Coexistence of Ultra-Wideband System and IEEE 802.11a WLAN

by

Amir Soltanian
Wireless Communication Technologies Group
National Institute of Standards and Technology
Gaithersburg, Maryland

April 2003

Revised October 2003
L. E. Miller

1. Introduction

This report describes a software tool that can be used to evaluate the impact of ultra-wideband (UWB) radio on the 802.11a WLAN. The simulator is written in C code and is developed and debugged in Microsoft visual C++ (version 6.00), environment. We first explain the simulation model for the 802.11a and UWB systems and then we describe the parameters involved in the simulation. We use a number of examples to clarify the details of the simulator. These examples can also be used as probing points for debugging. An analytical approach is also presented as a baseline for comparison. Finally, example simulation results are presented at the end of this report.

2. System model

Figure 1 shows the baseband model of the 802.11a system [1]. In general the model consists of a transmitter, an AWGN channel, an UWB interference and a receiver. Each part consists of different blocks that will be discussed in the following sections. The lightly shaded blocks are not implemented in the simulator; however their realization is fairly straightforward.

Table 1 presents the parameters of the physical layer of 802.11a. The OFDM symbol duration is 4 μs with a guard interval equal to 0.8 μs. There are a 48 data and 4 pilot subcarriers in one OFDM symbol. Hence, uncoded data rates from 12 to 72 Mbps can be achieved by using variable modulation types from BPSK to 64-QAM. To correct subcarriers in deep fades, forward-error correction across the subcarriers is used with variable coding rates, giving coded data rates from 6 up to 54 Mbps.

Data rate	6, 9, 12, 18, 24, 36, 48, 54 Mbps
Modulation	BPSK, QPSK, 16-QAM, 64-QAM
Coding Rate	½, 2/3, ¾
Number of subcarriers	52
OFDM Symbol Duration	4 μs
Guard Interval	800 ns
Subcarrier Spacing	312.5 kHz
-3 dB Bandwidth	15.56 MHz
Channel Spacing	20 MHz
Sampling Rate (FFT = 64)	50 ns
Sampling Rate (FFT = 8192)	.39 ns

Table 1- OFDM parameters.

3. The transmitter

The transmitter consists of a packet generator, a convolutional coder, a block interleaver, a baseband modulator, a pilot insertion block, an inverse fast Fourier transform (IFFT) block, guard interval extension, and windowing. We will describe these blocks in the upcoming sections.

3.1. Packet generator

The input data is formatted to different packet lengths according to the modulation type and coding rate. Table 2 describes different data rates and the corresponding packet lengths that can be achieved with this system. As we can see there is a minimum number of bits per packet for each data rate. In the simulation we select a packet length that is a multiple (twice) of this minimum length requirement.

Random binary data is generated for each packet using a pseudorandom noise generator based on the maximal length sequence characterized by the polynomial $1 + x^3 + x^{20}$ [7].

3.2. Convolutional coder and interleaver

The binary data is encoded by a standard rate ½ convolutional coder. The rate may be increased to 2/3 or ¾ by puncturing the coded output bits. The convolutional encoder is shown in Figure 2. It uses the industry standard generator polynomials, $g_0 = 133_8$ and $g_1 = 171_8$, of rate R=1/2. The bit denoted as "A" is output from the encoder before the bit denoted as "B." Higher rates are derived from it by employing puncturing as depicted in Figure 3. Puncturing is a procedure for omitting some of the encoded bits in the transmitter and inserting dummy "zero" metric into the convolutional decoder on the receive side in place of the omitted bits.

The output of the convolutional coder is interleaved by a block interleaver. In this model we did NOT include the interleaver in the simulator, however for further development an interleaver can easily be added to the model.

Data Rate	Modulation	Coding Rate	Coded bits per OFDM symbol	Min. data bits per OFDM symbol	Packet length in the simulation	OFDM symbol per packet
6 Mbps	BPSK	½	48	24	48 bits	2
9 Mbps	BPSK	¾	48	36	72 bits	2
12 Mbps	QPSK	½	96	48	96 bits	2
18 Mbps	QPSK	¾	96	72	144 bits	2
24 Mbps	16-QAM	½	192	96	192 bits	2
36 Mbps	16-QAM	¾	192	144	288 bits	2
48 Mbps	64-QAM	2/3	288	192	384 bits	2
54 Mbps	64-QAM	¾	288	216	432 bits	2

Table 2- Packet length in the simulator.

3.3. Baseband modulator

The encoded binary serial data is divided into groups of (1, 2, 4, or 6) bits and converted into complex numbers representing BPSK, QPSK, 16-QAM, or 64-QAM constellation points as shown in Table 3 to 6. The output values are normalized by a normalization factor, K_{NORM} that depends on the modulation, listed in Table 7. This normalization is to achieve the same average power (I^2+Q^2) for all mappings.

Input bit (b0)	Inphase	Quadrature
0	-1	0
1	1	0

Table 3- BPSK modulation IQ mapping.

Input bit (b0)	Inphase	Input bit (b1)	Quadrature
0	-1	0	-1
1	1	1	1

Table 4- QPSK modulation IQ mapping.

Input bits (b0 b1)	Inphase	Input bits (b2 b3)	Quadrature
00	-3	00	-3
01	-1	01	-1
11	1	11	1
10	3	10	3

Table 5- 16-QAM modulation IQ mapping.

Input bits (b0 b1 b2)	Inphase	Input bits (b3 b4 b5)	Quadrature
000	-7	000	-7
001	-5	001	-5
011	-3	011	-3
010	-1	010	1
110	1	110	1
111	3	111	3
101	5	101	5
100	7	100	7

Table 6- 64-QAM modulation IQ mapping.

Modulation	K_{NORM}
BPSK	1
QPSK	$1/\sqrt{2}$
16-QAM	$1/\sqrt{10}$
64-QAM	$1/\sqrt{42}$

Table 7- Normalization factor.

3.4. Pilot insertion

In each OFDM symbol, four of the subcarriers are dedicated to pilot signals in order to make coherent detection robust against frequency offsets and phase noise. If we number the 52 subcarriers (excluding zero subcarrier) from -26 to 26, these pilot signals shall be put in subcarriers –21, -7, 7 and 21. The pilots shall be BPSK modulated by a pseudo binary sequence to prevent the generation of spectral lines.

The stream of complex numbers representing IQ-mapped data is divided into groups of N_{SD}=48 complex numbers. We shall denote by writing complex number $d_{k,n}$, which corresponds to subcarrier k of OFDM symbol n, as follow:

$$d_{k,n} = d_{k+N_{SD}\times n}, \qquad k = 0,\dots N_{SD} - 1, \qquad n = 0,\dots N_{SYM} - 1$$

where N_{SYM} represents the number of OFDM symbols.

An OFDM symbol, $r_{DATA,n}(t)$, is defined as

$$r_{DATA,n}(t) = w_{TSYM}(t)\left(\sum_{k=0}^{N_{SD}-1} d_{k,n} e^{j2\pi M(k)\Delta_F (t-T_{GI})} + p_{n+1} \sum_{k=-N_{ST}/2}^{N_{ST}/2} P_k e^{j2\pi\Delta_F k(t-T_{GI})} \right)$$

where N_{ST} = 52 is the total number of subcarriers, Δ_F = 312.5 kHz is the subcarrier frequency spacing, T_{GI} =.8 μs is the guard interval duration, T_{SYM} = 4 μs is the OFDM symbol interval, $w_{TSYM}(t)$ is a window function that is not considered here, and $M(k)$ defines a mapping from the logical subcarrier number 0 to 47 into frequency offset index –26 to 26, while skipping the pilot subcarrier locations and the 0th (dc) subcarrier.

$$M(k) = \begin{cases} k-26 & 0 \le k \le 4 \\ k-25 & 5 \le k \le 17 \\ k-24 & 18 \le k \le 23 \\ k-23 & 24 \le k \le 29 \\ k-22 & 30 \le k \le 42 \\ k-21 & 43 \le k \le 47 \end{cases}$$

The contribution of the pilot subcarriers for the n^{th} OFDM symbol is produced by Fourier transform sequence P, given by

$$P_{-26,26} = \{0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,-1,0,0,0,0,0\}$$

The polarity of the pilot subcarriers is controlled by the sequence, p_n, which is a cyclic extension of the 127-element sequence that can be found in [1], page 23.

3.5. Inverse fast Fourier transform

The data and pilot values are modulated onto 52 subcarriers by applying the inverse fast Fourier transform (IFFT). The Discrete Fourier Transform (DFT) of a signal can be written as (ref. 2)

$$X[k] = \sum_{n=0}^{N-1} W^{nk} x[n], \qquad k = 0,1,\ldots,N-1$$

where

$$W_N \equiv e^{-j(2\pi/N)}$$

It can be shown that a DFT of length N can be rewritten as the sum of two discrete transforms, each of length N/2. One of the two is formed from the even-numbered points of the original N, the other from the odd-numbered points. The proof is simply this:

$$X[k] = \sum_{n \ \ even} x[n] W_N^{kn} + \sum_{n \ \ odd} x[n] W_N^{kn},$$

$$X[k] = \sum_{r=0}^{(N/2)-1} x[2r] W_N^{2rk} + \sum_{r=0}^{(N/2)-1} x[2r+1] W_N^{(2r+1)k}$$

$$X[k] = \sum_{r=0}^{(N/2)-1} x[2r] W_{N/2}^{rk} + W_N^K \sum_{r=0}^{(N/2)-1} x[2r+1] W_{N/2}^{rk}$$

$$= G[k] + W_N^k H[k].$$

Each of the sums in the above equation is recognized as an (N/2)-point DFT. If N is even we can consider computing each of the (N/2)-point DFTs by breaking each of the sum into two (N/2)-point DFTs. When N is a power of two this process can go on further till we have Fourier transforms of length 1. This last operation is an identity operation that copies its one input number to its one output slot.

The next consideration for using FFT is to sort out the data in the bit-reverse order. If we look at the process of successive subdivision of the data into even and odd segments, we see that these are tests of successive low-order (least significant) bits of n. If we take the original vector x[n] and rearrange it in bit-reverse order the bookkeeping of the algorithm becomes very simple. This rearranging makes the output to appear in the normal order.

The algorithm itself is taken from ref.3 and it basically consists of two parts. The first part rearranges the data in bit-reversed order and the next part takes the IFFT using the above recursive method. This algorithm is called Radix-2 FFT with decimation in time, since it successively decomposes the input time samples to even and odd segments.

3.6. Guard interval extension

To make the OFDM system robust to multipath propagation, a cyclic prefix is added to the OFDM symbol as illustrated in Figure 4. In this figure each subcarrier is depicted separately, however, in reality we send the sum of all these signals. The original OFDM signal consists of 64 complex samples (52 subcarriers and 12 zeros) for a 64-point FFT in a 3.2 μs interval. This waveform is extended in the guard interval (0.8 μs) in the beginning of the OFDM symbol. This is equal to appending 16 complex samples to the already existing 64 OFDM samples. We also add an additional guard sample at the end of the OFDM symbol. Thus, the total amount of samples per OFDM symbol is 81. The OFDM symbols are windowed and overlapped by one sample, so in the 802.11a system there are 80 samples per OFDM symbol interval. In our simulator we ignore windowing but not the appending, so each OFDM symbol consists of 81 samples.

3.7. A sample output of the transmitter

Table 8 through 11 shows the output of the 802.11a transmitter at different stages. In Table 8 a random packet of data is generated by the packet generator block. 48 bits are produced as a result of this function. These bits are located in packet[] and ppacket[] arrays. In ppacket[] the data values are bit-mapped where LSB is the first bit to be transmitted. The content of this array is

ppacket[0]= 0B29 hex
ppacket[1]= C128 hex
ppacket[2]= 2924 hex

#	Bits	#	Bits	#	Bits	#	Bits	#	Bits	#	Bits
0	1	8	1	16	0	24	1	32	0	40	1
1	0	9	1	17	0	25	0	33	0	41	0
2	0	10	0	18	0	26	0	34	1	42	0
3	1	11	1	19	1	27	0	35	0	43	1
4	0	12	0	20	0	28	0	36	0	44	0
5	1	13	0	21	1	29	0	37	1	45	1
6	0	14	0	22	0	30	1	38	0	46	0
7	0	15	0	23	0	31	1	39	0	47	0

Table 8- Data bits.

We consider uncoded BPSK with a 64-point FFT in order to be able to present the results. Table 9 shows the frequency domain representation of the signal. In this table the data content in Table 8 is BPSK modulated to yield frequency domain representation of the signal. These numbers are located in Inph[] and Quad[] arrays. Locations –21, -7, 7 and 21 are skipped and will be used for pilot insertion.

#	Re	Im	#	Re	Im	#	Re	Im	#	Re	Im
-32	0.0	0.0	-16	1.0	0.0	0	0.0	0.0	16	-1.0	0.0
-31	0.0	0.0	-15	-1.0	0.0	1	1.0	0.0	17	-1.0	0.0
-30	0.0	0.0	-14	1.0	0.0	2	-1.0	0.0	18	1.0	0.0
-29	0.0	0.0	-13	-1.0	0.0	3	-1.0	0.0	19	-1.0	0.0
-28	0.0	0.0	-12	-1.0	0.0	4	-1.0	0.0	20	-1.0	0.0
-27	0.0	0.0	-11	-1.0	0.0	5	-1.0	0.0	21	X	X
-26	1.0	0.0	-10	-1.0	0.0	6	-1.0	0.0	22	1.0	0.0
-25	-1.0	0.0	-9	-1.0	0.0	7	X	X	23	-1.0	0.0
-24	-1.0	0.0	-8	-1.0	0.0	8	1.0	0.0	24	1.0	0.0
-23	1.0	0.0	-7	X	X	9	1.0	0.0	25	-1.0	0.0
-22	-1.0	0.0	-6	-1.0	0.0	10	-1.0	0.0	26	-1.0	0.0
-21	X	X	-5	1.0	0.0	11	-1.0	0.0	27	0.0	0.0
-20	1.0	0.0	-4	-1.0	0.0	12	1.0	0.0	28	0.0	0.0
-19	-1.0	0.0	-3	1.0	0.0	13	-1.0	0.0	29	0.0	0.0
-18	-1.0	0.0	-2	-1.0	0.0	14	-1.0	0.0	30	0.0	0.0
-17	1.0	0.0	-1	-1.0	0.0	15	1.0	0.0	31	0.0	0.0

Table 9- Frequency domain representation.

Four pilot subcarriers are added by taking values {1.0, 1.0,1.0,-1.0}, multiplying them by the first element of sequence $p_{0..126}$ and inserting them into location {-21, -7, 7, 21}, respectively. The resulting frequency domain values are given in Table 10.

#	Re	Im	#	Re	Im	#	Re	Im	#	Re	Im
-32	0.0	0.0	-16	1.0	0.0	0	0.0	0.0	16	-1.0	0.0
-31	0.0	0.0	-15	-1.0	0.0	1	1.0	0.0	17	-1.0	0.0
-30	0.0	0.0	-14	1.0	0.0	2	-1.0	0.0	18	1.0	0.0
-29	0.0	0.0	-13	-1.0	0.0	3	-1.0	0.0	19	-1.0	0.0
-28	0.0	0.0	-12	-1.0	0.0	4	-1.0	0.0	20	-1.0	0.0
-27	0.0	0.0	-11	-1.0	0.0	5	-1.0	0.0	21	-1.0	0.0
-26	1.0	0.0	-10	-1.0	0.0	6	-1.0	0.0	22	1.0	0.0
-25	-1.0	0.0	-9	-1.0	0.0	7	1.0	0.0	23	-1.0	0.0
-24	-1.0	0.0	-8	-1.0	0.0	8	1.0	0.0	24	1.0	0.0
-23	1.0	0.0	-7	1.0	0.0	9	1.0	0.0	25	-1.0	0.0
-22	-1.0	0.0	-6	-1.0	0.0	10	-1.0	0.0	26	-1.0	0.0
-21	1.0	0.0	-5	1.0	0.0	11	-1.0	0.0	27	0.0	0.0
-20	1.0	0.0	-4	-1.0	0.0	12	1.0	0.0	28	0.0	0.0
-19	-1.0	0.0	-3	1.0	0.0	13	-1.0	0.0	29	0.0	0.0
-18	-1.0	0.0	-2	-1.0	0.0	14	-1.0	0.0	30	0.0	0.0
-17	1.0	0.0	-1	-1.0	0.0	15	1.0	0.0	31	0.0	0.0

Table 10-Frequency domain representation with pilots inserted.

The time domain representation is derived by performing an IFFT on the content of Table 10 and periodically extending it in the guard interval. As mentioned before windowing is not considered here. The resulting 81 sample vector which is in Inpho[] and Quado[] is presented in Table 11. We see that for each 48 input bits there are 81 complex samples.

#	Re	Im	#	Re	Im	#	Re	Im	#	Re	Im
0	0.063	0.000	20	0.010	-0.097	40	-0.035	0.044	60	-0.051	0.202
1	0.033	-0.044	21	-0.060	-0.124	41	0.017	-0.059	61	0.035	-0.116
2	-0.002	-0.038	22	-0.033	-0.044	42	0.053	-0.017	62	0.016	-0.174
3	-0.081	0.084	23	0.011	0.002	43	0.099	0.100	63	0.057	-0.052
4	0.007	-0.100	24	0.098	0.044	44	0.034	-0.148	64	0.063	0.000
5	-0.001	-0.113	25	0.136	0.105	45	-0.003	-0.094	65	0.033	-0.044
6	-0.021	-0.005	26	-0.021	0.005	46	-0.120	0.042	66	-0.002	-0.038
7	0.136	-0.105	27	-0.001	0.113	47	-0.136	-0.070	67	-0.081	0.084
8	0.098	-0.044	28	0.007	0.100	48	-0.031	0.000	68	0.007	-0.100
9	0.011	-0.002	29	-0.081	-0.084	49	-0.136	0.070	69	-0.001	-0.113
10	-0.033	0.044	30	-0.002	0.038	50	-0.120	-0.042	70	-0.021	-0.005
11	-0.060	0.124	31	0.033	0.044	51	-0.003	0.094	71	0.136	-0.105
12	0.010	0.097	32	0.063	0.000	52	0.034	0.148	72	0.098	-0.044
13	0.000	-0.008	33	0.057	0.052	53	0.099	-0.100	73	0.011	-0.002
14	0.018	-0.083	34	0.016	0.174	54	0.053	0.017	74	-0.033	0.044
15	-0.069	0.027	35	0.035	0.116	55	0.017	0.059	75	-0.060	0.124
16	-0.219	0.000	36	-0.051	-0.202	56	-0.035	-0.044	76	0.010	0.097
17	-0.069	-0.027	37	0.011	0.036	57	-0.049	0.008	77	0.000	-0.008
18	0.018	0.083	38	0.089	0.209	58	0.089	-0.209	78	0.018	-0.083
19	0.000	0.008	39	-0.049	-0.008	59	0.011	-0.036	79	-0.069	0.027
									80	-0.219	0.000

Table 11-Time domain representation of the signal.

4. Channel model

The Channel consists of AWGN noise and an UWB interference. Since the UWB pulses are very short the sampling rate of the system has to be increased in order to incorporate the UWB signal.[1] The symbol duration for the OFDM signal is 4 μs, which includes an 800 ns guard interval. Assuming a 64-point FFT, there are 64 OFDM samples in the 3.2 interval plus 16 samples in the guard interval. Adding one additional guard sample at the end of the OFDM symbol, the total number of samples is 81 samples. These samples are sent to the channel in a 4 μs interval. This translates to a sampling rate of 50 ns.

In order to use a FFT algorithm to speed up the simulation time, we need to increase the sampling rate in a way that the FFT points remain a power of two. Therefore, the sampling rate could not be increased linearly. With an 8192-point FFT, the total number of samples N_T, is

$$N_T = N_{FFT} + N_{guard} + 1 = 8192 + \frac{8192}{4} + 1 = 10241$$

[1] This assumes a technique for inserting arbitrarily positioned pulses representing the effect of UWB pulses at the receiver baseband, using a precomputed array of baseband filter impulse response samples. In the version of the simulation that is released, a different technique for inserting the UWB interference is used, as discussed in [8].

where N_{FFT} is the number of samples in the FFT period and N_{guard} is the number of samples in the guard interval. This is equal to a sampling rate of 4 µs/10241= .390 ns which seems sufficient for arbitrary positioning of the UWB pulses. Of course other FFT length numbers like 16384 can achieve a finer sampling time equal to 0.195 ns. In the following subsections we will explain the mathematical models for the noise and interference signals.

4.1. The Gaussian noise generator

A complex Gaussian random variable can be generated from two uniformly distributed random variables using the following equations [3]:

$$X = \sqrt{-2\sigma^2 \ln U_1}\cos(2\pi U_2),$$
$$Y = \sqrt{-2\sigma^2 \ln U_1}\sin(2\pi U_2).$$

where X and Y are the real and imaginary components of the Gaussian random variable and U_1 and U_2 are two random variables with uniform distribution over a [0, 1] interval using the random number generator in [6]. The power (variance) of the noise σ^2 has to be normalized to number of samples per OFDM symbol (TtlLength constant in the simulator). Moreover, since there are 48 data subcarriers out of 52 total subcarriers this factor must be taken into consideration. Since we would prefer to present the results versus Eb/N0 values for other modulations like QPSK, 16-QAM and 64-QAM the noise power is also normalized by the number of bits per symbol: 2,4 and 6 respectively.[2]

4.2. The UWB interference model

Our representation of UWB pulse relies upon a work by Miller [4]. In his model the UWB pulse is represented as an N-cycle sinusoidal burst. Based on this assumption the approximate output of the target receiver's bandpass filter is calculated. In general, the output can be presented by these equations,

$$I(t) = (a-b)h_0(t) + bh_0(t-NT)$$
$$Q(t) = (c-d)h_0(t) + dh_0(t-NT)$$

where NT is the UWB pulse duration and is equal to the minimum sampling time of the simulator, $h_0(t)$ is the impulse response of the filter and a, b, c, and d are the coefficients that depend on the carrier frequency of the victim receiver (ω_c), the initial phase of detection (ϕ). The total output of the UWB interference generator can be written as

$$I_t = \sum_k \sqrt{P_w}\, d_k I(t-k/PRF)$$
$$Q_t = \sum_k \sqrt{P_w}\, d_h Q(t-k/PRF)$$

[2] The calibration of noise power is explained in detail in [8].

where *PRF* is the pulse repetition frequency, P_w is the interference power and d_k is the random interference data. As you can see from the above equations the interference array is a series of superimposed $h_0(t)$ impulse responses at delays equal to k/PRF. In order to generate the interference we have to pump up the interference array with $h_0(t)$ at different delays with the corresponding coefficients. Also we need to generate a random pattern of data to represent d_k.

In order to calibrate the carrier to interference ratio (CIR) in the simulator we introduce a parameter called K_{CIRN} in the simulator.[3] The inphase and Quadrature components of the UWB signal looks like this,

$$I_t = \sum_k b_{adj} K_{CIRN} d_k I(t - k / PRF),$$
$$Q_t = \sum_k b_{adj} K_{CIRN} d_k Q(t - k / PRF).$$

In these equations b_{adj} is used to define different CIR values and K_{CIRN} is for CIR normalization. The calibration process starts with disconnecting the AWGN noise source. This could be achieved by simply multiplying the noise power to zero. The next step is to measure the power of the signal and interference components separately, when CIR=0 dB. The power measurement is done with this equation,

$$P_{C \ or \ I} = \sum_{K=0}^{TtlLength} I_k^2 + Q_k^2$$

Where I_k and Q_k are the inphase and quadrature samples of the desired signal or interference at the input of the receiver. By this measurement we can find the required normalization factor (K_{CIRN}) in order to make the signal and interference powers equal. Table 12 shows the values of K_{CIRN} for different modulations.

K_{CIRN}	Modulation type
√(52/1.30e-23)	BPSK
√ (52/1.30e-23)	QPSK
√ (50.9/1.30e-23)	16-QAM
√ (50.9/1.30e-23)	64-QAM

Table 12. Normalization factor for CIR ratio.

This method considers the samples of the signal in the guard interval and also pilot subcarriers as part of the desired signal power. In order to adjust the measurement we have to compensate for this factor. The ratio of the effective signal power to the above measurement is

$$F = \frac{P_{eff}}{P_m} = \frac{48}{52} \times \frac{64}{80} = .7385 = -1.32\, dB$$

[3] A method for calibrating the UWB signal power based on theory is presented in [8].

where P_m is the measured power and P_{eff} is the effective power. So this value must be considered for interference calculation. Finally, since the results are presented in terms of E_b/N_{UWB} the interference power must be normalized to the number of bits per symbol as the AWGN noise.

5. Receiver

The OFDM receiver basically performs the reverse operation of the transmitter, together with additional training tasks. In this simulator, the training tasks are ignored and the receiver only performs basic functions to recover the signal. First, the guard time is discarded and then by employing an FFT transform the signal returns to the frequency domain. The phase shift keyed values are then demapped into binary valued for hard decision decoding in the Viterbi decoder or they generate a series of number for soft decision decoding. The output of the Viterbi decoder is bit mapped in the decoder[] array.

5.1.The Convolutional Decoder

The convolutional encoder in Figure 2 has a single input data input and two outputs A and B, which are interleaved to form coded output sequence $\{A_1B_1A_2B_2 \ldots\}$. The trellis diagram generated by this encoder is presented in Figure 5. There are 64 states in this trellis that can be decomposed into 32 butterflies. This trellis diagram could be generalized to one single butterfly as depicted in Figure 6. In this figure, j represents the butterfly number that varies from 0 to 31. For the j^{th} butterfly, we can calculate the new metrics at the state j and j+32 of trellis, using the branch metrics and old accumulated metrics at node 2j and 2j+1.

The branch metrics at each butterfly can be calculated based on hard or soft decision decoding. In hard decision the output of the demodulator is hard-limited to generate the binary sequence needed for channel decoding. In soft decision the output of the demodulator is used to produce a series of numbers. The magnitude of this number represents the confidence that we have in the decoded bits. For the BPSK the received inphase value can be considered as the soft decision metrics. For QPSK, the demapping is simply taking the inphase and quadrature values as the two desired metrics. For the case of 16-QAM, the inphase and quadrature values are treated as independent 4 level PAM signals, which are demapped into 2 metrics as shown in Figure 7. The 64-QAM modulation can be decomposed to two 8 level PAM and the associated metrics can be calculated from Figure 8, [4].

6.Two examples

In order to have a better understanding of the array size in the simulation we will calculate the variables' lengths for two different scenarios. In both examples we start with the minimum packet length required to generate adequate samples (in the first case 24 bits and in the second case 288 bits). It is obvious that the real packet length can be a multiple of these values. In fact in the simulation we define a basic block length (PackBlock parameter) as 48 bits. According to different scenarios we can adjust the packet length to multiples of this value. For the first case we have a packet length equal to 48 bits (one PackBlock) that is twice the minimum packet length. For the second example the packet length is 384 bits (8*PackBlock) that is again twice the minimum length that is needed.

- **6 Mbps BPSK with ½ rate convolutional code.**

The uncoded 12 Mbps BPSK symbols consist of 48 data bits. Therefore, for ½ rate the symbol length is 24 bits. This symbol is encoded by the convolutional encoder that generates 48 bits, or 96 bits for a two-symbol packet. After adding four pilot values and 12 zeros, the input data is BPSK modulated and sent to the IFFT Block. For an 8192-point IFFT, the output consists of 8192 complex values. Adding a guard time increases the length of the array to 8192+8192/4+1=10241 points.

The bandpass filter is a root-squared raised cosine filter with 20 MHz bandwidth. This corresponds to a symbol interval equal to 50 ns, or 128 samples at a sampling rate of 0.39 ns. If we truncate the filter response to six symbol intervals the filter has 6*128=768 coefficients. Assuming a PRF= 10 MHz, the filter impulse response must be repeated every $1/PRF = 256$ samples in order to simulate a repetitive UWB signal. If we send one 24-bit packet, the interference array length is 10241. Since this is a casual system, the interference begins to appear in the interference array after one full filter length (768 samples). So we lengthen the interference array to 10241+768=11009 and then we use the values from 768 to 11009 to represent the interference array.

The input array to the receiver has a length of 10241 and it consists of signal, noise and interference. In the receiver we process this signal on a packet by packet basis. The first packet consists of 10241 samples. We discard the guard time that leaves us with 8192 samples. After taking a 8192-point FFT, the resulting 48 subcarriers are in the middle of the array. These values are demodulated and sent to the convolutional decoder to produce 24 bits of decoded data.

- **48 Mbps 64-QAM with 2/3 rate convolutional code.**

The uncoded 72 Mbps 64-QAM symbol consists of 288 bits. For the rate 2/3 code the symbol length is 192 bits. So, 192-bit symbols are sent to the convolutional coder, which generates 288 bits for each symbol. This packet is 64-QAM modulated and the output of the modulator consists of 48 QAM values per symbol. After adding the pilot channels the input consists of 52 samples and by taking a 8192-point IFFT the output is the same as the previous example.

The interference array has the same size as before (11009-point), since we used the minimum length uncoded block. Again, we truncate this array from 768 to 11009 elements and use this truncated array as the interference signal.

The input array to the receiver has 10241 complex values. The process at the receiver is the same as before however, after FFT transform the 48 QAM values are used to generate 288 soft values for the Viterbi decoder. The Viterbi decoder uses this information to generate 192 bits. Table 12 summarizes the array size for the above examples.

Array Name	Function	Length, Ex. 1	Length, Ex. 2
packet[]	Output of the packet generator	24	192
packet[]	Output of the convolutional coder	48	288
Inph[], Quad[]	Output of the modulator	48	48
Inph[], Quad[]	Input to the IFFT block	52	52
Inpho[], Quado[]	Output of the IFFT block	8192	8192
Inpho[], Quado[]	Output of the Add Guard block	10241	10241
Inpho[], Quado[]	Output of the AWGN channel	10241	10241
InphI[], QuadI[]	UWB Interference array	11009	11009
Inpho[], Quado[]	Output of the UwbInt block	10241	10241
Inph[], Quad[]	Output of the FFT block	8192	8192
dataf[]	Output of the demodulator	48	288
data[]	Output of the convolutional decoder	24	192

Table 12- Array size for different parts of the simulator.

8. Analytical approximation for the UWB interference

In this section we describe an analytical method in order to measure the BER performance of 802.11a in the presence of UWB interference. The main assumption here is that the UWB interference can be modeled as a Gaussian noise. Using this approach we could employ the BER equations for uncoded modulations as follows.

For BPSK and QPSK we have

$$P_b = Q\left(\sqrt{F.\frac{2E_b}{N_0 + N_{UWB}}}\right)$$

where P_b is the BER, E_b is energy per bit, and N_0 and N_{UWB} are the noise and interference power spectral densities, respectively. F represents the loss due to use of pilot channels and guard interval and is equal to .7385 or –1.32 dB.

For M-ary QAM (M=16, 64) the symbol error rate can be calculated as

$$P_M = 1 - (1 - P_{\sqrt{M}})^2,$$
$$P_{\sqrt{M}} = 2\left(1 - \frac{1}{\sqrt{M}}\right) Q\left(\sqrt{F \cdot \frac{3}{M-1} \cdot \frac{kE_b}{N_0 + N_{UWB}}}\right)$$

where P_M is the symbol error rate and k is the number of bits per symbol. Assuming Grey coding the BER is approximately

$$P_b = \frac{1}{k} P_M$$

The results will be presented in the next section.

9. Simulation results

Figures 9 to 14 present the simulation results for different bit rates and channel conditions. In figure 9 to 11 the BER performance of the coded and uncoded OFDM system is depicted in an AWGN channel. A 64-point FFT length is used for these simulations, since the UWB interference does not exist in these scenarios.

In figures 12 to 14 the BER performance is measured in noise and interference condition. An 8192-point FFT is used for this case and the UWB pulse only has one cycle ($N = 1$) with ($T = 0.39$ ns). These results are also compared with an analytical calculation from previous section in which the UWB is modeled as Gaussian interference. The blue curves present the simulation results and the red curves are the analytical results for the same value of E_b/N_0 and E_b/N_{UWB}.

References

1. IEEE 802.11a standard, "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," 1999.
2. Alan V. Oppenheim and Ronald W. Schafer, *Discrete-Time Signal Processing*. Prentice Hall, 1989.
3. William H. Press, Saul A. Teukolsky, William T. Vetterling and Brian P. Flannery, *Numerical Recipes in C*. Cambridge University Press, 1992.
4. Richard Van Nee and Ramjee Prasad, *OFDM for Wireless Multimedia Communications*. Boston: Artech House, 2000.
5. L. E. Miller, "Why UWB? A Review of Ultra-wideband Technology," NIST WCTG report, April 2003.
6. M. C. Jeruchim, P. Balaban, and K. S. Shanmugan, *Simulation of Communication Systems*. New York: Kluwer, 2000.
7. J. S. Lee and L. E. Miller, *CDMA Systems Engineering Handbook*. Boston: Artech House, 1998.
8. L. E. Miller, "Validation of 802.11a/UWB Coexistence Simulation," NIST WCTG report, October 2003.

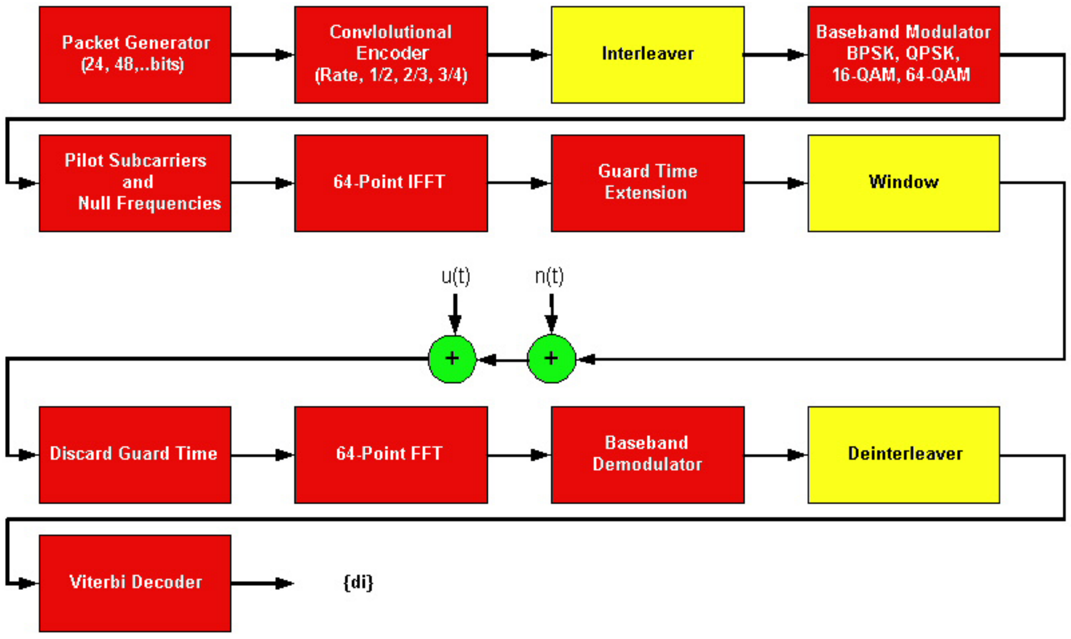

Packet Generator (24, 48,..bits)
Convlolutional Encoder (Rate, 1/2, 2/3, 3/4)
Interleaver
Baseband Modulator BPSK, QPSK, 16-QAM, 64-QAM
Pilot Subcarriers and Null Frequencies
64-Point IFFT
Guard Time Extension
Window
u(t)
n(t)
+
+
Discard Guard Time
64-Point FFT
Baseband Demodulator
Deinterleaver
Viterbi Decoder
{di}

Figure1- System diagram.

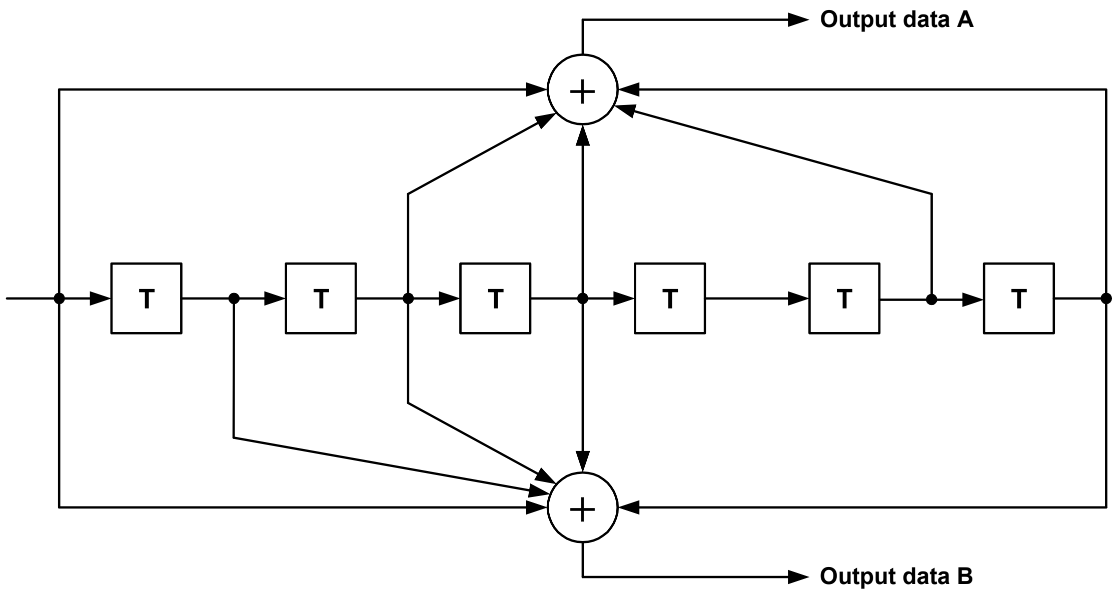


Figure2- Convolutional encoder (R=1/2).

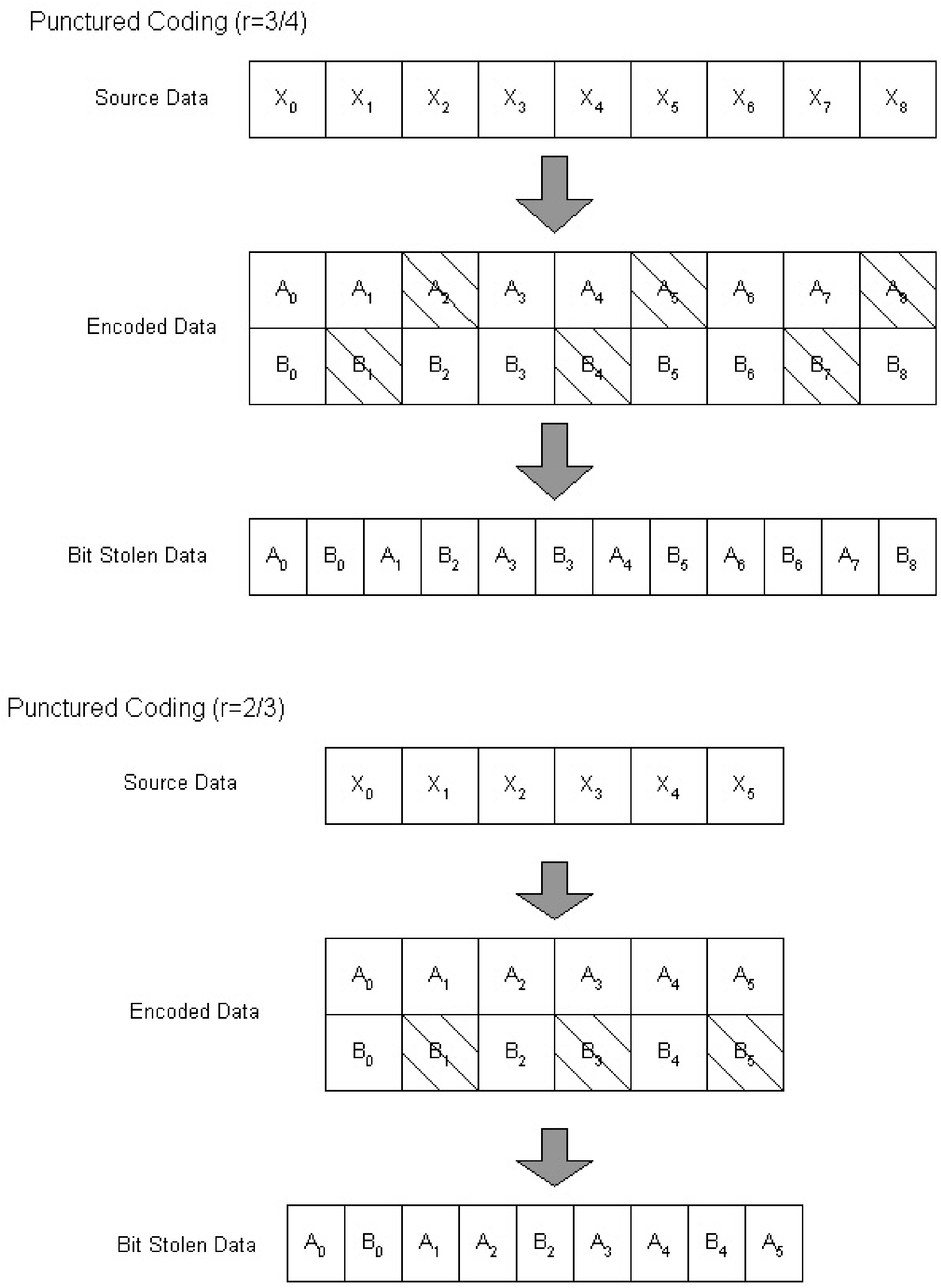

Punctured Coding (r=3/4)
Source Data
X_0 X_1 X_2 X_3 X_4 X_5 X_6 X_7 X_8
Encoded Data
A_0 A_1 A_2 A_3 A_4 A_5 A_6 A_7 A_8
B_0 B_1 B_2 B_3 B_4 B_5 B_6 B_7 B_8
Bit Stolen Data
A_0 B_0 A_1 B_2 A_3 B_3 A_4 B_5 A_6 B_6 A_7 B_8
Punctured Coding (r=2/3)
Source Data
X_0 X_1 X_2 X_3 X_4 X_5
Encoded Data
A_0 A_1 A_2 A_3 A_4 A_5
B_0 B_1 B_2 B_3 B_4 B_5
Bit Stolen Data
A_0 B_0 A_1 A_2 B_2 A_3 A_4 B_4 A_5

Figure 3- Puncturing method for higher rates.

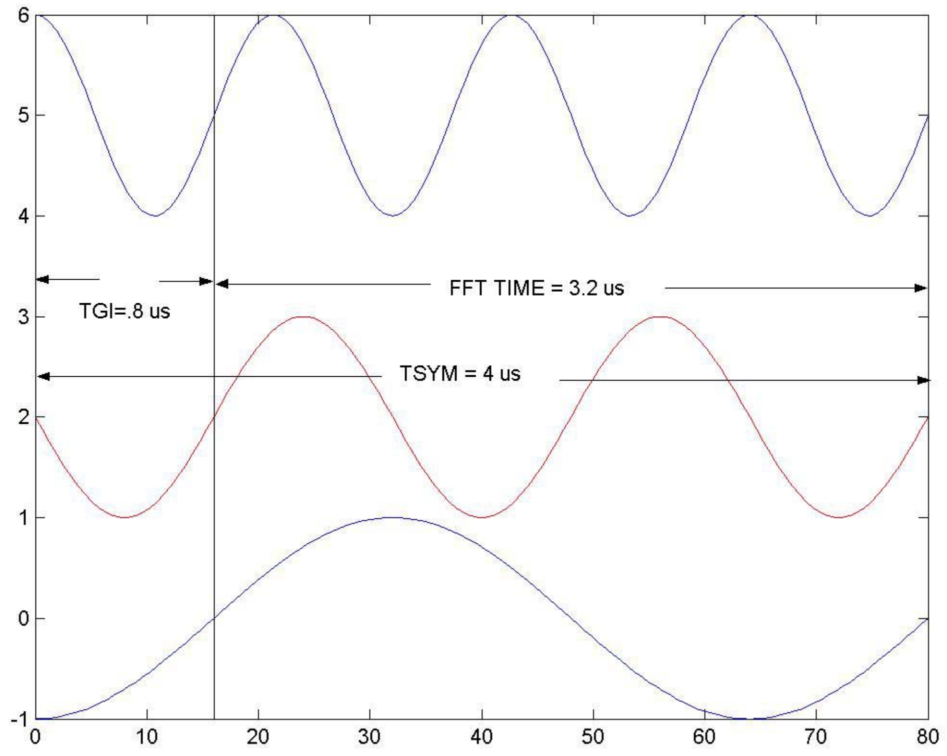


Figure 4- OFDM symbol with cyclic extension.

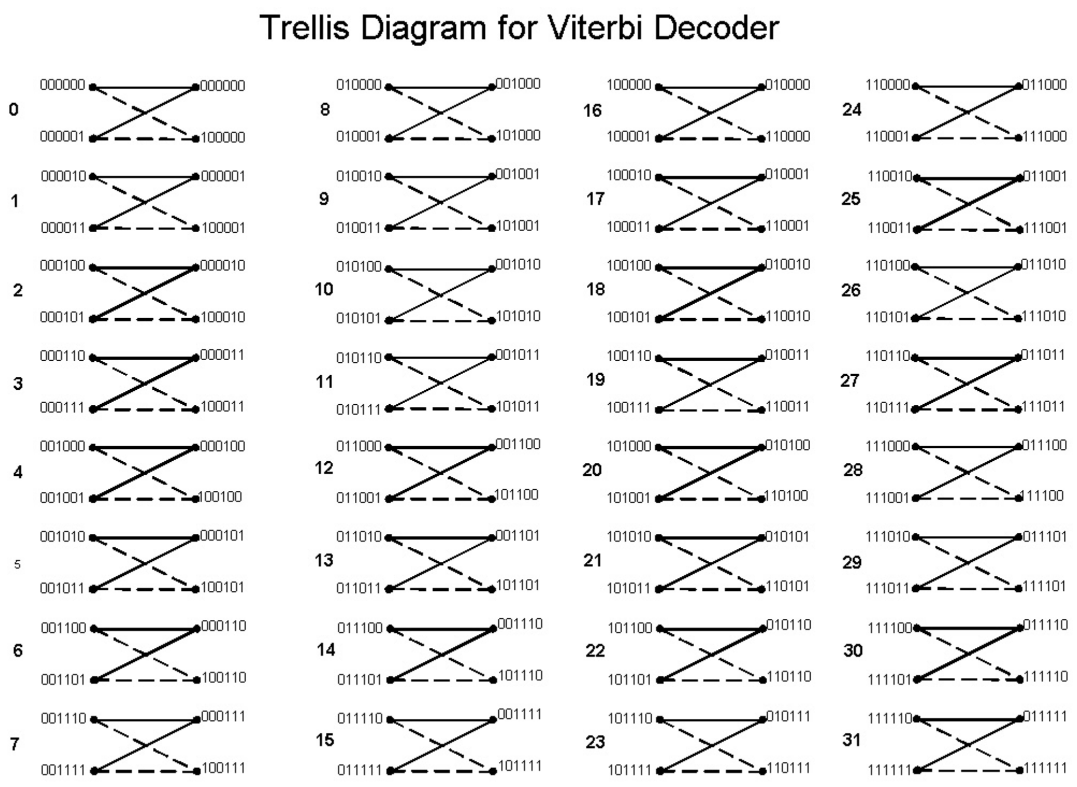
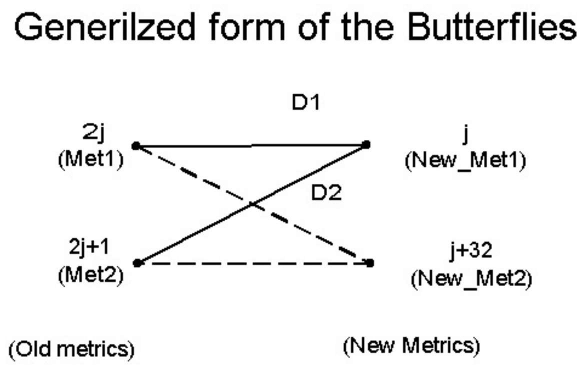


Figure 5- Trellis diagram for Viterbi decoder.



```
if(Met1+D1) > (Met2+D2)
{New_Met1=Met1+D1,d0=0}
else
{New_Met1=Met2+D2, d0=0}
```

Figure 6- Generalized form of butterfly.

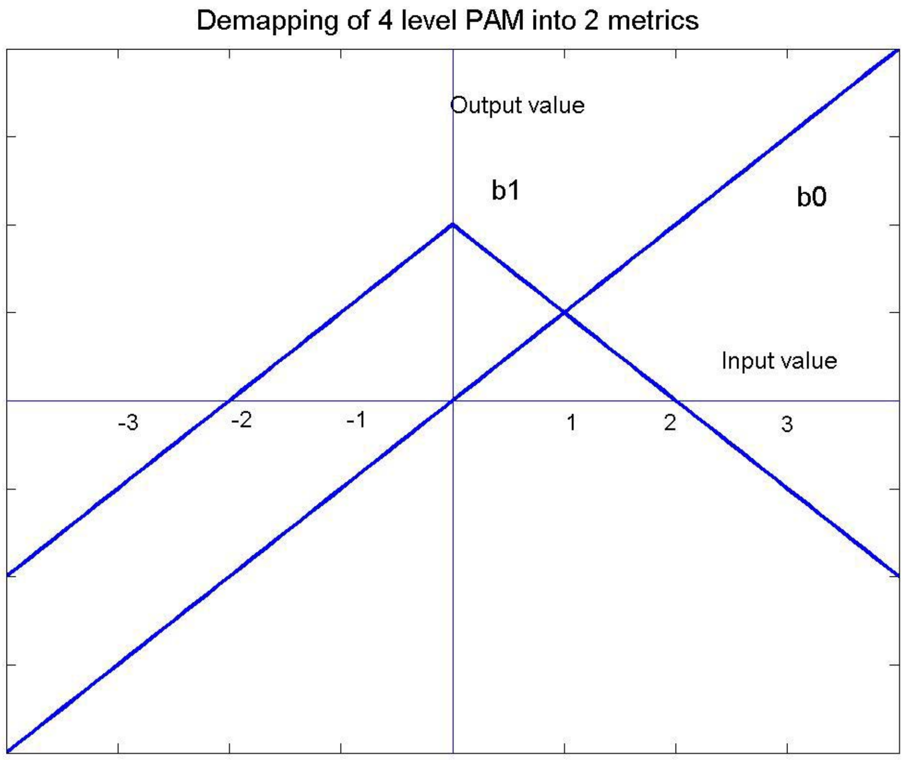


Figure 7- Demapping of 4 level PAM into 2 metrics.

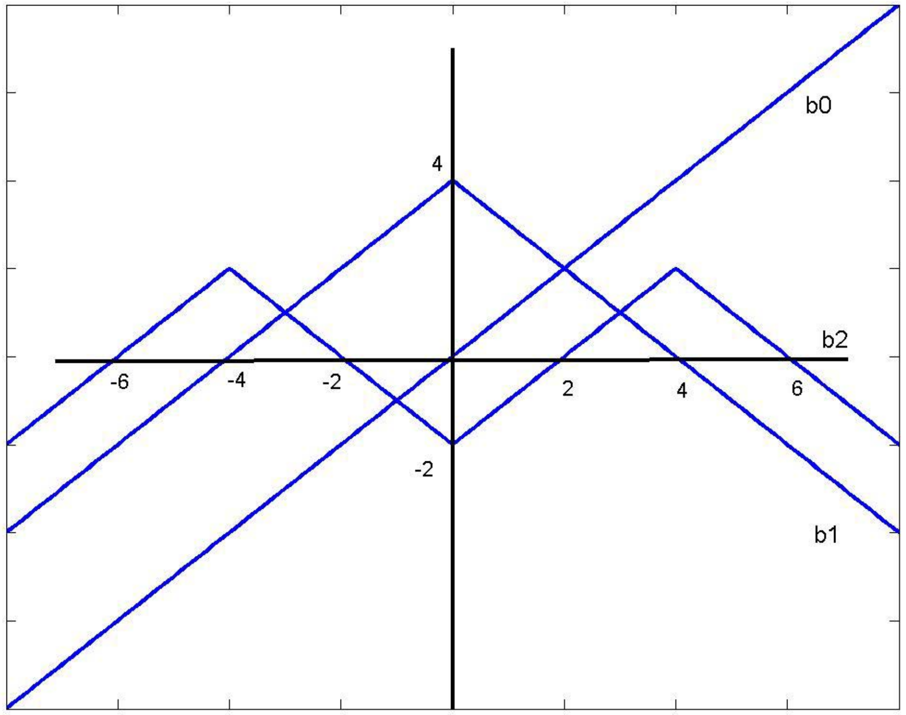


Figure 8-Demapping of 8 level PAM into 3 metrics.

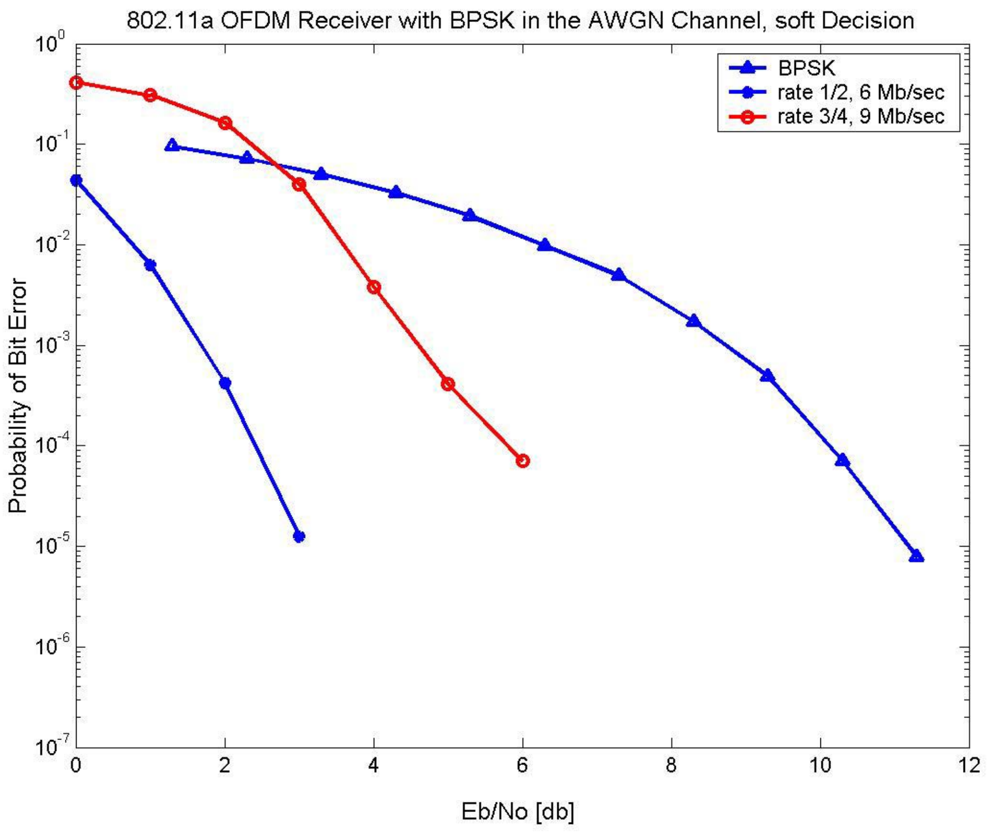


Figure 9- BPSK performance.

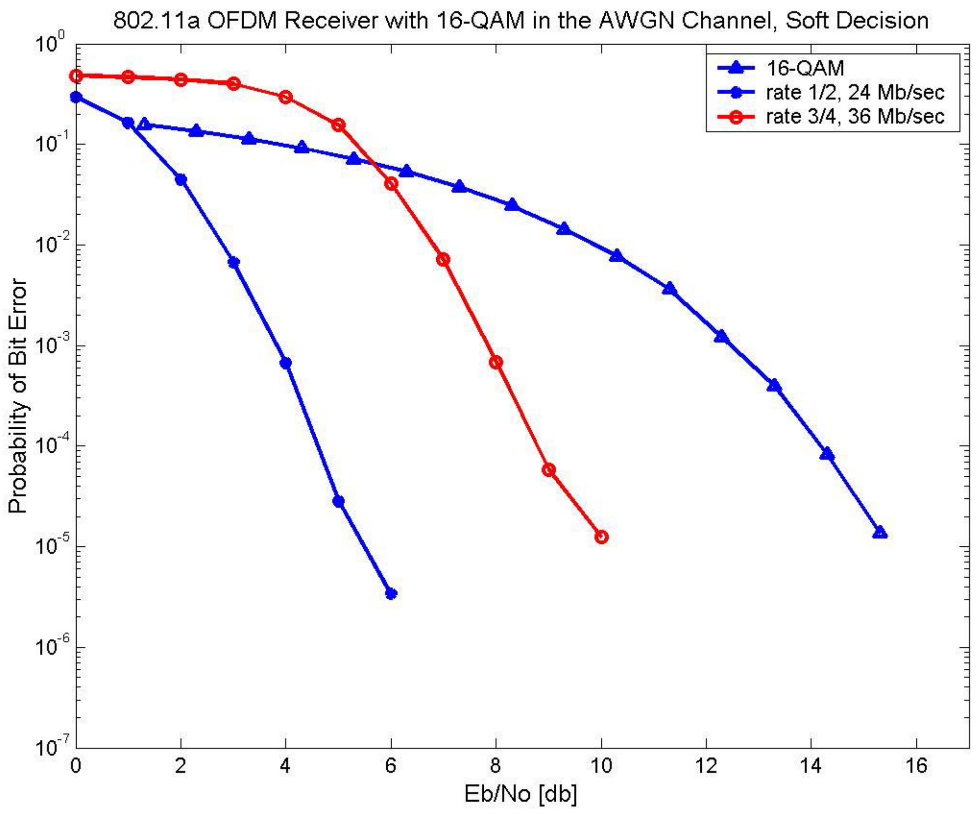


Figure 10- 16-QAM performance.

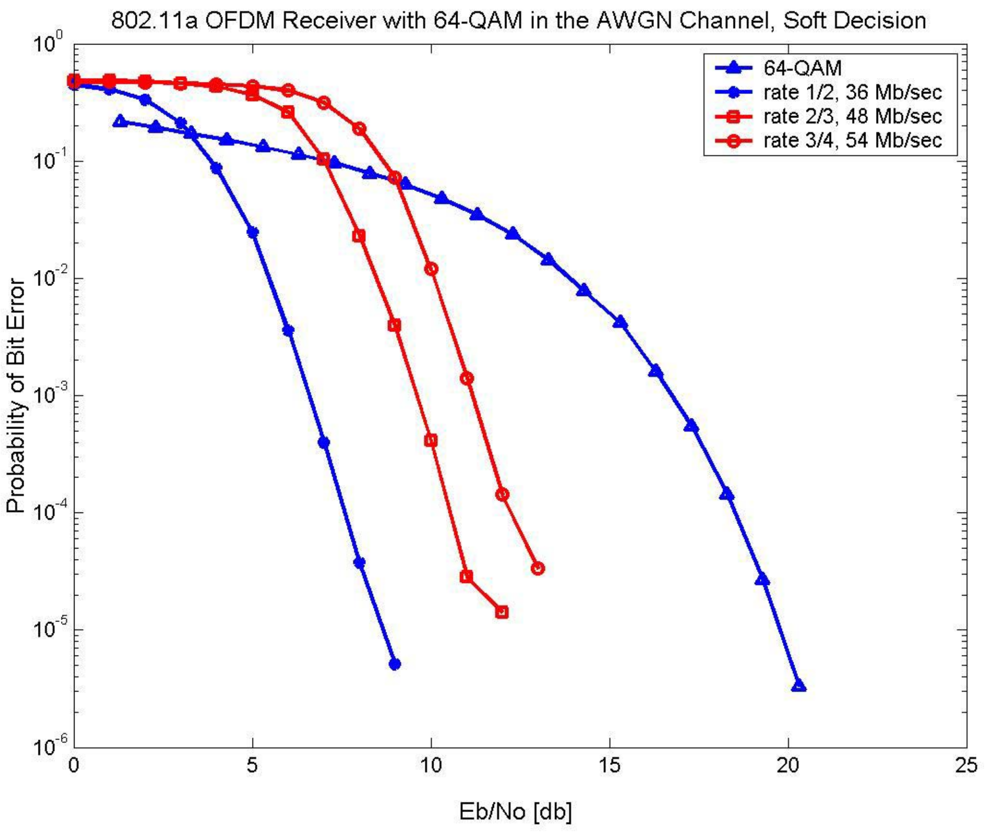


Figure 11- 64-QAM performance

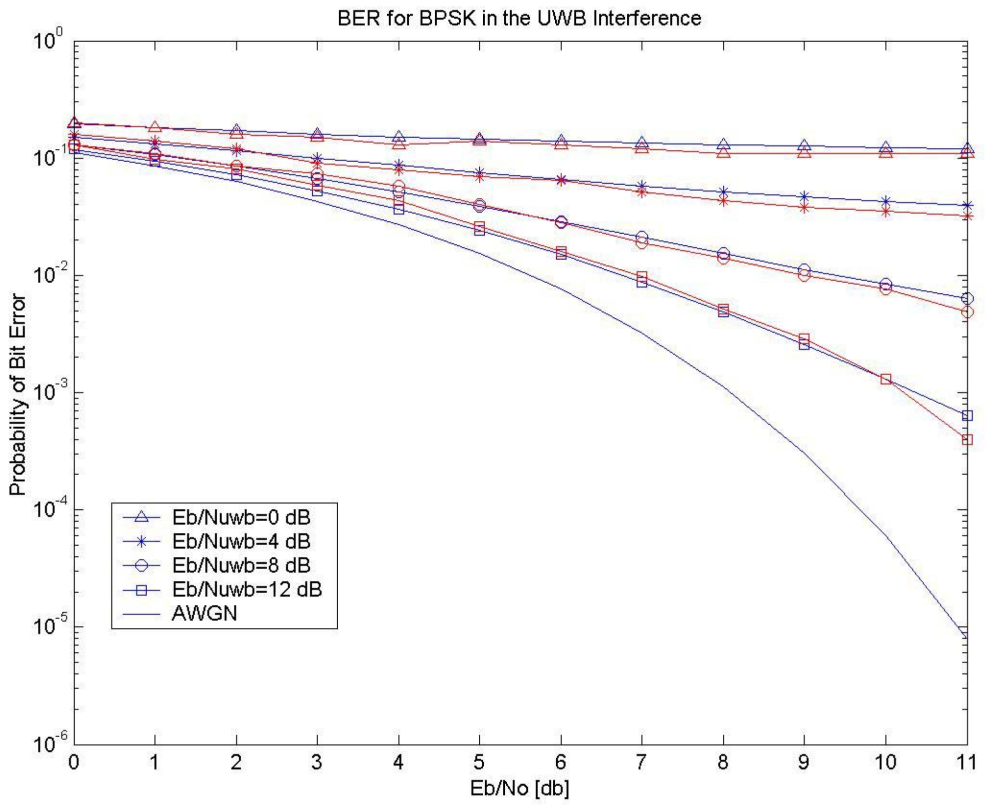


Figure 12- Uncoded BPSK with UWB interference, (simulation and Gaussian approximation).

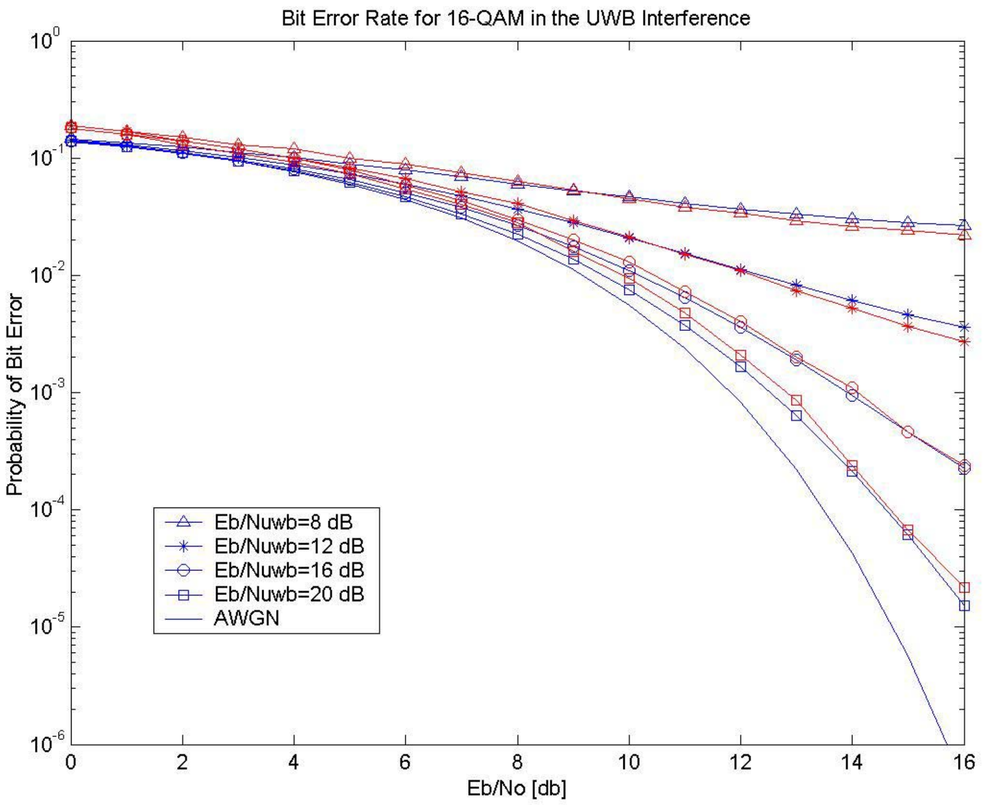


Figure 13- Uncoded 16-QAM with UWB interference (simulation and Gaussian approximation).

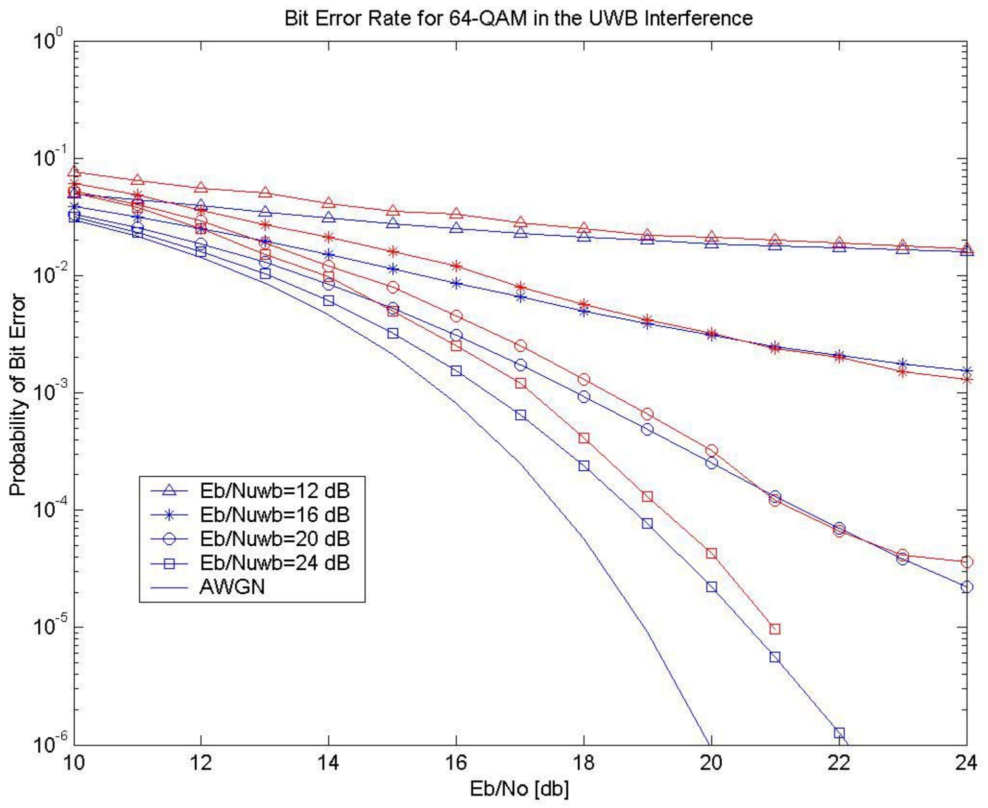


Figure 14- Uncoded 64-QAM with UWB interference, (simulation and Gaussian approximation).