C CODE GENERATION FROM THE Giotto MODEL OF COMPUTATION TO THE PRET ARCHITECTURE

Shanna-Shaye Forbes
Ben Lickly
Man-Kit Leung
EE290N Course Project, Spring 2009
May 15, 2009

Overview

- Why is real-time important?
- PRET/Giotto Overview
- Code Generation Infrastructure
- Approach
- Limitations/Future Work
- Elevator Controller Example
DEMAND PACEMAKER

Motivation

- Timing as important as functionality
- Traditional languages and architectures do not support timing on the same level as functionality.
- PRET and Giotto together support timing specifications and guarantees.
Precision Timed Architecture

- Architectural features support timing predictability
- Instruction set architecture includes instructions to manage timing
- Exception mechanism for missed deadlines

Giotto

- Giotto is a time-triggered language for embedded programming.
- Ideal for hard real-time applications with time-periodic and multimodal behavior.
- It specifies time-triggered sensor readings, task invocations, actuator updates, and mode switches independent of any implementation platform.
Giotto cont.

- Unit delay in task communication
- Tasks communicate through ports
- Drivers move input values from ports to inputs at the beginning of a task’s execution
- Drivers move input values from outputs to ports at the end of a task’s execution

Giotto in Ptolemy II

- Each actor is a separate task.
- Period/frequency specified with attributes.
- Models can be composed hierarchically, with modes represented through modal models.
Extending the Ptolemy II Code Generation Framework

Ptolemy II Code Generation

- Ptolemy II has an adapter based extensible code generation framework.
- We have continued this extension to support Giotto and PRET.
Mapping from Giotto to PRET

- Each Giotto task is mapped to a separate hardware thread.
- Threads communicate through shared memory.
- Input and output drivers are responsible for moving data.
- PRET’s timing instructions detect errors when deadlines are missed.

Limitations/Future Work

- Giotto does not specify error behavior.
  - We assume errors are fatal.

- This approach is tied to a PRET architecture.
  - Fixed number of hardware threads in PRET.
Elevator Controller Example
Generated Code Example

```c
int main(int argc, char *argv[]) {
    initialize();
    jmp_buf _deadline, trying, jmpbuf...
    register jmpbuf(0, &deadline, trying, jmpbuf...);
    if (setjmp(_deadline, trying, jmpbuf...) == 0) {
        gout("Timing failure\n");
        END_SIMULATION;
    }
    while (true) {
        #ifdef THREAD2
        #define twoFloorElevatorController.CompositeActor.OUT.PUT.DRIVER.WCET
        #warning "...
        #endif
        DEADBRANCH(125000 - twoFloorElevatorController.CompositeActor.OUT.PUT.DRIVER.WCET);
        twoFloorElevatorController.CompositeActor.driver_in();
        DEADBRANCH(twoFloorElevatorController.CompositeActor.OUT.PUT.DRIVER.WCET);
        twoFloorElevatorController.CompositeActor.driver_out();
        #endif
        #ifdef THREAD1
        #define twoFloorElevatorController.FloorController.OUT.PUT.DRIVER.WCET
        #warning "...
        #endif
        DEADBRANCH(50000 - twoFloorElevatorController.FloorController.OUT.PUT.DRIVER.WCET);
        twoFloorElevatorController.FloorController.driver_in();
        DEADBRANCH(twoFloorElevatorController.FloorController.OUT.PUT.DRIVER.WCET);
        twoFloorElevatorController.FloorController.driver_out();
        #endif
    }
}
```
Summary

- Ptolemy II allows simulation of timed models of computation.

- Generated code uses timing instructions of the target architecture to preserve these timing semantics.

- Provided a possible programming model for a new timed architecture.

Questions?

Comments?

Suggestions?