EasyChair Preprint

№ 9426

Memory Based FFT Implementation on Re-Configurable Hardware

Rama Krishna Thirumuru, Vani Sai Kumar Komarraju,
Sai Madhava Yellamraju and Varshith Rao Gundavaram

December 7, 2022

Memory based FFT Implementation on Re-configurable Hardware

Rama Krishna Thirumuru
Professor at Electronics & Communication Department
KL University
Vijayawada, India
ramakrishnathirumuru@gmail.com

Vani Sai Kumar Komarraju
Electronics & Communication Engineering
KL University
Vijayawada, India
k.vanisaikumar@gmail.com

Sai Madhava Yellamraju
Electronics & Communication Engineering
KL University
Vijayawada, India
saimadhava17@gmail.com

Varshith Rao Gundavaram
Electronics & Communication Engineering
KL University
Vijayawada, India
gundavaramvarshith1333@gmail.com

***Abstract*—In Digital Signal Processing irrespective of Application Fast Fourier Transform (FFT) is a critical processing method while operating on discrete-time signals. The traditional way of computing FFT using a software approach by DSP processor will execute serially which restricts execution speed. The FFT implementation with optimized functionalities in parallel processing using the latest FPGA hardware is discussed in this work. In this paper, a memory-based FFT implementation on reconfigurable hardware that follows a conflict-free strategy with predefined memory size and a few other combinational components is proposed. Based on the experimental results obtained through simulations targeting ZTEX field programmable gate array (FPGA) Using Xilinx ISE, we conclude that the algorithm developed is faster than conventional approaches, with a 7.556ns delay and power consumption of 12.68 mV.**

***Index Terms*—Digital signal processing Re-configurable hardware, ZTEX FPGA, Xilinx ISE,Discrete Fourier Transform (DFT),Fast Fourier Transform (FFT), Radix-4.**

I. Introduction

One of the most widely utilized mathematical operations is the Fast Fourier Transform(FFT). It is an essential algorithm in DSP Applications for computing Discrete Fourier Transfrom(DFT). It is part of numerous systems in a large variety of applications.It is used in medical, engineering and communication, and other fields because it transitions quickly from the T-domain to the F-domain and vice versa.[1] Pipelined FFTs are employed in a system that requires very high rate computation of FFT. When systems' performance requirements are less stringent. Instead, there are requirements in terms of the amount of space or hardware resources used by the architecture. Under these conditions, developers typically use memory-based FFTs, also known as In-place or Iterative FFTs.[3] In Memory FFT the data is stored in a memory or bank of memories.This data is retrieved from memory, processed by butterflies and rotators, and then put in memory again. This method is repeated until all stages of the FFT algorithm have been computed. The main advantage of memory based FFT's is it will be reusing previous iteration data for computing current stage iterations. There are numerous architectures in this literature, and mainly, we perform this process using butterflies and rotators. As butterflies and rotators are reused in multiple stages of the FFT, memory-based FFTs have a lower number of butterflies and rotators.[2] There is a separate access strategy to memory for providing free access and may also demand extra multiplexers, buffer, or cache memory. The radix-4 memory has many advantages compared to previous experiences, it uses fewer Memory N samples and some multiplexers.[6-8] Further, this has been implemented FPGA. This implementation will allow the integration of the hardware with distributed logic.

This brief is organized as follows. Section II explains the proposed memory-based FFT. Section III explains the implementation using FPGA . Section IV presents a discussion on Results. Section V describes about an Application. Section VI describes Conclusion. Finally references used for this work.

II. Proposed Idea

A. Architecture

The basic architecture uses radix-4 and computes the FFT algorithm with six iterations. The number of iterations depends on

$$It = \frac{\log_2 N}{\log_2 r} = \frac{n}{\log_2 r} \quad (1)$$

The proposed architecture has four memories elements, each having N/4 samples in parallel fashion rather than a single memory of n samples. Using this design will allow read and writing the memory simultaneously and reduce the latency, increasing the throughput of the circuit.

B. Conflict free access

Four memories are retrieved parallelly, And PE is processed in every clock cycle that comes from different memories. The samples of these memories are stored in a natural order.

Samples are 0 to N/4-1 are stored in MEMO, N/4 to N/2-1 in MEM1. The samples are from 0 to N-1 with an index I=b_{n-1},b_{n-2} ... b_0, the memory of each sample I is

$$P1 = b_n - 3, b_n - 2.....b_0 \mid b_n - 1, b_n - 2. \quad (2)$$

Bits b_{n-1} and b_{n-2} will indicate four memories in sample and store, whereas bits b_{n-3} to b_0 are address.

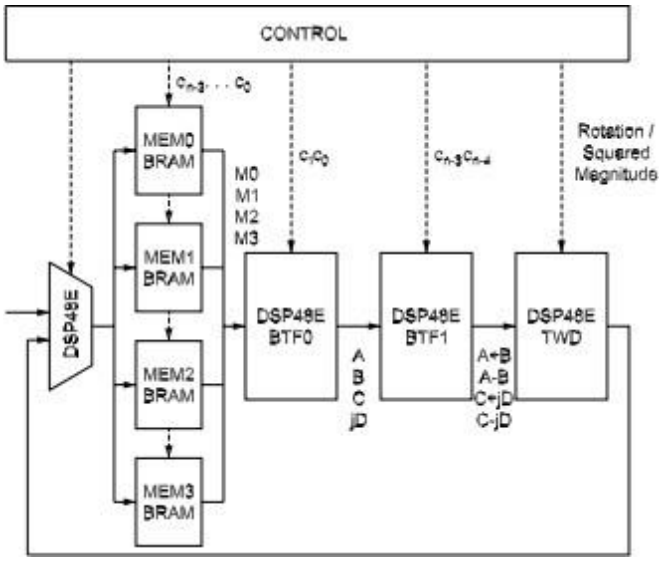

Fig. 1. Proposed Architecture

C. Radix-4 Algorithm

The Radix-4 is an additional fast Fourier Transform Algorithm (FFT) that can be obtained by moving the base from 2 to 4. The power/index diminishes in direct proportion to the size of the base. There are 50 fewer stages in radix-4 than in radix-2 since N=4M, indicating that stages have decreased by 50. It is explained in more detail in the later sections on how radix-4 simplifies complex calculations.

D. Functioning of radix-4 algorithm

For computing sequences, the radix-4 algorithm is comparable to the radix-2 technique in terms of type and speed management.The given sequence divides into four parts based on 'n'.[15] The given sequence layout in radix-4 is as follows:
n = [0, 4, 8, N - 4] results x(4n),
n = [1, 5, 9,..... N - 3] results x(4n+1),
n = [2, 6, 10,..... N - 2] results x(4n+2), and
n = [3, 7, 11,..... N - 1] results x(4n+3).[16-18]
After the division of N-point DFT, it can be computed as the sum of the outputs of 4 N/4-point DFTs, and these subsequences are interconnected with so-called twiddle factors.

$$W^{Ik}_N = e^{-(j2I\pi k/N)}I = 0, 1, 2, 3, \quad (3)$$

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

$$= \sum_{n=0}^{\frac{N}{4}-1} x(n)W_N^{nk} + \sum_{n=\frac{N}{4}}^{\frac{N}{2}-1} x(n)W_N^{nk} + \sum_{n=\frac{N}{2}}^{\frac{3N}{4}-1} x(n)W_N^{nk} + \sum_{n=\frac{3N}{4}}^{N-1} x(n)W_N^{nk} \quad (4)$$

The above equations are the results of a process called decimation in time. This process is performed because the samples of time are arranged into groups. The basic operation of the R4 butterfly is shown in Fig.2 Thus,

$$X(k) = \sum_{n=0}^{\frac{N}{4}-1} x(n)W^{nk}_N + W^{Nk/4}_N \sum_{n=n_1}^{\frac{N}{2}-1} x(n+\frac{n}{4})W_4^{nk} + W_N^{Nk/2} \sum_{n=n_2}^{\frac{3N}{4}-1} x(n+\frac{n}{4})W^{nk} + W^{3Nk/4}_N \sum_{n=\frac{3N}{4}-1}^{N-1} x(n)W^{nk}_N \quad (5)$$

Therefore,

$$X(k) = \sum_{n=0}^{\frac{n}{4}-1} [x(n)+(-j)^k x(n+\frac{N}{4})+(-1)^k x(n+\frac{N}{2}) + (j)^k x(n+\frac{3N}{4})]W^{nk}_N \quad (6)$$

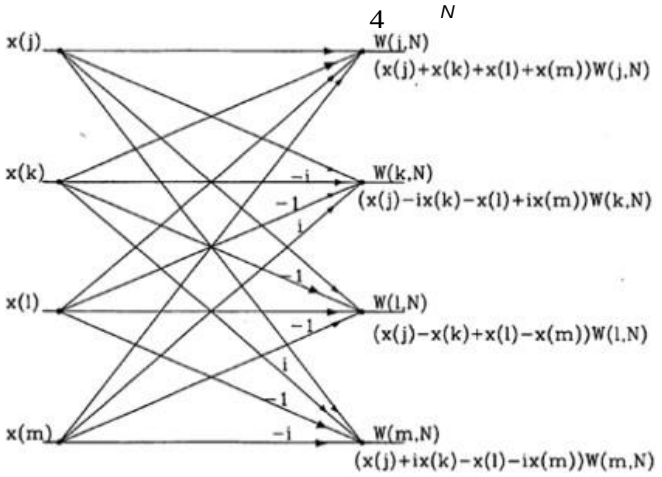

Fig. 2. Nodes of Radix-4 butterfly Algorithm

III. Experimental Setup

This concept has been introduced to ZTEX FPGA System. These VSX Systems include DSP48E arithmetic logic units and small elements of Distributed logic. Proposed Architecture makes the most of DSP48E and diminishes the use of distributed logic. But memories in this architecture make use of distributed logic. The perk of utilizing DSP Slices is that they can be clocked at high frequencies. And it allows large word lengths without disturbing the clock frequencies. But in Distributed logic clock frequency may slow down when improving word length.
The memories named MEM0, MEM1, MEM2, MEM3 are implemented by Block RAM(BRAM) memories. All memories elements contain 1024 addresses, and each address can store 24 + 24 bits of real and imaginary parts. Multiplexers would represent a remarkable cost if developed using Distributed logic. Connections among memory and PEs should be static i.e., outputs of memories should always be connected to PE without multiplexing.

BTF0 Module has two DSP48E in which multiplexers functionality is paralyzed and takes four input signals of 24 + 24 bits. The ALU unit of DSP48E is initialized by mode SIMD=TWO24. Similarly, BTF1 Consists of two DSP48E, which involves calculating the 2nd part of Radix-4 Butterfly. The execution of operations depends on the bits C_{n3}, C_{n4} of Control Counter.

Outputs of the Module BTF0 as a Function of the Control Bits				
Control Bits	**Outputs**			
c_1c_0	**Output 0**	**Output 1**	**Output 2**	**Output 3**
0 0	A=M0+M2	B=M1+M3	C=M0-M2	D=M1-M3
0 1	A=M1+M3	B=M0+M2	C=M1-M3	D=M0-M2
1 0	A=M2+M0	B=M3+M1	C=M2-M0	D=M3-M1
1 1	A=M3+M1	B=M2+M0	C=M3-M1	D=M2-M0

Fig. 3. Functioning of BTF0 Module

The TWD Module plans the multiplications by Twiddle factors. These Twiddle factors are banked in ROM Memory, these banked factors are used in further iterations of FFT. In the First iteration, Coefficients are read sequentially. In Next Iterations LSBs of Control Counter are removed for determining the address. As BTF1 outputs are shuffled so as Twiddle factors. In the last Iteration, there is no necessary for rotation. For finding power at each output frequency the squared magnitude of complex values is carried out.

Outputs of the Module BTF1 as a Function of the Control Bits				
Control Bits	**Outputs**			
$c_{n-3}c_{n-4}$	**Output 0**	**Output 1**	**Output 2**	**Output 3**
0 0	A+B	A-B	C+jD	C-jD
0 1	A-B	A+B	C-jD	C+jD
1 0	C+jD	C-jD	A+B	A-B
1 1	C-jD	C+jD	A-B	A+B

Fig. 4. Functioning of BTF1 Module

IV. Results and Discussions

There is a trade-off between the number of resources available to the architecture and the processing time in memory-based FFTs, T_{PROC}.

Approach	Radix	Parallel Process *	Memory Size	Memory Banks	Iterations **	Cycles Per Iteration	TP ***(Cycles)	Observations
[3]	2	2	N	2	$Log_2 N$	N/2	$N(log_2N)/2$	-
[4]	2	2	N	4	$Log_2 N$	N/2	$N(log_2N)/2$	Barrel Shifter for RD/WR Address
[5]	2	4	N	4	$Log_2 N - 1$	N/4	$N(log_2N-1)/4+1$	For real-valued data
[6]	4|2	4	2N	4	$(Log_2 N)/2$	N/4	$N(log_2N)/8$	Continous Flow
[7]	4	4	N	4	$(Log_2 N)/2$	N/4	$N(log_2N)/8$	Large Amount of Multiplexers
[8]	$2\|2^2\|2^3$	2	2N	4	$(Log_2 N)/3$	N/4	$N(log_2N)/12$	Uses pipelined MDC FFT's as PE
[9]	16	16	N	16	$(Log_2 N)/4$	N/16	$N(log_2N)/64$	For high Throughput
[10]	r	r	2N(Φ+1)	2r(Φ+1)	Logr N	N/r	$N(log_2N)/r$	The Radix r is a power of 2
Proposed	4	4	N	4	$(Log_2 N)/2$	N/4	$N(log_2N)/8$	Implemeted in DSP48E Slices

Fig. 5. Comparison Of Access Strategies in Memory-Based FFT's

*Parallel Process- Number of data that are proposed in Parallel. Which means how many data are read in parallel from the memories in each clock cycle.

** Related to Radix- Bigger Butterflies calculate a larger part of FFT Flow graph at each iteration, reducing the number of iterations.

*** TP – Processing Time, this is the product of number of iterations and the number of cycles per iteration.

The radix influences the trade-off between architectural resources and processing time. The bigger the radix, the greater usage of the Processing Elements(PE). A large PE expands the area, but processing time T_{PROC} is reduced.In the below equation memory access time is denoted by T_{MEM}

$$T_{PROC} = \frac{N}{r} lt.T_{MEM} \quad (7)$$

Below figures shows the area and Delay analysis achieved through proposed Architecture

	Device Utilization Summary			
A	**Logic Utilization**	**Used**	**Available**	**Utilization**
R	Number of Slices	14348	4656	308%
E	Number of Slice Flip Flops	17191	9312	184%
A	Number of 4 Input LUTs	9846	9312	105%
	Number of bonded IOBs	160	232	68%
	Number of MULTI18X18SIOs	2	20	10%
	Number of GCLKs	1	24	4%

Fig. 6. Area Analysis

	Cell: in->out	fanout	Gate Delay	Net Delay	Logical Name
D	IBUF: I->O	128	1.106	1.251	cn3_IBUF
E	LUT3: I0->O	2	0.612	0.449	m2/w4<9>1
L	LUT3: I1->O	1	0.612	0.357	m2/s<9>1
A	OBUF: I->O		3.169		y3_9_OBUF
Y	**Total**		7.556ns	(5.499ns Logic, 2.057ns route)	
				(72.8% logic, 27.2% route)	

Fig. 7. Delay Analysis

V. Application Case

The proposed memory based FFT can be employed in the procees of Medical Image compression which provides images of the human body and its components for clinical application.Using Memory based FFT compression in compression algorithm can process the image quickly coupled with the transformed domain compression.

VI. Conclusion

In this article, the new FFT algorithm based on radix-4 algorithm for DSP Applications was proposed and simulated on a target device ZTEX. This method has been successful in reducing the FPGA inner resources and passed the simulation and verification tests with above results. Moreover, the intended FFT has been successfully deployed on an FPGA utilizing DSP slices. The recommended design makes use of less distributed logic than earlier FPGA findings while maintaining a same resources in the terms of DSP slices and BRAM units.

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. S ánchez, 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.
[10]D. Reisis and N. Vlassopoulos, “Conflict-free parallel memory accessing techniques for FFT architectures,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 55, no. 11, pp. 3438–3447, Dec. 2008.
[11]C.-F. Hsiao, Y. Chen, and C.-Y. Lee, “A generalized mixedradix algorithm for memory-based FFT processors,” IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 57, no. 1, pp. 26–30, Jan. 2010.
[12]M. Garrido, “Efficient hardware architectures for the computation of the FFT and other related signal processing algorithms in real time,” Ph.D. dissertation, Dept. Signals, Syst. Radiocommun., Tech. Univ. Madrid, Madrid, Spain, 2009.
[13]M. Garrido, J. Grajal, and O. Gustafsson, “Optimum circuits for bit reversal,” IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 58, no. 10, pp. 657–661, Oct. 2011.
[14]M. Garrido and J. Grajal, “Efficient memoryless CORDIC for FFT computation,” in Proc. IEEE Int. Conf. Acoust. Speech Signal Process., vol. 2. Apr. 2007, pp. II-113–II-116.
[15] Chang, C.K., Hung, C.P. and Chen, S.G., 2003, May. An efficient memory based FFT architecture. In Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS’03. (Vol. 2, pp. II-II). IEEE.
[16] Ma, Z.G., Yin, X.B. and Yu, F., 2015. A novel memory-based FFT architecture for real-valued signals based on a radix-2 decimation-in-frequency algorithm. IEEE Transactions on circuits and systems II: Express Briefs, 62(9), pp.876-880.
[17] Bruguera, J.D. and Lang, T., 1996. Implementation of the FFT butterfly with redundant arithmetic. IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, 43(10), pp.717-723.
[18] White, S., 1981. A simple FFT butterfly arithmetic unit. IEEE Transactions on Circuits and Systems, 28(4), pp.352-355.
[19] Heydt, G.T., Fjeld, P.S., Liu, C.C., Pierce, D., Tu, L. and Hensley, G., 1999. Applications of the windowed FFT to electric power quality assessment. IEEE Transactions on Power Delivery, 14(4), pp.1411-1416.
[20] Yu, C., Yen, M.H., Hsiung, P.A. and Chen, S.J., 2011. A low-power 64-point pipeline FFT/IFFT processor for OFDM applications. IEEE transactions on consumer electronics, 57(1), pp.40-40.
[21] Han, X., Chen, J. and Rahardja, S., 2019, September. A new twiddle factor merging method for low complexity and high speed FFT architecture. In 2019 IEEE International Circuits and Systems Symposium (ICSyS) (pp. 1-5). IEEE.