Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2016, Article ID 9128683, 8 pages
http://dx.doi.org/10.1155/2016/9128683

Research Article

XOR-FREE Implementation of Convolutional Encoder for Reconfigurable Hardware

Gaurav Purohit,[1] Kota Solomon Raju,[2] and Vinod Kumar Chaubey[1]

[1]*Department of EEE, BITS-Pilani, Pilani, Rajasthan 333031, India*
[2]*Digital System Group, CEERI, Pilani, Rajasthan 333031, India*

Correspondence should be addressed to Gaurav Purohit; gp.bits@gmail.com

Received 30 September 2015; Accepted 11 January 2016

Academic Editor: Miriam Leeser

This paper presents a novel XOR-FREE algorithm to implement the convolutional encoder using reconfigurable hardware. The approach completely removes the XOR processing of a chosen nonsystematic, feedforward generator polynomial of larger constraint length. The hardware (HW) implementation of new architecture uses Lookup Table (LUT) for storing the parity bits. The design implements architectural reconfigurability by modifying the generator polynomial of the same constraint length and code rate to reduce the design complexity. The proposed architecture reduces the dynamic power up to 30% and improves the hardware cost and propagation delay up to 20% and 32%, respectively. The performance of the proposed architecture is validated in MATLAB Simulink and tested on Zynq-7 series FPGA.

1. Introduction

Data transmission reliability and stringent QoS requirements of modern 3GPP, 3GPP2, and LTE standards over unreliable wireless channel require efficient, cost-effective forward error correction (FEC) codes for the mobile equipment [1–4]. The usage and application of such FEC depend mainly on their error correcting capabilities and power efficient implementation. Convolution codes are preferred over block codes due to their economical soft decoding capability and, moreover, they yield higher coding gain [5]. While the error correction capabilities are related to polynomial strength, they are also designed to achieve higher free distance [6]. The conventional convolutional encoder is usually realized with shift register (SR) using delay elements and modulo-2 adders (XOR gates). The key operation in the process of convolution is multiplication, which is implemented using shifts and additions [7]. The addition operation dominates the complexity in comparison to the shift operations and consumes a significant amount of dynamic power while encoding. Hence, the optimization of adder utilization becomes a key factor while implementing with reconfigurable hardware. Few attractive heuristic approaches are reported in the literature to minimize adder count by eliminating the redundant computations [8–13]. Such approaches are mostly based on Common Subexpression Elimination (CSE) method which depends upon the commonalities of the used polynomials involving modulo-two adders and consumes more power.

Convolution codes find their extensive usage as channel codes in popular wireless standards like 3GPP-WCDMA, 3GPP2-CDMA2000, LTE, and IEEE-802.11 and software defined radio and cognitive radio applications. They are the key building blocks in many powerful concatenated codes, such as Turbo codes (parallel), Woven codes (serial) [14], and Ungerboeck Code (a.k.a. Trellis Coded Modulation (TCM)) [15]. This paper describes a new algorithm to implement an XOR-FREE architecture of a chosen generator polynomial having constraint length ($K \leq 9$) for popular wireless standards. The approach reduces the standard polynomial into a LUT comprising parity bits. The previous state of the encoder and the input bit jointly forms the decoding addresses of such LUT. The ROM based architecture provides ease in FPGA implementation with a powerful add-on feature of dynamic reconfiguration. The FPGAs have large resources of logic gates and ROM blocks. Hence, the hardware realization of this architecture is inferred in the desired approach and achieves greater flexibility. The paper is organized into five

sections. In Section 2, the preliminary of the convolutional code is presented with two different approaches. Section 3 covers in depth a new algorithm with an example of GSM-900 based standard generator polynomial. Section 4 states the results and discussion, whereas Section 5 provides a conclusion.

2. Preliminary

For realization of convolutional code, the theory of groups and finite fields is used. Two approaches are found in the literature. The first approach is taken by Massey [16] and McEliece and Onyszchuk [17]. The method defines the code first, as a k-dimensional subspace of an n-dimensional vector space over a suitable field, and then defines the encoder as a $k \times n$ matrix whose rows are a basis for the code. Such a convolutional code can be described by an "infinite matrix, G" as shown in (1). This matrix depends on $k \times n$, $\{G_i\}$, $i = 0 \cdots M$, submatrices:

$$G = \begin{pmatrix} G_0 & G_1 & \cdots & G_M & 0_{k\times n} & \cdots \\ 0_{k\times n} & G_0 & \cdots & G_{M-1} & G_M & \cdots \\ \vdots & \ddots & \ddots & \ddots & \vdots & \ddots \\ & \vdots & 0_{k\times n} & G_0 & G_1 & \\ & & \vdots & \ddots & G_0 & \ddots \\ & \vdots & & & 0_{k\times n} & \ddots \\ \vdots & & & & & \ddots \end{pmatrix}. \quad (1)$$

Here, k is the constraint length and G are the generator sequences of the encoder. $I_j = [I_{j1}, \ldots, I_{jk}]$ are the jth block of k information bits. $C_j = [C_{j1}, \ldots, C_{jn}]$ are the block of n coded bits at the output. This is similar to the block coding, $C = IG$, as shown in

$$(C_0, C_1, C_2, \ldots) = \{I_0, I_1, I_2, \ldots\} \cdot \begin{pmatrix} G_0 & G_1 & \cdots & G_M & 0_{k\times n} & \cdots \\ 0_{k\times n} & G_0 & \cdots & G_{M-1} & G_M & \cdots \\ \vdots & \ddots & \ddots & \ddots & \vdots & \ddots \\ & \vdots & 0_{k\times n} & G_0 & G_1 & \\ & & \vdots & \ddots & G_0 & \ddots \\ & \vdots & & & 0_{k\times n} & \ddots \\ \vdots & & & & & \ddots \end{pmatrix}. \quad (2)$$

The second approach is based on Forney's approach which defines the encoder as a k-input, n-output linear, sequential

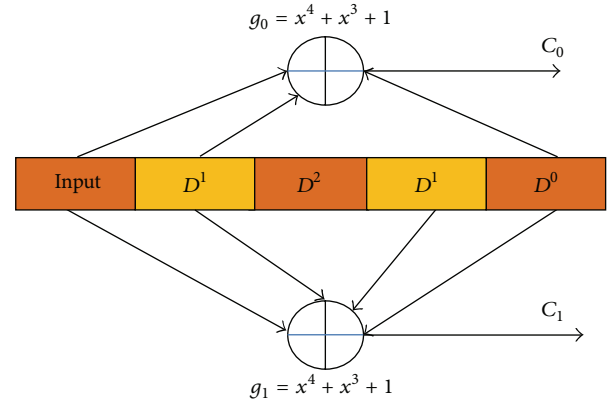

Figure 1: Conventional convolution encoder architecture for GSM-900 standard with generator polynomials $g_0(x) = (31)_8$ and $g_1(x) = (33)_8$ [6].

circuit [18, 19] and can be realized by shift register as follows:

$$C_J = \sum_{I=0}^{M} I_{j-I} G_I, \quad (3)$$

$$C_j = \left[C_{j1}, \ldots, C_{jn}\right] = \left[\sum_{I=0}^{M}\sum_{\alpha=1}^{k} I_{j-I}\alpha g_{\alpha 1}^{(I)}, \ldots, \sum_{I=0}^{M}\sum_{\alpha=1}^{k} I_{j-I}\alpha g_{\alpha n}^{(I)}\right], \quad (4)$$

$$C_{j\beta} = \sum_{\alpha=1}^{k}\sum_{I=0}^{M} I_{j-I}\alpha g_{\alpha\beta}^{(I)}, \quad (5)$$

where $g_{\alpha\beta}^{(I)}$ are elements of matrix G_I. Equation (3) can be expanded explicitly in n components, that is, $C_{j1}, C_{j2}, C_{j3}, \ldots, C_{jn}$ as shown in (4). Obviously, shift register of length L will have M^L different internal configurations or states. The behavior of such convolutional coder depends upon the present input I_j and M previous input blocks $I_{j-1}, I_{j-2}, \ldots, I_{j-M}$. $C_{j\beta}$ can be calculated by memorizing M input values in shift registers as expressed in (5). Here, one register $\alpha \in 1 \cdots k$, for each k bit of the input and memories for which $g_{\alpha\beta}^{(I)} = 1$ are connected to adder β. Such realization of convolutional encoder can be captured by a Deterministic Finite State Machine (DFSM) or deterministic automaton [20].

Using (5) shift register based realization of the convolutional encoder for a used generator polynomial can be obtained as shown in Figure 1. A combination of sequential logic and combinational logic is required in the shift register based realization. The convolution encoder as a sequential machine can be presented as an FSM with the finite states. While working for optimal synthesis of sequential circuit, the approaches like state minimization and the state assignment exist, whereas for the combinational circuit, the logic minimization based on different topologies, namely, AND-OR or AND-XOR decomposition of functions, is available.

The state minimization recognizes the equivalent states of a machine and then minimizes the internal states of a machine [21]. Such popular state minimization techniques are row based, state implication table based, and heuristic, which follows the thumb rule of state equivalency [22] while minimizing. These methods may not lead to a fruitful result as reducing the number of states may not always reduce the hardware. This happens because the number of eliminating states may not reduce the total state count by a power of two, since the convolutional encoder is a state minimal machine and hence the state minimization techniques will not reduce its hardware, whereas the other front is state assignment or state encoding. It is assigning a binary representation to the control states. The state assignment problem can also be viewed as decomposition of an FSM. Some of these popular approaches towards area minimization are Decomposition and Factorization [23], Modify and Restore [24], and Decomposition as Constrained Covering [25, 26]. With different viable approaches towards the state assignment, the heuristic seems to be the most promising [21]. The optimal state encoding methods require state assignment tools like NOVA [27] for area minimization of PLA implementations; JEDI [28] for multilevel implementations; Decomp; One Hot [23], and a lot more. These approaches are not promising while implementing state minimal convolution encoder with the fact that a significant source of dynamic power dissipation is modulo-two adders instead of shift registers. This orients the approach towards the combinational logic minimization instead of the sequential minimization.

The generator polynomials, that is, g_0 and g_1 as shown in Figure 1, can be realized with modulo-two adders with the constraint length K as 4 and the code rate k as 1/2. For higher QoS and error correcting capabilities, polynomials with larger K are required, which increases the logical depth of the adder. This adder logic uses XOR gates, which have a higher switching probability or transitions as compared to the basic gates, considering the fact that power dissipation depends on the data, switching probability, and activity of the functional topology. The switching probability P with respect to number of inputs can be calculated for XOR and the basic gates have been computed from Table 1 and depicted in Figure 2. The XOR gates have constant transition probability of 25% where others follow logarithmic means. In Table 1, signal probabilities P_A and P_B are assumed as 0.5 for static CMOS gates [11].

Table 1: Transition probability of XOR gate with basic gates [11].

Gate	Transition probability ($P_{0\to1}$)
AND	$(1 - P_A P_B)P_A P_B$
OR	$((1 - P_A))((1 - P_B))(1 - (1 - P_A)(1 - P_B))$
XOR	$(1 - (P_A + P_B - 2P_A P_B))(P_A + P_B - 2P_A P_B)$

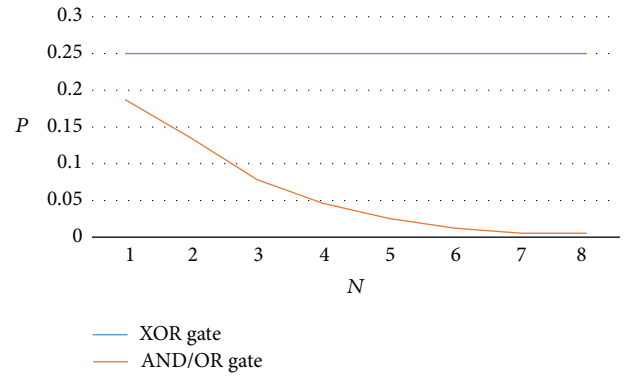

Figure 2: Activity as a function of topology [11].

3. The Construction Technique

The proposed algorithm is used for the implementation of XOR-FREE processing architecture of a chosen nonsystematic polynomial; that is, the input bit itself is not a part of encoded output and feedforward; that is, no feedback path exists in the design. The algorithm consists of five major steps which are summarized as follows.

Deterministic finite automaton of convolution transducer is a six-tuple, $(\Sigma, \Gamma, S, S_0, \delta, \omega)$, consisting of a finite set of input symbols called the alphabet (Σ), where Γ is the output alphabet (a finite, nonempty set of symbols), c is a finite set of states (S), with start state ($S_0 \in S$) and transition function ($\delta : S \times \Sigma \to S$), and $f(\omega)$ is the output function [20].

Algorithm 1 (for XOR-FREE processing of convolutional encoder architecture). Consider

Step 1 (conversion). This is Mealy to Moore conversion by assuming input bit as logic "0"; that is, $\Sigma = \{0\}$, for all possible encoder states S. Next state follows the transition function ($\delta : S + 1 \to S$) like behavior of $K - 1$ bit counter. Find the output responses using (3) and arrange them in tabular form.

Step 2 (grouping). Group encoder states "S" based only on their similar output alphabets Γ.

Step 3 (decomposition). Split the encoded bits used for the state representation in two subparts, row tag (RT) and column tag (CT). Hence,

$$\begin{aligned} \text{CT} &= k^{-1} \quad \text{//Least Significant Bits,} \\ \text{RT} &= [K - \{\text{CT} + 1\}] \quad \text{//Most Significant Bits.} \end{aligned} \tag{6}$$

Step 4 (Isomorphs). Find the Isomorphs* based on similar encoder state and output.

Step 5 (restoring). Restore Mealy back from Moore machine; that is, reconsider logic "1." While applying Even-Odd concept of XOR on the parity bits, the system regains its full functionality.

Algorithm 1 is explained with elaborated example of GSM-900 convolutional generator polynomial. The first step of the algorithm finds the output response of the chosen polynomial using a C-code. The code assumes an input bit as static logic "0," for the possible combinations of states

Table 2: The convolved output for input logic "0" and 2^{K-1} SR states with parity bits as output.

Encoder state	C_0	C_1
00000	0	0
00001	1	1
00010	0	1
00011	1	0
00100	0	0
00101	1	1
00110	0	1
00111	1	0
01000	1	1
01001	0	0
01010	1	0
01011	0	1
01100	1	1
01101	0	0
01110	1	0
01111	0	1

Table 3: Step 1 output is rearranged and then decomposed into RT and CT.

Encoder state	C_0	C_1
00000	0	0
00100	0	0
01001	0	0
01101	0	0
00010	0	1
00110	0	1
01011	0	1
01111	0	1
00011	1	0
00111	1	0
01010	1	0
01110	1	0
00001	1	1
00101	1	1
01000	1	1
01100	1	1

of $K - 1$ bit shift register, and convolves to compute the encoded output. The second step is based on grouping of SR states having a similar output state. The possible values for output states are {00, 01, 10, and 11} for code rate 1/2. The computation after Step 1 is as shown in Table 2. This results in four groups, with each having 2^2 states, rearranged, as shown in Table 3. The shown states are 2^2 instead of 2^3 since logic "1" is never considered as input yet. This can be easily interpreted by MSB bit of Tables 2 and 3 which is logic "0" throughout the computations.

Step 3 splits the encoder state assignment bits into two subparts, RT and CT, as shown in Table 3. Splitting is done using (6). It is highlighted in color code (orange) in the first column of Table 3. After splitting, Table 4 is prepared where color highlights the common rows or the so-called Isomorphs. The Isomorph* states are those which share the same RT and CT and result in the same parity bits. The Isomorph group is formed as shown in Table 5. Finally, the ROM is prepared by merging the common rows of Table 4 and assigning new reduced RT with the same CT. Table 6 is used as a ROM for hardware implementation where the current state of SR and input bit are used for decoding its addresses. The XOR-FREE architecture is then implemented using SR and the ROM formed from Table 6. The addresses of ROM are RT and CT, formed by current input bit and the previous state of shift register as shown in Figure 3. The outputs of the ROM are the encoded sequences which can be taken in serial or parallel sequence as per the requirements.

Table 4: The decomposed RT and CT arrangement.

Row	Column			
	00	01	10	11
00	00	11	01	10
01	11	00	10	01
10	11	00	10	01
11	00	11	01	10

Table 5: The isomorphs for both logic "0" and logic "1."

S. number	Row tag	Row Isomorphs
1	0	000, 001, 110, 111
2	1	010, 011, 100, 101

Table 6: The ROM formed for Table 4.

New RT	CT			
	00	01	10	11
0	00	11	01	10
1	11	00	10	01

4. Results and Discussion

The hardware implementation of the proposed architecture requires shift register and a ROM in addition to its decoding logic. The reconfigurable hardware like FPGA infers these elements from its standard library and synthesizes the architecture in desired manner. While keeping the constraint length K and code rate constant, any change in generator polynomial results in a new ROM with the same elements at different RT and CT positions. This gives an additional advantage of architectural reconfigurability with lesser design complexity. Further, this feature reduces the overhead in partial bit files and saves reconfiguration time. The proposed architecture is implemented in VHDL language and synthesized for Zynq-7 device "XC7Z020-CLG484," that is, Zedboard [29] using Xilinx ISE v14.7 tool. Further, the system structural model can be generated from the behavioral model of Figure 4. Such system structural model consists of a variety of system components such as processor, IPs, memories, and arbiters. The model

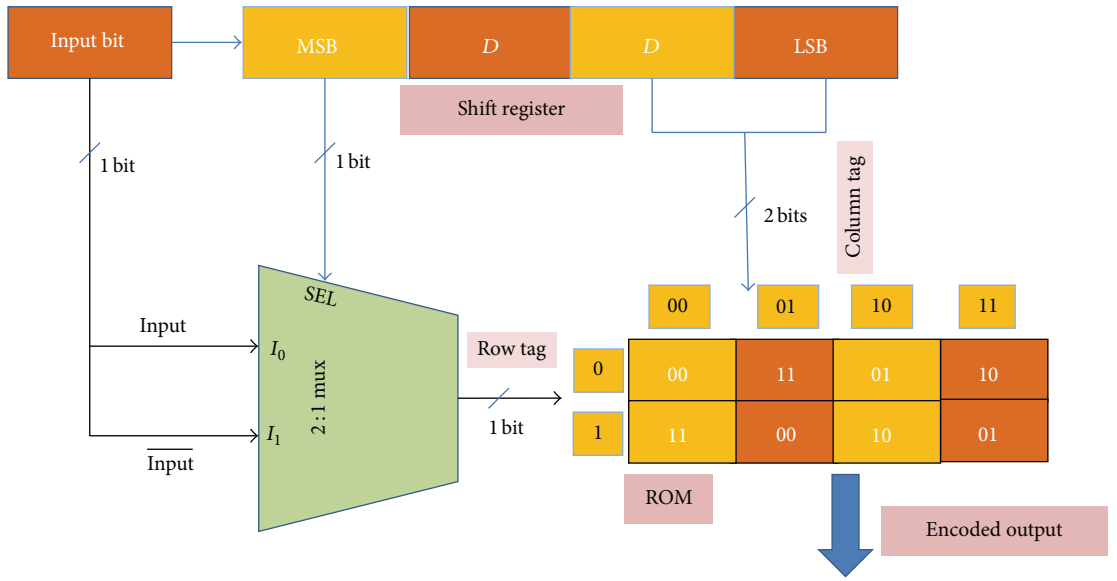

Figure 3: XOR-FREE architecture for GSM-900 standard with generator polynomials $g_0(x) = (31)_8$ and $g_1(x) = (33)_8$.

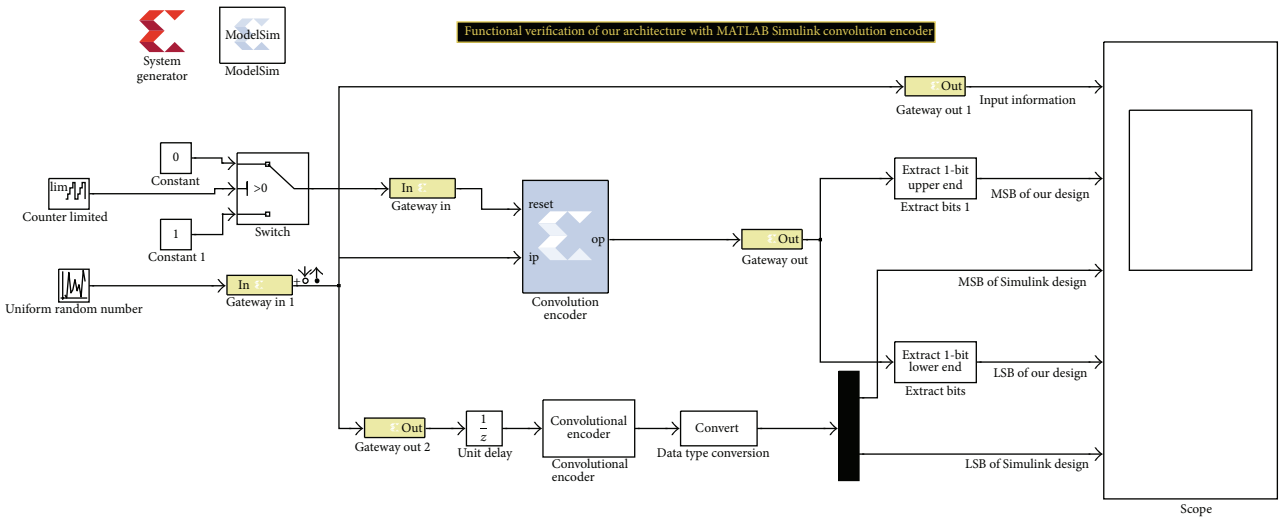

Figure 4: XOR-FREE architecture for GSM-900 standard with generator polynomials $g_0(x) = (31)_8$ and $g_1(x) = (33)_8$.

uses different bus protocols to connect its components [30, 31]. Such system level implementation structure on FPGA along with ROM is depicted in Figure 5. This figure presents the schematic of system level approach; however, detailed implementation is not considered presently. The present paper mainly focuses on the new XOR-FREE algorithm and its behavioral implementation on FPGA.

The dynamic power of system is estimated with XILINX XPower Analyzer tool. For estimating dynamic power, an additional test stimulus is written which generates a random sequence of 1000 bits, clocking at 200 MHz. The functional verification is done in MATLAB system generator environment [32] where the proposed architecture is validated with the convolutional encoder block of Simulink library as shown in Figure 4. The delay element in Figure 4 is used to adjust one clock cycle, consumed during the reset state of the proposed design. The same can be predicted from the waveforms of Figure 6. The hardware cosimulation is performed which tests the proposed design and its functional verification waveform with ModelSim environment as shown in Figure 7.

Finally, it is worth mentioning an interesting observation. The complicated state diagram of 2^{K-1} states is now reduced into two substate diagrams due to RT and CT split as depicted in Figure 8. The algorithm has been tested for various standard polynomials of communication as given in Table 7.

The resource utilization for two different architectures has been estimated and summarized in Table 8. The dynamic activity at different levels, namely, signal, IOs, logic, and

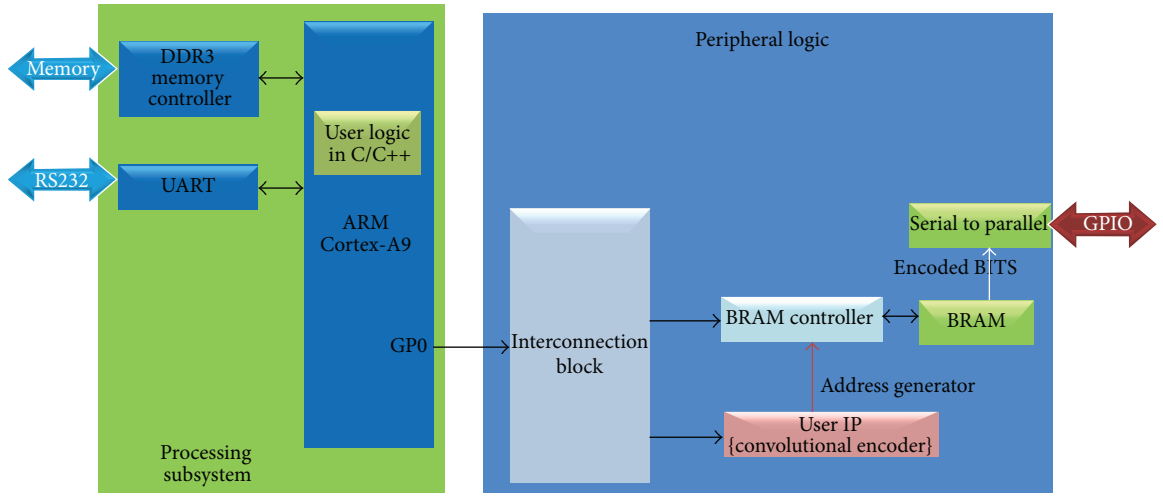

Figure 5: A system level implementation structure on FPGA along with ROM.

Table 7: The proposed algorithm is tested for the following wireless standards.

S. number	Application	M^L	$g_{11}(D)$	$g_{12}(D)$
1	GSM-900	16	31	33
2	802.11, 802.16d, 802.16e, DVB-T	64	133	171
3	802.11a	64	155	117
4	802.11b	64	133	175
5	IS-95	256	753	561
6	CDMA2000	256	753	561
7	WCDMA	256	561	753

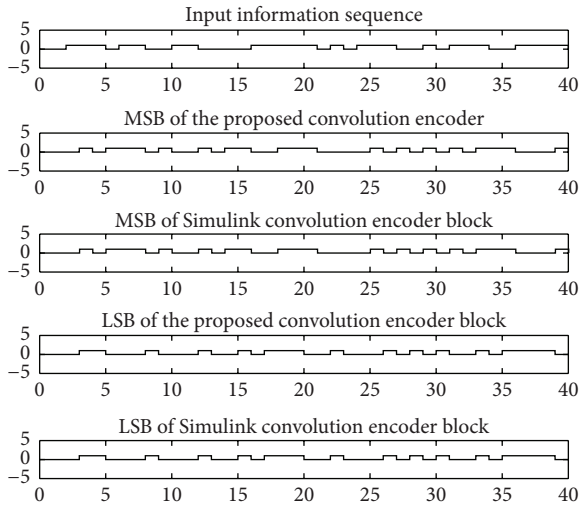

Figure 6: Functional verification of the proposed architecture in MATLAB Simulink.

clock domain, is calculated and compared for conventional and proposed encoder architecture as depicted in Figure 9. The power estimation and resource estimation are done for the highest K, that is, 9 for WCDMA/CDMA2000 standard. The algorithm completely removes the delay consumed

Table 8: Resource utilization comparison for the architectures.

Parameter	Conventional approach	XOR-FREE approach
F_{max} (MHz)	444.642 MHz	585.480 MHz
FFs	12	11
LUTs	03	03
LUT-FF pairs	13	11
BUFG/BUFGCTRLs	01	01

to compute the logical XOR while convolving and hence improves the propagation delay up to 32%, dynamic power up to 30%, and hardware cost up to 20% as compared to conventional architecture.

Usually, the size of the hardware depends upon the nature of the used polynomial, constraint length, and the code rate. While using the proposed algorithm, sometimes the size of multiplexer grows with the constraint length of the polynomial. Moreover, in few cases above $K \geq 12$, the algorithm fails to find commonalities in the chosen polynomials. This tends to increase the ROM size and degrades the maximum operating frequency. However, the present algorithm is validated up to $K = 9$ for various standard polynomials as shown in Table 7 and claims a significant improvement in the performance.

5. Conclusions

The paper addresses the problem of optimization of modulo-2 adders which are a significant source of dynamic power and proposes a novel algorithm to implement XOR-FREE architecture for nonsystematic, feedforward convolution encoder. The approach reduces the standard polynomial into a ROM which gives ease in FPGA implementation. Hardware implementation uses ROM which gives the benefit of reconfiguration by just modifying polynomial of the same constraint length and code rate. The algorithm is tested for known

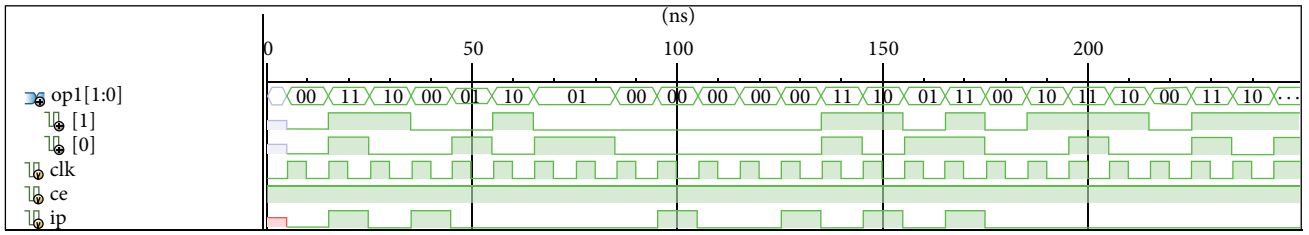

Figure 7: Functional verification of the proposed architecture in ModelSim cosimulation environment.

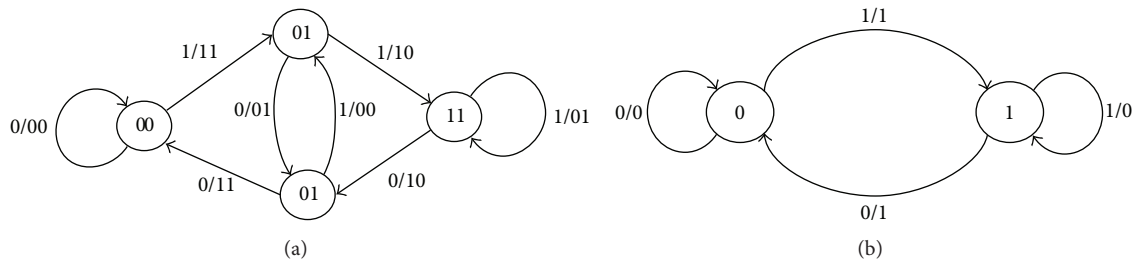

Figure 8: Reduced state diagram by the proposed algorithm (a) for CT {RT/CT} and (b) for RT {Input/RT}.

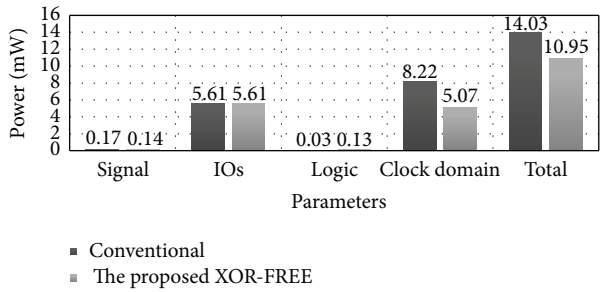

Figure 9: Dynamic power comparison for $K = 9$ and $k = 1/2$ for CDMA-2000 standard.

polynomials of wireless standards. The proposed architecture is novel and promises to improve the propagation delay, dynamic power, and hardware cost for reconfigurable hardware.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] J. Dielissen, N. Engin, S. Sawitzki, and K. van Berkel, "Multi-standard FEC decoders for wireless devices," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 55, no. 3, pp. 284–288, 2008.

[2] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, P. Whiting, and R. Vijayakumar, "Providing quality of service over a shared wireless link," *IEEE Communications Magazine*, vol. 39, no. 2, pp. 150–153, 2001.

[3] J. Hagenauer, "Forward error correcting for CDMA systems," in *Proceedings of the 4th International Symposium on Spread Spectrum Techniques and Applications*, pp. 566–569, Mainz, Germany, September 1996.

[4] G. P. Calzolari, M. Chiani, F. Chiaraluce, R. Garello, and E. Paolini, "Channel coding for future space missions: new requirements and trends," *Proceedings of the IEEE*, vol. 95, no. 11, pp. 2157–2170, 2007.

[5] A. J. Viterbi, "Convolutional codes and their performance in communication systems," *IEEE Transactions on Communication Technology*, vol. 19, no. 5, pp. 751–772, 1971.

[6] S. Lin and D. J. Costello Jr., *Error Control Coding*, Prentice-Hall, Englewood Cliffs, NJ, USA, 2nd edition, 2004.

[7] A. P. Vinod, E. Lai, D. L. Maskell, and P. K. Meher, "An improved common subexpression elimination method for reducing logic operators in FIR filter implementations without increasing logic depth," *Integration, the VLSI Journal*, vol. 43, no. 1, pp. 124–135, 2010.

[8] Y. Yibin, K. Roy, and R. Drechsler, "Power consumption in XOR-based circuits," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC '99)*, vol. 1, pp. 299–302, IEEE, Wanchai, Hong Kong, January 1999.

[9] M. Igor, "Power consumption analysis of XOR based circuits," *Informatics*, vol. 1, no. 9, pp. 97–103, 2006.

[10] T. Stockhammer, A. Shokrollahi, M. Watson, M. Luby, and T. Gasiba, *Application Layer Forward Error Correction for Mobile Multimedia Broadcasting Case Study*, Digital Fountain, 2009.

[11] J. Rabaey, *Low Power Design Essentials*, Springer Science & Business Media, New York, NY, USA, 1st edition, 2009.

[12] R. Pasko, P. Schaumont, V. Derudder, S. Vernalde, and D. Durackova, "A new algorithm for elimination of common subexpressions," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 1, pp. 58–68, 1999.

[13] C. Huang, J. Li, and M. Chen, "Optimizing XOR-based codes," US Patent 8209577 B2, 2012.

[14] S. Benedetto and G. Montorsi, "Role of recursive convolutional codes in turbo codes," *Electronics Letters*, vol. 31, no. 11, pp. 858–859, 1995.

[15] FP6 IST Project WINNER, "Identification of Radio Link Technologies," Deliverable 2.1, July 2004, http://www.ist-winner.org/.

[16] J. L. Massey, "Coding theory," in *Handbook of Applicable Mathematics*, W. Ledermann and S. Vajda, Eds., vol. 5, chapter 16, part B, pp. 623–676, John Wiley & Sons, New York, NY, USA, 1985.

[17] R. J. McEliece and L. Onyszchuk, "The extended invariant factor algorithm with application to the Forney analysis of convolutional codes," in *Proceedings of the IEEE International Symposium on Information Theory*, p. 142, San Antonio, Tex, USA, January 1993.

[18] G. D. Forney Jr., "Convolutional codes I: algebraic structure," *IEEE Transactions on Information Theory*, vol. 16, no. 6, pp. 720–738, 1970.

[19] G. D. Forney Jr., "Convolutional codes. II. Maximum-likelihood decoding," *Information and Computation*, vol. 25, pp. 222–266, 1974.

[20] G. De Micheli, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Optimal state assignment for finite state machines," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 4, no. 3, pp. 269–284, 1985.

[21] M. J. Avedillo, J. M. Quintana, and J. L. Huertas, "State merging and state splitting via state assignment: a new FSM synthesis algorithm," *IEE Proceedings—Computers and Digital Techniques*, vol. 141, no. 4, pp. 229–237, 1994.

[22] P. Ashar, S. Devadas, and A. R. Newton, "Optimum and heuristic algorithms for an approach to finite state machine decomposition," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 10, no. 3, pp. 296–310, 1991.

[23] S. Devdas and A. R. Newton, "Decomposition and factorization of sequential finite state machines," *IEEE Transactions on Computer-Aided Design*, vol. 8, no. 11, pp. 1206–1217, 1989.

[24] C. R. Mohan and P. P. Chakrabarti, "Factorizing FSM's with modify and restore method," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 44, no. 5, pp. 371–377, 1997.

[25] S. Rupesh, D. Madhav, and H. Narayanan, "Decomposition of finite state machines for area, delay minimization," in *Proceedings of the International Conference on Computer Design (ICCD '99)*, pp. 620–625, IEEE Computer Society, Austin, Tex, USA, October 1999.

[26] C. Y. Tsui, J. Monteiro, S. Devadas, C. Y. Tsui, A. M. Despain, and B. Lin, "Power estimation methods for sequential logic circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 3, no. 3, pp. 404–416, 1995.

[27] T. Villa and A. Sangiovanni-Vincentelli, "NOVA: state assignment of finite state machines for optimal two-level logic implementation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 9, no. 9, pp. 905–924, 1990.

[28] D. A. Bader and K. Madduri, "A parallel state assignment algorithm for finite state machines," in *High Performance Computing—HiPC 2004: 11th International Conference, Bangalore, India, December 19–22, 2004. Proceedings*, vol. 3296 of *Lecture Notes in Computer Science*, pp. 297–308, Springer, Berlin, Germany, 2005.

[29] XILINX FPGA Board user's guide, http://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html.

[30] D. Gajski, G. Schirner, S. Abdi, and A. Gerstlauer, *Embedded System Design: Modeling, Synthesis and Verification*, chapter 1, Springer Science & Business Media, London, UK, 1st edition, 2009.

[31] L.-T. Wang, Y.-W. Chang, and K.-T. T. Cheng, *Electronic Design Automation: Synthesis, Verification, and Test*, chapter 5, Morgan Kaufmann, Boston, Mass, USA, 1st edition, 2009, http://store.elsevier.com/Electronic-Design-Automation/isbn-9780123743640/.

[32] Xilinx System Generator User's Guide, http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_7/sysgen_user.pdf.