

# We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

**4,800**

Open access books available

**122,000**

International authors and editors

**135M**

Downloads

Our authors are among the

**154**

Countries delivered to

**TOP 1%**

most cited scientists

**12.2%**

Contributors from top 500 universities



**WEB OF SCIENCE™**

Selection of our books indexed in the Book Citation Index  
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?  
Contact [book.department@intechopen.com](mailto:book.department@intechopen.com)

Numbers displayed above are based on latest data collected.

For more information visit [www.intechopen.com](http://www.intechopen.com)



# A Brief Overview of CRC Implementation for 5G NR

Hao Wu

## Abstract

In fifth generation (5G) new radio (NR), the medium access control (MAC) layer organizes the data into the transport block and transmits it to the physical layer. The transport block consists of up to million bits. When the transport block size exceeds a threshold, the transport block is divided into multiple equal size code blocks. The code block consists of up to 8448 bits. Both the transport block and the code block have a cyclic redundancy check (CRC) attached. Due to the difference in the size of the transport block and the code block, the CRC processing scheme suitable for the transport block and that suitable for the code block are different. This chapter gives an overview of the CRC implementation in 5G NR.

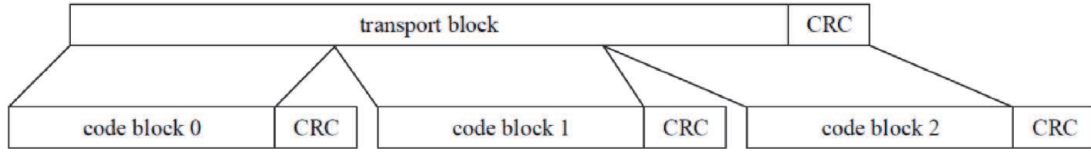
**Keywords:** 5G, NR, CRC, transport block, code block

## 1. Introduction

In order to provide high data transmission rates, the bandwidth of mobile communication systems is increasing. In fourth generation (4G) long term evolution (LTE), the maximum transmission bandwidth for one component carrier is 20 MHz [1]. In fifth generation (5G) new radio (NR), the frequency bands are divided into two parts: frequency range 1 (FR1) below 6 GHz and frequency range 2 (FR2) above 24.25 GHz. The maximum transmission bandwidth for one component carrier is 100 MHz and 400 MHz in FR1 and FR2 respectively [2]. The increasing system bandwidth brings new problems to the design of the transmitter and the receiver. In this chapter of the book, we focus on the cyclic redundancy check (CRC) implementation in 5G NR.

In 5G NR, there are many physical channels defined in the downlink and the uplink [3]. The downlink physical channels consist of the physical downlink shared channel (PDSCH), the physical downlink control channel (PDCCH), the physical broadcast channel (PBCH), etc. The uplink physical channels consist of physical uplink shared channel (PUSCH), the physical uplink control channel (PUCCH), the physical random access channel (PRACH), etc. The PDSCH and the PDSCH are mainly used to transmit data. The usage scenarios of 5G NR consist of enhanced mobile broadband (eMBB), massive machine-type communications (mMTC) and ultra-reliable and low latency communications (URLLC) [4, 5]. The usage scenario of the eMBB requires high data transmission rates. As a consequence, we focus on the PDSCH and the PUSCH in this chapter.

The medium access control (MAC) layer organizes the data into the transport block and transmits it to the physical layer. In 5G NR, the maximum transport block



**Figure 1.**  
The transport block and the code block.

size is 1,277,992 [6]. The processing of the transport block is shown in **Figure 1** [7]. If the transport block size is larger than 3824, a 16-bit CRC is added at the end of the transport block. Otherwise, a 24-bit CRC is added at the end of the transport block. The transport block is divided into multiple equal size code blocks when the transport block size exceeds a threshold. For quasi-cyclic low-density parity-check code (QC-LDPC) base graph 1, the threshold is equal to 8448. For QC-LDPC base graph 2, the threshold is equal to 3840. In 5G NR, the maximum code block size number is 8448. An additional 24-bit CRC is added at the end of each code block when there is a segmentation. Due to the difference in the size of the transport block and the code block, the CRC processing scheme suitable for the transport block and that suitable for the code block are different.

The rest of this chapter is organized as follows. Section 2 describes the system model of the transport block and the code block in 5G NR. Section 3 gives two properties of the CRC. Section 4 presents the overview of the CRC implementation. Finally, Section 5 gives the conclusion.

## 2. System model

Let  $\mathbf{a} = [a_0, a_1, \dots, a_{L-1}, a_L, a_{L+1}, \dots, a_{L+N-1}]$  be the transport block including the transport block level CRC, where  $L$  is the transport block size and  $N$  is the transport block level CRC size. Note that  $\mathbf{p} = [a_L, a_{L+1}, \dots, a_{L+N-1}]$  is the transport block level CRC. If  $L$  is smaller than or equal to 3824, then  $N$  is equal to 16 and  $\mathbf{p}$  is generated by the following cyclic generator polynomial:

$$g_{16}(x) = x^{16} + x^{12} + x^5 + 1 \quad (1)$$

If  $L$  is larger than 3824, then  $N$  is equal to 24 and  $\mathbf{p}$  is generated by the following cyclic generator polynomial:

$$g_{24A}(x) = x^{24} + x^{23} + x^{18} + x^{17} + x^{14} + x^{11} + x^{10} + x^7 + x^6 + x^5 + x^4 + x^3 + x + 1 \quad (2)$$

When  $L + N$  is larger than  $M$ , the transport block including the transport block level CRC is segmented into multiple code blocks. Let  $R$  be code rate of the initial transmission indicated by the modulation and coding scheme (MCS) index. If  $L > 292$  and  $R > 0.67$  or  $L > 3824$  and  $R > 0.25$ , then QC-LDPC base graph 1 is used and  $M$  is equal to 8448. Otherwise, QC-LDPC base graph 2 is used and  $M$  is equal to 3840.

When there is no segmentation, the number of code blocks  $C$  is equal to 1. When there is a segmentation, the number of code blocks  $C$  is equal to

$$C = \lceil (L + N) / (M - 24) \rceil \quad (3)$$

In the following sections, we mainly consider the case that there is a segmentation. Let  $c^i = [c_0^i, c_1^i, \dots, c_{K-1}^i]$  be the  $i$ th code block, where  $K$  is the code block size and is equal to

$$K = (L + N)/C + 24 \quad (4)$$

Note that the procedure of the transport block size determination guarantees that  $(L + N)$  is divisible by  $C$ .  $[c_{K-24}^i, c_{K-23}^i, \dots, c_{K-1}^i]$  is the code block level CRC, which is generated by the cyclic generator polynomial

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

$c_j^i$  is equal to

$$c_j^i = a_{i(K-24)+j} \quad (6)$$

where  $0 \leq j \leq K - 25$ . In the following, the processing of the transport block includes: QC-LPDC encoding, rate matching, bit interleaving and code block concatenation. The encoded transport block is transmitted over the air after the symbol level processing.

At the receiver side, the following steps are carried out for the transport block: code block segmentation, bit de-interleaving, de-rate matching, QC-LPDC decoding, code block concatenation. We need to check whether each code block and the transport block are correctly received. Let  $d^i = [d_0^i, d_1^i, \dots, d_{K-1}^i]$  be the  $i$ th received code block after the hard decision and  $e = [e_0, e_1, \dots, e_{L+N-1}]$  be the received transport block after the hard decision.  $e_j$  is equal to

$$e_j = d_u^v \quad (7)$$

where  $v = \lfloor j/(K - 24) \rfloor$ ,  $u = \text{mod}(j, K - 24)$  and  $0 \leq j \leq L + N - 1$ . The undetected error probability is required to be less than  $10^{-6}$  in 5G NR [8, 9]. Since the parity check capacity of QC-LDPC codes alone cannot meet the undetected error probability requirement of 5G NR [8, 9], we need to use the CRC check to determine whether  $d^i$  and  $e$  are correctly received.

### 3. Properties of the CRC

In this section, we give two properties of the CRC. These properties are useful in the CRC implementation. Before giving these properties, we define some variables. Let  $A(x)$  and  $B(x)$  be the polynomials. Let  $g(x)$  be the cyclic generator polynomial.  $\text{CRC}_{g(x)}[A(x)]$  is defined as the remainder when  $A(x)$  is divided by  $g(x)$ . The two properties are listed as follows.

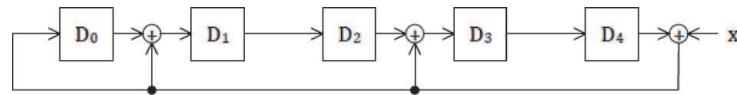
#### Property 1.

$$\text{CRC}_{g(x)}[A(x)B(x)] = \text{CRC}_{g(x)}[\text{CRC}_{g(x)}[A(x)]\text{CRC}_{g(x)}[B(x)]] \quad (8)$$

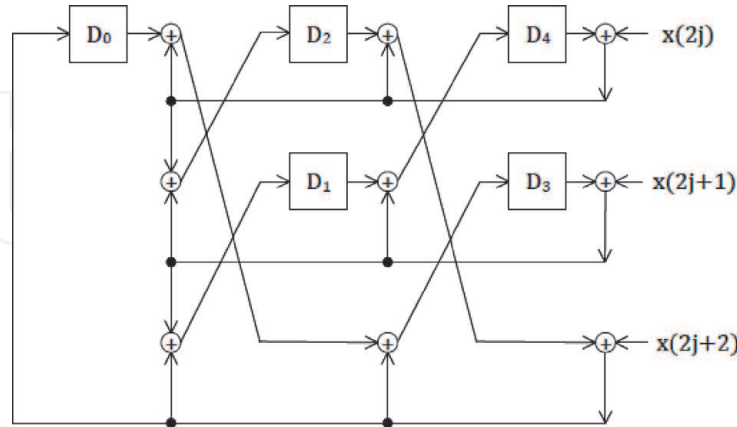
Property 1 implies that  $\text{CRC}_{g(x)}[A(x)B(x)]$  can be obtained by computing the CRC of  $A(x)$  and  $B(x)$  independently.

#### Property 2.





**Figure 3.**  
 CRC implementation for  $g(x) = x^5 + x^3 + x + 1$ .



**Figure 4.**  
 CRC implementation for  $g(x) = x^5 + x^3 + x + 1$ .

to speed the CRC calculation. For example, another CRC implementation for  $g(x) = x^5 + x^3 + x + 1$  is shown in **Figure 4** [14, 15]. The parallelism of this CRC implementation is 3 and thus three bits are processed on every clock cycle. From **Figures 3** and **4**, it is clear that parallelism comes at the expense of the increased circuit complexity.

#### 4.2 CRC implementation by parallel processing

In this scheme,  $e$  is segmented into multiple blocks and the CRC of each block is obtained by parallel processing.  $e$  is segmented into multiple blocks [16]:

$$e^0, e^1, \dots, e^{M-1} \quad (11)$$

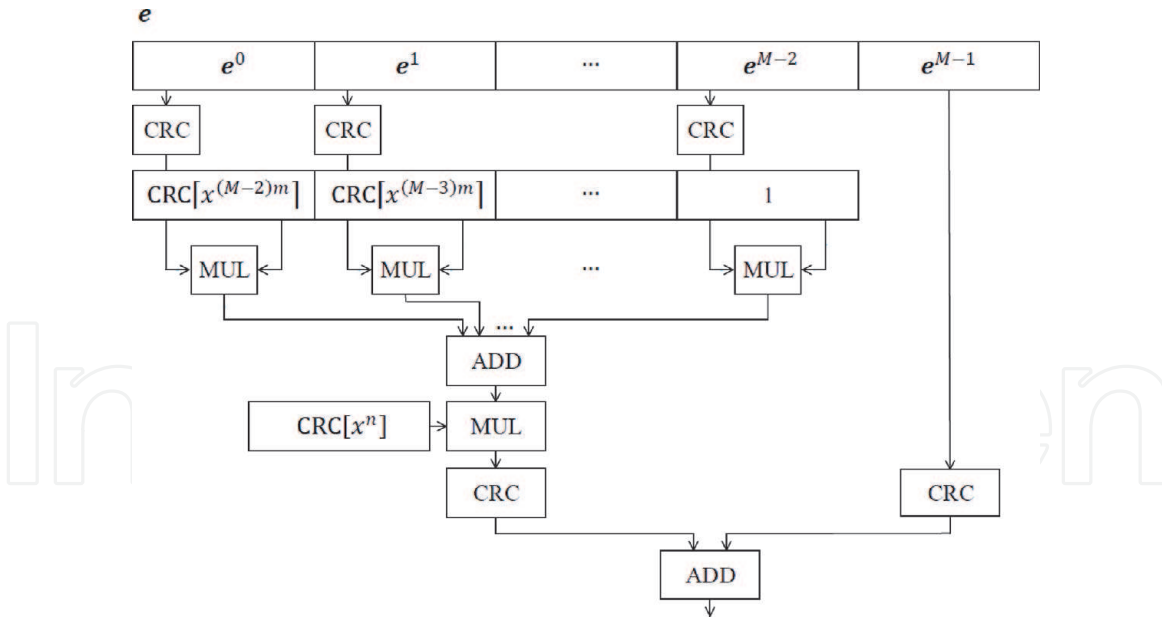
The size of  $e^{M-1}$  is  $n$  and the size of  $e^j$  is  $m$ , where  $0 \leq j \leq M - 2$ . Note that  $L + N$  is equal to  $n + m(M - 1)$ . As a consequence,  $e$  can be expressed as

$$e = e^0 x^{(M-2)m+n} + e^1 x^{(M-3)m+n} + \dots + e^{M-3} x^{n+m} + e^{M-2} x^n + e^{M-1}$$

The CRC of  $e$  is given by

$$\begin{aligned} \text{CRC}[e] &= \text{CRC} \left[ e^0 x^{(M-2)m+n} + \dots + e^{M-2} x^n + e^{M-1} \right] \\ &= \text{CRC} \left[ \text{CRC}[e^0] \text{CRC} \left[ x^{(M-2)m+n} \right] + \dots + \text{CRC}[e^{M-2}] \text{CRC} \left[ x^n \right] \right] + \text{CRC}[e^{M-1}] \\ &= \text{CRC} \left[ \left[ \text{CRC}[e^0] \text{CRC} \left[ x^{(M-2)m} \right] + \dots + \text{CRC}[e^{M-2}] \right] \text{CRC} \left[ x^n \right] \right] + \text{CRC}[e^{M-1}] \end{aligned} \quad (12)$$

The above expression explains how  $\text{CRC}[e]$  is obtained. The detail is shown in **Figure 5**.  $\text{CRC}[x^m]$ ,  $\text{CRC}[x^{2m}]$ , ...,  $\text{CRC}[x^{(M-2)m}]$  and  $\text{CRC}[x^n]$  do not depend on the transport block size and can be precomputed. Since  $n$  is in the range  $[0, m - 1]$ , variables that need to be precomputed include



**Figure 5.**  
CRC implementation by parallel processing.

$$\text{CRC}[x^1], \text{CRC}[x^2], \dots, \text{CRC}[x^{m-1}], \text{CRC}[x^m], \text{CRC}[x^{2m}], \dots, \text{CRC}[x^{(M-2)m}] \quad (13)$$

As a consequence, the number of variables that needs to be precomputed is  $m + M - 3$ .

It is clear that the memory that needs to store the variables increases with the transport block size. To reduce the memory,  $\text{CRC}[x^{am}]$  can be recursively calculated by using  $\text{CRC}[x^m]$  [17]. That is,  $\text{CRC}[x^{am}]$  is recursively obtained by the following expression

$$\text{CRC}[x^{am}] = \text{CRC}[\text{CRC}[x^{(a-1)m}] \text{CRC}[x^m]] \quad (14)$$

In this way, the variables that need to be precomputed include

$$\text{CRC}[x^1], \text{CRC}[x^2], \dots, \text{CRC}[x^{m-1}], \text{CRC}[x^m] \quad (15)$$

As a consequence, the number of variables that needs to be precomputed is  $m$ .

### 4.3 CRC implementation by serial processing

In this scheme,  $e$  is segmented into multiple blocks and the CRC of each block is obtained by serial processing.  $e$  is segmented into multiple blocks [18]:

$$e^0, e^1, \dots, e^{M-1} \quad (16)$$

The size of  $e^{M-1}$  is  $n$  and the size of  $e^j$  is  $m$ , where  $0 \leq j \leq M - 2$ . Note that  $L + N$  is equal to  $n + m(M - 1)$ .  $e$  can be expressed as

$$e = e^0 x^{(M-2)m+n} + e^1 x^{(M-3)m+n} + \dots + e^{M-3} x^{n+m} + e^{M-2} x^n + e^{M-1} \quad (17)$$

The CRC of  $e$  is given by

$$\begin{aligned}
 T_1 &= \text{CRC} \left[ e^0 x^{(p-1)m} + \dots + e^{1p-1} \right] \\
 T_2 &= \text{CRC} \left[ e^p x^{(p-1)m} + \dots + e^{2p-1} \right] + \text{CRC} [T_1 x^{mp}] \\
 &\dots \\
 T_e &= \text{CRC} \left[ e^{(e-1)p} x^{(p-1)m} + \dots + e^{ep-1} \right] + \text{CRC} [T_{e-1} x^{mp}] \\
 T_{e+1} &= \text{CRC} \left[ e^{ep} x^{(M-ep-2)m} + \dots + e^{M-2} \right] + \text{CRC} [T_e x^{(M-eP-1)m}] \\
 T_{e+2} &= \text{CRC} [T_{e+1} x^n] + \text{CRC} [e^{M-1}]
 \end{aligned} \tag{18}$$

where  $e = \lfloor (M - 1)/P \rfloor$ . The above expression explains how  $\text{CRC}[e]$  is calculated. The detail is shown in **Figure 6**.  $\text{CRC}[x^m]$ ,  $\text{CRC}[x^{2m}]$ , ...,  $\text{CRC}[x^{(p-1)m}]$  and  $\text{CRC}[x^n]$  do not depend on the transport block size and can be precomputed. Since  $n$  is in the range  $[0, m - 1]$ , variables that need to be precomputed include

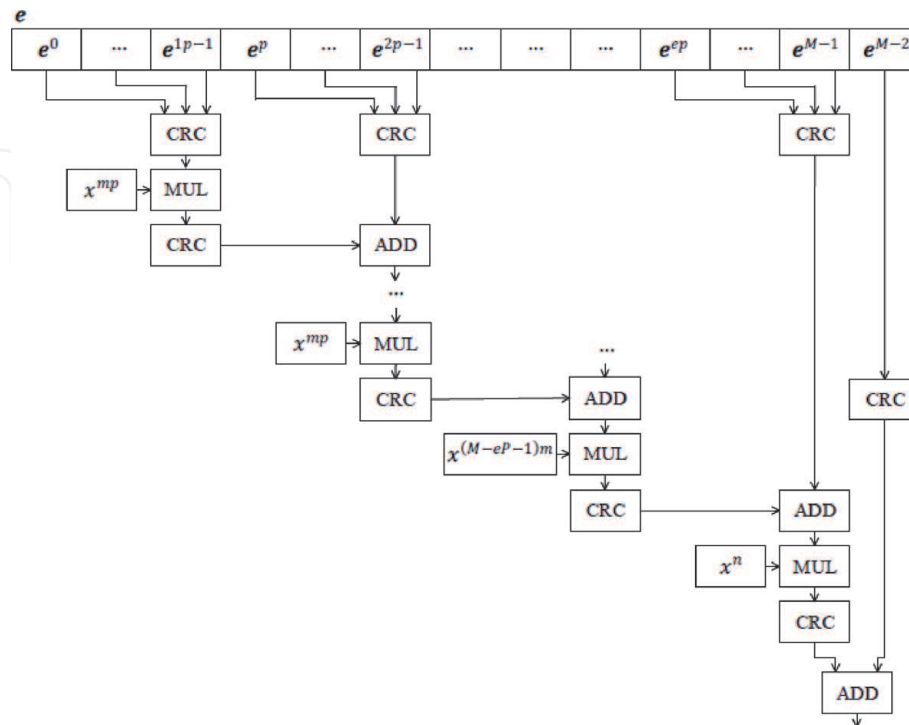
$$\text{CRC}[x^1], \text{CRC}[x^2], \dots, \text{CRC}[x^{m-1}], \text{CRC}[x^m], \text{CRC}[x^{2m}], \dots, \text{CRC}[x^{(p-1)m}] \tag{19}$$

As a consequence, the number of variables that needs to be precomputed is  $m + p - 2$ .

It is clear that the memory that needs to store the variables increases with the transport block size. To reduce the memory,  $\text{CRC}[x^{\alpha m}]$  can be recursively calculated by using  $\text{CRC}[x^m]$  [17]. That is,  $\text{CRC}[x^{\alpha m}]$  is recursively obtained by the following expression

$$\text{CRC}[x^{\alpha m}] = \text{CRC} \left[ \text{CRC} [x^{(\alpha-1)m}] \text{CRC}[x^m] \right] \tag{20}$$

In this way, the variables that need to be precomputed include



**Figure 6.**  
 CRC implementation by serial processing.



$$\text{CRC}[x^1], \text{CRC}[x^2], \dots, \text{CRC}[x^{m-1}], \text{CRC}[x^m] \quad (21)$$

As a consequence, the number of variables that needs to be precomputed is  $m$ .

#### 4.4 The Sarwate algorithm

Sarwate proposes an algorithm based on the lookup table [19]. The detail and the proof of the algorithm can be found in [19]. The Sarwate algorithm is shown in **Figure 7** [20]. The Sarwate algorithm uses a single table of 256 32-bit elements and reads the bits byte by byte. Modern processors usually access 32 bits or 64 bits at a

```

1 crc = INIT_VALUE;
2 while (p_buf < p_end)
3 {
4   crc = table[(crc ^ *p_buf++) & 0x000000FF] ^ (crc >> 8);
5 }
6 return crc ^ FINAL_VALUE;
```

**Figure 7.**  
*The Sarwate algorithm.*

```

1 crc = INIT_VALUE;
2 while (p_buf < p_end)
3 {
4   crc ^= *(uint32_t *) p_buf;
5   term1 = table_56[crc & 0x000000FF] ^
           table_48[(crc >> 8) & 0x000000FF];
6   term2 = crc >> 16;
7   crc = term 1 ^
           table_40[term2 & 0x000000FF] ^
           table_32[(term2 >> 8) & 0x000000FF];
8   p_buf += 4;
9 }
10 return crc ^ FINAL_VALUE;
```

**Figure 8.**  
*The slicing-by-4 algorithm.*

```

1 crc = INIT_VALUE;
2 while (p_buf < p_end)
3 {
4   crc ^= *(uint32_t *) p_buf;
5   p_buf += 4;
6   term1 = table_88[crc & 0x000000FF] ^
           table_80[(crc >> 8) & 0x000000FF];
7   term2 = crc >> 16;
8   crc = term 1 ^
           table_72[term2 & 0x000000FF] ^
           table_64[(term2 >> 8) & 0x000000FF];
9   term1 = table_56[(*(uint32_t *)p_buf) & 0x000000FF] ^
           table_48[((uint32_t *)p_buf) >> 8) & 0x000000FF];
10  term2 = (*(uint32_t *) p_buf) >> 16;
11  crc = crc ^
           term1 ^
           table_40[term2 & 0x000000FF] ^
           table_32[(term2 >> 8) & 0x000000FF];
12  p_buf += 4;
13 }
14 return crc ^ FINAL_VALUE;
```

**Figure 9.**  
*The slicing-by-8 algorithm.*

time. As a consequence, the Sarwate algorithm is not efficient. Some schemes have been proposed in the literatures to solve this problem.

#### **4.5 The slicing-by-4 and slicing-by-8 algorithms**

Kounavis and Berry propose the slicing-by-4 and slicing-by-8 algorithms based on the lookup table [20]. The detail and the proof of the algorithms can be found in [20]. The slicing-by-4 and slicing-by-8 algorithms are shown in **Figures 8** and **9** respectively [20]. The slicing-by-4 algorithm uses four tables of 256 32-bit elements and reads 32 bits at a time. The slicing-by-8 algorithm uses eight tables of 256 32-bit elements and reads 64 bits at a time. The performance of the slicing-by-4 and slicing-by-8 algorithms is improved compared to the Sarwate algorithm.

### **5. Conclusion**

In 5G NR, the transport block consists of up to million bits and the code block consists of up to 8448 bits. Due to the difference in the size of the transport block and the code block, the scheme of the CRC processing suitable for the transport block and that suitable for the code block are different. This chapter gives an overview of the CRC implementation in 5G NR.

### **Conflict of interest**

The authors declare no conflict of interest.

### **Author details**


Hao Wu<sup>1,2</sup>

1 Department of Wireless Product Research and Design Institute, ZTE Corporation, Shenzhen, China

2 State Key Laboratory of Mobile Network and Mobile Multimedia Technology, ZTE Corporation, Shenzhen, China

\*Address all correspondence to: [wu.hao19@zte.com.cn](mailto:wu.hao19@zte.com.cn)

### **IntechOpen**

© 2020 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/3.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. 

## References

- [1] Erik D, Stefan P, Johan S. 4G: LTE/LTE-advanced for Mobile Broadband. 2nd ed. Oxford, UK: Elsevier; 2014
- [2] Erik D, Stefan P, Johan S. 5G NR: The Next Generation Wireless Access Technology. London, UK: Elsevier; 2018
- [3] 3GPP TS 38.211, V15.3.0, NR; Physical channels and modulation (Release 15). 2018-09
- [4] Hyoungju J, Sunho P, Jeongho Y, Younsun K, Juho L, Byonghyo S. Ultra-reliable and low-latency communications in 5G downlink: Physical layer aspects. *IEEE Wireless Communications*. 2018;25:124-130. DOI: 10.1109/MWC.2018.1700294
- [5] Petar P, Kasper Floe T, Osvaldo S, Giuseppe D. 5G wireless network slicing for eMBB, URLLC, and mMTC: A communication-theoretic view. *IEEE Access*. 2018;6:55765-55779. DOI: 10.1109/ACCESS.2018.2872781
- [6] 3GPP TS 38.214, V15.3.0, NR; physical layer procedures for data (Release 15). 2018-09
- [7] 3GPP TS 38.212, V15.3.0, NR; multiplexing and channel coding (Release 15). 2018-09
- [8] R1-1713458. Qualcomm Incorporated, CRC attachment. 3GPP TSG RAN Meeting #90; August 21–25, 2017; Prague, Czechia
- [9] Hao W. Hard decision of the zero a posteriori LLR in 5G NR. *Internet Technology Letters*. 2020;3:e146. DOI: 10.1002/itl2.146
- [10] Yan S, Min SK. A table-based algorithm for pipelined CRC calculation. In: *Proceedings of the IEEE International Conference on Communications (ICC'10)*; 23-27 May 2010; Cape Town, South Africa: IEEE; 2010. pp. 1-5
- [11] Hao W, Fang W, Yuqing Y. A distributed CRC early termination scheme for high throughput QC-LDPC codes. In: *Proceedings of International Conference on Wireless Communications and Signal Processing (WCSP) (WCSP '18)*; 18-20 October 2018; Hangzhou, China: IEEE; 2018. pp. 1-5
- [12] TongBi P, Charles Z. High-speed parallel CRC circuits in VLSI. *IEEE Transactions on Communications*. 1992; 40:653-657. DOI: 10.1109/26.141415
- [13] Richard EB. *Algebraic Codes for Data Transmission*. Cambridge, UK: Cambridge University Press; 2003
- [14] Chao C, Keshab KP. High-speed parallel CRC implementation based on unfolding, pipelining, and retiming. *IEEE Transactions on Circuits and Systems II: Express Briefs*. 2006;53: 1017-1021. DOI: 10.1109/TCSII.2006. 882213
- [15] Keshab KP. *VLSI Digital Signal Processing Systems: Design and Implementation*. New York, USA: John Wiley & Sons; 1999
- [16] Ji HM, Killian E. Fast parallel CRC algorithm and implementation on a configurable processor. In: *Proceedings of the IEEE International Conference on Communications (ICC'02)*; 28 April-2 May 2002. New York, USA: IEEE; 2002. pp. 1813-1817
- [17] Hyeji K, Injun C, Wooseok B, Jong-yeol L, Ji-Hoon K. Distributed CRC architecture for high-radix parallel turbo decoding in LTE-advanced systems. *IEEE Transactions on Circuits and Systems II: Express Briefs*. 2015;62:

906-910. DOI: 10.1109/TCSII.2015.  
2435131

[18] Hao W, Tao L, Jin X, Fang W.  
Parallel CRC architecture for broadband  
communication systems. *Electronics  
Letters*. 2017;**53**:1439-1441. DOI:  
10.1049/el.2017.1029

[19] Sarwate DV. Computation of cyclic  
redundancy checks via table look-up.  
*Communications of the ACM*. 1988;**31**:  
1008-1014. DOI: 10.1145/63030.63037

[20] Michael EK, Frank LB. Novel table  
lookup-based algorithms for high-  
performance CRC generation. *IEEE  
Transactions on Computers*. 2008;**57**:  
1550-1560. DOI: 10.1109/TC.2008.85

IntechOpen