Vol.2021-ARC-244 No.1
Vol.2021-SLDM-194 No.1
Vol.2021-EMB-56 No.1
2021/3/25

Linuxが動作可能なRISC-V NVMMエミュレータの実装

大森　侑[1,a)]　木村　啓二[1,b)]

概要：バイトアクセス可能な不揮発性メモリ素子で構成された不揮発性メモリ（NVMM）が注目を集めている．NVMM は従来の主記憶に比べて記憶容量や消費電力に優れ，補助記憶と同じく不揮発でありながら高度なデバイスドライバを経由せずアクセスできる．その反面，レイテンシの大きさやリード性能とライト性能の非対称性などの特性を持つ．NVMM の性能と特性を十分に活かすには，ハードウェア・ソフトウェアの両面からシステムが最適化される必要がある．これに対し筆者等は，複数の NVMM アーキテクチャを柔軟かつ詳細に評価可能な NVMM エミュレータを ARM コアを持つ FPGA 評価ボード上に実装し，OS を含めたシステム全体を実用的な時間で評価できる環境を構築した．本稿では，RISC-V CPU を持ち Linux が動作する RISC-V NVMM エミュレータを提案する．本エミュレータは RISC-V コアを持つオープンソースの SoC をベースのデザインとして採用することにより，CPU コアの改変が可能であり，また，信頼実行環境（Trusted Execution Environment: TEE）である Keystone が利用可能となる．本エミュレータの NVM エミュレーション機構は，ARM ベースのエミュレータで使用していた高速なハード CPU の使用を前提とする手法を改良し，低速なソフト CPU でも局所性等のメモリアクセス特性を反映した評価を可能とする．さらに，ユーザ空間からのキャッシュ制御も可能とする．本エミュレータの Linux 及び Debian OS の動作確認後，Debian 上で SPEC CPU 2017 ベンチマークを用いた評価を行い，提案手法のみが局所性やリード・ライト比の NVMM 向け最適化手法で考慮されるべきアクセス特性を十分に反映できることを確認した．

1. はじめに

不揮発性メインメモリ（NVMM）は，バイトアクセス可能な不揮発性メモリ素子で構成された新たな記憶装置である．NVMM は従来の主記憶と補助記憶の特性を併せ持ち，様々な利点を持つ．主記憶（DRAM）に比べて単位面積あたりの記憶容量に優れ，リフレッシュ操作が不要なため低電力でデータを保持できる．また NVMM は CPU のロード・ストア命令でアクセス可能なメモリであるため，補助記憶のようなデバイスドライバを介さずアクセスでき，データの不揮発性保証に高コストなシステムコールが不要である．しかし，NVMM はこれらのメリットの反面でデメリットも持つ．まず，NVMM は DRAM に比べてアクセスレイテンシが長い．これは書き込みにおいて顕著であり，書き込み処理は電力消費も大きい．また，不揮発性を保証するためには CPU キャッシュのフラッシュとメモリバリアのコストを要する [1].

NVMM の性能と特性を十分に活かすには，ハードウェアのみでなく，OS やランタイム等のソフトウェアを含めた NVMM 向け最適化手法が求められる．NVMM のメモリ素子やアーキテクチャは研究段階であり，最適化手法は様々なアーキテクチャや性能に対して評価される必要がある．NVMM 実機は Intel Optane DC Persistent Memory [2] など極少数で評価環境が限定されるため，シミュレータもしくはエミュレータが評価環境として広く用いられている．シミュレータ [3], [4], [5], [6], [7] はアーキテクチャを詳細かつ柔軟に構成できる反面で評価時間が長く，OS 等を含めた大規模なプログラム評価は困難である．エミュレータは実機と同速度で評価可能であるが，アーキテクチャや実装が固定されるため，評価の柔軟性に欠ける．TUNA v1 [8], Quartz [9], [10], [11] 等のエミュレータは NVMM 向け大規模評価を可能としたが，これらの手法な NVMM のアーキテクチャを反映した評価ができない．TUNA v2 [12] は新規手法を用いて上記問題を解決したが，NVMM 向け最適化手法の指針が不明瞭であった．そこで筆者等は [13] において，TUNA v2 を参考にした NVMM エミュレータを構築し，NVMM を持つシステムにおいて考慮すべきパラメータを明らかにした．

先述のように NVMM はアクセスにデバイスドライバやシステムコールを必要としない．そのため，信頼実行環境（Trusted Execution Environment: TEE) [14] での利用が

[1] 早稲田大学
Waseda University
[a)] oy@kasahara.cs.waseda.ac.jp
[b)] keiji@waseda.jp

NVMM の有望な用途として考えられる．TEE は実行コードとデータを少ない計算資源で保護するための手法であり，IoT エッジなど活用できる．しかし TEE は高度なカーネルを持たないため，従来の補助記憶ではアクセスが大きく制限されてしまう．NVMM は前述の特性から，TEE 内から容易にアクセス可能な代替補助記憶として利用できる可能性がある．既存 TEE として，Intel SGX [15]，ARM TrustZone [16]，Keystone [17] が挙げられる．既存 TEE は NVMM を想定しておらず，NVMM 向けの実装拡張が必要であるため，オープンソースの Keystone を対象とする．Keystone は RISC-V ISA [18], [19] に準拠した RISC-V CPU 及び Linux 上で動作するため，TEE と Keystone を組み合わせるには，RISC-V CPU と統合され，かつ Linux が動作する NVMM エミュレータが必要となる．しかし，既存エミュレータは x86 または ARM に統合されている．また，筆者等が先に開発したエミュレータで用いた手法 [12], [13] は十分に高速な CPU を前提としているため，低速なソフトコアには適用できない．RISC-V ベースの NVMM エミュレータもあるが [20]，これは TUNA v1[8] の実装をベースとしており，NVMM のマイクロアーキテクチャを考慮した評価ができない．

本稿では，FPGA 上に実装された RISC-V CPU を持つ RISC-V NVMM エミュレータを提案する．エミュレータは Freedom SoC U500 Dev Kit [21] をベースとする．本エミュレータは，FPGA ベースの既存エミュレータ [12], [13] の手法を改良し，低速なソフトコアでも NVMM のマイクロアーキテクチャを反映した評価を可能とする．メモリシステムは DRAM と NVMM を持つヘテロジニアスメモリとし，専用モジュールによりメモリバスでのアクセス統計を取得できる．本エミュレータを用いることで，各メモリの使い分けによる評価や，詳細なアクセス統計による最適化も可能である．また，NVMM を扱う際は CPU キャッシュのフラッシュを考慮する必要がある [1]．Freedom SoC の持つ Rocket コア [22] はフラッシュ命令を持つが，ユーザ空間やカーネル空間では実行できない．本エミュレータは既存のフラッシュ命令を修正し，ユーザ空間からもキャッシュフラッシュを可能とした．加えて，Debian RISC-V Ports を導入し，通常の Linux カーネル/Debian OS 上での評価を行った．

本稿の要点は以下の通りである．

- RISC-V NVMM エミュレータの実装．RISC-V CPU を持つ NVMM エミュレータを実装した．提案手法を導入し，低速なソフトコアでも NVMM のアーキテクチャを考慮した評価を可能とした．
- ユーザ空間から実行可能なキャッシュフラッシュ命令の実装．既存実装では不可能だったユーザ空間からのキャッシュ制御を実現した．
- 既存エミュレーション手法と提案手法の比較．マイクロベンチマークを用いて既存手法と提案手法を比較し，提案手法のみが局所性やバンク並列性を反映した評価が可能であることを確認した．
- Debian RISC-V Ports の導入及び評価．Linux 環境が動作することを確認し，Debian 上で SPEC2017 を評価してエミュレータの実装を確認した．

本稿の構成は以下のとおりである．2 章で NVMM 評価環境及び RISC-V 評価環境の関連研究，及びそれらと本研究の差異を述べる．3 章で既存の NVMM エミュレーション手法，及び提案手法の概要と特性を述べる．4 章で本稿で提案するエミュレータの実装について詳述する．5 章では，SPEC 2017 ベンチマークを用いたエミュレーション手法を比較評価し，提案手法の有用性を確認する．最後に，6 章で本稿をまとめる．

2. 関連研究

本稿の関連研究は，NVMM 評価環境，RISC-V 評価環境に大別できる．

2.1 NVMM 評価環境

Gem5，NVMain，PCMSim，HMMSim [3], [4], [5], [6], [7] は NVMM シミュレータに分類される．これらはシステム構成をソフトウェアで模倣し，アーキテクチャやメモリ構成等のパラメータを，詳細かつ柔軟に変化させた評価が可能である．しかし，評価には同構成の実機比で数百 ~ 数万倍の時間を要するため，システム全体のハードウェアや，OS 等の大規模ソフトウェアの評価は困難である．

TUNA [8], [12]，Quartz [9]，及び文献 [10], [11] は NVMM エミュレータに分類される．これらは，リード・ライトそれぞれのメモリアクセスレイテンシを調整することで，DRAM を用いて NVMM の性能をエミュレーションしている．既存エミュレータの内，文献 [9], [10], [11] は実装の制限上，NVMM のマイクロアーキテクチャを反映した評価が不可能だった．TUNA は FPGA 上に構築されたエミュレータであり，TUNA v1 [8] では上記エミュレータと同様の問題があったが，TUNA v2 [12] では新規手法を導入し，NVMM のマイクロアーキテクチャを反映した評価を可能とした．しかし，TUNA v2[12] では新規手法の有効性に関する評価が不十分であった．筆者等は 文献 [13] において TUNA v2[12] の実装を参考に NVMM エミュレータを実装し，新規手法が NVMM のアーキテクチャを反映した評価が可能であることを確認した後，メモリアクセス特性がアプリケーション実行時間に与える影響を明らかにした．

しかし，既存の NVMM エミュレータは RISC-V 環境では利用できない．文献 [9], [10], [11] は特定の Intel プロセッサの機能を使用している．また TUNA v1/v2[8], [12], [13] はハードウェア実装された ARM コアを持つ SoC FPGA 上に構築され，十分に高速な CPU を前提としている．RISC-V

NVMM エミュレータでは，内部動作及びメモリシステムに変更を加えるために FPGA 上で実現されるソフトコアを利用しなければならない．よって，既存のエミュレーション手法では NVMM のアーキテクチャを反映できる RISC-V NVMM エミュレータを実現できない．RISC-V CPU を持つ RISC-V NVMM エミュレータ [20] も存在するが，これは文献 [9], [10], [11] の手法をベースとしており，NVMM のマイクロアーキテクチャを反映した評価ができない．本稿では，RISC-V ソフトコアを持つ RISC-V NVMM エミュレータを実装し，既存手法を改良して低速な CPU でも NVMM のアーキテクチャを反映した評価を可能とする．

2.2 RISC-V 評価環境

RISC-V 製品ボードの代表的なものとして，SiFive 社の HiFive Unmatched [23]，HiFive Unleashed [24]，BeagleBoard.org の Beagle V [25]，Microsemi 社の PolarFire SoC [26] などがある．これらの製品ボードは高速な RISC-V コアと豊富な周辺回路を持ち，容易に評価が可能である．しかし，製品ボードは実装が固定されており，ユースケースに合わせた変更は難しい．本稿で提案する NVMM エミュレーション手法は既存の FPGA を利用したエミュレータ [12], [13] をベースとしており，メモリコントローラの内部動作を変更する必要があるため，RISC-V 製品ボードでは実現できない．

カスタム可能な RISC-V SoC としては，Rocket/BOOM [22]，Freedom SoC [21]，Shakti [27]，Ariane SoC [28] などが挙げられる．これらは RISC-V ソフトコアと周辺回路を持ち，一般利用可能な FPGA 上に実装して動作させることができる．また，これらの多くは Linux を動作可能であることが明言されているが，実際に OS の RISC-V Ports を動作させた例は少ない．本稿では，Keystone の動作要件 [29] や SoC のカスタマイズ性から，Freedom SoC をベースとした RISC-V NVMM エミュレータを実装し，Debian RISC-V Ports を動作させて各種評価を行う．

3. NVMM エミュレーション手法

DRAM を用いて NVMM の性能を模倣するには，アクセスレイテンシを調整する必要がある．本節では，従来手法の概要と特徴，提案手法の概要を述べる．

3.1 従来手法：coarse-grain

coarse-grain は TUNAv1 [8] で提案された手法であり，メモリバスにモジュールを挿入してアクセスレイテンシを調整する．LLC（Last Level Cache）をミスした場合，CPU はメモリコントローラ（MEMC）にメモリリクエストを発行する．この際，CPU と MEMC はまずハンドシェイクを行い，通信を確立した後にリクエストを処理する．AXI4 プロトコルでのハンドシェイクは，CPU の Valid 信号と MEMC の Ready 信号が共にアサートされたクロックで成立する．coarse-grain では，Valid がアサートされるクロックを遅延させてレイテンシを調整する．CPU・MEMC 間にモジュールを挿入し，Valid 信号のアサートを遅延させる．また，ハンドシェイクが意図しないタイミングで成立しないように，Valid 遅延中は MEMC からの Ready 信号をブロックする．

本手法は単純かつ容易に実装できる反面，NVMM のアーキテクチャを考慮した評価ができない．coarse-grain は全てのリクエストに遅延を挿入するが，アクセスレイテンシは一意ではない．DRAM を考えた場合，モジュール内部のロウバッファのヒット有無でレイテンシが変化する．また，NVMM 実機の例として，DCPMM [2] もレイテンシを隠蔽するための内部キャッシュを持つ．内部キャッシュによるレイテンシ変化は，NVMM では DRAM に比べて大きくなるため，その影響は NVMM 向け最適化において重要な特性となる．また DRAM のように複数バンクを持つ場合，リクエスト間で並列処理が可能か否かでレイテンシが変化する．coarse-grain では，局所性及び内部並列性によるレイテンシ変化を反映できず，NVMM 向け最適化手法の評価には不十分である．

3.2 従来手法：fine-grain

fine-grain は TUNA v2 [12] で提案された手法であり，メモリコントローラの内部パラメータを変化させてアクセスレイテンシを調整する．本手法では，DRAM と同様にバンク・ロウの構成で，同様のプロトコルで操作される NVMM を想定する．メモリモジュールは，各バンクにライトバックキャッシュとして機能するロウバッファを持つ．DDR3 プロトコル [30] では，メモリモジュールは主に Activate (ACT)，Read/Write (R/W)，Precharge (PRE) の三コマンドで操作される．

- ACT：ページを開き，データをメモリセルからロウバッファに読み出す
- R/W：ロウバッファに対して読み書きを行う
- PRE：ロウバッファの内容をメモリセルに書き戻し，ページを閉じる
- レイテンシ軽減及びメモリセルの摩耗防止のため，メモリコントローラはロウバッファの状態を管理し，ダーティーな場合のみ PRE で書き戻す（ロウバッファがダーティーな場合のみ *tRP* を調整する）

メモリコントローラは ACT 発行から R/W 発行までに *tRCD*，PRE 発行から ACT 発行までに *tRP* だけ待機しなければならない．上記操作及び fine-grain で用いるタイミングパラメータを**図 1** に示す．

NVMM のレイテンシはメモリセルに依存し，fine-grain では上記の想定より，メモリセルは ACT または PRE でのみアクセスされる．そのため，*tRCD* がリードレイテンシ，*tRP* がライトレイテンシに相当する．fine-grain では，

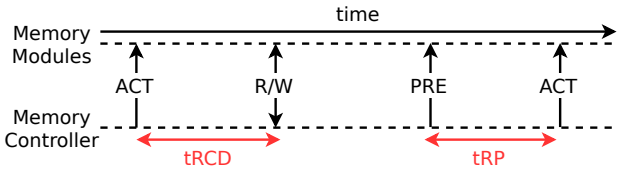

図 1 DDR3 プロトコルでのタイミングパラメータ

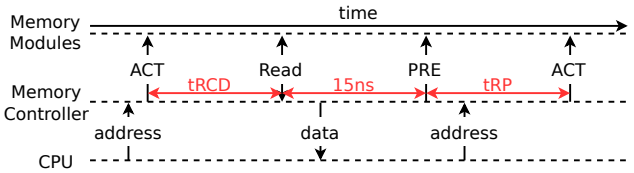

図 2 低速な CPU かつブロッキングキャッシュを持つシステムの動作

tRCD，*tRP* を変化させて NVMM の性能をエミュレートする．ロウバッファヒット時は ACT が省略されるため，アクセス局所性が高い場合はレイテンシが短縮される．また，DDR3 では異なるバンクへのアクセスは並列処理されるが，本手法はこの並列性も反映できる．筆者等は文献 [13] で実装した NVMM エミュレータにおいて，fine-grain は，coarse-grain で無視されるアクセス局所性・バンク並列性を反映した評価が可能であることを確認した．

3.3 提案手法

従来の fine-grain モデルでは，低速な CPU にそのまま適用してもその特性を活かした評価は困難である．本稿で提案する NVMM エミュレータの CPU 動作周波数は 50MHz であり，ブロッキングキャッシュなためメモリリクエストは並列処理できない．メモリコントローラは R/W 完了後，仕様上の待機時間の後，自動で PRE を発行する．CPU が低速かつブロッキングキャッシュである場合，後続リクエストの前に PRE が発行されるため局所性を評価に反映できない．

この様子をリードリクエストを例として**図 2** に示す．アクセス局所性を反映させるためには，メモリコントローラ内で Read が完了してから次のリクエスト処理が 15ns 以内に開始されなければならない．しかし，CPU 動作周波数が 50MHz の場合一クロックは 20ns であり，この制約を満たすことは不可能である．*tRP* 内に処理が開始されたとしても，PRE は発行済であるから再度 ACT が必要となる．よって，従来の fine-grain ではリードに関しては coarse-grain と同様に NVMM のマイクロアーキテクチャが無視される．ただし，ライトに関してはメモリコントローラのライトキューの影響でこの限りではない．

本稿では，従来の fine-grain を拡張し，低速かつブロッキングキャッシュを持つシステムでもアクセス局所性やバンク並列性を反映した評価が可能なエミュレーション手法を導入する．DDR3 プロトコルで使用される詳細なタイミングパラメータを**図 3** に示す．*tRAS* は ACT~PRE の最短

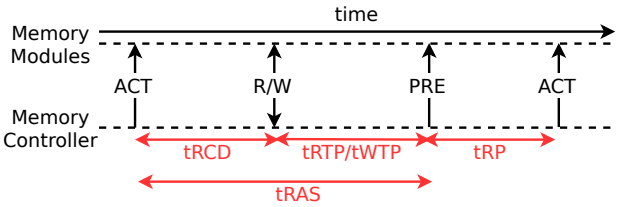

図 3 DDR3 プロトコルでの詳細なタイミングパラメータ

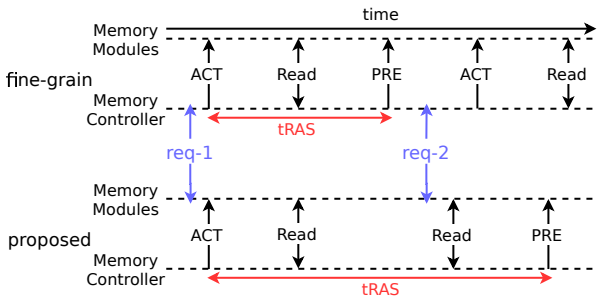

図 4 fine-grain（上部）と提案手法（下部）の動作比較

間隔，*tRTP* は Read~PRE の最短間隔，*tWTP* は Write~PRE の最短間隔である．提案手法では，新たにこれらのタイミングパラメータを調整する．*tRAS* を十分に長くした場合，PRE の発行が遅延されるため，後続リクエストが PRE の前に処理されアクセス局所性が反映されうる．*tRAS* による動作の違いを**図 4** に示す．fine-grain（図 4 上部）では，req-1 終了後，req-2 より前に PRE が発行されるため再度 ACT が必要となる．対して，提案手法で *tRAS* を十分に長くした場合，req-2 では ACT が不要となるためレイテンシが短縮される．次に，*tRAS* が正しく反映されるように，*tRTP*，*tWTP* を調整する．PRE 発行時には，*tRAS*，*tRTP*，*tWTP* の全てが満たされていなければならない．しかし，後者二つは暗黙的に *tRAS* を満たすため，メモリコントローラの実装によっては *tRAS* をチェックしない．よって，*tRTP* と *tWTP* を以下のように再定義する．Read/Write 終了時に，仕様値（spec）と *tRAS* の残り（rest）の最大値を取ることで，仕様と提案手法を満たすことを保証できる．

$$tRTP = max\{tRTP(spec),\ tRAS(rest)\}$$

$$tWTP = max\{tWTP(spec),\ tRAS(rest)\}$$

本手法では，DDR3 のタイミングパラメータの内，*tRCD*，*tRP*，*tRAS*，*tRTP*，*tWTP*（=*tWR*）を調整する．これらの値は DDR3 仕様書 [30] の中で**表 1** の通り定義されている．*tRAS* 以外は最大値の規定が無く，*tRAS* を極端に大きく設定しない限り DDR3 の仕様には違反しない．

4. NVMM エミュレータの実装詳細

本章では，提案エミュレータの概要図，NVMM エミュレーション手法，ユーザ空間からのキャッシュフラッシュ命令の実装，Linux カーネル・OS について順に詳細を述べる．

表 1 DDR3 タイミングパラメータの最小・最大値（DDR3-1600）
表中の “–” は未定義を表す

タイミングパラメータ	最小値 [ns]	最大値 [ns]
tRCD	13.75	–
tRP	13.75	–
tRAS	35	70200
tRTP	7.5	–
tWP	15	–

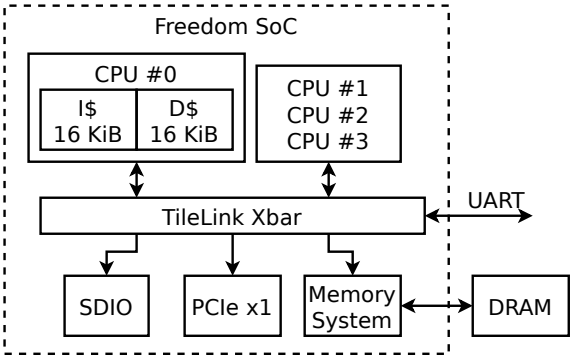

図 5 提案する NVMM エミュレータのブロック図（概要）

表 2 RISC-V NVMM エミュレータの各種仕様

FPGA	Xilinx Virtex-7 FPGA VC707
Device	Virtex-7 XC7VX485T-2FFG1761
CPU Core	Rocket-Core x4
RISC-V ISA	RV64IMAFDC
RISC-V Spec.	Unpriv. 2.1 / Priv. 1.11
L1 Cache	I=16 KiB/core, D=16 KiB/Core
DRAM	4 GiB, DDR3-1600, SO-DIMM
SoC 周波数	50 MHz
メモリシステム周波数	200 MHz
OS	GNU/Linux riscv64 5.6.0-dirty Debian GNU/Linux bullseye/sid
周辺回路	SDIO，Ethernet over PCIe JTAG，UART

4.1 評価環境の概要

本稿で提案する NVMM エミュレータは，SiFive 社が公開する Freedom U500 VC707 FPGA Dev Kit [21] を拡張して実装されている．エミュレータ全体のブロック図（概要）を**図 5** に示し，各種仕様を**表 2** に示す．RISC-V CPU として UC Berkeley の Rocket コア [22] を持つ．Rocket コアは RV64IMAFDC ISA に準拠し，single-issue，5-stage pipeline，in-order の CPU である．Keystone を実行可能な NVMM エミュレータを構築するため，Keystone 動作要件 [29] を満たしカスタム可能な Freedom SoC を採用した．

4.2 NVMM エミュレーション

Freedom SoC は，Xilinx 社の IP である MIG (Memory Interface Generator) を使用して DRAM を制御する．Rocket コア及び周辺回路は 50MHz，MIG は 200MHz で動作する．CPU が低速なため，3.2 節の fine-grain では局所性等を反映した評価ができず，3.3 節の手法を使用する必要がある．従来手法と提案手法を比較評価するため，coarse-grain，fine-grain 及び提案手法を実装し動的に切り替え可能としている．メモリシステム周辺のブロック図を**図 6** に示す．coarse-grain は，MIG の直前にハンドシェイクを遅延させるモジュールを挿入して実装する．fine-grain 及び提案手法は，MIG の RTL 記述を変更し，内部動作を変更して実装する．各手法の遅延サイクル数や DRAM の論理分割境界は，別にメモリマッピングされたコンフィギュレーションレジスタを介して設定できる．

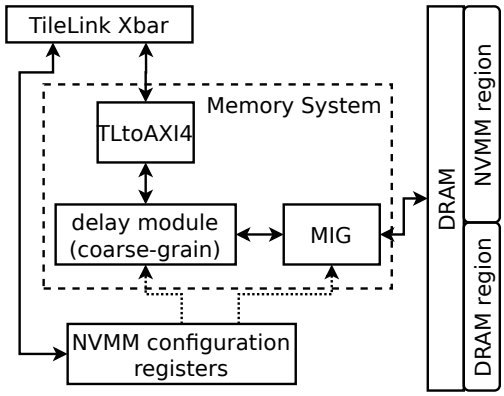

図 6 メモリシステム周辺のブロック図

VC707 FPGA はメモリスロットを一つしか持たず，DRAM と NVMM を別々のメモリに割り当てることができない．よって本稿で提案するエミュレータでは，一つのメモリを論理的に分割してヘテロジニアスメモリを実現する．遅延は NVMM 部分へのアクセスにのみ挿入される．また，DRAM と NVMM を明示的に使い分けるためには，NVMM を System RAM から除外してカーネル管理領域外として扱い，専用の API 経由で確保・解放する必要がある．本稿では，NVMM として扱うメモリ空間をデバイスツリーから除外し，NVMM 向けに改変されたアロケータで実現する．ARM はページテーブルで System RAM 外のメモリを uncacheable として扱うため，NVMM 領域を主・補助記憶の代替として評価するには文献 [13] で述べられるようなカーネル改変が必要だった．しかし，RISC-V ISA はキャッシャビリティをハードウェア生成時に固定する．物理アドレス空間がメモリ領域ならば System RAM 外であっても cacheable になるため，RISC-V 環境ではカーネル改変は必要ない．

4.3 キャッシュフラッシュ

RISC-V ISA はキャッシュ制御命令を定義していないため，フラッシュを行うにはカスタム命令を実装する必要がある．SiFive 社の商用コア U54 [31] などは L1 データキャッシュをフラッシュする CFLUSH.D.L1 命令を持ち，その実装は Rocket コアに移植されている．本稿で採用した Freedom SoC は Rocket コアを使用しているため CFLUSH.D.L1 命令

Vol.2021-ARC-244 No.1
Vol.2021-SLDM-194 No.1
Vol.2021-EMB-56 No.1
2021/3/25

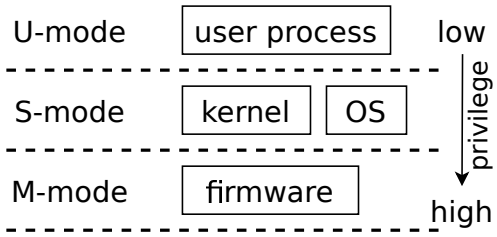

図 7 RISC-V の権限スタック

を実行可能だが，ユーザ空間からは実行できない．RISC-V の権限スタックを**図 7** に示す．CFLUSH.D.L1 は M-mode（図 7 最下部）でのみ実行可能で，ユーザ空間から実行するためには特殊な API 等が必要となり実行オーバヘッドが大きくなる．

本稿では，CFLUSH.D.L1 の実行権限に関する実装を修正し，ユーザ空間（U-Mode）やカーネル空間（S-Mode）での実行を可能とした．本修正による従来機能への影響は無い．CFLUSH.D.L1 はインラインアセンブラの.insn 命令で実行する．reg がゼロならキャッシュ全体をフラッシュし，非ゼロなら reg から仮想アドレスを読み該当ラインをフラッシュする．

```
(".insn i 0x73, 0, x0, %0, -0x340 :: "r"(reg));
```

4.4 Linux カーネル及び OS

本エミュレータでは，keystone リポジトリ [32] の e448fa3 からビルドした Linux カーネルを使用する．Linux 5.6.0 をベースとし，Keystone 用パッチ以外の改変は入っていない．

OS は Debian の RISC-V Ports [33] を使用する．debootstrap で rootfs を作成し，上記でビルドした Linux カーネルの起動パラメータを修正し rootfs を使用して起動するようにした．Debian のバージョンは bullseye/sid（unstable Debian 11）となる．

5. NVMM エミュレータの実装評価・検証

本章では，マイクロベンチマーク及び SPEC CPU 2017 ベンチマーク [34] を使用してエミュレータの実装評価・検証を行う．本章の全ての評価において，coarse-grain はリード・ライト共に 1000ns，fine-grain は $tRCD = tRP = 1000$ns，提案手法では fine-grain に加えて *tRAS* を 7000ns に設定した．

5.1 マイクロベンチマークによるアクセス局所性評価

図 9 に示すマイクロベンチマークを用いて，coarse-grain，fine-grain，提案手法を比較評価し，提案手法のみがアクセス局所性を十分に反映した評価が可能であることを確認する．図中の *STRIDE* を 4096 と 8192 に設定し，それぞれの平均アクセスレイテンシを測定する．ロウバッファのサイズは 8192 であるため，4096 の場合は複数リクエストがロウバッファにヒットしてレイテンシが隠蔽される状況が生じる．

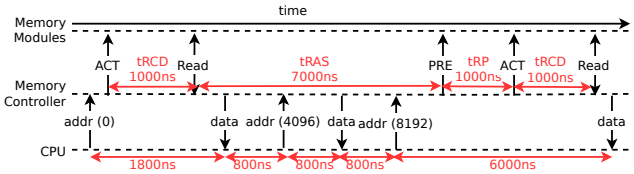

図 8 提案手法適用時の実動作とレイテンシ

測定した平均リードレイテンシを**表 3** に示す．表 3 より，coarse-grain と fine-grain は STRIDE に依存せず，提案手法では STRIDE の変化によってレイテンシが大きく隠蔽されていることが分かる．また，coarse-grain と fine-grain はほぼ同じ値を示し，手法間の差異が無いことも分かる．提案手法での理想的動作では，STRIDE が 4096 の場合はリクエストの 1/2 はロウバッファにヒットし，レイテンシは 1/2 になる．表 3 では 0.67 倍にとどまっているが，これは妥当な計測結果である．

提案手法適用時の実動作とレイテンシを**図 8** に示す．図中のレイテンシは簡略化しており，実際のレイテンシとは多少異なる．あるアドレスにアクセスした後，そこから 4096 バイト増加したアドレス addr(4096) にアクセスすると，ロウバッファにヒットして ACT が省略されレイテンシは約 1600ns となる（addr(4096)→data）．続いてさらに 4096 バイト増加したアドレス addr(8192) にアクセスすると，ロウバッファミスが発生するため，PRE と ACT が必要となる．この時，*tRAS* を満たす必要があるため待機時間が生じ，レイテンシは約 6000ns となる（addr(8192)→data）．STRIDE が 4096 の場合はヒットとミスが交互に発生するため，平均レイテンシは約 $(1600 + 6000)/2 = 3800$ns となり，表 3 にほぼ一致する．

次に，測定した平均ライトレイテンシを**表 4** に示す．表 4 では手法間に顕著な差は見られないが，これは妥当な測定結果である．CPU がライトバックキャッシュを持つ時，ライトはまずキャッシュで処理され，ライン置換時に初めてメモリに書き戻される．この時，ライト局所性の多くはキャッシュに吸収され，メモリバスでは局所性が低くなる．実測値として，提案手法使用時の STRIDE ＝ 4096 の場合でもロウバッファヒット率は 5%未満である．表 4 では fine-grain 使用時もレイテンシが隠蔽されているが，これは MIG 内部のライトキューによってリクエスト処理間隔が軽減されるためである．また，fine-grain が coarse-grain に比べてレイテンシが大きいのは，レイテンシ挿入動作の違いに起因する．キャッシュラインが書き戻される時，coarse-grain ではライトレイテンシ 1000ns のみが挿入されるが，fine-grain では ACT 及び PRE が必要なため合計で 2000ns が挿入されるためレイテンシが大きくなる．

本節の評価から，低速な CPU を持つシステムにおいて，従来手法ではアクセス局所性を反映した評価は困難である

```
SIZE := size of NVMM region
p    := head address of NVMM region
start = clock();
for (off = 0; off < SIZE ; off += STRIDE) {
    read from or write into (p+off) ;
}
end = clock();
ave = (end - start)/(num. of iterations);
```

図 9 アクセス局所性評価用マイクロベンチマーク

表 3 *STRIDE* を変化させた際の平均リードレイテンシ
(×N) は 8192 比でレイテンシが N 倍であることを示す

STRIDE	平均レイテンシ [ns]		
	coarse-grain	fine-grain	提案手法
4096	2702 (×0.94)	2708 (×0.94)	4081 (×0.67)
8192	2881	2872	6064

表 4 *STRIDE* を変化させた際の平均ライトレイテンシ
(×N) は 8192 比でレイテンシが N 倍であることを示す

STRIDE	平均レイテンシ [ns]		
	coarse-grain	fine-grain	提案手法
4096	4399 (×0.98)	4879 (×0.91)	14457 (×0.91)
8192	4479	5334	15913

が，提案手法ではこれを十分に反映した評価が可能であることが分かる．純粋なライトベンチマークでは一部の局所性のみ反映可能であるが，実アプリケーションはリード処理を多く含むため局所性を反映させることは可能である．

5.2 マイクロベンチマークによるバンク並列性評価

図 10 に示すマイクロベンチマークを用いて, coarse-grain, fine-grain，提案手法を比較評価し，提案手法のみがバンク並列性を十分に反映した評価が可能であることを確認する．図中の *NBANK* を 1 から 6 まで変化させて，平均アクセスレイテンシを測定する．バンク並列性を反映できるならば，*NBANK* の増加に伴ってレイテンシが隠蔽される．

測定したリードレイテンシを**表 5**，ライトレイテンシを**表 6** に示す．表 5 では，提案手法のみが *NBANK* の増加に伴って最大 55%レイテンシを隠蔽できることが分かる．表 6 では，提案手法が *NBANK* の増加に伴って最大 67%レイテンシを隠蔽できることが分かる．また，表 6 では fine-grain でもレイテンシが隠蔽されているが，これは先述の通りライトキューによってリクエスト処理感覚が軽減されるためである．fine-grain はライトに限定すればバンク並列性をある程度反映できるが，提案手法より早く一定値に達するため効果が小さい．

本節の評価から，低速な CPU を持つシステムにおいて，従来手法ではバンク並列性を反映した評価は困難であるが，提案手法ではバンク並列性を十分に反映した評価が可能であることが分かる．また，表 5 と表 6 において，*NBANK*

```
SIZE   := size of NVMM region
SZBANK := size of BANK
p      := head address of NVMM region
start = clock();
for (off = 0; off < SIZE ; off += SZBANK) {
  for (bank = 0; bank < NBANK; bank++) {
    read from or write into (p+bank*SZBANK+off);
  }
}
end = clock();
ave = (end - start)/(num. of iterations);
```

図 10 バンク並列性評価用マイクロベンチマーク

表 5 *NBANK* を変化させた際の平均リードレイテンシ

NBANK	平均レイテンシ [ns]		
	coarse-grain	fine-grain	提案手法
1	2598	2608	6060
2	2626	2634	4353
3	2637	2647	3062
4	2753	2736	2751
5	2887	2875	2839
6	2962	2966	2944

表 6 *NBANK* を変化させた際の平均ライトレイテンシ

NBANK	平均レイテンシ [ns]		
	coarse-grain	fine-grain	提案手法
1	4204	4892	15843
2	4207	3925	11104
3	4223	3655	8712
4	4307	3652	7070
5	4465	3658	5818
6	4507	3656	5153

の増加に伴うレイテンシ増加が見られる．この傾向は物理アドレスで動作するベアメタル環境では見られないため論物変換に伴うオーバヘッドと推測されるが，本稿で使用している Rocket コアは TLB ミス等のカウンタを持たないため，推測にとどまる．

5.3 SPEC CPU 2017 ベンチマークによる評価

本節では SPEC CPU 2017 ベンチマーク [34] を用いて，coarse-grain，fine-grain，提案手法を比較評価し，提案手法の有効性を確認する．SPEC CPU 2017 に含まれる 24 種の中から，RISC-V 環境で正常にコンパイル・実行可能で，動的メモリ確保を使用している 14 種を評価する．評価では，全ての動的確保（malloc，calloc，realloc，free）を，jemalloc [35] で上書きして行う．jemalloc は NVMM から領域確保するよう改変し，実行時に LD_PRELOAD で指定する．

評価結果を**図 11** に示す．図中の棒グラフは左から coarse-grain，fine-grain，提案手法の通常 DRAM 使用による実行時間に対する正規化実行時間（左縦軸）を示し，折れ線グ

ラフは提案手法使用時のメモリリクエストの頻度（右縦軸）を示す．横軸はプログラム名であり，正規化実行時間（提案手法）の昇順に並べられている．まず図 11 から，プログラムによってレイテンシの影響が大きく異なることが分かる．図中では，正規化実行時間（提案手法）とメモリリクエスト頻度の上昇順は傾向として一致するが，例外が存在する（図中の丸四角あるいは角四角で囲まれたベンチマーク）．本現象の詳細評価のため，アクセス局所性の低さを表す *ACT/REQ*，バンク並列性の高さを表す *BANK_PARA* を計測した．*ACT/REQ* は，メモリコントローラが NVMM 領域に対して発行した ACT 回数を，メモリバスで処理された NVMM 領域へのリクエスト回数で割った値で，低いほどアクセス局所性が高い．*BANK_PARA* は，メモリバスで処理された NVMM 領域へのリクエストの内，直前のリクエストと異なるバンクにアクセスした割合で，高いほどバンク並列性が高い．これらの，提案手法使用時の測定結果を**表 7** に示す．

図 11 の中で丸四角で示したベンチマークは，特に大きな例外を示している．511.povray_r，523.xalancbmk_r は 549.fotonik3d_r よりメモリアクセス頻度が高いが，正規化実行時間が小さい．同様の傾向が 505.mcf_r と 503.bwaves_r の間にも見られる．これらのベンチマークについて，正規化実行時間とメモリアクセス頻度, *ACT/REQ* を**表 8** に示す．表 8 より，511.povray_r，523.xalancbmk_r，549.fotonik3d_r は，極めて低い *ACT/REQ* を持ち，アクセス局所性が高いベンチマークであることが分かる．しかし，coarse-grain と fine-grain は *ACT/REQ* が 1.00 でありベンチマークのアクセス特性を評価に反映できていない．アクセス頻度に着目すると，従来の二手法では正規化実行時間とメモリアクセス頻度昇順が一致している．上記傾向は，505.mcf_r と 503.bwaves_r の間にも見られる．よって，図 11 中の丸四角で示される例外は，提案手法のみが反映できるアクセス局所性に起因することが分かる．これらのベンチマーク及び評価結果は，アプリケーションが高い局所性を持つ時，低速な CPU を持つシステムでは従来手法はそれを評価に反映できず，提案手法のみが反映できることを示す．

また, 図 11 中の角四角で示される, 525.x264_r と 531.deepsjeng_r，503.bwaves_r と 557.xz_r，508.namd_r と 519.lbm_r の間に小さな例外が見られる．これはアプリケーションのリード・ライト比に起因する例外である．メモリバスで，NVMM 領域に発行されたリードリクエスト数を，同領域に発行されたライトリクエスト数で割るとリード・ライト比が得られる．表 3 と表 4，表 5 と表 6 の比較から，提案手法では，リードレイテンシがライトレイテンシより 2.5~3.0 倍程度小さい．よって，メモリリクエスト数がほぼ同じ場合，リード・ライト比が大きいベンチマークの方がレイテンシの影響が小さい．提案手法適用時のメモリリクエスト頻度とリード・ライト比を**表 9** に示す．表 9 より，例外を示すベンチマークはリード・ライト比が大きいことが分かる．coarse-grain と fine-grain ではこれらの例外は見られないため，アプリケーションのリード・ライト比は，提案手法のみが評価に反映できる．これらのベンチマーク及び評価結果は，アプリケーションが大きなリード・ライト比を持つ時，低速な CPU を持つシステムでは従来手法はそれを評価に反映できず，提案手法のみが反映できることを示す．

表 7 提案手法使用時の *ACT/REQ* 及び *BANK_PARA*

ベンチマーク	*ACT/REQ*	*BANK_PARA*
525.x264_r	0.88	0.00
531.deepsjeng_r	0.83	0.10
541.leela_r	0.85	0.00
511.povray_r	0.48	0.00
523.xalancbmk_r	0.50	0.00
549.fotonik3d_r	0.93	0.26
505.mcf_r	0.52	0.02
503.bwaves_r	0.97	0.12
557.xz_r	0.93	0.01
520.omnetpp_r	0.90	0.00
507.cactuBSSN_r	0.88	0.00
554.roms_r	0.97	0.00
508.namd_r	0.87	0.00
519.lbm_r	0.96	0.00

本節の SPEC CPU 2017 を用いた評価より，アプリケーションが高い局所性やリード・ライト比を持つ時，提案手法のみがそれらを十分に評価に反映できることが分かる．これらの特性は，レイテンシが大きく，特にライトにおいて顕著である NVMM 向け最適化手法において重要な特性である．よって本節を通して，本稿で提案する新規 NVMM エミュレーション手法の，NVMM 向け最適化手法の評価における有効性が確かめられた．

6. まとめ

本稿では，Freedom U500 VC707 FPGA Dev Kit をベースとして，RISC-V CPU を持つ RISC-V NVMM エミュレータを FPGA 上に実装した．本エミュレータは従来手法を拡張した NVMM エミュレーション手法を導入し，従来手法では困難であった低速な CPU を持つシステムでのメモリアクセス特性を反映した NVMM 向け最適化手法の評価を可能とした．また，既存実装を修正し，ユーザ空間で不可能だったキャッシュ制御命令をユーザ空間で実行可能とし，ユーザプロセス実行中に低オーバヘッドかつ容易にキャッシュをフラッシュできるようにした．加えて，エミュレータ上での Linux カーネル及び Debian OS の動作を確認し，Debian 上でベンチマークプログラムを用いた性能評価を行った．

従来手法の coarse-grain，fine-grain 及び提案手法をそれぞれ，マイクロベンチマーク及び SPEC CPU 2017 ベンチマークを用いて評価した．マイクロベンチマークを用いた

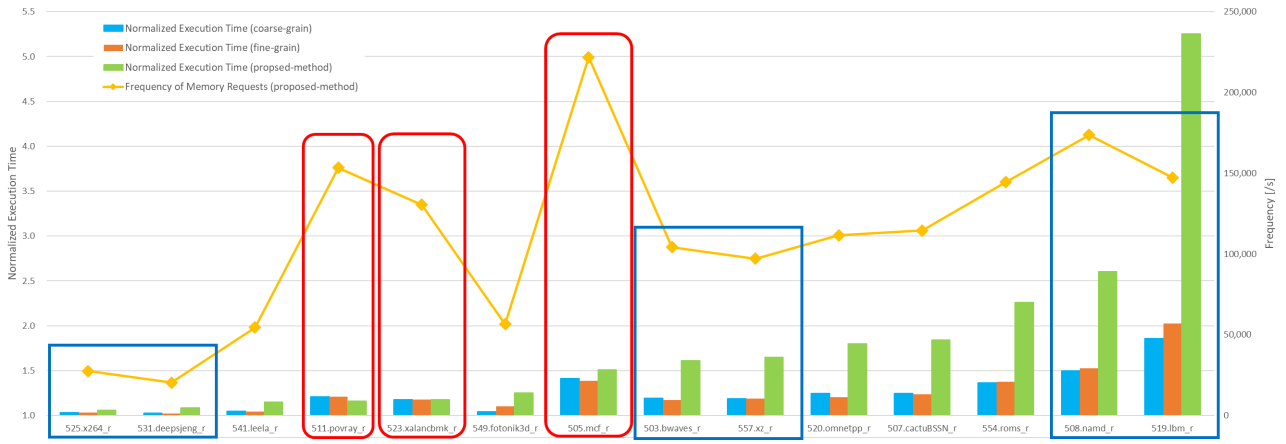

図 11 SPEC CPU 2017 評価結果

表 8 例外を示すベンチマークの詳細な測定結果

ベンチマーク	正規化実行時間			メモリアクセス頻度 [/s]			ACT/REQ		
	coarse-grain	fine-grain	提案手法	coarse-grain	fine-grain	提案手法	coarse-grain	fine-grain	提案手法
511.povray_r	1.21	1.21	1.16	147,073	147,629	153,479	1.00	1.00	0.48
523.xalancbmk_r	1.18	1.18	1.18	130,309	130,522	130,496	1.00	1.00	0.50
549.fotonik3d_r	1.04	1.10	1.25	67,918	64,256	56,604	1.00	0.99	0.93
505.mcf_r	1.42	1.39	1.51	237,336	235,658	221,879	1.00	1.00	0.52
503.bwaves_r	1.19	1.17	1.61	144,000	140,555	104,226	1.00	1.00	0.97

表 9 提案手法適用時のリクエスト頻度とリード・ライト比

ベンチマーク	リクエスト頻度 [/s]	リード・ライト比
525.x264_r	27,490	13.22
531.deepsjeng_r	20,391	1.21
503.bwaves_r	104,226	5.44
557.xz_r	97,026	1.77
508.namd_r	150,293	5.40
519.lbm_r	140,965	1.21

評価では，従来手法ではアクセス局所性とバンク並列性を反映した評価が困難であるが，提案手法ではこれらを十分に反映した評価が可能であることを確認した．SPEC CPU 2017 ベンチマークを用いた評価では，提案手法のみがアプリケーションの持つ特性を反映した評価が可能であることが分かった．従来手法を用いた場合，NVMM 上でのアプリケーション実行時間はメモリアクセス頻度にのみ依存する．しかし，提案手法ではそれに加えてアクセス局所性とリード・ライト比の NVMM 向け最適化手法で重要な特性を反映した評価が可能であることが確認できた．

本稿で提案するエミュレータにより，RISC-V 環境において，低速なソフトコア使用時でもメモリアクセス特性を十分に考慮した NVMM 向け最適化手法の評価が可能となった．本エミュレータを用いることで，TEE（Keystone）と NVMM を両立できる評価環境が実現された．NVMM 向け最適化手法で考慮すべき，キャッシュフラッシュオーバヘッドの評価は今後の課題となる．

謝辞 本研究の一部はキオクシア株式会社と早稲田大学との組織連携活動の一貫として実施した．

参考文献

[1] Pelley, S., Chen, P. M. and Wenisch, T. F.: Memory Persistency, *Proceeding of the 41st Annual International Symposium on Computer Architecuture*, ISCA '14, Piscataway, NJ, USA, IEEE Press, pp. 265–276 (online), available from ⟨http://dl.acm.org/citation.cfm?id=2665671.2665712⟩ (2014).

[2] Looi, L. and Xu, J. J.: INTEL® OPTANE™ DATA CENTER PERSISTENT MEMORY, *Hot Chips (HC) 31* (2019).

[3] Binkert, N., Beckmann, B., Black, G., Reinhardt, S. K., Saidi, A., Basu, A., Hestness, J., Hower, D. R., Krishna, T., Sardashti, S., Sen, R., Sewell, K., Shoaib, M., Vaish, N., Hill, M. D. and Wood, D. A.: The Gem5 Simulator, *SIGARCH Comput. Archit. News*, Vol. 39, No. 2, pp. 1–7 (online), DOI: 10.1145/2024716.2024718 (2011).

[4] Poremba, M. and Xie, Y.: NVMain: An Architectural-Level Main Memory Simulator for Emerging Non-volatile Memories, *2012 IEEE Computer Society Annual Symposium on VLSI*, pp. 392–397 (online), DOI: 10.1109/ISVLSI.2012.82 (2012).

[5] Poremba, M., Zhang, T. and Xie, Y.: NVMain 2.0: A User-Friendly Memory Simulator to Model (Non-)Volatile Memory Systems, *IEEE Computer Architecture Letters*, Vol. 14, No. 2, pp. 140–143 (online), DOI: 10.1109/LCA.2015.2402435 (2015).

[6] Wang, J. and Wang, B.: PCMSim: A Hybrid Memory System Simulator for the Cloud Storage, *2017 Fifth International Conference on Advanced Cloud and Big Data (CBD)*, pp. 81–86 (online), DOI: 10.1109/CBD.2017.22 (2017).

[7] Bock, S., Childers, B. R., Melhem, R. and Mosse, D.: HMMSim: a simulator for hardware-software co-design of hybrid main memory, *2015 IEEE Non-Volatile Memory System and Applications Symposium (NVMSA)*, pp. 1–6 (online), DOI: 10.1109/NVMSA.2015.7304374 (2015).

[8] Lee, T., Kim, D., Park, H., Yoo, S. and Lee, S.: FPGA-based prototyping systems for emerging memory tech-

nologies, *2014 25nd IEEE International Symposium on Rapid System Prototyping*, pp. 115–120 (online), DOI: 10.1109/RSP.2014.6966901 (2014).

[9] Volos, H., Magalhaes, G., Cherkasova, L. and Li, J.: Quartz: A Lightweight Performance Emulator for Persistent Memory Software, *Proceedings of the 16th Annual Middleware Conference*, Middleware '15, New York, NY, USA, ACM, pp. 37–49 (online), DOI: 10.1145/2814576.2814806 (2015).

[10] Koshiba, A., Hirofuchi, T., Akiyama, S., Takano, R. and Namiki, M.: Towards write-back aware software emulator for non-volatile memory, *2017 IEEE 6th Non-Volatile Memory Systems and Applications Symposium (NVMSA)*, pp. 1–6 (online), DOI: 10.1109/NVMSA.2017.8064479 (2017).

[11] Koshiba, A., Hirofuchi, T., Takano, R. and Namiki, M.: A Software-based NVM Emulator Supporting Read/Write Asymmetric Latencies, *IEICE Transactions*, Vol. 102-D, No. 12, pp. 2377–2388 (online), available from ⟨http://search.ieice.org/bin/summary.php?id=e102-d_12_2377⟩ (2019).

[12] Lee, T. and Yoo, S.: An FPGA-based platform for non volatile memory emulation, *2017 IEEE 6th Non-Volatile Memory Systems and Applications Symposium (NVMSA)*, pp. 1–4 (online), DOI: 10.1109/NVMSA.2017.8064466 (2017).

[13] Omori, Y. and Kimura, K.: Performance Evaluation on NVMM Emulator Employing Fine-Grain Delay Injection, *2019 IEEE Non-Volatile Memory Systems and Applications Symposium (NVMSA)*, pp. 1–6 (online), DOI: 10.1109/NVMSA.2019.8863522 (2019).

[14] Sabt, M., Achemlal, M. and Bouabdallah, A.: Trusted Execution Environment: What It is, and What It is Not, *2015 IEEE Trustcom/BigDataSE/ISPA*, Vol. 1, pp. 57–64 (online), DOI: 10.1109/Trustcom.2015.357 (2015).

[15] Gueron, S.: A Memory Encryption Engine Suitable for General Purpose Processors, , available from ⟨https://eprint.iacr.org/2016/204⟩ (accessed 2021-02-05).

[16] Limited, A.: ARM Security Technology Building a Secure System using TrustZone ® Technology, , available from ⟨https://developer.arm.com/documentation/genc009492/latest/⟩ (accessed 2021-02-05).

[17] Lee, D., Kohlbrenner, D., Shinde, S., Song, D. and Asanović, K.: Keystone: An Open Framework for Architecting TEEs, (online), available from ⟨http://arxiv.org/abs/1907.10119⟩ (2019).

[18] Waterman, A. and Asanović, K.: The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Document Version 20190608-Priv-MSU-Ratified, Technical report, RISC-V Foundation (2019).

[19] Waterman, A. and Asanović, K.: The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Document Version 20191213, Technical report, RISC-V Foundation (2019).

[20] Ruoheng, M.: An RISCV Emulation Board for Non-Volatile Memory (2020). Graduation Thesis at Fakultät für Informatik, Karlsruher Institut für Technologie.

[21] SiFive: Freedom SoC, SiFive Inc. (online), available from ⟨https://github.com/sifive/freedom⟩ (accessed 2021-02-05).

[22] Asanović, K., Avizienis, R., Bachrach, J., Beamer, S., Biancolin, D., Celio, C., Cook, H., Dabbelt, D., Hauser, J., Izraelevitz, A., Karandikar, S., Keller, B., Kim, D., Koenig, J., Lee, Y., Love, E., Maas, M., Magyar, A., Mao, H., Moreto, M., Ou, A., Patterson, D. A., Richards, B., Schmidt, C., Twigg, S., Vo, H. and Waterman, A.: The Rocket Chip Generator, Technical Report UCB/EECS-2016-17, EECS Department, University of California, Berkeley (2016).

[23] SiFive: HiFive Unmatched, , available from ⟨https://www.sifive.com/boards/hifive-unmatched⟩ (accessed 2021-02-05).

[24] SiFive: HiFive Unleashed, , available from ⟨https://www.sifive.com/boards/hifive-unleashed⟩ (accessed 2021-02-05).

[25] beagleboard.org: Beagle V, , available from ⟨https://beagleboard.org/beaglev⟩ (accessed 2021-02-05).

[26] Microsemi: PolarFire SoC FPGA Icicle Kit, , available from ⟨https://www.microsemi.com/existing-parts/parts/152514⟩ (accessed 2021-02-05).

[27] shakti.org: Shakti Processor, , available from ⟨https://shakti.org.in/⟩ (accessed 2021-02-05).

[28] Zaruba, F. and Benini, L.: The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 27, No. 11, pp. 2629–2640 (online), DOI: 10.1109/TVLSI.2019.2926114 (2019).

[29] Lee, D. and Kohlbrenner, D.: Keystone Enclave / RISC-V Background, , available from ⟨http://docs.keystone-enclave.org/en/latest/Getting-Started/How-Keystone-Works/RISC-V-Background.html⟩ (accessed 2021-02-05).

[30] ASSOCIATION, J. S. S. T.: DDR3 SDRAM Standard (Revision of JESD79-3E, July 2010), Technical report, JEDEC.

[31] SiFive: SiFive U54 Standard Core, , available from ⟨https://www.sifive.com/cores/u54⟩ (accessed 2021-02-05).

[32] Enclave, K.: Keystone Enclave (QEMU + HiFive Unleashed), , available from ⟨https://github.com/keystone-enclave/keystone⟩ (accessed 2021-02-05).

[33] snapshot.debian.org: debian-ports, , available from ⟨https://snapshot.debian.org/archive/debian-ports/⟩ (accessed 2021-02-05).

[34] spec.org: SPEC CPU(R) 2017, Standard Performance Evaluation Corporation (online), available from ⟨https://www.spec.org/cpu2017/⟩ (accessed 2021-02-05).

[35] jemalloc.net: jemalloc, , available from ⟨https://github.com/jemalloc/jemalloc⟩ (accessed 2021-02-05).