TSG-RAN Working Group 1 meeting #5
Helsinki, Filand
July 13-16, 1999

TSGR1#6(99)915

Agenda item: **Adhoc 10**

Source: **SAMSUNG Electronics Co.**

Title: **Multiple-Scrambling Code**

Document for: **Proposal**

Introduction

This paper includes proposals for new Multi-Scrambling code assignment and generation method. Ericsson proposed Multi-Scrambling code in the contribution TSGR1#5(99)724. In that proposal, each cell can use a primary scrambling code and 511 secondary scrambling codes for solving channelization code limitation.

However, we see some problem of the current method in terms of complexity. In this contribution, we propose several new methods to assign and generate multi-scrambling code reducing the complexity. Moreover, the new scrambling code structure is efficient for minimizing the signaling information.

Problems of the current method and text

The simple way to implement multi-scrambling code is to have multiple independent scrambling code generator. However, it is well kown that we can avoid this by using masking function, and have a considerable complexity decrease. Even though there was not enough information in TSGR1#5(99)724, we think Ericson tried to define mapping between the number and real code, and make it possible to do masking operation since it is necessary to know phase offsets between codes to calculate a masking function. The structure might be like figure 1 in the next page.

Unfortunetely, the mapping rule is not clear at all in the current text. So, the following definition is supposed to be included in the current text.

number 0 : $x_n(0)= x_n(1)=, \ldots =x_n(16)= x_n(17)=0$

number 1: $x_n(0)= x_n(1)=, \ldots =x_n(16)= 0 , x_n(17)=1$

And number N is the (N-1) phase offset of number 1.

In the current text, there is no idea to know what is the meaning of number 0, 1, and so on. So, appropriate description like above is necessary to avoid a confusion and calculate phase offset to make a masking operation possible.

We also need to limit the maximum number of secondary scrambling code to reduce signaling overhead. Based on the capacity analysis, it is very hard to expect that more than two secondary scrambling code is used because breaking orthogonality is a significant factor to impact capacity. So, we suggest 4 as a maximum, and I think we have a enough insurance and can reduce signaling overhead.

Beside above general comments, there are following problems in the current method.

1. For the primary scrambling code 0, there is no way to implement using masking function because sequence consist of all zeros.
2. Since there is no simple rule to find a masking function in the cirrent scheme, BS need to calculate every mask function for secondary scrambling codes in the set of secondary scrambling code corresponding to the cell-specific primary scrambling code, and MS need to calculate every mask function for all secondary scrambling code. This is not desirable at all.
3. Based on 2, complexity of a masking function seems to have a problem. Since have a simple masking function to reduce complexity, and find phase offset easily, a arbitary masking function gives us some complexity increase.

From now on, we propose new 3 options to mitigate above problems.

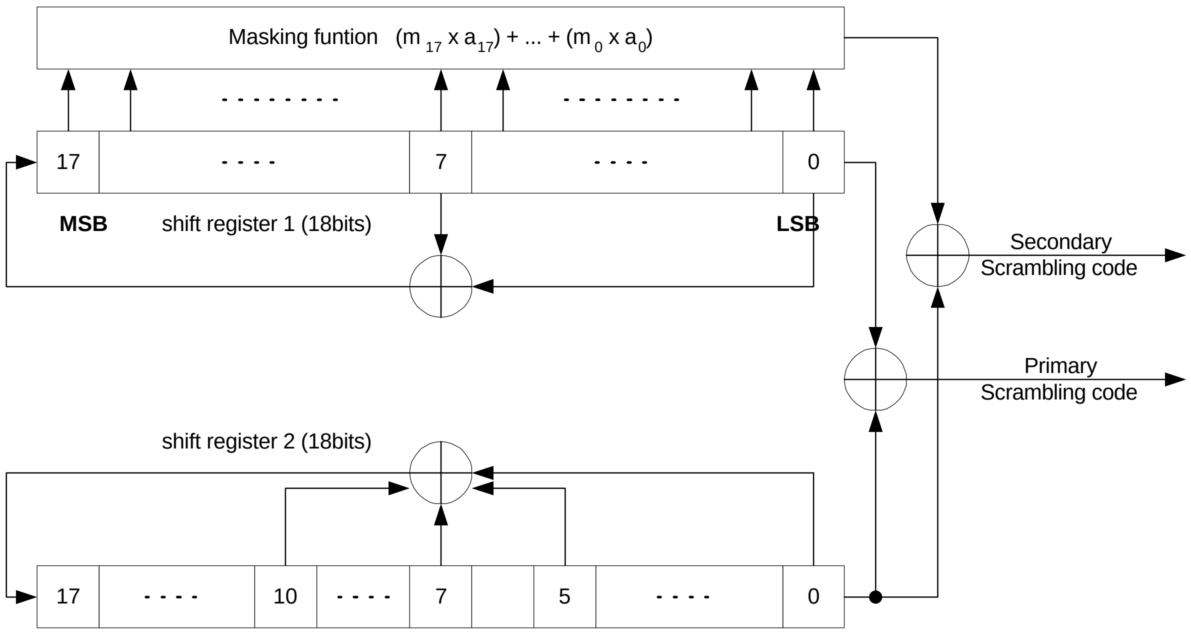

Masking funtion $(m_{17} \times a_{17}) + \ldots + (m_0 \times a_0)$
17
- - - -
7
- - - -
0
MSB
shift register 1 (18bits)
LSB
Secondary Scrambling code
Primary Scrambling code
shift register 2 (18bits)
17
- - - -
10
- - - -
7
5
- - - -
0

Figure 1. Structure of multiple scrambling code generator

Option 1.

The scrambling codes are divided into a set of *primary scrambling codes*, consisting of scrambling code 1, .,.512, and 512 sets of *secondary scrambling codes*, where the i:th set of secondary scrambling codes consists of scrambling codes i*M+513,...,i*M+513+(M-1), where i=0,.,511, and M is the maximum number of secondary scrambling code and less than 511. Following figure illustrates this option.

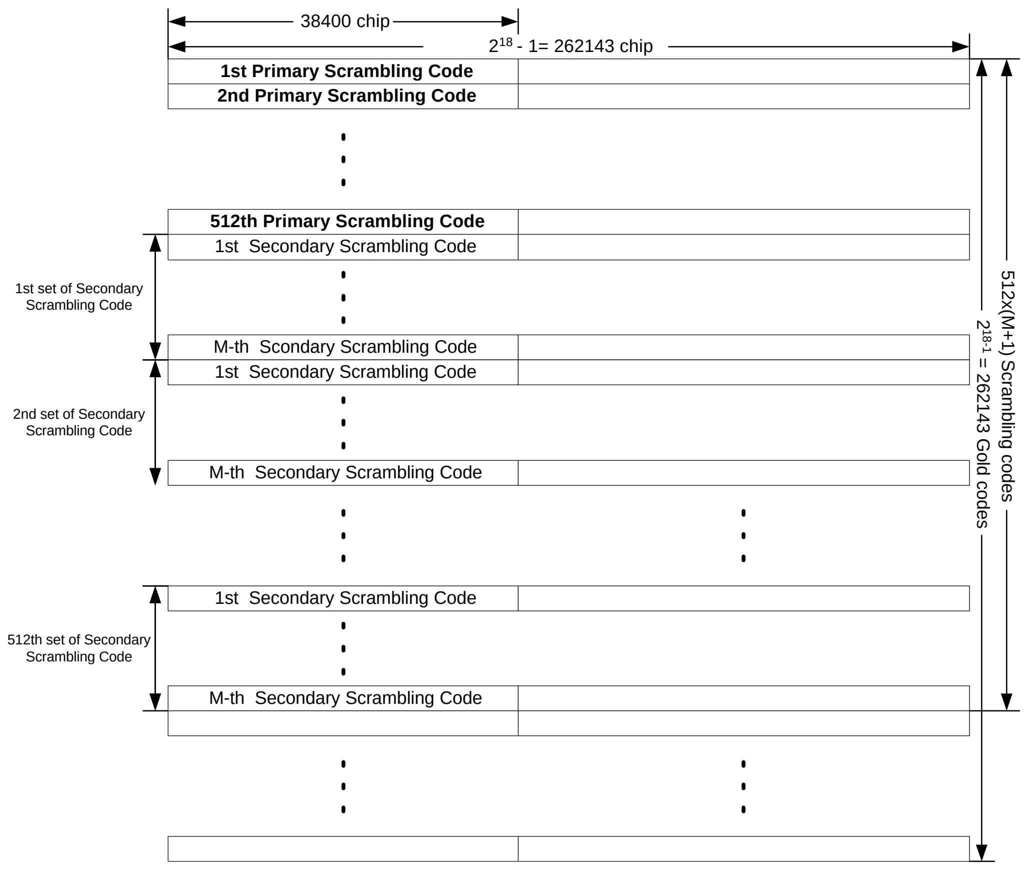

38400 chip
2^{18} - 1= 262143 chip
1st Primary Scrambling Code
2nd Primary Scrambling Code
512th Primary Scrambling Code
1st Secondary Scrambling Code
1st set of Secondary Scrambling Code
M-th Scondary Scrambling Code
1st Secondary Scrambling Code
2nd set of Secondary Scrambling Code
M-th Secondary Scrambling Code
1st Secondary Scrambling Code
512th set of Secondary Scrambling Code
M-th Secondary Scrambling Code
512x(M+1) Scrambling codes
2^{18-1} = 262143 Gold codes

Figure2. illustration of option 1

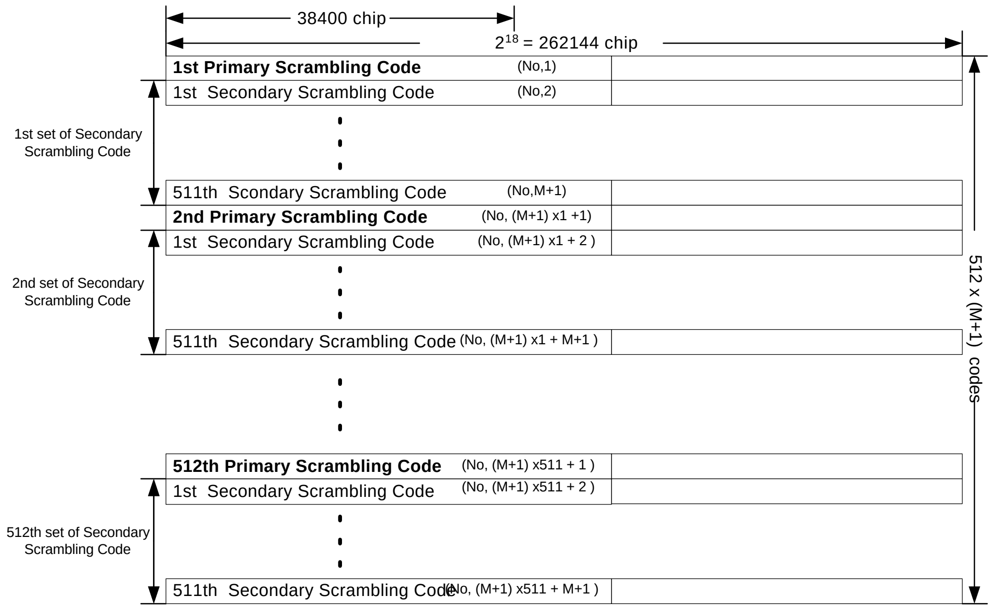
This is the closest mehtod with original Ericsson's proposal. It solves problem 1, but not problem 2 and 3. This method is very inefficient, because we must calculate and store all masking funtion . That is, there is no efficient rule to share the common masking function between all secondary scrambling code set. For this reason, Each cell has a masking function of cell-specific primary scrambling code and M masking functions of the secondary scrambling code corresponding to the primary Scrambling code, while, UE has 512 * M masking functions of the secondary scrambling code corresponding to the primary Scrambling code.

Option 2.

The scrambling codes are divided into a set of *primary scrambling codes*, consisting of scrambling code (M+1)*n+1, and 512 sets of *secondary scrambling codes*, where the i:th set of secondary scrambling codes consists of scrambling codes i*(M+1)+2,...,i*(M+1)+M+1, where n=0, .,.511,and M is the maximum number of secondary codes less than

511. There is a one-to-one mapping between each primary scrambling code and a set of secondary scrambling codes such that i:th primary scrambling code corresponds to i:th set of secondary scrambling codes.

Following figure illustrates this option.



38400 chip
2^{18} = 262144 chip
1st Primary Scrambling Code (No,1)
1st Secondary Scrambling Code (No,2)
1st set of Secondary Scrambling Code
511th Scondary Scrambling Code (No,M+1)
2nd Primary Scrambling Code (No, (M+1) x1 +1)
1st Secondary Scrambling Code (No, (M+1) x1 + 2)
2nd set of Secondary Scrambling Code
511th Secondary Scrambling Code (No, (M+1) x1 + M+1)
512th Primary Scrambling Code (No, (M+1) x511 + 1)
1st Secondary Scrambling Code (No, (M+1) x511 + 2)
512th set of Secondary Scrambling Code
511th Secondary Scrambling Code (No, (M+1) x511 + M+1)
512 x (M+1) codes

Figure 3 illustration of option 2

It is important to notice that secondary scrambling code need to have following common masking functions (Table 1)

Table 1. masking functions to corresponding to n-th secondary scrambling code

Secondary Scrambling Code	masking function for upper m-sequences
1st secondary scrambling code	000000000000000010
2nd secondary scrambling code	000000000000000100
3rd secondary scrambling code	000000000000001000
4th secondary scrambling code	000000000000010000
5th secondary scrambling code	000000000000100000
: : :	: : :

It means that secondary scrambling code generation is possible by no extra process (no addition) and just change input for the generation of secondary scrambling code in the figure 1. If we limit the maximum number of scrambling code as 17, we can such a optimal masking function in terms of complexity. As I mention before, total number of secondary scrambling code does not have to be more than 4. And the above masking function is the same for all secondary scrambling code sets. With this option, we can have a simple rule to calculate primary and secondary scrambling codes efficiently.

It solves all 3 problems described above, and provides very easy and nice implementation. It seems to be a very efficient method to solve all problems. For this reason, Each cell has a masking function of cell-specific primary scrambling code and M masking functions of the secondary scrambling code corresponding to the primary Scrambling code, while, UE has M masking functions of the secondary scrambling code corresponding to the primary Scrambling code. Furthermore, masking funtions are all common and so simple.

Option 3.

Since each code has 2^{18} = 262144 length, it is divided into 6 partitions with 38400 length. The first partition is used for the *primary scrambling codes*, and other 5 partitions are used for the *secondary scrambling codes* correspoding to the primary scrambling code. See Fig 3.

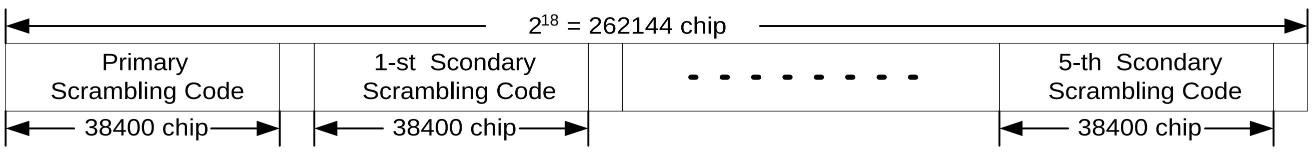
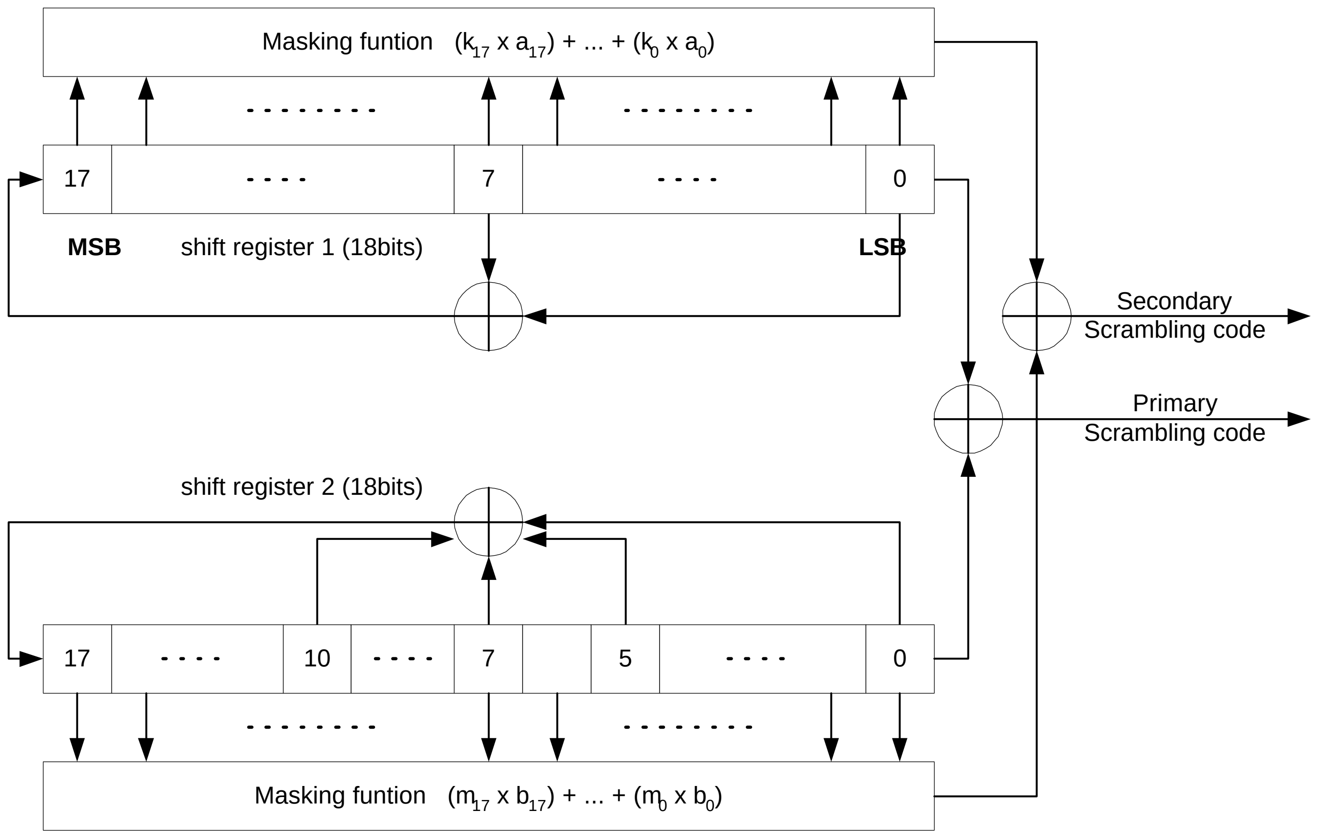


Fig 4. The primary scrambling codes and the secondary scrambling codes

To implement following structuture is needed.



Masking funtion (k17 x a17) + ... + (k0 x a0)
17
7
0
MSB
shift register 1 (18bits)
LSB
Secondary Scrambling code
Primary Scrambling code
shift register 2 (18bits)
17
10
7
5
0
Masking funtion (m17 x b17) + ... + (m0 x b0)

Figure 5 . scrambling code structure for option 3

This structure need the following mask function

Table 1. masking functions to corresponding to n-th secondary scrambling code

Secondary Scrambling Code	masking function for upper m-sequences	masking function for lower m-sequences
1st secondary scrambling code	011011001101011111	001110000101010111
2nd secondary scrambling code	011110010110011110	010111000101100010
3rd secondary scrambling code	101001111000111101	011001100010100101
4th secondary scrambling code	111001001101011010	110100101101101011
5th secondary scrambling code	011011100010010011	000111011011000101

This option inherently has a immunity in problem 1, and very straitforward to problem 2, but it has a complexity increase comparing option 2 (problem 3). However, this method is very natural, and it does not depend on the choice of primary scrambling code meaning that scrambling code 0 also can be selscted.

Conclusion

In this proposal, we described problems of the current scrambling code assignment and generation method and proposed new schemes minimizing the complexity. Since we are using many scrambling codes for primary and secondary to identify the cell and solve the code limitation, we need fine rule to cordinate them nicely. Calculating every mask or Having a complicated mask is not desirable especially for UE.

We enumerate some possible options to improve the current scheme. Among them, we recommend option 2 because this option cordinate nicely primary and secondary scrambling code, and make it very simple to implement secondary scrambling code. All cells use the same simple masking function for secondary scrambling codes, so we don’t have to calculate and memorize for both UE and BS.

In addition, the definition of mapping of primary and secondary scrambling code in the current text should be updated, and also decide total number of secondary scrambling code to reduce signaling overhead. We recommend 4 secondary scrambling codes for each cell.