PAPER • OPEN ACCESS

Implementation and Hardware Acceleration of Wiener filter algorithm using Vivado High Level Synthesis HLS for X-ray baggage Images

To cite this article: R Kayalvizhi and S Malarvizhi 2022 *J. Phys.: Conf. Ser.* **2335** 012041

View the article online for updates and enhancements.

You may also like

- Effects of post-deposition treatments on transparent conducting ZnO:B thin films grown by MOCVD
  Taizo Kobayashi and Tokio Nakada

- Prospects for kSZ^2–Galaxy Cross-correlations during Reionization
  Paul La Plante, Jackson Sipple and Adam Lidz

- Ground motion improvements in SPEAR3
  James A. Safranek, Yiton T. Yan, Domenico Dell'Orco et al.

ECS Toyota Young Investigator Fellowship

ECS TOYOTA

For young professionals and scholars pursuing research in batteries, fuel cells and hydrogen, and future sustainable technologies.

At least one $50,000 fellowship is available annually.
More than $1.4 million awarded since 2015!

Application deadline: January 31, 2023

Learn more. Apply today!

This content was downloaded from IP address 136.232.19.2 on 20/01/2023 at 08:48

Implementation and Hardware Acceleration of Wiener filter algorithm using Vivado High Level Synthesis HLS for X-ray baggage Images

R Kayalvizhi and S Malarvizhi

Department of Electronics and Communication Engineering, SRMIST, Kattankulathur, Chennai, India, 603 203.

malarvig@srmist.edu.in

Abstract. Many computer vision applications require a real-time image de-noising technique. In recent years to detecting hazardous materials using X-ray baggage images is important means of airport security applications. The image denoising functions to aid decision making by humans is a predictable area of critical need. However, implementing the image processing approach in Hardware Description Language is time consuming and difficult to debug. With High Level Synthesis HLS, the designer may take advantage of libraries to reap the benefits of working at a higher level of abstraction while producing high-performance hardware. HLS tools, on the other hand, provide prospective solutions to hardware design. HLS tools provide us with extensive design assessment capabilities as well as a wide range of optimization strategies. In this research article, Intellectual Property IP core of wienerfilter algorithm is built utilizing HLS and implemented on ZYNQ platform for X-ray baggage image screening applications. We used Pipeline Pragma techniques added to the HLS environment improved performance, and the results were far better than predicted. The solution using the Pragma method with pipelining was to optimize area and 89 percent clock cycle reduction, and the relevance of the pipeline with this speed was well suitable to improve real time performance.

1. Introduction

This paper in the electronic system design for any application in earlier stages, the design methodologies employ high level languages such as C and C++. The issue occurs when the code needs to be transformed into a Hardware Description Language (HDL) in order implement the code on hardware. It is frequently done manually, which is time-consuming and prone to mistakes [1-2]. In comparison, the C/C++ verification, validating algorithms with HDL is significantly more inefficient [3]. Xilinx's Vivado High Level Synthesis tool, translates C or C++ source file into Register Transfer Language (RTL) implementations, which are subsequently synthesized onto Xilinx FPGA [4]. The Vivado HLS workflow [5] can replace the method of HDL coding and behavioral simulation in a standard FPGA design flow. The hardware and software designer obtained benefits of an HLS design approach are reduction in development time. A high-level synthesis tool is employed in the creation of the image processing technique because it enables the inventor to utilize libraries such as OpenCV, a well-known and generally used library for computer vision applications by software designers [6-12]. High level synthesis saves time because designers are already familiar with libraries. A few literature [13-14] have

Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
Published under licence by IOP Publishing Ltd

compared HLS hardware with RTL and it was electronics system was coded using HDL. According to the findings of [15], the implementation time of a function or part of electronic system executed through HLS is marginally faster than with HDL. These solutions, however, are not tested on actual hardware. HLS based solutions offer a suitable trade-off between development time and performance. In [16-21], authors have used HLS to develop the hardware accelerators of image processing algorithms.

One of the image processing applications is processing of signals acquired in X-ray detection in the baggage inspection system. In X-ray baggage screening application, image has the variety of checked goods in bags, they are prone to overlapping, blocking, and mixing. Furthermore, things are easily impacted by numerous interferences causes during the X-ray image formation process, and the acquired security X-ray images may exist. Image denoising algorithm is easier for security personnel to identify the items in the X-ray image more accurately and quickly, ensuring the airport's safe operation.

In this paper discusses the procedural way of implementing an efficient hardware design Wiener filter useful for many applications in particular for imaging and in general for any time varying signal noise reduction. The developed accelerator Intellectual Property IP core performance with and without pipeline architecture is discussed and functionality verified using X-ray baggage image as input. As a result, this article examines and assesses HLS-based improvements, as well as compares resource efficiency across ZYNQ hardware platform. Section 2 discusses the image denoising wiener filter concepts and Vivado HLS workflow is discussed in section 3. HLS implementation with and without pipeline is dealt in section 4 and experimental conclusions are drawn as conclusion.

2. Image denoising -Wiener filter

Image denoising is the fundamental process in any image processing applications. The purpose of denoising is to reduce noise while preserving visual components like as edges, texturing, and so on as much as feasible. In X-ray baggage screening application, denoising is the important operation in the material discrimination process. The noise typically affects signal quality and it gets added with signal during acquisition and transmission processes and the amount of noise added depends on the type of scanner utilised, the X-ray energy level employed, penetration rate, random photon falling, and detector size. Several popular denoising techniques, [22-25] can significantly improve the medical images, while its performance is not been improved significantly for baggage imagery.

Though there are many filtering techniques available for image denoising, Wiener filter is chosen owing to its simplicity, effectiveness and it is one of the popular adaptive filter used in signal processing applications. In our experimentation, we deploy Wiener filter in HLS and compared for its improvement in the quality of the image and speed in realization in hardware. Constructed baggage Image F of size 512*640 (m,n) from data acquisition is taken for filtering process. Estimates of local mean and variance of each pixel in input image F(m,n) as follows,

$$\mathrm{X} = \frac{1}{\mathrm{ST}} \sum_{\mathrm{m,n} \in \mathrm{w}} \mathrm{F(m,n)} \text{--}(1)$$

$$\mathrm{Y} = \frac{1}{\mathrm{ST}} \sum_{\mathrm{m,n} \in \mathrm{w}} [\mathrm{F}^2(\mathrm{m,n}) - \mathrm{X}^2] \text{ --------------------------- ---------------------- } (2)$$

Where w is the size of window (5*5), S, T is the local neighborhood of each pixel in the image F. X is the local mean and Y is the local variance.

$$\mathrm{G(m,n)} = \mathrm{X} + \left(\frac{\mathrm{Y-Z}}{\mathrm{Y}}\right) [\mathrm{F(m,n)} - \mathrm{X}] \text{ --}(3)$$

Where Z is the noise variance and G is output image of wiener filter process. Using equation 3 the denoised output image G(m,n) chosen area is obtained. This process is repeated for all pixels of the image.

3. Workflow of Vivado HLS

This section describes the methodology of Xilinx HLS tool and the figure 1 provides its work flow. HLS tool synthesis a C or C++ function into an IP block for integrating into hardware. A hierarchy of sub-functions may exist inside the function of C/ C++ code, which describes the functionality of an electronic subsystem. Constraints and directives are also possible inputs for pipelining. FPGA part selection and setting the clock period are also essential limitations.

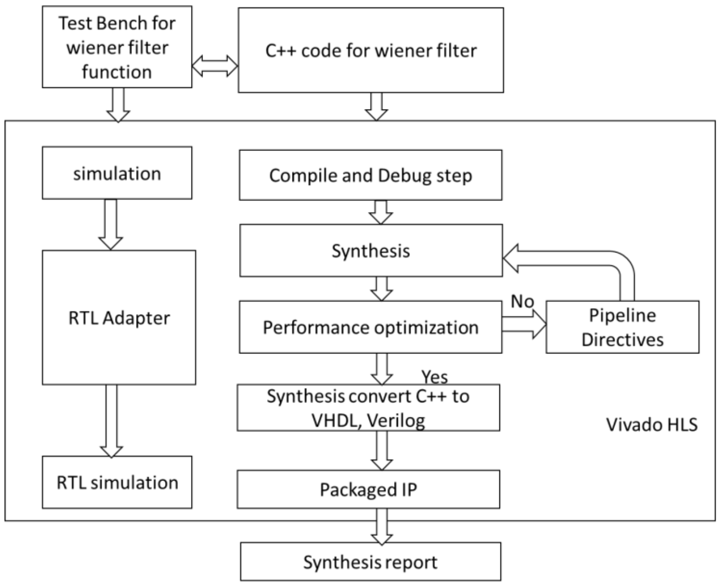

Figure 1. High Level Synthesis process flow of Vivado HLS for Wiener filter

In synthesis, if any C or C++ code has a hierarchy of sub-functions, the final RTL design contains a hierarchy of modules or entities that corresponds to the original C function hierarchy. The same RTL implementation or block is used by all occurrences of the function.

HLS based optimizations-Directives: There are two types of HLS optimizations namely software and hardware optimizations. Software optimizations, such as pipelining, loop (for, while) flattening, array partitioning correlate to compiler directives that enables parallelization. To optimize the hardware utilised of the chosen FPGA device, the constraint-aware architectural adaption and memory access directives can be employed during the flow of IP generation as the FPGA resources limitation. In this experiment pipelining directives and array partitioning are used to optimize the wiener filter code. To construct the source file without any pipeline directives, loop iteration at every time will utilize the same hardware resource as well to operation to remain in the same state. As a result, loops impose a maximal clock cycle for execution, based on condition of the loop and input image F. A one clock cycle required for incrementing the loop counter for each iteration.

In our Weiner filter hardware design, the wiener filter C code will have nested for-loop which requires number of clock cycle as image size, 512 x 640. In order to improve the algorithm's performance, we used Pipeline directives in the source file. As a result, the wiener filter IP core resource utilization must be optimized for hardware acceleration.

Test bench: When utilizing the Vivado HLS design approach, it takes time to synthesize an incorrect functionally in source file. To increase efficiency, employ a test bench to ensure accurate functionality of the function before synthesis and also automatically verifying the RTL simulation output.

The primary output of Vivado HLS is a graphical representation of design format in languages like as Verilog and VHDL in RTL step. As a result, the implementation files such as source files combine to generate that may be used with various Xilinx design tools such as Vivado. It can transform this IP core into a FPGA bit file. This bit file contains all configuration information from the IP block describing the FPGA's internal logic and digital circuit, as well as device-specific information from other files connected with the FPGA target.

4. Implementation of wiener filter and discussion

In the Vivado HLS, a source file of Wiener filter written in C and its test bench are to be defined followed by a path of the complier where the tests need to be run. The target clock period as 10 ns and device for implementation is xc7z020-clg400-2 for the generation of IP core of the wiener filter. IP core is created as output based on the source file – Wiener filter and the test code is for the functionality verification. Figure 2 shows the Data flow of wiener filter IP core built in Vivado HLS.

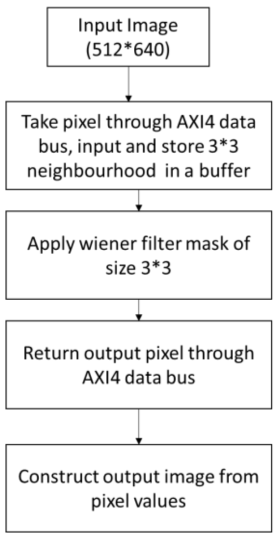
Input Image (512*640)
Take pixel through AXI4 data bus, input and store 3*3 neighbourhood in a buffer
Apply wiener filter mask of size 3*3
Return output pixel through AXI4 data bus
Construct output image from pixel values

Figure 2. Data flow of Wiener filter IP core built in HLS.

After preparing the C source code for wiener filter, synthesised and co-simulation step is performed to verify the RTL function of wiener filter. Following the verification of the RTL, the resultant hardware may be transferred to an FPGA as IP. The effect of introducing pragma to the HLS pipeline, on the other hand, is being examined. Pipelining is for parallelism to reducing the clock cycles used by the algorithm. As seen below, the created wiener filter IP core in figure 3.

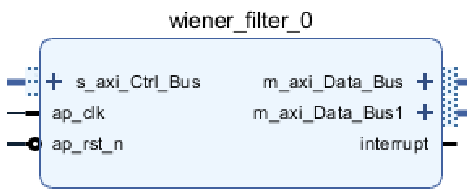

Figure 3. Wiener filter IP core block.

The synthesis report of wiener filter function and its speed of execution with and without pipelining is tabulated in table 1 and 2.

Table 1. Comparison of latency of Wiener filter for image size 512*640 with and without pipeline

	Without pipelining		With pipelining	
	Latency (cycles)	Interval (cycles)	Latency (cycles)	Interval (cycles)
Wiener filter	187	187	19	19

In table 1, the latency indicates the requirement of clock cycles for denoising a whole image, interval refer to clock cycles required to read the next set of input. It is observed from table 1 that 89 percent of clock cycles reduction, using pipeline directives in wienerfilter IP core generation.

Table 2. Comparison of resource utilisation of wiener filter IP core with and without pipeline.

	Without pipelining				With pipelining			
Name	BRAM_18K	DSP48E	FF	LUT	BRAM_18K	DSP48E	FF	LUT
DSP	-	2	-	-	-	4	-	-
Expression	-	6	0	810	-	3	0	778
Instance	4	9	2115	2324	4	6	1777	2131
Memory	7	-	0	-	7	-	0	-
Multiplexer	-	-	-	107	-	-	-	106
Register	-	-	1308	-	-	-	1223	-
Total	11	17	3423	3241	11	3	3000	3015
Available	280	220	106400	53200	280	220	106400	53200
Utilization(%)	3	7	3	6	3	1	2	5

From the synthesis report with the pipelining the performance is significant and the effect on parallelism is observed. From the table 2, the proportion of utilization has also reduced. The pipeline method utilizes just 1% of the DSP48E modules, whereas the non-pipeline version uses 7% of the DSP48E modules. Similarly the LUT and FF resource utilization percentage reduced. Figure 4 depicts a comparison of utilization percentages with and without pipelines, where the X-axis is the resources count and y axis represent the resource utilization.

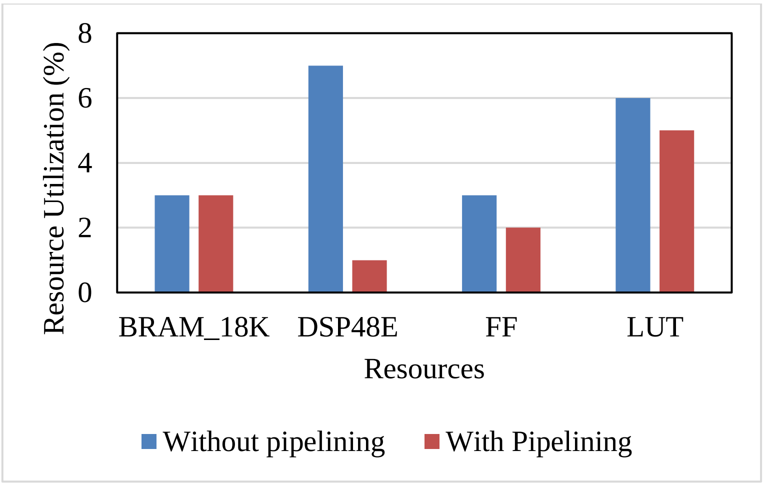

Figure 4. Comparison of hardware FPGA blocks utilization percentages

The Xilinx HLS is used to optimise, synthesise, and export the hardware core. Following that, it is included into the hardware design using Vivado suite. The purpose of this paper is to provide hardware acceleration of a Wiener filter using the Xilinx Vivado suite. Wiener filter IP core integrated ZYNQ processing system IP to generate bit file. Area of the wiener filter implementation shown in figure 5.

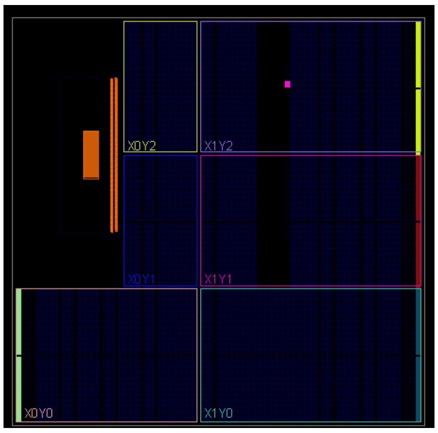

Figure 5. Vivado layout of wiener filter.

The purpose of this acceleration of hardware is to reduce latency. The Xilinx Zynq board was utilised for wiener filter algorithm implementation. Table 3 illustrates the hardware utilisation. In table 3, the first row refers to the resource utilization of the wiener filter IP core exported by HLS and second row refers after applying place and route of the wiener filter IP core. Compare the resource utilisation of wiener filter with existing papers in [26] [27]. From the synthesis report the LUTs, are much fewer than the number anticipated by the synthesis report. The entire design, including the HLS core and other IPs resource utilization is represented by the Vivado.

Table 3. Actual utilization of the wiener filter algorithm.

	LUT	FF	BRAM
HLS EXPORT	3241	3000	10
VIVADO	3551	4516	5
VIVADO [26]	6721	6186	16
VIVADO [27]	8360	2385	13

The experimental results shown in this section have been obtained providing as input to the system the X-ray baggage image, this consist of knife, cotton, sugar shown in figure 6a, with a size of 512×640 pixels and figure 6b, is the resultant image.

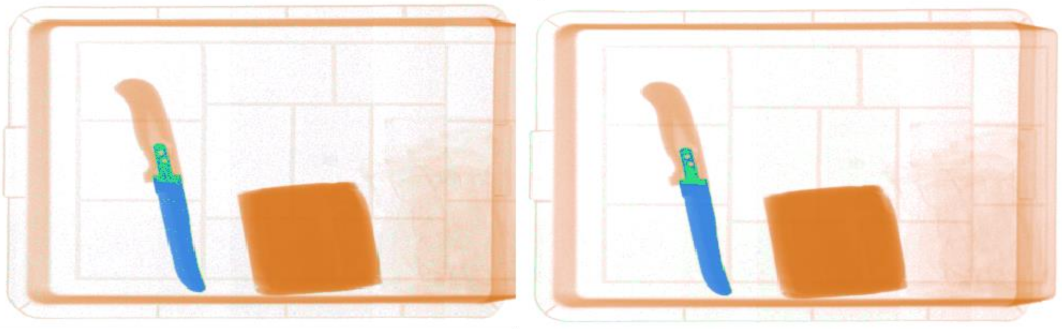

Figure 6. Experimental results (a) Before wiener filter (b) After wiener filter.

We evaluate the visual quality of input image using wiener filter by the peak signal to noise ratio (PSNR) and structure similarity index (SSIM), there is a significant improvement in denoising functionality of the filter design Table 4 list the performance metrics of the wiener filter.

Table 4. Performance metrics of Wiener filter.

Metrics	Before Filtering	After Filtering
PSNR	20.21	21.46
SSIM	0.41	0.43

We measure the PSNR and SSIM values between the filtered image and input image. These results proved that wiener filter is perfect to denoise the X-ray baggage imagery applications.

5. Conclusion

In this work, we demonstrate the HLS constraints and pipeline directives usage optimises the resource utilisation and latency for wiener filter IP core generation and verified the functionality using X-ray baggage image. Image denoising performance evaluated by PSNR and SSIM metrics. After filtering the X-ray baggage image the noise level reduces in 5.9 percent. It is described to construct wiener filter IP core using C-based source file in High Level Synthesis. The findings indicate that to optimise the latency and resource utilisation using pipeline directives, as a result to improve the acceleration speed of hardware. Hardware accelerators can help many computer vision applications increase the performance of real-time, high computational applications. The pipelining method solution using High level synthesis was 89% of clock cycle reduction and optimize the area, it will improve the real-time performance of X-ray baggage screening applications.

Acknowledgments

The authors would like to thank Xilinx WIT program 2021 for providing the tools to conduct experimentation.

References

[1] Martin G and Smith G 2009 High-level synthesis: Past, present, and future *IEEE DesignTest of Computers* **26** 18-25.
[2] Cong J, Liu B, Neuendorffer S, Noguera J, Vissers K and Zhang Z 2011 High-level synthesis for FPGAs: From prototyping to deployment *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **30** 473-491.
[3] Coussy P, Gajski D D, Meredith M, and Takach A 2009 An introduction to high-level synthesis *IEEE Design & Test of Computers* **26** 8-17.
[4] https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_3/ug902-vivado-high-level-synthesis.pdf.
[5] Abdelgawad H M, Safar M and Wahba A M 2015 High level synthesis of canny edge detection algorithm on Zynq platform *Int. J. Comput. Electr. Autom. Control Inf. Eng* **9** 148-152.
[6] Shi Z 2016 Rapid Prototyping of an FPGA-Based Video Processing System (Doctoral dissertation, Virginia Tech).
[7] NM M C and Reddy K R 2013 Implementation of canny edge detection algorithm on fpga and displaying image through vga interface *International Journal of Engineering and Advanced Technology* **2** 243-247.

[8] Lohithaswa M H 2015 Canny edge detection algorithm on fpga *IOSR Journal of Electronics and Communication Engineering.*

[9] Monson J, Wirthlin M and Hutchings B L 2013 Optimization techniques for a high level synthesis implementation of the Sobel filter *International Conference on Reconfigurable Computing and FPGAs* pp 1-6.

[10] Xu Q, Varadarajan S, Chakrabarti C and Karam L J 2014 A distributed canny edge detector: algorithm and FPGA implementation *IEEE Transactions on Image Processing* **23** 2944-2960.

[11] Pawar P H and Patil R P Fpga implementation of canny edge detection algorithm *International Journal of Engineering and Computer Science*, **3** 8704–8709.

[12] Cortes A, Velez I and Irizar A 2016 High level synthesis using Vivado HLS for Zynq SoC: Image processing case studies *Conference on Design of Circuits and Integrated Systems* pp. 1-6.

[13] Dobai R and Sekanina L 2013 Image filter evolution on the xilinx zynq platform *In NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2013)* pp. 164–171.

[14] Le Gal B, Casseaul E, Bomel P, Jego C, Le Heno N and Martin E 2004 High-level synthesis assisted rapid prototyping for digital signal processing *The16th International Conference on Microelectronics* pp. 746–749.

[15] Bhatti F, Greiner T, Heizmann M and Ziebarth M 2017 A new fpga based architecture to improve performance of deflectometry image processing algorithm *In 2017 40th International Conference on Telecommunications and Signal Processing (TSP) IEEE* pp. 559–562.

[16] Casseau E and Le Gal B 2009 High-level synthesis for the design of fpgabased signal processing systems *In 2009 International Symposium on Systems, Architectures, Modeling, and Simulation* pp.25–32.

[17] Garcia A, Santos L, Lopez S, Marrero G, Lopez J F and Sarmiento R 2013 High level modular implementation of a lossy hyperspectral image compression algorithm on a fpga *In 2013 5th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS) IEEE* pp. 1–4.

[18] Said Y, Saidani T and Atri M 2014 High-level design for image processing on fpga using xilinx acceldsp *In 2014 World Congress on Computer Applications and Information Systems (WCCAIS)* pp. 1–5.

[19] Canis A, Choi J, Aldham M, Zhang V, Kammoona A, Anderson J H, Brown S and Czajkowski T 2011 *In Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays - FPGA* pp. 33.

[20] Han Y and Oruklu E 2014 Real-time traffic sign recognition based on zynq fpga and arm socs *In IEEE International Conference on Electro/ Information Technology* pp. 373–376.

[21] Warne D J, Kelson N A and Hayward R F 2014 Comparison of high level fpga hardware design for solving tri-diagonal linear systems *Procedia Computer Science*, vol. 29, pp. 95–101.

[22] Hsieh J 2003 Computed tomography: principles, design, artifacts, and recent advances, *SPIE and John Wiley and Sons*.

[23] Seeram E 2001 Computed tomography: physical principles, clinical applications, and quality control *WB Saunders*.

[24] Yu H, Zeng K, Bharkhada D K, Wang G, Madsen M T, Saba O, Policeni B, Howard M A and Smoker W R K 2007 A segmentation-based method for metal artifact reduction *Academic Radiology*, vol. 14, pp. 495-504.

[25] Duan X, Zhang L, Xiao Y, Cheng J, Chen Z and Xing Y 2008 Metal artifact reduction in CT images by sinogram TV inpainting *In Nuclear Science Symposium Conference Record* pp. 4175-4177.

[26] Neetu A, 2018, Implementation ofWiener Filter on FPGA using Xilinx System Generator for Speech Enhancement. *Int. J. Sci. Innov.Res. Stud*, **6** 1–12.

[27] Yasodai A and Ramprasad A, 2015 Noise degradation system using Wiener filter and CORDIC based FFT/IFFT processor*, J. Cent. South Univ.* **22** 3849–3859.