Simple Design and Testing Description

XGE-PTPv2 Prototype Validation Platform

The XGE-PTPv2 prototype validation platform is an Electronic System Level (ESL) prototype validation platform independently developed by the author using SystemC TLM. It allows for hardware-software co-design and early development testing of related software and algorithms.

The top-level module (testbench) of the prototype platform is located in the xge-ptpv2/esl/sc directory, specifically in the C++ files testbench.h/.cpp.
The top-level module testbench instantiates one ptp_instance module (for loop back testing) or two ptp_instance modules (for PTPv2 Protocol testing).
The ptp_instance is equivalent to a SoC (System on Chip) containing PTP hardware engine functionality. Its sub-modules m_initiator_top and m_target_top are connected via an abstract bus m_bus. For specific implementation details, refer to the C++ files ptp_instance.h/cpp in the xge-ptpv2/esl/sc directory.
The m_initiator_top module includes the sub-module m_controller, which can be considered a simple abstraction of a CPU running an operating system. For more details, refer to the C++ files controller.h/.cpp in the xge-ptpv2/esl/sc directory.

PTPv2 application software, including loop back testing and Protocol and algorithm testing, runs on the m_controller. The functions declared in the controller's constructor using the SC_THREAD macro can be regarded as threads running in the operating system.
The controller_thread is the main program's thread, while the isr_thread is the interrupt service routine.

Readers interested in using a RISC instruction set simulator in place of the controller to better reflect real-world scenarios are encouraged to do so.

Software Design Description

Software code intended to run on the CPU is located in the xge-ptpv2/esl/sw directory. If readers are only interested in the PTPv2 software design or are not familiar with hardware description languages like SystemC/Verilog, reading the C++ source files in the sw directory along with controller.h/.cpp should suffice.

The design of the protocol software and clock servo algorithm referenced the following open-source project:
https://sourceforge.net/projects/ptpd/

The XGE-PTPv2 project has made significant modifications to the above open-source software, such as object-oriented transformation using the C++ language, deletion of irrelevant code, and adjustment of the software to accommodate the XGE-PTPv2 hardware interface.

Compilation and Simulation Execution

The design and testing are based on the Linux operating system and require the installation of Verilator, SystemC, and CMake. CMake can be installed directly using the Linux package management tool. For specific installation methods of Verilator and SystemC, refer to their official websites or the developer's public account articles.

To run the simulation, follow these steps:
```
cd /path/to/solution
rm -rf build && mkdir build && cd build
cmake ..
make
./ptpv2_tlm
```

If debugging is required, including setting breakpoints and single-step execution, install Visual Studio Code and open the ESL project directory xge-ptpv2/esl.

Software Testing Description

Figure 1 shows the interface after launching the ./ptpv2_tlm program.

```
[z[redacted]@CentOS8-[redacted] build]$ ./ptpv2_tlm

        SystemC 2.3.3-Accellera --- Jan 13 2023 21:07:32
        Copyright (c) 1996-2018 by all Contributors,
        ALL RIGHTS RESERVED

Select the Application
0 : Loop Back Test for Local Device
1 : PTPv2 Protocol Test
Please Input the Selection: 1

                Clock ID: 1
============================================================
            ####  PTPv2 Protocol Test Start!  ####


                Clock ID: 2
============================================================
            ####  PTPv2 Protocol Test Start!  ####

    Clock ID: 2   (ptpd info)         Clock ID: 1   (ptpd info)     08:00:00.000000  (____)
    Clock ID: 2   (ptpd info)     08:00:00.000000  (____)
    Clock ID: 1   (ptpd info)     08:00:00.000000  (____)  Starting ptpd2 daemon with parameters:      ptpv2d -e -z
    Clock ID: 2   (ptpd debug1)   08:00:00.000000  (____)  Starting ptpd2 daemon with parameters:      ptpv2d -e -z
    Clock ID: 1   (ptpd debug1)   08:00:00.000000  (____)  allocated 1608 bytes for protocol engine data
```

Figure 1

Selecting 0 and pressing Enter initiates the Loop Back Test, while selecting 1 initiates the PTPv2 Protocol Test.

The Loop Back Test is relatively simple. It involves sending a frame and receiving it via loop back to test the setting and reading of control registers, checking whether the status registers correctly reflect the hardware status, and verifying the generation of interrupts. Further details are omitted here.

The PTPv2 Protocol Test is used to verify the correct execution of the protocol software and the corresponding Clock Servo algorithm on the XGE-PTPv2 hardware.
Entry function: ptpd::exec()
Execution function: protocol::protocolExec(RunTimeOpts *rtOpts, PtpClock *ptpClock)

In essence, the protocolExec function is an infinite loop-based state machine. The events driving the state machine can be attributed to various interrupt signals, with timing interrupt signals being particularly important. The interval for generating timing interrupts is set to 7.8125ms, corresponding to 128 interrupts per second, which facilitates the transmission of various messages in accordance with the PTPv2 standard. Of course, readers can use the operating system's native timer when porting the program, although it may not be as precise.

In the file xge-ptpv2/esl/sw/dep/constants_dep.h, the macro constant PTP_SYNCE determines whether the Protocol test is in SyncE mode or non-SyncE mode. If PTP_SYNCE is set to 1, it is in SyncE mode; if set to 0, it is in non-SyncE mode.

In the simulation, the SyncE state refers to the scenario where the rtc_clk/tx_clk clock frequencies of the two ptp_instance modules are the same, and the values of INITIAL_TICK and LP_INITIAL_TICK are also identical.
In the non-SyncE state, theoretically, there should be a difference in clock frequencies. However, due to Verilator's simulation time precision being approximately 100ps, a 100PPM frequency difference is nearly indistinguishable in the simulation. Therefore, we achieve an

equivalent frequency difference by setting LP_INITIAL_TICK to a value different from INITIAL_TICK.

Figure 2 shows the results of the PTPv2 Protocol Test in SyncE mode with PTP_SYNCE set to 1.

```
    Clock ID: 1   (ptpd debug3)   12:46:17.150006  (slv)  Message Sync :
    Clock ID: 1   (ptpd debug3)   12:46:17.150006  (slv)  Integer 48 :
    Clock ID: 1   (ptpd debug3)   12:46:17.150006  (slv)  LSB : 1687236377
    Clock ID: 1   (ptpd debug3)   12:46:17.150006  (slv)  MSB : 0
    Clock ID: 1   (ptpd debug3)   12:46:17.150006  (slv)  nanoseconds 150001305
    Clock ID: 1   (ptpd debug3)   12:46:17.150006  (slv)
    Clock ID: 1   (ptpd debug3)   12:46:17.150006  (slv)  seconds : 0
    Clock ID: 1   (ptpd debug3)   12:46:17.150007  (slv)  nanoseconds 0
    Clock ID: 1   (ptpd debug3)   12:46:17.150007  (slv)  ==> updateOffset
    Clock ID: 1   (ptpd debug3)   12:46:17.150007  (slv)  offset filter 0
    Clock ID: 1   (ptpd debug3)   12:46:17.150007  (slv)  ==> updateClock
    Clock ID: 1   (ptpd debug1)   12:46:17.150007  (slv)       Observed_drift with AI component: 0
    Clock ID: 1   (ptpd debug1)   12:46:17.150007  (slv)       After PI: Adj: 0   Drift: 0   OFM 0
    Clock ID: 1   (ptpd debug2)   12:46:17.150008  (slv)       adjTickRate2: call adjTickRate to 0.000000 PPM
    Clock ID: 1   (ptpd debug3)   12:46:17.150008  (slv)
--Offset Correction--
    Clock ID: 1   (ptpd debug3)   12:46:17.150008  (slv)  Raw offset from master:           0s         140ns
    Clock ID: 1   (ptpd debug3)   12:46:17.150008  (slv)
--Offset and Delay filtered--
    Clock ID: 1   (ptpd debug3)   12:46:17.150008  (slv)  one-way delay averaged (P2P):           0s         140ns
    Clock ID: 1   (ptpd debug3)   12:46:17.150008  (slv)  offset from master:               0s           0ns
    Clock ID: 1   (ptpd debug3)   12:46:17.150009  (slv)  observed drift:             0,     0.000000 PPM
    Clock ID: 1   (ptpd debug3)   12:46:17.150009  (slv)  activity
 Simulation: 500 milliseconds elapsed!
 PTP Slave, Clock ID 1, reach synchronized convergence state,  Stop!

Info: /OSCI/SystemC: Simulation stopped by user.
```

Figure 2

In the SyncE state, the simulation results indicate that the Protocol and Clock Servo algorithm can synchronize the PTP Slave and PTP Master in less than 1 second with very high precision. The offsetFromMaster is around 0 ns, and the observed frequency difference is 0.0 PPM.
On the author's computer, the entire simulation from start to finish took approximately twenty minutes.

Figure 3 shows the results of the PTPv2 Protocol Test in non-SyncE mode with PTP_SYNCE set to 0, where a Master-Slave frequency difference of 10.0 PPM is configured.

```
    Clock ID: 1   (ptpd debug3)   14:05:11.840005  (slv)  Sync message received :
    Clock ID: 1   (ptpd debug3)   14:05:11.840005  (slv)  Message Sync :
    Clock ID: 1   (ptpd debug3)   14:05:11.840005  (slv)  Integer 48 :
    Clock ID: 1   (ptpd debug3)   14:05:11.840006  (slv)  LSB : 1687241111
    Clock ID: 1   (ptpd debug3)   14:05:11.840006  (slv)  MSB : 0
    Clock ID: 1   (ptpd debug3)   14:05:11.840006  (slv)  nanoseconds 840001216
    Clock ID: 1   (ptpd debug3)   14:05:11.840006  (slv)
    Clock ID: 1   (ptpd debug3)   14:05:11.840006  (slv)  seconds : 0
    Clock ID: 1   (ptpd debug3)   14:05:11.840006  (slv)  nanoseconds 0
    Clock ID: 1   (ptpd debug3)   14:05:11.840006  (slv)  ==> updateOffset
    Clock ID: 1   (ptpd debug3)   14:05:11.840007  (slv)  offset filter -24
    Clock ID: 1   (ptpd debug3)   14:05:11.840007  (slv)  ==> updateClock
    Clock ID: 1   (ptpd debug1)   14:05:11.840007  (slv)       Observed_drift with AI component: -4311
    Clock ID: 1   (ptpd debug1)   14:05:11.840007  (slv)       After PI: Adj: -4319   Drift: -4311   OFM -24
    Clock ID: 1   (ptpd debug2)   14:05:11.840007  (slv)       adjTickRate2: call adjTickRate to 10.055955 PPM
    Clock ID: 1   (ptpd debug3)   14:05:11.840007  (slv)
--Offset Correction--
    Clock ID: 1   (ptpd debug3)   14:05:11.840008  (slv)  Raw offset from master:           0s         -25ns
    Clock ID: 1   (ptpd debug3)   14:05:11.840008  (slv)
--Offset and Delay filtered--
    Clock ID: 1   (ptpd debug3)   14:05:11.840008  (slv)  one-way delay averaged (P2P):           0s         120ns
    Clock ID: 1   (ptpd debug3)   14:05:11.840008  (slv)  offset from master:               0s         -24ns
    Clock ID: 1   (ptpd debug3)   14:05:11.840008  (slv)  observed drift:         -4311,     -10.037329 PPM
    Clock ID: 1   (ptpd debug3)   14:05:11.840008  (slv)  activity
 PTP Slave, Clock ID 1, reach synchronized convergence state,  Stop!

Info: /OSCI/SystemC: Simulation stopped by user.
```

Figure 3

In the non-SyncE state, the simulation results indicate that the Protocol and Clock Servo algorithm require over 15 seconds to synchronize the PTP Slave and PTP Master. The offsetFromMaster is around 20 ns, and the observed Slave-Master frequency difference is approximately -10.03 PPM. Both the convergence speed and synchronization precision are inferior to those in the SyncE state.

On the author's computer, the entire non-SyncE PTPv2 Protocol and algorithm simulation took over six hours to complete. Running such simulations requires patience.