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

Hassan Harb, Cyrille Chavet

Journal of Signal Processing Systems

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

--Manuscript Draft--

Manuscript Number:	VLSI-D-20-00270R1
Full Title:	Back-to-Back Butterfly Network, an Adaptive Permutation Network for New Communication Standards
Article Type:	*ICASSP 2020
Keywords:	Butterfly Network, Back-to-Back Butterfly Network, Permutation, QC-LDPC, 5G
Order of Authors:	Hassan Harb
	Cyrille Chavet
Corresponding Author:	Cyrille Chavet, Ph.D. Lab-STICC Lorient, FRANCE
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	Lab-STICC
Corresponding Author's Secondary Institution:	
First Author:	Hassan Harb
First Author Secondary Information:	
Order of Authors Secondary Information:	
Funding Information:	
Abstract:	In this paper, we introduce a Back-to-Back Butterfly Network (B 2 BN) based on multiplexers (MUXs) in which any kind of permutation can be performed. However, for a given permutation, it is not an easy task to select the appropriate paths in B 2 BN without any conflict in terms of MUXs. In this paper, we propose a formal model to solve efficiently such conflicts. The proposed method relies on collecting the sets of potential paths that transfer an input to an output. Then, a path from each set is selected respecting a conflict free constraint. Once the appropriate paths are selected, the control signals of the MUXs are generated. This model has been experimented with 5G communication, showing how to process several frames in parallel with different permutation constraints.

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

HASSAN HARB, Universite Bretagne Sud, Lab-STICC (CNRS UMR 6285)
CYRILLE CHAVET, Universite Bretagne Sud, Lab-STICC (CNRS UMR 6285)

In this paper, we introduce a Back-to-Back Butterfly Network (B^2BN) based on multiplexers (MUXs) in which any kind of permutation can be performed. However, for a given permutation, it is not an easy task to select the appropriate paths in B^2BN without any conflict in terms of MUXs. In this paper, we propose a formal model to efficiently solve such conflicts. The proposed method relies on collecting the sets of potential paths that transfer an input to an output. Then, a path from each set is selected respecting a conflict free constraint. Once the appropriate paths are selected, the control signals of the MUXs are generated. This model has been experimented with 5G communication, showing how to process several frames in parallel with different permutation constraints.

CCS Concepts: • **Hardware → Digital signal processing**.

Additional Key Words and Phrases: 5G, Butterfly Network, Back-to-Back Butterfly Network, Permutation, QC-LDPC

ACM Reference Format:
Hassan Harb and Cyrille Chavet. 2020. Back-to-Back Butterfly Network, an Adaptive Permutation Network for New Communication Standards. 1, 1 (October 2020), 11 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

The impressive increase in data traffic in the 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) [5] 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 [22].

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 [21]. 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 ([19][24][8][23][14][26][15]…).

Authors' addresses: Hassan Harb, hassan.harb@univ-ubs.fr, Universite Bretagne Sud, Lab-STICC (CNRS UMR 6285), rue St Maude, Lorient, 56100; Cyrille Chavet, cyrille.chavet@univ-ubs.fr, Universite Bretagne Sud, Lab-STICC (CNRS UMR 6285), rue St Maude, Lorient, 56100.

Manuscript submitted to ACM

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 terms of architectural cost (i.e., area and power consumption). Thus, this paper is organized as follows. The next section presents the state of the art related to our contribution. Then, section III introduces a Butterfly Network and presents the basic notations and concepts for the rest of the paper. In section IV, we describe the proposed formal model and its associated modular architecture. In the final section, we present the application of our approach on a case study from 5G standard and we propose some comparisons with the most relevant approaches in the literature.

2 STATE OF THE ART

As described in the introduction, the 5G QC-LDPC standard [5] is dependent on the code's rate and size; with expansion factor sizes of 51 values, ranging between 2 and 384. Therefore, this validates the hardware point of view concerning the inevitable ability of the interconnection network in performing rotations on vectors of variable sizes, along with parallel processing on several heterogeneous concurrent frames for the sake of the throughput enhancement. A classical Barrel Shifter, as defined in the introduction, cannot handle all the required constraints of such design. For instance, the authors in [18] suggested a high throughput 5G LDPC encoder based on a Barrel Shifter, but without the capability to deal with different frame lengths and distinct circular shift rotation values because of the classical shifters deployed. Hence, flexibility should be added on the encoder architecture of their work. In [25] 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 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 [20] the authors propose an architecture that is able to deal with different standards, different frame lengths. This architecture relies on a Butterfly Network (BN) [16], 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. In [14] a hard-wired permutation architecture is proposed, based on the fact that parity check matrix shift values are not all consecutive integers. This solution offers good complexity reduction compared to classical barrel-shifters but it cannot handle different parallelism. Similarly, in [15] the authors proposed combinations of fixed cyclic shifters, achieving also good complexity reduction, but still with no frame parallelism. The most recent algorithm in this context is the Extended Barrel Shifter (EBS) [7]. In this paper, 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 [7] is

interestingly low. Nevertheless, [7] 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, [7] has restriction in terms of parallelism.

Considering the targeted standard and for the purpose of high throughput; the authors of [10], presented a parallel Wi-Max LDPC code architecture based on two classical Barrel Shifters of size 96 followed by a merge stage, making their proposal relatively costly compared to our approach. Meanwhile, other existing solutions are Benes [6] or Butterfly Networks [17] that are able to perform any permutation besides the circular-shift rotations, permitting both solutions to be able to process more than one frame simultaneously. However, these solutions are complex in terms of MUXs number and cost control. Furthermore, generating their control signals is not an easy task. However, Butterfly and Benes networks can permute more than one frame in parallel having different length and different circular-shift rotation. In [12] 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 a low number of expansion factor, which is not the case for 5G as seen before. In order to be able to deal with all the expansion factors of the 5G standard [26] proposes an approach to extend the parallelism of the designed network, and also to avoid some limitations of [7], by using Banyan networks. In our paper, we will refer to [26] as Extended Banyan Network (EBN) for the sake of simplicity. The idea behind EBN is interesting, but all the processed frames must have the same length. In this context, a cheaper and more flexible solution than [26] has been proposed in [11]. In this work an architecture called *Fully Parallel Circular-shift rotation Network* (FPCN) is introduced. This architecture has many advantages compared to state-of-art (lost cost, high frame parallelism…), but it has been designed in the context of high throughput 5G communications. Then, this means that it is able to process very efficiently more than one frame simultaneously, even with different sizes and/or different permutation constraints. However, the expressiveness of [11] in terms of permutation besides the circular-shift rotations, is limited.

Equally important, the writers of the study of [9] offered the combination of two partial Barrel Shifters with an addition of a merge network. This approach can conduct circular-shift rotation on a vector of varying length. But, although its cost is relatively low and the control signals can be generated in run-time, this approach cannot process many parallel frames. Among the vast research field on interconnection networks, very few works (as best as we know) target our studied drawback on how to design a low-cost parallel interconnection network architecture (in terms of area and power consumption), with high performances in terms of throughput and flexibility/adaptability.

In this work, we adopt a *Back-to-Back* dual Butterfly Network (B^2BN) for which we propose a new method to generate the control signals based on constraint tools. Note that the philosophy of this work can be also extended in case of a Benes network.

3 THE BUTTERFLY NETWORK

Let $N = 2^n$ be the size of the vector to be permuted, $n \in \mathbb{N}^+$. Hereafter, a Butterfly Network (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 M_{00} shown in Fig.1.a). M_{00} is controlled by a signal s, such that if $s = 0$ then the output $y_0^0 = x_0$, else the output $y_0^0 = x_1$. 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 $Y^i = \{y^i_{N-1}, \ldots, y^i_0\}$ (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 $M_{lj}(a, b)$ indicates that the inputs a and b are connected to MUX M_{lj} (i.e., MUX number j at $(l+1)^{th}$

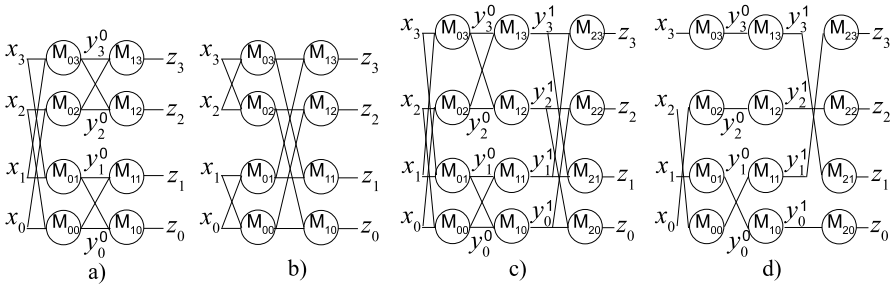

Fig. 1. a) 4×4-BN; b) Inverse 4×4-BN; c) 4×4-B^2BN and d) 4×4-B^2BN example.

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 $z_{(p-i) \mod (N)}$, $i = N-1, \ldots, 0$.

Fig.1.a) shows a 4×4-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.

Algorithm 1: The Butterfly Network algorithm

Part 1 - *At first stage of* MUXs $l = 0$:

for $j \leftarrow N-1$ **Down To** $N/2$
  $\mathrm{M}_{0j}(x_j, x_{j-N/2})$
end for
for $j \leftarrow N/2-1$ **Down To** 0
  $\mathrm{M}_{0j}(x_j, x_{j+N/2})$
end for

Part 2 - *For the rest of* $n-1$ *stages of* MUXs:

$N' = N$

for $l \leftarrow 1$ **to** $n-1$
  $N' = \dfrac{N'}{2^l}$
  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

Fig.1.b shows an example of a permutation based on the 4×4-BN. This permutation is a simple circular shift of 1 ($p_{c_s} = 1$), i.e., $\{z_3, z_2, z_1, z_0\} = \{x_0, x_3, x_2, x_1\}$. Every input is being moved through the stages till it is transferred to its appropriate position at the output. For instance, x_2 should be transferred to z_1 and hence $y_0^0 = x_2$ using M_{00}, then $z_1 = y_0^0 = x_0$ using M_{11}. The control signals of the MUXs are straightforward to be generated when BN is used to perform circular-shift rotation permutation [13].

N×N-BN can also perform other permutations rather than circular-shift rotation. In Fig.1.c we observe that 4×4-BN is also able to perform non-circular shift rotation where $\{z_3, z_2, z_1, z_0\} = \{x_1, x_2, x_3, x_0\}$. This permutation can be executed without any conflict in terms of MUXs. A BN is unable to perform all possible permutations. For example, in Fig.1.d there is a conflict since two different inputs must access the same MUX. In this example, $z_1 = x_3$ and $z_0 = x_1$ and hence x_3 and x_1 should be at the same output of M_{01}. Like in the context of memory access conflict (e.g., [23]), solving conflicting access to some shared resources is a complex task. Also, since we want to perform all possible permutations of the inputs set, we need to extend the set of permutations offered by the BN. Moreover, we want to compute several frames in parallel, each with its own length and its own permutation constraint.

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 [1]. Fig.1.b) shows the inverse of 4×4-BN. It is not easy to generate the control signals of the MUXs where BN is used for non-circular-shift rotation permutation [1, 2].

4 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 [2]. Once again, for a given N, we refer to B^2BN associated to a vector of size N as N×N-B^2BN. This N×N-B^2BN requires $2.n - 1$ stages each of N parallel MUXs. The intermediate results are called $Y^l = \{y^l_{N-1}, \ldots, y^l_0\}$, $l = 0, \ldots, 2.n - 2$. Fig.1.c) shows the 4×4-B^2BN architecture. Two complementary parts can be observed, a 4×4-BN connected to the "modified" inverse 4×4-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* $T_{x_i \rightarrow z_j}$, 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×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.

4.1 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 $T_{x_i \rightarrow z_j}$, where $\rightarrow$ 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: $x_i \rightarrow y^0_{k_0} \rightarrow \cdots \rightarrow y^{2.n-2}_{k_{2.n-2}} \rightarrow z_j$, 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).

4.2 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 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 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.

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
```

Using Algorithm 2, the control signals of the MUXs of the example shown in Fig.1.d) 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.

4.3 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.

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.

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

Read all paths associated to $N{\times}N$-B^2BN called $T^{N \times N}$.
$N' = 2.N$, $i = 0$ and $L_{T^{N \times N}} = 2^{3.n-1}$ is the length of $T^{N \times N}$.

for $j \leftarrow 0$ **to** $L_{T^{N \times N}} - 1$
  $V_0 = T^{N \times N}(j)$
  $T^{N' \times N'}(i) = \{V_0(0), V_0, V_0(2.n-1)\}$
  $T^{N' \times N'}(i+1) = \{V_0(0) + N, V_0, V_0(2.n-1)\}$
  $T^{N' \times N'}(i+2) = \{V_0(0), V_0, V_0(2.n-1) + N\}$
  $T^{N' \times N'}(i+3) = \{V_0(0) + N, V_0, V_0(2.n-1) + N\}$

  $V_1 = V_0 + N$
  $T^{N' \times N'}(i+4) = \{V_1(0), V_1, V_1(2.n-1)\}$
  $T^{N' \times N'}(i+5) = \{V_1(0) + N, V_1, V_1(2.n-1)\}$
  $T^{N' \times N'}(i+6) = \{V_1(0), V_1, V_1(2.n-1) + N\}$
  $T^{N' \times N'}(i+7) = \{V_1(0) + N, V_1, V_1(2.n-1) + N\}$
  $i = i + 8$
end for

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

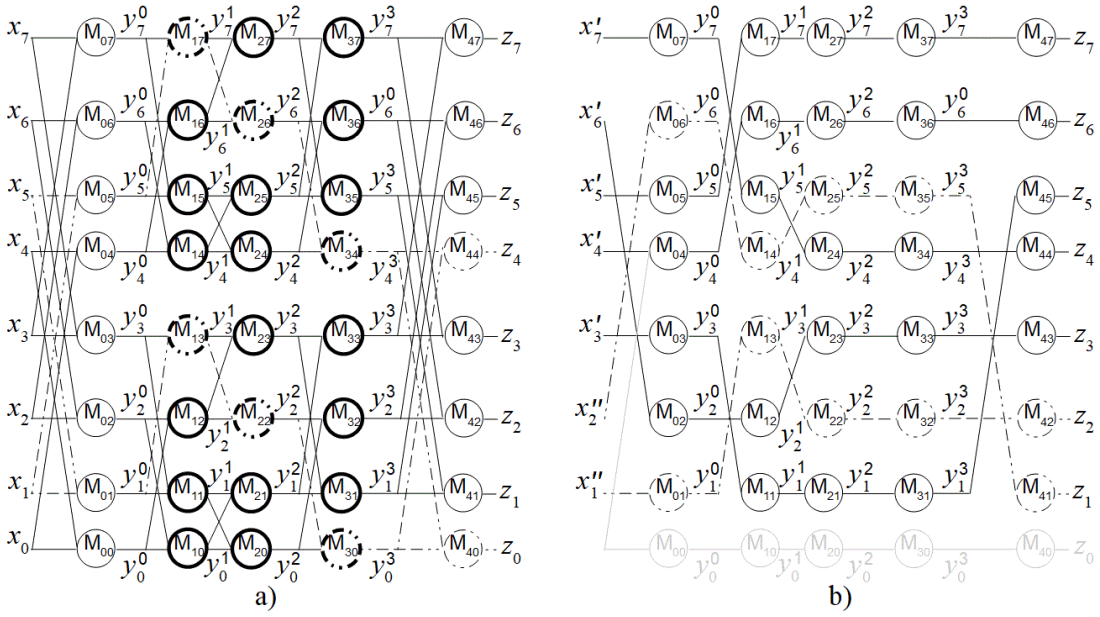

Fig. 2. a) 8×8-B^2BN shuffling 8 inputs and b) 8×8-B^2BN example shuffling two sets of elements with two different permutations.

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).

5 APPLICATION OF OUR MODEL ON A 5G TEST CASE

During experimentations, we used the proposed approach to design architectures relying on the QC-LDPC code proposed in the 5G standard [3]. 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 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 [4]). Finally, the generation of the control signal is performed by applying Algorithm 3 on the resulting

Table 1. Network complexity comparison (number of MUXs)

N	Based On	Parallelism	Nb. of MUXs	Critical path ($\times T_{MUX}$)
15	*Our model*	4	112	8
	[QSN]	1	104	5
	[EBS]	1	75	5
	[FPCN]	2	117	5
	[EBN]	2	120	5
80	*Our model*	11	1664	14
	[QSN]	1	945	8
	[EBS]	1	640	8
	[FPCN]	-	640	8
	[EBN]	-	880	8
384 (5G)	*Our model*	27	8704	18
	[QSN]	1	6273	10
	[EBS]	1	3840	10
	[FPCN]	22	4800	10
	[EBN]	22	4992	10

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 [3]. Thus, since N is a power of 2, a 512×512-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 1 shows the order of parallelism, the number of MUXs and the critical path of our model compared to QSN [25], EBS [7], FPCN [11] and EBN [26]. Recalling that QSN cannot perform several frames in parallel and that EBS, EBN and FPCN have some limitations in this context. It can be observed that our approach presents a limited over-cost between around ×0.92 to ×0.72 in these 5G test cases compared to QSN. While comparing to EBS, the limited over-cost varies between ×0.67 and ×0.44. Comparing to FPCN, the over-cost of this work varies between ×1.04 and ×0.55. Finally, the comparison with EBN shows that the over-cost varies between ×1.07 and ×0.57. These additional complexities are the minimal investment to take advantage of the high flexibility offered by our architectural model.

In terms of parallelism, the proposed model provides wider range in all cases of N. However, in terms of critical path, the proposed model requires more stages of MUXs and hence more critical path compared to the existing algorithms. The number of stages is equal to the number shown in Table I while the critical path is the number of stages multiplied by T_{MUX} (the time required to perform a multiplexer). For example, in case of $N = 15$, the proposed model consists of 8 stages and hence the critical path is equal to $8 \times T_{MUX}$.

6 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.
Of course, the reader should have noticed that the efficiency of our approach depends on the choice of the initial path provided to the constraints solver tool. This is a well-known NP-complete problem we already targeted in previous work, but in the particular context of B^2BN this aspect is going to be explored in incoming research.

7 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 (Post-Doc. IMT Atlantique) and Cedric Marchand (PhD - Engineer / Lab-STICC) for their corrections and suggestions to improve the paper.

REFERENCES

[1] [n. d.]. http://programming.sirrida.de/bit_perm.html.
[2] [n. d.]. http://pages.cs.wisc.edu/~tvrdik/10/html/Section10.html.
[3] [n. d.]. http://www.3gpp.org.
[4] [n. d.]. https://www.gecode.org/doc-latest/MPG.pdf.
[5] 3GPP 2020. 3rd Generation Partnership Project (3GPP). Retrieved June 06, 2020 from http://www.3gpp.org
[6] V. Benes. 1964. Optimal rearrangeable multistage connecting networks. *Bell Syst. Tech. J.* 43, 7 (1964), 1641–1656.
[7] E. Boutillon and H. Harb. 2020. Extended Barrel-Shifter for Versatile QC-LDPC Decoders. *IEEE Wireless Communications Letters* (2020), 1–1. https://doi.org/10.1109/LWC.2020.2964208
[8] C. Chavet and P. Coussy (Eds.). 2015. *Advanced Hardware Design for Error Correcting Codes* (springer-verlag ed.). 177–192, ISBN 978–3–319–10568–0.
[9] X. Chen, S. Lin, and V. Akella. 2010. QSN—A Simple Circular-Shift Network for Reconfigurable Quasi-Cyclic LDPC Decoders. *IEEE Trans. on Circuits and Systems II: Express Briefs* 57, 10 (Oct 2010), 782–786. https://doi.org/10.1109/TCSII.2010.2067811
[10] Y. Cui, X. Peng, Z. Chen, X. Zhao, Y. Lu, D. Zhou, and S. Goto. 2011. Ultra low power QC-LDPC decoder with high parallelism. In *2011 IEEE Int. SOC Conference*. 142–145.
[11] H. Harb and C. Chavet. 2020. Fully Parallel Circular-Shift Rotation Network for Communication Standards. *IEEE Transactions on Circuits and Systems II: Express Briefs* (2020), 1–1.
[12] J. Lin, Z. Wang, L. Li, J. Sha, and M. Gao. 2009. Efficient Shuffle Network Architecture and Application for WiMAX LDPC Decoders. *IEEE Transactions on Circuits and Systems II: Express Briefs* 56, 3 (March 2009), 215–219. https://doi.org/10.1109/TCSII.2009.2015353
[13] W. Lin, T. . Sheu, C. R. Das, T. . Feng, and C. . Wu. 1988. Fast data selection and broadcast on the Butterfly network. In *[1988] Proceedings. Workshop on the Future Trends of Distributed Computing Systems in the 1990s*. 65–72. https://doi.org/10.1109/FTDCS.1988.26681
[14] M. Milicevic and P. G. Gulak. 2018. A Multi-Gb/s Frame-Interleaved LDPC Decoder With Path-Unrolled Message Passing in 28-nm CMOS. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 26, 10 (2018), 1908–1921.
[15] H. Motozuka, N. Yosoku, T. Sakamoto, T. Tsukizawa, N. Shirakata, and K. Takinami. 2015. A 6.16Gb/s 4.7pJ/bit/iteration LDPC decoder for IEEE 802.11ad standard in 40nm LP-CMOS. In *2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. 1289–1292.
[16] H. Moussa, O. Muller, A. Baghdadi, and M. Jezequel. 2007. Butterfly and Benes-Based on-Chip Communication Networks for Multiprocessor Turbo Decoding. In *2007 Design, Automation Test in Europe Conference Exhibition*. 1–6. https://doi.org/10.1109/DATE.2007.364668
[17] H. Moussa, O. Muller, A. Baghdadi, and M. Jezequel. 2007. Butterfly and Benes-Based on-Chip Communication Networks for Multiprocessor Turbo Decoding. In *2007 Design, Automation Test in Europe Conference Exhibition*. 1–6. https://doi.org/10.1109/DATE.2007.364668
[18] T.; Lee H. Nguyen, T.T.B.; Nguyen Tan. 2019; 8(6):668.. Efficient QC-LDPC Encoder for 5G New Radio. *Electronics*. (2019; 8(6):668.).
[19] S. U. Reehman, C. Chavet, P. Coussy, and A. Sani. 2015. In-place memory mapping approach for optimized parallel hardware interleaver architectures. In *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*. 896–899. https://doi.org/10.7873/DATE.2015.1055
[20] A. Sani, P. Coussy, and C. Chavet. 2016. A dynamically reconfigurable ECC decoder architecture. In *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*. 1437–1440.
[21] A. H. Sani, P. Coussy, and C. Chavet. 2013. A First Step Toward On-Chip Memory Mapping for Parallel Turbo and LDPC Decoders: A Polynomial Time Mapping Algorithm. *IEEE Transactions on Signal Processing* 61, 16 (Aug 2013), 4127–4140. https://doi.org/10.1109/TSP.2013.2264057
[22] K. T. Sarika and P. P. Deepthi. 2014. 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*. 326–330. https://doi.org/10.1109/ICCSP.2014.6949855
[23] A. Tarable, S. Benedetto, and G. Montorsi. 2004. Mapping interleaving laws to parallel turbo and LDPC decoder architectures. *IEEE Transactions on Information Theory* 50, 9 (Sep. 2004), 2002–2009. https://doi.org/10.1109/TIT.2004.833353

[24] M. J. Thul, F. Gilbert, and N. Wehn. 2002. Optimized concurrent interleaving architecture for high-throughput turbo-decoding. In *9th International Conference on Electronics, Circuits and Systems*, Vol. 3. 1099–1102 vol.3. https://doi.org/10.1109/ICECS.2002.1046443

[25] S. Lin X. Chen and V. Akella. 2010. QSN : A Simple Circular-Shift Network for Reconfigurable Quasi-Cyclic LDPC Decoders. *IEEE Trans. on Circuits and Systems II: Express Briefs* 57, 10 (Oct 2010), 782–786.

[26] Z. Zhong, Y. Huang, Z. Zhang, X. You, and C. Zhang. 2020. A Flexible and High Parallel Permutation Network for 5G LDPC Decoders. *IEEE Transactions on Circuits and Systems II: Express Briefs* (2020), 1–1.

Response to the Reviewer's Comments

Hassan Harb Cyrille Chavet

October 28, 2020

1 Answer to the comments of the editor and reviewers

First of all, we would like to thank the editor and the reviewers for their interest in our work and for their positive opinions. We took into account the suggestions and comments coming from three different reviewers. We appreciate the opportunity to clarify our research objectives and results.
Finally, we also want to mention that between the submission of the first version of this paper and the reception of the reviewer's comments, we found some interesting papers. As a consequence, we decided to update the state of the art in this revised version. One of this new papers targets the same problem we are tackling in our works. Hence, this paper has been included in our experimental section, and we discussed the strength and weaknesses of this proposed solution versus our model.

2 Answers to Reviewer 1's comments

This paper proposed an interconnection network architecture that can handle several types of code with different permutation requirements with multiple frames in parallel. The proposed scheme is based on a back-to-back butterfly network with the path finding mechanism that finds the conflict-free path via appropriate constraints solver tools.

Your positive feedback is highly appreciated.

Although the authors presented the motivation and the challenges of this work well, and the proposed method sounds reasonable for addressing the challenges, there is lack of performance evaluation for the readers to understand the superiority of the proposed method.

Note that there is only one table showing the comparison of the required number of MUXs with the other networks. Let alone the proposed method is not the most efficient one in terms of the number of MUXs, it is not clear how such complexity comparison affects the overall processing speed. It would be desirable to include more performance evaluation,

such as the maximum frame parallelism for different frame lengths as it is shown in [11].

In this revised paper, we add more information in the result table. We try to follow your advice concerning the lack of information, and we also add comparison with a new paper we found in September.

3 Answers to Reviewer 2's comments

In this work, the authors proposed a formal model for selecting appropriate paths in a Back-to-Back Butterfly Network, a network which can be applied to 5G. The manuscript is an extended version of authors' ICASSP 2020 paper. The differences, between the current version and the previous one, are basically a new section for related work and an updated Table I. The new section 2 gives a better background for the state of the art. In the new Table I, the authors compare the network complexities (in terms of MUX numbers) between the proposed model and QSN/EBS/FPCN, which makes their results more convincing than the one reported in ICASSP.

Thank you so much for your positive feedback.

Though there are limits in QSN/EBS/FPCN, I would like to see the computing time required by these models as well.

In this version of the paper, the computing time analysis for each algorithm is added to Table I. As mentioned earlier, we also added new results in this section.

4 Answers to Reviewer 3's comments

Thank you for your submission. There are many emerging communication standards, and in particular, 5G requires several different configurations of error-control decoders. This paper addresses the hardware aspects of this issue by exploring techniques based on butterfly networks to make QC-LDPC decoder permutations configurable. The paper is fairly well written (albeit with some minor language issues that should be fixed but do not hinder the underlying technical aspects of the paper), with clear figures and pseudocode.

We highly appreciate your positive opinion. Thank you.

Please see my minor comments/suggestions below (line numbers according to the ones inserted on the left when submitting the paper):
p.1 line 19: "to solve efficiently" ⇒ "to efficiently solve"
p.1 line 29: "in wireless communication domain" ⇒ "in the wireless communication domain"
p.1 line 38: put "PEs" in parentheses

p.2 line 22: "5G QC-LDPC" $\Rightarrow$ "the 5G QC-LDPC"
p.3 lines 6 and 9: Should it be "down to"? (Algorithm 1)
p.5 line 50: "2.n" looks like "two point n". Is the dot really a multiplication? This also appears in Algorithms 1 and 3 and in many places in the text, e.g., p.6 lines 19-20.
p.7 line 28: "our experimentation's" $\Rightarrow$ "experimentations"
p.8: properly capitalize all acronyms in the Reference list, e.g., ref [7] should have "QC-LDPC"

Thank you for highlighting these points.
All the desired modifications are considered except the one that is related to the dot (multiplication). In fact, we decided to assign the dot as a multiplication symbol in order to mitigate the algorithms.