VciMultiTty

Functional Description

This VCI target is a TTY terminal controller. This hardware component controls one or several independant terminals. The number of emulated terminals is defined by the arguments in the constructor (one name per terminal).

Each terminal is acting both as a character display, and a keyboard interface. For each terminal, a specific IRQ is activated when a character entered at the keyboard is available in a buffer. IRQ is kept low as long as the buffer is not empty.

This hardware component cheks for segmentation violation, and can be used as a default target.

This component uses a <u>TtyWrapper</u> per terminal in order to abstract the simulated ttys. The terminal index i is defined by the ADDRESS[12:4] bits.

Each TTY controller contains 3 memory mapped registers:

```
• TTY_WRITE
```

This 8 bits pseudo-register is write only. Any write request will interpret the 8 LSB bits of the WDATA field as an ASCII character, and this character will be displayed on the addressed terminal.

```
• TTY_STATUS
```

This Boolean status register is read-only. A read request returns the zero value if there is no pending character. It returns a non zero value if there is a pending character in the keyboard buffer.

```
• TTY_READ
```

This 8 bits register contains one single ASCII character. This register is read-only. A read request returns the ACSII character in the 8 LSB bits of the RDATA field, and reset the status register.

For extensibility issues, you should access your terminal using globally-defined offsets.

You should include file soclib/tty.h in your software, it defines TTY_WRITE, TTY_STATUS, TTY_READ and TTY_SPAN.

A putc/getc implementation could be:

```
#include "soclib/tty.h"

static const volatile void* tty_address = 0xc0000000;

static inline void putc(const size_t term_no, const char x)
{
    soclib_io_set( tty_address, term_no*TTY_SPAN + TTY_WRITE, x );
}

static inline char getc(const size_t term_no)
{
    return soclib_io_get( tty_address, term_no*TTY_SPAN + TTY_READ );
```

VciMultiTty 1

}

(add -I/path/to/soclib/include to your compilation command-line)

Component definition

source:trunk/soclib/soclib/module/connectivity component/vci multi tty/caba/metadata/vci multi tty.sd?

Usage

See SoclibCc/VciParameters

```
Uses( 'vci_multi_tty', **vci_parameters )
```

CABA Implementation

CABA sources

- interface:
 - source:trunk/soclib/soclib/module/connectivity component/vci multi tty/caba/source/include/vci multi tty.h?
- implementation :

source:trunk/soclib/soclib/module/connectivity component/vci multi tty/caba/source/src/vci multi tty.cpp?

CABA Constructor parameters

Note: the name list MUST be terminated by NULL. Example instanciation:

```
VciMultiTty tty("tty_comp", IntTab(2,3), mapping_table, "term0", "term1", "term2", NULL);
```

CABA Ports

- sc_in<bool> p_resetn : Global system reset
- sc in<bool> p clk : Global system clock
- soclib::common::VciiTarget<vci_param> p_vci : The VCI port
- sc_out<bool> p_irq[] : Interrupt ports array.

TLM-T Implementation

The TLM-T implementation is not yet available.