

# Hardware debug support in the LEON processor

Jiri Gaisler

Gaisler Research, Stora Nygatan 13, 41108 Göteborg, Sweden, tel. +46-31802405

*jiri@gaisler.com*

Andre-Louis Pouponnot

ESA/ESTEC, Postbus 299, 2200 AG Noordwijk, The Netherlands, tel. +31-715653685

*Andre-Louis.Pouponnot@esa.int*

## 1 Introduction

In 1997, the European Space Agency (ESA) completed the development of a 32-bit microprocessor for embedded space-flight applications, denoted ERC32. The ERC32 is currently being used in many space projects, including the control computers of the Russian segment of the International Space Station (DMS-R). To meet the mission requirements for projects beyond year 2000, the development of a more performant (+100 MIPS) processor denoted LEON was started in 1998. This paper presents the design goals and architecture of the LEON debug support unit, which provides unique capabilities for debugging and monitoring of complex applications.

## 2 Design goals

The purpose of the debug support unit is to aid software development and debugging on target hardware. The debug support unit must support the following functions:

- non-intrusive debugging and stepping through programs
- access to all registers, cache memories, peripheral units and external buses
- synchronisation with external software validation facility (SVF)

The LEON processor uses on-chip caches, and it is not possible to analyse the processor activity by monitoring the external memory bus (as was the case for ERC32). In LEON-based *system-on-chip* (SOC) devices, it is also not possible to monitor data traffic between the processor and on-chip I/O units. To add visibility on processor execution and data traffic, an on-chip trace buffer is needed. The trace buffer should allow the following operations:

- tracing executed instructions and data transfers
- tracing activity on the AMBA AHB bus

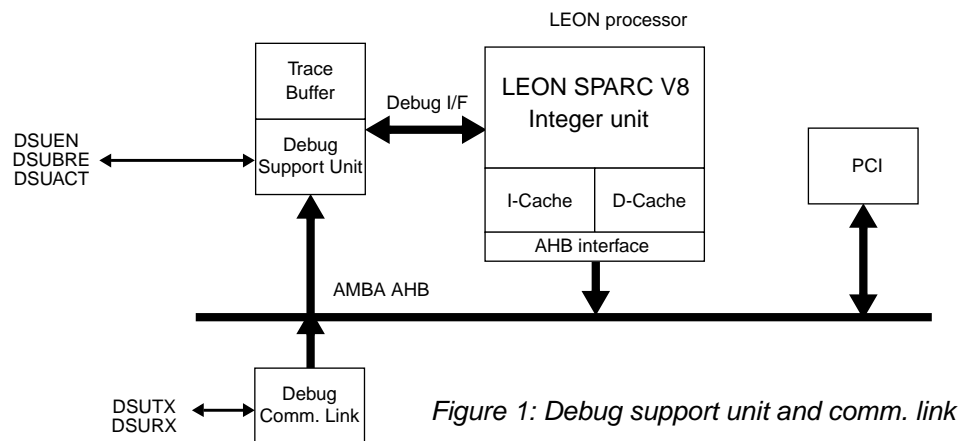
The trace should be stored in an on-chip memory block, and transferred to an external debugger on demand.

Considering that LEON will be used as *soft-core* in SOC devices, the debug support unit must be implemented in VHDL as part of the standard LEON VHDL model. Clock gating or JTAG interface should not be used since it complicates design portability and debugger interfacing.

### 3 Implementation

#### 3.1 Non-intrusive debugging

To allow non-intrusive source-level debugging on a hardware target, a debug mode has been added to the LEON processor pipeline. During the debug mode, the processor is halted and controlled by the debug support unit. A special interface allows the debug support unit to access all processor registers and cache ram contents. The debug support unit is connected as a slave to the LEON internal AMBA bus and can be accessed by any AHB master.



The processor can enter the debug mode on the following events:

- executing a breakpoint instruction (ta 1)
- hardware breakpoint/watchpoint hit
- asserting an external break signal
- receiving a break command from the debug support unit
- A trap that would cause the processor to enter error mode
- any or certain traps as programmed in the debug support unit

When the debug mode is entered, the following actions are taken:

- PC and nPC are saved in temporary registers (accessible by the debug unit)
- an output signal is asserted to indicate the debug state
- the LEON timer unit is (optionally) stopped to freeze the timers and watchdog

The debug mode is exited on a command from the debug support unit or by de-asserting an external signal. The timer unit will be re-enabled and execution will resume from the saved PC and nPC.

### **3.2 Communication link**

A dedicated communications link is provided to allow access from an external debug applications such as the GNU debugger (gdb). The link is implemented using a standard UART, identical to the two UARTs already provided in LEON. Auto-detect as well as software programmable baud-rate is supported. A simple read/write protocol is used, with block transfers support to increase the communication rate. The communication link is attached as a master to the AHB bus, and can thus access the full LEON address space, including memory, on-chip devices and PCI.

### **3.3 Trace buffer**

The trace buffer consists of a large circular buffer that can store an execution trace. The trace contains the following information:

- instruction address, opcode and result
- 30-bit time tag
- processor load/store address and data

The trace buffer is 128-bits wide with a configurable depth. A depth of 512 words is the baseline for the next LEON prototype. The trace buffer can also be used to trace AMBA AHB activity. In this mode, the trace will contain the following information:

- AHB address
- AHB data
- 30-bit time tag
- transfer parameters (AHB master, read/write, burst type, response...)

The trace buffer is frozen when the debug mode is entered or on a trace buffer breakpoint hit. To allow for an arbitrary trigger point, freezing the buffer can be delayed using a dedicated delay counter. Two separate AHB breakpoint registers are available to trigger on an arbitrary AHB or instruction address.

### **3.4 External interface**

The debug support unit uses a dedicated external interface consisting of the following signals:

- DSUEN            enable debug unit (input)
- DSUTX            UART transmit data (output)
- DSURX            UART receive data (input)
- DSUBRE           generate a break condition (input)
- DSUACT           processor is in debug mode (output)

Asserting the DSUBRE input at reset will force the processor to enter debug mode directly without executing any instructions. This will allow to download code and debug applications on systems without a boot-prom, or to program un-initialised flash-proms. The DSUACT signal can be used to synchronise with external software validation facilities.

## 4 Operation

### 4.1 Connection to debuggers

The protocol implemented by the communication link is simple, but not directly supported by any debugger. The communication between the debug support unit and an external debugger will therefore be done through a debug server, translating a debugger-specific protocol to read/write accesses on the LEON communication link.

With a debug server, it is possible to download and debug applications remotely, using a network and TCP/IP.

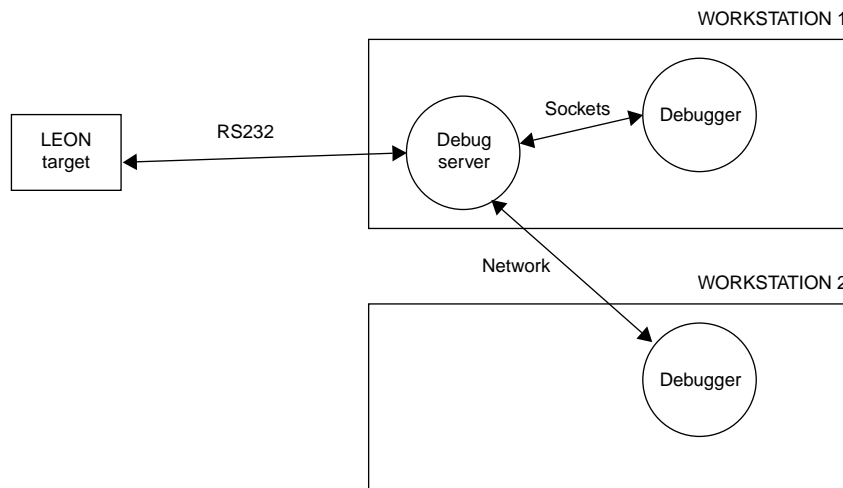


Figure 2: DSU <-> Debugger communication

The use of a standard UART for communications removes the need for (expensive) custom hardware and software interfaces, such as the JTAG used for TSC695E. In orbit, the debug communication link can be directly connected to the TM/TC sub-system, and debugging performed via the up- and downlink (!).

The separation of communication link and debug support unit also allows debugging to be performed using other interfaces. In the standard LEON device, the PCI interface is a master on the AHB bus and access to the debug support unit can be obtained by any PCI (master) unit. In custom SOC implementations, other I/O units could be used.

## **4.2 Breakpoints and watchpoints**

The current version of LEON contains four hardware breakpoints, all of which can be programmed to break on an instruction address, a data read address or a data write address. Each breakpoint has an associated mask which also allows marking of larger areas. When the debug support unit is enabled, a breakpoint hit will cause the processor to enter debug mode.

Additional breakpoints can be used by inserting the breakpoint instruction (ta 1) into the code. Even with software breakpoints it is possible to debug all parts of the program, also where traps are disabled. When a breakpoint trap is generated and traps are disabled, the processor would normally enter error. With the debug support unit enabled, the debug mode will be entered instead. and since the processor did not enter error mode, execution can be resumed (after the breakpoint is removed). In this manner, all hardware breakpoints can be saved for watchpoint use.

## **4.3 Trace buffer**

The trace buffer is normally not used during application debugging, but is an invaluable tool to perform post-mortem analysis. The trace buffer is typically left permanently running, tracing all executed instructions. An application crash and a following processor reset will turn the buffer off but preserve the contents, allowing it to be read out (and dumped to ground).

## **5 Conclusions**

The LEON debug support has been designed to allow simpler and more cost-effective debugging on target hardware. The separation of the debug support unit and communication link allows access to the debug functions from several interfaces. The use of a standard UART for communications removes the need for costly interface hardware and software, and allows debugging directly from any PC or workstation. The trace buffer adds visibility to instruction execution and data movement that otherwise would be impossible. The synchronous implementation in VHDL preserves portability and allows the debug functions to be used on any target technology, and in both ASIC and FPGA implementations.