

Non-Linear Trellis Codes for Optical Applications*

Miguel Griot, Andres I. Vila Casado, Wen-Yen Weng, Herwin Chan and Richard Wesel

Dept. of Electrical Engineering
University of California
Los Angeles, CA 90095
{mgriot,avila,wenyen,herwin,wesel}@ee.ucla.edu

Abstract

This paper presents nonlinear trellis codes that produce codewords with a relatively low density of ones. These trellis codes are designed specifically for the Z-Channel that arises in a multiple-user optical channel with non-coherent combining, when the other users are treated as noise. In conjunction with interleaver-division multiple access, these trellis codes provide a relatively low complexity solution for uncoordinated access in the optical multiple-user environment. Also, a union bound technique that predicts the performance of these codes is presented. An implementation on an FPGA is described and results are shown.

1 Introduction

Optical channels provide data rates in the range of tens to hundreds of gigabits per second. Wavelength division and time division are the most common forms of multiple access for optical channels, but they require considerable coordination. Completely uncoordinated transmission is theoretically possible with the same efficiency as WDMA or TDMA if joint decoding is employed. Joint decoding can be simplified to sequential decoding if the ones densities of each transmitter are carefully controlled [1], but this level of coordination is not qualitatively different from assigning wavelengths or time slots.

All of the schemes described above are fully efficient in that one useful bit of information may be transmitted for every time-wavelength slot available. However, neither joint decoding nor successive decoding are computationally feasible at optical speeds today. Completely uncoordinated transmissions using interleaver division multiple access (IDMA) and simple decoding that treats other users as noise is a practical alternative. Surprisingly, in the multiple-user optical channel with non-coherent combining this relatively low-complexity approach can theoretically achieve about 70% of full efficiency.

This paper presents an uncoordinated optical multiple access system employing IDMA in which the other users are treated as noise. To allow decoding at optical speeds in the near future, this paper investigates trellis codes which operate in a range of 30% of full efficiency. We are exploring turbo and LDPC solutions, which will approach the 70% limit with a complexity that will eventually be practical, in other work.

Section 2 reviews uncoordinated multiple access in the non-coherently combining channel. Section 3 presents the design of nonlinear trellis coded modulation (NL-TCM) for this application. Section 4 presents a transfer function bound for NL-TCMs operating on the Z-Channel. Section 5 presents performance results, and Section 6 concludes the paper.

2 Uncoordinated Multiple Access in a Non-Coherently Combining Channel

A simple communications model that can describe the multiple-user optical channel with non-coherent combining is the OR channel. In this channel, if all users transmit a zero, then the channel output is a zero. However, if even one user transmits a one, then the channel output is a one. The information-theoretic capacity region of this channel is the triangle of all rate pairs with a sum-rate (the sum of all the rates of the system) less than the maximum possible sum-rate, which is 1. As discussed above, this capacity may be achieved with time-sharing, wavelength-sharing, joint decoding of all the transmitted sequences, or sequential decoding if transmitted ones densities are carefully controlled. All of these solutions require either coordination of all users or a very complex decoder that is not currently feasible at optical speeds.

With joint decoding and successive decoding unavailable for complexity reasons, the other users must be treated as noise. This transforms the OR channel into the Z-Channel shown in Fig. 1. Because we assume that all users have the same transmitted ones density, the transition probability is

*This research is supported by the U.S. Navy through contract N66001-2-1-8398

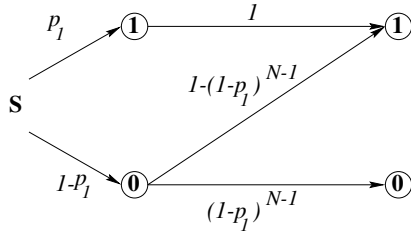


Figure 1: Z-channel resulting from the OR-MAC channel when other users are treated as noise.

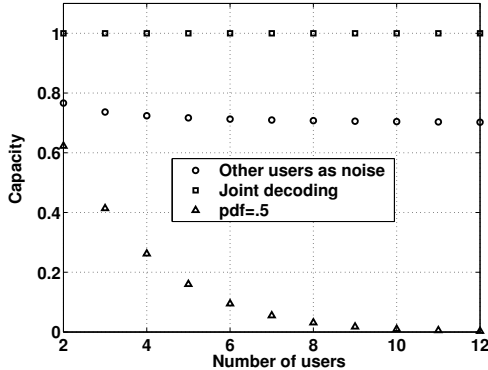


Figure 2: Maximum theoretical sum-rates as a function of number of users for three approaches to the OR multiple access channel.

a function of the same transmitted ones density employed by the desired user. The uncoordinated multiple access model does not strictly require that all users employ the same ones density, but we maintain that assumption throughout this paper for simplicity.

Fig. 2 shows the maximum theoretical sum-rate when treating the other users as noise as a function of the number of users (assuming each user employs the same ones density). As illustrated in the figure, this maximum theoretical sum-rate decreases to $\ln 2 = 0.6931$ as the number of users increases. This is a relatively small loss in rate for the substantial reduction in complexity. Fig. 2 also shows the maximum theoretical sum-rate for joint decoding (It's always 1.) and the maximum theoretical sum-rate when treating the other users as noise and maintaining the traditional ones density of $p_1 = 0.5$ (This sum-rate quickly converges to zero). It is not explicit in Fig. 2, but as the number of users increases, the ones density of each individual user decreases for the two curves where it is not fixed to be 0.5. As we can see from the poor performance of the $p_1 = 0.5$ curve, codes with lower ones densities are a requirement for this application.

One successful approach for uncoordinated multiple-access is Interleaver-Division Multiple-Access (IDMA) [2], [3]. With IDMA, every user

has the same channel code, but each user's code bits are permuted using a (hopefully) unique randomly drawn interleaver. The receiver is assumed to know the interleaver of the desired user. With IDMA in the OR multiple access channel (MAC), a receiver should see the desired signal corrupted by a memoryless Z-channel. We performed simulations comparing an NL-TCM code under two channels: a 6 user OR-MAC channel using IDMA and the equivalent Z-channel that the receiver would see if the errors were not generated by codewords but by random errors. The performance was the same. Thus, in the context of IDMA, the remaining challenge is the design of a good code with the desired ones density.

3 NL-TCM with Controlled Ones Density

Papers appearing since the 1950's have addressed the problem of designing codes with $p_1 = 0.5$ for the Z-channel. See [4] for a unified account on such codes and [5] for the most recent advances in this field. Only recently there has been work on LDPC codes with an arbitrary density of ones, see [6] and [7]. This manuscript is the first to our knowledge to address the design of trellis codes with an arbitrary density of ones for the Z-Channel.

In this section, we present a design technique for trellis codes for the Z-channel with an arbitrary density p_1 of ones. Our goal is to maximize the minimum directional Hamming distance (a metric we'll define below) between codewords, and the rates considered will be of the form $1/N$. We use a conventional feed-forward encoder in order to determine the branches of the trellis, but instead of using generator polynomials to compute the output of each branch, we will directly assign the output value.

3.1 Directional Hamming Distance

In the Z-Channel, a transmitted 1 will always induce a received 1. Thus, to make a decoding error, the decoder must see ones in all the bit positions where the incorrect codeword has ones. Let us define the directional Hamming distance $d_D(c_1, c_2)$ the number of ones that we have to add to c_1 so that all ones of codeword c_2 are ones in the received word.

In the Z-Channel with a probability of 0-to-1 transition less than 0.5, if two codewords have different Hamming weights, the codeword with the smaller Hamming weight will never be incorrectly decoded by a Maximum Likelihood (ML) decoder when the code with the larger Hamming weight is transmitted. Suppose a codeword c_1 has a larger Hamming weight than another codeword c_2 .

Whenever the received sequence of bits r contains all the ones from both codewords, the Hamming distance $d_H(r, c_1) < d_H(r, c_2)$ since $d_D(c_1, r) < d_D(c_2, r)$. Thus c_1 is more likely than c_2 . In this example, we say that the *operational* distance from c_1 to c_2 is ∞ , since this error cannot happen. When considering the pairwise behavior of these two codewords, the only distance that matters is $d_D(c_2, c_1)$. Note that in general, the directional distance that matters in such cases is the larger of the two.

On the other hand, if both codewords have the same Hamming weight, the directional Hamming distances are equal and errors can be made in either direction. In fact, whenever the received sequence of bits r contains all the ones from both codewords an ML decoder is faced with a tie between these two codewords. Whether the two codewords have the same Hamming weight or not, the relevant metric is the maximum of the two directional Hamming distances. Thus we define our pairwise design metric to be the maximum pairwise directional Hamming distance:

$$d(c_i, c_j) = d(c_j, c_i) = \max[d_D(c_i, c_j), d_D(c_j, c_i)] \quad (1)$$

and the overall design objective is to maximize:

$$\min_{(i,j=1\dots M:j \neq i)} \{ \max[d_D(c_i, c_j), d_D(c_j, c_i)] \} \quad (2)$$

where M is the number of codewords. This metric for the Z-Channel is well known, appearing in [4] and [5] among other papers.

3.2 Greedy definition of distance

Due to its non-linearity, this definition of distance cannot be applied branch-wise. A codeword c_1 that has more Hamming weight than a codeword c_2 can include branches with less Hamming weight than the corresponding branches of c_2 , and vice versa. Since it is impossible to tell from an individual branch which codeword will end up having more Hamming weight, we will use a ‘greedy’ definition of distance for our trellis code design. In order to maximize the minimum distance as in (2) we have to consider both $d_D(c_1, c_2)$ and $d_D(c_2, c_1)$. Namely, the safest branch-wise metric would be

$$d_{i,j} = \min[d_D(c_i, c_j), d_D(c_j, c_i)], \quad (3)$$

which is the branch-wise metric that will be maximized in our design.

With this branch-wise metric, codewords with equal Hamming weights produce larger values than codewords with different Hamming weights, so we will assign output values to the trellis branches with as similar Hamming weight as possible, preferably equal. Even with the standard metric of (2), better performance results are achieved with as similar

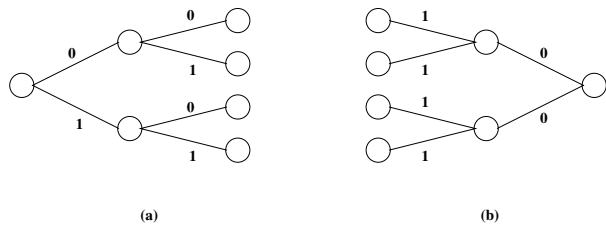


Figure 3: (a) Four paths that start on the same state in two trellis sections. (b) Four paths that arrive to the same state in two trellis sections.

Hamming weight codewords as possible. Very different Hamming weights seem good when looking at the pairwise difference of a low-weight codeword and high-weight codeword, but the worst-case performance will be driven by the pairwise difference of two low-weight codewords.

3.3 NL-TCM Code Design

As we mentioned before, the code design consists on assigning output values to the branches of the trellis code. Those outputs have to satisfy that the average density of ones is the desired density p_1 . Our goal is to maximize the minimum distance d_{min} using the greedy pairwise metric introduced in section (3.2).

A first idea is to apply Ungerboeck’s rules [8] in the context of our pairwise metric. That is, to maximize the distance between branches splitting from a state (splits) and branches merging to a same state (merges). We can extend Ungerboeck’s rules more deeply into the trellis. For instance, if we could choose all branches to have distance of at least 1 between them we would always increase the distance of different paths by at least 1 in each step. Moreover, we can try to maximize the distance not only of the splits, but also the distance between the four branches coming from a split in the previous step (see Fig. 3), and the eight branches that come from a split two steps before, and so on. And we can move backwards from a merge and try to maximize the distance between the previous four branches, and so on (see Fig. 3). We will explain all this in more detail in the following sub-sections.

3.3.1 Choosing the Hamming weight of the branches

The first step in the design is to assign Hamming weights to each of the branches. In order to do that, we will consider the subgraph shown in Fig. 4. That figure shows the subgraph that composes every trellis diagram that results from a conventional feed-forward encoder with one input. The branches produced by an input bit equal to 0 for

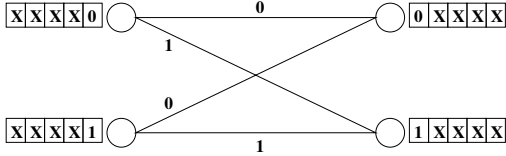


Figure 4: Basic sub-graph of the trellis diagram

both states go to the same state, and the same happens with an input bit of 1. Thus, we have in the same subgraph all the splits and merges produced by those four branches. If we have 2^v states, we will have 2^{v-1} subgraphs of this type. We will focus on these subgraphs to assign the Hamming weights in the following way. Suppose we want a density of ones p_1 , and rate $1/N$. Define

$$s = \text{floor}(p_1 \cdot N), i = \text{round}(4 \cdot (p_1 \cdot N - s)) \quad (4)$$

Then, assign a Hamming weight of $W_H = s + 1$ to i of the 4 branches, and $W_H = s$ to $4 - i$ of the branches. This way we are getting as close as possible to the desired density p_1 . For example, for $N = 18$ and $p_1 = 1/8$, we will have 3 branches with $W_H = 2$ and 1 branch with $W_H = 3$. For $N = 17$ and $p_1 = 1/8$, all 4 branches will have $W_H = 2$.

3.3.2 Choosing all branches to have distance of at least 1 between each other

If possible, it would be desirable to have all branches have distance of at least 1 between each other.

In the case that all branches have the same weight $W_H = s$, we can have up to $\binom{N}{s}$ branches with distance 1 between each other. For example, for $N = 17$ and $p_1 = 1/8$, all branches have $W_H = 2$. In that case we can have $\binom{17}{2} = 134$ different branches, so if we wanted to have all different branches, we could have a trellis with 64 states (represented by 6 bits) and 128 branches. In case we wanted to use a 128-states trellis (7 bits, 256 branches), we would have to repeat output values. If there are branches with different Hamming weights, the computation is a little more complicated.

It is clear that by doing this, in a trellis that has 2^v states, d_{min} will be at least $v + 1$, since the minimum number of branches that two paths can differ is $v + 1$ and each step adds at least 1 in the distance.

In case we need to repeat output values, the branches to be chosen to have equal output values are the following. If we look again to the subgraph in Fig. 4, there are two subgroups of 2 branches that neither merge nor split with each other. Those can be allowed to have equal output value and they will not affect d_{min} . The reason for that is that

paths that split and merge at some point and traverse different branches of one of these subgroups at some point of the path, will have a least $2v + 1$ branches from the split to the merge, and at least $2v$ of those will be different.

3.3.3 Ungerboeck's rule

Once we chose the weights of the branches, and the possible output values that will make all branches (or almost all, allowing some to be equal as explained in the previous section), we have to assign output values to branches. Up to this point, we have a d_{min} of at least $v + 1$, but we can further increase it by applying Ungerboeck's idea of maximizing the distance between splits and merges. Remember that the Hamming weights of the branches are all s , or both s and $s + 1$. We can always guarantee that all splits have a distance of at least s between each other, and also all merges. Thus, $d_{min} \geq 2s + v - 1$.

3.3.4 Extending Ungerboeck's rule into the trellis

We can extend Ungerboeck's rule more deeply into the trellis, and maximize not only the distance between splits, and the distance between merges, but the distance between the 4 branches coming from a split a trellis section before in the trellis, or the 8 branches coming from a split 2 sections before, and so on. We can do the same with the merges moving backwards in the trellis. Notice that by maximizing the distance between the 8 branches coming from a split two sections before, we are also maximizing the distance between all 4 branches coming from a split a trellis section before, and all splits. The same idea applies to the merges. If we move h sections from a split, and g sections from a merge (the split and the merge are each counted as a trellis section), our new bound for the minimum distance is $d_{min} \geq s \cdot (h + g) + v - (h + g)$.

The number of trellis sections we can move from a split and backwards from a merge is given by the maximum number of output values that can have maximum distance between each other. Let us explain this in more detail.

First, let us compute the number of branches that need to have maximum distance between each other if we try to move h sections from a split and g sections backwards from a merge. From the splitting point of view, we will have groups of 2^h branches that need to have maximum distance between each other. From the merging point of view, we will have groups of 2^g . Each branch belongs to one group of 2^h and one group of 2^g , so each branch has to have maximum distance with $2^h + 2^g - 2$ other branches.

Second, we can compute how many branches of maximum distance between each other we can have. Using the example where $N = 17$ and all the branches have Hamming weight $s = 2$, we can have $\text{floor}(\frac{N}{s}) = 8$ branches with maximum distance between each other. Let us denote this number as T .

The constraints we have then are (1) $2^h \leq T$, (2) $2^g \leq T$ and (3) each branch has to belong to one group of 2^h and one group of 2^g . If we choose h and g such that $2^h + 2^g - 2 \leq T$, all 3 constraints can be satisfied.

3.3.5 Designing for very low target density of ones

Actually, for a low enough target density of ones (a large enough number of users in the MAC case), all the branches can be chosen to have maximum distance between each other (we can choose them so that for any particular position of the output, there is at most 1 branch that has a 1 in that position), independently of the rate we want to achieve. Suppose we would like to design a NL-TCM code with S states and a target density of ones p_1 . Then, we have $2S$ different branches. Denote M the sum of all the ones from the outputs of all the $2S$ branches, for the desired p_1 . Then, supposing we can exactly achieve the desired p_1 :

$$M = 2S \cdot p_1 \cdot N$$

But if $p_1 \leq \frac{1}{2S}$, then $M \leq N$, so it is possible to choose all $2S$ branches so that for any particular position of the output, there is at most 1 branch that has a 1 in that position. For such low densities of ones, the design becomes straightforward. Compute the Hamming weight of each branch as explained in section 3.3.1, and for each branch, add ones in positions that hadn't been used in previous branches.

4 Transfer Function Bound for NL-TCM Codes

Ellingsen [9] provided a combinatorial expression for an upper bound on the BER of linear block codes over the Z-channel under ML decoding. For convolutional codes assuming binary PAM or QPSK, Viterbi [10] introduced the analytical technique using generating functions (also called transfer functions or enumerating functions) to provide a union bound on the BER. The technique is based on a 2^{ν_x} -state diagram for the convolutional encoder. In the case of general trellis codes where high level constellation introduce nonlinearity, Biglieri [11][12] generalized Viterbi's algorithm

by using the product state diagram with 2^{ν_x} -states. Biglieri's algorithm can be applied to nonlinear trellis codes over the Z-channel with modifications on the pairwise error probability measure.

Note that because of the asymmetric nature of the Z-channel, if we consider the whole binary sequence transmitted, X^n , received sequence, Y^n , and any other valid codeword, \hat{X}^n . The error probability of decoding X^n into \hat{X}^n under ML decoding is

$$P_e(X^n \rightarrow \hat{X}^n) = \begin{cases} \frac{1}{2} \cdot \alpha^{d_D(X^n, \hat{X}^n)} & , W_H(X^n) = W_H(\hat{X}^n) \\ \alpha^{d_D(X^n, \hat{X}^n)} & , W_H(X^n) < W_H(\hat{X}^n) \\ 0 & , W_H(X^n) > W_H(\hat{X}^n) \end{cases}$$

where $d_D(\cdot, \cdot)$ denotes the directional distance. If we consider the sequence pair (X^n, \hat{X}^n) , the error probability of transmitting one sequence and decoding into the other is

$$\begin{aligned} P_e(X^n \rightarrow \hat{X}^n) + P_e(\hat{X}^n \rightarrow X^n) \\ = \alpha^{\max(d_D(X^n, \hat{X}^n), d_D(\hat{X}^n, X^n))} \\ \leq \frac{1}{2}[\alpha^{d_D(X^n, \hat{X}^n)} + \alpha^{d_D(\hat{X}^n, X^n)}] \end{aligned} \quad (5)$$

Therefore, if $P_e(X^n \rightarrow \hat{X}^n)$ is replaced (not always upper-bounded) by $\frac{1}{2}\alpha^d(X^n, \hat{X}^n)$ for all the codewords X^n and \hat{X}^n , the transfer function bound technique can be readily applied to the NL-TCM to yield a valid upper bound because of the additive property of the directional distance.

As in [11], the product state diagram consists of state pairs, (s_e, s_r) , where s_e is the encoder state and s_r the receiver state. Following Biglieri's notation, the product states can be divided into two sets, the good states denoted by S_G and the bad states denoted by S_B defined as

$$S_G = \{(s_e, s_r) \mid s_e = s_r\}, \quad S_B = \{(s_e, s_r) \mid s_e \neq s_r\} \quad (6)$$

By suitably renumbering the product states, we get the transition matrix

$$S(W, I) = \left[\begin{array}{c|c} S_{GG}(W, I) & S_{GB}(W, I) \\ \hline S_{BG}(W, I) & S_{BB}(W, I) \end{array} \right] \quad (7)$$

Where the $N \times N$ matrix $S_{GG}(W, I)$ accounts for the transitions between good product states, the $N \times (N^2 - N)$ matrix $S_{GB}(W, I)$ accounts for the transition from good product states to bad product states, and so forth. N is the number of encoder states 2^{ν_x} . For each transition in the product state diagram, $S_1 \rightarrow S_2$, the branch is labelled by

$$p(S_1 \rightarrow S_2) W^{d_D(x_e, x_r)} I^{d_H(u_e, u_r)} \quad (8)$$

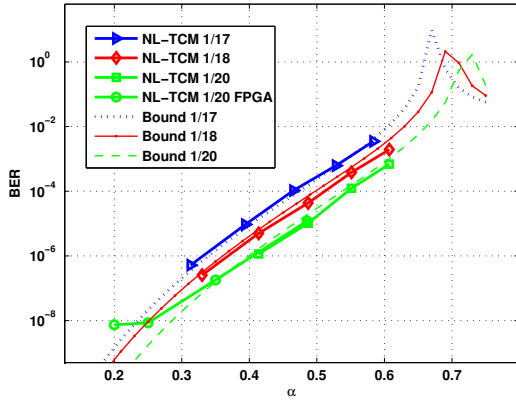


Figure 5: BER of NL-TCM codes for different values of crossover probabilities (α)

where u_e and x_e denote the input and output word for the encoder states respectively. Similar for the received states. $d_H(\cdot, \cdot)$ denotes the Hamming distance. Then the transfer function $T(W, I)$ becomes

$$T(W, I) = p_s \{ S_{GG} + S_{GB}(I - S_{BB})^{-1} S_{BG} \} \mathbf{1} \quad (9)$$

where $p_s = [\frac{1}{N} \frac{1}{N} \dots \frac{1}{N}]$ is the probability distribution of the encoder states and $\mathbf{1} = [11 \dots 1]^T$. The BER bound is computed as

$$BER \leq \frac{1}{2} \cdot \frac{1}{k} \cdot \frac{\partial T(W, I)}{\partial I} \Big|_{W=\alpha, I=1} \quad (10)$$

5 Performance Results

We have tested the NL-TCM performance in an uncoordinated OR multiple access channel, where every user treats the others as noise, and all users transmit with the same density of ones p_1 , as explained in section 2.

5.1 NL-TCM for 6-user OR-MAC

Fig. 5 shows the BER of various 64-state NL-TCM codes designed to work in a 6 user OR-MAC channel, along with their theoretical transfer function bounds. The densities of ones are close to 0.108, the density needed to achieve capacity for a 6-users OR-MAC when treating other users as noise. Simulations have been programmed in C for rate-1/17 NL-TCM code with $p_1 = 2/17$, rate-1/18 NL-TCM code with $p_1 = 1/8$ and rate-1/20 NL-TCM code with $p_1 = 1/8$.

It can be observed that that the transfer function bounds are tight for the proposed rate-1/17 NL-TCM code with $p_1 = 2/17$ and rate-1/18 NL-TCM code with $p_1 = 1/8$. Although the transfer function bound is an upper bound based on the expectation, it is fine for the simulation to be slightly above the

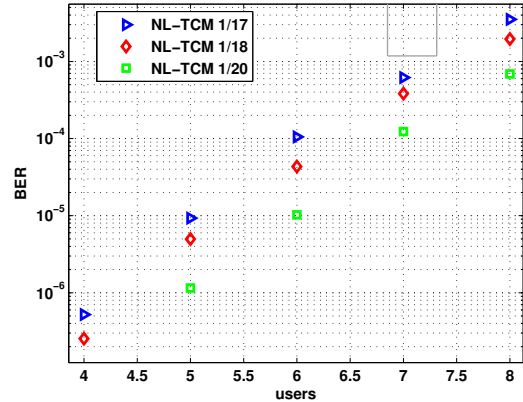


Figure 6: BER of NL-TCM codes vs. number of users

bound if the margin is within the usual variation of a simulation about the mean.

Also, in order to prove that NL-TCM codes are feasible today for optical speeds, a hardware simulation engine was built on the Xilinx Virtex2-Pro 2V20 FPGA. The simulator implemented on this device had equivalent gate count of 360K gates and is able to simulate the rate 1/20 code at 70Mbps.

Simulation results for rate-1/20 NL-TCM code obtained in the FPGA testbed, are also shown in Fig.5. Due to design constraints, the hardware Viterbi decoder has implementation differences comparing to the software simulation. The two main differences are the traceback depth and the maximum path metric. Both of these values are set to infinity for software simulation. The hardware implementation has a traceback depth of 35 and a path distance metric of 20. These differences cause the deviation from the theoretical bound at low bit error rates. Future versions of the simulator will address these issues to provide more accurate results.

Finally, Fig. 6 shows the BER of these codes in terms of the number of users present in an OR-MAC.

5.2 NL-TCM for 100-user OR-MAC

As explained in section 3.3.5, the design of NL-TCM codes becomes straight-forward for low enough density of ones. As an example, we have designed NL-TCM codes for the 100-user OR-MAC case. In that case, the optimal $p_1 \simeq 0.00691$. We designed codes for rates 1/400 and 1/360. Results are shown in Table 5.2.

5.3 Concatenation of NL-TCM code with a Block Code

Optical systems typically deliver a very low BER. In order to maintain this BER the rate of the NL-

Table 1: BER of NL-TCM for 100-user OR-MAC

Rate	Sum-rate	p_1	α	BER
1/360	0.2778	0.006944	0.49837	$4.54e^{-6}$
1/400	0.25	0.006875	0.49489	$9.45e^{-7}$

Table 2: BER of RS+NL-TCM for 6-user OR-MAC

Rate	Sum-rate	p_1	α	BER
0.0484	0.29	0.125	0.4652	$2.48e^{-10}$

TCM channel code would have to be very low. A better solution is found taking into account the distribution of the erred bits in a transmitted stream after the NL-TCM decoding. It is well known that when the Viterbi decoding fails only a few number of bits are in error. Thus, a high rate block code that can correct few errors can be attached as an outer code, dramatically lowering the BER. In this work, a Reed-Solomon (RS) code has been chosen to be implemented, because we found it to be the most suitable in our FPGA implementation. Another good reason for using an RS code, is the fact that it corrects bursts of up to 8 bits, and errors produced by the NL-TCM are bursty, due to path errors.

A concatenation of the rate-1/20 NL-TCM code with a (255 bytes,247 bytes) Reed-Solomon code has been tested for the 6-user OR-MAC scenario. The rate of this code is $(247/255) \cdot (1/20) \simeq 0.0484$. Results are shown in Table 5.3. In this example, we are achieving almost 30% of full capacity, with a suitable BER for optical systems (in the order of 10^{-10}), and a feasible complexity for today's technology at the required optical speeds.

Simulations have been ran in software, a hardware implementation of this concatenation is being finalized at this moment, and its results will also be presented in the camera-ready version.

6 Conclusions

This paper addressed the problem of designing codes for the optical-MAC channel. These codes are required to have an arbitrary density of ones. We also required them to have low enough complexity to be computationally feasible at optical speeds today. NL-TCM codes satisfy both requirements. A design criteria for NL-TCM codes was introduced, and bounds on their performance were computed. Furthermore, by concatenating these codes with high-rate block codes we can achieve a good part of the capacity of the channel with a very low BER and a very fast decoder. Also an IDMA solution that allows uncoordinated multiple access

was presented and tested.

References

- [1] A. Grant, B. Rimoldi, R. Urbanke, and P. Whiting. Rate-Splitting Multiple Access for Discrete Memoryless Channels. *IEEE Trans. on Inform. Theory*, 47(3):873–890, Mar. 2001.
- [2] Li Ping, K. Y. Wu, and L. Liu. A Simple, Unified Approach to Nearly Optimal Multiuser Detection and Space-time Coding. In *ITW 2002*, India, October 2002.
- [3] Li Ping, Lihai Liu, and W. K. Leung. A Simple Approach to Near-Optimal Multiuser Detection: Interleave-Division Multiple-Access. In *IEEE Wireless Comm. and Networking Conference*, pages 391–396, 2003.
- [4] T. Kløve. Error correcting codes for the asymmetric channel. In *Dept. Mathematics, Univ. Bergen, Bergen, Norway, Tech. Rep.1809-0781*, 1995.
- [5] Fang-Wei Fu, San Ling, and Chaoping Xing. New Lower Bounds and Constructions for Binary Codes Correcting Asymmetric Errors. In *IEEE Trans. on Inform. Theory*, volume 49, pages 3294–3299, March 2003.
- [6] Amir Bennatan and David Burshtein. On the Application of LDPC Codes to Arbitrary Discrete-Memoryless Channels. In *IEEE Trans. on Inform. Theory*, volume 50, pages 417–438, March 2004.
- [7] Edward A. Ratzer and David J.C. MacKay. Sparse Low-Density Parity-Check Codes for Channels with Cross-Talk. In *Proc. ITW 2003*, Paris, France, April 2003.
- [8] G. Ungerboeck. Channel Coding with Multi-level Phase Signals. In *IEEE Trans. on Inform. Theory*, volume 28, pages 52–67, 1982.
- [9] P. Ellingsen, S. Spinsante, and O. Ytrehus. Maximum Likelihood Decoding of Codes on the Asymmetric Z-channel. In *the 10th IMA International Conference on Cryptography and Coding 2005*.
- [10] A. J. Viterbi. Convolutional codes and their performance in communication systems. *IEEE Trans. on Comm.*, 19, Oct 1971.
- [11] E. Biglieri. volume 32, May 1984.
- [12] Y. J. Liu, I. Oka, and E. Biglieri. Error probability for digital transmission over non-linear channels with application to tcm. volume 36, Sep 1990.