Optimum MDC FFT Hardware Architectures in Terms of Delays and Multiplexers

Mario Garrido, *Senior Member, IEEE*, and Pedro Paz

***Abstract*—In this brief, we show how to derive all the optimum multi-path delay commutator (MDC) fast Fourier transform (FFT) hardware architectures in terms of delays and multiplexers and calculate the number of such architectures. The proposed approach is based on analyzing the orders at the FFT stages that lead to optimum number of delays and multiplexers. The results show that there exist a large number of optimum MDC FFTs. This large design space can be explored in the future in order to design efficient MDC architectures that not only optimize the number of delays and multiplexers, but also other figures of merit such as the number of rotators or the input/output data order.**

***Index Terms*—Bit-dimension permutations, FFT, MDC.**

I. Introduction

In THE research field of fast Fourier transform (FFT) hardware architectures [1], multi-path delay commutator (MDC) architectures [2]–[11] are among the most popular ones. They allow for very high throughput in the range of giga samples per second (GS/s) and a relatively small area for such a high data rate. Indeed, recent designs [7] report rates over 1 GS/s, while using only around 1% of a Virtex-6 field-programmable gate array (FPGA).

MDC FFT architectures consist of a series of stages with butterflies, rotators and shuffling circuits. They calculate the FFT in a continuous flow, and the parallelization of the architecture, P, is directly related to the throughput, which is equal to P times the clock frequency.

In the last years, it has been observed that MDC FFT architectures allow for different data orders at each stage of the architecture [7]. This flexibility in the order at each stage enables the search for orders that are more favorable in terms of the complexity of the rotators, as was done in [7]. However, not every set of orders at the FFT stages leads to the optimum number of delays and multiplexers. In fact, although some previous works present MDC architectures with optimum number of delays and multiplexers [2]–[7], the design space for optimum MDC FFTs is much wider and has not been explored in depth yet.

Manuscript received April 22, 2020; revised July 7, 2020; accepted September 2, 2020. Date of publication September 8, 2020; date of current version February 26, 2021. This work was supported by the Spanish Ministry of Science, Innovation and Universities under Ramón y Cajal Grant RYC2018-025384-I. This brief was recommended by Associate Editor J. H. Xie. *(Corresponding author: Mario Garrido.)*

The authors are with the Department of Electronic Engineering, ETSI de Telecomunicación, Universidad Politécnica de Madrid, 28040 Madrid, Spain (e-mail: mario.garrido@upm.es; p.pazm@alumnos.upm.es).

Digital Object Identifier 10.1109/TCSII.2020.3022528

The goal of this brief is to determine which orders lead to the optimum number of delays and multiplexers and how many optimum architectures in terms of delays and multiplexers exist, which is a step towards the exploration of the design space for optimum MDC FFTs. In order to accomplish this goal, we base this brief on the knowledge about the optimum circuits for bit-dimension permutations [12], as this is the type of permutations carried out in FFT architectures. Furthermore, the analysis in this brief is valid for any radix, due to the fact that radices only differ in the rotations, being any data order valid for any radix.

This brief is organized as follows. In Section II, we review the FFT algorithm. In Section III, we review the concepts related to optimum circuits for bit-dimension permutations that are used to derive the optimum FFT architectures in this brief. In Section IV, we review the MDC architecture and set up the context for the research in this brief. In Section V, we calculate the optimum number of delays and multiplexers for an MDC FFT architecture. In Section VI, we derive all the feasible orders that lead to optimum MDC FFT architectures in terms of delays and multiplexers. In Section VII, we count the number of these architectures. Finally, in Section VIII we summarize the main conclusions of this brief.

II. The FFT Algorithm

Figure 1 shows the flow graph of a 16-point radix-2^2 FFT according to the Cooley-Tukey algorithm, decomposed using decimation in frequency (DIF) [1]. The FFT consists of $n = \log_2 N$ stages. At each stage $s \in \{1, \dots, n\}$ butterflies and rotations are calculated. The lower edges of the butterflies are always multiplied by -1. These -1 are not depicted in order to simplify the graphs.

The numbers at the input of the flow graph represent the index of the input sequence, whereas those at the output are the frequencies, k, of the output signal $X[k]$. Finally, each number, ϕ, in between the stages indicates a rotation by $W_N^{\phi} = e^{-j\frac{2\pi}{N}\phi}$.

An index $I \equiv b_{n-1} \dots b_0$ is also added to the left and to the right of the flow graph, where $b_{n-1} \dots b_0$ is the binary representation of I. It can be observed that the butterflies at stage s operate on pairs of data that differ in bit b_{n-s} [5].

III. Bit-Dimension Permutations in the FFT

This section summarizes the main ideas presented in [12] that are needed for developing this brief. More detailed insight in this topic can be found in [12].

Let us consider an n-dimensional space $x_{n-1} \dots x_0$. In this space, a bit-dimension permutation σ is a permutation on the

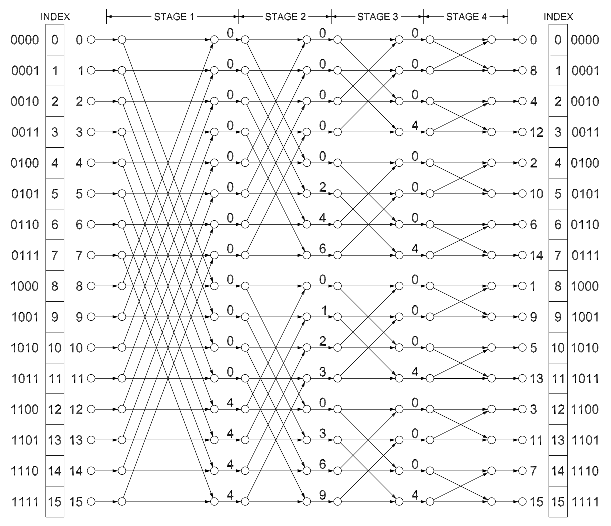
Fig. 1. Flow graph of a 16-point radix-2^2 DIF FFT.

n dimensions that infers a permutation on $N = 2^n$ elements. It has the form

$$\sigma(u_{n-1}u_{n-2}\dots u_0) = u_{\sigma(n-1)}u_{\sigma(n-2)}\dots u_{\sigma(0)}, \quad (1)$$

where $u_{n-1}u_{n-2}\dots u_0$ are the bits permuted into $u_{\sigma(n-1)}u_{\sigma(n-2)}\dots u_{\sigma(0)}$. The position of an element is calculated as

$$\mathcal{P} = \sum_{i=0}^{n-1} x_i 2^i. \quad (2)$$

Thus, there are 2^n positions numbered from 0 to $2^n - 1$ and occupied by the N elements. And the bit-dimension permutation σ changes the position of these elements.

In an FFT architecture, each dimensions x_i can be serial or parallel. Serial dimensions refer to data arriving in series at different time instants, whereas parallel dimensions refer to data arriving in parallel at the same time at different terminals.

In a P-parallel FFT, the number of parallel dimensions is $p = \log_2 P$ and correspond to the dimensions $x_{p-1}\dots x_0$. The other $n - p$ dimensions $x_{n-1}\dots x_p$ are serial dimensions. In order to highlight the nature of the dimensions, serial and parallel dimensions are generally separated by a vertical bar (|), i.e., it is suggested to represent them as $x_{n-1}\dots x_p|x_{p-1}\dots x_0$.

In the FFT, data are permuted in order to change the data order and adapt it between stages. The most basic permutations are those that only exchange two dimensions. These permutations are called elementary bit-exchanges (EBEs). As dimensions are only serial or parallel, there are three types of EBEs: Serial-serial (*ss*), which exchanges two serial dimensions, serial-parallel (*sp*), which exchanges a serial dimension with a parallel one, and parallel-parallel (*pp*), which exchanges two parallel dimensions.

Fig. 2 shows the circuits used to calculate elementary bit-exchanges for *ss* and *sp* permutations. The circuits consist in multiplexers and delays of length L. For *pp* permutations, the circuit is simply an interconnection between the inputs and outputs and, therefore, it does not have any hardware cost.

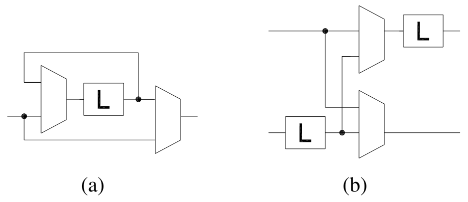

Fig. 2. Circuits for elementary bit-exchanges. (a) Serial-serial permutation. (b) Serial-parallel permutation.

TABLE I
COSTS OF ELEMENTARY BIT-EXCHANGES

Cost	Elementary Bit-Exchange		
	Serial-serial	Parallel-parallel	Serial-parallel
D	$2^j - 2^k$	0	2^j
L	$(2^j - 2^k)/2^p$	0	$2^j/2^p$
M	2^{p+1}	0	2^p

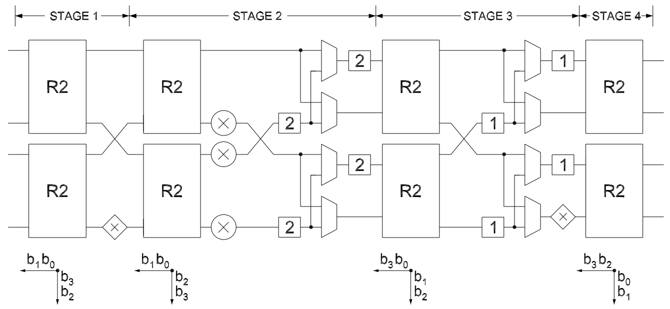

Fig. 3. A 4-parallel 16-point radix-2^2 MDC FFT architecture.

TABLE II
DATA ORDER FOR THE MDC FFT ARCHITECTURE IN FIG. 3

Stage (s)	$x_3x_2\|x_1x_0$
1	$b_1b_0\|b_2b_3$
2	$b_1b_0\|b_3b_2$
3	$b_3b_0\|b_2b_1$
4	$b_3b_2\|b_1b_0$

Table I summarizes the costs of the EBEs in terms of delays (D), length of the buffers (L) and multiplexers (M), for EBEs that exchange dimensions x_j and x_k, where $j > k$.

IV. THE MDC FFT ARCHITECTURE

Figure 3 shows a 4-parallel 16-point radix-2^2 MDC FFT architecture. It consists of butterflies (R2), rotators ($\oplus$ and diamond-shaped) and shuffling circuits like those described in Section III.

The data order at each stage of the architecture is shown at the bottom of the figure in terms of the bits b_i and it is summarized in Table II. The architecture has two serial dimensions, x_3x_2, and two parallel dimensions, x_1x_0. The lowest parallel dimension, x_0, corresponds to the pairs of data that arrive simultaneously at the inputs of the butterflies. It can be observed that $\forall s$, $x_0 = b_{n-s}$. This guarantees that butterflies operate correctly at each stage.

The transformation between the orders at stages s and $s+1$ is carried out by means of the shuffling circuits in the architecture. For instance, the permutation between stages 2 and 3 is

$\sigma(b_1b_0|b_3b_2) = b_3b_0|b_2b_1$. By applying the variable changes $u_3 = b_1$, $u_2 = b_0$, $u_1 = b_3$ and $u_0 = b_2$, the permutation σ can be rewritten as $\sigma(u_3u_2|u_1u_0) = u_1u_2|u_0u_3$. This permutation is calculated by means of two EBEs. The first one exchanges the parallel dimensions x_1 and x_0 and the second one exchanges the serial dimension x_3 with the parallel dimension x_0.

It is important to realize that an MDC FFT architecture can have any order at each stage, as long as the condition $x_0 = b_{n-s}$ is met. However, not every set of orders at different FFT stages leads to the optimum number of delays and multiplexers. In Sections V to VII, we study which sets of orders allow for the optimum number of delays and multiplexers, and how many different such sets there are.

V. Optimum Number of Delays and Multiplexers

Considering the FFT flow graph in Fig. 1, it can be observed that it consists of N input data and N output data. All the inputs are used in the calculation of each of the ouputs. This means that we cannot produce any output before all the inputs have been received. Furthermore, each intermediate computation keeps the amount of data due to the fact that each pair of data produces another pair of data in the butterflies. Therefore, even when some of the inputs have arrived and intermediate calculations have been carried out, the amount of data is never reduced below N until some outputs are provided.

According to this, in a P-parallel architecture, the first outputs can be produced when N data are available, which happens when P input data are at the input of the architecture and, at the same time, $N-P$ data are inside the architecture. As a consequence, the optimum number of delays for a P-parallel N-point FFT is

$$D_{\min} = N - P. \tag{3}$$

From another point of view, a P-parallel N-point FFT has n stages and n dimensions, where p of them are parallel and $n-p$ are serial. The minimum number of EBEs needed to calculate the FFT is $n-1$, which comes from the fact that all b_i must be moved to x_0 at the corresponding stage of the architecture. This minimum number of EBEs occurs when $x_0 = b_{n-1}$ is already provided at the first stage of the architecture. Thus, each EBE fetches a b_i and brings it to x_0, at the same time that another bit of the index is placed in the dimension where b_i comes from. As each EBE fetches a b_i that has not been fetched yet, each of the $n-1$ dimensions $x_{n-1}\ldots x_p|x_{p-1}\ldots x_1$ is exchanged with x_0. The EBEs between parallel dimension do not have any hardware cost according to Table I, whereas the cost for exchanging the parallel dimension x_0 with each serial dimension x_j is $D = 2^j$ delays, which results in a total number of delays for the MDC architecture:

$$D_{\min} = \sum_{j=p}^{n-1} 2^j = 2^n - 2^p = N - P. \tag{4}$$

This result is the same as in (3), which confirms that the optimum number of delays is achieved for the optimum number of EBEs.

Regarding the number of multiplexers, for an sp permutation it is fulfilled [12] that $M_{sp} = s_C P$, and for a pp permutation, $M_{pp} = 0$. As $s_C = 1$ for a sp EBE, the total number of multiplexers in the MDC architecture is

$$M_{\min} = \sum_{i=p}^{n-1} P = P \cdot (n-p) = P\log_2\left(\frac{N}{P}\right). \tag{5}$$

This is, indeed, the minimum number of multiplexers for an MDC FFT. We might wonder if an even smaller number of multiplexers could be achieved. However, all the serial dimensions must be accessed and the only way to access them is through an ss or an sp permutation. Among them, the sp permutation requires less multiplexers according to Table I, which is then the less costly way to access them and, therefore, no other set of permutations can decrease the multiplexer cost below (5).

Furthermore, the alternatives with a number of EBEs that is larger than the minimum one must be considered. First, in case of adding an ss EBE, both the number of delays and multiplexers increase, so this option is not feasible. Second, in case of adding an sp EBE, the number of delays may not increase, because delays in consecutive sp permutations may be simplified if they share a parallel dimension [12]. However, the number of multiplexers always increases when adding an sp EBE. Therefore, this option is not feasible either. Finally, in case of adding a pp EBE, neither the number of delays nor the number of multiplexers increase, as pp EBEs have no hardware cost.

As a result, the optimum number of delays and multiplexers is achieved when the architecture includes any number of pp EBEs and $n-p$ sp EBEs, where each sp EBE involves a different serial dimension. Table III shows the optimum number of delays/multiplexers for a P-parallel N-point MDC FFT.

TABLE III
Optimum Number of Delays/Multiplexers for a P-Parallel N-Point MDC FFT

	N						
P	16	32	64	128	256	512	1024
2	14/6	30/8	62/10	126/12	254/14	510/16	1022/18
4	12/8	28/12	60/16	124/20	252/24	508/28	1020/32
8	8/8	24/16	56/24	120/32	248/40	504/48	1016/56
16	0/0	16/16	48/32	112/48	240/64	496/80	1008/96

VI. Feasible Orders in Optimum MDC FFTs

A P-parallel N-point MDC FFT architecture has $n = \log_2 N$ dimensions from which $p = \log_2 P$ are parallel dimensions. As $x_0 = b_{n-s}$, the bit at stage s for the parallel dimension x_0 is fixed. For the rest of parallel dimensions, the bit can be any other b_i. Furthermore, the order in which the bits b_i at the parallel dimensions appear does not matter, because pp permutations can obtain any permutation of the orders in parallel dimension at no cost. This results in the fact that the number of possible combinations of parallel dimensions is

$$C(n-1, p-1) = \frac{(n-1)!}{(n-p)!(p-1)!}. \tag{6}$$

Note that these are combinations of $n-1$ elements taken in groups of $p-1$, because the element b_{n-s} is always a parallel dimension.

The number of combinations of parallel dimensions at any stage of a P-parallel N-point FFT is shown in Table IV.

TABLE IV
ALTERNATIVES FOR THE PARALLEL DIMENSIONS AT EACH STAGE OF A P-PARALLEL N-POINT MDC FFT

	N						
P	16	32	64	128	256	512	1024
2	1	1	1	1	1	1	1
4	3	4	5	6	7	8	9
8	3	6	10	15	21	28	36
16	1	4	10	20	35	56	84

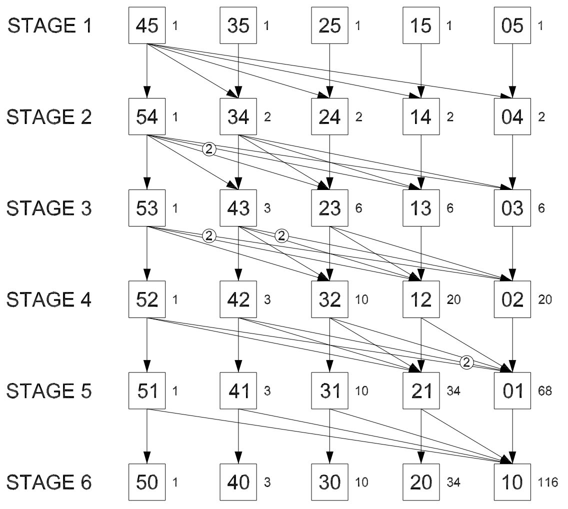

Fig. 4. Allowed transitions of parallel dimensions for a 4-parallel 64-point MDC FFT, and calculation of the number of sequences of parallel dimensions.

Let us consider the case of a 4-parallel 64-point FFT, which has $n = 6$ stages and 5 possible combinations of parallel dimensions per stage. These combinations are shown in Fig. 4. Each number in a square represents one such combination. For clarity, we have written ij instead of $b_i b_j$.

From the combinations of parallel dimensions in Fig. 4, we have to determine which transitions between stages allow for an optimum number of delays and multiplexers. In Section V, we showed that the optimum number of delays and multiplexers is obtained when the architecture carries out $n - p$ sp EBEs plus any number of pp EBEs or, in other words, any serial dimension is permuted only once with a parallel dimension. This means that any transition that results in an increase in the number of sp EBEs must be discarded. For instance, if the order 15 is considered at the first stage of Fig. 4, and the order 54 at the second stage, the bit b_1, which is at the first stage in a parallel dimension, is moved to a serial one. However, this serial dimension will have to be moved again to the parallel dimensions once b_1 must be computed by the butterflies. This will force to access the same serial dimension twice and, therefore, it will not lead to the optimum number of delays and multiplexeres.

In general, the rules that hold are:

- A bit b_i can only be moved from a serial to a parallel dimension before it has been processed at stage $s = n-i$.
- A bit b_i can only be moved move from a parallel to a serial dimension after it has been processed at stage $s = n - i$.

Any movement that contradicts these rules will increase the number of sp EBEs and, therefore, will not result in an architecture with the optimum number of delays and multiplexers.

According to these rules, the arrows in Fig. 4 show the allowed transitions between different orders. This means that any architecture that starts with one of the orders of stage 1 and follows the arrows results in the optimum number of delays and multiplexers. For instance, the sequence of orders $[45, 34, 13, 12, 21, 10]$ results in the optimum number of delays and multiplexers.

Algorithm 1 Checking if a Transition Between the Order O_s at Stage s and the Order O_{s+1} at Stage $s+1$ for an FFT With n Stages is a Valid Transition

```
for $i = 0 : n - (s + 1)$ do
    if $i \in O_s$ & $i \notin O_{s+1}$ then
        return false;
for $i = n - s : n - 1$ do
    if $i \notin O_s$ & $i \in O_{s+1}$ then
        return false;
return true;
```

Algorithm 1 shows the algorithm used to check if a transition between the order O_s at stage s and the order O_{s+1} at stage $s+1$ for an FFT with n stages is a valid transition. Here O_s and O_{s+1} are arrays of numbers that include the values i for the bits b_i that are in parallel dimensions at the corresponding stages.

The algorithm is based on the rules presented before. Therefore, if a certain i that fulfills $0 \leq i \leq n - (s + 1)$ is in O_s, it must also be in O_{s+1}. Otherwise, the transition is not valid. Likewise, a certain i that fulfills $n - s \leq i \leq n - 1$ and is in O_{s+1} must also be in O_s. Otherwise, the transition is not valid. In the end, if all the conditions are met and no false condition is reported, the transition is valid.

TABLE V
EXAMPLE OF CALCULATION OF THE ORDERS AT FFT STAGES

Stage (s)	Order A	Order B
1	0123|45	0123|45
2	0125|34	0125|34
3	0125|43	0125|43
4	0435|12	0345|12
5	0435|21	0345|21
6	2435|10	2345|10

The previous explanation has dealt with the transitions between parallel dimensions, but it remains to analyze the serial dimensions. Indeed, the order of the serial dimensions at stage 1, or any other stage, can be chosen arbitrarily, leading to $(n-p)!$ alternative orders. For instance, the orders 0123|45, 3201|45, and any other permutation of 0123 followed by |45 is a feasible input order at stage 1. However, once the order at stage 1, or another stage, and the sequence of orders of the parallel dimensions are fixed, to achieve an optimum number of delays and multiplexers forces to fix the orders at the other FFT stages. The only degree of freedom happens when more than one parallel dimension changes between two consecutive stages. In this case, any of the parallel dimensions can be moved to any serial position, being the number of alternatives multiplied by $d!$, where d is the number dimensions that change.

Algorithm 2 Counting the Number of Optimum MDC FFTs in Terms of Delays and Multiplexers

```
Seq = zeros(C,n);
Seq(:,1) = 1;
for $s = 1 : n-1$ do,
    for $i = 1$ : OrdersPerStage do
        for $j = 1$ : OrdersPerStage do
            if transition($O_s(i)$, $O_{s+1}(j)$) == true then
                d = diffDim($O_s(i)$, $O_{s+1}(j)$);
                Seq($j$,$s+1$) = Seq($j$,$s+1$) + d! · Seq($i$,$s$);
numOptMDC = sum(Seq(:,$n$)) · $(n-p)!$;
```

As an example, let us consider the order 0123|45 at stage 1 and the sequence of orders [45, 34, 43, 12, 21, 10] for the parallel dimensions. Then, the orders of the FFT are according to Table V. It can be noted that from stage 1 to stage 2, b_3 needs to be moved to a parallel dimension. This forces to store b_5 where b_3 was and the rest of the dimensions are unchanged. Between stages 2 and 3, there is no permutation. Between stages 3 and 4, b_3 and b_4 are exchanged with b_2 and b_1, which leads to $d! = 2! = 2$ alternatives that lead to orders A and B in the table. The orders for the rest of stages are obtained by following the same reasoning.

As a result, each sequence of orders of the parallel dimensions where d_s dimensions change between stage s and $s+1$ allows for a number of orders for the serial dimensions that is equal to

$$(n-p)! \cdot \prod_s (d_s!). \tag{7}$$

VII. Number of Optimum MDC FFTs in Terms of Delays and Multiplexers

The calculation of the number of optimum MDC architectures in terms of delays and multiplexers is based on counting the feasible orders. For this purpose, Fig. 4 includes small numbers outside the squares that keep track of the number of alternatives, whereas numbers in circles keep track of $d!$ Starting from stage 1 and moving down in the graph of Fig. 4, the count at a certain node of the graph is equal to the sum of the counts of the predecessors multiplied by values $d!$ of the transitions. For instance, the node 12 comes from 53, 43, 23 and 13, whose counts are 1, 3, 6 and 6, and $d!$ is 2, 2, 1, 1, respectively. As a result, the count for node 12 is $1 \cdot 2 + 3 \cdot 2 + 6 \cdot 1 + 6 \cdot 1 = 20$. In the end, the total count is the sum of all the counts at the last stage, e.g., $1 + 3 + 10 + 34 + 116 = 164$ in the example of Fig 4.

Note that the previous calculations include all the combinations of parallel dimensions and the effect of $d!$ Therefore, it remains to multiply by the $(n-p)!$ possible orders for the serial dimensions that can be set at the first stage.

Algorithm 2 provides the complete procedure to calculate the number of optimum MDC FFT architectures, where the function 'transition' refers to Algorithm 1. Algorithm 2 has been validated by generating the optimum orders and calculating the cost of the permutations of each of them according to [12] by using MATLAB. Due to the extremely large number of alternatives in large FFTs, the verification has been carried out up to $N = 128$.

TABLE VI
Number of Existing MDC FFT Architectures With an Optimum Number of Delays and Multiplexers, for a Given N and P

	N						
P	16	32	64	128	256	512	1024
2	6	24	120	720	5040	40320	362880
4	28	288	3936	67200	1376640	$3.29 \cdot 10^7$	$8.98 \cdot 10^8$
8	10	158	2516	102264	3693360	$1.59 \cdot 10^8$	$8.05 \cdot 10^9$
16	1	20	588	23424	1194528	$7.48 \cdot 10^7$	$5.57 \cdot 10^9$

As a result, Table VI shows the number of existing MDC FFT architectures with an optimum number of delays and multiplexers, all of them with different orders. This provides optimum results for a variety of input/output data orders. Likewise, in combination with rotator allocation [7], the results in this brief will allow for obtaining architectures that optimize not only the number of delays and multiplexers, but also the number of rotators and their complexity.

VIII. Conclusion

In this brief, we have explained how to derive all the optimum MDC FFT architectures in terms of delays and multiplexers, and we have calculated the number of them as a function of N and P. The results show that there exists a large number of optimum MDC FFT architectures and this number grows significantly with N. This opens new horizons with respect to designing MDC FFT architectures based on a given input/output order, and optimizing the number of rotators and their complexity.

References

[1] M. Garrido, F. Qureshi, J. Takala, and O. Gustafsson, "Hardware architectures for the fast Fourier transform," in *Handbook of Signal Processing Systems*, 3rd ed., S. S. Bhattacharyya, E. F. Deprettere, R. Leupers, and J. Takala, Eds. Cham, Switzerland: Springer, 2019.

[2] N. L. Ba and T. T. Kim, "An area efficient 1024-point low power radix-2^2 FFT processor with feed-forward multiple delay commutators," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 10, pp. 3291–3299, Oct. 2018.

[3] K.-J. Yang, S.-H. Tsai, and G. C. H. Chuang, "MDC FFT/IFFT processor with variable length for MIMO-OFDM systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 4, pp. 720–731, Apr. 2013.

[4] C. Chan, H. Lin, and C. Liu, "High-throughput 64k-point FFT processor for THz imaging radar system," in *Proc. Int. Symp. VLSI Design Autom. Test*, Apr. 2019, pp. 1–4.

[5] M. Garrido, J. Grajal, M. A. Sánchez, and O. Gustafsson, "Pipelined radix-2^k feedforward FFT architectures," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 1, pp. 23–32, Jan. 2013.

[6] M. Garrido, M. Acevedo, A. Ehliar, and O. Gustafsson, "Challenging the limits of FFT performance on FPGAs," in *Proc. Int. Symp. Integr. Circuits*, Dec. 2014, pp. 172–175.

[7] M. Garrido, S. J. Huang, and S. G. Chen, "Feedforward FFT hardware architectures based on rotator allocation," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 2, pp. 581–592, Feb. 2018.

[8] W. Tsai, S. Chen, and S. Huang, "Reconfigurable radix-$2^k \times 3$ feedforward FFT architectures," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2019, pp. 1–5.

[9] J. K. Jang, H. Keun Kim, M. H. Sunwoo, and O. Gustafsson, "Area-efficient scheduling scheme based FFT processor for various OFDM systems," in *Proc. IEEE Asia–Pac. Conf. Circuits Syst.*, Oct. 2018, pp. 338–341.

[10] J. K. Jang, M. G. Kim, and M. H. Sunwoo, "Efficient scheduling scheme for eight-parallel MDC FFT processor," in *Proc. Int. SoC Design Conf.*, Nov. 2015, pp. 277–278.

[11] A. X. Glittas, M. Sellathurai, and G. Lakshminarayanan, "A normal I/O order radix-2 FFT architecture to process twin data streams for MIMO," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 6, pp. 2402–2406, Jun. 2016.

[12] M. Garrido, J. Grajal, and O. Gustafsson, "Optimum circuits for bit-dimension permutations," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 5, pp. 1148–1160, May 2019.