Programmable Turbo Decoder Supporting Multiple Third-Generation Wireless Standards

Myoung-Cheol Shin and In-Cheol Park
Department of Electrical Engineering and Computer Science, KAIST
Yuseong-gu Daejeon, Korea
Tel : 042-869-4406 Fax : 042-869-4410
Email : mcshin@ics.kaist.ac.kr, icpark@ee.kaist.ac.kr

Abstract

A programmable turbo decoder is designed for multiple third-generation (3G) wireless communication standards. It mainly consists of a configurable SISO decoder and a 16-bit SIMD processor equipped with five processing elements and specialized instructions customized for interleaving. A fast and flexible incremental block interleaving algorithm that causes no timing overhead in changing interleaver structure is also proposed. The decoder occupies 8.90mm^2 in 0.25μm CMOS with five metal layers, and exhibits the maximum decoding rate of 5.83Mbps.

1. Introduction

As turbo codes [1], or parallel concatenated convolutional codes, have extremely impressive performances, they started to enter the field of standardized systems in recent years. One of the most important examples is the channel coding for high-speed data transmission of the 3G mobile radio systems such as W-CDMA [2] and cdma2000 [3]. Flexible and programmable turbo decoders are required for 3G communications because of two reasons: 1) global roaming is recommended between different 3G standards, and 2) even in a standard the frame size may change on a frame-by-frame basis. The turbo decoder consists of interleavers and soft-input-soft-output (SISO) decoders that decode recursive systematic convolutional (RSC) codes. Flexible and programmable implementation is especially needed for the turbo interleaver, as each 3G standard has a distinct and complicated interleaver.

The most common approach to implement the interleaver is to store the interleaved patterns in a ROM. The approach is not adequate for a turbo decoder supporting multiple wireless standards, as it needs a large-sized ROM to store all the possible interleaved patterns. In [4], Bekooij, et al. proposed a flexible turbo decoder as a form of VLIW microprocessor. However, the interleaver part as well as the SISO is implemented in a dedicated hardware unit that is not programmable.

Table 1. Differences between the cdma2000 and W-CDMA turbo codes

		cdma2000	W-CDMA
Coding rate		1/2, 1/3, 1/4, or 1/5	1/3
Puncturing		Yes, when the rate is 1/2 or 1/4	No
RSC code	Constraint length	4	4
	Transfer function	$\left[1 \quad \frac{1+D+D^3}{1+D^2+D^3} \quad \frac{1+D+D^2+D^3}{1+D^2+D^3}^{*1}\right]$	$\left[1 \quad \frac{1+D+D^3}{1+D^2+D^3}\right]$
	Trellis termination	Appending the bits that make the both encoder states all zero	The same as that of cdma2000 at rate 1/2
Interleaver	Type	Block interleaver	Block interleaver
	Frame size	One of the twelve numbers, 378, 570, 762, …, and 20730	Arbitrary integer between 40 and 5114
	Number of block rows	32 (8–14 of them are unused)	5, 10, or 20
	Permutation operations	Discard of MSB's and bit reversal	Prime numbers and modulo operation

*1) $\frac{1+D+D^2+D^3}{1+D^2+D^3}$ is not used when the rate is 1/2 or 1/3

We propose a multiple-standard turbo decoder implemented with a combination of the dedicated hardware part processing the computationally intensive but regular tasks such as SISO decoding, and the software part running on a programmable single-instruction-multiple-data (SIMD) processor for the tasks requiring flexibility. The turbo interleaving that differs largely depending on the standards is also implemented in software. We developed an incremental interleaving algorithm and specialized instructions suitable for 3G communications so that the SIMD processor can provide interleaved data at the speed of the hardware SISO decoder and change the interleaver structure in a very short period of time.

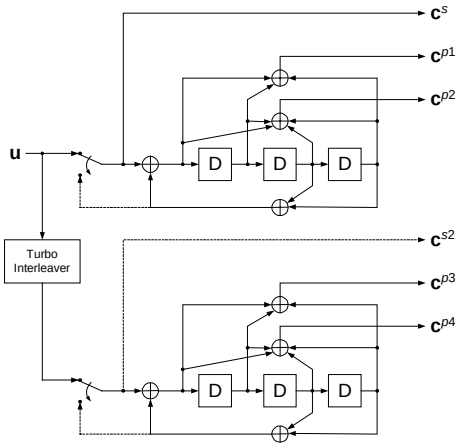


Fig. 1. Turbo encoder for cdma2000

2. Standard Turbo Codes

Targeting a multi-standard turbo decoder for the 3G mobile telecommunication standards, we discuss here about the similarities and differences of the turbo codes of two most popular standards, W-CDMA [2] and cdma2000 [3].

Table 1 shows how the two standards are different, and Fig. 1 shows the turbo encoder architecture of cdma2000. Rate 1/2, 1/3, and 1/4 turbo codes are realized with appropriate puncturing patterns. For the W-CDMA, the code rate is fixed to 1/3 and the encoder obtained simply by deleting the path to $\mathbf{c}^{p2}$ and $\mathbf{c}^{p4}$ in Fig. 1. The code rates other than 1/3 are obtained by a rate matching process [2].

For both cdma2000 and W-CDMA systems, turbo codes are terminated in a similar way. The dotted lines in Fig. 1 are active during trellis termination in order to bring the trellises back to the all-zero state. The W-CDMA termination sequence is the same as that of the rate 1/2 cdma2000 turbo code. We can conclude that a SISO compatible with both standards can be implemented without much difficulty, as the RSC code of W-CDMA is actually a subset of cdma2000.

On the other hand, the two standards have major differences in the interleaver implementations. They share the general concept described in Section 4, but their interleaver frame size, individual operations, and other parameters used in generating interleavers are quite different. The W-CDMA interleaver generation is more complicated and difficult to implement in hardware, whereas the cdma2000 block interleaver of the cdma2000 standard is suitable for hardware implementation.

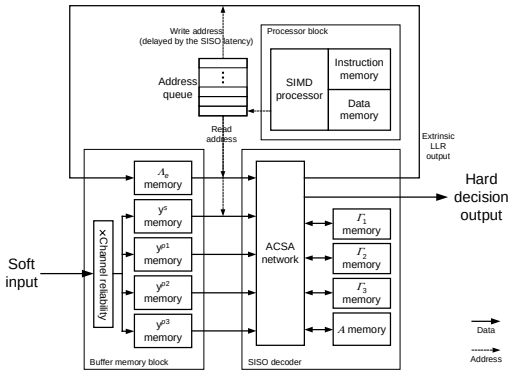


Fig. 2. Block diagram of the proposed turbo decoder.

3. The Turbo Decoder Architecture

As shown in Fig. 2, the proposed turbo decoder has the simplest time-multiplex architecture [5] that contains only one SISO, one interleaver, and one extrinsic LLR (Λ_e) memory, which are repeatedly used for both the first and the second decoding of each iteration. Data are always sequentially stored in Λ_e memory since they are always read and written *in-place*. The data are accessed alternately in a sequential order for the first SISO decoding of an iteration and in an interleaved order for the second decoding. As shown with the dotted lines in Fig. 2, the SIMD processor calculates the interleaved addresses of various standards, and the address queue whose length is the SISO decoder latency saves the read addresses in order to use them again as the write addresses. Besides, the processor controls the hardware blocks, interfaces with an external host, and processes the trellis termination and a stopping criterion during the first SISO decoding that does not need an interleaver. To reduce power consumption, the processor stops the decoding iteration based on a simple stopping criterion [6] that indicates if the further iterations improve BER.

3.1. SIMD Processor

To keep pace with the hardware SISO decoder, the parallel processing is indispensable for interleaved address generation. A simple SIMD architecture depicted in Fig. 3 is chosen, as it is suitable for the simple and repetitive address generation and has simpler control and lower power consumption than VLIW or superscalar architectures.

The number of processing elements (PE's) of

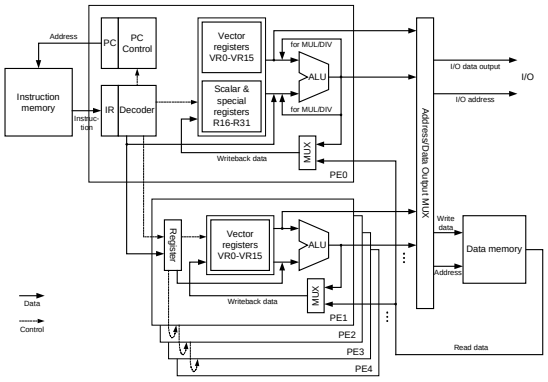


Fig. 3. Architecture of the SIMD processor.

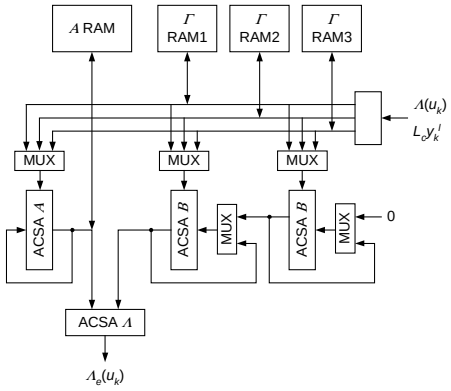


Fig. 4. Overall architecture of the SISO decoder.

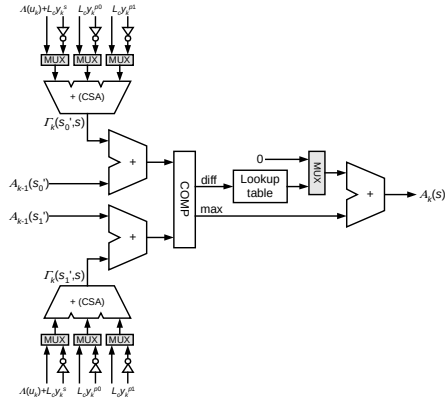


Fig. 5. Add-compare-select-add (ASCA) unit used to calculate a forward metric $A_k(s)$

the SIMD processor is set to five because the number of rows in the W-CDMA interleaver, which is the unit of interleaved address generation, is 5, 10, or 20. That of cdma2000 changes between 18 and 24 if we do not process the rows that always produce invalid addresses. The processor employs Harvard architecture, i.e. the instruction memory and data memory are separated, and has a separate I/O port. Thus the processor can fetch an instruction, load data, and send an interleaved address through the I/O port at the same time. The bit widths of instructions and data are all 16, and the processor has four pipeline stages: IF (instruction fetch), ID (instruction decode), EX (execution), and MEM (memory operation). The write back is carried out at the end of MEM stage. The branch delay is one cycle and the processor always executes the instruction in the branch delay slot.

The first PE, PE0, plays the role of controlling the other PE's and functions like a complete scalar processor. The scalar instructions running only on PE0 include control instructions such as branch and call, and multi-cycle operations such as multiply, divide, and remainder. Basic arithmetic/logic instructions and a few customized instructions for interleaving, which is described in detail in Section 4.2, can be performed by all the PE's. All the common register files of five PE's form a 5-element vector register file of 16 entries as shown in Fig. 3. PE0 has an additional 16-entry scalar register files to store non-vector data or special control data.

The five PE's share a single data memory port and a single I/O port, in order to save memory access power consumption and support simple I/O interface. For this reason, a SIMD instruction is not executed at the same time, but executed in one PE after another so that a data memory port and an I/O port can be shared in a time-multiplexed fashion.

3.2. SISO Decoder

Fig. 4 shows the architecture of the proposed SISO decoder similar to the *memory architecture* presented in [7], where the window size $L = 32$. Input data are read in a memory and used three times for calculating forward metric A's, backward metric B's, and extrinsic LLR Λ_e's. The SISO decoder contains one additional memory for temporarily storing the computed A's and an additional section of add-compare-select-add (ACSA) units to calculate dummy backward metrics for the sliding window algorithm.

As shown in Fig. 5, the SISO decoder employs a configurable ACSA units to support multiple standards. The ACSA can support the rate of 1/2 to 1/5 turbo codes with arbitrary transfer functions by configuring the input multiplexers in Fig. 5. For branch metric $\Gamma_k(s, s')$ calculation, inputs are added in a carry save adder (CSA) to reduce timing delay of multiple additions.

Generally the Log-MAP algorithm outperforms

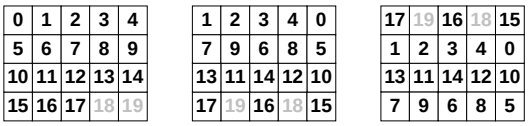


Fig. 6. A prunable interleaver with *N*=18: (a) incoming data indexes, (b) intra-row permutation, and (c) inter-row permutation

the Max-Log-MAP algorithm [8]. However, Worm et al. demonstrated in [9] that the Max-Log-MAP is more tolerant to the channel estimation error than the Log-MAP algorithm. The ACSA unit is therefore designed to select one of the two algorithms. If we can obtain reliable channel estimation, for example, the power control bit of the 3G communication systems, we can choose the Log-MAP algorithm for better performance. On the other hand, if nothing is known about the channel, we should use the Max-Log-MAP to avoid error caused by channel misestimation.

4. Turbo Interleaver Algorithm

The function of the interleaver is to take each incoming frame of N data bits and rearrange them in a pseudo-random fashion prior to the second encoding. Standard turbo codes commonly employ block interleavers. W-CDMA and cdma2000 share the general concept of prunable block interleavers presented in [10]. They can implement interleavers of arbitrary size by first building the *mother interleaver* of a predefined size and then pruning unnecessary indexes. Data are written in a two-dimensional mother interleaver matrix row by row, and permuted within each row. After the rows are also permuted one another, the interleaved data are read out column by column. An example of the prunable interleaver with N=18 is shown in Fig. 6, where the data indexes are written in a matrix form. The intra-row permutation rule applied to Fig. 6(b) is

$$y_{i,j} = b_i + \left[(j+1)\times q_i\right] \bmod 5 \qquad (1)$$

where i and j are row and column indexes, respectively, $\mathbf{b} = (b_0, b_1, b_2, b_3) = (0, 5, 10, 15)$, and $\mathbf{q} = (q_0, q_1, q_2, q_3) = (1, 2, 3, 7)$. Fig. 6(c) shows the inter-row permutation result, which will be read out column by column as a sequence of 17, 1, 13, 7, 2, 11, …, 5, since the elements exceeding the range of interest, 18 and 19, are pruned.

If the interleaver size changes by more than the granularity of the mother interleavers, the entire interleaver structure should be reconstructed. All the 3G mobile communication standards support variable bit rate that may change the interleaver size on a 10ms or 20ms frame basis, and W-CDMA supports multiple separately coded transport channels. This means the interleaver structure may change at every 10ms. Therefore an efficient method to change the interleaver structure in a short period of time is essential to the turbo decoding of 3G communications.

4.1. Preprocessing and On-the-Fly Generation

Generating the whole interleaved address pattern at once is time consuming and requires a large-sized RAM to store the pattern. We propose a solution to this problem by splitting the interleaver generation into two parts: preprocessing for interleaving and incremental on-the-fly address generation. When the bit rate changes, only the preprocessing is performed to prepare a relatively small number of seed variables, which are selected so as to make the on-the-fly generation as simple as possible. Whenever the interleaved address sequence is required, the processor generates it column by column using the variables. The splitting method reduces the timing overhead when the frame size changes. It also does not require a large-sized RAM because it saves only the minimal seed data, not the whole address pattern.

Let us explain our approach with the example of Fig. 6. In order to remove the computationally expensive multiplication and modulo operations that take a lot of clock cycles from (1), we use an increment vector $\mathbf{w}$ of $w_i = q_i \bmod 5$ instead of $\mathbf{q}$ and a cumulative vector $\mathbf{x}_j$ of

$$x_{i,j} = \left[(j+1)\times q_i\right] \bmod 5. \qquad (2)$$

Then (1) can be rewritten as

$$y_{i,j} = b_i + x_{i,j}, \qquad (3)$$

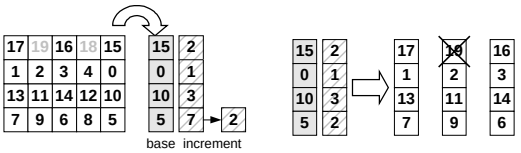
and $\mathbf{x}_j$ can be obtained recursively as

$$\begin{aligned} x_{i,j} &= \left[(j+1)\times (q_i \bmod 5)\right] \bmod 5 \\ &= \left[(j+1)\times w_i\right] \bmod 5 \\ &= \left[(jw_i \bmod 5) + w_i\right] \bmod 5 \\ &= (x_{i,j-1} + w_i) \bmod 5 \end{aligned} \qquad (4)$$

where $j = 1, \ldots, 4$ and $\mathbf{x}_0 = \mathbf{w}$. As $0 \le x_{i,j-1} < 5$ and $0 \le w_i < 5$, $0 \le x_{i,j-1} + w_i < 10$ and (4) is equivalent to

$$x_{i,j} = \begin{cases} x_{i,j-1} + w_i - 5 & \text{if } x_{i,j-1} + w_i \ge 5 \\ x_{i,j-1} + w_i & \text{otherwise} \end{cases} \qquad (5)$$

where the multiplication and modulo operations are replaced by cheaper operations, the multiplication by an addition, and the modulo by a SUBGE (subtract if greater or equal) instruction described in the next section.



(a) **(b)**

Fig. 7. Proposed algorithm applied to Fig. 8: (a) preprocessing for interleaving and (b) incremental on-the-fly address generation.

As shown in Fig. 7(a), **b**, **w** and $\mathbf{x}_0$ for the first column of the block interleaver are calculated and stored in vector registers of the SIMD processor in the preprocessing for interleaving. The figure shows that they are stored in the order of inter-row permutation such as (b_3, b_0, b_2, b_1) in advance so as to free the on-the-fly generation from the inter-row permutation. In the on-the-fly address generation shown in Fig. 7(b), the SIMD processor updates $\mathbf{x}_j$ according to (5) and calculates the addresses based on (3). The calculated addresses are sent to the address queue, if they are smaller than *N*.

We have applied this technique to W-CDMA [2], cdma2000 [3], and CCSDS [11] standard turbo interleavers. It is not difficult to apply the technique to other standard turbo interleavers. A multi-standard interleaver can be realized by loading several interleaver programs on a program memory and switch them whenever needed.

4.2. SIMD Instructions for Turbo Interleavers

To speed up the interleaved address generation, we introduced three SIMD processor instructions: STOLT (store to output port if less than), SUBGE (subtract if greater or equal), and LOOP. Each of them substitutes the sequence of three ordinary instructions but takes only one clock cycle to execute.

STOLT is the instruction that sends the address output to queue only if the generated address is in the range of the interleaver size, i.e. the pruning mentioned above. It is equivalent to the sequence of three instructions CMP (compare), BLT (branch if less than), and STO (store to output port). Another conditional instruction SUBGE, which is equivalent to the sequence of CMP, BGE (branch if greater or equal), and SUB (subtract), is quite useful for the block interleavers that commonly use modulo operations. It substitutes modulo or remainder operation a mod b if the condition $0 \le a < 2b$ is satisfied, which corresponds to (5). LOOP instruction is adopted from DSP processors to reduce the loop overhead. This instruction conforms to a sequence of CMP, BNE (branch if not equal), and SUB instructions, which at once decrements loop count and branches.

Using the special instructions, we can reduce the lengths of the on-the-fly generation loop of W-CDMA, cdma2000, and CCSDS to six, five, and four instructions, respectively. This indicates that the five PE's can provide one address per cycle for cdma2000 turbo decoding.

In addition to the three special instructions, there are a few instructions necessary to implement interleavers. MUL (multiply) and REM (remainder) operations, which take several cycles to complete, are required in the preprocessing of the W-CDMA interleaver. MASK instruction that disables a part of PE's is convenient when the column length of the block interleaver is not a multiple of five.

5. Experimental Results

We implemented an entire turbo decoder system that supports both W-CDMA and cdma2000 1x turbo codes in a 0.25μm CMOS technology. The summary of the chip characteristics and the layout is given in Table 2 and Fig. 8, respectively. The estimated maximum data rate of 5.83Mbps indicates that the decoder can easily cover 3G standards whose maximum rate is 2Mbps.

As shown in Table 3, the performance of the proposed interleaving algorithm is analyzed in terms of the cycle counts for four critical interleaver sizes of the W-CDMA turbo decoding. The last column of Table 3 shows that the on-the-fly address generation is almost as fast as one address per clock cycle for large-sized interleavers of high bit rates. In addition, the preprocessing time is shorter than the SISO decoding time, which completely hides the delay as the preprocessing completes during the first SISO decoding.

Table 2. Summary of the chip implementation.

Technology	0.25μm CMOS, 5-metal
Core size	3.10mm×2.87mm
Gate count of logic cells	34,400
Total RAM size	201kb (including input frame buffers of 123kb)
Supply voltage	2.5V
Maximum operating frequency	140MHz (estimated)
Maximum data rate	5.83Mbps (with 6 iterations)
Estimated power	105mW (at 100MHz)

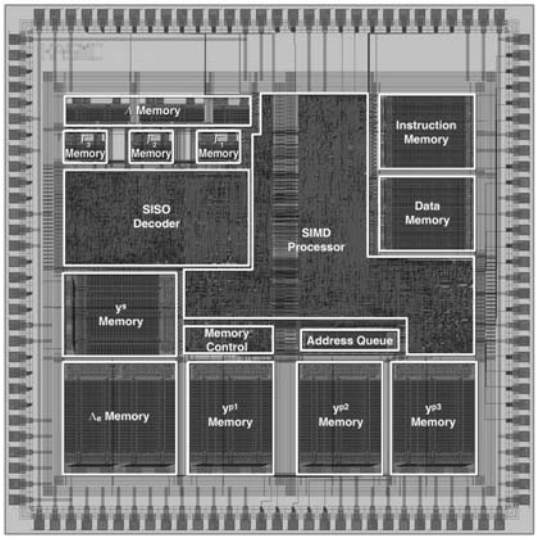


Fig. 8. Layout of the chip.

Table 3. Table of the cycle counts for the most critical interleaver sizes of W-CDMA.

Interleaver size	SISO decoding	Preproc-essing	On-the-fly generation	Bit/cycle
40	172	317	51	0.7843
41	176	295	58	0.7069
5,040	10,350	3,587	5,295	0.9518
5,114	10,498	3,048	5,375	0.9514

6. Conclusions

A programmable turbo decoder is designed for multiple third-generation (3G) wireless communication standards. It mainly consists of a configurable SISO decoder and a 16-bit SIMD processor equipped with five processing elements and custom instructions to perform an interleaving. We also proposed a SIMD processor with customized instructions to speed up the algorithm. The performance and power-efficiency of the hardware and the flexibility of the software are thus achieved together.

We also proposed an incremental interleaving algorithm running on the SIMD processor to support multiple 3G wireless standards. To hide the timing overhead of interleaver changing, the interleaver generation is split into two parts, preprocessing and incremental on-the-fly generation.

The experimental results show that the decoder has sufficient performance for 3G communication standards and the timing overhead of interleaver changing is hidden in most cases.

Acknowledgment

This work was supported in part by the Korea Science and Engineering Foundation through the MICROS center, and the Ministry of Science and Technology and the Ministry of Commerce, Industry and Energy through the project System IC 2010.

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, “Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes,” Proceedings of ICC ’93, pp. 1064–1070, 1993.

[2] 3rd Generation Partnership Project, Technical Specification Group Radio Access Network, Multiplexing and Channel Coding (FDD), 3GPP TS 25.212 v4.2.0, Sep. 2001.

[3] 3rd Generation Partnership Project 2, Physical Layer Standard for cdma2000 Spread Spectrum Systems, 3GPP2 C.S0002-A, ver. 5.0, July 2001.

[4] M. Bekooij, et al., “Power-Efficient Application-Specific VLIW Processor for Turbo Decoding,” ISSCC 2001 Digest of Technical Papers, pp. 180–181, 2001.

[5] J. Vogt, K. Koora, A. Finger, and G. Fettweis, “Comparison of Different Turbo Decoder Realization for IMT-2000,” in *Proc. IEEE GLOBECOM ’99*, pp. 2704–2708, 1999.

[6] R. Y. Shao, S. Lin, and M. P. C. Fossorier, “Two Simple Stopping Criteria for Turbo Decoding,” IEEE Trans. Commun., vol. 47, pp. 1117–1120, Aug. 1999.

[7] G. Masera, G. Piccinini, M.R. Roch, and M Zamboni, “VLSI Architectures for Turbo Codes,” IEEE Trans. VLSI Systems, vol. 7, no. 3, pp.369-379, Sep. 1999.

[8] P. Robertson, E. Villebrun, and P. Hoeher, “A Comparison of Optimal and Sub-Optimal MAP Decoding Algorithms Operating in the Log Domain,” Proceedings of ICC’95, Seattle, WA, pp. 1009–1013, June 1995.

[9] A. Worm et al, “Turbo-Decoding Without SNR Estimation,” IEEE Commun. Lett., vol. 4, pp. 193–195, June 2000.

[10] M. Eroz and A. R. Hammons Jr., “On the Design of Prunable Interleavers for Turbo Codes,” in Proc. VTC’99, Houston, TX, pp. 1669–1673, May 1999.

[11] Consultative Committee for Space Data Systems, Recommendation for Space Data System Standards: Telemetry Channel Coding, CCSDS 101.0-B-5, Blue Book, June 2001.