Glorifying the If-Statement

PHMon: A Programmable Hardware Monitor and Its Security Use Cases

Leila Delshadtehrani, Sadullah Canakci, Boyou Zhou, Schuyler Eldridge, Ajay Joshi, and Manuel Egele

---

Nathan S. Rutherford

Email: nathan.rutherford.2019@live.rhul.ac.uk

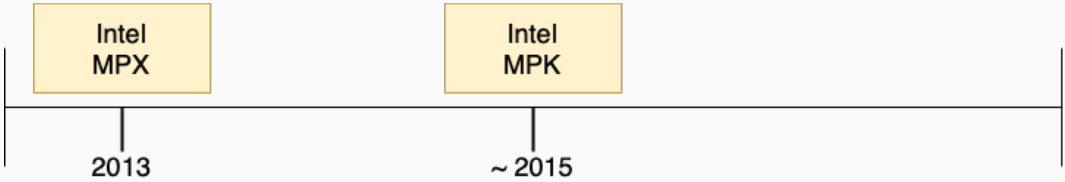
Twitter: @rutherfordns

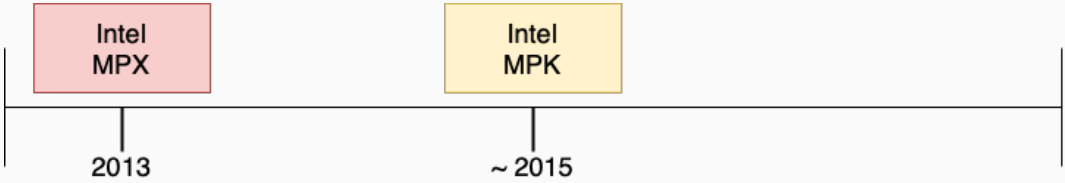
Website: nsrutherford.com

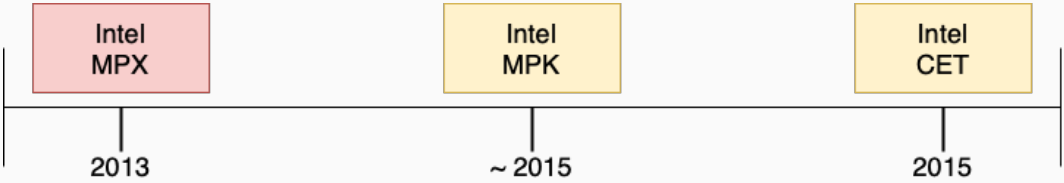
18^{th} November 2020

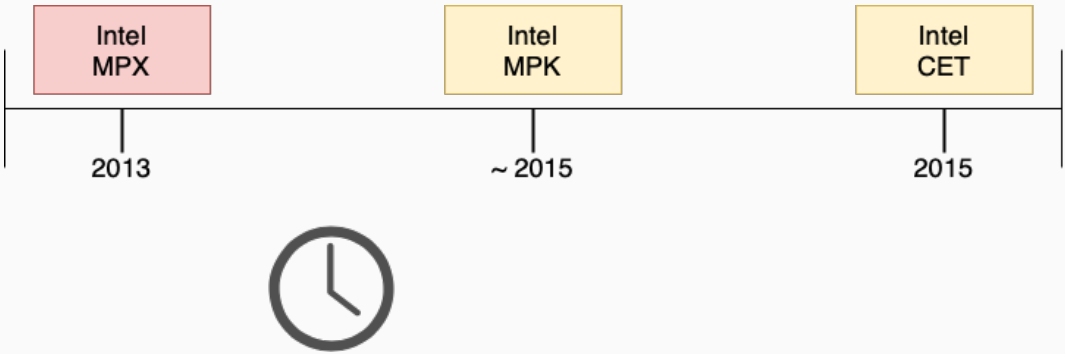
Royal Holloway, University of London

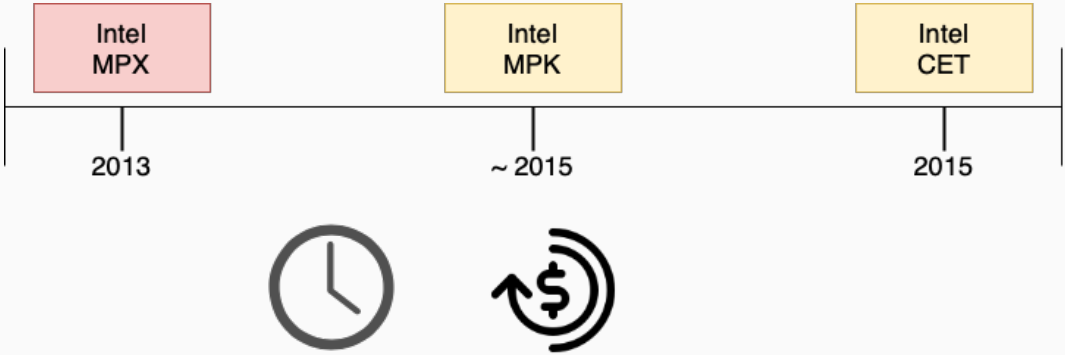

Intel
MPX
2013

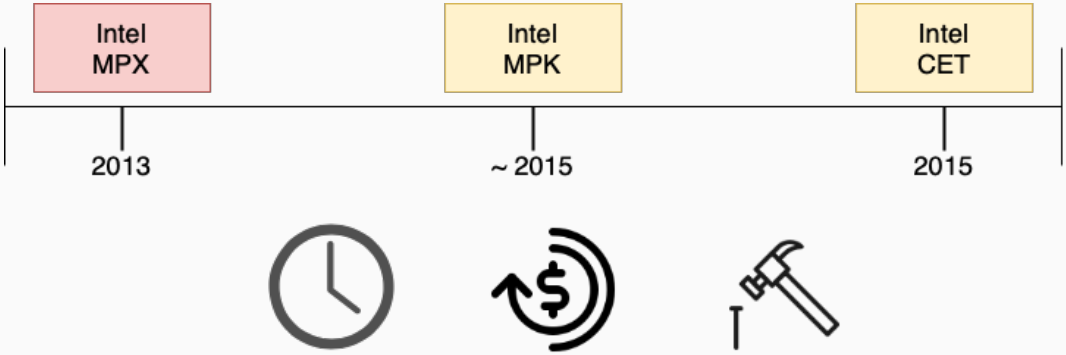

Intel
MPX
Intel
MPK
2013
~ 2015


Intel
MPX
Intel
MPK
2013
~ 2015


Intel
MPX
Intel
MPK
Intel
CET
2013
~ 2015
2015


Intel
MPX
Intel
MPK
Intel
CET
2013
~ 2015
2015


Intel
MPX
Intel
MPK
Intel
CET
2013
~ 2015
2015


Intel
MPX
Intel
MPK
Intel
CET
2013
~ 2015
2015

Outline

PHMon

Overview

Programmable Hardware Monitor

Overview

Programmable Hardware Monitor

Event-Action Monitoring Model

Overview

Programmable Hardware Monitor

Event-Action Monitoring Model

Three Stage Pipeline:

Overview

Programmable Hardware Monitor

Event-Action Monitoring Model

Three Stage Pipeline:

- Collect

Overview

Programmable Hardware Monitor

Event-Action Monitoring Model

Three Stage Pipeline:

- Collect
- Examine

Overview

Programmable Hardware Monitor

Event-Action Monitoring Model

Three Stage Pipeline:

- Collect
- Examine
- Action

Trace Unit (TU)

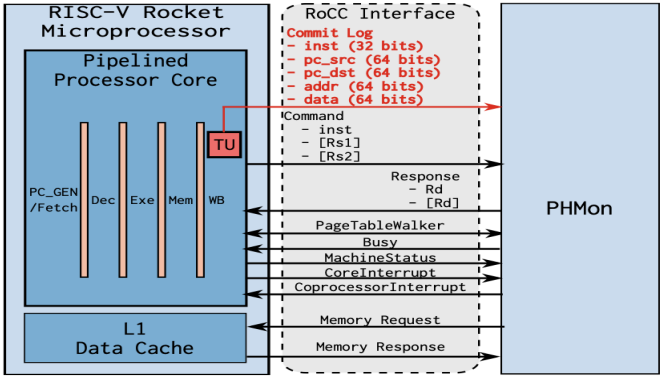


Figure 2: The RoCC interface extended with *commit log* execution trace.

Match and Action Units (MU & AU)

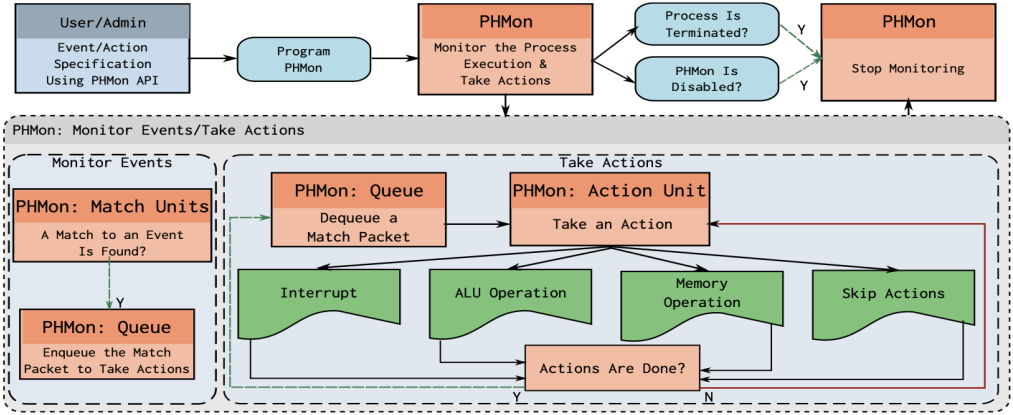


Figure 1: An overview of the *event-action* model provided in PHMon.

Use-cases

Potential Use-cases

Shadow Stack Protection

Potential Use-cases

Shadow Stack Protection

Hardware Accelerated Fuzzing

Potential Use-cases

Shadow Stack Protection

Hardware Accelerated Fuzzing

Prevention of Information Leakage

Potential Use-cases

Shadow Stack Protection

Hardware Accelerated Fuzzing

Prevention of Information Leakage

Watchpoints and Accelerated Debugger

Potential Use-cases

Shadow Stack Protection

Hardware Accelerated Fuzzing

Prevention of Information Leakage

Watchpoints and Accelerated Debugger

Hardware Accelerated Fuzzing

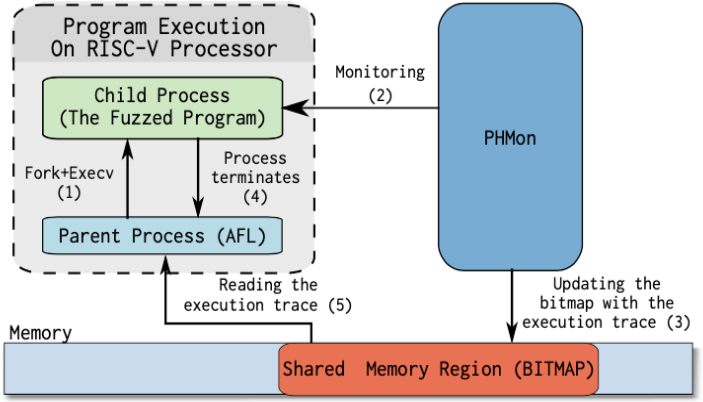


Figure 4: Integration of PHMon with AFL.

Evaluation

Overview

Evaluate performance of the aforementioned four use-cases

Overview

Evaluate performance of the aforementioned four use-cases

RISC-V Rocket Chip with PHMon as RoCC

Overview

Evaluate performance of the aforementioned four use-cases

RISC-V Rocket Chip with PHMon as RoCC

PHMon implemented on Xilinx FPGA

Overview

Evaluate performance of the aforementioned four use-cases

RISC-V Rocket Chip with PHMon as RoCC

PHMon implemented on Xilinx FPGA

Linux v4.15 Target

Overview

Evaluate performance of the aforementioned four use-cases

RISC-V Rocket Chip with PHMon as RoCC

PHMon implemented on Xilinx FPGA

Linux v4.15 Target

Disclaimer:

No L2 Cache on set-up that may have impacted performance

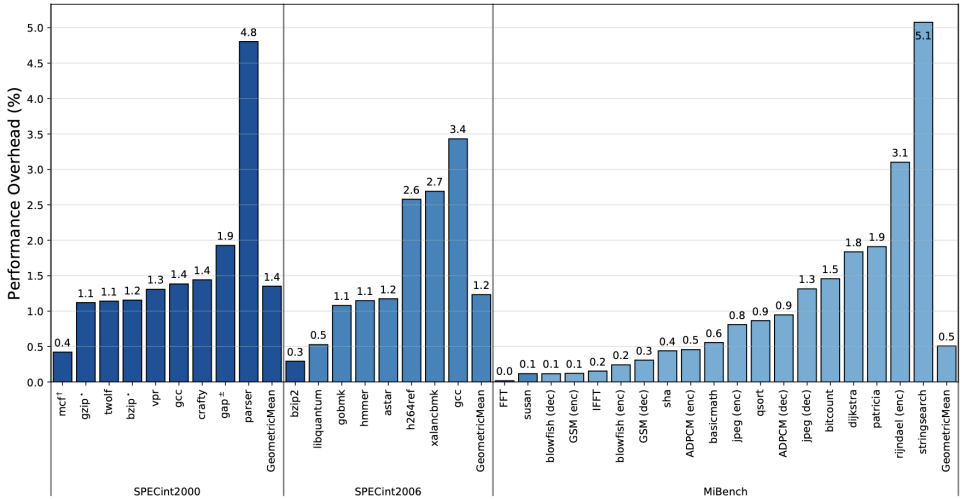


Figure 5: The performance overhead of PHMon as a shadow stack.

† We were not able to run mcf benchmark with `reference` input on our evaluation board; as a result, we used the `test` input for this benchmark.

* Due to the memory limitations of our evaluation board, we had to reduce the buffer size of the `reference` input to 3 MB for gzip and bzip2 benchmarks.

± We had to use `-O0` and an input buffer size of 96 MB to successfully run gap benchmark.

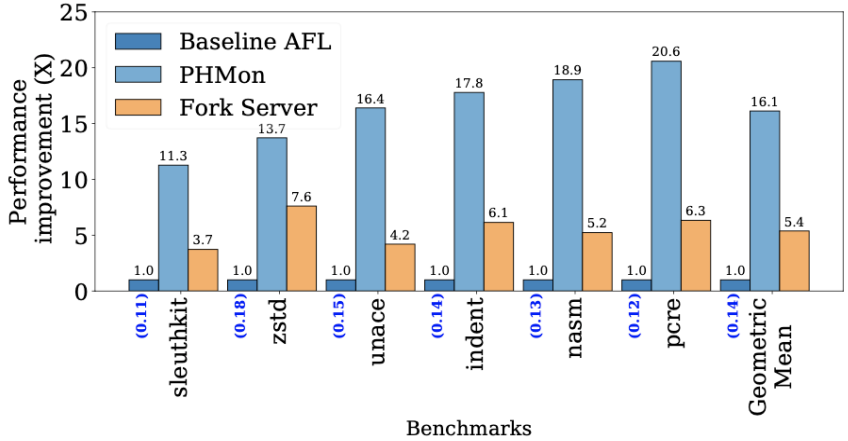


Figure 6: Performance improvement of PHMon over the baseline AFL compared to fork server AFL. The numbers below the "Baseline AFL" bars show the number of executions per second for the baseline AFL.

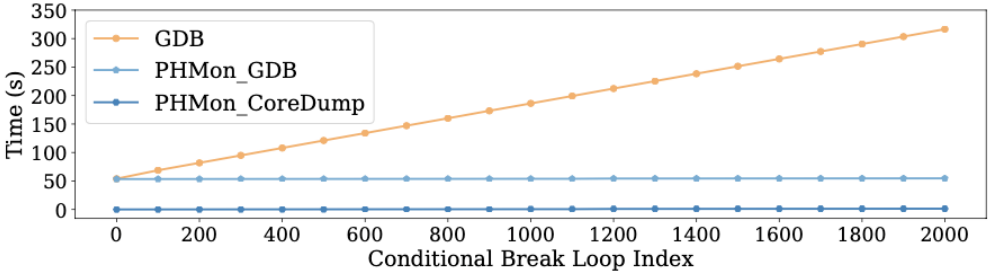


Figure 7: The performance overhead of PHMon compared to GDB for a loop conditional breakpoint.

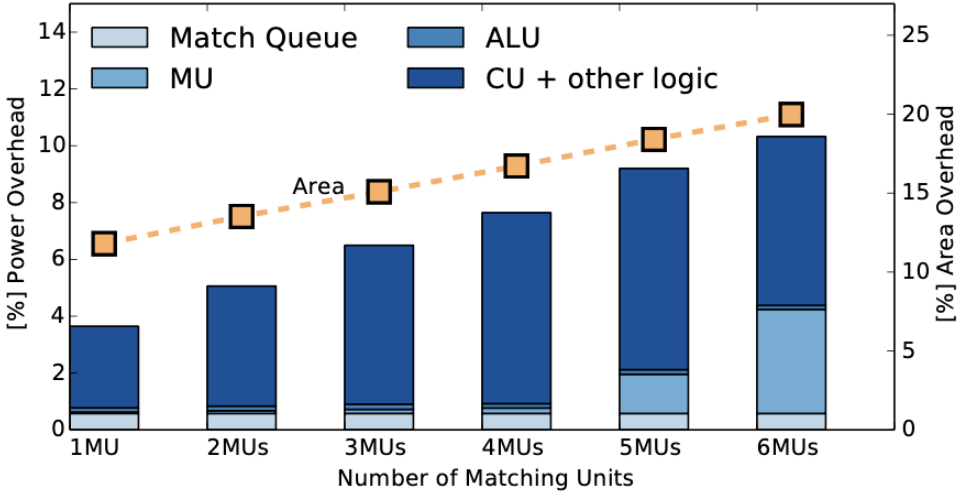


Figure 8: The power and area overheads of PHMon components compared to the baseline Rocket processor.

Conclusion

Discussion

Well motivated and reasonable approach

Discussion

Well motivated and reasonable approach

Recontextualising of Match-Action Pipeline for security monitoring in an OS

Discussion

Well motivated and reasonable approach

Recontextualising of Match-Action Pipeline for security monitoring in an OS

Fidelity and impact for research I would expect for USENIX

Discussion

Well motivated and reasonable approach

Recontextualising of Match-Action Pipeline for security monitoring in an OS

Fidelity and impact for research I would expect for USENIX

Simplistic Threat Model

Summary

Popular hardware approaches only enforce a static security policy

Summary

Popular hardware approaches only enforce a static security policy

PHMon is a flexible hardware monitor leveraging an Event-Action Pipeline

Summary

Popular hardware approaches only enforce a static security policy

PHMon is a flexible hardware monitor leveraging an Event-Action Pipeline

- Enforce range of security policies

Summary

Popular hardware approaches only enforce a static security policy

PHMon is a flexible hardware monitor leveraging an Event-Action Pipeline

- Enforce range of security policies
- Implemented as Co-processor in RISC-V system

Summary

Popular hardware approaches only enforce a static security policy

PHMon is a flexible hardware monitor leveraging an Event-Action Pipeline

- Enforce range of security policies
- Implemented as Co-processor in RISC-V system
- Not dependant on architecture

Summary

Popular hardware approaches only enforce a static security policy

PHMon is a flexible hardware monitor leveraging an Event-Action Pipeline

- Enforce range of security policies
- Implemented as Co-processor in RISC-V system
- Not dependant on architecture

Evaluation shows .9% overhead for shadow-stack protection

Summary

Popular hardware approaches only enforce a static security policy

PHMon is a flexible hardware monitor leveraging an Event-Action Pipeline

- Enforce range of security policies
- Implemented as Co-processor in RISC-V system
- Not dependant on architecture

Evaluation shows .9% overhead for shadow-stack protection

Improves performance of AFL by up to 16x, finding more bugs

Questions

1. Are there any caching opportunities for PHMon that could improve performance?
2. Does PHMon deal with branch prediction in monitoring?
3. Can monitoring rules be updated in real-time, or does it require PHMon to be offline?