INF624 - Real-Time Systems & Fault Tolerance

Flávia Maristela

Instituto Federal da Bahia)
Especialização em Computação Distribuída e Ubíqua (ECDU)

Salvador, Outubro de 2013

Schedule

- Computers which interact directly with hardware devices are commonly called *embedded systems*
- Timely response is an important factor in all embedded systems
- In some cases, very fast response is not necessary
- We may look at a real-time system as a stimulus/response system
- System behavior may be determined by listing
  - received stimuli
  - associated responses
  - the time at which the response must be produced.

- A real-time system has to respond to stimuli that occur at different times.
- System architecture must be organized so that as soon as a stimulus is received, control is transferred to the correct handler

Important

This does not work in sequential programs. So, **real-time systems are normally designed as a set of concurrent, cooperating processes**

- Model the application environment
- Real-time systems have important interactions with their environment
- Hardware issues are not totally “transparent”:
    - Maximum rate of interrupts
    - Maximum number of dynamic external objects
    - Failure Model

What must be modeled? (Artifacts)

- Data Flow
- Task Communication* (semaphores)
- Task Synchronization* (usually, message passing)
- Task Dependency
- Timing constraints

- Most important stage of development $\rightarrow$ generation of consistent design
- May be understood as **architectural design** and **detailed design**

RTS are no different from general purpose applications

Software engineering ensures that software production process is manageable and reliable programs are constructed

However...

Software engineering is not focus for most real-time systems

Design Process

- Design process involves deciding which capabilities should be implemented in:
  - software or
  - hardware.
- For embedded systems both power consumption and hardware cost are critical
- Processors must be designed to support embedded systems
- Special-purpose hardware may have to be designed and built.

Design Process

Important!

“a top-down design process - where the design starts with an abstract model that is decomposed and developed in a series of stages - is impractical for most real-time systems”

- For real-time systems, low-level decisions on hardware and system timing constraints must be considered early
- As a consequence:
  - System designer flexibility is limited
  - Maybe additional software functionality (eg.: battery, power) is required.

Design Process

Events (or stimuli) “rather than objects or functions should be central to the real-time software design process.”

Design Process

- Interleaved stages in design process [Sommerville]:
  1. Identify the stimuli that the system must process;
  2. Identify associated responses;
  3. Identify the timing constraints that apply to both stimulus and response processing;
  4. Choose an execution platform for the system (includes hardware, real-time operating system).
  5. Aggregate the stimulus and response processing into a number of concurrent processes;
  6. For each stimulus and response, design algorithms to carry out the required computations.
  7. Design a scheduling system that will ensure that processes are started in time to meet their deadlines

Design Process

Processes in a real-time system have to be coordinated

Processes in a real-time system have to be coordinated

- Process coordination mechanisms ensure mutual exclusion to shared resources.

Design Techniques

- For real-time systems purposes three design techniques are named:
  - Informal: based on natural languages. Ambiguous in most cases
  - Structured: based on well-defines graphical representation.. Despite rigorous, cannot be formally analyzed or manipulated
  - Formal: methods with mathematical properties. Sometimes cannot be easily understood. May be inadequate for large applications
- Some methods has shown to be more suitable: UML, PAMELA, HOOD V4, EPOS and HRT-HOOD

Design Methods

- A real-time system design method must be able to:
  1. Structure a system in concurrent tasks
  2. Support the development of reusable components through information hiding
  3. Define behavior aspects using finite-state machines
  4. Analyze the properties of a design to determine its real-time properties

UML

- Real-time UML is **standard UML**

“UML is adequate for real-time systems” Grady Booch

“Although there have been a number of calls to extend UML for real-time domain, experience had proven this is not necessary” Bran Selic

UML

- Real-time and embedded applications require special concerns about:
    - QoS (WCET, Predictability, Reliability)
    - Low-level programming
    - Safety and reliability

UML

- Most useful UML diagrams for real-time systems
  1. Behavioral:
     - Statechart diagrams
  2. Interaction:
     - Sequence
     - Timing Diagram
  3. Functional:
     - Use Case Diagram

UML on the other hand...

It may not be possible to use object-oriented development for hard real-time systems

Sometimes, it may be better to implement some system functions, in hardware rather than in software in other to meet timing constraints. “Hardware components deliver much better performance than the equivalent software.”

UML on the other hand...

- UML increases notation.
- ROPES: *Rapid Object-Oriented Process for Embedded Systems*, was proposed

Design Activities

- Decomposition: systematic breakdown of a complex system into smaller parts until components are isolated and can be engineered
  - Each decomposition level must have an appropriate documentation
- Abstraction: allows a simplified view of the system

- Programming languages must include facilities to access the system hardware
- It should be possible to predict the timing of particular operations in these languages

Assembly

- Enables full hardware control
- Difficult language to manipulate - low level programming language
- Specially used in hard real-time systems, so that tight deadlines can be met

C

- enables interaction with several hardware devices
- allows efficient code generation.
- allows the development of very efficient programs
- do not include constructs to support concurrency or the management of shared resources
    - concurrency is implemented through calls to the real-time operating system
    - compiler cannot check errors $\rightarrow$ favors more programming errors.
    - programs are more difficult to understand $\rightarrow$ real-time features are not explicit in the program.

Java

- there has been extensive work to extend Java for real-time systems development:
- More aspects that need to be adequate:
  1. For definition it is not possible to specify the time at which threads should execute.
  2. Timing behavior of threads may not be predicted
  3. Java Virtual Machine varies from one computer to another, so the same program can have different timing behaviors.
  4. There are no standard ways to access the hardware of the system.

Real-Time Systems and Programming Languages. Alan Burns and Andy Wellings.

Software Engineering. Ian Sommerville.