VciPCI

1) Functional Description

This VCI target is a memory mapped peripheral that serves as a bridge with the PCI bus.

The memory segment allocated to this component must be aligned on 4K bytes boundary.

This VCI target contains 5 memory mapped registers:

- PCI_VALUE

This 32 bits register corresponds to two 4 KB FIFO, one to write the data (FIFOW) sent to the PCI bus and one to read (FIFOR) the data received from the PCI bus. A write in a full FIFO lost the data and a read in an ampty FIFO returns 0!

- PCI_MODE

This register describes the DMA mode:

o PCI_DMA_WRITE_IRQ = DMA operated from the PCI bus to the FIFOR with an IRQ at the end of the transfer.

o PCI_DMA_WRITE_NO_IRQ = DMA operated from the PCI bus to the FIFOR with no IRQ at the end of the transfer.

o PCI_DMA_READ_IRQ = DMA operated from the FIFOW to the PCI bus with an IRQ at the end of the transfer.

o PCI_DMA_READ_NO_IRQ = DMA operated from FIFOW to the PCI bus with no IRQ at the end of the transfer.

o PCI_DMA_LOOPBACK = DMA operated automatically from the FIFOR when data are available, to the PCI bus with no IRQ at the end of the transfer. With this mode you can send data to a PCI target configured in loopback which send it back with default PCI physical address 23 (decimal).

- PCI_ADR

This 32 bits register defines the physical address on the PCI bus where to send/receive data. It must be equal to the BASE ADRESS (BADR0) of one PCI target.

- PCI_RESETIRQ

Any write request in this Boolean register will reset the pending IRQ. A read request returns the zero value when there is no pending interrupt, and returns a non zero value if there is a pending interrupt.

- PCI_NB

This 32 bits register defines the size of the DMA transfer in bytes (from 0 Byte to 4 Kbyte) and launches the DMA if FIFO are ready.

2) Component definition & usage

You should include file soclib/pci.h from your software, it defines PCI_VALUE, PCI_MODE, PCI_ADR, PCI_RESETIRQ.

Sample code:

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

static const volatile void* pci_address = 0xc0000000;

static pci_test()
{
 soclib_io_set(pci_address,PCI_NB,0);  // size of transfer = 0 first to avoid DMA launch
 soclib_io_set(pci_address,PCI_ADR,23);  // PCI physical address of the target
 soclib_io_set(pci_address,PCI_VALUE,8);  // 4 data values written in the FIFO  (8,12,14,16)
 soclib_io_set(pci_address,PCI_VALUE,12);
 soclib_io_set(pci_address,PCI_VALUE,14);
 soclib_io_set(pci_address,PCI_VALUE,16);
 soclib_io_set( pci_address,PCI_MODE,PCI_DMA_READ_NO_IRQ ); // DMA mode
 soclib_io_set(pci_address,PCI_NB,16);  // size in bytes of the transfer (4 32-bit words): the D
}
```

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

source:trunk/soclib/soclib/module/internal_component/vci_pci/caba/metadata/vci_pci.sd?

See SoclibCc/VciParameters

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

3) CABA Implementation

CABA sources

- interface : source:trunk/soclib/soclib/module/internal_component/vci_pci/caba/source/include/vci_pci.h?
- implementation : source:trunk/soclib/soclib/module/internal_component/vci_pci/caba/source/src/vci_pci.cpp?

CABA Constructor parameters

```
VciPci(
    sc_module_name name,   //  Component Name
    const soclib::common::IntTab & index,  // Target index
    const soclib::common::MappingTable &mt,   // MappingTable
    size_t badrval);   //  BASE Address register init value (BADR0)
```

CABA Ports

- sc_in<bool> p_resetn : Global system reset
- sc_in<bool> p_clk : Global system clock
- soclib::caba::VciTarget<vci_param> p_vci : The VCI port

- sc_out<bool> p_irq : Interrupts port
- sc_inout<sc_lv<4> > p_Cbe : Command/Bytes Enable PCI port
- sc_in_clk p_clkpci : PCI clock
- sc_in<bool> p_Sysrst : PCI reset
- sc_in<bool> p_Idsel : PCI chip select
- sc_inout<sc_logic> p_Frame : PCI Frame
- sc_inout<sc_logic> p_Devsel : PCI Device Select
- sc_inout<sc_logic> p_Irdy : PCI Initiator Ready
- sc_in<bool> p_Gnt : PCI Grant
- sc_inout<sc_logic> p_Trdy : PCI Target Ready
- sc_inout<sc_logic> p_Inta : PCI Interrupt
- sc_inout<sc_logic> p_Stop : PCI Stop
- sc_out<bool> p_Req : PCI request
- sc_inout<sc_logic> p_Par : PCI Parity
- sc_inout<sc_lv<32> > p_AD32 : PCI main bus (32 bits) for address and data

4) TLM-T Implementation

The TLM-T implementation is not available.