SocLib Components General Index

!!!! UNDER CONSTRUCTION !!!!

ST231 Processor Functional Description

The ST231 processor core is an hardware component implementing a 7-stage VLIW processor with register scoreboarding and 32-bit x 32-bit multiplies for integer and fractional data representations. Though a MMU was also added so the ST231 can be used as a host processor, this processor is mostly used in digital video consumer electronics.

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.

Component definition

Available in source:/trunk/soclib/soclib/lib/st231/metadata/st231.qd

Usage

Micro Blaze has no parameters.

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

Microblaze Processor ISS Implementation

The implementation is in

- source:trunk/soclib/soclib/lib/src/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/soclib/lib/src/iss/microblaze.cpp 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 35 of source:trunk/soclib/soclib/lib/src/iss/microblaze.cpp This is quite useful to check that the processor is really interpreting correctly a sequence of instructions. The Micro Blaze model is now able to connect to the GDB stub, so it is possible to use GDB for debugging software running on it.

Template parameters

This component has no template parameters.

Constructor parameters

```
MicroblazeIss(
     sc_module_name name,   //  Instance Name
```

```
            int   ident);    // processor id
```

Ports

The `IssWrapper` module is in charge of defining the communication ports.