

不揮発性メモリを活用した三次元積層 LSI による 高速電源断リカバリシステムの提案

浦川 晃[†] 松尾 俊輝[†] 青柳 昌宏^{††} 大川 猛^{††}

[†]熊本大学大学院自然科学教育部 〒860-8555 熊本市中央区黒髪 2-39-1

^{††}熊本大学半導体・デジタル研究教育機構 〒860-8555 熊本市中央区黒髪 2-39-1

E-mail: [†]{258d8818, 263d8856}@st.kumamoto-u.ac.jp, ^{††}{m-aoyagi, ohkawa-takeshi}@kumamoto-u.ac.jp

あらまし ロボットや IoT デバイスの信頼性向上に向け、不揮発性メモリへの迅速なデータ退避・復元を行う電源断リカバリ技術が重要である。本研究では、プロセッサ層と不揮発性メモリ層を TSV（シリコン貫通電極）で接続する三次元積層 LSI を用いた高速な電源断リカバリシステムを提案する。具体的には、FPGA 上の SoC と Zephyr RTOS を用いた疑似電源断リカバリシステムの実装と評価を行った。本稿ではこの実証結果を基に、遅延要因の分離と分析を行い、従来の平面配線における物理的ボトルネックを定量的に明らかにする。さらに、その根本的な解決策として三次元積層 LSI を用いたアーキテクチャの必要性と今後の展望について議論する。

キーワード ロボット, IoT, 三次元積層 LSI, 不揮発性メモリ, FPGA, Zephyr

1. はじめに

近年、ロボットや IoT デバイスは多様な分野で実運用が進んでいる。これらのシステムは、複雑な制御や通信処理を実現するため、OS 上で動作することが一般的である。しかし、実環境においてはバッテリーの脱落や配線不良などにより、予期せぬ電源断が発生するリスクが常に存在する。即ち、電源断によって姿勢推定値や自己位置、制御パラメータといった動的な制御コンテキストが揮発性メモリから消失すると、再起動後の制御不整合による転倒や衝突などの物理的被害を引き起こす危険性がある。したがって、電源断直前の状態を保持し、復旧後に安全かつ迅速に動作を再開する電源断リカバリ技術が極めて重要である。

本研究では、ロボット IoT システムに求められる要求条件に適したリカバリ構成を導出し、ボトルネックとなるデータ転送帯域を根本から解決するため、プロセッサ層と不揮発性メモリ層を直接接続する三次元積層 LSI を用いた高速な電源断リカバリシステムを提案する。本稿では、既存の平面配線アーキテクチャにおける課題を定量的に整理し、三次元積層 LSI の導入に向けた設計指針と今後の展望について議論する。

2. 電源断リカバリの要求仕様とアーキテクチャの導出

2.1. 設計対象と要求条件

本研究では、ロボットや IoT デバイスにおける電源断リカバリを対象とする。これらのシステムが電源断から安全かつ確実に復帰するためには、以下の 4 つの要求条件を同時に満たすアーキテクチャの設計が必要である。まず、(1)姿勢推定値や環境地図、制御パラメータ等の動的な制御コンテキストを扱うため、電源断

直前の最新状態を復元できることが求められる。同時に、(2)高頻度な制御ループを阻害しないよう定常稼働時の実行オーバーヘッドを最小限に抑える必要がある。さらに、(3)RTOS 上で動作する複雑なソフトウェア資産を大幅なコード改修なしに保護する柔軟性と、(4)数十 KBytes 以上に及ぶ大容量アプリケーションデータを限られた時間内に退避できる物理的な転送能力が必須である。

2.2. 電源断耐性アーキテクチャの設計軸と先行研究

電源断耐性システムのアーキテクチャは多様な観点から議論されてきたが、本研究では既存手法の技術的位置づけを体系的に整理するために、いくつかの先行研究に基づいて、「保存タイミング」「保存場所」「退避対象の範囲」という 3 つの設計軸を定義する。

第一の軸である状態保存のタイミングは、処理の区切りで状態を退避するチェックポイント手法と、電源断の予兆を検知して退避を行うイベント駆動手法に大別される。Mementos [1] は、コンパイラを用いてソフトウェア的にチェックポイント処理を挿入する先駆的な研究である。また、間欠コンピューティングの分野では、電源断をまたいだ再実行時のメモリ不整合を防ぐため、タスク分割や幂等性の保証に基づくチェックポイント手法 (DINO [2], Ratchet [3]) も広く提案されている。一方、Hibernus [4] 等は、ハードウェア割り込みを利用して電源断直前に一括退避を行うイベント駆動型のアプローチを提案している。

第二の軸であるデータの保存場所は、外部の不揮発性メモリを利用する構成と、回路レベルで状態を保持する構成に分類される。後者の極致として、RISC-V 等

の CPU のフリップフロップや SRAM 自体を不揮発性素子で実装する不揮発性プロセッサ (NVP) の研究 [5] が挙げられる。

第三の軸である退避対象のスコープは、CPU のレジスタやスタック、OS の管理領域を含めたシステム全体を保存する「システムレベル・チェックポイント」と、アプリケーションの実行継続に不可欠な変数データのみを抽出して保存する「アプリケーションレベル・チェックポイント」に二分される。Sytare [6] は周辺機器のレジスタ状態まで、IntOS [7] はマルチタスク環境における OS 状態までシステムレベルの退避対象を拡張した研究である。

2.3. 要求条件に基づくアーキテクチャの導出

前節の設計軸に基づき既存方式を検討すると、定期保存は定常稼働時のオーバーヘッドが大きく、RTOS 環境のリアルタイム性を阻害する課題がある。また、OS 内部状態まで保存するシステムレベルの手法は、復旧時の依存関係が複雑でソフトウェア遅延を増大させる。一方、Hibernus [4] 等のイベント駆動手法は理想的だが、数 KBytes 程度の小規模データのみを想定しており、ロボットに必要な大容量データへの対応が欠落している。

これらの検討を踏まえ、本研究ではロボット IoT システムに最適な構成として、定常負荷ゼロで最新状態を維持できる「イベント駆動保存」と、大容量データを収容可能な「外部不揮発性メモリ」の組み合わせを導出した。退避スコープについては、OS のクリーンな再起動を前提とし、制御継続に不可欠な変数データのみを保護する「アプリケーションレベル・チェックポイント」を採用する。本構成の実現に向け、電源断の予兆を捉える検知回路と、退避時間を稼ぐためのバックアップ電源を搭載したシステムを対象とする。

2.4. 物理的ボトルネックと三次元積層の必要性

導出された構成を実用化する上で、最大のボトルネックとなるのが退避処理にかかる総合的な遅延時間である。専用の検知回路とバックアップ電源を導入したとしても、ロボットの小型化やコストの制約上、搭載可能なコンデンサ容量には物理的な限界がある。

退避すべきデータが数十 KBytes 以上に及ぶ場合、限られた猶予時間内にすべてのデータを退避しきるには、プロセッサ・不揮発性メモリ間のデータ転送遅延と不揮発性メモリ内部の書き込み遅延の双方を極限まで削減しなければならない。しかし、既存の平面配線によるオフチップ接続では、シリアル通信の帯域制約が厳しいだけでなく、高速なメモリを並列駆動させるための配線実装にも限界があるため、総合的なメモリ

アクセス性能が頭打ちとなる。遅延に合わせてバックアップコンデンサを無尽蔵に巨大化させることは非現実的である。

したがって、本研究ではこの限界を根本から突破する解決策として、プロセッサ層と不揮発性メモリ層を最短距離で垂直に接続する三次元積層 LSI アーキテクチャを導入する。TSV (シリコン貫通電極) を用いた超広帯域通信によるデータ転送時間の削減と、高速メモリブロックとの密結合を可能にすることで、現実的なサイズのバックアップ電源の猶予時間内での確実な大容量データ退避を実現する。

3. 提案する高速電源断リカバリシステム

3.1. システム構成の概要

本研究で提案する電源断リカバリシステムは、プロセッサ層、不揮発性メモリ層、電源断検知回路、およびバックアップ電源を垂直に統合した三次元積層 LSI を中核とする。図 1 にシステムの全体像を示す。

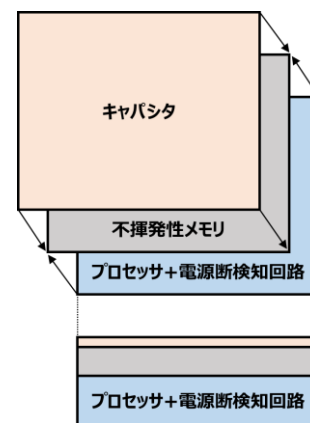


図 1 提案するシステムの全体像

本構成において、将来の完全な三次元積層化を見据え、プロセッサと不揮発性メモリに加え、電源断検知回路とバックアップ電源のすべてを統合したアーキテクチャを想定する。バックアップ電源には、現在熊本大学で開発が進められているチタン酸バリウム (BaTiO_3) を超える高誘電率材料を用いた三次元積層可能なキャパシタの統合を想定する。これにより、基板配線に起因する寄生インダクタンスが排除され、電源断直後の過渡的な大電流要求に対しても極めて高速かつ安定した電力供給が可能となる。

また、電源断検知回路を内部統合するにあたり、平滑コンデンサの放電による検知遅延を回避するため、主電源の最上流電圧を直接監視する専用のピンを設ける。この完全統合構成により、外部部品に一切依存することなく、電圧低下を捉えた瞬間にチップ内部の超高速な割り込み処理によって退避を開始し、大容量の制御コンテキストを極めて短い猶予時間内で保護する。

3.2. 三次元積層 LSI による超広帯域接続

本システムの基盤となるハードウェアとして、プロセッサと不揮発性メモリ間の接続に TSV（シリコン貫通電極）を用いた三次元積層構造を想定している。図 2 に、本システムで前提とする三次元積層 LSI の物理的な構造の概念図を示す。

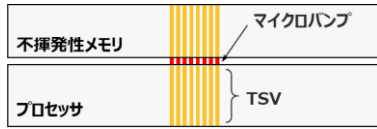


図 2 三次元積層 LSI の物理的な構造の概念図

従来の平面配線では、基板上の配線長に起因する寄生容量やインダクタンスの影響により、SPI や I2C といったシリアル通信によるデータ転送が一般的であり、その実効帯域は数十 Mbps 程度に制限される。これに対し、提案する三次元積層構造では、プロセッサ層の直上に不揮発性メモリ層を配置し、数千本規模の TSV で垂直に接続することで、配線長をマイクロメートルオーダーにまで劇的に短縮する。この物理構造により、従来のバスインターフェースの制約を超えた広帯域な並列データ転送が可能となる。

3.3. イベント駆動型リカバリシーケンス

本システムにおける電源断検知から動作再開までのシーケンスを図 3 に示す。

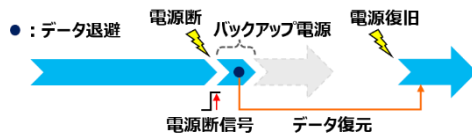


図 3 電源断検知から動作再開までのシーケンス

電源断検知回路が供給電圧の閾値下回りを検知すると、プロセッサに対して即座に割り込み信号を出力する。割り込みを受け取ったプロセッサは、実行中のタスクを中断し、予め指定された重要な変数データのメモリアドレス範囲を三次元積層バスを通じて不揮発性メモリへと一括転送する。この一連の処理は、バックアップ電源がシステム稼働限界電圧に達するまでの猶予時間内に完了する。

電源が復旧すると、システムはクリーンなブートシーケンスを開始する。OS の初期化完了後、アプリケーションは不揮発性メモリから退避データを読み出し、電源断直前の内部状態を復元する。本手法では、OS 全体のレジスタやスタックといったカーネル内部の状態保存を排除した「アプリケーションレベル・チェックポイント」を採用しているため、再起動後の OS の一貫性確保が容易であり、複雑なソフトウェアスタック上でも安定した復旧を実現している。

3.4. 退避データの構造化と整合性保証

本システムでは、退避すべきアプリケーションデータを特定の構造体として明示的に定義し、一括転送を行う。この際、電源断という極めて不安定な過渡状態においてデータを不揮発性メモリへ書き込むため、転送途中でシステムが完全にダウンし、データが破損するリスクを考慮する必要がある。

そこで本システムでは、退避データ群に対して「Magic ナンバー」と「CRC コード」をヘッダとして付与するデータ構造を用いる。エラー検出手法として単純なパリティやチェックサムではなく CRC を用いた理由は、電源断時の不安定な電圧降下によって発生しやすい連続的なビット反転を、少ない計算オーバーヘッドで高確率に検出できるためである。加えて、Magic ナンバーを併用することで、未初期化のメモリ領域や完全に欠損したデータ群を即座に判定し、無駄な検証処理を省くことができる。

電源復旧後の再起動時、アプリケーションは不揮発性メモリからデータを読み出す際、まず先頭の Magic ナンバーを確認し、続いてデータ長に基づく CRC 計算を行って保存された CRC 値と照合する。この整合性検証プロセスを経ることで、不完全な退避データによるロボットの誤動作をハードウェア・ソフトウェアの両面から防止している。

4. FPGA および Zephyr RTOS を用いた疑似電源断リカバリシステムの実装

4.1. ハードウェア評価基盤の構築

提案する三次元積層 LSI アーキテクチャの有効性を評価するためのベースラインとして、FPGA を用いた疑似電源断リカバリシステムを構築した。評価基盤は、Xilinx 社製 Artix-7 を搭載した Arty A7-100T FPGA ボードを用いた。

SoC（System on Chip）の構築には、オープンソースのハードウェア構築フレームワークである LiteX [8] を採用した。LiteX は、Python ベースの記述により CPU コア、バス構造、およびメモリコントローラを柔軟に統合・変更可能な環境を提供する。本評価系では、プロセッサとして 100 MHz で動作する 32 bit RISC-V コアの VexRiscv [9] を実装し、周辺回路として UART、タイマー、および LiteSPI コントローラを統合した。図 4 に、FPGA 上に構築した SoC の全体構成を示す。

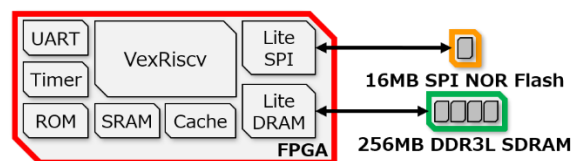


図 4 FPGA 上に構築した LiteX-VexRiscv SoC 構成

現状の平面配線における物理的なボトルネックを模擬するため、プロセッサと不揮発性メモリ間は SPI 通信で接続し、ボード上の SPI NOR Flash メモリ (Infineon S25FL128S) を退避先として利用した。

4.2. Zephyr RTOS の概要と選定理由

実用的なロボット制御環境を想定し、評価用 OS として Linux Foundation が主導するオープンソース RTOS である Zephyr [10] を用いた。Zephyr は、高度なモジュール性とスケーラビリティを備えた設計を特徴とする。本研究における選定理由は以下の 2 点にある。

第一に、業界標準のサポートと信頼性である。Zephyr は RISC-V アーキテクチャおよび LiteX フレームワークにおいて標準的なサポートを受けており、オープンソース・コミュニティによる継続的なメンテナンスが行われているため、容易に同一の評価環境を再現可能である。

第二に、実用的なソフトウェアスタックにおける性能検証である。実際のシステムでは、ベアメタル環境のような直接的なレジスタ操作ではなく、OS のセマフォや多層的なドライバ階層を経由してメモリへアクセスする。このような現実的なソフトウェア・オーバーヘッドを含めても、提案する高速リカバリ手法が電源断の猶予時間内に収まるかを実証することは、実用性を担保する上で不可欠である。

4.3. ソフトウェア実装の詳細

データの退避および復元処理において、以下の 2 つのロジックを実装した。

4.3.1. ベアメタル環境における実装

ベアメタル環境では、SPI コントローラの制御レジスタ (CSR) を直接操作し、1 Byte 単位で物理通信を制御する。

```
void save_context(uint32_t addr, uint8_t *data, int len){
    spiflash_cs_low();
    spi_write_byte(CMD_WRITE);
    for (int i = 0; i < len; i++){
        spi_write_byte(data[i]);
    }
    spiflash_cs_high();
    wait_flash_ready_polling();
}
```

ここでは、チップセレクトの制御から 1 Byte ごとのループ転送、および完了時のステータス監視のすべてを、プロセッサが直接管理する。

4.3.2. Zephyr RTOS 環境における実装

Zephyr 環境では、カーネルの標準 API を介して処理を行う。ハードウェアの具体的な制御はドライバ層に隠蔽されている。

```
void save_context(const struct device *dev, uint32_t
addr, void *data, int len){
    if (k_sem_take(&save_sem, K_FOREVER) == 0){
        flash_write(dev, addr, data, len);
    }
}
```

ここでは、割り込み処理から発行されるセマフォを待機し、取得後に flash_write() を呼び出す。アプリケーション層からは単純な関数呼び出しに見えるが、内部ではカーネルの排他制御や多層的なドライバ階層を経由するため、ベアメタルと比較してソフトウェア起因の実行遅延が発生する。

4.4. リカバリフローの実装と整合性保証

本評価系では、2 つのプッシュボタンを用いて電源断から復旧までの一連のサイクルを疑似的に再現した。図 5 に実証環境における制御フローを示す。

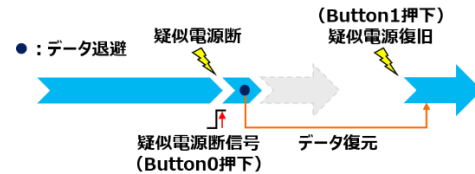


図 5 疑似電源断リカバリシステムの制御フロー

まず、疑似的な電源断信号として Button0 を用い、これをプロセッサへのハードウェア割り込み信号として入力する。信号検知後、割り込みサービスルーチン (ISR) が起動して実行中のタスクを中断し、制御コンテキストを Flash メモリへ退避する。データの整合性保証のため、退避時には CRC32 チェックサムと Magic ナンバーを付与する。退避完了後、システムは待機状態となり電源断状態を模擬する。

次に、疑似的な電源復旧信号として Button1 の押下を用いる。本実装では、Button1 の押下を契機に物理的な電源再起動は行わず、ソフトウェア的に OS のブートシーケンスを最初から実行させる構成とした。システムは再起動直後の初期化プロセスにおいて Flash メモリを確認し、Magic ナンバーと CRC の照合により整合性が確認された場合のみ、保存されたシーケンス番号に基づき電源断直前の状態からアプリケーションの実行を再開する。

このように、ボタン操作を起点とした OS の再ブートとデータ復元処理を組み合わせることで、実機にお

けるリカバリサイクルを正確かつ再現性高く評価可能な環境を実現した。

5. 遅延モデルの定式化と実機評価

5.1. リカバリ処理の統合遅延モデル

本研究では、電源断リカバリ処理に要する時間を定量的に評価し、ソフトウェア処理とハードウェア処理を統合した遅延モデルを定義する。電源断検出後、退避または復元処理が完了するまでに要する総時間 T_{total} [ms] は、PIO (Programmed I/O) によるデータ転送や OS の API スタック等を含むソフトウェア処理時間 T_{sw} [ms] と、物理的なデータ転送およびメモリアクセスに要するハードウェア処理時間 T_{hw} [ms] の和として表される。

$$T_{total} = T_{sw} + T_{hw} = T_{sw} + (T_{trans} + T_{acc}) \quad (1)$$

ここで、 T_{hw} は、通信バスを介したデータ転送時間 T_{trans} と、不揮発性メモリ内部の物理アクセス時間 T_{acc} から構成される。

データ転送時間 T_{trans} [ms] は、以下の式で定式化される。

$$T_{trans} = \frac{(S_{data} \times 8) + B_{cmd}}{W_{bus} \times f_{clk}} \times 1000 \quad (2)$$

ここで、 S_{data} は退避または復元する総データ量 [Bytes]、 W_{bus} は通信バス幅 [bits]、 f_{clk} は通信クロック周波数 [Hz]、 B_{cmd} は通信プロトコルに伴うコマンドやアドレス指定等のオーバーヘッド総ビット数 [bits] である。

また、不揮発性メモリ内部の物理アクセス時間 T_{acc} [ms] は、メモリデバイスの動作仕様に依存する。本実証で用いた Flash メモリ (S25FL128S) のデータ退避 (書き込み) においては、物理ページ単位での書き込み処理を要するため、次式で定義される。

$$T_{acc} = \left\lceil \frac{S_{data}}{S_{page}} \right\rceil \cdot t_{pp} = \left\lceil \frac{S_{data}}{256} \right\rceil \cdot 0.25 \quad (3)$$

ここで、 S_{page} はデバイスの物理ページサイズ [Bytes]、 t_{pp} はデータシートに基づく 1 ページあたりの物理的な書き込み時間 [ms] である。一方、データ復元 (読み出し) においては、通常の Read コマンド (03h) を使用している。同コマンドはアドレス入力完了直後のクロックからデータ出力が開始され、デバイス内部でのレイテンシが発生しない仕様である。したがって、データ復元時におけるメモリ内部の物理アクセス時間は $T_{acc} = 0$ と定義できる。

この統合遅延モデルを用いることで、後の実機評価において計測される総時間 T_{total} から、理論的なハードウェア処理時間 T_{hw} を差し引き、ソフトウェア

起因遅延 T_{sw} を定量的に分離・抽出することが可能となる。

5.2. 通信周波数およびデータ量変化による実測

構築した評価系を用い、SPI 通信周波数を 2.5 MHz および 25 MHz の 2 段階、データ量 S_{data} を 64 Bytes から 65,536 Bytes まで変化させ、総処理時間 T_{total} を実測した。計測にはプロセッサ内部のサイクルカウンタを使用し、Zephyr RTOS 環境とベアメタル環境の双方で評価を行った。退避時間の実測結果を図 6 に、復元時間の実測結果を図 7 に示す。全ての条件においてデータ量の増大に伴い処理時間は線形に増加したが、同一データ量であっても退避時間が復元時間に比べて長い傾向にあることが示された。これは、式(3)で示した Flash メモリ特有の物理書き込み遅延 T_{acc} が寄与しているためである。

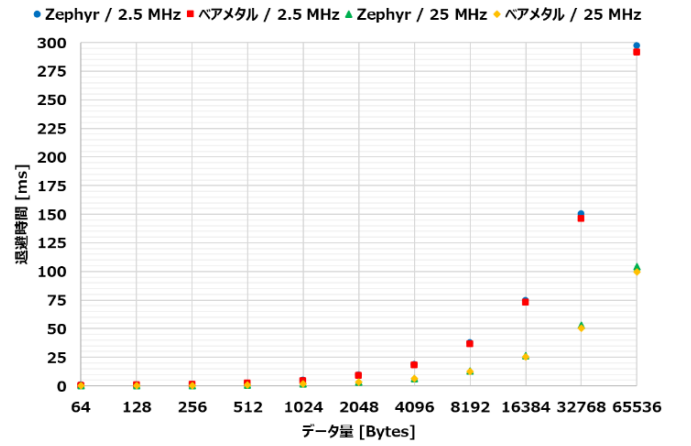


図 6 退避時間の実測結果

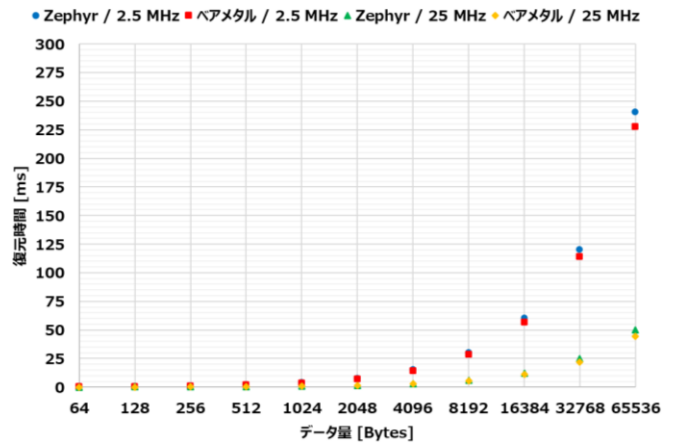


図 7 復元時間の実測結果

5.3. ソフトウェア・ハードウェア遅延の分離と考察

5.1 節で定義した統合遅延モデルに基づき、実測された退避・復元時間 T_{total} から理論的なハードウェア処理時間 T_{hw} を差し引くことで、ソフトウェア起因の遅延 T_{sw} を算出した。

本評価では、小規模な状態フラグから大容量の制御コンテキストまで幅広い退避要件を網羅するため、実測したデータの中から代表的な4つのデータサイズを抽出し、SPI通信周波数25 MHz (Zephyr RTOS 環境) における遅延要因の分解結果を表1に示す。

表1 実測値に基づく退避および復元処理の遅延要因分解 (SPI通信周波数25 MHz, Zephyr RTOS 環境)

処理	S_{data}	T_{total}	T_{trans}	T_{acc}	T_{sw}
退避	128	0.444	0.043	0.250	0.151
	1024	1.753	0.337	1.000	0.416
	8192	13.172	2.693	8.000	2.479
	65536	104.486	21.545	64.000	18.941
復元	128	0.160	0.043	0.000	0.117
	1024	0.834	0.330	0.000	0.504
	8192	6.251	2.623	0.000	3.628
	65536	49.935	20.973	0.000	28.962

表1から明らかな通り、データ量が増大するにつれてソフトウェア起因の遅延 T_{sw} も線形に増加する傾向が確認された。この要因は、Zephyr の SPI サブシステムが標準で備えるスキュッタ・ギャザー対応に伴うバッファ管理のオーバーヘッドにある。具体的には、転送単位が4 Bytes に細分化されていることで、データ量に比例してバッファの境界チェックや進捗ポイントの更新を行う管理関数が極めて高い頻度で反復実行されるため、累積的な管理コストが無視できないオーバーヘッドを生んでいる。一方、ハードウェア遅延 T_{hw} においては、通信帯域に依存するデータ転送時間 T_{trans} と、メモリデバイス特有の物理アクセス時間 T_{acc} が主要な要因となる。このうち T_{trans} は、従来の平面配線によるシリアル通信の帯域制限が物理的なボトルネックとなっており、データ量に対して転送時間が支配的に増加する。また、 T_{acc} は本実証で用いた Flash メモリ内部の物理的な書き込み待ち時間に起因する物理的限界であり、通信クロックの高速化のみでは短縮が困難である。以上の分析に基づき、高速なリカバリを実現するためには、多階層での設計の見直しが不可欠である。まずソフトウェア層では、DMA の導入による介入コストの削減が求められる。次にバス層では、三次元積層による垂直並列バス化を導入することで、2D アーキテクチャの限界である転送遅延 T_{trans} を劇的に短縮する。最後に物理メモリ層では、現状の Flash メモリから、より高速な書き込み特性を持つ MRAM 等の次世代不揮発性メモリへ置換することで、物理アクセス遅延 T_{acc} そのものを根本から削減することが、極めて重要な設計指針となる。本評価結果に基づく、三次元積層 LSI アーキテクチャを想定した広帯域データ転送の性能評価および定量的検証については、次なる課題として今後の研究で詳述する。

6. おわりに

本研究では、ロボットや IoT デバイスの信頼性向上に向け、プロセッサ層と不揮発性メモリ層を垂直に統合した三次元積層 LSI による高速電源断リカバリシステムを提案した。FPGA および Zephyr RTOS を用いた疑似リカバリシステムの実装と実機評価により、従来の平面配線におけるシリアル通信の帯域制限が物理的なボトルネックであることを定量的に明らかにした。

今後は、本稿で得られた実測値および遅延モデルを基に、三次元積層 LSI 環境における広帯域メモリアクセスを想定した詳細な性能評価を実施する。プロセッサと不揮発性メモリの密結合により、ロボットの制御周期を阻害しない高速リカバリの実現に向け、アーキテクチャレベルでの検証を進める予定である。

謝辞 本研究は、内閣府地方大学・地域産業創生交付金「半導体産業の強化及びユーザー産業を含めた新たな産業エコシステムの形成」の助成を受けたものである。また、本研究は JSPS 科研費 25K03095 の助成を受けたものである。

文 献

- [1] B. Ransford, J. Sorber, and K. Fu, "Mementos: System support for long-running computation on RFID-scale devices," ASPLOS, pp. 159–170, 2011.
- [2] B. Lucia and B. Ransford, "A simpler, safer programming and execution model for intermittent systems," ACM SIGPLAN Notices, vol. 50, no. 6, pp. 575–585, 2015.
- [3] J. Van Der Woude and M. Hicks, "Intermittent computation without hardware support or programmer intervention," OSDI 16, 2016.
- [4] D. Balsamo, et al., "Hibernus: Sustaining computation during intermittent supply for energy-harvesting systems," IEEE Embedded Systems Letters, vol. 7, no. 1, pp. 15–18, 2014.
- [5] M. Natsui, K. Sakamoto, and T. Hanyu, "Design of a nonvolatile-register-embedded RISC-V CPU with software-controlled data-retention and hardware-acceleration functions," Memories-Materials, Devices, Circuits and Systems, vol. 4, p. 100035, 2023.
- [6] G. Berthou, et al., "Sytare: A lightweight kernel for NVRAM-based transiently-powered systems," IEEE Transactions on Computers, vol. 68, no. 9, pp. 1390–1403, 2018.
- [7] Y. Wu, et al., "{IntOS}: Persistent embedded operating system and language support for multi-threaded intermittent computing," OSDI 24, 2024.
- [8] F. Kermarrec, et al., "LiteX: an open-source SoC builder and library based on Migen Python DSL," arXiv:2005.02506, 2020.
- [9] SpinalHDL, "VexRiscv: A FPGA friendly 32 bit RISC-V CPU implementation," GitHub repository. [Online]. Available: <https://github.com/SpinalHDL/VexRiscv>
- [10] Zephyr Project, "Zephyr RTOS," GitHub repository. [Online]. Available: <https://github.com/zephyrproject-rtos/zephyr>