Project	**IEEE 802.16 Broadband Wireless Access Working Group <http://ieee802.org/16>**
Title	**Enhanced Error Coding Scheme**
Date Submitted	**2000-10-30**
Source(s)	Dave Williams Advanced Hardware Architectures 2365 NE Hopkins Court Pullman, WA 99163-5601 Voice: 509.334.1000 Fax: 509.334.9000 mailto:davew@aha.com Sean Sonander Advanced Hardware Architectures 2 Venture Road Chilworth Science Park, Southampton, UK Voice: +44 2380 763716 Fax: +44 2380 763714 mailto:sean@aha.com Peter Close Advanced Hardware Architectures 2 Venture Road Chilworth Science Park, Southampton, UK Voice: +44 2380 763717 Fax: +44 2380 763714 mailto:pclose@aha.com Keith Pickavance Advanced Hardware Architectures 2 Venture Road Chilworth Science Park, Southampton, UK Voice: +44 2380 763728 Fax: +44 2380 763714 mailto:keith@aha.com
Re:	Call for Contributions: Initial PHY Proposals (IEEE 802.16.3-00/14)
Abstract	An enhanced FEC system is proposed and described. The scheme has been simulated with both Single carrier and Multi-carrier (OFDM) Modulation.
Purpose	This proposal is offered as a basis for FEC in the 802.16.3 PHY layer.
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 text 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>.

Enhanced Error Coding Scheme

Peter Close, Keith Pickavance, Sean Sonander, Dave Williams
Advanced Hardware Architectures, Inc.

1. Turbo Product Codes

Introduction

The following is offered to provide additional insights into Turbo Product Codes (also known as Block Turbo Codes) as well as to provide a summary of subsequent sections of this document.

- Generic Turbo Product Codes architectures for encoder and decoders are non-proprietary and can be supported by various suppliers who may or may not choose to use proprietary decoder algorithms. The use of product codes was described in published literature in 1954 [1] and the use of iterative decoding techniques for these codes were described in published papers and at least one textbook [2] in the early 1980's.
- For high code rates (R>0.7) Turbo Product Codes are capable of outperforming other known FEC coding schemes.
- Turbo Product Codes simulations have been verified with actual hardware that has been verified by independent third parties. It has been shown that simulations match hardware performance within normal measurement accuracy.
- In addition to both software simulations and hardware verification, a union-bound based analysis for AWGN channels and QPSK modulation has been generated, which supports the simulation and hardware results [5]. This analysis can be used to predict code performance to arbitrarily low BER's. The accuracy of these predictions has been verified with actual hardware measurements for selected codes down to approximately 10^{-11}.
- Encoder complexities for Turbo Product Codes are low (in the range of 10K gates), are non-proprietary and are constructed from Hamming and/or parity codes. Memory requirements are low and in the region of 500 - 1Kbits. Latency through such an encoder is less than a few bit periods at the highest data rates.
- Decoder complexities for Turbo Product Codes are higher than for Reed-Solomon based concatenated codes. This increase is offset by the increased performance available or can be traded off against reduced complexity in other system level components such as lower power amplifier requirements, smaller antennas, higher receiver noise figures, etc. The increase in complexity is estimated to be less than 5% of the total system complexity.

Turbo Code Description

The Block Turbo Code is a Turbo decoded Product Code (TPC). The idea of this coding scheme is to use well-known product codes in a matrix form for two-dimensional coding, or in a cubical form for three dimensions.

The matrix form of the two-dimensional code is depicted in Figure 1. The k_x information bits in the rows are encoded into n_x bits, by using a binary block (n_x,k_x) code. The binary block codes employed are one error correcting BCH-codes (Bose-Chaudhuri-Hocquenghem), or Hamming Codes.

The redundancy of the code is $r_x = n_x - k_x$ and d_x is the Hamming distance. After encoding the rows, the columns are encoded using another block code (n_y,k_y), where the check bits of the first code are also encoded. The overall block size of such a product code is $n = n_x \times n_y$, the total number of information bits $k = k_x \times k_y$ and the code rate is $R = R_x \times R_y$, where $R_i = k_i/n_i$, i=x, y. The Hamming distance of the product code is $d = d_x \times d_y$.

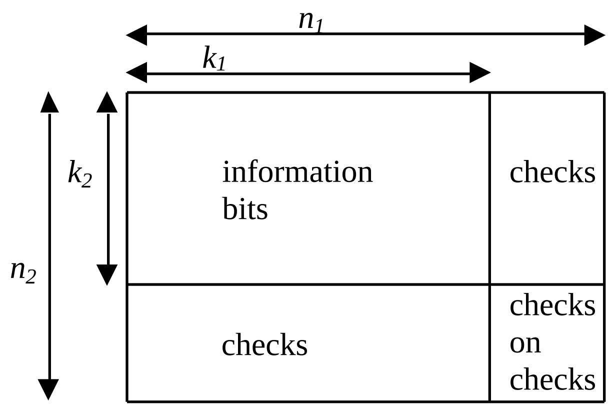


Figure 1 - Two-dimensional product code matrix

Encoding

The encoder for a TPCs has near zero latency, and is constructed of linear feedback shift registers (LFSRs), storage elements, and control logic. Encoding of a product code requires that each bit be encoded by 2 or 3 codes.

The constituent codes of TPCs are extended Hamming or parity only codes. Table 1 gives the generator polynomials of the Hamming codes used in TPCs. For extended Hamming codes an overall parity check bit is added at the end of each codeword.

N	K	Generator
7	4	$x^3 + x + 1$
15	11	$x^4 + x + 1$
31	26	$x^5 + x^2 + 1$
63	57	$x^6 + x + 1$
127	120	$x^7 + x^3 + 1$
255	247	$x^8 + x + 1$

Table 1 - Generators Polynomials of Hamming Codes

In order to encode the product code, each data bit is input both into a row encoder and a column encoder. Note that only one row encoder is necessary for the entire block, since data is input in row order. However, each column of the array must be encoded with separate encoders. Each column encoder is clocked for only one bit of the row, so a more efficient method of column encoding is to store the column encoder states in a $k_x \times (n_y-k_y)$ storage memory. A single encoder can then be used for all columns of the array. With each bit input, the appropriate column encoder state is read from the memory, clocked, and written back to the memory.

The encoding process will be demonstrated with an example. Assume a two-dimensional (8,4) × (8,4) extended Hamming Product code is to be encoded. This block has 16 data bits, and 64 total encoded bits. Figure 2 shows the original 16 data bits denoted by D_{yx}.

$$\begin{array}{cccc}
D_{11} & D_{21} & D_{31} & D_{41} \\
D_{12} & D_{22} & D_{32} & D_{42} \\
D_{13} & D_{23} & D_{33} & D_{43} \\
D_{14} & D_{24} & D_{34} & D_{44}
\end{array}$$

Figure 2 - Original Data for Encoding

The first four bits of the array are input to the row encoder in the order D_{11}, D_{21}, D_{31}, D_{41}. Each bit is also input to a unique column encoder. Again, a single column encoder may be used, with the state of each column stored in a memory. After the fourth bit is input, the first row encoder error correction coding (ECC) bits are shifted out.

This process continues for all four rows of data. At this point, 32 bits have been output from the encoder, and the four column encoders are ready to shift out the column ECC bits. This data is shifted out at the end of the row. This continues from the remaining 3 rows of the array. Figure 3 shows the final encoded block with the 48 generated ECC bits denoted by E_{yxz}.

$$\begin{array}{cccccccc}
D_{11} & D_{21} & D_{31} & D_{41} & E_{51} & E_{61} & E_{71} & E_{81} \\
D_{12} & D_{22} & D_{32} & D_{42} & E_{52} & E_{62} & E_{72} & E_{82} \\
D_{13} & D_{23} & D_{33} & D_{43} & E_{53} & E_{63} & E_{73} & E_{83} \\
D_{14} & D_{24} & D_{34} & D_{44} & E_{54} & E_{64} & E_{74} & E_{84} \\
E_{15} & E_{25} & E_{35} & E_{45} & E_{55} & E_{65} & E_{75} & E_{85} \\
E_{16} & E_{26} & E_{36} & E_{46} & E_{56} & E_{66} & E_{76} & E_{86} \\
E_{17} & E_{27} & E_{37} & E_{47} & E_{57} & E_{67} & E_{77} & E_{87} \\
E_{18} & E_{28} & E_{38} & E_{48} & E_{58} & E_{68} & E_{78} & E_{88}
\end{array}$$

Figure 3 - Encoded Block

Transmission of the block over the channel occurs in a linear fashion, with all bits of the first row transmitted left to right followed by the second row, etc. This allows for the construction of a near zero latency encoder, since the data bits can be sent immediately over the channel, with the ECC bits inserted as necessary. For the (8,4)×(8,4) example, the output order for the 64 encoded bits would be D_{11}, D_{21}, D_{31}, D_{41}, E_{51}, E_{61}, E_{71}, E_{81}, D_{12}, D_{22}, … E_{88}.

Notation:

- the codes defined for the rows (x-axis) are binary (n_x,k_x) block codes
- the codes defined for the columns (y-axis) are binary (n_y,k_y) block codes
- the codes defined for the z-dimension (z-axis) are binary (n_z,k_z) block codes
- data bits are noted $D_{y,x,z}$ and parity bits are noted $E_{y,x,z}$

Shortened TPCs

To match packet sizes, a product code can be shortened by removing symbols from the array. In the two-dimensional case rows, columns or parts thereof can be removed until the appropriate size is reached. Unlike one-dimensional codes (such as Reed-Solomon codes), parity bits are removed as part of shortening process, helping to keep the code rate high.

There are two steps in the process of shortening of product codes. The first step is to remove an entire row or column from a 2-dimensional code, or an entire X, Y, or Z plane from a 3-dimensional code. This is equivalent to shortening the constituent codes that make up the product code. This method enables a coarse granularity on shortening, and at the same time maintaining the highest code rate possible by removing both data and parity symbols. Further shortening is obtained by removing individual bits from the first row of a 2-dimensional code, or from the top plane of a 3-dimensional code.

Example of a Shortened Two-Dimensional TPC

For example, assume a 456-bit block size is required, with code rate of approximately 0.6. The base code chosen before shortening is the (32,26)×(32,26) code which has a data size of 676 bits. Shortening all rows by 5 and all columns by 4 results in a (27,21) × (28,22) code, with a data size of 462 bits. To get the exact block size, the first row of the product is shortened by an additional 6 bits. The final code is a (750,456) code, with a code rate of 0.608. Figure 4 shows the structure of the resultant block.

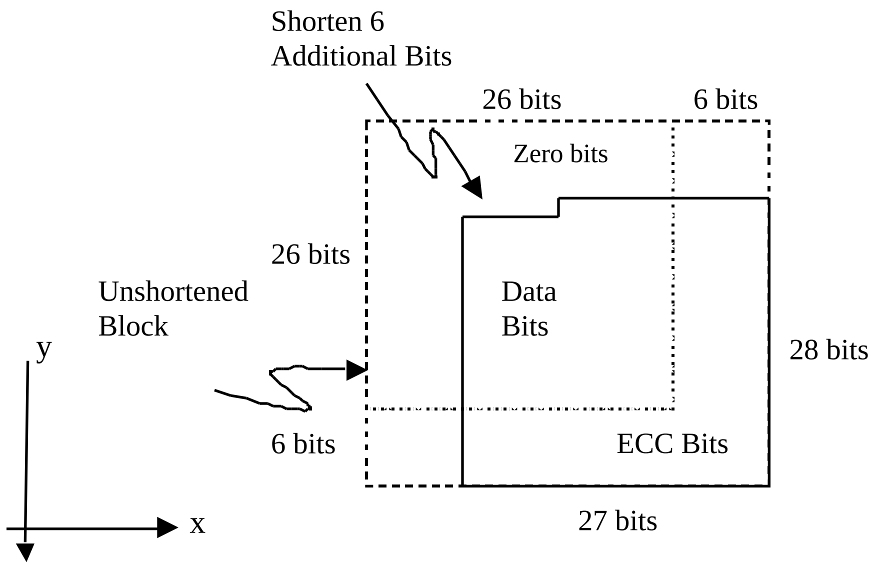


Figure 4 - Structure of Shortened 2 D Block

Modifications to the encoder to support shortening are minimal.

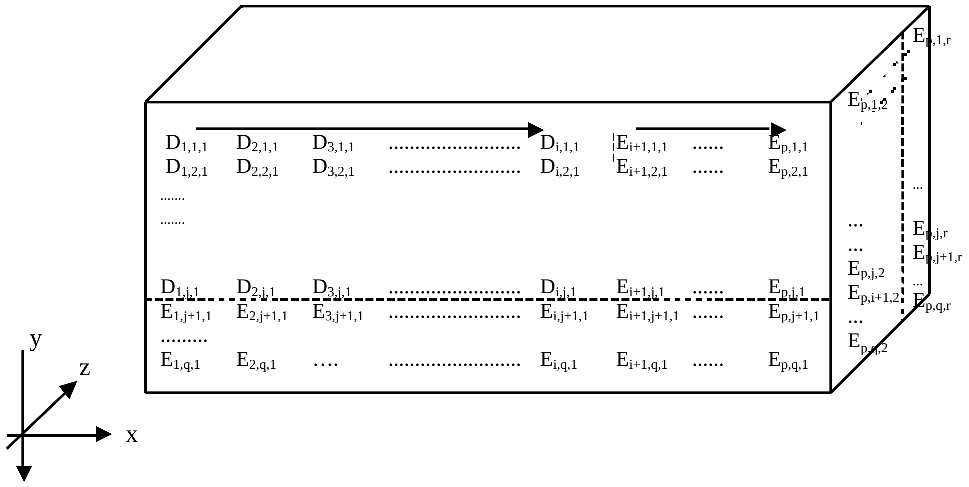
Three Dimensional TPC Encoding Example

For a three-dimensional TPC block, the element ordering for input/output for both encoding and decoding is usually in the order rows, columns then the Z-axis.

For a three-dimensional data block of size ($i \times j \times k$) and total size (($i \times j \times k$) + ECC bits), where there are p ECC bits for the x-axis, q ECC bits for the y-axis and r ECC bits for the z-axis, the bit order for input and output is:

$D_{111}, D_{211}, D_{311}, \ldots D_{j11}, \ldots E_{p11}, D_{121}, D_{221}, \ldots E_{p21}, \ldots E_{pq1}, D_{112}, D_{212}, \ldots E_{p12}, \ldots E_{pq2}, \ldots E_{pqr}$.

This is shown in Figure 5.



D1,1,1 D2,1,1 D3,1,1 Di,1,1 Ei+1,1,1 Ep,1,1
D1,2,1 D2,2,1 D3,2,1 Di,2,1 Ei+1,2,1 Ep,2,1
D1,j,1 D2,j,1 D3,j,1 Di,j,1 Ei+1,j,1 Ep,j,1
E1,j+1,1 E2,j+1,1 E3,j+1,1 Ei,j+1,1 Ei+1,j+1,1 Ep,j+1,1
E1,q,1 E2,q,1 Ei,q,1 Ei+1,q,1 Ep,q,1
Ep,1,r Ep,1,2 Ep,j,r Ep,j+1,r Ep,j,2 Ep,i+1,2 Ep,q,r Ep,q,2
x y z

Figure 5 - Structure of 3-Dimensional TPC

Suppose a 0.4 - 0.45 rate code is required with a data block size of 1096 bits. The following shows one possible method to create this code.

Start with a (32,26)×(32,26)×(4,3) code. The optimum shortening for this code is to remove rows and columns, while leaving the already very short z-axis alone. Therefore, since we desire a 1096 bit 3-Dimensional code, we can find the desired vector data size by taking the square root of 1096/3, and rounding up. This yields a row/column size of about 20. In fact, having a row size of 20, a column size of 19, and a z column size of 3 gives us the closest block size to 1096 bits.

The code size is now a (26,20)×(25,19)×(4,3) = (2600,1140). To get the exact data size, we further shorten the first plane of the code by 44 bits. This is accomplished by shortening 2 full rows from the first plane, with each row removing 20 bits from the data block, and shortening another 4 bits from the next row. This results in a (2544,1096) code, with rate = 0.43. The following diagram shows the original code, along with the physical location of the shortened bits.

Figure 6 shows the original code along with the physical location of the shortened bits.

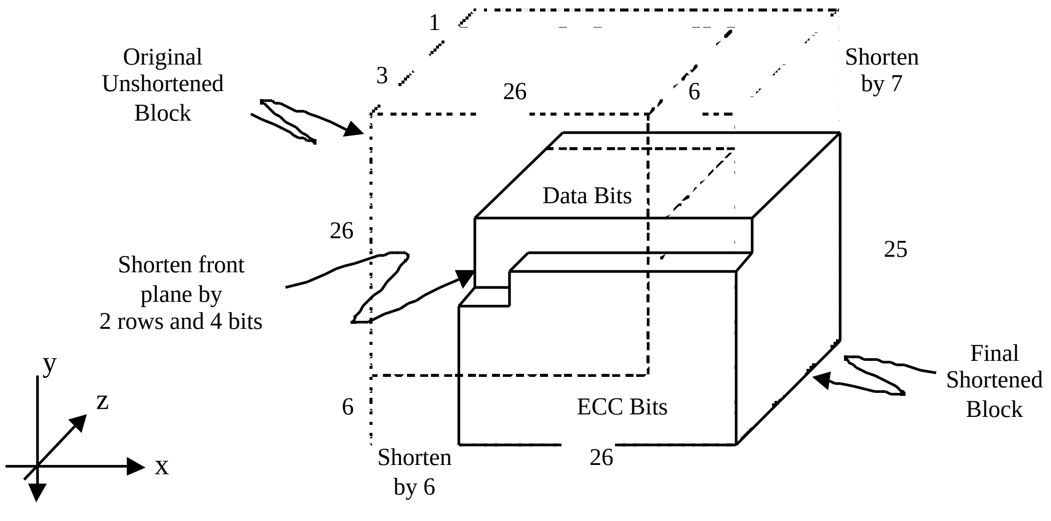


Figure 6 - Structure of Shortened 3 D Block

Fine adjustment of code shortening with 3-D codes is accomplished by shortening a given number of full rows from the top plane of the array.

Iterative Decoding

Each block code in a product code is decoded independently. First, all the horizontal blocks are decoded then all the vertical received blocks are decoded (or vice versa). The decoding procedure is generally iterated several times to maximize the decoder performance. To achieve optimal performance, the block by block decoding must be done utilizing soft information. This soft decision decoder must also output a soft decision metric corresponding to the likelihood that the decoder output bit is correct. This is required so that the next decoding will have soft input information as well. In this way, each decoding iteration builds on the previous decoding performance.

The core of the decoding process is the soft in soft out (SISO) constituent code decoder. High performance iterative decoding requires the constituent code decoders to not only determine a transmitted sequence, but to also yield a soft decision metric which is a measure of the likelihood or confidence of each bit in that sequence. Since most algebraic block decoders don't operate with soft inputs or generate soft outputs, such block decoders have been primarily realized using the Soft-Output Viterbi Algorithm (SOVA) [6] or a soft-output variant of the modified Chase algorithm(s). However, this does not limit the choice of decoding algorithms as other SISO block decoding algorithms can be used.

Block Error Detection

A further detection capability can be put in by considering Cyclic Redundancy Checks (CRC). A 32-bit linear feedback shift register with programmable taps can implement any desired CRC polynomial up to 32 bits in length. For situations where a strong block error detection mechanism is not required, inclusion of a CRC word can be turned off.

Inclusion of a CRC based block error detection scheme is both desired and highly practical from a complexity and performance standpoint. The following table lists some commonly used CRC's and their associated block error detection capability.

CRC	Size (bits)	Polynomial (hex)	Detection Capability
4	4	1f	0.9375
8	8	1d5	0.99609
12	12	180f	0.999756
ANSI	16	18005	0.999985
CCITT	16	11021	0.999985
SDLC	16	1a097	0.999985
24	24	1805101	0.9999999404
32A	32	1404098e2	0.99999999953
32B	32	104c11db7	0.99999999977

Table 2 - CRC Polynomials

Notes: The code rate loss due to the addition of an outer CRC code to a TPC is minimal. The Eb/No loss due to the addition of the CRC code can be computed with the following equation:

$$E_b/N_o \text{ loss (dB)} = 10*\log_{10}(k/(k+c))$$

where k is the data block size, and c is the CRC size. For example, the addition of a 16 bit CRC to a 188 byte (1504 bit) code results in a 0.046 dB loss. This CRC has a detection capability much higher than a t=8 Reed Solomon code.

The detection capability is the probability that an incorrect block is not marked in error. The probability of an undetected block is computed by multiplying the block error rate by (1-Detection Capability).

2. Combining TPC With Modulation Techniques

Multi-carrier Modulation

Coded Orthogonal Frequency Division Multiplexing (COFDM) is an enhanced multi-carrier modulation technique. Unlike its single carrier (SC) counterpart, COFDM uses groups of harmonically related carriers to send information across a link. Suppose that N carriers are used to transmit a signal in bandwidth B (Hz). The spacing between each carrier will be f_c, where:

$$f_c = B / N$$

In the frequency domain, such a set of un-modulated carriers would appear as shown in Figure 7 below (for the case when N is even).

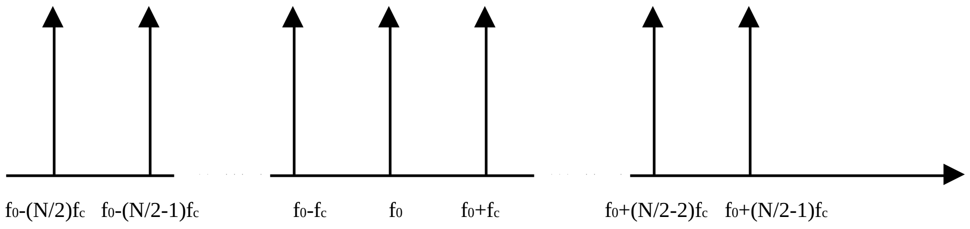


Figure 7 - A Typical COFDM Carrier Group

The time series corresponding to the frequency domain signal in Figure 7 is referred to as the OFDM symbol, and the minimum symbol duration is equal to the period of the fundamental component (at frequency f_c), i.e.

$$T_{sym} ? 1/ f_c$$

The constituent carriers are orthogonal since for any pair of harmonics, m and n, where $m ? n$

$$ {}_{0}^{1/f_c} e^{j2\pi m f_c t} ? e^{j2\pi n f_c t} dt = 0$$

A generic COFDM communication system using this multi-carrier technique is depicted in Figure 8.

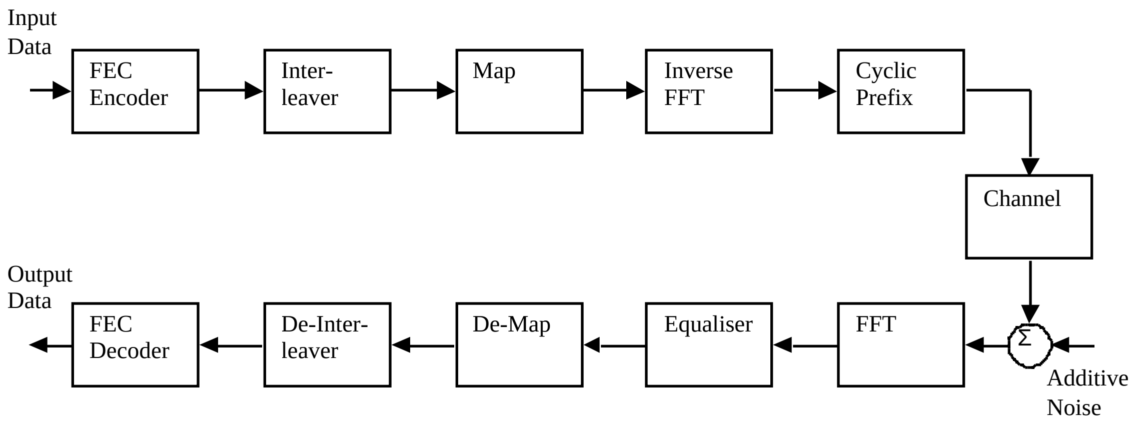


Figure 8 A Generic COFDM Communication System

Modulation

A modulated COFDM signal is most efficiently generated using an inverse Fast Fourier Transform (inverse FFT) of size N_{fft}, where $N_{fft} \geq N$. In the frequency domain, each carrier is independently modulated with data from the source to generate a complex vector of length N_{fft}. This process is referred to as 'mapping', and standard modulation schemes such as BPSK, QPSK, 16QAM or 64QAM may be used. Any pilot structure required for receiver synchronization and equalization is inserted at this point. The resultant complex vector is transformed into the time domain using the inverse FFT algorithm, yielding a complex (I/Q) time series, also of length N_{fft}.

If the channel is expected to introduce any time domain dispersion, then a cyclic prefix (or guard interval) is required. This prefix allows the receiver to maintain orthogonal detection of all constituent carriers in the presence of multiple signal paths. To make the prefix, a portion of the time domain series from the inverse FFT is either appended or pre-pended to the basic N_{fft} points. The length of the cyclic prefix is a system design parameter, and should be chosen to be just longer than the greatest amount of dispersion expected during system use. A cyclic prefix of excessive length results in wasted channel capacity. A cyclic prefix of inadequate length results in unavoidable inter-symbol interference (ISI) at the receiver.

Demodulation

The first step of demodulation requires the separation of the N orthogonal carriers. Taking an FFT of the received signal yields the original frequency domain components, corrupted by a combination of effects:

- Carrier frequency offset, which results in the rotation of the constellation of each carrier from symbol to symbol. Each carrier is affected in the same way, unless the offset arises from Doppler shift in the channel.
- Carrier phase noise, which results in inter-carrier interference (ICI). The demodulator can track 'low' frequency phase noise, but the ICI caused by high frequency noise is irreversible.
- Symbol timing error, i.e. mismatch between transmit and receive clock frequencies. In the short term, this leads to interference becoming progressively more severe with increasing distance from the 'DC' carrier.
- Symbol timing offset, which introduces a linear phase shift with gradient $-2\pi/N_{fft}$ for each unit delay offset of the FFT window from the reference position.
- Fading, which introduces frequency dependent attenuation and phase shift.
- Additive noise.

Carrier frequency offset, low frequency phase noise and symbol timing error can all be tracked out by the appropriate closed loop feedback systems, leaving the effects of the frequency selective channel to be undone by the equalizer. Unlike single carrier systems, which generally require a multi-tap feed-forward equalizer, decision feedback equalizer, or even both, the OFDM equalization process requires a single complex multiplication per carrier to re-align each received point with the appropriate decision axes.
The de-map process takes each equalized point, and determines the hard decision data and associated probability or likelihood. For optimum results, likelihood information is computed from both current and historical data.

Forward Error Correction and Interleaving

As in single carrier systems, forward error correction (FEC) can be used to provide coding gain, and subsequent improved performance. There are two areas requiring attention when trying to obtain optimum performance:

- Interleaving, which serves two purposes. Firstly, consecutive output bits from the FEC encoder are dispersed among non-adjacent carriers. This mitigates against the effects of a fading channel, and against narrow-band co-channel interference affecting small clusters of carriers. Secondly, when using Gray coded higher order constellations such as 16QAM, consecutive output bits from the FEC encoder are dispersed so that long runs of low confidence decisions at the input to the FEC decoder are made less likely.

- Channel State Identification, (CSI) is especially important in systems where there is a fading channel. Conventional FEC decoders using soft-decision inputs receive log-likelihood ratio (LLR) estimates based upon the Euclidean distance of a received point from its nearest neighbor in the constellation. In multi-carrier systems, it is possible to form an estimate of the medium-term reliability of each independent carrier. Thus it is possible for the de-map function to assign soft-decision information on a carrier-by-carrier basis, taking account of both instantaneous de-mapping and medium term reliability.

Combined with the appropriate interleaving scheme, CSI facilitates optimum use of the soft-decision FEC decoder.

Turbo Product Codes and Multicarrier Modulation

The purpose of this section is to demonstrate the performance of Turbo Product Codes (TPC's) when used as forward error correction with multicarrier modulation. Multicarrier modulation techniques have gained widespread support in recent times as a method of giving good protection against errors in multipath environments. A generic OFDM system was used as a basis to demonstrate the performance of TPC's combined with Multicarrier modulation.

Physical Layer Simulation

The simulated physical layer offers a number of data transmission rates, ranging from 6 Mbps to 54 Mbps. The simulation allows a choice of subcarrier modulation schemes ranges from QPSK to 64QAM. The chosen bit rate dictates modem parameters, according to Table 3.

Bit Rate (Mbps)	Subcarrier Modulation	Coding Rate	Data Bits per OFDM Symbol
6	BPSK	_	24
9	BPSK	_	36
12	QPSK	_	48
18	QPSK	_	72
24	16 QAM	_	96
36	16 QAM	_	144
48	64 QAM	2/3	192
54	64 QAM	_	216

Table 3- Data Rate/Modem Parameters

The OFDM symbol used in the simulation contained 52 subcarriers, formed from 48 data subcarriers and 4 pilot tones. Encoded data is mapped to the OFDM subcarriers in a gray code sense. Guard frequencies and times protect the OFDM packet against ISI and multipath.

Integrating Turbo Product Codes with OFDM Symbols

In an OFDM symbol, the number of coded bits that may fit into one symbol is generally constant from symbol to symbol. Therefore, Turbo Product Codes can be easily matched to fit into an integer number of OFDM symbols. A TPC may be described in part by its block length and the corresponding number of information symbols. Optimizing the TPC to an OFDM symbol will offer good packet filling.
In this proposal we present simulation results that show the performance of TPC's in an OFDM environment. The selection of TPC's has been undertaken with several factors taken into consideration, such as TPC code rate, TPC block length and TPC code performance.
The simulation included synchronization functions and equalization based on pilot tones. Switches select the inclusion of either convolutional codes or the corresponding TPC. A comparator block compares the received, demodulated and decoded block against the original data. The simulation setup as described is shown in Figure 9.

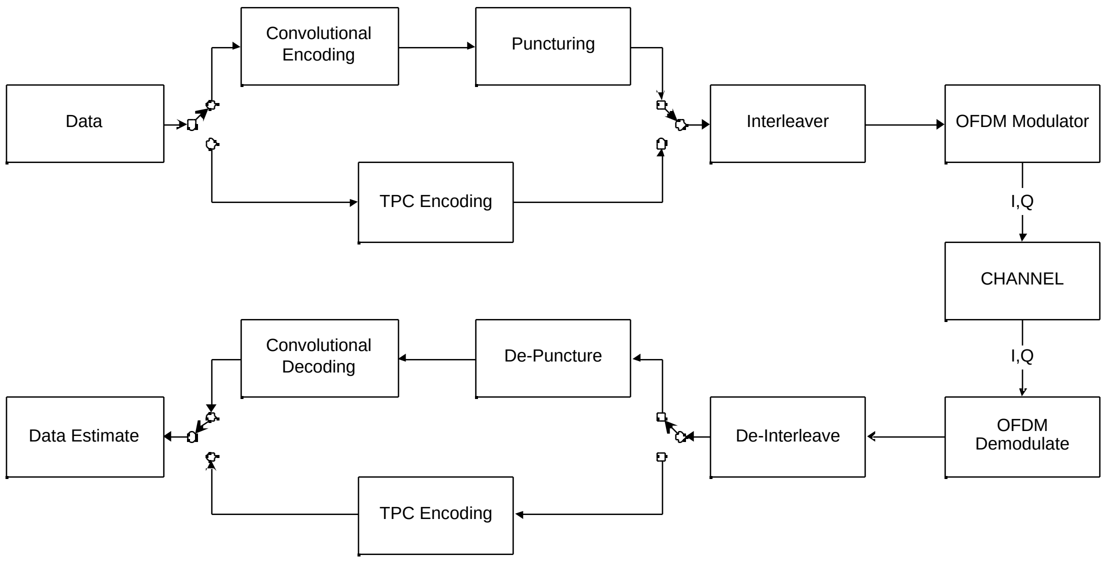


Figure 9 - Modem Simulation Overview

Simulation Results 1

The OFDM performance was simulated in a Guassian channel. The codes used were as chosen for the single carrier simulation. As expected the results were the same as those achieved by the single carrier system (see Figure 11). Results in Rayleigh and Ricean channels will be included in a later submission.

2.2. Single Carrier Modulation

QAM is a single carrier modulation method based on varying both the phase and frequency of the carrier. This scheme allows for a variety of bits per symbol, resulting in a more efficient use of bandwidth. The cost of this additional bandwidth is worse noise performance and TPC encoder/decoder can be used in these systems to regain much of this loss, giving an overall system with improved performance and error rates. Gray coding is used to map the QAM symbols to constellation points. Gray coding maps the bits of each symbol such that if a symbol error causes a transition to an adjacent constellation point, only one hard decision bit error occurs. Since the most common errors at the demodulator will be in adjacent symbols, this reduces the overall bit error rate of the system. The block diagram of the QAM system is shown in Figure 10.

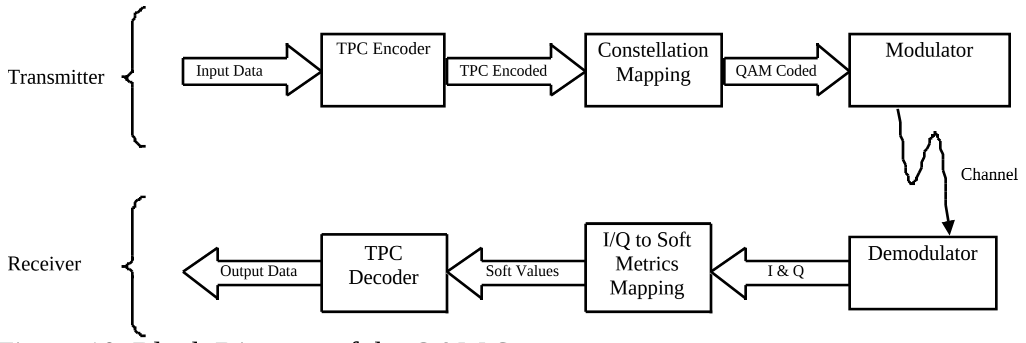
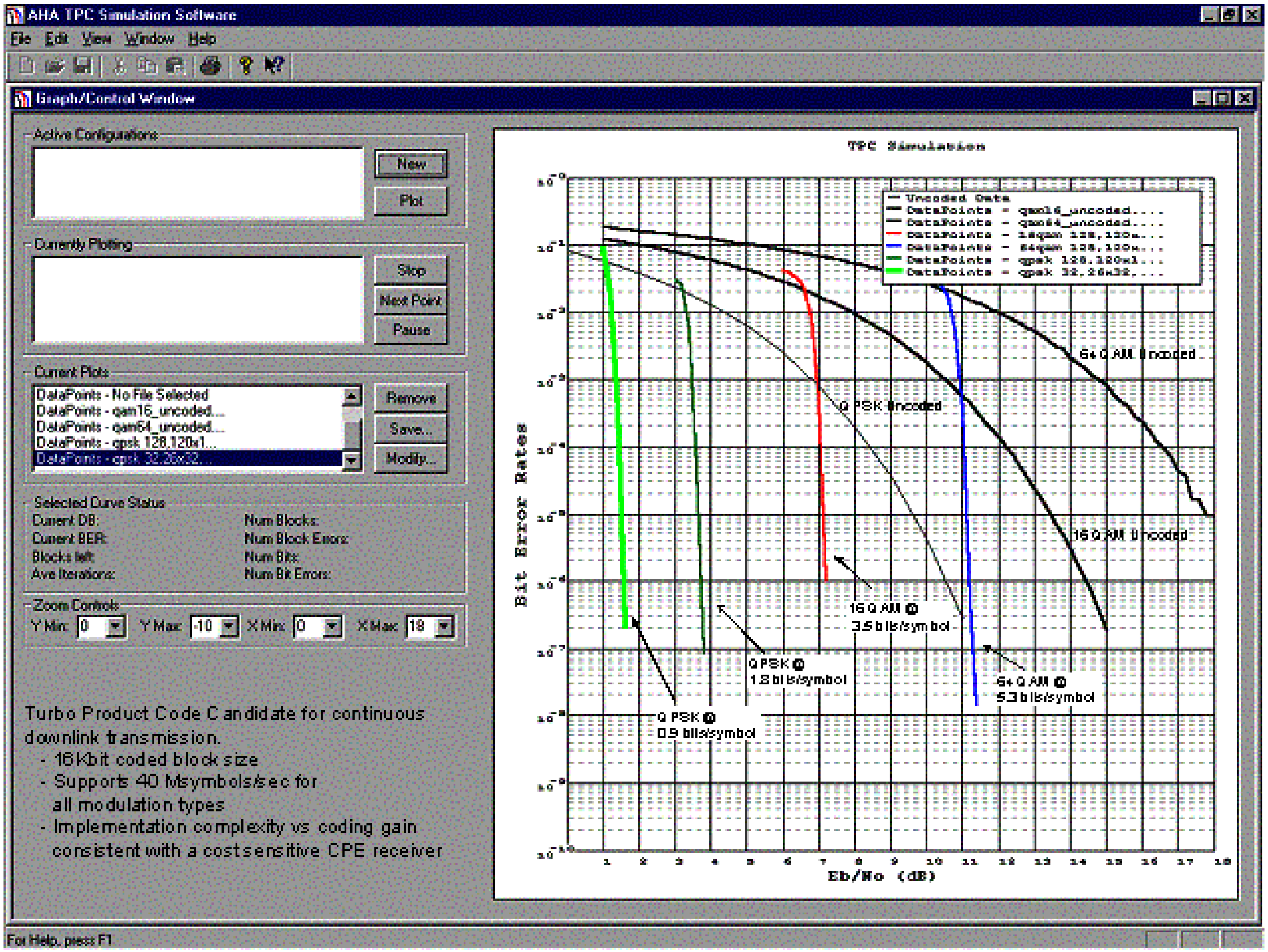


Figure 10. Block Diagram of the QAM System

The above system has been simulated for number of different channel model. Figure 11 shows the performance of TPC combined with different QAM techniques in additive white Gaussian channel and Figure 12 demonstrates the performance of TPC combined with the QAM in different channel models.



DataPoints - No File Selected
DataPoints - qam16_uncoded...
DataPoints - qam64_uncoded...
DataPoints - qpsk 128,120x1...
DataPoints - qpsk 32,26x32...

Current DB:
Current BER:
Blocks left:
Ave Iterations:
Num Blocks:
Num Block Errors:
Num Bits:
Num Bit Errors:

Y Min: 0 Y Max: -10 X Min: 0 X Max: 18

—	Uncoded Data
—	DataPoints - qam16_uncoded....
—	DataPoints - qam64_uncoded....
🟥	DataPoints - 16qam 128,120x....
🟦	DataPoints - 64qam 128,120x....
🟩	DataPoints - qpsk 128,120x1....
🟩	DataPoints - qpsk 32,26x32,....

Turbo Product Code Candidate for continuous downlink transmission.
- 16Kbit coded block size
- Supports 40 Msymbols/sec for all modulation types
- Implementation complexity vs coding gain consistent with a cost sensitive CPE receiver

Figure 11. Performance of TPC and QAM in AWGN Channel

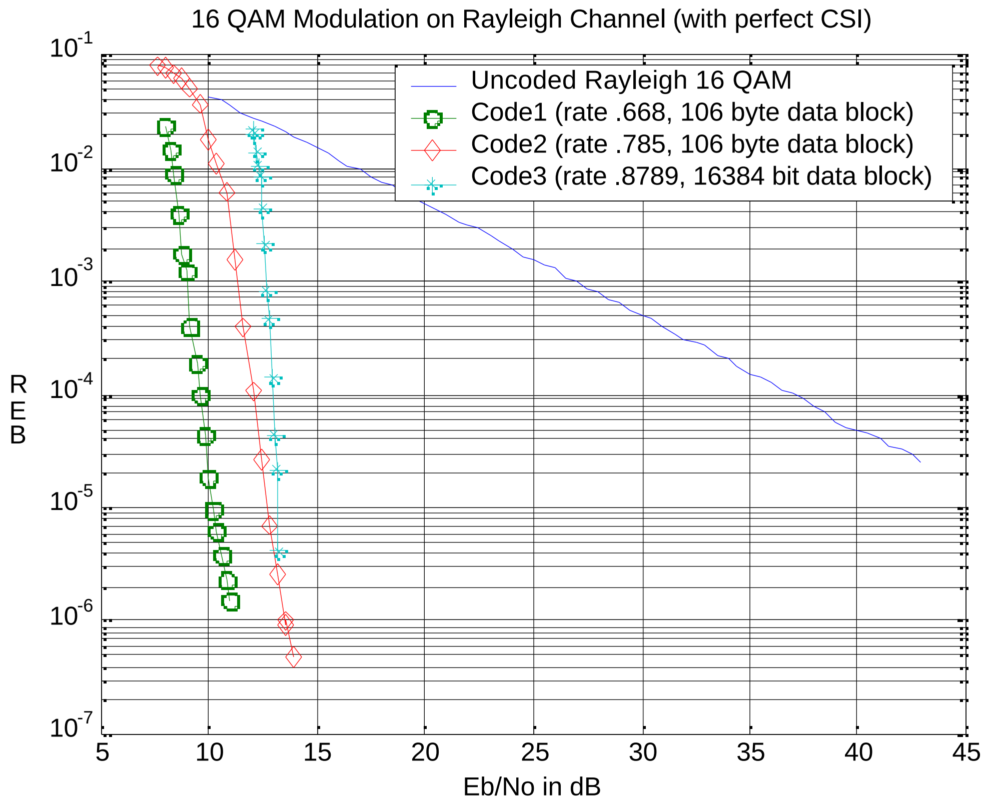


Figure 12. Performance of TPC and 16QAM in Rayleigh Channel

References

[1] P. Elias, "Error-Free Coding, " IRE Trans. Inf. Theory, PGIT-4, pp.29-37, September 1954

[2] Error Control Coding: Fundamentals and Applications, S. Lin and D. Costello, Prentice-Hall, 1983, pp. 274-277.

[3] Press Release, "AHA announces Turbo Product Code Forward Error Correction Technology", Nov. 2, 1998

[4] Press Release, "AHA announces Turbo Product Code Core Generator", September 20 1999

[5] Hewitt, E, "Turbo Product Codes for LMDS," IEEE Radio and Wireless Conference, August 1998

[6] A Viterbi Algorithm with Soft-Decision Outputs and its Applications, J. Hagenauer, P. Hocher, *IEEE Globecom '89*, Nov. 1989, pp. 1680-1685.

[7] 802.16.1 Phy working draft