www.ietdl.org

Published in IET Communications
Received on 22nd November 2011
Revised on 9th February 2012
doi: 10.1049/iet-com.2011.0865

ISSN 1751-8628

Competitive decoders for turbo-like chaos-based systems

A. Wagemakers[1] *F.J. Escribano*[2] *L. López*[3] *M.A.F. Sanjuán*[1]

[1]*Department of Physics, University Rey Juan Carlos, 28933 Móstoles, Spain*
[2]*Department of Signal Theory and Communications, University of Alcalá, 28805 Al-calá de Henares, Spain*
[3]*Department of Telematic Systems and Computing, University Rey Juan Carlos, 28933 Móstoles, Spain*
E-mail: francisco.escribano@ieee.org

Abstract: Recent work has shown that chaos-based communication systems can yield performances as good as their non-chaotic counterparts in additive white Gaussian noise (AWGN), and behave even better in flat-fading channels. However, much of this work relies on computer simulations and there is still a need to study in depth the implementation issues of such systems. The authors address for the first time a fixed-point arithmetic implementation of the iterative decoding algorithm for a recently proposed and successful class of parallel concatenated chaos-based coded modulations. The novel digital signal processor (DSP) results demonstrate that it is possible to implement in standard hardware competitive chaos-based communication systems.

1 Introduction

The possibility of using chaotic signals to carry information became a hot topic in non-linear science and engineering because chaotic signals were supposed to have good properties for a number of important applications [1–10]. However, this interest decayed because many of the systems proposed were not so useful as expected. Nevertheless, we have recently witnessed the arrival of some proposals with good performance as compared with their classical counterparts. Some works, for example, ascertain the possibilities of chaos in secure communications and chaos-based systems working at the waveform level have already shown to be of potential use in multipath fading channels [11], whereas chaos-based systems working at the coding level [12] have shown to provide good results in multi-user channels. Other publications have stressed the fact that chaos-based coded modulated (CCM) systems working at a joint waveform and coding level can be efficient in additive white Gaussian noise AWGN or flat-fading channels [13–16].

This relative success has been achieved by building a comprehensive bridge linking the fields of chaos theory and digital communications. This has made it possible to build parallel concatenated CCM (PCCCM) systems exhibiting coding gains comparable to those achieved with standard binary parallel concatenated systems [14]. It is a known fact that to exploit the low bit-error rate (BER) possibilities of concatenation it is necessary to provide an efficient implementation of the iterative decoding algorithm. In fact, it accounts for almost all the processing delay, and much effort has been traditionally put into improving its efficiency [17].

On the other hand, multi-purpose hardware platforms, such as digital signal processor (DSP), ones have been normally employed when addressing the implementation of the iterative decoding algorithm on a first approach [18]. The main issue that arises when implementing such decoding algorithms is the degrading effect of quantisation and fixed-point arithmetic [19]. There exists a trade-off between complexity and performance that has to be studied before considering more specific hardware solutions. The main aim of this article is to establish a first benchmark and to gain insight into the practical implementation issues of this new kind of system, focusing mainly on the effects of approximations and fixed-point arithmetics. The results hint that further work on this sort of chaos-based systems can help them to successfully join the turbo and turbo-like systems' growing popularity trend.

According to all this, in Section 2, we briefly review the main aspects of the chaos-based system. In Section 3, we focus on the implementation issues. In Section 4, we analyse the DSP-based results and the algorithm performance. In Section 5, we draw some conclusions.

2 System description

The PCCCM systems considered here are the same as proposed in [14]. Thus, we do not review them in full detail, and only recall the main definitions. The concatenated encoder consists of two CCM subsystems, which accept as input blocks of N bits, $\boldsymbol{b} = \{b_1, \ldots, b_N\}$ for one CCM and $\boldsymbol{c} = \{c_1, \ldots, c_N\}$ for the other, where $\boldsymbol{c}$ is the interleaved counterpart of $\boldsymbol{b}$. The first CCM produces a block of chaos-like samples x_{2n-1}, $n = 1, \ldots, N$, at a rate of one sample per input bit and the second CCM produces a second

block of chaos-like samples x_{2n}, $n = 1, \ldots, N$, at the same rate. The PCCCM outputs blocks of size 2 N (total rate $R = 1/2$ bits per chaotic sample). Each x_k, $k = 1, \ldots, 2N$, is a quantised sample that can only take 2^Q possible values distributed uniformly in $[-1, 1]$. The encoding process for each CCM can be described by an equivalent trellis encoder with 2^Q states [14].

We consider here the AWGN and the frequency non-selective flat-fading channel with perfect channel state information (CSI), so that the signal arriving at the decoder side will be $r_k = a_k \cdot x_k + n_k$, $k = 1, \ldots, 2N$, where n_k are samples of the AWGN process with power σ^2. The corresponding sample of the fading process is a_k, which follows a Rician pdf with parameter K [13], and $a_k = 1$ in the AWGN channel. The fading process is described by an uncorrelated sequence of amplitudes a_n which follow a Rician probability density function (pdf) given by

$$p(a_n) = 2a_n(1+K)e^{-a_n^2(1+K)-K} I_0(2a_n\sqrt{K(K+1)}) \quad (1)$$

where $I_0(\cdot)$ is the zeroth-order modified Bessel function of the first kind, K is the ratio of specular to diffuse energy, and $a_n \geq 0$. $K = 0$ corresponds to the Rayleigh case, and $K \to \infty$, to no fading. The mean and variance are given by

$$\eta_a = \frac{1}{2}\sqrt{\frac{\pi}{1+K}} e^{-(K/2)} \left[(1+K) I_0\left(\frac{K}{2}\right) + K I_1\left(\frac{K}{2}\right) \right]$$

$$\sigma_a^2 = 1 - \eta_a^2$$

where $I_1(\cdot)$ is the first-order modified Bessel function of the first kind. $E[a_n^2] = 1$, so that the signal to noise ratio is not affected. The block of r_k samples are input to an iterative decoder for further processing. Let us review briefly the maximum a posteriori (MAP) decoding algorithm for the chaos-based soft-input soft-output (SISO) modules [14]. If the first CCM encoder is initially at state s_i and an input bit b_n drives it at state s_j at time n, with associated output x_{2n-1}, the output log-probability ratio $\Lambda(b_n;O)$ is calculated as follows (In all the following, $(\cdot;O)$ denotes output values or probabilities, and $(\cdot;I)$, input values or probabilities.)

$$\Lambda(b_n; O) = \log \frac{p(b_n = 1; O)}{p(b_n = 0; O)} = \log \frac{\sum_{s_i \xrightarrow{b_n=1} s_j} \exp(\alpha_n(s_i) + \pi(x_{2n-1}; I) + \beta_n(s_j))}{\sum_{s_i \xrightarrow{b_n=0} s_j} \exp(\alpha_n(s_i) + \pi(x_{2n-1}; I) + \beta_n(s_j))} \quad (1)$$

where the summations are taken over all possible transitions $s_i \xrightarrow{b_n=b} s_j$ between pairs of states where b_n takes the value b, $n = 1, \ldots, N$. The quantities $\alpha_n(\cdot)$ and $\beta_n(\cdot)$ are the known forward–backward log-probabilities of the MAP algorithm. $\pi(x_{2n-1};I) = \log[p(r_{2n-1}|x_{2n-1})]$ is the transition log-probability of the channel, and it is the input metric for the algorithm, calculated as $\pi(x_{2n-1};I) = A(r_{2n-1} - a_{2n-1} \cdot x_{2n-1})^2 + B$, where A and B are constants, which depend on the signal-to-noise ratio E_b/N_0 [14].

The probabilities $\alpha_n(\cdot)$ and $\beta_n(\cdot)$ are calculated through a well-known forward–backward algorithm according to

$$\alpha_n(s_j) = \log\left(\sum_{s_i} \exp(\alpha_{n-1}(s_i) + \pi(x_{2n-1}; I) + \pi(b_n = b; I)) \right) + h_\alpha, \quad n = 1, \ldots, N \quad (2)$$

$$\beta_{n-1}(s_i) = \log\left(\sum_{s_j} \exp(\beta_n(s_j) + \pi(x_{2n-1}; I) + \pi(b_n = b; I)) \right) + h_\beta \quad n = N, \ldots, 1 \quad (3)$$

where h_α and h_β are normalisation constants needed to avoid overflows, and $\pi(b_n = b;I) = \log[p(b_n = b;I)]$ is the a priori input log-probability for $b_n = b$. The input log-probability ratio $\Lambda(b_n;I)$ equals zero in the initial step, since $p(b_n = b;I) = 1/2$, $b = 0,1$. To compute (2) and (3), we need the initial values for $\alpha_0(\cdot)$ and $\beta_N(\cdot)$. In each data block, the initial state will be always the zero state; we also perform trellis termination. Therefore $\alpha_0(s_j = 0) = \beta_N(s_i = 0) = 0$ and $\alpha_0(s_j \neq 0) = \beta_N(s_i \neq 0) = -\infty$. The second SISO can be described in the same way, just by writing c_n instead of b_n, x_{2n} instead of x_{2n-1} and r_{2n} instead of r_{2n-1}.

As described in [14], once the iterative decoding algorithm is working in its intended way, the output probability estimations of one SISO decoder ($\Lambda(b_n;O)$, $\Lambda(c_n;O)$) act as input probability estimations for the other one ($\Lambda(c_n;I)$, $\Lambda(b_n;I)$, after the interleaving and deinterleaving stages, respectively), until a fixed number of iterations is reached and a decision is made to get the block of decoded bits $\hat{b}_n$, $n = 1, \ldots, N$. Note that the main difference of this algorithm with respect to the decoding algorithm of a standard binary turbocode lies in the fact that there are 2^Q possible different symbols in the channel to be accounted for when calculating the log transition metrics.

To perform our tests, we have made use of two possibilities among the ones described in [14]: the PCCCM system consisting in the concatenation of two inverse multi-tent map (ImTM) encoding blocks, and the combination of one ImTM encoding block and one multi-tent map (mTM) multi-tent map encoding block. These two PCCCM systems exhibit good trade-offs respecting the location of the waterfall region (ImTM only system), and respecting the level of the error floor (ImTM/mTM system).

3 DSP implementation

When the decoding algorithm described in the previous section is implemented in fixed-point hardware, a number of adaptations are needed [18, 19]. They are mainly related to data precision, data storage, normalisation and arithmetic operations. In our case, the calculation of probabilities will be performed considering a 16-bit fixed-point data type. A floating-point version simulated on a computer is used as a benchmark reference for the performance of the fixed-point version. The platform chosen for the implementation of the algorithm is a TMS320C6713 DSP starter kit.

Owing to the chosen data type, a scaling of the different variables is performed to avoid overflows. This means that the number of bits for the integer and decimal parts are to be chosen to provide a good trade-off between overflow probability and precision. We will focus on the codification of the log-probabilities of the SISO decoder, which should be designed with special care since the overflows usually happen at this step of the algorithm. First of all, noticing that the log-probabilities are always negative, we can drop the sign bit. Thanks to this extra precision bit, it is possible to reach the same BER as with the floating-point data type with a significant increase in computation speed and improvement in memory use. However, some problems appear when using long frame sizes where the forward and backward log-probabilities saturate with higher probability. This saturation can be avoided with the normalisation constants h_α and h_β in (2) and (3), but at the cost of at least 10% speed loss for each loop. For example, for a frame length N lower than 5500 symbols, 12 bits are devoted to the integer part and 4 bits to the decimal part. We will denote this data format as (12, 4). When the frame length is higher than 5500 and lower than 10 000, the integer part has to be extended as (13, 3). This reduction of the available precision leads to a slightly higher BER in the waterfall region for such frame lengths.

One main issue arises when implementing (1)–(3), since they involve the calculation of the logarithm of a sum of exponentials of log-probabilities. This operation is performed using the max* operator (the Jacobian logarithm [20]), defined as

$$\begin{aligned}\log(e^x + e^y) &= \max^*(x, y) \\ &= \max(x, y) + \log(1 + e^{-|x-y|}) \end{aligned} \quad (4)$$

The term $\max(\cdot,\cdot)$ is easy to implement in hardware, whereas, to achieve good performance, the term $\log(1 + e^{-|.|})$ has to be approximated accurately enough [21]. In practice, this is done by using a lookup table (LUT), where a number of samples of such correction term are stored. We thus avoid some time-consuming operations. In this paper, we have followed two approaches: discretising the correction term with 32 values taken at equal intervals (LUT version v1), or using a rough approximation with two values (LUT version v2) [22].

Moreover, producing an efficient DSP code requires following some steps, so as to cope with the intensive computations of the SISO. We disclose here a general method:

- Identification of the parts of the algorithm with a high computational cost (usually loops).
- Elimination of the irrelevant code (e.g. dead loops).
- Changing the 2D arrays to 1D linear-indexed ones.
- Making convenient approximations in the algorithm (e.g. the mentioned max* function approximation).
- Taking advantage of the redundancy in the computations with the use of butterfly multiplication (see details below).
- Implementation of the algorithm with fixed-point arithmetics (with special attention to avoiding overflows or poor precision calculations).
- Optimisation of the DSP code with linear assembly language to improve the use of the instruction pipeline.
- Unrolling the loops wherever possible.

Further improvements of the code rely on the exact transition structure of the trellis. Analysing it more closely, we found out that two different states could be calculated using the same input data values. This fact recalls the butterfly multiplication used in the fast Fourier transform, where a large number of memory accesses are saved by wisely choosing the sequence of operations. We thus exploit the possible symmetry of the trellis, which is a property of the encoder. Depending on the combination of CCM blocks, the code could allow more specific adaptations. The ImTM and the mTM encoders have the following properties:

- The ImTM encoder possesses a nice symmetric trellis, so that we can simplify some loops with the butterfly structure. As a tradeoff, this encoding structure is weaker [14].
- The mTM encoder does not have the same symmetry as the ImTM one, and the state transition appears random, which is an advantage from the error floor level point of view, since the related distance spectrum is sparser. However, it is possible to find a butterfly structure that takes advantage of some repeated memory accesses and overcome this inconvenience to some degree.

Although these modifications do increase the size of the code itself, they save at least $12{\cdot}2^Q$ operations per bit, which represent a saving between 20 and 30% depending on the encoder.

The result, after applying these steps and carefully debugging the code, is an efficient iterative decoder, which keeps the chaos-based properties of the whole system. These properties are mainly related to the fact that they do not comply with the uniform error property (which leads to better perfomance in the flat-fading channel [13]), and that they are almost insensitive to the value of the quantisation factor Q in the error floor region [14]. Nevertheless, the BER at the waterfall region will be somewhat affected by the following critical implementation factors:

- The use of the approximated max* operator.
- The use of a fixed-point data format.

4 Results

We have performed our tests with the specific kind of PCCCM mentioned in Section 2, using different data block lengths and a fixed quatisation factor of $Q = 4$ (16-states encoders), in order to verify the implementation trade-offs. In all the cases, we have made use of S-random interleavers, with high S values [14]. As a comparison, we have included when needed the results for instances of the UMTS and Cdma2000 standard binary turbocodes, with 16-states constituent encoders, rate $R = 1/2$ and data block lengths as close as possible to the PCCCM ones. In all the cases, we have performed seven decoding iterations.

In Fig. 1, we can see the BER results for different cases of the PCCCM system in AWGN. If we focus on the PCCCM with two ImTM CCMs as constituent encoders, we see that we have a loss in the waterfall region when switching from floating-point to 16 bits fixed-point (fp16). This is owing to the trade-off imposed by the large data block (10 000 bits), so that, to avoid overflows, the number of bits devoted to the integer part have to be kept high. As a result, we lose precision when the algorithm starts to converge, but, once full convergence is reached, the BER at the error floor region remains the same. As we will see, when using the

doi: 10.1049/iet-com.2011.0865

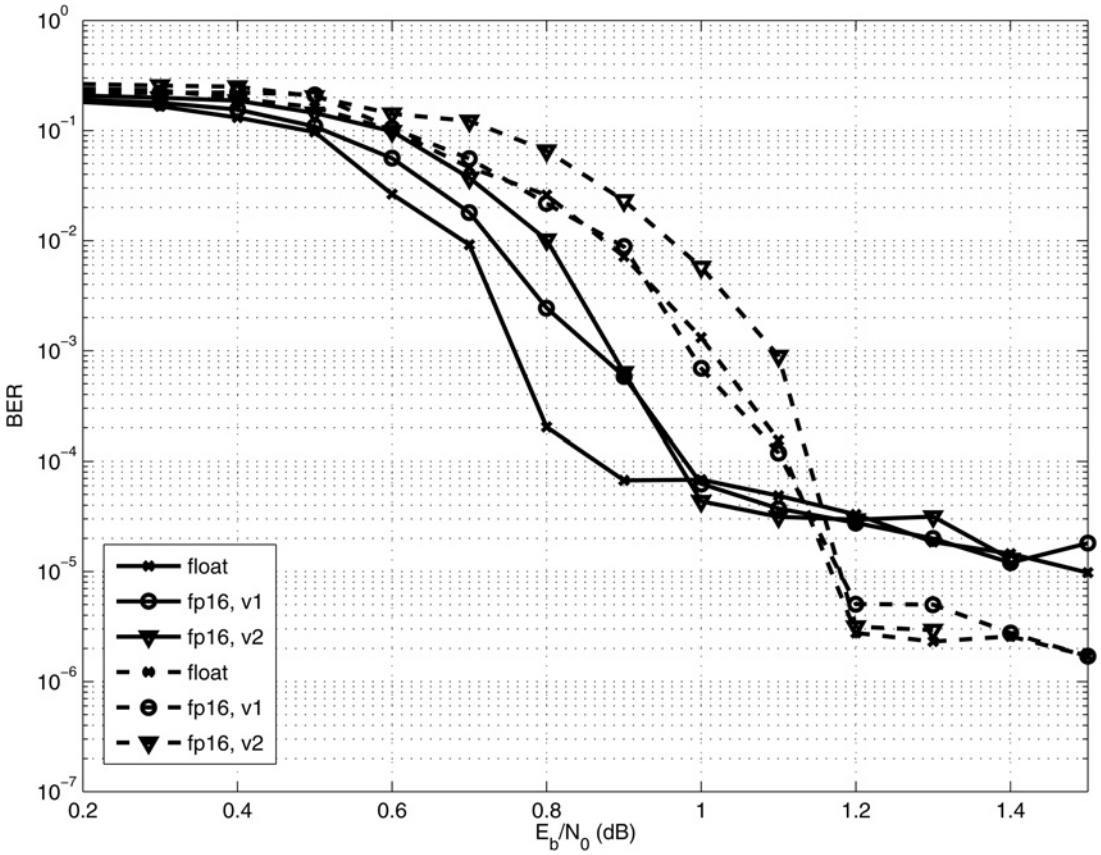

Fig. 1 *Results in the AWGN channel for the ImTM PCCCM with $N = 10\,000$ and $S = 23$ (solid line), and for the mTM/ImTM PCCCM with $N = 4096$ and $S = 14$ (dashed line)*

Floating-point results are with v1 LUT

v2 LUT with only two stored values, we can gain in processing speed, but at the cost of an additional loss in the waterfall region (see Fig. 1). On the other hand, we see that, with a data block length of 4096 (mTM/ImTM case), there is practically no loss between the floating-point and the fp16 cases. This is owing to the fact that we can devote extra bits to the decimal part and gain in precision without allowing overflows. Again, the use of the v2 LUT leads to a loss of a tenth of dB in the waterfall region, whereas the error floor remains basically the same. Therefore special attention has to be paid when using such systems in the vicinity of the waterfall region to meet the specific criteria of processing speed, BER, delay, block data length and so on. When in the error floor region, we can rely on the simplest and fastest implementation, and take into account as target BER the theoretical one.

In Fig. 2, we represent the DSP results in the flat fading channel for the 16-states UMTS and Cdma2000 binary rate 1/2 turbocodes, with standard interleaver lengths 5114 and 6138, respectively. We have made use of the corresponding interleavers provided in the standards for such data lengths. We also plot the results for a mTM/ImTM PCCCM with $Q = 4$, 5500 length S-random interleaver with $S = 19$, so that the encoder and decoder complexity, the data rate and the interleaver block length are comparable. We group the BER results for $K = 20$ (almost no fading), $K = 5$ and $K = 0$ (Rayleigh fading, no line-of-sight component). In all the cases, the tests have been performed with the v1 LUT and the 16 bits fixed-point data type. We can see that, though in the case closest to the AWGN channel both turbocodes outperform the PCCCM, as we get closer to the Rayleigh fading channel case, the latter behaves considerably better, losing up to some tenths of dB less that the former systems. The results presented in [13] pointed out that it could be convenient to exploit the sparse and complex distance spectra of the inherently non-linear CCM in the flat-fading channel, which is a novelty respecting the usual linear systems design criteria of just focusing on minimum distance. We verify here that this property can in fact be exploited in concatenated systems, and, more important, that it still holds when incorporating the trade-offs imposed by the fixed-point implementation.

In Table 1, we can see the processing speed and delay per iteration when using both PCCCM examples with $Q = 4$ and $N = 5500$. The standard turbocodes of Fig. 2 exhibit better delays of about one-eighth of the reported ones for PCCCM, owing to the fact that there is only one metric per bit to be calculated and introduced in the algorithm, whereas for the PCCCM we have 16 metrics per bit ($Q = 4$). This is clearly a disadvantage, but we have to take into account that with an accurate design of the CCM and the use of dedicated *ad hoc* hardware, there should be no problem to get throughputs as high as the ones needed to meet actual demands in wireless communications. Moreover, we have limited ourselves here to a multi-purpose non-optimised platform, just to gain insight on the implementation issues of PCCCM and circa 500 kbps per iteration is a good starting point, if we compare it with the initial implementations of the UMTS turbocode [23].

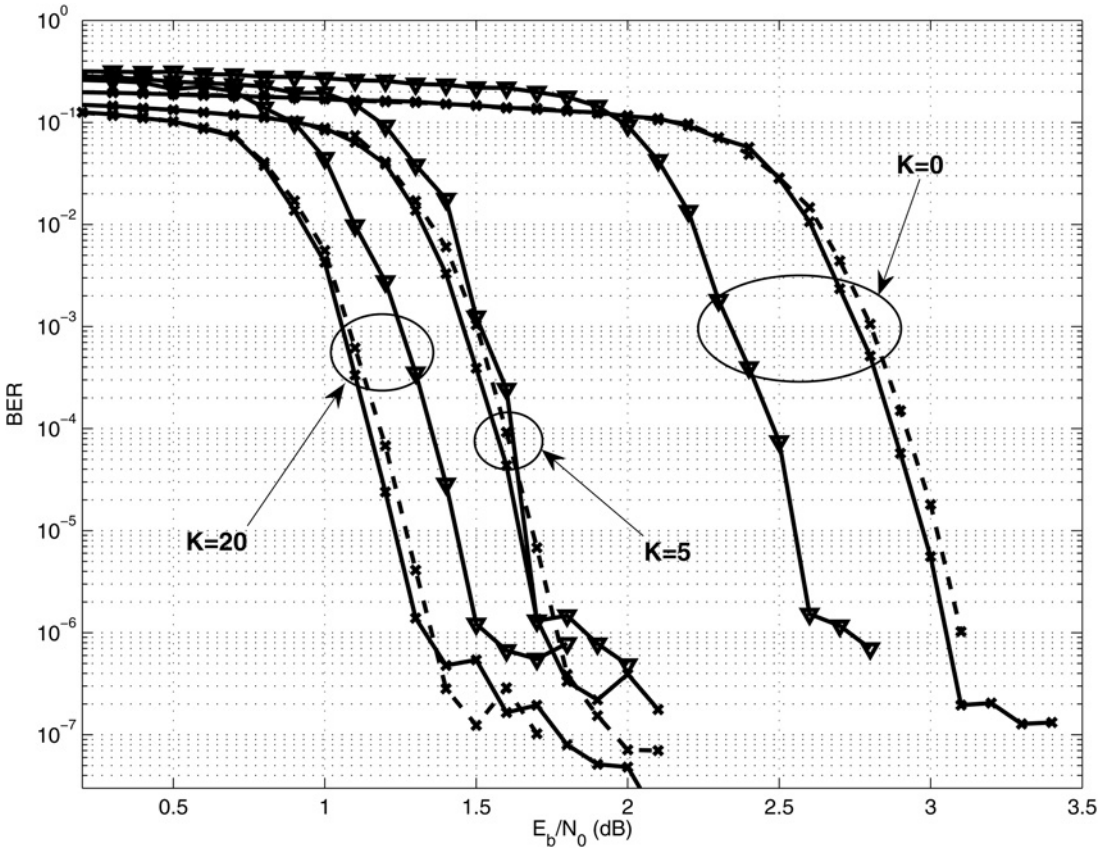

Fig. 2 *Results in the flat fading channel for different K values*
Cdma2000 turbocode is depicted in solid line, '*x*', UMTS turbocode is depicted in dashed line, '*x*' and mTM/ImTM PCCCM is depicted in solid line, '*v*'

Table 1 Performance results for different test cases: fourth column, processing speed; fifth column, processing delay

ImTM PCCCM, performance per iteration				
fp16	$Q = 4$	v1 LUT	462 kbps	12 ms
fp16	$Q = 4$	v2 LUT	644 kbps	8.5 ms
mTM/ImTM PCCCM, performance per iteration				
fp16	$Q = 4$	v1 LUT	493 kbps	11 ms
fp16	$Q = 4$	v2 LUT	627 kbps	8.7 ms

5 Conclusions

We have proposed a specific DSP implementation of the iterative decoding algorithm for a good performing chaos-based turbo-like system and we have illustrated the corresponding trade-offs. We have verified that we can get results in a real platform very close to the theoretical performance whereas keeping reasonable values for the speed and processing delay. Our main purpose was to offer a first benchmark for a new kind of chaos-based communication systems, but the results show that it can already be of potential use in digital communications, specially in flat fading channels. This work proves that chaos in communications is steadily approaching the point of real applicability.

6 Acknowledgment

The authors acknowledge financial support from the Spanish Ministry of Science and Innovation under Project no. FIS2009-09898.

7 References

1 Wang, S., Wang, X.: 'M-DCSK-based chaotic communications in MIMO multipath channels with no channel state information', *IEEE Trans. Circuits Syst. II: Express Briefs*, 2010, **57**, (12), pp. 1001–1005
2 Wren, T., Yang, T.: 'Orthogonal chaotic vector shift keying in digital communications', *IET Commun.*, 2010, **4**, (6), pp. 739–753
3 Min, X., Xu, W., Wang, L., Chen, G.: 'Promising performance of a frequency-modulated differential chaos shift keying ultra-wideband system under indoor environments', *IET Commun.*, 2010, **4**, (2), pp. 125–134
4 Zhang, H., Wang, X.: 'Resource allocation for downlink of DM system using noisy chaotic neural network', *Electron. Lett.*, 2011, **47**, (21), pp. 1201–1202
5 Vali, R., Berber, S., Nguang, S.: 'Accurate derivation of chaos-based acquisition performance in a fading channel', *IEEE Trans. Wirel. Commun.*, 2011, **PP**, (99), pp. 1–10
6 Long, M., Chen, Y., Peng, F.: 'Simple and accurate analysis of BER performance for DCSK chaotic communication', *IEEE Commun. Lett.*, 2011, **15**, (11), pp. 1175–1177
7 Ding, Q., Wang, J.: 'Design of frequency-modulated correlation delay shift keying chaotic communication system', *IET Commun.*, 2011, **5**, (7), pp. 901–905
8 Xu, W., Wang, L., Chen, G.: 'Performance of DCSK cooperative communication systems over multipath fading channels', *IEEE Trans. Circuits Syst. I: Regular Papers*, 2011, **58**, (1), pp. 196–204
9 Wang, L., Min, X., Chen, G.: 'Performance of SIMO FM-DCSK UWB system based on chaotic pulse cluster signals', *IEEE Trans. Circuits Syst. I: Regular Papers*, 2011, **58**, (9), pp. 2259–2268
10 Kaddoum, G., Vu, M., Gagnon, F.: 'On the performance of chaos shift keying in MIMO communications systems'. 2011 IEEE Wireless Communications and Networking Conf. (WCNC), March 2011, pp. 1432–1437
11 Wang, L., Zhang, C., Chen, G.: 'Performance of an SIMO FM-DCSK communication system', *IEEE Trans. Circuits Syst. II*, 2008, **55**, (5), pp. 457–461

doi: 10.1049/iet-com.2011.0865

12 Mazzini, G., Setti, G., Rovatti, R.: 'Chip pulse shaping in asynchronous chaos-based DS-CDMA', *IEEE Trans. Circuits Syst. I*, 2007, **54**, (10), pp. 2299–2314
13 Escribano, F.J., López, L., Sanjuán, M.A.F.: 'Chaos coded modulations over Rayleigh and Rician flat fading channels', *IEEE Trans. Circuits Syst. II*, 2008, **55**, (6), pp. 581–585
14 Escribano, F.J., Kozic, S., López, L., Sanjuán, M.A.F., Hasler, M.: 'Turbo-like structures for chaos coding and decoding', *IEEE Trans. Commun.*, 2009, **57**, (3), pp. 597–601
15 Kozic, S., Hasler, M.: 'Low-density codes based on chaotic systems for simple encoding', *IEEE Trans. Circuits Syst. I*, 2009, **56**, (2), pp. 405–415
16 Escribano, F.J., López, L., Sanjuán, M.A.F.: 'Improving the performance of chaos-based modulations via serial concatenation', *IEEE Trans. Circuits Syst. I*, 2009, **57**, (2), pp. 448–459
17 Valenti, M.C., Sun, J.: 'The UMTS turbo code and an efficient decoder implementation suitable for software-defined radios', *Int. J. Wirel. Inf. Netw.*, 2001, **8**, (4), pp. 203–215
18 Ngo, T., Verbauwhede, I.: 'Turbo codes on the fixed point DSP TMS320C55x'. IEEE Workshop on Signal Proc. Systems, Lafayette, Louisiana, USA, October 2000, pp. 255–264
19 Montorsi, G., Benedetto, S.: 'Design of fixed-point iterative decoders for concatenated codes with interleavers', *IEEE J. Select. Areas Commun.*, 2001, **19**, (5), pp. 871–882
20 Robertson, P., Hoeher, P., Villebrun, E.: 'Optimal and sub-optimal maximum a posteriori algorithms suitable for turbo decoding', *Eur. Trans. Telecommun.*, 1997, **8**, (2), pp. 119–125
21 Loo, K.K., Salman, K., Alukaidey, T., Jimaa, S.A.: 'Parallelised max-log-MAP model', *Electron. Lett.*, 2002, **38**, (17), pp. 971–972
22 Gross, W., Gulak, P.: 'Simplified MAP algorithm suitable for implementation of turbo decoders', *Electron. Lett.*, 1998, **34**, (16), pp. 1577–1578
23 Ebel, W.: 'Turbo-code implementation on C6x'. Technical Report Alexandria Research Institute, Virginia Polytechnic Institute and State University, 1999