Fenlin-I chip design
（V4.0）

1、Overview

This document introduces the design and implementation details of Fenglin-I chip, which is mainly divided into three parts: overview, overall design and detailed design.

1.1 Design goal

TSN enhances the traditional Ethernet in time synchronization, delay determinism, reliable transmission and management control. Its application scenarios have been expanded from the initial industrial Internet to operator network, vehicle network and aerospace network. With the enrichment and expansion of network applications in different fields, the features of different application scenarios have shown the characteristics of diversification and differentiation. In order to meet the diversified and differentiated application requirements in the above fields, We launched the open-source TSN chip, Fenglin-I. Fenglin-I is designed by extracting the appropriate subset of TSN standards, aiming to become the basis of diversified TSN chips.

1.2 Key indicator

1) Support IEEE 802.1AS, 802.1Qbv, 802.1Qcc standards;
2) Support ST, RC and BE traffic;
3) Support microsecond determinism;
4) Support 9 Gigabit Ethernet interfaces (8 network interfaces, 1 host interface);
5) Switching bandwidth arrives at 16Gbps;
6) The switching delay is less than 2us;
7) Support 16K flows;

2、 Overall design

2.1 Fenglin-I architecture

The top logic module of Fenglin-I architecture mainly includes the following modules, and the overall architecture is shown in Figure 1.

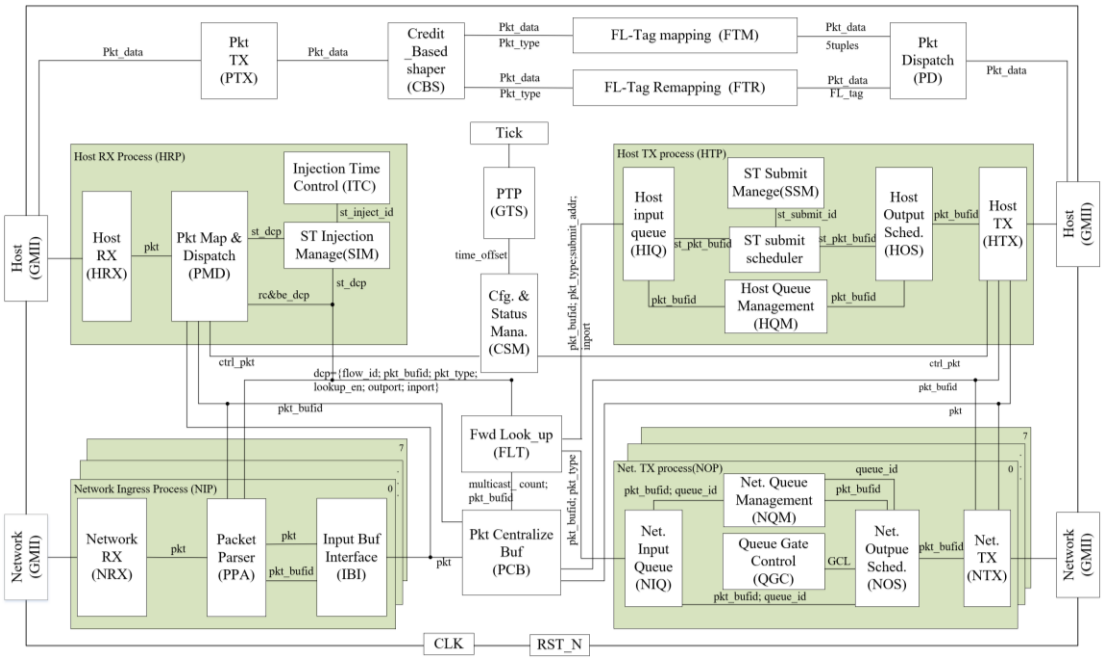

Fig.1 Top level architecture of Fenglin-I

The whole architecture is divided into five modules, (1) host receiving processing logic, (2) host sending processing logic, (3) network input processing logic, (4) network output processing logic, and (5) internal processing logic.

2.1.1Host receive processing logic

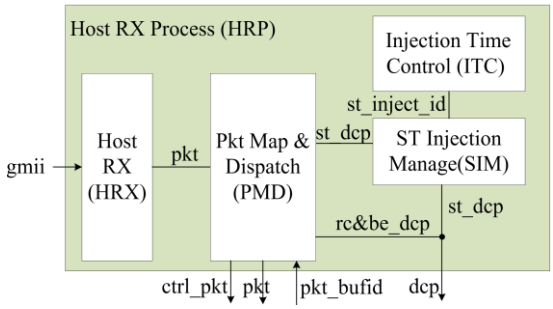

Fig.2 HRP process

HRX (host RX) module: the main function is to receive the packet sent by the host interface, switch the packet from the external clock domain to the internal clock domain of the core logic, and record the timestamp of the synchronization packet; At the same time, a register is maintained in the module to determine whether to receive or process packet according to the value of this register; In order to delete the miscellaneous packets, the module discards the miscellaneous packets according to whether the received packet carries the metadata (the metadata lives in the lower 45bit of the packet header, refer to appendix A.2 for the specific format).The packets received by the host interface carry metadata, and the addition of metadata is implemented by the software in the host.

PMD (packet mapping and dispatching) module: the main function is to construct packet descriptor (the format of packet descriptor is shown in Table 1), to monitor the BE flow and RC flow, to distinguish packet types and assign them to different destination modules. The packet descriptor comprises pkt_bufid and FL-tag (refer to Appendix A), which can identify the packet. Traffic monitoring is to monitor the transmission rate according to the pkt_type. When the remaining number of buffers is less than the threshold of RC flow, the RC flows are discarded; When the remaining number of buffers is less than the threshold value of BE flows, the BE flow is discarded. For dispatching packet, it is necessary to distinguish the type of packet from ST, BE and RC or NMAC protocol (refer to Appendix D), and then assign different types of packet descriptors to different destination modules. In addition, the PMD module also needs to discard the metadata of the packet sent from the host.

Table 1. Descriptor format

content	width	position	meaning
inject_addr/ submit_addr	5	[45:41]	The address of the memory when ST flow is injected / submitted at host_tx/host_rx.
reserve	1	[40]	reserve
inport	4	[39:36]	The input port number, used to construct the metadata.
pkt_type	3	[35:33]	Used to distinguish packet type and select queue during queue control.
flow ID/IMAC	14	[32:19]	Flow ID, used as the address index of FLT module when looking up the table.
lookup_en	1	[18]	Table lookup enable, which is used by FLT to determine whether the packet needs looking_up.
outport	9	[17:9]	Output port number, used to distinguish the output port of FLT module.
pkt_bufid	9	[8:0]	The ID number of the packet buffer

SIM (ST Injection Management) module: the main function is to write the received ST packet descriptor into corresponding buffer accordind the injection address in the FL-tag (the format of the buffer is shown in Table 2), and wait for the scheduling signal, which refers to injecting into the scheduling module, to promote reading and sending the ST packet descriptor.

Table 2. ST injection buffer packet format

name	meaning	remarks
descriptor [40:0]	The descriptor is used to identify the ST flow, which lives in the lower 41 bits of the above "descriptor packet format".	Using ram to store, the depth is 32.

ITC (Injection Time Control) module: the main function is to find the injection address according to the current time slot information (Table 3 below), determine

whether there is ST packet descriptor to be scheduled, and inform the ST injection management module to complete the extraction and injection of the ST packet.

Table 3. Injection schedule format

name	position	meaning	remarks
Value[15:0]	[15]	The valid signal of the table;	Ram is used for implementation, and the depth is 1024.
	[14:5]	Sending time-slot;	
	[4:0]	"Injection address" in FL-tag	

2.1.2Host send processing logic

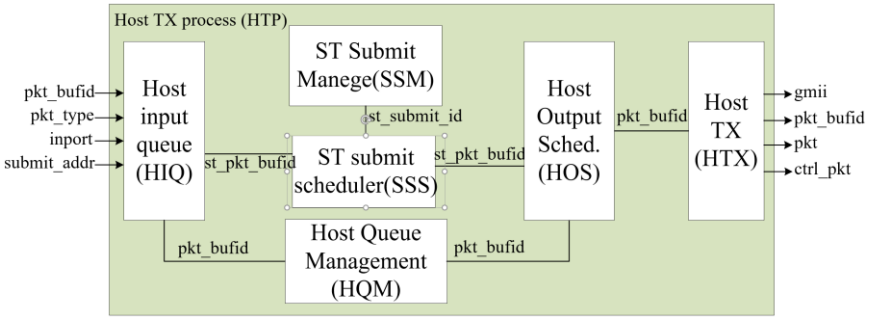

Fig.3 HSP process

HIQ (host Input Queue) module: the main function is to transmit the received pkt_bufid based on pkt_type to the corresponding destination module. The ST pkt_bufid is sent to ST submission management module for scheduling; RC and BE pkt_bufid is sent to the host queue management to wait for scheduling.

SSM (ST submission management) ST submission management module: the main function is to write the ST pkt_bufid to the memory according to the injection address identified in FL-tag (the format of the buffer is shown in Table 4), and waits for the scheduling signal from SSS module to execute the extracting and sending of pkt_bufid.

SSS (ST submition scheduling) module: the main function is to find the submission address according to the current time slot information (shown as Table 4), judge whether there is ST packet descriptors waiting to be scheduled, and inform the ST submission management module to complete the reading and sending of the pkt_bufid.

Table 4. Submission Schedule format

name	position	meaning	remarks
Value[15:0]	[15]	The significant bit of the table item;	Ram is used for implementation, and the depth is 1024.
	[14:5]	Submit slot;	

	[4:0]	"Submit addr" in FL-tag	

HQM (host queue management) host queue management module: the main function is to store the bufid of non-ST packets (the format of the pkt_bufid is shown in Table 5), and waits for the scheduling signal from the scheduling module for scheduling the pkt_bufid.

Table 5. Non-ST buffer format

name	meaning	remarks
pkt_bufid[8:0]	used to identify non ST pkt_bufid。	FIFO is used for implementation, and the depth is 256.

HOS (Host output schedule) module. The main function is to schedule pkt according to priority for scheduling and sending the pkt_bufid. When ST submits, the scheduling module schedules a ST pkt_bufid and transfer the pkt_bufid to the lower level module only when there is no-ST packet. Oney When there is no ST_pkt_bufid waiting for scheduling, the HQM module schedules the BE_pkt_bufid or RC_pkt_bufid.

HTX (host TX) module: the main function is to read the packet from the PCB module and release the corresponding pkt_bufid, transmit packet from the host interface to external logic. When reading a packet, the pkt_bufid is mapped to the packet reading address, and the packet is read from the PCB module according to this address. At the same time, this buffer is needed to release from PCB module for subsequent packets. When the CSM module needs to send the report packet, it needs to construct the frame preamble, frame start character and 8B metadata, and then directly forward the report packet to the CSM module.

2.1.3 Network input processing logic

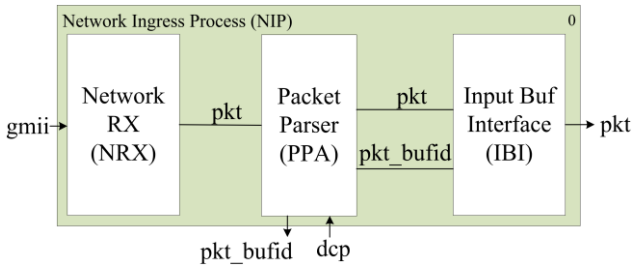

Fig.4 NIP process

NRX (network RX) module: the main function is to receive the packet from the network interface, complete the forwarding operation of the packet from the external clock domain to the internal clock domain, complete the time information record of the time synchronization packet, and write it into the FL-tag. The module maintains a register to determine whether to receive and process packet.

PPA (packet parse) module: the main function is to extract the information in DMAC (FL-tag) of frame and construct packet descriptor packet. It is also necessary to judge whether the packet has been mapped according to the Ethernet type field of the packet. If it has not been mapped, it will be sent to the host by default when constructing the descriptor. Before sending packet to the input buffer interface module, it is necessary to convert the 8-bit packets into 128-bit packets.

IBI (Input Bufm_Memory interface) module: the main function is to send the packet to the PCB module for caching. This module receives 128 bit packet from the PPA module and uses two registers for caching the packet beats. If any register has packet, it sends a write signal to the PCB gating and writes the packet received from PCB module into the other register.

2.1.4 Network output processing logic

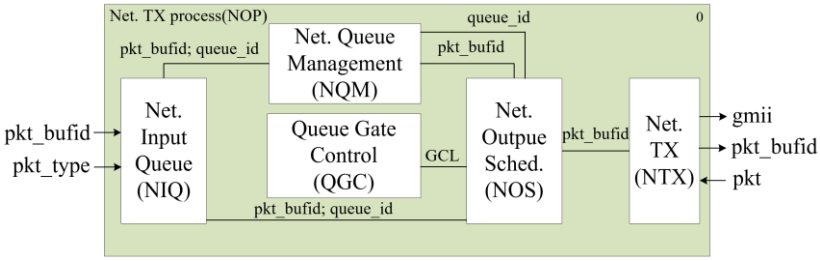

Fig.5 NOP process

NIQ (network input queue) module. The main function is to write the pkt_bufid into the network queue management module for caching. This module needs to map the packet type information and the gating information sent by the queue gating control module into pkt_bufid. Pkt_bufid and queue_ID is sent to the network queue management module for caching. At the same time, pkt_bufid and queue_ID is sent to the network output scheduling module to manage the header address of the queue. The module also needs to manage the status of the queue according to signal of the scheduling queue transmitted from the network output scheduling module.

QGC (queue gate control) module: the main function is to control the current gate state of queues in the corresponding physics port. The module reads the gate control list (GCL) according to the global clock, and sends the gating switch information of eight queues in the gating list (the format of GCL is shown in Table 8) to the network queue control module and the network output scheduling module.

Table 8. Packet format of gating list

name	meaning	remarks
gate_ctrl_vector[7:0]	Gating vector, corresponding to 8 gating status.	Using ram to achieve, the depth is 1024

NOS (network output scheduling) module. The main function is to extract the pkt from the network queue management module according to the scheduled queue information。This module needs to calculate according to the current queue information and the gating information sent by the queue gating module, get a queue with the highest priority scheduling, and get the pkt from the corresponding queue of the network queue management.

NTX (network TX) module: the main function is to read packet from packet buffer and release pkt_bufid, calculating the transparent clock of the time synchronization packet, completing the switch of the packet from the internal processing clock domain to the external PHY architecture clock domain, constructing the frame preamble and frame start character of the packet, and then transmitting it by the network interface. When reading a packet, the pkt_bufid is mapped to an address, and the packet is extracted to the PCB module according to this address. At the same time, this pkt is needed to return to the PCB module for the subsequent use of packets entering the architecture.

2.1.5 Internal processing logic

FLT (forwarding lookup table) module: the main function is to extract the flow according to the received packet descriptor, according to flow_ID to look up the forwarding table (the format of the forwarding table is shown in Table 9), and according to the output port of the table\.When the descriptor looks up_en is low (it means that the descriptor does not need to look up the table), the output field in the descriptor is directly extracted as the output port number for forwarding. When the number of bits of the output port is high, it means that the packet represented by the descriptor is a multicast pkt_bufid and pkt_type forward to multiple ports, at the same time pkt_bufid and the number of output ports are sent to the PCB module for multicast counting caching.

Table 9. Forwarding table packet format

name	meaning	remarks
outport[8:0]	Output port number (bitmap), a total of 8 network interfaces and 1 host interface.	Using ram to realize, the depth is 16K

PCB (Pkt Centralize Bufm_Memory) module: the main function is to cache all the packets that the architecture needs to forward (the format of packet cache is shown in table 10 below), and allocate a packet for each cached pkt_bufid is used to identify all idle pkt_bufid (the format of the free address buffer is shown in Table 11). For all pkt_bufid increases the register to count the number of output ports when pkt_bufid is used up and released, the pkt needs to be detected. The count value corresponding to pkt_bufid can only be released when the count value is 0. If it is not 0, it means there is

a multicast packet in pkt_bufid, and the packet has not been output from all the ports that need to be output.

The packet cache divides 64KB space into 512 packet cache blocks, and each packet cache block can cache a packet with a maximum length of 128B.

Table 10. Packet buffer packet format

Address [8:0]	Content [133:0]
0-127	The first packet buffer block
128-255	Second packet buffer block
…	..
65408-65535	512th packet buffer block

Table 11. Free address buffer packet format

name	meaning	remarks
pkt_pkt_bufid[8:0]	The ID number of the free packet cache block in the current packet cache.	It is implemented by ram with a depth of 512.

GTS (global time synchronization) time synchronization module. The main function is to extract the global time information and modify the time according to the offset value calculated by the external CPU.

Configuration and state management module of CSM. The main function is to receive the NMAC control packet from the external CPU, complete the analysis of the NMAC packet, construct the command flow to read and write the registers of each module, and construct the NMAC packet to report to the host.

3、 Detailed design

3.1 Design of HRX module

3.1.1 functional analysis

The main functions of HRX module are as follows:

(1) Remove the preamble and start character of the packet frame of the packet, and add a 1 bit head and tail identifier in front of each beat of the packet;

(2) The transmission of packet across clock domain is realized;

(3) Identify the time synchronization packet, record the 19bit receiving timestamp in FL-tag, and record the global time after synchronization in the corresponding field of PTP packet;

3.1.2 Internal function division

According to the function analysis of HRX module, the internal function division of HRX module is shown in the figure below.

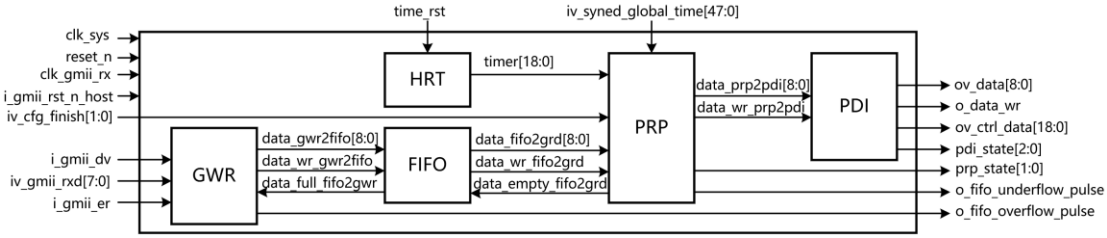

Fig9. HRX module

GWR (gmii write, gmii packet write asynchronous FIFO) module: the main function is to receive 8bit gmii packet, remove the frame preamble and frame start character, and write the packet into the asynchronous FIFO (asfifo) after adding a 1bit prefix to each beat of the packet for cross clock processing.

PRP (PTP receive process) module: the main function is to read the packet with head and tail identification from the asynchronous FIFO, and complete the register according to determine whether the current packet should be discarded. When the register is 0, all packets are discarded; When the register is 1, the non NMAC packet is discarded; When the register is 2, all ST packets are discarded; When the register is 3, all packets are processed normally. If the processed packet is PTP packet, a 19bit receiving timestamp is recorded in FL-tag, and the synchronized global time is recorded in the corresponding field of PTP packet; If the packet is not a PTP packet, it will be directly forwarded to the next module.

PDI（Packet DIstinguish）module: The main function of PDI is to identify whether the packet is a miscellaneous packet or a packet sent by the host according to the packet metadata. If it is a miscellaneous packet, it will be discarded. If it is not a miscellaneous packet, a key will be constructed according to the metadata and FL-tag. The next module can directly identify the packet according to this key and forward the packet to next module.

3.1.3 Signal definition

The signal definition of HRX module is shown in the table below.

Table 12. Signal definition of HRX module

Interface signal	width	direction	meaning
clk_sys	1	input	125MHz clock
reset_n	1	input	Reset, low effective
clk_gmii_rx	1	input	The clock signal of phy architecture receiving packet

Interface signal	width	direction	meaning
i_gmii_rst_n_host	1	input	PHY architecture receives the reset signal of packet
GWR signal definition			
iv_gmii_rxd	8	input	Packet received by PHY architecture
i_gmii_rx_dv	1	input	Valid packet signal received by PHY architecture
i_gmii_rx_er	1	input	Packet received by PHY architecture is wrong
o_fifo_overflow_pulse	1	output	FIFO overflow pulse output
Internal signal definition			
packet_gwr2fifo	9	GWR2FIFO	Packet transferred from GWR module to asynchronous FIFO
packet_wr_gwr2fifo	1	GWR2FIFO	Packet write signal from GWR module to asynchronous FIFO
packet_full_fifo2gwr	1	FIFO2GWR	Asynchronous FIFO full signal
packet_fifo2grd	9	FIFO2GRD	Packet transferred from asynchronous FIFO to PRP module
packet_rd_grd2fifo	1	GRD2FIFO	Packet read signal from PRP module to asynchronous FIFO module
packet_empty_fifo2grd	1	FIFO2GRD	Asynchronous FIFO null signal
packet_prp2pdi	9	PRP2PDI	Packet from PPP module to PDI module
packet_wr_prp2pdi	1	PRP2PDI	Packet write signal from PRP module to PDI module
PRP signal definition			
iv_syned_global_time	48	input	Global clock after synchronization
timer_rst	1	input	Synchronous reset signal of core clock and peripheral clock
o_fifo_underflow_pulse	1	output	FIFO underflow pulse output
prp_state	2	output	PRP module state machine output
PDI signal definition			
ov_packet	9	output	packet output
ov_packet_wr	1	output	Packet output valid signal
ov_ctrl_packet	19	output	Important packet output of packet
pdi_state	3	output	PDI module state machine output
iv_cfg_finish	2	input	Architecture configuration completion status

3.1.4 Processing flow

The packet packet transmitted from PHY architecture is written into asynchronous FIFO after adding head and tail identifiers to each beat;When the asynchronous FIFO is not empty, read out the packet from the FIFO. When the first beat packet of the packet is read out, judge the packet type according to the beat packet [7:5] (see table 13 below for details). If it is PTP packet, record the 19bit time stamp of the first beat packet of

metadata when it is read out from the asynchronous FIFO in FL-tag [18:0] of PTP packet, and record it in PTP sync packet and PTP delay packet_The lower 48 bits of the packet style timestamp field of the req packet record the global time after synchronization when the first packet of the metadata of the PTP packet is read out from the asynchronous FIFO;If it is not a PTP packet, whether the lower 45 bits of metadata are 0 is used to distinguish whether the packet is a miscellaneous packet or a packet sent by the host.If the packet is miscellaneous, it will be discarded;If the packet is not miscellaneous, the packet composed of important packet is constructed according to metadata and FL-tag, and is transmitted to the lower module in parallel with the packet.

Table 13. Packet types

packet[7:5]	pkt_type
000	ST packet
001	ST packet
010	ST packet
011	RC packet
100	PTP packet
101	Nmac packet
110	Be packet
111	Be packet

Table 14. FL-tagof time synchronization packet

width	name	position	describe
3	Flow type	[47:45]	Flow type.100: synchronous packet (the format of other packets is shown in the table below)
14	Flow id/IMAC	[44:31]	Static flow uses flowid, each static flow is assigned a unique flowid, dynamic flow uses IMAC address, if IMAC address is the same, it hits the same table entry in the exchange architecture.
12	Reserve	[30:19]	retain
19	Rx_timestamps	[18:0]	The local time information of the time synchronization packet received by the architecture is used to calculate the transparent clock when the architecture sends the packet.

Table 15. FL-tagof non time synchronization packet

width	name	position	describe
3	Flow type	[47:45]	Flow type.000: ST group 001: ST group 010: ST group 011: RC group 101: NMAC group 110: be group 111: be grouping
14	Flow id/IMAC	[44:31]	Static flow uses flowid, each static flow is assigned a unique flowid, dynamic flow uses IMAC address, if IMAC address

width	name	position	describe
			is the same, it hits the same table entry in the exchange architecture.
16	Seq id	[30:15]	It is used to identify the sequence number of the packet in each flow
1	Frag flag	[14]	It is used to identify the tail after fragmentation.0: intermediate packet after fragmentation 1: tail beat
4	Frag ID	[13:10]	It is used to indicate the fragment sequence number of the current fragment packet in the original packet
5	inject addr	[9:5]	Cache address of ST flow while waiting for sending schedule at source
5	submit addr	[4:0]	The ST flow caches the address when the terminal is waiting for receiving scheduling

Table 16. Metadata format carried by host when sending packet

position	width	name	meaning	remarks
[63:61]	3	Pkt_type	The type of packet.	000: ST group 001: ST group 010: ST group 011: RC group 100: PTP group 101: NMAC group 110: be group 111: be group
[60:56]	5	inject_addr	ST injection cache address	/
[55:47]	9	outport	The output port number of the packet	/
[46]	1	lookup_en	Look up enable signal	/
[45:0]	46	reserve	retain	/

3.2 Design of PMD module

3.2.1 functional analysis

The main functions of PMD module are as follows.

(1) The NMAC packet is identified and transmitted to the configuration and status management (CSM) module;

(2) Identify ST packet and monitor whether there is overflow error in ST packet;

(3) Analyze the packet and generate the packet descriptor (including the packet type pkt in metadata)_Type, when ST flow is waiting for sending schedule at source, cache address inject address, look-up enable signal lookup_en, the number of the output port of the packet, the address flowid of the packet lookup table in FL-tag, the pkt_bufid of the packet in the packet buffer, and the number of the input port of the packet;

(4) Remove the 8b metadata, use two registers to save the packet circularly, realize the bit width conversion of the packet (8bit to 134bit), and write the packet

to the packet buffer according to the address (pkt_bufid) of the allocated packet in the packet buffer;

(5) When the remaining number of pkt_bufid is less than the threshold of RC flow, the RC flow and be flow are discarded; When the remaining number of pkt_bufid is less than the threshold value of be flow, the be flow is discarded.

3.2.2 Internal function division

According to the function analysis of PMD module, the internal function of PMD module is divided as follows.

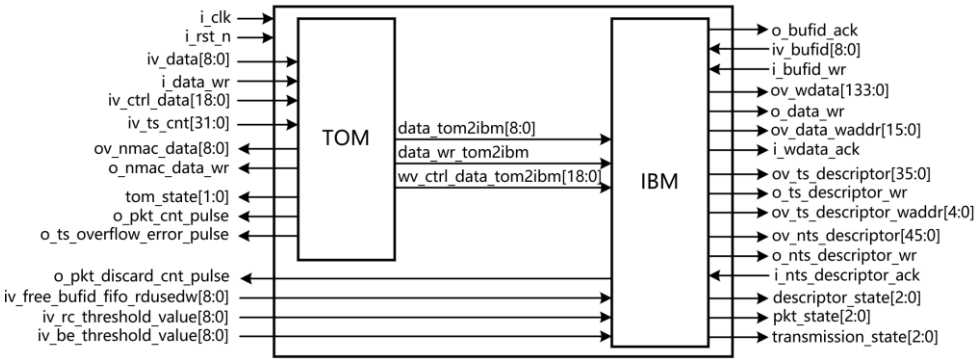

Fig.11 internal division of PMD module

Tom (time sensitive packet overflow monitor) module: the main function is to monitor whether the packet descriptors of 32 ST traffic have been read out from the descriptor buffer of SIM module, if so, forward the ST packet to IBM module; Otherwise, the ST packet will be discarded. The purpose of monitoring is to avoid this problem: the packet descriptor of a ST traffic has not been read out from the descriptor buffer of SIM module, and the host receiving module writes the packet descriptor to the same address in the descriptor buffer again to cover the packet descriptor that has not been read out. If the received packet is not ST flow, the NMAC packet is dispatched to the configuration and status management module; Other packets are normally forwarded to IBM module.

IBM (input buffer management) module: the main function is to remove the 8b metadata, realize the bit width conversion of the packet (8bit to 134bit), and use two 134bit registers to save the packet, and write the packet to the packet buffer according to the logical address (pkt_bufid) of the allocated packet in the packet buffer; At the same time, the packet descriptor.If the number of remaining bufids is less than the traffic monitoring threshold, the RC and be packets need to be discarded according to the RC traffic monitoring threshold and be traffic monitoring threshold.

3.2.3 Signal definition

The signal definition of packet mapping and dispatching (PMD) module is shown in the table below.

Table 17. Signal definition of PMD module

Interface signal	width	direction	meaning
i_clk	1	input	125MHz clock
i_rst_n	1	input	Reset, low effective
Hrx-tom signal definition			
iv_packet	9	input	Packet from HRX module to Tom module
i_packet_wr	1	input	Packet write signal from HRX module to Tom module
iv_ctrl_packet	19	input	Packet important packet input
Tom-ibm signal definition			
wv_packet_tom2ibm	9	TOM2IBM	Packet transferred from Tom module to IBM module
w_packet_wr_tom2ibm	1	TOM2IBM	Packet write signal from Tom module to IBM module
wv_ctrl_packet_tom2ibm	19	TOM2IBM	Important packet of packet from Tom module to IBM module
Tom-CSM signal definition			
ov_NMAC_packet	9	output	Nmac packet
o_NMAC_packet_wr	1	output	Valid signal of NMAC packet
Ibm-pcb signal definition			
o_pkt_bufid_ack	1	output	Pkt_bufid request allocation signal output by IBM module
iv_pkt_bufid	9	input	Pkt_bufid assigned by PCB module to IBM module
i_pkt_bufid_wr	1	input	Pkt_bufid write signal assigned by PCB module to IBM module
ov_wdata	134	output	Packet transferred from IBM module to PCB module
o_packet_wr	1	output	Packet write signal from IBM module to PCB module
ov_packet_waddr	16	output	Write address of packet from IBM module to PCB module
i_wdata_ack	1	input	The PCB module obtains the confirmation signal of the packet
Ibm-flm signal definition			
ov_nts_descriptor	46	output	Non ST packet descriptor
ov_nts_descriptor_wr	1	output	Non ST packet descriptor write signal

Interface signal	width	direction	meaning
i_nts_descriptor_ack	1	input	FLM module obtains the acknowledgement signal of non ST packet descriptor
Tom-tim signal definition			
iv_ST_cnt	32	input	There are 32 ST flows in total. Each bit of the signal corresponds to one ST flow. Every time the packet descriptor of a ST flow is read from the descriptor buffer, the corresponding position of the signal is patted high;According to the signal, the host receiving module judges whether the packet descriptor of ST traffic has been read out from RAM and looks up the table to avoid that a packet descriptor of ST traffic has not been read out from RAM. The host receiving module writes the packet descriptor to the same address in RAM again to cover the packet descriptor that has not been read out.
Ibm-tim signal definition			
ov_ST_descriptor	36	output	ST packet descriptor
o_ST_descriptor_wr	1	output	ST packet descriptor write signal
ov_ST_descriptor_waddr	5	output	Write address of ST packet descriptor
PMD_CSM signal definition			
descriptor_state	3	output	Output of descriptor state machine
Tom_state	2	output	Tom state machine output
pkt_state	3	output	Pkt state machine output
transmission_state	3	output	Transfer state machine output
o_pkt_cnt_pulse	1	output	Host receiving packet pulse signal output, used for CSM to count the number of packets received by host port
o_pkt_discard_cnt_pulse	1	output	The host discarding packet pulse signal output is used for CSM to count the number of packets discarded by the host port
o_ST_overflow_error_pulse	1	output	Host receives ST packet overflow pulse signal output
iv_free_pkt_bufid_fifo_rdusedw	9	input	Idle pkt_bufid remaining quantity input
iv_rc_threshold_value	9	input	When the number of remaining pkt_bufid is less than this value, the RC and be flows will be discarded
iv_be_threshold_value	9	input	When the number of remaining pkt_bufid is less than this value, the be flow will be discarded

3.2.4 Processing flow

3.2.4.1 ST packet coverage error monitoring module (TOM) processing flow

In order to avoid the problem that the pkt_bufid of a ST traffic packet has not been read out from the pkt_bufid buffer (RAM) of SIM module, the host receiving module writes the packet pkt_bufid to the same address in the pkt_bufid buffer again and covers the previously unread packet pkt_bufid, this module maintains a 1 bit register r for each ST packet. N = 0, 1,..., 31, when receiving a ST flow, add 1 to the corresponding register; When the pkt_bufid of an ST traffic is read out from the pkt_bufid buffer (IV). The corresponding bit in ack [31:0] is high), and the corresponding register is subtracted by 1;Only if n is 0, the corresponding ST traffic packet can be output to IBM module; Otherwise, the corresponding ST traffic packet will be discarded. If the packet is NMAC packet, the packet will be forwarded and output to CSM.

3.2.4.2 Processing flow of input cache management module (IBM)

Get pkt_bufid: every time the IBM module obtains a pkt_bufid, it sends a pkt_bufid confirmation signal to the PCB module PCB; When the IBM module consumes a pkt_bufid, it obtains a new pkt_bufid from the PCB module.

Traffic regulation: when IBM module receives a packet, it monitors the traffic according to the packet information. When the number of remaining pkt_bufid is less than the regulatory threshold of RC flow, RC flow and be flow are discarded; When the number of remaining pkt_bufid is less than the supervision threshold of be flow, the be flow is discarded; When the number of remaining pkt_bufid is 0, all packets are discarded.

Number statistics: every time the IBM module receives a packet, it outputs a received packet pulse signal to the CSM module; Similarly, every time IBM module discards a packet, it outputs a discarded packet pulse signal to CSM module.

Two registers are used to store packets circularly. Two 134 bit registers RV are designed in this module_data1[133:0]、rv_Data2 [133:0], which is used to store the packet packet circularly; After removing metadata, use RV first_Data1 to save packet, when RV_ data1 has finished storing 128 bits of effective packet of packet, 2 bits of head and tail identifier and 4 bits of invalid bytes, it can use RV_Data2 to store packet; When RV_ data2 stores the packet, it uses RV_Data1 to store packet. In this way, the packet is stored successively until the last beat of the packet.

Write the packet to the packet buffer: when RV_ data1 stores 134 bits of packet, convert the pkt_bufid to the base address of the packet in the packet buffer, and convert the RV to_Data1 is written to the base address, and the write signal is set high until the PCB module gets RV confirmation signal of data1. Put RV_Data2 is written to the next

address, and the write signal is set high until the PCB module receives confirmation signal of data2. The next packet is written into the packet buffer according to this logic until the last beat of the packet.

Generate packet descriptor：

(1) After receiving the first beat packet of the packet can be downloaded from IV_Packet [7:5] to get the packet type and Packet [4:0] obtains the cache address of the ST flow when waiting for the sending schedule at the source;

(2) From the second beat packet of received packet IV_Packet [7:0] and the third beat packet IV_Packet [7] obtains the output port number of the packet;

(3) From the third beat packet of received packet IV_Packet [6] obtains the lookup table enable signal of the packet_en；

(4) From the 16th beat of received packet packet IV_Packet [4:0], the 17th beat packet IV_Packet [7:0], the 18th beat packet IV_Packet [7:6] to obtain the flow ID (14bit) of the packet;

(5) According to the packet obtained above and the 9bit pkt_bufid allocated by the PCB module, the input port number of the packet generates the descriptor of the packet. If the packet is an ST packet, the injection addr is used as the write address, and the packet descriptor is written into the descriptor buffer of the ST packet injection management module; If the packet is not an ST packet, the packet descriptor will be output to the forwarding table lookup module, and the write signal of the packet descriptor will be set high until the confirmation signal of obtaining the packet descriptor from the forwarding table lookup module is received.

Table 18. Metadata format

position	width	name	meaning	remarks
[63:61]	3	Pkt type	The type of packet.	000: ST group 001: ST group 010: ST group 011: RC group 100: PTP group 101: NMAC group 110: be group 111: be group
[60:56]	5	inject_addr	ST injection cache address	/
[55:47]	9	outport	The output port number of the packet	/
[46]	1	lookup_en	Look up enable signal	/
[45:0]	46	reserve	retain	/

Table 19. FL-tag of time synchronization packet

width	name	position	describe
3	Flow type	[47:45]	Flow type.100: synchronous packet (the format of other packets is shown in table 20 below)
14	Flow id/IMAC	[44:31]	Static flow uses flowid, each static flow is assigned a unique flowid, dynamic flow uses IMAC address, if IMAC address is the same, it hits the same table entry in the exchange architecture.
12	Reserve	[30:19]	retain
19	Rx_timestamps	[18:0]	The local time information of the time synchronization packet received by the architecture is used to calculate the transparent clock when the architecture sends the packet.

Table 20. FL-tag of non PTP packet

width	name	position	describe
3	Flow type	[47:45]	Flow type.000: ST group 001: ST group 010: ST group 011: RC group 101: NMAC group 110: be group 111: be grouping
14	Flow id/IMAC	[44:31]	Static flow uses flowid, each static flow is assigned a unique flowid, dynamic flow uses IMAC address, if IMAC address is the same, it hits the same table entry in the exchange architecture.
16	Seq id	[30:15]	It is used to identify the sequence number of the packet in each flow
1	Frag flag	[14]	It is used to identify the tail after fragmentation.0: intermediate packet after fragmentation 1: tail beat
4	Frag ID	[13:10]	It is used to indicate the fragment sequence number of the current fragment packet in the original packet
5	inject addr	[9:5]	Cache address of ST flow while waiting for sending schedule at source
5	submit addr	[4:0]	The ST flow caches the address when the terminal is waiting for receiving scheduling

Table 21. Descriptor packet format

content	width	position	meaning
inject_addr/ submit_addr	5	[45:41]	The address of the cache when ST flow is injected / submitted.
reserve	1	[40]	retain.
inport	4	[39:36]	The input port of the packet is used to construct the metadata.
pkt_type	3	[35:33]	Packet type is used to distinguish packet type and select queue during queue control.
flow ID/IMAC	14	[32:19]	Flow ID, used for the address index of FLT module when looking up the table.
lookup_en	1	[18]	Table lookup enable, which is used by FLT to determine whether the packet needs table lookup.

content	width	position	meaning
outport	9	[17:9]	Output port number, used to distinguish the output port of FLT module.
pkt_pkt_bufid	9	[8:0]	The ID number of the packet cached in the buffer, which is used to identify each packet.

3.3 ST injection management (SIM) module design

3.3.1 functional analysis

The main functions of SIM module are as follows.

(1) Use a RAM (32_36bit) to cache the ST packet sending descriptor from the host receiving module HRX;

(2) Monitoring ST packet overflow error: determine whether to read the corresponding ST packet sending descriptor from the sending descriptor buffer according to whether each ST traffic has been written into the packet buffer at its sending time;Only when the ST traffic has been written into the packet buffer at its sending time can the sending descriptor be read from the sending descriptor buffer according to the read address of the ST packet sending descriptor sent by the ST packet injection scheduling module, and the sending descriptor is sent to the table lookup forwarding module for forwarding table lookup.

3.3.2 Internal function division

According to the function analysis of the SIM module, there is no need to further divide the SIM module.

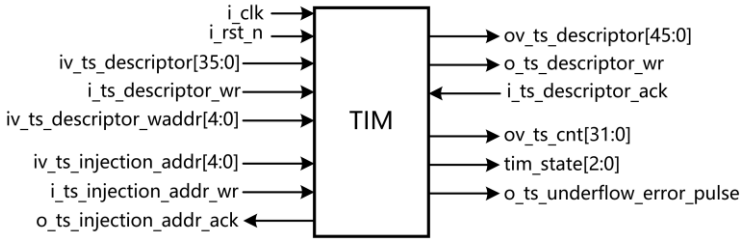

Fig.12 SIM module

3.3.3 Signal definition

The signal definition of ST packet injection management (SIM) module is as follows.

Table 22. Signal definition of SIM module

Interface signal	width	direction	meaning
i_clk	1	input	125MHz clock
i_rst_n	1	input	Reset, low effective
SIM-hrx signal definition			
iv_ST_descriptor	34	input	ST packet descriptor
i_ST_descriptor_wr	1	input	ST packet descriptor write signal
i_ST_descriptor_waddr	5	input	Write address of ST packet descriptor
ov_ST_cnt	32	output	There are 32 ST flows in total. Each bit of the signal corresponds to one ST flow. Every time the packet descriptor of a ST flow is read from the descriptor buffer, the corresponding position of the signal is patted high;According to the signal, the host receiving module judges whether the packet descriptor of ST traffic has been read out from RAM and looks up the table to avoid that a packet descriptor of ST traffic has not been read out from RAM. The host receiving module writes the packet descriptor to the same address in RAM again to cover the packet descriptor that has not been read out.
SIM-ITC signal definition			
i_ST_injection_addr_wr	1	input	ST packet sending address valid signal
iv_ST_injection_addr	5	input	ST packet sending address
o_ST_injection_addr_ack	1	output	ST packet sending address receiving confirmation signal
SIM-ltm signal definition			
ov_ST_descriptor	46	output	ST packet descriptor
o_ST_descriptor_wr	1	output	ST packet descriptor valid signal
i_ST_descriptor_ack	1	input	ST packet descriptor receives confirmation signal input
SIM-CSM signal definition			
o_ST_underflow_error_pulse	1	output	Host injection ST packet underflow pulse signal output
tim_state	3	output	SIM state machine output

3.3.4 Processing flow

The SIM module needs to instantiate a pseudo dual port RAM (32_The host receiving module HRX is responsible for writing the ST packet sending descriptor into the descriptor buffer (RAM). In order to judge whether each ST traffic has been written into the packet buffer when its sending time arrives, the SIM module determines whether the packet descriptor of each ST traffic has been written into the descriptor buffer,In this module, a 1 bit register R is maintained for each ST flow packet

descriptor_ST_cnt_N. N = 0, 1,..., 31. When a ST traffic descriptor is written to the descriptor buffer, the corresponding register is set to 1;When the descriptor of an ST traffic is read from the descriptor buffer, the corresponding register is set to 0;Only if R_ST_cnt_When n is not 0, the corresponding ST packet descriptor can be read from the descriptor buffer.

When SIM module receives the ST packet from ST packet injection scheduling module, the injection time has arrived (i.e_ST_injection_addr_WR is high) and the software static planning address IV of the ST packet when the injection time comes_ST_injection_Addr, according to IV_ST_injection_The value of addr determines whether the descriptor of the corresponding ST packet has been written into the descriptor buffer (R)_ST_cnt_Whether n is 1), if the descriptor of the corresponding ST packet has been written into the descriptor buffer (R)_ST_cnt_If n is 1), the buffer is read out from the descriptor buffer_ST_injection_The descriptor in the addr address.

3.4 ST packet injection scheduling (ITC) module design

3.4.1 functional analysis

The main functions of ITC module are as follows.

(1) With one ram (1024_16bit) to cache the injection schedule;

(2) Scheduling ST packet according to the injection schedule: read out a table entry (including the injection time of ST packet and the statically planned cache address of ST packet) from the injection schedule of ST packet. If the injection time in the table entry is the current global time, the statically planned cache address of ST packet in the table entry will be output,In other words, the ST packet corresponding to the table item is scheduled to output at its injection time;After the ST packet is checked and published, the next table entry is read out from the ST packet injection schedule. If the injection time in the table entry is not the current global time, it will stay in the table entry until the global time is the injection time in the table entry.

3.4.2 Internal function division

According to the function analysis of ITC module, the internal function division of ITC module is shown in the figure below.

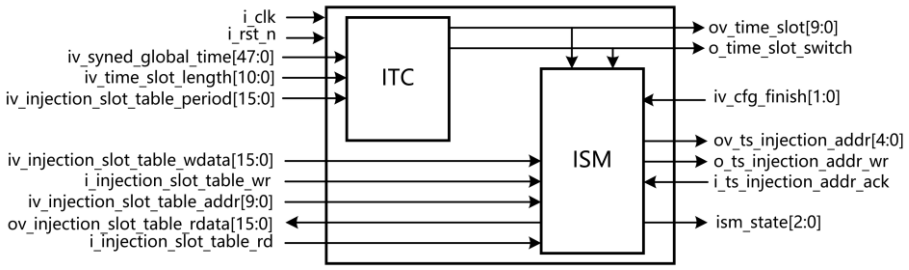

Figure 13 internal function division of ITC module

TSC (time slot calculation) / ITC: calculates the current injection time according to the global time and the length of time slot, and the ISM module searches the injection schedule according to the current injection time.

ISM**(injection schedule module)**: look up the injection time table and judge whether the injection time of the table item is equal to the current injection time; if so, output the static planning ST packet cache address of the table item; If not, it will stay in the table item until the current injection time equals the injection time of the table item.

3.4.3 Signal definition

The signal definition of ST packet injection scheduling (ITC) module is shown in the table below.

Table 23. Signal definition of ITC module

Interface signal	width	direction	meaning
i_clk	1	input	125MHz clock
i_rst_n	1	input	Reset, low effective
ITC-CSM signal definition			
iv_time_slot_length	11	input	Length of time slot, unit: US; According to the time slot length range [16us, 2ms] supported by hardware, the slot length and bit width are designed as 11bit.
iv_injection_slot_table_period	11	input	The number of time slots contained in a time slot period
iv_cfg_finish	2	input	Architecture configuration completion status
ITC-gts signal definition			
iv_syned_globe_time	48	input	Global time after synchronization
ISM-CSM signal definition			
iv_injection_slot_table_wdata	16	input	Injection schedule write packet input
i_injection_slot_table_wr	1	input	Injection schedule write signal input
iv_injection_slot_table_addr	10	input	Injection schedule read write address input
ov_injection_slot_table_rdata	16	output	Input schedule read packet output
i_injection_slot_table_rd	1	input	Injection schedule read signal input
ISM-ltm signal definition			
ov_ST_injection_addr	5	output	Descriptor read address output
o_ST_injection_addr_wr	1	output	Descriptor read address valid signal output

Interface signal	width	direction	meaning
i_ST_injection_addr_ack	1	input	LTM obtains ST packet descriptor acknowledgement signal
ITC-ism signal definition			
ov_time_slot	10	output	The current injection time in one time slot
o_time_slot_switch	1	output	Time slot switching signal
ism_state		output	State machine output

3.4.4 Processing flow

3.4.4.1 The processing flow of injection time calculation module (ITC)

This module calculates the current time slot according to the current global time and the configured time slot length. After each time slot length, it adds 1 to the time slot. When the time slot is recorded to the last time slot in the injection schedule cycle, the time slot is recalculated from 0.

In order to avoid time slot calculation error caused by time compensation in time synchronization: in a short period of time, the global time jumps from the nth time slot to the nth + 1 time slot, but due to clock compensation, the global time jumps from the nth + 1 time slot to the nth time slot, and then to the nth + 1 time slot.In this module, a 17bit counter R is designed_local_Time, after each clock cycle, R_local_Time plus one.

If the remainder of the global clock divided by the period of the injection schedule (the time taken for the injection schedule to poll once, in the unit of one time slot length) is less than the length of the time slot, then the current time is in the first time slot (i.e. time slot) in the period of the injection schedule_slot_ITC2ism is 10'd0), the counter R_local_Time clearing;Otherwise, if the synchronized global clock can divide the time slot length or R_local_If time is equal to the slot length, time_slot_ITC2ism plus 1, counter R_local_Time zero.

3.4.4.2 Processing flow of injection scheduling module (ISM)

After FLM obtains ST packet descriptor (i.e. I_ST_injection_addr _Read the content of the next table entry from the ram of the cache injection schedule, and judge whether the injection time of the table entry is equal to the current injection time: if it is equal to, output the static planning ST packet cache address of the table entry;If not, it will stay in the table item until the current injection time equals the injection time of the table item.

3.5 Design of HiQ module

3.5.1 functional analysis

The main function of HiQ (host input queue) module is to determine the type of received packet descriptor, and write the address (pkt_bufid) and input port number of

ST packet in packet buffer into descriptor buffer of SSM,Write the address of the non ST packet in the packet buffer (pkt_bufid) and the input port number inport to the queue in the host queue management module.

3.5.2 Internal function division

According to the function analysis of HiQ module, there is no need to further divide HiQ module.

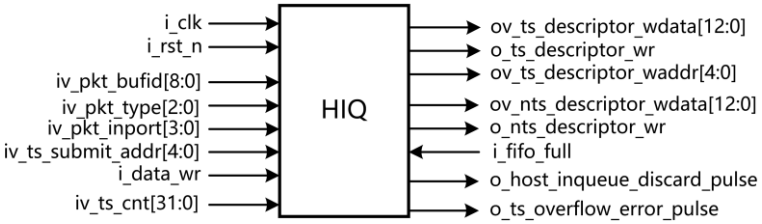

Fig.14 internal function division of HiQ module

3.5.3 Signal definition

The signal definition of HiQ module is shown in the table below.

Table 24. Signal definition of HiQ module

Interface signal	width	direction	meaning
i_clk	1	input	125MHz clock
i_rst_n	1	input	Reset, low effective
Hiq-flm signal definition			
iv_pkt_bufid	9	input	The address of the packet in the packet buffer
iv_pkt_type	3	input	Packet type: 000: ST packet 001: ST packet 010: ST group 011: RC group 100: PTP group 101: NMAC group 110: be group 111: be group
iv_ST_submit_addr	5	input	ST packet submission address of software static planning
iv_pkt_inport	4	input	The input port number of the packet
i_packet_wr	1	input	i_pkt_bufid,i_pkt_type, i_ST_submit_addr，iv_pkt_Inport packet input write signal
Hiq-tsm signal definition			
ov_ST_descriptor_wdata	13	output	The address of ST packet in packet buffer, namely descriptor
o_ST_descriptor_wr	1	output	Write signal of ST packet address in packet buffer

ov_ST_descriptor_waddr	5	output	The write address of the address of ST packet in the packet cache, that is, the static planning address of ST packet cache
iv_ST_cnt	32	input	In TSN architecture, 32 ST flows can be submitted to the host. Each bit of the signal corresponds to one ST flow. When the packet pkt_bufid of a ST flow is read out from the pkt_bufid buffer, the corresponding position of the signal is patted high;According to the signal, the host queue control module judges whether the ST traffic packet pkt_bufid has been read out from ram, so as to avoid that the ST traffic packet pkt_bufid has not been read out from RAM. The host queue control module writes the packet pkt_bufid to the same address in RAM again, and covers the packet pkt_bufid that has not been read out.
Hiq-hqm signal definition			
ov_nts_descriptor_wdata	13	output	The address of non ST packet in packet buffer, namely pkt_bufid
o_nts_descriptor_wr	1	output	Write signal of address of non ST packet in packet buffer
i_fifo_full	1	input	FIFO full signal of cache non ST packet descriptor
Hiq-CSM signal definition			
o_host_inqueue_discard_pulse	1	output	Host queue discard descriptor pulse signal output
o_ST_overflow_error_pulse	1	output	ST packet submission overflow error pulse signal output

3.5.4 Processing flow

In order to avoid the problem that the packet descriptor of a ST traffic has not been read out from the descriptor buffer (RAM), the host queue control module writes the packet descriptor to the same address in the descriptor buffer again, and covers the packet descriptor that has not been read out, this module maintains a 1 bit register when a ST flow descriptor is written to the descriptor buffer, the corresponding register is added with 1;When the descriptor of an ST traffic is read from the descriptor buffer, the corresponding register is subtracted by 1; Only if n is 0, the descriptor of the corresponding ST packet can be written from the descriptor buffer.

This module receives the address pkt_bufid, packet type pkt type and ST packet submission address submit of software static planning from forwarding table lookup module_Addr and the input port number pkt inport of the packet; If it is ST packet, and the corresponding n is 0, the address (pkt_bufid) of the packet in the packet buffer and

the input port number pkt of the packet will be changed_The inport is written to the descriptor buffer in the ST packet submission management module, and the write address is the static planning ST packet submission address of the software; If it is ST packet, and the corresponding n is 1, the address (descriptor) of the packet in the packet buffer is discarded; If it is a non ST packet and the FIFO of the non ST packet descriptor in the host queue management module is not satisfied, the address of the packet in the packet buffer (pkt_bufid) and the input port number pkt of the packet will be saved_Inport is written to FIFO of the host queue management module.

3.6 ST packet submission scheduling (TSS) module design

3.6.1 functional analysis

The main functions of TSS module are as follows:

(1) With one ram (1024_16bit) to cache the submission schedule;

(2) According to the submission schedule, the ST packet is submitted and scheduled: read out a table entry (including the submission time of ST packet and the statically planned ST packet cache address) from the ST packet Submission Schedule. If the submission time in the table entry is the current global time, the statically planned ST packet cache address in the table entry is output,In other words, the ST packet corresponding to the table item is scheduled to be output at its submission time;If the port output to the host is free, read the next table entry from the ST packet Submission Schedule. If the submission time in the table entry is not the current global time, stay in the table entry until the global time is the submission time in the table entry.

3.6.2 Internal function division

According to the function analysis of TSS module, the internal function division of TSS module is shown in the figure below.

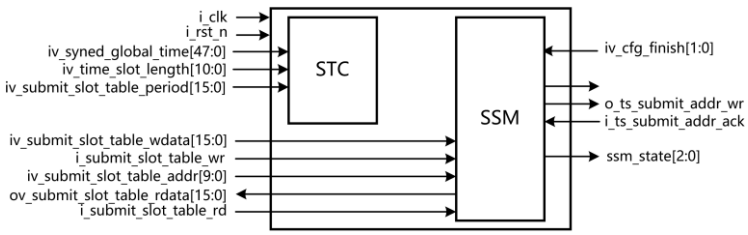

Fig.15 internal function division of ITC module

STC (submit time calculation) instantiation module: according to the global time and the length of submission slot, the current submission time is calculated, and

the SSM module searches the submission schedule according to the current submission time.

SSM (submit schedule module) module: look up the submission time table and judge whether the submission time of the table item is equal to the current submission time; if so, output the static planning ST packet cache address of the table item; If not, it will stay in the table item until the current submission time equals to the submission time of the table item.

3.6.3 Signal definition

The signal definition of ST packet submission scheduling (TSS) module is shown in the table below.

Table 25. Signal definition of TSS module

Interface signal	width	direction	meaning
i_clk	1	input	125MHz clock
i_rst_n	1	input	Reset, low effective
Stc-CSM signal definition			
iv_time_slot_length	11	input	Length of time slot; According to the time slot length range [4us, 1ms] supported by hardware, the slot length and bit width are designed as 11bit.
iv_submit_slot_table_period	11	input	The number of time slots contained in a time slot period
iv_cfg_finish	2	input	Architecture configuration completion status
Stc-gts signal definition			
iv_syned_globe_time	48	input	Global time after synchronization
Ssm-CSM signal definition			
iv_submit_slot_table_wdata	16	input	Injection schedule write packet input
i_submit_slot_table_wr	1	input	Injection schedule write signal input
iv_submit_slot_table_addr	10	input	Injection schedule read write address input
ov_submit_slot_table_rdata	16	output	Input schedule read packet output
i_submit_slot_table_rd	1	input	Injection schedule read signal input
Ssm-ltm signal definition			
ov_ST_submit_addr	5	output	Descriptor read address output
o_ST_submit_addr_wr	1	output	Descriptor read address valid signal output
i_submit_ack	1	input	LTM obtains ST packet descriptor acknowledgement signal

3.6.4 Processing flow

3.6.4.1 Processing flow of submit time calculation module (STC)

This module calculates the current submission time slot according to the current global time and the configured submission time slot length. After each submission time slot length, it adds 1 to the submission time slot. When the submission time slot is recorded to the last submission time slot in the injection schedule cycle, the submission time slot is recalculated from 0.

In order to avoid the calculation error of submission time slot caused by time compensation in time synchronization: in a short period of time, the global time jumps from the nth submission time slot to the nth + 1 submission time slot, but due to clock compensation, the global time jumps from the nth + 1 submission time slot to the nth submission time slot, Then jump to the N + 1 submission slot. In this module, a 17bit counter R is designed to record local_time, after each clock cycle, R_local_Time plus one.

If the remainder of the global clock divided by the period of submitting time table (the time taken to submit time table polling once, in the unit of the length of a submission time slot) is less than the length of the submission time slot, then the current time is in the first submission time slot (i.e. time slot) in the period of submitting time table_slot_ITC2ism is 10'd0), the counter R_local_Time clearing; Otherwise, if the global clock after synchronization can divide the submission slot length or R_local_ time is equal to the length of the submission slot, time_slot_ITC2ism plus 1, counter R_local_Time zero.

3.6.4.2 The processing flow of submission scheduling module (SSM)

After the pkt_bufid of the previous ST packet is scheduled, read the contents of the next table item from the ram of the cache Submission Schedule, and judge whether the submission time of the table item is equal to the current submission time: if it is equal to, output the static planning ST packet cache address of the table item; If not, it will stay in the table item until the current submission time equals to the submission time of the table item.

3.7 ST submission Management (SSM) module design

3.7.1 functional analysis

Main functions of SSM module:

(1) Use a RAM (32_13 bit) to cache the cache address of ST packet in the packet cache area, that is, pkt_bufid;

(2) Manage the cache address (pkt_bufid) of ST packet: according to whether each ST traffic has been written into the packet cache at the time of submission, decide whether to read the pkt_bufid of the corresponding ST packet from the pkt_bufid cache;Only when the ST traffic has been written into the packet buffer at the time of its submission, can it read the pkt_bufid from the pkt_bufid buffer according to the read address of the ST packet pkt_bufid sent by the ST packet submission scheduling module, and send the pkt_bufid to the host output scheduling module for scheduling.

3.7.2 Internal function division

According to the function analysis of SSM module, there is no need for further division in the interior of SSM module.

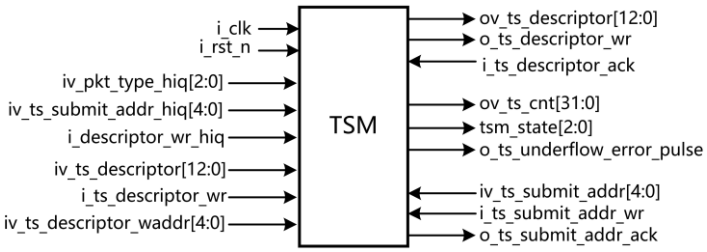

Figure 16 internal function division of SSM module

3.7.3 Signal definition

The signal definition of ST packet submission Management (SSM) module is shown in the table below.

Table 26. Signal definition of SSM module

Interface signal	width	direction	meaning
i_clk	1	input	125MHz clock
i_rst_n	1	input	Reset, low effective
Tsm-hiq signal definition			
iv_pkt_type_hiq	3	input	Packet type signal input
iv_ST_submit_addr_hiq	5	input	ST packet submission address input
i_descriptor_wr_hiq	1	input	Packet descriptor input valid signal
SIM-hrx signal definition			
iv_ST_descriptor	34	input	ST packet descriptor
i_ST_descriptor_wr	1	input	ST packet descriptor write signal
i_ST_descriptor_waddr	5	input	Write address of ST packet descriptor
ov_ST_cnt	32	output	There are 32 ST flows in total. Each bit of the signal corresponds to one ST flow. Every time the packet descriptor of a ST flow is read from the descriptor buffer, the

Interface signal	width	direction	meaning
			corresponding position of the signal is patted high;According to the signal, the host receiving module judges whether the packet descriptor of ST traffic has been read out from RAM and looks up the table to avoid that a packet descriptor of ST traffic has not been read out from RAM. The host receiving module writes the packet descriptor to the same address in RAM again to cover the packet descriptor that has not been read out.
SIM-ITC signal definition			
i_ST_submit_addr_wr	1	input	ST packet sending address valid signal
iv_ST_submit_addr	5	input	ST packet sending address
o_ST_submit_addr_ack	1	output	ST packet sending address receiving confirmation signal
SIM-ltm signal definition			
ov_ST_descriptor	46	output	ST packet descriptor
o_ST_descriptor_wr	1	output	ST packet descriptor valid signal
i_ST_descriptor_ack	1	input	ST packet descriptor receives confirmation signal input
SIM-CSM signal definition			
o_ST_underflow_error_pulse	1	output	Host submits ST packet underflow pulse signal output
tsm_state	3	output	SSM state machine output

3.7.4 Processing flow

The SSM module needs to instantiate a pseudo dual port RAM (depth 32, bit width 13). The host queue control module HiQ is responsible for writing the pkt_bufid of ST packet into the pkt_bufid buffer (RAM). The SSM module determines whether each ST traffic has been written into the pkt_bufid buffer when its sending time arrives, that is, whether the pkt_bufid of each ST traffic has been written into the pkt_bufid buffer,In this module, for each ST traffic packet pkt_bufid, a 1 bit register r of ST packet not coming error is maintained_ST_cnt_N. N = 0, 1,..., 31. When the pkt_bufid of an ST traffic is written to the pkt_bufid buffer, set the corresponding register to 1;When the descriptor of an ST traffic is read from the descriptor buffer, the corresponding register is set to 0;Only if R_ST_cnt_When n is not 0, the corresponding ST packet pkt_bufid can be read from pkt_bufid buffer.

When SSM module receives ST packet and submits the ST packet from scheduling module, the submission time has arrived (i.e_ST_submit_addr_WR is high) and the

software statically planned address IV of the ST packet submitted at the time of arrival_ST_submit_Addr, according to IV_ST_submit_The value of addr determines whether the pkt_bufid of the corresponding ST packet has been written to the descriptor buffer (R)_ST_cnt_Whether n is 1), if the pkt_bufid of the corresponding ST packet has been written into the pkt_bufid buffer (R)_ST_cnt_If n is 1), the buffer memory is read out from the pkt_bufid buffer_ST_submit_In addr address, the pkt_bufid and its effective signal are transmitted to the host output scheduling module HOS all the time;When ST packet pkt_bufid from HOS is received and scheduled, pkt_bufid output valid signal is set low.

3.8 Design of host queue management (HQM) module

3.8.1 functional analysis

The main function of HQM module is to use FIFO to cache the descriptors of non-ST messages (the logical address of non-ST messages in the message set cache module, namely bufid; And the input port number of the message).

3.8.2 Internal function division

According to the functional analysis of HQM module, there is no need to further divide the internal of HQM module.

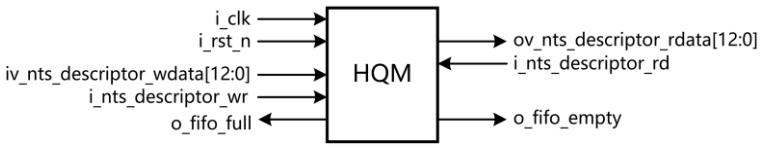

Figure 17 internal function division of HQM module

3.8.3 Signal definition

The signal definition of host queue management module (HQM) is shown in the table below.

Table 27. Signal definition of HQM module

Interface signal	Bit width	direction	meaning
i_clk	1	input	125MHz clock
i_rst_n	1	input	Reset, low effective
Hqm-hiq signal definition			
o_fifo_full	1	output	FIFO full signal
iv_nts_descriptor_wdata	13	input	Descriptor of non-ST message
i_nts_descriptor_wr	1	input	Descriptor write signal of non-ST message
Hqm-hos signal definition			

o_fifo_empty	1	output	FIFO null signal
ov_nts_descriptor_rdata	13	input	Descriptor of non-ST message
i_nts_descriptor_rd	1	input	Descriptor read signal of non-ST message

3.8.4 Processing flow

In this module, a FIFO with 13 bit width and 256 bit depth is instantiated to cache the descriptor of non-ST message.

3.9 Design of host output scheduling (HOS) module

3.9.1 functional analysis

Main functions of HOS module: schedule the message according to the priority scheduling strategy, first schedule the bufid of St message, and then schedule the bufid of non-ST message; Output the scheduled bufid to the host sending module HTX;

3.9.2 Internal function division

According to the function analysis of HOS module, there is no need to further divide the HOS module.

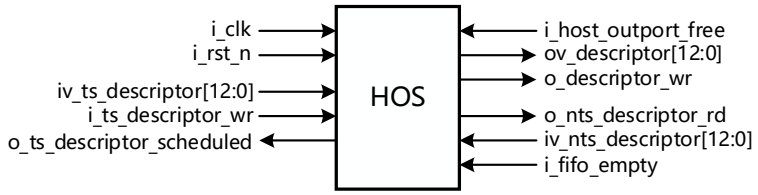

Figure 18 internal function division of HOS module

3.9.3 Signal definition

The signal definition of host output scheduling (HOS) module is shown in the table below.

Table 28. Signal definition of HOS module

Interface signal	Bit width	direction	meaning
i_clk	1	input	125MHz clock
i_rst_n	1	input	Reset, low effective
Hos-hqm signal definition			
i_fifo_empty	1	input	FIFO null signal
iv_nts_descriptor	13	input	Descriptor of non-ST message
o_nts_descriptor_rd	1	output	Descriptor read signal of non-ST message
Definition of hos-tsm signal			
iv_ts_descriptor	13	input	St message descriptor
i_ts_descriptor_wr	1	input	St message descriptor write signal

o_ts_descriptor_scheduled	1	output	The st message descriptor is sent to the scheduling module to schedule the next st message
Definition of hos-htx signal			
ov_descriptor	13	output	Message descriptor
o_descriptor_wr	1	output	Message descriptor write signal
i_host_outport_free	1	input	Host send port idle signal
Definition of hos-csm signal			
hos_state	2	output	Hos state machine output

3.9.4 Processing flow

Scheduling message descriptors according to priority scheduling policy: the HOS module can schedule the message descriptor only when the sending port of the host is idle. The following describes the processing flow of the message by the HOS module when the sending port of the host is idle. If the descriptor of St message has been sent (i.e. the descriptor of St message writes when signal i_ts_descriptor_wr is high), the ST message descriptor scheduling output is sent to the host sending module HTx, and the ST message descriptor scheduling signal is sent to the ST message submission management module TSM.

If the descriptor of ST message is not transmitted (i.e. the descriptor of St message writes when signal i_ts_descriptor_wr is low), and FIFO is not empty in the host queue management module (signal i_fifo_empty is zero),reads a non-ST message descriptor from the FIFO of the host queue management module and send it to the host sending module HTX.

3.10 Design of HTX module

3.10.1 Function analysis

The main functions of HTX module are as follows.

(1) The received bufid is converted into address and the message is read out from the message buffer;

(2) Add 7b preamble, 1b frame start character and 8b metadata before message output, and control the frame interval between messages at least 12 beats;

(3) In the metadata of PTP message, record the global time stamp of 48 bit synchronization sent by PTP message from the interface;The transparent clock of PTP message in the architecture is calculated and recorded in TSNtag of PTP message;

(4) If the CSM module needs to send a message, it constructs the message preamble, frame start character and 8b metadata to forward the message transmitted by CSM from the port;

(5) The 128 bit message per beat is converted into 8 bit message per beat and output to PHY architecture.

3.10.2 Internal function division

According to the function analysis of HTX module, the internal functions of HTX module are divided as follows.

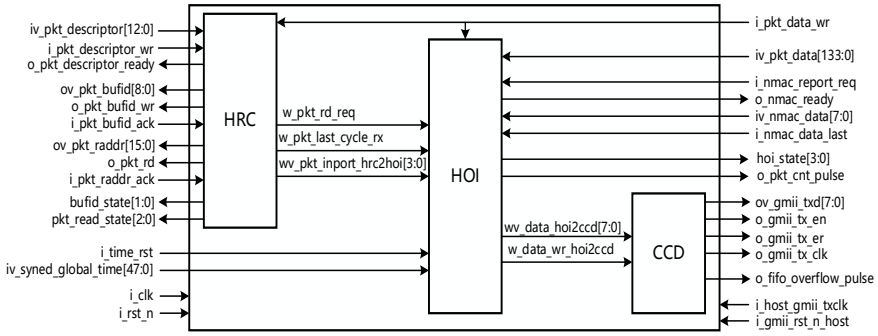

Figure 19 HTX module partition diagram

HRC (host read control) module: the main function is to transmit the received pkt_bufid is converted into address and the message data is read out from the message buffer;And the pkt_ bufid is sent to the message centralized cache module PCB for release.

HIO (host interface output) module: the main function is to use two register cycles to cache the message. When any of the registers is empty, it will send a request to read a beat message to the message reading control module; Before outputting the message, it is necessary to transmit 7B preamble, 1B frame start character and 8B metadata. When the Hoi receives the first beat data of the message from the PCB module, it will determine the type of the message. If it is PTP message, it will record the global time stamp of 48 bit synchronization of PTP message sent from the interface in metadata,The transparent clock of PTP message in the architecture is calculated and recorded in TSNtag of PTP message; If the CSM module needs to send a message, it will receive and transmit the message constructed by CSM after constructing the preamble, frame start character and 8b metadata; The frame interval between each packet is controlled at least 12 beats.

CCD (cross clock domain) clock domain switching module: The main function is to switch 9bit data from internal processing clock domain to external PHY clock domain, and finally send it to the network from gmii interface.

3.10.3 Signal definition

The signal definition of HTX module is shown in the table below.

Table 29. Signal definition of HTX module

Interface signal	Bit width	direction	meaning
i_clk	1	input	125MHz clock

Interface signal	Bit width	direction	meaning
i_rst_n	1	input	Reset, low effective
i_host_gmii_tx_clk	1	input	The clock signal of sending message in PHY architecture
i_gmii_rst_n_host	1	input	PHY architecture receives the reset signal of message
Definition of prc-hos signal			
iv_pkt_descriptor	13	input	Message descriptor data input
i_pkt_descriptor_wr	1	input	Message descriptor data valid signal input
o_pkt_descriptor_ready	1	output	Message descriptor data receiving preparation signal output
Prc-pcb signal definition			
ov_pkt_raddr	16	output	Read address of message
o_pkt_rd	1	output	Read signal of message
i_pkt_raddr_ack	1	input	The message centralized cache module obtains the confirmation signal of message read address
ov_pkt_bufid	9	output	Pkt to be released_bufid
o_pkt_bufid_wr	1	output	Pkt to be released_bufid write signal
i_pkt_bufid_ack	1	input	Packet centralized cache module obtains pkt_bufid confirmation signal
iv_pkt_data	134	input	Data from message centralized cache module
i_pkt_data_wr	1	input	Data write signal from message centralized cache module
Definition of hoi-gts signal			
i_time_rst	1	input	Synchronization signal of internal core clock and external clock
iv_syned_globe_time	48	input	Global time after synchronization
Definition of ccd-phy signal			
ov_gmii_txd	8	output	Gmii data output
o_gmii_tx_en	1	output	Gmii data output valid signal
o_gmii_tx_er	1	output	Gmii data output error signal
o_gmii_tx_clk	1	output	Gmii data output clock signal
HTX-CSM			
hoi_state	4	output	Hoi state machine output
bufid_state	2	output	bufid state machine output
pkt_read_state	3	output	PKT_Read state machine output
o_pkt_cnt_pulse	1	output	Host port sending message pulse signal output
o_fifo_overflow_pulse	1	output	FIFO overflow error pulse signal output of host sending module
Signal definition between modules			

Interface signal	Bit width	direction	meaning
w_pkt_rd_req	1	HOI2HRC	Message reading request signal;Initialize to 1, that is, the message has read request.
w_pkt_last_cycle_rx	1	HOI2HRC	The message reads the last data signal
wv_data_hoi2ccd	8	HOI2CCD	Message data
w_data_wr_hoi2ccd	1	HOI2CCD	Message data valid signal
wv_pkt_inport_hrc2hoi	4	HRC2HOI	Message input port number
w_pkt_descriptor_ready	1	HRC	Message descriptor preparation signal

3.10.4 Processing flow

3.10.4.1 Processing flow of host read control module (HRC)

Host read control module（HRC) received pkt_bufid and if Hoi sends a message read request, pkt_bufid is converted into the base address base addr for message reading, and pkt is converted into the base address base addr for message reading pkt_bufid is output to the message centralized cache module to release the pkt_bufid write signal is set high until the message centralized cache module returns to get pkt_bufid only when confirms the signal, will pkt_bufid write signal is set low; At the same time, the message read address base addr is output to the message centralized cache module to read the message, and the read signal of the message is always set high until the message centralized cache module returns the message read address confirmation signal, then the read signal of the message is set low; If the Hoi does not send a message read request, it will be in the waiting state until the Hoi sends a message read request. When the Hoi module sends a message read request again, it adds 1 to the read address of the message and outputs it to the message centralized cache module to read the message. The read signal of the message is always set high until the message centralized cache module returns the message read address confirmation signal, then the read signal of the message is set low. According to this logic, the message is read from the message centralized cache module. Until the whole message is read. When the PRC module receives the message transmission completion signal from the Hoi module, it waits for the next input of pkt_bufid and next message reading request.

3.10.4.2 Processing flow of host interface output module (Hoi)

In order to calculate the transparent clock of PTP message and the offset value between master clock and slave clock in software, a 19bit local clock is maintained in host interface output module. The host interface output module HOI module identifies the received message. If it is a PTP message, the 48 bit global time stamp of PTP message sent from the interface is recorded in metadata, and the transparent clock of PTP message in the architecture is calculated and recorded in the TSNtag of PTP message.

The HOI maintains two 134 bit registers r_data1[133:0],r_data2 [133:0], the received message is buffered with two register cycles: firstly use r_data1 stores 134 bits

of data, and then r_data2 stores 134 bits of message data, and then r_data1 stores 134 bits of data,... According to this logic, using r_data1,r_data2 circulates data.As long as one of the two register is empty (no stored data or all stored data is output to PHY), it will send message reading request to HRC module. At the beginning, the Hoi module sends a message reading request to the PRC module. After receiving the message, it starts to construct the preamble of 7B and the frame start character of 1B and output them. Then, it adds the metadata output of 8B, and then starts to transmit the received message. Each beat outputs 8bit data. When the 134bit data of the last beat of the message is received, Send message to HRC module, the last beat has received the signal (HRC module no longer reads message from message centralized cache module), send the signal to HOS module to prepare to schedule the next message. After the whole message is output, after 12 beats, if the data of the next message has been received, the preamble of the next message can be transmitted.

When the CSM module sends a report request, it starts to construct the frame preamble, frame start and 8b metadata. When the construction is completed, it needs to send an nmac confirmation signal in advance to inform the CSM to send the report message. When the CSM module completes the construction of the frame preamble, frame start and 8B metadata, it can directly receive and transmit the report message sent by the CSM until all transfers are complete.

Table 30. Metadata format carried by architecture when uploading message

position	Bit width	name	meaning	remarks
[63:16]	48	ts	Global time information of synchronous message	It is used to record the global time information of the req message received by the master clock device and the global time information of the sync message received by the clock device
[15:12]	4	inport	The input port number of the message	/
[11:0]	12	reserve	reserve	/

3.11 Design of NRX module

3.11.1 functional analysis

Receiving gmii interface physical side signal, removing the preamble and frame start delimiter of Ethernet frame header;

Although both the gmii interface clock and the system clock are 125MHz, they may have the same frequency and different phases, so it is necessary to process the input signal across clock domain and synchronize it to the input system clock CLK_Sys. Asynchronous FIFO is usually used to process high-speed data signals. The asynchronous FIFO is based on register (9bit_16), so it need not to be very deep.

The 19 bit register is used to count the system clock, and the receiving time stamp is recorded when the first beat of data is output.

According to the network interface configuration type and configuration completion status, choose to discard or forward the input packet data. When the port type is in the standard Ethernet message processing mode, if the configuration completion status is 0, the message cannot be sent through; If the configuration completion status is 1, 2 or 3, all input groups can pass. When the port type is mapped message processing mode, if the configuration completion status is 0, the message cannot be processed; if the configuration completion status is 1, the message can only be configured; If the configuration completion status is 2, all non-TSN messages can be sent through; If the configuration completion status is 3, all messages will be sent through.

For each passing message, a valid pulse signal is generated for counting and statistics by configuration and status management module.

3.11.2 Internal function division

According to the function analysis of NRX module, the internal functions of NRX module are divided as follows.

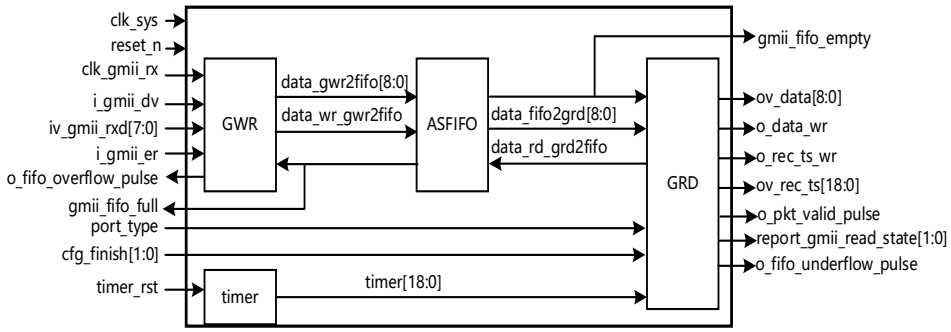

Figure 20 NRX module partition diagram

GWR (gmii interface, Gmii write data) module: the main function is to convert the received 8bit gmii data into 9bit pkt with head and tail identification_data； It is written into asynchronous FIFO for cross clock processing.

GRD (gmii interface, Gmii read data) module: the main function is to read and output 9bit pkt with head and tail identification from asynchronous FIFO under the system clock_And the receiving time stamp at the time of recording and outputting the first beat.

3.11.3 Signal definition

The signal definition of NRX module is shown in the table below.

Table 31. NRX module interface signal meaning

Interface signal	Bit width	direction	meaning
External interface signal			

Interface signal	Bit width	direction	meaning
clk_sys	1	input	System clock, 125MHz
reset_n	1	input	System asynchronous reset signal, low effective
GWR			
clk_gmii_rx	1	input	Gmii receive clock, 125MHz
i_gmii_dv	1	input	Gmii interface receives data write signal
iv_gmii_rxd	8	input	Gmii interface receives data
i_gmii_er	1	input	Gmii interface receives error signal
o_fifo_overflow_pulse	1	output	FIFO overflow signal, when FIFO is full, the signal is set to 1
i_gmii_rst_n	1	input	Write the reset signal of the module
GWR-ASFIFO			
gmii_fifo_empty	1	FIFO2GWR	gmii_FIFO null signal
gmii_fifo_rdfifo	1	FIFO2GWR	gmii_FIFO full signal
GRD			
port_type	1	input	Port configuration type.1: Non cooperative type, processing standard Ethernet message;0: cooperation type, processing mapped TSN message.
cfg_finish	2	input	Configuration completion status.0 represents that the port does not receive any messages, and 1 represents that the port can receive configuration messages.2 means that all non-ST messages can be received.3 means to receive all messages, which is only valid in the mode of processing mapped message port.
ov_data	9	output	Packet data with head and tail control bits
o_data_wr	1	output	Packet data write signal
ov_rec_ts	19	output	Time stamp of the first beat of the received message recorded
o_pkt_valid_pulse	1	output	Message valid pulse signal, used to input message count
o_fifo_underflow_pulse	1	output	When FIFO is read empty, the pulse is set to 1
gmii_read_state	2	output	gmii_Read state machine status register
Timer GRD (internal)			
timer	19	timer2GRD	Absolute count time, used to calculate transparent clock
data_gwr2fifo	9	GWR2FIFO	Write data state machine writes asynchronous FIFO packet data with head and tail control bits
data_wr_gwr2fifo	1	GWR2FIFO	Write data state machine write asynchronous FIFO packet data write signal

Interface signal	Bit width	direction	meaning
data_fifo2grd	9	FIFO2GRD	Read data state machine reads asynchronous FIFO packet data
data_rd_fifo2grd	1	GRD2FIFO	Read data state machine reads asynchronous FIFO packet data read signal
gmii_fifo_full	1	FIFO2GWR	Full signal of asynchronous FIFO
timer			
timer_rst	1	input	Timer clock reset signal

The format of pkt_data passed between modules is as follows.

Pkt transmitted between modules_ data bit width is 9 bits, including 1 bit header and 8 bit message data. Head and tail flag: 1 represents the header / tail data of the message;0 identifies the intermediate data.The details are shown in the figure below.

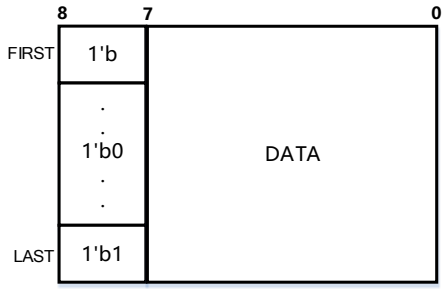

Figure 21 pkt data format

3.11.4 Processing flow

It is realized by two state machines and one asynchronous FIFO.

- **Write data state machine**

Receiving clock CLK on gmii interface_gmii_In Rx, the input signal needs to be buffered through the register for one beat in order to determine the tail position of the message.

The write data state machine judges whether the gmii interface receives the data write signal effectively_gmii_DV is highly effective, it needs to delay one beat and enter the state of starting to write FIFO in the next beat;When writing data to FIFO, first judge whether FIFO is full. If FIFO is full and it is the last beat, jump to start state. If it is not the last beat, jump to error state;The data transmitted by the current beat is the data saved by the previous beat. If it is the first beat or the last beat, it is at the highest position 1 of the beat.

- **Read data state machine**

In the system clock CLK_Sys, judge whether FIFO is empty, if not, delay 3 beats to start reading data, if it is empty, stop at the initial state.

When reading data, first transmit the first beat, judge whether it is the first beat according to the highest bit of each input beat (the highest position 1 represents the first beat and the last beat), and then transmit or discard the message according to the port type and the configuration completion register. When the port type is 1, the configuration completion register is 1, 2, and 3, the message can be transmitted normally (output to the host port). When the port type is 0, the message can be transmitted normally,When the register is set to 1, only nmac messages can be transmitted, non-ST messages can be transmitted when the register is set to 2, and all messages can be transmitted when the register is set to 3.

When the packet data of the first beat is output, the receiving time stamp is sent out. If the FIFO is empty but the last beat is not read out in the transmission process, it indicates that the message has been broken, then the FIFO is empty pulse is sent out.When reading out the tail beat, the state machine jumps to the initial state and the message transmission ends.

For each passing message, a valid pulse signal is generated for counting and statistics by configuration and status management module.

- **Time Counter**

In the system clock clk_sys, count the clock to get the absolute time; When timer_RST is high effective, the clock count is cleared; When the count reaches 4ms (the counter value is the number of beats, when the count reaches 0x7a120 beats, it represents 4ms), the counter is cleared.

3.12 Design of frame paser (FPA) module

3.12.1 Functional analysis

According to the network interface configuration mode, two different processing modes are selected: standard Ethernet message processing mode and mapped Ethernet message processing mode. By default, it is mapped Ethernet message processing mode.

In the standard Ethernet message processing mode, the table lookup enable bit of the descriptor is directly lookup_en is set to 0, the output port number is set to the host port number (9’b100000000, bitmap format), and the message type is pkt_type fill in 3’b111 to send it directly to the host for mapping.

In the mapped Ethernet message processing mode, the table lookup enable bit of the descriptor is called the look_up field is set to 1 and the output port number is set to 0 for table lookup forwarding. For other field information of descriptor, PTP message and other types (ST, RC, BE, NMAC) message need to be further distinguished. According to the flow type field of TSNtag (the end system has replaced DMAC with classification mapping result, the specific format is shown in the table below), if it is 3’b010, it is PTP message, otherwise it is other type. For PTP message, pkt is obtained from TSNtag_type (flow type), flow_id; For other types, get pkt_type, Flow id, submit addr from TSNtag。

Generate descriptor information. In the seventh beat, the above pkt_type, flow_id, submit addr, lookup_en, outport, and the input cache base address pkt_bufid will be calculated and input port number parameter values are filled in the corresponding fields of descriptor and output (the specific format is shown in the table below).If the new cache base address cannot be obtained, the message discarding pulse signal is output for the configuration and status management module to count and count the discarded messages.

Table 32. Descriptor data format

content	Bit width	position	meaning
inject_addr/ submit_addr	5	[45:41]	The address of the cache when st stream is injected / submitted.
Reserve	1	[40]	reserve.
inport	4	[39:36]	The input port of the message is used to construct the metadata.
pkt_type	3	[35:33]	Message type is used to distinguish message type and select queue during queue control.
flow ID/IMAC	14	[32:19]	Stream ID, used for the address index of FLT module when looking up the table.
lookup_en	1	[18]	Table lookup enable, which is used by FLT to determine whether the message needs table lookup.
outport	9	[17:9]	Output port number, used to distinguish the output port of FLT module.
pkt_bufid	9	[8:0]	The ID number of the message cached in the buffer, which is used to identify each message.

If the port is configured to the standard Ethernet processing mode, the packet should be discarded according to the unmapped traffic monitoring threshold; If the port is configured to map message processing mode, it is necessary to discard RC message and be message according to RC traffic monitoring threshold and be traffic monitoring threshold.

PTP message regenerates TSNtag information. Fill in the input receiving time stamp in the corresponding field of PTP message TSNtag and output it (the specific format is shown in the table below).

Concatenate 134 bit packet data. The 8-bit packet data is spliced into 128 bit data, and the 2-bit head and tail control bits and 4-bit invalid byte bits are added.

Table 33. TSNtag of time synchronization message

Bitwidth	name	position	describe
3	Flow type	[47:45]	Flow type.100: synchronous message (the format of other messages is shown in the table below)
14	Flow id/IMAC	[44:31]	Static flow uses flowid, each static flow is assigned a unique flowid, dynamic flow uses IMAC address, if IMAC address is the same, it hits the same table entry in the exchange architecture.
12	Reserve	[30:19]	reserve

19	Rx_timestamps	[18:0]	The local time information of the time synchronization message received by the architecture is used to calculate the transparent clock when the architecture sends the message.

Table 34. TSNtag of non time synchronization message

Bitwidth	name	position	describe
3	Flow type	[47:45]	Flow type.000: St group 001: St group 010: St group 011: RC group 101: nmac group 110: be group 111: be grouping
14	Flow id/IMAC	[44:31]	Static flow uses flowid, each static flow is assigned a unique flowid, dynamic flow uses IMAC address, if IMAC address is the same, it hits the same table entry in the exchange architecture.
16	Seq id	[30:15]	It is used to identify the sequence number of the message in each stream
1	Frag flag	[14]	It is used to identify the tail after fragmentation. 0: intermediate message after fragmentation 1: tail beat
4	Frag ID	[13:10]	It is used to indicate the fragment sequence number of the current fragment message in the original message
5	inject addr	[9:5]	Cache address of St stream while waiting for sending schedule at source
5	submit addr	[4:0]	The st stream caches the address when the terminal is waiting for receiving scheduling

3.12.2 Internal module division

According to the function analysis of FPA module, the internal function division of FPA module is shown in the figure below.

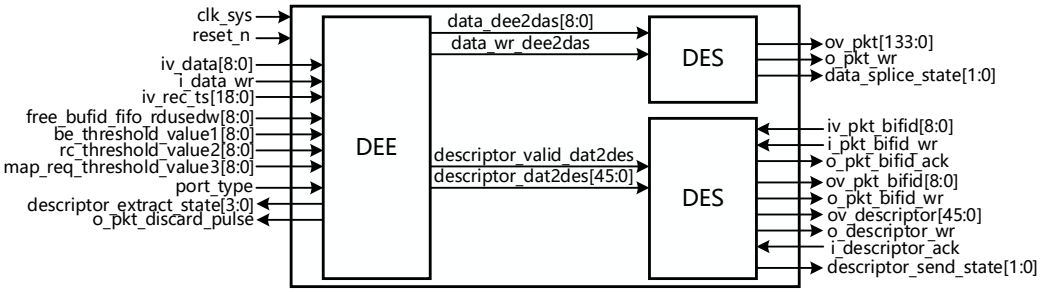

Figure 22 FPA module partition diagram

DEE(descriptor extract) module: forward packet data of message and extract descriptor information at the same time; For PTP message, the received time stamp is filled into TSNtag; According to the current number of remaining bufid and the configured unmapped traffic monitoring threshold, RC traffic monitoring threshold and be traffic monitoring threshold, the traffic monitoring of different packets in different modes is carried out.

DES(descriptor send) module: receive the base address of message cache and respond;Send the base address of message cache to the lower module;Generate and send descriptor information and receive response.

DAS(data cache) module: 9-bit packet data is spliced into 134 bit packet data with prefix and suffix identifier and invalid byte control bit.

3.12.3 Signal definition

The signal definition of FPA module is shown in the table below.

Table 35. FPA module interface signal meaning

Interface signal	Bit width	direction	meaning
External interface signal			
clk_sys	1	input	System clock, 125MHz
reset_n	1	input	System asynchronous reset signal, low effective
NRX-FPA signal definition			
iv_data	9	input	Packet data with head and tail control bits
i_data_wr	1	input	Packet data write signal
iv_rec_ts	19	input	Time stamp of the first hop of received message recorded
NRX-CSM signal definition			
port_type	1	input	Port configuration type.1: Non cooperative type, processing standard Ethernet message;0: cooperation type, processing TSN message.
free_bufid_fifo_rdusedw	9	input	Idle bufid remaining quantity input
map_req_threshold_value3	9	input	If the number of remaining bufid is less than this value, the unmapped flow will be discarded
rc_threshold_value2	9	input	When the number of remaining bufid is less than this value, the RC and be flows will be discarded
be_threshold_value1	9	input	When the number of remaining bufid is less than this value, the be stream will be discarded
o_pkt_discard_pulse	1	output	Message dropping pulse signal is used to count dropped messages
descriptor_extract_state	3	output	descriptor_Extract state machine state register
descriptor_send_state	1	output	descriptor_Send state machine status register
data_splice_state	1	output	data_Splice state machine state register
PCB-FPA signal definition			
i_pkt_bufid_wr	1	input	Message buffer base address write signal
iv_pkt_bufid	9	input	The base address of ram dynamically assigned to the interface message
o_pkt_bufid_ack	1	output	After using the base address of message cache, the response signal sent by the frame resolution module informs the message centralized cache module to

Interface signal	Bit width	direction	meaning
			output the next base address（1 beat effective)
ov_pkt	134	output	Packet data with header and invalid byte control bits
o_pkt_wr	1	output	Packet data write signal
Definition of IBA-FPA signal			
o_pkt_bufid_wr	1	output	Message buffer base address write signal (1 beat valid)
ov_pkt_bufid	9	output	The base address of ram dynamically assigned to the interface message
FLT-FPA signal definition			
ov_descriptor_wr	1	output	Message descriptor write signal
ov_descriptor	46	output	Message descriptor data is used to identify message data
i_descriptor_ack	1	input	After using the descriptor, look up the response signal from the table forwarding module to inform the frame parsing module that it can output the next descriptor（1 beat effective)
Internal interface signal			
data_dee2das	9	DEE2DAS	Packet data with head and tail control bits
data_wr_dee2das	1	DEE2DAS	Packet data write signal
descriptor_valid_dat2des	1	DEE2DES	Descriptor valid signal (1 beat valid)
descriptor_dat2des	46	DEE2DES	Extracted descriptor information

3.12.4 Processing flow

It is realized by three state machines, one is to extract descriptor information (including forwarding packet and modifying TSNtag receiving time stamp of PTP packet), one is to send descriptor (including processing packet centralized cache base address), and one is to splice packet data.

- Extracting descriptor information from state machine

Forwarding packet data by register.

In the first beat, when the packet data is input, the signal i_written_wr is high, i_ data [8] is high, the first packet of frame header is received.

Signal port_type according to port .If it is 1 (non cooperative type), it first determines whether the current number of remaining bufid is less than the unmapped traffic monitoring threshold. If it is 1, it discards the message and jumps to disc_s status, if not, jump to standard Ethernet message processing status; Otherwise (cooperation type) first judge whether the current number of remaining bufid is less than the RC / be traffic monitoring threshold, if less than the RC traffic monitoring threshold, the RC and be flows will be discarded;If it is less than the be traffic monitoring threshold, the be will be discarded;If not, the flow will be obtained from the first part of the group

data_type and flow_id [13:9], record the input iv_rec_ts, jump to the mapped message processing state.

(1) Standard Ethernet message processing status:

Forwarding the packet to the end of the message;

Look up the descriptor directly_ en field is set to 0, the output is set to 9'b100000000 (host interface), and the message type is pkt_Fill in 3'b111 for type, input port number, input parameter value for inport, and 0 for other fields by default.

In the sixth beat, the descriptor information is output.

(2) Mapped message processing status

Second beat, get flow_id[8:1]。

Third beat, from iv_data [7] get flow_id[0]；

4th beat, when flow_ type is 3'b010 (i.e. PTP message), the_Data [2:0] is changed to reg_rec_ts[18:16]；

5th beat, when flow_ type is 3'b010, the_Data [7:0] is changed to reg_rec_ts[15:8]。

6th beat, when flow_type is not 3'b010, from iv_data [4:0] to get submit addr.When flow_type is 3'b010, the ov_data [7:0] is changed to reg_rec_ts[7:0]。lookup_en is 1, output is 0, input port number is inport, input parameter value is filled in, descriptor information is obtained and output according to the above information.

For other beats, only forward the packet to the end.

- Send descriptor state machine

It includes two states.

Initial state: when the message buffer base address writes signal i_pkt_bufid_wr, descriptor valid signal i_descriptor_invalid is all highly effective, then i_descriptor[45:9], iv_pkt_bufid generate ov_descriptor output; At the same time, output the base address (1 beat valid), and give the response o_pkt_bufid_ack (set high, 1 beat valid), indicating that the current cache base address has been used; Jump wait descriptor response state. At the same time, the message discarding pulse signal is set low (1 beat is valid).

When descriptor valid signal i_descriptor_valid is highly effective, while the base address of message cache writes signal i_pkt_bufid_wr is low, the descriptor and base address are not output. At the same time, the message discarding pulse signal is set high (1 beat is valid).

Waiting for descriptor response status: when descriptor acknowledgement signal i_descriptor_ack is highly efficient, o_descriptor_wr set low, o_descriptor is cleared, otherwise, the original output state of the descriptor is kept.

- Splicing packet data state machine

It includes two states.

Initial state: when the message header arrives, ov_pkt_a [127:120] was set as iv_data [7:0], the received byte count value is set to 4'd1.

Continuous transmission status: continuously count the received packet bytes (the current count value is from 0 to 15);

Judge the head and tail state of packet data.When i_data_wr is high and iv_data [8] is 0, it means the middle group, and it is filled in the corresponding position of 128 bit splicing register according to the byte count value; When i_data_wr is high and iv_data [8] is 1, it is the end group, and the filling position and the number of invalid bytes in the 128 bit splicing register are obtained according to the byte count value. In the same way, we can get the status information of the head and tail identification.

3.13 Design of input buffer interface (IBI) module

3.13.1 functional analysis

Write the message to the message central buffer. Receive 134 bit packet data, and write the packet data into the message centralized buffer according to the time division multiplexing mechanism. When the response signal is received, the next packet can be sent.

3.13.2 Internal function division

According to the function analysis of IBI module, there is no need to further divide the FFR module.

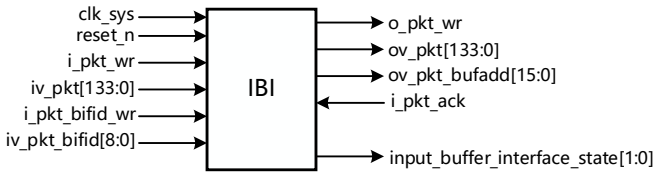

Figure 23 division diagram of IBI module

3.13.3 Signal definition

The signal definition of input buffer interface (IBI) module is shown in the table below.

Table 36. Meaning of IBI module interface signal

Interface signal	Bit width	direction	meaning
External interface signal			
clk_sys	1	input	System clock, 125MHz
reset_n	1	input	System asynchronous reset signal, low effective
Definition of FPA-IBI signal			
iv_pkt	134	input	Packet data with header and invalid byte control bits

i_pkt_wr	1	input	Packet data write signal
i_pkt_bufid_wr	1	input	Message buffer base address write signal
iv_pkt_bufid	9	input	The base address of ram dynamically assigned to the interface message
PCB-IBI signal definition			
ov_pkt	134	output	Packet data with header and invalid byte control bits
o_pkt_wr	1	output	Packet data write signal
ov_pkt_bufadd	12	output	The global address of ram corresponding to packet data.134 * 8bit storage space is allocated for each message (frame).
i_pkt_ack	1	input	After the packet data is written, the response signal given by the message centralized caching module informs the interface module that the next packet data can be output
CSM-PCB signal definition			
input_buf_interface_state	2	output	input_buffer_Interface state machine state register

3.13.4. Processing flow

When i_pkt_bufid_wr is highly efficient, the packet data has not yet arrived, so I need to_pkt_bufid_wr ,iv_pkt_bufid records it and gets reg_pkt_bufid_valid,reg_pkt_bufid。

In order to avoid that the output of the previous packet has not been released when the input packet data arrives, it is necessary to cache the input packet data while waiting for the write completion response and record it as reg_pkt_wr,reg_pkt。

It is realized by a state machine, including the following three states.

Initial state: judge whether the discard signal is high. If it is high, it means that the FPM module discards the descriptor, so the message data should also be discarded; Otherwise, when i_pkt_wr is highly effective, iv_pkt is the header and reg_pkt_bufid_valid is high, the input packet data I is written to the ram_pkt, the global address is [reg_pkt_bufid,3’b0];Jump wait write complete response state.

If reg_pkt_bufid_ value is low, the message will be discarded even if the message header arrives, and the initial state will be maintained all the time.

Waiting for response state: hold ov_pkt, ov_pkt_wr, ov_pkt_bufadd output such as is valid until received_pkt_ack reset release; When it is the end of the message, it will return to the initial state, otherwise it will jump to the continuous sending state.

When waiting for the response, it takes 10 beats to receive the response and the 11th beat to complete the release; When the input packet is at the end of the message and there are few effective bytes, the time gap between the packet and the previous packet is no longer 16 beats, that is, the output of the previous packet has not been released, and the end of the message has arrived, so it needs to be cached until it returns to the initial state.

Continuous sending status: when i_pkt_wr is highly effective, input packet data I is written to the ram_pkt, global address plus 1; Jump wait write complete response state.

3.14 Design of NIQ module

3.14.1 functional analysis

The main functions of niq module are as follows:

(1) According to the received pkt_type and gate_ctrl_vector selects the queue;

(2) Pkt to be received_bufid is written into the corresponding queue of the network queue management module;

(3) Set pkt_ information that bufid writes to the queue informs the network output scheduling module so that the network output scheduling module can record the queue status;

(4) Count the number of message descriptors in all queues.

3.14.2 Internal module division

According to the function analysis of the NIQ module, it can be realized without further module division.

3.14.3 interface design

The interface signal of NIQ module is shown in the figure below.

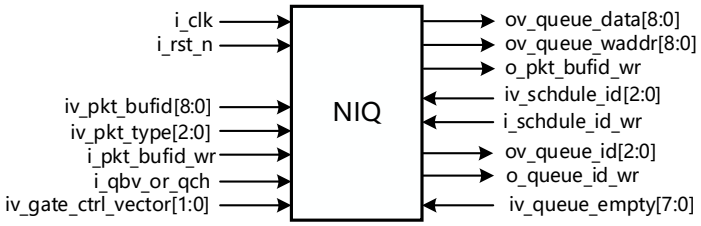

Figure 24 interface signal of niq module

The specific meanings of all the signals in the interface signal diagram of the niq module are shown in the table below.

Table 37. Signal meaning of NIQ module interface

Interface signal	Bit width	direction	meaning
i_clk	1	input	125MHz clock
i_rst_n	1	input	Reset, low effective
Definition of NIQ-FLT signal			
iv_pkt_bufid	9	input	Message cache ID data input
iv_pkt_type	3	input	Message type data input
i_pkt_bufid_wr	1	input	Message cache ID data input valid signal input

Interface signal	Bit width	direction	meaning
NIQ-QGC signal definition			
iv_gate_ctrl_vector	2	input	Queue gating information input
Definition of NIQ-CSM signal			
i_qbv_or_qch	1	input	Qbv / qch mode selection signal input
Definition of NIQ-QBA signal			
ov_queue_wdata	9	output	Queue cache write data output (null or netx)_bufid, refer to the processing flow for details)
ov_queue_waddr	9	output	Queue cache write address output
o_queue_wr	1	output	Queue cache write signal output
Definition of NIQ-NOS signal			
ov_queue_id	3	output	pkt_output of queue information written by bufid
ov_queue_id_wr	1	output	pkt_output of queue information valid signal written by bufid
iv_queue_empty	8	input	8 Queue empty status signal input
iv_schdule_id	3	input	Scheduling queue information input
i_schdule_id_wr	1	input	Scheduling queue information valid signal input

3.14.4 Processing flow

After receiving pkt, this module_bufid and pkt_type data, according to the gate input by the queue gating module_ctrl_vector signal and pkt_ type information is pkt_bufid selects the queue;And the pkt_bufid and queue address data are sent to the network queue management module for pkt_bufid cache.When the input pkt_type represents an ST message, which needs to determine whether the current mode is qbv or qch; If the current mode is qbv, directly according to pkt_type the queue that enters the queue;If the current mode is qch, it is necessary to use gate_ctrl_vector signal judges that the st message enters the corresponding queue.

This module also needs to maintain the tail address of the queue list. After the initialization of the module, the first pkt is entered_bufid is the tail address, and the write address to the network queue management module is pkt_bufid, the write data is "null", which is currently identified by 0.The next pkt_bufid, you need to judge the queue_Empty signal, if it is high, it means that the queue has been read empty, and this pkt is needed_bufid is the tail address, and the write address to the network queue management module is pkt_bufid, write data as null;If it is low, it means that there is still data in the queue, and this pkt is needed_bufid as the tail address, the address written to the network queue management module is the last tail address, and the data written is the current received pkt_bufid。

This module counts and manages the number of each queue, and adds 1 to the corresponding queue counter for each bufid written to the queue; After receiving a

scheduling information from the network scheduling module, according to the scheduling queue, the corresponding queue counter is subtracted by 1.

Every pkt processed pkt_bufid needs to transfer this pkt_bufid input queue_ id is sent to the output scheduling module to manage the data in each queue.

3.15 Design of QGC module

3.15.1 Functional analysis

The main functions of QGC module are as follows:

(1) Ram is used to cache the gating list;

(2) According to the current qbv / qch mode, judge the current input gating state;

(3) According to the global time information, look up the output gating list to get the current input and output gating information.

3.15.2 Internal module division

According to the function analysis of QGC module, a RAM (8bit_1024) can be instantiated cache the gating list, and then divide it into two modules to realize the parsing of control commands and the state control of time gating. The specific design is as follows:

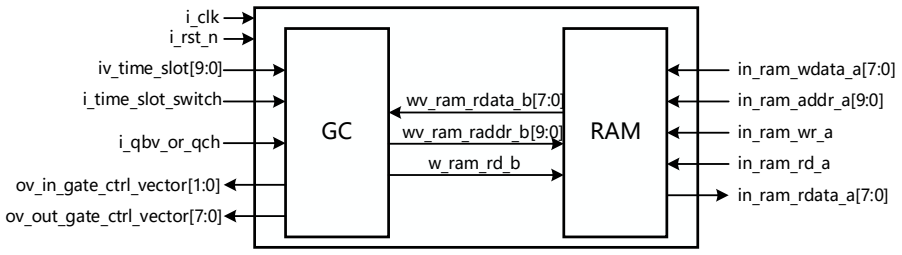

Fig. 25 division diagram of QGC module

GC (gate control) gating module: the main function is to get a time slot information according to the global clock and read out the gating information from the corresponding address in the gating list.

3.15.3 interface design

The specific meanings of all signals in the interface signal diagram of the queue gating (QGC) module are shown in the table below.

Table 38. QGC module interface signal meaning

Interface signal	Bit width	direction	meaning
i_clk	1	input	125MHz clock
i_rst_n	1	input	Reset, low effective

Interface signal	Bit width	direction	meaning
Qgc-niq & NOS signal definition			
ov_in_gate_ctrl_vector	2	output	Queue input gating information output
ov_out_gate_ctrl_vector	8	output	Queue output gating information output
Qgc-hrp signal definition			
iv_time_slot	10	input	Global slot information input
i_time_slot_switch	1	input	Global time slot switching signal input
Qgc-csm signal definition			
i_qbv_or_qch	1	input	Qbv / qch mode selection signal input
iv_ram_wdata_a	8	input	Output gated ram write data
iv_ram_addr_a	10	input	Read write address of output gated RAM
i_ram_wr_a	1	input	Output gated ram write signal
iv_ram_rdata_a	8	input	Read data from output gated RAM
i_ram_rd_a	1	input	Output gated RAM read signal
Signal definition between modules			
wv_ram_rdata_b	8	RAM2GC	Read data from output gated RAM
wv_ram_raddr_b	10	GC2RAM	Read address of output gated RAM
wv_ram_rd_b	1	GC2RAM	Output gated RAM read signal

3.15.4 Processing flow

Judge the current mode according to the input qbv / qch mode selection signal. If it is qbv mode, all the input gates will be opened; If it is in qch mode, it is necessary to judge whether the current time slot is odd or even according to the input global time slot information, and select to open the corresponding queue input gating.

The module searches the output gating list according to the current time slot, and takes the read data as the output gating.

3.16 Design of network queue management (nqm) module

3.16.1 functional analysis

NQM module main function: network output port queue centralized cache management.

3.16.2 Internal module division

According to the function analysis of nqm module, it can be realized without further module division.

3.16.3 interface design

The interface signal of network queue management (nqm) module is shown in the figure below.

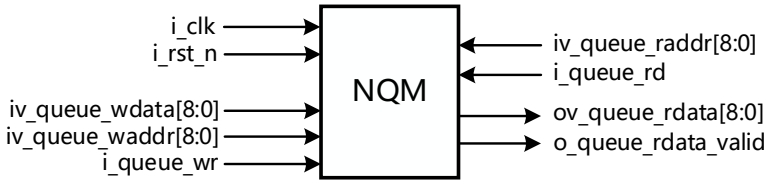

Figure 26 nqm module interface signal

The specific meanings of all signals in the interface signal diagram of network queue management (nqm) module are shown in the table below.

Table 39. Nqm module interface signal meaning

Interface signal	Bitwidth	direction	meaning
i_clk	1	input	125MHz clock
i_rst_n	1	input	Reset, low effective
Nqm-niq signal definition			
iv_queue_wdata	9	input	Queue write data input
iv_queue_waddr	9	input	Queue write address input
i_queue_wr	1	input	Queue write signal input
Nqm-nos signal definition			
iv_queue_raddr	9	input	Queue read address input
i_queue_rd	1	input	Queue read signal input
ov_queue_rdata	9	output	Queue read data output
ov_queue_rdata_valid	1	output	Queue read data valid signal output

3.16.4 Processing flow

A pseudo dual port RAM is instantiated in the nqm module to cache the queue.The content of the queue is pkt_bufid, this module needs to support 512 pkt_refore, the ram is designed as 9bits * 512.

The input write related signal is directly connected to the write interface of ram for queue write operation.

When reading the queue, because the read operation of ram needs to delay two beats to read out the data, this module needs to wait two beats after sending the read signal to ram, and then send the data from RAM to the lower module.

Since there is a delay of two beats when reading ram, special processing is needed if a signal to write the same address is received when reading ram. If the above situation occurs, the written data should be directly regarded as the data read out from RAM and transmitted to the network output scheduling module.

3.17 Design of network output scheduling (NOS) module

3.17.1 functional analysis

The main functions of NOS module are as follows:

(1) Manage the status of the queue;

(2) According to the queue state and gating information, the priority scheduling queue is calculated;

(3) The queue information to be scheduled is sent to the network queue management module, and the cached data is read out.

3.17.2 Internal module division

According to the function analysis of NOS module, it can be divided into two modules to realize virtual queue state management and output scheduling algorithm.The specific design is as follows:

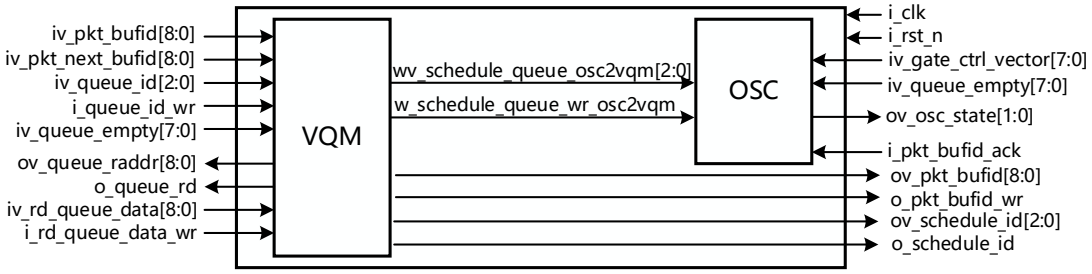

Figure 27 NOS module partition diagram

VQM (virtual queue management) virtual queue management module: the main function is virtual management of the queue cached in the network queue management module;According to the output scheduling module to get the priority scheduling queue, get the address to be read from the queue, and read the corresponding queue data from the network queue management module.The main function of virtual management is to maintain the first address of the queue and read the first data correctly when the queue needs to be scheduled.The first address of the first priority scheduling is sent to the network queue management module to read the queue data, and sent to the network sending module.

Output scheduling module of OSC: the main function is to calculate the priority queue information in the current time. Through the current time gating information, the priority scheduling queue is calculated, and the queue information is sent to the virtual queue management module. Receiving network sending module receives pkt_bufid confirms the signal, the next round of scheduling can be carried out.

3.17.3 interface design

The specific meanings of all signals in the interface signal diagram of the network output scheduling (NOS) module are shown in the table below.

Table 40. Signal meaning of NOS module interface

Interface signal	Bit width	direction	meaning
i_clk	1	input	125MHz clock
i_rst_n	1	input	Reset, low effective
Definition of nos-niq signal			
iv_pkt_bufid	9	input	Message cache ID data input
iv_pkt_next_bufid	9	input	Next message cache ID data input
iv_queue_id	2	input	pkt_Queue information input written by bufid
i_queue_id_wr	1	input	pkt_Valid signal input of queue information written by bufid
iv_queue_empty	8	input	8Queue empty status signal input
ov_schedule_id	3	output	Scheduled queue signal output
o_schedule_id_wr	1	output	Output of queue valid signal for scheduling
Nos-qgc signal definition			
iv_gate_ctrl_vector	8	input	Queue gating information input
Definition of nos-qba signal			
ov_queue_raddr	9	output	Queue read address output
o_queue_rd	1	output	Queue read signal output
iv_rd_queue_data	9	input	Data input for queue readout
i_pkt_queue_data_wr	1	input	Queue read data valid signal input
Nos-ntx signal definition			
ov_pkt_bufid	9	output	Pkt of queue read_bufid data output
o_pkt_bufid_wr	1	output	Pkt of queue read_bufid data valid signal output
i_pkt_bufid_ack	1	input	pkt_bufid data receiving confirmation signal input
Nos-csm signal definition			
o_osc_state	2	output	OSC module state machine output
Signal definition between modules			
wv_schdule_queue_osc2vqm	2	VQM2OSC	Queue information of current priority scheduling
w_schdule_queue_wr_osc2vqm	1	VQM2OSC	Queue information valid signal of current priority scheduling

3.17.4 Processing flow

The module maintains the address of the first data cache of each queue in the network queue management module according to the queue write signal given by the network queue control module and the queue read signal given to the queue cache module after the module scheduling. The first address is the address when the network queue control module writes the data to the queue during initialization and after the queue is read empty. When there is data in the queue, the first address is modified to read data.

The module calculates according to the maintained queue state information and the input gating information, gets the queue information of the highest priority scheduling, and sends the first address of the queue to the network queue management module, reads the queue cache data from the module, and then converts the first address to pkt_bufid is sent to the network sending module. After completing the algorithm, we need to wait for the received pkt returned by the network sending module_bufid confirmation signal, the output scheduling module can send the latest priority queue information to the virtual queue management module.

When the module transmits the scheduled queue information to the network queuing control module for queue counting, there will be two beats of delay. If a bufid is input to the queue within the two beats of delay time, and the queue empty signal is not updated in time due to the delay, the module will take the last read data (null) as the next first address. After two beat delay, due to the input of a bufid, null will be regarded as bufid in the next scheduling, resulting in an error. In order to avoid this phenomenon, the module judges whether the address written by the network queuing module to the network queue management module is consistent with the address read by the module after scheduling. If not, it will process normally. If it is consistent, the data written by the network queuing module to the network queue management module should be regarded as the next first address of the queue.

3.18 Design of NTX module

3.18.1 functional analysis

Main functions of NTX module:

(1) Receiving pkt_bufid and converting it to address and the corresponding message data is read out from the message buffer;

(2) The message is distinguished, and the transparent clock of time synchronization message is updated;

(3) The 128bit message data is converted into 8bit message data, and the internal clock domain is switched to the external PHY clock domain, and then it is continuously transmitted to the output architecture from gmii interface.

3.18.2 Internal module division

According to the function analysis of NTX module, it can be divided into two modules to realize pkt_ function of converting bufid to address and reading message from message centralized cache module, transparent clock update of time synchronization message and transmission from gmii interface after removing metadata. The specific design is as follows:

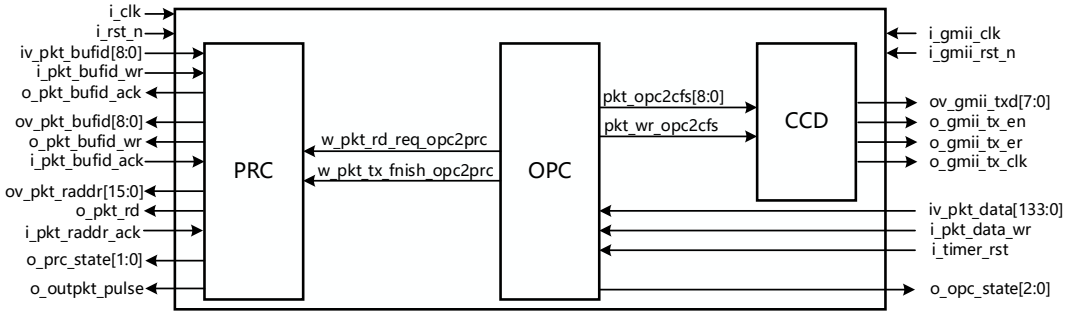

Figure 28 NTX module partition diagram

PRC (pkt read control) message reading control module: the main function is to transmit the received pkt_bufid is converted to address and the message data is read out from the message buffer. Pkt_bufid converts once and then returns to the output scheduling module to receive an acknowledgement signal; After each read signal is sent, it needs to wait for the message read request signal given by the output control module to be valid and the address receiving confirmation signal returned by the message centralized cache module to send the next read signal. A new round of pkt can be carried out only after receiving the completion signal of the message sent by the output control module_bufid conversion and subsequent read operation. Each time a message is read out and transmitted, a pulse signal is sent to the configuration and status management module for counting.

OPC (output control) output control module: The main function is to identify the message sent by the message centralization module and update the transparent clock of the synchronous message; Two registers are used to cache the message. When any of the registers is empty, it applies to the message reading control module to read the message once until all the messages are sent, and notifies the message reading control module; When the message is sent, it needs to be converted into 9bit data (including 1bit head and tail identification, 8bit message data) and transmitted to the clock domain switching module. When calculating the transparent clock, we need to maintain a counter in the module first, and get the transparent clock by subtracting the receive time stamp in metadata from the value of the counter; The counter is the time_rst given by the time synchronization module signal for the whole architecture synchronization.

CCD (cross clock domain) clock domain switching module: The main function is to switch 9bit data from internal processing clock domain to external PHY clock domain, and finally send it to the network from gmii interface.

3.18.3 interface design

The specific meanings of all signals in the interface signal diagram of NTX module are shown in the table below.

Table 41. NTX module interface signal meaning

Interface signal	Bit width	direction	meaning
i_clk	1	input	125MHz clock
i_rst_n	1	input	Reset, low effective
i_gmii_clk	1	input	Clock of gmii interface
i_gmii_rst_n	1	input	Reset, low effective
Ntx-nos signal definition			
iv_pkt_bufid	9	input	pkt_bufid data input
i_pkt_bufid_wr	1	input	pkt_bufid data valid signal input
o_pkt_bufid_ack	1	output	pkt_bufid data receiving confirmation signal output
Ntx-pcb signal definition			
ov_pkt_raddr	16	output	Read address output to message centralized cache
o_pkt_rd	1	output	Read signal output to message centralized cache
i_pkt_raddr_ack	1	input	Address receiving confirmation signal input of message centralized cache
ov_pkt_bufid	9	output	Pkt to cache in message set_bufid output
o_pkt_bufid_wr	1	output	Pkt to cache in message set_bufid valid signal output
i_pkt_bufid_ack	1	input	pkt_bufid data transmission confirmation signal input
iv_pkt_data	134	input	Data input of message centralized cache
i_pkt_data_wr	1	input	Message data input valid signal in message centralized cache
Ntx-gts signal definition			
i_timer_rst	1	input	Architecture internal time counter reset signal input
Nos-csm signal definition			
o_opc_state	3	output	OPC module state machine output
o_prc_state	2	output	PRC module state machine output
o_outpkt_pulse	1	output	Sending message pulse signal
o_fifo_overflow_pulse	1	output	FIFO overflow pulse signal
output			
ov_gmii_txd	8	output	Gmii data output
o_gmii_tx_en	1	output	Gmii data valid signal output
o_gmii_tx_er	1	output	Gmii data error signal output
o_gmii_tx_clk	1	output	Gmii data clock signal output

Interface signal	Bit width	direction	meaning
Signal definition between modules			
w_pkt_rd_req	1	OPC2PRC	Message read request signal
w_pkt_tx_finish	1	OPC2PRC	Message sending completion signal
wv_pkt_data_opc2cfs	9	OPC2CFS	Message data
w_pkt_data_wr_opc2cfs	1	OPC2CFS	Message data valid

3.18.4 Processing flow

The message reading control module will read the message according to the received pkt_bufid is converted into address, sent to the message centralized cache module to read the message data, and returned to the network output scheduling module with a pkt_bufid receives the acknowledgement signal. After sending a read signal to the message centralized cache module, it is necessary to judge whether the output control module gives a message sending completion signal. If not, wait until the address receiving confirmation signal returned by the message centralized cache module is high and the message reading request signal of the output control module is high before the next read operation; If the message is sent, the conversion of next pkt_bufid is performed, or in idle state.

The output control module needs to identify the received message data to update the transparent clock of the time synchronization message. The transparent clock can be calculated through the receiving time stamp in TSNtag and the sending time stamp of this module, and then added to the transparent clock field of the synchronization message. After the transparent clock is calculated, the subsequent message data is cached by register. Two registers are maintained in the module, and ping-pong method is used to cache message data; Each time, 8-bit data is taken out from the register and combined into 9-bit data, which is sent to the clock domain switching module until one register is empty, a message read request signal is returned, and the data in the other register is read and transmitted. When reading out the transparent field of synchronization message from the register, it is necessary to accumulate the previously calculated transparent clock into the field. When a complete message transmission is completed, send a message completion signal to the message reading control module to read and transmit the next message, or in idle state.

After receiving the data, the clock domain switching module switches the data from the internal processing clock domain to the external PHY clock domain, and then sends it to the network by gmii interface

3.19 Design of forwarding look up table (FLT) module

3.19.1 functional analysis

Main functions of FLT module:

(1) Use RAM (9bit)_16384);

(2) Receiving the descriptor of each port by time division multiplexing;

(3) Receiving the descriptor and judging whether the descriptor needs to look up the table;

(4) According to the flow_id in the message descriptor to look up the forwarding table, and look up the table to get the output port;

(5) According to the output port, the pkt_bufid of the message is changed and message type are sent to the corresponding port.

3.19.2 Internal module division

According to the function analysis of FLT module, a ram can be instantiated to store the republication, and then divided into four modules to realize the republication of writing, reading, keyword extraction, table lookup and forwarding. The specific design is as follows:

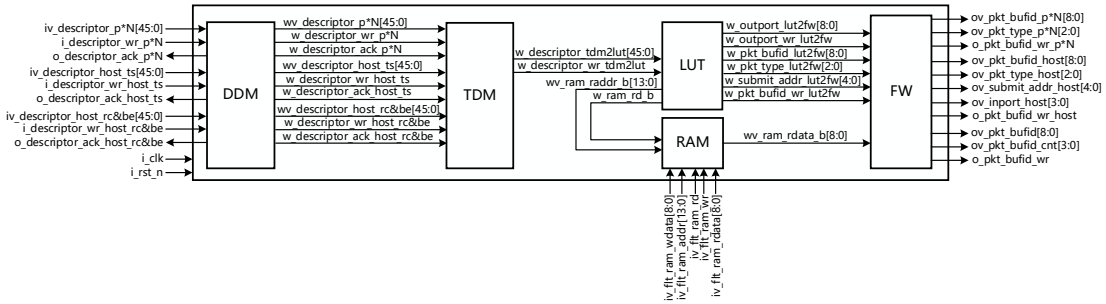

Figure 29 FLT module partition diagram

DDM (descriptor delay module) descriptor delay module: the main function is to delay the descriptor sending by several beats. When the host or network inputs a message, the first 14b of the received message can construct a descriptor to send it. After the FLT module looks up the table and forwards it, the message is scheduled at the output port and read from the buffer; At this time, the input processing module has not written the message to the buffer, so there will be a message reading error or read all zeros.In order to avoid this phenomenon, in this module, we need to delay the sending of descriptors. The specific delay number needs to be determined after the actual situation test (currently 9 beats).

TDM (time division multiplexing) module: The main function is to receive descriptor data from different interfaces;If the descriptor data is received in the bandwidth corresponding to the interface, the descriptor is sent to the table lookup module, and a descriptor is sent to the corresponding superior module to receive the confirmation signal.

LUT (look up table) look up module: the main function is to determine whether the descriptor needs to look up the table. If you need to look up the table, the flow_id in the interface descriptor will be extracted, and the ram is read (look-up table processing), and the pkt_bufid is saved, submit_addr and pkt_ type data is sent to the

forwarding module two beats later; If there is no need to look up the table, the output port in the descriptor is extracted as the output port number, and the pkt_bufid, s ubmit_addr and pkt_ type data is sent to the forwarding module together with the output port number.

FW (forward) forwarding module: the main function is to set pkt according to the output port number_bufid, submit_addr and pkt_type to the corresponding output port logic.

3.19.3 interface design

The specific meanings of all signals in the interface signal diagram of forwarding look up table (FLT) module are shown in the table below

Table 42. FLT module interface signal meaning

Interface signal	Bit width	direction	meaning
i_clk	1	input	125MHz clock
i_rst_n	1	input	Reset, low effective
Flt-ppa signal definition			
iv_descriptor_p*N	45	input	Message descriptor input of network interface (n = 0-7)
i_descriptor_wr_p*N	1	input	Message descriptor valid signal input of network interface (n = 0-7)
o_descriptor_ack_p*N	1	output	Output of received message descriptor confirmation signal to network interface (n = 0-7)
Flt-mlc signal definition			
iv_descriptor_host_ts	45	input	St message descriptor input of host interface
i_descriptor_wr_host_ts	1	input	St message descriptor valid signal input of host interface
o_descriptor_ack_host_ts	1	output	Receive st message descriptor acknowledgement signal output to host interface
iv_descriptor_host_rc&be	45	input	The input of RC and be message descriptors in host interface
i_descriptor_wr_host_rc&be	1	input	Effective signal input of RC and be message descriptors of host interface
o_descriptor_ack_host_rc&be	1	output	Output of receive RC and be message descriptor acknowledgement signal to host interface
Flt-niq signal definition			
ov_pkt_bufid_p*N	9	output	Pkt to network output interface_bufid output (n = 0-7)
ov_pkt_type_p*N	3	output	Message type output to network output interface (n = 0-7)
o_pkt_bufid_wr_p*N	1	output	Pkt to network output interface_bufid effective signal output (n = 0-7)
Flt-hiq signal definition			

Interface signal	Bit width	direction	meaning
ov_pkt_bufid_host	9	output	Message output to host output interface
ov_pkt_type_host	3	output	Message type output to host output interface
ov_submit_addr_host	5	output	The st message to the host output interface submits the cache address output
ov_inport_host	4	output	Message input port output to host output interface
o_pkt_bufid_wr_host	1	output	Pkt to host output interface_bufid valid signal output
ov_pkt_bufid	9	output	output signal
o_pkt_bufid_wr	1	output	Output enable signal
ov_pkt_bufid_cnt	4	output	Counter
ov_tdm_state	4	output	State machine output
Flt-csm signal definition			
iv_flt_ram_wdata	9	IQC2RAM	Ram write data
iv_flt_ram_addr	14	IQC2RAM	RAM read write address
iv_flt_ram_wr	1	IQC2RAM	Ram write signal
iv_flt_ram_rdata	9	RAM2OQC	RAM read data
iv_flt_ram_rd	1	OQC2RAM	RAM read signal

3.19.4 Processing flow

The TDM module will receive 10 groups of interface input message descriptors at the same time. Since multiple groups of signals will be valid at the same time for 10 groups of signals, it is necessary to allocate bandwidth for 10 groups of interfaces. In the allocated bandwidth, judge whether the corresponding interface has message descriptor input. If so, send the input descriptor to the table lookup module and return a receive confirmation signal to the corresponding interface logic. If not, nothing will be done.

The lookup module first uses the lookup in the descriptor_ en signal is used to determine whether it is necessary to look up the table for forwarding. If necessary, extract the flow_id from the descriptor, this flow_id is sent to ram as read address for read operation, and pkt is sent to ram_bufid, pkt_type and submit_addr data is sent to the forwarding module two beats later. If there is no need to look up the table, the output information is extracted from the descriptor and compared with pkt_bufid, pkt_type, submit_addr data is sent directly to the forwarding module.

Every pkt_bufid received by forwarding module, pkt_type and submit_addr data, first judge whether the output information sent by the table lookup module is valid, if it is valid, take the output as the output port, and use pkt_bufid, pkt_type and submit_addr data is sent to the corresponding output port logic;If it is invalid, read data is extracted from RAM as output port, and use pkt _bufid, pkt_type and submit_Addr data is sent to the corresponding output port logic.

3.20 Design of message centralized cache (PCB) module

3.20.1 functional analysis

The 134 bit packet data of 9-way Gigabit ports are written into RAM by time division multiplexing, that is, 134 bit packet data can be written to each port every 9 beats.

Through time division multiplexing, the centralized cache data is divided into 9 channels of 134 bit packet data of Gigabit ports, that is, each port can read 134 bit packet data every 9 beats.

By time division multiplexing, the idle packet cache base address is allocated to 9 Gigabit ports, that is, each port can get a packet cache base address every 9 beats.

By time division multiplexing, the base address of free message cache released by 9 Gigabit ports is written into FIFO free address cache, that is, each port can write one free message cache base address every 9 beats.

The pseudo dual port RAM with 134bit width and 65536 depth (address range [15:0]) is used as the centralized message cache. Each packet is allocated 128 units (134 bytes in total) for storage, while the ram can store 512 packets in total. In other words, the maximum length of each message is 134 bytes (including the first and last control bits and invalid bytes), each message corresponds to a base address (address range [15:8]), and each packet data corresponds to a global address (address range [16:0]).

The synchronous FIFO based on RAM with a width of 9 bits and a depth of 512 is used as the buffer of the base address of the idle message cache.

512 registers with a width of 4 bits are used to manage the number of base address usage of multicast message.

3.20.2 Internal function division

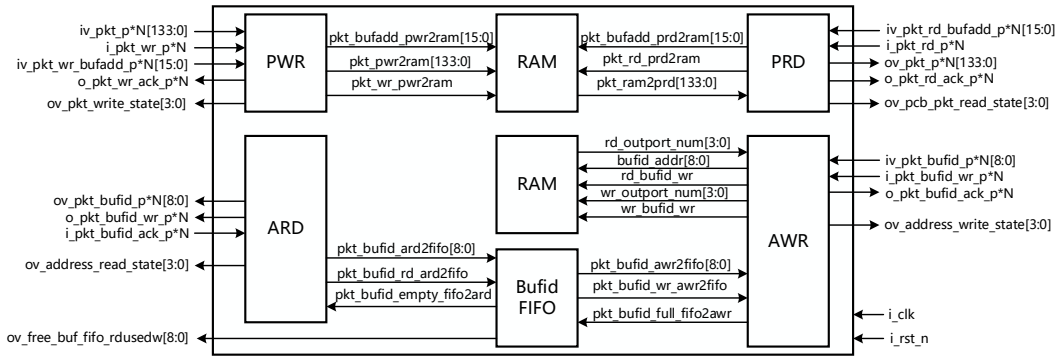

Figure 30 PCB module partition diagram

PWR(pkt write, write packet data) module: write 9-way Gigabit port 134 bit packet data to ram through TDM.

PRD(pkt read, read packet data) module: divide the centralized cache data into 9-way Gigabit port 134 bit packet data through time division multiplexing.

ARD(address read, read base address) module: through time division multiplexing, the idle message cache base address is allocated to 9 Gigabit ports.

AWR(address write) module: write the base address of message cache released by 9-way Gigabit port through time division multiplexing, and decide whether to write the base address of message cache released by port into the free address cache according to the number of times of using the base address of group broadcast.

3.20.3 Signal definition

Table 43. PCB module interface signal meaning

Interface signal	Bit width	direction	meaning
External interface signal			
clk_sys	1	input	System clock, 125MHz
reset_n	1	input	System asynchronous reset signal, low effective
Network / host input - PCB signal definition			
iv_pkt_p*N	134	input	Packet data received by network interface n (n = 0-7)
i_pkt_wr_p*N	1	input	Network interface n (n = 0-7) receives packet data write signal
iv_pkt_wr_bufadd_p*N	16	input	The global address of ram corresponding to packet data received by network interface n (n = 0-7)
o_pkt_wr_ack_p*N	1	output	After the packet data is successfully written, the response signal to network interface n (n = 0-7) is sent
ov_pkt_bufid_p*N	9	output	The base address of ram dynamically assigned to network interface n (n = 0-7) to receive messages
o_pkt_bufid_wr_p*N	1	output	Network interface n (n = 0-7) receives buffer base address write signal of message
i_pkt_bufid_ack_p*N	1	input	The response signal given by network interface n (n = 0-7) after using the buffer base address of received message
Network / host output - PCB signal definition			
iv_pkt_rd_bufadd_p*N	16	input	The global address of ram corresponding to the message sent by network interface n (n = 0-7)
i_pkt_rd p*N	1	input	Read signal of message sent by network interface n (n = 0-7)
o_pkt_rd_ack_p*N	1	output	After the message is read successfully, the response signal is sent to the network interface n (n = 0-7)
ov_pkt_p*N	134	output	Packet data of sending message to network interface n (n = 0-7)
o_pkt_wr_p*N	1	output	Send message packet data write signal to network interface n (n = 0-7)
iv_pkt_bufid_p*N	9	input	The base address of centralized cache (RAM) released by network interface n (n = 0-7)

i_pkt_bufid_wr_p*N	1	input	buffer base address write signal released by network interface n (n = 0-7)
o_pkt_bufid_ack_p*N	1	output	After the released cache base address is successfully written, a response signal is sent to the network interface n (n = 0-7)
Flt-pcb signal definition			
i_pkt_bufid_wr_flt	1	input	Look up table forwarding module to the group broadcast base address write signal
iv_pkt_bufid_flt	9	input	Look up table forwarding module to the group broadcast base address
iv_pkt_bufid_cnt_flt	4	input	The number of times the group broadcast base address is used by the table lookup forwarding module
Csm-pcb signal definition			
ov_pkt_write_state	4	output	pkt_Write state machine state register
ov_pcb_pkt_read_state	4	output	pkt_Read state machine status register
ov_address_write_state	4	output	address_Write state machine state register
ov_address_read_state	4	output	address_Read state machine status register
ov_free_bufid_fifo_rdusedw	9	output	Free base address cache FIFO usage depth, that is, the number of free base addresses
Internal interface signal			
pkt_pwr2ram	134	PWR2RAM	Group data written to ram by write data state machine
pkt_wr_pwr2ram	1	PWR2RAM	Packet data write signal written to ram by write data state machine
pkt_bufadd_pwr2ram	12	PWR2RAM	Global address of packet data written to ram by write data state machine
pkt_bufadd_prd2ram	12	PRD2RAM	Read data state machine reads the global address of ram packet data
pkt_rd_prd2ram	1	PRD2RAM	The read data state machine reads the read signals of ram packet data
pkt_ram2prd	134	RAM2PRD	Read data state machine reads ram packet data
pkt_bufid_rd_ard2fifo	1	ARD2FIFO	Read base address state machine reads free base address read signal of FIFO
pkt_bufid_fifo2ard	9	FIFO2ARD	Read base address state machine reads free base address of FIFO
pkt_bufid_empty_fifo2ard	1	FIFO2ARD	Null signal of base address FIFO
pkt_bufid_wr_awr2fifo	1	AWR2FIFO	Free base address write signal written to FIFO by write base address state machine
pkt_bufid_awr2fifo	9	AWR2FIFO	Write base address the free base address that the state machine writes to FIFO
pkt_bufid_full_awr2fifo	1	AWR2FIFO	Full signal of base address FIFO
rd_outport_num	4	RAM2AWR	Number of multicast RAM read data
bufid_addr	9	AWR2RAM	Number of multicast RAM read write address
rd_bufid_wr	1	AWR2RAM	Number of multicast RAM read signal
wr_outport_num	4	AWR2RAM	Number of multicast ram write data
wr_bufid_wr	1	AWR2RAM	Number of multicast ram write signal

3.20.4 Processing flow

Through 4 modules and 2 RAM (65536_134bit and 512 bit_4 bit), 1 FIFO (512_9bit).

Write data and read data operations are independent of each other.It is necessary that the read data period of each port is equal to the write data period, so as to ensure that the centralized cache (RAM) will not overflow or read empty.Because the designed write data cycle is 9 beats and read data cycle is 9 beats, and considering the actual port speed is 1Gbps, in fact, for 128 bit packet data, the read and write data cycle is 16 beats, so the write data cycle is equal to the read data cycle, and the centralized cache will not overflow or read empty.

Pkt write data module: set 9 states (wr_pkt_ch0_s~wr_pkt_ch8_s) , one for each port. In these nine states, each beat jumps to one state in turn, so repeatedly.

For example, in wr_pkt_ch0_s state:

(1) Will o_pkt_wr_ack_p1~ o_pkt_wr_ack_p8 is set low, it means that the packet data of port 2 ~ 9 is not written at the moment;

(2) Determine the port packet data write signal i_pkt_wr_p0 is highly effective when i_pkt_wr_p0 is highly effective, the global address is directly input_pkt_wr_bufaddr_0, grouping data iv_pkt_write_p0 to ram and write o_pkt_wr_ack_p0 is set high (1 beat is valid), it indicates that the packet data of port 1 is written successfully, and informs port 1 that the next packet can be output;

(3) Jump wr_pkt_ch1_s status.

Pkt read data module: set 9 states (rd_pkt_ch0_s~rd_pkt_ch8_s) , one for each port. In these nine states, each beat jumps to one state in turn, so repeatedly.

For example, in rd_pkt_ch0_in s state:

(1) Will o_pkt_rd_ack_p1~ o_pkt_rd_ack_p8 is set low, it means that the packet data of port 2 ~ 9 is not written at the moment;

(2) Judge the port packet data read signal i_pkt_rd_p0 is highly effective when i_pkt_wr_p0 is highly effective, the global address is directly input iv_pkt_rd_bufadd_p0 packet data from RAM, and o_pkt_rd_ack_p0 is set high (1 beat is valid), it indicates that the packet data of port 1 is read successfully, informing port 1 that the next packet can be read;

(3) Set read port 1 packet data identification bit rd_valid [0], so as to output the read packet data to port 1 after waiting for 2 beats.

(4) Judge that the port 7 packet data identification bit rd_valid[6] is high when rd_valid [6] is highly effective, the RAM read packet data is output to port 7.

(5) Jump to rd_pkt_ch1_s status.

Address read base address module: The initialization state is to output one free address (0 ~ 9) to each of the nine ports, such as network interface 1 output free base address_pkt_bufid_p0 was 0. The response processing state adopts time division multiplexing to set 9 states (rd_bufid_ch0_s~ rd_bufid_ch8_s) , one for each port. In these nine states, each beat jumps to one state in turn, so repeatedly. Since the ACK signal of each port may come randomly, it is necessary to cache the input ack in each state so that it can be processed in the corresponding port state.

For example, in rd_bufid_ch0_in s state:

(1) When i_bufid_ack_p0 or reg_bufid_ack_p0 is high efficient and FIFO is not empty, read one base address data in FIFO and read port 1 base address identification bit rd_value [0] is set to 1, so that the read base address is output to port 1 after waiting for 1 beat (timing of FIFO reading data based on RAM);After use, set the reg_bufid_ack_p0 zero.

(2) When i_bufid_ack_p1~i_bufid_ack_p8 is highly effective, its status is recorded, i.e reg_bufid_ack_p1~reg_bufid_ack_p8 is set to 1.

(3) When rd_valid [8] is highly effective, the read base address in FIFO is output to port 8; After use, use the rd_valid [8] clear.

(4) Jump to rd_bufid_ch1_s status.

Address write base address module: It includes two parts, one is the processing of the cache base address and multicast times input by the table lookup forwarding module, the other is the processing of the idle cache base address input by each output port.

A. 512 4-bit memories (registers) are used to record the usage times of the multicast cache base address input from the table lookup forwarding module, and the cache base address corresponds to the address index of the memory one by one.

When looking up the table, enter iv_pkt_bufid_flt and cached port input reg_pkt_bufid_wr is valid at the same time, if you look up the table and enter the base address iv_pkt_bufid_flt and cached port release base address reg_pkt_If the bufid is the same, the number of base address multicast is bufid_use_cnt[iv_pkt_bufid_FLT] to iv_pkt_bufid_cnt_flt - 4’d1；If not, the multicast times of each base address are updated, bufid_use_cnt[iv_pkt_bufid_FLT] was iv_pkt_bufid_cnt_flt , bufid_use_cnt[reg_pkt_bufid]-1；

If you only look up the table and enter iv_pkt_bufid_flt is highly efficient, only bufid needs to be updated_use_cnt[iv_pkt_bufid_FLT] was iv_pkt_bufid_cnt_flt；

When only cached port input reg_pkt_bufid_wr is highly efficient, only bufid needs to be updated_use_cnt[reg_pkt_bufid]-1。

B. The state machine is used to process the free cache base address input from the output port. The state machine mainly includes the following states:

Initialization status: initialize FIFO from 10 to 511. After reaching 511, jump to wr_bufid_ch0_S status.

Response processing status: time division multiplexing is used to set 9 states (wr_bufid_ch0_s~ wr_bufid_ch8_s) , one for each port.In these nine states, each beat jumps to one state in turn, so repeatedly.

For example, in wr_bufid_ch0_In s state:

(1) Will o_pkt_bufid_ack_p1~ o_pkt_bufid_ack_p8 is set low, indicating that the free base addresses of ports 2-9 are not written at this time;

(2) When the port is free, the base address writes signal i_pkt_bufid_wr_p0 is high and FIFO is not full, input base address and write signal are recorded, that is reg_pkt_bufid is set to iv_pkt_bufid_p0, reg_pkt_bufid_wr is set to i_pkt_bufid_wr_p0, and o_pkt_bufid_ack_p0 is set high (1 beat is valid), it indicates that the free base address of port 1 is written successfully, and port 1 is informed that the next free base address can be released; Otherwise, it will clear reg_pkt_bufid_wr, reg_pkt_bufid.

(3) If the recorded base address writes the signal reg_pkt_bufid_wr is highly efficient, and the number of base address usage is bufid_use_cnt[reg_pkt_bufid] is equal to 1, reg_pkt_bufid will be written into FIFO to release the base address.

(4) Jump wr_bufid_ch1_s status.

3.21 Design of global clock synchronization (GTS) module

3.21.1 functional analysis

GTS（global_time_ main functions of sync module are as follows.

(1) Maintain a global clock and repair the global clock: use a 48 bit counter to maintain a global clock, in which the low 17 bits represent beats and the high 31 bits represent milliseconds, which are used to represent global time.

(2) Maintain a clear counter to calculate the transparent clock: when the clear counter reaches the maximum value after a period of time, the clear pulse is sent to other modules (modules that need to calculate the transparent clock, such as each port module, host receiving and host sending module), and the clear counter of other modules is cleared at the same time,When calculating the transparent clock, the value of the clear counter is used to realize the time synchronization in the device.Since the maximum value of

time slot is 2ms, the maximum transmission delay of message in the architecture is 4ms. When using 125MHz clock, a 19 bit counter is needed.

(3) Maintain a report cycle counter, and give a report pulse signal when the internal clock goes through a cycle.

3.21.2 Internal function division

According to the function analysis of GTS module, the internal function division of GTS module is shown in the figure below.

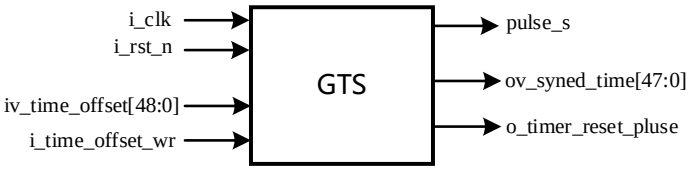

Figure 31 internal division of GTS module

The main function of GTS is to maintain a global clock and repair the global clock; Maintain a counter, clear it regularly, and send a clear pulse when it is cleared, which is used for synchronization in the device to facilitate the calculation of transparent clock.

3.21.3 Interface design

The interface information of time synchronization module GTS is shown in the figure above, and the specific meaning of interface signal is shown in the table below

Table 44. Interface signal meaning of GTS module

Interface signal	Bit width	direction	meaning
i_clk	1	input	125MHz clock
i_rst_n	1	input	Reset, low effective
Gts-csm signal definition			
iv_time_offset	49	input	This value is used to correct the global clock. The highest value is a negative number and a positive number. 0 is a positive number, indicating that the global clock needs to be increased. 1 is a negative number, indicating that the global clock needs to be decreased
i_time_offset_wr	1	input	Write signal of time offset
GTS pin signal definition			
pulse_s	1	output	Second pulse, per second / millisecond output a pulse signal to the pin
Gts-tis / TSS / QGC signal definition			
ov_syned_time	48	output	For synchronized global clocks, tis, TSS and QGC modules use global clocks for scheduling,
Gts-hrx / HTx / NRX / NTX signal definition			

Interface signal	Bit width	direction	meaning
o_timer_reset_pluse	1	output	The signal is used by HRX, HTx, NRX and NTX to clear the counter, and the clear counter is used to calculate the transparent clock
iv_offset_period	24	input	Offset clock period
iv_cfg_finish	2	input	Configuration completion signal
iv_report_period	12	input	Reporting cycle

3.21.4 Processing flow

Global clock processing flow:

(1) A 48 bit counter is used to represent the global clock, in which the lower 17 bits represent the number of beats and the higher 31 bits represent the millisecond, representing the global time of the device.When the low 17 bits count to 125000 beats, it means 1ms, carry to high position, and clear to low position;

(2) Judgment i_time_offset_Whether the write signal of WR is high or not. If it is high, it indicates that global clock correction is needed, and i_time_ value of offset, according to which the value of the global clock is increased or decreased;

(3) Output synchronized clock to tis, TSS and QGC module, which is convenient to use global clock for scheduling.

Clearing counter processing flow:

(1) The clear counter is a 19 bit counter. When the count reaches 4ms (the counter value is the number of beats, when the count reaches 0x7a120 beats, it represents 4ms), the counter will clear and send a clear pulse to HRX, HTx, NRX and NTX modules;

(2) After receiving the reset pulse, HRX, HTx, NRX and NTX modules clear the reset counter maintained by the module (it also needs to be automatically cleared when counting to 0x7a120), so as to achieve the synchronization among various modules in the equipment, and use the counter to calculate the transparent clock;

(3) The calculation of transparent clock is based on the premise that the delay of time synchronization message in a device is not more than 4 ms, and the time synchronization message comes in from the network receiving module and goes out from the network sending module. First, the network receiving module receives the time synchronization message and calculates it as T1 according to the value of the clearing counter. T1 is carried in matadata and transmitted with the message,When the time synchronization message is counted as T2 in the network sending module, compare the values of T1

and T2. If T2 is greater than T1, the value of transparent clock is t2-t1. If T2 is less than T1, the value of transparent clock is T2 + 0x7a120-t1.The processing flow of transparent clock calculation of other modules is the same as this.

Report pulse processing flow:

(1) According to the report cycle register, the report cycle is 1 second or 1 millisecond;

(2) When the reporting period is 1 millisecond and the lower 17bit of the global clock is 0, it means that the global clock has passed a whole millisecond and a reporting pulse signal is output;

(3) When the reporting period is 1 second and the lower 27bit of the global clock is 0, it means that the global clock has passed a whole millisecond and a reporting pulse signal is output;

(4) Since the global clock will adjust the clock according to the PTP synchronization, if the forward compensation adjustment results in missing a time when the lower 17bit (or 27bit) is 0, a counter needs to be maintained in this case. When the lower 17bit (or 27bit) of the counter is 0, a report pulse will be output;

(5) Since the global clock will adjust the clock according to the PTP synchronization, if a low 17bit (or 27bit) 0 time is repeated due to the backward compensation adjustment, there will be two very small whole second / millisecond time intervals. In this case, it is necessary to determine whether to output the report pulse according to whether the high 31bit (or 21bit) changes.

3.22 Design of configuration and state management (CSM) module

3.22.1 functional analysis

The main functions of CSM module are as follows:

(1) The received NMAC message is parsed and the register configuration or ram writing operation is completed.

(2) Internal maintenance input and output port send and receive message number counter.

(3) Receive the report pulse signal, construct the nmac report message and send it to the host sending module.

3.22.2 Internal function division

According to the function analysis of CSM module, the internal function division of CSM module is shown in the figure below.

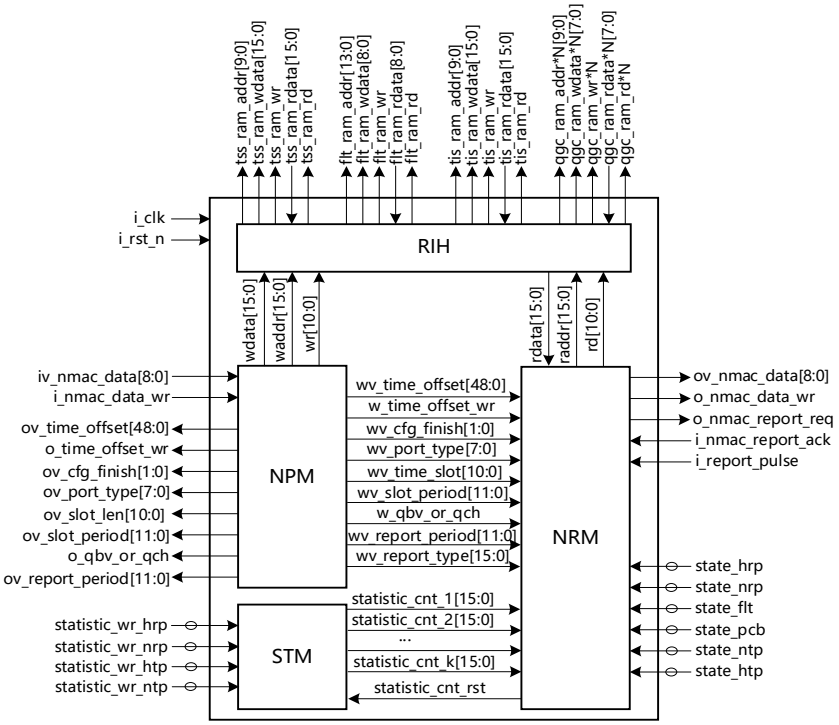

Figure 32 internal division of CSM module

NPM (nmac parse module) module The module parses the received nmac message and writes the register or RAM according to the parsed content.

STM module: the main function of this module is to receive the enable signal of each module's sending and receiving message, and maintain the sending and receiving message counter in this module.

NRM (nmac report module) module: when the report pulse is sent, the nmac message has a report request. When the preparation signal from the host sending module is high, the nmac report message is constructed. The specific content of the report is determined by the report type configured by the software.If the register is reported, the nmac message is constructed directly according to the input register signal;If the content of ram is reported, it is necessary to send the ram write signal and read the data to construct the nmac message.

RIH (RAM interface handle) module: The module is responsible for connecting the configuration and status management module with each configurable ram. When the configuration and status management module needs to write or read the ram, the module is responsible for sending the read-write request to the corresponding ram.

3.22.3 interface design

The interface information of configuration and status management module CSM is shown in the figure above, and the specific meaning of interface signal is shown in the table below

Table 45. Interface signal meaning of CSM module

Interface signal	Bit width	direction	meaning
i_clk	1	input	125MHz clock
i_rst_n	1	input	Reset, low effective
Csm-hrx signal definition			
iv_nmac_data	9	input	Received nmac message
i_nmac_data _wr	1	input	Nmac message write signal
CSM others signal definition			
ov_time_offset	49	output	Time synchronization compensation value
o_time_offset_wr	1	output	Time synchronization compensation value write signal
ov_cfg_finish	2	output	Architecture configuration completion signal
ov_port_type	8	output	Network port type (bitmap, 0: connection switching port;1: Connect to the host port)
ov_slot_len	10	output	Time slot length
ov_inject_slot_period	11	output	
ov_submit_slot_period	11	output	
ov_offset_period	24	output	
o_qbv_or_qch	1	output	Qbv or qch selection signal
ov_report_period	12	output	Reporting cycle
ov_rc_regulation_value	9	output	
ov_be_regulation_value	9	output	
ov_unmap_regulation_value	9	output	
Csm-htx signal definition			
ov_nmac_data	8	output	Reported nmac message
o_nmac_data_last	1	output	
o_nmac_report_req	1	output	Nmac report request signal
i_nmac_report_ack	1	input	Nmac report response signal
Csm-gts signal definition			
i_report_pulse	1	input	Nmac reports pulse signal
Csm-ram signal definition			
ov_tss_ram_addr	10	output	TSS module RAM read / write address

Interface signal	Bit width	direction	meaning
ov_tss_ram_wdata	16	output	TSS module ram write data
o_tss_ram_wr	1	output	TSS module ram write signal
iv_tss_ram_rdata	16	input	TSS module RAM read data
o_tss_ram_rd	1	output	TSS module RAM read signal
ov_tis_ram_addr	4	output	RAM read / write address of tis module
ov_tis_ram_wdata	9	output	Tis module ram write data
o_tis_ram_wr	1	output	Tis module ram write signal
iv_tis_ram_rdata	9	input	Reading data from ram of tis module
o_tis_ram_rd	1	output	Tis module RAM read signal
ov_flt_ram_addr	10	output	Read write address of FLT module RAM
ov_flt_ram_wdata	16	output	FLT module ram write data
o_flt_ram_wr	1	output	FLT module ram write signal
iv_flt_ram_rdata	16	input	Read data from ram of FLT module
o_flt_ram_rd	1	output	FLT module RAM read signal
i_host_inpkt_pulse	1	input	
i_host_discard_pkt_pulse	1	input	
i_host_outpkt_pulse	1		
i_host_in_queue_discard_pulse	1		
i_portN_inpkt_pulse	1	input	N=0-7
i_portN_discard_pkt_pulse	1	input	N=0-7
i_portN_outpkt_pulse	1	input	N=0-7
i_ts_inj_underflow_error_pulse	1	input	
i_ts_inj_overflow_error_pulse	1	input	
i_ts_sub_underflow_error_pulse	1	input	
i_ts_sub_overflow_error_pulse	1	input	
ov_qgcN_ram_addr	10	output	QGC module RAM read / write address n = 0-7
ov_qgcN_ram_wdata	8	output	QGC module ram write data n = 0-7
o_qgcN_ram_wr	1	output	QGC module ram write signal N = 0-7
iv_qgcN_ram_rdata	8	input	QGC module RAM read data n = 0-7
o_qgcN_ram_rd	1	output	QGC module RAM read signal N = 0-7
state			
iv_pkt_write_state	4	input	
iv_pcb_pkt_read_state	4	input	
iv_address_write_state	4	input	
iv_address_read_state	4	input	

Interface signal	Bit width	direction	meaning
iv_free_buf_fifo_rdusedw	9	input	
iv_pkt_state	3	input	Module status information input
iv_transmission_state	3	input	Module status information input
iv_pdi_state	3	input	Module status information input
iv_prp_state	2	input	Module status information input
iv_descriptor_state	3	input	Module status information input
iv_tim_state	3	input	Module status information input
iv_tom_state	2	input	Module status information input
iv_ism_state	3	input	Module status information input
iv_hos_state	2	input	Module status information input
iv_hoi_state	4	input	Module status information input
iv_pkt_read_state	3	input	Module status information input
iv_tsm_state	3	input	Module status information input
iv_bufid_state	2	input	Module status information input
iv_smm_state	3	input	Module status information input
iv_tdm_state	4	input	Module status information input
iv_osc_state_pN	2	input	Module status information input, n = 0-7
iv_prc_state_pN	2	input	Module status information input, n = 0-7
iv_gmii_read_state_pN	2	input	Module status information input, n = 0-7
iv_opc_state_pN	3	input	Module status information input, n = 0-7
i_gmii_fifo_full_pN	1	input	Module status information input, n = 0-7
i_gmii_fifo_empty_pN	1	input	Module status information input, n = 0-7
iv_descriptor_extract_state_pN	4	input	Module status information input, n = 0-7
iv_descriptor_send_state_pN	2	input	Module status information input, n = 0-7
iv_data_splice_state_pN	2	input	Module status information input, n = 0-7
iv_input_buf_interface_state_pN	2	input	Module status information input, n = 0-7

3.22.4 Processing flow

(1) Receive the pulse signal of receiving and sending messages sent by each module, and count the number counter of receiving and sending messages maintained;

(2) Analyze the received nmac message to complete the configuration of a single register or write to the specified ram;

(3) Receive the report pulse sent by the global clock module, start to construct nmac report message, and construct different nmac report message according to the configured report message type, for example: when a single register needs to be reported, fill in the values of all single registers in the nmac report message for reporting; When the status needs to be reported, all the number counters and the status of each module are filled in the nmac report message for reporting; When the content of ram needs to be reported, the data should be read out from RAM first, and then the read-out data should be filled in the nmac report message for reporting.

(4) When starting to construct the NMAC report message, first send a request signal to the host sending module. After receiving the response signal returned by the host sending module, start to construct the NMAC report message to ensure the continuous transmission of the NMAC report message until all the report messages are constructed.

Appendix A: data format definition

A. 1 TSNtag format

In the host of the sending end, it is necessary to classify and map the time sensitive, bandwidth reservation and best effort forwarding traffic according to the packet seven tuple (destination MAC, type and IP five tuple).The result of classification mapping is replaced with the DMAC field of the original message to exchange the network until the DMAC is restored in the receiving end system. The replaced DMAC field is defined as TSNtag.

A. 1.1 key of classification mapping

Table 46. Key of classification mapping

Bit width	name	describe
48	DMAC	Message destination MAC
16	ETHTYPE	Packet Ethernet type
8	protocol	Message protocol type
32	Sip	News source IP
32	Dip	Message destination IP
16	Sport	News source port
16	Dport	Message destination port

A. 1.2 classification mapping result action (TSNtag)

Since metadata is no longer carried inside the architecture, the schema receiving timestamp of time synchronization message needs to be transmitted inside the architecture by other means. The current scheme is to carry it in TSNtag, and the format of TSNtag is adjusted as follows.

Due to seq_id , frag_id, inject_addr and submit_addr in TSNtag of time sensitive packet information is useless, so these fields of the time synchronization message can be used to store the received timestamp information of the architecture. Other non time synchronization message architectures are useless to receive timestamp information, so the information of these fields can be extended.

Table 47. TSNtag of time synchronization message

Bit width	name	position	describe
3	Flow type	[47:45]	Flow type 100: synchronous message (the format of other messages is shown in the table below)
14	Flow id/IMAC	[44:31]	Static flow uses flowid, each static flow is assigned a unique flowid, dynamic flow uses IMAC address, if IMAC address is the same, it hits the same table entry in the exchange architecture.
12	Reserve	[30:19]	reserve
19	Rx_timestamps	[18:0]	The local time information of the time synchronization message received by the architecture is used to calculate the transparent clock when the architecture sends the message.

Table 48. TSNtag of non time synchronization message

Bit width	name	position	describe
3	Flow type	[47:45]	Flow type .000: St group 001: St group 010: St group 011: RC group 101: nmac group 110: be group 111: be grouping
14	Flow id/IMAC	[44:31]	Static flow uses flowid, each static flow is assigned a unique flowid, dynamic flow uses IMAC address, if IMAC address is the same, it hits the same table entry in the exchange architecture.
16	Seq id	[30:15]	It is used to identify the sequence number of the message in each stream
1	Frag flag	[14]	It is used to identify the tail after fragmentation.0: intermediate message after fragmentation 1: tail beat
4	Frag ID	[13:10]	It is used to indicate the fragment sequence number of the current fragment message in the original message
5	inject addr	[9:5]	Cache address of St stream while waiting for sending schedule at source
5	submit addr	[4:0]	The st stream caches the address when the terminal is waiting for receiving scheduling

A. 2 metadata format

The message data processed in the architecture does not carry metadata, but only carries metadata between the host and the architecture. When the host sends a message, it will carry a 64 Bytes metadata in the message header. The host receiving and processing logic puts forward the valid data in the metadata and discards the metadata after the host interface receives the message. The host send processing logic adds the 64b metadata to the header before the host interface sends the message.

Table 49. Metadata format carried by host when sending message

position	Bit width	name	meaning	remarks
[63:61]	3	pkttype	The type of message.	000: St group 001: St group 010: St group 011: RC group 100: PTP group 101: nmac group 110: be group 111: be group
[60:56]	5	inject_ addr	St injection cache address	/
[55:47]	9	outport	The output port number of the message	/
[46]	1	lookup_en	Look up enable signal	/
[45:0]	46	reserve	reserve	/

Table 50. Metadata format carried by architecture when uploading message

position	Bit width	name	meaning	remarks
[63:16]	48	ts	Global time information of synchronous message	It is used to record the global time information of the req message received by the master clock device and the global time information of the sync message received by the clock device
[15:12]	4	inport	The input port number of the message	/
[11:0]	12	reserve	reserve	/

A. 3. Pkt data format for internal transmission

There are two formats for transmission between modules, one is 9bit wide;The other is 134bit.The specific differences are as follows:

pkt_ data bit width is 9 bits, including 1 bit header and 8 bit message data.Head and tail flag: 1 represents the header / tail data of the message;0 identifies the intermediate data.The details are shown in the figure below:

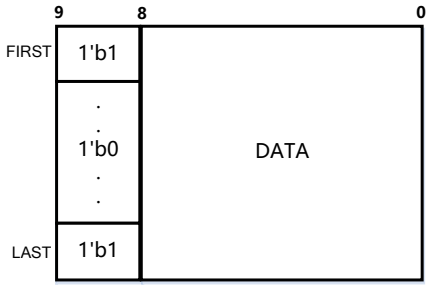

Figure 33 pkt_Data format definition

pkt_ data bit width is 134 bits, including 2bit head and tail flag, 4bit invalid byte and 128bit message data. Head and tail flag: 01 indicates the message header;11 is the intermediate data of newspaper style; 10 indicates the end of the message. Invalid bytes

are used to identify the number of invalid bytes at the end of the message.The details are shown in the figure below:

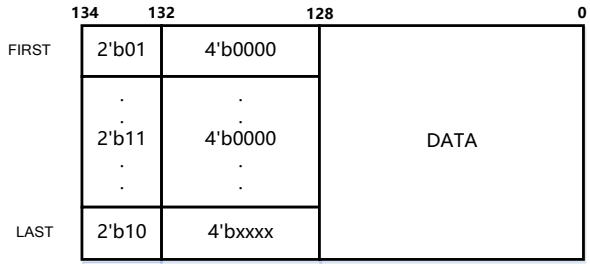

Figure 34 pkt_Data format definition

Appendix B: definition of internal table items

This chapter mainly introduces several key table items in the architecture in detail.

B.1 Design of injection time schedule

The format of the injection schedule is as follows.

Table 51. Injection schedule format

name	position	meaning	remarks
Value[15:0]	[15]	The significant bit of the table item;	Ram is used for implementation, and the depth is 1024.
	[14:5]	Send slot;	
	[4:0]	"Inject addr" in TSNtag	

The injection schedule is designed as the TS injection scheduling (TIS) module in the terminal receiving and processing logic. The injection schedule needs to work in combination with TS injection cache management. The format of TS injection cache is as follows:

Table 52. TS injection buffer data format

name	meaning	remarks
descriptor[40:0]	The descriptor used to identify the TS stream contains the lower 41 bits of the above "descriptor data format".	Using ram to achieve, the depth is 32.

The TS injection buffer is designed as a TS injection management (TIM) module in the terminal receiving processing logic. After receiving the descriptor, the module writes the descriptor into the TS injection buffer according to the injection address in the descriptor and waits for scheduling. The TS injection scheduling module looks up the TS injection schedule from the beginning. When the highest bit of the lookup result is high, it means that the lookup result is valid; when the bit is low, it means that the table lookup result is invalid, and the module will not continue to look up the table. The [4:0] of the table lookup result is the address of the next descriptor to be scheduled from the TS injection buffer, while the [14:5] of the table lookup result is the time slot

scheduled by the descriptor. After the corresponding time slot arrives, the descriptor is scheduled to transmit and read the data in the next address of the TS injection schedule. The depth of injection schedule is 32, which means that the host on a single TSNswitch architecture can support injection of 32 st streams at most.

B. 2. Design of Submission Schedule

The format of the timetable is as follows.

Table 53. Submission Schedule format

name	position	meaning	remarks
Value[15:0]	[15]	The significant bit of the table item;	Ram is used for implementation, and the depth is 1024.
	[14:5]	Send slot;	
	[4:0]	"Submit addr" in TSNtag	

The TS submission scheduling (TSS) module in the terminal receiving and processing logic is designed for the submission schedule. The injection schedule needs to work in combination with the TS submission cache management. The format of the TS submission cache is as follows:

Table 54. TS submission buffer data format

name	meaning	remarks
descriptor[12:0]	Descriptor used to identify TS stream, including pkt_bufid and inport.	Using ram to achieve, the depth is 32.

The TS submission buffer is designed as a TS submission Management (TSM) module in the terminal receiving processing logic. After receiving the descriptor, the module writes the descriptor into the TS submission buffer according to the submission address in the descriptor to wait for scheduling.The TS submission scheduling module looks up the TS Submission Schedule from the beginning. When the highest 1 bit of the query result is high, it means that the query result is valid;When the bit is low, it means that the table lookup result is invalid, and the module will not continue to look up the table.[4:0] of the table lookup result is the address of the next descriptor to be scheduled from TS submission buffer, while [14:5] of the table lookup result is the time slot scheduled by the descriptor. After the corresponding time slot arrives, the descriptor is scheduled to transmit and read the data in the next address of TS Submission Schedule. The depth of submitting schedule is 32, which means that the host on a single TSNswitch architecture can receive 32 st streams at most.

B. The design of 3-turn Publishing

The format of the forwarding table is as follows.

Table 55. Forwarding table data format

name	meaning	remarks

outport[8:0]	Output port number (bitmap), a total of 8 network interfaces and 1 host interface.	Using ram to realize, the depth is 16K

The forwarding design is in the forwarding lookup table (FLT) module, which extracts the flow according to the received descriptor_id, and then flow_id is used as the lookup address to look up the forwarding table. The result of table look-up is the output port number. If a bit is high, it means to forward to a corresponding output port. According to the result of table look-up, this module forwards the descriptor content to the corresponding output port. The depth of forwarding table is 16K, which means that the whole network composed of TSNswich can support 16K forwarding streams at most.

B.4 Design of gate control meter

The format of the gating table is as follows:

Table 56. Data format of gating table

name	meaning	remarks
gate_ctrl_vector[7:0]	Gating vector, corresponding to 8 gating information.	Using ram to achieve, the depth is 1024

The gating table is designed as a queue gating (QGC) module in the switch sending processing logic. The module looks up the table according to the received time slot information and time slot switching signal. Every time the module receives a time slot switch signal, it takes the ID of the time slot as the look-up address to look up the gating table. The result of the look-up table is the gating vector, corresponding to 8 gating signals. When a bit is high, it means to open the gating of the corresponding queue. The network output scheduling (NOS) module schedules the queue according to the gating. The depth of the gating table is 1024, which means that TSNswitch can support up to 1024 time slots as a cycle.

Appendix C: register description

Open source Fenglin TSN architecture internal configurable address space mainly has two parts, including: mdid module number and real address space. MDID module number is mainly used to distinguish different modules, and the last 20 bits are the address space used by each module. The 19th bit of the address is used to distinguish the address type. The control / table entry register is readable and writable, and the debug and version registers are read-only. The address space of each module is 1024k, of which 512k are readable and writable registers and 512k are read-only registers.The specific address meaning is as follows:

Table 57. Address format

ADDR[26:0]		
MDID[26:20]	ADDR[19]	ADDR[18:0]
MDID ： 0-127	0	The module’s control registers, table items, etc. are readable and writable

	1	read-only

The mdid number of each processing module is assigned as follows:

Table 58. Mdids and addresses in modules

processing module	CSM	TIS	TSS	QGC	GTS	FLT
MDID	0x0	0x1	0x2	0x3-0xa	0xb	0xc
address	0x0-0xfffff	0x100000-0x1fffff	0x200000-0x2fffff	0x300000-0xafffff	0xb00000-0xbfffff	0xc00000-0xcfffff

The address range is addr 0x0-0xffff

Table 59. CSM module registers

Addr	Data			
	[31:24]	[23:16]	[15:8]	[7:0]
0x0	offset_l			
0x1	offset_h			
0x2	time_slot			
0x3	cfg_finish			
0x4	port_type			
0x5	qbv_or_ach			
0x6	report_type			
0x7	report_en			
0x8	inject_slot_period			
0x9	submit_slot_period			
0xa	report_period			
0xb	offset_period			
0xc	rc_regulation_value			
0xd	be_regulation_value			
0xe	unmap_regulation_value			
0xf ~ 0xfffff	reserve			

Table 60. Specific meanings of registers

name	bit	R/W	description	default
offset_l	31:17	R/W	The lower 15 bits representing the higher value of the time offset represent milliseconds	0
	16:0	R/W	The low bit of the time offset, representing the number of beats	0
	31:17	R/W	Reserved bit	0

offset_h	16	R/W	Represents the positive and negative value of the time offset, 1 represents a positive value, and 0 represents a negative value	0
	15:0	R/W	The upper 16 bits representing the high bit value of the time offset represent milliseconds	0
time_slot	31:11	R/W	reserve	
	10:0	R/W	Slot size	0
cfg_finish	31:1	R/W	reserve	0
	0	R/W	Configuration complete register, 0Represents that the architecture is initializing and does not receive any messages, 1After initialization, it can receive nmac configuration message 2Represents the completion of configuration and can receive any message except st message 3The representative can receive any message	0
port_type	31:8	R/W	reserve	0
	7:0	R/W	Network port type register. There are 8 network ports in the architecture. Bits 0-7 of the register represent the type of port 0-7 respectively. Bits 1 represent the non cooperative type and process standard Ethernet type messages. Bits 0 represent the cooperative type and process TSN messages	0
qbv_or_ach	31:2	R/W	reserve	0
	1:0	R/W	Scheduling mode selection signal, the scheduling mechanism in network output logic is qbv mode or qch mode 0Represent qbv mode;1 stands for qch mode	0
report_type	31:16	R/W	reserve	0
	15:0	R/W	Please refer to Appendix D for reporting type	0
report_en	31:1	R/W	reserve	0
	0	R/W	Whether the report enable signal, configuration and status management module report periodically 0Representatives do not report;1 representative Report	0
inject_slot_period	31:12	R/W	reserve	0
	10:0	R/W	Injection time slot period, the period value of time slot switching within the architecture Configured value range: 1-1024	0
submit_slot_period	31:12	R/W	reserve	0

	10:0	R/W	Submit time slot period, the period value of time slot switching within the architecture Configured value range: 1-1024	0
report_period	31:12	R/W	reserve	0
	11:0	R/W	Reporting cycle, cycle value reported by configuration and status management module Configured value range: 1 (MS) or 1000 (MS)	0
offset_period	31:24	R/W	reserve	0
	23:0	R/W	Configuration period of offset compensation	
rc_regulation_value	31:9	R/W	reserve	0
	8:0	R/W	When the remaining number of bufid is less than this value, the RC message will be discarded	
be_regulation_value	31:9	R/W	reserve	0
	8:0	R/W	When the remaining number of bufid is less than this value, the be message and RC message will be discarded	
unmap_regulation_value	31:9	R/W	reserve	0
	8:0	R/W	When the remaining number of bufid is less than this value, the unmapped message will be discarded	
reserve	31:9	R/W	reserve	0

C. 2 TIS module

The address range is addr 0x100000-0x1ffff

Table 61. Address format

Addr	Data			
	[31:24]	[23:16]	[15:8]	[7:0]
send_table_N 0x100000-0x1003ff	Each item of St message sending schedule, n = 0, 1,..., 1023 send_table_0 means the 0 th sending table			
0x100400-0x1fffff	reserve			

Table 62. Specific meaning of register

name	bit	R/W	description	default
send_table_0	16	R/W	reserve	0
	15	R/W	Table entry significant bits, 0 for invalid, 1 for valid	0
	14:5	R/W	Injection time slot of St flow in one application cycle	0
	4:0	R/W	"Send addr" in TSNtag	0
……				
	16	R/W	reserve	0

send_table_one thousand and twenty-three	15	R/W	Table entry significant bits, 0 for invalid, 1 for valid	0
	14:5	R/W	Injection time slot of St flow in one application cycle	0
	4:0	R/W	"Send addr" in TSNtag	0
0x100400-0x1fffff			reserve	

C. 3 TSS module

The address range is addr 0x200000-0x2ffff.

Table 63. Address format

Addr	Data			
	[31:24]	[23:16]	[15:8]	[7:0]
submit_table_N 0x200000-0x2003ff	Each item of ST message Submission Schedule, n = 0, 1,..., 1023 submit_table_0 represents the 0 th submission table			
0x200400-0x2fffff	reserve			

Table 64. Specific meanings of registers

name	bit	R/W	description	default
submit_table_0	31:16	R/W	reserve	0
	15	R/W	Table entry significant bits, 0 for invalid, 1 for valid	0
	14:5	R/W	Submission slot of St stream	0
	4:0	R/W	"Send addr" in TSNtag	0
……				
submit _table_one thousand and twenty-three	31:16	R/W	reserve	0
	15	R/W	Table entry significant bits, 0 for invalid, 1 for valid	0
	14:5	R/W	Current slot	0
	4:0	R/W	Send_addr in TSNtag	0

C. 4 QGC module

The address range is addr 0x300000-0xaffff, where 0x300000-0x3ffff represents the gating table of the first port, and so on, there are 8 port gating.

Table 65. Address format

Addr	Data			
	[31:24]	[23:16]	[15:8]	[7:0]
port0_gate_table_N 0x300000-0x3003ff	0Gate control table of port No., n = 0, 1,..., 1023, output gate control port0_gate_table_0 represents the gating state of port 0 at the first time			
port1_gate_table_N 0x400000-0x4003ff	1Gate control table of port No., n = 0, 1,..., 1023			

port2_gate_table_N 0x500000-0x5003ff	2Gate control table of port No., n = 0, 1,..., 1023
port3_gate_table_N 0x600000-0x6003ff	3Gate control table of port No., n = 0, 1,..., 1023
port4_gate_table_N 0x700000-0x7003ff	4Gate control table of port No., n = 0, 1,..., 1023
port5_gate_table_N 0x800000-0x8003ff	5Gate control table of port No., n = 0, 1,..., 1023
port6_gate_table_N 0x900000-0x9003ff	6Gate control table of port No., n = 0, 1,..., 1023
port7_gate_table_N 0xa00000-0xa003ff	7Gate control table of port No., n = 0, 1,..., 1023

Table 66. Specific meanings of registers

name	bit	R/W	description	default
port0_gate_table_0	31:8	R/W	reserve	0
	7:0	R/W	0-7Bits represent the gating status of 8 queues from 0 to 7, 0 represents the gating status of the queue is closed, and 1 represents the gating status of the queue is on	0
……				
port7_gate_table_one thousand and twenty-three	31:8	R/W	reserve	0
	7:0	R/W	0-7Bits represent the gating status of 8 queues from 0 to 7, 0 represents the gating status of the queue is closed, and 1 represents the gating status of the queue is on	0

C. 5 FLT module

The address range is addr 0xc00000-0xcfffff

Table 67. Address format

Addr	Data			
	[31:24]	[23:16]	[15:8]	[7:0]
0xc00000-0xc03fff	forward_table_N. Represents the forwarding table, n = 0,1,2,... 16384, forward_table_0 represents the 0 th forwarding table			
0xc04000-0xcfffff	reserve			

Table 68. Specific meanings of registers

name	bit	R/W	description	default
forward_table_0	31:16	R/W	reserve	0
	8:0	R/W	The content of the forwarding table is in the form of bitmap. Bits 0-8 represent to port 0-8 respectively. The value of each bit 0 represents not to forward to the port, and 1 represents to forward to the port	0
……				

forward_table_sixteen thousand three hundred and eighty-four	31:16	R/W	reserve	0
	8:0	R/W	The content of the forwarding table is in the form of bitmap. 0-8 bits represent to port 0-8 respectively. The value of each bit is 0, which means not forwarding to the port, and 1, which means forwarding to the port	0
0xc04000-0xcfffff			reserve	

Appendix D: NMAC message format

In the Ethernet frame header, the type field is 0x1662, which means that it is encapsulated as NMAC message. NMAC is divided into configuration message and report message. Configuration message is sent from host to architecture, and report message is sent from architecture to host; Therefore, the default message sent by the host is the configuration message, and the default message sent by the architecture is the report message, which is not identified in the message.

D. 1 configuration message format

Configuration message format: the count field (8bit) is used in the message to indicate the number of configuration entries contained in the message. The minimum size of the message is 64 bytes. Finally, the message with less than 64 bytes needs to be zeroed.The encapsulation of nmac command in Ethernet message is shown in the figure

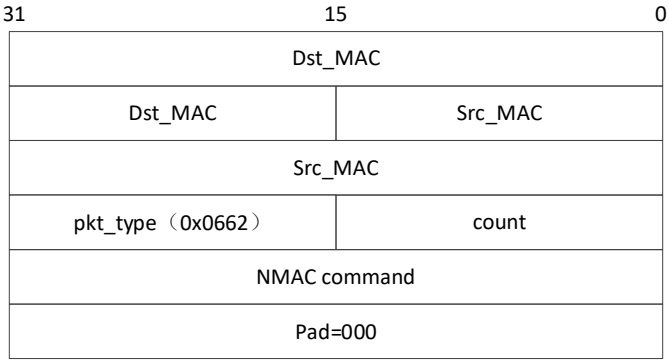

Figure 35 nmac configuration message format

The format of the nmac command is as follows:

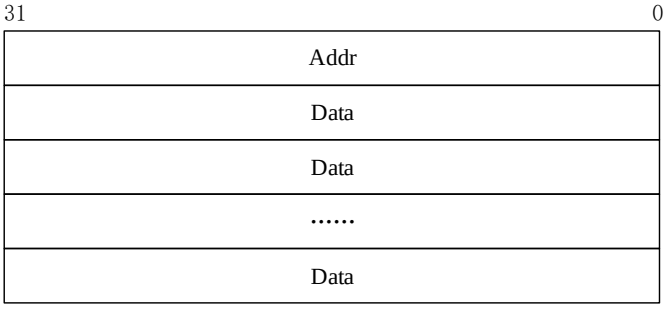

Figure 36 nmac command format

When the number of registers configured is 1, nmac command includes 32 bit addr and 32 bit data;When the number of registers configured is n (n > 1), the nmac command includes 32bitaddr and N * 32bit data. The first data uses addr as the write address of ram, and the second and subsequent data uses addr cycle plus 1 as the write address of ram.

D. 2 report message format

Report message format: the message type is nmac message (0x1662)

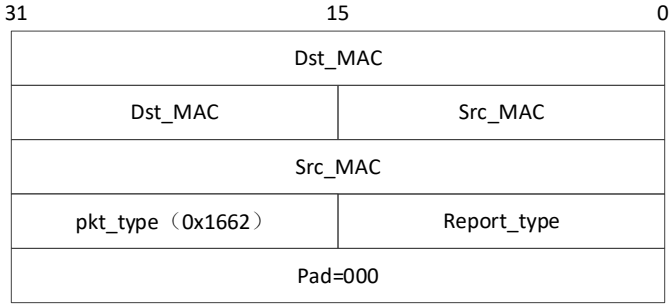

Figure 37 nmac report message format

Table 69. Report type format is as follows:

Reporting type (16bit)		meaning
High 6bit	Low 10bit	
000000 Single register	0	A single register for configuration, including configuration completion register, port status register Time slot size register, time offset register, report period register, report type register and application period register
000001 Forwarding table	0	Articles 0-63 for publication
	1	Sections 64-127, reprinted
	2-255	Sections 128-16383, reprinted
000010 Injection schedule	0	Clauses 0-63 injection schedule
	1	Clauses 64-127 injection schedule
	2-15	Section 128-1023 injection schedule
000011 Submit schedule	0	Clauses 0-63 injection schedule
	1	Clauses 64-127 injection schedule
	2-15	Section 128-1023 injection schedule
000100- 001011 P0-p7 output gating table	0	Clauses 0-63 injection schedule
	1	Clauses 64-127 injection schedule
	2-15	Section 128-1023 injection schedule
001100 xx_state	0	Refer to Appendix D for details

Single register, report_ type of high 6bit is 0000 and low 10bit is 0.

Report format_ high 6bit of type is 00000 1, and the low 10bit is the reported block. There are 16K forwarding tables in total, and each forwarding table takes up 2 bytes (9bit), so each message can carry 64 messages, and 256 messages in total.

Report format_ high 6bit of type is 000010, and the low 10bit is the reported block. There are 1024 injection timetables in total, and each forwarding takes 2 bytes (9bit). Therefore, each message can carry 64 messages, and a total of 16 messages are required.

Report format_ high 6bit of type is 000011, and the low 10bit is the reported block. There are 1024 injection timetables in total, and each forwarding takes up 2 bytes (9bit). Therefore, each message can carry 64 messages, and a total of 16 messages are required.

The gating table is divided according to ports. Each gating takes up one ram and has 8 ports in total.Each port has two pieces of ram, and a total of 16 pieces of RAM are required_ specific format of type is shown in the table above.

The status report content is provided by each module. For details, refer to appendix D.

The internal meaning of the message is as follows:

1）Single register report message:

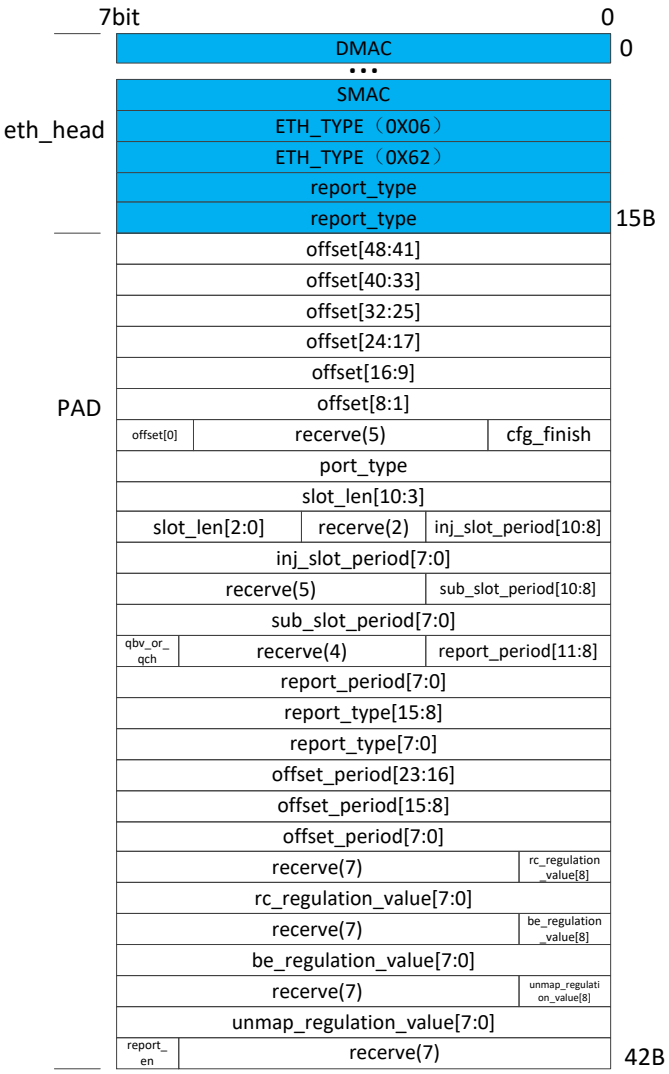

Figure 38 single register nmac report message format

2）Status report message of each module:

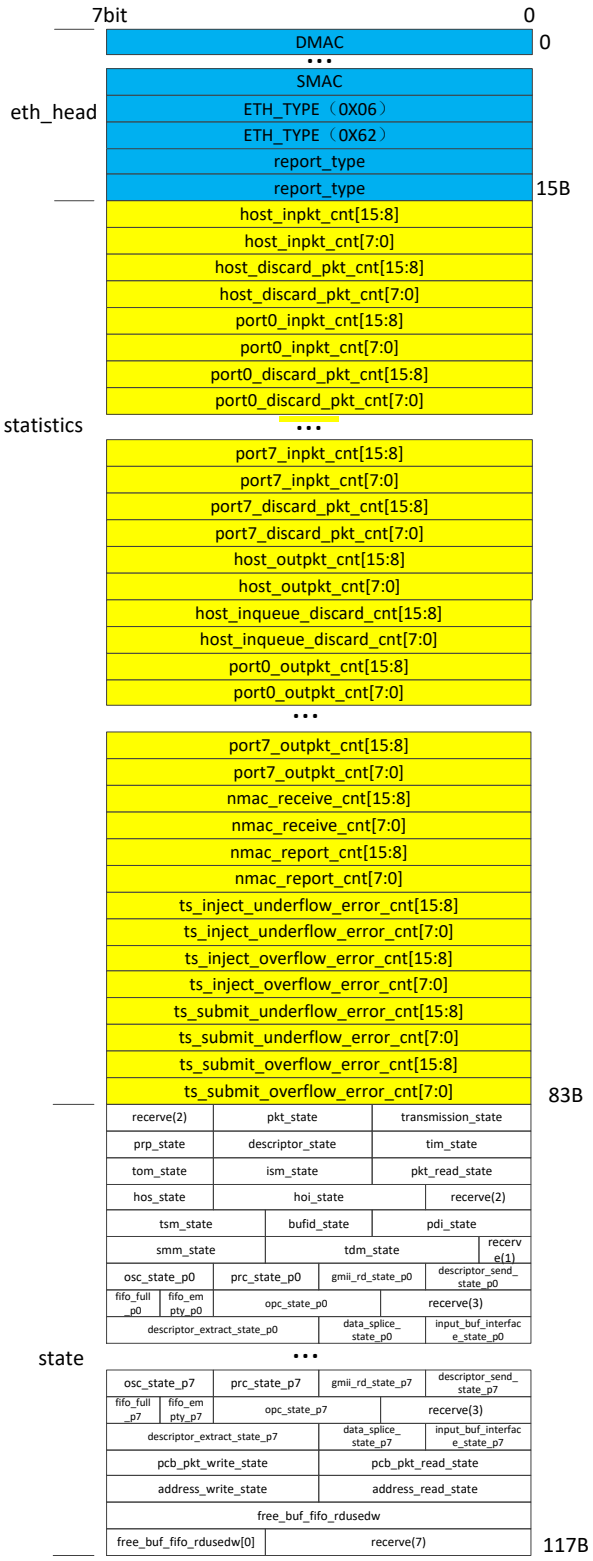

Figure 39 nmac report message format of each module status

3）Table item report message:

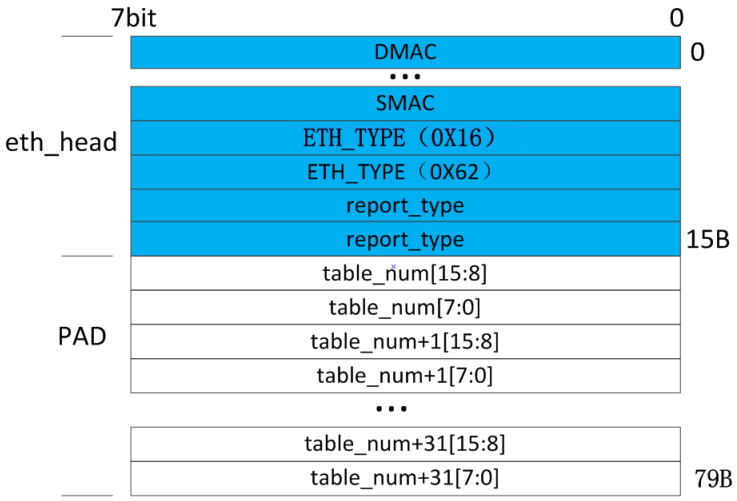

Figure 40 format of table item nmac report message

The status information reported by each module through nmac is shown in the following table. The status reported by nmac mainly includes the sending and receiving message counter of each port and the status information of each module.

Table 70. Statistics of counters reported by each module

name	Bit width	meaning
host_inpkt_cnt	16	Counter for the number of messages received by the host port
host_discard_pkt_cnt	16	Counter for the number of discarded messages received by the host port
port0_inpkt_cnt	16	0Counter of number of messages received by No.1 network port
port0_discard_pkt_cnt	16	0Counter for the number of discarded messages received by No.1 network port
port1_inpkt_cnt	16	1Counter of number of messages received by No.1 network port
port1_discard_pkt_cnt	16	1Counter for the number of discarded messages received by No.1 network port
port2_inpkt_cnt	16	2Counter of number of messages received by No.1 network port
port2_discard_pkt_cnt	16	2Counter for the number of discarded messages received by No.1 network port
port3_inpkt_cnt	16	3Counter of number of messages received by No.1 network port
port3_discard_pkt_cnt	16	3Counter for the number of discarded messages received by No.1 network port
port4_inpkt_cnt	16	4Counter of number of messages received by No.1 network port
port4_discard_pkt_cnt	16	4Counter for the number of discarded messages received by No.1 network port
port5_inpkt_cnt	16	5Counter of number of messages received by No.1 network port
port5_discard_pkt_cnt	16	5Counter for the number of discarded messages received by No.1 network port

port6_inpkt_cnt	16	6Counter of number of messages received by No.1 network port
port6_discard_pkt_cnt	16	6Counter for the number of discarded messages received by No.1 network port
port7_inpkt_cnt	16	7Counter of number of messages received by No.1 network port
port7_discard_pkt_cnt	16	7Counter for the number of discarded messages received by No.1 network port
host_outpkt_cnt	16	Counter for the number of messages sent by host port
host_inqueue_discard_cnt	16	Counter for the number of discarded messages when the host sending module is queued
port0_outpkt_cnt	16	0Counter for the number of messages sent by No.1 network port
port1_outpkt_cnt	16	1Counter for the number of messages sent by No.1 network port
port2_outpkt_cnt	16	2Counter for the number of messages sent by No.1 network port
port3_outpkt_cnt	16	3Counter for the number of messages sent by No.1 network port
port4_outpkt_cnt	16	4Counter for the number of messages sent by No.1 network port
port5_outpkt_cnt	16	5Counter for the number of messages sent by No.1 network port
port6_outpkt_cnt	16	6Counter for the number of messages sent by No.1 network port
port7_outpkt_cnt	16	7Counter for the number of messages sent by No.1 network port
nmac_receive_cnt	16	Counter for the number of nmac messages received by CSM module
nmac_report_cnt	16	Counter for the number of nmac messages reported by CSM module
ts_inj_underflow_error_cnt	16	St message injection underflow error counter
ts_inj_overflow_error_cnt	16	St message injection overflow error counter
ts_sub_underflow_error_cnt	16	St message submission underflow error counter
ts_sub_overflow_error_cnt	16	St message submission overflow error counter
total	/	34544bit, 68B

Table 71. Status statistics reported by each module

name	Module	Bit width	meaning
pkt_state	PMD>TRW	3	TRW state machine
transmission_state	PMD>TRR	3	TRR state machine
prp_state	HRX>PRP	2	PRP state machine
descriptor_state	PMD>PDG	3	PDG state machine
tim_state	TIM	3	Tim state machine
tom_state	PMD>TOM	2	Tom state machine
iv_ism_state	TIS>ISM	3	Ism injection scheduling state machine
pkt_read_state	HTX> HRC	3	HRC state machine

hos_state	HOS	2	Hos state machine
hoi_state	HTX>HOI	4	Hoi state machine
tsm_state	TSM	3	TSM state machine
bufid_state	TSM	2	bufid state machine
pdi_state	HRX	3	PDI state machine
smm_state	TSS>SSM	3	SSM submit scheduling state machine
tdm_state	FLT>TDM	4	State machine of TDM
osc_state_p*N	NOS>OSC	2	State machine of OSC (n = 0-7)
prc_state_p*N	NOS>PRC	2	The state machine of PRC (n = 0-7)
gmii_read_state*N	NRX>GR	2	State machine of GW (n = 0-7)
descriptor_send_state*N	FPM>DSM	2	State machine of DSM (n = 0-7)
gmii_fifo_full*N	NRX	1	Input clock domain switching FIFO full signal (n = 0-7)
gmii_fifo_empty*N	NRX	1	Input clock domain switching FIFO null signal (n = 0-7)
opc_state_p*N	NOS>OPC	3	OPC state machine (n = 0-7)
descriptor_extract_state*N	FPM>DEM	4	State machine of DEM (n = 0-7)
data_splice_state*N	FPM>DSP	2	State machine of DSP (n = 0-7)
input_buf_interface_state*N	IBI	2	State machine of IBI (n = 0-7)
pkt_write_state	PCB>PW	4	State machine of PW
pkt_read_state	PCB>PR	4	State machine of PR
address_write_state	PCB>AW	4	State machine of aw
address_read_state	PCB>AR	4	The state machine of AR
free_buf_fifo_rdusedw	PCB	9	The number of free address management FIFO