Review Paper on Generic N-Point FFT Processor Implementation Techniques on FPGA

[1]Ms. Kiran Chide
M.TECH(VLSI)
GHRCE, Nagpur

[2]Dr. Laxman P.Thakare
Assistant Professor
Department in electronics, GHRCE, Nagpur

Abstract: - The objectives of the project is to come up with the FPGA based Radix-4 FFT processor using CORDIC. As FFT take values in time domain and generates equivalent sample values in frequency domain, values of samples are taken as an input to the design, which is in time domain and using Decimation in Time (DIT) method for FFT computation, output is generated in terms of frequency samples. Due to in-place computation memory requirements have to reduce and using CORDIC for complex twiddle factor generation and multipli- cation the speed of the computation gets improved with less complexity. So, such kind of processor design using Radix-4 FFT instead of using Radix-2 FFT is main objective of this proposed work.

Keywords: - Fast Fourier transforms, FPGA, BIST, twiddle factor, RAM unit, and field programmable gate array

I. INTRODUCTION

FFT processor is an important building block of any digital processing circuit. This FFT is extensively used in applications, for instance, radar development, and Orthogonal Frequency Division Multiplexing (OFDM) [1]. The aim here is to reduce complexity of hardware and improve performance of processing, this project work proposes design of FFT processor using CORDIC algorithm which will be synthesized on Field Programmable Gate Array. The purpose of this project is to obtain an area efficient description of an FFT processor. To achieve this, Radix-4 FFT processor will develop using CORDIC algorithm. The N-POINT FFT Processor will be be able to calculate even higher orders FFT. The Processor Architecture IP is generic which makes it flexible to be integrated with any DSP Applications.

II. CORDIC ARCHITECTURE

CORDIC works by rotating the coordinate system through constant angles until the angle is reduces to zero. The angle offsets are selected such that the operations on X and Y are only shifts and adds. In this section, consider computation of vector rotation (rotation means transforming a vector (Xi, Yi) into a new vector (Xj, Yj)) illustrated in Figure 5.4, which maps vector (Xi, Yi) to (Xj, Yj) according to the equations

$$Xj = (Xi \cos \theta - Yi \sin \theta)$$

$$Y j = (Yi \cos \theta + Xi \sin \theta) \quad \ldots\ldots(6)$$

Where θ is a rotation angle. Note that, four multiplications and two additions are

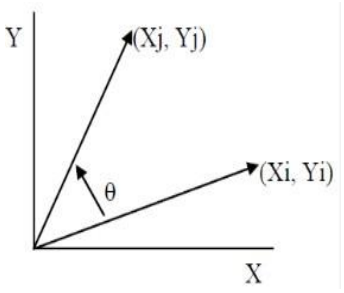

Figure 5.4: Rotate vector (Xi, Yi) to (Xj, Yj) by θ degree

needed to compute Xj and Yj provided that the values of cos θ and sin θ are available. If this is not the case, consider computing them digitally by series expansion, which is obviously complex. With some manipulation, this can also be computed with three multiplications and three additions as follows

$$Xj = (\cos \theta - \sin \theta)Yi + \cos \theta(Xi - Yi)$$

Concatenation of rotations:

As a first step towards the CORDIC implementation, we note that if θ = θa + θb, then first map (Xi, Yi) to (Xk, Yk) using the angle θa and then map (Xk, Yk) to (Xj, Yj) using the angle θb. So, it is possible to concatenate mappings for angles θn, where (n = 0, 1, 2, ..., ∞) in order to evaluate the mapping for

$$\theta = \sum_{n=0} \theta n \quad \ldots\ldots(8)$$

In the following, for n rotations are denoted with (Xn, Yn) and (Xn+1, Yn+1) the input and output to the rotation by :

$$Xn+1 = (Xn \cos \theta n - Yn \sin \theta n)$$

$$Yn+1 = (Yn \cos \theta n + Xn \sin \theta n) \quad \ldots\ldots(9)$$

Applying arithmetic shift and addition:

To proceed, let us assume that $-\pi/2 < \theta n < \pi/2$ Using $\tan \theta = \sin \theta / \cos \theta$, equation (9) can be rewritten as

$$Xn+1 = \cos \theta n(Xn - Yn \tan \theta n)$$

$$Yn+1 = \cos \theta n(Yn + Xn \tan \theta n) \quad \ldots\ldots(10)$$

this suggests the computational structure shown in Figure Note that $\cos \theta n = \cos(-\theta n)$ and $\tan \theta n = \tan(-\theta n)$, so the mapping for a

negative angle θn is the same as for θn except the change of signs in the terms involving the tangent. Multiplication by a power of two corresponds to the arithmetic shift operation, which is cheap to implement. The main idea of the CORDIC algorithm is that multiplication by tan θn can be based on shifting, when

tan θn = ±2−n, n ∈ 1, 2, 3, ...(11)

Under this condition, (10) becomes

Xn+1 = cos θn(Xn − dn ∗ Yn ∗ 2−n)

Yn+1 = cos θn(Yn + dn ∗ Xn ∗ 2−n)…….(12)

Where,

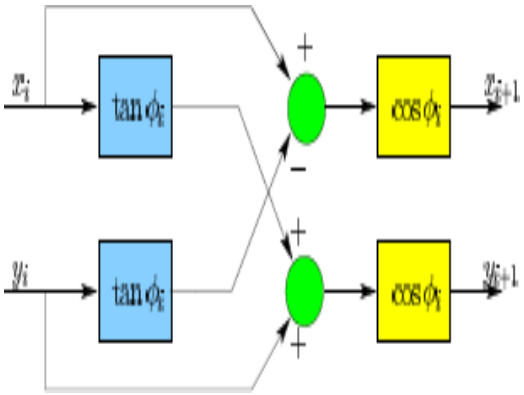

Figure 5.5: Organizing computations of the transform. For some specific angles θn, multiplication by tanθn can be replaced by an arithmetic shift and some sign manipulation.

dn = +1……fθn > 0

and

dn = -1……fθn < 0

Thus, substituting dn = (-1) for (+1) corresponds to swapping of signs of the second terms within parentheses, that is, subtraction becomes addition and vice versa.

Gain compensation:

However, (12) contains still multiplications by cos θn and if several rotations were con- catenated, then there have lots of multiplications. To solve the problem,first notice that

θn = arctan(2−n), n ∈ 1, 2, 3, ...(13)

and

$$\cos\theta_n = \cos(\arctan(2^{-n}) = \sqrt{\frac{1}{1+2^{-2n}}}$$

Because

$$\cos x = \pm\sqrt{\frac{1}{1+\tan^2 x}}$$

Then we divide both sides of (12) by

$$\cos\theta_n = \sqrt{\frac{1}{1+2^{-2n}}}$$

Which gives

$$X_{n+1} * a_n = (X_n - d_n * Y_n * 2^{-n})$$

$$Y_{n+1} * a_n = (Y_n + d_n * X_n * 2^{-n}) \quad (17)$$

$$a_n = \sqrt{\frac{1}{1+2^{-2n}}} \quad \text{……..(18)}$$

Now, the right hand sides contain just the the shift-and-addition parts of computation, and get Xn+1 and Yn+1 amplified by the gain an To see, how to compensate for the gain, let us multiply by a constant Ai both sides in (17), and let An+1 = anAn. As a result, get recursive equations

Xn+1 ∗ An+1 = (Xn ∗ An − dn ∗ YnAn ∗ 2−n)

Yn+1 ∗ An+1 = (Yn ∗ An + dn ∗ Xn ∗ An ∗ 2−n) (19)

These equations give the output of a chain of blocks, where just the shift-add parts of the rotations are computed, and multiplications by cos θn are neglected. After N such steps, we must multiply the results by 1/AN to get XN and YN . The value of the gain

AN can be calculated using

$$A_N = \prod_{n=0}^{N-1} \sqrt{\frac{1}{1+2^{2-n}}} \quad \text{……..(20)}$$

During iterations it would be wasteful and complicated to take the scale factors into account in computations because the value of AN does not depend on . Instead, they are usually pre-calculated and taken into account afterwards. In the table 5.1, see the gain values tabulated till iteration 10. Notice, that after the 4th iteration the

gain has converged for most applications and is approximately 1.6468 for N ≥ 9.

However, usually do not need to care about the gain as the scale factor can simply

Table 5.1: The gain values tabulated till iteration 9.

Iteration n	Gain AN	Iteration n	Gain AN
0	1.4142	5	1.6465
1	1.5811	6	1.6467
2	1.6298	7	1.6467
3	1.6425	8	1.6468
4	1.6457	9	1.6468

be considered as one of the components contributing to the gain of the whole system. Then, it can be taken into account, for example, in designing the gains of possible digital filters of the application. If gain compensation is necessary, it can be easily implemented, for example, by implementing the needed multiplication by using shift- add type fixed point arithmetics. Determining rotation directions: The angle θ of a composite rotation is uniquely defined by the sequence of elementary rotation directions, (d0, d1, dN−1). To determine this sequence on-the-fly at run-time, need an angle accumulator, that accumulates the elementary rotation angles, and tells in which direction the next rotation should be performed to reduce the angular error. The logic is based on the difference equation

$$Zn+1 = (Zn - dn * \arctan(2-n))$$

Where, Zn(n = 1, 1, ...) denotes the remaining rotation before performing the rotation by θn (Z0 = θ). The decision rule is

$$dn = -1 \text{.....(if : } Zn < 0)$$

$$dn = +1 \text{.....otherwise}$$

Every iteration improves the precision of CORDIC by one bit. Eight iterations result in the precision of 8 bits. Depending on implementation, the angular values arctan(2−n) in (21) can be precomputed and stored into look-up table or one may use a hardwired solution. Note that if there are small number of fixed rotation angles, extra accu- mulation/decision component might not be needed as dn's can be precomputed and tabulated. In the table 5.2 shows pre-calculated the first six entries of an arctan(2−n) look-up table.

III. FFT PROCESSOR PIPELINED ARCHITECTURE

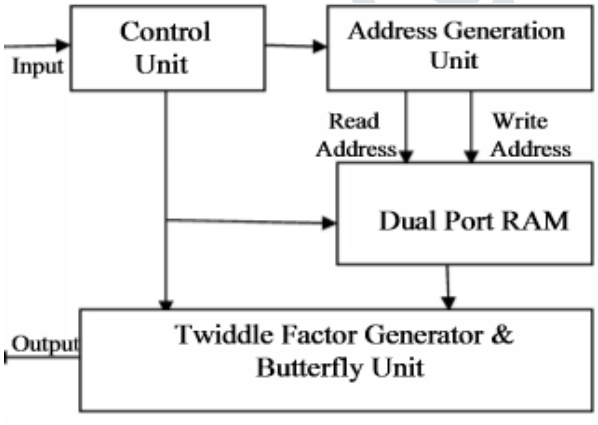

Fig.1. Block Diagram of FFT Architecture

The fundamental architecture of the FFT processor is spoken to in Fig. 1. It contains four essential units. Control unit, the part of the FFT processor.Butterfly unit (BU), which has three stage pipelined structure. Two dual port RAMs are used to store and process data. Besides this Address Generation Unit (AGU) is also present.

A. Control Unit

Control unit delivers all control signals for the whole structure, which is responsible for action and control of the processor.

B. Address Generation Unit

AGU is additionally very critical as contrasted and different units. It is utilized to make 8 read and 8 write addresses.

C. Butterfly Unit

For FFT calculation, the most imperative part is the BUthat figures the entirety and contrast of two info information, andplays an amazingly fundamental job in processing the result of thedifference and twiddle factors. In our plan the twiddlefactor generator is available inside the Butterfly unit.In our structure we utilize just 11 twiddle factor and the rest ofthe twiddle factors are created from these 11 twiddlefactors. These are: The engineering of BU is appeared in Fig. 2. In this TW_INprovides the 5 bit contribution for the choice of the best possible twiddlefactor from the multiplexers. In this two multiplexers are usedone is of 8: 1 and other 4: 1.

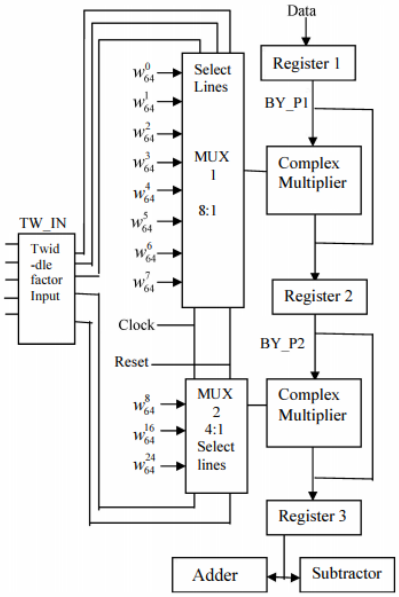

Fig.3. Block Diagram of Butterfly Unit

A. RAM Unit

In our structure 32 bit dual port RAMs are used which are RAMI and RAM2 independently

III. BUILT IN SELF TEST (BIST)

In this paper BIST is used for on chip testing of the FFT processor. BIST is a [mite state machine in which state change is control by the Test Mode (TM) input.
In BIST LFSR is a Linear Feedback Shift Register and MISR is a Multiple Input Signature Register.

IV. CONCLUSION

In this paper present the audit of Design of FPGA based superior 64bit FFT processor with BIST. To accomplish superior, pipelined structures have been utilized in the butterfly unit and the double port RAM. The butterfly unit itself produces the twiddle factor and performs complex increase because of which age of twiddle factor independently and getting to it not required. Because of this the

quantity of cycles required for complete task of FFT is decreased. The parallel-pipe lined FFT processor engineering can process input information at fast and the entire framework execution can be extraordinarily improved.

REFERENCE

[1] Shyue-Kung Lu, et al," Efficient Built-in Self-Test Techniques for Memory-Based FFT Processors"Proceedings of the 10th IEEE Pacific Rim International Symposium on Dependable Computing 2004 (PRDC'04).

[2] M Swetha et al,"Implementation of Restartable BTST Controller for Fault Detection in CLB of FPGA," International Journal of Scientific Engineering and Research, vol. I, pp 24-27, September 2013.

[3] H. T. Sorokin, et al"A Conflict-free parallel memory access scheme for FFT processors," international Symposium on Circuits and Systems, vol. 4, pp. 524-527, May 2003.

[4] Bingrui Wang, et al," Design of Pipelined FFT Processor Based on FPGA", Second International Conference on Computer Modeling and Simulation 2010.