4th International Conference on Advanced Technologies
For Signal and Image Processing - ATSIP' 2018
March 21-24, 2018 - Sousse, Tunisia

FFT Implementation and Optimization On FPGA

Tarek BELABED
National School of Engineering
University of Sousse
belabed.tarek@gmail.com

Sabeur JEMMALI
Laboratory of Advanced Technology and Intelligent Systems, ENISO
University of Sousse
sabeur.jemmali@telecom-paristech.org

Chokri SOUANI
Higher Institute of Applied Sciences and Technology
University of Sousse
chokri.souani@gmail.com

***Abstract*— Nowadays, the development of the Fast Fourier Transform (FFT) remains of a great importance due to its substantial role in the field of signal processing and imagery. This latter still attracts the attention of several researchers around the globe. In this paper, an optimized design of the FFT using the radix-2 algorithm, 32 point is proposed. The developed architecture was implemented using an FPGA regarding its flexibility as well as its parallelism and its computational speed. Though, the material resources of the FPGA are limited, particularly the integrated DSP blocks, a new calculation approach was introduced during the VHDL description with the aim to reduce the necessary number of multiplication operation. The experimental validation of the adopted architecture was realized using a Virtex 6, where the numerical synthesis and the post and route described in VHDL was realized using ISE Design Suite 14.7.**

Keywords—32 point FFT; radix-2; FPGA; DSP; multiplication operation

I. Introduction

Digital processing such as communications, audio and image processing are developing rapidly in various fields, where the Fourier Transform represents the most essential part [1]–[3]. This transform is an alternative method used to describe a discrete signal as a function of frequency [4]. There are several terms for defining the Fourier transform of continuous and discrete signals [5]: Fourier Transform, Fourier series Discrete Time Fourier Transform (DTFT), Discrete Fourier Transform (DFT) and Discrete Fourier Series (DFS).

The only transform used for signal processing is the Discrete Fourier Transform (DFT). This latter requires a large number of multiplications and additions, which makes it very difficult to implement in embedded systems. An optimization method is, therefore, required to facilitate calculations. This method is the Fast Fourier Transform (FFT), developed in 1965 by Cooley and Tukey[6]. Several algorithms, other than Cooley and Tukey's approach, have also been developed based in their approach. These algorithms are the prime factor, split radix, vector radix, vector split radix, Winograd Fourier transform, integer FFT etc [7].

This paper describes an, innovative, optimized architecture based on the FFT radix-2 algorithm. This algorithm allows us to have a massively parallel calculation. The FPGA (Field Programmable Gate Array), was chosen, as a development platform that provides this kind of calculation.

It should be noted that the FPGA hardware resources are still limited, especially the multiplicative blocks (as DSP45E in the virtex 6). The VHDL description was, therefore, optimized in order to have a less expensive architecture.

In the literature many works were deviated towards the implementation of the FFT on different versions of FPGAs such as radix-2, radix-4, R2^{2}SDF, pipelined etc. S. Zhou et al. have realized a high-speed FFT processor, radix-2, 1024 point. The development of this architecture was made under the Quartus II environment, on a Cyclone II FPGA. The results were validated using the Modelsim [8]. C. Yang and H. Chen have proposed the radix-2^2 algorithm implemented on FPGA to be used in a synthetic aperture radar (SAR) [9]. A. Tathode and R. Jassutkar have presented in their paper a 32-bit and 64-point pipelined FFT implementation with new addressing systems where the twaddle factor will be directly generated [10]. A. Haveliya has implemented an 32 point FFT on Virtex 6 FPGA and has verified the proper functioning by a testbench under the Xilinx Design Suite environment [11].

The remaining of this paper is organized as follows. The mathematical development of the DFT and the FFT is detailed in section II. The design, implementation and optimization of the proposed architecture are presented in section III. The simulation results as well as a comparative study of the proposed architecture with anticipated works are provided in section IV. Conclusions are finally depicted in section V.

II. Fourier Transform

A. Discrete Fourier Transform

The Discrete Fourier Transform (DFT) is a method used to describe a discrete digital signal in the frequency domain. In mathematics, we call discrete Fourier transform of a sequence of N terms x(0),x(1),...,x(N -1), the sequence N terms X(0),X(1),...,X(N -1), defined as:

$$X(p) = \sum_{n=0}^{(N-1)} x(n)e^{-j2\pi\frac{np}{N}} \qquad (1)$$

Practically, the N terms x(n) are N samples of a sampled analog signal: $x_n = x\,(nTe)$, where Te is the sampling period, and the N terms X(p) correspond to a frequency approximation of the Fourier transform of this signal to the N points.

B. Fast Fourier Transform

The number of arithmetical operations (multiplication and addition) required by the simple forms of the DFT are of the order N^2. For N = 1024, more than one million complex operations are needed! Consequently, the DFT will not be very useful for several practical applications, especially, for the embedded systems. Fortunately, the optimized algorithm of the FFT is able to reduce the number of these operations.

The Cooley-Tukey optimization algorithm reduces the number of arithmetical operations to $N\log_2 N$. Additionally, if N is power of two, the number of operations will be $N/2\log_2 N$. Compared to the DFT that requires over than one million operations for N = 1024, the FFT takes only around 5120 arithmetical operations. By developing equation 1, the following optimization is obtained:

$$X(p) = \sum_{n=0}^{\left(\frac{N}{2}-1\right)} x_{2n}\, e^{-j\frac{2\pi}{N}(2n)p} + \sum_{n=0}^{\left(\frac{N}{2}-1\right)} x_{2n+1} e^{-j\frac{2\pi}{N}(2n+1)p} \tag{2}$$

$$= \sum_{n=0}^{\left(\frac{N}{2}-1\right)} x_{2n}\, e^{-j\frac{2\pi}{N/2}(np)} + e^{-j\frac{2\pi}{N}p} \sum_{n=0}^{\left(\frac{N}{2}-1\right)} x_{2n+1} e^{-j\frac{2\pi}{N/2}(np)} \tag{3}$$

$$X(p) = A_p + W_N^P * B_p \tag{4}$$

Where A_p, W_N^p and B_p are respectively:

$$A_p = \sum_{n=0}^{\left(\frac{N}{2}-1\right)} x_{2n} e^{-j\frac{2\pi}{N/2}(np)}$$

$$B_p = \sum_{n=0}^{\left(\frac{N}{2}-1\right)} x_{2n+1} e^{-j\frac{2\pi}{N/2}(np)}$$

$$W_N^p = e^{-j\frac{2\pi}{N}p}$$

Since the DFT is periodic, we take this specification to obtain:

$$X\left(p + \frac{N}{2}\right) = A_p - W_N^p * B_p \tag{5}$$

Equations (4) and (5) define the basic structure of the FFT algorithm as the butterfly form represented in Fig.1. Where A_p and B_p are two complex inputs each consists of a real and an imaginary part:

$$A_p = \mathrm{Re}(A_p) + j\,\mathrm{Im}(A_p) \text{ and } B_p = \mathrm{Re}(B_p) + j\,\mathrm{Im}(B_p)$$

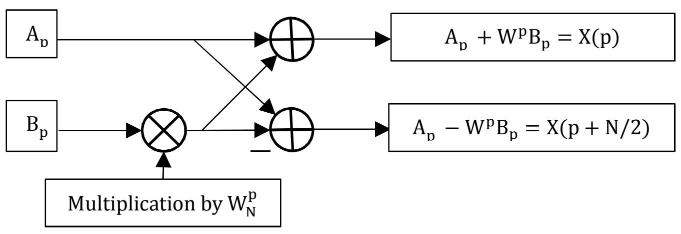


Fig. 1. The butterfly form.

In our architecture, the FFT unit consists of 32 real inputs and 32 imaginary inputs, and the same goes for the outputs. This choice also reduces the complexity of the optimization architecture which will be, briefly, discussed in the subsequent section.

III. OPTMIZATION AND IMPLEMENTATION OF FFT

A. Work with integer twiddle factor

Our architecture is dedicated to processing monochrome images, and then all input values in the 5 stages FFT unit are integer variables. The twiddle factor was subsequently transformed in integer number to increase the computation speed. All exponential coefficients were, therefore, multiplied by a fixed factor set to 2^m. Only the integer part is used, and at the output of the FFT the coefficients are divided by the same factor. This factor, was chosen, to facilitate the division of the obtained results at the output of the FFT unit, all using a simple shift_right equal to m. This operation is very fast and less expensive for an FPGA. Equations 4 and 5 can be written as:

$$X(p) = A_p + (W_N^P * B_p) \gg m \tag{6}$$

$$X\left(p + \frac{N}{2}\right) = A_p - (W_N^p * B_p) \gg m \tag{7}$$

This transformation can give rise to small errors as regards the results obtained, from which it is necessary to choose the correct value of 'm'. We must check this value that gives us the minimum of error. The solution is to apply the FFT then the IFFT on the real and imaginary part of an image 'Ii' to have another image 'If' with the use of different values of m. The variation of the quality of the image is ensured by the calculation of the PSNR [12]:

$$PSNR = 10 * \log_{10}\left(\frac{255^2}{MSE}\right) \tag{8}$$

Where MSE: $$: MSE = \frac{\sum_0^{h1}\sum_0^{h2}(Ii - If)}{h1 * h2} \tag{9}$$

- Ii: initial image of size h1*h2
- If : final image of size h1*h2

Table 1 shows the PSNR corresponding to different values of m. We take a factor with the highest PSNR values, which is the number m=10.

B. Optimisation

In embedded systems, multiplications are essential to achieve a compact system. These operations require various DSP blocks integrated in the FPGA. These integrated circuit are too fast, but very limited, from where rises the need for an optimized algorithmic of the FFT, in order to minimize the necessary DSP blocks.

TABLE I. THE PSNR OF DIFFERENT VALUES OF M

m		**2**	**4**	**6**	**8**	**10**	**12**	**14**
I1	Real part	19.8	36.51	43.85	51.07	51.2	51.18	51.12
	Imaginary part	30.24	45.94	55	87.97	84.83	83.07	85.15
I2	Real part	20.76	37.53	44.61	51.15	51.23	51.23	51.23
	Imaginary part	31.17	47.02	56.19	80.49	83.28	81.82	83.5
I3	Real	22.46	39.34	45.99	51.11	51.16	51.14	51.14
	Imaginary part	32.98	49.5	85.79	80.38	82.14	82.67	82.67

The adopted optimization consists in transforming the multiplication by a sum of left_shift. A left shift of a binary number corresponds to a multiplication by 2, which implies that:

$$\text{left shift by K} \ (\ll \text{K}) = \text{Multiplication by } 2^{\text{K}}$$

This kind of operation can only be done by the power coefficients of 2(2, 4, 8, 16, etc.). The idea here is to introduce other coefficients power of two, which their sum is equal to a twiddle factor. If, for example, $\text{Re}(W_{32}^{3}) = 0.8314$ it will be multiplied by 1024 to quantify it. The obtained result values 851. This twiddle factor is, then, developed as follows:

$$\text{Y1} = \text{X1} + (\text{W} * \text{X2}) \gg \text{m}, \text{with } \text{Re}(W_{32}^{3}) = 851 \qquad (10)$$

$$851 = 1024 - 256 + 64 + 16 + 2 + 1 \qquad (11)$$

$$\text{So: } 851 * \text{X2} = (1024 * \text{X2}) - (256 * \text{X2}) + (64 * \text{X2}) + 16 * \text{X2} + 2 * \text{X2} + 1 * \text{X2} \qquad (12)$$

The following variables are introduced:

$$\text{W10} = 1024 * \text{X2} = \text{X2} \ll 10 \qquad (13)$$

$$\text{W8} = 256 * \text{X2} = \text{X2} \ll 8 \qquad (14)$$

$$\text{W6} = \text{X2} * 64 = \text{X2} \ll 6 \qquad (15)$$

$$\text{W4} = 16 * \text{X2} = \text{X2} \ll 4 \qquad (16)$$

$$\text{W2} = 2 * \text{X2} = \text{X2} \ll 1 \qquad (17)$$

$$\text{W0} = 1 * \text{X2} = \text{X2} \qquad (18)$$

From the equations (12), (13), (14), (15), (16), (17), and, (18), we acquire:

$$\text{Z} = 851 * \text{X2} = \text{W10} - \text{W8} + \text{W6} + \text{W4} + \text{W} + \text{W0} \qquad (19)$$

From the equations (10) and (19), we distinguish:

$$\text{Y1} = \text{X1} + (\text{W} * \text{X2}) \gg \text{m} = \text{X1} + \text{Z} \gg \text{m} \qquad (20)$$

This decomposition is applied to all precalculated and quantified twiddle factors. These factors are represented in table 2.

TABLE II. VALUES OF THE PRECALCULATE TWIDDLE FACTORS

Tawddle factor	W_{32}^{0}	W_{32}^{1}	W_{32}^{2}	W_{32}^{3}	W_{32}^{4}	W_{32}^{5}	W_{32}^{6}	W_{32}^{7}	W_{32}^{8}	W_{32}^{9}	W_{32}^{10}	W_{32}^{11}	W_{32}^{12}	W_{32}^{13}	W_{32}^{14}	W_{32}^{15}
$\mathbf{Re(W_{32}^{p})}$	1024	1004	946	851	724	569	392	200	0	-200	-392	-569	-724	-851	-946	-1004
$\mathbf{Im(W_{32}^{p})}$	0	-200	-392	-569	-724	-851	-946	-1004	-1024	-1004	-946	-851	-724	-569	-392	-200

The number of multiplications used, following this optimization, is equal to zero, which implies that no multiplication blocks are needed in the FFT. These blocks are replaced with additions, subtractions and logical shift. The number of additions will, hence, increase without requiring any multiplicative block.

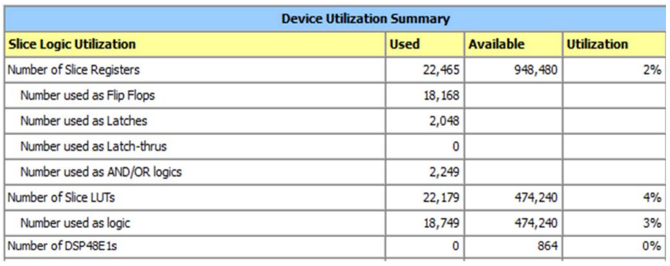
Fig.2 illustrates the number~~s~~ of the DSP45E1s blocks consumed after the synthesis and the place-and-route of the optimized architecture with ISE Design Suite 14.7. We used a Virtex 6 FPGA as a development platform.

C. *Bloc FFT*

The VHDL description is based on signals represented in 'SDT_LOGIC_VECTOR'. The minimum size of these signals should, then, be estimated to choose the size of the RAM memories afterwards. According to the principle of the FFT 32 point formed by 5 stages and with inputs varying between [0: 255], it is necessary to use at least 30-bit vectors for each input and each output, from which signals with 32 bits have been selected to eliminate any uncertainty calculation.

Thus, the FFT unit is formed by 32 buses with 32 bits constituting the real part, and 32 buses with 32 bits for the imaginary part at the input of the unit, and the same goes for the output.

The 5 stages are synchronized by the same clock 'clk', this clock is common for the different units used in our architecture. The reset is ensured by an asynchronous reset signal 'reset_n', and the start of processing is provided by a control signal 'start_fft' as shown in Fig.3.



Device Utilization Summary			
Slice Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	22,465	948,480	2%
Number used as Flip Flops	18,168		
Number used as Latches	2,048		
Number used as Latch-thrus	0		
Number used as AND/OR logics	2,249		
Number of Slice LUTs	22,179	474,240	4%
Number used as logic	18,749	474,240	3%
Number of DSP48E1s	0	864	0%

Fig. 2. Material consumption after the post and route.

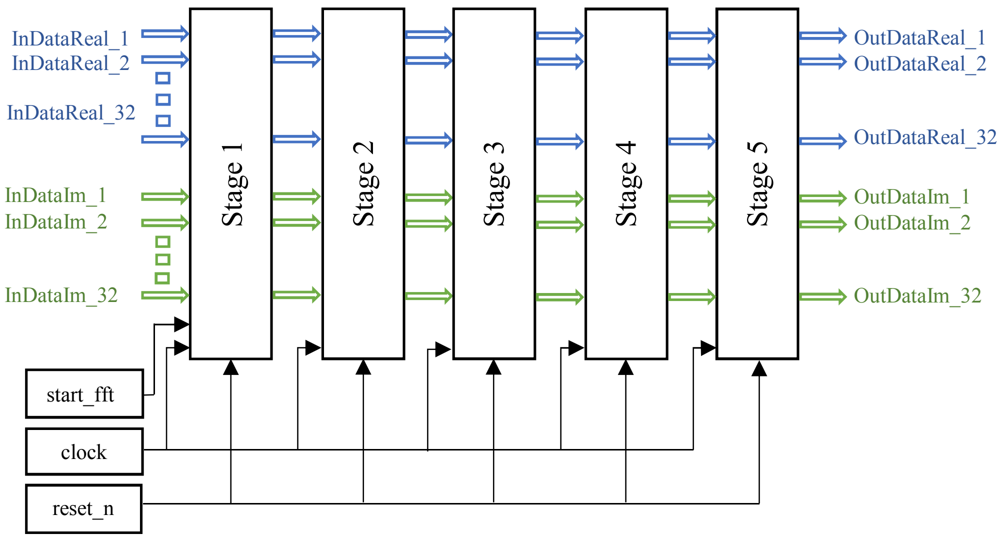


Fig. 3. The 5 stages of FFT 32 point described in VHDL.

The decomposition of the FFT into two parts, real and imaginary, will ensure a massively parallel calculation and minimize the number of clock cycles

D. Design and implementation

In the proposed architecture, we have minimized the read and write time by dividing the RAM into two parts, each has its own addressing bus, one to record the results of the real part of the FFT and the second for the imaginary part. Reading and writing in both RAMs is done at the same time.

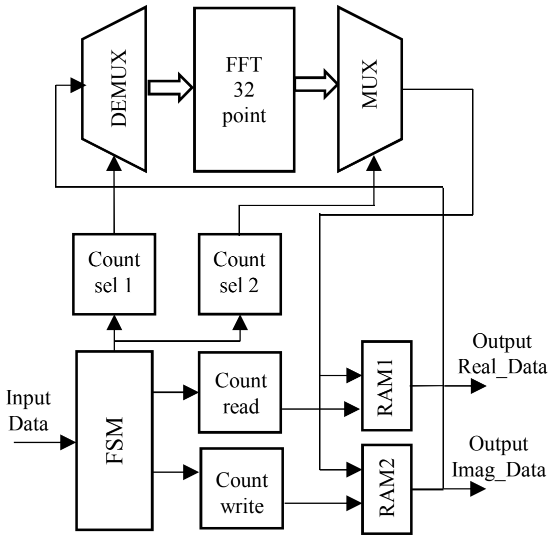


Fig. 4. The differents blocks of our architecture.

Two counters, counter_read and counter_write do the addressing of the two RAMs. The multiplexer and demultiplexer units consist of sending the data recorded in the two RAMs to the FFT, and returning its results to the same RAMs. The results take places of the already calculated data. This kind of recording reduce the RAM space needed.

Two counters counter_sel modulo 32 are used to address the MUX and DEMUX.

The different units constituting our architecture are manipulated by control signals to ensure the proper functioning of the system, thus an FSM (Finite State Machine) to manage and synchronize all the units. Fig.4 illustrates the different units of the proposed architecture.

IV. Results and Discussion

To verify the accuracy of the obtained results after optimization, the same method used to choose the value of m is adopted. Indeed, a 'cameraman' image has been transformed into a text file (.txt) in order to be read as an 8-bit input signal directly linked to the input of our architecture in_data. The calculation of the FFT was carried out on rows and columns to realize the FFT2D of the whole image. The inverse FFT2D was, then, calculated regenerating the reconstructed image typically identical to the initial one. The simulation results are saved as a text file as shown in figure 5. Only 20 clock cycles of the FFT unit are needed to calculate 32 point input data as shown figure 6.

To verify the accuracy of the obtained results, the SSIM index (Structural SIMilarity) between the initial and the reconstructed images have been calculated. We got an SSIM = 99.82%.

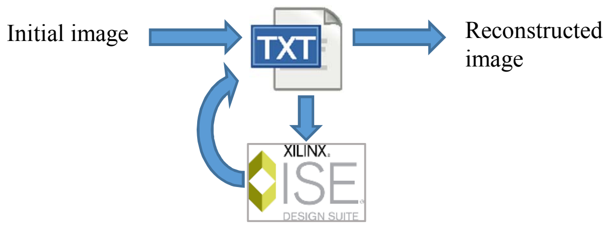


Fig. 5. Method of the verification results.

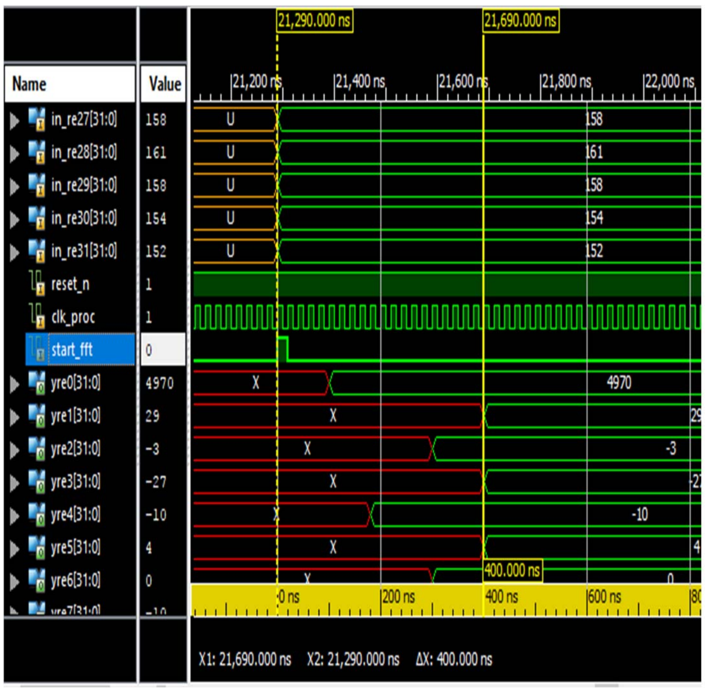


Fig. 6. The number of cycles required to complete the calculation.

The initial and the reconstructed images are shown in figure 7. The conversion of the image to a text file and the verification of the simulation results are realized under the MATLAB environment.

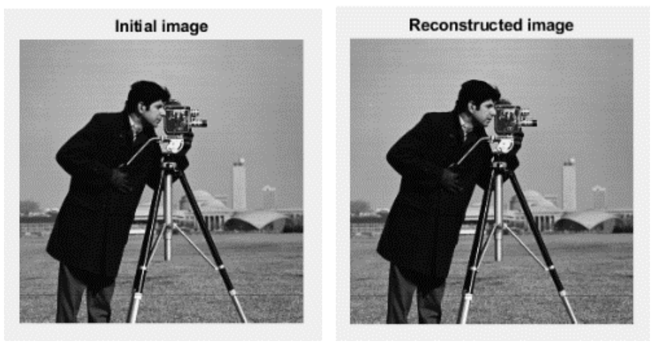


Fig. 7. The initial and reconstructed image .

Our final architecture contains the FFT unit, 1 FSM, 2 RAMs, 2 MUX, 2 DEMUX, 1 counter comp_read, 1 counter compt_write, 2 counter compt_sel. The synthesized results and the post and route of this architecture as compared to anticipated works are summarized in table 3. Our architecture shows the best performances in terms of the needed DSP blocks and the minimum number of cycle required to complete the processing.

V. CONCLUSION

In this paper a 32-point, radix-2, FFT architecture, based on an optimized VHDL description was proposed. The implementation of this architecture was realized on a Virtex 6 FPGA, and its simulation is provided by the ISE Design Suite 14.7 simulator. This optimization aims to minimize as much as possible the number of the needed DSP blocks. The synthesis demonstrates that no DSP blocks are needed in our architecture. The complex numbers were also decomposed into a real and an imaginary part to reduce the processing time and increase the calculation speed. Our architecture represents, therefore, the best solution for FPGAs implemented signal processing applications that require many multiplicative blocks such as FIR and IIR filters etc.

TABLE III. DEVICE UTILIZATION AND COMPARISON OF RESULTS

Device utilization summary				
	G.Zhong et al. [13]	C.Yang et al. [9]	Altera IP core	Our architecture
Algorithme	R2^{2}SDF 32 point	R2^{2}SDF 256 point	- 64 point	Radix-2 32 point
Slice registers	-	5264	-	22465
Slice LUTs	23425	6043	15605	22179
Number of DSP	-	36 DSP48E1s	24 DSP9-bit	**0**
Number of cycles	1046	24	64	**20**

REFERENCES

[1] R. Oshana, “Overview of digital signal processing algorithms, part II - Part 10 in a series of tutorials in instrumentation and measurement,” *IEEE Instrum. Meas. Mag.*, vol. 10, no. 2, pp. 53–58, Apr. 2007.

[2] E. Konguvel and M. Kannan, “A Survey on FFT/IFFT Processors for Next Generation Telecommunication Systems,” *J. Circuits, Syst. Comput.*, vol. 27, no. 3, p. 1830001, Mar. 2018.

[3] R. Tessier and W. Burleson, “Reconfigurable Computing for Digital Signal Processing: A Survey,” *J. VLSI Signal Process.*, vol. 28, no. 1/2, pp. 7–27, 2001.

[4] L. Tan and J. Jiang, “Discrete Fourier Transform and Signal Spectrum,” in *Digital Signal Processing*, Elsevier, 2013, pp. 87–136.

[5] V. Serov, *Fourier Series, Fourier Transform and Their Applications to Mathematical Physics*, vol. 197. Cham: Springer International Publishing, 2017.

[6] J. W. Cooley and J. W. Tukey, “An Algorithm for the Machine Calculation of Complex Fourier Series,” *Math. Comput.*, vol. 19, no. 90, p. 297, Apr. 1965.

[7] K. R. Rao, D. N. Kim, and J.-J. Hwang, *Fast Fourier Transform - Algorithms and Applications*. Dordrecht: Springer Netherlands, 2010.

[8] S. Zhou, X. Wang, J. Ji, and Y. Wang, “Design and implementation of a 1024-point high-speed FFT processor based on the FPGA,” *Image Signal Process. (CISP), 2013 6th Int. Congr.*, vol. 2, no. Cisp, pp. 1112–1116, 2013.

[9] C. Yang and H. Chen, “A efficient design of a real-time FFT architecture based on FPGA,” *Radar Conf. 2013, IET Int.*, vol. 0, no. 2, pp. 1–5, 2013.

[10] A. P. Tathode and R. W. Jassutkar, “Designing of FPGA Based High Performance 32 Bit FFT Processor With BIST,” *Commun. Signal Process. (ICCSP), 2014 Int. Conf.*, pp. 644–648, 2014.

[11] A. Haveliya, “Design and simulation of 32-point FFT using radix-2 algorithm for FPGA implementation,” *Adv. Comput. Commun. Technol. (ACCT), 2012 Second Int. Conf.*, pp. 167–171, 2011.

[12] A. Sghaier, A. Douik, and M. Machhout, “FPGA implementation of filtered image using 2D Gaussian filter,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 7, pp. 514–520, 2016.

[13] G. Zhong, H. Zheng, Z. Jin, D. Chen, and Z. Pang, “1024-Point pipeline FFT processor with pointer FIFOs based on FPGA,” *2011 IEEE/IFIP 19th Int. Conf. VLSI Syst. VLSI-SoC 2011*, pp. 122–125, 2011.