SMILIES

reSilient coMputer archItectures
and LIfE Sciences

Politecnico
di Torino

1859

Department of Control and
Computer Engineering

DAUIN

SCHEDULING PREDICTABILITY

STEFANO DI CARLO

PREDICTABLE HARDWARE SUPPORT

- Predictability: the OS should guarantee that all critical timing constraints are met
- Predictability requires determinism
  - Deterministic algorithm is an algorithm which, given a particular input, will always produce the same output, with the underlying machine always passing through the same sequence of states
- Determinism is influenced by
  - The architectural features of the hardware
  - The scheduling algorithm
  - The IPC mechanisms
  - …

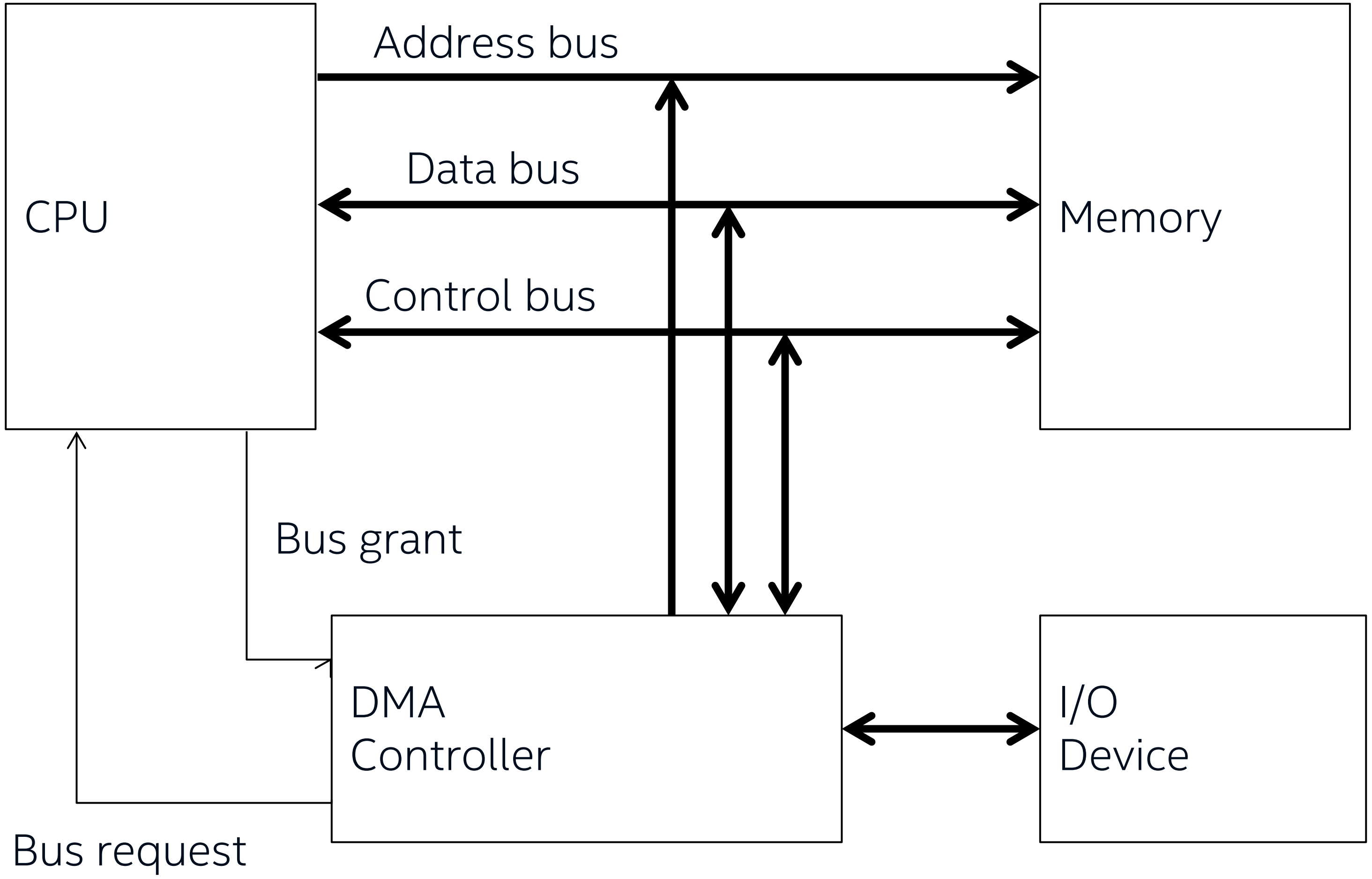
HW/SW SOURCES OF NONDETERMINISM

- Direct memory access (DMA)
- Cache memory
- Interrupts
- System calls
- Semaphores
- Memory management
- Programming language

DIRECT MEMORY ACCESS (DMA)

- DMA is used by many peripheral devices to transfer data between the device and the main memory
  - It relieves the CPU of the task of controlling the I/O transfer
- The CPU and the I/O device share the same bus
- The CPU is blocked when the DMA device is performing a data transfer

DIRECT MEMORY ACCESS (DMA)



DIRECT MEMORY ACCESS (DMA)

- Data transfer mode
  - Burst
    - An entire block of data is transferred in one contiguous sequence
    - The CPU remains inactive for relatively long periods of time (untill the whole transfer is completed)
  - Cycle stealing
    - DMA transfers one byte of data and then release the bus returning control to the CPU
    - It continually issues requests, transferring one byte of data per request, until it has transferred the entire block of data
    - It takes longer to transfer data/the CPU is blocked for less time
  - Transparent
    - DMA transfers data when the CPU is performing operations that do not use the system buses

DIRECT MEMORY ACCESS (DMA)

- DMA makes difficult to predict task response time
- The time the task remains blocked depends on
  - The size of the block of data
  - The transfer mode
- Real-time oriented transfer mode
  - Time-sliced method
    - Each memory cycle is split into two adjacent time slots: one reserved for the CPU and the other for the DMA device
    - More time expensive but more predictable
    - CPU and DMA never interfere with each other
    - Task completion time does not depend on the DMA transfer

CACHE MEMORY

- Fast memory inserted as a buffer between the CPU and the RAM to speed up task execution
- At each memory access the hardware checks whether the requested information is stored in the cache:
  - Cache hit → data are read from the cache (very fast)
  - Cache miss → data are read from the RAM; the data is copied in the cache along with a set of adjacent locations (very slow)
- Exploit the program locality concept

CACHE MEMORY

- Sources of nondeterminism
  - Hit-ratio improves the performance on average
    - 80% of the time the data is in cache, 20% it is not
    - How to compute precisely the worst-case execution time?
  - In preemptive system, hit-ratio is affected by the number of preemption
    - Preemption destroys program locality, and increases the number of cache misses
- Example
  - PowerPC MPC7410 with 2 MByte two-way associative L2 cache
  - 33% WCET increment due to cache interference w.r.t. non-preemptive mode

CACHE MEMORY

- Solutions
  - Disable the cache memory
    - Impact the performance
    - More deterministic behavior
  - Overestimate the WCET
    - Does not impact the performance
    - Uses only a fraction of the available CPU time

INTERRUPTS

- They are event generated by I/O devices to get the attention of the CPU when a data is available
- The arrival of an interrupt causes the execution of an interrupt service routine
  - Fragment of code dedicated to the management of its associated I/O device
- Example: to get data from an I/O device, each task must
  - Enable the hardware to generate interrupts
  - Wait for the interrupt
  - Read the data from a memory buffer shared with the driver

INTERRUPTS

- In general-purpose OS
  - Interrupts are served using a fixed priority scheme
  - Each driver is scheduled based on a static priority, higher than process priorities
    - Interrupt service routines usually deal with I/O devices that have real-time constraints, whereas most application programs do not
- In real-time OS
  - Tasks (e.g., control process) could be more urgent than an interrupt service routine
  - It is very difficult to bound a priori the number of interrupts that a task may experience
  - The delay introduced by interrupts on tasks execution is unpredictable

INTERRUPTS

- Solution 1
  - Disable all interrupts but timer interrupt (needed for scheduling tasks)
  - I/O devices are handled using polling
    - The application task is responsible for checking I/O devices periodically for available data
  - Deterministic but requires busy waiting for communicating with I/O device
  - The I/O handling has the same priority of the application task where it is implemented

INTERRUPTS

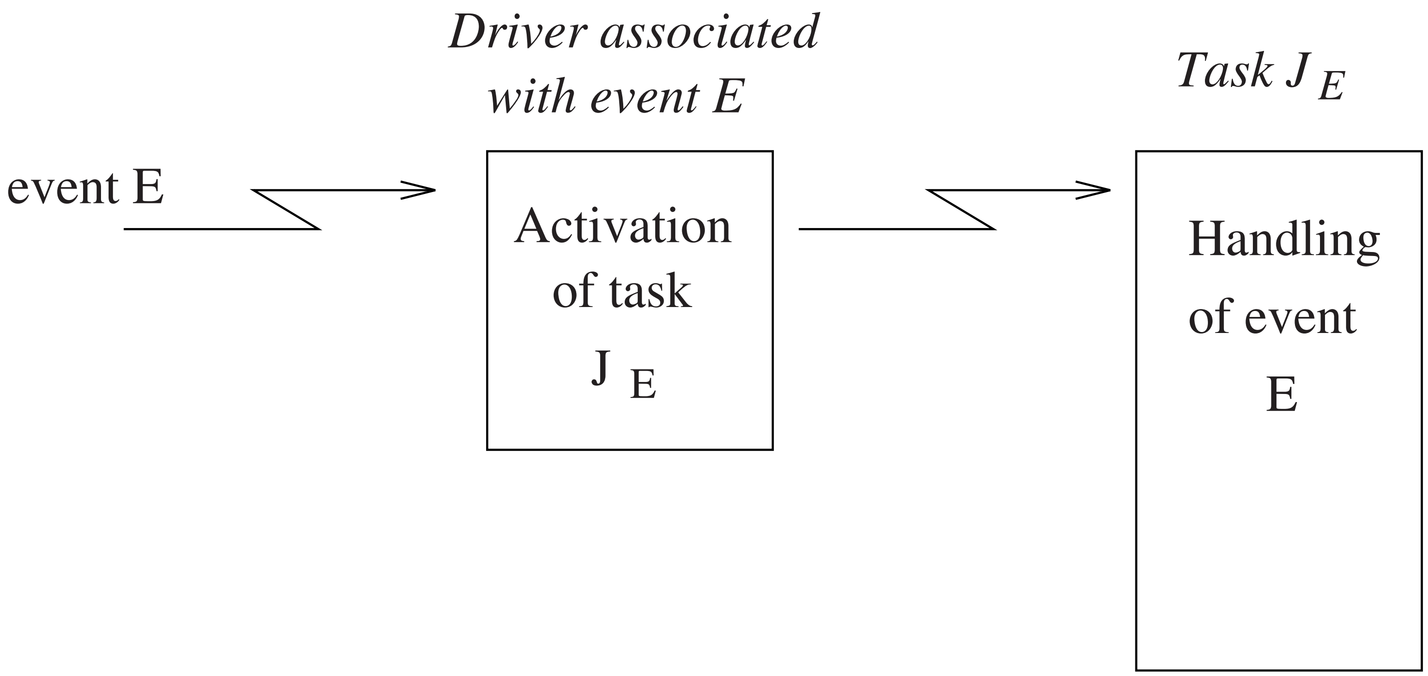
- Solution 2
  - Disable all interrupts but timer interrupt
  - Treat the I/O devices as aperiodic tasks managed through servers
    - Slow devices are multiplexed and served by a single server running at a low rate
    - Fast devices are served by dedicated servers running at higher frequencies
  - Servers are implemented by the OS
  - Still require busy waiting in the servers
  - Servers are scheduled accordingly to their priority, independently from those of application tasks

INTERRUPTS

- Solution 3
  - All interrupts are enabled
  - The interrupt handling is performed using
    - A driver: its only purpose is to activate the task associated to the I/O device
    - A task: it is responsible for handling the I/O device
  - The driver execution time is very limited → negligible impact on the tasks WCET
  - The I/O task is scheduled by the OS based on its priority, independent from those of application tasks
  - No busy waiting

INTERRUPTS

- Conceptual view of solution 3



SYSTEM CALLS

- The execution time of system calls must be characterized, and considered when evaluating if the scheduling is feasible
- System calls must be pre-emptible
  - Any non-pre-emptible section could delay the activation or the execution of critical tasks
- Example
  - Linux vanilla server is not suitable for real-time
    - The kernel functions (including system calls) is not preemptable
  - Kernel extensions are needed to make is more deterministic
    - Basically, the kernel must become pre-emptible

SEMAPHORES

- Traditional semaphores are not suitable for real-time systems → may produce the priority inversion problem
- The OS must implement into semaphores the solutions to priority inversion
- We will study this in detail later …

MEMORY MANAGEMENT

- Memory management techniques must not introduce nondeterministic delays during the execution of real-time activities
- We will study this in detail later …

PROGRAMMING LANGUAGE

- Restrictions that may be needed for real-time
  - No dynamic data structures: dynamic arrays must be eliminated because they would prevent a correct evaluation of the time required to allocate and deallocate memory
  - No recursion: If recursive calls were permitted, the schedulability analyzer could not determine the execution time of subprograms involving recursion or how much storage will be required during execution
  - Time-bounded loops: In order to estimate the duration of the cycles at compile time, programmer should specify for each loop construct the maximum number of iterations

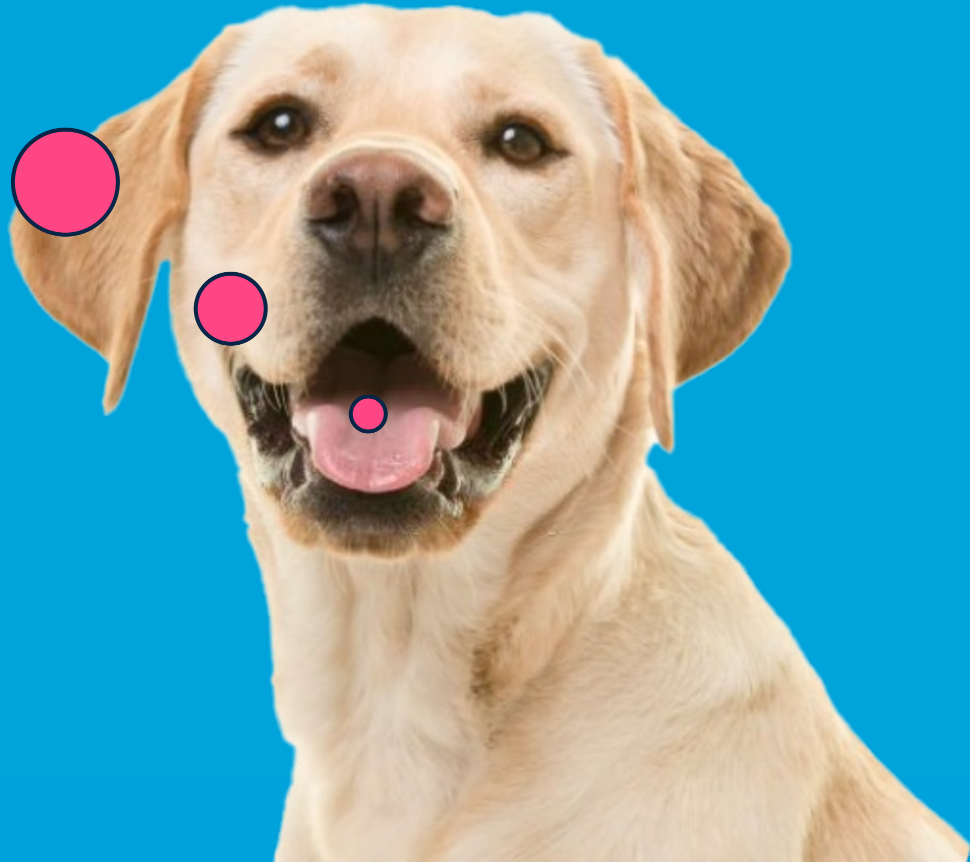
smilies
reSilient coMputer archItectures
and LIfE Sciences

Politecnico
di Torino

Department of Control and
Computer Engineering

DAUIN

QUESTIONS?



THANK YOU!