

# Implementation of Convolutional Encoder by XOR-Free Approach

Pankaj Kumar<sup>1</sup>, Deepak Kumar<sup>2</sup>

M.Tech Scholar Department of Electronics and Communication Engineering, Vidhyapeeth Institute of Science & Technology, Bhopal, India<sup>1</sup>

Assistant Professor, Department of Electronics and Communication Engineering, Vidhyapeeth Institute of Science & Technology, Bhopal, India<sup>2</sup>

**Abstract:** Due to the excellent error control performance in many communication systems Convolution encoder and Viterbi decoder are widely used. In coding techniques the number of symbols in the source encoded message is increased in a controlled manner in order to facilitate two basic demand at the receiver one is Error detection and other is Error correction. The amount of error detection and correction required and its efficiency depends on the signal to noise ratio (SNR). The technology based on Non Line of Sight (NLOS) ability to make the system very attractive for users, but there will be a little higher BER at low SNR. Coding is a technique where redundancy is added to original bit sequence to increase the reliability of the communication. This paper presents a review on hardware implementation of Convolution Encoder with power efficient architecture. The results of this architecture will decrease the dynamic power and HW cost with lower design complexity as comparing to conventional method.

**Keywords:** Automatic Repeat Request (ARQ), Forward Error Correction (FEC), Convolutional Code (CC), Non Line of Sight (NLOS).

## I. INTRODUCTION

This In a communication system, error detection and correction mechanisms are vital and numerous techniques exist for reducing the consequence of bit-errors and trying to make sure that the receiver eventually gets an error free version of the packet. The most important technique used are fault detection with Automatic Repeat Request (ARQ), Forward Error Correction (FEC) and hybrid forms of ARQ and FEC. This development focus on FEC techniques. Forward Error Correction (FEC) is an error control method for data transmission by adding redundant data to its messages to improve the capacity of a channel. This redundant data allows the receiver to detect and to correct a certain number of errors without asking the encoder to re-transmit more additional data. The process of adding this redundant information is known as channel coding. Mainly there are two major kinds of channel coding: block codes like Reed – Solomon coding and Convolution coding. Block codes work with fixed length blocks of code. Convolution codes deal with data sequentially. Block codes become very complex as their length increases and are therefore harder to implement. Convolution codes in comparison to block codes are less complex and therefore easier to implement.

### Convolutional Code (CC)

In Non Line of Sight (NLOS) Communication such as Mobile Wi-Max of CDMA part, the CC is the only required coding scheme. Its computations depend not only on the existing set of input symbols it also depends up on some of the previous used input symbols. A lattice description is used for convolution encoding that show

relation how each possible input to the encoder impacts on the output in shift register. Viterbi algorithm is used for decoding. In communication, a convolution code is a type of error-correcting code in which

Each m-bit information symbol (each m-bit string) to be encoded is transformed into an n-bit symbol, where  $m/n$  is the code rate ( $n \geq m$ ).

The transformation is a function of the last information symbols, where  $k$  is the constraint length of the code.

The convolutional codes are defined by three parameters which are as follow:

(a) Rate: Ratio of the number of input bits to the number of output bits. In this example, rate is  $1/2$  which means there are two output bits for each input bit.

(b) Constraint length: The number of delay elements in the convolution coding for example with  $k = 3$ , there are two delaying elements.

(c) Generator polynomial: Wiring of the input sequence with the delay elements to form the output. For example, generator polynomial is  $[7,5]_8 = [111,101]_2$ . The output from the 78 = 1112 arm uses the XOR of the current input, previous input and the previous to previous input. The output from 58 = 1012 uses the XOR of the current input and the previous to previous input.

Rate =  $1/2$

Constrain length,  $k=3$

Generator polynomial is  $[7, 5]_8 = [111,101]_2$

Generic Methods for Decoding Convolution code

There are many different decoding techniques for Convolution codes which are Feedback decoding, sequential decoding and maximum prospected decoding.

1. Threshold decoding – This decoding is called majority logic decoding. It is successfully applied only on definite classes of code. It applies to channel having a slight to good SNR. It is far away from optimal because of its inferior in bit error performance.

2. Sequential decoding –This decoding is sub optimal. This decoding has better performance than the previously used method. Virtually independent from the length of the particular code is the advantage of it. Unpredictable decoding latency & variable decoding time is its drawback. Also, it requires a large memory.

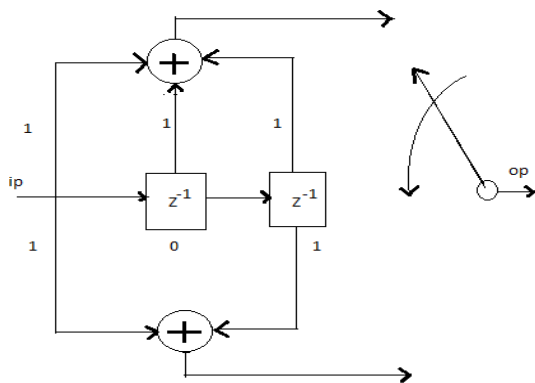


Figure 1: Convolutional code with Rate 1/2, K=3, Generator Polynomial octal

3. Viterbi decoding - It is optimal algorithm for decoding of Convolution code. It is the dominant technique for Convolution codes. It has advantages like satisfactory bit error performance, low cost, fixed decoding time. The most extensively decoding algorithms for Convolution codes is Viterbi code proposed in 1967. It is very useful method for forward error correction. In many wireless communication systems like IEEE 802.11a/g, Wi-Max, WCDMA and GSM to improve capacity of communication channel it is widely used. Due to high demand of the portable wireless communication devices by user. So need of high speed viterbi decoder increasing.

II. METHOD

Viterbi algorithm is the most possibility decode algorithm of convolution code. Viterbi decoder means the VLSI implementation of Viterbi algorithm. In the area of communication, convolution code is very popular, so how to improve the performance and reduce the power and area of the decoder is important. In the other hand, different protocols use different convolution code and varied applications have different requirement for throughput, area and power. So design of reusable Viterbi decoder is important, too. This decoder adopted the Process Element (PE) technique, which made it easy to adjust the throughput of the decoder by growing or falling the number of PE. By the method of Same Address Write Back (SAWB), we compact the number of register to half

in disparity with the method of ping-pong. This decoder supported punctured convolution code and was data-driven, which means the circuit was able to work under different data rate and avoid those invalid operations. PE Process Element, was one of the most popular architecture in digital signal processing .The PE architecture of Viterbi decoder has been introduced. Fig1 is the basic structure of PE in Viterbi decoder. Which consists of PMU, ACSU, BMU and LRU?

Convolution operation is realized using a deterministic finite state machine (DFSM). Its hardware implementation requires a combinational circuit and memory elements. The discrete convolution for the encoded sequence (C<sub>j</sub>) with the generator sequences (G<sub>i</sub>) by the following equation:

Further shift register (SR) based realization of (1) for encoded sequence (C<sub>j</sub>β) depends upon the length (L) of SR, the present input I<sub>j</sub> and M previous input blocks [I<sub>j</sub>1,....., I<sub>j</sub>-M] to yield (2).

$$C_{j\beta} = \sum_{\alpha=1}^n [ \sum_{l=0}^M I_{j-l} \alpha g_{\alpha\beta}^{(l)} ] \quad (2)$$

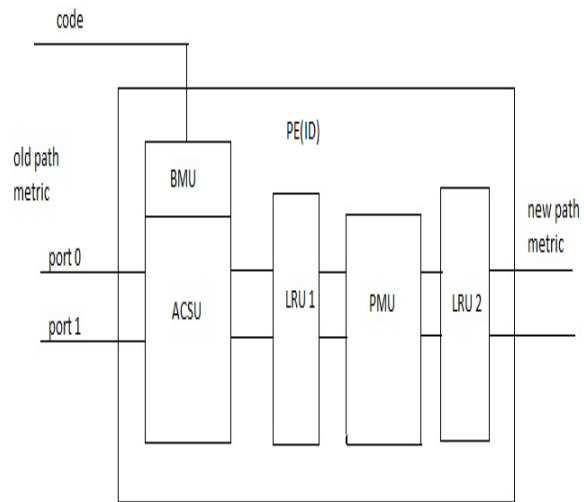
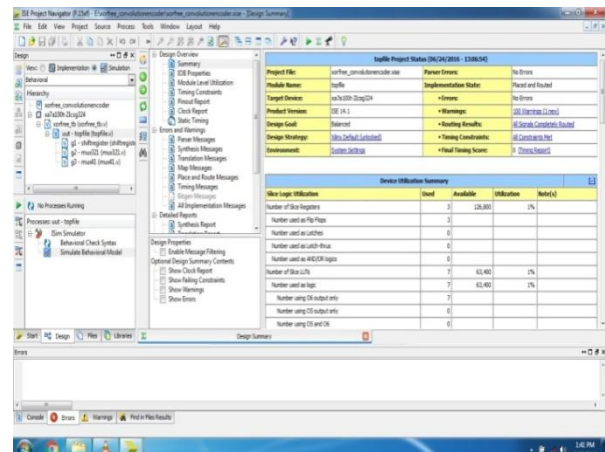
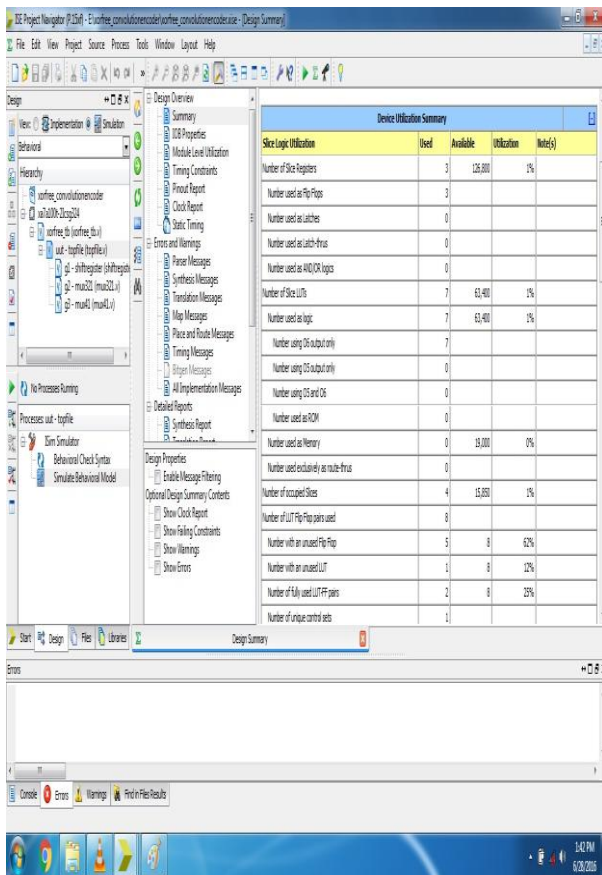


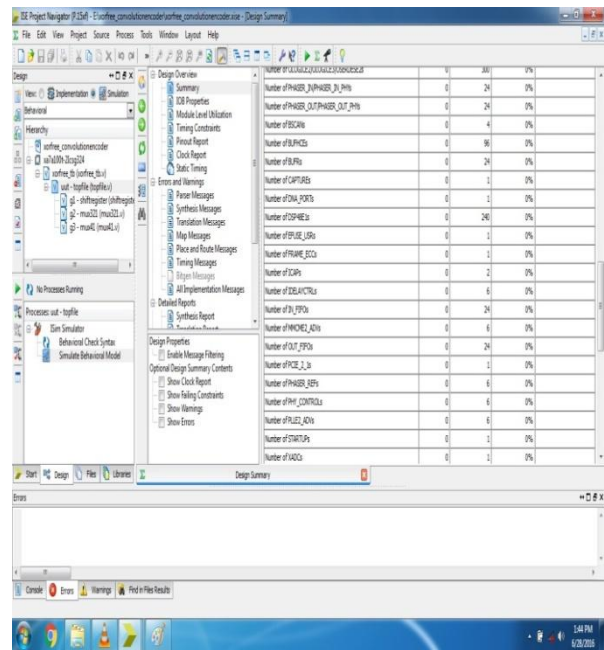
Figure 2: Structure of PE

III.RESULT ANALYSIS

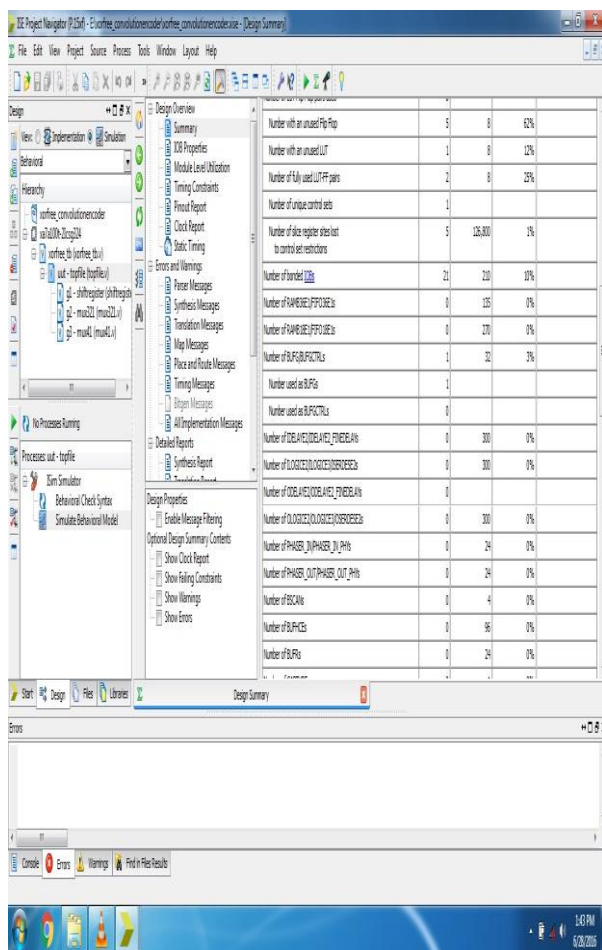




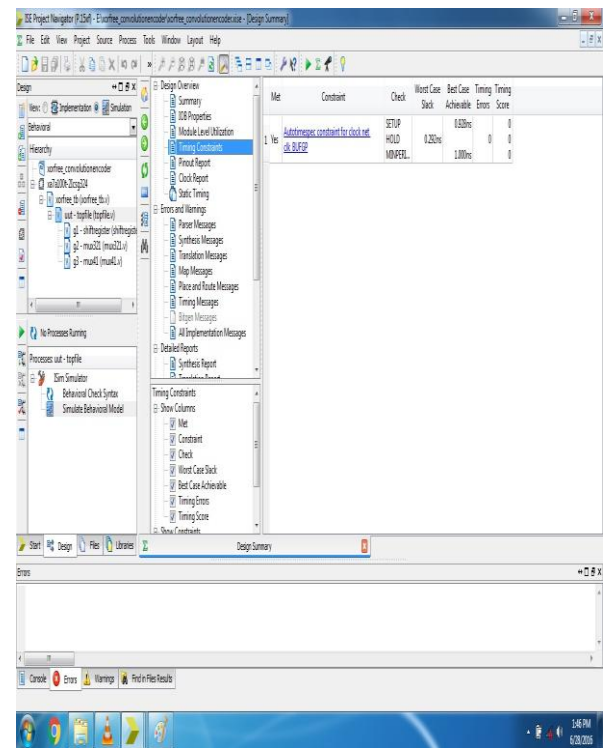
Device Utilization	Used	Available	Utilization	Note(s)
Number of Slice Registers	3	126,800	1%	
Number used as Flip-Flops	3			
Number used as Latches	0			
Number used as LUT-FFs	0			
Number used as 4KDCS Klags	0			
Number of Slice LUTs	7	61,401	1%	
Number used as logic	7	61,401	1%	
Number using 36-subopt only	0			
Number using 28-subopt only	0			
Number using 16-subopt only	0			
Number used as RAM	0			
Number used as Memory	0	18,000	0%	
Number used exclusively as multi-ports	0			
Number of occupied Slices	4	15,653	1%	
Number of LUT-Flop pairs used	8			
Number with an unused Flop	5	8	62%	
Number with an unused LUT	1	8	12%	
Number of fully used LUT-Flop pairs	1	8	12%	
Number of unique control acts	1			



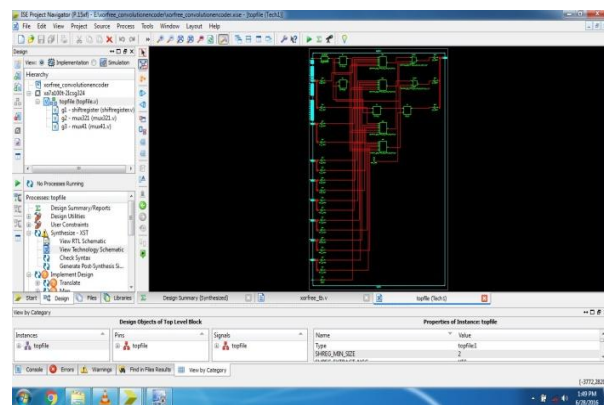
Design Overview	Value	Unit	Percentage
Number of PHASE_IN_PHASE_IN_2bits	0	24	0%
Number of PHASE_OUT_PHASE_OUT_2bits	0	24	0%
Number of BSC20s	0	4	0%
Number of BSC20s	0	96	0%
Number of BSC20s	0	24	0%
Number of BSC20s	0	1	0%
Number of BSC20s	0	1	0%
Number of BSC20s	0	240	0%
Number of BSC20s	0	1	0%
Number of BSC20s	0	1	0%
Number of BSC20s	0	2	0%
Number of BSC20s	0	6	0%
Number of BSC20s	0	24	0%
Number of BSC20s	0	6	0%
Number of BSC20s	0	24	0%
Number of BSC20s	0	1	0%
Number of BSC20s	0	6	0%
Number of BSC20s	0	6	0%
Number of BSC20s	0	1	0%
Number of BSC20s	0	1	0%
Number of BSC20s	0	2	0%



Design Overview	Value	Unit	Percentage
Number with an unused Flop	5	8	62%
Number with an unused LUT	1	8	12%
Number of fully used LUT-Flop pairs	1	8	12%
Number of unique control acts	1		
Number of slice register sites not to control set restrictions	0	126,800	1%
Number of bonded I/Os	21	210	10%
Number of PHASE_IN_PHASE_IN_2bits	0	24	0%
Number of PHASE_OUT_PHASE_OUT_2bits	0	24	0%
Number of BSC20s	0	4	0%
Number of BSC20s	0	96	0%
Number of BSC20s	0	24	0%



Met	Constraint	Check	Word Case	Best Case	Timing	Timing
1	Autotimese constant for clock out	Setup	0.030ns	0	0	0
	HOLD	0.320ns	0	0	0	0
	MINFREQ	1.000ns	0	0	0	0



Design Overview	Value	Unit	Percentage
Number of PHASE_IN_PHASE_IN_2bits	0	24	0%
Number of PHASE_OUT_PHASE_OUT_2bits	0	24	0%
Number of BSC20s	0	4	0%
Number of BSC20s	0	96	0%
Number of BSC20s	0	24	0%

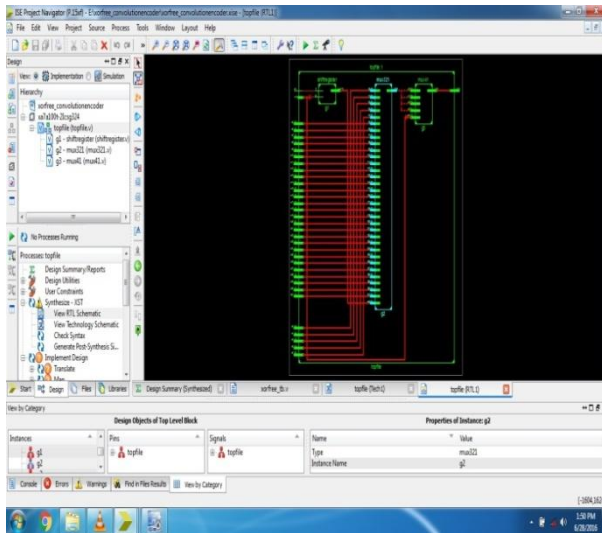


Table 1: - Comparison of Resource Utilization for the Architectures

Parameter	Conventional Encoder (K=1/3)	Reduced ROM XOR-FREE
FFs	12	08
LUTs	04	04
LUT-FF Pairs	08	03
BUFG/BUFGCTRLs	01	01
F <sub>max</sub> (MHz)	544.0	589.2

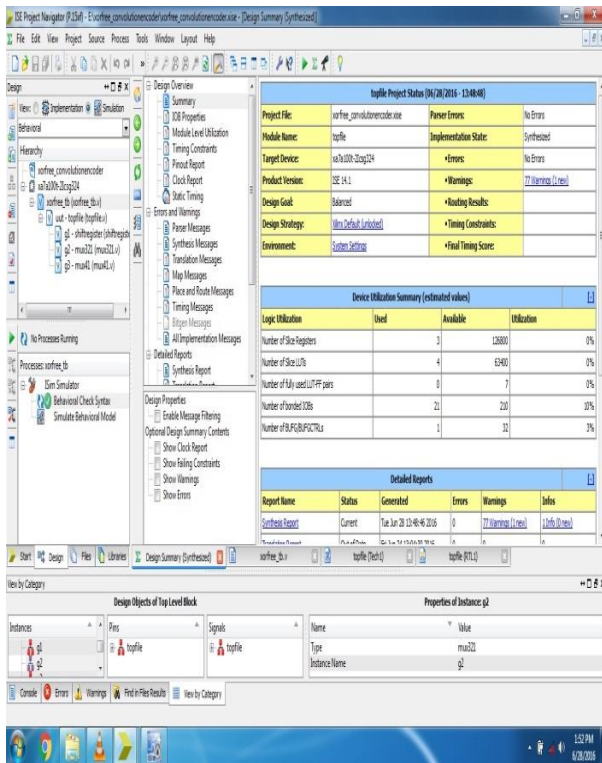


Table 2:- Comparison of Resource Utilization for the Architectures

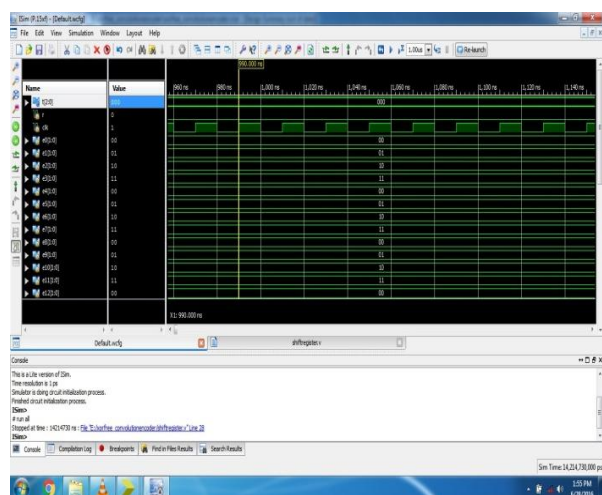
Parameter	Conventional Encoder (K=1/2)	Reduced ROM XOR-FREE
FFs	10	09
LUTs	03	03
LUT-FF Pairs	10	08
BUFG/BUFGCTRLs	01	01
F <sub>max</sub> (MHz)	420.6	556.5

IV. CONCLUSION

The improvement in delay can be done by the use of Convolution encoder Vedic multiplier. Which provide faster speed than the normal convolution encoder? Parallel generation of partial products and eliminates unwanted multiplication steps by this. A fast multiplication process and achieves a significantly less computational complexity over its conventional counterparts is allow by this algorithm. Viterbi decoder using parallel processing improves processing speed than normal viterbi decoder because the decoder do not need to wait trace back. It means trace back and decoder can simultaneously work. the design complexity will get reduced for inputs of large number of bits and will a provide faster speed. It will also used to design a PN sequence generator and spread spectrum modulation to improve the utilization of bandwidth.

REFERENCES

- [1] G. Purohit, K. S. Raju, V. K. Chaubey, "A New XOR-Free Approach for Implementation of Convolutional Encoder" IEEE EMBEDDED SYSTEMS LETTERS, VOL. 8, NO. 1, MARCH 2016
- [2] J. Dielissen, Eindhoven, N. Engin, S. Sawitzki, and K. van Berkel, "Multistandard FEC decoders for wireless devices," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 55, no. 3, pp. 284-288, Mar. 2008.
- [3] J. Hagenauer, "Forward error correcting for CDMA systems," in Proc. Int. Symp. Spread Spectr. Tech. Appl. Proc., Mainz, Sep. 1996, pp.566-569.



- [4] A. J. Viterbi, "Convolutional codes and their performance in communication systems," *IEEE Trans. Comm. Technol.*, vol. 19, no. 5, pp.751–772, Oct. 1971.
- [5] Y.Yibin, K.Roy, andR.Drechsler,"PowerconsumptioninXORbased circuits," in*Proc. ASP-DAC*, Jan.1999, pp. 299–302.
- [6] R. Pasko, P. Schaumont, V. Derudder, S.Vernalde, and D. Durackova,"A new algorithm for elimination of common subexpressions,"*IEEE Trans. Comput. Des. Integr. CircuitsSyst.*, vol. 18, no. 1, pp. 58–68, Jan. 1999
- [7] C. Huang, J. Li, and M. Chen, "Optimizing XOR-based codes," U.S. Patent 8209577 B2, Jun. 26, 2012.
- [8] H. J. Kang and I. C. Park, "A high-speed and low-latency Reed–Solomon decoder based on a dual-line structure," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, May 2002, vol. 3, pp. 3180–3183
- [9] H. Samuelli, "An improved search algorithm for the design of multi-plierless FIR filters with powers-of-two coefficients,"*IEEE Trans. CircuitsSyst.*, vol. 36, pp. 1044–1057, July 1989.
- [10] M. Potkonjak, M. B. Shrivasta, and P. A. Chandrakasan, "Multiple constant multiplication: Efficient and versatile framework and algorithms for exploring common subexpression elimination,"*IEEE Trans.Computer-Aided Design*, vol. 15, pp. 151–161, Feb. 1996.
- [11] M. Mehendale, S. D. Sherlekar, and G. Vekantesh, "Synthesis of multi-plierless FIR filters with minimum number of additions," in *Proceedings of the 1995 IEEE/ACM International Conference on Computer-Aided Design*. Los Alamitos, CA: IEEE Computer Society Press, 1995, pp.668–671.
- [12] R. I. Hartley, "Sub expression sharing in filters using canonic signed digit multipliers," *IEEE Trans. Circuits Syst. II*, vol. 43, pp. 677–688,Oct. 1996.
- [13] A. G. Dempster and M. D. Mcleod, "Use of minimum-adder multiplier blocks in FIR digital filters," *IEEE Trans. Circuits Syst. II*, vol. 42, pp.569–577, Sept. 1995.
- [14] D. R. Bull and D. H. Horrocks, "Primitive operator digital filters," *Proc. Inst. Elect. Eng.* , vol. 138, pt. G, no. 3, pp. 401–411, June 1991.
- [15] Y. C. Lim and S. R. Parker, "Discrete coefficient fir digital filter design based upon an LMS criteria," *IEEE Trans. Circuits Syst.* , vol. CAS-30, pp. 723–739, Oct. 1983.