Semi-Iterative Analog Turbo Decoding

Matthieu Arzel, Fabrice Seguin, Cyril Lahuec, Michel Jezequel

Semi-Iterative Analog Turbo Decoding

Matthieu ARZEL, Fabrice SEGUIN, Cyril LAHUEC and Michel JÉZÉQUEL
GET/ENST de Bretagne/PRACOM, CNRS-TAMCIC
Technopôle Brest Iroise, CS 83818 – 29238
BREST CEDEX 3, Brittany, France

***Abstract*— This paper presents a novel analog turbo decoding architecture allowing analog decoders for long frame lengths to be implemented on a single chip. This is made possible by suitably using slicing techniques which allow hardware reuse and re-configurability. The architecture is applied to a DVB-RCS-like code. It shows a reduction of occupied chip area by a factor of ten when compared to a conventional slice design with no significant performance degradation. A single 27mm² 0.25µm BiCMOS decoder can then decode any frame length from 40 up to 1824 bits.**

I. INTRODUCTION

Analog decoding has been plagued by the prohibitively large chip area required to implement decoders for long frame lengths and their lack of re-configurability. This increase in occupied chip area is due to the fact that the decoder usually has as many decoding sections as symbols to decode in a frame. This explains why most of the papers published so far deal with non-industrial codes. Only recently, three decoders have been proposed for industrial standards: UMTS (3GPP) in [1], IEEE 802.16a standard in [2] and DVB-RCS-like codes in [3]. Only the first two were implemented on chip and tested; the third one was only simulated. Nevertheless, these decoders were designed in order to tackle some of the shortest frame lengths of their targeted standard and are not re-configurable.

A possible solution to this size increase was introduced in [4]: sliding windows. In this architecture, the turbo decoder "sees" only a small portion of the frame to decode at a time, thus reducing the number of implemented decoding sections. A clear advantage of this solution is that the decoder is able to decode any frame length. However, in this architecture, while the computation of the extrinsic information is done by an analog core, the continuous exchange of extrinsic information between the two constituent decoders is lost. The exchange of extrinsic information is done iteratively. Moreover, the scheme requires data converters which are notoriously power hungry and slow.

This paper proposes an alternative solution which offers significant on-chip area reduction, keeps a (partial) continuous exchange of extrinsic information to help with convergence [3] and is re-configurable: semi-iterative analog turbo decoding. As an example, the semi-iterative architecture is applied to design a double-binary turbo decoder. A comparison in terms of size and performance is done between the fully-parallelized analog slice decoder as presented in [3] and its equivalent semi-iterative analog decoder.

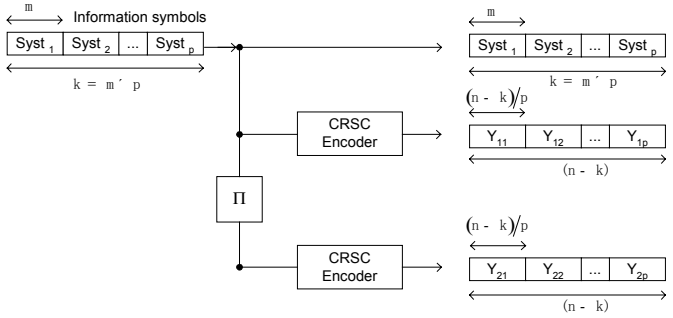

Figure 1. A *p*-slice turbo encoder.

The paper is organized as follows. Part II deals with conventional design using slices. Part III presents the semi-iterative analog decoder with the help of a simple example. Part IV explains how re-configurability can be easily implemented with this architecture. Part V compares the two types of decoders in terms of area and discusses the on-chip area reduction achieved. Finally, part VI presents some simulation results.

II. CONVENTIONAL DESIGN USING SLICES

Fully-parallelized analog slice turbo encoding/decoding was presented in [3] and it will be briefly explained here. In this scheme, a frame of length k is sliced down into p shorter ones of equal length $m=k/p$. As shown in Fig. 1, the resulting sub-frames are independently encoded by a circular recursive systematic convolutional (CRSC) encoder. The overall frame is then interleaved, sliced and encoded by the second encoder. The complete decoder is made up of $2p$ elementary APP decoders, two per slice to decode. In [5], an elementary APP decoder was designed for a frame length of 24 double-binary symbols and was used to design the 48 double-binary symbols

This work was partly funded by *Brest Métropole Océane* under the "Fond de Concours au Développement de la Recherche"

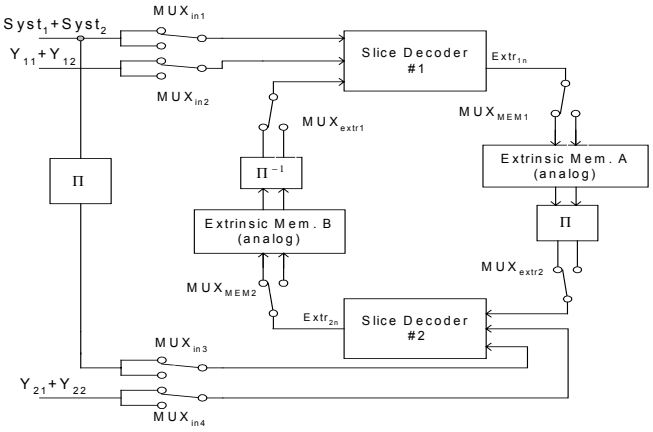

Figure 2. Semi-iterative architecture, 2-slice example.

slice turbo decoder proposed in [3]. Implementing the turbo decoder this way only facilitates its design. This solution does not yield a surface reduction due to the number of elementary decoders required to decode a long frame. The only way to achieve a significant area reduction is to reduce the parallel processing of the data as in the architecture presented below in the following section.

III. SEMI-ITERATIVE DECODING

The architecture of the semi-iterative decoder is shown in Fig. 2. It fully exploits the slicing technique to reduce the required number of elementary decoders to two, regardless of the number of slices. Analog memories, implemented as track-and-holds, and multiplexers are added to the structure. The decoding process is iterative and performs turbo iterations. The number of iteration depends on the performance sought for. Each turbo iteration is divided into steps. There are as many steps as slices. Turbo iterations and steps are detailed next.

For the sake of simplicity, a 2-slice example is taken, Fig. 2. The slice decoders are APP decoders. This example performs the same decoding as presented in [3]. Before the decoding process begins, the full frame is received and stored. The extrinsic memories A and B in Fig. 2 are set to values representing equiprobable data. A two-slice encoding yields two sub-frames: $|Syst_1|Y_{11}|Y_{21}|$ and $|Syst_2|Y_{12}|Y_{22}|$, with Y_{21} and Y_{22} resulting from the encoding of the interleaved frame, $\Pi(Syst_1\ Syst_2)$. With two sub-frames, each turbo iteration is made up of two steps.

Turbo iteration:

Step 1

By means of multiplexers MUX_{in1} and MUX_{in2}, $Syst_1$ and Y_{11} are fed to Slice Decoder #1. This decoder outputs the extrinsic data $Extr_{11}$ which are loaded into the analog memories. Since the extrinsic memories are sized up to equal the length of the overall systematic data, half of the memory remains at equiprobable values. All the extrinsic data loaded in extrinsic memory bank A are then interleaved and multiplexer Mux_{extr2} selects half of them, which are used by the second slice decoder. Some of these data are being computed and tracked by the analog memories, and others have fixed values representing equiprobable data.

In the meantime, the second slice decoder provides the extrinsic data $Extr_{21}$ (linked to the interleaved $Syst_1$, $\Pi(Syst_1)$). $Extr_{21}$ data are tracked by the first half of extrinsic memory bank B, the second half being filled in with equiprobable data. The $Extr_{21}$ data are deinterleaved before being fed back to the first decoder as in a usual turbo scheme. The loop is now closed and there is a continuous exchange of information between the two decoders. The extrinsic data eventually converge to stable values, at which stage they are held in memory and the next step can start.

Step 2

During *step 2*, all the multiplexers have switched from one input set to the other and the second slice is fed to the decoder that is: $Syst_2$, Y_{12}, Y_{22}. The only difference with *step 1* lies in the content of the memories which now contain the extrinsic data $Extr_{11}$ and $Extr_{12}$ computed during step 1. This further enhances the decoding of the second slice since the two decoders benefit from the extrinsic data computed for the first slice.

Once the first turbo iteration has been performed, the whole frame is decoded. However, it is better to run at least another turbo iteration to improve the overall result. This is necessary since during decoding *step 1*, the decoders did not benefit from any extrinsic data relative to the second slice. This second turbo iteration will eventually modify $Extr_{11}$ and $Extr_{12}$ and make them converge toward more reliable values. The number of turbo iterations can be increased but, as in a digital decoder, it is not necessary to run more than half a dozen to get the best results. This was confirmed by running behavioral simulations.

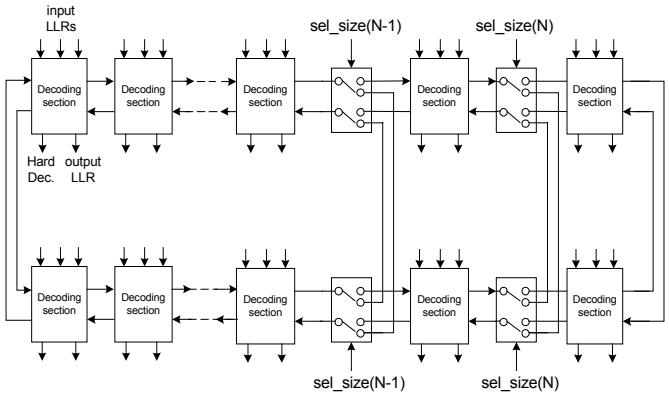

Figure 3. Reconfigurable circular slice decoder. Decoding section as defined in [5].

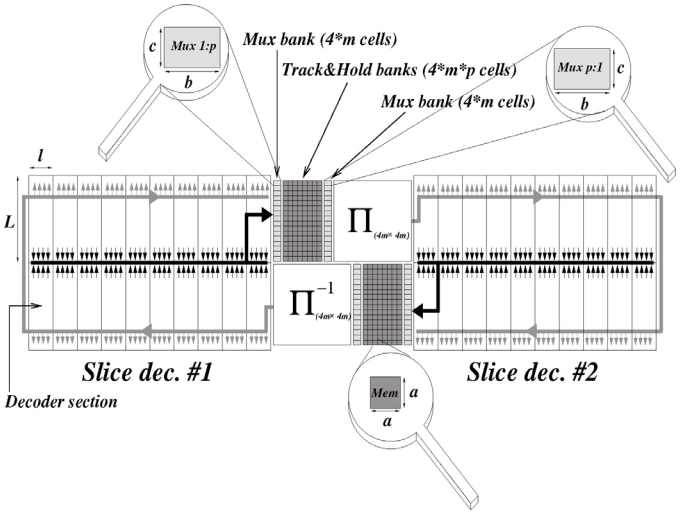

Figure 4. Semi-iterative architecture, element dimensions.

As can be seen from the above example, the iterative process introduced by this architecture does not affect the continuous exchange of extrinsic information which helps with convergence [3]. This is a key feature of the semi-iterative decoder.

The architecture can be easily extended to any number of slices p; a turbo iteration is then composed of p steps and during each step the semi-iterative analog decoder processes 1/p of the overall data. Providing that the slice length remains the same, the slice decoder need not to be redesigned. Only the multiplexers and the memories have to be changed. MUX_{extr1} and MUX_{extr2} are p to 1 multiplexers and MUX_{MEM1} and MUX_{MEM2} are 1 to p multiplexers. The number of memory cells in either memory bank A or B is p, the number of slices, times m, the number of extrinsic data to store per slice. This increase in the number of storage elements is much smaller than that of duplicating the elementary decoder in the case of a fully parallelized architecture as shown in part V. The hardware required to decode large frame length is thus significantly reduced.

IV. RECONFIGURABILITY

An important aspect required by industry is the capability of a single-chip decoder to accept various frame lengths. When using slices this can be easily implemented. First, the elementary slice decoder must be designed for the longest slice length. Since the code is circular, the slice decoder is implemented as a ring. It is fairly easy to reduce the ring size depending on the slice length by means of switches as shown in Fig. 3. The ring size is selected by means of an N-bit word: sel_size. Second, the extrinsic analog memory blocks and the multiplexers represented in Fig. 3 must be sized up according to the longest slice length. Then, when working with a shorter frame length only the memory accesses have to be changed.

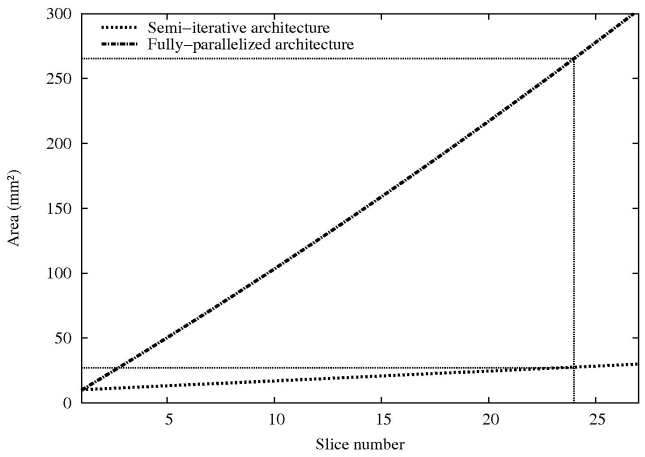

Figure 5. Total on-chip area comparison vs slice number for a fully parallelized and a semi-iterative implementation. The point on each curve corresponds to the decoding of 864 double-binary symbols frames with 24 slices.

To complete the scheme, programmable interleavers as in [6] can be used.

V. AREA STUDY

The architecture is now applied to decode a DVB-RCS-like code, with frame length between 20 and 912 symbols. The comparison is made for two types of decoders designed for the same 0.25µm BiCMOS process: a fully-parallelized decoder and a semi-iterative one. The area estimations for a decoding section and a memory cell are based on the double-binary decoder circuit designed in [5]. Referring to Fig. 2 and Fig. 4, the size of the decoder elements are given next as a function of the number of slices p and the number of systematic data per slice m.

Common to both decoders are the elementary APP decoder and the input memory cells. A decoding section has an area of $L \times l = 0.124 mm^2$; details about its design are given in [5]. The elementary decoder area is thus $L \times l \times m$. The area of a single input memory cell is $2565 \mu m^2$. It is made of two track-and-hold circuits put in parallel. While the first one holds the previous input value, the second samples the next frame. There are $2(p \times m)$ memory cells for the systematic part and $2(p \times m)$ cells per redundant part for a coding rate of one half.

An extra feature of the semi-iterative decoder is the extrinsic memory. Each decoding section deals with a double-binary symbol. Each of its four possible values are weighted by an extrinsic probability. The four extrinsic data output per decoding section are needed by the other decoder and thus each is stored in a memory cell which has an area $a^2 = 625 \mu m^2$. Hence, the total area of the extrinsic memory cells is given by $2 \times 4 \times p \times m \times a^2$.

Fig. 5 compares the total turbo decoder area for two decoders implementing either the fully-parallelized or the

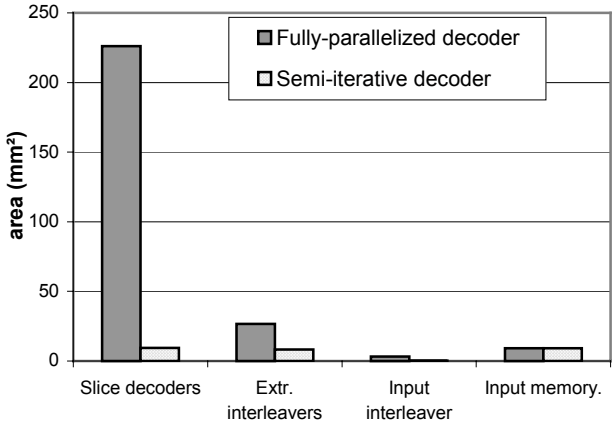

Figure 6. Area comparison, element wise, fully-parallelized decoder as in [3] vs semi iterative decoder for a 864 double-binary symbol frame length (0.25µm BiCMOS process).

semi-iterative architectures versus the number of required slices. In this figure, the interleavers are hard-wired using four metal layers. Taking the example of implementing the turbo decoder for the longest frame length of the DVB-RCS standard, i.e. 864 double-binary symbols, a fully-parallelized slice architecture occupies a 265mm² area while the semi-iterative requires only a 27mm² area. There is thus a reduction of on-chip area by a factor of ten!

Fig. 6 details the area per decoder's elements required to decode the longest frame of the DVB-RCS standard with 24 slices. It clearly shows the dramatic area reduction, the biggest gain coming, of course, from the slice decoders area required. Other significant gains in area are shown for the input interleaver and the extrinsic interleavers. The area of the latter, which includes the extrinsic memories and the multiplexers, is 0.14mm², too small to appear in the figure. Finally, the input memories occupy, of course, the same area since for both decoders, the whole frame has to be stored in memory before the decoding can take place.

VI. SIMULATION RESULTS

In this section, a performance comparison between three decoders decoding 48 double-binary symbol frames using two slices is made. The first decoder is a digital slice decoder, it uses floating point number representation and runs for 5 iterations. The second decoder is the analog decoder presented in [3]. It is fully parallel. The third decoder is a semi-iterative analog decoder. It runs for five turbo iterations. Behavioral simulations were run to obtain the Bit and Frame Error Rate curves shown in Fig. 7. As can be seen from this figure, the decoding performance of the semi-iterative analog decoder lies in between the performance of the fully parallelized analog slice decoder and the performance of the digital counterpart. Therefore, the proposed solution presents no significant performance degradation.

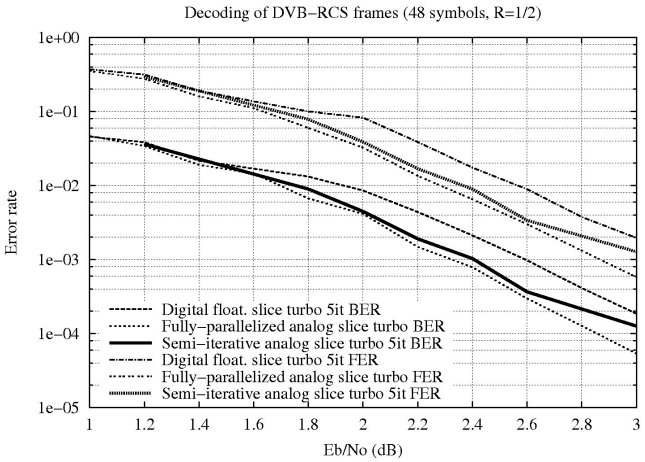

Figure 7. Performance comparison (behavioral).

VII. CONCLUSION

This paper has presented a novel semi-iterative analog turbo decoding architecture. It offers a significant reduction in occupied on-chip area when compared to previous architectures with no significant performance degradation. This was made possible by reducing parallel processing using slices and at the price of introducing iterations. At each iteration, the semi-iterative process keeps a partial continuous exchange of extrinsic information to improve the decoding speed and the correction performance. This is a key feature of the semi-iterative analog decoding which thus provides a good compromise between on-chip area and data rate. Finally, the semi-iterative architecture is fully re-configurable to allow a single 27mm^2 chip to treat different frame lengths, from 40 to 1824 bits as in the example of this paper.

REFERENCES

[1] D. Vogrig, A. Gerosa, A. Neviani, A. Graell i Amat, G. Montorsi, S. Benedetto, "A 0.35-µm CMOS analog turbo decoder for the 40-bit rate 1/3 UMTS channel code", in *IEEE J. of Solid-State Circuits*, vol. 40, n° 3, pp. 753-762.

[2] C. Winstead, "Analog iterative error control decoders", *Ph.D thesis*, University of Alberta, 2005.

[3] M. Arzel, C. Lahuec, F. Seguin, D. Gnaedig and M. Jézéquel, "Analog slice turbo decoding", in *Proc. IEEE International Symposium on Circuits And Systems 2005*, Kobe, Japan, pp. 332-335, May 23-26, 2005.

[4] M. Moerz, "Analog sliding window decoder for mixed signal turbo decoder", in *Proc. ITG Conf. Source and Channel Coding*, Erlangen, pp. 63-70, February 2004.

[5] M. Arzel, C. Lahuec, M. Jézéquel and F. Seguin, "Analogue decoding of duo-binary codes", in *Proc. International Symposium on Information Theory and its Applications 2004*, Parma, Italy, pp. 332-335, October 10-13, 2004.

[6] V. C. Gaudet, R. J. Gaudet, P. G. Gulak, "Programmable interleaver design for analog iterative decoders", in *IEEE Trans. on Circuits and Systems II*, vol. 49, n° 7, pp. 457-464.