RISC-V 版 FabScalar 用機能シミュレータへの同期例外及び割り込み処理機構の実装と性能評価

愛知県立大学 村井 公哉　　指導教員：佐々木 敬泰

1 はじめに

現在のプロセッサは年々複雑化と大規模化をしており，故にプロセッサの自動設計ツールの需要が高まっている．プロセッサの自動設計ツールセットとして FabScalar[1] が提案されており，スーパースカラプロセッサの自動設計が可能である．また，FabScalar には自動設計したプロセッサの動作及び性能を効率的に検証するための FabScalar 用機能シミュレータが含まれている．特徴として挙げられるのは，論理レベルでのシミュレーションを行う HDL シミュレータとの協調動作を，命令フェッチ，実行などプロセッサの動作レベルで可能という点である．

FabScalar 用機能シミュレータは大きく分けて MIPS(MIPS32R2) 版と RISC-V 版が存在する．前者は，OS を動作させるための FS(フルシステムシミュレーション) をサポートしているが，対応している命令セットが比較的古い．そのため，最近注目されている RISC-V という命令セットに対応すべく，後者の開発が進められており，FS を行うためには OS の動作に必要な例外処理機構とページングなどのメモリ管理機構を実装する必要がある．しかし，例外処理機構の実装は既存の RISC-V シミュレータの実装手法では，命令フェッチ，実行などのプロセッサの動作レベルでの実装とはなっておらず，その手法を用いたとしても協調動作が不可能という問題がある．

そこで本研究では，RISC-V 版 FabScalar 用機能シミュレータに協調動作可能な例外処理機構の実装手法を提案し，動作検証および性能評価を行う．

2 例外処理機構の実装手法

2.1 RISC-V の例外処理概要

RISC-V 公式の仕様 [2] によれば，例外はそれぞれ同期例外及び割り込みの 2 つに分類される．前者はメモリアクセス違反や命令違反，後者はタイマー割り込み及びソフトウェア割り込み，外部割込みを指す．また RISC-V は特権命令，特権レジスタ，特権モードが定義されており，例外発生時は特権レジスタに各種設定が行われる．同期例外発生時は特権レジスタ及び特権モードの設定を行った後，例外ハンドラへ処理を移行するが，割り込み発生時は加えてタイマー割り込み及びソフトウェア割り込みの設定や待ち状態を管理する CLINT[3] 及び，外部割り込みを適切にルーティングする PLIC[4] が利用される．

2.2 提案手法の実装及び動作検証

RISC-V の例外処理において必要となる機構の仕様及び MIPS 版機能シミュレータの実装について調査を行った．その結果，既存の RISC-V シミュレータの実装手法では例外処理機構について協調動作ができないという問題があることが分かった．この問題を解決するためには，例外捕捉については命令フェッチ，実行のレベルで行うという MIPS 版機能シミュレータと同様の手法をとればよいこと，特権レジスタへの設定については MIPS と異なり同じ手法を用いることができないため，独自に実装する必要がある事が分かった．そこで本研究では，例外捕捉を命令フェッチ，実行のレベルで行い，捕捉した例外の情報を基に特権レジスタの設定を行うという手法を提案する．動作検証を行った結果，特権レジスタへの設定が正しく行われ，意図したとおりに例外ハンドラへ処理を移行させることに成功した．

表 1　動作内容と平均実行実時間

動作内容	平均実実行時間 (ms)
NOP 命令 256 回	4.0
同期例外 256 回	6.2 ～ 7.0
NOP 命令 256 回&割込み (CLINT)	7.0
NOP 命令 256 回&割込み (PLIC)	7.0 ～ 7.6

3 性能評価

例外処理機構によるオーバーヘッドを調査するために今回実装した例外処理機構の性能評価を行う．性能評価方法としては NOP 命令を 256 回実行，同期例外を 256 回発生，NOP 命令 256 回実行と 100 サイクル事に CLINT または PLIC を通じてタイマー割り込みを発生するというそれぞれの場合の実行時間の測定をして平均するという方法である．表 1 はその結果である．この結果から，NOP 命令 256 回と同期例外 256 回では，約 2.2ms から 3.0ms の差があるが，これは例外ハンドラの実行によるものと考えられ，CLINT と PLIC について NOP 命令 256 回と比較をしても約 3.0ms から 3.6ms の差であり，このことからオーバーヘッドは少ないことが分かる．

4 まとめ

今回の研究では，RISC-V 版の FabScalar 用機能シミュレータの例外処理機構について，既存の RISC-V シミュレータの実装手法では協調動作を行うためには問題点があるため，新たな実装手法を提案し，動作検証及び性能評価を行った．今後の展望としては，FS を行えるよう更なる実装をしていきたいと思う．

参考文献

[1] N. K. Choudhary et al., ”FabScalar: Composing synthesizable RTL designs of arbitrary cores within a canonical superscalar template,” 2011 38th Annual International Symposium on Computer Architecture (ISCA), 2011, pp. 11-22, doi: 10.1145/2000064.2000067.

[2] RISC-V Foundation, ”The RISC-V Instruction Set Manual Volume II: Privileged Architecture Document Version 20211203”

[3] RISC-V Foundation, ”RISC-V Advanced Core Local Interruptor Specification”, https://github.com/riscv/riscv-aclint/blob/main/riscv-aclint.adoc, 最終閲覧日 2022/1/7

[4] RISC-V Foundation, ”RISC-V Platform-Level Interrupt Controller Specification”, https://github.com/riscv/riscv-plic-spec/blob/master/riscv-plic.adoc, 最終閲覧日 2022/1/7