

## UNDER DEVELOPMENT

# ARM7TDMI Processor Functional Description

This hardware component is a ARM7TDMI 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 ARM7TDMI pipeline.

Currently it only exists in bigendian form.

## Component definition

Available in source:trunk/soclib/soclib/lib/arm7tdmi/metadata/arm7tdmi.sd

## Usage

ARM7TDMI has no parameters.

```
Uses('iss_wrapper', iss_t = 'common:arm7tdmi')
```

Before compiling any SoClib simulator using the ARM7TDMI 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](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'/'guest' for username and password respectively. Once you have downloaded UNISIM you will need to create a link in trunk/soclib/lib/arm7tdmi/include/iss/ and trunk/soclib/lib/arm7tdmi/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`

## ARM7TDMI Processor ISS Implementation

The implementation is in

- source:trunk/soclib/lib/arm7tdmi/include/iss/arm7tdmi.h
- source:trunk/soclib/lib/arm7tdmi/src/iss/arm7tdmi.cpp

The previous files use the ARM7TDMI implementation provided in the UNISIM library.

## Template parameters

This component has no template parameters.

## Constructor parameters

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

or

```
MipsEbIss( name, ident);
```

## Visible registers

The following internal registers define the processor internal state, and can be inspected:

- `r_pc` : Program counter
- `m_ins` : Instruction register
- `r_gpr[i]` : General registers (  $0 < i < 32$  )
- `r_hi` & `r_lo` : Intermediate registers for multiply / divide instructions
- `r_cp0[i]` : Coprocessor 0 registers ( $0 \leq i < 32$ ). Implemented values:
  - ◆ 8: BAR : Bad address register
  - ◆ 12: SR : Status register
  - ◆ 13: CR : Cause register
  - ◆ 14: EPC : Exception PC register
  - ◆ 15: INFOS : CPU identification number on bits [9:0]

## Interrupts

Mips defines 6 interrupts lines. ~~The lowest number has the highest priority.~~ The handling and prioritization of the interrupts is deferred to software.

## Ports

None, it is to the wrapper to provide them.