2025 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

MUNICH, GERMANY
OCTOBER 14-15, 2025

Simulation Time Federation

Mark Burton
Director, Qualcomm France SARL.

Mahmoud Kamel
Staff Engineer, Qualcomm Germany GmbH.

Alwalid Salama
Staff Engineer, Qualcomm Germany GmbH.

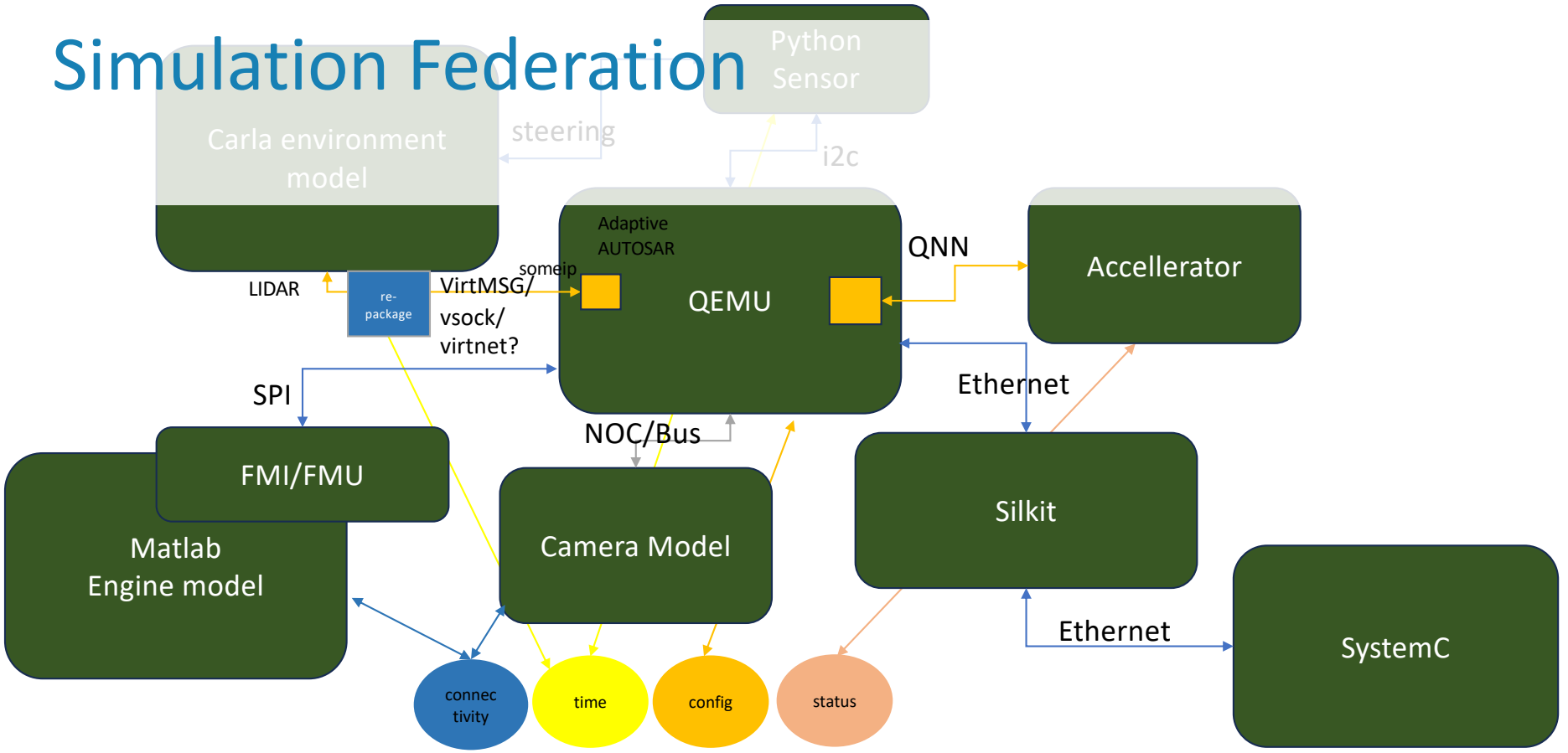
Qualcomm

accellera SYSTEMS INITIATIVE

“What time is it, Eccles?”

“Err, Just a minute. I, I’ve got it written down 'ere on a piece of paper”

Simulation Federation



Connecting to the outside world

Connectivity Interfaces

- Topology
- Time Sync — This talk
- Config/Control
- Status (Simulation state)
- Data

2023 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE 10 YEAR ANNIVERSARY

Single time interface

Implemented using Zenoh

accellera SYSTEMS INITIATIVE

Goals

Prove sync can work over a Zenoh interface

Show how to integrate with different sync algorithms

Sync algorithms

Steppers

Central controller

“Do step” – and wait for everybody to complete

Watchers

Central time source

Everybody should do their best to be in sync

Talkers

Each node broadcasts it’s time (window) along with all communication

Each node should do their best to stay in sync with it’s neighbours

Implementing nodes as “talkers” allows them to operate within a central controller (either time, or stepper based)

Time Sync: From Time To Time

- Between any two simulators, simply exchange time windows

  {From Time, To Time}

- Always try to advance to the “from” time,
- Try not to advance beyond the “to” time
- When you arrive at the “to” time (or at any point), send a new window to the other side.

- Can be used to handle FMI/FMU, Silkit, SystemC, EDA247, etc.

2023 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

10 YEAR ANNIVERSARY

QEMU

accellera SYSTEMS INITIATIVE

Goals

Use Instruction counts to better approximate time, as seen by each CPU.

("fix" icount's 'warped' time mechanism)

Allow different CPUs to operate at different Instructions-Per-Second

Maintain some synchronization between CPUs.

Don't worry (too much) about determinism

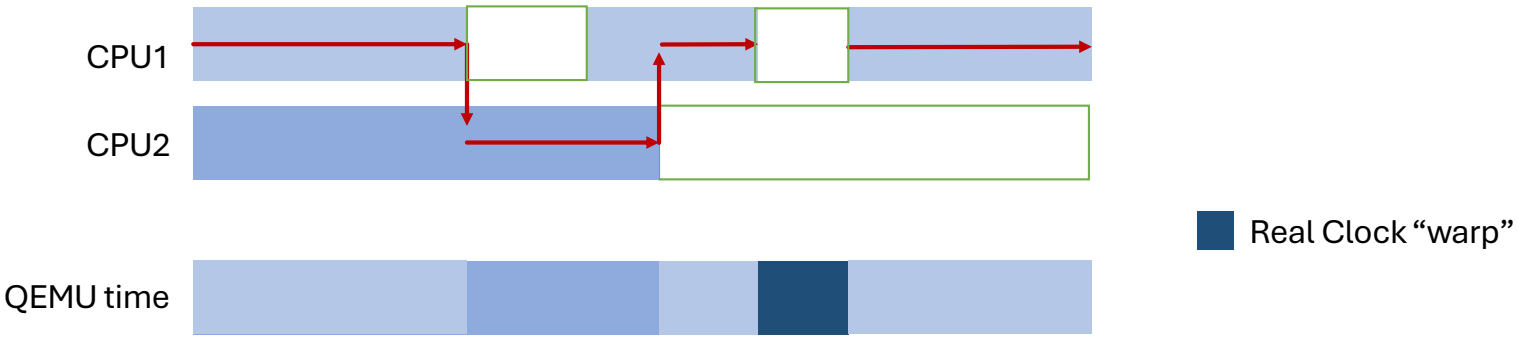
Work with MTTCG

Use the TCG Plugin API.

Proposed “multi-CPU icount”

Choose 1 CPU to be the owner of “time” at any one time. Advance QEMU time as that CPU (alone) advances

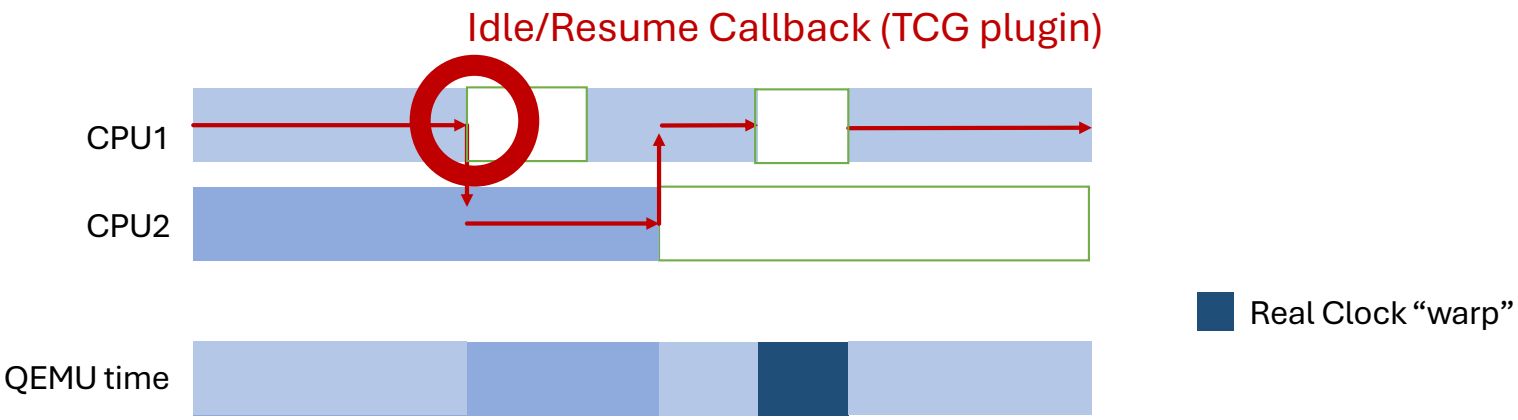
(Note, in principle this is also deterministic)



Proposed “multi-CPU icount”

Choose 1 CPU to be the owner of “time” at any one time. Advance QEMU time as that CPU (alone) advances

(Note, in principle this is also deterministic)



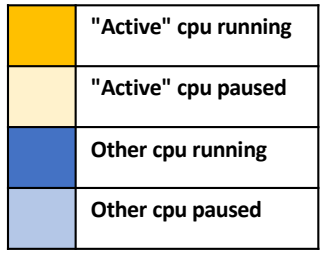
A quantum (or 2) from QEMU time synchronisation

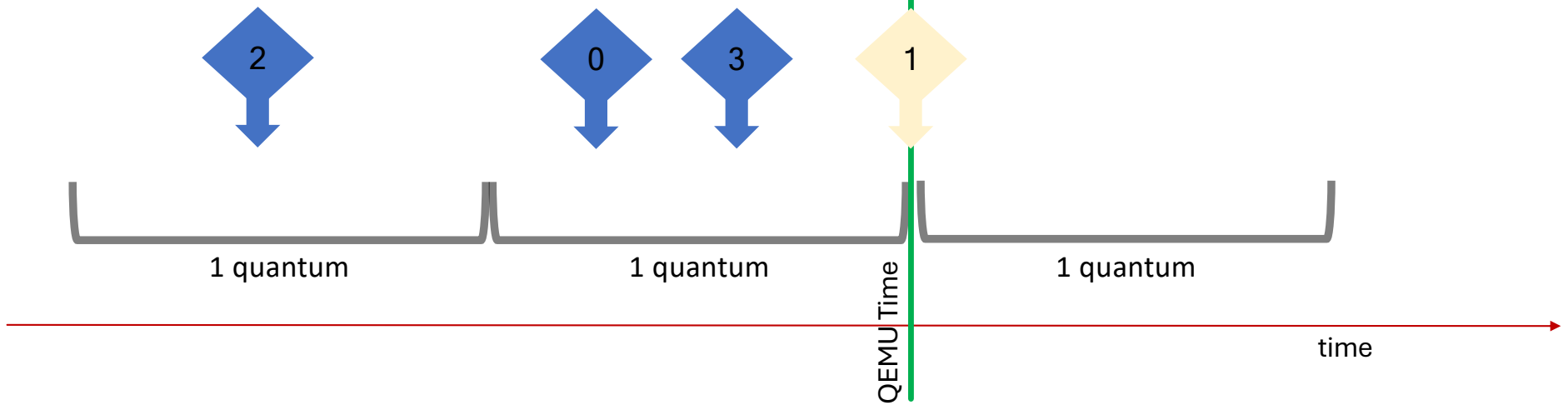
One CPU is the “active” CPU, it determines QEMU time

Golden rules:

1. Only a cpu can pause itself.
2. any cpu can resume any other cpu.

cpu 1 will pause itself, since cpu 2 is more than one quantum behind.



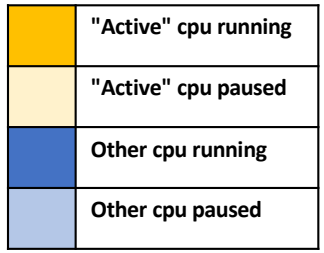


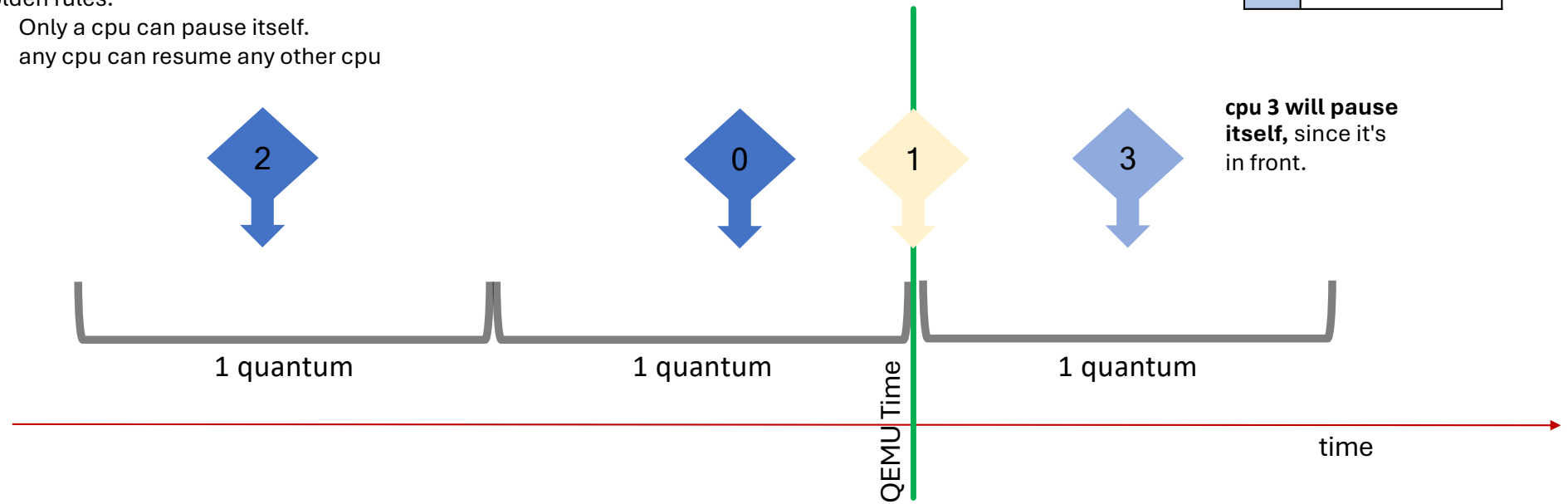
A quantum (or 2) from QEMU time synchronisation

One CPU is the “active” CPU, it determines QEMU time

Golden rules:

1. Only a cpu can pause itself.
2. any cpu can resume any other cpu



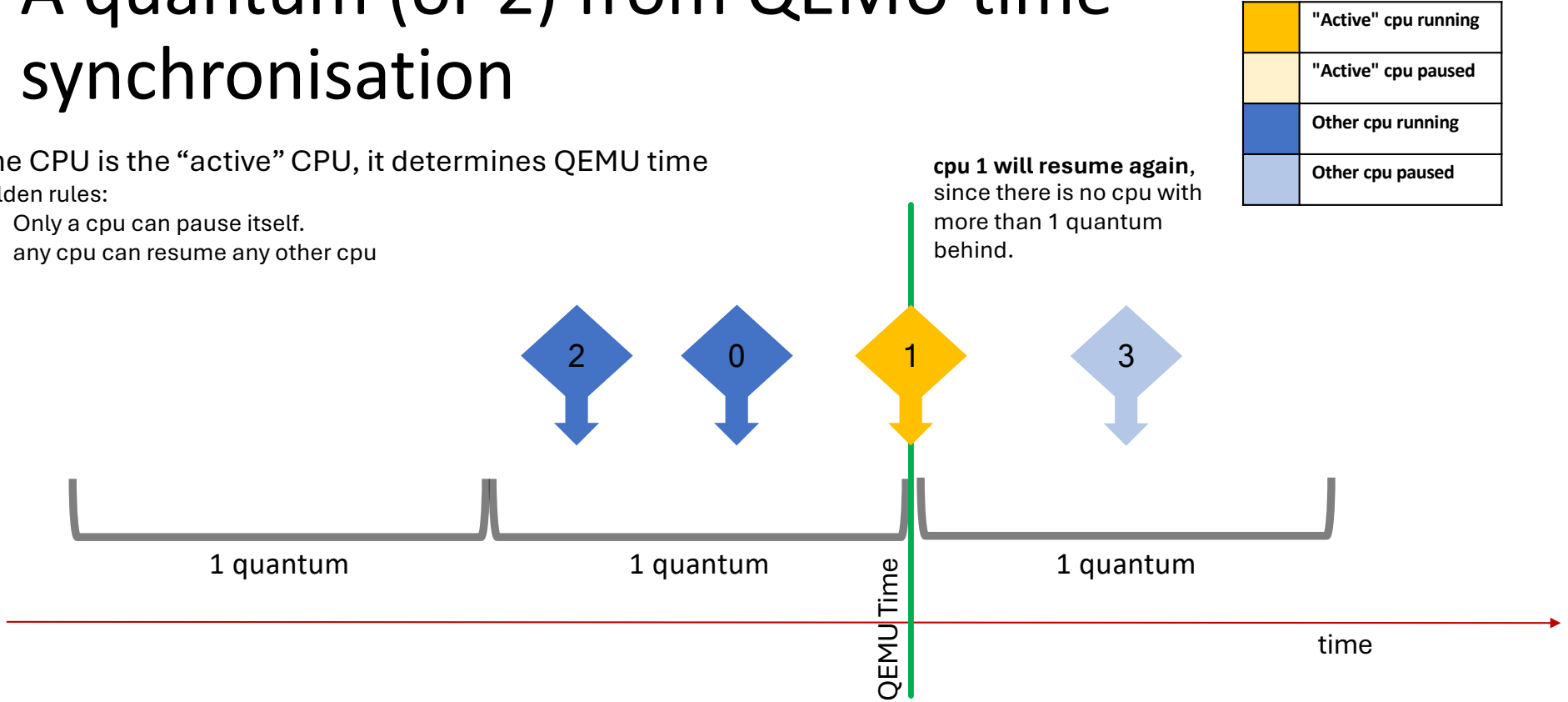


A quantum (or 2) from QEMU time synchronisation

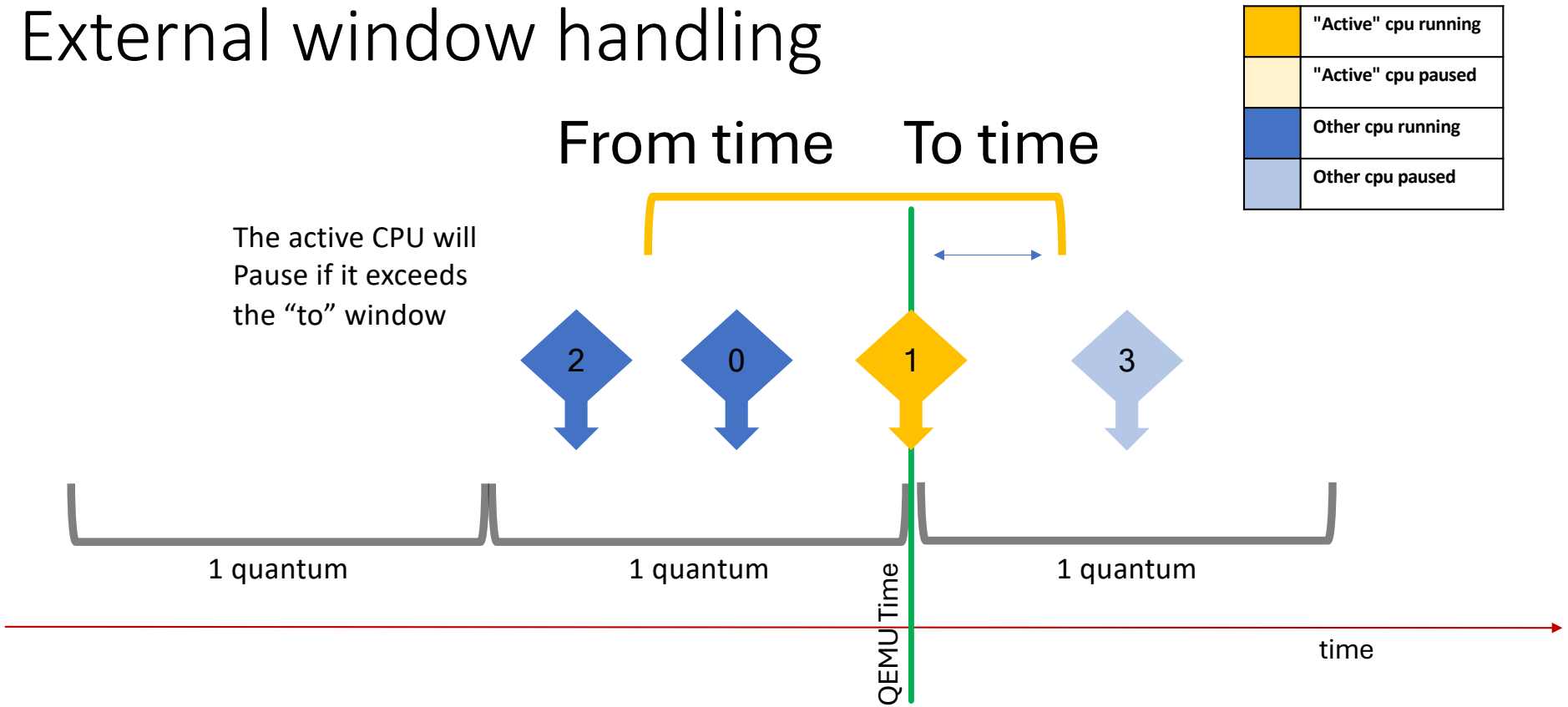
One CPU is the “active” CPU, it determines QEMU time

Golden rules:

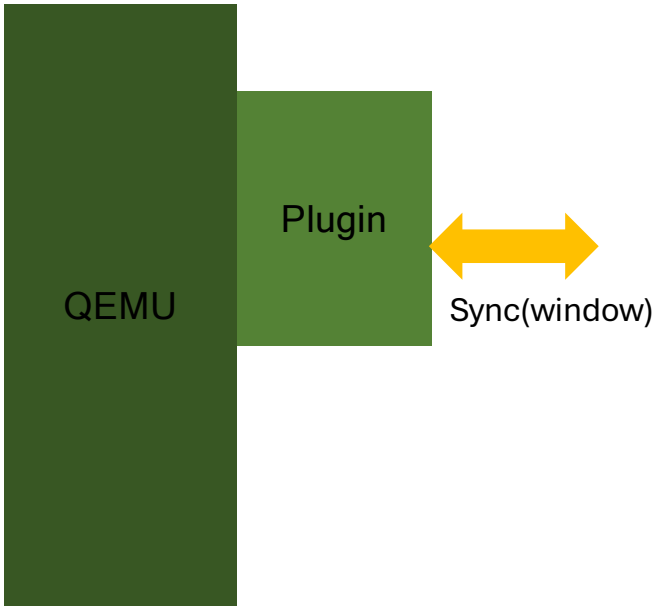
1. Only a cpu can pause itself.
2. any cpu can resume any other cpu



External window handling

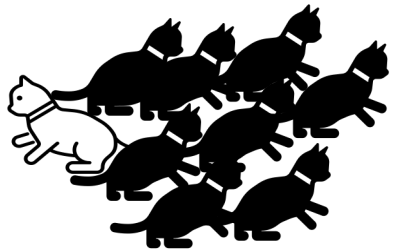


Part 3: Sync with the outside world



- Set a window on the “Active CPU”
- Stop the active CPU if it advances beyond the front of the window
- Each “Quantum”, inform the outside world of our time (done in the QEMU get time callback)
- In short – control “just” the active CPU, hence, only consider “QEMU time”.
- Plugin provides single reflexive interface “sync_window” which takes a time window.

Results (9 out of 10 cats like it)



- Making use of the TCG API to manage time is a ~ 13% performance hit due to the scoreboard callbacks etc over normal ”icount” mode.
  (single TCG CPU running coremark)

Icount mode in MTTCG – what's not to like?

2023 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE 10 YEAR ANNIVERSARY

SystemC

accellera SYSTEMS INITIATIVE

Goals

Allow parts of SystemC to be parallelised

Reduce the single thread constraints for specific modules

Maintain single name hierarchy

Allow communication and events

Flexibly synchronise

Parallel SystemC : The Plan !

1. Make sim context thread local
2. Make sure events are handled in the right sim context
3. Fix all the bugs…

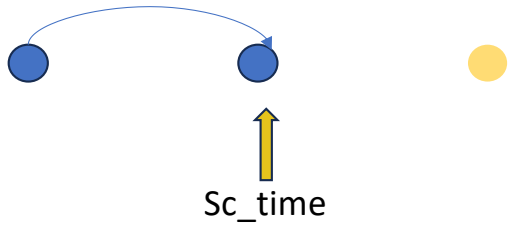
4. SC_PARALLEL(your_module, sync_algorithm)

5. **BUT What about time synchronization!**

accellera SYSTEMS INITIATIVE

Observer Event

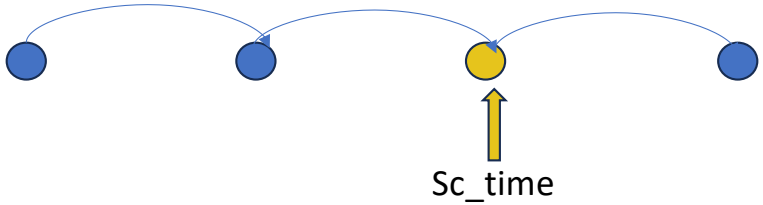
- An event which fires ONLY when time passes it



- Time will not advance until there is a future event

Observer Event

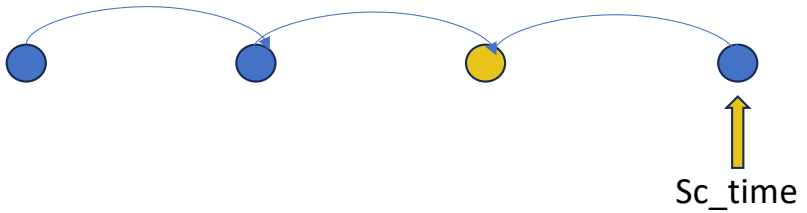
- An event which fires ONLY when time passes it



- Time will not advance until there is a future event

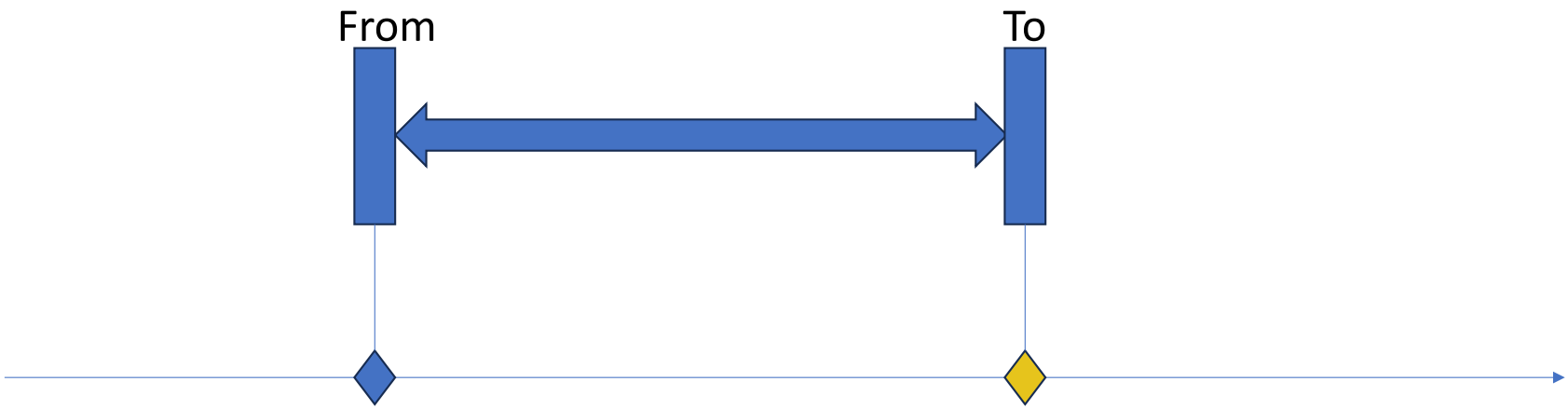
Observer Event

- An event which fires ONLY when time passes it



- Time will not advance until there is a future event

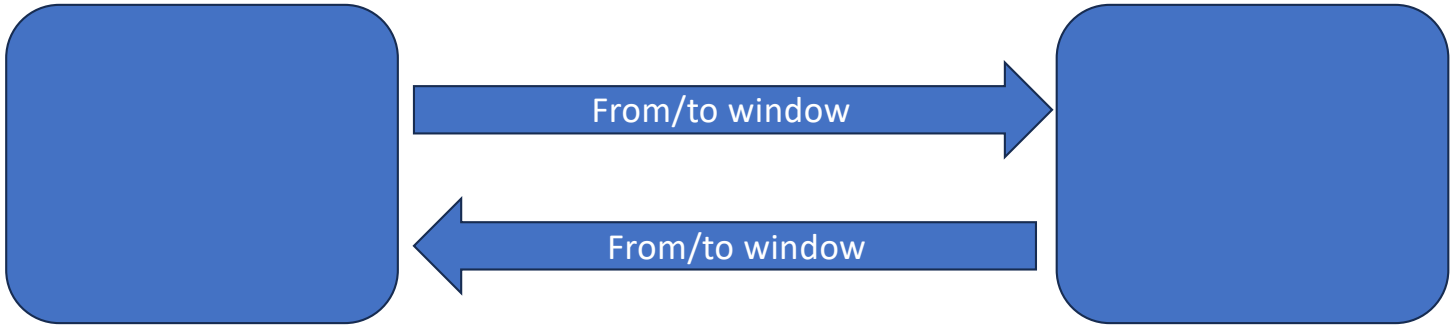
Handling a window in SystemC



Normal event:
Advance time (as quickly as possible) to this event

Observer event:
Advance time to this event only if there are other things driving the simulation forward

Sync Algorithms?



To = + 1 quantum

To = “next event”

Other things we had to change in SystemC

01

Move simcontext to be thread local

02

Redirect all event notifications to happen on the “target” simcontext

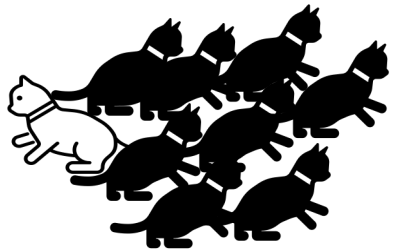
03

Make b_transport to execute on the ”target” simcontext

04

Provide a wrapper around an sc_module to run on a separate thread

Results (9 out of 10 cats like it)



Parallel SystemC – you kidding me, of course all cats like it!

All together!

SystemC <-> SystemC

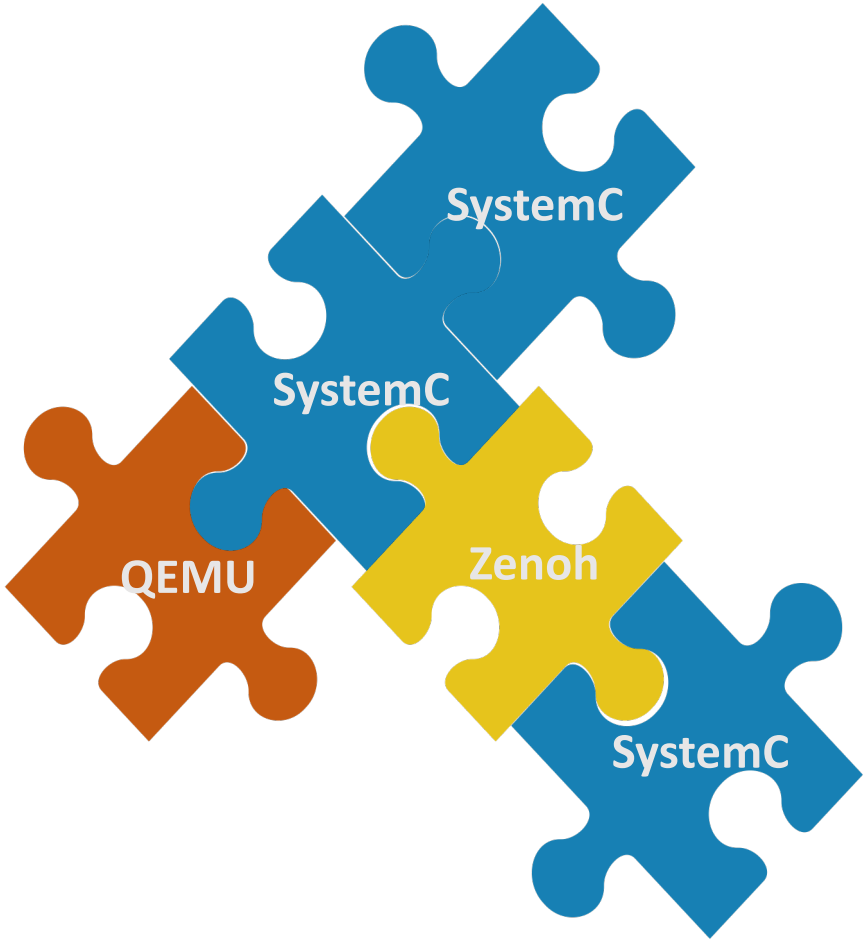
- New Observer Event
- Parallel SystemC (multiple sim contexts)

SystemC <-> QEMU

- New TCG scoreboard Plugin
- Sync between CPUs
- Works with MTTCG

SystemC <-> Zenoh

- POC for many different simulation backbones



Questions

Come to SCED to discuss !