Whitepaper

Using TCM Techniques to Decrease BER Without Bandwidth Compromise

Nutaq® INNOVATION TODAY FOR TOMORROW®

Using Trellis Coded Modulation Techniques to Decrease Bit Error Rate Without Bandwidth Compromise

Written by Jean-Benoit Larouche

INTRODUCTION

The challenge

In Nutaq's OFDM reference design HD video quality is sent over the air between two radio antennas. Digitally, the transmission corresponds to a throughput rate of roughly 9.6 Mbps, using a QAM-64 modulation. To achieve a low bit error rate (BER) using such a high throughput modulation, a high signal to noise ratio (SNR) is needed.

The most intuitive way to increase such a ratio is to increase the received power, either by increasing the transmitted power and/or by using amplifiers at the reception. However, in practice, sometimes this is not a viable option (ex: limited battery power in cellular phones). It is known that error correction codes (ECC) are a good way to achieve a target BER, at a lower SNR ratio. So, ECC can give us the possibility to reduce the BER at a certain SNR, or to achieve a certain BER, at a lower SNR, or, in other words, at a lower transmitted/received power. That gain, however, does not come without a cost.

Linear block codes, convolutional codes, turbo codes and even LDPC (low-density parity-check) codes are all ECC that come with a coding rate below unity. The coding rate of a code is expressed as follows:

$$R=k/n$$

where **k** is the number of information bits and n is the actual number of bits sent over a certain medium. In other words, to keep an information bit rate of 9.6 Mbps, the actual bit rate of the system would need to be 9.6 Mbps, divided by the coding rate **R** (which will bring the data rate of the system to a number greater than 9.6 Mbps).

However, in practical applications like in the Nutaq FPGA-based OFDM reference design, we are presented with real-world physical constraints (ex: limited FPGA resources, timing constraints, hardware implementation complexity, ADCs sampling rate, etc.). These constraints can actually put a limit on the date rate of the system.

By supposing that the system is limited at 9.6 Mbps, we can achieve a maximum information bit rate of 9.6 Mbps multiplied by the coding rate **R** (which is below 1).

In other words, it is likely that HD video quality won't be achievable under these conditions. That being said, is it possible to implement a coding technique which will offer us an actual coding gain for this system, without sacrificing video quality? The answer is yes: by using a coding scheme called **Trellis Coded Modulation** (TCM).

Trellis Coded Modulation

Trellis coded modulation, or TCM, was invented by Gottfried Ungerboeck in 1976, as a method to improve the reliability of a digital transmission system without bandwidth expansion or reduction of data rate. Basically, TCM is the joint effort of a convolutional coder and an M-QAM/M-PSK modulator. The following example illustrates well the concept of TCM:

In Figure 1 a), we can see that by using the QPSK modulation, we are sending 2 bits per symbol, per T seconds.

In Figure 1 b), we are using a convolutional coder of rate 2/3, to achieve some sequence coding. Since we now need to send 3 bits to be able to decode the actual 2 bits of information, a QPSK modulation would bring the symbol rate to 1.5 per T seconds, thus increasing the required bandwidth of the system. By using a higher constellation order (8-PSK), we are able to send the 3 bits, without any bandwidth expansion.

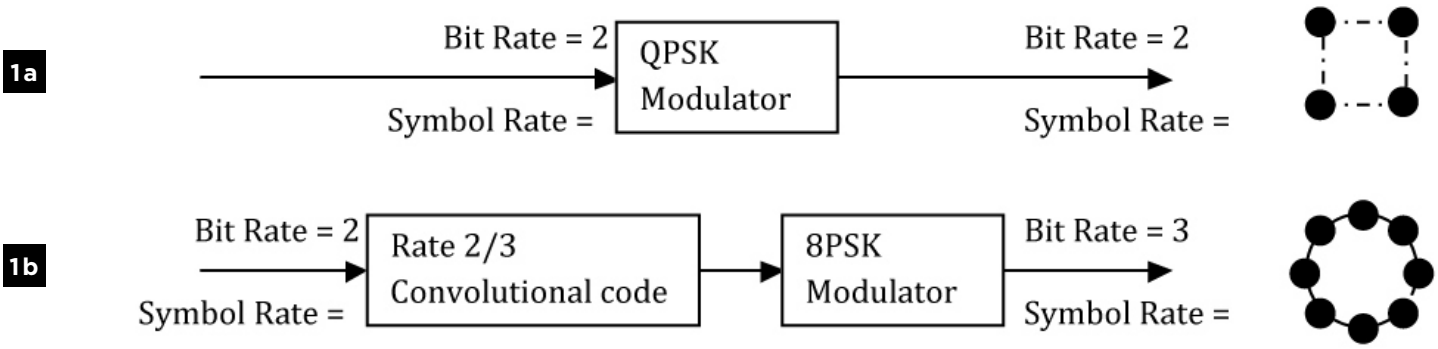


Figure 1: a*) Standard QPSK modulator, b) 8-PSK-TCM modulator*

The innovation of the TCM approach arises from the fact that the convolutional coding and the modulation are treated as one operation. In a system using convolutional coding only, we do the coding then do the modulation without any regards to the bandwidth expansion. At the receiver, demodulation would select the estimated received symbol in the constellation then use a Viterbi decoder with hamming distance as a metric, to select the Maximum Likelihood transmitted sequence (known also by the name of hard decision based Viterbi decoding). In TCM, the modulation part is done as shown in the previous figure (coding and mapping done as one operation) and at the receiver, the demodulation and the decoding is done at the same time, using a soft decision based Viterbi decoder. It's the same Viterbi algorithm but now the Euclidian distances from the different constellation symbols are used as a metric, to make a decision on the Maximum Likelihood transmitted symbol sequence.

However, if we go back to **Figure 1**, the 8-PSK symbols neighbors are closer to each other compared to QPSK, which intuitively means that in the presence of complex AWGN, a given BER will be achievable at a higher SNR, compared to the QPSK modulation scheme.

Trellis Coded Modulation *cont'd*

However, if we go back to **Figure 1**, the 8-PSK symbols neighbors are closer to each other compared to QPSK, which intuitively means that in the presence of complex AWGN, a given BER will be achievable at a higher SNR, compared to the QPSK modulation scheme. The question now is: Can a TCM coding scheme gives us a coding gain high enough so that this TCM scheme would work?

To answer this question, we need to evaluate the coding gain of such an approach. The present white paper will show, as an example, the coding gain achieved by using a QAM-16 constellation mapping with a 2/3 rate convolutional coder, compared to an uncoded 8-PSK modulation scheme. The 2/3 convolutional coder used will be the following:

2

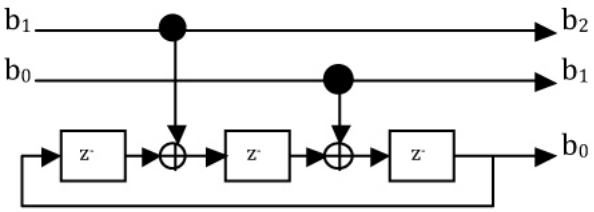


Figure 2*: A good 2/3 rate systematic convolutional coder for TCM*

3

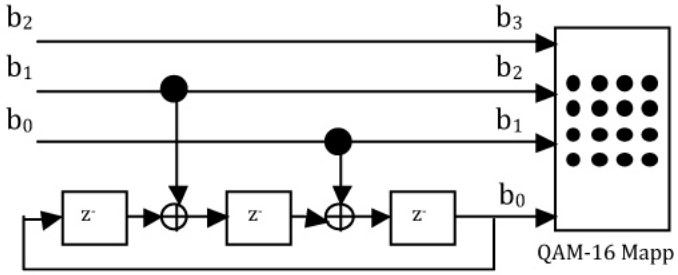


Figure 3*: A good 2/3 rate systematic convolutional coder for TCM, with its uncoded bit*

Based on this block diagram, one could ask oneself: "Why not use a 3/4 rate convolutional coder instead of using a 2/3 convolutional coder and leave one bit uncoded?"

This approach is the actual approach which was proposed by Ungerboeck and this turns out to be the key to larger coding gains for TCM schemes. The goal is to let the uncoded bit (b3) take care of itself by using a specific constellation mapping, created using a technique called Set Partitioning. We need to introduce that technique before going further.

Set Partitioning

Set partitioning is the technique of successively portion an actual constellation into cosets with increasing smallest Euclidian distance each time. The figure below shows how the QAM-16 (QAM-16?) constellation can be portioned that way:

4

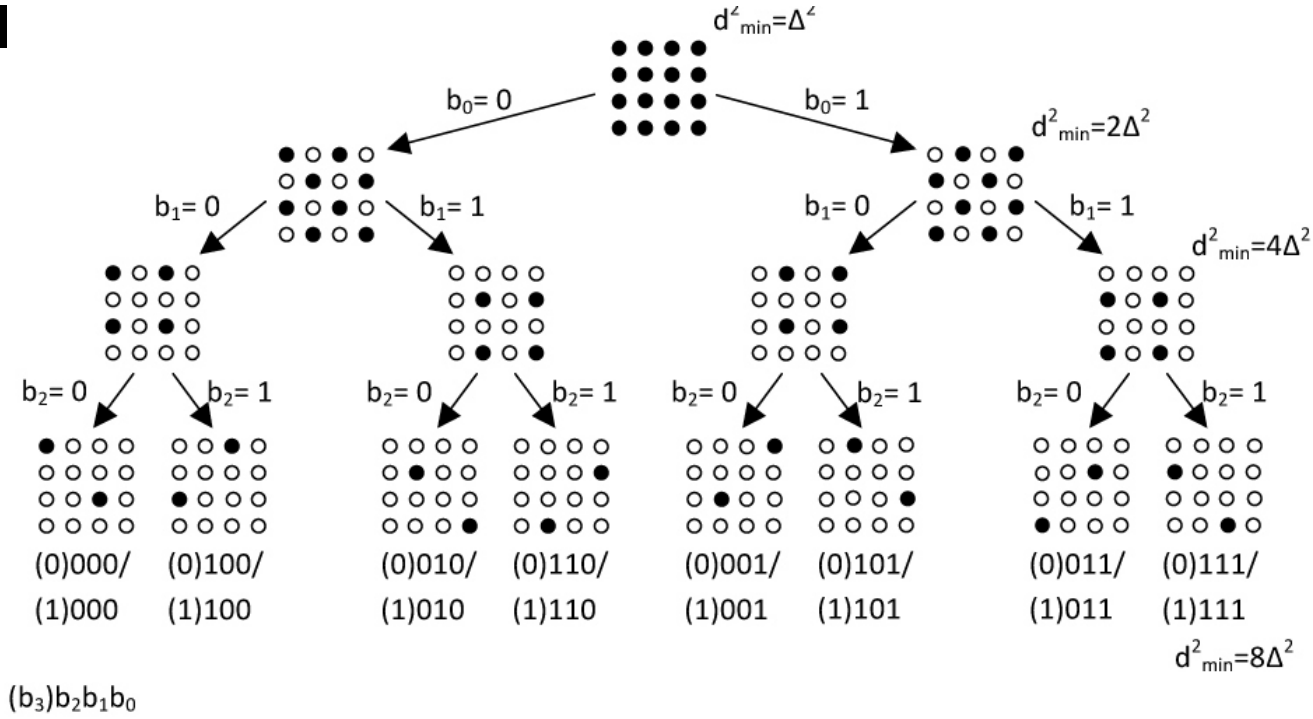


Figure 4: *Partitioning of the QAM-16 constellation for TCM*

At the beginning, all the 16 symbols can be sent over the air, with equal probability. The minimum Squared Euclidian Distance (SED) is Ð2. Once b0 has been selected to be '0' or '1', the possible sent symbols are now located in subset of the QAM-16 original constellation. Since only 8 symbols can be possibly sent now, the minimum SED is 2Ð2 for this subset. We are repeating the process for b1 and b2.

From the last coset, the uncoded bit (b3) will chose one of the two points. The bits b0 to b2 are the ones in the cosets having the smallest squared Euclidian distances, so we will map the coded bits to these cosets, since errors on those bits are more likely to occur in a complex AWGN channel. The uncoded bit (b3), will be the one selecting the transmitted symbol from the large Euclidian distance of that coset, since it does not have any sequence coding.

Set Partitioning *cont'd*

By using this approach, we are able to protect the most common bits in errors by using a 2/3 convolutional coder (which gives better performance than a 3/4 one) and still protect the last bit, from the large Euclidian distance of his coset. Recalling the convolutional encoder proposed (see **Figure 3**), we can also distinguish another advantage of using this partitioning method.

If we check the connections for b_0, we can see that there is a box of delay before its output. Which means, that at time 't', b_2 and b1 won't have any influence on b_0, since the sent symbol at time 't' will be composed of $b_3(t)$, $b_2(t)$, $b_1(t)$ and $b_0(t-1)$. So, during decoding, a diverging trellis path will occur only on b_2 and b_1, which are mapped into a coset with a larger squared Euclidian distance then b_0.

Using the set partitioning technique, the general structure of a TCM encoder is as follows:

5

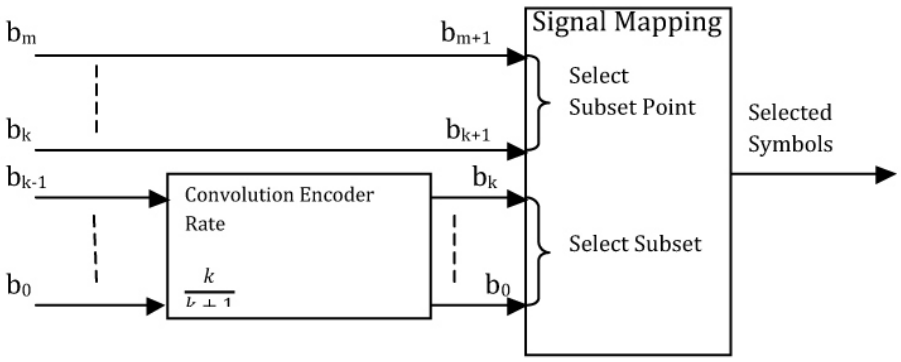


Figure 5: *General TCM encoder block diagram*

From this block diagram, we can that by using different convolutional encoders and by using a different number of uncoded bits, we can reach different coding rate. However, not all convolutional encoders can be considered 'good' for TCM. In his paper [1,2], Ungerboeck list the encoders found by computer search, which provide the best results:

	g_0	g_1	g_2	g_3	$d^2_{free}/\triangle^2$	A_{dfree}	B_{dfree}	Asympt. Gain (dB) QAM-16/8-PSK
4	5	2	-	-	4.0			4.4
8	11	2	4	-	5.0	3.656	12.344	5.3
16	23	4	16	-	6.0	9.156	53.5	6.1
32	41	6	10	-	6.0	2.641	16.063	6.1
64	101	16	64	-	7.0	8.422	55.688	6.8

Table 1: *Table of good TCM encoders*

The numbers gi are the connection polynomials in octal format. Given example, our encoder presented in **Figure 2** is represented by the polynomials stated on the second line: 11, 2 and 4. Using that encoder, we should get a coding gain of 5.3 dB. We can also distinguish from **Table 1** that increasing the number of states in the convolutional coder increases the free distance of the code, however this is not in a linear fashion. Let's see how we can validate the theoretical coding gains in the next section.

Coding Gain

The coding gains of TCM schemes are evaluated asymptotically at high SNR, by the following formula:

$$G = 10log\left[\left(\frac{\delta^2_{free}}{\delta^2_{free,u}}\right)\left(\frac{E_{s,u}}{E_s}\right)\right]$$

where δ^2_{free} and $\delta^2_{free,u}$ are the squared free distances of the TCM and uncoded schemes, respectively. E_s and $E_{s,u}$ denote the average signal energies of the TCM and uncoded systems, respectively. Normalizing the average symbol energy to 1 for each constellation can remove these terms from the equation.

However, the squared free distance ratio can be changed dramatically by the usage of different convolutional encoders, thus δ^2_{free} is the main parameter affecting the coding gain. The free distance in a convolutional encoder can be evaluated through its trellis diagram and is a measure of the error correcting capability of the code. Since in TCM the coding and the modulation are done together, the squared free distances are defined as the smallest Euclidian distance of a coded symbol sequence to the all-zero symbol sequence. Let's evaluate $\delta^2_{free,u}$ of the uncoded 8-PSK modulation scheme first:

6

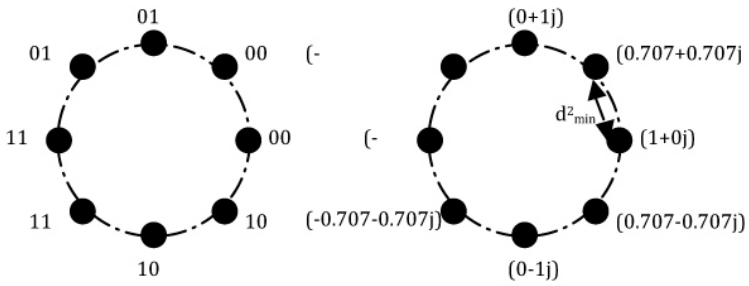


Figure 6: *8-PSK constellation with normalized signal energy*

In the uncoded case, since no convolutional coding is involved, a received symbol is not related to the previous sent symbols, thus $\delta^2_{free,u} = d^2_{min}$ which is the minimal squared Euclidian distance between one point and its closest neighbor. In the 8-PSK case, it can be easily calculated from the constellation:

$$d^2_{min} = (1 - 0.707)^2 + 0.707^2 = 0.586$$

Coding Gain *cont'd*

For the coded case, the squared free distance depends on the convolutional coder. Recalling the proposed coder (see **Figure 2**), we can list the actual state transitions depending on the input bits:

From **Table 2**, we can list the shortest possible paths which start at S0 and end at S0:

- S0-S2-S0
- S0-S1-S4-S0
- S0-S1-S6-S0
- S0-S3-S4-S0
- S0-S3-S6-S0

Input bits (b1, b0),t		R0,t	R1,t	R2,t	Output bits (b2, b1, b0),t			R0,t+1	R1,t+1	R2,t+1	Next State
0	0	0	0	0	0	0	0	0	0	0	S0=>S0
0	1	0	0	0	0	1	0	0	0	1	S0=>S1
1	0	0	0	0	1	0	0	0	1	0	S0=>S2
1	1	0	0	0	1	1	0	0	1	1	S0=>S3
0	0	0	1	0	0	0	1	1	0	0	S1=>S4
0	1	0	1	0	0	1	1	1	0	1	S1=>S5
1	0	0	1	0	1	0	1	1	1	0	S1=>S6
1	1	0	1	0	1	1	1	1	1	1	S1=>S7
0	0	0	0	1	0	0	0	0	0	0	S2=>S1
0	1	0	0	1	0	1	0	0	0	1	S2=>S0
1	0	0	0	1	1	0	0	0	1	0	S2=>S3
1	1	0	0	1	1	1	0	0	1	1	S2=>S2
0	0	0	1	1	0	0	1	1	0	0	S3=>S5
0	1	0	1	1	0	1	1	1	0	1	S3=>S4
1	0	0	1	1	1	0	1	1	1	0	S3=>S7
1	1	0	1	1	1	1	1	1	1	1	S3=>S6
0	0	1	0	0	0	0	0	0	0	0	S4=>S2
0	1	1	0	0	0	1	0	0	0	1	S4=>S3
1	0	1	0	0	1	0	0	0	1	0	S4=>S0
1	1	1	0	0	1	1	0	0	1	1	S4=>S1
0	0	1	1	0	0	0	1	1	0	0	S5=>S6
0	1	1	1	0	0	1	1	1	0	1	S5=>S7
1	0	1	1	0	1	0	1	1	1	0	S5=>S4
1	1	1	1	0	1	1	1	1	1	1	S5=>S5
0	0	1	0	1	0	0	0	0	0	0	S6=>S3
0	1	1	0	1	0	1	0	0	0	1	S6=>S2
1	0	1	0	1	1	0	0	0	1	0	S6=>S1
1	1	1	0	1	1	1	0	0	1	1	S6=>S0
0	0	1	1	1	0	0	1	1	0	0	S7=>S7
0	1	1	1	1	0	1	1	1	0	1	S3=>S6
1	0	1	1	1	1	0	1	1	1	0	S3=>S5
1	1	1	1	1	1	1	1	1	1	1	S3=>S4

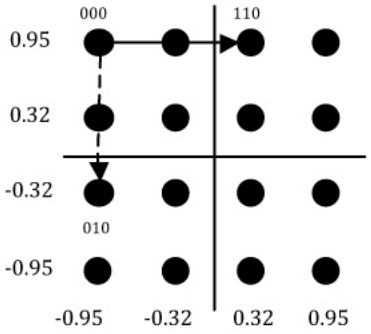
Table 2 : *State transition table for the 2/3 convolutional coder*

Coding Gain *cont'd*

All these paths correspond to an actual QAM-16 symbol sequence. With the QAM-16 partitioning provided in the previous section and the state transitions table, we can evaluate the cumulative squared Euclidian distance of each path.

For example, the path S0-S2-S0, corresponds to the symbol sequence (X)100-(X)010. The (X) can be either 1 or 0 and account for the uncoded bit. These are called parallel transitions. By using the QAM-16 constellation with their generated and normalized real and imaginary values, we can now calculate the squared Euclidian distance of this path:

7a



7b

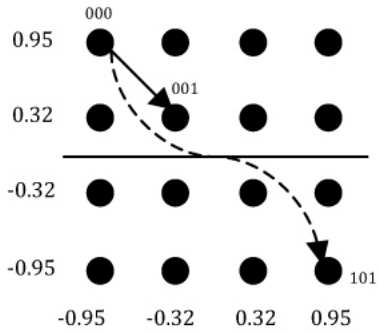


Figure 7: a) *The two possible generated symbols for a S0 to S2 transition,*
b) *The two possible generated symbols for a S2 to S0 transition*

In **Figure 7 a)**, for the transition S0-S2, we send the symbol 0100 or 1100. Both have the same squared Euclidian distance from 0000. Similarly, in **Figure 7 b)**, the symbol 0100 has the smallest squared Euclidian distance (see **Figure 4** for the complete symbol mapping). For this specific path, we have:

$$\delta^2_{free} = [2*(0.95-0.32)]^2+[2*(0.95-0.32)^2] = 2.3814$$

Doing this for all the previously listed paths, we get the following distances:

- S0-S2-S0: (X)100-(X)010: 2.3814
- S0-S1-S4-S0: (X)010-(X)001-(X)100: 2.7783
- S0-S1-S6-S0: (X)010-(X)101-(X)110: 1.9845
- S0-S3-S4-S0: (X)110-(X)011-(X)100: 2.7783
- S0-S3-S6-S0: (X)110-(X)111-(X)110: 1.9845

The smallest free squared Euclidian distance is 1.9845, but we are not done yet. What about the uncoded bit? Since this bit is not part of the sequence coding, it has its own free distance but thanks to the set partitioning technique, this bit selects the symbol in the coset with the largest squared Euclidian distance. If we look at the QAM-16 constellation, the squared Euclidian distance of the last coset is:

$$d^2_{b3}=2*[2*(0.95-0.32)]^2=3.1752$$

Thus, the actual free distance of this TCM scheme, is given by the following equation:

$$\delta^2_{free} = \min[1.9845,3.1752]=1.9845$$

The free distance of the convolutional encoder being smaller, errors regarding the trellis decoding are more likely to happen and this is the number we choose to determine the coding gain of this TCM scheme. The coding gain is then:

$$G = 10log\left[\left(\frac{\delta^2_{free}}{\delta^2_{free,u}}\right)\right] = 10log\left[\left(\frac{1.9845}{0.586}\right)\right] = 5.2975\ dB$$

as listed in **Table 1.**

Decoding

Since TCM is a mix of convolutional coding and constellation mapping, the decoding process will use a Viterbi decoder, with a soft metric: the Euclidian distance from the constellation point. This decoding process will be used to make a maximum likelihood decision on the sent coded bits, but we still have an uncoded bit to decode. The general decoding process will be as follows:

1. At *t=0*, initialize the cumulative distance at $S0_t$ at a value of 0, since we are 100% sure that the coder starts in that state. Increment t to 1

2. For each value of t higher than 0, calculate the branch metric of each path ongoing to each state, for all the states. For example, starting at S0 at *t=0*, a transition to S1 means the transmission of the symbol (X)010, or in complex values, -0.32+0.32j/0.95-0.95j. There are two possibilities for the transmitted symbol because of the uncoded bit. Let's assume we received 0.90-0.84j; then we can easily state that the Euclidian distance from the symbol 0.95-0.95j will be the smallest one with a value of 0.12. Also, we can be pretty confident that the uncoded bit is a '1', from the constellation mapping.

3. Since the cumulative distance at *t=0*, from S0, is 0 and the transition from S0 to S1 has a value of 0.12, the cumulative Euclidian distance is then: 0+0.12=0.12. However, even if this value is small, from S0 at t=0, we can go to many other states (S0, S1, S2, S3). Calculating the Euclidian distance for all these possible transitions for each *'t'*, will give us soft metrics to help us decode the maximum likelihood sent sequence.

4. Once the most likely sent sequence has been found, we need to find the sequence of information bits. We do this by simply removing the coded bit (remember that the encoder is systematic, so there is only one coded bit (b_0), the 3 others are the actual information bits.

5. Calculate the Bit Error Rate.

Here is an example of the process at an E_b/N_0 of 10 dB: (See **Figure 8)**

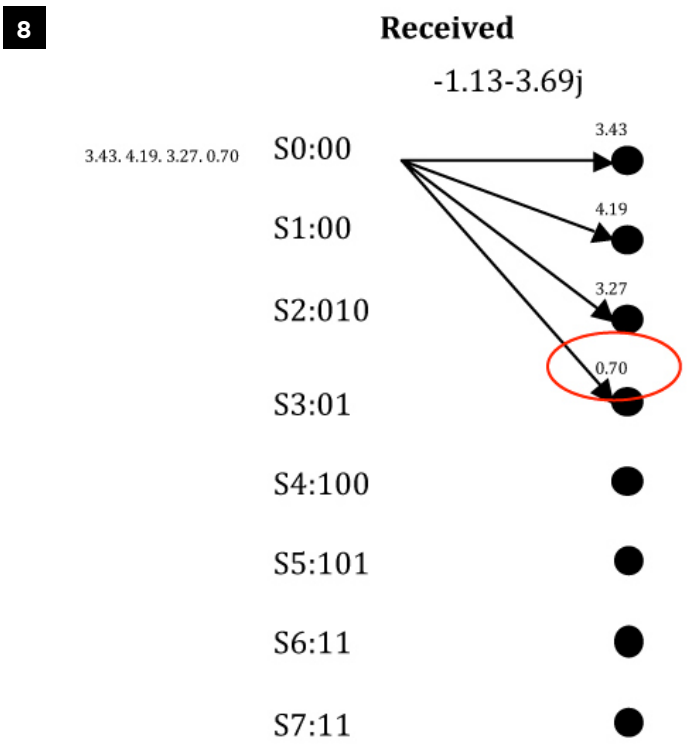


Figure 8: *First transition trellis decoding*

Decoding *cont'd*

At first, we are certain to start at S0. On the left, you have the branch metric for the transitions S0-S0, S0-S1, S0-S2, S0-S3, respectively. Since the distance from S0 at time 0, is 0, the branch metrics becomes the cumulative metric. We can see that the transition from S0 to S3 is the maximum likelihood transition since its Euclidian distance from the constellation points leading to this state are the smallest ((X)110=>-1-3j or 3+1j). Let's move to the second step: (see **Figure 9**)

9

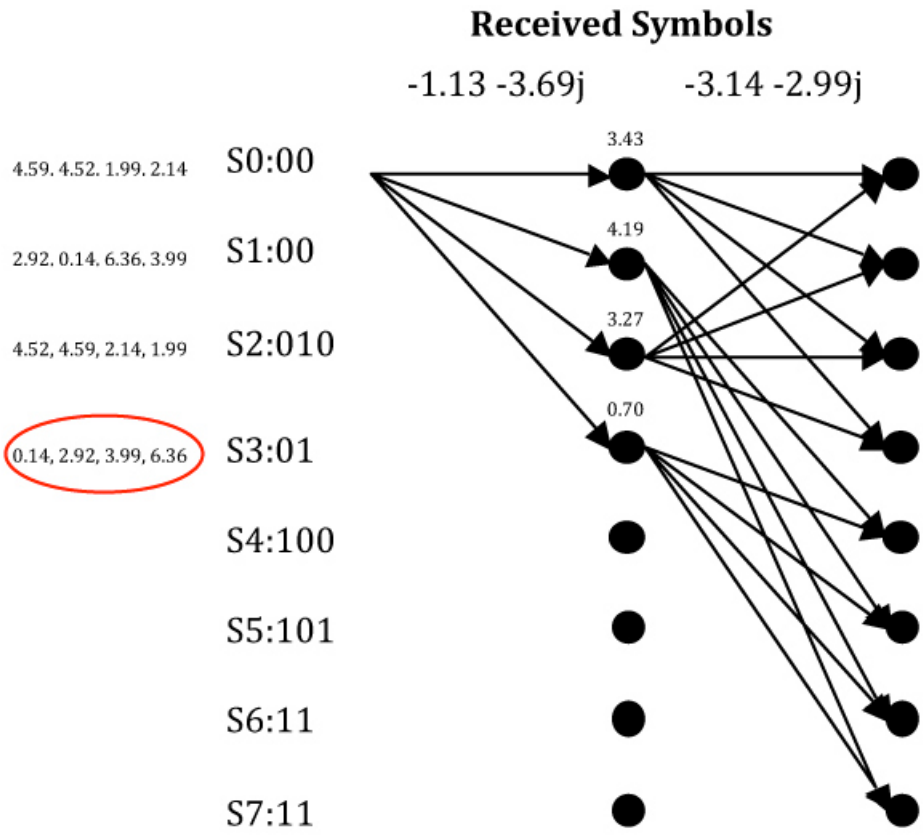


Figure 9: *Second transition trellis decoding*

Decoding *cont'd*

At the second time period, the second received symbol is -3.14-2.99j. Calculating the Euclidian distance from the constellation points linked to the different transitions, we find that the transition from S3 to S4 is the maximum likelihood transition with a value of 0.14. But a transition to S4 can also occur from S1. Let's now calculate the cumulative metric for each state and keep the transitions with the lowest cumulative Euclidian distance: see **Figure 10**)

10

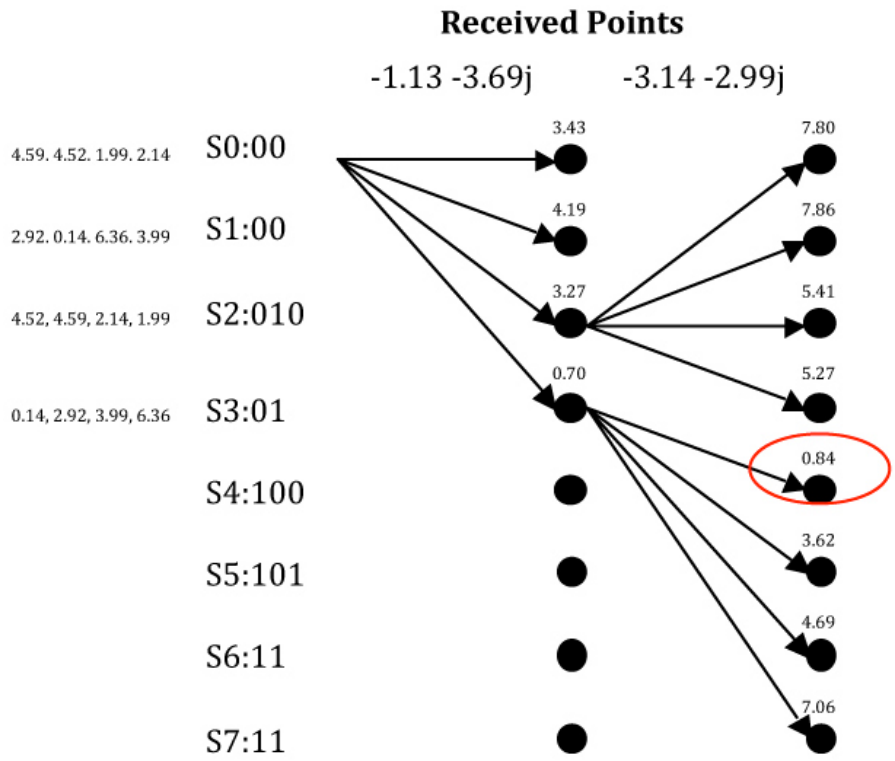


Figure 10: *Filtering of the highest cumulative distances*

In that third step, half of the transitions were removed. Only one path is going to each state and we see that the transition from S1 to S4 was removed, since that cumulative Euclidian distance (4.19+0.14) was much higher than 0.84 (0.70+0.14).

The same 3 steps need to be repeated as needed, depending on the number of symbols in the coded sequence. However, for the sake of this example, let's suppose that the sequence coding ends here. The selected sequence would be S0-S3-S4, leading to a decision regarding the sent bits of (X)110-(X)011. Since the encoder is systematic, removing the parity bit gives us the information bits ((X)11 and (X)01).

Now, we need to make a decision on the uncoded bit, again based the Euclidian distance metric. We can calculate easily that the S0-S3 transition occurred with the value of 0.70, because of the symbol 0110 (-1-3j) which was pretty close to the received symbol. In the same fashion, the transition S3-S4 occurred because of the symbol 0011 (-3-3j) which was very close to the second received symbol. The final decision on the transmitted information bits is: 011 and 001.

Bit Error Rate Simulation

In his paper [1,2], Ungerboeck approximated the error-event probability with the following formula:

$$\Pr(e) \approx N_{free} Q\left[{}^{d_{free}}/_{2\sigma}\right]$$

where Nfree is the average number of nearest neighbor signal sequences with distance dfree that diverge at any state from a transmitted signal sequence, and remerge with it after one or more transitions. Let's rewrite the standard deviation (Ð) to get an expression as a function of Eb/N0. Since:

$$\mathrm{E_s} = \mathrm{RE_b}$$

$$\frac{\mathrm{E_s}}{\mathrm{RN_0}} = \frac{\mathrm{E_b}}{\mathrm{N_0}}$$

$$\sigma^2 = \mathrm{N_0}$$

with'R' being the numbers of bits per symbol (4 in our case) multiplied by the coding rate (3/4 in our case). We can now rewrite the variance by the following expression:

$$\sigma^2 = \left(R\frac{\mathrm{E_b}}{\mathrm{N_0}}\right)^{-1}$$

Using this expression, we get the following error-event probability expression:

$$\Pr(e) \approx N_{free} Q\left[\sqrt{\frac{d_{free}^2 RE_b}{4N_0}}\right]$$

However, this expression is only the error-event probability expression. Todd K. Moon, in his book on error control coding [4], mentions a pretty similar expression for the bit-error probability:

$$\mathrm{P_b} \approx \frac{1}{k} B_{free} Q\left[\sqrt{\frac{d_{free}^2 RE_b}{4N_0}}\right]$$

where Bfree is the average number of bit errors on the paths of distance dfree from the all-zero sequence and 'k' is the number of information bits. The values of Bfree are listed in Table 1. Thus, the final expression is as follows:

$$\approx \frac{1}{3} * 18.313 * Q\left[\sqrt{\frac{5 * 3/4 * E_b}{N_0}}\right]$$

Using this expression, we can compare the BER of an actual simulated TCM system with the theoretical curve.

Matlab Simulation

For the Matlab simulation, the encoder and the decoder proposed in the previous sections were used. The length of the symbol sequence was fixed to 3000 symbols. Below is the comparison of the simulated QAM-16 TCM scheme versus the theoretical bit-error probability curve, the uncoded 8-PSK modulation scheme and the uncoded QAM-16 BER curve:

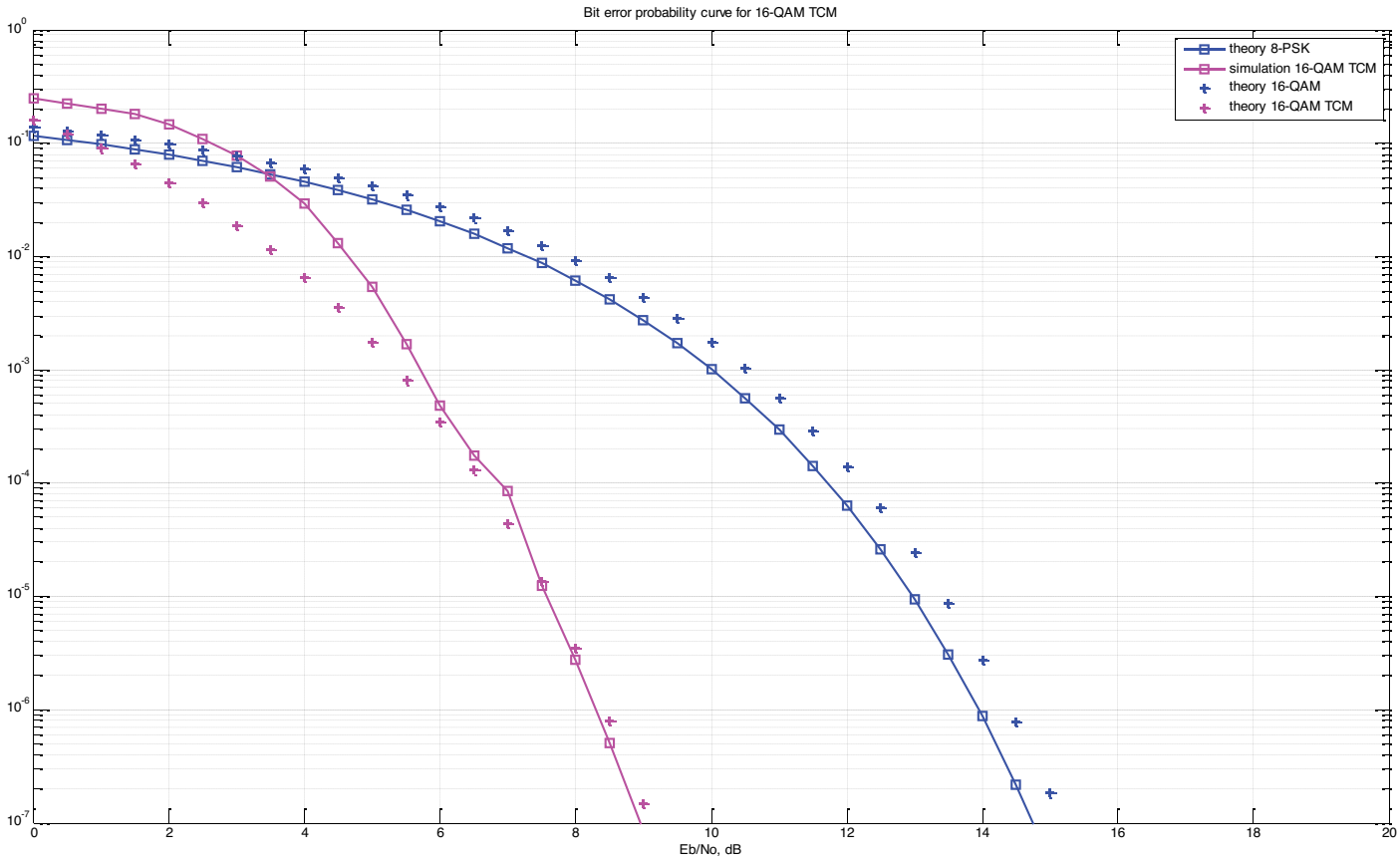


Figure 11: *Simulation results versus theoretical expressions*

Matlab Simulation *cont'd*

We can see that at a low SNR, the theoretical curve is not following the simulated results. However, at a higher SNR, the simulated results are following quite nicely the theoretical results. Also, we can confirm the asymptotic coding gain of 5.3 dB of this TCM scheme approximately at 7 dB of E_b/N_0. Here is the pseudo code of the Matlab program:

for all E_b/N_0 values
    while *Error number is below threshold*
        Generate (3000 * 3 information bits per symbol) random bits;
        Put information bits in blocks of 3 bits => leads to '3000';
        State of the encoder => S0;

            for *all blocks of 3 bits*
                [block of 4 bits, Next Encoder State]=Convolutional Coder(block of 3 bits, Current Encoder State);
                Current Encoder State = Next Encoder State;
            end for;
            for each block of 4 bits
                QAM-16 symbol mapping by using a Look Up Table;
            end for;

        Generation of number of symbols complex Gaussian noise samples (3000);
        Noise addition to each QAM-16 symbol;

        for *first received noisy QAM-16 symbol to the last received noisy QAM-16 symbol*

            Calculate branch metric with Euclidian distance metric;
            Cumulative Euclidian distance calculation for each path, for each state;
            Keep the smallest cumulative Euclidian distance value for each state;
            Move to the next received symbol;

        end for;

        for *all the trellis*

        Decode the informations bits, based on the path with the lowest cumulative Euclidian distance metric;
        Decode the uncoded bit for each received symbol, depending on the closest symbol from the 2 possible symbols;
        Compare decoded bits with the actual sent information bits;
        Calculate BER for the specific E_b/N_0;

        end for;

    end while;

end for;

Plot 8-PSK theoretical BER curve;
Plot QAM-16 theoretical BER curve;
Plot QAM-16 TCM scheme theoretical BER curve;
Plot QAM-16 TCM simulation BER curve;

Conclusion and discussion

This white paper has explained the concept of Trellis Coded Modulation and how we can get a coding gain, without any bandwidth expansion, by the usage of higher power constellations. High coding gains possible through the technique of Set Partitioning are still high enough to be definitely useful in any bandwidth limited system. Also, TCM is much simpler to implement in a hardware architecture (like an FPGA) compared to other techniques like LDPC, Reed-Solomon and turbo coders.

The encoder is simple to implement with delay registers and the decoder uses a soft Viterbi decoder which requires only Euclidian distance calculations. By using the squared Euclidian distance as a metric (removing the square root), the implementation will requires only adders, multipliers and comparators. All these basics mathematics operation are offered as part of the System Generator library (used to develop the Nutaq OFDM reference design). So, we can see that we would require mostly linear operations which are easily implemented in an FPGA.

Now, one could ask oneself: "Can I improve those results even further than those in this paper?" The answer is yes and there are two ways to do this:

1. The most intuitive approach is via the number of states in the convolutional coder. Having a higher number of states will increase the number of possible paths and a good convolutional coder for TCM will take advantage of this to increase the minimum SED of the coder (**see Table 1** for an example of different convolution encoders which were found to be good for TCM). Ungerboeck presents a more exhaustive list of these coders [1,2]. However, the gains of this approach are not linear, and at a higher number of states the coding gains will be limited by the minimum SED of the uncoded bit.
2. The second option is a technique that we call Multidimensional Trellis Codied Modulation. The TCM we saw in this paper is called 2LD-16QAM-TCM, with L=1 (a 2D constellation with a dimensionality factor L=1). The main concept in multi-dimensionality relates toincreasing the number of symbols created in one processing period. The transmitted symbols are generated together and this co-generation creates dependence and allows for better performance. The advantages of this approach are (see [3] for an excellent paper on the subject):

    A. Fractional information rates. We can reduce the TCM code overhead by affecting more than one symbol.
    B. Better bit efficiency possible.
    C. Smaller peak to average ratio (since random coherence problems are lessened).
    D. No additional hardware complexity.

The combination of both approaches can raise the performance of TCM to a completely new level, and all of this is achievable without any additional bandwidth requirement. A successful implementation of such an approach is the V.34 (also known as V.fast) modem protocol.

In today's modern communications, TCM is mentioned in many well known standards. In the 802.11 IEEE standard, TCM is specified as a way to improve data rate. In IEEE 802.15.3, TCM is used for data coding in the physical layer. In WiMAX (IEEE 802.16 standard), the physical layer provides FEC with concatenated Reed Solomon and Pragmatic TCM with optional interleaving.

This white paper has primarily served as an introduction to the basic concepts of Trellis Coded Modulation. For additional information, the reader is referred to the references in **Annex I.**

ANNEX I

References

1. Gottfried Ungerboeck, "Trellis-Coded Modulation with Redundant Signal Sets Part I: Introduction", *IEEE Communications Magazine*, vol. 25, no. 2, February 1987, pp. 5-11

2. Gottfried Ungerboeck, "Trellis-Coded Modulation with Redundant Signal Sets Part II: State of the Art", *IEEE Communications Magazine*, vol. 25, no. 2, February 1987, pp. 12-21

3. Steven S. Pietrobon, Robert H. Deng, Alain Lafanechere, Gottfried Ungerboeck, Daniel Costello, "Trellis-Coded Multidimensional Phase Modulation", *IEEE Transactions on Information Theory*, Vol 36, no. 1, January 1990

4. T.K. Moon, "Error Correction Coding: Mathematical Methods and Algorithms", *Wiley-Interscience*, New-York, 1985

5. Mei Hong, "Analysis of the Bit Error Rate of the Trellis-Coded Modulation", *Chalmers University Of Technology*, Göteborg, Sweden, December 2002, EX043/2002

Nutaq products are constantly being improved; therefore, Nutaq reserves the right to modify the information herein at any time and without notice.

Nutaq®

INNOVATION TODAY
FOR TOMORROW®

2150 Cyrille-Duquet, Quebec City (Quebec) G1N 2G3 CANADA
T. 418-914-7484 | 1-855-914-7484 | F. 418-914-9477
info@nutaq.com