An Programming Language with Fixed-Logical Execution Time Semantics for Real-time Embedded Systems

embedded systems perspectives and visions

Dr. Marco A.A. Sanvido
University of California at Berkeley
EmSys Summer School
Salzburg June 30th – July 2th, 2003

This is a joint work with:
A. Ghosal, Dr. C. Kirsch, Prof. T. Henzinger
This work is funded by:
GSRC, CHESS

More info: www-cad.eecs.berkeley.edu/~xgiotto
Motivation

- Extending Giotto semantics for event-triggered systems
  - Giotto is purely time-triggered
- A structured programming for E code
  - Giotto generates only single-triggered E code (triggers only a timer)
- Functionality part of the Language
  - Single stand-alone language
Safety Critical Embedded Systems: Avionics

Number of displays in commercial airplane cockpits

Complexity is too high, pilots unable to handle it!

Embedded System Programming

- Number of language features
- Number of APIs

Complexity is too high, programmer is overloaded with information!

Domain Specific Languages
Domain Specific Languages

■ Advantages:
  ○ “Ultimate abstraction”: capturing precisely the semantics of the application domain

■ Disadvantages:
  ○ Design and implementation is difficult
  ○ Resist evolution

Domain of the xGiotto Language

- Driven by interaction with environment
  - The environment dictates the speed
- Limited resources
  - The hardware imposes constraints
- Reliability
- Hard real-time constraints
The xGiotto Compiler

Similar to RISC/CISC situation:

“Something to keep in mind while reading the paper was how lousy the compilers were of that generation. C programmers had to write the word register next to variables to try to get compilers to use registers. As a former Berkeley Ph.D. who started a small computer company said later, "people would accept any piece of junk you gave them, as long as the code worked." Part of the reason was simply the speed of processors and the size of memory, as programmers had limited patience on how long they were willing to wait for compilers.”

Outline

- Giotto key-aspects
- xGiotto key-aspects
  - Fixed Logical Execution Time
  - Semantics
  - Conference Program
- Analysis
  - Environment assumptions
  - Race conditions detection
  - Time Safety
  - Program Classes
- Implementation
- Future Research Perspectives
- Related Work
- Questions
Time Determinism:

The Giotto compiler chooses for a given platform a platform timeline that is value equivalent to the environment timeline defined by the Giotto semantics.

Value Determinism:

For a given time-triggered sequence of sensor readings, the corresponding time-triggered sequence of actuator settings is uniquely determined (i.e., there are no race conditions).
xGiotto Key-Aspects

Platform Independence ensures Predictability

Event Determinism:

The xGiotto compiler chooses for a given platform a platform timeline that is value equivalent to the environment timeline defined by the xGiotto semantics.

Value Determinism:

For a given event-triggered sequence, the corresponding sequence of port values is uniquely determined.
Giotto Timeline

```
mode normal() period 1000 {
  taskfreq 1 do P1(getP1in);
  taskfreq 2 do P2(getP2in);
}
```

period 1000
Fixed Logical Execution Time

fixed logical execution time

computation → P1

invocation → termination
Fixed Logical Execution Time

The platform will allocate the computation.
PORT
    INTEGER p;
    INTEGER q;

EVENT
    INTEGER A AT X;

TASK P1(INTEGER a)
    OUTPUT (INTEGER b) {
        b := b + 1;
    }

TIMING T1() {
    P1(p)(q)[A];
}

{[A]T1();}
xGiotto Timeline
**xGiotto Semantics**

- **Task Invocation:**
  - \( t(\pi) \ (p_0)[E] \): When \( E \) arrives \( p_0 \) is written with the evaluation of \( t \) (on the value of \( \pi \)).

- **Events:**
  - External events: generated externally by an interrupt.
  - Completion events: internal events generated at the end of a task execution.
  - Combined events: concatenation of event expressions.

- **Reaction:** (event, action)
  - Timing reactions: action is an invocation of a block of xGiotto code.
  - Deactivation reactions: action is removal of a timing reaction.
  - Termination reactions: action is updating of output ports with the evaluation of task.
Example: A Conference

PROGRAM conference {

TYPE paper ARRAY 15 OF INTEGER;
PORT
    paper p1;
    paper p2;
EVENT
    INTEGER A AT CallForPaper;
    INTEGER D AT Deadline;

TASK Writing (INTEGER pages, INTEGER seed)
    OUTPUT (paper p) VAR (INTEGER j) {
        j = 1;
        while (j < pages) {p[j] = MakeScience(seed); j++;}
    }

TIMING Call() {
    Writing(15, 1)(p1)[D];
    Writing(15, 2)(p2)[D];
    [A]Call();
}

{[A]Call()}
}
Analysis

- Configuration Graph
- Port Abstract Reaction Graph
- Termination Reaction Graph
  - Port Update Conflicts
  - Adding Environment Assumption
  - Time Safety
Configuration: 
(port state, \(Y\) timing reaction set, \(\Psi\) deactivation reaction set, termination reaction set) 
c = (p, tir, der, ter) 
size for propositional Program: 
\(2^p \times |TI| \times |U| \times 2^P|T||E|\)
xGiotto Graph Abstractions

Port State Abstracted Configuration Graph

Termination Reaction Graph
Race Condition Detection
(Port Update Conflict)

- Importance
  - A port may be updated by multiple values at an instance

- Problem Definition
  - Given a xGiotto program, the port update conflict verification returns the answer YES if the program is conflict free, NO otherwise

- Hardness
  - The problem is PSPACE-complete for propositional xGiotto program

- Analysis
  - Search on Termination Reaction Graph
PROGRAM Example {

PORT
  BOOLEAN X1;
  BOOLEAN X2;

EVENT
  BOOLEAN A AT interrupt1;
  BOOLEAN B AT interrupt2;
  BOOLEAN C AT interrupt3;

TASK Inverter (BOOLEAN in) OUTPUT (BOOLEAN out) {out = !in; }
TASK Buffer (BOOLEAN in) OUTPUT (BOOLEAN out) {out = in;}

TIMING T1() {
  Inverter(FALSE)(X1)[A];
  Inverter(TRUE)(X2)[B];
  [C]T2();
}
TIMING T2() {
  Inverter(TRUE)(X1)[A];
  Buffer(TRUE)(X2)[B];
  [C]T1();
}
}
Port Update Conflict
Environment Assumption

Program

Reduction of Termination Reaction Graph
Adding time information for Schedulability

Event time Automaton:
Arrival of events and their time
(simplified version of Alur-Dill Timed Automaton, i.e. one single clock, integer time)
Time Safety

- xGiotto: EXPTIME
- Constant-FLET: P
- Giotto: P

All
Acyclic Deadline Monotonic
Earliest Deadline First
xGiotto Program Classes

xGiotto

FLET

Constant-FLET

Giotto
Implementation

xGiotto

FCode → ECode

decoupling: computation, reaction, scheduling

FMachine

EMachine

OS (SMachine) + Platform
Related Works

- Giotto
- Esterel
- nesC (TinyOS)

Future Research Opportunities

- More freedom for platform-dependent optimizations [LCTES03 papers!]
  - Dynamic Voltage Scaling (DVS)
  - Code size/speed/power

- Anytime scheduler
  [Real-Time Adaptive Resource Management for Advanced Avionics
  - Task able to deliver partial, but valid outputs
An Programming Language with Fixed-Logical Execution Time Semantics for Real-time Embedded Systems

embedded systems perspectives and visions

Dr. Marco A.A. Sanvido
University of California at Berkeley
EmSys Summer School
Salzburg June 30th – July 2th, 2003

This is a joint work with:
A. Ghosal, Dr. C. Kirsch, Prof. T. Henzinger
This work is funded by:
GSRC, CHESS

More info: www-cad.eecs.berkeley.edu/~xgiotto