Extended Barrel-Shifter for Versatile QC-LDPC Decoders

Emmanuel Boutillon, Hassan Harb

Extended Barrel-Shifter for Versatile QC-LDPC Decoders

Emmanuel Boutillon *Senior Member, IEEE,* and Hassan Harb
Université de Bretagne-Sud, Lab-STICC, UMR 6285 CNRS – Lorient, France
emmanuel.boutillon@univ-ubs.fr and hassan.harb@univ-ubs.fr

***Abstract*—In this paper, we present a Barrel-Shifter of size n extended by an additional layer that can handle any circular permutation on a vector of size m, $m \leq n$, thanks to a specific initial positioning of the data. The construction of the so-called Extended Barrel-Shifter is motivated by the hardware decoder constraint related to the Low-Density Parity-Check (LDPC) code recently adopted for the 5G mobile standard. The proposed algorithm requires 42 % less multiplexers than the best state-of-the-art solution for n = 384 of the 5G LDPC standard. This proposal is also able to process several inputs in parallel without extra hardware cost.**

***Index Terms*—Barrel-Shifter, circular-shift network, LDPC decoder, 5G standard.**

I. Introduction

A Barrel-Shifter (BS) is a hardware component that allows to perform any circular rotation on a vector of size n. BS's are key components of Quasi-Cyclic Low-Density Parity-Check (QC-LDPC) decoder [1]. In fact, QC-LDPC codes are hardware friendly error control codes based on a prototype matrix where each null element is replaced by an $n \times n$ null matrix and each non-null element is replaced by an $n \times n$ circularly shifted identity matrix. This particular structure allows implementing a decoder with n Processing Elements (PE) in parallel without any memory access conflict, as shown in Fig. 1. The BS of size n inserted between the memories and the PE allows the reordering of the data sent from the RAM's to PE's, according to the structure of the QC-LDPC code.

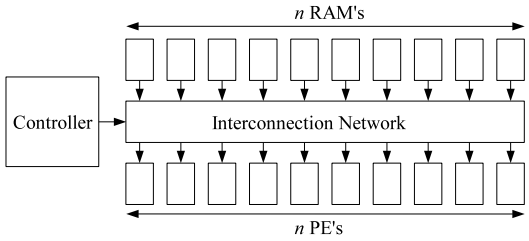

Fig. 1. Parallel structure of a QC-LDPC decoder.

Recently, a new family of QC-LDPC codes has been adopted for the next 3GPP standard (the so-called 5G) [2]. The particularity of this standard is that the expansion factor sizes of the QC-LDPC code can take 51 values ranging between $m = 2$ and $m = 384$ according to the code rate and the size of the code. This new characteristic translates in hardware by architectures that are able to perform rotations on vectors of variable size. The idea is to set the number n of PE's to the maximum expansion factor of the code to achieve the maximum decoding throughput. When the effective expansion factor m of the QC-LDPC code is smaller than n, then only m PE's among the n PE's are used. This property is also required for a design of a versatile QC-LDPC decoder able to process several types of codes (WIFI, DVB-S2 for example).

A Benes Network (BN) of size n, with n a power of 2 is able to perform any kind of permutation [3]. In [4], a network inspired from the Benes's network is proposed to perform permutations for any value of n without extra cost. However, since only a reduced subset of permutations (i.e., rotations) is required, those solutions are not optimal. The BN requires a total of D_{BN} layers of multiplexers (MUX's) in its critical path and a hardware complexity given by C_{BN} MUX's with

$$D_{\text{BN}} = 2\log_2(n) - 1, \tag{1}$$

$$C_{\text{BN}} = (2\log_2(n) - 1)n. \tag{2}$$

This high number of layers introduces a high propagation delay, thus reducing the clock frequency of the design. Pipelining the BN has also its own drawback, since it introduces latency in a processing loop. The work in [5] proposes to prune the BN in order to tailor it exactly to the requirement of the application. The resulting architecture can be significantly smaller than the initial BN when the number of required expansion factor sizes is low, which is not the case for the 5G standard. Moreover, the resulting structure inherits from the long path delay of the mother BN. For a low number of distinct sizes, [6] proposed a two stage network where the first stage is composed of an n/d parallel BS's, each of size d, with d the greatest common divisor of all the required sizes. This method is effective when d is high enough but of little help for the 5G where $d = 1$ (expansion factors of 2 and 3 are both required).

The best reference is given in [7] where a new architecture of a flexible BS called Quasi-Cyclic LDPC Shift Network (QSN) is proposed. When the vector size m is smaller than n, QSN proposes to select the first m PE's and the first m RAM's of the architecture and put the other $n-m$ PE's and RAM's in the idle mode (see Fig. 2.a). The QSN interconnection network is then able to process any cyclic rotation between PE's and RAM's of index $0, 1, \ldots, m-1$, $m \leq n$ thanks to a clever association of two partial size-n BS's and a final MUX's stage. Its overall complexity, C_{QSN} in terms of MUX's is given as

$$C_{\text{QSN}} = 2(\lceil \log_2(n) \rceil + 1)n - 2^{\lceil \log_2(n) \rceil + 1} + 1, \tag{3}$$

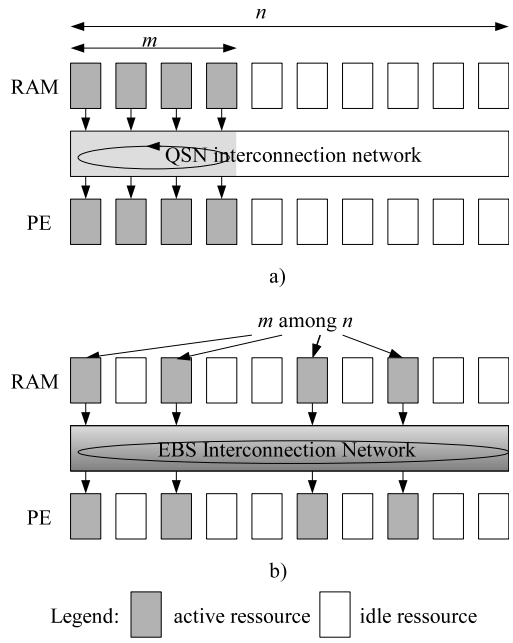

Fig. 2. Comparison between QSN and EBS principles. a) with QSN, only the first m PE are used. b) With EBS, the m active PE's are spread regularly among the n PE

.

where $\lceil x \rceil$ represents the smallest integer greater than or equal to x. In summary, QSN roughly doubles the complexity compared to a simple BS while its number of layers, D_{QSN}, is equal to the number of layers of a classical BS plus one, i.e.,

$$D_{\text{QSN}} = \lceil \log_2(n) \rceil + 1. \tag{4}$$

The hardware cost and the critical path of QSN is not negligible with regards to the other blocks. For example, for the 5G LDPC decoder, a combinational QSN costs roughly 80% of the area required for the PE's, moreover, without pipeline layers, the QSN limits the maximum clock frequency due to its long critical path.

In this paper, we tackle the same problem with a new approach. In fact, nothing prevents the designer to select any subset of m PE's and m memories to relax the constraint on the design. Based on this idea, we propose a solution called Extended Barrel-Shifter (EBS) composed of a classical BS of size n extended with an extra layer of MUX's. Fig. 2.b illustrates the principle of EBS compared to the QSN. Note that the EBS has exactly the same number of layers than the QSN but it significantly reduces the number of MUX's (from 38.3 % for $n = 48$ up to 44.5 % for $n = 512$). Moreover, EBS also offers the possibility to work in parallel with T vectors of size m, provided that $Tm \leq n$.

The rest of the paper is divided into three sections. Section II presents the principle of the EBS. Section III describes the hardware architecture and compares EBS with the state of the art. Finally Section IV concludes the paper.

II. Extended Barrel-Shifter

In this section we first mathematically describe the EBS, and then we explain its extension for the parallel processing of several vectors.

A. Formal description of EBS

Let X^0 be a vector of size $m \leq n$, i.e., $X^0 = (x_0^0, x_1^0, \ldots, x_{m-1}^0)$. The rotation R^p of index p applied on X^0 to generate $X^p = R^p(X^0)$ is the permutation that shifts the i^{th} element x_i^0 of X^0 to the position $(i+p) \mod n$ in X^p, thus

$$X^p = R^p(X^0) = (x_{(-p)}^0, x_{(-p+1)}^0, \ldots, x_{(m-p-1)}^0), \tag{5}$$

where $0 \leq p < m$ and operations of the indexes are performed modulo m, e.g. $x_{(-p)}^0$ refers to x_{m-p}^0 when $p > 0$.

The proposed EBS allows to perform the operation $R^p(X^0)$ for any value of m, $1 < m \leq n$ and p, $0 \leq p < m$ thanks to operations on a vector of size n. To simplify notations, in the sequel, X refers always to a vector of size m and Y and Z to vectors of size n. Moreover, rotation on a vector of size n will be denoted with a bar as $\bar{R}$. Mathematically, EBS can be defined by the three following functions:

1) Initial mapping: An injective map π_m between X^0 and Z^0 defined as

$$Z^0 = \pi_m(X^0). \tag{6}$$

This map associates the i^{th} coordinate x_i^0 of X^0 to the $\pi_m(i)$ coordinate $z_{\pi_m(i)}^0$ of Z^0, i.e., $z_{\pi_m(i)}^0 = x_i^0$ with

$$\pi_m(i) = \lfloor in/m \rfloor, i = 0, 1, \ldots, m-1, \tag{7}$$

where $\lfloor a \rfloor$ is the floor function that returns the highest integer smaller than or equal to a. For example, for $n = 8$ and $m = 5$, the vector $X^0 = (x_0^0, x_1^0, x_2^0, x_3^0, x_4^0)$ is mapped into Z^0 as $Z^0 = (x_0^0, x_1^0, \varnothing, x_2^0, x_3^0, \varnothing, x_4^0, \varnothing)$ where $\varnothing$ stands for 'idle', or equivalently, unused position. Note that the mapping spread regularly the value on X^0 into the vector Z^0.

We also define Z^p as the result of the mapping π_m applied on X^p, i.e.,

$$Z^p = \pi_m(R^p(X^0)). \tag{8}$$

Back to the previous example, $p = 3$ would lead to $Z^3 = \pi_5(R^3(X^0)) = \pi_5((x_2^0, x_3^0, x_4^0, x_0^0, x_1^0)) = (x_2^0, x_3^0, \varnothing, x_4^0, x_0^0, \varnothing, x_1^0, \varnothing)$.

2) Rotation: The rotation $\bar{R}^{\bar{p}}$ of index $\bar{p} = \pi_m(p)$ applied to Z^0 is given by

$$Y^p = \bar{R}^{\pi_m(p)}(Z^0). \tag{9}$$

Back to the previous example and considering $p = 3$, we have $\pi_5(p = 3) = \lfloor 3 \times 8/5 \rfloor = 4$, and thus $Y^3 = \bar{R}^4(Z^0) = (x_3^0, \varnothing, x_4^0, \varnothing, x_0^0, x_1^0, \varnothing, x_2^0)$. We can notice that the initial position of the first coordinate x_0^0 of X^0 in Z^0 is 0. After a rotation of index $\pi_m(p)$, x_0^0 gets in position $\pi_m(p)$ in Y^p. This position is identical to the position of x_0^0 in Z^p. Since the values of X^0 are spread regularly in Z^0 thanks to (6), this property is not affected by the rotation. In other words, after the rotation $\bar{R}^{\pi_m(p)}(Z^0)$, x_0^0 is in its final position in Z^p and

all the other values are also close to their final position. Let us examine what is happening for the rest of the values of Y^p.

3) Final Mapping: For a given index $i = 0, 1, \ldots, m-1$, the position of x_i^0 in Y^p is given by the summation (modulo n) of its initial position $\pi_m(i)$ in Z^0 and the rotation factor $\pi_m(p)$, i.e., $\pi_m(p) + \pi_m(i)$. The final position of x_i^0 in Z^p should be $\pi_m(i+p)$. Let us define $\delta_{m,p}(i)$ as the difference between the theoretical position of x_i in Z^p and its effective position in Y^p, i.e.,

$$\delta_{m,p}(i) = \pi_m(i+p) - \pi_m(p) - \pi_m(i). \tag{10}$$

According to (7), it is possible to rewrite (10) as

$$\delta_{m,p}(i) = \lfloor (i+p)n/m) \rfloor - \lfloor in/m \rfloor - \lfloor pn/m \rfloor. \tag{11}$$

Theorem 1: For all $(a, b) \in \mathbb{R}^2$, $\lfloor a+b \rfloor - \lfloor a \rfloor - \lfloor b \rfloor \in \{0, 1\}$.

Proof: The real a can be decomposed as a sum of its integer part $n_a = \lfloor a \rfloor$ and a fractional part $r_a = a - n_a \in [0, 1[$, i.e, $a = n_a + r_a$. Similarly, b can be written as $b = n_b + r_b$ with $r_b \in [0, 1[$. Thus $\lfloor a+b \rfloor = n_a + n_b + \lfloor r_a + r_b \rfloor$. Since $r_a + r_b \in [0, 2[$, $\lfloor r_a + r_b \rfloor$ can only take the values 0 or 1 □

Corollary: $\delta_{m,p}(i) \in \{0, 1\}$.

Proof Use theorem 1 with $a = in/m$ and $b = pn/m$.

From the corollary, we deduce that from Y^p, we still have to perform either one shift right when $\delta_{m,p}(i) = 1$ or none when $\delta_{m,p}(i) = 0$ for each index i to obtain Z^p. Thanks to this additional permutation, for all p, m, n, $0 \leq p < m \leq n$, we have

$$Z^p = \delta_{m,p}(\bar{R}^{\pi_m(p)}(Z^0)). \tag{12}$$

Fig. 3.a shows graphically the equivalence of the two paths to obtain Z^p while Fig. 3.b gives an example for $n = 8$, $m = 5$ and $p = 3$.

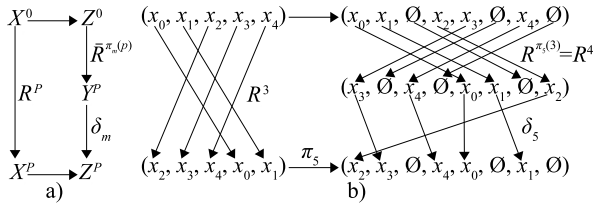

Fig. 3. Illustration of the proposed method, a) general case, b) example for $n = 8$, $m = 5$ and a rotation index of $p = 3$.

Let us examine how to implement in practice the final mapping $\delta_{m,p}(i)$. The Euclidean division of ni (respectively, np) by m gives the quotient $q_{m,i}$ and the remainder $0 \leq r_{m,i} < m$, i.e., $ni = q_{m,i}m + r_{m,i}$ (respectively, $np = q_{m,p}m + r_{m,p}$). Thus, (11) yields:

$$\begin{aligned} \delta_{m,p}(i) =& \lfloor q_{m,i} + q_{m,p} + (r_{m,i} + r_{m,p})/m \rfloor \\ & - \lfloor q_{m,i} + r_{m,i}/m \rfloor - \lfloor q_{m,p} + r_{m,p}/m \rfloor \\ =& \lfloor (r_{m,i} + r_{m,p})/m \rfloor. \end{aligned} \tag{13}$$

Note that $\delta_{m,p}$ is a vector of size m and that $\delta_{m,p}(i)$ indicates the local control signal associated to the i^{th} coordinate x_i^0 of X^0. Vector $\delta_{m,p}$ should follow the same transformation than X^0 in (8) in order to send the local control signal associated to the i^{th} coordinate of x_i^0 of X^0 to the final position of x_i^0 in Z^p, i.e.,

$$\epsilon_{m,p} = \pi_m(R^p(\delta_{m,p})). \tag{14}$$

Thus, the local signal control $\epsilon_{m,p}(j)$ allows to select either $z_j^p = y_j^p$ if $\epsilon_{m,p}(j) = 0$ or $z_j^p = y_{j-1}^p$ if $\epsilon_{m,p}(j) = 1$, ($j = 0, 1, 2, \ldots, n-1$). Back to the previous example, given $m = 5$ and $p = 3$, $r_{5,3} = 3 \times 8 \mod 5 = 4$ and $r_{5,i} = 8i \mod 5 = \{0, 3, 1, 4, 2\}_{i=0,1,\ldots,4}$, thus $\delta_{5,3} = (\lfloor 4/5 \rfloor, \lfloor 7/5 \rfloor, \lfloor 5/5 \rfloor, \lfloor 8/5 \rfloor, \lfloor 6/5 \rfloor) = (0, 1, 1, 1, 1)$ and thus $\epsilon_{5,3} = \pi_5(R^3(\delta_{5,3})) = (1, 1, \varnothing, 1, 0, \varnothing, 1, \varnothing)$, where $'\varnothing'$ stands for idle position, as shown in Fig. 3.b. In the sequel, all the idle positions are set with a zero value, i.e., $\epsilon_{5,3} = (1, 1, 0, 1, 0, 0, 1, 0)$.

B. Parallelism

When $m \leq n/2$, if only one vector is processed, most of the PE's stay in the idle mode. EBS offers the possibility to take profit of the idling PE to either process several independent frames in parallel, and thus increase the processing throughput, or to process the same frame with several decoding hypothesis to increase the decoding performance using techniques likes [8] or [9] to name only a few. In both cases a 'Single Instruction, Multiple Data (SIMD)' type of processing should be used, i.e., vectors of same size should be concurrently rotated by the same index. EBS can process in parallel T vectors with the same rotation index p, provided that $Tm \leq n$. Let $\{X_k^0\}_{k=0,1,\ldots,T-1}$ be the $T > 1$ vectors to be processed in parallel, with $x_{k,i}^0$ representing the i^{th} coordinate of the k^{th} vector. From those T vectors, we generate a single vector X'^0 of size Tm as the concatenation of the elements of vectors $\{X_k^0\}_{k=0,1,\ldots,T-1}$, i.e, $j = 0, 1, 2, \ldots, mT-1$

$$x_j'^0 = x_{r_j, q_j}^0, \tag{15}$$

where q_j and r_j are given by the Euclidean division of j by T, i.e, $j = q_j T + r_j$. For example, if $T = 2$ and $m = 3$, then the $T = 2$ vectors $X_0^0 = (x_{0,0}^0, x_{0,1}^0, x_{0,2}^0)$ and $X_1^0 = (x_{1,0}^0, x_{1,1}^0, x_{1,2}^0)$ are mapped to $X'^0 = (x_{0,0}^0, x_{1,0}^0, x_{0,1}^0, x_{1,1}^0, x_{0,2}^0, x_{1,2}^0)$. Thus by construction, any rotation of index pT on X' generates a rotation of index p on each component vectors X_k^0 of X'^0. It is thus possible to perform those operations on $Z'^0 = \pi_m(X'^0)$ with the method described in section II-A.

III. Hardware architecture

In this section, we first review the general architecture, then we focus on the particular case of EBS for the 5G standard requirement.

A. Global architecture

EBS can be split up into two parts: first, a BS of size n that can perform any rotation on a vector Z of size n. Let $0 \leq \bar{p} < n$ be the index of rotation on Z, $L = \lceil \log_2(n) \rceil$ the minimum number of bits to encode in binary the value $\bar{p}$ and

$(\bar{p}(L-1), \bar{p}(L-2), \ldots, \bar{p}(0))_2$ the binary representation of $\bar{p}$. The BS is composed of L layers, each one is composed of n multiplexers (MUX's) that allows either to perform the identity function if $\bar{p}(l) = 0$, with l the index of the layer, $0 \leq l < L$, or a circular rotation of index 2^l if $\bar{p}(l) = 1$.

Second, the final extra stage of Multiplexers performs the δ permutation. Contrary to the BS, in the final stage, the n multiplexers are controlled independently by the signal $\epsilon_{m,p}(j)$, $j = 0, 1, \ldots, n-1$. The overall complexity C_{EBS} in number of MUX's is thus given as

$$C_{\text{EBS}} = (\lceil \log_2(n) \rceil + 1)n, \tag{16}$$

and the number of layers of MUX's D_{EBS} is equal to D_{QSN} given in (4). Fig. 4 shows the datapath of EBS for $n = 8$. The determination of the effective rotation index $\bar{p} = \pi_m(p)$ and the generation of control vector $\epsilon_{m,p}$ as a function of m and p can be obtained directly by using a Read Only Memory (ROM) that stores all configurations. For a given value of m, p can take the values $0, 1, \ldots, m-1$, i.e., m possible values. Thus, it is possible to generate a unique address $A(m,p)$ for a given couple (m,p) as

$$A(m,p) = (m-1)m/2 + p, \tag{17}$$

to access a word in a ROM of size $S_R = n(n+1)/2$ (addresses vary from 0 up to $A(n, n-1)$). The word at address $A(m,p)$ is of size $n+L$ bits, containing the n values of vector $\epsilon_{m,p}$ and the L bits of the binary representation of $\pi_m(p)$, respectively.

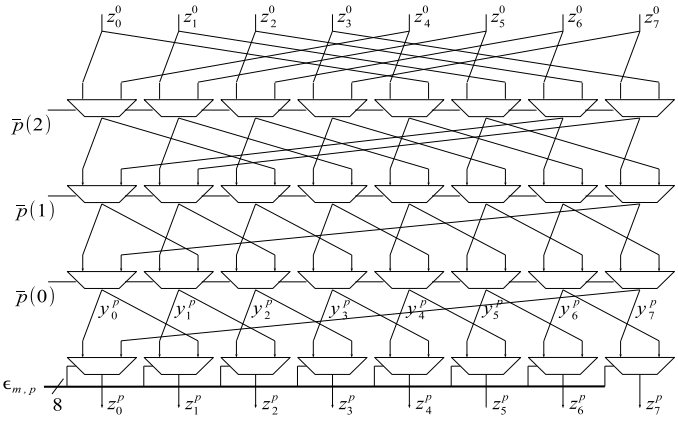

Fig. 4. Detailed EBS datapath for $n = 8$.

B. Complexity analysis

Table I compares the number of MUX's and the number of layers between BN, QSN and EBS. The table also gives the relative percentage saving S in number of MUX's between QSN and EBS with S defined as

$$S = \frac{C_{\text{QSN}} - C_{\text{EBS}}}{C_{\text{QSN}}} \times 100\%. \tag{18}$$

For the size $n = 384$, this reduction reaches $S = 42.3\%$ which is very significant. Table I also compares the synthesis results of QSN [7] and EBS using a Virtex 4 LX160-10 FPGA.

TABLE I
COMPLEXITY COMPARISON (IN NUMBER OF MUX'S) AND DELAY (IN NUMBER OF LAYERS OF MUX'S) BETWEEN THE BENES, QSN AND THE PROPOSED EBS NETWORKS.

n	48	64	128	256	384	512
D_{BN}	11	11	13	15	17	17
D_{QSN}	7	7	8	9	10	10
D_{EBS}	7	7	8	9	10	10
$\tilde{D}_{\text{EBS}}$	7	7	8	9	10	10
C_{BN}	528	704	1664	3840	6528	8704
C_{QSN}	497	705	1665	4097	6657	9217
C_{EBS}	336	448	1024	2304	3840	5120
S	32.4%	36.4%	38.5%	43.8%	42.3%	44.5%
$\tilde{C}_{\text{QSN}}$	4290	5996	13589	-	-	-
$\tilde{C}_{\text{EBS}}$	2712	3600	8192	18634	39752	53080
$\tilde{S}$	36.8%	40%	39.7%	-	-	-

The complexity $\tilde{C}_{\text{QSN}}$ and $\tilde{C}_{\text{EBS}}$ are given in number of Look Up Table (LUT). The observed LUT reduction $\tilde{S}$ between $\tilde{C}_{\text{QSN}}$ and $\tilde{C}_{\text{ESB}}$ is consistent with the theoretical reduction S. Although the number of layers is equal for both architectures, the reduction in MUX's translates also in a reduction in the number of interconnection wires, which in turn, reduces the complexity of the place and route operation in an Application Specific Integrated Circuit (ASIC) or in a Field Programmable Gate Array (FPGA). This reduction of complexity enables the reduction of both area and power consumption, along with a higher clock frequency. However, EBS has also some drawbacks compared to QSN. First it requires some memories to store the control configuration which is not the case for QSN where control signals are generated thanks to few simple logical gates. Second, the initialization of the RAM with the incoming frames and the output of the decoded frames are more complex than [7] due to the mapping π_m. Nevertheless, those operations are performed once during the initialization phase and the output phase so their latency is out of the global latency of the iterative decoding process.

The next section describes the particular case of the permutation network associated to a 5G LDPC decoder.

C. The EBS for the 5G LDPC decoder

In this section, we describe the EBS applied to the 5G LDPC decoder. Although described for this special case, some optimisation methods can also be used in the general case. First, when $p = 0$, there is no rotation, and thus, no need to apply the permutation $\delta_{m,p}$ (note that (11) is always 0 when $p = 0$). In other words, there is no need to store the vector $\epsilon_{m,0}$, nor the value of $\pi_m(p = 0)$ since it is always equal to 0. Moreover, for the 5G LDPC code, $n = 384$ is not a prime number. Thus, there exist values of m that divide n. In that case, $n/m = q$ is an integer and thus, (11) yields to $\delta_{m,p}(i) = 0$, for all values of p. Moreover, in that case, according to (7), $\pi_m(p) = pq$, thus a simple multiplier is enough to determine the value of $\pi_m(p)$. For the 5G, only a subset $\mathcal{S}$ of size 51 values of m among 383 are used. It is thus possible to identify the value of m by its position in the set $\mathcal{S} = \{m_0, m_1, ..., m_{50}\}$, or equivalently, $m_a = \mathcal{S}(a)$, with a an index varying from 0 to 50 (m_a as the form $m_a = M_a 2^{e_a}$,

with $M_a \in \{2,3,5,7,9,11,13,15\}$, $e_a \in \{0,1,\ldots,7\}$ and $m_a \leq 384$). Based on these remarks, the size of the ROM can be reduced by computing the address thanks to a table B generated recursively as $B(0) = 0$, $B(a) = B(a-1)$ if m_a divides n, $B(a) = B(a-1) + m_a - 1$ otherwise. Then from B, the address of the ROM is computed as $A(a,0) = 0$ for all $a = 0,1,\ldots,50$, $A(a,p) = 0$ for all $p = 0,1,\ldots,m_a - 1$ if m_a divides n, and otherwise, $A(a,p) = B(a) + p$. A second ROM C of size 51 words can store the value of $q_a = n/m_a$ when q_a is integer, or 0 otherwise. Thus, the test $q_a = 0$ and $p > 0$ indicates that address $A(a,p) = B(a) + p$ should be used to access ROM E that contains the values $\epsilon_{m_a,p}$, and the ROM P that contains the values $\pi_m(p)$. The size of ROM E and P, for the 5G code is 2766 words (note that, in practice, ROM E and P can be merged in a single ROM). When the conditions $q_a \neq 0$ and $p > 0$ is not fulfilled, then the 0 address is used in ROM E and P. It gives respectively the null vector for $\epsilon_{m_a,p}$ and a dummy zero value for $\pi_{m_a}(p)$ added to pq_a to generate the $\pi_{m_a}(p)$. Note that when ROM P gives a non-null value for $\pi_{m_a}(p)$, then pq_a is necessarily equal to 0, thus the final addition gives $\pi_{m_a}(p)$ in all cases. Fig. 5 illustrates the particular control structure of EBS for the 5G LDPC decoders. Finally, Fig. 6 shows the contents of ROM E. From the arabesques of the ROM contents, we can deduce that there is still redundancy that can be suppressed in this memory.

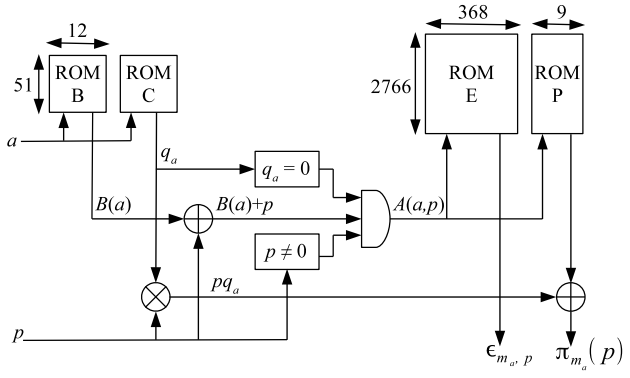

Fig. 5. Control of the EBS for the 5G LDPC code.

Finally, after determining the contents of ROM E, we realized that $\epsilon_{m,p}(j)$ is always equal to zero for 16 values of j defined as $j = 24 \times c$, $c = 1,2,\ldots,16$. This observation allows the reduction of the word size of ROM E from 384 to 368 bits, and also allows us to eliminate 16 multiplexers in the final layer of EBS.

IV. Conclusion

In this paper we show an EBS of size n can perform any circular-shift rotation of vector of size $m \leq n$ albeit to an appropriate initial positioning of the data in the processing element. We also extend the method to process an identical rotation on T vectors of size m, $Tm \leq n$, which allows either to increase the processing throughput and/or to enhance the decoding performance of the LDPC decoder. Compared to the QSN, the EBS requires between 38 % and 50 % less MUX's

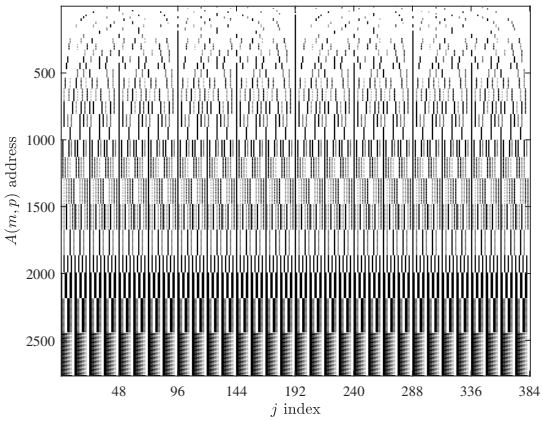

Fig. 6. Graphical representation of the ϵ ROM E contents. A 'one' is represented by a black dot, a 'zero' by a white dot.

for $n \geq 48$. This reduction of MUX's translates in a reduction of wires and thus, overall, in a significant area and critical path reductions. EBS has some drawbacks: it requires a more complex generation of control signals than for the QSN (we propose using a ROM to store the control signals) and requires more complex input/output operation, since each data should be stored in the appropriate memory bank. We showed that the proposed architecture is suitable for the 5G LDPC code where all the lifting sizes are covered.

Acknowledgment

This work has been funded by the Brittany region and the EU through the FEDER program in the frame of the FLEXDEC-5G project. The authors would like to thank Jérémie Nadal, Alireza Tasdighi, Ali Al-Ghouwayel, Cédric Marchand, Laura Conde-Canencia and Franklin Cochachin for their corrections and suggestions to improve the paper.

References

[1] D. E. Hocevar, "A Reduced Complexity Decoder Architecture Via Layered Decoding of LDPC Codes," in *IEEE Workshop on Signal Processing Systems, 2004. SIPS 2004.*, Oct 2004, pp. 107–112.

[2] 3GPP, Available Online, https://www.3gpp.org/.

[3] V. Benes, "Optimal Rearrangeable Multistage Connecting Networks," *Bell Syst. Tech. J.*, vol. 43, no. 7, pp. 1641–1656, 1964.

[4] D. Oh and K. K. Parhi, "Low-Complexity Switch Network for Reconfigurable LDPC Decoders," *IEEE Transactions On Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 1, pp. 85–94, Jan 2010.

[5] J. Lin, Z. Wang, L. Li, J. Sha, and M. Gao, "Efficient Shuffle Network Architecture and Application For WiMAX LDPC Decoders," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 56, no. 3, pp. 215–219, March 2009.

[6] M. Rovini, G. Gentile, and L. Fanucci, "Multi-Size Circular Shifting Networks for Decoders of Structured LDPC Codes," *Electronics Letters*, vol. 43, no. 17, pp. 938–940, August 2007.

[7] X. Chen, S. Lin, and V. Akella, "QSN−A Simple Circular-Shift Network for Reconfigurable Quasi-Cyclic LDPC Decoders," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 57, no. 10, pp. 782–786, Oct 2010.

[8] T. Tonnellier, C. Leroux, B. Le Gal, B. Gadat, C. Jego, and N. Van Wambeke, "Lowering the Error Floor of Turbo Codes With CRC Verification," *IEEE Wireless Communications Letters*, vol. 5, no. 4, pp. 404–407, Aug 2016.

[9] F. Leduc-Primeau, S. Hemati, S. Mannor, and W. J. Gross, "Lowering Error Floors Using Dithered Belief Propagation," in *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, Dec 2010, pp. 1–6.