

# 5G PUCCH

讲师：捻叶成剑

# PUCCH概述

PUCCH, Physical Uplink Control Channel, 物理上行链路控制信道

PUCCH的作用：承载UCI（Uplink Control information 上行链路控制信息）。

由于PDCCH承载DCI（下行控制信息），因此PUCCH承载UCI。DCI和UCI之间的最大区别在于，根据情况，  
**UCI可以由PUCCH或PUSCH承载，而DCI只能由PDCCH承载**（无论如何都不由PDSCH承载）。

目前不支持PUCCH和PUSCH同时传输，因此，在两个物理信道传输时间冲突的时候，选择一种信道传输UCI，  
具体原则是：

当PUSCH上面不存在用户数据时，由PUCCH传输，反之就由PUSCH传输UCI了

# UCI（上行链路控制信息）

UCI的种类：

- **SR**: Scheduling Request调度请求。在UE有上行数据需要发送，但是无上行资源（PUSCH）时，通过发送SR请求，向网络申请上行调度。---**只能由PUCCH发送**
- **HARQ ACK/NACK**: 对在PDSCH上发送的下行数据**正确与否**进行HARQ确认。
- **CSI**: Channel State Information，包括CQI、PMI、Rank等信息。用于告诉eNodeB下行信道质量等，以帮助eNodeB进行下行调度

这三种UCI可以以各种方式组合传输：

比如HARQ ACK+SR 或者HARQ ACK+CSI等

# PUCCH格式

PUCCH支持5种格式，format0到format4，根据PUCCH信道占用的时域符号长度分为：  
**短格式** (format0,2) 和**长格式** (format1,3,4)

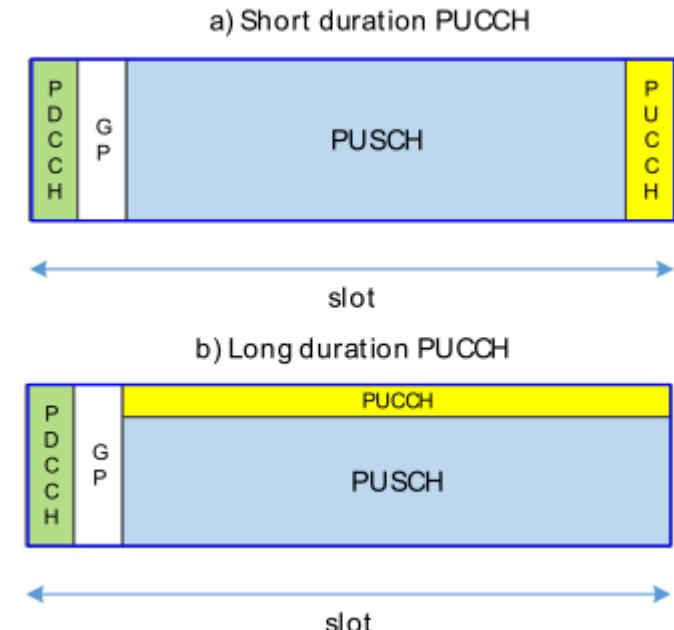
其中，**短格式**的PUCCH可以在一个时隙的最后1或2个符号上传输，对同一时隙的PDSCH的HARQ-ACK、CSI进行反馈，从而达到**低时延**的目的

目前现网PUCCH格式主要以**手动配置**为主，使用**长格式**。

PUCCH format	Short / long duration	Length (symbols)	Number of UCI bits	Number of PRBs
0	Short	1~2	1, 2	1
1	Long	4~14	1, 2	1
2	Short	1~2	> 2	1~16
3	Long	4~14	> 2	1~6, 8~10, 12, 15, 16
4	Long	4~14	> 2	1

PUCCH可以支持**跳频**：

其中，所有格式都支持**时隙内**跳频，格式1,3,4支持**时隙间**跳频，



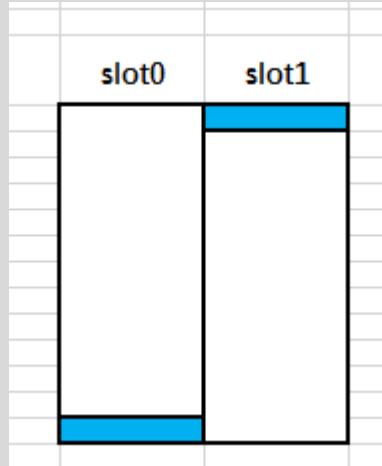
# UE如何知道哪种类型跳频?

在高层信令PUCCH-Config当中：

PUCCH-FormatConfig (格式配置) 里面有是否支持时隙间跳频

```
PUCCH-FormatConfig ::=  
    interslotFrequencyHopping  
    additionalDMRS  
    maxCodeRate  
    nrofSlots  
    pi2PBSK  
    simultaneousHARQ-ACK-CSI  
}  
  
SEQUENCE {  
    ENUMERATED {enabled} OPTIONAL, -- Need R  
    ENUMERATED {true} OPTIONAL, -- Need R  
    PUCCH-MaxCodeRate OPTIONAL, -- Need R  
    ENUMERATED {n2,n4,n8} OPTIONAL, -- Need S  
    ENUMERATED {enabled} OPTIONAL, -- Need R  
    ENUMERATED {true} OPTIONAL, -- Need R
```

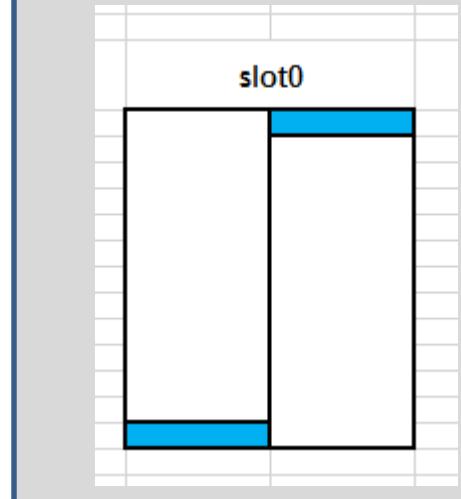
以长格式为例



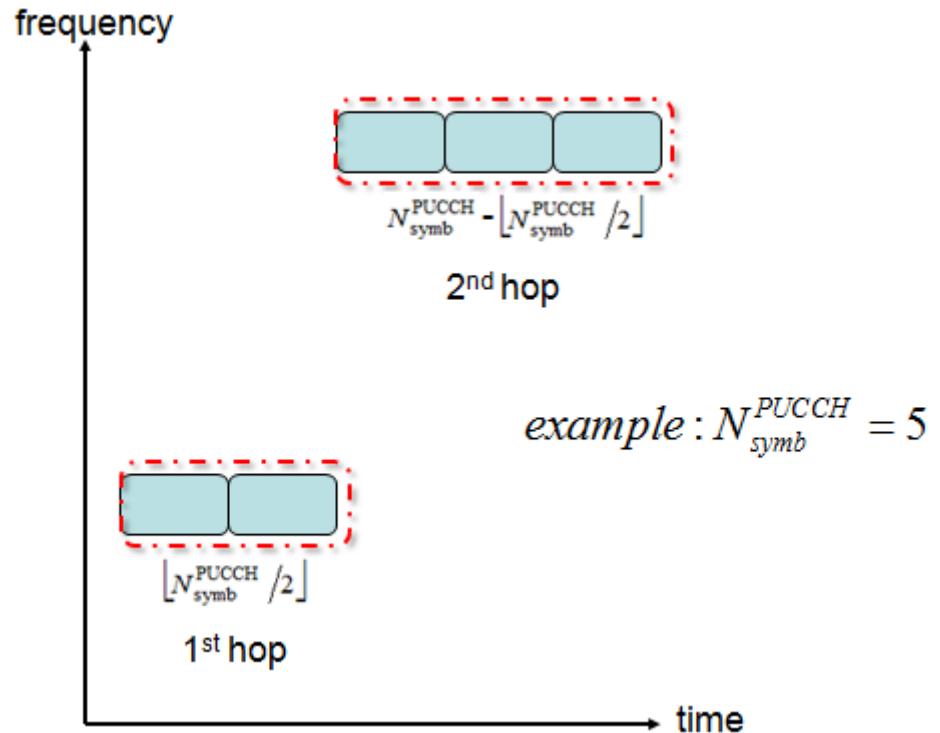
PUCCH-Resource (资源配置) 中，有是否支持时隙内跳频

```
PUCCH-Resource ::=  
    pucch-ResourceId  
    startingPRB  
    intraSlotFrequencyHopping  
    secondHopPRB  
  
    format      CHOICE {  
        format0    PUCCH-format0,  
        format1    PUCCH-format1,  
        format2    PUCCH-format2,  
        format3    PUCCH-format3,  
        format4    PUCCH-format4  
    }  
  
    SEQUENCE {  
        PUCCH-ResourceId,  
        PRB-Id,  
        ENUMERATED { enabled } OPTIONAL, -- Need R  
        PRB-Id      OPTIONAL, -- Need R  
    }
```

slot0

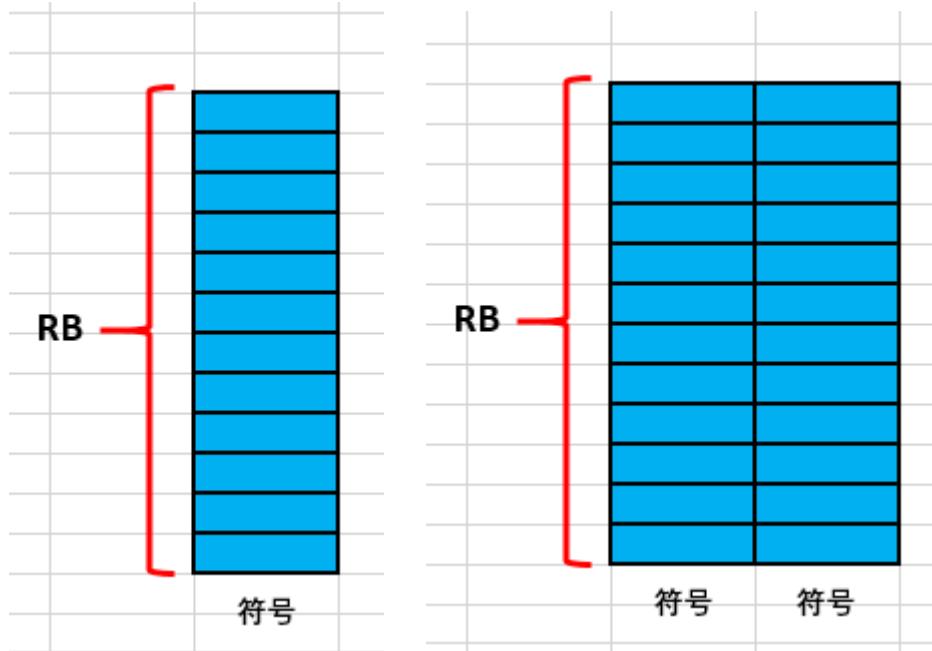


# 跳频的符号分配?



# PUCCH format0

PUCCH的format0格式发送的UCI为1bit或者2bit，用于发送HARQ的ACK/NACK反馈，也可以携带SR调度请求  
PUCCH format0格式频域上占用1个RB，时域上占用1-2个符号。



```
PUCCH-format0 ::=  
    initialCyclicShift  
    nrofSymbols  
    startingSymbolIndex  
}
```

```
SEQUENCE {  
    INTEGER(0..11),  
    INTEGER (1..2),  
    INTEGER(0..13)
```

initialCyclicShift : 初始循环移位 (0-11)

nrofSymbols: 时域符号个数 (1-2)

startingSymbolIndex : 时域开始符号索引 (0..13)

PUCCH的format0格式里面不含有DMRS

支持时隙内跳频

当UCI为1bit的时候，最大复用用户为6个

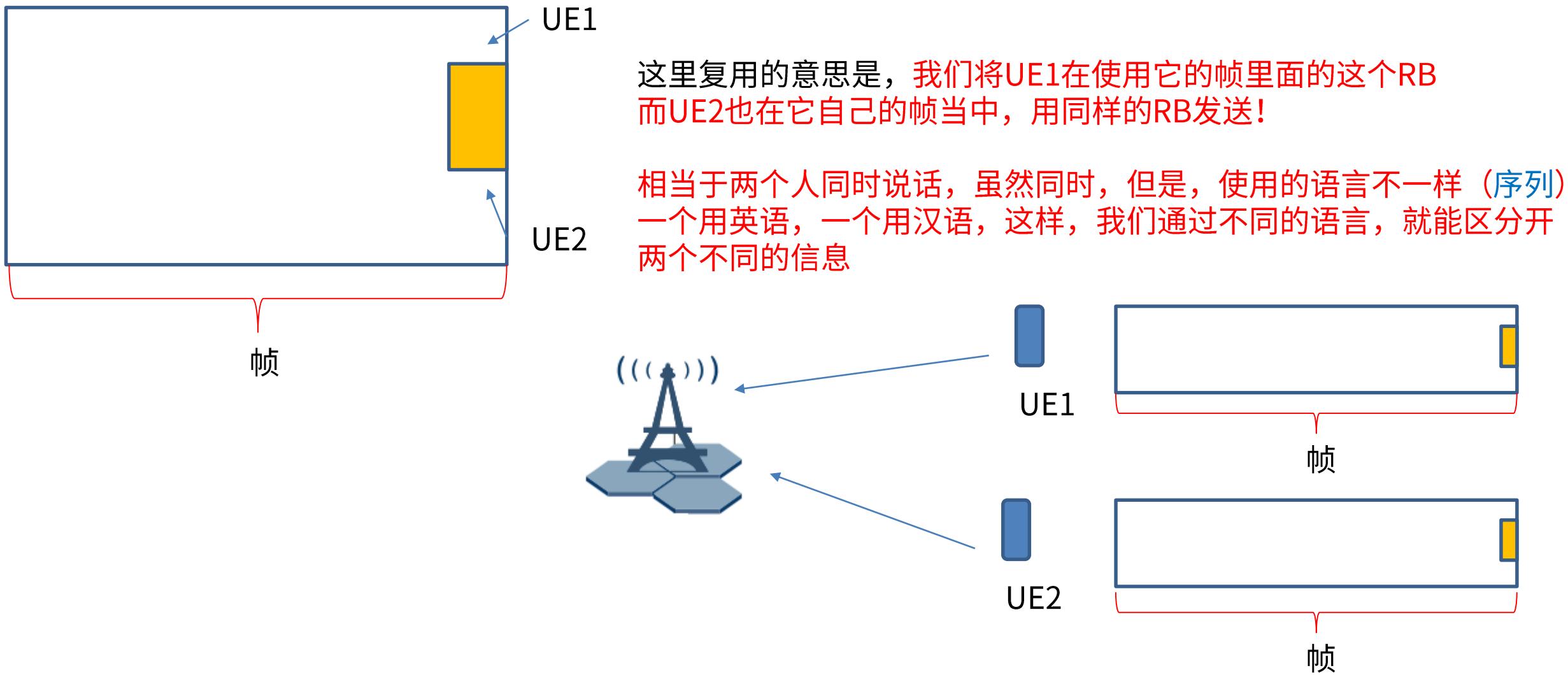
当UCI为2bit的是，最大复用用户3个

如果仅传输SR，可以复用12个UE

PUCCH format0时域资源配置1个或者2个符号时，  
不影响复用的UE个数，当配置为2个符号时，  
可以提升ACK反馈的可靠性

# PUCCH复用是什么意思？

是同一个PUCCH RB，同时发送2个用户的数据？ 错了，不是这个意思！



# PUCCH format0

PUCCH Format0不直接携带bit信息，而是通过“序列”来间接表示信息。

通俗理解的话，类似于这样：

对于HARQ：1代表传输的数据正确，0代表传输的数据错误，  
在传输数据的时候，我不直接传1或者0

用XCDBSBSBFNF 代表1  
用KFJGFJJHDHFJ 代表0

这样，我收到第一个字母“序列”，我就知道我收到了1，而收到第二个“序列”，我就收到了0

实际上，这个序列当然不可能是字母序列，而依然类似PRACH里面的preamble一样，  
是欧拉复数序列

# PUCCH format0 序列

序列公式

$$r_{u,v}^{(\alpha,\delta)}(n) = [e^{j\alpha n}] \bar{r}_{u,v}(n), \quad 0 \leq n < M_{ZC}$$

对于format0来说，这个序列长度为12 ( $M_{ZC}=12$ )

循环移位

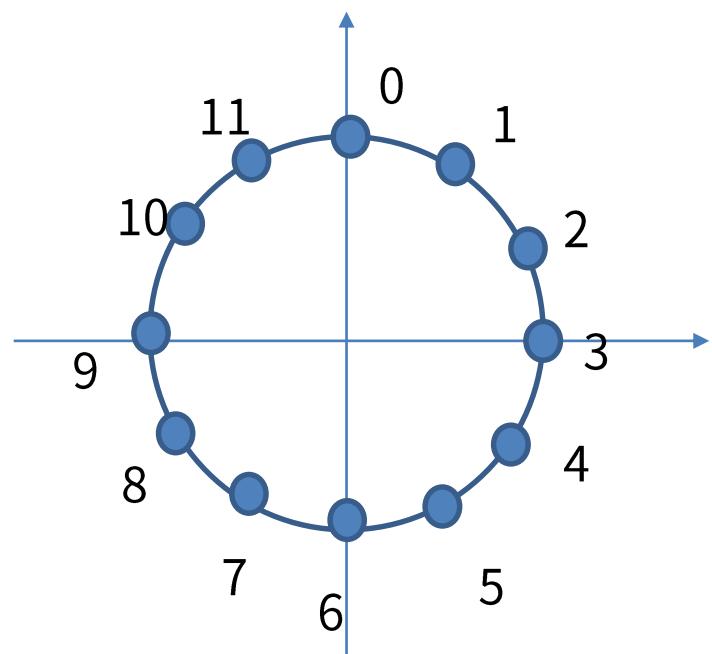
$$\bar{r}_{u,v}(n) = e^{j\varphi(n)\pi/4}, \quad 0 \leq n \leq M_{ZC}-1$$

基序列

回忆一下preamble的生成，我们知道，这种序列，如果从图形上看，就是圆上的一堆点（因为序列长12，因此就是12个点）。而循环移位，相当于序列的起始位置不同。

假设循环位移为0：那么序列是0,1,2,3,4,5,6,7,8,9,10,11，

假设循环位移为2，那么序列就是2,3,4,5,6,7,8,9,10,11,0,1



# 循环移位 $\alpha$

Cyclic Shift

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

时隙编号和符号位置

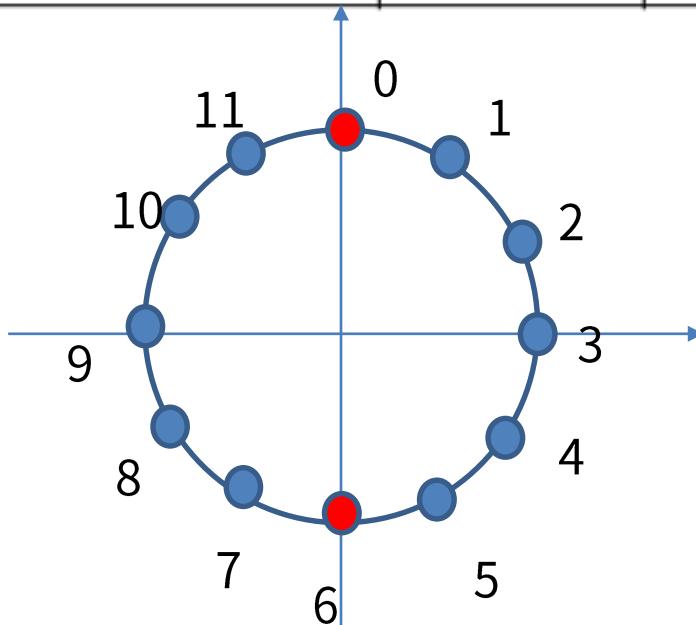
```
PUCCH-format0 ::= {  
    initialCyclicShift  
    nroSymbols  
    startingSymbolIndex  
}
```

```
SEQUENCE {  
    INTEGER(0..11),  
    INTEGER (1..2),  
    INTEGER(0..13)
```

假设 $M_0$ 和 $n_{cs}(n_{sf}^{\mu}, l + l')$ 都是0  
这个时候， $M_{CS}$ 决定了序列的起始位置  
而 $M_0$ 和 $n_{cs}(n_{sf}^{\mu}, l + l')$ 无非就是使 $m_{CS}$ 变大或者变小

< 38.213-Table 9.2.3-3 >

HARQ-ACK Value	0	1
Sequence cyclic shift	$m_{CS} = 0$	$m_{CS} = 6$



事情的核心在于，MCS=0这个序列代表0  
MCS=6这个序列代表1。

Format0占用1个RB，12个符号，因此，一个符号传序列一位

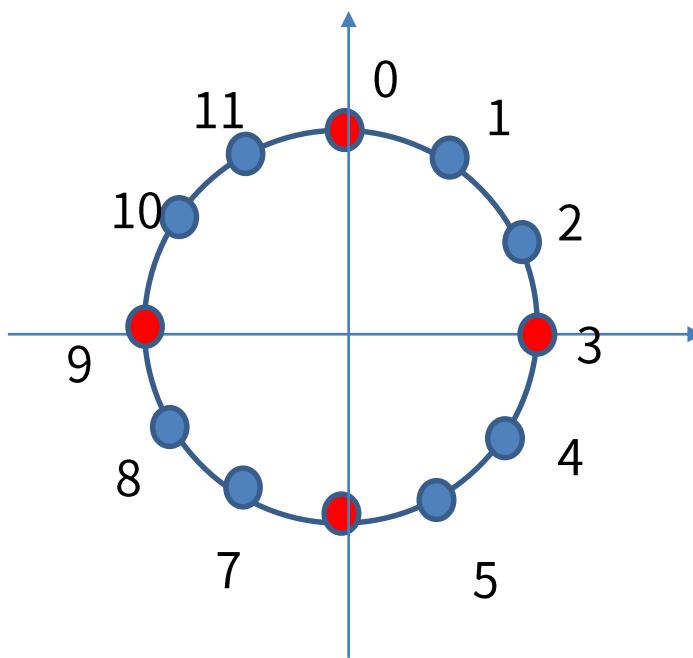
一个UE占用2个序列（因为他既要反馈ACK也要反馈NACK）这样，总共12个序列情况下，可以给6个用户使用

# 双bit HARQ-ACK反馈

双bit的HARQ-ACK反馈，是因为我们系统中传输块可以是2个传输块（MIMO里面讲过），因此他的反馈就是针对这两个传输块来反馈的

< 38.213-Table 9.2.3-4 >

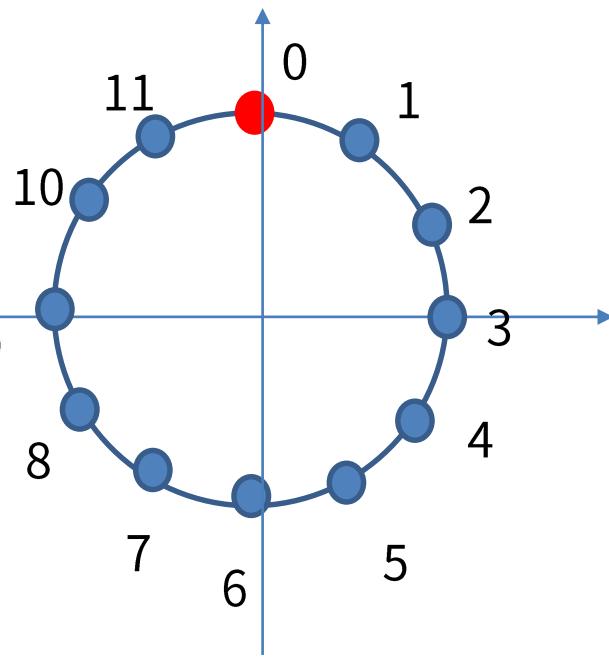
HARQ-ACK Value	{0, 0}	{0, 1}	{1, 1}	{1, 0}
Sequence cyclic shift	$m_{CS} = 0$	$m_{CS} = 3$	$m_{CS} = 6$	$m_{CS} = 9$



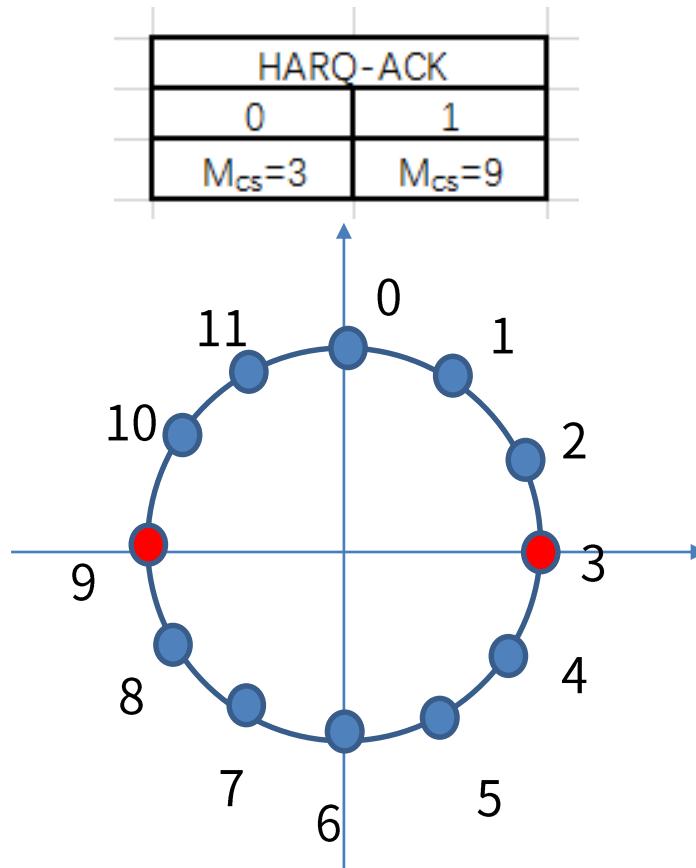
一个UE占用4个序列  
这样，总共12个序列情况下，  
可以给3个用户使用

# SR调度请求反馈

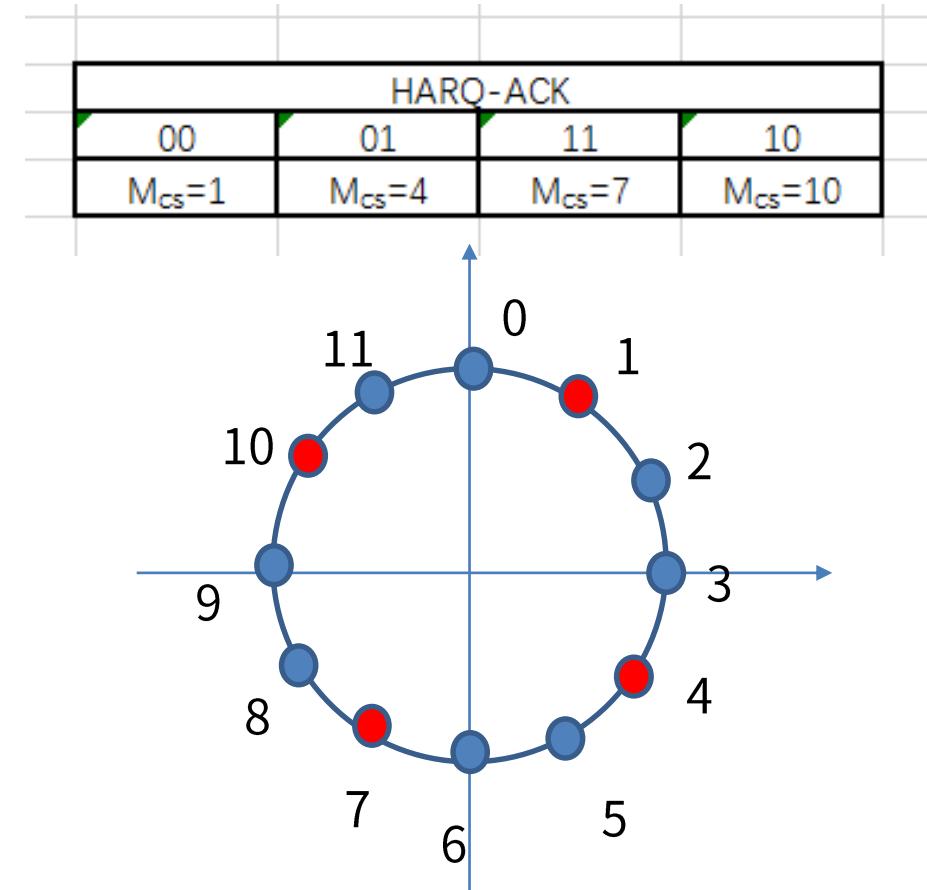
只发SR  $M_{cs}=0$



1bit ACK-NACK+SR

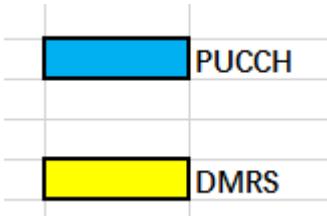
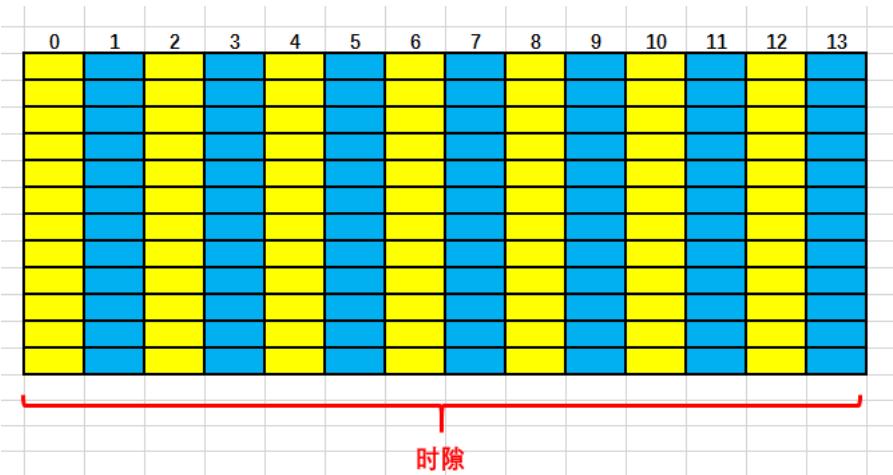


2bit ACK-NACK+SR



# PUCCH format1

PUCCH 格式1属于长格式，PUCCH格式1在频域上占用1个RB，在时域上占用4-14个符号承载的UCI为 1bit或者2bit，用于发送HARQ的ACK、NACK反馈，也可以携带SR信息。



```
PUCCH-format1 ::=  
    initialCyclicShift  
    nrofSymbols  
    startingSymbolIndex  
    timeDomainOCC  
}
```

```
SEQUENCE {  
    INTEGER(0..11),  
    INTEGER (4..14),  
    INTEGER(0..10),  
    INTEGER(0..6)
```

initialCyclicShift：初始循环移位 (0-11)

nrofSymbols: 时域符号个数 (4-14)

startingSymbolIndex : 时域开始符号索引 (0..10)

timeDomainOCC : 时域OCC配置的索引 (0..6)

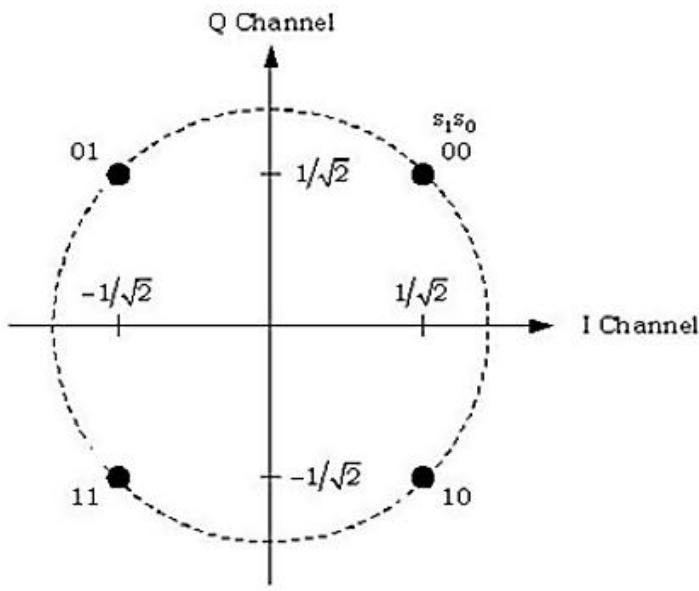
DMRS位置:时域固定为符号0,2,4,6…12， 占用RB内的全部子载波

格式1支持12\*OCC个用户复用

# PUCCH format1 序列

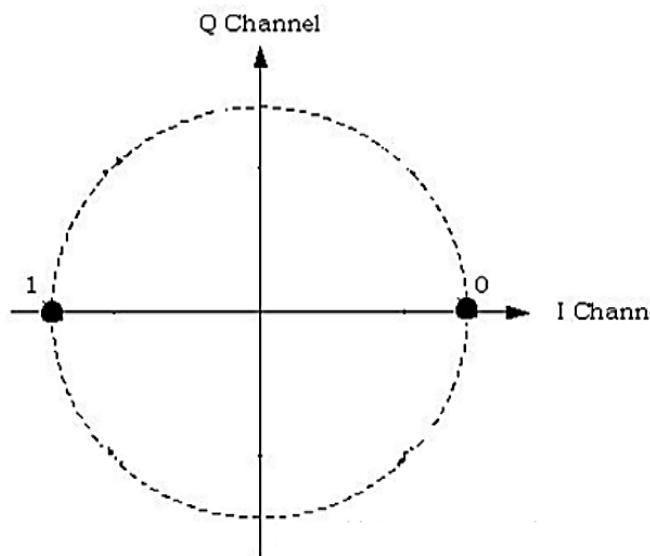
对于PUCCH 格式1：Mcs=0，也就是没有通过序列来区分0和1

为了区分ACK和NACK，格式1采样了调制方式区分0和1，  
调制符号与序列相乘，最终生成了PUCCH发送的序列。  
这里面，序列的作用是区分用户



4个点分别对应4个相位： $\pi/4, 3\pi/4, 5\pi/4, 7\pi/4$

QPSK



BPSK

M0的变化来区分用户

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

Cyclic Shift

Diagram illustrating the cyclic shift mechanism. A pink arrow labeled "Cyclic Shift" points to the term  $(l + l')$  in the equation. Green arrows point to the terms  $m_0$ ,  $m_{es}$ , and  $n_{es}(n_{s,f}^{\mu}, l + l')$  to indicate they are part of the cyclic shift calculation.

1bit使用BPSK，2bit使用QPSK

# PUCCH format0序列

为了进一步搞清楚格式1和格式0的区别，我们举个例子

假设序列里面的每一个值都使用 $\beta$ 来表示，这样，我们的基序列就是这样：

$\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7, \beta_8, \beta_9, \beta_{10}, \beta_{11}$

每一个值对应圆上的一个点

对于格式0，使用序列来代表0和1，这就形成如下局面：

0:  $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7, \beta_8, \beta_9, \beta_{10}, \beta_{11}$

1:  $\beta_6, \beta_7, \beta_8, \beta_9, \beta_{10}, \beta_{11}, \beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5$

0:  $\beta_{11}, \beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7, \beta_8, \beta_9, \beta_{10}$

1:  $\beta_5, \beta_6, \beta_7, \beta_8, \beta_9, \beta_{10}, \beta_{11}, \beta_0, \beta_1, \beta_2, \beta_3, \beta_4$

0:  $\beta_{10}, \beta_{11}, \beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7, \beta_8, \beta_9$

1:  $\beta_4, \beta_5, \beta_6, \beta_7, \beta_8, \beta_9, \beta_{10}, \beta_{11}, \beta_0, \beta_1, \beta_2, \beta_3$

0:  $\beta_9, \beta_{10}, \beta_{11}, \beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7, \beta_8$

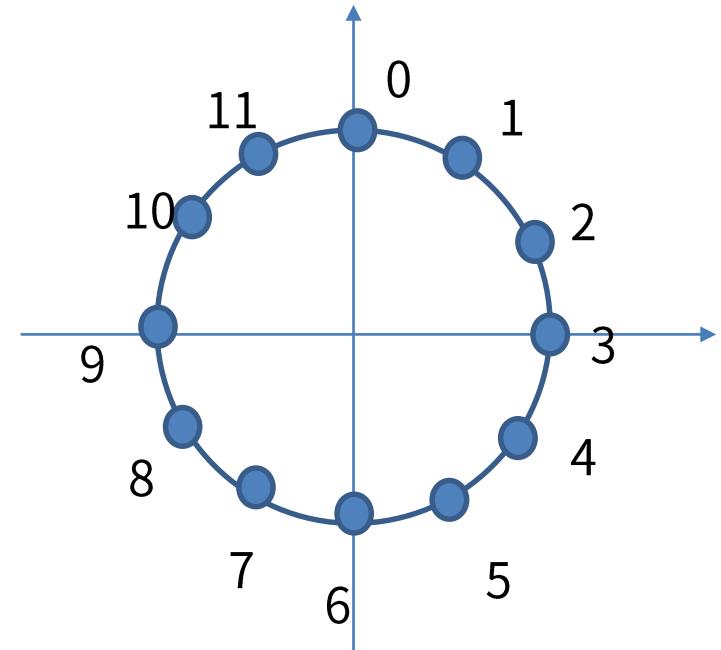
1:  $\beta_3, \beta_4, \beta_5, \beta_6, \beta_7, \beta_8, \beta_9, \beta_{10}, \beta_{11}, \beta_0, \beta_1, \beta_2$

0:  $\beta_8, \beta_9, \beta_{10}, \beta_{11}, \beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7$

1:  $\beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7, \beta_8, \beta_9, \beta_{10}, \beta_{11}, \beta_0, \beta_1$

0:  $\beta_7, \beta_8, \beta_9, \beta_{10}, \beta_{11}, \beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6$

1:  $\beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7, \beta_8, \beta_9, \beta_{10}, \beta_{11}, \beta_0$

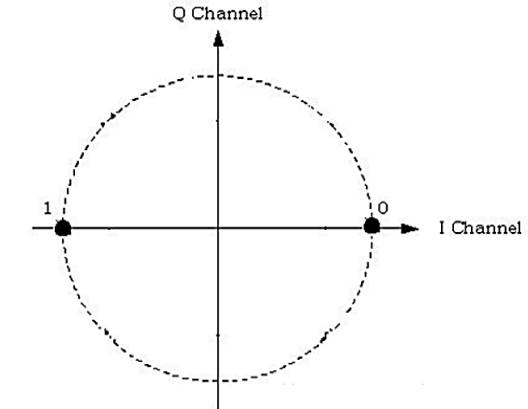


一个用户用一对序列，因此，格式0里，序列既充当区分0和1，也充当区分用户。

# PUCCH format1 序列

为了方便说明格式1，我们将序列用一个符号 $\alpha$ 来表示：

$$\alpha = \{\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7, \beta_8, \beta_9, \beta_{10}, \beta_{11}\}$$



当UE使用格式1时，通过调制区分0和1，以最简单的BPSK为例，

0-----cos(0)	发送0的时候: $\cos(0) * \alpha_0$
1-----cos( $\pi$ )	发送1的时候: $\cos(\pi) * \alpha_0$

以此类推第二个用户会使用循环位移产生新的序列，就会这样：

发送0的时候: $\cos(0) * \alpha_1$
发送1的时候: $\cos(\pi) * \alpha_1$

格式1中序列的作用，仅仅是区分用户，不区分0和1

```
PUCCH-format1 ::=  
    initialCyclicShift    初始循环移位  
    nrofSymbols  
    startingSymbolIndex  
    timeDomainOCC  
}
```

```
SEQUENCE {  
    INTEGER(0..11),  
    INTEGER (4..14),  
    INTEGER(0..10),  
    INTEGER(0..6)
```

用户	发送信息	
	0	1
用户1	$\alpha_0 * \cos(0)$	$\alpha_0 * \cos(\pi)$
用户2	$\alpha_1 * \cos(0)$	$\alpha_1 * \cos(\pi)$
用户3	$\alpha_2 * \cos(0)$	$\alpha_2 * \cos(\pi)$
用户4	$\alpha_3 * \cos(0)$	$\alpha_3 * \cos(\pi)$
用户5	$\alpha_4 * \cos(0)$	$\alpha_4 * \cos(\pi)$
用户6	$\alpha_5 * \cos(0)$	$\alpha_5 * \cos(\pi)$
用户7	$\alpha_6 * \cos(0)$	$\alpha_6 * \cos(\pi)$
用户8	$\alpha_7 * \cos(0)$	$\alpha_7 * \cos(\pi)$
用户9	$\alpha_8 * \cos(0)$	$\alpha_8 * \cos(\pi)$
用户10	$\alpha_9 * \cos(0)$	$\alpha_9 * \cos(\pi)$
用户11	$\alpha_{10} * \cos(0)$	$\alpha_{10} * \cos(\pi)$
用户12	$\alpha_{11} * \cos(0)$	$\alpha_{11} * \cos(\pi)$

# PUCCH format1

基于以上讲的内容，我们发现，一个格式1的PUCCH可以12个用户复用，但是，这并没完，格式1为了进一步支持更多的用户，格式1支持了时域扩展。

timeDomainOCC: 时域正交扩展序列

Table 6.3.2.4.1-2: Orthogonal sequences  $w_i(m) = e^{j2\pi\phi(m)/N_{SF,m}^{\text{PUCCH},1}}$  for PUCCH format 1.

$N_{SF,m'}^{\text{PUCCH},1}$	$i = 0 \leftrightarrow$	$i = 1 \leftrightarrow$	$i = 2 \leftrightarrow$	$i = 3 \leftrightarrow$	$i = 4 \leftrightarrow$	$i = 5 \leftrightarrow$	$i = 6 \leftrightarrow$
1 $\leftrightarrow$	[0] $\leftrightarrow$	$\leftrightarrow$	$\leftrightarrow$	$\leftrightarrow$	$\leftrightarrow$	$\leftrightarrow$	$\leftrightarrow$
2 $\leftrightarrow$	[0 0] $\leftrightarrow$	[0 1] $\leftrightarrow$	$\leftrightarrow$	$\leftrightarrow$	$\leftrightarrow$	$\leftrightarrow$	$\leftrightarrow$
3 $\leftrightarrow$	[0 0 0] $\leftrightarrow$	[0 1 2] $\leftrightarrow$	[0 2 1] $\leftrightarrow$	$\leftrightarrow$	$\leftrightarrow$	$\leftrightarrow$	$\leftrightarrow$
4 $\leftrightarrow$	[0 0 0 0] $\leftrightarrow$	[0 2 0 2] $\leftrightarrow$	[0 0 2 2] $\leftrightarrow$	[0 2 2 0] $\leftrightarrow$	$\leftrightarrow$	$\leftrightarrow$	$\leftrightarrow$
5 $\leftrightarrow$	[0 0 0 0 0] $\leftrightarrow$	[0 1 2 3 4] $\leftrightarrow$	[0 2 4 1 3] $\leftrightarrow$	[0 3 1 4 2] $\leftrightarrow$	[0 4 3 2 1] $\leftrightarrow$	$\leftrightarrow$	$\leftrightarrow$
6 $\leftrightarrow$	[0 0 0 0 0 0] $\leftrightarrow$	[0 1 2 3 4 5] $\leftrightarrow$	[0 2 4 0 2 4] $\leftrightarrow$	[0 3 0 3 0 3] $\leftrightarrow$	[0 4 2 0 4 2] $\leftrightarrow$	[0 5 4 3 2 1] $\leftrightarrow$	$\leftrightarrow$
7 $\leftrightarrow$	[0 0 0 0 0 0 0] $\leftrightarrow$	[0 1 2 3 4 5 6] $\leftrightarrow$	[0 2 4 6 1 3 5] $\leftrightarrow$	[0 3 6 2 5 1 4] $\leftrightarrow$	[0 4 1 5 2 6 3] $\leftrightarrow$	[0 5 3 1 6 4 2] $\leftrightarrow$	[0 6 5 4 3 2 1] $\leftrightarrow$

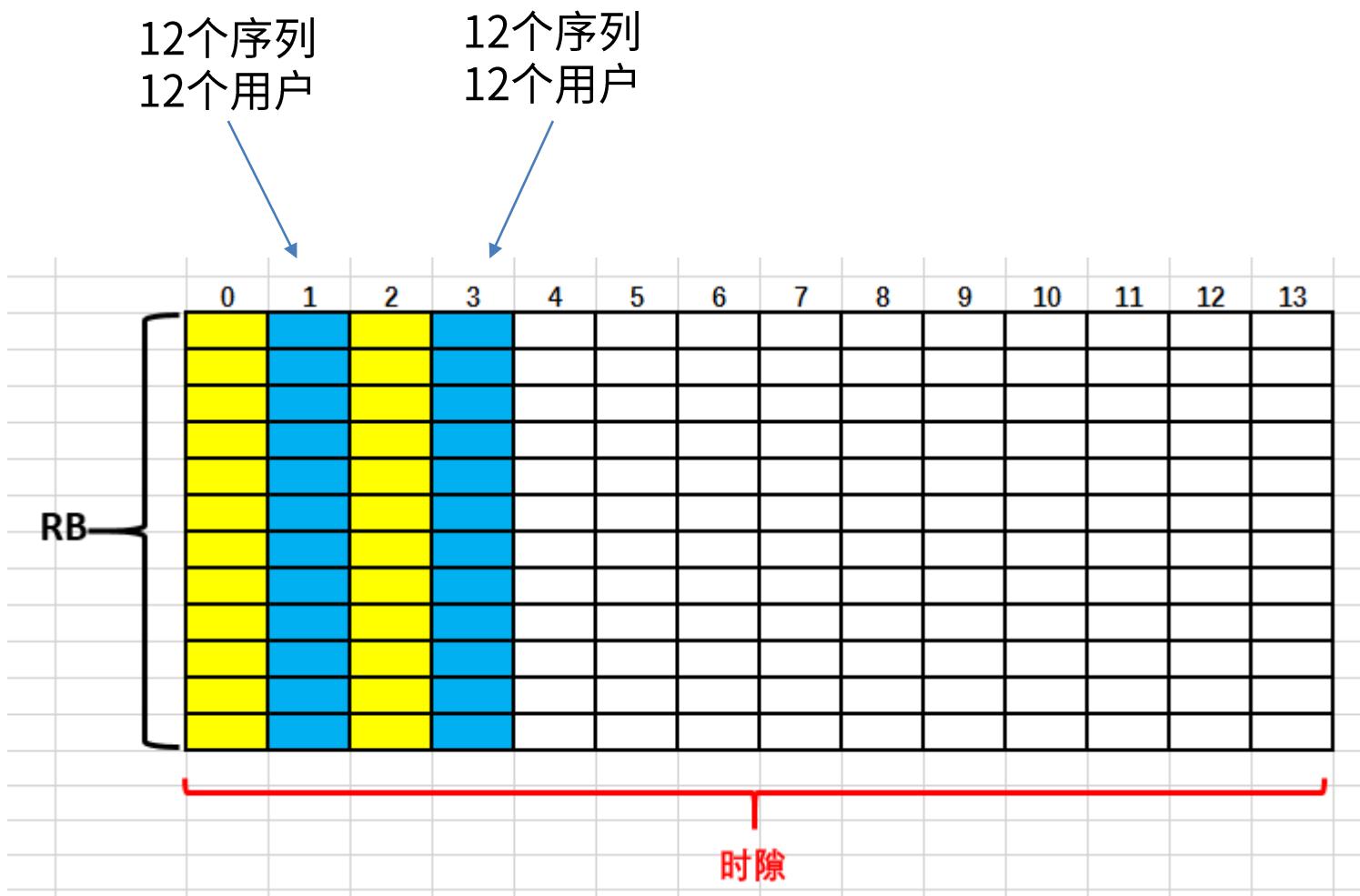
$\leftarrow$

这是什么意思呢？

# 时域OCC (正交序列)

假设格式1时域占用4个符号，这个时候，PUCCH占用符号1和符号3这两个RB。

前面我们知道一个RB就可以通过序列，来给12个用户去用，而2个RB，就需要区分24个用户了，但是这两个RB所用的序列都是这12个，也就是说，不同的用户，可能会用相同的序列，这个时候仅仅靠序列，是无法区分用户了。



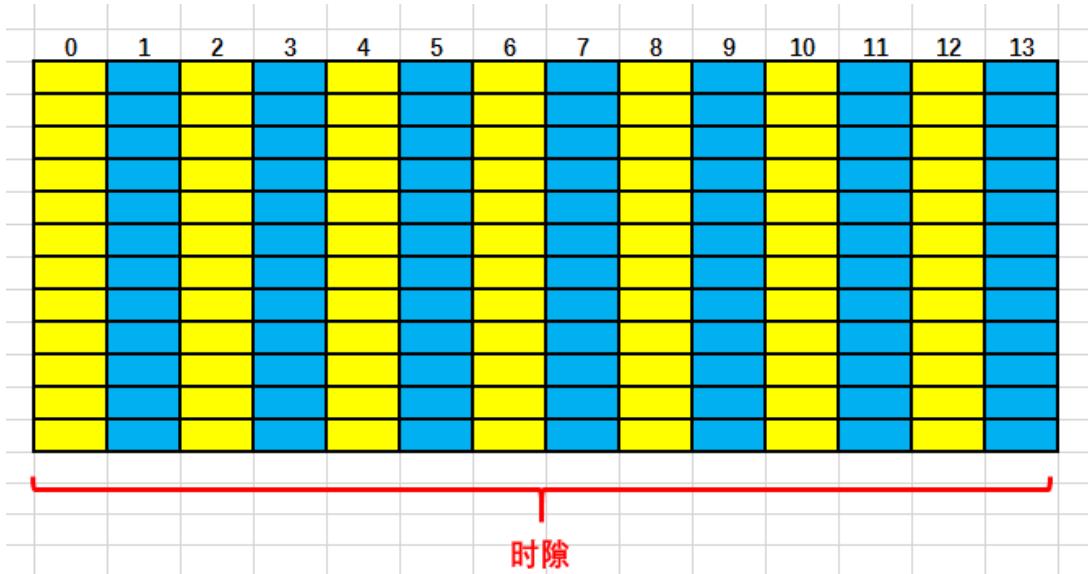
OCC正交序列，就是让UE在使用相同的序列的时候，出现在不同的时域位置，以来区分用户

最终，格式1的数据，是这样一个过程

调制\*序列\*OCC ← 在哪个RB上

0 or 1 使用哪个序列的用户

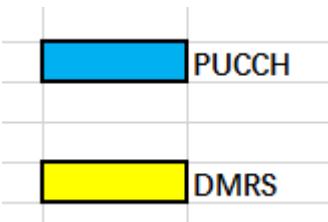
# PUCCH format1最大复用用户数



```
PUCCH-format1 ::=  
  initialCyclicShift  
  nrofSymbols  
  startingSymbolIndex  
  timeDomainOCC  
}
```

→ 格式1时域占用符号最大14  
→ OCC: 最大7

因此，格式1最多支持 $12 \times 7 = 84$ 个用户复用



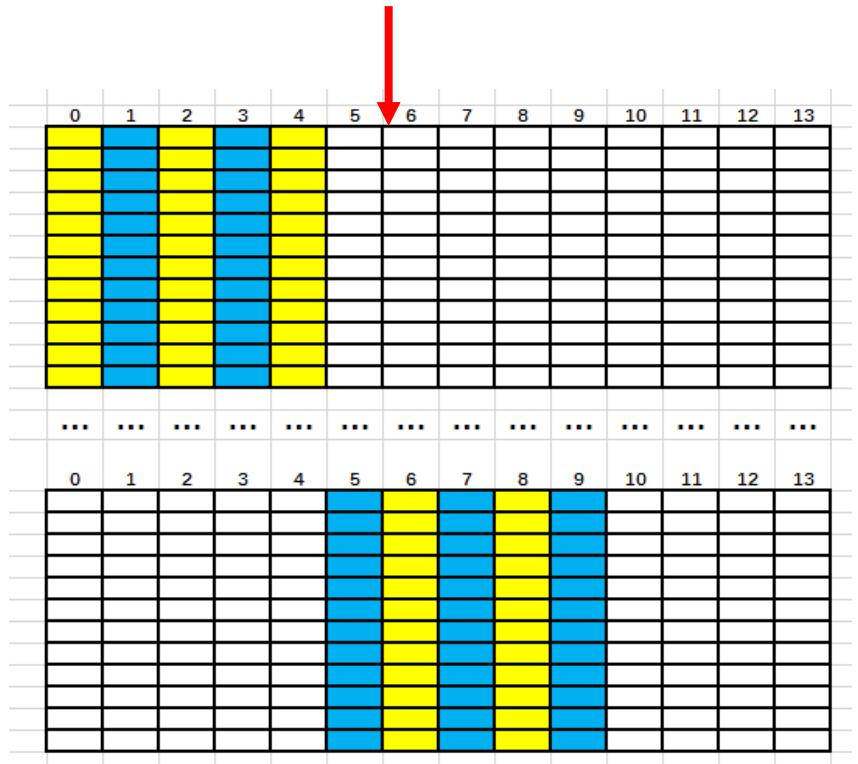
# PUCCH format1跳频

格式1在跳频的情况下，取决于第一跳的PUCCH符号个数

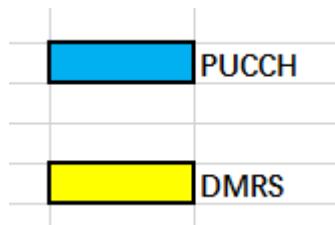
Table 6.3.2.4.1-1: Number of PUCCH symbols and the corresponding  $N_{SF,m'}^{\text{PUCCH},1}$

PUCCH length, $N_{\text{symb}}^{\text{PUCCH},1}$	No intra-slot hopping $m' = 0$		$N_{SF,m'}^{\text{PUCCH},1}$	Intra-slot hopping $m' = 1$
	$m' = 0$	$m' = 1$		
4	2	1	1	1
5	2	1	1	1
6	3	1	2	
7	3	1	2	
8	4	2	2	
9	4	2	2	
10	5	2	3	
11	5	2	3	
12	6	3	3	
13	6	3	3	
14	7	3	4	

举个例子，比如格式1的总符号数是10，在跳频的情况下，第一跳只有2个PUCCH符号，因此，复用用户数为 $2 * 12 = 24$ 个用户



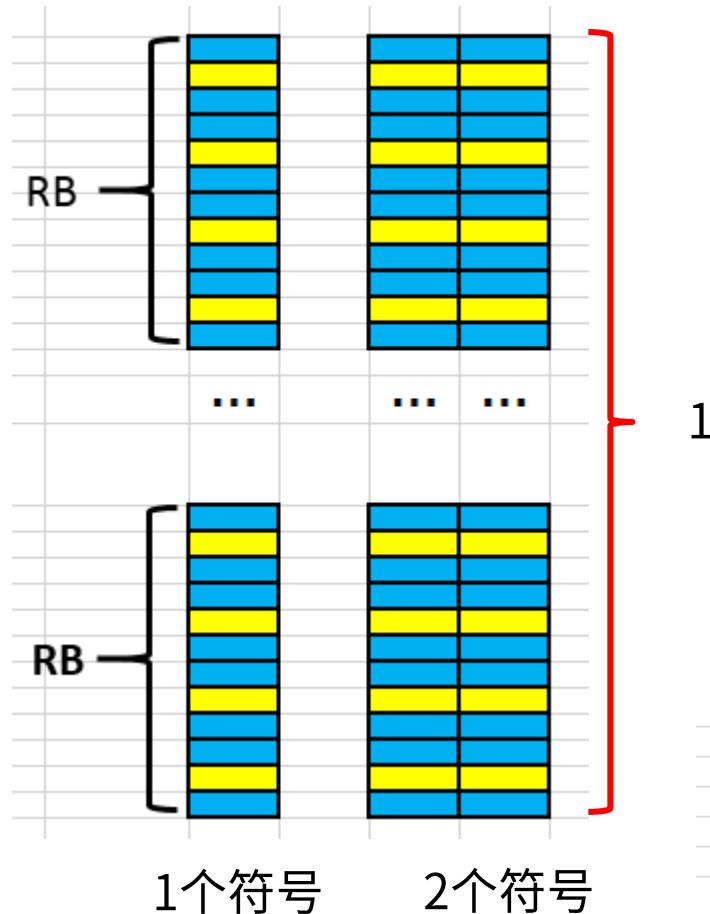
格式1在不跳频的情况下，最多支持 $12 * 7 = 84$ 个用户复用



格式1在跳频的情况下，最多支持 $12 * 3 = 36$ 个用户复用

# PUCCH format2

PUCCH format2可以发送HARQ-ACK, SR以及CSI信息  
由于CSI信息长度较大，格式0和1都不能传CSI  
PUCCH在时域上占用1-2个符号，在频域上占用1到16个RB



```
PUCCH-format2 ::=  
    nrofPRBs  
    nrofSymbols  
    startingSymbolIndex  
}
```

```
SEQUENCE {  
    INTEGER (1..16),  
    INTEGER (1..2),  
    INTEGER (0..13)
```

nrofPRBs : 频域占用PRB个数      INTEGER (1..16),

nrofSymbols : 时域占用符号数      INTEGER (1..2),

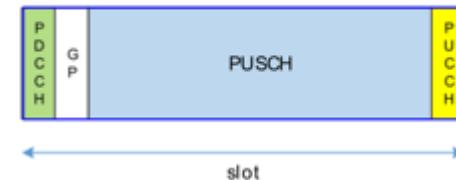
startingSymbolIndex : 起始符号索引      INTEGER(0..13)

PUCCH格式2不支持多UE复用

DMRS位置：固定为一个RB中的1,4,7,10符号位置

一般位于时隙最后

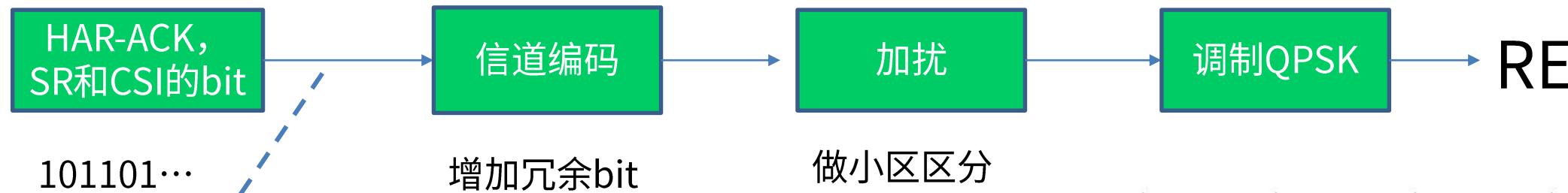
a) Short duration PUCCH



# PUCCH format2

格式2用于传输大于2bit的 UCI，这个时候，就不是单纯的1bit对应个啥序列，2bit对应个啥序列了  
格式2采样的是PUCCH的RB直接承载信息的模式。

对于需要传输的HAR-ACK，SR和CSI的bit信息，进行信道编码，加扰，调制，映射到RE中



101101…

增加冗余bit  
抗干扰

做小区区分

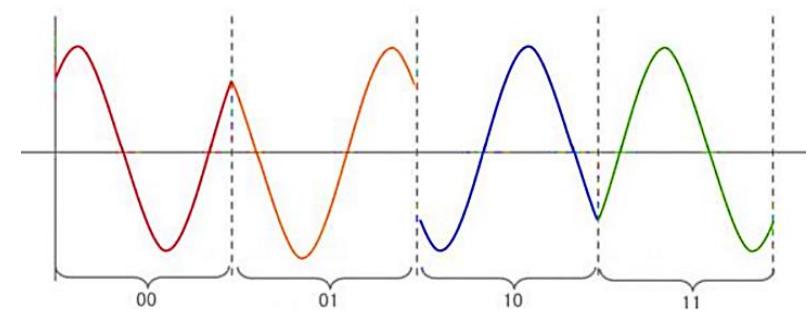
RE

110011110011…

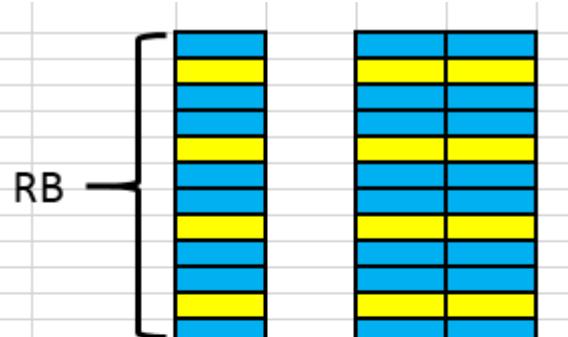
110011110011  
10100101…

bit数>11的话还要加CRC校验位

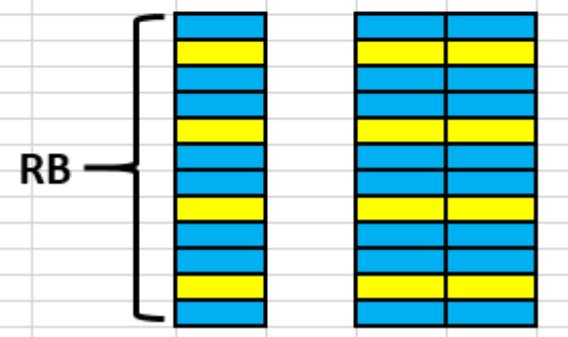
对于某天线来说，  
它只想解码本小区的UE



# PUCCH format2



格式2最少占用1个RB，一个RB当中，有8个RE是用来传数据的，而每个RE使用的调制方式都是QPSK，因此，PUCCH一个RB能传的bit数是 $8 \times 2 = 16$ bit



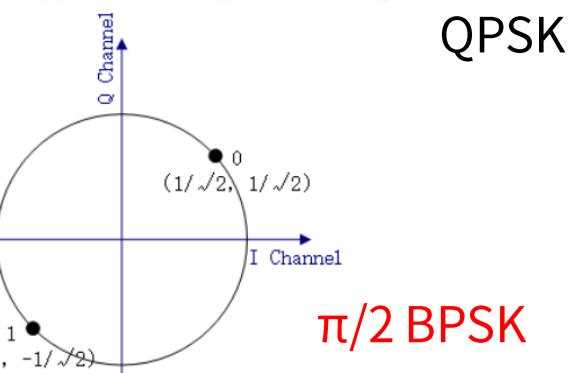
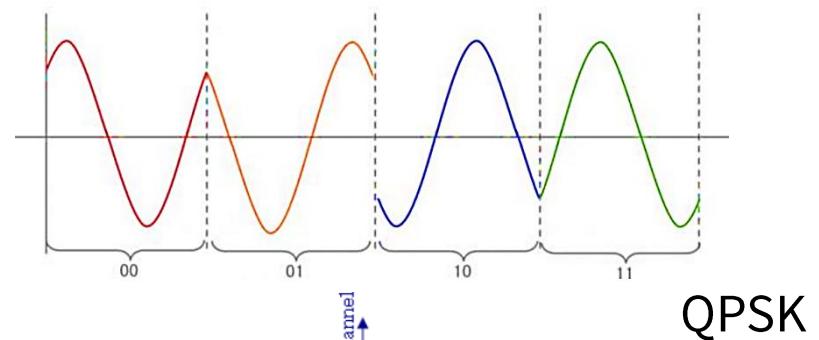
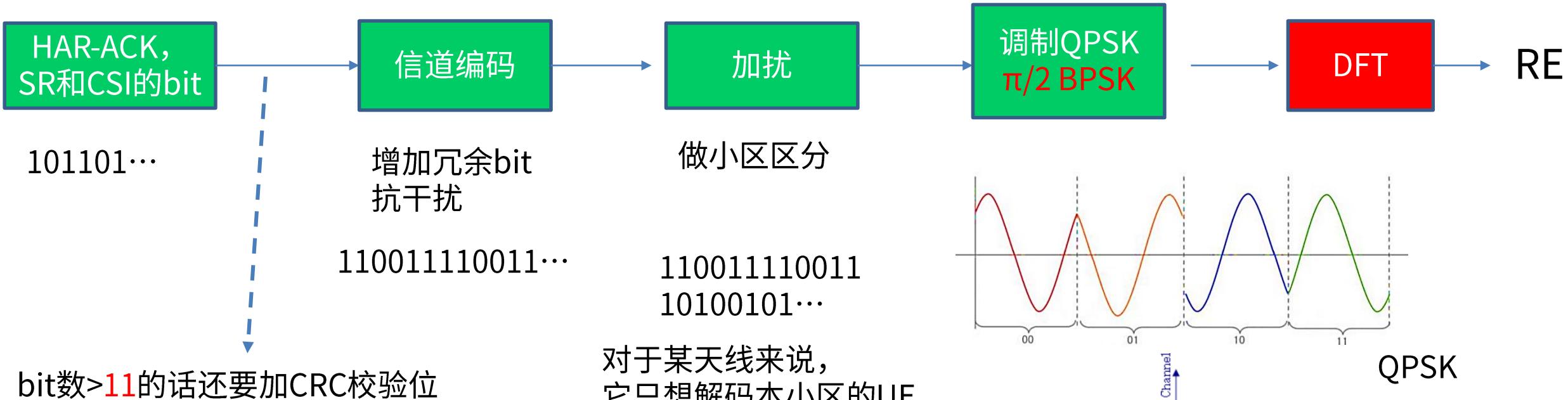
格式2频域最多16个RB，时域2个符号，因此，格式2最多占用系统资源32RB  
而一个RB能传16bit数据，因此，格式2最多传 $32 \times 16 = 512$ bit

需要注意的是，这里的bit数并不是UCI的bit数，是UCI信道编码和加扰之后的bit  
实际传送的UCI bit 比这个小

# PUCCH format3

格式3依然采样的是PUCCH的RB直接承载信息的模式，整体UCI处理流程与格式2基本一致

与格式2的区别是，增加了1个步骤DFT（降低峰均比），调制也增加了一种 $\pi/2$  BPSK

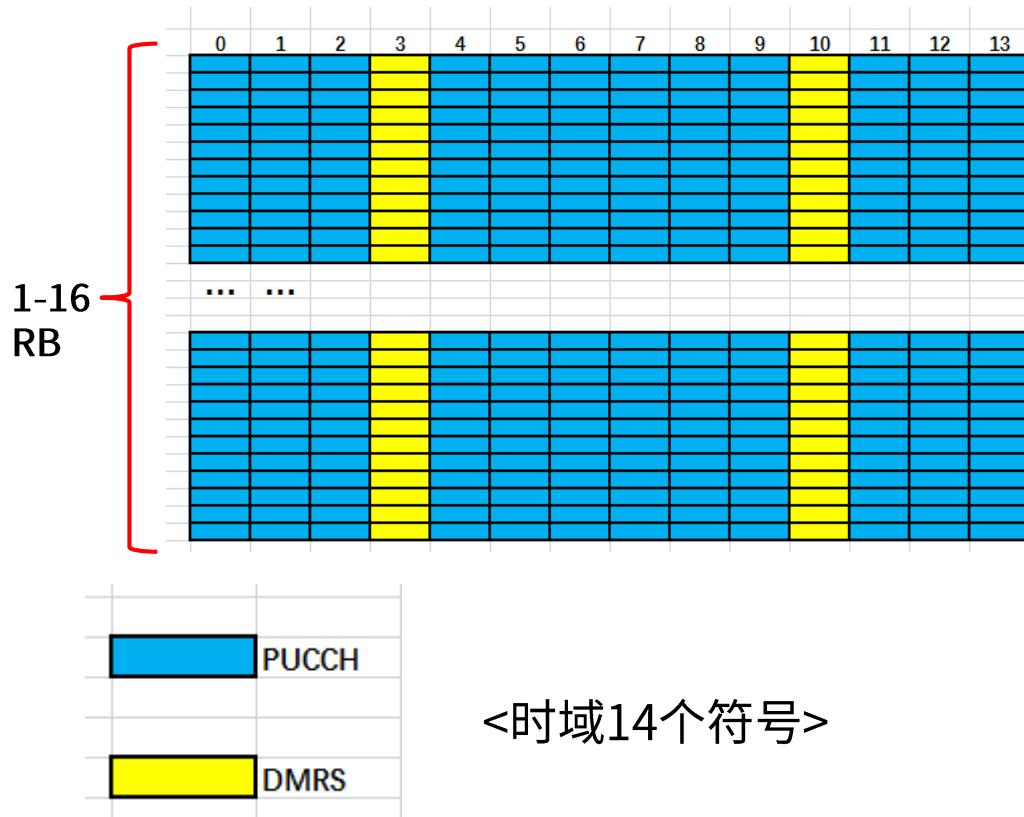


# PUCCH format3

PUCCH format3可以发送HARQ-ACK, SR以及CSI信息

PUCCH在时域上占用4-14个符号，在频域上占用1到16个RB (RB中取值为2、3、5的幂次方的乘积的RB数量)

频域RB数实际取值只有12个：1.2.3.4.5.6.8.9.10.12.15.16



```
PUCCH-format3 ::=  
    nrofRBs  
    nrofSymbols  
    startingSymbolIndex  
}
```

```
SEQUENCE {  
    INTEGER (1..16),  
    INTEGER (4..14),  
    INTEGER(0..10)
```

nrofRBs : 占用PRB个数                   INTEGER (1..16),

nrofSymbols : 占用符号数                   INTEGER (4..14),

startingSymbolIndex : 起始符号索引    INTEGER(0..10)

PUCCH 格式3不支持多UE复用，当一个时刻存在多个UE，则在频域上做区分。

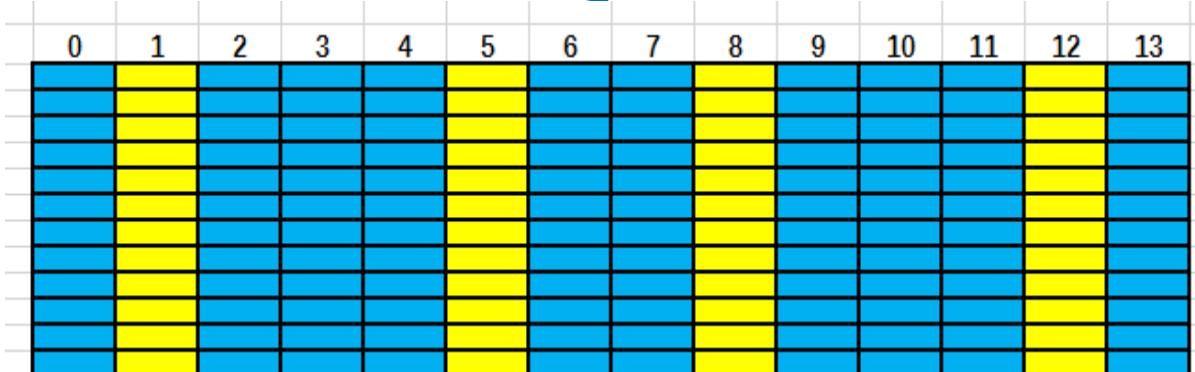
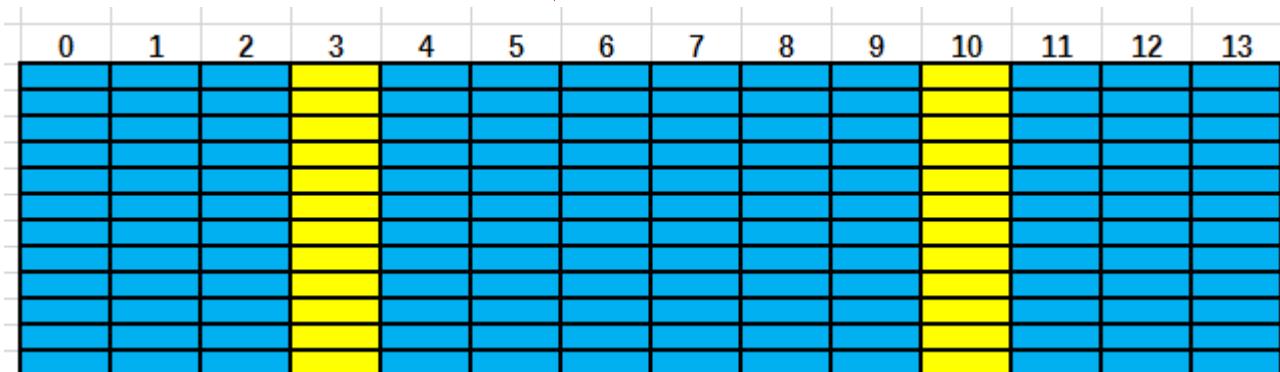
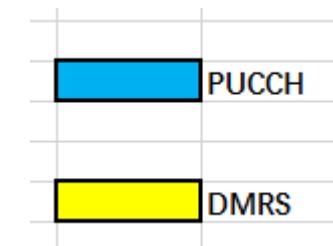
DMRS的位置不固定，根据PUCCH的时域长度，查表确定

# PUCCH format3 DMRS位置

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	3, 10	1, 5, 8, 12	

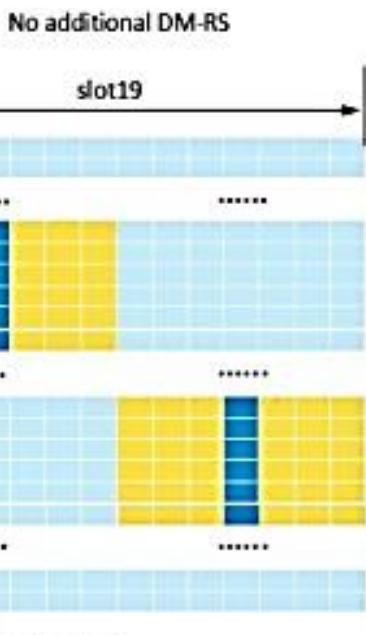
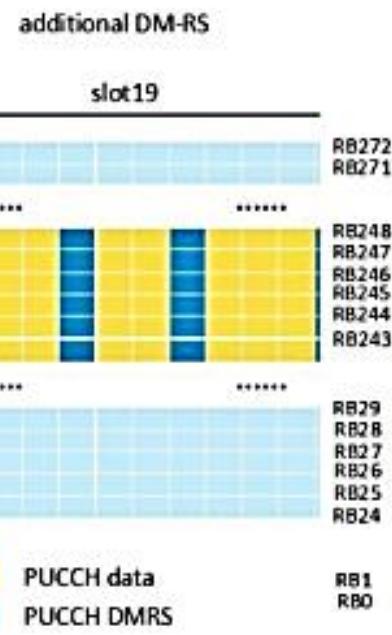
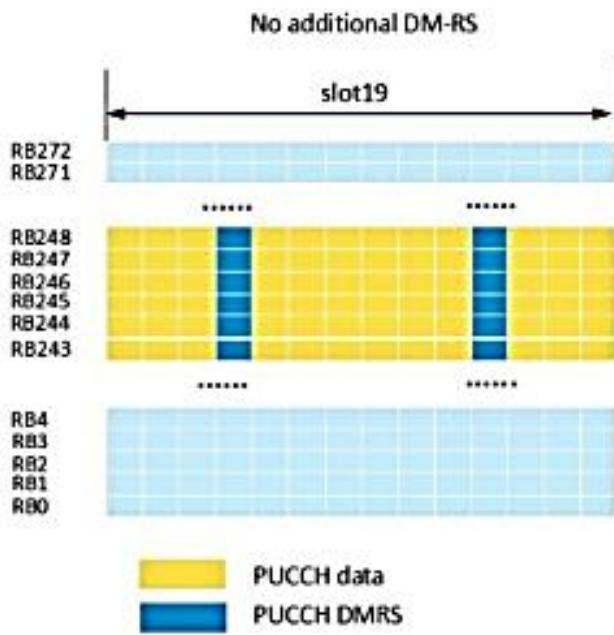
PUCC 格式3的DMRS位置与PUCCH的时域长度有关  
是否存在Additional DM-RS可以通过信令解码得知



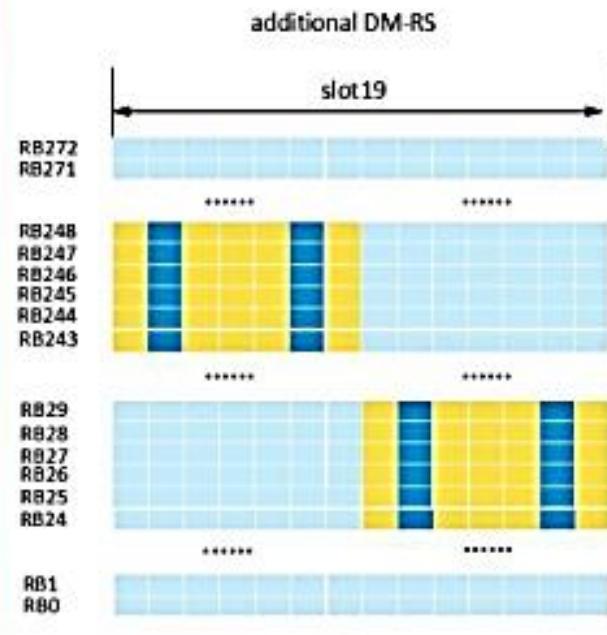
# DMRS位置区别

跳频与不跳频的DMRS位置

不跳频



跳频



# PUCCH format3传送最少bit个数

格式3最少占用的资源情况：

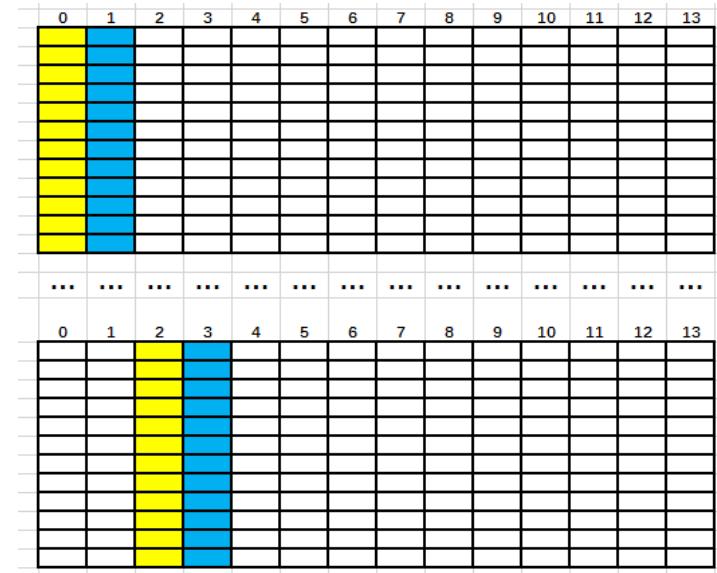
频域1个RB，时域4个符号，跳频，其中有2个RB是DMRS，因此可以用的RB为2个

按照  $\pi/2$  BPSK调制

1个RE传1bit

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	



一个RB=12个RE，所以2个RB=24个RE

1个RE传1bit，因此，这种情况下，格式3的PUCCH最多可传 $24 * 1 = 24$ bit

# PUCCH format3传送最多bit个数

格式3最多占用的资源情况：

频域16个RB，时域14个符号，不跳频，其中有2个RB是DMRS，因此可以用的RB为12个 按照 QPSK调制  
1个RE传2bit

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	



总RB数是 $16 \times 14 = 224$ 个

可用的RB数= $16 \times 12 = 192$ 个

可用bit= $192 \times 12 \times 2 = 4608$ bit

# Additional DM-RS

在高层信令PUCCH-Config当中：

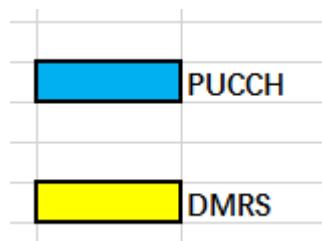
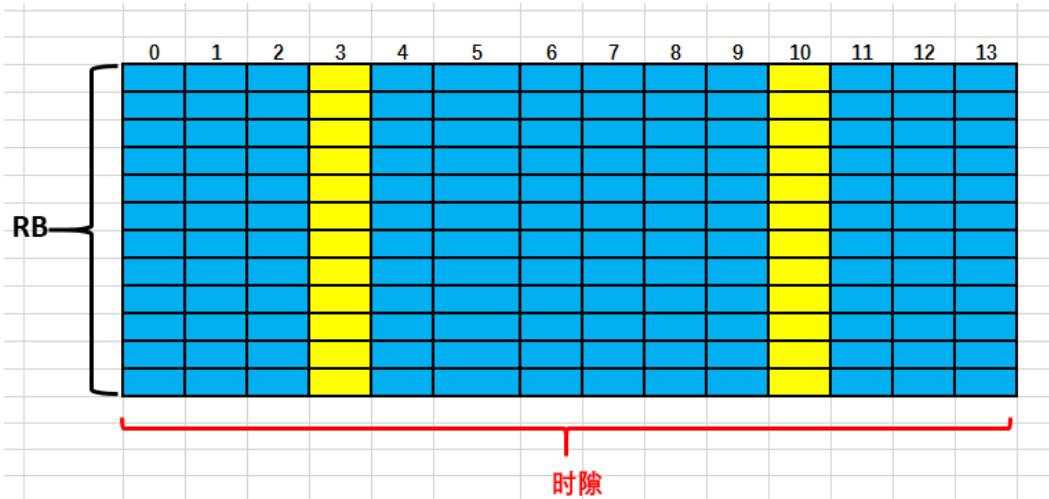
PUCCH-FormatConfig (格式配置) 里面有是否支持Additional DM-RS

```
PUCCH-FormatConfig ::= SEQUENCE {  
    interslotFrequencyHopping      ENUMERATED {enabled}    OPTIONAL, -- Need R  
    additionalDMRS                ENUMERATED {true}       OPTIONAL, -- Need R  
    maxCodeRate                   PUCCH-MaxCodeRate     OPTIONAL, -- Need R  
    nrofSlots                     ENUMERATED {n2,n4,n8}  OPTIONAL, -- Need S  
    pi2PBSK                       ENUMERATED {enabled}    OPTIONAL, -- Need R  
    simultaneousHARQ-ACK-CSI      ENUMERATED {true}       OPTIONAL -- Need R  
}
```

# PUCCH format4

PUCCH format4可以发送HARQ-ACK, SR以及CSI信息

PUCCH在时域上占用4-14个符号，在频域上占用1个RB



<时域14个符号>

```
PUCCH-format4 ::=  
nrofSymbols  
occ-Length  
occ-Index  
startingSymbolIndex  
}
```

```
SEQUENCE {  
INTEGER (4..14),  
ENUMERATED {n2,n4},  
ENUMERATED {n0,n1,n2,n3},  
INTEGER(0..10)
```

nrofSymbols: 占用符号数 (4..14),

occLength: OCC长度{n2,n4},

occ-Index: OCC索引号 {n0,n1,n2,n3},

startingSymbolIndex: 起始符号索引 (0..10)

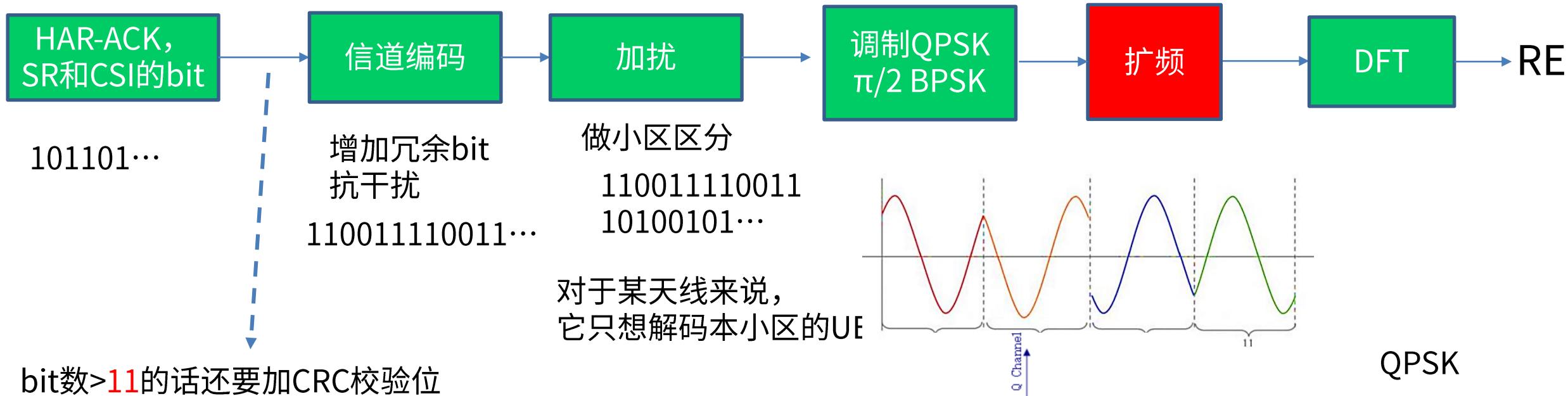
DMRS的位置不固定，根据PUCCH的时域长度，查表确定  
(与格式3共用一个表格)

格式4支持2-4用户复用

# PUCCH format4

格式4依然采样的是PUCCH的RB直接承载信息的模式，整体UCI处理流程与格式3基本一致

与格式3的区别是：增加了扩频这一步，使格式4可以支持用户复用



bit数>11的话还要加CRC校验位

# PUCCH format4 扩频

为了支持相同资源的复用，格式4使用类似CDMA的扩频码OCC，来区分不同的用户。

我们之前讲过扩频码的逻辑，这里可以再复习一下：

A用户数据\*Oa----- A用户数据\*Oa\*Oa=》 A用户的数据

B用户数据\*Ob----- B用户数据\*Ob\*Ob=》 B用户的数据

A用户数据\*Oa----- A用户数据\*Oa\*Ob=》 解码失败

Table 6.3.2.6.3-1: Orthogonal sequences  $w_n(m)$  for PUCCH format 4 when  $N_{SF}^{PUCCH,4} = 2$ .

$n \Leftarrow$	$w_n \Leftarrow$
0 $\Leftarrow$	[+1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1]
1 $\Leftarrow$	[+1 +1 +1 +1 +1 +1 -1 -1 -1 -1 -1 -1]

扩频码：2个

Table 6.3.2.6.3-2: Orthogonal sequences  $w_n(m)$  for PUCCH format 4 when  $N_{SF}^{PUCCH,4} = 4$ .

$n \Leftarrow$	$w_n \Leftarrow$
0 $\Leftarrow$	[+1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1]
1 $\Leftarrow$	[+1 +1 +1 -j -j -j -1 -1 -1 +j +j +j]
2 $\Leftarrow$	[+1 +1 +1 -1 -1 -1 +1 +1 +1 -1 -1 -1]
3 $\Leftarrow$	[+1 +1 +1 +j +j +j -1 -1 -1 -j -j -j]

扩频码：4个

PUCCH-format4 ::=  
nrOfSymbols  
occ-Length  
occ-Index  
startingSymbolIndex

SEQUENCE {  
INTEGER (4..14),  
ENUMERATED {n2,n4},  
ENUMERATED {n0,n1,n2,n3},  
INTEGER(0..10)}

occLength: OCC长度{n2,n4},

occ-Index: OCC索引号 {n0,n1,n2,n3},

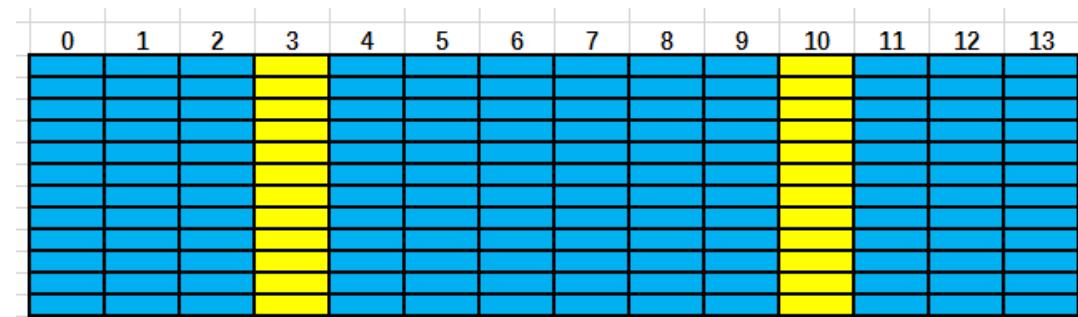
# PUCCH format4 扩频

格式4的DMRS位置设计与格式3是公用一张表，位置一样

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	

时域14个符号时



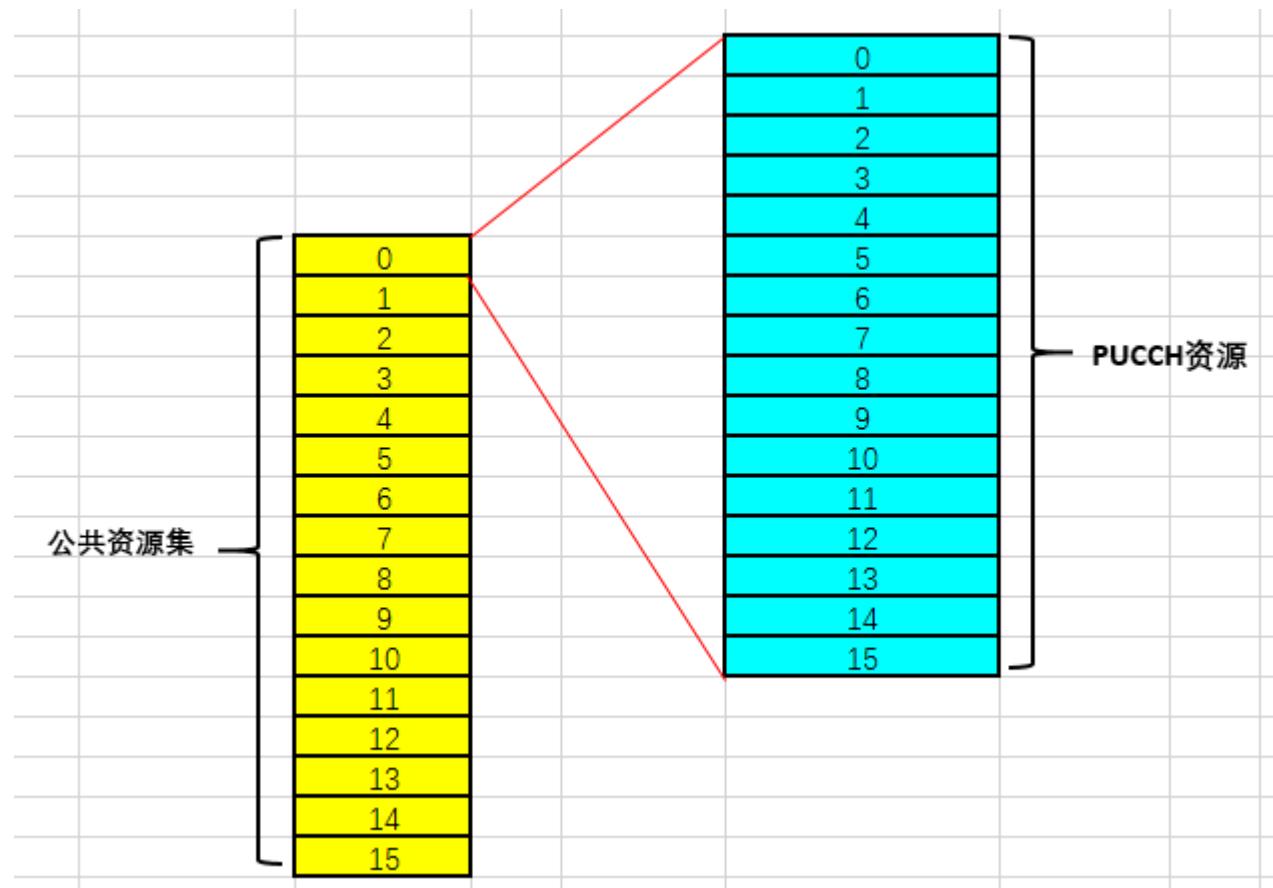
# PUCCH format总结

格式		格式0	格式1	格式2	格式3	格式4
时域	时隙内起始符号位置	0-13	4-10	0-13	0-10	0-10
	占用符号个数	1-2	4-14	1-2	4-14	4-14
频域	PRB数	1	1	1-16	1.2.3.4.5.6.8.9.10.12.15.16	1
复用用户方法	每对序列代表1个用户	序列区分一个RB内UE 时域OCC区分不同RB	不能复用	不能复用	正交扩频OCC	
复用用户数	1bit : 6个 2bit : 3个 仅传输SR, 可以复用12个UE	12*OCC	1	1	2个或者4个	
承载信息方法	序列代表0和1	BPSK或者QPSK	资源直接承载	资源直接承载	资源直接承载	
能够承载UCI bit数	1,2	1,2	>2	>2	>2	
DMRS	没有	时域固定为符号0,2,4,6…12	频域固定为一个RB中的1,4,7,10符号位置	时域位置查表	时域位置查表	
格式类型	短格式	长格式	短格式	长格式	长格式	

# 公共PUCCH的资源配置

pucch-ResourceCommon

UE在idle状态的时候，使用公共资源集反馈ACK-NACK



一个小区使用的公共资源集可以在16种可能中选择。  
而每一个资源集，又含有16个PUCCH资源

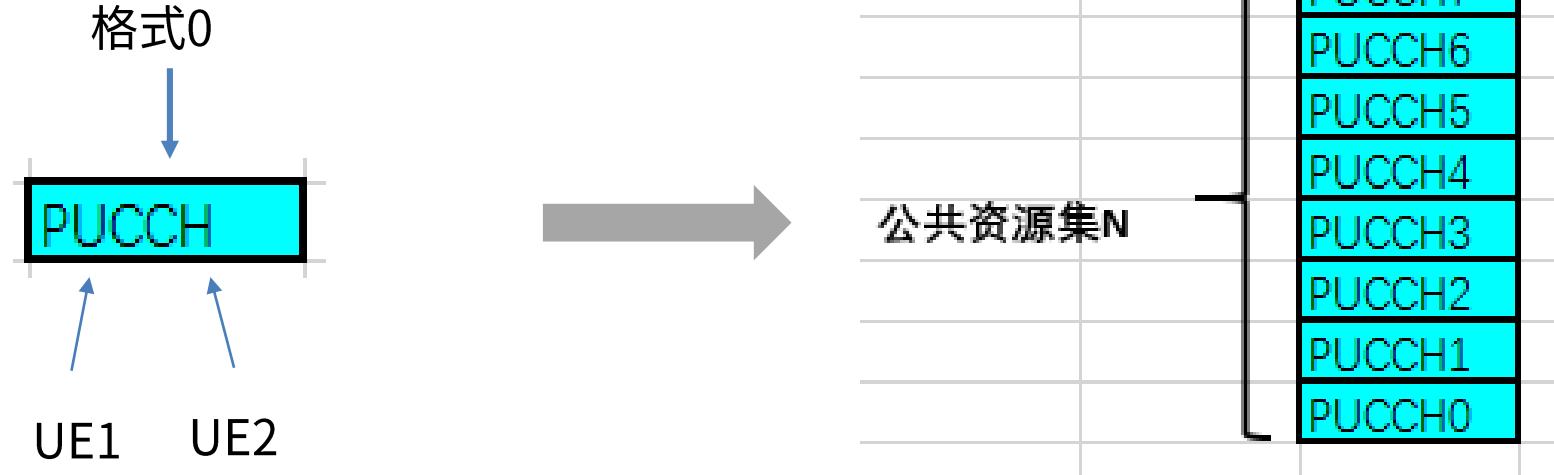
这里面的PUCCH资源，其实代表了能装16个用户的PUCCH资源

相邻小区使用的公共资源集不同，  
为了避免干扰

# 举个例子

16个PUCCH资源代表了能装16个用户的PUCCH资源

比如：一个格式为0的PUCCH，理论上可以支持6个用户，那我也可以让他支持2个用户，然后弄8个格式0的PUCCH，就能容纳16个用户，也就形成了一个公共资源集。



这个资源集就能装16个用户  
也称为有16个PUCCH资源

我们让格式0的PUCCH仅仅复用2个用户

# 公共PUCCH的资源集配置

手机idle状态时，使用SIB1当中的 PUCCH-ConfigCommon配置公共PUCCH资源集

仅用于传送单比特的HARQ

SIB1---BWP-Uplinkcommon----PUCCH configcommon

固定使用跳频方式

```
PUCCH-ConfigCommon ::= SEQUENCE {  
    pucch-ResourceCommon  
    pucch-GroupHopping  
    hoppingId  
    p0-nominal  
    ...  
}
```

pucch-ResourceCommon

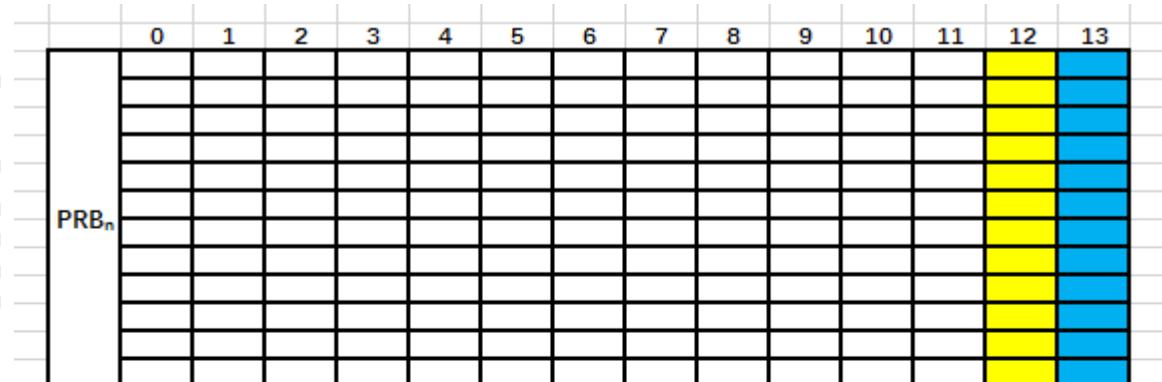
Index	PUCCH format	First symbol	Number of symbols	PRB offset $RB_{\text{BWP}}^{\text{offset}}$	Set of initial CS indexes
0	0	12	2	0	{0, 3}
1	0	12	2	0	{0, 4, 8}
2	0	12	2	3	{0, 4, 8}
3	1	10	4	0	{0, 6}
4	1	10	4	0	{0, 3, 6, 9}
5	1	10	4	2	{0, 3, 6, 9}
6	1	10	4	4	{0, 3, 6, 9}
7	1	4	10	0	{0, 6}
8	1	4	10	0	{0, 3, 6, 9}
9	1	4	10	2	{0, 3, 6, 9}
10	1	4	10	4	{0, 3, 6, 9}
11	1	0	14	0	{0, 6}
12	1	0	14	0	{0, 3, 6, 9}
13	1	0	14	2	{0, 3, 6, 9}
14	1	0	14	4	{0, 3, 6, 9}
15	1	0	14	$N_{\text{BWP}}^{\text{size}} / 4$	{0, 3, 6, 9}

循环位移  
索引

# 公共资源集举例

Table 9.2.1-1: PUCCH resource sets before dedicated PUCCH resource configuration

Index	PUCCH format	First symbol	Number of symbols	PRB offset $RB_{\text{offset}}^{\text{PUCCH}}$	Set of initial CS indexes
0	0	12	2	0	{0, 3}
1	0	12	2	0	{0, 4, 8}
2	0	12	2	3	{0, 4, 8}
3	1	10	4	0	{0, 6}
4	1	10	4	0	{0, 3, 6, 9}



假设 **pucch-ResourceCommon=3** 资源集3

这个资源集使用的是**格式1**的PUCCH

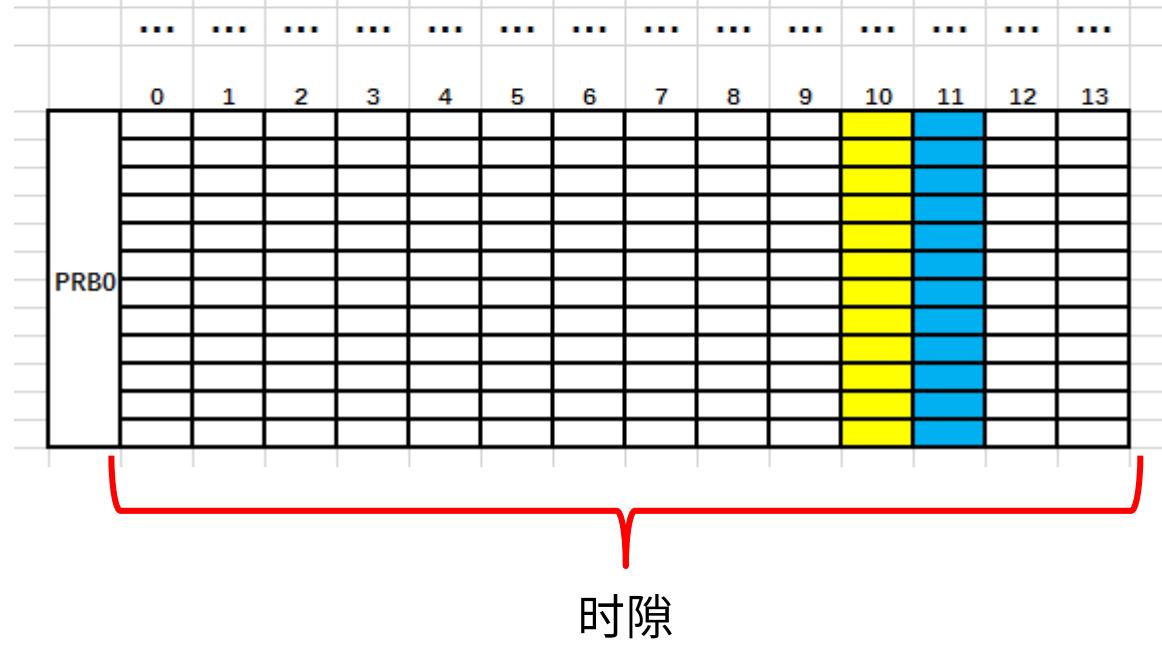
格式1的PUCCH起始符号位置是时隙当中**第10个符号**

时域**符号长度是4**

频域PRB起始位置为**PRB0**

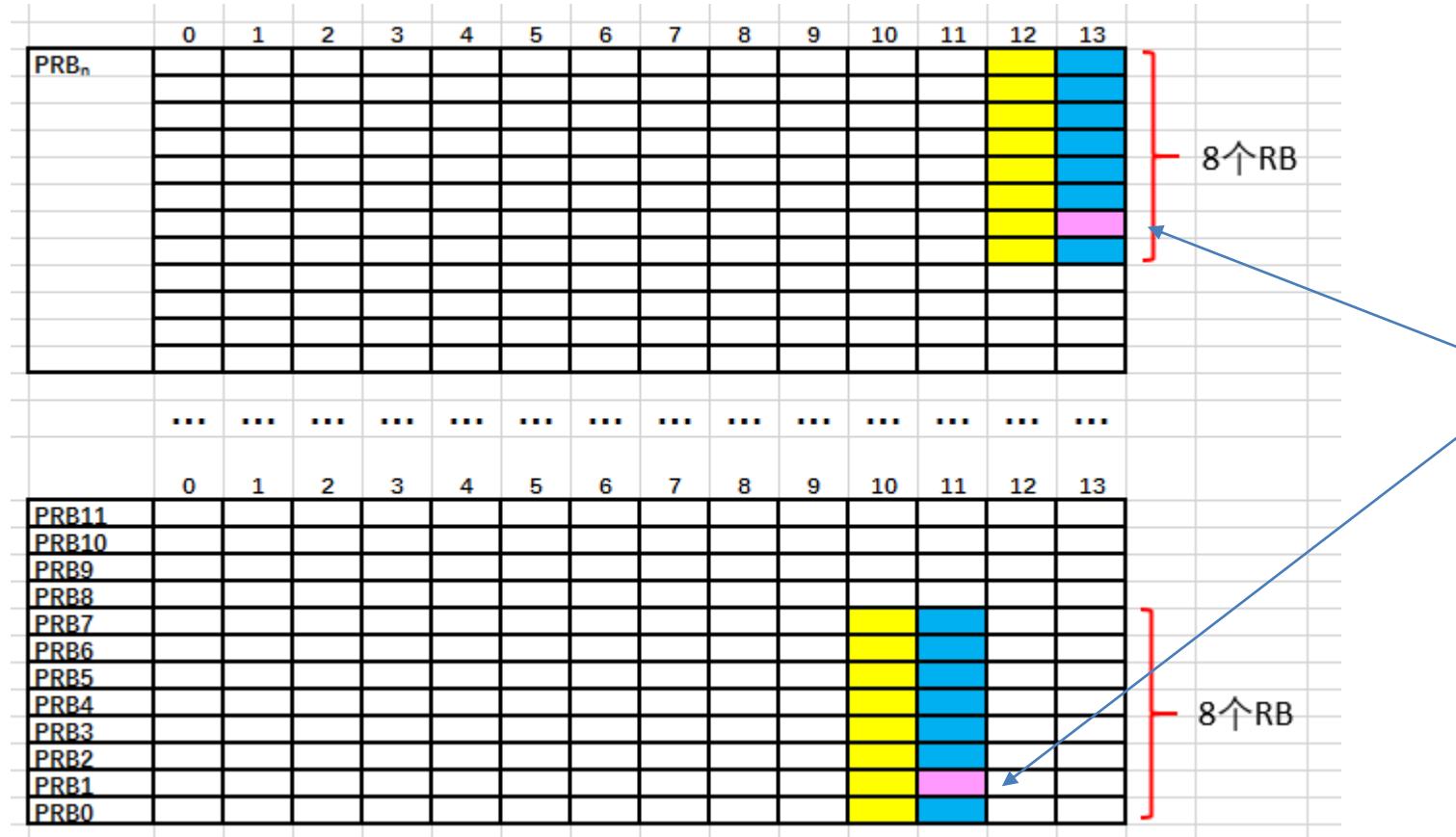
公共资源集**固定跳频**

循环位移索引: **0和6** 循环位移只有2个, 说明现在这个结构只包含了2个PUCCH资源 (复用2个用户)



# 公共资源集举例

由于公共资源集含有16个PUCCH资源（换句话说支持16个用户），因此，在上一页的基础上，在频域上依次继续取到8个RB，就可以得到16个PUCCH资源



对于某一个UE，他使用这8个RB当中的  
的某一个，然后使用0或者6的循环移位  
生成的序列发送自己的ACK-NACK信息

UE具体使用哪个RB呢？

# 公共资源集具体的PUCCH资源确定

$$r_{\text{PUCCH}} = \left\lfloor \frac{2 \cdot n_{\text{CCE},0}}{N_{\text{CCE}}} \right\rfloor + 2 \cdot \Delta_{\text{PRI}}$$

取整运算

$N_{\text{CCE}}$  为收到DCI的CORESET的CCE数量。

$n_{\text{CCE},0}$  是收到PDCCH的第一个CCE的索引，

$\Delta_{\text{PRI}}$  是format 1\_0或format 1\_1格式DCI中指示的PUCCH资源 (0-7)

$0 \leq r_{\text{PUCCH}} \leq 15$

如果  $r_{\text{PUCCH}} \leq 7$ : 则 PUCCH 传输的第一跳 PRB 索引为  $RB_{\text{BWP}}^{\text{offset}} + \lfloor r_{\text{PUCCH}} / N_{\text{cs}} \rfloor$

第二跳 PRB 索引为  $N_{\text{BWP}}^{\text{size}} - 1 - RB_{\text{BWP}}^{\text{offset}} - \lfloor r_{\text{PUCCH}} / N_{\text{cs}} \rfloor$

如果  $r_{\text{PUCCH}} > 8$ : 则 PUCCH 传输的第一跳 PRB 索引为  $N_{\text{BWP}}^{\text{size}} - 1 - RB_{\text{BWP}}^{\text{offset}} - \lfloor (r_{\text{PUCCH}} - 8) / N_{\text{cs}} \rfloor$

第二跳 PRB 索引为  $RB_{\text{BWP}}^{\text{offset}} + \lfloor (r_{\text{PUCCH}} - 8) / N_{\text{cs}} \rfloor$

$N_{\text{cs}}$  是循环位移索引个数

# 举个例子

假设

coreset里面有8个CCE， CCE编号就是0-7，

收到第一个CCE编号为3

DCI中的PUCCH resource indicator=2

第一跳PRB索引为  $RB_{\text{BWP}}^{\text{offset}} + \lfloor r_{\text{PUCCH}} / N_{\text{CS}} \rfloor$

第一跳的PRB索引为:  $0 + [4/2] = 2$

第二跳PRB索引为:  $N_{\text{BWP}}^{\text{size}} - 1 - RB_{\text{BWP}}^{\text{offset}} - \lfloor r_{\text{PUCCH}} / N_{\text{CS}} \rfloor$

第二跳PRB索引为:  $273 - 1 - 0 - 2 = 270$

$$r_{\text{PUCCH}} = \left\lfloor \frac{2 \cdot n_{\text{CCE},0}}{N_{\text{CCE}}} \right\rfloor + 2 \cdot \Delta_{\text{PRI}}$$

假设BWP PRB数273

$$r_{\text{PUCCH}} = [2 * 3 / 8] + 2 * 2 = 0 + 4 = 4$$

Table 9.2.1-1: PUCCH resource sets before dedicated PUCCH resource configuration

Index	PUCCH format	First symbol	Number of symbols	PRB offset $RB_{\text{BWP}}^{\text{offset}}$	Set of initial CS indexes
0	0	12	2	0	{0, 3}
1	0	12	2	0	{0, 4, 8}
2	0	12	2	3	{0, 4, 8}
3	1	10	4	0	{0, 6}
4	1	10	4	0	{0, 3, 6, 9}

# 例子的图

第二跳PRB索引为:270

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
PRB273														
PRB272														
PRB271														
PRB270														
PRB269														
PRB268														
PRB267														
PRB266														
PRB265														
PRB264														
PRB263														
PRB262														

8个RB

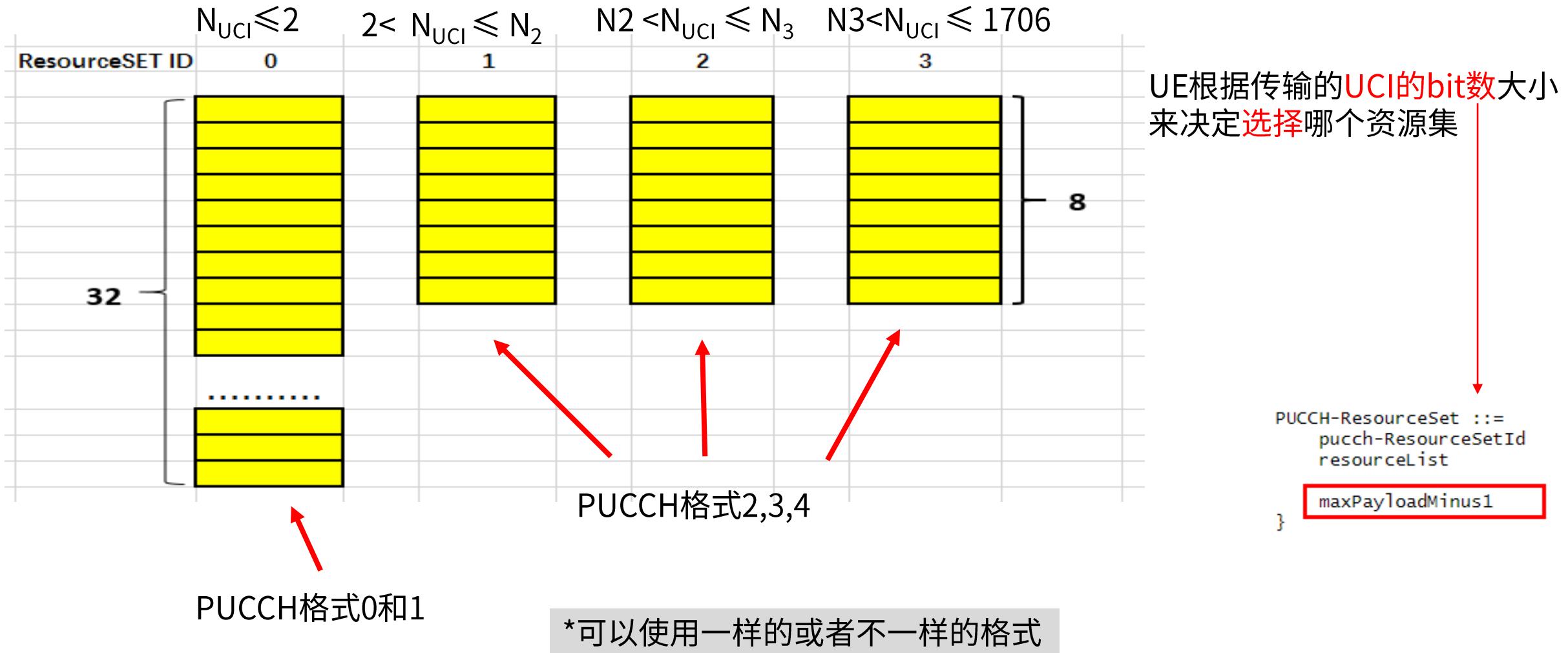
第一跳的PRB索引为：2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
PRB11														
PRB10														
PRB9														
PRB8														
PRB7														
PRB6														
PRB5														
PRB4														
PRB3														
PRB2														
PRB1														
PRB0														

8个RB

# PUCCH专用资源集配置

PUCCH专用资源集可以配置4种，每种包含一个或者多个PUCCH resource资源,对应不同PUCCH格式配置(RRC连接状态)



# PUCCH专用资源集配置

PUCCH专用资源集合由高层信令PUCCH-config所配置

信令PUCCH-Config---- PUCCH-ResourceSet

```
PUCCH-ResourceSet ::=  
  pucch-ResourceSetId  
  resourceList  
  maxPayloadMinus1  
}  
  
maxNrofPUCCH-ResourcesPerSet ::= 4  
PUCCH-ResourceSetId ::=  
  SEQUENCE {  
    pucch-ResourceSetId,  
    SEQUENCE (SIZE (8..maxNrofPUCCH-ResourcesPerSet))  
      OF PUCCH-ResourceId,  
      INTEGER (4..256) OPTIONAL -- Need R  
  }  
  INTEGER (0..maxNrofPUCCH-ResourceSets-1)
```

pucch-ResourceSetId :**资源集ID编号**, 取值 0-3 ,即 UE 最多被配置 4 个 PUCCH 资源集

resourceList: 即一个资源集中包含的 **PUCCH 资源列表** , 对应 PUCCH Resource ID

maxPayloadMinus1: UE 在 PUCCH 资源集中所能传送的 **UCI最大的信息比特**

maxNrofPUCCH-ResourcesPerSet: 每个**资源集**配置最大PUCCH**资源数目**

# PUCCH专用资源

## PUCCH Resource

```
PUCCH-Resource ::=  
  pucch-ResourceId,  
  startingPRB,  
  intraSlotFrequencyHopping,  
  secondHopPRB  
  
  format      CHOICE {  
    format0    PUCCH-format0,  
    format1    PUCCH-format1,  
    format2    PUCCH-format2,  
    format3    PUCCH-format3,  
    format4    PUCCH-format4  
  }  
  
SEQUENCE {  
  pucch-ResourceId,  
  PRB-Id,  
  ENUMERATED { enabled } OPTIONAL, -- Need R  
  PRB-Id      OPTIONAL, -- Need R
```

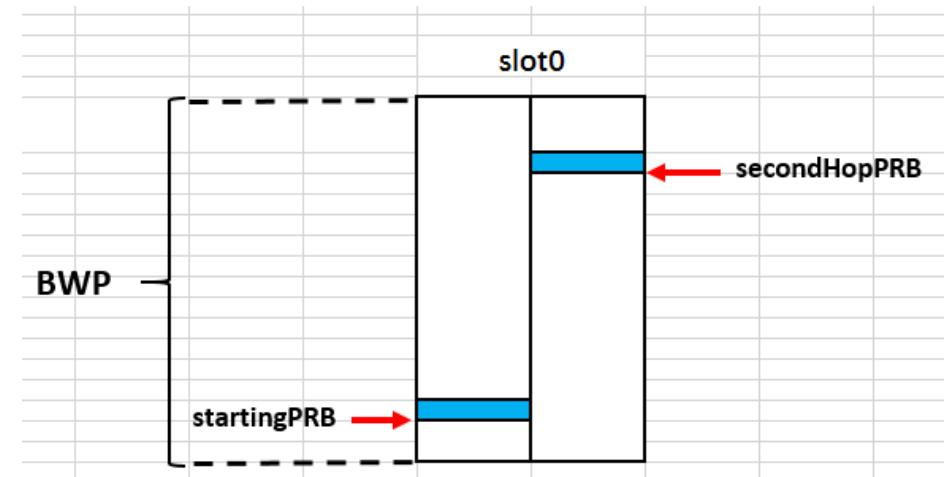
pucch-ResourceId : PUCCH资源ID

startingPRB: PUCCH起始PRB ID

intraSlotFrequencyHopping: 是否进行时隙内跳频

secondHopPRB: 跳频的PRB ID

format: PUCCH格式配置，以及不同格式时频域及相关参数



# 具体PUCCH资源的确定---第一种情况

当第一PUCCH资源集合仅配置小于等于8个PUCCH资源时，可通过DCI1-0或者DCI1-1中的PUCCH resource indicator (3bit) 字段直接指示PUCCH资源。当第一PUCCH资源集合配置超过8个PUCCH资源时，就需要使用公式确定PUCCH资源。

## Format 1\_0

This is used for the scheduling of PDSCH in one cell.

< DCI format 1\_0 with CRC scrambled by C-RNTI >

Field (Item)	Bits	Reference
Identifier for DCI formats	1	Always set to 1, meaning this is for DL
Frequency domain resource assignment	Variable	Variable with DL BWP N_RB $\lceil \log_2(N_{RB}^{DL,BWP}(N_{RB}^{DL,BWP}+1)/2) \rceil$
Time domain resource assignment	4	Carries the row index of the items in <a href="#">pdsch_allocationList in RRC</a>
VRB-to-PRB mapping	1	According to 38.212 Table 7.3.1.1.2-33 0 : Non-Interleaved 1 : Interleaved
Modulation and coding scheme	5	<a href="#">38.214 - Table 5.1.3.1-1: MCS index table 1 for PDSCH</a> <a href="#">38.214 - Table 5.1.3.1-2: MCS index table 2 for PDSCH</a>
New data indicator	1	
Redundancy version	2	
HARQ process number	4	
Downlink assignment index	2	
TPC command for scheduled PUCCH	2	
PUCCH resource indicator	3	See <a href="#">here</a> , <a href="#">here</a>
PDSCH-to-HARQ_feedback timing indicator	3	maps to k1={1,2,3,4,5,6,7,8}

## Format 1\_1

This is used for the scheduling of PDSCH in one cell.

Field (Item)	Bits	Reference
Carrier indicator	0,3	
Identifier for DCI formats	1	Always set to 1, indicating a DL DCI format
Bandwidth part indicator	0,1,2	
Frequency domain resource assignment	Variable	Variable with <a href="#">Resource Allocation Type</a>
Time domain resource assignment	4	Carries the row index of the items in <a href="#">pdsch_allocationList in RRC</a>
VRB-to-PRB mapping	0,1	0 bit if only resource allocation type 0 is configured or if interleaved VRB-to-PRB mapping is not configured by high layers; 1 bit according to Table 7.3.1.1.2-33 otherwise, only applicable to resource allocation type 1
PRB bundling size indicator	0,1	0 bit if the higher layer parameter prb-BundlingType is not configured or is set to 'static' 1 bit if the higher layer parameter prb-BundlingType is set to 'dynamic'
Rate matching indicator	0,1,2	Bit size is determined by higher layer parameters rateMatchPatternGroup1 and rateMatchPatternGroup2.
TPC command for scheduled PUCCH	2	
PUCCH resource indicator	3	See <a href="#">here</a> , <a href="#">here</a>
PDSCH-to-HARQ_feedback timing indicator	0,1,2,3	<a href="#">Row number(index) of K1</a> Number of bit is determined by $\log_2(I)$ . 'I' is the number of elements in the IE PUCCH-Config.dl-DataToUL-ACK

# 具体PUCCH资源的确定---第一种情况

DCI1-0或者DCI1-1中的PUCCH resource indicator (3bit) 字段

Table 9.2.3-2: Mapping of PUCCH resource indication field values to a PUCCH resource in a PUCCH resource set with maximum 8 PUCCH resources

PUCCH resource indicator	PUCCH resource
'000'	1 <sup>st</sup> PUCCH resource provided by <i>pucch-ResourceId</i> obtained from the 1 <sup>st</sup> value of <i>resourceList</i>
'001'	2 <sup>nd</sup> PUCCH resource provided by <i>pucch-ResourceId</i> obtained from the 2 <sup>nd</sup> value of <i>resourceList</i>
'010'	3 <sup>rd</sup> PUCCH resource provided by <i>pucch-ResourceId</i> obtained from the 3 <sup>rd</sup> value of <i>resourceList</i>
'011'	4 <sup>th</sup> PUCCH resource provided by <i>pucch-ResourceId</i> obtained from the 4 <sup>th</sup> value of <i>resourceList</i>
'100'	5 <sup>th</sup> PUCCH resource provided by <i>pucch-ResourceId</i> obtained from the 5 <sup>th</sup> value of <i>resourceList</i>
'101'	6 <sup>th</sup> PUCCH resource provided by <i>pucch-ResourceId</i> obtained from the 6 <sup>th</sup> value of <i>resourceList</i>
'110'	7 <sup>th</sup> PUCCH resource provided by <i>pucch-ResourceId</i> obtained from the 7 <sup>th</sup> value of <i>resourceList</i>
'111'	8 <sup>th</sup> PUCCH resource provided by <i>pucch-ResourceId</i> obtained from the 8 <sup>th</sup> value of <i>resourceList</i>

# 具体PUCCH资源的确定---第二种情况

当第一PUCCH资源集合配置超过8个PUCCH资源时，就需要使用公式确定PUCCH资源（编号）

$$r_{\text{PUCCH}} = \left\{ \begin{array}{ll} \left\lfloor \frac{n_{\text{CCE},p} \cdot \lceil R_{\text{PUCCH}} / 8 \rceil}{N_{\text{CCE},p}} \right\rfloor + \Delta_{\text{PRI}} \cdot \left\lfloor \frac{R_{\text{PUCCH}}}{8} \right\rfloor & \text{if } \Delta_{\text{PRI}} < R_{\text{PUCCH}} \bmod 8 \\ \left\lfloor \frac{n_{\text{CCE},p} \cdot \lceil R_{\text{PUCCH}} / 8 \rceil}{N_{\text{CCE},p}} \right\rfloor + \Delta_{\text{PRI}} \cdot \left\lfloor \frac{R_{\text{PUCCH}}}{8} \right\rfloor + R_{\text{PUCCH}} \bmod 8 & \text{if } \Delta_{\text{PRI}} \geq R_{\text{PUCCH}} \bmod 8 \end{array} \right\}$$

其中  $N_{\text{CCE}}$  是DCI调度CORESET中的CCE数量， $n_{\text{CCE}}$  是DCI第一个CCE在CORESET中的索引。 $R_{\text{PUCCH}}$  是resourceList 配置的PUCCH资源数量， $\Delta_{\text{PRI}}$  是PUCCH resource indicator的值。

这里就不详细讲了

# 专用PUCCH资源总结

PUCCH-Config ::=

PUCCH-ResourceSet ::=

pucch-ResourceSetId 1  
resourceList {

资源集ID (0-3)

PUCCH-Resource ::=

pucch-ResourceId 0  
startingPRB  
intraSlotFrequencyHopping  
secondHopPRB

资源0

频域：是否时隙内跳频，第一跳起始PRB，第二跳起始RB

format

CHOICE {  
PUCCH-format0,

PUCCH-Resource ::=  
pucch-ResourceId 1  
startingPRB  
intraSlotFrequencyHopping  
secondHopPRB

PUCCH-format0 ::=  
initialCyclicShift  
nrofSymbols  
startingSymbolIndex

时域：从哪个符号开始，持续几个符号

format

CHOICE {  
PUCCH-format0,

资源≤8

DCI1-0或者DCI1-1

PUCCH resource indicator

资源>8 公式决定选哪个资源

}  
maxPayloadMinus1

maxNrofPUCCH-ResourcesPerSet ::= 4

资源集中，资源的个数

# PUCCH的时隙

前面：

不管是公共资源集，还是专用资源集，我们发现他的详细配置中，都是在一个**时隙内PUCCH到底怎么占用资源的**

那么接下来的问题是，

到底PUCCH在哪些时隙当中呢？

# HARQ-ACK信息上报时机

## HARQ-ACK信息上报

如果UE检测到在时隙n接收PDSCH， UE在时隙(n+k)发送相应的HARQ-ACK信息

对于DCI format 1\_0， PDSCH-to-HARQ-timing-indicator字段的值映射{1,2,3,4,5,6,7,8}。

对于DCI format 1\_1， PDSCH-to-HARQ-timing-indicator字段的值根据表格TS38.213-Table-9.2.3-1映射为高层参数dl-DataToUL-ACK所指示的时隙数， dl-DataToUL-ACK参数在PUCCH-Config中配置， size从1到8。

Table 9.2.3-1: Mapping of PDSCH-to-HARQ\_feedback timing indicator field values to numbers of slots

PDSCH-to-HARQ_feedback timing indicator			Number of slots $k$
1 bit	2 bits	3 bits	
'0'	'00'	'000'	1 <sup>st</sup> value provided by dl-DataToUL-ACK
'1'	'01'	'001'	2 <sup>nd</sup> value provided by dl-DataToUL-ACK
	'10'	'010'	3 <sup>rd</sup> value provided by dl-DataToUL-ACK
	'11'	'011'	4 <sup>th</sup> value provided by dl-DataToUL-ACK
		'100'	5 <sup>th</sup> value provided by dl-DataToUL-ACK
		'101'	6 <sup>th</sup> value provided by dl-DataToUL-ACK
		'110'	7 <sup>th</sup> value provided by dl-DataToUL-ACK
		'111'	8 <sup>th</sup> value provided by dl-DataToUL-ACK

# DCI format 1\_0

DCI1-0中PDSCH-to-HARQ\_feedback timing indicator字段

## Format 1\_0

This is used for the scheduling of PDSCH in one cell.

< DCI format 1\_0 with CRC scrambled by C-RNTI >

Field (Item)	Bits	Reference
Identifier for DCI formats	1	Always set to 1, meaning this is for DL
Frequency domain resource assignment	Variable	Variable with DL <a href="#">BWP</a> <a href="#">N_RB</a> $\lceil \log_2(N_{\text{RB}}^{\text{DL,BWP}}(N_{\text{RB}}^{\text{DL,BWP}} + 1) / 2) \rceil$
Time domain resource assignment	4	Carries the row index of the items in <a href="#">pdsch_allocationList</a> in RRC
VRB-to-PRB mapping	1	According to 38.212 Table 7.3.1.1.2-33 0 : Non-Interleaved 1 : Interleaved
Modulation and coding scheme	5	<a href="#">38.214 - Table 5.1.3.1-1: MCS index table 1 for PDSCH</a> <a href="#">38.214 - Table 5.1.3.1-2: MCS index table 2 for PDSCH</a>
New data indicator	1	
Redundancy version	2	
HARQ process number	4	
Downlink assignment index	2	
TPC command for scheduled PUCCH	2	
PUCCH resource indicator	3	See <a href="#">here</a> , <a href="#">here</a>
PDSCH-to-HARQ_feedback timing indicator	3	maps to k1={1,2,3,4,5,6,7,8}

# DCI format 1\_1

在UE接入网络时，RRC信令为UE配置一个PDSCH到PUCCH时隙索引差的集合 $dl\text{-}DataToUL\text{-}ACK$ ，用于确定UE接收PDSCH的时隙索引到UE反馈PUCCH的时隙索引差

```
PUCCH-Config ::= SEQUENCE {  
    resourceSetToAddModList      SEQUENCE (SIZE (1..maxNrofPUCCH-ResourceSets)) OF  
        PUCCH-ResourceSet          OPTIONAL,   -- Need N  
    resourceSetToReleaseList      SEQUENCE (SIZE (1..maxNrofPUCCH-ResourceSets)) OF  
        PUCCH-ResourceSetId       OPTIONAL,   -- Need N  
    resourceToAddModList         SEQUENCE (SIZE (1..maxNrofPUCCH-Resources)) OF  
        PUCCH-Resource            OPTIONAL,   -- Need N  
    resourceToReleaseList        SEQUENCE (SIZE (1..maxNrofPUCCH-Resources)) OF  
        PUCCH-ResourceId         OPTIONAL,   -- Need N  
    format1                     SetupRelease { PUCCH-FormatConfig } OPTIONAL,   -- Need M  
    format2                     SetupRelease { PUCCH-FormatConfig } OPTIONAL,   -- Need M  
    format3                     SetupRelease { PUCCH-FormatConfig } OPTIONAL,   -- Need M  
    format4                     SetupRelease { PUCCH-FormatConfig } OPTIONAL,   -- Need M  
  
    schedulingRequestResourceToAddModList  SEQUENCE (SIZE (1..maxNrofFDD-Resources)) OF  
        schedulingRequestResourceToRe...  
  
    multi-CST-PUCCH-ResourceList  
    dl-DataToUL-ACK  
    spatialRelationInfoToAddModList  
    spatialRelationInfoToReleaseList
```

Table 9.2.3-1: Mapping of PDSCH-to-HARQ\_feedback timing indicator field values to numbers of slots ↵

PDSCH-to-HARQ_feedback timing indicator ↵			Number of slots $k$ ↵
1 bit ↵	2 bits ↵	3 bits ↵	↪
'0' ↵	'00' ↵	'000' ↵	1 <sup>st</sup> value provided by $dl\text{-}DataToUL\text{-}ACK$ ↵
'1' ↵	'01' ↵	'001' ↵	2 <sup>nd</sup> value provided by $dl\text{-}DataToUL\text{-}ACK$ ↵
↪	'10' ↵	'010' ↵	3 <sup>rd</sup> value provided by $dl\text{-}DataToUL\text{-}ACK$ ↵
↪	'11' ↵	'011' ↵	4 <sup>th</sup> value provided by $dl\text{-}DataToUL\text{-}ACK$ ↵
↪	↪	'100' ↵	5 <sup>th</sup> value provided by $dl\text{-}DataToUL\text{-}ACK$ ↵
↪	↪	'101' ↵	6 <sup>th</sup> value provided by $dl\text{-}DataToUL\text{-}ACK$ ↵
↪	↪	'110' ↵	7 <sup>th</sup> value provided by $dl\text{-}DataToUL\text{-}ACK$ ↵
↪	↪	'111' ↵	8 <sup>th</sup> value provided by $dl\text{-}DataToUL\text{-}ACK$ ↵

# SR调度请求信息上报时机

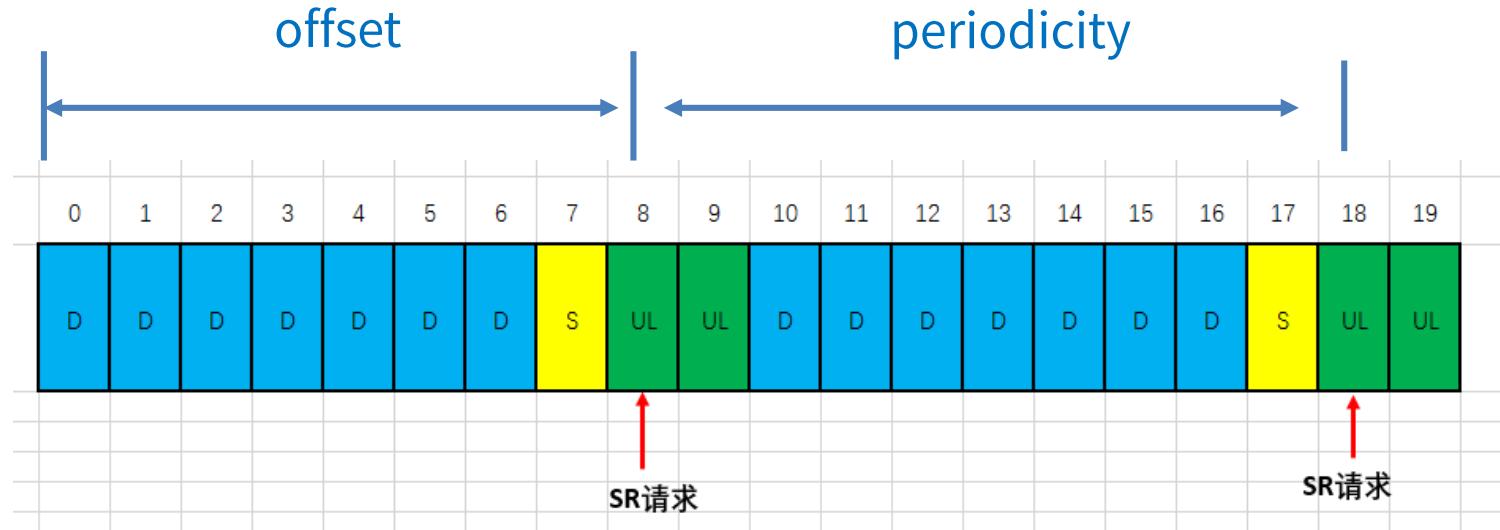
可以通过PUCCH-config中的SR资源配置信令SchedulingRequestResourceConfig中的字段配置周期  
periodicityAndOffset：确定了SR的发送时间周期periodicity和偏移量offset，其最小周期为2个符号  
周期的单位为时隙或符号，偏移的单位为时隙

```
SchedulingRequestResourceConfig ::= SEQUENCE {
    schedulingRequestResourceId           SchedulingRequestId,
    schedulingRequestID                 SchedulingRequestId,
    periodicityAndOffset              CHOICE {
        sym2                           NULL,
        sym6or7                        NULL,
        s11                            NULL,      -- Recurs in every slot
        s12                            INTEGER (0..1),
        s14                            INTEGER (0..3),
        s15                            INTEGER (0..4),
        s18                            INTEGER (0..7),
        s110                           INTEGER (0..9),
        s116                           INTEGER (0..15),
        s120                           INTEGER (0..19),
        s140                           INTEGER (0..39),
        s180                           INTEGER (0..79),
        s1160                          INTEGER (0..159),
        s1320                          INTEGER (0..319),
        s1640                          INTEGER (0..639)
    }                                OPTIONAL,   -- Need M
    resource                         PUCCH-ResourceId   OPTIONAL   -- Need M
}
```

# 举个例子

以移动8:2帧结构为例

```
schedulingRequestResourceToAddModList{  
    SchedulingRequestResourceConfig {  
        schedulingRequestResourceId 1,  
        schedulingRequestId 0,  
        periodicityAndOffset sl10: 8,  
        resource 5  
    }  
},
```



# CSI信息上报时机

CSI的上报，分为3种：周期的，半持续的和非周期

周期性上报：CSI上报的周期和slotOffset按照参数RRC信令reportSlotConfig配置的执行

半持续上报：通过MAC CE激活，并触发（或者由DCI触发）

非周期上报：通过DCI触发

这里面，我们简单说一下周期性的上报

CSI Report 类型	上报使用PUSCH Or PUCCH?
Periodic	PUCCH 2, 3, 4
SPS on PUCCH	PUCCH 2
SPS on PUSCH	PUCCH3, 4
Aperiodic	PUSCH

# CSI信息上报时机

信令配置

```

CSI-ReportConfig ::= SEQUENCE {←
    reportConfigId
    carrier
    resourcesForChannelMeasurement
    csi-IM-ResourcesForInterference
    nzp-CSI-RS-ResourcesForInterference
    reportConfigType
        periodic ← 周期性
            reportSlotConfig
            pucch-CSI-ResourceList
        },←
        semiPersistentOnPUCCH ← 半持续
            reportSlotConfig
            pucch-CSI-ResourceList
        },←
        semiPersistentOnPUSCH ← 非周期
            reportSlotConfig
            reportSlotOffsetList
            p0alpha
        },←
        aperiodic ←
            reportSlotOffsetList
        }←
    },←
    reportQuantity
        none
    CHOICE {←
        CSI-ReportConfigId,←
        ServCellIndex
        OPTIONAL, -- Need S←
        CSI-ResourceConfigId,←
        CSI-ResourceConfigId
        OPTIONAL, -- Need R←
        CSI-ResourceConfigId
        OPTIONAL, -- Need R←
    CHOICE {←
        SEQUENCE {←
            CSI-ReportPeriodicityAndOffset,←
            SEQUENCE (SIZE (1..maxNrofBWP)) OF PUCCH-CSI-Resource←
        SEQUENCE {←
            CSI-ReportPeriodicityAndOffset,←
            SEQUENCE (SIZE (1..maxNrofBWP)) OF PUCCH-CSI-Resource←
        SEQUENCE {←
            ENUMERATED {s15, s110, s120, s140, s180, s1160, s1320},←
            SEQUENCE (SIZE (1.. maxNrofUL-Allocations)) OF INTEGER(0..32),←
            P0-PUSCH-AlphaSetId←
        SEQUENCE {←
            SEQUENCE (SIZE (1..maxNrofUL-Allocations)) OF INTEGER(0..32)←
    NULL,←
}

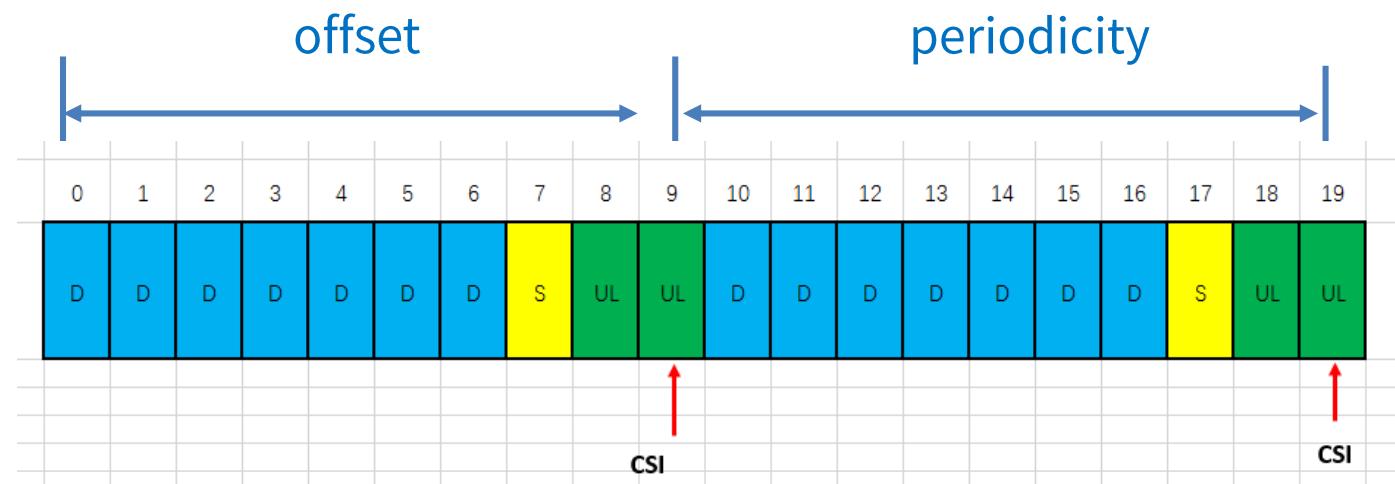
```

# CSI信息上报时机

```
CSI-ReportPeriodicityAndOffset ::= CHOICE {  
    slots4          INTEGER(0..3),  
    slots5          INTEGER(0..4),  
    slots8          INTEGER(0..7),  
    slots10         INTEGER(0..9),  
    slots16         INTEGER(0..15),  
    slots20         INTEGER(0..19),  
    slots40         INTEGER(0..39),  
    slots80         INTEGER(0..79),  
    slots160        INTEGER(0..159),  
    slots320        INTEGER(0..319)  
}  
}
```

假设：

reportSlotConfig:  
SI10:9



感谢观看

