OpenTSN 时钟同步设计文档

（V1.0）

一、设计目标

时间同步的总目标是实现端系统与交换机之间的亚微秒级时间同步。主从时钟通过同步报文(即 Sync，Delay_req，Delay_resq 三类报文）收集信息以实现时钟同步。同步原理如图 1-1 所示，其中t_1为 Sync 报文的发送时间戳，t_2为 Sync 报文的接收时间戳，t_3为 Delay_req 报文的发送时间戳，t_4为 Delay_req 报文的接收时间戳。

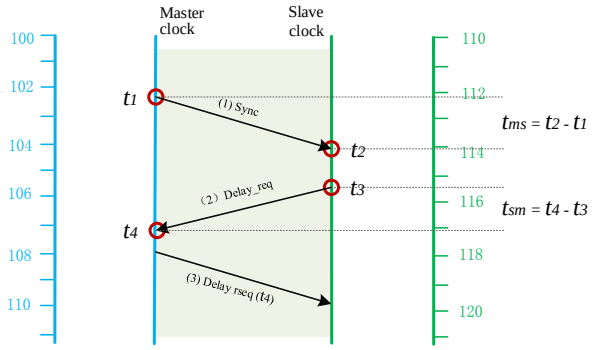


图 1-1 时钟同步原理

主从时钟偏移量计算如下公式所示：

$$t_{ms} = t_2 - t_1 = \text{delay} + \text{offset} \qquad \text{公式 1}$$

$$t_{sm} = t_4 - t_3 = \text{delay} - \text{offset} \qquad \text{公式 2}$$

$$\text{offset} = \frac{(t_{ms} - t_{sm})}{2} = \frac{(t_2 - t_1) - (t_4 - t_3)}{2} \qquad \text{公式 3}$$

实验拓扑如图 1-2 所示， Pm，Ps，SW 皆为 Openbox-s4，SW 为交换机。Pm 扮演主时钟的角色，Ps 扮演从时钟的角色，SW 扮演从时钟的角色并可以计算非本设备 PTP 报文的透明时钟。同步过程可分为 5 步：

1） 主时钟 Pm 定时发送 Sync 报文（报文中携带时间戳t_1）；

2） 交换 SW 接收到 Sync 报文后获取t_1与接收时间戳t_2，构造并返回 Delay_req 报文（获取 Delay_req 的发送时间戳t_3），并转发 Sync 报文给 Ps。Sync 为广播报文，会广播到 Ps0 和 Ps1 所在端口。

3） 从时钟 Ps 接收 Sync 报文后获取t_1与接收时间戳t_2，构造并返回 Delay_req 报文（获取 Delay_req 的发送时间戳t_3）；

4） 主时钟 Pm 接收到 Delay_req 报文后返回 Delay_resq 时，在 Delay_resq 报文中填充 Delay_req 报文的接收时间戳t_4；

5） 从时钟 Ps 和 SW 接收到 Delay_resq 后，从此报文中提取t_4，而后根据公式 3 计算主从时钟的偏移量。

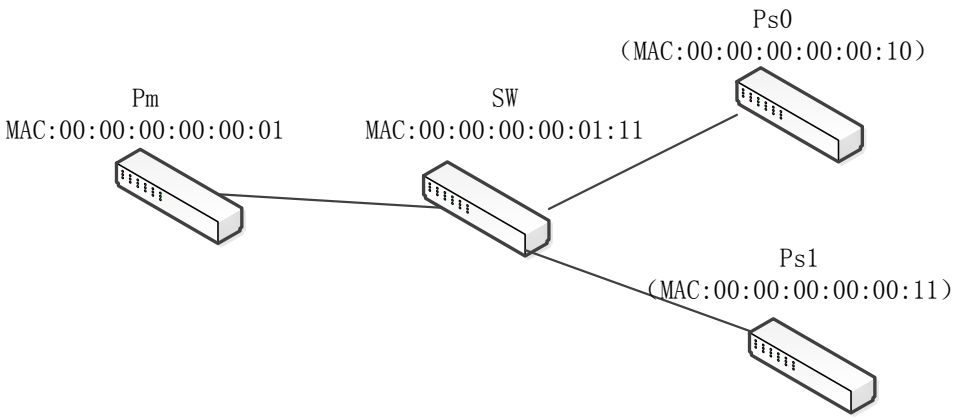


图 1-2 时钟同步实验拓扑

二、概要设计

2.1 设备内时间同步

为了实现时钟同步目标，Openbox-s4 设备内的所有接口时钟需要同步。Openbox-s4 时钟分布如 2-1 所示。其中四个输入端口的时钟分别为 Rx_clk1，Rx_clk2，Rx_clk3 与 Rx_clk4，四个输出端口的时钟分别为 Tx_clk1，Tx_clk2，Tx_clk3 与 Tx_clk4，以及 PTP 的时钟为 Core_clk。其中 Core_clk 为核心时钟，所有接口时钟统称为外围时钟。

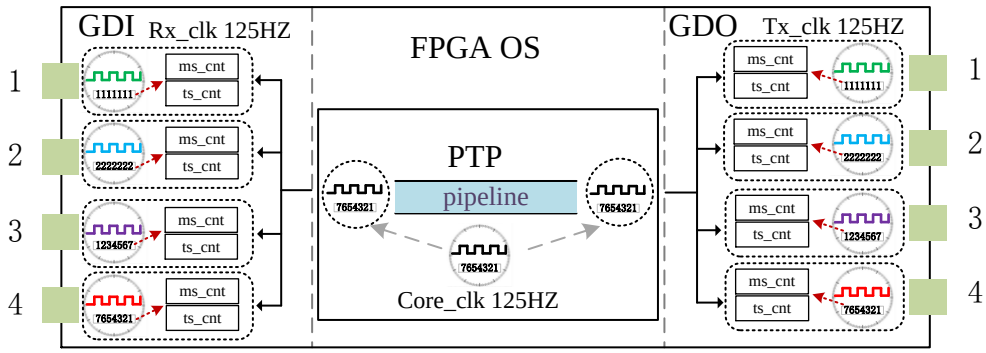


图 2-1 Openbox-S4 时钟分布

Openbox-s4 设备核心时钟与外围时钟同步设计如图 2-2 所示。

1）将 PTP 的核心时钟 Core_clk 设置为主时钟。核心时钟频率为 125MHZ，时钟周期为 8ns，并且维持一个 31 位的计数器 master_ms_cnt 与一个 17 位的计数器 master_ts_cnt。其中，master_ms_cnt 以每毫秒为时间单位，master_ts_cnt 以 8ns 为时间单位；

2) 将所有接口的外围时钟设置为从时钟。外围时钟频率统一为 125MHZ，时钟周期为 8ns，并且每个外围时钟也都维持一个 31 位的计数器 rxN_ms_cnt(txN_ms_cnt，N 为设备端口号)与一个 17 位的计数器 rxN_ts_cnt(txN_ts_cnt)。其中，rxN_ms_cnt(txN_ms_cnt)以每毫秒为时间单位，rxN_ts_cnt(txN_ts_cnt)以 8ns 为时间单位；

3）核心时钟 Core_clk 与所有外围时钟通过 temp_cnt 同步时间。

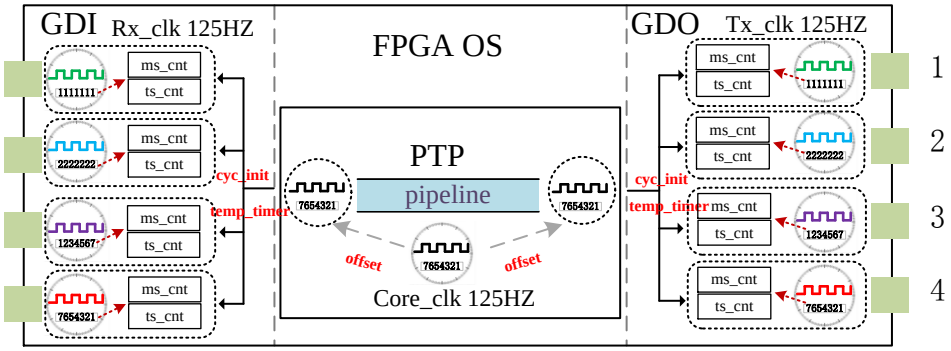


图 2-2　Openbox-S4 设备内时钟同步示意

设备内核心时钟与外围时钟同步流程如图 2-2 所示。

1) 核心时钟 Core_clk 以 1ms 为同步时间，即核心时钟的 master_ts_cnt 计数器值达到 124999 时(同步周期可以修改范围为大于等于 100us 小于 1s)。将 temp_cnt 寄存器的低 48 位赋值给{master_ms_cnt, master_ts_cnt}。同时，将 cyc_init 位置 1，送给所有的接口，并将 temp_cnt、cyc_init 保持三个时钟周期以确保数据稳定传输。

2) 外围时钟检测到 cyc_init 位为 1 后(通过采样上升沿异步处理方式实现)，将 rxN_ts_cnt(txN_ts_cnt)置为 master_ts_cnt，并将 rxN_ms_cnt(txN_ms_cnt)置为 master_ms_cnt。

2.2 设备间时间同步

当多台设备进行同步时，从设备中的 PTP 模块计算出设备间的时钟偏移量 offset 后，从设备根据 offset 值调整核心时钟 Core_clk 的两个计数器的值（master_ms_cnt 与 master_ts_cnt），以达到设备间的时钟同步。而后，从设备中的核心时钟 Core_clk 需要根据调整后的计数器 master_ms_cnt 值与 master_ts_cnt 同步从设备外围时钟的 rxN_ms_cnt(txN_ms_cnt)与 rxN_ts_cnt(txN_ts_cnt)值。具体方法如图 2-3 所示。

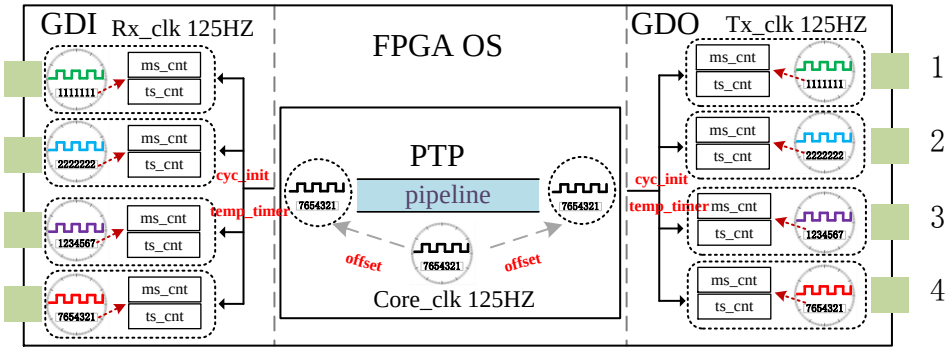


图 2-3 计数器同步示意

当核心时钟 Core_clk 中的计数器 master_ms_cnt 与 master_ts_cnt 根据偏移量 offset 同步时，将两个计数器值{master_ms_cnt, master_ts_cnt }与 offset 的差值存储在 48 位寄存器 temp_cn 中，而后，将 temp_cnt 寄存器的 48 位赋值给{master_ms_cnt, master_ts_cnt}。同时，将 cyc_init 位置 1,并保持三个时钟周期，以确保将 temp_cnt 的 48 位数据有效的赋给{rxN_ms_cnt, rx_ts_cnt}。若同步过程中，offset 的值大于 1ms，则返回同步失败。

2.3 设备间主从时间同步

如图 3-1 所示， Pm 为主时钟，Ps 为从时钟，SW 为交换机作为从时钟的同时用于传递透明时钟。如图 3-1 所示，主从时钟的整体流程如下。

1) 主时钟构造并发送 Sync 报文（携带时间戳$t1$）给从时钟；

2) 从时钟接收到 Sync 报文，获取$t1'$，$t2$与 T1($T_1 = T_{1.1} + T_{1.2} + T_{1.3}$)，并根据$t1 = t1' + T_1$更新 $t1$；

3) 从时钟发送 Delay_req 报文，并获取该报文的发送时间戳$t3$；

4) 主时钟接收到 Delay_req 报文，获取$t4'$与 T2($T_2 = T_{2.1} + T_{2.2} + T_{2.3}$)，并根据$t4 = t4' - T_2$更新 $t4$；

5) 主时钟构造并发送 Delay_resq 报文（携带更新后的$t4$）给从时钟；

6) 从时钟接收到 Delay_resq 报文，获取$t4$；

7) 从时钟根据公式 3 获取 offset，并根据该 offset 值修正从时钟片内所有时钟计数器。

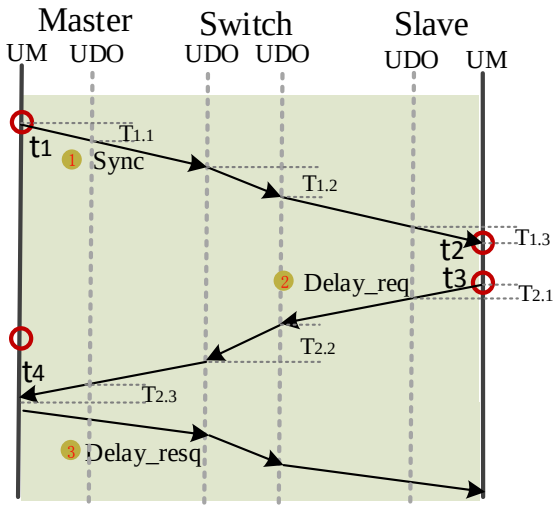


图 3-1 设备间主从时钟同步流程

各设备的工作内容如下：

主时钟（Pm）

- Pm 定时构造并发送 Sync 报文（携带$t1$）给 Ps；
- Pm 解析接收报文，若为 Delay_req 报文，获取$t4'$与 T2($T_2 = T_{2.1} + T_{2.2} + T_{2.3}$)，并根据$t4 = t4' - T_2$更新 $t4$后，给相应的对端返回 Delay_resq 报文(含更新后的$t4$)；
- 若接收的时间同步报文 DMAC 地址不为本设备 MAC 地址的则丢弃；若报文不为时间同步报文，则转发至下一功能模块。

从时钟（Ps）

- Ps 解析接收报文，若报文为 Sync 报文，则获取$t1'$，$t2$与 T1($T_1 = T_{1.1} + T_{1.2} + T_{1.3}$)，并根据

$t1 = t1' + T_1$更新 $t1$。并且，构造 Delay_req 报文返回给 Pm，期间获取 Delay_req 报文的发送时间戳$t3$；

- 若报文为 Pm 返回的 Delay_resq 报文，则从该报文中提取 Delay_req 接收时间戳$t4$，并如公式 3 计算时间差值以进行时间同步；
- 若接收的时间同步报文 DMAC 地址不为本设备 MAC 地址的则丢弃；若报文不为时间同步报文，则转发至下一功能模块。

交换机（SW）

- 交换机实现从时间的角色功能的同时在接收到非本设备的单播时间同步报文、广播时间同步报文以及非时间同步的报文不会丢弃而是转发给下一级模块。对于非本设备时间同步报文交换机则加上经过交换机所花费的时间 T（T=输出时间-输入时间）后转发报文。

三、详细设计

3.1 PTP 模块

3.1.1 PTP 功能分析

- 以 1ms 为周期（周期可调整），设备内的核心时钟与外围时钟进行一次对齐；
- 从时钟的起始时钟等于第一个 sync 报文的发送时间戳+该报文的透明时钟 T1，而后核心时钟将该值通过寄存器 temp_cnt 传给外围时钟；
- 按指定时间周期性的构造并发送 Sync 报文；
- 解析 Sync 报文，记录 Sync 报文的接收时间戳$t2$，发送时间戳$t1$以及透明时钟 T1，根据 $t1 = t1' + T_1$更新 $t1$；
- 构造 Delay_Req 报文，并记录该报文的发送时间戳$t3$；
- 解析 Delay_Req 报文，记录 Delay_Req 报文接收时间戳$t4$与 T2，根据公式$t4 = t4' - T_2$更新 $t4$；
- 按要求构造并发送 Delay_Rseq 报文；
- 解析 Delay_Resq 报文，记录更新后的时间戳$t4$；
- 通过收集的四个时间戳计算主从时钟偏移量，通过偏移量调整核心时钟的计数器。而后，根据修正后的核心时钟的计数器值修正外围时钟计数器值；

3.1.2 PTP 模块详细设计

如图 4-1 所示，整体模块结构包括 PTP、GDI 和 GDO 模块，其中 PTP 模块可分为 DMAX，MAX,Manage_CTRL，PTP_CTRL，RX_PROC，TX_PROC 与 CYC_SYNC 七个模块。

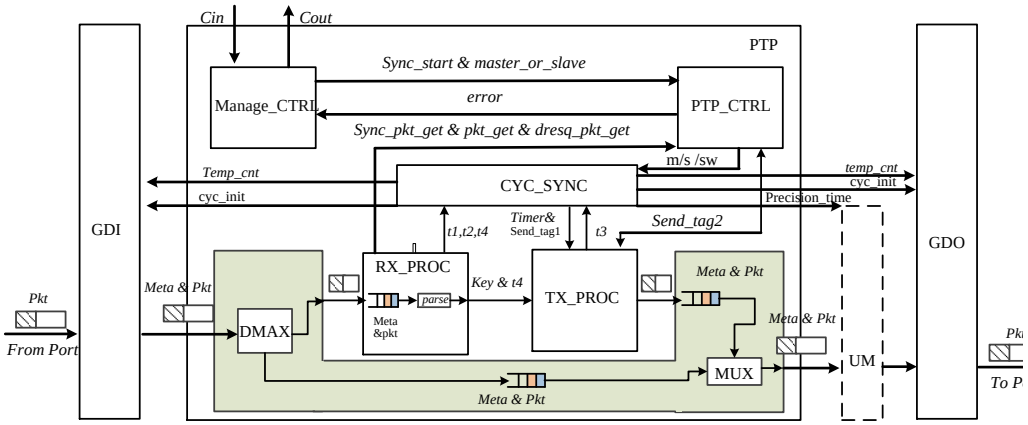


图 4-1 模块结构

各模块功能如下：

1) Manage_CTRL 模块用于配置设备角色为主端、从端及交换机以及是否开始同步；

2) PTP_CTRL 模块通知 CYC_SYNC 模块是作为主时钟或从时钟，并接收 RX_PROC 的报文通告，发送信号给 TX_PROC 模块，使其发送指定的报文输出。

3) CYC_SYNC 模块维持一个 48 位的计数器 timer_cnt，timer_cnt = {master_ms_cnt, master_ts_cnt}，每当 master_ts_cnt 计数为同步周期值时，将 cyc_init 信号置为高，将高 31 位置为 master_ms_cnt，低 17 位置为 master_ts_cnt；并将 cyc_init 信号和 timer_cnt 维持三拍有效。

   从时钟的 CYC_SYNC 模块用于接收同步的 t1、t2、t3 以 t4 值，并根据此值按公式 3 进行计算，获得 offset 值，并用 offset 值更新从时钟的 core_clk（{master_ms_cnt, master_ts_cnt}）以及 GDO、GDI 端时钟。

   主时钟 CYC_SYNC 模块的 core_clk 每间隔指定的周期通知 TX_PROC 模块发送 Sync 报文（Sync 报文中的目的地址由硬件代码指定，避免 Sync 报文与高位 cyc_init 同时到达 GDO/GDI）；

4) 从时钟的 RX_PROC 模块对 Metedata 与报文进行解析，提取 key 与记录 Sync 的发送时间戳$t1$与经过交换机的时间 T1，并根据$t1 = t1 + T_1$更新 $t1$，而后将更新后的$t1$与$t2$的使能位提交给 CYC_SYNC 模块。同时，将 key 传给 TX_PROC 模块，同时，通知 PTP_CTRL 模块已获取 Sync 报文。其中，key 字段为 54 位的源 MAC 地址与输入端口号；

   主时钟的 RX_PROC 模块对 Metedata 和报文进行解析，提取构建 Delay_resq 报文（含 Metedata）所需的关键字 key，$t4$与透明时钟 T2，根据$t4 = t4 - T_2$更新 $t4$，而后将 key 与更新后的$t4$传给 TX_PROC 模块。key 字段为 54 位的源 MAC 地址与输入端口号；

5) 从时钟的 TX_PROC 模块按照要求构造并发送 Delay_req 报文，同时将时间戳$t3$使能位置高通知 CYC_SYNC 模块记录时间戳$t3$，而后将时间戳$t3$填充在报文中；

   主时钟的 TX_PROC 模块按照要求构造 Sync 报文（含$t1$）、Delay_resq 报文（含$t4$），而后将 Sync 和 Delay_resq 报文发送给 MAX 模块；

6) DMAX 模块对根据设备角色来判断报文的转发方向。

   若设备角色为交换机并且报文以太网类型为 PTP 报文（16’h88F7），的情况下判断报文 DMAC

地址是否为本机 MAC 地址或广播地址，若是，则将报文转发给 RX_PROC 模块。否则，将报文直接转发给 MAX 模块。另外，广播的 Sync 报文在转发给 RX_PROC 模块的同时也转发给 MAX 模块一份。若 MAC 地址不匹配，则报文转发给 MAX 模块。

若设备角色为主端/从端并且报文以太网类型为 PTP 报文（16’h88F7），判断报文 DMAC 地址是否为本机 MAC 地址或广播地址，若是，则将报文转发给 RX_PROC 模块（主端不会接收 PTP 的广播报文）。若 DMAC 地址非本设备的 MAC 地址的 PTP 报文则直接丢弃，其它类型报文则转发给 MAX 模块。

7) MAX 模块用于实现两路数据汇聚转发功能，其中 PTP 报文优先转发。

8) GDO、GDI 通过检测 cyc_init 信号，若检测到 cyc_init 信号稳定，则将 master_ms_cnt 值赋给 txN_ms_cnt/rxN_ms_cnt，master_ts_cnt 值赋给 tx_ts_cnt/rx_ts_cnt。

GDO 模块根据报文以太网类型域识别报文类型。若该位为 88f7，则计算透明时钟 TC，并将 TC 值与报文修正域的值相加以更新修正域的值，而后将报文发送出去。若该位不为 88f7，则 UDO 不做任何操作直接转发；GDI 则在报文输入时为每个报文打上时间戳。

3.2 DMUX 模块

3.2.1 DMUX 功能分析

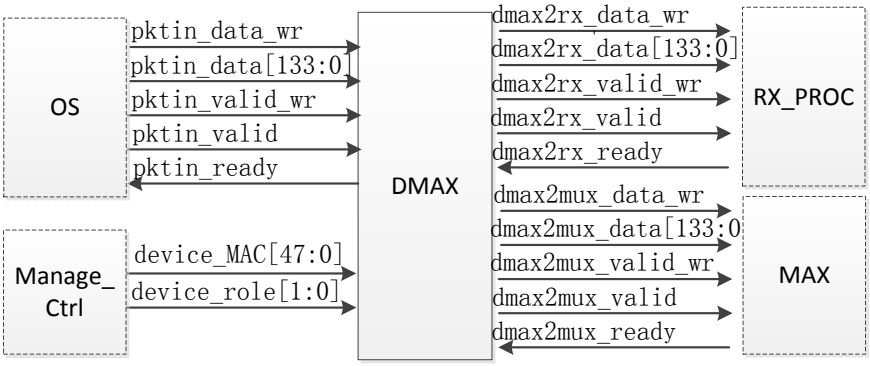
功能描述：

a) 根据角色判断是端设备（主时钟端/从时钟端）还是交换机。

b) 端设备用于根据 EthTYPE 域判断是否为 PTP 报文。若是则判断 DMAC 地址是否匹配本设备的 MAC 地址，若匹配（或为广播地址）则送给 RX_PROC 模块处理，否则输出给 MAX 模块。若不是 PTP 报文则送给 MAX 模块。

c) 交换设备用于根据 EthTYPE 域判断是否为 PTP 报文。若是则判断 DMAC 地址是否匹配本设备的 MAC 地址，若匹配（或为广播地址）则送给 RX_PROC 模块处理，并将广播地址的 PTP Sync 报文同时输出给 MAX 模块；若 MAC 地址不匹配，则输出给 MAX 模块。若不是 PTP 报文则送给 MAX 模块。

3.2.2 DMUX 接口定义



接口信号	位宽	方向	含义

clk	1	Input	时钟
reset	1	Input	复位，低使能
OS 与 DMAX 模块间交互信号			
pktin_data_wr	1	Input	数据输入有效，高有效
pktin_data	134	Input	数据输入
pktin_data_valid_wr	1	Input	数据有效位的有效，高有效
pktin_data_valid	1	Input	数据有效位
pktin_ready	1	Output	输入使能，高有效
DMAX 与 RX_PROC 模块间交互信号			
dmax2rx_data_wr	1	Output	数据输出有效
dmax2rx _data	134	Output	数据输出
dmax2rx_data_valid_wr	1	Output	数据有效位的有效
dmax2rx_data_valid	1	Output	数据有效位
dmax2rx_ready	1	Input	输出使能，高有效
DMAX 与 MAX 模块间交互信号			
dmax2max_data_wr	1	Output	数据输出有效
dmax2max _data	134	Output	数据输出
dmax2max_data_valid_wr	1	Output	数据有效位的有效
dmax2max_data_valid	1	Output	数据有效位
dmax2max_ready	1	Input	输出使能，高有效
Manager_Ctrl 与 DMAX 模块间交互信号			
device_MAC	48	Input	设备 MAC 地址
device_role	2	Input	设备角色 00：从时钟端，01 主时钟端，10 交换机

3.2.3 DMUX 模块详细设计

DMUX 模块包含两个“always”逻辑，第一个“always”逻辑用于将报文输入给 Parser 模块或快速通道，第二个“always”逻辑转发 Gen_Sel 模块与快速通道中的报文。

在第一个“always”逻辑中，含有状态机 cmp_state，用于判断报文是否传给 Parser 模块或快速通道。

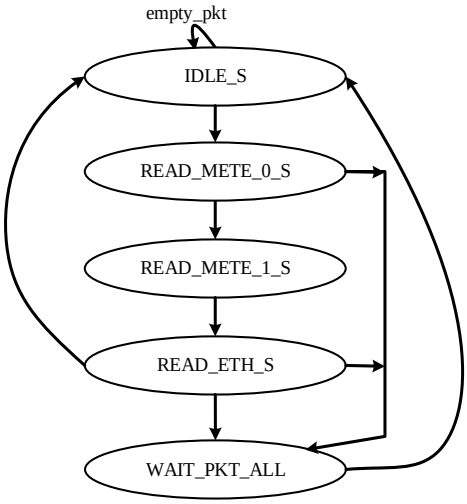


图 4-3 状态机 cmp_state 状态转换

如图 4-3 所示，cmp_state 状态机包含以下几个状态：

1） IDLE_S: 初始化参数，判断 fifo_pkt 是否不空（！emtpy_pkt），若有数据，则使能 rdreq_pkt，进入 READ_META_0_S 状态；
2） READ_META_0_S：判断 q_pkt[133:132]是否为 2'b01，若是，则将该拍数据存储在寄存器 reg1 中，并跳转到 READ_DREQ_META_1_S。否则，将寄存器 state_reg（用于标识进入 WAIT_PKT_ALL 前的状态）置为 2 后进入 WAIT_PKT_ALL 状态，将该报文排空；
3） READ_META_1_S：读取该拍数据（数据为空），使用寄存器 reg2 存储该拍数据后跳转到 READ_DREQ_ETH_S 状态；
4） READ_ETH_S：判断该拍中的协议类型是否为 0880(PTP)，若是，使用寄存器 reg3 存储该拍数据后跳转到 READ_IP_S。否则，将寄存器 state_reg 置为 4 后跳转到 WAIT_PKT_ALL 状态；
5） WAIT_PKT_ALL：若 state_reg 值为 2，则将此报文排空；若 state_reg 值为 4，则将 reg1，reg2 与 reg3 写入快速通道的 fifo_pkt 中后，将此报文的后续拍的数据也写入此 fifo_pkt 中；若一个报文处理完毕后，将状态跳转为 IDLE_S 状态。

在第二个“always”逻辑中，首先判断存储 Gen_Sel 模块传输报文的 FIFO 是否为空，若不为空，则将此 FIFO 中的报文转发出去。若为空，接着判断快速通道中的 FIFO 是否为空，若不为空，则转发该 FIFO 中的报文。

3.3 RX_proc 模块

3.3.1 RX_proc 功能分析

Pm RX_proc 的功能

a) 解析 PTP 的类型；
b) PTP 为 Delay_req 时，向 PTP_CTRL 发送 REQ_GET 信号；向 TX_PROC 发送 SMAC、IN_PORT 信息（用 FIFO 存储）；
c) 根据当前时间与 Metadata 的时间戳，计算透明时间，并根据透明时间计算 T4 的值，并输出给 TX_PROC 模块。

Ps RX_proc 的功能

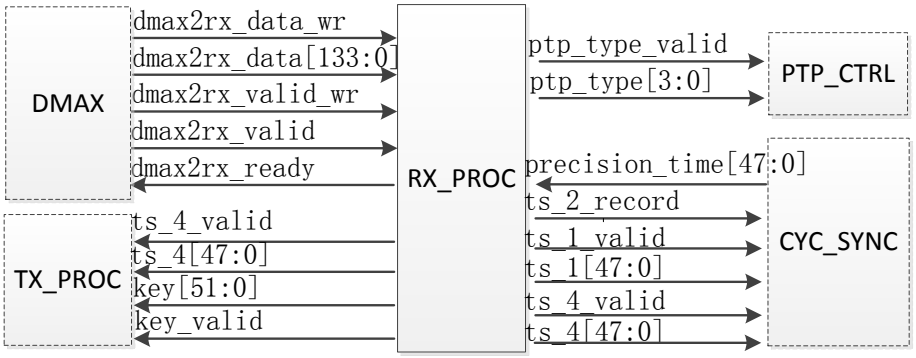
解析 PTP 的 TX_PROC 类型；

a) PTP 为 Sync 时，通知 CYC_SNYC 记录 T2 的时间，根据当前时间与 Metadata 的时间戳，计算透明时间，并根据透明时间计算 T1 的时间送给 CYC_SNYC 模块。向 PTP_CTRL 发送 SYNC_GET(4'd1)信号，向 TX_PROC 发送 SMAC、IN_PORT 信息（用 FIFO 存储）。
b) PTP 为 Delay_resq 时，并将 T4 的时间送给 CYC_SNYC 模块。向 PTP_CTRL 发送 RESQ_GET 信号。

Pm 与 Ps 功能合并：

a) 解析 PTP 的类型；
b) PTP 为 Sync 时，通知 CYC_SNYC 记录 T2 的时间，根据当前时间与 Metadata 的时间戳，计算透明时间，并根据透明时间计算 T1 的时间送给 CYC_SNYC 模块。向 PTP_CTRL 发送 SYNC_GET(4'd1)信号，向 TX_PROC 发送 SMAC、IN_PORT 信息。
c) PTP 为 Delay_req 时，向 PTP_CTRL 发送 REQ_GET(4'd3)信号；向 TX_PROC 发送 SMAC、IN_PORT 信息（用 FIFO 存储）。 根据当前时间与 Metadata 的时间戳，计算透明时间，并根据透明时间计算 T4 的值，并输出给 TX_PROC（用 FIFO 存储）。
d) PTP 为 Delay_resq 时，并将 T4 的时间送给 CYC_SNYC 模块。向 PTP_CTRL 发送 RESQ_GET(4'd4)信号。

3.3.2 RX_proc 接口定义



接口信号	位宽	方向	含义
Pktin/ pktout			
dmax2rx_data_wr	1	Input	数据输入有效，高有效
dmax2rx _data	134	Input	数据输入
dmax2rx_data_valid_wr	1	Input	数据有效位的有效，高有效
dmax2rx_data_valid	1	Input	数据有效位
dmax2rx_ready	1	Output	输入使能，高有效
Rx_proc 与 PTP_Ctrl 之间的交互信号			
Ptp_type_valid	1	Output	用于输出 PPT 报文类型有效信号
Ptp_type	4	Output	PTP 报文类型，4'd1：SYN_GET，为 4'd3:REQ_GET，4'd4:RESQ_GET

RX_PROC 与 TX_PROC 之间的交互信号			
key_valid	1	Output	关键字输入有效，高有效
key	51	Output	源 MAC 地址与输入端口号
ts_4_valid	1	Output	高有效，用于通知记录时间戳 t4
T4	48	Output	用于存储 delay_req 报文经过的透明时钟
RX_PROC 与 CYC_SYNC 之间的交互信号			
ts_1_valid	1	Output	时间戳 t1 有效位，高有效
ts_1	48	Output	时间戳 t1
ts_4_valid	1	Output	时间戳 t4 有效位，高有效
ts_4	48	Output	时间戳 t4
ts_2_record	1	Output	高有效，用于通知记录时间戳 t2
Procision_time	48	input	精确时间

3.3.3RX_proc 详细设计

Rx_proc 模块包含一个状态机 parser_state，用于提取关键字 key 与时间戳。

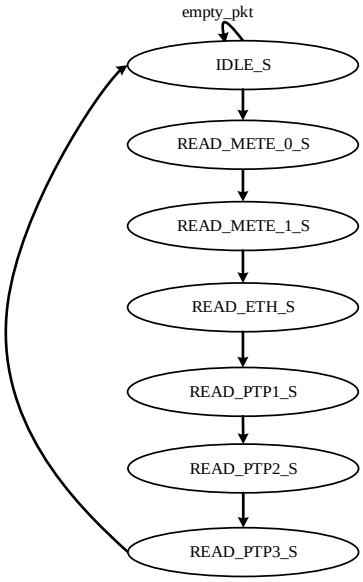


图 4-3 状态机 parser_state 状态转换

如图 4-3 所示，parser_state 包含以下几个状态：

1) IDLE_S：初始化参数，判断 fifo_pkt 是否不空（！emtpy_pkt），若有数据，则使能 rdreq_pkt，进入 READ_DREQ_META_0_S 状态；
2) READ_META_0_S：数据跳转到状态 READ_META_1_S；
3) READ_META_1_S：读取该拍数据（数据为空），直接跳转到 READ_ETH_S 状态；
4) READ_ETH_S：将该拍数据的源与目的 MAC 地址存入 key 后状态跳转为 READ_PTP1_S；
5) READ_PTP1_S：若消息类型为 sync 报文，则记录修正域的值 T1，并将 ts_2_record 标志位置为

高，而后状态跳转为 READ_PTP2_S；若消息类型为 delay_req 报文，则记录修正域的值 T2，而后状态跳转为 READ_PTP2_S；若消息类型为 delay_resq 报文，则直接跳转到 READ_PTP2_S 状态。

6) READ_PTP2_S：状态直接跳转为 READ_PTP3_S。
7) READ_PTP3_S：若消息类型为 sync 报文，则记录时间戳$t1$，将 ts_2_record 标志位置为高，并根据$t1 = t1 + T_1$更新 $t1$，而后状态跳转为 IDLE_S；若消息类型为 delay_req 报文。则将$t4_record$置为高，而后状态跳转为 IDLE_S；若消息类型为 delay_resq 报文，则将该拍数据中时间戳$t4$记录下来后，而后状态跳转为 IDLE_S；

3.4 TX_proc 模块

3.4.1 TX_proc 功能分析

1） Pm TX_proc 的功能

a) 根据 CYC_SYNC 模块发送的 send_sync_pkt 信号，构造 Sync 的 PTP 报文发送，透明时钟域填为 0；在 Metadata 时间域打上时间戳（用于计算ＵＭ－ＵＤＯ的延时），DMAC 为广播地址，SMAC 为 Manage_CTRL 配置 MAC 地址。
b) 缓存 RX_PROC 发送的 SMAC、INPORT 的信息，以及 T4 的时间戳信息；
c) 根据 PTP_CTRL 发送的 send_dresq_pkt 信号，构造 Delay_resq 的 PTP 报文发送，并将 T4 的时间，填充于时间戳域，在 Metadata 时间域打上时间戳（用于计算ＵＭ－ＵＤＯ的延时），DMAC 为 KYE FIFO 中的 MAC，SMAC 为 CTRL 配置 MAC 地址，输出的 DMID 为 5，输出端口号为 KEY FIFO 的 INPORT 值。

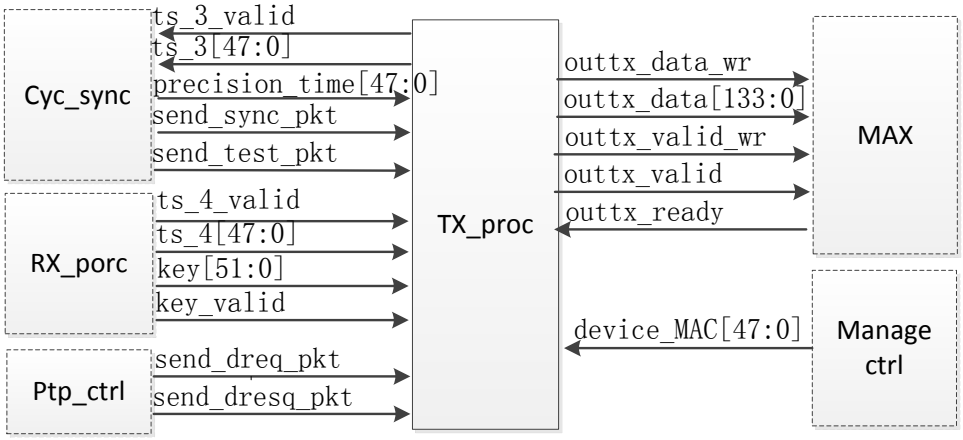
2） Ps TX_proc 的功能

a) 根据 PTP CTRL 发送的 send_dreq_pkt 信号，构造 Delay_req 的 PTP 报文发送，并将 T3 的时间，填充于时间戳域，透明时钟域填为 0，SMAC 为 CTRL 配置 MAC 地址， 输出端口为 KEY FIFO 的 INPORT 值，并将 T3 时间给送 CYC_SYNC 模块。Metadata 时间域打上时间戳（用于计算ＵＭ－ＵＤＯ的延时）。

Pm 与 Ps 功能合并：

a) 根据 CYC_SYNC 模块发送的 send_sync_pkt 信号，构造 Sync 的 PTP 报文发送，透明时钟域填为 0；在 Metadata 时间域打上时间戳（用于计算ＵＭ－ＵＤＯ的延时），DMAC 为广播地址，SMAC 为 Manage_CTRL 配置 MAC 地址。
b) 根据 PTP CTRL 发送的 send_dreq_pkt 信号，构造 Delay_req 的 PTP 报文发送，并将 T3 的时间，填充于时间戳域，透明时钟域填为 0，SMAC 为 CTRL 配置 MAC 地址， 输出端口为 KEY FIFO 的 INPORT 值，并将 T3 时间给送 CYC_SYNC 模块。Metadata 时间域打上时间戳（用于计算ＵＭ－ＵＤＯ的延时）。
c) 根据 PTP_CTRL 发送的 send_dresq_pkt 信号，构造 Delay_resq 的 PTP 报文发送，并将 T4 的时间，填充在时间戳域，在 Metadata 时间域打上时间戳（用于计算ＵＭ－ＵＤＯ的延时），DMAC 为 KYE FIFO 中的 MAC，SMAC 为 CTRL 配置 MAC 地址，输出的 DMID 为 5，输出端口号为 KEY FIFO 的 INPORT 值。

3.4.2 TX_proc 接口定义



接口信号	位宽	方向	含义
内部接口信号			
Manage_CTRL 与 TX_PROC			
Device_MAC	48	Input	配置的 MAC 地址
CYC_SYNC 与 TX_PROC 之间的交互信号			
ts_3_valid	1	Output	T3 输出有效信号。
Ts_3	48	output	T3 输出的值。
timer	48	input	用于存储 Clock 模块的计数器值
send_sync_pkt	1	input	高有效，用于通知发送 sync 报文
send_test_pkt	1	input	高有效，用于通知发送 test 报文
PTP_Ctrl 与 TX_PROC 之间的交互信号			
send_dreq_pkt	1	Input	高有效，通知发送 deley_req 报文
send_dresq_pkt	1	input	高有效，通知发送 deley_resq 报文
Rx_proc 与 Tx_proc 之间的交互信号			
key_valid	1	R2T	关键字输入有效，高有效
key	51	R2T	源 MAC 地址及输入端口号
ts_4_valid	1	R2T	高有效，用于通知记录时间戳 t4
T4	48	R2T	用于存储 T4
MUX 与 TX_PROC 之间的交互信号			
outtx_data_wr	1	Output	数据输出有效
outtx_data	134	Output	数据输出
outtx_data_valid_wr	1	Output	数据有效位的有效
outtx_data_valid	1	Output	数据有效位
outtx_ready	1	Input	输出使能，高有效

3.4.3 TX_proc 模块详细设计

Tx_proc 模块包含 Gen_Sel_state 一个状态机，用于构造并发送报文。

如图 4-4 所示，Gen_Sel_state 状态机包含以下状态：

1) IDLE_S：初始化参数，判断(send_sync_pkt || send_dreq_pkt || send_resq_pkt || send_test_pkt)是

否为高，若为高，则进入 GEN_META_0_S 状态；

2) GEN_SYNC_META_0_S：若 send_sync_pkt 为高，则构造 sync 报文的 metedata0 后状态跳转为 GEN_META_1_S；若 send_dreq_pkt 为高，则构造 delay_req 报文的 metedata0 后状态跳转为 GEN_META_1_S；若 send_dresq_pkt 为高，则构造 delay_resq 报文的 metedata0 后状态跳转为 GEN_META_1_S；若 send_test_pkt 为高，则构造 test 报文的 metedata0 后状态跳转为 GEN_META_1_S。

3) GEN_META_1_S:该拍数据填充全 0，而后状态跳转为 GEN_ETH_S；

4) GEN_ETH_S：若 send_sync_pkt 为高，则构造 sync 报文的 ETH 拍后状态跳转为 GEN_IP_S；若 send_dreq_pkt 为高，则构造 delay_req 报文的 ETH 拍，而后状态跳转为 GEN_IP_S；若 send_dresq_pkt 为高，则构造 delay_resq 报文的 ETH 拍，而后状态跳转为 GEN_IP_S; 若 send_test_pkt 为高，则构造 test 报文的 ETH 拍后状态跳转为 GEN_IP_S。

5) GEN_ PTP_1_S：GEN_UDP_S：若 send_sync_pkt 为高，则构造 sync 报文的第一拍 PTP 拍后状态跳转为 GEN_PTP2_S；若 send_dreq_pkt 为高，则构造 delay_req 报文的第一拍 PTP，而后状态跳转为 GEN_PTP2_S；若 send_dresq_pkt 为高，则构造 delay_resq 报文的第一拍 PTP，而后状态跳转为 GEN_PTP2_S；若 send_test_pkt 为高，则构造 test 报文的第一拍 PTP 拍后状态跳转为 GEN_PTP2_S。

6) GEN_PTP_2_S：若 send_sync_pkt 为高，则构造 sync 报文的第二拍 PTP 拍后状态跳转为 GEN_PTP3_S；若 send_dreq_pkt 为高，则构造 delay_req 报文的第二拍 PTP，而后状态跳转为 GEN_PTP3_S；若 send_dresq_pkt 为高，则构造 delay_resq 报文的第二拍 PTP，而后状态跳转为 GEN_PTP3_S；若 send_test_pkt 为高，则构造 test 报文的第二拍 PTP 拍后状态跳转为 GEN_PTP3_S。

7) GEN_PTP_3_S:若 send_sync_pkt 为高，则构造 sync 报文的第三拍 PTP 拍后状态跳转为 IDLE_S；若 send_dreq_pkt 为高，则构造 delay_req 报文的第三拍 PTP，而后状态跳转为 IDLE_S；若 send_dresq_pkt 为高，则构造 delay_resq 报文的第三拍 PTP，而后状态跳转为 IDLE_S；若 send_test_pkt 为高，则构造 test 报文的第三拍 PTP 拍后状态跳转为 IDLE_S；。

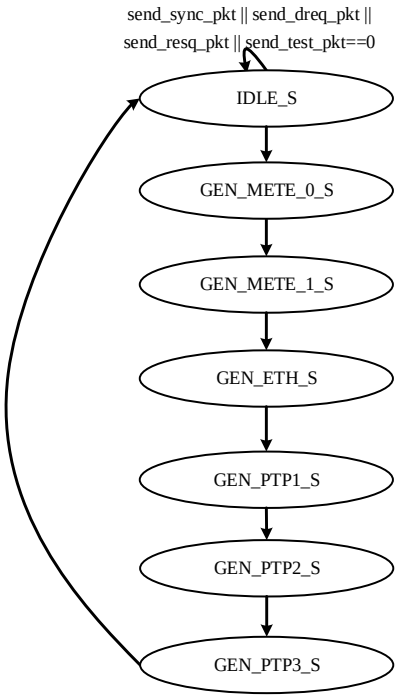


图 4-4　Gen_Sel_state 状态转换

3.5 Manage_CTRL 模块

3.5.1 Manage_CTRL 功能分析

通过 Cin 下发同步启动命令并配置设备以及 MAC 地址。

接口信号	位宽	方向	含义
configuration			
cin_data	134	Input	控制输入有效，高有效
cin_data_wr	1	Input	控制输入
cin_ready	1	Output	输入使能，高有效
cout_data	134	Output	控制输出有效
cout_data_wr	1	Output	控制输出
cout_ready	1	Input	输出使能，高有效
Manage_CTRL 与 PTP_CTRL 之间的交互信号			
Sync_start	1	output	通知同步是否开始
device_role	2	output	设备角色 00：从时钟端、01 主时钟端、10 交换机
error	1	input	用于反馈同步失败
Manage_CTRL 与 TX_PROC 之间的交互信号			
device_MAC	48	Output	输出的 MAC
Manage_CTRL 与 CYC_SYNC 之间的交互信号			
error1	1	input	Offset 超过 1ms 报错
Manage_CTRL 与 DMAX 之间的交互信号			
device_MAC	48	Output	输出的 MAC

device_role	2	output	设备角色 00：从时钟端、01 主时钟端、10 交换机

3.6 PTP_CTRL 模块

3.6.1 PTP_CTRL 功能分析

1） Pm PTP_CTRL：

a) 根据 Sync_start 及 master_or_slave 信号判断配置的主/从，当 Sync_start 为 1，master_or_slave 为 1，则 m_or_s 信号为 1，即当前运行于主状态（Pm）；否则运行于从状态(Ps)，默认为从。

b) 监听 req_get 信号，若有效，则输出 send_resq_pkt 信号。

2） Ps PTP_CTRL：

a) 监听 sync_get 信号，若有效，则输出 send_ req_pkt 信号，则状态跳转等待 resq_get 信号，若有效，则状态跳转等待 sync_get，输出 status_ok 信号。

b) 当状态出错时，则输出 error 信号。

Pm 与 Ps 功能合并：

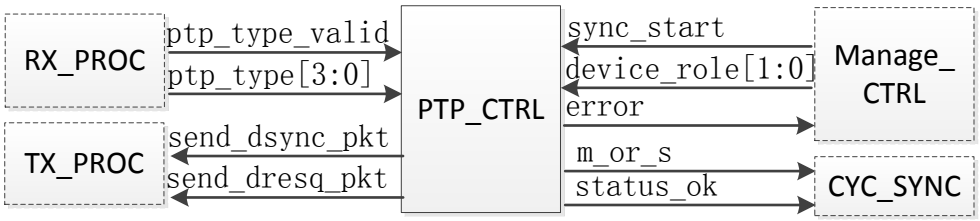
a) 根据 Sync_start 及 master_or_slave 信号判断配置的主/从，当 Sync_start 为 1，master_or_slave 为 1，则 m_or_s 信号为 1，即当前运行于主状态（Pm）；否则运行于从状态(Ps)，默认为从。

b) 监听 sync_get 信号，若有效，则输出 send_ req_pkt 信号，则状态跳转等待 resq_get 信号，若有效，则状态跳转等待 sync_get，输出 status_ok 信号。

c) 当状态出错时，则输出 error 信号。

d) 监听 req_get 信号，若有效，则输出 send_resq_pkt 信号。

3.6.2 PTP_CTRL 接口定义



接口信号	位宽	方向	含义
Manage_CTRL 与 PTP_CTRL 之间的交换信号			
Sync_start	1	input	通知同步是否开始
device_role	2	input	设备角色 00：从时钟端、01 主时钟端、10 交换机
error	1	output	用于反馈同步失败
PTP_Ctrl 与 Tx_proc 之间的交互信号			
send_dreq_pkt	1	output	高有效，通知发送 deley_req 报文
send_dresq_pkt	1	output	高有效，通知发送 deley_resq 报文

PTP_Ctrl 与 TX_PROC 之间的交互信号			
send_dreq_pkt	1	output	高有效，通知发送 deley_req 报文
send_dresq_pkt	1	output	高有效，通知发送 deley_resq 报文
PTP_Ctrl 与 CYC_SYNC 之间的交互信号			
m_or_s	1	output	主从控制信号
status_ok	1	output	从交互状态跳转正常信号

3.7 CYC_SYNC 模块

3.7.1 CYC_SYNC 模块功能分析

1） Pm 功能描述：

a) 维持一个 48 位的计数器{ms_cnt，cycle_cnt}，当 m_or_s 为 1 时，开始 cyc_cnt 的计数，当 cyc_cnt 计数为 125000 时，则 ms_cnt 加 1，cyc_cnt 清 0。

b) 当 cyc_cnt 为 512 时，则 Send_sync_pkt 置 1 一个周期，当 cyc_cnt 为 1024 时置 Send_test_pkt 为 1 一个周期。

2） Ps 功能描述：

a) 维持一个 48 位的计数器{ms_cnt，cycle_cnt}，复位开始后 cyc_cnt 开始计数，当 cyc_cnt 计数为 125000 时，则 ms_cnt 加 1，cyc_cnt 清 0。

b) 当 ts_1_valid 有效时，则缓存 T1 时间戳；当 ts_2_record 有效时，则缓存 T2 时间戳；当 ts_3_valid 有效时，则缓存 T3 时间戳；当 ts_4_valid 有效时，则缓存 T4 时间戳，待 status 信号到来之后根据公式计算 OFFSET，更新{ms_cnt，cycle_cnt}值，并更新 temp_cnt 的值。

c) 若 offset 超过 1ms，则输出 error1。

Pm 与 Ps 功能合并：

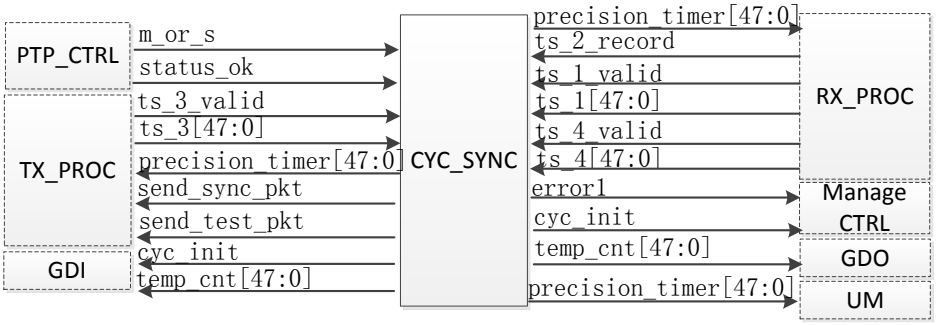
a) 维持一个 48 位的计数器{ms_cnt，cycle_cnt}，复位开始后 cyc_cnt 开始计数，当 cyc_cnt 计数为 125000 时，则 ms_cnt 加 1，cyc_cnt 清 0。

b) 当 cyc_cnt 为 512 时，则 Send_sync_pkt 置 1 一个周期，当 cyc_cnt 为 1024 时置 Send_test_pkt 为 1 一个周期。

c) 当 ts_1_valid 有效时，则缓存 T1 时间戳；当 ts_2_record 有效时，则缓存 T2 时间戳；当 ts_3_valid 有效时，则缓存 T3 时间戳；当 ts_4_valid 有效时，则缓存 T4 时间戳，待 status 信号到来之后根据公式计算 OFFSET，更新{ms_cnt，cycle_cnt}值，并更新 temp_cnt 的值。

d) 若 offset 超过 1ms，则输出 error1。

3.7.2 CYC_SYNC 接口定义



PTP_Ctrl 与 CYC_SYNC 之间的交互信号			
m_or_s	1	input	主从控制信号
status_ok	1	input	从交互状态跳转正常信号
CYC_SYNC 与 TX_PROC 之间的交互信号			
ts_3_valid	1	input	T3 输出有效信号。
Ts_3	48	input	T3 输出的值。
procision_time	48	output	精确时间值
send_sync_pkt	1	output	高有效，用于通知发送 sync 报文
RX_PROC 与 CYC_SYNC 之间的交互信号			
ts_1_valid	1	input	时间戳 t1 有效位，高有效
ts_1	48	input	时间戳 t1
ts_4_valid	1	input	时间戳 t4 有效位，高有效
ts_4	48	input	时间戳 t4
ts_2_record	1	input	高有效，用于通知记录时间戳 t2
procision_time	48	output	精确时间值
Ctrl 与 CYC_SYNC 之间的交换信号			
error1	1	Outpu	Offset 超过 1ms 报错
CYC_SYNC 与 GDO,GDI 之间的交互信号			
temp_cnt	48	Output	用于根据调整后的计数器值同步外围时钟
cyc_init	1	Output	用于根据调整后的计数器值同步外围时钟的有效信号
CYC_SYNC 与 UM 之间的交互信号			
Procision_time	48	output	精确时间值

3.7.3 CYC_SYNC 模块详细设计

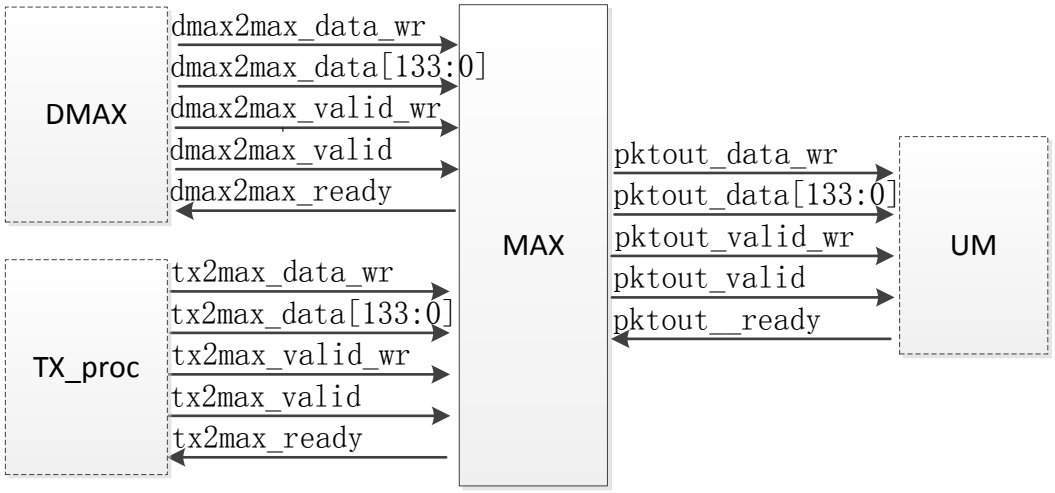
该模块维持一个 48 位的计数器{ms_cnt，cycle_cnt}。若 ts_2_record 信号为高，则记录此时的计数器值 t2。若 ts_3_record 信号为高，则记录此时的计数器值 t3。若 m_or_s 为高，则 Clock 模块每隔 1ms 将 send_sync_pkt 为高，同时将 cyc_init 信号置为高。1000 个周期后，将 send_test_pkt 置为高，其每次置高的时间间隔也为 1ms。若 ts_4_valid 为高，则开始计算时间偏移量 offset，而后根据 offset 值调整计数器值，并将计数器的值赋给 temp_cnt 信号。

3.8 MUX 模块

3.8.1 MUX 功能分析

用于调度输出 PTP 的报文以及普通以太网报文，调度时优先高度 PTP 报文输出。

3.8.2MUX 接口定义



接口信号	位宽	方向	含义
clk	1	Input	时钟
reset	1	Input	复位，低使能
MAX 与 UM 模块间交互信号			
pktout_data_wr	1	Output	数据输入有效，高有效
pktout_data	134	Output	数据输入
pktout_data_valid_wr	1	Output	数据有效位的有效，高有效
pktout_data_valid	1	Output	数据有效位
Pktout_ready	1	input	输入使能，高有效
TX_PROC 与 MAX 模块间交互信号			
tx2max_data_wr	1	Input	数据输出有效
tx2max_data	134	Input	数据输出
tx2max_data_valid_wr	1	Input	数据有效位的有效
tx2max_data_valid	1	Input	数据有效位
tx2max_ready	1	Output	输出使能，高有效
DMAX 与 MAX 模块间交互信号			
dmax2max_data_wr	1	Input	数据输出有效
dmax2max _data	134	Input	数据输出
dmax2max_data_valid_wr	1	Input	数据有效位的有效
dmax2max_data_valid	1	Input	数据有效位
dmax2max_ready	1	Output	输出使能，高有效

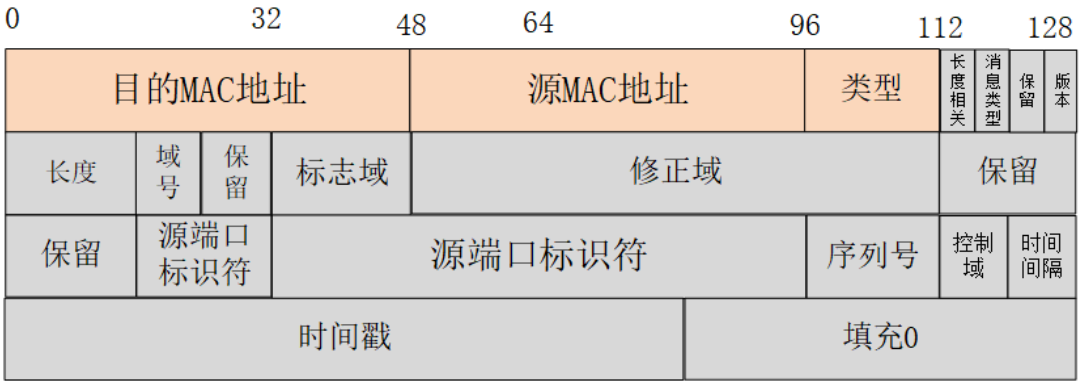
附录一、metadata 格式以及 PTP 同步报文格式

1、Metadata 格式

表 5-1　Metadata 格式

Metadata 0			
[127]	1	pktsrc	分组的来源，0 为网络接口输入，1 为 CPU 输入
[126]	1	pktdst	分组目的，0 为网络接口输出，1 为送 CPU
[125:120]	6	inport	分组的输入端口号
[119:118]	2	outType	00:单播；01：组播；10：泛洪；11：从输入接口输出
[117:112]	6	outPort	单播：分组输出端口 ID，组播/泛洪：组播/泛洪表地址索引
[111:109]	3	priority	分组优先级
[108]	1	discard	丢弃位
[107:96]	12	len	包含 MetaData 字段的分组长度（用于状态管理）
[95:88]	8	smid	最近一次处理分组的模块 ID
[87:80]	8	dmid	下一个处理分组的模块 ID
[79:72]	8	pst	标准协议类型（增加 HTTP GET/POST，SMTP，FTP 定义）
[71:64]	8	seq	分组接收序列号
[63:50]	14	flowid	流 ID/IDS 规则标识，即 ruleID
[49:32]	18	reserve	保留
[31:0]	32	ts	报文时间戳信息

2、Sync，Delay_req，Delay_resq 与 test 报文格式



类型为：16'h88F7；

消息类型：sync 为 4'd1，delay_req 为 4'd3，delay_resq 为 4'd4，delay_test 为 4'd5；

长度为：16'd64 字节；

修正域:透明时钟，起始时，该域为 0；

时间戳：为时间戳（其他无需关系的 PTP 字段填充 0）

附录二、版本管理

进度	任务
2019.01.02-2019.01.4	完成代码编写并单模块仿真
2019.01.07-2019.01.8	完成整合并整体仿真
2019.01.09-2019.01.12	完成 FAST 工程合并并上板测试

任务安排：

任务	人员
FAST_OS 输入及输出 UDO 模块、MAX 模块	王耀祥
PTP_CTRL、CYC_SYNC、RX_Proc	付文文
CTRL、TX_Proc、port_cmp	张彦龙
整体仿真	王耀祥
与 FAST 工程合并	张彦龙
上板测试	张彦龙、付文文、王耀祥
项目整体进度	张彦龙