサに埋め込む例が増えている。この場合は FPGA に内蔵しているコンフィギュレーション可能な論理ブロック(Configurable Logic Block: CLB)、および Look Up Table (LUT)を組み合わせてセンサ出力信号の処理アルゴリズムを実装する。すなわち、MCU が外付けメモリ素子上にアプリケーションプログラムをバイナリ・イメージで書込むのに対し、FPGA は MCU に書き込まれるアプリケーションと同様なセンサ出力信号処理アルゴリズムを動作合成技術により CLB と LUT にて構成するハードウェア回路の結線情報として実装し、センサ出力信号を処理する。MCU を FPGA で置き換えることによりプログラムを格納する外付けメモリ素子とそのメモリ制御回路を不要とすることが可能となり、ソフト・エラー発生源自体を無くすことができる。

MCU を FPGA で置き換えることにより、ソフト・エラーを減らすことがアーキテクチャ上可能となるが、CLB と LUT の配置によってセンサの出力信号を処理するアルゴリズムを実装するに当たっての非効率性も指摘されている [29]。この問題に対して本研究では

二つのアプローチを取った。一つは、インタネット等のネットワーク回線との通信制御機能を実装するに当たってはソフトウェア資産を最大限に活用することが現実的であるから、エッジコンピューティング層には引き続き MCU を用いることを前提とし､実世界と直に接するデバイスコンピューティング層に FPGA や CBIC を用いてデータの校正や、選択、圧縮などのインテリジェントな処理機能を実装することとした。もう一つは、上述の非効率性を改善するために、従来のソフトウェア技術で効率化のために良く用いられる関数定義を FPGA や CBIC のアーキテクチャに応用した。特に重点対象を書き換え可能な FPGA とした。FPGA のアーキテクチャとして関数定義の応用の有効性が認められれば、CBIC にそれを応用することは容易である。

ソフトウェア技術で用いられる関数定義を FPGA 等のハードウェアに活用するための手段としてはハードマクロ化技術が応用できる。すなわち、ハードウェア回路の入出力を明確に定義し、演算処理内容を入出力間定義に結び付けることにより、ソフトウェア技術で用いられる関数に対応させることができる。センサ出力信号を処理するにあたり、関数定義を活用することの有効性については既に報告しており[42, 48]、概要を Appendix D に述べた。したがって、ハードマクロ化技術をいかに効率化し、小型化、および低消費電力化を実現するかが課題となるが、本研究では小型化を主とした効率化を追及し、低消費電力化はナノブリッジや動的再構成デバイス技術を応用することを主眼とした。FPGA のマクロ化技術については種々の報告がされている。[72]ではハードマクロ化技術を用いて動作周波数や配置配線性能を向上させている。[73]では論理クラスタ回路中の Basic Logic Elements (BLE) の数と処理速度および配置面積の依存性を論じている。[74]では回路のパッキング技術を用いて実装面積を削減している。[75]ではマクロゲートを実装することにより、回路遅延と実装面積を減らすことを論じている。[76]ではハードマクロの効率的な生成方法を論じており、ハードウェア記述言語(Hardware description language: HDL)による記述を高位記述として活用している。これらは FPGA の基本素子である CLB の対象としている論理演算に対してハードマクロの利点を論じているが、回路設計上の工夫について改善を図っているものである。これに対して、本研究ではソフトウェアの関数定義と同様な大域的な実装規模の改善を図るものであり、これらの関連研究では言及されていない、新しい視点に基づくものである。

2.5.2. 関数定義を活用するアーキテクチャに関する関連研究

ソフトウェア技術で用いられている関数定義を応用するためには、必然的に高位合成技術を必要とする。高位合成技術は C 言語などの高級プログラミング言語によりハードウェア回路を生成するものであり、C 言語も属する関数型言語の技術を継承している。高位合成技術を活用する報告は種々なされている [77, 78, 79, 80, 81, 82, 83]。しかしながら、これらは高級プログラミング言語の利用による開発効率の向上にとどまっている。[84, 85, 86, 88, 89]では高位合成ツールを使いつつ、動作合成の効率を改善する方法を提案しているが、

回路生成技術そのものの改善を論じている。高位合成技術と DRP を活用して FPGA の非効率性を改善するための提案も為されている [30, 31, 32, 33]。これらの方式による DRP では汎用性を意図して基本演算を備えた演算モジュールと、これらをメッシュ状に接続するスイッチマトリクスを実装している。これらの方式は量産化を意図して設計されているものである。しかしながら、IoT アプリケーションの中で使われるセンサは、例えば 8、12、14、または 16 ビットなどの様々なデータインタフェースを有している。このため、柔軟にビット幅を変えられる PE を配置することが望まれる。従来の DRP はデータパスのビット幅はある値に固定されており、基本的な論理演算装置(Arithmetic Logic Unit: ALU)の配列の間にあらかじめ配置されたデータパスはレイアウトエリアの最適化と消費電力の削減の障害となる。[86]ではマクロを再利用することによりコンパイル時間の短縮を達成している。本研究ではマクロの再利用の概念を更に発展させ、関数レベルの記述により再利用を可能とするのみならず、関数間のリソース共有を達成することにより、実装面積の削減を達成している。さらに本研究では、アプリケーションに応じて最適化された演算機能を有する ALU と柔軟なデータパスを動的に再構成できるよう、高位合成ツールの拡張機能とハードウェアを対応させることで従来の DRP の制約を払拭している。

2.5.3. 高信頼性アーキテクチャに関する関連研究

高信頼性化するには MCU を多重化することが良く行われており、ここでは従来の多重化方式の比較・評価と各方式の課題について述べる[25]。評価判断基準を明確化するため、本研究では宇宙機搭載用機器をモチーフとし、MCU 多重化の目的は単一点故障（シングル・ポイント・フェイリャ）を除去することと定義している。すなわち、一部分の異常・故障によりシステム全体の機能が失われることを防止することを目的とする。

MCU の多重化方式はコールド・スタンバイ方式、およびホット・スタンバイ方式に大別される。前者は待機冗長方式に代表されるものであり、通常は一系統のみ運用しており、異常・故障が生じた場合には他系に切り替えて運用を再開するものである。我が国ではこの方式は姿勢軌道制御装置に早くから用いられている。衛星のミッション機器やアンテナを常に地上に向け地球を周回する地球指向制御の多い実用衛星に比べ、地球を含めた各種天体にセンサを向けるために比較的任意な姿勢軌道制御を行う科学衛星の姿勢制御装置は、バス系機器の中では最もインテリジェント化が進んだものと言える。このコールド・スタンバイ方式は、その性格上、主系から冗長系への切り替えに時間がかかるため、主としてフェイル・セーフ機能に重点が置かれたものである。

主系から冗長性への切り替えの時間を短縮化するために一部分だけ主系と冗長系に常時電源を供給し、双方に同時にデータを書き込んでおく OBC (Onboard Computer)を開発した。図 2-4 に示す OBC は我が国最初の人工衛星搭載用の汎用コンピュータである[34]。運用計画が格納されるデータ領域が大きいため、主系から冗長系に運用を切り替える際、改めて運用データを冗長系に書き込むためにはかなりの時間がかかる。本 OBC の例では、無

線回線を介して運用データを書き込むためには 7 時間を要する。この時間を節約するため、予め運用データを両方の系に同時に書き込んでおき、冗長系に切り替えた際にもデータ領域の再書き込みを不要としている。

図 2-4　ADEOS 搭載用 OBC　©JAXA

後者のホット・スタンバイ方式は多重構成系を同時に運用するものである。この方式には以下のようなものがあげられる。

a) 多数決方式

各 MCU の入出力に多数決回路を設け、一つの MCU が異常・故障を起こしても、全体的には停止しないシステムである。しかし、本方式は、多数決回路自身の異常・故障には対応できていない。すなわち、MCU は多重化されているものの、通常は多数決回路で信号出力は一点に集中しており、多数決回路自身の異常・故障はシステムダウンにつながる。したがって、本システムでは、多数決回路自身が単一故障点（シングル・ポイント・フェイリャ）につながっている。

通常は、多数決回路として耐放射線性、および高信頼性を有する特殊なデバイスを用いることで対処しているが、これは、デバイスの故障率に依存した確率的な故障率低下を狙ったものであり、機能的な解決策とは異なる。また、多数決回路自体を多重化する対応策もあるが、いずれにしても、入出力が一点に集まる点があり、システム的にも重量、消費電力などが増大する。さらに、多重度を増すにあたり、例えば、三重多数決を四重多数決に拡張する場合、多数決回路周りが新規設計となる。すなわち、多数決回路の存在が、システムの拡張性を阻んでいる。なお、この方式は最も早くから検討されており、使用実績も多く、さまざまな改良が施されている。

我が国における代表的なシステムとしては、平成 6 年度に打ち上げられた技術試験衛星 VI 型(ETS-VI)に搭載された姿勢制御装置があげられる [35]。このシステムでは、システムバスの二重化、記憶部の三重多数決化などが用いられ、信頼度の高いシステムが構成されている。また、姿勢制御装置に対する要求を考慮することにより多数決回路に起因する単

一点故障を排除する構成を実現している。ETS-VI 搭載用 ACE（姿勢制御電子回路）内に実装された計算機は、ETS-VI の頭脳とも言えるもので、16 ビットの CPU モジュール 4 台を内蔵している。各 CPU モジュールにはそれぞれ 3 重多数決ローカル RAM をもち、システムバス、共有メモリモジュール、出力ポートなども 2 重化されており、さらに共有メモリモジュール内の RAM も 3 重多数決化されている。このように各モジュール内の RAM が個々に 3 重多数決化されているが、一方、システムバス・インタフェースに関しては、Duplex 方式がとられている。計算機部をフォールトトレラント化する際に、2 CPU 並列運転(Duplex)方式と 3 重多数決方式との比較が行われているが、姿勢制御において数サイクルの制御の中断は許容可能なため、信頼度・消費電力の観点から Duplex 方式が採用されている。本方式は大規模なシステムであるが、多数決回路に起因する単一点故障を回避するとともに、システム要求の範囲内においてフェイル・オペラティブを実現しうるシステムと言える。

b) ソフトウェア通信方式

これは、各 MCU が基本的には並行して処理を行い、あらかじめ決められたチェックポイントにおいて互いに通信を行い、データの照合を行いながら処理を進めていくものである。この方式は、異常・故障が起こった場合、故障の分離、システムの再構成、および再同期（正常な MCU にマスタ機能を移し、ロールバック処理などを行ってシステムの動作を続行する）が比較的容易に行える。

組み込まれるソフトウェアは常に通信・照合方式をインプリメントしなければならず、その結果、本方式が適用されるシステムは比較的大規模なシステムに限定される。また、複数の MCU をシステムバス接続する場合、やはり多数決回路などによって信号を一つにまとめる必要があり、このハードウェア自体は本方式にとっては本質的でないにもかかわらず、単一故障点（シングル・フェイリャ・ポイント）を内在せざるを得ないジレンマをもっている。

一方、本方式はソフトウェアによる通信を利用することから、既設計の計算機を流用することを可能にする。スペースシャトルに用いられているガイダンス・コンピュータは、この特徴をうまく利用している [36]。我が国では科学衛星「ひてん」に搭載された OBC が本方式を用いている [37]。この OBC はデバイスのハイブリッド化を行い、小型化を達成している。我が国の高度なデバイス技術の好例と言えよう。

c) FRM (Functional Redundancy Monitor) 方式

本方式は MPU に、自己の（アドレス、データ、ステータス）バス出力と外部のバス信号の内容とが一致しているか否かの比較監視機能をもたせるものである [38]。マイクロプロセッサは監視モード／通常モードの走行モードを指定できる。通常モード走行のプロセッサはこの比較回路を動作させずチップ内の処理結果をそのままバスに出力する。監視モー

ドのプロセッサはチップ内の処理結果をバスに出力せず比較回路の入力とし、また外部からの信号を比較回路の入力とする。したがって、各プロセッサのデータバス、アドレスバス、ステータス信号線はおのおの単純に Pin-to-Pin 接続するだけでプロセッサ間でお互いの処理内容の監視をすることができる。比較回路が不一致を検出すると監視モードのプロセッサは専用端子を通して外部回路に対してこれを通知する。

この方式は、上記の 2 方式に対し、以下のような利点を有している。

・比較・検出回路が各 MCU にそれぞれ内蔵されているため、多数決方式と異なり、これらの回路が単一故障点とならない。

・ソフトウェア通信方式と異なり、常時信号の比較が行われる。また、MCU 上のソフトウェアは、比較が行われていることを通常は意識する必要がない。

しかしながら、宇宙機搭載を考慮した場合、解決すべき問題は少なくない。主要なものとして下記のものがあげられる。

・MCU 内蔵の機能であるため、使用する MCU が限定される。国産の宇宙機搭載用 MCU としては、宇宙航空研究開発機構認定品として μPD55040-028 (8bit)、T4915 (16bit)、V70 (32bit)があり、また認定品ではあったが収束した H32 (32bit)がある。さらに、我が国初の宇宙機搭載用 16bit MPU であり、搭載実績も豊富な μPD55090-012、および COSMO16 等、多様な MCU がある。しかしながら、FRM 機能を有するものは V70 のみである。また、我が国で入手可能な海外製の宇宙機搭載可能な MCU でも FRM 機能を有するものはなく、実質的に本方式を用いることのできる MCU が限定されてしまう。

・再構成を行うためには、多数決回路相当のものが必要となる。具体的には、N 重系を構成する各 MCU の出力する異常検出信号をまとめて判断し、正常な MCU を選択した上でバス上に信号を出力する権利をもつ MCU を指定する機能が必要となる。なぜなら、各 MCU が誤って異常検出信号を出力してくる可能性があるためである。したがって、この異常検出信号を取りまとめて判断する部分は、構成法によっては多数決回路のように単一故障点となりうる。

この方式を用いた代表的な OBC としては、宇宙ステーションに本実験モジュール(JEM)に搭載されている通信制御装置(JCP)があげられる [39]。

本研究では以上の課題を克服し、演算処理、故障検出、故障分離、再構成の各機能の分散化を実現し、単一故障点を排除することに成功している。更に、本研究では新たに研究を進めた関数演算器化機能により従来の方式では困難であった冗長管理部自体の耐故障性を実現しており、これを第 5 章で高信頼性化アーキテクチャとしてまとめている。

第3章
ナノブリッジを活かすプロセッサエレメントのアーキテクチャ

本章ではセンサに組込むプロセッサとして、特にセンシングデバイスそのものに埋め込むのに適したプロセッサアーキテクチャを提案する。この新しいアーキテクチャを提案するに当たっては、外付けメモリ素子を要するという MCU や FPGA の課題を改善しつつ、それらの利点を取り込んだ。また、提案アーキテクチャでは DRP の特徴を活用しているが、従来の DRP にはセンサ出力信号を処理するために高位合成技術を活用する上での制約もあり、それらの制約を克服している。これらの過程を経て確立したプロセッサアーキテクチャについて述べる。

3.1. 新しいアーキテクチャに求められる要件

MCU の利用形態を考慮すると、センシングデバイスに埋め込むために必要な処理速度性能と消費電力を達成する上で、外付けメモリ素子を要しないアーキテクチャが有効である。これはストアードプログラムアーキテクチャとは異なるプログラム実行方式を実現することになる。センシングデバイスに埋め込むプロセッサは、従来の技術分野では所謂組込みマイコンに相当する。そこで本研究では 組込みマイコンに求められるアーキテクチャを前提としたうえで、外付けメモリ素子を要しないことを要求として設定した。以下、組込みマイコンを Micro-Controller Unit (MCU)として総称する。

MCU はエッジコンピューティング向けに広範に使われるプロセッサである。センサ信号処理は MCU の主要なターゲットアプリケーションである一方で、センシングデバイス自体に埋め込むためには、MCU を凌ぐ小型化、低消費電力化が求められるようになってきている。従来の MCU をセンシングデバイスに埋め込むことを考えると、次の二点が問題点として識別される。

一つの問題点は、従来の MCU はストアードプログラム方式を基本としている。ストアードプログラム方式にはオーバーヘッドが内在する。具体的にはチップ面積のかなりの部分を算術演算および論理演算といった演算リソース以外の周辺回路（メモリシステムやメモリコントローラ等）に使用しており，消費電力や演算速度が犠牲になっている。また，割り込み処理や並行処理をソフトウェアで仮想化して擬似的に実行しており，ソフトウェア処理負荷のオーバーヘッドが大きい。例えば割り込み処理がソフトウェアのレジスタアク

セスになっており，並行処理がマルチプロセス・オペレーティングシステム（Operating System: OS）やマルチタスク OS で擬似的に実装されている。

もう一つの問題点は，SoC（System on Chip）設計の生産性を向上させるために実用レベルに達し、広く使用されるようになった高位合成技術が活かされていないことである。従来の MCU では命令セットが固定されており，LSI (Large Scale Integration)プロセスをフルに活かした性能が得られていない。従来の MCU 上で動作するコンパイラは C 言語または C ++言語などの高級言語で書かれた所与のユーザプログラムから実行可能なバイナリコードを生成する。このコードはオンチップの演算論理ユニット（Arithmetic Logic Unit: ALU）と I / O インタフェース、またはコプロセッサなどの外部ロジックチップのいずれかによって実行される。シーケンシャルな命令からなるバイナリコードは、四則演算等のオンチップの汎用の Instruction Set Processor（ISP）により処理される。高位合成技術を活用すれば、C 言語または C++言語で書かれたユーザプログラム専用のデータパスとステートマシン・コントローラを生成することができる。これは、動作合成によりユーザプログラムが回路として合成された後に実装されて状態機械として埋め込まれ、専用のハードウェアが生成されることを意味する。固定的な ISP ハードウェア向けの従来のソフトウェアコンパイルと比較して柔軟に演算ユニットを構成することができ、実装面積と電力効率を大幅に向上させることが見込まれる。組込みプロセッサ・アプリケーションの開発には、高級プログラミング言語による反復的かつ段階的な設計が必要であるため、ハードウェア回路であってもプログラマビリティは必要である。そのようなプログラマビリティは、動作合成技術と動的再構成可能プロセッサアーキテクチャの両方を採用することによっても維持することができる。

前者の問題点については DRP による改善例が報告されている[23, 24]。同じ演算量であればストアードプログラム方式のプロセッサと比較して DRP は 2 桁に及ぶ消費電力の低減が可能とされる。これは、チップ面積の大部分が計算リソースとして使用できるためである。後者の問題点については，従来は高位合成を活かしたプロセッサのアーキテクチャが確立しておらず，高位合成技術はフルカスタムないしセミカスタム LSI などのシステム・オン・チップ（System on Chip）LSI の設計効率の改善手法として論じられるに留まってきたことが課題である。

IoT に活用するセンサに埋め込むプロセッサとしては上記二点を改善したアーキテクチャを適用することが望まれる。リコンフィギャラブルプロセッサ上での実用的なアプリケーションの実装は、近年成熟してきた高位合成／動作合成技術によって可能であり、本論文では CyberWorkBench（CWB）[9, 10, 11]を活用した。高位合成技術を利用した DRP は従来の MCU を置き換えることができ、IoT アプリケーションの効率的かつ高性能な実装を可能にする。また、その利用方法については、リコンフィギャラブルであることを活かし、伝送するデータはシステムの運用状況に伴って選択基準を変更したり、データ圧縮方法を変更するなどの対策も必要になるものと見込まれ、これらへの対応も課題である。す

なわち、ライフサイクルを考慮した機能の実現を図る必要があるものと見込まれる。

3.2. 開発課題を明らかにするための計算モデル

メモリ関連回路を削減するにあたっては、エンタープライズアプリケーション用のマイクロプロセッサとネットワーク通信機器用に用いられる FPGA との違いに見られるような組込みシステムアプリケーションの特性を利用することができる。IoT アプリケーションに用いるセンシングデバイスに埋め込むのに適したプロセッサアーキテクチャの開発課題を洗い出し、適したアーキテクチャを調べるために、本研究ではまず新しい計算モデルを策定した [40]。これは IoT アプリケーション向けの組込みシステムの開発を目的として定義したものである。これを組込みオートマトン（Embedded Automaton: EA）と称する。この計算モデルについて説明する。

組込みオートマトンのモデルにおいては PE の入力は実環境であり, それらは時刻, Input / Output（I/O)信号，および隣接 PE とのインタフェース信号とすることができる。IoT アプリケーション開発の観点からは、以下の 4 つの前提を置くことができる。

a）製品を出荷した後のプログラムの修正回数は限られており、ほとんどの場合プログラムは滅多に変更されない。ほとんどの場合、アプリケーションプログラムが PROM (Programmable Read Only Memory) に書込み可能であり、一旦アプリケーションプログラムを製品の PROM に書き込んで市場に出荷した後は、プログラムの変更はほとんど想定されない。出荷前の調整を含めても、プログラムは高々数回書き換えることができれば良い。すなわち、開発環境には柔軟性が求められるものの、アプリケーションをハードウェアロジックとして実装しても生産性には影響しない。

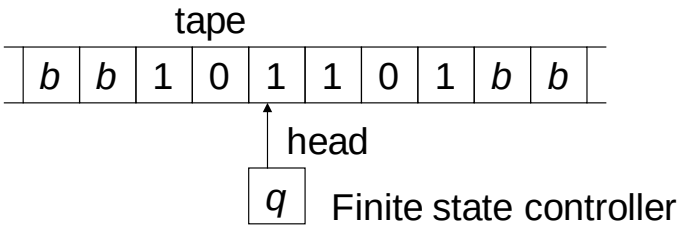
b）組込みシステムの入力シーケンスは現実の外界からの逐次入力であり、プロセッシングエレメント（PE）の入力データは逐次データ列として入力される。データ並びに対して後戻りしてアクセスする必要は無い。

c）アプリケーションプログラムをソフトウェアとして実装する必要は無く、ハードウェア回路として実装しても支障は無い。組込み用 PE の設計結果を適用するのは、各バージョン毎に特定のロットに限られ、組込みシステム製品のハードウェアが次のバージョンでは変更されることは珍しくない。このため、アプリケーションプログラムをハードウェア回路として実装することができる。

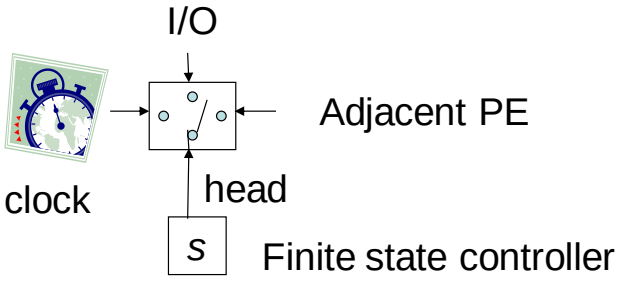
d）入力データはリアルタイムクロックの時間タグで識別される。故に入力は時系列のデータ列と見なせる。

これらの 4 つの前提条件を EA に反映した。EA の入力データは現実世界の物理的な信号であり、従来の有限オートマトンでテープとしてモデル化されているような蓄積されたデータではない。EA の入力データは入出力（I / O）信号、隣接 PE からのインタフェース信号、および時刻データとしてモデル化されている。

この組込みシステム開発の立場から要求する計算モデルとして設定した組込みオートマトン(embedded automaton)と従来の有限オートマトン(finite automaton)を比較したものを図 3-1 に示す。



(a) 有限オートマトン



(b) 組込みオートマトン（Embedded Automaton: EA）

図 3-1　有限オートマトンと組込みオートマトン

図 3-1 において、*b* はブランクを示し、*q* は従来の有限オートマトンの有限状態制御部を示し、*s* は Embedded Automaton の有限状態制御部を示す。EA は Finite Automaton と同様に次の 5 項組で定義される。

$$EA = (S, \Sigma, \delta, s_0, F)。 \qquad (1)$$

ここで S は取り得る状態の集合，Σは想定され得る入力データの集合，δは状態遷移関数の集合，s_0は S の初期状態に対応する一つの要素，F は入力データのシーケンスを受理することによって遷移する状態の集合を示す。S と δ はそれぞれ有限な状態の集合、および有限の数の状態遷移関数であり，動作合成ツールにより記述される。Σはシステム設計により予め精度の決められたデジタル値であり，センサ信号の A/D コンバータ出力と見てもよい。したがってΣも有限集合である。F は S に含まれ、最終状態となり得る部分集合であるから有限な集合の部分集合であり、後述する高位合成／動作合成ツールで記述し得る。

入力の初期時点と終端時点は文字通り時刻で表される。したがって、EA の計算可能性は従来の Finite Automaton と同様に評価され，計算可能であることが示される。

従来の有限オートマトンにおいてテープ上のデータとしてモデル化されているのは入力データと有限状態制御部のパラメータである。言い換えれば、従来の有限オートマトンの状態は、入力データとしてテープ上にモデル化されたパラメータと、有限状態コントローラ q 内の状態によって定義される。 一方、組込みオートマトンの状態は、有限状態コントローラ s 内の状態によってのみ定義され、アプリケーションプログラムに内在するステートマシンは、有限状態コントローラのプリミティブなステートマシンに統合されていることを意味する。この違いは、ストアードプログラムアーキテクチャに基づく MCU の本質的なオーバーヘッドを説明している。

従来の有限オートマトンと EA を比較し、EA 計算モデルを評価することにより、従来の有限オートマトンに基づいて実装されている MCU に見られる固有のオーバーヘッドが容易に識別でき、2 つの問題を特定することができた。これらの問題を検討することにより、センシングデバイスに組み込むプロセッサとして相応しいアーキテクチャが見出せる。

一つの問題は、MCU が図 3-1 に示す従来の有限オートマトンに基づくことに起因する。従来の有限オートマトンではアプリケーションに内在する状態遷移を模擬する仮想的な Finite State Machine（FSM）が図 3-1 でテープとしてモデル化されている。このテープに記録されている仮想的な FSM は、MCU ではリニアなアドレスを有する外付けメモリ素子のメモリセル内に実装される。アプリケーションから抽出された FSM の状態遷移は、有限状態コントローラによりリニアなアドレスで指定される一連のメモリセル上のアドレス指定動作としてシミュレートされる。すなわち、有限状態コントローラは、プロセッサというよりもシミュレータとして使用されている。言い換えれば、従来の有限オートマトンを参照したストアードプログラムアーキテクチャに基づいて実装された MCU は一種のシミュレータである。MCU では 2 種類の FSM が存在し、メモリセル上に記述された 1 つの仮想的な FSM が、有限状態コントローラである MCU 内のもう一つの FSM 上でシミュレートされる。物理的には、リニアなアドレスを持つ一連のメモリセルのアドレスをアドレッシング機能により指定することにより機能を実現するものであり、メモリアクセス上の制約から生じる性能低下はフォンノイマンボトルネックと呼ばれる。

割り込み処理はソフトウェアにおけるレジスタアクセス動作として実装され、並行プロセスやタスクはマルチプロセス・オペレーティングシステム、ないしマルチタスク・オペレーティングシステムにより逐次化されたプロセスやタスクとしてソフトウェアによりシミュレートされる。この実装は、プログラム書き換え回数を原理的に無制限にすべく、量産用の汎用的な機能を備えたプロセッサとして実現することを目的としている。しかし、組込みシステムのアプリケーションでは無制限なプログラムの書き換えは必須ではない。有限状態コントローラ内のステートマシンがプログラマブルであり、外界からの入力に対する処理や並行処理を直接ハードウェア回路として実装できれば、メモリセル上の仮想的

なステートマシンは必要ではなく、このシミュレーションのオーバーヘッドを排除することができる。すなわち、有限状態コントローラをシミュレータではない、文字通りのプロセッサとして使用することができ、消費電力とレイテンシは大幅に改善される。

従来の有限オートマトンに基づくストアードプログラムアーキテクチャはまた、次のようなオーバーヘッドを有する。MCU の LSI ダイのチップ面積の大部分は、状態遷移、割り込み処理、および並行プロセスをメモリアドレス操作としてシミュレートすべく、算術および論理演算リソース以外のメモリ制御に用いられる周辺回路によって占められており、その結果、実装面積の増大と消費電力の増加を招いている。外付けメモリ素子の制御が不要になれば、実装面積と消費電力は大幅に改善できる。

この問題を克服するために有効な 1 つの対策は、DRP を使用することである。DRP では FSM およびデータパスはこれらの切り替え機能を備えたハードウェア回路によって実装される。この切り替え可能は FSM の状態遷移は、動作合成ツールによりプログラム可能である。割り込み処理と並行処理は、ハードウェア回路として直接実装することが可能である。このため、応答速度の改善と、消費電力の大幅な削減が見込める。実際、参考文献[23, 24]では、DRP により同等の演算量をストアードプログラムアーキテクチャに基づく MCU で処理する場合に比べて消費電力が 1/100 に低減されることが報告されている。これは、高速な応答時間と低消費電力の要求される IoT アプリケーションに適した実装である。すなわち、ソフトウェアによってプログラムされる MCU の性能のボトルネックが DRP を実装することにより解消される。

キャッシュメモリを含むプログラムメモリや、メモリ制御回路を排除するためには、FPGA も MCU の代替候補となり得る。FPGA と MCU の違いは、EA に照らして演算デバイスとしての視点から見ると明らかとなる。FPGA は構造化されていないため、EA 計算モデルで示している切り替え可能な FSM とデータパスは含まれていない。また、FPGA には高級プログラミング言語のコンテキストトレース機能も同様に含まれていない。これらは FPGA の有するコンフィギュレーション可能な論理ブロック（Configurable Logic Block: CLB）ないしルックアップテーブル（Look Up Table: LUT）を用いてハードウェア回路として構成することになる。しかしながら、ハードウェア回路を構成する上で、CLB や LUT を使用した配置配線、およびルーティングの非効率性が指摘されている[29]。一方で、これらの非効率性があるにしても、ハードウェア回路で信号処理が可能であることから Wire rate でセンサ信号の処理が可能である。この FPGA の特長をうまく活用すべきである。

もう一つの問題は高位合成／動作合成技術に関するものである。高位合成／動作合成技術は近年、System on Chip (SoC)設計の生産性を向上させるために広く使用されており、EA 計算モデルにある有限状態コントローラに対応する PE を生成するには有望な技術であるが、高位合成／動作合成技術を使用して PE を設計するための手法は確立されているとは言えない。高位合成／動作合成技術により MCU を上回る効率を実現するプロセッサの設計手法が望まれる。MCU では、C 言語ないし他の高級言語のコンパイラは、与えられたユー

ザプログラムから実行可能なバイナリコードを生成する。このバイナリコードは、ターゲットとなるプロセッサの命令セット（Instruction Set Processor: ISP）を活用した逐次命令列から構成される。これらの ISP は、可能な限り多用途に設計されているものの、MCU の大量生産のために汎用的な演算に固定化されている。アプリケーションに適した専用演算のためのハードウェアマクロが実装され、動作合成ツールでこのハードウェアマクロを活用して PE を生成することができれば、固定化された ISP ハードウェアを前提とした従来のソフトウェアコンパイルよりも大幅に柔軟性が改善され、チップサイズと消費電力は大幅に改善可能となる。これを実現するためには、アプリケーションに最適化した専用の演算のためにハードウェアマクロを実装し、高位合成技術を用いてそのハードウェアマクロを活用して PE を生成することができなければならない。また、アプリケーションプログラムに内在する状態遷移を実行する FSM や、割り込み処理、および並行プロセスについても、一つ目の問題として識別したオーバーヘッドを解消するために、ハードウェア回路で実現する必要がある。このように、状態に応じて最適な演算器を接続するという DRP に相当する機能が高位合成／動作合成により実現できれば、実装面積と消費電力を削減することができるようになる。

3.3. Generic な PE アーキテクチャの提案

前節に述べた計算モデルの分析により、外付けメモリ素子を要せず、また、センシングデバイスに埋め込むのに適した PE のアーキテクチャは以下の特長を有することが要求として定義できる。

a) 高級言語でプログラム可能な MCU の利点を取り込む。

b) Wire rate processing が可能な FPGA の利点を取り込む。

c) 状態に応じて最適な演算器を接続できる DRP の利点を取り込む。

各プログラマブルデバイスにはそれぞれ克服すべき課題があり、それらの課題を克服しつつ、利点を取り込む必要がある。コンフィギュレーションメモリはナノブリッジを活用することにより削除可能であるから、PE のアーキテクチャとしては上記の 3 つのアプローチに基づき、外付けメモリ素子を活用したプログラムメモリを削除することが求められる。さらに、実装できるプログラムの容量に制約が生じないよう、スケーラビリティを確保することも重要である。

3.3.1. MCU の利点の取り込みと課題の克服

C 言語等の高級言語でプログラムすることができる MCU の利点を取り込むにあたり、外付けメモリ素子を要せずに取り込むための配慮を述べる。ここで着目するのは、図 3-2 に示す外部メモリである。MCU のアーキテクチャでは、プログラムとデータはこの外部メモリに格納され、Register に取り込まれた後、ALU で処理される。この Register と ALU の

組は関数で表現することとし、関数の引数を Resigter に、関数の本体を ALU に対応させる。この関数表現を記載したプログラムを高位合成技術、および動作合成技術により、ハードウェア回路として実装する。ナノブリッジを活用することにより回路の結線は書き換えが可能であり、高位合成技術を活用することからプログラムのソースコードは高級言語で記述することが可能である。

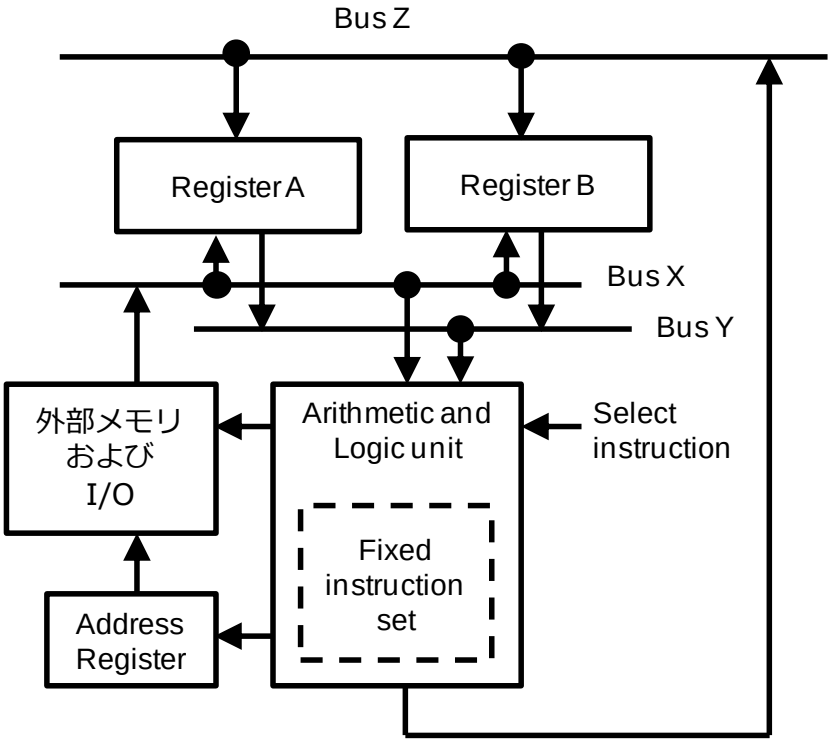


図 3-2　MCU のアーキテクチャ

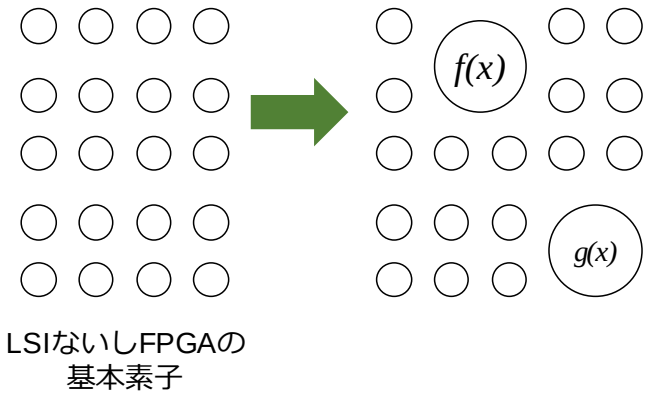


図 3-3　関数形式による実装

この実装形式の物理的な概念を図 3-3 に示す。LSI 下地にあらかじめ用意されている論

理演算素子や FPGA の LUT をナノブリッジによる金属架橋で接続する。関数をハードウェア回路として実装するため、プログラムを記述する高級言語は関数型言語が望ましい。本研究では C 言語を用いることにより高級言語でプログラムする MCU の利点を取り込んだ。外付けメモリ素子を要しないことにより、放射線などによるソフト・エラーも抑制できる。

現時点ではナノブリッジの配線数は先端的な FPGA に比べて未だ限られており、実装できる回路規模も制約がある。このため、一度ハードウェアとして構成した関数は極力活用したい。また、関数の定義自体も、アプリケーションの実装要求を反映しつつ、そのアプリケーション内で広く活用できるようなしくみが必要となる。このように関数を効率的に実装する技術はこれは高位合成技術、および動作合成技術に対する拡張要求となり、その内容については後述する。

3.3.2. FPGA の利点の取り込みと課題の克服

センシングデバイスの出力信号の伝送を中断せずに処理できるという FPGA により可能となる Wire rate processing の利点を取り込むにはナノブリッジを活用している。これは本研究に先立って実現されており、このアーキテクチャを図 3-4 に示す。

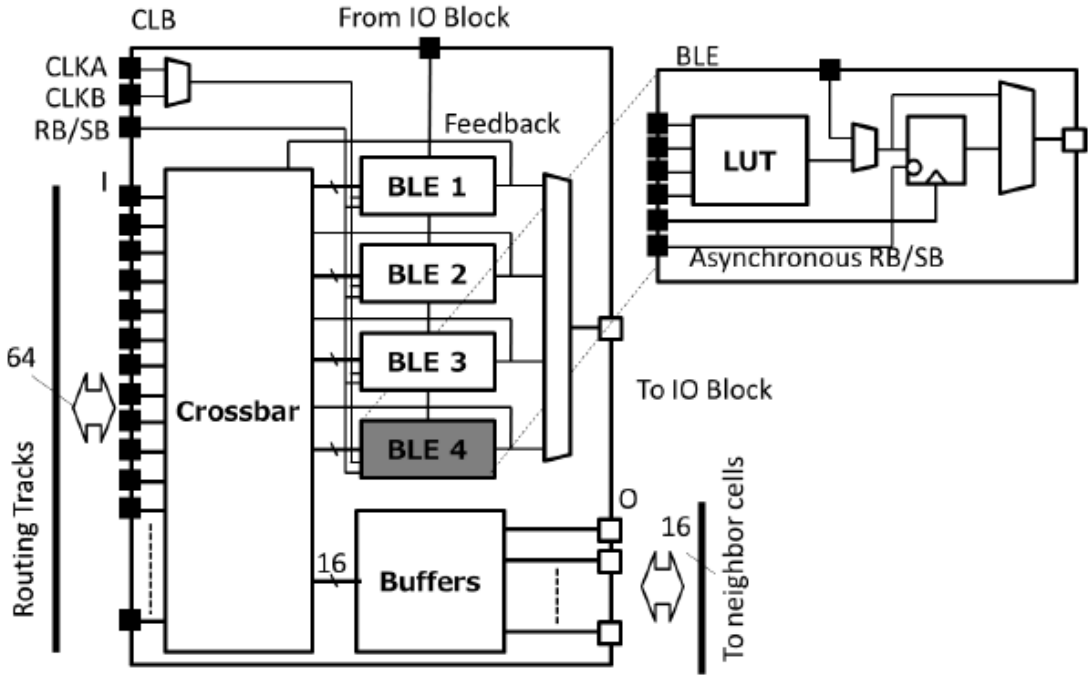


図 3-4　ナノブリッジを用いた FPGA のアーキテクチャ [3]

ここで、CLB は Configurable Logic Block、BLE は Basic Logic Element、LUT は Look Up Table、CLK は Clock、RB は Reset Block、SB は Set Block を示す。回路の結線情報を LUT のコンフィギュレーションデータとして設定するに当たり、一般の FPGA ではこの

データを SRAM、EEPROM、ないしは FLASH メモリなどのメモリ素子に格納している。ナノブリッジによりこの回路結線情報を金属架橋に置き換え、コンフィギュレーションメモリをなくした。これにより放射線などによるソフト・エラーも抑制できる。

商用化されている FPGA では、デバイスの大量生産を目的として Basic Logic Block は固定的なスイッチマトリックス上に配置する構造が採用されている。データパスを接続できるルートは予め固定されており、前述のように、ハードウェア回路を構成する上で、CLB や LUT を使用した配置配線、およびルーティングの非効率性が指摘されている[29]。BLE 間のあらかじめ定義されたデータパスは、動作合成を用いてレイアウトサイズの最適化と消費電力の削減を図る際の制約となる。このため、センシングデバイスにプロセッサを埋め込むためには、最適化された ALU と柔軟なデータパスが必要である。この非効率性を改善するために、高位合成を用いた動的再構成可能プロセッシングエレメント（PE）が提案されている[30, 31,32]。またデータパスのビット幅は 16bit や 32bit などのいくつかの値に予め固定されている。これに対して IoT アプリケーションで使用されるセンサは、8,12,14 ビットなどのさまざまなデータ幅を持つことから、柔軟なビット幅を有する PE が求められる。これら要求事項は従来の FPGA のアーキテクチャの改善を要し、後ほど議論する。

3.3.3. DRP の利点の取り込みと課題の克服

PE の状態に応じて最適な演算器を接続できる DRP は、柔軟なプログラミング能力を有しつつ、実装効率を向上させることも可能であるため、取り込むべき利点が多い。DRP の一つの実装例として FRRA (Flexible Reliability Reconfigurable Array)の例を図 3-5 に示す[17, 18]。FRRA では論理記述言語に加え、CyberWorkBench（CWB）[9, 10, 11]という安定した高位構成／動作合成ツールを使用して高級プログラミング言語である ANSI-C 言語および SystemC によりプログラミングできる。また、動作合成時には汎用 ALU、LUT、およびメモリクラスタの接続を柔軟に変更できる。このような工夫を施すことにより、FPGA に比肩する高速処理と低消費電力を追求する一方で、より高い抽象度を持つプログラムの作成を可能とし、複数 ALU の多数決構成も可能とすることで高い信頼性を兼ね備えている。

一方、FRRA を含む既発表の DRP では ALU として MCU に類似した汎用の演算器を複数実装している。これはプロセッサとしての量産を意図した汎用化設計である反面、ALU の利用効率が落ちると共に、汎用性を追及すると一つ一つの ALU の占有面積が増えるという課題を有している。ALU の利用効率を上げ、実装面積を減らすためには、アプリケーション毎に最適な演算器を必要なだけ並べる必要がある。この演算器構成の最適化の例を図 3-6 に示す。この例では、全体で 8 状態の演算処理を行う PE において、状態 4 および 5 においては加算器といった通常の ALU を構成する演算器と、粗粒度の演算器を 2 つ接続している。この粗粒度の演算器としては、画像処理におけるエッジ検出などの演算を充てることを想定する。このように状態に応じて最適な演算器を必要なだけ並べることにより演算

器の利用効率を上げ、実装面積を減らすことが可能となるが、このためにはアプリケーションに応じて実装する粗粒度の演算器を活用して動作合成する機能が高位合成／動作合成ツール側に必要となる。これについては後述する。

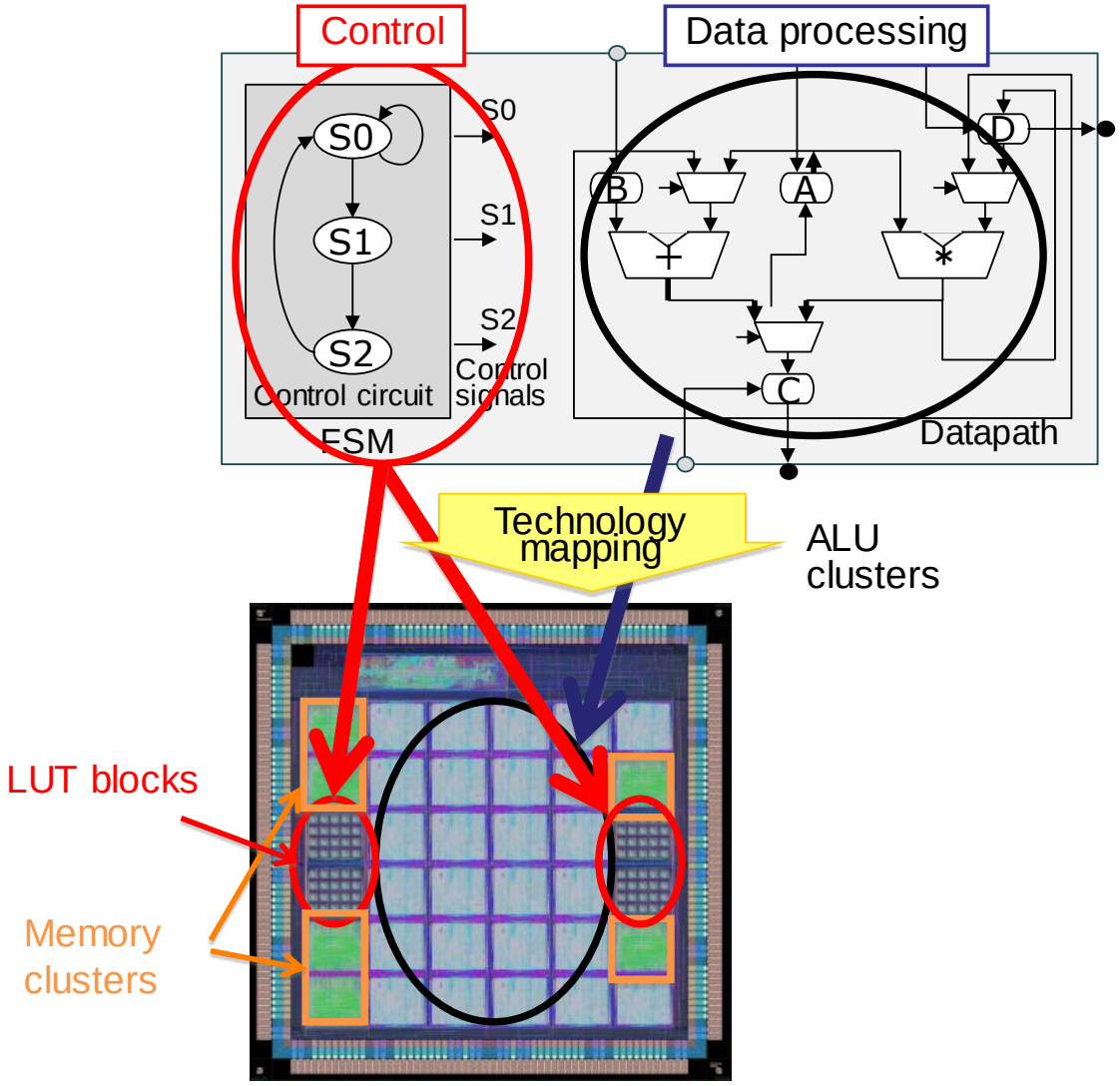


図 3-5　DRP の実装例（FRRA）[17, 18]

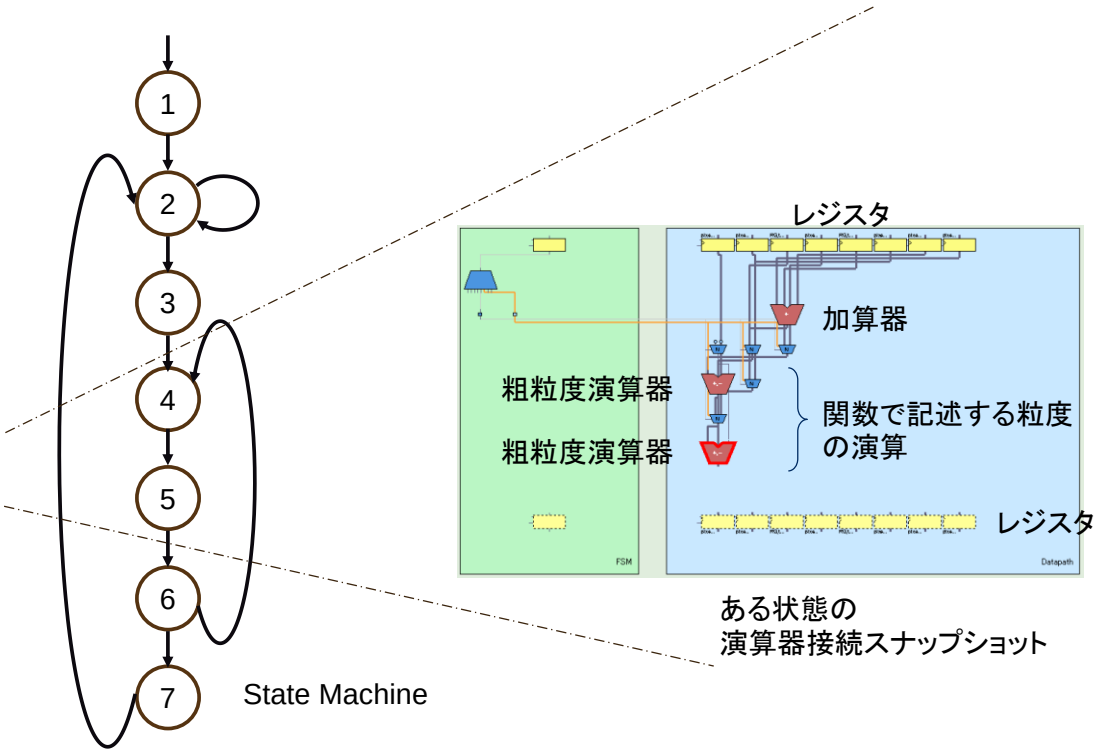


図 3-6　演算器構成の最適化

DRP の利点を融合するメリットを、FRRA による実装例を元に、MCU および FPGA と対比させて述べる[17, 18]。図 3-7 には演算サイクルの削減効果の例を示す。同図に示した C 言語ソースプログラムは一般的な MCU の ISP を用いると図 3-7(a)に示すように 5 サイクルの演算を要する。これに対し、DRP では図 3-7(b)に示すように 2 サイクルの演算で処理を終える。これは演算処理速度のみならず、消費電力の削減に直接的な効果を有する。図 3-8 には実装面積の削減効果の例を示す。図 3-8(a)の FPGA では演算器とデータパスをフラットに展開しており、図 3-8(b)に示す DRP による実装は演算処理サイクルは 1 サイクル増加するものの、実装面積は削減できる。演算処理全体の状態数が増えるほど、演算器は更に使い回しが可能となり、実装面積の更なる削減が見込まれる。図 3-7 に示す例でも DRP は外付けメモリ素子を有しないことから実装面積の削減効果も得られており、これらを合わせることにより、[23, 24]で報告されているように、従来の MCU と比較して 1/100 にも及ぶ低消費電力化が可能となるケースも出てくることになる。これは、チップ面積の大部分が計算リソースとして使用できるためである。

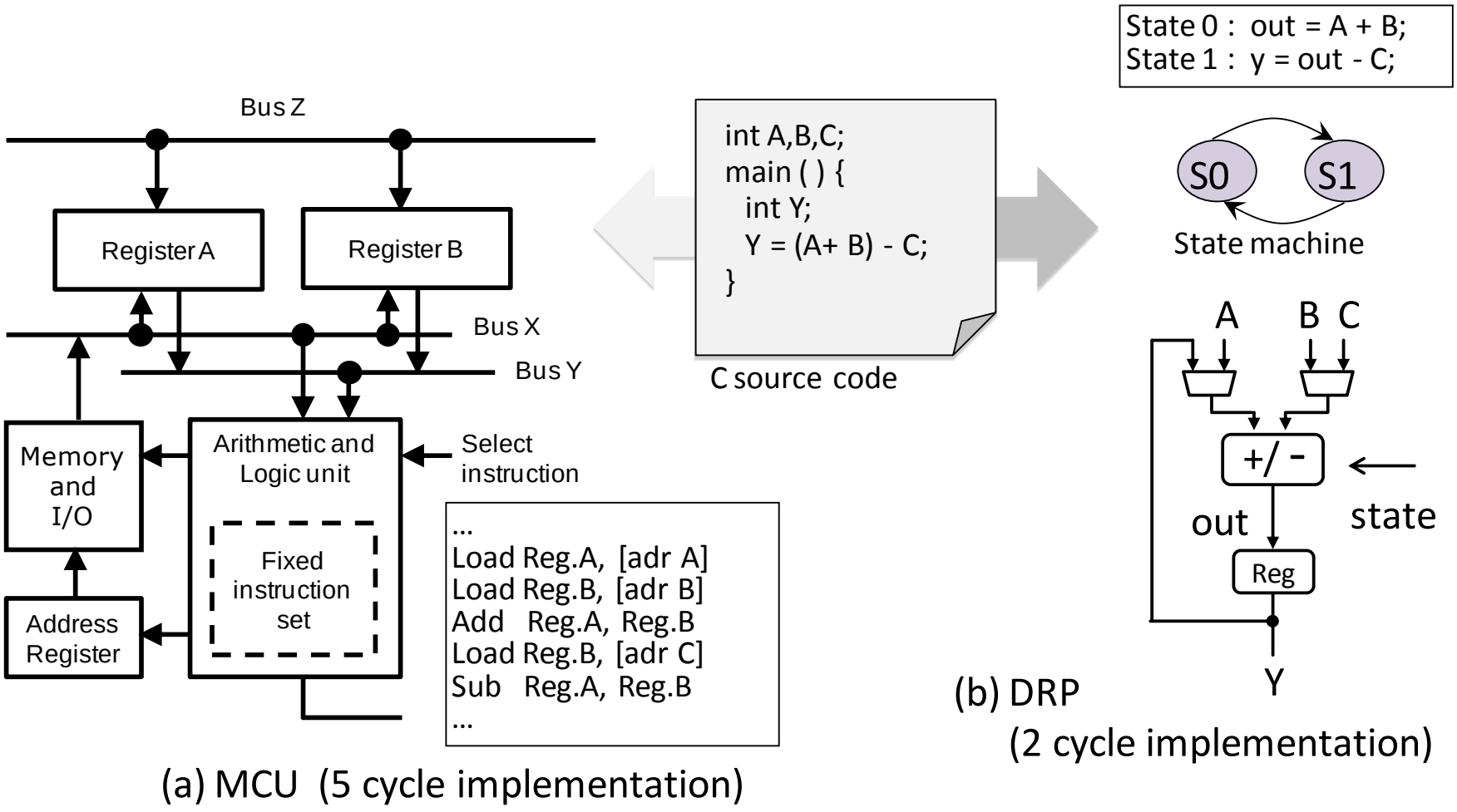
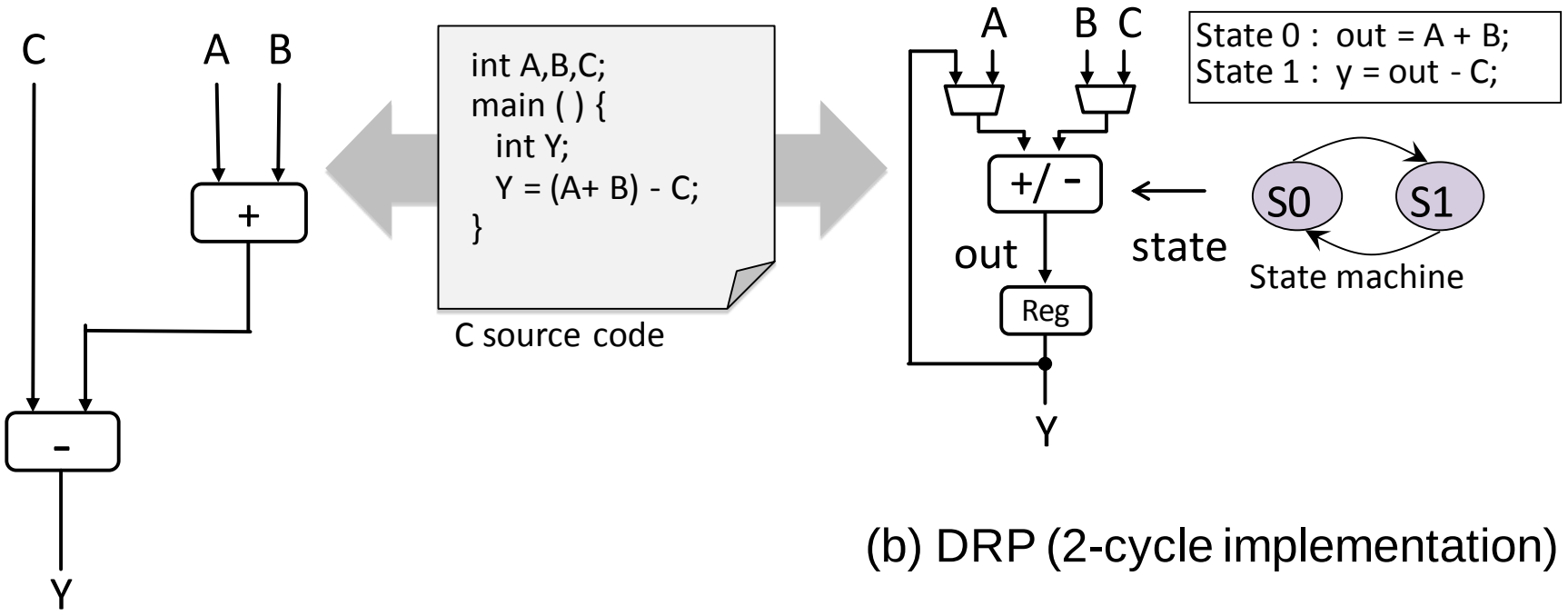


図 3-7　DRP による演算サイクルの削減例



(a) FPGA (1-cycle implementation)

(b) DRP (2-cycle implementation)

図 3-8　DRP による実装面積の削減例

3.3.4. Generic Processing Element (GPE)

3.2 節に述べたように、従来の Finite Automaton に基づくストアードプログラムアーキテクチャを有する MCU について、新たに提案する計算モデルである EA を参照することにより、二つの解決すべき課題を識別した。これらを以下に再掲する。

a) 外付けメモリ素子内に格納されたアプリケーションに内在する状態遷移コントローラを、有限状態制御部内（すなわち MCU の演算器内）の状態遷移コントローラでシミュレーションするオーバーヘッドの解消

b) 演算器の定義に対する高位合成技術および動作合成技術の活用

これらについて対策を講じると共に、更に 3.3.1～3.3.3 節に示すプログラマブルデバイスの利点を取り込んだ Processing Element (PE)を提案する。これは設計上の工夫として実装することも可能であるが、開発ツールに組み込むことができれば、センシングデバイスにプロセッサを埋め込む PE を設計するための汎用的な設計手法として活用することができる。このためには PE の基本的なアーキテクチャを定義し、その定義内容を高級プログラミング言語で指定できるようにする必要がある。

本研究では PE のアーキテクチャを策定するに当たっては、C 言語などの関数型言語で全体を設計できるプログラミング環境を維持することを目標とした。また、ソースコードのデバッグを行うに当たっては、デバッグツールはアプリケーションプログラムから抽出されたコンテキストに沿ってプログラムを実行することができる必要がある。すなわち、関数型言語で記述された一連の操作のコンテキストをトレースできる必要がある。

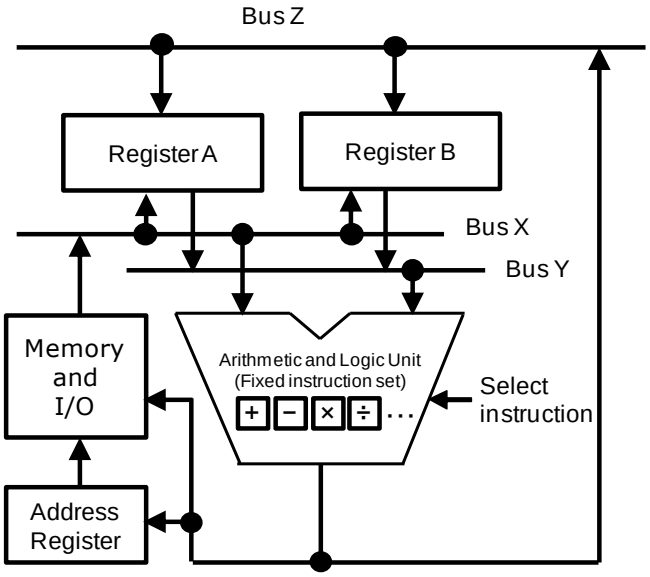
本研究で提案する PE のアーキテクチャは、意味論ないしラムダ計算に基づいて高級プログラミング言語のコンテキストを表現し、2 つのタイプのコンテキストを表現できることとした。 1 つは、ファイルに格納された情報として言及されることの多い、置き換え可能な静的コンテキストであり、もう 1 つは、ヒープレジスタに格納された情報として表わされる動的コンテキストである。これらを関数記述として表現できるようにした。

アプリケーションの動的コンテキストは、以下のように関数で表現される。

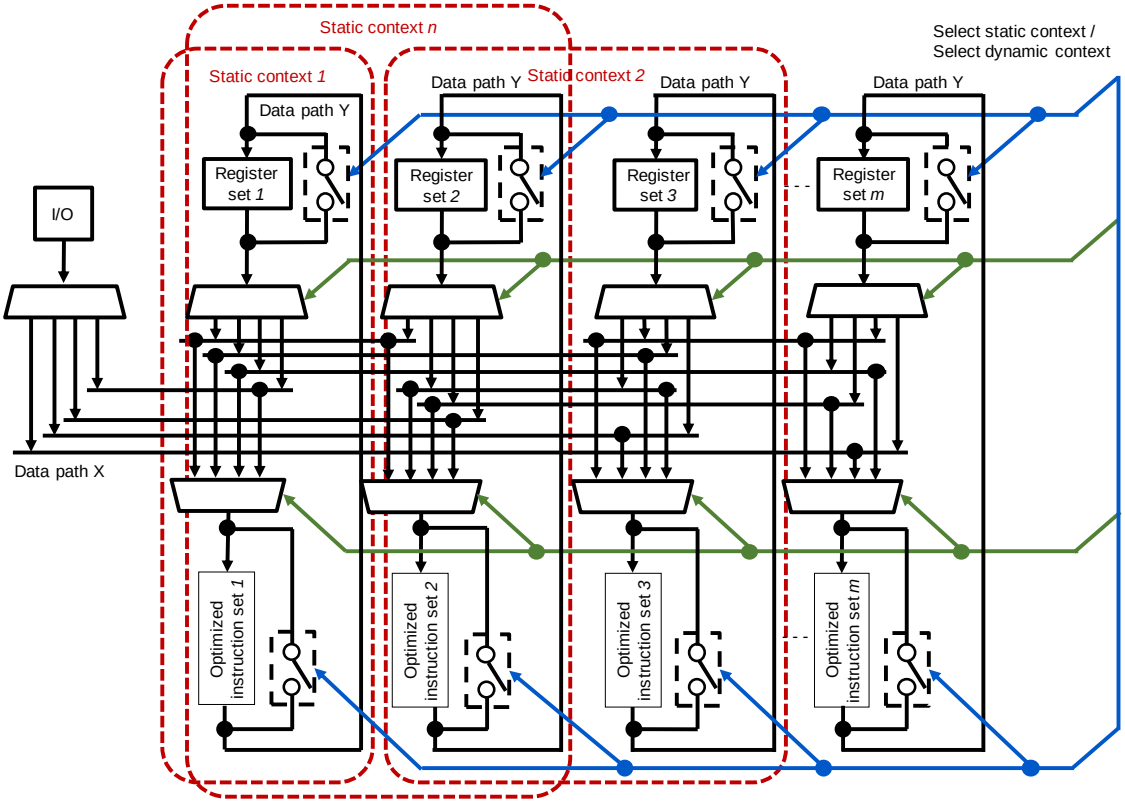
$$f(A), f(B), \ \ldots \qquad (1)$$

引数は Register に対応させ、関数本体はアプリケーションに応じて最適化された Instruction Set を有する演算器として実装する。アプリケーション要求に沿った m 組のレジスタと演算器の組を定義し、各組は状態に応じて必要なレジスタと演算器をグルーピングする。これらのグループ同士の接続は以下のように関数で記述できることとする。これは関数間の関係をバイパススイッチで切り替え選択することとして定義した。

$$f\left(g(\ \cdot\ )\right), f\left(g(\ \cdot\ ), h(\ \cdot\ )\right), \ \ldots \qquad (2)$$



(a)



(b)

図 3-9　PE の比較（a）MCU の PE（b）Generic Processing Element (GPE)

このPEはデバイス製造時に予め作り込むものではなく、高位合成／動作合成ツールで実装できるようにすることからGenericなものであり、Generic Processing Element (GPE)と呼ぶ。MCUのPEとGPEの比較を図 3-9に示す。図 3-9（a）はMCUのPEを示し、図 3-9（b）はGPEの実装を示す。Instruction Setはアプリケーションの要求に基づいて最適化された演算器として実装され、これらの最適化された演算器をそれぞれの static context間で共有することができるものとし、これを指定できる機能は高位合成機能に用意されている機能を用いる。これについては後述する。

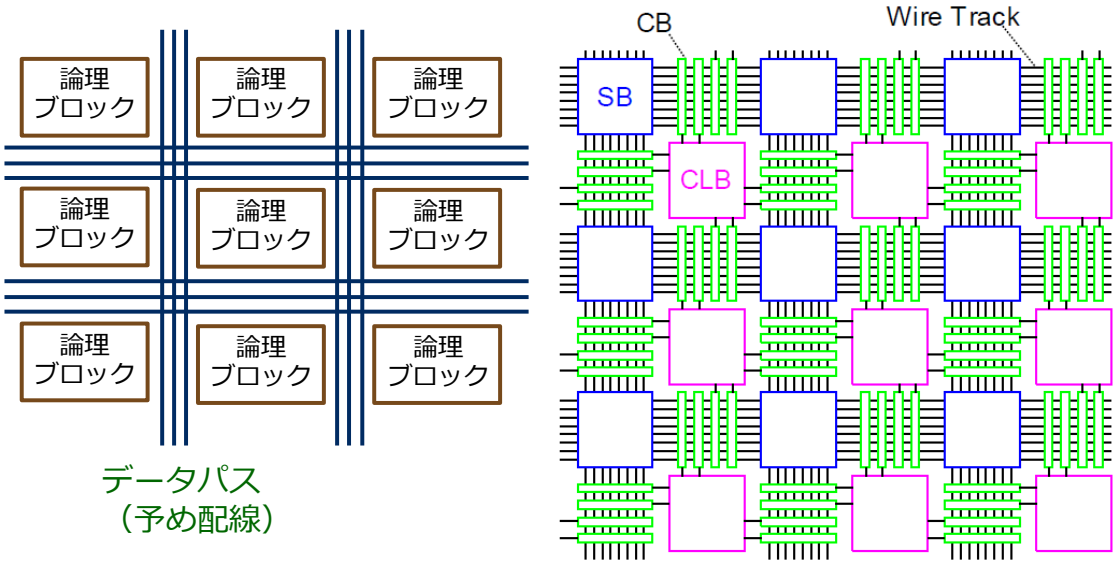
GPEではアプリケーションの静的コンテキストを関数間の接続として図 3-9 (b) に示すように関数を選択するバイパススイッチとして表現している。静的コンテキストの切り替え機能を実現するためには、高位合成により生成されるステートマシン・コントローラを用いる。高位合成では3.3.3節に述べたようにアプリケーションに応じたFSMがプロセッサ内に自動的に生成される。FSM内の各状態にそれぞれの静的コンテキストを対応付ければ、状態が進むにつれて各状態に対応した関数の選択が行われる。GPEでは状態に応じた静的コンテキストである各関数グループの切り替えは、図 3-9 (b) 内の select static context信号および select dynamic context信号により行われ、これらの select static context信号と select dynamic context信号はプロセッサ内に実装されたFSMから出力される。この動きはDRPにおけるコンテキスト切替えと似ている。これはすなわち、GPEではアプリケーションに応じて最適なDRPを都度生成していることに相当する。

MCUでは、アプリケーションに固有のFSMはメモリセル上に実装されているため、プロセッサ内のFSMはアプリケーション内のFSMとは異なる。これに対してGPEではアプリケーションに応じたFSMは実際のPEのハードウェア上で直接実行されるため、MCUと比較して消費電力も改善される。これは、MCUが外付メモリ素子のアドレッシング機能によってメモリの読み書きをすることによってアプリケーションに応じたFSMをシミュレートするため、そのオーバーヘッドが余分に掛かるためである。

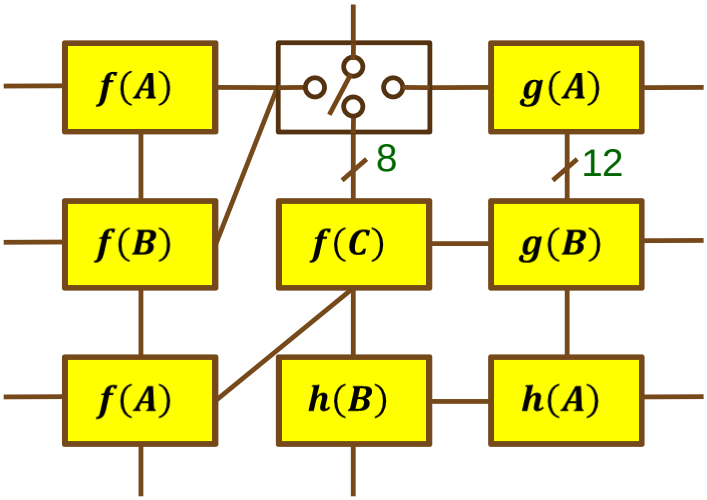
上述のようにGPEの具体化にはDRPに良く似たしくみを用いているが、現状のDRPは高位合成の観点からみると制約がある。この制約は現行のMCUやFPGAにも見られるものであり、具体的には図 3-10に示すように、各論理ブロックがメッシュ状のデータパスによって規則的に配置配線されており、これは高位合成ツールが自在な配置配線を行う上での制約となる。図 3-10 (a)に示すのは従来のMCU、FPGA、およびDRPの例であり [94]、CLB (Configurable Logic Block, 論理ブロック)、SB (Switch Block, 配線接続スイッチ)、CB (Connection Block, 入出力スイッチ)、およびWire Trackから構成される[94]。CLBは可変論理であるLUTと順序回路であるFlip-Flop (FF)から構成され、SBは縦横の配線接続の切り替え、CBはCLBと配線の接続、およびWire TrackはSB間の配線に用いられる。これに対して、EAの解析からアーキテクチャを策定したGPEでは、PE間の接続は隣接PE間とし、全体的な配置配線制約は課さない。この様子を図 3-10 (b)に示す。このよ

うに GPE では予めデータパスを固定しないことから、関数として表現された論理ブロックを自在に接続可能である。また、バイパスのみの関数も定義可能とすることにより、高位合成ツールに対する制約を回避している。演算器のビット幅も任意としているが、これは予め汎用の演算器を用意せず、高位合成ツールにより実装時に確定することを前提としているためで、これにより実装効率を高めることが可能となる。また PE を動的に増やしていくことも容易であり、各 PE は Appendix C に示すように複数の処理をすることもできる。プロセッサエレメントのスケーラビリティを図 3-11 および図 3-12 に示す。図 3-11 にはスケーラブルなプロセッサエレメント・モデル同図(a)の構成要素の論理モデルに示すように、アプリケーションに応じた演算器と、外界からの入力に応じたステートマシンを各 PE が内蔵している。同図(b)にはスイッチ、およびバイパス処理の論理モデル演算器の出力はコンテキストに応じて出力ポートを選択し、必要に応じて演算器をバイパスすることも可能としている。これにより、通常の FPGA 等に用いられている配置配線スイッチ（Switch Block）、入出力スイッチ（Connection Block）、および Wire Track が不要となり、LSI のチップ全面を演算リソースで埋め尽くすことも可能と見込まれる。これらの PE は図 3-12 に示すように原理的に N 重接続が可能であり、第 5 章に述べる冗長構成方式を併用することにより、ネットワークをダイナミックに増やしていき、複数の処理をする方式も可能となる。

図 3-11 に示すプロセッサエレメントに基づき FPGA を構成する場合の例を図 3-13 に示す。同図(a)は従来の書き換え可能な FPGA の Logic Element であり[93]、(b)は図 3-11 のプロセッサ・エレメントモデルである GPE を適用した場合の FPGA の Logic Element である。従来の書き換え可能な FPGA の Logic Element の出力は Wire Track に接続することを前提としていることから 3-state buffer となっている。これに対して GPE に基づく Logic Element の出力は 3-state buffer である必要は無く 2 値論理出力でよいことから配置・配線ツールへの制約を抑制できる。また、2 段の MUX を介してパイパスが可能であることから、Logic Element としては隣接同士の接続となるものの、論理的には離れた Logic Element とも直接接続が可能である。遅延時間の増大等のペナルティが想定されるが、本研究ではセンサ信号処理を主眼としたデバイスコンピューティング用途として、前述した LSI のチップ全面を演算リソースで埋め尽くせる可能性に重点を置いた。



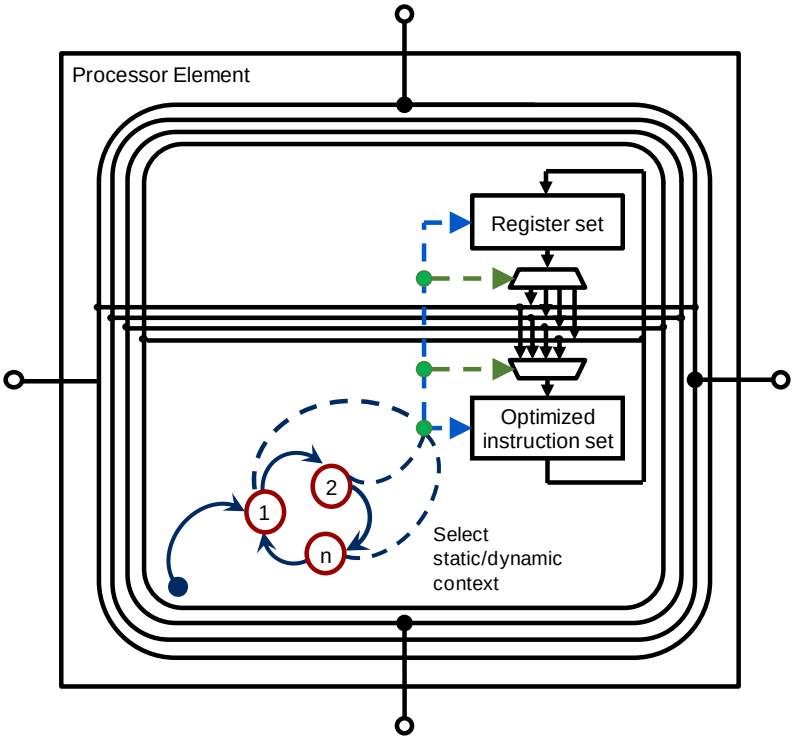
(a)



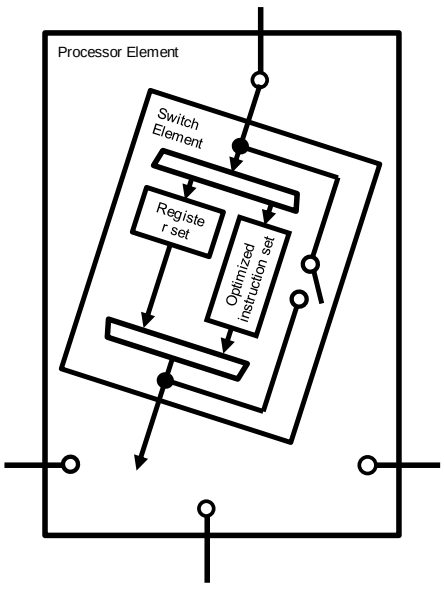
(b)

図 3-10 論理ブロックの配置配線

(a) 従来の MCU, FPGA, DRP の例、(b) GPE による論理ブロック接続

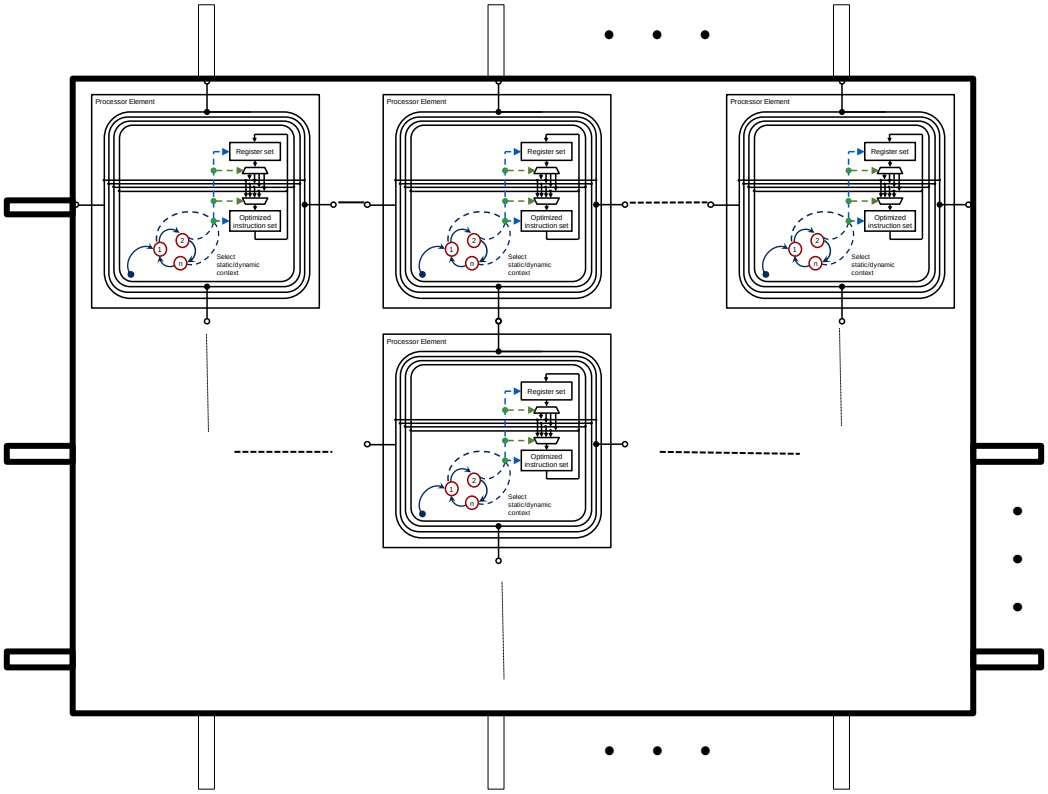


(a)



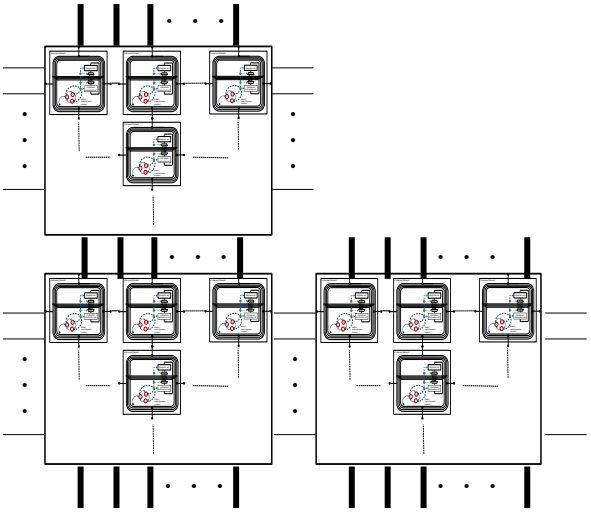
(b)

図 3-11　スケーラブルなプロセッサエレメント・モデル
(a) 構成要素の論理モデル、(b) スイッチ／バイパス処理の論理モデル



(b)

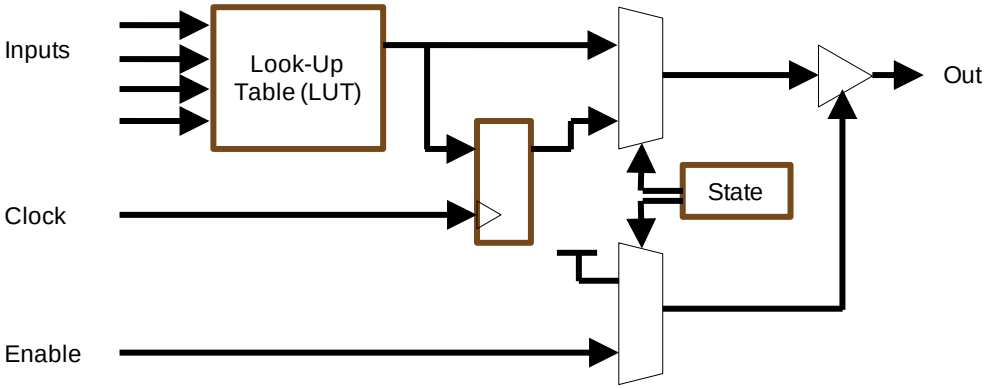
(a)



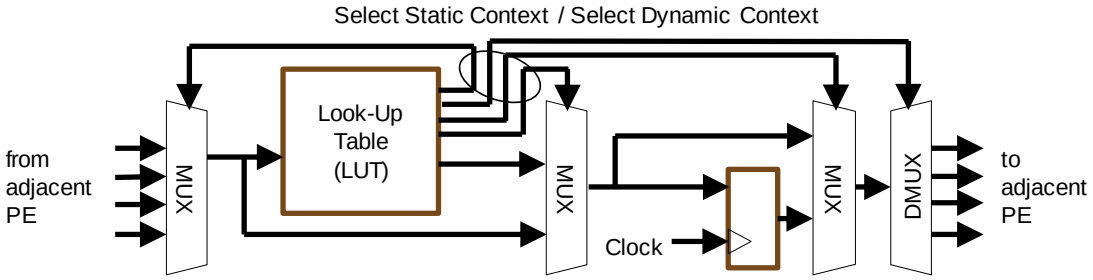
(b)

図 3-12　プロセッサ・エレメントのスケーラビリティ

(a) LSI 内スケーラビリティ、(b) LSI 間スケーラビリティ



(a)



(b)

図 3-13 本研究によるプロセッサ・エレメントモデルに基づく FPGA 構成例
(a) 従来の書き換え可能な FPGA の Logic Element [93]
http://www.ecs.umass.edu/ece/tessier/courses/636/
(b) GPE によるプロセッサ・エレメントに基づく FPGA の Logic Element

3.4. 本アーキテクチャの評価

本研究により新しく提案した計算モデルである EA に基づく検討から、新たなプロセッサアーキテクチャとして提案した GPE は従来の有限オートマトンでテープとしてモデル化されている外付けメモリ素子は不要となる。ナノブリッジを活用することにより回路配線レベルでの書き換えが可能となることから、高位合成技術を適用することにより、高級言語でプログラムすることを可能としつつ、プログラムメモリが不要となる。これは、IoT アプリケーションにおける外付けメモリ素子のソフト・エラー発生確率を低減し、高信頼性要求を満たすために望ましい特性である。またプログラムメモリに関連したメモリコントローラ等の周辺回路をチップ上に実装する必要が無くなり，チップ面積の大部分に演算リソースを配置できることから，従来 DRP を活用する際のメリットとなっていた低消費電力化も享受できることになる。従来の Finite Automaton に基づくリニアなアドレスで指定されたメモリアドレス空間を活用した状態遷移のシミュレーションはもはや必要とされない。プログラムメモリに関連するペリフェラルはもはやプロセッサチップ上に必要でなく、ほとんどの領域が計算資源によって占有されるため、消費電力も低減される。

演算セットの定義自体が設計プロセスとなることは本アーキテクチャの特徴である。この仕様はハードマクロを作り込むことを前提としており、高位合成技術を最大限に活かすことになる。最適化された ALU は実装効率の向上に資すると共に、デファクトスタンダードとなっている特定の ISP によるマイクロプロセッサ・アーキテクチャに依存しなくなることから、プロセッサ設計の自由度を高めることができる。デファクトスタンダードとなっている ISP を有する PE を混載することも可能であるから、既存の設計資産を活用することもできる。PE 自体は常に技術革新を反映した変更を行うことが可能であり，高位合成のフレームワークで記述するプログラムに関する高級言語記述に影響を与えないようにすることが可能と見込んでおり，このスキームによりプログラムの互換性を図り、設計資産を継承することが可能となる。

本仕様に基づくプロセッサアーキテクチャにとって互換性を維持する必要があるのは ISP ではなく PE 間の入出力通信プロトコルである。ここではハードウェアのみで実装が機能であり、オープンな国際標準規格である SpaceWire 等を候補として考えている。これにはオープンスタンダードである SpaceWire [59, 60, 61, 62, 63, 64]の採用等が有望と考えている。各 PE の入力と出力が互換性を維持する限り、それぞれの PE の設計は独立しているため、最新のコンピュータ技術を適用することができる。

GPE に基づく PE 上に実装するシステム性能に関しては，時刻と入力信号がハードウェア回路で直接処理されるため、従来の MCU で使用されている時刻割り込みやソフトウェア割り込みハンドラの実装を使用した時刻の仮想化は必要ない。また，プログラムはハードウェアで並列実装が可能なため並列処理をマルチタスク OS やマルチプロセス OS で処理をシリアライズして仮想化する必要が無い。このため，バリア同期を粗粒度ないし細粒度で

実装することにより並行リアルタイムシステムを仮想化せずに実装することを見込んでいる。

プログラムの書き換え回数などには制約が生じるが、エンタープライズシステム向けのプログラミングとは異なり、組込みシステムアプリケーションでは問題にならない。組込みシステム製品を開発するにあたっては、出荷前に数回のプログラム修正ができれば十分である。なぜなら、組込みシステム用プロセッサの設計は、製品ロットごとにしばしば変更されるからである。無制限のプログラム修正機能は IoT アプリケーションでは不要であり、コスト、サイズ、消費電力、および信頼性が優先する。この組込みシステムに特有の要求に応えるためには MCU と FPGA が統合されることが期待される。なぜなら、柔軟なプログラミング機能と効率的な実装はどちらもシステム構築に必要であるためである。MPU と FPGA は GPE により統合することができ、また GPE による PE の設計手法は System on a Chip (SoC)の設計にも有効と見込まれる。GPE の設計手法は高信頼性を有する動的再構成 VLSI (Very Large Scale Integration)と整合性が高く、これについては第 5 章で述べる。

GPE で実装される 2 つのタイプのコンテキストは高級言語で書かれたプログラムのコンテキストに従うことができ、そのコンテキストに沿ったデバッグ機能の実装が可能と見込まれる。このことから、従来の MCU のソースコードデバッガと同じデバッグツールを実現することが可能と見込まれる。

本研究にて提案した GPE を従来の MCU、FPGA、および DRP と比較した評価結果を表 3-1 に示す。

表 3-1　センシングデバイス向けプロセッサアーキテクチャの比較

	メリット	デメリット	備考
Micro-controller Unit (MCU)	高級言語でプログラムする。	ソフトエラーに弱い主記憶メモリ	誤動作リスク
	最適化されたALU (Arithmetic logic operation unit)	ALUは汎用機能に限定	プログラムはALUの機能のみで記述
		主記憶に機械語を配置してアプリケーションをシミュレートするオーバーヘッド	所謂Von Neumannボトルネック
Field Programmable Gate Array (FPGA)	Wire rate processing	回路面積の増大	機能制約
	種々の機能を実装可能なALU	ソフトエラーに弱いConfiguration Memory	誤動作・デバイス破損リスク
Dynamically Reconfigurable Processor (DRP)	状態に応じて必要な演算器を接続	ソフトエラーに弱い主記憶とConfiguration Memory	誤動作・デバイス破損リスク
	最適化されたALU	ALUは汎用機能に限定	プログラムはALUの機能のみで記述
提案アーキテクチャ	**高級言語によるプログラマビリティ**	**プログラム容量に制約があるが実用上問題ない。**	**容量を増やすチップ間接続方式が今後の研究課題**
	Wire rate processingの演算速度	**MCUを上回るがASIC/FPGAとは同等**	**プログラマビリテイを維持した速度**
	センシングデバイスに埋め込める小型化	**現時点では複合モジュール**	**デバイスとして一体化する技術が今後の研究課題**
	チップ面積の活用率拡大による低価格化	**配置配線を自動化するにはツールの改造が必要**	**配置配線ツールの適応が今後の研究課題**

第4章
新しいプロセッサエレメントアーキテクチャを実現する高位合成拡張

本章では、センシングデバイスに埋め込むプロセッサを設計するために新たに提案する設計手法について述べる。前章に述べた GPE に基づいたプロセッサを設計するに際して、本研究では粒度別階層化設計手法を提案している。これにより現状の高位合成技術の課題が明らかとなり、それを解決すべく高位合成拡張機能と対応付けた結果、前章で提案している GPE を効果的に活用できるようになったことを示す。これらを受け、GPE アーキテクチャを用い、階層化設計手法を統合した設計フローを提案する。

4.1. 粒度別階層化設計手法

前章にて定義した EA に基づく組込み用 PE である GPE を実装するに当たり，4 階層の構造を定義した。これらは、下層から上層に向けて順番にインタフェース回路層、細粒度定義層、粗粒度関数定義層、およびパイパス接続層から構成される。以下、これらの各層について説明する。

最下層のインタフェース回路層はランダムロジックやミックスドシグナル（アナログ・デジタル混在）回路を用いて実装する入出力インタフェース回路である。各入出力インタフェース回路はその上位層となる細粒度定義層の各構成要素に接続して使用する。

細粒度定義層を構成するのは、主としてデータパスと Finite State Machine (FSM)である。FPGA を用いる場合には FSM を再構成可能な LUT を用いて構成することもできる。FSM とデータパスは、高級プログラミング言語で記述されたコンテキストに沿って処理を進めるために、切り替え可能とする。この設計は CyberWorkBench (CWB)[9, 10, 11] のような近年、技術的に成熟してきた高位合成／動作合成ツールを使用して記述することにより、きめ細かな設計をすることができる。2 種類のコンテキストの切り替えに用いる setect static context 信号と select dynamic state 信号は GPE の特徴的な入力信号であり、細粒度定義層で定義する FSM から出力される。CWB により高位合成される回路のサイズは、VHDL (VISIC Hardware Description Language)や Verilog 等のハードウェア記述言語（Hardware Description Language: HDL）で直接実装した場合と比較してオーバーヘッドは数パーセントに留まると報告されており、CWB などの高位合成ツールを使用して生成された FSM と Data Path は細粒度定義層の設計を実装するにあたって十分効率的である。

細粒度定義層の上位層として粗粒度関数層を定義する。この層では対象とするアプリケ

ーション向けに最適化された関数定義に基づく演算器を高位合成することを主目的としている。この層には効率的な操作を可能とする基本演算機能を備えた専用の算術論理ユニット（Arithmetic Logic Unit: ALU）などの演算ユニットを含んでも良い。この粗粒度関数定義層にて定義された単位の演算ユニットは組込みシステム用の MCU における ISP に相当し、操作命令（Opcode）と同等に扱うものとする。例えば、粗粒層関数定義層の演算単位として信号処理で用いられる FFT (Fast Fourier Transform)や画像処理で用いられる画像圧縮演算を定義し、高位合成により演算器を生成すれば、プログラムのソースコード内の独立した演算ニーモニックとしてこれらを指定でき、その機能を ALU の命令として利用することができる。これらの演算器は CWB のような高位合成／動作合成ツールを用いて生成しても良いし，通常のカスタム ASIC（特定用途向け集積回路: Application Specific Integrated Circuit)の開発に用いられる設計プロセスとツールを用いて最適化したハードマクロ IP として実装してもよい。さらには、後述するように FPGA の LUT を使用してハードマクロとして実装しても小型化・低消費電力化が可能であることが本研究により判明している。

最上層では各 PE を接続する。各 PE の接続は各 PE を n 次元配列したルータとみることができる。この n 次元ルータの実現方法は種々考え得るが，なるべく高位合成の制約にならないよう，PE 自体をバイパススイッチとして指定することも可能とした。バイパススイッチでは、PE の入力と出力間の接続が任意に設定可能である。これは粗粒度層上のルータを実現するのに便利であり、n 次元バイパススイッチにより粗粒度機能ブロック、細粒度機能ブロック、およびインタフェース回路ブロックのグループを作成することで、高位合成機能に制約を与えることなく n 次元ルータを生成できる。

各階層の回路はナノブリッジを活用することにより，従来の書換え可能な FPGA で用いられるようなコンフィギュレーションメモリを使用せずに、物理的にプログラマブルな論理回路として実現することができる。また、これらの階層設計を通じて構成する PE である GPE は、前述した EA により計算可能性が示され、プログラムメモリを要しない組込みプロセッサとして実現できたことになる。これらプログラムメモリとコンフィギュレーションメモリの双方を要せずにすむことから、宇宙放射線や自然放射線によって引き起こされるソフト・エラーを避けることができ、IoT 用センサが配置される屋外環境での信頼性を高めることが可能となる。階層化設計手法が参照する階層構造を図 4-1 に示す。

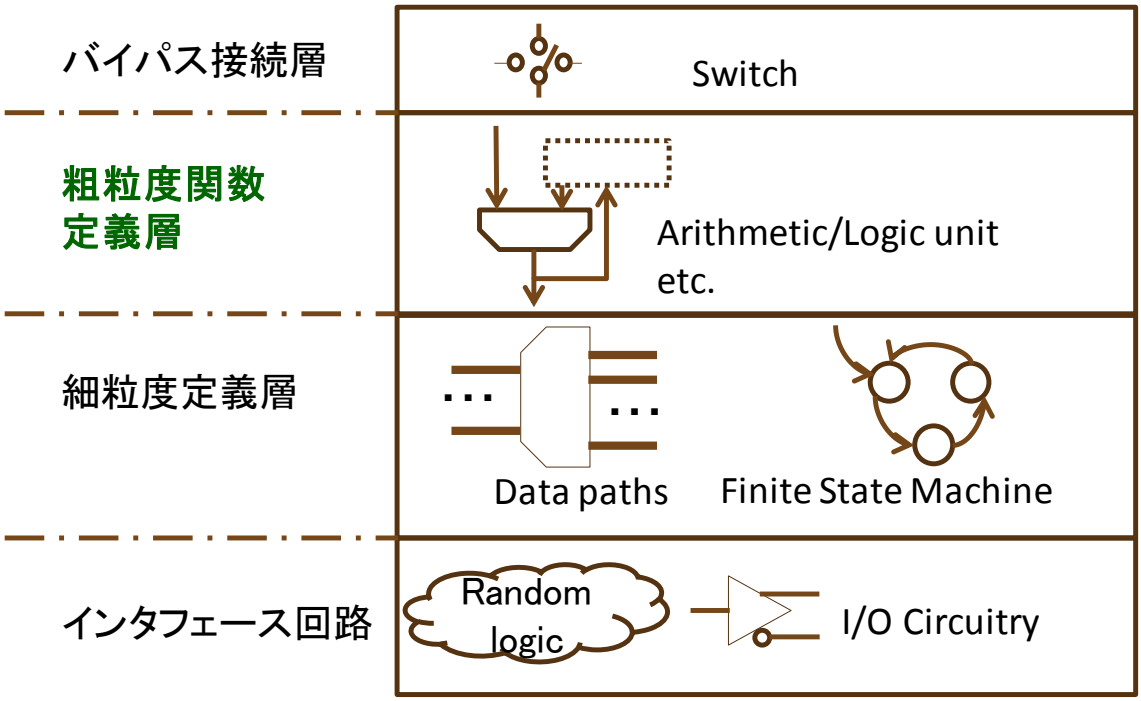


図 4-1　階層化設計手法の参照モデル

この階層化設計手法を策定することにより、高位合成と DRP の双方に課題が識別された。一般に高位合成が対象としているのは細粒度定義層までであり、粗粒度定義層は ALU や専用演算器などが予め用意されていることが前提である。しかしながら、本研究にて提案する GPE を活用し、階層化設計手法で PE を実現するに当たっては、アプリケーションに適した処理機能を関数表記し、関数本体を演算器としてハードウェア実装し、レジスタとセットにして活用する必要があり、これらが動作合成機能によりハードウェア回路に実装できるように予め高位合成して HDL に落とせるようにしなければならない。すなわち、高位合成の枠組みの中に粗粒度関数定義層対応の機能を追加し、定義された関数や ALU を動作合成で再度活用できるようにしなければならない。また、DRP の実装についても、粗粒度定義層に対応する ALU として MCU の ALU と同様な汎用の ALU を複数並べることが一般に行われている。これは DRP を量産するために汎用化設計することに起因しているが、反面 ALU の利用効率の悪化を招き、センシングデバイスに埋め込めるような小型化、および低消費電力化を実現するための妨げとなる。

このように識別された高位合成と DRP の課題に対処すべく、高位合成ツールの対象範囲に粗粒度関数定義層を追加して粗粒度関数定義層で定義した関数を実装した演算器をデータベース化し、高位合成にてこのデータベースから最適な演算器を選択し、動作合成で粗粒度の演算器定義をプリミティブな処理として使い回す機能拡張を施した。この拡張には、CyberWorkBench の開発チームの協力を得た。このしくみを図 4-2 に示す。高位合成にて

最適な演算器を選択する機能はバインディングと呼ばれ、これは NP 困難な処理とされている。GPE ではプログラムソースコード中にバインディングを明示的に指定する記述ができる。前章に述べた GPE を参照することにより、これには二種類の仕組みが必要であることがわかる。ひとつは各レジスタと演算器のセットのグルーピングおよび接続関係を指定する機能であり、もう一つは各グループ間で可能な限り演算器を共有できる機能である。これらを CWB 上に C 言語のコメントの書式を活用したプラグマ的な機能として実装している。この記述方法についは後述する。DRP の抱える課題については、DRP を構成する ALU を動作合成で生成できるようマクロライブラリを整備することで対処した。これによりアプリケーションに適した関数機能に対応する演算器を状態に応じて必要な種類の演算器を必要な数だけ接続できるようにした。これらの拡張機能を開発した結果、メモリを要しない PE をセンシングデバイスに埋め込めるほど小型化、低消費電力化することが可能となった。なお、DRP を構成する ALU は一般に LSI 設計レベルで最適化することが多く、高位合成で生成した演算器を活用することについてのメリットは懐疑的であった。しかしながら、後述するように市販の FPGA を活用した試作評価についても小型化の効果が確認され、本手法が有効であることが確認できた。

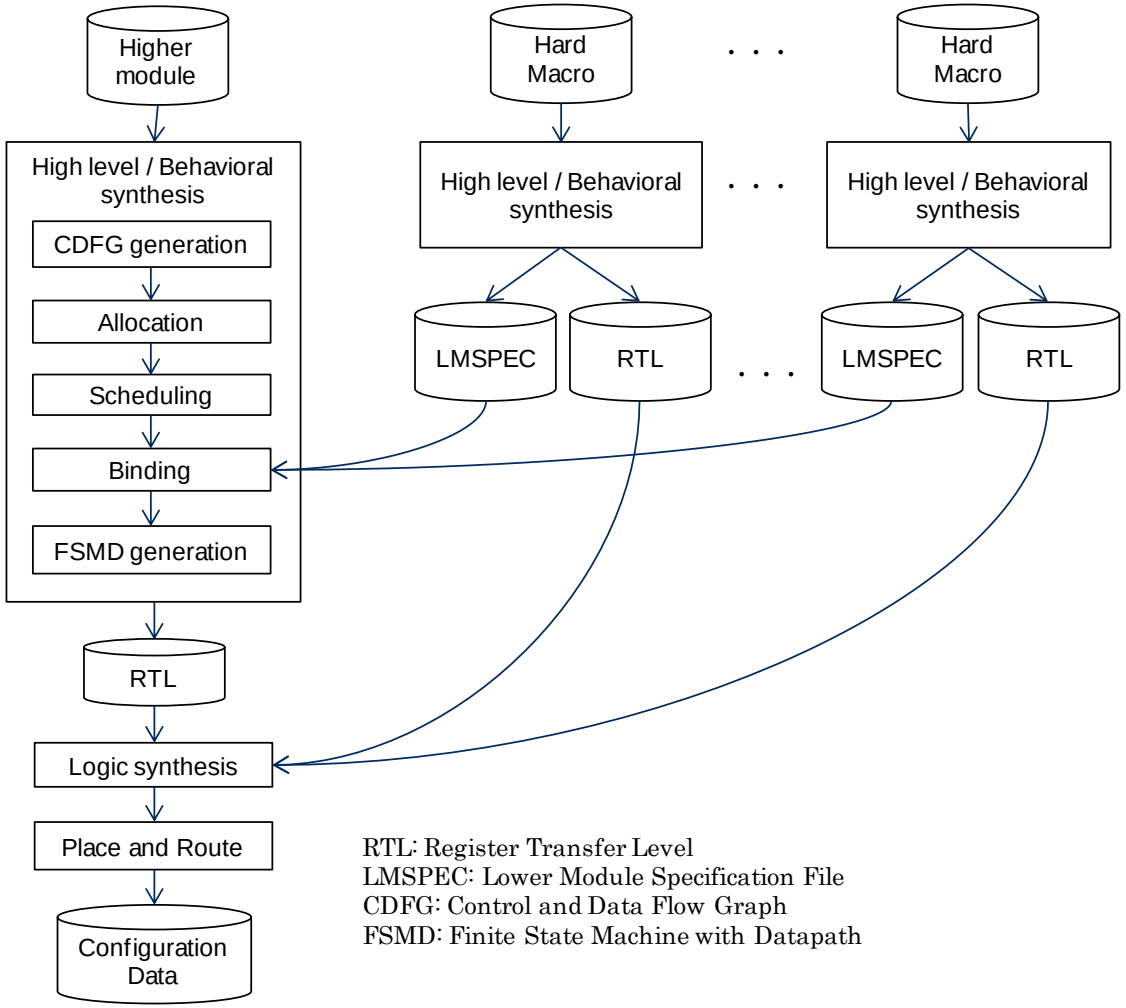


図 4-2　関数演算器化機能のしくみ

4.2. 階層化設計フロー

前節までに述べた GPE アーキテクチャ、および粒度別階層化設計手法を用いて実際の外界フィールドの厳しい環境で使用されるセンシングデバイスに組み込む PE を設計することを主目的として確立した設計フローについて述べる。プログラムメモリやコンフィギュレーションメモリなどの外付けメモリ素子を持たないアーキテクチャはソフト・エラーが懸念事項となるアプリケーションに適しており、これらのメモリが無くとも、高級プログラミング言語を使用して開発ができることを設計フローの必須要件とした。この設計フローは前節で述べた粒度別階層化設計手法に対応し、以下に示す 6 つの Step から構成する。

[Step 1]: 対象とするアプリケーションについて、システム全体をハードウェア／ソフトウェアの区別なく、C 言語等の高級言語で記述する。ここで使用する言語はアプリケーションソフトウェアの開発にも使用できるものを選択すると便利である。

[Step 2]: 粗粒度の演算機能の定義を関数で記述する。この機能ブロックは ASIC 開発ツールなどを用いて最適化したハードマクロに対応させてもよく、回路設計レベルにおいても高性能を追求する。通常のマイクロプロセッサに使用されている基本的な算術演算である四則演算や積和演算のみならず，画像圧縮や画像認識、特長抽出などの画像処理回路や FFT 等の信号処理回路を実装できることも前提とする。動作合成の際には、これらの処理をニーモニックとして指定して使用することができる。CWB ではこの機能を関数演算器化機能として実現した。

[Step 3]: 動作合成を行い, ハードウェア記述言語 (Hardware Description Language: HDL) で記述されたソースコードを生成する。動作合成時には Step 2 で定義したハードマクロを十分活用して回路を生成する。この要求は CWB の最適化機能として新しく実装した。

[Step 4]: 論理合成を行うステップである。通常の LSI や FPGA の論理合成プロセスと同様である。

[Step 5]: レイアウト設計を行うステップである。通常の LSI や FPGA のレイアウト設計プロセスと同様である。

[Step 6]: 検証（Verification and Validation）を行うステップである。遅延解析の結果に基づいて、レイアウト設計とバックアノテーションを行う。このフェーズはレイアウトに依存する遅延解析や，その結果を基にしたバックアノテーションも含み，Step 5 を含む Iteration を考慮に入れることになる。ステップ 3 の論理合成までさかのぼるバックアノテーションもまた予想され、さらに、バックアノテーションは高水準言語で処理できる必要があるため、ステップ 1 まで戻ることも想定する。

本設計手法の物理イメージを図 4-3 に示す。

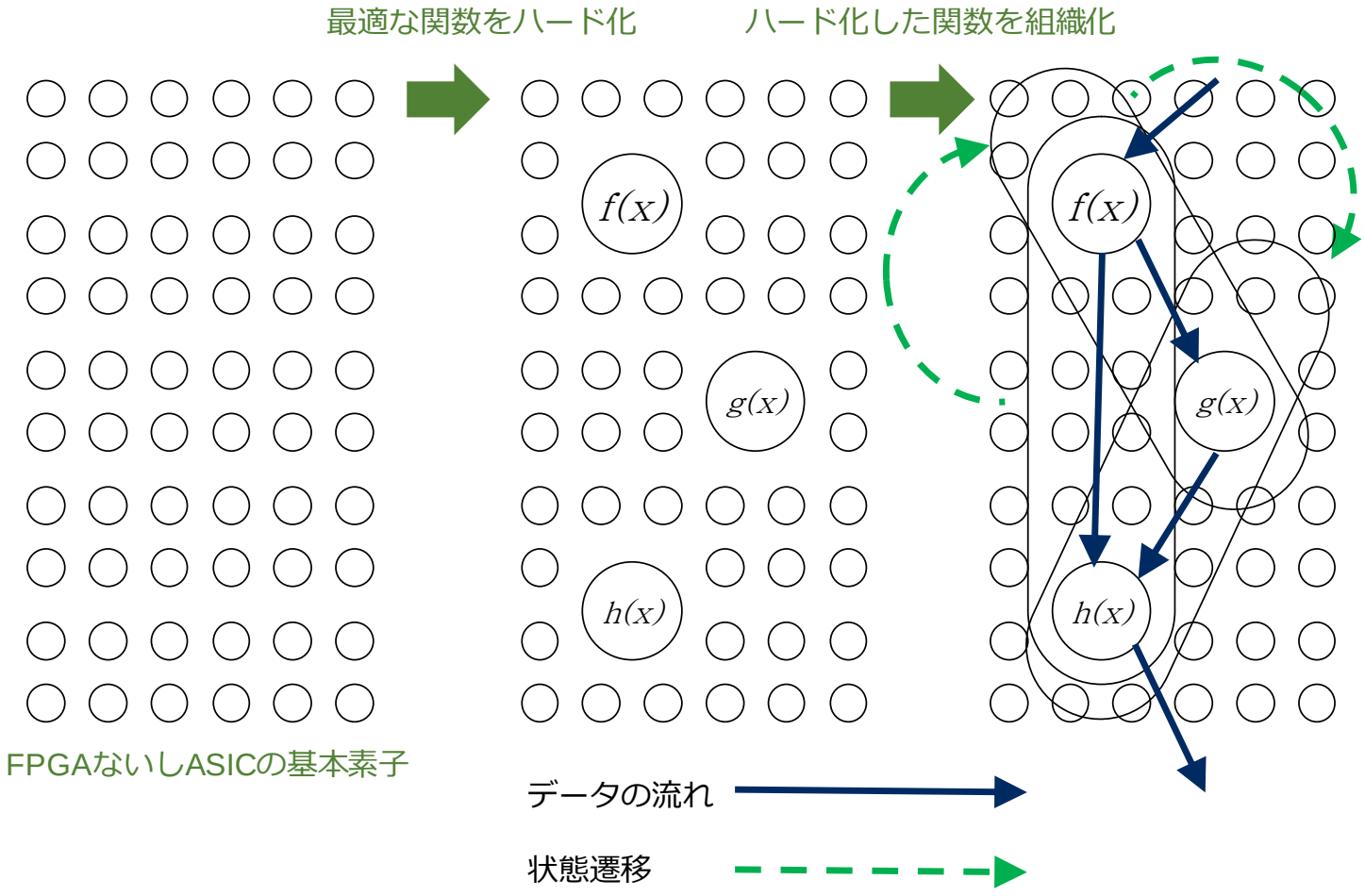


図 4-3　本設計手法の物理イメージ

本設計手法では、まず FPGA ないし ASIC の基本素子が並んでいることを前提とする。アプリケーション要求を分析し、ナノブリッジを活用して最適な関数を演算器化してハード化実装する。演算器の機能は関数記述を前提としているので関数で定義された可能がナノブリッジを用いて LSI の回路として凝集することになる。このハード化した関数である演算器の実装情報をデータベース化し、高位合成ツールにより解析されたデータフローグラフとコントロールフローグラフにより各状態に対応したグループに組織化し、状態間の遷移条件に基づいて全体回路接続を行う。最後に外界との入力と出力を定義する。各関数はデータの流れと共に組織化され、組織化した関数の組は状態遷移に応じて活性化する。

Step 2 についてはソフトウェア実装をしたものについて、文献[42]などに実装例を報告している。そこでは画像圧縮命令や画像認識のための基本処理がニーモニックとして指定できる。この実装は、最適化された動作のための FPGA とソフトウェアによって実現された。画像圧縮操作および画像認識の基本操作要素は、そのインタプリタ型言語記述のニーモニックとして指定され、その有効性が検証されている。

第5章
単一故障点を排除する高信頼性化アーキテクチャ

本章では、本研究にて確立した設計手法を高信頼化設計に応用することを検討した結果について述べる。本章で述べる内容は、これまでに宇宙機に適用することを目的として進めてきた耐放射線性高信頼性計算機に関する研究の成果に、本研究で確立した設計手法を応用することにより更なる高信頼性化を実現したものである。本節では、宇宙機搭載用高信頼性計算機について関連研究を概観し、これまでに提案されている高信頼性化設計手法の課題について述べる。

5.1. 高信頼性化目標

人工衛星や宇宙ステーションなどの宇宙機に搭載されるオンボードコンピュータ(Onboard Computer: OBC)を開発するに当たっては、通常のコンピュータと比較するとその使用される環境において考慮しなければならないことが多い。宇宙機に搭載するコンピュータを開発する上で考慮すべき環境条件に付いては 1.3 節に述べた。これらの環境条件のうち、特に放射線の影響に対して耐性をもたせるアーキテクチャを提案する。近年の半導体プロセスの微細化により、宇宙環境のみならず地球上においても自然放射線の影響は無視できなくなってきている [67]。したがって、本研究の成果は宇宙機搭載用コンピュータのみならず、民生品にも適用可能である。

宇宙機に搭載するコンピュータは打ち上げた後に保守を行うことがまず不可能であるため、MCU 上のプログラムの暴走や他の機器から受信したテレメトリ・ステータスエラーなどの機能レベルのエラーについても十分考慮する必要がある。特に衛星自体の運用にかかわるバス系と呼ばれる機器については故障が検出されても破局的な故障には至らず、最悪の場合においてもいったんミッションを中断し、マニュアル操作によってオペレーション／復帰ができるようなフェイル・セーフ機能が必要とされてきた。

近年の宇宙開発においてはロケットや人工衛星などの宇宙機自体の開発から、宇宙機の提供する機能、サービスの高度化へと重点が移ってきている。有人活動のサポートが要求される一方で、微小重力を必要とし、ごくわずかな振動さえも排除するために意図的に無人環境とするミッション要求もあり、無人機に対する要求も高度化してきている。また、有人活動を協調的に支援するための遠隔操作などの自動化技術も求められるようになってきた。

人工衛星に関してみれば、そのミッションは、通信衛星や放送衛星に代表される通信機能、および気象衛星や地球観測衛星、深宇宙プローブに代表されるリモートセンシング機能が主である。特に後者のリモートセンシング機能を使用するミッション要求の高度化は顕著であり、ミッション機器のインテリジェント化が進み、MCU を内蔵したセンサが増えてきている。さらに個々のセンサを有機的に統合して運用する自動化・自律化運用を司るための OBC も求められるようになってきた。通信機能についても自動追尾・捕捉のための通信装置やアンテナ駆動装置のインテリジェント化が進んでおり、これらにも MCU が使われるようになってきた。有人活動の支援、あるいは代替としての軌道上作業機に搭載されるロボットの運用も、これから重要性が増してくるミッションの一つである。これらのミッション機器では従来のフェイル・セーフからさらに進み、異常・故障が生じてもミッションを継続するフェイル・オペラティブ機能が求められるようになってきている。

宇宙機自体の運用に用いられるバス系機器についても、宇宙往還機による物資の輸送や軌道上作業機による衛星・プラットフォームなどへの軌道上サービスの提供のためのランデブ・ドッキング技術の要求などにより、フェイル・オペラティブ機能が求められてきている。バス系機器は宇宙機の寿命や安全性を直接左右するものだけに、ここに求められるフェイル・オペラティブ機能は信頼性の高いものでなければならない。したがって、これらのミッション機器、およびバス機器に使用する OBC への耐故障性の要求はますます高度化してきており、フェイル・セーフ、さらにフェイル・オペラティブ機能を実現するためのさまざまなアーキテクチャが開発されている。本研究で提案する高信頼性化アーキテクチャはフェイル・オペラティブに対応している。

5.2. 粗粒度関数定義の高信頼性設計への応用

本節では、本研究成果である粗粒度関数定義による演算器の設計が高信頼性設計にも有用であることを示す。

5.2.1. 高信頼性設計の目的

GPE アーキテクチャに基づく粒度別階層化設計手法とナノブリッジの併用により、外付けメモリ素子を要しない PE を構成可能であることを前章までに述べた。これによりメモリのソフト・エラーの影響は排除できるが、ナノブリッジが活用されるのは LSI の配線層であり、CMOS 下地については別途高信頼性設計が必要である。半導体物理の観点から放射線耐性を持たせる対策も採られているが、SEE については物理的な手法では完全に対処することはできず、回路技術的な対策が併用される。

本研究では、高信頼性化設計手法とは単一故障点を除去することと定義する。すなわち、回路設計技術として単一故障点の除去を目的とした手法を取り、一点が故障してもシステム全体には波及しないようにする。具体的には、以下のような分散化設計を行う。

(1) 演算機能の分散化

この例としては、多数決方式、複数回演算して結果を確認する等の時間軸上の冗長化設計などが用いられる。

(2) 故障検出機能の分散化

この例としては、誤り検出・訂正符号や、プロセッサ間の出力の比較等の設計手法が用いられる。

(3) 故障分離機能の分散化

故障分離機能としては、波及故障の防止を防ぐための種々の対策が採られる。

(4) 再構成機能の分散化

一部の機能の縮退、性能低下は許容した上で、残存している機能を活用してシステム全体で処理の継続を図る設計手法が採られる。

(5) 再同期機能の分散化

冗長系を構成するプロセッサのうち、新たに加わったものや、機能が復活したものの処理内容を他のプロセッサと同期させる設計手法が採られる。

5.2.2. 信頼性設計フロー

前節に述べた高信頼化設計を粒度別階層化設計手法に対応させるに際しては、これらの各階層において故障検出、故障分離、再構成（Fault Detection, Isolation, Reconfiguration）の各機能を実装する。すなわち、粒度別階層化設計手法の各階層に対して主要な高信頼性設計手法を対応させる。信頼性設計フローを確立するに当たっては、回路設計手法としては粗粒度関数定義層とパイパス接続層での対策が主眼となる。細粒度定義層とインタフェース回路層では、主に素子レベルの回路設計対策が採られる。これらを図 5-1 に示す。

最下層のインタフェース回路層では耐放射線性を有するフリップフロップやバッファ、ないしは SET 耐性を考慮した Phase Lock Loop (PLL)などの回路設計レベル，ないしは Silicon on Insulator (SOI)や Silicon on Saphire (SOS)等のプロセス設計レベルで耐放射線性を考慮して実装され，これらを論理設計のプリミティブとして活用することができる。細粒度定義層では、回路全体に亘って評価することにより信頼度に関する感度の高いところに冗長構成を適用することにより，高信頼性を確保しつつ，消費電力や実装面積，速度などのリソース上のオーバーヘッドを最小限に留めるように設計する。このような種類の冗長性がチップ全体に採用されると高い実装密度は期待できないため、選択的に冗長構成を適用する必要があり、これらの判断によって消費電力、レイアウト面積、冗長性による処理速度の過剰なリソースオーバーヘッドを回避する [17, 18]。粗粒度関数定義層では PE 単位で TMR 等を適用するとリソース上のオーバーヘッドが過大になるばかりでなく，信頼度計算上も必ずしも信頼度を向上できるとは限らない。これに対しては後述する方式や[37]に述べられているソフトウェア通信による冗長構成などを用いると信頼度を効果的に高めることができる。最上層のバイパス接続層では実質的に通信ネットワークの高信頼化

設計になる。ここでは宇宙機搭載実績の豊富な SpaceWire [42, 59, 60, 61, 62, 63, 64]などのハードウェアのみでルーティングを実現できるような方式を活用して高信頼性化を図ることにより，オープンな国際標準規格の下にシステム設計を遂行することが可能となる。以上の信頼性設計フローを表 5-1 に示す。

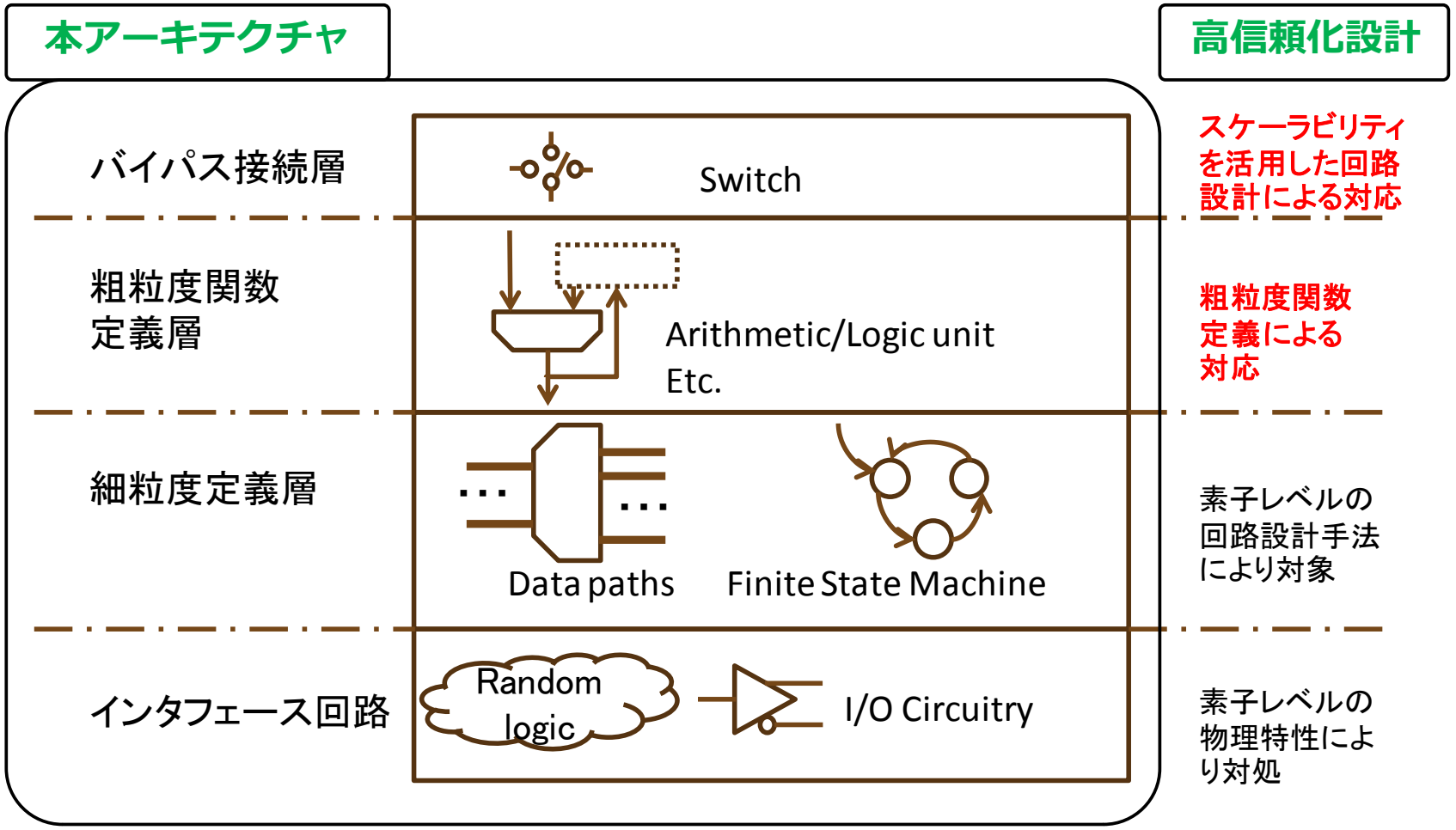


図 5-1　高信頼化設計手法の統合

表 5-1　信頼性設計フロー

設計階層	Implementation	Remarks
パイパス接続層	SpaceWire 規格 [59]を用いたルーティング等	システム的な FDIR 設計による。
粗粒度関数定義層	TMR , CRAFTSYSTEM [25, 26, 27, 28], ソフトウェア比較方式[37]など	システム的な FDIR 設計による。
細粒度関数定義層	三重冗長化多数決方式 (Triple Modular redundancy with a voter: TMR), 等	システム的な FDIR 設計の視点も取り入れた回路設計による。
インタフェース回路層	耐放射線性ライブラリ等	デジタル／アナログレベルの回路設計による。

5.2.3. 粗粒度関数定義を冗長管理部に応用する例

多数決方式や、ソフトウェア通信方式のようなシステム的アプローチに対し、前述したようにフォールトトレラント計算機の持つ各機能（演算機能、故障検出機能、故障分離／再構成機能、および再同期機能）の分散化を図ることによって単一点故障を回避するアプローチがある。粗粒度関数定義を冗長管理部に応用するに際しては、後者のアプローチによる高信頼性化設計について効果が大きい。なぜなら、多数決方式の Voter 回路などは単一故障点であり、この部分の機能を増やすと、定量的に信頼度が低下するためである。この後者のアプローチによる多重化方式について、下記の実現目標を設定した。

- ・原理的に単一故障点を有しないこと。すなわち、一つのデバイスの故障によって、システムダウンにつながらないこと。
- ・どのようなデバイスにも適用可能であること。プロセッサの世代交代は早い。特定のプロセッサに依存した方式としたのでは、プロセッサの寿命とともに多重化方式の寿命も尽きてしまう。また、汎用のプロセッサばかりでなく、ディジタル・シグナル・プロセッサ(DSP)などの特殊用途プロセッサや、周辺素子などにも拡大して適用できることが望ましい。
- ・多重度を容易に高められること。たとえば、三重冗長システムから四重冗長システムへはほとんどハードウェアの再設計を必要とせずに拡張できることが望ましい。
- ・ソフトウェアは原則として多重化されていることを意識せずに済むこと。宇宙機上の OBC に搭載されるソフトウェアは年々大規模になってきている。ソフトウェアを開発する場合に、なるべく負荷を少なくする必要がある。

・システムがリアルタイムで再構成されること。バスの制御権を有するプロセッサに異常が発見された場合、バスマスタの交代のためにシステムの動作を中断する時間を極力なくすものとする。

これらの実現目標を反映して開発した高信頼性方式に粗粒度関数定義を組み合わせたものを図 5-2 に示す。

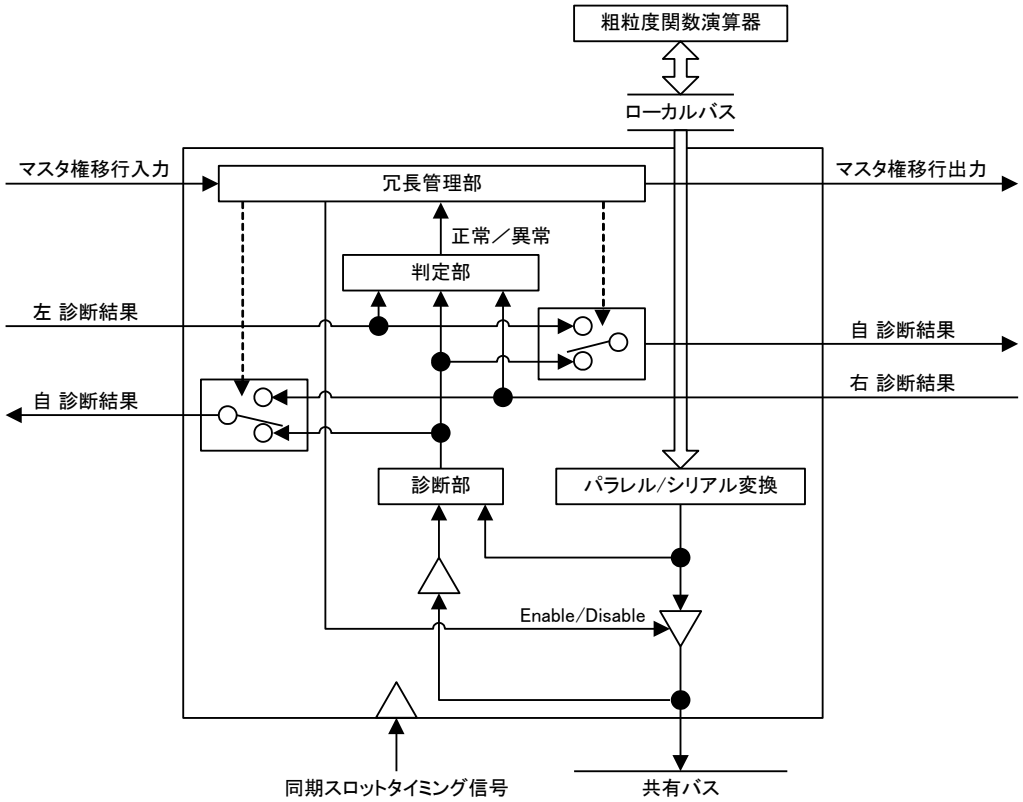
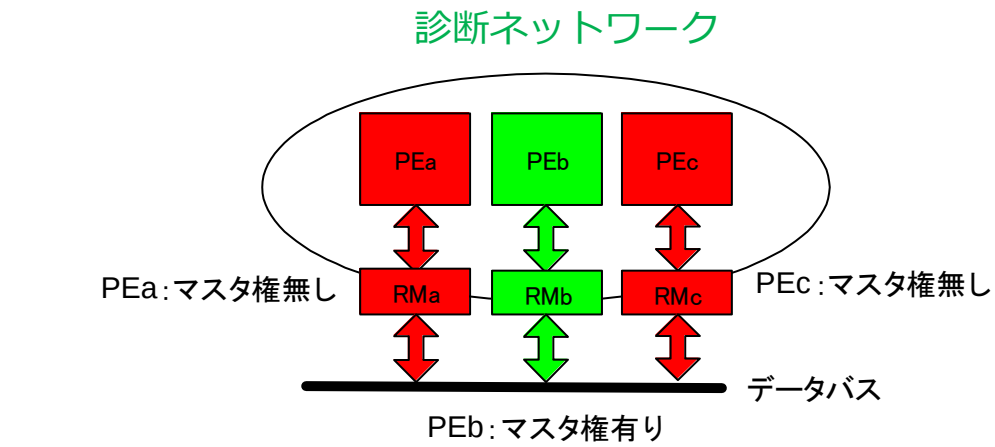


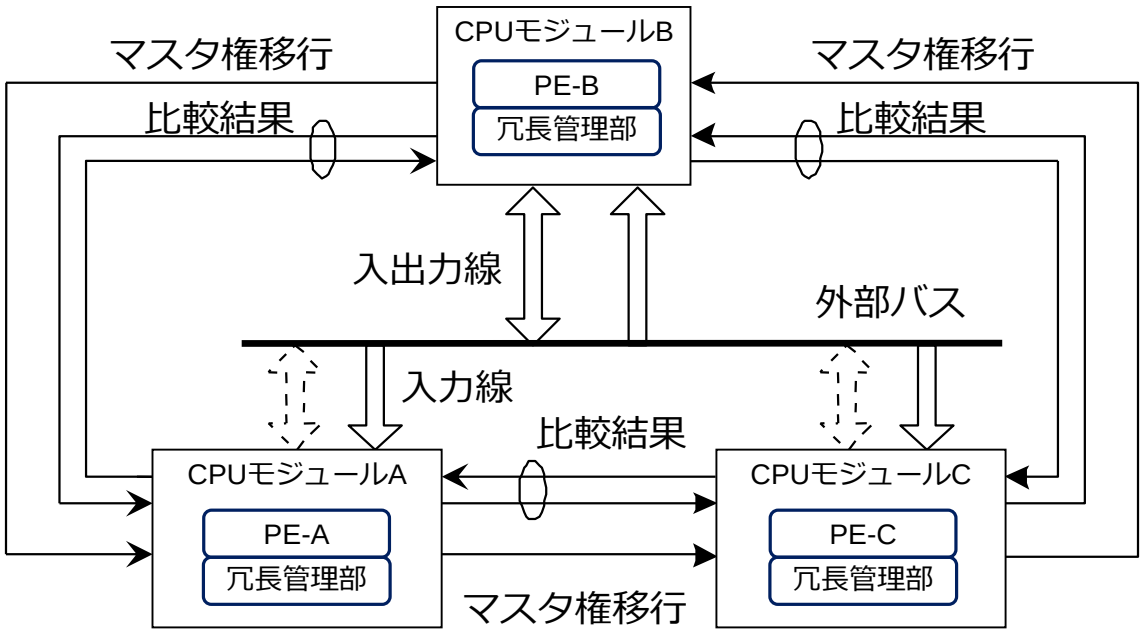
図 5-2　粗粒度関数定義を冗長管理部に応用した例

5.2.4. 故障検出，および再構成機能

前節の冗長管理部を用いた多重化システム構成例の全体構成を図 5-3 (a)に示す。Micro Controller Unit（MCU）および周辺回路からなるおのおの独立したプロセッサエレメント(PE)は診断ネットワーク、およびそれぞれ外部に出力を伝えるデータバスに入出力線を介して接続されている。各 PE は比較結果を伝達する出力比較信号線を通じてリング状に接続された診断ネットワークを構成している。各 PE は外部バスの内容を取り込む入力線を有し，この例では外部バスの内容と自 PE 内の MCU からの入出力とを比較回路で比較する。診断ネットワークとデータバスを構成する信号の詳細を図 5-3 (b)に示す。



(a)



(b)

図 5-3　多重化システム構成例、
(a) 診断ネットワークとデータバス、(b) 接続信号詳細

これらの PE は外部バスにデータを出力する際に同期して動作しており，外部入出力線に対する出力権はその中の一つの PE にだけ与えられる。これをマスタ PE と呼び，それ以外をチェッカ PE と呼ぶ。図 5-3 では図 5-2 に示した冗長管理部を各 PE に内蔵し、PE-B がマスタ PE，それ以外はチェッカ PE として構成している。チェッカ PE は出力バッファにより外部バスへの出力を断たれているので入出力線が点線で表されている。冗長管理部に

は各 PE 内部に設けられた比較回路の出力である比較結果と，両隣に接続された PE から出力される比較結果との合計三つの信号とから自 PE が正常か異常かを判断する検出回路を有する。これらについて以下に述べる。

(1) 診断ネットワーク

PE および冗長管理部（Redundancy Management module: RM）からなる 3 系統のおのおの独立した CPU モジュールは、それぞれ外部に出力を伝えるデータバスに入出力線を介して接続されている。各 CPU モジュールはデータバスの内容と自モジュール内の PE からの入出力とを診断部で比較するために、データバスの内容を取り込む入力線を有している。

これら 3 系統の CPU モジュールは同一システムクロックで同期して動作しているが、外部入出力線に対する出力権はその中の一つの CPU モジュールにだけ与えられる。これをマスタ CPU と呼び、それ以外をチェッカ CPU と呼ぶ。図 5-3 (b)では CPU モジュール B がマスタ CPU、それ以外はチェッカ CPU であり、チェッカ CPU は出力バッファによりデータバスへの出力が断たれているので入出力線が赤色で表されている。

各 CPU モジュールは診断結果信号線を通じてリング状に接続され診断ネットワークを構成しており、各 CPU モジュール内部に設けられた診断部の出力である診断結果と、両隣に接続された CPU モジュールから出力される診断結果との合計三つの信号とから自 CPU モジュールが正常か異常かを判断する判断部をもつ。

(2) 故障検出、および再構成機能

各冗長管理部は自 CPU モジュールが異常と判断した場合、自身をネットワークから論理的に切り離し、マスタ権を隣接した CPU モジュールに移管し、自モジュールの診断結果と隣接 CPU モジュールからの診断結果入力とを切り替えて（すなわちバイパスして）隣接 CPU モジュールに出力する回路を有している。

各 CPU モジュールはバスサイクル毎に診断回路により自モジュール出力と外部バスに出力されているマスタ CPU モジュール出力との比較を行い、自モジュールの比較結果と両隣接モジュールからの比較結果を検出回路に取り込む。両隣接モジュールと自身の診断結果から自身の正常／異常は完全に判定できる。そしてもし自モジュールが異常と判断された場合は処理回路からのコントロール信号によりスイッチが切り替えられ隣接 CPU モジュールからの診断結果入力が異常 CPU モジュールをバイパスして隣接 CPU モジュールへ出力される。これにより異常な CPU モジュールはシステムから隔離され、N 個の CPU モジュールから構成されていたシステムから N-1 個の CPU モジュールから構成される縮退システムへの再構成が瞬時に行われる。

マスタ CPU に異常が発生した場合は、冗長管理部からの出力禁止信号により出力バッファが出力禁止状態となり、マスタ CPU としての機能は取り除かれる。その後、冗長管理部からのマスタ移行出力信号により隣接 CPU モジュールが制御権を得、マスタとして機能するようになる。隣接 CPU モジュールからのマスタ移行入力信号により外部出力線への出力権が得られた CPU モジュールは冗長管理部からの出力禁止信号を解除することによってマ

スタ CPU モジュールとなる。これらの動作により本システムでは異常出力はマスクされ、システムからは常に正常な出力が得られる。

5.2.5. 本方式の特徴

一般に高信頼性計算機（Fault Tolerant Computer: FTC）は，冗長な PE を持ち，適当な選択回路により各 PE の出力で多数決をとることなどによって，PE に故障が生じてもシステムダウンしないような工夫がなされている。しかし，このようなシステムでは，選択回路が壊れてしまうとやはりシステムがダウンしてしまい，選択回路の信頼性がシステム全体の信頼性に対して支配的になってしまう（単一故障点）。

本方式では，各 PE の内部では PE の異常を検出して，システムから自らを切り離す仕組みを採用することで，多数決方式による多数決回路のような外部共通回路を必要とせず，単一故障点をシステムから排除している。さらに各 PE は同一構成にて実現可能であるため，3 モジュール以上であれば原理的に N 重（N≧3）まで回路の変更なしに拡張可能であり，システム冗長性要求の変化に対して容易に対応可能である。また本方式は対象となるモジュールが MPU や MCU である必要はなく，プロセッサ・アーキテクチャに依存しない方式であることから各 PE の独立動作も可能であり、DSP（Digital Signal Processor）や近年高度化している I/O（Input / Output）プロセッサなど他の機能デバイスにも応用可能である。一つの構成例として、信頼度の異なるプロセッサと、デジタルシグナルプロセッサが混在可能であり、この例を Appendix C に示す。

また，本方式は冗長構成をハードウェアの機能で実現しているため，ソフトウェアは冗長性を意識せず，あたかもシングルプロセッサ上にシステムを実現するように開発することができる。民生部品は一般的にライフサイクルが非常に短い。このため，ソフトウェアが介在する耐故障性方式を採用すると部品が変更される度にソフトウェアも変更を余儀なくされる。本方式では冗長構成を構成するにあたりソフトウェア処理を必要としないことから，部品が変更されてもソフトウェアの移植性が非常に高く，ライフサイクルの短い民生部品を効果的に活用することが可能という特徴を有している。

縮退モードへの遷移がソフト・エラーによるものであった場合、隔離したモジュールを再同期させることにより初期状態への復帰が可能である。この操作は障害発生モジュールの再同期、およびそれがソフト・エラーか永久故障かを判断するためのチェックシーケンスを含むため、処理の複雑化が予想される半面、縮退モードでの運転による処理能力の低下はないので本シーケンス起動に対する時間的制約は低い。ゆえに再同期処理はハードウェア的に処理されるのではなく、ソフトウェアの処理負荷の低いタイミングを見計らって起動されるものと考えられ、オペレーティングシステムのシステムコールとして実現するのも適当であろう。これらは図 5-4 に示す多数決方式では得られない特徴である。多数決方式では多数決回路が単一故障点となっており、また、冗長構成の再構成には別途回路が必要となる。

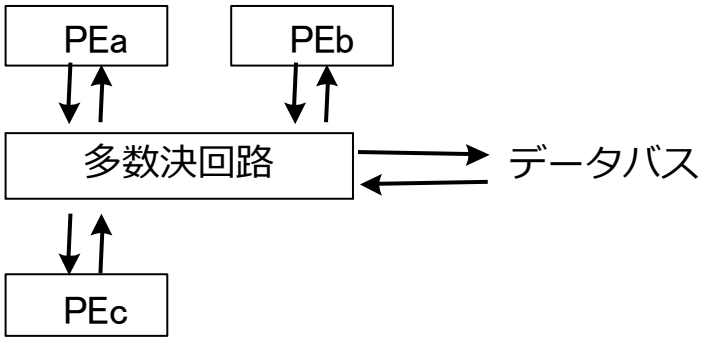


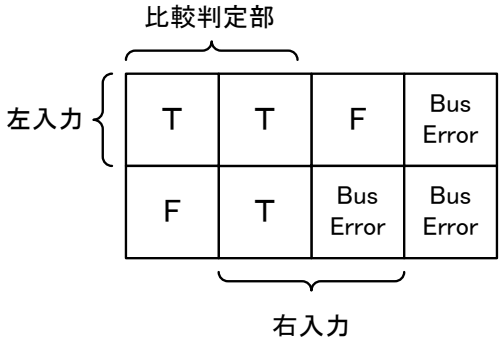
図 5-4　多数決方式

本節で述べた粗粒度関数定義を活用した冗長システムについて従来の多数決を用いたシステムに比べて信頼度が高いことを定量的に検証する。本方式のようなセルフパージング方式を用いた N 重冗長フォールトトレラント方式は、単一故障点を有していない。したがって、信頼度を検討する際には、従来の直並列モデルではその特性を十分に表現することができない。このような構成方式関する解析モデルとしては Appendix A に示す森永らによって提案された領域分割モデル [43, 44, 45, 46]を用いて解析を行うことができる。本方式は本数学モデルにより予測される最適解に非常に近い方式であることが示されており [45, 46]，多数決方式よりも高い信頼度を実現している。

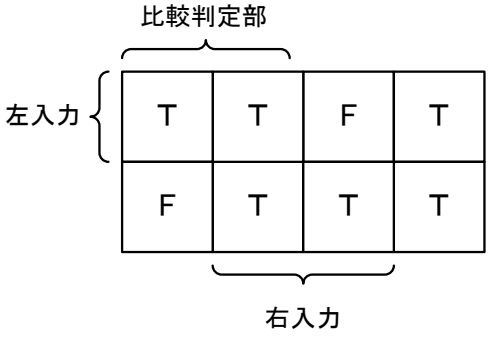
5.3. 粗粒度関数定義を冗長管理部に応用した評価

5.3.1. 比較判定関数

前節の説明では診断部はデータの比較としているが、診断部に特殊な関数を組み込むことで、比較判定部自身の異常も検出できるようにした。この様子を図 5-5 に示す。これは多数決方式などの冗長構成方式では得られない特徴である。なぜなら、多数決方式は通常の 3 重冗長構成では Voter 自身の故障は判断できないためである。このため、Voter 部には高信頼性デバイスを使わざるを得ず、さらに Voter 部の信頼度が全体信頼度に大きな影響を与えていた。前節で述べた高信頼性システムを用い、比較判定部自身の故障を検出する関数を組込むことは、粗粒度関数定義層の設計と親和性が高く、今回設計手法を確立した GPE アーキテクチャの有望な応用先である。



(a)



(b)

T(En) ≡ {Eom=Bus(Eon)}・T(E(左側))
　　∨not(T(E(左側)))・T(E(右側))
　　∨not(Eom=Bus(Eon：自モジュール))・not(T(E(右側)))
F(En) ≡ not T(En)
Bus(Eon) ≡ バスを通った Eon

(c)

図 5-5　比較判定部自身の故障を検出し得る粗粒度関数定義

(a) マスタ権を有する冗長管理モジュールの比較判定式（カルノー図）、(b) マスタ権を持たない冗長管理モジュールの比較判定式（カルノー図）、(c) 比較判定部自身の故障を検出する粗粒度関数定義

5.3.2. 比較判定関数の実装評価

上述した比較判定関数の実装規模を見積るため、本研究で確立した GPE フレームワークを活用して実装評価を行った。本方式は多数決と異なり二重化構成でも信頼度向上効果を有すると共に、実際の応用に於いてもコスト、実装面積、および消費電力の削減のために二重化構成がよく用いられるため、本評価は二重冗長構成で実施した。

一つのモジュールは図 5-6 に示す 3 つのモジュールで構成した。最上位層である Cluster core は二つの Unit から構成される。各ユニットは演算処理を行う Reconfigurable Cell Unit (RCU)と冗長管理を行う RedunDancy control Unit (RDU)であり、RDU には比較判定関数の処理回路を含む。

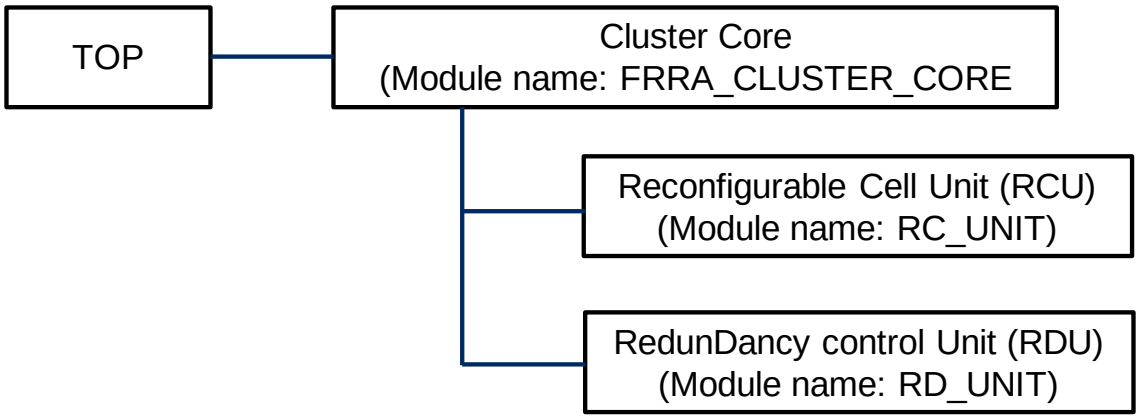


図 5-6　モジュール構成

本実装評価で設定した冗長構成回路をシミュレーションする際に設定した性能は以下のとおりである。

・システム動作クロック（2MHz）
・リセット　：High Active
・RCU 内に実装した演算：四則演算（加算、減算、乗算、除算）
・四則演算結果 16bit/1 ワードデータ単位による 4 線シリアルバス出力（I/O バス出力）
（4 線(I/O バス)：データ、クロック、スロット信号、Valid/Invalid 信号）
・四則演算結果 16bit ＝ MSB first でのシリアルデータ転送
・システム動作クロックの 2 分周（1MHz）でのシリアルクロック生成
・シリアルバス転送中における Busy 信号の生成
・外部 I/F 経由によるループバック後のシリアルデータ比較
・シリアルデータ比較による結果通知機能
・マスタ権保有時、シリアルデータを出力
・マスタ権の移譲　：比較判定関数に基づきチェッカ側へマスタ権を移譲する。マスタ権

をチェッカ側へ移譲した後のマスタ側の比較結果は F(偽)固定とする。マスタ権をチェッカ側に委譲した後のチェッカ側の比較判定関数はマスタのものを適用する。

・Bus Error 発生時の処理：二重系構成の場合は Bus Error と演算部の両系故障の区別ができないことから、系全体をリブートする等の処理を行う。

Cluster Core の回路構成を図 5-7 に、各端子の機能を表 5-2 に示す。本実装では RCU、および RDU 内蔵の比較判定部に粗粒度関数定義を適用した。RCU は動的再構成手法により構成し、Cluster core の状態 ALU_MODE に応じての演算モードが変わる。演算モード設定を表 5-3 に示す。

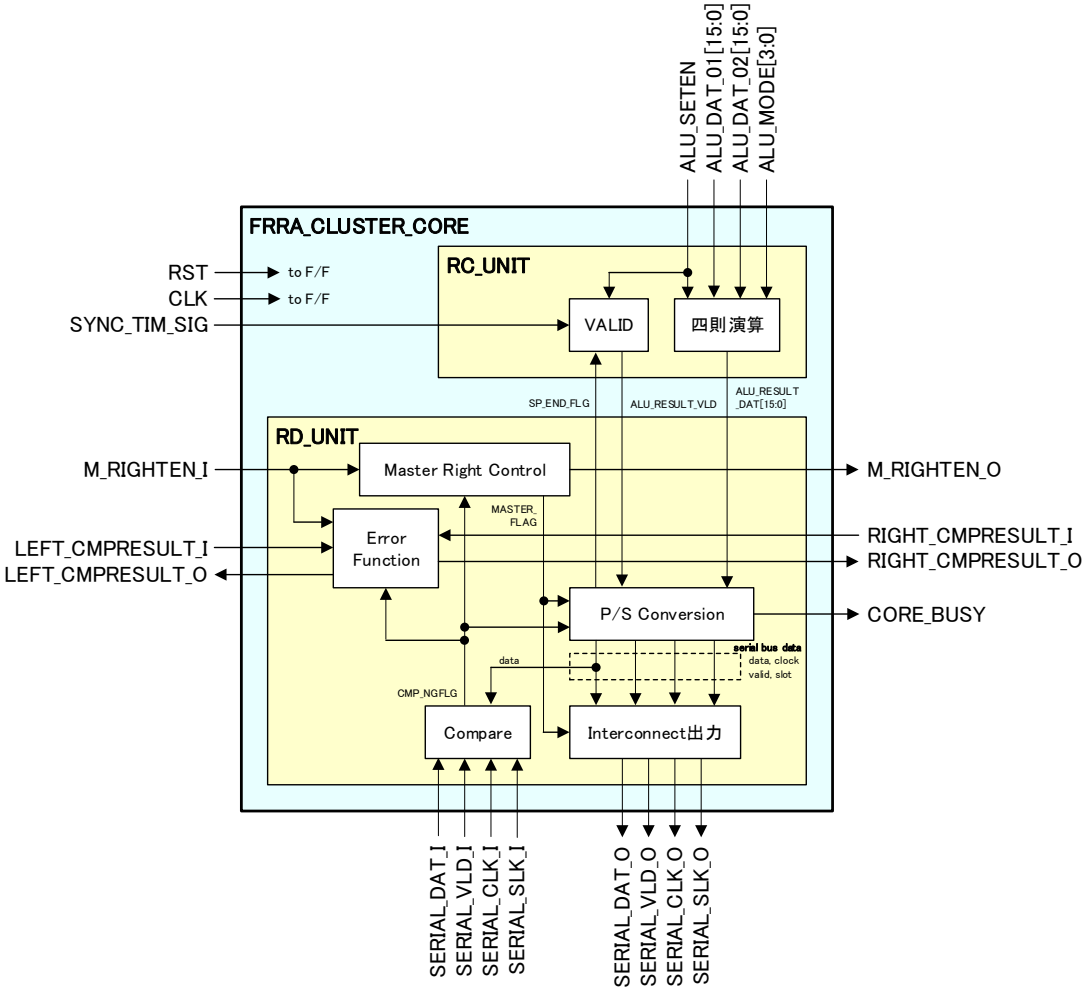


図 5-7　Cluster Core 回路構成

表 5-2　Cluster Core 端子表

端子名	I/O	Bit	Act	機能説明
RST	I	1	H	System Reset
CLK	I	1	↑	System Clock　2MHz
SYNC_TIM_SIG	I	1	H	回路間におけるタイミング同期信号
CORE_BUSY	O	1	-	Cluster Core Busy 信号
ALU_DAT_01	I	16	-	Arithmetic and Logic Unit データ信号　01
ALU_DAT_02	I	16	-	Arithmetic and Logic Unit データ信号　02
ALU_MODE	I	4	-	Arithmetic and Logic Unit モード信号
ALU_SETEN	I	1	H	Arithmetic and Logic Unit セット信号
M_RIGHTEN_I	I	1	H	マスタ権移譲イネーブル入力信号
M_RIGHTEN_O	O	1	H	マスタ権移譲イネーブル出力信号
LEFT_CMPRESULT_I	I	1	-	シリアルデータ比較結果入力(左辺)
LEFT_CMPRESULT_O	O	1	-	シリアルデータ比較結果出力(左辺)
RIGHT_CMPRESULT_I	I	1	-	シリアルデータ比較結果入力(右辺)
RIGHT_CMPRESULT_O	O	1	-	シリアルデータ比較結果出力(右辺)
SERIAL_DAT_I	I	1	-	4 線シリアルデータ入力（データ）
SERIAL_VLD_I	I	1	H	4 線シリアルデータ入力（VALID）
SERIAL_CLK_I	I	1	↑	4 線シリアルデータ入力（クロック） 1MHz
SERIAL_SLT_I	I	1	-	4 線シリアルデータ入力（スロット）
SERIAL_DAT_O	O	1	-	4 線シリアルデータ出力（データ）
SERIAL_VLD_O	O	1	H	4 線シリアルデータ出力（VALID）
SERIAL_CLK_O	O	1	↑	4 線シリアルデータ出力（クロック） 1MHz
SERIAL_SLT_O	O	1	-	4 線シリアルデータ出力（スロット）

表 5-3　RCU の演算モード設定

ALU_MODE[3:0]	四則演算	式 (ALU_RESULT_DAT[15:0]=****)
“0001”	加算(＋)	ALU_DAT_01[15:0] ＋ ALU_DAT_02[15:0]
“0010”	減算(－)	ALU_DAT_01[15:0] － ALU_DAT_02[15:0]
“0100”	乗算(×)	ALU_DAT_01[15:0] × ALU_DAT_02[15:0]
“1000”	除算(÷)	ALU_DAT_01[15:0] ÷ ALU_DAT_02[15:0]
Others	－	16’h0000

RDU は、Master right control、Error function、P/S Conversion、比較判定関数（Compare)、および主系（マスタ）と従系（チェッカ）間の Interconnect 出力で構成する。Master Right Control はマスタ権の保持ないし委譲を管理する機能であり 4 線シリアルデータの出力マスクとマスタ権の移譲の制御を行う。図 5-5 に示す比較判定関数（Compare）の出力が F であった場合に、4 線シリアルデータの出力停止とマスタ権移譲を行う。Error function は Compare の結果を取り込み、比較結果の出力を制御する。P/S Conversion は RCU のパラレルデータ[15:0]出力を MSB first でシリアル出力に変換する。シリアル変換開始と同時に Interconnect 出力へのシリアルクロック、スロット信号、Valid/Invalid 信号、および Busy 信号を生成する。また、1 トランザクション分のデータをバッファリングし、スロット信号の L 区間(CHK)の完了とともにバッファしたデータに対して再度シリアル変換を行う。計 2 回分のデータ[15:0]のシリアル変換を持って、転送を終了する。Compare は I/O バスへ出力したシリアルデータと、I/O バスから帰ってきたシリアルデータ（ループバック）を比較する。比較結果を Master right control と Error function へと通知する。Interconnect 出力は Master right control の指示に従い、シリアル変換したデータを I/O バスへ出力するかどうかのマスク処理を行う。本 Cluster Core を用いて構成した二重冗長系の全体構成図を図 5-8 に示す。マスタ用比較判定関数は図 5-9 に示すように二重系では左入力は常に真（T）とみなし、左入力は偽になり得ないものとする。これを Don't care で示している。チェッカ用比較判定関数は図 5-10 に示すように二重系では右入力を常に真(T)とみなし、右入力は偽になり得ないものとする。これを Don't care で示している。

Cluster Core の回路規模を見積もるため、以下の FPGA にて確認した。

・使用ツール　　：Vivado 2015.1

・デバイス指定　　：xc7a100tcsg324-3

この時の Cluster Core 1 系統における RDU の回路別規模を表 5-4 に示す。これは VHDL 記述から各回路部の回路規模を推定したものである。この中で P/S Conversion はバス仕様に依存するものであり、比較判定関数に関連するものは P/S Conversion を除く F/F 数 3、LUT 数 12 と見積もられる。この値は全体回路の LUT 数 63400 と比較して十分小さく、実用的であることが確認された。

表 5-4　RDU 回路別規模見積もり

RDU 回路部	F/F 数	LUT 数
Master Right Control	1	2
Error function	1	4
P/S Conversion	42	21
Compare	1	3
Interconnect 出力(I/O 出力)	0	3

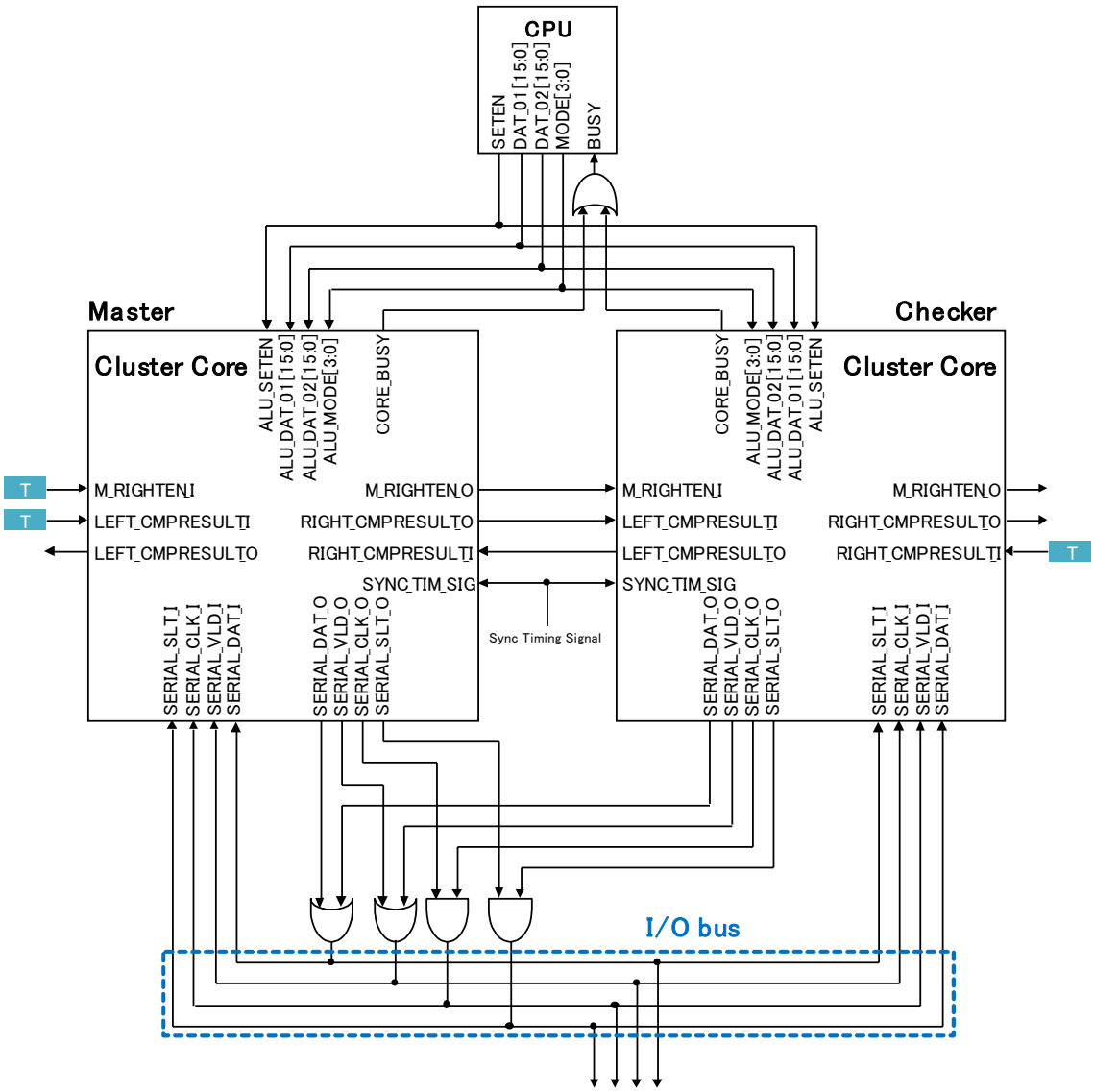

CPU
SETEN
DAT_01[15:0]
DAT_02[15:0]
MODE[3:0]
BUSY
Master
Checker
Cluster Core
Cluster Core
ALU_SETEN
ALU_DAT_01[15:0]
ALU_DAT_02[15:0]
ALU_MODE[3:0]
CORE_BUSY
CORE_BUSY
ALU_MODE[3:0]
ALU_DAT_02[15:0]
ALU_DAT_01[15:0]
ALU_SETEN
T
T
T
M_RIGHTEN_I
LEFT_CMPRESULT_I
LEFT_CMPRESULT_O
M_RIGHTEN_O
RIGHT_CMPRESULT_O
RIGHT_CMPRESULT_I
SYNC_TIM_SIG
Sync Timing Signal
M_RIGHTEN_I
LEFT_CMPRESULT_I
LEFT_CMPRESULT_O
SYNC_TIM_SIG
M_RIGHTEN_O
RIGHT_CMPRESULT_O
RIGHT_CMPRESULT_I
SERIAL_SLT_I
SERIAL_CLK_I
SERIAL_VLD_I
SERIAL_DAT_I
SERIAL_DAT_O
SERIAL_VLD_O
SERIAL_CLK_O
SERIAL_SLT_O
SERIAL_DAT_O
SERIAL_VLD_O
SERIAL_CLK_O
SERIAL_SLT_O
SERIAL_SLT_I
SERIAL_CLK_I
SERIAL_VLD_I
SERIAL_DAT_I
I/O bus

図 5-8　二重冗長構成

表 5-5　二重冗長構成における Error Function

	左 入 力 (固定)	マスタ 自己比較結果	チェッカ 自己比較結果	右入力 (固定)	動作
①	T	T	T	T	正常時：マスタが出力
②	T	T	F	T	チェッカが不一致検出：マスタが出力
③	T	F	T	T	マスタが不一致検出：チェッカに移譲 チェッカが出力：(マスタ比較器エラー)
④	T	F	F	T	Bus error：出力なし。エラー通知
⑤	T	－	T	T	マスタ権移譲後のチェッカ一致：チェッカが出力
⑥	T	－	F	T	マスタ権移譲後のチェッカ不一致：Bus error。出力なし。エラー通知

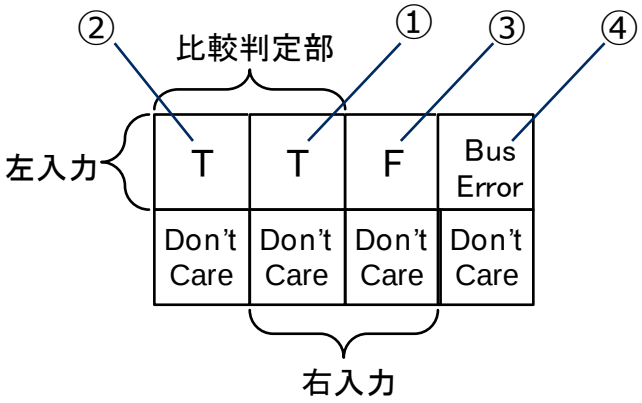


図 5-9　マスタ用比較判定関数

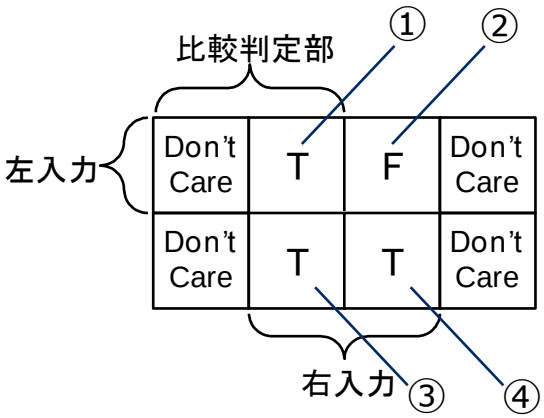
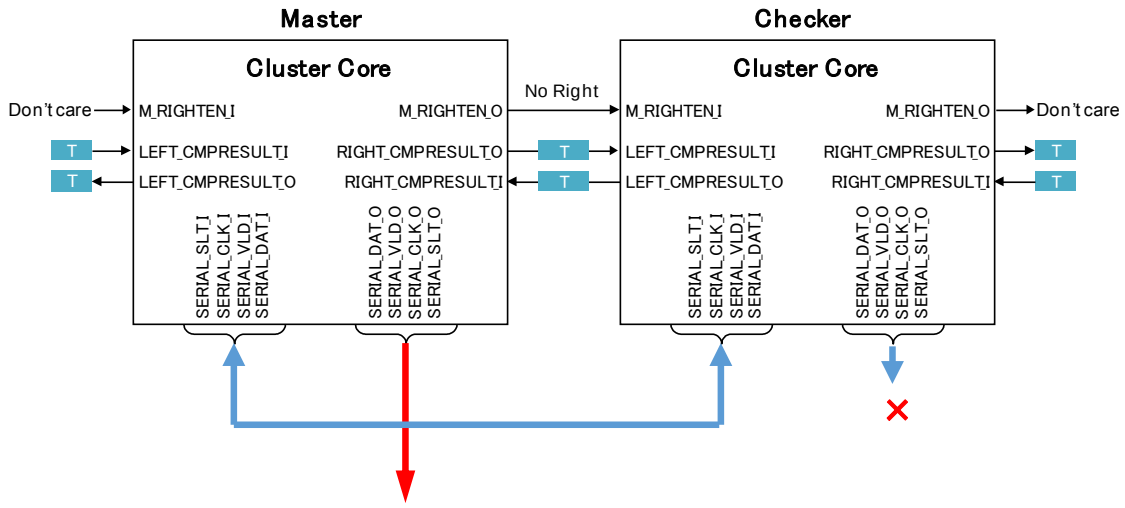


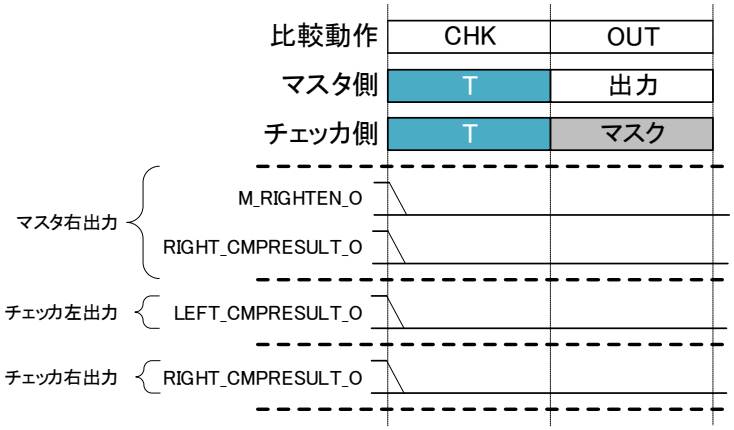
図 5-10　チェッカ用比較判定関数

本構成の動作を図 5-11～図 5-16 に示す。GPE フレームワークを用いて構成した回路がこれらに示した通りに動作することを論理シミュレーションにより確認した。冗長構成管理は表 5-5 に示した Error Function に基づき、同表に示した①～⑥のケースに応じて次のように実行される。図 5-11 に示す正常時にはマスタ権は Master 側の Cluster Core に保持され、演算結果は Master 側の Cluster Core から出力される。図 5-12 は Cheker 側の Cluster Core で演算結果を照合した結果不一致が検出された場合である。この場合は図 5-9 に示したように演算処理が正しいか Bus Error が生じたかの区別がつかない。このため図 5-12 ではマスタ権は Cluster Core に保持され、演算結果は Master 側の Cluster Core から出力されるが、系全体としてリブート等の処置をするのが望ましい。図 5-13 は Master 側の Cluster Core で演算結果を照合した結果不一致が検出された場合である。この場合は図 5-9 の判定基準に基づき Cheker 側の Cluster Core にマスタ権が委譲され、演算結果は Checker 側の Cluster Core から出力される。以降は Checker 側の Cluster Core でマスタ用の比較判定関数が用いられる。図 5-14 は Master 側の Checker 側の Cluster Core 双方で演算結果を照合した結果不一致が検出された場合である。この場合は図 5-9 の判定基準に基づき Bus error でありシステムとしての継続動作は危険であることから系全体としてリブート等の処置をするのが望ましい。比較判定関数に基づき Cheker 側の Cluster Core にマスタ権が委譲されるが、演算結果は Checker 側の Cluster Core においてもマスクされてシステムへの委譲出力は阻止される。図 5-15 は Checker 側の Cluster Core にマスタ権が委譲されたのち、旧 Master 側の Cluster Core で演算結果の不一致が検出された場合である。この場合は既に旧 Master 側の Cluster Core の判断には信憑性は無いため、Checker 側の Cluster core は自身の比較判定関数に基づき演算結果の出力を継続する。図 5-16 は Checker 側の Cluster Core にマスタ権が委譲されたのち、旧 Checker（現 Checker でもある）側の Cluster Core で演算結果の不一致が検出された場合である。この場合は比較判定関数からもはやどの Cluster Core の判断にも信憑性は無いため、系全体としてリブート等の処置をするのが望ましい。

以上の処理は SERVIS-2 プロジェクトにより SERVIS-2 衛星に搭載したフォールトトレラント計算機（CRAFT）により軌道上にて動作実績が得られた[26, 27, 28]。この結果を受けて人工衛星搭載用汎用オンボードコンピュータとして実用化されている[61, 64]。



(a)



(b)

○マスタ右出力

マスタ権の移譲（M_RIGHTEN_O）　　=L(No Right)：マスタ権の移譲無し

マスタ比較結果（RIGHT_CMPRESULT_O）　　=L(T)：マスタデータ比較一致

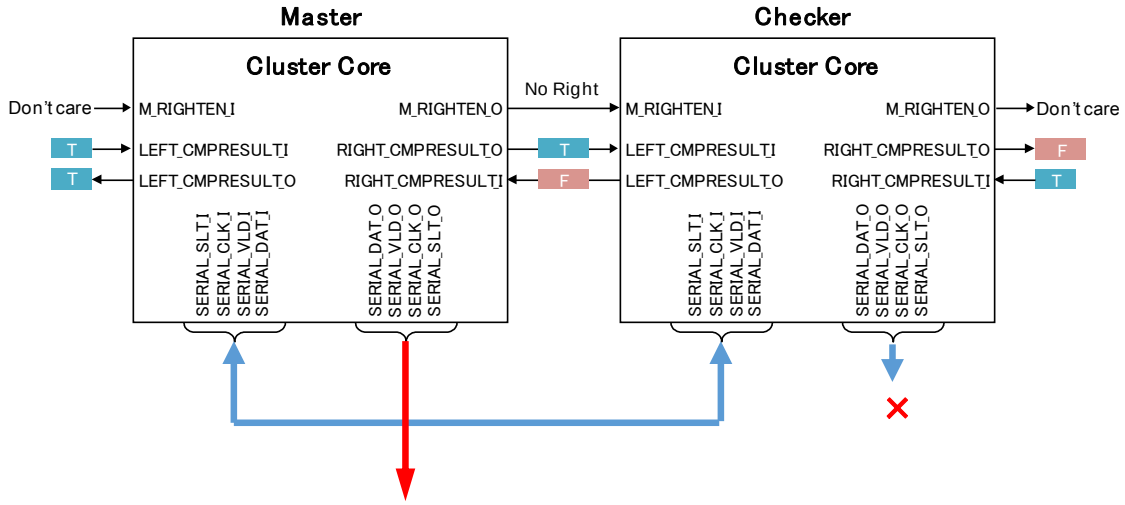
○チェッカ左出力

チェッカ比較結果（LEFT_CMPRESULT_O）　　=L(T)：チェッカデータ比較一致

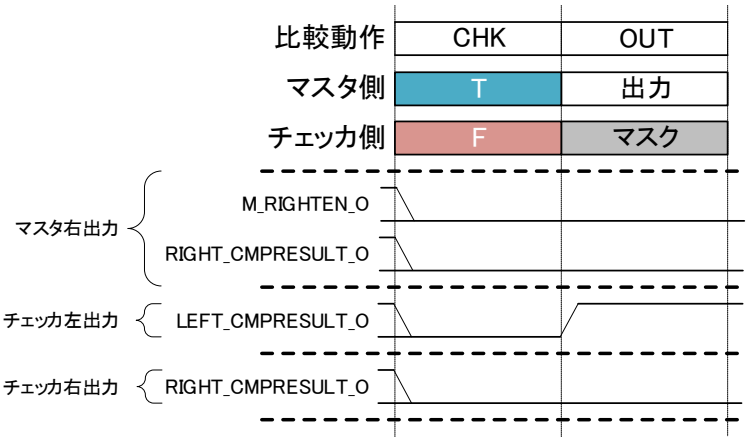
○チェッカ右出力

バスエラー結果（RIGHT_CMPRESULT_O）　　=L(T)：チェッカデータ比較一致

図 5-11　正常時出力（a）信号経路、(b）出力信号



(a)



(b)

○マスタ右出力

マスタ権の移譲（M_RIGHTEN_O）　　=L(No Right)：マスタ権の移譲無し

マスタ比較結果（RIGHT_CMPRESULT_O）　　=L(T)：マスタデータ比較一致

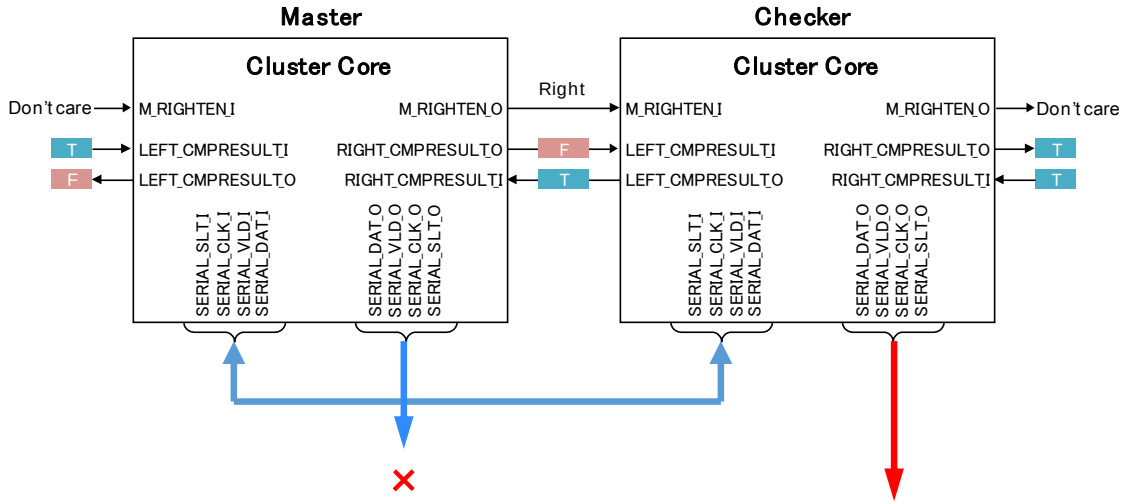
○チェッカ左出力

チェッカ比較結果（LEFT_CMPRESULT_O）　　=H(F)：チェッカデータ比較不一致

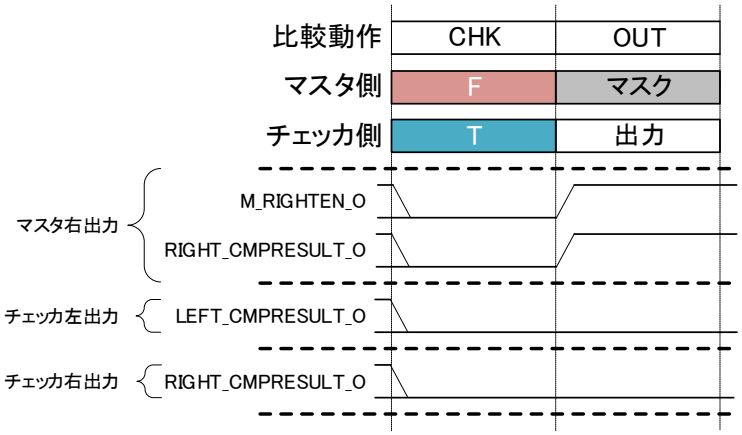
○チェッカ右出力

バスエラー結果（RIGHT_CMPRESULT_O）　　= H(F)：チェッカデータ比較不一致

図 5-12　チェッカによる不一致検出時出力（a）信号経路、(b）出力信号



(a)



(b)

○マスタ右出力

マスタ権の移譲（M_RIGHTEN_O）　　=H(Right)：マスタ権の移譲有り

　マスタ比較結果（RIGHT_CMPRESULT_O）　　=H(F)：マスタデータ比較不一致

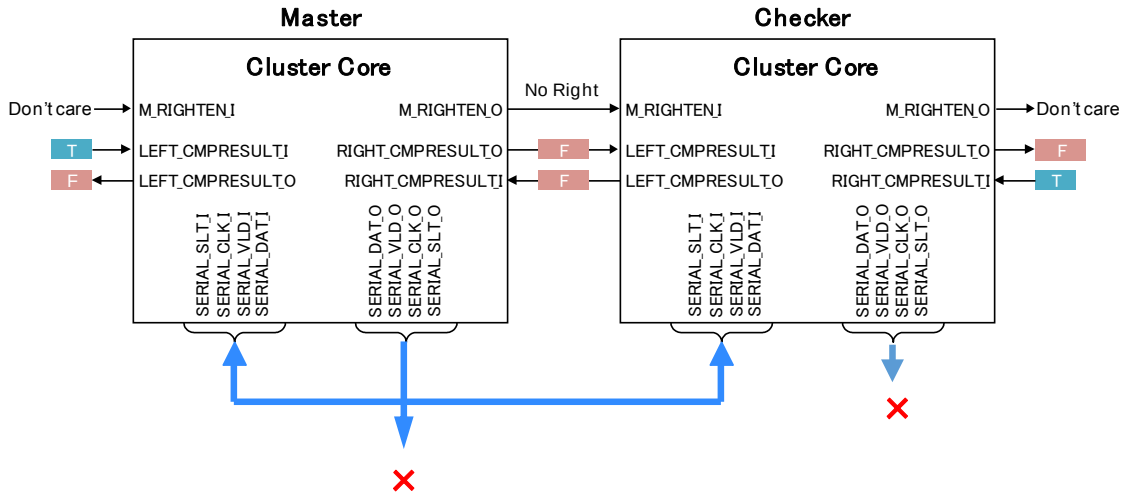
○チェッカ左出力

　チェッカ比較結果（LEFT_CMPRESULT_O）　　=L(T)：チェッカデータ比較一致

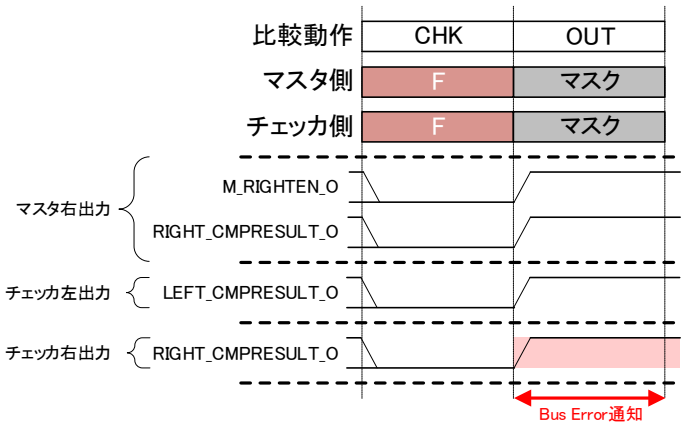
○チェッカ右出力

　バスエラー結果（RIGHT_CMPRESULT_O）　　=L(T)：チェッカデータ比較一致

図 5-13　マスタによる不一致検出時出力（a）信号経路、(b) 出力信号



(a)



(b)

○マスタ右出力

マスタ権の移譲（M_RIGHTEN_O）　　=H(Right)：マスタ権の移譲有り

　マスタ比較結果（RIGHT_CMPRESULT_O）　　=H(F)：マスタデータ比較不一致

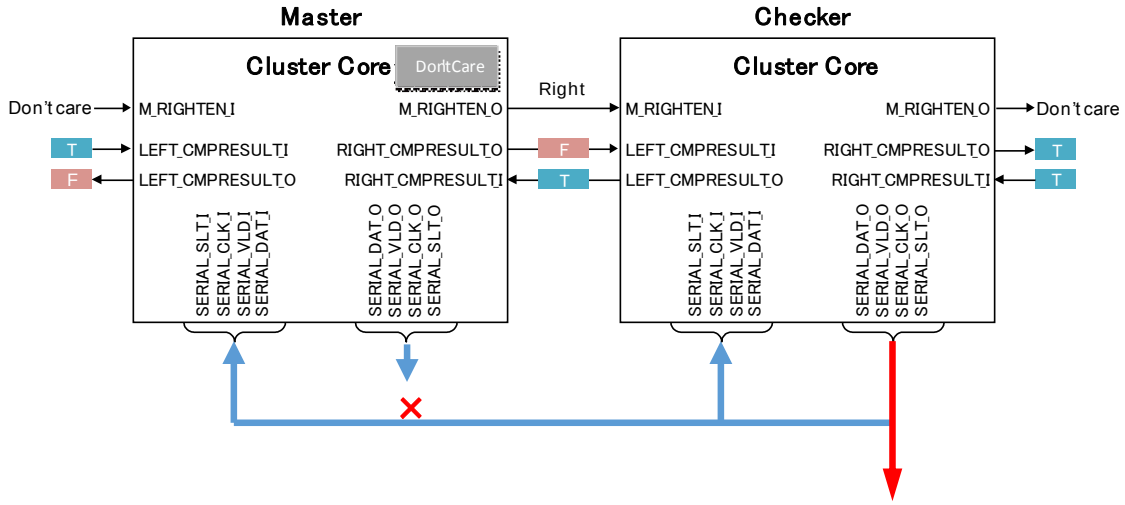
○チェッカ左出力

　チェッカ比較結果（LEFT_CMPRESULT_O）　　=H(F)：チェッカデータ比較不一致

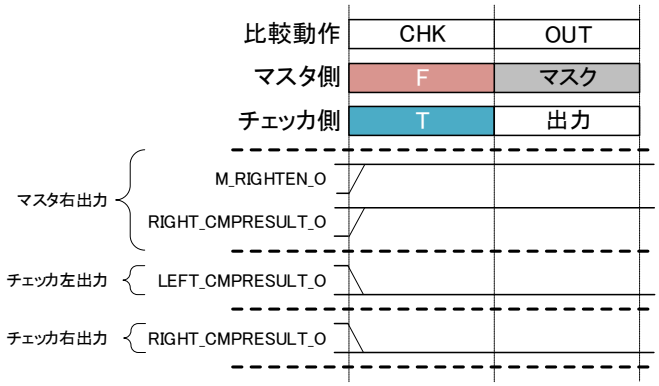
○チェッカ右出力

　バスエラー結果（RIGHT_CMPRESULT_O）　　=H(F)：チェッカデータ比較不一致

図 5-14　バスエラー時：出力無し (a) 信号経路、(b) 出力信号



(a)



○マスタ右出力

マスタ権の移譲（M_RIGHTEN_O）　　= H(Right)：マスタ権の移譲有り（固定）

マスタ比較結果（RIGHT_CMPRESULT_O）=H(F)：マスタデータ比較不一致（固定）

○チェッカ左出力

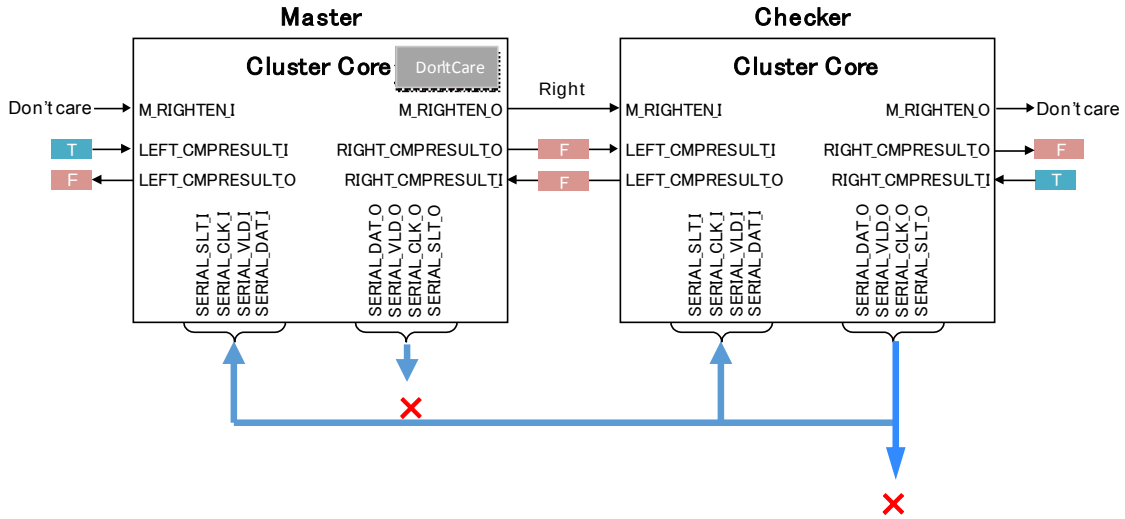
チェッカ比較結果（LEFT_CMPRESULT_O）=L(T)：チェッカデータ比較一致

○チェッカ右出力

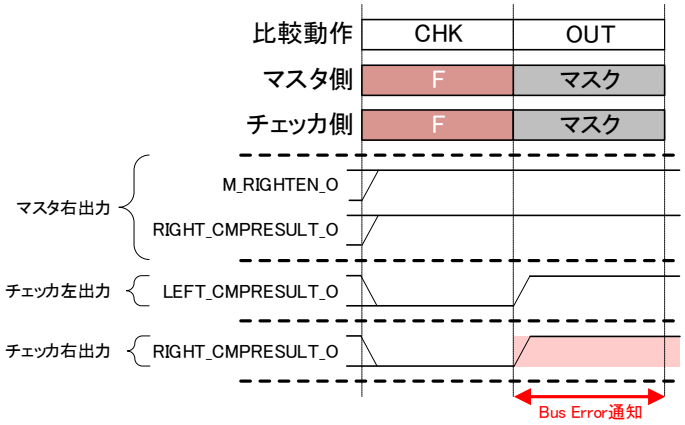
バスエラー結果（RIGHT_CMPRESULT_O）= L(T)：チェッカデータ比較一致

図 5-15　マスタ権移譲後のチェッカ一致時：チェッカが出力

(a) 信号経路、(b) 出力信号



(a)



(b)

○マスタ右出力

マスタ権の移譲（M_RIGHTEN_O）　　= H(Right)：マスタ権の移譲有り（固定）

マスタ比較結果（RIGHT_CMPRESULT_O）= H(F)：マスタデータ比較不一致（固定）

○チェッカ左出力

チェッカ比較結果（LEFT_CMPRESULT_O）=H(F)：チェッカデータ比較不一致

○チェッカ右出力

バスエラー結果（RIGHT_CMPRESULT_O）= H(F)：チェッカデータ比較不一致

図 5-16　マスタ権移譲後のチェッカ不一致時：出力無し

(a) 信号経路、(b) 出力信号

第6章
ナノブリッジを活かすアーキテクチャの評価

6.1. 粒度別階層化設計フレームワーク

第 3 章に述べた GPE アーキテクチャ、および第 4 章に述べた粒度別階層化設計手法を統合し、IoT アプリケーションのデバイスコンピューティングとしてセンサノードに埋め込まれた PE を設計できるようにするため、Generic Processing Element (GPE)としてデザインフレームワークを確立した。ここではそれを Generic Processing Element (GPE) Design Framework と称している。GPE Design Framework は以下の 3 つの特徴を有している。

a) 粗粒度の関数を状態に応じて最適接続する Processing Element (PE)アーキテクチャ

b) 粗粒度の関数を使いまわして動作合成する機能との統合

c) 設計プロセスの確立

また、高信頼性システムへの適用指針を確立し、多数決方式を凌駕する冗長管理方式が構成可能であることを第 5 章で示した。

この実装例として、関数を埋め込んだ典型的な赤外線センサのブロック図を図 6-1 に示す。本ブロック図を参照し、上述した GPE デザインワークフレームを二種類の実験を行って評価した。本実験は消費電力と実装規模の削減効果を検証し、以下の三点の見通しを得ることを目的として実施した。

a) GPE Desing Framework でナノブリッジが使いこなせること

b) センシングデバイスにナノブリッジを活用したプロセッサを内蔵できること

c) 再現性のある消費電力の低減メカニズム

最初の実験では人工衛星搭載用赤外線センサをモチーフとし、赤外線センサの出力信号処理をナノブリッジ FPGA に実装して評価した。もう一つの実験では、一つ目の実装実験の結果得られた低消費電力化の効果を生み出すメカニズムを調査するために、従来の FPGA により画像処理を実装して評価した。この結果従来の FPGA にも有効性が認められた。

粗粒度関数定義の実用性については、軌道上で運用されている人工衛星搭載機器の運用状況を調査した。この結果、本デザインフレームワークによるセンサ信号処理回路の実用性が確認できた。これについては Appendix D に示す。

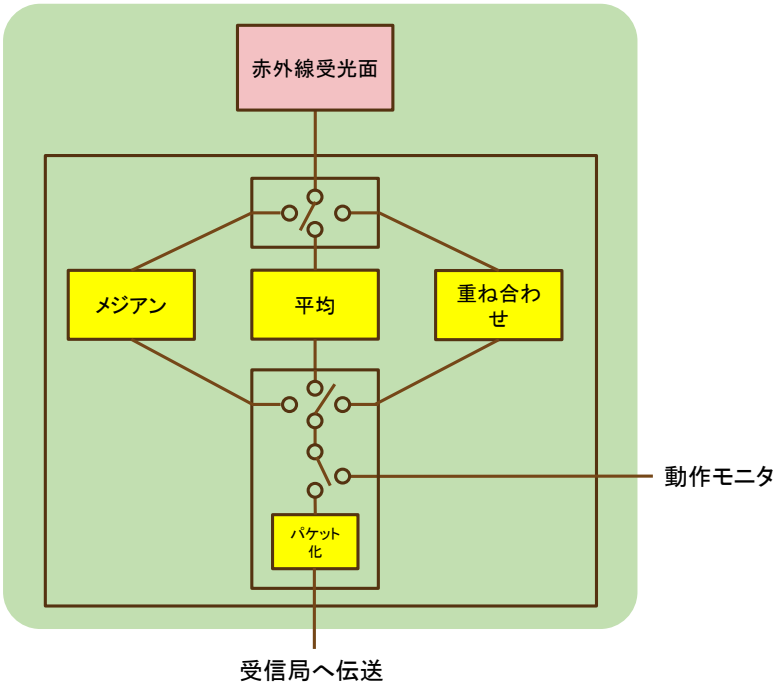


図 6-1　関数を埋め込んだ赤外線センサ

6.2. GPE Design Framework による消費電力の削減

赤外線センサシステムは、地球観測衛星に搭載されると共に、惑星探査の一環として惑星や小惑星の性質や形成過程を調べるためによく用いられる。この実験では人工衛星に搭載する赤外線センサの出力信号処理装置をモチーフとし、ナノブリッジを用いた FPGA をセンシングデバイスに埋め込む見通しを得るべく、以下の目標の実現性を検証した。

第 1 に、センシングデバイスに埋め込む PE にはセンサから出力する信号を止めずに処理する Wire rate Processing と称する信号処理性能を必要とする。Wire rate で処理をすることにより、センサ出力信号を一旦外付けメモリ素子に入れる必要が無くなることから外付けメモリ素子を削減できる。この結果、小型化、および軽量化が可能となる。さらに、メモリはソフト・エラーの発生源でもあることから、メモリが削除できればソフト・エラーの発生源も削除され、信頼度の高いセンサシステムを構築することができる。すなわち、センシングデバイスに埋め込む PE は、外付けメモリ素子を用いずに Wire rate でセンサ出力信号を処理できる必要がある。

第 2 に、赤外線検出器の出力特性を校正するために柔軟なプログラミングができる必要がある。検出信号のダイナミックレンジを向上させ、出力信号を正規化するために、赤外線センサの信号処理装置にはいくつかの種類の校正機能が必要である。白傷などの正常動作しない画素の埋め合わせ処理も必要である。装置に搭載する赤外線イメージセンサを設計するには、シミュレーション作業が必須であるため、高級プログラミング言語（例えば C

言語）によるプログラミングができることが望ましい。

3 番目の要求は、低消費電力化と軽量化である。従来の MCU に使われているプログラムメモリや、従来の FPGA に使われているコンフィギュレーションメモリを用いずに、これらの要求を満たすことが目標である。

赤外線センサシステムは、赤外線イメージセンサの信号出力強度を補償する前処理機能を必要とし、地上局にデータを伝送する通信回線チャネルの限られた容量を活用すべくし画像圧縮を行う等の処理を行って高品質のデータ伝送を行う。典型的な人工衛星搭載用イメージセンサの信号処理部のブロックダイヤグラムを図 6-2 に示す [55, 56, 57, 58]。現状の人工衛星搭載用のセンサの各検出素子は 16 ビットのデジタルデータ出力ポートを備えているものとしている。各画素の感度は、Static Random Access Memory（SRAM）またはフラッシュメモリを用いて実装されたテーブルに格納されたパラメータを使用して補正される。補正されたデータは 16 ビットのデジタルデータとして表される要素の輝度として扱われる。これらの補正されたデータは感度補正を行い、感度が低い赤外線センサなどではいくつかのフレームにわたって重ね合わせて積分の効果を得ることにより感度を高める処理がなされた後、後処理のために必要に応じて再配置される。これらのオンボード較正処理の後、各エレメントの輝度は 16 ビットのデジタル数（Digital Number: DN）として正規化される。この後、各データをタイルと呼ぶ区画に分割し、データ量を削減するためにビニングやデータ圧縮処理を施す。圧縮したデータは宇宙機システムの事実上の標準となっている CCSDS 勧告[51]に従ってフォーマットされ、地上局に伝送される。今回の実験では、図 6-2 に示す機能のうち、Luminance compensation の処理をセンシングデバイスに埋め込むことのできる目途を得ることを目標とした。

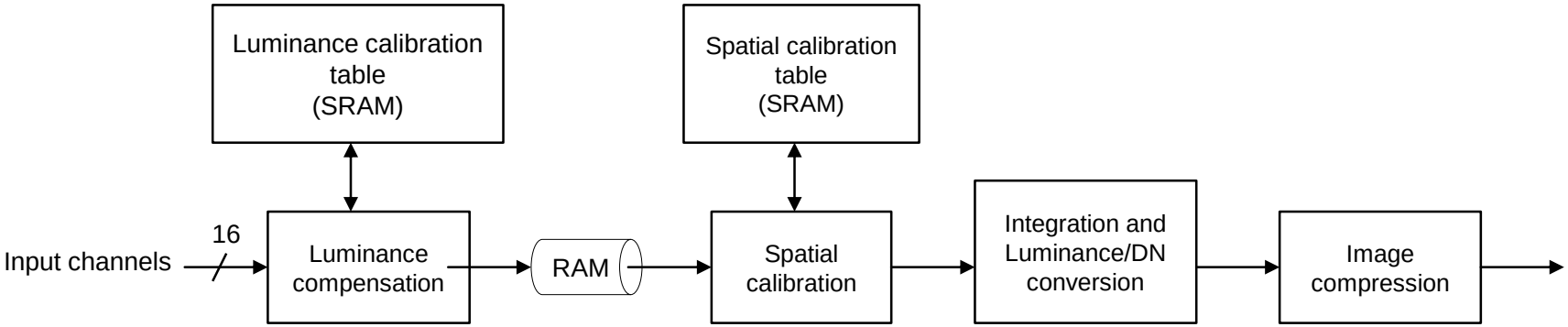


図 6-2　典型的な人工衛星搭載用イメージセンサの信号処理部のブロック図

人工衛星に搭載した赤外線センサから地上局に赤外線センサデータを伝送するためのオンボードの信号処理としては、次のような要求により、高速かつコンパクトな回路が必要となる。第 1 に、地上局への通信チャネルの限られた伝送容量を活用して高品質のデータを得るために、人工衛星に搭載した赤外線検出器の赤外線イメージセンサの強度を補償する出力信号較正機能が必要とされる。第 2 に、センサシステムのリソースが限られているにもかかわらず、赤外線イメージセンサデータの利用者からは画像データ品質の劣化が無いことを要求される。したがって、人工衛星に搭載する電子回路部には、サイズ、質量、消費電力を抑えつつ、無損失の画像データ圧縮機能を実現する必要がある。

金星気象観測衛星であるあかつきや、小惑星探査衛星であるはやぶさ 2 等の現状の一般的な実装では、MCU ボードと FPGA ボードの組み合わせを採用している [42, 48, 49]。これらのボードは、Wire rate の処理を要する機能は FPGA ボードに実装し、フィールドで書き換えが想定される機能は MCU ボードに実装するなどの使い分けがなされている。一方、将来のミッションでは、より多くのセンサを収容するために、更に小さいサイズとより低消費電力が要求される。この今後想定される需要を満たすために、ナノブリッジを活用した PE をセンシングデバイスに埋め込み事ができれば大変有用である [49]。

ナノブリッジを用いた FPGA はコンフィギュレーションメモリなしでハードウェアプログラミングが可能になるため、論理回路接続上のソフト・エラーは完全に排除される。これは、従来の書き換え可能な FPGA では得られない特長であり、宇宙機システムのアプリケーションにとって注目すべき利点である。コンフィギュレーションメモリが無くなることで従来の書き換え可能な FPGA と比較して消費電力の削減も期待される。上述の MCU ボードと FPGA ボードに実装していた処理を纏めてナノブリッジ FPGA に 1 チップ化できるようにすることが目標となる。

本実験のモチーフとした、あかつきに搭載している赤外線イメージセンサ[50]のデジタル信号処理装置を図 6-4 に示す。本装置には赤外線画像検出器の感度を向上させるためにいくつかの専用の演算機能を実装している。これらの関数は重ね合わせ、平均、およびメジアンであり、最終的に正規化される。これらの機能は、検出器ピクセルの分散を補償するのに最も効果的な演算はどれであるかをオンボードで評価するために実装されたものであり、既開発機器では前述のように MCU ボードと FPGA ボードの 2 枚で構成されている。MCU ボードには JAXA 認定の 64 ビットマイクロプロセッサ HR5000 を搭載しており、FPGA ボードには 2 つの ACTEL 社製 RTSX72SU FPGA を搭載している。

上記の構成を模擬した実験装置のブロック図を図 6-5 に示す。この実験装置では赤外線センサを市販の CMOS カメラで模擬している。市販の FPGA ボードを用いて衛星システムの処理を模擬し、CMOS カメラを駆動する制御機能を実装している。地上局から送信されるコマンドを模擬してパーソナルコンピュータから IF ボードにコマンドを送信する。このコマンドによりセンサの信号処理が選択される。赤外線センサを模擬した信号とコマンドから生成された制御信号は IF ボードを介してナノブリッジ FPGA を実装した実験基板に伝

送される。ナノブリッジ FPGA で処理されたセンサ信号は IF ボードを介して FPGA ボードに戻され、出力形式の整合を取り、処理後の画像は実験装置に接続されたディスプレイに表示される。実験装置を図 6-6 に示す。上述の既存設計（A5 サイズプリント基板 2 枚）を前節で述べた GPE Design Framework を用いてナノブリッジ FPGA による 1 チップ化処理として実装した。これにより、信号処理が Wire rate の処理の目安となる 30 msec 以下で完了すれば、1 チップ化の目途を得たことになる。この実験装置が前提としている赤外線センサの模擬動作を図 6-3 に示す。

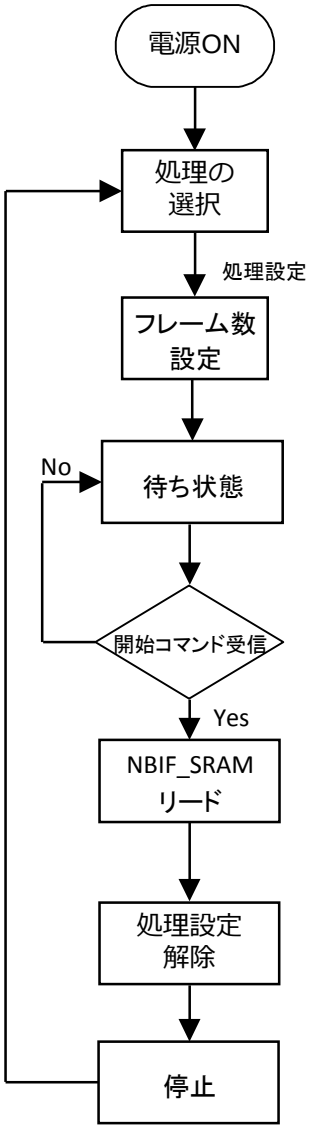
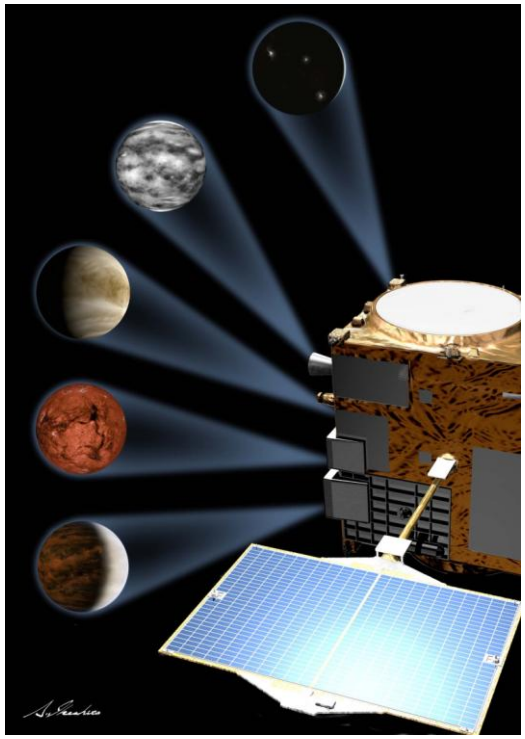
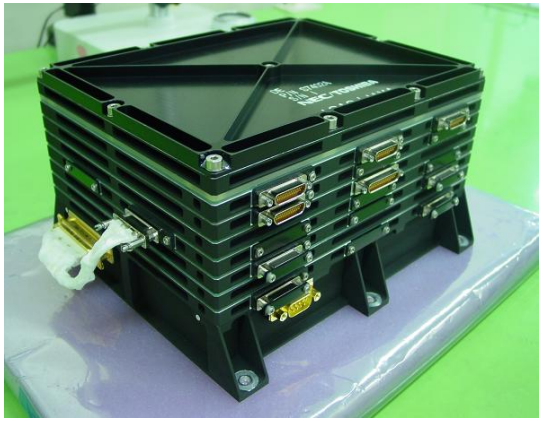


図 6-3　赤外線センサの模擬動作



© JAXA



(a)　　(b)

図 6-4　あかつき搭載用デジタル信号処理装置、(a) 金星探査衛星あかつき、(b) 既開発デジタル信号処理装置

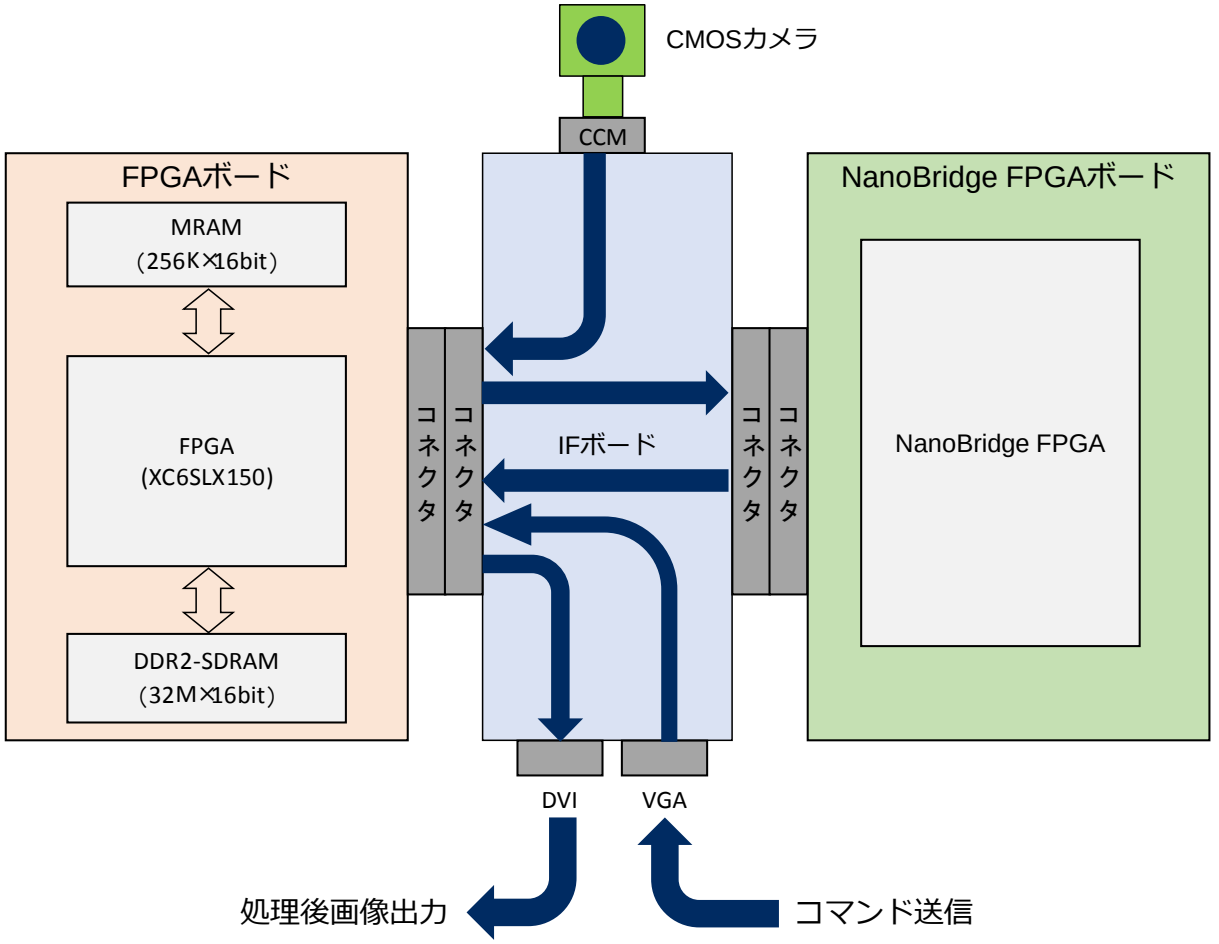


図 6-5　実験装置のブロック図

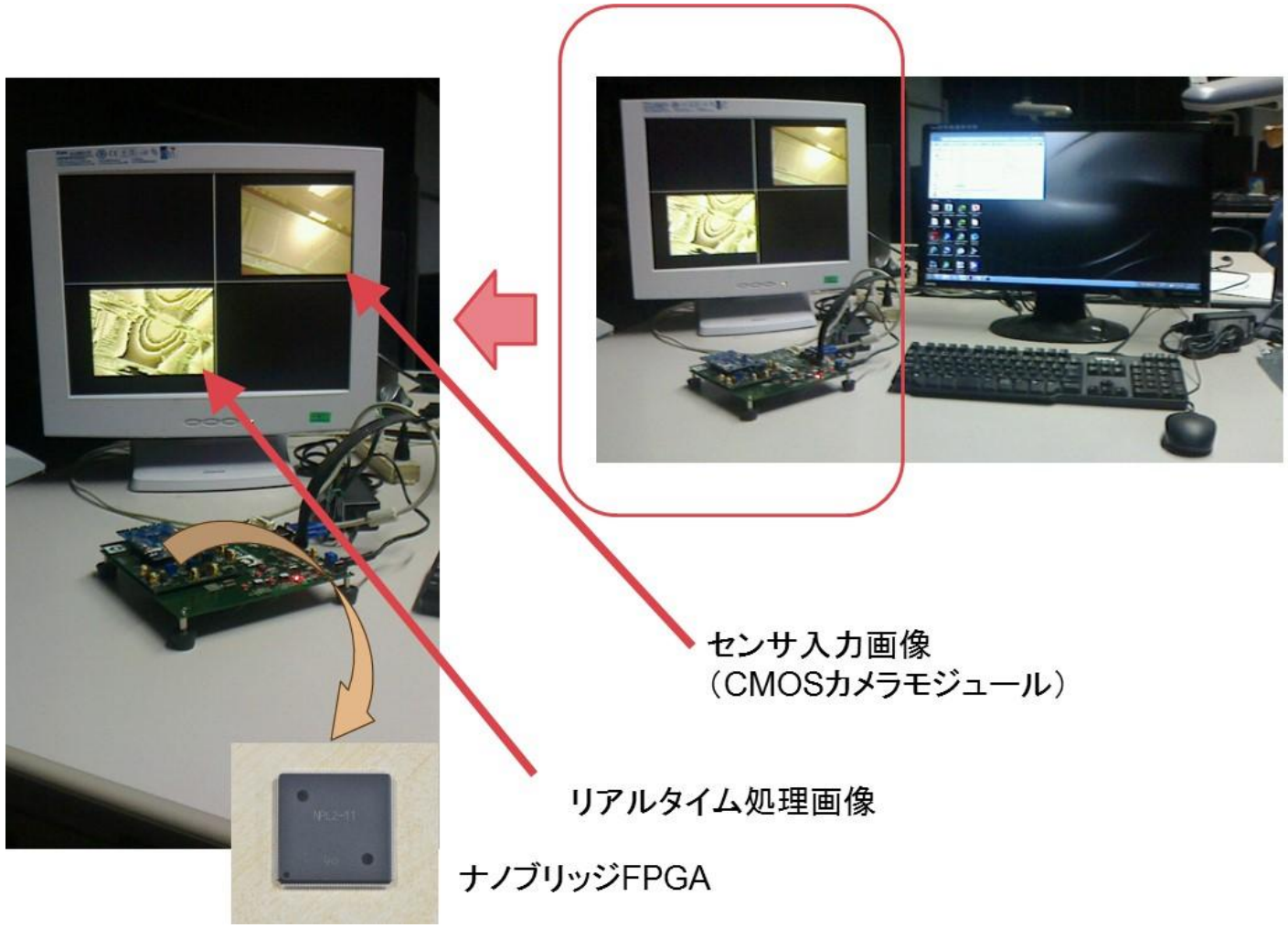


図 6-6　実験装置

ここでは第 4 章で述べた階層化設計フローによりナノブリッジ FPGA の回路を組織化する。まず Step 2 に対応する粗粒度のマクロ処理を定義すべく関数を記述する。今回機能拡張した高位合成ツールでは、各関数の前に/* Cyber func = operator */という指示を記載し、各関数を Step 3 の動作合成で活用できるよう、データベース化することをツールに指示する。この様子を図 6-7 に示す。

```
/* Cyber func = operator */
void f_Superposition(void){   ←重ね合わせ処理関数
(省略)
      sum = data + tmp;

      result = sum / 2;

(省略)
}

/* Cyber func = operator */
void f_Mean(void){   ←平均化処理関数
(省略)
}

/* Cyber func = operator */
void f_Median(void){   ←メジアン処理関数
(省略)
}
```

図 6-7　階層化設計フローの Step 2 における粗粒度関数定義

次に、これらの関数を第 4 章で述べた階層化設計フローの Step 3 により、状態遷移に応じて連結し、組織化すべく、メイン関数に埋め込む処理として記述する。この記述を図 6-8 に示す。While 文の前で指定している/* Cyber folding = 1*/という記述は、待ちの無いパイプラインを生成することを指示するものである。この関数記述を元に高位合成することにより、まず Step 2 でデータベース化された各関数が本研究により拡張された関数演算器化機能により、切り替えスイッチとして動作するマルチプレクサ、およびステートマシンと共にナノブリッジ FPGA 内の基本素子により凝集されたハードウェアとして生成される。次に、ステートマシンが生成する信号に応じて処理経路を切り替えるべく、マルチプレクサを含む各関数がデータパスにより連結され、組織化される。今回の実験ではナノブリッジ FPGA の回路はスタティックに接続したため、これらの処理はハードウェア回路として固定される。この Step 3 における高位合成の様子を図 6-9、および図 6-10 に示す。また、生成された回路を装置イメージで記述したものを図 6-11 に示す。

図 6-10 に示した組織化されたハードウェアは、今回の赤外線センサ信号処理機能のための専用の算術論理装置（ALU）を生成したことになる。図 6-7 に示すように関数中には通常の四則演算も使用しており、これらと共に重ね合わせ、平均、およびメジアンの各処理

を専用演算として備えたプロセッサを生成したことに相当する。この様子を図 6-12 に示す。

```
process IPU()          ←メイン関数
{
  u_var(2) mode;

  /* Cyber folding = 1 */
  while(1){
    mode = I_MODE;
    if(mode == 1){
      f_Superposition();      ←重ね合わせ処理の関数コール
    } else if(mode == 2){
      f_Mean();       ←平均処理の関数コール
    } else if(mode == 3){
      f_MedianWEI();      ←メジアン処理の関数コール
    } else { //スルー
      if(DATA_REQ){
        O_DATA = I_DATA;
      }
    }
  }
}
```

図 6-8　関数演算器化機能によるハードウェア回路を高位合成する C 言語プログラム

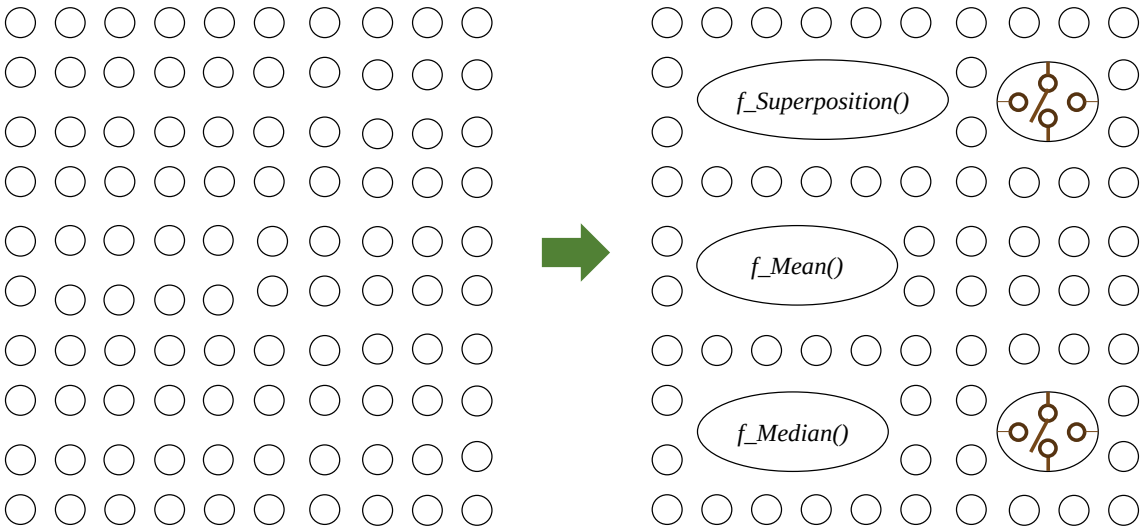


図 6-9　関数演算器化によるナノブリッジ FPGA 基本素子の凝集

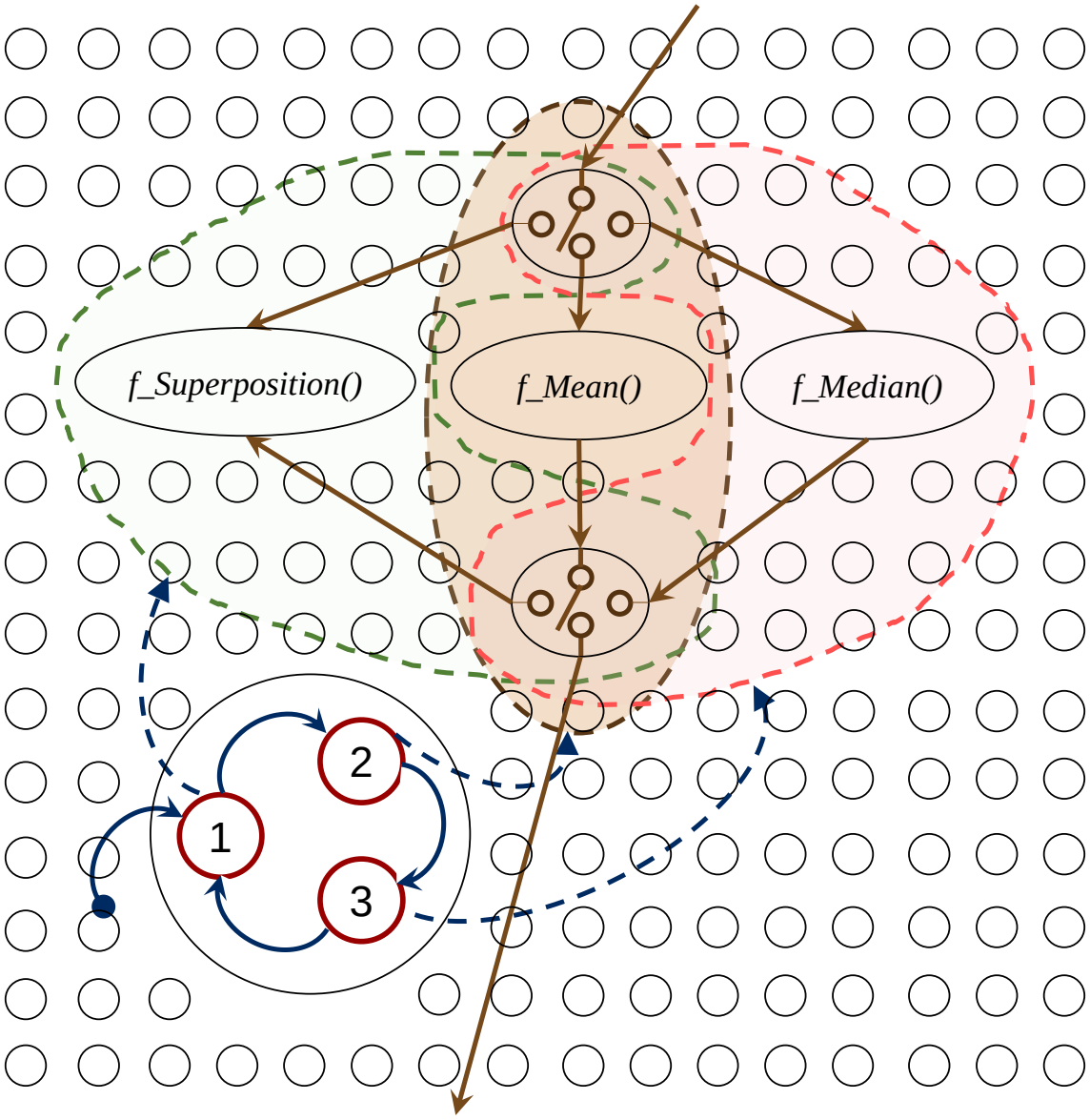


図 6-10　関数演算器化機能の組織化

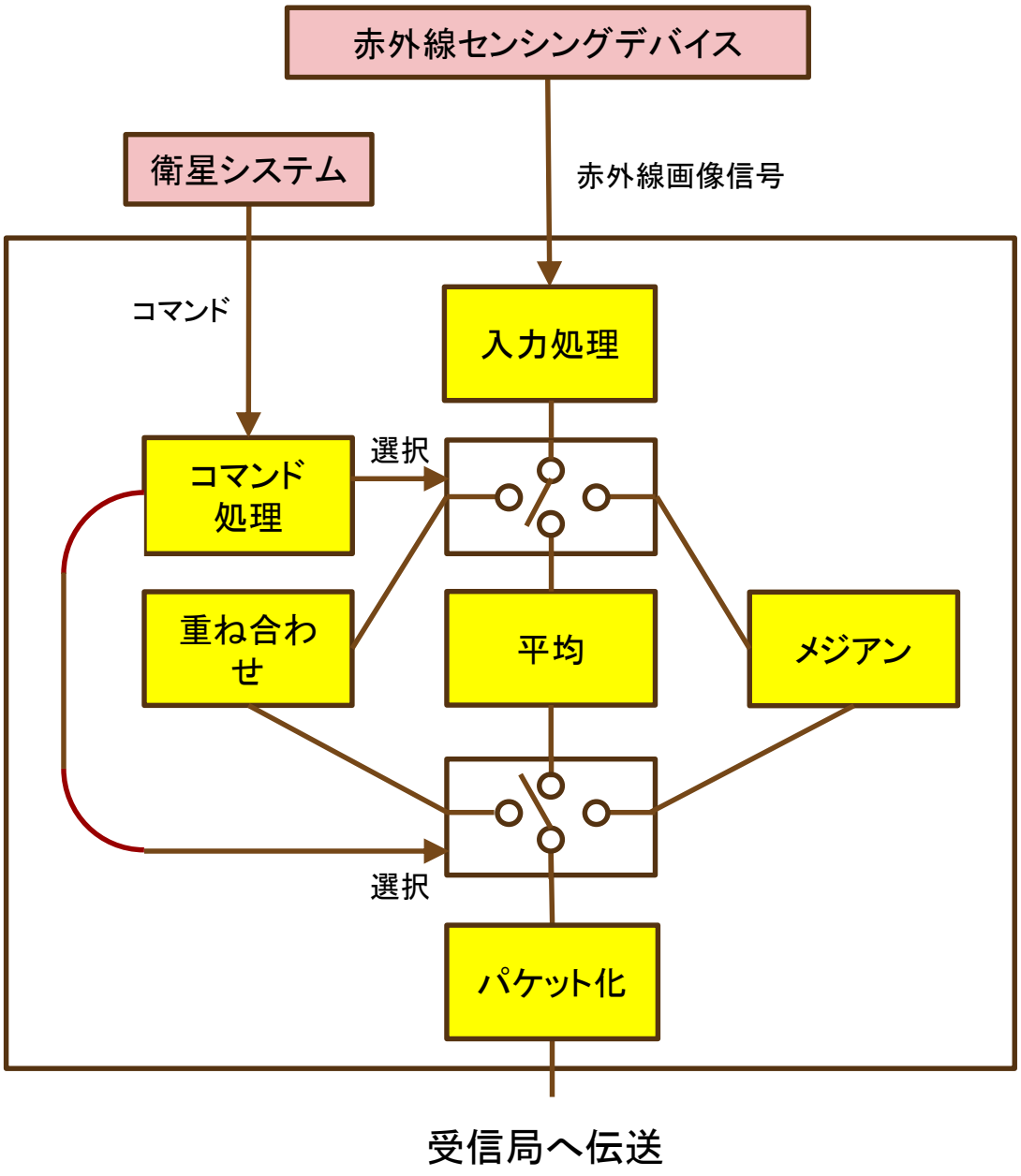


図 6-11　生成された回路の装置イメージ

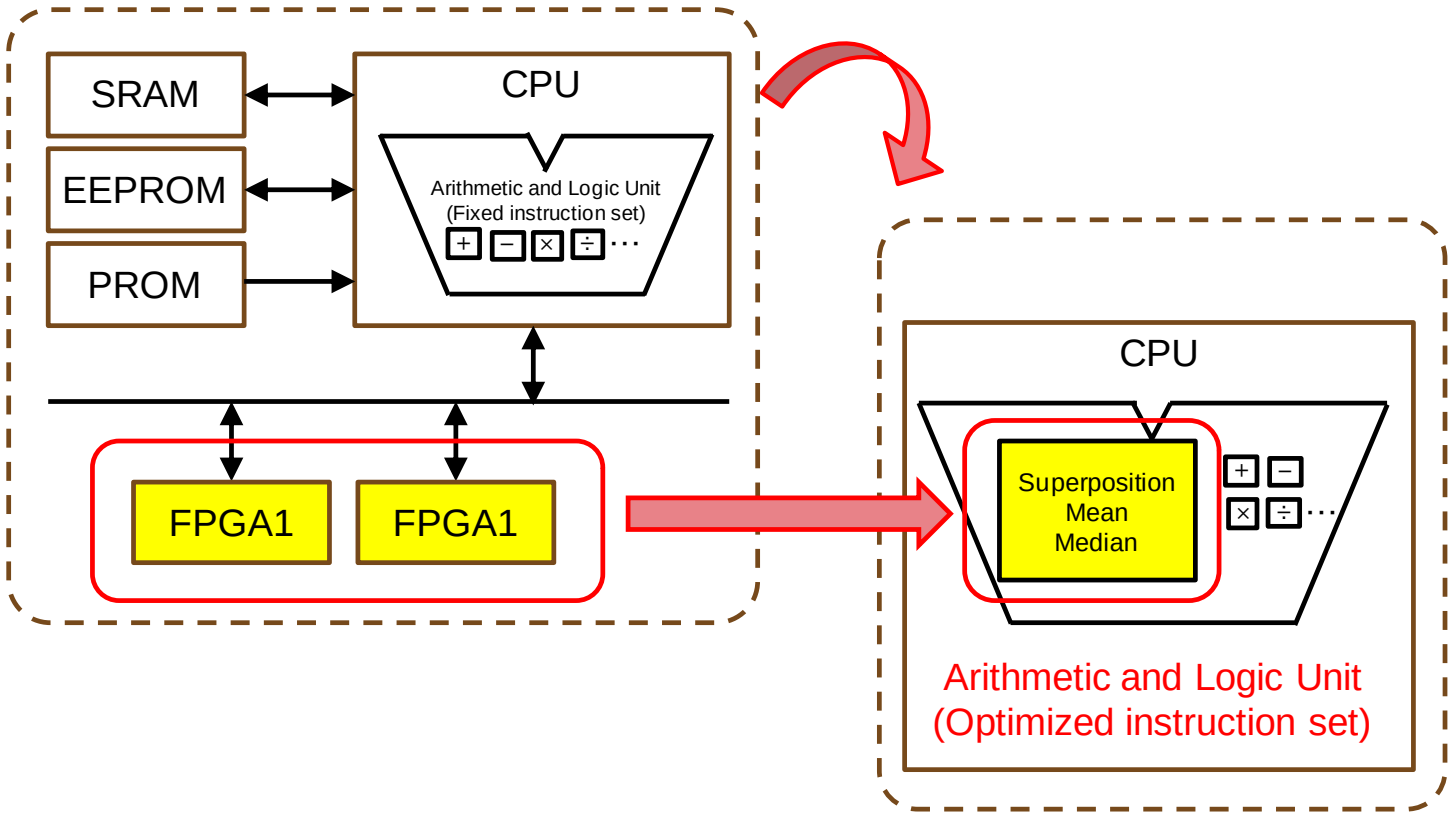


図 6-12　ナノブリッジ FPGA による専用プロセッサの 1 チップ化

本実験で試作したプロセッサアーキテクチャは、高級プログラミング言語を用いた高位合成、および動作合成によって再構成可能である。この結果、信号処理機能としては、MCU ボードと FPGA ボードの 2 枚で構成していた従来の赤外線信号処理回路と同等の機能を維持しながら、従来のプログラムメモリを伴う MCU を用いることなく 1 チップ化に成功した。これにより、本研究により高位合成／動作合成ツールの機能拡張に対応させたことにより、赤外線信号処理回路の小型化設計が実証でき、各赤外線信号検出器の出力チャネルに PE を埋め込む見通しが得られた。

元の信号プロセッサのノミナルの消費電力は装置全体の測定値を元に 4.9W と見積もられている。FPGA については、FPGA ボードに実装されている 1 つの FPGA のノミナル消費電力は、装置全体の消費電力と設計図から 500mW と見積もられている。これに対して、同等の機能をナノブリッジ FPGA を用いて 1 チップ化した PE の消費電力は LSI テスタを用いて測定した結果、図 6-13 に示すように 18mW 以下であった。ナノブリッジ FPGA の LUT の使用率は 21%である。このように回路を削減できた理由としては、プロセッサ機能全体が 1 つのチップに実装され、デバイス間を接続するインタフェース回路が削除されていることがあげられる。この小型化設計は高位合成／動作合成ツールである CyberWorkBench を用いて行い、信号処理機能は C 言語でプログラムした。これらの機能に応じて最適化された ALU を実装したことにより、最適化された回路が動作合成によって生成され、ALU は一連の演算処理で繰り返し使用された。FPGA 自体の低消費電力化があるため、これについては設計データからナノブリッジ FPGA の寄与分は 1/56 と見積もられた。これ以外の消費電力削減が、本研究にて提案した Generic な PE アーキテクチャと粒度別階層化設計手法によるものであり、この効果は 1 / 4.9 と求められた。

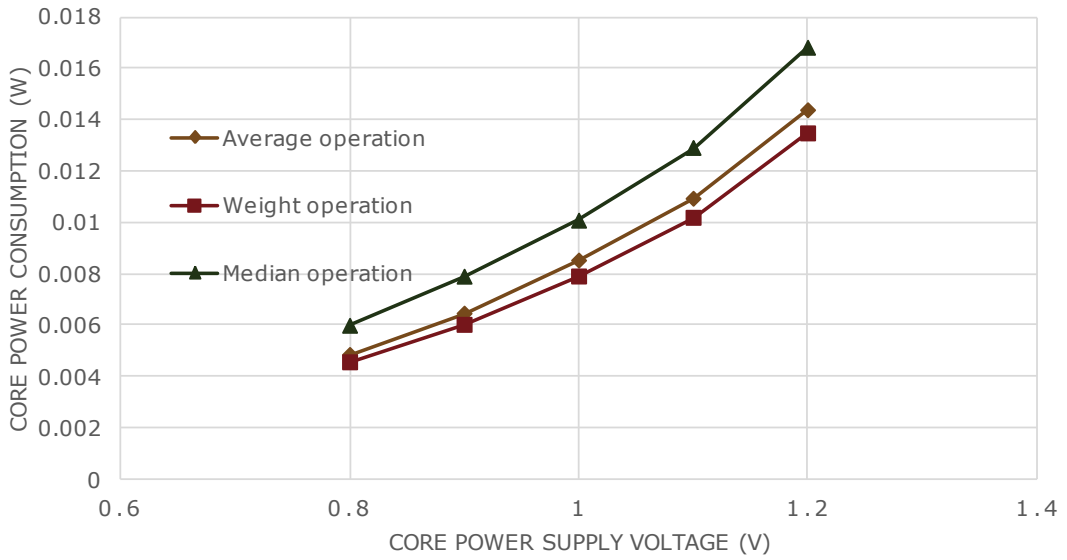
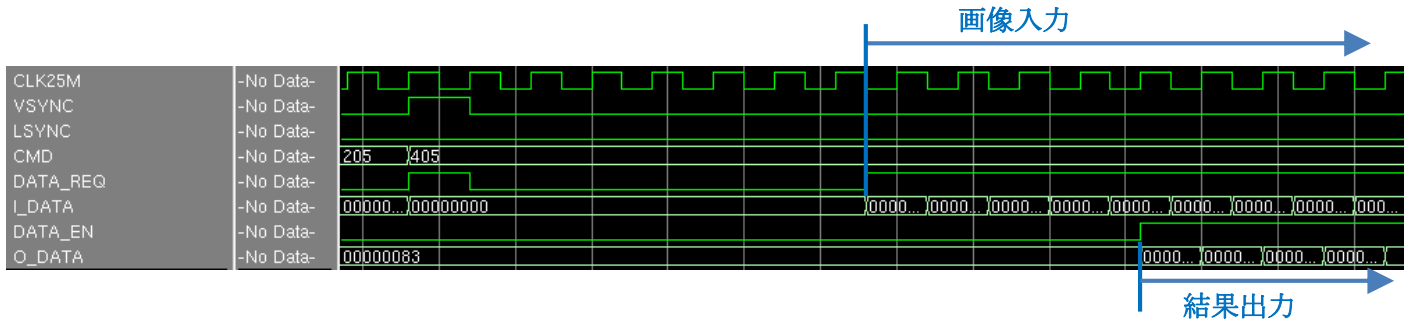


図 6-13　信号処理機能を実装した NanoBridge FPGA の消費電力測定結果

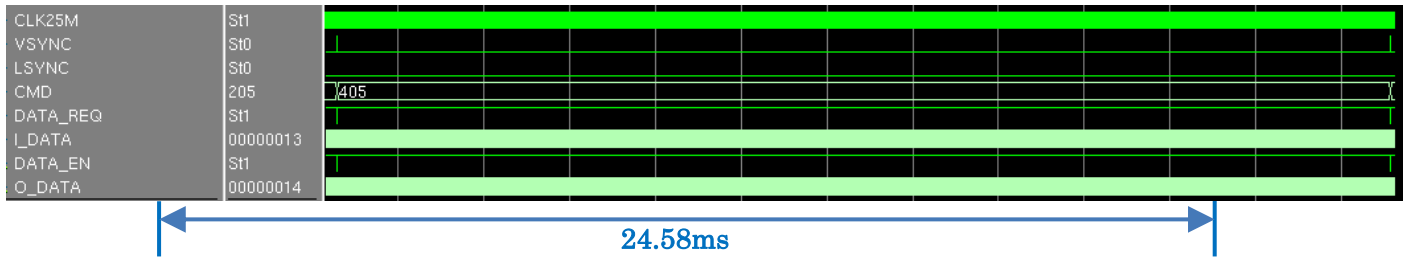
表 6-1　消費電力の削減効果

評価対象	構成	消費電力	備考
あかつき DE (Digital Electronics)	JAXA 認定 64bit MPU (HR5000) FPGA (RTSX72SU) 2 個	4.9W （ノミナル）	
NanoBridge FPGA （LSI テスタで測定）	NanoBridge FPGA 1 個	18mW 以下 @1.2V	FPGA 単体の削減効果 (500mW→18mW) GPE design framework による削減効果　1/4.9

次に、処理速度に関しては、赤外線センサの信号出力の wire rate 処理を達成するには前述したように 30 msec 以下の処理速度を達成することが目安となる。この処理速度を評価した結果を図 6-14～図 6-16 に示す。この評価結果は実際の配置配線結果を反映してシミュレーションにより測定したものである。この測定結果から重ね合わせ処理、平均化処理、およびメジアン処理が全て 24.58ms で処理可能なことを確認した。これにより、Wire rate の信号処理が実現できることが確認され、プログラムメモリやコンフィギュレーションメモリのない PE の実現性が確認できたことから、センシングデバイスに PE を埋め込む目途が得られた。



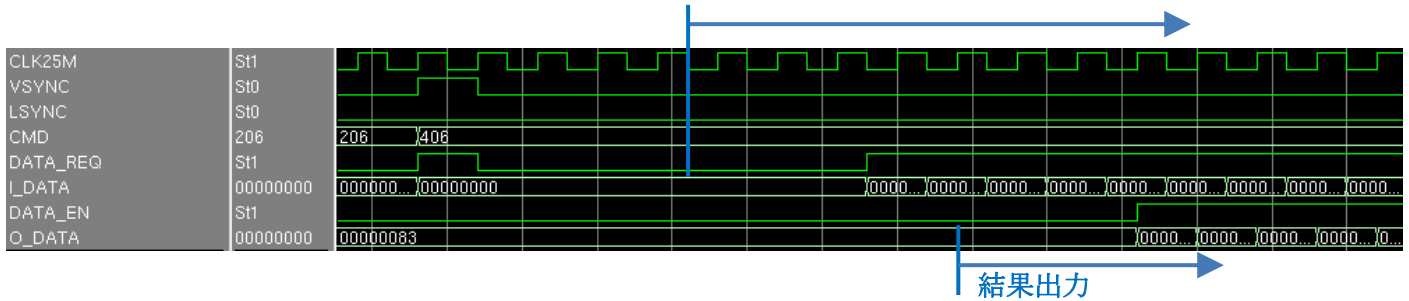
(a)



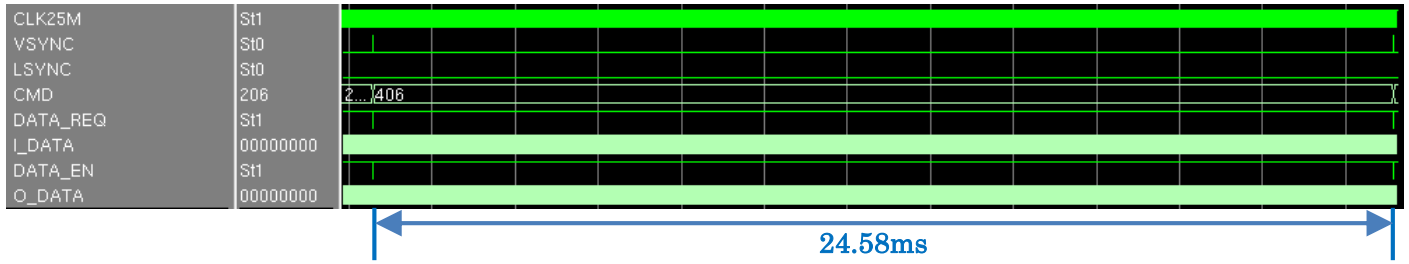
(b)

図 6-14　重ね合わせ処理速度測定結果

(a) 重ね合わせ処理拡大図、(b) 1 フレームの処理時間



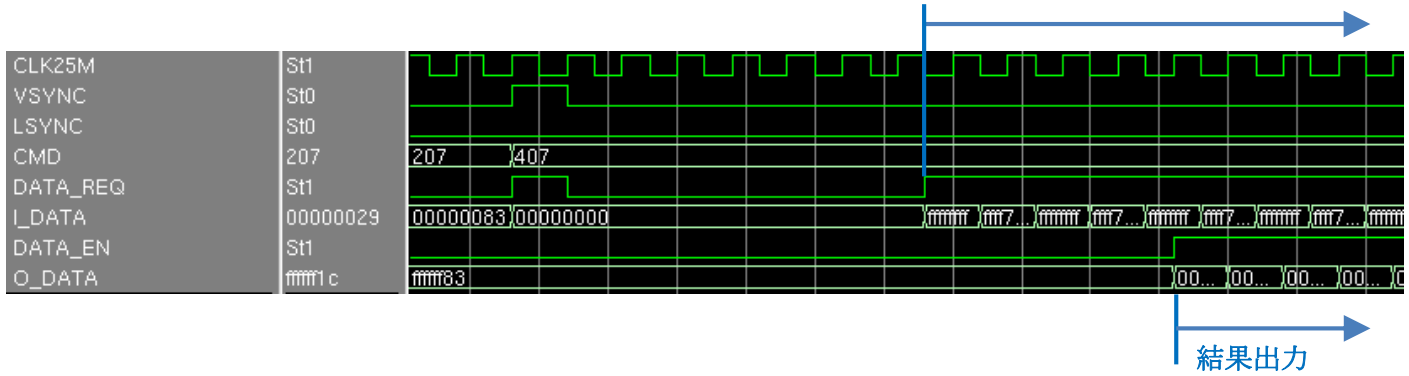
(a)



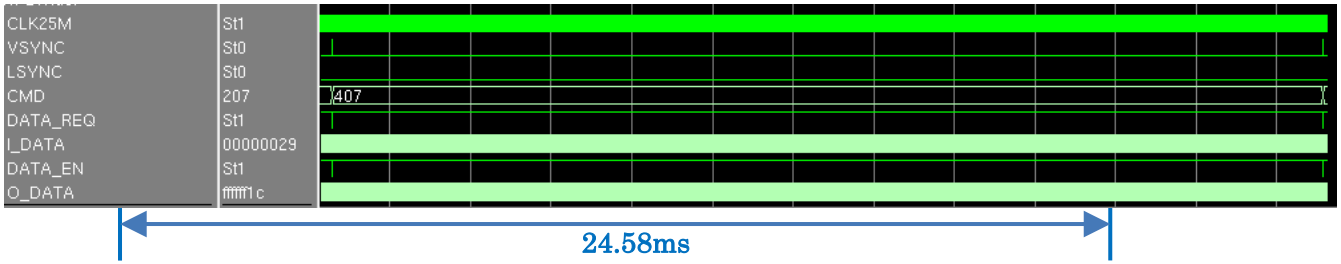
(b)

図 6-15　平均化処理速度測定結果

(a) 平均化処理拡大図、(b) 1 フレームの処理時間



(a)



(b)

図 6-16　メジアン処理速度測定結果

(a) メジアン処理拡大図、(b) 1 フレームの処理時間

6.3. GPE 設計フレームワークによる実装サイズ削減

前節に述べた低消費電力化の再現性を確認するため、どのようなメカニズムで消費電力が減ったのかを調べるべく、市販の FPGA を使用して解析を行った。近年、人工衛星の自動化処理が高度化し、深宇宙探査機などが小惑星に着陸する場合等で画像認識機能が用いられる。このような画像認識機能として広く用いられている機能の一つにエッジ検出機能がある。エッジ検出機能はラプラシアンフィルタなどで輝度の変化点を抽出することにより実現されるが、前処理としてガウシアンフィルタなどによりノイズ除去を図ることによって、よりロバストな処理が可能となる。このような実用的な画像認識機能をモチーフとして、ここでは粗粒度の演算としてガウシアンフィルタとラプラシアンフィルタを用いた。ガウシアンフィルタとラプラシアンフィルタをカスケード接続してエッジ検出を行うことにより上述のようにノイズの影響を受けにくいロバストな処理が可能になるが、これらの関数形は類似性のある畳み込み行列演算であり、違いはパラメータのみであるため、関数の使いまわし効果の測定には効果的である。ここでは Xilinx XC7A200T FPGA を使用し、画像処理の畳み込み演算を以下の 3 つの条件で実装した。

a) 畳み込み関数を 1 つの演算子として定義した。
b) a)に加えて、[17, 18]で報告されている Flexible Reliability Reconfigurable Array（FRRA）として称される動的再構成技術を使用して、基本的な 4 つの算術演算、すなわち加算、減算、乗算、および除算を実装した。
c) プリミティブな FPGA ライブラリを使用して設計内容を展開した。

最初の評価ケースでは、上記の 3 つの条件でラプラシアンフィルタを実装した。 次に、2 番目の評価ケースでガウスフィルタとラプラシアンフィルタを用いたカスケード演算を実装した。 演算子を定義するために、特定の C 言語のコメント記述/ * Cyber func = operator * /を使用した。 / * Cyber share_name = NAME * /として、他の特定の C 言語のコメント記述を使用し、粒度の比較的小さい操作間で比較的細かい操作を共有できる。NAME は任意の指定である。フィルタ処理の内容を図 6-17、および図 6-18 に示す。これらの畳み込みフィルタは、図 6-19 に示すような記述を使用して実装することができる。第 3 の条件の GPE 設計フレームワークのステップ 3 において、上記の a)と b)のケースでは CWB のデータベースにガウシアンフィルタ行列関数とラプラシアンフィルタ行列関数を登録した。これらの行列関数は、c)のケースではデータベースに登録されない。登録された関数は、前者の場合の高水準および動作合成プロセスで繰り返し使用された。

(a)

+1/16	+2/16	+1/16
+2/16	+4/16	+2/16
+1/16	+2/16	+1/16

(b)

-1	-1	-1
-1	8	-1
-1	-1	-1

(c)

図 6-17　ガウシアンフィルタとラプラシアンフィルタによるエッジ検出

(a) 元画像、(b) ガウシアンフィルタ係数行列と処理結果、(c) ラプラシアンフィルタ係数行列と処理結果

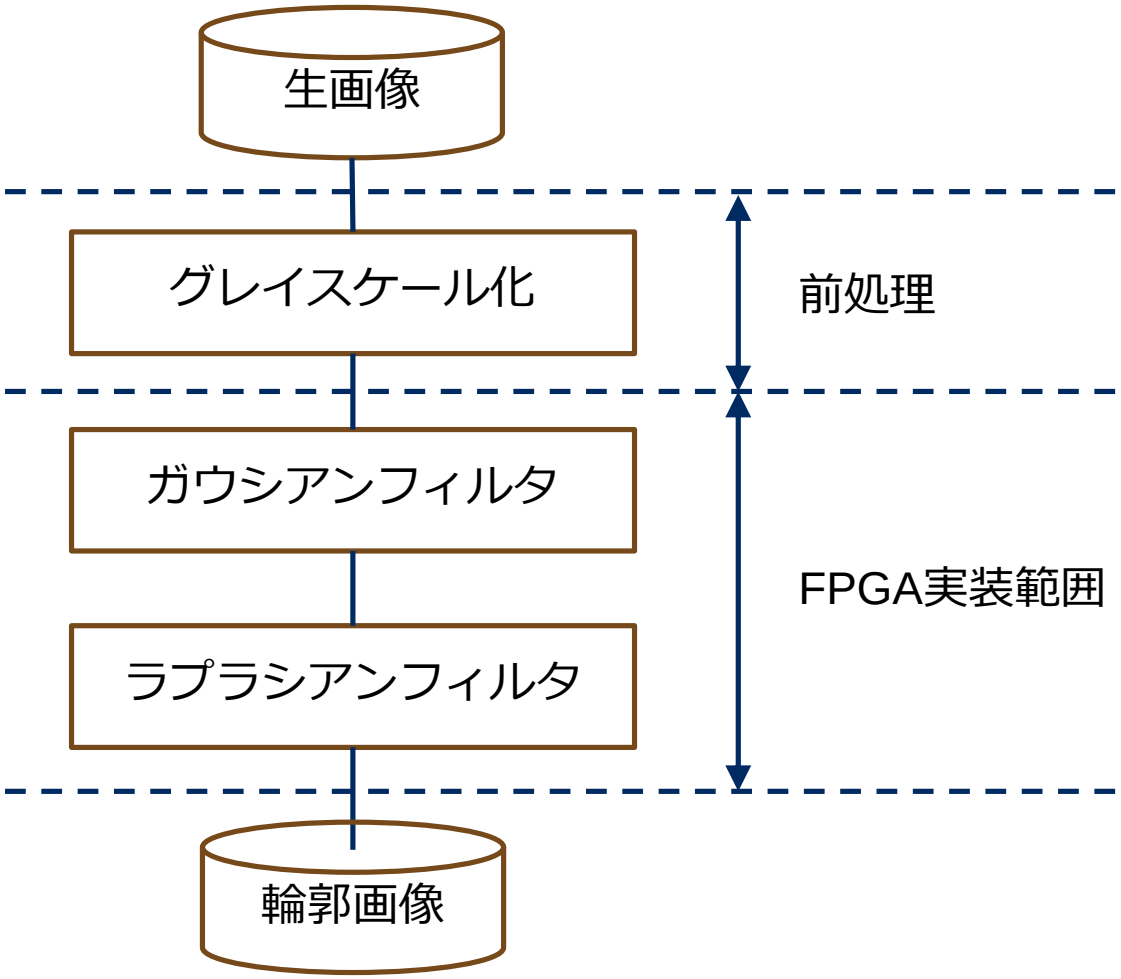


図 6-18　フィルタ処理内容

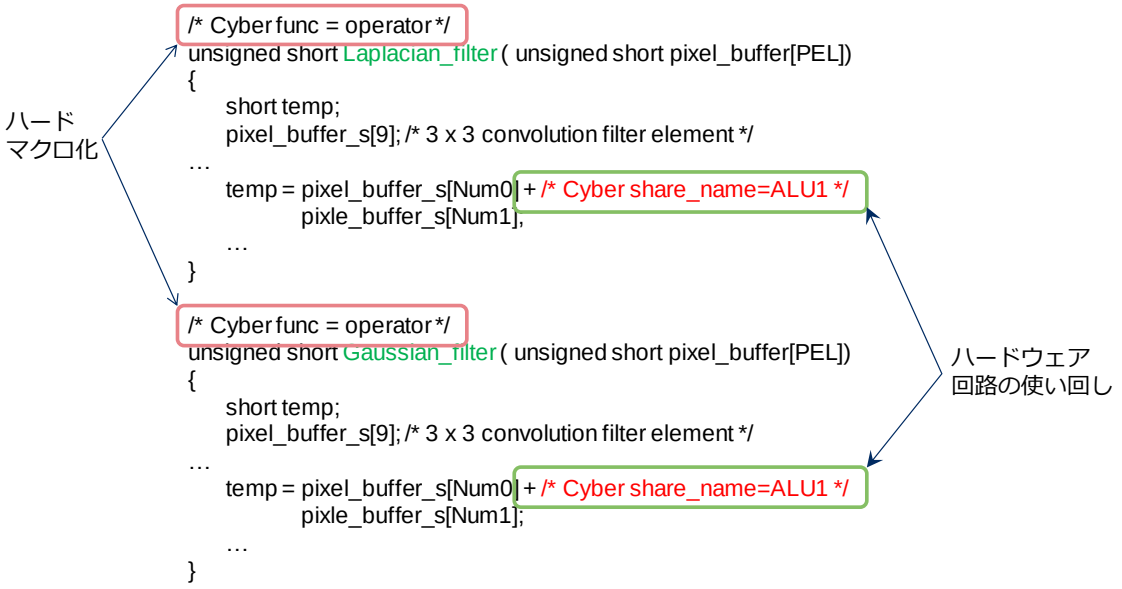


図 6-19　演算子の定義と共有

各評価ケースで使用された LUT の使用数を図 6-20 に示す。ここでは回路規模の評価のために演算器内の LUT 数は比較の対象から除外している。ガウシアンフィルタとラプラシアンフィルタの行列関数は似ており、その差は 3×3 行列のパラメータである。ここに示す LUT 数の削減は、GPE 設計フレームワークのステップ 3 で演算機能を登録する際に、前章で説明した CWB の新機能を使用して登録したオペレータを繰り返し使用することによって実現された。LUT 使用数の顕著な差異はカスケード接続の場合に現れた。FPGA ライブラリを使用した c)のケースではカスケード接続の LUT 使用数は、単一動作に比べて 2 倍以上になった。これに対して本研究で提案した GPE Design Framework により実現した場合には、高々1.5 倍程度である。この結果は、動作合成における演算の粒度を活用することが期待通りにうまくいったことを示している。GPE Design Framework は当初、ナノブリッジを活用した FPGA や ASIC などの SoC の実装効率を改善することを目的として提案したが、評価の結果、通常の FPGA の設計に対しても実装規模の削減効果を確認することができ、ナノブリッジを活用した FPGA のみならず既存の FPGA でも効果が見られたことを示している。これは新たに見出された成果であり、関数の共有（使いまわし）の効果であることが確認できた。この様子を図 6-21 に示す。更に、動的再構成手法を使用した FRRA を使用した 1 つの操作とカスケード接続の差異は、畳み込み操作での演算子定義を使用した場合の差より 22%小さかった。これは演算処理の粒度が粗ければ粗いほど、LUT の使用数が少なくなることを示していると考えられ、動的再構成手法を併用することにより更にハードウェアリソースの有効活用が可能なことを示しているとみられることから、本研究で確立した GPE design framework を発展される一つの方向性を示していると考える。

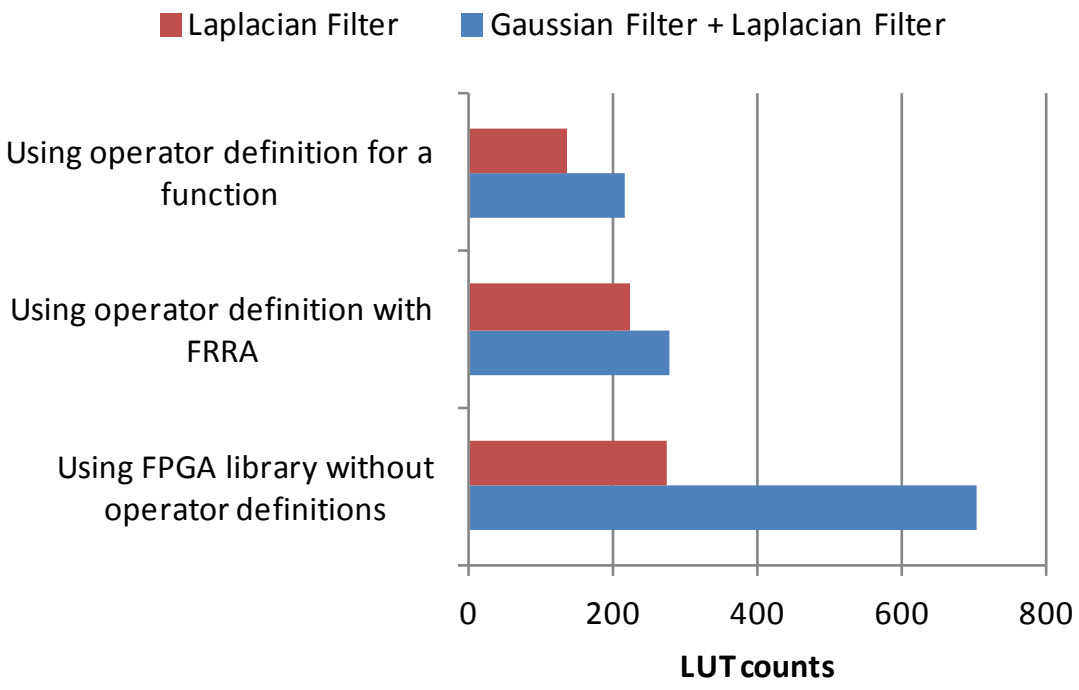


図 6-20　LUT 使用数の比較

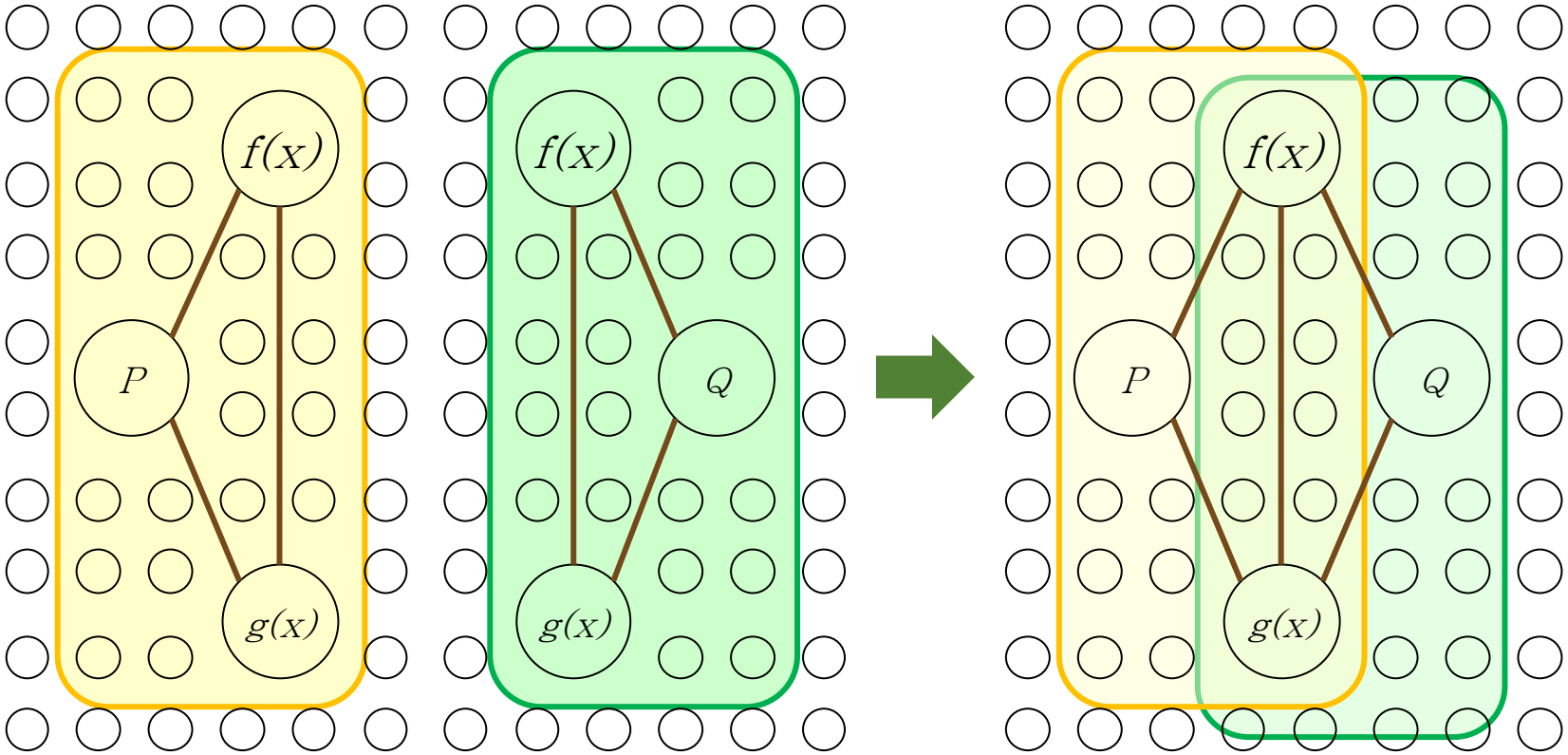


図 6-21　関数の使いまわし効果

第7章
結論と今後の展望

本節ではこれまでに述べた成果を纏めると共に、さらなる応用の可能性として得られた知見についても述べる。

7.1. GPE アーキテクチャの評価結果

本研究で提案する GPE アーキテクチャを実装し評価した結果、目的としたセンシングデバイスへ埋め込むための小型、低消費電力、および高信頼性を有するプロセッサ・エレメントが設計できることを確認した。本研究により拡張した高位合成技術を用いてアプリケーションに適したアルゴリズムがハードウェアとして実装され、プログラムメモリはもはや必要とされない。メモリコントローラのようなプログラムメモリに関連する周辺回路は、もはやプロセッサチップ上には不要であり、プロセッサの大部分は演算リソースに使用することができる。これにより IoT アプリケーションにおけるメモリのソフト・エラーは大幅に低減でき、従来の DRP で実現されている消費電力の大幅な削減効果も継承している。従来のデータフローコンピュータの特徴である、演算器に対するデータが揃った時点で直ちに演算が行われる利点も本アーキテクチャは有している。ただし従来のデータフローコンピュータは、汎用性を持たせるべく演算器の構成情報をメモリに格納する[95, 96, 97, 98, 99, 100]。これに対し本アーキテクチャでは演算器の構成情報も直接ハードウェア化しつつ、高位合成技術との整合性を取り高級言語の利用を可とする。一方、アプリケーションプログラムの書き換えには制約があり、例えば動的プログラムロードは現時点では容易ではない。しかしながら、組込みシステムアプリケーションのユースケースに基づく限り、組込みシステム製品は出荷前に数回アプリケーションプログラムを変更できれば十分である。

提案するプロセッサアーキテクチャの特徴は、ISP 自体の定義が標準的な設計プロセスに含まれていることである。この機能は、本研究にて高位合成技術を拡張したことによりもたらされた利点であり、アプリケーションに対して最適化されたハードマクロを専用 ISP として実装できる。デファクトスタンダードとなっている特定の ISP を有するマイクロプロセッサ・アーキテクチャとの互換性を考慮する必要は無い。PE の互換性と相互運用性は、ISP のレベルではなく、PE 間の入出力通信プロトコルレベルで確保する。このために SpaceWire のオープンスタンダードを活用することができる。各 PE の入力と出力が互換性を維持している限り、各 PE の設計は独立しているため、高級プログラミング言語で記述された設計はテクノロジに依存せず、結果として、最新の製造技術を PE 実装に使用することができる。

提案するアーキテクチャに基づく PE は、次に述べる 2 種類の単純化が実現されることからリアルタイム性が向上する。まず、システムによって基本データとして時刻が処理される。したがって、時間の割込みとしてソフトウェア処理を使用する時間の仮想化はもはや必要ではない。次に、システム設計に伴う並列性が物理的に並行したプロセスとして実現されるため、マルチプロセス・オペレーティングシステムまたはマルチタスク・オペレーティングシステムによるプロセスまたはタスクのシリアル化を使用した仮想化はもはや必要ない。並行するタスクやプロセスの同期はバリア同期で実現できると考えられ、これは PE の細粒度または粗粒度の実装で対応できる。この結果、リアルタイムシステムは複数のタスク、またはプロセスの単純化された同期により実現できることが見込まれる。

7.2. システム LSI の技術トレンド

本研究で提案するプロセッサ・エレメントのアーキテクチャを技術ロードマップの観点から考察する。半導体デバイスの技術トレンドは図 7-1 に示す Extended Makimoto's Wave により予見できるとされている[47]。この Extended Makimoto's Wave によれば、現在は System-on-a-Chip（SoC）および System in package（SiP）から次の時代への移行期に差し掛かっており、SoC/SiP の次にはシステムレベルのプログラマブルデバイスの標準化が見込まれている。これは" Highly flexible super integration（HFSI）"と呼ばれており、システムレベルの機能がプログラマブルになるデバイスとして予見されている。

本研究では粗粒度の関数を基本演算単位として活用できるよう、高位合成技術を拡張し、さらにその機能を実際に高位合成ツールである CyberWorkBench を活用して有効性を確認した。この粗粒度の関数はシステムレベルの機能にも容易に拡張可能であり、本研究の世紀により、次世代に HFSI に向けた基本技術開発が為されたものと考える。

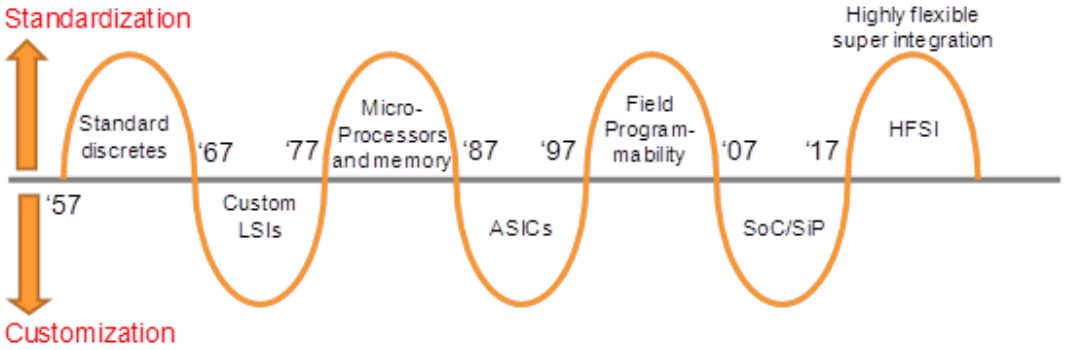


図 7-1　Extended Makimoto's Wave [47]

システムレベルの動的な変更はナノブリッジに期待できる利点である。機能を動的に変えることができるコントローラの開発や、ファームウェアの更新のしくみの実装などは今後の研究課題である。これらは金属架橋（銅線）の切り替えや機能の書換えの可否などの

ソフトウェア的な処理の開発支援機能をこれから充実させる必要がある。また、回路としての展開の有無や、FPGA のみならず PROM としての応用を追求することにより、画像処理に要するテンプレートマッチングのためのメモリ機能の実装なども今後の研究テーマとして視野に入れることができる。ファームウェアの更新のしくみについて、現在 JAXA プロジェクトである革新実証小型衛星による FPGA の再プログラミング機能の軌道上実験が予定されており、このような機会を活かして着実に開発を進めたい。さらに、6.2 節に報告したように、センサ出力信号を Wire rate で入力信号を逐次処理できることを確認したことから、多数のエレメントを処理する Synthetic Aperture Radar (SAR)の信号処理や、ルータなどのネットワーク機器に於いてトラフィックの学習結果をルーティングに活かすような応用も考えられ、今後の研究テーマとしたい。

7.3. ニューロモルフィックへの応用

ナノブリッジは LSI デバイス内に動的に結線を形成することからニューロモルフィックへの応用も有望な用途であり、各種の研究が進められている。ニューロモルフィックへの応用方法については二つのタイプが提唱されている[69]。一つは脳のはたらきを模擬する方向であり、もう一つは必ずしも脳の動きにとらわれずに機械学習の効果的な実装を目指すものである。ナノブリッジ FPGA のデジタル値に基づくネットワークを活用する場合には後者の応用が主となるが、アナログ的な結線、すなわち抵抗値の変化を活用できれば前者の応用も可能となる。これについては、ナノブリッジのアナログ応用の根拠として、印加時間・電流に対して伝導度が変化する実験結果が報告されている。この様子を図 7-2 に示す[70]。

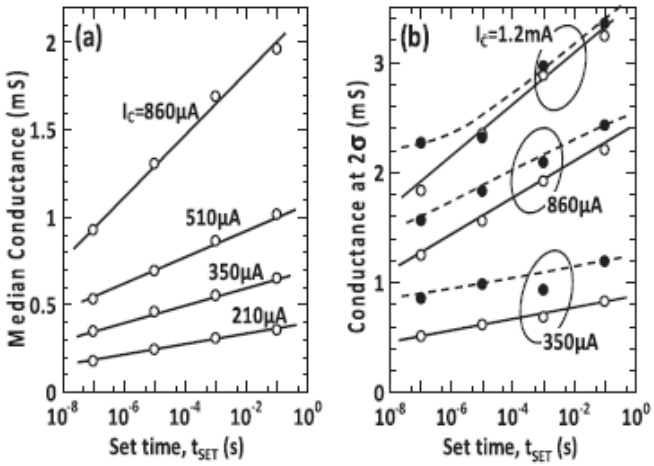


図 7-2　ナノブリッジのアナログ特性

(a) Median conductances for different set times (tSET) and control currents (IC). (b) Conductance at þ2value in Figs. 1(b) and 1(c). þ2 values for IC's of 1.2mA and 860μA are also shown [70].

本現象を活用し、アナログ的な抵抗値の変化をニューロモルフィックに活用できれば、前者の脳のはたらきを模擬する応用が可能になると見込まれる。具体的には以下の二点が今後の研究指針となる。一つには現在デジタル的に行っている状態遷移をアナログ化することが考えられ、もう一つには各 PE が状態に応じて冗長構成を変更する機能の応用である。前者について、現時点では各 PE に内蔵しているステートマシンは外界からの stimulus に対してデジタル的に遷移する。ナノブリッジのアナログ特性を活用し、外界からの stimulus に対応して状態遷移が行われるようにすれば、状態そのものは離散的ではあるものの、アナログ的な状態変化が模擬できることが見込まれる。この様子を図 7-3 に示す。

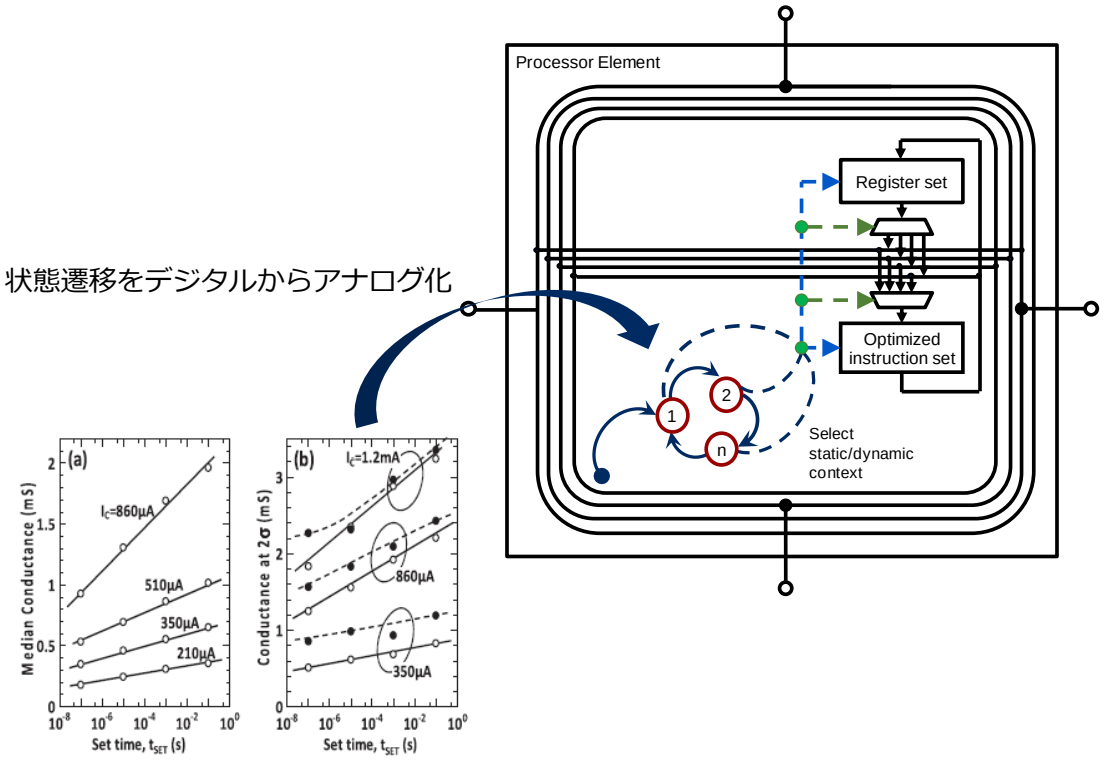


図 7-3　各 PE の状態遷移のアナログ化

後者については、第 5 章で述べた冗長構成は、Appendix C で述べたように全系同一演算、全系独立演算、および部分同一／部分独立演算の各モードで系として動作することが可能である。この場合、全系同一演算モードでは同一 stimulus に対して全 PE が同じ出力を生成するが、独立演算 PE がある場合には、全 PE が同じ出力を生成することは保証されない。この様子を図 7-4 に示す。この動きと前者のアナログ的な状態遷移を併用することにより、本研究にて提案したアーキテクチャにて生物知識処理にみられる少ないリソースを用いて多機能化するしくみを模擬できる可能性が見込まれる。

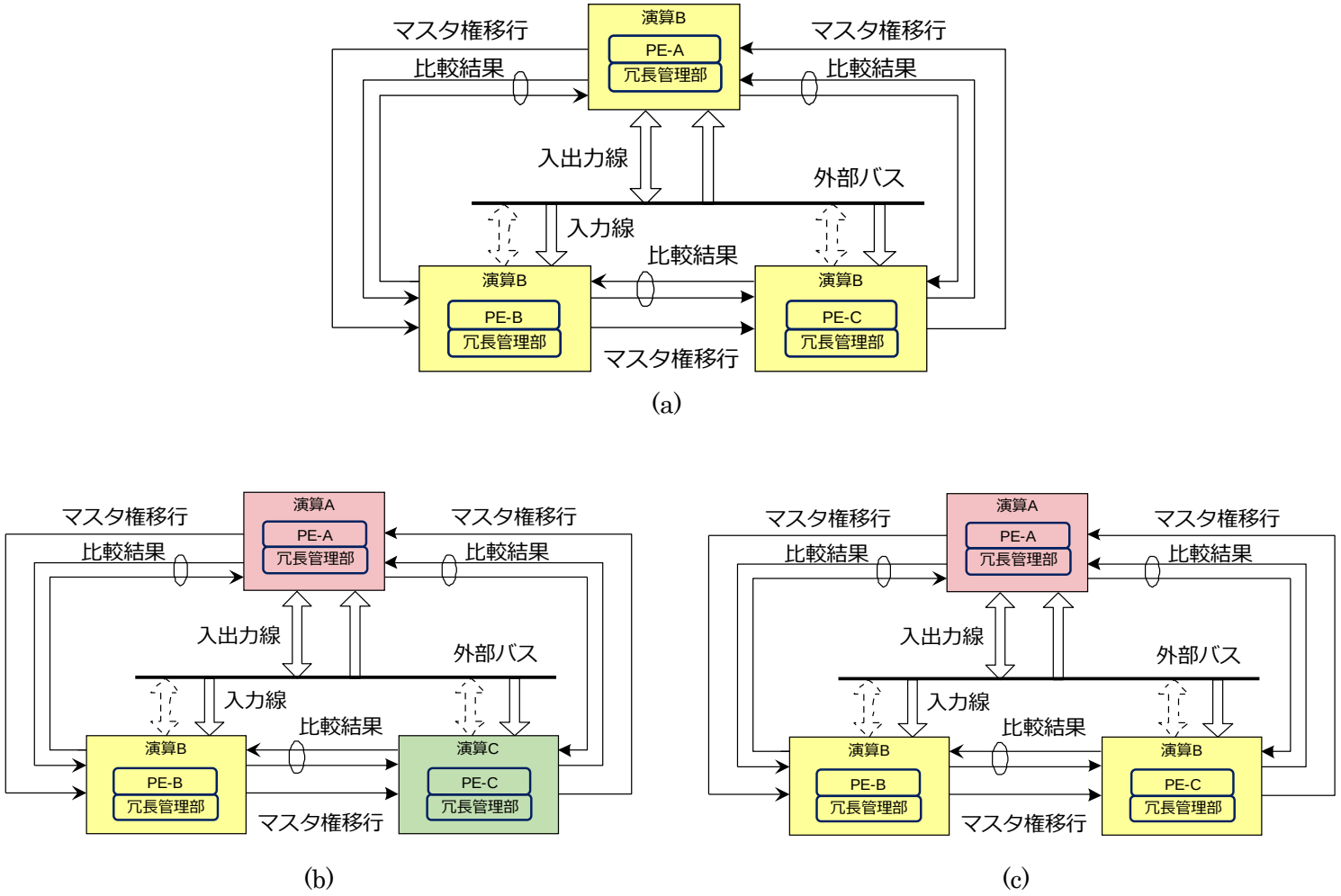


図 7-4 全体冗長/部分冗長（部分独立構成）

(a) 全系同一演算、(b) 全系独立演算、(c) 部分同一／部分独立演算

4.2 節において、ある関数に対応して生成された回路が他の関数に活用される機能について述べた。これは一度構成された基本素子が再利用され、複数の機能実現に用いられることを示した。図 7-5 に 6.3 節に述べた評価実験の演算器の共有例を示す。これは少数ニューロンによるネットワークで機能を増してゆく生物知識処理を模擬する可能性があることを示唆している。また、少ないリソースでいかに多機能を実現するかという今後の研究の重要性につながるものと考える。さらに、5.3 節では本研究成果を反映した高信頼性設計手法は原理的に PE の設計変更をせずに N 重冗長構成が可能であると共に、各 PE は機能を変更できることを述べた。これにより、一つの PE が複数の機能を兼ねつつ、N 重冗長を構成して正しい演算処理を兼ねる機構は、生物の神経細胞のような統計的判断により信頼性を確保する動きを模擬できるものと考える。また、この機能により、広い範囲の情報からの判断と、特定の情報への反応を兼ねる回路が生成できることからこの点においても生物知的処理を模擬する研究に繋げられるものと見込まれる。この様子を図 7-6 に示す。

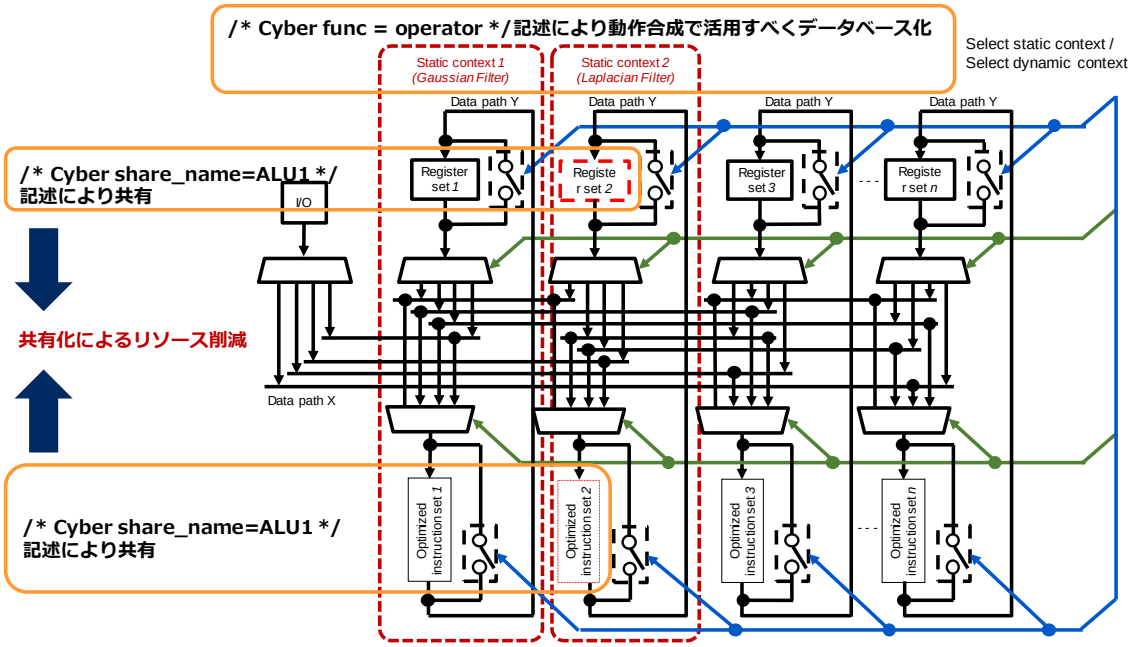


図 7-5　演算器の共有例と指定方法

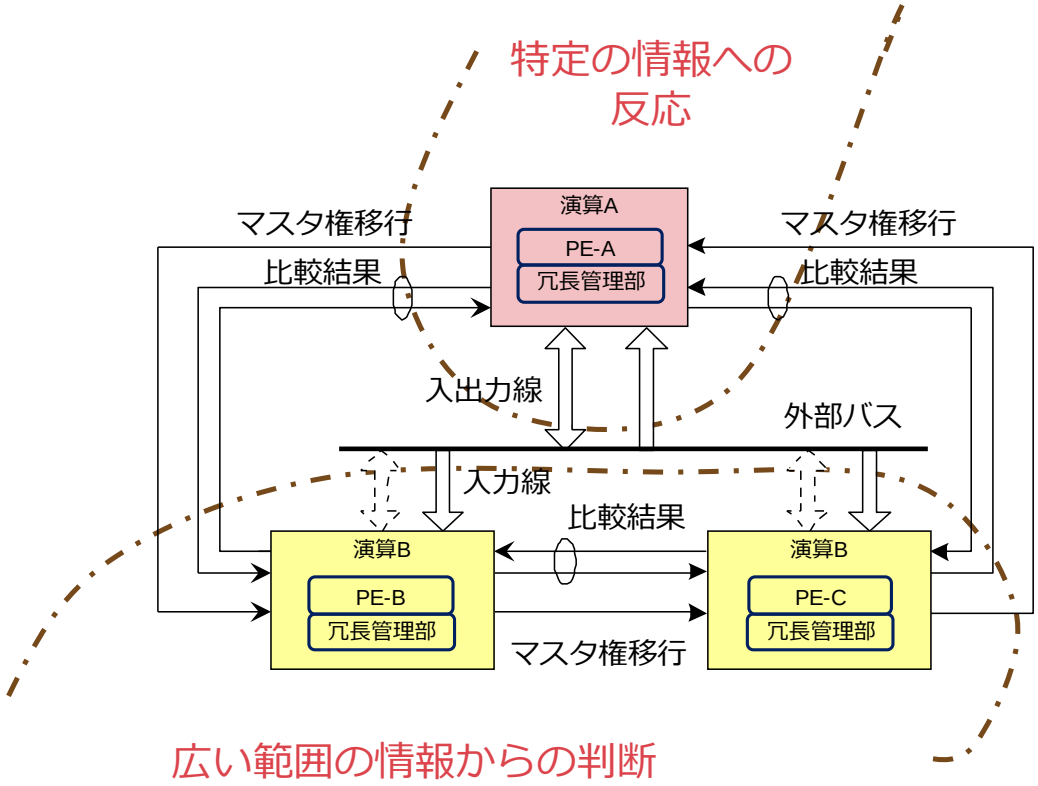


図 7-6　部分同一／部分独立演算による広い情報からの判断と、特定の情報への判断の模擬例

7.4. 関数の粒度の選択

本研究では、センシングデバイスに演算処理機能を埋め込むことにより、データの選択や圧縮を行ってデータ伝送量を削減することを提案した。これについて、センサ自体がデータを取捨選択すると不要とされたデータに意味があることが実際の運用を通じて気付くこともあり、あるいはデータを圧縮することにより大事な情報が失われることもある、といったことも考察する必要がある。本研究では粗粒度の関数は最適化した回路としてハードマクロ化することを想定しているが、データの選択や圧縮を行う関数はナノブリッジを活用してプログラマブルな回路として実装し、運用中にデータの選択や圧縮のアルゴリズムの変更に備える必要があると考えられる。実装例としてはに示すようにプロセッサエレメントに内蔵している FSM の遷移条件を運用に応じて最適化し、データ圧縮パラメータなどを運用に応じて調整することが考えられる。この様子を図 7-7 に示す。

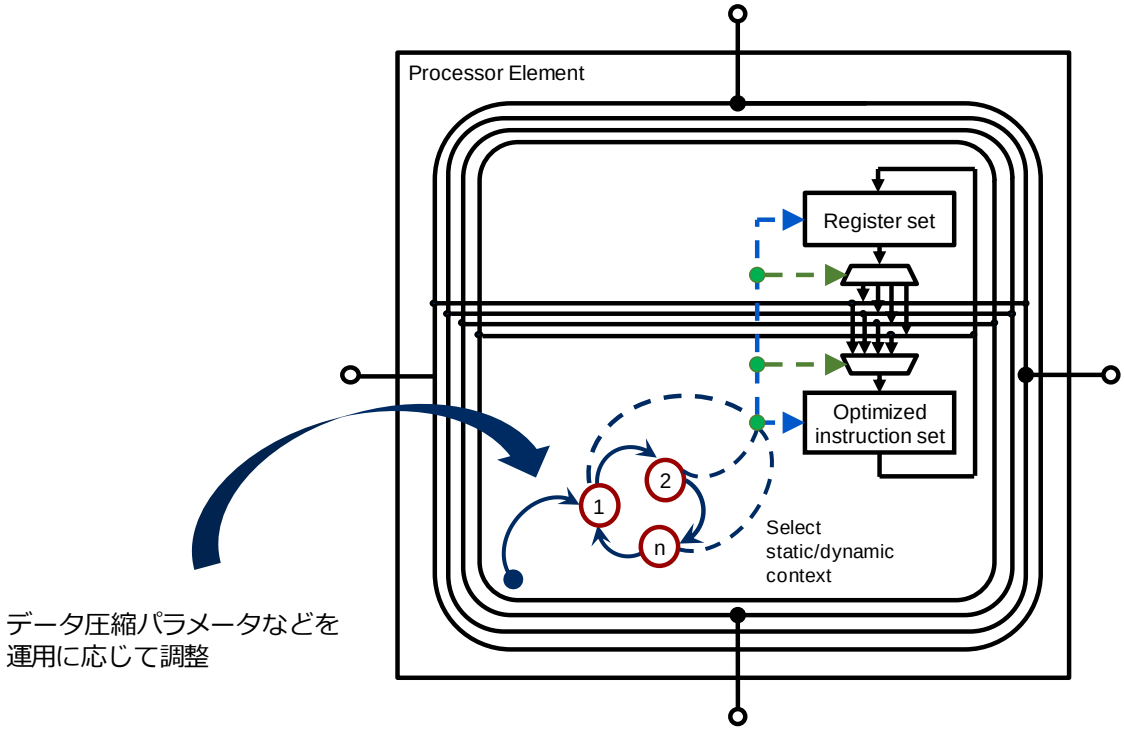


図 7-7　運用に応じた遷移確率の調整

謝辞

三年間に亘ってご指導頂きました、東京大学先端科学技術研究センター・知能工学研究室の岩崎晃教授に心から感謝いたします。本研究は人工衛星搭載用として研究開発を進めてきた小型・軽量・高信頼性アーキテクチャと、各研究所および大学と共同研究を進めてきたナノブリッジ応用技術、信頼性可変粗粒度再構成可能アーキテクチャ、および高位合成／動作合成技術の拡張を集約すべく開始され、岩崎教授のご指導により、最先端のセンサアプリケーションとして具現化すべく体系化することができました。センサアプリケーションに必要な最先端の画像処理や機械学習技術の習得、研究論文を論文誌に投稿して外部評価を得るに際しての論文執筆に関する詳細なご指導、および本研究にて提案したアーキテクチャを検証するための評価実験の計画を立案するにあたっての着眼ポイント等、企業で製品開発に従事する過程では得られない、貴重なご指導を頂くことができました。貴重な議論をいただきました、神崎亮平教授、森川博之教授、越塚登教授、矢入健久准教授には心から感謝いたします。本研究を始めるきっかけとなりました産学連携共同研究に携わる各大学・研究所の先生方をご紹介いただくと共に博士課程への進学を勧めて頂いた国立研究開発法人宇宙航空研究開発機構の高橋忠幸教授、粒度別階層化アーキテクチャの着想が新しいプロセッサアーキテクチャに繋がると動機づけ頂いた JST CREST DVLSI 領域リーダーの浅井彰二郎博士、Makimoto's Wave [47]の予見に基づきナノブリッジと高位合成技術の組み合わせが次の 10 年の技術トレンドに則しているとの貴重なアドバイスを頂いた牧本次生博士には心から感謝をいたします。大阪学院大学の菊野亨教授、大阪大学の尾上孝雄教授には進学先について親身になって相談にのって頂き、岩崎晃教授にご指導頂く機会を得ることができました。心から感謝いたします。業務と並行して社会人博士課程に進学することを快諾頂いた NEC スペーステクノロジー(株)の米田誠良技術主幹、安藤英昭本部長代理に感謝をいたします。投稿論文、および信頼性可変粗粒度再構成可能アーキテクチャについて詳細な議論をする機会とご指導を頂いた大阪大学の橋本昌宜教授、高知工科大学の密山幸男准教授、立命館大学の越智裕之教授、京都高度技術研究所の神原弘之博士、高信頼性・耐放射線性 LSI についてご指導を頂いた京都大学の小野寺秀俊教授に心から感謝いたします。CyberWorkBench の機能拡張を快諾頂いた日本電気(株)システムプラットフォーム研究所の若林一敏博士、ナノブリッジ FPGA の評価キットを用意して頂いた同研究所の杉林直彦研究部長、坂本利司博士、宮村信博士、多田宗弘博士に感謝いたします。大阪大学工学部・大学院修士前期課程を通じて御指導頂き、固体物理を通じてさまざまな分野の研究開発を進める素養を養ってくださいました濱口智尋大阪大学名誉教授には心から感謝を致します。最後に、夜間や週末には不在となり、通常業務と合わせるとあまり家におらずに研究に打ち込むことを辛抱強く支援してくれた家族に心から感謝します。

本研究の消費電力とレイテンシの評価の一部は、新エネルギー・産業技術開発機構（NEDO）のエネルギー・環境新技術主導プログラムの助成を受けて行われました。

研究業績

【主執筆の査読付き論文】

[1] H. Hihara, A. Iwasaki, M. Hashimoto, H. Ochi, Y. Mitsuyama, H. Onodera, H. Kanbara, K. Wakabayashi, T. Sugibayashi, T. Takenaka, H. Hada, M. Tada, M. Miyamura, T. Sakamoto, “Extended high-level synthesis with the layered architecture of processing elements,” IEEE letters of Embedded systems, in review, 2017.

[2] H. Hihara, A. Iwasaki, M. Hashimoto, H. Ochi, Y. Mitsuyama, H. Onodera, H. Kanbara, K. Wakabayashi, T. Sugibayashi, T. Takenaka, H. Hada, M. Tada, “Applications of Reconfigurable Processors as Embedded Automatons in the IoT Sensor Networks in Space,” in VLSI Design and Test for Systems Dependability, 1st ed., S. Asai, Ed., Springer, 2017, in print.

[3] H. Hihara, K. Moritani, M. Inoue, Y. Hoshi, A. Iwasaki, J. Takada, H. Inada, M. Suzuki, T. Seki, S. Ichikawa, J. Tanii, “Onboard Image Processing System for Hyperspectral Sensor,” Sensors 2015, 15, 24926-24944; doi:10.3390/s151024926C.

【主執筆の国際学会】

[1] H. Hihara, “Atom-Switch FPGA Application for IoT Sensing System in Space,” International Symposium on Atomic Switch: - Invention, Practical Use and Future Prospect, March, 2017.

[2] H. Hihara, M. Nomachi, S. Fukuda, T. Ishida, T. Takahashi, “Keynote Presentation: Scalable interconnection with SpaceWire,” 2016 International SpaceWire Conference (SpaceWire), 2016.

[3] H. Hihara, N. Tamagawa, T. Imamura, H. Sugaya, T. Sugibayashi, M. Miyamura, T. Sakamoto, M. Tada, H. Hada, K. Wakabayashi, A. Iwasaki, “Programmable SpaceWire interface with atom switch,” 2016 International SpaceWire Conference (SpaceWire), 2016, pp. 1-4, DOI: 10.1109/SpaceWire.2016.7771606

[4] H. Hihara, A. Iwasaki, N. Tamagawa, M. Kuribayashi, M. Hashimoto, Y. Mitsuyama, H. Ochi, H. Onodera, H. Kanbara, K. Wakabayashi, M. Tada, “Novel processor architecture for onboard infrared sensors,” in Proceedings Volume 9973: Infrared Remote Sensing and Instrumentation XXIV, December 2016.

[5] H. Hihara, Y. Takano, J. Sano, K. Iwase, S. Kawakami, H. Otake, T. Okada, R. Funase, J. Takada, T. Masuda, “Infrared sensor system using robotics technology for inter-planetary mission,” in Proceedings Volume 9608: Infrared Remote Sensing and

Instrumentation XXIII, October 2015.

【主執筆の研究会・全国大会】

[1] 檜原 弘樹, 玉川 信生, 今村 貴之, 菅谷 寿之, 福田 盛介, 石田 貴行, "NB-FPGA の応用," 第 60 回宇宙科学技術連合講演会, 2016.

[2] 檜原 弘樹, 岩崎 晃, 橋本昌宜, 越智 裕之, 密山 幸男, 小野寺 秀俊, 神原 弘之, 若林一敏, 杉林 直彦, 竹中 崇, 波田 博光, 多田 宗弘, “センサの知能化に適したプロセッサアーキテクチャの考察,” 電子情報通信学会 DC 研究会, 2015.

[3] 檜原 弘樹, 北出 賢二, “開かれた活動拠点, 知的創造の場として宇宙,” 第 58 回宇宙科学技術連合講演会, 2014.

【登録特許】

[1] DYNAMIC CIRCUIT DEVICE, Application No.: 15863634.0 – 1810, Patent No.: PCT/JP2015/005793, Jun 30, 2017.

参考文献

[1] G.E. Moore, “Cramming More Components onto Integrated Circuits,” Electronics, vol. 38, no. 8, 1965, pp. 114-117.

[2] Sakamoto, T., Kaeriyama, S., Mizuno, M., Terabe, K., Hasegawa, T., and Aono, M., "Nanobridge Technology for Reconfigurable LSI," NEC Tech. J. 2(1), 72-75 (2007).

[3] M. Miyamura, T. Sakamoto, M. Tada, N. Banno, K. Okamoto, N. Iguchi, and H. Hada, “Low-power Programmable-logic Cell Arrays using Nonvolatile Complementary Atom Switch,” 15th Int'l Symposium on Quality Electronic Design, pp.330-334, 2014

[4] M. Tada, T. Sakamoto, M. Miyamura, N. Banno, K. Okamoto, N. Iguchi, and H. Hada, “ Improved OFF-State Reliability of Nonvolatile Resistive Switch With Low Programming Voltage,” IEEE Trans. on Electron Devices, Vol. 59, No. 9, September 2012.

[5] http://jpn.nec.com/solution/m2m/whats_connexive.html

[6] 日本航空宇宙学会：大山聖，桜井誠人，東京大学：岩崎晃，情報通信研究機構：高橋靖宏，豊嶋守生，日本学術会議第 201 回幹事会配布資料, http://www.scj.go.jp/ja/member/iinkai/kanji/pdf22/siryo201-5-10-7.pdf, 2014 年 9 月 19 日

[7] http://jpn.nec.com/solution/space/solution/bousai.html

[8] 檜原弘樹, "設計品質確保の思想," Design Wave Magazine, CQ 出版株式会社, February 2006, pp.36-45.

[9] K. Wakabayashi and B. C. Schafer, "All-in-C" SoC Synthesis and Verification with CyberWorkBench., Springer, P. Coussy and A. Morawiec (Eds.), “High-Level Synthesis from Algorithm to Digital Circuit”, XVI, ISBN: 978-1-4020-8587-1, Chapter 7, pp.113-127, 2008.

[10] K.Wakabayashi and T. Okamoto, “C-based SoC Design Flow and EDA Tools”, IEEE Tran. On CAD, 2000 年 12 月, pp.1507-1522.

[11] K.Wakabayashi, “CyberWokBench: Integrated design environment based on C-based behavior synthesis and verification”, IEEE VLSI-TSA, pp.173-176, 2005.

[12] http://www.nec.com/en/global/about/vision/closeup/01.html

[13] J. Rose, R. J. Francis, D. Lewis and P. Chow, “Architecture of Programmable Gate Arrays: The Effect of Logic Block Functionality on Area Efficiency,” IEEE Journal of Solid State Circuits, Oct. 1990, pp. 1217 - 1225.

[14] J. Kouloheris and A. El Gamal, “FPGA Area vs. Cell Granularity -- Lookup Tables and PLA Cells,” ACM Workshop on Field-Programmable Gate Arrays, 1992, pp. 9 - 14.

[15] D. Hill and N-S Woo, “The Benefits of Flexibility in Look-up Table FPGAs,” in

FPGAs, W. Moore and W. Luk, Eds., Abingdon: 1991, pp.127 - 136.
[16] D. Tavana, W. Yee, S. Young, and B. Fawcett, “Logic Block and Routing Considerations for a New SRAM-Based FPGA Architecture,” CICC, 1995, pp. 24.6.1 - 24.6.4.
[17] D. Alnajjar, H. Konoura, Y. Mitsuyama, H. Shimada, K. Kobayashi, H. Kanbara, H. Ochi, T. Imagawa, S. Noda, K. Wakabayashi, M. Hashimoto, T. Onoye, and H. Onodera, "Reliability-Configurable Mixed-Grained Reconfigurable Array Supporting C-To-Array Mapping and Its Radiation Testing," in Proc. IEEE Asian Solid-State Circuits Conference (A-SSCC 2013), pp. 313-316, Nov. 2013.
[18] H. Konoura, D. Alnajjar, Y. Mitsuyama, H. Shimada, K. Kobayashi, H. Kanbara, H. Ochi, T. Imagawa, K. Wakabayashi, M. Hashimoto, T. Onoye, and H. Onodera, "Reliability-Configurable Mixed-Grained Reconfigurable Array Supporting C-based Design and Its Irradiation Testing," IEICE Trans. on Fundamentals of Electronics, Communications and Computer sciences, Dec. 2014.
[19] T. Toi et al., “Optimizing Time and Space Multiplexed Computation in a Dynamically Reconfigurable Processor,” in Prof. of FPT2013, 2013, pp. 106–111.
[20] J. Lee et al., “Real-time Ray Tracing on Coarse-grained Reconfigurable Processor,” in Prof. of FPT2013, 2013, pp. 192–197.
[21] H.Amano et.al, “MuCCRA Chips: Configurable Dynamically-Reconfigurable Processors,” Proc. of ASSCC, pp. 384–387, 2007.
[22] S. Goldstein, H. Schmit, M. Budiu, S. Cadambi, M. Moe, and R. Taylor, “Piperench: a reconfigurable architecture and compiler,” Proc. 26th Annual International symposium on Computer Architecture, vol. 33, no. 4, pp. 70–77, 2000.
[23] 粟島亨, “Full Software Implementation of Real-time ISDB-T Modulator on Dynamically Reconfigurable SoC Using Practical Co-design Environment,” 招待講演 CoolChips XIV, 横浜情報文化センター, 2011 年 4 月 20 日
[24] T. Awashima: “Full Software Implementation of Real-time ISDB-T Modulator on Dynamically Reconfigurable SoC Using Practical Co-design Environment,” Invited talk, CoolChips XIV, Yokohama Industrial Development Corporation, 20th, 2011.
[25] 檜原弘樹, 大塚誠, 水島泰彦, 森里司, 大島武, 三好弘晃, 馬場研一, “宇宙搭載用フォールトトレラントコンピュータ,” 情報処理, Vol.35, No.6, p.p.497-503, 1994.
[26] H. Hihara, K. Yamada, M. Adachi, K. Mitani, M. Akiyama, and K. Hama, "Autonomous Fault Tolerant Computer for SERVIS-2 Satellite.", Technical Report of IEICE, DC2002-75, Vol. 102, No. 492, pp. 19-24, December, 2002.
[27] K. Yamada, H. Hihara, M. Adachi, K. Mitani, K. Hama, “Autonomous Fault Tolerant Computer for SERVIS-2,” Proceedings of the 46th Space Sciences and

Technology Conference, pp. 261-266, October, 2002.
[28] H. Hihara, K. Yamada, M. Adachi, K. Mitani, M. Akiyama and K. Hama, CRAFT: AN EXPERIMENTAL FAULT TOLERANT COMPUTER SYSTEM FOR SERVIS-2 SATELLITE, 21st International Communications Satellite Systems Conference and Exhibit, AIAA 2003-2291, April. 2003.
[29] R. Hartenstein, "The Microprocessor Is No Longer General Purpose: Why Future Reconfigurable Platforms Will win," in Proc. Second Annual IEEE Intl. Conf. on Innovative Systems in Silicon, 1997, pp. 2-12.
[30] M. Meribout and M. Motomura, "A Combined Approach to High-Level Synthesis for Dynamically Reconfigurable Systems," IEEE Trans. on Computers, vol. 53, no. 12, 2004, pp. 1508-1522.
[31] M. Meribout and M. Motomura, "Efficient Metrics and High-Level Synthesis for Dynamically Reconfigurable Logic," IEEE Trans. on Very Large Scale Integration (VLSI) Systems, vol. 12, no. 6, pp. 603-621, 2004.
[32] T. Toi, N. Nakamura, Y. Kato, T. Awashima and K. Wakabayashi, "High-Level Synthesis Challenges and Solutions for a Dynamically Reconfigurable Processor," in 2006 IEEE/ACM Intl. Conf. Computer Aided Design, 2006, pp. 702-708.
[33] Y. Hasegawa, S.Tsutsumi, V.Tanbunheng, T.Nakamura, T.Nisimura, and H.Amano, "Design Methodology and Trade-offs Analysis for Parameterized Dynamically Reconfigurable Processor Arrays," in Proc. of FPL 2007, 2007, pp. 796-799.
[34] http://www.eorc.jaxa.jp/hatoyama/satellite/satdata/adeos_config_j.html
[35] 川田恭裕, 石毛康夫, "人工衛星姿勢制御用コンピュータシステム," 信学誌, Vol.73, No.11, pp.1215-1221, 1990.
[36] Sklaroff, J. R., "Redundancy Management Technique for Space Shuttle Computers," IBM J. R&D, 20, 1, p.20, 1976.
[37] 金川信康, 井原廣一, "宇宙機搭載用コンピューター汎用 LSI によるシステム構成ー," 電子情報通信学会誌, Vol.73, No.11, pp.1209-1214, 1990.
[38] 古城隆, 矢野陽一, "汎用マイクロプロセッサチップ," 信学誌, Vol.73, No.11, pp.1222-1227, 1990.
[39] K. Yamasaki, K. Hosaka, N. Morita, S. Ishii, T. Kikuchi, T. Nakamura, "Development Model of Main Processing Controller for Japanese Experiment Module," NEC R&D, Vol.34, No.3, pp.385-395, 1993.
[40] 檜原弘樹, 岩崎晃, 橋本昌宜, 越智裕之, 密山幸男, 小野寺秀俊, 神原弘之, 若林一敏, 杉林直彦, 竹中崇, 波田博光, 多田宗弘, "センサの知能化に適したプロセッサアーキテクチャの考察," 電子情報通信学会 DC 研究会, 2015
[41] ECSS-E-ST-50-12C, "SpaceWire . Links, nodes, routers and networks," ECSS

Secretariat, ESA- ESTEC, Requirements & Standards Division, 2008
[42] H. Hihara, K. Iwase, J. Sano, H. Otake, T. Okada, R. Funase, R. Kashikawa, I. Higashino, T. Masuda, "SpaceWire-based thermal-infrared imager system for asteroid sample return mission HAYABUSA2," Journal of Applied Remote Sensing, Vol.8, pp.084987-1-13, 2014.
[43] Satoshi Morinaga, Masato Edahiro, Tomoyuki Fujita, Representation and Analysis of Fault-Tolerant Systems using Domain-Partition Model. Technical Report of IEICE, pp. 31-38, April, 1995.
[44] Satoshi Morinaga, A General Model for Reliability Maximization Problem under Given Redundancy. Twenty-Seventh Annual International Symposium on Fault-Tolerant Computing (FTCS-27), pp. 363-372, June, 1997.
[45] Satoshi Morinaga, "Domain-Partition Model of Fault-Tolerant Systems Formulation and Analysis of Reliability Maximization Problem under Given Redundancy," The Trans. of the IEICE (D-I), Vol. J82-D-I, No. 10, pp. 1276-1285, October, 1999.
[46] Nobuhiko Ido, Tatsuhiro Tsuchiya, and Tohru Kikuno, A Study of the Partitioning in the Domain Partition Model., The 39th FTC Workshop, pp. 1-10, July 16-18, 1998.
[47] T. Makimoto: "Implications of Makimoto's Wave," IEEE Computer, 46, 12, (2013) 32, http://doi.ieeecomputersociety.org/10.1109/MC.2013.294
[48] H. Hihara, Y. Takano, J. Sano, K. Iwase, S. Kawakami, H. Otake, T. Okada, R. Funase, J. Takada, T. Masuda, "Infrared sensor system using robotics technology for inter-planetary mission," in Proc. of SPIE, Infrared Remote Sensing and Instrumentation XXIII, vol. 9608, Sept. 2015, pp. 96080C-1-11.
[49] H. Hihara, A. Iwasaki, N. Tamagawa, M. Kuribayashi, M Hashimoto, Y. Mitsuyama, H. Ochi, H. Onodera, H. Kanbara, K. Wakabayashi, M. Tada, "Novel processor architecture for onboard infrared sensors," in Proc. of SPIE, Infrared Remote Sensing and Instrumentation XXIV, vol. 9973, Aug. 2016, pp. 99730S-1-8.
[50] Fukuhara, T., Taguchi, M., Imamura T., Nakamura, M., Ueno, M., Suzuki M., Iwagami, N., Sato, M., Mitsuyama, K., Hashimoto, G., L., Ohshima, R., Kouyama T., Ando, H., and Futaguchi, M., "LIR: Longwave Infrared Camera onboard the Venus orbiter Akatsuki," Earth Planets Space 63, pp.1009-1010, 2011.
[51] CCSDS.org, "The official web site of the Consultative Committee for Space Data Systems," http://public.ccsds.org/default.aspx,
[52] Yano, H., Kubota, T., Miyamoto, H., Okada, T., Scheeres, D., Takagi, Y., Yoshida, K., Abe, M., Abe, S., Barnouin-Jha, O., Fujiwara, A., Hasegawa, S., Hashimoto, T., Ishiguro, M., Kato, M., Kawaguchi, J., Mukai, T., Saito, J., Sasaki, S., Yoshikawa, M., "Touchdown

of the Hayabusa Spacecraft at the Muses Sea on Itokawa," Science 312, 1350-1353 (2006).

[53] Okada, T., Fukuhara, T., Tanaka, S., Taguchi, M., Nakamura, R., Sekiguchi, T., Hasegawa, S., Ogawa, Y., Kitazato, K., Matsunaga, T., Imamura, T., Wada, T., Arai, T., Yamamoto, Y., Takaki, R., Tachikawa, S., Helbert, J., Mueller, T., "Thermal-Infrared Imager TIR on HAYABUSA-2 to investigate physical properties of C-Class near-Earth asteroid 1999JU3," Lunar Planet. Sci. Conf., 42, #1498, (2012).

[54] Taguchi, M., Fukuhara, T., Futaguchi, M., Sato, M., Imamura, T., Mitsuyama, K., Nakamura, M., Ueno, M., Suzuki, M., Iwagami, N., Hashimoto, G. L., "Characteristic features in Venus' nightside cloud-top temperature obtained by Akatsuki/LIR," Icarus 219, 502-504 (2012).

[55] Takada, J., Senda, S., Hihara, H., Hamai, M., Oshima, T., Hagino, S., "A fast progressive lossless image compression method for space and satellite images," Geoscience and Remote Sensing Symposium, IGARSS 2007, IEEE International, 479-481 (2007)

[56] Hihara, H., Yoshida, J., Ishida, J., Takada, J., Senda, Y., Suzuki, M., Seki, T., Ichikawa, S., Ohgi, N., "Fast compression implementation for hyperspectral sensor," SPIE 7857, 78570C (2010)

[57] Nambu, T., Takada, J., Kawashima, T., Hihara, H., Inada, H., Suzuki, M., Seki, T., Ichikawa., S., "Development of onboard fast lossless compressors for multi and hyperspectral sensors," SPIE 8527, 85270W (2012)

[58] H. Hihara, K. Moritani, M. Inoue, Y. Hoshi, A. Iwasaki, J. Takada, H. Inada, M. Suzuki, T. Seki, S. Ichikawa, J. Tanii, "Onboard Image Processing System for Hyperspectral Sensor," Sensors 2015, 15, 24926-24944; doi:10.3390/s151024926C.

[59] European Space Agency, ECSS-E-ST-50-52C "Space engineering, SpaceWire – Remote memory access protocol," (2010).

[60] Space Technology Centre, School of Computing, University of Dundee, "SpaceWire-D, Deterministic Control and Data Delivery Over SpaceWire Networks," (2010).

[61] Takahashi, T., Mitsuda, K., Kelley, R., Aharonian, F., Akimoto, F., Allen, S., Anabuki, N., Angelini, L., Arnaud, K., Awaki, H., Bamba, A., Bando, N., Bautz, M., Blandford, R., Boyce, K., Brown, G., Chernyakova, M., Coppi, P., Costantini, E., Cottam, J., Crow, J., de Plaa, J., de Vries, C., den Herder, J.-W., Dipirro, M., Done, C., Dotani, T., Ebisawa, K., Enoto, T., Ezoe, Y., Fabian, A., Fujimoto, R., Fukazawa, Y., Funk, S., Furuzawa, A., Galeazzi, M., Gandhi, P., Gendreau, K., Gilmore, K., Haba, Y., Hamaguchi, K., Hatsukade, I., Hayashida, K., Hiraga, J., Hirose, K., Hornschemeier, A.,

Hughes, J., Hwang, U., Iizuka, R., Ishibashi, K., Ishida, M., Ishimura, K., Ishisaki, Y., Isobe, N., Ito, M., Iwata, N., Kaastra, J., Kallman, T., Kamae, T., Katagiri, H., Kataoka, J., Katsuda, S., Kawaharada, M., Kawai, N., Kawasaki, S., Khangaluyan, D., Kilbourne, C., Kinugasa, K., Kitamoto, S., Kitayama, T., Kohmura, T., Kokubun, M., Kosaka, T., Kotani, T., Koyama, K,. Kubota, A., Kunieda, H., Laurent, P., Lebrun, F., Limousin, O., Loewenstein, M., Long, K., Madejski, G., Maeda, Y., Makishima, K., Markevitch, M., Matsumoto, H., Matsushita, K., McCammon, D., Miller, J.,Mineshige, S., Minesugi, K., Miyazawa, T., Mizuno, T., Mori, K., Mori, H., Mukai, K., Murakami, H., Murakami, T., Mushotzky, R., Nakagawa, Y., Nakagawa, T., Nakajima, H., Nakamori, T., Nakazawa, K., Namba, Y., Nomachi, M., O'Dell, S., Ogawa, H., Ogawa, M., Ogi, K., Ohashi, T., Ohno, M., Ohta, M., Okajima, T., Ota, N., Ozaki, M., Paerels, F., Paltani, S., Parmar, A., Petre, R., Pohl, M., Porter, S.; Ramsey, B., Reynolds, C., Sakai, S., Sambruna, R., Sato, G., Sato, Y., Serlemitsos, P., Shida, M., Shimada, T., Shinozaki, K., Shirron, P., Smith, R., Sneiderman, G., Soong, Y., Stawarz, L., Sugita, H., Szymkowiak, A., Tajima, H., Takahashi, H., Takei, Y., Tamagawa, T., Tamura, T., Tamura, K., Tanaka, T., Tanaka, Y., Tanaka, Y., Tashiro, M., Tawara, Y., Terada, Y., Terashima, Y., Tombesi, F., Tomida, H., Tozuka, M., Tsuboi, Y., Tsujimoto, M., Tsunemi, H., Tsuru, T., Uchida, H., Uchiyama, Y., Uchiyama, H., Ueda, Y., Uno, S., Urry, M., Watanabe, S., White, N., Yamada, T., Yamaguchi, H., Yamaoka, K., Yamasaki, N., Yamauchi, M., Yamauchi, S., Yatsu, Y., Yonetoku, D., Yoshida, A., "The ASTRO-H Mission," SPIE 7732, 77320Z, 18 (2010).

[62] Yamada, T., Takahashi, T., "Standard Onboard Data Handling Architecture Based on SpaceWire," International SpaceWire Conference 2008, 253-256 (2008).

[63] Yamada, T., "Proposal for Defining Standard Services Over SpaceWire –Revision A," The sixteenth SpaceWire working group meeting ESTEC, (2011).

[64] Yuasa, T., Takahashi, T., Ozaki M., Kokubun, M., "A Deterministic SpaceWire Network Onboard the ASTRO-H Space X-Ray Observatory," International SpaceWire Conference 2011, 348-351 (2011).

[65] Fujita, Y., "Personal Robot PaPeRo," J. Robotics & Mechatronics, Vol.14, No.1 (2002).

[66] Takano, Y., Yamashita, N. and Fujita, Y., "Robot-Type Integrated User Interface Platform," NEC Technical Journal, Vol.2, No.2 (2007).

[67] Robert Baumann : "The Impact of Technology Scaling on Soft Error Rate Performance and Limits to the Efficiency of Error Correction" IEDM 2002 IEEE, 2002.

[68] バスタブ曲線（故障率曲線）http://anzeninfo.mhlw.go.jp/yougo/yougo59_1.html

[69] S. B. Eryilmaz, D. Kuzum, S. Yu, H. -S. P. Wong, "Device and System Level Design Considerations for Analog-Non-Volatile-Memory Based Neuromorphic Architectures,"

IEDM15-64, pp.4.1.1-4.1.4, 2015.
[70] T. Sakamoto, M. Tada, M. Miyamoto, N. Banno, K. Okamoto, N. Iguchi, H. Hada, “Impact of overshoot current on set operation of atom switch,” J. Jrnl. of. Appl. Phys. 53, 04ED07, 2014.
[71] 南谷 崇, “フォールトトレラントコンピュータ,” オーム社, pp.11-14, 1991.
[72] C. Lavin, B. Nelson, B. Hutchings, “Impact of hard macro size on FPGA clock rate and place/route time,” 2013 23rd International Conference on Field programmable Logic and Applications, pp. 1-6, 2013, DOI: 10.1109/FPL.2013.6645510.
[73] A. Marquardt, V. Betz, J. Rose, “Speed and area tradeoffs in cluster-based FPGA architectures,” IEEE Trans. on Very Large Scale Integration (VLSI) Systems, Vol. 8, No. 1, pp. 84-93, 2000, DOI: 10.1109/92.820764.
[74] V. Garg, V. Chandrasekhar, M. Sashikanth, V. Kamakoti, “A novel CLB architecture and circuit packing algorithm for logic-area reduction in SRAM-based FPGAs,” Proc. of the ASP-DAC 2005. Asia and South Pacific Design Automation Conference, 2005, Vol. 2, pp. 791-794, 2005, DOI: 10.1109/ASPDAC.2005.1466462.
[75] Y. Hu, S. Das, S. Trimberger, L. He, “Design, synthesis and evaluation of heterogeneous FPGA with mixed LUTs and macro-gates,” 2007 IEEE/ACM Intl. Conf. on Computer-Aided Design, pp.188-193, 2007, DOI: 10.1109/ICCAD.2007.4397264.
[76] S. Korf, D. Cozzi, M. Koester, J. Hagemeyer, M. Porrmann, U. Rückert, M. D. Santambrogio, “Automatic HDL-Based Generation of Homogeneous Hard Macros for FPGAs,” 2011 IEEE 19th Annual International Symposium on Field-Programmable Custom Computing Machines, pp. 125-132, 2011, DOI: 10.1109/FCCM.2011.36.
[77] F. B. Muslim, L. Ma, M. Roozmeh, L. Lavagno, “Efficient FPGA Implementation of OpenCL High-Performance Computing Applications via High-Level Synthesis,” IEEE Access, Vol. 5, pp. 2747-2762, 2017, DOI: 10.1109/ACCESS.2017.2671881.
[78] auJ. Escobedo, auM. Lin, “Tessellation-based multi-block memory mapping scheme for high-level synthesis with FPGA,” 2016 Intl. Conf. on Field-Programmable Technology (FPT), pp. 125-132, 2016, DOI: 10.1109/FPT.2016.7929517.
[79] A. Kojima, “Trax player implementation on FPGA using high level synthesis tool,” 2016 Intl. Conf. Field-Programmable Technology (FPT), pp. 327-330, 2016, DOI: 10.1109/FPT.2016.7929565.
[80] J. P. Pinilla, S. J. E. Wilton, “Enhanced source-level instrumentation for FPGA in-system debug of High-Level Synthesis designs,” 2016 Intl. Conf. on Field-Programmable Technology (FPT), pp. 109-116, 2016, DOI: 10.1109/FPT.2016.7929514.
[81] H. Li, W. Ye, “Efficient implementation of FPGA based on Vivado High Level

Synthesis," 2016 2nd IEEE Intl. Conf. on Computer and Communications (ICCC), pp. 2810 - 2813, 2016, DOI: 10.1109/CompComm.2016.7925210.
[82] W. Ahmad, J. Iqbal, M. Martina, G. Masera, "High Level Synthesis based FPGA Implementation of H.264/AVC Sub-Pixel Luma Interpolation Filters," 2016 European Modelling Symposium (EMS), pp. 79-82, 2016, DOI: 10.1109/EMS.2016.024.
[83] F. Schwiegelshohn, F. Kästner, M. Hübner, "FPGA design of numerical methods for the robotic motion control task exploiting high-level synthesis," 2016 IEEE Intl. Conf. on the Science of Electrical Engineering (ICSEE), pp. 1-5, 2016, DOI: 10.1109/ICSEE.2016.7806074.
[84] R. Zhao, M. Tan, S. Dai, Z. Zhang, "Area-efficient pipelining for FPGA-targeted high-level synthesis," 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 1-6, 2015, DOI: 10.1145/2744769.2744801.
[85] S. Hadjis, A. Canis, R. Sobue, Y. H. Azumi, H. Tomiyama, J. Anderson, "Profiling-driven multi-cycling in FPGA high-level synthesis," 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 31-36, 2015, DOI: 10.7873/DATE.2015.0512.
[86] M. Gort, J. Anderson, "Design re-use for compile time reduction in FPGA high-level synthesis flows," 2014 Intl. Conf. on Field-Programmable Technology (FPT), pp. 4-11, 2014, DOI: 10.1109/FPT.2014.7082746.
[87] K. Fujiwara, S. Abe, K. Kawamura, M. Yanagisawa, N. Togawa, "A floorplan-aware high-level synthesis algorithm for multiplexer reduction targeting FPGA designs," 2014 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), pp. 244-247, 2014, DOI: 10.1109/APCCAS.2014.7032765.
[88] B. C. Schafer, "Allocation of FPGA DSP-macros in multi-process high-level synthesis systems," 2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 616-621, 2014, DOI: 10.1109/ASPDAC.2014.6742959.
[89] D. C. Zaretsky, G. Mittal, R. P. Dick, P. Banerjee, "Balanced Scheduling and Operation Chaining in High-Level Synthesis for FPGA Designs," 8th International Symposium on Quality Electronic Design (ISQED'07), pp. 595-601, 2007, DOI: 10.1109/ISQED.2007.41.
[90] H. Hihara, K. Wakabayashi, "PCT/JP2015005793 (EN) DYNAMIC CIRCUIT DEVICE, (FR) DISPOSITIF DE CIRCUIT DYNAMIQUE, (JA) 動的回路装置,"
Publication Date: 02.06.2016
[91] 檜原 弘樹, "特開平 11-143729 フォールトトレラントコンピュータ," 平成 11 年 (1999) 5 月 28 日
[92] H. Hihara, "US006334194B1 Fault tolerant computer employing double-redundant

structure," Grant Number: 6334194 Grant Date: 25.12.2001
[93] Univ. of Massachusetts Lecture "ECE636 Reconfigurable Computing"
http://www.ecs.umass.edu/ece/tessier/courses/636/
[94] 小林和淑, "PLD と FPGA リフレッシュ教育 2007/12,"
http://www-vlsi.es.kit.ac.jp/~kobayasi/refresh/0712/sdoc/slide/FPGA.pdf
[95] J. B. Dennis, "Data Flow Supercomputers," Computer, Vol. 13, No. 11, pp. 48-56, 1980, DOI: 10.1109/MC.1980.1653418
[96] T. Temma, M. Iwashita, K. Matsumoto, H. Kurokawa, T. Nukiyama, "Data flow processor chip for image processing, " IEEE Trans. on Electron Devices, Vol. 32, No. 9, pp. 1784-1791, 1985
[97] M. Iwashita, T. Temma, "Data flow chip ImPP and its system for image processing," ICASSP '86. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing, Vol. 11, pp. 785-788, 1986, DOI: 10.1109/ICASSP.1986.1169141
[98] K. Hiraki, S. Sekiguchi, T. Shimada, "Efficient vector processing on a dataflow supercomputer SIGMA-1, "Supercomputing '88:Proc. of the 1988 ACM/IEEE Conference on Supercomputing, Vol. I, pp. 374-381, 1988, DOI: 10.1109/SUPERC.1988.44675
[99] A. Sohn, M. Sato, S. Sakai, Y. Kodama, Y. Yamaguchi, "Nonnumeric search results on the EM-4 distributed-memory multiprocessor," Supercomputing '94:Proc. of the 1994 ACM/IEEE Conf. on Supercomputing, pp. 301-310, 1994, DOI: 10.1109/SUPERC.1994.344293
[100] A. Sohn, M. Sato, S. Sakai, Y. Kodama, Y. Yamaguchi, "Parallel bidirectional heuristic search on the EM-4 multiprocessor," Proc. of 1994 6th IEEE Symposium on Parallel and Distributed Processing, pp. 100-107, 1994, DOI: 10.1109/SPDP.1994.346176

Appendix A
領域分割モデル

本節では第 5 章に述べた高信頼性設計手法による冗長構成の信頼度を計算するための数理モデルを説明する [72, 48, 49, 50, 51]。まず一般的な確率論に基づく信頼度モデルをまとめる。システムにおける障害の発生は統計的分布をもつ確率事象と考えられる。したがって、時刻 0 にシステムに障害が無いとして、次に障害が起きる時刻を T (≧0) とすると、T は確率変数であり、Tに関する累積分布関数 $F(t)$は、

$$F(t) = P[T \leqq t] \tag{A-1}$$

で与えられる。ここで、$P[X]$は事象 X が生起する確率を表す。すなわち、$F(t)$は時刻 t 以前にシステムに障害が起こる確率である。また、Tに関する確率密度関数 $f(t)$は、

$$f(t) = \frac{dF(t)}{dt} \tag{A-2}$$

で与えられる。システムの信頼度(reliability) $R(t)$は、「時刻 0 に障害が無いという条件の下で、時刻 t までに障害が起きない確率」として定義される。したがって、

$$R(t) = P[t<T] = 1-F(t) \tag{A-3}$$

である。システムの障害率 (failure rate) $Z(t)$は、次の式で定義される。

$$Z(t) = \frac{f(t)}{R(t)} \tag{A-4}$$

微小時間$\triangle t$を考えると、

$$Z(t)\triangle t = \frac{\triangle \mathrm{F(t)}}{\mathrm{R(t)}} = \frac{P\left[t < T \leqq t + \triangle t\right]}{p[t<T]} \tag{A-5}$$

であるから、$Z(t)\triangle t$は時刻 t まで障害の起きなかったシステムに tから t +$\triangle t$までの時間に障害が発生する確率を表している。したがって、障害率 $Z(t)$は正常に動作するシステムの時刻 tにおける瞬間的な障害発生確率を表していることがわかる。

一般に、多数の電子部品で構成されるシステムが稼働する一生の間にその障害率は図 A-1 のような曲線をたどると考えられる。この障害率曲線はその形からバスタブ曲線と呼ばれる。システムの幼年期には、素子の製造欠陥を除去した後でも、ハードウェアおよびソフトウェアのバグが残るため、多くの障害が発生する。これらはシステムの試運転中のデバッグによって除去されるため、障害率は急激に減少していきやがて定常状態に落ち着く。システムが安定する実用期には、障害率はほぼ一定になると考えられる。磨耗期には、部

品の寿命などによって再び障害率は増加する。

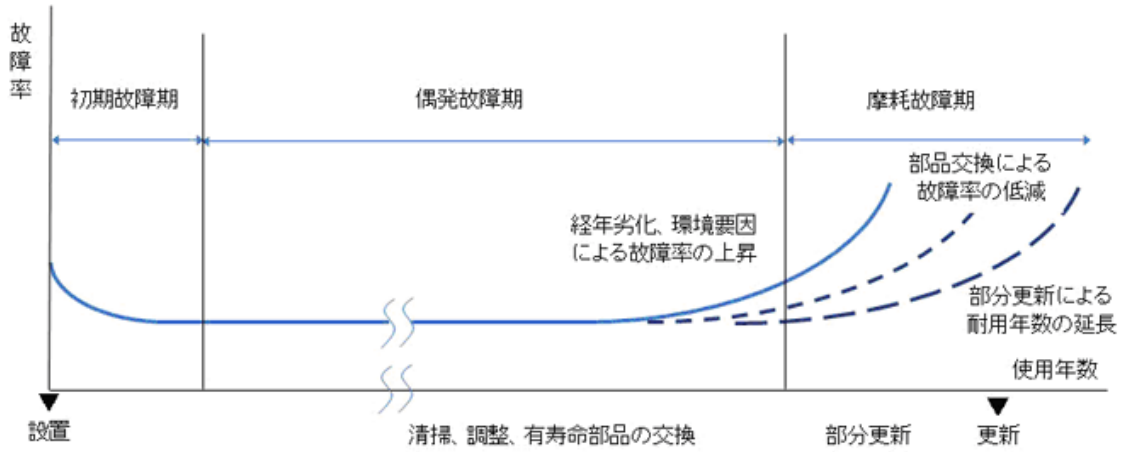


図 A-1 システムの障害率曲線（バスタブ曲線） [69]

普通、システムの実用期には障害率は一定値λになると仮定する。すなわち、

$$Z(t) = \lambda \tag{A-6}$$

そうすると、式(A-1)～(A-5)から、

$$\lambda = \frac{f(t)}{R(t)} \tag{A-7}$$

$$\lambda\, dt = \frac{f(t)dt}{R(t)} \tag{A-8}$$

$$= \frac{dF(t)}{R(t)}$$

$$= -\frac{dR(t)}{R(t)}$$

が得られる。両辺を積分すると、

$$\lambda \int_0^t t\, dt = -\int_t^{R(t)} \frac{dR(t)}{R(t)} \tag{A-9}$$

となる。ここで、積分区間は $t=0$ のとき $R(t)=1$ であることから定められる。その結果、

$$\lambda t = -\log_e R(t) \tag{A-10}$$

$$t = -\frac{1}{\lambda}\log_e R(t) \tag{A-11}$$

したがって、

$$R(t) = \exp(-\lambda t) \tag{A-12}$$

が得られる。すなわち、障害率が一定値λならば、信頼度 $R(t)$は指数関数で表される。

確率変数 T の平均値（期待値）$E(T)$は「障害が起きるまでの平均時間」であり、次式で与えられる。

$$E(T) = \int_0^\infty tf(t)dt = \int_0^\infty R(t)dt \qquad \text{(A-13)}$$

すなわち、$E(T)$は信頼度 $R(t)$の曲線と時間軸[0, ∞]間の面積として与えられる。したがって、障害率がλならば、$R(t)$は式(A-12)で表されるから、

$$E(T) = \frac{1}{\lambda}$$

となる。すなわち、「障害が起きるまでの平均時間」は障害率の逆数で与えられる。

次に、領域分割モデルについて概観する。領域分割モデルは、Fault tolerant system (FTS) の冗長構成を、形式的、定量的、汎用的（＝広い記述能力）に記述するモデル化手法である。モデルによる信頼性評価手法は一般的であり、様々な FTS を統一的に論じることができる。領域分割モデルは、FTS の構成要素であるリソースを領域で表す。リソース当たりの故障発生率が一定（＝均質な FTC）という仮定の下では、ある故障率を持つリソースは、故障率に対応した（故障率が高いほど広い）面積の領域となる。FTC の冗長構成中のシステムとして機能する最小構成は対応する領域にマッピングされ、この構成の重なりが領域の重なりに反映される[48, 49, 50, 51]。

FTC の構成要素が決まれば、故障率に応じて各々の領域のサイズが決まり、全体のサイズも決まる。また、システムとして機能する構成は、その構成要素の領域の合計サイズの領域であり、重なりを持って全体のサイズの中に納められる。この重なりの大きさが冗長構成を定量的に表すことになる。個々の領域をドメイン、システムとして機能する最小構成に対応する領域をパタンと呼ぶこととし、モデルの定式化をすると、以下のようになる。

(1) 領域全体のサイズを 1 に正規化する。
(2) パタンの数を p とする。
(3) ドメインがどのパタンに含まれているかを、2 値ベクトル k で表す。たとえば、p ＝ 3 の時、あるドメインが、パタン 1, 3 に含まれている場合、これを k ＝ (1, 0, 1)で表す。逆に、パタン 1, 3 に含まれているドメインは、k ＝ (1, 0, 1)で特定される。
(4) ドメイン k ＝ (1, 0, 1)の面積を N(k)とする。すべての k に対して N(k)を定めることにより、冗長構成が特定される。

ドメインのサイズは非負であるから、N(k) ≧ 0、合計を 1 に正規化したことにより、Σ kN(k) ＝ 1 となる。LSI のチップ上に冗長構成を持たせる場合を考えると、以上のようなモデルでの領域の面積には、チップの面積との直接的な対応が見出だせる。

ある装置の故障の数が平均 x であるとき、故障がランダムに発生する（ポアソン生起）という仮定の下では、故障率 f は指数分布に従い f ＝ 1 － e^{-x}となる。また、故障が発生しない確率 P（以下、これを信頼度と呼ぶ）は、P ＝ 1 － f ＝ e^{-x}となる。2 つの装置の平均故障数をそれぞれ x、x'とし、これに比例するようなドメインのサイズ N, N'を、正規

化（N ＋ N’ ＝ 1）して決めるとする。比例することにより、x ＝ λN、x’ ＝ λN’、装置の故障率はそれぞれ、 f ＝ 1 － $e^{-\lambda N}$、f’ ＝ 1 － $e^{-\lambda N'}$となる。これらを整理すると、以下のようになる。

$$1 - f = e^{-\lambda N} \tag{A-13}$$

$$1 - f' = e^{-\lambda N'} \tag{A-14}$$

$$\log(1 - f) = -\lambda N \tag{A-15}$$

$$\log(1 - f') = -\lambda N' \tag{A-16}$$

$$\{\log(1 - f) + \log(1 - f')\} = -\lambda(N + N') \tag{A-17}$$

$$= -\lambda \quad (\because N + N' = 1)$$

$$\text{比例定数 } \lambda = -\{\log(1 - f) + \log(1 - f')\} \tag{A-18}$$

故障率によってドメインサイズを表すと、以下のようになる。

$$N = \frac{x}{\lambda} \tag{A-19}$$

$$x = -\log(1 - f) \tag{A-20}$$

$$\therefore N = -\frac{1}{\lambda}\log(1 - f) \tag{A-21}$$

$$N' = -\frac{1}{\lambda}\log(1 - f') \tag{A-22}$$

2 値ベクトル k で表される複数ドメインについても同様に、

$$\lambda = -\Sigma k \log(1 - fk) \tag{A-23}$$

$$N(k) = -\frac{1}{\lambda}\log(1 - fk) \tag{A-24}$$

となる。信頼度 P(k) ＝ 1 － fk で表すと、次式のようになる。

$$\lambda = -\Sigma k \log P(k) \tag{A-25}$$

$$N(k) = -\frac{1}{\lambda}\log P(k) \tag{A-26}$$

$$P(k) = e^{-\lambda(N(k))} \tag{A-27}$$

冗長構成を持たせた LSI チップの場合では、λはチップ面積当たりの平均故障数に対応づけられ、以後これを故障面積と呼ぶ。

Appendix B
本方式の信頼性評価

本章では第 5 章で述べた CRAFT 方式と、単純な三重化多数決システム（Majority Voting System: MVS）構成方式を領域分割モデルによって表現し，故障率の比較評価を行う[27, 29]。MVS では外部バスの値は各 PE の出力を多数決回路によって多数決をとったものである。多数決回路に故障が存在しなければ異常出力がマスクされ，正常な値が外部バスに出力される。ここでは，両方式とも PE を三重化し，外部バスの値を多数決で決定するシステムとする。以下の故障の仮定の下で，本方式と MVS を領域分割モデルにより表現し，比較する。

・故障は，MCU・検出回路・多数決回路の出力が正常の場合と異なった値をとるものとする。
・二つ以上の故障が一サイクル内に同時に発生することは無い。すなわち，二つ以上の故障が発生する場合には、それぞれの故障の発生時刻はお互いにずれている。
・PE が正常な出力をする確率を R_C とする。
・検出回路が正常な出力をする確率を R_D とする。
・多数決回路が正常な出力をする確率を R_E とする。

この時，文献[44，45，46]に示されている領域分割モデルによる表現を用いた三重系の CRAFT の動作パタンは図 B-1 のように表される。ここでは同時に 2 つの PE が正常動作をしており、互いの診断情報を参照することによって自 PE の異常を判断できる。

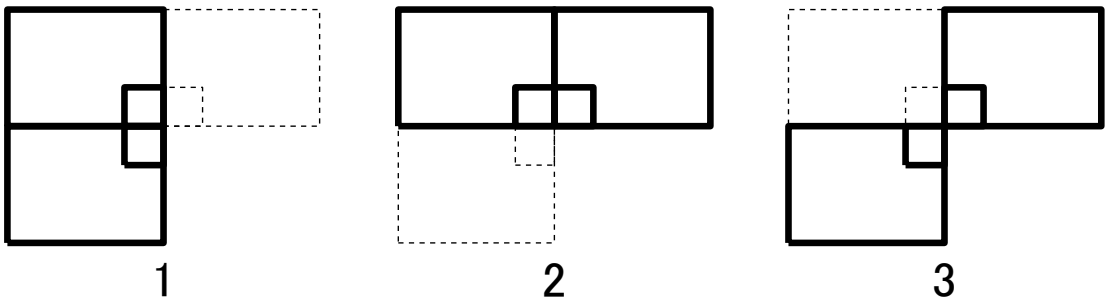


図 B-1　領域分割モデルによる CRAFT の動作パタン

ここで一つの PE を図 B-2 のように示している。検出回路以外の領域は演算リソースとして用いられている。

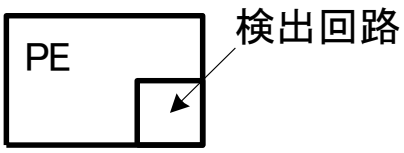


図 B-2　プロセッサ・エレメント(PE)の模式図

仮定した故障の下で定義（文献[44, 45, 46]）にしたがってシステムを分解して行くと，図 B-1 に示される太線のモジュールが正常であれば点線で書いたモジュールが故障しても，正常な出力が保証されることがわかる。また，任意の順序で故障が起きても正常な出力が保証されるためにはこれ以上の故障は許されない。この状態で一つの PE が残存して動作している場合の動作パタンによる領域分割を図 B-3 に示す。ここで，各領域の面積は R_C，R_D の値による。規格化定数λは$-3(ln(R_C \cdot R_D))$となり，各領域の面積は明らかに 1/3 となる。

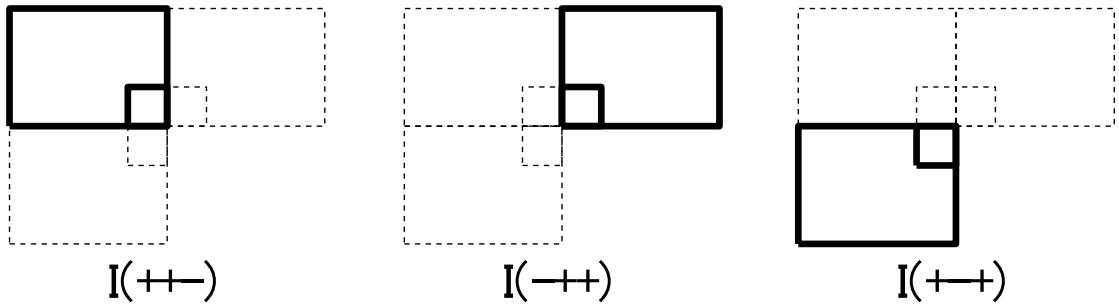


図 B-3　CRAFT の動作パタンによる領域分割

MVS の動作パターンを図 B-4 に示す。中央の正方形は多数決回路を，そのまわりの三つの長方形は PE を示す。同時に 2 つの PE と多数決回路が正常動作をしている場合の MVS の動作パタンによる領域分割を図 B-5 に示す。

ここで，各領域の面積は R_C，R_E の値による。この場合λは$-ln R_E - 4ln(R_C)$となり，各領域の面積は I(+++)は$-1/\lambda \cdot \ln RE$ ，その他は$-1/\lambda \cdot ln R_C$ になる。

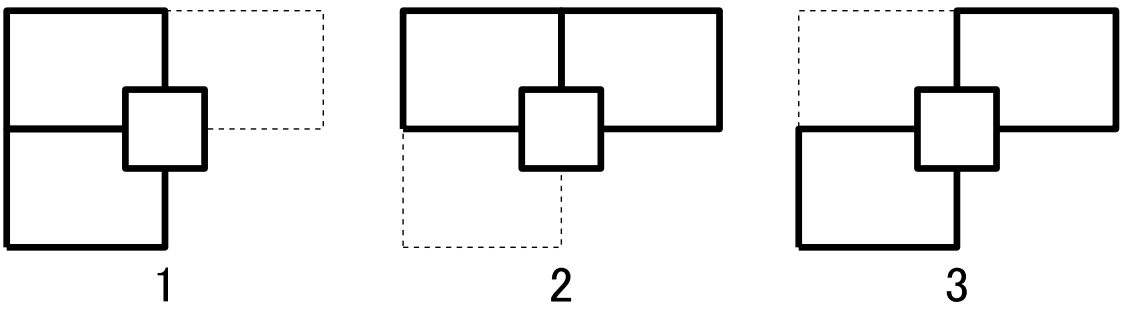


図 B-4　領域分割モデルによる MVS の動作パタン

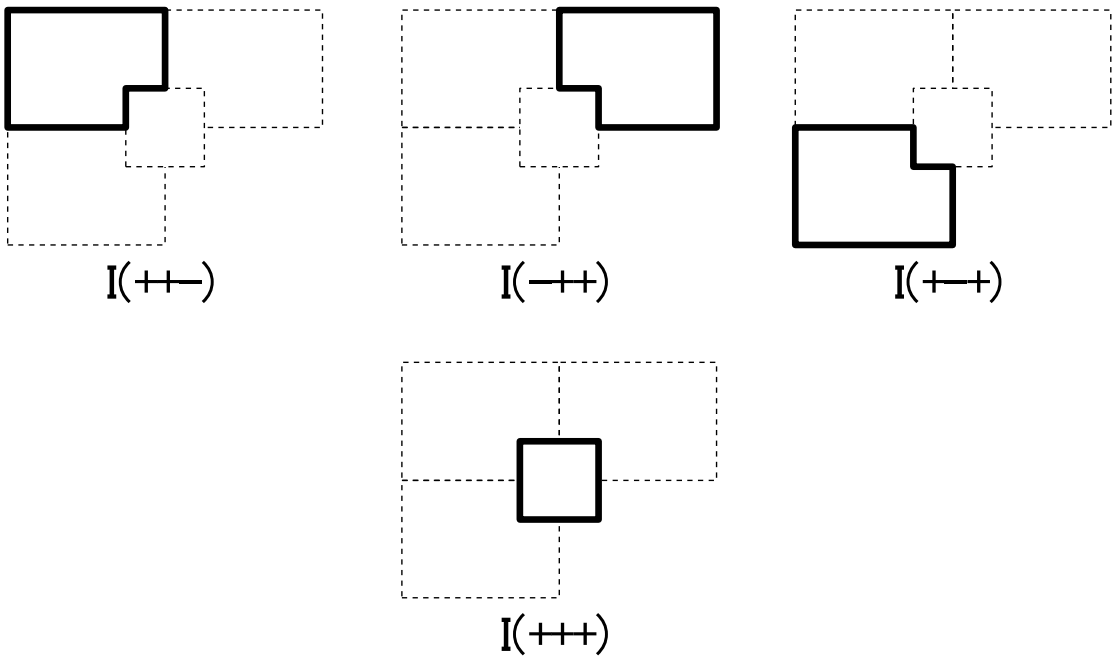


図 B-5　MVS の動作パタンによる領域分割

CRAFT, MVS とも PE の信頼度はどちらも同じであるとしているので，信頼度の差は検出回路，多数決回路に起因するものとなる。定性的には，多数決システムは多数決回路に故障が生じるとシステムダウンしてしまう可能性があるのに対して，CRAFT では PE に故障が生じているモジュールの検出回路が故障してもシステムダウンしないことから，CRAFT の方が信頼度が高いものと考えられる。次に両者の信頼度を定量的に比較する。

信頼度の定義を文献[44, 45, 46]に従い両システムの信頼度を求めると各 R_C，R_D，R_E が十分 1 に近いとした時の主要項は次のようになる。

$$R_{CRAFT} = 1-3(2-R_C-R_D)^2$$
$$= 1-3((1-R_C)+(1-R_D))^2 \quad (1)$$
$$R_{MVS} = R_E$$
$$= 1-(1-R_E) \quad (2)$$

CRAFT の信頼度が 1−(微小量)²，MVS の信頼度が 1−(微小量)で，CRAFT の方が信頼度が高くなる。RC＝RD＝RE＝0.990 とすると CRAFT の信頼度は 99.9%となり，多数決システムの信頼度は 99.0%となる。この結果、各構成要素の信頼度が同じであれば、CRAFT 方式の信頼度が MVS を上回ることが確認できた。

Appendix C
CRAFT 実証システム

本章では、実際に軌道上実証された冗長構成システムを粗粒度関数定義の観点から検証する。対象とするのは宇宙環境信頼性実証システム（Space Environment Reliability Verification Integration System: SERVIS）に搭載した自律フォールトトレラント計算機（CRAFT）である [52, 53, 54]。SERVIS プロジェクトは先端的な民生部品を使用して高性能・低コストな衛星システムを構築することを目的としたもので、新エネルギー・産業技術総合開発機構（NEDO）からの委託を受けて財団法人 無人宇宙実験システム研究開発機構（USEF）が開発を進めたものである。

2 号機（SERVIS-2）衛星にミッション機器として搭載した自律フォールトトレラント計算機（CRAFT）は，民生部品・民生技術を使用して低コスト化，小型・軽量化，高機能化し，低コスト衛星バス等に適用することを目標に開発を進めた。本装置は，前述した多数決方式を上回る信頼度を有する独自の耐故障性技術により，民生部品を活用しつつ高信頼性を実現したものである[52, 53, 54]。具体的には，単一故障点を有しない耐故障性技術，およびシステムレベルの FDIR（Fault Detection, Isolation, Re-configuration）機能を用いて高信頼性化を図り，極限環境での使用に適合させた。

CRAFT は放射線被爆環境における SEU（Single Event Upset）の発生が，全体システムの誤動作に波及しないようにするため第 5 章章に述べた Compare and Rational Abort for Fault-Tolerant [58]という独自の耐故障性方式を適用した複数のプロセッサ・エレメント(Processor Element: PE)で構成される多数決回路を持たないフォールトトレラントシステムであり，Micro Controller Unit （MCU)の処理能力を落とすことなく高信頼性を実現したものである。CRAFT の外観図を図 C-1 に示す。本装置はスライスタイプの構造を有しており，モジュラー構造を有している。このため，各モジュールが目的に応じたものに交換可能となっており，ミッション目的に応じた柔軟な構成が取れるようになっている。

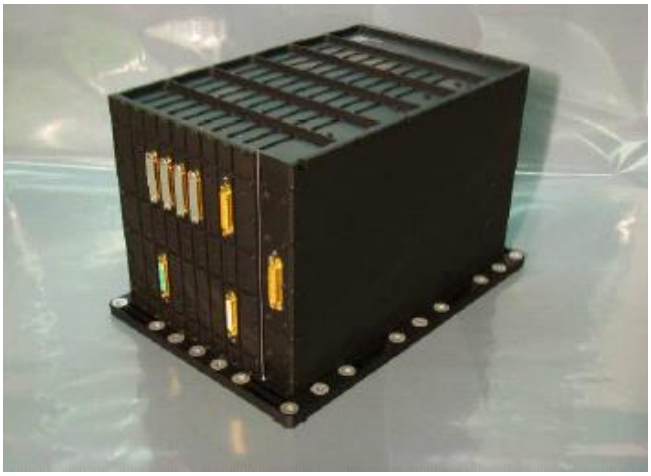


図 C-1　CRAFT の外観

本装置のブロック図を図 C-2 に示す。プロセッサ･エレメントは CRAFT 方式を用いたバスにより接続され，民生部品を使用しながら信頼度の高い計算機ユニットを構成している。

衛星バスとのインタフェースは電源部（DC-DC Converter），およびテレメトリ･コマンド・インタフェース部（Satellite Bus Interface）からなる。これらは SERVIS-2 衛星が規定する技術要求を満足するように製造され，民生部品を使用した部分に誤動作が起こったとしても，衛星バスには異常を波及させないものとなっている。これにより，民生部品を使用しながら確実に宇宙環境における軌道上実証評価が遂行できるようにしている。

Virtual Peripheral Emulation Interface では衛星バス上の他の機器を模擬しており，FDIR 機能の検証に用いられる。

CRAFT は，宇宙機に搭載するフォールトトレラントコンピュータとして単一点故障を防ぐことを主目的として開発された方式であり，日・米・欧で特許が成立しているオリジナルの技術である。単一点故障を防ぐ手段として，耐故障性を実現するために必要な各機能の分散度を高めるアプローチを採用した。

CRAFT の全体構成を図 C-3 に示す。MCU および周辺回路からなるおのおの独立した PE は異常検出ネットワークを構成しており，それぞれ外部に出力を伝える外部バスに入出力線を介して接続されている。各 PE は外部バスの内容を取り込む入力線を有し，外部バスの内容と自 PE 内の MCU からの入出力とを比較回路で比較する。

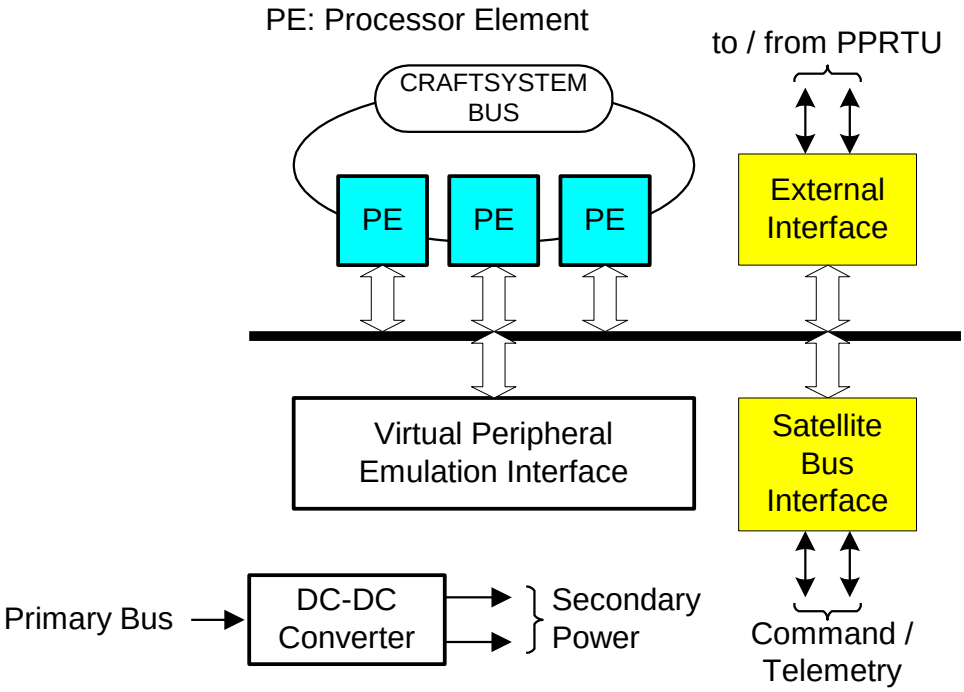


図 C-2　CRAFT のブロック図

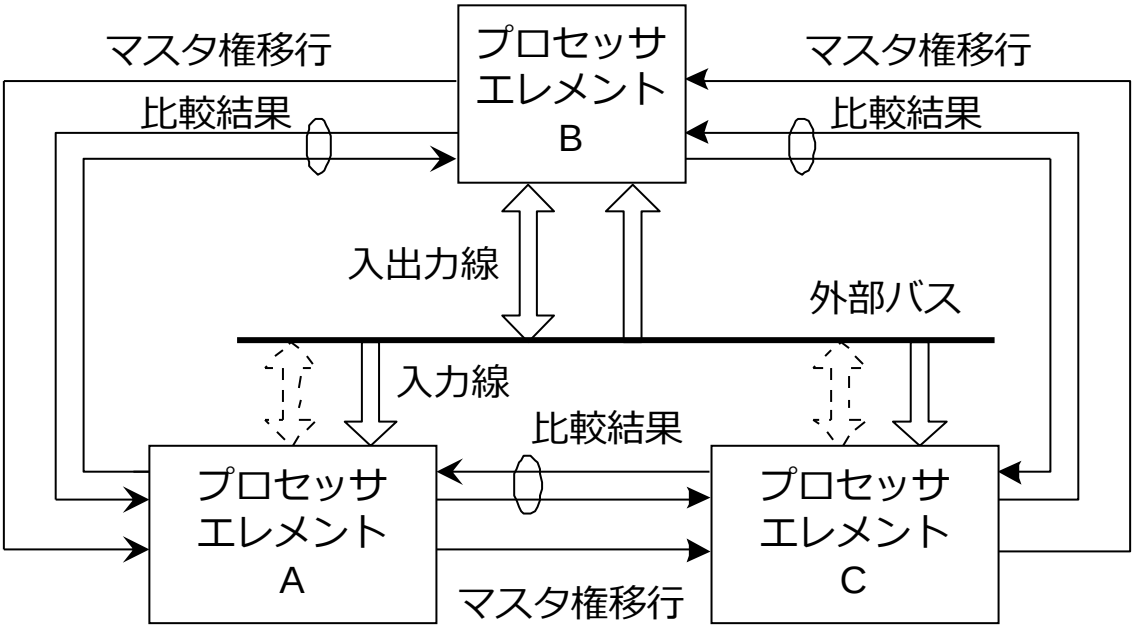


図 C-3　CRAFT の全体構成図

これらの PE は外部バスにデータを出力する際に同期して動作しており，外部入出力線に

対する出力権はその中の一つの PE にだけ与えられる。これをマスタ PE と呼び，それ以外をチェッカ PE と呼ぶ。図 C-3 では PE-A がマスタ PE，それ以外はチェッカ PE であり，チェッカ PE は出力バッファにより外部バスへの出力を断たれているので入出力線が点線で表されている。

各 PE は比較結果を伝達する出力比較信号線を通じてリング状に接続された異常検出ネットワークを構成しており，各 PE 内部に設けられた比較回路の出力である比較結果と，両隣に接続された PE から出力される比較結果との合計三つの信号とから自 PE が正常か異常かを判断する検出回路を有する。本方式では各 PE は独立に異なる処理を進めることが可能であり、図 C-4 に示すようにそれぞれの PE が別々の構成、かつ動作を行うことが軌道上実証された。

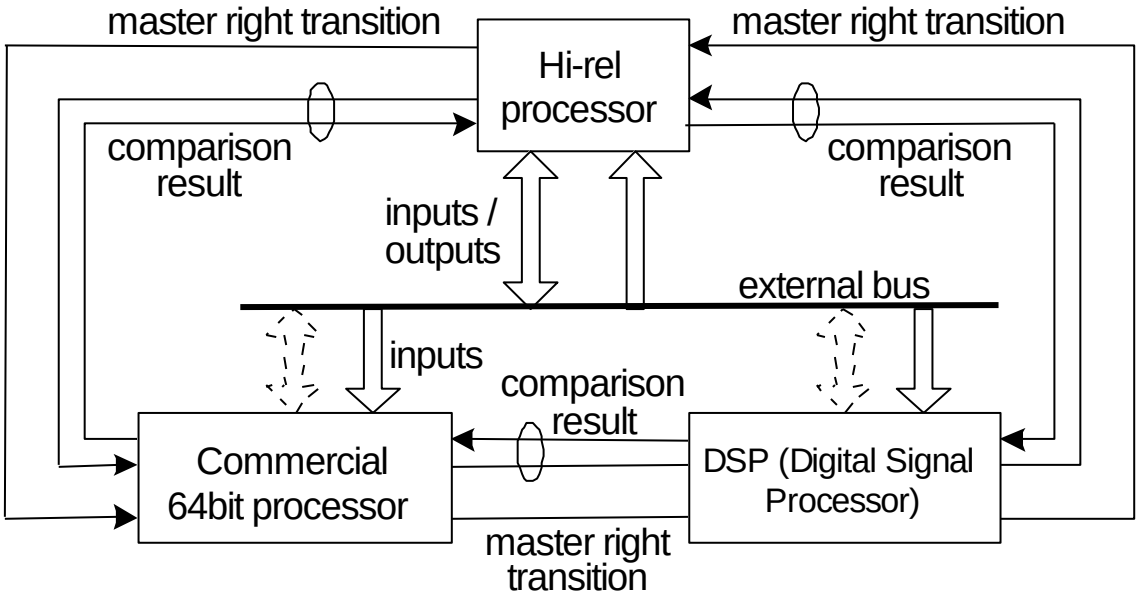


図 C-4　各 PE の独立動作

Appendix D
実用性の評価

粗粒度関数定義層の実用性を評価するために、同等の機能をソフトウェアにより実装した、はやぶさ 2 搭載用デジタル信号処理装置[49]の仕様を再評価した。はやぶさ 2 は、地球から数億キロ離れた C クラスの近地球小惑星 162173（1999JU3）の探査のために 2014 年に打ち上げられた小惑星探査機である。はやぶさ 2 の外観を図 D-1 に示す。はやぶさ 2 には光学航法カメラ（Optical Navigation Camera: ONC）、熱赤外線イメージャ（Thermal Infrared ImagerR: TIR）、3000nm の吸収帯を識別する近赤外分光器（Near Infrared Spectrometer 3: NIRS3）、LIDAR（Light Detection and Ranging）、および LRF（Laser Range Finder）を搭載しており、小惑星の地質、熱物性、有機物などを調査することを目的としている[53, 54, 55]。ONC は、はやぶさ初号機の開発設計を継承しており、科学的観測目的、ならびに試料採取のための着陸地点の選択および小惑星表面への安全な降下を支援するためにも使用される。TIR は JAXA / ISAS で開発された金星気象衛星あかつき用に開発された LIR（Long-wavelength Infrared Imager）の設計を継承しており、熱赤外線観測に用いられる。ONC、NIRS3、および TIR の緒元を表 D-1 に示す。

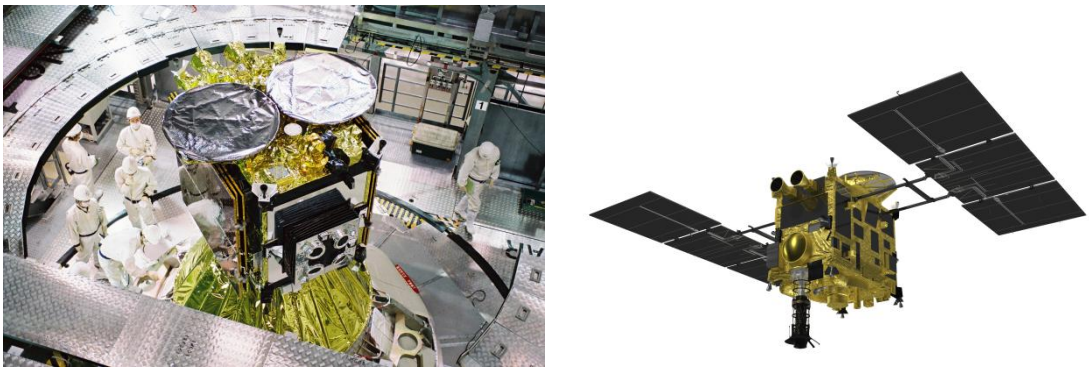


図 D-1　はやぶさ 2　© JAXA

表 D-1　ONC、NIRS3、および TIR の緒元

Parameter / Sensor	ONC-T	NIRS3	TIR
Wave length	300 - 1100 nm	1800 - 3200 nm	8000 - 12000 nm
FOV	5.7 deg × 5.7 deg	0.1 deg × 0.1 deg	12 deg × 16 deg
IFOV	0.1 mrad	-	0.877 mrad
Pixel Number	1024 × 1024	128	320 × 240
Data size	2 MB per raw view	-	0.15 MB/image
MTF (@nyquist freq。)	-	-	>0.3
Wave dispersion	-	20 nm/pixel	-
Detector	CCD (silicon charge coupled device)	InAs linear image sensor	non-cooled bolometer
Temp。 range	-	-	250 - 400K
NETD	-	-	<0.5K (@350K)
Absolute T resolution	-	-	<5K (@350K)
Temp。 Calibration	-	-	Shutter Open/Close
Shutter	-	100 Hz (typical)	1 sec for open or close
filter	ul, b, v, Na, w, x, p, Wide	-	-
ADC	12 bit	16 bit	12 bit
Heritage	HAYABUSA ONC and AKATSUKI UVI	HAYABUSA (3 μ m range is new)	Akatsuki, a Venus climate orbiter

はやぶさ 2 の信号処理システムは ONC-E、および DE で構成している。ONC-E、および DE の電子機器ユニットは、ONC および TIR によって取得されたデータをオンボードで処理するデジタル画像処理機能を有する。図 D-2 に ONC-E の外観を示す。

図 D-2　ONC-E / DE

地球局との間のダウンリンク伝送容量が 32kbps までに制限されているため、オンボード機能として自動および自律機能を実装している。これらの自動および自律機能の操作シナリオはプログラム可能とした。ONC-E と DE のセンサ制御と画像操作の処理手順は専用のスクリプト言語（観測プログラム）によりプログラミングする。観測プログラムには、いつ、どのタイミングで、どのようなトリガーで、どのようなシーケンスで、およびどのような条件で各処理が実行されるかを記述する。この観察プログラムを作成するために、スクリプトをチェックするためのシミュレーション機能を備えた GUI エディタとハードウェアシミュレータを用意した。この自動・自律機能はロボット技術を応用して開発した[66, 67]。人工衛星の動作環境は一般的なロボットの動作環境とは異なるため、はやぶさ 2 にロボット技術を適用するためには修正が必要であった。主な違いは、ロボットのソフトウェア開発フレームワークがイベント駆動型スキームに基づいているため、タイムラインの概念を新たに追加する必要があった。はやぶさ 2 の自動自律運用のために、タイムラインの枠組みとイベント駆動方式との間の一貫性を確立しなければならなかった。センサ制御動作及び画像動作はマクロコマンドとして実現され、エラー回復はマクロコマンドシーケンスとしてプログラムされる。

取得されたすべてのデータを記録・伝送するには、画像処理が十分高速でなければならない。画像処理は ONC-E および DE の限られたリソース内で実行しなければならないため、ハードウェアおよびソフトウェアの実装は、慎重にトレードオフして実装した。このために、前述した C 言語ベースの高位合成設計ツールである CyberWorkBench をソフトウェア設計と FPGA 設計の両方に使用した。

オンボードで実装された信号処理機能を表 D-2 に示す。これに示されるように、基本的

な四則演算と画像処理機能や画像圧縮機能などの複合演算を同レベルの OPE-code として使用できるものとしている。画像認識機能のプリミティブはハードウェアとして実装されており、これを観測プログラムから利用するための API として OPE-code が実装されている。画像圧縮機能は、採用している StarPixel アルゴリズムの処理速度が十分速いため、ソフトウェアとして実装されている。複数画像の加算、ダーク画像減算、およびデッドピクセルの補正などの画像処理は、ハードウェアとソフトウェアの両方によりオンボードで処理される。これまでの約二年間の運用実績から、この粗粒度関数定義の概念に基づいた OPE-code は問題無く運用に供されていることが確認できた。この結果により、実用性は検証できたものと考える。

表 D-2 オンボード信号処理機能の実装

Category	OPE-code	Operation
Arithmetic Operation	ADD, SUB, MULT, DIVOP, ADD(C), MULT(C), DIV(C)	Arithmetic image operation between stored images, (C) means constant
Compression	SPIX_LOSSLESS, SPIX_FLEX	Lossless/lossy compression with StarPixle®
Operation Control	NOP, SWB_CLR, BUF_RESET, BUF_SETUP, BUFPTR_INC, IMG_SETUP	Do nothing, Clear software operation buffer, Reset buffer management information, Initialize buffer number, Increment buffer number, Setup image information
Transfer Control	COPY, OB_COPY, REC, TRANS	Copy an image, Copy optical black data, Record image without compression, Transfer an image to specified address
Image processing	FILTER_THRES, CUT, TRANSPOS, DEAD_PIXEL, MEDIAN, MEAN, MODE, BIN	Threshold manipulation, Fill outside area with 0, Transpose image data, Processing dead pixel, Calculate median, mean, mode of images or make 2x2 binning image

索引