TSG-RAN Working Group 1 meeting No. 6
July 13-16, Espoo, Finland

TSGR1-899/99

Agenda item:

Source: InterDigital, Nokia

Title: Text proposal for RACH preambles

Document for: Decision

---

1 Introduction

Nokia has proposed long codes to be used for RACH preambles. InterDigital has proposed, and TSG RAN WG1 has accepted differential preambles additionally as a working assumption. This document provides corresponding text proposal for 25.2 series of 3GPP specifications.

2 Text proposal for 25.211

The following text is proposed to replace the subclause 5.2.2.1.2 in 25.211:

5.2.2.1.2 RACH preamble part

The preamble part of the random-access burst consists of a a 4096 chip long binary code generated the same way as the binary valued codes used to generate the long complex valued scrambling code for dedicated channel. The code is modulated by *signature* of length 16. The symbols in the signature are complex and of the form~~symbols~~ ±~~1~~(1+j~~). Each preamble symbol is spread with a 256 chip real Orthogonal Gold code.~~ The resulting spreading factor is 256 for the preamble part. There are two sets of signatures, each with a total of 16 different signatures. One set, the coherent signature set, is~~,~~ based on the Orthogonal Gold code set of length 16 (see 25.213 for more details). The second set, the differentially encoded set, is obtained by differentially encoding the first set.

3 Text proposal for 25.213

The following text is proposed to replace the subclauses 4.3.3.1, 4.3.3.2, and 4.3.3.4: Based on the Ad Hoc 3 agreements The idea was that the primary signatures are intended for use with uplink random access receivers that perform coherent preamble detection, which is optimal for low Doppler. Respectively the differentially encoded signatures are intended for use with uplink random access receivers that perform differential decoding, which is optimized for high Doppler.

4.3.3.1 Preamble codes

The spreading code for the preamble part is cell specific and is broadcast by the base station. More than one preamble code can be used in a base station if the traffic load is high. The preamble codes must be code planned, since two neighbouring cells should not use the same preamble code.

The code generating method is the same as for dedicated channel. Only the first 4096 chips of the code are used for preamble spreading with the chip rate of 3.84 Mchips/s. The long code c_1 for the phase component is used directly on both in phase and quadrature branches wihout offset between the branches. <Note: The PAR reduction method is FFS>.

The code used is a real-valued 256 chip Orthogonal Gold code. All 256 codes are used in the system.

The code sequences are constructed with the help of two binary *m*-sequences of length 255, *x*, and *y*, respectively. The *x* sequence is constructed using the polynomial $1+X^2+X^3+X^4+X^8$. The *y* sequence is constructed using the polynomial $1+X^3+X^5+X^6+X^8$.

Let $n_7 \ldots n_0$ be the binary representation of the code number *n* (decimal) with n_0 being the least significant bit. The *x* sequence depends on the chosen code number *n* and is denoted x_n in the sequel. Furthermore, let $x_n(i)$ and $y(i)$ denote the *i*:th symbol of the sequence x_n and *y*, respectively

The *m*-sequences x_n and *y* are constructed as:

Initial conditions:

$x_n(0)=n_0$, $x_n(1)= n_1$, $\ldots = x_n(6)= n_6$, $x_n(7)= n_7$

$y(0)=y(1)= \ldots =y(6)= y(7)=1$

Recursive definition of subsequent symbols:

$x_n(i+8) = x_n(i+4) + x_n(i+3) + x_n(i+2) + x_n(i)$ modulo 2, $i=0,\ldots, 246$,

$y(i+8) = y(i+6)+ y(i+5)+ y(i+3)+y(i)$ modulo 2, $i=0,\ldots, 246$.

The definition of the *n*:th code word follows (the left most index correspond to the chip transmitted first in each slot):

$C_{RACH,n} = < 0, x_n(0)+y(0), x_n(1)+y(1), \ldots, x_n(254)+y(254) >$,

All sums of symbols are taken modulo 2.

The preamble spreadingcode is described in Figure 8.

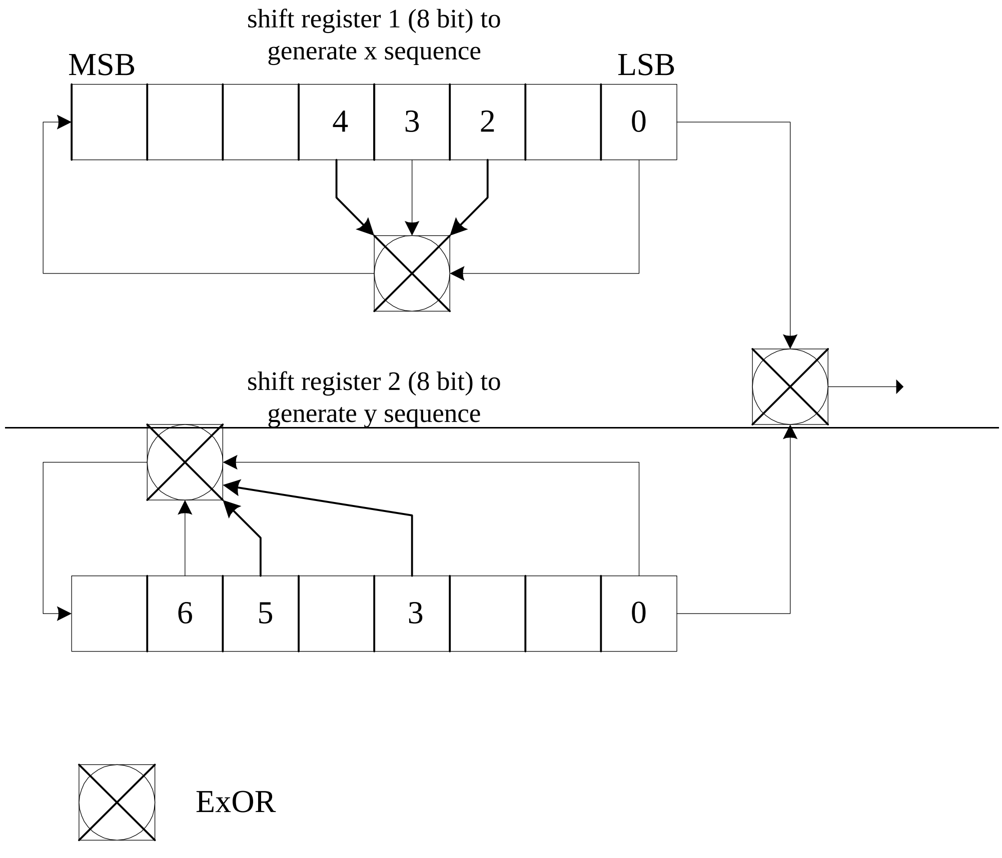

Figure 8. Preamble spreadingcode generator

Note that the code words always start with a constant '0' symbol.

~~Before modulation and transmission these binary code words are converted to real valued sequences by the transformation ‘0’ -> ‘+1’, ‘1’ -> ‘-1’.~~

4.3.3.2 Preamble signature

There shall be two sets of signatures; Primary Signatures and Differentially Encoded signatures. Each Base Station may be designed to support either one of the signature sets and shall transmit the choice of signature set on its Broadcast Channel. All UEs shall support both signature sets and, when associated with a given Base Station, shall use the signature set indicated on that Base Station's Broadcast Channel.

4.3.3.2.1 Primary Signatures

The preamble part carries one of 16 different orthogonal complex signatures of length 16, $<P_0, P_1, ..., P_{15}>$. The signatures are based on a set of Orthogonal Gold codes of length 16 and are specified in Table.

~~Note: WG1 has accepted differential preambles additionally as a working assumption.~~

	Preamble symbols															
Signature	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}
1	A	A	A	-A	-A	-A	A	-A	-A	A	A	-A	A	-A	A	A
2	-A	A	-A	-A	A	A	A	-A	A	A	A	-A	-A	A	-A	A
3	A	-A	A	A	A	-A	A	A	-A	A	A	A	-A	A	-A	A
4	-A	A	-A	A	-A	-A	-A	-A	-A	A	-A	A	-A	A	A	A
5	A	-A	-A	-A	-A	A	A	-A	-A	-A	-A	A	-A	-A	-A	A
6	-A	-A	A	-A	A	-A	A	-A	A	-A	-A	A	A	A	A	A
7	-A	A	A	A	-A	-A	A	A	A	-A	-A	-A	-A	-A	-A	A
8	A	A	-A	-A	-A	-A	-A	A	A	-A	A	A	A	A	-A	A
9	A	-A	A	-A	-A	A	-A	A	A	A	-A	-A	-A	A	A	A
10	-A	A	A	-A	A	A	-A	A	-A	-A	A	A	-A	-A	A	A
11	A	A	A	A	A	A	-A	-A	A	A	-A	A	A	-A	-A	A
12	A	A	-A	A	A	A	A	A	-A	-A	-A	-A	A	A	A	A
13	A	-A	-A	A	A	-A	-A	-A	A	-A	A	-A	-A	-A	A	A
14	-A	-A	-A	A	-A	A	A	A	A	A	A	A	A	-A	A	A
15	-A	-A	-A	-A	A	-A	-A	A	-A	A	-A	-A	A	-A	-A	A
16	-A	-A	A	A	-A	A	-A	-A	-A	-A	A	-A	A	A	-A	A

Table 5. Preamble signatures. A = 1+j.

4.3.3.2.2 Differentially Encoded Preamble signature

The preamble part carries one of 16 different complex signatures of length 16, <P_0, P_1, ..., P_{15}>. The signatures are the result of performing the following two steps.

First, modify Table 5 by multiplying all symbols by (-1) in any row which starts with -A. This results in Table 6.

Then, differentially encode the rows of Table 6; i.e.

Using R_i; i=0 to 15 as the modified original signature

Using P_i; i=0 to 15 as the differentially encoded signature

$P_0 = A$

for i=1 to 15,

if $R_i=P_{i-1}$, $P_i=A$; else $P_i=-A$

The resulting differentially encoded signature set is shown in Table 7

Table 6. Modified Preamble Signatures

	Preamble symbols															
Signature	R_0	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}	R_{11}	R_{12}	R_{13}	R_{14}	R_{15}
1	A	A	A	-A	-A	-A	A	-A	-A	A	A	-A	A	-A	A	A
2	A	-A	A	A	-A	-A	-A	A	-A	-A	-A	A	A	-A	A	-A
3	A	-A	A	A	A	-A	A	A	-A	A	A	A	-A	A	-A	A
4	A	-A	A	-A	A	A	A	A	A	-A	A	-A	A	-A	-A	-A
5	A	-A	-A	-A	-A	A	A	-A	-A	-A	-A	A	-A	-A	-A	A
6	A	A	-A	A	-A	A	-A	A	-A	A	A	-A	-A	-A	-A	-A
7	A	-A	-A	-A	A	A	-A	-A	-A	A	A	A	A	A	A	-A
8	A	A	-A	-A	-A	-A	-A	A	A	-A	A	A	A	A	-A	A
9	A	-A	A	-A	-A	A	-A	A	A	A	-A	-A	-A	A	A	A
10	A	-A	-A	A	-A	-A	A	-A	A	A	-A	-A	A	A	-A	-A
11	A	A	A	A	A	A	-A	-A	A	A	-A	A	A	-A	-A	A
12	A	A	-A	A	A	A	A	A	-A	-A	-A	-A	A	A	A	A
13	A	-A	-A	A	A	-A	-A	-A	A	-A	A	-A	-A	-A	A	A
14	A	A	A	-A	A	-A	-A	-A	-A	-A	-A	-A	-A	A	-A	-A
15	A	A	A	A	-A	A	A	-A	A	-A	A	A	-A	A	A	-A
16	A	A	-A	-A	A	-A	A	A	A	A	-A	A	-A	-A	A	-A

The orthogonal set of Table 6 can be reconstructed using the following rule:

Using P_i; i= 0 to 15 as the transmitted character and

Q_i; i=0 to 15 as the decoded set

Set $Q_0 = A$

For all i >0, if $P_i=P_{i-1}$ then $Q_i= A$; else $Q_i=-A$

The base station broadcasts which signatures are allowed to be used in a cell.

Table 7. Differentially Encoded Preamble signatures. A = 1+j.

	Preamble symbols															
Signature	P_0	P_A	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}
1	A	A	A	-A	A	-A	-A	A	-A	-A	-A	A	A	-A	-A	-A
2	A	-A	-A	-A	A	-A	A	A	-A	A	-A	-A	-A	A	A	-A
3	A	-A	-A	-A	-A	A	A	A	-A	-A	-A	-A	A	A	-A	-A
4	A	-A	-A	A	A	A	A	A	A	-A	-A	A	A	-A	A	-A
5	A	-A	A	-A	A	A	A	-A	A	-A	A	A	-A	A	-A	-A
6	A	A	-A	-A	A	A	-A	-A	A	A	A	-A	A	-A	A	-A
7	A	-A	A	-A	-A	-A	A	-A	A	A	A	A	A	A	A	-A
8	A	A	-A	A	-A	A	-A	-A	-A	A	A	A	A	A	-A	-A
9	A	-A	-A	A	-A	-A	A	A	A	A	-A	A	-A	-A	-A	-A
10	A	-A	A	A	-A	A	A	-A	-A	-A	A	-A	-A	-A	A	-A
11	A	A	A	A	A	A	-A	A	A	A	-A	-A	-A	A	-A	-A
12	A	A	-A	-A	-A	-A	-A	-A	A	-A	A	-A	-A	-A	-A	-A
13	A	-A	A	A	A	-A	A	-A	-A	A	A	-A	A	-A	-A	-A
14	A	A	A	-A	-A	A	-A	A	-A	A	-A	A	-A	-A	A	-A
15	A	A	A	A	-A	-A	-A	A	A	-A	-A	-A	A	A	A	-A
16	A	A	-A	A	A	-A	-A	-A	-A	-A	A	A	-A	A	A	-A

4.3.3.4 Scrambling code for the message part

In addition to spreading, the message part is also subject to scrambling with a 10 ms complex code. The scrambling code is cell-specific and has a one-to-one correspondence to the spreading code used for the preamble part.

The scrambling codes used are from the same set of codes as is used for the other dedicated uplink channels when the long scrambling codes are used for these channels. The first 256 of the long scrambling codes are used for the random access channel. The phases 4096..42496 of the codes are used for the message part(phases 0..4095 of c1 are used in preamble spreading) with the chip rate of 3.84 Mchips/s.

The generation of these codes is explained in Section 4.3.2.2. The mapping of these codes to provide a complex scrambling code is also the same as for the other dedicated uplink channels and is described in Section 4.3.2.