SocLib Components General Index

UNDER DEVELOPMENT

ARM966 Processor Functional Description

This hardware component is a ARM966 processor core. This is only an ISS, which should be wrapped with an IssWrapper.

The simulation model is actually an instruction set simulator with an ARM966 pipeline.

Currently it only exists in bigendian form.

IMPORTANT: steps to apply before using the ARM966

Before compiling any SoClib simulator using the ARM966 you will need to download the UNISIM (?http://www.unisim.org) library (well, just a piece of it, the unisim_lib).

To do so just download it using svn from ?https://unisim.org/svn/devel/unisim_lib with the following command:

- svn import ?https://unisim.org/svn/devel/unisim_lib

You will have to enter a username and password. If you do not have access to the UNISIM development, you can simply use 'guest' for username (no password required). Once you have downloaded UNISIM you will need to create a link in trunk/soclib/lib/arm966/include/iss/ and trunk/soclib/lib/arm966/src/iss/ to <your_path_to_unisim_lib>/unisim.

If you wish you can download the full UNISIM library by downloading unisim_tools and unisim_simulators:

- svn import ?https://unisim.org/svn/devel/unisim_tools
- svn import ?https://unisim.org/svn/devel/unisim_simulators

Finally you will have to set your soclib.conf (source:trunk/soclib/utils/conf/soclib.conf) file to compile correctly the ARM966 component. Here you have an example of configuration that correctly sets the flags to compile ARM966:

```
# -*- python -*-

def _platform():
        """
        Retrieves platform information and make it look-like systemc's
        lib-xxx thing.

        Working so far with:
         * linux
         * darwin
        """
        import sys
        pf = sys.platform
        # Strip numeric suffix from platform name
        while pf[-1] in "0123456789":
                pf = pf[:-1]

        remap_pf = {'darwin':'macosx'}
        if pf in remap_pf:
```

```
                pf = remap_pf[pf]
        return pf

config.systemc = Config(
        base = config.systemc,
        dir = "${SYSTEMC}",
        os = _platform(),
        )

config.my_toolchain = Config(
        base = config.toolchain,
        cflags = ['-ggdb', '-DSOCLIB', '-D__STDC_CONSTANT_MACROS', '-Wall', '-Wno-pmf-conversion
        )

config.default = Config(
        base = config.build_env,
        systemc = config.systemc,
        toolchain = config.my_toolchain,
        repos = "/tmp/build/sc",
        )
```

The flags you will need to compile the ARM966 component are: -DSOCLID and -D__STDC_CONSTANT_MACROS. In the previous example you can see that the default toolchain has been augmented to define those flags.

Component definition

Available in source:trunk/soclib/soclib/lib/arm966/metadata/arm966.sd

Usage

ARM966 has no parameters.

```
Uses('iss_wrapper', iss_t = 'common:arm966')
```

ARM966 Processor ISS Implementation

The implementation is in

- source:trunk/soclib/lib/arm966/include/iss/arm966.h
- source:trunk/soclib/lib/arm966/src/iss/arm966.cpp

The previous files use the ARM966 implementation provided in the UNISIM library.

Template parameters

This component has no template parameters.

Constructor parameters

```
ARM966Iss(
     sc_module_name name,   //  Instance Name
     int  ident);   // processor id
```

Visible registers

UNDER DEVELOPMENT

Interrupts

UNDER DEVELOPMENT

The handling and prioritization of the interrupts is deferred to software.

Ports

None, it is to the wrapper to provide them.

Compiling programs for ARM966 with SoClib

Before compiling a program for the ARM966 with the SoClib framework you will need to define some system variables (usually on the ~/.soclib/soft_compilers.conf) needed to find the ARM compiler. Below you have an example:

```
arm966_CC_PREFIX = armv5b-softfloat-linux-
arm966_CFLAGS = -nostdinc -gstabs+
arm966_LDFLAGS = -nostdlib
```