Micro Blaze Processor Functional Description

This hardware component is a Micro Blaze processor core as described in [www.xilinx.com/ise/embedded/mb_ref_guide.pdf]. Note that the 9.2 version of ISE contains a pretty major evolution of the Micro Blaze that integrates a MMU, but this is not the version available within SoCLib (at least yet!). This component is an ISS, which should be wrapped with an IssWrapper for integration into a complete platform.

This instruction set simulator acts as a slave to the IssWrapper and is organised identically to the other Isses available within the library. Currently, the execution timings are pretty rough, and are typically one cycle per instruction.

The support for symetric and asymetric multiprocessing is hardwired using the `fsl` feature of the Micro Blaze. The processor number is given at instanciation time, and accessible through `get` on `fsl0`. Using other `fsl` will lead to an abort.

Component definition

Available in source:trunk/soclib/desc/soclib/microblaze.sd

Usage

Micro Blaze has no parameters.

```
Uses( 'microblaze')
```

Microblaze Processor ISS Implementation

The implementation is in

- source:trunk/soclib/systemc/include/common/iss/microblaze.h This defines the resources associated to the Micro Blaze along with a few minimal helper functions (or methods, as they call them)
- source:trunk/soclib/systemc/src/common/iss/microblaze.cc This is a large switch (as opposed to calling insn execution through pointers to functions) and a few macros, as it is overall not worse to traverse a switch than to move from tag to tag, seen the context necessary to the execution of one instruction (at least in the Micro Blaze case).

  It is possible to compile a version of the Micro Blaze that issues the instruction address along with the instruction being executed by defining `MBDEBUG` at `1` line 32 of source:trunk/soclib/systemc/src/common/iss/microblaze.cc This is quite useful to check that the processor is really interpreting correctly a sequence of instructions (and also for debugging software until a debugging stub is written).

Template parameters

This component has no template parameters.

Constructor parameters

```
MicroblazeIss(
     sc_module_name name,   //  Instance Name
     int  ident);   // processor id
```

Visible registers

None

Interrupts

There is a single interrupt line on the Micro Blaze, so all the handling is software based.

Ports

None, it is to the wrapper to provide them.