TSG-RAN Working Group 1 meeting #7
Hanover , Germany
August 30-September 3, 1999

TSGR1#7(99)B87

Agenda Item: Ad Hoc 10

Source: Siemens

Title: A modified generator for Multiple-Scrambling Codes

Document for: decision

1 Introduction

In a previous Tdoc [1] the proponents pointed out the desirability of generating multiple downlink scrambling codes simultaneously using a single basic Gold code generator plus some additional "masking" or linear algebra circuitry. The motivation for this would seem to be that under certain circumstances generation of both primary and secondary downlink scrambling codes would be necessary. This Tdoc cites [2] in which the generation of primary and secondary scrambling codes is defined purely by initialisation value which leads, co-incidentally, to limits on concurrent generators.

The current working assumption, which was introduced in [3] and further described in [4] as an example, is an Gold code generator which can be extended by an algebraic function (or other implementation) to produce a "Q "sequence which is a shift of the "I" sequence by a constant number of chips. This work was optimised by the proponent to minimise complexity of the reference implementation when used as standalone generators. The current text represents a compromise between hardware complexity and initialisation (DSP) complexity. In selecting a scrambling code generation scheme the following points are normally considered:

- hardware (gate) complexity
- estimated power consumption, power saving options
- DSP complexity
- Flexibility
- Re-use or multiple use (e.g. common elements)
- Code quality (does this meet minimum standards? E.g. no bad overlap)

Changing the current scheme at this stage in the standardisation process should only be contemplated with great care and any new proposals should seek to minimise impact on the current schemes.

With this in mind the following method of multiple scrambling code generation is offered as a compromise.

2 A multiple code generator for offset I/Q scrambling sequences.

Consider the figure below which represents the current scheme.

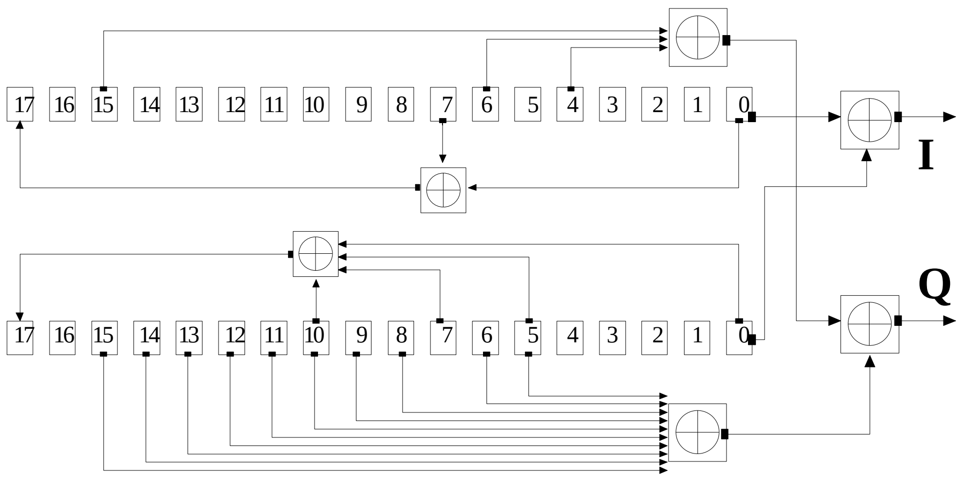

Figure 1 - current downlink scrambling code generator

This is a Gold code generator with conventional output arrangements for I and output logic for Q which represents the shifted versions of the constituent m-sequences for Q. The shifted versions are obtained so that the hardware complexity of the logic to implement the linear algebra is minimised taking the shift as a free parameter.

In order to obtain multiple scrambling codes the proponents of [1] outlined several options and a reference diagram reproduced below. Ideally the "masking function" at the top is as simple as possible.

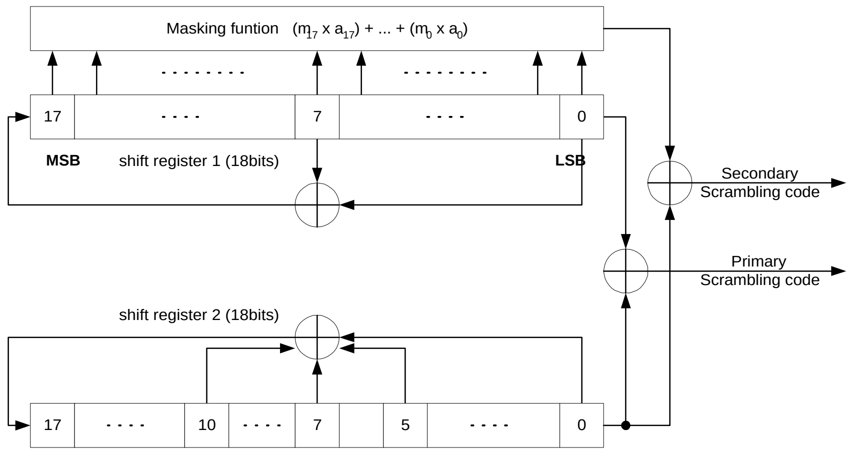
Masking funtion (m17 x a17) + ... + (m0 x a0)
17
7
0
MSB
shift register 1 (18bits)
LSB
Secondary Scrambling code
Primary Scrambling code
shift register 2 (18bits)
17
10
7
5
0

Figure 2 - structure of multiple scrambling code generator (from [1])

In the summary of [1] the proponent presented their preferred option and wrote that a rule was required to co-ordinate any such multiple generation. Also they proposed the use of 4 secondary scrambling codes for each primary code. The proponents of [2] defined 511 secondary codes for each primary. More recent e-mail discussion in Ad Hoc 10 has mentioned approximately 16 secondary codes for each primary. Note that there is no problem with higher layers if layer 1 over provides secondary scrambling codes in the generator. The number of secondary scrambling codes used may be limited by rule.

Now consider the desirable properties of a modified generator as noted:

- minimum impact to current scheme

- should provide approximately 16 secondary codes per primary
- should be neutral between multiple and single generator implementations
- should minimise hardware and DSP complexity

Starting from the current scheme is one way to minimise impact. The core generator can be left alone. Secondary Gold codes may be implemented for the "I" sequences by taking successive taps from the upper LFSR as shown below by example for the first secondary code. The equivalent generation of shifted versions of the (offset) "Q" sequences may be obtained in similar fashion by using a masking function on the lower LFSR (marked B) together with shifted versions of the sequence produced by the upper LFSR (marked A).

Minimising hardware complexity is done by choosing the I/Q shift as a free parameter so as to minimise total complexity in the "masking function" blocks.

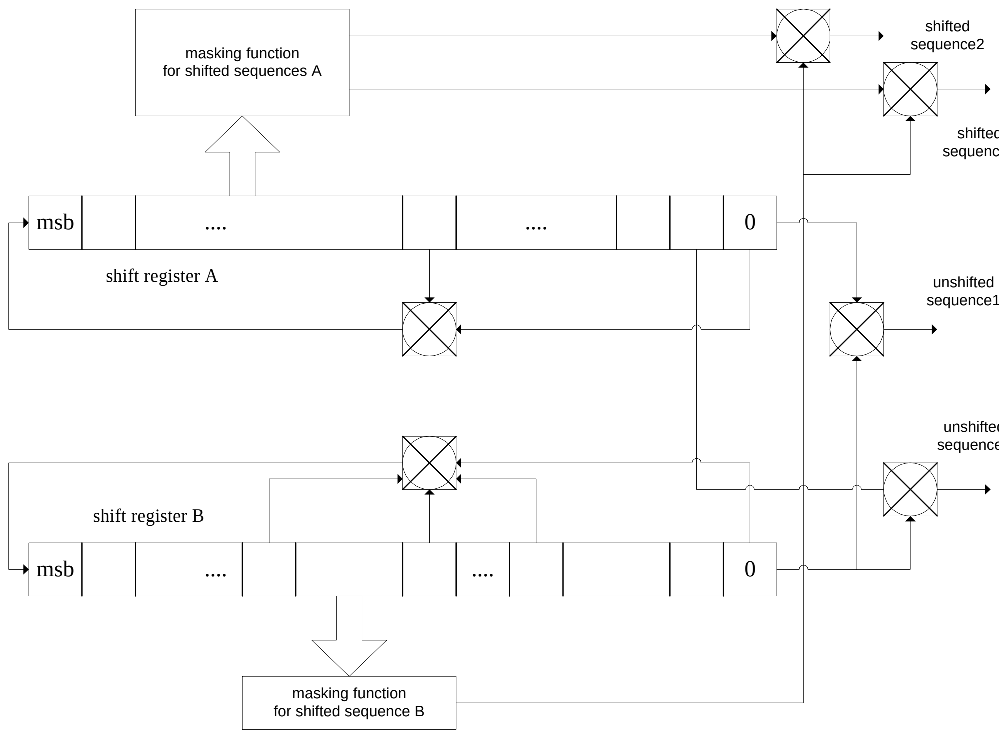

Figure 3 - multiple offset I/Q scrambling code generator (shows only 1 secondary)

3 Initialisation of the generator for (primary) scrambling codes

It has been pointed out that care should be taken in initialising the scrambling code generator. The problem seems to be as follows.

There are 2^18 possible initialisation values.

There are 2^9 = 512 primary scrambling sequences required.

Many assignments of initialisation values for the 512 required primary sequences may lead to sequences which are not sufficiently distinct. This may occur for random values, linear initialisation values or sequential use of 0..511 as initialisation values. Indeed only by examining the Galois field formed by the states of the (equivalent) linear feedback shift register (LFSR) can one be sure of the nature of the sequences.

This process may be done offline and fixed.

Given that ω is the root of the generator polynomial for the LFSR which is a primitive element of the field the initialisation values could be pre-computed as:

$1 = \omega^{0}, \omega^{1xN}, \omega^{2xN},\ldots\omega^{nxN},\ldots \omega^{511xN}$, where N is sufficiently large to make the sequences distinct in the multi-path radio environment.

This would ensure that the sequences were adequately separated along the m-sequence of the LFSR.

4 References

[1] Tdoc 915, "Multiple-Scrambling Code"

[2] Tdoc 724, " Multiple-Scrambling Codes"

[3] Tdoc 588, " Text proposal for downlink scrambling code phase shift parameter "

[4] Tdoc 806, " Text proposal for the figure of a downlink scrambling code generator "