PUCCH : Introduction

PUCCH → Physical Uplink Control Channel.

PUCCH carries UCI (Uplink control information).

UCI has mainly 3 parts.

① HARQ (ACK/NACK for DL Transport block)

Whatever the PDSCH is carrying, whether that is successfully decoded or not. This information is sent back to the gNB through UCI.

② Scheduling Request (SR).

If UE wants to transmit data, and it doesn't have resources, then it asks for resources by sending SR..

③ Channel State Information (CSI)

UE decodes CSI-RS, and sends the Channel State Information back to gNB, for better scheduling.

PUCCH has 5 formats, which have different capacity, different NO. of bits, different UE multiplexing, different resource allocation, different modulation, and soon. Based on the above, we find the right format to use.

There are 2 Categories of PUCCH formats.

First Category:

① Short Format → F0 and F2

→ Occupies less number of Symbols

(i) 1 or 2 Symbols at max.

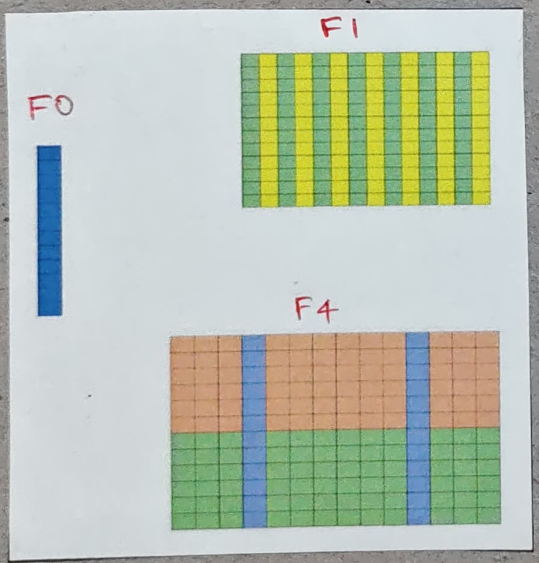
→ So, they are short in time.

⊙ Long Format → F1, F3 and F4
→ Occupies more PRBs (in frequency)
→ Occupies more Symbols (in time)
(ii) 4 to 14 symbols.
→ So, they are long in time.

Second Category : (Based on information they carry).

⊙ Carries only HARQ and SR
→ F0 and F1
→ Takes 1 or 2 bits each
→ They do not carry CSI

⊙ carries HARQ, SR and CSI
→ F2, F3 and F4



Figures show the visualization of four different formats.

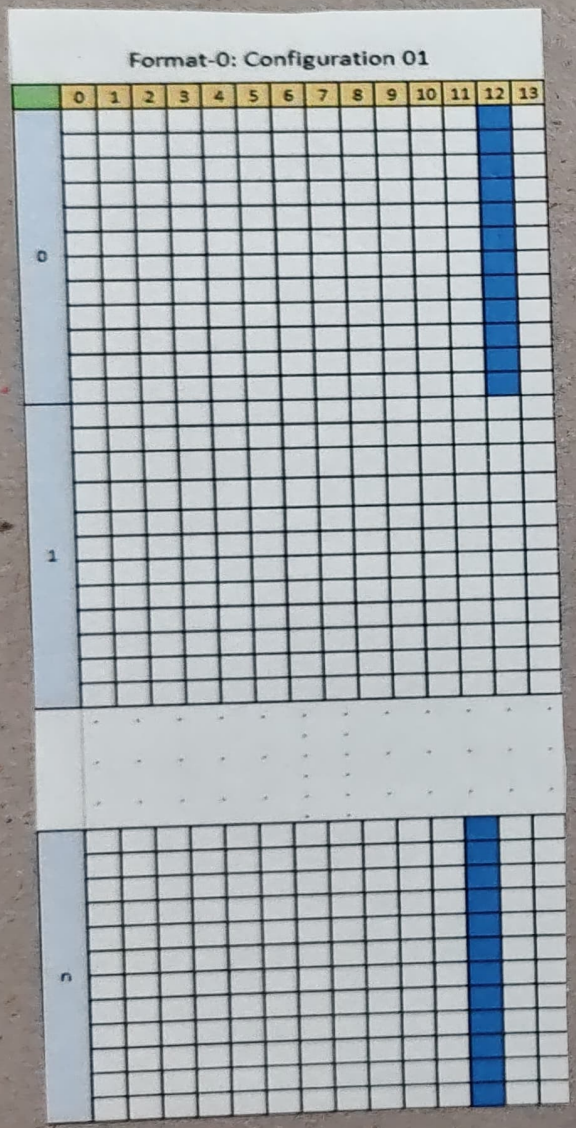
⊙ F0 has 1 PRB and 1 symbol. It can take 2 Symbols as well at max.

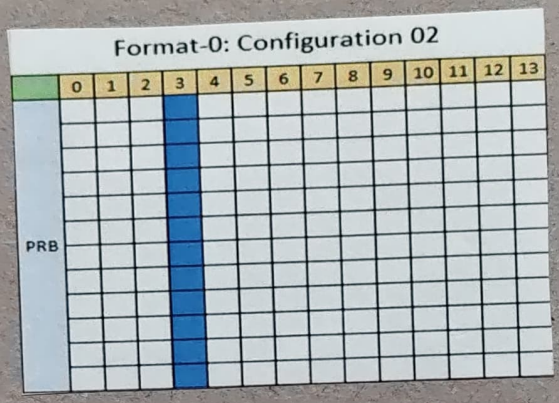
- F1 can take One PRB and 4 to 14 Symbols.
- F2, where Data and DMRS are interleaved in frequency. This can take 1 to 2 symbols, and 1 to 16 PRBs.
- F3 is the largest format, and having largest capacity as well. This can take 4 to 14 Symbols, and 1 to 16 PRBs. We have only selective combination in terms of PRBs, but at max it can take 16 PRBs.
- F4 takes 4 to 14 symbols, and 1 PRB.

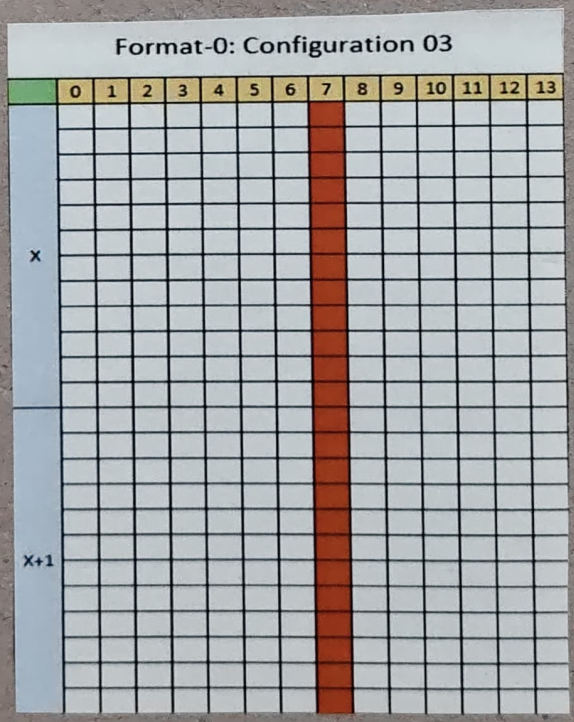
- Formats 0, 1 and 4 allow UE multiplexing
- Formats 2 and 3 do not allow UE multiplexing

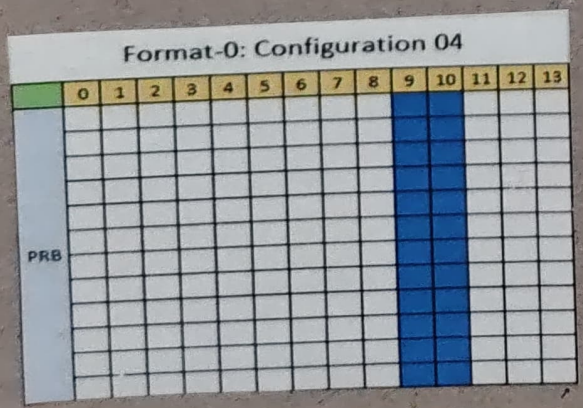
PUCCH Format 0 : Signal Generation and Resource Mapping.

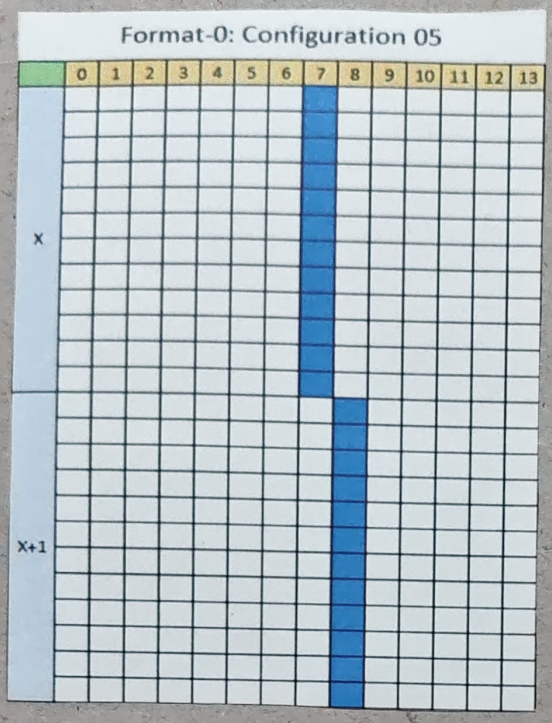
As we know, PUCCH Format 0 is a short Format. It takes 1 PRB, and 1 or 2 symbols for allocation. This short format carries ACK/NACK and Scheduling Request (SR).











How the ACK/NACK and SR are mapped to different signals or different sequences generated, We'll see in next section.

In this section, we'll understand how the signal is generated and how it is mapped to the resources.

Figures show the visualization of Signal generation & Resource mapping.

As we see, Format 0: Configuration 02, F0 is allocated 1 PRB and Symbol 3. Usually, this is how F0 allocation looks like.

It can take upto 2 symbols. As we see, in Format 0: configuration 01, F0 takes PRB 0 and Symbol 12, as well as PRB n and Symbol 11. This configuration, where the same data is hopped to a different location, is called "Frequency hopping"

F0 allocation can also take 1 PRB and 2 Symbols, as shown in Figure Format 0: Configuration 04. In this case, there is no frequency hopping.

In Format 0: configuration 03, we have two consecutive PRBs and same symbol allocated. This particular configuration is invalid (not valid for PUCCH). Here, instead of having PUCCH in same symbol, we can have this

allocated in different symbol, as shown in Format 0: Configuration 05.

When to choose configuration 4 and configuration 5 ?

Configuration 4 uses same PRB, and configuration 5 uses different PRBs.

When the channel is varying over frequency, in that case, it is better to allocate different PRBs, coz the channel is bad at one PRB and is better at another PRB, hence the PUCCH going through the better channel helps to decode the transmitted data. This is the advantage in configuration 5 over configuration 4.

This phenomenon is basically called as "Frequency Diversity".

PUCCH Format 0, the signal generation and Mapping is very straight forward.

The data to be transmitted takes 2 different type of information.

HARQ (ACK/NACK) — 2 bits
SR — 1 bit
} 3 bits for a single UE.

PUCCH Format 0 is similar to PRACH, where we take the information, and based on the information we generate a Zc sequence. This Zc sequence is directly transmitted.

There can be different Zc sequences (different cyclic shifts) and transmit all of them in Frequency domain

(c) after IFFT and CP insertion, the sequence is transmitted.

At the receiver end, gNB generates the base sequence, correlates the base sequence with the received sequence, and finds out which sequence was transmitted. And, from the sequence, it finds out the data (HARQ and SR) which was transmitted.

As we see, there is no CRC check like we have in other channels. So, there is no way to find out whether the data that we received is correct or not. So, the sequence/signal generated at UE should be designed in a way that it is very robust, and the probability of false detection should be very low.

As per 3GPP, the sequence should be in such a way that, the probability of error should be $< 1\%$.

(i) $\left(\frac{\text{ACK detected}}{\text{NO transmission from UE}}\right) < 1\%$.

(ii) $\left(\text{UE did not send anything, but the gNB still detected ACK}\right) < 1\%$.

This is actually applicable for all PUCCH formats.

PUCCH sequence generation follows this.

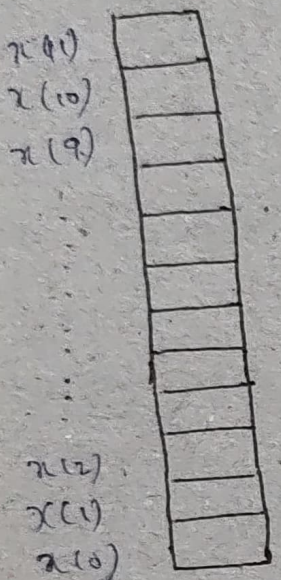
$$x(l \cdot N_{sc}^{RB} + n) = r_{u,v}^{(\alpha,\delta)}(n)$$

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

$$l = \begin{cases} 0 & \text{for single-symbol PUCCH transmission} \\ 0,1 & \text{for double-symbol PUCCH transmission} \end{cases}$$

This is a pretty straight forward formula, where we generate the Zc sequence $r_{u,v}^{(\alpha,\delta)}(n)$, and we map it to x, where x is the final generated sequence.

Here, N_{sc}^{RB} is 12 .. The sequence length is also 12, which is mapped to a single symbol single PRB.



The No. of symbols can also be 2. In that case, different sequence is generated. Here, we consider the simple case, when $l = 0$ (i) single symbol.

For single symbol, the sequence becomes

$$x(n) = r_{u,v}^{(\alpha,\delta)}(n)$$

and the mapping is very straight forward.

$\delta \rightarrow$ used to calculate the length of ZC sequence.

$n \rightarrow 0, 1, 2, \ldots, 11$

$u \rightarrow$ Provides Group hopping

$v \rightarrow$ Sequence hopping.

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

In PUCCH case, there are 30 root sequences. Out of 30, we pick one of them. Since, each root sequence has only one sequence, here sequence hopping is disabled in PUCCH Fo case. (i) $v = 0$ always

only group hopping (u) is allowed.

So, u is used to pick one of the 30 sequences, using the parameter $(f_{gh} + f_{ss})$.

$f_{gh} = 0$
$f_{ss} = n_{ID} \bmod 30$
$v = 0$

→ This is the case when the group hopping is disabled.

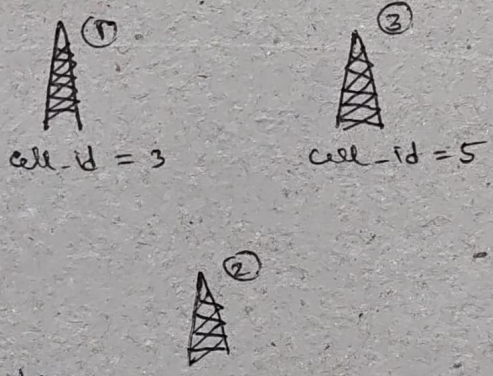
→ v is used for sequence hopping
f_{gh} is used for Group hopping
$f_{ss} = n_{ID} \bmod 30$

→ n_{ID} can be a cell ID or an Hopping ID.

→ f_{ss} can be set as a single value for a particular cell, which removes the Intra cell Interference.

How ?

If there are multiple cells, then n_{ID} can be set as cell_id.



So, for cell 1, $f_{ss} = 3 \bmod 30$

for cell 2, $f_{ss} = 4 \bmod 30$

for cell 3, $f_{ss} = 5 \bmod 30$.

(ii) we can provide different f_{ss} values for different cells. So, this way, we are actually providing different root sequences to different cell.

→ If PUCCH Format 0 is transmitted at the same time and frequency to different neigeboring cells, and if we use different root sequences, then the interference between different PUCCH Format 0 transmitted will be low, since the correlation between different root sequences is very low.

$$f_{gh} = \left(\sum_{m=0}^{7} 2^m c\left(8\left(2n_{s,f}^{\mu} + n_{hop}\right) + m\right)\right) \bmod 30$$

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

$$v = 0$$

→ This is the case when the group hopping is enabled.

→ f_{gh} is a function of <u>Slot number</u> $\left(n_{s,f}^{\mu}\right)$ and the <u>frequency hopping index</u> $\left(n_{hop}\right)$. That means, we can have different sequence for different slot and for a different frequency hop.

→ $n_{hop} = 0$ or 1.

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

Now, coming back to sequence generation. $r_{u,v}^{(\alpha,\delta)}(n)$ is defined as above, where $e^{j\alpha n}$ is used to provide cyclic shift. And α is defined as follows for F0.

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

where, m_0 → Initial cyclic shift
→ This value is different for different UEs.

m_{cs} → Most important parameter
→ Carries HARQ (ACK/NACK) and SR

As we know, the sequence length is 12. So, α_l can take values from 0 to 11. Initial cyclic shift (m_0) also take value from 0 to 11. m_{cs} also takes value from 0 to 11.

n_{cs} → function of slot number $\left(n_{s,f}^{\mu}\right)$ and OFDM symbol number $(l + l')$

n_{cs} is used to randomize the cyclic shift across the slot and OFDM symbols. How?

$$n_{cs}(n_{s,f}^{\mu}, l) = \sum_{m=0}^{7} 2^m \, c(8N_{symb}^{slot} n_{s,f}^{\mu} + 8l + m)$$

$$c_{init} = n_{ID}$$

where, $N_{symb}^{slot} = 14$

$n_{s,f}^{\mu} \rightarrow$ slot number

$l \rightarrow$ symbol number

$m \rightarrow 0$ to 7

This PN sequence, $c(8 N_{symb}^{slot} n_{s,f}^{\mu} + 8l + m)$ is initialized with $c_{init} = n_{ID}$, where n_{ID} can be Hopping ID or cell ID.

If slot number = 0, symbol number = 0.

$$\Rightarrow n_{cs}(0,0) = \sum_{m=0}^{7} c(0 + 0 + m)$$

If slot number = 0, symbol number = 1

$$\Rightarrow n_{cs}(0,1) = \sum_{m=0}^{7} c(0 + 8 + m)$$

If slot number = 0, symbol number = 2

$$\Rightarrow n_{cs}(0,2) = \sum_{m=0}^{7} c(0 + 16 + m).$$

For the given slot number and symbol number, the way we pick $c(\cdot)$ is continuous.

We know, how PN sequence is generated. we neglect first 1600 values, and take values from 1601 onwards.

So, for slot 0, symbol 0, we pick first 8

values. For next symbol, we pick next 8 values, and soon. And, for the next slot,

If slot number = 1, Symbol number = 0;

$$\Rightarrow \underbrace{n_{cs}(1,0)} = \sum_{m=0}^{7} c\left((8\times 14\times 1) + 0 + m\right)$$

and it goes on..

So, basically, we take 8 values from PN sequence and convert them into an integer, so that the integer could be totally a random number for a given slot number and Symbol number. $n_{cs}(\cdot)$ value can take anywhere from 0 to 255.

This is to provide different cyclic shift for a given slot and symbol number. (ii) Randomize the cyclic shift for different slot and Symbol.

Now, the n_{cs}, m_{cs} and m_0 altogether contribute to find the α_ℓ value (final cyclic shift).

This α_ℓ value is used by $r_{u,v}^{(\alpha,\delta)}(n)$, which in turn gives the final PUCCH sequence $x(\cdot)$.

This $x(\cdot)$ is accordingly mapped to the resources.

Note: In case of double-Symbol PUCCH transmission,

(i) there are 2 OFDM symbols, $l = 0, 1$.

So, we find two α values (α_0 and α_1)

Even if Group hopping (f_{gh}) is enabled, it is not a function of Symbol number. It is only a function of slot number.

So, in case of double-Symbol PUCCH transmission, the only parameter that needs attention is α_l.

Using α_0 and α_1, different sequences will be generated, and is mapped to different Symbol where the resource is allocated.

This is how PUCCH Format 0 signals are transmitted. (Signal generation and Mapping).

PUCCH Format 0 : UE Multiplexing and Receiver Design

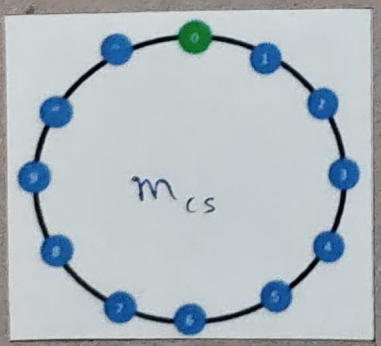
$$\alpha_l = \frac{2\pi}{N_{sc}^{RB}}\left(\left(m_0 + m_{cs} + n_{cs}(n_{s,f}^{\mu}, l+l')\right) \bmod N_{sc}^{RB}\right)$$

In the previous section, we've seen this α_l, which is a function of Initial cyclic shift (m_{i0}), m_{cs} (carries HARQ and SR), n_{cs} (cyclic shift, function of slot number and symbol number). And α_l takes values from 0 to 11.

Followed by the α_l calculation, we generate Zadoff-Chu sequence, and then we transmit the signal.

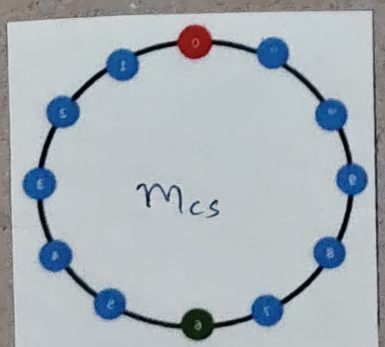
	0	1	2	3	4	5	6	7	8	9	10	11
SR	SR											
1 bit HARQ	N						A					
2 bit HARQ	N, N			N, A			A, A			A, N		
1 bit HARQ and SR	N			N, SR			A			A, SR		
2 bit HARQ and SR	N, N	N, N, SR		N, A	N, A, SR		A, A	A, A, SR		A, N	A, N, SR	

Table shows how the information is encoded. The goal is that, the Gap between two cyclic shifts should be maximum; so that we can minimize the error.



If we want to send SR, then $m_{cs} = 0$ (in case of Positive SR).

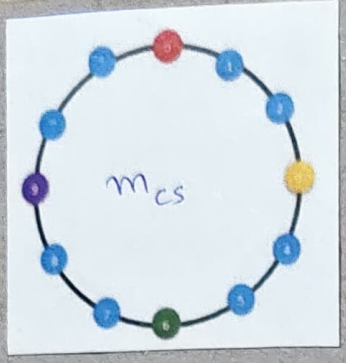
In case of negative SR, UE does not send anything.



If we send 1-bit HARQ, then, out of 12, we pick the two with maximum distance. (i) 0 and 6.

0 → NACK

6 → ACK



If we send 2 bit HARQ, then out of 12, we pick 4 values with maximum distance.

0 → NACK, NACK
3 → NACK, ACK
6 → ACK, ACK
9 → ACK, NACK

Note that, where we move from one cyclic shift (m_{cs}) to another, there is only single information change, so that, even if we falsely detect the signal at the receiver end, only one information goes wrong.

For example, m_{cs} moves from 0 to 3. (Min. distance)

0 → NACK, NACK
3 → NACK, ACK

(or) m_{cs} moves from 0 to 9 (Min. distance)

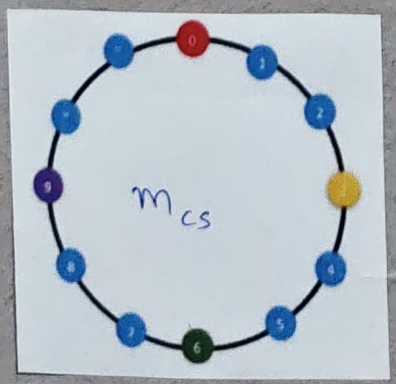
0 → NACK, NACK
9 → ACK, NACK

When the distance between cyclic shifts is maximum, say, m_{cs} moves from 9 to 3 (or) 0 to 6, then

Both the information goes wrong:

9 → ACK, NACK
3 → NACK, ACK

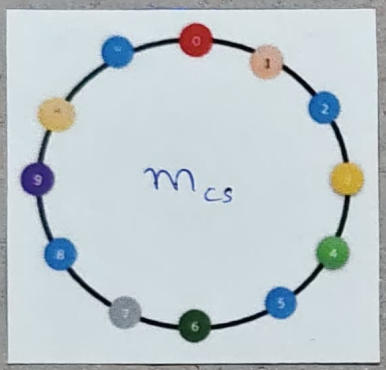
0 → NACK, NACK
6 → ACK, ACK



If we send 1 bit HARQ and SR,

0 → NACK, NO SR (-ve SR)
6 → ACK, NO SR (-ve SR)
3 → NACK, +ve SR
9 → ACK, +ve SR

Notice here, if we send $m_{cs} = 0$, and is falsely detected as $m_{cs} = 3$, it is okay since the SR is falsely detected but not the HARQ. This is the reason why HARQ variations is kept far apart. (we prioritize HARQ than SR).



If we send 2-bit HARQ and SR,

0 → NACK, NACK, -ve SR
1 → NACK, NACK, +ve SR
3 → NACK, ACK, -ve SR
4 → NACK, ACK, +ve SR.
6 → ACK, ACK, -ve SR
7 → ACK, ACK, +ve SR
9 → ACK, NACK, -ve SR
10 → ACK, NACK, +ve SR

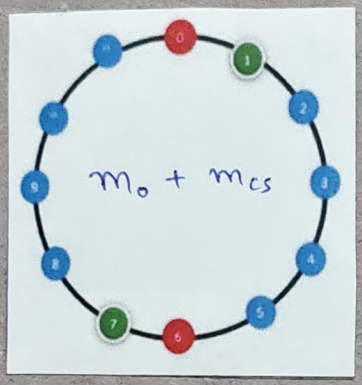
As we saw in the previous case, it's okay if SR goes wrong. So, it is less prioritize than the HARQ (ACK/NACK).

This is how the information is encoded.

For example, when UE wants to send 2-bit HARQ and +ve SR, it takes $m_{cs} = 7$, with initial cyclic shift (n_0) and m_{cs}, generates α_l, generate zc sequence, map it to REs, and then transmit the signal.

How multiple UEs are multiplexed?

WKT, the Initial cyclic shift (m_0) is different for different UEs.



Let us take the case of 1-bit HARQ.

For UE1, initial cyclic shift (m_0) = 0

For UE2, initial cyclic shift (m_0) = 1.

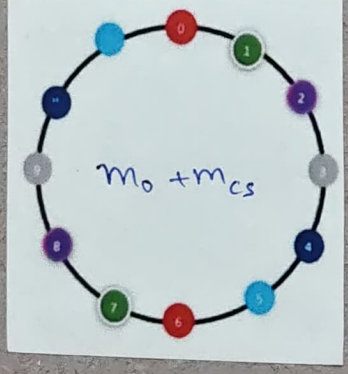
WKT, in 1-bit HARQ case,

$$m_{cs} = \begin{cases} 0 \rightarrow \text{NACK} \\ 6 \rightarrow \text{ACK} \end{cases}$$

For UE1, $m_0 + m_{cs}$ = 0 and 6

For UE2, $m_0 + m_{cs}$ = 1 and 7

This is how two UEs are multiplexed.



In case of 3 UEs multiplexing,

For UE1, m_0 = 0

For UE2, m_0 = 1

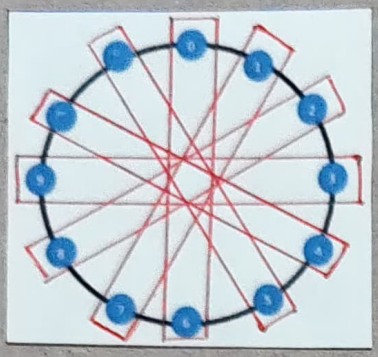
For UE3, m_0 = 2

For UE1, $m_0 + m_{cs}$ = 0 and 6

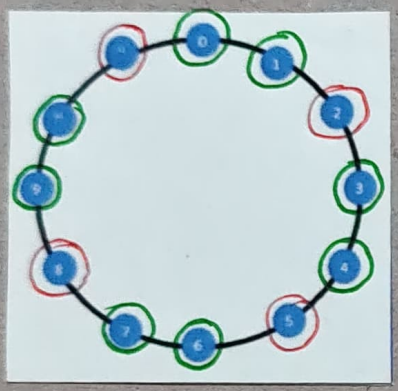
For UE2, $m_0 + m_{cs}$ = 1 and 7

For UE3, $m_0 + m_{cs}$ = 2 and 8

In similar fashion, the multiplexing can be done for SR, 2-bit HARQ, .. as well.



For sending 1 bit (SR / 1 bit HARQ), 6 UEs can be multiplexed.



If One UE is sending "2 bit HARQ and SR", the cyclic shifts marked green are already taken. So, Out of 12, 8 are taken. In this case, if we want to do UE multiplexing, there are 4 cyclic shifts marked red are available, using which 2 more bits can be sent. So, the second UE can do either "2 bit HARQ" or "1 bit HARQ and SR".

This is how UE multiplexing is done. (i) both the UEs wants to send the signal at the same time and frequency resources of Format 0.

Why do we want multiple UEs to be multiplexed at the same resources ?

In Figure (a), we allocate 3 PRB s, one for each UE to send the PUCCH data.

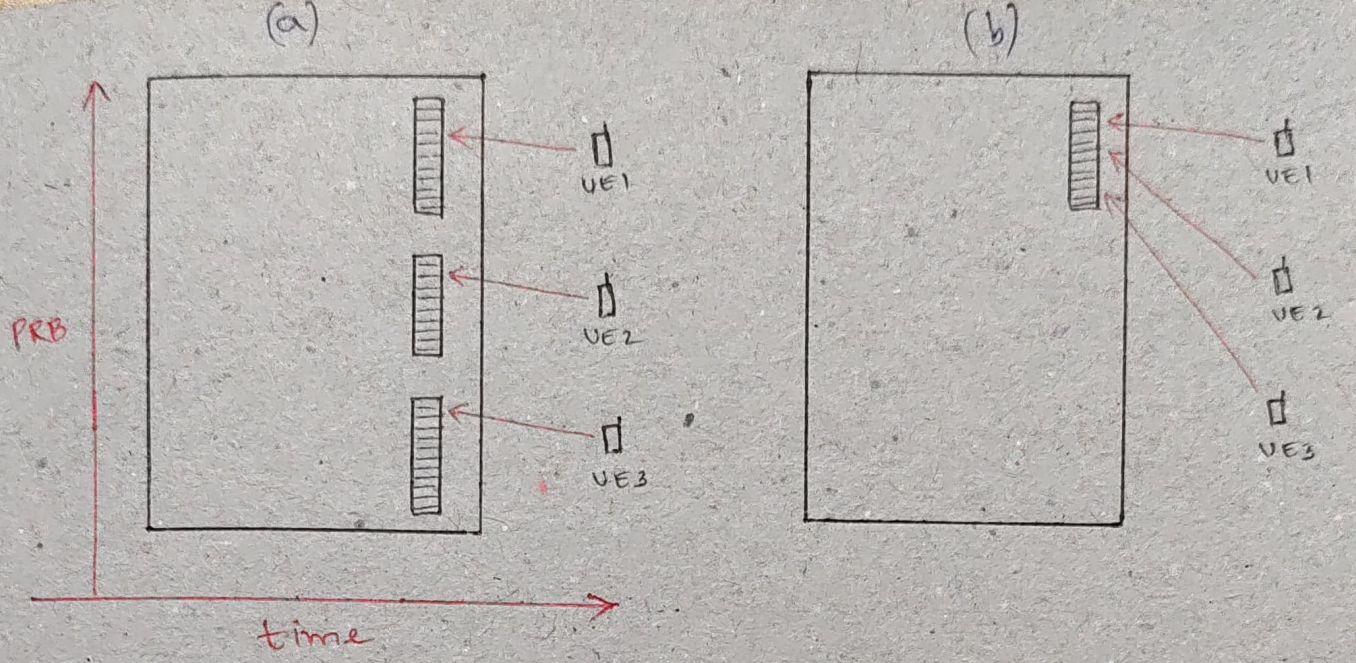


Figure : PUCCH Format 0 (with / without UE multiplexing).

In order to effectively utilize the resources, the PUCCH data from multiple UEs can be multiplexed and sent using single PRB., saving two PRBs.

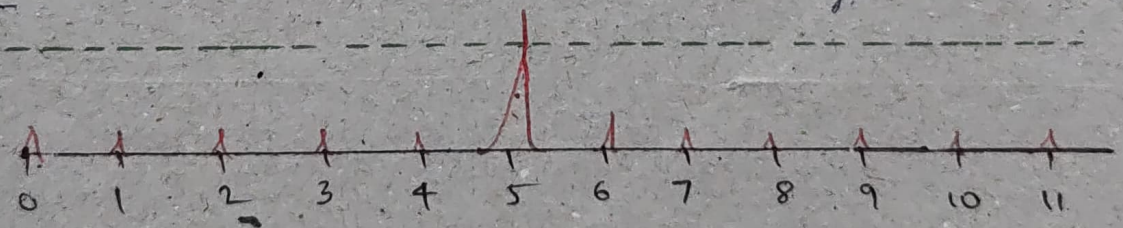
This is the reason, we can multiplex upto 6 UEs (where each UE sends 1 bit) in a single PRB.

At the receiver end, gNB knows the Root Sequence Index (u). And, it needs to find out what information was transmitted by the UE or those multiple UEs.
(i) ACK/NACK, SR transmitted by UEs.

So, gNB has a received signal on a PRB (if frequency hopping enabled, then 2 PRBs). Received signal implies IQ samples.

Since, the gNB knows the configuration it has given to the UE to transmit PUCCH, it can generate the Base sequence. (ii) it knows Symbol number, slot number, Root Sequence index, initial cyclic shifts.

Now, the gNB correlates the received signal (IQ samples) with the Base sequence, gives peak value wherever the correlation is High.



If the correlation output is above the threshold, then it can be considered as received signal.

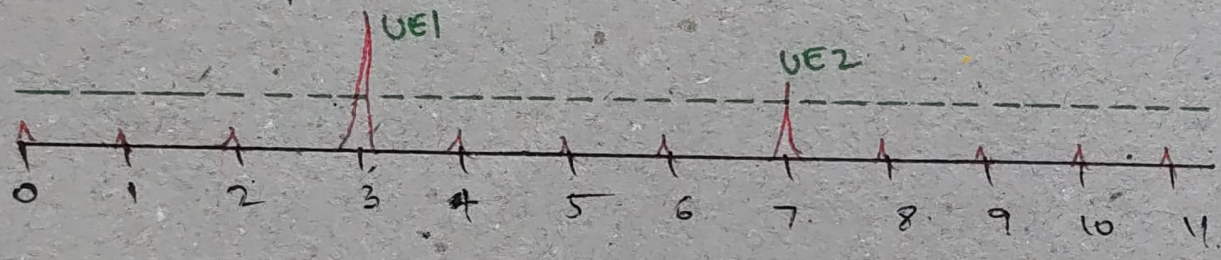
Since, the gNB knows m_0 and m_{cs} value, it subtracts the same from the received signal (high correlation) and finds the m_{cs}.

Once m_{cs} is calculated, the gNB will find out what was transmitted, using the table (referred above).

Say for example, gNB finds $m_{cs} = 3$, and it knows that, its a 2-bit HARQ, hence the table shows [N, A], which means, for first TB, it is NACK
for second TB, it is ACK

This is how the gNB gets the data from UE.

Note that, in case of 2 UE's multiplexed, the correlation output looks like below.



And, gNB gets the data separately for UE1 and UE2.

PUCCH Format 1 : Signal Generation and Resource Mapping

Lets look into the basic equations that we have, to generate the sequence.

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

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

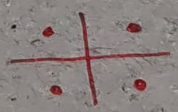
$$m = 0,1,\ldots, N_{SF,m'}^{PUCCH,1} - 1$$

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

As we know, PUCCH F1 is ZC sequence based transmission. Here $z(\cdot)$ is the final sequence that is mapped to the resources, which is a function of $w_i(m)$ and $y(n)$.

$$y(n) = d(0) \cdot r_{u,v}^{(\alpha,\delta)}(n)$$

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

$y(n)$ is the ZC sequence, where $d(0)$ is the information bit where the information is encoded. As we know, PUCCH F1 carries HARQ (ACK/NACK) or SR. If it is 1 bit information, we use BPSK  . If it is 2 bit information, then we use QPSK  modulation.

So, at the end, $d(0)$ is a single sample, based on either it is 1 bit or 2 bit information. The Root Sequence $r_{u,v}^{(\alpha,\delta)}(n)$ is multiplied with the information sample $d(0)$ to generate the sequence $y(n)$.

Now, $r_{u,v}^{(\alpha,\delta)}(n)$ is defined as below.

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

This ZC sequence $r_{u,v}^{(\alpha,\delta)}(n)$ has two parts.

$e^{j\alpha n}$ is to provide cyclic shift

$\bar{r}_{u,v}(n)$ is the ZC sequence

$$\alpha_l = \frac{2\pi}{N_{sc}^{RB}}\left(\left(m_0 + n_{cs}(n_{s,f}^{\mu}, l+l')\right) \bmod N_{sc}^{RB}\right) \qquad N_{sc}^{RB} = 12$$

α_l is a function of Initial cyclic shift (m_0) and $n_{cs}(\cdot)$. m_0 takes values from 0 to 11, and this can be different for different UEs, which is used to provide different cyclic shifts to multiplex multiple UEs.

n_{cs} is a function of Slot number and Symbol number.

$$n_{cs}(n_{s,f}^{\mu}, l) = \sum_{m=0}^{7} 2^m c(8N_{symb}^{slot} n_{s,f}^{\mu} + 8l + m)$$

n_{cs} is a function of PN sequence $c(\cdot)$, and c is a function of Slot number and Symbol number. So basically, with this PN sequence, we take 8 randomly generated bits, convert them to an integer and assign it to n_{cs}. So, basically n_{cs} is a Pseudo-random cyclic shifts, as a function of Slot and frame number. So, for each slot, and for each symbol, n_{cs} will be different (random).

With n_{cs} and m_0, we'll get $\alpha_l = \frac{2\pi}{12}$ (0 to 11)

Now, coming to $\bar{r}_{u,v}(n)$, there are 2 variables here. $u \rightarrow$ provides Group Hopping

$v \rightarrow$ provides Sequence Hopping.

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

For PUCCH Format 0 and Format 1, sequence hopping is disabled ($v = 0$) always. Only Group hopping is allowed. Basically, u is the Root Sequence Index. So, there are 30 sequences, out of which we pick one. The way we pick one out of 30 is defined as below.

$$f_{gh} = 0$$
$$f_{ss} = n_{ID} \bmod 30$$
$$v = 0$$

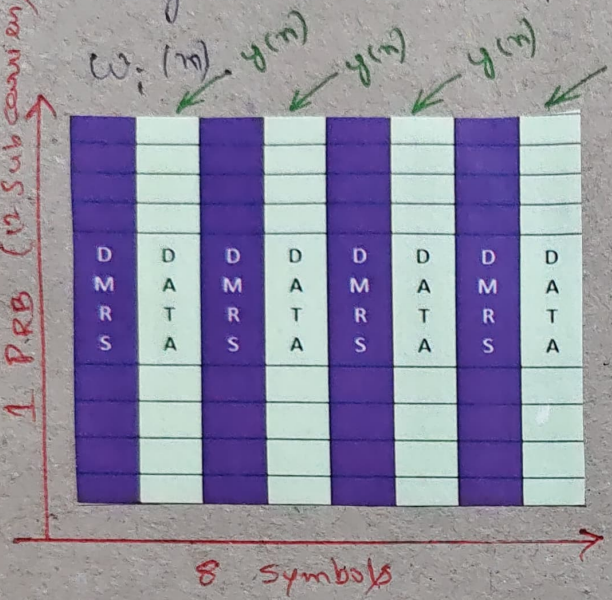
$$f_{gh} = \left(\sum_{m=0}^{7} 2^m c\left(8\left(2n_{s,f}^{\mu} + n_{hop}\right) + m\right)\right) \bmod 30$$
$$f_{ss} = n_{ID} \bmod 30$$
$$v = 0$$

If Group hopping is disabled ($f_{gh} = 0$), then u is a function of only f_{ss}. n_{ID} is the Hopping ID or it can be cell-ID also. Lets take n_{ID} as cell ID, then for each cell, there can be different Root Sequences. So, different Root Sequences can be given to different neighboring cells. This is to ensure that, if PUCCH allocation is same across the neighboring cells, then one cell's sequence appears random to the other cell.

If Group hopping is enabled, f_{gh} is defined as a function of PN sequence $c(i)$, which in turn a function of slot number and n_{hop}.

If intra-slot frequency hopping is enabled, then n_{hop} will take two values (0 and 1) i.e. for first hop, $n_{hop} = 0$; for second hop, $n_{hop} = 1$. So, for two different hops, we can have two different sequences.

This is how we pick u and v, using which we compute $r_{u,v}^{(\alpha,\delta)}(n)$, using which we compute $y(n)$, using which we compute $z(\cdot)$ by multiplying it with $w_i(n)$.



→ This is how normally a PUCCH Format 1 allocation in a PRB looks like.

→ Format 1 takes 1 PRB and 4 to 14 OFDM Symbols.

→ The allocation starts with DMRS (first symbol)

→ Every alternate symbol is DMRS, between which we have Data.

→ In this example, there are 8 symbols, out of which 4 symbols are for DMRS.

→ Now, 4 different sequences 'y(n)' are generated for 4 different data symbols.

Now, these sequences 'y(n)' are multiplied with $w_i(n)$, which provides the Orthogonal Code Cover. (ii) we use the sequence 'y(n)' to multiplex multiple UEs in this resource. (ii) Generate different sequences for different UEs from single y(n) sequence.

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

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

Orthogonal Sequences, $w_i(m) = e^{\frac{j2\pi\phi(m)}{N_{SF,m'}^{PUCCH,1}}}$

where, $N_{SF,m'}^{PUCCH,1}$ ⟶ No. of symbols allocated for PUCCH

For example, $N_{SF,m'}^{PUCCH,1} = 4$

$\Rightarrow w_i(m) = e^{\frac{j2\pi\phi(m)}{4}}$

In this case, for one PUCCH data symbol, 4 different orthogonal sequences are defined with $i = 0, 1, 2, 3$. UE will pick one of them.

i = 0	i = 1	i = 2	i = 3
0	0	0	0
0	2	0	2
0	0	2	2
0	2	2	0

ϕ

$w_i(m)$ ⟶

UE1 UE2 UE3 UE4

i = 0	i = 1	i = 2	i = 3
1	1	1	1
1	-1	1	-1
1	1	-1	-1
1	-1	-1	1

4 different orthogonal sequences.

Lets take $i = 1$. $\Rightarrow w_i(m) = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}$

So, $Z(\cdot) = w_i(m) \cdot y(m)$

$= \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix} \cdot y(m)$

This is how $w_i(m)$ is multiplied with the sequence $y(m)$. for UE 2. For other UEs, different $w_i(m)$ is multiplied with the sequence $y(m)$.

Lets now understand how symbols are allocated between DMRS and Data.

PUCCH length, $N_{symb}^{PUCCH,1}$	$N_{SF,m'}^{PUCCH,1}$ No intra-slot hopping $m'=0$	Intra-slot hopping $m'=0$	Intra-slot hopping $m'=1$
4	2	1	1
5	2	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

No. of Symbol in PUCCH

As we know, PUCCH can take anywhere between 4 to 14 symbols. Out of these, there will be some DMRS and some DATA symbols. And, WKT, DMRS takes first OFDM symbol in PUCCH allocation, and it goes on every alternate symbols. So, if

* PUCCH length → Even Number → DMRS and DATA will take exact half symbol.

Example: If PUCCH length = 4, then
No. of DMRS symbols = 2
No. of Data symbols = 2

* PUCCH length → Odd Number → DMRS takes one symbol higher than DATA, always. Coz DMRS starts first.

Example: If PUCCH length = 5, then
No. of DMRS symbols = 3
No. of Data symbols = 2

This is how the DMRS and DATA symbol distribution occurs.

If there is no intra-slot hopping, then the No. of Data Symbols is given in (Column 2) of the table. For example, if PUCCH length = 11, then 5 symbols are for DATA, and remaining 6 symbols are for DMRS.

If intra-slot hopping is enabled, then how the symbol distribution across the Hop occurs? Lets understand this with an example.

→ Intra-Slot hopping disabled.

PUCCH length = 13

No. of DMRS Symbols = 7

No. of DATA Symbols = 6

→ Intra-Slot hopping enabled.

PUCCH length = 13

In the first hop ($m' = 0$),

No. of DMRS Symbols = 3

No. of DATA Symbols = 3

In the second hop ($m' = 1$),

No. of DMRS Symbols = 4

No. of DATA Symbols = 3

Note: In this example, the PRBs are shown consecutive. But in actual, it need not be.

This is how we read this Table.

Now, lets get back to the very first equation. To ease the simplification, lets assume intra-slot frequency hopping ($m' = 0$) is disabled.

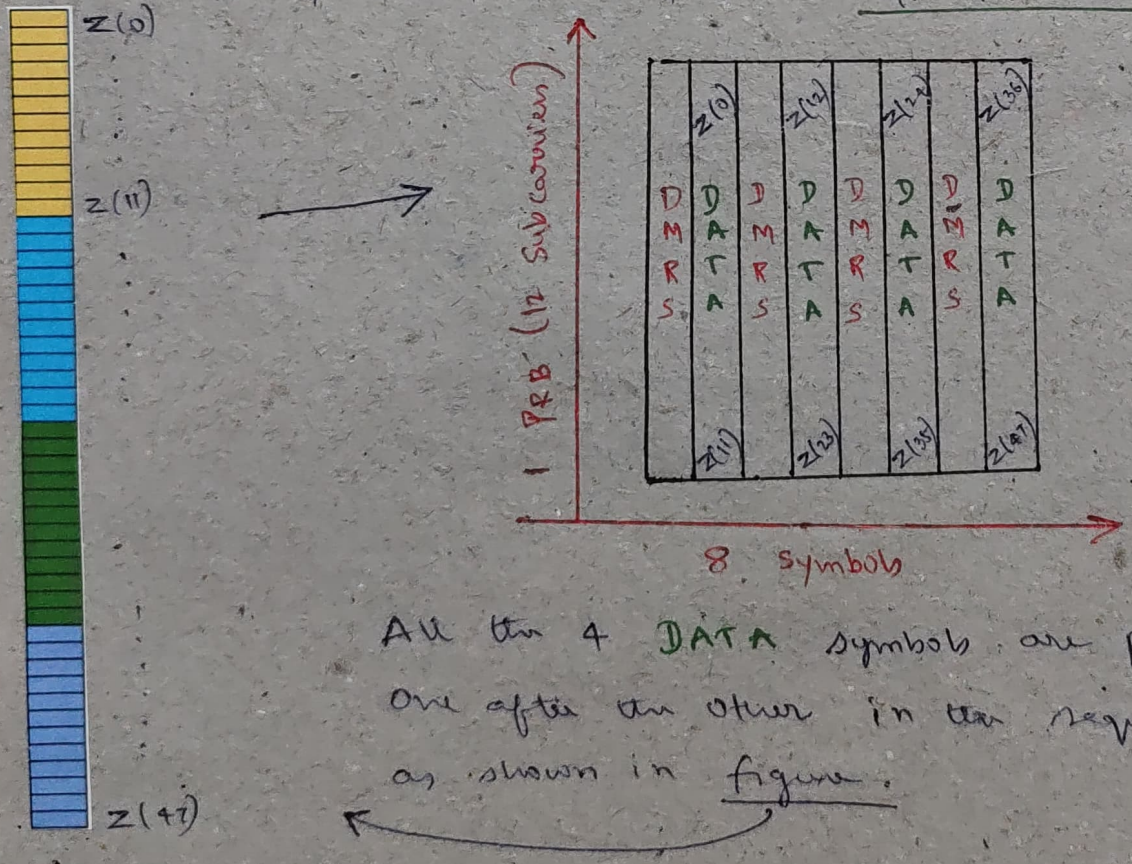
$$\Rightarrow z(0 + m \cdot 12 + n) = w_i(m) \cdot y(n)$$

where, $n \rightarrow 0$ to 11

$w_i(m) \rightarrow$ depends on the No. of symbols allocated for DATA for PUCCH Format 1

So, the Output $z(\cdot)$ is appended one after the other, to create a single sequence.

Lets take our previous example where we have total 8 symbols, out of which 4 for data.

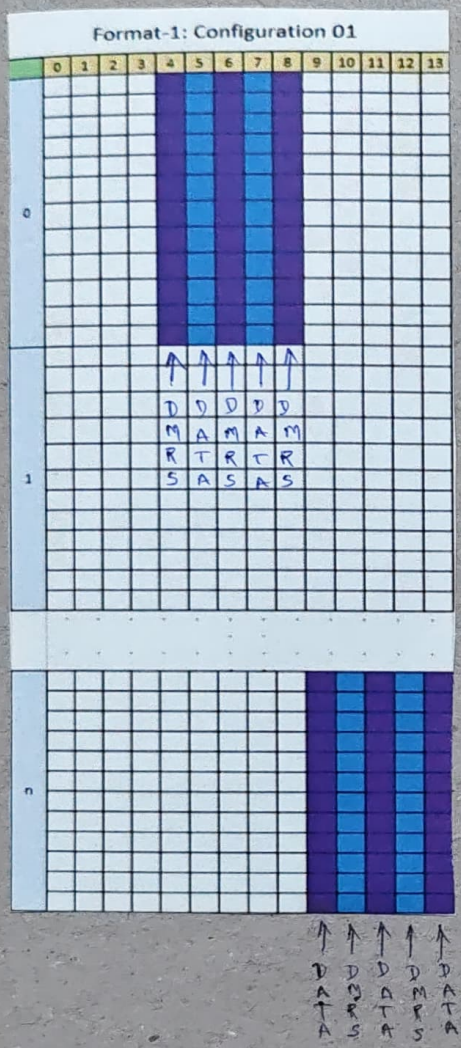


All the 4 DATA symbols are put together one after the other in the sequence $z(\cdot)$ as shown in figure.

Note: Even if Intra-slot hopping is enabled, that that data will append after this sequence $z(47)$.

This is how, the mapping of PUCCH sequence to PRB's is done.

Now, lets see what are the valid PUCCH Format 1 configurations.



→ In this case, frequency hopping is enabled.

→ In the first hop ($m' = 0$),

No. of DMRS symbols = 3

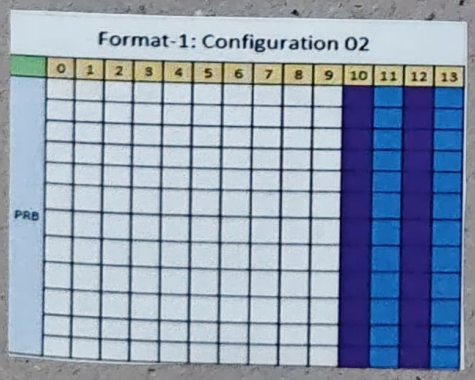
No. of DATA symbols = 2

→ In the second hop ($m' = 1$),

No. of DMRS symbols = 2

No. of DATA symbols = 3

→ PUCCH length = 10.

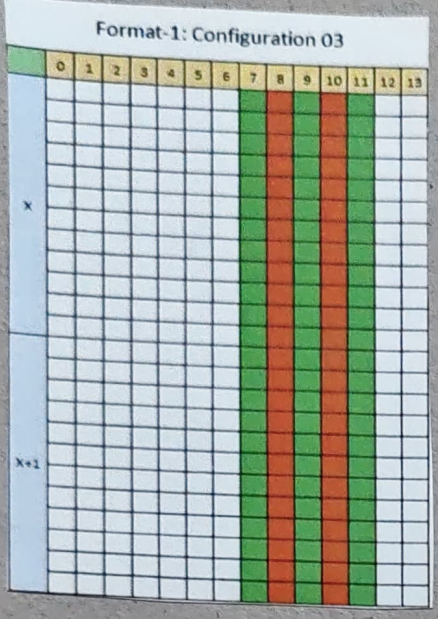


→ In this case, there is no frequency hopping.

→ PUCCH is allocated at the end

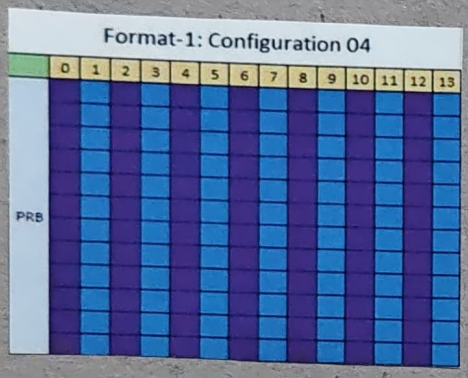
No. of DMRS symbols = 2

No. of DATA Symbols = 2

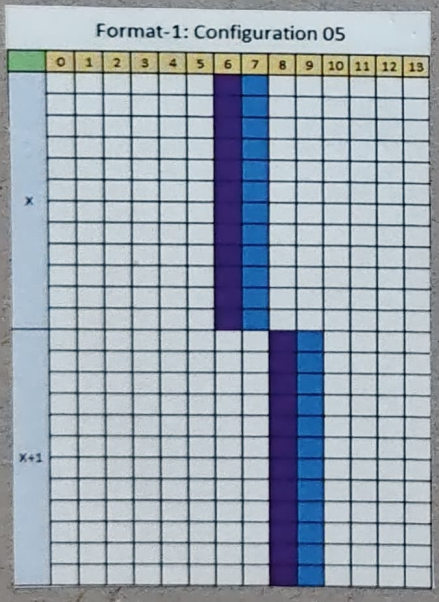


→ In this case, two consecutive PRBs are allocated.

→ This configuration is INVALID, since two PRBs cannot be allocated in a Symbol.



→ In this case, all 14 symbols are allocated for PUCCH F1.



→ In this case, frequency hopping is enabled, where the next hop is in next PRB.

→ PUCCH length = 4

→ In the first hop ($m' = 0$)

No. of DMRS symbol = 1

No. of DATA symbol = 1

→ In the next hop ($m' = 1$)

No. of DMRS symbol = 1

No. of DATA symbol = 1

This is all about PUCCH Format 1 Sequence (DATA) generation and Resource Mapping.

PUCCH Format 1 : DMRS

Lets understand DMRS generation and mapping for PUCCH Format 1.

In the last section, we've seen the Zadoff-chu Sequence and Signal generation for PUCCH F1 (DATA)

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

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

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

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

In this section, we'll see the same for PUCCH F1 (DMRS)

$$z(m' N_{sc}^{RB} N_{SF,0}^{PUCCH,1} + m N_{sc}^{RB} + n) = w_i(m) \cdot r_{u,v}^{(\alpha,\delta)}(n) \qquad \text{DMRS}$$

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

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

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

As we see, the DMRS generation and DATA generation are mostly the same. In both the case, we generate ZC sequence using u, v, α and δ. (with Sequence hopping $v=0$, different cyclic shift for each symbol).

Here, in DMRS sequence generation also, we multiply ZC sequence with Orthogonal code (over $w_i(m)$, to generate the sequence $z(m)$, which is mapped to the resources as follows.

$$a_{k,l}^{(p,\mu)} = \beta_{PUCCH,F} z(m)$$

$$l = 0,2,4,\ldots$$

$z(m)$ is multiplied with the power factor $\beta_{PUCCH,1}$, and simply mapped to $a_{k,l}^{(p,\mu)}$ (Resource mapped output, with frequency index k and time index l). It is explicitly mentioned that, DMRS takes symbols $0, 2, 4, \ldots$ inside the PUCCH allocation.

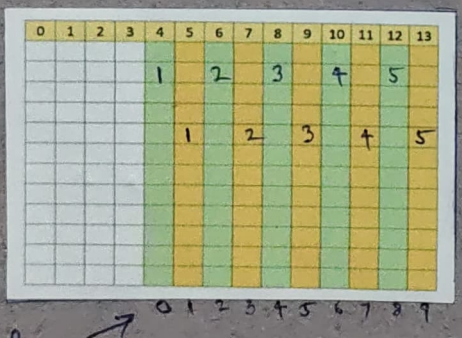
PUCCH length, $N_{symb}^{PUCCH,1}$	$N_{SF,m'}^{PUCCH,1}$		
	No intra-slot hopping	Intra-slot hopping	
	$m'=0$	$m'=0$	$m'=1$
4	2	1	1
5	2	1	1
6	3	1	2
7	3	1	2
8	4	2	2
9	4	2	2
10	5	2	3
11	5	2	3
12	6	3	3
13	6	3	3
14	7	3	4

← DATA

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

← DMRS

There are two Symbol index tables for DATA and DMRS respectively. Both the tables looks more or less same except some places. Lets take an example, where PUCCH length, $N_{symb}^{PUCCH,1} = 10$.



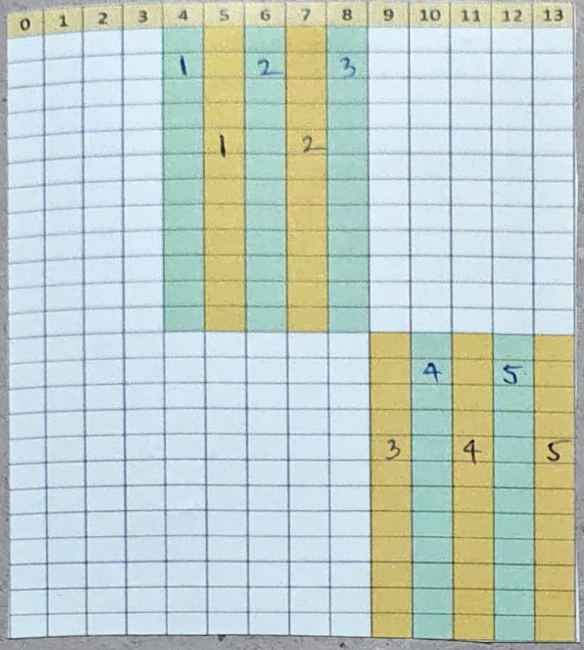
→ Intra-slot hopping disabled

→ No. of DMRS symbols = 5

No. of DATA symbols = 5.

→ DMRS takes symbol index

$l = 0, 2, 4, \ldots$



→ Intra - slot hopping Enabled

→ In the first hop ($m' = 0$)

No. of DMRS symbols = 3

No. of DATA symbols = 2

→ In the next hop ($m' = 1$)

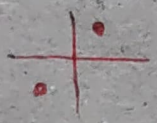
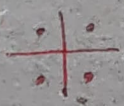
No. of DMRS symbols = 2

No. of DATA Symbols = 3

This is all about PUCCH Format1 Sequence (DMRS) generation and Resource Mapping.

PUCCH Format 1 : UE Multiplexing and Receiver Design

Lets understand how the data is being transmitted (how HARQ / SR encoded) in PUCCH Format 1. We also understand, how many UEs can be multiplexed. And we also understand how Receiver can be designed.

As we have seen in previous sections, if it is 1 bit of information, we do BPSK  modulation. If it is 2 bits of information, we do QPSK  modulation, and the modulated data is multiplied with the ZC sequence, and then transmitted.

Lets see how it is mapped to the resources. Basically, a UE is always allocated with two different resources (2 PRBs), One for HARQ and One for SR, at two different locations in the frequency.

When UE needs to send "1 bit HARQ", then it will do the BPSK modulation, and will send it on the HARQ resources.

When UE needs to send "2 bit HARQ", then it will do the QPSK modulation, and will send it on the HARQ resources.

When UE needs to send "SR", then it will do the BPSK modulation, and will send it on the SR resources. (if SR is +ve)

When UE needs to send "1 bit HARQ + SR" on the same slot; then it will do BPSK modulation, and will send the 1 bit HARQ on the SR resources (if SR is +ve). If SR is -ve, then the 1 bit HARQ will be sent on HARQ resources.

When UE needs to send "2 bit HARQ + SR" on the same slot; then it will do QPSK modulation, and will send the 2 bit HARQ on the SR resources (if SR is +ve). If SR is -ve, then the 2 bit HARQ will be sent on HARQ resources.

So, for gNB, detecting "1 bit HARQ" and "2 bit HARQ", without SR is simple. It will directly look into HARQ resources, and identify the ACK / NACK.

If UE has sent both HARQ and SR, say "1 bit HARQ + SR" and "2 bit HARQ + SR", in this case, gNB will look into both HARQ and SR resources and identify where the data (ACK/NACK) is transmitted. If the data is found on HARQ resources, then gNB considers that SR is not sent by UE. If data is found on SR resources, then gNB considers that UE is requesting SR.

This is how the Data is encoded. (ACK/NACK/SR).

Now, lets understand UE multiplexing.

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

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

In the previous section, we've seen the Orthogonal sequences $w_i(m)$, which is based on the NO. of PUCCH Data Symbols $N_{SF,m'}^{PUCCH,1}$. Say for example, if there are 7 symbols, then there are 7 orthogonal sequences. This is called Orthogonal Code Cover (OCC) sequences.

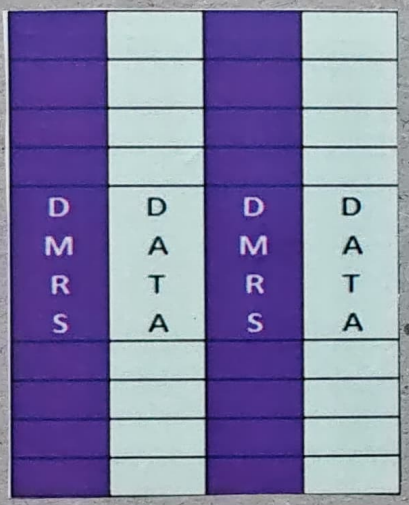
$$\alpha_l = \frac{2\pi}{N_{sc}^{RB}}\left(\left(m_0 + n_{cs}\left(n_{s,f}^{\mu}, l+l'\right)\right) \bmod N_{sc}^{RB}\right)$$

Also, we've seen the Cyclic shift, where m_0 is the Initial shift. This is also used, to multiplex different UEs.

So, both OCC and cyclic shift can be used to multiplex different UEs. Lets take an example and understand how the UE multiplexing can be done.

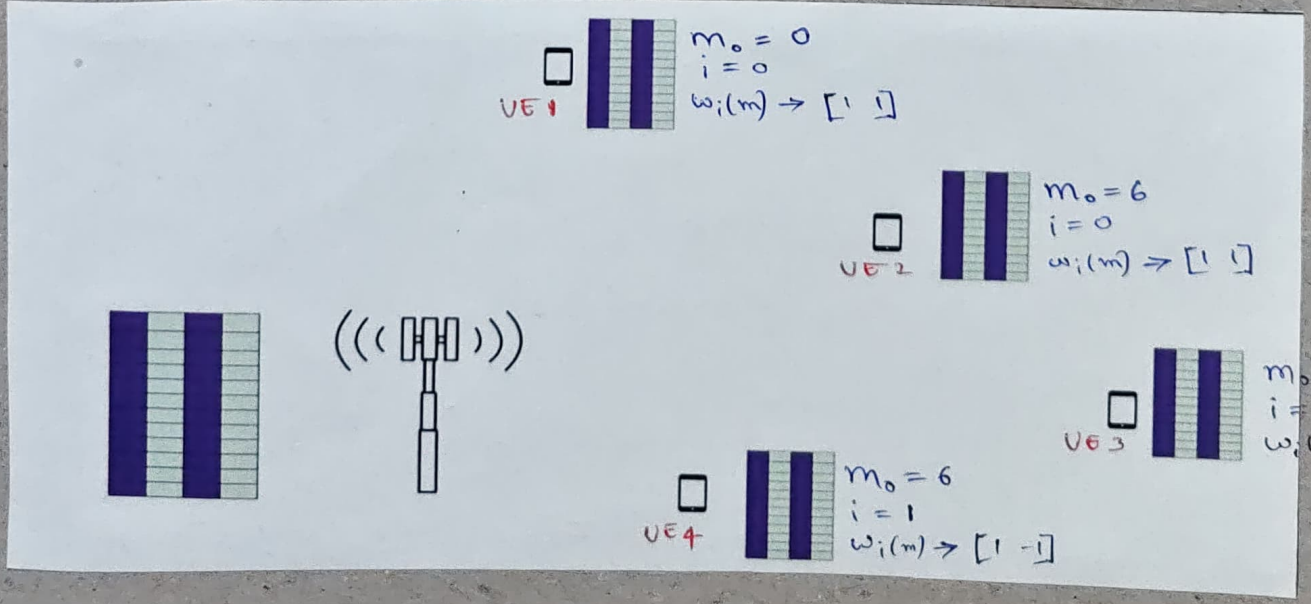
Consider, there are 4 UEs, and we want 2 UEs multiplexed using cyclic shift and 2 UEs multiplexed using OCC and cyclic shift. This way, we'll cover all the different combinations.

> Note: The same rule is applicable when we multiplex more than / less than 4 UEs.



Lets assume, PUCCH is scheduled in 4 symbols. Out of 4, two are DMRS and 2 are DATA Symbols.

Lets consider the scenario as below.



Assume that, for UE1 and UE2, we have signalled the cyclic shifts $m_0 = 0$ and $m_0 = 6$ respectively. With these cyclic shifts, the generated ZC sequence is Uncorrelated. (ii) for different cyclic shifts, the generated signal is Uncorrelated.

Similarly, for UE3 and UE4, we have signaled the cyclic shifts $m_0 = 0$ and $m_0 = 6$.

From Table 6.3.2.4.1-2, For two Symbols (ii) $\left(N_{SF,m'}^{PUCCH,1} = 2\right)$, the orthogonal sequence is [0 0] and [0 1]. Since $w_i(m) = e^{\frac{j 2\pi \phi(m)}{N_{SF,m}^{PUCCH,1}}}$,

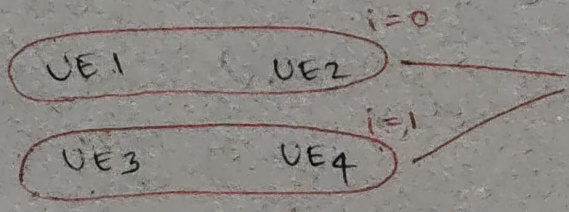
$$[0\ 0] \rightarrow w_i(m) \rightarrow [1\ 1]$$
$$[0\ 1] \rightarrow w_i(m) \rightarrow [1, -1]$$

So, to make it orthogonal, we have signaled UE1 and UE2 with [1 1] (ii) $i = 0$ is signaled to both UE1 and UE2. Whereas we have signaled UE3 and UE4 with [1 -1] (ii) $i = 1$ is signaled to both UE3 and UE4.

Thus, the generated signals of UE1 and UE2 are Uncorrelated with each other. Similarly, the generated signals of UE3 and UE4 are Uncorrelated with each other.

UE1 and UE2 has one OCC.

UE3 and UE4 has different OCC.



This way, UE1 and UE3 are orthogonal,
UE2 and UE3 are orthogonal.
UE1 and UE4 are orthogonal.
UE2 and UE4 are orthogonal.

This way, we provide orthogonality and also uncorrelated signals.

If we want to multiplex more UEs, we need to provide more cyclic shifts. (α_l).

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

Cyclic shift (α_l) can take 12 values (0 to 11).

(ii) $\alpha_l = \frac{2\pi}{12}$ (0 to 11)

So, based on 12 different α_l, 12 different uncorrelated sequences can be generated.

And, if we use 7 symbols of data (i) $N_{SF,m'}^{PUCCH,1} = 7$,
(ii) PUCCH length = 14, out of 14, 7 are DATA., we can generate 7 OCC as given in Table 6.3.2.4.1-2.
And for each OCC, there are 12 sequences. So, in total 84 UEs (12 x 7) are multiplexed. (ideally).
But, based on different Channel conditions, the actual number of UEs that can be multiplexed is quite less.

If we want to keep all UEs on different OCC, then we can provide the same cyclic shift (α_l) and multiplex 7 UEs, where all of them are orthogonal.

If we want to keep all UEs on different cyclic shifts (α_i), then we can provide the same OCC, and multiplex 12 UEs, i.e. all of them are orthogonal.

(ii) 12 UEs sending uncorrelated sequences.

Now, the gNB will receive data of all 4 UEs. (ie) the resource will have the summed up data of all UEs (UE1 + UE2 + UE3 + UE4). The gNB need to utilize the DMRS signals, OCC sequence and cyclic shift, to decode and find out what was transmitted from each UE.

Now, lets understand how the receiver works.

UE, in order to generate the sequence, first it picks the Root sequence (u), then it picks the cyclic shift (m_0) and generate ZC sequence. Then, on top of that, it applies Orthogonal Cover code (OCC), then does IFFT and CP insertion, and then transmit. This is done by all 4 UEs, where UEs have picked different cyclic shifts ($m_0 = 0/6$) and different OCC ($i = 0/1$).

At the gNB, it has to do reverse of the above. First it receives the transmitted data (here PRB with 4 symbols (2 for DMRS, 2 for DATA)), and performs channel Estimation, which is proprietary. So, every vendor can implement different complex algorithms for a better receiver design. We have two symbols for

DMRS (i) $[d_1]$ and $[d_2]$, so we remove the OCCs.

The receiver already knows what OCC was applied at the transmitter side. So, at the receiver, we remove the OCCs from $[d_1]$ and $[d_2]$ by multiplying the conjugate of OCCs with the DMRS symbols.

Once the OCCs are removed, we will get two channels H_1 and H_2. We've seen there are two groups. H_1 is for Group1 (for UE1 and UE2)

H_2 is for Group2 (for UE3 and UE4).

From H_1, we need to find the channel estimate for UE1 and UE2. From H_2, we need to find the channel estimate for UE3 and UE4. Since the receiver knows u, m_0 and n_{cs}, it generates the base ZC sequence. The receiver then multiplies the conjugate of the base ZC sequence with H_1 to get the channels of UE1 and UE2.

(i) H_1^{UE1} and H_1^{UE2} (∵ UE1 and UE2 have sent uncorrelated DMRS)

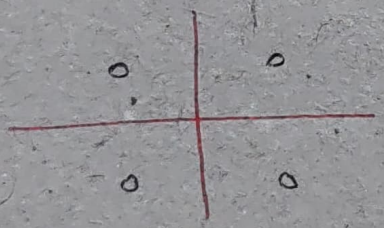
Similarly, the receiver multiplies the conjugate of the base ZC sequence with H_2 to get the channels of UE3 and UE4.

(i) H_2^{UE3} and H_2^{UE4} (∵ UE3 and UE4 have sent uncorrelated DMRS)

So, now we have the channels between each UEs and the BS. (ii) we have estimated the channels between BS and UE1, BS and UE2, BS and UE3, BS and UE4.

Now, our goal is to get the data. From the DATA symbols also, we remove the OCC, separate the data from each UE, using the base sequence generated by u, m_0 and n_{cs} (Same step as in DMRS).

Now, we have separated the data from each UE, also we have the DMRS for each UE, the receiver performs Equalization (removes the channel effect) and then we'll get the Transmitted symbols.



This is how the receiver works.

Note:

When we are multiplexing the UEs, we need to decide whether, we want to multiplex them using different OCCs or we want to multiplex them using different cyclic shifts.

Both are two different methods and may give two different results.