FAST-ANT 2.0网络测试与分析套件

|  |  |  |  |
| --- | --- | --- | --- |
| 文档版本号 | 修改人 | 修改时间 | 备注 |
| 2.0 | 蒋越 | 2019.05.23 | FAST-ANT 2.0初步设计 |
| 2.1 | 蒋越、吴尚明 | 2019.05.26 | 修改硬件架构设计、基本功能设计 |
|  |  |  |  |
|  |  |  |  |

FAST-ANT（Agile Network Tester）作为一款基于FPGA/CPU的轻量级网络测试分析器，支持针对吞吐、延迟等相关指标的精确测量。其功能可定制、轻量级、低成本的特点，适合中小型研究机构进行网络性能与功能的测试与分析。

FAST-ANT 1.0将支持的基本功能有：

1. 测量设备对于不同大小、不同协议类型报文的精确/粗略时延；
2. 测量设备对于不同大小、不同协议类型报文的吞吐率；
3. 测量设备对于不同大小、不同协议类型报文的丢包率；
4. 数据统计功能。

但随着实际应用场景复杂化、多样化，FAST-ANT 1.0所提供的测试功能已经难以满足针对新的协议或功能的全面测试，如：并发的多条多类型流、流量回放、更高发包速率等。因此，基于上述现状，我们将设计并开发FAST-ANT 2.0来满足新的功能需求和性能指标。

FAST-ANT 2.0将基于FAST-ANT 1.0的基础上，优化重构软硬件设计实现，并在OpenBox-S4上先实现原型系统，后移植至NUDT-NSG团队自主开发的40G NP上。

# 1 引言

FAST-ANT 1.0采用FPGA/CPU的软硬件协同设计思想，充分发挥软件在灵活可定制和硬件在高性能精度上的优势，支持针对吞吐、延迟等指标的高精度测量。目前，FAST-ANT 1.0被广泛应用于测量网络设备（如：交换机）的转发速率和处理延迟，以及构造背景流量。FAST-ANT 1.0已经成功支撑不同设备、算法、协议的测试和验证工作，具有较高的应用价值和意义。

如图1所示，在FAST-ANT 1.0的硬件设计中，我们在FAST 2.0的五级流水线基础上，扩充两个功能模块PGM（报文生成模块）和SCM（统计收集模块）构成七级单流水线，提供了通用的高性能精度的报文发送和收集功能。



图1 FAST-ANT 1.0硬件设计

如图2所示，在软件设计中，在FAST API的基础上，向用户封装底层实现细节，支持用户通过配置相关测试参数，如：发送速率、发送端口等，来控制流量发送的模式；如：报文大小、内容等，来构造不同类型的流量；如：捕获报文类型，来收集统计特定类型报文的相关测试数据信息。

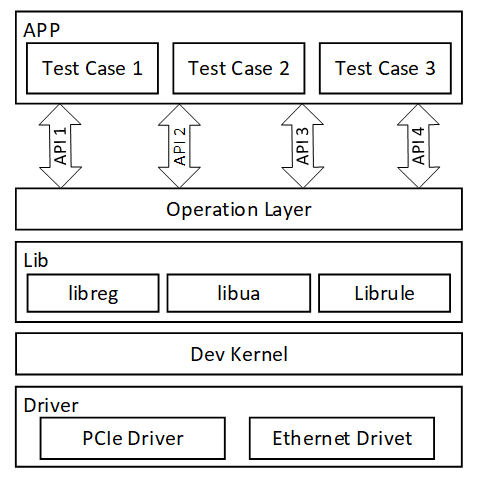


图2 FAST-ANT 1.0软件设计

在FAST-ANT 1.0中，采用的是FAST五级流水线的设计思想，仅支持发送单条特定类型的流量，并只能对指定类型的报文进行简单的数据统计。但随着测试场景的复杂化、待测协议和功能多样化，FAST-ANT 1.0已经难以满足新的测试需求。

首先，难以模拟复杂的测试场景，无法支持不同类型的流量并发存在。并且，发送流量的模式单一，仅支持对发送速率的控制；

其次，对于流量的捕获统计只是简单地统计某一指定类型的报文个数，缺乏对捕获报文的保存来支持后续的流量重放；

最后，FAST-ANT 1.0仅支持1Gbps的报文发送速率，在性能方面有所限制，需要进一步提高整体性能。

因此，基于以上分析，我们将开发FAST-ANT 2.0，以更好地支持满足不同测试功能和测试指标。

# 2 测试功能

## 2.1 多类型流量并发

FAST-ANT 2.0扩充的第一个功能是支持并发的多条多类型流量。FAST-ANT在测试中常常被用于构造测试环境的背景流量，例如：模拟流量爆发，产生一条高速率流量来测试网络转发设备的丢包率、最大吞吐或时延等，如图3所示。



图3 针对网络设备的单/双端口测试

但是在一些复杂的测试场景中，例如：TSN，需要产生复杂的背景流量，这样的背景流量中可能包含了不同优先级和不同类型的流，如图4所示。FAST-ANT需要向待测设备生成一条多类型流量并发，并且需要按照测试场景的需求来设定流量的发送模式、速率等多项参数。



图4 复杂背景流测试场景

该功能的思路是：用户可以在Linux端通过构造不同的测试报文（报文类型、大小），配置报文发送的相关参数（速率）以及设定报文发送模式（乒乓报文、周期报文），同时指定并发流的发送端口。在FPGA上根据Linux端用户的设定来实现流量的精确、高性能发送，生成符合测试场景需求的复杂背景流量，满足并发、多协议的要求。

## 2.2 流量回放

在FAST-ANT 1.0中，对于捕获到的报文，我们仅统计报文个数、吞吐量以及计算指定类型报文的延迟。当获取了以上统计数据后，捕获到的报文将被丢弃。但在某些测试场景中，例如：在相同测试场景中对比不同算法的效果或不同设备的性能，需要保存某一时间段内所捕获到的所有报文并再次使用；或是在测试结束后，对某一时段产生的流量进行进一步分析来获取网络的某一时段状态。

因此，在FAST-ANT 2.0中，我们将支持流量回放。该功能的思路是：用户需要在Linux端设定捕获报文类型、捕获时间段或捕获报文个数；在FPGA中根据设定的捕获参数，将捕获的报文存储在硬件上，允许用户在Linux端读取存储的报文生成可视化结果并用于后续的分析。

## 2.3 FAST-ANT 1.0测试功能

FAST-ANT 2.0将继续支持FAST-ANT 1.0中所提供的测试功能，包括：时延、吞吐量等相关测试功能和内容。详细功能描述见FAST-ANT 1.0设计文档。

# 3 总体架构

FAST-ANT 2.0基于FAST平台设计开发，分别在软硬件层面借助了FAST所提供的数据结构、接口函数以及FPGA OS和UM中的部分代码（根据扩充功能去掉UM流水线中部分不需要的功能模块）。原型系统在OpenBox-S4上进行开发实现。

本章首先介绍FAST-ANT 2.0的软件架构设计。接着，介绍FAST-ANT 2.0的硬件架构设计。最后，以多类型流量并发和流量回放这两个测试功能为例，详述FAST-ANT 2.0的功能运行流程。

## 3.1 FAST-ANT 2.0软件架构

FAST-ANT 2.0软件端负责接收用户在可视化界面上选定的测试功能和配置的测试参数，并在测试完成后以图表和数据文字相结合的方式展示相应的测试结果。其整体结构如图5所示。



图5 FAST-ANT 2.0软件架构图

FAST-ANT 2.0的软件端部分基于FAST库所提供的API和数据结构，通过Linux NIC同硬件中LCM模块进行消息交互来完成测试模式和测试参数的配置工作。在继承了FAST-ANT 1.0的基础上，将软件端从ARM移植至标准Linux机器上，并增加可视化界面，简化测试仪的使用流程，提升用户体验。同时，软件端在标准的Linux机器上实现便于用户进行移植和扩展。

首先，用户在提供的FAST-ANT GUI界面中选定测试功能并配置相对应的测试参数。GUI中的输入值将传递到FAST-ANT OS层，并映射到对应的FAST数据结构中；接着，将调用相应的FAST API和库，将相关测试参数配置到FPGA中并运行相应的测试功能；最后，FAST-ANT将同样通过调用相应的FAST API读取相关FPGA内寄存器内和存储内存上的数据，并在FAST-ANT OS层经过处理后以图标和数据文字的形式反馈至GUI界面，形成最终的测试结果的展示。

## 3.2 FAST-ANT 2.0硬件架构

FAST-ANT 2.0的硬件部分通过对FAST流水线进行扩展修改实现，其整体架构如图6所示。为了支持多类型流量并发和流量回放功能，FAST-ANT 2.0在FAST-ANT 1.0的基础上增加了LCM（Local Control Module）、OSM（Output Schedule Module）和UDC（User Define Counter）。同时，为了精简FAST流水线的设计，我们去掉了功能无关的原五个基础功能模块：GPP，GKE，GME，GAC以及GME。



图6 FAST-ANT 2.0硬件架构图

其中，LCM是整个FAST-ANT 2.0的核心模块，负责根据报文类型调度分派到不同硬件功能模块以及读写硬件功能模块中寄存器并同软件Linux端通信。为了满足硬件FPGA设计中双功能并行的需求，原FAST 2.0中的环形控制通路不再适用，转而由LCM模块作为硬件FPGA中的“中枢大脑”进行调度控制。每一个硬件功能模块中将读寄存器作为输出（Output），将写寄存器作为输入（Input），同LCM进行连接。这样，LCM可以实时获取不同模块内寄存器的值的情况。同时，LCM内将实现报文上送机制，周期性向软件Linux端上送硬件寄存器信息。

PGM和OSM两个功能模块作为多类型流量并发的核心组成部分，分别负责根据软件端用户配置参数生成流量和多条流并发的调度。PGM模块接收来自在LCM分派的构造报文和配置参数（例如：发送速率），写入模块相应的存储空间并修改相应寄存器值。当产生流量时，同时读取每个类型的报文并按照配置参数发送到OSM模块；OSM模块根据所选择的发送模式和配置的发送参数（例如：发送时长），调度不同类型流量的转发，实现多类型流量的并发操作。

SCM和UDC两个功能模块作为流量回放的核心组成部分，分别负责统计特定类型流量的测试数据（包括：吞吐量、延迟等）和根据用户需求存储报文。SCM模块根据LCM来修改自身内部的相关寄存器值，包括：流量类型等。对于流入的流量，统计符合类型的流量的相关数值并写入对应的寄存器。接着，反馈至LCM并上送至软件端；UDC模块同样根据LCM来修改自身内部的相关寄存器值，包括：统计类型、统计时长以及统计数量等。对于经过UDC的流量，根据统计要求将符合的报文存储至DDR 3上。待测试结束后，软件端可以从DDR 3存储中将相应的报文读出，用于后续的分析。

## 3.3 基本功能设计

FAST-ANT 2.0在FAST-ANT 1.0的基础上，根据新的测试需求，增加了多类型流量并发和流量回放两个核心功能。本节将以这两个功能为例，从运行流程的角度说明FAST-ANT 2.0的运行流程，并介绍内部运行机制。

### 3.3.1 多类型流量并发

多类型流量并发功能的运行流程如图7所示。



图7 多类型流量并发功能的运行流程图

用户首先在Linux端的图形化界面上进行参数配置，并将配置信息以报文的形式发送到LCM模块。LCM模块接收到报文后，首先判断该报文类型是否属于本功能的报文。然后解析报文内容，如果属于修改寄存器值来配置发送参数（如：发送速率、发送时长等），则直接修改对应模块内的寄存器；如果属于用户构造的不同类型报文，则传递至PGM并保存在不同的存储空间。当开始生成流量后，则同时读取存储的不同类型报文，并按照配置的参数生成流量并发送。在OSM模块中对不同流进行调度来实现多条流的并发操作。图8展示了本功能在硬件FPGA中的示意图。



图8 多类型流量并发功能在硬件FPGA中的示意图

### 3.2.2 流量回放

流量回放功能的运行流程如图9所示。



图9 流量回放功能的运行流程图

用户首先在Linux端的图形化界面上进行参数配置，并将配置并将配置信息以报文的形式发送到LCM模块。LCM模块接收到报文后，首先判断该报文类型是否属于本功能的报文。然后解析报文内容，如果属于修改寄存器值来配置统计参数（如：流量类型、统计时长、统计个数等），则直接修改对应模块内的寄存器；如果是进入FAST-ANT的测试流量，则先在SCM模块中对特定协议类型的流量进行统计，然后传递至下一模块。对于不符合协议类型的则直接传递至下一模块。在UDC模块中根据用户需求存储报文至DDR 3存储上，并在测试结束后允许软件端读取报文，用于后续分析。LCM模块将会定期向软件端发送报文，反馈相关寄存器内的统计数据。图10展示了本功能在硬件FPGA中的示意图。



图10 流量回放功能在硬件FPGA中的示意图

# 4 FAST-ANT 2.0详细设计

# 5 进一步工作与开发计划

## 5.1 进一步工作

1. 在OpenBox-S4上实现FAST-ANT 2.0原型系统后，移植至40G NP并测试相关功能是否正常、相关指标是否满足；
2. 在40G NP上进行硬件功能模块的细化，充分发挥不同FPGA的优势，将高性能和精度相关的功能模块移植到Altera FPGA上，将发送控制模块移植到Xilinx FPGA上；
3. 关于FAST架构的思考：在FPGA OS提供类似于OSNT的Wrapper，支持端口和流水线之间的自定义映射（多对多映射关系）。UM部分只给出标准的数据帧的位宽，而数据帧的内容格式是如何定义的则完全交给开发者自定义。这样，FAST将UM部分进一步开放，开发者可以在UM内实现更多自定义的网络功能，无需遵循之前传统的五级流水线设计，简化了UM内部的结构。而多对多映射的实现可以支持多条流水线并行处理，也可以支持开发者对硬件调度的相关研究和开发；
4. 关于FAST-ANT的思考：随着测试场景和需求的不断复杂化和多样化，测试功能也在不断丰富扩充。部分功能为了充分利用硬件的特性，例如：偏向于最大化发挥精度或性能，而通用的架构则不再适用，定制化的内部架构更是一种趋势。硬件FPGA可重构化则是一种发展趋势。针对不同的硬件功能生成对应的硬件烧录温江，软件端则根据用户测试需求，编写加载脚本，支持动态重构FPGA，真正实现软件配置、硬件可重构的测试架构。

## 5.2 开发计划

### 5.2.1 项目人员

孙志刚，全巍，徐东来，杨翔瑞，蒋越，吴尚明。

### 5.2.2 项目分工

1. ***设计文档指导***

孙志刚，全巍，杨翔瑞；

1. ***项目开发指导***

徐东来，杨翔瑞；

1. ***软件开发***

蒋越：

1. Linux端和硬件FPGA的LCM模块之间的通信；
2. 图形化界面输入和输出的数据结构。

华芯通：

1. FAST-ANT 2.0软件端图形化界面。
2. ***硬件开发***

蒋越：

1. LCM模块设计实现，定义数据帧格式，实现定期上送软件端机制；
2. OSM调度机制设计。

吴尚明：

1. PGM模块重写，适配新的硬件架构；
2. SCM模块重新设计实现，增加同LCM通信，增加新的统计功能；
3. UDC模块设计实现；
4. OSM调度机制实现。

### 5.2.3 进度安排

|  |  |  |
| --- | --- | --- |
| 日期 | 工作内容 | 参与人员 |
| 2019.05.27 | 针对设计文档2.0进行讨论修改。 | 孙志刚，全巍，徐东来，杨翔瑞，蒋越，吴尚明 |
| 2019.05.28  ~  2019.05.30 | 完成详细设计部分。  软件部分完成核心数据结构、函数设计；  硬件部分完成核心功能模块接口信号和状态机设计。 | 蒋越，吴尚明 |
| 2019.05.31 | 针对设计文档2.1进行讨论修改。 | 孙志刚，全巍，徐东来，杨翔瑞，蒋越，吴尚明 |
| 2019.06.01  ~  2019.06.02 | 完善软件和硬件详细设计部分。 | 蒋越，吴尚明 |
| 2019.06.03  ~  2019.06.16 | 开发硬件功能模块并进行仿真测试。 | 蒋越，吴尚明 |
| 2019.06.17  ~  2019.07.05 | 开发软件部分并测试；  上板联合调试软硬件代码。 | 蒋越，吴尚明，杨翔瑞，徐东来 |