密级：

**物理层理解报告**

编 号：

版本/修订：V1.0

编 制 王妮娜

审 核

会 签

批 准

**文档修订历史**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **版本** | **作者** | **版本变化对象** | **变化内容描述** | **审核人** | **批准人** | **修订日期** |
| V1.0 | 王妮娜 |  | 新建文档及纲要 |  |  | 2018.3.13 |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

目 录

[1. 前言 1](#_Toc511204860)

[1.1. 编写目的 1](#_Toc511204861)

[1.2. 背景 1](#_Toc511204862)

[1.3. 定义 1](#_Toc511204863)

[1.4. 参考资料 1](#_Toc511204864)

[2. 平台介绍（王妮娜） 2](#_Toc511204865)

[2.1. 硬件平台 2](#_Toc511204866)

[2.2. 软件平台 2](#_Toc511204867)

[2.3. 物理层软件 3](#_Toc511204868)

[3. 物理层测试（张英英） 4](#_Toc511204869)

[3.1. 测试架构 4](#_Toc511204870)

[3.2. 测试方法 4](#_Toc511204871)

[4. 集成测试（税敏） 5](#_Toc511204872)

[4.1. 测试环境搭建 5](#_Toc511204873)

# 

# 引言

## 编写目的

该文档依据Intel提供的英文资料、结合实际实践经验整理而成，主要目的为供初学者快速理解物理层架构、实现方法、操作步骤等。

## 预期读者及阅读建议

该文档的预期读者为从事物理层研发、测试、集成等工作的工程师；

## 参考资料

1. FlexRAN\_LTE\_RefPHY-R1.3.0\_Doxygen\_Documentation；
2. 548166\_FerryBridge\_UserGuide\_rev1\_0；
3. 571742\_Intel\_FlexRAN\_LTE\_eNB\_L2\_L1\_API\_Rev\_1\_4；
4. 572010\_FlexRAN\_Framework\_SAS\_Rev1\_2；
5. 574223\_FerryBridge\_1\_0\_Programmers\_Guide；
6. 574867\_FlexRAN\_Framework\_User\_Guide\_\_Rev1\_0；
7. 572004\_FlexRAN\_SDK\_Programmer Guide\_1\_3；
8. 572003\_FlexRAN\_SDK\_Getting\_Started\_Guide\_Rev1\_4；
9. 571741\_FlexRAN\_LTE\_eNB\_\_L1\_XML\_Configuration\_UG\_R1\_3；

# 平台介绍

## 硬件平台

实训平台物理层涉及的硬件及与软件的对应关系如下：

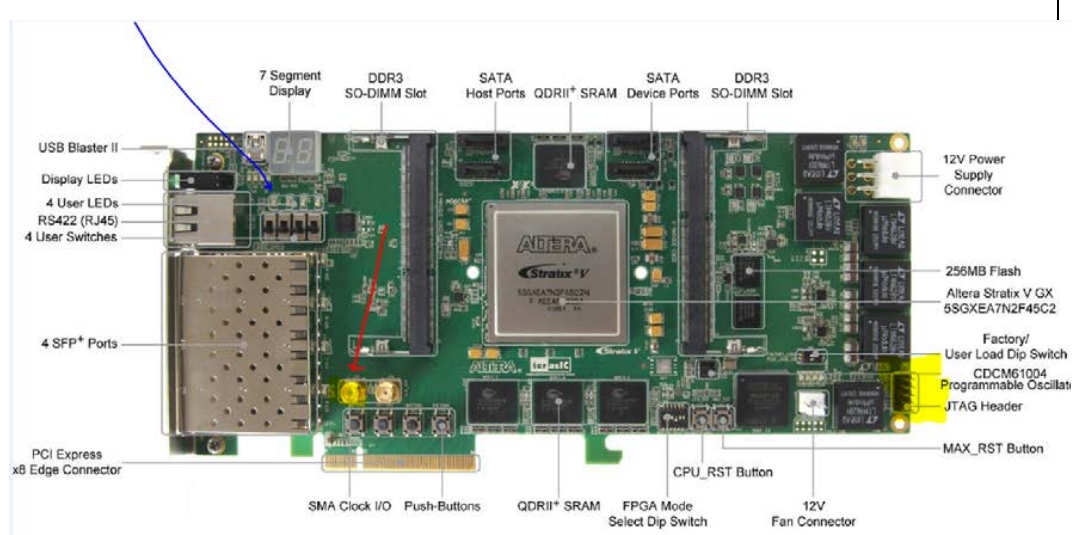
1. Server：主要运行物理层CPU软件；
2. Ferrybrigh：主要运行物理层FPGA软件；

### 硬件配置

当前实训平台Server硬件配置情况如下：

1. Artesyn\* SharpServer\* PCIe-7410 with two 8-core Intel® Xeon® processors at 2.1 GHz
2. Intel® 82599ES 10 Gigabit Ethernet NIC
3. 当前验证服务器型号为HP DL80 Gen9 HPE 2U机架服务器，Intel 至强2颗E5-2600V4 10核2.2G、550V电源、64G内存、2块SSD960G硬盘、i350-t4网卡、X520SR2网卡
4. 服务器内RAID卡要跳过

当前实训平台的Ferrybrigh的硬件配置情况如下：



## 软件平台

### 操作系统

Server上的物理层软件开发基于Linux系统，CentOS 7.0以上版本，内核版本为3.10.0-327.e17.X86\_64 #1 SMP，选择不带界面的最简单的选项，instruct server，此外bois里需要关闭超线程，具体操作为进入BIOS，Advanced->Processor Configuration中Intel Hyper-Threading Tech Disabled。

### 开发软件

Server：ICC、DPDK、GCC、Cmake、gtest；

Ferrybrigh：QUARTUS。

### 底层驱动

DPDK驱动：提供Intel架构下用户空间高效的数据包处理提供库函数和驱动的支持。

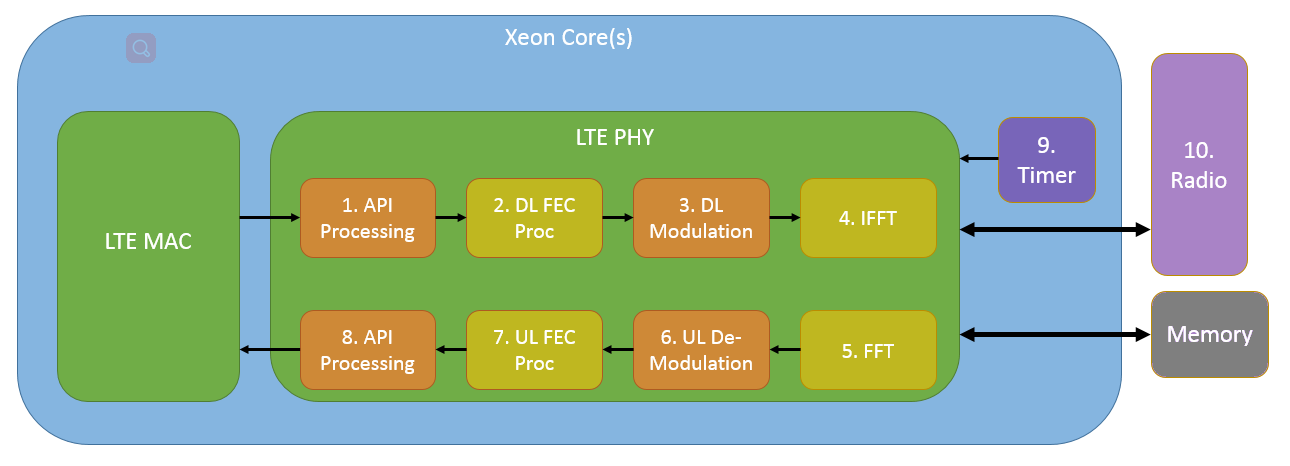
CPRI驱动：提供基带信号的数据接口，能够提供数据进行远端传输。

以太网驱动：提供数据包传输的协议接口。

## 物理层软件

### LTE物理层软件架构

物理层软件可以分为以下几个基本部分，即1-8部分，每部分具体功能将会在图下做详细介绍。



下行方向：（1-4）

1. API Processing，主要处理MAC发给PHY的APIs；
2. DL FEC Proc，主要执行下行编码处理，该部分功能既可以在核内部用软件实现，也可以核外部用FPGA实现；
3. DL Modulation，主要进行符号级处理，该模块的输出为一个完整子帧的所有子载波；
4. IFFT，将下行信号从频域变换到时域，该模块既可以在核内部用软件实现，也可以核外部用FPGA实现；

上行方向：（5-8）

1. FFT，将上行信号由时域变换到频域，该模块既可以在核内部用软件实现，也可以核外部用FPGA实现；
2. UL De-Modulation，主要处理解调等处理，输出为待译码的软比特信息；
3. UL FEC Proc，主要完成译码等处理，该部分既可以在核内部用软件实现，也可以核外部用FPGA实现；
4. API Processing，主要处理PHY发给MAC的APIs；

此外，9，10 模块可以给物理层软件提供数据接口的TTI信息，具体采用哪种方式提供TTI信息，可以通过L2在PHY-START API中设置。

### 接口

#### MAC-PHY接口

MAC-PHY APIs被分为两类，分别是：非实时APIs和实时APIs。其中非实时APIs包括：Phy-Init，Phy-Start，Phy-Stop，Phy-Shutdown。实时APIs主要是在每一个TTI中完成上行、下行配置。 MAC-PHY APIs分类，具体见表1所示。MAC-PHY APIs在lte\_phy\_l2\_api.h文件中定义，详细MAC-PHY APIs涉及的结构体及其参数，可参见lte\_phy\_l2\_api.h文件。MAC-PHY 接口详细内容可参加《LTE物理层CPU软件与协议栈软件接口集成测试设计》文档中第八章（附录）.

表1 MAC-PHY APIs分类

|  |  |  |
| --- | --- | --- |
|  | 非实时APIs | 实时APIs |
| MAC to PHY | Phy\_init\_req  Phy\_start\_req  Phy\_stop\_req | Phy\_txstart\_req（txvector）  Phy\_rxstart\_req（rxvector）  Phy\_txsdu\_req/phy\_txhiadciul\_req/  phy\_txdciulsdu\_req/phy\_txhisdu\_req |
| PHY to MAC | Phy\_init\_ind  Phy\_start\_cofg  Phy\_stop\_ind | Phy\_rxsdu\_int/Phy\_rxstatus\_ind  Phy\_txstart\_ind |

#### CPU-PHY与Ferrybridge接口（李思超）

各类数据包的格式可以统一为如下形式：

起始于21字节

20字节

Data部分

RPE数据包头

其中，RPE数据包头包含以太网地址等信息，data部分为消息内容，后续按照消息类型分别介绍。

##### RPE数据包头

CPU和FPGA之间通过以太网连接，每个数据包头都包含以太网地址。

消息格式如图5-1所示，各部分定义见表5-1。

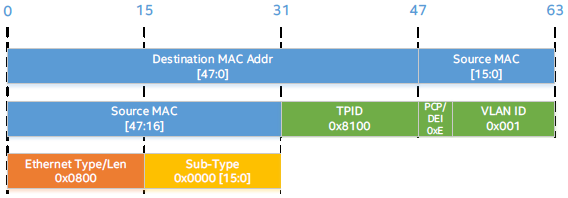


图5-1 RPE数据包头格式

表5-1 RPE数据包头定义

|  |  |
| --- | --- |
| 变量名称 | 说明 |
| d\_addr | 目的地址,48bit |
| s\_addr | 源地址，48bit |
| TPID | tag协议识别，固定值0x8100，表示VLAN tag，16bit |
| PCP | 3bit，数据包优先级，最高为7，最低为0 |
| DEI | 合理丢弃指示，1bit，表明该帧是否可丢弃 |
| VID | 12bit，VLAN标识 |
| Ethernet type | 以太网类型，固定值0x0800,16bit |
| Sub-type | 数据包类型，16bit， |

##### RPE\_LTE\_PDSCH\_PKT\_SUB\_TYPE

下行数据包。消息格式如图5-2所示。保留位为32bit，状态为27bit，配置部分即为消息头，其余为payload。

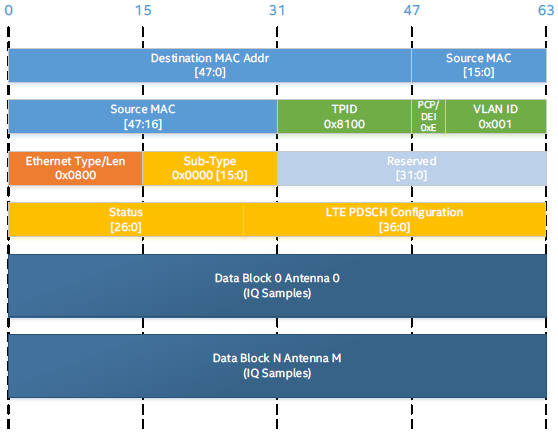


图5-2 PDSCH数据包格式

方向：CPU->FPGA

消息组成:

[reserved][pdsch\_hdr][payload]

消息内容:

pdsch\_hdr部分



图5-3 pdsch\_hdr部分

表5-2 Status功能描述

|  |  |
| --- | --- |
| 位域 | 描述 |
| Status[0] | 该位为真，使能RB选择表 |
| Status[1] | 该位为真，表明为特殊子帧 |
| Status[26:2] | 保留 |

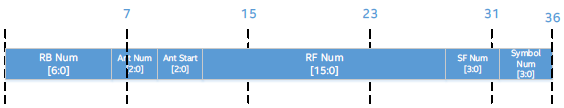


图5-4 Configuration部分的格式

表5-3 PDSCH configuration描述

|  |  |
| --- | --- |
| 位域 | 描述 |
| RB Num | 7bit，PDSCH包的RB数目。所有RB长度相同。对于单个LTE符号，所有RB都应包含在数据包里。RB必须按顺序存放。 |
| Ant Num | 3bit，表示PDSCH数据包所包含的天线数。0表示1根天线，1表示2根天线，一次类推。 |
| Ant Start | 3bit，表示起始天线编号。例如，起始天线为2，天线数为2，则数据包包含天线2和3. |
| RF Num | 16bit，数据包所在帧号。 |
| SF Num | 4bit，表示数据包所在的当前子帧号 |
| Symbol Num | 4bit，表示数据包所在的当前LTE符号 |

Payload部分

##### RPE\_CSR\_PKT\_SUB\_TYPE

上下行数据包。消息格式如图5-5所示。保留位为32bit，payload0即为消息头csr\_hdr，其余为payload。

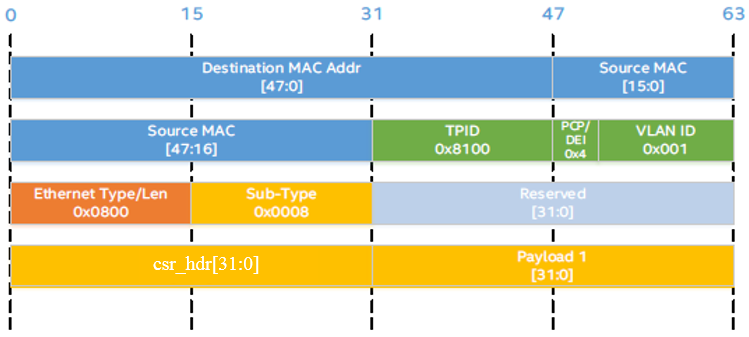


图5-5 CSR数据包格式

方向：CPU<->FPGA

消息组成:

[reserved][csr\_hdr][payload]

消息内容:

csr\_hdr部分

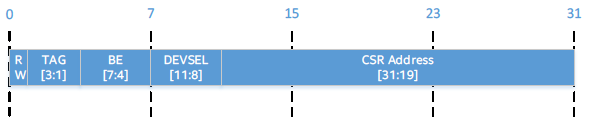


图5-6 csr\_hdr格式

表5-4 csr\_hdr功能描述

|  |  |
| --- | --- |
| 位域 | 描述 |
| RW | 1bit，表示读或写请求 |
| TAG | 3bit，表示CSR请求序列，硬件不关注 |
| BE | 4bit，表示寄存器中哪些字节被写入，对应CSR Address。当前硬件未实现 |
| DEVSEL | 4bit，表示设备选择。见表5-5 |
| CSR Address | 19bit，所选寄存器的地址 |

表5-5 设备选择

|  |  |
| --- | --- |
| 位域 | 4bit标志 |
| Ethernet PHY | 0000 |
| Ethernet MAC | 0001 |
| CPRI | 0100 |
| RPE Local Registers | 1111 |

Payload部分

##### RPE\_TIMING\_PKT\_SUB\_TYPE

上行数据包。消息格式如图5-7所示。RPE包头后为消息头timing\_hdr。

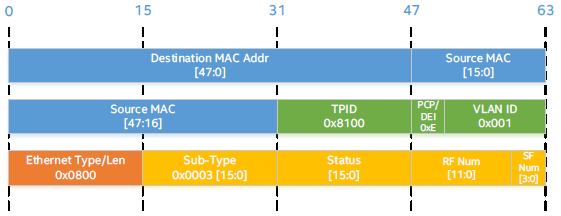


图5-7 定时数据包格式

方向：CPU<-FPGA

消息组成:

[timing\_hdr]

消息内容:

timing\_hdr部分



图5-8 timing\_hdr部分

表5-6 定时包功能描述

|  |  |
| --- | --- |
| 位域 | 描述 |
| Status | 32bit，包含RPE设备和IO接口的各类状态。 |
| RF Num | 12bit，表示当前帧号 |
| SF Num | 4bit，表示当前子帧号 |

表5-7 status定义

|  |  |
| --- | --- |
| 位域 | 4bit标志 |
| Bit[0] | 下行同步，CPRI连接的当前同步状态。1表示同步，0表示未同步 |
| Bit[1] | 表示下行FIFO上溢。0表示未溢出，1表示溢出 |
| Bit[2] | 下行FIFO下溢。0表示未溢出，1表示溢出。 |
| Bit[3] | PRACH 流错误。位于定时包窗口内部，由PRACH RTL模块触发。 |
| Bit[4] | PRACH 帧错误。位于定时包窗口内部，由PRACH RTL模块触发。 |
| Bit[6:5] | PRACH第一错误标识。00：无错误；01：定时包的第一个错误为流错误；10：定时包的第一个错误为帧错误；11：定时包的第一个错误为流错误+帧错误； |
| Bit[15:7] | 保留 |

表5-7 bypass模式下status定义

|  |  |
| --- | --- |
| 位域 | 4bit标志 |
| Bit[0] | 下行同步，CPRI连接的当前同步状态。1表示同步，0表示未同步 |
| Bit[1] | 表示下行读错误 |
| Bit[2] | 表示下行写错误 |
| Bit[3] | 表示上行写错误 |
| Bit[4] | PRACH 流错误。位于定时包窗口内部，由PRACH RTL模块触发。 |
| Bit[5] | PRACH 帧错误。位于定时包窗口内部，由PRACH RTL模块触发。 |
| Bit[7:6] | PRACH第一错误标识。00：无错误；01：定时包的第一个错误为流错误；10：定时包的第一个错误为帧错误；11：定时包的第一个错误为流错误+帧错误； |
| Bit[15:8] | 保留 |

##### RPE\_CPRI\_PKT\_SUB\_TYPE

上下行数据包。消息格式如图5-9所示。RPE包头后为32位的消息头，之后为payload。Bypass模式采用。

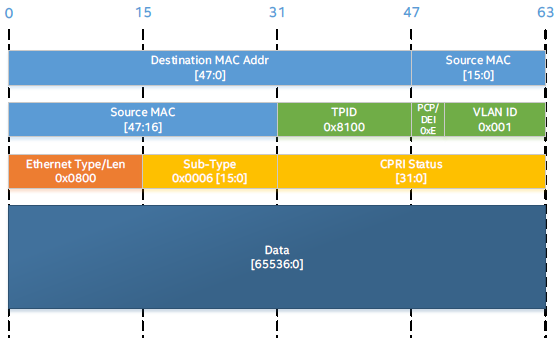


图5-9 CPRI数据包格式

方向：CPU<->FPGA

消息组成:

[cpri\_hdr][payload]

消息内容:

cpri\_hdr部分



图5-10 cpri\_hdr部分

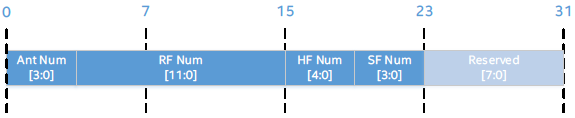


图5-11 cpri\_hdr格式

表5-8 cpri\_hdr功能描述

|  |  |
| --- | --- |
| 位域 | 描述 |
| Ant Num | 超帧所在的天线数目。0-8。第一版仅支持4天线 |
| RF Num | 当前超帧所在帧号，0-4095 |
| HF Num | 数据包对应的超帧号，0-14 |
| SF Num | 数据包对应的子帧号，0-9 |
| Status | 8bit，标识CPRI接口的错误状态 |

表5-9 cpri\_hdr的status状态定义

|  |  |
| --- | --- |
| 位域 | 4bit标志 |
| 0x00 | 无错误 |
| 0x01 | 下溢 |
| 0x02 | 上溢 |
| 0x03 | 信号丢失 |

payload部分

##### RPE\_TIME\_DOMAIN\_PKT\_SUB\_TYPE

上下行数据包。消息格式如图5-12所示。RPE包头后为32位消息头，之后为payload。Bypass模式采用。

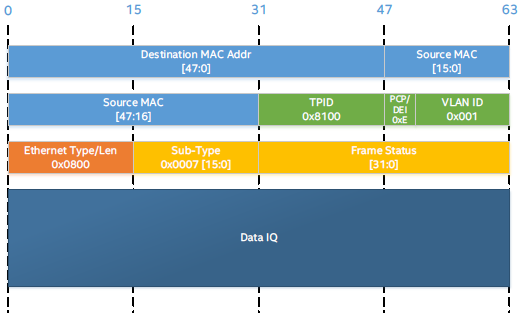


图5-12 时域数据包格式

方向：CPU<->FPGA（待定）

消息组成:

[timedomain\_hdr][payload]

消息内容:

timedomain\_hdr部分

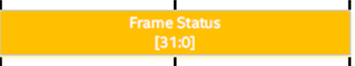


图5-13 timedomain\_hdr部分

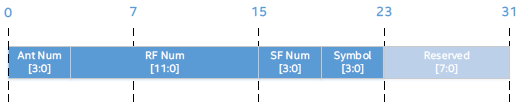


图5-14 timedomain\_hdr格式

表5-10 timedomain\_hdr功能描述

|  |  |
| --- | --- |
| 位域 | 描述 |
| Ant Num | 4bit。LTE符号对应的天线数，0-7 |
| RF Num | 12bit，LTE符号所在帧号 |
| SF Num | 4bit，LTE符号所在子帧号 |
| Symbol | 4bit，LTE符号对应的符号编号 |
| Reserved | 保留 |

payload部分

##### RPE\_PUSCH\_PKT\_SUB\_TYPE

上行数据包。消息格式如图5-15所示。保留位为32位，消息头包含status，configuration及gain部分，之后为payload。

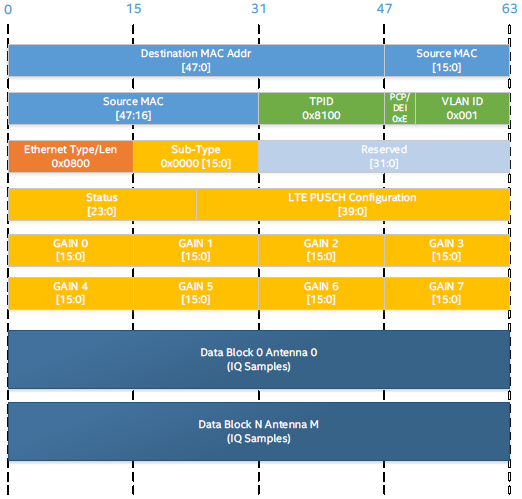


图5-15 PUSCH数据包格式

方向：CPU<-FPGA

消息组成:

[reserved][pusch\_hdr][payload]

消息内容:

pusch\_hdr部分



图5-16 pusch\_hdr部分

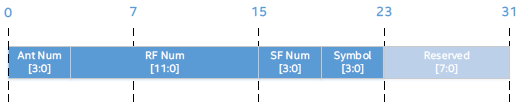


图5-17 pusch\_hdr格式

表5-11 status功能描述

|  |  |
| --- | --- |
| 位域 | 描述 |
| Status[0] | 保留 |
| Status[1] | 若为真，表示为特殊子帧；第一阶段不支持特殊子帧 |
| Status[23:2] | 保留 |



图5-18 configuration的格式

表5-12 configuration的描述

|  |  |
| --- | --- |
| 位域 | 描述 |
| RB数目 | PUSCH数据包的RB数目 |
| Rsvd1[0] | 1bit，保留 |
| Rsvd2[1:0] | 2bit，保留 |
| Ant Num | 数据包包含的天线数，0-7。有效输入为1,3,7,分别表示2,4,8根活动天线 |
| Ant start | 起始天线号。例如，Ant Num为2，Ant Start位2，表示数据包对应天线2和3.天线数据须在数据中顺次排列。 |
| RF Num | LTE包对应的帧号 |
| SF Num | LTE包对应的子帧号 |
| Symbol Num | LTE包对应的当前符号编号 |

表5-13 Gain部分的描述

|  |  |
| --- | --- |
| 位域 | 描述 |
| GAIN[8:0] | AGC输出 |
| GAIN[9] | 保留 |
| GAIN[15:10] | AFFT模块指数 |

payload部分

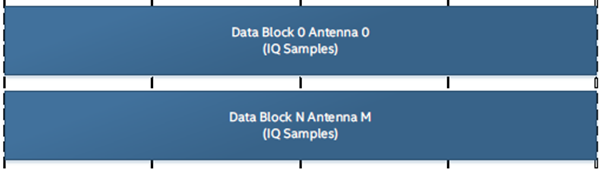


图5-19 PUSCH数据包payload部分

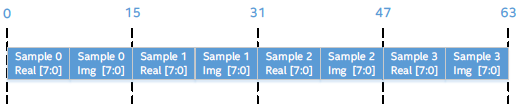


图5-20 data block格式

##### RPE\_PRACH\_PKT\_SUB\_TYPE

上行数据包。消息格式如图5-21所示。消息头包含status，configuration部分，之后为payload。

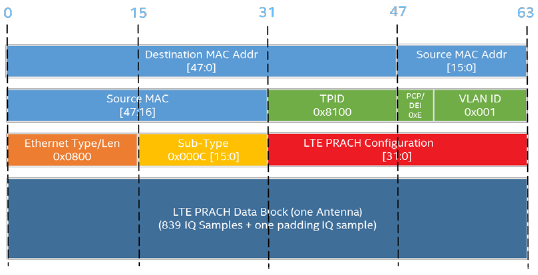


图5-21 PRACH数据包格式

方向：CPU<-FPGA

消息组成:

[prach\_hdr][payload]

消息内容:

prach\_hdr部分



图5-22 prach\_hdr部分

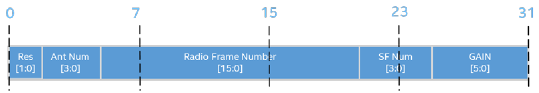


图5-23 configuration格式

表5-14 configuration功能描述

|  |  |
| --- | --- |
| 位域 | 描述 |
| reserved | 2bit，保留 |
| Radio Frame Number | 16bit，表示该天线数据对应的帧号 |
| SF Num | 4bit，表示该天线数据对应的子帧号 |
| Ant Num | 4bit，该数据对应的天线数 |
| Gain | FFT移位数；有符号数，负数表明右移，正数表明左移。为使输出数据符合需要的动态范围，需要通过相反的操作做补偿。例如，当gain为负数时，数据块中的数据，需要左移，移位数为gain的绝对值。 |

payload部分

##### RPE\_LTE\_PDCCH\_PKT\_SUB\_TYPE

下行数据包。消息格式如图5-24所示。消息头包含status，configuration部分，payload包含DRBMAP Block，URBMAP，PDCCH Block和PHICH Block。

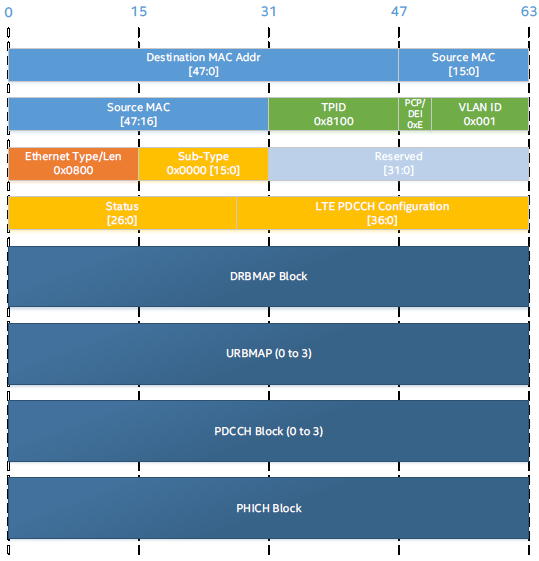


图5-24 PDCCH数据包格式

方向：CPU->FPGA

消息组成:

[reserved][pdcch\_hdr][payload]

消息内容:

pdcch \_hdr部分



图5-25 pdcch\_hdr部分

表5-15 status描述

|  |  |
| --- | --- |
| 位域 | 描述 |
| Status[0] | 保留 |
| Status[1] | 若为真，表示该子帧为特殊子帧 |
| Status[2] | 若为真，表示该子帧包含PBCH信道 |
| Status[3] | 若为真，表示该子帧包含PSS信道 |
| Status[26:4] | 保留 |

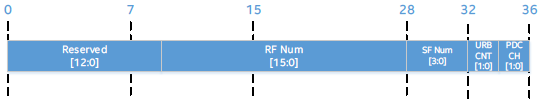


图5-26 configuration格式

表5-16 configuration功能描述

|  |  |
| --- | --- |
| 位域 | 描述 |
| reserved | 12bit，保留 |
| RF Number | 16bit，表示该数据包对应的帧号 |
| SF Num | 4bit，表示该数据包对应的子帧号 |
| URB CNT | 2bit，该数据包中 URBMAP block的数目，0-3 |
| PDCCH CNT | 2bit，表示数据包中PDCCH块的数目，0-3 |

payload部分

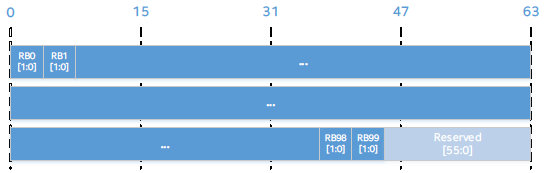


图5-27 DRBMAP block格式

表5-17 DRB译码信息

|  |  |
| --- | --- |
| RB值 | 描述 |
| 00 | RB未使用 |
| 01 | RB使用 |
| 10 | 保留 |
| 11 | 保留 |

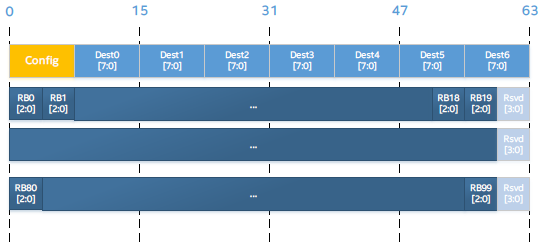


图5-28 URBMAP block格式

表5-18 URBMAP config描述

|  |  |
| --- | --- |
| 位域 | 描述 |
| Config[0] | 保留 |
| Config[1] | 若为真，则该子帧为特殊子帧 |
| Config[2] | 若为真，表示URB block指向该数据包下一帧的一个子帧 |
| Config[6:3] | URB Block的子帧号 |
| Config[7] | 保留 |

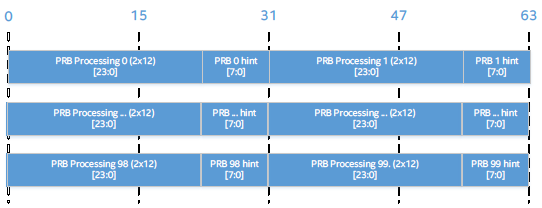


图5-29 PDCCH Block格式



图5-30 PRB processing格式

表5-19 PRB Hint描述

|  |  |  |
| --- | --- | --- |
| 位域 | 值 | 描述 |
| PRB N Hint[7:6] | 4元符号分组方法 | |
| 00 | x0~x11用作RE数据。有三个4元符号，{x0,x1,x2,x3}，{x4,x5,x6,x7}，{x8,x9,x10,x11}.RE映射结果为{map(x0),…,map(x11)} |
| 01 | x0~x7,xx,xx,xx,xx用作RE数据。有两个四元符号{x0,x1,x2,x3}，{x4,x5,x6,x7}。RE映射结果为{0,map(x0),map(x1),0,map(x2),map(x3)}和{0,map(x4),map(x5),0,map(x6), map(x7)} |
| 10 | x0~x7,xx,xx,xx,xx用作RE数据。有两个四元符号{x0,x1,x2,x3}，{x4,x5,x6,x7}。RE映射结果为{map(x0),0,map(x1),map(x2),0,map(x3)}和{map(x4),0,map(x5),map(x6),0, map(x7)} |
| 11 | 假设只有一个PDCCH：x0~x7,xx,xx,xx,xx用作RE数据。有两个四元符号{x0,x1,x2,x3}，{x4,x5,x6,x7}。RE映射结果为{map(x0),map(x1),0,map(x2),map(x3),0}和{map(x4),map(x5),0,map(x6), map(x7),0}。 |
| PRB N Hint[5:4] | 处理第一个四元符号 | |
| 00 | 由0取代 |
| 01 | 采用QPSK |
| 10 | 解压缩PHICH的x0,x1,x2,x3后，将PHICH读指针移至下一行 |
| 11 | 解压缩PHICH的x0,x1,x2,x3后，将PHICH读指针移至下一行 |
| PRB N Hint[3:2] | 处理第二个四元符号 | |
| 00 | 由0取代 |
| 01 | 采用QPSK |
| 10 | 解压缩PHICH的x0,x1,x2,x3后，将PHICH读指针移至下一行 |
| 11 | 解压缩PHICH的x0,x1,x2,x3后，将PHICH读指针移至下一行 |
| PRB N Hint[1:0] | 处理第三个四元符号（若PRBx Hint[7:6]=00） | |
| 00 | 由0取代 |
| 01 | 采用QPSK |
| 10 | 解压缩PHICH的y0,y1,y2,y3后，将PHICH读指针移至下一行 |
| 11 | 解压缩PHICH的y0,y1,y2,y3后，将PHICH读指针移至下一行 |

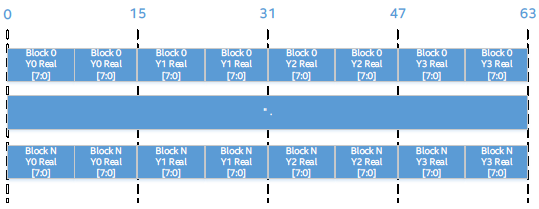


图5-31 PHICH Block格式

##### RPE\_LTE\_PBCH\_PKT\_SUB\_TYPE

下行数据包。消息格式如图5-32所示。保留位为32位，之后为payload。

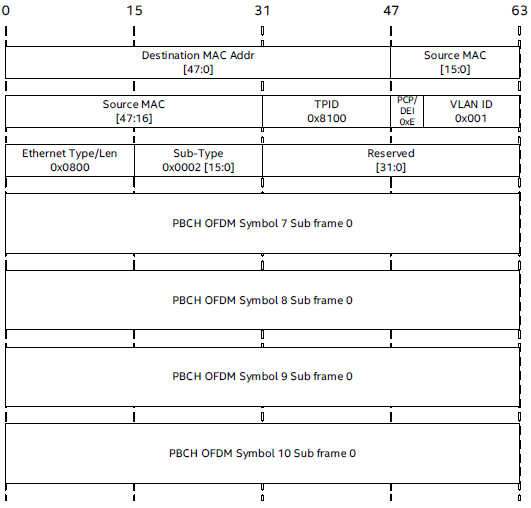


图5-32 PBCH数据包格式

方向：CPU->FPGA

消息组成:

[reserved][payload]

消息内容:

payload部分

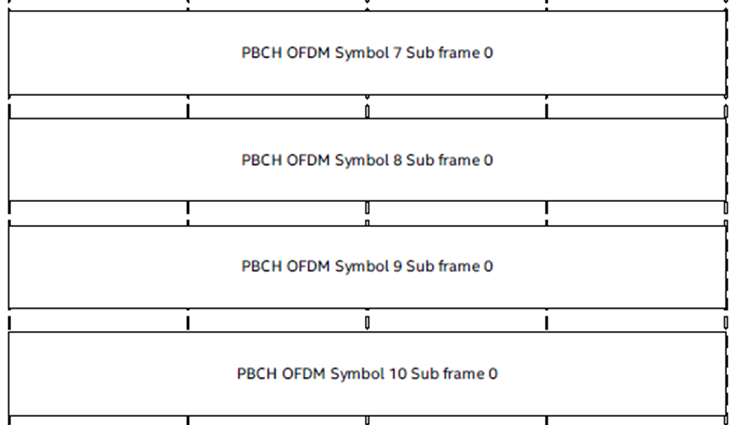


图5-33 payload部分

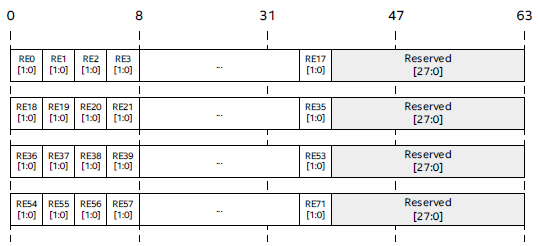


图5-34 PBCH符号格式

##### RPE\_LTE\_PSS\_PKT\_SUB\_TYPE

下行数据包。消息格式如图5-35所示。RPE包头后为32位的保留位，之后为payload。

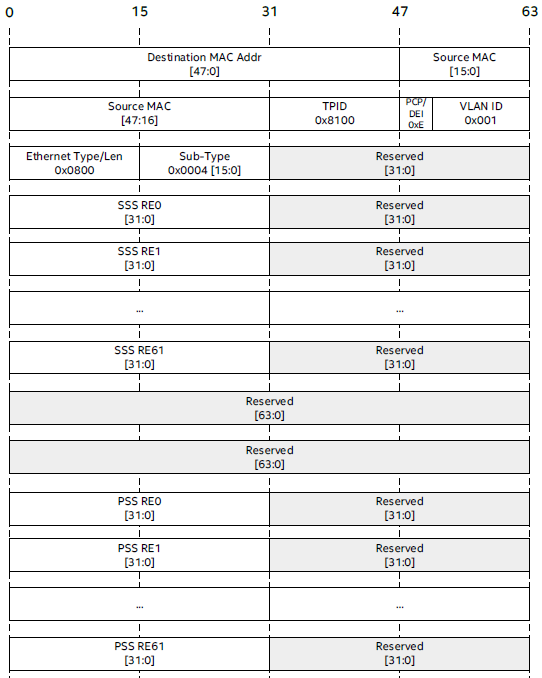


图5-35 PSS数据包格式

方向：CPU->FPGA

消息组成:

[reserved][payload]

消息内容:

payload部分

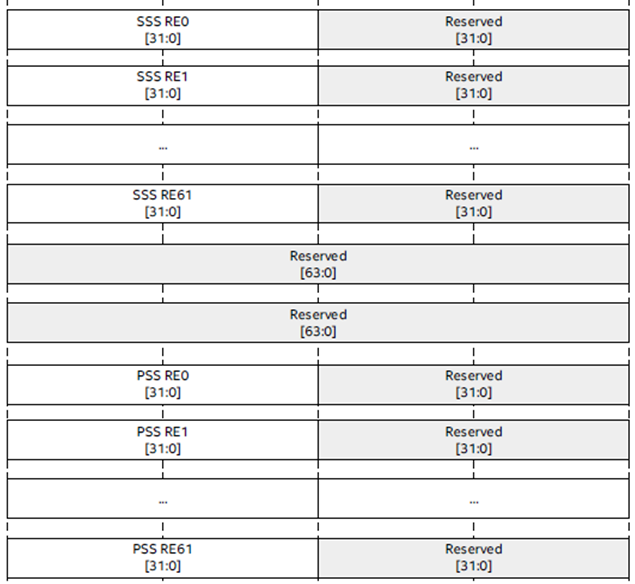


图5-36 PSS信号payload部分

##### RPE\_LTE\_RS\_PKT\_SUB\_TYPE

下行数据包。消息格式如图5-37所示。RPE包头后为24位的保留位，之后为数据包头（Ant Bit Mask）和payload。

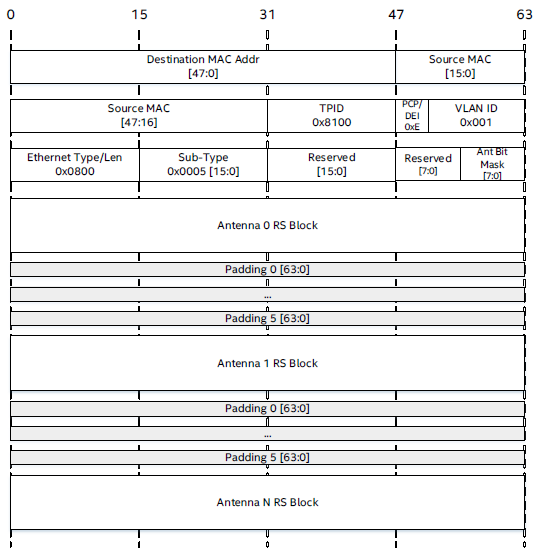


图5-37 RS数据包格式

方向：CPU->FPGA

消息组成:

[reserved][rs\_hdr][payload]

消息内容:

rs\_hdr部分



图5-38 rs\_hdr部分

表5-20 Ant Bit Mask描述

|  |  |
| --- | --- |
| 掩码值 | 活动天线数 |
| 0x01 | 天线0为唯一活动天线 |
| 0x03 | 天线0和1为活动天线 |
| 0x0F | 天线0-3为活动天线 |
| 0xFF | 天线0-7为活动天线 |

payload部分

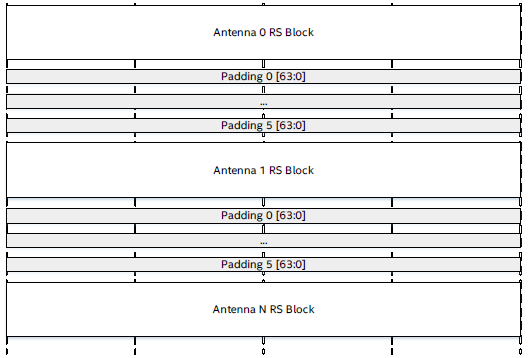


图5-39 RS数据包的payload

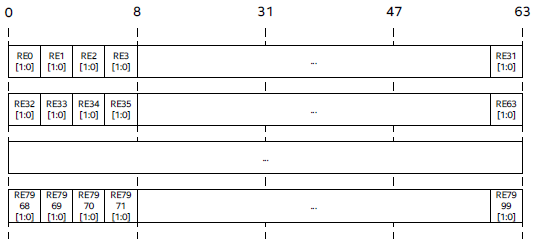


图5-40 RS Block格式

### 线程及触发关系

详见《code研读笔记》。

### Framework

Framework主要基于基带单元池（Base Band Unit Pooling）在多核处理器上完成生成任务和调度任务。它为任务提供资源处理、缓存缺失或延迟处理，同时最小化调度开销。基带单元池框架包含三个组件：基带单元池任务、基带单元池事件、基带单元池缓冲区管理。

基带单元池任务：解决无线接入网数据单元密集的计算，并提供了并行的方法和基于信号流图的资源扩展方法。

基带单元池事件：解决无线接入网控制单元密集的数据包处理，并提供了并行的方法和基于DPDK事件设备的资源扩展方法。

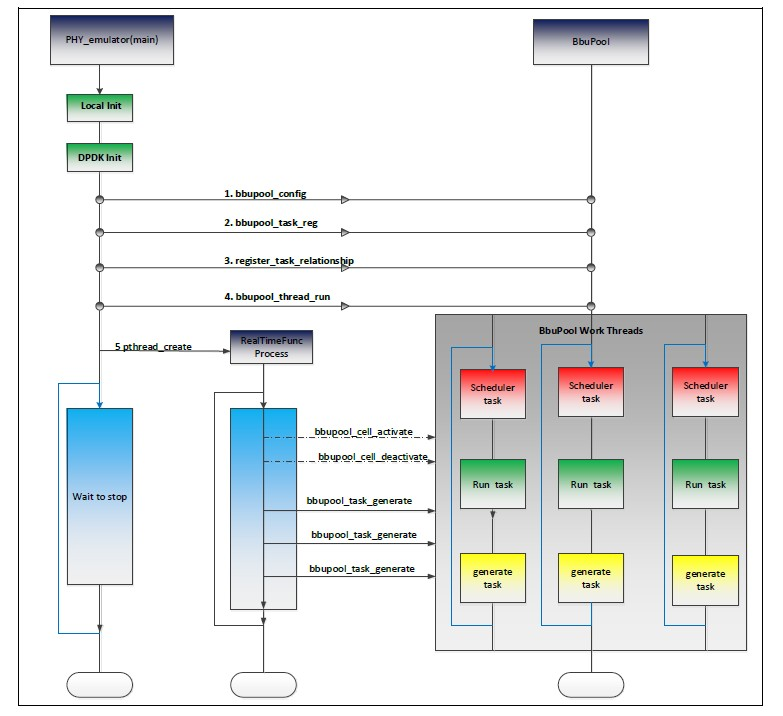
基带单元池缓冲区管理：提供一种有效的缓冲区管理方法用于无线接入网控制单元的应用程序。

注释：此部分理解有限，请具体参见Framework相关原始文档

**2.3.4.1 任务调度过程**

Framework调度机制主要通过bbupool调度完成，详细过程如图1所示。根据图1，以及对代码（bbupool\_rt\_thread.c/bbupool\_scheduler.c）的梳理，对任务调度过程作简单分析。

首先通过PHY完成本地以及DPDK的初始化、参数配置、注册任务关系、分配任务优先级等工作。然后通过lte\_rt\_thread函数（即实时线程）生成任务（bbupool\_task\_generate）以及完成小区的建立和小区的删除，此后再由bbupool调度任务。Bbupool首先调度由RT线程生成的任务，再运行此任务，然后再生成下一个任务。Bbupool以循环的模式调度任务、运行任务、以及生成任务，直到没有任务生成，即完成所有的任务。



**图1 任务调度流程图**

### 代码结构

物理层代码文件夹用途如下：

* Source/ ：LTE源文件夹结构

Common/ ：通用定义与函数

Phy/ ：LTE物理层源代码

Api/ ：定义MAC与PHY之间的API，能实现发送和接收API的功能

Mac2phy ：接收和处理MAC发给PHY的API的函数和实体

Phy2mac ：产生API的函数和实体以从PHY发给MAC

Dl\_mod：下行调制任务的函数与结构

Ul\_demod：上行解调任务的函数与功能

fec\_common：FEC编码解码通用的函数与结构

fec\_enc：FEC编码器的函数与结构

fec\_dec：FEC解码器的函数与结构

fec\_wck：FEC瓦德尔溪加速器的函数与结构

phystats：PhyStatst调试效用的函数与结构

tables：PHY库所用的表格与数组

utils：PHY库所用的效用函数

threads：LTE PHY线程

auxlib：支持PHY的辅助函数

cline：解析通过命令行传过来的参数并依据这些参数配置应用程序的效用

cmgr：命令管理接口函数。包含运行通过用户在应用控制台能敲的命令实现的预注册函数的效用

phycfg：用从phycfg.xml文件读出来的静态变量来初始化L1的XML配置文件接口函数

phydi：初始化和销毁Timer/Radio/其它接口的物理层数据接口函数，这些接口能提供TTI和PHY接收/发送的有效IQ数据

radio：无线接口层。它使接口生效，用DPDK和Ferry Bridge库在基带单元和远程遥控处理单元（FPGA）传递IQ数据

sys：有和操作系统与L1用户空间应用程序交互的效用的系统接口函数

timer：在L1的周期间隔内产生中断请求的Timer接口函数。这用来调试和介绍L1软件

Wls：无线子系统接口。该模块在MAC和PHY之间提供共享内存接口，来交换API

Test：LTE测试应用程序源文件夹结构

Config：解析配置文件，用下行和上行测试配置测试应用程序的函数

Testapp：testapp应用程序的函数

Testmac：testmac应用程序的函数

Utils：host应用程序使用的效用

Wireless\_channel：用来添加信道模型、噪声、时延频偏等

目录结构及编译如下图所示，其中，do.sh、run.sh的使用区别见《开发环境配置文档》[1] 4.2节。



图 2‑1 代码目录结构

#### 物理层CPU软件代码结构

Timer模式代码结构如下：



图 2‑2 timer模式代码结构

Radio模式代码结构如下图所示，其中，lte\_bs\_mac2phy\_api\_recv\_scan函数接收mac发来的api消息并进行了处理。Radio模式有三个中断，lte\_radio模式只注册了两个中断。包含api、dl、ul、prach和fft五个主线程。

图 2‑3 radio模式代码结构

BBU Pooling代码结构如下图所示：

图 2‑4 BBU Pooling代码结构

#### CPU\_PHY和FPGA通信的代码结构

CPU\_PHY和FPGA通信的代码结构如下图所示：

图 2‑5 CPU\_PHY和FPGA通信的代码结构

### API时序

将API时序分为两部分：一部分是上行处理时序，另一部分是下行处理时序。其中WLS（共享内存）是MAC和PHY的公用部分，作为MAC和PHY的信息传递枢纽。

上行处理是提前一帧调度。图1中调度指MAC在N-1帧时通知PHY接收第N帧的数据，处理后将其存入WLS中。在第N帧开始，Ferrybridge做FFT/IFFT等处理。CPU大约从第四个符号开始做其他相应处理。然后CPU将处理好的数据存入自己的buffer中，在第N+2帧头部放入WLS中。具体见图1所示。

下行处理是提前两帧调度。MAC在N-2帧时调度第N帧的数据，处理后将其存入WLS中。PHY在N-2帧尾部，大约在提前三个符号的时刻开始解析数据，数据解析并不能占据这三个符号的所有时刻，数据解析结束紧接着交由CPU处理，然后再由Ferrybridge处理。在第N-1帧尾部留极短一点时间给RRU，使其通过空口将信息传递给UE，最后在第N帧时实现空口对齐。具体见图2所示。

图1 上行处理时序



图2 下行处理时序

# 物理层测试

本节主要描述用来测物理层CPU软件功能和性能的测试框架等。Testapp与testmac的代码处理流程及测试实例见《测试方法说明》文档。其中，Testmac物理层是正式代码，testapp把物理层的处理流程重新组织调用顺序（在lte\_bs\_host\_app\_api\_proc.c文件里），只是函数实现。

## 测试架构

物理层CPU软件的功能和性能可以通过testapp和testmac以及端到端方式等来进行测试。其中，testapp和testmac这两个应用程序的主要目的是：

1. 产生发送到PHY的API和接收PHY发来的API。
2. 上行测试开始的时候提供参考IQ数据，下行测试结束的时候接收从PHY发来的IQ数据。
3. 提供了多个测试需要的文件，包括xml格式的配置文件、参考上行IQ数据（它在上行测试中作为PHY输入）、参考上行IQ数据、参考上行有效载荷（指在一个数据包或者其它传输单元中运载的基本必要数据）文件（它有PUSCH 、PUCCH 、 PRACH 、SRS等的所有有效载荷）。
4. 在测试结束的时候，两个应用程序都比较了参考下行IQ数据和上行有效载荷，并产生一个通过或者失败的报告。

我们有五种不同类型的测试：

DL：只有下行

UL：只有上行

FD：全双工（上行和下行）

RCTDL：RCT下行

RCTUL：RCT上行

## 测试方法

### Testapp

本工具和PHY一起编译，在应用程序的入口点被调用。如图2-1所示，它生成API、发送给PHY、等待PHY完成并从PHY接收上行API，然后继续运行PHY。它是单线程的，把处理流程串在一起，不测试PHY的实时方面，只测试功能和位准确性。用这个工具，我们可以一次进行上行或者下行测试，PHY的函数逐一运行，没有创建和L1 PHY应用程序平行的线程。

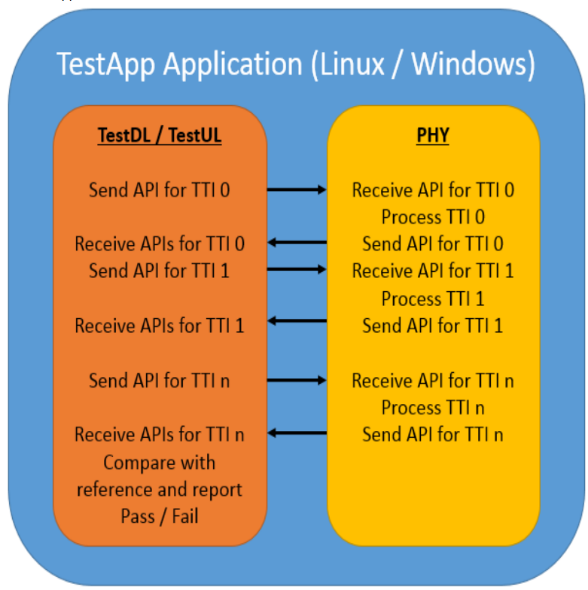


图 3‑1 TestApp工作流程

本工具能够在Windows环境（visual studio 2013 和 Parallel studio）或者Linux环境（icc 17.0）下被建立，源代码目录是：**source\test\testapp**，编译文件和工程目录是：**build\testapp**，编译过程完成之后，应用程序二进制文件被放在**lte\_host\test\lte\_testapp**目录下。

1. 在shell命令行下运行本应用程序，可以用以下命令行参数来启动**lte\_host\test\lte\_testapp**应用程序：**lte\_testapp [测试序号] [上行/下行，0表示下行，1表示上行] [载波数，取值为1~4] [载波配置文件位置]**。例如，输入**lte\_testapp 2 0 1 dl/2/dl\_testconfig\_tst2.cfg**来运行下行单载波，由dl/2/dl\_testconfig\_tst2.cfg配置的2号测试例。
2. 在**lte\_host\test**目录下也有一个可以运行这些测试的带有上述参数的脚本文件，windows环境下用c\_master.bat脚本或者在linux环境下用c\_master.sh脚本，脚本运行的语法是：**c\_master.bat/c\_master.sh test\_choice test\_num**  **test\_bandwidth**

* **test\_choice**：all/dl/ul/rctdl/rctul；
* **test\_num[optional]**:[1-n]，n表示testcfg文件中的最后一个测试序号,只有当test\_choice是rctul的时候才需要设置此参数，且只有需要运行一个测试的时候此参数才被用；
* **test\_bandwidth [optional]**：3/5/10/15/20，只有当test\_choice是rctul的时候才需要设置此参数，如果没有明确定义，则所有的带宽都会默认运行。

例如，可以在linux环境下，输入**c\_master.sh rctul 1 20**来运行20MHz带宽时rctul类型下的1号测试例。

### Testmac

本工具作为一个独立的应用程序，能够像真实的MAC一样和PHY一起运行，这个应用程序模拟了发消息的过程，目的是测试位准确性功能和PHY的实时运行。我们可以利用它进行下行、上行和全双工测试，可以运行 L1 PHY应用程序（l1app），其中l1app创建了它自己的线程以尝试和并行化L1的处理。

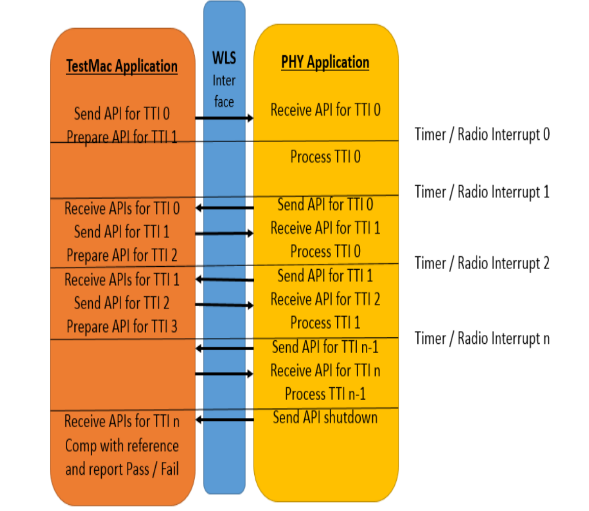


图 3‑2 TestMac工作流程

本工具只能在Linux环境（使用icc 17.0）下被建立，它的源代码目录是： **source\test\testmac**，编译文件和工程目录是：**build\testmac**，编译过程完成之后，应用程序二进制文件被放在**lte\_host\test\lte\_testmac**目录下。

运行这个应用程序的步骤如下：

1. 首先启动l1app应用程序：运行**lte\_phy\l1.sh –e**，（其中，-e选项表示Timer模式，不加-e表示Radio模式）；然后，启动**lte\_testmac\l2.sh**脚本文件。
2. 一旦应用程序启动，将看到**TESTMAC>**提示符，可以用以下命令行运行单元测试：

**run testtype testnum**

* **Testtype**：0（DL），1（UL），2（FD）；
* **Testnum**：测试序号，如果该参数为空，将运行所选的testtype下的所有测试。

1. 如果用户想要运行更多的子帧或者改变模式或者改变测试的TTI间隔，那么，phystart命令可以这样用：**phystart mode interval num\_tti**。

* **Mode**：0（Radio）或者1（Timer）；
* **Interval**：timer模式下的TTI持续时间，单位是ms，它1ms运行一次并以1ms开始；
* **num\_tti**是运行测试的TTI的总个数，若是0，则由测试配置文件定义运行的子帧个数；若非0 ，就运行num\_tti个子帧；如果num\_tti的值大于配置文件中的子帧个数，那么配置将会一直重复，直到测试结束。

在testmac开始的时候相关配置文件配置的参数是（**phystart 1 10 0**），即timer模式，10ms TTI间隔，TTI总个数在每个测试配置文件中确定。一旦用户在L2控制台上手动写出此phystart命令，所有后来的测试就会用这个phystart设置，直到用户再次修改或者testmac重启。

1. 如果用户想要运行cfg后缀文件中编写的一套测试，例如test\_customer.cfg，可以写下面的命令：**./l2.sh – testfile = tests\_customer.cfg**，这将运行配置文件中列出的所有的测试。

### Testapp与Testmac区别

Testapp和Testmac原理如下图所示：

图 3‑3 TestApp和TestMac工作流程

其中，testphy是为了测试物理层功能，而对物理层代码进行封装组成的物理层流程函数，体现在lte\_bs\_host\_app\_api\_proc.c文件中；而testmac中的phy为真实的物理层代码。

testapp与testmac的区别如下：

1. Testapp只测试功能和位准确性，不测试PHY的实时性，testmac测试测试位准确性功能和PHY的实时运行；
2. Testapp与testmac一起组成一个应用程序，共用一个main函数；而testmac作为一个独立的应用程序，能够像真实的MAC一样和PHY一起运行；
3. 对于testapp，只能运行dl, ul, rctdl和rctul测试例[2]；对于testmac，只能运行dl,ul和fd测试例[3]。

### 端到端测试

可以用一个测试设备模仿LTE UE来对L1应用程序进行端到端的测试。该测试的流程及组成如下图所示：

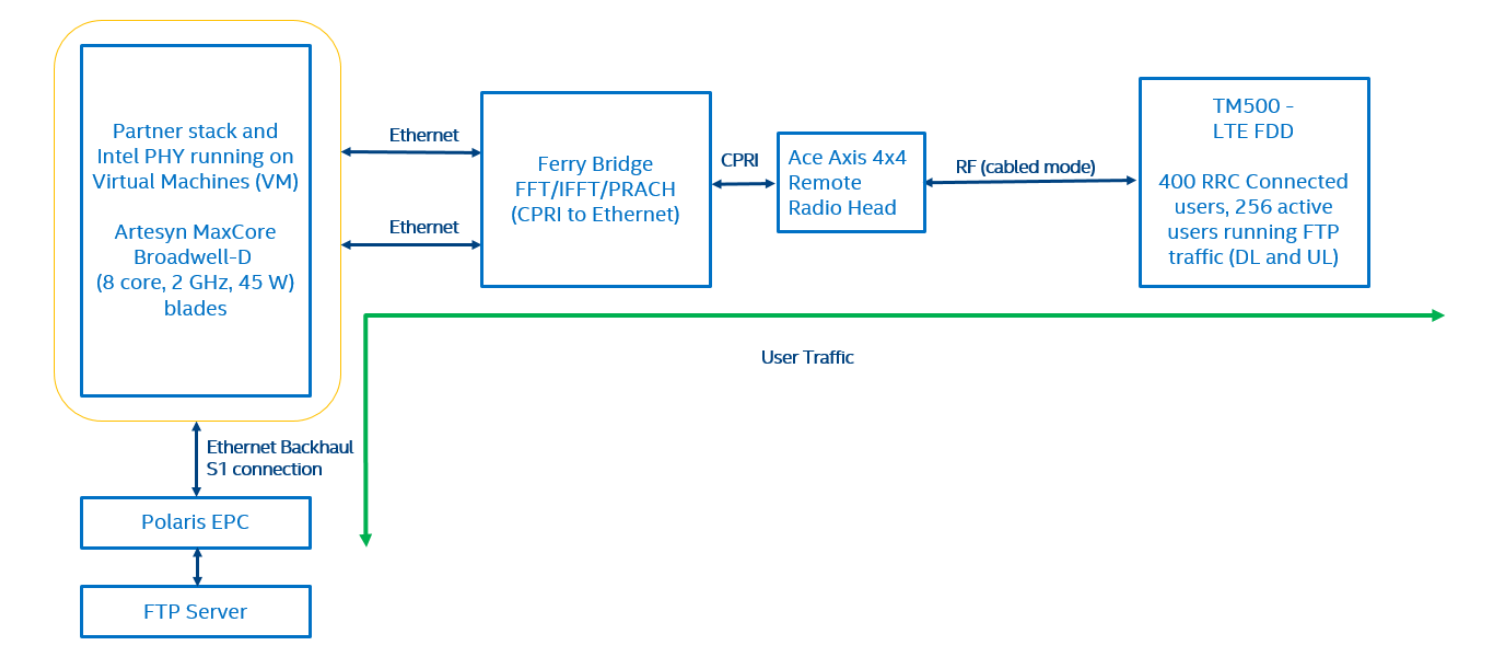


图 3‑4 端到端测试流程及组成

以下是是组成成分的详细介绍：

* **FTP Server**：这是一个能承载256个同步FTP双向流的i7机器，可以运行Debian&reg操作系统。
* **Polaris EPC**：这是一个携带来自北极星网络的EPC（增强包核心）的i7机器。[*http://www.polarisnetworks.net/*](http://www.polarisnetworks.net/)
* **Artesyn MaxCore chassis and blade：**多用途和密集计算媒体平台。[*https://www.artesyn.com/computing/products/product/max-core/*](https://www.artesyn.com/computing/products/product/max-core/)
* **Broadwell-D：**Intel® Xeon® Processor D-1500 产品簇。[*http://www.intel.com/content/www/us/en/embedded/products/broadwell-de/documentation.html*](http://www.intel.com/content/www/us/en/embedded/products/broadwell-de/documentation.html)
* **Ferry-Bridge：**负责从以太网到CPRI格式的转换和一些LTE前端处理的FPGA。
* **Ace Axis Remote Radio Head (Advanced Radio Tester):** 支持宽带LTE带宽的LTE4X4的射频拉远头。[*http://aceaxis.co.uk/website/resources/*](http://aceaxis.co.uk/website/resources/)
* **TM500：**LTE网络测试。[*http://cobhamwireless.com/product/wireless-network-traffic-emulation/tm500-network-tester/*](http://cobhamwireless.com/product/wireless-network-traffic-emulation/tm500-network-tester/)

经过以上步骤，我们已经验证了拥有如下配置的LTE端到端连接：

* 20MHz FDD， 2X2 MIMO，140Mbps下行和40Mbps上行
* 20MHz FDD， 4X4 MIMO（下行支持四层），260Mbps下行和40Mbps上行
* 40MHz FDD， 拥有2个聚合的载波的2X2 MIMO，280Mbps下行和40Mbps上行
* 400个连接了RRC的用户，能保持其中的256个连接，运行FTP双向流量，能达到下行280Mbps和40Mbps上行的吞吐量。
* 在上行和下行方向上，每个小区每个TTI有8个用户
* 每个TTI内达到32个PUCCH（包含周期性的SR，CQI，RI和PMI）

# 集成测试

## 测试环境搭建

Radio模式集成测试需要的设备：X86服务器+Ferrybigh蓝盒子+烽火RRU

4.1.1 测试设备连接线的说明



* 连接线①：“测试”PC机通过访问的形式连接服务器
* 连接线②：用光纤连接服务器和蓝盒子上的光口
* 连接线③：用光纤连接蓝盒子和烽火RRU上的光口
* 连接线④：用网线连接烽火RRU和“启动”笔记本电脑上的网口，以及用串口线连接串口

4.1.2 测试设备的启动

测试设备启动的先后顺序如下：（默认服务器正常运作）

第一步：启动烽火RRU，此时RRU外盒插网线处附件的红灯亮，RRU内侧有一个红色小灯亮；

第二步：间隔第一步约一分钟后，再启动蓝盒子，此时RRU外盒插网线处附件的红灯灭，RRU内侧有一个红色小灯亮。

第三步：通过putty软件输入建立通信的命令。

第四步：通过网口软件设置信息；

第五步：运行测试代码。

第六步：RRU断电，蓝盒子断电；

第七步：RRU上电，间隔约1分钟，蓝盒子上电；

第八步：通过putty软件输入建立通信的命令。

第九步：通过网口软件设置信息；

第十步：观察RRU内侧之前亮红色的小灯变成绿色，说明启动成功。

4.1.3 代码调试步骤

4.1.3.1编译

遇到以下几种情况时，在运行代码前需要编译代码：

（1）若修改Framework部分函数，则需要先编译Framework部分代码；再编译物理层，如（3）所示。

（2）若修改Ferrybridge部分函数，则需要先编译Ferrybridge部分代码；再编译物理层，如（3）所示。

（3）若修改物理层内部代码，则需要全局编译。

4.1.3.2 运行

运行代码需要以下几步：

第一步：启动物理层：

第二步：启动核心网：

第三步：启动协议栈：./l2.sh

**附录**（图片待后续插入）

**1 Radio模式测试环境与操作**

1.1 测试设备的启动

1.1.1 测试设备启动的条件

测试设备需要启动的两种情况：（默认服务器正常运作）

（1）若上一次测试将烽火RRU和蓝盒子断电，则本次测试需要重新启动；

（2）运行物理层代码时，提示烽火RRU和蓝盒子连接不成功，则需要重启烽火RRU和蓝盒子。

（图1）

1.1.2 测试设备启动的顺序

测试设备启动的先后顺序如下：（默认服务器正常运作）

第一步：启动烽火RRU；

第二步：间隔第一步约一分钟后，再启动蓝盒子。

注意：连接正常的判断条件：烽火RRU的告警指示灯由红色熄灭为标志；

（图2）

第三步：通过putty软件输入建立通信的命令。

提示：在该界面出现：config FPGA1 …….时，输入命令：bspLoadLmt()；

（图3）

第四步：通过网口软件设置信息；

（图4）（图5电脑ip）

第五步：运行测试代码。

1.2 代码调试步骤

1.2.1 编译

遇到以下几种情况时，在运行代码前需要编译代码：

（1）若修改Framework部分函数，则需要先编译Framework部分代码；再编译物理层，如（3）所示。

编译路径： /home/自己的目录/DEMO/SRC/framework/bbupool

编译脚本：Makefile

执行步骤：

第一步：make clean （清空之前的编译）

第二步：make all （编译现在的所有Framework部分代码）

（2）若修改Ferrybridge部分函数，则需要先编译Ferrybridge部分代码；再编译物理层，如（3）所示。

编译路径： /home/自己的目录/DEMO/SRC/ferrybridge/lib

编译脚本：Makefile

执行步骤：

第一步：make clean

第二步：make all

（3）若修改物理层内部代码，则需要全局编译。

编译路径： /home/自己的目录/DEMO/SRC/build/l1app

编译脚本：Makefile 、build.sh

执行命令：./build.sh

1.2.2 运行

运行代码需要以下几步：

第一步：启动物理层：

路径：/home/自己的目录/DEMO/SRC/lte\_phy

执行命令：./l1.sh

脚本l1.sh的说明：定义DPDK、WLS、SDK、以及各部分代码的路径，按Timer/Radio模式获取相应的配置文件，完成物理层的启动。（其中Timer模式对应配置文件：phycfg\_timer.xml； Radio模式对应配置文件：phycfg.xml）

第二步：启动核心网：

路径：/home/lte\_realmac

执行命令：./om

第三步：启动协议栈：./l2.sh

路径：/home/lte\_realmac

执行命令：./l2.sh

注意：协议栈启动通常在物理层之后，大致时间是物理层代码运行到标志为：2and….时；核心网的启动时间没过多要求，只需要在物理层之后，协议栈之前即可。

（图5）