TSNNic Hardware Design Documentation

(V1.0)

OpenTSN TSNNic

content

An introduction to the **TSNNic** project

TSN Network Interface Adapter (TSNNic) is a data adaptation node applied between TSN networks.

It can be used to inject application data into the TSN network according to the traffic transmission requirements and the TSN network encapsulation format.

transmission, and decapsulate the data received from the TSN network back to the application and perform related performance statistics.

Specifically, it includes generating and sending time-sensitive flows at a certain point in time, concurrency of time-sensitive flows/non-time-sensitive flows, network

Network traffic capture and analysis, etc.

The specific requirements are as follows:

1) Multiple streams are concurrent, and the message header information function of each stream can be dynamically updated during the test process;

2) Time-aware shaping (TAS) function;

3) Five-tuple matching with mask, and the function of counting the number of packets;

4) Frequency division sampling function;

5) Test the performance of the network/device under test, that is, test the network/device under test for different sizes and types of reports.

The precise delay, throughput rate and packet loss rate of the text;

6) With software configuration capability, users can simulate different traffic according to application requirements.

Two **TSNNic** hardware overall design

2.1 TSNNic platform-related logic customization

TSNNic is based on OpenBox-S4 based on FAST (FPGA Accelated Switching Platform) architecture

For development on the FAST architecture, please refer to the FAST design document "FAST Platform Principles and Applications 2.0".

At present, the standard FPGA OS of OpenBox-S4 provides input and output processing logic of four gigabit interfaces.

According to the functional requirements of TSNNic, the interface applications are mapped (respectively for control, data output, data input, stream

sample), and tailor the FPGA OS accordingly:

1. The FPGA OS does not perform MUX operations on the input side, and directly sends packets from the CPU and input from each port to the

FASTUM;

2. The FPGA OS does not perform DMUX operations on the output side, and sets a separate signal for each output interface;

3. In order to support time synchronization within the TSNNic device and between the TSNNic and TSNSwitch devices,

Add a transparent time revision mechanism;

4. Delete the table lookup engine function;

5. Remove the mechanism for communicating with the CPU over the PCIe bus.

2.2 TSNNic Platform Independent UM Architecture

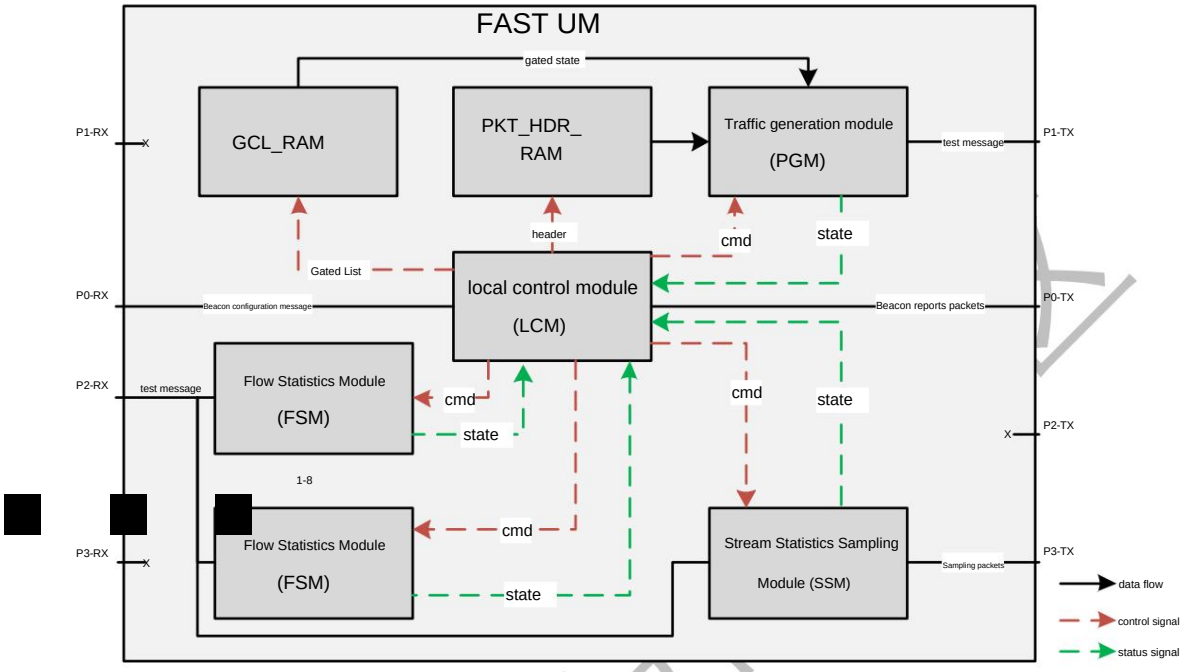

Figure 2-2-1 TSNNic hardware UM architecture diagram

Note: PN_RX/TX indicates that the packet is input/output from the N data interface, N=0, 1, 2, 3; "×" indicates that in UM the corresponding slave

The grouping of interface input/output is not logically processed.

The TSNNic hardware UM architecture is shown in Figure 2-2-1. The modules in the UM are described as follows:

LCM (Local Control Module) module: The local control module of TSNNic. Mainly responsible for FAST UM

Each module controls the update of related registers and periodic reporting of status information.

PGM (Packet Generation Module) module: The traffic generation module of TSNNic. based on token bucket

Mechanism, Time Aware Shaping (TAS) to achieve the generation and transmission of multiple streams.

FSM (Flow Statistic Module) module: The flow statistics module of TSNNic. Support 8 groups with mask

Five-tuple match, count the number of matched packets.

SSM (Statistic and Sample Module) module: Traffic statistics and sampling module of TSNNic. statistics

TSNNic receives the total number of packets, parses the received packets, extracts quintuple, and collects them according to a certain sampling frequency.

Encapsulates (FAST header) and samples the received packets at a rate.

GCL_RAM (Gate Control List RAM) module: Gated list centralized cache module. with one RAM

to cache the gated list.

PKT_HDR_RAM (Packet Header RAM) module: Centralized buffering module for packet headers. with one RAM

to cache 2 groups of 8 headers.

The overall processing flow of UM is as follows:

The LCM module parses the received packets. If it is a Beacon configuration packet (there are two types of Beacon configuration packets)
One is used to configure 8 packet headers, and the other is used to configure the gate control list and command array), then configure the Beacon
The information carried in the set message is read out and output to the corresponding module, as shown by the red line in Figure 2-2-1; the gate control list and
After the command array and message header are configured in sequence, the LCM module gives the test start signal, and the PGM module starts the test.
start running;

The PGM module generates 8 types of concurrent traffic based on the token bucket mechanism and the gated list. The token bucket mechanism is used to control each
There are various traffic generation rates, and the gated list is used to control the time when each traffic is sent; the PGM module receives the test start signal.
After the number of tokens, several tokens are injected into the token bucket every time slot, and if the scheduling conditions are met (the remaining tokens in the token bucket are
The number of cards ÿ the length of the message (bytes), and the current gating state is open), the message headers are scheduled according to the priority, and the message header
On the basis of the header, it is expanded according to the length of the message to generate a message (adding the timestamp and serial number fields of the message)
And send; count the number of each kind of message sent, and send the statistical result to the LCM module. The FSM module responds to the received report
The text matches based on the quintuple with mask, counts the number of hit packets, and transmits the statistical result to the LCM module.
The SSM module counts the total number of received packets, and transmits the statistical results to the LCM module; it encapsulates the received packets with a
FAST header, extract the quintuple of the message and put it in the encapsulated metadata1; receive it according to a certain sampling frequency
The packets are sampled and output from interface 3. During the test process, the message header information can be dynamically updated, and the LCM mode
The block generates a Beacon report message every 1s to report the current UM status (as indicated by the green line in Figure 2-2-1).
Show);

When the LCM module receives the test stop signal, it generates and sends the last Beacon report message after 1s.
Then the register reset signal is set high, the PGM module clears the number of each message sent, and the FSM module resets each
The number of received messages is cleared, and the SSM module clears the total number of received messages.

Three **TSNNic** module design

3.1 LCM Module Design

3.1.1 Analysis of LCM Requirements and Functions

The LCM module is responsible for UM global information acquisition, update, and control, and needs to support the following functions:

(1) The LCM module periodically generates a Beacon report message that encapsulates the FAST header, and fills in the register information into
corresponding position;

(2) After receiving the register configuration message (encapsulated Beacon update message) from the software, the readable field will be
The value of the module is read out, and the value in the corresponding register in the module is updated;

(3) Receive two kinds of Beacon configuration messages, which are used to configure 8 types of headers, gate control list and command array respectively.

(4) After configuring the gate control list, command array, and 8 types of headers, the tester starts to work;

After stopping, the last Beacon report message is generated after 1s, and then the register reset signal is set high.

3.1.2 LCM module design

The overall structure of the LCM module is shown in Figure 3-1-1.

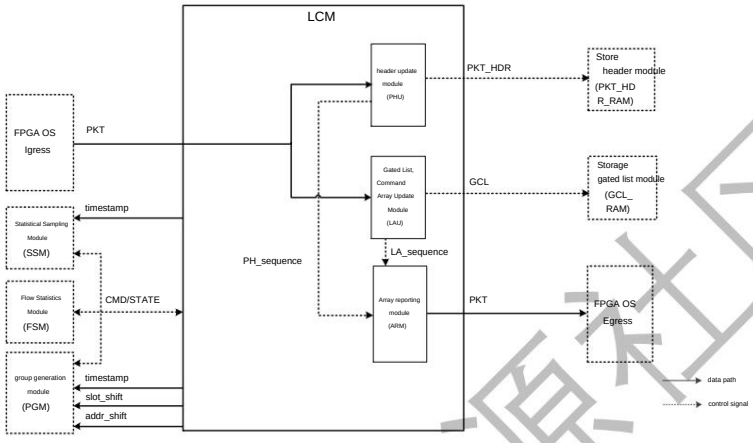

Figure 3-1-1 Overall architecture of LCM module

PHU (PKT Header Update) module: The packet header update module, after receiving the packet from the FPGA OS Igress

Then, judge whether it is a Beacon packet used to update the message header: if so, read the message header and write it to

PKT_HDR_RAM, and pass the serial number of the packet to the ARM module; if not, discard the packet

abandoned.

LAU (List and Array Update) module: The gated list and command array update module, when received from the FPGA OS

After the grouping of the Igress, determine whether it is a Beacon grouping used to update the gated list and command array: if so, then

Read out the gate control list, write it to GCL_RAM, and read out the command array to update the corresponding registers in the module

The value of the packet is passed to the ARM module at the same time; if not, the packet is discarded.

ARM (Array Report Module) module: Array report module, the local clock counter value elapses every 1s, according to

The Beacon Report message format definition constructs a Beacon Report group and sends it to the software to report the current

Full status information for FAST UM.

3.2 PGM module design

3.2.1 Analysis of PGM Requirements and Functions

The PGM module is responsible for generating multiple concurrent traffic of different types at specified rates, and needs to support the following functions:

1) Support to generate and send 8 types of messages;

2) Support speed limit based on token bucket mechanism;

3) Support scheduling data based on Time Aware Scheduler (TAS);

4) Support scheduling according to absolute priority in the queue with the gate of 8 data queues open;

5) Support software to dynamically update 8 types of headers during testing

6) Supports packet expansion based on the packet header configured by the user, adding timestamps, serial numbers, etc.

field;

7) Support communication with LCM module, so as to update, report and reset different parameter registers.

3.2.2 PGM module design

The overall structure of the PGM module is shown in Figure 3-2-1.

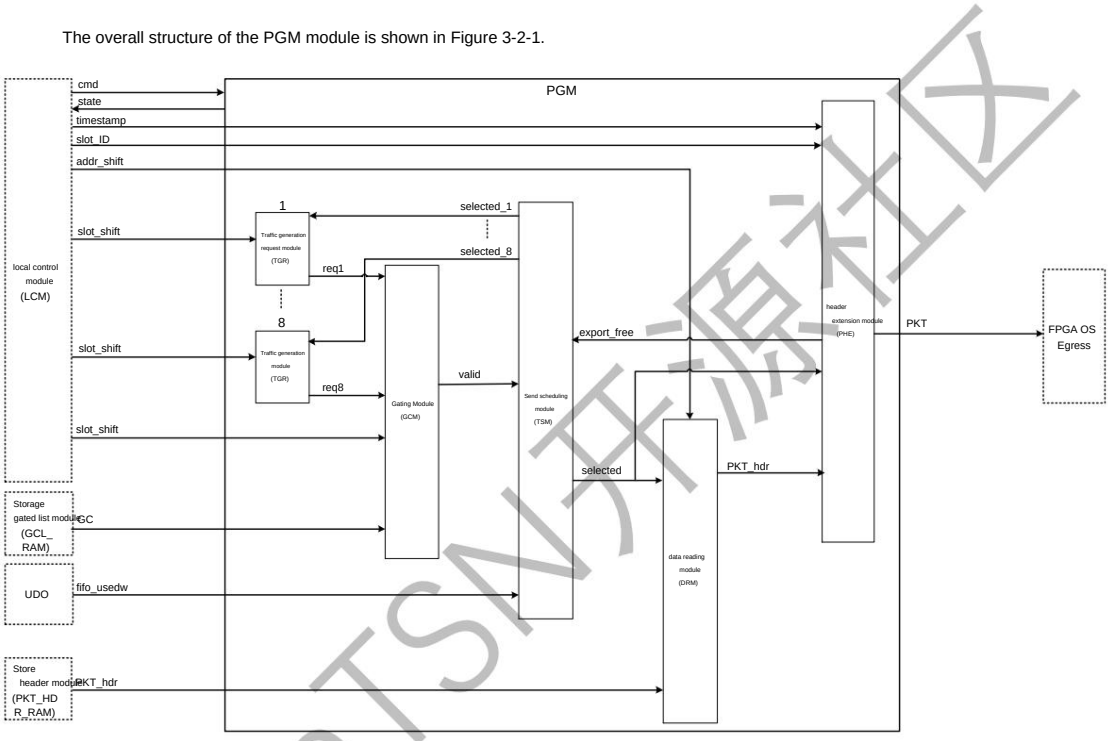

Figure 3-2-1 Overall architecture of PGM module

TGR (Traffic Generation Request) module: traffic generation request module, traffic based on token bucket mechanism

speed limit. After each time slot, a certain number of tokens are added to the token bucket, each token represents 1B, when the token

When the number of remaining tokens in the bucket is greater than or equal to the number of message bytes + 4 bytes (CRC), there is a traffic generation request for this type of message.

When this type of packet is scheduled to be sent, the number of bytes in the packet and 4 bytes (CRC) are subtracted from the number of remaining tokens in the token bucket.

GCM (Gate Control Module) module: gate control module, after the gate control list and command array are updated (this

When the test has not yet started), the GCM module reads out a 128-bit gated state from the GCL RAM; in the gated list,

Every 16 time slots after the command array and the message header are updated, the GCM module reads 128 from the GCL RAM.

The gate control state of the bit; if the gate control corresponding to a certain type of message in the current time slot is ON, the request for this type of message is valid;

Otherwise invalid. When the test end signal is received, all valid signals transmitted to the sending scheduling module are set to 0.

TSM (Transmitting and Scheduling Module) module: send scheduling module, just reset or port

Idle, generate a request valid signal according to the received 8-bit traffic, schedule it in the order of priority, and put the

The degree results are passed to the TGR, DRM and PHE modules.

DRM (Data Reading Module) module: data reading module, according to the scheduling result from TSM and two sets of message header switching signals

from LCM, it transmits the read signal and address of the corresponding message header to PKT_HDR_RAM; when

After receiving the header, forward it to the PHE module.

PHE (PKT_header Extension) module: message header extension module, according to the message header transmitted by DRM, LCM transmission

The length of the incoming packet, the scheduling result from the TSM generates the packet: add 2 beats of metadata, before the 128-bit data

Add 6 bits to the header (2-bit header and tail identifier, 4-bit invalid number of bytes); add the sending timestamp and message after the message header

Serial number and other fields, and fill the remaining free fields with 0. The number of sent 8 types of packets is counted.

3.3 PKT_HDR_RAM Design

PKT_HDR_RAM is used to store 8 types of headers. Software-side constructs specific features for generating concurrent traffic

The message header is sent to the hardware LCM module through the Beacon message, and the LCM module writes the message headers in sequence

PKT_HDR_RAM. After the initialization of the test, the PGM module changes the data from PKT_HDR_RAM according to the

The corresponding header is read out. In order to facilitate the software to dynamically update 8 types of headers during the test,

PKT_HDR_RAM needs two address fields to store two sets of 8 headers.

Based on the above process, PKT_hdr RAM adopts Simple Dual-Port RAM, namely

One port is used for writing data and the other port is used for reading data.

Each message header constructed by the software is 4 beats of data, and the two groups of message headers have a total of 2*8*4=64 beats, and each beat of data

The bit width is 128 bits and requires 64*128 bits of storage space. The basic unit of Block RAM is 18Kb Block RAM

36Kb Block RAM composed of 2 18Kb Block RAMs, other configurations are based on these two basic units;

Choose 64*128 for RAM, which will consist of two 36Kb Block RAMs.

To sum up, the data bit width of PKT_HDR_RAM is 128 bits, and the address bit width is 6 bits, occupying a total of 2

36Kb block RAM.

3.4 GCL_RAM Design

GCL_RAM is used to store the gate control list corresponding to 8 types of message headers, which is used to determine the

Whether the header can be scheduled. The software side sends the Beacon message to the hardware LCM module, and the LCM module will

Tables are sequentially written to GCL_RAM. After the test initialization is complete, the PGM module starts from every 16 time slots

The gated state of one beat is read from GCL_RAM.

Based on the above process, GCL_RAM adopts Simple Dual-Port RAM (Simple Dual-Port RAM).

One port is used for writing data and the other port is used for reading data.

The gate control list is 32 beats of data, and the bit width of each beat is 128 bits, including 8 types of reports in 16 time slots.

Gating state of the header; requires 32*128 bits of storage space. The basic unit of Block RAM is 18Kb Block RAM

36Kb Block RAM composed of 2 18Kb Block RAMs, other configurations are based on these two basic units;

RAM selection is 32*128, which will consist of two 36Kb Block RAMs.

To sum up, the data bit width of GCL RAM is 128 bits, and the address bit width is 5 bits, occupying a total of 2 36Kb Blocks

RAM.

3.5 FSM Module Design

3.5.1 Analysis of FSM Requirements and Functions

The FSM (Flow Statistical Module) module is responsible for the statistics of packets and needs to support the following functions:

1) Support receiving quintuple and quintuple mask parameters of 8 types of messages sent by the software;

2) Support IP packets with vlan header and without vlan header for received packets (currently only for tcp and udp

message) for quintuple information extraction;

3) Support the matching of the packet extraction quintuple and the concerned quintuple issued by the software, and make statistics;

4) Support communication with the LCM module and report the value of the register.

3.5.2 FSM module design

As the flow statistics module, the FSM first receives the quintuple information of the packets sent by the control flow and needs to be counted.

(lcm2fsm_5tuple) and quintuple mask information (lcm2fsm_5tuplemask), when the test packet enters the FSM,

Distinguish IP packets with vlan headers and those without vlan headers, and extract the quintuple information of the packets and the information sent by the control flow.

The quintuple information (lcm2fsm_5tuple) is matched with the quintuple mask information (lcm2fsm_5tuplemask), only

Matches the quintuple information that the mask information cares about. If the match is found, the packet is counted, and the statistical value is reported.

Local control module LCM.

3.6 SSM Module Design

3.6.1 Analysis of SSM Requirements and Functions

The SSM module is responsible for packet encapsulation and facilitates the software side to parse the packet. It needs to support the following functions:

1) Encapsulate the packets entering the SSM, add FAST hdr2 and ETH hdr2, and add the ip with vlan

The 5-tuple information of the message (tcp and udp message) is extracted and stored in the lower 104 bits of FAST hdr MD1;

2) Receive the sampling frequency sent by the software, and sample the message according to the

frequency; 3) Support communication with the LCM module and report the value of the register; 4) Support

the maximum sending of 1514byte messages.

3.6.2 SSM module design

The overall structure of the module is shown in Figure 3-6-1

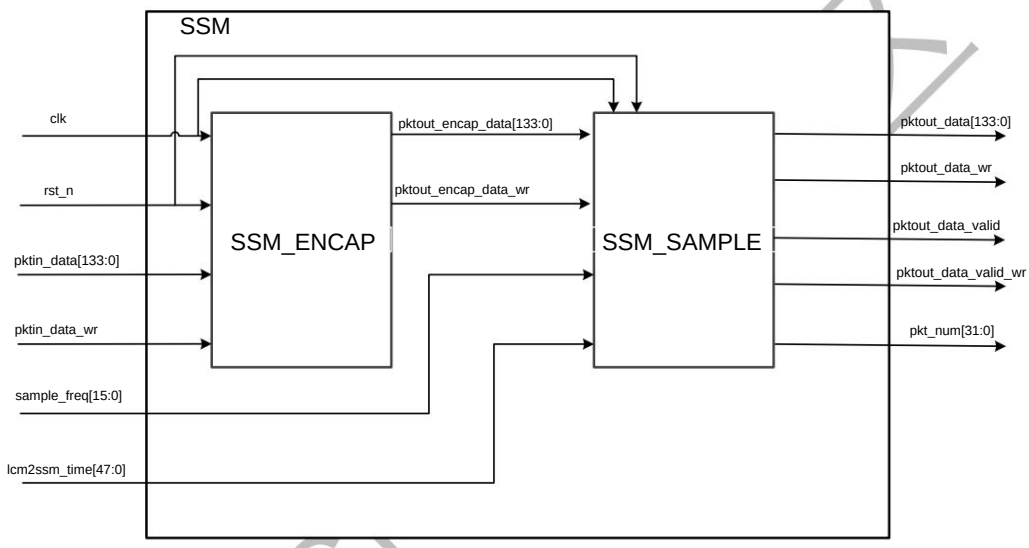

Figure 3-6-1 SSM Internal Architecture

The SSM is divided into the encapsulation module SSM_ENCAP and the sampling module SSM_SAMP. The sampling module has a 16

Bit latch reg_pkt_num, record the number of received messages.