
**Motivation**

- **Real-time parallel models are now common**
  - But little exists on fine-grained parallelism within real-time languages and runtimes

- **Ada is a language of choice for reliable real-time systems**
  - It incorporates models of computation which are amenable for real-time analysis
  - Ada 2012 supports many real-time multiprocessor scheduling schemes, global and partitioned

- **Existent multiprocessor support in Ada follows a coarse-grained model**
  - It needs to be augmented with lightweight fine-grained parallelism.
Definition of a parallel non-schedulable unit (Tasklet)

- Explicit or implicit parallelization
- Ada Tasks execute graphs of Tasklets

```ada
task body My_Task is
begin
  -- Code of A
  parallel
    -- Code of B
    for I in parallel Some_Rage loop
      -- D, E, ...
      end loop;
    and
    -- Code of C
  end;
  -- A again
  parallel
    -- Code of F
    and
    -- Code of G
    and
    -- Code of H
  end;
  Code;
  -- A again
end;
```
Tasklets are executed by Executors
E.g. OS threads, but can be bare metal entities

Limited form of run-to-completion
A tasklet is mapped to one executor, except if blocking
Executor might be scheduled in a preemptive, global or partitioned scheduling

Allocation of tasklets to executors, and of executors to cores is left to the implementation
Models are defined to guarantee safeness and progress, even with potential blocking operations
Real-time issues

- Each Ada task (or priority) is provided with a specific executor pool
  - All executors carry the same priority/deadline of the task
  - Does not support graph decomposition techniques

- Maps to a synchronous fork-join model
  - for which analysis already exists

Important open issues

- Supported scheduling models, such as limited preemption
- Run-to-completion and tasklet (work-)stealing
- Parallel models in languages introduce additional issues
  - (e.g. integration with task abortion, exceptions, etc.)