Study of Radix-4 FFT Processor using CORDIC

Chetan Korde
M. E. Student
Department of Electronics and Telecommunication Engineering (VLSI & Embedded Systems)
D. Y. College of Engineering, Akurdi, Pune, India

P. Malathi, Ph.D
Professor
Department of Electronics and Telecommunication Engineering (VLSI & Embedded Systems)
D. Y. College of Engineering, Akurdi, Pune, India

ABSTRACT
This paper focus on the study of an efficient Radix-4 FFT and CORDIC algorithm. Due to use of Radix-4 speed gets doubled than Radix-2 in FFT computation. For twiddle factor calculation Co-ordinate Rotation Digital Computer (CORDIC) algorithm is used, which will reduce the computation time and make processor faster. The CORDIC offers the opportunity to calculate all the required functions in a rather simple and elegant fashion. In the next phase of this paper, actual Implementation of FFT processor on FPGA will be done using VHDL.

Keywords
FFT, Radix-4, Radix-2, CORDIC, VHDL, FPGA

1. INTRODUCTION
The digital signal processing has been dominated by microprocessors with enhancements such as special addressing modes and single cycle multiply-accumulate instructions. While these processors offer extreme flexibility with low cost, they are often not fast enough for most of Digital Signal Processing (DSP) tasks. The arrival of reconfigurable logic computers offers hardware solutions for higher speed at cost which are less than traditional software approach. Unfortunately, algorithms used for these microprocessor based systems cannot be mapped into hardware. In such hardware efficient algorithms, there is a class of solutions for trigonometric functions that uses shifts and additions in its operation. The trigonometric functions are found out using vector rotations, while other functions like square root are realized using an incremental expression. So, implementation of all these function in to hardware is difficult.

Also, Discrete Fourier Transform (DFT) is one of the core operations in digital signal processing and communication systems. Many fundamental algorithms can be realized by DFT, such as convolution, spectrum estimation, and correlation. Furthermore, DFT is widely used in standard embedded system applications such as wireless communication protocols requiring Orthogonal Frequency Division Multiplexing, Radar image processing using Synthetic Aperture Radar and Software Defined Radio etc. However, DFT is difficult to implement directly due to its computational complexity.

In practice, Fast Fourier transform (FFT) is used for reducing the complexity of computations. For FFT processors, butterfly operation is the most computationally demanding stage. Traditionally, a butterfly unit is composed of complex adders and multipliers and the multiplier is usually the speedup bottleneck in the pipeline of the FFT processor. The CORDIC algorithm is an alternative method to realize the butterfly operation without using any dedicated multiplier hardware. The trigonometric algorithm is also called as CORDIC. The CORDIC algorithm is versatile and hardware efficient since it requires only add and shift operations, making it suitable for the butterfly operations in FFT. This CORDIC algorithm is further used in Radix-4 FFT for faster computation. While converting samples taken in real time into equivalent frequency domain samples, FFT computation is used in which twiddle factor computation is done using CORDIC algorithm. Instead of storing actual twiddle factors in a ROM, the CORDIC-based FFT processor needs to store only the twiddle factor angles in a ROM for the butterfly operation.

Conventionally, a CORDIC-based FFT processor needs a dedicated memory bank to store the necessary twiddle factor angles for the rotation. This study proposes a modified CORDIC algorithm for FFT processors which eliminates the need for storing the twiddle factor angles. The algorithm generates the angles successively by an accumulator. With this approach, memory requirements of an FFT processor can be reduced by more than 20%. Memory reduction improves with the increased radix size. Furthermore, the angle generation circuit consumes less power consumption than angle memory accesses. Hence, system throughput does not change by using modified CORDIC angle calculation,

2. RADIX-4 FFT ALGORITHM
In this section the procedure of Radix-4 FFT algorithm is given.

Figure (1) shows an example of Radix-4 decimation in time method used for N=16 points FFT algorithm. As shown in FFT flow-graph inputs are in normal order while the outputs are in digit-reversed order. At input side the samples are taken from time domain which are processed with Radix-4 FFT and get equivalent components in frequency domain. The numbers over flow lines indicates the twiddle factor to be multiplied with the samples. Figure 2 (a) and (b) shows the basic butterfly structure of Radix-4 which have four inputs and four outputs, inputs are as x(n), x(n + n/4), x(n + n/2) and x(n + 3n/4) outputs are in digit reversed order X(k). FFTs in which N = r*k, and where the butterflies used in each stage are the same, are called Radix-r algorithms.

A Radix-r FFT uses N/r Radix-r butterflies for each stage and has logrN stages. That's why in this case for 16 point Radix-4 FFT requires k=2 stages. The signal flow graph of Radix-4 DIT butterfly operation is illustrated in figure 3.

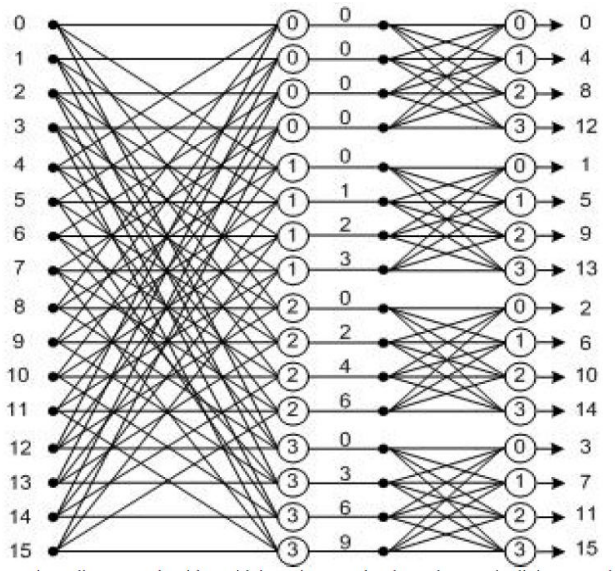

Figure 1: 16-point radix-4 FFT DIT algorithm

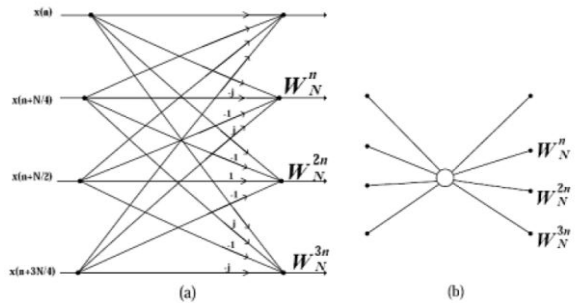

Figure 2: The basic butterfly for radix-4 FFT algorithm

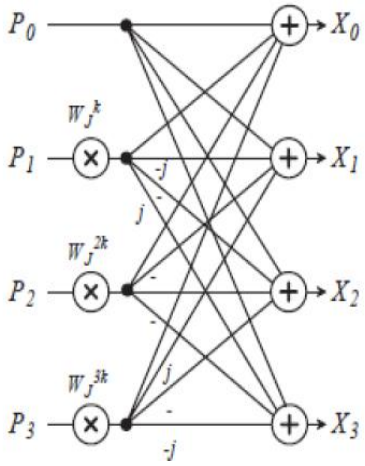

Figure 3: Butterfly structure to show operation

Radix-4 algorithms have a computational advantage over Radix-2 algorithms because one Radix-4 butterfly does the work of four Radix-2 butterflies, and the Radix-4 butterfly requires only three complex multipliers compared to four complex multipliers of four Radix-2 butterflies. The arithmetic kernel of Radix-4 DIT FFT is the buttery operation defined as

$$X0 = P_0 + W_1 P_1 + W_2 P_2 + W_3 P_3 \quad (1)$$

$$X1 = P_0 - jW_1 P_1 - W_2 P_2 + jW_3 P_3 \quad (2)$$

$$X2 = P_0 - W_1 P_1 + W_2 P_2 - W_3 P_3 \quad (3)$$

$$X3 = P_0 + jW_1 P_1 - W_2 P_2 - jW_3 P_3 \quad (4)$$

3. CORDIC

CORDIC algorithm was introduced in 1959 by Jack E. Volder for implementing a real-time navigation computer for aeronautical applications. The algorithm was initially formulated for computing the values of trigonometric functions. In early 1970s the CORDIC techniques were extended to exponential, logarithmic, forward and inverse circular and hyperbolic functions, ratios and square roots (Walther 1971). Its concepts have also been developed to include calculation of the Discrete Fourier Transform (Despain 1974). More recently (Bajard 1994) efficient hardware technique known as BKM for computing complex exponentials and trigonometric functions was proposed and has since been very widely applied.

The calculus courses provide with tools to compute the values of trigonometric functions, for example, via series expansions, polynomial, and rational function approximations. However, these implementations tend to require multiplication and division operations that make them expensive in hardware.

In contrast, CORDIC algorithms need only adders, shifters and comparators for computing a wide range of elementary functions. The method is especially efficient when fixed point implementations of signal processing algorithms on hardware are considered. For example, CORDIC is extremely popular in hardware accelerators and also in SIMD realizations. Furthermore, almost all function calculators employ CORDIC. Intel processors used CORDIC for trigonometric functions till 80486. CORDIC is a good choice for hardware solutions such as FPGA in which cost (gate count) minimization is more important than throughput maximization. In software implementations CORDIC enables most of the code and data be shared between routines for trigonometric and hyperbolic functions, helping to conserve memory. CORDIC algorithm is often used to implement rotations needed in modulators and demodulators.

4. DESIGN OF FFT PROCESSOR USING CORDIC ALGORITHM

As part of this project, the plan is to come up with the FPGA based hardware that uses radix-4 FFT processor using CORDIC algorithm. The design flow of FFT Processor using CORDIC is shown in figure 4. The selector block is nothing but a memory path buffer which computes respective memory of input samples. When Active signal is asserted and there are some input data, the address generator block assigns a memory position for each input sample. Now when Dual port Ram gets write Address signal from address generator block, it saves both memory path along with respective input samples. The 4 point FFT block has buttery unit within it. The overall structure of processor is CORDIC based FFT processor model is shown in Figure 5. The entire model is made of the address generation unit, the control unit, the dual port RAM unit, the 4-point butterfly unit and the CORDIC twiddle factor generation unit. This model is characterized by setting the parameter, sampling points and the accuracy to meet the actual needs. To perform these operations concurrently, a dual port RAM has been employed. The control unit involves the timing control of the data storage, reading and writing to make the corresponding data and rotating factor coefficient flow into butterfly and CORDIC computing unit in sequence in FFT operation. Data and

address of the 'twiddle factor' can be easily generated by the counter.

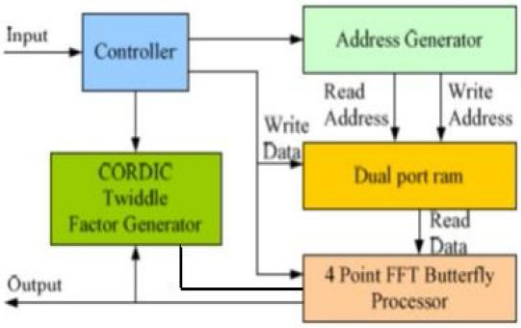

Figure 4: Proposed CORDIC -based FFT architecture

The address generation logic is very simple and does not limit the throughput of the system. When a start signal is asserted, at the same time, both to 4 Point FFT and Rotation factor generator block, the FFT block sends a Signal to CORDIC block for computing necessary twiddle factors consisting of sine-cosine terms.

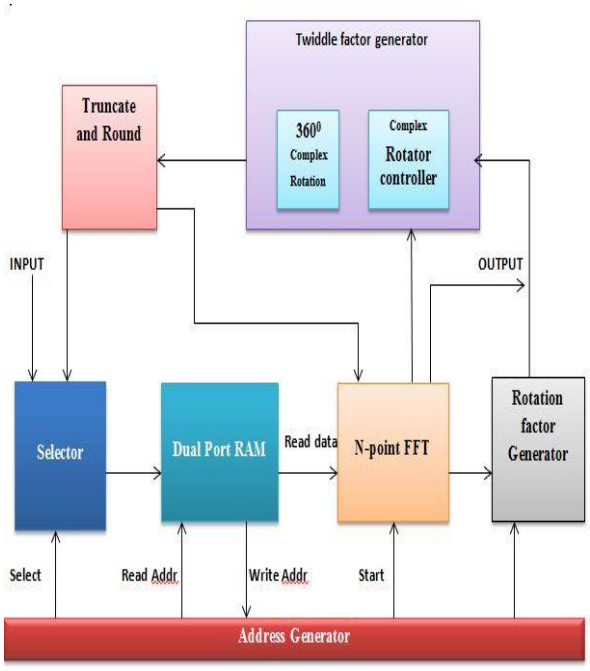

Figure 5: Block diagram of Radix-4 FFT Processor using CORDIC

This block is controlled by Rotation factor generator block. In truncate and round block, remapping of memory path and twiddle factors are held and fed back to FFT block. Now when address generator block sends read address signal to DRAM, it sends stored input data samples along with memory path in FFT block. Finally this twiddle factors are applied to the output of the butterflies, and a bit reverse scramble is done.

5. APPLICATION

1. Television terrestrial broadcasting systems.
2. Phase correlation system.
3. Mobile receiver.
4. Implementation of digital communication system.
5. Fault characterization and classification.
6. Radar.
7. Medical Imaging
   a. X-ray computed tomography (CT).
   b. Magnetic Resonance Imaging (MRI).

6. CONCLUSION

In this paper Radix-4 FFT processor architecture is studied. For generation of twiddle factor CORDIC algorithm is used. Various parts of FFT architecture such as Butterfly unit, Control Unit, Delay-Feedback model are discussed. In the next phase of this paper actual Implementation of FFT processor on FPGA will be done using VHDL. Proposed research work emphases on the use of techniques to reduce the computational complexity of processor design and the algorithm used which result into improvement of the design significantly.

7. REFERENCES

[1] Yasodai A. and Ramprasad A. V., "A new memory reduced Radix-4 CORDIC processor for FFT operation". IOSR Journal of VLSI and Signal Processing, Volume 2, Issue 5, May-Jun 2013.

[2] V. Charishma, K. Sreekanth Yadav and Neelima koppala. "Design and simulation of 64 Point FFT using Radix 4 algorithm for FPGA implementation." International Journal of Engineering Trends and Technology, Volume 4, Issue 2, 2013.

[3] M. Vidya M. Vijaya kumar and G. Sriramulu. "Design and VLSI implementation of a radix-4 64-point FFT processor". International Journal of Research in Computer and Communication technology, Volume-1, Issue-7.

[4] A. S. Padekar and S. S. Belsare. "Design of a CORDIC based radix-4 FFT processor." International Journal of Computer Science and Information Technologies, Volume-5 (4), 2014.

[5] Hsi-Chin, Hsin Tze-Yun Sung and Lu-Ting Ko. "Reconfigurable VLSI architecture for FFT processor". WSEAS TRANSACTIONS on CIRCUITS and SYSTEMS, Volume 8, Issue 6, June 2009.

[6] Javier Valls. "The use of CORDIC in software defined radios: A tutorial." Sep 2006.

[7] J. E. Volder. "The CORDIC trigonometric computing technique." IEEE Communications Magazine, 1959.

[8] Erdal Oruklu, Xin Xiao and Jafar Saniie. "Reduced memory and low power architectures for CORDIC-based FFT processors". Springer Science and Business Media, (66): pp 129-134, 16 April 2011.

[9] Ajay S. Padekar; Prof. S. S. Belsare, "Radix-4 FFT Architecture", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 5, May 2014.

[10] Wang Mingxing, Shi Jiangi, Tian Yinghui, Yang Zhe, "A Novel design of 1024-point pipelined FFT processor based on CORDIC algorithm", Intelligent System Design And engineering Application (ISDEA) 2012 second International Conference on Digital Object Identifier.