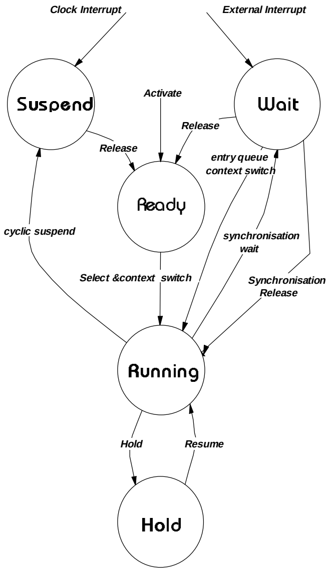
The new ESA 32bit processor chipset is a fundamental building block for the next generation of space systems. Spacebel Informatique's Scheduler Simulator is a crucial tool that helps in developing and testing characterised and dependable software systems.

ERC32 Scheduler Simulator

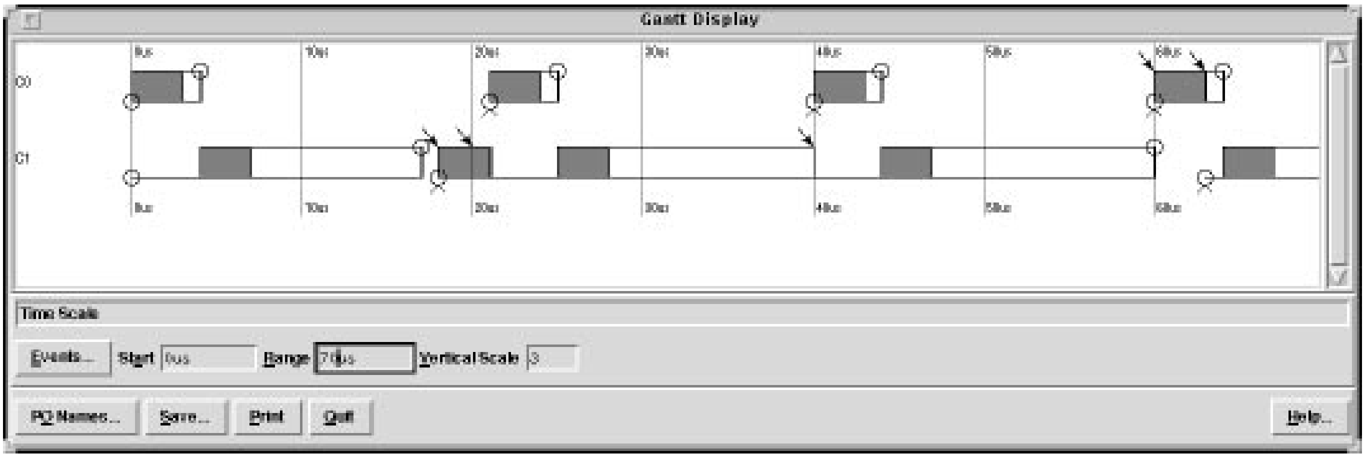
Overview

The Scheduler Simulator provides a real-time system designer with the means to visualise the execution of a task set and scheduling characteristics. The visualisation provides the ability to investigate the circumstances under which deadlines are missed, the tightness of schedules and the CPU utilisation. The Scheduler Simulator complements the Schedulability Analyser's formal approach by showing how the designed system responds under different release patterns for interrupts and when relaxing the critical instant assumption.

The Scheduler Simulator provides the user with detailed reports and summaries of the progression of a task set and the supporting ADA run-time. The characteristics are presented through the display of *scheduling events* that occur during the modelled application's execution lifetime. The scheduling events are displayed either in graphical form or textual. The display is not dynamically generated whilst the simulation is in progress but generated after the simulation has ended.
The Scheduler Simulator is fully integrated with the Schedulability Analyser.



Computational Model



Start 0us

Range 70us

Vertical Scale 3

Capability

The Scheduler Simulator provides the following capabilities:

- Configuration of real-time system to be simulated
- Automatic assignment of task and protected object priorities through the Schedulability Analyser
- Complete GUI interface
- Simulation of task set over an *Observation Window*, and including:
  - An *On-Line Trace* to monitor for erroneous task behaviour - *Missed-Deadlines*,
  - Configuration of *Interrupt Release patterns* according to different statistical distribution criteria: poisson, normal, uniform, constant.
- Generation of significant *Scheduling Events*
- Control of Simulation including suspension and resumption
- Saving *Simulation Context* with later resumption for long simulations
- On screen Generation of Reports from simulation filtered by task and including: Gantt Reports and text Reports
- Characterisation of Scheduling events by:
  - Thread to which an event occurs
  - Event kind
  - Time of occurrence
  - Priority of thread
- Generation of Utilisation Factors including individual threads, overall Thread set and RTS
- Generation of reports on tightness of schedule and deadline margins on a per thread basis
- Generation of reports in postscript
- Gantt Report providing visualisation of task execution
- Direct indications of missed deadline
- Direct indication of the early release of Sporadic and Interrupt tasks
- Generation of Reports based on historic simulations
- Search event facilities including:
  - searching for event sequences for particular events and sequences of events,
  - automatic update of Gantt and Textual displays on successful searches.

Interfaces

Scheduler Simulation Window - used to configure the simulation for a particular application,

Simulation Window - used to control simulation and resume historic simulations,

On-line Trace Pane - used to follow a simulation *on-line* ,

Interrupt Configuration - used to configure release patterns of interrupts,

Report Window - used to configure and search through a Simulation to be reported,

Textual Window - used to display a textual report of a simulation,

Gantt Window - used to display a visualisation of task scheduling and interaction

ERC32

Scheduler Simulator

Producing dependable and predictable systems is a difficult task. The Scheduler Simulator provides a valuable help to constantly monitor the scheduling feasibility during the complete development process.

Models

Application Model

The Application Model is identical to one used in the Schedulability Analyser.

Computational Model

The computational model adopted for Scheduler Simulation takes an architectural view of the Ada runtime system. The model assumes a number of tasks that executes on a single processor system. Each task has an associated priority that for hard tasks is unique but for soft tasks several may share the same priority. The model assumes a Priority pre-emptive Scheduler that always executes the highest priority task that is ready to run. The task states and state transitions are depicted in figure on the front page.

The computational model supports the Immediate Priority Ceiling Inheritance blocking protocol. A task's priority is increased to its ceiling priority when entry to a protected object is made and subsequently reduced to its previous value when it is left. The model supports nested protected object calls.

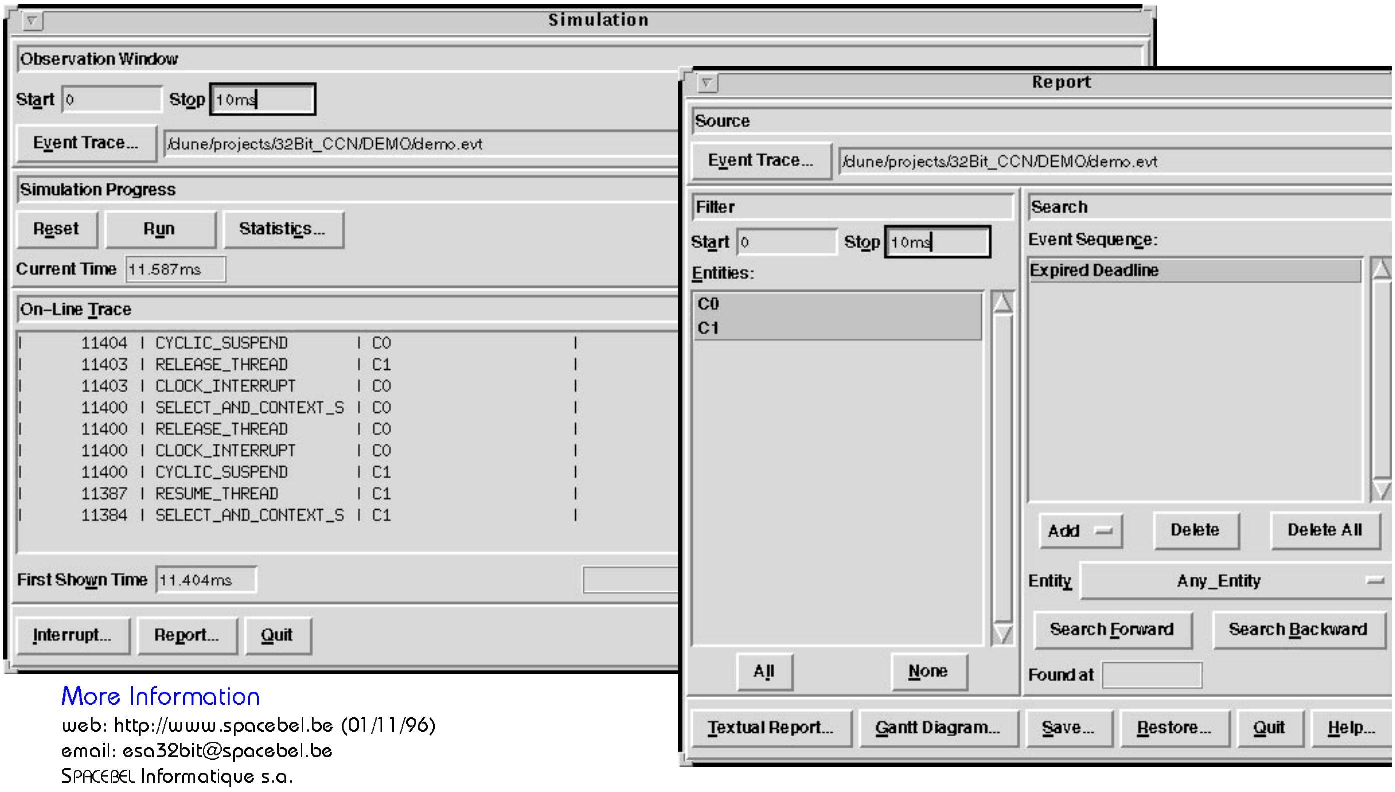
The model includes two fixed queues and other queues that generalise the ADA run-time system. These include:

- A Ready Queue that contains a list of tasks that are ready to run but have not sufficient priority to pre-empt the executing task. The list consists of all tasks in the Ready and Hold state.
- A Delay Queue that contains a list of all tasks that are awaiting expiration of a time interval. The queue is maintained in chronological order. The queue corresponds to those cyclic tasks that are in the Suspend state.
- An Entry Queue for each synchronisation object. This queue contains at most one task that is awaiting entry to the synchronisation object.

The ADA run-time system Delay Queue is implemented through an Interval Timer. This is a timer that may be pre-set and which only interrupts the processor after expiration of the pre-set time interval. On expiration of the timer all tasks that have expired are transferred to the Ready Queue and the timer is re-set for the least delay of all suspended tasks.

The ADA run-time system is supplied with a number of metrics that quantify the system cost of performing run-time system operations on behalf of the task set. These metrics are used by the Scheduler Simulator to determine the interference of the ADA run-time system on the task set.

- Timer Interrupt Overhead is the time used by the run-time system to process the interrupt occurring at the expiration of the hardware interval timer up to the start of the loop that processes tasks on the Delay Queue.
- Ready After Delay is the time to move a collection of tasks, whose delay has expired, from the Delay Queue to the Ready Queue. More than one task may be transferred and consequently the time is a function of the number of tasks to be readied.
- Enter Delay Until Overhead is the time to enter the run-time system from a cyclic task that has finished the execution of its current period and now requires to be entered into the Delay Queue. This overhead is a function of the remaining time before the next release of the task. It may involve the queuing of the task on the Delay queue (if the next release has not occurred yet) and a re-set of the timer. The time to select a new task to run is not included in this metric.
- Exit Delay Until Overhead is the time to continue a cyclic task when it resumes its execution.
- Context Switch Time is the time to switch to a new task such that the new task has command of the processor.
- Scheduler Select Time is the time taken to select a new task from the Ready Queue.
- Entry Queue Servicing Time is the time from entry to the ADA run-time system (after an accept statement) to the start of the context switch to the task awaiting entry to the protected object.
- Enter Passive task and Exit Passive Task are the times to enter and leave a passive task.
- Enter Software Sporadic Wait is the time to enter a passive task whose guard is closed to the start of the next selection of a task to execute. This time does not include guard evaluation.
- Exit Software Sporadic is the time that a task takes to re-enter the synchronisation object within which the task was previously suspended.
- Enter Interrupt Sporadic Wait is the time to enter the ADA run-time system from an accept statement within an Interrupt Sporadic up to the selection of a new task to execute.
- Exit Interrupt Sporadic Wait is the time that a task takes to re-enter an Interrupt Sporadic.
- Interrupt Handling Overhead is the time from the occurrence of an external interrupt to the time at which the selection of a new task occurs.
- Maximum Non Pre-emption Time is the maximum time when all scheduling is disabled in any of the ADA run-time system Code.



Start 0 Stop 10ms

/dune/projects/32Bit_CCN/DEMO/demo.evt

Current Time 11.587ms

```
|      11404  | CYCLIC_SUSPEND         | C0                   |
|      11403  | RELEASE_THREAD         | C1                   |
|      11403  | CLOCK_INTERRUPT        | C0                   |
|      11400  | SELECT_AND_CONTEXT_S   | C0                   |
|      11400  | RELEASE_THREAD         | C0                   |
|      11400  | CLOCK_INTERRUPT        | C0                   |
|      11400  | CYCLIC_SUSPEND         | C1                   |
|      11387  | RESUME_THREAD          | C1                   |
|      11384  | SELECT_AND_CONTEXT_S   | C1                   |
```

First Shown Time 11.404ms

Source: /dune/projects/32Bit_CCN/DEMO/demo.evt

Filter: Start 0 Stop 10ms

Entities: C0, C1

Event Sequence: Expired Deadline

Entity Any_Entity

Found at

More Information

web: http://www.spacebel.be (01/11/96)
email: esa32bit@spacebel.be
SPACEBEL Informatique s.a.

111, rue Colonel Bourg
1140 Brussels - Belgium
Tel: +32 2 730 46 11 Fax: +32 2 726 85 13

It is believed that the contents of this data sheet is correct at publishing time. As the product still evolves, future versions of the product might have slightly different characteristics and features.

Early access available through Estec evaluation programme.

SPACEBEL Informatique is an engineering company that specialises in dependable, real-time, parallel, reusable object oriented systems for the various space segments