# Back-to-Back Butterfly Network, an Adaptive Permutation Network for New Communication Standards

Hassan Harb, Cyrille Chavet

# Back-to-Back Butterfly Network, an Adaptive Permutation Network for New Communication Standards

Hassan Harb, Post-Doctoral, and Cyrille Chavet, IEEE senior member
Universite Bretagne Sud, Lab-STICC (CNRS UMR 6285)

***Abstract*—In this paper, we introduce an adaptive Back-to-Back Butterfly Network ($B^2BN$) dedicated to next communication standards. It can perform any kind of permutation, and its architecture is based on a concatenation of basic networks. However for a set of permutations, the selection of non-conflicting paths is still a complex task. Hence, in our paper we rely on the properties of the proposed architecture, to introduce a formal model that efficiently solve such conflicts. From the collection of all possible paths in the targeted $B^2BN$, the proposed method select the conflict-free paths to transfer data from the input to their permuted output (w.r.t., a ad hoc constraints model). Once the appropriate paths are selected, the control signals for $B^2BN$ are generated. This model has been experimented with 5G communication, showing how to process several frames in parallel with different permutation constraints.**

***Index Terms*—Butterfly Network, Back-to-Back Butterfly Network, Permutation, QC-LDPC, 5G.**

## I. Introduction

The impressive increase in data traffic in wireless communication domain overloads network capacity. Researchers are still exploring new techniques to efficiently target high throughput applications. Designed to push a step further these technical limits of telecommunication, the incoming communication standards (5G, cognitive radio, SDR...) will be the foundations of a deep revolution in our widely connected world. For example, the next 3GPP standard (the so-called 5G) [1] gathers several different radio signals (LTE-A, Wi-Fi/WISE...) in a combined heterogeneous and flexible network. In this standard, a new family of Error Correction Code (ECC) have been adopted: Quasi-Cyclic Low Density Parity Check (QC-LDPC) decoder [2].

As any ECC decoders, they are based on parallel architectures in order to achieve high throughput requirements. In such parallel designs, several Processing Elements PEs are concurrently used to decode the received information [3]. In this context, several memory banks RAMs are connected with these PEs through a dedicated interconnection network. This network transfers data between PEs and RAMs according to predefined access orders, i.e. the interleaving rules. Designing efficient parallel hardware architecture (i.e., with no access conflicts in memory nor in the interconnection network) is a very complex and time consuming task. Several approaches have been proposed in state of the art in order to solve such "collision problem" in ECC architectures ([4][5][6][7]...).

In the case of QC-LDPC codes, they are based on a prototype matrix where each $non\ null$ elements are replaced by a $N \times N$ circularly shifted identity matrix. This structure allows to implement a decoder with $N$ parallel PEs without any memory access conflict. Hence, in this case a simple barrel shifter of size $N$ can be used to reorder the data between PEs and RAMs. The particularity of the 5G standard is that the expansion factor sizes of the QC-LDPC code can take $51$ values ranging between $m = 2$ and $m = 384$ (depending on the code rate and the size of the code). Translated in hardware, this means that the architecture should be able to perform rotations on vectors of variable size. Moreover, in the incoming context of heterogeneous and flexible network, this decoder must be able to process several types of code (LTE-A, WiFi/WISE, DVB-T, DVB-S...).

Our objective in this paper is to propose an interconnection network architecture that could handle several different frames in parallel, potentially from different ECC, and the permutations of these frames should be different. All these constraints must be achieved in the minimum envelop in term of architectural cost (i.e., area and power consumption).

A barrel shifter, as defined previously cannot be used in this context. In [8] a flexible barrel shifter architecture for Quasi-Cyclic LDPC is proposed ($QSN$). This approach proposes an elegant solution if the frame size $m$ is smaller than $N$. The idea is to restrict the interconnection to the first $m$ PEs and RAMs. To be able to perform the required shift rotations, it relies in particular on a smart combination of two partial size-$N$ barrel shifters. However, even if the cost is relatively small, this approach cannot process several different frames in parallel. In [9] the authors propose an architecture that is able to deal with different standards, different frame lengths. This architecture relies on a Butterfly Network (BN) [10], that offers much more permutation flexibility than a barrel shifter. However, the main limitation of this solution is that only one single frame can be processed at a time. The most recent algorithm in this context is the Extended Barrel Shifter (EBS) [11]. The key idea is that the elements of the frame are split up appropriately at the inputs of the barrel shifter such that after performing a circular-shift rotation, each element will be at its desired position at the output or only one shift will still be required to be performed on it. For that, an extra stage is added at the output of the barrel shifter to perform the one extra shift when it is necessary. The area consumption of [11] is interestingly low. Nevertheless, [11] can process more than one frame having same length (less than or equal $N/2$) and they must have same desired circular-shift rotation value. Again, [11] has restriction in terms of parallelism. The Back-to-Back BN ($B^2BN$) [10] (or similarly Benes network [12]) could be explored. These networks are able to perform any permutation of $N$ inputs (i.e., $N!$ permutations). For that, these architectures are relatively costly. Furthermore, generating their control signals is not an easy task. However, ($B^2BN$) and Benes networks can permute more than one frame in parallel having different length and different circular-shift rotation. In [13] the authors propose to prune the Benes network in order to tailor it exactly to the requirement of the applications. The resulting architecture can be smaller than the initial Benes network, but it requires that a low number of expansion factor, which is not the case for 5G as seen before.

In this work, we adopt the $B^2BN$ network where we propose a new method to generate the control signals based on constraint tools. Note that the philosophy of this work can be performed in case of Benes network. Thus, this paper is organized as follows. Section II introduces a BN and presents the basic notations and concepts for the rest of the paper. In section III, we describe the proposed formal model and its associated modular architecture. In the final section IV, we present the application of our approach on a case study from 5G standard.

**Algorithm 1:** The BN algorithm

**Part 1**

At stage of MUXs $l = 0$:
**for** $j \leftarrow N-1$ **to** $N/2$
  $\mathrm{M}_{0j}(x_j, x_{j-N/2})$
**end for**
**for** $j \leftarrow N/2-1$ **to** $0$
  $\mathrm{M}_{0j}(x_j, x_{j+N/2})$
**end for**

**Part 2**

At the rest of $n-1$ stages of MUXs:
$N' = N$
**for** $l \leftarrow 1$ **to** $n-1$
  $N' = \frac{N'}{2^l}$
  **for** $k \leftarrow 0$ **to** $2^l - 1$
    **for** $j \leftarrow N-1-k.N'$ **to** $N-(k+\frac{1}{2^l}).N'$
      $\mathrm{M}_{lj}(y_j^{l-1}, y_{j-N'/2}^{l-1})$
    **end for**
    **for** $j \leftarrow N-1-(k+\frac{1}{2}).N'$ **to** $N-(k+\frac{1}{2^l}+\frac{1}{2}).N'$
      $\mathrm{M}_{lj}(y_j^{l-1}, y_{j+N'/2}^{l-1})$
    **end for**
  **end for**
**end for**

## II. The Butterfly Network

Let $N = 2^n$ be the size of the vector to be permuted, $n \in \mathbb{N}^+$. Hereafter, a BN associated to a vector of size $N$ is called $N\times N$-BN. In general, for a given $N = 2^n$, there are $n$ stages, each consisting in $N$ MUXs. The MUX is represented by a circle such as $\mathrm{M}_{00}$ shown in Fig.1.a). $\mathrm{M}_{00}$ is controlled by a signal $s$, such that if $s = 0$ then the output $y_0^0 = x_0$, else the ouput $y_0^0 = x_1$. The generation of the control signals is a key contribution of our work. However, the control signals are removed from the figures in order to read them easily.

The inputs of the first stage is the set $X = \{x_{N-1}, \ldots, x_0\}$, the $n-2$ intermediate stages are represented by the set $Y^i = \{y_{N-1}^i, \ldots, y_0^i\}$ (i.e., the set of results of stage $i$, where $i = 0, \ldots, n-2$), and the final stage is the resulting permutation set $Z = \{z_{N-1}, \ldots, z_0\}$ (i.e., the results of the $n^{th}$ stage). Algorithm 1 shows the connections at every stage, where $\mathrm{M}_{lj}(a, b)$ indicates that the inputs $a$ and $b$ are connected to MUX $\mathrm{M}_{lj}$ (i.e., MUX number $j$ at $(l+1)^{th}$ stage), $l = 0, \ldots, n-1$ and $j = N-1, \ldots, 0$.

$N\times N$-BN can perform any circular-shift rotation value $p_{c_s}$, where $0 \leq p_{c_s} < N$. For a given $p_{c_s}$, every input $x_i$ should be transferred to $z_{(p-i) \bmod (N)}$, $i = N-1, \ldots, 0$.

Fig.1.a) shows a $4\times4$-BN architecture. $X = \{x_3, \ldots, x_0\}$, $Y^0 = \{y_3^0, \ldots, y_0^0\}$ and $Z = \{z_3, \ldots, z_0\}$ are respectively the inputs set, the intermediate permutations set and the final permutation results set.

A BN is unable to perform all possible permutations. To tackle all these challenges, in this paper we propose a modular construction of a $B^2BN$. To construct a $B^2BN$, the concept of $Inverse\ BN$ is required [14]. Fig.1.b) shows the inverse of $4\times4$-BN.

## III. Back-to-Back Butterfly Network

A $B^2BN$ is a combination of a BN and its corresponding inverse. A modification should be done on the inverse BN. The first stage of inverse BN is removed since it is similar to last stage of BN (see Fig.1.a) and Fig.1.b). A $B^2BN$ has the same expressiveness that a Benes network, which means that all possible permutations can be generated with this model [15]. Once again, for a given $N$, we refer to $B^2BN$ associated to a vector of size $N$ as $N\times N$-$B^2BN$. This $N\times N$-$B^2BN$ requires $2.n-1$ stages each of $N$ parallel MUXs. The intermediate results are called $Y^l = \{y_{N-1}^l, \ldots, y_0^l\}$, $l = 0, \ldots, 2.n-2$. Fig.1.c) shows the $4\times4$-$B^2BN$ architecture. Two complementary parts can be observed, a $4\times4$-BN connected to the "modified" inverse $4\times4$-BN. Fig.1.d) shows an example of a permutation: $z_3 = x_2$, $z_2 = x_0$, $z_1 = x_3$ and $z_0 = x_1$. The inputs are being transferred through the stages to perform the desired permutation as shown in Fig.1.d). The "line" that transfers an input $x_i$ to an output $z_j$ is called a $path\ T_{x_i \to z_j}$, with $i, j = N-1, \ldots, 0$. It is not an easy task to find out the paths of the inputs through the stages given a desired permutation. In other words, it is not easy to generate the control signals of the MUXs to perform all the desired permutations. $N!$ possible permutations are possible in case of $N\times N$-$B^2BN$. Thus, the higher the $N$ value the higher the number of stages, and hence the more complex is the problem to find out the appropriate paths for every desired permutation with no conflict along the path. In this paper, we also propose a formal approach to solve this problem.

### A. Proposed Method to Generate the Control Signals

For a given permutation, the proposed method relies on collecting all the possible paths from $x_i$ to $z_j$, where $i, j = N-1, \ldots, 0$. We call this set $T_{x_i \to z_j}$, where $\to$ refers to a transition. In total, there will be $N$ sets (one set of path for every inputs) each of $2^{n-1}$ possible paths. In more details, to transmit $x_i$ to $z_j$ we have: $x_i \to y_{k_0}^0 \to \cdots \to y_{k_{2.n-2}}^{2.n-2} \to z_j$, where $i, k_0, \ldots, k_{2.n-2}$ and $j \in [N-1, \ldots, 0]$. Thus, the path is presented as $(i, k_0, \ldots, k_{2.n-2}, j)$.

For example, let $x_2$ should be transferred to $z_3$ as shown in Fig.1.d). We have $z_3 = y_1^1 = y_0^0 = x_2$ and hence the path is $(2, 0, 1, 3)$. Considering the permutation shown in Fig.1.d), the original set of all possible paths is reduced to the subsets in which the input $x_i$ corresponds to the output $z_j$, $i, j = 3, \ldots, 0$, with respect to the required permutation:

$T_{x_3 \to z_1} = \{(3,3,3,1), (3,1,1,1)\}$
$T_{x_2 \to z_3} = \{(2,2,3,3), (2,0,1,3)\}$
$T_{x_1 \to z_0} = \{(1,1,0,0), (1,3,2,0)\}$
$T_{x_0 \to z_2} = \{(0,2,2,2), (0,0,0,2)\}$

The solution shown in Fig.1.d) is found based on $T_{x_3 \to z_1}$, $T_{x_2 \to z_3}$, $T_{x_1 \to z_0}$ and $T_{x_0 \to z_2}$. Let $A$ be a matrix of size $N \times 2.n$ where:

$$A = \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 3 & 3 & 3 & 1 \\ 2 & 0 & 1 & 3 \\ 1 & 1 & 0 & 0 \\ 0 & 2 & 2 & 2 \end{pmatrix}$$

The elements of the first row of $A$ belong to $T_{x_3 \to z_1}$ (i.e., $(a_{00}, a_{01}, a_{02}, a_{03})$ is equal to $(3,3,3,1)$ or $(3,1,1,1)$). The elements of the second row of $A$ belong to $T_{x_2 \to z_3}$, the elements of the third row of $A$ belong to $T_{x_1 \to z_0}$ and the elements of the fourth row of $A$ belong to $T_{x_0 \to z_2}$. The elements of every column of $A$ should be different, i.e., no conflict in terms of MUXs at every stage. Thus, one possible solution is the solution that $A$ equal to. Therefore, $A$ is the matrix representation of the solution shown in Fig.1.d).

### B. Control Signals Generation

As described previously, the generation of the control signals consists in fixing the control signal $s$ of each MUX. We recall that the output of $\mathrm{M}_{00}$ shown in Fig.1.a) is equal to $x_0$ (direct input) when $s = 0$ while it is equal to $x_2$ (indirect input) when $s = 1$ (every MUX has its own $s$). In other words, to compute the

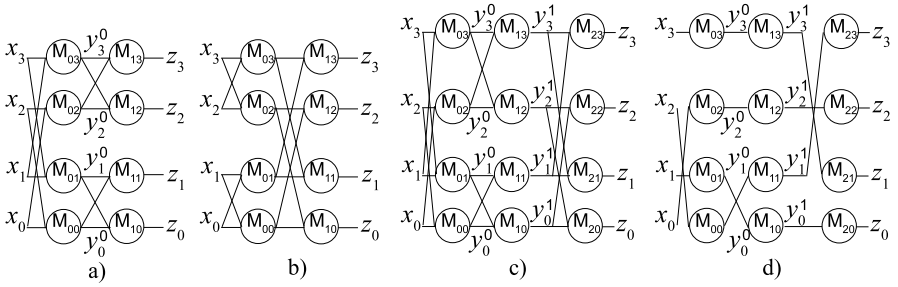


Fig. 1. a) 4×4-BN; b) Inverse 4×4-BN; c) 4×4-$B^2BN$ and d) 4×4-$B^2BN$ example.

**Algorithm 2:** Algorithm of control signals generation

```
for i ← 0 to N − 1
  for j ← 0 to 2.n − 2
    if a_{ij+1} = a_{ij}
      b_{ij} = 0
    else
      b_{ij} = 1
    end if
  end for
end for
```

$s$ control value of a given MUX, we have to check whether the output is equal to the direct input or the indirect input. This can be done by analyzing $A$ matrix. Let $B$ be the matrix that contains the control signals of every MUX in $B^2BN$. Thus, $B$ is of size equal to $N \times 2.n-1$. Algorithm 2 shows how the control signals are being generated Based on $A$. In this algorithm, the $j^{th}$ column in $B$ is associated to $j^{th}$ stage of MUXs in $B^2BN$. Using Algorithm 2, the control signals of the MUXs of the example shown in Fig.2.g) are:

$$B = \begin{pmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \\ b_{30} & b_{31} & b_{32} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

$b_{00}$ is the control signal of $M_{03}$, $b_{01}$ is the control signal of $M_{13}$, $b_{02}$ is the control signal of $M_{23}$, ... and $b_{32}$ is the control signal of $M_{20}$. Then, it is clear that as soon as we have the set of paths required by the applications, the control signals generation is straightforward. Hence, the key problem of our approach is to be able to explore the set of all possible paths.

### C. Path-Finder Generation Approach

Our solution exploits the modularity offered by the $B^2BN$ architecture. More formally, every $x_i$, $i = N-1, \ldots, 0$, can be transferred to any $z_j$, $j = N-1, \ldots, 0$. We recall that $T_{x_i \to z_j}$ is the set of possible paths that transfer $x_i$ to $z_j$. $T_{x_i \to z_j}$ is of length equal to $2^{n-1}$. Thus, there are $2^{n-1} \times N = 2^{n-1} \times 2^n = 2^{2.n-1}$ paths that connect $x_i$ to $\{z_{N-1}, \ldots, z_0\}$. Therefore, in total, there are $2^{2.n-1} \times N = 2^{2.n-1} \times 2^n = 2^{3.n-1}$ paths that connect $\{x_{N-1}, \ldots, x_0\}$ to $\{z_{N-1}, \ldots, z_0\}$ in $N \times N$-$B^2BN$. The sets that are having the $2^{3.n-1}$ paths is called $T^{N \times N}$.

To extend the case from $N \times N$-$B^2BN$ up to $N' \times N'$-$B^2BN$ ($N' = 2.N$), $N \times N$-$B^2BN$ should be duplicated and a stage of MUXs at the inputs and a stage of MUXs at the outputs should be added. The connections at the inputs of both added stages are being done based on the initialization loops described in **part 1** of algorithm 1 ($N$ is then replaced by $N'$). This modularity allows to derive all possible $N' \times N'$-$B^2BN$ paths from $T^{N \times N}$. Algorithm 3 shows how the $N' \times N'$-$B^2BN$ paths are generated from the $N \times N$-$B^2BN$ paths.

**Algorithm 3:** Generation of the $N' \times N'$-$B^2BN$ paths

```
Read all paths associated to N×N-B²BN called T^{N×N}.
N' = 2.N, i = 0 and L_{T^{N×N}} = 2^{3.n−1} is the length of T^{N×N}.

for j ← 0 to L_{T^{N×N}} − 1
  V_0 = T^{N×N}(j)
  T^{N'×N'}(i) = {V_0(0), V_0, V_0(2.n − 1)}
  T^{N'×N'}(i + 1) = {V_0(0) + N, V_0, V_0(2.n − 1)}
  T^{N'×N'}(i + 2) = {V_0(0), V_0, V_0(2.n − 1) + N}
  T^{N'×N'}(i + 3) = {V_0(0) + N, V_0, V_0(2.n − 1) + N}

  V_1 = V_0 + N
  T^{N'×N'}(i + 4) = {V_1(0), V_1, V_1(2.n − 1)}
  T^{N'×N'}(i + 5) = {V_1(0) + N, V_1, V_1(2.n − 1)}
  T^{N'×N'}(i + 6) = {V_1(0), V_1, V_1(2.n − 1) + N}
  T^{N'×N'}(i + 7) = {V_1(0) + N, V_1, V_1(2.n − 1) + N}
  i = i + 8
end for
```

For a given permutation to be performed by $N' \times N'$-$B^2BN$, the relevant set of paths $T_{x_i \to z_j}$ are selected from $T^{N' \times N'}$ and the solution is found depending on them as shown previously.

Fig.2.a shows 8×8-$B^2BN$ designed from 4×4-$B^2BN$. The bold MUXs $M_{ij}$, $i = 1, 2, 3$ and $j = 3, 2, 1, 0$, constitute 4×4-$B^2BN$ as well as $M_{ij}$, $i = 1, 2, 3$ and $j = 7, 6, 5, 4$, constitute another 4×4-$B^2BN$. $T^{8\times8}$ can be generated from $T^{4\times4}$ based on algorithm 3. The eight pointed paths shown in Fig.2.a) are derived from the path $V_0 = \{1, 3, 2, 0\}$ associated to 4×4-$B^2BN$. Of course, for real life test cases, it is not possible to solve such problem by hand. In the next section shows how to execute the proposed method using dedicated tools.

Fig.2.b) shows an example where two sets of elements are processed simultaneously. A set $X' = \{x'_7, \ldots, x'_3\}$ where $p'_{c_s} = 2$ is performed and hence $\{z_7, z_6, z_5, z_4, z_3\} = \{x'_5, x'_4, x'_3, x'_7, x'_6\}$ (straight lines) and $X'' = \{x''_2, x''_1\}$ where $p''_{c_s} = 1$ is performed and hence $\{z_2, z_1\} = \{x''_1, x''_2\}$ (pointed lines).

## IV. Application of our model on a 5G test case

During our experimentation's, we used the proposed approach to design architectures relying on the QC-LDPC code proposed in the 5G standard [1]. The resulting interconnection network should be able to permutes distinct frames in parallel, with distinct permutation constraints. As previously mentioned, the design space is too huge to be explored by hand, that is why we decided to provide our constraint model to a constraints solver tool. Many constraints solver tool exist in the literature, depending on the problem to solve. For example, if

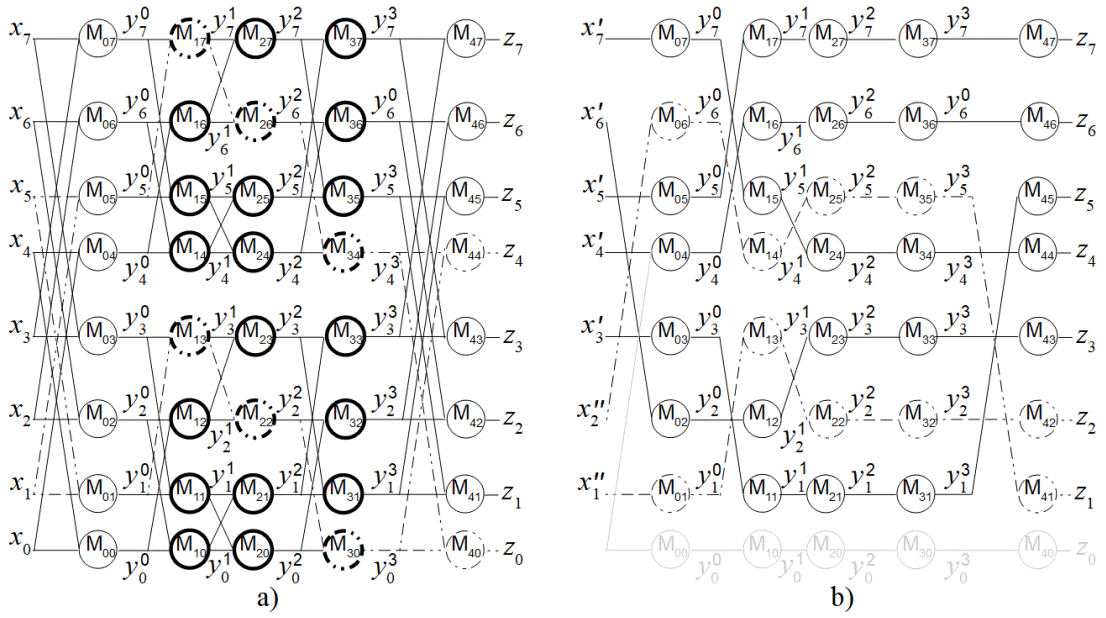


Fig. 2. a) $8 \times 8$-$B^2BN$ shuffling 8 inputs and b) $8 \times 8$-$B^2BN$ example shuffling two sets of elements with two different permutations.

the constraints are correctly defined, a solver could find solutions to $SAT$isfaction problems, linear inequality...

In our context, from a $N' \times N'$-$B^2BN$ for a given permutation $P$, we defined the matrix $A$ (see previous section) in which each of the $N$ rows is considered as a $variable$. Then, from the set of all possible path $T^{N \times N}$ (generated by Algorithm 3), we extract all the paths that match with permutation $P$ for each $variable$ in matrix $A$. In this way, we define the $N$ $domains$ for the $N$ $variables$ in matrix $A$. Then we provide the matrix $A$, the associated $domains$ and the constraint defined in section III (i.e., each element in a column of $A$ must be different) to a constraint solver (in our experiments we used Gecode [16]). Finally, the generation of the control signal is performed by applying Algorithm 3 on the resulting matrix. If the architecture has to deal with several permutations, the idea is to generate the control signals for every permutations off line, and store them in a ROM. The time consumption to find out a solution varies depending on the size $N$ and the desired permutation.

The lifting size of the 5G LDPC codes varies from 2 up to 384 [1]. Thus, since $N$ is a power of 2, a $512 \times 512$-$B^2BN$ should be designed to cover all lifting sizes. However, since $B^2BN$ can perform all kind of permutations, and thanks to its modular construction, the parallelism can be covered where more than one frame has to be permuted simultaneously. For instance, a frame of 384 elements can be processed simultaneously with a frame of $512 - 384 = 128$ elements.

Table I shows the number of MUXs of our model compared to QSN [8] and EBS [11]. Recalling that QSN cannot perform several frames in parallel and EBS is limited in this context. It can be observed that our approach presents a limited over-cost between around $\times 0.92$ to $\times 0.72$ in these 5G test cases compared to QSN. While comparing to EBS, the limited over-cost varies between $\times 0.67$ and $\times 0.44$. This additional complexity is the minimal investment to take advantage of the high flexibility offered by our architectural model.

TABLE I
NETWORK COMPLEXITY COMPARISON (NUMBER OF MUXS)

| $N$ | Based On | Nb. of MUXs |
|---|---|---|
| 15 | *Our model* | 112 |
| | [QSN] | 104 |
| | [EBS] | 75 |
| 80 | *Our model* | 1664 |
| | [QSN] | 945 |
| | [EBS] | 640 |
| 384 | *Our model* | 8704 |
| | [QSN] | 6273 |
| | [EBS] | 3840 |

## V. CONCLUSION

In this paper, we proposed a new method to resolve the problem of finding the appropriate paths for a given permutation using $B^2BN$. The solution should take into account that the output of a MUX should not present two different data. We have shown how the sets of all possible paths can be generated and how a solution can be found using these sets, with a dedicated constraints model. $B^2BN$ can be employed for different applications such as 5G LDPC codes where the parallelism is needed to increase the throughput rate. This proposal could be enhanced (e.g., by replacing the control ROM by a RAM) if the constraint solver could be replaced by a dedicated real-time algorithm.

## VI. ACKNOWLEDGEMENTS

This work has been founded the Brittany region and the EU that funded the project through the FEDER program in the frame of the FLEXDEC-5G project. The authors would like also to thank Jeremie Nadal and Cedric Marchand for their corrections and suggestions to improve the paper.

## REFERENCES

[1] http://www.3gpp.org.

[2] K. T. Sarika and P. P. Deepthi, "A channel coder design for a high speed and less complex communication system using qc - ldpc codes," in *2014 International Conference on Communication and Signal Processing*, April 2014, pp. 326–330.

[3] A. H. Sani, P. Coussy, and C. Chavet, "A first step toward on-chip memory mapping for parallel turbo and ldpc decoders: A polynomial time mapping algorithm," *IEEE Transactions on Signal Processing*, vol. 61, no. 16, pp. 4127–4140, Aug 2013.
[4] S. U. Reehman, C. Chavet, P. Coussy, and A. Sani, "In-place memory mapping approach for optimized parallel hardware interleaver architectures," in *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2015, pp. 896–899.
[5] M. J. Thul, F. Gilbert, and N. Wehn, "Optimized concurrent interleaving architecture for high-throughput turbo-decoding," in *9th International Conference on Electronics, Circuits and Systems*, vol. 3, Sep. 2002, pp. 1099–1102 vol.3.
[6] C. Chavet and P. Coussy, Eds., *Advanced Hardware Design for Error Correcting Codes*, springer-verlag ed., Sep. 2015, pp. 177–192, ISBN 978–3–319–10 568–0.
[7] A. Tarable, S. Benedetto, and G. Montorsi, "Mapping interleaving laws to parallel turbo and ldpc decoder architectures," *IEEE Transactions on Information Theory*, vol. 50, no. 9, pp. 2002–2009, Sep. 2004.
[8] S. L. X. Chen and V. Akella, "Qsn : A simple circular-shift network for reconfigurable quasi-cyclic ldpc decoders," *IEEE Trans. on Circuits and Systems II: Express Briefs*, vol. 57, no. 10, pp. 782–786, Oct 2010.
[9] A. Sani, P. Coussy, and C. Chavet, "A dynamically reconfigurable ecc decoder architecture," in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2016, pp. 1437–1440.
[10] H. Moussa, O. Muller, A. Baghdadi, and M. Jezequel, "Butterfly and benes-based on-chip communication networks for multiprocessor turbo decoding," in *2007 Design, Automation Test in Europe Conference Exhibition*, April 2007, pp. 1–6.
[11] E. Boutillon and H. Harb, "Extended barrel-shifter for versatile qc-ldpc decoders," *IEEE Wireless Communications Letters*, pp. 1–1, 2020.
[12] V. Benes, "Optimal rearrangeable multistage connecting networks," *Bell Syst. Tech. J.*, vol. 43, no. 7, p. 1641–1656, 1964.
[13] 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.
[14] http://programming.sirrida.de/bit_perm.html.
[15] http://pages.cs.wisc.edu/ tvrdik/10/html/Section10.html.
[16] https://www.gecode.org/doc-latest/MPG.pdf.