BENEFITS GAINED FROM USING ADA ON THE ASIM PROJECT

18th International Conference on Reliable Software Technologies — Ada-Europe 2013

Mark Lorenzen, Terma A/S
mal@terma.com

Background: The ASIM Payload

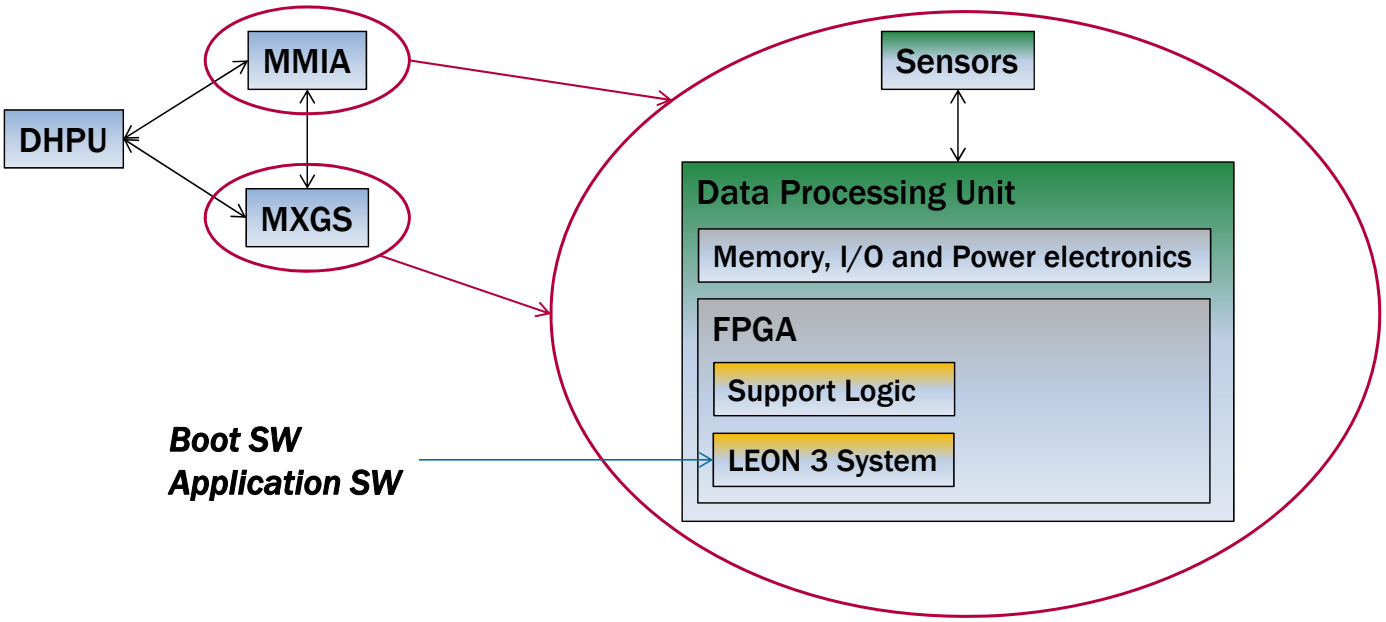
The Atmosphere-Space Interactions Monitor (ASIM):

- **Will be mounted on the Columbus module of the International Space Station:**
- **Will be used to detect lightning formations known as “red sprites”, “blue jets”, “elves” and to detect X-ray and γ-ray discharges.**

The objective is to search for a correlation between these formations and large thunderstorms, improving our understanding of these phenomena and their influence on the Earth’s climate.

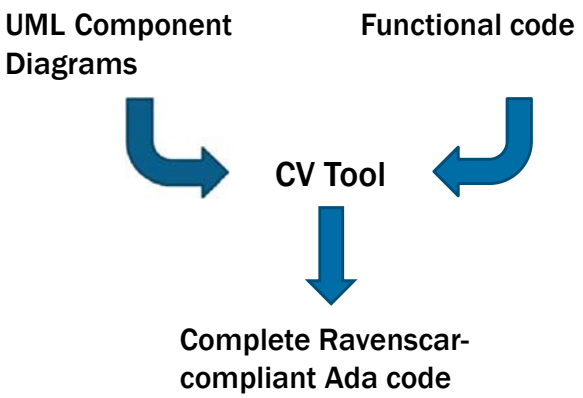
Terma A/S is – under contract to the European Space Agency (ESA) – the prime contractor for the development of the ASIM payload including development of the on-board software.

Background: System architecture



DHPU: Data Handling and Power Unit
MMIA: Modular Multi-spectral Imaging Array
MXGS: Modular X-ray and Gamma-ray Sensor

Model-driven development



UML	Ada
<<interrupt sporadic>>	Protected procedure
<<protected>>	Protected subprogram
<<sporadic>>	Task with protected queue
<<cyclic>>	Cyclic task
<<hardware>>	(nothing)
<<passive>>	Subprogram

In-house developed *Concurrency View* tool:

- **Generates concurrency code and subprogram calls that connect required interfaces to their connected provided interfaces;**
- **Supports task heartbeat command and response code;**
- **Supports task-level exception handling code;**
- **Performs schedulability analysis;**

Programmer focuses exclusively on *Functional View* (i.e. non-concurrency) code.

Ada 2012 features used

```
function Is_Full (Q : in Queue) return Boolean;

procedure Insert (E : in Element; Q : in out Queue)
  with Pre => not Is_Full (Q);

procedure Transmit_Data
  (Data     : in     Element_Array;
   Capacity : in     Element_Count;
   Count    :    out Element_Count)
  with Post => (Count <= Data'Length and Count <= Capacity);
```

Aspect Pre is used to avoid ”defensive programming”.

Aspect Post and pragma Assert are used to provide additional run-time checks.

Specifies the ”can never happen” cases and avoids problems obtaining test coverage of these cases.

Ada 2012 features used

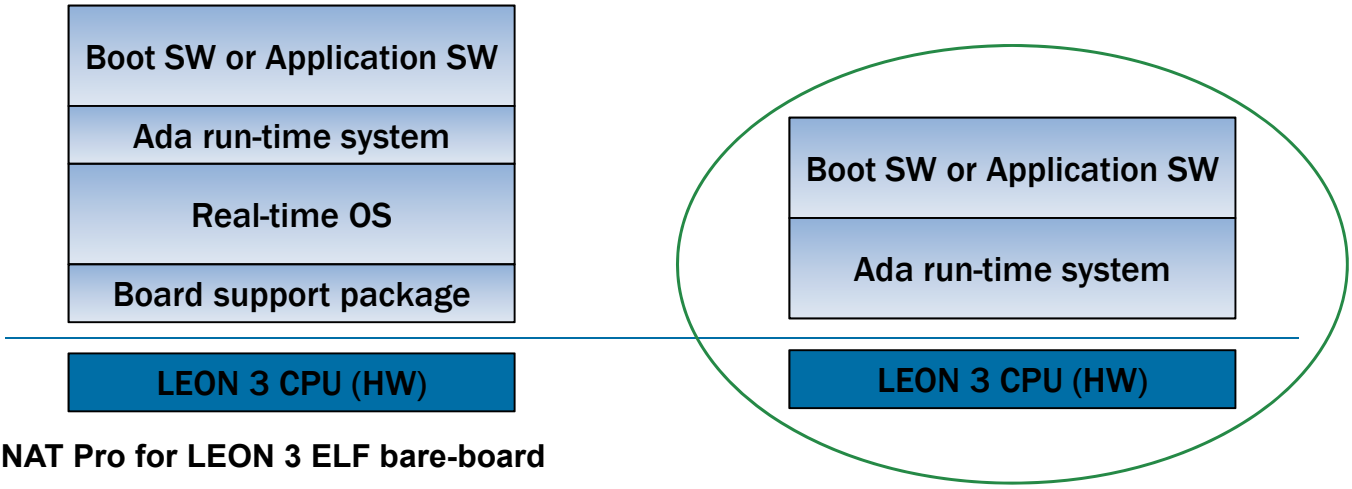
```
type External_Command_ID is mod 2 ** 16;

type Command_ID is range 1 .. 60_000
  with Static_Predicate => Command_ID in 1 | 3_000 .. 3_010 | 8_000 | 60_000;

function Convert is new
  Ada.Unchecked_Conversion (Source => External_Command_ID, Target => Command_ID);
…
ID := Convert (External_ID);

if ID'Valid then
  …
end if;
```

Aspect Static_Predicate is used e.g. when defining an integer type with ”holes”.

The check for Valid takes the Static_Predicate aspect into account ensuring that the value check is always performed together with the validity check.

Aspect Dynamic_Predicate is not used (yet).

Software stack



GNAT Pro for LEON 3 ELF bare-board

- **Ravenscar.**

Reduction of SW stack size and complexity.
Reduction of build complexity.
Optimized run-time system.
No device drivers.

Static analysis

The GNAT Pro tool chain is used to:

- **Check the code against a coding standard;**
- **Pretty-print the code;**

} Helps ensuring compliance to coding standard

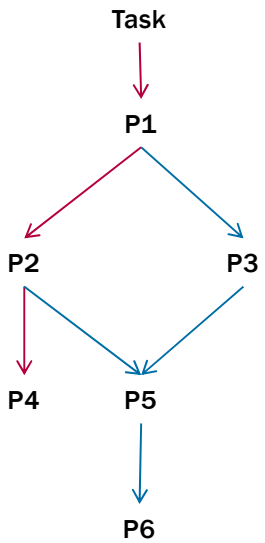
- **Calculate complexity and other metrics.**

GNATstack is used to:

- **Calculate an upper bound of the stack size requirements of all subprograms;**
- **Calculate an upper bound of the stack size requirements of all program entry points (tasks and interrupt handlers).**

The upper bound of the stack size requirements of a program entry point is determined from the worst case of all possible subprogram call chains.

Very useful for avoiding the hard-to-find ”blown stack” errors!



Static analysis

A surprise! Stack size requirements of a task suddenly increases by 1504 octets although no additional variables are declared.

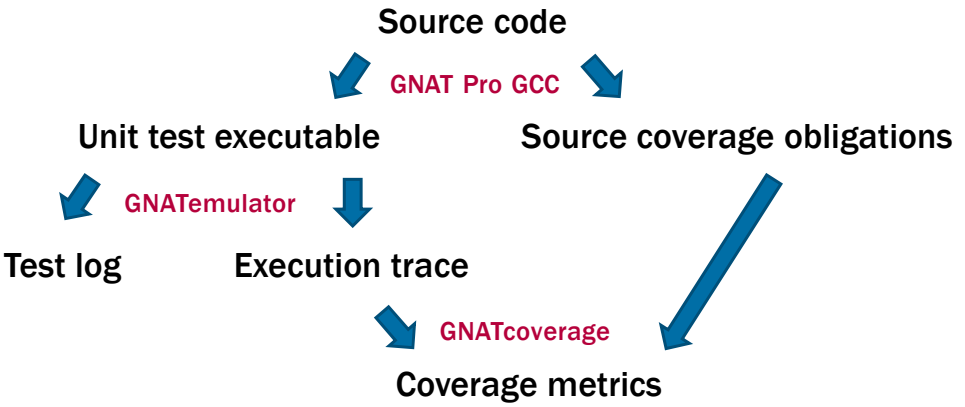
```
procedure P (Valid : in Boolean; Data : in Element_Array) is
  Frame : Communication_Frame;
begin
  if Valid then
    Frame.Valid := True;
    Frame.Data  := Data;
  else
    Frame.Valid := False;
    Frame.Data  := …;
  end if;
  …
end P;
```

```
type Communication_Frame is
  record
    Valid : Boolean;          <-- 4 octets
    Data  : Element_Array;    <-- 1500 octets
  end record;
```

```
procedure P (Valid : in Boolean; Data : in Element_Array) is
  Frame : Communication_Frame;
begin
  if Valid then
    Frame := Communication_Frame'(Valid => True, Data => Data);
else
    Frame := Communication_Frame'(Valid => False, Data => …);
end if;
…
end P;
```

Anonymous object created?
Extra data copying performed?

Dynamic analysis (unit test)

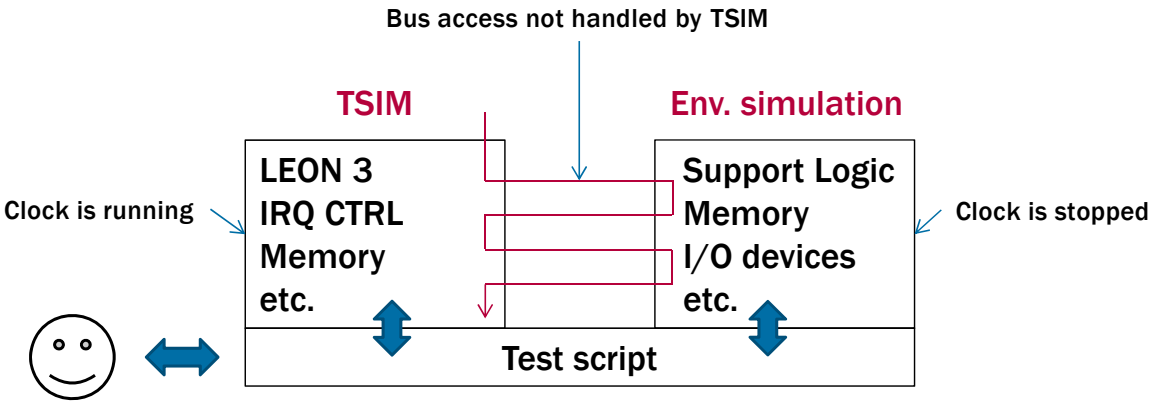


Unit testing is done on the development host machine using the LEON 3 emulator GNATemulator (based on QEMU):

- **Not timing accurate;**
- **Fast;**
- **Unit under test is not instrumented.**

Test coverage metrics obtained without instrumenting the unit under test!

Dynamic analysis (system test)

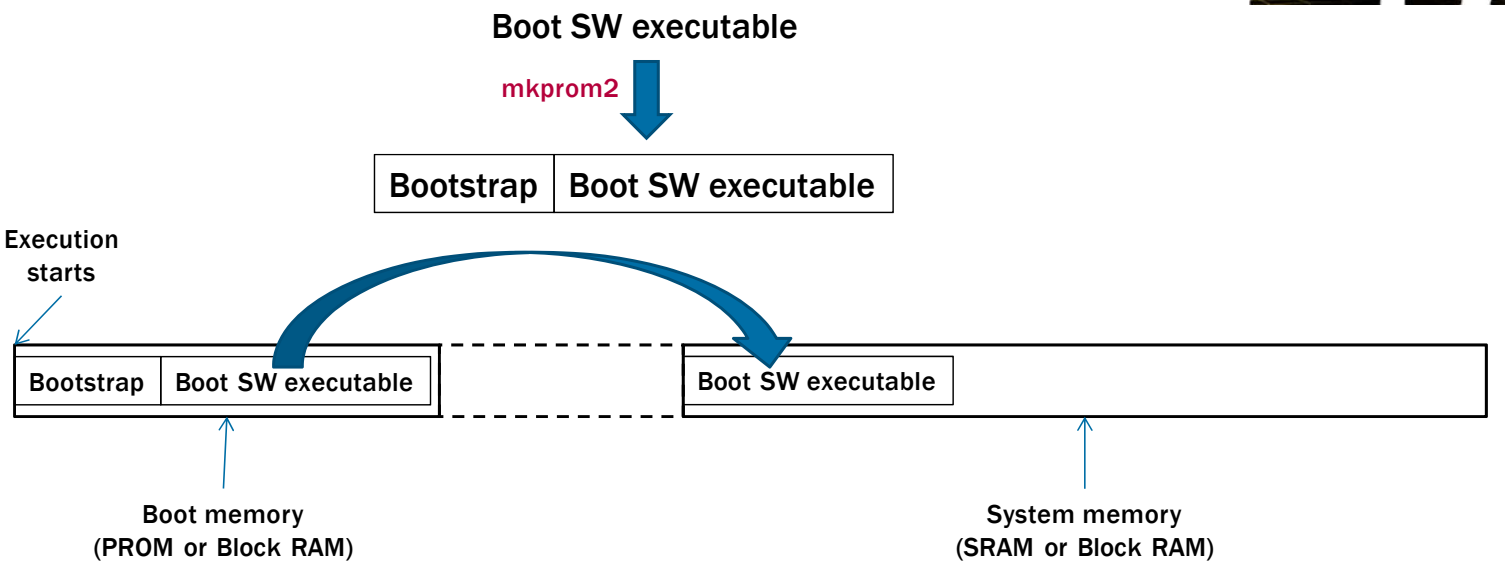


System testing is done on the development host machine using the LEON 3 simulator TSIM:

- **Timing accurate;**
- **Simulated environment;**
- **Complete control over the execution and environment.**

Fidelity of the environment simulation is only limited by the detail of implementation and hardware specification!

Implementation of Bootstrap



Boot SW and Application SW are ”regular” executables without any knowledge of basic board bring-up.

Programmer does not need to write any board bring-up code at all!

Conclusion

- **Component-based design:**
  - Automatic generation of concurrency view code;
  - Programmer fills in functional code.
- **Ravenscar-optimized bare-board run-time system.**
- **Ada 2012 aspects avoids defensive programming and eases obtaining test coverage.**
- **Low-level ”black art” bootstrap code is generated automatically.**
- **Static analysis:**
  - Coding standard checker;
  - Complexity and other metrics;
  - Stack size requirements (eliminating the hard-to-find ”blown stack” errors).
- **Dynamic analysis:**
  - Performed using emulators;
  - Test coverage metrics are obtained without instrumenting the code.

Some of the tools only support a de-facto implementation of a LEON 3 known as UT699 from Aeroflex. It is very important to keep the LEON3 in FPGA implementation close to the UT699!

A little secret: We have written two lines of assembler – one for putting the CPU into low power mode and one for jumping from Boot SW to Application SW.

Questions

?

Mark Lorenzen, Terma A/S, mal@terma.com