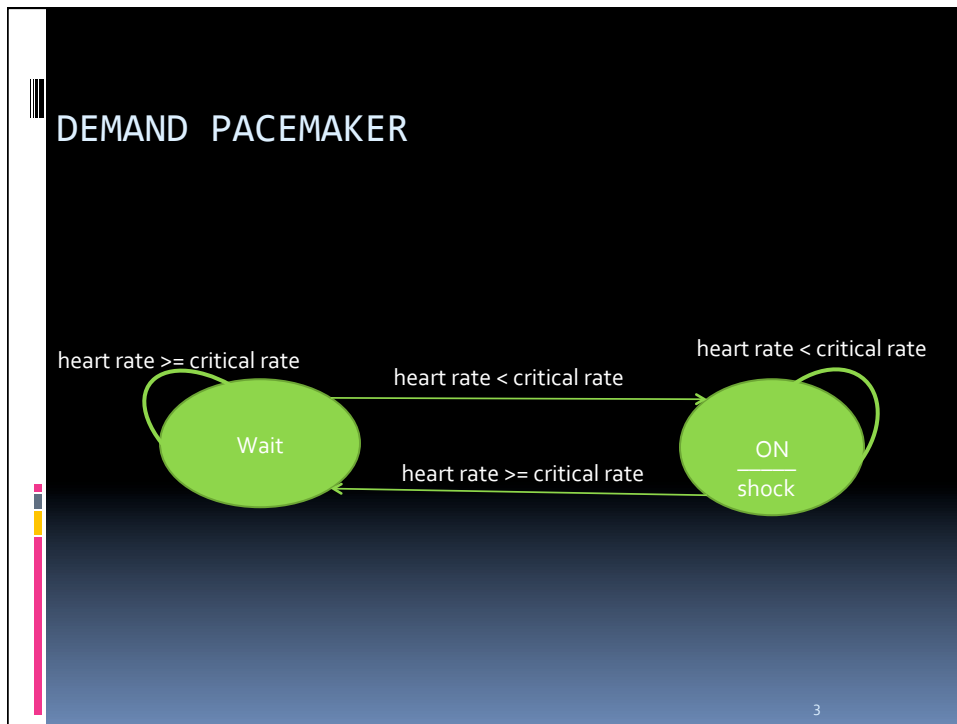


C CODE GENERATION FROM THE GIOTTO MODEL OF COMPUTATION TO THE PRET ARCHITECTURE

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



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.

4

Precision Timed Architecture

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

5

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.


6

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 CodeGenration Framework

9



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.

11

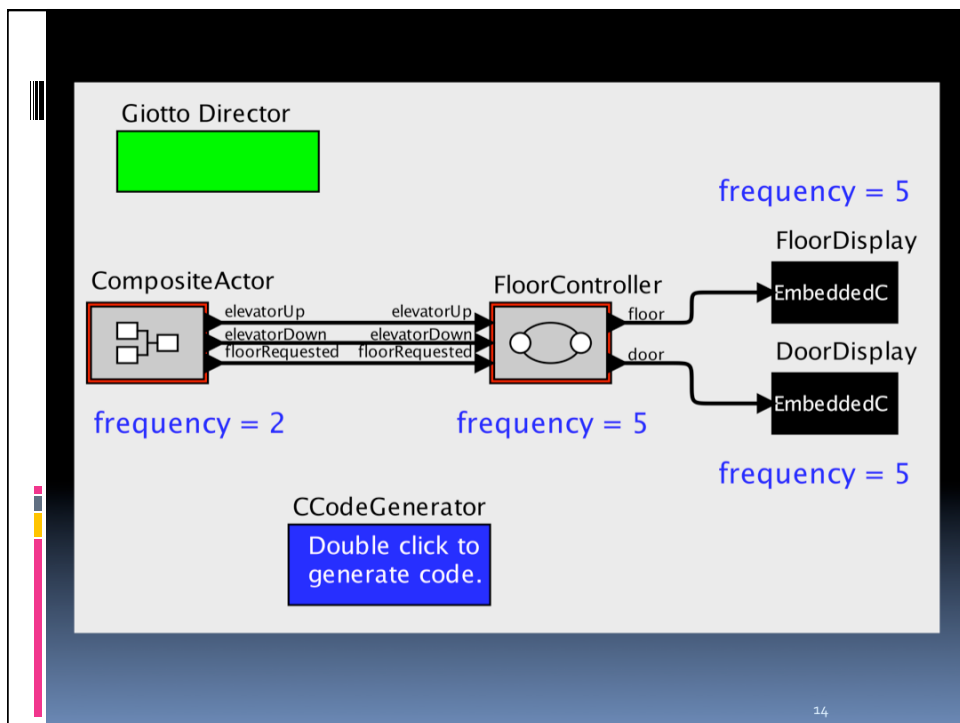
Limitations/Future Work

- 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.

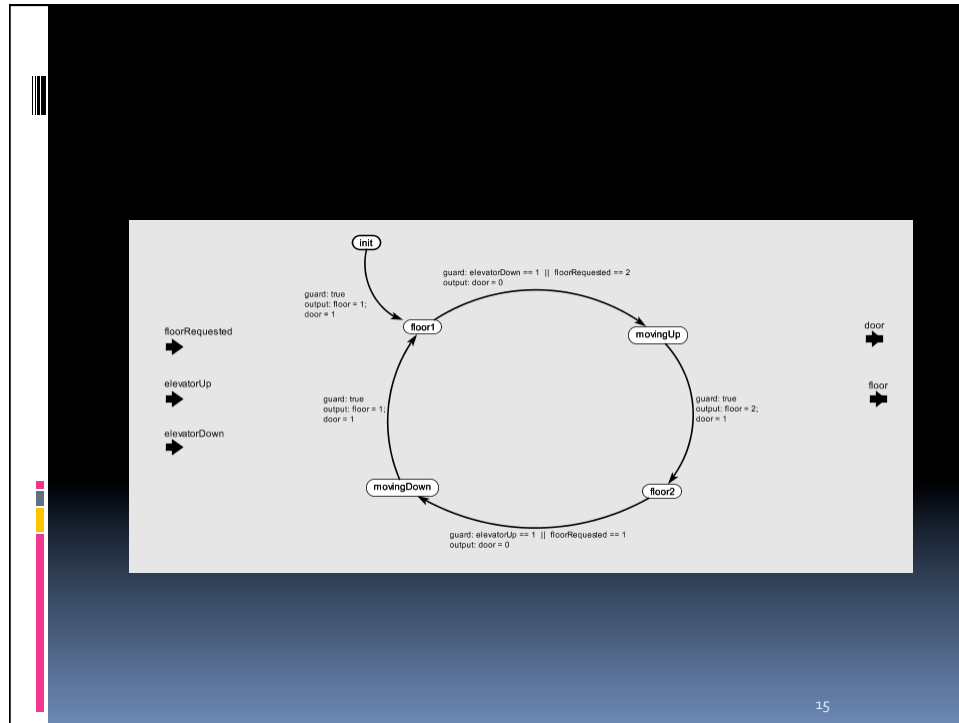
12

Elevator Controller Example

13



14



15

Generated Code Example

```

int main(int argc, char *argv[]) {
    initialize();
    jmp_buf __deadline_trying_jmpbuf__;
    register_jmpbuf(0, &__deadline_trying_jmpbuf__);
    if (setjmp(__deadline_trying_jmpbuf__) != 0) {
        puts("Timing-failure!\n");
        END_SIMULATION;
    }
    while (true) {
#ifdef THREAD_0
#ifdef twoFloorElevatorController_CompositeActor_OUTPUT_DRIVER_WCET
#warning "... "
#define twoFloorElevatorController_CompositeActor_OUTPUT_DRIVER_WCET 1000
#endif
        DEADBRANCH0(125000 - twoFloorElevatorController_CompositeActor_OUTPUT_DRIVER_WCET);
        twoFloorElevatorController_CompositeActor_driver_in();
        twoFloorElevatorController_CompositeActor();
        DEADBRANCH0(twoFloorElevatorController_CompositeActor_OUTPUT_DRIVER_WCET);
        twoFloorElevatorController_CompositeActor_driver_out();

#endif /* THREAD_0 */
#ifdef THREAD_1
#ifdef twoFloorElevatorController_FloorController_OUTPUT_DRIVER_WCET
#warning "... "
#define twoFloorElevatorController_FloorController_OUTPUT_DRIVER_WCET 1000
#endif
        DEADBRANCH0(50000 - twoFloorElevatorController_FloorController_OUTPUT_DRIVER_WCET);
        twoFloorElevatorController_FloorController_driver_in();
        twoFloorElevatorController_FloorController();
        DEADBRANCH0(twoFloorElevatorController_FloorController_OUTPUT_DRIVER_WCET);
        twoFloorElevatorController_FloorController_driver_out();

#endif /* THREAD_1 */
    }
}

```


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.

17

Questions?

Comments?

Suggestions?

18