Processor Functional Description

This hardware component is a lm32 ?(lattice mico32) processor core.

The lm32 is an open source soft core processor distributed by lattice semiconductors. It is intended to be used with lattice's FPGA but can be easily used with other targets as the RTL (verilog) code is available and the distribution license does not make special restrictions about this. For more information about RTL code licensing check lattice web site.

This ISS uses the ISS2 API and can be wrapped in a CABA or TLM-T Wrapper.

- It also supports the gdb server.

It implements all instructions defined in the Lattice Mico32 Processor Reference Manual.

Component definition & implementation

- source:trunk/soclib/soclib/iss/lm32/metadata/lm32.sd?
- source:trunk/soclib/soclib/iss/lm32/include/lm32.h?
- source:trunk/soclib/soclib/iss/lm32/src/lm32.cpp?
- source:trunk/soclib/soclib/iss/lm32/src/lm32_isa.cpp?
- source:trunk/soclib/soclib/iss/lm32/src/lm32_load_store.cpp?
- source:trunk/soclib/soclib/iss/lm32/src/lm32_debug.cpp?

Interrupts

LM32 architecture supports up to 32 external interrupt lines. This ISS implements all these interrupt lines by default.

Ports

None, it is to the wrapper to provide them.

Notes

MMU support

The lm32 provided by lattice does not have any mmu support.

Interrupt sensitivity

The ISS is sensitive to high level interrupts whereas the RTL version is sensitive to low level interrupts. This has been done to facilitate the integration with the rest of SoCLib modules.

Configuration of the ISS

The lm32 as provided by lattice is configurable, thus we could specify which functions are available (multiplier, divider..., # of irq...). For simplicity reasons, this ISS assumes that all the options are available (If somebody wants to make a configurable version using c++ templates for example, contributions are welcomed).

To have correct timing estimations we can modify gcc's specific options according to the desired physical implementation and modify the configuration register (r_CFG) of the ISS.

nb. The revision number of the ISS is set by default to 63.

The ISS has a template parameter "lEndianInterface" to specify whereas the lm32 acts externally as little endian (to conform with SocLib convention) or as big endian as the real CPU. This template must be specified when instantiating the lm32 ISS.

nb. If used in big endian configuration the gdb server wrapper will not behave correctly, and byte order will not be respected when reading/writing from/to memory in a gdb client.

Compiling programs for lm32 with SoCLib

A complete gcc toolchain (with a uClinux port) is available at ?Theobroma Systems website or at lattice web site.

The lm32 binutils, gdb and gcc are available in the respective mainstream distributions (starting from binutils 2.20, gdb 7.0 and gcc 4.5). Newlib libc also support lm32 cpu starting from version 1.17.0, libgloss specific files have to be modified to work with SocLib.

A non official ?freeRTOS port is also available ?here

Before compiling a program for the lm32 with the SoCLib framework you could modify some system variables (usually on the ~/.soclib/soft_compilers.conf) needed to find the good lm32 compiler and to define specific compilation and link options. Below you have an example:

```
lm32_CC_PREFIX = lm32-elf-
lm32_CFLAGS = -O2 -g -mmultiply-enabled -mdivide-enabled -msign-extend-enabled -mbarrel-shift-en
lm32_LDFLAGS = -nostdlib
```