SimLEON : a LEON processor emulator

Claude Cazenave[1], Bernard Delatte[2]

[1] Astrium Satellites
31 rue des Cosmonautes , 31402 Toulouse, Cedex 4, FRANCE
Email: claude.cazenave@astrium.eads.net

[2] Centre National d'Etudes Spatiales (CNES)
18, Avenue Edouard Belin, 31401 Toulouse, Cedex 9, FRANCE
Email: bernard.delatte@cnes.fr

BACKGROUND

CNES and ASTRIUM have a long experience (over than 20 years) in emulator. Development of such software begins with the 1750 processor, then the ERC32 processor follows and now emulators of LEON processors are used.

1750

CNES started the development of a 1750 emulator in 1985 for the SPOT family, it is in reuse, since 1998, for the PROTEUS family.
ASTRIUM has developed a 1750 emulator in 1998 for the first E3000 (ASTRIUM Telecom satellite platform) operation simulator. When replacing hardware benches used for the software validation of METOP, an improved version of the emulator has been delivered. In effect, the emulator is currently used for 15 telecom spacecrafts and 2 earth observation spacecraft (Rocsat and Theos). In addition to operation simulator, this emulator is now used to support E3000 software validation (SVF) and to prepare the integration and test procedures before running on the real spacecraft (SIMAIT).

ERC32

In 2001/2002, CNES and ASTRIUM co-funded the development of an ERC32 emulator. Its name is SimERC32. It is now widely used in different simulators use cases:

- SVF: Software Validation Facility is a full numerical bench used to the validation of the flight software
- SIMAIT is a full numerical bench used to prepare AIT (Assembly, Integration and Test of the spacecraft) procedures to test the real spacecraft.
- SIMEFM : Electrical Functional Model is a hybrid bench : a numerical on board-computer with a hardware connection to a real equipment.
- Operation simulator is a full numerical bench used to train and help real spacecraft operators

CNES, ASTRIUM satellites, ASTRIUM ST, Dutch-Space and Airbus are currently users of SimErc32, through Pléiades, Galiléo, Bépi-Colombo, Vega launcher, Ariane5, Gaia and A350 projects.

DEVELOPMENT PROCESS

SimLEON development started in 2006 with the same partnership between CNES and ASTRIUM.

As the name should tell, SimERC32 is an ancestor of SimLEON. This is why SimLEON user interface is a reuse from SimERC32: users used to the ERC32 emulator will have no efforts to produce to adapt themselves to the LEON emulator. In addition, specification requirements are directly derived from SimERC32 requirements, nevertheless some additions have been made as watch point mechanism, cache simulation and configurability, and a timing accuracy requirement taken in account the new specificity of the processor. In the same time, the target platforms have been refined to standard PC with Linux or Windows.

On the one hand, the father of SimLEON, SimERC32 was the emulator of a "black-box" processor; the ERC32 chip was delivered in form of a physical chip and as a result was not modifiable. This is why different on board computer based on ERC32 can be modeled with the same emulator.
On the other hand, the LEON processor is much more versatile, because the VHDL source of the processor is available, and thus can be modified. Therefore each electronic manufacturer adapts the LEON VHDL to their particular needs. For instance, they can use a different memory controller or they can change the behavior of a timer. Thus, by contrast with the ERC32, different on board computer based on LEON, can't be modeled with the same emulator anymore. Though they will probably share a same core of functionality: an ADD instruction will probably do the same on different on board computer, even if a STORE instruction will triggers different mechanism.

As a consequence, SimLEON has been designed taken in account this flexible hardware design; the emulator is implemented in modules, which are responsible for a single function. For instance, you will find a module responsible of the modeling of UART, one for the timer… Each module implements an interface which is the same for all the different flavors of the module. They are fully exchangeable at compilation time. However the challenge behind this software flexibility is to keep a high level of performances for each flavor of each module.
In addition to this flexibility, each module offers a high level of configurability at compilation time. For instance, a set of property is used to configure, during the compilation phase, the memory module. With this approach it is easy to add a project specific memory area, without impacting the performances of the global product.

Another difference between the ERC32 and the LEON processors is the pipeline and cache mechanism. The ERC32 processor has a 4-stages instruction pipeline and no cache. The ERC32's pipeline is relatively simple and if an instruction has to take more than 1 cycle to execute, the entire pipeline is stopped and waits for this particular instruction. It is exactly what happens, when a load or store is issued on a slow memory. Otherwise, the LEON pipeline (5-stages for LEON2 and 7-stages for LEON3), is much more complex and the LEON processor comes with 2 caches mechanism, one for the data, and one for the instruction.
There is a tight relation between the cache and the pipeline. On modern memory, instructions are loaded in line; the instruction cache is responsible to deliver a line of instructions (usually 8 instructions) to the pipeline. If there is a miss on the instruction cache; instructions are not presents, the instructions are directly forwarded to the pipeline from the memory. It is a "streaming" mode. If there is a hit on the cache, the instructions are forwarded from the cache memory, and no access to the main memory is needed.
In effect, the data cache is responsible for the operations in case of load and store instruction. The bus for the main memory is shared between the instruction cache and the data cache. This is why the load/store operations are usually waiting for the instruction line loading to be finished before operating. And due to this interlock, the instruction line loading operation can wait for data operation to finish.

When the pipeline detects a special instruction, an instruction with more than 1 cycle of execution, the pipeline is stopped, but not in totality; already engaged instructions continue their execution. During this stop, the streaming from the memory to the pipeline is stopped too. Instructions continue to be forwarded from the memory in instruction cache (if enable) and when the instruction cache finishes a line loading, if a data operation was in a waiting state, it is executed.
In fact time management modeling is a challenge for a LEON emulator. The timing requirement asks 80% of accuracy on a global test. Current accuracy has been measured to 93% against a real board on a Stanford benchmark.

CURRENT STATUS & USERS

A validation of SimLEON has occurred on the leon2 flavor and the report is available to the users. Furthermore, the validation of the leon3 flavor is currently in progress and should be available for the end of the year.
The validation process involves the VHDL source of the LEON2/LEON3 processors which have been instrumented to produce traces on the current state of the pipeline, on internal registers values, and on timing management. The traces produce by the execution of the VHDL becomes the reference for the validation.
As a matter of fact, the emulator has a service which is used to produce traces too. A comparison of the two files (reference and emulator trace) is automatically achieved via a set of scripts. Thereby we are sure that the behavior of the emulator is accurate.

We have previously seen that the timing management was a challenging task for a LEON emulator, but it is challenging in term of performances too. The cache and pipeline models take the half of the CPU power needed for emulation. This is why with cache and pipeline mechanisms activated, a LEON2 40MHz emulation achieves 51% of real time on a standford benchmark. On the contrary, with an average timing system (cache and pipeline mechanisms deactivated) the emulation is at 91% of real time. Such performances are satisfying when considering other emulator performances, like SimERC32. Nevertheless, it is clearly not sufficient for a hybrid bench with hardware in the loop.

SimLEON has been already delivered, and is currently used on SVF (Software Validation Facility). The LEON2 flavor is used by the AlphaSat payload and a LEON3 flavor of SimLEON has been delivered for the AstroSat250 platform.

FUTURE WORK

Future of SimLEON will see an optimization stage. Local optimization based on known and well-proved techniques, as profiling and memory usage, will be conducted. But with such techniques, we can not expect more than 20% of improvement.

A step in performance will probably comes from a step in emulator technology. This is why a prototype of a dynamic binary translator has been developed and tested. The prototype had a limited scope: only 4 instructions; add, subcc, bicc and sethi emulated, but it allows us to run long tests

with a big number of instructions. As a matter of fact, this prototype has shown that this technology is promising as significant performance improvement (greater than 10 times) as been demonstrated with this study.

CONCLUSION

The SimLEON emulator is already operational and qualified; timing management requirement is met (93% of accuracy on a standford benchmark) and real-time performances allow an operation simulator.
Hardware in the loop benches force a performance improvement (20%), but in the mid term, a dynamic binary translator will add a huge acceleration factor, making us confident for the future of numerical simulator with on board computer build around LEON processor.