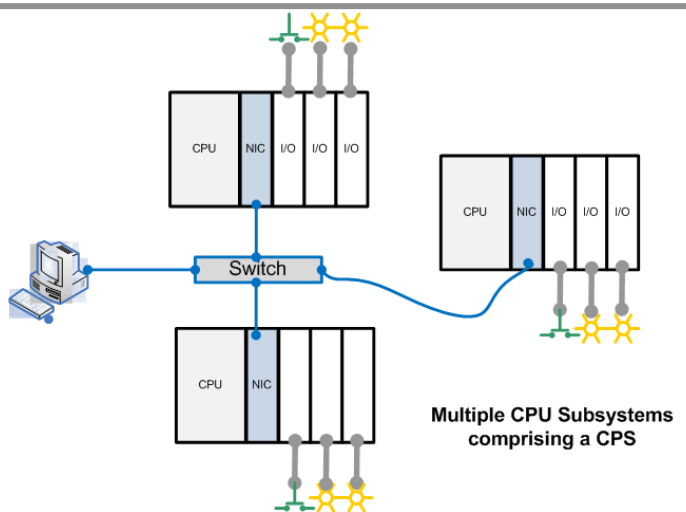
New Clockwork: Time in CPUs

Sundeep Chandhoke
Arkadeb Ghosal
Hugo Andrade

Motivation

- Precise Time Synchronization is a foundational technology enabling a broad array of applications in Cyber Physical Systems (CPS)
  - Continues to gain momentum in IEEE 1588-2008, 802.1AS, 802.11v …
- Some Application Areas:
  - Instrumentation – distributed data acquisition, logic analyzers
  - Telecom-cell towers synchronized for seamless handoff
  - Industrial Automation/Control
  - Synchronized A/V Streaming (802.1 AVB) in studios, autos and homes
  - Energy, Big Physics, Railway, Medical and many others

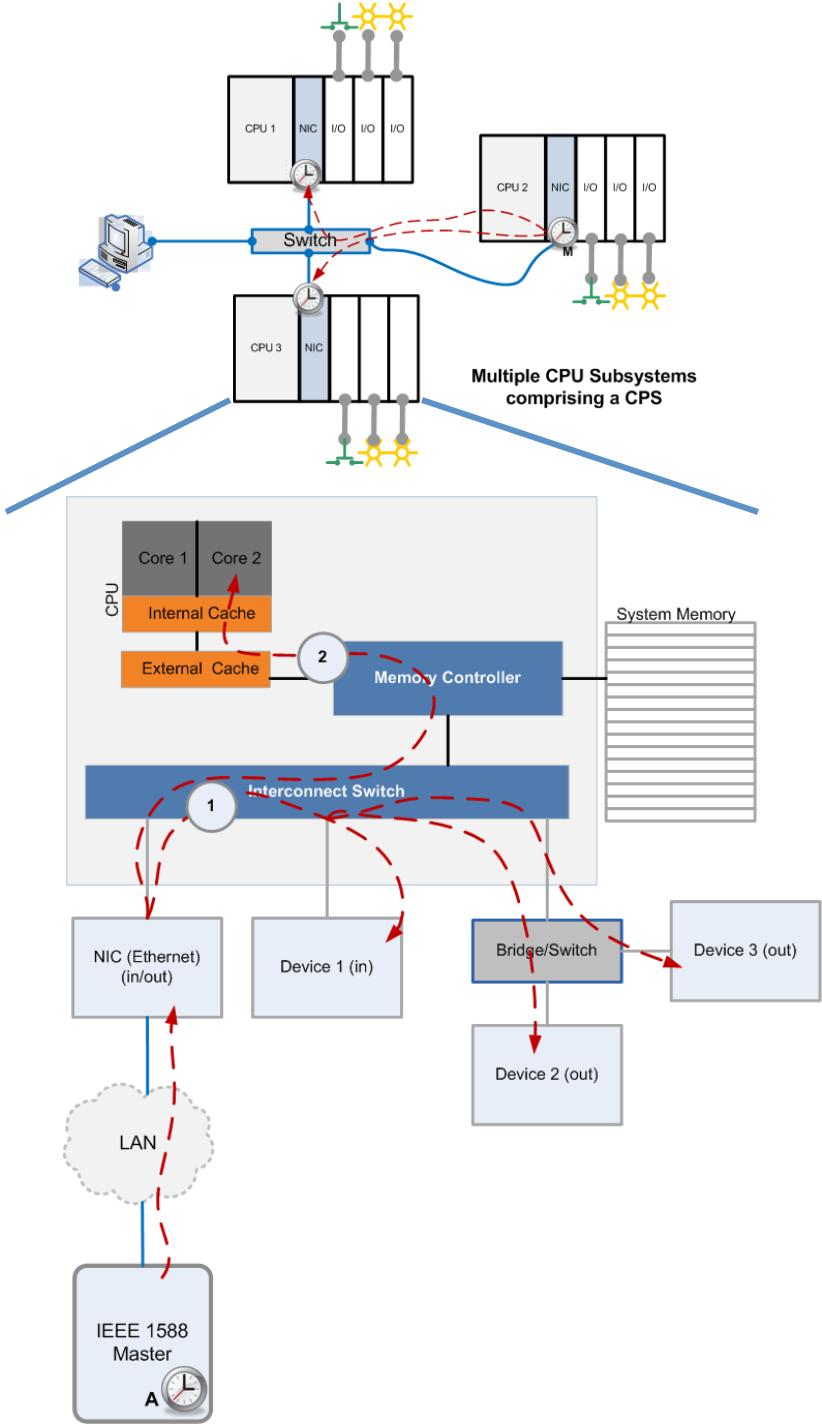


CPUs are the ubiquitous technology that interconnect IO and logic in these systems

Inside Each CPS unit (CPU Subsystem)

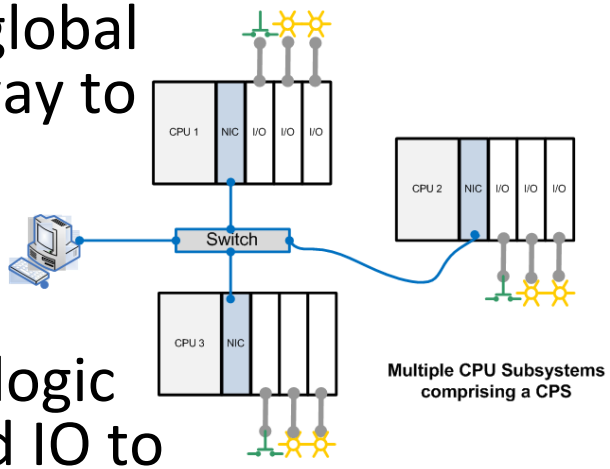
Network time is propagated …

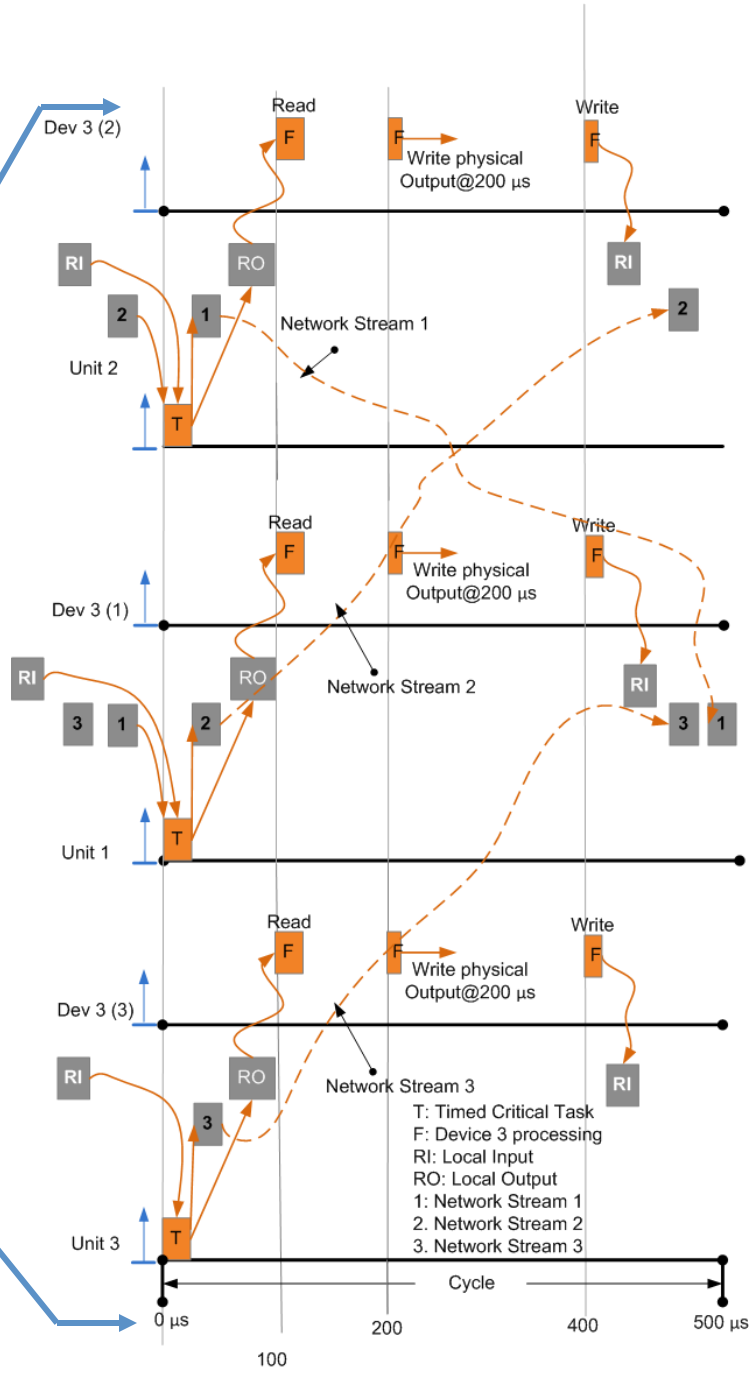
1. To the peripheral devices implementing IO interfaces
   - To synchronize IO across CPS precisely
2. To the CPU
   - To precisely coordinate logic execution with data transfer so that global CPS schedule can be maintained reliably



Challenges

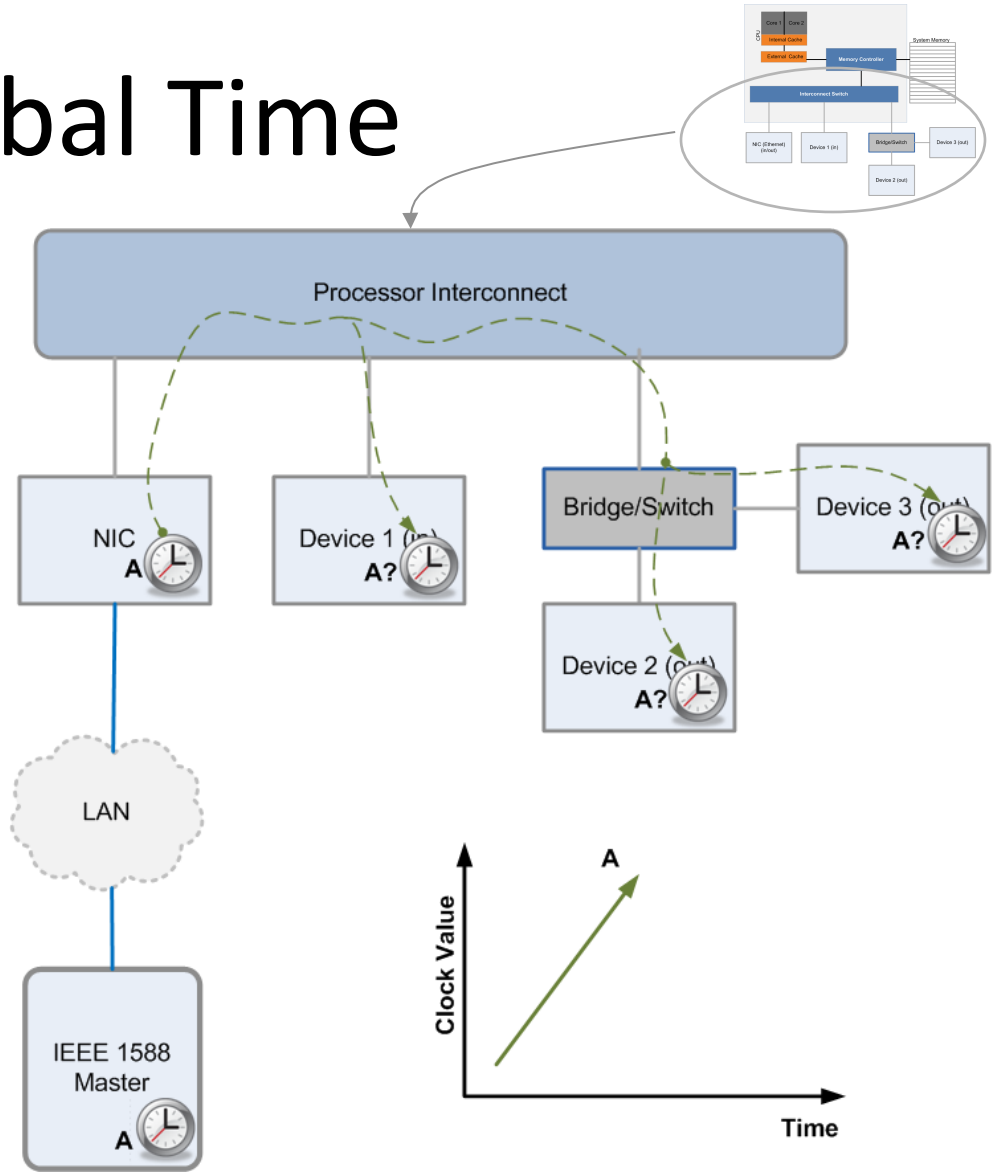
1. Maintaining clock precision while propagating global time all the way to the IO
2. Precisely coordinating logic execution and IO to reliably meet the global CPS schedule





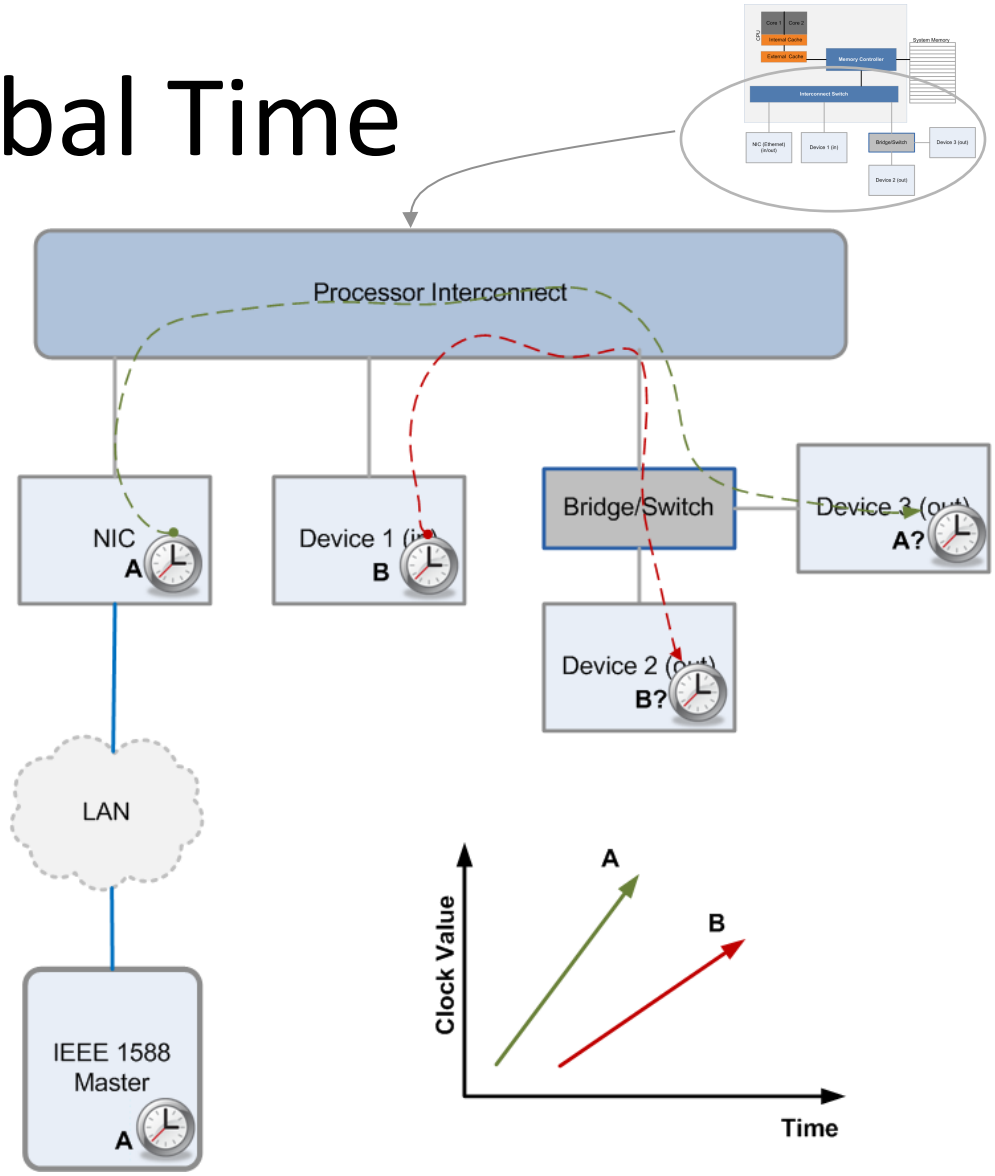
Propagation of Global Time

- **Possible Solution:** Using the NIC as the CPU Subsystem clock master
  - E.g., NIC handles propagation of Clock A to all devices
- **Problem:** Interconnect typically a switched fabric
  - E.g., Switching delays, contention, retries can delay synchronization packets
- **Effect:** Clock precision is compromised
  - E.g., Clock precision measured at IO may be 10 times worse than that that measured at the NIC



Propagation of Global Time

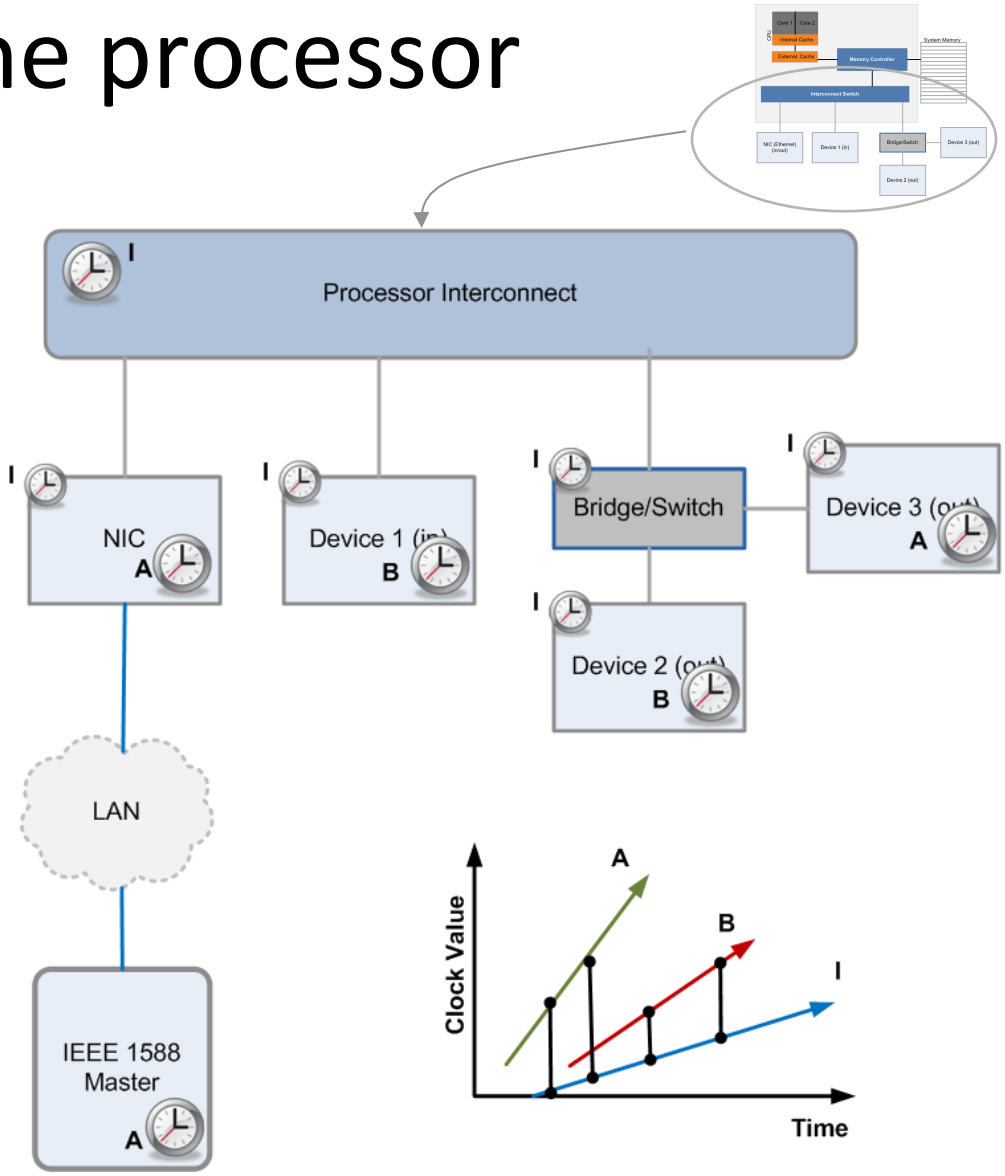
- How to maintain two independent clocks domains?
  - Very common in measurement/ instrumentation applications
- How to account for the differences in time and frequency of the clock domain?



Bringing Time into the processor interconnect

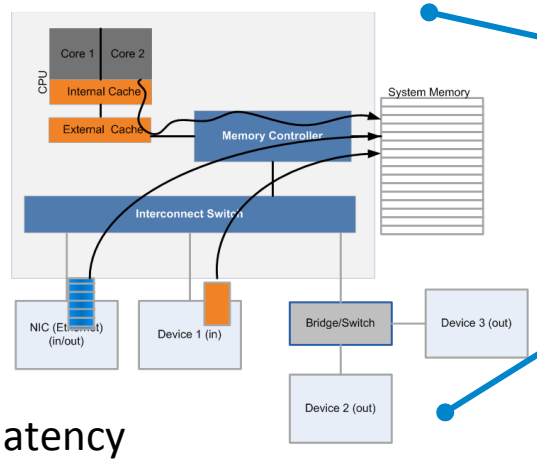
- Processor Interconnect Clock (I) propagated to all devices
- Device simultaneously captures I and either clock A or clock B
- HW/SW can compute time and frequency offsets across platforms
- Interop with external clocks OR use Interconnect clock directly

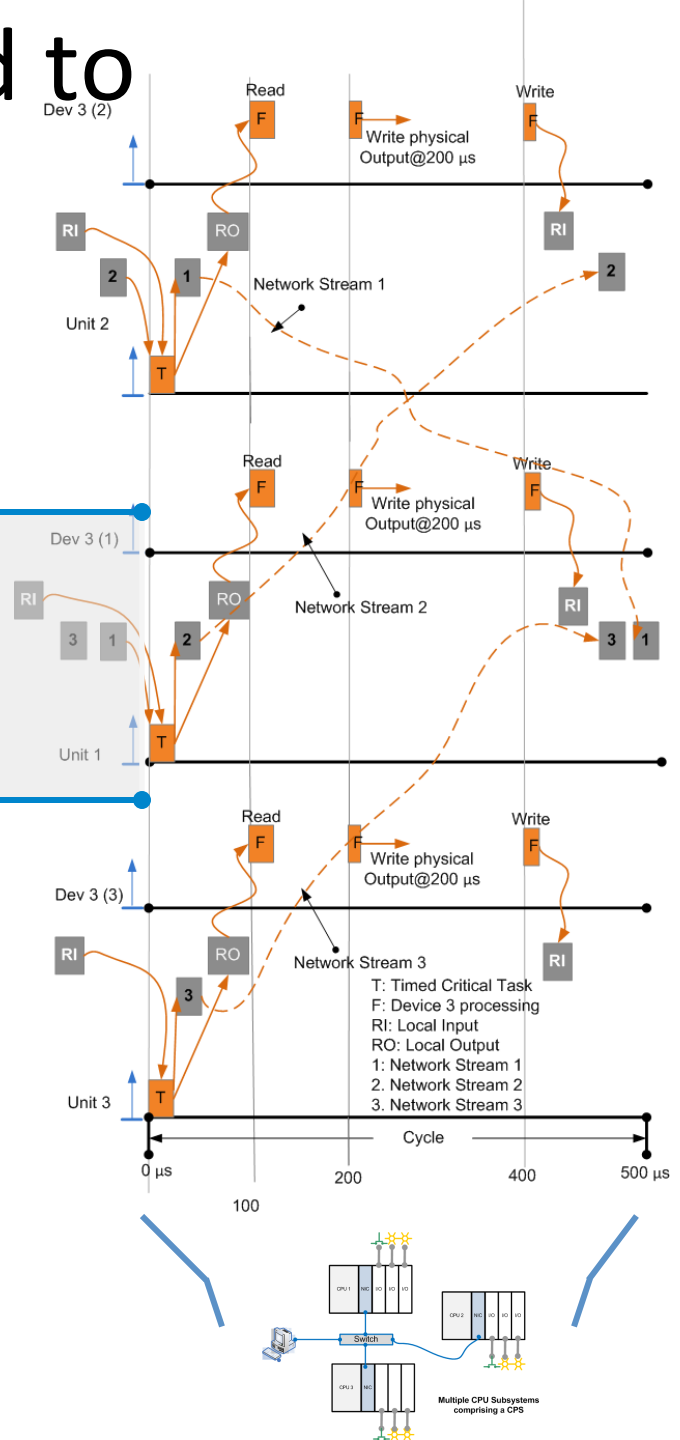
Enables network level clock precision to be propagated all the way to the IO



Why Global Schedule is hard to Achieve for Real-Time CPS?

- CPS need to handle both
  - Time Sensitive Data
    - — Periodic
    - — Fixed payload
    - — Bounded latency
  - Asynchronous Data
    - — Bursty
    - — Dynamic payload
    - — No hard bound on latency
    - — Requires confirmed transfer (retries)

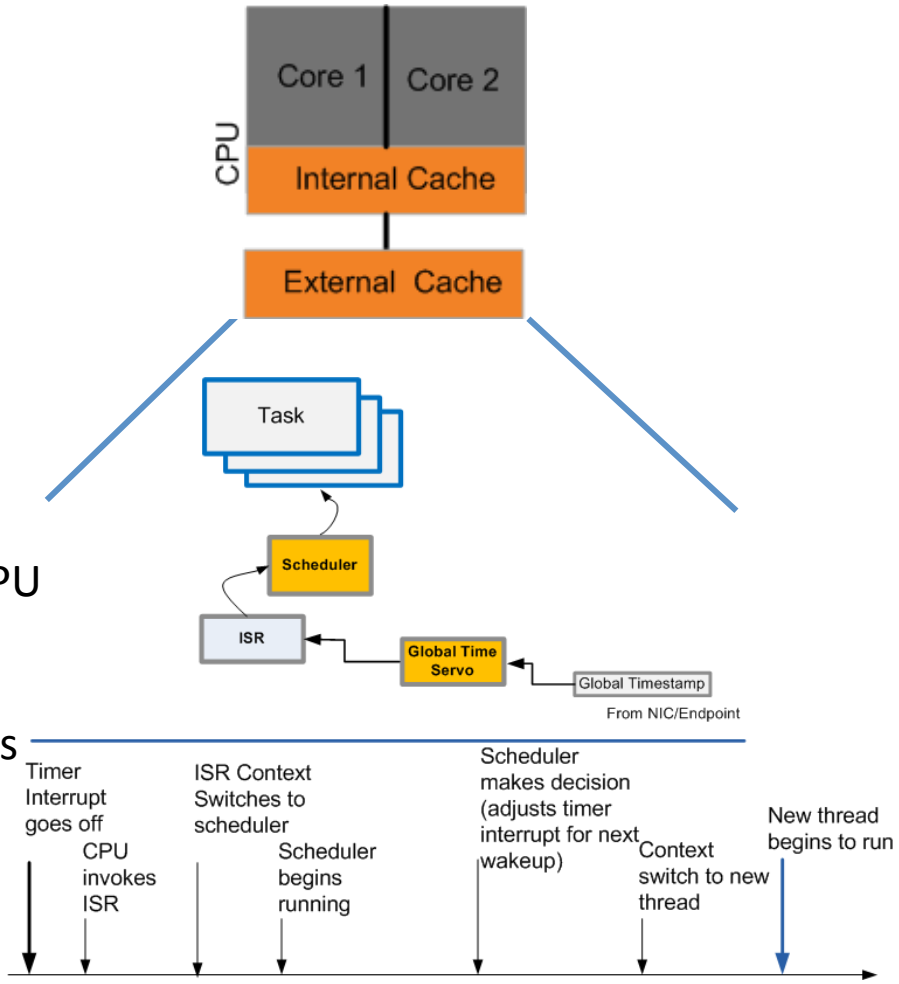




- **Problem:** Untimely arrival of asynchronous data in one unit can adversely impact the entire CPS schedule

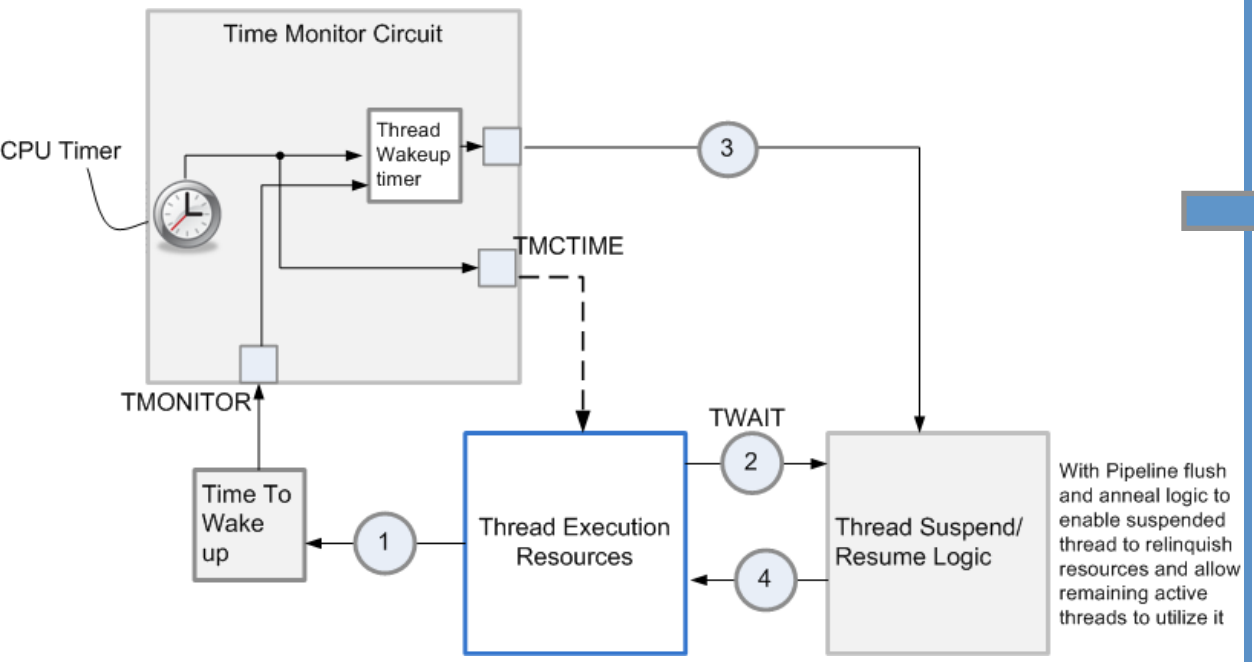
CPU Loosely Synchronized to Time

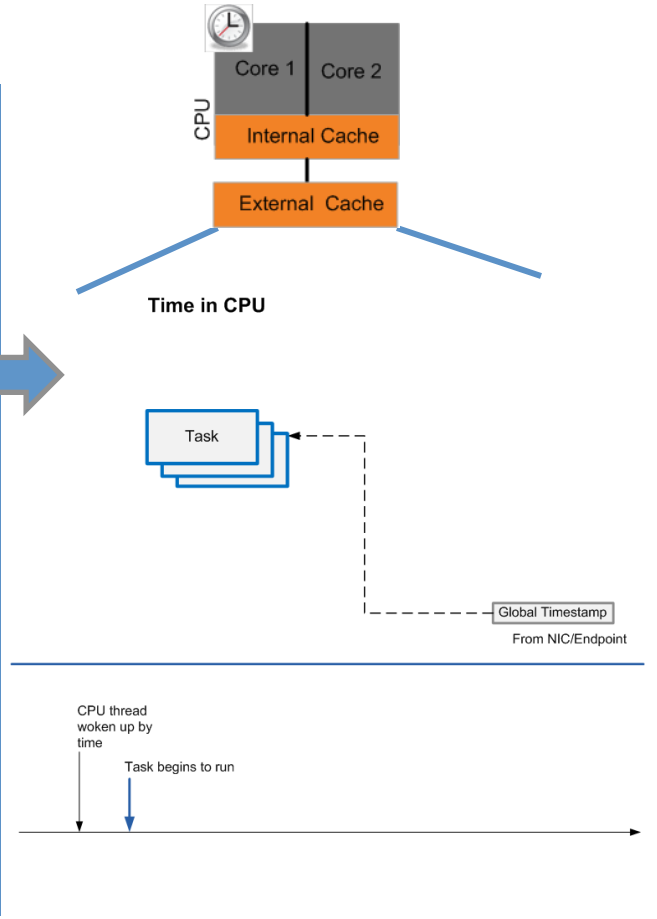
- CPU is not a processor interconnect peripheral
- Time based scheduling uses the CPU Timer Interrupt
  - Handling interrupts is inherently unpredictable
    - Servicing a timer interrupt can be delayed based on the state of the CPU
    - Multiple context switches required
    - Requirement for higher rates increases the frequency of interrupts
- Timer interrupt is required to be disciplined to synchronize with global time
- Difficult to synchronize execution of tasks across a CPS using this model



Bringing Time into the CPU

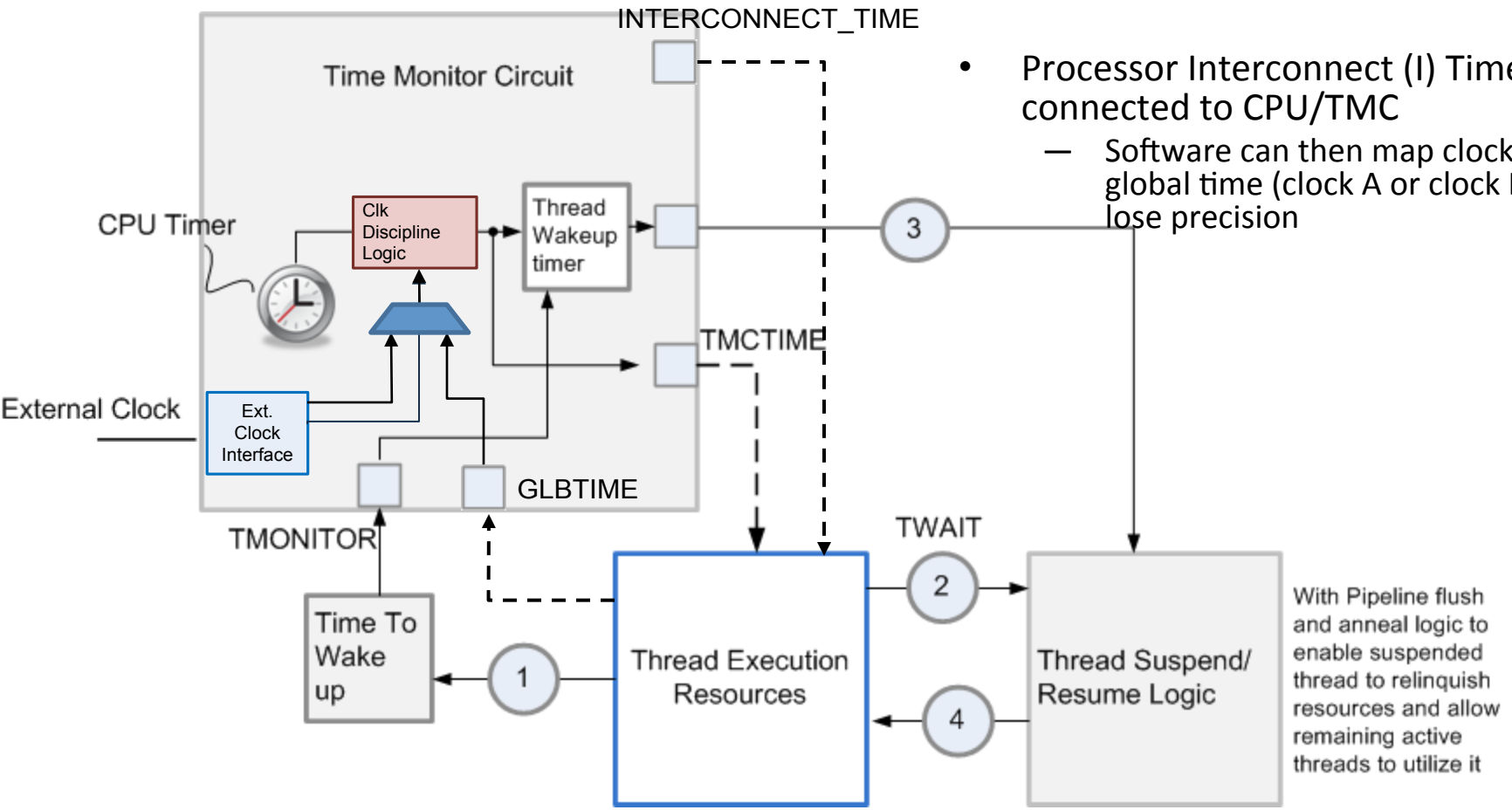
- Time Monitor Circuit Suspends/Resumes threads based on time
  - Suspended thread relinquishes all CPU resources to other active threads in core
  - Built on the Monitor-MWait instruction idea





Time Monitor Circuit Disciplined to Global Time

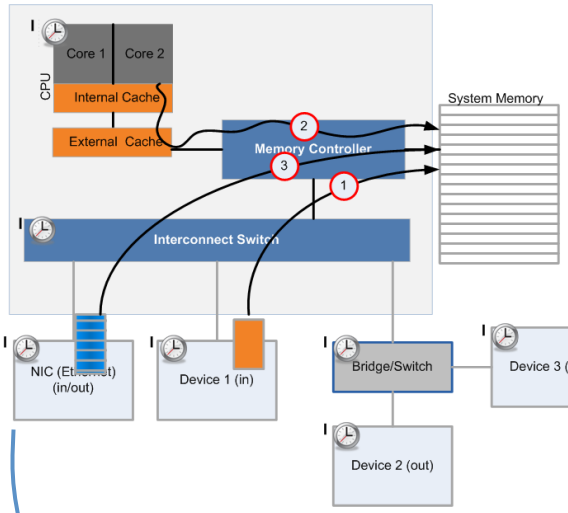
- External Clock (hardware signal based)
  - Signal provides epoch and clock
- Software based
  - Software provides global time
- Processor Interconnect (I) Time connected to CPU/TMC
  - Software can then map clock (I) to global time (clock A or clock B) and not lose precision

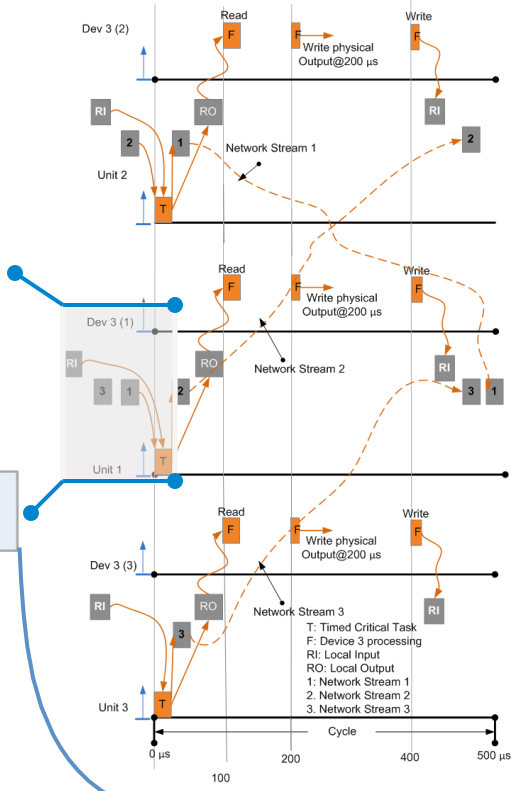


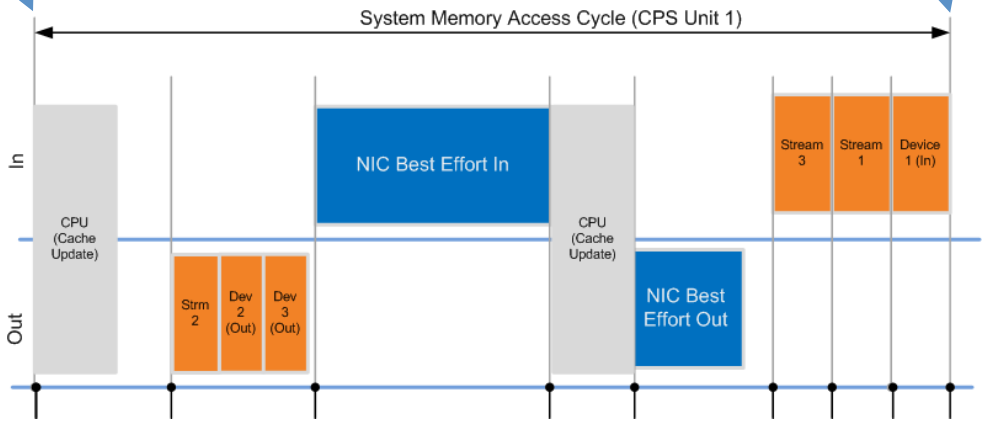
Time in CPU Subsystem

Improves Reliability and Precision of the Global CPS Schedule

- CPU executes logic on global time
- Bus Master capable peripherals use global time for data transfer
  - Eliminates need for CPU to manage data transfer thereby reducing scheduling overhead and jitter
- Enables propagation/integration of global schedule into subsystems thereby improving reliability
- Enables efficient bandwidth management for Asynchronous Data







Prototype and Results

- Architecture
  - Bringing Time into Processor Interconnects
    - Used custom signaling to get bus peripherals to share a common clock
  - Bringing Time into CPUs
    - Used Monitor-MWait instruction, and performed memory write to wake up thread using peripheral device (with shared clock)
  - Using Global Time for CPS Scheduling
    - Implemented timed data transfer logic for bus interface on FPGA
    - Computed and used global schedule to accommodate cache updates and asynchronous transfers
- Effect on Jitter
  - Maximum task jitter on CPU reduced to 1 µs (from 5 µs)
  - Frequency of jitter drops to 1 - 5% of the time (from 60 - 70%)

Summary

- Enabling time in CPUs and Processor Interconnects in Cyber Physical Systems …
  - Allows higher precision in IO synchronization
  - Improves reliability and precision of global schedule