Hardware-based Kernel-Bypass Exceptions to Accelerate Floating Point Tracing and Virtualization

Karl Hallsby
Northwestern University
Evanston, IL, USA
kgh@u.northwestern.edu

Liam Strand
Northwestern University
Evanston, IL, USA
ltrs@u.northwestern.edu

Peter Dinda
Northwestern University
Evanston, IL, USA
pdinda@northwestern.edu

Abstract

Delivering instruction exceptions to user-space code enables a range of services, including floating point tracing and virtualization. Unfortunately, the usual forms of such delivery, signals or even specialized kernel modules, have high latency and overhead. We propose kernel-bypass exceptions (KBEs), hardware support to deliver certain exceptions safely and directly to user-space handlers. We then describe RAFT-V, a proof-of-concept, open-source prototype that extends the SonicBOOM RISC-V core and Linux to implement KBEs. RAFT-V also implements floating point trap support, which was not previously available on RISC-V in any form. RAFT-V lowers the latency and overhead of delivering these and other instruction exceptions to a user-space handler by 30× compared to signals. We use this functionality in RISC-V ports of the open-source FPSpy and FPVM floating point tracing and virtualization tools, reducing their costs by a factor of 3×. Our changes to add KBE and floating point trap features to RISC-V only marginally increase the overall hardware footprint and do not affect its critical path.

Keywords

kernel-bypass exceptions (KBEs), RISC-V, instruction exceptions, floating point arithmetic, correctness

ACM Reference Format:
Karl Hallsby, Liam Strand, and Peter Dinda. 2026. Hardware-based Kernel-Bypass Exceptions to Accelerate Floating Point Tracing and Virtualization. In *The 35th International Symposium on High-Performance Parallel and Distributed Computing (HPDC '26), July 13–16, 2026, Cleveland, OH, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3806645.3820063

1 Introduction

Delivering exceptions directly to user-space enables and accelerates a wide range of services which rely on exceptions to function. For example, a far memory system might be driven by page faults delivered to the kernel (e.g. [1]), while it is known the remote network access can be much better facilitated in user-space (e.g. [16]). Far memory systems swap frequently, thus reducing the costs of delivering page faults to user-space in such a system would have significant benefits. In this work we focus on delivering floating point traps directly to user-space, to accelerate floating point tracing or virtualization, as in FPSpy [8] and FPVM [9, 18].

We focus on precise synchronous exceptions [5, Chapter 8] that are the result of the processor finding a problem with the continued interpretation of a machine instruction. This event (e.g, page fault, invalid opcode, floating point trap, etc.) triggers a privilege mode change and dispatch of the event with additional data (e.g., access type, address) to the kernel via a handler table. The handler code can be assured that *the faulting instruction and subsequent instructions have not been committed* by the hardware and the data includes the address of the faulting instruction (the exception is "precise"). Instructions that deliberately produce events (e.g, breakpoint instructions, syscalls) behave similarly.

2 Kernel-Bypass Exceptions (KBEs)

Normally, the kernel gets the first shot at handling an exception, even those that are irrelevant to it, such as a floating point trap in tracing or virtualization services, or one that the application or runtime intends to handle, such as an `mprotect()`ed region in a garbage collector or far memory system. The prevailing ideology is that exceptions are by their very nature *exceptional*, i.e. rare, meaning the exception-handling path should not be heavily optimized.

Kernel-Bypass Exceptions (KBEs) are exactly what they sound like, some or all exceptions are delivered directly to user-space, bypassing the kernel. KBEs are not an entirely new concept; prior work proposed a similar concept in 1994 [17]. However, to the best of our knowledge, no hardware implementation of KBEs has been reported, and using KBEs for floating point traps is entirely new.

Prior work focused on allowing and building user-level *interrupts*, instead of exceptions, with a case being made as far back as 2002 [15]. Intel introduced user-level interprocessor interrupts (UIPIs) with Sapphire Rapids [12, Chapter 7] and it exhibits good performance [14]. UIPIs enabled a range of services; such as an entirely user-level preemptive threading library [10]. In addition, very recent work, [3] shows how to extend UIPIs to handle external interrupts to accelerate other user-level services.

Delivering exceptions directly to user-space means there is no context switch out of the user-process, because the process has explicitly *opted into* handling them. This has a variety of benefits, two of the largest are that the usual hardware need to flush the entire pipeline for security can be omitted (the core is simply entering a different part of the same user program), and various caches do not need to be flushed (there is no possibility of a side channel between actors since there is only one actor).

This effort was partially supported by the United States National Science Foundation (NSF) under awards CNS-2211315, CCF-2119069, and CNS-2211508.

HPDC '26, Cleveland, OH, USA

ACM ISBN 979-8-4007-2640-8/26/07
https://doi.org/10.1145/3806645.3820063

3 RAFT-V

RAFT-V is our prototype implementation of KBEs in the SonicBOOM [19] RISC-V core. SonicBOOM [19] (often shortened to just BOOM) is a modern high-performance micro-coded Out-Of-Order RISC-V CPU that plugs into Chipyard's SoC framework [2] and has previously been fabricated [6]. Our RAFT-V is also the first RISC-V core to support precise floating point traps (also referred to as floating point exceptions, FPEs, by software), which serves as the primary exception evaluated in this paper.

3.1 FPEs

Bringing FPE support to BOOM required deep understanding of how BOOM manages floating point state for µops. Our changes included: choosing an exception number for FPEs, adding the enable mask CSR (`fflags_enable`), modifying the Re-Order Buffer (ROB) so each µop entry stores a copy of the `fflags_enable` CSR at dispatch time, and finally adding logic to use the installed mask to raise exceptions. Each µop **must** carry a copy of `fflags_enable` so that floating point exceptions are raised correctly and precisely while executing out-of-order.

3.2 KBEs

Unlike the FPE changes, implementing KBEs for RAFT-V required very targeted changes. KBEs required defining a handful of new CSRs, defining a new mode-return instruction (`URET`), and re-enabling a bit-field on an already-existing CSR; all of which mirror the already-existing privilege-mode management hardware. This creates an additional layer in the Mux tree that determines which privilege level an exception will be delegated to.

Additionally, we created `ESTEP`, a custom instruction that acts as a "KBE native" instruction exception with its own custom exception number. This only involved allocating: a new exception number, adding a new instruction to RISC-V, and then decoding it.

3.3 OpenSBI and Linux

We modified OpenSBI (the firmware) and Linux to make our hardware changes visible and behave as expected. Linux handles all of the new architectural state like the already-existing pieces, swapping them on context switches, clearing them on new process invocation, and a kernel module provides an interface for user programs to request KBEs. We also provide a minimal user-level library shim that allows users to treat KBEs *as if* they were regular signals, but signals that are delivered significantly faster.

4 Evaluation

Our evaluation employs the widely-used and well-validated FireSim tool [13], which is a framework for FPGA-accelerated high-speed cycle-accurate simulation. Performance critical parts of RAFT-V are synthesized through the Vivado toolchain, permitting RAFT-V to boot a full Linux kernel to an interactive user shell quickly while remaining cycle-accurate, permitting real programs to be evaluated.

4.1 KBE Delivery Latency

We first measured the benefit KBEs can provide in their best case: exceptions are frequent and the "null" exception handler does no work. Figure 1 shows that for identical "null" handlers, KBEs can speed up exceptions by 30× over normal signals! In particular, because KBEs never need to change privilege modes, any additional work the hardware does to ensure a clean separation between user programs and the kernel is (essentially) removed.

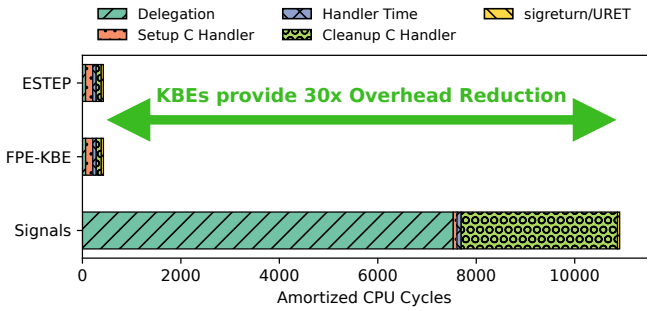

Figure 1: KBEs don't enter the kernel, reducing overhead greatly.

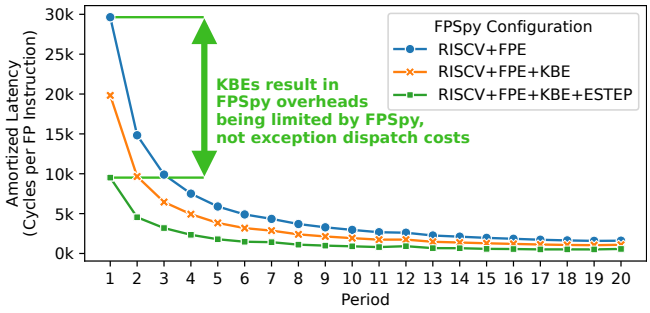

Figure 2: KBEs reduce FPSpy overheads by up to 3×. Period k means that every kth floating point instruction causes a trap.

Figure 1 shows that most of the time handling exceptions is delegation to and return from the kernel. For KBEs, the majority of time moves to software instead of hardware, such as context management; saving and restoring GPRs and CSRs to the stack.

4.2 Effects on Program Performance

FPSpy is fully ported and can leverage all of RAFT-V's features. We swept the frequency of FPEs to be handled by FPSpy in Figure 2 to determine the benefits FPSpy can expect when running on RAFT-V. We also evaluated various mini-apps and benchmarks, including Miniaero [7] and NAS 3.0 [4].

FPVM is partially ported; its core exception system, decoder, and emulator are operational and produce correct results, while some peripherals require more work. The effects of KBEs on FPVM were studied by feeding programs crafted for FPVM's current capabilities and we recorded a 2× improvement in end-to-end latency.

4.3 Hardware Footprint & Critical Path

The changes required to implement FPEs and KBEs on RAFT-V were small, making a taped-out realization likely. RAFT-V only consumed an additional 1–3% of major components (LUTs, Flip-Flops) while specialty components (DSPs, BRAM) are unchanged. RAFT-V's FPE

support produced a 65.11% increase in SRL (Shift Register Lookup) consumption. None of these changes affect BOOM's critical path.

5 Conclusion

KBEs provide significant speedups in microbenchmarks (30×) and 2–3× in regular usage. However, floating point exceptions are just a convenient exception to start with. Using RAFT-V as a starting point, we believe other exceptions, such as page faults, could be passed to user-space to enable new services or accelerate existing ones. Additional information about RAFT-V and its implementation is available in [11], with the design available on GitHub[1].

References

[1] Emmanuel Amaro, Christopher Branner-Augmon, Zhihong Luo, Amy Ousterhout, Marcos K. Aguilera, Aurojit Panda, Sylvia Ratnasamy, and Scott Shenker. 2020. Can Far Memory Improve Job Throughput?. In *Proceedings of the* 15^{th} *European Conference on Computer Systems* (Heraklion, Greece) *(EuroSys '20)*. Association for Computing Machinery, New York, NY, USA, Article 14, 16 pages. doi:10.1145/3342195.3387522

[2] Alon Amid, David Biancolin, Abraham Gonzalez, Daniel Grubb, Sagar Karandikar, Harrison Liew, Albert Magyar, Howard Mao, Albert Ou, Nathan Pemberton, Paul Rigge, Colin Schmidt, John Wright, Jerry Zhao, Yakun Sophia Shao, Krste Asanović, and Borivoje Nikolić. 2020. Chipyard: Integrated Design, Simulation, and Implementation Framework for Custom SoCs. *IEEE Micro* 40, 4 (2020), 10–21. doi:10.1109/MM.2020.2996616

[3] Berk Aydogmus, Linsong Guo, Danial Zuberi, Tal Garfinkel, Dean Tullsen, Amy Ousterhout, and Kazem Taram. 2025. Extended User Interrupts (xUI): Fast and Flexible Notification without Polling. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2025)*. 373–389.

[4] D.H. Bailey, E. Barszcz, J.T. Barton, D.S. Browning, R.L. Carter, L. Dagum, R.A. Fatoohi, P.O. Frederickson, T.A. Lasinski, R.S. Schreiber, H.D. Simon, V. Venkatakrishnan, and S.K. Weeratunga. 1991. The Nas Parallel Benchmarks. *International Journal of High Performance Computing Applications* 5, 3 (Sept. 1991), 63–73.

[5] Randal E. Bryant and David R. O'Hallaron. 2015. *Computer Systems: A Programmer's Perspective* (3 ed.). Pearson.

[6] Christopher Celio, Pi-Feng Chiu, Krste Asanović, Borivoje Nikolić, and David Patterson. 2019. BROOM: An Open-Source Out-of-Order Processor With Resilient Low-Voltage Operation in 28-nm CMOS. *IEEE Micro* 39, 2 (March 2019), 52–60. doi:10.1109/MM.2019.2897782

[7] Paul Crozier, Heidi Thornquist, Robert Numrich, Alan Williams, H. Edwards, Eric Keiter, Mahesh Rajan, James Willenbring, Douglas Doerfler, and Michael Heroux. 2009. *Improving performance via mini-applications.* Technical Report SAND2009-5574. Sandia National Laboratories. doi:10.2172/993908

[8] Peter Dinda, Alex Bernat, and Conor Hetland. 2020. Spying on the Floating Point Behavior of Existing, Unmodified Scientific Applications. In *Proceedings of the 29th ACM Symposium on High-performance Parallel and Distributed Computing (HPDC 2020)*. Best Paper.

[9] Peter Dinda, Nick Wanninger, Jicheng Ma, Alex Bernat, Charles Bernat, Souradip Ghosh, Chris Kraemer, and Yehea Elmasry. 2022. FPVM: Towards a Floating Point Virtual Machine. In *Proceedings of the 31st ACM Symposium on High-performance Parallel and Distributed Computing (HPDC)*.

[10] Linsong Guo, Danial Zuberi, Tal Garfinkel, and Amy Ousterhout. 2025. The Benefits and Limitations of User Interrupts for Preemptive Userspace Scheduling. 1015–1032. https://www.usenix.org/conference/nsdi25/presentation/guo

[11] Karl Hallsby, Liam Strand, Nick Wanninger, Nadharm Dhiantravan, and Peter Dinda. 2025. *Architecture-independent Floating Point Spying and an Architecture for Floating Point Spying.* Technical Report NU-CS-2025-27. Northwestern University. https://www.mccormick.northwestern.edu/computer-science/documents/nu-cs-2025-27.pdf

[12] Intel Corporation. 2023. *Intel 64 and IA-32 Architectures Software Developers Manual, Volume 3A: System Programming Guide, Part 1.*

[13] Sagar Karandikar, Howard Mao, Donggyu Kim, David Biancolin, Alon Amid, Dayeol Lee, Nathan Pemberton, Emmanuel Amaro, Colin Schmidt, Aditya Chopra, Qijing Huang, Kyle Kovacs, Borivoje Nikolic, Randy Katz, Jonathan Bachrach, and Krste Asanović. 2018. FireSim: FPGA-accelerated Cycle-exact Scale-out System Simulation in the Public Cloud. In *Proceedings of the 45th Annual International Symposium on Computer Architecture (ISCA '18)*. IEEE Press, Piscataway, NJ, USA, 29–42. doi:10.1109/ISCA.2018.00014

[14] Sohit Mehta. 2021. User Interrupts: A Faster Way to Signal. In *Proceedings of the Linux Plumbers Conference (LBC 2021)*.

[15] Mike Parker. 2002. A case for user-level interrupts. *SIGARCH Computer Architecture News* 30, 3 (June 2002), 17–18.

[16] Zhenyuan Ruan, Malte Schwarzkopf, Marcos K. Aguilera, and Adam Belay. 2020. AIFM: High-Performance, Application-Integrated Far Memory. In *Proceedings of the* 14^{th} *USENIX Symposium on Operating Systems Design and Implementation (OSDI '20)*. USENIX Association, 315–332. https://www.usenix.org/conference/osdi20/presentation/ruan

[17] Chandramohan A. Thekkath and Henry M. Levy. 1994. Hardware and software support for efficient exception handling. In *Proceedings of the Sixth International Conference on Architectural Support for Programming Languages and Operating Systems* (San Jose, California, USA) *(ASPLOS VI)*. Association for Computing Machinery, New York, NY, USA, 110–119. doi:10.1145/195473.195515

[18] Nick Wanninger, Nadharm Dhiantravan, and Peter Dinda. 2025. Virtualization So Light, It Floats!: Accelerating Floating Point Virtualization. In *Proceedings of the 34th ACM Symposium on High-performance Parallel and Distributed Computing (HPDC)*.

[19] Jerry Zhao, Ben Korpan, Abraham Gonzalez, and Krste Asanovic. 2020. SonicBOOM: The 3rd Generation Berkeley Out-of-Order Machine. (May 2020).

[1] https://github.com/Constellation-FPGA/chipyard