数据处理说明文档（服务器）

# 1 数据处理流程

图 1.1 数据处理流程

## 1.1 发送端

|  |  |  |
| --- | --- | --- |
|  | 处理流程 | 数据 |
| 1 | 根据时频两域资源以及帧结构定义**计算最大传输符号数**，再根据最高CQI反馈值计算最大传输块大小。 | 初始传输块数据：二进制比特流  数据类型：uin8\_t  文档：帧结构定义 |
| 2 | 对每个传输块进行**CRC添加(24A)**，**码块分割**，对每个码块进行**CRC添加(24B)**  问题：协议中的<NULL>比特不太理解，暂设为0 | 码块数据：二进制比特流  数据类型：uin8\_t  文档：3GPP TS 36.212 |
| 3 | 对添加CRC后的码块，进行**Turbo编码** | 码块数据：二进制比特流  数据类型：uin8\_t  文档：3GPP TS 36.212 |
| 4 | 根据协议计算每个码块速率匹配后的长度，据此进行**速率匹配** | 码块数据：二进制比特流  数据类型：uin8\_t  文档：3GPP TS 36.212 |
| 5 | 对于每个流，将**码块级联**起来成为一个传输块，根据CQI反馈值对其进行**符号映射**（由于之前的速率匹配，不管初始传输块多长，符号映射后的符号个数都应该等于（1）中计算得到的最大传输符号数） | 传输块数据：定点数  数据类型：int  文档：3GPP TS 36.211 |
| 6 | 根据帧结构定义进行**组包**，添加符号头和子帧头 | 包数据：定点数  数据类型：int  文档：帧结构定义 |

## 1.2 模拟通过信道

|  |  |  |
| --- | --- | --- |
|  | 处理流程 | 数据 |
| 1 | 信道矩阵(R, L, C, S)与调制后符号(L, C, S)相乘（对于每个资源元素，信道矩阵维数为(R, L)，发送信号维数为(L, 1)，接收信号为(R, 1)）得到(R, C, S)的接收信号 | 接收信号：复数，浮点数  数据类型：lapack\_complex\_double |
| 2 | 对接收信号加上高斯白噪声  问题：信噪比 | 接收信号：复数，浮点数  数据类型：lapack\_complex\_double |

## 1.3 接收端

|  |  |  |
| --- | --- | --- |
|  | 处理流程 | 数据 |
| 1 | 根据子帧头和符号头对收到的包进行**解包**，定点数转为浮点数 | 解包后的数据：复数，浮点数  数据类型：lapack\_complex\_double  文档：帧结构定义 |
| 2 | 解包后的数据即接收信号，再根据导频进行**信道估计** | 信道矩阵：复数，浮点数  数据类型：lapack\_complex\_double  文档：程序说明报告-信道估计 |
| 3 | 根据估计出来的信道进行**信号检测**，将检测后的信号估计值分成8个流 | 信号估计值：复数，浮点数  数据类型：lapack\_complex\_double  文档：程序说明报告-信号检测 |
| 4 | 对每个流进行**解符号映射**（调制阶数由CQI确定） | 后验信息：浮点数  数据类型：double |
| 5 | 对每个流进行**解码块级联**，分为多个码块；对于每个码块，进行**解速率匹配**和**Turbo译码**  问题：码块分割参数暂时设为已知，实际通信如何获取码块分割参数 | 解速率匹配后数据：浮点数  数据类型：float  译码后数据：二进制比特流  数据类型：uin8\_t |
| 6 | 对于每个流，先对每个码块进行**CRC校验(24B)**，如果有错，该流直接丢掉；如果没错，再对传输块进行**CRC校验(24A)**，以免漏检 | 校验后数据；二进制比特流  数据类型：uin8\_t |

## 1.4 系统配置

#### (1) Ubuntu 16.04 LTS

## (2) Intel® Parallel Studio XE

# 2 模块函数说明

## 2.1 CRC添加

**函数：** srslte\_crc\_init

**功能：**初始化CRC参数和表

**语法：**int srslte\_crc\_init(srslte\_crc\_t \*h, uint32\_t srslte\_crc\_poly, int srslte\_crc\_order);

**包含文件：**crc.h

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| uint32\_t | srslte\_crc\_poly | 1 | CRC版本 |
| int | srslte\_crc\_order | 1 | CRC长度 |

**输出参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| srslte\_crc\_t \* | h | 1 | CRC参数 |

**返回值：**初始化状态

**函数：**srslte\_crc\_attach

**功能：**添加CRC

**语法：**uint32\_t srslte\_crc\_attach(srslte\_crc\_t \*h, uint8\_t \*data, int len);

**包含文件：**crc.h

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| srslte\_crc\_t \* | h | 1 | CRC参数 |
| int | len | 1 | 输入码长 |

**输出参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| uint8\_t \* | data | len + crc\_order | 输出比特流 |

**返回值：**CRC比特流

## 2.2 码块分割

**函数：**srslte\_cbsegm

**功能：**码块分割

**语法：**int srslte\_cbsegm(srslte\_cbsegm\_t \*s, uint32\_t tbs);

**包含文件：**cbsegm.h

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| uint32\_t | tbs | 1 | 传输块码长 |

**输出参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| srslte\_cbsegm\_t \* | s | 1 | 码块分割参数 |

**返回值：**码块分割状态

## **2.3 Turbo编码**

**函数：**srslte\_tcod\_init

**功能：**初始化编码器

**语法：**int srslte\_tcod\_init(srslte\_tcod\_t \*h, uint32\_t max\_long\_cb);

**包含文件：**turbocoder.h

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| uint32\_t | max\_long\_cb | 1 | 最大输入码长 |

**输出参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| srslte\_tcod\_t \* | h | 1 | 编码器参数 |

**返回值：**初始化状态

**函数：** srslte\_tcod\_encode

**功能：**Turbo编码

**语法：**int srslte\_tcod\_encode(srslte\_tcod\_t \*h, uint8\_t \*input, uint8\_t \*output, uint32\_t long\_cb);

**包含文件：**turbocoder.h

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| srslte\_tcod\_t \* | h | 1 | 编码器参数 |
| uint8\_t \* | input | max\_long\_cb | 输入信息 |
| uint32\_t | long\_cb | 1 | 输入码长 |

**输出参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| uint8\_t \* | output | 3 \* max\_long\_cb + 12 | 输出信息 |

**返回值：**编码状态

**函数：**srslte\_tcod\_free

**功能：**释放编码器内存

**语法：**void srslte\_tcod\_free(srslte\_tcod\_t \*h);

**包含文件：**turbocoder.h

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| srslte\_tcod\_t \* | h | 1 | 编码器参数 |

**输出参数：**无

**返回值：**无

## 2.4 速率匹配

**函数：**srslte\_rm\_turbo\_tx

**功能：**速率匹配

**语法：**int srslte\_rm\_turbo\_tx(uint8\_t \*w\_buff, uint32\_t w\_buff\_len, uint8\_t \*input, uint32\_t in\_len, uint8\_t \*output, uint32\_t out\_len, uint32\_t rv\_idx);

**包含文件：**rm\_turbo.h

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| uint32\_t | w\_buff\_len | 1 | 循环缓冲器大小 |
| uint8\_t \* | input | 3\*6144+12 | 输入信息 |
| uint32\_t | in\_len | 1 | 输入码长 |
| uint32\_t | out\_len | 1 | 输出码长 |
| uint32\_t | rv\_idx | 1 | 循环缓冲器冗余版本 |

**输出参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| uint8\_t \* | w\_buff | w\_buff\_len(6176\*3) | 循环缓冲器 |
| uint8\_t \* | output | out\_len | 输出信息 |

**返回值：**速率匹配状态

**函数：**srslte\_rm\_turbo\_gentables

**功能：**生成速率匹配查找表

**语法：**void srslte\_rm\_turbo\_gentables();

**包含文件：**rm\_turbo.h

**输入参数：**无

**函数：**srslte\_rm\_turbo\_tx\_lut

**功能：**速率匹配（查找表方式）

**语法：**int srslte\_rm\_turbo\_tx\_lut(uint8\_t \*w\_buff, uint8\_t \*systematic, uint8\_t \*parity, uint8\_t \*output, uint32\_t cb\_idx, uint32\_t out\_len, uint32\_t w\_offset, uint32\_t rv\_idx);

**包含文件：**rm\_turbo.h

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| uint8\_t \* | systematic | 6148 / 8 + 1 | 系统信息（字节） |
| uint8\_t \* | parity | 6148 / 8 + 1 | 校验信息（字节） |
| uint32\_t | cb\_idx | 1 | 交织表索引值 |
| uint32\_t | out\_len | 1 | 输出码长 |
| uint32\_t | w\_offset | 1 |  |
| uint32\_t | rv\_idx | 1 | 循环缓冲器冗余版本 |

**输出参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| uint8\_t \* | w\_buff | w\_buff\_len(6176\*3) | 循环缓冲器 |
| uint8\_t \* | output | out\_len | 输出信息 |

## 2.5 码块级联

无相关函数，直接在相关函数内完成。

## 2.6 加扰

暂未添加，保留。

## 2.7 符号映射

**函数：**QAM\_Modulation

**功能：**调制

**语法：**void QAM\_Modulation(uint8\_t \*inbit, lapack\_complex\_double \*Sig, int SymbolBitN, int SigLen);

**包含文件：**无

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| uint8\_t \* | inbit | SymbolBitN\*SymbNum | 码字，即编码模块的输出 |
| int | SymbolBitN | 1 | 每个符号对应的二进制码字数目 |
| int | SigLen | 1 | 符号数目 |

**输出参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| lapack\_complex\_double\* | Sig | SymbNum | 调制后符号 |

**返回值：**无

## 2.8 组包

暂无相关函数，直接实现。

## 2.9 解包

暂无相关函数，直接实现。

## 2.10 信道估计

**函数：**chestLS\_init

**功能：**DCT信道估计初始化

**语法：**void chestLS\_init(struct chestLS\_t \*chestLS\_p, int CarrierNum, int inter\_freq, int SymbNum);

**包含文件：**无

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| int | CarrierNum | 1 | 子载波数 |
| int | inter\_freq | 1 | 频域插值间隔 |
| int | SymbNum | 1 | 符号数目 |

**输出参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| struct chestSL\_t \* | chestLS\_p | 1 | 信道估计参数 |

**返回值：**无

**函数：**ChannelEstimator\_LS

**功能：**采用基于LS的信道估计方法得到信道矩阵的估计值

**语法：**ChannelEstimator\_LS(lapack\_complex\_double \*SignalRec, lapack\_complex\_double \*PilotSymb, int \*PilotSymbIndex, int TxAntNum, int RxAntNum, int CarrierNum, int PilotSymbNum, int SymbNum, int inter\_freq, lapack\_complex\_double \*CFR\_est, struct chestLS\_t \*chestLS\_p);

**包含文件：**无

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| lapack\_complex\_double\* | SignalRec | (RxAntNum，CarrierNum, SymbNum) | 接收信号矩阵 |
| lapack\_complex\_double\* | PilotSymb | (TxAntNum, PilotSymbNum) | 导频符号 |
| int \* | PilotSymbIndex | (PilotSymbNum,1) | 导频的位置信息 |
| int | TxAntNum | 1 | 发射天线数目 |
| int | RxAntNum | 1 | 接收天线数目 |
| int | CarrierNum | 1 | 载波数目 |
| int | PilotSymbNum | 1 | 每个子帧中导频占有的资源粒子的总数目 |
| int | SymbNum | 1 | 每个子帧中的符号的数目 |
| int | freq | 1 | 频域插值间隔 |

**输出参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| lapack\_complex\_double\* | CFR\_est\_R | (RxAntNum,TxAntNum,CarrierNum,SymbNum) | 信道矩阵估计值 |

**返回值：**无

**函数：**chestLS\_ free

**功能：**LS信道估计参数释放

**语法：**void chestLS\_free(struct chestLS\_t \*chestLS\_p)

**包含文件：**无

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| struct chestLS\_t \* | chestLS\_p | 1 | 信道估计参数 |

**输出参数：**无

**返回值：**无

**函数：**chestDCT\_init

**功能：**DCT信道估计初始化

**语法：**void chestDCT\_init(struct chestDCT\_t \*chestDCT\_p, int CarrierNum, int inter\_freq, int SymbNum);

**包含文件：**无

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| int | CarrierNum | 1 | 子载波数 |
| int | inter\_freq | 1 | 频域插值间隔 |
| int | SymbNum | 1 | 符号数目 |

**输出参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| struct chestDCT\_t \* | chestDCT\_p | 1 | 信道估计参数 |

**返回值：**无

**函数：**ChannelEstimator\_DCT

**功能：**采用基于DCT的信道估计方法得到信道矩阵的估计值

**语法：**void ChannelEstimator\_DCT(lapack\_complex\_double \*SignalRec, lapack\_complex\_double \*PilotSymb, double \*R\_DCT, int \*PilotSymbIndex, int TxAntNum, int RxAntNum, int CarrierNum, int UserNum, int MaxBeam, int PilotSymbNum, int SymbNum, double sigma, lapack\_complex\_double \*CFR\_est);

**包含文件：**无

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| lapack\_complex\_double\* | SignalRec | (RxAntNum，CarrierNum, SymbNum) | 接收信号矩阵 |
| lapack\_complex\_double\* | PilotSymb | (TxAntNum, PilotSymbNum) | 导频符号 |
| double\* | R\_DCT | (RxAntNum\*UserNum,MaxBeam\*UserNum,CarrierNum) | 信道的相关阵 |
| int \* | PilotSymbIndex | (PilotSymbNum,1) | 导频的位置信息 |
| int | TxAntNum | 1 | 发射天线数目 |
| int | RxAntNum | 1 | 接收天线数目 |
| int | CarrierNum | 1 | 载波数目 |
| int | UserNum | 1 | 小区内用户数 |
| int | MaxBeam | 1 | 为每个用户发送数据的最大波束数目 |
| int | PilotSymbNum | 1 | 每个子帧中导频占有的资源粒子的总数目 |
| int | SymbNum | 1 | 每个子帧中的符号的数目 |
| int | inter\_freq | 1 | 频域插值间隔 |
| double | sigma | 1 | 噪声的标准差 |

**输出参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| lapack\_complex\_double\* | CFR\_est\_R | (RxAntNum,TxAntNum,CarrierNum,SymbNum) | 信道矩阵估计值 |

**返回值：**无

**函数：**chestDCT\_ free

**功能：**DCT信道估计参数释放

**语法：**void chestDCT\_free(struct chestDCT\_t \*chestDCT\_p)

**包含文件：**无

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| struct chestDCT\_t \* | chestDCT\_p | 1 | 信道估计参数 |

**输出参数：**无

**返回值：**无

## 2.11 信号检测

**函数：**CalSymb\_init

**功能：**信号检测初始化

**语法：**void CalSymb\_init(struct CalSymb\_t \*CalSymb\_p, int LayerNum, int RxAntNum)

**包含文件：**无

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| int | LayerNum | 1 | 发射信号的层数 |
| int | RxAntNum | 1 | 接收天线数目 |

**输出参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| struct CalSymb\_t \* | CalSymb\_p | 1 | 信号检测参数 |

**返回值：**无

**函数：**CalSymb\_mmse\_1

**功能：**单流信道估计

**语法：**void CalSymb\_mmse\_1(lapack\_complex\_double \*h\_tmp, lapack\_complex\_double \*Signal, int RxAntNum, int CarrierNum, int SymbNum, double sigma, int Step, int flg\_ave, lapack\_complex\_double \*SymbEst, double \*SymbVar, double \*SINR\_est)

**包含文件：**无

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| lapack\_complex\_double\* | h\_tmp | (RxAntNum, LayerNum, CarrierNum, SymbolNum) | 信道矩阵 |
| lapack\_complex\_double\* | Signal | (RxAntNum, CarrierNum, SymbolNum) | 接收信号矩阵 |
| int | RxAntNum | 1 | 接收机天线数目 |
| int | CarrierNum | 1 | 子载波数目 |
| int | SymbNum | 1 | 符号的数目 |
| double | sigma | 1 | 噪声标准差 |
| int | Step | 1 | 计算平均信道矩阵时所需的子载波数目 |
| int | flg\_ave | 1 | 表示是否对信道矩阵取平均，0表示不取平均，1表示取平均 |
| struct CalSymb\_t \* | CalSymb\_p | 1 | 信号检测参数 |

**输出参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| lapack\_complex\_double\* | SymbEst | (LayerNum, CarrierNum, SymbolNum) | 符号估计值 |
| double \* | SymbVar | (LayerNum, CarrierNum, SymbolNum) | 符号的估计方差矩阵 |
| double \* | SINR\_est | (LayerNum, CarrierNum, SymbolNum) | 信干噪比 |

**返回值：**无

**函数：** CalSymb\_mmse

**功能：**采用MMSE-QR分解的方法计算符号的估计值和方差

**语法：**void CalSymb\_mmse(lapack\_complex\_double \*h\_tmp, lapack\_complex\_double \*Signal, int RxAntNum, int CarrierNum, int LayerNum, int SymbNum, double sigma, int Step, int flg\_ave, lapack\_complex\_double \*SymbEst,double \*SymbVar);

**包含文件：**无

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| lapack\_complex\_double\* | h\_tmp | (RxAntNum, LayerNum, CarrierNum, SymbolNum) | 信道矩阵 |
| lapack\_complex\_double\* | Signal | (RxAntNum, CarrierNum, SymbolNum) | 接收信号矩阵 |
| int | RxAntNum | 1 | 接收机天线数目 |
| int | CarrierNum | 1 | 子载波数目 |
| int | LayerNum | 1 | 发射信号的层数 |
| int | SymbNum | 1 | 符号的数目 |
| double | sigma | 1 | 噪声标准差 |
| int | Step | 1 | 计算平均信道矩阵时所需的子载波数目 |
| int | flg\_ave | 1 | 表示是否对信道矩阵取平均，0表示不取平均，1表示取平均 |
| struct CalSymb\_t \* | CalSymb\_p | 1 | 信号检测参数 |

**输出参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| lapack\_complex\_double\* | SymbEst | (LayerNum, CarrierNum, SymbolNum) | 符号估计值 |
| double \* | SymbVar | (LayerNum, CarrierNum, SymbolNum) | 符号的估计方差矩阵 |
| double \* | SINR\_est | (LayerNum, CarrierNum, SymbolNum) | 信干噪比 |

**返回值：**无

**函数：**CalSymb\_free

**功能：**信号检测参数释放

**语法：**void CalSymb\_free(struct CalSymb\_t \*CalSymb\_p);

**包含文件：**无

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| struct CalSymb\_t \* | CalSymb\_p | 1 | 信号检测参数 |

**输出参数：**无

**返回值：**无

## 2.12 解符号映射

**函数：**QAM\_Demodulation

**功能：**软解调

**语法：**void QAM\_Demodulation(double\* LLRD, lapack\_complex\_double\* Recv, double\* Sigma2, double\* LLRA, int SymbNum, int SymbolBitN, int TxAntNum);

**包含文件：**无

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| lapack\_complex\_double\* | Recv | (TxAntNum\*CarrierNum, SymbNum) | 信号检测模块输出的信号估计值 |
| double\* | Sigma2 | (TxAntNum\*CarrierNum, SymbNum) | 信号检测模块输出的信号估计方差 |
| double\* | LLRA | (SymbolBitN,TxAntNum, SymbNum) | 编码比特序列的先验信息 |
| int | SymbNum | 1 | 符号数目 |
| int | SymbolBitN | 1 | 每个符号对应的二进制码字数目 |
| int | TxAntNum | 1 | 发送天线的数目 |

**输出参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| double\* | LLRD | (SymbolBitN,TxAntNum, SymbNum) | 编码比特序列的后验信息 |

**返回值：**无

## 2.13 解扰

暂未添加，保留。

## 2.14 解码块级联

**函数：**srslte\_cbsegm

**功能：**码块分割

**语法：**int srslte\_cbsegm(srslte\_cbsegm\_t \*s, uint32\_t tbs);

**包含文件：**cbsegm.h

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| uint32\_t | tbs | 1 | 传输块码长 |

**输出参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| srslte\_cbsegm\_t \* | s | 1 | 码块分割参数 |

**返回值：**码块分割状态

## 2.15 解速率匹配

**函数：**srslte\_rm\_turbo\_rx

**功能：**解速率匹配

**语法：**int srslte\_rm\_turbo\_rx(float \*w\_buff, uint32\_t w\_buff\_len, float \*input, uint32\_t in\_len, float \*output, uint32\_t out\_len, uint32\_t rv\_idx, uint32\_t nof\_filler\_bits);

**包含文件：**rm\_turbo.h

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| uint32\_t | w\_buff\_len | 1 | 循环缓冲器大小 |
| float \* | input | in\_len | 输入信息 |
| uint32\_t | in\_len | 1 | 输入码长 |
| uint32\_t | out\_len | 1 | 输出码长 |
| uint32\_t | rv\_idx | 1 | 循环缓冲器冗余版本 |
| uint32\_t | nof\_filler\_bits | 1 | 填充比特 |

**输出参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| float \* | w\_buff | w\_buff\_len(6176\*3) | 循环缓冲器 |
| float \* | output | 3\*6144+12 | 输出信息 |

**返回值：**速率匹配状态

**函数：**srslte\_rm\_turbo\_rx\_lut

**功能：**解速率匹配（查找表方式）

**语法：**int srslte\_rm\_turbo\_rx\_lut(int16\_t \*input, int16\_t \*output, uint32\_t in\_len, uint32\_t cb\_idx, uint32\_t rv\_idx);

**包含文件：**rm\_turbo.h

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| int16\_t \* | input | in\_len | 输入信息 |
| uint32\_t | in\_len | 1 | 输入码长 |
| uint32\_t | cb\_idx | 1 | 交织表索引值 |
| uint32\_t | rv\_idx | 1 | 循环缓冲器冗余版本 |

**输出参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| int16\_t \* | output | 3\*6144+12 | 输出信息 |

**返回值：**速率匹配状态

## 2.16 Turbo译码

**函数：**srslte\_tdec\_init

**功能：**初始化

**语法：**int srslte\_tdec\_init(srslte\_tdec\_gen\_t \* h, uint32\_t max\_long\_cb);

**包含文件：**turbodecoder.h

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| uint32\_t | max\_long\_cb | 1 | 最大码长 |

**输出参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| srslte\_tdec\_gen\_t \* | h | 1 | 解码参数 |

**返回值：**初始化状态

**函数：**srslte\_tdec\_run\_all

**功能：**调用函数进行解码

**语法：**int srslte\_tdec\_run\_all(srslte\_tdec\_gen\_t \* h, int16\_t \* input, uint8\_t \* output, uint32\_t nof\_iterations, uint32\_t long\_cb);

**包含文件：**turbodecoder.h

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| srslte\_tdec\_gen\_t \* | h | 1 | 解码参数 |
| int16\_t \* | input | SRSLTE\_TCOD\_MAX\_LEN\_CODED | 输入信息 |
| uint32\_t | nof\_iterations | 1 | 迭代次数 |
| uint32\_t | long\_cb | 1 | 码长 |

**输出参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| uint8\_t \* | output | SRSLTE\_TCOD\_MAX\_LEN\_CB / 8 | 输出信息比特 |

**返回值：**调用状态

**函数：**srslte\_tdec\_free

**功能：**释放内存

**语法：**void srslte\_tdec\_free(srslte\_tdec\_gen\_t \* h);

**包含文件：**turbodecoder.h

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| srslte\_tdec\_gen\_t \* | h | 1 | 解码参数 |

**返回值：**无

**函数：**srslte\_tdec\_reset

**功能：**重置解码器

**语法：**int srslte\_tdec\_reset(srslte\_tdec\_gen\_t \* h, uint32\_t long\_cb);

**包含文件：**vector.h turbodecoder.h turbodecoder\_gen.h turbodecoder\_sse.h

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| srslte\_tdec\_gen\_t \* | h | 1 | 解码参数 |
| uint32\_t | long\_cb | 1 | 码长 |

**返回值：**重置状态

**函数：**srslte\_tdec\_iteration

**功能：**迭代译码

**语法：**void srslte\_tdec\_iteration(srslte\_tdec\_gen\_t \* h, int16\_t \* input, uint32\_t long\_cb);

**包含文件：**vector.h turbodecoder.h turbodecoder\_gen.h turbodecoder\_sse.h

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| srslte\_tdec\_gen\_t \* | h | 1 | 解码参数 |
| int16\_t \* | input | SRSLTE\_TCOD\_MAX\_LEN\_CODED | 输入信息 |
| uint32\_t | long\_cb | 1 | 码长 |

**返回值：**无

**函数：**srslte\_tdec\_decision

**功能：**判决输出

**语法：**void srslte\_tdec\_decision(srslte\_tdec\_gen\_t \* h, uint8\_t \*output, uint32\_t long\_cb);

**包含文件：**vector.h turbodecoder.h turbodecoder\_gen.h turbodecoder\_sse.h

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| srslte\_tdec\_gen\_t \* | h | 1 | 解码参数 |
| uint32\_t | long\_cb | 1 | 码长 |

**输出参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| uint8\_t \* | output | SRSLTE\_TCOD\_MAX\_LEN\_CB | 输出信息 |

**返回值：**无

**函数：**srslte\_tdec\_decision\_byte

**功能：**判决输出

**语法：**void srslte\_tdec\_decision\_byte(srslte\_tdec\_gen\_t \* h, uint8\_t \*output, uint32\_t long\_cb);

**包含文件：**vector.h turbodecoder.h turbodecoder\_gen.h turbodecoder\_sse.h

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| srslte\_tdec\_gen\_t \* | h | 1 | 解码参数 |
| uint32\_t | long\_cb | 1 | 码长 |

**输出参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| uint8\_t \* | output | SRSLTE\_TCOD\_MAX\_LEN\_CB / 8 | 输出信息 |

**返回值：**无

## 2.17 CRC校验

**函数：** srslte\_crc\_checksum

**功能：**计算CRC

**语法：**uint32\_t srslte\_crc\_checksum(srslte\_crc\_t \*h, uint8\_t \*data, int len)

**包含文件：**crc.h

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| srslte\_crc\_t \* | h | 1 | CRC参数 |
| int | len | 1 | 输入码长 |

**输出参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| uint8\_t \* | data | len + crc\_order | 输出比特流 |

**返回值：**CRC比特流

## 2.18 解码块分割

无相关函数，直接在相关函数内完成。

# 3 多核并行计算思路

## 3.1 线程池

**函数：**pool\_init

**功能：**创建数量为threadNum的线程

**语法：**void pool\_init(int coreId\_start, int \_threadNum, int pool\_index);

**包含文件：**thread\_pool.h

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| int | coreId\_start | 1 | 分配线程初始序号 |
| int | threadNum | 1 | 创建线程的数量 |
| int | pool\_index | 1 | 线程池序号 |

**输出参数：**无

**返回值：**无

**函数：**pool\_add\_task

**功能：**将函数myfun添加到任务队列

**语法：**void pool\_add\_task(Fun myfun, void \*arg, int pool\_index);

**包含文件：**thread\_pool.h

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| Fun | myfun | 1 | 添加到任务队列的函数 |
| void \* | arg | 1 | 函数参数 |
| int | pool\_index | 1 | 线程池序号 |

**输出参数：**无

**返回值：**无

**函数：**pool\_destroy

**功能：**等待任务完成后销毁线程

**语法：**void pool\_destroy(int pool\_index);

**包含文件：**thread\_pool.h

**输入参数：**无

**输出参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| int | pool\_index | 1 | 线程池序号 |

**返回值：**无

## 3.2 线程池分配

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 线程池序号 | 0 | 1 | 2 | 3 |
| coreID | 0 | 1 | 2~7 | 8~63 |
| 功能 | 发送端任务调度 | 接收端任务调度 | 发送 | 接收 |

共开辟4个线程池，不同线程池绑定不同的核，互不干扰。线程池0用于发送端任务调度，线程池1用于接收端任务调度，各分配1个核，并采用轮询机制；线程池2用于发送端数据处理，分配6个核，线程池3用于接收端数据处理，分配54个核。

## 3.3 任务分割

### 3.2.1 传输块CRC添加

以传输块为基本单位。每个流1个任务。

### 3.2.2 码块分割

以传输块为基本单位。每个流1个任务。

### 3.2.3 码块CRC添加

以码块为基本单位。每个流若干个任务。

### 3.2.4 Turbo编码

以码块为基本单位。每个流若干个任务。

### 3.2.5 速率匹配

以码块为基本单位。每个流若干个任务。

### 3.2.6 码块级联

以码块为基本单位。每个流若干个任务。

### 3.2.7 加扰

以码块为基本单位。每个流若干个个任务。

### 3.2.8 符号映射

以码块为基本单位。每个流若干个个任务。

### 3.2.9 组包

以码块为基本单位。每个流若干个个任务。

### 3.2.10 解包

待定。

### 3.2.11 信道估计

以若干个SB为基本单位。每个流若干个任务。

### 3.2.12 信号检测

以若干个SB为基本单位。每个流若干个任务。

### 3.2.13 解符号映射

以若干个SB为基本单位。每个流若干个任务。

### 3.2.14 解扰

以码块为基本单位。每个流若干个任务。

### 3.2.15 解码块级联

以码块为基本单位。每个流若干个任务。

### 3.2.16 解速率匹配

以码块为基本单位。每个流若干个任务。

### 3.2.17 Turbo译码

以码块为基本单位。每个流若干个任务。

### 3.2.18 码块CRC校验

以码块为基本单位。每个流若干个任务。

### 3.2.19 解码块分割

以码块为基本单位。每个流1个任务。

### 3.2.20 传输块CRC校验

以传输块为基本单位。每个流1个任务。

# 4 多线程并行模块

注：paraNum\_tx 并行子帧数

taskNum\_tx 任务数

## 4.1 CRC添加-码块分割

**函数：** crc\_cbsegm

**功能：**CRC添加，码块分割

**语法：**void crc\_cbsegm(void \*arg);

**包含文件：**crc\_cbsegm.h

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| srslte\_crc\_t \* | crc\_p | 1 | CRC参数 |
| uint32\_t | crc\_poly | 1 | CRC版本 |
| int | crc\_length | 1 | CRC长度 |
| int | tbs | 1 | 传输块长度 |
| int | subframeIndex | 1 | 子帧序号 |
| int \* | ServiceEN\_tx | paraNum\_tx \* taskNum\_tx | 发送端使能信号 |
| int | taskNum\_tx | 1 | 发送端任务数 |

**输出参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| uint8\_t \* | tb | tbs + crc\_length | 输出比特流 |
| srslte\_cbsegm\_t \* | cb\_tx | 1 | 码块分割参数 |

**返回值：**无

## **4.2 CRC添加-Turbo编码-速率匹配-符号映射-组包**

**函数：** crc\_tcod\_rm

**功能：**CRC添加，Turbo编码，速率匹配，符号映射，组包

**语法：**void crc\_tcod\_rm(void \*arg);

**包含文件：**crc\_tcod\_rm.h

**输入参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| int | cbindex |  | 码块索引值 |
| srslte\_cbsegm\_t \* | cb\_tx | 1 | 码块分割参数 |
| uint8\_t \* | tbp | tbs + crc\_length | 添加CRC后的传输块 |
| srslte\_crc\_t | crc\_p |  | CRC参数 |
| uint32\_t | crc\_poly |  | CRC版本 |
| int | crc\_length |  | CRC长度 |
| uint8\_t \* | cb | 6144 | 码块 |
| srslte\_tcod\_t \* | tcod | 1 | 编码器参数 |
| int | cbs\_rm |  | 速率匹配后的码块长度 |
| uint8\_t \* | w\_buff | 6176 \* 3 | 循环缓冲器 |
| uint32\_t | rv\_idx |  | 循环缓冲器冗余版本 |
| int | conindex |  | 码块位置 |

**输出参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **变量名** | **大小** | **注释** |
| uint8\_t \* | cb\_tcod | 3 \* 6144+ 12 | 编码后的码块 |
| uint8\_t \* | cb\_rm | cbs\_rm | 速率匹配后的码块 |
| uint8\_t \* | tb | max\_rm\_data\_len\_tx | 码块级联后的传输块 |

**返回值：**无

## 4.3 解包

## **4.4 信道估计-信号检测- 解符号映射**

**函数：** chest\_calsym

**功能：**信道估计，信号检测-，解符号映射

**语法：**void chest\_calsym(void \*arg);

**包含文件：**chest\_calsym.h

## **4.5 解速率匹配-Turbo译码-CRC校验**

**函数：** derm\_crc

**功能：**解速率匹配，Turbo译码，CRC校验

**语法：**void derm\_crc(void \*arg);

**包含文件：**derm\_crc.h

## 4.6 **CRC校验**

**函数：** crc\_check

**功能：**CRC校验

**语法：**void crc\_check(void \*arg);

**包含文件：**crc\_check.h

# 5 并行计算架构

## 5.1 发送端



图5.3 发送端并行架构

## 5.2 接收端

图5.6 接收端并行架构

## 5.3 子帧间并行架构

图5.4 子帧间并行架构

## 5.4 轮询架构

图5.5 任务调度器轮询架构

# 6 性能测试

## 6.1 单流误块率

## 6.2 单线程运行时间

服务器性能：2.40GHz × 72核

子载波数：1200

流数：8

接收天线数：8

子帧数：1000

CQI：8流都设为15

信息量：1200子载波 \* 12OFDM符号 \* 8流 \* 6调制阶数 \* 1000子帧 = 660Mbit

### 6.2.1 发送端

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 模块 | CRC添加 | Turbo编码 | 速率匹配 | 符号映射 | 其他 | 总和 |
| 时间/s | 1.77 | 1.79 | 3.11 | 0.87 | 1.01 | 8.55 |
| 占比/% | 20.69 | 20.99 | 36.35 | 10.22 | 11.76 | 100.00 |

**吞吐量：660Mbit / 8.55s = 77.19Mbit/s**

### 6.2.2 接收端

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 模块 | 信道估计 | 信号检测 | 链路  自适应 | 解符号  映射 | 解速率  匹配 | Turbo  译码 | CRC  校验 | 其他 | 总和 |
| 时间/s | 5.26 | 19.50 | 7.26 | 5.61 | 1.39 | 31.00 | 1.78 | 4.85 | 76.72 |
| 占比/% | 6.85 | 25.41 | 9.46 | 7.42 | 1.81 | 40.41 | 2.32 | 6.32 | 100.00 |

**吞吐量：660Mbits / 76.72s = 8.60Mbit/s**

## 6.3 多线程并行度

子载波数：1200

流数：8

接收天线数：8

子帧数：1000

CQI：8流都设为15

发送端子帧并行数：2

接收端子帧并行数：5

### 6.3.1 发送端

### 6.3.2 接收端

表2 接收端并行度

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 线程数 | 1 | 2 | 4 | 8 | 16 | 24 | 32 | 40 | 48 | 56 |
| 时间/s | 73.665 | 37.541 | 18.913 | 9.721 | 5.355 | 3.584 | 2.741 | 2.220 | 1.915 | 1.622 |
| 并行度/% | 100 | 99.67 | 97.37 | 94.72 | 85.98 | 85.64 | 83.99 | 82.97 | 80.14 | 81.10 |