The Challenges of Embedded System Design

Edward A. Lee

Robert S. Pepper Distinguished Professor
UC Berkeley

Invited Talk

Xilinx Emerging Technology Symposium (ETS)
San Jose, CA
February 1, 2012

Key Collaborators on work shown here:

- *Steven Edwards*
- *Jeff Jensen*
- *Sungjun Kim*
- *Isaac Liu*
- *Slobodan Matic*
- *Hiren Patel*
- *Jan Reinke*
- *Sanjit Seshia*
- *Mike Zimmer*
- *Jia Zou*

Abstract

All widely used software abstractions lack temporal semantics. The notion of correct execution of a program written in every widely-used programming language today does not depend on the temporal behavior of the program. But temporal behavior matters in almost all systems, particularly in networked systems. Even in systems with no particular real-time requirements, timing of programs is relevant to the value delivered by programs, and in the case of concurrent and distributed programs, also affects the functionality. In systems with real-time requirements, including most embedded systems, temporal behavior affects not just the value delivered by a system but also its correctness.

This talk will argue that time can and must become part of the semantics of programs for a large class of applications. It will argue that temporal behavior is not always just a *performance* metric, but is often rather a *correctness* criterion. To illustrate that this is both practical and useful, we will describe recent efforts at Berkeley in the design and analysis of timing-centric software systems. In particular, we will focus on two projects, PRET, which seeks to provide computing platforms with repeatable timing, and PTIDES, which provides a programming model for distributed real-time systems.

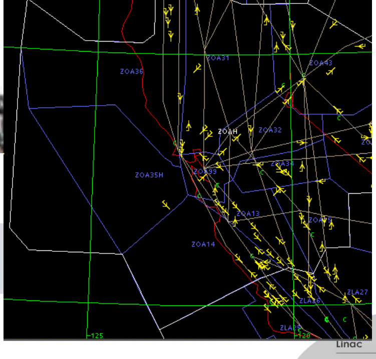
al Systems (CPS):

tworked computational

ysical systems

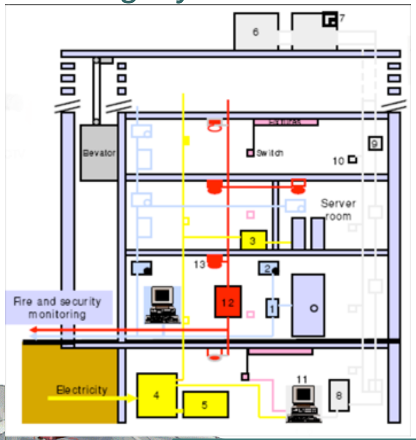
Automotive

Avionics

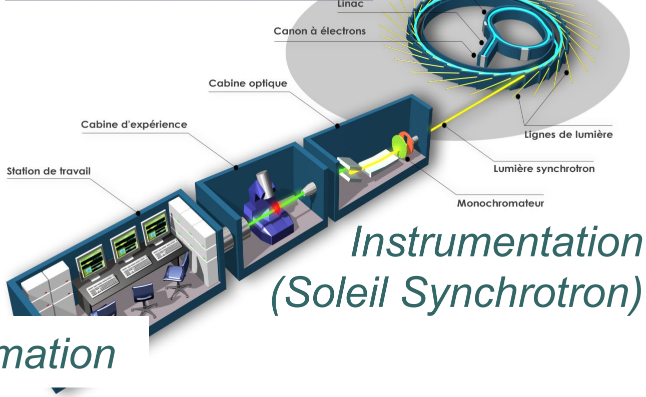


Transportation (Air traffic control at SFO)

Building Systems



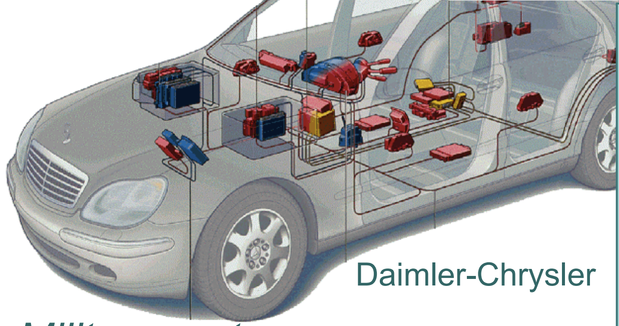
Telecommunications



Instrumentation (Soleil Synchrotron)

Factory automation

E-Corner, Siemens

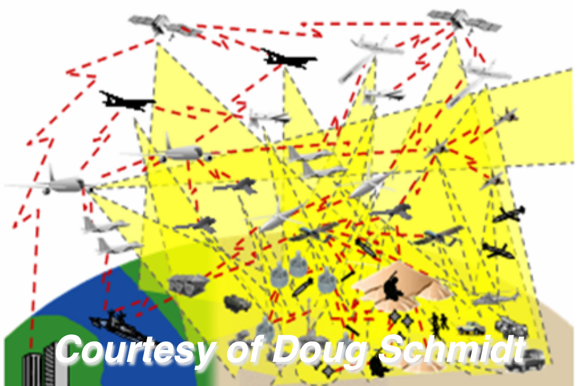


Daimler-Chrysler

Power generation and distribution

Courtesy of General Electric

Military systems:



Courtesy of Doug Schmidt

Courtesy of Kuka Robotics Corp.

Claim

For CPS, *programs* do not adequately specify *behavior*.

A Story

The Boeing 777 was Boeing's first fly-by-wire aircraft, controlled by software. It is deployed, appears to be reliable, and is succeeding in the marketplace. Therefore, it must be a success. However…

Boeing was forced to purchase and store an advance supply of the microprocessors that will run the software, sufficient to last for the estimated 50 year production run of the aircraft and another many years of maintenance.

Why?

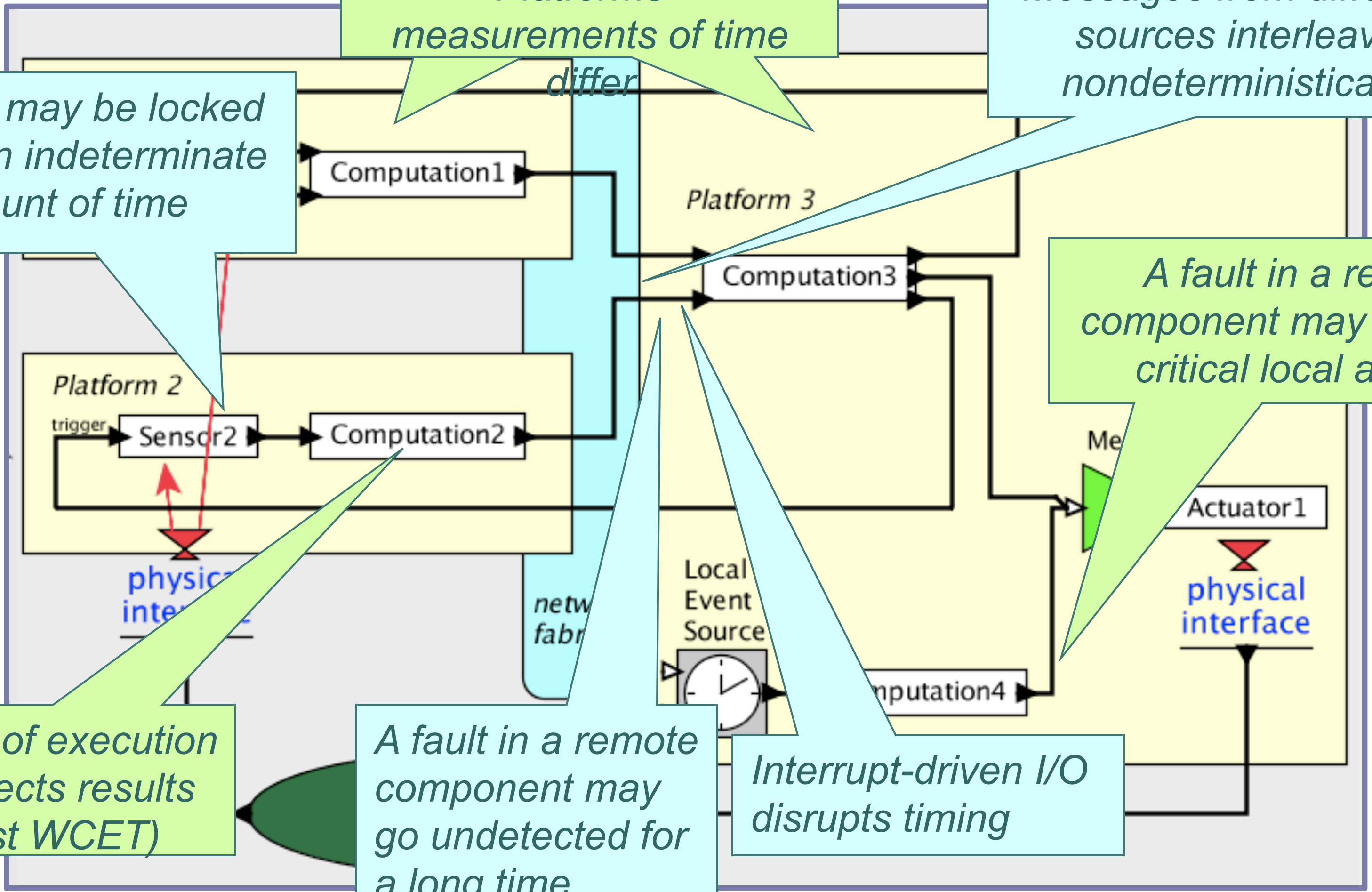
Lesson from this example:

Apparently, the software does not specify the behavior that has been validated and certified!

Unfortunately, this problem is very common, even with less safety-critical, certification-intensive applications. Validation is done on complete system implementations, not on software.

Structure of a Cyber-Physical System

Problems that complicate analysis of *system* behavior:



Etc…

Platforms' measurements of time differ

Messages from different sources interleave nondeterministically

Sensors may be locked out for an indeterminate amount of time

A fault in a remote component may disrupt a critical local activity

Variability of execution times affects results (not just WCET)

A fault in a remote component may go undetected for a long time

Interrupt-driven I/O disrupts timing

A Key Challenge:
Timing is not Part of Software Semantics

Correct execution of a program in C, C#, Java, Haskell, OCaml, etc. has nothing to do with how long it takes to do anything. All our computation and networking abstractions are built on this premise.

Programmers have to step *outside* the programming abstractions to specify timing behavior.

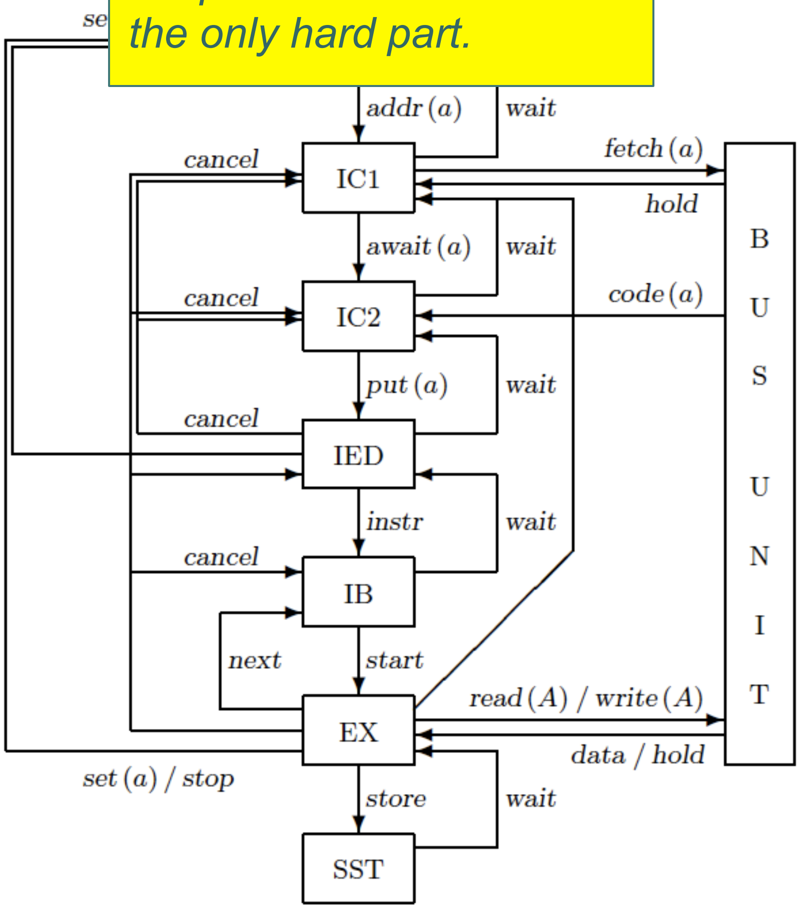
Execution-time analysis, by itself, does not solve the problem!

Analyzing software for timing behavior requires:

- Paths through the program (undecidable)
- Detailed model of microarchitecture
- Detailed model of the memory system
- Complete knowledge of execution context
- Many constraints on preemption/concurrency
- Lots of time and effort

And the result is valid only for that exact hardware and software!

Fundamentally, the ISA of the processor has failed to provide an adequate abstraction.



Our first goal is to reduce the problem so that this is the only hard part.

Wilhelm, et al. (2008). "The worst-case execution-time problem - overview of methods and survey of tools." ACM TECS 7 (3): p1-53.

Part 1: PRET Machines

- **PRE**cision-**T**imed processors = **PRET**
- **P**redictable, **RE**peatable **T**iming = **PRET**
- **P**erformance *with* **RE**peatable **T**iming = **PRET**

```
// Perform the convolution.
for (int i=0; i<10; i++) {
  x[i] = a[i]*b[j-i];
  // Notify listeners.
  notify(x[i]);
}
```

+

= PRET

Computing

With time

Dual Approach

- Rethink the ISA
  - Timing has to be a *correctness* property not a *performance* property.
- Implementation has to allow for multiple realizations and efficient realizations of the ISA
  - Repeatable execution times
  - Repeatable memory access times

Example of one sort of mechanism we would like:

```
tryin (500ms) {
   // Code block
} catch {
     panic();
}
```

```
jmp_buf  buf;

if ( !setjmp(buf) ){
  set_time r1, 500ms
  exception_on_expire r1, 0
  // Code block
  deactivate_exception 0
} else {
    panic();
}


exception_handler_0 () {
     longjmp(buf)
}
```

Pseudocode showing the mechanism in a mix of C and assembly.

If the code block takes longer than 500ms to run, then the panic() procedure will be invoked.

But then we would like to verify that panic() is never invoked!

Extending an ISA with Timing Semantics

[V1] Best effort:

```
set_time r1, 1s
// Code block
delay_until r1
```

[V2] Late miss detection

```
set_time r1, 1s
// Code block
branch_expired r1, <target>
delay_until r1
```

[V3] Immediate miss detection

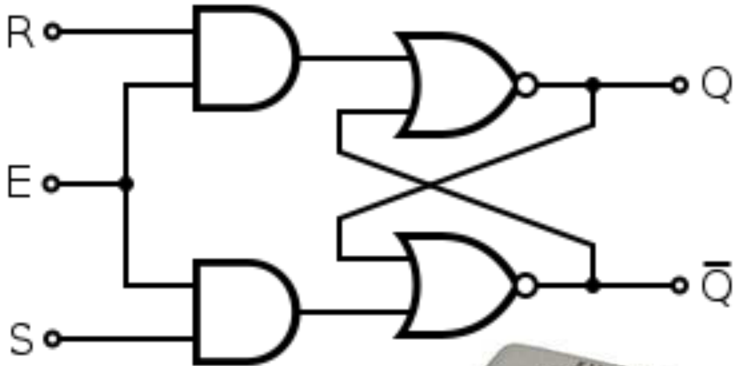
```
set_time r1, 1s
exception_on_expire r1, 1
// Code block
deactivate_exception 1
delay_until r1
```

[V4] Exact execution:

```
set_time r1, 1s
// Code block
MTFD r1
```

To provide timing guarantees, we need implementations that deliver repeatable timing

Fortunately, electronics technology delivers highly reliable and precise timing…



… but the overlaying software abstractions discard it. Chip architects heavily exploit the lack of temporal semantics.

```
// Perform the convolution.
for (int i=0; i<10; i++) {
  x[i] = a[i]*b[j-i];
  // Notify listeners.
  notify(x[i]);
}
```

To deliver repeatable timing, we have to rethink the microarchitecture

Challenges:

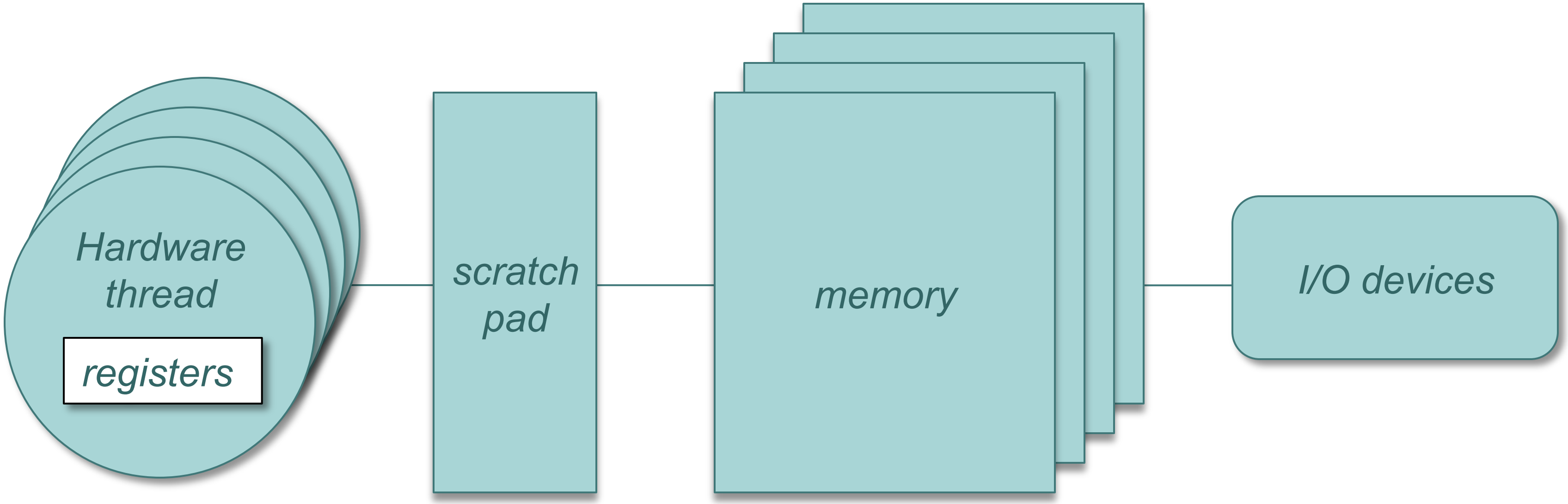
- Pipelining
- Memory hierarchy
- I/O (DMA, interrupts)
- Power management (clock and voltage scaling)
- On-chip communication
- Resource sharing (e.g. in multicore)

Our Current PRET Architecture

PTArm, a soft core on a Xilinx Virtex 5 FPGA

Note inverted memory compared to multicore!

Fast, close memory is shared, slow remote memory is private!



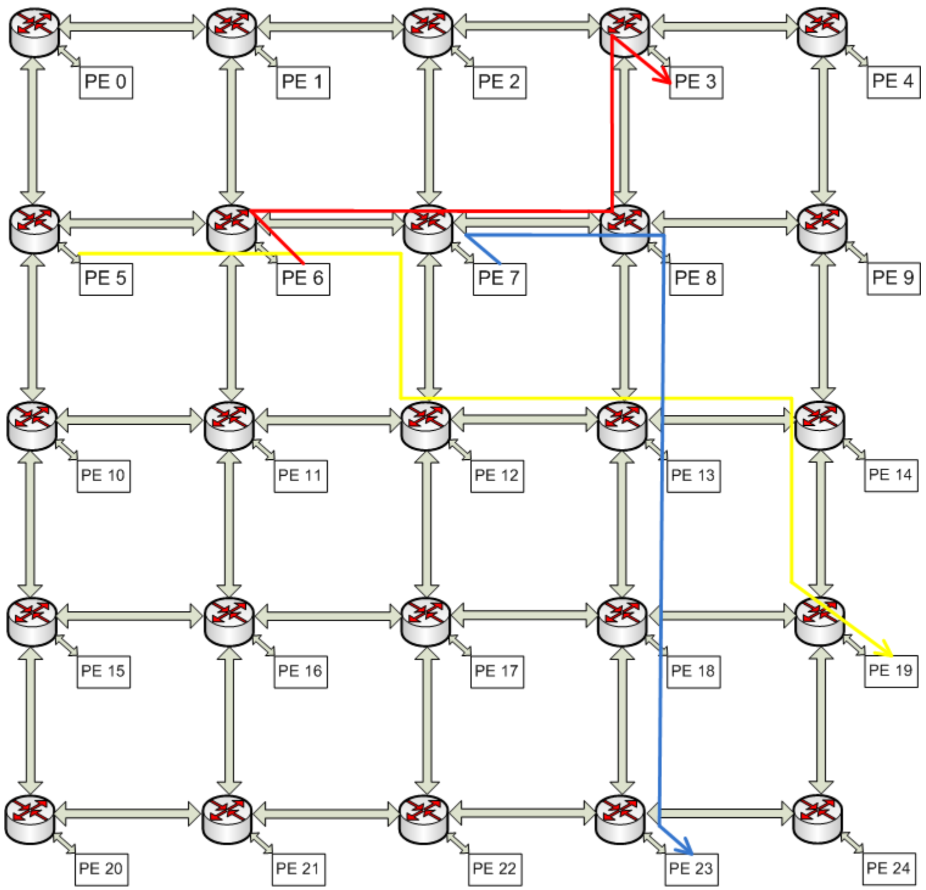
Interleaved pipeline with one set of registers per thread

SRAM scratchpad shared among threads

DRAM main memory, separate banks per thread

Multicore PRET

In today's multicore architectures, one thread can disrupt the timing of another thread *even if they are running on different cores and are not communicating*!



Our preliminary work shows that control over timing enables conflict-free routing of messages in a network on chip, making it possible to have non-interfering programs on a multicore PRET.

Status of the PRET project

- Results:
  - PTArm implemented on Xilinx Virtex 5 FPGA.
  - UNISIM simulator of the PTArm facilitates experimentation.
  - DRAM controller with repeatable timing and DMA support.
  - PRET-like utilities implemented on COTS Arm.
- Much still to be done:
  - Realize MTFD, interrupt I/O, compiler toolchain, scratchpad management, etc.

```
set_time r1, 1s
// Code block
MTFD r1
```

A Key Next Step: Parametric PRET Architectures

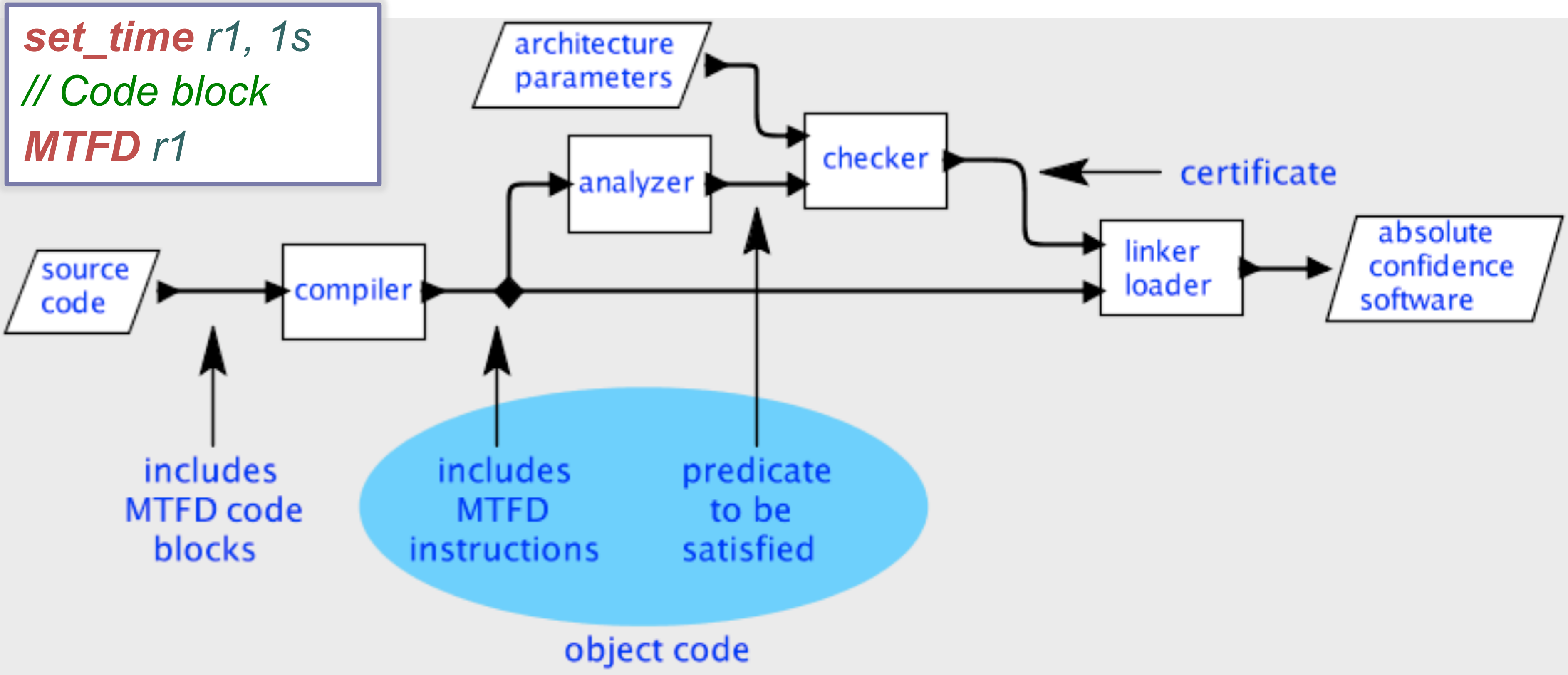
ISA that admits a variety of implementations:

- Variable clock rates and energy profiles
- Variable number of cycles per instruction
- Latency of memory access varying by address
- Varying sizes of memory regions
- …

A given program may meet deadlines on only some realizations of the same parametric PRET ISA.

Realizing the MTFD instruction on a parametric PRET machine

```
set_time r1, 1s
// Code block
MTFD r1
```


architecture parameters
checker
certificate
analyzer
source code
compiler
linker loader
absolute confidence software
includes MTFD code blocks
includes MTFD instructions
predicate to be satisfied
object code

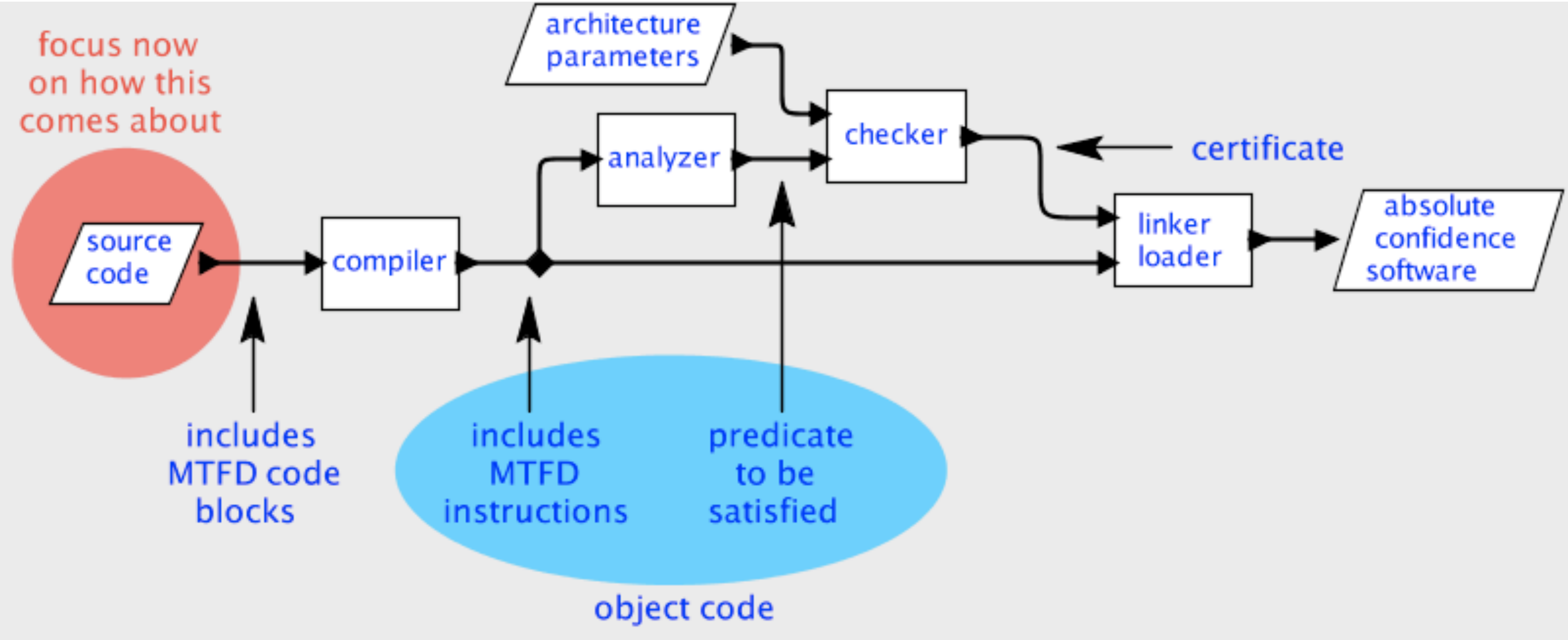
The goal is to make software that will run correctly on a variety of implementations of the ISA, and that correctness can be checked for each implementation.

PRET Publications

http://chess.eecs.berkeley.edu/pret/

- S. Edwards and E. A. Lee, "**The Case for the Precision Timed (PRET) Machine**," in the *Wild and Crazy Ideas* Track of DAC, June 2007.
- B. Lickly, I. Liu, S. Kim, H. D. Patel, S. A. Edwards and E. A. Lee, "**Predictable programming on a precision timed architecture**," CASES 2008.
- S. Edwards, S. Kim, E. A. Lee, I. Liu, H. Patel and M. Schoeberl, "**A Disruptive Computer Design Idea: Architectures with Repeatable Timing**," ICCD 2009.
- D. Bui, H. Patel, and E. Lee, "**Deploying hard real-time control software on chip-multiprocessors**," RTCSA 2010.
- Bui, E. A. Lee, I. Liu, H. D. Patel and J. Reineke, "**Temporal Isolation on Multiprocessing Architectures**," DAC 2011.
- J. Reineke, I. Liu, H. D. Patel, S. Kim, E. A. Lee, **PRET DRAM Controller: Bank Privatization for Predictability and Temporal Isolation** (to appear), CODES +ISSS, Taiwan, October, 2011.
- S. Bensalem, K. Goossens, C. M. Kirsch, R. Obermaisser, E. A. Lee, J. Sifakis, **Time-Predictable and Composable Architectures for Dependable Embedded Systems**, Tutorial Abstract (to appear), EMSOFT, Taiwan, October, 2011

Part 2: How to get the Source Code?

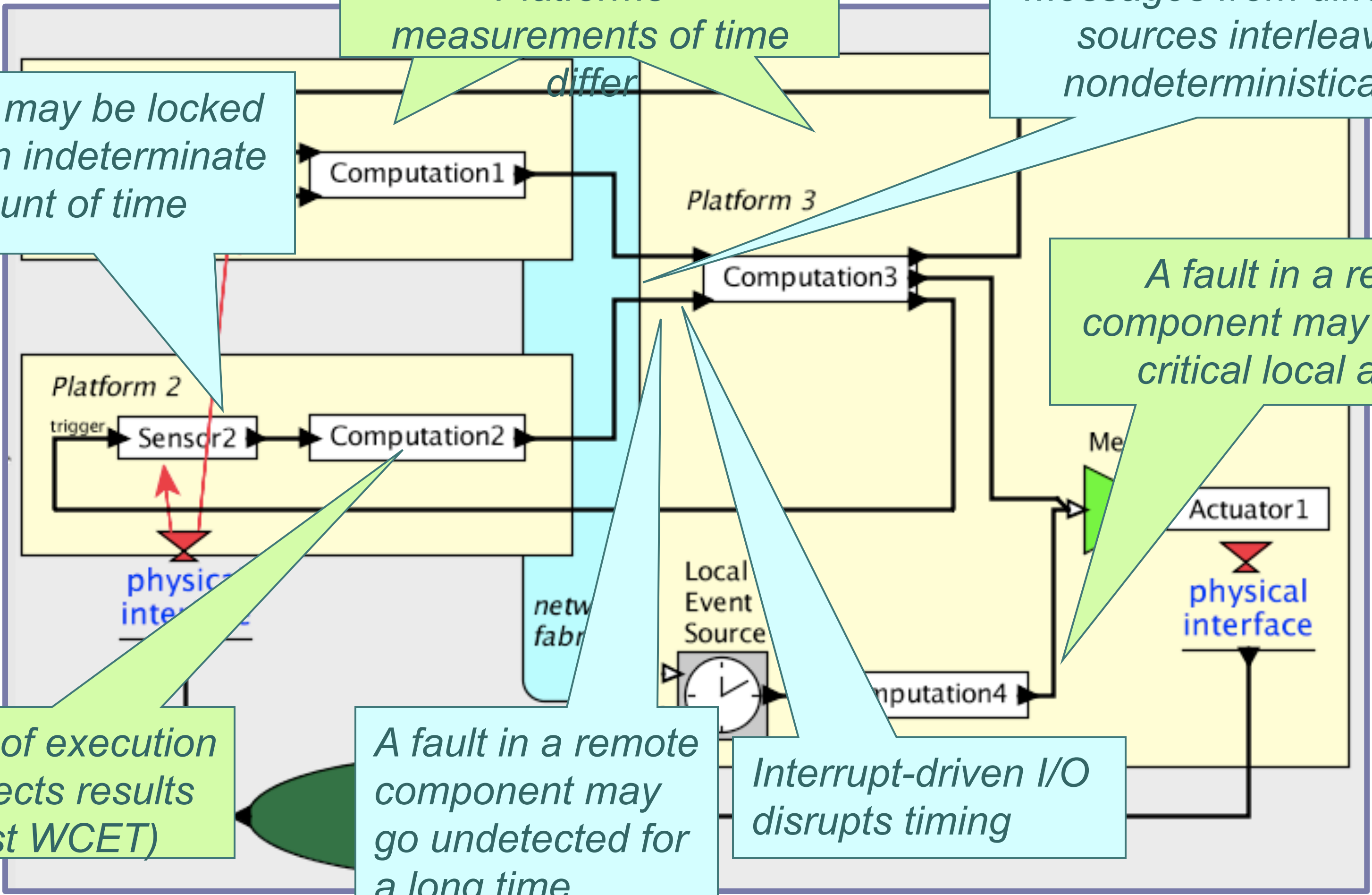

focus now on how this comes about
architecture parameters
checker
certificate
analyzer
source code
compiler
linker loader
absolute confidence software
includes MTFD code blocks
includes MTFD instructions
predicate to be satisfied
object code

The input (mostly likely C) will ideally be generated from a model, like Simulink or SCADE. The model specifies temporal behavior at a higher level than code blocks, and it specifies a concurrency model that can limit preemption points. However, Simulink and SCADE have naïve models of time.

Recall Structure of a Cyber-Physical System

Problems that complicate analysis of *system* behavior:

Etc…



Platforms' measurements of time differ

Messages from different sources interleave nondeterministically

Sensors may be locked out for an indeterminate amount of time

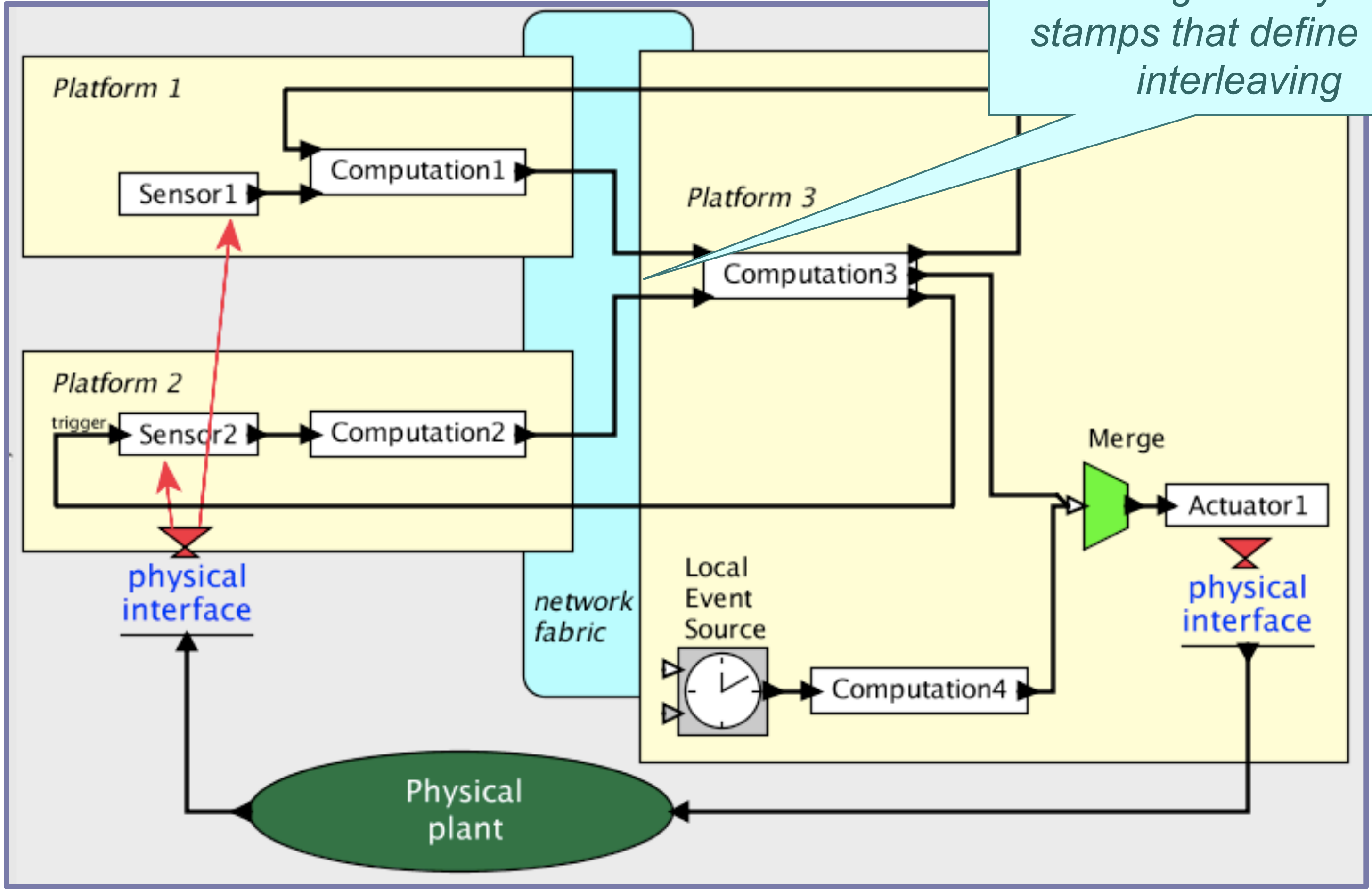
A fault in a remote component may disrupt a critical local activity

Variability of execution times affects results (not just WCET)

A fault in a remote component may go undetected for a long time

Interrupt-driven I/O disrupts timing

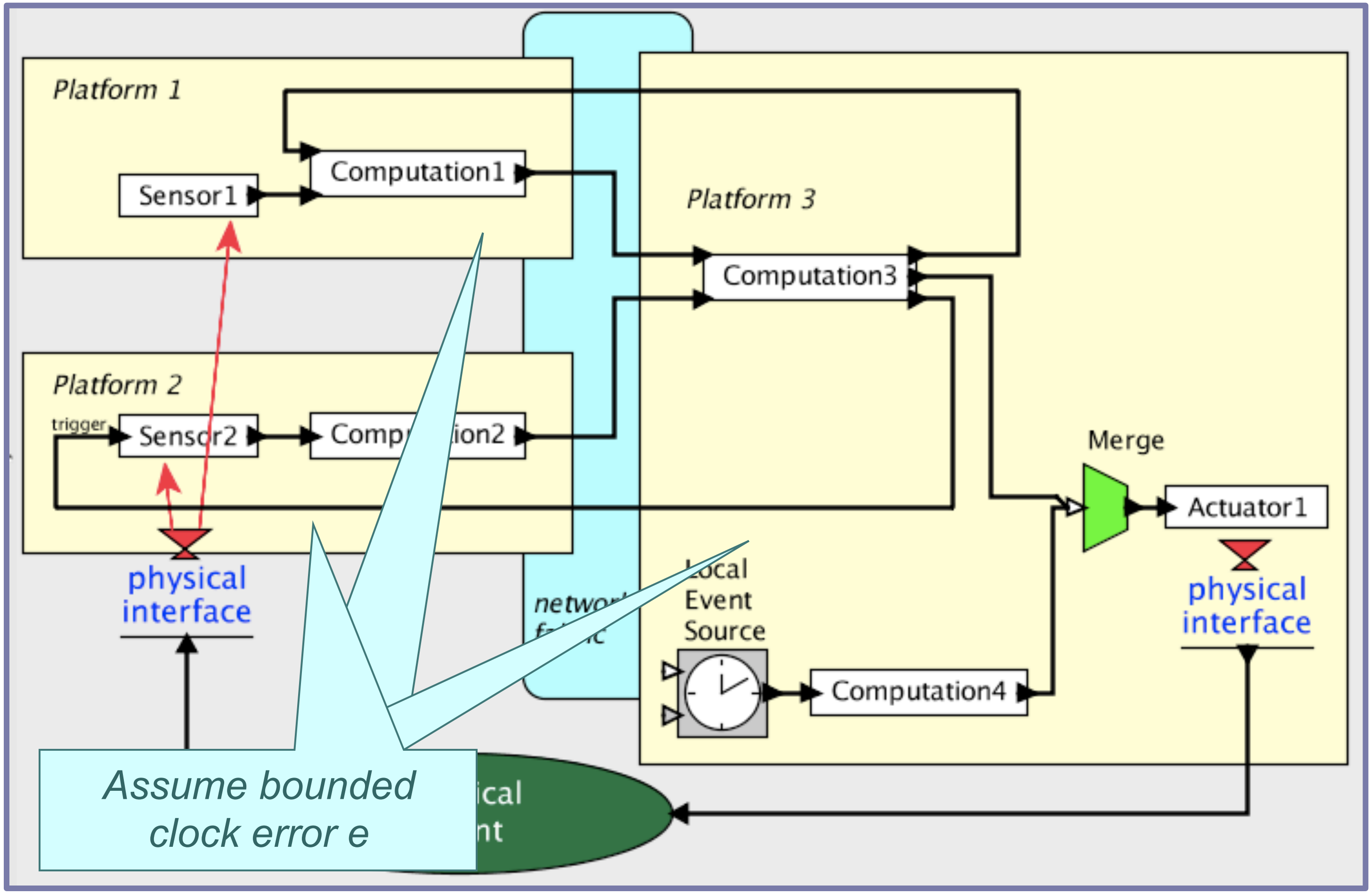
Ptides: First step: Time-stamped messages.



Messages carry time stamps that define their interleaving

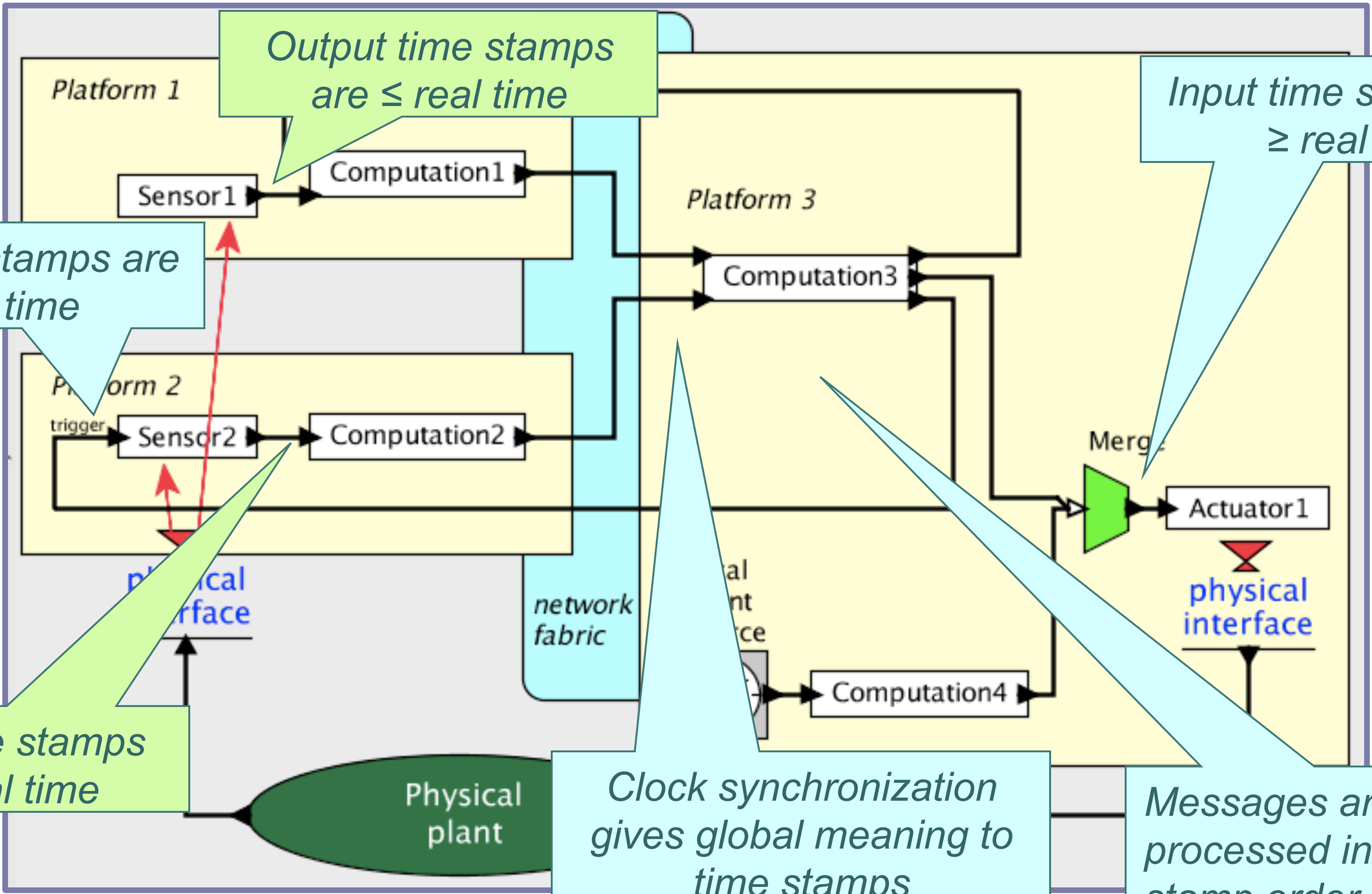
Ptides: Second step: Network time synchronization

GPS, NTP, IEEE 1588, time-triggered busses, etc., all provide some form of common time base. These are becoming fairly common.



Assume bounded clock error e

Ptides: Third step:
Bind time stamps to real time at sensors and actuators



Output time stamps are ≤ real time

Input time stamps are ≥ real time

Input time stamps are ≥ real time

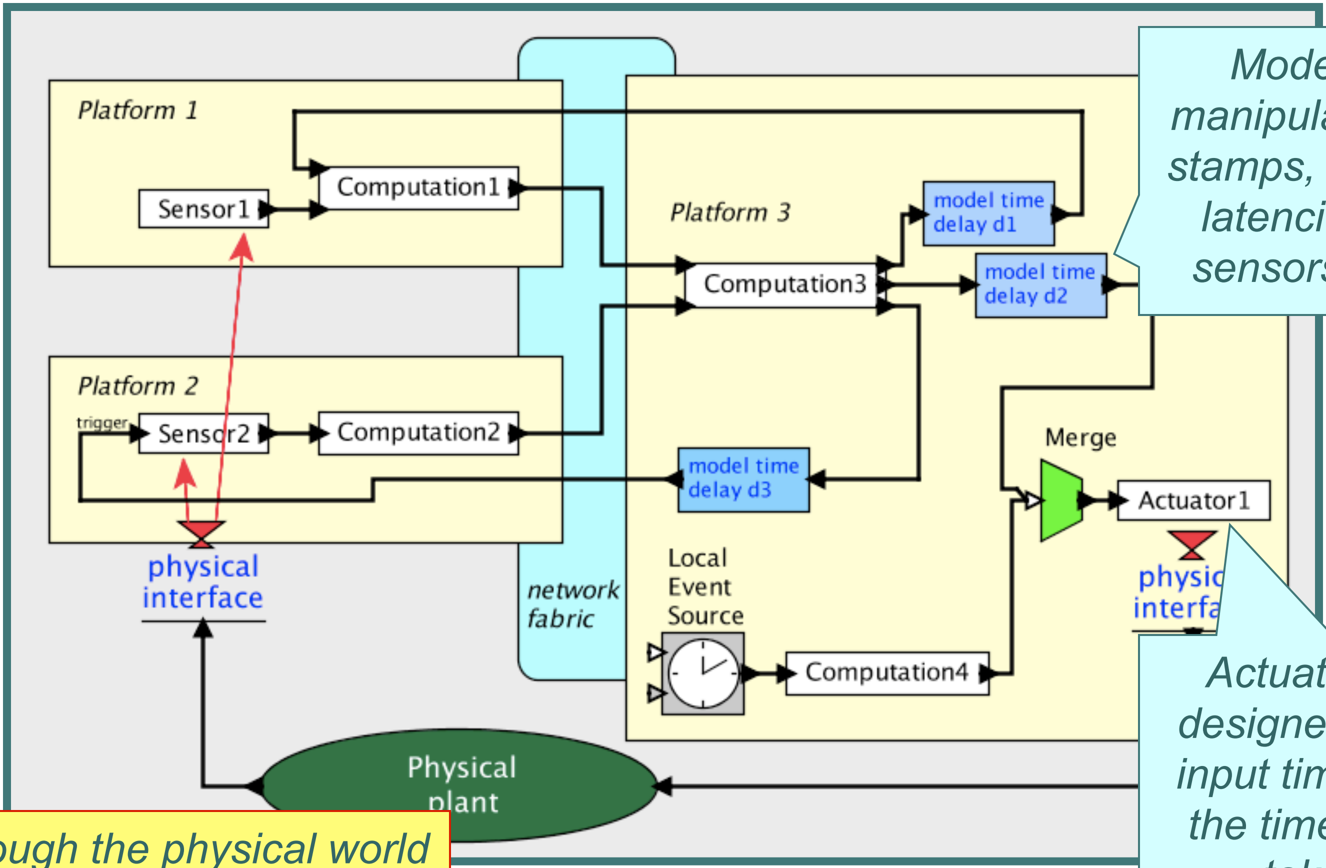
Output time stamps are ≤ real time

Clock synchronization gives global meaning to time stamps

Messages are processed in time-stamp order.

Ptides: Fourth step:
Specify latencies in the model

Global latencies between sensors and actuators become controllable, which enables analysis of system dynamics.



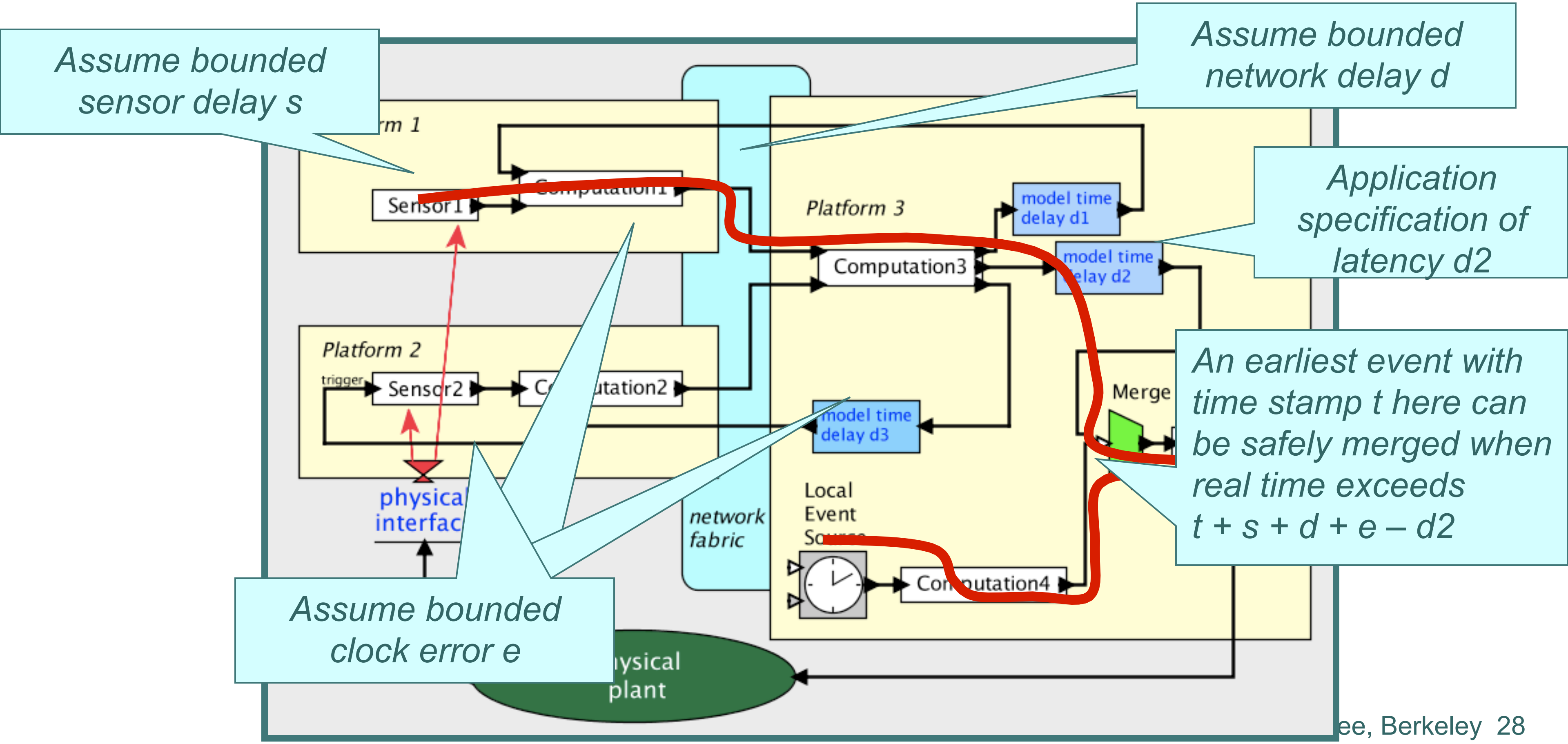
Model includes manipulations of time stamps, which control latencies between sensors and actors

Actuators may be designed to interpret input time stamps as the time at which to take action.

Feedback through the physical world

Ptides: Fifth step
Safe-to-process analysis (ensures determinacy)

Safe-to-process analysis guarantees that the generated code obeys time-stamp semantics (events are processed in time-stamp order), given some assumptions.



Assume bounded sensor delay s

Assume bounded network delay d

Application specification of latency d2

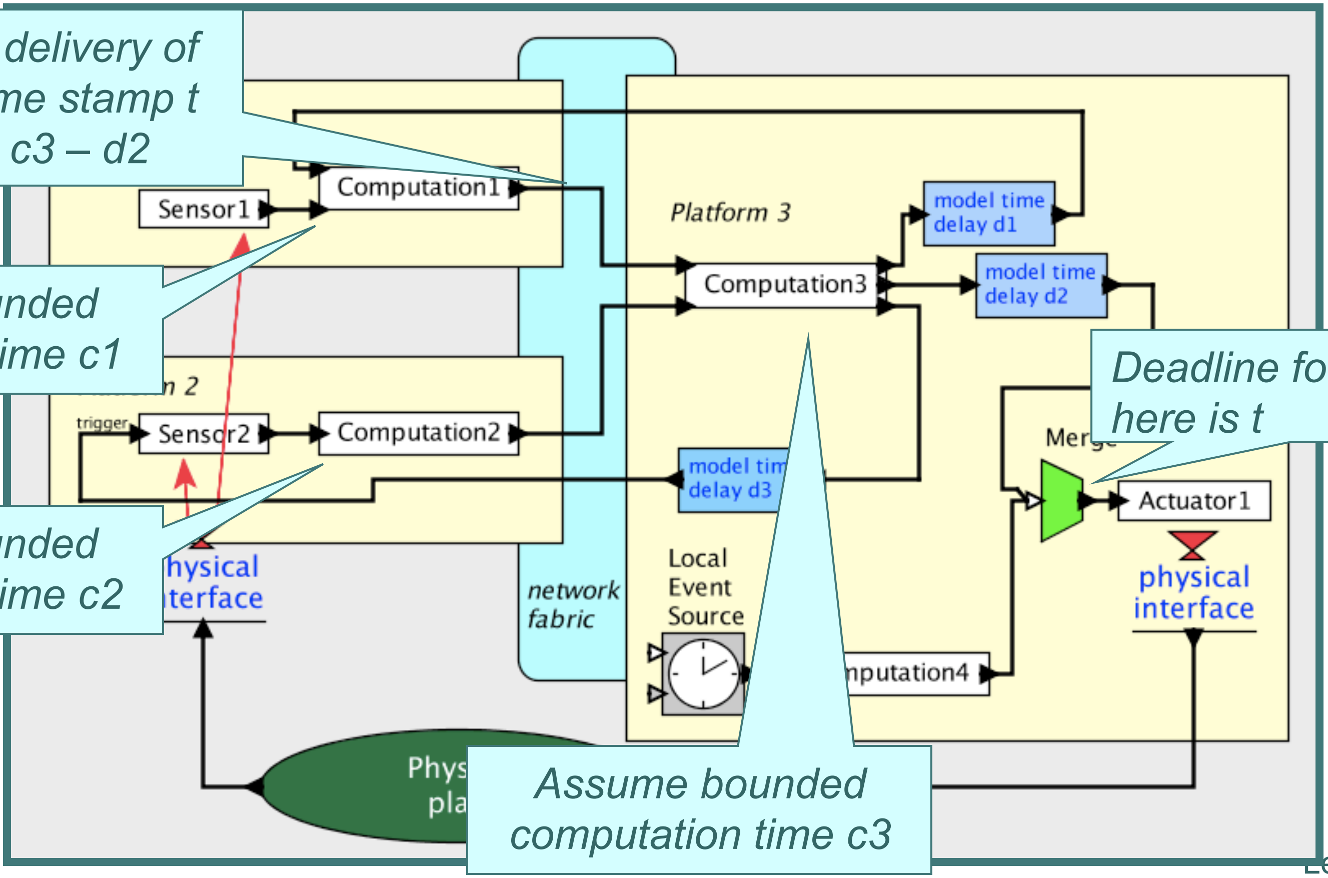
An earliest event with time stamp t here can be safely merged when real time exceeds t + s + d + e – d2

Assume bounded clock error e

Ptides Schedulability Analysis

Determine whether *deadlines* can be met

Schedulability analysis incorporates computation times to determine whether we can guarantee that deadlines are met.



Deadline for delivery of event with time stamp t here is t – c3 – d2

Assume bounded computation time c1

Assume bounded computation time c2

Assume bounded computation time c3

Deadline for delivery here is t

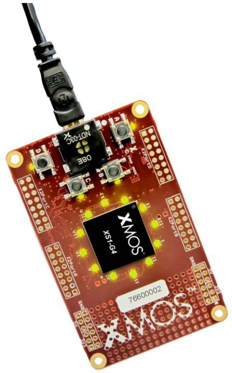
PtidyOS: A lightweight microkernel supporting Ptides semantics

An interesting property of PtidyOS is that despite being highly concurrent, preemptive, and EDF-based, it does not require threads.
A single stack is sufficient!

PtidyOS runs on

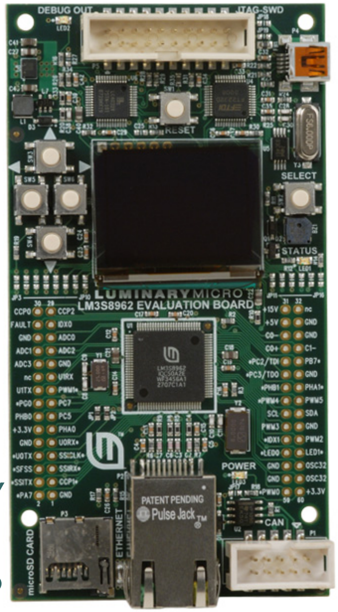
- Arm (Luminary Micro)
- Renesas
- XMOS

Occupies about 16 kbytes of memory.



XMOS development board with 4 XCores.

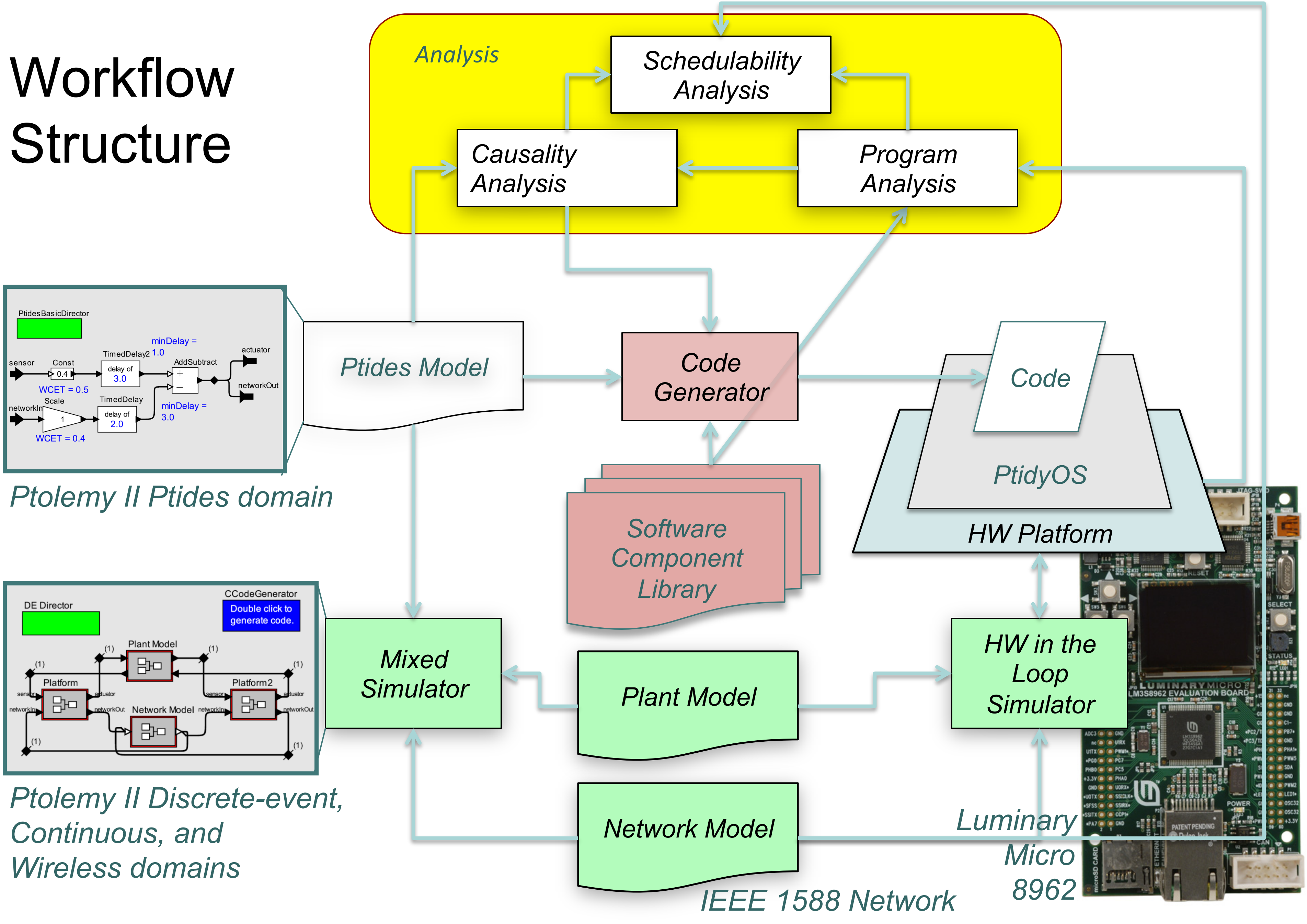
Renesas 7216 Demonstration Kit



Luminary Micro 8962

The name "PtidyOS" is a bow to TinyOS, which is a similar style of runtime kernel.

Workflow Structure



Analysis

Schedulability Analysis

Causality Analysis

Program Analysis

Ptides Model

Code Generator

Code

PtidyOS

HW Platform

Software Component Library

Ptolemy II Ptides domain

Mixed Simulator

Plant Model

HW in the Loop Simulator

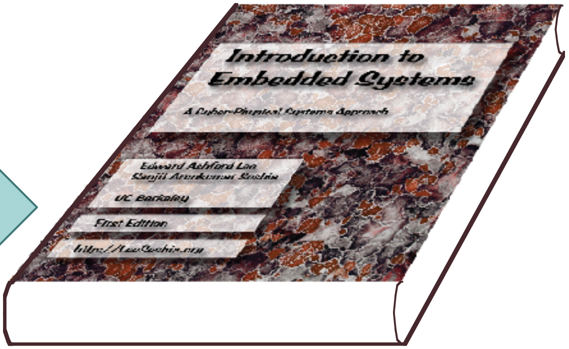
Network Model

Ptolemy II Discrete-event, Continuous, and Wireless domains

IEEE 1588 Network time protocol

Luminary Micro 8962

A Typical Cyber-Physical System
Printing Press

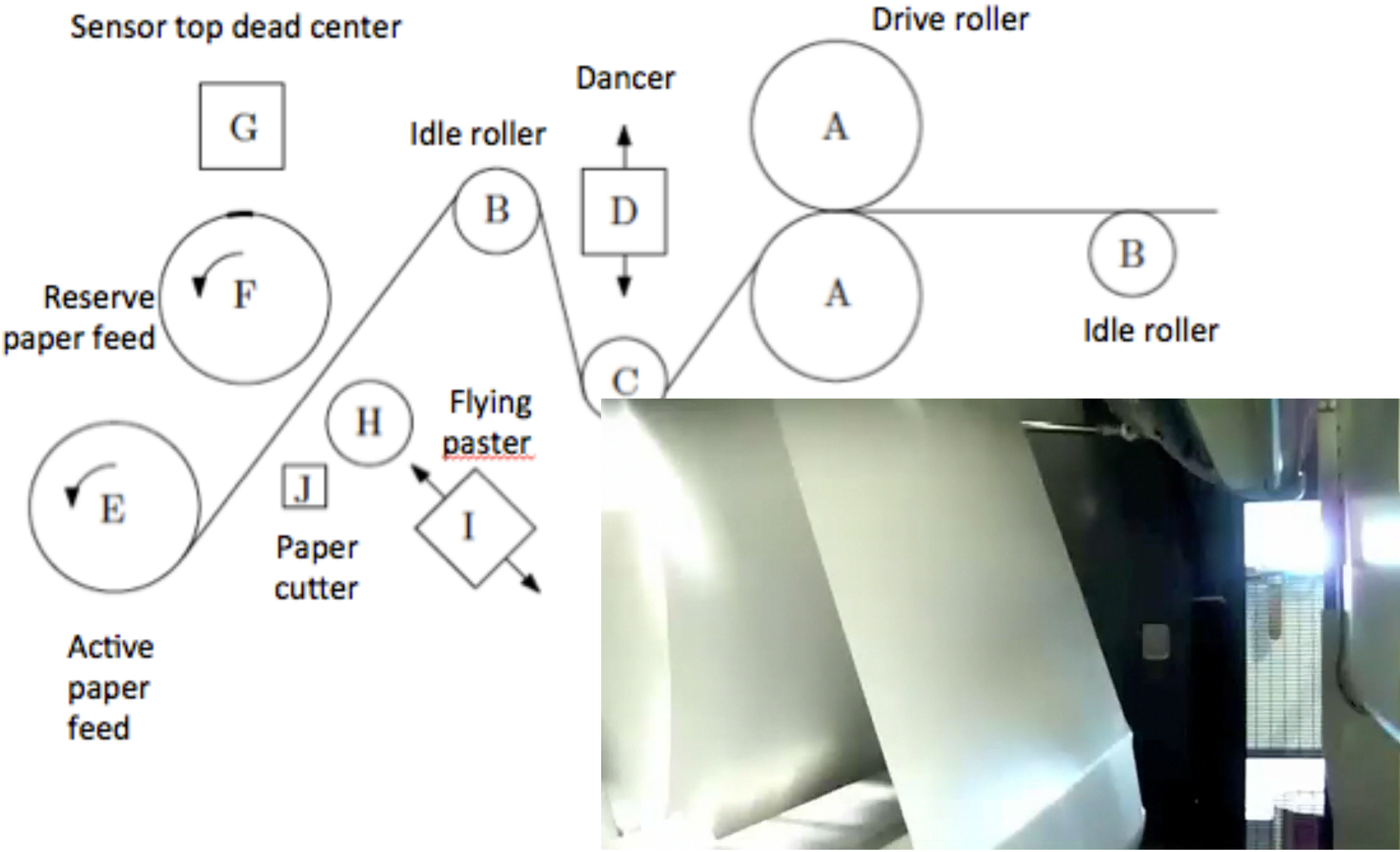


Bosch-Rexroth

- Application aspects
  - local (control)
  - distributed (coordination)
  - global (modes)
- Open standards (Ethernet)
  - Synchronous, Time-Triggered
  - IEEE 1588 time-sync protocol
- High-speed, high precision
  - Speed: 1 inch/ms
  - Precision: 0.01 inch
    -> Time accuracy: 10us

Goal: Orchestrated networked resources built with **sound design principles** on **suitable abstractions**

Example – Flying Paster



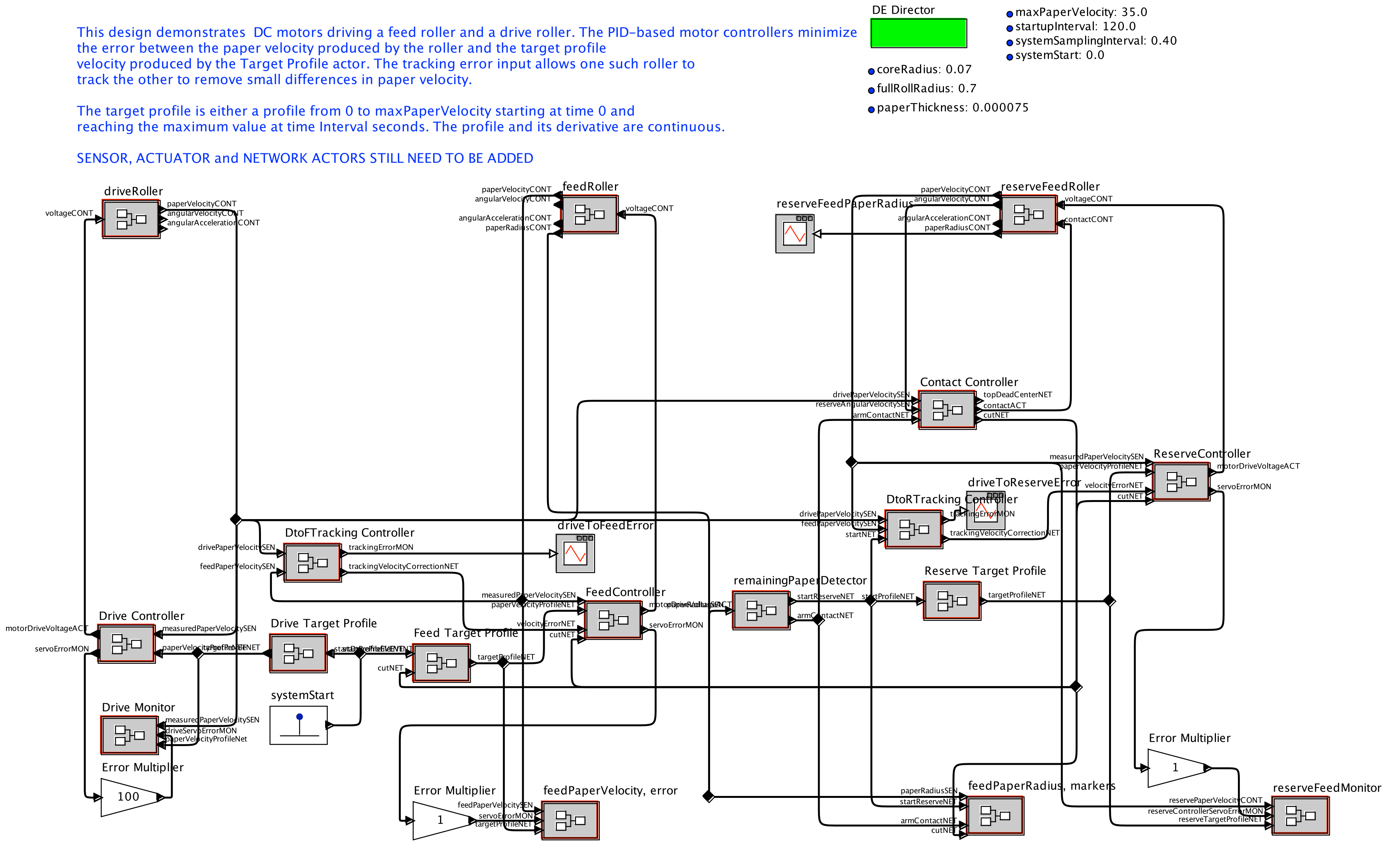
Source: http://offsetpressman.blogspot.com/2011/03/how-flying-paster-works.htr

Source: http://offsetpressman.blogspot.com/2011/03/how-flying-paster-works.html

Flying Paster

Model by Patricia Derler

Printing Press – Model in Ptolemy II



This design demonstrates DC motors driving a feed roller and a drive roller. The PID-based motor controllers minimize the error between the paper velocity produced by the roller and the target profile velocity produced by the Target Profile actor. The tracking error input allows one such roller to track the other to remove small differences in paper velocity.

The target profile is either a profile from 0 to maxPaperVelocity starting at time 0 and reaching the maximum value at time Interval seconds. The profile and its derivative are continuous.

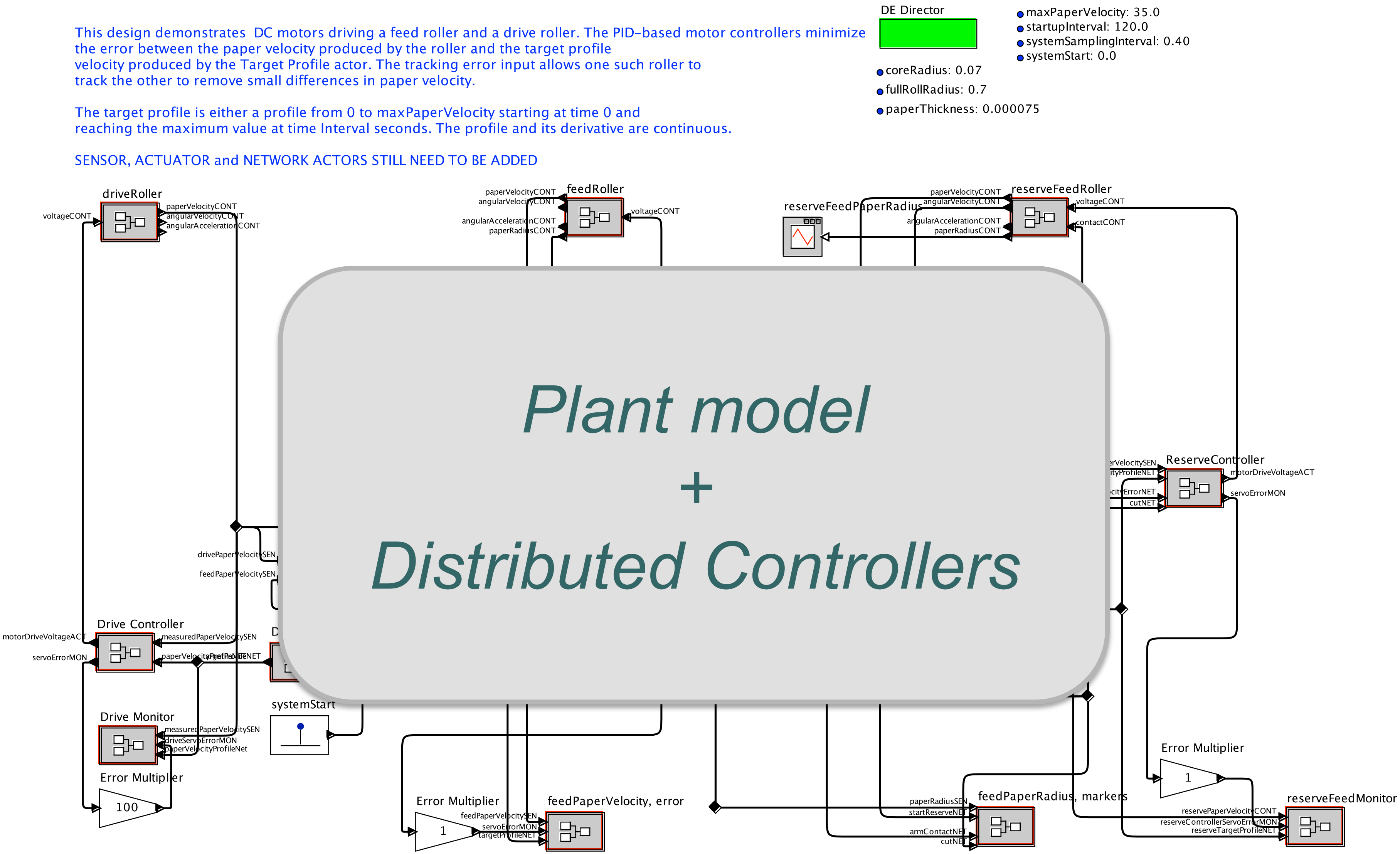
SENSOR, ACTUATOR and NETWORK ACTORS STILL NEED TO BE ADDED

DE Director

- maxPaperVelocity: 35.0
- startupInterval: 120.0
- systemSamplingInterval: 0.40
- systemStart: 0.0
- coreRadius: 0.07
- fullRollRadius: 0.7
- paperThickness: 0.000075

Model by Patricia Derler

Printing Press – Model in Ptolemy II



This design demonstrates DC motors driving a feed roller and a drive roller. The PID-based motor controllers minimize the error between the paper velocity produced by the roller and the target profile velocity produced by the Target Profile actor. The tracking error input allows one such roller to track the other to remove small differences in paper velocity.

The target profile is either a profile from 0 to maxPaperVelocity starting at time 0 and reaching the maximum value at time Interval seconds. The profile and its derivative are continuous.

SENSOR, ACTUATOR and NETWORK ACTORS STILL NEED TO BE ADDED

DE Director

- maxPaperVelocity: 35.0
- startupInterval: 120.0
- systemSamplingInterval: 0.40
- systemStart: 0.0
- coreRadius: 0.07
- fullRollRadius: 0.7
- paperThickness: 0.000075

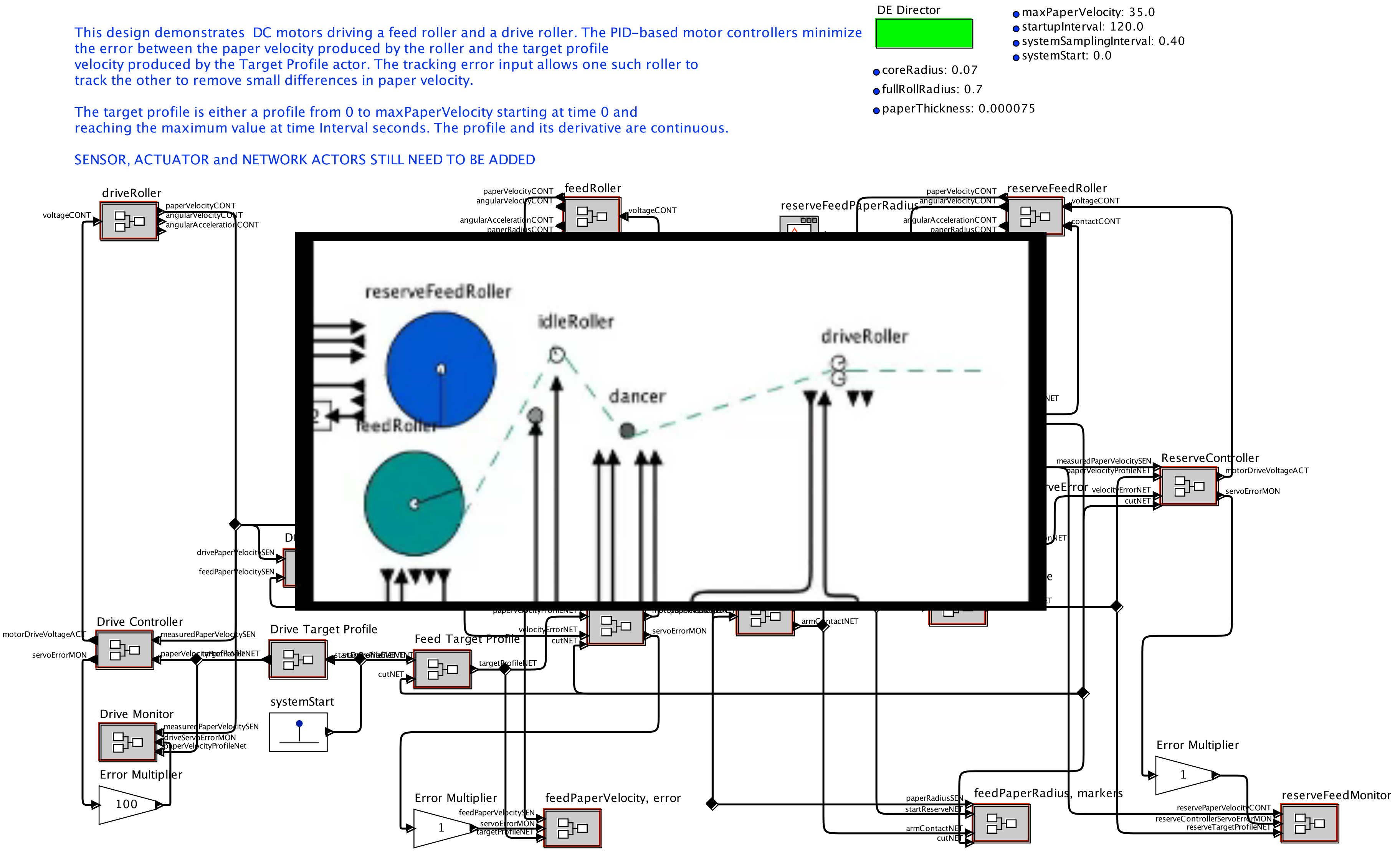
Plant model

+

Distributed Controllers

Model by Patricia Derler

Printing Press – Model in Ptolemy II



This design demonstrates DC motors driving a feed roller and a drive roller. The PID-based motor controllers minimize the error between the paper velocity produced by the roller and the target profile velocity produced by the Target Profile actor. The tracking error input allows one such roller to track the other to remove small differences in paper velocity.

The target profile is either a profile from 0 to maxPaperVelocity starting at time 0 and reaching the maximum value at time Interval seconds. The profile and its derivative are continuous.

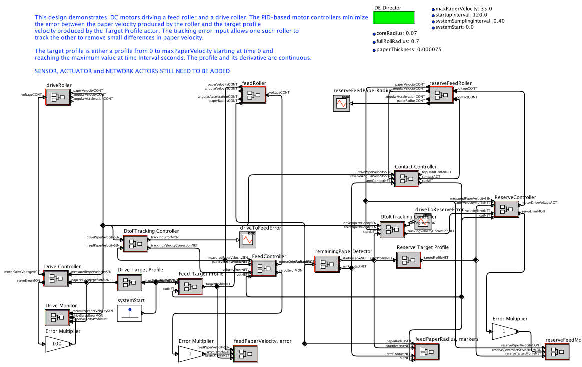
SENSOR, ACTUATOR and NETWORK ACTORS STILL NEED TO BE ADDED

DE Director 🟩

- maxPaperVelocity: 35.0
- startupInterval: 120.0
- systemSamplingInterval: 0.40
- systemStart: 0.0
- coreRadius: 0.07
- fullRollRadius: 0.7
- paperThickness: 0.000075

Determinate timing at sensors and actuators

Platform independent model of functional and timing behavior



This design demonstrates DC motors driving a feed roller and a drive roller. The PID-based motor controllers minimize the error between the paper velocity produced by the roller and the target profile velocity produced by the Target Profile actor. The tracking error input allows one such roller to track the other to remove small differences in paper velocity.

The target profile is either a profile from 0 to maxPaperVelocity starting at time 0 and reaching the maximum value at time Interval seconds. The profile and its derivative are continuous.

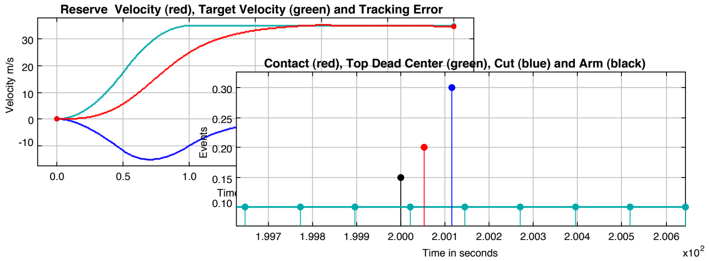
SENSOR, ACTUATOR and NETWORK ACTORS STILL NEED TO BE ADDED

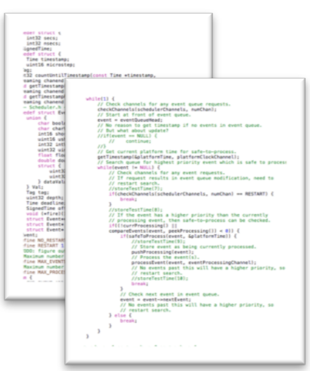
DE Director

maxPaperVelocity: 35.0
startupInterval: 120.0
systemSamplingInterval: 0.40
systemStart: 0.0
coreRadius: 0.07
fullRollRadius: 0.7
paperThickness: 0.000075

Code Generation to multiple target platforms

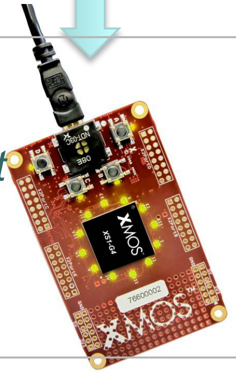
Simulation





Same I/O behavior w.r.t. value and timing

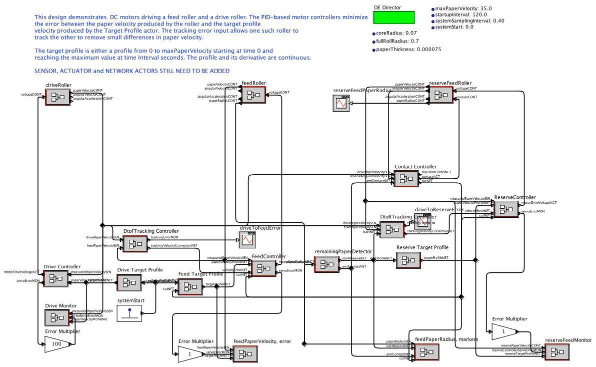
e.g.: XMOS development board with 4 XCores.



e.g.: Renesas 7216 Demonstration Kit

Determinate timing at sensors and actuators

Platform independent model of functional and timing behavior



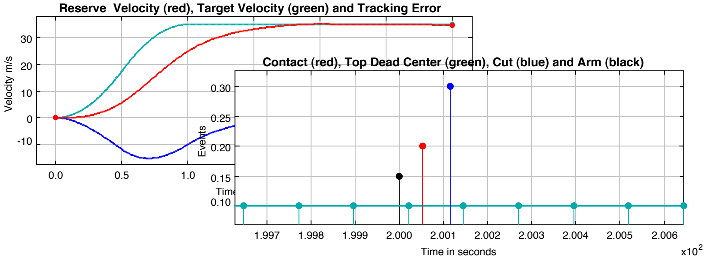
This design demonstrates DC motors driving a feed roller and a drive roller. The PID-based motor controllers minimize the error between the paper velocity produced by the roller and the target profile velocity produced by the Target Profile actor. The tracking error input allows one such roller to track the other to remove small differences in paper velocity.

The target profile is either a profile from 0 to maxPaperVelocity starting at time 0 and reaching the maximum value at time Interval seconds. The profile and its derivative are continuous.

SENSOR, ACTUATOR and NETWORK ACTORS STILL NEED TO BE ADDED

Code Generation to multiple target platforms

Simulation

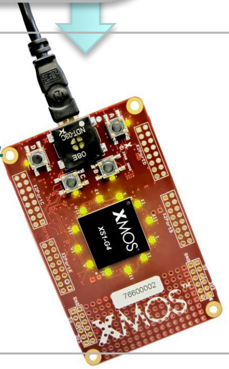


XMOS
Predictable timing
Multiple cores
No analog I/O
No FPU
No hardware clock

Renesas
PHY chip for accurate timestamping of inputs,
Analog I/O

Same I/O behavior w.r.t. value and timing

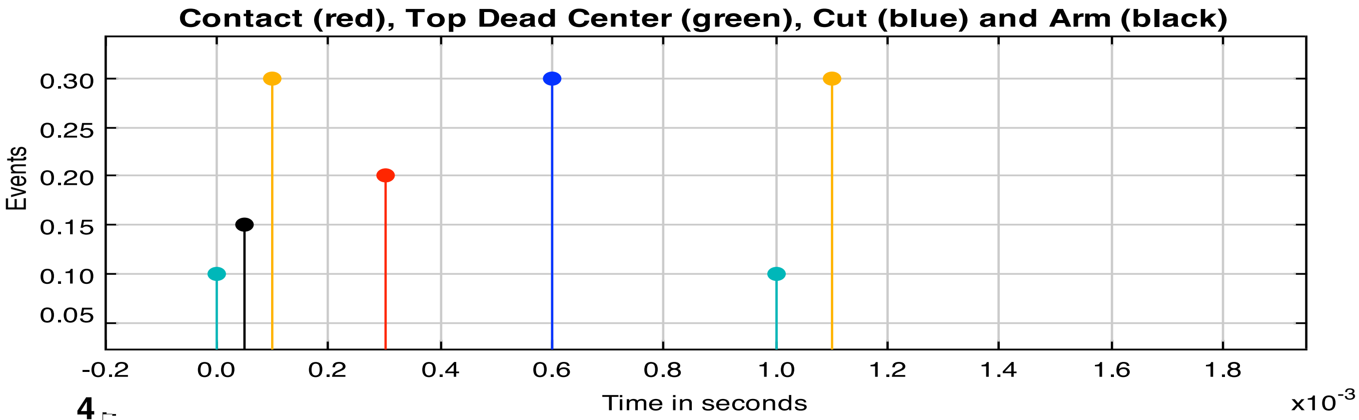
e.g.: XMOS development board with 4 XCores.



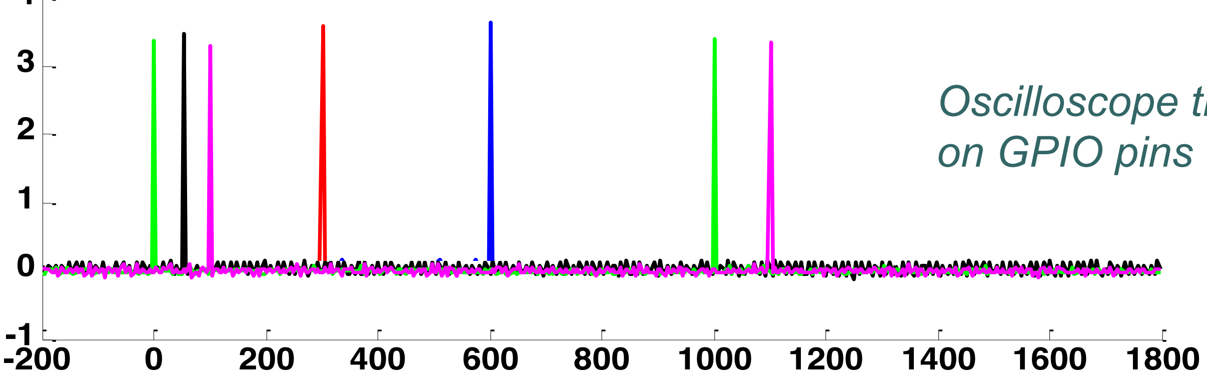
e.g.: Renesas 7216 Demonstration Kit

Renesas vs. XMOS: Measured I/O timing

Simulation

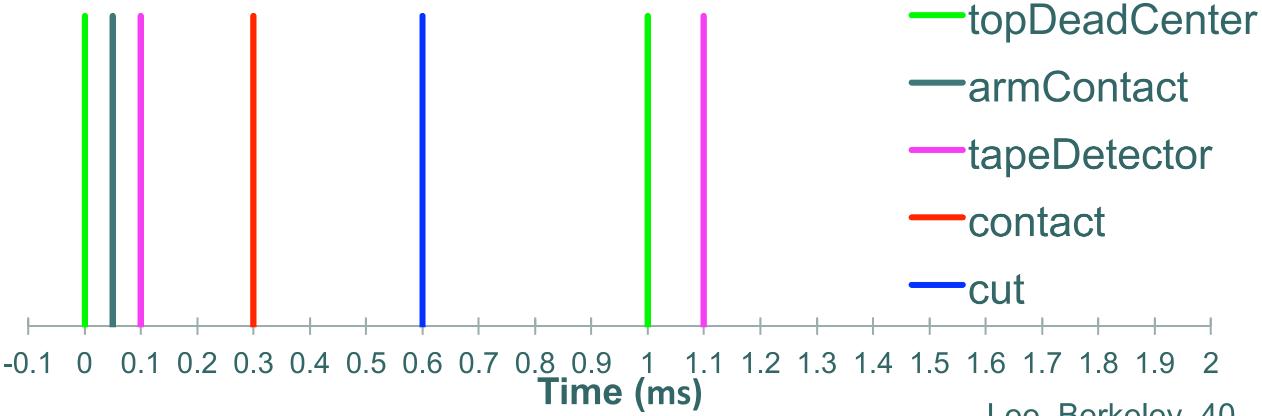


Renesas



Oscilloscope traces on GPIO pins

XMOS

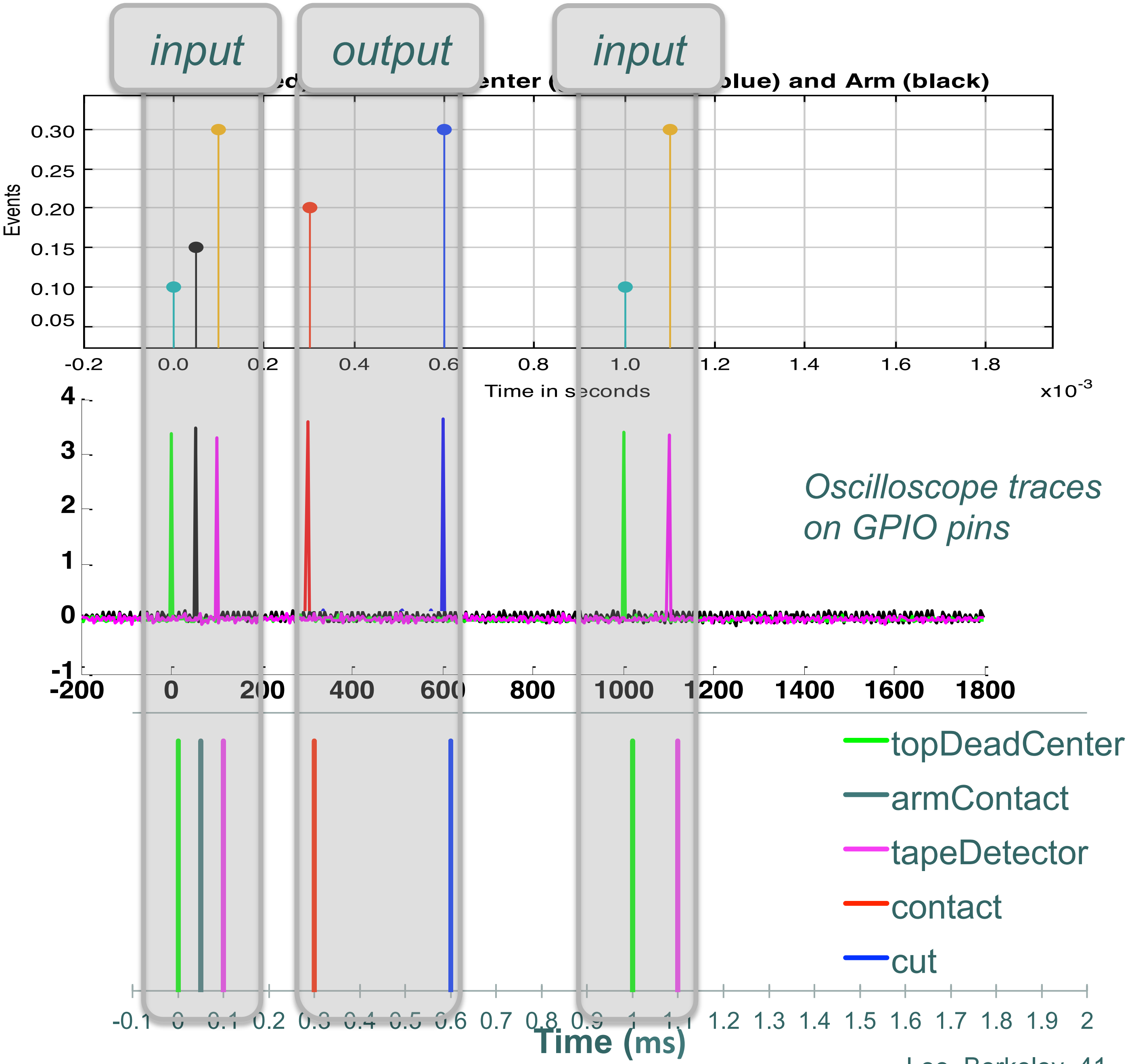


Renesas vs. XMOS: I/O timing

Simulation

Renesas

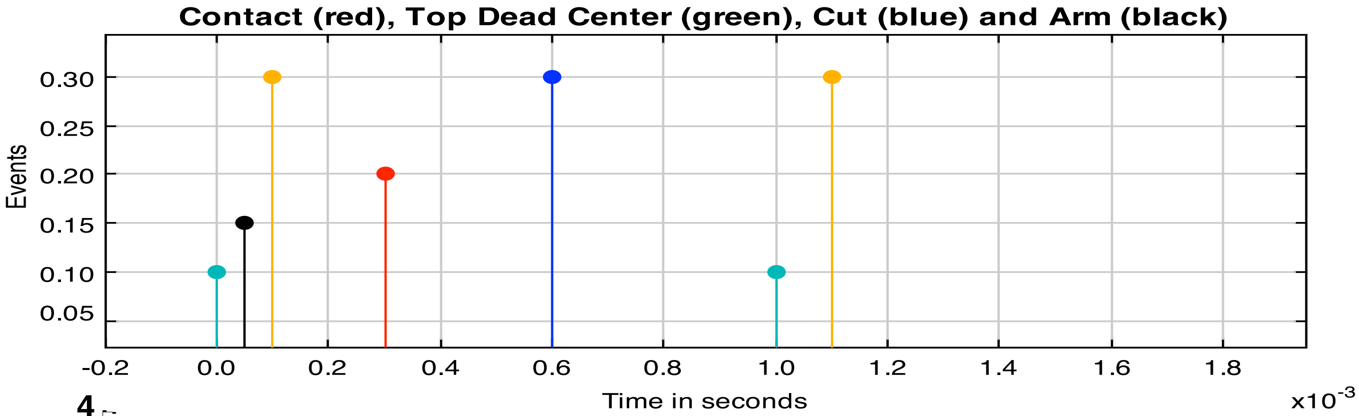
XMOS



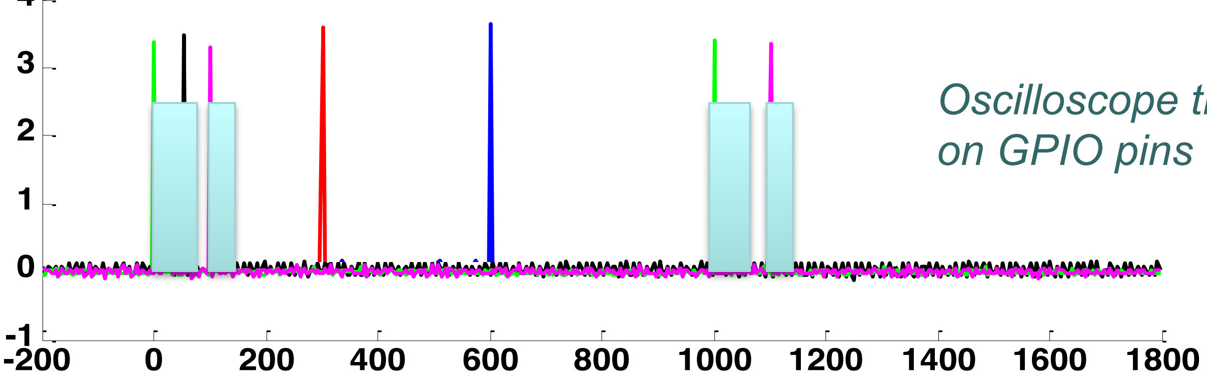
Oscilloscope traces on GPIO pins

Renesas vs. XMOS: Busy vs. Idle Time

Simulation

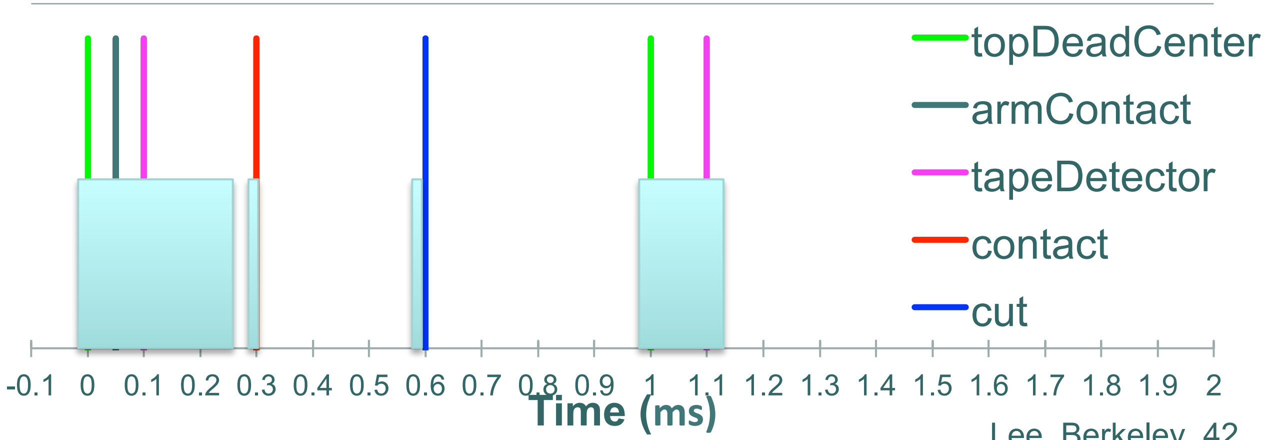


Renesas



Oscilloscope traces on GPIO pins

XMOS



http://chess.eecs.berkeley.edu/ptides/

Ptides Publications

- Y. Zhao, J. Liu, E. A. Lee, "**A Programming Model for Time-Synchronized Distributed Real-Time Systems**," RTAS 2007.
- T. H. Feng and E. A. Lee, "**Real-Time Distributed Discrete-Event Execution with Fault Tolerance**," RTAS 2008.
- P. Derler, E. A. Lee, and S. Matic, "**Simulation and implementation of the ptides programming model**," DS-RT 2008.
- J. Zou, S. Matic, E. A. Lee, T. H. Feng, and P. Derler, "**Execution strategies for Ptides, a programming model for distributed embedded systems**," RTAS 2009.
- J. Zou, J. Auerbach, D. F. Bacon, E. A. Lee, "**PTIDES on Flexible Task Graph: Real-Time Embedded System Building from Theory to Practice**," LCTES 2009.
- J. C. Eidson, E. A. Lee, S. Matic, S. A. Seshia and J. Zou, "**Time-centric Models For Designing Embedded Cyber-physical Systems**," ACES-MB 2010.
- J. C. Eidson, E. A. Lee, S. Matic, S. A. Seshia, and J. Zou, **Distributed Real-Time Software for Cyber-Physical Systems**, To appear in *Proceedings of the IEEE* special issue on CPS, December, 2011.

Conclusions

Overview References:

- *Lee.* ***Computing needs time****. CACM, 52(5):70–79, 2009*
- *Eidson et. al, Distributed Real-Time Software for Cyber-Physical Systems, Proc. of the IEEE January, 2012.*

Today, timing behavior is a property only of *realizations* of software systems.

Tomorrow, timing behavior will be a semantic property of *programs* and *models*.

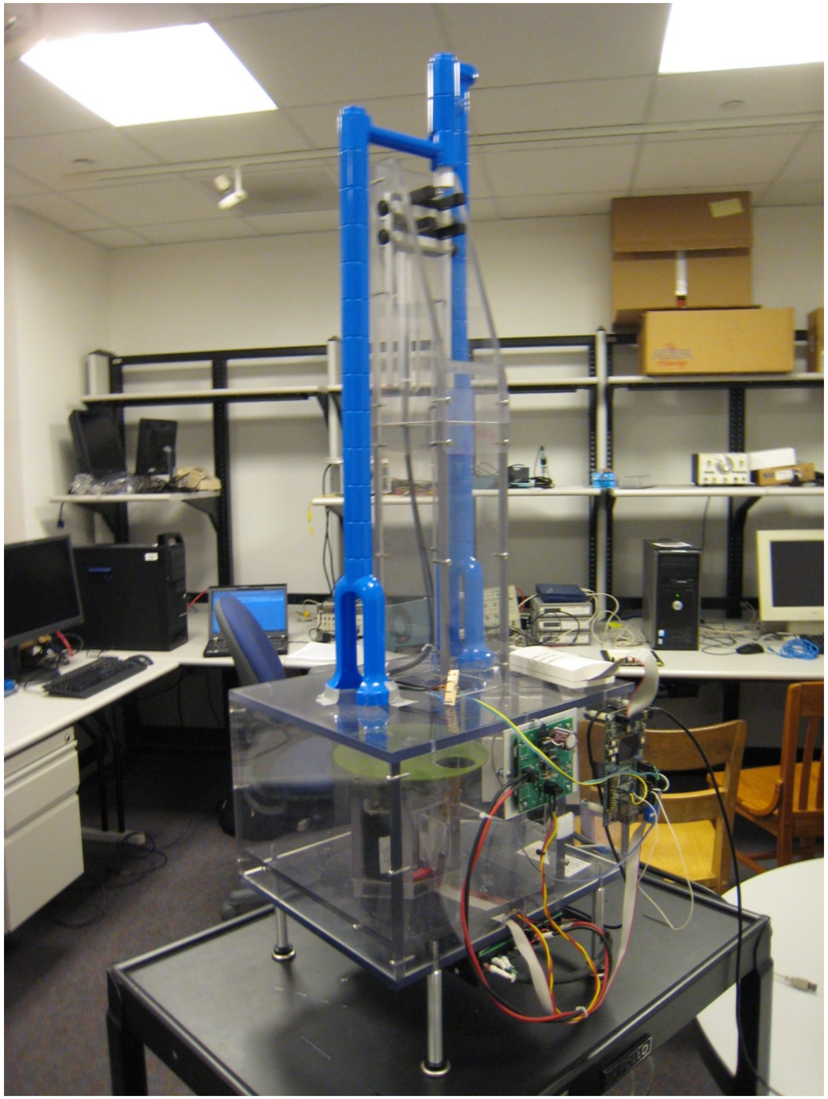
Raffaello Sanzio da Urbino – The Athens School

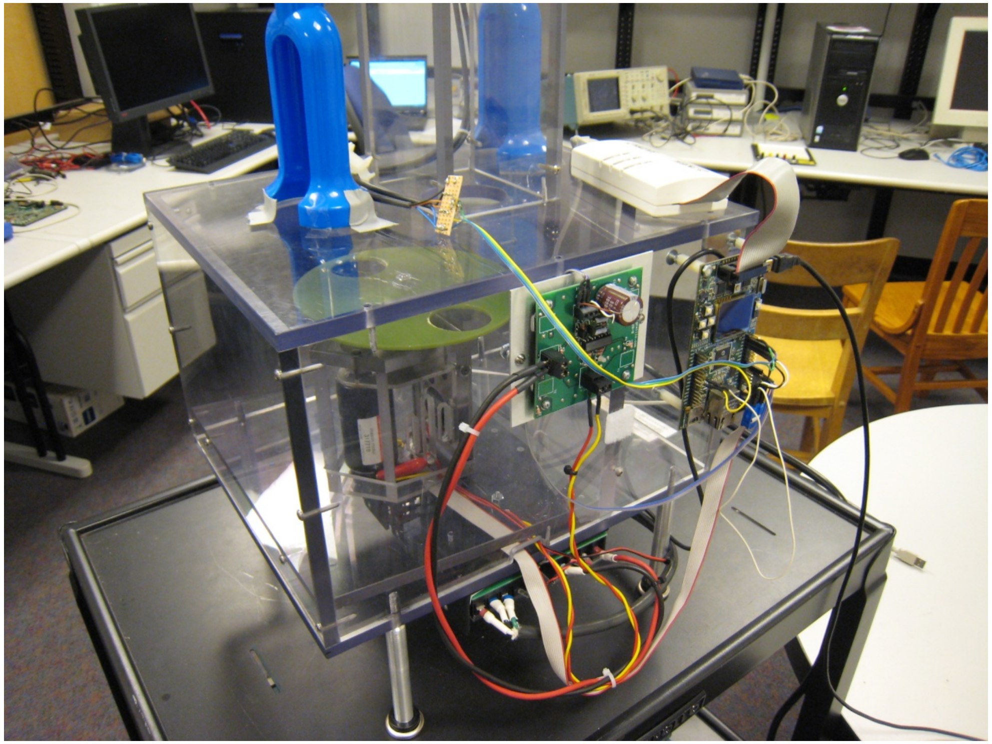
A Test Case for PtidyOS

This device, designed by Jeff Jensen, mixes periodic, quasi-periodic, and sporadic real-time events.

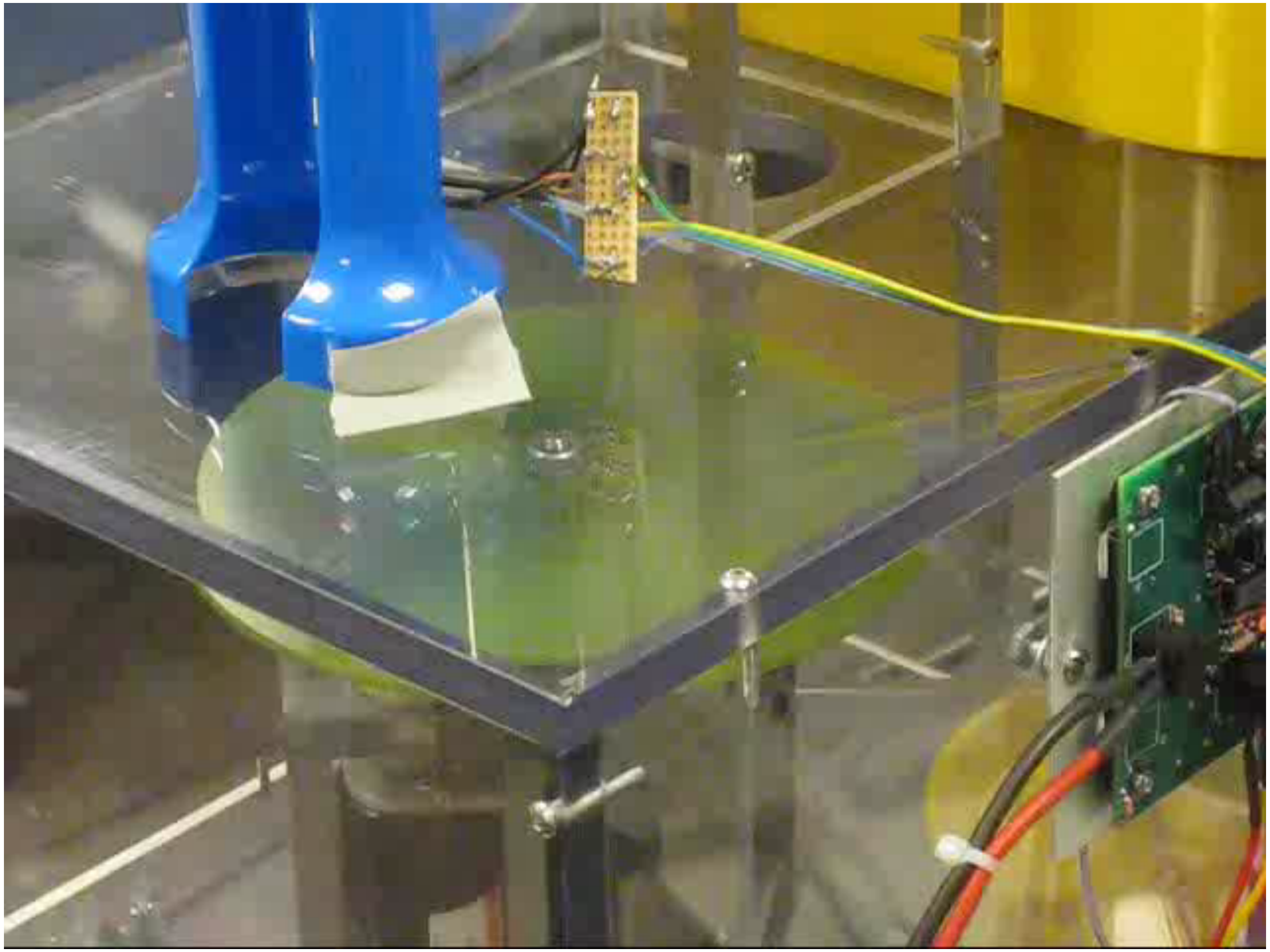
Tunneling Ball Device

- *sense ball*
- *track disk*
- *adjust trajectory*

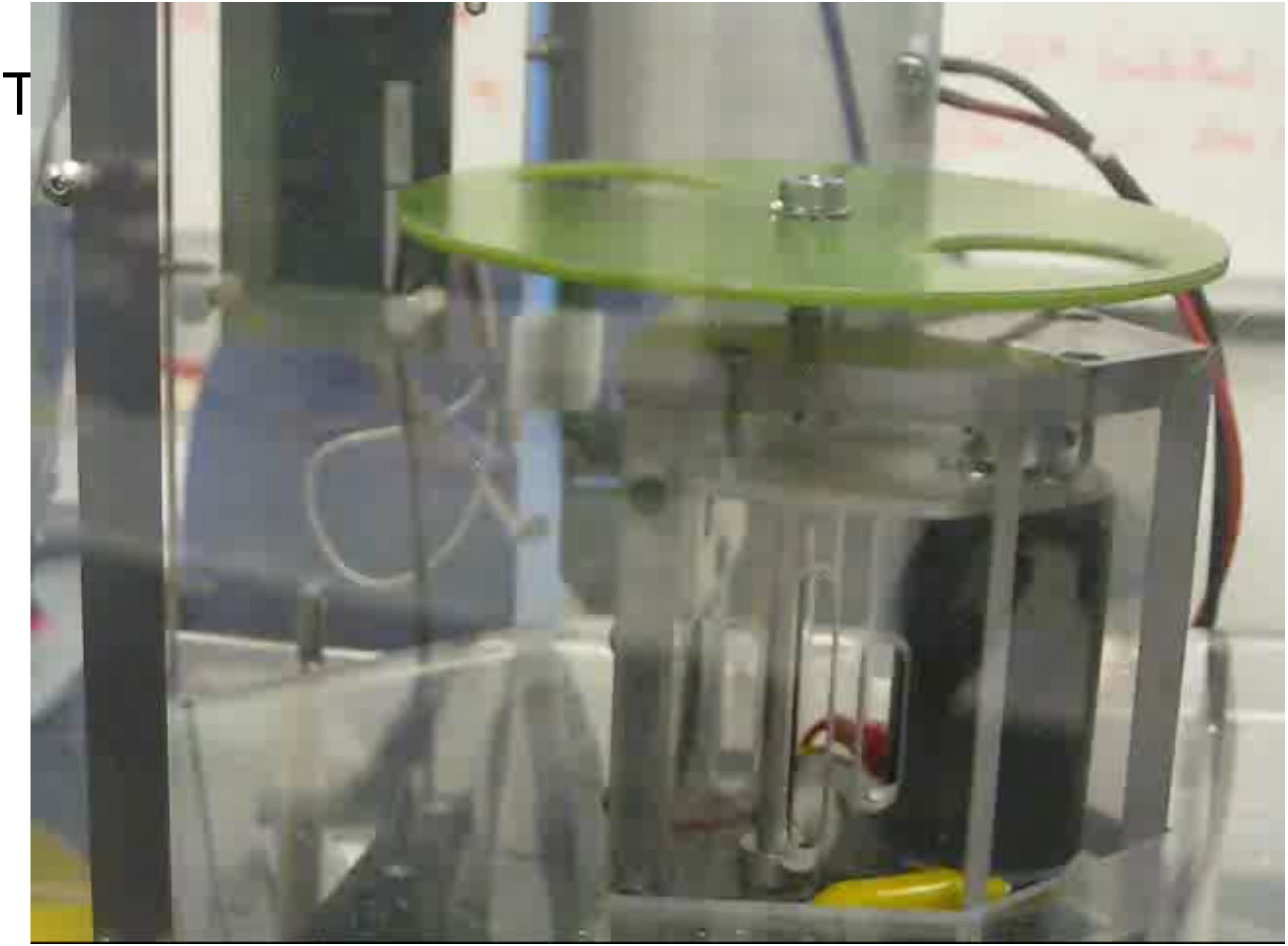




Tunneling Ball Device in Action

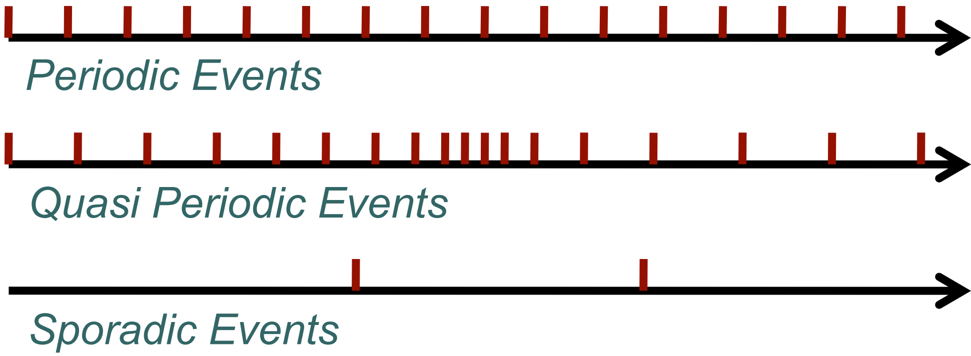


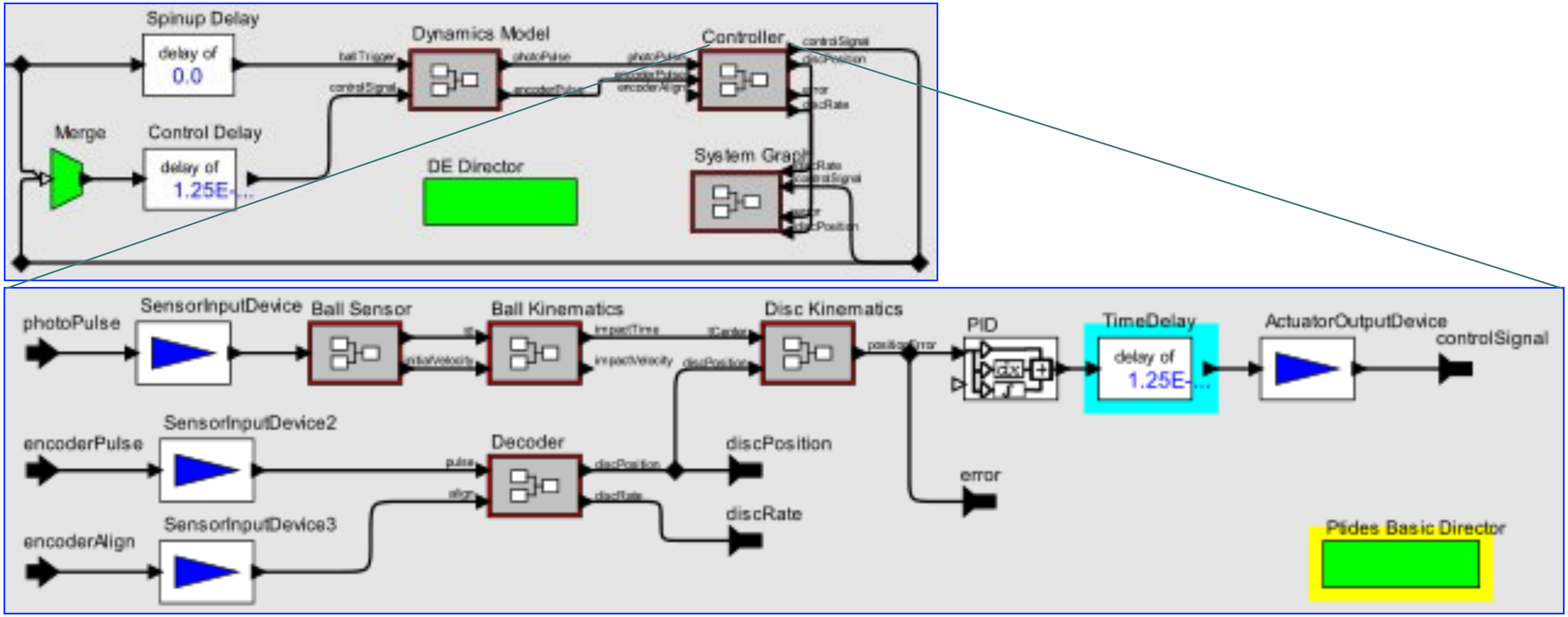
T



Tunneling Ball Device

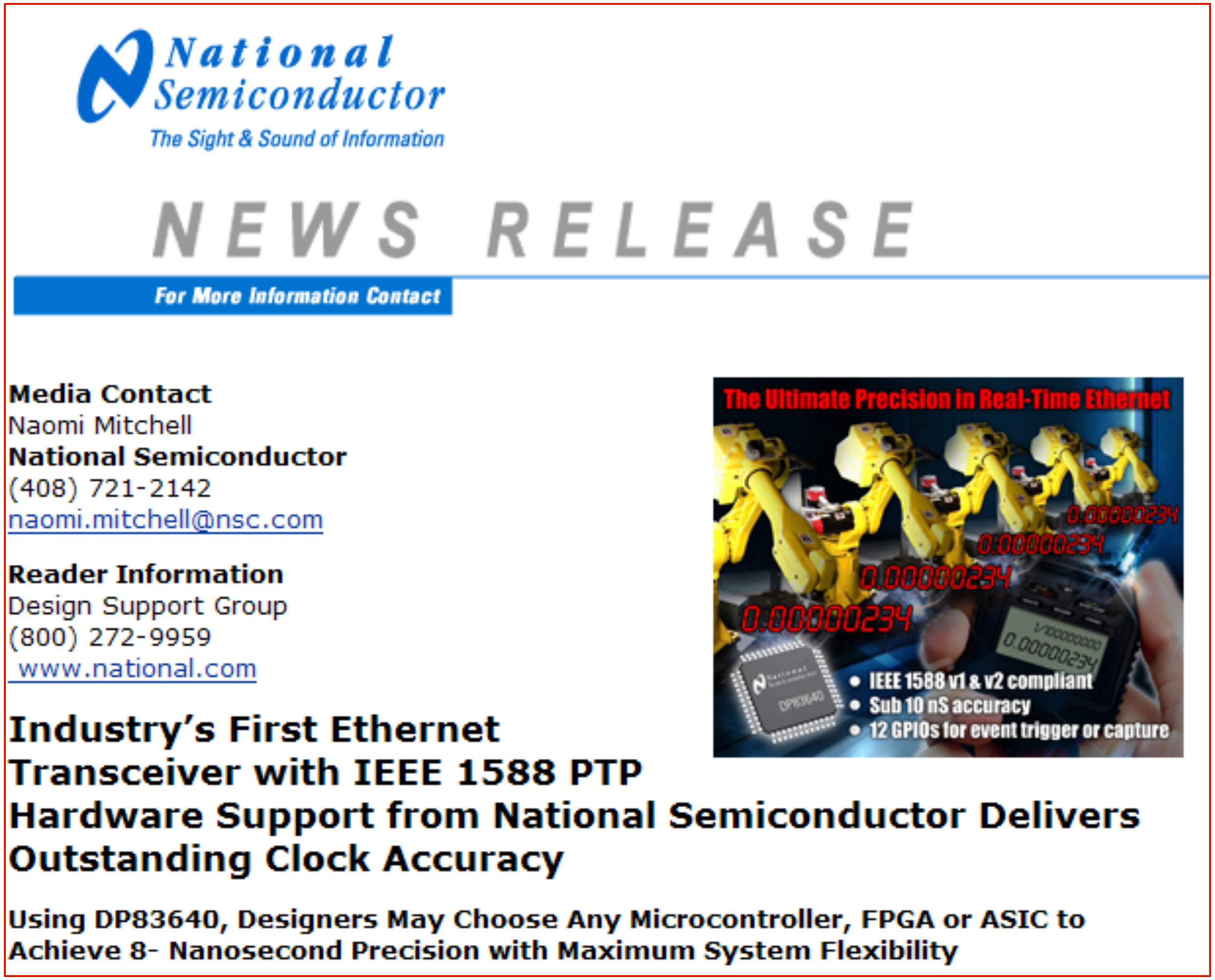
Mixed event sequences





Distributed PTIDES Relies on Network Time Synchronization with Bounded Error

Press Release October 1, 2007



National Semiconductor
The Sight & Sound of Information

NEWS RELEASE

For More Information Contact

Media Contact
Naomi Mitchell
National Semiconductor
(408) 721-2142
naomi.mitchell@nsc.com

Reader Information
Design Support Group
(800) 272-9959
www.national.com

Industry's First Ethernet Transceiver with IEEE 1588 PTP Hardware Support from National Semiconductor Delivers Outstanding Clock Accuracy

Using DP83640, Designers May Choose Any Microcontroller, FPGA or ASIC to Achieve 8- Nanosecond Precision with Maximum System Flexibility

The Ultimate Precision in Real-Time Ethernet
0.00000234
0.00000234
0.00000234
0.00000234
1/100000000
0.00000234
DP83640
IEEE 1588 v1 & v2 compliant
Sub 10 nS accuracy
12 GPIOs for event trigger or capture

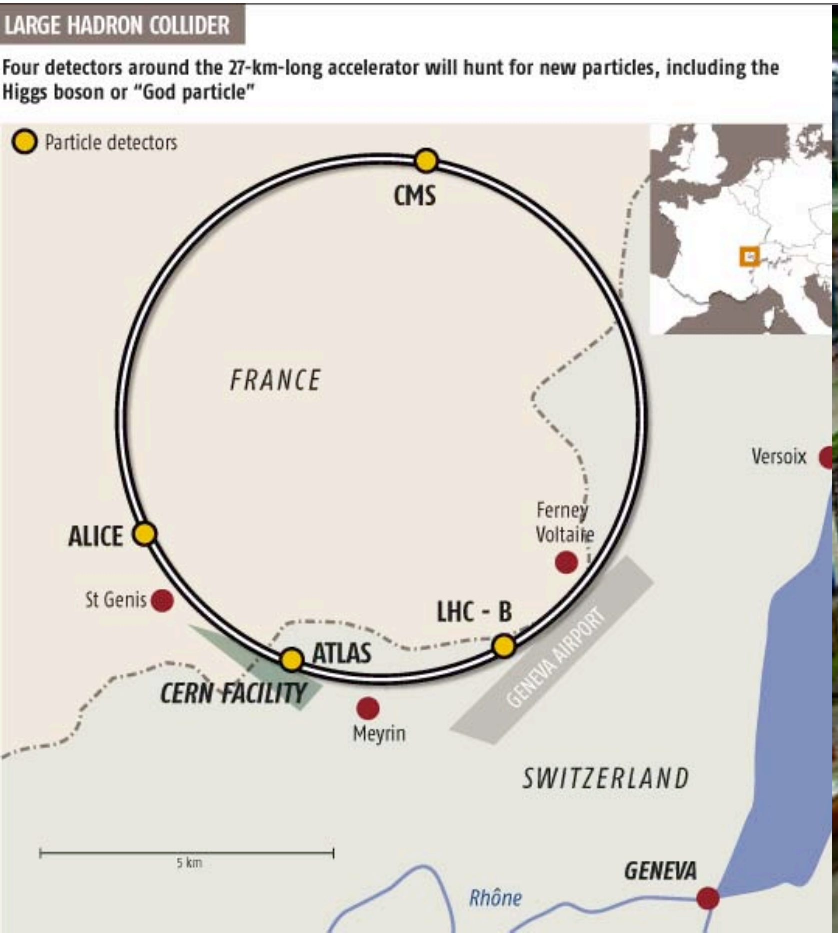
This may become routine!

With this PHY, clocks on a LAN agree on the current time of day to within 8ns, far more precise than older techniques like NTP.

A question we are addressing at Berkeley: How does this change how we develop distributed CPS software?

An Extreme Example: The Large Hadron Collider

The WhiteRabbit project at CERN is synchronizing the clocks of computers 10 km apart to within about 80 psec using a combination of IEEE 1588 PTP and synchronous ethernet.


LARGE HADRON COLLIDER
Four detectors around the 27-km-long accelerator will hunt for new particles, including the Higgs boson or "God particle"
Particle detectors
CMS
FRANCE
Versoix
Ferney Voltaire
ALICE
St Genis
LHC - B
ATLAS
CERN FACILITY
GENEVA AIRPORT
Meyrin
SWITZERLAND
5 km
GENEVA
Rhône