libPLN + libPTP
Usage on Windows

1 Document version history

Version	Date	Author
0.01-very-pre-alpha	August 2015	Wolfgang Wallner

2 Goal

We would like to evaluate the PTP protocol with network simulation in OMNeT++. To simulate PTP in OMNeT, it is useful to have realistic oscillator noise. Neither of these to things are readily available in standard OMNeT or INET. This is solved by the following two projects:

• libPLN:	Provides realistic oscillator noise
• libPTP:	Provides a PTP implementation in OMNeT++

3 Introduction

LibPLN is a C++ library to simulate **P**ower**L**aw **N**oise. This is the type of random noise from whic oscillators suffer. The basic noise generation principle is based on a batch method which was developed by Kasdin and Walter.

LibPTP is an OMNeT++ libary to simulate **P**recision **T**ime **P**rotocol relevant hardware. It contains simulation modules for clocks, and these in turn can use the functions provided by libPLN.

4 Project relationships

Both libPLN as well as libPTP use the open source Boost C++ library. LibPLN additionally uses the FFTW library for its FFT implementation. As from the Boost library only header-only modules are used, this library does not need to be compiled manually.

The library libPLN only provides the necessary entities to generate powerlaw noise, but no actual instances where they are used. The goal of the library libPLN_Examples is to provide such implementations. Currently it only supports an example of an arbitrary oscillator.

LibPLN comes also with a small utility called TestBench, that uses both libPLN and libPLN_Examples and can be used to find simple errors.

LibPTP can be configured to use libPLN. In this case, libPTP obviously depends on libPLN.

Any OMNeT++ project that used PTP hardware from libPTP, e.g. the PTP Simulations project, depends on libPTP. If libPTP was configured to used libPLN, such projects need to link the necessary libraries: FFTW, libPLN and libPLN_Examples.

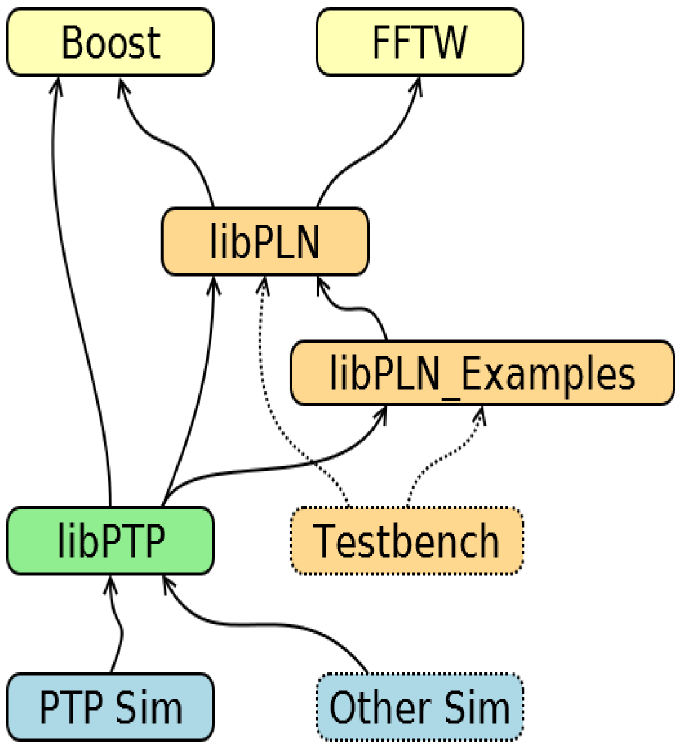

Illustration 1: Project relationships

5 Compiling libPLN

Required steps

- Install Toolchain
- Install external libraries
- Build libPLN
- Build libPLN_Examples
- Using libPLN

Install Toolchain

• Cmake for Windows	http://www.cmake.org/download/
• OMNeT++	https://omnetpp.org/omnetpp

Install external libraries

• Get the precompiled FFTW lib (32-bit)	http://www.fftw.org/install/windows.html
• Get the Boost C++ libary	http://www.boost.org/users/download/

   - The boost library does not ned to be compiled, as only header-only modules are used

Building libPLN and libPLN_Examples

- Check out libPLN repository
- Create build directory
  - e.g. `<libPLN-Dir>/build`
- Launch the OMNeT++ version of MinGW
  - `<OMNeT-Dir>/tools/win32/mingw32.bat`
- Change to the build directory of libPLN
- Call Cmake
  - e.g. `cmake-gui ..`
- Select to generate UNIX Makefiles
- Configure the project
  - On Windows, the libPLN CMake project requires you to set the paths for the following libraries:
    - FFTW e.g. `c:\FFTW`
    - Boost e.g. `c:\boost`
- Generate the project files
- Compile and install the libPLN libraries
  - `make install`
  - This generates the static libraries in `<libPLN-Dir>/lib/static`
- To make the libPLN DLLs available to other projects, you have to either
  - Copy the compiled DLLs to the installation folder of your program
  - Set the global **PATH** variable to the installation directory of the libPLN libraries
- **[optional]** Compile the TestBench to test libPLN
  - `make TestBench`
  - This generates `<libPLN-Dir>/Tests/TestBench.exe`
  - Calling it without arguments lists available arguments
  - This testbench is only for basic tests during development, and is thus mainly undocumented and may change at any time

6 PTP Library: libPTP

Required steps

- Configure Includes, Library paths and Libraries
- Configuring libPLN usage
- Building the simulation library

Configuring paths

LibPTP needs the Include and Library paths for libPLN, FFTW and Boost

There are two alternatives to specify the Include and Library paths:

- In the OMNeT project settings:
  - Includes: **`Properties → C/C++ General → Paths and Symbols → Includes → C++`**
  - Library Paths: **`Properties → C/C++ General → Paths and Symbols → Library Paths`**
  - Libraries: **`Properties → OMNeT++ → Makemake → src → Options → Link → More`**

- In the file **`src/makefrag`**:
  - Add the paths to the `INCLUDE` and `LIBS` variables, e.g.:
    - **`INCLUDE_PATH += -I"C:/FFTW32"`**
    - **`LIBS += -L"C:/FFTW32"`**
    - **`LIBS += -lPLN_Examples -lPLN -lfftw3-3`**

Configuring libPLN usage

To enable the usage of the libPLN library in the libPTP simulation models, the `HAS_LIBPLN` macro needs to be defined.

- In the OMNeT project settings:
  - Go to: **`Properties → C/C++ General → Paths and Symbols → Symbols → C++`**
  - Define **`HAS_LIBPLN`**

Building the simulation library

Once the paths are correctly set, it should be possible to compile the libPTP simulation library as usual.

7 PTP_Simulations

Required steps

- Configure Library paths and Libraries
- Building the simulation library
- Supplying the FFTW DLL

Configuring paths

- The library paths and the libraries need to be configued similar as for the libPLN configuration

Supplying the FFTW DLL

- The FFTW library is a DLL
  - Either copy the needed DLL into the folder of the simulation executable, or
  - add the FFTW directory to system-wide PATH variable

Running a simulation

- Run the Simulation as usual

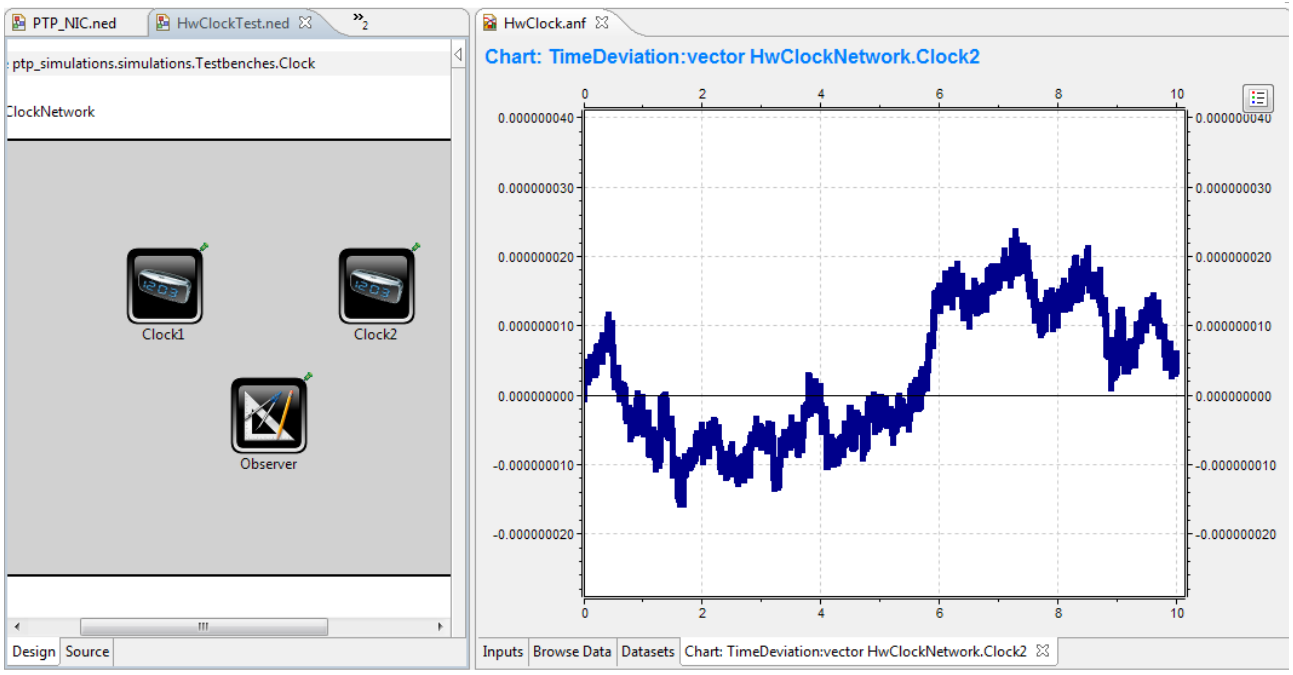

Illustration 2: OMNeT++ Screenshot showing a trace of oscillator noise