数据处理说明文档（服务器）

1 数据处理流程

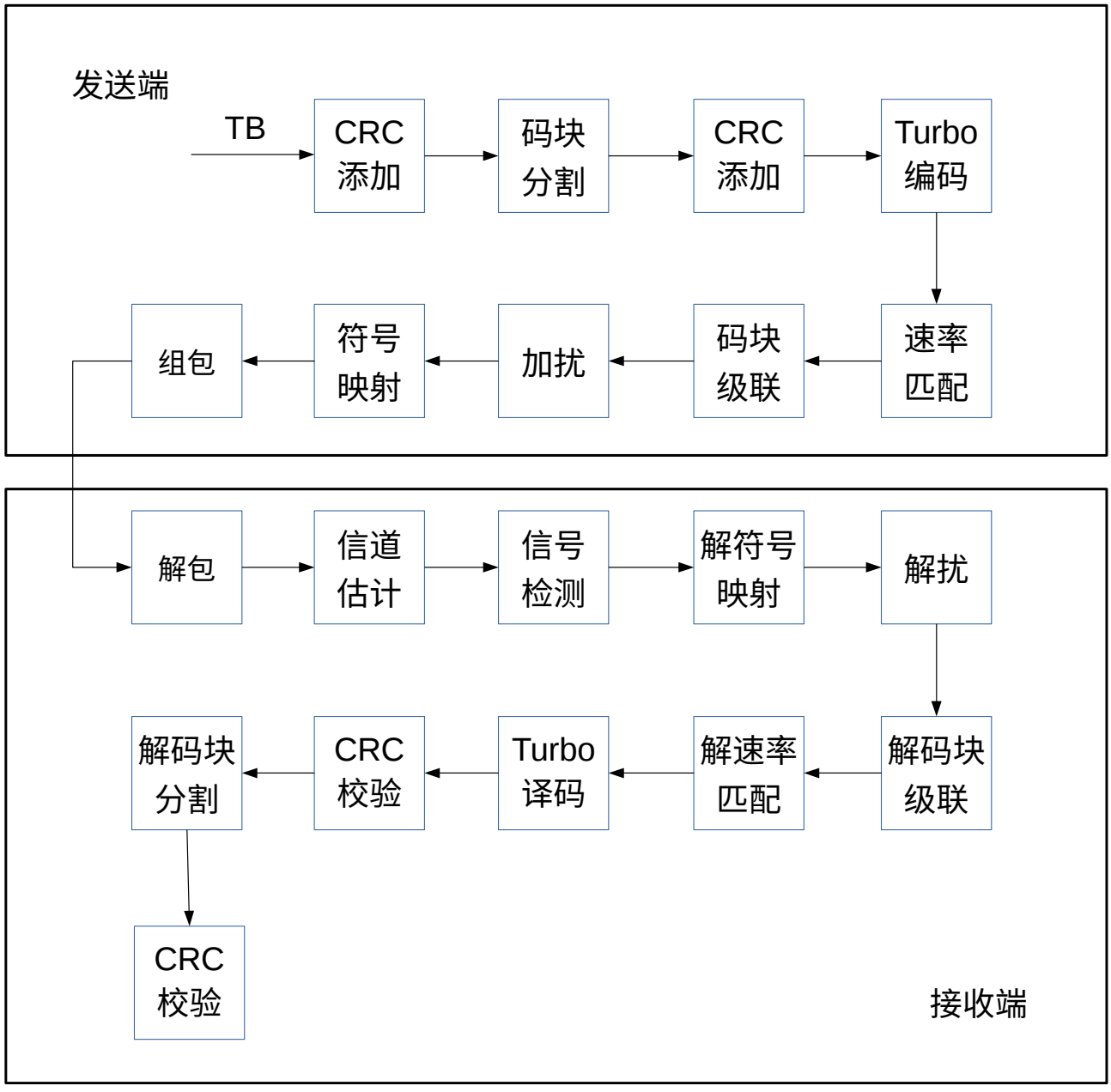

图 1.1 数据处理流程

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_	CFR_est_R	(RxAntNum,TxAntNum,Carri	信道矩阵估计值

double*		erNum,SymbNum)	

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_d	Signal	(RxAntNum, CarrierNum,	接收信号矩阵

ouble*		SymbolNum)	
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	输入信息

		_LEN_CODED	
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

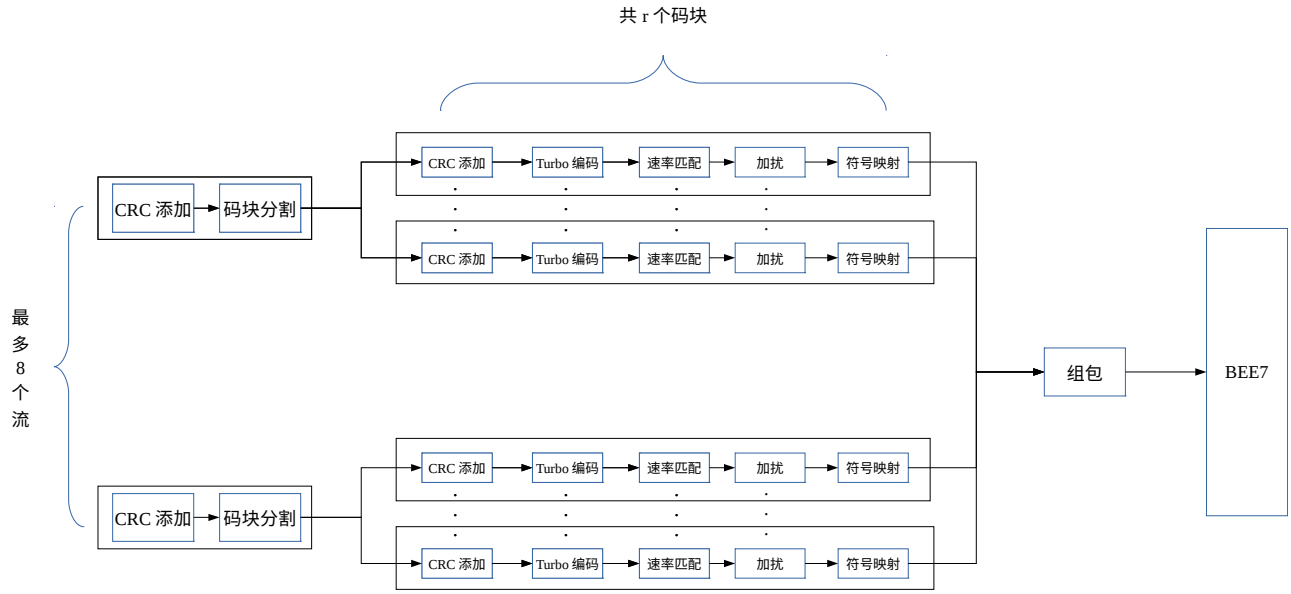

图 5.3 发送端并行架构

5.2 接收端

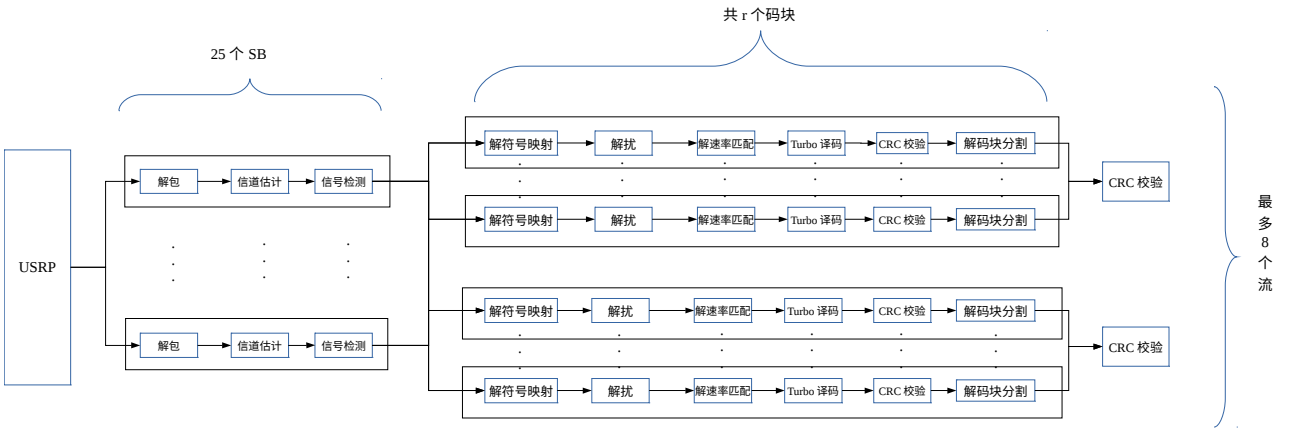

图 5.6 接收端并行架构

5.3 子帧间并行架构

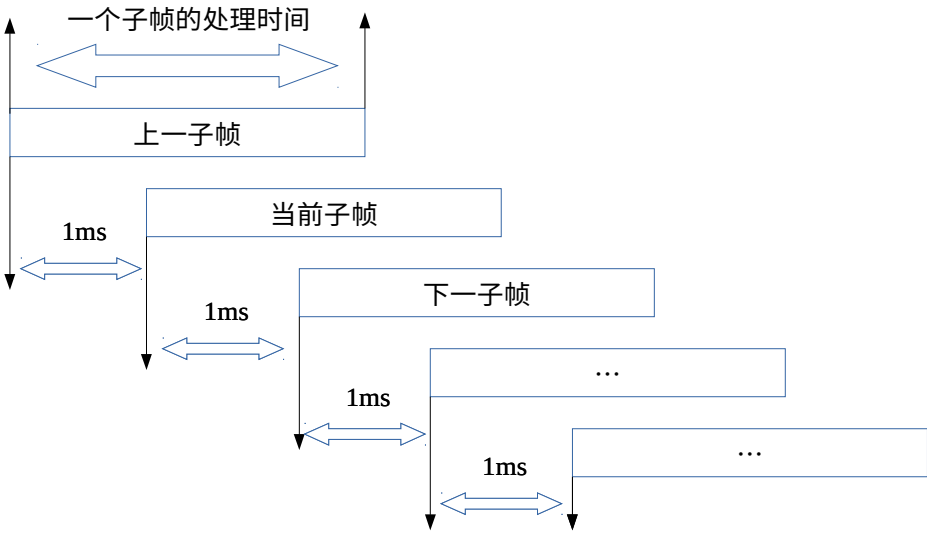

图 5.4 子帧间并行架构

5.4 轮询架构

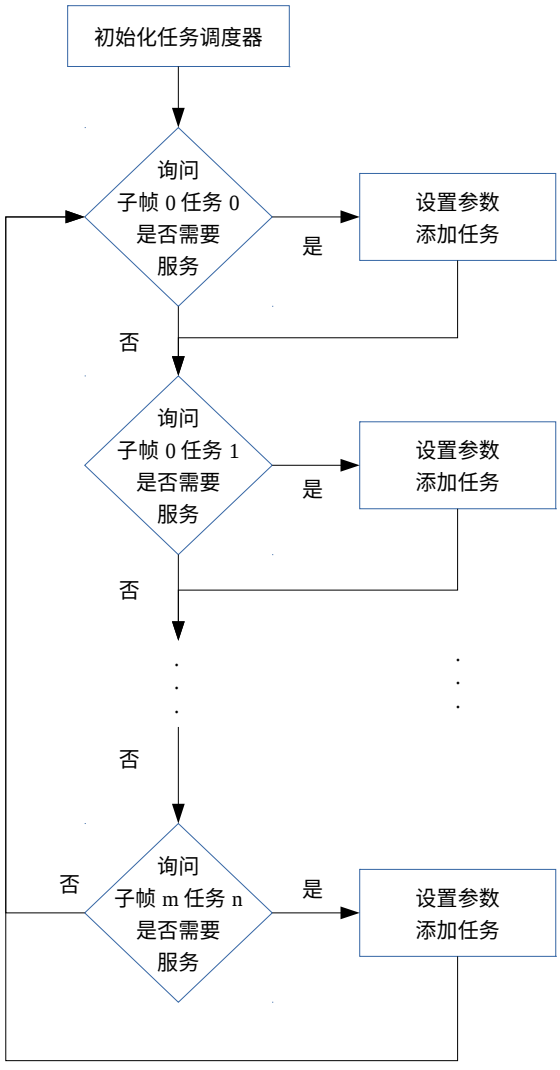

图 5.5 任务调度器轮询架构

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