TSNNic 硬件设计文档

（版本 1.13）

OpenTSN

OpenTSN 开源项目组

2021 年 5 月

版本历史

版本	修订时间	修订内容	修订人	文件标识
1.0	2019.11.27	初版编制		OpenTSN1.0
1.0	2019.07.17	1.TSNNic 的概要设计		
1.1	2019.07.18 ~2019.07.21	1.讨论 TSNNic 的整体设计方案和 LCM 模块、PGM 模块、FSM 模块、SSM 模块的详细设计方案		
1.2	2019.07.26	1.完成附录中 Beacon 报文格式设计； 2.分析、设计存储 8 个报文头的 RAM（PKT_HDR_RAM）和存储门控列表的 RAM（GCL_RAM）； 3.完成 PGM 模块的概要设计和 TGR、GCM、		

OpenTSN

		TSM、DRM 的 4 个子模块的详细设计方案。		
1.3	2019.07.27	1.完成 PGM 子模块 PHE 的详细设计，PGM 各子模块的接口信号定义		
1.4	2019.07.28	1.在 PGM 设计中增加 UDO 传来的 fifo_usedw 信号； 2.修改 Beacon 报文格式：将 Beacon 上报报文中的门控列表、报文头信息用对应的 Beacon 配置报文的序列号来代替，通过查看该序列号来判断门控列表、报文头是否配置成功；		

OpenTSN

		3.完成 LCM 模块概要设计方案。		
1.5	2019.07.29	1.完成 LCM 模块顶层模块、PHU、LAU 子模块详细设计方案。		
1.6	2019.07.30	1.完成 LCM 模块 ARM 子模块详细设计方案。		
1.7	2019.08.08	1.在两种 Beacon 配置帧中分别添加配置门控列表、命令阵列的报文序列号（4bit）和配置报文头的报文序列号（8bit）； 2.在 LCM 模块中添加测试开始、结束逻辑； 3.在 TGR 模块增加一个 always 块用来判断该类型流量是否有流量		

OpenTSN

		生成请求；在 GCM 模块增加一个 always 块用来对时间槽进行计数；在 TSM 模块增加 in_tsm_fifo_used w 阈值的分析		
1.8	2019.08.09	1.将 TSM 模块的功能改为用状态机来实现； 2.在 PHE 模块不用 fifo 来缓存输入的报文头，改用两个锁存器来暂存报文头数据，并修改 PHE 模块的状态机实现。		
1.9	2019.11.07	1.针对硬件上板调试、软硬件联调、测试 OpenTSN 网络时代码的修改，更		

OpenTSN

		新 LCM 模块、PGM 模块、UM 顶层模块的设计文档。 2.在附录中添加 TSNNic 性能。		
1.10	2019.11.12	1. 在 SSM 模块增加提取报文五元组功能，把提取的五元组放在 metadata1； 2.针对硬件上板调试、软硬件联调、测试 OpenTSN 网络时代码的修改，更新 FSM 模块、SSM 模块的设计文档。		
1.11	2019.11.27	1.完成 TSNNic 硬件需求分析、硬件 UM 总体设计两部分内容； 2.完成附录中的		

OpenTSN

		FPGA 资源评估。		
1.12	2019.12.17	1.修改文档。		
1.13	2021.05.28	1.调整文档格式		

OpenTSN

目录

OpenTSN

OpenTSN

1. TSNNic 硬件需求分析

TSN 网络接口适配器（TSNNic）是应用于 TSN 网络之间的数据适配节点，主要功能是将应用数据按照流量传输需求以及 TSN 网络封装格式注入到 TSN 网络中进行数据传输，以及将从 TSN 网络接收的数据进行封装返回应用程序并进行相关性能统计。具体包括在确定的时间点生成发送时间敏感流，时间敏感流/非时间敏感流的并发，网络流量捕获与分析等。

具体需求如下：

1）多条流并发，并且在测试过程中可动态更新每条流的报文头信息功能；

2）时间感知整形（TAS）功能；

3）带掩码的五元组匹配，统计报文个数功能；

4）分频采样功能；

5）测试被测网络/设备的性能，即测试被测网络/设备对于不同大小、不同类型报文的精确时延，吞吐率，丢包率；

6）具备软件配置能力，用户可根据应用需求模拟不同的流量。

2. TSNNic 硬件总体设计

2.1 TSNNic 平台相关逻辑定制

TSNNic 是在基于 FAST（FPGA Accelated Switching Platform）架构的 OpenBox-S4 上进行开发，FAST 硬件架构原理见附录五。

目前 OpenBox-S4 的标准 FPGA OS 提供了四个千兆接口的输入输出处理逻辑，根据 TSNNic 功能需求，对接口应用做了映射（分别

用于控制、数据输出、数据输入、流量采样），并对 FPGA OS 进行相应剪裁：

1. FPGA OS 在输入侧不做 MUX 操作，直接将来自 CPU 和各端口输入的分组送给 FAST UM；

2. FPGA OS 在输出侧不做 DMUX 操作，为每个输出接口设置单独的信号；

3. 为了支持在 TSNNic 设备内、TSNNic 与 TSNSwitch 设备间进行时间同步，在端口添加透明时间修订机制；

4. 删掉查表引擎功能；

5. 删掉通过 PCIe 总线与 CPU 进行通信的机制。

2.2 TSNNic 平台无关 UM 架构

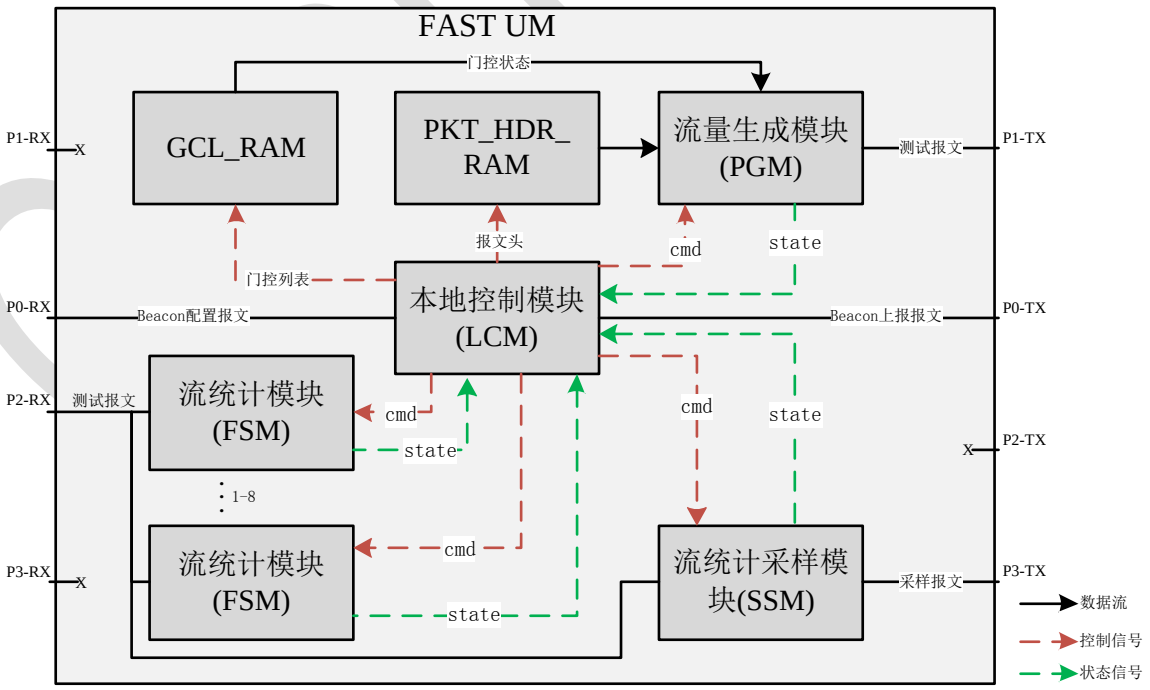


图1　TSNNic 硬件 UM 架构图

注：PN_RX/TX 表示分组从 N 号数据接口输入/输出，N=0、1、2、3；“×”表示在 UM 中针对从相应接口输入/输出的分组未做逻辑处理。4 个数据接口与外部的连接关系见附录四。

TSNNic 硬件 UM 架构如图 2-2-1 所示，UM 中各模块的介绍如下：

LCM（Local Control Module）模块：TSNNic 的本地控制模块。主要负责 FAST UM 各模块控制相关寄存器的更新、状态信息周期性上报。

PGM（Packet Generation Module）模块：TSNNic 的流量生成模块。基于令牌桶机制、时间感知整形（TAS）来实现多条流的生成与发送。

FSM（Flow Statistic Module）模块：TSNNic 的流量统计模块。支持 8 组带掩码的五元组匹配，统计命中的报文个数。

SSM（Statistic and Sample Module）模块：TSNNic 的流量统计与采样模块。统计 TSNNic 总的接收报文个数，并对接收的报文进行解析、提取五元组，按一定的采样频率对接收的报文进行封装（FAST 头）与采样。

GCL_RAM（Gate Control List RAM）模块：门控列表集中缓存模块。用一个 RAM 来缓存门控列表。

PKT_HDR_RAM（Packet Header RAM）模块：报文头集中缓存模块。用一个 RAM 来缓存 2 组 8 种报文头。

UM 整体处理流程如下：

LCM 模块对接收的报文进行解析，若为 Beacon 配置报文（有两种 Beacon 配置报文，一种用来配置 8 种报文头，另一种用来配置门控列表和命令阵列），则将 Beacon 配置报文携带的信息读出，输出

给相应的模块，如图 2-2-1 中红色线条所示；门控列表和命令阵列、报文头依先后顺序配置完成后，LCM 模块给出测试开始信号，PGM 模块开始运转；

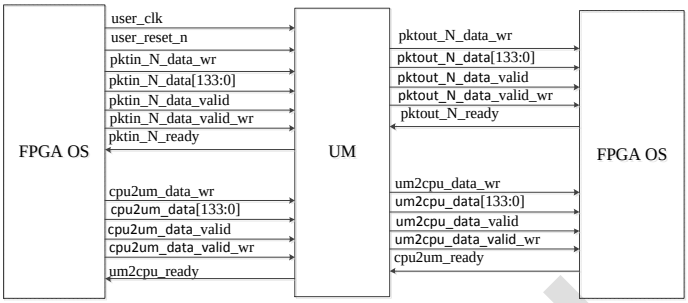
PGM 模块基于令牌桶机制和门控列表生成 8 种并发流量，令牌桶机制用来控制每种流量生成速率，门控列表用来控制每种流量发送的时间；PGM 模块收到测试开始信号后，每隔一个时间槽往令牌桶中注入若干个令牌，对满足调度条件（令牌桶中剩余令牌数≥报文长度（字节），且当前门控状态为开）的报文头按照优先级进行调度，在报文头的基础上根据报文长度进行扩展，生成报文（增加报文发送时间戳、序列号字段）并发送；统计每种报文的发送个数，把统计结果传给 LCM 模块。FSM 模块对接收的报文基于带掩码的五元组进行匹配，统计命中的报文个数，把统计结果传给 LCM 模块。SSM 模块统计接收的报文总个数，把统计结果传给 LCM 模块；并对接收的报文封装一个 FAST 头，提取该报文的五元组放在被封装的 metadata1；按一定的采样频率对接收的报文进行采样，从 3 号接口输出。在测试过程中，可动态更新报文头信息，LCM 模块每隔周期 1s 生成 Beacon 上报报文，上报当前 UM 的状态（如图 2-2-1 中绿色线条所示）；

当 LCM 模块接收到测试停止信号，经过 1s 生成发送最后一个 Beacon 上报报文，然后将寄存器复位信号置高，PGM 模块将每种报文发送个数等清零，FSM 模块将每种报文接收个数等清零，SSM 模块将报文总接收个数等清零。

2.3 FPGA OS 与 UM 模块接口定义

FAST 2.0 规范中 FPGA OS 与 UM 的接口为 1 组输入与 1 组输出，实现了 CPU 通路。由于 TSNNic 针对四个千兆接口具有其特殊的功能需求（分别用于控制、数据输出、数据输入、流量采样），将 FPGA

OS 与 UM 的接口设计为 4 组输入与 4 组输出，实现了 CPU 通路；具体接口定义如图 2 所示，接口信号的含义如下表 1 所示。



UM 模块接口信号定义图

表1 接口信号定义及列表

信号名	位宽	方向	备注
user_clk	input	1	125Mhz 的输入时钟
user_reset_n	input	1	复位信号，低有效
FPGA OS Ingress to UM 信号定义			
pktin_N_data_wr	input	1	报文数据写信号，N 为 0-3
pktin_N _data	input	134	报文数据，N 为 0-3
pktin_N _data_valid	input	1	报文数据标志位,1 为有效分组，0 为无效分组，N 为 0-3
pktin_N_data_valid_wr	input	1	报文数据标志位写信号，N 为 0-3
pktin_N _ready	output	1	数据 ready 信号，N 为 0-3
FPGA OS CDC to UM 信号定义			
cpu2um_data_wr	input	1	报文数据写信号
cpu2um _data	input	134	报文数据
cpu2um _data_valid	input	1	报文数据标志位,1 为有效分

			组，0 为无效分组。
cpu2um_data_valid_wr	input	1	报文数据标志位写信号
um2cpu _ready	output	1	数据 ready 信号
UM to FPGA OS Egress 信号定义			
pktout_N _data_wr	output	1	输出报文写信号，N 为 0-3
pktout_N _data	output	134	输出报文数据，N 为 0-3
pktout_N_data_valid	output	1	输出报文标志位，N 为 0-3
pktout_N_data_valid_wr	output	1	输出报文标志位写信号，N 为 0-3
pktout_N _ ready	input	1	输出报文 ready 信号，N 为 0-3
UM to FPGA OS CDC 信号定义			
um2cpu _data_wr	output	1	报文数据写信号，N 为 0-3
um2cpu _data	output	134	报文数据
um2cpu _data_valid	output	1	报文标志位
um2cpu_data_valid_wr	output	1	报文标志位写信号
um2cpu _ ready	input	1	报文 ready 信号

2.4 分组数据结构定义

2.4.1 硬件的分组数据结构定义

TSNNic 硬件中的分组数据结构定义与 FAST 2.0 规范中的一样，此处不在赘述，详见附录 A。

2.4.2 软硬件通信的分组数据结构定义

在 FAST 2.0 中的 FAST APP 运行在本地，通过 PCIe 总线与 FPGA OS 连接，而 TSNNic 是纯 FPGA 设计，在远程 Linux 主机上运行 FAST APP，所以需要重新设计 FAST lib，同时定义通过以太网传输的 FAST 分组的格式，详见附录三。

2.5 Metadata 定义

为实现硬件模块之间以及硬件模块和软件模块间的交互功能，需要将模块处理后的中间状态写在 Metadata 中。根据目前 TSNNic 的硬件功能需求，只需使用 FAST_2.0 规范中的 Metadata0 格式的分组长度和时间戳字段，并在自定义的 Metadata1 中维护接收报文的五元组信息（软件可直接用此五元组信息与 8 种报文头中的五元组信息进行匹配，识别该报文是否为 8 种类型报文中某一种报文），使用到的关键字段如下表 2 所示：

表2 Metadata 格式

Metadata 0			
[127]	1	pktsrc	分组的来源，0 为网络接口输入，1 为 CPU 输入
[126]	1	pktdst	分组目的，0 为网络接口输出，1 为送 CPU
[125:120]	6	inport	分组的输入端口号
[119:118]	2	outtype	00:单播；01：组播；10：泛洪；11：从输入接口输出
[117:112]	6	outport	单播：分组输出端口 ID，组播/泛洪：组播/泛洪表地址索引

[111:109]	3	priority	分组优先级
[108]	1	discard	丢弃位
[107:96]	12	len	包含 Metadata 字段的分组长度
[95:88]	8	smid	最近一次处理分组的模块 ID
[87:80]	8	dmid	下一个处理分组的模块 ID
[79:72]	8	pst	标准协议类型
[71:64]	8	seq	分组接收序列号
[63:50]	14	flowid	流 ID
[49:48]	2	reserve	保留
[47:0]	48	ts	时间戳（位宽由 FAST 2.0 规范中 32 位扩展到 48 位，用于存放报文的接收时间戳信息）
Metadata 1			
[127:104]	24	reserve	保留
[103:72]	32	src_ip	源 IP
[71:40]	32	dst_ip	目的 IP
[39:32]	8	protocol	协议
[31:16]	16	src_port	源端口号
[15:0]	16	dst_port	目的端口号

附录 A. FAST 2.0 规范中分组数据结构定义

UM 中数据分组包括 Metadata 头部及有效数据分组两部分，格式如图 A-1 所示，Metadata 在 FAST 报文的前 32 字节携带，每个分组进出 UM 的第 1 拍 16 字节为 Metadata0，第二拍数据为 Metadata1。

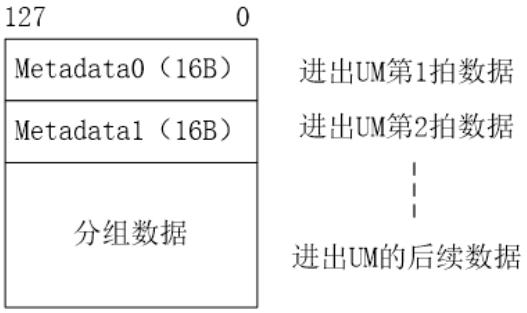


图 A-1 分组数据传输格式

接口分组(packet)是应用在 FPGA OS 与 UM 接口上的 134bit 的数据格式，其中高 6 位为控制信息，低 128 位为报文数据。分组的前两拍为 FPGA OS 添加的 32 字节的 metadata，两拍后的数据为有效分组数据。134 位的数据由 2 位的头尾标识，4 位无效字节数，128 位的有效数据组成。

其中，[133:132]位为报文数据的头尾标识，01 代表报文头部，11 代表报文中间数据，10 代表报文尾部；[131:128]位为 4 位的无效字节数，其中 0000 表示 16 个字节全部有效，0001 表示最低一个字节无效，最高 15 个字节有效，依次类推，1111 表示最低 15 个字节无效，最高一个字节有效。格式如图 A-2 所示。

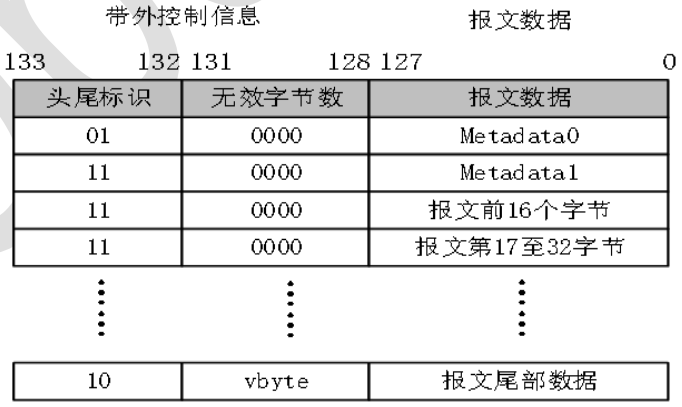


图 A-2 报文分组传输格式

附录 B. Beacon 报文格式设计

本部分介绍 Beacon 报文格式的详细设计。BEACON 报文目前总体分为两类：Update Message 与 Report Message。为了方便软件在测试过程中动态更新 8 种类型的报文头，将 Update Message 进一步细划为用于更新报文头的 Update PKT_HDR Message 和用于更新门控列表、硬件需配置的寄存器的 Update GCL_CA Message，三种消息（Update PKT_HDR Message、Update GCL_CA Message、Report Message）的 MsgType 分别为 0x1、0x2 与 0x3。在命令阵列、状态阵列的每个模块消息域后保留一拍，预防以后信号的增加、修改等。Beacon 报文以太网帧头的格式见图 B-1，其中 4bit 的 MsgType 字段表示 Beacon 报文消息类型；4bit 的 LAU_sequence 字段表示配置门控列表、命令阵列的报文序列号，用来判断门控列表、命令阵列是否配置成功；8bit 的 PHU_sequence 字段表示配置报文头的报文序列号，用来判断报文头是否配置成功。

7bit 0	偏移量
目的MAC	0~5
源MAC	6~11
以太网类型	12~13
MsgType, LAU_sequence	14~14
PHU_sequence	15~15

图 B-1 Beacon 报文以太网帧头的格式

1. Beacon Update PKT_HDR 报文格式

配置报文头的 Beacon 报文格式如图 B-1 所示，具体的报文消息域划分表如表 14 所示。

7bit 0	偏移量
以太网帧头（ETH hdr）	0~15
FAST头（FAST hdr）	16~47
8种类型报文头（PH）	48~559

图 B-2 配置报文头的 Beacon 报文格式

表3 配置报文头的 Beacon 报文消息域划分表

模块	拍数	字段	信号名	含义
PKT_HDR_RAM	6~9	[127:0]	-	第 1 种类型报文头
	10~13	[127:0]	-	第 2 种类型报文头
	14~17	[127:0]	-	第 3 种类型报文头
	18~21	[127:0]	-	第 4 种类型报文头
	22~25	[127:0]	-	第 5 种类型报文头
	26~29	[127:0]	-	第 6 种类型报文头
	30~33	[127:0]	-	第 7 种类型报文头
	34~37	[127:0]	-	第 8 种类型报文头

2. Beacon Update GCL_CA 报文格式

配置门控列表、命令阵列的 Beacon 报文格式如图 B-2 所示，具体的报文消息域划分表如表 15 所示。

OpenTSN

7bit 0	偏移量
以太网帧头（ETH hdr）	0~15
FAST头（FAST hdr）	16~47
门控列表（GCL）	48~559
命令阵列（CA）	560~1407

图 B-3 配置门控列表、命令阵列的 Beacon 报文格式

表4 配置门控列表、命令阵列的 Beacon 报文消息域划分表

模块	拍数	字段	信号名	含义
GCL_RAM	6~37	[127:0]	-	一个周期的门控列表，包含 16 个 slot 的 8 种类型报文的门控状态
PGM	38	[32]	test_stop	测试结束信号
		[31:0]	gcl_time_slot_cycle	门控列表的一个 $\frac{\text{时间槽大小}}{8}$
	39	[95:80]	tb3_size	第 3 种类型报文对应的令牌桶的桶深
		[79:64]	tb3_rate	每隔 1 个 slot 往第 3 种类型报文对应的令牌桶中添加的令牌数量
		[63:48]	tb2_size	第 2 种类型报文对应的令牌桶的桶深

		[47:32]	tb2_rate	每隔 1 个 slot 往第 2 种类型报文对应的令牌桶中添加的令牌数量
		[31:16]	tb1_size	第 1 种类型报文对应的令牌桶的桶深
		[15:0]	tb1_rate	每隔 1 个 slot 往第 1 种类型报文对应的令牌桶中添加的令牌数量
	40	[95:80]	tb6_size	第 6 种类型报文对应的令牌桶的桶深
		[79:64]	tb6_rate	每隔 1 个 slot 往第 6 种类型报文对应的令牌桶中添加的令牌数量
		[63:48]	tb5_size	第 5 种类型报文对应的令牌桶的桶深
		[47:32]	tb5_rate	每隔 1 个 slot 往第 5 种类型报文对应的令牌桶中添加的令牌数量
		[31:16]	tb4_size	第 4 种类型报文对应的令牌桶的桶深

OpenTSN

		[15:0]	tb4_rate	每隔 1 个 slot 往第 4 种类型报文对应的令牌桶中添加的令牌数量
	41	[63:48]	tb8_size	第 8 种类型报文对应的令牌桶的桶深
		[47:32]	tb8_rate	每隔 1 个 slot 往第 8 种类型报文对应的令牌桶中添加的令牌数量
		[31:16]	tb7_size	第 7 种类型报文对应的令牌桶的桶深
		[15:0]	tb7_rate	每隔 1 个 slot 往第 7 种类型报文对应的令牌桶中添加的令牌数量
	42	[127:112]	pkt_8_len	第 8 种类型报文的字节数（不包括 4B 的 CRC）
		[111:96]	pkt_7_len	第 7 种类型报文的字节数（不包括 4B 的 CRC）
		[95:80]	pkt_6_len	第 6 种类型报文的字节数（不包括 4B 的

OpenTSN

				CRC）
		[79:64]	pkt_5_len	第 5 种类型报文的字节数（不包括 4B 的 CRC）
		[63:48]	pkt_4_len	第 4 种类型报文的字节数（不包括 4B 的 CRC）
		[47:32]	pkt_3_len	第 3 种类型报文的字节数（不包括 4B 的 CRC）
		[31:16]	pkt_2_len	第 2 种类型报文的字节数（不包括 4B 的 CRC）
		[15:0]	pkt_1_len	第 1 种类型报文的字节数（不包括 4B 的 CRC）
	43	保留		
FSM	44	[103:0]	{src_ip_1,dst_ip_1,protocol_1,src_port_1,dst_port_1}	第 1 种类型报文的五元组
	45	[103:0]	mask_1	第 1 种类型报文的五元组掩码
	46	[103:0]	{src_ip_2,dst_ip_2,protocol_2,src_port_2,dst_port_2}	第 2 种类型报文的五元组

OpenTSN

	47	[103:0]	mask_2	第 2 种类型报文的五元组掩码
	48	[103:0]	{src_ip_3,dst_ip_3,protocol_3,src_port_3,dst_port_3}	第 3 种类型报文的五元组
	49	[103:0]	mask_3	第 3 种类型报文的五元组掩码
	50	[103:0]	{src_ip_4,dst_ip_4,protocol_4,src_port_4,dst_port_4}	第 4 种类型报文的五元组
	51	[103:0]	mask_4	第 4 种类型报文的五元组掩码
	52	[103:0]	{src_ip_5,dst_ip_5,protocol_5,src_port_5,dst_port_5}	第 5 种类型报文的五元组
	53	[103:0]	mask_5	第 5 种类型报文的五元组掩码
	54	[103:0]	{src_ip_6,dst_ip_6,protocol_6,src_port_6,dst_port_6}	第 6 种类型报文的五元组
	55	[103:0]	mask_6	第 6 种类型报文的五元组掩码
	56	[103:0]	{src_ip_7,dst_ip_7,protocol_7,src_port_7,dst_port_7}	第 7 种类型报文的五元组
	57	[103:0]	mask_7	第 7 种类型报文的五

OpenTSN

				元组掩码
	58	[103:0]	{src_ip_8,dst_ip_8,protocol_8,src_port_8,dst_port_8}	第 8 种类型报文的五元组
	59	[103:0]	mask_8	第 8 种类型报文的五元组掩码
	60	保留		
SSM	61	[15:0]	samp_freq	用于控制读取报文的频率

注：8 种类型的报文长度(字节数)不包含 2 拍 metadata 的长度。

3. Beacon Report 报文格式

Beacon 上报报文格式如图 B-3 所示，具体的报文消息域划分表如表 16 所示。

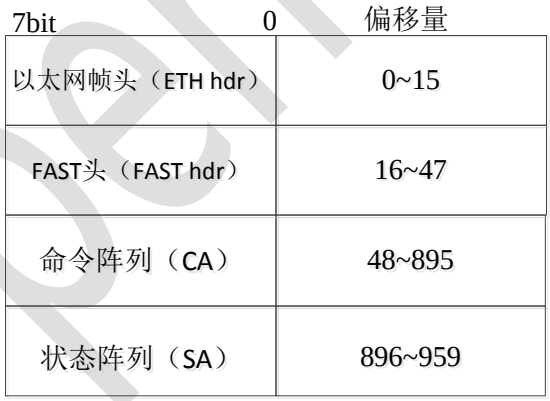


图 B-4 Beacon 上报报文格式

表5 Beacon 上报报文消息域划分表

模块	拍数	字段	信号名	含义
PGM (CA)	6	[32]	test_stop	测试结束信号。 0：测试开始；1：测试

				结束
		[31:0]	gcl_time_slot_cycle	门控列表的一个 $\frac{\text{时间槽大小}}{8}$
	7	[95:80]	tb3_size	第 3 种类型报文对应的令牌桶的桶深
		[79:64]	tb3_rate	每隔 1 个 slot 往第 3 种类型报文对应的令牌桶中添加的令牌数量
		[63:48]	tb2_size	第 2 种类型报文对应的令牌桶的桶深
		[47:32]	tb2_rate	每隔 1 个 slot 往第 2 种类型报文对应的令牌桶中添加的令牌数量
		[31:16]	tb1_size	第 1 种类型报文对应的令牌桶的桶深
		[15:0]	tb1_rate	每隔 1 个 slot 往第 1 种类型报文对应的令牌桶中添加的令牌数量
	8	[95:80]	tb6_size	第 6 种类型报文对应的令牌桶的桶深
		[79:64]	tb6_rate	每隔 1 个 slot 往第 6 种类型报文对应的令牌桶中添加的令牌数量
		[63:48]	tb5_size	第 5 种类型报文对应的

OpenTSN

				令牌桶的桶深
		[47:32]	tb5_rate	每隔 1 个 slot 往第 5 种类型报文对应的令牌桶中添加的令牌数量
		[31:16]	tb4_size	第 4 种类型报文对应的令牌桶的桶深
		[15:0]	tb4_rate	每隔 1 个 slot 往第 4 种类型报文对应的令牌桶中添加的令牌数量
	9	[63:48]	tb8_size	第 8 种类型报文对应的令牌桶的桶深
		[47:32]	tb8_rate	每隔 1 个 slot 往第 8 种类型报文对应的令牌桶中添加的令牌数量
		[31:16]	tb7_size	第 7 种类型报文对应的令牌桶的桶深
		[15:0]	tb7_rate	每隔 1 个 slot 往第 7 种类型报文对应的令牌桶中添加的令牌数量
	10	[127:112]	pkt_8_len	第 8 种类型报文的字节数（包括 4B 的 CRC）
		[111:96]	pkt_7_len	第 7 种类型报文的字节数（包括 4B 的 CRC）
		[95:80]	pkt_6_len	第 6 种类型报文的字节

OpenTSN

				数（包括 4B 的 CRC）
		[79:64]	pkt_5_len	第 5 种类型报文的字节数（包括 4B 的 CRC）
		[63:48]	pkt_4_len	第 4 种类型报文的字节数（包括 4B 的 CRC）
		[47:32]	pkt_3_len	第 3 种类型报文的字节数（包括 4B 的 CRC）
		[31:16]	pkt_2_len	第 2 种类型报文的字节数（包括 4B 的 CRC）
		[15:0]	pkt_1_len	第 1 种类型报文的字节数（包括 4B 的 CRC）
	11	保留		
FSM (CA)	12	[103:0]	{src_ip_1,dst_ip_1,protocol_1,src_port_1,dst_port_1}	第 1 种类型报文的五元组
	13	[103:0]	mask_1	第 1 种类型报文的五元组掩码
	14	[103:0]	{src_ip_2,dst_ip_2,protocol_2,src_port_2,dst_port_2}	第 2 种类型报文的五元组
	15	[103:0]	mask_2	第 2 种类型报文的五元组掩码
	16	[103:0]	{src_ip_3,dst_ip	第 3 种类型报文的五元

OpenTSN

			_3,protocol_3,src_port_3,dst_port_3}	组
	17	[103:0]	mask_3	第 3 种类型报文的五元组掩码
	18	[103:0]	{src_ip_4,dst_ip_4,protocol_4,src_port_4,dst_port_4}	第 4 种类型报文的五元组
	19	[103:0]	mask_4	第 4 种类型报文的五元组掩码
	20	[103:0]	{src_ip_5,dst_ip_5,protocol_5,src_port_5,dst_port_5}	第 5 种类型报文的五元组
	21	[103:0]	mask_5	第 5 种类型报文的五元组掩码
	22	[103:0]	{src_ip_6,dst_ip_6,protocol_6,src_port_6,dst_port_6}	第 6 种类型报文的五元组
	23	[103:0]	mask_6	第 6 种类型报文的五元组掩码
	24	[103:0]	{src_ip_7,dst_ip_7,protocol_7,src_port_7,dst_por	第 7 种类型报文的五元组

OpenTSN

			t_7}	
	25	[103:0]	mask_7	第 7 种类型报文的五元组掩码
	26	[10:0]	{src_ip_8,dst_ip_8,protocol_8,src_port_8,dst_port_8}	第 8 种类型报文的五元组
	27	[103:0]	mask_8	第 8 种类型报文的五元组掩码
	28	保留		
SSM (CA)	29	[15:0]	samp_freq	用于控制读取报文的频率
	30	保留		
PGM (SA)	31	[127:96]	pgm_pkt_4_cnt	第 4 种类型报文的发送个数
		[95:64]	pgm_pkt_3_cnt	第 3 种类型报文的发送个数
		[63:32]	pgm_pkt_2_cnt	第 2 种类型报文的发送个数
		[31:0]	pgm_pkt_1_cnt	第 1 种类型报文的发送个数
	32	[127:96]	pgm_pkt_8_cnt	第 8 种类型报文的发送个数
		[95:64]	pgm_pkt_7_cnt	第 7 种类型报文的发送个数

OpenTSN

		[63:32]	pgm_pkt_6_cnt	第 6 种类型报文的发送个数
		[31:0]	pgm_pkt_5_cnt	第 5 种类型报文的发送个数
	33	保留		
FSM (SA)	34	[127:96]	ssm_pkt_4_cnt	接收第 4 种类型报文个数
		[95:64]	ssm_pkt_3_cnt	接收第 3 种类型报文个数
		[63:32]	ssm_pkt_2_cnt	接收第 2 种类型报文个数
		[31:0]	ssm_pkt_1_cnt	接收第 1 种类型报文个数
	35	[127:96]	ssm_pkt_8_cnt	接收第 8 种类型报文个数
		[95:64]	ssm_pkt_7_cnt	接收第 7 种类型报文个数
		[63:32]	ssm_pkt_6_cnt	接收第 6 种类型报文个数
		[31:0]	ssm_pkt_5_cnt	接收第 5 种类型报文个数
	36	保留		
SSM (SA)	37	[31:0]	pkt_cnt	接收的报文个数

注：

1. PGM 模块在报文（不包括两拍 metadata）的第四拍[47:0]打上发送时间戳。
2. 被封装的 FAST 头的长度字段 metadata0[107:96]不包含在 FPGA OS 加的 2 拍 FAST 头长度和以太网头的长度。
3. 优先级：第 1 种类型报文>第 2 种类型报文>…>第 8 种类型报文。
4. 每一拍门控列表数据[127:120]、…、[7:0]分别对应第 16 个 slot、…、第 1 个 slot。
5. 单位时间报文的最大发送/接收个数

$$=\frac{1\text{Gbit}}{(\text{最小报文长度 }64\text{B+ 以太网最小帧间距 }12\text{B+以太网帧前导码 }7\text{B+ 帧开始符 }1\text{B})*8}=1.488*10^6\text{个；}$$

1h 报文的最大发送/接收个数=$1.488*10^6*3600=5.357*10^9$个。

其中$2^{16}=65536$，$2^{32}=4.3*10^9$。

6. 软件配置的 slot 数值为$\frac{\text{时间槽大小}}{8}$；时间槽最大取200μs。
7. 往令牌桶添加令牌的速率与时间槽的关系：每隔一个时间槽往令牌桶中加一次令牌，一个令牌代表报文 1B，单位时间槽往令牌桶中增加令牌数

$$\text{tb\_rate}=\frac{\text{需限定的速率(bps)}}{\frac{8bit}{1B}}*\frac{\text{时间槽大小(ns)}}{\frac{10^9\text{ns}}{1s}}=\frac{\text{需限定的速率(bps)}*\text{时间槽大小(ns)}}{8*10^9}$$

tb_rate 的位宽：设定时间槽最大取200μs，若要支持报文以满速率 1Gbps 发送，则单位时间槽加的令牌数

$$\text{tb\_rate}=\frac{1024^3*200\_000}{8*10^9}=26844\text{个}$$

而$2^{16} = 65536$，所以 tb_rate 选择 16 位的位宽。

附录 C. 基于以太网连接的 FAST 分组传输格式

在 FAST 2.0 中的 FAST APP 运行在本地，通过 PCIe 总线与 FPGA OS 连接的，而 TSNNic 是纯 FPGA 设计，在远程 Linux 主机上运行 FAST APP，所以需要重新设计 FAST lib，同时定义通过以太网传输的 FAST 分组的格式。如下图 6-1-1 所示。

（1）FAST lib 提供给 APP 的编程接口 API 保持不变；

（2）FAST 分组通过以太网传输时，需要封装到新的以太网帧中；

（3）FPGA OS 收到 FAST 分组时，除了去掉 CRC 外，还要在送给 UM 的 P0-rx 接口前，再增加一个 FAST 头；

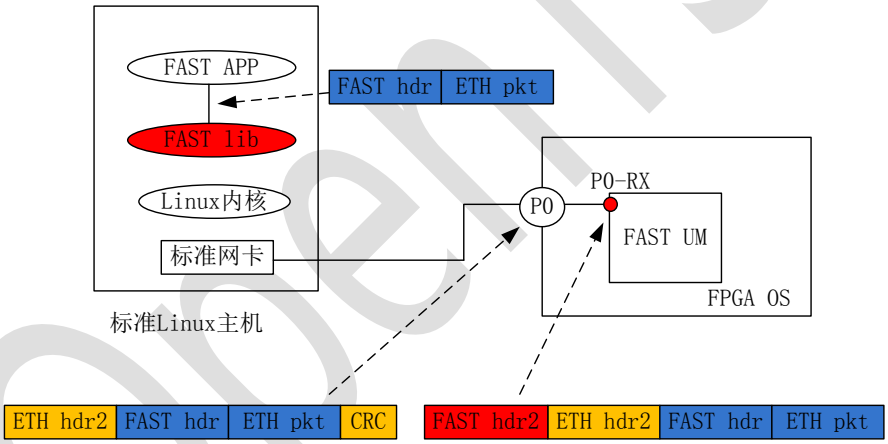


图 C-1 基于以太网连接的 FAST 分组传输格式

ETH hdr2 的定义为：

目的 MAC：0xffffffffffff

源 MAC：不关心

长度类型域：0xff01

FAST UM 通过 P0-TX 向外部主机 FAST APP 发送分组时，也遵循上述格式。

Openbox-s4 的 P0 端口默认为连接外部控制器的接口，在 FAST UM 内部，需要对进出 P0 端口的 FAST 分组进行封装和解封装处理。

附录 D. TSNNic 测试连接图

TSNNic 测试连接图如图 D-1 所示。TSNNic 控制器发送 Beacon 配置报文经过交换机从 0 号接口进入 TSNNic；TSNNic 生成的 Beacon 上报报文从 0 号接口输出，经过交换机给 TSNNic Insight；TSNNic 生成测试报文从 1 号接口输出到被测网络，经过被测网络后从 2 号接口回到 TSNNic；TSNNic 对回来的测试报文进行封装采样后，从 3 号接口输出，经过交换机给 TSNNic Insight。

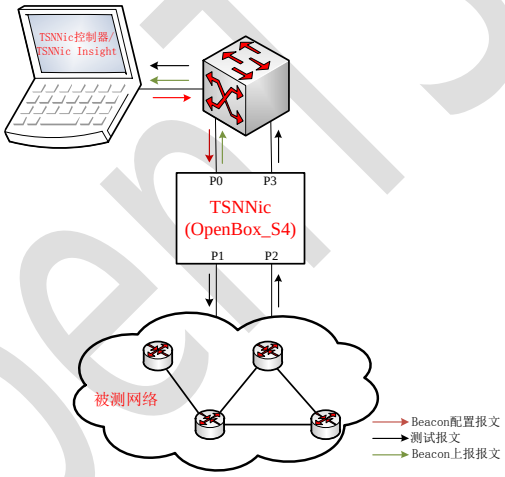


图 D-1 TSNNic 测试连接图

附录 E. FAST 硬件架构原理

FAST 平台的硬件架构如图 E-1 所示。FAST 定义了平台相关逻辑/代码和平台无关逻辑/代码之间的接口规范。其中 FPGA 中的硬件流水线的实现与具体的平台无关，称为用户模块（UM）；平台相关的逻辑，称为 FPGA OS。

FPGA OS 主要有两个功能，一是屏蔽平台的异构性，使得 UM 的 verilog 代码具有更好的跨平台移植能力；二是为 UM 的实现提供

共性的服务。FPGA OS 屏蔽不同 FPGA 平台的异构性主要包括五方面：

一是不同的 FPGA 型号，如来自不同厂商，具有不同的开发工具，FPGA 具有不同的容量和速度等级等；

二是不同的网络接口，如千兆以太网接口和万兆以太网接口，以及不同的接口数目等；

三是不同的外部接口类型，如有的连接 TCAM 协处理器，有的连接 DDR 存储器，有的连接 SRAM 存储器等；

四是不同的接口逻辑实现方式，例如有的以太网 MAC 逻辑在 FPGA 内嵌的 IP 核实现，有的 MAC 逻辑由 FPGA 外部的 PHY/MAC 一体的芯片实现。不同的接口逻辑实现方式获取接口状态，如 up/down，的方式也不同；

五是 FPGA 与 CPU 不同的通信方式，如果 FPGA 与 CPU 在同一个板卡上或机箱内， FPGA 可能通过 PCIe 总线与 CPU 进行通信，否则 FPGA 可能通过以太网与物理位置相分离的 CPU 进行通信。

FPGA OS 为简化 UM 设计提供各种通用服务，如分组的接收和发送，外部存储器的访问，与 CPU 通信的高速 DMA，以及分组处理中需要的规则匹配等。

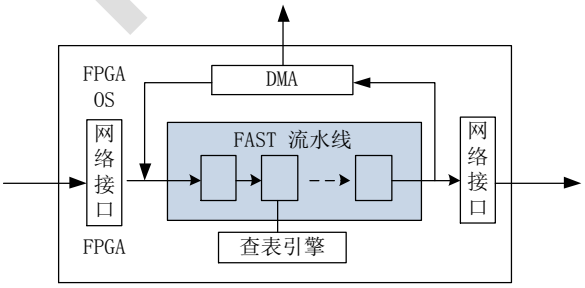


图 E-1 FAST 平台的硬件架构

附录 F. TSNNic 性能

1. 用商用测试仪 Ixia 测试 TSNNic 发送带宽。

测试场景：用 TSNNic 发包，报文长度 512B（包括 4B 的 CRC），用 Ixia 测 TSNNic

的实际发送带宽。Ixia 显示的速率值波动很小，每次测试时取三组数据求平均值得到实际发送带宽。测得的数据如表 17 所示。

表6 TSNNic 带宽测试数据

TSNNic 理论发送带宽(bps)	用 Ixia 测得的实际发送带宽(bps)	差值(bps)
200,000,000	200,000,648	648
400,000,000	400,009,503	9,503
600,000,000	600,000,072	72
800,000,000	800,008,197	8,197

数据分析：由差值一览可得：TSNNic 理论发送带宽与用 Ixia 测得的实际发送

带宽相差小于 10,000bps。

2. 用商用测试仪 Ixia 测试 TSNNic 最大发送带宽和吞吐量，并与 Ixia 做比较。

测试场景：

1）TSNNic 发包，用 Ixia 来测试 TSNNic 在发送不同报文长度（包括 4B 的 CRC）

时的最大带宽和吞吐量；Ixia 显示的值波动很小，每次测试时取三组数据求平均值得到最大带宽和吞吐量；测得的数据如表 18 和表 19 所示。

2）商用测试仪 Ixia 自己发包，回环测试 Ixia 的最大发送带宽和吞吐量；测得的数

据如表 18 和表 19 所示。

表7 TSNNic 和 Ixia 的最大发送带宽比较

最大发送带宽(bps)	TSNNic	Ixia	差值(bps)
64B	761,912,625	761,905,402	7,223
128B	864,873,619	864,864,028	9,591
256B	927,545,901	927,537,389	8,512
512B	962,415,725	962,406,107	9,618
1518B	987,010,570	987,007,541	3,029

表8 TSNNic 和 Ixia 的吞吐量比较

吞吐量(pps)	TSNNic	Ixia	差值(pps)
64B	1,488,111	1,488,095	16
128B	844,603	844,594	9
256B	452,903	452,899	4
512B	234,965	234,964	1
1518B	81,275	81,273	2

数据分析：由差值一览可得：TSNNic 与 Ixia 的最大发送带宽相差小于 10,000bps；TSNNic 与 Ixia 的吞吐量相差小于 20pps。

附录 G. FPGA 资源评估

当前使用的资源如下表 20 所示：

表9 使用的资源情况

模块	资源使用		
	LUT	FF	BRAMS
FPGA OS	21299	32652	34.5
um_test_0	6996	6463	4
小计	28295	39115	38.5

FPGA 总资源以及剩余资源如下表 21 所示：

表10 总资源与剩余资源情况

FPGA 总资源	LUT	FF	BRAMS
	53200	106400	140（560KB）
FPGA 剩余资源	LUT	FF	BRAMS
	24905	67285	101.5

UM 中各模块所使用的资源如下表 22 所示：

表11　UM 中各模块使用资源情况

模块	LUT	FF	BRAMS
LCM 模块	1741	2672	0
PGM 模块	1594	1179	0
FSM_1 模块	390	228	0
FSM_2 模块	391	228	0
FSM_3 模块	392	228	0
FSM_4 模块	391	228	0
FSM_5 模块	392	228	0
FSM_6 模块	391	228	0
FSM_7 模块	391	228	0
FSM_8 模块	391	228	0
SSM 模块	535	788	0
GCL_RAM 模块	0	0	2
PKT_HDR_RAM 模块	0	0	2
小计	6996	6463	4

OpenTSN