# The Guardian Council: Parallel Programmable Security

Sam Ainsworth and Timothy M. Jones

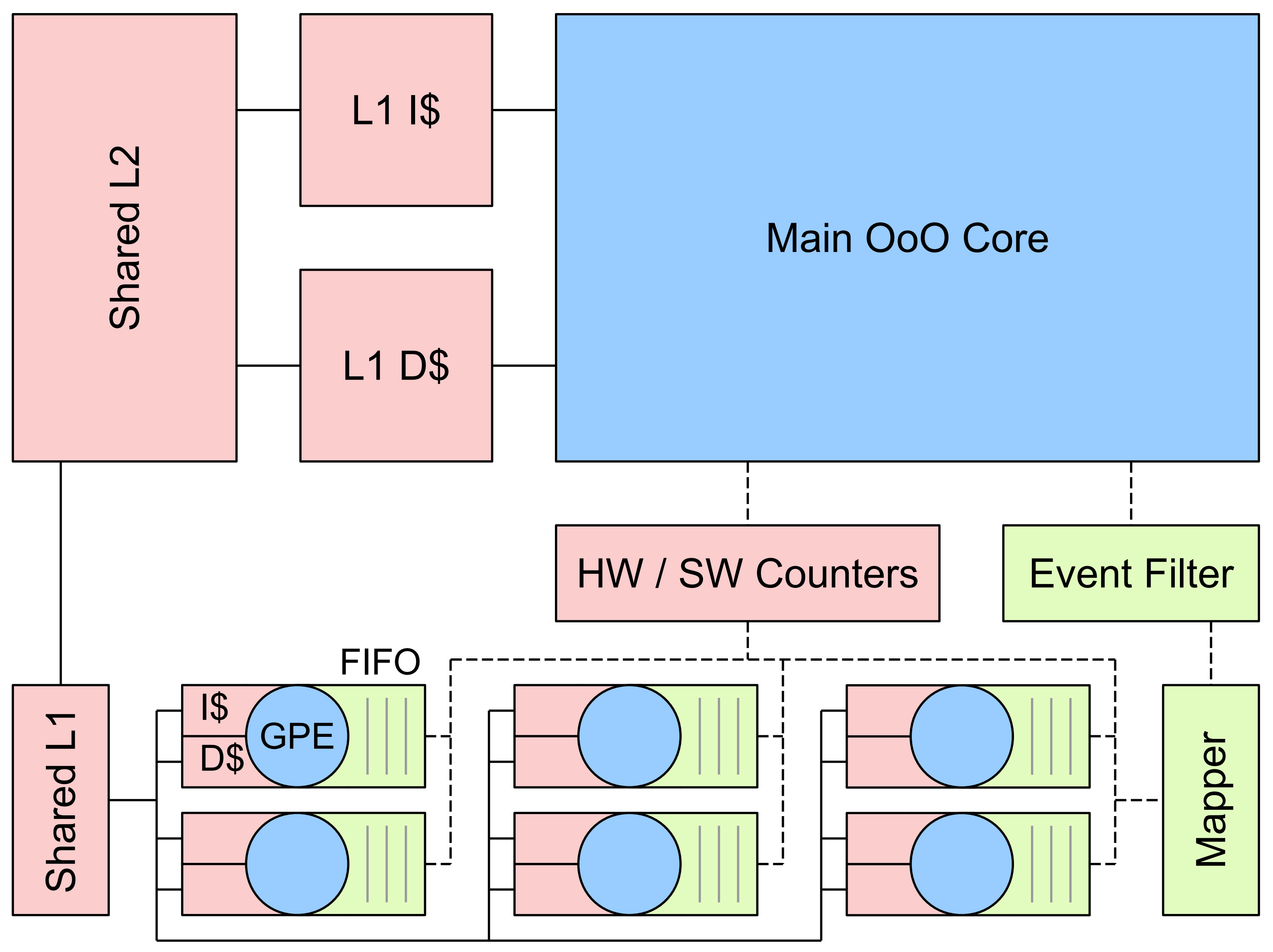
University of Cambridge

## The Best of Both Worlds

Systems security is becoming more challenging in the face of untrusted programs and system users. Safeguards against attacks currently in use, such as buffer overflows, control-flow integrity, side channels and malware, are limited. Software protection schemes, while flexible, are often too expensive, and hardware schemes, while fast, are too constrained or out-of-date to be practical.

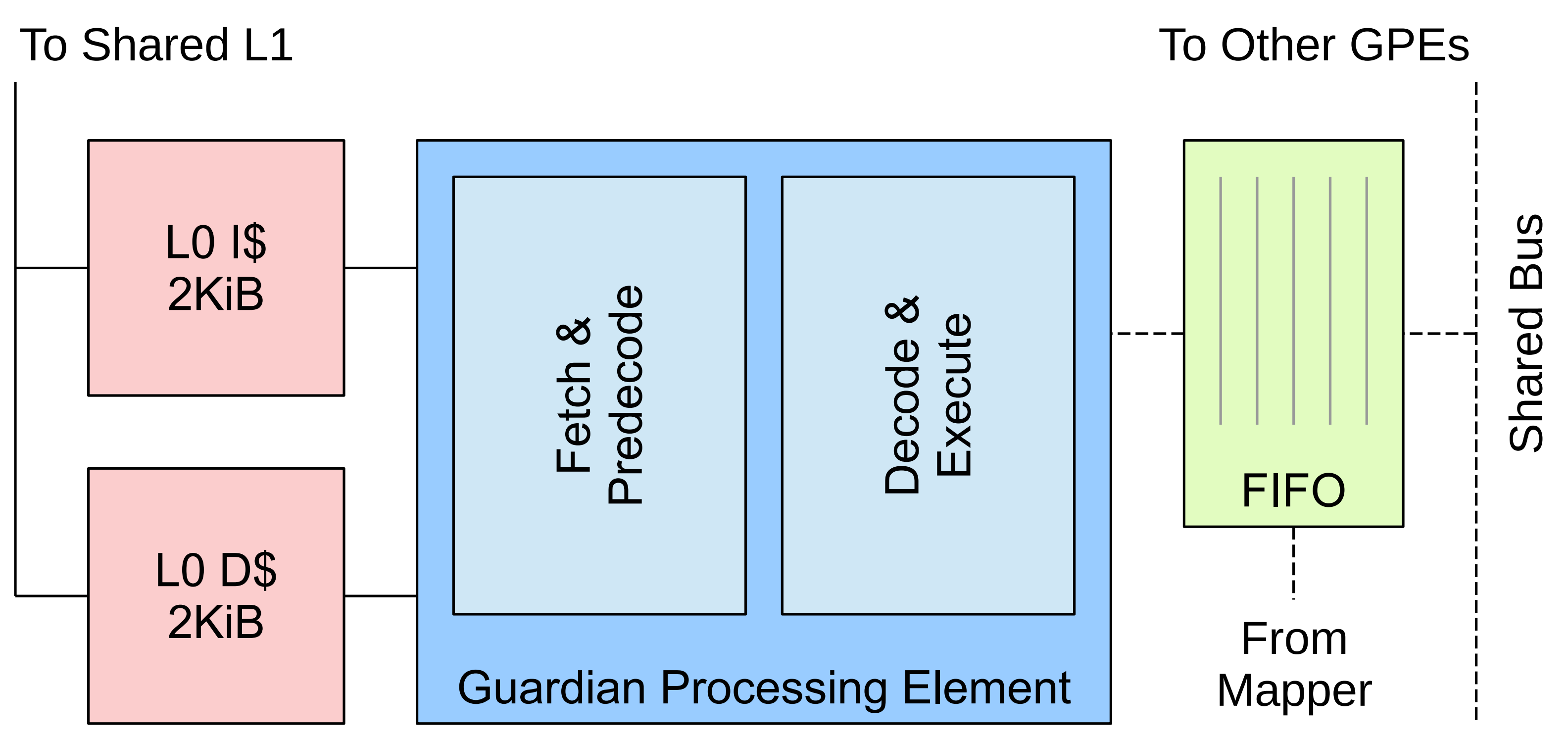
We demonstrate the best of both worlds with the Guardian Council, a novel parallel architecture to enforce a wide range of highly customisable and diverse security policies.

## The Guardian Council



We leverage heterogeneity and parallelism to perform security enforcement for a large high-performance core on a set of small microcontroller-sized Guardian Processing Elements. Alongside these highly parallel cores we provide fixed-function logging and communication units, and a powerful programming model, as part of an architecture for security.

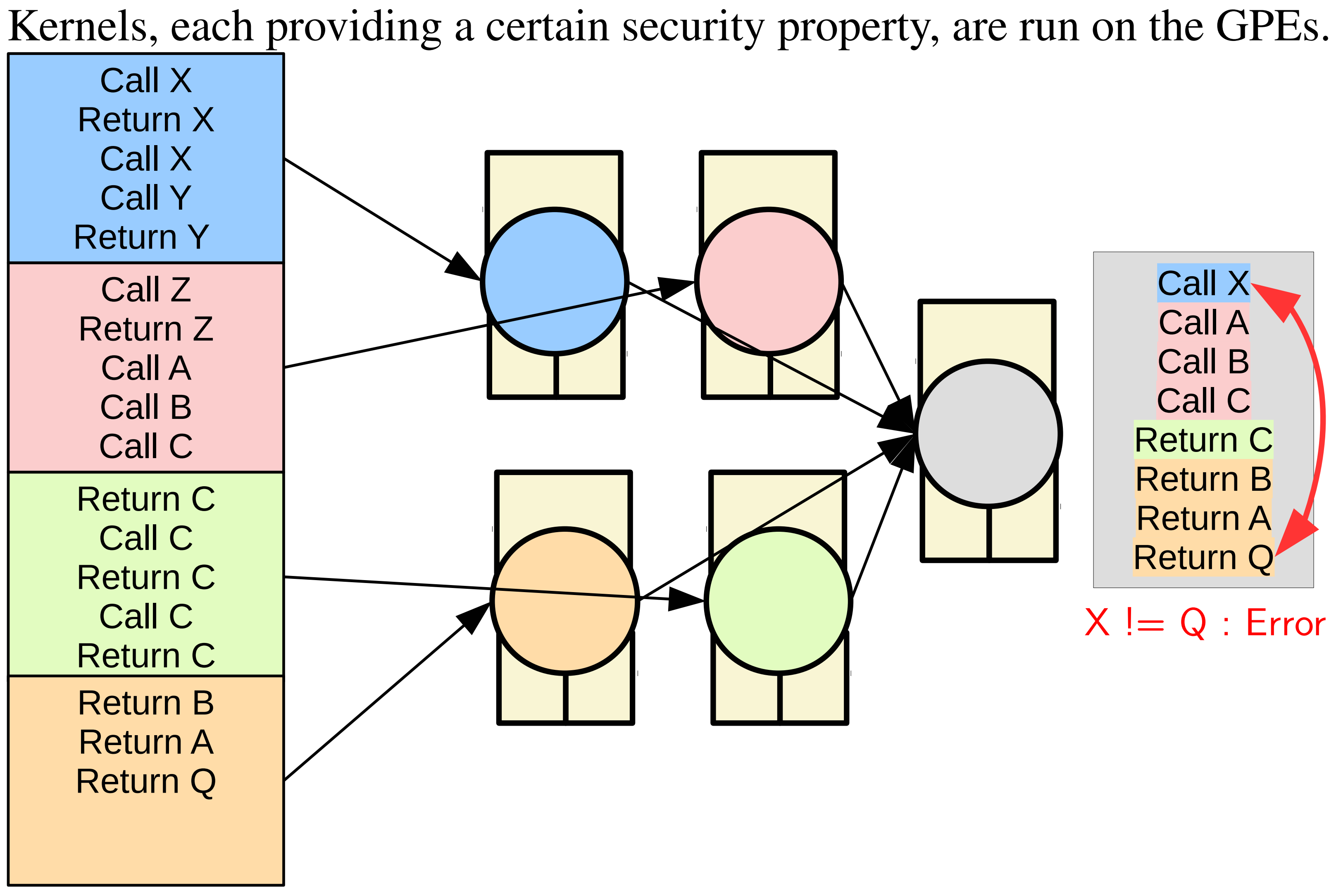
## Guardian Processing Elements



Guardian Processing Elements (GPEs) are many orders of magnitude more efficient than conventional out-of-order superscalar processors, bringing high-performance security at very low power and area overheads, through inter- and intra-task-level parallelism of security defences.

## Guardian Kernels

For a given process running on the main core, a number of Guardian Kernels, each providing a certain security property, are run on the GPEs.

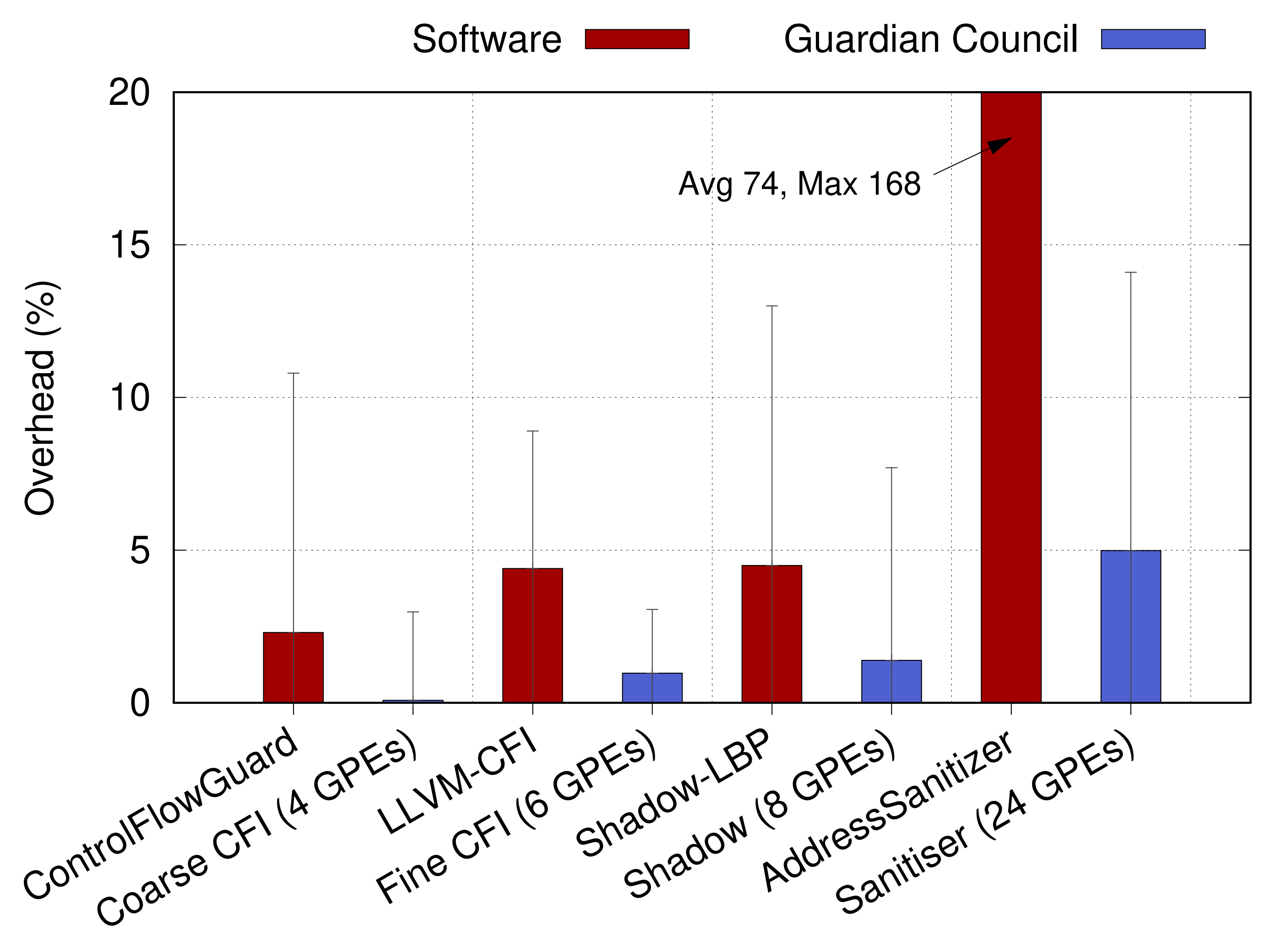


Here we show a shadow stack from observations of the main core's executed instructions (left), parallelised across five GPEs. The work is split up and filtered for common push-and-pops in a segment, before being aggregated to find inconsistent behaviour.

```
(op, addr) = get_fifo();
 if (op == call) stack.push(addr);
 else {
   if (stack.size() > 0) {
     assert(addr == stack.top());
     stack.pop();
   }
   else  put_fifo(0,(return, addr));
 }
```

Kernels are written in standard C/C++, with the addition of primitives for accessing events from the GPEs' FIFO stacks, via the main core's filter and mapper, or from other GPEs.

## Acceleration of Diverse Defences



## Acknowledgements

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC), through grant references EP/K026399/1, EP/P020011/1 and EP/M506485/1, and ARM Ltd.