This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TBC.2017.2704425, IEEE Transactions on Broadcasting



# Low latency Parallel Turbo Decoding implementation for future terrestrial broadcasting systems

Hua Luo, Yue Zhang, Wei Li, Li-Ke Huang, John Cosmas, Dayou Li, Carsten Maple, Xun Zhang

***Abstract*—As a class of high-performance forward error correction codes, turbo codes, which can approach the channel capacity, could become a candidate of the coding methods in future terrestrial broadcasting (TB) systems. Among all the demands of future TB system, high throughput and low latency are two basic requirements that need to be met. Parallel turbo decoding is a very effective method to reduce the latency and improve the throughput in the decoding stage. In this paper, a parallel turbo decoder is designed and implemented in Field-Programmable Gate Array (FPGA). A reverse address generator is proposed to reduce the complexity of interleaver and also the iteration time. A practical method of modulo operation is realized in FPGA which can save computing resources compared with using division operation. The latency of parallel turbo decoder after implementation can be as low as 23.2us at a clock rate of 250 MHz and the throughput can reach up to 6.92Gbps.**

***Index Terms*— FPGA, interleave, low latency, parallel turbo decoding, terrestrial broadcasting**

## I. INTRODUCTION

TERRESTRIAL broadcasting technologies are facing a challenge that data rate demand from the users is increasing dramatically. The latest television standards such as HDTV (High Definition TV) and UHDTV (Ultra-High Definition TV) [1]-[3] require that the broadcasting system should support higher throughput and lower latency. Besides, future digital terrestrial TV broadcasting systems are expected to not only support traditional rooftop receivers but also mobile receivers. This makes the demand of mobile data traffic more urgent and drives the research of new digital terrestrial TV technologies [4]. Nowadays, people are not just satisfied with watching TV at home only, they also expect to enjoy broadcasting services with their mobile devices. Therefore the future broadcasting system has to support other services such as WiFi and Cellular Networks [5]. It is also a trend that the mobile broadband and broadcast services, indoor and outdoor services can converge together in the future [6].

As a high performance forward error correction code, turbo codes [7]-[12] are believed to be one of the most robust channel coding methods for wireless communications. In particular, turbo codes are able to facilitate near-capacity transmission throughputs, leading to a wide deployment in the state-of-the-art communication standards such as WiMAX [13] and LTE [14] and could be employed in future potential broadcasting standard [15]. The Logarithmic Bahl-Cocke-Jelinek-Raviv (Log-BCJR) algorithm is employed for the iterative decoding of turbo codes. The decoding process is time-consuming because of the serial nature of Log-BCJR algorithm, which is caused by data dependencies of its forward and backward recursions [17]. This makes it hard to meet the demand of system throughput and latency. More specifically, the target transmission throughput should be multi-Gbps and ultra-low end-to-end latencies can be expected to be targets for future wireless communication standards [16]. Therefore, parallelization of traditional turbo decoding is a practical and effective way to improve the throughput and reduce the system latency at the decoding stage.

Note that a number of parallel turbo decoders have been proposed previously, and most of them mainly tried to improve the level of parallelism in order to get a higher throughput and lower latency. In [18], a fully-parallel turbo decoder was implemented using analog decoder, but only short message lengths are supported. According to [19], a parallel turbo decoder algorithm that operates on the basis of stochastic bit sequences was proposed which requires more processing time than Log-BCJR algorithm. A high performance parallel turbo decoder was introduced in [20] with configurable interleaving network which is implemented on very-large-scale integration (VLSI). A fully-parallel turbo decoding algorithm was studied in [21] which can support all LTE and WiMAX standards. However, the computing complexity is too high and is not practical for hardware platform like FPGA.

For the sake of concept proving for future generation terrestrial systems, it is important that the parallel turbo decoding can be implemented on platform like FPGA due to the high cost of VLSI or Application Specific Integrated Circuits (ASIC). Besides, FPGA is believed to be a keystone for the Centralized/Cloud Radio Access Network (C-RAN), which is one of the promising evolution paths for future mobile network architecture [27]. In this paper, a parallel turbo decoder is implemented on a Testbed which is designed to support multi-Gbps throughput and deployed with several FPGA processors. A reasonable level of parallelism is chosen in order to meet the demand of throughput, latency and acceptable computing complexity as well. A reverse address

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TBC.2017.2704425, IEEE Transactions on Broadcasting



generator is proposed in order to reduce the interleaver complexity and reduce the iteration time at the same time. Modulo operation is an essential part of interleave index generation. We designed a practical method of modulo operation which helps to reduce the complexity in FPGA especially when the parallelism level is high. The contribution of this paper is that we provided a feasible solution of parallel turbo decoder implementation on FPGA with latency reduced and throughput improved.

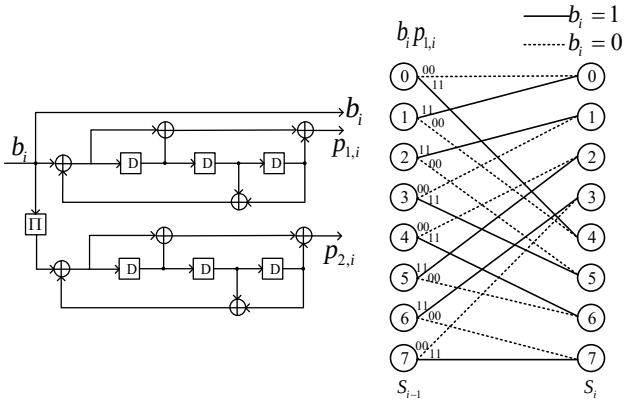
The rest of the paper is organized as follows. Section II provides the background knowledge of turbo encoding and traditional serial turbo decoding algorithm. In Section III, the parallel turbo decoder is introduced plus the proposed reverse interleaving address generator. The implementation of the decoder on FPGA is described in Section IV, in which the simplified modulo operation is introduced. Finally, the experimental results and latency/throughput comparisons are given in Section V and conclusions are made in Section VI.

## II. TURBO ENCODER AND DECODER

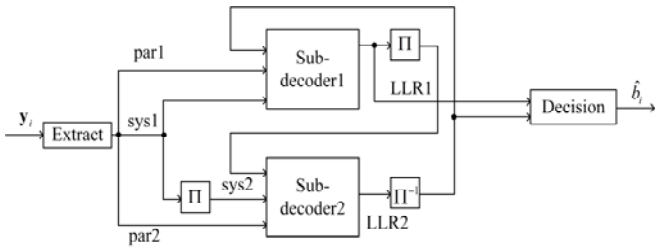
In this section, the background knowledge of turbo encoder and decoder is introduced.

### *A. Turbo encoder*

A turbo encoder is made up of two tail-biting recursive systematic convolutional (RSC) encoders in parallel, as shown in Fig. 1(a). The second RSC encoder is placed after an interleaver ( $\Pi$ ). These two encoders generate two $N$ -bit encoded frames, named a parity frame and a systematic frame. Each RSC coding rate is $R$=1/2 with a codeword length of $N$ and a constraint length of $l=4$. The encoder can also be represented by a trellis diagram as shown in Fig. 1(b) below. Since the message frame uses three encoded frames, the systematic frame ( $b_i$ ), the two parity frames ( $p_{1,i}$ and $p_{2,i}$ ), the turbo encoder produces a total length of $3N$ bits frame $x_i$ and the overall coding rate is $R$=1/3. Following turbo encoder, the encoded frames are modulated and transmitted to the receiver.




(a) (b)




(c)

Fig. 1. (a) Turbo encoder; (b) trellis diagram; (c) turbo decoder

The RSC encoder operates on the basis of an $M=8$ -state transition diagram as shown in Fig. 1(b). The encoder begins from an initial state of $S_0=0$ and transits into each subsequent state $S_i \in \{0,1,2,...,M-1\}$ according to the corresponding message bit $b_i \in \{0,1\}$. Since the message bit $b_i \in \{0,1\}$ has two possibilities, there will be 2 potential transitions from the previous state $S_{i-1}$ to the current state $S_i$.

### *B. Turbo decoder*

At the receiver side, the received frame $y_i$ can be extracted into 3 encoded frames: systematic frame (sys1), parity frame 1 (par1) and parity frame 2 (par2), according to the encoder. Turbo decoder includes two sub-decoders to perform iterative decoding. The sys1 and par1 are transmitted into sub-decoder 1 while sys2, which is generated from sys1 by interleave, and par2 is input into sub-decoder 2. The structure of the decoder is shown in Fig. 1(c). Firstly, sub-decoder 1 generates extrinsic information LLR1 according to systematic, parity and a priori bits. LLR1 is utilized as a priori information by sub-decoder 2 after interleaving. Secondly, the new extrinsic information LLR2 generated by sub-decoder 2 is fed back to decoder 1 after the process of deinterleaver ( $\Pi^{-1}$ ). Therefore, the decoding iteration begins and after sufficient iterations, the performance of the decoder can approach to optimal.

An algorithm named BCJR was proposed in [22] for decoding convolutional codes and was updated by the authors of [23] to process tail-biting codes. For the encoded sequence $\mathbf{x}=\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,...,\mathbf{x}_N$, $\mathbf{x}_i=[x_{i1},x_{i2},x_{i3}]$ is the code word for each input bit $b_i$ and $x_{i1},x_{i2},x_{i3}$ are the sys1, par1 and par2 respectively. As the message bit $b_i$ has two possible values: 0 or 1, we can define the log-likelihood ratio (LLR) as

$$L(b_i)=\ln\frac{P(b_i=1)}{P(b_i=0)} \tag{1}$$

The received sequence $\mathbf{y}=\mathbf{y}_1,\mathbf{y}_2,...,\mathbf{y}_N$ is delivered to the decoder for the estimation of the original bit $b_i$. The decoding algorithm computes a posteriori LLR given by

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TBC.2017.2704425, IEEE Transactions on Broadcasting



$$L(b_i \mid \mathbf{y}) = \ln \frac{P(b_i = 1 \mid \mathbf{y})}{P(b_i = 0 \mid \mathbf{y})} \tag{2}$$

The $L(b_i \mid \mathbf{y})$ can be converted to a bit value through hard decision afterwards. More specifically, if $L(b_i \mid \mathbf{y}) < 0$, the estimation of the message bit will be $\hat{b}_i = 0$ and $\hat{b}_i = 1$ if $L(b_i \mid \mathbf{y}) > 0$. Therefore, the key problem of decoding is the calculation of LLR. After LLR calculation, the extrinsic information will be obtained.

According to [28], the LLR can be defined by the joint probabilities of three parameters, the forward variable $\alpha$, the backward variable $\beta$, and the transition probability $\gamma$. $\alpha$ and $\beta$ can be computed by forward and backward recursions, which means that, to compute the LLR, at least $4N$ times of sampling periods are needed including interleaving and deinterleaving. Let $I$ be the iteration times of the decoding, the overall decoding latency can be given as:

$$D = 4N \cdot I. \tag{3}$$

This is the bottleneck of decoding in terms of latency. Therefore, parallel decoding is needed to reduce the decoding latency.

## III. PARALLEL TURBO DECODING

In this section, the principle of parallel turbo decoding is introduced. The structure of parallel interleaver, which is one of the most complex part of parallel decoding, is explained as well. Moreover, a reverse address generator is proposed for parallel interleaving, which can reduce the time of the decoding process.

### *A. Parallel decoding*

A parallel decoder is performing in parallel by separating the whole block into $P$ sub-blocks, where $P$ is the level of parallelism. In this way, the decoding time is reduced because the length of sub-block $K = N / P$ is much smaller than the whole block. Generally speaking, the higher the level of parallelism, the less decoding time is needed. According to the parallel decoding algorithm proposed in [23], as shown in Fig. 2, the last forward variable $\alpha_{(p-1)K,j}(s)$ and backward variable $\beta_{pK+1,j}(s)$ from the previous iteration of the neighbor sub-blocks ($(p-1)$th and $(p+1)$th) are utilized as the initial value of computation for this $p$th sub-block. Here, $\alpha_{i,j}(s)$ and $\beta_{i,j}(s)$ represent the forward and backward information at $j$th trellis stage of the $i$th decoding bit with state $s$. $\alpha_0$ and $\beta_0$ denote the forward and backward initial variable values of each sub-block in the first iteration and also represents the first and last sub-blocks in the later iterations respectively.

Note that between each iteration, the output LLR of the previous iteration will be processed by interleaving/deinterleaving. For simplicity, the interleaving/deinterleaving process is not shown in Fig. 2. All the decoders of each sub-block are performed in parallel and simultaneously so that the parallel decoder can reduce the decoding time to $1/P$ of the sequential decoding time.

### *B. Interleave and deinterleaving*

The interleaver is a very important part of channel coding performance of turbo codes. For the cooperation of parallel decoding iteration, the interleaver/deinterleaver should be designed to be parallel as well. A memory access contention may occur during the interleaving of extrinsic information. Therefore, based on some algebraic constructions, contention-free interleavers have been proposed in [24], [25] and references therein. In our case, the block size is $N$, the interleaver is defined as

$$\begin{cases} \Pi(i) = A(i) \bmod N \\ A(i) = f_1 i + f_2 i^2 \end{cases}, i = 0,1,2...,N-1, \tag{4}$$

where $f_1$ is an odd number and $f_2$ is even, $i$ is the index number of input data $y_i$ and $\Pi(i)$ is the index number after

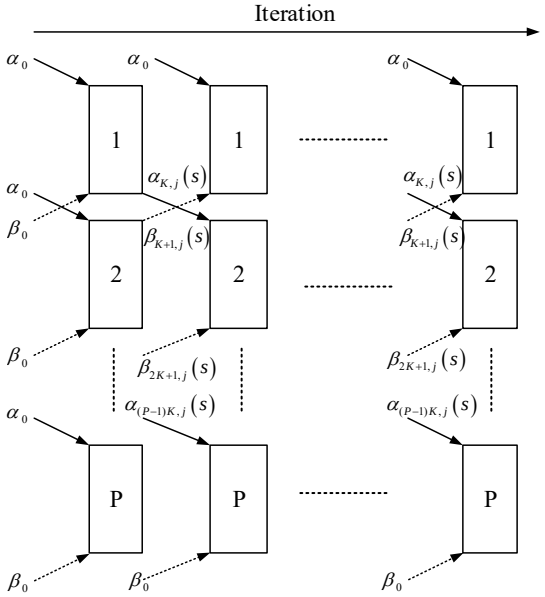



Fig. 2. Parallel forward and backward computation

interleaving. For the parallel interleaver, if the parallel level $P$ can divide block size $N$, then this interleaver is contention-free [20].

In order to generate the target interleaving address according to (4), the compute complexity is quite high if using realtime multiply operation to calculate $A(i)$ because the index $i$ increases progressively till $N-1$. Therefore, an optimized address generator is proposed in [20], which has low complexity. The address generation is accomplished by recursion and the derivation is as follows. According to (4),

$$\begin{aligned} &A(0) = 0, A(1) = f_1 + f_2 \\ &A(i+1) - A(i) = f_1 + f_2 + 2i \cdot f_2 \\ &A(i+2) - A(i+1) = f_1 + 3f_2 + 2i \cdot f_2 \end{aligned} \tag{5}$$

then

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TBC.2017.2704425, IEEE Transactions on Broadcasting



$$A(i+2)-2A(i+1)+A(i)=2f_2\,. \tag{6}$$

Since $A(0)$ and $A(1)$ are known initial factors, by recursion, the following interleaving index can be generated from (6). In this way, no multiplication is needed, which helps reduce complexity dramatically. This address generator cannot be used in a parallel interleaver directly because all the sub-blocks are processing simultaneously hence a parallel address generator is needed.

The memory of parallel interleaver is divided into $P$ banks corresponding to $P$ sub-blocks. The $i$th extrinsic information will be stored in the $\Pi(i)/K$th bank at the address of $\Pi(i) \bmod K$ after interleaving. In addition, deinterleaving is the inverse operation of interleaving for which the principle of address generation is the same as interleaving.

*C. Reverse address generator*

Based on the forward and backward computation structure of turbo codes, the sequence of backward variables $\beta_{i,j}(s)$ should be reversed in order to calculate the extrinsic information. This process adds processing time by at least $N$ clock cycles for sequential decoding or $K$ clock cycles for parallel decoding as shown in Fig. 3(a). Utilizing the characteristics of interleaving, the sequence of extrinsic information can remain reversed and does not affect interleaving while the processing time can be reduced to 3/4 of the original sequence interleaving (see Fig. 3(b)). $L_{I,k}, k=1,2,...,K$ and $L_{O,k}$ represent the LLR of a sub-block before interleaving/deinterleaving and after interleaving/deinterleaving respectively.

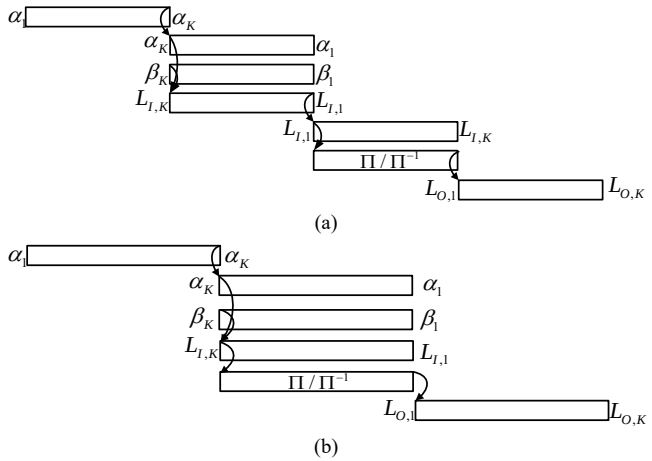



Fig. 3. Timing diagram of each iteration; (a) original sequence interleaving; (b) reversed sequence interleaving.

Since the sequence of interleaver input is reversed, the address generator should be changed accordingly. Therefore, we proposed a reversed address generator for parallel turbo decoding to reduce the computation complexity and processing latency as well.

The address of target memory bank $\Pi(i) \bmod K$ can be modified as

$$\begin{cases} \Lambda_{p,k} = \Pi(pk) \bmod K \\ \quad\quad = A(pk) \bmod K \\ A(pk) = f_1 pk + f_2 (pk)^2 \end{cases}, k = K-1, K-2, ..., 0,\ p = 1,2,...,P\,. \tag{7}$$

Note that the first two addresses of each sub-block that need to be generated are $\Lambda_{p,K-1}$ and $\Lambda_{p,K-2}$. According to (7),

$$\begin{aligned} \Lambda_{p,K-1} &= f_1 p(K-1) + f_2 (p(K-1))^2 \bmod K \\ \Lambda_{p,K-2} &= f_1 p(K-2) + f_2 (p(K-2))^2 \bmod K \end{aligned}, \tag{8}$$

since $f_1$, $f_2$ and $p$ are all integers, (8) can be simply modified to

$$\begin{aligned} \Lambda_{p,K-1} &= -f_1 + 2f_2 \bmod K \\ \Lambda_{p,K-2} &= -2f_1 + 4f_2 \bmod K \end{aligned}. \tag{9}$$

Using similar derivation as (5), the following address of each filter bank can be generated by recursion

$$\Lambda_{p,k+2} - 2\Lambda_{p,k+1} + \Lambda_{p,k} = 2f_2 \bmod K\,. \tag{10}$$

From (10), we can find that the recursion process and initial values have nothing to do with $p$ hence the addresses of all these sub-blocks are the same and only one channel of address generator is necessary for this parallel interleaver.

$$\Lambda_{k+2} - 2\Lambda_{k+1} + \Lambda_k = 2f_2 \bmod K \tag{11}$$

The destination bank that the LLR of a sub-block should be mapped into is decided by the value of $\Pi(i)/K$. The division operation here is costly therefore recursive computation is needed for this reverse address generator. Let $\Pi(i)/K$ be redefined as

$$\begin{aligned} \Gamma_{p,k} &= \Pi(pk)/K \\ &= [A(pk) \bmod N]/K \end{aligned} \tag{12}$$

The recursion has two dimensions. First, the recursion direction is from $k=K-1$ to $k=0$. $\Gamma_{p,K-1}$ and $\Gamma_{p,K-2}$ are the initial values. Second, another recursion is performed from $p=1$ to $p=P$ where $\Gamma_{1,k}$ and $\Gamma_{2,k}$ are the initial values. In order to accomplish this two dimensional recursion, $\Gamma_{1,K-1}$, $\Gamma_{1,K-2}$, $\Gamma_{2,K-1}$ and $\Gamma_{2,K-2}$ must be known before the computation.

$$\begin{cases} \Gamma_{p,k+2} - 2\Gamma_{p,k+1} + \Gamma_{p,k} = [2p^2 f_2 \bmod N]/K \\ \Gamma_{p+2,k} - 2\Gamma_{p+1,k} + \Gamma_{p,k} = [2K^2 f_2 \bmod N]/K \end{cases} \tag{13}$$

Since the interleaver/deinterleaver is placed after the whole computation of $\alpha_{i,j}(s)$, $\Gamma_{1,K-1}$, $\Gamma_{1,K-2}$, $\Gamma_{2,K-1}$ and $\Gamma_{2,K-2}$ can be calculated via pipeline of the multiplication cell before the address generator, as well as $[2p^2 f_2 \bmod N]/K$ and $[2K^2 f_2 \bmod N]/K$. By this recursion, no realtime multiplication is needed during the address generation. With the reverse address generator mentioned above, the parallel interleaver/deinterleaver can

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TBC.2017.2704425, IEEE Transactions on Broadcasting



reduce processing time compared to the method in [20] even though it may cost a little more computation resources.

## IV. TURBO DECODER IMPLEMENTATION

Due to its low cost and short development cycle, FPGA is one of best hardware platform choices for a real-time proof of concept system. In this work, the parallel LTE turbo decoder including the proposed interleaving address generator is implemented on Xilinx Virtex VII. In this section, the detail of decoding implementation is introduced. This decoder can support all the block sizes of the LTE standard. Different parallel level $P$ can be configured according to the specific block size. Considering that the higher the parallel level is, the more complex the decoder will be and the more computing resources will be used, $P$ is set to be 64 when the block size $N$ ranges from 2048 to 6144 and $P=8$ when $256 \le N < 2048$, otherwise $P=1$.

### A. Extraction

The LTE received data before turbo decoding has a certain format, with all the systematic bits and two frames of parity bits included. Therefore, before the calculation of LLR, the extraction of the received data frame is needed. An interleaver is located here as well in order to generate sys2 to match par2 for sub-decoder 2, as shown in Fig. 4. A FIFO is placed after interleaver in order to synchronize with par2.

Since during the iteration of decoding, all these systematic and parity frames that are going to be reused, sys1, par1 and par2, are stored in block RAMs and the read of RAMs is controlled by the request signals (req_1 and req_2) from the LLR calculation module. As the extraction will generate a group of parallel input data for LLR calculation, a configurable parameter is used here to make this decoder compatible with different parallelism levels. Moreover, to produce the same number of block RAMs according to the parallel level, the method of source code generation is utilized. For example, this generate operation was created in Verilog HDL.

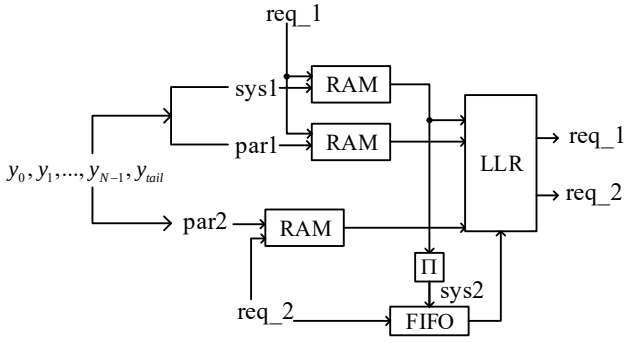



Fig. 4. Extraction of received data.

### B. Extrinsic information

Extrinsic information calculation includes forward variable $\alpha$ and backward variable $\beta$ calculation. As shown in Fig. 5, a block RAM is placed after $\alpha$ module in order to reverse the sequence as mentioned in Section III. Source code generation is used here as well to produce $P$ groups of $\alpha$ modules, $\beta$ modules and LLR modules.

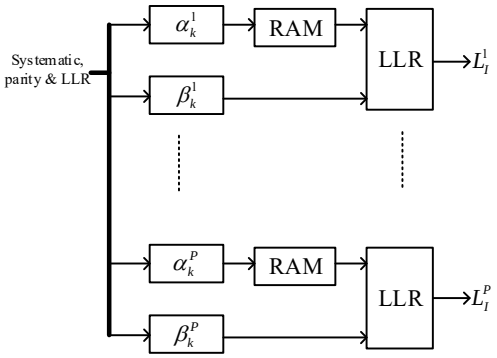



Fig. 5. Parallel extrinsic information calculation.

Note that in the theoretical calculation of $\alpha$ and $\beta$,

$$\alpha_{i,j}(s) = -\infty, s \neq 1$$
$$\beta_{i,j}(s) = -\infty, s \neq 1 \qquad (14).$$

However, minus infinity does not exist in practical fixed-point calculation. A logical comparitor is utilized because if $\alpha = -\infty$, then $\alpha + x = -\infty$ where $x$ can be any value except infinity. Hence $-\infty$ in FPGA is replaced by a least signed value. More specifically, in our case, a 16 bit hexadecimal 2's complement value 8000H is used. By comparison, if $\alpha$ equals 8000H, then $\alpha + x$ is still 8000H. The same method is used to deal with $\beta$.

### C. Interleaver/Deinterleaver

As mentioned in Section III, the interleaver is contention-free as long as the parallel level $P$ can divide the block size $N$. Memory contention does not happen in our study because all the block sizes can be divided by $P$. The interleaver/deinterleaver is a memory dynamic mapping process. The target address and memory bank are generated by the proposed reverse address generator. As shown in Fig. 6, for the interleaving process, using multiplexer in FPGA, the realtime LLR can be written into the related block RAMs and read from them sequentially after writing has been completed.

As mentioned in Section III, the RAM write address $\Lambda_{p,k}$ of each sub-block is independent of $p$ so only $\Lambda_{1,k}$ is produced from the address generator. The LLR results of LLR modules are mapped to different RAMs according to $\Gamma_{p,k}$. On the other hand, the write process is sequential for deinterleaver while read address $\Lambda(i)$ and bank number $\Gamma(i)$ are generated by the address generator.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TBC.2017.2704425, IEEE Transactions on Broadcasting



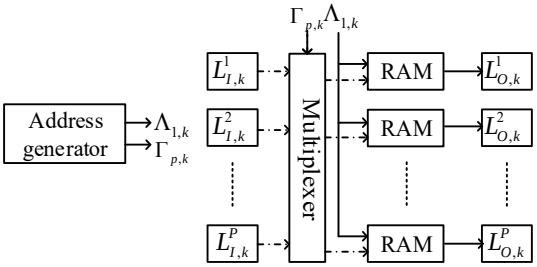



Fig. 6. Parallel Interleaving.

### D. Modulo operation

Modulo operation $C\%D$ is a costly part of the address generator. The result of modulo is the remainder of a division operation. For Xilinx FPGA, the only existing function for modulo operation is the division intellectual property (IP) core which takes many logic units. Some other faster methods like the bitwise operation also exist but they assume $D$ as a constant or the number of powers of 2 [26]. In our study, $D$ is not a constant or a power of 2. A modulo function based on Verilog HDL should be designed with less computing resource and fast speed as well.

Inspired by the bitwise operation, we designed a modulo function that uses a shifter and comparator to get the remainder of the division but not the quotient. Let $E$ be the maximum bit width of $C$, $F$ be the maximum bit width of $D$. The procedure of the proposed modulo function is as Fig. 7 below.

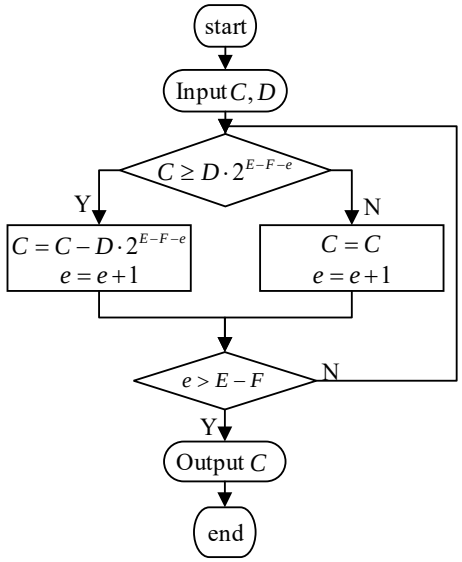



Fig. 7. Modulo function flowchart.

### E. Double buffering

In order to maximize the throughput of the turbo decoder, double buffering is utilized in this design. Since the calculation of $\alpha$, $\beta$ and LLR is sequential, the previous module is idle when the latter module is working. For instance, as shown in Fig. 3, $\beta$ module works after the whole sub-block calculation of $\alpha$. Obviously, another sub-block of $\alpha$ can keep calculating during that period, as shown in Fig.8 below. In this way, the whole decoder can decode two frames simultaneously which can nearly double the throughput. Double buffering is very functional as only double storage space is used, however the logic and compute resources, e.g. lookup tables (LUTs), Flip-Flop, Multipliers, are reused so the utilization of FPGA resource is more efficient.

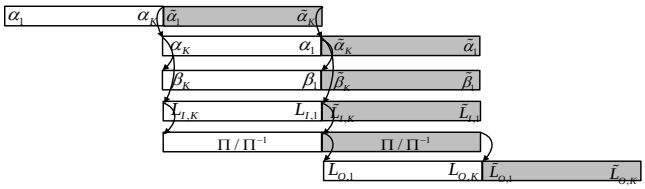



Fig. 8.Double buffering

Based on the proposed reverse address generator and the double buffering technique, the parallel decoding latency after implementation can be give as:

$$D' = (3N + 12)/4 + (3N/P)\cdot(I+1) + \Delta t, \qquad (15)$$

where $\Delta t$ is the latency brought by the FPGA modules such as RAMs, multipliers, FIFOs, modulo operation, and so forth. The value of $\Delta t$ depends on how the decoder is implemented.

## V. EXPERIMENT RESULTS

In this Section, the Testbed system and the results of parallel turbo decoder implementation are introduced. In order to meet the requirements of future broadcasting system, this Testbed is designed to support multi-Gbps decoding throughput. The structure of it can be found in Fig. 9.

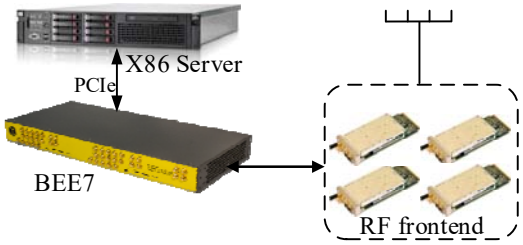



Fig. 9. Testbed system structure.

The X86 Server is the control center of this Testbed, which is connected via Peripheral Component Interconnect Express (PCIe) with BEE7. BEE7 is a programmable hardware platform used for algorithm exploration, research, prototyping and so on. Four Xilinx Virtex-7 FPGAs are allocated on this platform. With one FPGA processor, the throughput requirement cannot be met. BEE7 is linked with several RF frontends to build a MIMO transceiver.

As we know that higher parallelism means lower latency, it also takes more computing resources especially logic resources such as LUTs and Flip-Flops. For one FPGA in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TBC.2017.2704425, IEEE Transactions on Broadcasting



BEE7, the parallel level can reach to $P = 64$ with a latency of 23.2 us at 250 MHz clock rate where the iteration times is 8. Although the latency is quite low compared to lower parallelism, the throughput of this system is only 2.12 Gbps which is not enough. The throughput and latency comparison of different parallel levels are listed in Table I.

Table I. Throughput and latency comparison

| | This work | | [20] | |
|---|---|---|---|---|
| Iteration | 8 | | | |
| Block size | 6144 | | | |
| $P$ | 8 | 64 | 8 | 64 |
| Clock(MHz) | 250 | | | |
| Extraction time (us) | 18.4 | | | |
| Latency (us) | 56.8 | 23.2 | 69.6 | 24.8 |
| Iteration time (us) | 38.4 | 4.8 | 51.2 | 6.4 |
| Throughput (Gbps) | 6.92 | 2.12 | 2.82 | 0.99 |

The results above are obtained via ModelSim simulation after placement and routing. This simulation can measure how many clock cycles are needed for a whole decoding process. By some simple calculation, the latency and throughput can be calculated. It can be seen that when the parallel level is 8, which is 8 times lower than 64, then the latency is not 8 times larger. This is because the extraction of the received data takes a fixed amount of time. Moreover, since it takes much fewer resources when $P = 8$, 8 parallel turbo decoders can be put on a single FPGA at the same time, which makes the throughput reach to 6.92Gbps. Even though the throughput is low when $P = 64$ because only one decoder can be put on the FPGA, its good latency performance can still be used for the case of a strict latency requirement.

The implementation validity is evaluated by Integrated Logic Analyzer (ILA) of Xilinx. A fixed test block is stored in a block RAM. By capturing the output of the turbo decoder, the decoding results can be examined. As shown in Fig. 10, the original frame before turbo encoding is a square wave, and we can see that the output is a square wave that matches the original frame.

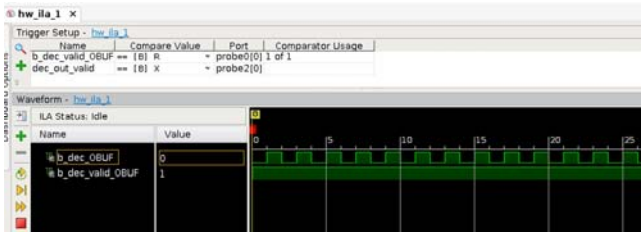



Fig. 10. Decoding results capture

For LTE standard, the maximum number of $C$ is $A(6143)$ in (4), the bit width $E$ cannot be larger than 35 and $F$ cannot be greater than 13. Therefore, it only takes 22 clock cycles to finish modulo operation. The latency and complexity comparison between this function and the division IP can be found in Table II.

Table II. Modulo operation comparison

| | Modulo function | | IP | |
|---|---|---|---|---|
| Latency (clock) | 22 | | 38 | |
| Complexity (1 module) | Slice LUT | Slice register | Slice LUT | Slice register |
| | 754 | 539 | 627 | 2407 |
| Complexity (64 modules) | 48256 | 34496 | 40128 | 154048 |

Table II shows that modulo function we designed can save nearly 3/4 slice registers compared to the IP from Xilinx although it takes a little more slice LUTs. It is significant that modulo function can save much more slice registers when the parallel level is high, e.g. $P = 64$, and it uses less clock cycles to complete the computation.

For LTE standard, block error rate (BLER) is used to test the decoder performance. The BLER of this decoder is evaluated via MATLAB simulation and ModelSim simulation. MATLAB simulation results are used as the reference of decoding BLER. The encoded frames with Gaussian white noise are first written into a test file and then read by Verilog test file. By Monte Carlo simulation, 1000 random frames for each signal-to-noise ratio (SNR) value, the BLER of different SNR can be obtained as shown in Fig. 11. Since the simulation is performed without rate mapping and modulation, it works well even at very low SNR. The purpose of this simulation is to make sure that the decoder implementation is working as expected. We can see that the BLER results of our FPGA parallel decoder are similar to its MATLAB theoretical simulation. The BLER is slightly higher because of the fixed point quantification error. Moreover, although the parallel decoder can increase system throughput and reduce latency, the BLER performance will be degraded.

There exist better ways to test the decoder, such as transfer the decoding results back to the server. By comparing the original bits and the decoded bits at the server side, the realtime block error rate (BLER) can be obtained. However, this part is not available at this moment and will be a part of our further research in the near future.

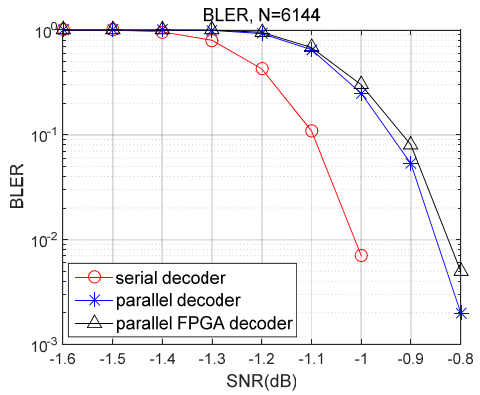



Fig. 11. BLER simulation results

## VI. CONCLUSION

Parallel turbo decoding is a practical way to increase the system throughput and reduce the latency in order to meet the requirements of future terrestrial broadcasting systems. In this paper, the implementation on FPGA of the parallel turbo decoder is introduced. A reverse address generator of interleaver/deinterleaver is proposed to reduce the processing time of each iteration and decrease latency further. The address generator uses recursion to generate the realtime address needed by the interleaver, which saves computing resources. A modulo function, that uses fewer clock cycles and logic resources compared to the Xilinx division IP, is designed to perform modulo operation. Moreover, in order to utilize the limited FPGA resources more efficiently, a double buffering technique is used to double the throughput in this parallel turbo decoder, which needs more storage space but reuses the logic resources.

The implementation of this decoder is accomplished on a Testbed system with 4 FPGA processors. On the one hand, by capturing the decoding results via Xilinx ILA, the validity of the parallel decoder is evaluated. On the other hand, the latency and BLER is tested by ModelSim simulation after placement and routing. The system decoding throughput is calculated based on the latency measured. Although the throughput of this Testbed is less than 10Gbps, which is the demanded requirement of next generation wireless communication systems, our research gives a clue that parallel turbo decoder can be implemented on FPGA and meet multi-Gbps throughput requirement at the same time and that the throughput can be further improved by using more hardware resources.

## VII. REFERENCE


[1]. S. I. Park, G. Lee, H. M. Kim, N. Hur, S. Kwon and J. kim, "ADT-Based UHDTV Transmission for the Existing ATSC Terrestrial DTV Broadcasting," *IEEE Trans. Broadcast.,* vol. 61, no. 1, pp. 105-110, March 2015.

[2]. T. Biatek; W. Hamidouche; J. F. Travers; O. Deforges, "Optimal Bitrate Allocation in the Scalable HEVC Extension for the Deployment of UHD Services." *IEEE Trans. Broadcast*., vol.PP, no.99, pp.1-16, Sep. 2016.

[3]. S. Saito et al., "8K Terrestrial Transmission Field Tests Using Dual-Polarized MIMO and Higher-Order Modulation OFDM," *IEEE Trans. Broadcast.,* vol. 62, no. 1, pp. 306-315, Mar. 2016.

[4]. D. Vargas, Y. J. D. Kim, J. Bajcsy, D. Gómez-Barquero, and N. Cardona, "A MIMO-Channel-Precoding Scheme for Next Generation Terrestrial Broadcast TV Systems." *IEEE Trans. Broadcast*., vol.61, no.3, pp. 445-456, 2015.

[5]. J. Calabuig, J. F. Monserrat and D. Gómez-Barquero, "5th generation mobile networks: A new opportunity for the convergence of mobile broadband and broadcast services," *IEEE Comm. Mag*., vol. 53, no. 2, pp. 198-205, Feb. 2015.

[6]. L. Dai, Z. Wang and Z. Yang, "Next-generation digital television terrestrial broadcasting systems: Key technologies and research trends," *IEEE Comm. Mag*., vol. 50, no. 6, pp. 150-158, June 2012.

[7]. J. P. Woodard and L. Hanzo, "Comparative study of turbo decoding techniques: An overview," *IEEE Trans. Veh. Technol.*, vol. 49, no. 6, pp. 2208-2233, Nov. 2000.

[8]. Y. G. Debessu, H. C. Wu, H. Jiang and S. Mukhopadhyay, "New Modified Turbo Decoder for Embedded Local Content in Single-Frequency Networks," *IEEE Trans. Broadcast.*, vol. 59, no. 1, pp. 129-135, March 2013.

[9]. I. A. Chatzigeorgiou, M. R. D. Rodrigues, I. J. Wassell and R. A. Carrasco, "Comparison of Convolutional and Turbo Coding for Broadband FWA Systems," *IEEE Trans. Broadcast*., vol. 53, no. 2, pp. 494-503, June 2007.

[10]. M. Brejza, L. Li, R. Maunder, B. Al-Hashimi, C. Berrou, and L. Hanzo, 20 years of turbo coding and energy-aware design guidelines for energy-constrained wireless applications," *IEEE Commun. Surveys Tuts*., vol. PP, no. 99, pp. 11, Jun. 2015.

[11]. Z. He, P. Fortier, and S. Roy, ``Highly-parallel decoding architectures for convolutional turbo codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst*., vol. 14, no. 10, pp. 11471151, Oct. 2006.

[12]. O. Muller, A. Baghdadi, and M. Jezequel, "From parallelism levels to a multi-ASIP architecture for turbo decoding," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst*., vol. 17, no. 1, pp. 92102, Jan. 2009.

[13]. "IEEE Standard for Air Interface for Broadband Wireless Access Systems," *IEEE Std 802.16-2012 (Revision of IEEE Std 802.16-2009)*, vol., no., pp.1-2542, Aug. 17, 2012.

[14]. LTE; *Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and Channel Coding*, ETSI, Sophia Antipolis, France, Feb. 2013.

[15]. L. Christodoulou, O. Abdul-Hameed, A. M. Kondoz and J. Calic, "Adaptive Subframe Allocation for Next Generation Multimedia Delivery Over Hybrid LTE Unicast Broadcast," *IEEE Trans. Broadcast.*, vol. 62, no. 3, pp. 540-551, Sep. 2016.

[16]. "5G Radio Access," *Ericsson White Paper*, Tech. Rep., June 2013.

[17]. P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," *IEEE Int. Conf. on Communications*, vol. 2, Seattle, WA, USA, June 1995, pp. 1009–1013.

[18]. D. Vogrig, A. Gerosa, A. Neviani, A. Graell I Amat, G. Montorsi, and S. Benedetto, "A 0.35-µm CMOS analog turbo decoder for the 40-bit rate 1/3 UMTS channel code," *IEEE J. Solid-State Circuits*, vol. 40, no. 3, pp. 753–762, 2005.

[19]. Q. T. Dong, M. Arzel, C. J. Jego, and W. J. Gross, "Stochastic decoding of turbo codes." *IEEE Trans. Signal Processing*, vol. 58, no. 12, pp. 6421–6425, Dec. 2010.

[20]. Z. Yan, G. He, We. He, S. Wang, Z. Mao, "High performance parallel turbo decoder with configurable interleaving network for LTE application", *Integration, the VLSI Journal*, Vol. 52, pp. 77-90, Jan. 2016.

[21]. R. G. Maunder, "A Fully-Parallel Turbo Decoding Algorithm," *IEEE Trans. Communications*, vol. 63, no. 8, pp. 2762-2775, Aug. 2015.

[22]. L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inf. Theory*, vol. 20, no. 2, pp. 284–287, Mar. 1974.

[23]. Seokhyun Yoon and Y. Bar-Ness, "A parallel MAP algorithm for low latency turbo decoding," *IEEE Communications Letters*, vol. 6, no. 7, pp. 288-290, July 2002.

[24]. A. Tarable, G. Montorsi, and S. Benedetto, "Mapping of interleaving laws to parallel turbo decoder architectures," *in Proc. 3rd Int. Symp. Turbo Codes and Related Topics*, Brest, France, pp. 153–156, Sep. 2003.

[25]. O. Y. Takeshita, "On maximum contention-free interleavers and permutation polynomials over integer rings," *IEEE Trans. on Info. Theory*, vol. 52, no. 3, pp. 1249-1253, March 2006.

[26]. Butler, J.T., Sasao, T., "Fast hardware computation ofx mod z". *Proc. 18th Reconfigurable Architectures Workshop (RAW 2011)*, Anchorage, Alaska, USA, pp. 294–267, May 2011.

[27]. Altera, "Baseband-C-RAN", https://www.altera.com/solutions/industry/wireless/applications/baseband/c-ran.html, Jun. 2016.

[28]. Hamid R. Sadjadpour, "Maximum a posteriori decoding algorithms for turbo codes", *Proc. SPIE 4045, Digital Wireless Communication II*, 73 July 26, 2000.