Project	**IEEE 802.16 Broadband Wireless Access Working Group <http://ieee802.org/16>**
Title	**Proposal to Amend Section 8.3.5.5.2.1 of TG3&4 Draft Document**
Date Submitted	**2001-07-06**
Source(s)	Brian Eidson Conexant Systems, Inc. 9868 Scranton Rd San Diego, CA 92121 Voice: (858) 713-4720 Fax: (858) 713-3555 mailto: brian.eidson@conexant.com
Re:	TG3&4 Draft document
Abstract	This document proposes a new section 8.3.5.5.2.1 to replace the old one in the 802.16ab draft document. Section 8.3.5.5.2.1 describes the Concatenated Reed Solomon + Convolutional FEC within the Single Carrier mode.
Purpose	To propose replacement of section 8.3.5.5.2.1 and its dependent, descending subsections with the contents of this document.
Notice	This document has been prepared to assist IEEE 802.16. It is offered as a basis for discussion and is not binding on the contributing individual(s) or organization(s). The material in this document is subject to change in form and content after further study. The contributor(s) reserve(s) the right to add, amend or withdraw material contained herein.
Release	The contributor grants a free, irrevocable license to the IEEE to incorporate material contained in this contribution, and any modifications thereof, in the creation of an IEEE Standards publication; to copyright in the IEEE's name any IEEE Standards publication even though it may include portions of this contribution; and at the IEEE's sole discretion to permit others to reproduce in whole or in part the resulting IEEE Standards publication. The contributor also acknowledges and accepts that this contribution may be made public by IEEE 802.16.
Patent Policy and Procedures	The contributor is familiar with the IEEE 802.16 Patent Policy and Procedures (Version 1.0) <http://ieee802.org/16/ipr/patents/policy.html>, including the statement "IEEE standards may include the known use of patent(s), including patent applications, if there is technical justification in the opinion of the standards-developing committee and provided the IEEE receives assurance from the patent holder that it will license applicants under reasonable terms and conditions for the purpose of implementing the standard." Early disclosure to the Working Group of patent information that might be relevant to the standard is essential to reduce the possibility for delays in the development process and increase the likelihood that the draft publication will be approved for publication. Please notify the Chair <mailto:r.b.marks@ieee.org> as early as possible, in written or electronic form, of any patents (granted or under application) that may cover technology that is under consideration by or has been approved by IEEE 802.16. The Chair will disclose this notification via the IEEE 802.16 web site <http://ieee802.org/16/ipr/patents/notices>.

Proposal to Amend Section 8.3.5.5.2.1 of TG3&4 Draft Document

Brian Eidson
Conexant Systems, Inc.

8.3.5.5.2.1.1 Concatenated Reed-Solomon + Convolutional Code Details

As indicated 8.3.5.5.2, a standard FEC for the single carrier downlink (and uplink) is the concatenation of an outer Reed Solomon code and an inner trellis code derived from a convolutional code. Figure 1 illustrates the general topology for the concatenated FEC. As Figure 1 demonstrates, source bits are encoded by the outer encoder, optionally interleaved (in the downlink case), encoded again by the inner encoder, and mapped to the I and Q components of complex signaling constellation using a bits-to-constellation signal map. Details on the outer FEC are found in 8.3.5.5.2.1.1; the inner FEC, in 8.3.5.5.2.1.2; the optional interleaver in 8.3.5.5.2.1.3; and the bits-to-symbols maps (for BPSK, QPSK, 16-QAM, 64-QAM, and 256-QAM) in 8.3.5.5.2.1.4.

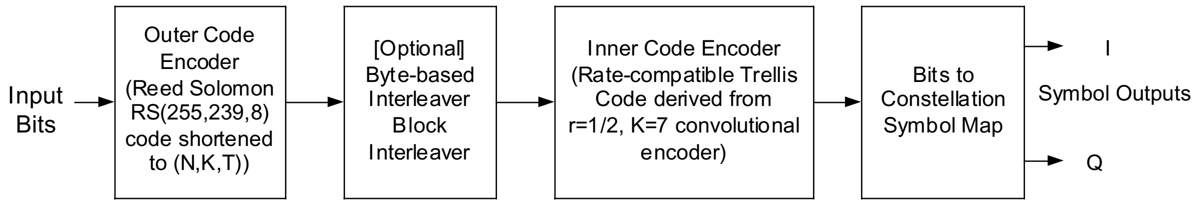
Input Bits
Outer Code Encoder (Reed Solomon RS(255,239,8) code shortened to (N,K,T))
[Optional] Byte-based Interleaver Block Interleaver
Inner Code Encoder (Rate-compatible Trellis Code derived from r=1/2, K=7 convolutional encoder)
Bits to Constellation Symbol Map
I
Symbol Outputs
Q

Figure 1: General topology for concatenated FEC used by single carrier mode

8.3.5.5.2.1.1 Outer FEC

The outer FEC is based on a systematic Reed Solomon RS(255,239, T=8) code. As this description implies, the baseline Reed-Solomon encoder takes a source block of 239 bytes and encodes it into the code block of 255 bytes, with a error correction capability of T=8 bytes. Since the option exists to independently encode user allocations, and many allocations may require fractions of the base block size to be delivered, the baseline RS(255,239, T=8) code may be shortened and punctured to an arbitrary RS(N,K,T) at the end of an allocation, to encapsulate a fractionally-sized block.

The following polynomials are used for the systematic RS(255,239,8) code:

- Code generator polynomial: $g(x) = (x+\lambda^0)(x+\lambda^1)(x+\lambda^2)...(x+\lambda^{2T-1}), \lambda = 02_{hex}$
- Field Generator polynomial: $p(x) = x^8 + x^4 + x^3 + x^2 + 1$

8.3.5.5.2.1.2 Inner FEC

The inner FEC is a rate-compatible pragmatic TCM (Trellis Coded Modulation) code. References for pragmatic TCM include [1] and [2]. The modulation types and code rates supported by this inner FEC are listed in Table 1.

Modulation	Code Rates	Bits/symbol
BPSK [optional]	_, _	1/2, 3/4
QPSK	_,2/3,3/4,5/6,7/8	1, 4/3, 3/2, 5/3, 7/4
16-QAM	_, _	2, 3
64-QAM	2/3, 5/6	4, 5
256-QAM [optional]	_, 7/8	6, 7

Table 1: Modulations and Code Rates for Single Carrier Mode

The rate-compatible pragmatic TCM encoder is constructed from a nonsystematic rate _, K=7 binary convolutional code. As its description implies, the binary convolutional code generates two bit outputs, X, and Y, per every bit input. The generator polynomial used to realize the X and Y outputs are:

$$G_1 = 171_{oct} \text{ For X}$$
$$G_2 = 133_{oct} \text{ For Y}$$

An apparatus which can be used to generate the rate _ encoder is illustrated in Figure 2.

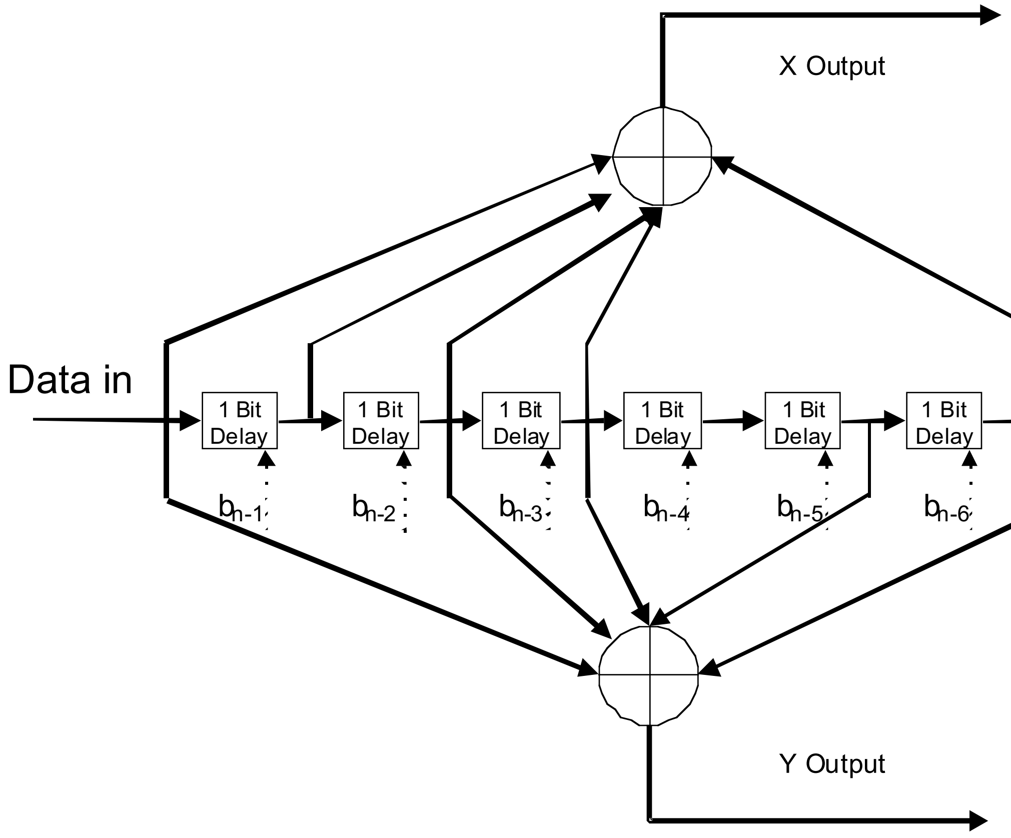

Figure 2: Rate _ binary convolutional encoder

Binary code rates of 2/3, _, 5/6, and 7/8 are required in some implementations of the pragmatic trellis encoder. To realize these higher binary code rates, the baseline rate 1/2 encoder outputs may be punctured. The puncture patterns which shall be used to realize the punctured code rates are defined in Table 2. To interpret this table, note that the notation used by Figure 2 is used, with (X, Y) indicating the bit pairs at the output of the binary convolutional encoder. 'X' indicates the top 171 (octal) polynomial output of the encoder, and Y indicates the bottom 133 (octal) output of the encoder. What's more, a '1' indication within a puncture pattern mask implies a transmitted bit while a '0' indication implies a non-transmitted, punctured bit.

Original Code			Code rates									
			_		2/3		3/4		5/6		7/8	
K	G1 (X)	G2 (Y)	P	d_{free}	P	d_{free}	P	d_{free}	P	d_{free}	P	d_{free}

7	171	133	X: 1 Y: 1 I=X_1 Q=Y_1	10	X: 10 Y: 11 I= $X_1Y_2Y_3$ Q= $Y_1X_3Y_4$	6	X:101 Y:110 I= X_1Y_2 Q= Y_1X_3	5	X :10101 Y :11010 I= $X_1Y_2Y_4$ Q= $Y_1X_3X_5$	4	X:1000101 Y:1111010 I= $X_1Y_2Y_4Y_6$ Q= $Y_1Y_3X_5X_7$	3

Table 2: Puncture patterns used by rate _ binary convolutional encoder to realize higher rates

8.3.5.5.2.1.1.1 Encoding for BPSK and QPSK Modulations, All Rates

The binary outputs of the punctured binary encoder may be sent directly to the symbol mapper for QPSK transmissions, using the format for I and Q designated at the bottom of the appropriate puncture column of Table 2. Likewise, the punctured encoder outputs ready for BPSK are also ready for constellation mapping, except the bit designated by 'I' in Table 2 is transmitted during one symbol interval, and the bit designated by 'Q' during the next interval.

8.3.5.5.2.1.1.2 Encoding for Rate _ 16-QAM

The rate _ pragmatic TCM encoder for 16-QAM, which delivers 2 source bits per 16-QAM symbol, is illustrated in Figure 3. Note that the baseline rate _ binary convolutional encoder first generates a two-bit constellation index, b_1b_0, associated with the I symbol coordinate. Provided the next encoder input, it generates a two bit constellation index, b_1b_0, for the Q symbol coordinate. The I index generation should precede the Q index generation. Note that one might want to interpret this encoder as rate 2/4 encoder, because it generates one 4 bit code symbol per two input bits. For this reason, input records that are divisible by two are required for operation of this encoder.

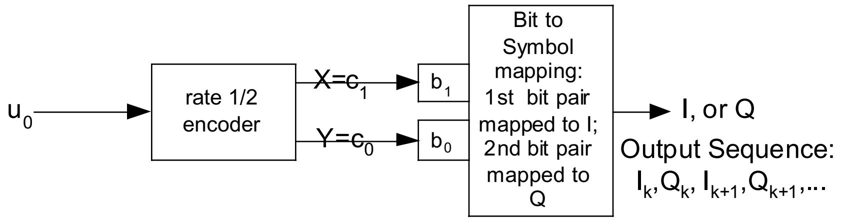
u_0
rate 1/2 encoder
X=c_1
Y=c_0
b_1
b_0
Bit to Symbol mapping: 1st bit pair mapped to I; 2nd bit pair mapped to Q
I, or Q
Output Sequence:
$I_k,Q_k, I_{k+1},Q_{k+1},...$

Figure 3: Pragmatic TCM encoder for rate _ 16-QAM

8.3.5.5.2.1.1.3 Encoding for Rate 3/416-QAM

The rate 3/4 pragmatic TCM encoder for 16-QAM, which delivers 3 source bits per 16-QAM symbol, is illustrated in Figure 4. Note that this encoder uses the baseline rate _ binary convolutional encoder, along with two systematic bits that are passed directly from the encoder input to the encoder output. With this structure, the encoder is capable of simultaneously generating 4 output bits per three input bits. The sequence of arrival for the $u_2u_1u_0$ input into the encoder is u_2 arrives first, u_1 second, u_0 last. During the encoding process, the encoder generates a two bit constellation index, b_1b_0, for the I symbol coordinate, and simultaneously generates another two bit constellation index, also designated b_1b_0 (but valued independently), for the Q symbol coordinate. Note that whole symbols must be transmitted, so the number of bits in an input record to be encoded by the rate _ 16-QAM encoder must be evenly divisible by 3.

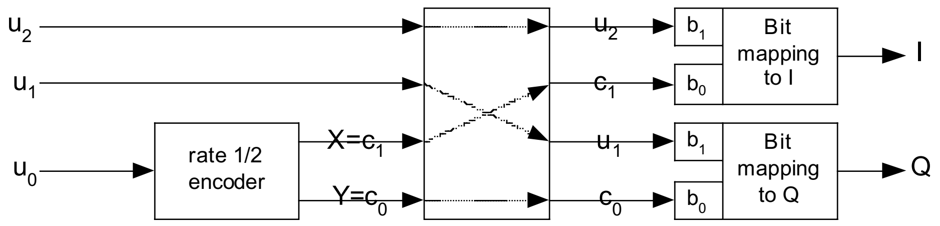

Figure 4: Pragmatic TCM encoder for rate 3/4 16-QAM

8.3.5.5.2.1.1.4 Encoding for Rate 2/3 64-QAM

The rate 2/3 pragmatic TCM encoder for 64-QAM, which delivers 4 source bits per 64-QAM symbol, is illustrated in Figure 5. Note that this encoder uses the baseline rate _ binary convolutional encoder, along with one systematic bit that is passed directly from the encoder input to the encoder output. The sequence of arrival for the u_1u_0 input into the encoder is u_1 arrives first, u_0 last. Note that the encoder (as a whole) first generates a three bit constellation index, $b_2b_1b_0$, which is associated with the I symbol coordinate. Provided another two-bit encoder input, it generates a three bit constellation index, $b_2b_1b_0$, which is associated with the Q symbol coordinate. The I index generation should precede the Q index generation. Note that one might want to interpret this encoder as a rate 4/6 encoder, because it generates one six bit code symbol per four input bits. For this reason, the number of bits in an input record to be encoded by the rate 4/6 64-QAM encoder must be evenly divisible by 4.

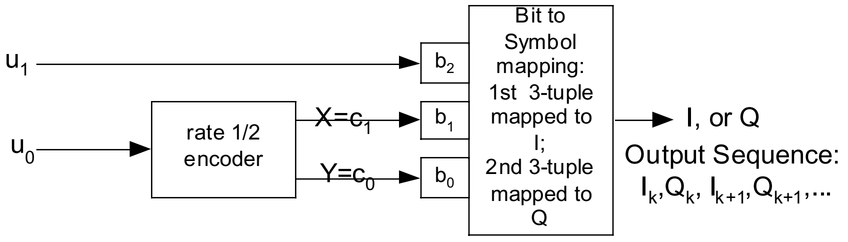

Figure 5: Pragmatic TCM encoder for rate 2/3 64-QAM

8.3.5.5.2.1.1.5 Encoding for Rate 5/6 64-QAM

The rate 5/6 pragmatic TCM encoder for 64-QAM, which delivers 5 source bits per 64-QAM symbol, is illustrated in Figure 6. Note that this encoder uses a rate _ punctured version of the rate baseline rate _ binary convolutional encoder, along with two systematic bits that are passed directly from the encoder input to the encoder output. The rate _ punctured code is generated using the appropriate puncture mask definition listed in Table 2. Puncture samples are sequenced c_3 first, c_2 second, c_1 third, and c_0 last. The sequence of arrival for the $u_4u_3u_2u_1u_0$ input into the encoder is u_4 arrives first, u_3 arrives second, u_2 arrives third, u_1 arrives next to last, and u_0 arrives last. During the encoding process, the encoder generates a three bit constellation index, $b_2b_1b_0$, for the I symbol coordinate, and simultaneously generates another three bit constellation index, also designated $b_2b_1b_0$ (but valued independently), for the Q symbol coordinate. Note that whole symbols must be transmitted, so the number of bits in an input record to be encoded by the rate 5/6 64-QAM encoder must be evenly divisible by 5.

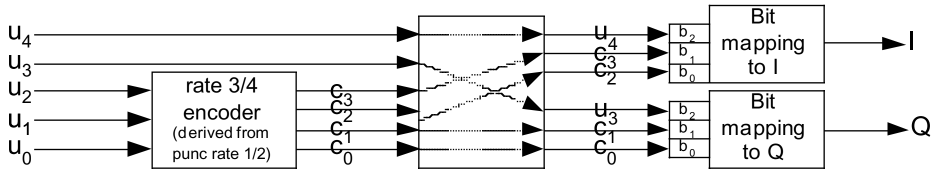

Figure 6: Pragmatic TCM encoder for rate 5/6 64-QAM

8.3.5.5.2.1.1.6 Encoding for Optional Rate 3/4 256-QAM

An optional rate 3/4 pragmatic TCM encoder for 256-QAM, which delivers 6 source bits per 256-QAM symbol, is illustrated in Figure 7. Note that this encoder uses the baseline rate _ binary convolutional encoder, along with two systematic bits that are passed directly from the encoder input to the encoder output. The sequence of arrival for the $u_2u_1u_0$ input into the encoder is u_2 arrives first, u_1 next, u_0 last. Note that the encoder (as a whole) first generates a four bit constellation index, $b_3b_2b_1b_0$, which is associated with the I symbol coordinate. Provided another four bit encoder input, it generates a four bit constellation index, $b_3b_2b_1b_0$, which is associated with the Q symbol coordinate. The I index generation should precede the Q index generation. Note that one might want to interpret this encoder as a rate 6/8 encoder, because it generates one eight bit code symbol per six input bits. For this reason, the number of bits in an input record to be encoded by the rate 6/8 256-QAM encoder must be evenly divisible by 6.

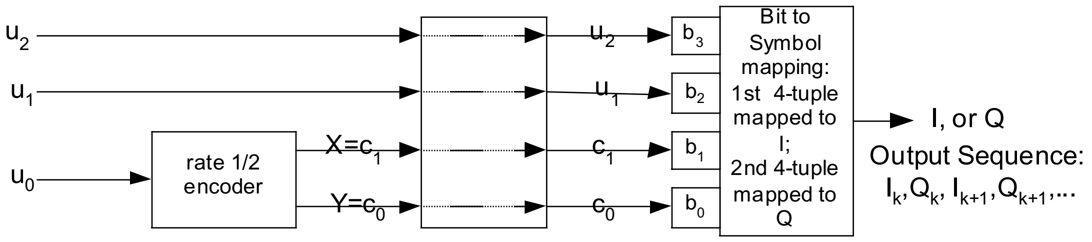

Figure 7: Optional pragmatic TCM encoder for rate _ 256-QAM

8.3.5.5.2.1.1.7 Encoding for Optional Rate 7/8 256-QAM

An optional rate 7/8 pragmatic TCM encoder for 256-QAM, which delivers 7 source bits per 256-QAM symbol, is illustrated in Figure 8. Note that this encoder uses a rate _ punctured version of the rate baseline rate _ binary convolutional encoder, along with two systematic bits that are passed directly from the encoder input to the encoder output. The rate _ punctured code is generated using the appropriate puncture mask definition listed in Table 2. Puncture samples are sequenced c_3 first, c_2 second, c_1 third, and c_0 last. The sequence of arrival for the $u_6u_5u_4u_3u_2u_1u_0$ input into the encoder (as a whole) is u_6 arrives first, u_5 arrives second, u_4 arrives third, u_3 arrives fourth, u_2 arrives fifth, u_1 arrives next to last, and u_0 arrives last. During the encoding process, the encoder generates a four bit constellation index, $b_3b_2b_1b_0$, for the I symbol coordinate, and simultaneously generates another four bit constellation index, also designated $b_3b_2b_1b_0$ (but valued independently), for the Q symbol coordinate. Note that whole 256-QAM symbols must be transmitted, so the number of bits in an input record to be encoded by the rate 7/8 256-QAM encoder must be evenly divisible by 7.

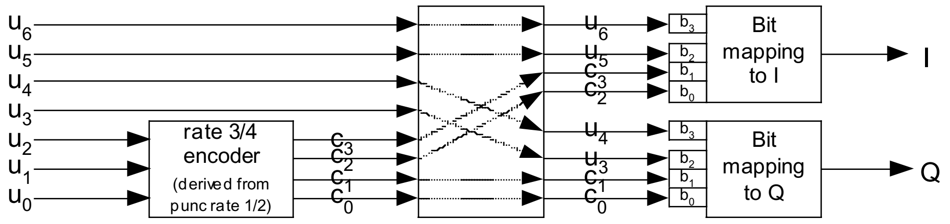

Figure 8: Optional pragmatic TCM encoder for rate 7/8 256-QAM

8.3.5.5.2.1.1.8 Inner Code Termination

Inner code blocks are to be terminated in transitions between modulation types, between allocations, at the beginning and ends of bursts, or as instructed by the 802.16 MAC. Two termination options are supported:

1) Zero-state termination
2) Tail biting

Zero-state termination shall be supported, with tail biting being an optional mode.

When using zero state termination, the basic rate _ convolutional encoder is initialized with its registers in the all-zeros state. Inner encoding begins from this state, by accepting data inputs. To zero state terminate at the end of the code block, a sufficient number of zero inputs are fed the baseline rate _ binary convolutional encoder so that its register memory is flushed, i.e., its state memory is driven to zero. This requires a minimum of 6 zero-valued inputs into the K=7 binary convolutional encoder, regardless of whether puncturing is performed or not. Once the first bit is used to begin flushing the binary convolutional encoder memory, all input bits, including the systematic input bits that are parallel to the binary convolutional encoder inputs in the pragmatic TCM encoder, should be set to zero. However, the systematic input bits set that are set to zero do not count toward the minimum 6 bit total of binary convolutional encoder bits requisite to flush the encoder memory. Note that the input bits associated with zero state termination should be accounted when computing the code rate-induced record size granularities for a pragmatic TC encoder.

When using tail biting, no extra termination bits are required. The initial state of the rate _ convolutional encoder is established so that it is the same as the final state of the encoder, at the end of a block. Since the baseline rate _ binary convolutional encoder is a shift register, this initial shift registers are loaded with the last 6 binary inputs in the block that would have been fed to the rate _ convolutional encoder. Note that these bits are not necessarily consecutive input bits, if the pragmatic TCM encoder incorporates systematic bit inputs, as well.

8.3.5.5.2.1.3 [Optional] Block Interleaver

Interleaving is an option on the downstream. If performed, the interleaving is performed on byte-sized words, before they are fed to the inner encoder. If used, this interleaving must be block-based, and must accommodate variable block sizes. The interleaver address generation algorithm is TBD. It is not necessarily a simple row-column block interleaver.

8.3.5.5.2.1.4 Symbol Mapping

For the concatenated Reed Solomon/trellis code, code bits will be mapped onto the I and Q axes using a pragmatic trellis code map. For BPSK and QPSK, the pragmatic trellis code maps are equivalent to the Gray code maps, and are illustrated in Figure 9 and Figure 10, respectively. The pragmatic trellis code maps for 16-QAM, 64-QAM, and 256-QAM are illustrated in Figure 11, Figure 12, and Figure 13, respectively. In each of these figures, a value for the normalization factor d may be chosen such that the average signal energy of the constellation is constrained to be 1.

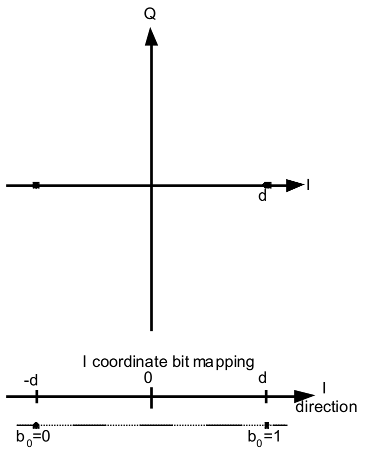

Figure 9: Pragmatic trellis code (Gray code) map for BPSK

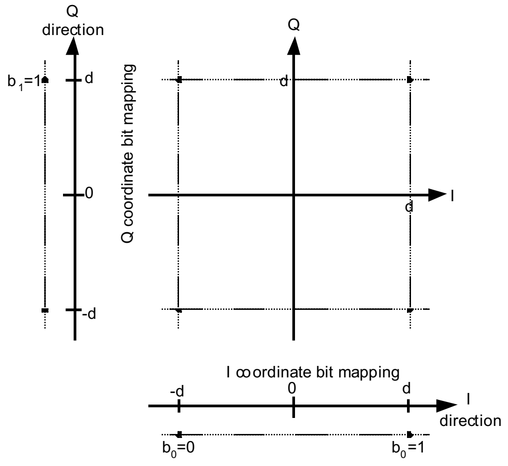

Figure 10: Pragmatic trellis code map (Gray code map) for QPSK.

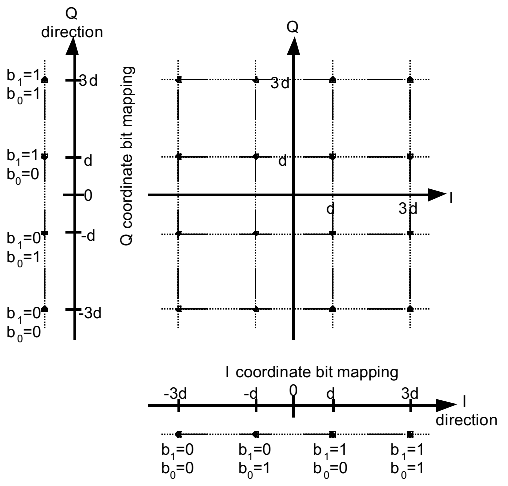

Figure 11: Pragmatic trellis code map for 16-QAM

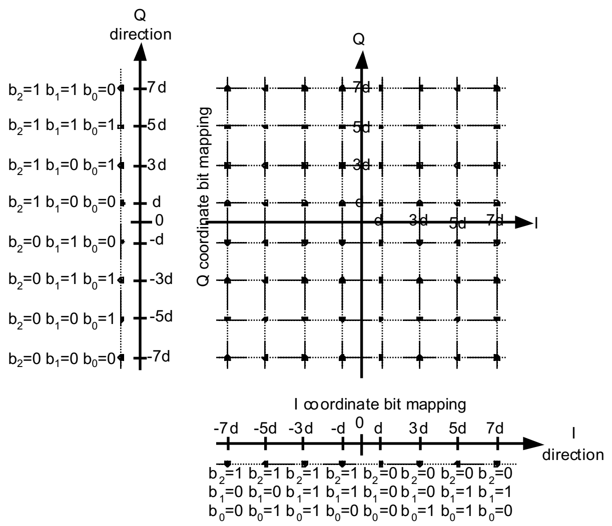
Q direction
Q
Q coordinate bit mapping
$b_2=1$ $b_1=1$ $b_0=0$ 7d
$b_2=1$ $b_1=1$ $b_0=1$ 5d
$b_2=1$ $b_1=0$ $b_0=1$ 3d
$b_2=1$ $b_1=0$ $b_0=0$ d
0
$b_2=0$ $b_1=1$ $b_0=0$ -d
$b_2=0$ $b_1=1$ $b_0=1$ -3d
$b_2=0$ $b_1=0$ $b_0=1$ -5d
$b_2=0$ $b_1=0$ $b_0=0$ -7d
I
I coordinate bit mapping
-7d -5d -3d -d 0 d 3d 5d 7d
I direction
$b_2=1$ $b_2=1$ $b_2=1$ $b_2=1$ $b_2=0$ $b_2=0$ $b_2=0$ $b_2=0$
$b_1=0$ $b_1=0$ $b_1=1$ $b_1=1$ $b_1=0$ $b_1=0$ $b_1=1$ $b_1=1$
$b_0=0$ $b_0=1$ $b_0=1$ $b_0=0$ $b_0=0$ $b_0=1$ $b_0=1$ $b_0=0$

Figure 12: Pragmatic trellis code map for 64-QAM

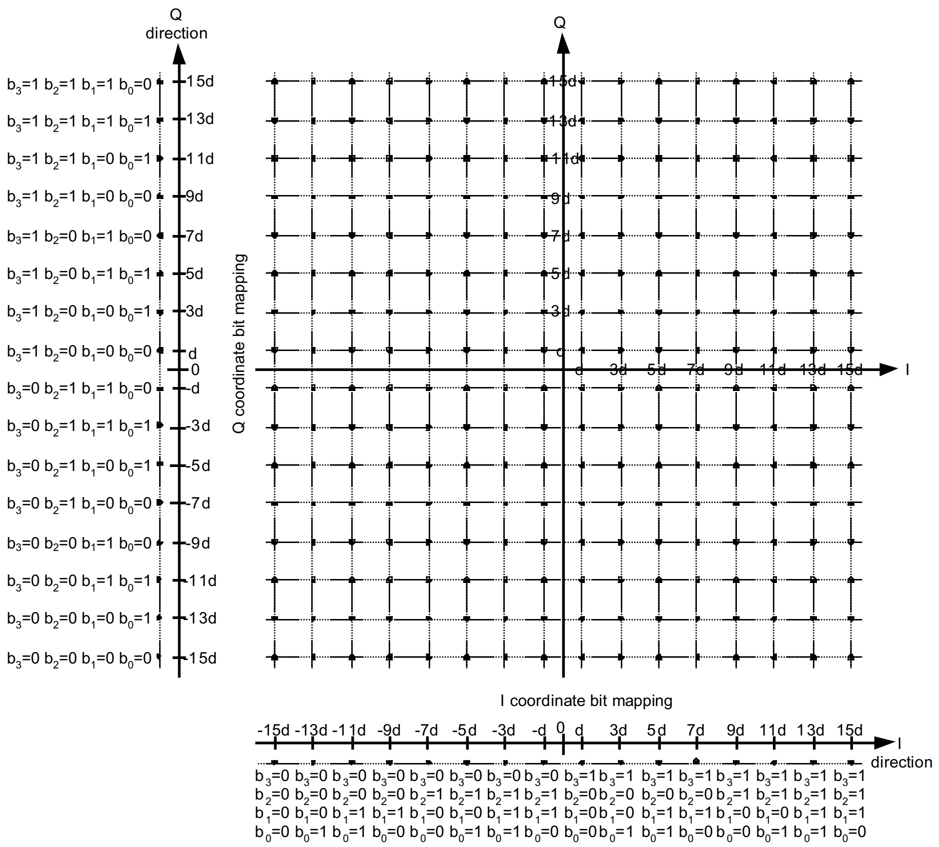
Q direction
Q
Q coordinate bit mapping
$b_3=1$ $b_2=1$ $b_1=1$ $b_0=0$ 15d
$b_3=1$ $b_2=1$ $b_1=1$ $b_0=1$ 13d
$b_3=1$ $b_2=1$ $b_1=0$ $b_0=1$ 11d
$b_3=1$ $b_2=1$ $b_1=0$ $b_0=0$ 9d
$b_3=1$ $b_2=0$ $b_1=1$ $b_0=0$ 7d
$b_3=1$ $b_2=0$ $b_1=1$ $b_0=1$ 5d
$b_3=1$ $b_2=0$ $b_1=0$ $b_0=1$ 3d
$b_3=1$ $b_2=0$ $b_1=0$ $b_0=0$ d
0
$b_3=0$ $b_2=1$ $b_1=1$ $b_0=0$ -d
$b_3=0$ $b_2=1$ $b_1=1$ $b_0=1$ -3d
$b_3=0$ $b_2=1$ $b_1=0$ $b_0=1$ -5d
$b_3=0$ $b_2=1$ $b_1=0$ $b_0=0$ -7d
$b_3=0$ $b_2=0$ $b_1=1$ $b_0=0$ -9d
$b_3=0$ $b_2=0$ $b_1=1$ $b_0=1$ -11d
$b_3=0$ $b_2=0$ $b_1=0$ $b_0=1$ -13d
$b_3=0$ $b_2=0$ $b_1=0$ $b_0=0$ -15d
I
I coordinate bit mapping
-15d -13d -11d -9d -7d -5d -3d -d 0 d 3d 5d 7d 9d 11d 13d 15d
I direction
$b_3=0$ $b_3=0$ $b_3=0$ $b_3=0$ $b_3=0$ $b_3=0$ $b_3=0$ $b_3=0$ $b_3=1$ $b_3=1$ $b_3=1$ $b_3=1$ $b_3=1$ $b_3=1$ $b_3=1$ $b_3=1$
$b_2=0$ $b_2=0$ $b_2=0$ $b_2=0$ $b_2=1$ $b_2=1$ $b_2=1$ $b_2=1$ $b_2=0$ $b_2=0$ $b_2=0$ $b_2=0$ $b_2=1$ $b_2=1$ $b_2=1$ $b_2=1$
$b_1=0$ $b_1=0$ $b_1=1$ $b_1=1$ $b_1=0$ $b_1=0$ $b_1=1$ $b_1=1$ $b_1=0$ $b_1=0$ $b_1=1$ $b_1=1$ $b_1=0$ $b_1=0$ $b_1=1$ $b_1=1$
$b_0=0$ $b_0=1$ $b_0=1$ $b_0=0$ $b_0=0$ $b_0=1$ $b_0=1$ $b_0=0$ $b_0=0$ $b_0=1$ $b_0=1$ $b_0=0$ $b_0=0$ $b_0=1$ $b_0=1$ $b_0=0$

Figure 13: Pragmatic trellis code map for 256-QAM

<u>References:</u>

[1] A. J. Viterbi, J. K. Wolf, E. Zehavi, and R. Padovani, ‘A pragmatic approach to trellis-coded modulation,’ *IEEE Communications Magazine*, July 1989, pp. 11-19.

[2] J. K. Wolf and E. Zehavi, ‘P2 codes: pragmatic trellis codes utilizing punctured convolutional codes,’ *IEEE Communications Magazine*, February 1995, pp. 94-99.