Onur ATAR[1], Murat H. SAZLI[2], Hakkı Gökhan İLK[2]

TUBITAK Uzay(1), Ankara University(2)

FPGA Implementation of Turbo Decoders Using the BCJR Algorithm

Abstract. *The most challenging design issue for turbo codes, which is a successful channel coding method to approach the channel capacity limit, is the design of the iterative decoders which perform calculations for all possible states of the encoders. BCJR (MAP) algorithm, which is used for turbo decoders, embodies complex mathematical operations such as division, exponential and logarithm calculations. Therefore, BCJR algorithm was avoided and the sub-optimal derivatives of this algorithm such as Log-MAP and Max-Log-MAP were preferred for turbo decoder implementations. BCJR algorithm was reformulated and wrapped into a suitable structure for FPGA implementations in previous works. Previously reformulated BCJR algorithm was implemented and discussed in this paper for several design issues. Implemented system is verified through simulations. It is observed that the BER performance of the proposed algorithm is better than the Log-MAP algorithm as expected. Despite its superior BER performance, the proposed BCJR turbo decoder has a clear throughput disadvantage. For this reason the decoder has been duplicated. This is done by simply inserting another BCJR turbo decoder on the same FPGA platform, enabling two operating decoders at the same time interval. This simple yet effective modification yields almost doubled throughput results compared to the single BCJR decoder.*

Streszczenie. *W artykule przedstawiono koncepcję budowy turbo dekodera opartego na algorytmie BCJR, zaimplementowanego w układzie FPGA. W celu ułatwienia programowania, zastosowano specjalną strukturę opracowanej metody. Ze względu na ograniczenia przepustowości dekodera, zastosowano dwa takie algorytmy, działające na platformie sterującej równolegle. Pozwoliło to na prawie dwukrotne zwiększenie przepustowości. (**Implementacja turbo-dekodera na platformie FPGA z wykorzystaniem algorytmu BCJR**).*

Keywords: *Duplicated decoder, MAP algorithm, Turbo codes, Turbo coding*.
Słowa kluczowe: podwójny dekoder, MAP, turbo-kody, turbo-kodowanie.

Introduction

Shannon's 1948 paper entitled "A Mathematical Theory of Communication" [1] has been considered as the birth of a new field, "error control coding". In that paper Shannon defined the concept of "channel capacity". He then showed that there exist error control codes that can yield arbitrarily low errors at the receiver output, so long as the transmission rate through the channel is less than the channel capacity. Although he showed the existence of such codes, he did not specify how to construct them.

Even though many efficient coding and decoding schemes have been developed following Shannon, not until the invention of the state-of-the-art turbo codes in 1993 have we been able to approach to the channel capacity limit within just a few tenths of a dB. In other words, turbo codes have closed the significant gap between the coding gains so far achieved using the conventional coding and decoding schemes, and the channel capacity limit.

In order to implement an efficient turbo decoder, a suitable decoding algorithm has to be chosen. Turbo codes have been originally implemented with BCJR (Bahl, Cocke, Jelinek, Raviv) [2] algorithm. However, this algorithm performs complex mathematical operations such as multiplication, division and logarithmic calculations. Therefore, engineers have avoided implementing this complex algorithm and preferred the sub-optimal derivatives of the BCJR (MAP) algorithm such as the Log-MAP and the Max-Log-MAP algorithms which are much simpler to implement but yield worse BER performances [3].

With the advent of the technology, it is possible to implement the BCJR algorithm on a single FPGA. The details of this approach and detailed information about turbo encoders and decoders are given in [4].

The paper is organized as follows. In the next section, reformulation of the BCJR algorithm which is necessary for implementation using FPGA is presented. Then, in Section III, implementation details are given in terms of BER performance, total throughput and multiple turbo decoders, that is duplicating the BCJR turbo decoder within one FPGA platform. Finally, in Section IV conclusions are drawn.

Reformulation of the BCJR algorithm

In this section we will reformulate the BCJR algorithm via some matrix manipulations [5, 6]. In the following we will consider a recursive convolutional encoder with a constraint length K and code memory v = K – 1. There are 2v states of this encoder. We also suppose that BPSK modulation is used, i.e. bit one is mapped to +1, and bit zero is mapped to -1.

A. Calculation of the Forward Metrics (Alpha Coefficients)

Let us begin with the recursive equation to obtain the "α coefficients" of the BCJR algorithm according to eq. 1.

(1)
$$\alpha_k(m) = \frac{\sum_{m'}\sum_{i=-1}^{i=+1}\alpha_{k-1}(m')\gamma_i(R_k, m', m)}{\sum_{m}\sum_{m'}\sum_{i=-1}^{i=+1}\alpha_{k-1}(m')\gamma_i(R_k, m', m)}$$

where m = 0, 1, 2, …, M, is the index of the states with M = 2v – 1.

After some derivations defined in [5] and [6], the reformulation of this recursive equation is turned into an implementable structure of the alpha calculator which is given in Figure 1.

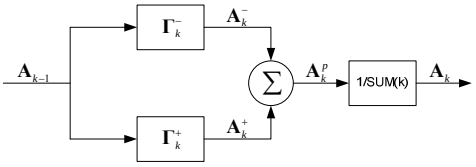

Fig.1. The structure of the alpha calculator

B. Calculation of the Backward Metrics (Beta Coefficients)

As given in eq. 2, a similar procedure was followed to formulate the "β coefficients" of the BCJR algorithm in matrix notation in [5] and [6].

(2)
$$\beta_k(m) = \frac{\sum_{m'}\sum_{i=-1}^{i=+1}\beta_{k+1}(m')\gamma_1(R_{k+1}, m, m')}{\sum_{m}\sum_{m'}\sum_{i=-1}^{i=+1}\alpha_k(m)\gamma_1(R_{k+1}, m, m')}$$

After some derivations defined in [5] and [6], the reformulation of this recursive equation is turned into an implementable structure of the beta calculator which is given in Figure 2.

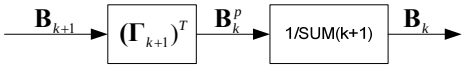

Fig.2. The structure of the beta calculator

C. Calculation of the Logarithmic Likelihood Ratios (LLRs)

Now, we will show how we can compute the logarithm of the likelihood ratios (LLR) associated with each bit d_k using the previously computed $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{\Gamma}$ matrices. LLR associated with each bit d_k is calculated as given in Eq. 3.

(3)

$$\Lambda(d_k) = \ln \frac{\sum_m \sum_{m'} \gamma_{+1}(R_k, m', m)\alpha_{k-1}(m')\beta_k(m)}{\sum_m \sum_{m'} \gamma_{-1}(R_k, m', m)\alpha_{k-1}(m')\beta_k(m)}$$

After some derivations defined in [5] and [6], the reformulation of this recursive equation is turned into an implementable structure of the LLR calculator which is given in Figure 3.

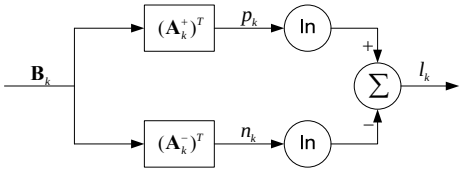

Fig.3. The structure of the LLR calculator

FPGA Implementation of Turbo Decoders Using BCJR Algorithm

Turbo decoders are highly configurable systems. Some of the configuration parameters are chosen by the design engineer, some of them are defined by the 3GPP standard [7]. The parameters used in this work and their values are given in Table 1.

Table 1. Design parameters

Parameter	***Value***
Block Length	***128-4096***
# of RSC Encoders	***2***
Constraint Length of RSC Encoders	***4***
Generator matrix of RSC Encoders	***G={7,5}(oktal)***
Code Rate	R=1/3
Code Puncturing	None
Decoding Algorithm	BCJR (MAP)
Iteration #	1-20

A. General Hardware Structure of the System

General hardware structure of the implemented system is given in Figure 4. The received data stream $R_k(x_k, y_{1k}, y_{2k})$ is handled by the "input handling system" in order to be stored and provided to the appropriate decoders.

As indicated before, reformulated BCJR algorithm is used in component decoders [5]. The hardware structure of this algorithm is given in Figure 5.

Coefficients used by the BCJR algorithm and the noisy received data stream are defined as real numbers. Therefore, these numbers must be represented as "fixed-point" on hardware. Nine-bit numbers are used for both the received data stream and the internal coefficients. Figure 6 illustrates the number representation.

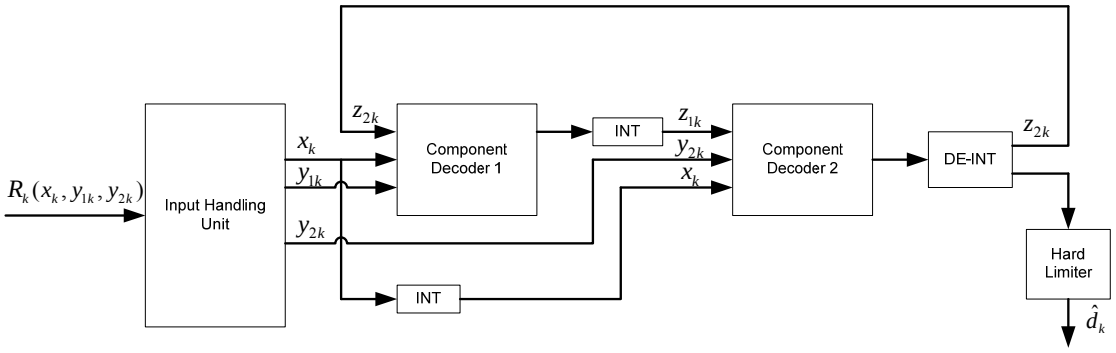

Fig.4. General hardware structure of the system

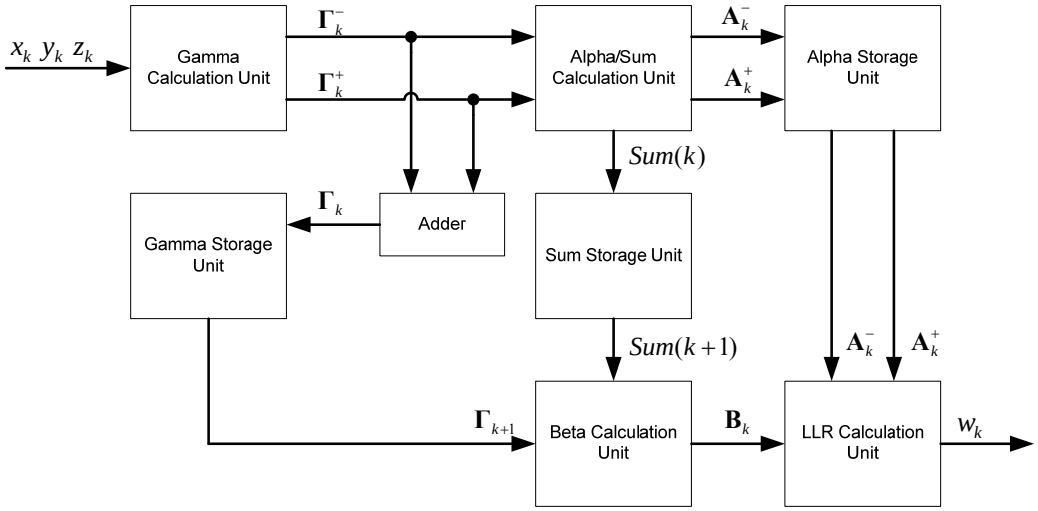

Fig.5. Hardware structure of the component decoders

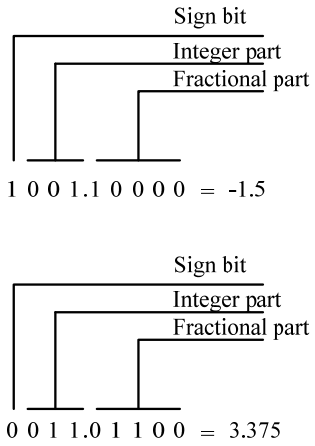

Fig.6. Number representation

B. Gamma Calculation Unit

The state transition metrics (gamma coefficients) is calculated as given in eq. 4.

(4)

$$\gamma_i(R_k, m', m) = \frac{1}{2\pi\sigma^2}\frac{\exp(z_k/2)}{1+\exp(z_k)}\exp(iz_k/2) \cdot q(d_k = i \mid S_k = m, S_{k-1} = m') \cdot \exp\{\frac{-1}{2\sigma^2}[(x_k - i)^2 + (y_k - Y_k)^2]\}$$

Eq. 4 can also be considered as product of three terms. The $\frac{1}{2\pi\sigma^2}\frac{\exp(z_k/2)}{1+\exp(z_k)}\exp(iz_k/2)$ term is the first term, the $q(d_k = i \mid S_k = m, S_{k-1} = m')$ term is the second term and the $\exp\{\frac{-1}{2\sigma^2}[(x_k - i)^2 + (y_k - Y_k)^2]\}$ term is the third term of this product. These three terms are multiplied via the digital elements on hardware. Constants and exponential expressions in these terms are not calculated but read from look-up tables which results in reduced usage of hardware elements (except block memories).

C. Alpha, Beta and LLR Calculation Units

These units implement not only multiplication and addition operations but also division operation which runs slowly on hardware despite the high speed modern FPGAs. Therefore, the division operation is implemented via look-up tables.

Logarithmic calculation is the only complex calculation implemented in the LLR unit. Therefore, it is also implemented via look-up tables similar to alpha and beta calculation units.

D. BER Performance of the BCJR Turbo Decoder

BCJR turbo decoder implemented at this work yields better BER performance than the Xilinx Log-MAP turbo decoder [8]. BER performance comparison between these two turbo decoders is given in Table 2.

E. Total Throughput Performance of the BCJR Turbo Decoder

As indicated before, turbo decoding operation is an iterative operation. For every data block, more than one iteration is completed by the decoder. Thus, turbo decoders yield better BER performance. However, these iterations decrease the total throughput of the decoder. Total throughput performance comparison between the BCJR and the Log-MAP turbo decoders is given in Table 3.

F. Duplicating the BCJR turbo decoder

Despite its superior BER performance, our proposed BCJR turbo decoder has a clear throughput disadvantage. Therefore, for high throughput requiring applications, we can duplicate our decoder. This is done by simply inserting another BCJR turbo decoder on the same FPGA platform. Thus, we can have two operating decoders at the same time interval.

As given in Table 4, this simple retouch yields almost doubled throughput results compared to the single BCJR decoder. It is clear that multiple decoders (as many as needed) can be inserted to the same platform and provided as a monolithic solution at reasonable costs.

Table 2. BER performance comparison of R=1/3 BCJR turbo decoder (using single structure as explained in [4]) and the Log-MAP turbo decoder after 5 iterations

Block Length	*512 Bits*		*1024 Bits*		*2048 Bits*	
SNR	*BCJR*	*Log-MAP*	*BCJR*	*Log-MAP*	*BCJR*	*Log-MAP*
SNR = 0.5 dB	3.10^{-3}	4.10^{-2}	4.10^{-4}	3.10^{-2}	10^{-4}	2.10^{-2}
SNR = 1 dB	8.10^{-4}	10^{-2}	2.10^{-5}	2.10^{-3}	6.10^{-6}	6.10^{-4}
SNR = 1.5 dB	2.10^{-5}	10^{-3}	10^{-6}	4.10^{-5}	4.10^{-7}	2.10^{-6}
SNR = 2 dB	2.10^{-6}	2.10^{-4}	$<10^{-7}$	10^{-6}	$<10^{-7}$	10^{-7}

Table 3. Total throughput performance comparison of R=1/3 BCJR turbo decoder (using single structure as explained in [4]) and the Log-MAP turbo decoder at 139 MHz and 349 MHz clock frequency respectively (Mbps)

Block Length	*512 Bits*		*1024 Bits*		*2048 Bits*	
Iteration #	*BCJR*	*Log-MAP*	*BCJR*	*Log-MAP*	*BCJR*	*Log-MAP*
Iteration = 3	*7,87*	*32,93*	*7,88*	*35,58*	*7,90*	*37,96*
Iteration = 5	*4,69*	*20,68*	*4,72*	*22,42*	*4,74*	*24,00*
Iteration = 7	*3,35*	*15,08*	*3,36*	*16,37*	*3,38*	*17,54*
Iteration = 9	*2,61*	*11,86*	*2,62*	*12,89*	*2,63*	*13,82*

Table 4. Total throughput performance comparison of R=1/3 BCJR turbo decoder (using duplicated structure as explained in Section F. Duplicating the BCJR turbo decoder) and the Log-MAP turbo decoder at 139 MHz and 349 MHz clock frequency respectively (Mbps)

Block Length	*512 Bits*		*1024 Bits*		*2048 Bits*	
Iteration #	*BCJR*	*Log-MAP*	*BCJR*	*Log-MAP*	*BCJR*	*Log-MAP*
Iteration = 3	*15,59*	*32,93*	*15,61*	*35,58*	*15,65*	*37,96*
Iteration = 5	*9,29*	*20,68*	*9,35*	*22,42*	*9,39*	*24,00*
Iteration = 7	*6,64*	*15,08*	*6,66*	*16,37*	*6,70*	*17,54*
Iteration = 9	*5,17*	*11,86*	*5,19*	*12,89*	*5,21*	*13,82*

Implementation Report of the BCJR Turbo Decoder

BCJR algorithm is a computationally complex algorithm. Implementing complex mathematical operations such as multiplication and division significantly increases the usage of hardware elements. Therefore, BCJR algorithm uses more hardware elements and runs slower on hardware due to its complex mathematical operations. The impact of this disadvantage is reduced by using look-up table for complex operations except multiplication. Implementation reports of the BCJR turbo decoder and the Xilinx Log-MAP turbo decoder which operate on exactly the same platform are given in Table 5 and 6 respectively.

Table 5. Implementation report of the R=1/3 rate BCJR turbo decoder

Preferences		
Xilinx FPGA	*XC6VLX75T*	
LUT/FF Pairs	*2264*	
Slice LUT	*36254*	
Slice Register	*2404*	
Blok RAM (36k)	*142*	
Blok RAM (18k)	*0*	
DSP Blocks	*0*	
Speed Grade	*-1*	*-3*
Maximum Clock Frequency	*130 MHz*	*139 MHz*

Table 6. Implementation report of the R=1/3 rate Log-MAP turbo decoder

Preferences		
Xilinx FPGA	*XC6VLX75T*	
LUT/FF Pairs	*3765*	
Slice LUT	*3712*	
Slice Register	*4062*	
Blok RAM (36k)	*6*	
Blok RAM (18k)	*7*	
DSP Blocks	*0*	
Speed Grade	*-1*	*-3*
Maximum Clock Frequency	*285 MHz*	*349 MHz*

Conclusion

The turbo decoding structure based on a previous work [6] is implemented in this work. As indicated in the introduction section, the BCJR turbo decoder is compared with the Xilinx Log-MAP turbo decoder [8]. It is observed that the BCJR turbo decoder yields a better BER performance than the Xilinx Log-MAP turbo decoder as expected.

In spite of its superior BER performance, implementation of the BCJR algorithm has been avoided because of its complexity considering the past VLSI technology. However, modern VLSI technology allows us to implement this algorithm at reasonable costs. The prospective application areas of our proposed implementation are:

- Applications that require low BER with a disclaimed throughput performance. Despite its superior BER performance, the proposed BCJR turbo decoder has a clear throughput disadvantage. For this reason the decoder has been duplicated. This is done by simply inserting another BCJR turbo decoder on the same FPGA platform, enabling two operating decoders at the same time interval. This simple yet effective modification yields almost doubled throughput results compared to the single BCJR decoder. This modification leads to the fact that multiple decoders (as many as needed) can be inserted to the same platform and provided a monolithic solution at reasonable costs.
- Power constraint applications where the desired BER is claimed at low SNR which means low power consumption.
- Applications where both throughput and BER are important design issues. In such a case the proposed approach can be used in parallel by using multiple turbo decoding engine which can provide very high throughput at an already provided low BER. However, this cannot be achieved by Xilinx Log-MAP turbo decoder approach.

REFERENCES

[1] Shannon, C. E., “A Mathematical Theory of Communications”, Bell System Technical Journal, Vol. 27, pp.379-423, 623-656, 1948.

[2] Bahl, L. R., Cocke, J., Jelinek, F., Raviv, J., “Optimal Decoding of Linear Codes for Minimizing The Symbol Error Rate”, IEEE Transactions on Information Theory, Vol. 20, pp. 284-287, 1974.

[3] Robertson, P., Hoeher P., “Optimal and Sub-Optimal Maximum a Posteriori Algorithms Suitable for Turbo Decoding”, European Transactions on Telecommunications, Vol. 8, pp. 119-125, 1997.

[4] Atar, O., Sazlı, M.H., İlk, H.G., “FPGA Implementation of Turbo Decoders”, KTTO 2011 11th International Conference on Knowledge in Telecommunication Technologies and Optics, pp. 103-108, Szczyrk, Poland, June 22-24 2011.

[5] Sazlı, M., H., “Neural Network Implementation of BCJR Algorithm Based on Reformulation Using Matrix Algebra”, IEEE International Symposium on Signal Processing and Information Technology, pp. 832-837, 2005.

[6] Sazlı, M., H., “Neural Network Implementation of BCJR Algorithm”, Digital Signal Processing, Elsevier, Vol 17; pp. 353-359, 2007.

[7] 3GPP, “3GPP Technical Specification”, http://www.3gpp.org, 2010.

[8] Xilinx, Inc., “3GPP Turbo Decoder v4.0 Product Specification”, Technical Journal, http://www.xilinx.com, 2009.

***Authors**: Onur ATAR, TUBITAK Tuzay, The Scientific and Technological Research Council of Turkey, Space Technologies Research Institute, 06100 Ankara, Turkey, E-mail: onur.atar@uzay.tubitak.gov.tr; Assist. Prof. Dr. Murat H. SAZLI, Prof. Dr. Hakkı Gökhan İLK, Ankara University, Electronics Eng. Dept., Dögol Caddesi, 06100 Tandogan, Ankara, Turkey, E-mail: sazli@eng.ankara.edu.tr, ilk@eng.ankara.edu.tr.*