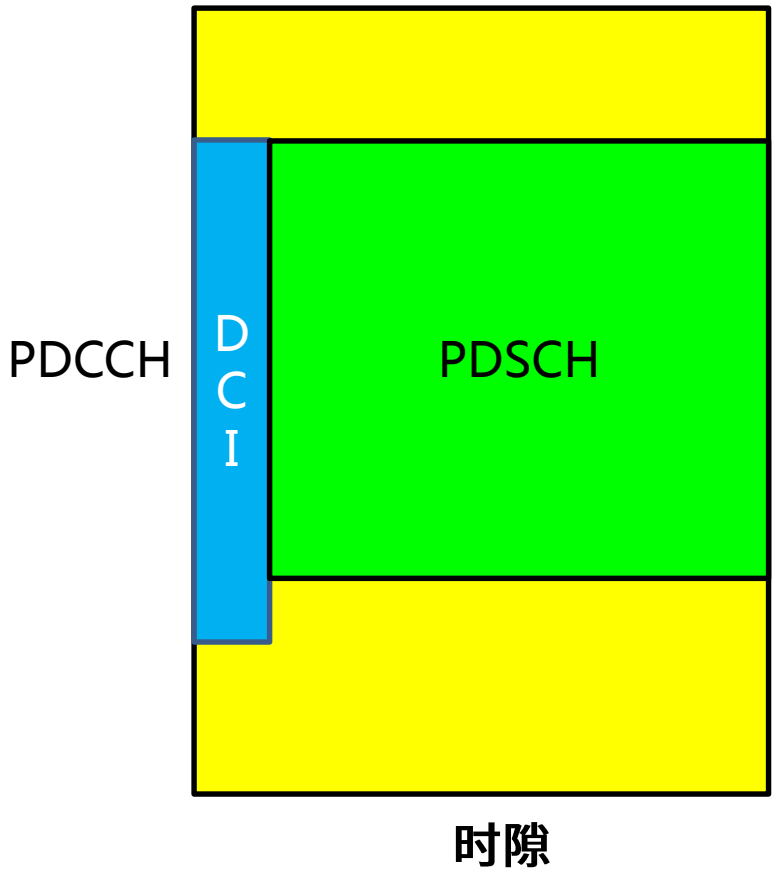
5G PDCCH

讲师：捻叶成剑

PDCCH

PDCCH: Physical Downlink Control Channel, 物理下行控制信道

PDCCH作用就是承载DCI信息（Downlink control information下行控制信息）



DCI指示了该用户的PDSCH或者PUSCH的调度信息，
只有解码了DCI，用户才可以解码自己的PDSCH或者PUSCH

DCI类似于电影院门票，有了票，你才知道自己的座位
PDSCH或者PUSCH在哪里。

一个用户占用一个PDCCH

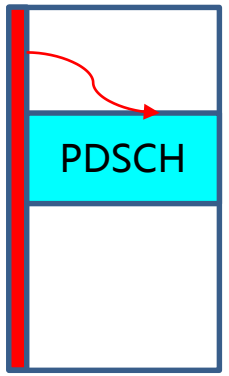
彻底理解PDCCH的作用

举个例子

去哪找？→PDCCH在帧结构里面的位置在哪？

UE想收取paging消息 → UE去找到 P-RNTI CRC+PDCCH （含有DCI） → UE使用P-RNTI去解码 P-RNTI CRC+PDCCH （含有DCI）

UE得到DCI → DCI指示PDSCH的位置 → UE解码PDSCH → UE得到paging消息



RNTI：无线网络临时标识符

CRC循环冗余检验

- 5G NR使用24位的CRC校验位
- 校验位交织在信息内

CRC基本原理

举例子：

原始数据M：1010111　　M后面补N个0　　收发双方都知道的除数P：110

```
        11001 01
110 ) 1010111 00
      110
      ---
       110
       110
       ---
        111
        110
        ---
         100
         110
         ---
          10   <-- 校验位
```

发送的数据：10101111**0**

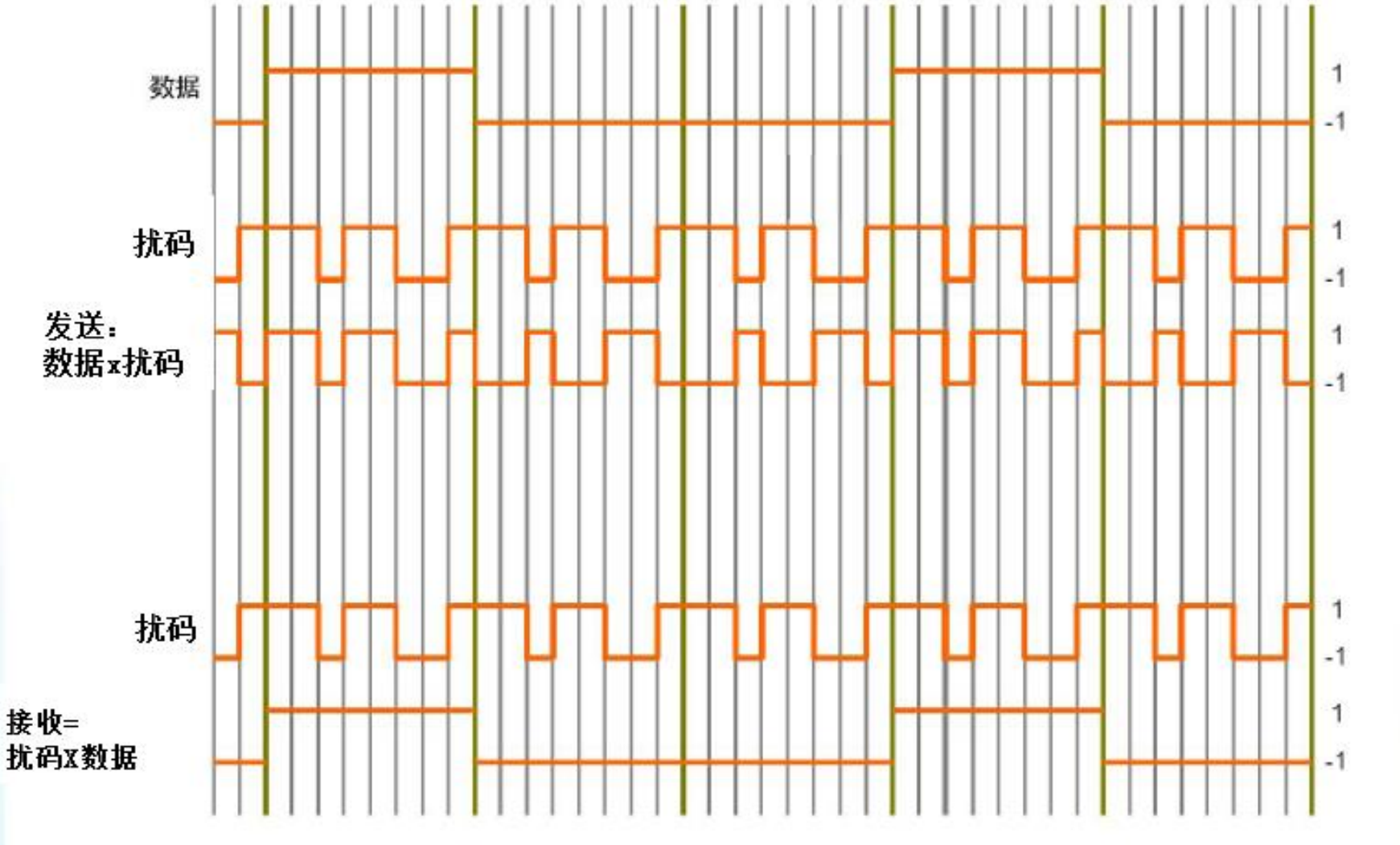
CRC校验位

接收端：

```
110 ) 101011110
      110
      ---
       110
       110
       ---
        111
        110
        ---
         110
         110
         ---
           0
```

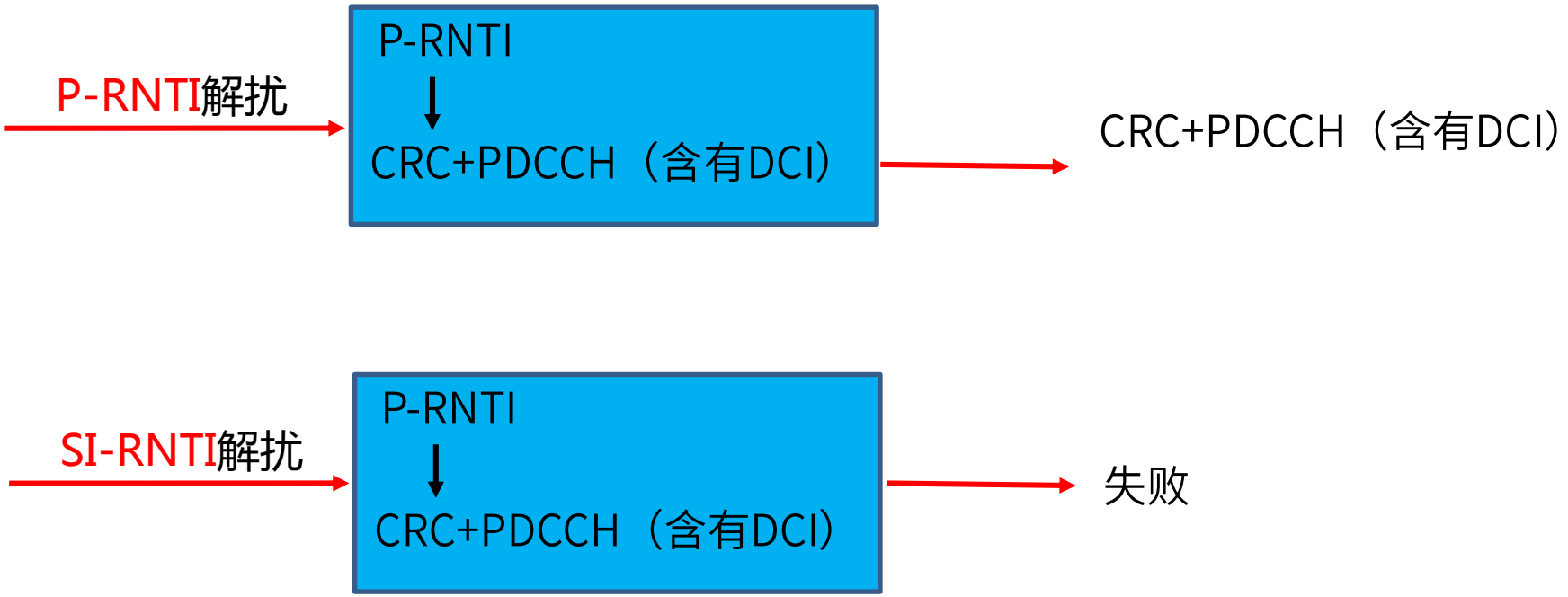
0说明数据无误

加扰



RNTI的加扰

RNTI对PDCCH的CRC进行加扰



消息与RNTI

UE需要什么信息，就用什么RNTI去解扰PDCCH

加扰的RNTI类型	内容
C-RNTI、CS-RNTI、MCS-C-RNTI、TC-RNTI	UE调度/RA
C-RNTI、CS-RNTI、MCS-C-RNTI、SP-CSI-RNTI	UE调度
SI-RNTI	SIB1\OSI
RA-RNTI	RA
TC-RNTI	
C-RNTI	UE调度
P-RNTI	paging
C-RNTI、CS-RNTI、MCS-C-RNTI	UE调度
SFI-RNTI	时隙格式
INT-RNTI	UE通知不可用的PRB和OFDM符号
TPC-PUSCH-RNTI、TPC-PUCCH-RNTI	功率控制
TPC-SRS-RNTI	功率控制

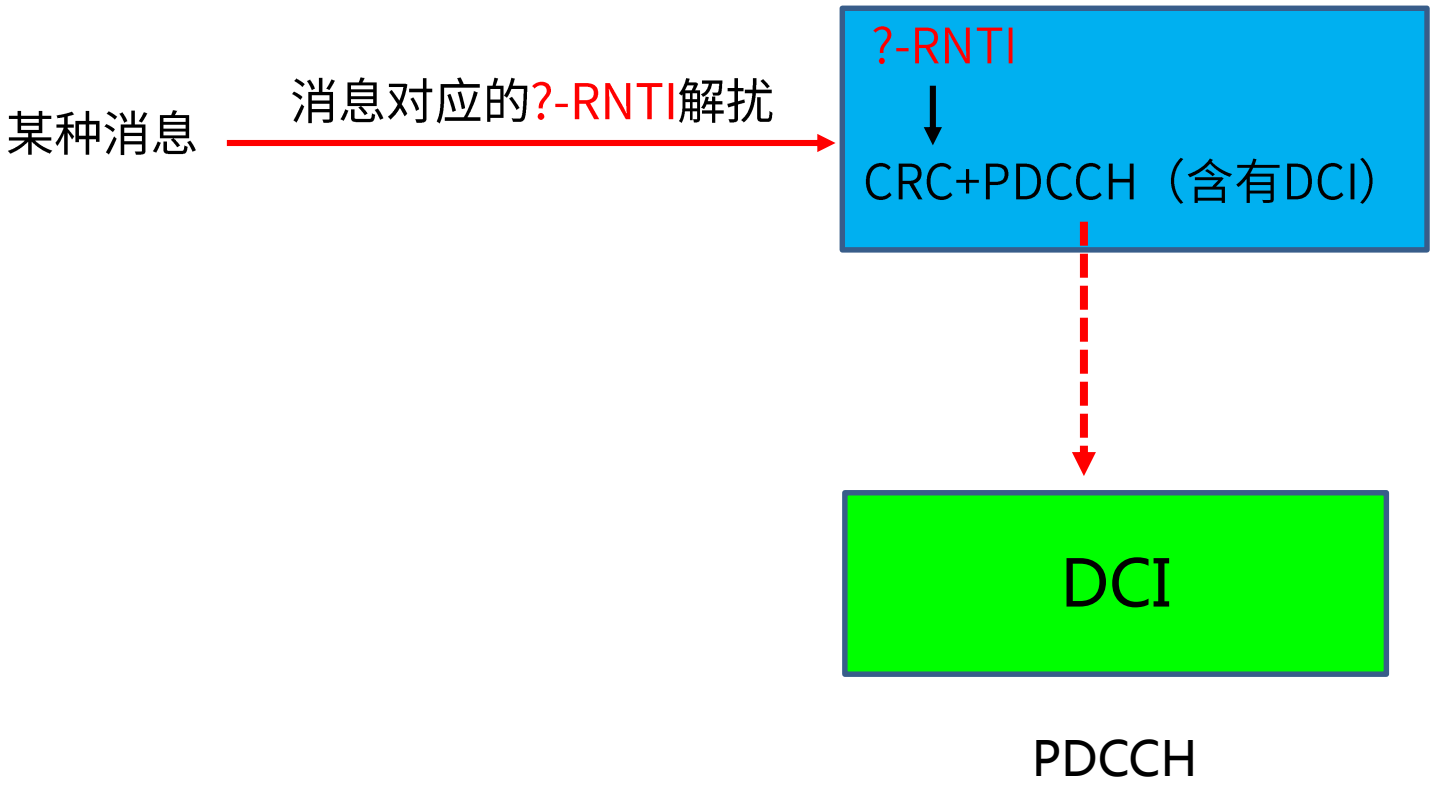
CS-RNTI(Configured Scheduling RNTI)：半静态调度，一次设定后，使用固定周期来调度数据

MCS-C-RNTI：，使用 MCS-C-RNTI 解扰 PDCCH，根据 CRC 结果决定 PUSCH/PDSCH使用的 MCS 表格

SP-CSI-RNTI：用于指示半静态CSI 在 PUSCH 的上报（CSI：信道状态信息，含CQI，PMI，rank，layer 等）

DCI

UE通过消息对应的RNTI去解扰PDCCH，来获取DCI，**是否所有的消息，使用的DCI都是同一种格式?**



DCI格式与RNTI

不同种类的消息，使用的DCI格式会有一定的区别

某种消息 —?-RNTI解扰→ ?-RNTI ↓ CRC+PDCCH（含有DCI）

R15版本定义了8种类型的DCI （*R16版本中，将DCI类型扩充至15种）

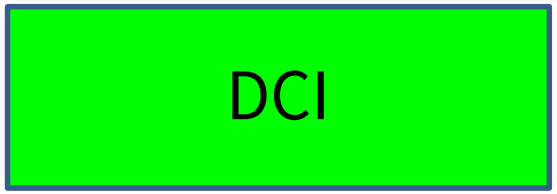
DCI格式	加扰的RNTI类型	内容
DCI0-0	C-RNTI、CS-RNTI、MCS-C-RNTI、TC-RNTI	UE调度/RA
DCI0-1	C-RNTI、CS-RNTI、MCS-C-RNTI、SP-CSI-RNTI	UE调度
DCI1-0	SI-RNTI	SIB1\OSI
	RA-RNTI	RA
	TC-RNTI	
	C-RNTI	UE调度
	P-RNTI	paging
DCI1-1	C-RNTI、CS-RNTI、MCS-C-RNTI	UE调度
DCI2-0	SFI-RNTI	时隙格式
DCI2-1	INT-RNTI	UE通知不可用的PRB和OFDM符号
DCI2-2	TPC-PUSCH-RNTI、TPC-PUCCH-RNTI	功率控制
DCI2-3	TPC-SRS-RNTI	功率控制

箱子的大小

PDCCH里面装了DCI，接下来好好了解一下这个PDCCH这个箱子了。

这个箱子有多大？------占用系统多少资源（符号，子载波）

一个PDCCH占用的资源大小，是以CCE为单位的，因为RB小，只有12个RE，
并且PDCCH也不一定只占用一个CCE，可以占用多个CCE传输。接下来我们看看这个问题



PDCCH

CCE控制信道单元

CCE是PDCCH传输的最小资源单位，一个PDCCH可以包含一个或多个CCE

- 一个CCE包含6个REG（资源粒子组）
- 一个REG由频域上一个RB，时域上一个符号组成 ⟶ 实际上一个REG就是一个RB
- REG bundle（REG束）：L个REG组成，L取值：2,3,6，当L=6时，一个REG bundle同时也就是一个CCE

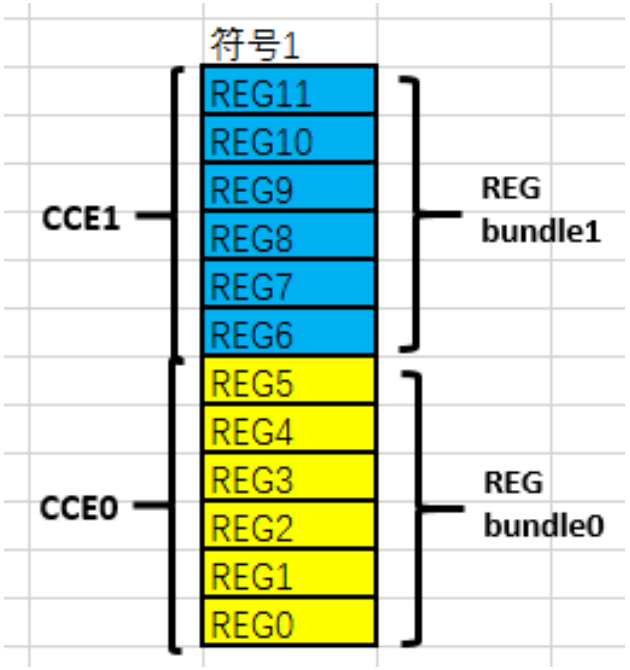
- 一个CCE可以由REG交织或者非交织而成，在CORESET中REG以时域优先的方式按递增顺序编号

CCE-to-REG映射

CCE-to-REG分为交织映射和非交织映射，在CORESET中REG以时域优先的方式按递增顺序编号，起始编号为0

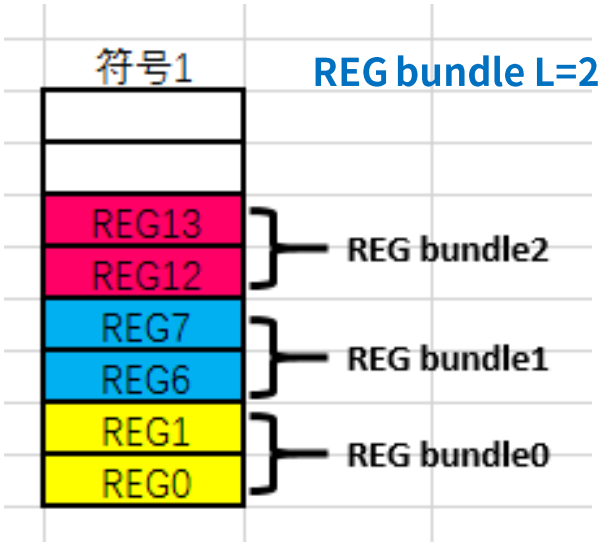
非交织映射：3种情况

REG bundle L=6，与此同时， REG bundle 的编号与CCE的编号是相同的

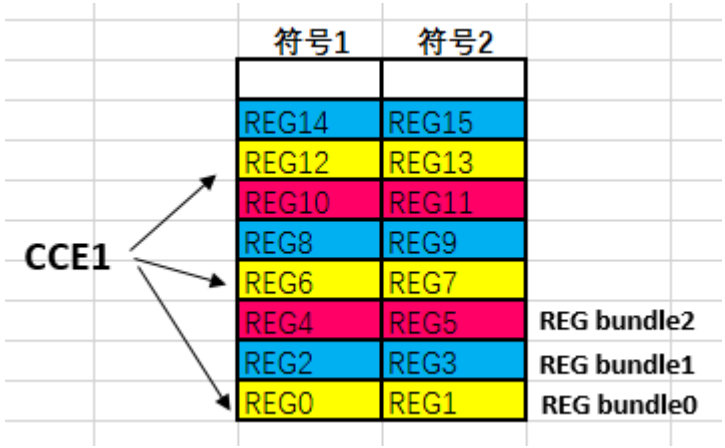


CCE-to-REG映射

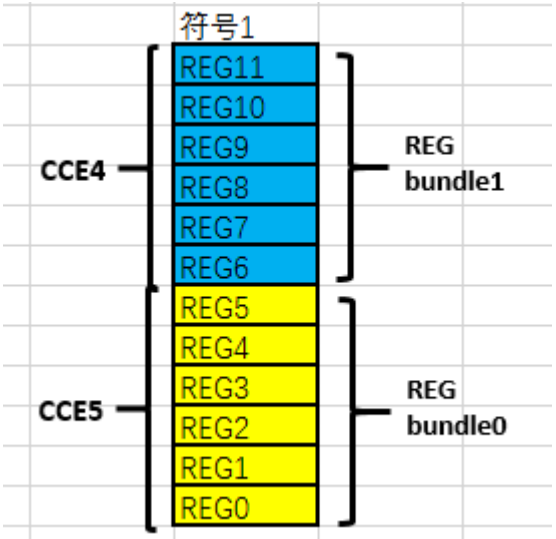
交织映射6种情况



REG bundle L=2



REG bundle L=3



REG bundle L=6， REG bundle 的编号与CCE的编号不同

CCE-to-REG交织算法

使用的是矩形交织器，主要有三个参数，一个是行数R，一个是列数C，循环位移数：nshift

R取值：2,3,6

列数C=总REG数/R*L　L是REG bundle 大小（2,3,6），C必须是整数
矩形交织的总逻辑就是：“行进列出”

这个区域后面我们叫做coreset

举个例子

假设有一个区域有24个REG
REG bundle：L=3，也就是3个REG捆绑一起
假设交织的行R取值2，这样C=24/(3*2)=24/6=4
这样就得到了一个2*4的交织器,经过交织，就得到了CCE应该对应的REG bundle的编号

当考虑到nshif的时候，CCE1的取值就可能从（0,4）变成（1,5）了

C=4

REG bundle0	REG bundle1	REG bundle2	REG bundle3
REG bundle4	REG bundle5	REG bundle6	REG bundle7

R=2

CCE1

REG21	REG22	REG23	REG bundle7
REG18	REG19	REG20	REG bundle6
REG15	REG16	REG17	REG bundle5
REG12	REG13	REG14	REG bundle4
REG9	REG10	REG11	REG bundle3
REG6	REG7	REG8	REG bundle2
REG3	REG4	REG5	REG bundle1
REG0	REG1	REG2	REG bundle0

CCE聚合等级

与PBCH不同，PBCH的大小，固定48个RB，而一个PDCCH占用资源大小，由CCE的聚合等级来决定。也就是一个PDCCH可以占，1、2、4、8、16个CCE。一个CCE6个RB，这样，最多一个PDCCH可以占用96个RB。

一个PDCCH占用的CCE越多，抗干扰能力越强。现网可以设置自适应或者固定某一种聚合方式。

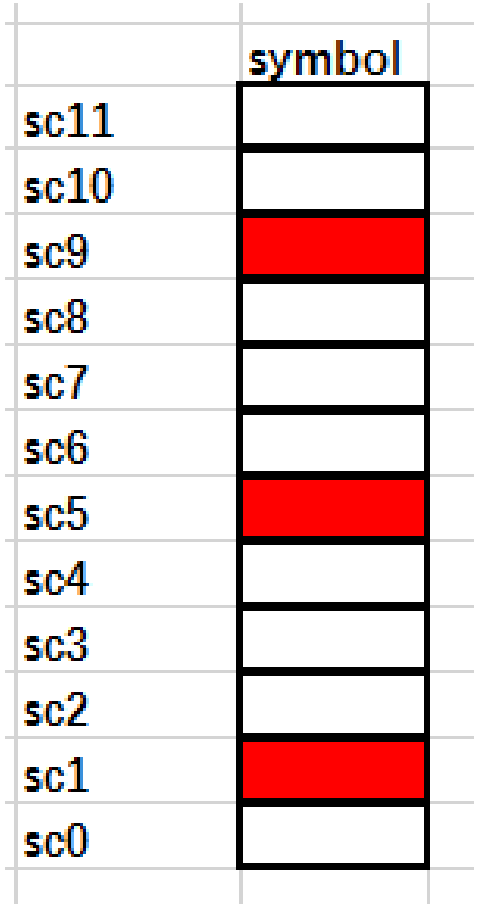
使用自适应方式进行聚合，对于用户来说，信号质量好的用户，分配的CCE就会少，而信号质量差的用户，就会分配更多的CCE来抵抗干扰。

Aggregation level	Number of CCEs
1	1
2	2
4	4
8	8
16	16

PDCCH里面的DMRS

一个REG中并不是每一个符号都用来传PDCCH，
其中也包含了DMRS解调参考信号

PDCCH里每个REG（RB）里面有3个的DMRS，
位置是固定：1、5、9如右图所示.



REG（RB）

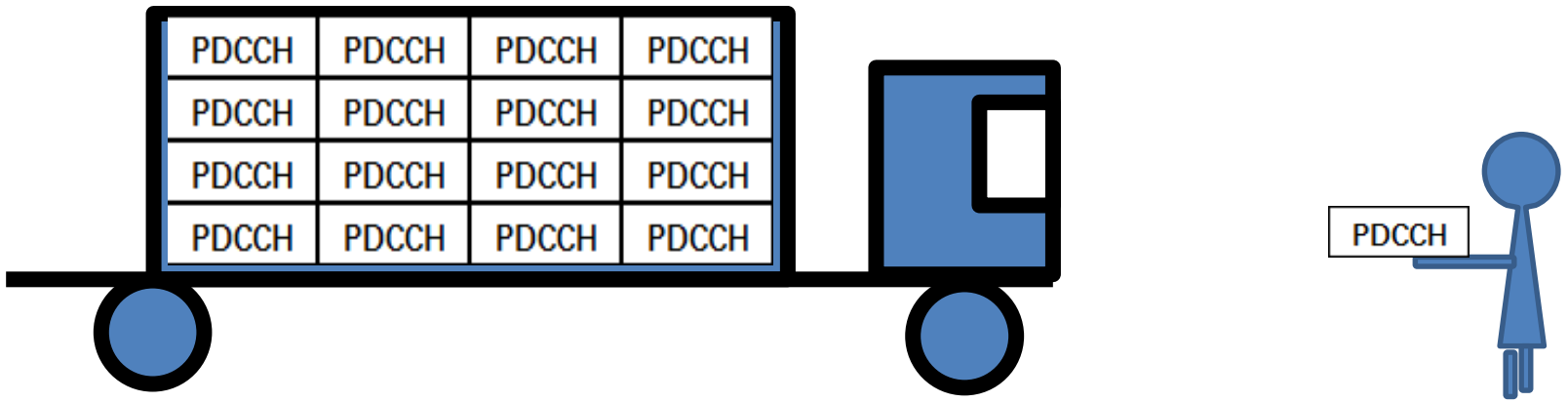
PDCCH的集合

我们深入讨论了一个PDCCH的大小，接下来，系统中定义了PDCCH的集合，也就是，每一个用户，想找到自己的PDCCH，必须在这个集合当中去寻找。

这个集合定义为：

CORESET控制资源集（COntrol REsource SET）

一个coreset当中会有多个PDCCH，UE需要在coreset这个资源池当中寻找自己的PDCCH。

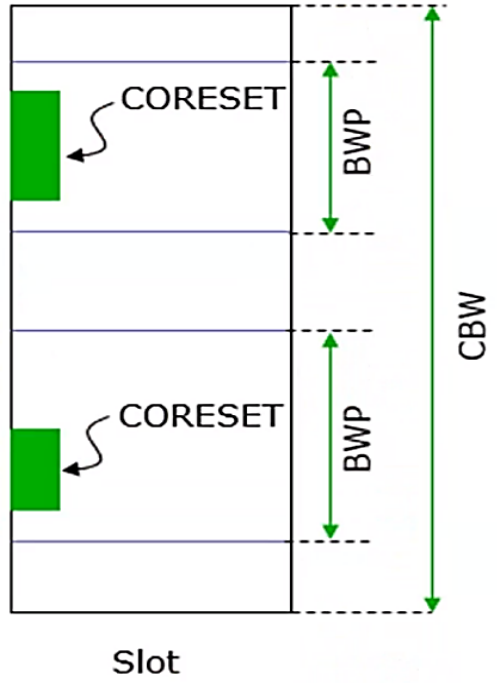


CORESET控制资源集

由于5G存在BWP的概念，因此，对于BWP来说，PDCCH的集合coreset存在于每一个BWP当中

时域上可以出现在slot的任何位置，频域上处在BWP之内

UE每个BWP最多可以配置3个CORESET, 每个小区一共可配置12个CORESETs(4个BWP*3), CORESET index范围0~11.



BWP

Initial BWP（初始BWP）：UE处于空闲态的BWP

First Active BWP（初次激活BWP）：UE第一个专有的BWP

Default BWP（默认BWP）：UE专有BWP，如果没有配置，则将Initial BWP认为是default BWP

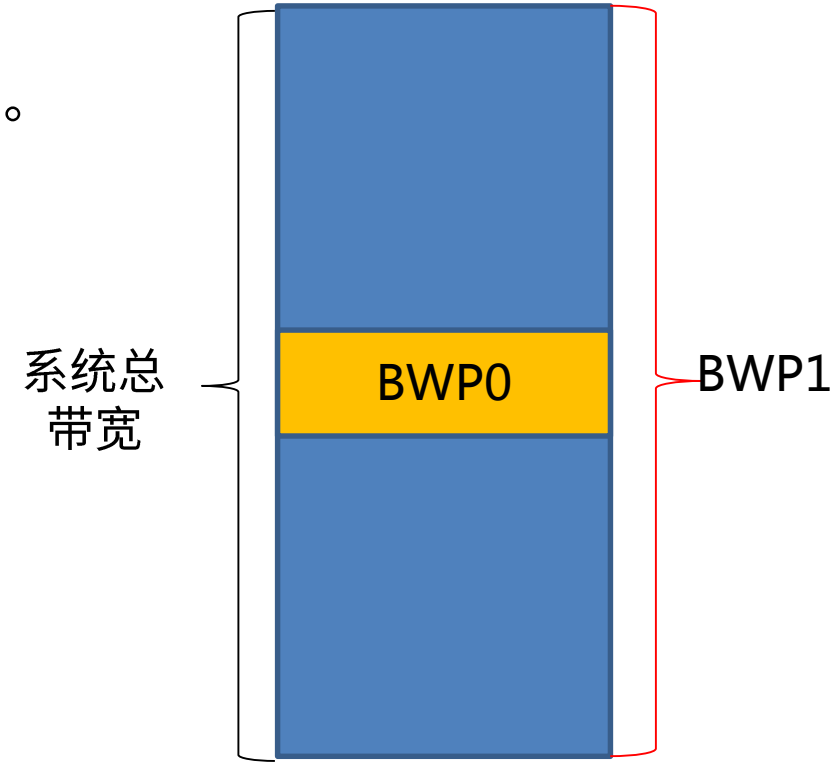
现网总共最多可以配置5个BWP：一个Initial BWP，和4个Default BWP
Initial BWP一个，default BWP最多4个，first BWP 包含在default当中。

现网举例（华为）：现网定义了2个BWP:

BWP ID=0：定义的Initial BWP是48RB，

BWP ID=1：定义的First Active BWP/Default BWP:273RB。

也就是空闲态，UE只解码48个RB的带宽，以收取自己需要的信息，
而业务态，UE可以占用100M带宽进行
数据传输。



CORESET控制资源集

CORESET在频域上有 $N_{RB}^{CORESET}$ 个RB，在时域上有 $N_{symb}^{CORESET} \epsilon \{1,2,3\}$ 个symbol组成

RRC层ControlResourceSet IE中给出了具体配置

```
ControlResourceSet ::=              SEQUENCE {
   controlResourceSetId                 ControlResourceSetId,

   frequencyDomainResources                 BIT STRING (SIZE (45)),
   duration                          INTEGER
(1..maxCoReSetDuration),
   cce-REG-MappingType                  CHOICE {
      interleaved                    SEQUENCE {
         reg-BundleSize                   ENUMERATED {n2, n3, n6},
         interleaverSize                  ENUMERATED {n2, n3, n6},
         shiftIndex
INTEGER(0..maxNrofPhysicalResourceBlocks-1)
OPTIONAL   -- Need S
      },
      nonInterleaved                NULL
```

Coreset 编号（0-11）

frequencyDomainResources：CORESET频域资源（一个bit对应6个RB）

duration：CORESET连续符号数（取值，1,2,3）

REG束的大小L值（2,3,6）

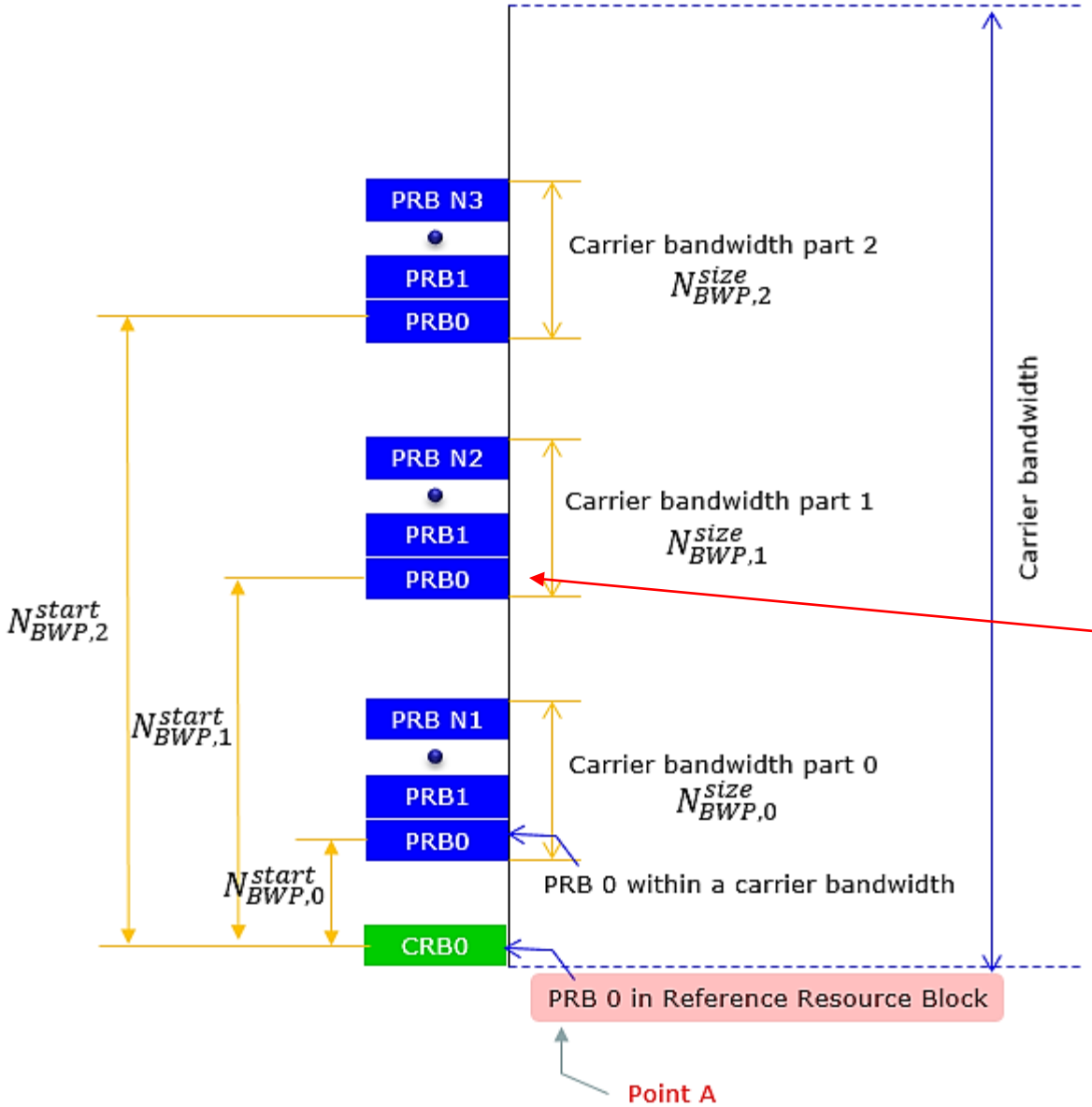
REG到CCE的映射关系

交织器的行数R（2,3,6）

bit的分布，决定了coreset在BWP中的位置，从PRB0开始

```
{
  controlResourceSetId 1,   For CORESET ID = 1
  frequencyDomainResources '11111111 00000000 00000000 00000000 00000000 00000'B,
  duration 1,
```

复习一下CRB与PRB



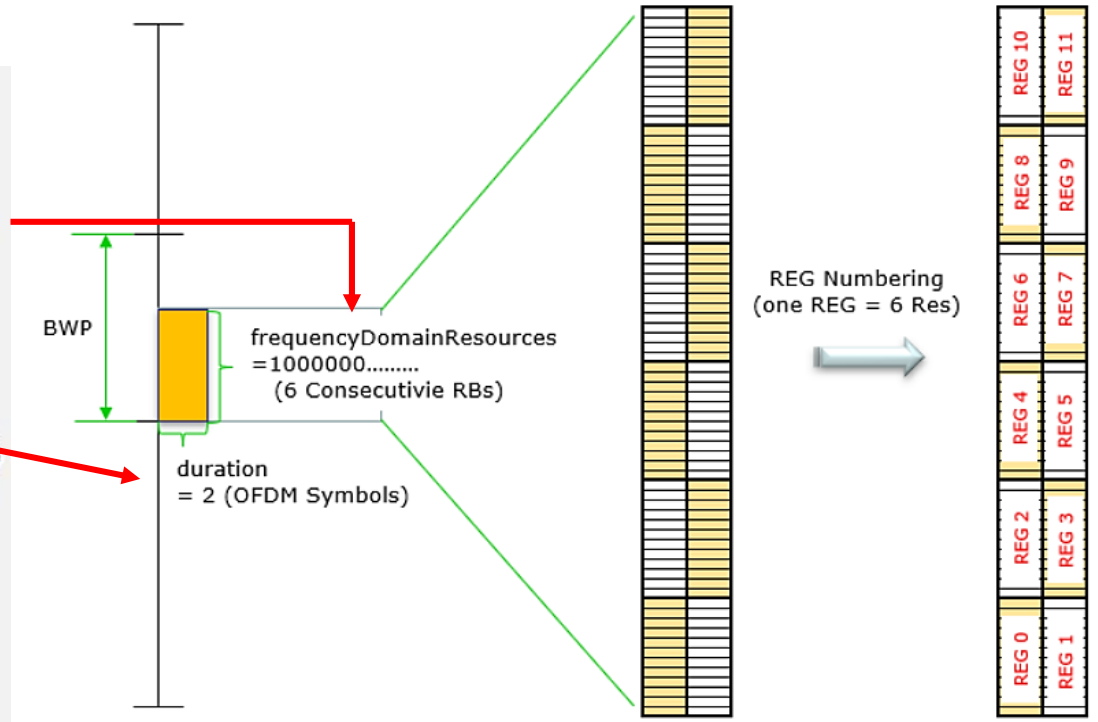
CRB：公共资源块
PRB：部分带宽（BWP）里面的资源块

PRB0就是每一个BWP的第一个RB

CORESET例子

举个例子

```
ControlResourceSet ::=              SEQUENCE {
   controlResourceSetId                 ControlResourceSetId,

   frequencyDomainResources             BIT STRING (SIZE (45)),
   duration                         INTEGER
(1..maxCoReSetDuration),
   cce-REG-MappingType                  CHOICE {
      interleaved                   SEQUENCE {
         reg-BundleSize                 ENUMERATED {n2, n3, n6},
         interleaverSize                ENUMERATED {n2, n3, n6},
         shiftIndex
INTEGER(0..maxNrofPhysicalResourceBlocks-1)
OPTIONAL   -- Need S
      },
      nonInterleaved                NULL
```



L	6
Aggregation Level	1

==> reg-BundleSize = n6

REG	0	1	2	3	4	5	6	7	8	9	10	11
	i L	*i L+1*	*i L+2*	*i L+3*	*i L+4*	*i L+L-1*	*i L*	*i L+1*	*i L+2*	*i L+3*	*i L+4*	*i L+L-1*
	*0*6*	*0*6+1*	*0*6+2*	*0*6+3*	*0*6+4*	*0*6+(6-1)*	*1*6*	*1*6+1*	*1*6+2*	*1*6+3*	*1*6+4*	*1*6+(6-1)*
REG Bundle	0 *(i = 0)*						1 *(i = 1)*					
	f(j) = j --> f(0) = 0, where j = 0						f(j) = j --> f(1) = 1, where j = 1					
CCE	0						1					

蛋生鸡悖论

从之前的coreset定义中，我们得到一个“鸡生蛋和蛋生鸡” 的悖论

RRC指示了coreset的设定，之后UE通过coreset里面去寻找PDCCH，然后又找到PDSCH
而PDSCH里面才含有RRC命令。也就是如果解码RRC消息，必须先解码PDSCH，而解码PDSCH，又必须先知道RRC消息才有后面的解码PDSCH。


```
MIB ::= SEQUENCE {
    systemFrameNumber        6bit      系统帧号SFN高6位比特位，1024为
    subCarrierSpacingCommon  1bit      子载波宽度
    ssb-SubcarrierOffset     4it       Kssb：频域间隔（低4位），单位:
    dmrs-TypeA-Position      1bit      用于表示第一个PDSCH的DMRS符号的时
    pdcch-ConfigSIB1         8bit      PDCCH信道配置
    cellBarred               1bit      小区驻留状态：禁止驻留情况下，手机可以切换
    intraFreqReselection     1bit      小区是否允许同频重选
    spare                    1bit      空闲（备用）
}
```

为了解决这个问题，系统中定义了Coreset0这个资源集，由MIB来直接获取coreset0的设定，
不需要RRC消息，因此，UE可以先通过coreset0，来获取PDCCH，进而获取PDSCH，进而获取RRC命令（SIB1），
SIB1定义了各种消息如何在coreset0里面搜索，因此接下来，就是UE在coreset0当中解码，找到PDCCH，进而解码PDSCH，获取自己的消息了

coreset0

UE待机状态

MIB-coreset0-PDCCH-PDSCH-RRC（sib1）

被叫

UE想要paging->coreset0->PDCCH->PDSCH->得到paging

UE想要RA->coreset0->PDCCH->PUSCH、PDSCH->RA

UE想要OSI->coreset0->PDCCH->PDSCH->OSI

主叫

SIB1

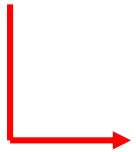
```
monitoringSymbolsWithinSlot: 8000 [bit length 14, 2 LSB pad bits,
nrofCandidates
    aggregationLevel1: n0 (0)
    aggregationLevel2: n0 (0)
    aggregationLevel4: n4 (4)
    aggregationLevel8: n2 (2)
    aggregationLevel16: n1 (1)
searchSpaceType: common (0)
    common
        dci-Format0-0-AndFormat1-0
searchSpaceSIB1: 0
pagingSearchSpace: 1
ra-SearchSpace: 1
```

定义了paging和ra的搜索方法

通过SIB1里面的配置，UE可以进行RA/paging，从而进入到连接态，进入连接态的过程中的RRC信令，会配置别的coreset，比如coreset1。这样，第二个coreset就配置上了.

别的coreset

MIB-coreset0-PDCCH-PDSCH-RRC（sib1）

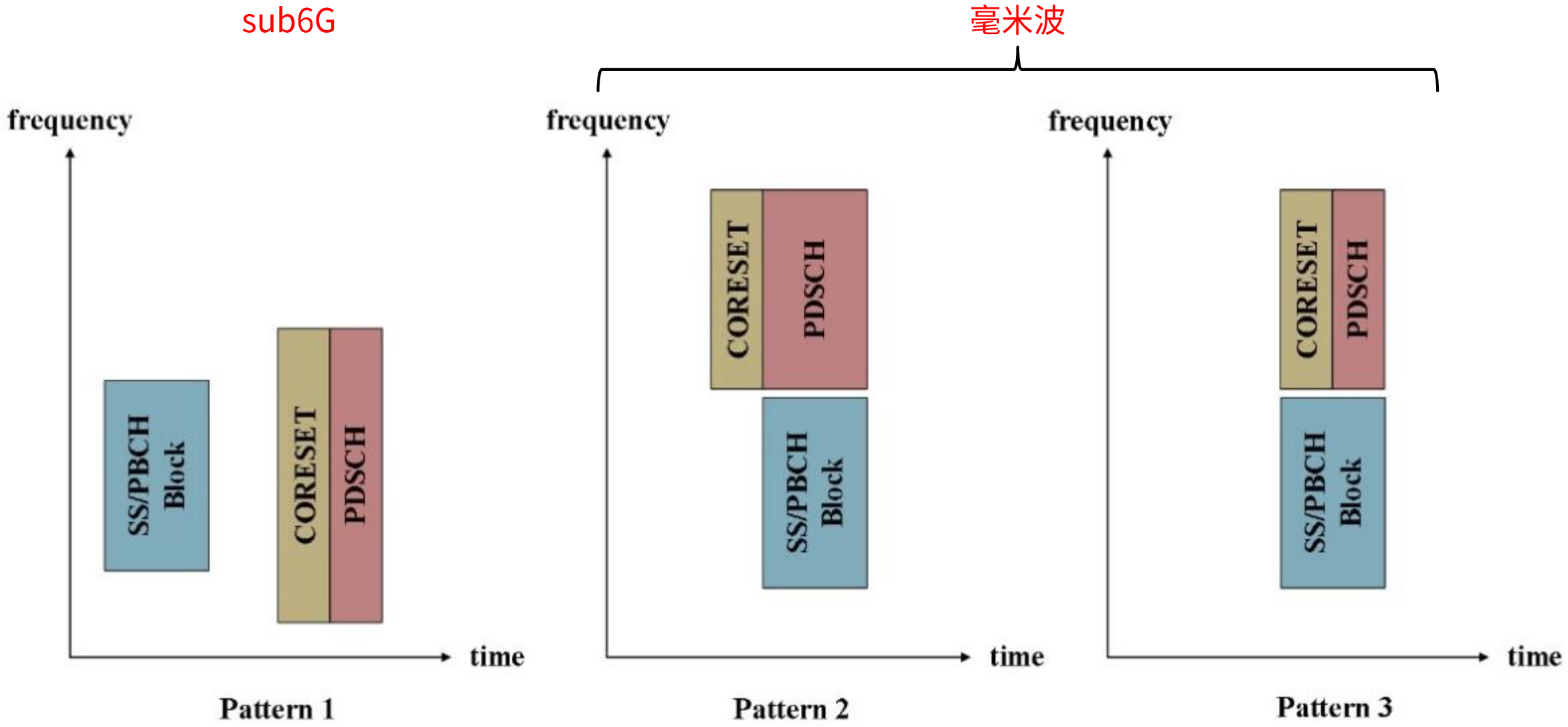


UE想要RA->coreset0->PDCCH->PUSCH、PDSCH->RA

用户数据<- PDSCH<- PDCCH<-coreset1<-过程RRC信令

CORESET0分布样式

Coreset0的分布于SSB相关，这样一种设置，避免了SSB与coreset0 的冲突，总体分为3种样式



MIB

Coreset0的设置，由MIB承载

- mib
  - systemFrameNumber = 100011
  - subCarrierSpacingCommon = scs30or120
  - ssb-SubcarrierOffset = 6
  - dmrs-TypeA-Position = pos2
  - pdcch-ConfigSIB1
    - controlResourceSetZero = 10
    - searchSpaceZero = 4
  - cellBarred = notBarred
  - intraFreqReselection = allowed
  - spare = 1

Index	SS/PBCH block and CORESET multiplexing pattern	Number of RBs $N_{RB}^{CORESET}$	Number of Symbols $N_{symb}^{CORESET}$	Offset (RBs)
0	1	24	2	0
1	1	24	2	1
2	1	24	2	2
3	1	24	2	3
4	1	24	2	4
5	1	24	3	0
6	1	24	3	1
7	1	24	3	2
8	1	24	3	3
9	1	24	3	4
10	1	48	1	12
11	1	48	1	14
12	1	48	1	16
13	1	48	2	12
14	1	48	2	14
15	1	48	2	16

前4bit，解码出coreset 0占用的资源大小

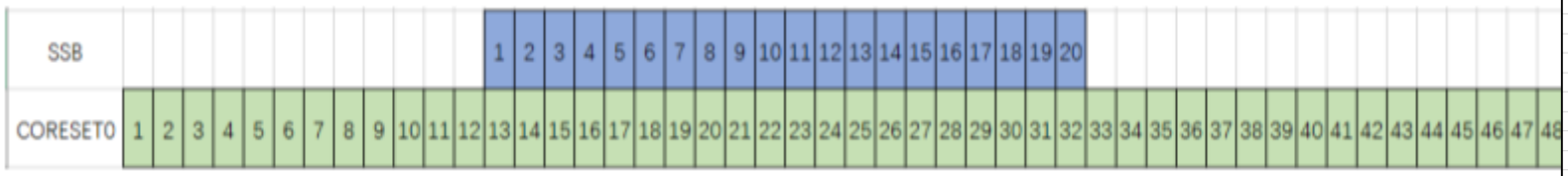
MIB

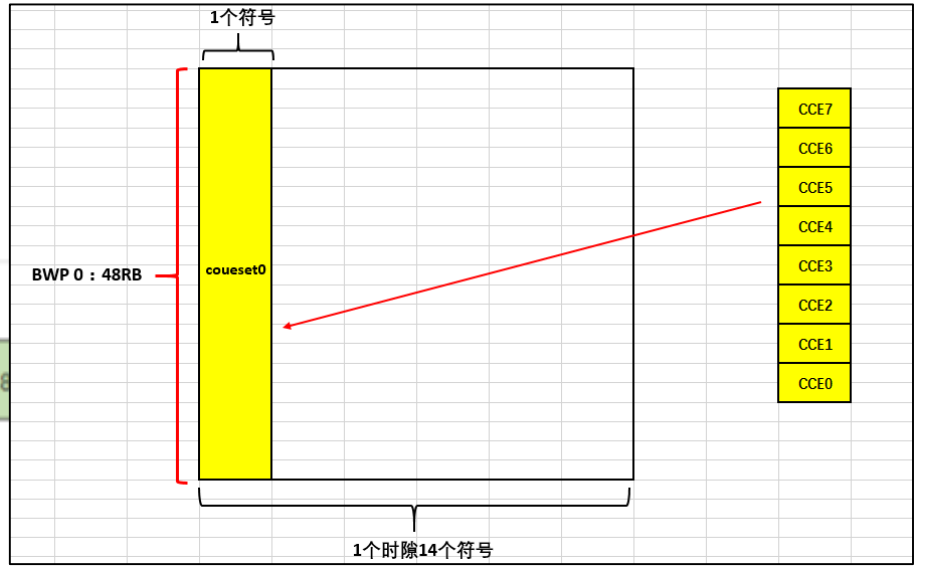
<38.213-Table 13-4>

Index	SS/PBCH block and CORESET multiplexing pattern	Number of RBs $N_{RB}^{CORESET}$	Number of Symbols $N_{symb}^{CORESET}$	Offset (RBs)
0	1	24	2	0
1	1	24	2	1
2	1	24	2	2
3	1	24	2	3
4	1	24	2	4
5	1	24	3	0
6	1	24	3	1
7	1	24	3	2
8	1	24	3	3
9	1	24	3	4
10	1	48	1	12
11	1	48	1	14
12	1	48	1	16
13	1	48	2	12
14	1	48	2	14
15	1	48	2	16

- mib
  - systemFrameNumber = 100011
  - subCarrierSpacingCommon = scs30or120
  - ssb-SubcarrierOffset = 6
  - dmrs-TypeA-Position = pos2
  - pdcch-ConfigSIB1
    - controlResourceSetZero = 10
    - searchSpaceZero = 4
  - cellBarred = notBarred
  - intraFreqReselection = allowed
  - spare = 1

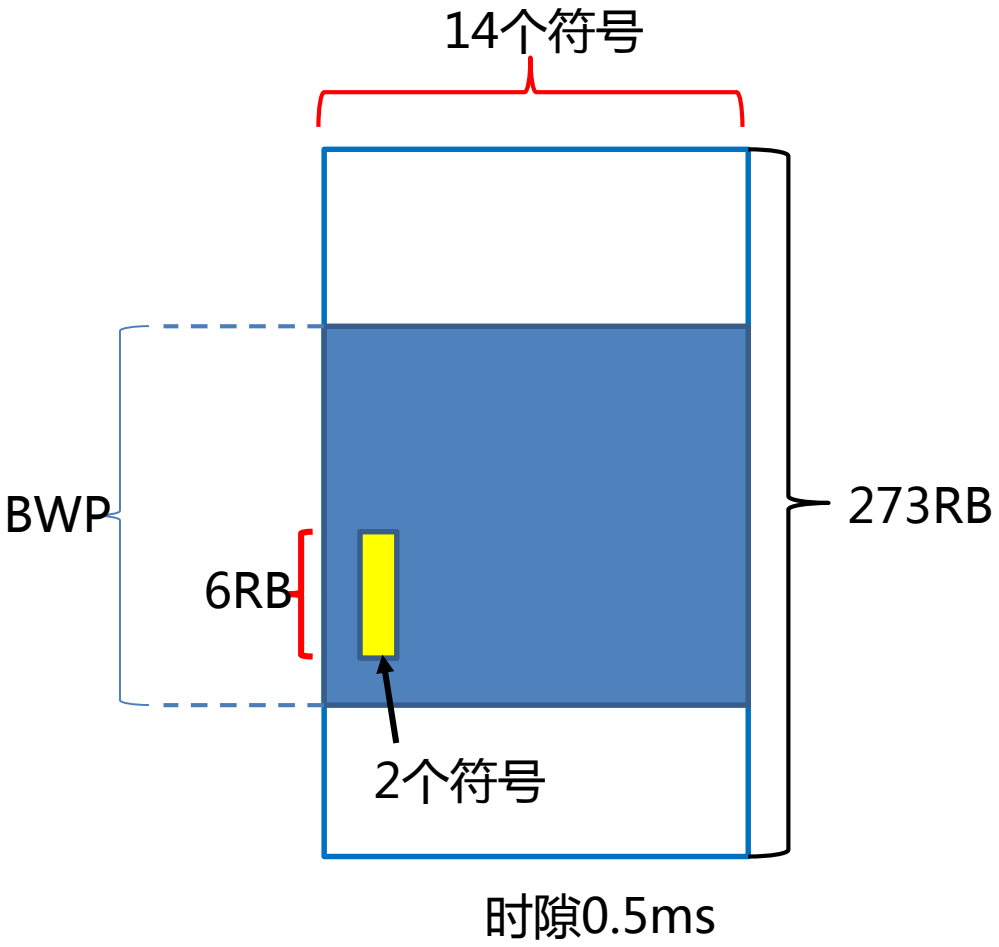
48个RB（8CCE），1个符号，相对SSB偏移12个RB





*构成coreset0的CCE固定使用交织的REG

去哪找到CORESET?



通过coreset的参数，我们发现一个事实：Coreset的参数配置仅仅指示了接收PDCCH的频域资源位置及符号个数，也就是PDCCH集合占用系统资源的大小。
Coreset在整个帧当中的占用那些时隙，以及在时隙内的符号位置，都是由另外一个参数来定义的，Search Space。

搜索空间search space

搜索空间，我们第一次接触，暂时可以这样理解：

coreset在整个帧结构当中，在哪个时隙有？起始符号是哪个符号？

重点在于时间轴上

对于搜索空间，比较特殊的是SIB1的搜索空间-SearchSpaceZERO（在MIB中）

其他的搜索空间相关设定，在统一的信令字段Search Space中（SIB1或者RRC信令中）。

我们先研究统一的，再看特殊的

搜索空间的参数配置

1个BWP可以配10个搜索空间，
因此，4个bwp就是40个搜索空间编号

```
SearchSpace ::=                 SEQUENCE {
    searchSpaceId                       SearchSpaceId,

    controlResourceSetId                ControlResourceSetId

    monitoringSlotPeriodicityAndOffset  CHOICE {
        sl1                                 NULL,
        sl2                                 INTEGER (0..1),
        sl4                                 INTEGER (0..3),
        sl5                             INTEGER (0..4),
        sl8                                 INTEGER (0..7),
        sl10                                INTEGER (0..9),
        sl16                                INTEGER (0..15),
        sl20                                INTEGER (0..19)
    }

    monitoringSymbolsWithinSlot             BIT STRING (SIZE (14))

    nrofCandidates                      SEQUENCE {
        aggregationLevel1                   ENUMERATED {n0, n1, n2, n3, n4, n5, n6, n8},
        aggregationLevel2                   ENUMERATED {n0, n1, n2, n3, n4, n5, n6, n8},
        aggregationLevel4                   ENUMERATED {n0, n1, n2, n3, n4, n5, n6, n8},
        aggregationLevel8                   ENUMERATED {n0, n1, n2, n3, n4, n5, n6, n8},
        aggregationLevel16                  ENUMERATED {n0, n1, n2, n3, n4, n5, n6, n8}
    }
```

搜索空间索引号，取值{0...39}：
- SS#0即为MIB定义的Type0公共搜索空间
- 搜索空间索引号在一个小区的所有BWP中唯一

搜索空间所属的CORESET号

搜索空间周期及偏置（以时隙为单位）

多少个时隙

偏置是哪一个

开始于哪一个符号，可以开始于2个符号

一个时隙中的PDCCH监听样式，以Bit Map形式给出CORESET在一个时隙中的起始符号

针对每种CCE聚合度给出PDCCH候选机会数

搜索空间参数举例1

搜索空间1

searchSpaceId: ---- 0x1(1) ---- ******00,0001****

搜索空间对应的coreset0

controlResourceSetId: ---- 0x0(0) ---- ****0000

每个时隙都有coreset

monitoringSlotPeriodicityAndOffset

sl1: ---- (0)

Coreset的起始符号为每时隙第一个符号

monitoringSymbolsWithinSlot: ---- '10000000000000'B ---- ****1000,00000000,00******

nrofCandidates

aggregationLevel1: ---- n0(0) ---- **000***

aggregationLevel2: ---- n0(0) ---- *****000

aggregationLevel4: ---- n4(4) ---- 100*****

aggregationLevel8: ---- n2(2) ---- ***010**

aggregationLevel16: ---- n1(1) ---- ******00,1*******

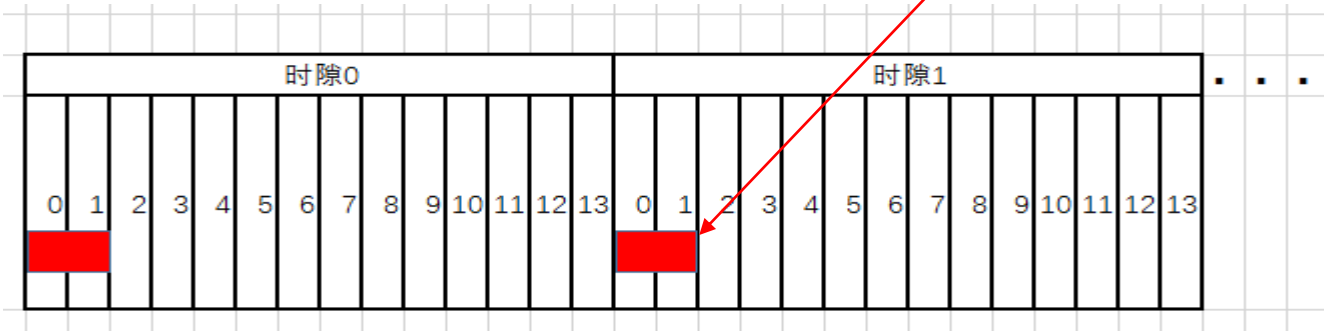
searchSpaceType

common

dci-Format0-0-AndFormat1-0: ---- (0) ---- *******0

假设coreset0：频域6RB，时域2符号

coreset0



搜索空间参数举例2

- controlResourceSetToAddModList
  - NRControlResourceSet
    - controlResourceSetId: ---- 0x1(1) ---- ***0001*
    - frequencyDomainResources: ---- '111100000000000000000000000000000000000000000000'B ---- *******1,
    - duration: ---- 0x1(1) ---- ****00**
    - cce-REG-MappingType
      - nonInterleaved: ---- (0)
    - precoderGranularity: ---- sameAsREG-bundle(0) ---- *******0
    - tci-StatesPDCCH-ToAddList
      - NRTCI-StateId: ---- 0x0(0) ---- ******00,00000***
    - pdcch-DMRS-ScramblingID: ---- 0xad8c(44428) ---- *****101,01101100,01100***
- searchSpacesToAddModList
  - NRSearchSpace
    - searchSpaceId: ---- 0x5(5) ---- *******0,00101***
    - controlResourceSetId: ---- 0x1(1) ---- *****000,1*******
    - monitoringSlotPeriodicityAndOffset
      - sl1: ---- (0)
    - monitoringSymbolsWithinSlot: ---- '10000000000000'B ---- *****100,00000000,000*****
    - nrofCandidates
      - aggregationLevel1: ---- n0(0) ---- ***000**
      - aggregationLevel2: ---- n6(6) ---- ******11,0*******
      - aggregationLevel4: ---- n2(2) ---- *010****
      - aggregationLevel8: ---- n2(2) ---- ****010*
      - aggregationLevel16: ---- n2(2) ---- *******0,10******
    - searchSpaceType
      - ue-Specific
        - dci-Formats: ---- formats0-1-And-1-1(1) ---- ****1***

定义了一个coreset，编号为coreset1

占用4*6=24个RB也就是4个CCE
频域上，从BWP的PRB0开始

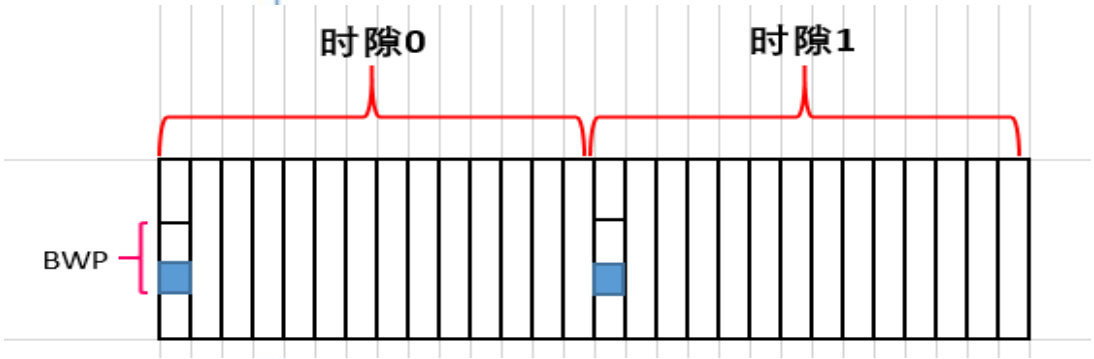
占用1个符号

定义了search space 5

Search space5与coreset1相对应

每个时隙都有coreset1

Coreset1从时隙的第一个符号开始



SIB1的搜索空间search space zero

Sib1的搜索空间，由MIB里面的字段定义

- mib
  - systemFrameNumber = 100011
  - subCarrierSpacingCommon = scs30or120
  - ssb-SubcarrierOffset = 6
  - dmrs-TypeA-Position = pos2
  - pdcch-ConfigSIB1
    - controlResourceSetZero = 10
    - searchSpaceZero = 4
  - cellBarred = notBarred
  - intraFreqReselection = allowed
  - spare = 1

后4bit解码出sib1的搜索空间searchsapcezero，
在哪个符号搜索，以及在哪个时隙搜索，在哪个帧搜索

Index	O	Number of search space sets per slot	M	First symbol index
0	0	1	1	0
1	0	2	1/2	{0, if i is even}, {$N_{symb}^{CORESET}$, if i is odd}
2	2	1	1	0
3	2	2	1/2	{0, if i is even}, {$N_{symb}^{CORESET}$, if i is odd}
4	5	1	1	0
5	5	2	1/2	{0, if i is even}, {$N_{symb}^{CORESET}$, if i is odd}
6	7	1	1	0
7	7	2	1/2	{0, if i is even}, {$N_{symb}^{CORESET}$, if i is odd}
8	0	1	2	0
9	5	1	2	0
10	0	1	1	1
11	0	1	1	2
12	2	1	1	1
13	2	1	1	2
14	5	1	1	1
15	5	1	1	2

search space zero

- mib
  - systemFrameNumber = 100011
  - subCarrierSpacingCommon = scs30or120
  - ssb-SubcarrierOffset = 6
  - dmrs-TypeA-Position = pos2
  - pdcch-ConfigSIB1
    - controlResourceSetZero = 10
    - searchSpaceZero = 4
  - cellBarred = notBarred
  - intraFreqReselection = allowed
  - spare = 1

Index	O	Number of search space sets per slot	M	First symbol index
0	0	1	1	0
1	0	2	1/2	{0, if i is even}, {$N_{symb}^{CORESET}$, if i is odd}
2	2	1	1	0
3	2	2	1/2	{0, if i is even}, {$N_{symb}^{CORESET}$, if i is odd}
4	5	1	1	0
5	5	2	1/2	{0, if i is even}, {$N_{symb}^{CORESET}$, if i is odd}
6	7	1	1	0
7	7	2	1/2	{0, if i is even}, {$N_{symb}^{CORESET}$, if i is odd}
8	0	1	2	0
9	5	1	2	0
10	0	1	1	1
11	0	1	1	2
12	2	1	1	1
13	2	1	1	2
14	5	1	1	1
15	5	1	1	2

O=5,M=1

UE监听偶数号帧，每个偶数帧里面，UE从时隙n_0开始，连续监听2个时隙

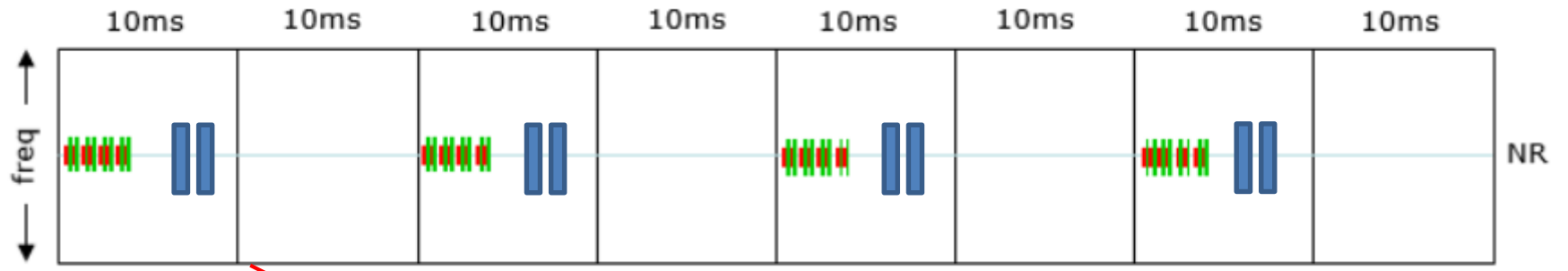
$$n_0 = (O * 2^{\mu} + \lfloor i * M \rfloor) mod N_{slot}^{frame,\mu}$$

=(5*2+[I*1])mod20=(10 +[I*1])mod20

公式中i这个参数是SSB的index（编号）
现网中，编号是0-7，8个SSB

由于是30KZH子载波间隔，因此μ=1

search space zero所在时隙



SSB0 0 1 2 3 4 5 6 7 8 9 **10 11** 12 13 14 15 16 17 18 19

SSB1 0 1 2 3 4 5 6 7 8 9 10 **11 12** 13 14 15 16 17 18 19

SSB2 0 1 2 3 4 5 6 7 8 9 10 11 **12 13** 14 15 16 17 18 19

SSB3 0 1 2 3 4 5 6 7 8 9 10 11 12 **13 14** 15 16 17 18 19

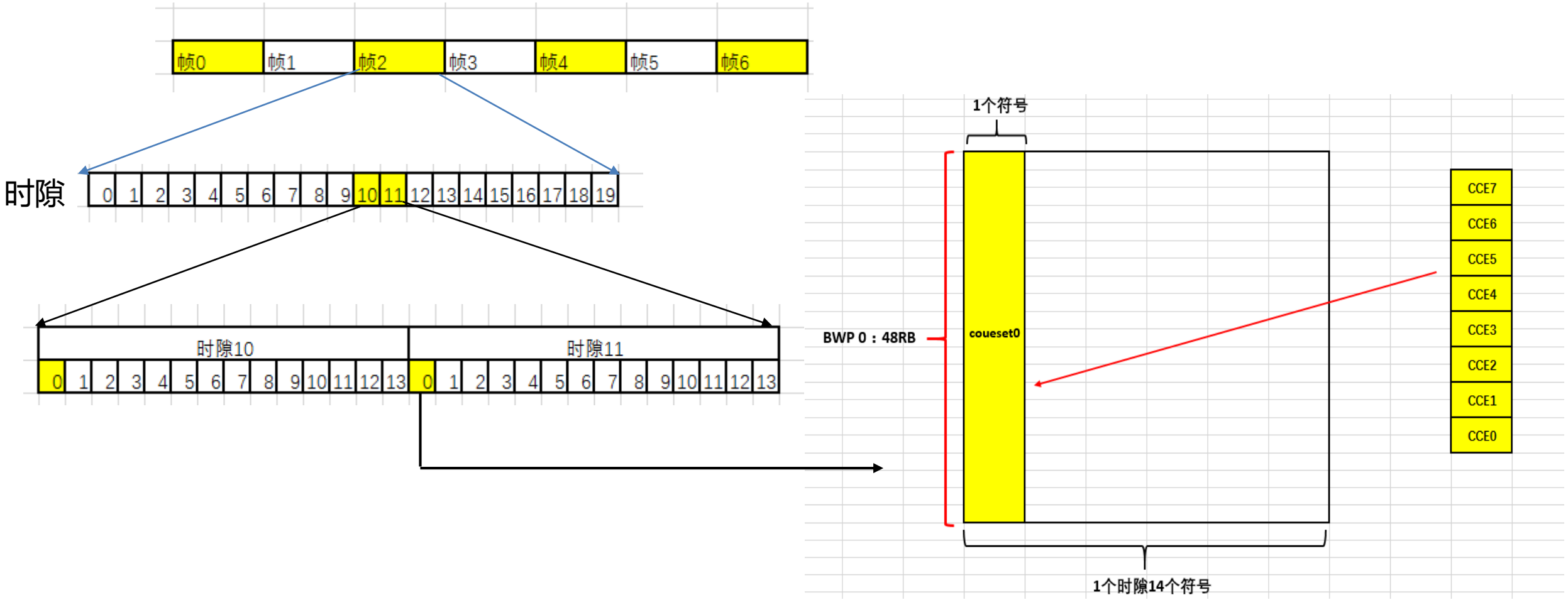
SSB4 0 1 2 3 4 5 6 7 8 9 10 11 12 13 **14 15** 16 17 18 19

SSB5 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 **15 16** 17 18 19

SSB6 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 **16 17** 18 19

SSB7 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 **17 18** 19

CORESET0所在符号



搜索空间的分类

搜索空间分为公共搜索空间(Common search space, CSS)和UE特定的搜索空间(UE-specific search space, USS)。CSS用于广播、寻呼、随机接入等相关的控制信息(小区级公共信息)。USS用于传输与DL-SCH、UL-SCH等相关的控制信息(UE级信息)。

搜索空间类型	RRC状态	CRC加扰	应用场景
Type 0 CSS	RRC空闲态	SI-RNTI	SIB1调度
Type 0A CSS		SI-RNTI	其他SIB消息调度
Type 1 CSS		RA-RNTI、TC-RNTI、C-RNTI	随机接入
Type 2 CSS		P-RNTI	寻呼调度
Type 3 CSS	RRC连接态	SFI-RNTI、INT-RNTI、TPC-PUSCH-RNTI、TPC-PUCCH-RNTI、TPC-SRS-RNTI、C-RNTI、MCS-C-RNTI,、 CS-RNTI	公共信令
USS		C-RNTI, MCS-C-RNTI, SP-CSI-RNTI,、 CS-RNTI	UE上下行调度

定义了多种搜索空间，这意味着，不同种类的信息，接收的时隙和符号位置，可以不同，换句话说，不同种类的信息，接收的节奏可以不同（也可以相同）。

搜索空间类型的定义

在SIB1当中(或者RRC信令中服务小区配置)，定义了每一种搜索空间

```
SIB1 or
BWP-DownlinkCommon in ServingCellConfig or ServingCellConfigCommon

PDCCH-ConfigCommon ::=                  SEQUENCE {

    commonControlResourceSets                   SEQUENCE (SIZE(1..2)) OF ControlResourceSet

    commonSearchSpaces                          SEQUENCE (SIZE(1..4)) OF SearchSpace

    searchSpaceSIB1                             SearchSpaceId

    searchSpaceOtherSystemInformation           SearchSpaceId

    pagingSearchSpace                           SearchSpaceId

    ra-ControlResourceSet                   ControlResourceSetId

    ra-SearchSpace                          SearchSpaceId
    ...
}
```

公共CORESET集，定义了1~2个公共CORESET：
- CORESET#0即为MIB定义的CORESET，在切换或(P)SCell添加时通过信令给到UE
- CORESET#1（如果配置的话）用于RAR

公共搜索空间集，定义了1~4个公共搜索空间

Type0公共搜索空间（相当于MIB中的*pdcch-ConfigSIB1*字段）

Type0A公共搜索空间

Type2公共搜索空间

Type1公共搜索空间的CORESET

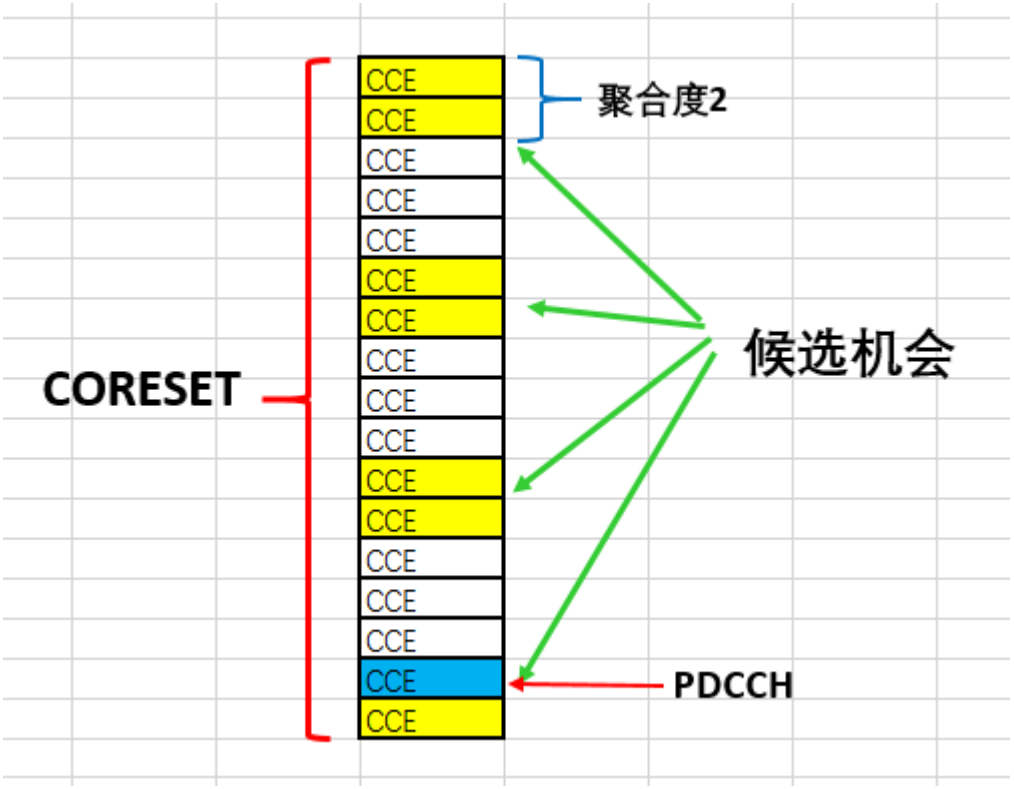
Type1公共搜索空间

搜索空间类型的定义

候选集Candidates

在coreset里面，去搜索用户的CCE，并不是所有的CCE都会去搜索，而是定义了一个候选集Candidates，UE需要的CCE，放在候选集里面的某一个里面，只需要在候选集当中搜索即可。这样可以让UE减少搜索时间

举个例子



候选集Candidates

- 对于USS和CSS，聚合等级和聚合等级对应的候选集个数都在SearchSpace参数中配置，nrofCandidates 配置聚合级别L={1,2,4,8,16}包含的PDCCH候选数目。

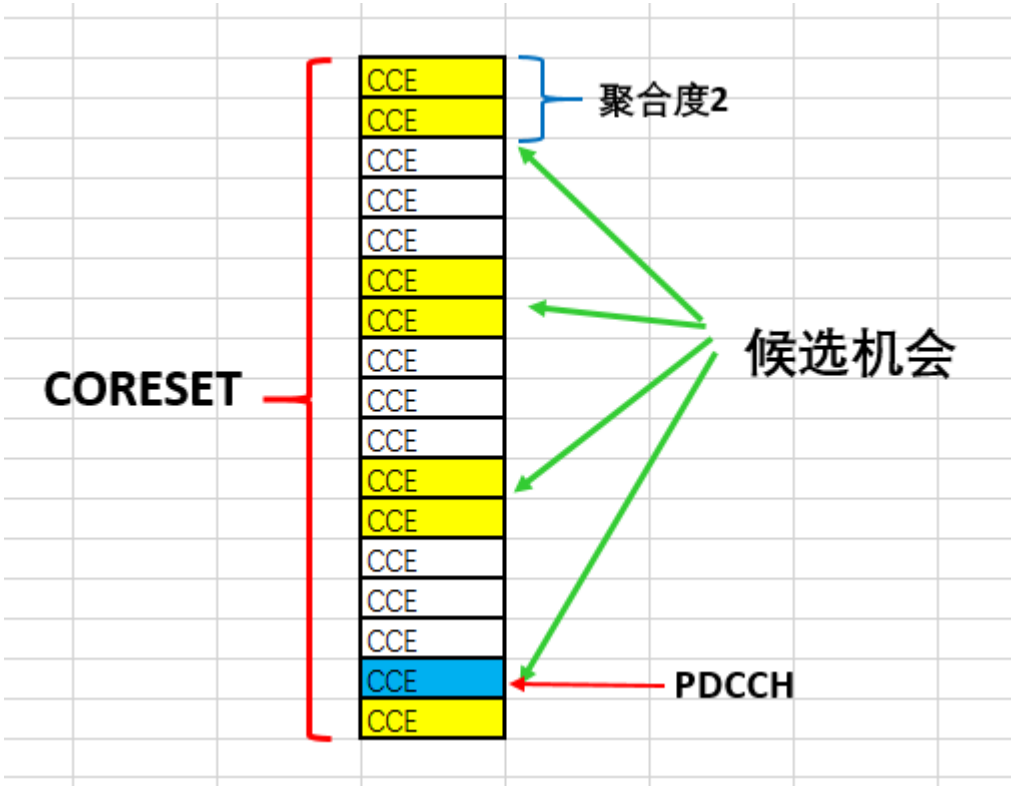
```
SearchSpace ::=                    SEQUENCE {
   searchSpaceId                     SearchSpaceId,
   controlResourceSetId                 ControlResourceSetId                    OPTIONAL,
-- Cond SetupOnly
   monitoringSlotPperiodicityAndOffset     CHOICE {

   nrofCandidates                    SEQUENCE {
      aggregationLevel1                 ENUMERATED {n0, n1, n2, n3, n4, n5, n6, n8},
      aggregationLevel2                 ENUMERATED {n0, n1, n2, n3, n4, n5, n6, n8},
      aggregationLevel4                 ENUMERATED {n0, n1, n2, n3, n4, n5, n6, n8},
      aggregationLevel8                 ENUMERATED {n0, n1, n2, n3, n4, n5, n6, n8},
      aggregationLevel16                 ENUMERATED {n0, n1, n2, n3, n4, n5, n6, n8}
```

- 对于CSS，在参数未配置时，采用下表默认的定义

CCE Aggregation Level	Number of Candidates
4	4
8	2
16	1

PDCCH聚合等级和searchspace聚合等级

PDCCH聚合等级和search聚合等级不是一回事

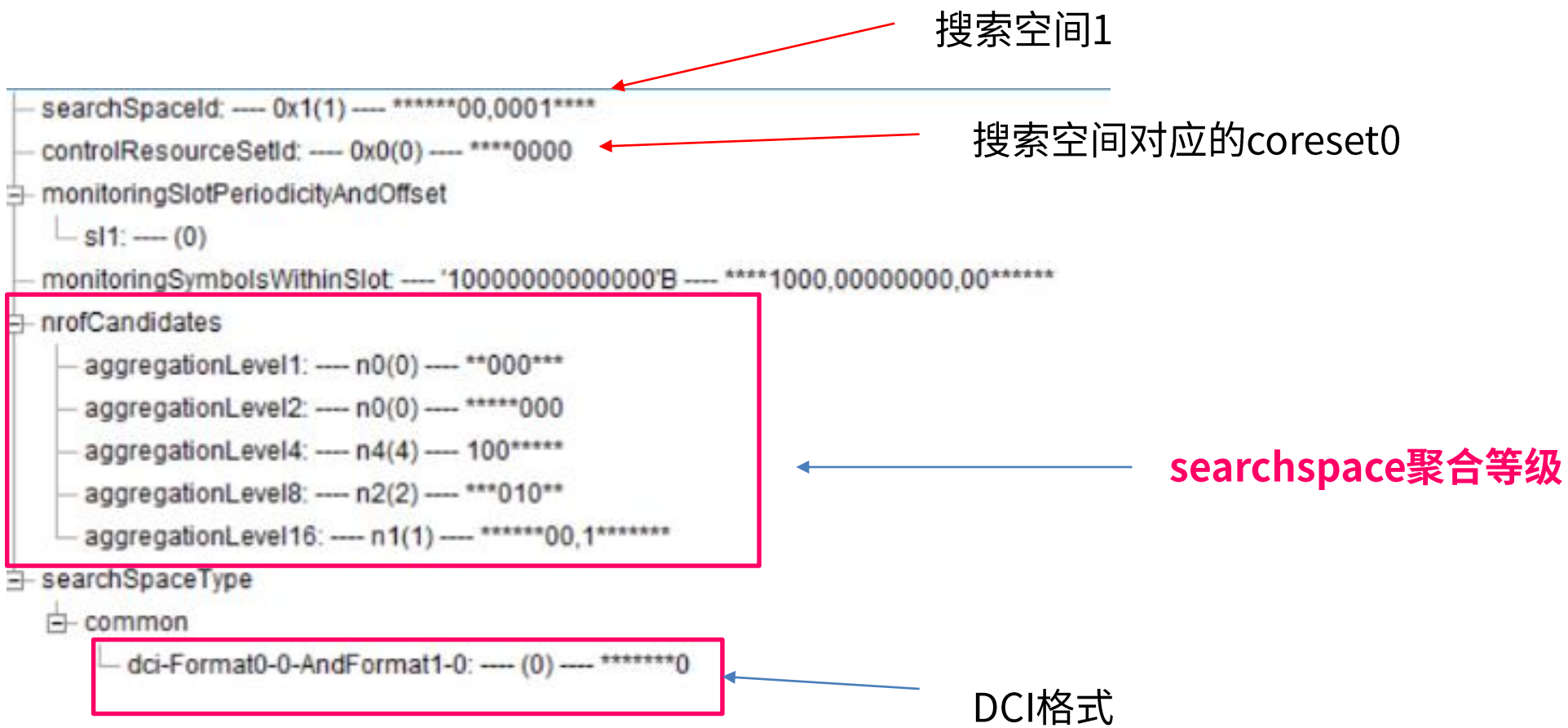


PDCCH聚合等级代表了一个PDCCH占用几个CCE

而Search Space聚合等级是在coreset当中一次搜索几个CCE

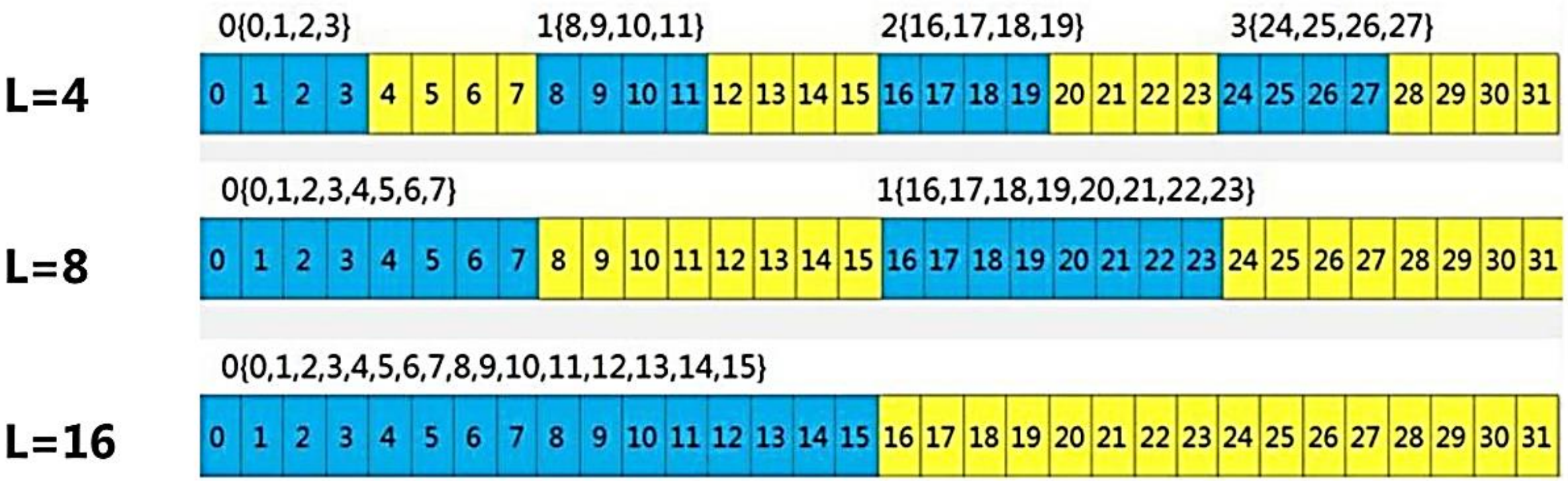
左图中，PDCCH的聚合等级为1cce，而search space的聚合等级为2cce。

searchspace聚合等级



候选集举例

- 对于公共搜索空间，CORESET中CCE个数为32，聚合等级L=4/8/16时，PDCCH候选集个数等于4/2/1。



搜索空间与DCI，RNTI之间的对应关系

搜索空间	DCI格式	加扰的RNTI类型	内容
type1 CSS\USS	DCI0-0	C-RNTI、CS-RNTI、MCS-C-RNTI、TC-RNTI	UE调度/RA
USS	DCI0-1	C-RNTI、CS-RNTI、MCS-C-RNTI、SP-CSI-RNTI	UE调度
type0\TYPE0A CSS	DCI1-0	SI-RNTI	SIB1\OSI
type1 CSS		RA-RNTI	RA
		TC-RNTI	
USS		C-RNTI	UE调度
type2 CSS		P-RNTI	paging
USS	DCI1-1	C-RNTI、CS-RNTI、MCS-C-RNTI	UE调度
type3 css	DCI2-0	SFI-RNTI	时隙格式
	DCI2-1	INT-RNTI	UE通知不可用的PRB和OFDM符号
	DCI2-2	TPC-PUSCH-RNTI、TPC-PUCCH-RNTI	功率控制
	DCI2-3	TPC-SRS-RNTI	功率控制

找到PDCCH　　PDCCH里面DCI的格式　　需要啥信息，就用啥RNTI来解扰PDCCH　　DCI的功能


感谢观看