White Rabbit PTP Core release tests

CERN BE-CO-HT

Grzegorz Daniluk, December 19, 2013

1 Introduction

This document describes a set of acceptance tests for White Rabbit PTP Core (*WRPC*). Developers of the core have to make sure, all of them are passed before each new stable release is announced. Users of the *WRPC* can also use the document as an example of test scenarios for their projects combining WR PTP Core with other modules.

2 Requirements

Hardware required to perform the release tests:

- 2x Simple PCIe FMC Carrier (SPEC) with FMC DIO 5-channel module
- 1x WR Switch
- 1x 2-channel oscilloscope (200 MHz, 2 GS/s)
- 1x non-WR PTP (IEEE1588-2008) Switch

3 Acceptance tests

1. Verify with Xilinx ISE that all timing constraints for Release Candidate gateware synthesis are met.
2. Compile *wrpc-sw* trying different Kconfig settings and make sure the compilation succeeds without warnings and errors (*make -s*):
   - 2.1 compile with Etherbone support disabled
   - 2.2 compile with Etherbone support enabled
3. Compile *wrpc-sw* with default SPEC configuration (PPSI engine selected) for further tests.
4. Calibrate Release Candidate gateware to WR Switch
5. Create a basic WRPC init script:

```
wrc# init erase
wrc# init add ptp stop
wrc# init add sfp detect
wrc# init add sfp match
wrc# init add mode master
wrc# init add calibration
wrc# init add ptp start
```

Execute it on SPEC using *init boot* shell command and verify if the output doesn't contain any information about errors. Example of correct output:

```
wrc# init boot
executing: ptp stop
executing: sfp detect
AXGE-3454-0531
executing: sfp match
SFP matched, dTx=180456, dRx=148066, alpha=-73685416
executing: mode master
Locking PLL...
executing: calibration
Found phase transition in EEPROM: 6850ps
executing: ptp start
```

6. Set WRPC mode to Grand Master (*mode gm*) and try setting current time manually using *time set*, *time setsec*, *time setnsec* commands. Check if those commands work independently e.g. if *time setsec* does not reset the nanoseconds counter. You can check that by observing on the oscilloscope the offset between 1-PPS from the WRPC and from an external time source to which the WRPC is synchronized.

7. Set WRPC to Master mode (*mode master*) and connect it to the Slave port of a WR Switch, connect another SPEC running the WRPC in Slave mode to one of the Master ports of the WR Switch. Don't forget to run automated t24p calibration on both SPECs (for that you must set Master WRPC to Slave state for a moment and let it calibrate):

   7.1 check if 1-PPS signals from 2 SPECs are in sync with the 1-PPS of WR Switch

   7.2 check if SPEC produces stable 1-PPS (use oscilloscope with persistent mode turned on)

   7.3 run *stat cont* and *gui* commands for few minutes and check if they don't they disturb timing by measuring 1-PPS offset/jitter on the oscilloscope

   7.4 collect logs from *stat cont* WRPC Shell command to the file and use *wr_graph.py* script to analyze them:

   (a) check if *lock* is always 1
   (b) check, once WR link is established in Slave mode, if the WR servo is always in *TRACK_PHASE* state
   (c) check if *sec* counts up monotonically
   (d) check if *mu* changes without any sudden few-ns jumps
   (e) check if *cko* is always in a range -50ps; +50ps

   Expected output from *wr_graph.py* if all tests are passed:

```
LOCK:    Success, always locked
STATE:   Success, always TRACK_PHASE
TIME:    Success, always growing
RTT:     Success, no jumps detected
CKO:     Success, no values outside accepted range
```

8. Without restarting the PTP deamon, disconnect the Slave SPEC and connect it to another port of the WR Switch:

   8.1. check if the Slave has synchronized to the new WR Switch port

8.2. check *gui* or *stat [cont]* to verify the servo state is *TRACK_PHASE* and clock offset is in the range -50ps; +50ps

9. Connect two SPECs together, configure one as WR Slave, second as WR Master
   - perform the same tests as in scenario described in point 7.
10. Connect two SPECs together, configure one as Slave, second as Grand Master
    - perform the same tests as in scenario described in point 7.
11. Connect SPEC to a regular Gigabit network card, run *Wireshark* or *tcpdump* to collect frames sent by SPEC:
    11.1. set mode to Grand Master, wait until it's locked and check if the clock class in the captured Announce Message is **6**
    11.2. check with the oscilloscope if SPEC produces 1-PPS output in Grand Master mode
    11.3. re-lock SPEC to Grand Master mode few times and check if the 1-PPS skew to the external clock (e.g. Cesium) doesn't change on each lock
    11.4. while SPEC is running, disconnect 10MHz input and check if the clock class in captured Announce Message was degraded to **52**
    11.5. try locking Grand Master (*mode gm*) when 10MHz is not connected, it should fail with *Lock timeout. Command "mode": error -116* error message. Check then if the clock class in the Announce Message is **52**
    11.6. set mode to Master, check if the clock class in captured Announce Message is **187**
    11.7. check with the oscilloscope if SPEC produces 1-PPS output in Master mode
12. Connect SPEC to WR Switch and set its mode to Slave, then without restarting the PTP daemon, unplug the fiber and connect SPEC to a non-WR Master (e.g. a regular PTP Switch)
    12.1. check if WR PTP Core switched to non-WR Slave mode
    12.2. connect back to WR-Switch and check if SPEC switched back to WR Slave mode

4 Tests results

1.	
2.1	
2.2	
3.	
4.	
5.	
6.	
7.1	
7.2	

7.3	
7.4.a	
7.4.b	
7.4.c	
7.4.d	
7.4.e	
8.1	
8.2	

9.1	
9.2	
9.3	
9.4.a	
9.4.b	
9.4.c	
9.4.d	
9.4.e	

10.1	
10.2	
10.3	
10.4.a	
10.4.b	
10.4.c	
10.4.d	
10.4.e	

11.1	
11.2	
11.3	
11.4	
11.5	
11.6	
12.1	
12.2	