

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



1. Why modular design and module reuse?
2. The ETH module as an example
3. Module simulation test bench
4. Altera SOPC Builder system using the ETH module
5. Software functions module
6. Design simulation test bench
7. Verification on hardware
8. Documentation
9. Conclusion on modular design and module reuse



Why modular design and module reuse?

ASTRON

- 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

ASTRON

- 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

- 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

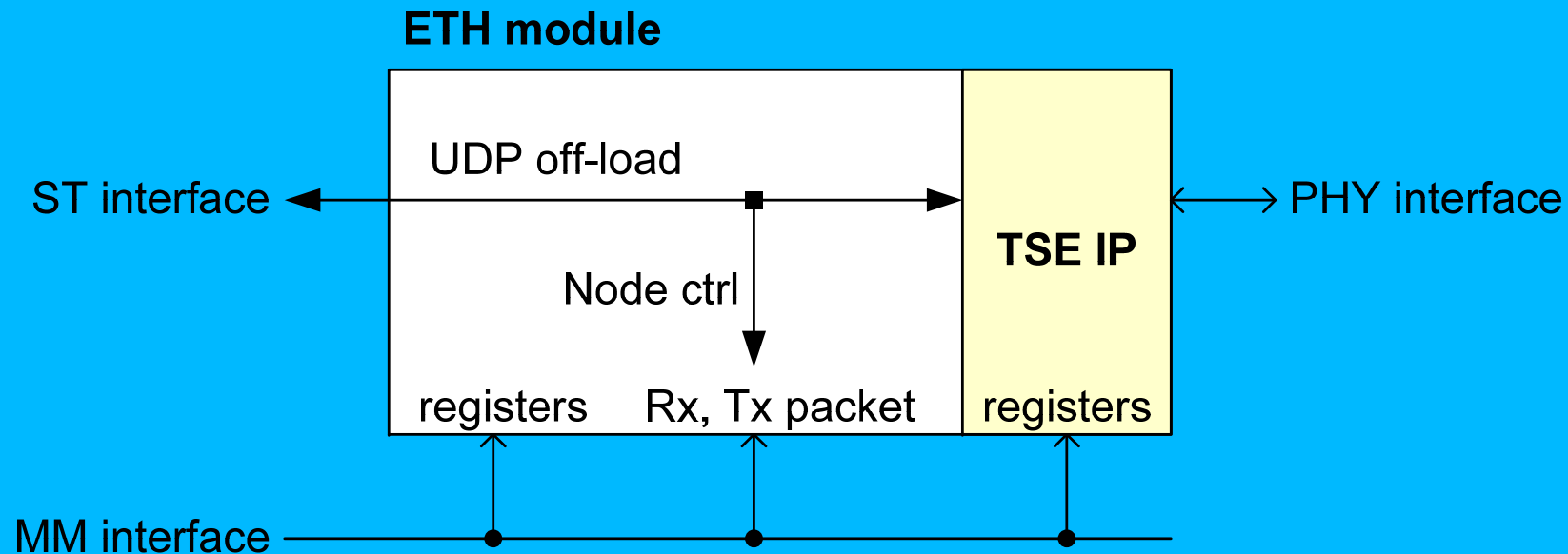
(UNB = https://svn.astron.nl/UniBoard_FP7/UniBoard/trunk/Firmware/)



Available modules in \$UNB/modules/

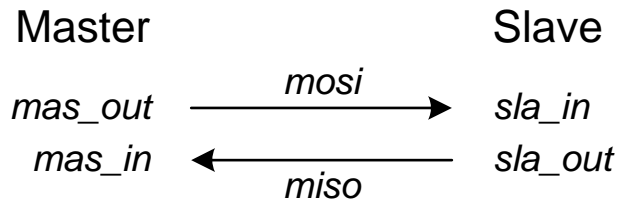
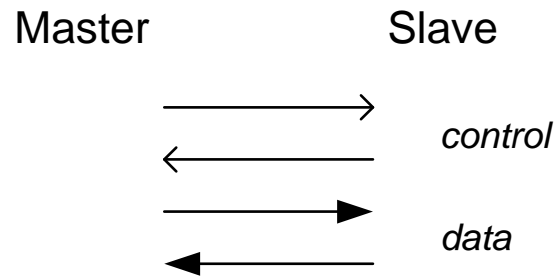
ASTRON

- 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

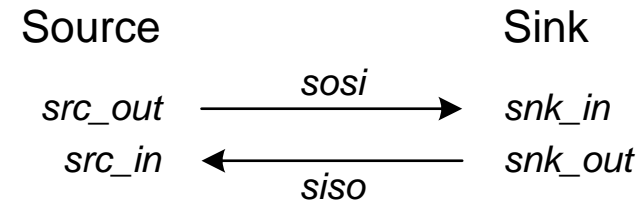
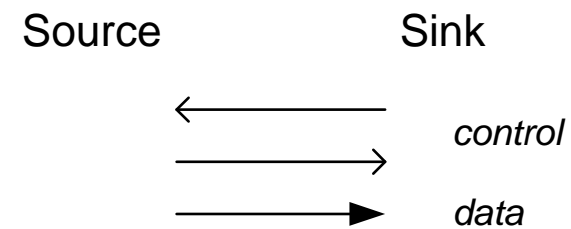


- The ETH module is kept at \$UNB/modules/tse
- Purpose is to:
 - Minimize the processing load for the microprocessor
 - Provide a Memory-Mapped (MM) interface to the TSE MAC IP
 - Provide a Streaming (ST) interface to off-load UDP frames

MM = Memory-Mapped



ST = Streaming



■ MM interface:

```
TYPE t_eth_reg_mm_bus IS RECORD
  -- Master In Slave Out (MISO)
  rddata   : STD_LOGIC_VECTOR(c_eth_data_w-1 DOWNT0 0);
  -- Master Out Slave In (MOSI)
  address  : STD_LOGIC_VECTOR(c_eth_reg_addr_w-1 DOWNT0 0);
  wrdata   : STD_LOGIC_VECTOR(c_eth_data_w-1 DOWNT0 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 DOWNT0 0);
  valid    : STD_LOGIC;
  sop      : STD_LOGIC;
  eop      : STD_LOGIC;
  empty    : STD_LOGIC_VECTOR(c_eth_empty_w-1 DOWNT0 0);
  channel  : STD_LOGIC_VECTOR(c_eth_demux_channel_w-1 DOWNT0 0);
END RECORD;
```

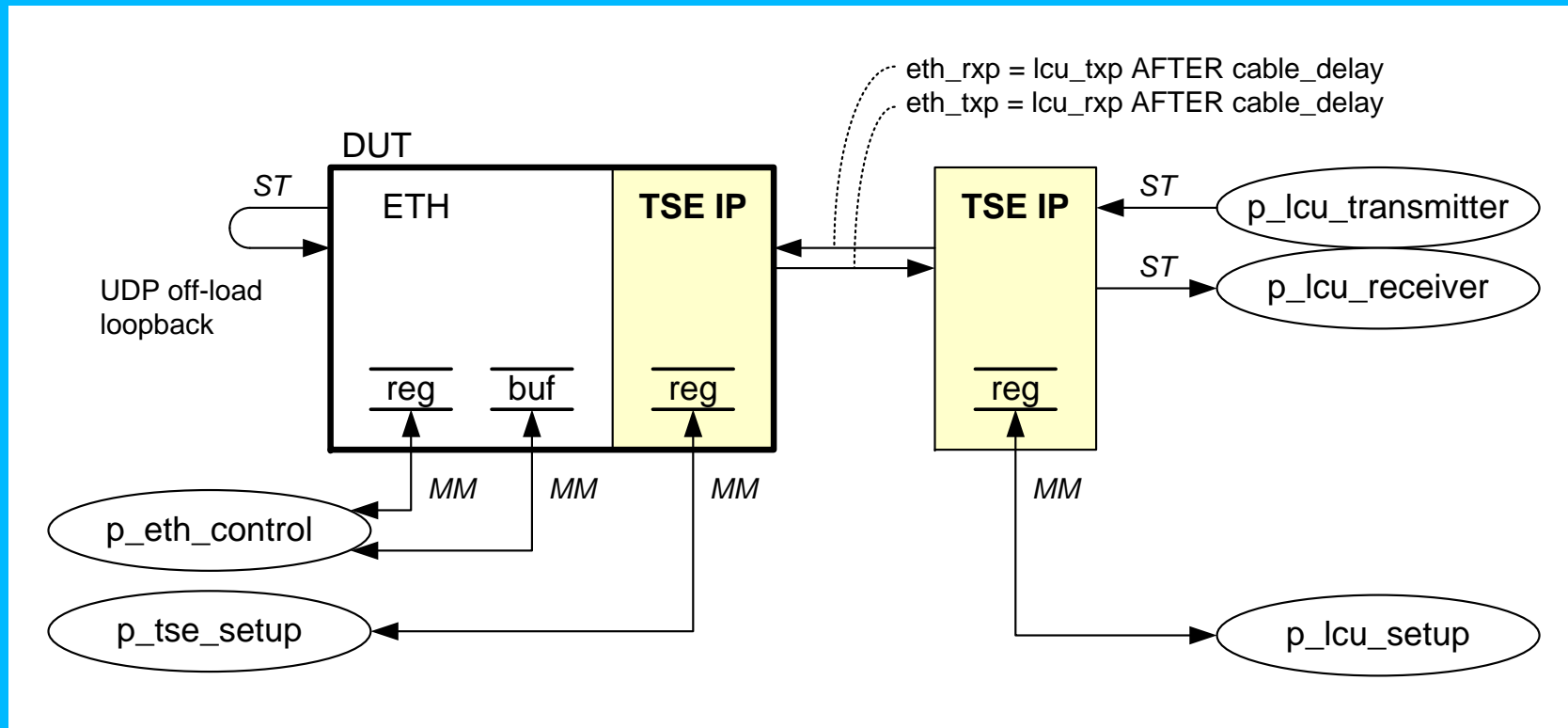


VHDL wrapper to encapsulate IP



- 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

- Test bench to simulate the ETH module DUT in Modelsim



- ETH module
- Nios2 microprocessor

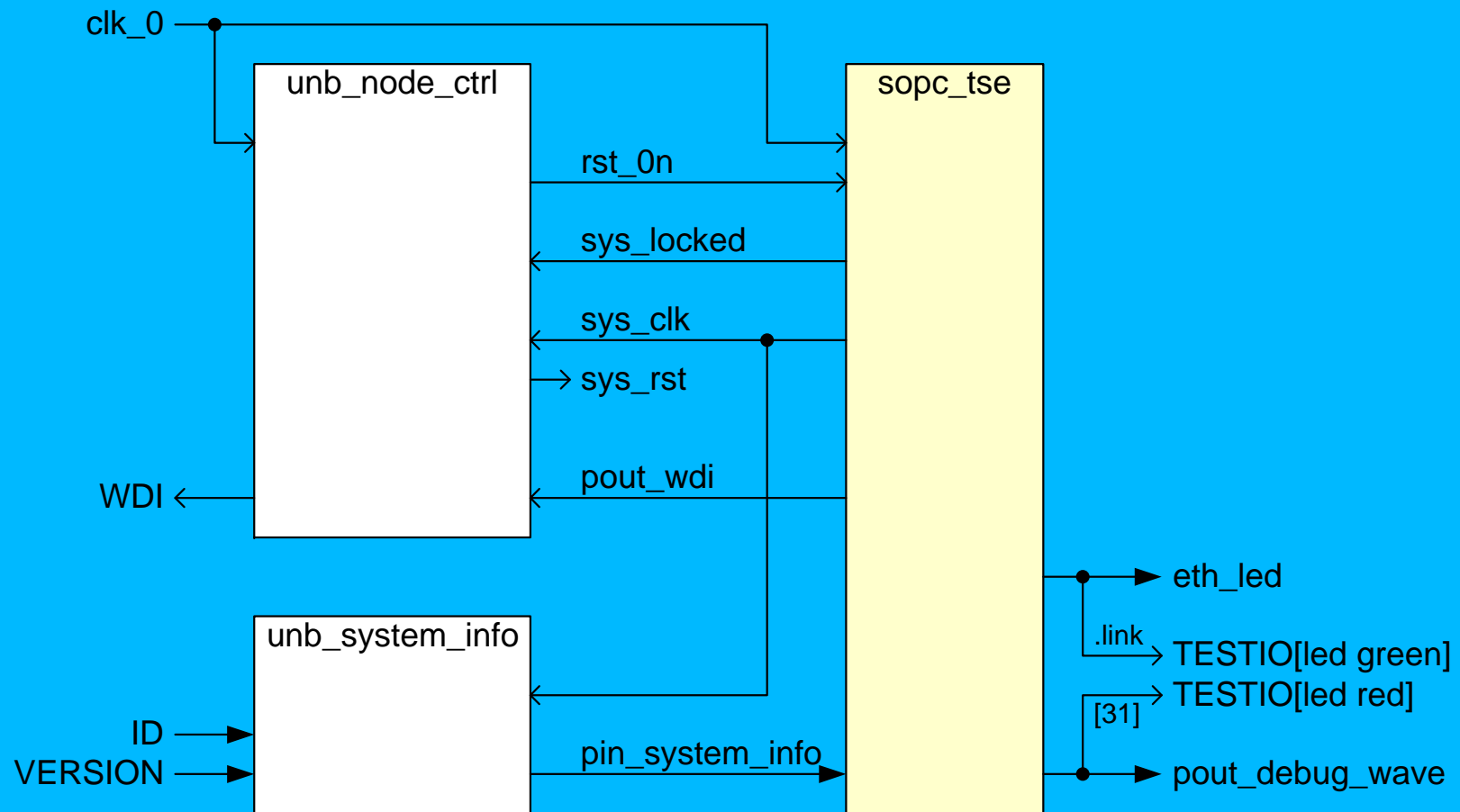
The screenshot shows the Altera SOPC Builder interface. The 'System Contents' tab is active, displaying a table of components and their properties. A red arrow points from the 'ETH module' bullet point to the 'av_eth' component in the table. Another red arrow points from the 'Nios2 microprocessor' bullet point to the 'cpu_0' component in the table.

Use	Conn...	Module Name	Description	Clock	Base	End	Tags	IRQ
<input checked="" type="checkbox"/>		sysid	System ID Peripheral					
<input checked="" type="checkbox"/>		control_slave	Avalon Memory Mapped Slave	sys_clk	0x000430a8	0x000430af		
<input checked="" type="checkbox"/>		altpll_0	Avalon ALTPLL	sys_clk				
<input checked="" type="checkbox"/>		pll_slave	Avalon Memory Mapped Slave	clk_0	0x00043090	0x0004309f		
<input checked="" type="checkbox"/>		onchip_memory2_0	On-Chip Memory (RAM or ROM)					
<input checked="" type="checkbox"/>		s1	Avalon Memory Mapped Slave	sys_clk	0x00020000	0x0003ffff		
<input checked="" type="checkbox"/>		jtag_uart_0	JTAG UART					
<input checked="" type="checkbox"/>		avalon_jtag_slave	Avalon Memory Mapped Slave	sys_clk	0x000430a0	0x000430a7		
<input checked="" type="checkbox"/>		cpu_0	Nios II Processor					
<input checked="" type="checkbox"/>		instruction_master	Avalon Memory Mapped Master	sys_clk				
<input checked="" type="checkbox"/>		data_master	Avalon Memory Mapped Master				IRQ 0	IRQ 31
<input checked="" type="checkbox"/>		jtag_debug_module	Avalon Memory Mapped Slave		0x00042800	0x00042fff		
<input checked="" type="checkbox"/>		timer_0	Interval Timer					
<input checked="" type="checkbox"/>		s1	Avalon Memory Mapped Slave	sys_clk	0x00043040	0x0004305f		
<input checked="" type="checkbox"/>		pio_system_info	PIO (Parallel I/O)					
<input checked="" type="checkbox"/>		s1	Avalon Memory Mapped Slave	sys_clk	0x00043060	0x0004306f		
<input checked="" type="checkbox"/>		pio_debug_wave	PIO (Parallel I/O)					
<input checked="" type="checkbox"/>		s1	Avalon Memory Mapped Slave	sys_clk	0x00043070	0x0004307f		
<input checked="" type="checkbox"/>		pio_wdi	pio_debug_wave.s1	clk	0x00043080	0x0004308f		
<input checked="" type="checkbox"/>		av_eth_0	av_eth					
<input checked="" type="checkbox"/>		mms_tse	Avalon Memory Mapped Slave	sys_clk	0x00040000	0x00040fff		
<input checked="" type="checkbox"/>		mms_reg	Avalon Memory Mapped Slave		0x00043000	0x0004303f		
<input checked="" type="checkbox"/>		mms_ram	Avalon Memory Mapped Slave		0x00041000	0x00041fff		



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



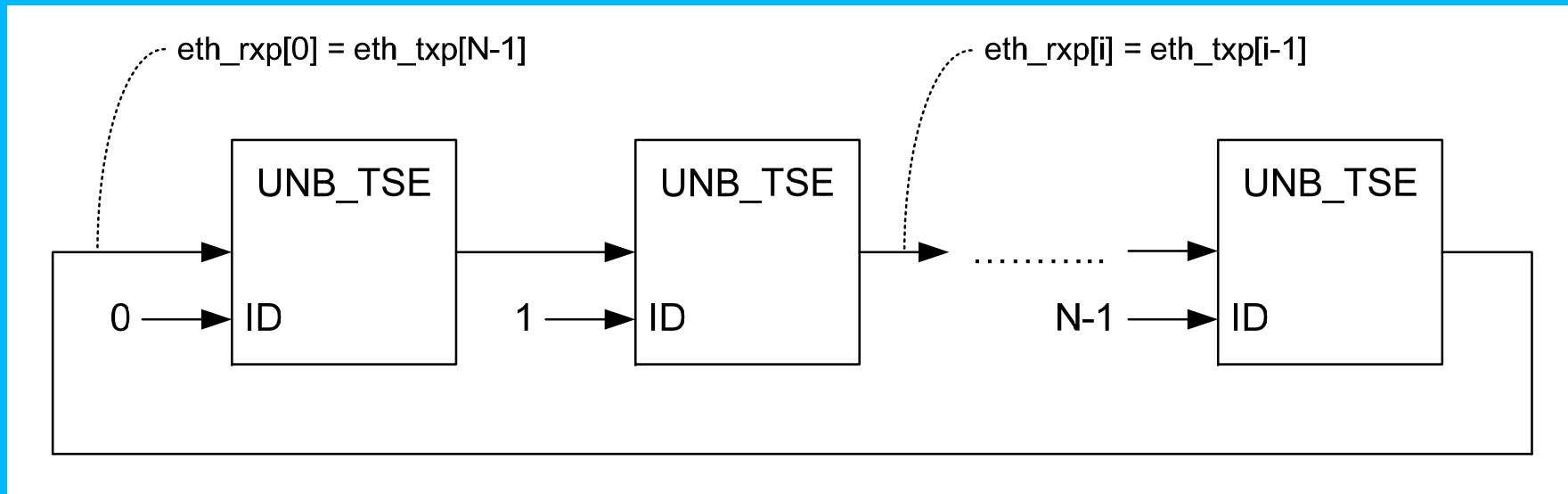


Software functions module



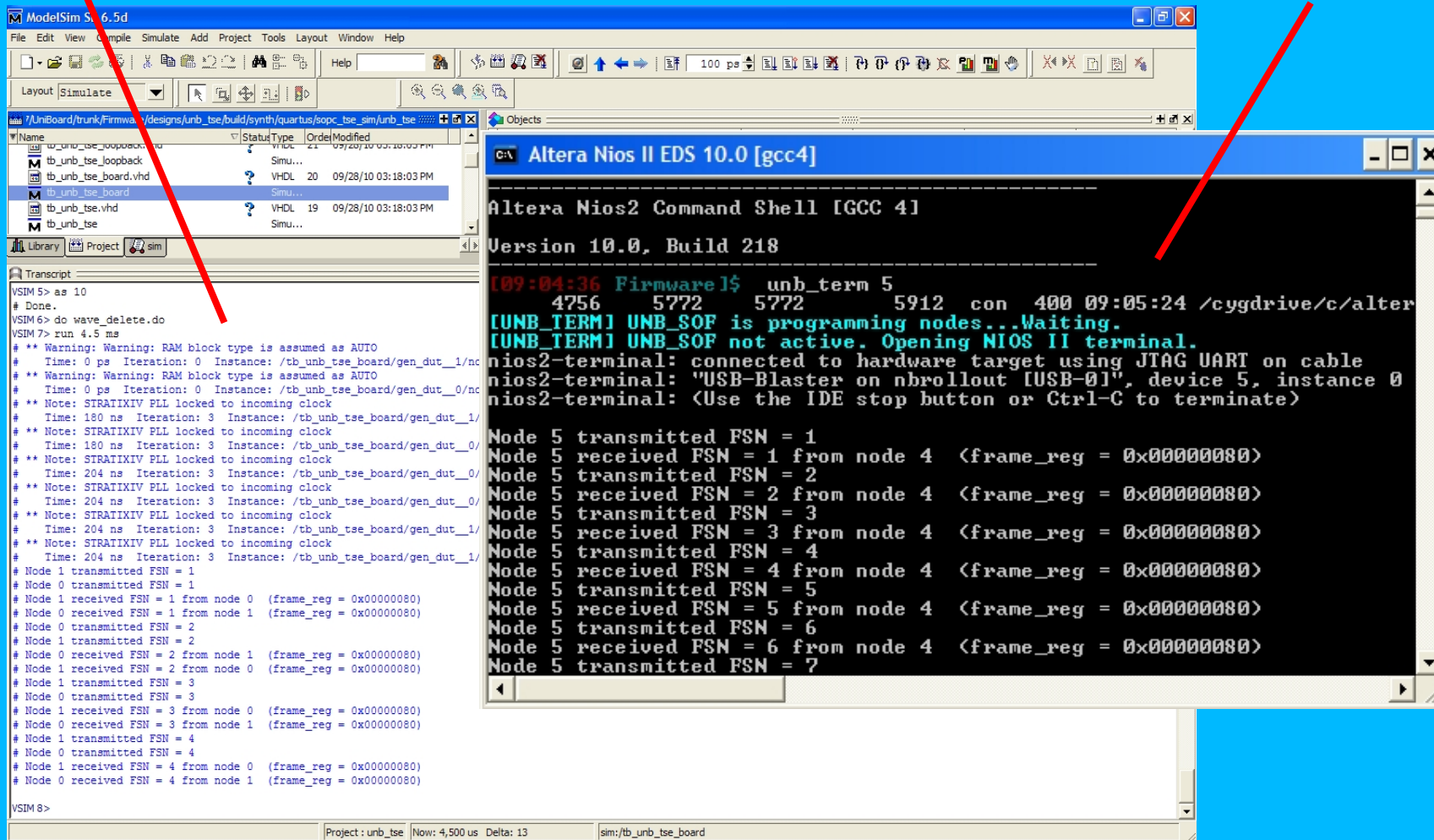
- The SW functions module for the ETH module consists of:
 - `avs_eth.h` → public functions to control the ETH module
 - `avs_eth.c` → private implementation
 - `avs_eth_regs.h` → interface registers defines
- The module software is kept in `$UNB/software/modules/src`
- The module software functions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

- 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



- In simulation

- On hardware



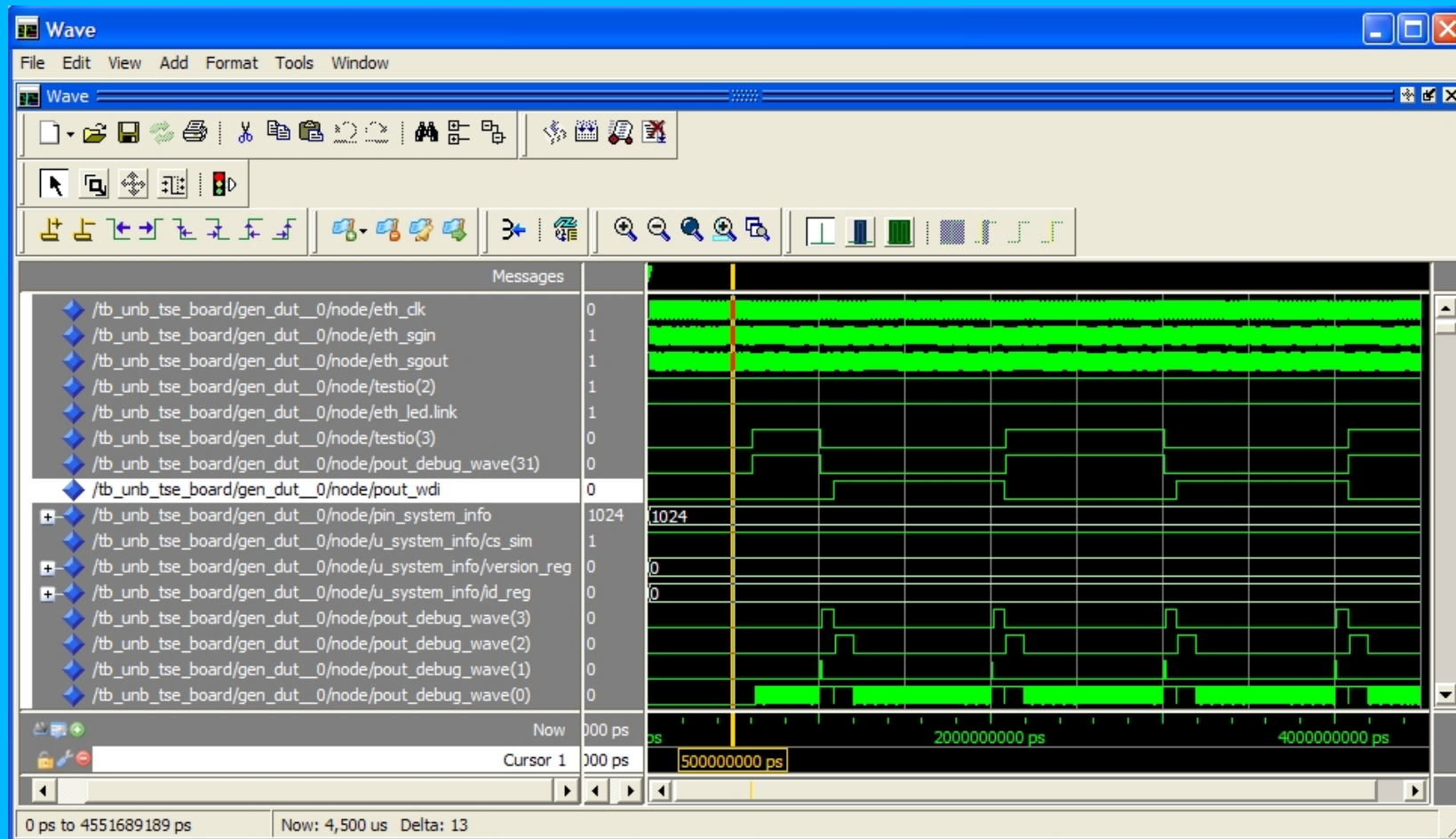
The screenshot shows the ModelSim 6.5d interface. The top toolbar includes icons for simulation control. The main workspace displays a project tree with files like `tb_unb_tse_board.vhd` and `tb_unb_tse.vhd`. The bottom pane shows a transcript of simulation commands and messages, including warnings about RAM block types and PLL locking. A terminal window titled "Altera Nios II EDS 10.0 [gcc4]" is open, displaying the output of the `unb_term 5` command. The terminal output shows the Nios2 command shell version (10.0, Build 218) and the execution of the `unb_term 5` command, which connects to a hardware target using JTAG UART. The terminal output shows a series of transmitted and received FSN (Frame Sequence Number) values, indicating successful communication between nodes 4 and 5.

```

[09:04:36 Firmware] $ unb_term 5
4756 5772 5772 5912 con 400 09:05:24 /cygdrive/c/alter
[LUNB_TERM] UNB_SOF is programming nodes...Waiting.
[LUNB_TERM] UNB_SOF not active. Opening NIOS II terminal.
nios2-terminal: connected to hardware target using JTAG UART on cable
nios2-terminal: "USB-Blaster on nbrollout [USB-01]", device 5, instance 0
nios2-terminal: <Use the IDE stop button or Ctrl-C to terminate>

Node 5 transmitted FSN = 1
Node 5 received FSN = 1 from node 4 <frame_reg = 0x00000080>
Node 5 transmitted FSN = 2
Node 5 received FSN = 2 from node 4 <frame_reg = 0x00000080>
Node 5 transmitted FSN = 3
Node 5 received FSN = 3 from node 4 <frame_reg = 0x00000080>
Node 5 transmitted FSN = 4
Node 5 received FSN = 4 from node 4 <frame_reg = 0x00000080>
Node 5 transmitted FSN = 5
Node 5 received FSN = 5 from node 4 <frame_reg = 0x00000080>
Node 5 transmitted FSN = 6
Node 5 received FSN = 6 from node 4 <frame_reg = 0x00000080>
Node 5 transmitted FSN = 7
    
```

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



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

- 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.