

Demapping

1) Functional Description

The demapping operation consists in demodulating the inputs symbols into bits. In our system, the used demapping is a QPSK (Quadrature Phase-Shift Keying) demodulation. The architecture of the demapping component is presented in the figure 1. It is composed of a demapping core and a MWMR wrapper. The wrapper is used to interface the core and the MWMR controller available here [VciMwmrController](#).



3) CABA Implementation

a) Component definition & usage

Component definition

- [source:trunk/soclib/soclib/module/ofdm_chain_components/demapping/caba/metadata/demapping.sd?](#)

Usage

Demapping has a *fifo_depth* parameter, which defines the fifo depth for the input. For example with a FIFO depth equal to 16 :

```
Uses('Demapping', fifo_depth = 16);
```

b) CABA sources

- interface :
[source:trunk/soclib/soclib/module/ofdm_chain_components/demapping/caba/source/include/demapping.h?](#)
- implementation :
[source:trunk/soclib/soclib/module/ofdm_chain_components/demapping/caba/source/src/demapping.cpp?](#)

CABA Constructor parameters

```
Demapping(  
    sc_module_name name, // Instance name  
    int ncycles) // Number of computation cycles
```

CABA Ports

- `sc_in<bool> p_resetn` : hardware reset
- `sc_in<bool> p_clk` : clock
- `soclib::caba::FifoOutput<uint32_t> p_to_ctrl` : interface from the demapping to the MWMR controller
- `soclib::caba::FifoInput<uint32_t> p_from_ctrl` : interface from the MWMR controller to the demapping

3) TLM-DT Implementation

a) Component definition & usage

Component definition

- source:trunk/soclib/soclib/module/ofdm_chain_components/demapping/tlmdt/metadata/demapping.sd?

b) TLM-DT sources

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

TLM-DT Constructor parameters

```
Demapping(sc_core::sc_module_name name, // Instance name
          uint32_t id,
          uint32_t read_fifo_depth, // Depth of input buffer
          uint32_t write_fifo_depth, // Depth of output buffer
          uint32_t n_read_channels, // Number of read channels
          uint32_t n_write_channels, // Number of write channels
          uint32_t n_config, // Number of configurations
          uint32_t n_status); // Number of status
```

TLM-DT Ports

- `std::vector<tlm_utils::simple_target_socket_tagged<Demapping,32,tlm::tlm_base_protocol_types>*>`
p_config: configuration port
- `std::vector<tlm_utils::simple_target_socket_tagged<Demapping,32,tlm::tlm_base_protocol_types>*>`
p_status: status port
- `std::vector<tlm_utils::simple_initiator_socket_tagged<Demapping,32,tlm::tlm_base_protocol_types>*>`
p_read_fifo: port from the MWMR controller to the demapping
- `std::vector<tlm_utils::simple_initiator_socket_tagged<Demapping,32,tlm::tlm_base_protocol_types>*>`
p_write_fifo: port from the demapping to the MWMR controller