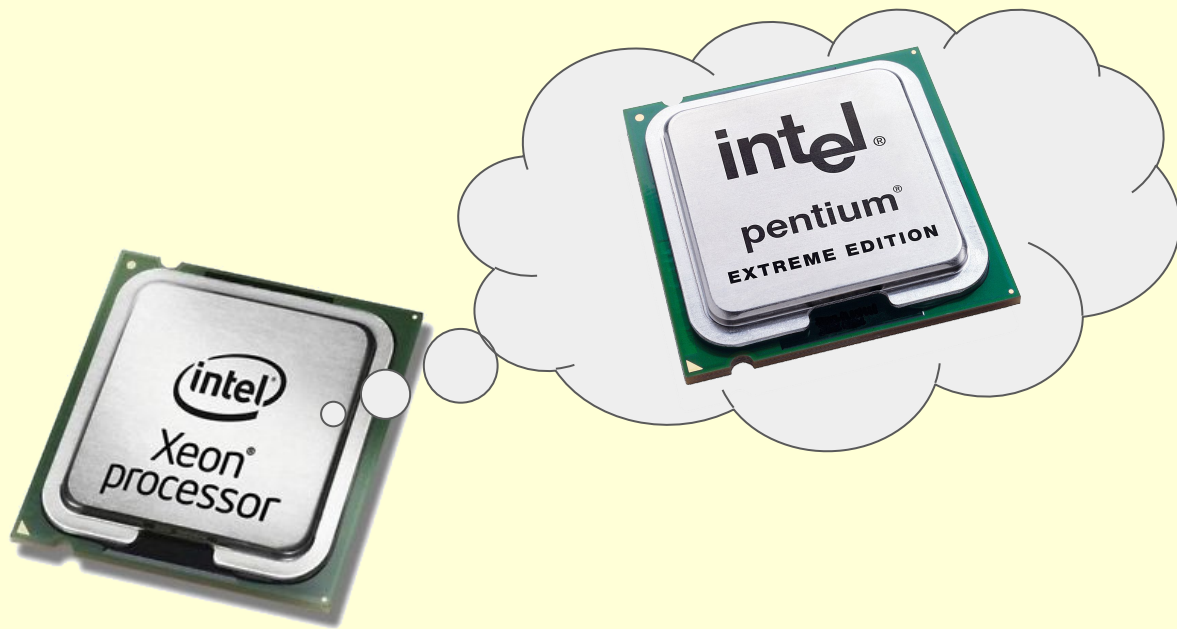


Simulators

5SIA0

Processors Processing Processors

The meta-lecture

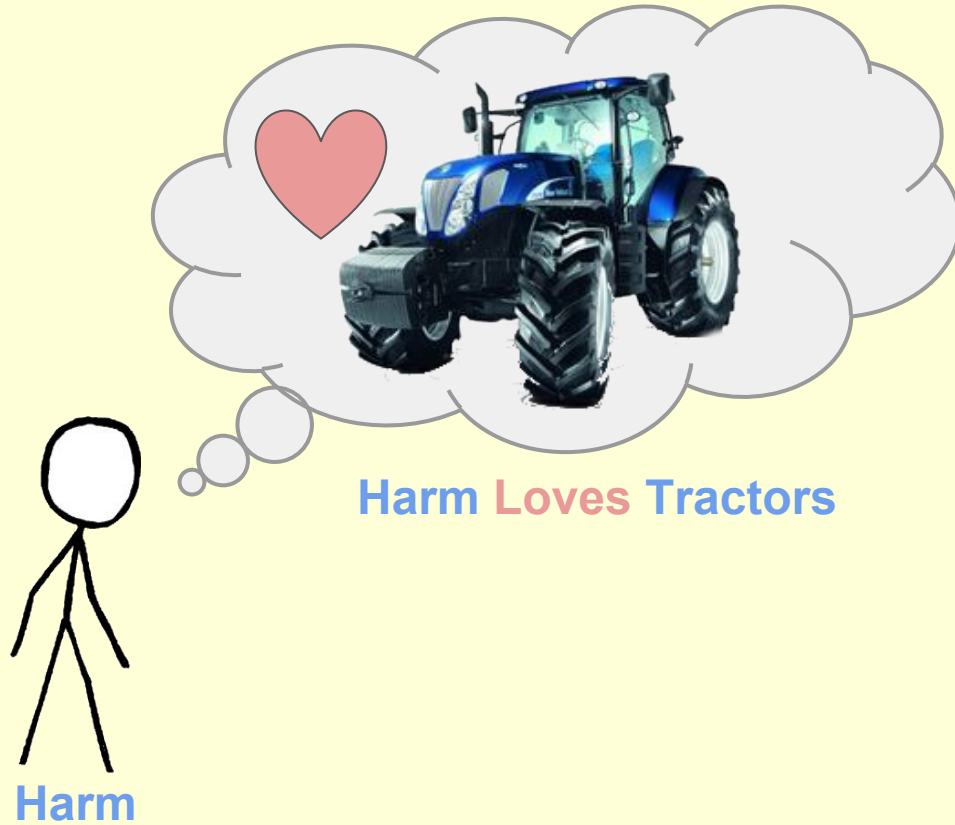


Why Simulators?

Your Friend Harm



Why Simulators?



Why Simulators?

Unfortunately for Harm
you need to go outside
to drive tractors



Harm

The outside world



Why Simulators?

And the outside world
is filled with dangers



Harm

The outside world



Why Simulators?

And the outside world
is filled with dangers



Harm

The outside world



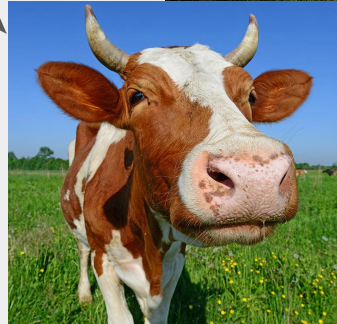
Why Simulators?

And the outside world
is filled with dangers



Harm

The outside world



Scary Animals!

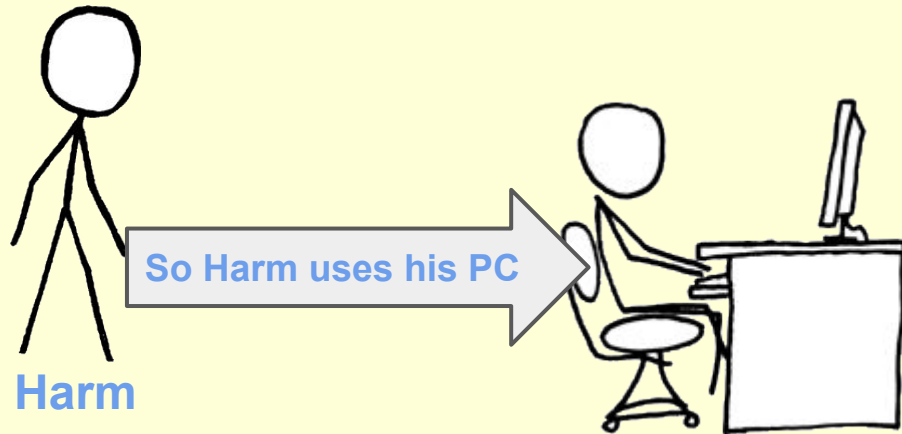


Why Simulators?



Harm

Why Simulators?



Why Simulators?



Harm

Why Simulators?

Oh No! My PC is
too slow to run
Farming Simulator



Harm

Why Simulators?

Oh No! My PC is
too slow to run
Farming Simulator



Why Simulators?

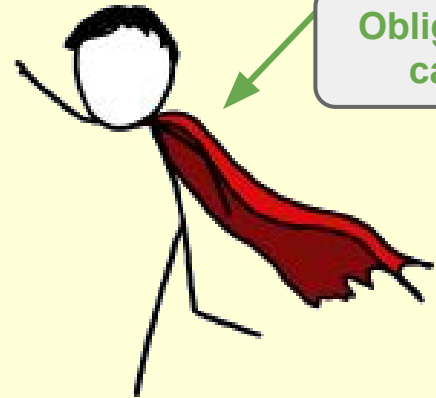
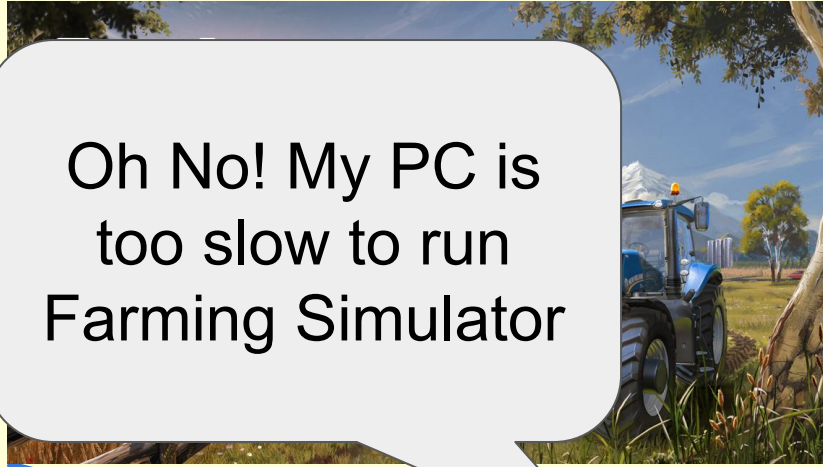
Oh No! My PC is
too slow to run
Farming Simulator

Stand back!
I'm a **computer
architect**!

Obligatory
cape

Harm

You



How to help Harm?

Of course you
have **many ideas**
on how to
speed-up Harms
computer.

But **which ones**
should you apply?



You

Design Space Exploration Options

Design Space Exploration Options

- Buy (or build) all hardware options



Design Space Exploration Options

- Buy (or build) all hardware options



Gee that
sounds
expensive...

Design Space Exploration Options

- Buy (or build) all hardware options
- Use analytical models



Design Space Exploration Options

- Buy (or build) all hardware options
- Use analytical models



How reliable
is that?

Design Space Exploration Options

- Buy (or build) all hardware options
- Use analytical models
- Simulate the design points!



Design Space Exploration Options

- Buy (or build) all hardware options
- Use analytical models
- Simulate the design points!



Hey, I **like** simulators,
That sounds promising :)

What to simulate for?

What to simulate for?

- Performance
- Energy
- Power (!=Energy)
- Thermal

What to simulate for?

- Performance
- Energy
- Power (!=Energy)
- Thermal

What details to simulate?

What to simulate for?

- Performance
- Energy
- Power (!=Energy)
- Thermal

What details to simulate?

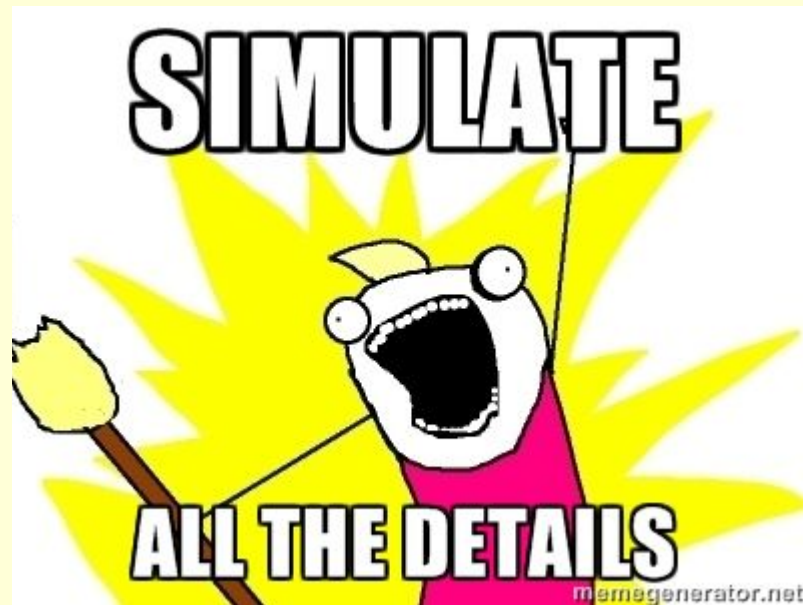
- Cycle accurate vs Functionality
- Caches
- Full operating system
- Disk accesses
- Background tasks
- ...

What to simulate for?

- Performance
- Energy
- Power (**!=Energy**)
- Thermal

What details to simulate?

- Cycle accurate vs Functionality
- Caches
- Full operating system
- Disk accesses
- Background tasks
- ...



All the details: RTL Simulation

All the details: RTL Simulation

Simulate at **gate level**:

All the details: RTL Simulation

Simulate at **gate level**:

- modelsim/questasim (Mentor)
- ncsim (Cadence)
- VCS (Synopsys)
- Icarus Verilog (Open Source!)
- ...

All the details: RTL Simulation

Simulate at **gate level**:

- modelsim/questasim (Mentor)
- ncsim (Cadence)
- VCS (Synopsys)
- Icarus Verilog (Open Source!)
- ...

Advantages:

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

All the details: RTL Simulation

Simulate at **gate level**:

- modelsim/questasim (Mentor)
- ncsim (Cadence)
- VCS (Synopsys)
- Icarus Verilog (Open Source!)
- ...

Advantages:

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

Disadvantage:

- **Horribly** slow for realistic designs

All the details: RTL Simulation

Simulate at **gate level**:

- modelsim/questasim (Mentor)
- ncsim (Cadence)
- VCS (Synopsys)
- Icarus Verilog (Open Source!)
- ...

Advantages:

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

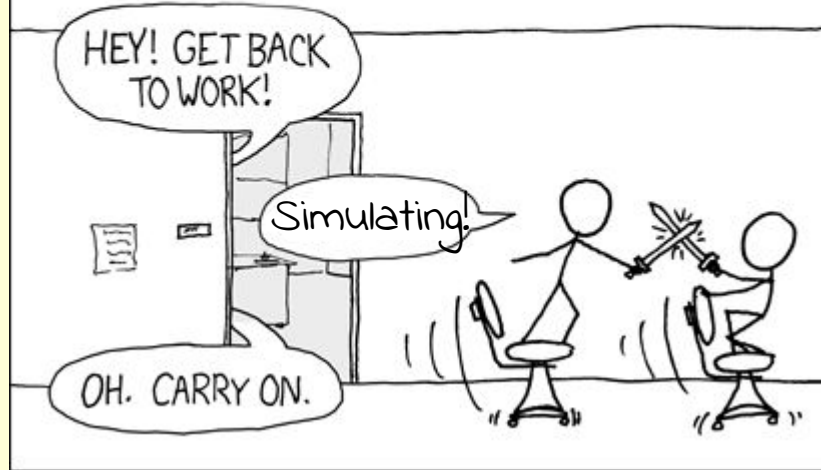
Nvidia GPU with > 1 Billion transistors
Small tests take **over 8 hours!** [1]

Disadvantage:

Horribly slow for realistic designs

THE #1 Computer Architect EXCUSE
FOR LEGITIMATELY SLACKING OFF:

"MY CODE'S Simulating"



Slightly less horribly slow: Hardware Emulation



RTL
description
of Target
Architecture

Slightly less horribly slow: Hardware Emulation

RTL
description
of Target
Architecture

Synthesize for
FPGA (**slow**)



Slightly less horribly slow: Hardware Emulation

RTL
description
of Target
Architecture

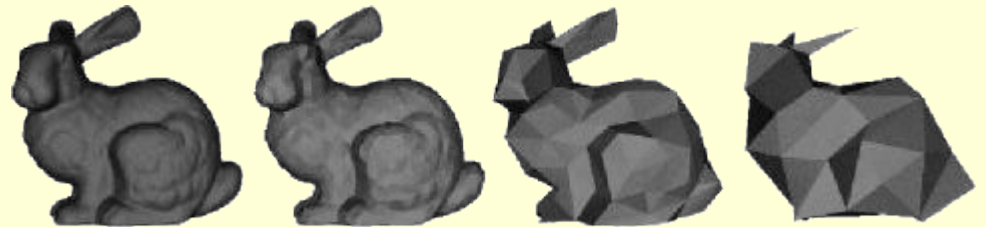
Synthesize for
FPGA (**slow**)



Emulate on FPGA (**fast!**)
Note: instrumentation
required to get detailed
information out!

Levels of detail in Simulation

- Full-System versus User-level
- Cycle Accurate versus Functional
- Execution- versus Trace-driven

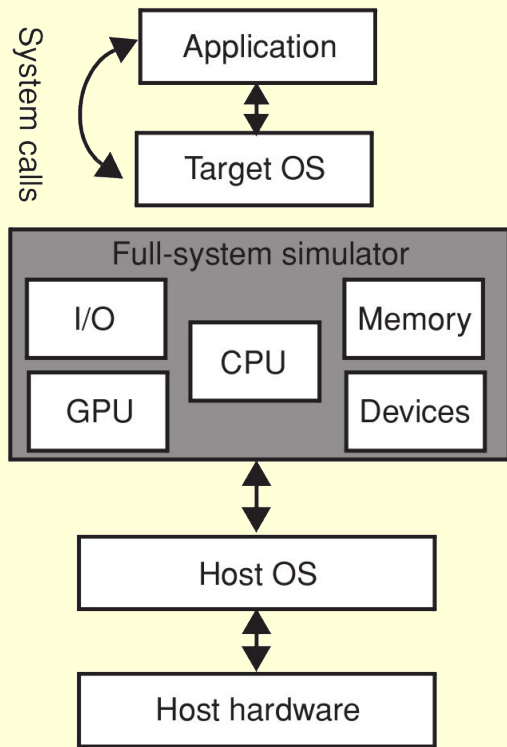


Full-system versus User-Level

To OS or not to OS?

Full-system versus User-Level

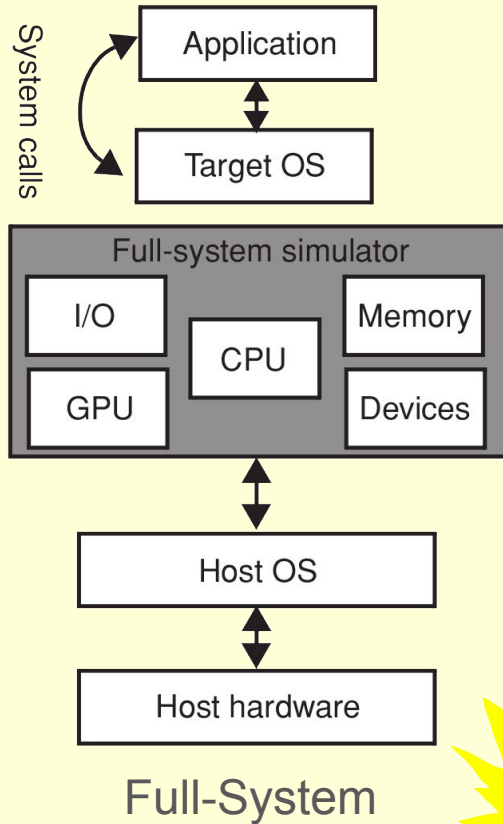
To OS or not to OS?



Full-System

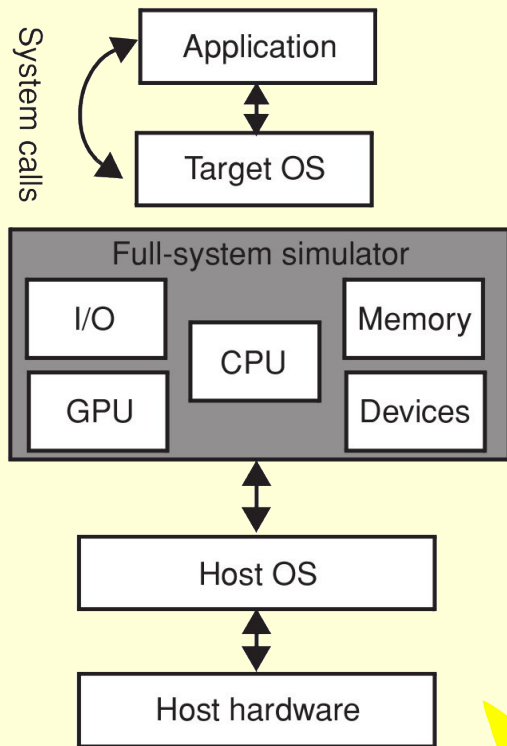
Full-system versus User-Level

To OS or not to OS?

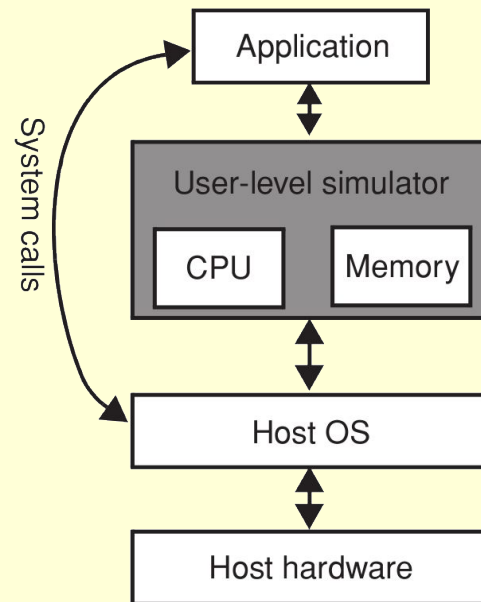


Full-system versus User-Level

To OS or not to OS?



Full-System



User-Level



User-Level

Famous example: **Simple Scalar** [1]

Advantages

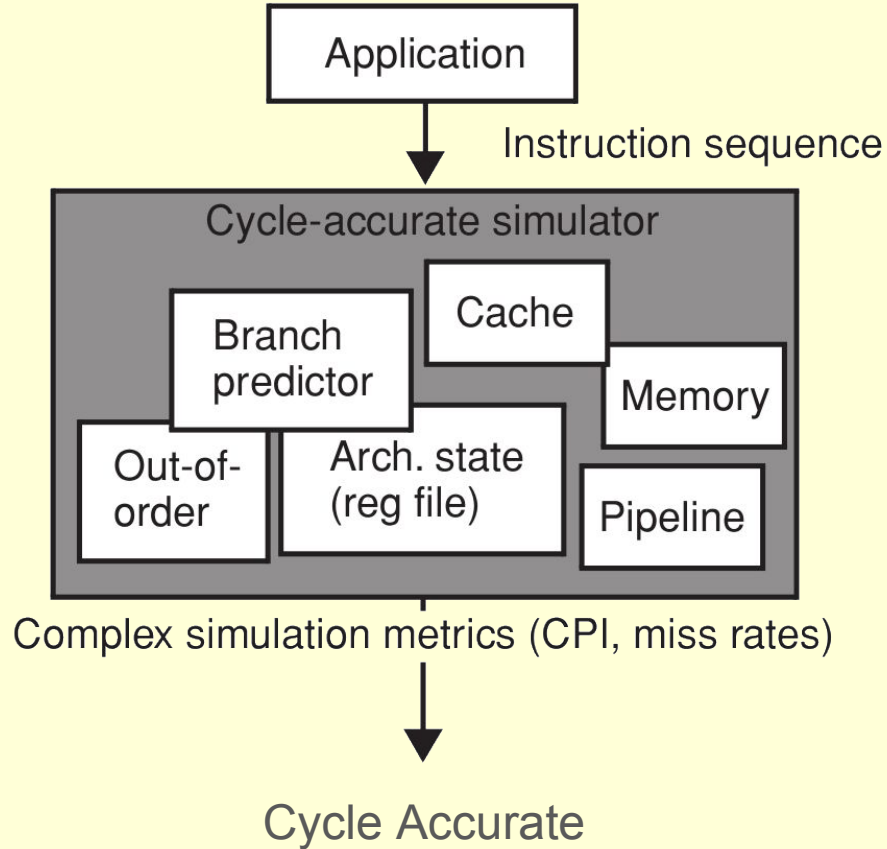
- Fast to develop and update to new architectures
- Usually 'accurate enough'

Disadvantages

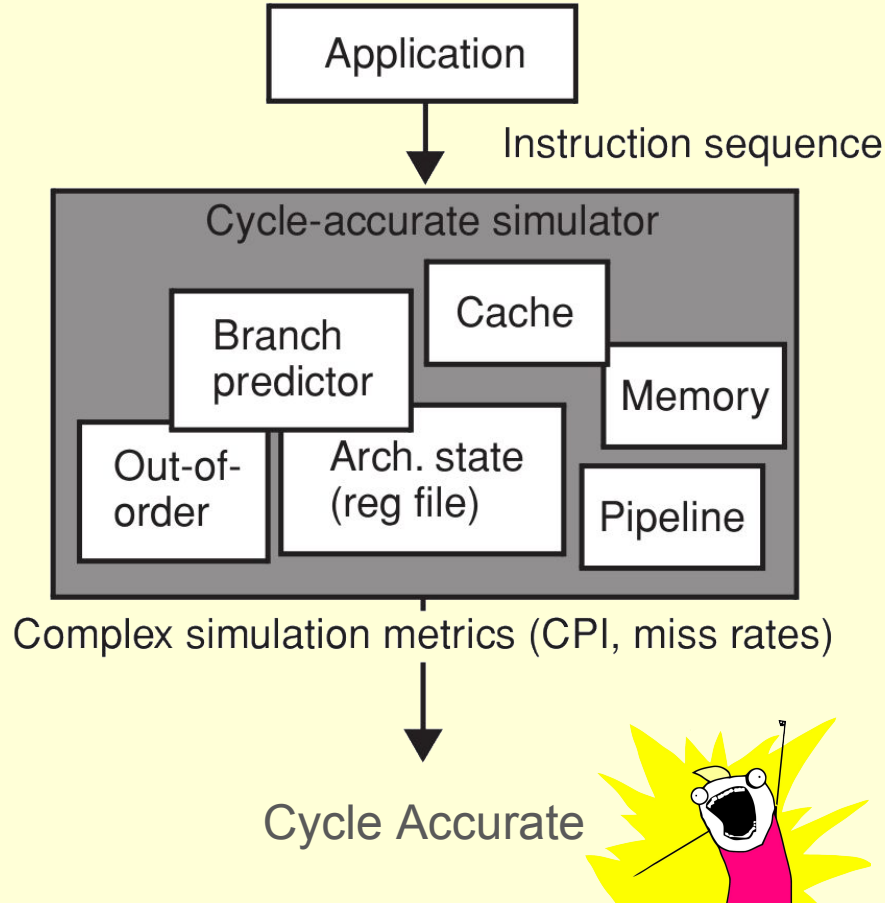
- Any time spent in the OS is not modelled accurately. Can have severe impact, database applications spent 20-30% of their time in OS mode.

Cycle Accurate versus Functional

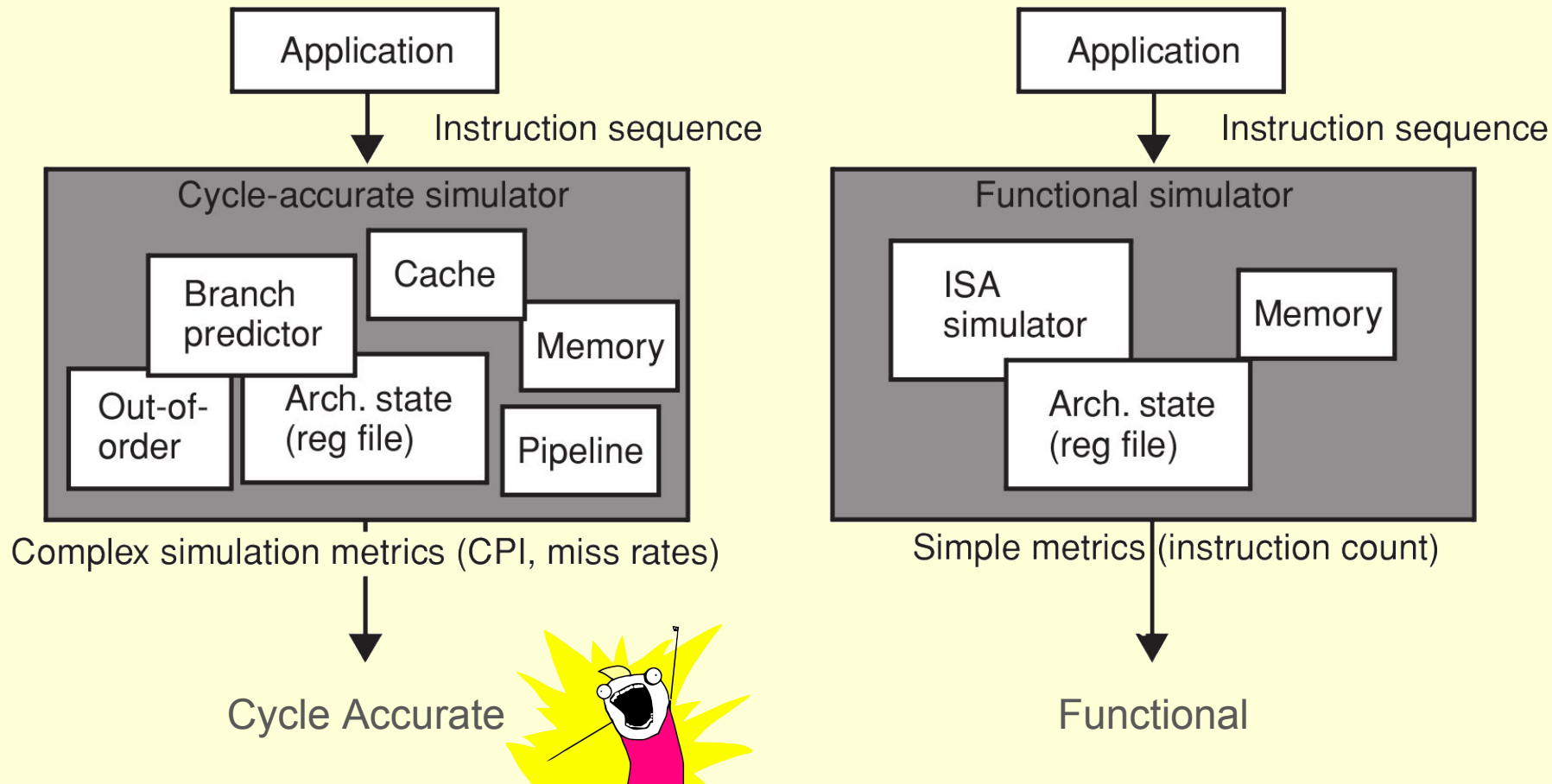
Cycle Accurate versus Functional



Cycle Accurate versus Functional



Cycle Accurate versus Functional



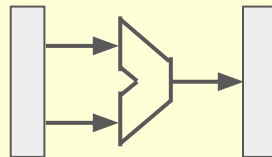
Cycle Accurate versus Functional

Functional - **no/limited** model of the micro architecture

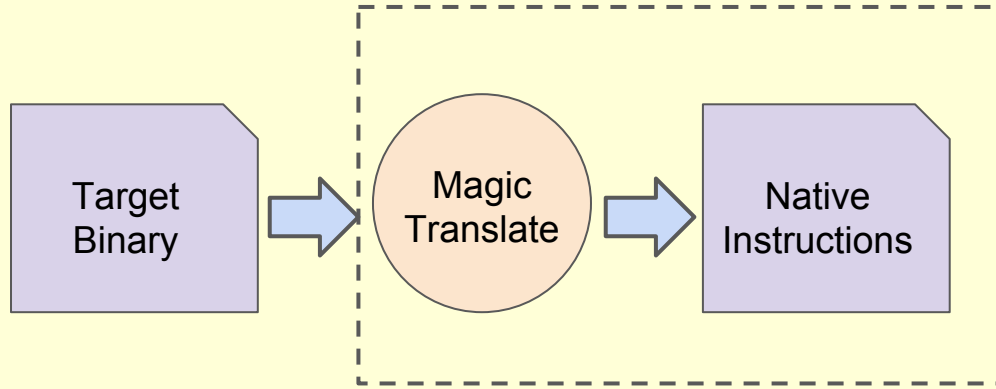
- An (add) instruction of the target can be translated to an (add) instruction on the host, and be simulated that way.
- Example 1: Simple Scalar sim-fast
- Example 2: QEMU, Full-system emulator using **dynamic translation**

Cycle Accurate - **includes** model of the micro architecture

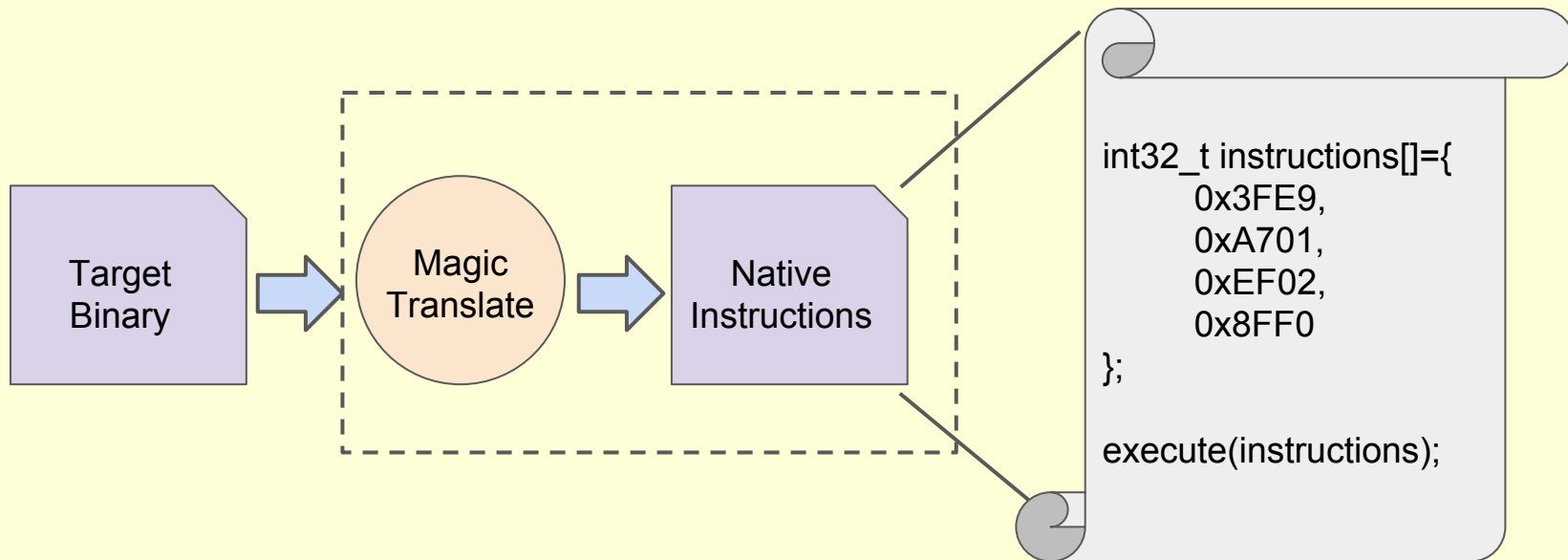
- Block resources in the pipeline when instruction executes
- Use target branch predictor scheme
- Out-of-order execution
- Example: Simple Scalar sim-outorder



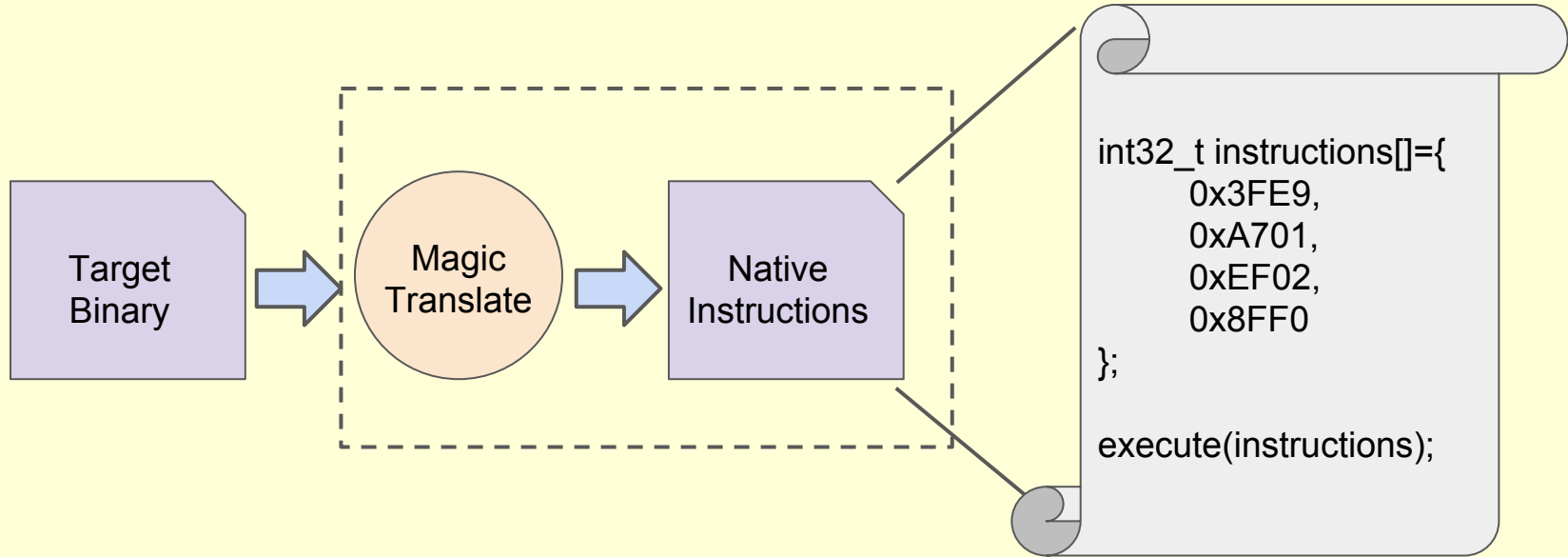
Intermezzo - Internals of dynamic translation



Intermezzo - Internals of dynamic translation



Intermezzo - Internals of dynamic translation



Question

Implement the execute function in regular C

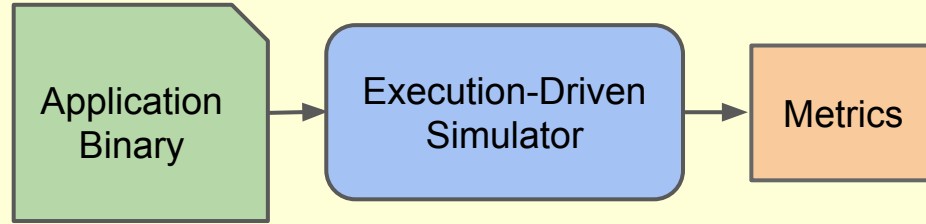
Intermezzo - Internals of dynamic translation

```
void execute(int32_t* instructions){  
    //declare a pointer to a function that returns void  
    // and has no arguments  
    void (fp*)(void);  
  
    //set the function pointer to the first instruction  
    fp=instructions;  
  
    //call the function  
    //Note: make sure the last instruction in the list returns  
    fp();  
}
```

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

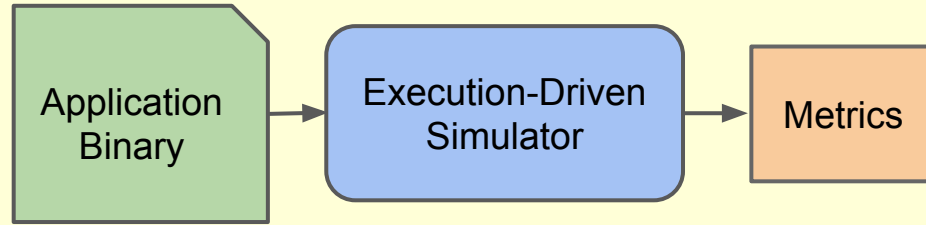
Execution- versus Trace-driven

Execution- versus Trace-driven

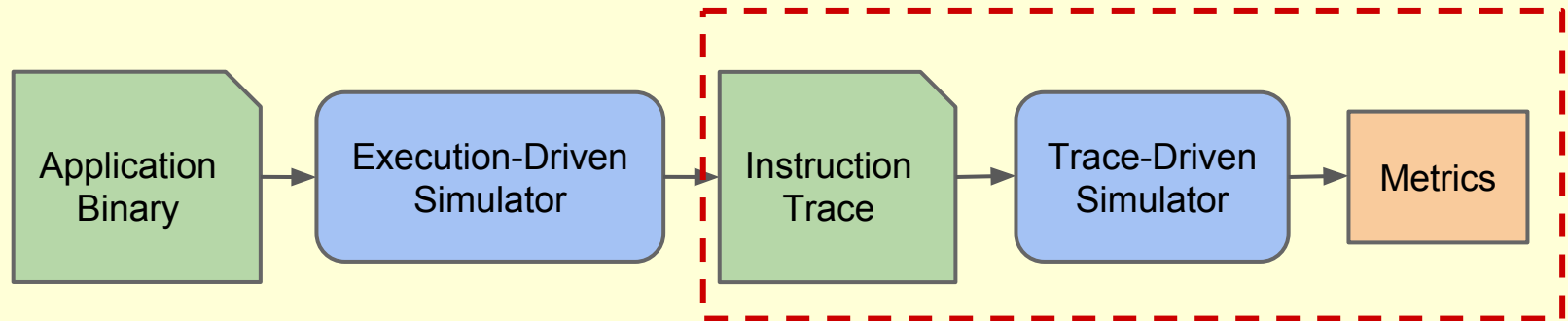


Execution Driven: Application executes on simulator

Execution- versus Trace-driven

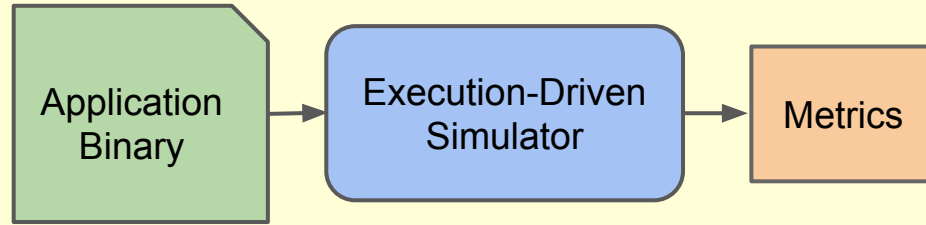


Execution Driven: Application executes on simulator



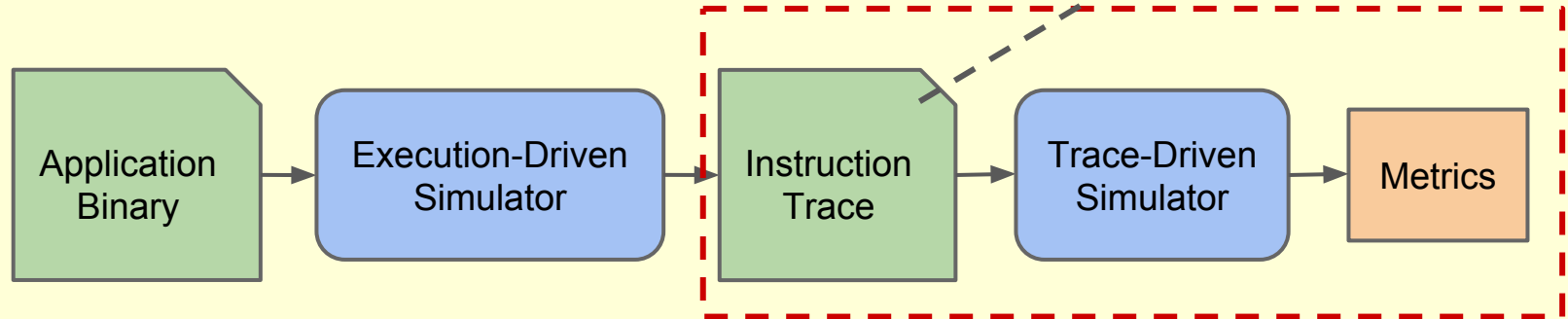
Trace Driven: simulator uses trace as input

Execution- versus Trace-driven



Execution Driven: Application executes on simulator

```
mov edx,len
mov ecx,msg
mov ebx,1
mov eax,4
int 0x80
```

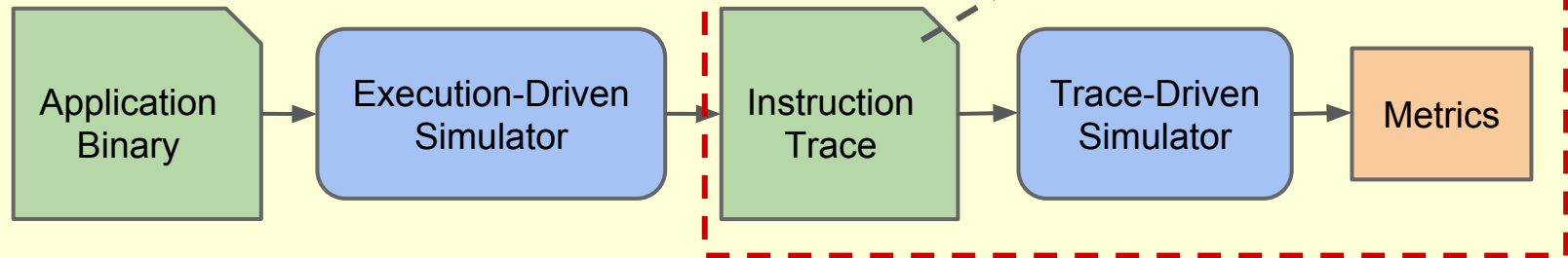


Trace Driven: simulator uses trace as input

Execution- versus Trace-driven

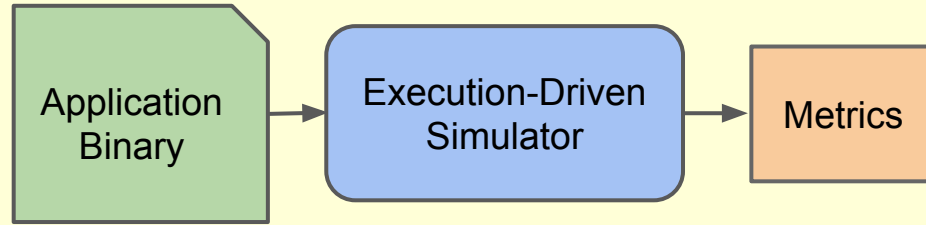


Why would a sane person do this?

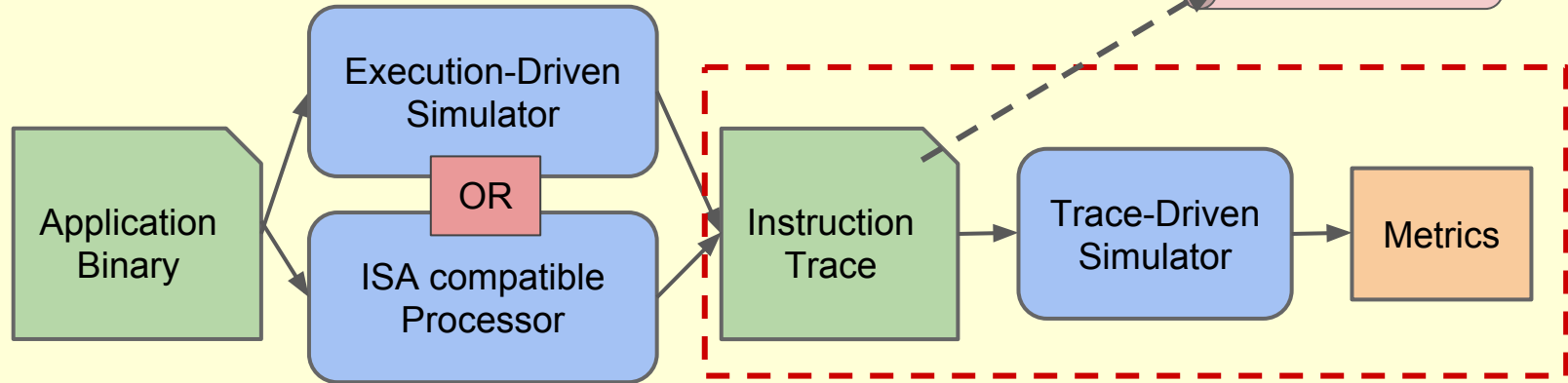


Trace Driven: simulator uses trace as input

Execution- versus Trace-driven



Execution Driven: Application executes on simulator



Trace Driven: simulator uses trace as input

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

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

Disadvantages

- Cannot speculatively execute code (trace is fixed)
- Trace file can become huge for large applications (hundreds of GBs)

Mixing Simulation Strategies

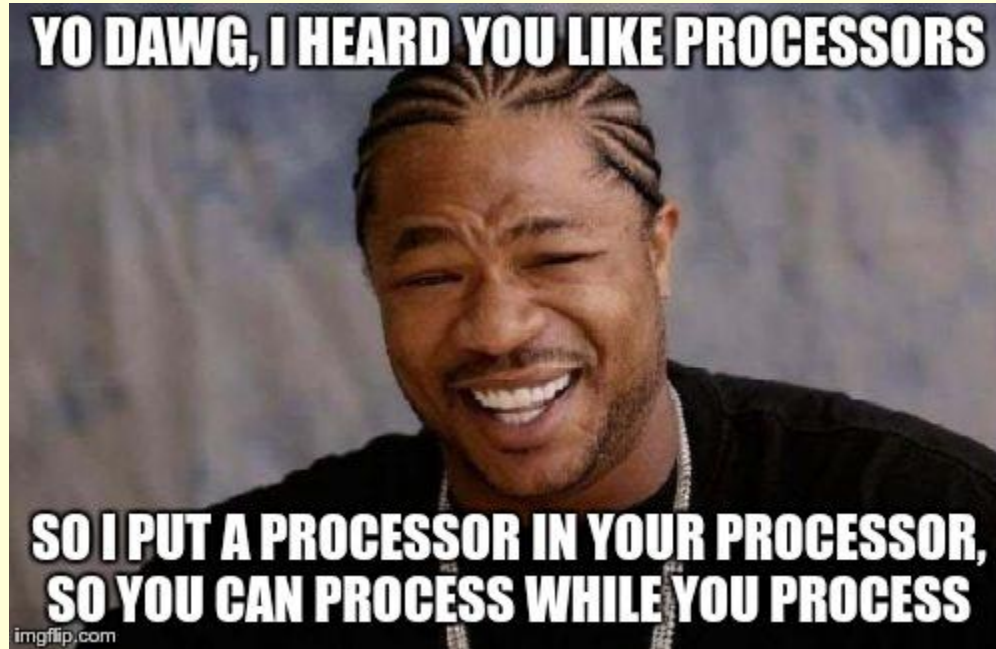
Direct-execution

- Parts execute directly on the host (e.g. using dynamic translation such as QEMU)
- Other parts are executed on cycle accurate simulation

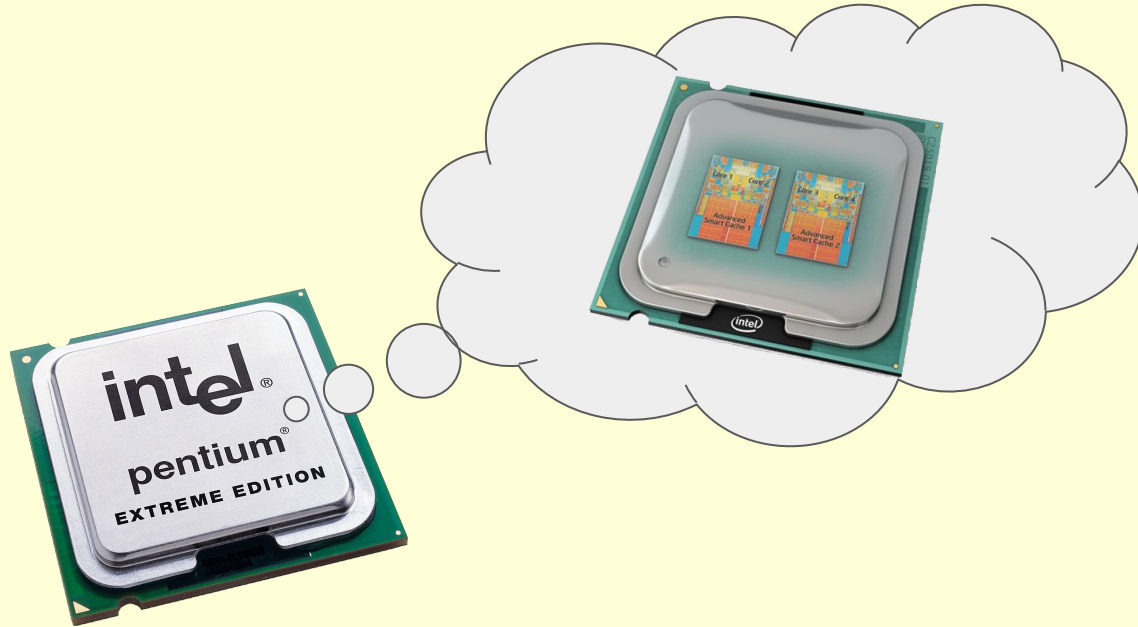
Use case:

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

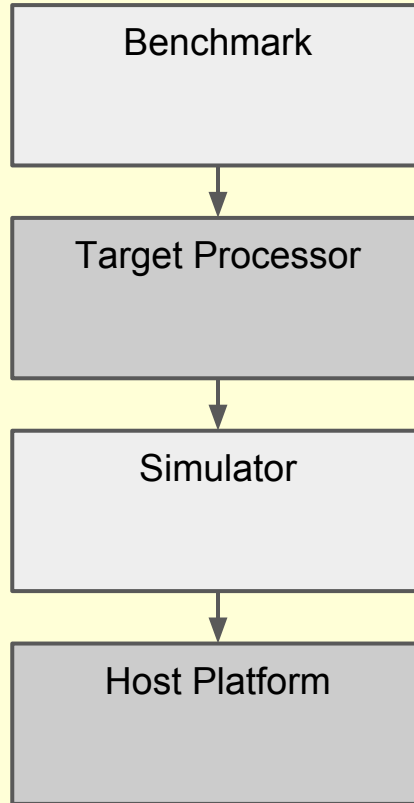
Simulation in the Multiprocessor Era



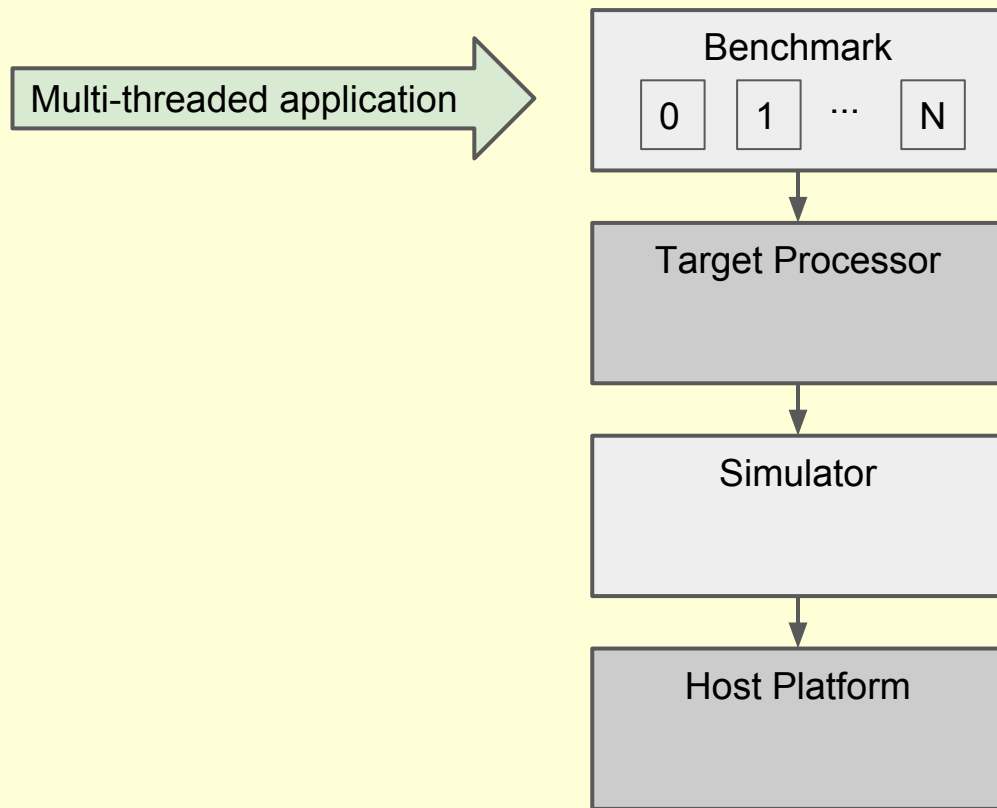
Simulation in the Multiprocessor Era



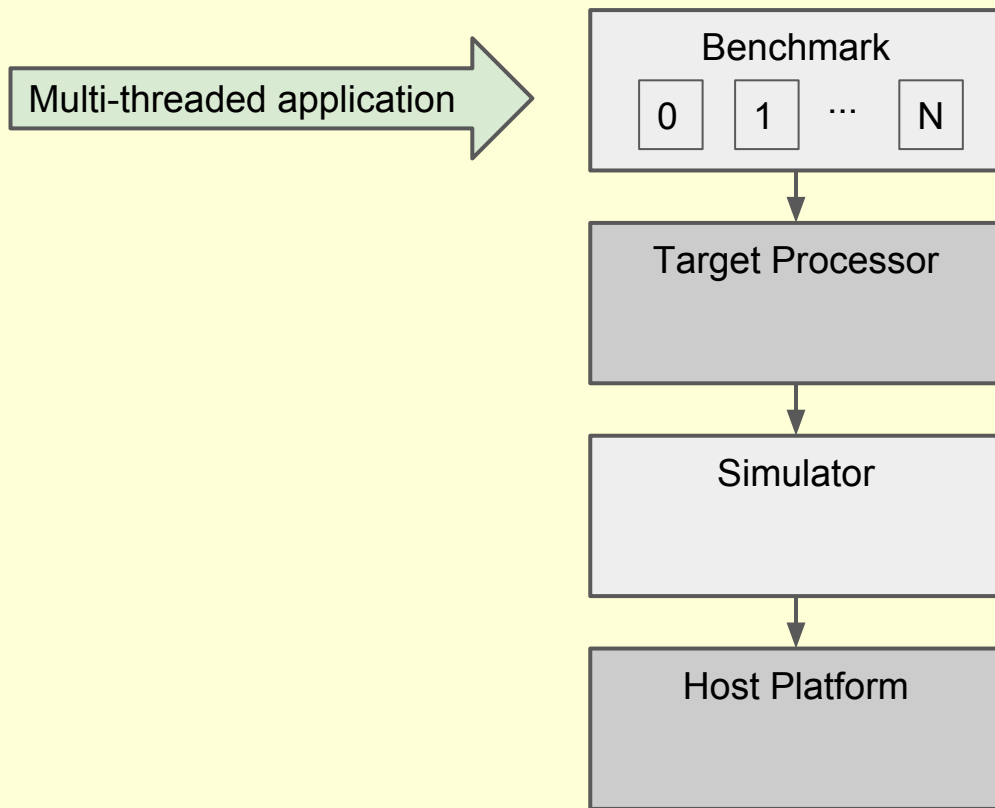
Parallelisation in all levels of the simulation stack



Parallelisation in all levels of the simulation stack



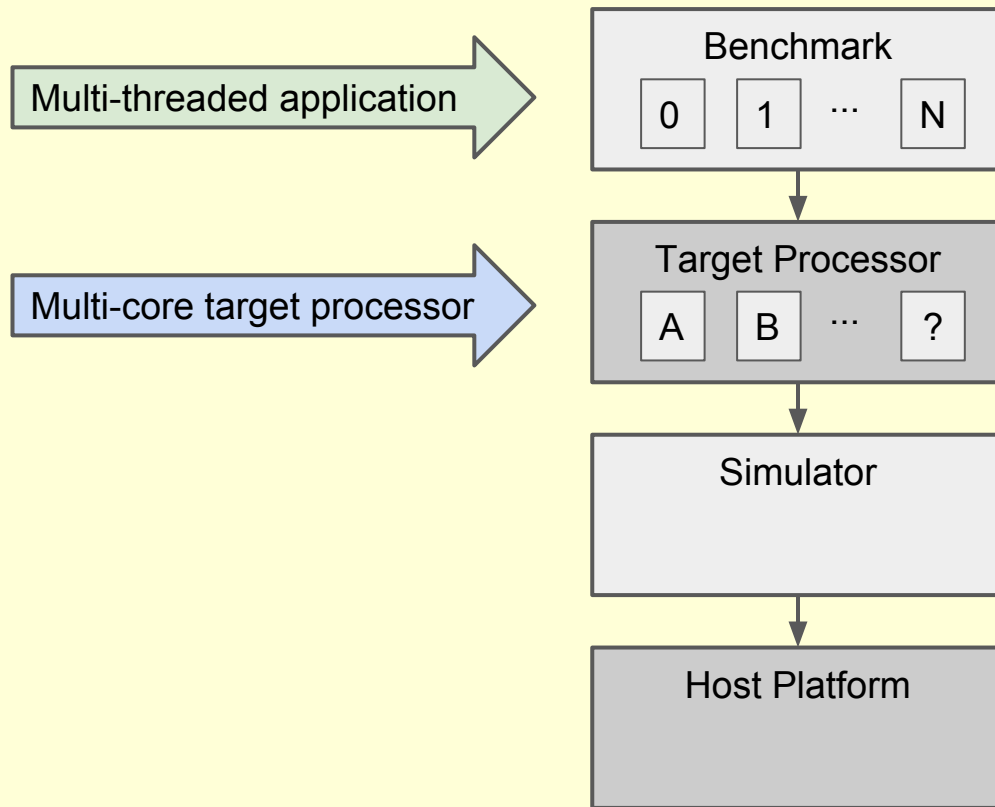
Parallelisation in all levels of the simulation stack



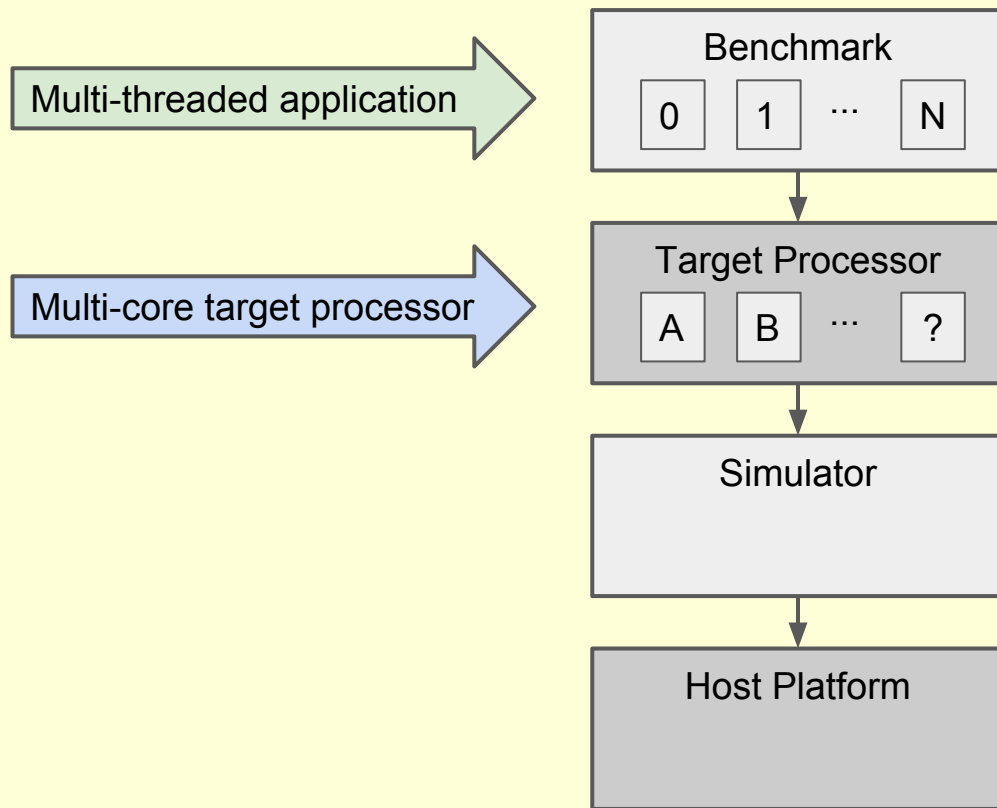
A **multi-threaded application** running on a **single core target processor**.

Question: Does this make sense?

Parallelisation in all levels of the simulation stack



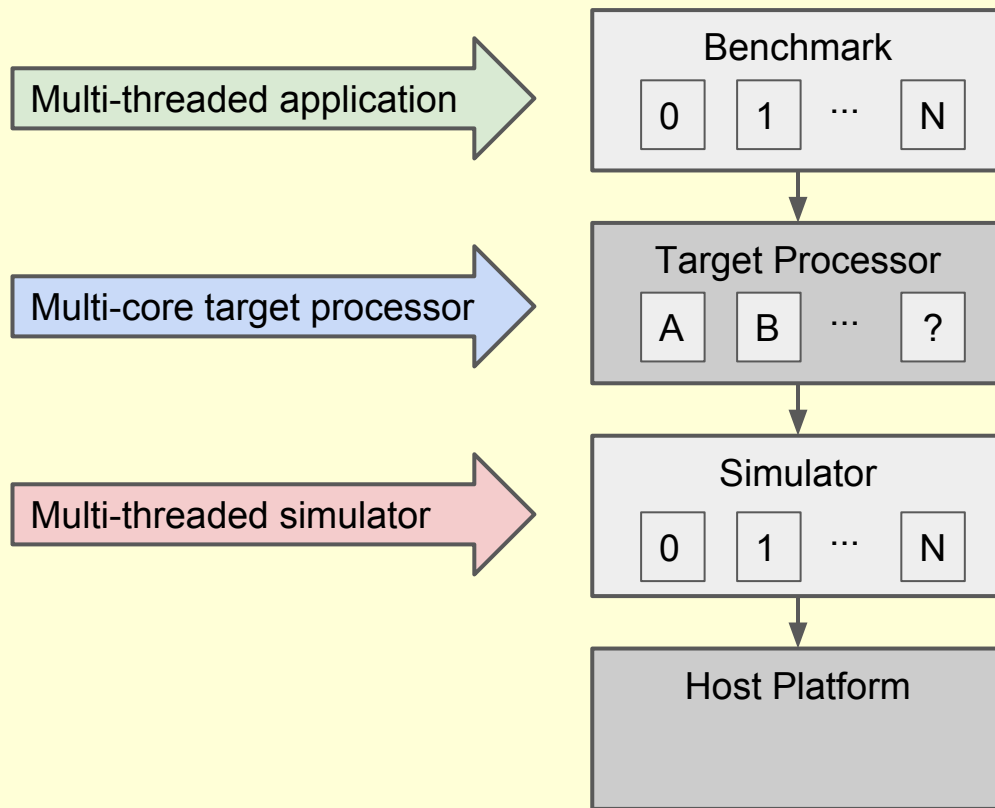
Parallelisation in all levels of the simulation stack



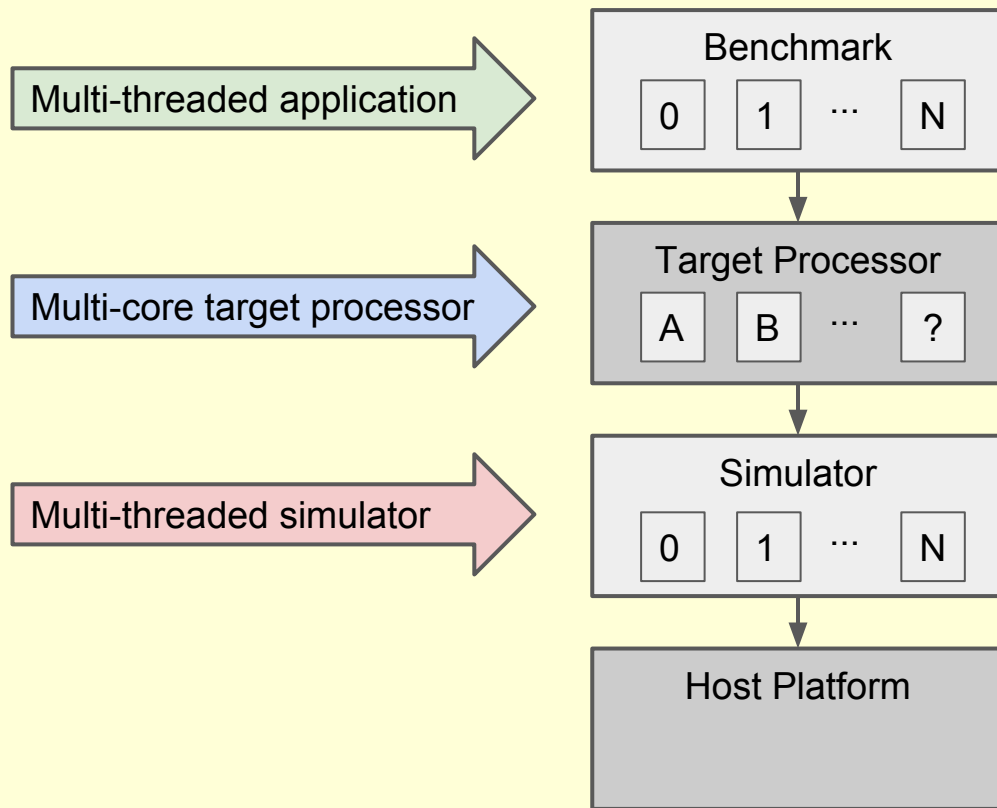
A multi-core processor running on a single threaded simulator.

Question: Does this make sense?

Parallelisation in all levels of the simulation stack



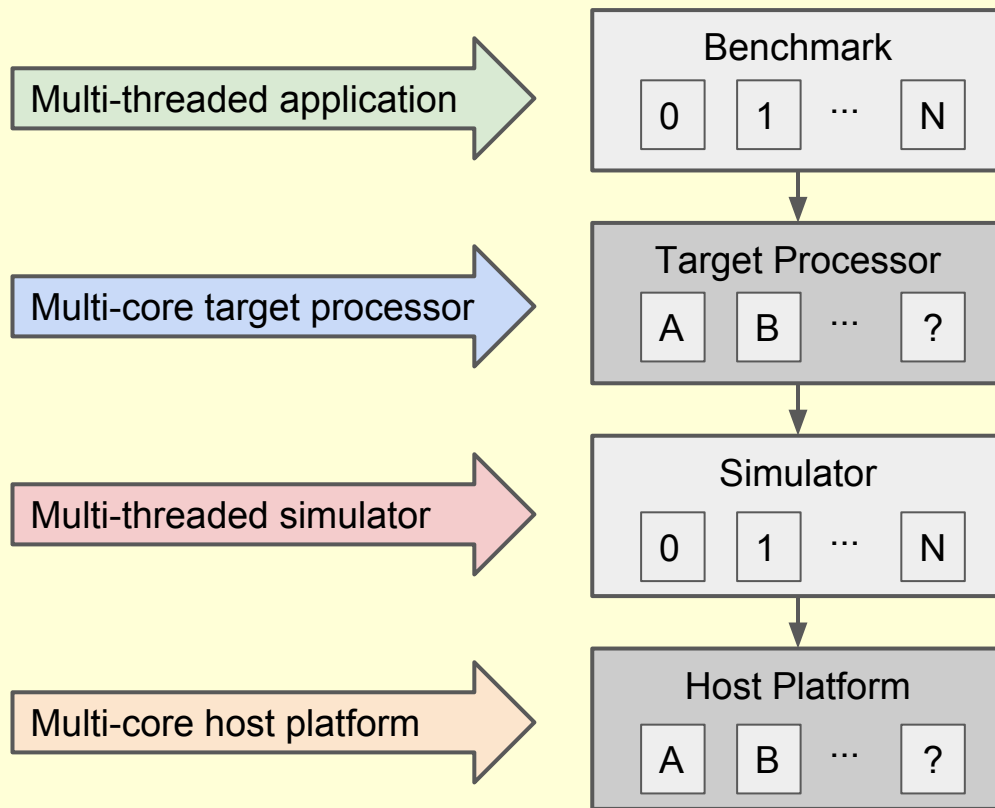
Parallelisation in all levels of the simulation stack



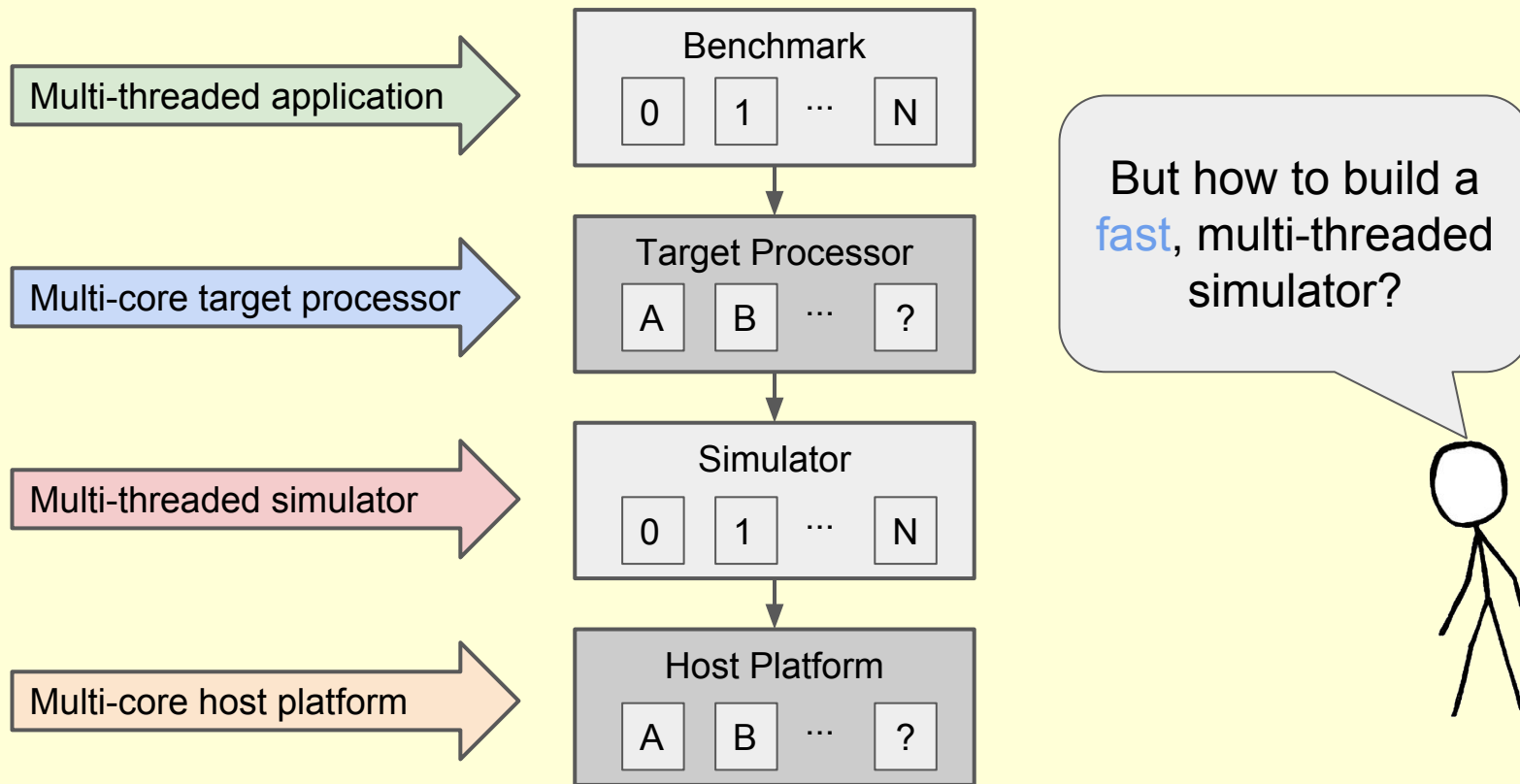
A **multi-threaded simulator** running on a **single-core host**.

Question: Does this make sense?

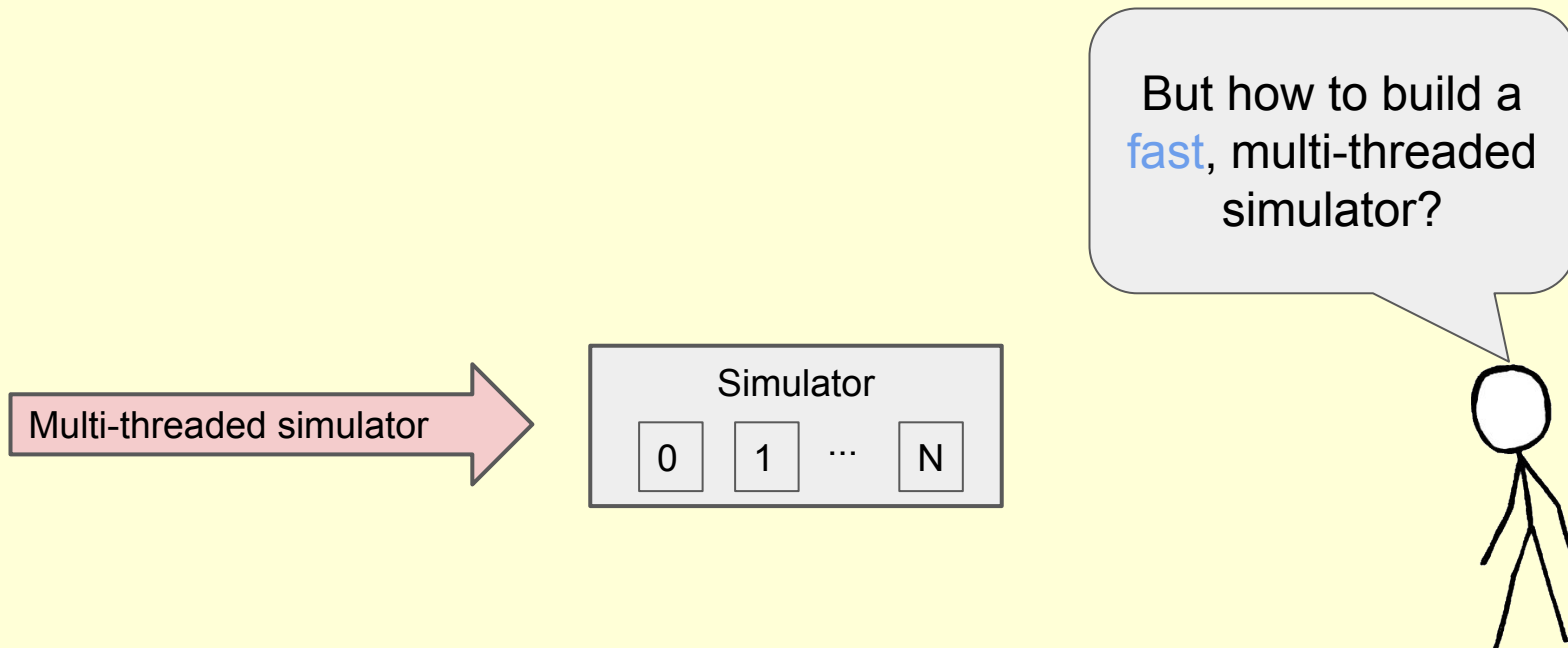
Parallelisation in all levels of the simulation stack



Parallelisation in all levels of the simulation stack



Parallelisation in all levels of the simulation stack



Parallel Simulation Techniques

- Discrete event simulation
- Quantum simulation
- Slack simulation



Parallel Simulation Techniques

- Discrete event simulation

- Quantum simulation

Not schrödinger's cat
'quantum' though

- Slack simulation



Space Granularity

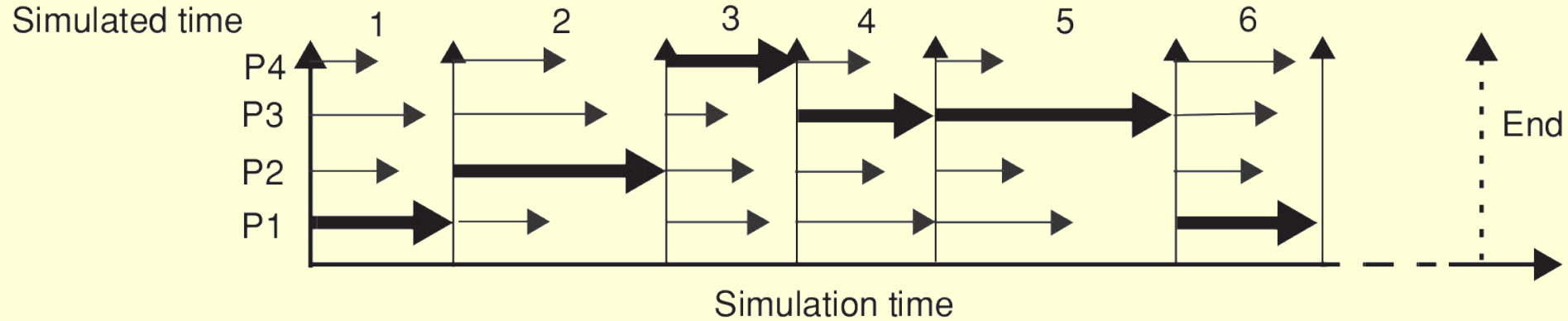
- The textbook implicitly assumes the **smallest hardware block** that can be mapped to a simulator thread is a **full target core**.
- Holds for almost all real-world simulators, which severely **limits the parallelism**

Space Granularity

- 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, which severely **limits the parallelism**

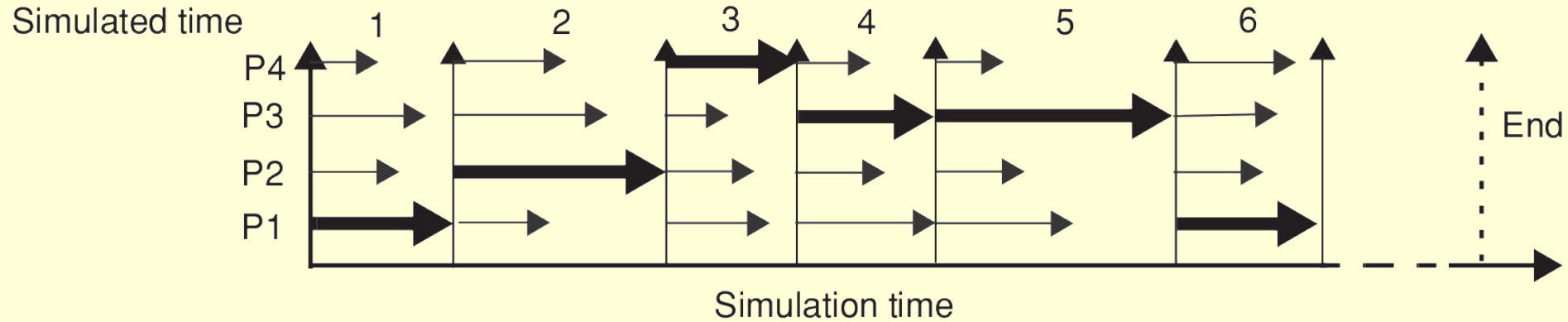
Exception is RTL simulation, there the blocks can be smaller.
The Rocketick simulator even appears to use GPUs! [1]

Discrete-Event Simulation



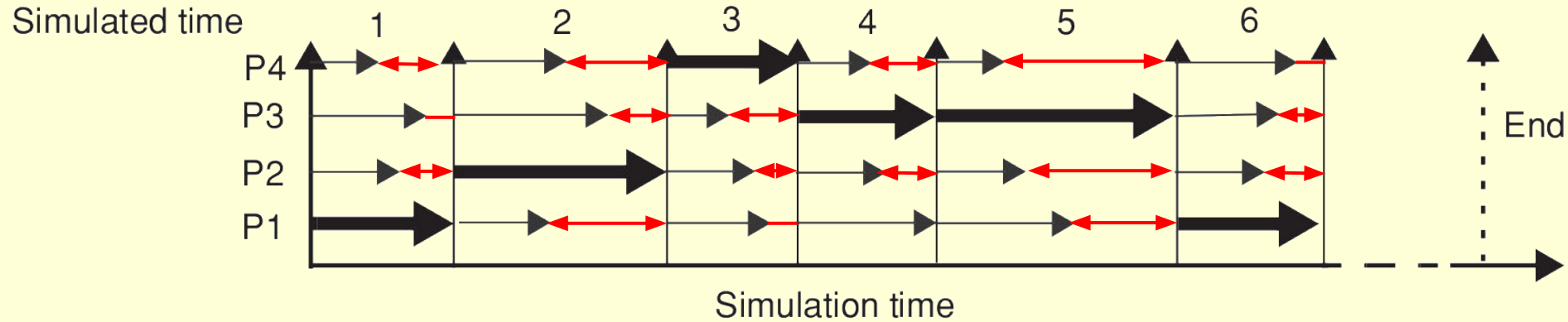
A logical choice for a simulator “time step” is one cycle for the fastest core.

Discrete-Event Simulation



Disadvantage

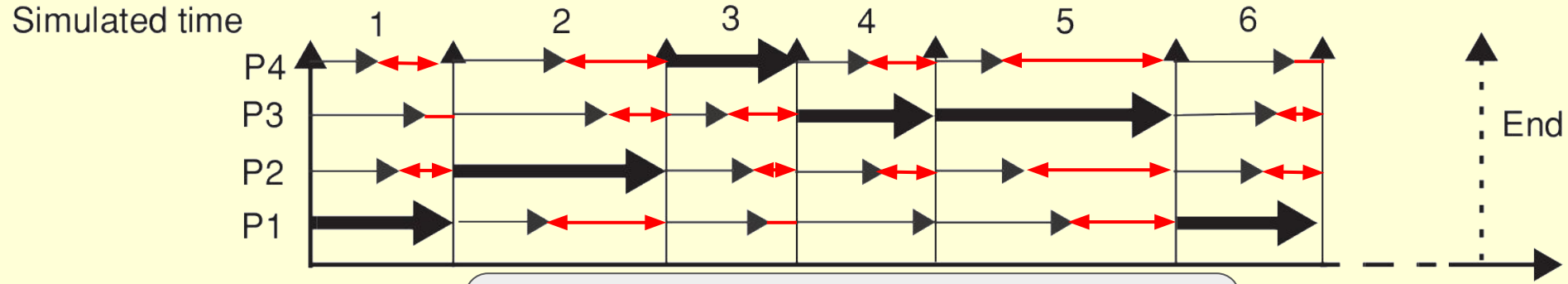
Discrete-Event Simulation



Disadvantage

Under utilisation of the host platform if threads are idle for synchronisation

Discrete-Event Simulation



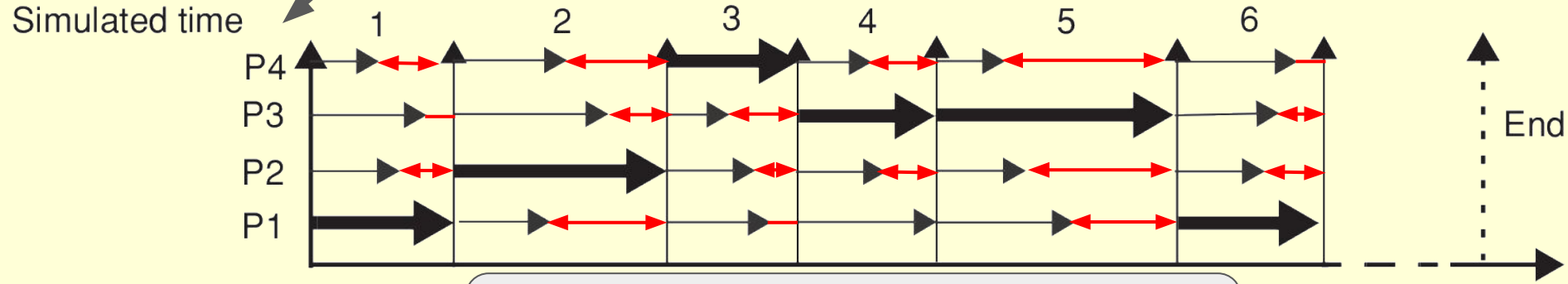
Is it really this bad? What assumption did the author of the book make here?

Disadvantage

Under utilisation of the host platform if threads are idle for synchronisation

Discrete-Event Simulation

Every target processor P_n is mapped to a separate host core



Is it really this bad? What assumption did the author of the book make here?

Disadvantage

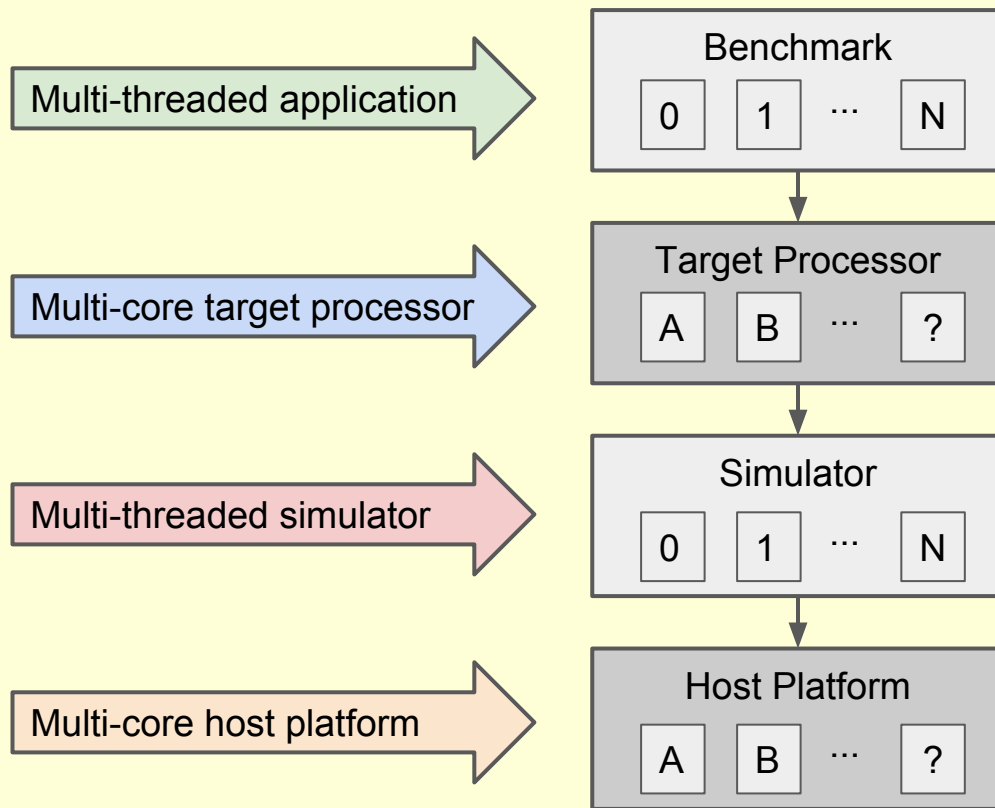
Under utilisation of the host platform if threads are idle for synchronisation

Target vs Host Cores

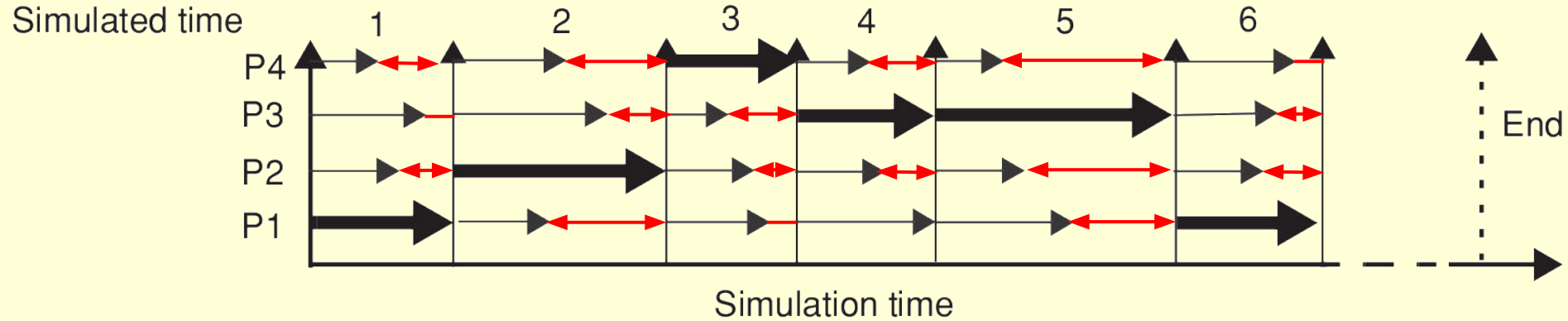
There is a difference between
the number of target cores and
the number of host cores!!!

Target vs Host Cores

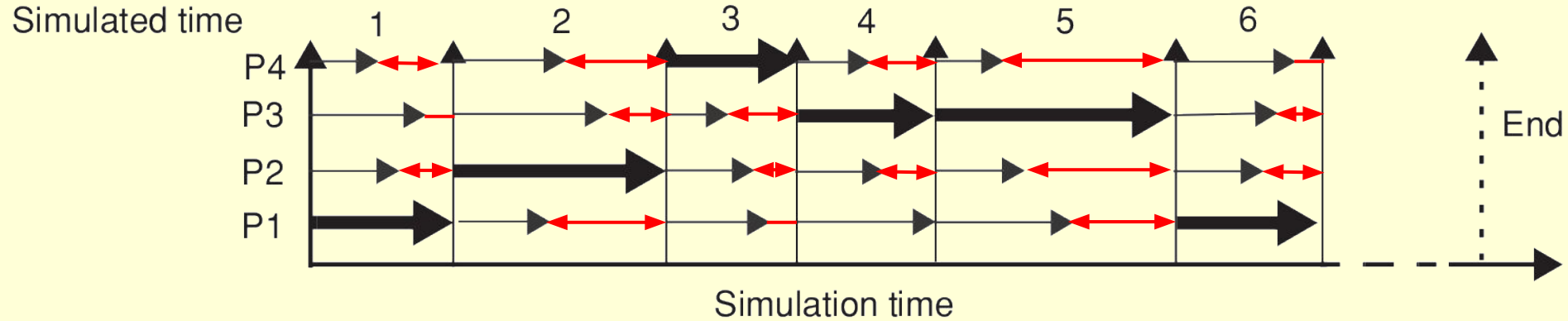
There is **no relation** between the number of target cores and the number of host cores!!!



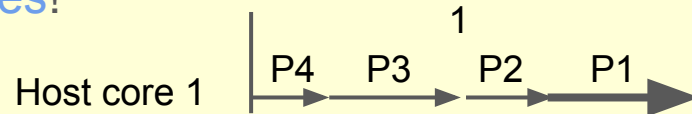
Discrete-Event Simulation



Discrete-Event Simulation

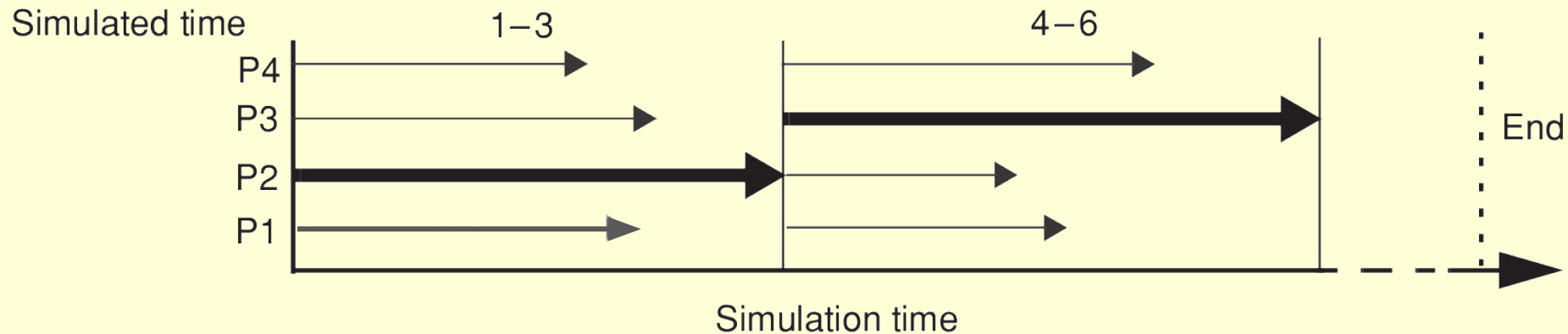


Utilisation of host depends on variation in processing time of a cycle, but also on the amount of host cores!



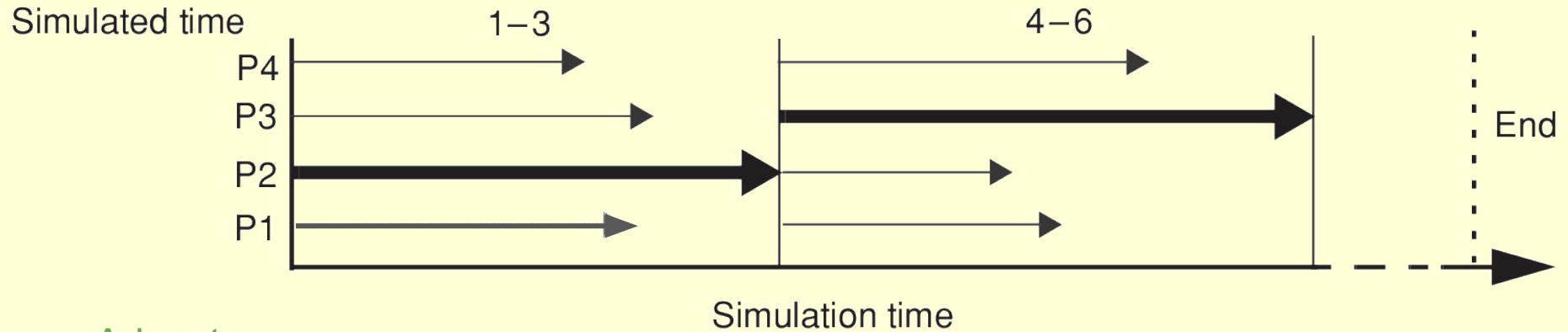
Quantum Simulation

Synchronize threads at larger time-steps, e.g. 3 cycles



Quantum Simulation

Synchronize threads at larger time-steps, e.g. 3 cycles



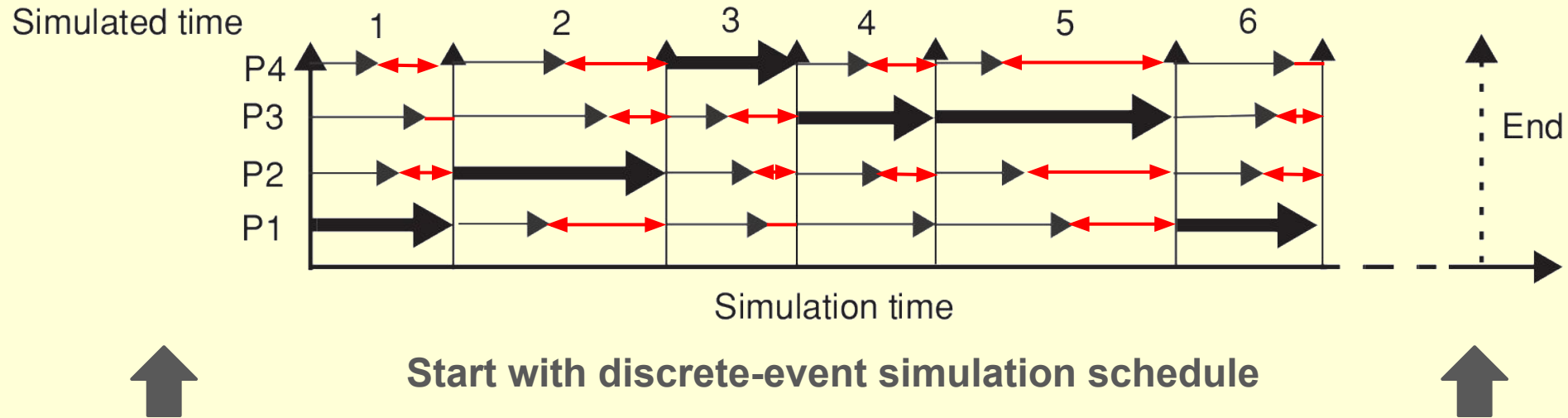
Advantage

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

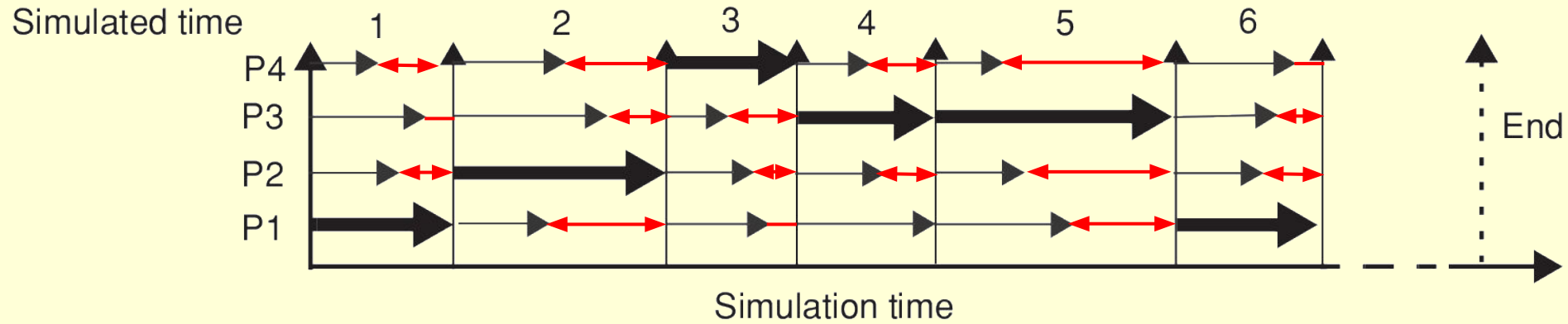
Disadvantage

No longer cycle accurate

Slack Simulation



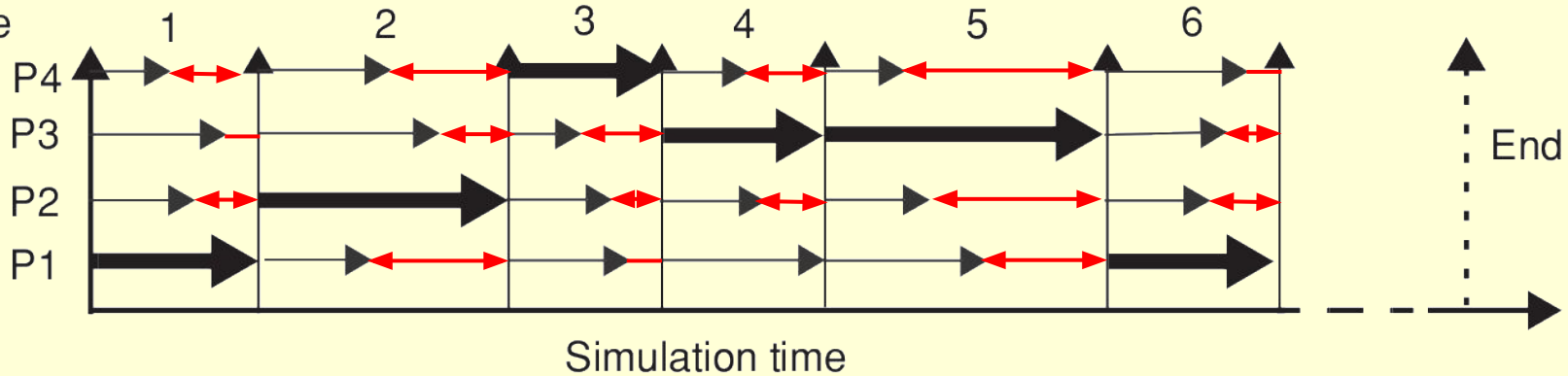
Slack Simulation



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

Slack Simulation

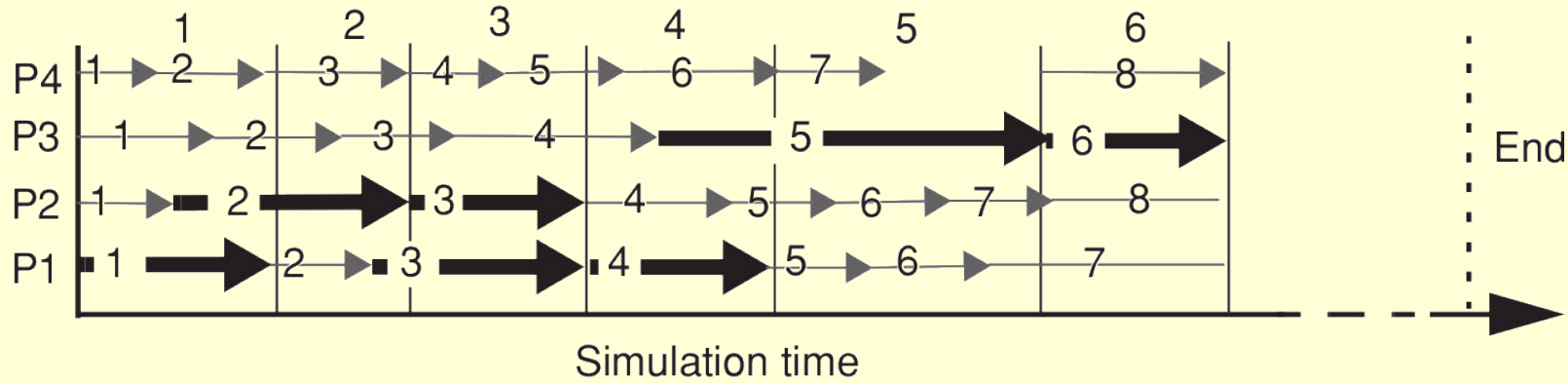
Simulated time



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



Global time



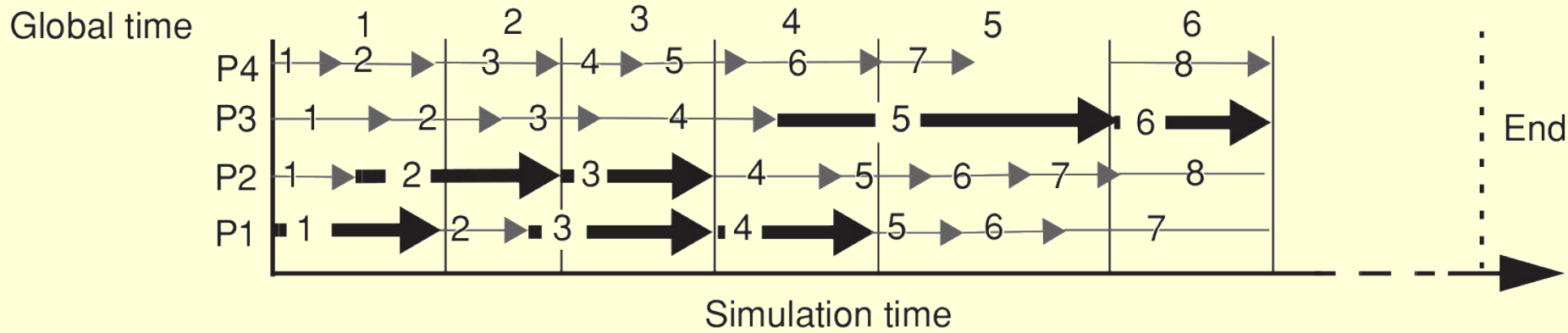
Slack Simulation

Side-effect: Drift

The cores might be simulating different points in time, and could drift apart

Mitigation

Allow a maximum drift (or **slack**), and synchronize when this value is exceeded



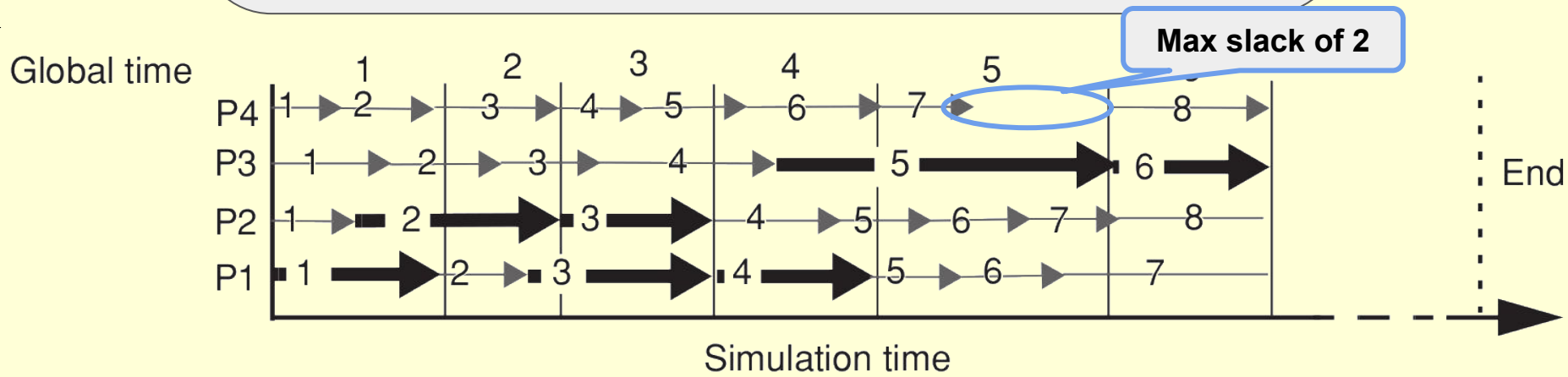
Slack Simulation

Side-effect: Drift

The cores might be simulating different points in time, and could drift apart

Mitigation

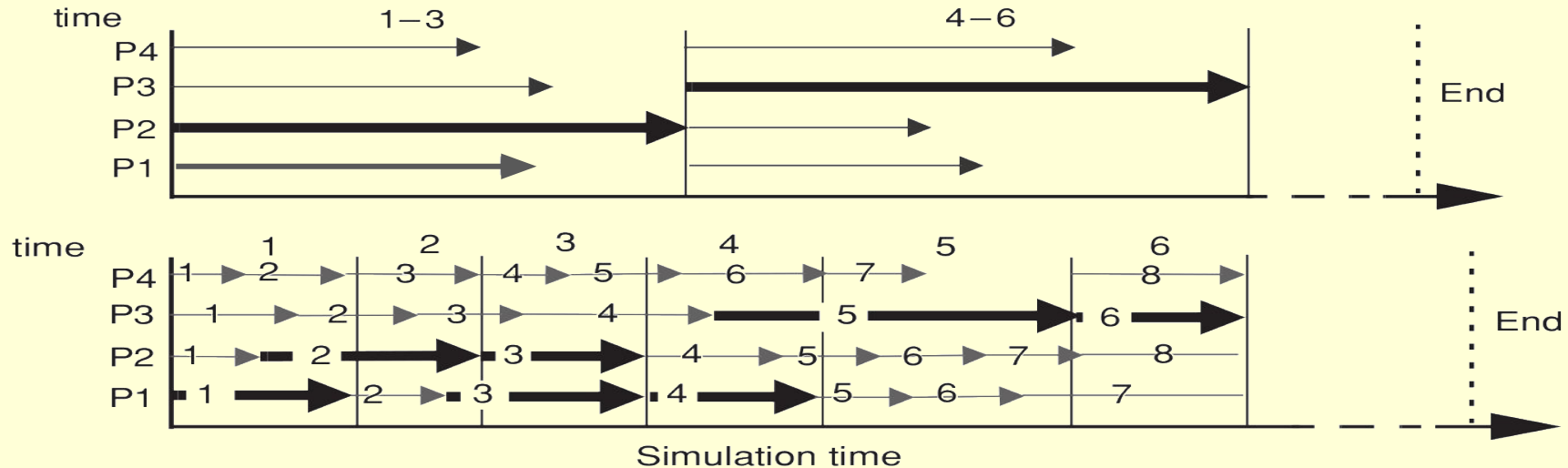
Allow a maximum drift (or **slack**), and synchronize when this value is exceeded



Slack versus Quantum simulation

In **quantum simulation**, the core simulation times always stay within a **cycle window**, which is fixed in global time.

Also in slack simulation the simulation times stay within a window, but with the key difference that this is a **sliding window**.

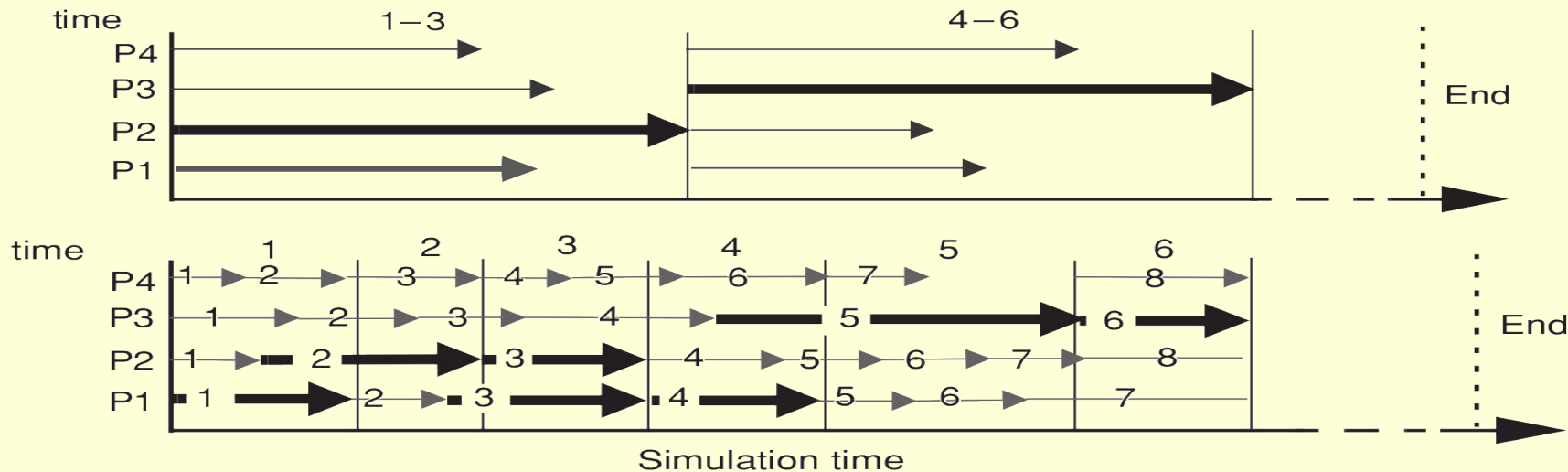


Slack versus Quantum simulation

In **quantum simulation**, the core simulation times always stay within a **cycle window**, which is fixed in global time.

Also in slack simulation the simulation times stay within a window, but with the key difference that this is a **sliding window**.

Typically much less synchronisation!



Still not good enough

From the paper *Graphite: a Distributed Parallel Simulator for Multicores*

“Simulation slowdown is as low as 41× versus native execution”

Still not good enough

From the paper *Graphite: a Distributed Parallel Simulator for Multicores*

“Simulation slowdown is as low as 41× versus native execution”

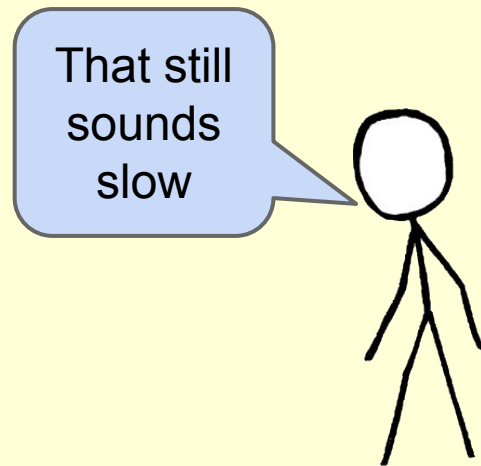
That still
sounds
slow



Still not good enough

From the paper *Graphite: a Distributed Parallel Simulator for Multicores*

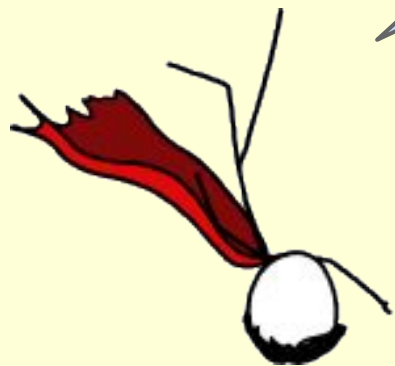
“Simulation slowdown is as low as 41× versus native execution”



Still not good enough

From the paper *Graphite: a Distributed Parallel Simulator for Multicores*

“Simulation slowdown is as low as 41× versus native execution”



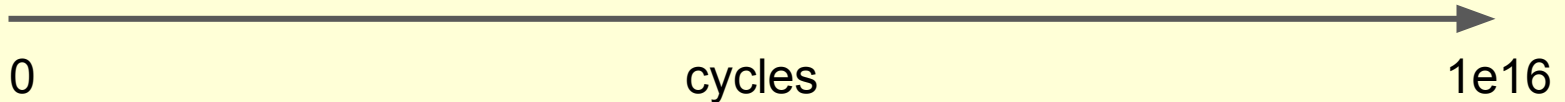
Yes :(

That still
sounds
slow



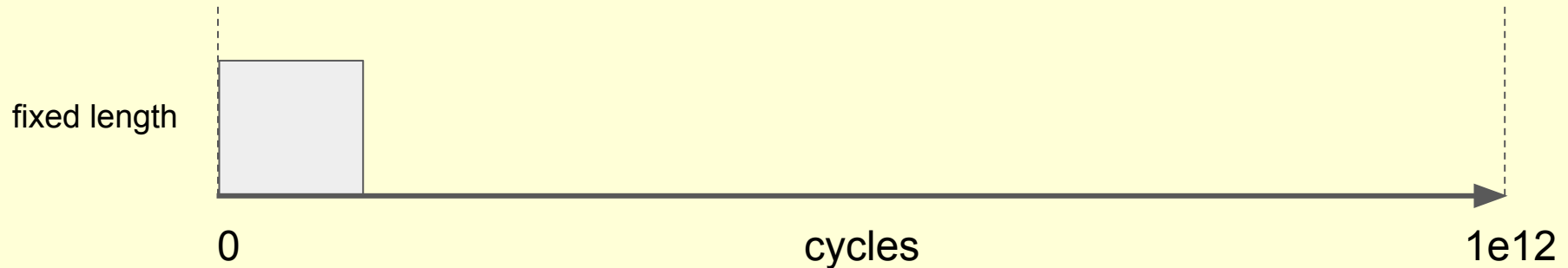
Question

What can we do if it still takes weeks or months to simulate a full benchmark?

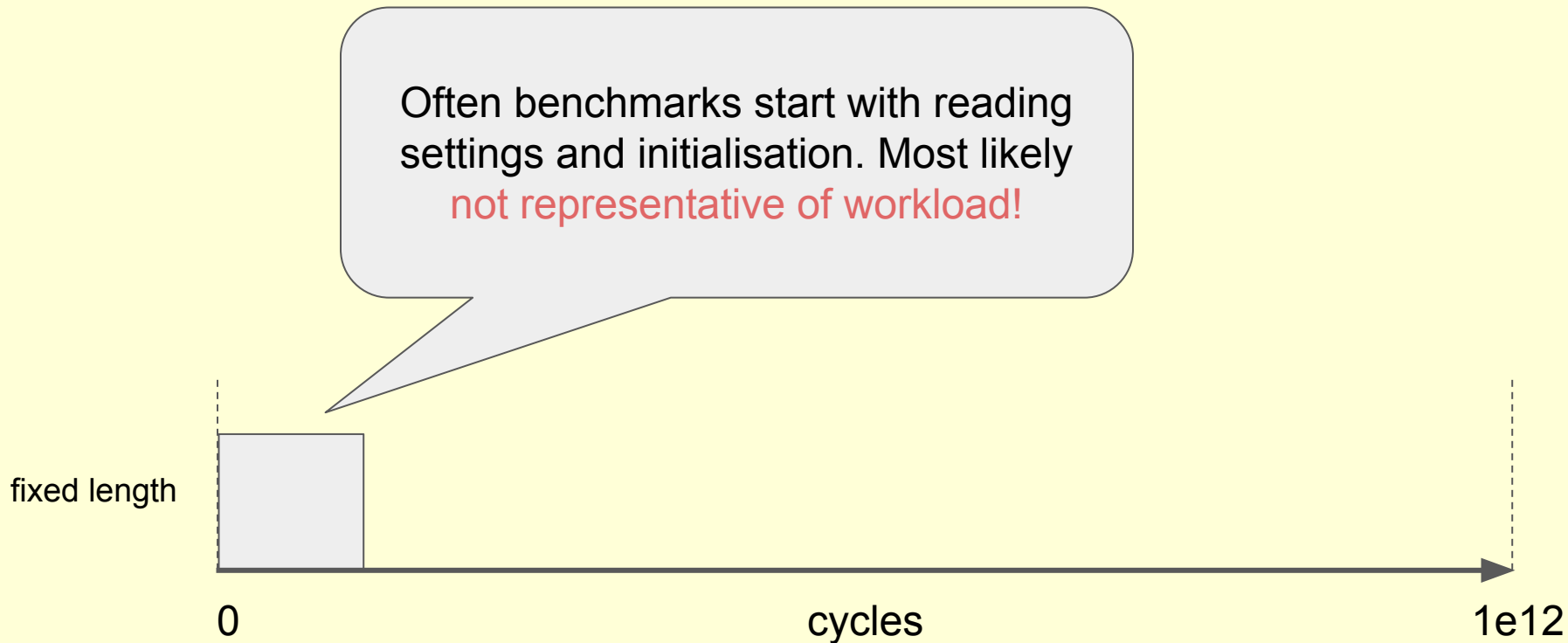


Workload Sampling

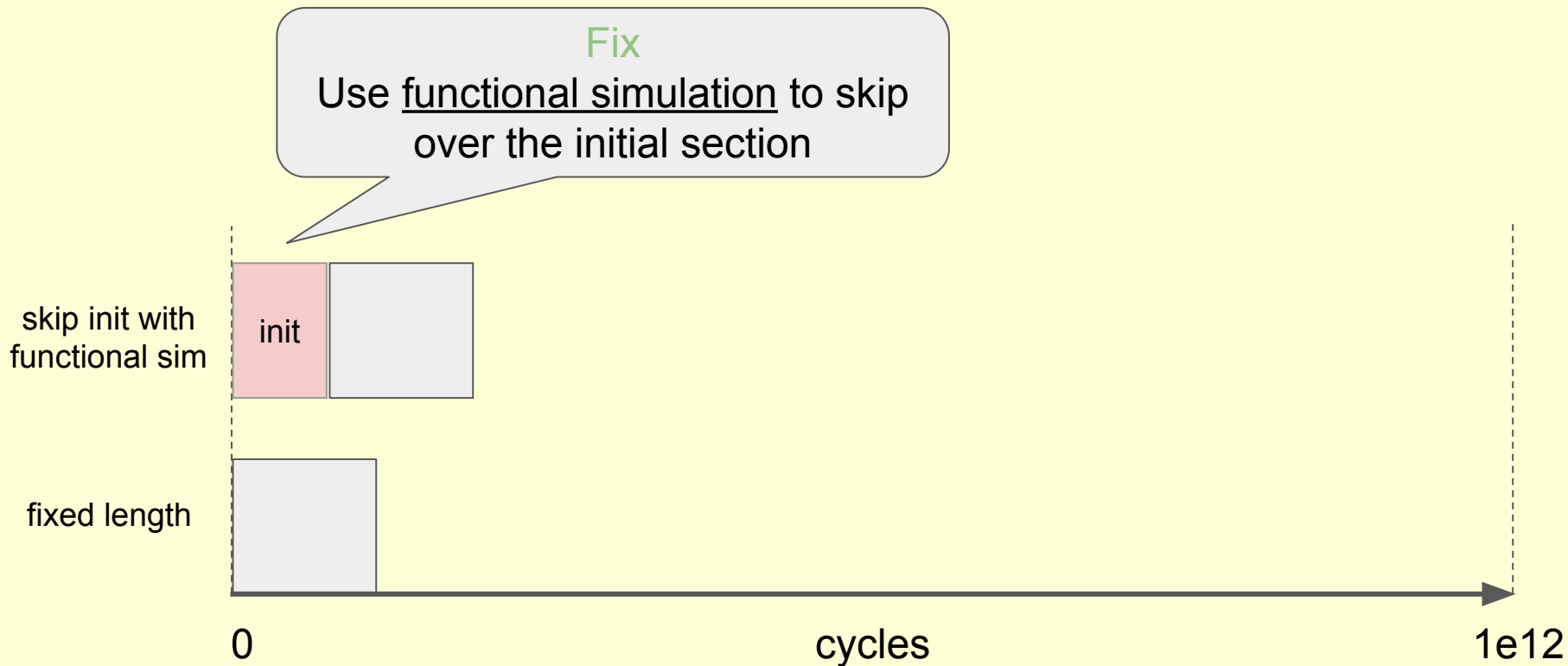
Naive Approach
Only simulate first X cycles



Workload Sampling



Workload Sampling



Workload Sampling

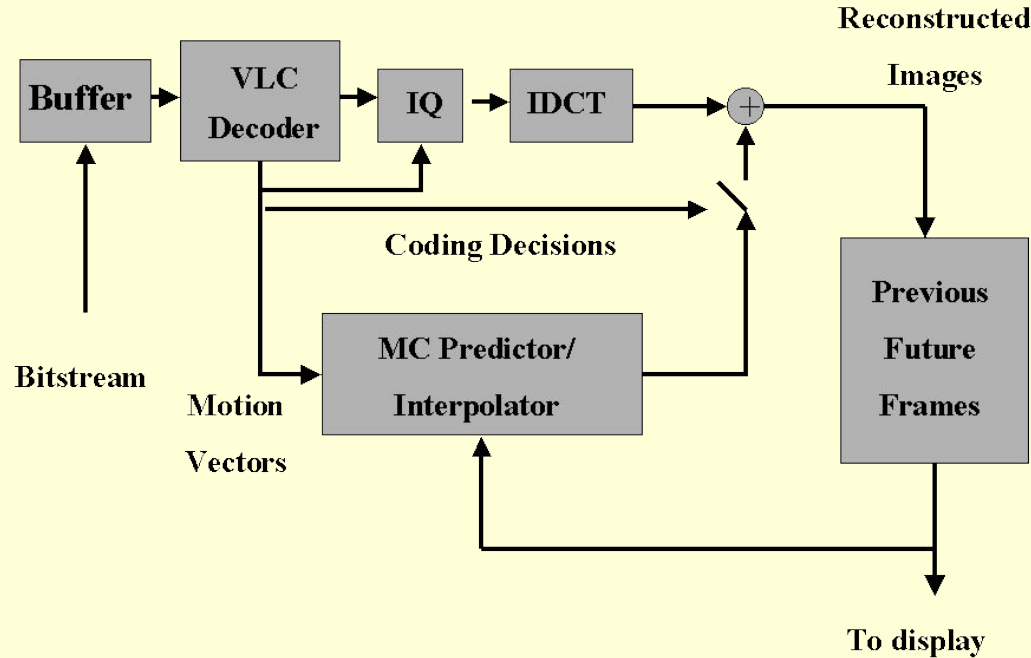
Question

Is the window always a good
representation of the benchmark?
Why/why not?



Program Modes

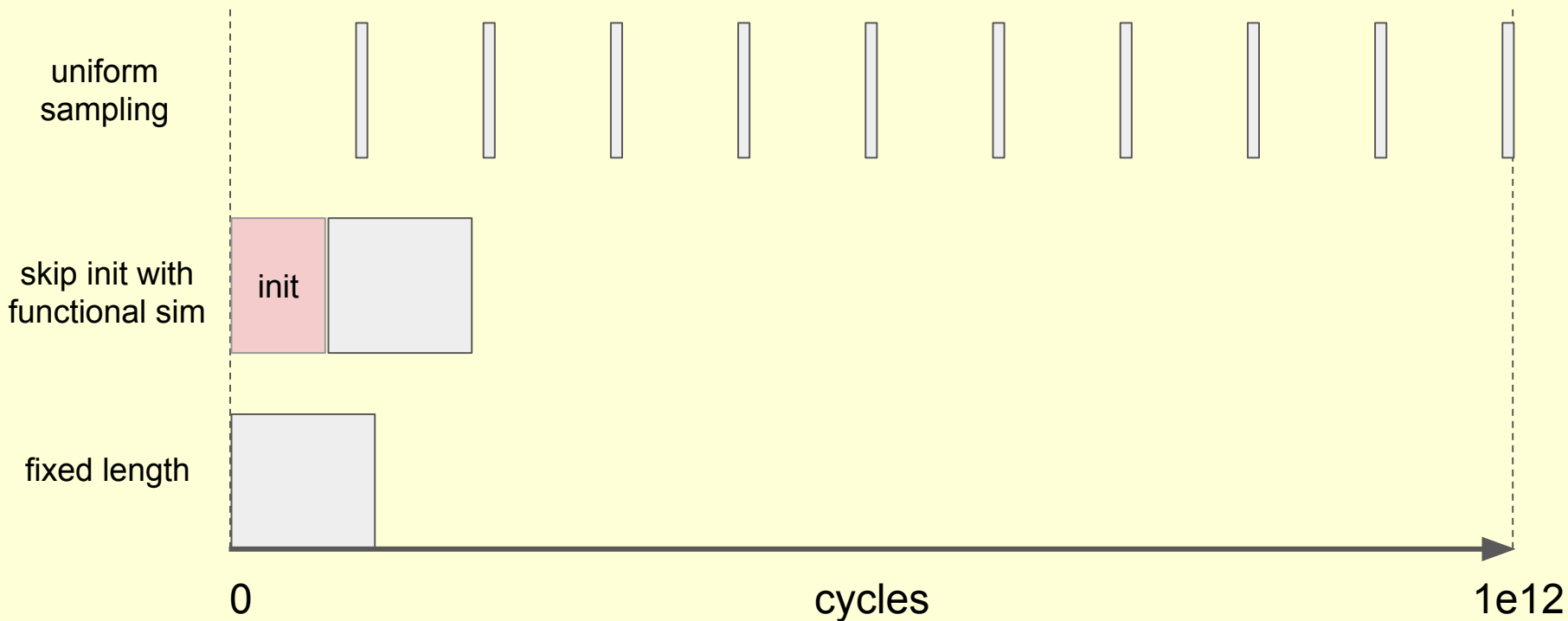
MPEG-2 DECODER



Real world programs spend time in different modes,
which can have very different characteristics

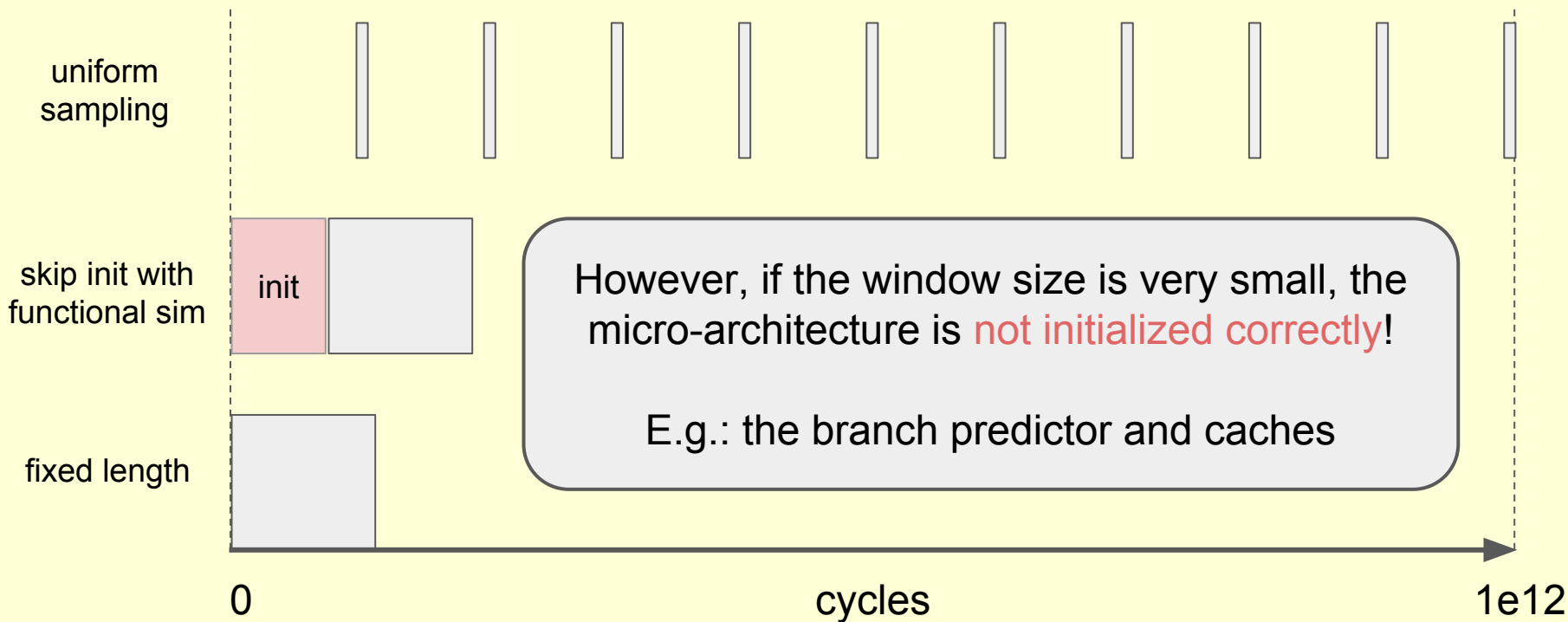
Workload Sampling

Sample uniformly over the program, hopefully capturing the dominant modes



Workload Sampling

Sample uniformly over the program, hopefully capturing the dominant modes

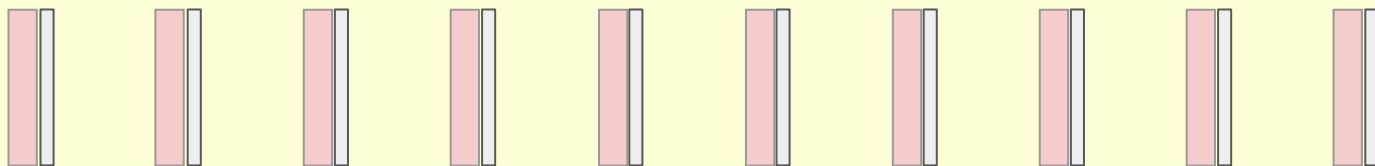


Workload Sampling

Solution: add warm up period before every window



uniform
sampling



skip init with
functional sim



fixed length



0

cycles

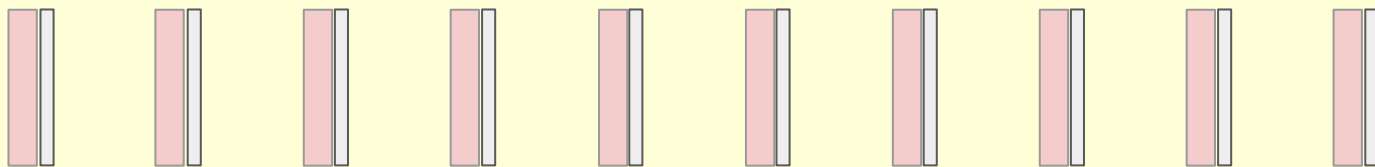
1e12

Workload Sampling

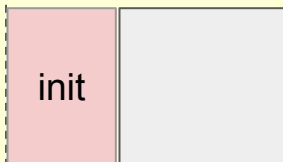
Solution: add warm up period before every window



uniform
sampling



skip init with
functional sim



fixed length



Question
How long should we warm-up?

0

cycles

1e12

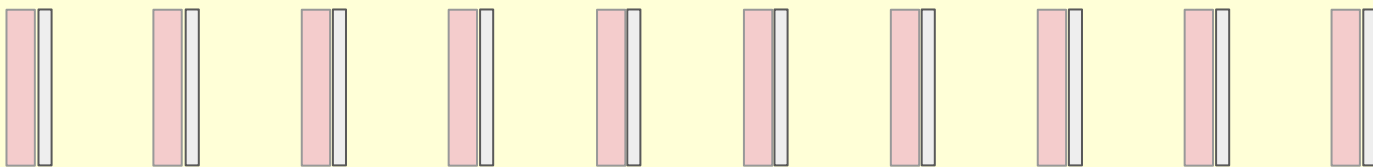
Workload Sampling

[1] *SMARTS: accelerating microarchitecture simulation via rigorous statistical sampling* - Roland E. Wunderlich et al.

Solution: add warm up period before every window



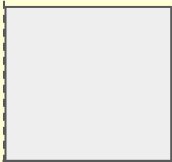
uniform
sampling



skip init with
functional sim



fixed length



Some numbers suggested by SMARTS [1] to get a feeling for the scale:

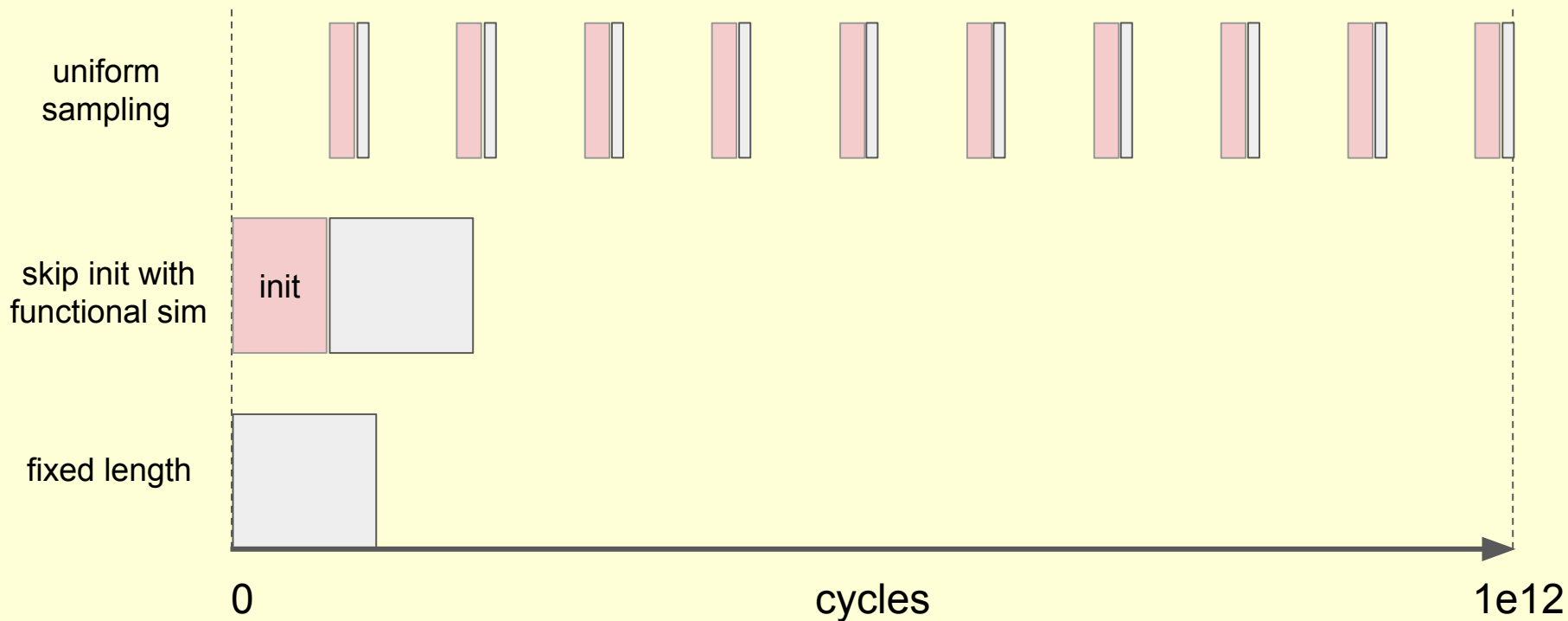
- Initializing caches 500.000 cycles
- Initializing branch prediction, reorder buffers, etc (micro architectural structures.) 4000 cycles
- window size 1000 cycles

0

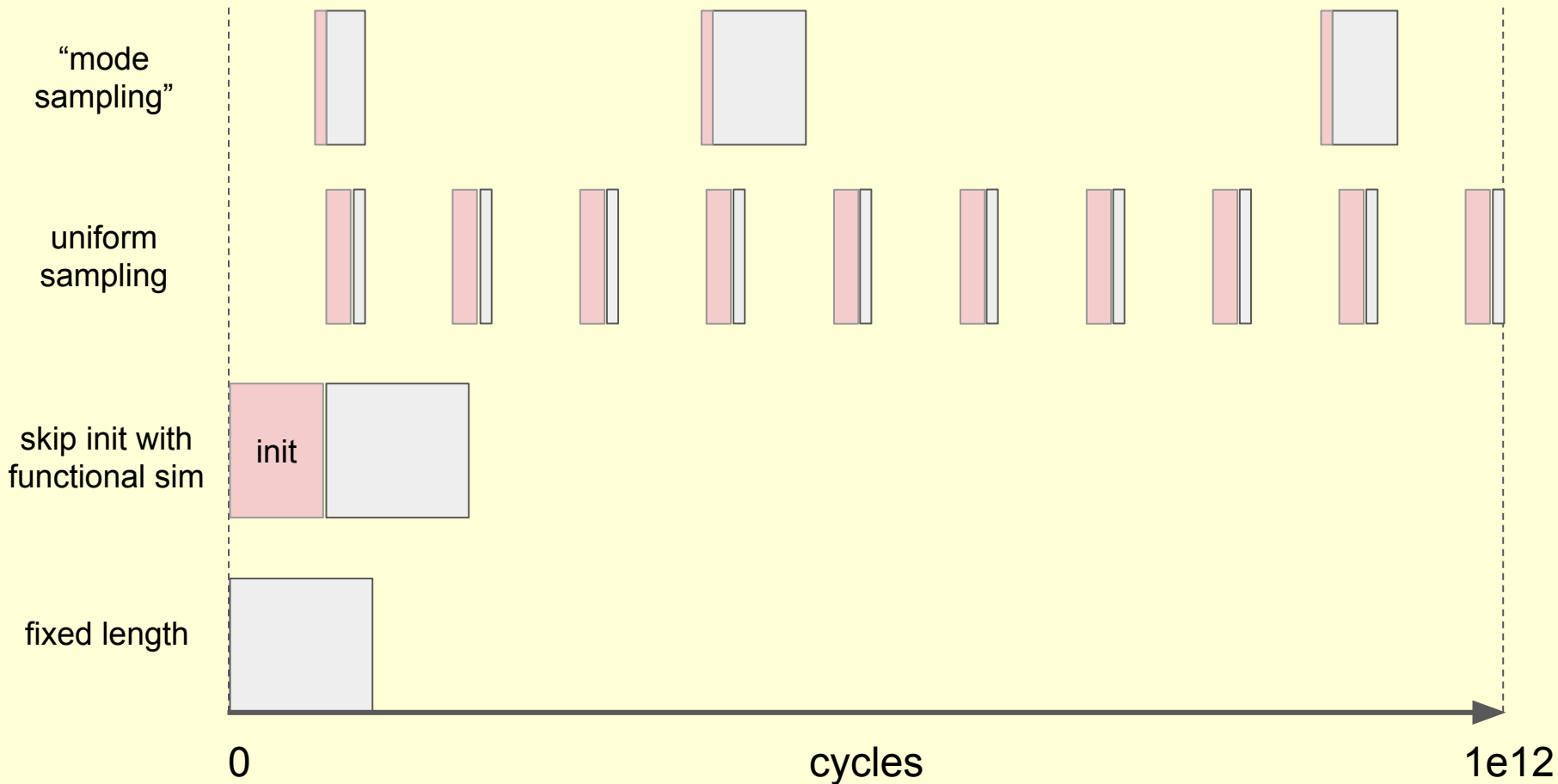
cycles

1e12

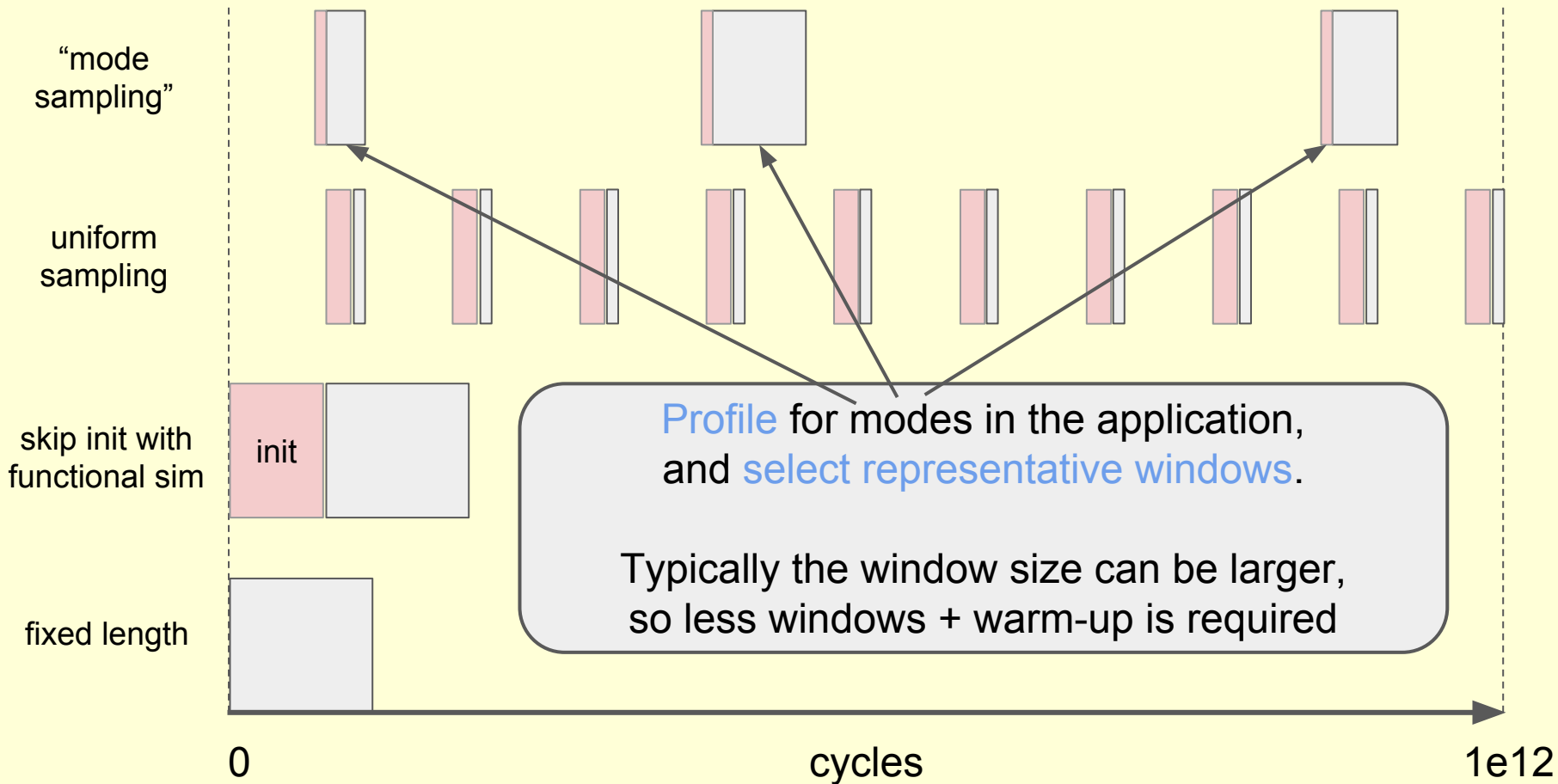
Workload Sampling



Workload Sampling



Workload Sampling



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 (the meta lecture)

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 (the meta lecture)



You can read about
all of this in your
[textbook, chapter 9](#)

