VciMultiTty Functional Description

This VCI target is a TTY terminal controller. This hardware component controls up to 256 terminals, emulated as XTERM windows. The number of emulated terminals is defined by the arguments in the constructor. The constructor creates as many UNIX XTERM processes as the number N of emulated terminals. It creates N PTY pseudo-terminals (one for each XTERM process), to support the bidirectional communication between the TTY controller process and the XTERM processes. Each terminal is acting both as a character display, and a keyboard interface. For each terminal, a specific IRQ is activated when a character is entered at the keyboard. The terminal index i is defined by the ADDRESS[12:4] bits. This hardware component cheks for segmentation violation, and can be used as a default target.

Each TTY controller contains 3 memory mapped registers:

- **TTY_DISPLAY** : ADDRESS[3:0] = 0x0

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_KEY_STS** : ADDRESS[3:0] = 0x4

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_KEY_BUF** : ADDRESS[3:0] = 0x8

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

VciMultiTty CABA Implementation

The caba implementation is in

- source:trunk/soclib/systemc/include/caba/target/vci_multi_ram.h
- source:trunk/soclib/systemc/src/caba/target/vci_multi_ram.cc

Template parameters:

- The VCI parameters

Constructor parameters

```
VciMultiTty(
     sc_module_name name,   // Instance name
     const soclib::common::IntTab &index,   //  Target index
     const soclib::common::MappingTable &mt,   // Mapping Table
     const char *first_tty_name,   // TTY names (as many names as terminals)
     ...);
```

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** : Pointer on the interrupt ports array.