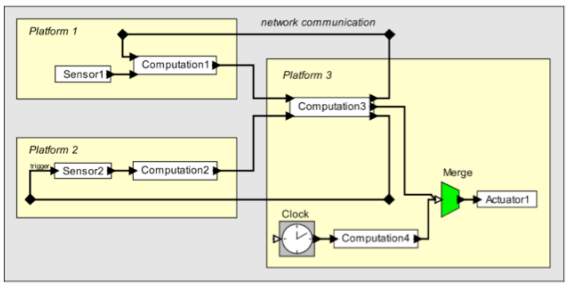
Code Generation For PTIDES Models

Jia Zou
Jeff C. Jensen
Isaac Liu

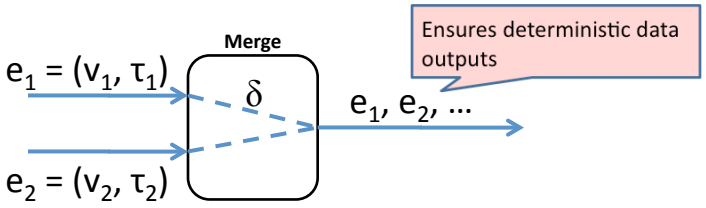
EE290N Project Presentation
5/15/09

PTIDES Programming Model

- Programming Temporally Integrated Distributed Embedded Systems
  - Time is part of the abstraction
  - Based on Discrete-Event model of computation
    - Actors process events in time-stamp order
    - **Deterministic** under simple causality conditions

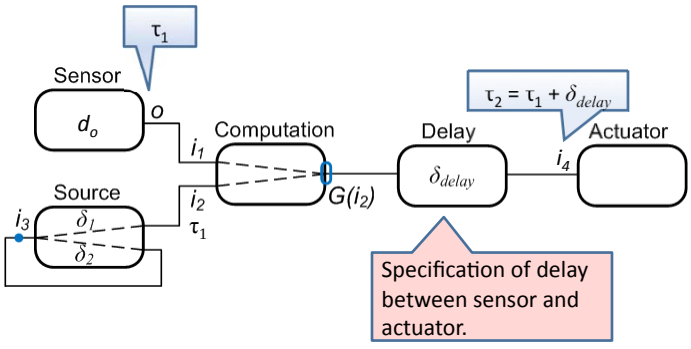


Deterministic Data Outputs

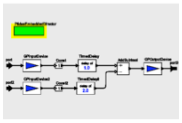


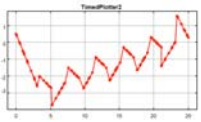
Deterministic Time Outputs

- At sensors and actuators
  - Relate **model** time (τ) to **physical** time (t)



PTIDES Workflow





Design

Simulation

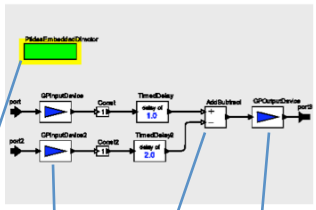
Code Generation

PtidyOS Runtime

Schedulability Analysis

Program Analysis

PtidyOS



Scheduler.lib

Sensor.lib

Computation.lib

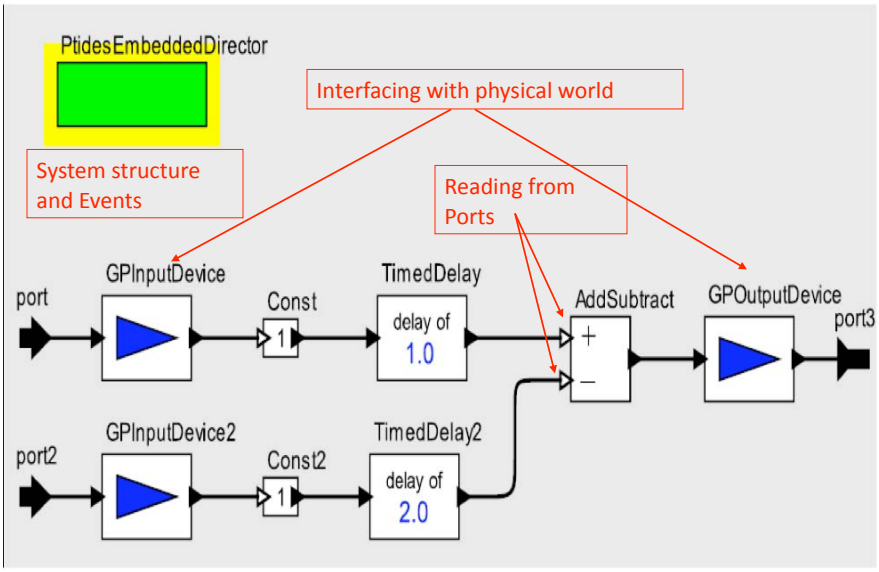
. . . . . .

Main() {
...
}

PtidyOS.exe

-- All event processing is implemented within interrupt service routines.
-- All interrupts are reentrant, interrupts do not have priority only events do.
-- No dynamic memory allocation.
-- Combines PTIDES semantics with traditional scheduling algorithms (for example Earliest-Deadline-First(EDF)).

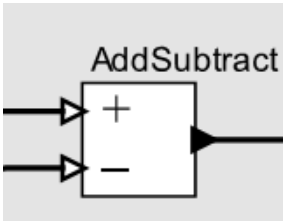
Example PTIDES Model



System Structure and Events

- Model Structure
  - Partial evaluation:
    - Static model graph - Use function calls instead of generating actual actor structures
    - Decreases code size and execution time
- Event Structure
  - Events store the data tokens as well as the destination fire function

PTIDES vs Data Flow



Data Flow:
Fires the actor when the firing rule matches (both ports receive an event).

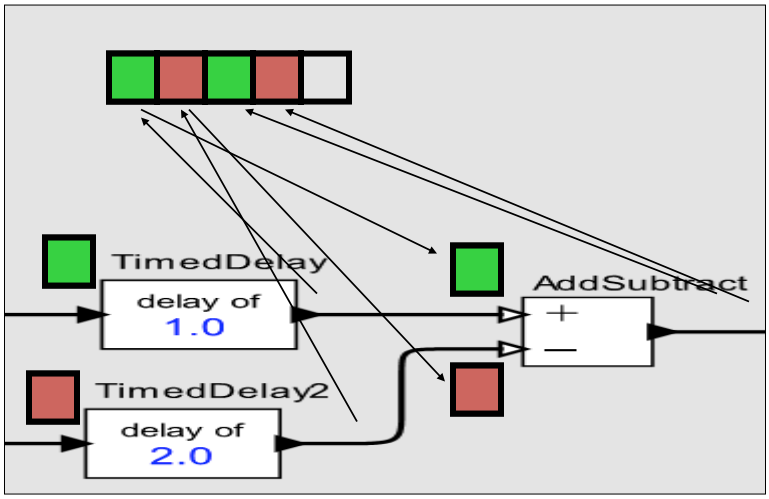
PTIDES:
Fires the actor whenever an input event is received, regardless of which port.

For Code Generation:

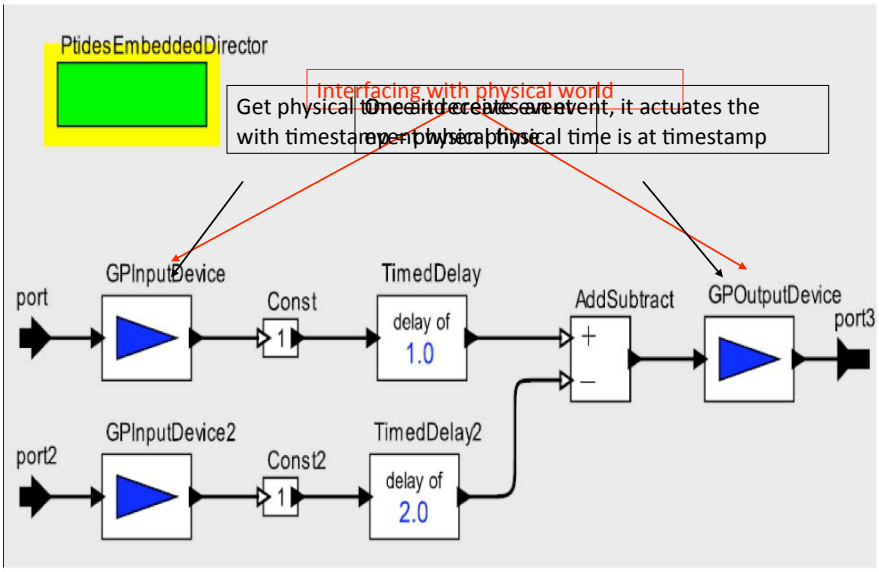
Data Flow:
Fire function assumes data from input ports, so no need to check.

PTIDES:
Fire function needs to check which input port has an event, since it could be from different timestamps.

Reading from Inputs



Interfacing with real world



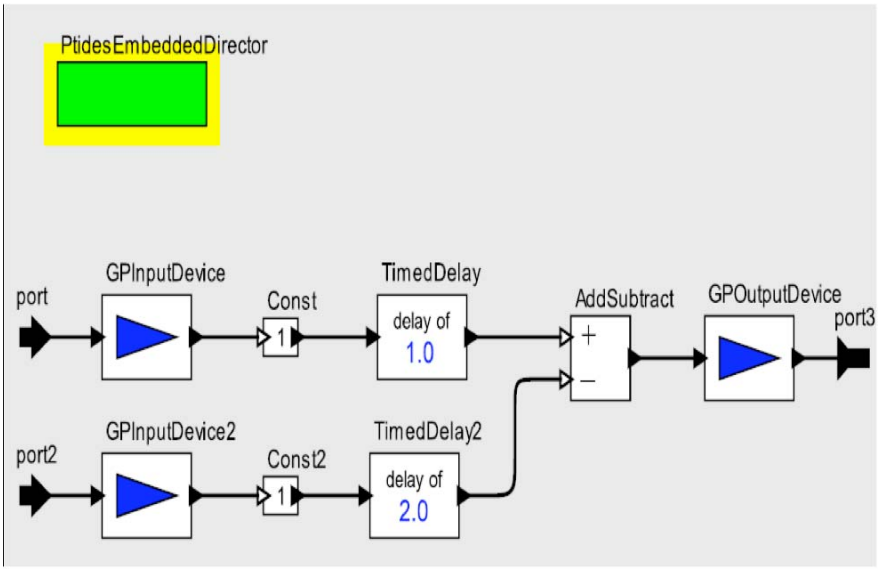
Sensor generated code

- It is an interrupt service routine
  1. Gets Physical Time
  2. Read Value
  3. Generate event with timestamp = physical time
  4. Add event to event queue

Actuator generated code

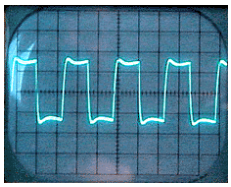
- A function to setup a timer interrupt
  1. Read event timestamp
  2. Get difference between physical and event timestamp
  3. Setup timer with the difference
- And a fire function which handles the timer interrupt
  - Actually do the actuation

Example PTIDES Model

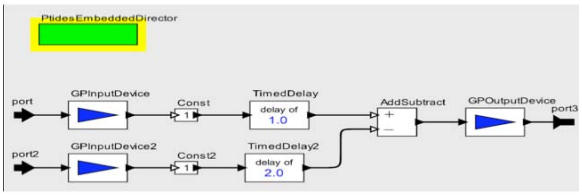


Live Demo: Time-Triggered Music

- Periodic sensor input
- Expect periodic output
- Correctly played music verifies
  - Periodic actuator output
  - Correct event order (timestamps)



Input: 1-10Hz square wave



Output: Legend of Zelda