# 软基带多线程具体实现：

## 启动进程进入main。

## 进行初始化操作：（初始化所得结果都定义为全局的）

运行一次得到preamble.

generatePreamble\_csd();

运行一次得到比特干扰码表。

Creatnewchart();

运行一次得到BCC编码表。

init\_BCCencode\_table();

运行一次得到生成导频的分流交织表。

initial\_streamwave\_table();

运行一次得到CSD表。

initcsdTableForHeLTF();

## 初始化内存：（注意此处的Ring和mempool都是全局的）

创建并初始化环形缓冲区Ring-Beforescramble用于将原始数据流传递到到scramble；

创建并初始化环形缓冲区Ring-scramble-2-BCC用于scramble到BCC之间传递数据；

创建并初始化环形缓冲区Ring-BCC-2-modulation用于BCC到modulation之间传递数据；

创建并初始化环形缓冲区Ring-modulation-2-CSD用于modulation到CSD之间传递数据；

创建并初始化环形缓冲区Ring-AfterCSD存储CSD以后的数据；

创建并初始化内存池mempool1与环形缓冲区Ring-Beforescramble相对应；

创建并初始化内存池mempool2与环形缓冲区Ring-scramble-2-BCC相对应；

创建并初始化内存池mempool3与环形缓冲区Ring-BCC-2-modulation相对应；

创建并初始化内存池mempool4与环形缓冲区Ring-modulation-2-CSD相对应；

创建并初始化内存池mempool5与环形缓冲区AfterCSD相对应；

## 启动线程（假设有10个核）

### 线程种类，功能和操作如下表;

|  |  |  |
| --- | --- | --- |
| 线程 | 功能 | 主要操作 |
| 线程ReadData\_Loop () | 进行线程跳转或执行读取源数据操作 | Ring-Beforescramble满则跳转到线程GenDataAndScramble\_Loop () |
| 否则执行ReadData () |
| 线程GenDataAndScramble\_Loop () | 进行线程跳转或执行扰码操作 | Ring-Beforescramble为空或Ring-BCC-2-modulation满则跳转到到线程ReadData\_Loop或BCC\_encoder\_Loop |
| 否则执行GenDataAndScramble() |
| 线程BCC\_encoder\_Loop () | 进行线程跳转或执行BCC编码操作 | Ring-scramble-2-BCC为空或Ring-scramble-2-BCC满则跳转到到线程GenDataAndScramble\_Loop ()或modulate\_Loop() |
| 否则执行BCC\_encoder() |
| 线程modulate\_Loop() | 进行线程跳转或执行调制映射操作（带插入导频与分流交织） | Ring-BCC-2-modulation为空或Ring-modulation-2-CSD满则跳转到到线程BCC\_encoder\_Loop ()或Data\_CSD\_Loop () |
| 否则执行modulate() |
| 线程Data\_CSD\_Loop () | 进行线程跳转或执行CSD操作 | Ring-modulation-2-CSD为空为空或Ring-AfterCSD满则跳转到到线程modulate\_Loop() |
| 否则执行Data\_CSD() |

## 五种线程详解

* 1. 一个核启动线程ReadData\_Loop ()，线程ReadData\_Loop ()中执行死循环，死循环中进行如下判断。如果Ring-Beforescramble满则跳转到GenDataAndScramble\_Loop ()，否则执行ReadData (),ReadData ()的输入参数为rte\_mempool\_get(message\_pool1, &Data)的数据块指针。
  2. 一个核启动线程GenDataAndScramble\_Loop ()，线程GenDataAndScramble\_Loop ()中执行死循环，死循环中进行如下判断。如果Ring-Beforescramble为空则跳转到ReadData\_Loop ()，否则进行下面判断：如果Ring-scramble-2-BCC满则跳转到BCC\_encoder\_Loop ()，否则执行GenDataAndScramble ()，GenDataAndScramble ()的输入参数为rte\_ring\_dequeue(Ring\_Beforescramble, &Data\_In)得到的数据块指针Data\_In和rte\_ring\_enqueue(Ring\_scramble\_2\_BCC, Data\_Out)得到的数据块指针Data\_Out。
  3. 三个核启动线程BCC\_encoder\_Loop ()，线程BCC\_encoder\_Loop ()中执行死循环，死循环中进行如下判断。如果Ring-scramble-2-BCC为空则跳转到GenDataAndScramble\_Loop ()，否则进行下面判断：如果Ring-BCC-2-modulation满则跳转到modulate ()\_Loop，否则执行BCC\_encoder ()，BCC\_encoder ()的输入参数为rte\_ring\_dequeue(Ring-scramble-2-BCC, &Data\_In)得到的数据块指针Data\_In和rte\_ring\_enqueue(Ring-BCC-2-modulation, Data\_Out)得到的数据块指针Data\_Out。
  4. 三个核启动线程modulate\_Loop()，线程modulate\_Loop()中执行死循环，死循环中进行如下判断。如果Ring-BCC-2-modulation为空则跳转到BCC\_encoder \_Loop ()，否则进行下面判断：如果Ring-modulation-2-CSD满则跳转到Data\_CSD\_Loop ()，否则执行modulation()，modulation()的输入参数为rte\_ring\_dequeue(Ring-BCC-2-modulation, &Data\_In)得到的数据块指针Data\_In和rte\_ring\_enqueue(Ring-modulation-2-CSD, Data\_Out)得到的数据块指针Data\_Out。。此处modulate ()过程中已经将插入导频部分完成。
  5. 两个核启动线程Data\_CSD\_Loop ()，线程Data\_CSD\_Loop ()中执行死循环，死循环中进行如下判断。如果Ring-modulation-2-CSD为空则跳转到modulate\_Loop ()，否则进行下面判断：如果Ring-AfterCSD满则执行跳转到modulate\_Loop ()，否则执行Data\_CSD ()，Data\_CSD ()的输入参数为rte\_ring\_dequeue(Ring-modulation-2-CSD, &Data\_In)得到的数据块指针Data\_In和rte\_ring\_enqueue(Ring-AfterCSD, Data\_Out)得到的数据块指针Data\_Out。。此处modulate ()过程中已经将插入导频部分完成。

## 线程操作函数种类及其操作

|  |  |  |
| --- | --- | --- |
| 线程操作函数 | 功能 | 主要操作 |
| ReadData () | 执行读取源数据操作  ReadData ()的输入参数为rte\_mempool\_get(message\_pool1, &Data)的数据块指针Data | 从将文件读取数据复制给Data即原始数据流 |
|
| 线程GenDataAndScramble\_Loop () | 进行线程跳转或执行扰码操作  GenDataAndScramble ()的输入参数为rte\_ring\_dequeue(Ring\_Beforescramble, &Data\_In)得到的数据块指针Data\_In和rte\_ring\_enqueue(Ring\_scramble\_2\_BCC, Data\_Out)得到的数据块指针Data\_Out | 从Data\_In获取原始输入数据通过查比特扰码表，对原始数据进行比特化并加扰  最终输出数据到Data\_Out |
|
| 线程BCC\_encoder\_Loop () | BCC\_encoder ()的输入参数为rte\_ring\_dequeue(Ring-scramble-2-BCC, &Data\_In)得到的数据块指针Data\_In和rte\_ring\_enqueue(Ring-BCC-2-modulation, Data\_Out)得到的数据块指针Data\_Out | 从Data\_In获取原始输入数据  （扰码后的数据流）输出BCC编码后的数据到Data\_Out。 |
|
| 线程modulate\_Loop() | modulation()的输入参数为rte\_ring\_dequeue(Ring-BCC-2-modulation, &Data\_In)得到的数据块指针Data\_In和rte\_ring\_enqueue(Ring-modulation-2-CSD, Data\_Out)得到的数据块指针Data\_Out。 | 从Data\_In获取原始输入数据（BCC编码）  对BCC编码后的数据查分流交织表进行分流交织，调制映射和插入导频。最终输出数据到Data\_Out |
|
| 线程Data\_CSD\_Loop () | Data\_CSD ()的输入参数为rte\_ring\_dequeue(Ring-modulation-2-CSD, &Data\_In)得到的数据块指针Data\_In和rte\_ring\_enqueue(Ring-AfterCSD, Data\_Out)得到的数据块指针Data\_Out。 | 输入数据为调制映射并插入导频后的数据Data\_I，通过初始化CSD表，通过查表利用  向量指令进行CSD计算。输出为CSD后的数据Data\_Out。 |
|
|  |  |  |

## 线程间主要数据结构：

|  |  |  |
| --- | --- | --- |
| 数据结构 | 功能 | 主要操作 |
| 环形缓冲区Ring-Beforescramble | 将原始数据流传递到到scramble | ReadData ()将原始数据流入队到Ring-Beforescramble |
| GenDataAndScramble ()将对Ring-Beforescramble原始数据流进行出队操作 |
| 环形缓冲区Ring-scramble-2-BCC | 用于scramble到BCC之间传递数据 | GenDataAndScramble ()将扰码后数据流入队到Ring-scramble-2-BCC |
| BCC\_encoder ()将对Ring-scramble-2-BCC的扰码后数据流进行出队操作 |
| 环形缓冲区Ring-BCC-2-modulation | 用于BCC到modulation之间传递数据 | BCC\_encoder ()将BCC编码后数据流入队到Ring-BCC-2-modulation |
| modulate ()将对Ring-BCC-2-modulation的BCC编码后数据流进行出队操作 |
| 环形缓冲区Ring-modulation-2-CSD | 用于modulation到CSD之间传递数据 | modulate ()将调制映射后数据流入队到Ring-modulation-2-CSD |
| Data\_CSD ()将对Ring-modulation-2-CSD的调制映射后数据流进行出队操作 |
| 环形缓冲区Ring-AfterCSD | 缓冲CSD以后的数据 | Data\_CSD ()将CSD后数据流入队到Ring-AfterCSD |
|  |
| 内存池mempool1 | 与环形缓冲区Ring-Beforescramble相对应 | ReadData ()为原始数据流从mempool1申请内存 |
| GenDataAndScramble()将原始数据流之前从mempool1申请内存中取出并将内存资源释放回mempool1 |
| 内存池mempool2 | 与环形缓冲区Ring-scramble-2-BCC相对应 | GenDataAndScramble ()为扰码后数据流从mempool2申请内存 |
| BCC\_encoder()将扰码后数据流从mempool2申请内存中取出并将内存资源释放回mempool2 |
| 内存池mempool3 | 与环形缓冲区Ring-BCC-2-modulation相对应 | BCC\_encoder()为BCC编码后数据流从mempool3申请内存 |
| modulate()将BCC编码后数据流从mempool3申请内存中取出并将内存资源释放回mempool3 |
| 内存池mempool4 | 与环形缓冲区Ring-modulation-2-CSD相对应 | modulate()为调制映射后数据流从mempool4申请内存 |
| Data\_CSD()将调制映射后数据流从mempool4申请内存中取出并将内存资源释放回mempool4 |
| 内存池mempool5 | 与环形缓冲区AfterCSD相对应 | Data\_CSD()为CSD后数据流从mempool5申请内存 |
|  |

# 软基站DATA部分优化后的函数接口和数据结构

## 流程框图

具体流程框图见PDF文件。

## 函数接口及作用

1. **Data部分数据产生主要函数，和辅助参数函数**
   1. **void mcs\_table\_for\_20M(unsigned char \*rate\_type, int \*N\_BPSCS, int \*N\_DBPS, int \*N\_CBPS, int \*N\_ES);**

**接口：**输入为宏定义的MCS等级，输出为一些辅助参数，若MCS等级不变，这些参数的值也不会变。

**操作：**MCS查表，根据MCS等级获得调制方法和编码速度等。

* 1. **void Creatnewchart();**

**操作：**生成比特扰码表：

Scramble\_t ChartTable[256][128];。

* 1. **void init\_BCCencode\_table();**

**操作：**初始化生成BCC编码表，BCCStruct\_t BCCTable[256][128];

* 1. **void initial\_streamwave\_table();**

**操作：**生成含有导频的分流交织表：

unsigned int \*streamweave\_table[N\_STS];

1. **生成比特并扰码**
   1. **void GenDataAndScramble(unsigned char \*data\_scramble, int ScrLength, unsigned char \*databits, int valid\_bits);**

**接口：**输入为原始数据流和有效数据位，输出为byte转换为bit存储并加扰码后的数据流。

**操作：**通过查比特扰码表，对原始数据进行比特化并加扰。

1. **BCC编码**
   1. **void BCC\_encoder\_OPT(unsigned char \*data\_scramble, int ScrLength, int N\_SYM, unsigned char \*\*code\_out, int CodeLength);**

**接口：**输入为扰码后的数据流及其有效数据长度和符号数，输出为BCC编码后的数据此时数据。

1. **调制映射**
   1. **void modulate\_mapping(unsigned char \*BCC\_output, complex32 \*\*subcar\_map\_data);**

**接口：**输入为BCC编码后的数据，输出为经过分流交织，调制映射和插入导频后的数据。

**操作：**对BCC编码后的数据查分流交织表进行分流交织，调制映射和插入导频。

* 1. **void parser\_stream\_interweave(unsigned char \*output,unsigned int \*\*stream\_interweave\_dataout,unsigned int \*\*stream\_parser\_weave\_table)**

**接口：**输入为BCC编码后的数据和分流交织表，输出为分流交织后的数据。

**操作：**按照当前数据流的符号数，通过查分流交织表对BCC编码之后的数据进行分流交织，分流交织后的数据相应位置中含有导频标志。

* 1. **void \_\_bi2de\_opt(unsigned int \*\*code\_out,int mode,int num,int Nov\_STS);**

**接口：**输入为经过分流交织表后的二进制数据，调制方式，调制后数据长度，和数据流数。

**操作：**对分流交织后的数据进行二进制数到十进制数的转换，转换后的数据仍保存在原数组中。只相当于对原数据占用的内存空间进行了压缩。此步骤数据对应位置仍包含有导频信息。

* 1. **void \_\_Modulation\_11ax\_opt(unsigned int \*\*code\_out, int mode,int num, complex32 \*\*sym\_mod,int Nov\_STS);**

**接口：**输入数据为进行调制后的数据，输出为调制映射后的最终数据。

**操作：**对二进制转换为十进制后的数据按照设定的调制方式进行星座映射，同时在相应的导频位置插入相应的导频成分。

1. **数据的CSD移位**
   1. **void \_\_Data\_CSD\_aux(complex32 \*\*subcar\_map\_data, int N\_SYM, complex32 \*\*csd\_data,int NTXindex);**

**接口：**输入数据为调制映射并插入导频后的数据，输出为CSD后的数据。

**操作：**初始化CSD表，通过查表利用向量指令进行CSD计算。

* 1. **void initcsdTableForHeLTF();**

**操作：**初始化CSD表：complex32 csdTableForHeLTF[N\_TX][256]。