FFT Architectures: A Review

Shubhangi M. Joshi.
Sathyabhama University,
Chennai

ABSTRACT

Fast Fourier Transform (FFT) is one of the most efficient algorithm widely used in the field of modern digital signal processing to compute the Discrete Fourier Transform (DFT).FFT is used in everything from broadband to 3G and Digital TV to radio LAN's. Due to its intensive computational requirements, it occupies large area and consumes high power in hardware. Different efficient algorithms are developed to improve its architecture. This paper gives an overview of the work done of different FFT processor previously. The comparison of different architecture is also discussed.

Keywords

Fast Fourier Transform (FFT), FFT architectures

1. INTRODUCTION

FFT processors are involved in a wide range of applications today. Not only a savery important block in broadband systems, digital TV etc., but also in are as like radar, medical electronics, imaging and the SETI project(Search for Extra-terrestrial Intelligence).Many of these systems are real-time systems, which mean that the systems has to produce a result within a specified time.

The work load for FFT computations are high and a better approach than a general purpose processor is required, to fulfil the requirements at a reasonable cost. The major concerns for researchers are to meet real-time processing requirements and to reduce hardware complexity mainly with respect to are and power and to improve processing speed of processor.

The DFT Algorithm: A DFT transform that is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} \text{ , k} = 0,1,2,\dots,\text{N} - 1 \qquad \text{(Eq.1)}$$

$$W_N^{kn} = \mathrm{e}^{-\mathrm{j}2\pi\mathrm{nk/N}} \qquad \text{(Eq.2)}$$

These equations show that to compute all N values DFT requires N^2complex multiplications and N(N-1)complex additions Since the amount of computation and thus the computation time, is approximately proportional to N^2, it will cost a long computation time for large values of N. For this reason, It is very important to reduce the number of multiplications and additions. The algorithm is an efficient algorithm to compute the DFT, is called Fast Fourier Transform (FFT) algorithm.The FFT algorithm deals with these complexity problems by exploiting regularities in the DFT algorithm.

1.1 FFT Processor

The FFT structure of FFT processor contains a butterfly processing unit, a RAM and ROM unit for the storage of data, address generation unit and a sequential control unit. The main units of FFT processor are butterfly processing unit and address generation unit. The dualport RAM used to store input data and intermediate results and output. Twiddle factors are stored in ROM. The address generation unit generates the address for reading data for butterfly operations and also for storing the output data results in RAM. Sequential control unit generates the control signals for each module.[1]

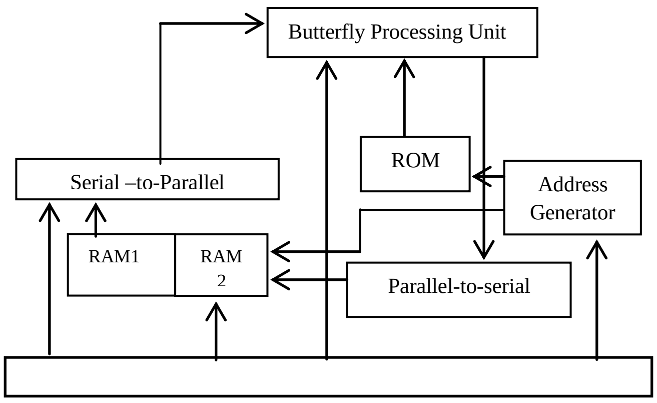


Fig.1 Block Diagram of FFT processor

2. FFT ARCHITECTURES:

Different FFT architectures are classified as

1. Memory Based
2. Cache Memory Based
3. Sequential
4. Parallel
5. Parallel Iterative
6. Array Architecture
7. Pipelined

2.1 Memory Based

Memory based- architectures mainly rely on the use of the memory for its operation. These Architectures generally consists of one or more processing elements (PE) or butterflies depending on computation, memory blocks and control unit.

Memory based architectures are classified into

- Single memory architecture
- Dual memory architecture

2.1.1 Single memory architecture

In this architecture processing element is connected to a single memory unit by bidirectional bus. Data exchanges are taken place between the processor and memory at every stage using this bus.

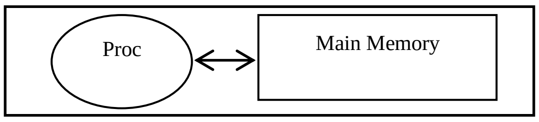


Fig.2 Single Memory Architecture

2.1.2 Dual memory architecture
In this type of architecture both memories are connected to processing element with two separate bidirectional data buses. Data inputs are passed from one memory to another memory through the processing elements (PE) and vice versa till the transform is completed.[2]

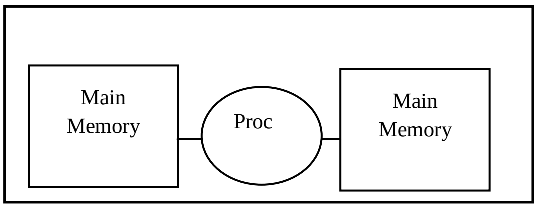


Fig.3 Dual Memory Architecture

2.2 Cache memory Architecture
This architecture is mainly used to increase the speed of the memory access, energy efficiency and for reducing the power consumption. Architecture is similar to that of single memory architecture except that the cache between the processer and main memory to pre fetch the data. This architecture is not widely used due to extra hardware and controller complexity.[2]

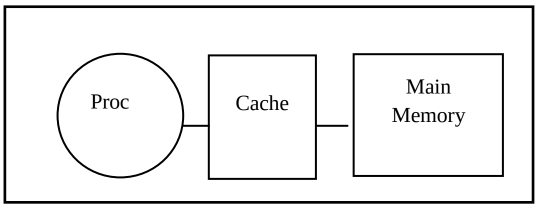
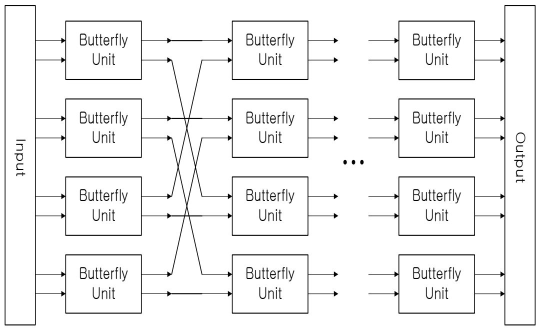


Fig.4 Cache Memory Architecture

2.3 Sequential Architecture
The basic sequential processer uses processing elements (PE) for computing butterfly. The same memory can be used to store input data, output data intermediate results and twiddle factors. The amount of hardware involved is very small and it requires N/2 $\log_2 N$ sequential operation to compute the FFT.

2.4 Parallel Architecture
This is also known as In- Place architecture. It consists of butterfly unit and three multiport buffers, one to parallelize the input data, one for processing data and one for the output. At the butterfly output a switching module branches the result to the right memory locations. The control of this type of architecture is complicated as there is lot of resource sharing. It is used for low to moderate speed applications. The feature of this architecture is high throughput but worst hardware efficiency.[17][18]



(b) Parallel Architecture

2.5 Parallel Iterative Architecture
Performance of FFT processor can be improved further by adding more processing elements in every sequential pipeline stage. Butterflies are computed in parallel in every stage. Total execution time requires is $\log_2 N$ cycles.

2.6 Array Architecture
A fully parallel structure can be obtained by having a PE for each of the butterfly operations. A number of processing elements with local buffers are interconnected in a network fashion to compute FFT. As the architecture requires huge area and a lot of hardware this is not the attractive option for large N.

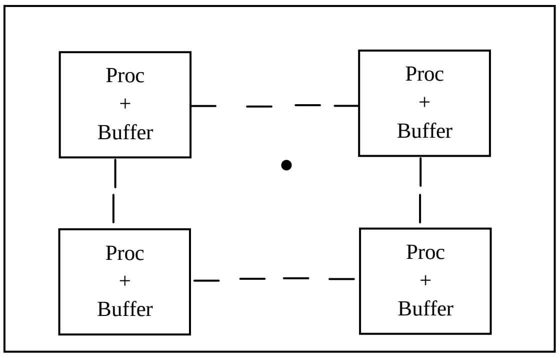


Fig.6: Array Architecture

2.7 Pipeline Architecture
This architecture is also known as cascaded FFT architecture, and used in most of the designs. the basic structure of pipelined is as shown in fig. between each stage of radix-r pe's there is a commutator and last stage is unscrambling stage. the commutator records the output data from previous stage and feed to the next stage. the unscramble rearranges data in natural sorted order.in figure a denotes the stage number in the pipeline. the number in boxes gives the size of that fifo r in complex sampling. c2 is a switch and radix-4 butterfly element.

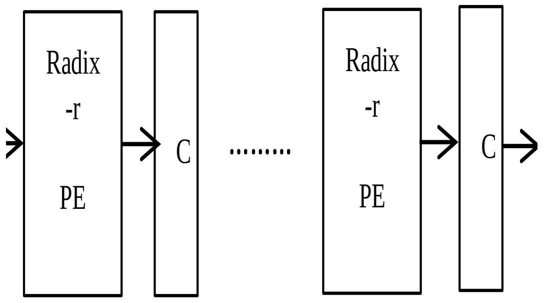


Fig.7:General structure of a pipelined FFT architecture.

Performance of this architecture can be improved by Parallelism using separate arithmetic unit for each stage of FFT processer and through put can be increased by factor $\log_2 N$ using different units in pipelined. Pipelined FFT processers have features like high throughput, simplicity, fast, small area and energy efficient implementation.

The most commonly used pipelined architectures such as Multipath Delay Commutator (MDC) , Single Path Delay Commutator (SDC) and Single Path Delay Feedback (SDF) [2][18]

2.7.1 Multipath Delay Commutator

In this architecture, input sequence is first divided into multiple parallel data streams by commutator. This data is then goes to butterfly unit for computation. Butterfly operation is then followed by twiddle factor multiplication with proper delay at each data streams. All butterflies and multiplier units are 100% utilised with proper input buffering.

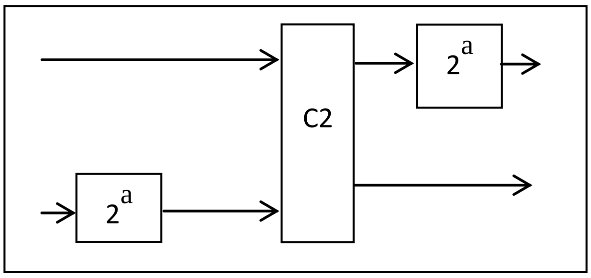


Fig.8: Multipath Delay Commutator structure.

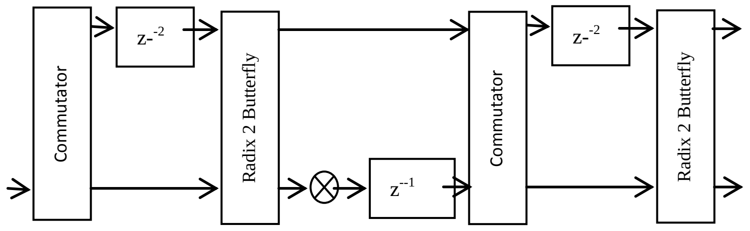


Fig.9: Radix-2 Multipath Delay Commutator structure (N=16).

2.7.2 Single Path Delay Commutator

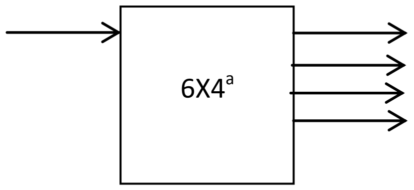


Fig.8: Single path Delay Commutator structure.

2.7.3 Single Path Delay Feedback

In this single data stream goes through multiplier in every stage. The commutator used for SDF is somewhat different because it also feeds data backwards. The delay units are more efficiently utilised by sharing the same storage between input and output of butterfly unit. Multiplier and Butterfly units can be utilised 50% because they are bypassed half the time.

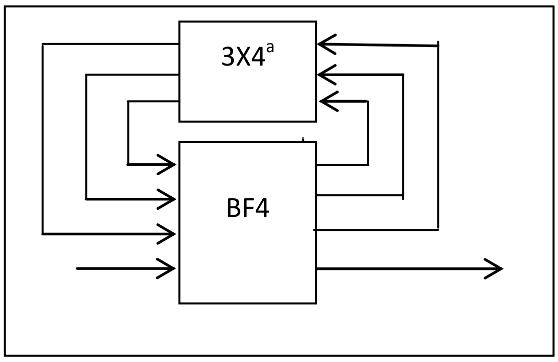


Fig.10:Single Path Delay Feedback Structure

3. RESULTS AND CONCLUSION

Among these various architectures, memory based architectures and Pipelined architectures are most widely used.

Table 1: Comparison of Pipelined FFT Architectures

Architecture	R2-SDF	R4-SDF	R2-MDC	R4-MDC
Delay Buffer	N-1	N-1	3N/2-2	5N/2-4
Complex Adder	$2\log_2N$	$8\log_4N$	$2\log_2N$	$8\log_4N$
Adder Utilization	50%	25%	100%	100%
Complex Multiplier	$\log_2N$-1	Log_4N-1	$\log_2N$-1	3Log_4N-1
Multiplier Utilization	50%	75%	100%	100%
Clock Rate	1	1	0.5	0.25
Control	Simple	Medium	Simple	Simple

Above comparison shows that in case of multipath delay commutator (MDC) two samples can be processed in parallel which improves the performance than designs which are serial in nature but requires larger memory.

Table2showsthe comparison between pipelined Single Delay feedback (SDF) architecture and memory based architecture for radix-r N point FFT implementation. The comparison is made in terms of Storage Requirement Memory banks, Complex multipliers and Complex adders. Power consumption can be reduced in the pipelined SDF architecture with the efficient implementations of sequential buffers whereas in memory based architecture, to achieve a conflict free memory access, random addressing is necessary. So, pipelined architectures are preferred when performance and power are the main concern than the complexity of hardware. On the other hand memory based architectures are good choice where complexity is of main concern.[2]

Further the performance can be improved by using high radix algorithm, higher parallel architectures or using folding technique.

Table 2: Comparison between Memory Based and Pipelined SDF FFT Architectures

Architecture →	Memory Based Architecture	Single Path delay feedback architecture
Algorithm	Radix-r	Radix-r
Storage Requirement	N	N-1
Memory banks (dual port)	r	Log_2N

Memory access times	$2N\log_r N$	$2N\log_2 N$
Complex multipliers	r-1	$\log_r N$-1
Complex adders	2r	$2\log_2 N$

4. REFERENCES

[1] Miss. Jaishri Katekhaye, Mr.Amit Lamba, Mr. Vipin "REVIEW ON FFT PROCESSOR FOR OFDM SYSTEM" IJAICT Vol-1,NOV:2014

[2] Anwar Bhasha Pattan, Dr. Madhavi Latha "FastFourier Transform Architectures: A Survey" "IJAECT

[3] Weidong Li and Lars Wanhammar "LOW- POWER FFT PROCESSORS" "IJAECT

[4] Weidong Li and Lars Wanhammar"VLSI based FFT processor with improvement in computation speed and area reduction" "IJECSE 2013.

[5] H. Sorensen, D. Jones, M. Heideman, and C. Burrus, "Real-valued fast Fourier transform algorithms," IEEE Trans. Acoust., Speech Signal Process., vol. 35, no. 6, pp. 849–863, Jun. 1987.

[6] S. He and M. Torkelson, "Design and implementation of a 1024-point pipeline FFT processor," in Proc. IEEE Custom Integr. Circuits Conf.,

[7] J. Lee, H. Lee, S. I. Cho, and S. S. Choi, "A high-speed two parallel radix-24 FFT/IFFT processor for MB-OFDM UWB systems," in Proc. IEEE Int. Symp. Circuits Syst., May 2006, pp. 4719–4722.

[8] [4] M. Ayinala, M. Brown, and K. K. Parhi, "Pipelined parallel FFT architectures via folding transformation," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 20, no. 6, pp. 1068–1081, Jun. 2012.

[9] R. Radhouane, P. Liu, and C. Modin, "Minimizing the memory requirement for continuous flow FFT implementation: Continuous flow mixed mode FFT (CFMM-FFT)," in Proc. IEEE Int. Symp. Circuits Syst., May 2000, pp. 116–119.

[10] 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.

[11] A. T. Jacobson, D. N. Truong, and B. M. Baas, "The design of a reconfigurable continuous-flow mixed-radix FFT processor," inProc. IEEE Int. Symp. Circuits Syst., May 2009, pp. 1133–1136.

[12] C. F. Hsiao, Y. Chen, and C. Y. Lee, "A generalized mixed-radix algorithm for memory-based FFT processors," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 57, no. 1, pp. 26–30, Jan. 2010.

[13] 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.

[14] 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.

[15] H. Chi and Z. Lai, "A cost-effective memory-based real-valued FFT and Hermitian symmetric IFFT processor for DMT-based wire-line transmission systems," in Proc. IEEE Int. Symp. Circuits Syst., May 2005, vol. 6,

[16] A. Wang and A. P. Chandrakasan, "Energy-aware architectures for a real-valued FFT implementation," in Proc. Int. Symp. Low Power Electron. Design, Aug. 2003, pp. 360–365.

[17] M. Garrido, K. K. Parhi, and J. Grajal, "A pipelined FFT architecture for real-valued signals," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 56, no. 12, pp. 2634–2643, Dec. 2009.

[18] ManoharAyinala, Yingjie Lao, and Keshab K. Parhi, "An In-Place FFT Architecture for Real-Valued Signals," IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS, VOL. 60, NO. 10, OCTOBER 2013.