From BigBang to 5G Part I

v.1.0 版

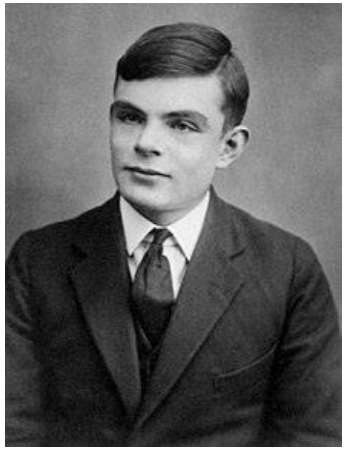
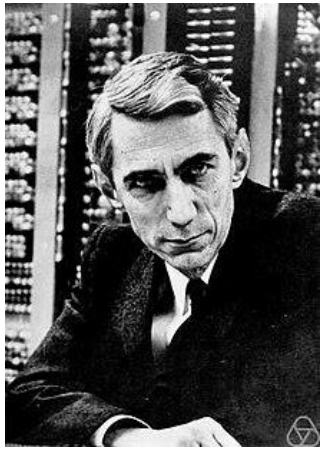
Gu Panzer
16.12.2017

NOKIA

向伟大的先贤们致敬

拜拜祖师爷，祥瑞御免

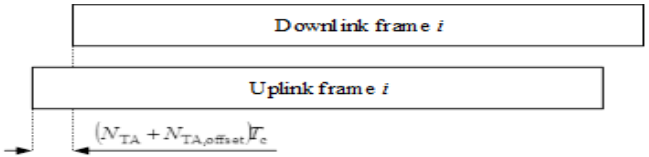
吴之雪风，祥瑞御免，我以外，全员大破…(咦，有乱入的)

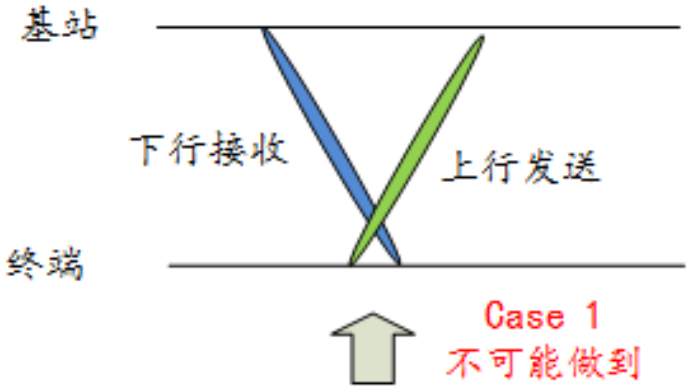

序章一：物理层先验概念

一些基本概念， 听上去像废话，但是要理解-1

- 电磁波的传送，也是需要时间的，距离越远，传输时间越长（废话吧）， 那么下面的图意味着什么，注意这个定理对Rach的处理，配置很大的影响，后面会讲（想想为什么？小区半径啊，亲）



- 如果还是不清楚，那么看下图，这种情况是不可能做到的，在一个频率上，发射接收机不可能同时工作在发射和接收两个频率上，。

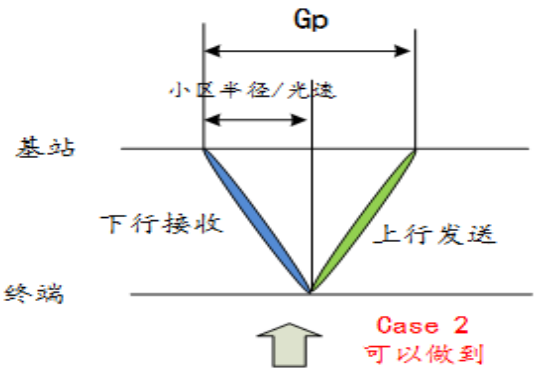


一些基本概念， 听上去像废话，但是要理解-2

- 而右面的情况，则是任何一个发射接收机是能够做到的，

  其中的Gp，就是(小区半径*2)/c （C为光速）

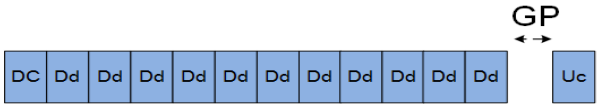
  另外任何的上行发射，都是提前至少(小区半径)/c， 才自



- 我们说至少是因为，除了电磁波在绝对距离上的消耗，

  还有多径（折射）引起的时间的消耗（高频段也许好一些）

- 在3G TDSCDMA，4G TDD和5G中Gp的位置（李世鹤老先生的专利），就是为了适应上行的传送要比下行接收提前，就是要空出来一定的时间，出现在DL symbol到UL symbol的switch位置，如下图：



- 电磁波（其实所有的波都是这样），频率越高，衰减越快，频率越高，绕射和衍射现象越弱→越高的频率，波动性越不明显，而粒子性越明显（咦，有一点点牛顿和波尔的感觉，但是我喜欢薛定谔）
- 电磁波频率越高，方向性越好（无绕射和衍射现象），打开了beam forming的天地，有阴影衰弱哦，亲

5G中3GPP的物理层帧格式定义（1）

- SCS(sub-carrier spacing): $\Delta f = 2^N .15kHz$ where N ranges from 0 to 5.*（想一下为什么？，频率越高，子载波间隔越大！！！）*

$$\Delta f = 15.2^{N_{band}}\,\text{kHz} \qquad \Delta f = 15.2^{N_{SS}}\,\text{kHz}$$

Note: the dedicated SCS for nominal and SS block(30kHz) are different

- Sample time units(码片长度): $T_s(N) = 1/(15000 \times 2^N \times 4096)$ seconds
- 无线帧(Radio frame) (10ms): $T_f = 614400\, 2^N T_s(N) = 10\,\text{ms}$
- Subframe and Slot: subframe=1ms, each subframe is made of 2^u slots
- Symbols: each slot has 14 OFDM symbols(normal CP), 12 OFDM symbols(extend CP), (想一下为什么)
- Slot types (38.211中定义了一张巨大的表)
  a. Slot including DL control channel and DL data channel,
  b. Slot including DL control channel, DL data channel and UL control channel,
  c. Slot including DL control channel and UL data channel,
  d. Slot including DL control channel, UL data channel and UL control channel.
  e. Slot including DL control channel, SS block and UL control channel
  f. Slot including UL RACH channel

关于物理层帧格式的一些思考(1)

- Verizon 的5G格式，是以子载波间隔定在75K，TTI的长度为200us，其目的是：
  - ✓ 用一种物理层帧格式去适配所有的应用场景。从理论上说200us的格式，的确是最接近所有应用场景的，而且一种只用物理层帧格式，的确是从2G->3G->4G的惯性思维。但是这个方案接近（讨好）了所有的场景，却不能彻底满足任何一个场景，也使得这种格式最后没有被3GPP采用的原因。
- 3GPP的5G格式，是以灵活配置子载波间隔，TTI的长度为基础的，其目的是：
  - ✓ 没有一种物理层帧格式能所有的应用场景。所以只能定义多种不同的格式
  - ✓ 同一技术多种不同的物理层帧格式，这在3GPP历史上，或者无线通讯历史上，似乎没有出现过，对手机芯片厂商是一个的挑战，有一定的实现难度（也许吧）。

关于物理层帧格式的一些思考（2）

- Slot在5G概念上的强化
  - ✓ 在4G LTE中也有slot的概念，一个subframe（1ms）中 2个slot，每个slot 0.5ms，但是slot这个概念在LTE中体现的不强。这个slot的除非支持Slot base的跳频，基本上没啥用，整个LTE系统是工作在sub frame的时序上的。调度器和L1都是以sub frame位周期运行。
  - ✓ 4G LTE对于slot概念上的弱化，原因在于4G依然是一个高速宽带无线接入网的设计初衷。Slot这个概念没啥用
  - ✓ 5G则大大不同，5G要面对的不是简单的一个高速接入网，而是不同的应用场景。那么体现不同应用场景具体就落实在slot这个概念上了。不同的slot格式带来不同使用场景的变化。调度器和L1都是工作在以slot为周期的节奏上了

物理层其他一些基本概念的定义-1(p，k,l……)

天线端口(antenna port)

- ✓ 这是一个很容易混淆的概念，很容易和天线发射单元混淆
- ✓ 实际上在物理层规范中，antenna port一般我称作逻辑天线端口(不知道是不是准确)，实际上是指物理层处理单元对要发送或者接受的数据/符号的处理方式，不同的逻辑天线端口用来区分不同的处理方式

要发送的符号为：$b_0 b_1 b_2$

我们定义两个port：P0 和 P1，这两个端口通过矩阵$\begin{pmatrix}2\\j\end{pmatrix}$来定义，那就分成两个逻辑端口信号

在P0上发送的是每个符号乘以2， 在P1上发送的是每个符号乘以j

RE(Resource element)

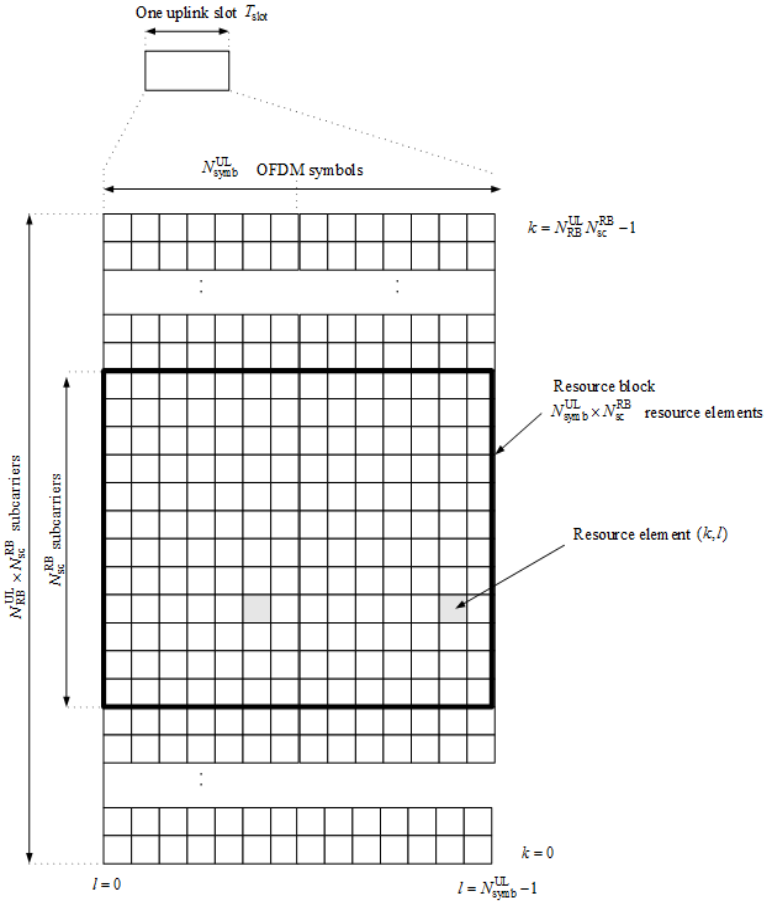
- ✓ 这是物理层定义的最小时间-频率资源单元，其定义为在时间上一个symbol，频率上一个子载波这样一个单元格
- ✓ 一个RE承载一个调制后的符号，具体几个bit呢，取决于调制方式 ->顺便提一下速率匹配，后面会详细讲

RB或者说PRB(Resource element)

- ✓ 这是L2（一般是指PS）所能看到的时频资源块，是一个Slot内的，12个子载波构成一个PRB..

PRB和RE关系看后一张图，为什么标题要写P，K，L???

物理层其他一些基本概念的定义-2（p，k, l……）



- PRB和RE的图例， 如左图
- 两个奇怪的定义（等待3GPP更新）
  - ✓ Reference resource blocks
  - ✓ Reference resource blocks
- Virtual resource block
  - ✓ LTE时代就有，我不喜欢，有点吃饱了撑着了

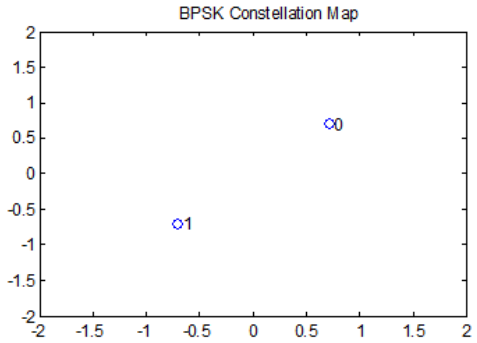
NOKIA

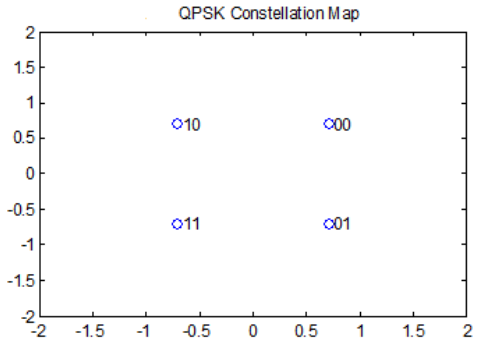
物理层其他一些基本概念的定义-3（调制方式）

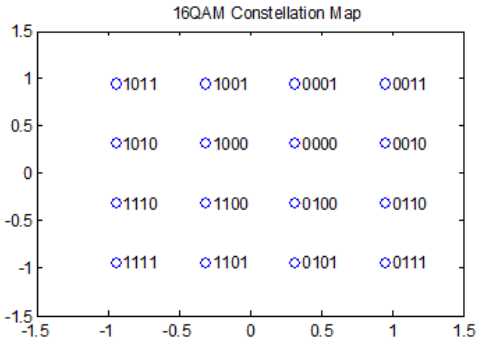
5G中引入了256QAM，

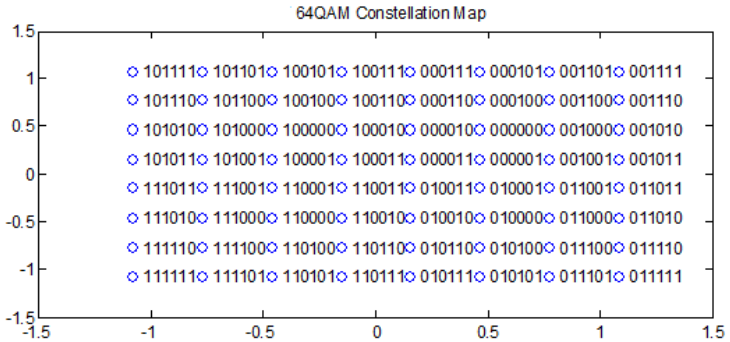
天啊，天雷滚滚

在这里我们可以多说几句，关于CQI，频谱效率，调制方式方面的取舍，是很有意思的。









序章二：两个奇怪的伪随机序列

！！！！极重要！！！！

这两个序列，特别是后一个（Z-C序列）贯穿整个物理层技术

NOKIA

Golden随机序列（Pseudo-random sequence）

◻ 序列的产生(from 3GPP)，Nc:1600

$$c(n) = (x_1(n+N_c) + x_2(n+N_c)) \bmod 2$$
$$x_1(n+31) = (x_1(n+3) + x_1(n)) \bmod 2$$
$$x_2(n+31) = (x_2(n+3) + x_2(n+2) + x_2(n+1) + x_2(n)) \bmod 2$$

其生成的过程解释如下：

- ✓ 我们可以看到序列有$x_{1\,(n)}$ 和$x_{2(n)}$序列比特加取模而成
- ✓ $x_{1(n)}$31位序列的初始化为 $x_1(0) = 1, x_1(n) = 0, n = 1, 2, \ldots, 30$
- ✓ $x_{2(n)}$31位序列的初始是应用指定的，可以是物理小区ID，身份证号码，林志玲的电话号码……(这就是序列的根)，而且这是Golden序列唯一的变量
- ✓ $x_{1(0)}$到$x_{1(31)}$确定了以后，那么第32位，33……位都可以算出来了，$x_{2(n)}$序列也是这样
- ✓ 然后一直算到1600位，其实是1600次迭代，伪随机序列的第一个数字，是决定于$x_{1(1600)}$和$x_{2(1600)}$， 伪随机序列的第二个数字取决为1601位的…以此类推

对于Golden随机序列的分析

- Golden序列从CDMA/GSM时代就开始应用了，从CDMA到WCDMA.LTE，5G，Golden序列一直有应用
- 在无线通讯中一般用于加扰
- Golden序列的优点：
  - ✓ Golden序列0和1的比例非常平衡，不会有多个连续0和连续1出现
  - ✓ Golden序列接近高斯白噪声
  - ✓ 有良好的相关特性，族序列数相对较大（2^{31}个序列）
- Golden序列的缺点：
  - ✓ 线性复杂度很低，扛干扰性能差
  - ✓ 有PAPR(功率的峰均比)问题，会导致线性功放的失真，特别是手机的功放……（在CDMA或者WCDMA中问题不严重，但是LTE这个宽带系统比较严重，到5G带宽更大导致PAPR问题非常严重）

Low-PAPR sequence（Zadoff－Chu 序列）

- 在3GPP中序列 $r_{u,v}^{(\alpha,\delta)}(n)$ 定义为 $r_{u,v}^{(\alpha,\delta)}(n) = e^{j\alpha\left(n+\delta\frac{\omega \bmod 2}{2}\right)}\bar{r}_{u,v}(n), \quad 0 \le n < M_{\mathrm{ZC}} - 1$
- 这个序列由两部分组成
  - ✓ 前半部分 $e^{j\alpha\left(n+\delta\frac{\omega \bmod 2}{2}\right)}$ 是一个相位旋转
  - ✓ 后面一部分 $\bar{r}_{u,v}(n)$ 定义为 $\bar{r}_{u,v}(n) = x_q\left(n \bmod N_{\mathrm{ZC}}\right)$ 然后 $q = \lfloor \bar{q} + 1/2 \rfloor + v \cdot (-1)^{\lfloor 2\bar{q} \rfloor}$
    $x_q(m) = e^{-j\frac{\pi q m(m+1)}{N_{\mathrm{ZC}}}}$ $\bar{q} = N_{\mathrm{ZC}} \cdot (u+1)/31$
  - ✓ 其中 N_{ZC} 是序列的长度
- 如果被公式吓到或者想快速入门，那么只要知道u，v，**α**和**δ**，唯一决定了一个序列，u和v一般用作序列的根（小区级别的），不同的用户可以有不同的**α**和**δ**值，实际就是各个信道的资源（听上去很重要吧）。然后跳过下一页
- 如果想研究，这个序列的生成的解释和具体分析在下一页

Low-PAPR sequence（Zadoff－Chu 序列)的生成

- 从 $q=\lfloor \bar{q}+1/2 \rfloor+v\cdot(-1)^{\lfloor 2\bar{q} \rfloor}$ 定义可以看出 如果u和v 定了那么这个q就定了

  $\bar{q}=N_{ZC}\cdot(u+1)/31$

- 所以 u 和 v 唯一的决定了 q ，也就决定了 这个根序列 $\bar{r}_{u,v}(n)=x_q(n \bmod N_{ZC})$

  $x_q(m)=e^{-j\frac{\pi qm(m+1)}{N_{ZC}}}$

- 从这个根序列看 $\bar{r}_{u,v}(m+1)/\bar{r}_{u,v}(m)$ 是

$x_q(m)=e^{-j\frac{\pi q(m+1)(m+2)-\pi q(m)(m+1)}{N_{ZC}}}$ 得到结果 $e^{-j\frac{\pi q2(m+1)}{N_{ZC}}}$

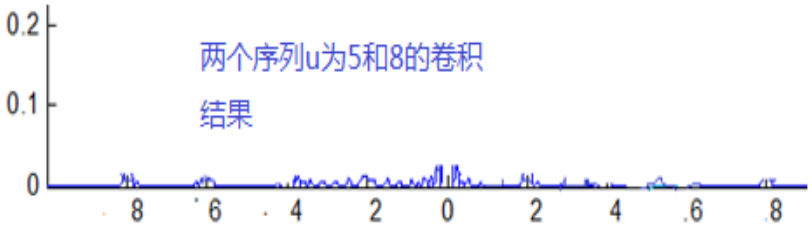
所以我们有结论：

$$\bar{r}_{u,v}(m+1)=\bar{r}_{u,v}(m)*e^{-j\frac{\pi q2(m+1)}{N_{ZC}}}$$

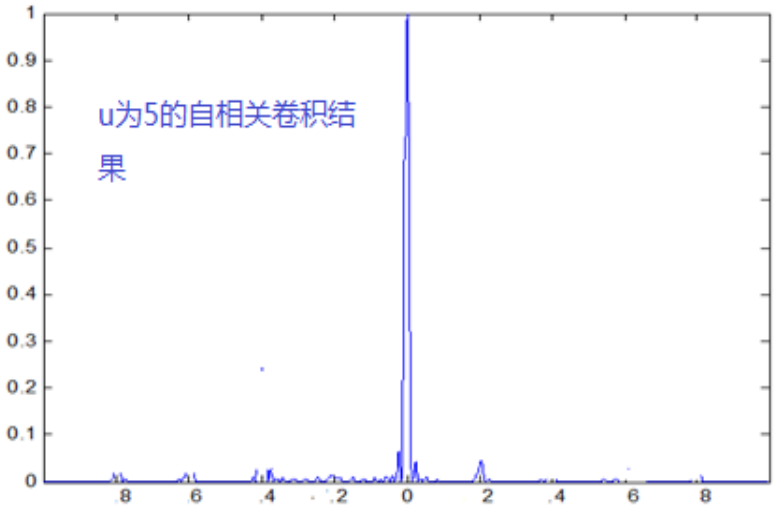
可以看出跟序列每一个符号，都是前一个符号相位旋转而来，不同的m个符号，相位旋转是线性的， 而Low-PAPR sequence 前一部分 $e^{j\alpha\left(n+\delta\frac{\omega \bmod 2}{2}\right)}$ 则是一个固定的相位旋转

Low-PAPR sequence （Zadoff－Chu 序列)的特性-1

- 如果两个根不一样的序列做相关卷积，我取了u = 5 和u = 8做下面的相关，如右图，我们可以看到什么？基本检测不到什么能量，有一些小起伏，因为我用N为16，还不够长，码越长则随机性更强



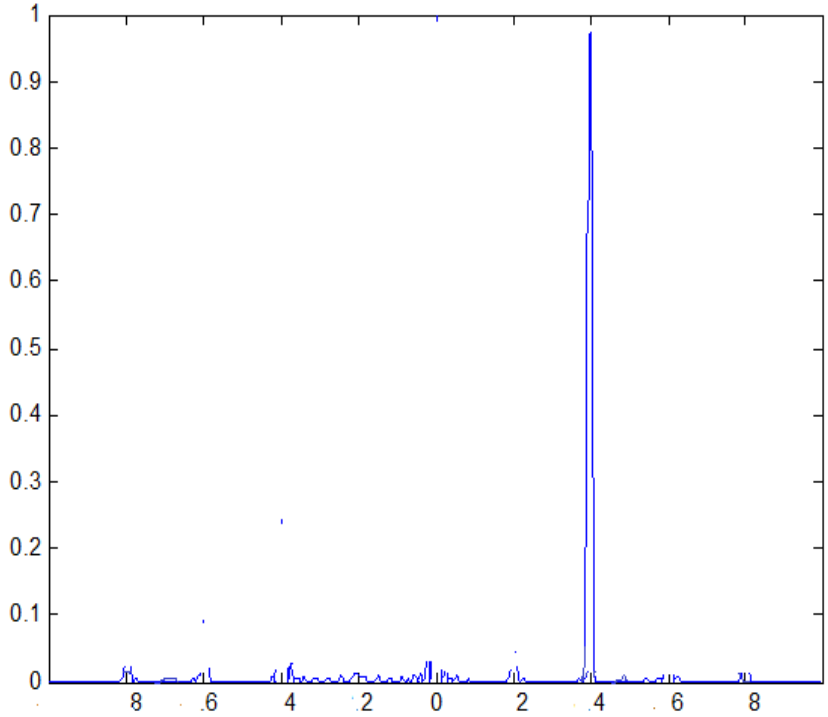
- 如果用一样的u=5，同样的序列，做一下相关，产生结果如左下图，我们可以看出这个序列有比较好的子相关性



Low-PAPR sequence（Zadoff－Chu 序列)的特性-2

- 如果两个根一样的序列做相关卷积，但是我们取不同的α和δ，那会是什么结果呢？为了简单α的取值不同，而δ都为0.
  - ✓ 序列1：u为5，v设为0，α设为0，δ为0
  - ✓ 序列1：u为5，v设为0，α设为4，δ为0

  这两个序列的卷积相关结果如下：



- 我们看到了什么？能量集中的位置偏移就是两个α值的差，即假如让两个根一致的 ZC 序列做卷积运算，则会在循环移位差值处出

那么这个特点可以用来干什么呢？

哈哈！！太多了

Low-PAPR sequence (Zadoff-Chu 序列)的特点分析

- 从Zadoff chu序列定义可以看出他是它是 CAZAC的一种，能量归一化的一个序列，也就是说它是恒幅值，所以绝对没有峰均比过高的问题。
- 序列具有非常好的自相关性和很低的互相关性，这种性能可以被用来产生同步信号，作为对时间和频率的相关运送
- 它的傅立叶变换仍然是 CAZAC 序列（OFDM 系统开心死了）

Low-PAPR sequence（Zadoff－Chu 序列)的性能分析-1

- 从Zadoff chu序列的根序列本质上是一个在相位上的进行随机旋转序列，然后再加上 $e^{j\alpha\left(n+\delta\frac{\omega \bmod 2}{2}\right)}$

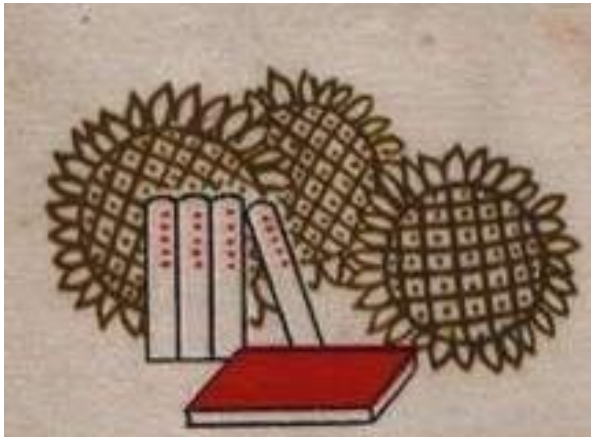
这么一个固定的相位旋转/偏差，有时候也叫Z-C序列的Cycle shift.

- ✓ 但是这个序列的优点和缺点都来自于相位旋转，优点是因为，他在能量在一个圆上，是归一化的，但是缺点是他本身对在有时偏和频偏的情况下，性能开始恶化
- ✓ 例如：对一个根产生的根序列，有839位，如果这是根是小区ID产生(一般都这么用)，那么一共理论上可以有839个Cycle shift,可以分给至少800多个UE使用，比如，一个UE1的cycle shift取3，另一个UE2取4，按照前面的分析，在用根序列做相关的时候，在3这个点，UE1会检测出峰值，UE2会在4这个点检测出峰值，理论上是，但是……

Low-PAPR sequence（Zadoff－Chu 序列)的性能分析-2

- 我我们要发送的信号总是$A*e^{jwt}$，如果频率上发生了偏移，比如由用户高速移动多普勒，则发出来的信号就 $A*e^{j(w+w0)t}$ 是,但是对方不知道，还是按照w这个频率去分析，这样就会产生jw_0*t 这样一个多出来的相位分量，使得相位发生了偏转（想想Z-C就是靠相位旋转来区分的）
- 如果UE因为上行同步不准确，在时间上发生了偏移，比如UE使用了乾坤大挪移，移动到了新位子上，是的基站原来收到的信号应该是$A*e^{jwt}$，而现在收到的是$A*e^{jw(t+t0)}$站,那么我们同样会多出来这样一个相位分量jwt_0
- 如果我们多出来的相位偏差，在前面一页的例子中，超过了1，那么UE1的信号，不仅检测不到，还会污染UE2的信号
- 所以实际的使用中，Cycle shift间隔都比较大，具体要看抗频率和相位抖动的要求了

序章三：两个凡是



凡是物理层数据信道，都坚决维持用LDPC码

凡是物理层控制信道，都始终不渝地遵循用极化码

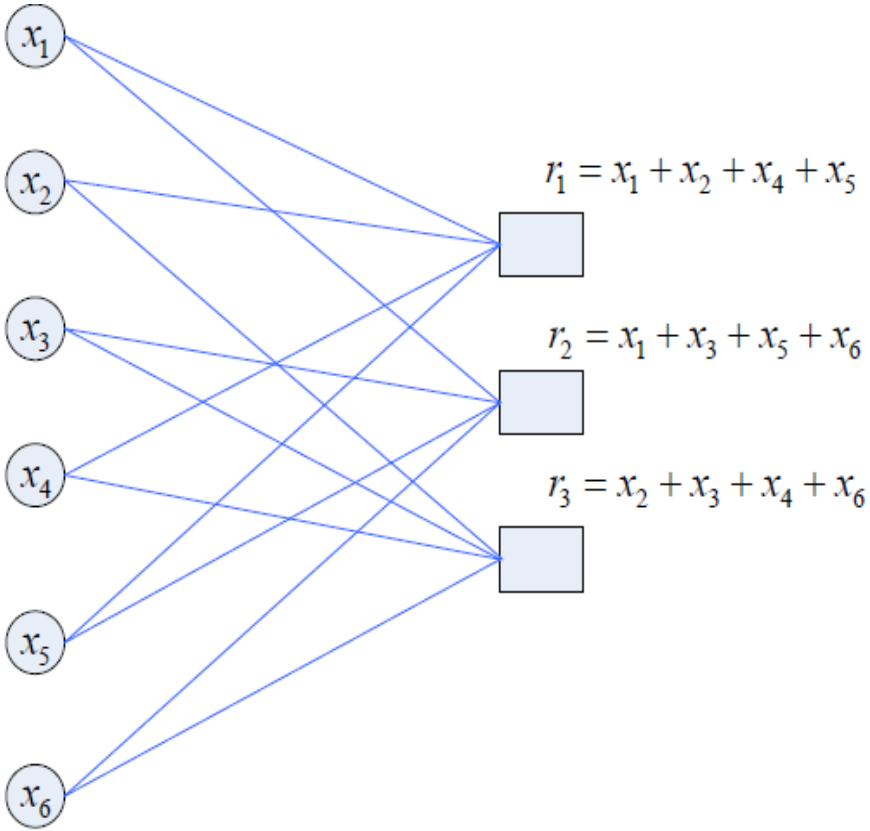
3GPP！！！！！你办事，我放心

LDPC码介绍--1

- LDPC码是基于校验比特的设置，其校验比特由稀疏矩阵来表述，如下图

$$H=\begin{bmatrix}1&1&0&1&1&0\\1&0&1&0&1&1\\0&1&1&1&0&1\end{bmatrix}$$

- 矩阵的行对应着校验方程，列对应着传输的校验比特上面的矩阵所对应的校验比特如右面的Tanner图
- 在5G中，校验比特的长度为信息比特发的两倍，所以和Turbo码一样，码率是1/3
- 译码的时候，原理上是一种投票的过程



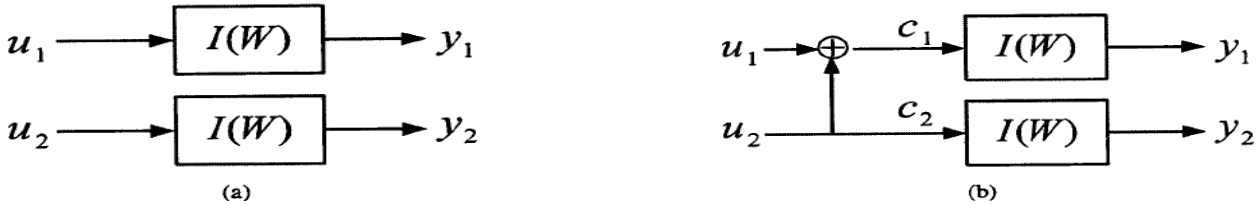
LDPC码介绍--2

- LDPC码是一种线性分组码，它于1962年由Gallager提出，之后很长一段时间没有被重视。直到1993年Berrou等提出了turbo码，人们发现turbo码从某种角度上说也是一种LDPC码，近几年的研究重新认识到LDPC码所具有的性能和实用价值。
- LDPC码可以用非常稀疏的校验矩阵或二分图来描述，也就是说LDPC码的校验矩阵的矩阵元除一小部分不为0外，其它绝大多数都为0。通常我们说一个(n, j, k) LDPC码是指其码长为n，其奇偶校验矩阵每列包含j个1，其它元素为0；每行包含k个1，其它元素为0。j和k都远远小于n，以满足校验矩阵的低密度特性。(思考一下为什么要低密度->防止错误传递)
- LDPC码所面临的一个主要问题是其较高的编码复杂度和编码时延。
- 写PPT的人对LDPC码，只是知道概念和原理，没有过多的研究，因为不喜欢，LDPC在数学上毫无美感，可能写PPT的人对这部分还没有悟道！！！！

极化码(Polar码，菠萝码)原理介绍—1

快来看啊，上帝的奇迹啊！极化码在数学上完美的像一个奇迹。

- 假设如左下的信道，我们可以认为每一个输入比特(u_1和u_2）的成功译码概率为p，那么这两个比特都是一样的概率p，p是由信道决定的，关键点在一样！！！



- 但是如b方案的处理，我们把(u_1和u_2）做一下处理再输入，那么根据(y_1和y_2）成功解码出(u_1和u_2）是不是还一样呢？（我们假设p为0.5） 见证奇迹的时刻到了：

✓ 对于u_1根据y_1和y_2解码出的概率C_1和C_2都为p，那C_1和C_2错误的概率都是1-p. 那么u_1就是C_1和C_2的比特加，只其中一个比特错误， u_1就会错误，所以u_1成功概率是概率是p^2，就是0.25

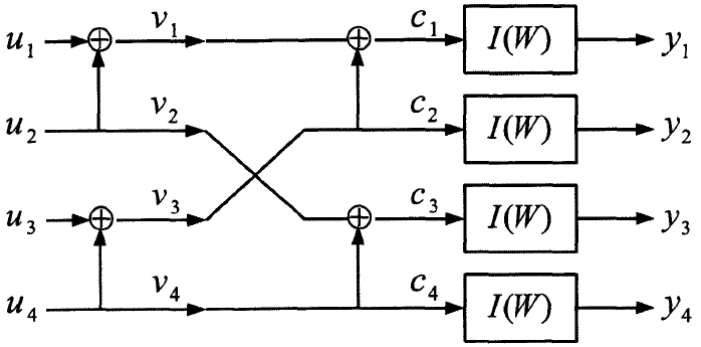
✓ 对于u_2根据y_1和y_2解码出的概率C_1和C_2都为p，那C_1和C_2错误的概率都是1-p. 但是如果C_2发生错误的话，那么u_2通过C_2的译码就会错，但是！！！！！ u_2根据u_1和C_1还可以译码。，他有两条路可走（条件概率），只要C_1和C_2都错， u_2才会错，他们都错的概率为$(1-p)^2$, 所以u_2的译码概率为[1- $(1-p)^2$]还可以, 如果p为0.5，这个值是0.75

✓ u_1和u_2的概率夹在一起平均一下是多少： (0.25 + 0.75)/2就是0.5, 还是0.5

NOKIA

极化码(Polar码，菠萝码)原理介绍—2

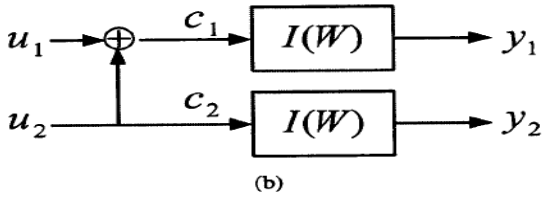
- 如果我们再扩展一下，用4个值，进一步迭代，会发生什么



- 好吧，我直接给结论，有兴趣的可以自己推算一下，P还是0.5：
  - ✓ u_1的概率为 0.0625 u_2的概率为 0.4375 u_3的概率为 0.5625 u_4的概率为 0.9375
- 随着N的增加，我们看出信道已经发生了极化分裂：好的越来越好，差的越来越差
  - ✓ 数学上可以证明：当N趋向无穷大的时候，有些位置解码成功概率无限趋向于1，有些则为无限接近0，达到1位置的比例为p个……而且每一个位置的解码概率都是可以算出来的，只要p给定。伟大啊.
  - ✓ 就是说，实际上极化码把损失放到固定的位置上让有些位置越来越好。在实际应用上就是在不好的位置上填0，在好的位置上放发送的信息，太聪明了

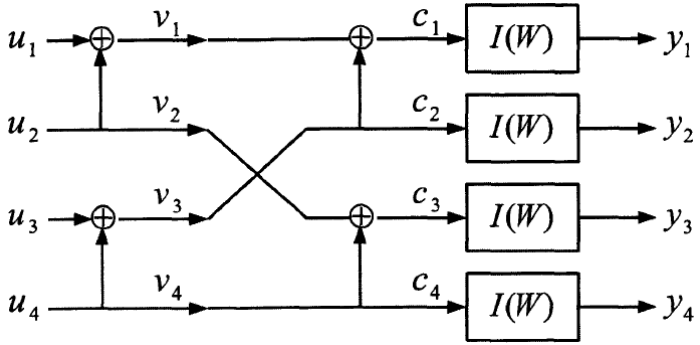
极化码(Polar码，菠萝码)原理介绍—3 （换一个角度辩证的看问题）

- 假设如我们再从另外一个角度去看下面的图



(b)

- 从上面的图我们可以发现 u_1和u_2的确是不平等的，因为在进入信道之前（发送）， u_2的能量已经进入了u_1，就是发送的c_1和c_2都带有u_2 的能量，而看看下面的图，当然可以发现，实际发送的bit，发送的时候包含谁的能量/信息最多呢？

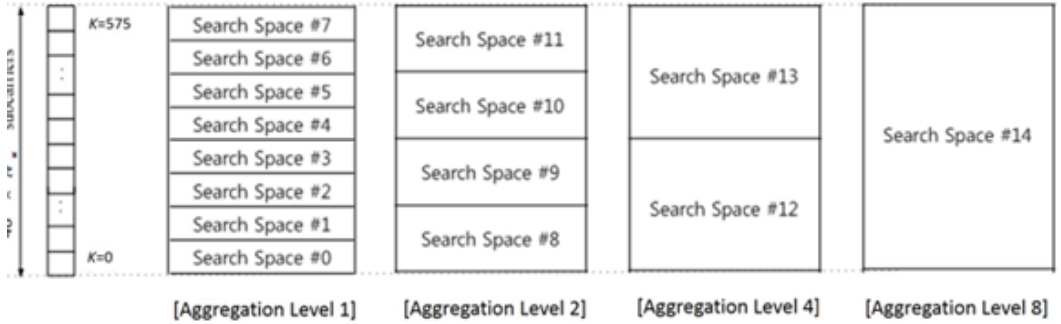


- 我们就能理解为什么出现极化了

极化码(Polar码，菠萝码)原理介绍—4

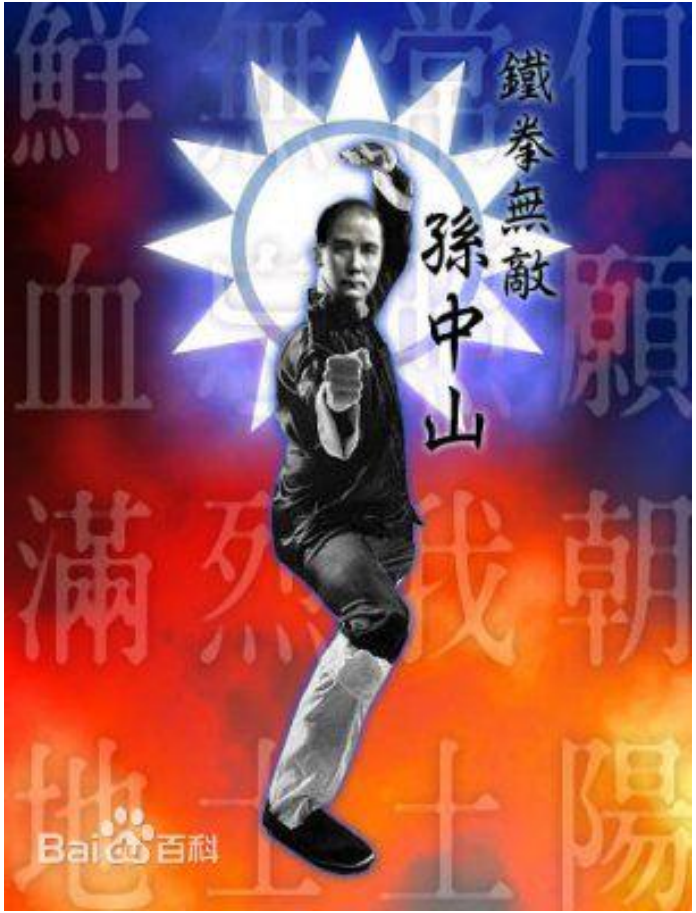
- 当极化码用于控制信道的时候，我们一般都要谈到基础N的值，我们知道N越大，极化现象越明显，也就是说同样传输K个比特，从N为1024，找到的K个位置，要比N为256中找到的K个位置，解码成功概率高的多

- 比如说，在LTE中，下行PDCCH,使用的是咬尾卷积码，不同的Agg level，就是传送所用的CCE的数量，从下面图可以看出，CCE越多，资源越多，那么所发送的信息就有更多次的重复



- 在5G中，CCE和Agg level的概念是一样，但是在原理上，更多的资源，意味和发送的基础N越大
- Notes:基础N越大，和重复次数越多，实际上是一回事情，意味着所用来传送信息的能量更多

如果对前面的内容，都已经吃透了，那么恭喜，再加上一点傅里叶变换的基础和OFDM系统的基础，那么技术已经没有任何阻碍，我们就可以开始物理层解析之旅了，可以练习神功了



第一章：随机接入信道和过程

第一节：PRACH物理信道

随机接入的过程(简单描述，不要紧，后面还有更加精细的描述)

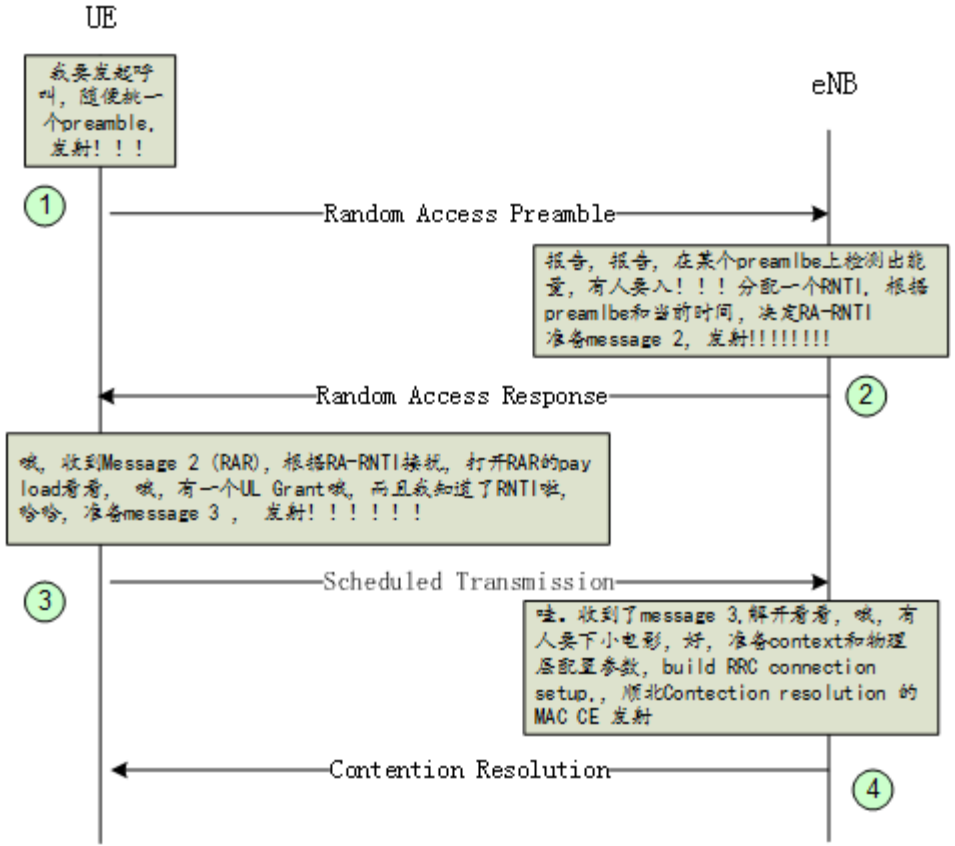
- UE在下列这几种情况下，发起随机接入的过程

  ✓用户发起呼叫 / 切换 / 用户在TA失步 /被PDCCH order触发，无线链路失步，Beam tracking失步的情况下发起重建

  ✓整个随机接入过程在做面的图中描述，只是一个简单的描述，随后有更加具体的介绍

- 头脑风暴一下，这里再从另外一个角度来理解一下随机接入

  其实RACH随机接入的作用是完成接入网络时的上行同步当且仅当上行定时OK、UE取得上行同步之后，UE才会被eNB调度上行时频资源并进行上行数据的传输的。所以，当UE未取得或已丢失与基站的上行同步时，就一定会牵扯到随机接入过程了。因为RACH的重要作用便是用于取得上行时间同步。而反过来说，一旦UE获得上行同步，UE就可以占用PUCCH等信道申请资源（SR/BSR），进而eNB也就可以调度上行的正交的SC-FDMA传输资源给UE，UE才可能传输数据。



上行随机接入信道-1 序列的产生1

- 上行随机序列的产生通过左面的公式完成 $x_{u,v}(n) = x_u((n + C_v) \bmod L_{RA})$

$$x_u(i) = e^{-j\frac{\pi u i(i+1)}{L_{RA}}}, i = 0,1,\ldots,L_{RA} - 1$$

- 从这个公式看，明显是有一个Z-C序列，就是奇怪随机序列中第二个。
- $L_{RA} = 839$或者139 则是序列的长度，从原则上说序列越长，小区覆盖范围就越大。所以序列长度139，是对应覆盖范围很小的小区的
- u指定了序列的根，u（*PRACHRootSequenceIndex* ）值在3GPP中由38.211中的6.3.3.1-3中定义很多值（某一个小区任意指定一个或者多个序列），然后这个根序列是在系统消息中广播的
- C_v 的值则是用户自己指定的(实际上就是preamble ID)，关于这个值在下一页中解释

上行随机接入信道-1 序列的产生2

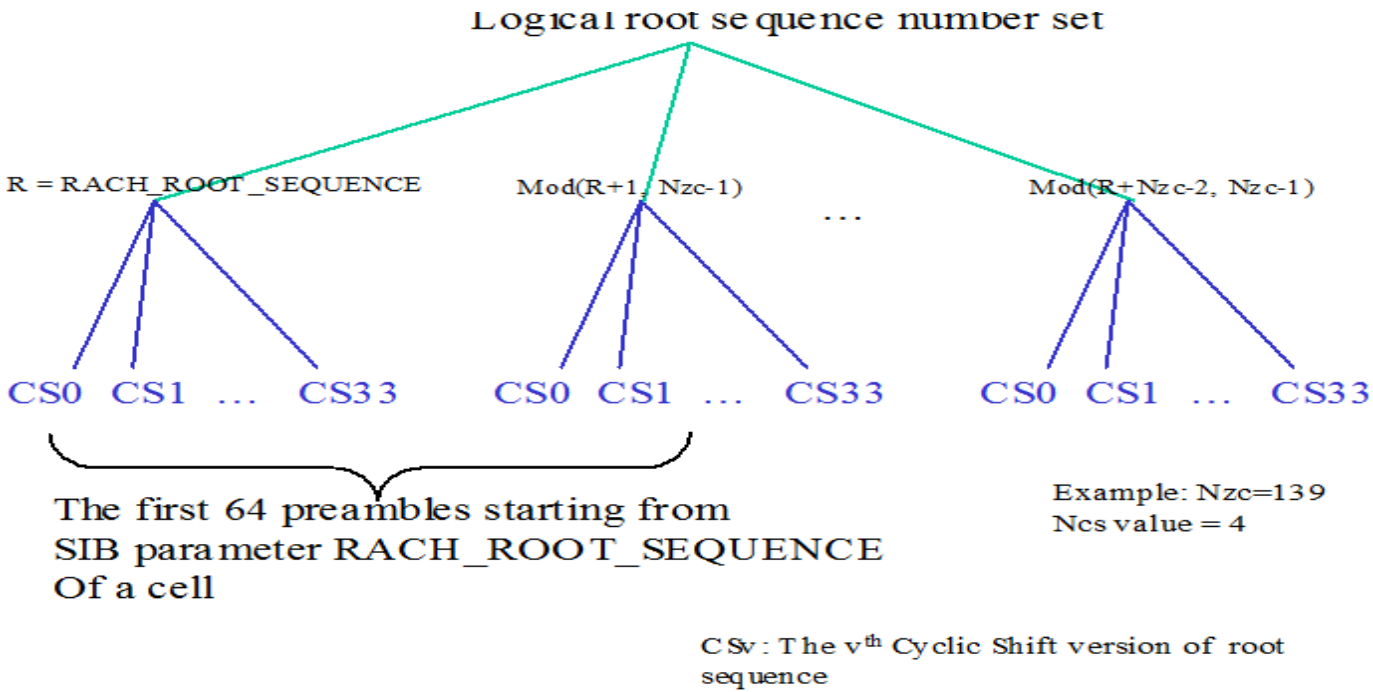
- C_v 的值在3GPP中的定义吓死个人，足足有3-4页的内容，但是不要怕，主要的复杂在于restrictedSetConfig A和B的情况，这是针对于高速移动的场景（目前Nokia不支持）
- 那么我们只谈论最简单的配置，如下
- C_v 就是 vN_{CS} ，也就是v个(v为1,2,3,….)个Ncs的数字
- 那么这个n乘以Ncs就是Preamble ID的值，整个preamble的逻辑的分析如下，
  - ✓ 如果一个小区要使用64个preamble，使用时在其中选取一个进行接入，64个preamble的产生是首先使用一个ZC根产生一个839的序列，然后通过Ncs参数对这个序列进行循环移位，如果移位步长较大而不够64个preamble，则再拿一个根序列的ZC序列进行循环移位（也可以就算了，少一点preamble不会死人的），直到满足个数要求。这么做的原因是不同的循环位移步长和小区接入半径有关，所以有不同的Ncs参数，Ncs是通过系统消息广播下来的。最初选择的根也是通过配置下来的。
  - ✓ 简单理解：例如在Ncs=1的情况下（当然不可能），序列01011100110，表示0号preamble，往右循环移位1位00101110011表示1号，往右循环移位1位10010111001表示2号…
  - ✓ 从原理上来看，如果Ncs的值比较大，那么preamble之间的间隔比较大，对于检测有好处，抗时偏的能力强，可以支持更大覆盖的小区，但是大的Ncs值，使得可以用的preamble数量变少。

上行随机接入信道-1 Ncs取值的分析-1

- 在前面已经讨论过，Ncs和小区半径有关，例如在LTE中小区半径为10公里的话，Ncs的取值为93。一个经验公式是$N_{cs} > 1.05 * (20 * r/(3) + 9)$，其中r为小区半径，单位为公里
- 对于Ncs为93的值，那么839长度的根序列，可以支持的preamble的个数为839/93,向下取整为9个preamble.所以为了支持64个preamble ID，我们一共需要8个根序列。这8个根序列在后面一页PPT表述
- Ncs所有可以用的值，在3GPP 38.211中表6.3.3.1-5和6.3.3.1-6中

上行随机接入信道-1 Ncs取值的分析-2

- 在下图中给出了Ncs为4的例子

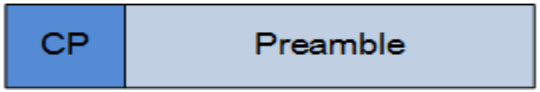


上行随机接入信道-PRACH的格式-1

- 在3GPP中，38.211表6.3.3.1-1和2定义了PRACH的不同格式，那么这些格式的作用是什么呢。我们首先看一下这些序列长度是839的格式，TTI长度1ms，15K子载波间隔的RACH格式

Format	L_{RA}	Δf^{RA}	N_u	N_{CP}^{RA}	Support for restricted sets
0	839	1.25 kHz	24576κ	3168κ	Type A, Type B
1	839	1.25 kHz	$2 \cdot 24576\kappa$	21024κ	Type A, Type B
2	839	1.25 kHz	$4 \cdot 24576\kappa$	4688κ	Type A, Type B
3	839	5 kHz	$4 \cdot 6144\kappa$	3168κ	Type A, Type B

- 一个PRACH的格式如右，这里先忘掉CP是什么，后面会讲。



从这里可以看出格式0定义Preamble占有24576个k，也就是大约800us。（$T_c = 1/(\Delta f_{max} \cdot N_f)$），其中 Δf_{max} 是480000，N_f 是4096，一个k是64个Tc，所以很容易得到一个k是32.552ns

同理CP长度为3168k，也就是103.1us，那么1ms（1000us）还留下多少时间：还有（1000 - 103.1 - 800） = 96.9us。

那么96.9us，乘以c(光速)，也就是29公里不到，我们知道这种配置最大支持14公里的小区，

因为UE不知道自己离开基站的距离，下行发送到上行接受，一共需要96us的时间

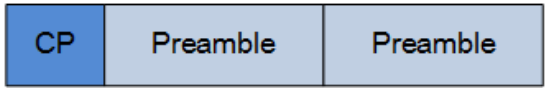
上行随机接入信道-PRACH的格式-2

- 我们再来看第二种格式，Format

这种format对应至少3ms一次

的RACH 长度， 其格式如下

Format	L_{RA}	Δf^{RA}	N_u	N_{CP}^{RA}	Support for restricted sets
0	839	1.25 kHz	24576κ	3168κ	Type A, Type B
1	839	1.25 kHz	$2 \cdot 24576\kappa$	21024κ	Type A, Type B
2	839	1.25 kHz	$4 \cdot 24576\kappa$	4688κ	Type A, Type B
3	839	5 kHz	$4 \cdot 6144\kappa$	3168κ	Type A, Type B



- 同理可得，两个preamble一共占据1600us，CP有684.4us，一共需要2284.4us，如果3ms一次，可以支持107km
- Preamble发射两次相当于增强3db的覆盖

上行随机接入信道-PRACH的格式-3

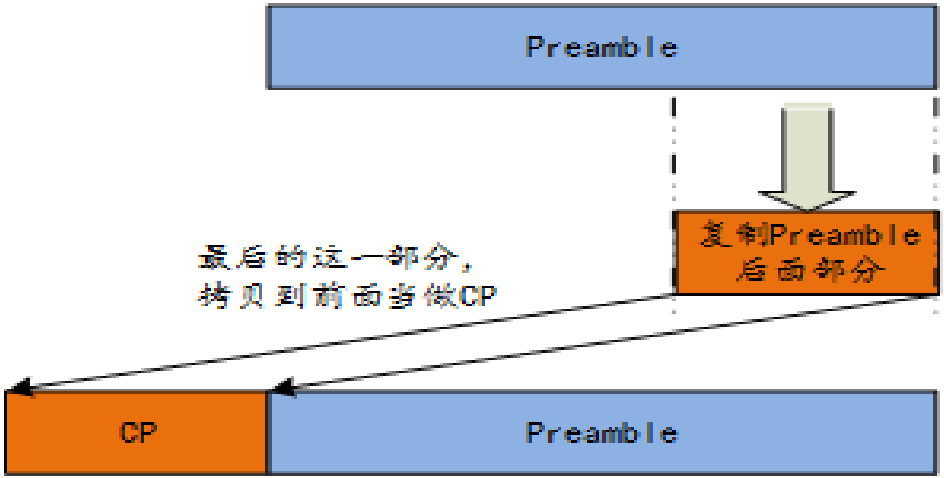
- 对于30K,60k..的子载波间隔，3GPP定义的表如下

Format	L_{RA}	Δf^{RA}	N_u	N_{CP}^{RA}	Support for restricted sets
A1	139	$15 \cdot 2^{\mu}$ kHz	$2 \cdot 2048\kappa \cdot 2^{-\mu}$	$288\kappa \cdot 2^{-\mu}$	-
A2	139	$15 \cdot 2^{\mu}$ kHz	$4 \cdot 2048\kappa \cdot 2^{-\mu}$	$576\kappa \cdot 2^{-\mu}$	-
A3	139	$15 \cdot 2^{\mu}$ kHz	$6 \cdot 2048\kappa \cdot 2^{-\mu}$	$864\kappa \cdot 2^{-\mu}$	-
B1	139	$15 \cdot 2^{\mu}$ kHz	$2 \cdot 2048\kappa \cdot 2^{-\mu}$	$216\kappa \cdot 2^{-\mu}$	-
B2	139	$15 \cdot 2^{\mu}$ kHz	$4 \cdot 2048\kappa \cdot 2^{-\mu}$	$360\kappa \cdot 2^{-\mu}$	-
B3	139	$15 \cdot 2^{\mu}$ kHz	$6 \cdot 2048\kappa \cdot 2^{-\mu}$	$504\kappa \cdot 2^{-\mu}$	-
B4	139	$15 \cdot 2^{\mu}$ kHz	$12 \cdot 2048\kappa \cdot 2^{-\mu}$	$936\kappa \cdot 2^{-\mu}$	-
C0	139	$15 \cdot 2^{\mu}$ kHz	$2048\kappa \cdot 2^{-\mu}$	$1240\kappa \cdot 2^{-\mu}$	-
C2	139	$15 \cdot 2^{\mu}$ kHz	$4 \cdot 2048\kappa \cdot 2^{-\mu}$	$2048\kappa \cdot 2^{-\mu}$	

- 同理，我们可以算出，对于A1的配置，也是要发两次preamble,加上CP的时间，对于30k的子载波(0.5 TTI)，需要71us，大家什么感觉，很短是吧，是的，这里导致了5G的RACH和4G有很大的不同

上行随机接入信道-PRACH的格式-4 CP的处理1

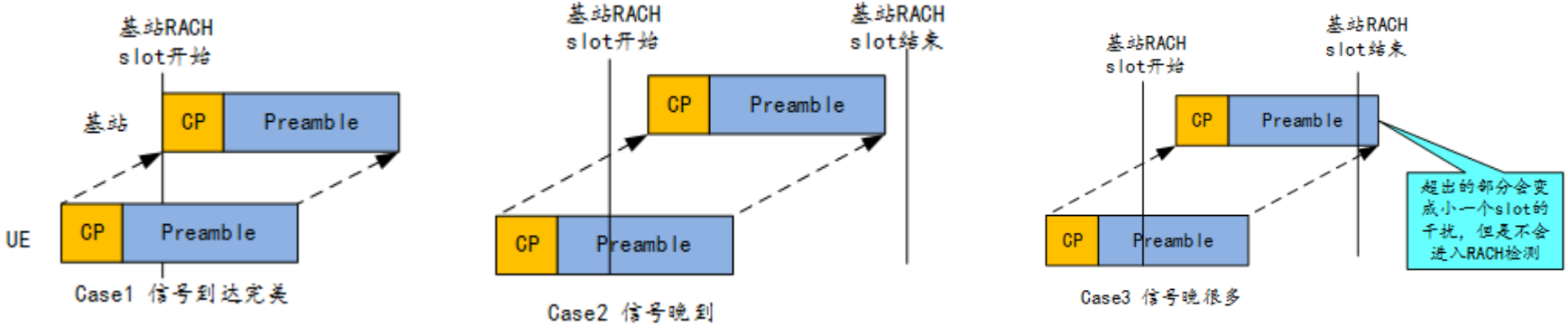
- CP的长度在RACH的配置，有很重要的意义，RACH的CP和普通PDSCH和PUSCH的CP不一样。PDSCH和PUSCH的CP是每一个symbol单独加的，RACH的CP在整个信号时间上展开，如下图



- 在做分析之前，我们先给出结论，小区覆盖的范围越大，CP越长，所以我们可以对比一下，发现5G的RACH CP比LTE短很多，高频的RACH CP要比低频的RACH CP短很多

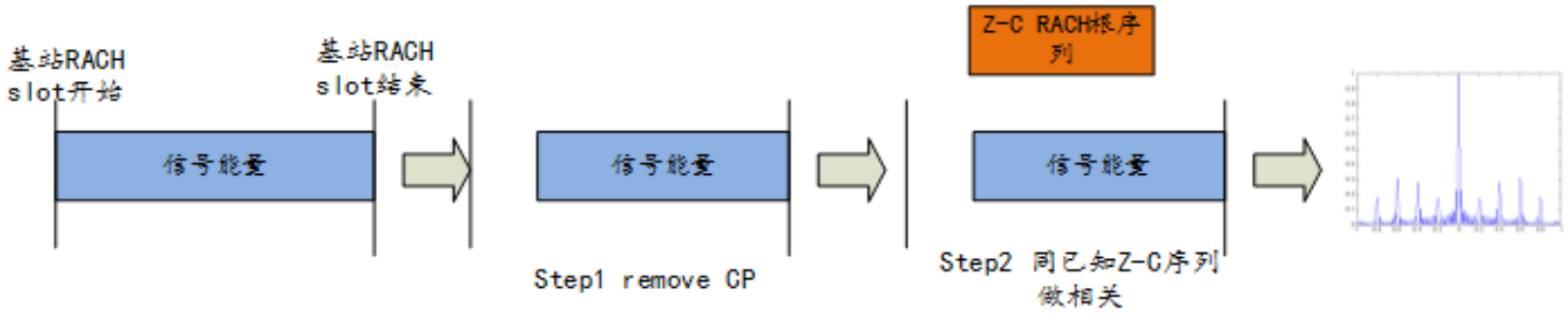
上行随机接入信道-PRACH的格式-4 CP的处理2

- CP的作用主要是为了抗时偏，如下边的图case1所示，这是最Perfect的case，因为信号CP加上preamble,完美的达到TTI/slot的边界。但是现实往往不这么完美
- 手机在接入之前，手机只知道下行的时间同步，就是只知道电磁波传到手机侧，手机知道哪里是下行TTI的开始，但是手机并不知道自己距离基站的距离，所以在上行发起RACH的时候，信号一般情况下会晚点，例如右边case2所示
- 比糟糕的情况如下面的case 3，但是实际上基于CP的帮助，case2和case3都能解析出，除了类似于case3，超出大大多于CP部分的情况（实在太晚了，这种情况，小区覆盖要重做了。）

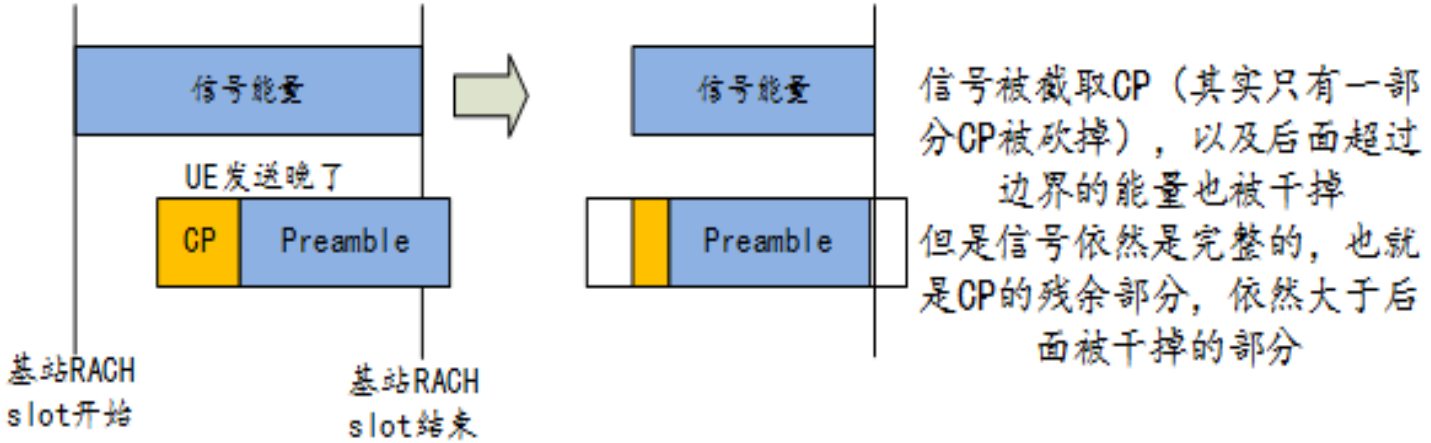


上行随机接入信道-PRACH的格式-4 CP的处理3

- 我们先来看，基站是如何处理RACH的，最简单的处理情况如下图



- 即使信号晚到了，基站是依然能检测出来的，因为有CP的帮助，具体过程看下图



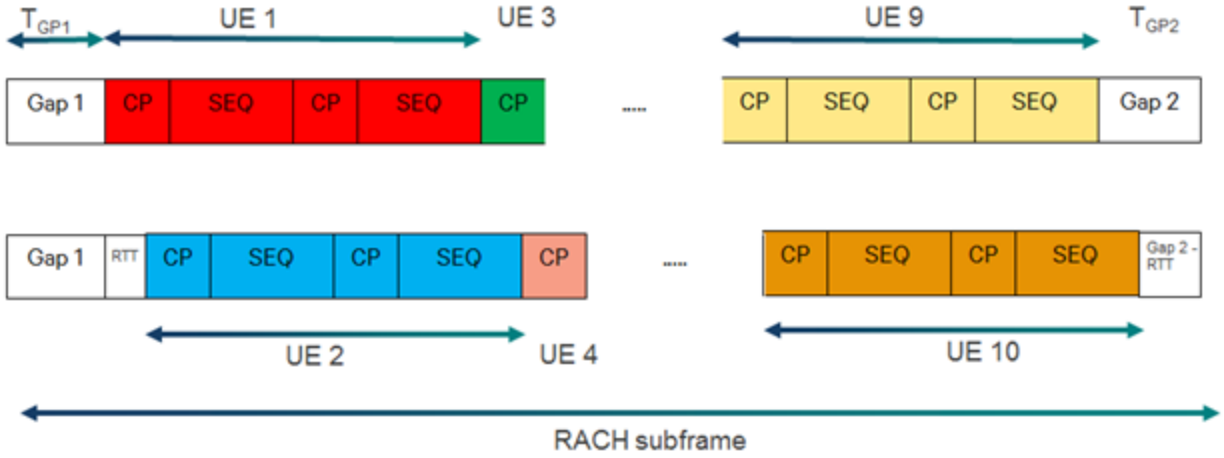
上行随机接入信道-PRACH的格式-4 CP的处理4

- 对于晚到的Preamble，为了便于理解，我们举一个不恰当的例子
  - ✓ 如果发送的preamble是：马英九是一个几乎没有缺点的人
  - ✓ 如果CP长度是6，那么整个发送的的序列是：没有缺点的人马英九是一个几乎没有缺点的人
  - ✓ 如果整个序列晚到了3个字符的时间，后面被截掉一个字符，那么基站能检测到的序列是

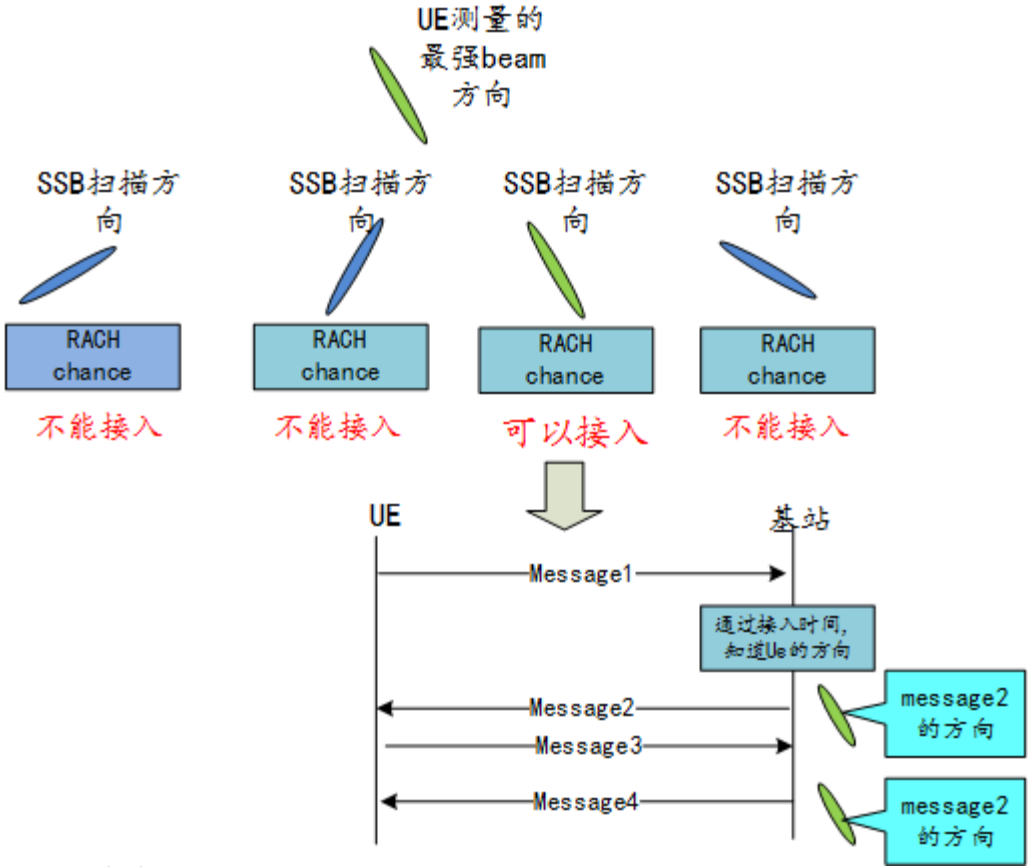
    点的人马英九是一个几乎没有缺点的
- 即使信号晚到了，我们看到因为CP的保护，信号依然是完整的，但是整个信号在相位上发生的旋转。这是很有意思的事情
- 因为RACH序列，NCS的值本来就是一个相位的旋转，只要这个CP的保护，引起的相位旋转，也就是检测波峰的地方发生了偏移，如果移动没有超出Ncs的间隔，基站依然能正常检测出preamlbe,而这个相位的旋转，基站当然能够发现，这个相位差就是时间差，想想RAR中的TA是如何获得的！！哈哈
- RACH检测在算法上很关键，有很多论文和研究，很多系统在实现上并不是从前面讲的完美位置开始检测，而是推后CP/2开始检测的。有兴趣的可以研究，好玩的

上行随机接入信道-PRACH的格式-5

- 在4G LTE中，每一个RACH的帧1ms或者2ms，通过前面的分析，我们可以知道，基本在时域上，整个CP+Preamlbe的时间占据了大部分的时间
- 在5G中，一个CP+Preamble只是占据一个TTI的很短的时间，如下图，所以在symbol等级，可以划分更加细的资源（有点点mini slot的感觉，但只是感觉上，概念上不完全是）
- 所以我们就能理解在RA RNTI要加入symbol id的原因



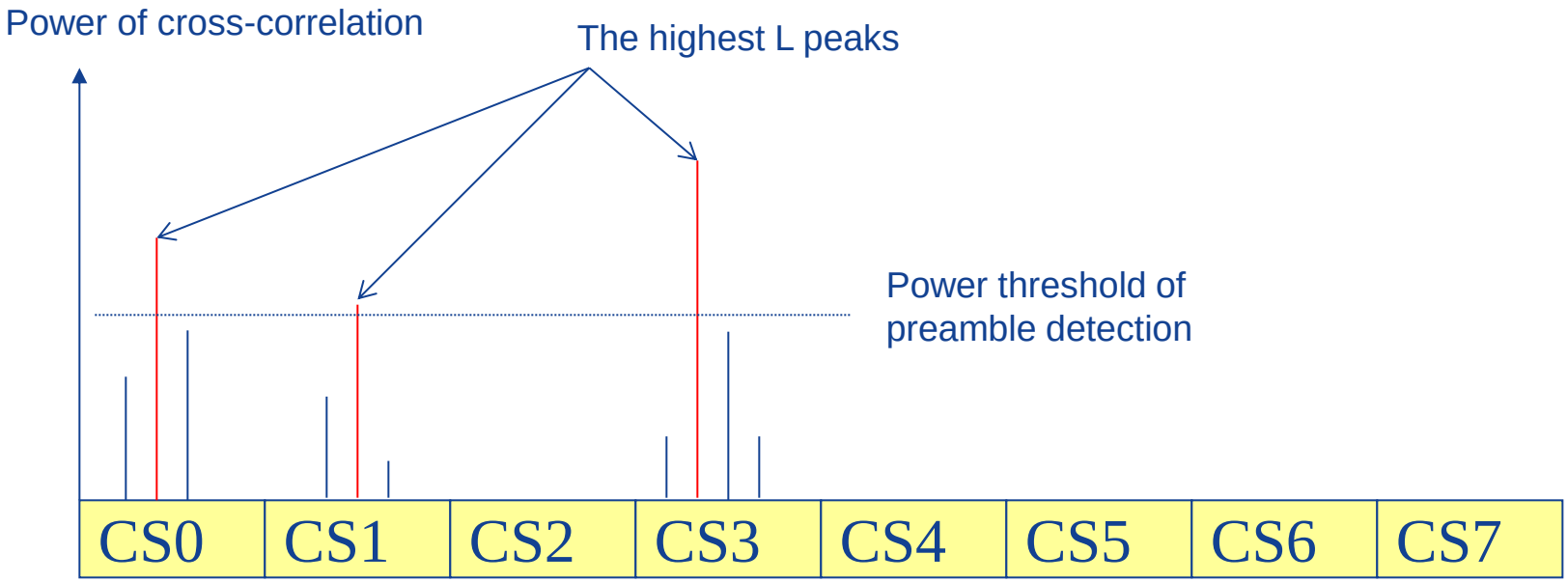
上行随机接入信道-发送的时间



- 5G的RACH和4G LTE相比在接入时机上，有很大不同，主要原因是Beam forming引起的。
  - ✓ 在任何一个RACH的机会就能够接入
  - ✓ 一直要等到SSB扫描到相同的beam方向的时候(slot)才能接入，因为不然基站不知道message2发哪个方向，也就是这里发送RACH的时机，隐含的指示了UE所处的方向角（这部分内容，在讲SSB的时候，会详细讲），下图会给出简要说明

NOKIA

上行随机接入信道 基站侧RACH 检测的例子



- 那么在这一组结果中，我们会选Cs3这个preamble

第一章：随机接入信道和过程

第二节：RACH接入处理过程

接纳是
最好的温柔

随机接入第2步 – RAR

- 当UE发送了preamble以后，在一个时间窗内监听PDCCH,检测RAR,这个窗口由ra-ResponseWindow 定义，在系统消息中广播
- 这个由ra-ResponseWindow指定的时间窗开始于RACH preamble发送后的第一个PDCCH的起始时间
- 如果在时间窗口内没有检测到RAR,说明基站不理你，则UE重发preamble,并且增加发射功率
- 如果基站捕获到preamble，则应该立即组RAR消息并下发
  - ✓ RNTI：新分配的RNTI
  - ✓ TA:通过preamble检测发现UE的时间偏差，并通过RAR中的TA command通知UE进行纠正
  - ✓ UL grant（message 3的grant）
  - ✓ Backoff indication：请手机滚蛋，暂时不要接入，表示网络正忙（因为不回Backoff indication的话，也不理手机）,手机会不断的重发preamble. Backoff机制用来阻止手机在一定时间内接入，属于拥塞控制的机制

随机接入第2步 - RA RNTI的概念

- 如果Ue在RACH的资源上发送了Preamble以后，网络捕获到了能量，并且要发送message2 (RAR)，而RAR是通过RA RNTI进行加扰的，
- RA RNTI在3GPP 38.321中的定义为

  RA-RNTI= 1 + s_id + 14*t_id + 14*X*f_id + 14*X*Y*ul_carrier_id

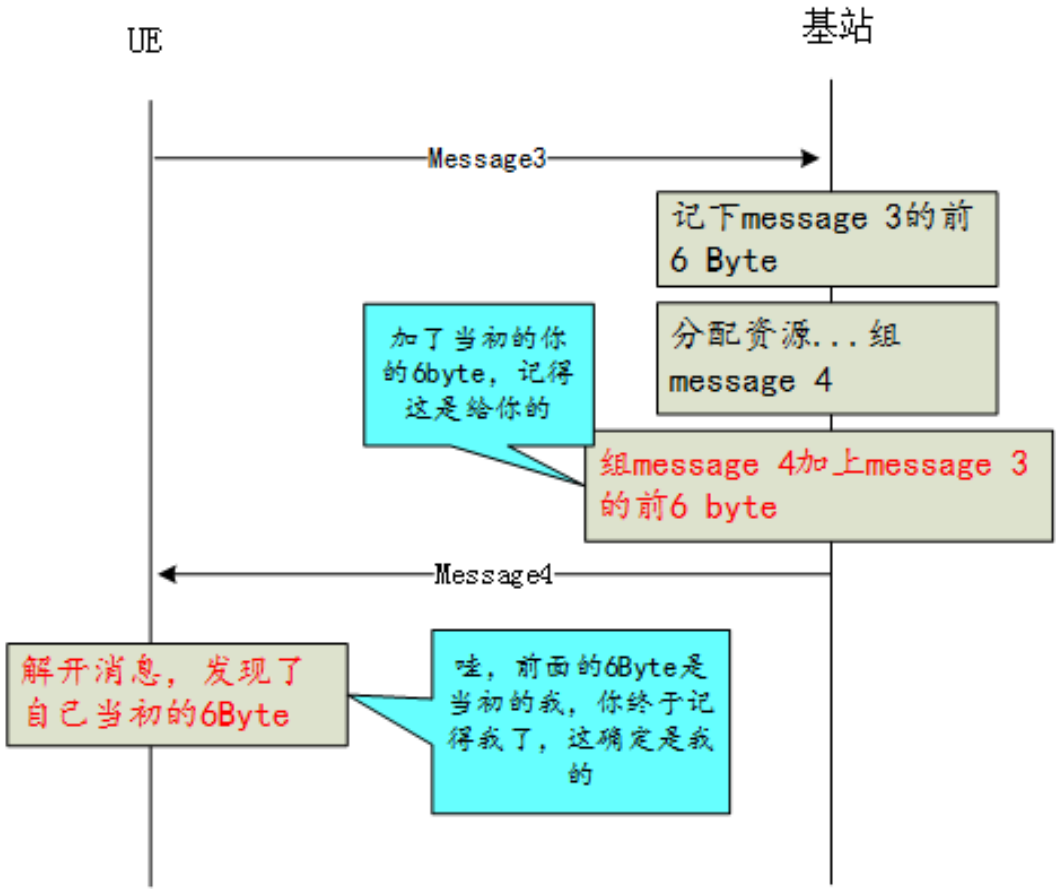
  - ✓ S_id是发送RACH preamlbe所对应的开始的symbol号
  - ✓ t_id是发送RACH preamlbe在一个10ms系统帧中所对应slot的号
  - ✓ f_id是发送RACH preamlbe所对应的第几个频率资源（理论上，一个TTI可以配多个RACH的PRB,但是一般一个就够了）
  - ✓ ul_carrier_id是载频id
  - ✓ X，Y定义在38,213中，对不起我没有找到，哈哈，有谁发现就告诉我（应该有一个是preamble吧）。
- 无论怎样RA RNTI实际明确指定了PRACH message 1的频率，和时间以及所用的preamble. 这个是UE和gNB都心知肚明的，彼此都知道，所以RAR用RA RNTI加扰以后，不是自己的RAR是解不出来的(CRC KO的)
- RAR是用过RA RNTI进行加扰的

随机接入第3步 - Message3的发送

- 当UE获取RAR以后，按照RAR中的UL grant的指示，发送message 3
  - ✓ 对于初始接入，message 3中的payload是RRC connection request（这条RRC消息中，包含身份标示IMSI和接入原因-establish cause）
  - ✓ 对于切换，message 3中的payload是RRC connection re-configuration complete
  - ✓ 对于失步以后的接入，message 3中的payload主要是RNTI
- 当发送message 3以后，
  - ✓ UE启动contention resolution Timer，等待contention问题的解决
  - ✓ 这里有一个理解的误区，认为这个Timer是在等待Message 4（其实不完全是，但是大部分情况下是….逻辑绕吧，因为contention的问题解决大部分情况下是在Message 4的MAC PDU中携带的MAC CE解决的），关于contention的问题，后面专门讨论
  - ✓ 如果contention resolution Timer超时，UE会重新启动RACH过程，重发preamble

随机接入第四步 - Message4的处理

- 当基站接受到了Message 3以后，U plane拔RRC消息转交给C-plane，经过ASN.1的decode以后，Cplane解析这条RRC消息，根据其中的内容，如果是初始接入的话，一般在U-plane上建立Ue的context以及分配相应的物理资源和配置，然后这些物理资源和logical channel等配置要通知给UE，这些信息就包含在Message 4中
- 如果Contention的问题还没有解决（这个topic会后面详细讲），Message 4中必须带有Contention resolution的MAC CE,而这个Contention resolution就是所对应Message 3的钱6个Byte.这个过程如右图



第一章：随机接入信道和过程

第三节：RACH竞争问题解决过程

随机接入Contention问题 -1---也就是解决你是谁的问题

- 从概念上来讲，RACH有两种方式：
  - ✓ 基于Contention的(Contention based)，就是竞争接入的

    网络基站并不知道UE要接入，主要用于初始接入和失步以后的接入

    特点是手机随机挑选preamble，然后尝试接入，网络并不知道这个试图接入的用户是谁，会分配相应的RNTI

  - ✓ Contention free的接入，非竞争的接入

    基站知道用户要接入，主要用于切换，RNTI和preamble都是事先分配的，在源小区通过RRC reconfig通知用户。

- 从小区的RACH preamble资源的分配来看，竞争接入的preamble和非竞争的preamble的数量和范围，是在小区SIB中广播的。

随机接入Contention问题 -2---如何解决你是谁的问题

- 对于非竞争的随机接入，因为preamble是事先分配的，而且是唯一的，随机接入的时候，如果Target preamlbe被捕获，则认为基站已经知道该来的人来了，下发msg 2 RAR以后，当UE用所对应的RA RNTI成功解出，则UE认为网络已经认出自己了，则Contention的问题在message2的时候解决

- 对于基于竞争的随机，我们在前一页已经已经讨论了，在RAR下发的时候，手机不能确定，已经可能这时候有两个人使用了同样的preamble.手机还不能确定就是自己，所以发送message 3(可能另外一个人也在发送message3)，等到message4到达的时候，通过Contention resolution的MAC CE同手机当初的message3前6byte一致的时候，UE认为Contention的问题解决

  - ✓ 从3GPP的定义来说，UE不必等到message 4，只要收到Contention resolution的MAC CE就可以认为问题解决了，手机就 不会再发RACH了,所以这里有个好玩的玩法，当message4来不及的时候，UP可以先发这个MAC CE，告知手机，知道了，让手机等一下

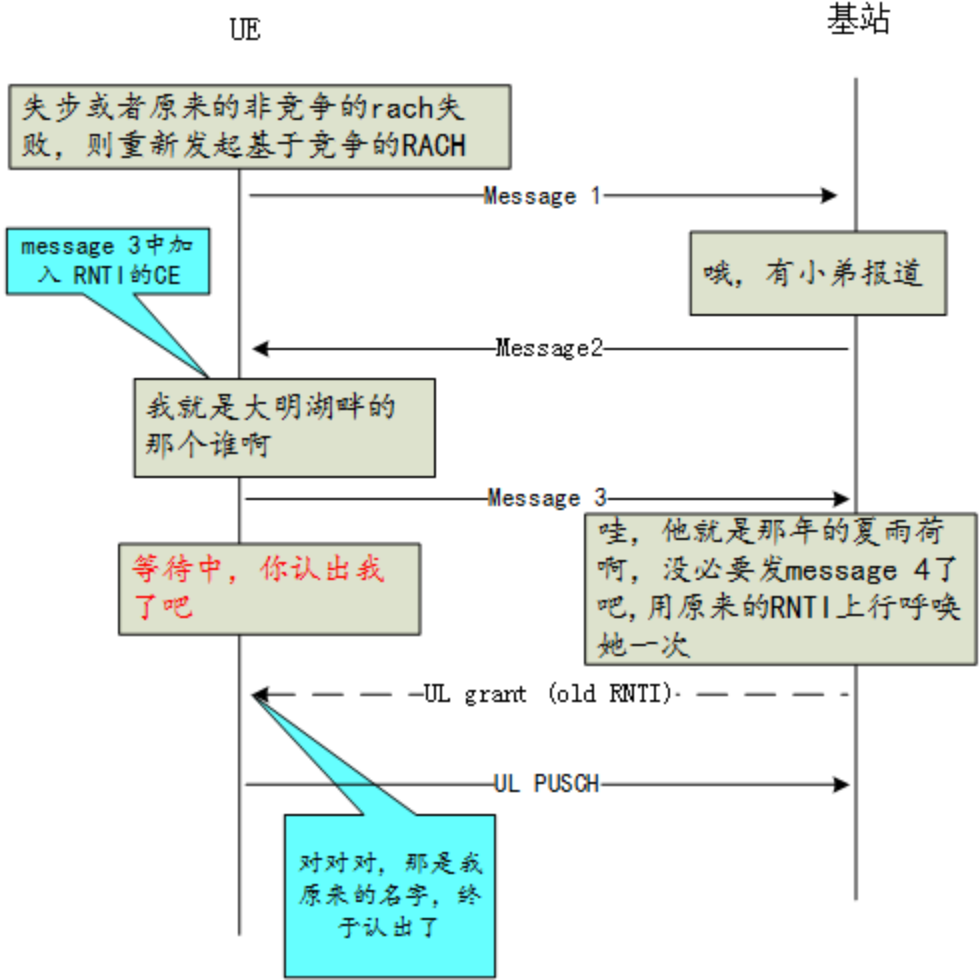
随机接入Contention问题 -3---曾记得你是谁，但是如今沧海难为水，那怎么办

- 这里有两个特殊的情况：
  - ✓ 从基于非竞争的随机接入过程，如果接入失败，则UE会转入基于竞争的随机接入
  - ✓ 如果用户检测到失步，就必须重新发起基于竞争的随机接入
- 但是在逻辑上，这里有一个矛盾：
  - ✓ 一旦发起基于竞争的随机接入，基站不再知道“你是谁”，比如切换时候用的非竞争的接入，这时候用户Context和preamble都已经好了，一直在原来的preamble上等她，但是UE换了一个马甲（preamble），基站是不可能知道她就是原来的她
  - ✓ 而如果UE探测到失步，主动发起随机的接入，基站并不知道，她就是曾经的她
- 解决知道是在message3中，把原来的马甲(RNTI)报上来。具体描述看一下页

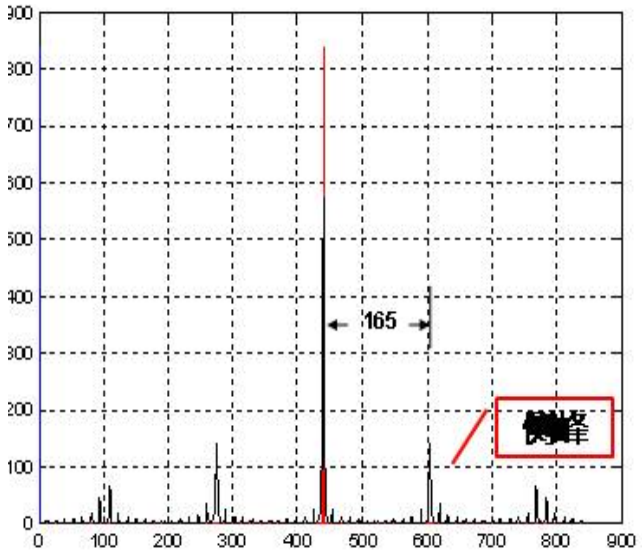
NOKIA

随机接入Contention问题 -4--曾记得你是谁，但是如今沧海难为水，那怎么办（续）

- 从右图可以看出两点
  - ✓ Contention问题的解决是通过在message3中加入RNTI的MAC CE来解决的
  - ✓ 用户一定要等到基站用其原来的RNTI进行一次上行的调度， 才算Contention问题的真正解决



上行随机接入信道-PRACH的格式-最终战（菊水特攻）--1



- 在法国TGV或者中国高铁的情况下，列车时速达到350km以上，如果是5G 3.5G的频段，上行多普勒效应所造成的频偏已经达到2.3K，（伟大的多普勒频偏，结合红移，揭示了宇宙从大爆炸开始的）这种疯狂的情况下怎么办呢？3GPP中吓死人的Cv计算大部分在处理这种情况

- 在出现了多普勒频偏以后，我们看出preamble的检测情况出现了什么？如左图：基站会在频域做检测的时候，出现多个峰值，侧峰在 d_u 的整数倍的位置出现，当频偏交大的时候，侧峰会超过主峰，造成严重虚检，很神奇的事情是，这个du居然是能算出来的，就是产生侧峰的地方，这个和Z-C序列的特性有关，但是我不知道为什么，只知道结论，du的定义为：

先定义q： 其中q为(p*u) Mode L = 1 的情况下，最小的一个整数。L为RACH序列的长度，u为序列的根

$$d_u = \begin{cases} q & 0 \le q < L_{\mathrm{RA}}/2 \\ L_{\mathrm{RA}} - q & \text{otherwise} \end{cases}$$

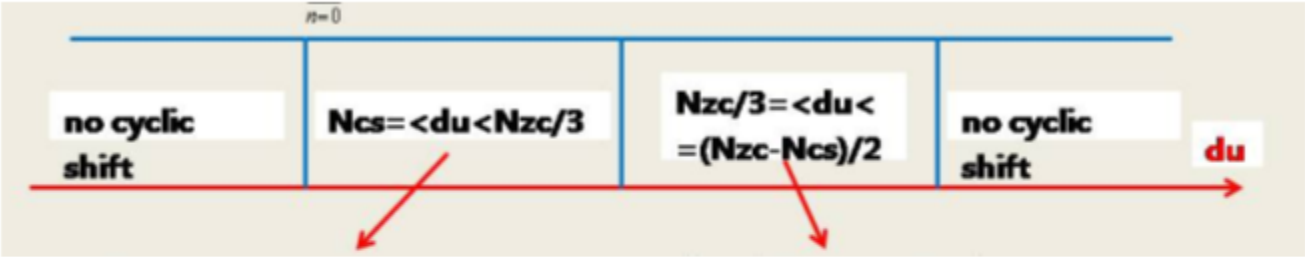
上行随机接入信道-PRACH的格式-最终战（菊水特攻）--2

Physical u	du
1	1
2	419
3	280
4	210

415	93
416	240
417	336
418	279
419	2
420	2
421	279
422	336
423	240
424	93

835	210
836	280
837	419

- 对于不同的u,在839点的L的长度来看，相对应的du如左图所见，Du的值以419，420为界限，呈镜像对称。对于不同的du值，在du处会产生侧峰，所以相同的du值，采取的方式是一样的。
- 在有了du以后，我们要算出四个中间值（下面给出的公式是对应于restricted set type A,是分两段的，而restricted set type B分六段）。



$$n_{shift}^{RA} = \lfloor d_u / N_{CS} \rfloor$$
$$d_{start} = 2d_u + n_{shift}^{RA} N_{CS}$$
$$n_{group}^{RA} = \lfloor L_{RA} / d_{start} \rfloor$$
$$\bar{n}_{shift}^{RA} = \max\left(\lfloor (L_{RA} - 2d_u - n_{group}^{RA} d_{start}) / N_{CS} \rfloor, 0\right)$$

$$n_{shift}^{RA} = \lfloor (L_{RA} - 2d_u) / N_{CS} \rfloor$$
$$d_{start} = L_{RA} - 2d_u + n_{shift}^{RA} N_{CS}$$
$$n_{group}^{RA} = \lfloor d_u / d_{start} \rfloor$$
$$\bar{n}_{shift}^{RA} = \min\left(\max\left(\lfloor (d_u - n_{group}^{RA} d_{start}) / N_{CS} \rfloor, 0\right), n_{shift}^{RA}\right)$$

NOKIA

上行随机接入信道-PRACH的格式-最终战（菊水特攻）--3

- 好了，我们有四个中间变量了，看看3GPP对Cv完整的定义是什么
- 可以看出通过这四个中间变量，我们可以算出高速环境下的所有preamble

$$C_v = \begin{cases} vN_{\text{CS}} & \text{v} = 0,1,...,\lfloor L_{\text{RA}}/N_{\text{CS}} \rfloor - 1, N_{\text{CS}} \neq 0 \quad \text{for unrestricted sets} \\ 0 & N_{\text{CS}} = 0 \quad \text{for unrestricted sets} \\ d_{\text{start}}\lfloor v/n_{\text{shift}}^{\text{RA}} \rfloor + \left(v \bmod n_{\text{shift}}^{\text{RA}}\right)N_{\text{CS}} & v = 0,1,...,w-1 \quad \text{for restricted sets type A and B} \\ \bar{\bar{d}}_{\text{start}} + (v-w)N_{\text{CS}} & v = w,...,w+\bar{\bar{n}}_{\text{shift}}^{\text{RA}} - 1 \quad \text{for restricted sets type B} \\ \bar{\bar{\bar{d}}}_{\text{start}} + \left(v-w-\bar{\bar{n}}_{\text{shift}}^{\text{RA}}\right)N_{\text{CS}} & v = w+\bar{\bar{n}}_{\text{shift}}^{\text{RA}},...,w+\bar{\bar{n}}_{\text{shift}}^{\text{RA}}+\bar{\bar{\bar{n}}}_{\text{shift}}^{\text{RA}} - 1 \quad \text{for restricted sets type B} \end{cases}$$

$$w = n_{\text{shift}}^{\text{RA}} n_{\text{group}}^{\text{RA}} + \bar{n}_{\text{shift}}^{\text{RA}}$$

上行随机接入信道-PRACH的格式-最终战（菊水特攻）--3

- 下面这张表给出了所有有效根序列的，4个中间变量和产生的preamble的个数

Logical root sequence number	Logical root sequence number	Physical root sequence number	Ncs	du	Ncs=<du<Nzc/3					Nzc/3=<du<=(Nzc-Ncs)/2					other value	结果	以u作为起始号，产生64个码需要的连续的根序列数量
					Nshiftra	Dstart	Ngroupra	Nshiftra-	产生的接入码数量	Nshiftra	Dstart	Ngroupra	Nshiftra-	产生的接入码数量	no cyclic shift		
0-23	0	129	15	13	0	26	32	0	0	54	1623	0	0	0	#VALUE!	#VALUE!	
	1	710		13	0	26	32	0	0	54	1623	0	0	0	#VALUE!	#VALUE!	
	2	140		6	0	12	69	0	0	55	1652	0	0	0	#VALUE!	#VALUE!	
	3	699		6	0	12	69	0	0	55	1652	0	0	0	#VALUE!	#VALUE!	
									…								
	22	1		1	0	2	419	0	0	55	1662	0	0	0	#VALUE!	#VALUE!	
	23	838		1	0	2	419	0	0	55	1662	0	0	0	#VALUE!	#VALUE!	
24-29	24	56		15	1	45	18	0	18	53	1604	0	1	1	#VALUE!	18	4
	25	783		15	1	45	18	0	18	53	1604	0	1	1	#VALUE!	18	5
	26	112		412	27	1229	0	1	1	1					#VALUE!	14	5
	27	727		412	27	1229	0	1	1	1					#VALUE!	14	5
	28	148		17	1	49	17	0	17	53					#VALUE!	17	5
	29	691		17	1	49	17	0	17	53					#VALUE!	17	5
30-35	30	80				1223	0	1	1	1					#VALUE!	11	5
	31	759				1223	0	1	1	1	36	11	0	11	#VALUE!	11	5
	32	42				55	15	0	15	53	1594	0	1	1	#VALUE!	15	5
	33	797		20	1	55	15	0	15	53	1594	0	1	1	#VALUE!	15	5
	34	40		21	1	57	14	0	14	53	1592	0	1	1	#VALUE!	14	5
	35	799		21	1	57	14	0	14	53	1592	0	1	1	#VALUE!	14	5
									…								
810-815	810	309		410	27	1225	0	1	1	1	34	12	0	12	#VALUE!	12	5
	811	530		410	27	1225	0	1	1	1	34	12	0	12	#VALUE!	12	5
	812	265		19	1	53	15	0						1	#VALUE!	15	5
	813	574		19	1	53	15	0						1	#VALUE!	15	4
	814	233		18	1	51	16	0						1	#VALUE!	16	4
	815	606		18	1	51	16	0						1	#VALUE!	16	5
816-819	816	367		16	1	47	17	0						1	#VALUE!	17	
	817	472		16	1	47	17	0	17	53	1602	0	1	1	#VALUE!	17	
	818	696				1227	0	1	1	1	32	12	1	13	#VALUE!	13	
	819	543				1227	0	1	1	1	32	12	1	13	#VALUE!	13	
820-837	820	336				1239	0	0	0	0	5	83	0	0	#VALUE!	#VALUE!	
	821	503		417	27	1239	0	0	0	0	5	83	0	0	#VALUE!	#VALUE!	
									…								

根序列取值范围

这部分落选了

根序列u能够产生的接入码数量

以u为起始号产生64个前导码所需根序列总数

上行随机接入信道-PRACH的格式-最终战（菊水特攻）--4

- 我们给出不同的u，以及在Ncs 15的情况下，我们可以看到有时候Preamble是等间隔的，有时候是不等间隔的

u=56

v	Ncs	Nshiftra	Dstart	Cv	间隔
0				0	
1				45	45
2				90	45
3				135	45
4				180	45
5				225	45
6				270	45
7				315	45
8	15	1	45	360	45
9				405	45
10				450	45
11				495	45
12				540	45
13				585	45
14				630	45
15				675	45

Cv集合有18个值，Nshiftra =1，等间隔

u=28

v	Ncs	Nshiftra	Dstart	Cv	间隔
0				0	
1				15	15
2				90	75
3				105	15
4				180	75
5				195	15
6				270	75
7				285	15
8	15	2	90	360	75
9				375	15
10				450	75
11				465	15
12				540	75
13				555	15
14				630	75
15				645	15

Cv集合非等间隔

上行随机接入信道-PRACH的格式-最终战（菊水特攻）--5

- 我们对于高铁的应用，做出的网络规划，可以考虑一下原则
  - 首先选择对应的Ncs,这个Ncs首先考虑小区半径
  - 然后选择根序列
    - ✓ 对应不同的Ncs,选择一个可用根序列的取值范围
    - ✓ 以可用的起始根序列，遍历所有的根序列，选取产生N个preamlbe所需要的最大的连续根序列的值MAX_K
    - ✓ 相邻小区根序列以MAX_K为间隔，避免相邻小区根序列冲突

第二章： PUSCH以及相关信道和处理

第一节：PUSCH信道和处理过程

PUSCH后端处理第一步（refer to 38.211内的6.3.1.1）： 加扰(Scrambling)

- 加扰的目的

  ✓防止出现长0和长1，因为对于长0和长1，对于编码和解码都是有坏处的

- 加扰要跳过ACK/RI出现的位置(skip place holder bit):

  ✓假设我们有一串PUSCH比特（调制前），为 $b^{(q)}(0),...,b^{(q)}(M_{\text{bit}}^{(q)}-1)$ 如果是ACK/RI或者其repetition的位置，预留比特 $\tilde{b}^{(q)}(i)$ 为1

  ✓对于其他位置加扰后的比特为信息比特和加扰特比的异或操作:

  $$\tilde{b}^{(q)}(i)=\left(b^{(q)}(i)+c^{(q)}(i)\right)\text{mod}$$

  ✓ $c^{(q)}(i)$ 就是前面说的两个奇怪随机序列中的第一个(Golden序列)，其根为

  $c_{\text{init}}=n_{\text{RNTI}}\cdot 2^{15}+n_{\text{ID}}$ 其中如果*Data-scrambling-Identity使用的的话*

  n_{ID} *就是Data-scrambling-Identity不然* $n_{\text{ID}}=N_{\text{ID}}^{\text{cell}}$

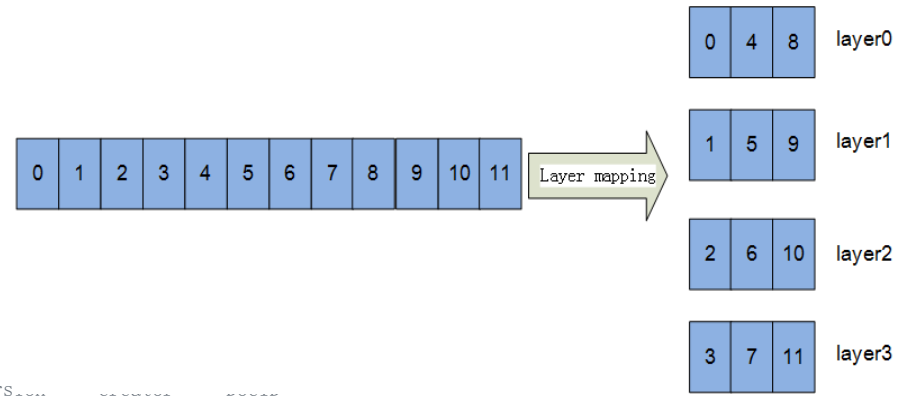
PUSCH后端处理第二步（refer to 38.211内的6.3.1.2）： 加扰(Scrambling)

- 调制方式，上行PUSCH的调制由PDCCH DCI0确定，由网络层决定
- 经过了调制以后，我们要发送的就从bit变成了矢量符号
- 注意上行可以到256QAM,这种高级货
- Panzer：comment：为什么，Transform pre-coding enable不能用BPSK

Transform precoding disabled	Transform precoding enabled
	π/2-BPSK
QPSK	QPSK
16QAM	16QAM
64QAM	64QAM
256QAM	256QAM

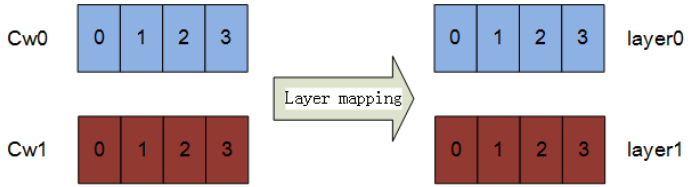
PUSCH处理第三步（refer to 38.211内的7.3.1.3）：层映射-1（DL层映射一样）

- 层映射中有两个基本概念
  - ✓ 层的概念：所有的层(layer)依然在同一组物理天线的所有震子上发送，层实际上是指手机和基站间能够被区分的不同信道，手机能检测出来多少个层，由手机通过Rank indication在PUCCH上报
  - ✓ Codeword的概念，是指调度器所能调度出来的TB块，一个或者两个，四个TB块分别映射到同一个时频资源的不同层上，Codeword的数量总是小于层的数量的。
- 第一个例子：一个codeword的数据映射到4个layer上，如下图描述
- 映射到2,3层，逻辑一样

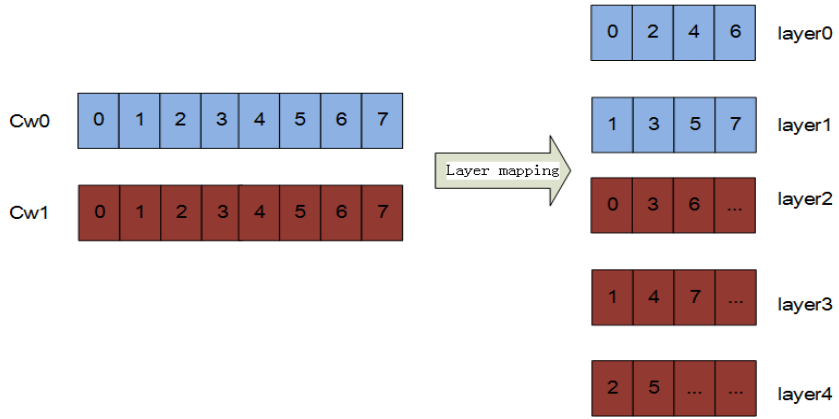


PUSCH后端处理第三步（refer to 38.211内的7.3.1.3）：层映射-2

- 第二个例子：两个codeword的数据映射到两个layer上，如下图描述



- 第三个例子：两个codeword的数据映射到五个layer上(不对称的模式)，如下图描述



NOKIA

PUSCH后端处理第四步（refer to 38.211内的6.3.1.4）：传输预编码(Transform precoding)

- 这个处理在2017年11月的规范讨论中才出现，是华为的提案,是3GPP处理的可选项(Nokia目前物理层实现不支持)，针对每一层的符号其处理流程如下：

$$y^{(0)}(l \cdot M_{sc}^{PUSCH} + k) = \frac{1}{\sqrt{M_{sc}^{PUSCH}}} \sum_{i=0}^{M_{sc}^{PUSCH}-1} x^{(0)}(l \cdot M_{sc}^{PUSCH} + i) e^{-j\frac{2\pi ik}{M_{sc}^{PUSCH}}}$$

$$k = 0,\ldots, M_{sc}^{PUSCH} - 1$$

$$l = 0,\ldots, M_{symb}^{layer} / M_{sc}^{PUSCH} - 1$$

- 实际上就是对要发送的信号，做了一个DFT(离散的傅里叶变换)：
  - ✓ 进行DFT的处理是为了PAPR的问题，因为把符号能量做归一化处理，只在上行处理，是因为PAPR对手机的发射影响更大
  - ✓ 在4G LTE中，这是必选项，因为LTE主要是广覆盖的宏站
  - ✓ 在5G中，是可选项，因为5G在讨论初期都是小覆盖的站，但是后来的讨论发现并非如此

PUSCH后端处理第五步（refer to 38.211内的6.3.1.5）：预编码矩阵

- 在3GPP中，预编码矩阵的处理如下，讲各个逻辑天线端口的符号，左乘一个预编码的矩阵

$$\begin{bmatrix} z^{(p_0)}(i) \\ \vdots \\ z^{(p_{\rho-1})}(i) \end{bmatrix} = W \begin{bmatrix} y^{(0)}(i) \\ \vdots \\ y^{(\upsilon-1)}(i) \end{bmatrix}$$

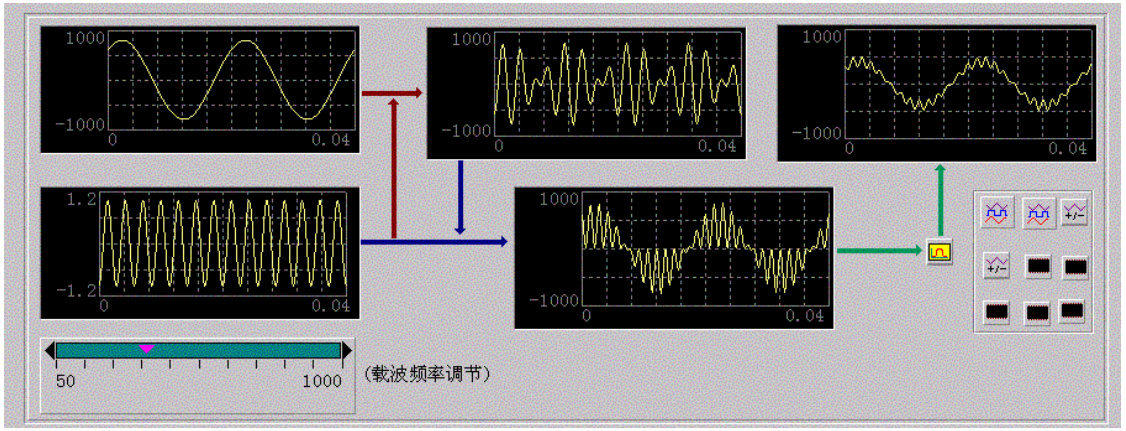
- 各个预编码矩阵在3GPP中38.211中，由表6.3.1.5.1到表6.3.1.5.1定义，右边类似于如下2个CW，4层的某一个矩阵（注意和LTE的不同，没有PMI）

$$\frac{1}{2}\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -j \end{bmatrix}$$

- 进行预编码矩阵处理的原因是如下，因为任何接受信号的过程为
  - ✓ R = $H * S + n$，信号响应H在多个层的情况下是一个矩阵，
  - ✓ 任何一个H可以表示为H = U * E * V,其中E为单位矩阵 $\begin{matrix} 1 & \\ & 1 \end{matrix}$
  - ✓ 而上述的预编码矩阵就是U的逆矩阵，这样手机的H就变成$H = U^T * U * E * V$，就是$H = V$,这样手机做信道估计处理起来方便

第二章： PUSCH以及相关信道和处理

第二节：PUSCH的DRMS



PUSCH的DMRS，解调参考信号的作用

- 对每一个Symbol，每一个子载波上， 接受到的信号为： $R = H * S + n$
  - ✓ 其中S为对端发射的信号，R是收到的信号
  - ✓ H为信道相应函数
  - ✓ n为噪音
- 而解调参考信号的作用就是为了估计出H. 其原理是
  - ✓ 从$R = H * S + n$ ，有两个未知数H和S，n暂时忽略不计，已知的只有R
  - ✓ 为了估计出H,只能用一组已知信号，就是DMRS，使得S是已知的，然后根据其所对应的R，解出H来，然后把H当做其他位置上的H，从而把用户的PUSCH数据解出

PUSCH的DMRS，解调参考信号-1(参考信号的生成方式1—不支持传输预编码方案)

- DMRS的产生公式有两种，定义在3GPP 38.211 -- 6.4.1.1.1， 第一种在不支持那个传输预编码的情况下：

$$r(m)=\frac{1}{\sqrt{2}}(1-2\cdot c(2m))+j\frac{1}{\sqrt{2}}(1-2\cdot c(2m+1))$$

- 而其中的r(m)是一个Golden序列，其根定义为

$$c_{\text{init}}=\left(2^{17}(14n_s+l+1)(2N_{\text{ID}}^{n_{\text{SCID}}}+1)+2N_{\text{ID}}^{n_{\text{SCID}}}+n_{\text{SCID}}\right)\bmod 2$$

  - ✓ 其中 l 是一个slot内的Symbol号，看其出现在哪个Symbol，
  - ✓ n_s 就是slot号
  - ✓ $n_{\text{SCID}}\in\{0,1\}$ 以及 $N_{\text{ID}}^{n_{\text{SCID}}}\in\{0,1,\ldots,65535\}$ 由高层配置参数UL-DMRS-Scrambling-ID 指定，如果没有这个参数的话

$$n_{\text{SCID}}=0 \text{ 并且 } N_{\text{ID}}^{n_{\text{SCID}}}=N_{\text{ID}}^{\text{cell}}$$

PUSCH的DMRS，解调参考信号-2(参考信号的生成方式2—支持传输预编码方案)

- 第二种在支持那个传输预编码的情况下，在DMRS的产生公式为，啊哈，一个Z-C序列，我们奇怪序列中的第二个（Nokia目前不支持）：

$$r^{(p_0)}(m) = r_{u,v}^{(\alpha,\delta)}(m)$$

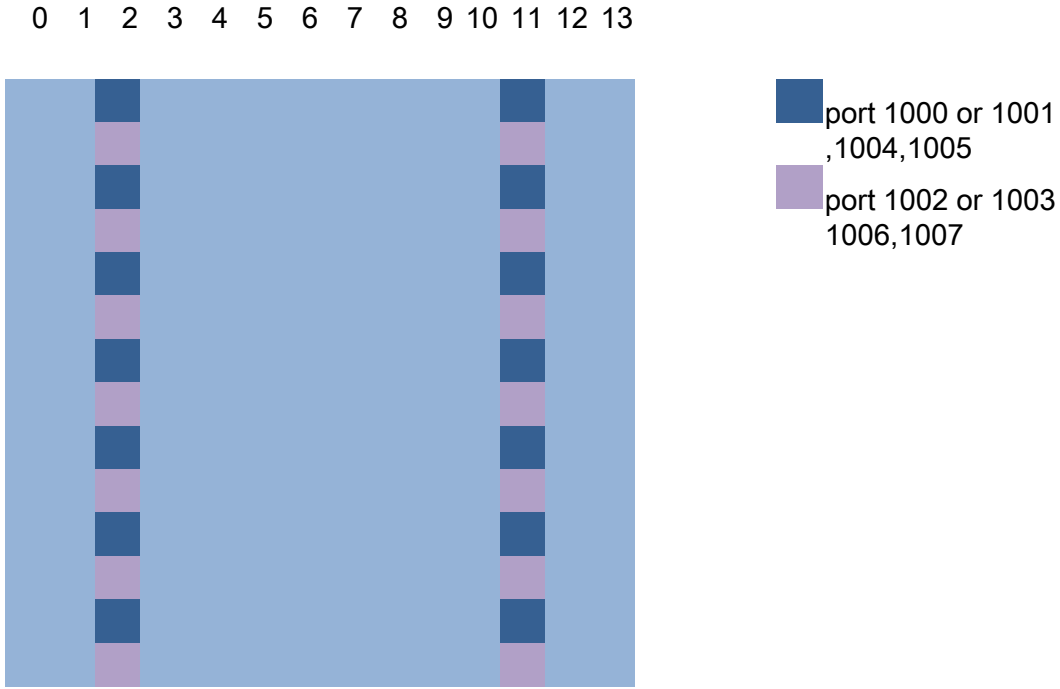
- 而其中生成序列各个参数的定义为
  - ✓ δ 始终为0
  - ✓ u,v以及 α 的取值，3GPP还在讨论中，看后续版本的更新了

PUSCH的DMRS，解调参考信号-3 参考信号的预编码矩阵

- 参考信号的预编码矩阵和其所对应的PUSCH的预编码矩阵是一样的，必须的
- 因为上行的参考信号是用来给基站做信道估计用的，也就是算出$R = H * S + n$中的H，所以对DMRS的预编码矩阵的处理，必须和PUSCH一致

PUSCH的DMRS，解调参考信号-4-2 参考信号的资源映射1

- 下图给出最简单的PUSCH的资源位置，注意凡是DMRS不用的地方，就是PUSCH的位置



PUSCH的DMRS，解调参考信号-4-2 参考信号的资源映射2（refer to 38.211内的6.4.1.1.3）

- 参考信号的资源映射在3GPP中定义为如左的公式，不要怕，一点一点讲

$$a_{k,l}^{(p_j,\mu)} = \beta_{\text{DMRS}} w_{\text{f}}(k') \cdot w_{\text{t}}(l') \cdot \tilde{r}^{(p_j)}(2n+k')$$

$$k = \begin{cases} 4n + 2k' + \Delta & \text{Configuration type 1} \\ 6n + k' + \Delta & \text{Configuration type 2} \end{cases}$$

$$k' = 0,1$$

$$l = \bar{l} + l'$$

- $w_{\text{f}}(k')$和$w_{\text{t}}(l')$ 这两个值，取决于序列的位置，port和符号位置

- $\tilde{r}^{(p_j)}$ 就是我们前面讲的DMRS生成的序列，而从这个 $\cdot\tilde{r}^{(p_j)}(2n+k')$ 我们可以看出这个序列被分成了奇数和偶数两个序列，当然 $k'=0$就是这个序列的偶数部分，为第0,2,4…个比特或者符号，反之 $k'=1$ 就是他的奇数部分

- 而Δ 是用来区分不同的port口，在port 1000和1001,其值取0，而1002和1003其值取1

- K决定了DMRS在子载波上的位置，

  - ✓ 那么对于configure 1这种配置，port 1000和1001， 其在子载波上的位置为
    - 序列中的偶数符号（ $k'=0$),位置为：0,4,8……序列中的奇数符号位置为：2,6,10……
    - 同样port1002和1003，偶数符号位置为1,5,9，奇数符号位置为3,7,11…

PUSCH的DMRS，解调参考信号-4-4 参考信号的资源映射4（refer to 38.211内的6.4.1.1.3）

- 参考信号的资源在时间上的位置由 l 指定，其定义为：$l = \bar{l} + l'$
  - ✓ 对于DMRS在时间上的位置，取决于下列高层配置
    - DM-RS configuration type、PUSCH mapping type，PUSCH symobols
    - DMRS-add-pos
    - single-symbol or double-symbol
  - ✓ 对DM-RS configuration type和DMRS-add-pos等决定了 $\bar{l}$ ， 通过3GPP 38.211表6.4.1.1.3-3和6.4.1.1.3-4
  - ✓ Single 还是Double symbol决定了 l' 是0还是1

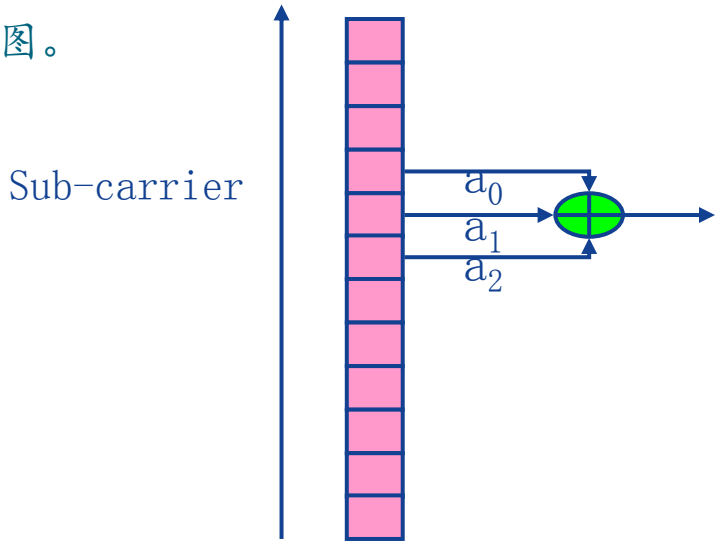
PUSCH mapping type A，DMRS-add-pos是1，PUSCH 使用14个symbol

那么PUSCH出现在2（如果 $l_0 = 2$ ），和11这两个symbol上

PUSCH的DMRS，解调参考信号-5 参考信号的使用-1

- 对5G UL DRMS的算法还在研究中，为了先加深理解，暂且给出LTE对UL DMRS的使用，估计都差不多，等待以后更新、反正都高度近似
- 因为接受到的信号R = H*S+n，所以第一步做信道估计的时候，对于DRMS因为S是已经知道的，所以直接H = R/S，而n是高斯白噪声
- 第二部要做FDA (Frequency domain average),就像左下图。

  对于相邻的子载波的H，做一下平滑



PUSCH的DMRS，解调参考信号-5 参考信号的使用-2

- 对于如果有两列以上的PUSCH DRMS，因为已经得到了每一列的H（H_0和H_1），可以做时域上的内插，来得到每一列的H，比如说，在某一个子载波上H_0的值是4.1， H_1的值是5.9。后一个子载波H_0的值是5.2， H_1的值是6.1 （当然H理论上是一个虚数，但是为了简单，我们理解成浮点数）
  - ✓ 则这两个子载波上每一个symbol的H分别为如下图
  - ✓ 时域上的内插反应的是，默认为信道在时间上的变化是连续的

0	1	2	3	4	5	6	7	8	9	10	11	12	13
3.7	3.9	4.1	4.3	4.5	4.7	4.9	5.1	5.3	5.5	5.7	5.9	6.1	6.3
5	5.1	5.2	5.3	5.4	5.5	5.6	5.7	5.8	5.9	6	6.1	6.2	6.3

  - ✓ 如果只配置了一列的DMRS，则我们可以理解为因为slot过于的短，信号在时间上没啥变化

PUSCH的DMRS，解调参考信号-5 参考信号的使用-3（估计信噪比）

- PUSCH的DMRS还可以用来估计UL的SINR，我们UL调度器中的SINR上报，就是从DMRS中估计出来的，对于第 i 个PRB上噪音的能量为

$$\sigma_i^2 = \frac{1}{2 \cdot 12 \cdot P} \cdot \sum_{p=0}^{P-1} \sum_{l=2,11} \sum_{k=i \cdot 12}^{i \cdot 12+11} \left| H_k^p(l) - \hat{H}_k^p \right|^2$$

  - ✓ 其中的 $H_k^p(l)$，就是在频域平滑和时域平滑之前H值，
  - ✓ 其中的 $\hat{H}_k^p$ ， 就是在做过时域和频域平滑以后的H值

  从上面的公式，我们可以看出，实际上这个噪音能量的估计，就是H的抖动幅度，从原理上说，噪音因为是高斯白噪声，其能量越大，估计出来的H的抖动就越大，H的抖动范围直接就是噪声功率

PUSCH的DMRS，解调参考信号-5 参考信号的使用-4（信道均衡）

- 收到的信号为$Y = H * S + n$，如果从DMRS获得了H，为了获得S(UE发送的信号)，则S能够从下列公式得出 $S = (H^*Y)/|H|^2$从来得到

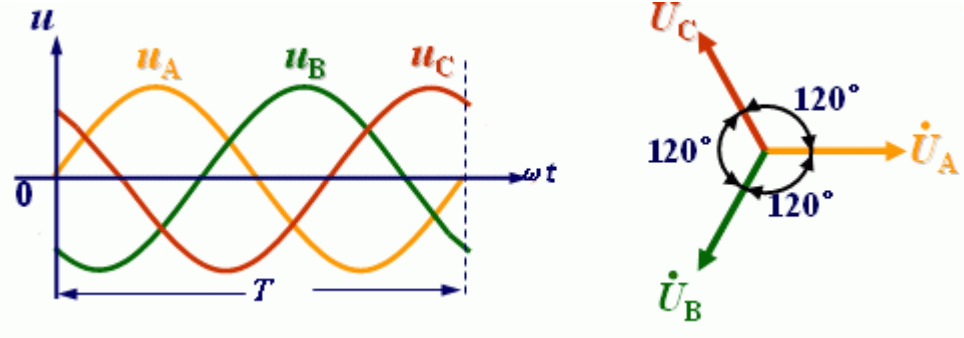
- 两个Code word可以从左边的公式得到. $\tilde{Y}_k = \frac{\left(\hat{H}_k^0\right)^* Y_k^0 + \left(\hat{H}_k^1\right)^* Y_k^1}{\left|\hat{H}_k^0\right|^2 + \left|\hat{H}_k^1\right|^2}$
- 但是这个方法对于噪音(n)比较大的的时候效果不好，因为实际上放大了噪音
- 所以MMSE算法被引入对抗噪音

$$\tilde{Y}_k = \frac{\left(\hat{H}_k^0\right)^* Y_k^0 + \left(\hat{H}_k^1\right)^* Y_k^1}{\left|\hat{H}_k^0\right|^2 + \left|\hat{H}_k^1\right|^2 + \frac{\sigma_k^2}{\beta}}$$

第二章： PUSCH以及相关信道和处理

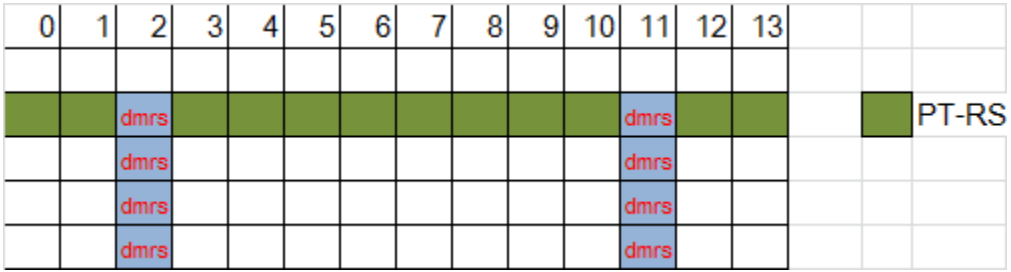
第三节：Phase-tracking reference signals

（相位跟踪参考信号）



相位参考信号(PT-RS)的作用

- PT-RS并不是必须的，一般用于子载波间隔比较大的5G，子载波间隔比较小的可用可不用，其作用在于纠正信号的相位差，并且弥补时间上的差距
  - ✓ 子载波间隔比较大的，则TTI的长度比较小，时间上的偏差更容易造成信号的损失
  - ✓ 所以如下图，PT-RS的信号是沿着Symbol在时间上展开的



  - ✓ 对于时偏的估计，有很多方法，在4G LTE时代，但不同频率上（比如LTE-A中是RE位置相差6的两个DMRS位置上的H）的DMRS H做频域相关然后估计时间偏移，原理是如果时域上有偏移，则在频域上表现为相位上有旋转，所以时偏估计就是利用了频域相关来求取频域上的相位旋转了多少，进而来估计时间上偏移了多少；从5G来看，原理上来说，每一个symbol的时间差是固定的，利用时域上每一个信号的相位差也能算出时间偏差。

相位参考信号(PT-RS)的生成

- PT-RS信号是基于Golden序列的，在不支持transform precoding基本公式如下：

$$r_{n_s}(m)=\frac{1}{\sqrt{2}}\left(1-2\cdot c(2m)\right)+j\frac{1}{\sqrt{2}}\left(1-2\cdot c(2m+1)\right),\quad m=0,1,\ldots,N_{sc}^{RB}N_{\mathrm{RB}}^{\max,\mathrm{UL}}-1$$

  - ✓ 其中C(m)就是Golden序列，其序列的根定义为

$$c_{\mathrm{init}}=\left(2^{17}\left(14n_s+l+1\right)\left(2N_{\mathrm{ID}}^{n_{\mathrm{SCID}}}+1\right)+2N_{\mathrm{ID}}^{n_{\mathrm{SCID}}}+n_{\mathrm{SCID}}\right)\bmod 2^{31}$$

  - ✓ 我们可以发现其序列的生成和DRMS是一样的

- PT-RS信号在支持transform precoding基本公式如下：

$$\bar{r}(m')=\frac{1}{\sqrt{2}}\left[\left(1-2c(m')\right)+j\left(1-2c(m')\right)\right]$$

$$r\left(N_{\mathrm{samp}}^{\mathrm{group}}s'+k'\right)=e^{jm\pi/2}w(k')\bar{r}(m')$$

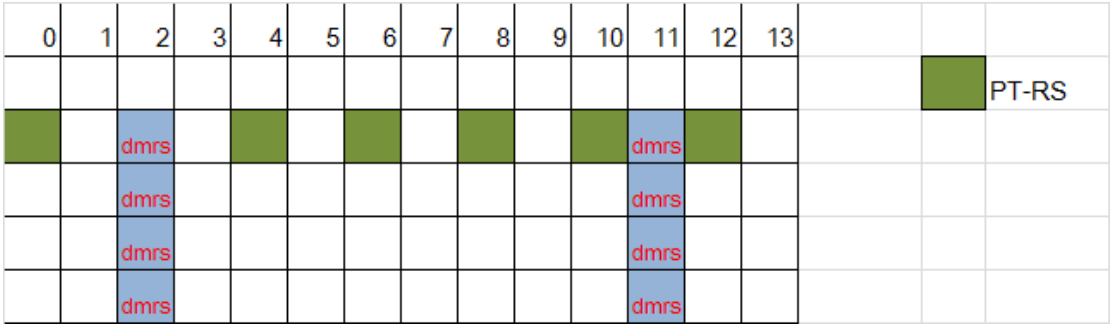
$$s'=0,1,\ldots,N_{\mathrm{group}}^{\mathrm{PTRS}}-1$$

$$k'=0,1,N_{\mathrm{samp}}^{\mathrm{group}}-1$$

  - ✓ 就是在原来的Golden序列上，进行了一下相位旋转 $e^{jm\pi/2}$，并乘一个正交序列 $w(k')$

相位参考信号(PT-RS)的资源位置

- PT-RS信号是要避开DMRS的位置，其发射功率通过 β_{PTRS} 进行加权
- PT-RS在时间上的分布通过L_{PT-RS} 来实现，这个值的取值是(1,2,4)，实际上表示隔几个Symbol放一个PTRS的symbol.如果L_{PT-RS} 的值取1，就是连续放(跳过DMRS的位置)，如果PT-RS遇到，DMRS的信号，则从DMRS开始，重新往后跳，下图给了一个L_{PT-RS} 的值取2的例子



- PT-RS信号在频域上分布在3GPP定义如下：K_{PTRS} 的取值为2或者4，定义为每2个或者4个PRB放一个PT-RS,具体在哪一个RE由下列公式定义，基本只要知道是RNTI Mod而来。

$$k_{\mathrm{ref}}^{\mathrm{RB}} = \begin{cases} n_{\mathrm{RNTI}} \bmod K_{\mathrm{PTRS}} & \text{if } N_{\mathrm{RB}} \bmod K_{\mathrm{PTRS}} = 0 \\ n_{\mathrm{RNTI}} \bmod \left(N_{\mathrm{RB}} \bmod K_{\mathrm{PTRS}}\right) & \text{otherwise} \end{cases}$$

第三章： PUCCH以及相关信道和处理

第一节：PUCCH信道和处理过程

PUCCH信道的作用

- 对于PUCCH的UE间资源分配（Cplane和U plane共同作用），解码和处理(L1 phy的算法和性能)是比较各个厂商算法和性能的关键所在，PUCCH的策略也是影响任何OFDM系统的关键所在。

- 在4G中，PUCCH的所有内容，如果同时有PUSCH的数据要发送，PUCCH是嵌入到PUSCH的资源中的，就是打掉PUSCH的某些symbol。5G中，也会如此，但是这部分规范还在讨论中

- PUCCH主要用于承载/传送下列信息

  - ✓ SR ： UE发送SR请求基站的上行调度，其实是UE告诉基站，UE有数据要发送

    注意，在4G LTE时代，因为SR只有一个比特，所以SR的错误概率非常高，各个厂商基本都在10%左右。

  - ✓ CQI： UE告诉基站所测量的整个带宽上和某些子带上的信道质量，主要用于下行的调度
  - ✓ RI： UE告诉基站，能够支持多少个不相关的信道，主要用于下行的发送模式(TX-dix or MIMO)的选择
  - ✓ ACK/NACK：顾名思义，这个不用解释，用于指示DL grant的接受结果
  - ✓ BSI： UE告诉基站，所测量的下行最强的beam的id

PUCCH信道的格式

- PUCCH的格式在3GPP 38.211 的表6.3.2.1-1中定义。
- 在4G中，那种信息用什么Format是严格定义，如Format 0只用于SR,但是在5G的3GPP中，似乎没有这么严格的规定，目前为止是规定比特数，反正Format的号越大，能支持的比特数越多

Table 6.3.2.1-1: PUCCH formats.

PUCCH format	Length in OFDM symbols	Number of bits
0	1 – 2	=2
1	4 – 14	=2
2	1 – 2	>2
3	4 – 14	>2
4	4 – 14	>2

PUCCH信道的序列生成-1

- PUCCH的序列生成暂时不讨论Group hopping和sequence hopping，因为在可以预见的两年内，没有人（设备厂商和终端厂商）实现这个东西。

- 除了group hopping意外，PUCCH format0，1,3,4的序列生成又是一个$\delta = 0$ Z-C序列，（Format2用Golden序列）Z-C序列其根的定义如下：

$$u = \left(f_{gh} + f_{ss}\right) \bmod 30$$

$$f_{gh} = 0$$

$$f_{ss} = n_{ID} \bmod 30$$

$$v = 0$$

在不支持Group hopping和sequence hopping的情况下

nID规范中描述将会在213中定义(目前还没有)，

应该就是小区ID

PUCCH信道的序列生成-2

- PUCCH的Z-C序列，α 定义如下

$$\alpha_l = \frac{2\pi}{N_{sc}^{RB}}\left(\left(m_0 + m_{cs} + n_{cs}(n_s, l+l')\right) \bmod N_{sc}^{RB}\right)$$

其中

- ✓ n_s 就是slot的号
- ✓ l 和 l' 分别是PUCCH开始传送的Symbol和当前传送的Symbol号
- ✓ $m_0 + m_{cs}$ 较为复杂，取决于应用，下一页PPT会谈到
- ✓ 而 n_{cs} 是一个如下的方程：其中的c就是一个以小区ID为根的Golden序列

$$n_{cs}(n_s, l) = \sum_{m=0}^{7} 2^m c\left(14 \cdot 8n_s + 8l + m\right)$$

PUCCH信道的序列生成-3 Cycle shift的决定

- 在此之前我们解释了PUCCH序列的生成，公式中的 $m_0 + m_{cs}$ 没有谈到
- 其在3GPP中的定义如下：
  - ✓ $m_{CS} = m + m_0$ 中的 m_0 是在UE建立的时候，从RRC高层配下来，PUCCH-F0-F1-initial-cyclic-shift（好玩了，这里面有戏，C-plane的活），因为Cycle shift是不能重叠的
  - ✓ 而 m 的取值来自于ACK还是NACK，比如一个比特的时候

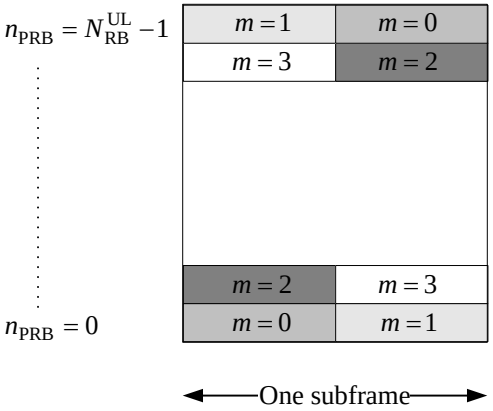
HARQ-ACK Value	0	1
Sequence cyclic shift	m = 0	m = 6

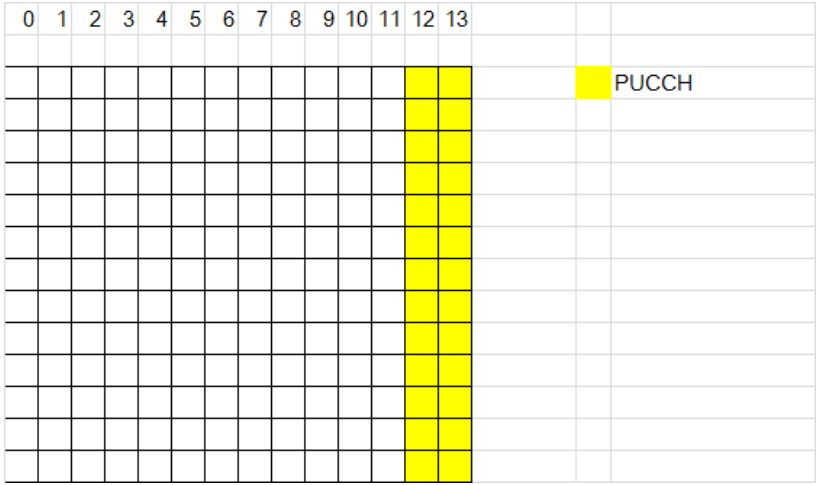
  - ✓ 而两个比特的时候，则如下表

HARQ-ACK Value	{0, 0}	{0, 1}	{1, 1}	{1, 0}
Sequence cyclic shift	m = 0	m = 3	m = 6	m = 9

PUCCH信道的资源位置综述

- 5G PUCCH的资源位置，和LTE时代有很大的不同。
  - 在4G LTE中，PUCCH的资源位置如左下图（最简单的Format 0的例子）， 而在5G中，PUCCH的资源位置如右下图，其中到底多少个Symbol取决于格式和配置
  - 理论上5G可用的符号更多，而且可以灵活的配置





PUCCH信道的资源位置-1

- PUCCH Format 0的资源位置。
  - ✓ Format0最多用2列Symbol，可用于传送SR和ACK NACK，其直接作用于PUCCH的Z-C序列上。
- PUCCH Format 1的资源位置。
  - ✓ Format1最可以用4-14列Symbol，启用BPSK或QPSK调制，可用于传送ACK、NACK
  - ✓ 其传送的信息除了Z-C序列以外，还要加上一个扩频正交序列，解释如下
    - 发送的序列第一步处理为： 发送系列$b(0),...,b(M_{\text{bit}}-1)$ 最多为2bit，经过调制后，变成一个符号 $d(0)$ ，
    - 这个符号 乘上12位的Z-C序列（意味着其单位是一个RB），变成12个符号 $y(n)=d(0)\cdot r_{u,v}^{(\alpha,\delta)}(n)$ $n=0,1,...,N_{\text{sc}}^{\text{RB}}-1$
    - 然后乘上一个正交序列，其过程在下一页中表述，只要记住到这一步还是12个符号

PUCCH信道的资源位置-2

- PUCCH Format 1的资源位置 （续）。
  - ✓ 乘上12个Z-C序列以后，还要乘上一排正交的符号，才能发送，这个正交符号序列的长度 $N_{SF,m'}^{PUCCH,1}$ 基本上取决有Format 1所使用的符号数，这个正交序列在3GPP 38.211表6.3.2.4.1-1中定义。其3GPP定义具体过程如下

$$z\left(m'N_{sc}^{RB}N_{SF,0}^{PUCCH,1}+mN_{sc}^{RB}+n\right)=w_i(m)\cdot y(n)$$

$$n=0,1,\ldots,N_{sc}^{RB}-1$$

$$m=0,1,\ldots,N_{SF,m'}^{PUCCH,1}-1$$

$$m'=\begin{cases}0 & \text{no intra-slot frequency hopping}\\ 0,1 & \text{intra-slot frequency hopping enabled}\end{cases}$$

如果我们不支持跳频，那么 m' 为0，怎整个公式简化为

$$z\left(mN_{sc}^{RB}+n\right)=w_i(m)\cdot y(n)$$

我们可以看出，似乎是扩大了m个symbol的尺度，就是原来的一个符号，m*12个符号

PUCCH信道的资源位置-3

- PUCCH Format 2的资源位置映射

  - ✓ Format2第一步要做的先是加一个扰码序列，在3GPP中定义为

$$\tilde{b}(i) = \left(b(i) + c(i)\right) \bmod 2$$

其中的C序列就是根为 $c_{\text{init}} = n_{\text{RNTI}} \cdot 2^{15} + n_{\text{ID}}$ 的一个Golden序列，其中n_id为小区ID

  - ✓ 第二步则是将要发送的序列进行QPSK调制，则调制以后的序列为

$$d(0),\ldots,d(M_{\text{symb}} - 1)$$

  - ✓ 第三步则是将调制以后的每一个符号，直接放到时频资源上去，这种做法和Format比较类似，当然不要忘记功率控制因子

PUCCH信道的资源位置-4

- PUCCH Format 3/4的资源位置映射
  - ✓ Format3/4第一步要做的先是加一个扰码序列，在3GPP中定义为

$$\tilde{b}(i) = \left(b(i) + c(i)\right) \bmod 2$$

其中的C序列就是根为 $c_{\text{init}} = n_{\text{RNTI}} \cdot 2^{15} + n_{\text{ID}}$ 的一个Golden序列，其中n_id为小区ID

  - ✓ Format 3或者Format 4都要映射到多个RB. 其区别在于
    - Format 3不进行正交因子处理
    - Format 4进行正交因子的处理（和Format 1比较类似）
  - ✓ 而Format 3或者4在上行都按照PUSCH的处理方式进行发射预编码。

第三章： PUCCH以及相关信道和处理

第二节：PUCCH信道物理层过程

这个Topic对做UP的人非常重要！！！！！

而规划是做CP的人定的，也重要

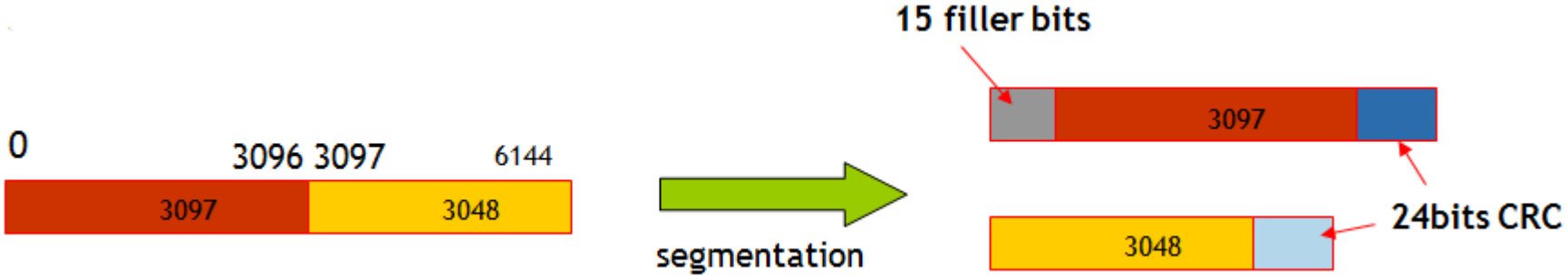
PUCCH 的资源分配

- 对于每个UE,在UE建立的时候，通过RRC的配置消息，给UE下发从 PUCCH-resource-config-PF0 到PUCCH-resource-config-PF4，每一种格式可以分配一到几个。
- 在下发的资源定义中，包含
  - ✓ Starting Symbol， Number of Symbol
  - ✓ Starting PRB number of PRB
  - ✓ PUCCH-F0-F1-initial-cyclic-shift;
  - ✓ 如果是F3/4 还有正交序列的信息
- 这个分配是由C-plane的算法决定的，但是U plane也是要知道的
- 如果同一种资源分配了多个，具体用哪一个由DL grant决定

听上去挺简单的，呵呵，但是真相只有一个

PUCCH ACK/NACK的上报-综述

- ACK/NACK无论在4G还是5G，每一次UCI (ACK/NACK)所携带的都是一次或者多次HARQ的结果，而一次HARQ可能最多是2个(LTE)或者4个(5G)的code word TB块的结果，所以UCI的比特是是远远多于1的，具体原因如下
  - ✓ 在4G LTE中，随着CA的引入，PUCCH的ACK/NACK资源变得紧张起来，其原因是ACK/NACK的指示都在Pcell上完成。也就是说下行无论哪个载频的调度，其上行反馈的UCI(ACK/NACK)都是在Pcell上报的，另外各个载频的CQI也是在Pcell(主小区)上完成的，为此LTE引入了Pucch format3 (和5G的format 3不一样)and 1bcs
  - ✓ 在LTE TDD中，因为娘胎里带来的不对称性，使得上行每次ACK/NACK都要携带一个或者多个下行调度的
- 在4G中，至少UCI都是基于TB(MAC PDU)块，而5G里面，天雷滚滚的引入了基于CB(code Block)的ACK/NACK，从而使得重传可以基于CB块。所谓CB块实际上完全是L1 PHY的概念，现在这个概念渗透到了L2,这个让人非常不愉快，关于CB快的概念如下图(切割的大小可能不in-line 3GPP最新,不管了，以后更新)。以后会详细讲



PUCCH ACK/NACK的上报-UE如何组织要上报的比特序列(CBG上报的概念)

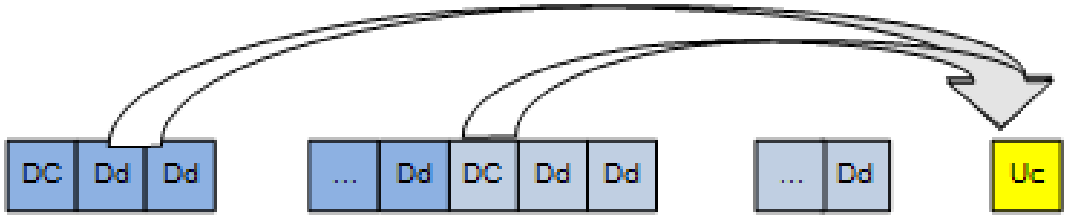
- 我们首先来讨论基于CBG的上报方式，也就是基于Code Block group的上报方式，每个group上报一个ACK或者NACK，是不是基于CBG来上报ACK/NACK，是通过基站在DL Grant中指示的。

- UE通过RRC的配置了参数 $N_{\text{HARQ-ACK}}^{\text{CBG/TB,max}}$，这个参数指示UE，最大的Code Block group的数量，也就是说对于任何TB块，回馈的ACK/NACK不会超过这个数，Code Block和Code block group的映射关系如下

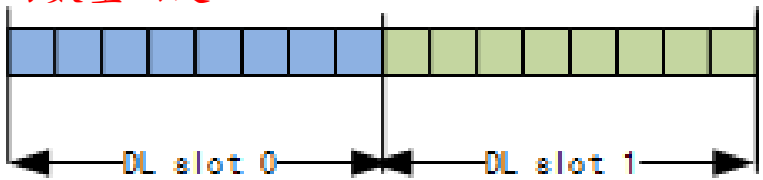
  - ✓ 假设一个TB块被分为10个Code block.而参数 $N_{\text{HARQ-ACK}}^{\text{CBG/TB,max}}$ 配成3
    - 那么第一个group包含CB块0,1,2，3 第二个包含4,5，6,7.第三个包含8,9,
  - ✓ 假设一个TB块被分为3个Code block.而参数 $N_{\text{HARQ-ACK}}^{\text{CBG/TB,max}}$ 配成5
    - 那么第一个group包含CB块0 第二个包含CB块1.第三个包含CB块2,
    - 如果配置是HARQ-ACK-codebook的配置是semi-static的方式，那么前三个比特是相应CB的ACK/NACK，后两个填成NACK（因为不存在）

PUCCH ACK/NACK的上报-UE如何组织要上报的比特序列-最简单的semi static方式-1

- 在semi static方式下，ue report的比特数是固定大小的，也就是说在所有的被调度的slot,所有的Scell，所有的code word一字排开
  - ✓ 举一个最简单的例子，只用一个code word，8个Scell,如下的DL/UL配置



  - ✓ 比特数为如下图，每一个DL slot对应每一个sub carrier，最多能对应多少个slot，是由PDCCH中能支持的K1的数量而定

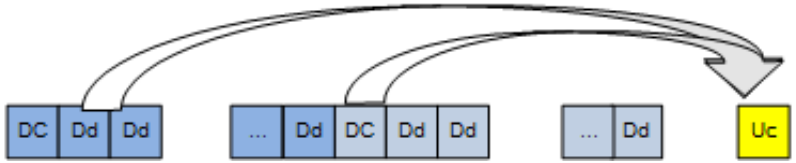


PUCCH ACK/NACK的上报-UE如何组织要上报的比特序列-最噩梦的dynamic方式-1

- 关于Dynamic的方式，目前Nokia在18中并不支持，19A中不确定，其来源于TD LTE的固有的方式，换种方式说，TD LTE没有semi Static方式，只有dynamic的方式。
- 从前一页的semi static的方式来看，是不是很浪费，因为如果在2个slot内，在8个Scell上如果只有一次调度，理论上只需要一个或者两个比特（取决于Code word的数量），但是我们要传很多个比特，而Dynamic的方式就是只传递必须的比特，因为理论上根据下行调度的数量，UE和基站都知道传几个比特。但是问题出在DL grant可能会丢，为了解决这个问题从TDD LTE到5G，引入了DAI的概念，就是每次grant告诉UE这是在这一次ACK反馈中的第几个Grant.
- 因为有了DAI，如果UE收到DAI 0,1,3.就可以知道DAI为2的grant已经丢失，所有对2可以回NACK.但是问题如果基站下发了0,1,2,3，而UE只收到0,1,2，最后一次没有收到依然是个问题
- 特别是引入SPS以后，这个问题会变得更加复杂，因为SPS是不需要grant的。

PUCCH ACK/NACK的上报-UE如何组织要上报的比特序列-最噩梦的dynamic方式-2 DAI的概念

- 为了支持Dynamic的方式，3GPP在LTE TDD中一开始就引入了DAI，而在release 13中，为了CA，更加细化了DAI,引入了total DAI和Counter DAI这个概念，我们还是要看下面的图，来讲这两个概念



- 我们可以看到，一个UL的PUCCH的机会，对应两个DL调度机会，而我们有3个Scell，如果每一个Scell在每一个Slot都有调度
  - ✓ 那么在第一个Slot,第一个Sub carrier上 DL grant 中Counter DAI是1，total DAI是3…以此类推，第三个sub carrier上，C-DAI是3，T-DAI是3
  - ✓ 那么在第2个Slot,第一个Sub carrier上 DL grant 中Counter DAI是4，total DAI是3+3为6，但是表示为(因为DAI是从1到4)是2 …以此类推，第三个sub carrier上，C-DAI是2，T-DAI是2

PUCCH ACK/NACK的上报-UE如何组织要上报的比特序列-最噩梦的dynamic方式-3 DAI的概念

- 3GPP中DAI的定义如下

Table 9.1.3-1: Value of counter DAI and total DAI in DCI format 1_0 or DCI format 1_1

DAI MSB, LSB	V_{C-DAI}^{DL} or V_{T-DAI}^{DL}	Number of {serving cell, PDCCH monitoring occasion}-pair(s) in which PDSCH transmission(s) associated with PDCCH or PDCCH indicating downlink SPS release is present, denoted as Y and $Y \geq 1$
0,0	1	$\mathrm{mod}(Y-1,4)+1=1$
0,1	2	$\mathrm{mod}(Y-1,4)+1=2$
1,0	3	$\mathrm{mod}(Y-1,4)+1=3$
1,1	4	$\mathrm{mod}(Y-1,4)+1=4$

- 根据收到的DAI，UE不断增大要发送的ACK/NACK的比特

PUCCH ACK/NACK的上报-UE如何组织要上报的比特序列-最噩梦的dynamic方式- 4 关于DAI的再思考

- 3GPP中既然引入了DAI，那么给系统设计带来很大的影响，在LTE的时代，这个TDD所特有的东西，带来无尽的烦恼，就是在CA的情况下
- 因为C-DAI和T-DAI的概念，每一个DL grant中都带有C-DAI，其实就实际上暗示着，每一个Sub Carrier的调度单元，都知道其他调度单元的情况（因为DL grant是在各自的sub carrier上下发），或者说有一个锚点，其知道所有Sub carrier的调度情况，然后由这个锚点统管所有sub carrier的PDCCH payload，其他调度单元都向这个锚点报告。
- 在3GPP中定义了最多有15个Sub carrier，那么在系统设计的时候，这为一个UE服务的这15个调度器，不可能在一个CPU process上，甚至不一定在一个RACK内，这样明显带来了大量的消息开销
- 如果再加上PUCCH可能在PUSCH上上报，哇，天雷滚滚

PUCCH ACK/NACK的上报-时间

- ACK/NACK在哪一个slot的Uc symbol上进行上报，是通过DI grant中的PDSCH-to-HARQ-timing-indicator这个域进行指定的，其定义了在收到DL grant以后，在k1个slot以后进行PUCCH ACK/NACK的上报，如果在DL中没有携带PDSCH-to-HARQ-timing-indicator这个域，则UE默认在4个slot以后上报， PDSCH-to-HARQ-timing-indicator是一个0-7的数，这0-7个数所表示具体K1值，则是通过RRC配置Slot-timing-value-K1来完成的
- 另外ACK、NACK的比特，如果RRC中指定了合并方式，那么ACK/NACK在code word上进行一次比特 AND操作

PUCCH ACK/NACK的上报-资源位置

- 那么ACK/NACK上报到底使用哪个PUCCH的resource呢？
  - ✓ 到底哪个resource是由PDCCH的DL grant中的PUCCH-resource-index来指定的，看上去很简单
  - ✓ 但是一次PUCCH的ACK/NACK可以包含多次DL grant所对应的内容，答案是最后一次。UE总是选择最后一次收到的PUCCH-resource-index进行上报
  - ✓ 记得我们前面谈到的DAI最后错误的问题吗？如果最后一次没有收到，那么UE会报错位置吗？当然的，这个基本上无解。

    也就是最后一个没有收到，那么UE就会在错误的位置上

    发送PUCCH的信息，就是我怎么知道你是最后一个 ->

  - ✓ 如果UE收到的PDSCH没有所对应PDCCH DL grant怎么办，比如SPS这种模式，则UE使用默认的*n1PUCCH-AN*，其值通过RRC消息获得

PUCCH SR的上报

- SR是由UE触发，告知基站UE有数据要发送。目前3GPP对SR只定义了一些RRC的配置数据
  - *SR-resource：指示SR使用Format 0或者Format1的resource*
  - *SR的period和offset，单位是Symbol，如下表*

UE-specific SR periodicity and offset configuration (in symbols)			
μ	CP		
0	Normal	$SR_{PERIODICITY}$	2, 7, n*14, where n={1, 2, 5, 10, 20, 32, 40, 64, 80}
		SR_{OFFSET}	
1	Normal	$SR_{PERIODICITY}$	2, 7, n*14, where n={1, 2, 5, 10, 20, 32, 40, 64, 80, 160}
		SR_{OFFSET}	
2	Normal	$SR_{PERIODICITY}$	2, 7, n*14, where n={1, 2, 5, 10, 20, 32, 40, 64, 80, 160, 320}
		SR_{OFFSET}	
2	Extended	$SR_{PERIODICITY}$	2, 6, n*12, where n={1, 2, 5, 10, 20, 32, 40, 64, 80, 160, 320}
		SR_{OFFSET}	
3	Normal	$SR_{PERIODICITY}$	2, 7, n*14, where n={1, 2, 5, 10, 20, 32, 40, 64, 80, 160, 320, 640}
		SR_{OFFSET}	

多种PUCCH的同时上报/复用-1

- PUCCH的四种类型，SR,ACK/NACK,CQI,BSI，可能在同一个时间，同一个symbol都要上报。那么各个类型，按照3GPP的定义，他们是组合上报
  - ✓ 比如如果ACK/NACK和SR都要上报，那么SR就不在原来的resource上发了，而是在ACK/NACK的resource上报，那么怎么报呢？
  - ✓ 如果SR是1，和ACK/NACK(format 1)在一起的时候，我们看到什么东西变了？

HARQ-ACK Value	0	1
Sequence cyclic shift	$m=3$	$m=9$

  - ✓ 那么网络是怎么知道手机是同时报SR和ACK/NACK的
    - 因为SR的周期是网络配下来，网络也知道ACK/NACK的时间，所以理论上可以知道，但是这是理想的情况
    - 如果UE没有解出Grant,那么UE不会回ACK/NACK,SR依然在原来的资源上发，这就意味着这里有不确定的情况，那么网络两个资源处都要解，那么四个资源混用的情况呢？？？？哈哈哈

NOKIA

多种PUCCH的同时上报/复用-2

- PUCCH的如果ACK/NACK和SR同时要上报，并且ACK/NACK的比特数比较多，使用了Format 2/3/4这种方式的话，那么就会多一个比特来放置SR的信息
- 那么CQI/ACKNACK/SR同时存在要一起上报会怎样呢？

  - ✓ 首先不是所有的UE都能做到，看价钱喽，如果UE的能力做不到 ->扔掉CQI
  - ✓ 基站是知道UE能力的，但是为了不让UE drop CQI，基站一般做调度避免。

    但是这里有一个悖论，UE通过RRC上报能力之前，基站是不知道的，那怎么办？？？

    但是UE刚刚接入，基站特别想知道他的CQI，基站只好一律规避……
  - ✓ 如果Ue有能力上报，能么按照 $O_{\text{ACK}} + O_{\text{SR}} + O_{\text{CSI}} + O_{\text{CRC}}$ 的次序来放置比特，具体的规则在下一页

多种PUCCH的同时上报/复用-3

- PUCCH在Format 2/3/4复用的时候,要考虑如下的公式，我给他取个名字，叫比特容量

  从左到右依次为RB数，子载波数（去除DMRS），symbol数，调制比特数，码率

  其中，码率是通过L3 配置

$$M_{\mathrm{RB}}^{\mathrm{PUCCH}} \cdot N_{\mathrm{sc,ctrl}}^{\mathrm{RB}} \cdot N_{\mathrm{symb}}^{\mathrm{PUCCH}} \cdot Q_m \cdot r$$

  我们可以发现这四个数字一乘，就是能容纳的PUCCH的比特数字

- 然后理论上是选择最小的能装下所有复用比特的资源，如果相对应的资源放不下怎么办？
  - ✓ 砍掉CQI的比特，按照 $\mathrm{Pri}_{CSI}(y,s,c,t)$ 所选择的优先级，保证一部分CQI比特能上去（这个我们以后再讲，是个大topic）

PUCCH信息的repetition

- 在3GPP从RL14以后，在4G中为了物联网手机，引入了PUCCH的repetition，主要是为了3-12db的PUCCH的gain.
- 在38.213中，$N_{\mathrm{PUCCH}}^{\mathrm{repeat}}$ 指定了repeat的次数，对于不同的Format，这个repeat是可以不一样的。
- PUCCH的repeat在加上PUCCH的跳频，理论上的确极大的增加了PUCCH的抗干扰的能力。但是这个要看算法，因为跳频不一定有好处
- 但是我们从PUCCH的repetition上能玩一些什么呢？
  - 既然PUCCH的整个序列在不同的Slot上重复，那么其所对应的PUCCH的DMRS也在跳。
  - 这样我们对PUCCH的DMRS进行cross TTI的合拼处理，可以更加精确的获得PUCCH的信道相应，但是一旦跳频，那么就完蛋了

PUCCH在PUSCH上发送

- 写PPT的人一度以为这个大杀器不会出现，有一点点侥幸的心理，3GPP关于这部分内容还在制定中，定义了部分内容，所以这页PPT只谈一下大概的内容，等待3GPP的更新

  - ✓ PUCCH内嵌在PUSCH中发送，也就是打掉了PUSCH的部分symbol，这实际上升高了PUSCH的码率，对PUSCH的解码能力是有不好的影响的，但是PUSCH的能量较大，DMRS符号多，所以对PUCCH的解码能力有帮助。在调度PUSCH的时候如果有PUCCH的内嵌，则PUSCH的MCS要做相应的处理

  - ✓ 如果当前TTI，有PUCCH和PUSCH同时出现，PUCCH必须内嵌在PUSCH中上报。如果没有PUSCH则在PUCCH的资源上上报，好了，但是PUSCH的grant，UE没有收到怎么办？？？UE就会在PUCCH上上报，所以基站要做双解码…

  - ✓ 但是UL CA出现了以后，如果同一个slot有多个UL Scell都有调度，那么在Cell index较小的那个PUSCH上上报。但是任何一个grant都会丢失的，所以这个逻辑，写PPT的人要去撞墙了

第三章： PUCCH以及相关信道和处理

第三节：PUCCH信道的DMRS

PUCCH Format 1 DMRS序列的生成

- 在LTE 4G中，PUCCH是没有DMRS的，这是4G的小错误，在5G中修正了
- 3GPP中PUCCH format 1的DMRS的序列生成方式如下，很明显这是一个Z-C序列

$$z\left(m'N_{\text{sc}}^{\text{RB}}N_{\text{SF},m'}^{\text{PUCCH},1}+mN_{\text{sc}}^{\text{RB}}+n\right)=w_i(m)\cdot r_{u,v}^{(\alpha,\delta)}(n)$$

$$n=0,1,\ldots,N_{\text{sc}}^{\text{RB}}-1$$

$$m=0,1,\ldots,N_{\text{SF},m'}^{\text{PUCCH},1}-1$$

$$m'=\begin{cases}0 & \text{no intra-slot frequency hopping}\\ 0,1 & \text{instra-slot frequency hopping enabled}\end{cases}$$

- 如果我们不考虑intra slot hopping，那就变成，就是Z-C乘一个正交码（还没定义），序列的u和a同PUCCH信号一样

$$z\left(mN_{\text{sc}}^{\text{RB}}+n\right)=w_i(m)\cdot r_{u,v}^{(\alpha,\delta)}(n)$$

PUCCH Format 2 DMRS序列的生成

- 3GPP中PUCCH format2的DMRS的序列生成方式如下，很明显这是一个golden序列

$$r(m)=\frac{1}{\sqrt{2}}\left(1-2\cdot c(2m)\right)+j\frac{1}{\sqrt{2}}\left(1-2\cdot c(2m+1)\right)$$

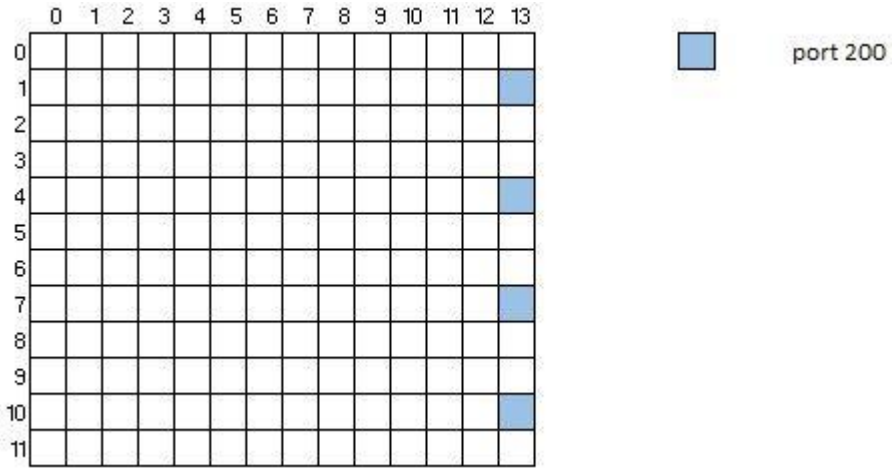
- 其中的根生成方式为

$$c_{\text{init}}=\left(2^{17}\left(14n_{\text{s}}+l+1\right)\left(2N_{\text{ID}}^{0}+1\right)+2N_{\text{ID}}^{0}\right)\bmod 2^{31}$$

- 其中的 $N_{\text{ID}}^{0}\in\{0,1,\ldots,65535\}$，是通过RRC的配置Scrambling-ID

PUCCH Format 1 and 2 DMRS序列的资源映射

- Format1，总是在PUCCH发送的第一个symbol上（如果有两个Symbol给PUCCH的话），子载波就在PUCCH发送的子载波上
- Format2，也总是在PUCCH发送的第一个symbol上（如果有两个Symbol给PUCCH的话），但是其子载波总是位于 $k = 3m + 1$，就是1,4,7……这些子载波上,如下图



PUCCH Format 3/4 DMRS序列的生成

- 3GPP中PUCCH format3/4的DMRS的序列生成方式如下，很明显这是一个Z-C序列，其根和PUCCH信号本身一样

$$r(m) = r_{u,v}^{(\alpha,\delta)}(m)$$

- 其中的cycle shift，是通过Symbol和slot号来定义的

$$\alpha(n_s, l) = 2\pi \cdot n_{cs}(n_s, l) / N_{sc}^{RB}$$

PUCCH Format 3/4 DMRS序列的资源映射

- 3GPP中PUCCH format3/4的DMRS的子载波位置总是在RB0的Sub Carrierr0
- 其在Symbol上的位置通过下表来表示

Table 6.4.1.3.3.2-1: DM-RS positions for PUCCH format 3 and 4.

PUCCH length	DM-RS position l within PUCCH span			
	No additional DM-RS		Additional DM-RS	
	No hopping	Hopping	No hopping	Hopping
4	1	0, 2	1	0, 2
5	0, 3		0, 3	
6	1, 4		1, 4	
7	1, 4		1, 4	
8	1, 5		1, 5	
9	1, 6		1, 6	
10	2, 7		1, 3, 6, 8	
11	2, 7		1, 3, 6, 9	
12	2, 8		1, 4, 7, 10	
13	2, 9		1, 4, 7, 11	
14	3, 10		1, 5, 8, 12	

第四章： Sounding RS以及相关信道和处理

SRS信号的恩怨情仇

- “你为什么要坚持使用SRS,你为什么把一切算法的依据都基于SRS，人家爱//公司就不怎么用SRS”某贝尔实验室院士说，……“好吧，那你为什么在贝尔实验室，你去那里爱//公司好了”，法国某资深算法专家说（现在是5G的某个leading SE）…… “我无语飘过”..写PPT的人在边上心里想…… “坦克，你知不知道SRS是唯一没有保护的信道”，哦，点写PPT人的名字了
- 所以SRS是一个有故事的信道。其在5G的规范制定的过程中的经历，可以描述为“方生方死，方死方生”
- SRS总体上的原理是，分配UE一定的时频资源（往往是全频带的，时间上是周期性从5ms到160ms…），UE上报一个基站已知的序列，基站通过对这个已知序列的分析，可以获得如下的信息
  - ✓ UE在各个UL PRB上的SINR，或者相应的频率有效性。提供给上行调度器（类似于下行的CQI）
  - ✓ 测量UE的TA
  - ✓ 获得UE的WB SINR，作为是不是UE radio link failure的测量
  - ✓ 获得UE的速度

如果没有SRS会怎么样

- 如果没有SRS，我们就没有UE在全频带上的SINR.或者无法测量UE在全屏带上的TA，那么有替代的东西吗？有…PUSCH的DMRS，但是

  - ✓ PUSCH的DMRS并不是全频段，也不是周期出现的，如果PUSCH没有调度到的PRB，基站是没有信息的
  - ✓ 因为PUSCH的DMRS不是全频段，所以基站很难滤波
  - ✓ PUSCH DMRS的发送功率不是恒定的，是随着PUSCH的功率走的（加权）
- 但是SRS也有问题，如果用户过多的话，SRS的周期可以达到160ms甚至更大，这个更新周期…哎，两难

SRS的符号生成和资源映射

- 很明显，SRS是基于 $271 \cdot N_{sc}^{RB} / K_{TC}$ 的Z-C序列，用几个Symbol，起始Symbol是哪一个，是由RRC消息配置给UE的，在User 建立的时候，其中 K_{TC} 由RRC消息配置SRS-TransmissionComb

  其中cycle shift的定义为如右

  n_{SRS}^{cs} 是由RRC参数SRS-CyclicShiftConfig配置

  $n_{SRS}^{cs,max}$ 为12或者8，看 K_{TC} 是4还是2

  $N_{ap}^{SRS} \in \{1,2,4\}$ 是port的数量，由NrofSRS-Ports决定

$$\alpha_i = 2\pi \frac{n_{SRS}^{cs,i}}{n_{SRS}^{cs,max}}$$

$$n_{SRS}^{cs,i} = \left(n_{SRS}^{cs} + \frac{n_{SRS}^{cs,max} p_i}{N_{ap}} \right) \bmod n_{SRS}^{cs,max}$$

- SRS的根序列定义为 $n_{ID}^{SRS} \bmod 30$，这个Nid是由RRC消息中的SRS-SequenceId来配置
- SRS的资源位置，我们会在下一个版本中详细谈（给3GPP一点时间），但是目前大家只要记得，K_{TC} 这个参数定义了SRS的符号，到底隔几个子载波放一个SRS信号