3GPP TSG RAN WG1 NR Ad-Hoc Meeting #3 **R1-1715775**
Nagoya, Japan, 18th – 21st September 2017

Agenda item: 6.4.2.3
Source: AccelerComm
Title: On the hardware implementation of frozen bit insertion and removal
Document for: Discussion

I. Introduction

Several polar code sequences were proposed and compared in [1]. Based on this, the following agreement was reached at 3GPP TSG RAN WG1 Meeting #90 [2, AI 6.1.4.2.2].

Agreement:

- Huawei sequence from R1-1712174 is selected.

The Huawei sequence from [1] is defined for a maximum mother code block length of $N_{\max} = 1024$ bits and the sequence $\mathbf{Q}_N$ for a shorter power-of-two mother block length N can be extracted by exploiting the sequence's nested property. For example, the sequence for $N = 64$ is $\mathbf{Q}_{64} =$ [0, 1, 2, 4, 8, 16, 32, 3, 5, 9, 6, 17, 10, 18, 12, 33, 20, 34, 24, 36, 7, 11, 40, 19, 13, 48, 14, 21, 35, 26, 37, 25, 22, 38, 41, 28, 42, 49, 44, 50, 15, 52, 23, 56, 27, 39, 29, 43, 30, 45, 51, 46, 53, 54, 57, 58, 60, 31, 47, 55, 59, 61, 62, 63]. Here, each successive element $Q_N[u]$ (where $u \in [0, N-1]$) of the sequence $\mathbf{Q}_N$ indicates the position (in the range $[0, N-1]$) of the next more reliable uncoded bit of the polar code, where $Q_N[0]$ and $Q_N[N-1]$ give the positions of the least and most reliable bits, respectively. For example, $Q_{64}[5] = 16$ indicates the that bit in position 16 is more reliable than the bits in positions $Q_{64}[0]$ to $Q_{64}[4]$, but less reliable than the bits in positions $Q_{64}[6]$ to $Q_{64}[63]$.

Two polar code rate matching schemes were proposed and compared in [3]. Based on this, the following working assumption was reached at 3GPP TSG RAN WG1 Meeting #90 [2, AI 6.1.4.2.3].

Working Assumption:

- Polar rate matcher: Option 2 from R1-1715000 with corrections of typos:
  - Slide 13: in top part of figure, second "25" -> 26
  - Slide 15: 0.7/16 -> 7/16

Option 2 from [3] defines a sub-block interleaver, which decomposes the polar encoded bits into 32 equal-length sub-blocks, which are reordered according to the interleaver pattern $\boldsymbol{\pi} =$ [0, 1, 2, 4, 3, 5, 6, 7, 8, 16, 9, 17, 10, 18, 11, 19, 12, 20, 13, 21, 14, 22, 15, 23, 24, 25, 26, 28, 27, 29, 30, 31]. Here, each element $\pi[m]$ (where $m \in [0, 31]$) of the interleaver pattern $\boldsymbol{\pi}$ indicates the position (in the range $[0, 31]$) that the interleaved sub-block in position m is sourced from. For example, $\pi[9] = 16$ indicates that the interleaved sub-block in position 9 is sourced from the sub-block that was in position 16 before interleaving. Furthermore, dependent on the uncoded block length K and the encoded block length M, Option 2 from [3] defines rules which govern the selection of the mother code block length N and the selection of puncturing, shortening or repetition. Crucially for this paper, Option 2 from [3] also defines rules which govern the selection of frozen bits, which depends on all of the other aspects of this rate matching scheme.

More specifically, the rate matching scheme influences which of the N uncoded bits are provided by the K information and Cyclical Redundancy Check (CRC) bits. The remaining $N - K$ uncoded bits are provided by frozen bits, which may be scrambled by User Equipment Identification (UE-ID) bits. In the absence of rate matching, the positions of the K information and CRC bits would be selected by using the sequence $\mathbf{Q}_N$ to identify the K uncoded bits having the highest reliability, with all other uncoded

bits becoming frozen. However, when rate matching is employed, this requires a set of frozen bits to be identified independently of and before applying the sequence. Following this, the K information and CRC bits are positioned within the remaining uncoded bits by using the sequence $\mathbf{Q}_N$ to identify those having the highest reliability, with all other remaining uncoded bits becoming frozen.

This paper proposes hardware implementations that can perform the frozen bit insertion and removal processes for several bits at a time, allowing them to be completed using a small number of clock cycles. More specifically, this allows frozen bits to be interlaced with information bits and CRC bits, before polar encoding. Likewise, this allow the frozen bits to be deinterlaced from the information and CRC bits, following polar decoding. The proposed approach may also be adapted to interlace and deinterlace Parity Check (PC) bits. The proposed hardware implementations do not require circuits for sorting, interleaving or performing other complex operations, nor do they require an excessive amount of Read Only Memory (ROM) for storing pre-computed frozen bit positions or intermediate variables. The proposed hardware implementations are detailed in Section II. Conclusions are offered in Section III.

II. Proposed hardware implementations for frozen bit insertion and removal

This section describes the proposed hardware implementations for frozen bit insertion and removal. During a first sub-process, the proposed hardware implementations consider w_Q uncoded bit positions at a time in order of decreasing reliability, considering whether each successive uncoded bit is frozen by rate matching. This continues until K number of bits that are not frozen by rate matching have been found, whereupon the reliability of the Kth-most reliable unfrozen bit is determined and referred to as the threshold reliability. During a second sub-process, w_R uncoded bit positions are considered at a time in their natural order. Each of the w_R uncoded bit positions is determined to be an information or CRC bit if its reliability is no less than the threshold reliability and if it is not frozen by rate matching, otherwise it is determined to be a frozen bit. In this way, a bit pattern is generated w_R bits at a time throughout the second sub-process, which identifies whether each uncoded bit is an information or CRC bit, or if it is a frozen bit. At the same time, the bit pattern may be used to interlace or deinterlace w_R uncoded bits at a time in their natural order. More specifically, the information and CRC bits may be interlaced with the frozen bits throughout the second sub-process, in order to implement frozen bit insertion during polar encoding. Likewise, the information and CRC bits may be deinterlaced from the frozen bits throughout the second sub-process, in order to implement frozen bit removal during polar decoding.

The proposed hardware implementations for frozen bit insertion and removal are detailed in the schematic of Figure 1, where the top and bottom halves corresponds to the first and second sub-processes, respectively. This schematic includes four sets of ROMs, as detailed in Section II-A. The operation of these ROMs and the logic shown in Figure 1 is coordinated by the controller, as detailed in Section II-B.

A. ROMs

As shown in Figure 1, the proposed hardware implementations employ four sets of ROMs, as follows.

- A set of reversed sequence ROMs stores the set of reversed sequences $\{\mathbf{Q}_{32}^{\leftarrow}, \mathbf{Q}_{64}^{\leftarrow}, \mathbf{Q}_{128}^{\leftarrow}, \ldots, \mathbf{Q}_{1024}^{\leftarrow}\}$. Here, each successive element $Q_N^{\leftarrow}[u] = Q_N[N-u-1]$ (where $u \in [0, N-1]$) of the reversed sequence $\mathbf{Q}_N^{\leftarrow}$ indicates the position (in the range $[0, N-1]$) of the next less reliable uncoded bit of the polar code, where $Q_N^{\leftarrow}[0]$ and $Q_N^{\leftarrow}[N-1]$ give the positions of the most and least reliable bits, respectively.
- A set of rank ROMs stores a set of rank sequences $\{\mathbf{R}_{32}, \mathbf{R}_{64}, \mathbf{R}_{128}, \ldots, \mathbf{R}_{1024}\}$. Here, each element $R_N[u]$ (where $u \in [0, N-1]$) of the rank sequence $\mathbf{R}_N$ indicates the reliability ranking (in the range $[0, N-1]$) of the corresponding uncoded bit of the polar code, where a lower value $R_N[u]$ indicates a higher reliability. For example, $R_N[u_1] = 0$ and $R_N[u_2] = N-1$ indicate that the uncoded bits u_1 and u_2 are the most and least reliable bits, respectively. The relationship between the reversed sequence $\mathbf{Q}_N^{\leftarrow}$ and the rank sequence $\mathbf{R}_N$ is such that $Q_N^{\leftarrow}[R_N[u]] = u$.

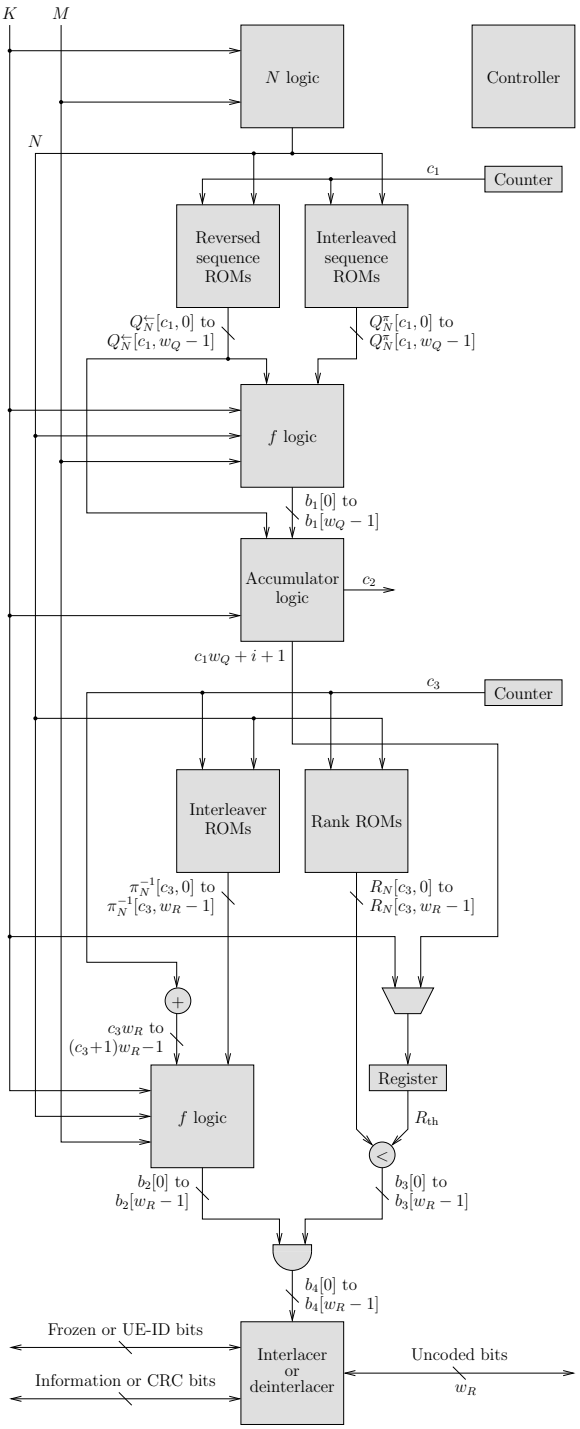

Fig. 1: Schematic of the proposed hardware implementations for frozen bit insertion and removal.

- A set of deinterleaver ROMs stores a set of deinterleaver patterns $\{\boldsymbol{\pi}_{32}^{-1}, \boldsymbol{\pi}_{64}^{-1}, \boldsymbol{\pi}_{128}^{-1}, \ldots, \boldsymbol{\pi}_{1024}^{-1}\}$. Here, each element $\pi_N^{-1}[u]$ (where $u \in [0, N-1]$) of the deinterleaver pattern $\boldsymbol{\pi}_N^{-1}$ indicates the position (in the range $[0, N-1]$) that the polar encoded bit in position u is interleaved to, during rate matching. The relationship between the deinterleaver pattern $\boldsymbol{\pi}_N^{-1}$ and the interleaver pattern $\boldsymbol{\pi}$ is such that $\pi[\lfloor \pi_N^{-1}[u] \cdot 32/N \rfloor] = \lfloor u \cdot 32/N \rfloor$. Furthermore, all elements $\pi_N^{-1}[u]$ in $\boldsymbol{\pi}_N^{-1}$ that evaluate to the same value of $\lfloor \pi_N^{-1}[u] \cdot 32/N \rfloor$ appear in consecutive positions within $\boldsymbol{\pi}_N^{-1}$, in ascending order.
- A set of interleaved sequence ROMs stores a set of interleaved sequences $\{\mathbf{Q}_{32}^{\pi}, \mathbf{Q}_{64}^{\pi}, \mathbf{Q}_{128}^{\pi}, \ldots, \mathbf{Q}_{1024}^{\pi}\}$. Here, each element $Q_N^{\pi}[u]$ of the interleaved sequence $\mathbf{Q}_N^{\pi}$ is obtained as $Q_N^{\pi}[u] = \pi_N^{-1}[Q_N^{\leftarrow}[u]]$.

Each address in each reversed sequence ROM and each interleaved sequence ROM stores w_Q elements of the respective sequences, where w_Q is a power of two. More specifically, each successive group of w_Q consecutive elements of each reversed sequence $\mathbf{Q}_N^{\leftarrow}$ are stored in successive addresses of the corresponding reversed sequence ROM, as exemplified for $N = 64$ and $w_Q = 8$ in Table I. Likewise, each successive group of w_Q consecutive elements of each interleaved sequence $\mathbf{Q}_N^{\pi}$ are stored in successive addresses of the corresponding interleaved sequence ROM, as exemplified for $N = 64$ and $w_Q = 8$ in Table II. More specifically, each element in these ROMs is obtained according to $Q_N^{\leftarrow}[c, i] = Q_N^{\leftarrow}[c \cdot w_Q + i]$ and $Q_N^{\pi}[c, i] = Q_N^{\pi}[c \cdot w_Q + i]$, where $c \in [0, N/w_Q - 1]$ is the corresponding address and $i \in [0, w_Q - 1]$ is the index of the element within that address.

By contrast, each address in each deinterleaver ROM and each rank ROM stores w_R elements of the respective sequences, where w_R is a power of two that may be selected independently of w_Q. More specifically, each successive group of w_R consecutive elements of each deinterleaver pattern $\boldsymbol{\pi}_N^{-1}$ are stored in successive addresses of the corresponding deinterleaver ROM, as exemplified for $N = 64$ and $w_R = 4$ in Table III. Likewise, each successive group of w_R consecutive elements of each rank sequence $\mathbf{R}_N$ are stored in successive addresses of the corresponding rank ROM, as exemplified for $N = 64$ and $w_R = 4$ in Table IV. More specifically, each element in these ROMs is obtained according to $\pi_N^{-1}[c, i] = \pi_N^{-1}[c \cdot w_R + i]$ and $R_N[c, i] = R_N[c \cdot w_R + i]$, where $c \in [0, N/w_R - 1]$ is the corresponding address and $i \in [0, w_R - 1]$ is the index of the element within that address.

Note that in cases where $N < w_Q$ or $N < w_R$, each sequence stored in a corresponding ROM may be appended with $w_Q - N$ or $w_R - N$ dummy elements having the value $N - 1$, in order to fill a single address of the ROM. Note that rather than storing sequences of the same type in separate ROMs corresponding to each supported value of N, these sequences could be stored within different address spaces of a single larger ROM. In this case, the value of N may be used to index a lookup table, which identifies the start address of the corresponding sequence.

	index i							
address c	0	1	2	3	4	5	6	7
0	63	62	61	59	55	47	31	60
1	58	57	54	53	46	51	45	30
2	43	29	39	27	56	23	52	15
3	50	44	49	42	28	41	38	22
4	25	37	26	35	21	14	48	13
5	19	40	11	7	36	24	34	20
6	33	12	18	10	17	6	9	5
7	3	32	16	8	4	2	1	0

TABLE I: Elements $Q_N^{\leftarrow}[c, i]$ of the reversed sequence ROM for $N = 64$ and $w_Q = 8$, where $i \in [0, w_Q - 1]$ and $c \in [0, N/w_Q - 1]$.

Assuming that all entries in the ROMs are stored using fixed point numbers having a width of $\log_2(N_{\max}) = 10$ bits, the total capacity required for the ROMs to store all sequences $\mathbf{Q}_N^{\leftarrow}$, $\mathbf{Q}_N^{\pi}$, $\boldsymbol{\pi}_N^{-1}$ and $\mathbf{R}_N$ for $N \in \{32, 64, 128, \ldots, 1024\}$ is 78.75 kbit. Alternatively, the total capacity required can be reduced to 71.62 kbit, if different widths of $\log_2(N)$ bits are used to store the fixed-point numbers for different values of N.

	index i							
address c	0	1	2	3	4	5	6	7
0	63	62	61	59	57	47	45	60
1	58	55	56	53	46	51	43	44
2	39	41	31	37	54	29	52	15
3	50	42	49	38	40	35	30	28
4	33	27	36	23	25	14	48	13
5	21	34	11	9	26	32	22	24
6	19	12	20	10	17	8	7	5
7	3	18	16	6	4	2	1	0

TABLE II: Elements $Q_N^{\pi}[c,i]$ of the interleaved sequence ROM for $N = 64$ and $w_Q = 8$, where $i \in [0, w_Q - 1]$ and $c \in [0, N/w_Q - 1]$.

	index i			
address c	0	1	2	3
0	0	1	2	3
1	4	5	8	9
2	6	7	10	11
3	12	13	14	15
4	16	17	20	21
5	24	25	28	29
6	32	33	36	37
7	40	41	44	45
8	18	19	22	23
9	26	27	30	31
10	34	35	38	39
11	42	43	46	47
12	48	49	50	51
13	52	53	56	57
14	54	55	58	59
15	60	61	62	63

TABLE III: Elements $\pi_N^{-1}[c,i]$ of the deinterleaver ROM for $N = 64$ and $w_R = 4$, where $i \in [0, w_R - 1]$ and $c \in [0, N/w_R - 1]$.

	index i			
address c	0	1	2	3
0	63	62	61	56
1	60	55	53	43
2	59	54	51	42
3	49	39	37	23
4	58	52	50	40
5	47	36	31	21
6	45	32	34	19
7	28	17	15	6
8	57	48	46	35
9	44	33	30	18
10	41	29	27	16
11	25	14	12	5
12	38	26	24	13
13	22	11	10	4
14	20	9	8	3
15	7	2	1	0

TABLE IV: Elements $R_N[c,i]$ of the rank ROM for $N = 64$ and $w_R = 4$, where $i \in [0, w_R - 1]$ and $c \in [0, N/w_R - 1]$.

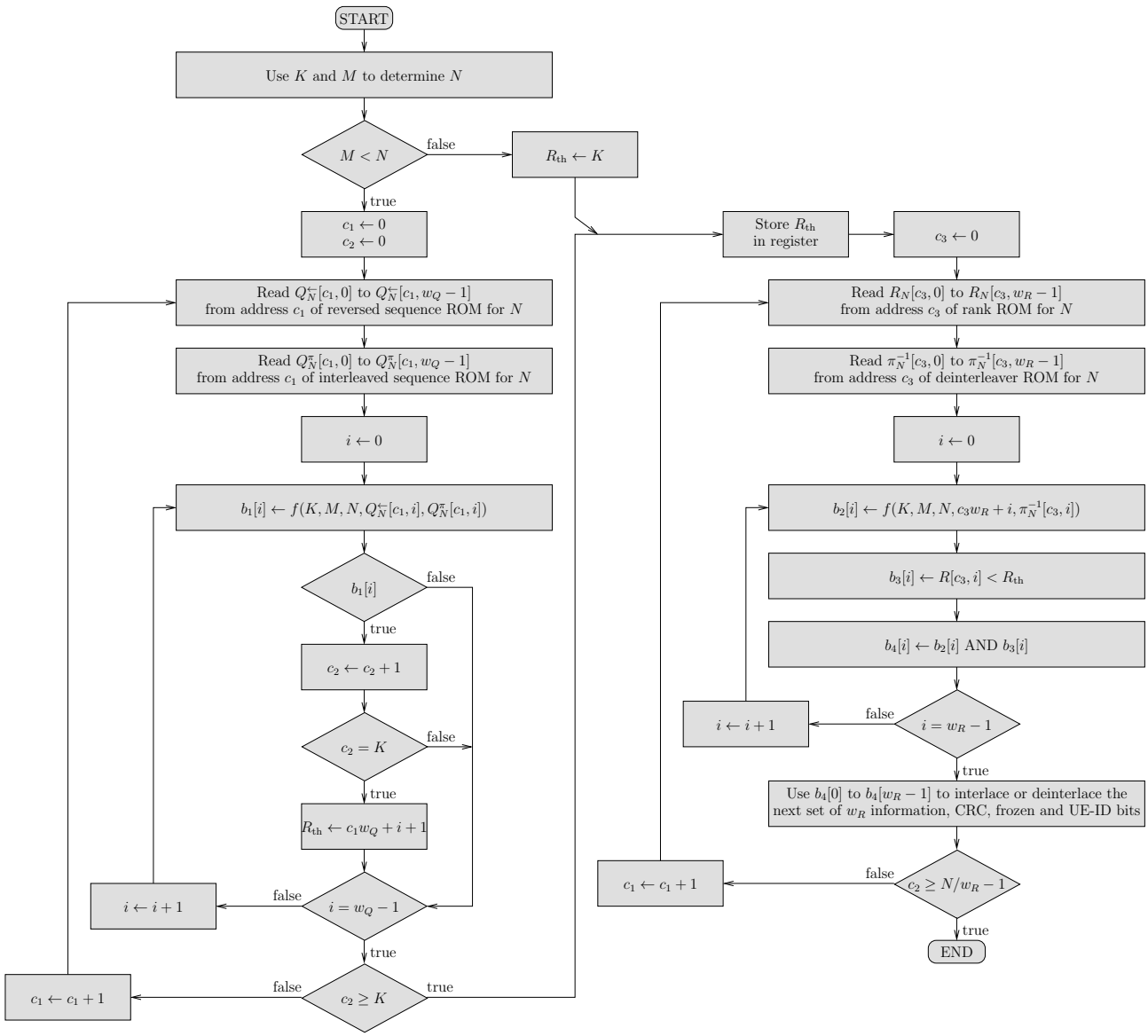

Fig. 2: Flow chart of the proposed hardware implementations for frozen bit insertion and removal.

B. Logic and controller

As shown in Figure 1, the proposed hardware implementations for frozen bit insertion and removal comprise four sets of ROMs and various logic circuits. These operate under the coordination of the controller shown in Figure 1, according to the flowchart of Figure 2. As described above, the proposed hardware implementations complete the processes of frozen bit insertion or removal using two sub-processes, which correspond to the left and right halves of Figure 2.

At the beginning of the first sub-process, the N logic of Figure 1 is used to compute the mother code block size N, as a function of the number K of information and CRC bits, as well as of the number M of polar encoded bits that remain after rate matching. As shown in Figure 2, if $M < N$ is not satisfied, then the first sub-process can be immediately concluded by setting the rank threshold R_{th} equal to K, where R_{th} implements the reliability threshold mentioned above. Otherwise, the first sub-process must use further computations in order to determine the rank threshold R_{th}.

In this case, the controller resets the counters c_1 and c_2 shown in Figure 1 to zero. In successive clock cycles, successive addresses of the reversed sequence ROM and the interleaved sequence ROM

corresponding to the particular value of N are indexed using the counter c_1, which is incremented in each clock cycle. As shown in Figures 1 and 2, the w_Q consecutive elements $Q_N^{\leftarrow}[c_1, 0]$ to $Q_N^{\leftarrow}[c_1, w_Q - 1]$ and $Q_N^{\pi}[c_1, 0]$ to $Q_N^{\pi}[c_1, w_Q - 1]$ of the reversed sequence $\mathbf{Q}_N^{\leftarrow}$ and the interleaved sequence $\mathbf{Q}_N^{\pi}$ are read from the reversed sequence ROM and the interleaved sequence ROM, respectively.

Each successive set of elements read from the reversed sequence and interleaved sequence ROMs in each successive clock cycle is provided to the first set of f logic shown in Figure 1. As shown in Figure 2, this f logic obtains a set of w_Q binary flags by computing $b_1[i] = f(K, M, N, Q_N^{\leftarrow}[c_1, i], Q_N^{\pi}[c_1, i])$ for each value of $i \in [0, w_Q - 1]$ in parallel, where

$$\begin{aligned} f(K, M, N, u, \pi_N^{-1}[u]) \;=\; & M \geq N \text{ OR } \left(\frac{K}{M} > \frac{7}{16} \text{ AND } \pi_N^{-1}[u] < M\right) \\ & \text{OR } \left(\frac{K}{M} \leq \frac{7}{16} \text{ AND } \pi_N^{-1}[u] \geq N - M \right. \\ & \text{AND } \left(\left(M \geq \frac{3N}{4} \text{ AND } u \geq \left\lceil \frac{3N}{4} - \frac{M}{2} \right\rceil\right)\right. \\ & \left.\left.\text{OR } \left(M < \frac{3N}{4} \text{ AND } u \geq \left\lceil \frac{9N}{16} - \frac{M}{4} \right\rceil\right)\right)\right). \end{aligned} \tag{1}$$

The binary flags $b_1[0]$ to $b_1[w_Q - 1]$ obtained in each clock cycle are provided to the accumulator logic shown in Figure 1. As shown in Figure 2, this uses an index i which is initially set to 0 and is incremented towards $w_Q - 1$, in order to consider the binary flags in order from $b_1[0]$ to $b_1[w_Q - 1]$. At the same time, the counter c_2 is incremented once for each of the binary flags having the value 1. When the counter c_2 reaches the value K, the threshold rank R_{th} is set equal to $c_1 w_Q + i + 1$, whereupon the first sub-process is completed. More specifically, the first sub-process continues through successive clock cycles until $c_2 \geq K$ is satisfied, which will typically occur before c_1 reaches the index of the final address of the reversed sequence and interleaved sequence ROMs.

As shown in Figures 1 and 2, the threshold rank R_{th} is stored in a register, so that it can be used throughout the second sub-process. At the start of the second sub-process, the controller resets the counter c_3 shown in Figure 1 to zero. In successive clock cycles, successive addresses of the deinterleaver ROM and the rank ROM corresponding to the particular value of N are indexed using the counter c_3, which is incremented in each clock cycle. As shown in Figures 1 and 2, the w_R consecutive elements $\pi_N^{-1}[c_3, 0]$ to $\pi_N^{-1}[c_3, w_R - 1]$ and $R_N[c_3, 0]$ to $R_N[c_3, w_R - 1]$ of the deinterleaver pattern $\boldsymbol{\pi}_N^{-1}$ and the rank sequence $\mathbf{R}_N$ are read from the deinterleaver ROM and the rank ROM, respectively.

Each successive set of elements read from the deinterleaver ROM in each successive clock cycle is provided to the second set of f logic shown in Figure 1. Note that since the first and second sets of f logic are not used simultaneously, they may share the same hardware by multiplexing between the inputs provided in the first sub-process and those provided in the second sub-process. As shown in Figure 2, this f logic obtains a set of w_R binary flags by computing $b_2[i] = f(K, M, N, c_3 w_R + i, \pi_N^{-1}[c_3, i])$ of (1) for each value of $i \in [0, w_R - 1]$ in parallel. At the same time, each successive set of elements read from the rank ROM in each successive clock cycle is provided to the set of w_R comparators shown in Figure 1. As shown in Figure 2, these comparators obtain a set of w_R binary flags by computing $b_3[i] = R_N[c_3, i] < R_{\text{th}}$ for each value of $i \in [0, w_R - 1]$ in parallel. Then, the binary flags $b_2[0]$ to $b_2[w_R - 1]$ and $b_3[0]$ to $b_3[w_R - 1]$ are provided to a set of w_R AND gates, which obtain a set of w_R binary flags by computing $b_4[i] = (b_2[i] \text{ AND } b_3[i])$ for each value of $i \in [0, w_R - 1]$ in parallel, as shown in Figure 2. Tables V to VII illustrate the bit patterns $b_4[0]$ to $b_4[w_R - 1]$ that are generated in each clock cycle of the second sub-process, for examples in which repetition, shortening and puncturing are used.

In each successive clock cycle of the second sub-process, the bit pattern $b_4[0]$ to $b_4[w_R - 1]$ may be used to interlace or deinterlace each successive set of w_R uncoded bits in parallel, as shown in Figures 1 and 2. Each of the bits in the bit pattern $b_4[0]$ to $b_4[w_R - 1]$ having the value 1 indicates that the corresponding uncoded bit is provided by an information or CRC bit. Likewise, each of the bit pattern bits having

clock cycle c_3	bit pattern $b_4[0]$	$b_4[1]$	$b_4[2]$	$b_4[3]$
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	1
4	0	0	0	0
5	0	0	1	1
6	0	0	0	1
7	1	1	1	1
8	0	0	0	0
9	0	0	1	1
10	0	1	1	1
11	1	1	1	1
12	0	1	1	1
13	1	1	1	1
14	1	1	1	1
15	1	1	1	1

TABLE V: Elements of the bit pattern generated in each of the $N/w_R = 16$ clock cycles of the second sub-process for $K = 32$, $M = 68$, $N = 64$ and $w_R = 4$. In this case, repetition is used and $R_{\text{th}} = 32$. Since $M < N$ is not satisfied, no clock cycles are used to complete the first sub-process, irrespective of w_Q.

clock cycle c_3	bit pattern $b_4[0]$	$b_4[1]$	$b_4[2]$	$b_4[3]$
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	1	1	1
4	0	0	0	0
5	0	1	1	1
6	0	1	1	1
7	1	1	1	1
8	0	0	0	1
9	0	1	1	1
10	0	1	1	1
11	1	1	1	1
12	1	1	1	1
13	1	1	0	0
14	1	1	0	0
15	0	0	0	0

TABLE VI: Elements of the bit pattern generated in each of the $N/w_R = 16$ clock cycles of the second sub-process for $K = 32$, $M = 56$, $N = 64$ and $w_R = 4$. In this case, shortening is used and $R_{\text{th}} = 40$. When $w_Q = 8$, five clock cycles are used to complete the first sub-process.

the value 0 indicates that the corresponding uncoded bit is a frozen bit, which may be scrambled by the UE-ID. During polar encoding, the interlacer of Figure 1 operates on the basis of First-In First-Out (FIFO) buffering. In each clock cycle, an input FIFO buffer supplies a number of information and CRC bits equal to the number of 1s in the corresponding bit pattern. Meanwhile, a second input FIFO buffer supplies a number of UE-ID scrambled frozen bits equal to the number of 0s in the bit pattern. Alternatively, if UE-ID scrambling is not used and all frozen bits adopt a value of 0, then the second FIFO buffer can be replaced with a circuit that supplies the corresponding number of 0-valued bits. The interlacer of Figure 1 may then interlace the information, CRC and frozen bits according to the corresponding bit pattern, producing w_R number of uncoded bits in parallel, in each clock cycle of the second sub-process. Likewise, during polar decoding, the deinterlacer of Figure 1 may perform the reverse operation for w_R number of uncoded bits in each clock cycle, where the information and CRC bits are provided to an output FIFO buffer.

clock cycle c_3	bit pattern $b_4[0]$	$b_4[1]$	$b_4[2]$	$b_4[3]$
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	0	0	0	1
6	0	0	0	1
7	0	1	1	1
8	0	0	0	0
9	0	0	0	1
10	0	0	0	1
11	0	1	1	1
12	0	0	1	1
13	1	1	1	1
14	1	1	1	1
15	1	1	1	1

TABLE VII: Elements of the bit pattern generated in each of the $N/w_R = 16$ clock cycles of the second sub-process for $K = 24$, $M = 56$, $N = 64$ and $w_R = 4$. In this case, puncturing is used and $R_{\text{th}} = 25$. When $w_Q = 8$, four clock cycles are used to complete the first sub-process.

The total number of clock cycles required to complete the frozen bit insertion and removal processes is given by the sum of the number used in each of the first and second sub-processes. Figure 3 characterises the number of clock cycles required to complete the first sub-process as a function of K and M, for the worst case where $w_Q = 1$. When w_Q adopts the value of a higher power of two, the number of clock cycles required may be obtained by linearly scaling down those of Figure 3 and taking the ceiling. It may be observed that greater numbers of clock cycles are required at coding rates of $K/M > 7/16$, where shortening is employed. This is because shortening uses some of the most reliable uncoded bit positions for frozen bits. A smaller number of clock cycles is required when employing puncturing, since this typically uses the least reliable bit positions for frozen bits. More specifically, the number of clock cycles used in the first sub-process with $w_Q = 1$ is equal to R_{th} in the case of shortening or puncturing. By contrast, no clock cycles are required when employing repetition, as described above. Note however that the first sub-process may be completed in parallel with CRC generation and interleaving during polar encoding and in parallel with channel interleaving during polar decoding. Owing to this, the first sub-process does not necessarily impose additional latency. The number of clock cycles required to complete the second sub-process is given by $\lceil N/w_R \rceil$, as characterised in Figure 4, for the worst case where $w_R = 1$. When w_R adopts the value of a higher power of two, the number of clock cycles required may be obtained by linearly scaling down those of Figure 4 and taking the ceiling. The second sub-process can stream uncoded bits into a polar encoder kernal or stream uncoded bits out of a polar decoder kernal alongside their operation, without imposing additional latency.

III. Conclusions

In this paper, we have proposed hardware implementations that can perform the frozen bit insertion and removal processes for several bits at a time, allowing them to be completed using a small number of clock cycles. More specifically, this allows frozen bits (which may be scrambled using UE-ID bits) to be interlaced with information bits and CRC bits, before polar encoding. Likewise, this allow the frozen bits to be deinterlaced from the information and CRC bits, following polar decoding. The proposed hardware implementations do not require circuits for sorting, interleaving or performing other complex operations, nor do they require an excessive amount of ROM for storing pre-computed frozen bit positions or intermediate variables. All operations of the proposed hardware implementations can be performed alongside other polar encoding or decoding operations and so they do not impose any additional latency.

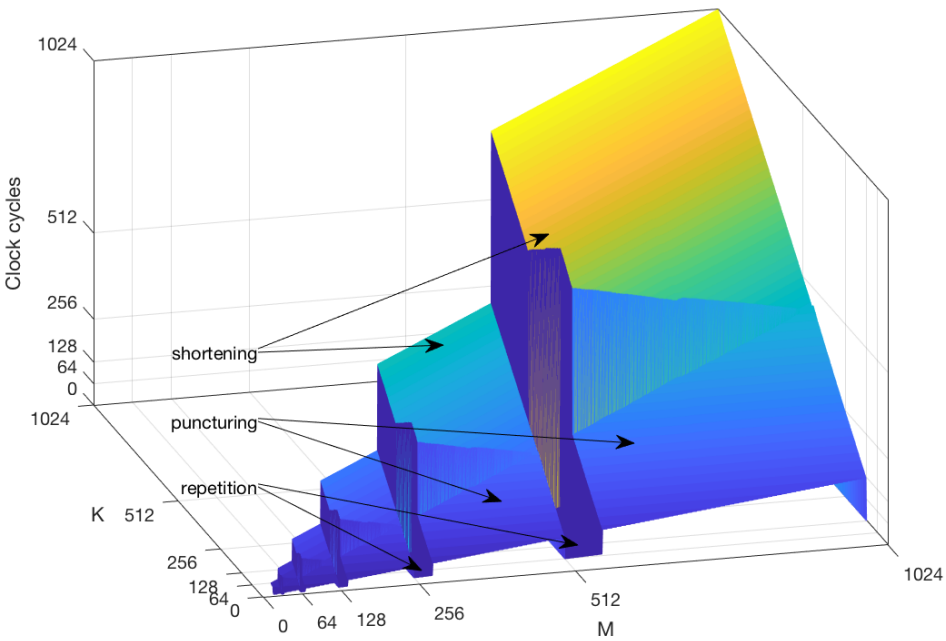

Fig. 3: Plot of the number of clock cycles required by the first sub-process as a function of $M \in [17, 1024]$ and $K \in [\lceil M/8 \rceil, M-1]$, for the worst case where $w_Q = 1$. When w_Q adopts the value of a higher power of two, these numbers of clock cycles may be linearly scaled down and then rounded up to the nearest integer.

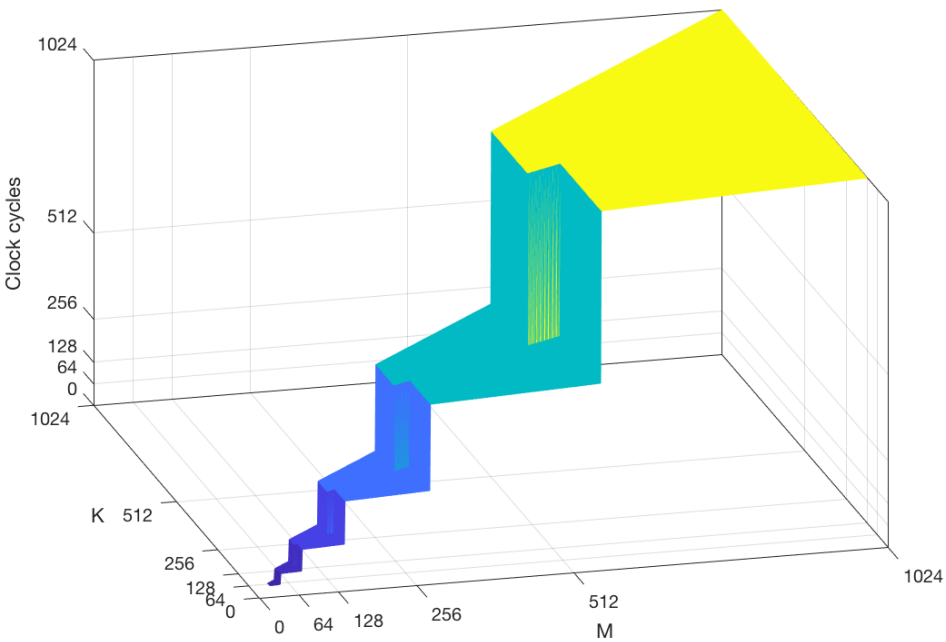

Fig. 4: Plot of the number of clock cycles required by the second sub-process as a function of $M \in [17, 1024]$ and $K \in [\lceil M/8 \rceil, M-1]$, for the worst case where $w_R = 1$. When w_R adopts the value of a higher power of two, these numbers of clock cycles may be linearly scaled down and then rounded up to the nearest integer.

Observation 1: The frozen bit insertion and removal processes associated with rate matching Option 2 from [3] do not impose significant complexity or additional latency.

Proposal 1: If no problems are identified with Option 2 from [3] in other areas, then agree the related working assumption.

REFERENCES

[1] Huawei, "Summary of email discussion [NRAH2-11] Polar code sequence," in *3GPP TSG RAN WG1 Meeting #90*, Prague, Czech Republic, August 2017, R1-1712174.

[2] MCC Support, "Draft Report of 3GPP TSG RAN WG1 #90 v0.1.0," in *3GPP TSG RAN WG1 Meeting #90*, Prague, Czech Republic, August 2017.

[3] MediaTek, Qualcomm, Samsung, ZTE, "Way Forward on Rate-Matching for Polar Code," in *3GPP TSG RAN WG1 Meeting #90*, Prague, Czech Republic, August 2017, R1-1715000.