接收机同步相关算法

密级**: 绝密**

**自组网通信组冯茂乔**



**版权声明 Copyright Notification**

本文档由成都奥特为通讯有限公司起草

中国 四川 成都

未经书面许可 禁止打印、复制及通过任何媒体传播

© 2017成都奥特为通讯有限公司 版权所有

# 定时同步

定时同步中分为帧同步和符号同步，AGC模块中包括AGC\_SP(signal process)和AGC两个模块，包含延迟相关算法和互相关算法，完成帧同步，而符号同步设计在接收机rx\_top的init\_sync中。

## 帧同步(分组检测)

分组检测是寻找数据分组起始地近似估算，是接收机工作的第一步。

常用分组检测方法：

1. 接收信号能量检测

通过滑动窗口求取共轭相乘再窗口内累加结果达到，监控一段时隙内的无线信号能量。

1. 双滑动窗口分组检测
2. 采用前导结构进行分组检测

采用延时相关算法进行相关检测简单有效，方法是通过利用短训练符号的16周期性。



图 1 基本延迟相关算法框图

利用延迟相关算法，可以有效区别噪声和OFDM有效信号，注意除法和绝对值的简化处理。

### 延迟自相关算法

延时相关算法基本公式：

(1-1-1)

(1-1-2)

(1-1-3)

其中，为接收信号样值，L为短训练符号长度，滑动窗口C求得互相关系数，D为接收信号延时采样点数(16)，滑动窗口P计算接收信号能量，为帧检测的判决函数。

为了避免除法运算，而且滑动窗口求和困难，因此将延迟相关值做如下推导：

(1-1-4)

将接收信号能量做如下推导：

(1-1-5)

得到移动递归公式，不仅简化了延迟相关自相关算法的乘加运算，而且更容易硬件实现。采用该算法的电路框图如下：



图 2 延迟相关值计算框图



图 3 接收信号能量计算框图

在延迟相关算法里，采用量化后的复数数据根据正负选择原数据的加减组合得到复数乘法结果，取反是取复数数据的共轭，利用共轭进行复数相关。关键代码如下：

--IQ加减组合

iq\_add **<=** signed**(**i\_inbd\_sextd**)** **+** signed**(**q\_inbd\_sextd**);**

iq\_add\_comp **<=** "000000000000" **-** iq\_add\_int**;**

iq\_sub **<=** signed**(**i\_inbd\_sextd**)** **-** signed**(**q\_inbd\_sextd**);**

iq\_sub\_comp **<=** "000000000000" **-** iq\_sub\_int**;**

--取符号位量化

iq\_ff\_mux\_sel **<=** i\_ff\_sel**(**i\_ff\_sel'**high)** **&** q\_ff\_sel**(**q\_ff\_sel'**high);**

**case** iq\_ff\_mux\_sel **is**

--根据符号位选择加减组合合成IQ

**when** "00" **=>**

i\_add\_in1\_int **<=** iq\_ff\_add**;** --new R = i+q

q\_add\_in1\_int **<=** iq\_ff\_sub\_comp**;** --new I = -i+q

**when** "01" **=>**

i\_add\_in1\_int **<=** iq\_ff\_sub**;** --new R = i-q

q\_add\_in1\_int **<=** iq\_ff\_add**;** --new I = i+q

**when** "10" **=>**

i\_add\_in1\_int **<=** iq\_ff\_sub\_comp**;** --new R = -i+q

q\_add\_in1\_int **<=** iq\_ff\_add\_comp**;** --new I = -i-q

**when** **others** **=>**

i\_add\_in1\_int **<=** iq\_ff\_add\_comp**;** --new R = -i-q

q\_add\_in1\_int **<=** iq\_ff\_sub**;** --new I = i-q

**end** **case;**

这样分别得到接收信号能量ca\_rl和延迟相关值ca\_ac。得到ca\_rl和ca\_ac之后还需要进行阈值判决，该处理过程在AGC模块中的ofdm\_det，该模块对输入的ca\_rl和ca\_ac进行阈值相乘计算，并进行比较大小，以此判决帧同步。为了避免判决过程中的除法运算，将公式做如下变换：

(1-1-6)

即当相关值大于能量值乘以阈值时，检测到OFDM帧，否则没有OFDM数据帧到达。逻辑实现关键代码如下：

--阈值相乘

ca\_rl\_ac\_mult\_i **<=** rnd\_unsigned\_slv**((**unsigned**(**ca\_rl**)** **\*** unsigned**(**thr\_ac\_i**)),** 5**);**

--比较判决

**if** **(**en\_20m **=** '1' and **(**unsigned**(**ext**(**ca\_ac**,** ca\_rl\_ac\_mult\_i'**length))** **>** unsigned**(**ca\_rl\_ac\_mult\_i**)))** **then**

ac\_det\_d **<=** '1'**;**

**else**

ac\_det\_d **<=** '0'**;**

**end** **if;**

### 互相关算法

当完成帧同步延迟相关算法后，采用互相关算法得到相关峰位置。实际做法是将接收信号存入移位寄存器中，由移位寄存器来实现接收信号的延时。然后分别与本地存储的16个短训练符号的共轭值一一相乘。电路实现框图是：



图 4 互相关算法框图

在ofdm\_cross\_corr模块中完成复数相乘累加，取模，按16时钟输出。关键代码如下：

mod\_approx\_1 **:** mod\_approx

**generic** **map** **(**

data\_size\_g **=>** 16**,**

num\_ct\_g **=>** 4

**)**

**port** **map** **(**

data\_in\_i **=>** add\_out\_i**,**

data\_in\_q **=>** add\_out\_q**,**

--

data\_out **=>** ca\_cc\_int

**);**

-- Mux output

ca\_cc\_mux **<=** **(others** **=>** '0'**)** **when** **(**first16\_sample **=** '1'**)** **else**

ca\_cc\_int**;**

同样在ofdm\_det模块中进行阈值判定，由于帧同步时ca\_rl为一台阶信号，因此用互相关值ca\_cc首先乘以阈值，再和ca\_rl比较，若大于ca\_rl则判定为互相关峰，得到互相关峰时输出到后级进行精帧同步。关键逻辑代码如下：

**if(**en\_20m**=**'1'and**(**unsigned**(**ext**(**ca\_cc**,**ca\_rl\_cc\_mult\_i'**length))>**unsigned**(**ca\_rl\_cc\_mult\_i**)))** **then**

cc\_det\_d **<=** '1'**;**

**else**

cc\_det\_d **<=** '0'**;**

**end** **if;**

### 帧同步输出

得到延迟自相关结果和互相关结果后，软件根据实际情况设置帧同步的结果，帧同步分为混合模式和单一模式，单一模式只根据延迟自相关或者互相关结果，混合模式综合考虑自相关结果和互相关结果。

-- OFDM Detection output mixing setting

-- OFDM 检测模式选择，根据寄存器选择只用延迟相关算法还是混合算法

**if** **(**det\_mix\_i **=** '0'**)** **then**

**if** **(**ac\_det\_q**=**'1' or cc\_peak\_i**=**'1'**)** **then** -- Detection

det\_i **<=** '1'**;**

**else**

det\_i **<=** '0'**;**

**end** **if;**

**else**

**if** **(**ac\_det\_q**=**'1' and cc\_peak\_i**=**'1'**)** **then** -- Detection

det\_i **<=** '1'**;**

**else**

det\_i **<=** '0'**;**

**end** **if;**

**end** **if;**

电路框图如下：

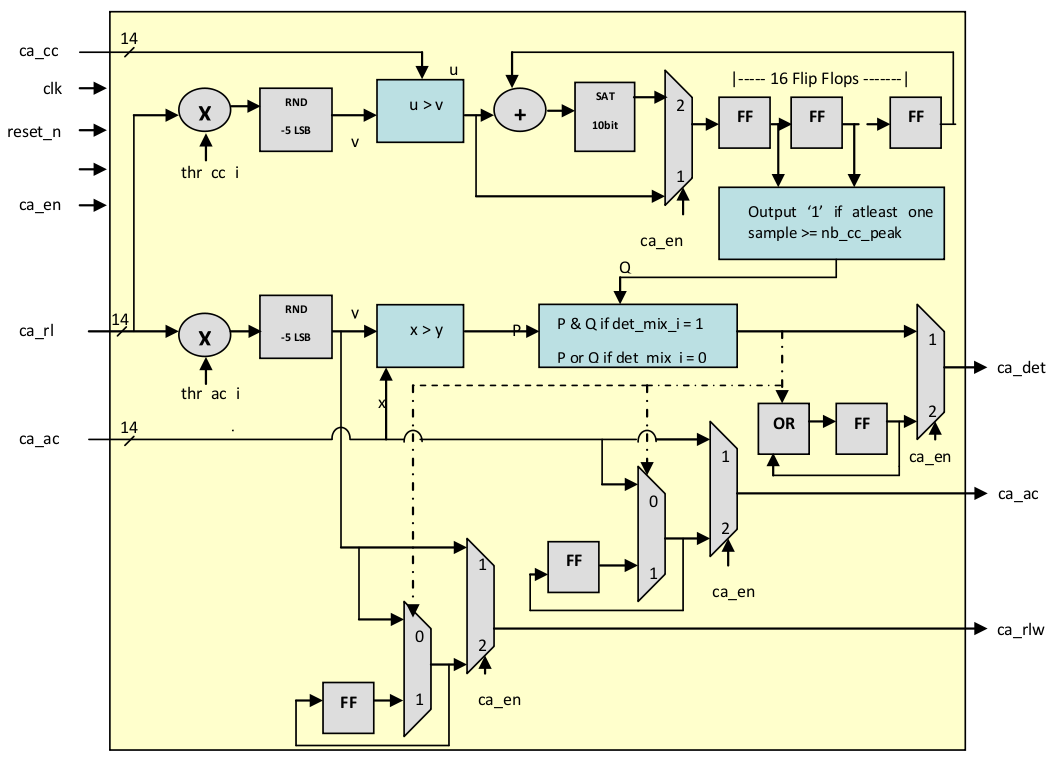


图 5 帧同步结果输出

当前模式不使用快速模式，ca\_en为1。

## 符号同步

符号同步的做法是将载波同步模块的输出与本地已知的短训练符号做互相关，从而将分组检测模块对数据分组的初略估算进行进一步的精确估算，确定短训练符号的结束点或者长训练符号的起始点。

在逻辑代码里，是确定长训练符号的起始点preamb\_detect，是一个脉冲信号，cp2\_detected与之同时出现，为一个电平信号，指示前端CCA可以停止帧搜索。

### 长训练符号preamb\_detect起始点的获取

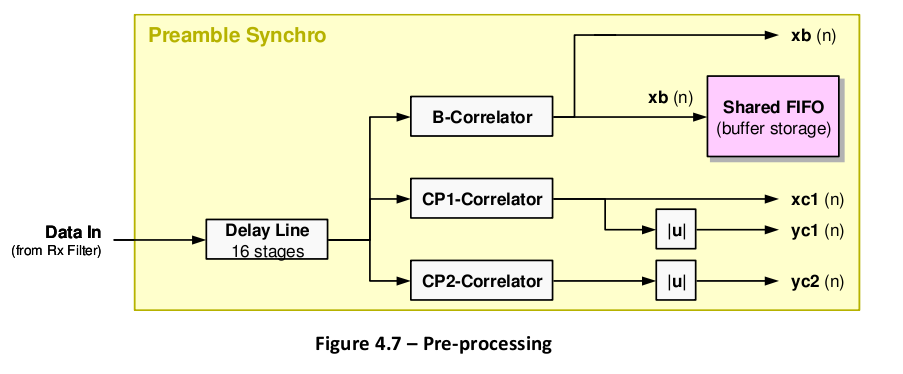
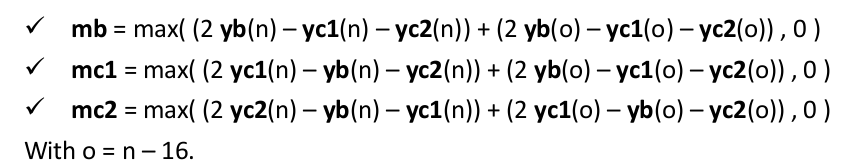


图 6 三次互相关

电路上的做法是在Pre-processing模块中，将基带数据按16进行滑动分别与短训练、循环前缀CP1，循环前缀CP2进行互相关，得到xb(n)，yc1(n)，yc2(n)，这样出现的结果是按时间进行下去依次出现9个短训练相关峰，1个CP1相关峰，1个CP2相关峰。

为了提高判决准确性稳定性，算法进行了相互的差值计算：



从判决相关值，转为判决相关值的差值最大值。

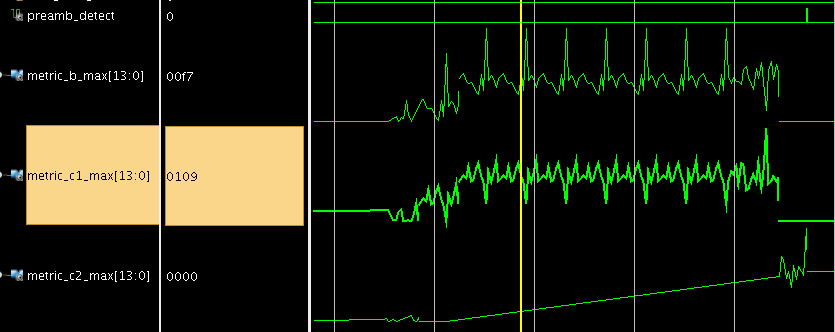
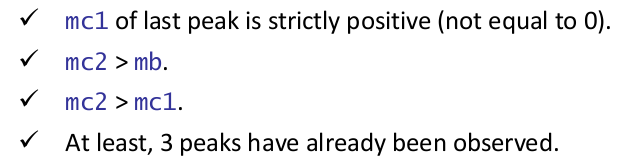


图 7 三次互相关结果

显然，当到达CP2末尾时，mb，mc1，mc2的大小关系应是mc2最大，此时判决preamb\_detect有效，preamb\_detect即是帧开始有效标志。但光有该条件可能存在误判，因此综合mc1和mb的结果，当cp2完结的时候，条件应为：1、mc2大于mc1和mb，且上一次的mc1(n-1)为正值。



### 传输开始和符号开始标志

Start\_of\_burst和start\_symbol分别标志标识传输开始和符号开始标识，由sample\_fifo模块产生，该模块主要根据preamb\_detect信号去移除保护间隔，产生传输开始和符号开始两个标志。

模块内含状态机，设计有为移除循环前缀的缓冲和输出模块。状态机内，根据计数器计数决定状态的转移。关键代码如下：

**when** short\_preamble **=>**

**if** frame\_start\_valid\_i **=** '1' **then**

sp\_fifo\_next\_state **<=** long\_preamble**;**

**end** **if;**

-- count 128 (T1-T2 are sent)

**when** long\_preamble **=>**

**if** data\_valid\_i **=** '1' and cnt\_rs **=** T1\_T2\_LENGTH\_CT **then**

sp\_fifo\_next\_state **<=** guard\_interval**;**

**end** **if;**

-- GUARD INTERVAL : ignore 16 data

**when** guard\_interval **=>**

**if** data\_valid\_i **=** '1' and cnt\_rs **=** GI\_LENGTH\_CT **then**

sp\_fifo\_next\_state **<=** symbol**;**

**end** **if;**

-- Symbol is sent : data are sent transparently

**when** symbol **=>**

**if** data\_valid\_i **=** '1' and cnt\_rs **=** SYMBOL\_LENGTH\_CT **then**

sp\_fifo\_next\_state **<=** guard\_interval**;**

**end** **if;**

其中frame\_start\_valid即是preamb\_detect，T1\_T2\_LENGTH\_CT，GI\_LENGTH\_CT，SYMBOL\_LENGTH\_CT分别为127，15，63，data\_valid\_i由IQ补偿模块产生。

当数据传输初始化后init\_i=1，一旦数据有效，下一状态立刻转移到symbol，若上一状态为空闲，则可判断进入burst，使start\_of\_burst信号有效，开始载波同步和解码。计数值cnt64每64个输出一次start\_of\_symbol信号提示OFDM符号有效。

**if** init\_i **=** '1' **then**

**elsif** out\_modes\_cur\_state **=** idle and out\_modes\_next\_state **=** symbol **then**

-- start\_of\_burst = '1' when leaving idle state

start\_of\_burst **<=** '1'**;**

**elsif** data\_ready\_i **=** '1' **then**

start\_of\_symbol **<=** '0'**;**

**if** data\_ready **=** '1' and data\_valid **=** '1' and cnt64 **=** CNT\_MAX63\_CT **then**

-- start\_of\_symbol = '1' when the 64 symbols have been sent

start\_of\_symbol **<=** '1'**;**

data\_valid **<=** '0'**;** -- new symbol is arriving

# 载波同步

载波同步采用粗细结合的方式。首先利用短训练符号进行粗略估计频率误差，然后采用该估算对长训练符号进行修正，修正后再采用长训练符号得到细频偏估计。

### 粗同步

在init\_sync中，已经做了基带数据与短训练序列B，循环前缀CP1，循环前缀CP2的相关计算，在载波同步的时候，根据短训练序列，CP1，CP2的符号间频偏进行载波粗同步。具体做法是：

1. 找到最后三次的短训练相关峰并使用cordic算法计算角度。
2. 找到CP1的相关峰位置并计算角度。
3. 将结果放在xp\_buffer中。

**type** GET\_CORDIC\_TYPE **is** **(**idle**,** -- wait for a f\_position

calc\_cp1\_cordic**,** -- calculate cordic for CP1

calc\_b\_cordic**,** -- calculate cordic for B

wait\_for\_init**);** -- wait for new reception

1. 得到4点相位值后，首先进行相位解缠恢复相位到正负无穷，然后进行根据4点相位拟合斜率估计出频率均值，最终由斜率换算出频率得到粗频偏估计。

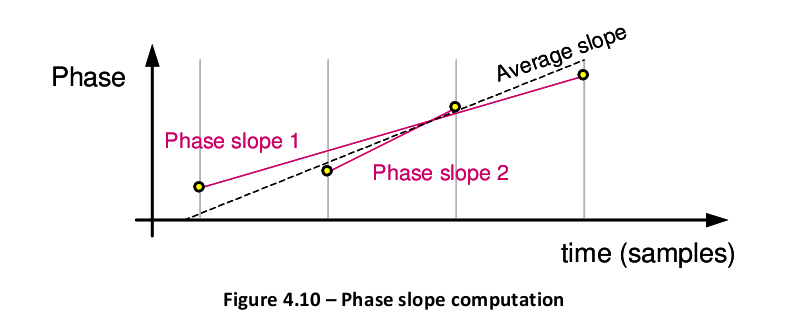


图 8 绝对相位拟合斜率估计频率

### 频偏校正

粗频偏校正值首先会进入频率校正模块，一旦有效后，模块就会对两个长序列进行校正，当start\_of\_burst信号有效后，64个采样值就会处理完，一次start\_of\_burst接收一次粗频偏校正值。

长序列在粗频偏校正完之后，会进入到细频偏估计模块。细频偏估计会计算出细频偏估计值并反馈到频率校正模块。该估计值是对粗频偏估计的一次再修正。修正的时候通过移位寄存器跳过保护间隔的16个采样点。

在细频偏校正后，长序列被送入信道估计模块，随后的符号都直接由细频偏校正模块校正。细频偏校正使用的是与粗频偏校正相同的算法。