ISSN 2348-375X

Unique Journal of Engineering and Advanced Sciences

Available online: www.ujconline.net

Research Article

LUT BASED ARCHITECTURE AND BCJR ALGORITHM FOR ENERGY EFFICIENT AND HIGH SPEED TURBO DECODING IN WIRELESS SENSOR NETWORKS

Dhamayanthi N[1*], Marimuthu CN[2]

[1]PG student, Dept. of electronics and communication engineering,Nandha Engineering College, Erode-638052,Tamil nadu, India
[2] Project Guide & Dean, Dept. of electronics and communication Engineering, Nandha Engineering College, Erode-638052,Tamil nadu, India

Received: 25-12-2013; Revised: 24-01-2014; Accepted: 22-02-2014

*Corresponding Author: **Dhamayanthi. N**
PG student, Dept. of electronics and communication engineering, Nandha Engineering College,
Erode-638052, Tamil nadu, India Email: dhamayanthiece@gmail.com

ABSTRACT

Turbo codes have recently been considered for energy-constrained wireless communication applications, since they facilitate a low transmission energy consumption. However, in order to reduce the overall energy consumption, Look-Up- Table-Log-BCJR (LUT-Log-BCJR) architectures having low processing energy consumption are required. In this paper, we decompose the LUT-Log-BCJR architecture into its most fundamental Add Compare Select (ACS) operations and perform them using a novel low-complexity ACS unit. We demonstrate that our architecture employs an order of magnitude fewer gates than the most recent LUT-Log-BCJR architectures, facilitating a 71% energy consumption reduction. Compared to state-of-the-art Maximum Logarithmic Bahl-Cocke-Jelinek-Raviv (Max-Log-BCJR) implementations, our approach facilitates a 10% reduction in the overall energy consumption at ranges above 58 m.

Keywords: Turbo codes, LUT-Log-BCJR, Split codes, Algorithm, Wireless communication.

INTRODUCTION

In environmental monitoring WSNs for example, despite employing low transmission duty cycles and low average throughputs of less than 1 Mbit=s, the sensors' energy consumption[1-2] is dominated by the transmission energy Etx b (measured in J=bit), since they may be separated by up to 1 km. For this reason, turbo codes have recently found application in these scenarios[3-4], since their near-capacity coding gain facilitates reliable communication when using a reduced transmission energy Etxm b . Note however that this reduction in Etx b is offset by the turbo decoder's energy consumption Epr b , as well as the (typically negligible) energy consumption of the turbo encoder[4]. Therefore, turbo codes designed for energy constrained scenarios have to minimize the overall energy consumption (Etx b + Epr b). Recent Application-Specific Integrated Circuit (ASIC) based turbo decoder architectures[5-7] have been designed for achieving a high transmission throughput, rather than for a low transmission energy. For example, turbo codes have facilitated transmission throughputs in excess of 50 Mbit=s in cellular standards, such as the 3rd Generation Partnership Project 3GPP Long Term Evolution (LTE) and recent ASIC turbo decoder architectures have been designed for throughputs that are in excess of 100 Mbit=s[5,6]. This has been achieved by employing the Max-Log-BCJR turbo decoding algorithm, which is a low-complexity approximation of the optimal Logarithmic Bahl-Cocke-Jelinek-Raviv (Log-BCJR) algorithm[8]. The Max-Log-BCJR algorithm appears to lend itself to both high-throughput scenarios, as well as to the abovementioned energy-constrained scenarios. This is because a low turbo decoder energy consumption Epr b is implied by Max-Log-BCJR algorithm's low complexity. However, this is achieved at the cost of degrading the coding gain by 0.5 dB compared to the optimal Log-BCJR algorithm[9], increasing the required transmission energy Etx b by 10%. As we shall demonstrate in Section IV, this disadvantage of the Max-Log-BCJR outweighs its attractively low complexity, when optimizing the overall energy consumption Etx b + Epr b of sensor nodes that are separated by dozens of meters. This motivates the employment of the Look-Up-Table-Log- BCJR (LUT-Log-BCJR) algorithm[8] in energy-constrained scenarios, since it approximates the optimal Log-BCJR more closely than the Max-Log-BCJR and therefore does not suffer from

the associated coding gain degradation. However, to the best of our knowledge, no LUT-Log-BCJR ASICs have been specifically designed for energy-constrained scenarios. Previous LUT-Log-BCJR turbo decoder designs[10-13] were developed as a part of the on-going drive for higher and higher processing throughputs, although their throughputs have since been eclipsed by the Max-Log-BCJR architectures.

MODULE DESCRIPTION

In this module, it first load the data to input register which represents the encoded bits at the receiving side. These bits are load by using input buffers. In this block, we arrange the given bit pipelined architecture to encode the input bit by bit. This was done by registers to encode the input data bit by bit as in serially.

Split Code

In this module, it split the input encoded bits for 3 directions. Two is passed to LUT-Log-BCJR upper and lower blocks. Another direction was passed to phase shifter block. This was done by using demultiplexer in the receiver side after getting the encoded bits.

LUT-LOG-BCJR

A number of variants of the LUT-Log-BCJR architecture of have been proposed for further increasing the decoding throughput. For example, employs parallel repetitions of the blocks to "parallel-process" the schedule. Alternatively, employs a radix-4 variant, which processes two sets of or state metrics at a time

Decoded Bits

The proposed architecture can be readily applied to any LUT-Log-BCJR decoder, regardless of the corresponding convolution encoder parameters employed.

PERFORMANCE ANALYSIS

This motivates the novel architecture of which is specifically designed to have a minimal hardware complexity and hence a low energy consumption. We validate our architecture in the context of an LTE turbo decoder and demonstrate that it has an order of magnitude lower chip area, hence reducing the energy consumption of the state-of-the-art LUT-Log-BCJR implementation by 71%. Compared to state-of-the-art Max-Log-BCJR implementations, our approach facilitates a 10% reduction[14].

Turbo Codes

The basic idea of turbo codes is to use two convolutional codes in parallel with some kind of interleaving in between. Convolutional codes can be used to encode a continuous stream of data, but in this case we assume that data is configured in finite blocks - corresponding to the inter leaver size.

Encoding

The frames can be terminated - i.e. the encoders are forced to a known state after the information block as shown in figure 1. The performance depends on the weight distribution - not only the minimum distance but the number of words with low weight. Convolutional codes have usually been encoded in their feed-forward form, like (G1,G2)=(1+D2,1+D+D2). However, for these codes a single 1, i.e. the sequence ...0001000..., will give a codeword which is exactly the generator vectors and the weight of this codeword will in general be very low.

First Decoding

Decoding of error correcting codes is basically a comparison of the probabilities for different code words - or with convolutional codes, different paths in the trellis. When it talks about probabilities, it is always the probability of some event given a certain amount of information about this event.

PROJECT DESCRIPTION

Conventional LUT-LOG-BCJR Architecture

A turbo encoder comprises a parallel concatenation of two convolutional encoders, each of which has a structure comprising m number of memory elements, where m = 3 is used in the LTE encoders, for example. Each encoder converts an uncoded bit sequence b1 = fb1; jgN j=1 into the corresponding encoded bit sequence b2 = fb2; jgN j=1, where N is the length of the input bit sequences. Correspondingly, it depicts a turbo decoder in Figure 2, which comprises a parallel concatenation of two decoders.

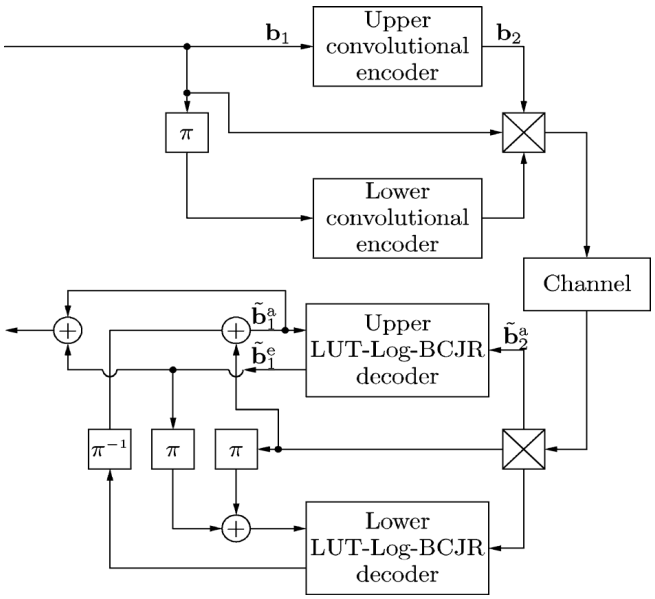

Figure 1: Conventional LUT BCJR Architecture

Note that in Figure 2, the energy consumption of the conventional LUT-Log-BCJR architecture cannot be significantly reduced by simply reducing the clock frequency, in order to meet the lower throughput demands of energy-constrained scenarios.

Proposed LUT-LOG-BCJR Architecture

In summary, efforts to slow down the conventional LUT-Log-BCJR architecture result in energy wastage, which cannot be avoided without completely redesigning the architecture

Decomposition of the LUT-Log-BCJR Algorithm

In this section, we propose a novel LUT-Log-BCJR architecture for energy-constrained scenarios, which avoids the wastage of energy that is inherent in the conventional architecture of it and its components to be merged.

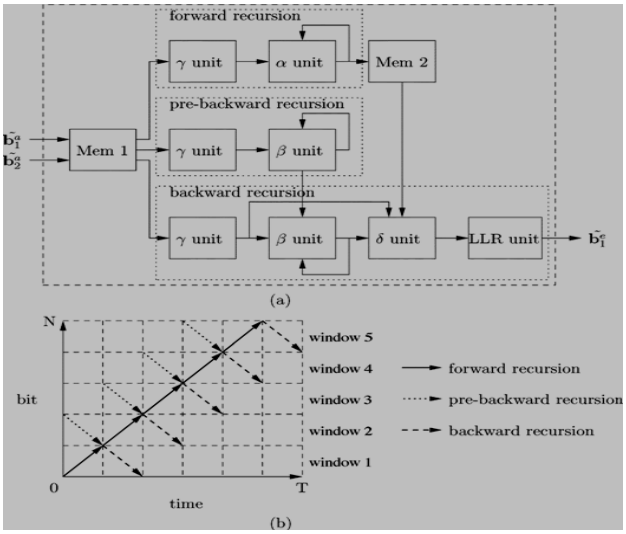

Figure 2: Parallel concatenation of two decoders

Table 1: Decomposition of operation

OP 1	SIMULTANEOUSLY CALCULATE (MAX (P,Q) AND |P-Q|)
OP 2	DETERMINE IF |P-Q |>0.75
OP 3	DETERMINE IF |P-Q |>0 OR |P-Q |> 2 DEPENDING ON THE OPERATION 2
OP 4	ADD MAX (P,Q) TO THE VALUE SELECTED FROM THE SET {0.75,0.5,0.25,0}

This produces an architecture comprising only a low number of inherently low complexity functional units, which are collectively capable of performing the entire LUT-Log-BCJR algorithm as shown in Table 5.1 by using split ACS operations. Further wastage is avoided, since the critical paths of our functional units are naturally short and equally lengthened, eliminating the requirement for additional hardware to manage them.

Proposed Energy Efficient LUT-Log-BCJR Architecture

Inspired by the analysis of it, the proposed energy-efficient LUT-Log-BCJR architecture is shown in Figure 5.3. Unlike conventional architectures, it does not use separate dedicated hardware for the three recursions shown. Instead, our architecture implements the entire algorithm using 2m ACS units in parallel, each of which performs one ACS operation per clock cycle. At the first register level, each ACS unit is paired with a set of general purpose registers R1, R2 and R3. These are used to store intermediate results that are required by the same ACS unit in consecutive clock cycles. For example, this allows the four ACS operations equivalent to a max* calculation to be performed in four consecutive clock cycles using a single ACS unit, as detailed in it[16].

The second register level comprises REG bank 1 and REG bank 2 of Figure 3, which are used to temporarily store the LUT-Log-BCJR variables between consecutive values of the bit index j during the recursions decoding processes. The REG bank 1 comprises registers for the a priori LLRs and dummy registers for the required LUT constants of Equation (2). Meanwhile, the sets of _, _ or _ metrics are stored in REG bank 2. The main memory stores all the required a priori LLR sequences and extrinsic LLR sequences during the decoding process and the _ state metrics from the previous window, which facilitates the processing of the entire LUT-Log-BCJR algorithm.

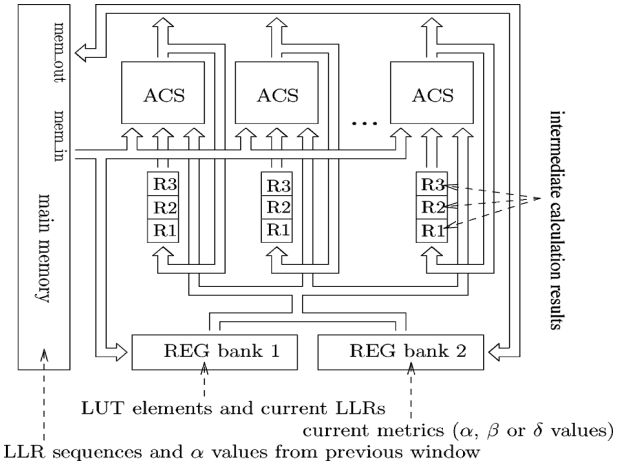

Figure 3: Energy-efficient LUT-Log-BCJR architecture and Novel ACS unit

The control signals of the ACS unit are provided by the the operation code O = 1011002 approximates the absolute difference between two operands, as required by Equation (2). Its result is equivalent to ~r = j~p □ ~qj for ~p _ ~q. However, for ~p < ~q, the result is given by ~p □ ~q. In the two's complement operand representation employing z = 2 fraction bits, this is equivalent to decrementing the binary representation of (~q □ ~p), which is equivalent to subtracting 2□z = 0:25.

Example controller design

The controller meets the timing diagram of Figure 4, which was designed to implement the sliding-window based LUT-Log-BCJR algorithm. To reduce the memory required for storing the _ state metrics of Equation (1), the sliding-window implementation performs the forward and backward recursions of the LUT-Log-BCJR

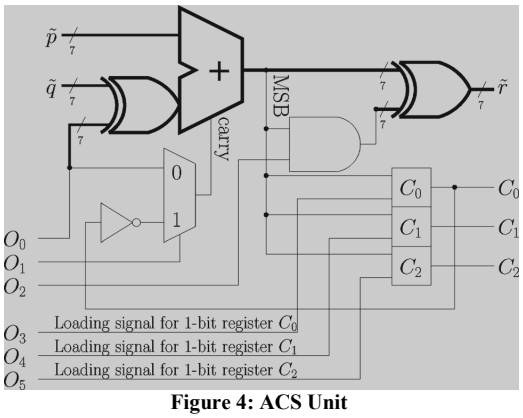

Figure 4: ACS Unit

Turbo Decoder Complexity And Energy Analysis

To analyze the complexity and the energy efficiency of the proposed LUT-Log-BCJR architecture, we implemented an

LTE turbo decoder using Taiwan Semiconductor Manufacturing Company (TSMC) 90 nm technology.

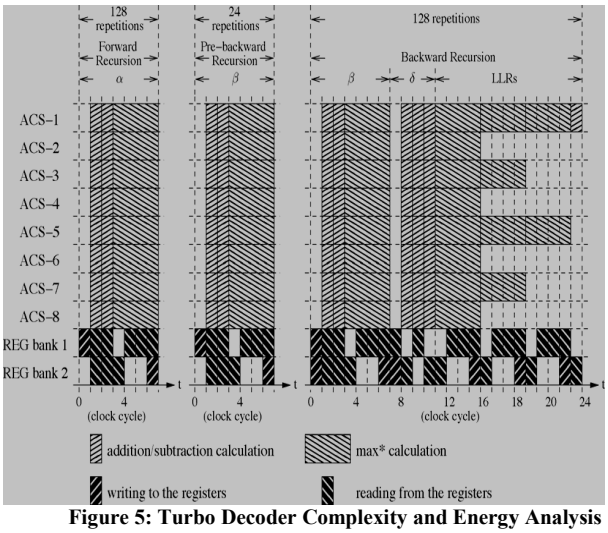

Figure 5: Turbo Decoder Complexity and Energy Analysis

The turbo decoder comprises four parts, namely a LUT-Log-BCJR decoder, an interleaver, a controller and the memory. The interleaver was implemented according to the latest low-complexity LTE interleaver designs[17,18].

SYSTEM COMPARISON

LUT-Log-BCJR architecture is within a tiny fraction of a decibel from that achieved by the ideal Log-BCJR algorithm. As a result, the LUT-Log-BCJR algorithm facilitates an overall energy consumption including the energy consumed during both transmission and decoding that is 10% lower than that of the Max-Log-BCJR at long transmission ranges.

Implemented Turbo Decoder

Furthermore, as discussed in Section I, the low complexity of the Max-Log-BCJR is achieved at the cost of requiring a 0.5 dB higher transmission energy per bit to achieve a BER of 10□4, as shown in Figure 6.

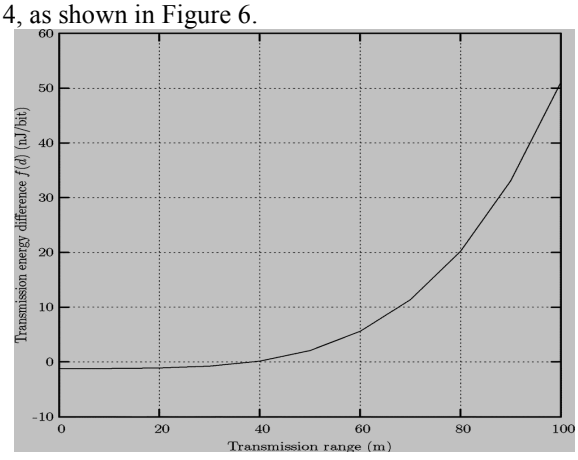

Figure 6: BER Performance of Various

Energy difference between LUT and Max log BCJR algorithm are compared here, where the energy consumption Epr^b of the turbo decoder as shown in Figure 6 is negligible compared to the transmission energy Etx^b required as shown in Figure 7 which will be discussed as follows:

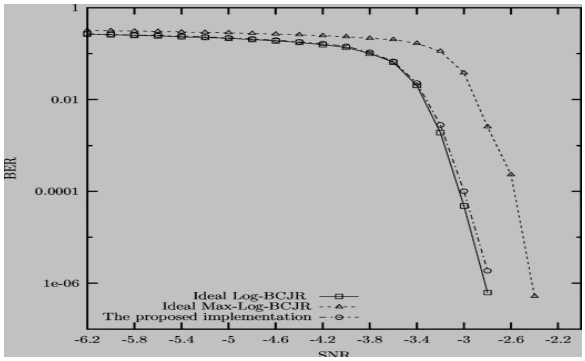

Figure 7: Energy consumption difference between the Max-Log-BCJR decoder of and the proposed architecture

SIMULATION RESULTS

a) BCJR SIMULATION

The BCJR algorithm is an algorithm for maximum a posteriori decoding of error correcting codes defined on trellises (principally convolutional codes).

b) FUNCTIONAL DIAGRAM

The original MAP decoding algorithm, which is referred to here as the BCJR algorithm (from the authors initials), requires the whole sequence to have been received before the decoding as shown in Figure 8.

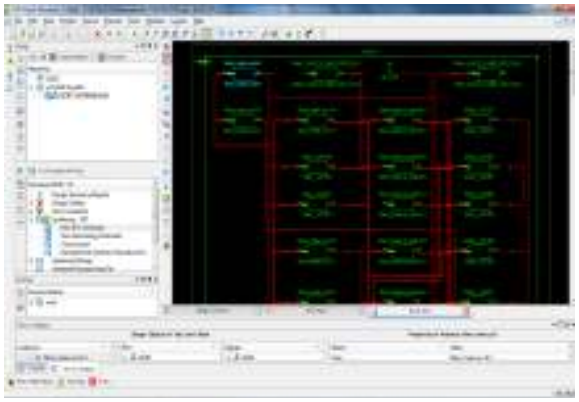

Figure 8: BCJR functional diagram

c) SIMULATION RESULT

The ISim GUI opens and loads the design. The simulator time remains at 0 until you specify a run time.

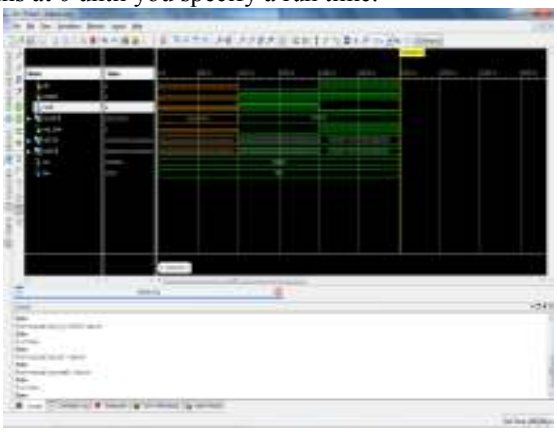

Figure 9: Simulation Result of BCJR

We call this new algorithm the sliding window BCJR algorithm (SWBCJR) as shown in Figure 9.

CONCLUSION

In this system, it is demonstrated that, upon aiming for a high throughput, conventional LUT-Log-BCJR architectures may have wasteful designs requiring high chip areas and hence high energy consumptions. However, in energy-constrained applications, achieving low energy consumption has a higher priority than having a high throughput. This motivated our low-complexity energy-efficient architecture, which achieves a low area and hence a low energy consumption by decomposing the LUT-Log-BCJR algorithm into its most fundamental ACS operations. It consists of an order-of-magnitude lower area than conventional LUT-Log-BCJR decoder implementations and an approximately 71% lower energy consumption of 0.4 nJ/bit/iteration. Compared to state of the art Max-Log-BCJR implementations, our approach facilitates a 10% reduction in the overall energy consumption at transmission ranges above 58 m. Furthermore,it is demonstrated that the implementation has a throughput of 1.03 Mb/s, which is appropriate for energy-constrained applications, such as in environmental monitoring WSNs.

SOURCE CODE

Here, the codings are implemented by using the XILINX tool.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY BCJR IS
PORT (
clk    : IN   STD_LOGIC;
enable  : IN   STD_LOGIC;
reset   : IN   STD_LOGIC;
d_in    : IN   STD_LOGIC_VECTOR(7 DOWNTO 0);
out_enb : OUT  STD_LOGIC;
rr     : OUT  STD_LOGIC_VECTOR(31 DOWNTO 0)
);
END BCJR;
-- Parallel Encoder (ARCHITECTURE)
-- automatically generated by program 'venomgen'
--
-- Date Thu May 14 12:37:59 1998
--
-- poly: x^32 + x^26 + x^23 + x^22 + x^16 + x^12 + x^11 + x^10 + x^8 + x^7
+
-- x^5 + x^4 + x^2 + x^1 + 1

SIGNAL xx       : STD_LOGIC_VECTOR(nn-1 DOWNTO 0);
-- copy of redundancy register for reading
--------------------------------------------------------------------------------
BEGIN
main:
PROCESS (clk,d_in,xx,reset)
VARIABLE xx_next   : STD_LOGIC_VECTOR(nn-1 DOWNTO 0);
VARIABLE uu        : STD_LOGIC_VECTOR(nn-1 DOWNTO 0);
BEGIN
-- combinatorial part:
FOR i IN 0 TO bw-1 LOOP
uu(i) := d_in(i) XOR xx(nn-bw+i);
END LOOP;
xx_next(0) := uu(6) XOR uu(0);
xx_next(1) := uu(7) XOR uu(6) XOR uu(1) XOR uu(0);
xx_next(2) := uu(7) XOR uu(6) XOR uu(2) XOR uu(1) XOR uu(0);
xx_next(3) := uu(7) XOR uu(3) XOR uu(2) XOR uu(1);
xx_next(4) := uu(6) XOR uu(4) XOR uu(3) XOR uu(2) XOR uu(0);
xx_next(5) := uu(7) XOR uu(6) XOR uu(5) XOR uu(S4) XOR uu(3) XOR
uu(1)
XOR uu(0);
xx_next(6) := uu(7) XOR uu(6) XOR uu(5) XOR uu(4) XOR uu(2) XOR
uu(1) ;
xx_next(7) := uu(7) XOR uu(5) XOR uu(3) XOR uu(2) XOR uu(0);
xx_next(8) := xx(0) XOR uu(4) XOR uu(3) XOR uu(1) XOR uu(0);
xx_next(9) := xx(1) XOR uu(5) XOR uu(4) XOR uu(2) XOR uu(1);
xx_next(10) := xx(2) XOR uu(5) XOR uu(3) XOR uu(2) XOR uu(0);
xx_next(11) := xx(3) XOR uu(4) XOR uu(3) XOR uu(1) XOR uu(0);
xx_next(12) := xx(4) XOR uu(6) XOR uu(5) XOR uu(4) XOR uu(2) XOR
uu(1)
XOR uu(0);
xx_next(13) := xx(5) XOR uu(7) XOR uu(6) XOR uu(5) XOR uu(3) XOR
uu(2)
XOR uu(1);
xx_next(14) := xx(6) XOR uu(7) XOR uu(6) XOR uu(4) XOR uu(3) XOR
uu(2) ;
xx_next(15) := xx(7) XOR uu(7) XOR uu(5) XOR uu(4) XOR uu(3);
xx_next(16) := xx(8) XOR uu(5) XOR uu(4) XOR
xx_next(27) := xx(19) XOR uu(7) XOR uu(5) XOR uu(4) XOR uu(1);
xx_next(28) := xx(20) XOR uu(6) XOR uu(5) XOR uu(2);
xx_next(29) := xx(21) XOR uu(7) XOR uu(6) XOR uu(3);
xx_next(30) := xx(22) XOR uu(7) XOR uu(4);
xx_next(31) := xx(23) XOR uu(5);
-- sequential part:
IF reset = '1' THEN
FOR i IN 0 TO nn-1 LOOP
rr(i) <= '0';
xx(i) <= '0';
END LOOP;
out_enb <= '0';
ELSIF clk = '1' AND clk'EVENT THEN
IF enable = '1' THEN
--read input
out_enb <= '1';
FOR i IN 0 TO nn-1 LOOP
rr(i) <= xx_next(i);
xx(i) <= xx_next(i);
END LOOP;
END IF;
END IF;
END PROCESS;
END rtl; -- OF BCJR
```

REFERENCES

1. Ang WP, Garg HK. A new iterative channel estimator for the log-MAP & max-log-MAP turbo decoder in Rayleigh fading channel, in Proc. Global Telecommun. Conf. 2001; 6: 3252.
2. Berrou C, Glavieux A, Thitimajshima P. Near Shannon limit error correcting coding and decoding: Turbo codes, in Proc. IEEE Int. Conf. Commun. 1993; 1064.
3. Bickerstaff MA, Garrett D, Prokop T, Thomas C, Widdup B, Zhou G, Davis LM, Woodward G, Nicol C, Yan RH. A unified turbo/Viterbi channel decoder for 3GPP mobile wireless in 0.18- m CMOS, IEEE J. Solid-State Circuits. 2002; 37(11): 1555.
4. Bickerstaff M, Davis C, Thomas D, Garrett C, Nicol. C. A 24 Mb/s radix-4 log-MAP turbo decoder for 3GPP-HSDPA mobile wireless, in Proc. IEEE Int. Solid-State Circuits Conf. 2003; 150.
5. Chargers C, Cather F, Engels M. Memory optimization of MAP turbo decoder algorithms, IEEE Trans. Very Large Scale Integer. (VLSI) Syst. 2001; 9(2): 305.

6. Corke P, Wark T, Jurdak R, Wen H, Valencia P, Moore D. Environmental wireless sensor networks, Proc. IEEE. 2010; 98: 11.
7. Hanzo L, Liew TH, Yeap BL, Tee R, Ng SX. Turbo Coding, Turbo Equalisation and Space-Time Coding. New York: Wiley. 2011.
8. Hanzo L, Woodard JP, Robertson P. Turbo decoding and detection for wireless applications, Proc. IEEE. 2007; 95(6): 1178.
9. Howard SL, Schlegel C, Iniewski K. Error control coding in low-power wireless sensor networks: When is ECC energy-efficient, EURASIP J. Wirel. Commun. Netw. 2006; 1.
10. Li FM, Lin CH, Wu AY. Unified convolutional/turbo decoder design using tile-based timing analysis of VA/MAP kernel, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 2008; 16(10): 1063.
11. Li L, Maunder RG, Al-Hashimi BM, Hanzo L. An energy-efficient error correction scheme for IEEE 802.15.4 wireless sensor networks, Trans. Circuits Syst. II. 2010; 57(3): 233.
12. Maseru G, Maze M, Piccinini G, Viglione F, Zamboni M. Architectural strategies for low-power VLSI turbo decoders, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 2002; 10(3): 279.
13. May M, Ilnseher T, When N, Raab W. A 150 Mbit/s 3GPP LTE turbo code decoder, in Proc. Design, Autom. Test in Euro. Conf. Exhib. 2010; 1420.
14. Robertson P, Hoeher P, Villebrun E. Optimal and sub-optimal maximum a posteriori algorithms suitable for turbo decoding," Euro. Trans. Telecommun. 1997; 8(2): 119.
15. Robertson P, Villebrun E, Hoeher P. A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain, in Proc. IEEE Int. Conf. Common. 1995; 1009.
16. Studer C, Benkeser C, Belfanti S, Huang Q. Design and implementation of a parallel turbo-decoder ASIC for 3GPP-LTE, IEEE J. Solid-State Circuits. 2011; 46: 8.
17. Wang. Z. High-speed recursion architectures for MAP-Based turbo decoders," IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 2007; 15(4): 470.
18. Wong C, Lee Y, Chang H. A 188-size 2.1 mm reconfigurable turbo decoder chip with parallel architecture for 3GPP LTE system, in Proc. Symp. VLSI Circuits. 2009; 288.

Source of support: Nil, Conflict of interest: None Declared