VciAvalonBus

1) Functional Description

This hardware component is a generic Avalon switch fabric allowing the interconnection of *Nb_Master* Avalon masters and *Nb_Slave* Avalon slaves. The master-to-slave relationship can be one-to-one, one-to-many, many-to-one, or many-to-many. Masters and slaves have the same data and address widths and operate in the same clock domain.

VCI-Avalon wrappers do not require to support full Avalon features, so not all Avalon slave and master ports are supported (AvalonSwitchMaster?, AvalonSwitchSlave?).

Implemented functionnalities :

- fundamental read, fundamental write with variable wait-state
- burst transfer
- flow control (dataavailable)
- round robin arbitration

Unimplemented functionnalities :

- wait state insertion
- pipelined read transfers
- tristate transfert
- setup and hold time
- dynamic bus sizing
- interrupt requests

An AVALON bus instanciates the three following components :

Address decoding logic (ADL) (AvalonAddressDecodingLogic) in the system interconnect fabric distributes an appropriate address and produces a chipselect signal for each slave.

Datapath multiplexing (MUX) (AvalonMux) in the system interconnect fabric drives the *writedata* signal from the granted master to the selected slave, and the *readdata* signal from the selected slave back to the requesting master.

Multiple Avalon masters can simultaneously perform transfers with independent slaves. The system interconnect fabric provides shared access to slaves using a technique called slave-side arbitration. Slave-side arbitration moves the arbitration logic (Arbiter) (AvalonArbiter) close to the slave, such that the algorithm that determines which master gains access to a specific slave in the event that multiple masters attempt to access the same slave at the same time. The arbiter grants shares in a round-robin order.

AvalonSwitchConfig describes the implemented switch fabric.

2) Component definition & usage

3) CABA Implementation

CABA sources

- interface :
  source:trunk/soclib/soclib/module/network_component/avalon_switch_fabric/caba/source/include/avalon_switch_fab
- implementation :
  source:trunk/soclib/soclib/module/network_component/avalon_switch_fabric/caba/source/src/avalon_switch_fabric.c

CABA Constructor parameters

```
AvalonSwitchFabric<Nb_Master, Nb_Slave, avalon_param>  SwitchFabric("SwitchFabric", config_swit
```

CABA Ports

- sc_in<bool> **p_resetn** : Global system reset
- sc_in<bool> **p_clk** : Global system clock
- AvalonSwitch_Master ***p_avalon_master**: Nb_Master ports from Avalon masters
- AvalonSwitch_Slave ***p_avalon_slave**: Nb_Slave ports to Avalon slaves

CABA Implementation Notes

The configuration of the switch fabric is platform dependant. The AvalonSwitchConfig component is used for this purpose.

```
AvalonSwitchConfig<nb_master, nb_slave> config_switch;
```

where *Nb_Master, Nb_slave* : defined in the platform description (**top.cpp** file).

For each master the routing table SwitchFabricParam_Master[0]->route[] describes the connection between this given master and the slaves. SwitchFabricParam_Master[]->mux_n_slave is the number of slaves connected to this master (number of MUX inputs).

For each slave the routing table SwitchFabricParam_Slave[0]->route[] describes the connection between this slave and the masters. SwitchFabricParam_Slave[]->arbiter_n_master is the number of masters connected to this slave (number of Arbiter inputs). SwitchFabricParam_Slave[]->Base_Address, SwitchFabricParam_Slave[]->Address_Span is the addressing space of this slave (decoded in ADL)

4) TLM-T implementation

There is no TLM-T implementation for this component.