# TSN 芯片设计方案
# （V4.0）



OpenTSN 开源项目组

2020 年 8 月

# 版本历史

| 版本 | 修订时间 | 修订内容 | 修订人 | 文件标识 |
|---|---|---|---|---|
| v2.0 | 2020/6/6 | 描述符格式：将 outport 字段位宽增加了 5bit，描述符整体位宽相应增加了 5bit。 | | |
| | 2020/6/6 | metadata 格式：将 outport 字段位宽增加了 5bit，metadata 位宽相应增加了 5bit。 | | |
| | 2020/6/6 | 转发表的表项:表项内容增加了 5bit，用 bitmap 形式代表需要转发的输出端口。 | | |
| | 2020/6/6 | 转发查表模块的处理：转发处理，需要根据输出端口号,对多播报文的描述符进行复制与转发,同时需要将复制的个数发往报文集中缓存模块。 | | |
| | 2020/6/6 | 报文集中缓存:增加 512 个寄存器进行多播报文计数的缓存。在 pkt_bufid | | |

| | | | | |
|---|---|---|---|---|
| | | 使用完之后进行释放时，需要先检查对应的计数值是否为0，为0则释放；不为0则不释放。 | | |
| | 2020/6/6 | 去除队列缓存仲裁模块。将队列的管理划分到各个输出端口处理逻辑。 | | |
| | 2020/6/6 | 主机输出端口处理逻辑增加一个FIFO进行队列的管理。 | | |
| | 2020/6/6 | 网络输出端口处理逻辑增加一个深度为512，位宽为9bit的RAM进行4个队列的集中缓存管理，与之前设计一致，但不需要做时分复用。 | | |
| v3.0 | 2020/6/19 | 芯片内部处理的报文不再携带metadata信息，metadata只用于主机与芯片通信，在主机接口接收到报文后将metadata丢弃，在主机接口往主机发送报文时需要加上metadata信 | | |

| | | | | |
|---|---|---|---|---|
| | | 息。 | | |
| | 2020/6/19 | Metadata 格式：由于 metadata 只用于主机与芯片通信，格式进行了调整 | | |
| | 2020/6/19 | TSNTag 格式：由于按照原 TSNtag 的字段定义，ptp 报文很多字段没有使用到，因此将芯片的接收 ptp 报文的本地时间信息记录在 TSNTag 中。而其他报文的 TSNTag 还是保持原来的格式。 | | |
| | 2020/6/19 | 主机发送模块在处理时间同步报文时，不再是将三个时间信息(芯片接收到时间同步报文的本地时间信息、芯片发送时间同步报文的本地时间信息、主/从时钟设备接收 ptp 报文的全局时间信息）发往主机，而是将芯片内部所产生的延迟计算到时间同步报文的透明时 | | |

| | | | | |
|---|---|---|---|---|
| | | 钟域，只将 timestamps 发往主机。 | | |
| | 2020/6/19 | 网络接收处理逻辑需要将长报文进行切片，第一片包含 128B 的数据，之后的每一片不大于 122B 数据，原因是为软件预留 6B 的位置添加 TSNTag。每一个切片分组都使用一个 pkt_bufid 进行数据的缓存处理。 | | |
| | 2020/6/19 | 主机发送模块需要将不足 64B 的数据进行识别并填 0 以保证数据是合法的，并在 metadata 中标识此分组中有效的字节数。 | | |
| | 2020/6/19 | 主机接收模块根据接收到的报文的 metadata 中的 pkt_type 信息构造描述符，而不是使用 TSNTag 中的 pkt_type。 | | |

| | | | | |
|---|---|---|---|---|
| | 2020/6/19 | 网络输出处理逻辑需要为需重组的长报文规划一个专门的队列进行描述符的缓存。 | | |
| | 2020/6/19 | 网络输出调度模块在管理队列时，需要等待长报文的分片全部进入队列之后才能允许该队列被调度。 | | |
| | 2020/6/19 | 当开始调度这一队列时，下一次调度就需要有效调度此队列，直到报文全部调度完成。 | | |
| | 2020/6/19 | 网络发送模块需要将从此队列调度出来的数据连续的从 gmii 接口输出。 | | |
| v3.1 | 2020/6/22 | 由于在芯片内部表项未配置完成时，无法对输入的报文进行正确的处理，因此需要等待内部的表项都配置完成后，才能通知接口接收数据。主机完成了芯片所有的配置后，下发 | | |

| | | | | |
|---|---|---|---|---|
| | | 一个配置完成的命令，芯片解析后将值配置到对应寄存器上。 | | |
| v3.2 | 2020/6/29 | 由于讨论后将 metadata、描述符以及 TSNTag 的格式进行了调整，在文档中也相应地进行修改。 | | |
| v3.4 | 2020/7/11 | 对 TSNTag 以及 metadata 中报文类型字段进行重新定义，将值为 0、1、2 的都映射成 TS 流。 | | |
| V4.0 | 2020/8/6 | 由于之前的设计是将报文分成 128+122+122+…形式，在报文重组时会出现问题。现需要将报文分成 128+112+112+…，保证除最后一片以外都是 16 的倍数。设计方案是在后续分片的报文都增加 6B 的 TSNTag 与 10B 的全零数据。 | | |

# 目录

# 1.总体架构

## 1.1 接口说明

| 信号名 | 位宽 | 类型 | 说明 |
|---|---|---|---|
| i_osc_clk | - | 输入 | 晶振时钟 |
| i_rst_n | - | 输入 | 系统复位 |
| 9*GMII | 9*22bit | 输入/输出 | 9 路 GMII 接口，具体信号如下（gmii_txd、gmii_tx_en、gmii_tx_er、gmii_tx_clk、gmii_rxd、gmii_rx_dv、gmii_rx_er、gmii_rx_clk） |
| o_timer_pulse | 1 | 输出 | 秒脉冲 |

## 1.2 主要功能和指标

### 1.2.1 主要功能

端系统功能：

■根据报文七元组信息进行流映射，并将映射结果(TSNTag)替换掉报文 DMAC 字段。（具体格式可参考附录 A）

■根据注入时刻表对时间敏感报文进行缓存管理以及延迟注入（发往网络）

■根据提交时刻表对时间敏感报文进行缓存管理以及延迟提交（发往主机）

交换功能：

■根据 TSNTag 中的 Flow ID/IMAC 进行查表交换

■基于 Qbv、Qch 的 TSN 确定性交换

通用功能：

■基于 1588 时间同步协议进行节点间以及节点内的时间同步

■以硬件调度周期为调度单位，对时间敏感流量、带宽预约流量和尽力转发流量进行严格优先级调度

■集中式报文交换缓存管理

### 1.2.2 设计指标

端系统指标：

■每个终端节点支持 32 条流

■作为端系统时 TS 流量最大支持 1024 个硬件调度时间槽的延迟

交换指标：

■交换容量 16Gbps

■交换延迟小于 30us

■支持 16K 条流的转发配置

通用指标

■支持 9 个千兆以太网接口（8 个网络接口，1 个主机接口）

■同时支持时间敏感、带宽预约和尽力转发三种流量的转发交换

■支持 IEEE 802.1AS、802.1Qch、802.1Qbv、802.1Qcc 标准

■硬件调度时间槽设置范围为[16us，2ms]

■单跳时间同步精度优于 50ns

## 1.3 内部功能划分

TSN 芯片顶层逻辑主要包含以下几个模块，如下图所示。

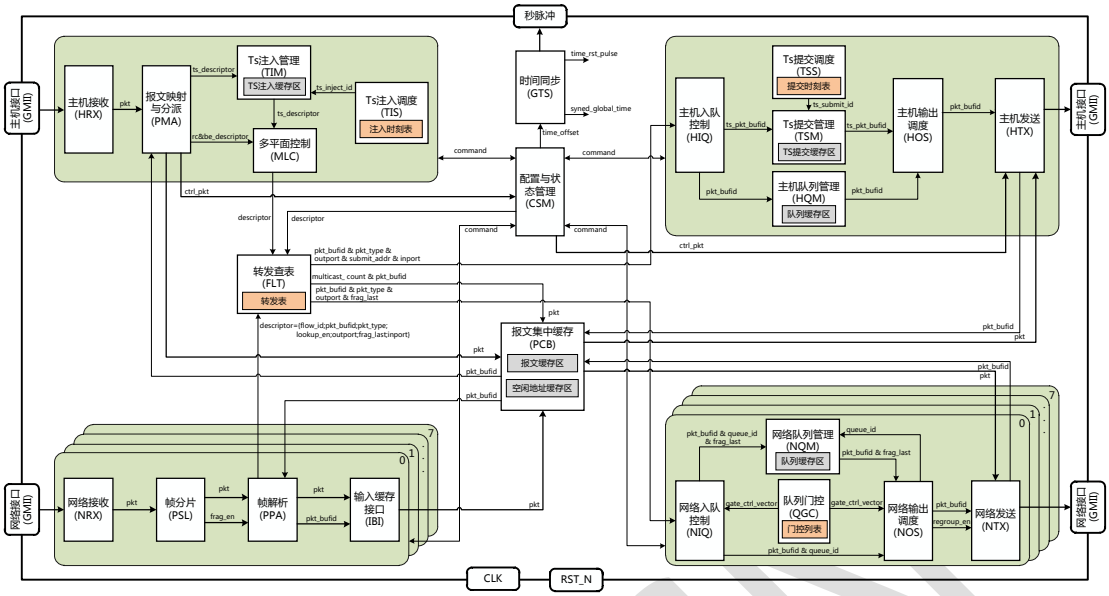


图1-1 TSN 芯片顶层架构图

下面将顶层划分为五大模块进行介绍，分别对应顶层中的：端输入处理逻辑、端输出处理逻辑、交换输入处理逻辑、交换输出处理逻辑、以及内部处理逻辑。

### 1.3.1 端输入处理逻辑

**HRX(Host RX)主机接收模块：**主要功能是接收主机接口下发的报文，完成报文从外部时钟域到芯片内部时钟域的切换，以及完成时间同步报文的时间戳记录。主机接口接收的报文都携带了 metadata，这部分逻辑由主机中的软件进行实现。模块内部维护一个寄存器，根据此寄存器的值判断是否接收并处理数据。

**PMD(Pkt Map and Dispatch)报文映射与分派模块：**主要功能是根据报文的七元组信息进行映射（目前在主机的软件实现映射功能）、构造报文描述符（报文描述符的格式如下）以及区分报文类型并分派到不同的目的模块。映射的功能主要是将外部的报文映射成内部处理的报文格式，并将映射结果(参考附录 A：TSNTag 格式)填充到报文

的 DMAC 位置。报文描述符的构造，是根据报文集中缓存模块分配给本接口的 pkt_bufid 以及报文映射后的信息(参考附录 A 和 B)构造一个能够标识报文的描述符数据。报文分派，需要先区分报文的类型是 TS、BE 和 RC 还是 NMAC 协议（NMAC 报文格式定义参考附录 E），再将不同类型的报文描述符分派到不同的目的模块，并将报文数据也分派到不同目的模块。另外，本模块还需将主机下发报文的 metadata 丢弃。

表1-1 描述符格式

| 内容 | 位宽 | 位置 | 含义 |
|---|---|---|---|
| inject_addr/ submit_addr | 5 | [45:41] | TS 流注入/提交时缓存的地址。 |
| frag_last | 1 | [40] | 用于标识分片后的报文最后一个片，未分片默认为 1。 |
| inport | 4 | [39:36] | 报文的输入端口，用于最终构造 metadata。 |
| pkt_type | 3 | [35:33] | 报文类型，用于入队控制时区分报文类型，选择队列。 |
| flow ID/IMAC | 14 | [32:19] | 流 ID，用于 FLT 模块查表时的地址索引。 |
| lookup_en | 1 | [18] | 查表使能，用于 FLT 判断该报文是否需要进行查表操作。 |
| outport | 9 | [17:9] | 输出端口号，用于 FLT 模块区分输出端口。 |
| pkt_bufid | 9 | [8:0] | 报文在缓存区中缓存的 ID 号，用于标识每个报文。 |

**TIM(TS Inject manage)TS 注入管理模块：**主要功能是将接收到的 TS 报文描述符写入到由发送端主机的软件映射后的地址（在

TSNTag 中标识）进行缓存（缓存区的格式如下），并等待 TS 注入调度模块的调度信号进行 TS 报文描述符的读取和发送。

表1-2 TS 注入缓存区数据格式

| 名称 | 含义 | 备注 |
| --- | --- | --- |
| descriptor[40:0] | 用于标识 TS 流的描述符，所包含的内容如上“Descriptor 数据格式”的低 41 位。 | 使用 RAM 进行实现，深度为 32。 |

**TIS(TS Inject Schedule)TS 注入调度模块：**主要功能是根据当前时间槽信息进行查找注入时刻表（表内容如下），断定是否有 TS 报文描述符需要调度，并通知 TS 注入管理模块完成 TS 报文描述符的提取与发送。

表1-3 注入时刻表格式

| 名称 | 位置 | 含义 | 备注 |
| --- | --- | --- | --- |
| Value[15:0] | [15] | 表项有效位； | 使用 RAM 进行实现，深度为 1024。 |
| | [14:5] | 发送 Slot ； | |
| | [4:0] | TSNTag 中的“inject addr” | |

**MLC(Multiple Level Control )多平面控制模块：**主要功能是判断报文描述符所代表的报文体是否需要经过查表转发。根据多平面控制寄存器对报文描述符进行修改。如果支持多平面转发则由软件进行帧复制与消除控制，描述符不变。如果不支持，则根据配置的输出端口修改描述符信息、查表控制信息。（目前这部分逻辑暂不实现）

### 1.3.2 端输出处理逻辑

**HIQ(Host Input Queue)主机入队控制模块：**主要功能是将接收到

的 pkt_bufid 根据接收到的 pkt_type 数据进行报文类型的区分后发送给对应的目的模块。将 TS 的 pkt_bufid 发送给 TS 提交管理模块等待调度；将非 TS 的 pkt_bufid 发送给主机队列管理等待调度。

**TSS(TS Submit Manage)TS 提交管理模块：**主要功能是将接收到的 TS 的 pkt_bufid 写入到由发送端主机的软件映射后的地址（在 TSNTag 中标识）进行缓存（缓存区的格式如下），并等待 TS 提交调度模块的调度信号进行 TS 的 pkt_bufid 的提取并发送。

表1-4 TS 提交缓存区数据格式

| 名称 | 含义 | 备注 |
| --- | --- | --- |
| pkt_bufid[8:0] | 用于标识 TS 流的 pkt_bufid。 | 使用 RAM 进行实现，深度为 32。 |

**TSM(TS Submit Schedule)TS 提交调度模块：**主要功能是根据当前时间槽信息进行查找提交时刻表（表内容如下），判断是否有 TS 报文描述符需要调度，并通知 TS 提交管理模块完成 TS 的 pkt_bufid 的读取与发送。

表1-5 提交时刻表格式

| 名称 | 位置 | 含义 | 备注 |
| --- | --- | --- | --- |
| Value[15:0] | [15] | 表项有效位； | 使用 RAM 进行实现，深度为 1024。 |
| | [14:5] | 提交 Slot ； | |
| | [4:0] | TSNTag 中的“submit addr” | |

**HQM (Host Queue Manage)主机队列管理模块：**主要功能是对发往主机的非 TS 流的 pkt_bufid 进行缓存管理（缓存区的格式如下），等待主机输出调度模块的调度信号进行 pkt_bufid 的调度输出。

| 名称 | 含义 | 备注 |
| --- | --- | --- |
| pkt_bufid[8:0] | 用于标识非 TS 流的 pkt_bufid。 | 使用 FIFO 进行实现，深度为 256。 |

**HOS (Host Output Schedule)主机输出调度模块**：主要功能是按照优先级调度策略对 pkt_bufid 进行调度与发送。当 TS 提交调度模块调度出一个 TS 的 pkt_bufid 时，本模块需要先将此 pkt_bufid 发送给下级模块，只有当没有 ts_pkt_bufid 调度时才能向主机队列管理模块进行调度的申请完成 be_rc_pkt_bufid 的读取与发送。

**HTX(Host TX)主机发送模块**：主要功能是从报文缓存区中读取报文并释放 pkt_bufid、将数据报文由主机接口传输给外部主机。读取报文时，需要先将 pkt_bufid 映射成报文读取地址，并根据此地址往报文集中缓存模块进行报文数据的读取，同时需要将此 pkt_bufid 释放给报文集中缓存模块以便后续进行芯片的报文使用。

### 1.3.3 交换输入处理逻辑

**NRX(Network RX)网络接收模块**：主要功能是接收网络接口发送的报文，完成报文从外部时钟域到芯片内部时钟域的切换，以及完成芯片接收时间同步报文的时间信息记录，并在 TSNTag 中进行记录。模块内部维护一个寄存器，根据此寄存器的值判断是否接收并处理数据。

**FFR(Frame FRagment)帧分片模块**：主要功能是根据报文的长度域字段对报文进行转发或分片后转发。当报文的长度少于128B，则直接将报文进行发送；当报文的长度大于128B，需要将报文切分成

每一片不大于128B且不少于64B的报文。切分后的第一片报文长度为128B，第二片以及之后的非最后两片报文长度为112B（保证是16B的倍数，且为主机添加TSNTag预留6B的位置），倒数第二片必须是16B的倍数，最后一片报文不小于64B。每传输一个分片都进行计数，再根据报文长度域字段得知剩余的报文长度，当长度少于256B时，需要将剩下的报文进行平均分配，由此来保证每一片不少于64B。

**FPA(Frame PArse)帧解析模块：**主要功能是提取帧的 DMAC（TSNTag）中的信息，并构造报文描述符数据。还需根据报文的以太网类型字段判断此报文是否经过分类映射，若未经过分类映射则在构造描述符时默认送往主机。在接收到帧完整性检测模块的长报文分片信号时，第二片以及之后的分片报文都需要使用根据第一片报文构造描述符，只是存放的 pkt_bufid 会不同。在往输入缓存接口模块发送数据之前，需要将一拍 8bit 的数据转换成一拍 128bit 的数据。

**IBI(Input Bufm_memory Interface)输入缓存接口模块：**主要功能是将报文数据发送给报文集中缓存模块进行缓存。本模块接收帧解析模块的 128bit 数据使用两个寄存器进行缓存，其中任何一个寄存器有数据则往报文集中缓存门控发送一个写信号，同时将进入本模块的数据写入另一个寄存器。

### 1.3.4 交换输出处理逻辑

**NIQ(Network Input Queue)网络入队控制模块：**主要功能是将 pkt_bufid 写入到网络队列管理模块中进行缓存。本模块需要根据接收到的报文类型信息、队列门控模块发送的门控信息进行 queue_id 的

映射，并将 pkt_bufid 与 queue_id 发送给网络队列管理模块进行缓存。同时将 pkt_bufid 与 queue_id 发送给网络输出调度模块，以便对队列状态进行管理。

**QGC(Queue Gate Control)队列门控模块：**主要功能是控制当前输出端口队列打开或关闭。本模块实现遵循 Qbv 标准，根据芯片的全局时钟进行门控列表的读取，并将门控列表(门控列表的格式如下)中 8 个队列的门控开关信息发送给网络入队控制模块和网络输出调度模块。

表1-6 门控列表数据格式

| 名称 | 含义 | 备注 |
| --- | --- | --- |
| gate_ctrl_vector[7:0] | 门控向量，对应 8 个门控信息。 | 使用 RAM 进行实现，深度为 1024。 |

**NOS(Network Output Schedule)网络输出调度模块：**主要功能是根据调度后的队列信息从网络队列管理模块提取出 pkt_bufid。本模块需要根据当前队列信息与队列门控模块发送的门控信息进行计算，得出一个最优先调度的队列，并从队列缓存仲裁模块的对应队列中得到 pkt_bufid。在调度“需重组的分组”队列时，需要将完整的长报文对应的 pkt_bufid 提取出来之后才能调度其他的队列。

**NTX(Network TX)网络发送模块：**主要功能是主要功能是从报文缓存区中读取报文并释放 pkt_bufid、计算时间同步报文的透明时钟、完成报文从内部处理时钟域到外部 PHY 芯片的时钟域的切换、将数据报文由网络接口传输。读取报文时，需要先将 pkt_bufid 映射成地址，并根据此地址往报文集中缓存模块进行报文数据的提取，同时需

要将此 pkt_bufid 归还给报文集中缓存模块以便后续进入芯片的报文使用。当网络输出调度模块开始调度“需重组的分组”队列时，本模块需要将所有分片后的报文 pkt_bufid 提取出来的报文连续地传输出去。

1.3.5 内部处理逻辑

**FLT(Forward Lookup Table)转发查表模块：**主要功能是根据接收到的报文描述符提取出 flow_id、根据 flow_id 进行查找转发表（转发表的格式如下）、根据查表得到的输出端口进行 pkt_bufid 和 pkt_type 的转发。当描述符中 lookup_en 为低（代表本描述符不需要查表）时，直接提取出描述符中的 outport 字段作为输出端口号进行转发。当得到的输出端口多位高，意味处理的描述符代表的报文是个多播报文，需要将 pkt_bufid 和 pkt_type 往多个端口进行转发，同时将 pkt_bufid 与输出端口的数量发送给报文集中缓存模块进行组播计数的缓存。

表1-7 转发表数据格式

| 名称 | 含义 | 备注 |
| --- | --- | --- |
| outport[8:0] | 输出端口号(bitmap)，总共 8 个网络接口与 1 个主机接口。 | 使用 RAM 进行实现，深度为 16K。 |

**PCB(Pkt Centralize Bufm_memory)报文集中缓存模块：**主要功能是对芯片需要转发的所有报文进行集中缓存（报文缓存区的格式如下）、为每个缓存的报文分配一个 pkt_bufid 进行标识、对所有空闲的 pkt_bufid 进行缓存（空闲地址缓存区的格式如下）。为所有 pkt_bufid 增加寄存器对输出端口数量进行计数，当 pkt_bufid 使用完进行释放的时候需要检测该 pkt_bufid 对应的计数值，只有当计数值为 0 才能

进行释放，当不为 0 意味着该 pkt_bufid 存着一个组播报文并且该报文还未从所有需要输出的端口输出。

报文缓存区将 64KB 的空间划分成 512 个报文缓存块，每个报文缓存块能缓存一个长度最多为 128B 的报文。

表1-8 报文缓存区数据格式

| 地址[8:0] | 内容[133:0] |
|---|---|
| 0-7 | 第 1 个报文缓存块 |
| 8-15 | 第 2 个报文缓存块 |
| … | .. |
| 4088-4095 | 第 512 个报文缓存块 |

表1-9 空闲地址缓存区数据格式

| 名称 | 含义 | 备注 |
|---|---|---|
| pkt_bufid[8:0] | 当前“报文缓存区”中空闲的报文缓存块 ID 号。 | 使用 RAM 进行实现，深度为 512。 |

**GTS(Global Time Synchromzation)时间同步模块**：主要功能是提取全局的时间信息、根据外接 cpu 计算的 offset 值进行时间的修正。

**CSM(Configuration and State Manage )配置与状态管理模块**：主要功能是接收外接 cpu 下发的 NMAC 控制报文、完成 NMAC 报文的解析、构造命令流对各个模块的可读写的寄存器进行读写控制、构造 NMAC 报文往主机上报。

## 1.4 芯片主要处理流程

芯片内部对报文的处理可分为两部分，一部分是将报文的数据进行集中地缓存；另一部分是构造报文的描述符来标识报文，根据描述符进行查表、转发、bv 调度、ts 的发送调度、ts 的接收调度等一系列

的操作。

报文缓存部分处理流程：

（1）端/交换输入处理模块从主机/网络接口接收到报文数据，经过各自输入处理之后将报文数据构造成每拍 128bit 数据的形式，并发送给报文集中缓存模块进行集中缓存。

（2）报文缓存区将整个缓存区划分成 512 个缓存块，每个缓存块能储存 2KB 字节数据，并为每个缓存块分配一个 ID 号进行标识。同时，使用另外一块缓存区对所有缓存块的 ID 号进行统一管理，在初始化的时候，需要将 512 个缓存块的 ID 号写入到空闲地址缓存区，并预先依次为每个输入端口取出 ID 号供后续进入的报文使用。

（3）端/交换输出处理模块根据各自的输出调度，将报文从报文集中缓存模块读取出来，并由主机/网络接口发送到主机/网络。

报文描述符部分处理流程：

（1）端/交换输入处理模块根据从主机/网络接口接收到报文信息，以及报文集中缓存模块给的缓存块 ID 号（pkt_bufid），构造报文描述符并发送给查表转发模块。

（2）查表转发模块根据报文描述符中的 flow_id 进行查表，得到输出端口号，并将报文描述符中的 pkt_type 和 pkt_bufid 信息发送给对应的输出端口处理逻辑。当得到的输出端口多位高，意味处理的描述符代表的报文是个多播报文，需要将 pkt_bufid 和 pkt_type 往多个端口进行转发，同时将 pkt_bufid 做为写地址，输出端口的数量为写数据，写入到报文集中缓存模块进行组播计数的缓存。

（3）端/交换输出处理模块接收到 pkt_type 后对报文进行识别，将不同类型的 pkt_bufid 写入到对应的队列进行缓存。各端口的输出处理模块会经过调度，将对应的pkt_bufid读取出来并根据此pkt_bufid从报文集中缓存模块进行报文数据的读取。

下面，以网络接口接收、网络接口发送；主机接口接收、网络接口发送；网络接口接收、主机接口接收，这三种数据流进行芯片主要处理流程的介绍。

### 1.4.1 网络接口接收、网络接口发送

芯片中传输的报文可分为数据、同步与控制三种报文，下面从这三种报文的处理上进行介绍。

（1）数据报文的处理

报文由 p0-p7 接口进入芯片，经过网络接收模块完成 gmii 时钟域到芯片内部处理时钟域的转换，并记录接收时间戳、构造 metadata 与报文数据并行往下级模块发送；在帧完整性检测模块进行长报文与短报文的处理；再由帧解析模块对帧进行解析，并提取出 flow_id、pkt_type、lookup_en、outport、submit_addr 信息与报文缓存模块分配的 pkt_id 信息一起构造成报文描述符，将描述符发送给查表模块进行查表；而报文数据需要与 pkt_bufid、metadata 再一起发送给接口输入缓存模块，在接口输入缓存模块中将 8bit 的数据写入到 134bit 的寄存器中，将 metadata 在报文头之前作为缓存数据处理。当一个寄存器写满了 134bit 便将寄存器中的数据写入到报文缓存区，由于报文缓存区是分时复用的方式进行数据的写入，因此本模块需要等到报文集中

缓存模块返回一个确认信号之后才能进行下一次的数据写入。当其中一个寄存器写满，且还未收到报文集中缓存模块返回的确认信号时，后续进行的报文就写入到另一个寄存器，写满再发送给报文集中缓存模块进行报文的缓存，以此轮询，直到报文全部写入完成。

报文描述符数据由帧解析模块构造后发送给转发查表模块处理，转发查表模块会根据报文描述符的 lookup_en 信号判断该报文是否需要查表转发，若不需要查表，则直接将报文描述符中的 outport 信息作为转发的输出端口，将 pkt_bufid 与 pkt_type 信息发送给对应输出端口逻辑；若需要查表，则从报文描述符中提取出 flow_id 进行查找转发表，得到输出端口号，并将 pkt_bufid 与 pkt_type 信息发送给对应输出端口逻辑。输出端口逻辑中的网络入队控制模块接收到 pkt_bufid 与 pkt_type 后，根据队列门控模块的门控信息与 pkt_type 进行缓存队列的选择，并将选择好的 queue_id 与 pkt_bufid 一起发送给网络队列管理模块进行缓存。网络输出调度模块根据队列门控模块的门控信息以及每个队列的状态信息计算出最优先调度的 queue_id，并将此 queue_id 中的第一个 pkt_bufid 缓存的地址发送给网络队列管理模块，等待队列缓存仲裁模块将对应的队列中的数据读取出来，将第一个 pkt_bufid 的地址作为 pkt_bufid 发送给网络输出模块，并根据读取出来的数据更新此 queue_id 中第一个 pkt_bufid 所缓存的地址。网络输出调度模块接收到 pkt_bufid 之后发送给网络发送模块，网络发送模块根据 pkt_bufid 往报文集中缓存模块进行报文的提取。网络发送模块内部维持两个寄存器，依次将 134bit 的数据转换成 8bit 的数

据往后发送，最后往 gmii 接口输出；当一个寄存器读空之后再往报文集中缓存模块进行下一个 134bit 数据的读取，直到报文数据全部读取完成。另外，在报文发送的时候需要将 metadata 去除。

（2）同步报文的处理

同步报文会在网络发送模块进行识别，完成透明时钟的更新，其他的处理与数据报文的处理一样，因此不再过多赘述。

（3）控制报文的处理

控制报文会映射成 BE 数据报文进行处理。

1.4.2 主机接口接收、网络接口发送

（1）数据报文的处理

主机下发的数据报文可分为 TS、BE、RC 三种，由于 BE 与 RC 报文的处理一样，因此可分成 TS 报文的处理、非 TS 报文的处理两种情况进行介绍。

1、TS 报文的处理

主机下发的 TS 报文，经过主机接收模块实现跨时钟域后发送给报文映射与分派模块；在报文映射与分派模块将报文数据与 1.4.1 中的数据处理一样，从报文缓存区得到 pkt_bufid，并将 pkt_bufid 作为基地址，把报文数据存放到报文缓存区。将 pkt_bufid 信息与报文 flow_id、pkt_type、lookup_en、outport、submit_addr 信息组合构造成报文描述符，发送给 TS 注入管理模块。TS 注入管理模块将报文描述符进行缓存，并等待 TS 注入调度模块经过查找注入时刻表来判断当前时间是否有 TS 流需要被注入。若有，TS 注入调度模块则将需要调

度信息发送给 TS 注入管理模块进行描述符的调度，发送给多平面控制模块；若没有则不做任何处理。多平面控制模块判断该报文是否需要查表（暂不实现），并对描述符中的 lookup_en 信号进行使能或不使能，若不使能，还需为报文规划一个输出端口号，处理完成后发送给转发查表模块。之后的处理与 1.4.1 中数据报文的处理一样，不再过多赘述。

2、非 TS 报文的处理

主机下发的非 TS 报文处理，与 TS 报文的处理大致相同，从报文缓存区得到 pkt_bufid，并将 pkt_bufid 作为基地址，把报文数据存放到报文缓存区，构造报文描述符。不同的地方是将报文描述符直接发送给多平面控制模块，经过多平面控制模块之后发送给转发查表模块进行接下来的处理。

（2）同步报文的处理

主机下发的同步报文，需要在主机接收模块进行时间同步主时钟发送时间戳的记录，并写入到同步报文体内。之后在流水线中与非 TS 流的处理流程一样，因此不再过多赘述。

（3）控制报文的处理

主机下发的控制报文，在报文映射与分派模块进行识别出来并发送给配置与状态管理模块；在配置与状态管理模块内进行控制报文的解析，并构造命令数据对其他模块进行寄存器的控制。

1.4.3 网络接口接收、主机接口发送

（1）数据报文的处理

需要往主机发送的数据报文可分为 TS、BE、RC 三种，由于 BE 与 RC 报文的处理一样，因此可分成 TS 报文的处理、非 TS 报文的处理两种情况进行介绍。

1、TS 报文的处理

网络接口接收到 TS 报文之后，与 1.4.1 中的数据报文处理流程一样，从报文缓存区得到 pkt_bufid，并将 pkt_bufid 作为基地址，把报文数据存放到报文缓存区，构造报文描述符并进行转发查表。不同的是在查表之后，得到的输出端口是往主机发送的，因此将 pkt_bufid 与 pkt_type 信息发送给对应的主机输出端口逻辑；主机输出端口逻辑的主机入队控制模块会将 ts 的 pkt_bufid 发送给 TS 提交管理模块，TS 提交管理模块将 pkt_bufid 进行缓存，并等待 TS 提交调度模块经过查找提交时刻表来判断当前时间是否有 TS 流需要被提交，若有，TS 提交调度模块则将需要调度信息发送给 TS 提交管理模块进行 pkt_bufid 的调度。主机输出调度模块收到调度后的 ts 报文的 pkt_bufid 则直接发送给主机发送模块；主机发送模块根据 pkt_bufid 往报文集中缓存模块进行报文的提取。主机发送模块内部维持两个寄存器，依次将 134bit 的数据转换成 8bit 的数据往后发送，最后往 gmii 接口输出；当一个寄存器读空之后再往报文集中缓存模块进行下一个 134bit 数据的读取，直到报文数据全部读取完成。另外，从报文缓存区读出的报文 metadata 也一样从 gmii 接口发送给主机。

2、非 TS 报文的处理

需要往主机发送的非 TS 报文的处理，与 TS 报文的处理大致相同。

从报文缓存区得到 pkt_bufid，并将 pkt_bufid 作为基地址，把报文数据存放到报文缓存区，构造报文描述符并进行转发查表，查表后将 pkt_bufid 与 pkt_type 信息发送给对应的主机输出端口逻辑。不同的是，主机输出端口逻辑的主机入队控制模块会将 ts 的 pkt_bufid 发送给主机队列管理模块进行缓存；主机输出调度模块需要在没有 TS 报文的提交调度时才能去调度非 TS 报文的 pkt_bufid，从主机队列管理模块得到 pkt_bufid 后发送给主机发送模块进行处理。

（2）同步报文的处理

网络接口接收到同步报文后与本节 1.4.3 中的“非 TS 报文的处理”基本一样，唯一的不同是在主机发送模块需要将发送时间戳信息写入到 metadata 中，并在 metadata 相应字段记录同步报文的从时钟接收时间戳信息。

（3）控制报文的处理

往主机接收发送的控制报文都是由芯片内部的配置与状态管理模块构造的 NMAC 报文，配置与状态管理模块从报文缓存区得到 pkt_bufid，并将 pkt_bufid 作为基地址，把报文数据存放到报文缓存区，构造报文描述符后进行转发查表，而后与 1.4.3 节中的非 TS 数据报文的处理流程一样由 gmii 接口发送给主机。

### 1.4.4 长报文的处理流程

**长报文发送端的处理：**发送端从网络接口接收到一个长报文，在进入网络接收模块之前保证每个报文都不大于 1518B。且该发送端发送的所有报文都是未经过分类映射的报文。

网络接收模块将报文从外部时钟域切换到芯片内部时钟域的处理；

帧分片模块根据报文的长度域字段值将报文切分成：第一片不大于 128B 的报文，而之后的每一片不小于 64B 且不大于 122B（为主机预留 6B 位置填充 TSNTag）；

帧解析模块根据报文信息构造描述符（由于报文未经过分类映射，默认送主机），由于第二片及之后的分片报文都不携带以太网头，因此需要沿用第一片的报文描述符，只修改其中的 pkt_bufid 即可；

输入缓存接口模块将报文写入报文集中缓存模块，在处理第二片及之后的分片报文时，需要将预留的最高 6B 填 0。

芯片内部其他模块将此报文当正常报文处理，并发送到主机；

主机完成分类映射以及 TSNTag 添加后发送到芯片；

芯片从主机口接收到分类映射后的报文之后正常地处理以及转发即可。

**长报文接收端的处理：**

网络接收模块接收到分片后的报文，经过转发查表之后送往主机；

主机将分片后的第一片报文中的 DMAC 还原，其他分片报文中的 TSNTag 丢弃，并在 metadata 中标识这一条流，进入芯片内部一条固定的队列；

主机接收模块接收到主机下发的分片报文后，根据 metadata 构造描述符，转发查表模块只转发不查表，发往对应的输出端口；

网络发送处理逻辑将所有分片后的报文描述符都写入到对应的队列中，在完成所有的分片写入后才能开始调度；

网络输出调度模块根据门控进行队列调度，但当开始调度上述缓存了分片后的报文描述队列时，之后的调度需要优先调度此队列，直到调度出分片后报文的最后一片才能开始其他队列的调度。

网络发送模块需要将所有分片后的报文数据从报文集中缓存模块读出，并连续地从网口输出。

## 1.5 模块间信号定义

### 1.5.1 信号说明

| 信号 | 位宽 | 含义 |
|---|---|---|
| pkt（报文集中缓存接收或发送的信号） | 134 | 报文体数据，具体格式参考 1.5.2 |
| pkt（非报文集中缓存模块的信号） | 9 | 报文体数据，具体格式参考 1.5.2 |
| ctrl_pkt | 9 | 控制报文体数据 |
| descriptor | 35 | 报文描述符数据，用于标识报文数据 |
| ts_descriptor | 35 | 主机下发的 TS 报文描述符数据 |
| rc&be_descriptor | 35 | 主机下发的 RC 和 BE 报文描述符数据 |
| pkt_bufid | 9 | 报文数据在报文缓存区中缓存的 ID 号 |
| ts_pkt_bufid | 9 | ts 报文数据在报文缓存区中缓存的 ID 号 |
| rc&be_pkt_bufid | 9 | rc&be 报文数据在报文缓存区中缓存的 ID 号 |
| pkt_type | 3 | 报文类型数据 |
| flow_id | 14 | 报文数据的流 ID，用于标识流 |
| lookup_en | 1 | 查表使能 |
| outport | 9 | 输出端口号（bitmap） |

| | | |
|---|---|---|
| inport | 4 | 输入端口号 |
| queue_id | 3 | queue_id 号，用于标识队列缓存的 ID 号 |
| frag_last | 1 | 分片后的最后一拍标识 |
| frag_en | 1 | 分片使能信号 |
| regroup_en | 1 | 重组使能信号 |
| multicast_ outport_ count | 4 | 组播报文的输出端口数量 |
| ts_inject_id | 5 | ts 描述符在注入调度时存放的 ID 号 |
| ts_submit_id | 5 | Ts 的 pkt_bufid 在提交调度时存放的 ID 号 |
| gate_ctrl_vector | 1 | 门控信号 |
| time_rst_pulse | 1 | 接口逻辑中时间计数器的复位脉冲 |
| syned_global_time | 48 | 同步过的芯片全局时间信息 |

### 1.5.2 数据传输格式

1）pkt_data 数据传输格式

在模块之间传输的有两种格式，一种位宽为 9bit；另一种位宽为 134bit。具体区别如下：

pkt_data 位宽为 9 位，包含 1bit 头尾标志、8bit 报文数据。头尾标志： 1 表示报文头/尾数据；0 标识报文体中间数据。具体如下图所示：

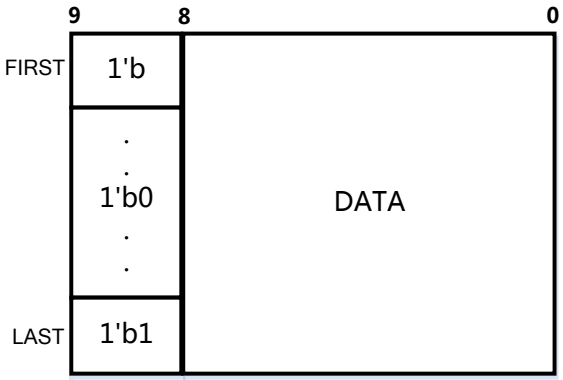


图2-1 Pkt_data 格式定义图

pkt_data 位宽为 134 位，包含 2bit 头尾标志、4bit 无效字节、128bit 报文数据。头尾标志： 01 表示报文头；11 表示报文体中间数据；10 表示报文尾。无效字节用于标识报文尾中无效的字节数。具体如下图所示：

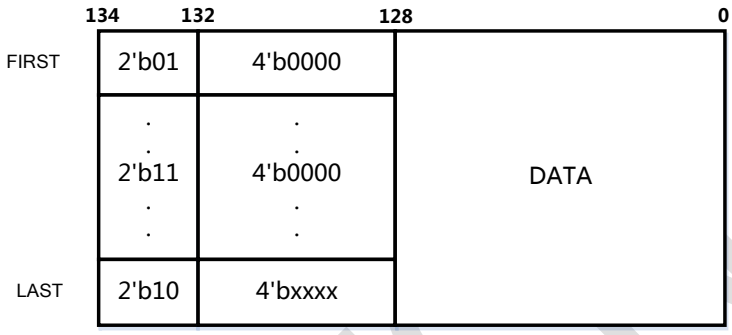


图2-2 Pkt_data 格式定义图

2）metadata 数据格式

在主机与芯片之间传输的 pkt_data 数据携带 Metadata（MD），MD 共有 64bit，随报文一起在流水线中传输，包含报文当前处理状态的数据结构。MD 处在 PKT 报文的前 8 字节。具体 MD 的格式可参考附录 B。

# 附录 A TSNTag 格式

在发送端系统内部需要根据报文七元组（目的 mac、type、IP 五元组）对时间敏感、带宽预约、尽力转发流量进行分类映射。将分类映射的结果与原报文的 DMAC 字段进行替换，以此进行网络的交换，直到接收端系统内部进行 DMAC 还原。被替换的 DMAC 字段被定义成 TSNTag。

1. 分类映射关键字 Key

| 位宽 | 名称 | 描述 |
|---|---|---|
| 48 | DMAC | 报文目的 MAC |
| 16 | ETHTYPE | 报文以太网类型 |
| 8 | protocol | 报文协议类型 |
| 32 | Sip | 报文源 ip |
| 32 | Dip | 报文目的 ip |
| 16 | Sport | 报文源端口 |
| 16 | Dport | 报文目的端口 |

2. 分类映射结果 ACTION(TSNTag)

由于芯片内部不再携带 metadata，因此时间同步报文的芯片接收时间戳需要通过其他方式在芯片内部传输，目前的方案是在 TSNTag 中携带，将 TSNTag 的格式调整如下：

由于时间同步报文的 TSNTag 中“seq_id”、“frag_flag”、“frag_id”、“inject_addr”、“submit_addr”信息是无用的，因此可以将时间同步报文的这些字段用来存放芯片的接收时间戳信息。而其他非时间同步报文的芯片接收时间戳信息是无用的，因此可以延用这些字段的信息。

讨论之后对 TSNTag 进行了修改，时间同步报文和非时间同步报文的 TSNTag 高 17 位一样，分别为 3bits 的“flow_type”和 14bits 的 flow_id；而低 31 位是不同的，时间同步报文的低 31 位用来存放芯片接收时间戳信息，非时间同步报文的低 31 位用来存放“seq_id”、“frag_flag”、“frag_id”、“inject_addr”、“submit_addr”信息。修改之后的 TSNTag 格式如下：

表 A-1 时间同步报文的 TSNtag

| 位宽 | 名称 | 位置 | 描述 |
|---|---|---|---|
| 3 | Flow type | [47:45] | 流类型。 100：同步报文 （其他报文的格式如下表） |
| 14 | Flow id/IMAC | [44:31] | 静态流量使用 flowID，每条静态流分配一个唯一 flowID，动态流使用 imac 地址，imac 地址相同的则在交换节点命中同一条表项。 |
| 12 | Reserve | [30:19] | 保留 |
| 19 | Rx_timestamps | [18:0] | 芯片接收到时间同步报文的本地时间信息，用于芯片发送报文时计算透明时钟。 |

表 A-2 非时间同步报文的 TSNTag

| 位宽 | 名称 | 位置 | 描述 |
|---|---|---|---|
| 3 | Flow type | [47:45] | 流类型。000:TS 分组 001:TS 分组 010: TS 分组<br>011：RC 分组 101：NMAC 分组 110：BE 分组<br>111：需重组分组（时间同步报文的格式如上表） |
| 14 | Flow id/IMAC | [44:31] | 静态流量使用 flowID，每条静态流分配一个唯一 flowID，动态流使用 imac 地址，imac 地址相同的则在交换节点命中同一条表项。 |
| 16 | Seq id | [30:15] | 用于标识每条流中报文的序列号 |
| 1 | Frag flag | [14] | 用于标识分片后的尾。0：分片后的中间报文<br>1：尾拍 |
| 4 | Frag ID | [13:10] | 用于表示当前分片报文在原报文中的分片序列号 |
| 5 | inject addr | [9:5] | TS 流在源端等待发送调度时缓存地址 |
| 5 | submit addr | [4:0] | TS 流在终端等待接收调度时缓存地址 |

# 附录 B metadata 格式

芯片内部处理的报文数据都不携带 metadata，只在主机与芯片之间携带 metadata。主机发送一个报文时，会在报文头携带一个 64B 的 metadata。主机接收处理逻辑在主机接口接收到报文后，将 metadata 中的有效数据提出并将 metadata 丢弃掉。而主机发送处理逻辑在主机接口发送报文前，将 64B 的 metadata 添加到报文头。

主机下发报文时，携带的 metadata 格式如下：

| 位置 | 位宽 | 名称 | 含义 | 备注 |
|---|---|---|---|---|
| [63:61] | 3 | pkttype | 报文的类型。 | 000:TS 分组 001:TS 分组 010: TS 分组 011：RC 分组 100：PTP 分组 101：NMAC 分组 110：BE 分组 111：需重组分组 |
| [60:56] | 5 | inject_addr | TS 注入缓存地址 | / |
| [55:47] | 9 | outport | 报文的输出端口号 | / |
| [46] | 1 | lookup_en | 查表使能信号 | / |
| [45] | 1 | frag_last | 重组报文的最后一个片 | 与 pkttype 配合使用，报文类型为“需重组分组”时，frag_last 才是有效的。默认为 1 |
| [44:0] | 45 | reserve | 保留 | / |

芯片上传报文时，携带的 metadata 格式如下：

| 位置 | 位宽 | 名称 | 含义 | 备注 |
|---|---|---|---|---|
| [63:16] | 48 | ts | 同步报文全局时间信息 | 用于记录时间同步报文在主时钟设备接收到 req 报文的全局时间信息，以及从时钟设备接收到 sync 报文的全局时间信息 |
| [15:12] | 4 | inport | 报文的输入端口号 | / |
| [11:0] | 12 | reserve | 保留 | / |

# 附录 C 芯片管脚

芯片管脚列表如下：

| 名称 | 位宽 | 方向 | 说明 | 默认值 |
|---|---|---|---|---|
| 时钟复位接口，4pin | | | | |
| OSC_CLK | 1 | Input | 外部晶振信号输入，晶振频率为 125MHz | 1’b0 |
| HARD_RST_N | 1 | Input | 上电复位管脚 | 1’b0 |
| BUTTON_RST_N | 1 | Input | 按钮复位管脚 | 1’b0 |
| ET_RESETC_ChipReset | 1 | Input | 外部芯片复位管脚 | 1’b0 |
| GMII 接口，共 9 个接口，198pin | | | | |
| GMII_RXC | 1 | Input | 输入时钟 | 1’b0 |
| GMII_RXD | 8 | Input | 输入数据 | 8’b0 |
| GMII_RXEN | 1 | Input | 输入使能 | 1’b0 |
| GMII_RXER | 1 | Input | 输入错误 | 1’b0 |
| GMII_GTX | 1 | Output | 千兆输出时钟 | 1’b0 |
| GMII_TXD | 8 | Output | 输出数据 | 1’b0 |
| GMII_TXEN | 1 | Output | 输出使能 | 1’b0 |

| | | | | |
|---|---|---|---|---|
| GMII_TXER | 1 | Output | 输出错误 | 1’b0 |
| 脉冲,1pin | | | | |
| pulse_ms | 1 | Output | 毫秒脉冲 | 1’b0 |
| 总计：203pin | | | | |

# 附录 D 寄存器说明

TSN 芯片地址空间主要有两部分，包括：MDID 模块号和真实地址空间，其中 MDID 模块号主要用来区分不同模块，而后 20 位为各个模块使用的地址空间。地址的第 19bit 位用于区别地址类型，控制/表项寄存器可读可写，调试和版本寄存器只读，每个模块的地址空间为 1024k,其中可读可写和只读寄存器各有 512k。具体地址含义如下：

表 D-1 地址格式

| ADDR[26:0] | | |
|---|---|---|
| MDID[26:20] | ADDR[19] | ADDR[18:0] |
| MDID ： 0-127 | 0 | 该模块的控制寄存器,表项等,可读可写 |
| | 1 | 只读 |

每个处理模块的 MDID 号分配如下：

表 D-2 模块中的 MDID 和地址

| 处理模块 | CSM | TIS | TSS | QGC | GTS | FLT |
|---|---|---|---|---|---|---|
| MDID | 0x0 | 0x1 | 0x2 | 0x3-0xa | 0xb | 0xc |
| 地址 | 0x0-0xfffff | 0x100000-0x1fffff | 0x200000-0x2fffff | 0x300000-0xafffff | 0xb00000-0xbfffff | 0xc00000-0xcfffff |

- CSM 模块

地址范围为 Addr 0x0-0xffff

表 D-3 CSM 模块寄存器

| 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 | rc_regulation_value | | | |
| 0xc | be_regulation_value | | | |
| 0xd | unmap_regulation_value | | | |
| 0xe ~ 0xfffff | reserve | | | |

表 D-4 寄存器的具体含义

| name | bit | R/W | description | default |
|---|---|---|---|---|
| offset_l | 31:17 | R/W | 代表时间偏移的高位值的低 15 位，表示毫秒 | 0 |
| | 16:0 | R/W | 时间偏移的低位，表示拍数 | 0 |
| offset_h | 31:17 | R/W | 保留位 | 0 |

| | | | | |
|---|---|---|---|---|
| | 16 | R/W | 代表时间偏移的正负值，1 代表正值，如果为 0，则代表负值 | 0 |
| | 15:0 | R/W | 代表时间偏移的高位值的高 16 位，表示毫秒 | 0 |
| time_slot | 31:11 | R/W | 保留 | |
| | 10:0 | R/W | 时间槽大小 | 0 |
| cfg_finish | 31:2 | R/W | 保留 | 0 |
| | 1:0 | R/W | 配置完成寄存器，<br>0 代表芯片正在初始化，不接收任何报文，<br>1 代表初始化完成，可以接收 NMAC 配置报文<br>2 代表配置完成，可以接收除 TS 报文的任何报文<br>3 代表可以接收任何报文 | 0 |
| port_type | 31:8 | R/W | 保留 | 0 |
| | 7:0 | R/W | 网络端口类型寄存器，芯片共有 8 个网络端口，寄存器的 0-7 位分别代表 0-7 端口的类型，1 代表处理经过映射后的报文，0 代表处理未经映射的报文 | 0 |
| qbv_or_ach | 31:1 | R/W | 保留 | 0 |
| | 0 | R/W | 调度模式选择信号，网络输出逻辑中的调度机制是 QBV 模式还是 QCH 模式<br>0 代表 QBV 模式；1 代表 QCH 模式 | 0 |

| | | | | |
|---|---|---|---|---|
| report_type | 31:16 | R/W | 保留 | 0 |
| | 15:0 | R/W | 上报类型，具体参考附录 E | 0 |
| report_en | 31:1 | R/W | 保留 | 0 |
| | 0 | R/W | 上报使能信号，配置与状态管理模块是否进行周期性上报<br>0 代表不上报；1 代表上报 | 0 |
| inject_slot_period | 31:12 | R/W | 保留 | 0 |
| | 10:0 | R/W | 注入时间槽周期，芯片内部时间槽切换的周期值<br>配置的值范围：1-1024 个 | 0 |
| submit_slot_period | 31:12 | R/W | 保留 | 0 |
| | 10:0 | R/W | 提交时间槽周期，芯片内部时间槽切换的周期值<br>配置的值范围：1-1024 个 | 0 |
| report_period | 31:12 | R/W | 保留 | 0 |
| | 11:0 | R/W | 上报周期，配置与状态管理模块上报的周期值<br>配置的值范围：1（ms）或 1000（ms） | 0 |
| rc_regulation_value | 31:9 | R/W | 保留 | 0 |
| | 8:0 | R/W | RC 流的监管阈值，当 BUFID 的剩余个数小于该值，开始丢弃 RC 报文 | |
| be_regulation_value | 31:9 | R/W | 保留 | 0 |
| | 8:0 | R/W | BE 流的监管阈值，当 BUFID 的剩余个数小于该值，开始丢弃 BE 报文和 RC 报文 | |
| unmap_regulation_value | 31:9 | R/W | 保留 | 0 |

| | | | | |
|---|---|---|---|---|
| | 8:0 | R/W | 非映射流的监管阈值，当 BUFID 的剩余个数小于该值，开始丢弃非映射报文 | |
| reserve | 31:9 | R/W | 保留 | 0 |

- TIS 模块

地址范围为 Addr 0x100000-0x1fffff

表 D-5 地址格式

| Addr | Data | | | |
|---|---|---|---|---|
| | [31:24] | [23:16] | [15:8] | [7:0] |
| send_table_N 0x100000-0x1003ff | TS 报文发送时刻表每项内容，N=0、1、…、1023 send_table_0 表示第 0 个发送表 | | | |
| 0x100400-0x1fffff | 保留 | | | |

表 D-6 寄存器的具体含义

| name | bit | R/W | description | default |
|---|---|---|---|---|
| send_table_0 | 16 | R/W | 保留 | 0 |
| | 15 | R/W | 表项有效位，0 代表无效，1 代表有效 | 0 |
| | 14:5 | R/W | TS 流在一个应用周期内的注入时间槽 | 0 |
| | 4:0 | R/W | TSNTag 中的“send addr” | 0 |
| …… | | | | |
| send_table_1023 | 16 | R/W | 保留 | 0 |
| | 15 | R/W | 表项有效位，0 代表无效，1 代表有效 | 0 |
| | 14:5 | R/W | TS 流在一个应用周期内的注入时间槽 | 0 |
| | 4:0 | R/W | TSNTag 中的“send addr” | 0 |
| 0x100400-0x1fffff | | | 保留 | |

- TSS 模块

地址范围为 Addr 0x200000-0x2fffff

表 D-7 地址格式

| Addr | Data | | | |
|---|---|---|---|---|
| | [31:24] | [23:16] | [15:8] | [7:0] |
| submit_table_N<br>0x200000-0x2003ff | TS 报文提交时刻表每项内容，N=0、1、…、1023<br>submit_table_0 表示第 0 个提交表 | | | |
| 0x200400-0x2fffff | 保留 | | | |

表 D-8 寄存器的具体含义

| name | bit | R/W | description | default |
|---|---|---|---|---|
| submit_table_0 | 31:16 | R/W | 保留 | 0 |
| | 15 | R/W | 表项有效位，0 代表无效，1 代表有效 | 0 |
| | 14:5 | R/W | TS 流的提交时间槽 | 0 |
| | 4:0 | R/W | TSNTag 中的“submit addr” | 0 |
| …… | | | | |
| submit<br>_table_1023 | 31:16 | R/W | 保留 | 0 |
| | 15 | R/W | 表项有效位，0 代表无效，1 代表有效 | 0 |
| | 14:5 | R/W | 当前 Slot | 0 |
| | 4:0 | R/W | TSNTag 中的“submit addr” | 0 |

- QGC 模块

地址范围为 Addr 0x300000-0xaffffff，其中 0x300000-0x3fffff 表示第一个端口的门控表，以此类推，共有 8 个端口门控。

表 D-9 地址格式

| Addr | Data | | | |
|---|---|---|---|---|
| | [31:24] | [23:16] | [15:8] | [7:0] |
| port0_gate_table_N<br>0x300000-0x3003ff | 0 号端口的门控表，N=0、1、…、1023，输出门控<br>port0_gate_table_0 表示 0 号端口的第一个时刻的门控状 | | | |

| | 态 |
|---|---|
| port1_gate_table_N<br>0x400000-0x4003ff | 1 号端口的门控表，N=0、1、…、1023 |
| port2_gate_table_N<br>0x500000-0x5003ff | 2 号端口的门控表，N=0、1、…、1023 |
| port3_gate_table_N<br>0x600000-0x6003ff | 3 号端口的门控表，N=0、1、…、1023 |
| port4_gate_table_N<br>0x700000-0x7003ff | 4 号端口的门控表，N=0、1、…、1023 |
| port5_gate_table_N<br>0x800000-0x8003ff | 5 号端口的门控表，N=0、1、…、1023 |
| port6_gate_table_N<br>0x900000-0x9003ff | 6 号端口的门控表，N=0、1、…、1023 |
| port7_gate_table_N<br>0xa00000-0xa003ff | 7 号端口的门控表，N=0、1、…、1023 |

表 D-10 寄存器的具体含义

| name | bit | R/W | description | default |
|---|---|---|---|---|
| port0_gate_table_0 | 31:8 | R/W | 保留 | 0 |
| | 7:0 | R/W | 0-7 位分别代表 0-7 共 8 个队列的门控状态，0 代表该队列的门控关闭，1 代表开启 | 0 |
| …… | | | | |
| port7_gate_table_1023 | 31:8 | R/W | 保留 | 0 |
| | 7:0 | R/W | 0-7 位分别代表 0-7 共 8 个队列的门控状态，0 代表该队列的门控关闭，1 代表开启 | 0 |

- GTS 模块

地址范围为 Addr 0xb00000-0xbfffff

表 D-11 地址格式

| Addr | [31:24] | [23:16] | [15:8] | [7:0] |
|---|---|---|---|---|
| / | / | | | |

表 D-12 寄存器的具体含义

| name | bit | R/W | description | default |
|---|---|---|---|---|
| / | / | / | / | / |

- FLT 模块

地址范围为 Addr 0xc00000-0xcfffff

表 D-13 地址格式

| Addr | Data | | | |
|---|---|---|---|---|
| | [31:24] | [23:16] | [15:8] | [7:0] |
| 0xc00000-0xc03fff | forward_table_N，表示转发表，N=0,1,2，…16383，forward_table_0 表示第 0 个转发表 | | | |
| 0xc04000-0xcfffff | 保留 | | | |

表 D-14 寄存器的具体含义

| name | bit | R/W | description | default |
|---|---|---|---|---|
| forward_table_0 | 31:16 | R/W | 保留 | 0 |
| | 8:0 | R/W | 转发表的内容，使用 bitmap 的形式，0-8 位分别代表向 0-8 号端口，每位的值 0 代表不向该端口转发，1 代表向该端口转发 | 0 |
| …… | | | | |
| forward_table_16383 | 31:16 | R/W | 保留 | 0 |
| | 8:0 | R/W | 转发表的内容，使用 bitmap 的形式，0-8 位分别代表向 0-8 号端口，每位的值为 0 代表不向该端口转发，1 代表向该端口转发 | 0 |
| 0xc04000-0xcfffff | | | 保留 | |

# 附录 E NMAC 报文格式

以太网帧头部中，类型字段为 0x0662，表示封装为 NMAC 报文。NMAC 分为配置报文和上报报文，配置报文是从主机发送给芯片，上报报文是从芯片发送给主机；因此将主机发送的默认为配置报文，芯片发送的默认为上报报文，不在报文内部进行标识。

配置报文格式：在报文中用 count 字段（8bit）表示报文中包含的配置条目数，报文最小为 64 字节，最后不够 64 字节的报文需要补零。NMAC 命令在以太网报文中的封装如图所示，NMAC 命令的格式如附录 F 所示：

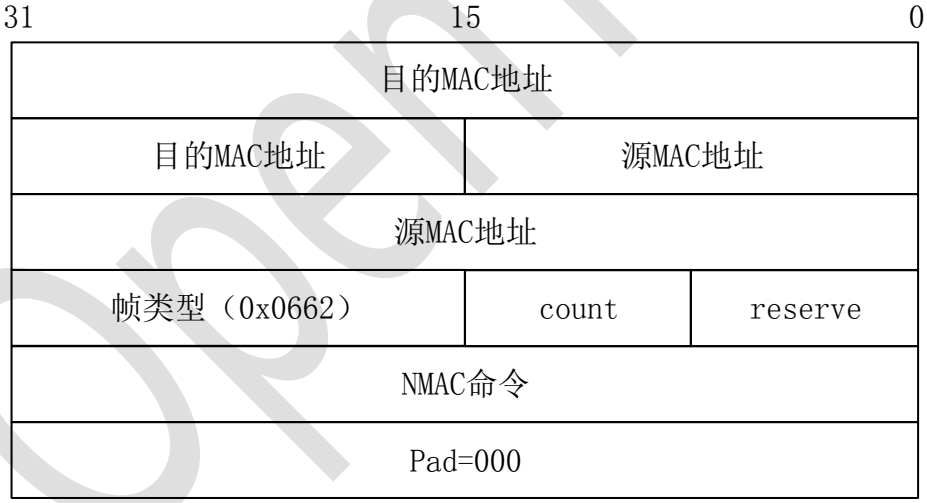


图 E-1 NMAC 配置报文格式

上报报文格式：其中报文类型为 NMAC 报文（0x1662），上报类型格式如下：

| 上报类型（16bit） | | 含义 |
|---|---|---|
| 高 6bit | 低 10bit | |
| 000000 | 0 | 配置的单个寄存器，包含 |

| | | |
|---|---|---|
| 单个寄存器 | | 配置完成寄存器、<br>端口状态寄存器、<br>时间槽大小寄存器、<br>时间偏移寄存器、<br>上报周期寄存器、<br>上报类型寄存器、<br>应用周期寄存器 |
| 000001<br>转发表 | 0 | 第 0-63 条转发表 |
| | 1 | 第 64-127 条转发表 |
| | 2-255 | 第 128-16383 条转发表 |
| 000010<br>注入时刻表 | 0 | 第 0-63 条注入时刻表 |
| | 1 | 第 64-127 条注入时刻表 |
| | 2-15 | 第 128-1023 条注入时刻表 |
| 000011<br>提交时刻表 | 0 | 第 0-63 条注入时刻表 |
| | 1 | 第 64-127 条注入时刻表 |
| | 2-15 | 第 128-1023 条注入时刻表 |
| 000100-001011<br>P0-P7 输出门控表 | 0 | 第 0-63 条注入时刻表 |
| | 1 | 第 64-127 条注入时刻表 |
| | 2-15 | 第 128-1023 条注入时刻表 |
| 001100<br>xx_state | 0 | 具体参考附录 H |

- 单个寄存器，report_type 高 6bit 为 0000，低 10bit 为 0。
- 转发表上报报文格式，report_type 高 6bit 为 000001，低 10bit 为上报的第几块，转发表一共有 16K 条，每条转发表占用 2 字节（9bit），因此每个报文可以携带 64 条，总共需要 256 个报文。
- 注入时刻表上报报文格式，report_type 高 6bit 为 000010，低 10bit

为上报的第几块，注入时刻表一共有 1024 条，每条转发表占用 2 字节（9bit），因此每个报文可以携带 64 条，总共需要 16 个报文。

- 提交时刻表上报报文格式，report_type 高 6bit 为 000011，低 10bit 为上报的第几块，注入时刻表一共有 1024 条，每条转发表占用 2 字节（9bit），因此每个报文可以携带 64 条，总共需要 16 个报文。
- 门控表按照端口划分，每个门控占用一块 RAM，共有 8 个端口。每个端口两块 RAM，总共需要 16 块 RAM，report_type 的具体格式如上表所示。
- 状态上报内容，由各个模块提供需要上报的内容，详细参考附录 H。

NMAC 上报报文的格式如下：

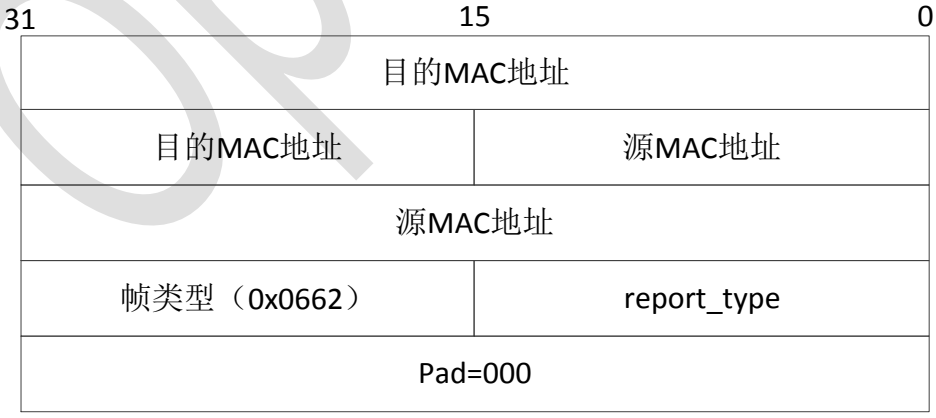


图 E-2 NMAC 上报报文格式

报文内部具体含义如下：

1）单个寄存器上报报文：

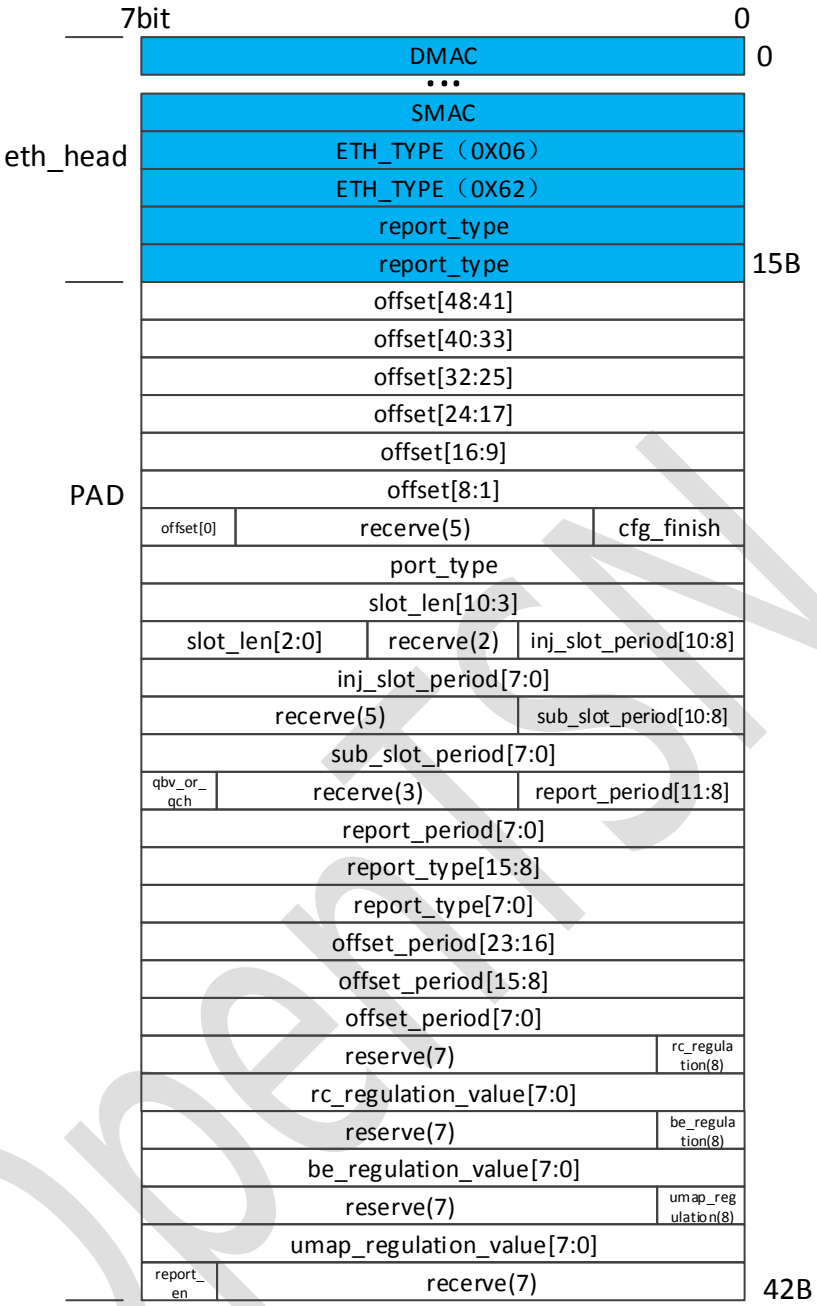


图 E-3 单个寄存器 NMAC 上报报文格式

2）各模块状态上报报文：

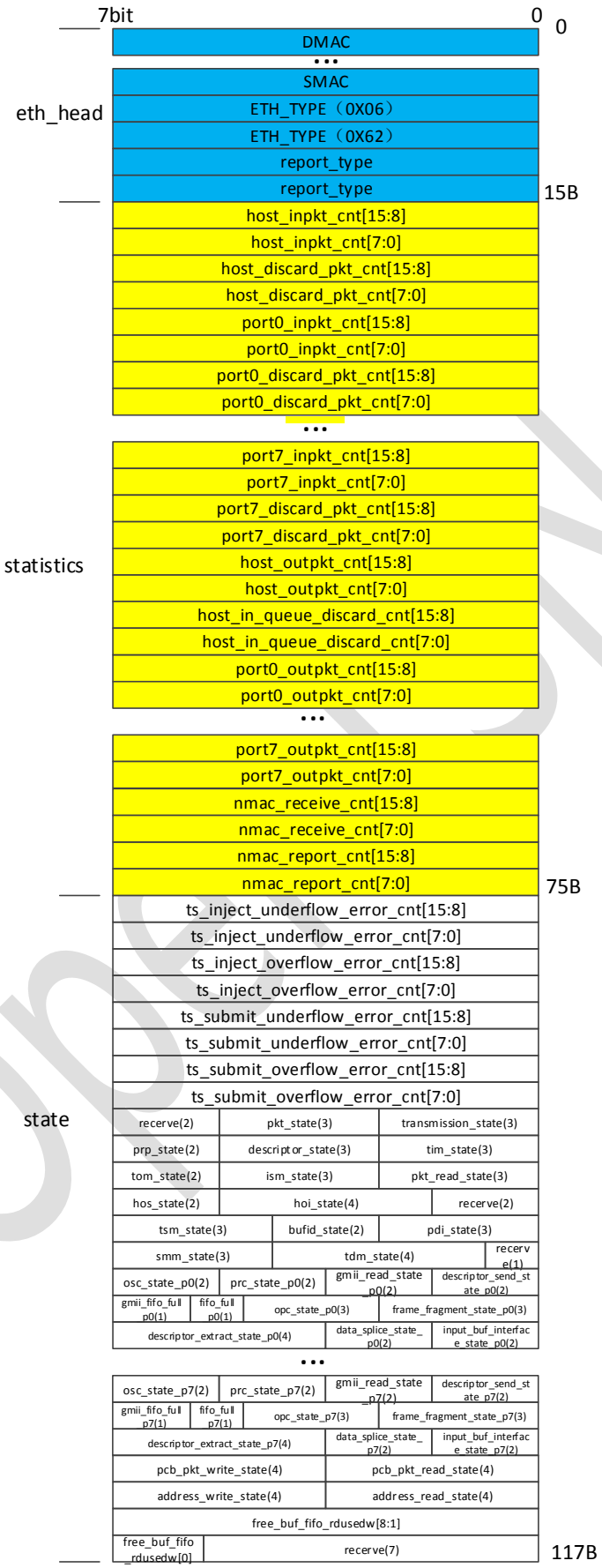


图 E-4 各模块状态 NMAC 上报报文格式

3）表项上报报文：

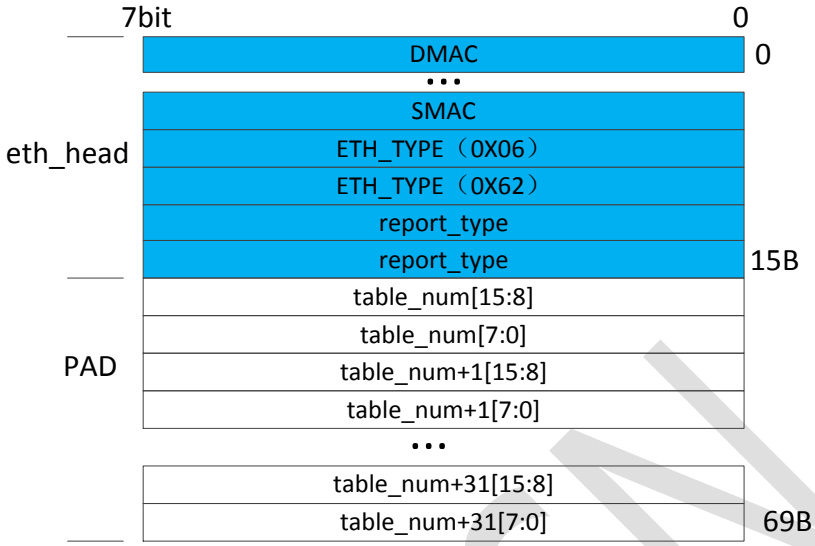


图 E-5 表项 NMAC 上报报文格式

# 附录 F NMAC 命令格式

NMAC 命令的格式如下：

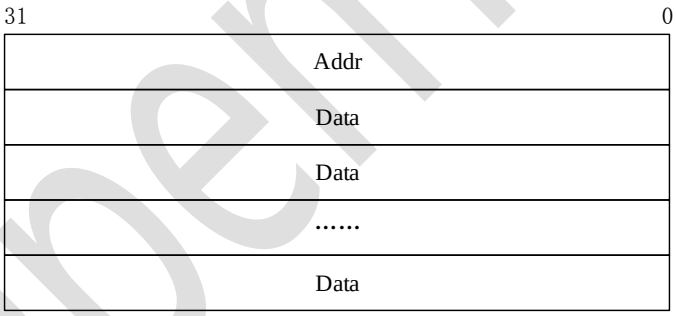


图 F-1 NMAC 命令格式

当配置的寄存器数量为 1，NMAC 命令就包括 32bit 的 ADDR 和 32bit 的 DATA；当配置的寄存器数量为 N（N>1），NMAC 命令就包括 32bitADDR 和 N*32bit 的 DATA，第一个 DATA 以 ADDR 作为 RAM 写地址，第二个以及后续 DATA 以 ADDR 循环加 1 作为 RAM 写地址。

# 附录 H NMAC 上报的状态信息

NMAC 上报的状态主要包括各个端口的收发报文计数器和各个模块的状态信息，具体如下：

表2-1 各模块上报的计数器统计

| 名称 | 位宽 | 含义 |
|---|---|---|
| host_inpkt_cnt | 16 | 主机口接收报文个数计数器 |
| host_discard_pkt_cnt | 16 | 主机口接收后丢弃的报文个数计数器 |
| port0_inpkt_cnt | 16 | 0 号网络口接收报文个数计数器 |
| port0_discard_pkt_cnt | 16 | 0 号网络口接收后丢弃的报文个数计数器 |
| port1_inpkt_cnt | 16 | 1 号网络口接收报文个数计数器 |
| port1_discard_pkt_cnt | 16 | 1 号网络口接收后丢弃的报文个数计数器 |
| port2_inpkt_cnt | 16 | 2 号网络口接收报文个数计数器 |
| port2_discard_pkt_cnt | 16 | 2 号网络口接收后丢弃的报文个数计数器 |
| port3_inpkt_cnt | 16 | 3 号网络口接收报文个数计数器 |
| port3_discard_pkt_cnt | 16 | 3 号网络口接收后丢弃的报文个数计数器 |
| port4_inpkt_cnt | 16 | 4 号网络口接收报文个数计数器 |
| port4_discard_pkt_cnt | 16 | 4 号网络口接收后丢弃的报文个数计数器 |
| port5_inpkt_cnt | 16 | 5 号网络口接收报文个数计数器 |
| port5_discard_pkt_cnt | 16 | 5 号网络口接收后丢弃的报文个数计数器 |
| port6_inpkt_cnt | 16 | 6 号网络口接收报文个数计数器 |

| | | |
|---|---|---|
| port6_discard_pkt_cnt | 16 | 6 号网络口接收后丢弃的报文个数计数器 |
| port7_inpkt_cnt | 16 | 7 号网络口接收报文个数计数器 |
| port7_discard_pkt_cnt | 16 | 7 号网络口接收后丢弃的报文个数计数器 |
| host_outpkt_cnt | 16 | 主机口发送报文个数计数器 |
| host_in_queue_discard_cnt | 16 | 主机口输入报文丢弃的数量 |
| port0_outpkt_cnt | 16 | 0 号网络口发送报文个数计数器 |
| port1_outpkt_cnt | 16 | 1 号网络口发送报文个数计数器 |
| port2_outpkt_cnt | 16 | 2 号网络口发送报文个数计数器 |
| port3_outpkt_cnt | 16 | 3 号网络口发送报文个数计数器 |
| port4_outpkt_cnt | 16 | 4 号网络口发送报文个数计数器 |
| port5_outpkt_cnt | 16 | 5 号网络口发送报文个数计数器 |
| port6_outpkt_cnt | 16 | 6 号网络口发送报文个数计数器 |
| port7_outpkt_cnt | 16 | 7 号网络口发送报文个数计数器 |
| nmac_receive_cnt | 16 | 接收 nmac 报文的数量 |
| nmac_report_cnt | 16 | nmac 上报报文的数量 |
| 总计 | / | 30 个，60B |

表2-2　各模块上报的状态统计

| 名称 | 所属模块 | 位宽 | 含义 |
|---|---|---|---|
| ts_inj_underflow_error_cnt | TIM | 16 | TS 报文注入下溢错误计数器 |
| ts_inj_overflow_error_cnt | TOM | 16 | TS 报文注入上溢错误计数器 |

| | | | |
|---|---|---|---|
| ts_sub_underflow_error_cnt | TSM | 16 | TS 报文提交下溢错误计数器 |
| ts_sub_overflow_error_cnt | HIQ | 16 | TS 报文提交上溢错误计数器 |
| pkt_state | PMD>TRW | 3 | TRW 状态机 |
| transmission_state | PMD>TRR | 3 | TRR 状态机 |
| prp_state | HRX>PRP | 2 | PRP 状态机 |
| descriptor_state | PMD>PDG | 3 | PDG 状态机 |
| tim_state | TIM | 3 | TIM 状态机 |
| tom_state | PMD>TOM | 2 | TOM 状态机 |
| iv_ism_state | TIS>ISM | 3 | ISM 注入调度状态机 |
| pkt_read_state | HTX> HRC | 3 | HRC 状态机 |
| hos_state | HOS | 2 | HOS 状态机 |
| hoi_state | HTX>HOI | 4 | HOI 状态机 |
| tsm_state | TSM | 3 | TSM 状态机 |
| bufid_state | TSM | 2 | BUFID 状态机 |
| pdi_state | HRX | 3 | PDI 状态机 |
| smm_state | TSS>SSM | 3 | SSM 提交调度状态机 |
| tdm_state | FLT>TDM | 4 | TDM 的状态机 |
| osc_state_p*N | NOS>OSC | 2 | OSC 的状态机 |

| | | | |
|---|---|---|---|
| | | | （N=0-7） |
| prc_state_p*N | NOS>PRC | 2 | PRC 的状态机（N=0-7） |
| gmii_read_state*N | NRX>GR | 2 | GW 的状态机（N=0-7） |
| opc_state_p*N | NOS>OPC | 3 | OPC 的状态机（N=0-7） |
| gmii_fifo_full*N | NRX | 1 | 输入时钟域切换 fifo 满信号（N=0-7） |
| gmii_fifo_empty*N | NRX | 1 | 输入时钟域切换 fifo 空信号（N=0-7） |
| descriptor_extract_state*N | FPM>DEM | 3 | DEM 的状态机（N=0-7） |
| frame_fragment_state*N | FFM | 2 | FFM 的状态机（N=0-7） |
| descriptor_send_state*N | FPM>DSM | 2 | DSM 的状态机（N=0-7） |
| data_splice_state*N | FPM>DSP | 2 | DSP 的状态机（N=0-7） |
| input_buf_interface_state*N | IBI | 2 | IBI 的状态机（N=0-7） |
| pkt_write_state | PCB>PW | 4 | PW 的状态机 |

| | | | |
|---|---|---|---|
| pkt_read_state | PCB>PR | 4 | PR 的状态机 |
| address_write_state | PCB>AW | 4 | AW 的状态机 |
| address_read_state | PCB>AR | 4 | AR 的状态机 |
| free_buf_fifo_rdusedw | PCB | 9 | 空闲地址管理 fifo 的个数 |
| 总计 | / | / | 29 个，160bit，20B |