Bulletin of Electrical Engineering and Informatics
Vol. 7, No. 2, June 2018, pp. 230~235
ISSN: 2302-9285, DOI: 10.11591/eei.v7i2.1167

Signal-to-noise Ratio Study on Pipelined Fast Fourier Transform Processor

S. L. M. Hassan[1], N. Sulaiman[2], S. S. Shariffudin[3], T. N. T. Yaakub[4]
[1, 3, 4]Faculty of Electrical Engineering, Universiti Teknologi Mara, 40450 Shah Alam, Malaysia
[1, 2]Faculty of Engineering, Universiti Putra Malaysia, 43400 Serdang, Malaysia

Article Info

Article history:

Received Feb 01, 2018
Revised May 01, 2018
Accepted May 15, 2018

Keywords:

FPGA
Pipelined FFT
Radix-4
Radix-8
SNR

ABSTRACT

Fast Fourier transform (FFT) processor is a prevailing tool in converting signal in time domain to frequency domain. This paper provides signal-to-noise ratio (SNR) study on 16-point pipelined FFT processor implemented on field-programable gate array (FPGA). This processor can be used in vast digital signal applications such as wireless sensor network, digital video broadcasting and many more. These applications require accuracy in their data communication part, that is why SNR is an important analysis. SNR is a measure of signal strength relative to noise. The measurement is usually in decibles (dB). Previously, SNR studies have been carried out in software simulation, for example in Matlab. However, in this paper, pipelined FFT and SNR modules are developed in hardware form. SNR module is designed in Modelsim using Verilog code before implemented on FPGA board. The SNR module is connected directly to the output of the pipelined FFT module. Three different pipelined FFT with different architectures were studied. The result shows that SNR for radix-8 and R4SDC FFT architecture design are above 40dB, which represent a very excellent signal. SNR module on the FPGA and the SNR results of different pipelined FFT architecture can be consider as the novelty of this paper.

Corresponding Author:

S. L. M. Hassan,
Faculty of Electrical Engineering,
Universiti Teknologi Mara, 0450 Shah Alam, Malaysia.
Email: sitilailatul.mohdhassan@yahoo.com

1. INTRODUCTION

Discrete Fourier transform (DFT) is a well-known algorithm used in digital signal processing applications. While FFT is the resulting algorithm from exploitation of DFT symmetry and periodicity characteristic. FFT reduced the complexity and computational requirement of the DFT from N^2 to N $\log_2$ N, based on Cooley-Turkey algorithm [1]. Shown in Equation 1 is the FFT implementation to compute the complex DFT, where N is number of FFT point and k = 0,1,2, … ,N-1.

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

Coley-Turkey proposed algorithm decomposed the N-point DFT into recurrent 2-point DFT operations, called radix-2 algorithm. This algorithm implements the butterfly block, which consist of an addition and a subtraction operation of complex numbers. Higher radix such as radix-4 and radix-8 applied the same basic algorithm as radix-2 but with reduced multiplication complexity. However, the butterfly structure has higher complexity and its required multiple input complex adders [2]. Previously, this algorithm was on software based. However, with development of semiconductor technology, FFT algorithms is now

possible for hardware implementation. This paper proposed pipelined FFT processor and SNR modules implemented on FPGA board [3], [4], with simulation designs on Modelsim.

Pipelined FFT processor is a popular choice in digital signal processing. Pipelined architecture has the advantage of parallelism and pipeling making it very fast. Futhermore, small number of basic cells can be used repeatedly, reducing the design complexity.[5] Examples of digital design applications using pipelined FFT are ortoghonal frequency division multiplexing (OFDM), code division multiple access (CDMA) and in some of wireless receiver processing block [6]. Other available FFT architectures are memory-based, cache memory, and array architecture. To ensure accuracy and good communication, SNR [7] of this FFT module need to be study. SNR is the ration of the signal relative to the noise in decibels. SNR measure the signals based on this Equation;

$$SNR_{db} = 10log_{10}\left(\frac{P_{signal}}{P_{noise}}\right) \quad (2)$$

Ideally, P_{signal} must be greater than P_{noise} to obtain positive SNR for the signal to be clearly readable. If P_{signal} is equal to P_{noise}, then SNR is equal to zero, the signal is almost unreadable and in digital communication, it will cause reduction in data speed and will force the transmitter to resend back the data. The worse case is when P_{signal} is less then P_{noise} where reliable communication is not possible.

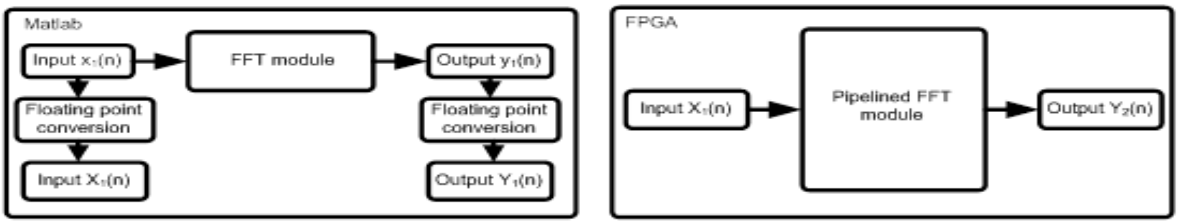
This paper studied SNR on pipelined FFT processor. Three types of pipelined FFT discsuss in this paper are radix-8 [1], [8], [9], radix-4 single-path delay feedback (R4SDF) [7], [9], [10] and radix-4 single-path delay commutator (R4SDC) [9], [11].

2. METHODOLOGY

In this section, the research methodology can be deivided into two stages. First is the pipelined FFT design and verification, and next is the SNR analysis.

2.1. Pipelined FFT Design and Verification

Pipelined FFT and its sub-modules are designed in both Matlab and Modelsim as shown in Figure 1. Random data generated as input $x_1(n)$ to the FFT module in Matlab and converted from floating point to digital form, $X_1(n)$. Matlab output, $y_1(n)$ are also converted to digital value which can be in hexadecimal, binary or any other digital values. Input $X_1(n)$ are then supplied to the pipelined FFT module for hardware simulation for FPGA implementations. Output $Y_2(n)$ are then compared to $Y_1(n)$ for verification, as shown in Table 1, in results and discussion section.



(a) (b)

Figure 1. (a) FFT Module in matlab (b) Pipelined FFT module in modelsim

Figure 2 shows the example of test-bench structure for pipelined FFT modules with sample input and output signal for hardware simulation in Modelsim. Simulation is a very important process since in this process, the desired output of the pipelined FFT is verified.

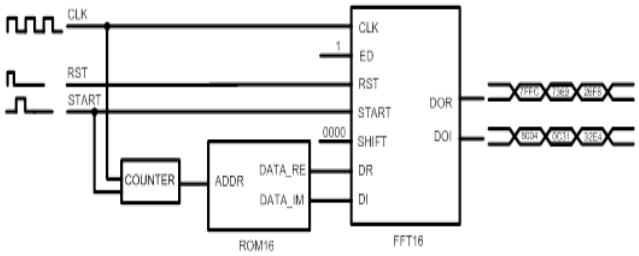


Figure 2. Pipelined FFT testbench

A Read-Only Memory (ROM) is connected to the FFT module inputs. This ROM store inputs data in .txt or .dat file. These data were obtained from Matlab simulation. As a benchmark, pipelined FFT modules were also design in Matlab. This Matlab coding can be modified to suit any pipelined FFT point or radix used. The generated input obtained from Matlab are then supply to the pipelined FFT module in Modelsim. Counter in this test-bench is for generating the address sequence to the ROM module. The ROM will supply the complex signal input data to the FFT modules within the period clock cycle set. Typically, if 16-point pipelined FFT were used, the clock cycle will be 16, and so on depending on the FFT point. Then, the simulation process begins. From the simulation, the inputs and outputs can be displayed, usually in waveform. The results obtain are verified with Matlab.

In this paper, three different FFT architecture were analyzed. First architecture is implemented from radix-8 algorithm. Although it has more complicated control, the design requires fewer twiddle factor multiplication, reducing the memory for stroring the twiddle factor. That is why FFT hardware solution favor higher radix in the implementation. Radix-8 FFT uses several special terms or known as twiddle factor ($W_N^{N/8}W_N^{3N/8}W_N^{5N/8}$ and $W_N^{7N/8}$) to reduce the algorithm complexity as shown Equation 3 and 4, where a is *cos (2π/N)* and b is *sin (2π/N).*

$$(a+jb)W_N^{\frac{N}{8}} = -(a+jb)W_N^{\frac{5N}{8}} = \frac{\sqrt{2}}{2}[(a+b)+j(b-a)] \quad (3)$$

$$(a+jb)W_N^{\frac{3N}{8}} = -(a+jb)W_N^{\frac{7N}{8}} = \frac{\sqrt{2}}{2}[(a-b)+j(b+a)] \quad (4)$$

These complex multiplications can be manipulated using two real-constant multiplications and two additions. Constant multiplication can also be replaced by shift-and-add operation.

Another two pipelined FFT designs taken for comparison are from radix-4 algorithms. In radix-4 algorithm, it utilized four-way symmetry of W_N^{nk} ($W_N^{nk+\frac{N}{4}} = -W_N^{nk+\frac{3N}{4}} = -jW_N^{nk}$) to minimize the number of complex multiplications. Its formula can be derived as Equation 5. For m = 0, 1, 2, …, N/4-1,

$$\begin{aligned}
X[4m] &= \sum_{n=0}^{N/4-1}\left\{x[n]+x\left[n+\frac{N}{4}\right]+x\left[n+\frac{N}{2}\right]+x\left[n+\frac{3N}{4}\right]\right\}.W_{N/4}^{nm},\\
X[4m+1] &= \sum_{n=0}^{N/4-1}\left\{x[n]-jx\left[n+\frac{N}{4}\right]-x\left[n+\frac{N}{2}\right]+jx\left[n+\frac{3N}{4}\right]\right\}.W_N^{n}W_{N/4}^{nm},\\
X[4m+2] &= \sum_{n=0}^{N/4-1}\left\{x[n]-x\left[n+\frac{N}{4}\right]+x\left[n+\frac{N}{2}\right]-x\left[n+\frac{3N}{4}\right]\right\}.W_N^{2n}W_{N/4}^{nm},\\
X[4m+3] &= \sum_{n=0}^{N/4-1}\left\{x[n]+jx\left[n+\frac{N}{4}\right]-x\left[n+\frac{N}{2}\right]-jx\left[n+\frac{3N}{4}\right]\right\}.W_N^{3n}W_{N/4}^{nm}.
\end{aligned} \quad (5)$$

From this radix-4 algorithms, two architecture which are the R4SDFand R4SDC were considered. R4SDF provides high speed operation however it did not reduce the hardware utilization and power consumption. R4SDC give better speed and power as well as area reduction since stages are connected with commutator instead of feedback. [9].

2.2. Signal-to-noise Module

Shows in Figure 3 is the methodology used to evaluate the SNR fitness [7], [12] for FFT modules.

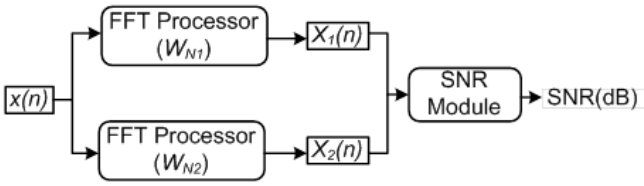


Figure 3. SNR fitness methodology

Initially random input data *x(n)* supplied to FFT processor W_{N1} and this processor will calculate the outputs $X_1(k)$. W_{N1} act as reference solution, solve in Matlab environtment. Next, with the same *x(n)*, W_{N2} FFT processor recalculates the output $X_2(k)$. This FFT processor is in Modelsim (Verilog coding). $X_1(k)$ and $X_2(k)$ are then supplied to SNR module in Modelsim for error calculation, based on equation (1). Pnoise or error calculation *e(k)* can be calculate using Equation 6.

$$e(k) = [R_1(k) - R_2(k)] + j[I_1(k) - I_2(k)] \quad (6)$$

Where R_1, R_2, I_1, and I_2 are the real and imaginary output of FFT for W_{N1} and W_{N2}. Finally, SNR at the output of FFT is calculate using Equation (7).

$$SNR = 10log_{10} \sum_{k=0}^{N-1} \frac{X_1(k)^2}{e(k)^2} \quad (7)$$

Where N is the size of the FFT. Equation (7) can be rewritten as equation (8).

$$SNR = 10log_{10} \sum_{k=0}^{N-1} \frac{[R_1(k)]^2 + I_1(k)]^2}{[R_1(k) - R_2(k)] + j[I_1(k) - I_2(k)]} \quad (8)$$

3. RESULTS AND ANALYSIS

In this section, functionality analysis results on pipelined FFT and the SNR results when the wordlength varied is discussed.

3.1. Pipelined FFT Fuctionality Analysis

Figure 4 shows sample input and output of pipelined FFT from Modelsim simulation. Input start with '1' START signal and ouput start with '1' RDY signal. There is gap between the first input data and first output data because RDY signal can only be initiated when all input data passed all the sub-module in the overall circuit. This output delay is expected in a pipelined architecture.

Input and output of pipelined FFT can be separated into two parts, 16 most-significant -bits (MSB) are real and 16 less-significant-bit (LSB) are the imaginary part. As shown in Figure 4, DR and DI represent real and imaginary input, respectively. While DOR and DOI represent real and imaginary output, respectively. The result shows are in hexadecimal. These hexadecimal values represent complex input in time domain and complex output for frequency domain.

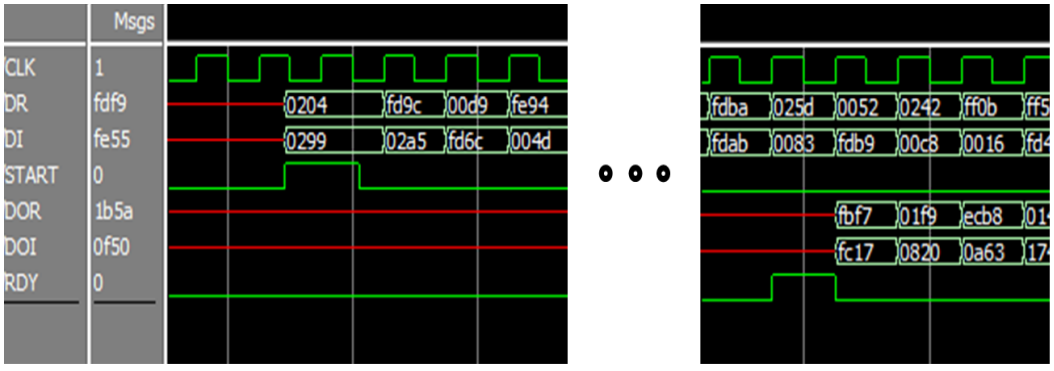


Figure 4. Sample input and output of pipelined FFT

Random inputs data are generated and passed through FFT module in Matlab, producing FFT outputs. These inputs and outputs data are taken as reference data. The same inputs are then supplied to the pipelined FFT modules in Modelsim. The output from this simulation are then compared to the outputs produced in Matlab. Table 1 show inputs and outputs from Matlab as reference and hardware simulation results obtain from three different 16-point pipeline FFT. All generated input and output shown are in hexadecimal. The original input data are in floating point, however, on hardware implementation, pipelined FFT cannot process this type of data. Using Matlab function, input and output data are converted to digital data, hexadecimal.

Table 1. Pipelined FFT Output from Software and Hardware Simulation

	Software simulation (Matlab)		Hardware simulation (FPGA)		
	Input (Hex)	Output (Hex)	Radix-8 Output (Hex)	R4SDC Output (Hex)	R4SDF Output (Hex)
0	F9FD07AE	E605E3F5	E620E3ED	E61FE3EB	E5FBE3EB
1	FE3E0A83	EC53203B	EC75203B	EC752037	EC512037
2	F7DAF9F3	C4F61D3C	C5161D3B	C5171D3A	C4F31D3A
3	F6ECF6AE	3303F654	332BF657	3329F652	3305F652
4	0974020A	2C6B FE9B	2C8E FE99	2C8DFE99	2C69FE99
5	0147F6E8	15C41773	05E91775	15E91775	15C51775
6	090A0320	00C7D4A2	00ECD4A1	00EBD4A0	00C7D4A0
7	FC2E0057	04C71670	04EB166D	04EB1670	04C71670
8	FD7CF524	197A31DE	19A031E1	199F31DF	197B31DF
9	F957F65A	DDC7FEE5	DDE9FEE7	DDE9FEE7	DDC5FEE7
10	F7E8F957	CE0CEBD	CE2EEBCF	CE2DEBD0	CE09EBD0
11	FDE1F478	E1F10F85	E2130F87	E2110F86	E1ED0F86
12	0A4F0B63	007E1493	00A21495	00A11495	007D1495
13	FFC3FFB9	EC3416FD	EC5916FD	EC5916FD	EC3516FD
14	FBD80A3E	FA82DFEB	FAA8DFE9	FAA9DFEA	FA85DFEA
15	FCA6F60B	FD1D2A6F	FD3F2A71	FD3F2A70	FD1B2A70

From table above, the final simulation outputs using Modelsim shows significant and almost similar output to the output generated from Matlab for all different 16-point pipelined FFT used in this research. For example, first generated output from Matlab is E605E3F5, where this paper result is E620E3ED, E61FE3EB for R4SDC and R4SDF obtain E5FBE3EB. Next part on SNR will confirm this statement.

3.2. Signal-to-noise Ratio

Discuss in this sub-section are the SNR results for three different 16-point pipelined FFT. The wordlength are varied from 16-bits to 8-bits for real and imaginary output of the pipelined FFT. Number of bits used will affect all key parameters of design especially power, as well as speed and area. When number of bit reduces, switching activities also reduces, lowering the switched capacitance. This is desirable for power optimization design. Furthermore, lower bits number will result in reduce number of transfer lines and average interconnect length and capacitance.

Shown in Figure 5 are average SNR value for radix-8, R4SDC and R4SDF with varied wordlength. 16 sets randomdly generated input passed through three different pipelined FFT modules. Output of these FFT modules, along with references output are passed through the SNR modules. From graph obtain, the best SNR obtain for radix-8 and R4SDC is at 11-bit word length, while for R4SDF the highest SNR is at 16-bit wordlenght. Table 2 shows the average SNR value for three pipelined FFT architecture studied. Radix-8 have the highest average SNR value compared to the rest. Higher SNR indicates better signal accuracy [1], [7], [12].

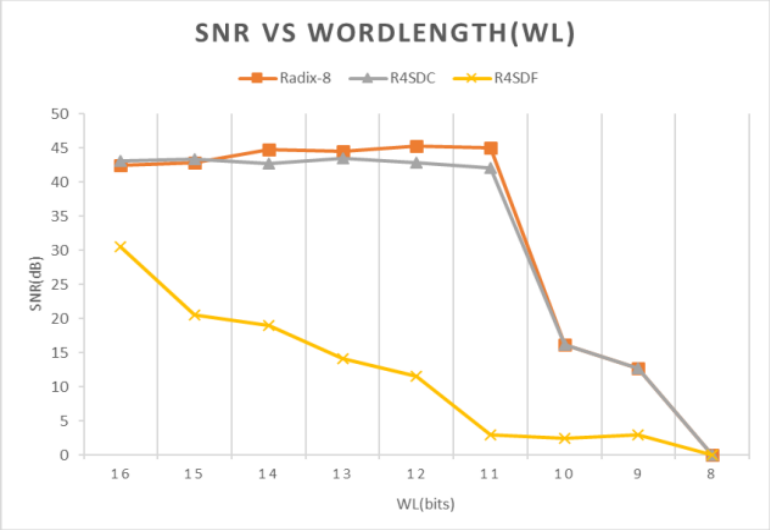


Figure 5. SNR versus wordlength

Table 2. Average SNR Value

FFT Type	SNR (dB)
Radix-8	41.97
R4SDC	40.92
R4SDF	14.87

4. CONCLUSION

In conclusion, all pipelined FFT architectures analyses in this paper are functioning accordingly. Radix-8 pipelined FFT processor have the highest SNR value which is 41.97dB compared to the other architecture studied in this paper. R4SDC also show a very good SNR result, above 40dB indicating an excellent signal. Based on these results, it can be summarized that pipelined FFT in hardware implementation have a good SNR results and can be used in digital signal applications. This study can be futher extend in the futurekk for variation of FFTs size and other architectures.

ACKNOWLEDGMENT

The authors would like to thank Institute of Research Management and Innovation (IRMI), UiTM under LESTARI grant 600-RMI/DANA 5/3/LESTARI (103/2015) for providing necessary financial support and Faculty of Engineering, UPM for their facilities support.

REFERENCES

[1] Sun M, Tian L, Dai D. Radix-8 FFT Processor Design Based on FPGA. *Proceeding in 5th Int. Congress on Image and Signal Processing.* 2012.
[2] Kumar K, Dahiya S, Prakash V. A Comparative Study on High Speed and Low Power Radix FFT. *International* Journal *of Engineering Science and Computing*.2017;7(5):11753-55.
[3] Manjula B. M, Chirag S, FPGA Implementation of BCG Signal Filtering Scheme by Using Weight Update Process. *Indonesian Journal of Electrical Engineering and Computer Science*. 2016;4(2):373-382.
[4] Santhosh K B, Ayyappa S K. FPGA Implementation of a Nakagami-m fading channel Simulator using Random Number Generator. *Indonesian Journal of Electrical Engineering and Computer Science.* 2016;4(2):133-140.
[5] Chiueh T, Tsai P. OFDM Baseband Receiver Design for Wireless Communications.Sinagpore: John Wiley & Sons.2007: 201-208.
[6] Pratima M, Soni M K. Optimized OFDM Model Using CMA Channel Equalization for BER Evaluation. *Bulletin of Electrical Engineering and Informatics*. 2017;6(2):133-139.
[7] Hong Pang J, Sulaiman N. Design of a Reconfigurable FFT Processor using Multi-Objective Genetic Algorithm. *Proceding in International Conference on Intelligent and Advanced Systems*. 2010.
[8] *Rajasekhar* M, Manjula K, Design and Simulation of 512 Point FFT using RADIX-8 Algorithm. *International Journal of Applied Sciences, Engineering and Management.* 2016;5(4):30-33.
[9] Jayaram K, Arun C. Survey report for Radix 2, Radix 4, Radix 8 FFT Algorithms. *International Journal of Innovative Research in Science, Engineering and Technology*.2015;4(7):5149-5154.
[10] Aghaee N, Eshghi M. Design of a Pipeliend R4SDF Processor. *17th European Signal Processing Conference.*2009.
[11] Manimaran A, Suder S.K. A novel VLSI Based Pipelined Radix-4 Single-Path Delay Commutator(R4SDC). *International Journal of Computer Technology and Applications.* 2016; 9(6):2767-2775.
[12] Penchalailah P, Seshadri R. Method for reducing of Noise by Improving Signal-to-Noise-Ratio in Wireless LAN. *International Journal of Network Security&Its Applications*.2011;3(5):115-120.