OpenEmulator 使用手册

（版本 1.0）

OpenTSN 开源项目组

2022 年 6 月

目录

一、概述.....2
1.1 图形化操作.....2
1.2 灵活构建和运行 TSN 应用.....3
1.3 支持多种网络拓扑.....3
二、运行环境.....4
三、快速运行示例.....5
3.1 IP 核配置.....5
3.2 运行.....5
3.2.1 检查可执行文件权限.....5
3.2.2 运行图形化程序.....5
四、详细使用说明.....6
4.1 软件程序编译.....7
4.1.1 依赖库安装.....7
4.1.2 执行编译.....7
4.2 应用程序接口.....7
4.2.1 接口定义.....7
4.2.2 接口参数说明.....9
4.2.3 接口使用及程序编译.....10
4.3 OpenEmulator 图形界面使用.....10
4.3.1 动态链接库编译.....10
4.3.2 依赖库及安装说明.....10
4.3.3 使用说明.....10

一、概述

TSN 芯片验证是 TSN 芯片设计和实现过程中的重要一环，为推动 TSN 技术的快速发展，一种便捷高效的 TSN 芯片验证环境不可或缺。基于专用硬件仿真器的芯片验证环境通常面向 SoC（System-on-Chip）这类设计非常复杂的芯片。相比于 SoC，TSN 芯片逻辑相对简单，芯片验证环节对验证环境的性能要求相对较低，通用计算机已能满足 TSN 芯片验证的需求。此外，TSN 芯片验证需要与软件进行协同验证调试，而且还需要实现软件与硬件的时间同步。

基于上述背景，OpenTSN 项目组基于通用计算机以及 HDL 仿真软件搭建了面向 TSN 芯片的联合仿真验证平台，可与真实的软件程序进行联合验证调试。并具有以下几个特性：

1.1 图形化操作

OpenEmulator 支持图形化操作，可一键启动和停止网络以及真实软件程序（包括集中控制器 TSNLight 以及时钟同步控制程序 PTP），如图 1 所示。

OpenEmulator 可实时展示仿真进度，包括网络仿真的时长（Simulation Time）、墙钟时间（Wall Clock）以及两者的比值——时间比例尺（Time Scale，用于评估平台的运行性能）。

图 1 图形化仿真控制程序

1.2 灵活构建和运行 TSN 应用

OpenEmulator 支持运行集中控制器 TSNlight 以及基于 PTP 协议的时钟同步控制程序，并为用户提供了应用程序接口，以支持运行用户实现的 TSN 应用程序。

1.3 支持多种网络拓扑

OpenEmulator 内置了 6 节点的哑铃型网络拓扑，如图 2。同时支持通过修改 TSN 端系统和交换机连接方式实现自定义的网络拓扑。

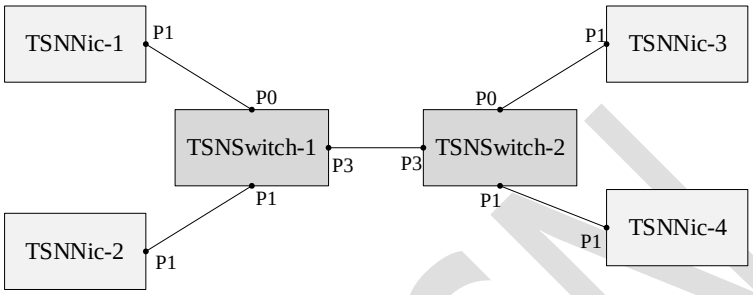

图 2 节点哑铃型网络拓扑

二、运行环境

- Linux 系统：在 CentOS6.9 以及 CentOS7 上通过测试；
- HDL 仿真软件：用于网络仿真；
- C 语言开发环境（GCC/G++）：用于硬件代码和软件程序的编译，在 GCC/G++4 上通过测试。

三、快速运行示例

本节介绍如何通过 OpenEmulator 快速运行基于 OpenTSN3.4 的组网示例。组网示例位于 OpenEmulator/demo 目录下。目录结构说明如下所示：

```
OpenEmualtor
    ├── demo      //软硬件联合仿真组网示例，组网示例环境使用参考
    ├── doc       //设计文档、环境安装、使用手册等文档
    ├── lib       //依赖的库文件说明
    ├── script    //OpenEmulator 可执行程序及源码
    └── src       //仿真平台源码
```

3.1 IP 核配置

在以下路径中需要依赖 IP 核，根据情况按照路径下的 README 文件说明进行配置。需要配置 IP 核的路径有以下：

- FPGA 器件依赖库文件：OpenEmulator\demo\libs
- 网卡逻辑 IP 核文件：

  opentsn3.4\HARDWARE\script\TSNNic3.4_FPGA_2port\ipcore\
- 交换机逻辑 IP 核文件：

  open-tsn3.4\HARDWARE\script\TSNSwitch3.4_FPGA_4port\ipcore\

3.2 运行

3.2.1 检查可执行文件权限

表 1 路径及对应的可执行文件名称

序号	路径	可执行文件名称
1	OpenEmulator/src/emulation_lock	interlock
2	OpenEmulator/demo/tsn_applications/ptp	ptp_app
3	OpenEmulator/demo/tsn_applications/ptp_bc	ptp_app
4	OpenEmulator/demo/tsn_applications/tsnlight	tsnlight
5	OpenEmulator/script	OpenEmulator

分别进入表 1 所示的路径，检查对应的可执行文件在当前用户下是否有执行权限。若没有执行权限，需要使用“chmod 755 [文件名称]”命令修改可执行文件的权限，如进入“OpenEmulator/src/emulation_lock”目录，执行“chmod 755 interlock”，以获取可执行文件 interlock 的执行权限。

3.2.2 运行图形化程序

进入路径：OpenEmulator/script，执行命令：./OpenEmulator，运行后出现如图 3 所示界面。

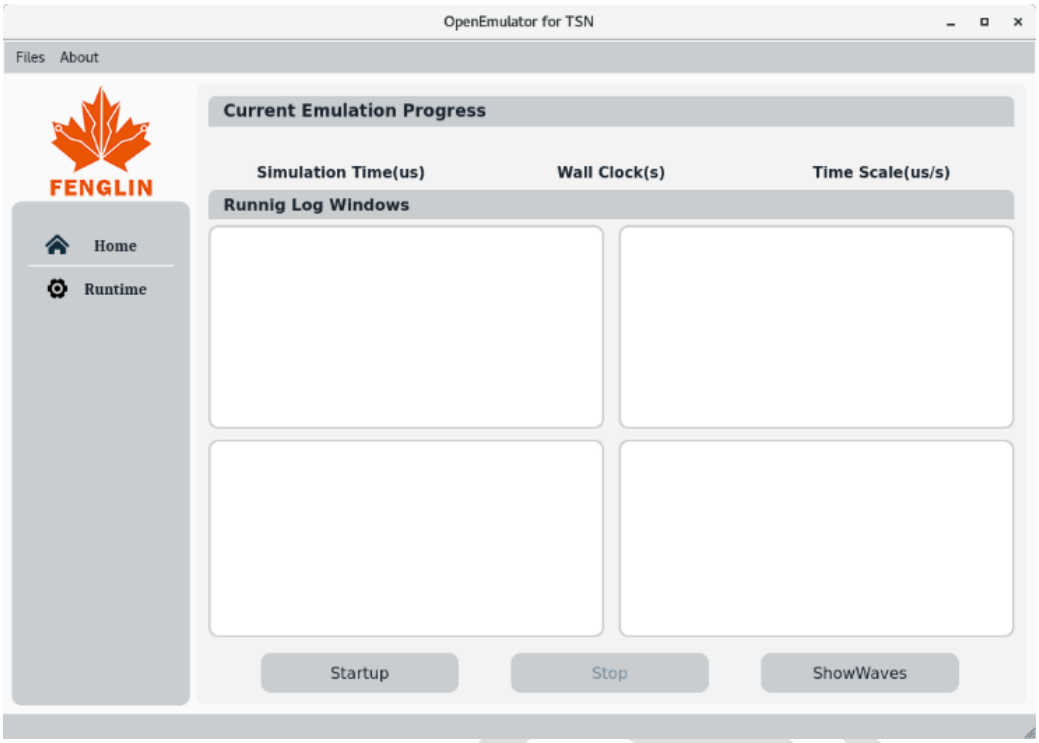

图 3 OpenEmulator 启动界面

点击“Startup”，在图 4 所示的弹窗中输入需要仿真的时长（ms 为单位），点击“Confirm”后，网络仿真开始执行，在运行日志窗口（Running Log Windows）中会输出调试信息，如图 1 所示。四个窗口分别对应：HDL 仿真软件日志窗口、TSNlight 日志窗口、PTP_BC 日志窗口和 PTP 窗口。

当网络仿真的时长（Simulation Time）达到设置的时间时，仿真会自动停止，或通过手动点击“Stop”也可提前停止仿真。停止后，点击“ShowWaves”可调用第三方程序查看仿真过程中的波形。

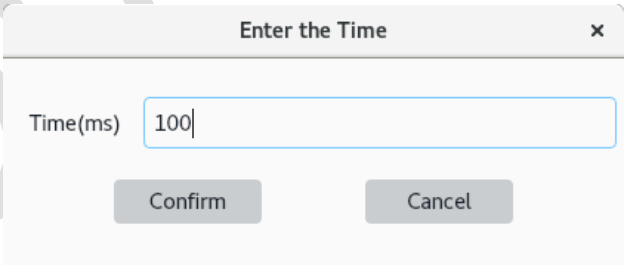

图 4 输入需要仿真的时长（ms 为单位）

四、详细使用说明

本节对 OpenEmulator 的使用进行详细说明，包括用于仿真验证的软件程序（TSNlight 时钟同步控制程序以及用于联合仿真中软硬件时间同步的时间互锁程序）的编译、仿真验证环境应用程序接口使用说明、OpenEmulator 图形界面的使用。

4.1 软件程序编译

4.1.1 依赖库安装

TSNlight 和基于 PTP 的时钟同步控制程序的开发基于 libxml2-dev 库，程序编译需要安装 libxml2-dev 库。

在 CentOS 下，libxml2-dev 可通过以下命令安装：

yum install libxml2-devel

ln -s /usr/include/libxml2/libxml /usr/include/libxml

4.1.2 执行编译

（1）TSNlight 程序编译

进入路径：OpenEmulator\demo\tsn_applications\tsnlight

执行命令：make clean && make TYPE=TSN_SIM

（2）时钟同步控制程序编译

时钟同步控制程序分为两部分。

进入路径：OpenEmulator\demo\tsn_applications\ptp

执行命令：make clean && make TYPE=TSN_SIM

进入路径：OpenEmulator\demo\tsn_applications\ptp_bc

执行命令：make clean && make TYPE=TSN_SIM

（3）时间互锁程序编译

时间互锁程序编译依赖于联合仿真环境提供的静态链接库，静态链接库编译方法：

进入路径：OpenEmulator\lib

执行命令：make clean && make static

执行该命令后生成 libsim.a 文件，进一步编译时间互锁程序：

进入路径：OpenEmulator\src\emulation_lock

执行命令：make clean && make

4.2 应用程序接口

4.2.1 接口定义

为支持用户自定义的 TSN 应用程序，如周期性流量注入应用，OpenEmulator 可以静态链接库(libsim.a)或动态链接库（libsim.so）的形式提供了应用程序接口：

(1)仿真网络时间获取接口

仿真网络时间获取接口可用于应用程序控制应用发送的周期，如时钟同步控制程序的同步报文发送周期、时间敏感流量报文周期等。

函数定义	struct timeval gettimeofdaytxt(u8* txtpath)

功能描述	获取仿真网络时间
输入参数	软件时间文本路径
输出参数	若成功则返回仿真时间，若失败则返回 0

（2）报文接收接口

接收报文接口包括数据报文接收初始化接口(data_pkt_receive_init)、数据报文接收处理接口(data_pkt_receive_dispatch_1)和数据报文接收销毁接口(data_pkt_receive_destroy)。

①数据报文接收初始化接口(data_pkt_receive_init)：

函数定义	int data_pkt_receive_init(u8* txtpath)
功能描述	完成数据报文接收资源的初始化。包括初始化 buff 空间、打开文本等
输入参数	报文接收文本路径
返回结果	成功返回 0，失败返回-1

②数据报文接收处理接口(data_pkt_receive_dispatch_1)：

函数定义	u8 *data_pkt_receive_dispatch_1(u16 *pkt_len);
功能描述	每次从文本中读取一个报文，然后对报文字符进行格式处理，再输出报文首地址指针和报文长度
输入参数	报文长度指针
返回结果	成功返回报文首地址指针，失败返回 NULL

③数据报文接收销毁接口(data_pkt_receive_destroy)：

函数定义	int data_pkt_receive_destroy ( )
功能描述	完成数据报文接收资源的销毁，包括关闭文本句柄，释放 buff 空间等
输入参数	无
返回结果	成功返回 0，失败返回-1

(3）发送报文接口

发送报文接口主要包括数据报文发送初始化接口(data_pkt_send_init)、数据报文发送处理接口(data_pkt_send_handle)和数据报文发送销毁接口(data_pkt_send_destroy)。

①数据报文发送初始化接口(data_pkt_send_init)：

函数定义	int data_pkt_send_init(u8* txtfile_data,u8* txtfile_state)
功能描述	完成数据报文发送资源的初始化。包括初始化 buff 空间、打开文本句柄等。
输入参数	报文发送文本路径，发送状态文本路径
返回结果	成功返回 0，失败返回-1

②数据报文发送处理接口(data_pkt_send_handle)：

函数定义	int data_pkt_send_handle(u8* pkt,u32 len)
功能描述	先对字符串进行文本格式的转换，然后再将数据写入文本，完成数据报文的发送，并向发送状态文本中写入状态标志字符
输入参数	数据报文指针、数据报文长度
返回结果	成功返回 0，失败返回-1
功能描述	完成数据报文的发送处理

③数据报文发送销毁接口(data_pkt_send_destroy)：

函数定义	int data_pkt_send_destroy()
功能描述	完成数据报文发送相关资源的销毁，包括关闭文本句柄、释放 buff 空间等
输入参数	无
返回结果	成功返回 0，失败返回-1

4.2.2 接口参数说明

在调用报文接收和报文使用接口时，在 6 节点哑铃型拓扑网络下规定了每个 TSN 网卡使用的报文交互文件，这些文件作为报文发送和接收接口的参数，具体每个 TSN 网卡对应的报文交互文件作为参数的使用说明如下（以 TSNNic-1 为例）：

接口名称	参数名称	参数值
data_pkt_receive_init	txtpath	data014.txt
data_pkt_send_init	txtfile_data	data114.txt
	txtfile_state	data214.txt

其他网卡（TSNNic-2、TSNNic-3、TSNNic-4）对应的交互文件分别为：data015/data115/data215、data016/data116/data216、data017/data117/data217，使用方法同 TSNNic-1。

提示：目前软硬件交互接口支持 data010~ data017 一共 8 个交互接口，通过报文交互接口文件可获取仿真过程中的报文内容。

4.2.3 接口使用及程序编译

1.引入头文件

在程序头部引入 OpenEmulator\lib 目录下的 sim.h 文件

2.加入编译选项

GCC 编译时加入以下选项：-L/path/of/lib -lsim（/path/of/lib 是指 libsim.a 文件所在的目录）

4.3 OpenEmulator 图形界面使用

4.3.1 动态链接库编译

OpenEmulator 图形界面依赖 libsim 动态链接库，调用了动态库中定义的获取仿真网络时间的接口，编译方法如下:

进入路径：OpenEmulator\lib

执行命令：make clean && make dynamic

执行以上命令后目录下生成 libsim.so 文件

4.3.2 依赖库及安装说明

图形界面程序源码位于 OpenEmulator/script 目录下，基于 Python3 以及以下依赖库实现：

- PyQt5：使用 Qt5 图形库构建图形界面程序；
- pyqt5-tools：用于绘制图形界面（pyqt5-tools designer）并转换位 Python 源文件（pyuic5）；
- qdarkstyle：用于图形界面美化；
- pyinstaller：将图形界面程序打包为一个可执行文件。打包脚本为 OpenEmulator/script 目录下的 installer.sh 文件，可通过命令行执行该脚本实现程序打包。

以上依赖库可通过“pip install”命令安装，如“pip install PyQt5”可安装 PyQt5。

4.3.3 使用说明

OpenEmulator 图形界面程序依赖于相对路径，可通过在 script 目录下执行以下相应命令启动。**未安装以上的依赖库情况下执行：**./OpenEmulator；**安装依赖库的情况下执行：**python ./main.py

（1）快速运行示例

启动后的界面如图 3 所示，根据第三节的描述可快速运行示例仿真网络。

（2）修改程序路径

OpenEmulator 对调用的外部程序进行了默认配置，点击图 5 所示侧边栏中的“Runtime”可修改程序运行路径。

Emulation Process、Interlock Process、TSNLight Process、PTP Boundary Clock Process、PTP Process 和 Show Waves 分别代表：仿真程序、互锁程序、集中控制器和两个基于 PTP 协议的时钟同步控制程序以及调用的展示波形的外部程序。

每个程序可配置所在路径以及运行命令（分别对应 1、3），通过修改输入框内容修改程序所在路径以及运行命令，或可点击 2 对应的按钮快速选择路径。

修改完成后点击“Save”按钮更新程序的路径和命令，此时点击“Startup”即可按修改后的配置运行。

图 5 程序路径和执行命令配置