The Problem we are addressing

For cyber-physical systems, programs do not adequately specify behavior.

When interactions with physical processes are central, the temporal dynamics of software is a critical part of its behavior.

Today, temporal dynamics emerges from an implementation of the system, rather than being part of the model and/or program(s) specifying the system.

As a consequence, systems are brittle and non-portable.
Our Goal: Determinate timing at sensors and actuators

Platform independent model of functional and timing behavior

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

Our Goal: Determinate timing at sensors and actuators

Success on a test case: Flying Paster

Timing given by simulation of the model

Timing measured by oscilloscope on a Renesas board.

Timing measured by software instrumentation on an XMOS board.
Structure of a Cyber-Physical System

Problems that complicate analysis of system behavior:

- Sensors may be locked out for an indeterminate amount of time.
- Messages from different sources interleave nondeterministically.
- 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.
- Platforms’ measurements of time differ.
- A fault in a remote component may disrupt a critical local activity.
- A fault in a remote component may go undetected for a long time.
- Platforms’ measurements of time differ.
- Messages from different sources interleave nondeterministically.

Our Programming Model

PTIDES:
Programming Temporally Integrated Distributed Embedded Systems

Based on a determinate discrete-event (DE) model of computation (MoC), originally developed for simulation.
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.
Ptides: Third step:  
Bind time stamps to real time at sensors and actuators

- Input time stamps are $\geq$ real time
- Output time stamps are $\leq$ real time
- Messages are processed in time-stamp order.

- Clock synchronization gives global meaning to time stamps

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

Ptides: Fourth step:  
Specify latencies in the model

- 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 network delay $d$
Assume bounded clock error $e$
An earliest event with time stamp $t$ here can be safely merged when real time exceeds $t + s + d + e - d^2$

Application specification of latency $d_2$

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 $t - c_3 - d_2$

Assume bounded computation time $c_1$
Assume bounded computation time $c_2$
Assume bounded computation time $c_3$
PtidyOS: A lightweight microkernel supporting Ptides semantics

PtidyOS runs on
- Arm (Luminary Micro)
- Renesas
- XMOS

Occupies about 16 kbytes of memory.

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!

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

Workflow Structure

Ptolemy II Ptides domain

Ptolemy II Discrete-event, Continuous, and Wireless domains

Mixed Simulator

Plant Model

Network Model

Ptolermy II Ptides domain

Analysis

Schedulability Analysis

Causality Analysis

Program Analysis

Code

Code Generator

Software Component Library

HW Platform

HW in the Loop Simulator

e.g., Luminary Micro 8962

IEEE 1588 Network time protocol

A Typical Cyber-Physical System
Printing Press

- Application aspects
  - local (control)
  - distributed (coordination)
  - global (modes)
- Ethernet network
  - Synchronous, Time-Triggered
  - IEEE 1588 time-sync protocol
- High-speed, high precision
  - Speed: 1 inch/ms (~100km/hr)
  - Precision: 0.01 inch
    -> Time accuracy: 10us

Example – Flying Paster

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

Flying Paster

Printing Press – Model in Ptolemy II

See talk Thursday by Patricia Derler at workshop on Model-Based Design with a Focus on Extra-Functional Properties
See talk Thursday by Patricia Derler at workshop on Model-Based Design with a Focus on Extra-Functional Properties.
Determinate timing at sensors and actuators

Platform independent model of functional and timing behavior

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.

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

e.g.: Renesas 7216 Demonstration Kit
Lee, et al. Berkeley 22
Renesas vs. XMOS: Measured I/O timing

Simulation

Renesas

XMOS


Contact (red), Top Dead Center (green), Cut (blue) and Arm (black)

-0.2  0.0  0.2  0.4  0.6  0.8  1.0  1.2  1.4  1.6  1.8
Time in seconds

-0.1  0.0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  1.0
Time (ms)


Input, Output, Input

Simulation

Renesas

XMOS


Contact (red), Top Dead Center (green), Cut (blue) and Arm (black)

-0.2  0.0  0.2  0.4  0.6  0.8  1.0  1.2  1.4  1.6  1.8
Time in seconds

-0.1  0.0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  1.0
Time (ms)
Renesas vs. XMOS: Busy vs. Idle Time

Simulation

Renesas

XMOS

Ptides Publications


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

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

Overview References:
• Lee, Computing needs time. CACM, 52(5):70–79, 2009