Netherlands Institute for Radio Astronomy

RadioNET FP7: Modular design and module reuse: the ETH module as an example

UniBoard Face-to-face Meeting, Bordeaux, 12-13 October 2010

Eric Kooistra

Contents

Why modular design and module reuse?

- The aim is to have plug and play modules to speed up FPGA firmware development and make it easier.
- This is necessary to be able to use the latest FPGA technology for our designs (note Stratix V is already there).
- Casual reuse (meaning copy paste from existing code) also speeds up development, but less than module reuse.

Module reuse does not come for free

- The initial development time increases, because it involves:
  - keeping general usage in mind while designing
  - thorough testing in simulation and on target
  - providing a reference design
  - complete set of source files (HDL, C, project, scripts, …)
  - proper coding style (even though the user may not see this)
  - proper documentation

What is a VHDL module?

- The definition is not strict, but typically a design consists of modules and a module consists of components.

1. Design: top level entity that can run on the FPGA
2. Module: a more elaborate function or a group of related low level functions
3. Component: a low level function

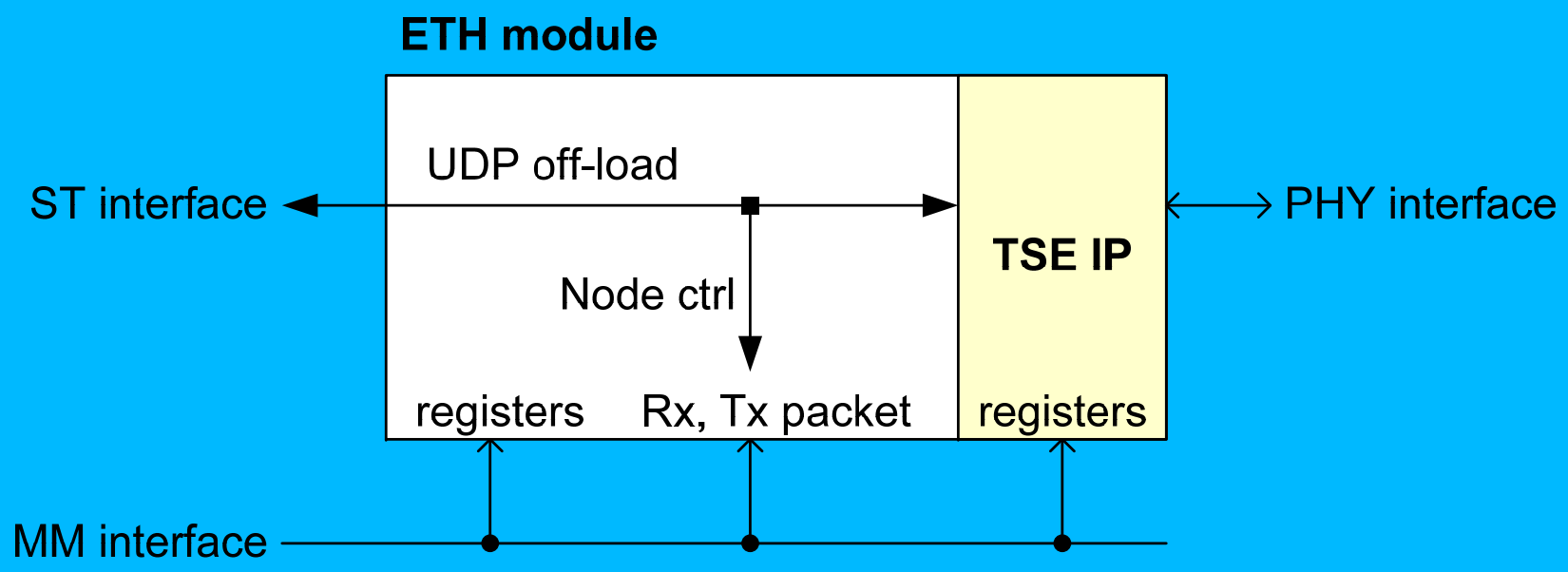
- The designs are kept in $UNB/designs
  The modules are kept in $UNB/modules

  (UNB = https://svn.astron.nl/UniBoard_FP7/UniBoard/trunk/Firmware/)

Available modules in $UNB/modules/

- COMMON → e.g. counter, memory, FIFO
- UNB_COMMON → UniBoard auxiliary
- LOFAR/I2C → I2C master
- LOFAR/MDIO → MDIO master
- LOFAR/DIAG → test sequence generator
- DP(1) → packetizing data and de-packetizing data
- DP(2) → streaming components, e.g. mux, latency adapter
- TR_NONBONDED → giga bit transceivers
- ETH → 1GbE

Top level of the ETH module



- The ETH module is kept at $UNB/modules/tse
- Purpose is to:
  - Minimize the processing load for the microprocessor
  - Provide a Memory-Mapped (MM) interface to the TSE MAC IP
  - Provide a Streaming (ST) interface to off-load UDP frames

Standard MM and ST interfaces

MM = Memory-Mapped

Master Slave

control

data

ST = Streaming

Source Sink

control

data

Master Slave

mas_out —*mosi*→ *sla_in*

mas_in ←*miso*— *sla_out*

Source Sink

src_out —*sosi*→ *snk_in*

src_in ←*siso*— *snk_out*

MM and ST interface record examples

- MM interface:

```
TYPE t_eth_reg_mm_bus IS RECORD
  -- Master In Slave Out (MISO)
  rddata     : STD_LOGIC_VECTOR(c_eth_data_w-1 DOWNTO 0);
  -- Master Out Slave In (MOSI)
  address    : STD_LOGIC_VECTOR(c_eth_reg_addr_w-1 DOWNTO 0);
  wrdata     : STD_LOGIC_VECTOR(c_eth_data_w-1 DOWNTO 0);
  wr         : STD_LOGIC;
  rd         : STD_LOGIC;
END RECORD;
```

- ST interface:

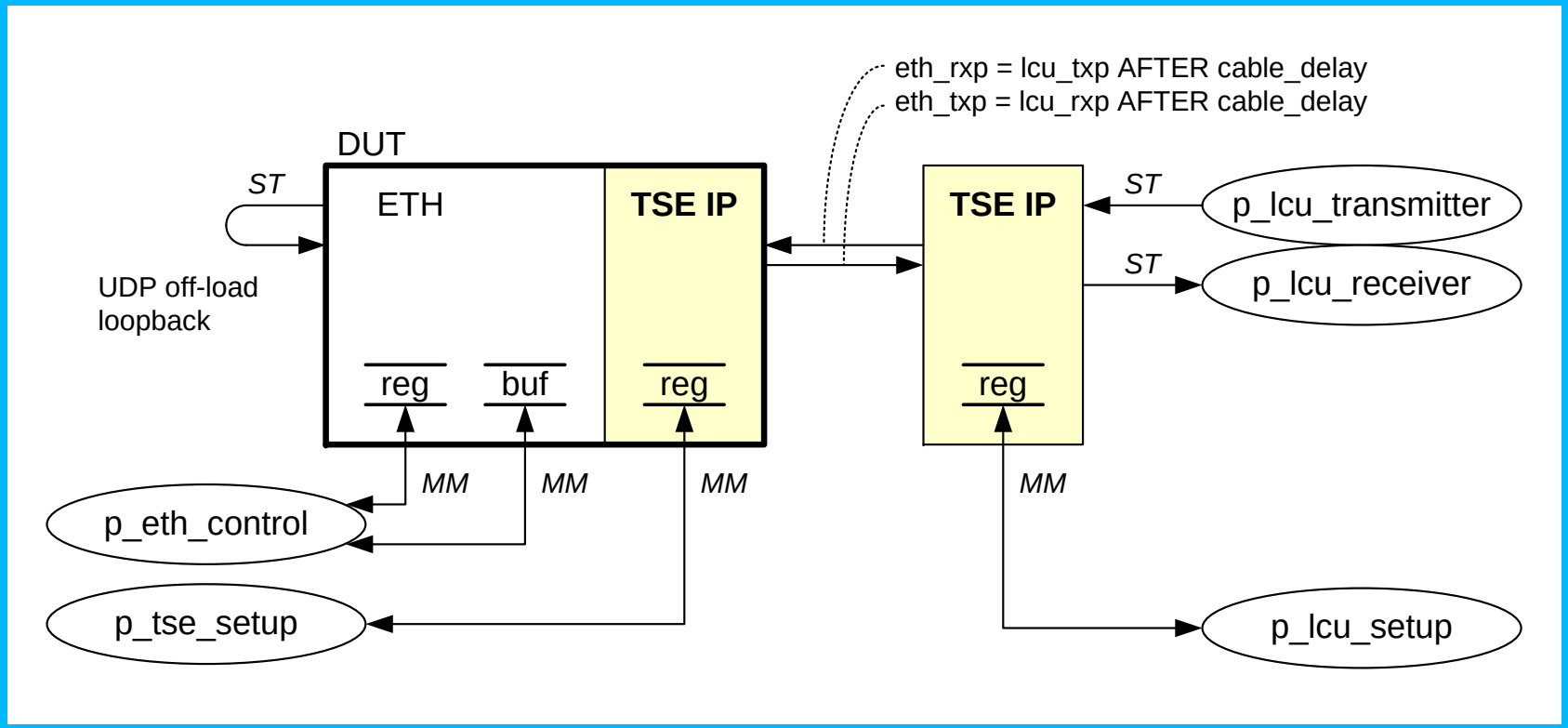
```
TYPE t_eth_udp_stream IS RECORD
  -- Source In or Sink Out (SISO)
  ready     : STD_LOGIC;
  -- Source Out or Sink In (SOSI)
  data      : STD_LOGIC_VECTOR(c_eth_data_w-1 DOWNTO 0);
  valid     : STD_LOGIC;
  sop       : STD_LOGIC;
  eop       : STD_LOGIC;
  empty     : STD_LOGIC_VECTOR(c_eth_empty_w-1 DOWNTO 0);
  channel   : STD_LOGIC_VECTOR(c_eth_demux_channel_w-1 DOWNTO 0);
END RECORD;
```

VHDL wrapper to encapsulate IP

- Purpose of a VHDL wrapper is to have a central file via which the IP is instantiated in our designs.
- Advantages of using a wrapper are:
  - Ensures that all instances use the vendor IP in the same way
  - Clearly isolates vendor IP from our own generic VHDL
  - Eases porting to other vendor IP should this be necessary
  - The wrapper may also contain some extra (glue) logic
  - Corrections in the wrapper automatically effect all instances

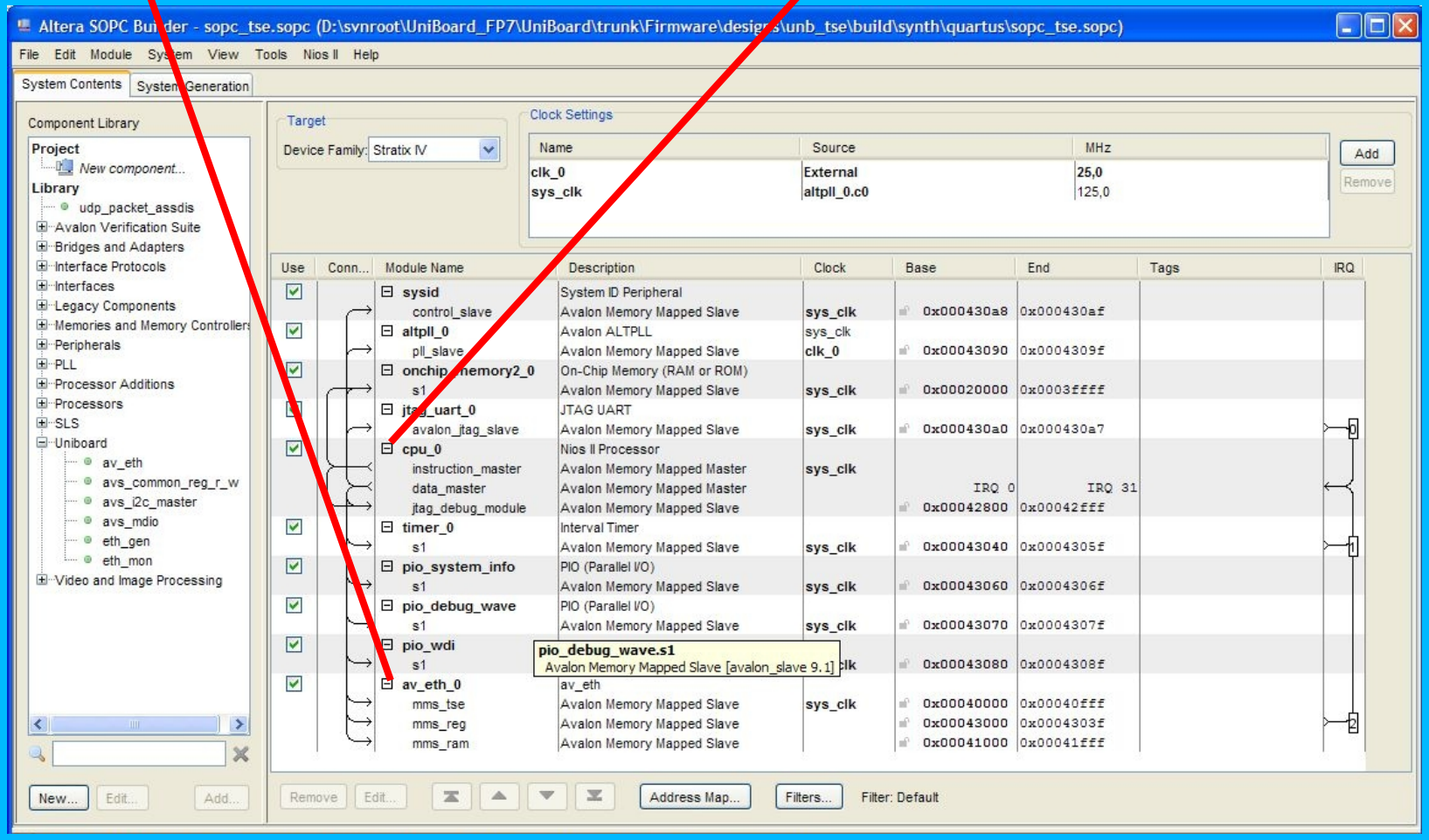
Module VHDL test bench

- Test bench to simulate the ETH module DUT in Modelsim



Altera SOPC system using the ETH module

- ETH module
- Nios2 microprocessor

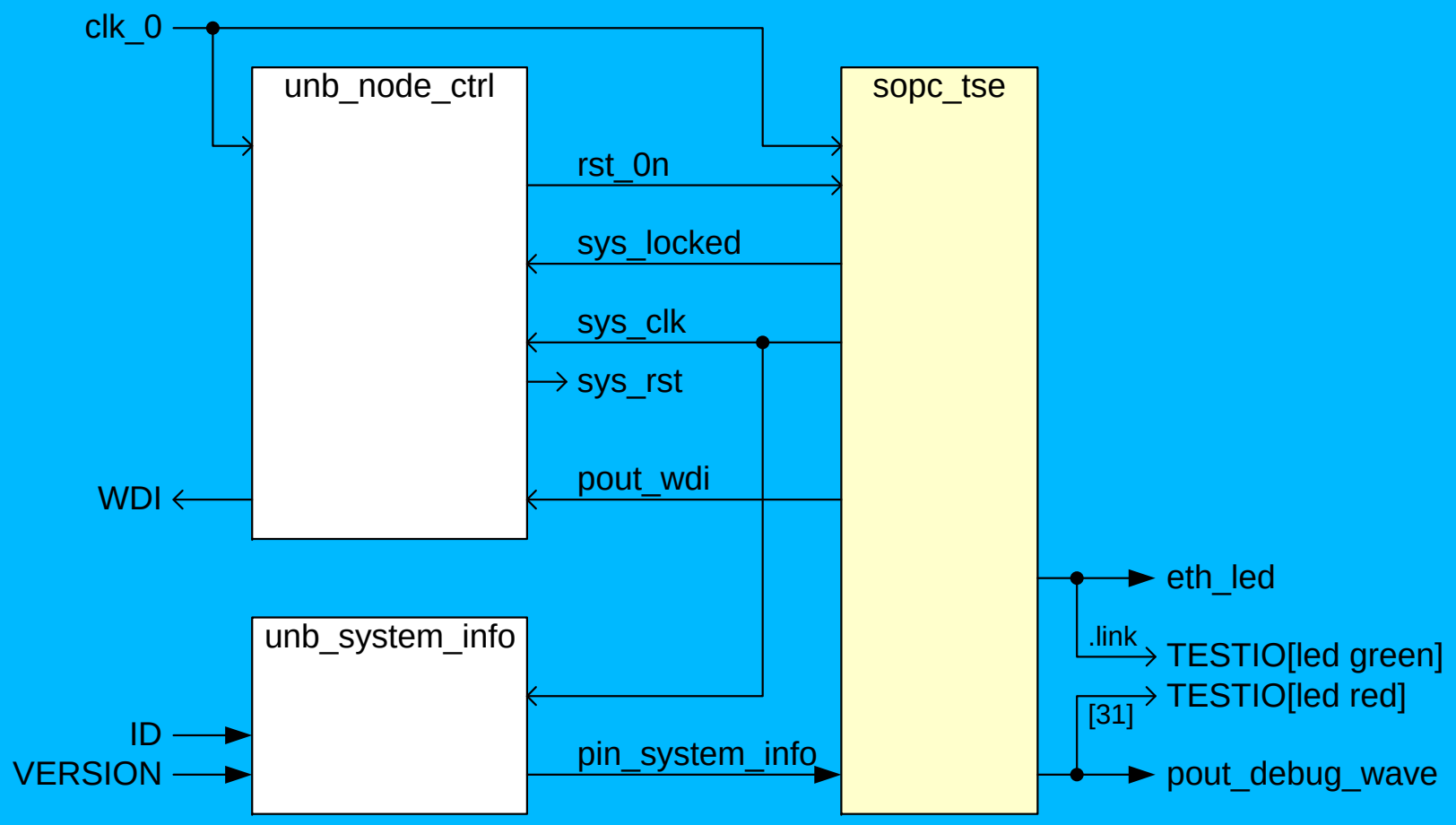


Avalon VHDL wrapper for ETH module

- The ETH module can be made available in SOPC Builder via a hardware description TCL file. This hw_tcl file can be created using the SOPC Component Editor (see $UNB/doc/howto)
- The purpose of the Avalon VHDL wrapper is:
  - To map the ETH entity ports to the Avalon interface convention
  - To allow that the ETH entity definitions can be kept vendor tool independent

Design instantiating the SOPC system

- Design unb_tse can run on all 8 UniBoard nodes (hence name unb_*)

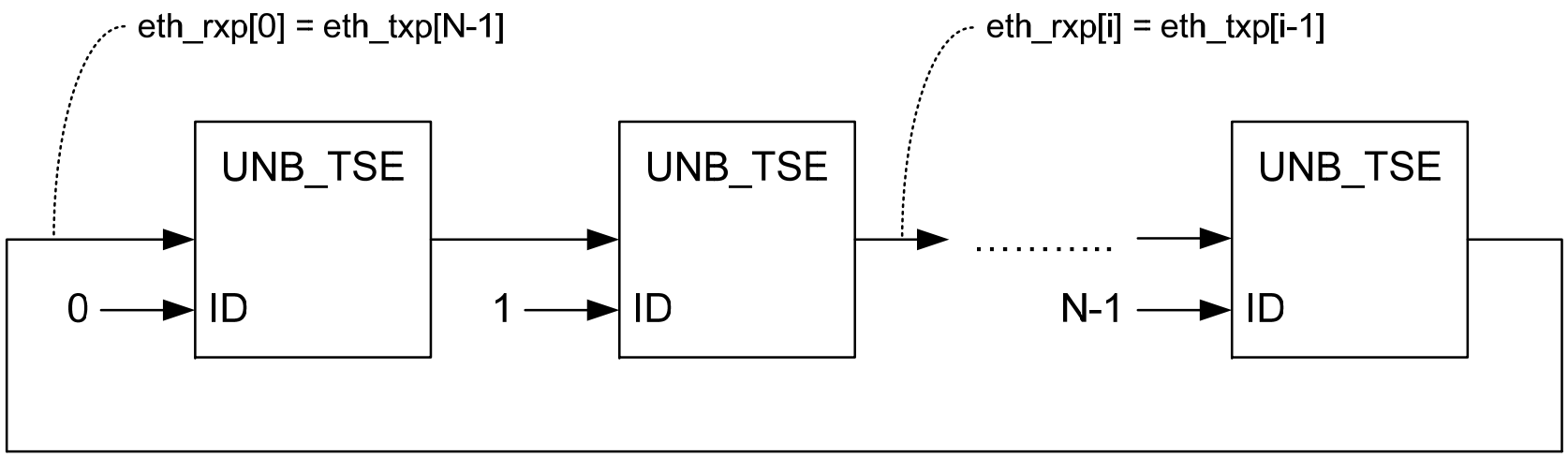


Software functions module

- The SW functions module for the ETH module consists of:
  - avs_eth.h → public functions to control the ETH module
  - avs_eth.c → private implementation
  - avs_eth_regs.h → interface registers defines
- The module software is kept in $UNB/software/modules/src
- The module software functions can be used in a main()
- The SW main() functions are kept in $UNB/software/apps
- The /eth_main_tx_rx/main.c sends and receives frames using interrupts and a tasks loop

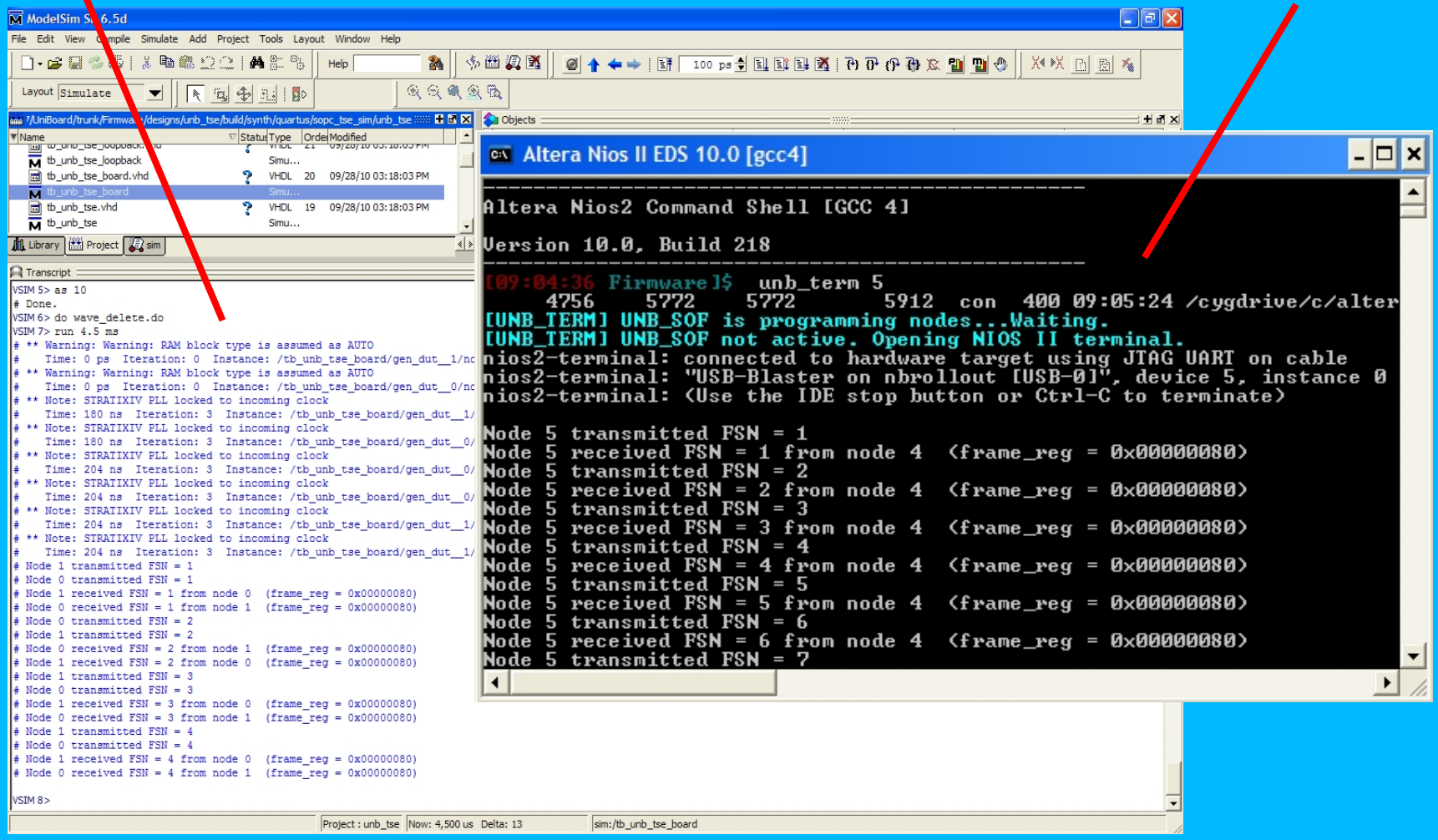
Design VHDL test bench

- Test bench with one or more devices under test (unb_tse)
- Ring provides simple, but adequate model of an Ethernet switch, provided that the nodes transmit to their next neighbour



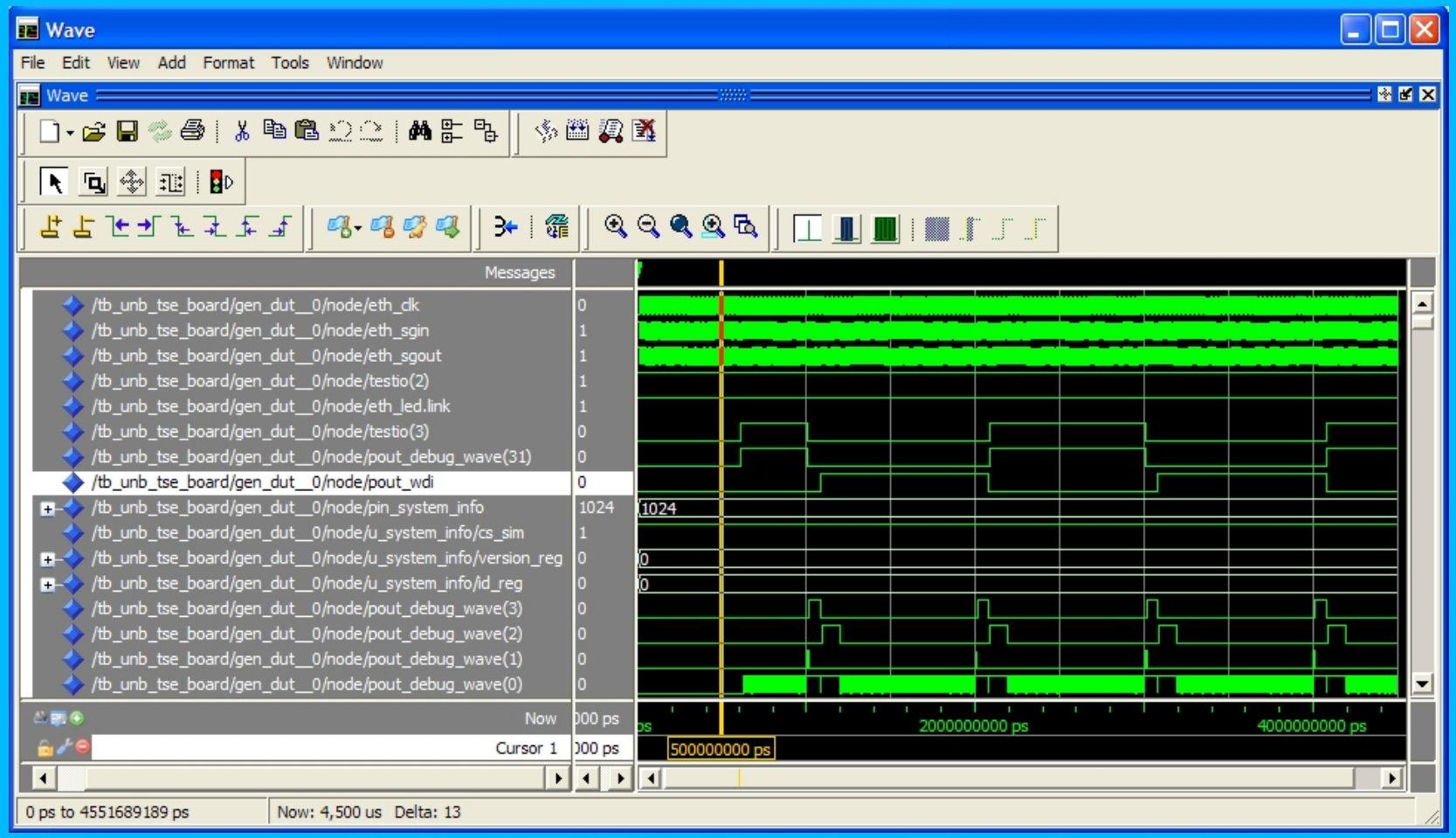
Design verification

- In simulation
- On hardware



Simulation wave window

- Use PIO debug wave to track SW progress in Modelsim wave window



Module Description Document

ETH module document contents:

1. Introduction → Purpose, overview
2. Interface → MM registers and SW module functions (all a SW engineer needs to know), ST ports, …
3. Design → Top level architecture, clock domains
4. Implementation → Lower level architectures
5. Application → SOPC design, synthesis
6. Verification → VHDL test benches, target HW
7. Appendices

Conclusion

- Modular design and module reuse is not new but still applicable
  - The amount of effort to make a module reusable depends on its complexity
  - If a module has an MM interface then it also needs a SW functions module
  - In any case keep general usage in mind while designing
- Modules can be made available in Altera SOPC Builder to allow creating firmware systems via the GUI.
- DSP functions can also be put or grouped into modules. The MM interface and ST interface also suit DSP functions.
- If DSP modules are made available in Altera SOPC Builder then a complete DSP design can be created via the GUI.