Simulators

Processors Processing Processors

5SIA0 - Glenn Bergmans

Why simulators?

Axiom

We want to create a better computing system

We want to create a better computing system

- Faster
- Cheaper
- More power efficient
- Smaller
- Optimise general purpose vs application specific
- Etc, etc, etc…

You’re the computer architect

So you design this computer

You’re the computer architect

Right?

How can you explore the design space and verify results?

How can you explore the design space and verify results?

- Buy (or build) *all* hardware options
  - But that's probably a bit expensive…
- Use analytical models
  - Is it reliable? Reusable?
- Simulate the design points
  - That'd be awesome!

You didn't see that one coming, did you?

How do you know your new computer is better?

How do you know your new computer is better?

Simulate for:

- Performance
- Energy
- Power **(≠ Energy!)**
- Thermal
- Area

How much detail do you need?

How much detail do you need?

- Cycle accuracy vs Functionality?
- Caches?
- Full operating system?
- Disk accesses?
- Background tasks?
- Etc, etc, etc…

SiFive RTL Simulation Details

Truely bit accurate simulation

RTL Simulation

Simulate at gate level

For example:

- modelsim/questasim (Mentor)
- ncsim (Cadence)
- VCS (Synopsys)
- Icarus Verilog (Open Source)
- Etc, etc, etc…

RTL Simulation

Simulate at gate level

Advantages

- No need to build a custom simulator if you need RTL to build hardware anyway
- Highest level of precision and detail

(Obvious) disadvantage

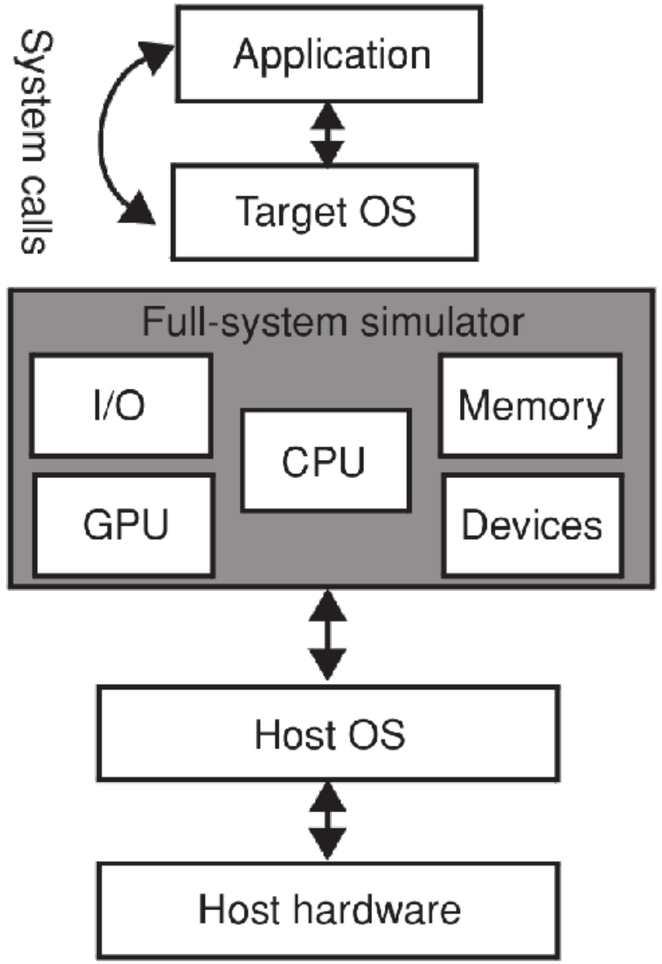
- It is *horribly* slow

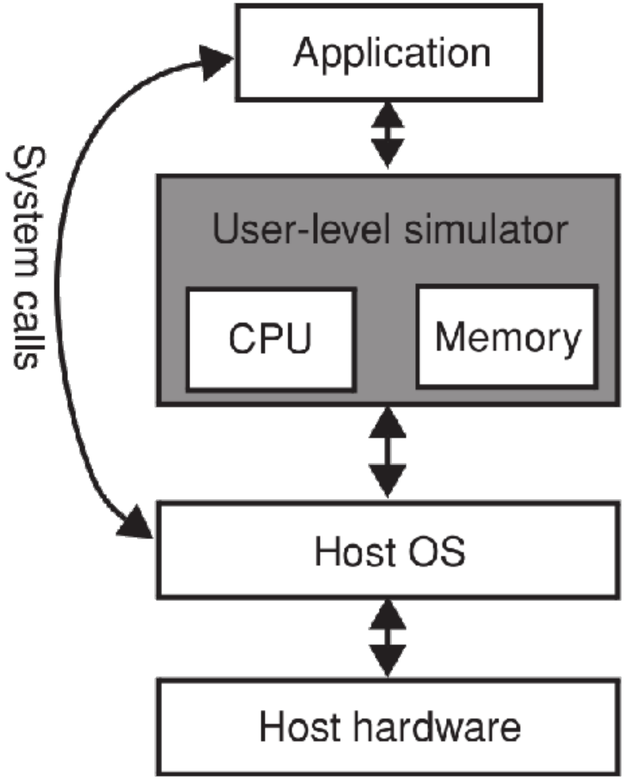
Improve RTL simulation speed with FPGA

- RTL designs can be run on an FPGA
  - First synthesise (slow) the RTL description
  - Then simulate (fast) at near actual speed
- Instrumentation needed to required to get detailed information out
  - Add debug cores and monitoring hardware

Maybe not *all* details

Full system vs User level





User level simulation

Famous example: Simple Scalar

(http://www.simplescalar.com/)

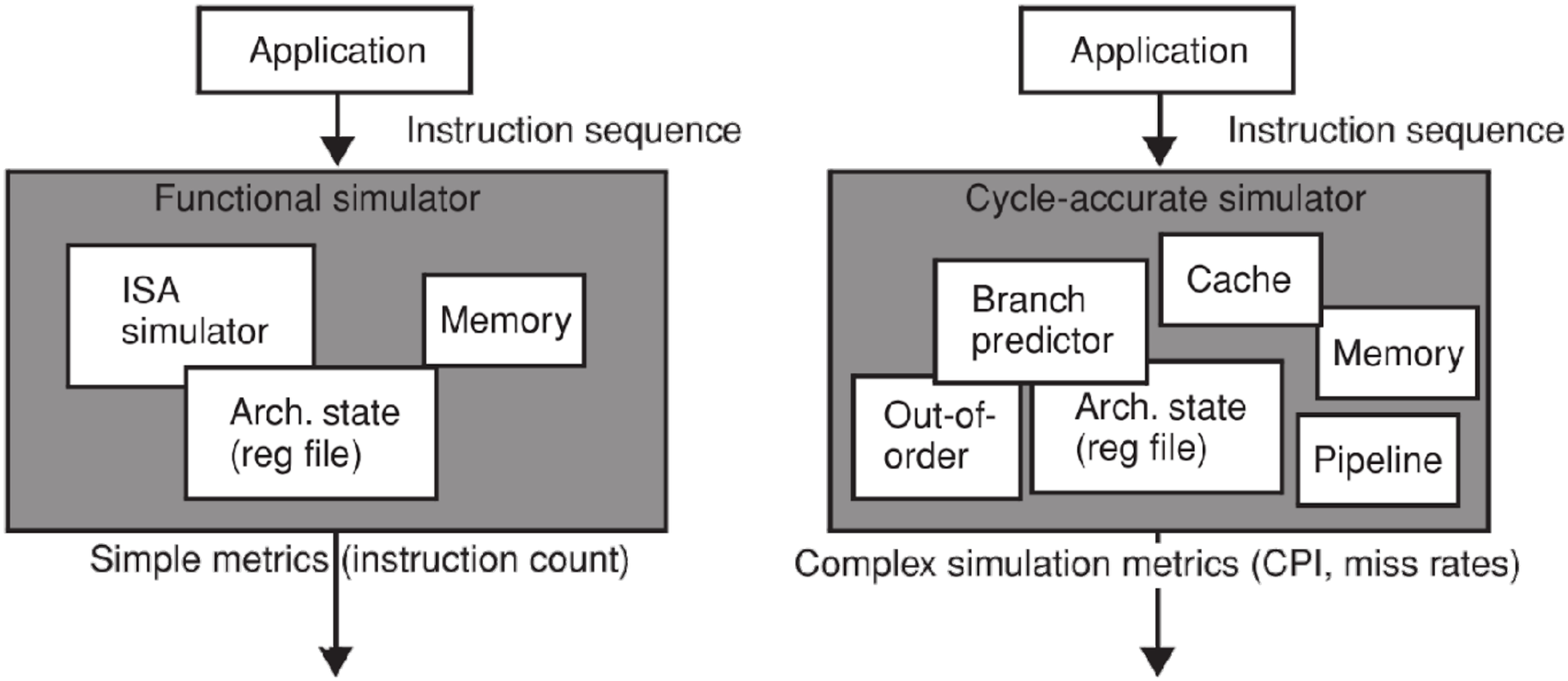
Advantages

- Fast to develop and update to new architectures
- Usually ‘accurate enough’

Disadvantage

- Any time spent in OS mode is not simulated accurately
  - Sometimes that’s not ‘accurate enough’
  - For example a database application typically spends 20%-30% in OS mode

Cycle accurate vs Functional



Cycle accurate vs Functional

Functional No/limited model of micro architecture

- An (add) instruction of the target can be translated to an (add) instruction on the host computer, and be simulated that way.
- Examples: Simple Scalar (sim-fast mode), QEMU (Full system using dynamic translation)

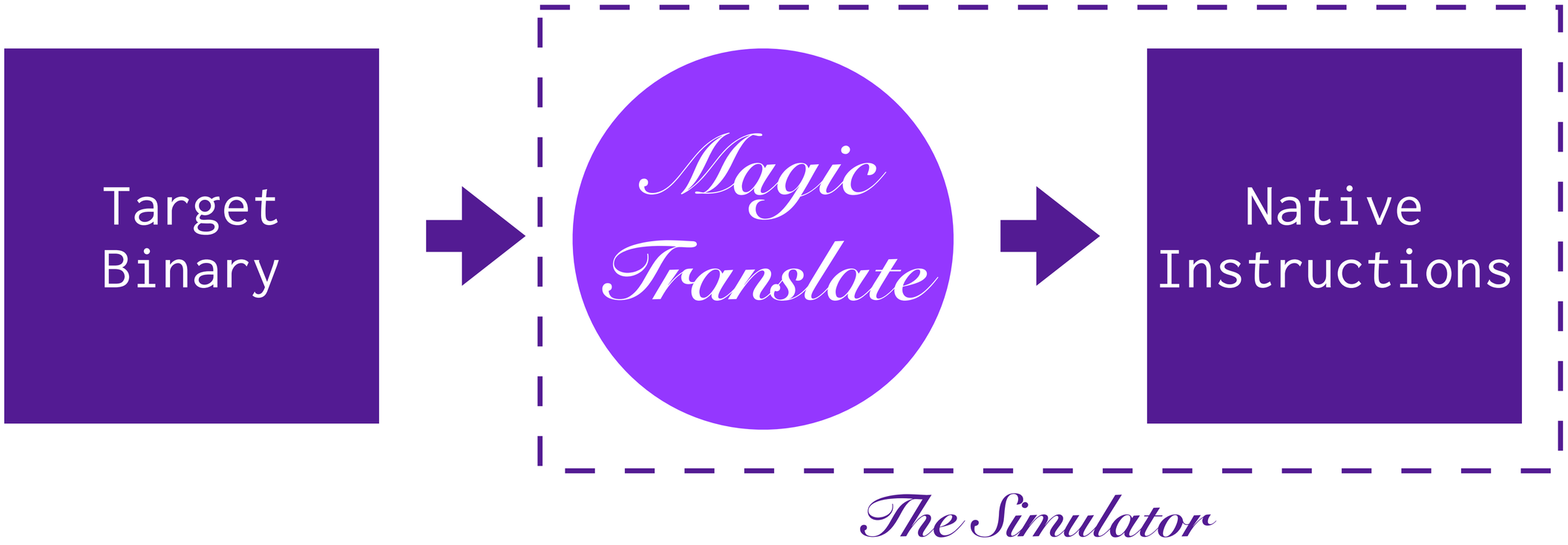
Cycle accurate Includes a model of micro architecture

- Block resources in the pipeline when an instruction executes
- Use target branch predictor scheme
- Out-of-order execution
- Example: Simple Scalar (sim-outorder mode)

Intermezzo

The internals of dynamic translation

The Magic of Dynamic Translate

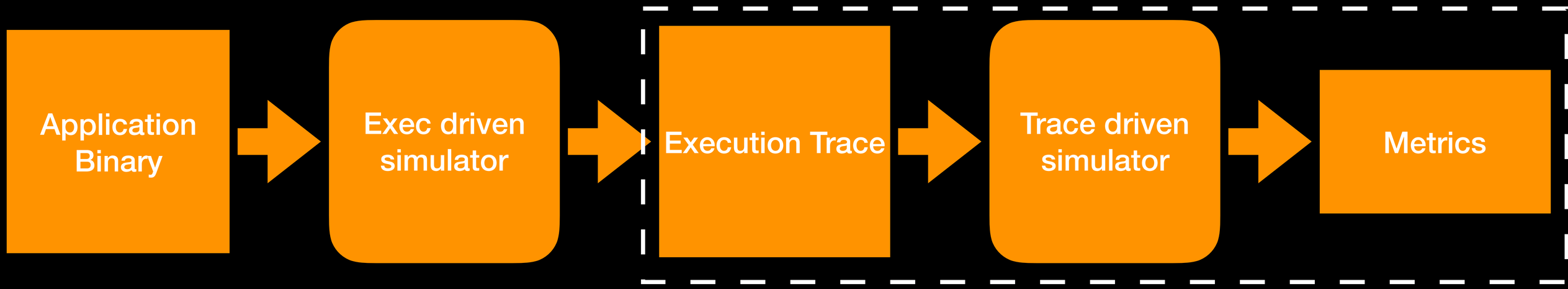


```
void execute(int32_t* instructions) {
    // Declare a function pointer that takes no arguments
    void (fp*)(void);

    // Set pointer to the first instruction
    fp = instructions;

    // Run! (And hope the last instruction in the list returns)
    fp();
}

int main() {
    int32_t instructions[] = {
        0x3FE9,
        0xA701,
        0xEF02,
        0x8FF0};

    execute(instructions);
}
```

Back to business…

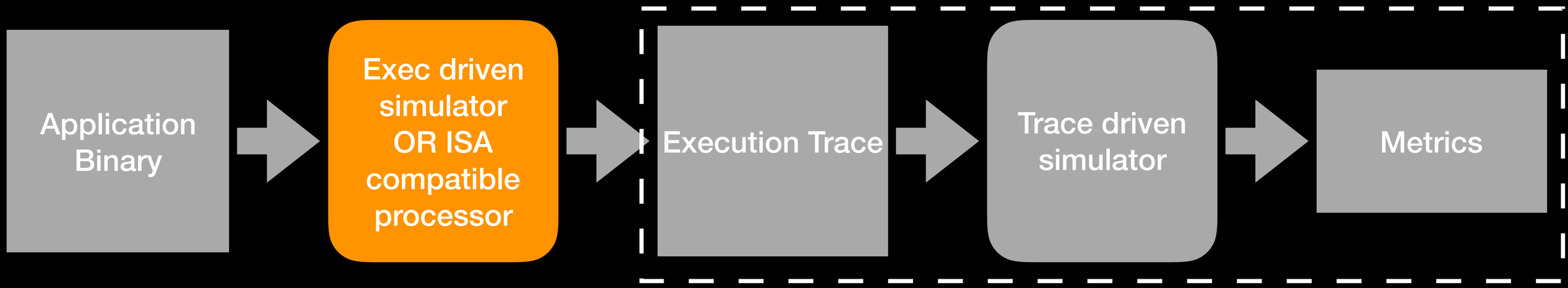
Execution- vs Trace-driven

Execution- vs Trace-driven



Why trace-driven?

Execution- vs Trace-driven



Trace-driven simulation

Advantages

- Trace collection only required once
- Trace collection can be done with ISA compatible processor
- Trace simulator does not need to simulate all instructions, can skip ahead in trace if not implemented

Disadvantage

- Trace file can become huge for large applications
- Cannot speculatively execute code (trace is fixed)
  - But then there are elastic traces
  - Check: Exploring System Performance using Elastic Traces: Fast, Accurate and Portable by Radhika Jagtap et al.

Mixing simulation strategies

For best of both worlds

Mixing simulation strategies

Direct execution

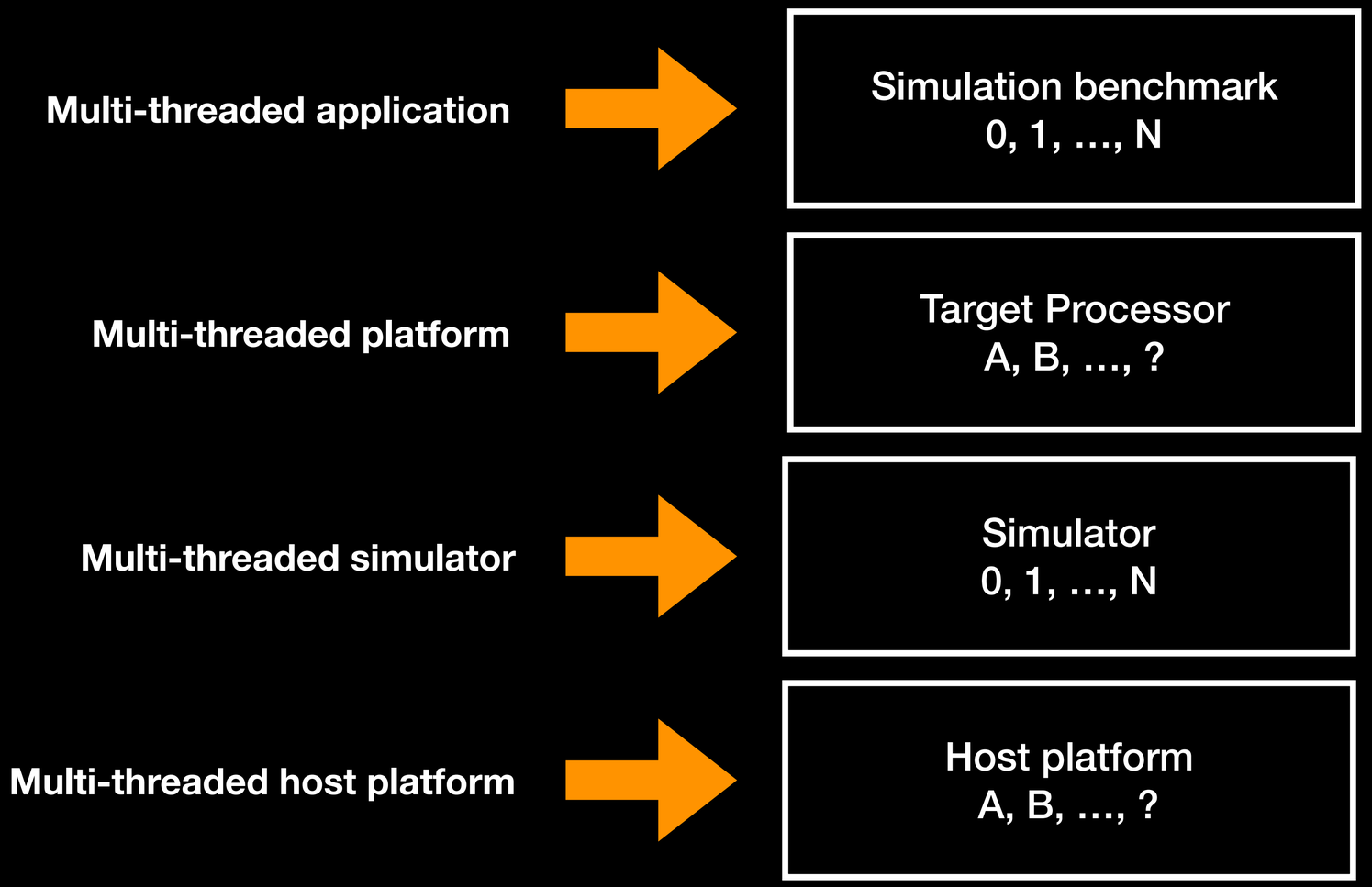
- Parts run directly on the host (e.g. with dynamic translation such as QEMU)
- Other parts with cycle accurate simulation

Use case

Interested in memory accesses and memory behavior. Execute only loads and stores on the simulator, emulate the rest directly on the host machine.

Simulating a multi-processor

Simulating a multi-processor



Does every layer have to have the same number of threads/cores?

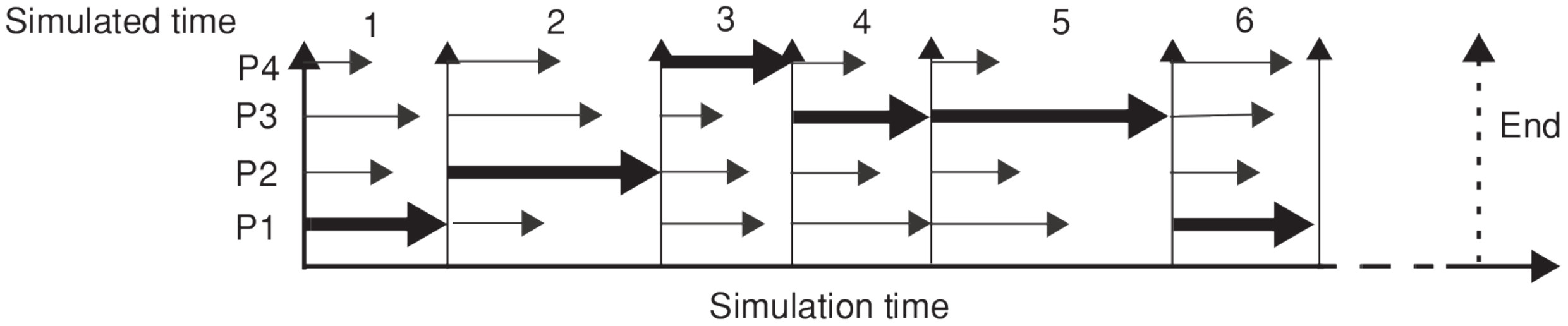
Parallel simulation techniques

Mixing simulation strategies

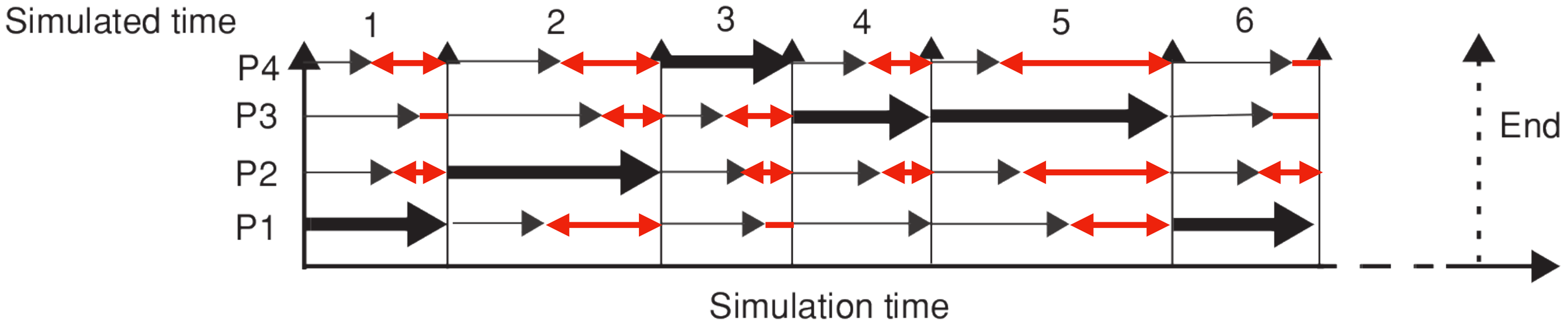
- Discrete event simulation
- Quantum simulation
- Slack simulation

- The textbook implicitly assumes the smallest hardware block that can be mapped to a simulator thread is a full target core.
- Holds for almost all real-world simulators (not RTL simulators), which severely limits the parallelism

A logical choice for a simulator “time step” is one cycle for the slowest core.



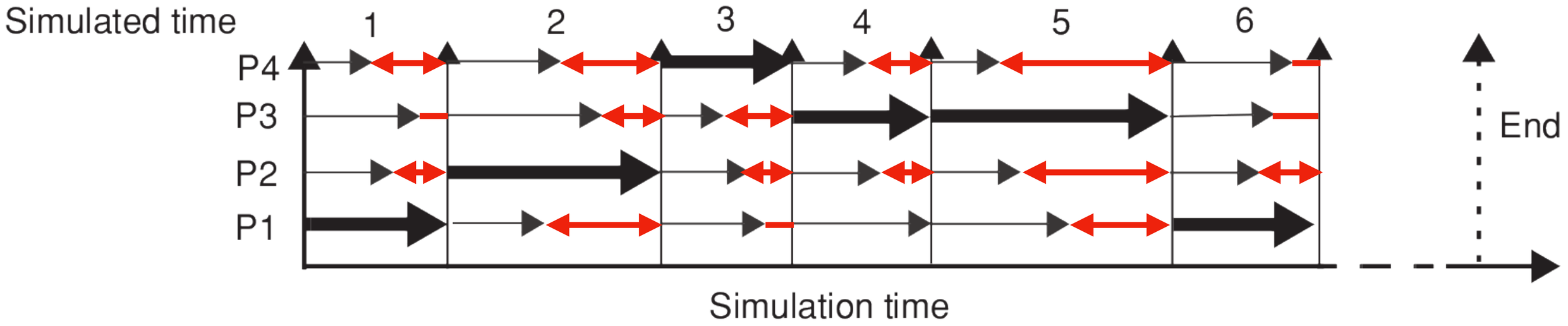
A logical choice for a simulator “time step”
is one cycle for the slowest core.



Disadvantage: under utilisation of the host platform
if threads are idle for synchronisation

Wait!
Is that even true?

This example assumes every target core is mapped to a single host core

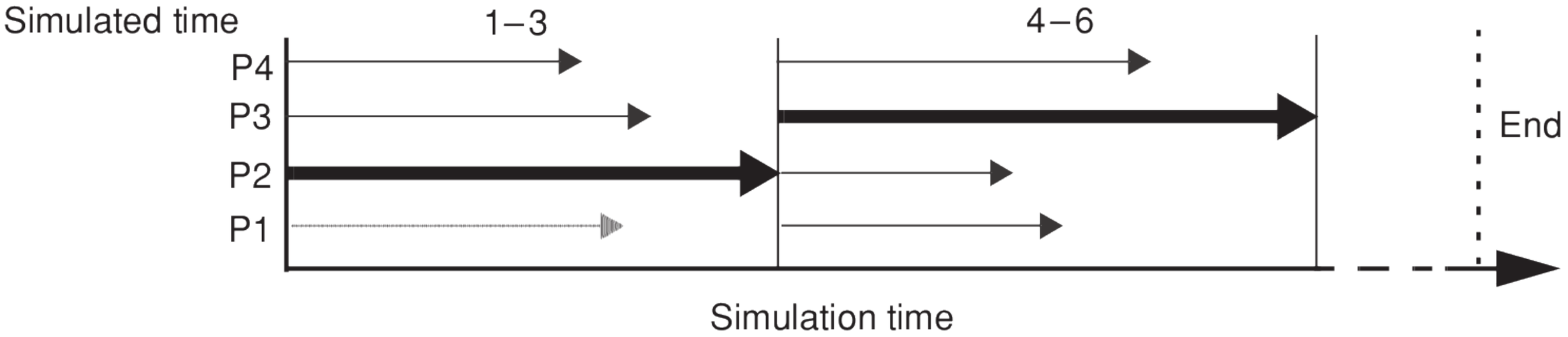


There is

absolutely no relation

between the number of target cores and the number of host cores

Quantum simulation
Synchronise threads at larger time steps (e.g. 3 cycles)



Quantum simulation

Synchronise threads at larger time steps (e.g. 3 cycles)

Advantage

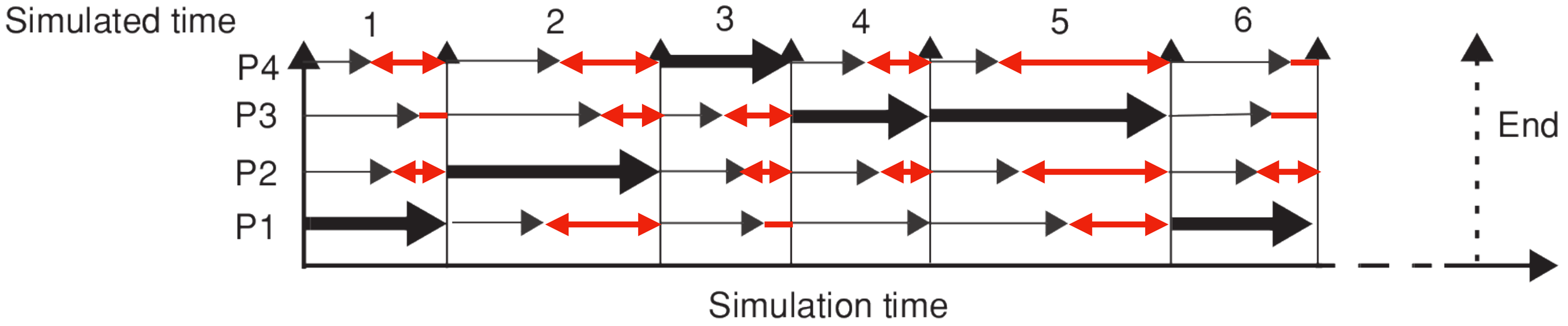
Utilisation improves, because the variation of processing is amortised over longer sections of simulation

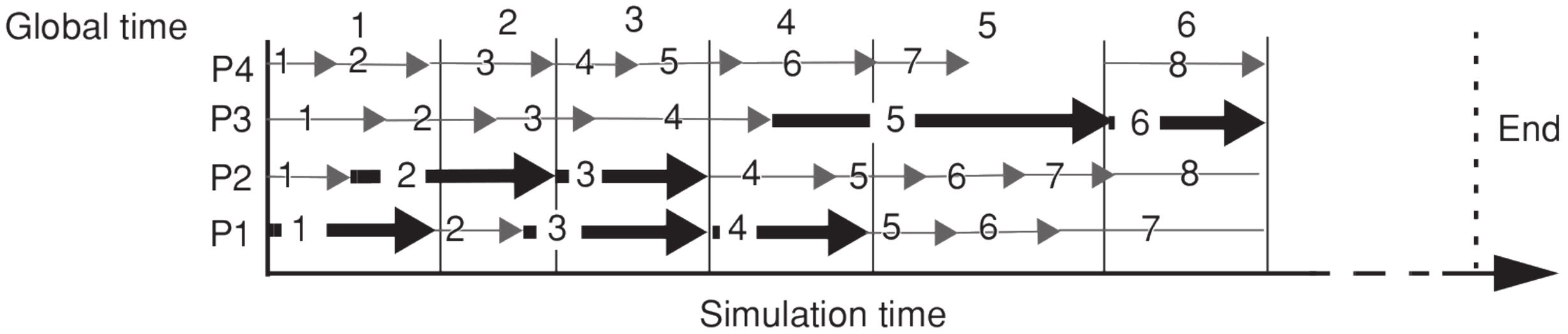
Disadvantage

No longer cycle accurate

Slack simulation

Instead of waiting in the red areas, use slack to process ahead





Slack simulation

Side-effect

Drift: the cores might be at a different moment in time and could drift apart

Mitigation

Allow for a maximum drift and synchronise when this value is exceeded

Generally allows for less synchronisation

All considering, simulation is still terribly slow…

Workload sampling

Summary

Summary

- Why Simulators
  - More accurate than models
  - Cheaper than building hardware
- Simulation detail
  - Full-System vs User-level
  - Functional vs Cycle Accurate (micro-arch.) vs Gate-Level
  - Execution- vs Trace-driven
- (Fast) Multiprocessor Simulation
  - Discrete event
  - Quantum
  - Slack
- Workload Sampling
- Summary

Read more in
chapter 9 of the text book

Thank you!

5SIA0 - Glenn Bergmans

GEM5 assignment

Optimising an EEG application

5SIA0 - Glenn Bergmans

Step-through

Step-through

- There will be a cookbook that you should follow
  - Install VirtualBox and run tools
  - First benchmark and run with single core platform
  - Extend platform to a multi-core platform
- Run your own design experiments
  - Come up with an optimal solution

What is optimal?

What is optimal?

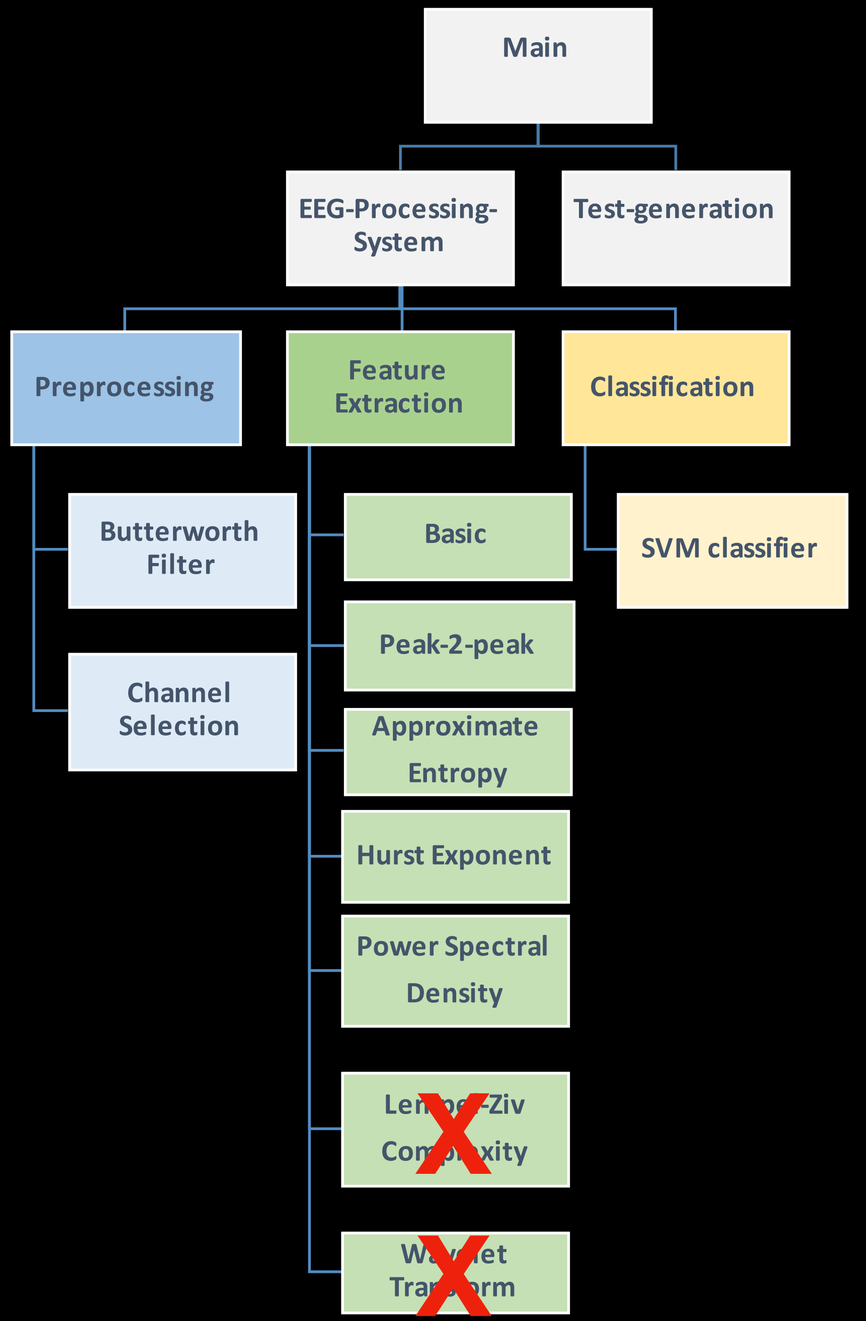
- Best performance with lowest energy
  - Optimise power by:
    - Reducing cache sizes
    - Minimising memory accesses
  - Optimise performance by:
    - Parallelisation of C-code
    - Distributing code optimally over cores
    - Improving algorithm

What is optimal?

- Parallelisation with OpenMP
  - Simple pragma based C add-on to
    to aid in parallelising your code
- You get a pre-compiled version of GEM5 and a cross-compiler for ARM platforms

More information in the cookbook!

Application overview


Main
EEG-Processing-System
Test-generation
Preprocessing
Feature Extraction
Classification
Butterworth Filter
Channel Selection
Basic
Peak-2-peak
Approximate Entropy
Hurst Exponent
Power Spectral Density
Lempel-Ziv Complexity
Wavelet Transform
SVM classifier

Deadline: January 7th