Variable Parallelism Cyclic Redundancy Check Circuit for 3GPP-LTE/LTE-Advanced

Variable Parallelism Cyclic Redundancy Check Circuit for 3GPP-LTE/LTE-Advanced

Carlo Condo, Maurizio Martina, Gianluca Piccinini, Guido Masera

***Abstract*—Cyclic Redundancy Check (CRC) is often employed in data storage and communications to detect errors. The 3GPP-LTE wireless communication standard uses a 24-bit CRC with every turbo coded frame, thus, the CRC can be exploited to detect residual errors and to enable early stopping of iterations as well. The current state of the art lacks specific CRC implementations for this standard, and most current solutions adopt a fixed degree of parallelism, unsuitable for many turbo decoder architectures. This work proposes a variable parallelism circuit targeting the 3GPP-LTE/LTE-Advanced 24-bit CRC, that can adapt to input data of different sizes. Low complexity is achieved through careful functional sharing among the various parallelisms: comparison with the state of the art shows comparable or superior speed and extremely low complexity.**

***Index Terms*—CRC, 3GPP-LTE, LTE-Advanced, turbo codes**

I. Introduction

The Cyclic Redundancy Check (CRC) is a common technique employed in a variety of fields to detect errors in sequences of bits [1]–[3]. Its straightforward implementation and high reliability has led to ample usage in memories, wired and wireless communications including the 3GPP Long Term Evolution (LTE) standard [4]. In particular, the 3GPP-LTE/LTE-Advanced wireless communication standard relies on turbo codes [5] for forward error correction. Coded frame sizes ranging from 40 bits to 6144 bits with a granularity of down to 8 bits are foreseen: the last 24 bits of each coded frame are the remainder of a 24-bit CRC performed on the remaining bits (known as CRC-24b) [4]. Since the turbo codes decoding algorithm is iterative, the CRC can be used as an early stopping criterion [6]. The CRC is performed on the frame after the error correction process and compared to the received remainder: if they match, the process is stopped, on the contrary, in case of discrepancy, further decoding iterations (each composed of two half iterations) are required. The state of the art is ripe with turbo code decoder designs, and the 3GPP-LTE/LTE-Advanced standard is often considered in both actual application and cases of study. However, very few designs implement the CRC-based early stopping, and scant details on the implementation are given [7]–[9].

The problem of CRC computation has been analyzed in depth in the past. In addition to serial implementations, many solutions for parallel CRC computation have been proposed [10]–[17]. Research on flexible CRC implementations has been particularly prolific, with programmable circuits that can handle different CRC polynomials [10], [11] and flexible design methodologies that produce efficient dedicated circuits given a particular CRC polynomial [13]–[15]. In the vast majority of these solutions, the degree of parallelism is fixed at design time. This poses problems in 3GPP-LTE turbo decoders, since the data on which the CRC is performed are often scattered in various memories, that can be of different size from the CRC parallelism, with an uneven usage within the decoder.

This work proposes a CRC circuit targeting the CRC-24b of 3GPP-LTE that can adapt on-the-fly to the size of the incoming data. By observing how the CRC calculation changes with the parallelism, a flexible low-complexity solution has been devised: three implementations are proposed, each based on different design choices. The rest of the paper is organized as follows: Section II describes the devised CRC parallelization process, while Section III details the designed circuit and its modifications. Implementation results are presented and compared to the state of the art in Section IV and conclusions are drawn in Section V.

II. Parallelization of the CRC operation

Let us recall the CRC-24b polynomial for coded frames as defined by the 3GPP-LTE standard:

$$p_{24} = x^{24} + x^{23} + x^6 + x^5 + x + 1. \quad (1)$$

In the CRC computation, the frame is sequentially divided by the constituent polynomial: if at the end of the division the remainder is equal to the received CRC, the frame is considered correct and the decoding is interrupted. Let d be a frame made of K bits and c a vector of $M+1$ bits holding the current remainder value left-shifted by one position; M is the remainder length and in this case $M = 24$. Thus, in the following, $d(i)$ and $c(i)$ will be referred to as the i-position bit of d and c respectively. The CRC polynomial division is performed in the binary domain as detailed in Alg. 1, where

Algorithm 1 CRC-24b

1: $rm =$ all zeros
2: **for** $i = 0 : K - M - 1$ **do**
3: $\quad c(24:1) = rm$
4: $\quad c(0) = d(i)$
5: $\quad$ **if** $c(24)$=0 **then**
6: $\quad\quad rm = c(23:0)$
7: $\quad$ **else if** $c(24)$=1 **then**
8: $\quad\quad rm = c \oplus p_{24}^b$
9: $\quad$ **end if**
10: **end for**

The authors are with the Department of Electronics and Telecommunications, Politecnico di Torino, Corso Duca degli Abruzzi, 24, 10129 Turin, Italy. E-mail: <name.surname>@polito.it.

$\oplus$ is the binary XOR operation, rm is the new remainder that is used to update c and p_{24}^b is the binary representation of p_{24}, where its 25^{th}, 24^{th}, 7^{th}, 6^{th}, 2^{nd} and first bits are '1' and the others are '0'. The operations in Alg. 1 suppose that the frame bits $d(i)$, without the M CRC bits, are shifted into c one at a time: this operation would be very time consuming, especially when long frame sizes are involved. Assuming to perform one run of the CRC loop (line 2-10 in Alg. 1) in one clock cycle, we define as $N_{CRC} = K - M$ as the number of cycles necessary to complete the computation. Depending on the decoder architecture, it is possible to perform the CRC computation every iteration or every half iteration. Considering the most common parallel turbo decoder architectures, a half iteration requires at least $N_{dec} = K/D$ clock cycles, where D is the number of decoder cores. Since the CRC computation might need to be completed in a half iteration, recent LTE/LTE-Advanced decoder implementations, that use $D = 8$ or $D = 16$, would be forced to wait for the CRC completion. Indeed, even when $D = 1$, a serial CRC calculation could not be sustained. As an example, for $K = 6144$ and allowing up to twelve half iterations (i.e. six iterations, a conservative choice), to obtain a throughput of 450 Mb/s the decoder would need to run at a clock frequency of 5.3 GHz, that is not realistic.

A common approach to speed up the CRC is the parallelization of the polynomial division [13] by unfolding the operation described in Alg. 1, meaning that P bits of the frame are loaded at each cycle in the shift register containing c, leading to $N_{CRC} = (K - M)/\mathrm{P}$. The inherent degree of parallelism in CRC-24b is P= 24, that is $N_{CRC} = 255$. However, 24 is not always an implementation-friendly degree of parallelism. Indeed, the parallel computation requires a 24-bit vector r_d containing part of the hard decision vector d, but in 3GPP-LTE/LTE-Advanced the frame size has a granularity of 8 bits. For this reason, data are usually stored in memories that have a width of one, two or four bytes, and assembling 24-bit vectors is not straightforward. Moreover, there is no guarantee that all windows are composed of the same number of bits: border windows in parallel decoders can be truncated. The proposed low-cost CRC circuit has a variable degree of parallelism, that can adapt on-the-fly to the size of the incoming data and is suitable for diverse decoder structures.

Table I reports the equations necessary for the computation of the updated remainder vector rm, bit per bit, in case of three different degrees of parallelism (8,16 and 24). They have been obtained by unfolding the serial operation described in Alg. 1. The 24-bit vector r_d contains the hard decision bits: since their number varies according to the degree of parallelism, the useful bits are the 8, 16 or 24 most significant bits. In the following, when square brackets are used, the $\oplus$ operation must be applied among the values in the indicated interval, that is

$$r_d(k) \oplus [c(i) \dots c(j)] = r_d(k) \oplus \left(\bigoplus_{l=i}^{j} c(l) \right). \qquad (2)$$

It has been noticed that many of the $\oplus$ of each parallelism degree are contained in those required for the higher degrees of parallelism. Most of the superpositions between the different parallelisms are explicitly shown in Table I: $rm^8(i)$ and $rm^{16}(i)$ are the i-th bit of the remainder as computed with a CRC parallelism 8 (P-8) and 16 (P-16) respectively. To this purpose, the table entries highlighted in light gray are those equations that are completely included in the degree of parallelism just higher, whereas dark gray shading means a partial inclusion. These inclusions allow for an interesting flexible implementation of a variable parallelism CRC at very low cost, as it will be detailed in the next sections.

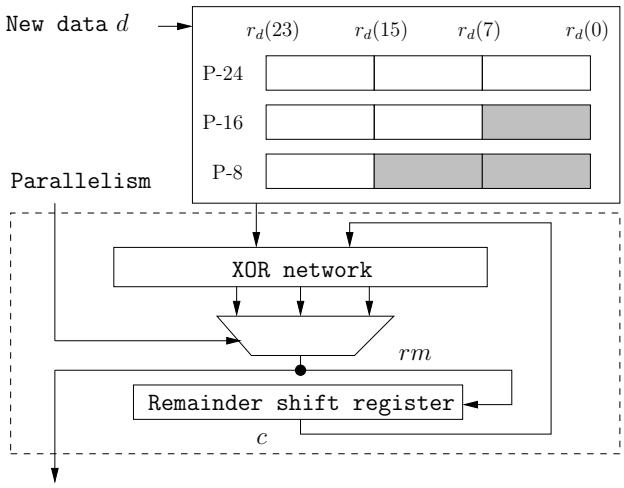

Figure 1. Variable Parallelism CRC circuit.

III. Variable parallelism CRC circuit

The proposed variable parallelism CRC system is pictured in Fig. 1. At every clock cycle, a vector d of new data is injected in the system and stored in a register r_d. Although r_d is 24-bit long, d can be 8, 16 or 24 according to the selected parallelism. In case of P-8 or P-16, d is stored in the 8 or 16 most significant bits of r_d. Together with d, the current remainder c is given to the XOR network: here the equations shown in Table I are implemented. Three possible remainders are obtained, and the correct one is selected according to the current `Parallelism` signal. As d changes at every clock cycle, also the degree of parallelism can change: for example, let us assume a two-core decoder architecture that stores hard decision bits in 16-bit registers. When considering $K = 80$ bits, each core fills two 16-bit registers, while only 8 bits of a third register are used. Consequently, after two P-16 operations, a single P-8 is required, immediately followed by the first P-16 of the second core.

A. Low complexity XOR network

The XOR network is structured as a combinational network of XOR gates that performs the $\oplus$ operations needed by the equations in Table I: $\oplus$ operations among multiple bits within the same equations are tackled via a binary tree of XOR gates. This structure brings a critical path equal to the deepest tree in the system. Five-level deep XOR trees are present in both P-24 ($rm^{24}(0)$, $rm^{24}(5)$, $rm^{24}(23)$) and P-16 ($rm^{16}(0)$, $rm^{16}(23)$), while in P-8 only four-level XOR

Table I
CRC - REMAINDER COMPUTATION FOR PARALLELISM 8, 16 AND 24 CONSIDERING A SINGLE IMPLEMENTATION.

	P- 8	P-16	P-24
$rm(0)$	$r_d(16) \oplus [c(16)..c(23)]$	$r_d(8) \oplus [c(8)..c(14)] \oplus rm^8(23)$	$r_d(0) \oplus [c(0)..c(17)] \oplus c(23)$
$rm(1)$	$r_d(17) \oplus c(16)$	$r_d(9) \oplus c(8)$	$r_d(1) \oplus rm^{16}(16)$
$rm(2)$	$r_d(18) \oplus c(17)$	$r_d(10) \oplus c(9)$	$r_d(2) \oplus rm^{16}(17)$
$rm(3)$	$r_d(19) \oplus c(18)$	$r_d(11) \oplus c(10)$	$r_d(3) \oplus rm^{16}(18)$
$rm(4)$	$r_d(20) \oplus c(19)$	$r_d(12) \oplus c(11)$	$r_d(4) \oplus rm^{16}(19)$
$rm(5)$	$r_d(21) \oplus [c(16)..c(19)] \oplus [c(21)..c(23)]$	$r_d(13) \oplus [c(8)..c(11)] \oplus [c(13)..c(23)]$	$r_d(5) \oplus [c(0)..c(3)] \oplus [c(5)..c(17)] \oplus c(22) \oplus c(23)$
$rm(6)$	$r_d(22) \oplus c(16) \oplus c(21)$	$r_d(14) \oplus c(8) \oplus c(13)$	$r_d(6) \oplus c(0) \oplus c(5) \oplus c(18)$
$rm(7)$	$r_d(23) \oplus c(17) \oplus c(22)$	$r_d(15) \oplus c(9) \oplus c(14)$	$r_d(7) \oplus c(6) \oplus rm^{16}(17)$
$rm(8)$	$c(0) \oplus c(18) \oplus c(23)$	$r_d(16) \oplus c(10) \oplus c(15)$	$r_d(8) \oplus c(7) \oplus rm^{16}(18)$
$rm(9)$	$c(1) \oplus c(19)$	$rm^8(1) \oplus c(11)$	$rm^{16}(1) \oplus rm^{16}(19)$
$rm(10)$	$c(2) \oplus c(20)$	$rm^8(2) \oplus c(12)$	$rm^{16}(2) \oplus rm^{16}(20)$
$rm(11)$	$c(3) \oplus c(21)$	$rm^8(3) \oplus c(13)$	$rm^{16}(3) \oplus rm^{16}(21)$
$rm(12)$	$c(4) \oplus c(22)$	$rm^8(4) \oplus c(14)$	$rm^{16}(4) \oplus c(6)$
$rm(13)$	$c(5) \oplus c(23)$	$r_d(21) \oplus c(15) \oplus c(20)$	$r_d(13) \oplus c(7) \oplus c(12)$
$rm(14)$	$c(6)$	$rm^8(6)$	$rm^{16}(6)$
$rm(15)$	$c(7)$	$rm^8(7)$	$rm^{16}(7)$
$rm(16)$	$c(8)$	$rm^8(8)$	$rm^{16}(8)$
$rm(17)$	$c(9)$	$rm^8(9)$	$rm^{16}(9)$
$rm(18)$	$c(10)$	$rm^8(10)$	$rm^{16}(10)$
$rm(19)$	$c(11)$	$rm^8(11)$	$rm^{16}(11)$
$rm(20)$	$c(12)$	$rm^8(12)$	$rm^{16}(12)$
$rm(21)$	$c(13)$	$rm^8(13)$	$rm^{16}(13)$
$rm(22)$	$c(14)$	$rm^8(14)$	$rm^{16}(14)$
$rm(23)$	$[c(15)..c(23)]$	$[c(7)..c(14)] \oplus rm^8(23)$	$r_d(23) \oplus [c(0)..c(16)] \oplus c(22) \oplus c(23)$

trees are found ($rm^8(0)$, $rm^8(23)$). Consequently, the critical path T_{crit} is equal to $5 \times T_{XOR}$, where T_{XOR} is the delay introduced by a two-input XOR gate. To obtain a minimum complexity XOR network for P-8, P-16 and P-24 CRC and at the same time estimate the cost and gain of the proposed variable parallelism architecture, successive optimizations are applied to the remainder computation. These optimizations can be difficult to perform for automated logic synthesis tools, since they do not reach their optimum in presence of arithmetic dominated by XOR gates [18].

- *Unoptimized parallel CRC:* the starting point of this analysis is the number of XOR gates required for the $\oplus$ operations in Table I without considering the superpositions. The P-8 CRC requires 38 XOR gates, P-16 78 and P-24 105, for a total of 221 XOR gates.
- *Intra-parallelism optimized CRC:* some of the XOR gates counted in the previous analysis are repetitions of other XOR gates within the same parallelism. By reusing them, P-8 CRC is reduced to 24 XOR gates, P-16 needs only 47, and 73 gates are sufficient for P-24, amounting to a total of 144 XOR gates.
- *Inter-parallelism optimized CRC:* by sharing XOR gates between P-8, P-16 and P-24 according to Table I, it is possible to substantially reduce the circuit complexity. In particular, assuming P-8 is implemented with 24 XOR gates, only 31 additional gates are needed for P-16. With both P-8 and P-16 present, P-24 can be implemented with 37 XOR gates, with the total gate count descending to 92.

As it can be observed, starting from an internally optimized P-24 (73 XOR gates), two degrees of parallelism can be added to the CRC at the cost of 19 additional XOR gates (+26% increment).

B. Pipelined XOR network

To shorten the critical path and increase the achievable frequency, the XOR network can be pipelined. Four pipeline stages have been added to the previous architecture, separating the levels of all binary XOR trees and propagating signals accordingly, and requiring 154 delay elements in total. They introduce a latency of four clock cycles, but allow to reduce T_{crit} from $5 \times T_{XOR}$ to T_{XOR}.

C. Extension to 32-bit parallelism

Even though the inherent degree of parallelism of CRC-24b is P= 24, 32-bit registers are common enough in turbo decoders, and P= 32 can be useful. Implementing a degree of parallelism higher than the CRC parallelism, however, comes at a higher complexity cost than the previous ones. Apart from the additional bits in r_d, the equations for P-32 are much more complex than those presented in Table I, and the degree of superposition among P-32 and the smaller parallelisms is reduced. Indeed, with P-8, P-16 and P-24 already present (92 XORs), 54 additional XOR gates are necessary to implement also P-32, for a total of 144 XOR gates. The critical path becomes $T_{crit} = 6 \times T_{XOR}$, but with P-32 only 192 clock cycles are necessary to perform the remainder calculation on largest turbo code frame size in 3GPP-LTE.

IV. IMPLEMENTATION

The proposed circuits have been synthesized with Synopsys Design Compiler on a CMOS 90 nm standard cell technology. The smallest area obtained implementing the CRC circuit with P-8, P-16 and P-24 is 1061 μm^2, with a target maximum frequency of 1 GHz ($T_{crit} = 1$ ns), meaning that the correct functionality is guaranteed for frequencies ≤ 1 GHz. The vast

Table II
COMPARISON AMONG DIFFERENT CRC CIRCUITS: CRC POLYNOMIAL TYPE (CRC), NUMBER OF XOR GATES, NUMBER OF DELAY ELEMENTS, CRITICAL PATH (T_{crit}), REQUIRED CLOCK CYCLES.

Circuit	CRC	XOR gates	Delay elements	T_{crit}	Cycles for m bits
Prop-C	CRC-24b	92	24	$5 \times T_{XOR}$	m/(8-16-24)
Prop-P	CRC-24b	92	154	T_{XOR}	$\frac{m+4}{8-16-24}$
Prop-32	CRC-24b	144	24	$6 \times T_{XOR}$	m/(8-16-24-32)
[13]	CRC-16	137	28	$4 \times T_{XOR}$	$(m+3)/16$
[14]	CRC-16	72	16	$4 \times T_{XOR}$	$(m+16)/16$
[15]	CRC-16	80	16	$16 \times T_{XOR}$	$(m+16)/16$
[13]	CRC-32	467	35	$4 \times T_{XOR}$	$m/32$
[14]	CRC-32	452	32	$5 \times T_{XOR}$	$(m+32)/32$
[15]	CRC-32	614	32	$20 \times T_{XOR}$	$(m+32)/32$

majority of decoders has a much lower working frequency [8], [9]: it is consequently improbable that the system critical path resides in the variable parallelism CRC circuit. However, relaxing the area constraint (and thus disrupting the binary tree structure), maximum frequencies as high as 2.5 GHz ($T_{crit} = 0.4$ ns) can be obtained with an area of 1548 μm^2. Another implementation has been carried out taking into account the pipelined XOR network. The maximum achievable frequency is 5.26 GHz ($T_{crit} = 0.19$ ns), obtained with an area occupation of 4088 μm^2: as expected, it is roughly five times faster than when enforcing the binary tree structure in the combinational XOR network. Due to the very high frequencies we have obtained, in most implementations the pipelined architecture would be unnecessary, bringing an overhead of four clock cycles of latency and no actual benefit in speed. A final implementation takes in account also the fourth degree of parallelism P-32: maintaining the tree structure in the XOR network, an area of 2338 μm^2 and a maximum frequency of 833 MHz have been obtained.

Table II presents the characteristics of the proposed CRC together with other solutions present in literature: the combinational circuit is labeled Prop-C, the pipelined architecture is Prop-P and the circuit with P-32 is Prop-32. The number of XOR gates needed for each implementation is listed alongside the number of delay elements, the critical path and the number of cycles needed to perform the CRC calculation on m bits. Since the proposed circuits implements concurrently different degrees of parallelism that can be selected on-the-fly, the number of cycles depends on the chosen mode (P-8, P-16, P-24 or P-32). To the best of our knowledge, no detailed CRC circuit implementations for 3GPP-LTE exist in the state of the art. Comparison is consequently attempted with other parallel CRC circuits targeting similar polynomials: in particular, the 16-bit CRC-16 and 32-bit CRC-32 are taken in account. CRC-16 and CRC-24b share similar polynomial structures, with a large number of zero coefficients between the 2^{nd} and 3^{rd} highest order nonzero coefficients, while CRC-32 allows to evaluate how circuit complexity rises with the order of the polynomial and is closer to Prop-32. The CRC solution employed in [9] is not included in Table II: the lack of implementation details prevents a fair comparison, but a qualitative estimation is possible. A 4×24 bit look-up table is necessary for every decoding core to implement the distributed part of the CRC, while the recombination circuit requires a tree of adders that widens and deepens with the increasing of D, leading to a fast but costly implementation. The circuit designed in [13] starts from the sequential Linear Feedback Shift Register (LFSR) implementation and applies unfolding, pipelining and retiming to reduce the critical path and increase the parallelism. Our approach is different in concept, since starting from the desired parallelisms we have devised a minimum-complexity circuit sharing as many logic functions as possible. The architecture in [13] relies on a four-level pipeline: it requires a total of 28 delay elements, performing the CRC-16 calculation on m bits in $(m+3)/16$ cycles, and has a T_{crit} of $4 \times T_{XOR}$. The proposed architecture, on the contrary, requires 24 delay elements, and achieves $T_{crit} = 5 \times T_{XOR}$ without the need of pipelining. Supposing to use P-16 only, m bits are handled in $m/16$ clock cycles. Regardless of the additional degrees of parallelism (P-8 and P-24) and of the more complex polynomial, the proposed architecture requires only 92 XOR gates, against the 137 of [13]. The LSFR-based parallel CRC-16 presented in [14] and [15] rely on a number of XOR gates comparable to both Prop-C and Prop-P, and they need fewer delay elements. However, they require a larger number of cycles to complete the computations, and [15] has a very large T_{crit}. All three approaches [13]–[15] experience a steep increment in complexity w.r.t. CRC-16 when implementing CRC-32. Even though the calculations involved in CRC-32 are comparable in complexity to those involved in P-32 for CRC-24b, Prop-32 requires less than 1/3 of the XOR gates of [13], [14] and less than 1/4 of [15]. This achievement is even more significant considering that the effectiveness of [13]–[15] is reduced in presence of CRC polynomials of order different from a power of two.

V. CONCLUSION

In this work, a novel circuit for the parallel computation of CRC-24b employed in the 3GPP-LTE/LTE-Advanced standard has been proposed. It is able to perform the CRC calculation with a variable degree of parallelism, that can be changed on-the-fly. Three versions of the circuit have been designed: a low-complexity circuit supporting three degrees of parallelism, a fast pipelined version, and an extended design supporting four different parallelisms. Implemented in CMOS 90 nm technology, they all show very good complexity and speed figures: comparison with the state of the art reveals unmatched on-the-fly adaptivity at an extremely low complexity cost and comparable or superior speed.

VI. ACKNOWLEDGMENT

This work has been partially funded by the NEWCOM# project, and developed within its work package 2.3.2 “Tools for embedded hardware/software architectures”.

REFERENCES

[1] F. Song, X. N. Liu, and Q. H. Wu, "A multi-CRC selective HARQ scheme for MIMO systems," in *Wireless Communications and Signal Processing (WCSP), 2010 International Conference on*, 2010, pp. 1–6.

[2] Y.-R. Chuang, J.-W. Chen, and C.-S. Hsu, "Investigate partial CRC-32 characteristic and performance for real-time multimedia streamings in 802.11 wireless mesh networks," in *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*, 2011, pp. 3415–3420.

[3] Y. Wu and Y. Qiu, "The 8-bit parallel CRC-32 research and implementation in usb 3.0," in *Computer Science Service System (CSSS), 2012 International Conference on*, 2012, pp. 1079–1082.

[4] *Multiplexing and Channel Coding*, 3GPP Std. TS36.212, 2012.

[5] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error correcting coding and decoding: Turbo codes," in *IEEE International Conference on Comm.*, 1993, pp. 1064–1070.

[6] L. Guerrieri, D. Veronesi, and P. Bisaglia, "Stopping rules for duo-binary turbo codes and application to HomePlug AV," in *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, 2008, pp. 1–5.

[7] S.-J. Lee, M. Goel, Y. Zhu, J.-F. Ren, and Y. Sun, "Forward error correction decoding for WiMAX and 3GPP LTE modems," in *Signals, Systems and Computers, Asilomar Conference on*, 2008, pp. 1143–1147.

[8] M. May, T. Ilnseher, N. Wehn, and W. Raab, "A 150Mbit/s 3GPP LTE turbo code decoder," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, 2010, pp. 1420–1425.

[9] S. Belfanti, C. Roth, M. Gautschi, C. Benkeser, and Q. Huang, "A 1Gbps LTE-advanced turbo-decoder ASIC in 65nm CMOS," in *VLSI Circuits (VLSIC), Symposium on*, 2013, pp. C284–C285.

[10] C. Toal, K. McLaughlin, S. Sezer, and X. Yang, "Design and implementation of a field programmable CRC circuit architecture," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 17, no. 8, pp. 1142–1147, 2009.

[11] M. Grymel and S. Furber, "A novel programmable parallel CRC circuit," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 19, no. 10, pp. 1898–1902, 2011.

[12] J. Cho, B. Sung, and W. Sung, "Block-interleaving based parallel CRC computation for multi-processor systems," in *Signal Processing Systems (SIPS), 2010 IEEE Workshop on*, 2010, pp. 311–316.

[13] C. Cheng and K. Parhi, "High-speed parallel CRC implementation based on unfolding, pipelining, and retiming," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 53, no. 10, pp. 1017–1021, 2006.

[14] G. Campobello, G. Patane, and M. Russo, "Parallel CRC realization," *Computers, IEEE Transactions on*, vol. 52, no. 10, pp. 1312–1319, 2003.

[15] M. Sprachmann, "Automatic generation of parallel CRC circuits," *Design Test of Computers, IEEE*, vol. 18, no. 3, pp. 108–114, 2001.

[16] M. Ayinala and K. Parhi, "Efficient parallel VLSI architecture for linear feedback shift registers," in *Signal Processing Systems (SIPS), 2010 IEEE Workshop on*, 2010, pp. 52–57.

[17] ——, "High-speed parallel architectures for linear feedback shift registers," *Signal Processing, IEEE Transactions on*, vol. 59, no. 9, pp. 4459–4469, 2011.

[18] A. Verma and P. Ienne, "Improving XOR-dominated circuits by exploiting dependencies between operands," in *Design Automation Conference. Asia and South Pacific*, 2007, pp. 601–608.