LOW-COMPLEXITY HIGH-SPEED 4-D TCM DECODER

Jinjin He, Zhongfeng Wang*, Senior Member, IEEE, and* Huaping Liu
School of EECS, Oregon State University
Corvallis, OR, USA
Email: {hej, wangz, liuhua} @onid.oregonstate.edu

ABSTRACT

This paper presents a low-complexity, high-speed 4-dimensional 8-ary Phase Shift Keying Trellis Coded Modulation (4-D 8PSK TCM) decoder. In the design, an efficient architecture for the transition metrics unit (TMU) is proposed to significantly reduce the computation complexity without degrading the performance. In addition, pipelining and parallel processing techniques are exploited to increase the decoding throughput. Synthesis results show that the FPGA implementation of the TCM decoder can achieve a maximum throughput of 1.062 Gbps.

Index Terms— VLSI, FPGA, Trellis coded modulation

1. INTRODUCTION

Trellis Coded Modulation (TCM) is a technique to combine error-correcting coding and modulation used in digital communications [1]. TCM gains noise immunity over uncoded transmission without expanding the signal bandwidth or increasing the transmitted power. By partitioning signal set into groups, TCM uses signal mapping to combine error correction coding with modulation [2], [3]. Usually, 2*m*+1 constellation points will be used for transmitting *m* bits in 1 dimensional TCM systems. TCM schemes using multi-dimensional constellations have been shown to further improve the error performance. In [4], a systematic approach to partitioning multi-dimensional signals is proposed. More recently, iterative Error Correction Codes (ECC) such as Turbo codes [5] and Low Density Parity-Check (LDPC) codes [6] have become popular due to their near-Shannon-limit decoding performance. However, TCM still has a significant advantage of low encoding/decoding latency over those iterative codes. Nowadays, 4-D 8PSK TCM is employed by the Management Council of the Consultative Committee for Space Data System (CCSDS) for high data rate transmission [7]. The convolutional encoder and constellation mapper for the 4-D 8PSK TCM system are defined in [7]:

- 64 trellis states;
- ¾ rate of convolutional coder;
- rate of modulation: Rm=*m*/(*m*+1), *m* =8, 9 , 10 or 11.

The corresponding convolutional encoder is shown in Fig. 1.

Reducing the complexity while maintaining high speed and good performance is a challenging issue for multi-dimensional TCM decoder. However, until very recently, only a few existing articles have discussed the implementation aspects of TCM decoder [8-10], and implementation of multi-dimensional TCM encoders has received even less attention. In [11], a general architecture for 4-D 8PSK TCM decoders is presented. The detail of FPGA implementation for Rm = 8/9 case is discussed in [11], where it was shown that their decoder can work at a data rate up to 460 Mbps for Rm = 8/9 case.

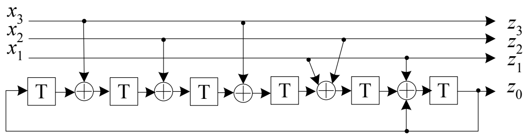

Fig. 1: Convolutional encoder for 4-D 8PSK TCM system.

In this paper, an efficient architecture for Transition Metrics Unit (TMU) is proposed and the high-speed Viterbi Decoder (VD) is designed which leads to a much higher throughput than the work presented in [11]. The rest of the paper is organized as follows. General information of the 4-D 8PSK TCM system will be introduced in Section 2. In Section 3, the low-complexity high-speed TMU is discussed. Section 4 focuses on the design of a high speed VD. Section 5 discusses the FPGA implementation, followed by conclusions in Section 6.

2. 4-D 8PSK TCM SYSTEM OVERVIEW

For clarify, we discuss the proposed scheme using the example of Rm=11/12. Construction of optimal TCM codes is beyond the scope of this paper, and the related details can be found in [4]. From VLSI implementation perspective, a TCM encoder consists of four main parts: a differential encoder, a binary convolutional encoder, a multidimensional mapper and the 8PSK modulator, as shown in Fig. 2. The mapping equation is given by Eq. (1), where Z_i, i = 0, 1, 2 and 3 correspond to 4 sets of 8PSK symbols and $x^{(k)}$, k = 0, 1, 2 …, 11, represent 12 binary input bits [10].

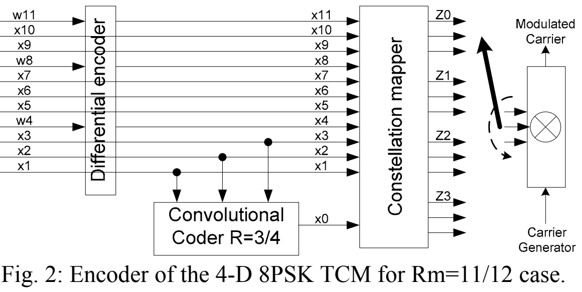

Fig. 2: Encoder of the 4-D 8PSK TCM for Rm=11/12 case.

This project is supported by NASA under grant NNX07AF84G.

$$\begin{pmatrix} Z_0 \\ Z_1 \\ Z_2 \\ Z_3 \end{pmatrix} = \left\{ \left(4x^{(11)} + 2x^{(8)} + x^{(4)}\right) \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} + 4 \begin{pmatrix} 0 \\ x^{(10)} \\ x^{(9)} \\ x^{(10)} + x^{(9)} + x^{(7)} \end{pmatrix} + 2 \begin{pmatrix} 0 \\ x^{(6)} \\ x^{(5)} \\ x^{(6)} + x^{(5)} + x^{(3)} \end{pmatrix} + \begin{pmatrix} 0 \\ x^{(2)} \\ x^{(1)} \\ x^{(2)} + x^{(1)} + x^{(0)} \end{pmatrix} \right\} (\text{mod } 8) \quad (1)$$

The TCM decoder includes a soft decision VD, which can provide optimal decoding performance for convolutional codes, a de-mapper, and a differential decoder. The main difference between the trellis diagrams of a TCM system and the corresponding single convolutional encoding system is that in TCM system there are parallel paths in state transmissions. Therefore, before performing Viterbi decoding, a TMU is used to find the optimal paths among each group of parallel paths as the Branch Metric (BM). In a multi-dimensional TCM decoder, the TMU could be very complex. The complete 4-D 8PSK TCM decoder diagram is shown in Fig. 3.

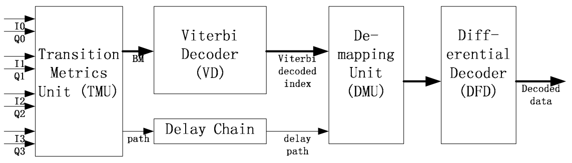

Fig. 3: 4-D 8PSK TCM decoder diagram.

In Fig. 3, the delay chain is used to buffer the paths associated to each BM generated by TMU. The "path" refers to the 12 bits of information from 4 dimensions of 8PSK signals. Since there are only 16 BMs [11] in this case, using a 4-bit index instead of 12-bit path information inside the VD is more efficient. When the VD outputs the decoded index, the corresponding path will be sent to the DMU for further processing.

3. LOW-COMPLEXITY TMU DESIGN

There are two typical options for the design of TMU: 1) ROM based approach, 2) arithmetic (on-line) computation based solution. In order to reduce hardware as well as to pipeline the computation unit for high clock speed, the second approach is adopted in our design. Based on the auxiliary trellis [11], two efficient methods are developed to reduce the computation complexity.

3.1. Simplified computation o f Euclidian metrics

Theoretically, the BM for a soft-decision Viterbi decoder in multi-dimensional TCM is represented as the sum of the Euclidian distance from each dimension of the signal sets. Given a received 8PSK symbol (I_r, Q_r) and a target constellation point A: (I_s, Q_s) (s = 0, 1, 2, …, 7 for 8 PSK signals), the Euclidian distance is computed as:

$$d_s = (I_r - I_s)^2 + (Q_r - Q_s)^2$$
$$= (I_r^2 + I_s^2 + Q_r^2 + Q_s^2) - 2\, I_r I_s - 2\, Q_r Q_s. \quad (2)$$

Note that for all d_s, ($I_r^2 + I_s^2 + Q_r^2 + Q_s^2$) remains the same. Finding the min (d_0, d_1, d_2 …, d_7) (the closest constellation point to the received signal) is equivalent to finding the max ($I_rI_0 + Q_rQ_0$, $I_rI_1 + Q_rQ_1$, $I_rI_2 + Q_rQ_2$, …, $I_rI_7 + Q_rQ_7$). Thus, we only need to consider the following distance:

$$d'_s = I_rI_s + Q_rQ_s \quad (3)$$

Furthermore, it has been found in [11] that since all the possible constellation points are equally spaced on a circle with the phase of $s\pi/4$, s = 0,1,2…7, I_s and Q_s satisfy the following conditions:

$$I_s = - I_{(s+4) \bmod 8}$$
$$Q_s = - Q_{(s+4) \bmod 8}$$

Hence, only 4 sets of Eq. (3) need to be calculated, which is simplified as Eq. (4). From equation 4, it is clear that only 2 multiplications are required, where C_i (i=0, 1, 2, 3) denotes the Euclidean Metric given as

$$\begin{aligned} C_0 &= |\, d'_0 \,| = |\, I_r \,|; \\ C_1 &= |\, d'_1 \,| = |\, (I_r + Q_r)\times 0.707 \,|; \\ C_2 &= |\, d'_2 \,| = |\, Q_r \,|; \\ C_3 &= |\, d'_3 \,| = |\, (Q_r - I_r)\times 0.707 \,|. \end{aligned} \quad (4)$$

3.2. Computation sharing of branch metrics

In the 4-D 8PSK TCM decoder, 16 BMs are selected from 256 candidates every cycle. According to Fig. 1, the 16 BMs are related to the 16 combinations of x3, x2, x1 and x0. Let us use BM 0000 to denote the BM related to x3, x2, x1, x0=0, 0, 0, 0; BM 0001 to denote the BM related to x3, x2, x1, x0 =0, 0, 0, 1, …, and so on. Each candidate is the sum of the 4 Euclidian metrics of the received signal set (Z_0, Z_1, Z_2, Z_3). A straightforward implementation requires 3 serial addition stages to compute the candidates and several comparison steps to find the BMs. Since the results at each addition stage can be shared, using auxiliary trellis [11] can eliminate duplicate addition operations.

Fig. 4 shows an example that illustrates the process of computing BM 0000. From the left to the right, C_i (i=0, 1, 2, 3) in each step denotes the Euclidean Metrics of the received symbol from Z_0, Z_1, Z_2 and Z_3, respectively. The same rule applies to all equations and figures in the rest part of the paper. The 16 candidates for BM 0000 are $C_0+C_0+C_0+C_0$, $C_1+C_1+C_1+C_1$, $C_0+C_0+C_2+C_2$, $C_1+C_1+C_3+C_3$, $C_0+C_2+C_0+C_2$, $C_1+C_3+C_1+C_3$, $C_0+C_2+C_2+C_0$, $C_1+C_3+C_3+C_1$, $C_2+C_2+C_2+C_2$, $C_3+C_3+C_3+C_3$, $C_2+C_2+C_0+C_0$, $C_3+C_3+C_1+C_1$, $C_2+C_0+C_2+C_0$, $C_3+C_1+C_3+C_1$, $C_2+C_0+C_0+C_2$, $C_3+C_1+C_1+C_3$.

By using the auxiliary trellis, the total number of additions is reduced from 768 (*i.e.*, 3×16×16) to 336 (*i.e.*, $4^2+4^3+4^4$) compared with the straightforward implementation. However, the number of comparisons remains the same. We propose a new approach that further reduces the number of addition operations by half. In addition, we can save about 2/3 of the comparison operations.

The key ideas of the improvement are as follows. Let us again take BM 0000 as an example. In the auxiliary trellis approach as shown in Fig. 4, to compute each candidate branch metric, 3 serial addition stages are performed to compute 16 candidate branch metrics in parallel. Then, one survival branch metric is chosen from the 16 candidate branch metrics. By a careful observation of the added Euclidian distances, we found that in the last addition stage, there are only 4 different values to be added onto the 16 candidates as listed in Table I.

TABLE I. GROUPING OF THE CANDIDATES FOR BM 0000

	Z_0	Z_1	Z_2	Z_3		Z_0	Z_1	Z_2	Z_3
G1	C_0	C_0	C_0	C_0	G2	C_1	C_1	C_1	C_1
	C_0	C_2	C_2			C_1	C_3	C_3	
	C_2	C_2	C_0			C_3	C_3	C_1	
	C_2	C_0	C_2			C_3	C_1	C_3	
G3	C_0	C_0	C_2	C_2	G4	C_1	C_1	C_3	C_3
	C_0	C_2	C_0			C_1	C_3	C_1	
	C_2	C_2	C_2			C_3	C_3	C_3	
	C_2	C_0	C_0			C_3	C_1	C_1	

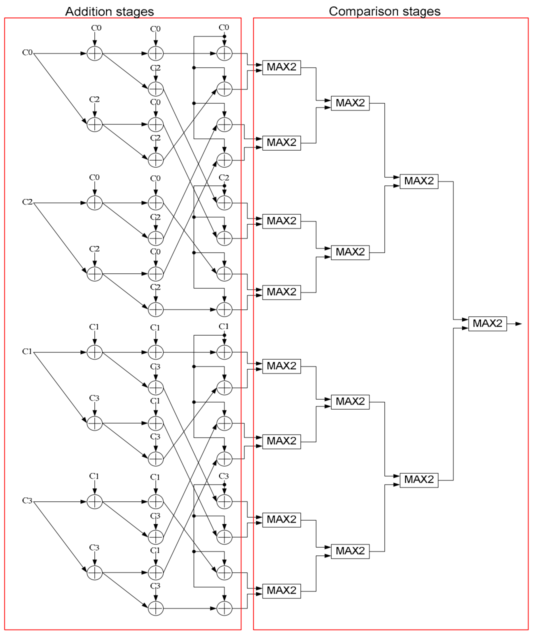

Fig. 4: An example of auxiliary trellis to simplify the BM calculation.

This property can be exploited to reduce the addition operations in the last stage. In the proposed approach, the 16 candidates are divided at the end of the second addition stage into 4 groups, which is indicated as G*k*, *k*=0, 1, 2, 3, in Table I. The 4 candidates in each group are supposed to add the same Euclidian metric from Z_3 in a later stage. Before the last addition operation, if one survival candidate is chosen from each group, then in the last addition stage, only 4 additions, instead of 16, are required. Based on the two-step comparison method, an algorithmic strength reduction scheme is introduced, which significantly reduces the needed comparison operations, as shown in Fig. 5. Furthermore, the survival candidates after the comparison step 1 can be shared by 4 BM computations (*i.e.*, BM 0000, BM 0001, BM 1000 and BM 1001 share the same data, as listed in Table I to Table IV).

Therefore, the computation procedure is divided into 4 parts in Fig. 5. The other 12 BMs have the same property and can be categorized into 3 big groups; in each group, 4 BMs share the data after the comparison step 1. The total number of addition operations of the proposed architecture is $4^2 + 4^3$ from the first two addition stages plus 4×16 from the third addition stage. The computation complexities of the 3 methods are summarized in Table V.

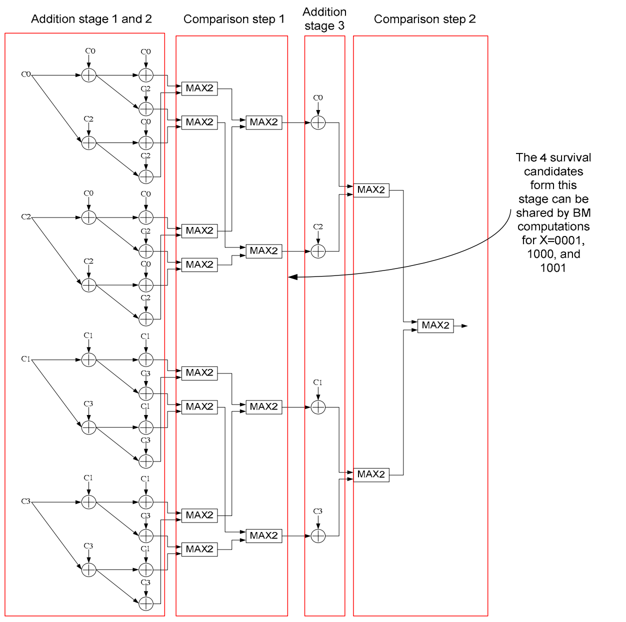
Addition stage 1 and 2
Comparison step 1
Addition stage 3
Comparison step 2
The 4 survival candidates form this stage can be shared by BM computations for X=0001, 1000, and 1001
MAX2

Fig. 5: Computation of BM 0000 using the proposed 2-step comparison method.

TABLE II. GROUPING OF THE CANDIDATES FOR BM 0001

	Z_0	Z_1	Z_2	Z_3		Z_0	Z_1	Z_2	Z_3
G1	C_0	C_0	C_0	C_1	G2	C_1	C_1	C_1	C_2
	C_0	C_2	C_2			C_1	C_3	C_3	
	C_2	C_2	C_0			C_3	C_3	C_1	
	C_2	C_0	C_2			C_3	C_1	C_3	
G3	C_0	C_0	C_2	C_3	G4	C_1	C_1	C_3	C_0
	C_0	C_2	C_0			C_1	C_3	C_1	
	C_2	C_2	C_2			C_3	C_3	C_3	
	C_2	C_0	C_0			C_3	C_1	C_1	

TABLE III. GROUPING OF THE CANDIDATES FOR BM 1000

	Z_0	Z_1	Z_2	Z_3		Z_0	Z_1	Z_2	Z_3
G1	C_0	C_0	C_0	C_2	G2	C_1	C_1	C_1	C_3
	C_0	C_2	C_2			C_1	C_3	C_3	
	C_2	C_2	C_0			C_3	C_3	C_1	
	C_2	C_0	C_2			C_3	C_1	C_3	
G3	C_0	C_0	C_2	C_0	G4	C_1	C_1	C_3	C_1
	C_0	C_2	C_0			C_1	C_3	C_1	
	C_2	C_2	C_2			C_3	C_3	C_3	
	C_2	C_0	C_0			C_3	C_1	C_1	

TABLE IV. GROUPING OF THE CANDIDATES FOR BM 1001

	Z_0	Z_1	Z_2	Z_3		Z_0	Z_1	Z_2	Z_3
G1	C_0	C_0	C_0	C_3	G2	C_1	C_1	C_1	C_0
	C_0	C_2	C_2			C_1	C_3	C_3	
	C_2	C_2	C_0			C_3	C_3	C_1	
	C_2	C_0	C_2			C_3	C_1	C_3	
G3	C_0	C_0	C_2	C_1	G4	C_1	C_1	C_3	C_2
	C_0	C_2	C_0			C_1	C_3	C_1	
	C_2	C_2	C_2			C_3	C_3	C_3	
	C_2	C_0	C_0			C_3	C_1	C_1	

TABLE V. COMPUTATION COMPLEXITY COMPARISON

Methods	Addition operations	Comparison operations
Straightforward	786	240
Conventional auxiliary	336	240
Proposed 2-step comparison	144	96

4. HIGH-SPEED VITERBI DECODER DESIGN

The most computation intensive module of the 4D-TCM decoder is the Viterbi Decoder (VD). The VD consists of an ACSU (add-compare-select unit) and a survivor memory unit (SMU). The ACSU finds the survived path for each state while the SMU keeps a record of the survivors. Register-Exchange (RX) scheme is used in the SMU for low-latency purpose.

The bottleneck of high speed design is the ACSU as it involves recursive computations. The ACSU in this case is very complex that each state takes 8 path metrics (PM) and 8 BMs as inputs and outputs 1 PM and 3-bit index to indicate the survivor path for the corresponding starting state. On the other hand, in conversional approach of VD design, only two paths merge to the same states. For high speed design, a fully parallel architecture is employed. Also, a Radix-2 structure is used, considering that there are many incoming branches to each state.

In order to further increase the clock speed and the throughput, a 2-level comparison scheme is used in the ACSU, instead of the traditional binary tree architecture, which takes 3-level comparisons. Fig. 6 shows the architecture of the proposed 2-level comparison scheme, where LUT, a look-up table, is usually implemented with combinational logic. Based on the 6 comparison results, it decides which of the 4 inputs has the maximum value.

5. FPGA IMPLEMENTATION

The 4-D 8PSK TCM system was first modeled in Matlab for performance simulation and finite wordlength analysis. Fig. 7 shows the simulation results using the Matlab model with various quantization levels for the input signal for the case with Rm=11/12. The signal-to-noise ratio (SNR) to achieve a bit-error rate (BER) of 10^{-4} for the unquantized case, 8-bit quantization, 7-bit quantization, and 6-bit quantization are 13.05 dB, 12.88 dB, 13.20 dB and 13.82 dB, respectively. When the input signal is quantized using 5 bits, a dramatic performance loss is observed in Fig. 6. Generally, 8 bits or 7 bits of quantization of the input signal is a good choice for implementation, since the performance loss is less than 0.2 dB compared with the unquantized system. The system with 8 bits of quantization even performs slightly better than the floating point system. For applications that do not have very strict performance requirements or when the front-end cannot provide more bits, data with 6-bit quantization is still feasible. Quantization level less than 6 bits should not be employed because of the resulting poor performance. In this design, 7 bits of quantization for input signal are chosen.

The decoder part was then modeled in Verilog HDL, simulated in ModelSim and synthesized for Xilinx Virtex 4 FPGA. Table VI summarizes the logic resource report from Xilinx synthesis tool. The placement-and-routing software reported a critical path of 10.35 ns for the 4-D 8PSK TCM decoder design. Therefore, the implementation can achieve a maximum throughput of

$$11\times1/(10.35\times10^{-9}) = 1.062 \text{ Gbps}.$$

Since the TMU and VD are actually designed to meet the requirements for all cases, the maximum throughput for Rm = 8/9 case is: $8\times1/(10.35\times10^{-9})$ = 772.95 Mbps, which is much faster than the 460 Mbps data rate reported in [11].

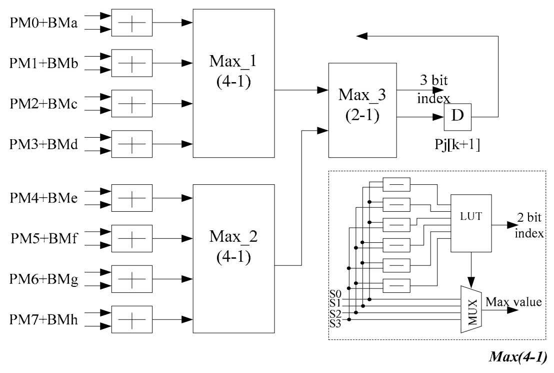

Fig. 6: The proposed 2-level comparison architecture in ACSU.

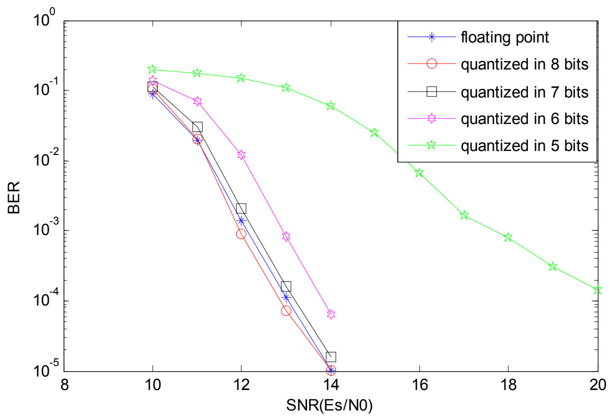

Fig. 7: Quantization analysis.

TABLE VI. XILINX VIRTEX 4 FPGA UTILIZATION STATISTICS

Xc4vlx160-12ff1148 Virtex 4, speed scale 12					
Device Type	Slice Flip-flop	4-input LUT	Bonded IOB	Block RAM	GCLK
Number of Devices	12258	87769	65	16	1
Utilization	9%	64%	8%	5%	3%

6. CONCLUSION

In this paper, a high speed low-complexity 4-D 8PSK TCM decoder is implemented in FPGA. The pipelining technique is used in TMU to improve the throughput, and a new architecture is proposed for TMU to reduce the computation complexity without sacrificing the BER performance. In the VD, a fully parallel structure is used. Inside the VD, a 2-level comparison scheme is adopted in the ACSU, instead of the traditional binary tree architecture, which takes 3-level comparisons, to further increase the clock speed. The FPGA implementation with Xilinx Virtex 4 can achieve a throughput over 1 Gbps.

7. REFERENCES

[1] E. Bigilri, D. Divsalar, P. J. McLane, and M. K. Simon, *Introduction to Trellis-Coded Modulation With Applications*. New York: MacMillan, 1991.

[2] S. G. Wilson, *Digital Modulation and Coding. Englewood Cliffs*, NJ: Prentice-Hall.

[3] M. Y. Rhee, *Error-Correcting Coding Theory*. New York: McGraw-Hill.

[4] Steven S. Pietrobon, Robert H. Deng, 'Trellis-Coded Multidimensional Phase Modulation," *IEEE Transactions on Information Theory*, vol.36, no.1, Jan. 1990.

[5] C. Berrou, A. Clavieux and P. Thitimajshia, "Near Shannon limit error correcting coding and decoding: turbo codes", ICC'93, pp 1064-70.

[6] D. MacKay and R. Neal, "Near Shannon limit performance of low density parity check codes," *Electronics Letters*, Vol. 33, Issue 6, Mar. 1997, pp: 457-58.

[7] "Bandwidth-efficient Modulations", CCSDS 401(3.3.6) Green Book, April 2003.

[8] Lou, H.-L.; Tong, P.; Cioffi, J.M., "A programmable codec design for trellis coded modulation", Volume 2, 3-8 Nov. 1997, page(s):944 - 947 vol. 2.

[9] Jia-Yin Wang and Mao-Chao Lin, "A trellis coded modulation scheme with a convolutional processor," Proc. *IEEE International Symposium on Information Theory*, pp: 389, Jun. 1997.

[10] Anh Dinh and Xiao Hu, "A Hardware-Efficient Technique to Implement a Trellis Code Modulation Decoder," *IEEE Transactions on VLSI systems*, Vol. 13, No. 6, Jun. 2005.

[11] D. Servant, "Design and Analysis of a Multidimensional Trellis Coded Demodulator," Master's Thesis, KTH Signals, Sensors and Systems, Stockholm, Sweden 2004.