

[0001] METHOD AND APPARATUS FOR ENCODING
 CHANNEL QUALITY INDICATOR AND PRECODING
 CONTROL INFORMATION BITS

[0002] FIELD OF INVENTION

[0003] The present invention is related to wireless communication systems.

[0004] BACKGROUND

[0005] In the conventional third generation partnership project (3GPP) wireless communication systems, the downlink data channel, (i.e., high speed downlink shared channel (HS-DSCH)), may carry different amount of data using a different spreading code and a different modulation and coding scheme (MCS). The amount of data that can be delivered to a user equipment (UE) over the HS-DSCH depends in part on the downlink channel quality. The UE reports the downlink channel quality to a Node-B using a channel quality indicator (CQI). The Node-B schedules the downlink transmissions based on the CQI reported by the UE.

[0006] For generation of the CQI, the UE performs channel quality measurements. The channel quality measurements are converted into a CQI value. Conventionally, a CQI value between 1 and 31 is generated using a lookup table. This CQI value is then encoded into CQI bits that are sent over the uplink channel by the UE.

[0007] Multiple-input multiple-output (MIMO) is a scheme that both the UE and the Node-B use more than one antenna in transmission and reception. When MIMO is implemented, multiple data streams may be transmitted between the UE and the Node-B and the UE may need to report multiple CQI values to the Node-B. In addition to the CQI, the UE also sends precoding control information (PCI) to instruct the Node-B as to preferences for downlink precoding.

[0008] When in a non-MIMO mode, the UE sends a five (5) bit CQI to the

Node-B. The CQI bits are encoded using a (20, 5) code. When in a MIMO mode, the UE may send two types of information to the Node-B. Type-A information is for support of two streams and type-B information is for support of a single stream. Type-A information currently requires ten (10) information bits: two (2) for PCI and eight (8) for CQI values, (e.g., four (4) CQI bits for each stream). Type-B information requires seven (7) information bits: five (5) for CQI and two for PCI. Different bit combinations may be used, but generally type-A information requires more bits than type-B information. Type A and B information are interspersed in uplink transmission as instructed by the network.

[0009] The CQI bits in a non-MIMO mode, and type A and type B information bits are block coded using (20, 5), (20, 10), and (20, 7) codes, respectively, to a total 20 coded bits using a generator matrix including a plurality of basis vectors. A linear (20, 10) code with a minimum distance of 6 is currently used for the type A information. The basis sequences for encoding type A information is shown in Table 1. The basis sequences for the type-B code are a subset of the basis sequences for the type-A code. For type B information, a linear combination of the first seven (7) basis sequences of the (20, 10) code is used for a (20, 7) code which also results in a minimum distance of 6 that is not the lowest that can be found for a (20, 7) code.

i	M _{i,0}	M _{i,1}	M _{i,2}	M _{i,3}	M _{i,4}	M _{i,5}	M _{i,6}	M _{i,7}	M _{i,8}	M _{i,9}
0	[1]	[0]	[0]	[0]	[0]	[0]	[0]	[0]	[0]	[0]
1	[0]	[1]	[0]	[0]	[0]	[0]	[0]	[0]	[0]	[0]
2	[0]	[0]	[1]	[0]	[0]	[0]	[0]	[0]	[0]	[0]
3	[0]	[0]	[0]	[1]	[0]	[0]	[0]	[0]	[0]	[0]
4	[0]	[0]	[0]	[0]	[1]	[0]	[0]	[0]	[0]	[0]
5	[0]	[0]	[0]	[0]	[0]	[1]	[0]	[0]	[0]	[0]
6	[0]	[0]	[0]	[0]	[0]	[0]	[1]	[0]	[0]	[0]
7	[0]	[0]	[0]	[0]	[0]	[0]	[0]	[1]	[0]	[0]
8	[0]	[0]	[0]	[0]	[0]	[0]	[0]	[0]	[1]	[0]
9	[0]	[0]	[0]	[0]	[0]	[0]	[0]	[0]	[0]	[1]
10	[1]	[0]	[0]	[0]	[1]	[0]	[1]	[1]	[1]	[1]
11	[1]	[1]	[0]	[0]	[0]	[1]	[0]	[1]	[1]	[1]
12	[0]	[1]	[1]	[0]	[0]	[0]	[1]	[0]	[1]	[1]
13	[1]	[0]	[1]	[1]	[0]	[0]	[0]	[1]	[0]	[1]
14	[1]	[1]	[0]	[1]	[1]	[0]	[0]	[0]	[1]	[0]
15	[1]	[1]	[1]	[0]	[1]	[1]	[0]	[0]	[0]	[1]
16	[1]	[1]	[1]	[1]	[0]	[1]	[1]	[0]	[0]	[0]
17	[0]	[1]	[1]	[1]	[1]	[0]	[1]	[1]	[0]	[0]
18	[1]	[0]	[1]	[1]	[1]	[1]	[0]	[1]	[1]	[0]
19	[0]	[1]	[0]	[1]	[1]	[1]	[1]	[0]	[1]	[1]

Table 1

[0010] Since the CQI and PCI bits are transmitted to the Node-B without a cyclic redundancy check (CRC), the transmitted CQI bits may be incorrectly received and consequently incorrect CQI bits may be used for the downlink scheduling by the Node-B, which results in system capacity degradation. Unlike errors in other decoding, the size of the error matters in CQI decoding. The CQI value ranging from 1 to 31 is mapped to CQI values, an error in a most significant bit (MSB) would generate a bigger error than an error in a non-MSB. Conventional codes are designed to minimize the occurrence of any errors and do not necessarily minimize the average 'size' of the errors.

[0011] Therefore, it would be desirable to use codes that minimize the size of the errors. In addition or alternative, a better code with a higher minimum distance and weight would improve the detection probability of type-B information.

[0012] SUMMARY

[0013] The present invention is related to a method and apparatus for encoding CQI and PCI bits. Each of input bits, such as CQI bits and/or PCI bits, has a particular significance. The input bits may be encoded with a linear block coding. The input bits are provided with an unequal error protection based on the significance of each input bit. The input bits may be duplicated based on the significance of each input bit and equal protection coding may be performed. A generator matrix for the encoding may be generated by elementary operation of conventional basis sequences to provide more protection to an MSB.

[0014] BRIEF DESCRIPTION OF THE DRAWINGS

[0015] A more detailed understanding of the invention may be had from the following description of a preferred embodiment, given by way of example and to be understood in conjunction with the accompanying drawings wherein:

[0016] Figure 1 is a block diagram of an example WTRU in accordance with the present invention;

[0017] Figure 2 is a block diagram of an example Node-B in accordance with the present invention;

[0018] Figure 3 shows throughput comparison between the conventional 3GPP CQI code and the above code modified in just one bit of its generator matrix;

[0019] Figure 4 shows an RMS error performance comparison between an unequal error protection code constructed in accordance with the present invention and a maximum minimum Hamming distance code; and

[0020] Figure 5-12 show simulation results for the coding scheme in accordance with the present invention.

[0021] DETAILED DESCRIPTION

[0022] When referred to hereafter, the terminology "wireless transmit/receive unit (WTRU)" includes but is not limited to a user equipment (UE), a mobile station, a fixed or mobile subscriber unit, a pager, a cellular telephone, a personal digital assistant (PDA), a computer, or any other type of user device capable of operating in a wireless environment. When referred to hereafter, the terminology "base station" includes but is not limited to a Node-B, a site controller, an access point (AP), or any other type of interfacing device capable of operating in a wireless environment.

[0023] It should be noted that although the present invention is described with reference to the specific application of CQI and/or PCI, the coding scheme of the present invention is applicable to coding any information. It should also be noted that specific dimension of the generator matrices are given only as an example, not as a limitation, and the exact presentation of the code depends on the bit placement of PCI and CQI.

[0024] Figure 1 is a block diagram of an example WTRU 100 in accordance with the present invention. The WTRU 100 may include a data generator 102, an encoder 104 and a transmitter 106. It should be noted that the WTRU 100 may further include any conventional processing components. The data generator 102 may include a CQI generator 108 and/or a PCI generator 110. The

CQI generator 108 generates at least one set of CQI bits. The PCI generator 110 generates PCI bits. The data generator 102 generates either CQI bits only, or type A or type B CQI/PCI bits. The CQI bits or the type A or type B CQI/PCI bits are encoded by the encoder 104. The detailed encoding schemes will be explained in detail below. The encoded CQI bits or CQI/PCI bits are transmitted by the transmitter 106.

[0025] Figure 2 is a block diagram of an example Node-B 200 in accordance with the present invention. The Node-B 200 may include a receiver 202, a decoder 204 and a scheduler 206. It should be noted that the Node-B 200 may further include any conventional processing components. The receiver 202 receives the encoded CQI bits or CQI/PCI bits from the WTRU 100. The decoder 204 decodes the encoded CQI or CQI/PCI bits to recover the CQI and/or PCI bits. The recovered CQI bits and/or PCI bits are used by the scheduler 206 for scheduling the next transmission to the WTRU 100.

[0026] In accordance with one embodiment, the error protection of the input bits is given in accordance with significance of each of the bits such that the most error protection is given to an MSB and a least protection is given to a least significant bit (LSB). The input bits can be thought of as a bit vector. For example, the range of the CQI value is 1 to 31, the CQI value is converted into 5 bit vector, $\mathbf{b} = [b_0, \dots, b_4]$, where b_0 is the MSB and b_4 is the LSB. It should be noted that the position of the MSB and the LSB may be opposite. The linear block coding may be described by an $n \times k$ generator matrix, where k is the number of input bits, (e.g., 5 CQI bits, 8 CQI bits and 2 PCI bits, or 5 CQI bits and 2 PCI bits), and n is the number of output bits, (e.g., 20 bits). The k input bits are multiplied by the generator matrix to produce an n -bit codeword as follows:

$$\mathbf{c} = \mathbf{mG}; \quad \text{Equation (1)}$$

where $\mathbf{c}$ is the output codeword, $\mathbf{m}$ is the input vector and $\mathbf{G}$ is the generator matrix.

[0027] Each one of the output bits (elements in $\mathbf{c}$) can be thought of a parity-check on a subset of bits in $\mathbf{m}$. Which bits "participate" in each one of the

parity check is determined by $\mathbf{G}$. The more parity checks a particular bit participates in, the more redundancy is generated from it and the better protected it will be.

[0028] The input bits are ordered in descending or ascending need of protection. A list of k positive integers $m_0, \dots, m_{k-1}$, is generated such that:

$$n \geq m_0 \geq \dots \geq m_{k-1} \geq 1. \quad \text{Equation (2)}$$

The generator matrix $\mathbf{G}$ is then generated as follows: the row of $\mathbf{G}$ corresponding to b_i , (i.e., row $i+1$), is populated with m_i 1's and the rest of the entries are 0's. Thus, the input bits participate in progressively less and less parity checks and therefore incur progressively less error protection.

[0029] This method reduces the probability of large errors at the expense of the probability of much smaller errors which are insignificant to system performance. This offers substantial freedom in the design of the code and depending on the specific placement on 1's a codes may turn out to be good or bad. In particular, the matrix $\mathbf{G}$ should remain full row-rank. Preferably, the minimum number of 1's in any non-zero linear combination of the rows should be minimized. A pseudo-random code design is likely to result in a good code especially as block length increases. For very large lengths such codes are a sub-family of irregular low density parity check (LDPC) codes. Two alternative 5×20 generator matrix examples for the conventional CQI configuration are shown below (the MSB in the input vector is the leftmost bit and the LSB is the rightmost bit in this example).

[0030] Generator matrix example 1:

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0
1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0

[0031] Generator matrix example 2:

```

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0
1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0

```

[0032] The WTRU 100 may need to transmit two or more CQI values that pertain to a single stream or to two or more MIMO streams. In such case, the WTRU 100 generates multiple CQI values and mapped the CQI values to multiple sequences of CQI bits. For example, the WTRU 100 may generate three CQIs including one CQI for a single stream case and two CQIs for a two stream case. One of the CQI bits, (e.g., the single stream CQI bits), is denoted as a primary sequence (P), and the others are denoted as a secondary sequence, (e.g., S'1 and S'2). Then, the primary and the secondary sequences are concatenated for coding according to the above coding scheme, (i.e., the input to the encoder 104 is defined as $m=[P \ S'1 \ S'2]$). Alternatively, the primary sequence and the secondary sequences may be combined, (e.g., $S1=S'1-P$ and $S2=S'2-P$) and concatenated to be input into the encoder 104, (e.g., $m=[P \ S1 \ S2]$). Either way, the rows in the generator matrix must reflect the fact that the weights of the bits of m are now not exponential, although the weight of its components is.

[0033] Alternatively, the input bits may be unequally duplicated and then coded using an equal protection code, such as Reed-Muller (RM) code.

[0034] The CQI bits may be concatenated with acknowledgement information. In the 3GPP frequency division duplex (FDD) Release 6 standard, the high speed uplink control channel (HS-DPCCH) is constructed such that the first two time slots carry CQI information and the third time slot carries acknowledgement information, (i.e., positive acknowledgement (ACK) or negative acknowledgement (NACK)). Since the WTRU 100 usually generates a CQI value based on channel estimation prior to the CRC check on the received data block, the WTRU 100 may transmit the CQI first, and then transmit the ACK/NACK so that the latency in the link adaptation can be reduced. In order to preserve this advantage while using an unequal error protection coding scheme, the CQI bits and acknowledgement bits are concatenated and encoded as follows:

$$\mathbf{c} = \mathbf{m}\mathbf{G} = [\mathbf{m}_{\text{CQI}} \mathbf{m}_{\text{Ack}}]\mathbf{G}; \text{Equation (3)}$$

where $\mathbf{m}_{\text{CQI}}$ is CQI bits, $\mathbf{m}_{\text{Ack}}$ is acknowledgement information, and $\mathbf{G}$ is a generator matrix. For example, $\mathbf{m}_{\text{CQI}}$ is a 1×5 row vector, $\mathbf{m}_{\text{Ack}}$ is one bit for a single stream (in general $1 \times m$ row vector for m streams), and $\mathbf{G}$ is a 6×30 matrix with the first 20 elements in the last row being 0s.

[0035] This embodiment may be extended to encode a set of multiple codes which require different transmit time as follows:

$$[\bar{c}_1, \bar{c}_2, \dots, \bar{c}_N] = [\bar{m}_1, \bar{m}_2, \dots, \bar{m}_M] \begin{bmatrix} G_{11} & G_{12} & \dots & G_{1N} \\ 0 & G_{22} & \dots & G_{2N} \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & \dots & G_{MN} \end{bmatrix}. \quad \text{Equation (4)}$$

[0036] In accordance with another embodiment, the conventional 3GPP type A or type B code is changed by moving one or more bits in the generator matrix to improve the performance. The basis sequences for the new (20, 5) code generated in this way are shown in Table 2. The code is based on the conventional 3GPP (20,5) code in the non-MIMO mode. In Table 2, the fifth column corresponds to the MSB and the fourth column corresponds to the next MSB. The bit '1' in the fifth column and 20th row is changed to '0' and the bit '0' in the fourth column and 20th row is changed to '1' to give more protection to the next MSB at the expense of the MSB protection.

i	$M_{i,0}$	$M_{i,1}$	$M_{i,2}$	$M_{i,3}$	$M_{i,4}$
0	1	0	0	0	1
1	0	1	0	0	1
2	1	1	0	0	1
3	0	0	1	0	1
4	1	0	1	0	1
5	0	1	1	0	1
6	1	1	1	0	1
7	0	0	0	1	1
8	1	0	0	1	1
9	0	1	0	1	1
10	1	1	0	1	1
11	0	0	1	1	1
12	1	0	1	1	1
13	0	1	1	1	1
14	1	1	1	1	1
15	0	0	0	0	1
16	0	0	0	0	1
17	0	0	0	0	1
18	0	0	0	0	1
19	0	0	0	1	0

Table 2

[0037] Figure 3 shows throughput comparison between the conventional 3GPP CQI code and the code in Table 2. Figure 3 show the performance improvement with the new code.

[0038] The generator matrix in Table 2 or its variance may be generated by starting with a maximum minimum Hamming distance code of a size somewhat smaller than the desired code. For example, if a (20, 5) code is desired, a maximum minimum Hamming distance code of size (17, 5), (i.e., 5×17 generator matrix), is first found. Once the 5×17 generator matrix is found, the size of the generator matrix is extended by adding a 5×3 matrix Z of zeros to the 5×17 generator matrix. Then, some or all of the zeros in the p -th row of the Z matrix is set to '1', where p corresponds to the position of the MSB in the data being encoded. This makes the distance between codewords with MSB=1 and those with MSB=0 larger, (i.e., more protection to the MSB). If not all elements in row p are set to '1', then the element in row q and in column that is not set to '1' in p -th row is set to '1' where q corresponds to the position of the next MSB in the data being encoded. In this way, the more important bits are better protected. However it comes at the cost of having a minimum distance set by the smaller

(n,k) code.

[0039] Generator matrix example 3 for an $(20, 8)$ unequal error protection code generated in the similar way is shown below. The generator matrix example 3 is generated from a $(17, 8)$ maximum minimum Hamming distance code. The right most 8×3 submatrix is added to the 8×17 matrix. In this example, the first row corresponds to the MSB and the second row corresponds to the next MSB. All elements of the submatrix are zeros, except the first row and the element in the second row and the last column for better protection of the next MSB.

[0040] Generator matrix example 3:

1	0	0	0	0	0	0	0	1	1	1	0	0	1	1	0	0	1	1	1
0	1	0	0	0	0	0	0	1	1	1	0	0	0	0	1	1	0	0	1
0	0	1	0	0	0	0	0	1	0	0	1	1	1	0	1	0	0	0	0
0	0	0	1	0	0	0	0	1	0	0	1	0	1	1	0	1	0	0	0
0	0	0	0	1	0	0	0	0	1	0	1	1	0	1	0	1	0	0	0
0	0	0	0	0	1	0	0	0	1	0	0	1	1	0	1	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1	1	0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	1	0	0	1	0	1	1	1	1	0	0	0	0

[0041] Figure 4 shows an RMS error performance comparison between an unequal error protection code constructed in accordance with this embodiment and a maximum minimum Hamming distance code. The RMS error is better for the unequal error protection code in accordance with this embodiment than the maximum minimum Hamming distance code in the region of interest.

[0042] In current 3GPP standards, type A information is coded with $(20, 10)$ code and type B information is coded with a $(20, 7)$ code which is a subset code of the $(20, 10)$ code for the type A information. A subset code refers to the case where the basis vectors of the $(20, 7)$ code are a subset of the basis vectors of the $(20, 10)$ code.

[0043] In accordance with one embodiment, a non-subset, linear $(20, 7)$ code is used for type B information and the conventional type A information coding (the $(20, 10)$ coding specified in the current 3GPP standards) is used for type A information. The non-subset $(20, 7)$ code may be the subset $(20, 7)$ code of the conventional $(20, 10)$ code with one (or more) bit flipping. The minimum

distance of the (20, 7) code is eight (8). The non-subset (20, 7) code is generated such that the weight distribution of the code is non-optimal and the code offers better protection to the MSB of the CQI value. A generator matrix example 4 for this embodiment is shown below (in this example, the MSB in the input vector is the rightmost bit and the LSB is the leftmost bit).

[0044] Generator matrix example 4:

```

1 0 0 0 0 0 0 1 1 0 0 0 1 1 1 0 1 0 1 0
0 1 0 0 0 0 0 1 0 1 1 0 0 1 0 0 1 1 1 0
0 0 1 0 0 0 0 1 0 1 0 1 1 0 0 1 1 0 1 0
0 0 0 1 0 0 0 1 0 1 0 0 1 1 1 1 0 1 0 0
0 0 0 0 1 0 0 1 0 0 1 1 1 0 1 0 1 1 0 0
0 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 1 0
0 0 0 0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1

```

[0045] A better protection of the MSB of CQI bits reduces the probability of large magnitude errors in CQI at the Node-B at the expense of a slightly higher probability of error. Both PCI errors and small magnitude CQI errors have a small impact on system performance while large magnitude CQI errors have a significant impact. Therefore better error protection provided to the MSB of CQI bits is advantageous.

[0046] In accordance with another embodiment, a non-subset (20, 7) linear code that has a minimum distance of 8 is used and equal error protection is provided for type B information, and the conventional type A information coding (the (20, 10) coding specified in the current 3GPP standards) is used for type A information. One way of generating this non-subset (20, 7) code is to start with a good smaller code and search for extensions. A generator matrix example 5 for this embodiment is shown below (in this example, the MSB in the input vector is the rightmost bit and the LSB is the leftmost bit).

[0047] Generator matrix example 5:

```

1 0 0 0 0 0 0 1 1 0 0 0 1 1 1 0 1 0 1 0
0 1 0 0 0 0 0 1 0 1 1 0 0 1 0 0 1 1 1 0
0 0 1 0 0 0 0 1 0 1 0 1 1 0 0 1 1 0 1 0
0 0 0 1 0 0 0 1 0 1 0 0 1 1 1 1 0 1 0 0
0 0 0 0 1 0 0 1 0 0 1 1 1 0 1 0 1 1 0 0
0 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 1 1
0 0 0 0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1

```

[0048] In accordance with another embodiment, for the case where CQI and PCI use 4 and 2 bits, respectively, approximately equal error protection code is used for type B information coding and the conventional type A information coding is used for type A information. A (20, 6) generator matrix example 6 for this embodiment is given below (in this example, the MSB in the input vector is the rightmost bit and the LSB is the leftmost bit).

[0049] Generator matrix example 6:

```

1 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 0 0 0 1
0 1 0 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1
0 0 1 0 0 0 0 1 0 1 1 0 1 0 1 0 0 1 0 1
0 0 0 1 0 0 0 0 1 1 1 0 0 1 1 0 0 0 1 1
0 0 0 0 1 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1
0 0 0 0 0 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1

```

[0050] In accordance with another embodiment, type B input bits are unequally duplicated before encoding. There are many variations on this scheme in which one or more of the CQI MSBs and/or some or all of the PCI bits are emphasized to create a 10-bit input to the (20, 10) encoder used for type A information coding. For example, the MSB of type B CQI may be repeated 3 times. Note that this is still a (20, 7) code with a generator matrix that can be constructed from linear combinations of the basis vectors of the generator matrix of the (20, 10) code.

[0051] In accordance with another embodiment, both type A and type B codes are changed from the conventional code and the type B code is a subset code composed of the first 7 columns of the type A code. A new type A generator matrix is created from the conventional (20, 10) generator matrix by using elementary column operations, (such as a column is altered by adding one other

column to it). It is well known that these operations do not lead to codewords that are not also part of the original code. They do, however, enable to construct a better type B subset code.

[0052] The appropriate type B code can be determined in terms of its separation vector. For a linear (n,k) binary code C , the separation vector (SV) $s(G) = (s(G)_1, \dots, s(G)_k)$ of length k with respect to a generator matrix G of C is defined by:

$$s(G)_i = \min\{wt(mG) \mid m \in GF(2)^k, m_i \neq 0\}, i = 1, \dots, k. \quad \text{Equation (5)}$$

[0053] It is generally understood that the value of an element of the SV corresponds to the protection that is given to the corresponding information bit akin to minimum distance properties of codes in general. Thus, for approximately equal error protection code, all the elements of the separation vector are equal. By finding codes with unequal element values, it is possible to protect better some desired information bits. Specifically, if a code is found that has $\min_i(s(G)_i) \geq 6$ with larger elements for the MSB of CQI bits, then this code is an improvement over the conventional type B coding without changing the properties of type A code.

[0054] For example, this code may be obtained by the following operations:

- 1) The 5th column basis sequence is replaced by the XOR operation of column 5 and column 8;
- 2) The 6th column basis sequence is replaced by the XOR operation of column 6 and column 9; and
- 3) The 7th column basis sequence is replaced by the XOR operation of column 7 and column 10.

[0055] The generator matrix obtained in this way (in a transpose form) with separating vector of $s(G) = (7, 6, 6, 6, 7, 7, 7)$ is shown below.

[0056] Generator matrix example 7:

```

1 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1 0 0
0 0 0 0 1 0 0 1 0 0
0 0 0 0 0 1 0 0 1 0
0 0 0 0 0 0 1 0 0 1
1 0 0 0 0 1 0 1 1 1
1 1 0 0 1 0 1 1 1 1
0 1 1 0 0 1 0 0 1 1
1 0 1 1 1 0 1 1 0 1
1 1 0 1 1 1 0 0 1 0
1 1 1 0 1 1 1 0 0 1
1 1 1 1 0 1 1 0 0 0
0 1 1 1 0 0 1 1 0 0
1 0 1 1 0 0 0 1 1 0
0 1 0 1 1 0 0 0 1 1

```

[0057] Further coding examples may be obtained by permutation of the above codes or by starting with a smaller code and searching for the best extensions. Further generator matrix examples are provided herein.

[0058] Generator matrix example 8:

```

1 0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 1 1 0
0 1 0 0 0 0 0 0 0 1 1 0 0 1 0 1 0 1 1 1
0 0 1 0 0 0 0 0 0 1 1 0 0 0 1 1 1 0 1 0
0 0 0 1 0 0 0 0 0 1 0 1 1 0 0 1 0 0 1 1
0 0 0 0 1 0 0 1 1 0 1 0 0 1 0 0 0 1 0 1
0 0 0 0 0 1 0 1 0 0 0 1 1 1 1 0 0 0 0 1
0 0 0 0 0 0 1 0 0 1 0 0 1 1 1 0 1 0 1 1
0 0 0 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0
0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 1 1 0

```

[0059] Generator matrix example 9:

1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0	1	0
0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0	1
0	0	1	0	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	1	1
0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	1
0	0	0	0	1	0	1	1	0	0	1	1	1	1	1	1	1	1	1	0	0
0	0	0	0	0	1	0	1	0	1	0	1	1	0	0	0	1	1	0	0	0
0	0	0	0	0	0	1	1	1	1	1	0	1	1	0	0	0	1	0	0	0
0	0	0	0	0	0	0	1	0	0	1	1	0	1	0	0	0	1	1	0	0
0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	0	1	1	0	1	0
0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	0	1	1	0	1	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	1

[0060] Generator matrix example 10:

1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	1	1	1	0
0	1	0	0	0	0	0	0	1	1	0	0	0	1	1	1	0	1	0	1
0	0	1	0	0	0	0	0	1	0	1	1	0	0	1	0	0	1	1	1
0	0	0	1	0	0	0	0	1	0	1	0	1	1	0	0	1	1	0	1
0	0	0	0	1	0	0	0	1	0	1	0	0	1	1	1	1	0	1	0
0	0	0	0	0	1	0	0	1	0	0	1	1	1	0	1	0	1	1	0
0	0	0	0	0	0	1	0	1	0	0	1	1	0	1	1	1	0	0	1
0	0	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	1	0	0	1	1	1	0	1	1	0	0	1	0	1	0
0	0	0	0	0	0	1	0	1	0	1	1	1	1	1	0	1	1	1	0

[0061] Generator matrix example 11:

1	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0	1	0
0	1	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0	1
0	0	1	0	0	0	1	1	0	1	1	0	1	1	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1
0	0	0	0	1	0	1	1	0	0	1	1	1	1	1	1	1	1	0	0
0	0	0	0	0	1	0	1	0	1	0	1	1	0	0	0	1	1	0	0
0	0	0	0	0	0	0	0	1	0	1	1	1	0	1	0	0	0	1	1
0	0	0	0	0	0	0	1	0	0	1	1	0	1	0	0	0	1	1	0
0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	0	1	1	0	1
0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	1

[0062] Generator matrix example 12:

```

1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 0 1 0
0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 0 1
0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1
0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 1 0 1 1 1
0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 1 0 1 1
0 0 0 0 0 0 1 1 1 1 1 0 1 0 0 0 1 1 0 1 1
0 0 0 0 0 0 0 1 0 0 1 1 0 1 0 0 0 1 1 0 1
0 0 0 0 0 0 0 0 1 0 0 1 1 0 1 0 0 0 1 1 0
0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 1 0 0 0 1 1
0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 1 0 0 0 1

```

[0063] Generator matrix example 13:

```

1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 0 1 0
0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 0 1
0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1
0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 1 0 1 1 1
0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 1 1 0 1 1
0 0 0 0 0 0 1 1 1 0 1 0 1 0 0 0 1 1 0 1 1
0 0 0 0 0 0 0 1 1 1 0 1 0 1 0 0 0 1 1 0 1
0 0 0 0 0 0 0 0 1 0 0 1 1 0 1 0 0 0 1 1 0
0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 1 0 0 0 1 1
0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 1 0 0 0 1

```

[0064] Generator matrix example 14:

```

1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 0 1 0
0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 0 1
0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1
0 0 0 0 1 0 0 1 0 0 1 0 0 0 1 1 0 1 1 1 1
0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 1 1 0 1 1 1
0 0 0 0 0 0 1 0 0 1 1 0 1 0 0 0 1 1 0 1 1
0 0 0 0 0 0 0 1 0 0 1 1 0 1 0 0 0 1 1 0 1
0 0 0 0 0 0 0 0 1 0 0 1 1 0 1 0 0 0 1 1 0
0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 1 0 0 0 1 1
0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 1 0 0 0 1

```

[0065] Further subset coding examples may be created in the following manner. First, the 'best' type A code is identified. For example, for the (20,10) codes, it is known that there are 1,682 non-equivalent maximal minimum

distance linear binary (20,10,6) codes having a minimum distance of six (6). A maximal (n,k) minimum distance code is a code that achieves the theoretical maximum minimal distance between all pairs of codewords. Among these codes, codes with a lowest block error rate (BLER) are found. The BLER may be computed from its weight distribution. For the case of the (20,10) codes, the best code in the state of the art is unique and easily identified as it has a minimum distance of six (6) and the fewest number of minimum distance codewords (40) by a large margin. When looking for a good subset code, it is only needed to investigate the (20,10,6,40) codes, all equivalent to the code in the state of the art, from which to draw a subset code.

[0066] Subset codes for then searched from these codes. By taking an (n, k) code and applying any sequence of elementary operations, other codes are created whose codewords are either the same as the original codewords of (n,k) code, which makes the codes equivalent, or a smaller subset of those codewords. The codes with smaller subsets of codewords are degenerate and are not worth investigation. The others are potential candidates to be used to create a subset code.

[0067] For a (n, k) type A code, all sequences of elementary operations may be represented as a pre-multiplying $k \times k$ matrix of 1's and 0's to the generator matrix. The pre-multiplying matrix is called a translation matrix, (e.g., 10×10 or alternatively 7×10). The translation matrix may be generated deterministically or randomly. Each of these subset codes are then checked for good properties, such as achieving max min distance, counting the number of minimum weight codewords, and computation of the separation vector.

[0068] For all (20,10) coding examples there are many bit mapping possibilities that may result in different coding properties. For example, there may be two CQI values of four (4) bits each mapped as pci0, pci1, cqi1_0, cqi1_1, cqi1_2, cqi1_3, cqi2_0, cqi2_1, cqi2_2, cqi2_3, where cqi1_3 and cqi2_3 are MSBs. This sequence is mapped to rows 1:10 of the generating matrix. Alternatively, the sequence may be mapped to rows 10:1 of the generator matrix in reverse order.

[0069] Alternatively, the two four bit CQI values may be combined into a single eight bit CQI composite (CQIC). For example, it has been proposed that $CQIC = \{15 \cdot CQI1 + CQI2 + 31\}$ for two streams, and {SingleCQI} for one stream. CQI1 and CQI2 have 15 possible values each and SingleCQI has 30 possible values. Then, cqic7, cqic6, cqic5, cqic4, cqic3, cqic2, cqic1, pci1, pci0, cqic0 may be applied to rows 1:10 of the generator matrix, or alternatively 10:1 in reverse order. For type B (20, 7) coding, cqi4, cqi3, cqi2, cqi1, cqi0, pci1, pci0 may be mapped to rows 1:7, or alternatively 7:1 in reverse order. A generator matrix example 15 obtained in this way is shown below.

[0070] Generator matrix example 15:

1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	1	0
0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	1
0	0	1	0	0	0	0	0	0	0	1	1	1	0	0	0	0	1	1	1
0	0	0	1	0	0	0	1	0	0	1	1	1	1	0	0	1	1	0	0
0	0	0	0	1	0	0	1	1	0	1	1	0	0	0	1	1	0	0	1
0	0	0	0	0	1	0	0	1	0	1	1	1	0	0	1	0	1	1	0
0	0	0	0	0	0	1	1	0	1	1	1	0	0	1	1	0	1	0	0
0	0	0	0	0	0	0	1	0	0	1	1	1	0	1	0	0	0	1	1
0	0	0	0	0	0	0	0	1	1	0	1	0	1	0	1	1	1	0	0
0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	1

[0071] Type B PCI/CQI information coding results in lower transmitted power requirements due to different coding rate and coding gain. Such lower transmit power is desirable as it lowers the average uplink interference. Type A power level is currently determined by the network as a multiple of an uplink reference channel, (i.e., dedicated physical control channel (DPCCH). In accordance with the present invention, different transmission powers are used for type A and type B information. Both type A and type B power may be independently defined, (i.e., as power multiplier or addition in dBs), in terms of the power of another channel. The multipliers or additions are provided by the network. The reference channel may be downlink channel or uplink channel. The downlink reference channel may be common pilot channel (CPICH), primary common control physical channel (P-CCPCH), synchronization channel (SCH), high speed shared control channel (HS-SCCH) or any other channel. Power of

some of those may in itself be defined in terms of power of other channels. The uplink reference channel may be DPCCH, enhanced uplink DPCCH (E-DPCCH), random access channel (RACH) (after ACK/NACK) or any other channel. Power of some of those may in itself be defined in terms of power of other channels.

[0072] Conventionally, after CQI bits are spread to the chip rate, the real-valued spread signals are weighted by a gain factor. The gain factor β_{hs} for the CQI bits on the HS-DPCCH is derived from the quantized amplitude ratio ($A_{hs} = \beta_{hs}/\beta_c$), which is translated from Δ_{CQI} signaled by a higher layer. β_c is the gain factor for the DPCCH. The translation of Δ_{CQI} into the quantized amplitude ratio A_{hs} in accordance with the present invention is shown in Table 3.

Signalled values for Δ_{CQI}	Quantized amplitude ratios $A_{hs} = \beta_{hs}/\beta_c$
9	38/15
8	30/15
7	24/15
6	19/15
5	15/15
4	12/15
3	9/15
2	8/15
1	6/15
0	5/15

Table 3

[0073] For HS-DPCCH slots carrying CQI bits, if the WTRU is not configured in MIMO mode, A_{hs} equals the quantized amplitude ratio translated from the signaled value Δ_{CQI} . If the WTRU is configured in MIMO mode, A_{hs} equals the quantized amplitude ratio translated from the signaled value Δ_{CQI} when a CQI of type B is transmitted, and A_{hs} equals the quantized amplitude ratio translated from the signaled value $\Delta_{CQI} + 1$ when a CQI of type A is transmitted.

[0074] Alternatively, once one of the type A or type B power is defined as explained above and transmission power for the other type may be determined as a network provided multiplier or dB addition or may be computed from the defined one based on a pre-defined formula. For example, type B power may be

computed by the WTRU using a look up table (LUT_A). Conventionally, the LUT_A is used to compute type A power from an index Δ_{CQI} signaled by the network. Given Δ_{CQI} , type B power may be computed using a separate lookup table LUT_B that implements the function $P_B = LUT_B(\Delta_{CQI})$. An example LUT_B is shown in Table 4.

Signalled values for Δ_{CQI}	Quantized amplitude ratios $A_{hs} = B_{hs}/B_c$
8	21/15
7	17/15
6	14/15
5	11/15
4	9/15
3	7/15
2	6/15
1	5/15
0	5/15

Table 4

[0075] Alternatively, a function g that operates on Δ_{CQI} may be used to create an input to LUT_A thus keeping the same set of power levels that are used for type A. This operation may be described as $P_B = LUT_A(g(\Delta_{CQI}))$. The function g may itself be implemented as a lookup table. For example, $g(\Delta_{CQI}) = \text{maximum}(0, g(\Delta_{CQI})-1)$.

[0076] Figure 5 shows an individual bit error rate using the generator matrix example 4. It shows that the MSB of CQI bits (cqi4) is about 0.6 dB better than the rest of PCI/CQI bits. As a reference, the bit error rate of the conventional coding scheme (labeled as "pcicqi in [1]") is also plotted. The BER measurement using the generator matrix example 4 is about 0.8 dB better than the conventional coding scheme. Figure 6 shows a block error rate (BLER) for the conventional type B 7-bit PCI/CQI and the coding using the fourth generator matrix example. At 1% BLER, there is about a 0.8 dB difference in coding again. Figure 7 shows BLER vs. the coded bit signal-to-noise ratio (SNR) for the conventional type A 10 bit PCI/CQI, conventional type B 7-bit PCI/CQI and the coding using the fourth generator matrix example. For comparison between the type A and type B reporting formats, the power required at 1% BLER for type A

PCI/CQI format is about 1.5 dB higher than the one using the generator matrix example 8 for type B. Figure 8 is a plot of CQI mean square error (MSE) measurement in two type B coding schemes. For the same amount of MSE errors, there is about > 1 dB in SNR required.

[0077] Figures 9-12 show performance comparisons for conventional coding and the coding using the fifth and/or the seventh generator matrix examples. Note that in Figures 9-12, "[1]" refers to prior art for type-B, "optimal EEP" represents the case using the fifth generator matrix example, and "G1" represents the case using the seventh generator matrix example. Figure 9 shows that type-A matrix transformation does not change the BLER of type-A code. Figure 10 shows a comparison for PCI BLER for prior art type-B code, the case using the fifth generator matrix example and the case using the seventh generator matrix example. Figure 11 shows a graphical comparison of CQI BLER for the prior art type-B code, the case using the fifth generator matrix example and the case using the seventh generator matrix example. Figure 12 shows comparison of CQI standard deviation, (computed as root mean square error (RMSE)), for prior art type-B coding, and the case using the fifth generator matrix example and the case using the seventh generator matrix example. As can be seen in Figures 9-12, both the fifth and the seventh generator matrix examples perform better than the prior art.

[0078] Embodiments.

[0079] 1. A method for encoding information bits.

[0080] 2. The method of embodiment 1 comprising generating input bits, each input bit having a particular significance.

[0081] 3. The method of embodiment 2 comprising performing a linear block coding on the input bits to generate an output codeword, the input bits being provided with a specific level of error protection based on the significance of each input bit.

[0082] 4. The method of embodiment 3 comprising sending the output codeword.

[0083] 5. The method as in any one of embodiments 2-4, wherein the

input bits are CQI bits.

[0084] 6. The method of embodiment 5 further comprising generating multiple sets of CQI bits, one of the multiple sets of CQI bits being denoted as primary CQI bits and the remaining sets of CQI bits being denoted as secondary CQI bits.

[0085] 7. The method of embodiment 6 comprising concatenating the primary CQI bits and the secondary CQI bits as the input bits for encoding.

[0086] 8. The method of embodiment 5 further comprising generating multiple sets of CQI bits, one of the multiple sets of CQI bits being denoted as primary CQI bits and the remaining sets of CQI bits being denoted as secondary CQI bits.

[0087] 9. The method of embodiment 8 combining the secondary CQI bits with the primary CQI bits.

[0088] 10. The method of embodiment 9 comprising concatenating the primary CQI bits and the combined secondary CQI bits as the input bits for encoding.

[0089] 11. The method as in any one of embodiments 2-10, wherein the input bits are five (5) bits, and the generator matrix is for (20,5) coding as follows:

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0
1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0

[0090] 12. The method as in any one of embodiments 2-10, wherein the inputs bits are five (5) bits, and the generator matrix is for (20,5) coding as follows:

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

[0091] 13. The method as in any one of embodiments 2-4, wherein the input bits include at least one set of CQI bits and at least one set of PCI bits that are concatenated.

[0092] 14. The method as in any one of embodiments 2-13, wherein at least one input bit is duplicated depending on the significance of the input bit before performing the encoding.

[0093] 15. The method as in any one of embodiments 2-14, wherein the input bits are at least two different information bits that are concatenated and need to be transmitted in different timing.

[0094] 16. The method of embodiment 15 wherein the inputs bits are CQI bits and acknowledgement bits.

[0095] 17. The method of embodiment 1 comprising generating k input bits, each input bit having a particular significance.

[0096] 18. The method of embodiment 17 comprising encoding the input bits using a $k \times n$ generator matrix to generate an output codeword, the generator matrix comprising a $k \times m$ maximum minimum Hamming distance code submatrix and a $k \times (n-m)$ submatrix with all '0s' but at least one element corresponding to a most significant bit (MSB) and the next MSB being set to '1'.

[0097] 19. The method of embodiment 18 comprising sending the output codeword.

[0098] 20. The method as in any one of embodiments 18-19, wherein the inputs bits are five (5) bits and the generator matrix is for (20,5) coding as follows:

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	1
0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	0
0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	0	0	0
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0	0	0

[0099] 21. The method as in any one of embodiments 18-19, wherein the inputs bits are eight (8) bits and the generator matrix is for (20,8) coding as follows:

```

1 0 0 0 0 0 0 0 1 1 1 0 0 1 1 0 0 1 1 1
0 1 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 0 0 1
0 0 1 0 0 0 0 0 1 0 0 1 1 1 0 1 0 0 0 0
0 0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 1 0 0 0
0 0 0 0 1 0 0 0 0 1 0 1 1 0 1 0 1 0 0 0
0 0 0 0 0 1 0 0 0 1 0 0 1 1 0 1 1 0 0 0
0 0 0 0 0 0 1 0 0 0 1 1 0 0 1 1 1 0 0 0
0 0 0 0 0 0 0 1 0 0 1 0 1 1 1 1 0 0 0 0

```

- [00100] 22. The method of embodiment 1 comprising generating input bits, each input bit having a particular significance.
- [00101] 23. The method of embodiment 22 comprising duplicating each of the input bits, the number of duplication of each input bit depending on the significance of each input bit.
- [00102] 24. The method of embodiment 23 comprising performing a linear block coding on the duplicated input bits to generate an output codeword using an equal protection code.
- [00103] 25. The method of embodiment 24 comprising sending the output codeword.
- [00104] 26. The method as in any one of embodiments 24-25, wherein the equal protection code is Reed-Muller code.
- [00105] 27. A method for encoding type B information bits.
- [00106] 28. The method of embodiment 27 comprising generating type B information bits including CQI bits and PCI bits.
- [00107] 29. The method of embodiment 28 comprising encoding the type B information bits using a (20, 7) code to generate an output codeword, a minimum distance of the (20, 7) code being eight (8), weight distribution of the (20, 7) code being non-optimal, and the (20, 7) code providing better protection to a most significant bit (MSB) of the CQI bits.
- [00108] 30. The method of embodiment 29 comprising sending the output codeword.
- [00109] 31. The method as in any one of embodiments 29-30, wherein the (20, 7) code is a non-subset code of (20, 10) code for type A information bits.

[00110] 32. The method as in any one of embodiments 29-30, wherein the (20, 7) code is a subset code of (20, 10) code for type A information bits with one bit flipping.

[00111] 33. The method as in any one of embodiments 29-30, wherein a generator matrix for the (20, 7) code is as follows:

```

1 0 0 0 0 0 0 1 1 0 0 0 1 1 1 0 1 0 1 0
0 1 0 0 0 0 0 1 0 1 1 0 0 1 0 0 1 1 1 0
0 0 1 0 0 0 0 1 0 1 0 1 1 0 0 1 1 0 1 0
0 0 0 1 0 0 0 1 0 1 0 0 1 1 1 1 0 1 0 0
0 0 0 0 1 0 0 1 0 0 1 1 1 0 1 0 1 1 0 0
0 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 1 0
0 0 0 0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1

```

[00112] 34. The method of embodiment 27 comprising generating type B information bits including CQI bits and PCI bits.

[00113] 35. The method of embodiment 34 comprising encoding the type B information bits using a (20, 7) code to generate an output codeword, a minimum distance of the (20, 7) code being eight (8), and the (20, 7) code providing equal protection to the type B information bits.

[00114] 36. The method of embodiment 35 comprising sending the output codeword.

[00115] 37. The method as in any one of embodiments 35-36, wherein a generator matrix for the (20, 7) code is as follows:

```

1 0 0 0 0 0 0 1 1 0 0 0 1 1 1 0 1 0 1 0
0 1 0 0 0 0 0 1 0 1 1 0 0 1 0 0 1 1 1 0
0 0 1 0 0 0 0 1 0 1 0 1 1 0 0 1 1 0 1 0
0 0 0 1 0 0 0 1 0 1 0 0 1 1 1 1 0 1 0 0
0 0 0 0 1 0 0 1 0 0 1 1 1 0 1 0 1 1 0 0
0 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 1 1
0 0 0 0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1

```

[00116] 38. The method of embodiment 27 comprising generating type B information bits including CQI bits and PCI bits.

[00117] 39. The method of embodiment 38 comprising encoding the type B information bits using a (20, 6) code to generate an output codeword, the (20, 6)

code providing approximately equal protection to the type B information bits.

[00118] 40. The method of embodiment 39 comprising sending the output codeword.

[00119] 41. The method as in any one of embodiments 39-40, wherein a generator matrix for the (20, 6) code is as follows:

1	0	0	0	0	0	1	1	1	0	1	1	1	0	1	0	0	0	1
0	1	0	0	0	0	1	0	0	1	1	1	0	0	1	0	1	0	0
0	0	1	0	0	0	0	1	0	1	1	0	1	0	1	0	0	1	0
0	0	0	1	0	0	0	0	1	1	1	0	0	1	1	0	0	0	1
0	0	0	0	1	0	1	1	1	1	0	0	0	0	0	1	1	1	1
0	0	0	0	0	1	0	0	0	0	0	1	1	1	1	1	1	1	1

[00120] 42. The method of embodiment 27 comprising generating type B information bits including CQI bits and PCI bits, each type B information bit having a particular significance.

[00121] 43. The method of embodiment 42 comprising duplicating at least one type B information bit based on the significance of the duplicated type B information bit.

[00122] 44. The method of embodiment 43 comprising encoding the type B information bits to generate an output codeword.

[00123] 45. The method of embodiment 44 comprising sending the output codeword.

[00124] 46. A method for encoding type A and type B information bits.

[00125] 47. The method of embodiment 46 comprising generating type A and type B information bits, the type A information bits including eight CQI bits and two PCI bits, and the type B information bits including five CQI bits and two PCI bits.

[00126] 48. The method of embodiment 47 comprising encoding the type A information bits using a (20, 10) code and encoding the type B information bits using a (20, 7) code to generate output codewords, the (20, 7) code being a subset code of the (20, 10) code, the (20, 10) code being generated by performing an elementary column operation on basis sequences as follows:

```

1 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 1
1 0 0 0 1 0 1 1 1 1
1 1 0 0 0 1 0 1 1 1
0 1 1 0 0 0 1 0 1 1
1 0 1 1 0 0 0 1 0 1
1 1 0 1 1 0 0 0 1 0
1 1 1 0 1 1 0 0 0 1
1 1 1 1 0 1 1 0 0 0
0 1 1 1 1 0 1 1 0 0
1 0 1 1 1 1 0 1 1 0
0 1 0 1 1 1 1 0 1 1

```

[00127] 49. The method of embodiment 48 comprising sending the codeword.

[00128] 50. The method as in any one of embodiments 48-49, wherein a generator matrix for the (20, 10) code is as follows:

```

1 0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 1 1 0
0 1 0 0 0 0 0 0 0 1 1 0 0 1 0 1 0 1 1 1
0 0 1 0 0 0 0 0 0 1 1 0 0 0 1 1 1 0 1 0
0 0 0 1 0 0 0 0 0 1 0 1 1 0 0 1 0 0 1 1
0 0 0 0 1 0 0 1 1 0 1 0 0 1 0 0 0 1 0 1
0 0 0 0 0 1 0 1 0 0 0 1 1 1 1 0 0 0 0 1
0 0 0 0 0 0 1 0 0 1 0 0 1 1 1 0 1 0 1 1
0 0 0 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0
0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 1 1 0

```

[00129] 51. The method as in any one of embodiments 48-49, wherein a

generator matrix for the (20, 10) code is as follows:

1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0	1	0
0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0	1
0	0	1	0	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	1	1
0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1
0	0	0	0	1	0	1	1	0	0	1	1	1	1	1	1	1	1	1	0	0
0	0	0	0	0	1	0	1	0	1	0	1	1	0	0	0	1	1	0	0	0
0	0	0	0	0	0	1	1	1	1	1	0	1	1	0	0	0	1	0	0	0
0	0	0	0	0	0	0	1	0	0	1	1	0	1	0	0	0	1	1	0	0
0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	0	1	1	0	1	0
0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	0	1	1	0	1	0
0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	1	0

[00130] 52. The method as in any one of embodiments 48-49, wherein a generator matrix for the (20, 10) code is as follows:

1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	1	1	1	0
0	1	0	0	0	0	0	0	1	1	0	0	0	1	1	1	0	1	0	1
0	0	1	0	0	0	0	0	1	0	1	1	0	0	1	0	0	1	1	1
0	0	0	1	0	0	0	0	1	0	1	0	1	1	0	0	1	1	0	1
0	0	0	0	1	0	0	0	1	0	1	0	0	1	1	1	1	0	1	0
0	0	0	0	0	1	0	0	1	0	0	1	1	1	0	1	0	1	1	0
0	0	0	0	0	0	1	0	1	0	0	1	1	0	1	1	1	0	0	1
0	0	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	1	0	0	1	1	1	0	1	1	0	0	1	0	1	0
0	0	0	0	0	0	1	0	1	0	1	1	1	1	1	0	1	1	1	0

[00131] 53. The method as in any one of embodiments 48-49, wherein a generator matrix for the (20, 10) code is as follows:

1	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0	1	0
0	1	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0	1
0	0	1	0	0	0	1	1	0	1	1	0	1	1	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1
0	0	0	0	1	0	1	1	0	0	1	1	1	1	1	1	1	1	0	0
0	0	0	0	0	1	0	1	0	1	0	1	1	0	0	0	1	1	0	0
0	0	0	0	0	0	0	0	1	0	1	1	1	0	1	0	0	0	1	1
0	0	0	0	0	0	0	1	0	0	1	1	0	1	0	0	0	1	1	0
0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	0	1	1	0	1
0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	1

[00132] 54. The method as in any one of embodiments 48-49, wherein a generator matrix for the (20, 10) code is as follows:

1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0	1	0
0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0	1
0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1
0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	1	0	1	1	1
0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	1	0	1	1
0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	1	0	1	1
0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	1	1	0	1
0	0	0	0	0	0	0	0	1	0	0	1	1	0	1	0	0	0	1	1	0
0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	1	0	0	0	1	1
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	1

[00133] 55. The method as in any one of embodiments 48-49, wherein a generator matrix for the (20, 10) code is as follows:

1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0	1	0
0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0	1
0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1
0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	1	0	1	1	1
0	0	0	0	0	1	0	0	0	0	1	0	1	0	0	0	1	1	0	1	1
0	0	0	0	0	0	1	1	1	0	1	0	1	0	0	0	1	1	0	1	1
0	0	0	0	0	0	0	1	0	0	1	1	0	1	0	0	0	1	1	0	1
0	0	0	0	0	0	0	0	1	0	1	1	1	0	1	0	0	0	1	1	0
0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	1	0	0	0	1	1
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	1

[00134] 56. The method as in any one of embodiments 48-49, wherein a generator matrix for the (20, 10) code is as follows:

```

1 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 0 1 0
0 1 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 0 1
0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 0
0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1
0 0 0 0 1 0 0 1 0 0 1 0 0 0 1 1 0 1 1 1
0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 1 1 0 1 1
0 0 0 0 0 0 1 0 0 1 1 0 1 0 0 0 1 1 0 1
0 0 0 0 0 0 0 1 0 0 1 1 0 1 0 0 0 1 1 0
0 0 0 0 0 0 0 0 1 0 0 1 1 0 1 0 0 0 1 1
0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 1 0 0 0 1
0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 1 0 0 0 1

```

[00135] 57. The method as in any one of embodiments 48-56, wherein the type A information bits include two sets of four (4)-bit CQI bits concatenated.

[00136] 58. The method as in any one of embodiments 48-56, wherein the type A information bits include two sets of four (4)-bit CQI bits, and the two sets of CQI bits are combined.

[00137] 59. The method as in any one of embodiments 48-49, wherein a generator matrix for the (20, 10) code is as follows:

```

1 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 0 1 0
0 1 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 0 1
0 0 1 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 1
0 0 0 1 0 0 0 1 0 0 1 1 1 1 0 0 1 1 0 0
0 0 0 0 1 0 0 1 1 0 1 1 0 0 0 1 1 0 0 1
0 0 0 0 0 1 0 0 1 0 1 1 1 0 0 1 0 1 1 0
0 0 0 0 0 0 1 1 0 1 1 1 0 0 1 1 0 1 0 0
0 0 0 0 0 0 0 1 0 0 1 1 1 0 1 0 0 0 1 1
0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 1 1 1 0 0
0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 1 0 0 0 1

```

[00138] 60. The method as in any one of embodiments 48-59, wherein different transmit powers are used for type A and type B information bits.

[00139] 61. The method of embodiment 60 wherein transmit powers for type A and type B information bits are defined independently in terms of a reference channel.

[00140] 62. The method of embodiment 61 wherein the reference channel is one of a CPICH, a P-CCPCH, an SCH, an HS-SCCH, a DPCCH, an E-DPCCH,

and a RACH.

[00141] 63. The method of embodiment 60 wherein a transmit power for one of the type A and type B information bits is defined independently in terms of a reference channel and a transmit power for the other information bits is determined as a network supplied formula.

[00142] 64. The method of embodiment 60 wherein a transmit power for the type B information bits is computed using a look up table (LUT_A).

[00143] 65. A WTRU for encoding information bits.

[00144] 66. The WTRU of embodiment 65 comprising a data generator for generating input bits, each input bit having a particular significance.

[00145] 67. The WTRU of embodiment 66 comprising an encoder for performing a linear block coding on the input bits to generate an output codeword, the input bits being provided with a specific level of error protection based on the significance of each input bit.

[00146] 68. The WTRU of embodiment 67 comprising a transmitter for sending the output codeword.

[00147] 69. The WTRU as in any one of embodiments 66-68, wherein the data generator includes a CQI generator for generating at least one set of CQI bits as the input bits.

[00148] 70. The WTRU of embodiment 69 wherein the CQI generator generates multiple sets of CQI bits, one of the multiple sets of CQI bits being denoted as primary CQI bits and the remaining sets of CQI bits being denoted as secondary CQI bits and concatenates the primary CQI bits and the secondary CQI bits as the input bits for encoding.

[00149] 71. The WTRU of embodiment 69 wherein the CQI generator generates multiple sets of CQI bits, one of the multiple sets of CQI bits being denoted as primary CQI bits and the remaining sets of CQI bits being denoted as secondary CQI bits, combines the secondary CQI bits with the primary CQI bits and concatenates the primary CQI bits and the combined secondary CQI bits as the input bits for encoding.

[00150] 72. The WTRU as in any one of embodiments 66-71, wherein the

input bits are five (5) bits, and the generator matrix is for (20,5) coding as follows:

```

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0
1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0
1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0

```

[00151] 73. The WTRU as in any one of embodiments 66-71, wherein the inputs bits are five (5) bits, and the generator matrix is for (20,5) coding as follows:

```

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0
1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0

```

[00152] 74. The WTRU as in any one of embodiments 66-69, wherein the data generator comprises a CQI generator for generating at least one set of CQI bits and a PCI generator for generating PCI bits, wherein the CQI bits and the PCI bits are concatenated as the input bits.

[00153] 75. The WTRU of embodiment 74 wherein the input bit generator duplicates at least one input bit depending on the significance of the input bit.

[00154] 76. The WTRU as in any one of embodiments 66-75, wherein the input bits are at least two different information bits that are concatenated and need to be transmitted in different timing.

[00155] 77. The WTRU of embodiment 76 wherein the inputs bits are CQI bits and acknowledgement bits.

[00156] 78. The WTRU of embodiment 65 comprising a data generator for generating k input bits, each input bit having a particular significance.

[00157] 79. The WTRU of embodiment 78 comprising an encoder for encoding the input bits using a k×n generator matrix to generate an output codeword, the generator matrix comprising a k×m maximum minimum Hamming distance code submatrix and a k×(n-m) submatrix with all '0s' but at least one

element corresponding to a MSB and the next MSB being set to '1'.

[00158] 80. The WTRU of embodiment 79 comprising a transmitter for sending the output codeword.

[00159] 81. The WTRU as in any one of embodiments 79-80, wherein the inputs bits are five (5) bits and the generator matrix is for (20,5) coding as follows:

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	1
0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	0
0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	0	0	0
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0	0	0

[00160] 82. The WTRU as in any one of embodiments 79-80, wherein the inputs bits are eight (8) bits and the generator matrix is for (20,8) coding as follows:

1	0	0	0	0	0	0	0	1	1	1	0	0	1	1	0	0	1	1	1
0	1	0	0	0	0	0	0	1	1	1	0	0	0	0	1	1	0	0	1
0	0	1	0	0	0	0	0	1	0	0	1	1	1	0	1	0	0	0	0
0	0	0	1	0	0	0	0	1	0	0	1	0	1	1	0	1	0	0	0
0	0	0	0	1	0	0	0	0	1	0	1	1	0	1	0	1	0	0	0
0	0	0	0	0	1	0	0	0	1	0	0	1	1	0	1	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1	1	0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	1	0	0	1	0	1	1	1	1	0	0	0	0

[00161] 83. The WTRU of embodiment 65 comprising a data generator for generating input bits, each input bit having a particular significance and duplicating each of the input bits, the number of duplication of each input bit depending on the significance of each input bit.

[00162] 84. The WTRU of embodiment 83 comprising an encoding for performing a linear block coding on the duplicated input bits to generate an output codeword using an equal protection code.

[00163] 85. The WTRU of embodiment 84 comprising a transmitter for sending the output codeword.

[00164] 86. The WTRU as in any one of embodiments 84-85, wherein the equal protection code is Reed-Muller code.

[00165] 87. The WTRU of embodiment 65 comprising a data generator for generating type B information bits including CQI bits and PCI bits.

[00166] 88. The WTRU of embodiment 87 comprising an encoder for encoding the type B information bits using a (20, 7) code to generate an output codeword, a minimum distance of the (20, 7) code being eight (8), weight distribution of the (20, 7) code being non-optimal, and the (20, 7) code providing better protection to a most significant bit (MSB) of the CQI bits.

[00167] 89. The WTRU of embodiment 88 comprising a transmitter for sending the output codeword.

[00168] 90. The WTRU as in any one of embodiments 88-89, wherein the (20, 7) code is a non-subset code of (20, 10) code for type A information bits.

[00169] 91. The WTRU as in any one of embodiments 88-89, wherein the (20, 7) code is a subset code of (20, 10) code for type A information bits with one bit flipping.

[00170] 92. The WTRU as in any one of embodiments 88-89, wherein a generator matrix for the (20, 7) code is as follows:

1	0	0	0	0	0	0	1	1	0	0	0	1	1	1	0	1	0	1	0
0	1	0	0	0	0	0	1	0	1	1	0	0	1	0	0	1	1	1	0
0	0	1	0	0	0	0	1	0	1	0	1	1	0	0	1	1	0	1	0
0	0	0	1	0	0	0	1	0	1	0	0	1	1	1	1	0	1	0	0
0	0	0	0	1	0	0	1	0	0	1	1	1	0	1	0	1	1	0	0
0	0	0	0	0	1	0	1	0	0	1	1	0	1	1	1	0	0	1	0
0	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1

[00171] 93. The WTRU of embodiment 65 comprising a data generator for generating type B information bits including CQI bits and PCI bits.

[00172] 94. The WTRU of embodiment 93 comprising an encoder for encoding the type B information bits using a (20, 7) code to generate an output codeword, a minimum distance of the (20, 7) code being eight (8), and the (20, 7) code providing equal protection to the type B information bits.

[00173] 95. The WTRU of embodiment 94 comprising a transmitter for sending the output codeword.

[00174] 96. The WTRU as in any one of embodiments 94-95, wherein a

generator matrix for the (20, 7) code is as follows:

```

1 0 0 0 0 0 0 1 1 0 0 0 1 1 1 0 1 0 1 0
0 1 0 0 0 0 0 1 0 1 1 0 0 1 0 0 1 1 1 0
0 0 1 0 0 0 0 1 0 1 0 1 1 0 0 1 1 0 1 0
0 0 0 1 0 0 0 1 0 1 0 0 1 1 1 1 0 1 0 0
0 0 0 0 1 0 0 1 0 0 1 1 1 0 1 0 1 1 0 0
0 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 1 1
0 0 0 0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1

```

[00175] 97. The WTRU of embodiment 65 comprising a data generator for generating type B information bits including CQI bits and PCI bits.

[00176] 98. The WTRU of embodiment 97 comprising an encoder for encoding the type B information bits using a (20, 6) code to generate an output codeword, the (20, 6) code providing approximately equal protection to the type B information bits.

[00177] 99. The WTRU of embodiment 98 comprising a transmitter for sending the output codeword.

[00178] 100. The WTRU as in any one of embodiments 98-99, wherein a generator matrix for the (20, 6) code is as follows:

```

1 0 0 0 0 0 1 1 1 0 1 1 1 0 1 0 0 0 0 1
0 1 0 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1
0 0 1 0 0 0 0 1 0 1 1 0 1 0 1 0 0 1 0 1
0 0 0 1 0 0 0 0 1 1 1 0 0 1 1 0 0 0 1 1
0 0 0 0 1 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1
0 0 0 0 0 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1

```

[00179] 101. The WTRU of embodiment 65 comprising a data generator for generating type B information bits including CQI bits and PCI bits, each type B information bit having a particular significance, and duplicating at least one type B information bit based on the significance of the duplicated type B information bit.

[00180] 102. The WTRU of embodiment 101 comprising an encoder for encoding the type B information bits to generate an output codeword.

[00181] 103. The WTRU of embodiment 102 comprising a transmitter for sending the output codeword.

[00182] 104. A WTRU for encoding type A and type B information bits.

[00183] 105. The WTRU of embodiment 104 comprising a data generator for generating type A and type B information bits, the type A information bits including eight CQI bits and two PCI bits, and the type B information bits including five CQI bits and two PCI bits.

[00184] 106. The WTRU of embodiment 105 comprising an encoder for encoding the type A information bits using a (20, 10) code and encoding the type B information bits using a (20, 7) code to generate output codewords, the (20, 7) code being a subset code of the (20, 10) code, the (20, 10) code being generated by performing an elementary column operation on basis sequences as follows:

1	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	0	1	1	1	1
1	1	0	0	0	1	0	1	1	1
0	1	1	0	0	0	1	0	1	1
1	0	1	1	0	0	0	1	0	1
1	1	0	1	1	0	0	0	1	0
1	1	1	0	1	1	0	0	0	1
1	1	1	1	0	1	1	0	0	0
0	1	1	1	1	0	1	1	0	0
1	0	1	1	1	1	0	1	1	0
0	1	0	1	1	1	1	0	1	1

[00185] 107. The WTRU of embodiment 106 comprising a transmitter for sending the codeword.

[00186] 108. The WTRU as in any one of embodiments 106-107, wherein a generator matrix for the (20, 10) code is as follows:

1	0	0	0	0	0	0	1	1	0	0	1	1	0	0	0	0	1	1	0
0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	1	1	1
0	0	1	0	0	0	0	0	0	1	1	0	0	0	1	1	1	0	1	0
0	0	0	1	0	0	0	0	0	1	0	1	1	0	0	1	0	0	1	1
0	0	0	0	1	0	0	1	1	0	1	0	0	1	0	0	0	1	0	1
0	0	0	0	0	1	0	1	0	0	0	1	1	1	1	0	0	0	0	1
0	0	0	0	0	0	1	0	0	1	0	0	1	1	1	0	1	0	1	1
0	0	0	0	0	0	0	1	0	1	0	0	1	1	0	1	1	1	0	0
0	0	0	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0

[00187] 109. The WTRU as in any one of embodiments 106-107, wherein a generator matrix for the (20, 10) code is as follows:

1	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0	1	0
0	1	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0	1
0	0	1	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	1	1
0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1
0	0	0	0	1	0	1	1	0	0	1	1	1	1	1	1	1	1	0	0
0	0	0	0	0	1	0	1	0	1	0	1	1	0	0	0	1	1	0	0
0	0	0	0	0	0	1	1	1	1	1	0	1	1	0	0	0	1	0	0
0	0	0	0	0	0	0	1	0	0	1	1	0	1	0	0	0	1	1	0
0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	0	1	1	0	1
0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	1

[00188] 110. The WTRU as in any one of embodiments 106-107, wherein a generator matrix for the (20, 10) code is as follows:

1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	1	1	1	0
0	1	0	0	0	0	0	0	1	1	0	0	0	1	1	1	0	1	0	1
0	0	1	0	0	0	0	0	1	0	1	1	0	0	1	0	0	1	1	1
0	0	0	1	0	0	0	0	1	0	1	0	1	1	0	0	1	1	0	1
0	0	0	0	1	0	0	0	1	0	1	0	0	1	1	1	1	0	1	0
0	0	0	0	0	1	0	0	1	0	0	1	1	1	0	1	0	1	1	0
0	0	0	0	0	0	1	0	1	0	0	1	1	0	1	1	1	0	0	1
0	0	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	1	0	0	1	1	1	0	1	1	0	0	1	0	1	0
0	0	0	0	0	0	1	0	1	0	1	1	1	1	0	1	1	1	0	

[00189] 111. The WTRU as in any one of embodiments 106-107, wherein a

generator matrix for the (20, 10) code is as follows:

1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0	1	0
0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0	1
0	0	1	0	0	0	1	1	0	1	1	0	1	1	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1
0	0	0	0	1	0	1	1	0	0	1	1	1	1	1	1	1	1	1	0	0
0	0	0	0	0	1	0	1	0	1	0	1	1	0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	0	1	0	1	1	1	0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	1	0	0	1	1	0	1	0	0	0	1	1	0	0
0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	0	1	1	0	1	0
0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	0	1	1	0	1	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	1

[00190] 112. The WTRU as in any one of embodiments 106-107, wherein a generator matrix for the (20, 10) code is as follows:

1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0	1	0
0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0	1
0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1
0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	1	0	1	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	1	0	1	1	1
0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	1	1	0	1	0
0	0	0	0	0	0	0	1	0	0	1	1	0	1	0	0	0	1	1	0	0
0	0	0	0	0	0	0	0	1	0	1	1	1	0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	1

[00191] 113. The WTRU as in any one of embodiments 106-107, wherein a generator matrix for the (20, 10) code is as follows:

1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0	1	0
0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0	1
0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1
0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	1	0	1	1	1
0	0	0	0	0	1	0	0	0	1	0	1	0	0	0	1	1	0	1	1	1
0	0	0	0	0	0	1	1	1	0	1	0	1	0	0	0	1	1	0	1	0
0	0	0	0	0	0	0	1	0	0	1	1	0	1	0	0	0	1	1	0	0
0	0	0	0	0	0	0	0	1	0	1	1	1	0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	1

[00192] 114. The WTRU as in any one of embodiments 106-107, wherein a generator matrix for the (20, 10) code is as follows:

```

1 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 0 1 0
0 1 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 0 1
0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 0
0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1
0 0 0 0 1 0 0 1 0 0 1 0 0 0 1 1 0 1 1 1
0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 1 1 0 1 1
0 0 0 0 0 0 1 0 0 1 1 0 1 0 0 0 1 1 0 1
0 0 0 0 0 0 0 1 0 0 1 1 0 1 0 0 0 1 1 0
0 0 0 0 0 0 0 0 1 0 1 1 1 0 1 0 0 0 1 1
0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 1 0 0 0 1

```

[00193] 115. The WTRU as in any one of embodiments 106-114, wherein the type A information bits include two sets of four (4)-bit CQI bits concatenated.

[00194] 116. The WTRU as in any one of embodiments 106-114, wherein the type A information bits include two sets of four (4)-bit CQI bits, and the two sets of CQI bits are combined.

[00195] 117. The WTRU as in any one of embodiments 106-107, wherein a generator matrix for the (20, 10) code is as follows:

```

1 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 0 1 0
0 1 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 0 1
0 0 1 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 1
0 0 0 1 0 0 0 1 0 0 1 1 1 1 0 0 1 1 0 0
0 0 0 0 1 0 0 1 1 0 1 1 0 0 0 1 1 0 0 1
0 0 0 0 0 1 0 0 1 0 1 1 1 0 0 1 0 1 1 0
0 0 0 0 0 0 1 1 0 1 1 1 0 0 1 1 0 1 0 0
0 0 0 0 0 0 0 1 0 0 1 1 1 0 1 0 0 0 1 1
0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 1 1 1 0 0
0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 1 0 0 0 1

```

[00196] 118. The WTRU as in any one of embodiments 106-117, wherein different transmit powers are used for type A and type B information bits.

[00197] 119. The WTRU of embodiment 118 wherein transmit powers for type A and type B information bits are defined independently in terms of a

reference channel.

[00198] 120. The WTRU of embodiment 119 wherein the reference channel is one of a CPICH, a P-CCPCH, an SCH, an HS-SCCH, a DPCCH, an E-DPCCH, and a RACH.

[00199] 121. The WTRU of embodiment 118 wherein a transmit power for one of the type A and type B information bits is defined independently in terms of a reference channel and a transmit power for the other information bits is determined as a network supplied formula.

[00200] 122. The WTRU of embodiment 121 wherein a transmit power for the type B information bits is computed using a look up table (LUT_A).

[00201] 123. A method for encoding CQI bits in a WTRU.

[00202] 124. The method of embodiment 123 comprising generating CQI bits.

[00203] 125. The method of embodiment 124 comprising encoding the CQI bits.

[00204] 126. The method of embodiment 125 comprising applying a gain factor β_{hs} to the CQI bits.

[00205] 127. The method of embodiment 126 comprising sending the encoded CQI bits, wherein the gain factor β_{hs} for the CQI bits is derived from a quantized amplitude ratio ($A_{hs} = \beta_{hs}/\beta_c$), which is translated from Δ_{CQI} , β_c being a gain factor for a DPCCH, and the translation of Δ_{CQI} into the quantized amplitude ratio A_{hs} is performed in accordance with the following table:

Signalled values for Δ_{CQI}	Quantized amplitude ratios $A_{hs} = \beta_{hs}/\beta_c$
9	38/15
8	30/15
7	24/15
6	19/15
5	15/15
4	12/15
3	9/15
2	8/15
1	6/15
0	5/15

[00206] 128. The method of embodiment 127 wherein for time slots carrying the CQI bits, if the WTRU is not configured to a MIMO mode, A_{hs} equals the quantized amplitude ratio translated from the signalled value Δ_{CQI} .

[00207] 129. The method of embodiment 127 wherein for time slots carrying the CQI bits, if the WTRU is configured to a MIMO mode, A_{hs} equals the quantized amplitude ratio translated from the signalled value Δ_{CQI} when a CQI of type B is transmitted, and A_{hs} equals the quantized amplitude ratio translated from the signalled value $\Delta_{CQI} + 1$ when a CQI of type A is transmitted.

[00208] 130. A WTRU for encoding CQI bits.

[00209] 131. The WTRU of embodiment 130 comprising a CQI generator for generating CQI bits.

[00210] 132. The WTRU of embodiment 131 comprising an encoder for encoding the CQI bits.

[00211] 133. The WTRU of embodiment 132 comprising a transmitter for applying a gain factor β_{hs} to the CQI bits and sending the encoded CQI bits, wherein the gain factor β_{hs} for the CQI bits is derived from a quantized amplitude ratio ($A_{hs} = \beta_{hs}/\beta_c$), which is translated from Δ_{CQI} , β_c being a gain factor for a DPCCH, and the translation of Δ_{CQI} into the quantized amplitude ratio A_{hs} is performed in accordance with the following table:

Signalled values for Δ_{CQI}	Quantized amplitude ratios $A_{hs} = \beta_{hs}/\beta_c$
9	38/15
8	30/15
7	24/15
6	19/15
5	15/15
4	12/15
3	9/15
2	8/15
1	6/15
0	5/15

[00212] 134. The WTRU of embodiment 133 wherein for time slots carrying the CQI bits, if the WTRU is not configured to a MIMO mode, A_{hs} equals the quantized amplitude ratio translated from the signalled value Δ_{CQI} .

[00213] 135. The WTRU of embodiment 133 wherein for time slots carrying the CQI bits, if the WTRU is configured to a MIMO mode, A_{hs} equals the quantized amplitude ratio translated from the signalled value Δ_{CQI} when a CQI of type B is transmitted, and A_{hs} equals the quantized amplitude ratio translated from the signalled value $\Delta_{CQI} + 1$ when a CQI of type A is transmitted.

[00214] Although the features and elements are described in the preferred embodiments in particular combinations, each feature or element can be used alone without the other features and elements of the preferred embodiments or in various combinations with or without other features and elements of the present invention. The methods or flow charts provided in the present invention may be implemented in a computer program, software, or firmware tangibly embodied in a computer-readable storage medium for execution by a general purpose computer or a processor. Examples of computer-readable storage mediums include a read only memory (ROM), a random access memory (RAM), a register, cache memory, semiconductor memory devices, magnetic media such as internal hard disks and removable disks, magneto-optical media, and optical media such as CD-ROM disks, and digital versatile disks (DVDs).

[00215] Suitable processors include, by way of example, a general purpose processor, a special purpose processor, a conventional processor, a digital signal processor (DSP), a plurality of microprocessors, one or more microprocessors in association with a DSP core, a controller, a microcontroller, Application Specific Integrated Circuits (ASICs), Field Programmable Gate Arrays (FPGAs) circuits, any other type of integrated circuit (IC), and/or a state machine.

[00216] A processor in association with software may be used to implement a radio frequency transceiver for use in a wireless transmit receive unit (WTRU), user equipment (UE), terminal, base station, radio network controller (RNC), or any host computer. The WTRU may be used in conjunction with modules, implemented in hardware and/or software, such as a camera, a video camera module, a videophone, a speakerphone, a vibration device, a speaker, a microphone, a television transceiver, a hands free headset, a keyboard, a Bluetooth® module, a frequency modulated (FM) radio unit, a liquid crystal

display (LCD) display unit, an organic light-emitting diode (OLED) display unit, a digital music player, a media player, a video game player module, an Internet browser, and/or any wireless local area network (WLAN) module.

*

*

*

Claims

1. A method for encoding channel quality indicator (CQI) bits in a wireless transmit/receive unit (WTRU), the method comprising:

generating CQI bits;

encoding the CQI bits;

applying a gain factor β_{hs} to the CQI bits; and

sending the encoded CQI bits, wherein the gain factor β_{hs} for the CQI bits is derived from a quantized amplitude ratio ($A_{hs} = \beta_{hs}/\beta_c$), β_c being a gain factor for a dedicated physical control channel (DPCCH), and wherein if the WTRU is configured in multiple-input multiple-output (MIMO) mode, A_{hs} equals the quantized amplitude ratio translated from a signaled value $\Delta CQI + 1$ when a CQI of type A is transmitted.

2. The method of claim 1 wherein for time slots carrying the CQI bits, if the WTRU is not configured in multiple-input multiple-output (MIMO) mode, A_{hs} equals the quantized amplitude ratio translated from the signaled value ΔCQI .

3. The method of claim 1 wherein for time slots carrying the CQI bits, if the WTRU is configured in multiple-input multiple-output (MIMO) mode, A_{hs} equals the quantized amplitude ratio translated from the signaled value ΔCQI when a CQI of type B is transmitted.

4. The method of claim 1, wherein β_{hs} is a gain factor for a high speed dedicated physical control channel (HS-DPCCH).

5. A wireless transmit/receive unit (WTRU) for encoding channel quality indicator (CQI) bits, the WTRU comprising:

a CQI generator for generating CQI bits;

an encoder for encoding the CQI bits; and

a transmitter for applying a gain factor β_{hs} to the CQI bits and sending the encoded CQI bits, wherein the gain factor β_{hs} for the CQI bits is derived from a quantized amplitude ratio ($A_{hs} = \beta_{hs}/\beta_c$), β_c being a gain factor for a dedicated physical control channel (DPCCH), and wherein A_{hs}

equals to the quantized amplitude ratio translated from a signaled value $\Delta_{CQI} + 1$ when a CQI of type A is transmitted.

6. The WTRU of claim 5 wherein for time slots carrying the CQI bits, if the WTRU is not configured in multiple-input multiple-output (MIMO) mode, A_{hs} equals the quantized amplitude ratio translated from the signaled value Δ_{CQI} .

7. The WTRU of claim 5 wherein for time slots carrying the CQI bits, if the WTRU is configured in multiple-input multiple-output (MIMO) mode, A_{hs} equals the quantized amplitude ratio translated from the signaled value Δ_{CQI} when a CQI of type B is transmitted.

8. The WTRU of claim 5, wherein β_{hs} is a gain factor for a high speed dedicated physical control channel (HS-DPCCH).