VciLocalCrossbar

1) Functional Description

This hardware component is a VCI compliant full crossbar, it contains two independant crossbars for VCI commands and VCI responses. It must only be used in clusterized architecture, to interconnect a limited number of VCI initiators and targets in a local sub-system. The associated sub-system is identified by a global index.

- The number of VCI initiators is a parameter that should not be larger than 4.
- The number of VCI targets is a parameter that should not be larger than 4.

The VciLocalCrossbar component has a dedicated VCI interface (both initiator and target) to connect the local subsystem to the global VCI interconnect.

When several initiators try to reach the same target, the arbitration policy is round-robin.

As any VCI advanced compliant interconnect, this component uses the MSB bits of the VCI ADDRESS field to route the command packets to the proper target, thanks to a routing table, implemented as a ROM. This routing table is build by the constructor from the informations stored in the mapping table. It uses the VCI RSRCID field to route the response packet to the initiator.

2) Component definition & usage

source:trunk/soclib/module/network_component/vci_local_crossbar/caba/metadata/vci_local_crossbar.sd

See SoclibCc/VciParameters

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

3) CABA Implementation

CABA sources

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

CABA Constructor parameters

```
VciLocalCrossbar(
     sc_module_name name,  //  instance name
     const soclib::common::MappingTable &mt,  // mapping table
     const soclib::common::IntTab &initiator_index,  //  global initiator index
     const soclib::common::IntTab &target_index,  //  global target index
     size_t nb_initiator,  //  number of VCI initiators
     size_t nb_target ); // number of VCI targets
```

CABA Ports

- sc_in<bool> **p_resetn** : Global system reset
- sc_in<bool> **p_clk** : Global system clock
- soclib::caba::VciTarget<vci_param> **p_from_initiator[]** : Ports from VCI initiators
- soclib::caba::VciInitiator<vci_param> **p_to_target[]** : Ports to VCI targets
- soclib::caba::VciTarget<vci_param> **p_initiator_to_up** : Initiator VCI port to micro-network
- soclib::caba::VciInitiator<vci_param> **p_target_to_up** : Target VCI port to micro-network

4) TLM-DT implementation

TLM-DT sources

- interface : source:trunk/soclib/soclib/module/network_component/vci_local_crossbar/tlmdt/source/include/vci_local_crossbar.h
- implementation : source:trunk/soclib/soclib/module/network_component/vci_local_crossbar/tlmdt/source/src/vci_local_crossbar.cpp?

TLM-DT Constructor parameters

```
VciLocalCrossbar(
     sc_module_name name,  //  instance name
     const soclib::common::MappingTable &mt,  // mapping table
     const soclib::common::IntTab &index,  //  global index
     size_t nb_initiator,  //  number of VCI initiators
     size_t nb_target ); // number of VCI targets
```

TLM-DT Ports

- std::vector<tlm_utils::simple_initiator_socket_tagged<VciLocalCrossbar,32,tlm::tlm_base_protocol_types> *> p_vci_initiator; *VCI initiator ports*
- std::vector<tlm_utils::simple_target_socket_tagged<VciLocalCrossbar,32,tlm::tlm_base_protocol_types> *> p_vci_target; *VCI target ports*