An Efficient Implementation for Linear Convolution with Reduced Latency in FPGA

Dingli Xue,[1] Linda S. DeBrunner,[2] Victor DeBrunner,[2] Zhen Huang,[1] Ying Xiao,[3] and Zhaohang Zhang[3]

[1]*Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing*

[2]*Department of Electrical & Computer Engineering, Florida State University, Tallahassee, FL*

[3]*Department of Electronic Engineering, Tsinghua University, Beijing*

huangzhen@tsinghua.edu.cn

A recently developed linear convolution filter based on Hirschman theory has shown its advantage in saving computations compared with other convolution filters. In this paper, we ameliorate the Hirschman convolution filter with the usage of split-radix algorithm and explore its latency-reduced advantage for the first time. We present a comparison of hardware resource in FPGA for the proposed Hirschman-based filter and other convolution filters. Simulation results indicate that the split-radix Hirschman convolution filter achieves a promising reduction in latency by averagely 18.15% with an acceptable power consumption rise, compared with the main competitor using extended SRFFT. In the case of device capacity limited, the proposed Hirschman convolution filter is still computationally attractive as it performs small-size originator function, instead of larger Fourier transform required by other convolution filters.

1. Introduction: Developing a new approach to compute linear convolution more efficiently is still attractive for digital signal processing and deep learning. Traditional convolution using the Fast Fourier Transform (FFT) has drawbacks such as repetitive usage of twiddle factors and expensive computational load for large signal size. Specifically, it realizes a linear convolution by computing extended cyclic convolutions of zero-padded signal with numerous trivial multiplications by zero and minus one. Some algorithms that consider decomposing large convolution of specific sizes into smaller skew-cyclic computations have been shown in [1–6].

A promising convolution algorithm based on Hirschman theory has been developed with a configuration optimization strategy [7–9]. Meanwhile, fast Discrete Hirschman transforms (DHTs) have been studied in [10, 11] and motivated some related work [12, 13]. Among these fast Hirschman Transforms, a split-radix DHT (SRDHT) introduces an "L"-shaped structure that could reduce nontrivial real computations at an expense of additional permutations. In this paper, we estimate the computational performance for **Hirschman convolution filter** with the assistance of the split-radix algorithm not considered yet. Because the Hirschman convolution has an originator function that shares a similar structure of smaller size with the FFT-based convolution, it is feasible to implement the Hirschman convolution filter using repetitive butterfly network already working well in industry. For the split-radix Hirschman convolution filter and other Hirschman-based/FFT-based filters [10, 14–16], they are all built using an FPGA (Cyclone IV EP4CE6E22C6, Altera) with each filtering procedure clarified. We concentrate on the performance in hardware, typically, latency and power consumption. The split-radix Hirschman convolution filter introduces a reduction of latency in most cases, except for the convolution size approaches the next exponential. It works at high clock frequency (Fmax) with an acceptable power rise. While the device capacity is sometimes limited, the proposed Hirschman convolution filter can perform more efficiently than the traditional FFT-based ones, since the originator function of the SRDHT is relatively smaller and steady in general. Please note that, we currently just propose a prospective Hirschman-based implementation for linear convolution and stimulate future work, rather than surpassing the existing comprehensive tools in industry.

2. Discrete Hirschman Transform: The conventional Heisenberg-Weyl uncertainty is not applicable to describe the joint localization for a discrete signal due to its inherent limitation of merely keeping translation invariance in the continuous-time case [17, 18]. We prefer an entropy-based measure, the Hirschman uncertainty principle (H_p) firstly mentioned in Hirschman's note [19]. For a finite discrete signal x with unit energy, we describe a class of H_p by $H_p(x) = pS(x) + (1-p)S(X)$, where $S(x) = -\sum_{n=0}^{L_x-1} |x[n]|^2 \log_2 |x[n]|^2$ gives the Shannon entropy and X is the Fourier transform of x. Parameter p ($0 \leq p \leq 1$) allows a trade-off between concentrations in both time and frequency domains. We have a lower limit $H_p \geq \frac{1}{2}\log N$ for $p = \frac{1}{2}$ where N is the length of x. There is an unique orthogonal set of basis that minimizes H_p called the Hirschman Optimal Transform (HOT), which is superior to the Discrete Fourier Transform (DFT) in terms of high-resolution and computational complexity [10, 18, 20, 21]. The DHT is the generalization of the HOT with increased hardware flexibility. For $N = LK$, $0 \leq r \leq K-1$ and $0 \leq l \leq L-1$, the DHT is given by

$$X_{DHT}(Lr+l) = \frac{1}{\sqrt{K}} \sum_{n=0}^{K-1} x[Ln+l] W_K^{nr} \tag{1}$$

For $0 \leq n \leq K-1$ and $0 \leq l \leq L-1$, its inverse transform (IDHT) is

$$x_{IDHT}[Ln+l] = \frac{1}{\sqrt{K}} \sum_{r=0}^{K-1} X(Lr+l) W_K^{-nr} \tag{2}$$

where $W_K = e^{-j\frac{2\pi}{K}}$ gives twiddle factors. Note that an LK-point DHT can be realized by periodic shifts of smaller FFTs, specifically, L, K-point FFTs where each K-point FFT is the originator function of the DHT. In this case, the DHT benefits from some familiar structures like butterfly network already working well with FFT-based applications.

3. Split-Radix Hirschman Convolution: Algorithm 1 shows how the Hirschman convolution convolves two finite discrete-time signals x and h, where the length of x is greater ($L_x > L_h$). We normally consider K as a power of 2 and then determine other parameters. We extend the pre-computed Fourier spectrum of h to a proper size LK, permute x, and multiply its LK-point DHT by the extended spectrum of h. We next permute and separate the LK-point IDHT of the product. Segments are overlap-added with residual elements truncated for our desired linear convolution result. Both the Hirschman-based and FFT-based filters share a similar architecture that consists of 5 main parts: Input Part, Transform Part, Multiplication Part, Inverse-Transform Part and Output Part. Total clock cycles are given by $C_{SRDHTconv} = c_{in1} + c_{DHT} + c_{MUL1} + c_{DHT} + c_{out1}$ and $C_{FFTconv} = c_{in2} + c_{FFT} + c_{MUL2} + c_{FFT} + c_{out2}$. Please note that, the fourth terms c_{DHT} and c_{FFT} there refer to the Inverse-Transform Parts in fact. We use the same denotations there since both the Transform and Inverse-Transform Parts have functionally identical structures, except for some multiplications of conjugate twiddle factors. Next, each filtering procedure will be elaborated.

3.1. Input Part: We consider the FFT convolution filter propagates input stream x attached by zeros through a first-in-first-out (FIFO) in 1 clock cycle by streaming. The Hirschman convolution filter requires more operations by loading the input through a FIFO and permuting it to a register in $LJ+1$ clock cycles (lines 5-9, Algorithm 1). It then needs LK clock cycles to duplicate each filter coefficient (line 4, Algorithm 1). If we consider parallel pipelined method here, only $LK+1$ clock cycles are required. So the clock cycle counts are given by $c_{in1} = LK+1$ and $c_{in2} = 1$ for this part.

3.2. Transform & Inverse Transform Parts: For both convolution filters, their Transform and Inverse-Transform Parts are functionally similar, so we prefer to discuss the Transform Part to which the Inverse-Transform Part corresponds. A split-radix algorithm is under consideration for both the Hirschman and Fourier transforms. The split-radix algorithm introduces a special "L"-shaped butterfly network slightly different from that of the radix-2 algorithm, based on a fact that the radix-2 form can be transformed into radix-4 form by simply modifying the exponents of some twiddle factors. It could be realized through a parallel pipelined and shared substructure manner [22]. The Hirschman convolution filter propagates data from the Input Part to a register in LK clock cycles. Next filtering procedure is an LK-point SRDHT computation that contains "L"-shaped butterfly stages by $\log_2 K$ times. This procedure requires $\frac{LK}{3}\log_2 K$ clock cycles for all arithmetic operations.

Algorithm 1: Linear convolution algorithm based on the Hirschman Theory

Input : x and h, of lengths L_x-point and L_h-point ($L_x > L_h$); K, length of originator function
Parameter: J, length of segment separated from x; L, number of segment
Output : C, convolution result

1 Given K, $J \leftarrow K - L_h + 1$ and $L \leftarrow \lceil \frac{L_x}{J} \rceil$
2 Zero-pad x to LK-point
3 $H \leftarrow K$-point DFT of h
4 $\hat{H} \leftarrow$ Repeat each element in H by L times
5 **for** $i \leftarrow 0$ **to** $J - 1$ **do**
6 **for** $j \leftarrow 0$ **to** $L - 1$ **do**
7 $\hat{x}(i * L + j) \leftarrow x(j * J + i)$
8 **end**
9 **end**
10 $\hat{X} \leftarrow LK$-point DHT of $\hat{x}$
11 $\hat{Z} \leftarrow$ Point-wise multiplication of $\hat{X}$ and $\hat{H}$
12 $\hat{z} \leftarrow LK$-point IDHT of $\hat{Z}$
13 **for** $i \leftarrow 0$ **to** $L - 1$ **do**
14 **for** $j \leftarrow 0$ **to** $K - 1$ **do**
15 $z(i * K + j) \leftarrow \hat{z}(j * L + i)$
16 **end**
17 **end**
18 Separate z into L, K-point segments
19 $C \leftarrow$ Overlap-add one segment and the next by $K - J$ elements
20 Truncate the tail for $L_x + L_h - 1$ points C
21 Output the desired convolution result

It next completes an LK-point order reversal and then outputs LK-point result. Similarly, the FFT-based filter loads data in N clock cycles, computes an N-point split-radix FFT (SRFFT) through $\frac{N}{3}\log_2 N$ clock cycles, reverses and then stores N-point result. The clock cycles are given by $c_{DHT} = LK + \frac{LK}{3}\log_2 K + LK + LK = LK(3 + \frac{1}{3}\log_2 K)$ and $c_{FFT} = N + \frac{N}{3}\log_2 N + N + N = N(3 + \frac{1}{3}\log_2 N)$ for different Transform Parts, respectively. We clarify that the Hirschman-based and FFT-based convolution filters perform identically for $K = N$ and $L = 1$. However, it is feasible for the Hirschman-based filter to save clock cycles as $K < N$ since it requires less split-radix butterfly stages than that of its FFT-based competitor.

3.3. Multiplication Part: In the Multiplication Part, the Transform Part output is multiplied by the pre-computed filter coefficients, H. The Hirschman-based and FFT-based convolution filters perform LK-point and N-point complex multiplications, respectively, with clock cycles given by $c_{MUL1} = LK$ and $c_{MUL2} = N$. Two products are inverse-transformed in the Inverse-Transform Part using the same clock cycles as the Transform Part mentioned in *Section IV. B*.

3.4. Output Part: Unlike the FFT-based filter where the first $L_x + L_h - 1$ elements of its output are exactly the linear convolution result, the proposed Hirschman-based filter requires extra LK clock cycles to regroup and overlap-add the output in a manner of addressing (lines 13-17, Algorithm 1). Redundant elements are truncated to keep only the first $L_x + L_h - 1$ numbers remained for the linear convolution result (lines 19-20, Algorithm 1). The clock cycle counts of the Output Part are given by $c_{out1} = LK + (L_x + L_h - 1)$ and $c_{out2} = L_x + L_h - 1$. Consequently, we express the requirements in clock cycles for both split-radix convolution filters by

$$\begin{aligned} C_{SRDHTconv} &= c_{in1} + 2 \times c_{DHT} + c_{MUL1} + c_{out1} \\ &= LK(9 + \frac{2}{3}\log_2 K) + L_x + L_h \end{aligned} \quad (3)$$

$$\begin{aligned} C_{FFTconv} &= c_{in2} + 2 \times c_{FFT} + c_{MUL2} + c_{out2} \\ &= N(7 + \frac{2}{3}\log_2 N) + L_x + L_h \end{aligned} \quad (4)$$

The Hirschman-based filter reduces clock cycles through its Transform/Inverse-Transform Parts, while it requires more to permute and regroup the output additionally. On FPGA environment, a device normally spends more time to realize a complex computation than a simple loading/shifting operation, even though they are both counted by 1 clock cycle. It is inadequate to estimate the latency just according to clock cycles, we should further discuss clock frequency and power consumption.

4. Consumption in Hardware Resources: In this section, we firstly verify the advantage in saving hardware resources for the proposed split-radix Hirschman convolution filter. As explained in [8], a basic Hirschman convolution filter has an optimal configuration among multiple parameter choices for the best computational complexity, which leads to the largest reduction in latency. We conjecture the requirement in hardware resources would vary for different convolution sizes, thus we investigate sizes from 200 to 1288 points involving 4 exponential numbers (256, 512, 1024, and 2048 points). Suppose that the filter size is 30 ($L_h = 30$), the optimal configuration of the Hirschman convolution filter has the originator function to be $K = 128$ points. We implement the split-radix Hirschman convolution filter with other 4 competitors: 2 advanced Hirschman-based filters [10, 16], and 2 FFT-based filters. Since a DHT computation regards K-point DFT as its originator function [18, 21], we consider the usage of fully parallel pipelined manner and repetitive application of butterfly network. All convolution filters are built with an FPGA, Cyclone IV EP4CE6E22C6, Altera and simulated in Quartus II with Intel Core i7-10700 CPU, 2.90GHz. We choose the Slow 1200mV 85C Model. Data set consists of 16-bit fixed-point unscaled numbers and phase factor width is also 16-bit.

4.1. Clock Cycles: Tab. 1 verifies our analysis in Section IV.E that the clock cycles of the SRFFT convolution filter increases massively as the convolution size crosses 256, 512, 1024 points. It is because that these sizes indicate different Fourier computations of 512, 1024, 2048 points. The split-radix Hirschman convolution filter generally introduces reduction in clock cycles but brings more for the convolution size approaching the next exponential due to an increased usage of the originator function. However, its originator function is normally smaller and its butterfly network size is fairly changeless, which is attractive for limited device capacity.

4.2. Clock Frequency: Maximum clock frequency, F_{max}, is a measurement for paths where the source and destination registers or ports are driven by the same clock. Some FPGA providers recommend that the F_{max} should be always regarded as a clock constraint for signal analysis. We make all convolution filters well perform under the device limit. As shown in Tab. 1, the split-radix Hirschman convolution filter generally requires the F_{max} of mean 175.15MHz higher than 154.79MHz and 156.76MHz for the regular and extended SRFFT-based filters, respectively. We also notice that the split-radix algorithm brings more efficiency than the radix-2 version for the Hirschman convolution filter.

4.3. Power: Tab. 1 indicates that the proposed Hirschman convolution filter requires averagely 3.05% more power than that of the main competitor using extended SRFFT [15]. It is because that the Hirschman-based filter yields higher clock frequency due to its efficient computation of smaller-size originator function. Meanwhile, the SRFFT-based competitors have to run larger butterfly-networks of bidirectional Fourier computations for different convolution sizes. The fixed-point bit-shifting design results in the least power dissipation due to its direct accesses to memory and fast multiplications/divisions by 2 using bit-shifting.

4.4. Latency: Fig. 1 compares the latency for all convolution filters in discussion. Two SRFFT-based filters show a similar stepped growth in latency. Compared with the main competitor using the extended SRFFT, the proposed Hirschman convolution filter generally requires less latency by averagely 18.15% through the whole filtering for different convolution sizes. But for some convolution sizes approaching the next exponential numbers, its latency-reduced advantage diminishes since the Hirschman-based filter has increasing requirements of originator function for overlapping procedures. The radix-2 Hirschman-based filter yields more latency as the radix-2 DHT is less efficient than the SRDHT [11]. The fixed-point bit-shifting design requires the largest latency due to its numerous and redundant loading operations. Moreover,

Table 1. Comparison in hardware resources for the FFT-based and Hirschman convolution filters using Cyclone IV EP4CE6E22C6 Slow 1200mV 85C Model, Altera. (convolution size: point; Fmax: MHz, power: W)

Conv Size	FFT-based filter (SRFFT)[14]			FFT-based filter (extended SRFFT)[15]			Hirschman-based filter (R2DHT) [10]			Hirschman-based filter fixed-point shifting) [16]			Hirschman-based filter (SRDHT)		
	Fmax	Clock	Power	Fmax	Clock	Power	Fmax	Clock	Power	Fmax	Clock	Power	Fmax	Clock	Power
200	141.38	3358	0.605	143.04	3358	0.599	163.93	6089	0.619	95.49	7592	0.580	175.80	3700	0.619
264	151.48	6921	0.614	153.70	6921	0.608	165.62	9097	0.625	96.77	10120	0.582	177.42	5513	0.625
328	148.49	6985	0.611	150.07	6985	0.605	165.56	12105	0.622	95.96	12648	0.582	177.10	7326	0.622
392	150.14	7049	0.612	152.19	7049	0.606	164.83	12169	0.624	96.54	15176	0.583	176.93	7390	0.624
456	145.83	7113	0.610	147.76	7113	0.604	165.55	15177	0.625	95.47	17704	0.584	177.21	9204	0.625
520	157.12	14516	0.620	159.35	14516	0.614	165.19	15241	0.626	96.15	20232	0.585	176.92	9268	0.626
584	156.89	14580	0.615	158.77	14580	0.609	164.71	18249	0.628	95.53	22760	0.585	176.20	11081	0.628
648	154.82	14644	0.616	156.95	14644	0.611	163.23	21257	0.627	95.25	25288	0.587	175.49	12894	0.627
712	154.23	14708	0.614	156.42	14708	0.608	163.01	21321	0.630	94.89	27816	0.588	174.94	12958	0.630
776	155.60	14772	0.613	157.41	14772	0.607	162.70	24329	0.633	95.02	30344	0.588	174.33	14772	0.633
840	156.32	14836	0.614	158.31	14836	0.608	162.85	27337	0.632	95.30	32872	0.589	174.84	16585	0.632
904	155.20	14900	0.612	157.51	14900	0.606	161.89	27401	0.633	95.04	35400	0.591	173.86	16649	0.633
968	153.78	14964	0.620	155.74	14964	0.614	161.97	30409	0.631	94.67	37928	0.591	173.50	18462	0.631
1032	162.12	30388	0.630	164.01	30388	0.624	163.08	33417	0.633	95.63	40456	0.592	174.77	20276	0.633
1096	161.76	30452	0.628	163.88	30452	0.622	162.53	33481	0.635	95.14	42984	0.593	174.21	20340	0.635
1160	160.84	30516	0.626	162.74	30516	0.620	162.05	36489	0.638	94.87	45512	0.593	173.63	22153	0.638
1224	160.28	30580	0.627	162.49	30580	0.621	160.88	39497	0.638	94.43	48040	0.594	172.94	23966	0.638
1288	159.89	30644	0.622	161.82	30644	0.617	160.76	39561	0.639	94.53	50568	0.595	172.54	24030	0.639

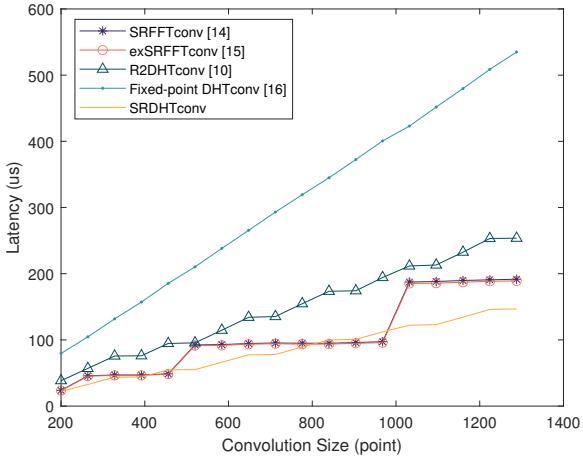

Fig 1 *Comparison in latency for the split-radix Hirschman-based and SRFFT-based convolution filters.*

the latency-reduced advantage of the split-radix Hirschman convolution filter could become even more enhanced with increased convolution size.

5. Conclusion: We verify the latency-reduced advantage for the split-radix Hirschman convolution filter. This proposed filter is implemented using an FPGA (Cyclone IV EP4CE6E22C6, Altera) along with other Hirschman-based/FFT-based convolution filters. It could save latency in most cases of convolution sizes but require more as the convolution size approaches the next exponential. Compared to the main competitor using the extended SRFFT, the proposed filter performs well at higher clock frequency and introduces a reduction in latency by averagely 18.15%, with an acceptable power consumption rise by about 3.05%. In general, we consider this split-radix Hirschman convolution filter as a computationally attractive substitute to realize fast and efficient linear convolution for many applications in signal processing and deep learning.

References

1. Nussbaumer, H.J.: Fast Fourier Transform and Convolution Algorithms. Berlin, Heidelberg: Springer Verlag (1981)
2. Mou, Z.J., Duhamel, P.: Short-length FIR filters and their use in fast nonrecursive filtering. IEEE Trans. Signal Process. 39(6), 1322–1332 (Jun. 1991)
3. Cheng, C., Parhi, K.K.: Hardware efficient fast parallel FIR filter structures based on iterated short convolution. IEEE Trans. Circuits Syst. I 51(8), 1492–1500 (Aug. 2004)
4. Narasimha, M.J.: Linear convolution using skew-cyclic convolutions. IEEE Signal Process. Lett. 14(3), 173–176 (Mar. 2007)
5. Coleman, J.N., et al.: Arithmetic on the european logarithmic microprocessor. IEEE Trans. Comput. 49(7), 702–715 (Jul 2000)
6. Albu, F., et al.: Pipelined implementations of the a priori error-feedback lsl algorithm using logarithmic arithmetic. In: Proc. 2002 IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 3, pp. III 2681–2684. (2002)
7. Xue, D., DeBrunner, L.S., DeBrunner, V.: Linear convolution filter to reduce computational complexity based on discrete hirschman transform. IEEE Signal Process. Lett. 26(12), 1935–1939 (Dec. 2019)
8. Xue, D., DeBrunner, L.S., DeBrunner, V.: Reduced complexity optimal convolution based on the Discrete Hirschman Transform. IEEE Trans. Circuits Syst. I: Reg. Papers 68, 1–9 (Mar. 2021)
9. Wang, W., DeBrunner, V., DeBrunner, L.S.: Fast convolution algorithm for real-valued finite length sequences. In: Proc. IEEE Int'l. Conf. Acoust., Speech, Signal Process. (ICASSP), , pp. 577–580. (May 2023)
10. Xue, D., DeBrunner, L.S., DeBrunner, V.: On computing the Discrete Hirschman Transform. IEEE Trans. Signal Process. 68, 6444–6452 (Nov. 2020)
11. Xue, D., et al.: Split-radix algorithm for the Discrete Hirschman Transform. IEEE Signal Process. Lett. 29, 199–203 (Dec. 2021)
12. Liu, Y., et al.: One-step calculation circuit of FFT and its application. IEEE Trans. Circuits Syst. I, Reg. Papers 69(7), 2781–2793 (2022)
13. Kazemian, M., Abouei, J., Anpalagan, A.: A low complexity enhanced-noma scheme to reduce inter-user interference, ber and papr in 5g wireless systems. Phys. Commun. 48, 101 (2021)
14. Duhamel, P., Hollmann, H.: Split radix fft algorithm. Electron. Lett. 20, 14–16 (1984)
15. Takahashi, D.: An extended split-radix FFT algorithm. IEEE Signal Process. Lett. 8(5), 145–147 (May 2001)
16. Thomas, R., DeBrunner, V., DeBrunner, L.: Fixed-point implementation of Discrete Hirschman Transform. In: Proc. Asilomar Conference on Signals, Systems and Computers, pp. 1507–1511. (Oct. 2018)
17. DeBrunner, V., Ozaydin, M., Przebinda, T.: Resolution in time-frequency. IEEE Trans. Signal Process. 47(3), 783–788 (Mar. 1999)
18. DeBrunner, V., et al.: Entropy-based uncertainty measures for $L^2(\mathbb{R}^n)$, $l^2(\mathbb{Z})$, and $l^2(\mathbb{Z}/N\mathbb{Z})$ with a Hirschman Optimal Transform for $l^2(\mathbb{Z}/N\mathbb{Z})$. IEEE Trans. Signal Process. 53(8), 2690–2699 (Aug. 2005)
19. Hirschman, I.I.: A note on entropy. Amer. J. Math. 79, 152–156 (1957)
20. Przebinda, T., DeBrunner, V., Ozaydin, M.: Using a new uncertainty measure to determine optimal bases for signal representations. In: Proc. IEEE Int'l. Conf. Acoust., Speech, Signal Process. (ICASSP), vol. 3, pp. 1365–1368. (Mar. 1999)
21. Przebinda, T., DeBrunner, V., Ozaydin, M.: The optimal transform for the discrete Hirschman uncertainty principle. IEEE Trans. Inf. Theory 47(5), 2086–2090 (Jul. 2001)
22. Hazarika, J., Ahamed, S.R., Nemade, H.B.: Low-complexity, energy-efficient fully parallel split-radix FFT architecture. Electronics Letters 58(18), 678–680 (2022)