**Goals**

**Today:**
- Introduction to White Rabbit and WR PTP Core (WRPC)
- Setting up environment, creating ISE projects & simulating WRPC designs in Modelsim
- Outcome: simplest functional WRPC design

**Next day:**
- Streaming data via WRPC
- Packet latency measurement
Today's agenda

9:00  Introduction to White Rabbit
10:00 Overview of the White Rabbit PTP Core
10:30 Coffee break
11:00 Demo of the WR Switch and WR Core
12:00 Lunch
14:00 HDL: Setting up environment (ISE, Modelsim)
14:30 HDL: Our development platform: SPEC and DIO Mezzanine
15:15 Coffee break
15:30 HDL: Simplest (non-functional) WR Core design. Simulation and synthesis.
16:00 HDL: Basic synchronization WR Core design.
Lessons for this course (all code included):
http://www.ohwr.org/projects/wr-cores/wiki/Handson_training

Repositories with original sources of all HDL components:

• The WR core and PHY modules: **wr-cores**
  http://www.ohwr.org/projects/wr-cores/repository

• General purpose and Wishbone components: **general-cores**
  http://www.ohwr.org/projects/general-cores/repository/show?rev=proposed_master

**SPEC Card Wiki:**
http://www.ohwr.org/projects/spec/wiki

**DIO Mezzanine Wiki:**
http://www.ohwr.org/projects/fmc-dio-5chtla/wiki
Setting up environment

Required software/tools:

- Xilinx ISE 13.3 or above (Webpack edition is OK)
- Xilinx JTAG cable
- Modelsim SE 10.0 or above. SE version necessary, testbenches will not work with Altera or Xilinx Modelsim editions
- Compiled Xilinx simulation libraries
- A terminal program – Putty or something alike.
- CP21xx USB serial driver installed
  
  http://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx
Building Xilinx simulation libraries

Use the Compxlib wizard:
You may need to copy library assignments from Xilinx's modelsim.ini to system-wide modelsim.ini.
The SPEC Card

- Digital I/O mezzanine
- 12V power jack
- FMC connector
- White Rabbit oscillators
- JTAG connector
- USB UART
- SFP transceiver
- 4 LEDs
- Reset button
- User button
- Spartan-6 XC6SLX45T FPGA
Custom design requirements

- 125 MHz low-jitter reference oscillator. 2.5 ppm accuracy, min. Must be tunable: 10ppm range, BW > 3 kHz, resolution: 16 bits

- Helper oscillator. Frequency must allow the PLL inside the FPGA to produce 62.5 MHz. 10 ppm accuracy. Must be tunable: 100 ppm range, BW > 3 kHz, resolution: 16 bits.

- Transceiver clock driven directly from 125 MHz reference (through dedicated input). Xilinx Spartan-6 GTP officially supported, Altera GXB (Arria & Stratix) and Virtex-6 GTX coming soon.

- Standard SFP socket

- 24AA64 or similar I2C EEPROM

In case of doubt: use SPEC schematics as the reference
Directory structure

**ip_cores**  IP core directory. Contains easy-to-use single file version of BE-CO-HT's core library.

**modules**  Shared VHDL modules developed for this course

**sim**  Verilog/SystemVerilog simulation models

**top**  Top level entity & constraint file for each lesson

**testbench**  Testbenches for each lesson

**syn**  Synthesis directories & ISE projects for the lessons
Simplest functional WR Core design

Spartan 6 FPGA

WRPC

PHY if

clk_sys_i

dac_pll_load_p1_o

dac_pll_data_o

dac_pll_load_p1_o

dac_pll_data_o

clk_dmtd_i

WRF Source

WRF Sink

Timecode if

1-PPS

user-defined module

DAC

VCXO

CLK GEN

spec_serial_dac arb

dac_cs_n_o(0)

dac_clr_n_o

dac_slk_o

dac_din_o

dac_cs_n_o(1)

PLL_BASE

IBUFDS

DS18B20

EEPROM

wr_gtp_phy_spartan6

sfp_bp_o

sfp_bnm_o

sfp_en_i

sfp_exit_i

sfp_sda_i/o

sfp_scl_i/o

sfp_rst_i

sda_i/o

scl_i/o

owr_en_o

owr_i

uart_rxd_i

uart_bxd_o

WRF Source

WRF Sink

Timecode if

1-PPS

62.5MHz

20MHz

62.5MHz
WR Core CPU Firmware

The WR Core has a LM32 CPU inside, that needs a program containing the PTP stack to run:

• *wrc-simulation.ram* – firmware for simulations. Contains a minimalistic initialization procedure that starts up the networking components as fast as possible to speed up simulations. Used when \( g\_simulation = 1 \). Put this file in your testbench directory.

• *wrc-release.ram* – the real firmware, used with synthesis. Put this file in the ISE project directory.
HDL Lessons

- **01**: very basic design, blinking LED on the SPEC
- **02**: WR Core without PHY and oscillators
- **03**: WR Core capable of synchronizing
  - **03a**: Sending Ethernet frames via the WR Core
  - **03b**: Simulating Fabric Interface models
- **04**: Simulating streamers
  - **04a**: Simple streamer demo – trigger distribution
Lesson 03b

VHDL/Verilog signals

Software bus transactions

Packets

T. Włostowski / G.Daniluk

White Rabbit Core Hands-On Training
Lesson 03a

White Rabbit Core Hands-On Training

T. Włostowski / G. Daniluk
Streamer principles

- Streamers transfer data words and transparently encapsulate them into Ethernet frames
- Data words can be n * 16 bits – wide
- Data words can be grouped in blocks
- Blocks can have arbitrary size
- Each block has independent sequence number and CRC

Streamer user interface (data, valid, first, last)

Ethernet frames outputted by the streamer
Streamner interface

Record 1
Word1 Word2 Word3

Record 2
Word1 Word2

Record 3
(lost due to CRC error, no data on Rx_data/valid)

Record 4
Word1 Word2 ...

\textbf{tx/rx\_data}

\textbf{tx/rx\_valid}

\textbf{tx/rx\_last}

\textbf{tx/rx\_dreq}

\textbf{rx\_first}

when \(tx/rx\_dreq = 0\), next clock cycle must not contain a valid data word to transfer (i.e. \(tx/rx\_valid = 0\))

\textbf{rx\_lost}

\textbf{rx\_first} indicates the 1st word of the received record

\textbf{rx\_lost} indicates that one or more preceding records have been lost in this case, it's record 3.