White Rabbit Training
Lab 1

Greg Daniluk

CERN BE-CO-HT

October 4, 2016
Outline

1. WR Devices
2. Running WRPC
3. WR Switch diagnostics
Outline

1. WR Devices
2. Running WRPC
3. WR Switch diagnostics
WR network

GPS

Time & Data Master

Control

Data

WR Switch

WR Switch

WR Switch

WR Switch

WR Switch

WR Node

WR Node

WR Node

Standard GbE Switch

Database

Other Node

Other Node

Other Node

PC

2000 nodes

10km
WR Switch

- Central element of WR network
- 18 port gigabit Ethernet switch with WR features
- Optical transceivers: up to 10km, single-mode fiber
- Fully open design, commercially available
WR Switch operation modes

- **Freerunning Master**: source of time for WR network
- **GrandMaster**: source of time for WR network, synchronized to external reference
- **BoundaryClock**: synchronizes to WR Master and provides synchronization to other WR Slaves
WR Switch - inside

- ARM CPU
- Xilinx Virtex6 FPGA
- Power supply 12V DC 80W
- 64MB DDR2
- 256MB NAND
- 8MB boot flash
- 18 SFP cages
- 1-PPS in
- 62.5 MHz out
- 10 MHz in
- Debug ports
- Cooling FANs
- Back panel
- Front panel
- Management ports
- Clocking resources

G.Daniluk
White Rabbit Training
WR network

Time & Data Master

GPS

10km

2000 nodes

Database

PC

Time

Control

Data

WR Devices

Running WRPC

WR Switch diagnostics

G.Daniluk

White Rabbit Training
WR Node: FMC carriers and mezzanines

FMC-based Hardware Kit

- Carrier boards in PCI-Express, VME, PXIe
- All carrier cards are equipped with a White Rabbit port.
- Mezzanines can use the accurate clock signal and “TAI” (synchronous sampling clock, trigger time tag, ...).
- White Rabbit PTP Core running in the FPGA
White Rabbit PTP Core

- HDL IP-Core running on both Xilinx and Altera FPGAs
- Ethernet MAC implementation
- Implements White Rabbit protocol for precise timing
- Provides user shell over UART for configuration
- SNMP for configuration and monitoring in next stable release
WRPC operation modes

- **Master**: propagates WR time
- **GrandMaster**: WR Master externally disciplined
- **Slave**: synchronizes to another WR Master
WRPC interfaces

- clocks, oscillators control and reset
- PHY (SerDes) interface
- timecode and 1-PPS output
- frame interface (WR Fabric)
**WRPC interfaces**

- **UART**: WRPC shell
- **SPI**: external Flash
- **$I^2C$**: SFP identification
- **1-Wire**: digital thermometer
- **GPIO**: LEDs, SFP detection
WRPC in FPGA design

Spartan 6 FPGA

WRPC

phy_i/f

clock_sys_i

clock_ref_i

rst_n_i

DAC

CLK

GEN

CDCM61004

IBUFGDS

62.5MHz

spec_serial_dac_arb

dac_cs_n_o(0)

dac_dpll_load_p1_o

dac_dpll_data_o

dac_hpll_load_p1_o

dac_hpll_data_o

dac克拉_n_o

dac_dclk_o

dac_din_o

dac克拉_s_n_o(1)

user-defined module

WRF Source

WRF Sink

Timecode i/f

1-PPS

G.Daniluk

White Rabbit Training
Outline

1. WR Devices
2. Running WRPC
3. WR Switch diagnostics
Few steps to run the WR PTP Core

- develop your own FPGA firmware with WRPC
Few steps to run the WR PTP Core

- develop your own FPGA firmware with WRPC
- ...or download demo FPGA firmware from OHWR
Few steps to run the WR PTP Core

- develop your own FPGA firmware with WRPC
- ...or download demo FPGA firmware from OHWR
- program SPEC board with the FPGA firmware
Few steps to run the WR PTP Core

- develop your own FPGA firmware with WRPC
- ...or download demo FPGA firmware from OHWR
- program SPEC board with the FPGA firmware
- configure the WRPC through UART
Few steps to run the WR PTP Core

- develop your own FPGA firmware with WRPC
- ...or download demo FPGA firmware from OHWR
- program SPEC board with the FPGA firmware
- configure the WRPC through UART
- verify the synchronization quality
Download demo FPGA firmware from OHWR

- check our wiki page for the latest stable release:
  www.ohwr.org/projects/wr-cores/wiki/Current_release
- FPGA binary already downloaded for you
- can be found in /lib/firmware/fmc/spec-3.0.bin
program SPEC board with the FPGA firmware

- use Linux drivers and user tools package for SPEC board
- available in the *git* repository:
  
  git://ohwr.org/fmc-projects/spec/spec-sw.git
program SPEC board with the FPGA firmware

- use Linux drivers and user tools package for SPEC board
- available in the `git` repository:
  
git://ohwr.org/fmc-projects/spec/spec-sw.git
- already downloaded in `~/spec-sw`
- to build drivers and tools:
  
cd `~/spec-sw`
make clean; make
Loading kernel drivers for SPEC board

- sudo insmod fmc-bus/kernel/fmc.ko
- sudo insmod kernel/spec.ko
- sudo insmod fmc-bus/kernel/fmc-trivial.ko
  gateware=fmc/spec-3.0.bin
Loading kernel drivers for SPEC board

- sudo insmod fmc-bus/kernel/fmc.ko
- sudo insmod kernel/spec.ko
- sudo insmod fmc-bus/kernel/fmc-trivial.ko
gateway=fmc/spec-3.0.bin

At this point you have the card programmed.
Loading kernel drivers for SPEC board

- `sudo insmod fmc-bus/kernel/fmc.ko`
- `sudo insmod kernel/spec.ko`
- `sudo insmod fmc-bus/kernel/fmc-trivial.ko`  
gateware=fmc/spec-3.0.bin

At this point you have the card programmed.

To connect to the WRPC shell:

```
picocom -b 115200 /dev/ttyUSB0
```
Some most important commands:

- **mode** `<slave/master/gm>` - sets the timing mode
- **ptp** `<start/stop>` - starts/stops the synchronization
- **gui** - shows the synchronization status
- **stat** - same as **gui** but in a parsing-friendly format

Full list available in the WRPC manual.
Init script

Set of shell commands executed on every power-up
Init script

Set of shell commands executed on every power-up

- `init erase` - erases all commands in the init script
- `init add <command>` - adds a new command
- `init show` - shows current content of the script
- `init boot` - executes the script
Exercise 1

Type your own init script and verify the synchronization quality.
Exercise 1

Type your own init script and verify the synchronization quality.

- `init erase`
- `init add ptp stop`
- `init add mode slave`
- `init add ptp start`
- `init show`
Link delay model

- static hardware delays: $\Delta_{TXM}, \Delta_{RXM}, \Delta_{TXS}, \Delta_{RXS}$
- semi-static hardware delays: $\epsilon_M, \epsilon_S$
- fiber asymmetry coefficient: $\alpha = \frac{\delta_{MS} - \delta_{SM}}{\delta_{SM}}$
Link delay model

- static hardware delays: $\Delta_{TXM}, \Delta_{RXM}, \Delta_{TXS}, \Delta_{RXS}$
- semi-static hardware delays: $\epsilon_M, \epsilon_S$
- fiber asymmetry coefficient: $\alpha = \frac{\delta_{MS} - \delta_{SM}}{\delta_{SM}}$

- default calibration parameters: [link]
- calibration procedure to find best $\Delta_{TXM}, \Delta_{RXM}, \Delta_{TXS}, \Delta_{RXS}$ and $\alpha$ for your setup.
Calibration values: $\Delta_{TX}$, $\Delta_{RX}$, $\alpha$ associated with SFP transceiver - we call it SFP database
Entering calibration values

Calibration values: $\Delta_{TX}, \Delta_{RX}, \alpha$ associated with SFP transceiver - we call it SFP database

- `sfp erase` - erases all entries in the SFP database
- `sfp add <part number> <\Delta_{TX}> <\Delta_{RX}> <\alpha>` - adds a new entry to the SFP database
- `sfp show` - shows all stored entries
- `sfp detect` - reads the part number of currently used SFP
- `sfp match` - loads calibration values from the SFP database for currently used transceiver
Exercise 2

1. Enter calibration values to improve the synchronization quality.
Exercise 2

1. Enter calibration values to improve the synchronization quality.

- sfp erase
- sfp add AXGE-1254-0531 180625 148451 72169888
- sfp add AXGE-3454-0531 180625 148451 -73685416
- sfp show
Exercise 2

2. Improve init script to automatically load calibration values.
Exercise 2

2. Improve init script to automatically load calibration values.

- init erase
- init add ptp stop
- init add sfp detect
- init add sfp match
- init add mode slave
- init add ptp start
Exercise 2

3. Re-program the SPEC board to make sure everything is set properly.
Exercise 2

3. Re-program the SPEC board to make sure everything is set properly.
   - open a new terminal
   - cd ~/spec-sw
   - sudo rmmod fmc-trivial
Exercise 2

3. Re-program the SPEC board to make sure everything is set properly.

- open a new terminal
- cd ~/spec-sw
- sudo rmmod fmc-trivial
- sudo insmod fmc-bus/kernel/fmc-trivial.ko
gateware=fmc/spec-3.0.bin
Outline

1. WR Devices
2. Running WRPC
3. WR Switch diagnostics
WR Switch monitoring and configuration

- Monitoring
  - SNMP
  - Syslog

- Configuration
  - local configuration file
  - fetching configuration on boot time (HTTP / TFTP / FTP)
  - web-interface
WR Switch SNMP

- wrsMainSystemStatus
- wrsOSStatus
  - wrsBootSuccessful
  - wrsTemperatureWarning
  - wrsMemoryFreeLow
  - wrsCpuLoadHigh
  - wrsDiskSpaceLow
- wrsTimingStatus
  - wrsPTPStatus
  - wrsSoftPLLStatus
  - wrsSlaveLinksStatus
  - wrsPTPFramesFlowing
- wrsNetworkingStatus
  - wrsSFPsStatus
  - wrsEndpointStatus
  - wrsSwcoreStatus
  - wrsRTUStatus
Exercise 3

Let’s create some errors on the switch.
Need more information?

- WR PTP Core project page:
  www.ohwr.org/projects/wr-cores/wiki/Wrpc_core

- WR Switch project page:
  www.ohwr.org/projects/white-rabbit/wiki/Switch
WRPC: Clocks and reset

- System clock (62.5 MHz)
- Reference clock (125 MHz)
- DMTD clock (~62.5 MHz)
- Disciplines two VCXOs
- Optional external clock (10 MHz) input
WRPC: PHY (SerDes) interface

- has to be used with PHY component from WR repositories
- transmission and reception paths
- fed with 125 MHz reference clock
WRPC: Timecode and 1-PPS interface

- link up/down indicator
- time valid indicator
- WR time
  - seconds
  - fractional part of second (in 8ns cycles)
- 1 pulse-per-second output
**WRPC: Miscellaneous interfaces**

- **UART:** WRPC shell
- **SPI:** external Flash
- **I²C:** SFP identification
- **1-Wire:** digital thermometer
- **GPIO:** LEDs, SFP detection
**WRPC: Fabric interface**

- for sending/receiving user-defined Ethernet frames
- two separate pipelined Wishbone buses
- **WRF Source** = WB Master
- **WRF Sink** = WB Slave