International Journal of Advanced Multidisciplinary Research
ISSN: 2393-8870
www.ijarm.com
DOI: 10.22192/ijamr Volume 5, Issue 6 -2018

Research Article

DOI: http://dx.doi.org/10.22192/ijamr.2018.05.06.009

Implementation Of Memory Based FFT Architecture Based Processor

Dr. A.V. Prathapkumar, Dr. S. Karthikeyan, K. Yoganand.

Dhanalakshmi Srinivasan Engineering College - Perambalur

Keywords

FFTs,
Architecture,
Verilog HDL,
Xilinx tool.

Abstract

In this brief, we propose a novel approach to implement multiplier less unity-gain single-delay feedback fast Fourier transforms (FFTs). Previous methods achieve unity-gain FFTs by using either complex multipliers or non unity-gain rotators with additional scaling compensation. Conversely, this brief proposes unity-gain FFTs without compensation circuits, even when using non-unity-gain rotators. This is achieved by a joint design of rotators, so that the entire FFT is scaled by a power of two, which is then shifted to unity. This reduces the amount of hardware resources of the FFT architecture. The proposed approach can be Implemented using Verilog HDL and Simulated by Modelsim 6.4 c. Finally its Synthesized by Xilinx tool and Implemented in FPGA Spartan 3 XC3S 200 TQ-144.

Introduction

Digital signal processing (DSP) world, there is often a need to convert signals between time and frequency domains. For this reason, the fast Fourier transform (FFT) has become one of the most important algorithms in the field. In order to calculate the FFT efficiently, various hardware architectures have been proposed. When high performance is required, feedback and feed forward , hardware FFT architectures are attractive options, as they offer high throughput capabilities. Single-delay feedback (SDF) FFT architectures consist of a series of stages that process one sample per clock cycle. Each stage contains a butterfly and a rotator. The butterfly calculates additions, and the rotator carries out rotations in the complex plane by given rotation angles, called twiddle factors. Compared with the additions of the butterfly, rotations are more costly operations. For this reason, different approaches to implement rotators have been proposed in the past.

The most straightforward approach is to use a complex multiplier, which consists of four real multipliers and two adders. In addition, it requires a memory to store the twiddle factors. Another option is to implement the rotators as shift-and-add operations. Following this idea, the CORDIC algorithm breaks down the rotation angle into several successively smaller angles and rotates each of them with a fixed shift-and-add network. Another alternative is to use multiplier-based shift-and-add rotators. By using techniques, such as multiple constant multiplications, these rotators carry out the rotation by reducing the complex multiplier to shift-and-add operations. Among all these alternatives, multiplier-based shift-and-add rotators , are the most efficient option for small set point SDF FFT architecture, using n = 6 stages. Internal structure of an SDF stage. of twiddle factors, whereas the CORDIC-based rotators are the best alternative for large ones. However, CORDIC-based approaches and some multiplier-based shift-and-add rotators, scale the output by a scaling factor R≠ 1. This scaling allows for more accurate and hardware-efficient rotations. In order to achieve unity gain, previous works have compensated the scaling factor by adding a scaling stage to the rotator.

However, this increases the usage of hardware resources. In this brief, we present the novel multiplier less unity-gain SDF FFTs. They are obtained by designing the rotators in all FFT stages simultaneously, so that the output of the FFT has unity gain. Thus, the proposed approach neither requires the use of costly unity-gain rotators, nor circuits to compensate the scaling. This reduces the complexity of the FFT rotators and guarantees unity gain for the FFT.

This brief presents a novel memory-based FFT. The proposed design has several advantages. With respect to the previous approaches, it uses the minimum memory of N samples and a few additional multiplexers. Furthermore, the proposed approach has been implemented using Spartan 3 slices on a field programmable gate array (FPGA). The implementation allows integrating the components of the architecture. This reduces the hardware especially the amount of distributed logic. This reduces the complexity of the FFT rotators and guarantees unity gain for the FFT. In this brief, we study different FFT sizes and propose suitable solutions for each size. The 16 point DIT-FFT architecture is implemented as a Memory based architecture. We done Sixteen point DIT-FFT architecture consist of two points, four points, and Eight Point and Sixteen point as Stages. We are Design a FFT architecture based on Decimation in Time.

Existing system

There exist numerous memory-based FFT architectures in the literature. They mainly differ in the size of the processing element (PE) (butterflies and rotators). This brief presents an approach for improving the accuracy of rotations implemented by complex multipliers, based on scaling the complex coefficients that define these rotations. A method for obtaining the optimum coefficients that lead to the lowest error is proposed. This brief analyzes two different situations where the optimization method can be applied: rotations that can be optimized independently and sets of rotations that require the same scaling. These cases appear in important signal processing algorithms such as the discrete cosine transform and the fast Fourier transform (FFT).

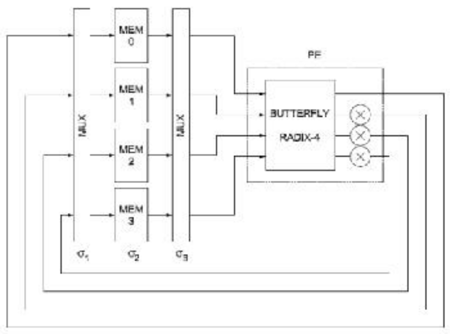


Fig1.Existing memory

Proposed System

We present the novel multiplier less unity-gain SDF FFTs. They are obtained by designing the rotators in all FFT stages simultaneously, so that the output of the FFT has unity gain. Thus, the proposed approach for each size foreach size. neither requires the use of costly unity-gain rotators, nor circuits to compensate the scaling. This reduces the complexity of the FFT rotators and guarantees unity gain for the FFT. In this brief, we study different FFT sizes and propos suitable solutions

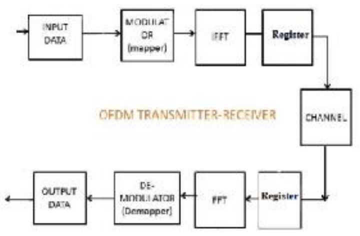


Fig 2.Proposed memory based radix-4 FFT architecture.

Proposed memory based FFT

Basic Architecture

The radix-2 FFT algorithms are used for data vectors of lengths N = 2K. They proceed by dividing the DFT into two DFTs of length N/2 each, and iterating. There are several types of radix- 2 FFT algorithms, the most common being the decimation-in-time (DIT) and the decimation-in-frequency (DIF). This terminology will become clear in the next sections. In this section we present several methods for computing the DFT efficiently.

In view of the importance of the DFT in various digital signal processing applications, such as linear filtering, correlation analysis, and spectrum analysis, its efficient computation is a topic that has received considerable attention by many mathematicians, engineers, and applied scientists.

From this point, we change the notation that *X(k)*, instead of *y(k)* in previous sections, represents the Fourier coefficients of *x(n)*.

Basically, the computational problem for the DFT is to compute the sequence $\{X(k)\}$ of N complex-valued numbers given another sequence of data $\{x(n)\}$ of length N, according to the formula

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \qquad 0 \le k \le N-1$$

$$W_N = e^{-j2\pi/N}$$

In general, the data sequence $x(n)$ is also assumed to be complex valued. Similarly, The IDFT becomes

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k) W_N^{-nk}, \qquad 0 \le n \le N-1$$

Since DFT and IDFT involve basically the same type of computations, our discussion of efficient computational algorithms for the DFT applies as well to the efficient computation of the IDFT.

We observe that for each value of k, direct computation of $X(k)$ involves N complex multiplications ($4N$ real multiplications) and N-1 complex additions ($4N$-2 real additions). Consequently, to compute all N values of the DFT requires N^2 complex multiplications and N^2-N complex additions.

Direct computation of the DFT is basically inefficient primarily because it does not exploit the symmetry and periodicity properties of the phase factor W_N. In particular, these two properties are :

$$\text{Symmetry property}: W_N^{k+N/2} = -W_N^k$$

$$\text{Periodicit y property}: W_N^{k+N} = W_N^k$$

The computationally efficient algorithms described in this sectio, known collectively as fast Fourier transform (FFT) algorithms, exploit these two basic properties of the phase factor.

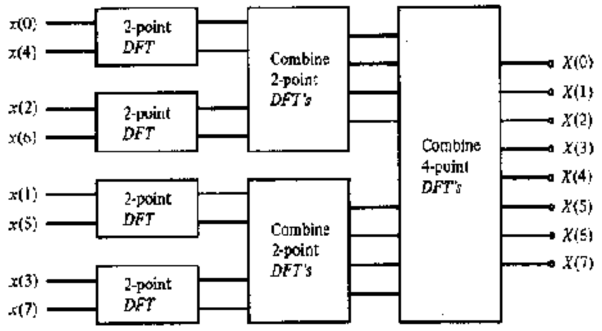


Fig 7.Three stages in the computation of an N = 8-point DFT

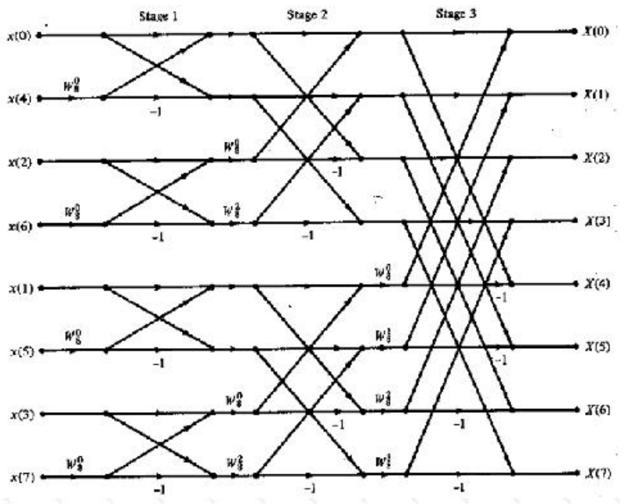


Fig 8.Eight-point decimation-in-time FFT algorithm

$$X(k) = \sum_{n=0}^{(N/2)-1} x(n)\,W_N^{kn} + \sum_{n=0}^{(N/2)-1} x(n)\,W_N^{kn}$$

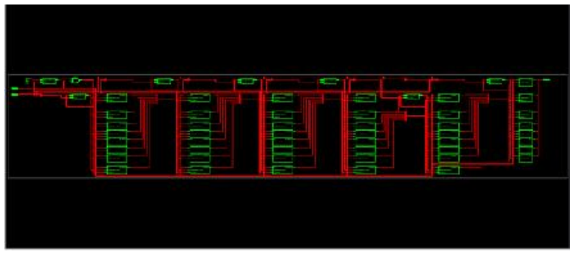
$$= \sum_{n=0}^{(N/2)-1} x(n)\,W_N^{kn} + W_N^{Nk/2} \sum_{n=0}^{(N/2)-1} x\left(n + \frac{N}{2}\right) W_N^{kn}$$

Since $W_N^{kN/2} = (-1)^k$

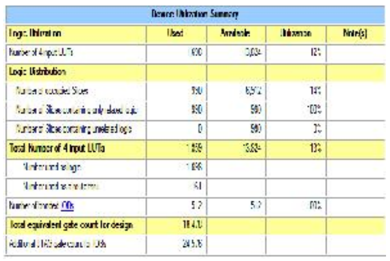
Now, let us split (decimate) X(k) into the even- and odd-numbered samples.

RTL SCHEMATIC:

RTL



Device utilization memory:



Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of 4 input LUTs	1,600	13,824	12%	
Logic Distribution				
Number of occupied Slices	990	6,912	14%	
Number of Slices containing only related logic	990	990	100%	
Number of Slices containing unrelated logic	0	990	0%	
Total Number of 4 input LUTs	1,859	13,824	13%	
Number used as logic	1,798			
Number used as route-thru	61			
Number of bonded IOBs	512	512	100%	
Total equivalent gate count for design	[illegible]			
Additional JTAG gate count for IOBs	24,576			

Conclusion

This brief shows how to design multiplier less unity-gain 8 Point SDF FFT architectures. The proposed architectures are not only multipierless and achieve Less area and Delay, but also require the smallest number of adders among current SDF FFTs. The proposed architectures achieve good figures of merit in terms of clock frequency, area. The 16 Point FFT implement by Verilog HDL and Simulated in Modelsim 6.4 c and Synthesized in Xilinx 9.1 tool.

References

[1] S. He and M. Torkelson, “Design and implementation of a 1024-point pipeline FFT processor,” in Proc. IEEE Custom Integr. Circuits Conf., May 1998, pp. 131–134.

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

[3] D. Cohen, “Simplified control of FFT hardware,” IEEE Trans. Acoust., Speech, Signal Process., vol. 24, no. 6, pp. 577–579, Dec. 1976.

[4] Y. Ma and L. Wanhammar, “A hardware efficient control of memory addressing for high-performance FFT processors,” IEEE Trans. Signal Process., vol. 48, no. 3, pp. 917–921, Mar. 2000.

[5] Z.-G. Ma, X.-B. Yin, and F. Yu, “A novel memory-based FFT architecture for real-valued signals based on a radix-2 decimation-in-frequency algorithm,” IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 62, no. 9, pp. 876–880, Sep. 2015.

[6] B. G. Jo and M. H. Sunwoo, “New continuous-flow mixed radix (CFMR) FFT processor using novel in-place strategy,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 52, no. 5, pp. 911–919, May 2005.

[7] X. Xiao, E. Oruklu, and J. Saniie, “Fast memory addressing scheme for radix-4 FFT implementation,” in Proc. IEEE Int. Conf. Electro/Inf.Technol., Jun. 2009, pp. 437–440.

[8] P.-Y. Tsai and C.-Y. Lin, “A generalized conflict-free memory addressing scheme for continuous-flow parallel-processing FFT processors with rescheduling,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 19, no. 12, pp. 2290–2302, Dec. 2011.

[9] S.-J. Huang and S.-G. Chen, “A high-throughput radix-16 FFT processor with parallel and normal input/output ordering for IEEE 802.15.3c systems,”IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 59, no. 8,pp. 1752–1765, Aug. 2012.

Access this Article in Online	
	Website: www.ijarm.com
Quick Response Code	Subject: Architecture
DOI:10.22192/ijamr.2018.05.06.009	

How to cite this article:
A.V. Prathapkumar, S. Karthikeyan, K. Yoganand. (2018). Implementation Of Memory Based FFT Architecture Based Processor. Int. J. Adv. Multidiscip. Res. 5(6): 64-68.
DOI: http://dx.doi.org/10.22192/ijamr.2018.05.06.009