ESD Electronic Systems Design

Synthesis Verification Testing Power Communication

GIOTTO

Dott. Luigi Di Guglielmo

Prof. Tiziano Villa

University of Verona

Dep. Computer Science

Italy

Outline

- Introduction
- GIOTTO concepts
- The language
- The semantics
- Example
- Conclusion

Introduction (I)

- GIOTTO is a time-triggered programming language that aims at implementing embedded control systems on distributed platforms
- It has been developed by the Embedded System Design Group at Berkeley, University of California (US)

Introduction (II)

- Embedded software development for control applications consists of two phases: first *modeling* then *implementation*
- Modeling consists of defining the logic of the control application
  - Done by control engineers
  - Supported by tools that offer limited code-generation facilities (e.g., Matlab)
- Efficient implementation takes into account the model, code and platform constraints
  - Done by software engineers
  - Generation of code for specific platforms

Introduction (III)

- Development issues can be distinguished in
  - Platform independent
    - Platform independent issues include application functionality and timing
  - Platform dependent
    - Platform dependent issues include scheduling, communication and physical performance

Introduction (IV)

- A key to automating the embedded software development is to define an interface between platform-independent and platform-dependent issues
- Such an interface, i.e., abstract programmer's model for embedded systems, enables decoupling software design from implementation

GIOTTO

- Giotto provides an abstract programmer's model that enables design decoupling from implementation
  - A Giotto *program* determines the functionality (i.e., input and output behavior) and the timings of the software
  - The Giotto compiler synthesizes the embedded software for a given platform
    - The synthesis problem is difficult for distributed platforms, thus, the compiler may fail

Giotto Program

- A Giotto program is defined by specifying a set of *modes* and *mode switches*
- Each mode specifies a set of concurrent tasks and the switches from the current mode to the others
  - At every instant the program execution is in one specific mode, e.g., M
  - Each task of M has a real-time frequency and is invoked at this frequency as long as the mode M remains unchanged
  - Each mode switch of M to a mode M' has a real-time frequency and a condition that is evaluated at this frequency. If the condition evaluates to true, then the new mode is M'
  - In the new mode, some tasks may be removed and others added

GIOTTO Semantics

- Giotto has a formal semantics that specifies the meaning of mode and mode switches, of inter-task communication and communication with the program environment
- In Giotto, the environment consists of sensors and actuators

GIOTTO LANGUAGE

Basic Components

- The main components used by Giotto for defining a *program* are:
  - Ports
  - Tasks
  - Drivers
  - Modes
  - Mode switches

Giotto's Language Primitives (I)

- A *port*
  - is a physical location in memory
  - It may either be associated with a sensor or actuator (i.e., environment communication), or be used for inter-task communication
- A *task*
  - is a periodic job characterized by input and output ports and an implementation, written in any programming language, with known WCET (worst-case execution time)
  - At each invocation, a task reads its inputs and computes new values for its output ports

Giotto's Language Primitives (II)

- A *driver* represents a connection
  - A driver computes a function on its source ports and passes the result to its destination ports
    - Actuator driver
      - The driver destination ports are actuators ports
    - Task driver
      - The driver destination ports are input ports of a task
  - A Giotto driver has a guard, i.e., a predicate on its source ports; if the guard does not evaluate to true, then the driver is not executed

Giotto's Language Primitives (III)

- A *mode* represents a functional unit
  - Set of concurrent tasks
    - Invocation frequency
    - Task drivers
  - Set of actuator updates
    - Invocation frequency
    - Actuator drivers
  - Set of mode switches

Giotto's Language Primitives (IV)

- A *mode switch* represent a jump between functional units
  - When a mode switch is enabled, it causes the program to switch instantaneously from one mode to another
  - It has an invocation frequency and a *mode driver*, whose guard is evaluated with the given frequency
    - When the guards evaluates to true the switch is enabled and the driver is executed
  - Guards of mode drivers are disjoint
    - It guarantees determinism

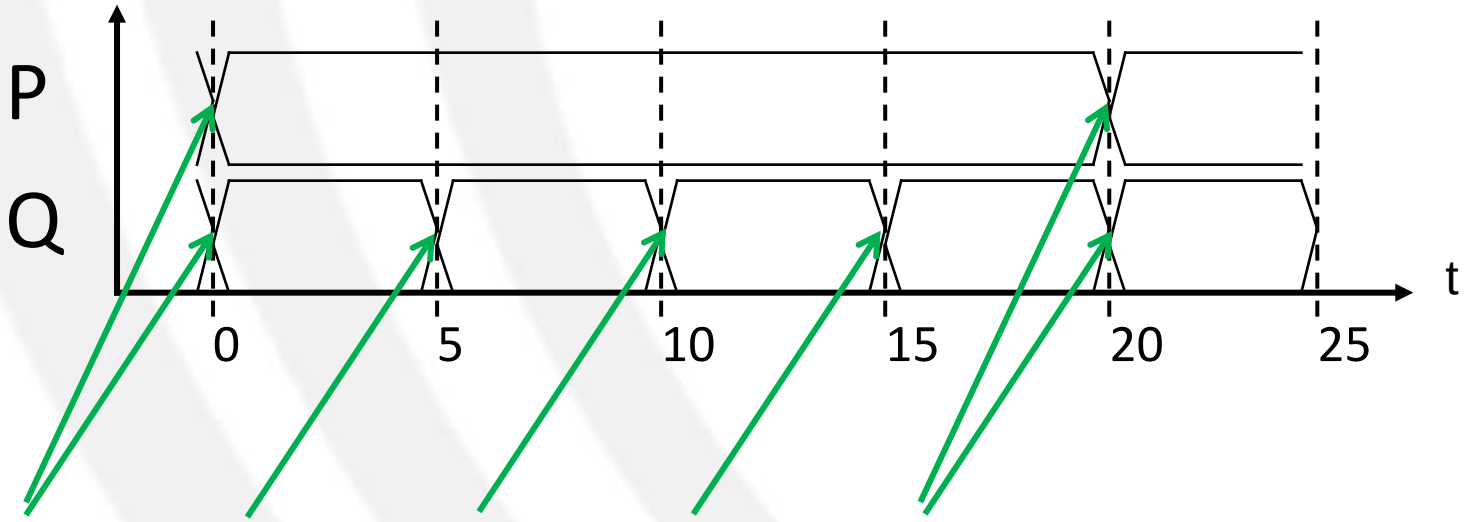
Giotto's Language Primitives (V)

- A *mode driver* connects modes
  - *Source ports of the mode driver are task output ports of the current mode, and the destination ports are the task output ports of the next mode*

Language Semantics: Task (I)

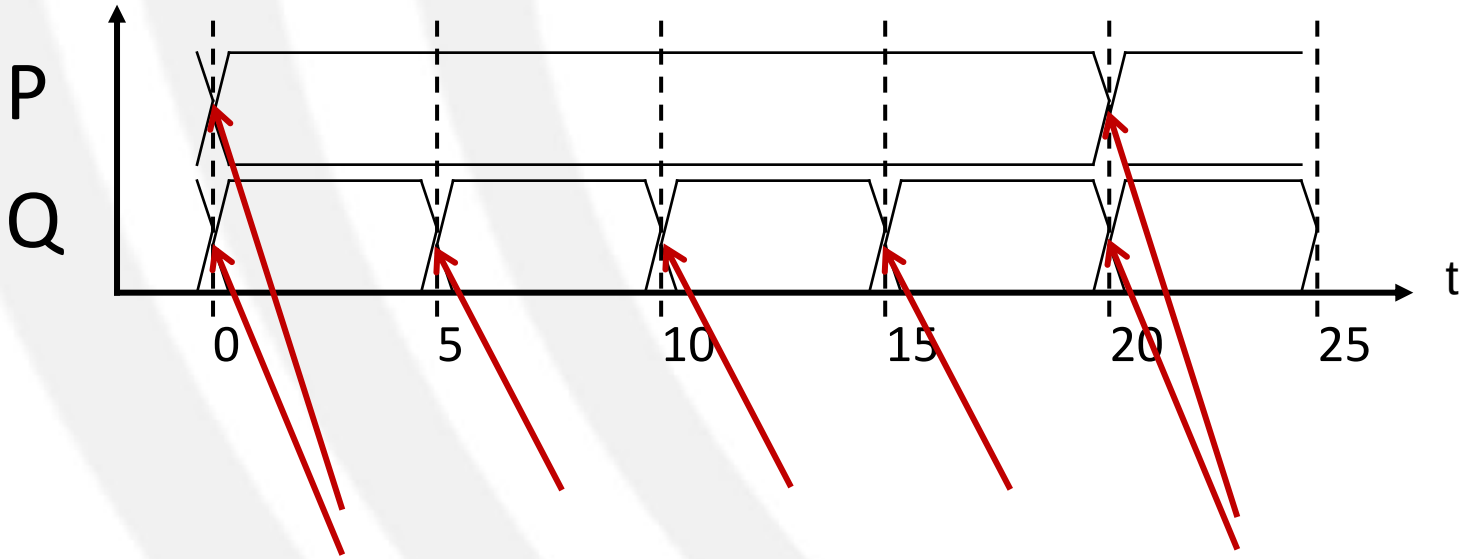
- Input and output ports of a Giotto task are updated logically at the beginning and at the end of the task period, respectively
- A Giotto task does not have to be started at the beginning of its period, it has only to be started and be finished sometime during its period

Language Semantics: Task (II)



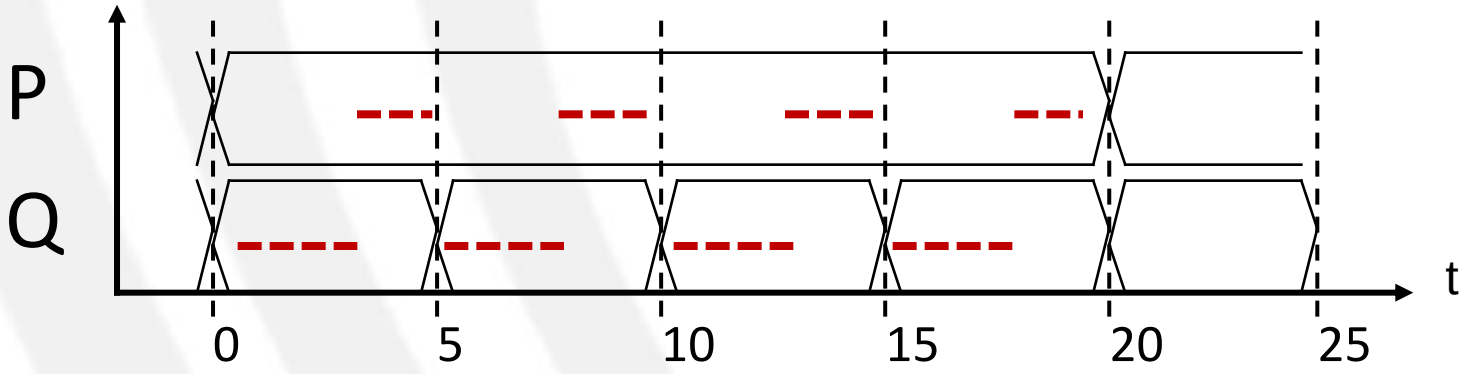
Input port updates

Language Semantics: Task (II)



Output port updates

Language Semantics: Task (II)

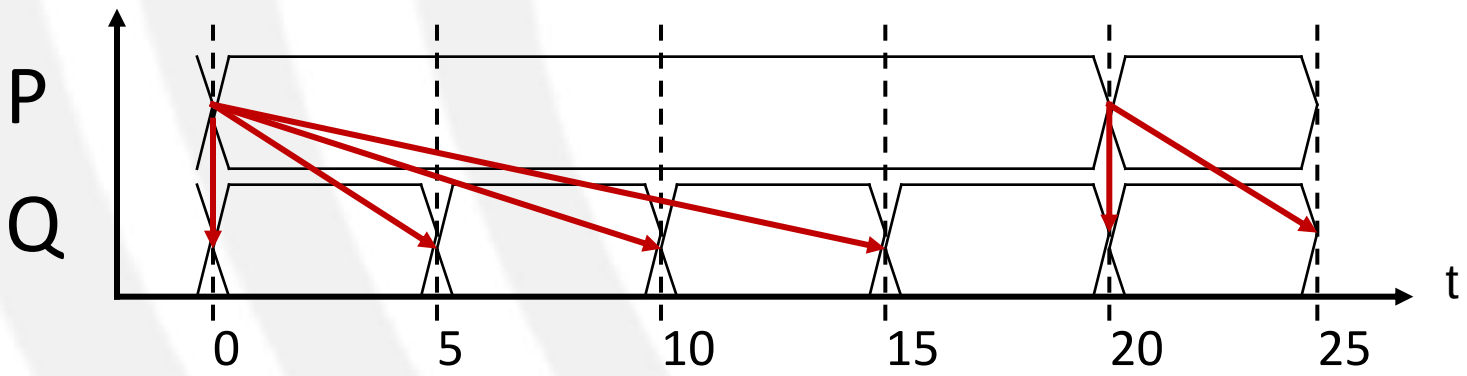


Task code execution

Language Semantics: Driver (III)

- Since the result of a task computation is written at the end of the task period, task drivers only cause data flow from past task invocations to current invocations, and not between current invocations
- Note that no matter when a task P finishes, its results are buffered until the end of its period
- Only at the end of the period, the concurrent process Q can see that results

Language Semantics: Driver (IV)



Data flow using drivers

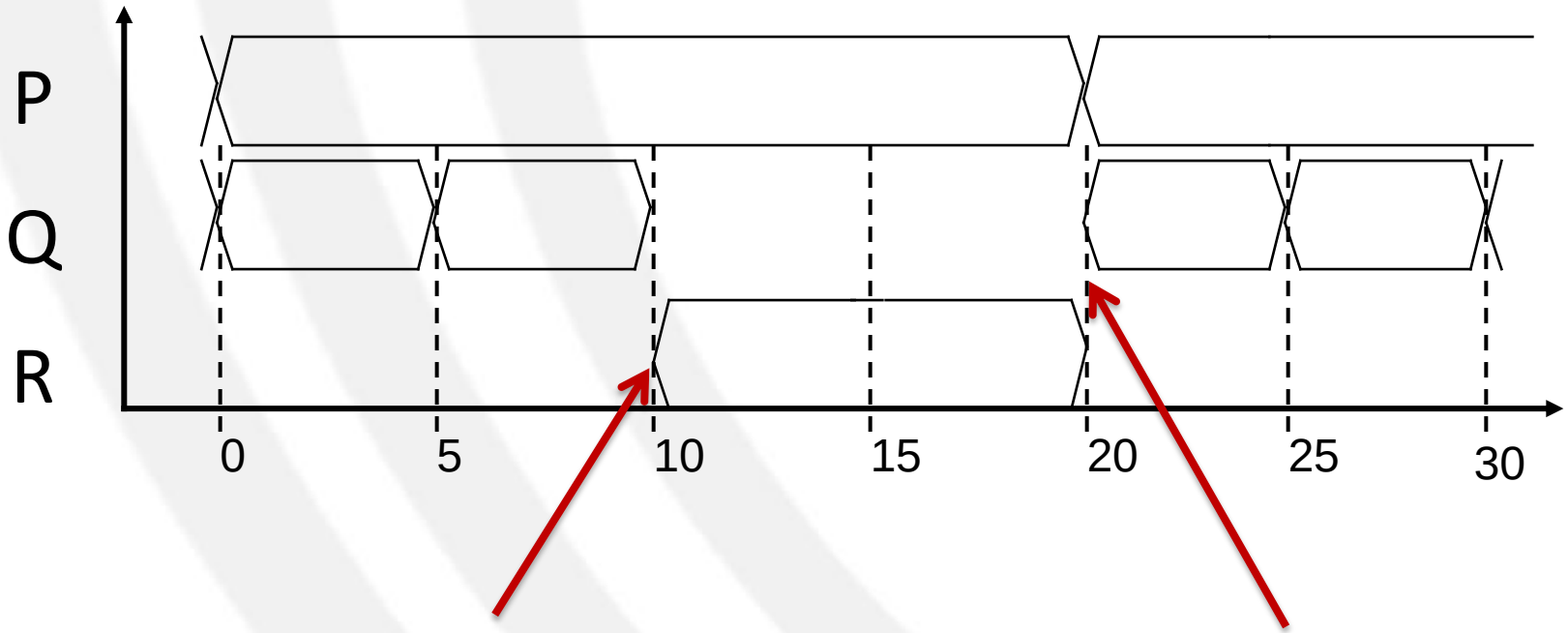
Language Semantics: Mode Switch (V)

- In Giotto, a task is considered a unit of work, which, once started, must be allowed to complete
- A mode switch may cease the periodic invocation of a task if that task period ends at the time the mode-switch guard is evaluated true
- A mode switch must not terminate any task whose period has not ended

Language Semantics: Mode Switch (VI)

- If a task is active when a mode switch occurs, then the Giotto semantics requires that the next mode again contains the task
- The least common multiple of all task periods, actuator update periods and mode switch periods of a Giotto mode determines the *period* of the mode

Language Semantics: Mode Switch (VII)

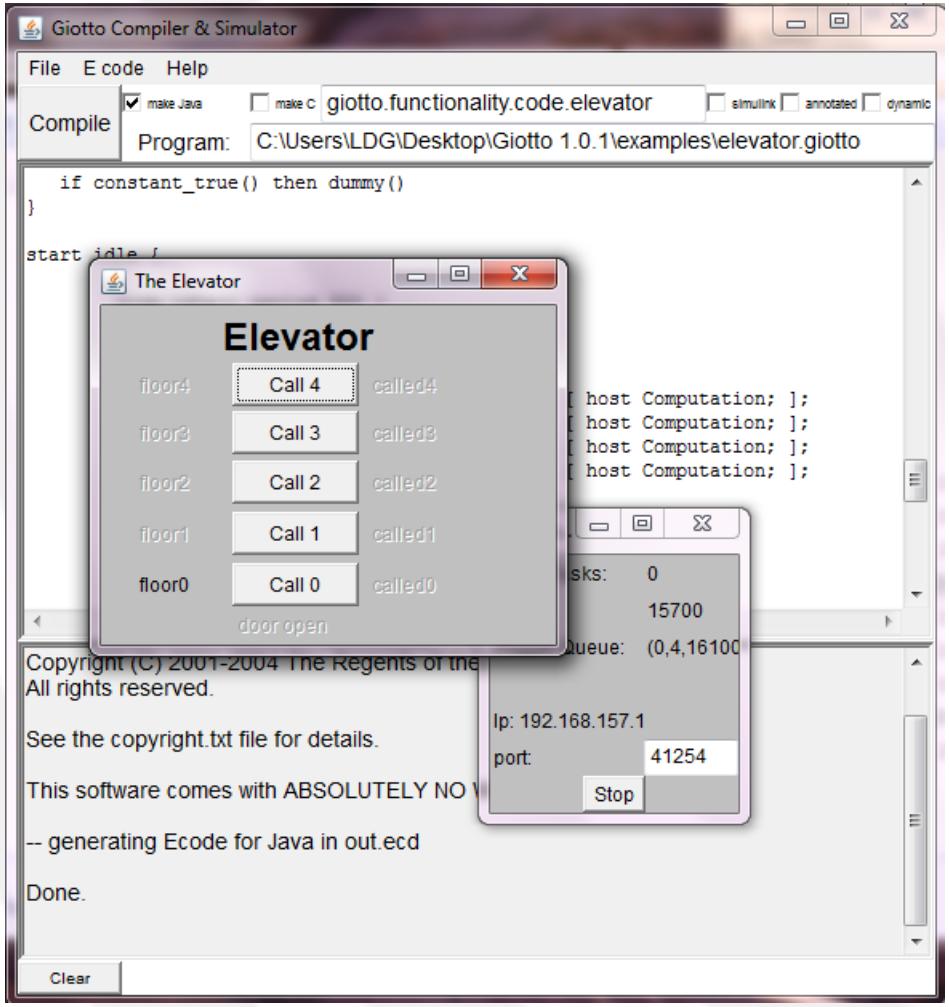


Summary: Giotto

- Giotto is a design methodology for embedded control systems.
- The programmer specifies the platform-independent programmer's model in the time-triggered programming language.
- The Giotto compiler produces executables combined with a run-time library for a particular platform.

EXAMPLE

The Elevator (I)



The Elevator (II)

- It is provided within the GIOTTO archive (`~dgglgu08/archives/Giotto.1.0.1.tgz`)
- Let's analyze the code identifying the different components required by a Giotto program
  - Modes
  - Mode switches
  - Drivers
  - Ports