International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

FFT with Minimum Hardware Utilization and Latency Using NEDA

Deepaa S S[1], Sheela Devi Aswathy Chandran[2]

[1]P G Scholar, VLSI and Embedded Systems, Department of ECE, T K M Institute of Technology, Kollam, India

[2]Assistant Professor, Department of ECE, T K M Institute of Technology, Kollam, India

Abstract: ***New Distributed Arithmetic (NEDA) technique is being used in many digital signal processing systems that require MAC (multiply and accumulate) units. FFT (Fast Fourier Transform) is a method for computing the DFT with reduced number of computations. Distributed arithmetic technique is used to implement the sum of product terms and this technique uses ROM, Adder and Shifter for the purpose of implementation, but in NEDA technique only Adder and Shifter is used. So, the size of the architecture is reduced with respect to Distributed arithmetic technique, and thus the speed and throughput of the architecture is enhanced. The advantages of this method are reduced hardware and improved latency. The advantage of using Radix-4 algorithm is that it retains the simplicity of Radix-2 algorithm and gives the output with lesser complexity. Design of FFT using NEDA improves performance of the system in terms of speed, power and area. The VHDL language is used for coding, synthesis can be done by means of Xilinx-ISE and Model-Sim can be used for simulation.***

Keywords: **CO**rdinate **R**otation **DI**gital **C**omputer (CORDIC), Distributed Arithmetic (DA), Discrete Fourier Transform (DFT), Fast Fourier Transform (FFT), Multiply and Accumulate Unit (MAC), New Distributed Arithmetic (NEDA), Radix-4.

1. Introduction

Today's electronic systems mostly run on batteries thus making the designs to be hardware efficient and power efficient. Application areas such as digital signal processing, communications, etc. employ digital systems which carryout complex functionalities. Hardware efficient and power efficient architectures for these systems are most required to achieve maximum performance.

The Fourier transform is the one of the several mathematical tools for analyzing the signals. It involves the decomposition of the signals in the frequency-domain in terms of sinusoidal or co sinusoidal components. The mathematical definition of a continuous Fourier Transform is given in the following.

$$X(\omega) = 1/\sqrt{2\pi} \int_{-\infty}^{\infty} x(t) e^{-j\omega t} dt \quad (1)$$

Where x(t) is the original signal, $X(\omega)$ is the representation of signal in the frequency-domain, j is the imaginary number, ω is the angular frequency and t is the time index.

Fourier transform is developed to represent the continuous signals in frequency domain. But, it is necessary to analyze the discrete signals. Because, all real time processors are made to be dealt with the digital signals alone. For analyzing the discrete signals, the Discrete Time Fourier Transform is used. The transformed value should be discrete. Since, the digital signal processors cannot work with the continuous frequency signals. DFT is developed to represent the discrete signal in discrete frequency domain.

DFT is also considered as one of the major tools to perform frequency analysis of discrete time signals. A discrete time sequence can be represented by samples of its spectrum in the frequency domain, using DFT. The mathematical representation of the transform is shown below.

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{\left(\frac{-j2\pi kn}{N}\right)} \quad n=0,1.....\ N-1 \quad (2)$$

Many efficient ways have been put up for direct implementation of DFT due to its computational complexity. FFT is one of the most efficient and common ways to implement DFT. Reduced computational complexity and low latency are two driving factors for implementing DFT using FFT. The forward and inverse equations of an N point FFT are given below

$$X(k) = \sum_{n=0}^{N-1} x[n] W_N^{kn} \quad (3)$$

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] W_N^{-kn} \quad (4)$$

As seen from equations (3) and (4), the basis of both forward and inverse equations remain same thus increasing the scope of the architecture to both forward and backward. Due to increased employability of FFT in modern electronic systems, higher radix FFTs such as radix – 4, radix – 8, radix – 2^k, split radix, etc. are designed for improved timing and reduced hardware. The basic difference of the mentioned methods lies in the structure of their butterfly units.

2. Theory

Many works has been carried out in the field of transforms like DFT and FFT, etc. DA [6] has become one of the most efficient tools in VLSI implementation of digital signal processing architectures. It efficiently computes inner products of vectors, which is a key requirement in many DSP systems. One of the key computational blocks in DSP is MAC, which is implemented by a standard adder unit and a multiplier. Using DA, MAC unit can be implemented by pre computing all possible products and using a ROM to store them. By using DA is in its exponential increase of the size of ROM with increase in internal precision and number of inputs. An approach to overcome this drawback is by distributing the coefficients to inputs. One of such examples is NEDA. As in [5], it can be used to implement any transform that is based on Fourier basis. This approach helps

in finding out the redundancy in computing vector inner product thus reducing the number of computational blocks, especially adder.

Many architecture models for DFT, based on DA, are implemented in [4]. The disadvantage of this implementation is that they use ROM or RAM which makes the designs with increased architecture. Other approach in implementing DFT is based on employing CORDIC units [2][3]. Even this has the structure of a CORDIC unit is comparable to that of a multiplier thus making it more hardware complex.

The approach is based on NEDA, which does not require any ROM thus making to have reduced hardware core. The distribution of the coefficients is done optimally to further reduce the redundant hardware units.

3. Methodology

New Distributed Arithmetic (NEDA) is being used in many digital signal processing systems that possess MAC units as their computational blocks. Transforms such as FFT, DCT, etc. have many MAC like structures that it requires much hardware. Design of such transforms using multiplier-less and ROM-less architectures using blocks like NEDA can improve the performance of the system in terms of power, area and speed.

3.1 System Overview

The basic block diagram for the 16 point FFT using NEDA is shown in the Figure 1.The design of 16-point FFT using radix-4 method has been designed. Complex multiplications required during the process have been designed by using NEDA. According to the radix-4 algorithm, to design 16-point FFT, eight radix-4 butterflies are required. Four radix-4 butterflies are used in the first stage and the other four being used in the second/final stage. The output of each radix-4 butterfly is multiplied by the respective twiddle factors. In the shown block diagram, the first stage consists of four radix-4 butterflies. The inputs to the butterflies are x(n), x(n+4), x(n+8), x(n+12) where n is 0 for first butterfly,1 for second butterfly, 2 for the third butterfly, and 3 for the last butterfly, all of first stage.

The twiddle factors are given by $W^0{}_{16}, W^q{}_{16}, W^{2q}{}_{16}, W^{3q}{}_{16}$ where q is 0 for first butterfly,1 for second butterfly, 2 for third butterfly, and 3 for the last butterfly. The outputs of first stage are multiplied with respective twiddle factors and are given as inputs to the second stage. As the work, the complex twiddle multiplications required at the stage-1 output have been fed into NEDA blocks. Overall 9 NEDA blocks are required at the output of first stage of the 16 point FFT processor.

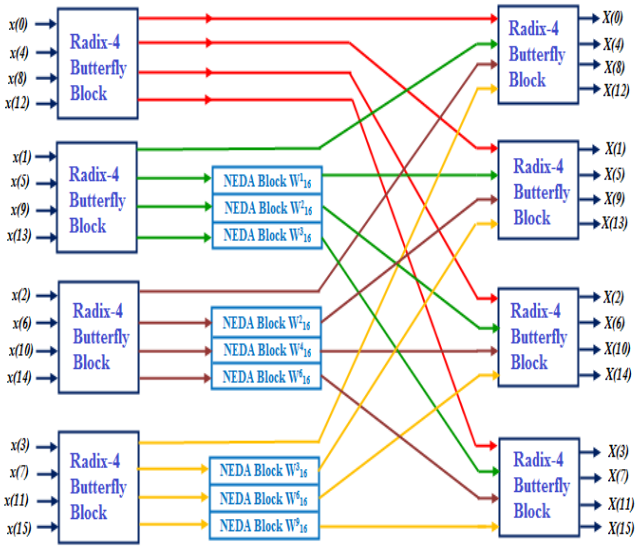

Figure 1: Block Diagram of 16-Point Radix-4 FFT using NEDA

In the second stage, 4 more radix-4 butterfly blocks are used. The first radix-4 butterfly in the second stage takes the first output of the 4 radix-4 butterfly blocks used in the first stage. The second radix-4 butterfly in the second stage takes the second output of the 4 radix-4 butterfly blocks followed by the NEDA block (if required). This process continues for the rest radix-4 butterfly blocks present in the second stage. There is no need of using any NEDA block after second stage as the twiddle factor $W^0{}_{16}$ that is 1 is multiplied to the outputs of the second stage. The advantage of using radix-4 algorithm is that it retains the simplicity of radix-2 algorithm and gives the output with lesser complexity. The NEDA block shown in the block diagram does the complex multiplication of the output of the first stage and the respective twiddle factor. The twiddle factor values used here are as follows,

$$W^1{}_{16} = \cos\frac{\pi}{8} - j\sin\frac{\pi}{8} = 0.9238\text{-}j0.3826$$
$$W^2{}_{16} = \cos\frac{2\pi}{8} - j\sin\frac{2\pi}{8} = 0.7071\text{-}j0.7071$$
$$W^3{}_{16} = \cos\frac{3\pi}{8} - j\sin\frac{3\pi}{8} = 0.3826\text{-}j0.9238$$
$$W^4{}_{16} = \cos\frac{4\pi}{8} - j\sin\frac{4\pi}{8} = 0\text{-}j$$
$$W^6{}_{16} = \cos\frac{6\pi}{8} - j\sin\frac{6\pi}{8} = -0.7071\text{-}j0.7071$$
$$W^9{}_{16} = \cos\frac{9\pi}{8} - j\sin\frac{9\pi}{8} = -0.9238\text{+}j0.3826 \quad (5)$$

The product of a complex number and a twiddle factor is given by $(R+jI)(\cos\theta + j\sin\theta) = (R\cos\theta - I\sin\theta) + j(R\sin\theta + I\cos\theta)$. For a constant θ, cosine and sine values are constant.

3.2 Radix-4 Butterfly Structure

In Radix-4 butterfly structure, N has been assumed a power of 4, i.e.$N = 4^L$. When $N = 16 = 4^2$, the given sequence x(n) is decimated into four sequences of length $\frac{N}{4}$. Radix-4 FFT algorithm is developed for evaluating the DFT for N = 16. The symmetry and periodicity of $W^r{}_N$ can be exploited to obtain further reductions in computation. The multiplications by $W^0{}_N = 1$, $W_N^{\frac{N}{2}} = -1$, $W_N^{\frac{N}{4}} = j$, $W_N^{\frac{3N}{2}} = -j$ can be avoided in the DFT computation process in order to save the computational

complexity. The basic structure of Radix-4 butterfly is shown in figure 2.

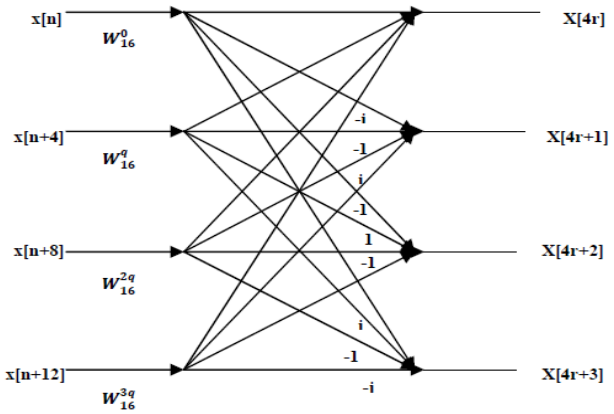

Figure 2: Basic Structure of Radix-4 Butterfly Block

3.3 NEDA Architecture

NEw Distributed Arithmetic (NEDA) is being used in many digital signal processing systems that possess MAC unit as the computational blocks. Transforms such as DWT, DCT, etc have many MAC like structures that in turn require much hardware. Design of such transform using multiplier-less and ROM-less architectures using blocks like NEDA improves performance of the system in terms of power, area and speed.

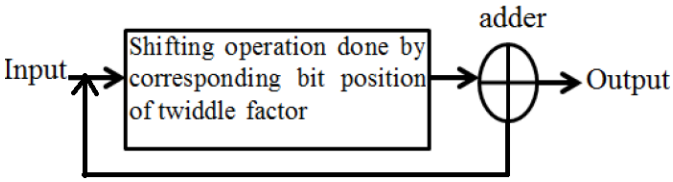

Figure 3: NEDA Architecture.

In figure 3, binary output of the radix-4 butterfly block is given as input to the NEDA block. In NEDA block, only addition and shifting operation is done. Input is shifted by corresponding bit position of ones in the twiddle factor. If the bit position of twiddle factor is 2^k, then the direction of shifting is determined by the sign of k, where k represents the bit position. After the shifting operation addition takes place.

If the input is a complex number, then the real value of input has shifting and addition operation with corresponding real and imaginary value of twiddle factor. The imaginary part of input has shifting and addition operation with corresponding real and imaginary value of twiddle factor. And combining the shift and addition of real and imaginary part the output is obtained.

In figure 1, twiddle factors are combined with the output of radix-4 butterfly block of first stage and the output of NEDA bock is again fed as input to the radix-4 butterfly block of second stage and corresponding output is obtained.

4. Results and Discussion

The Blocks are modeled using VHDL in Xilinx ISE Design Suite 13.2 and the simulation of the design is performed using ModelSim SE 6.3f to verify the functionality of the design. Here a structural model of proposed system is been developed. The Architecture contains modules such as Radix-4 Butterfly Block and NEDA Block.

4.1 Simulation of Radix-4 Butterfly Block

The butterfly of a radix-4 algorithm consists of four inputs and four outputs. The input signals are s0,s1,s2,s3 and output signals are g0,g1,g2,g3. Simulation result of Radix-4 butterfly block is shown in figure 4.

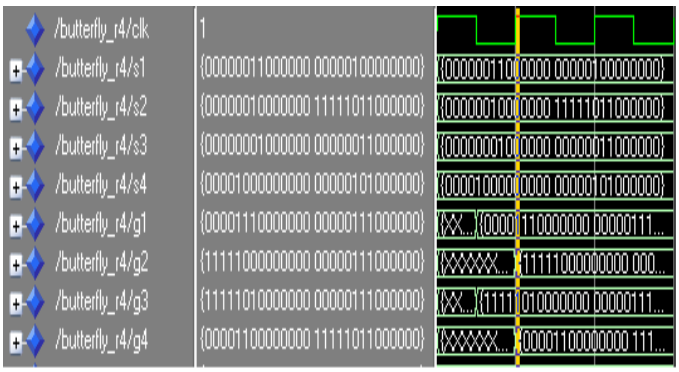

Figure 4: Simulation result of Radix-4 butterfly block

4.2 Simulation of NEDA Block

In NEDA block, each state corresponds to the values of the twiddle factor. But instead of multiplying the input with the corresponding value of the twiddle factor binary addition and shifting operation takes place. Simulation result of NEDA architecture is shown in figure 5.

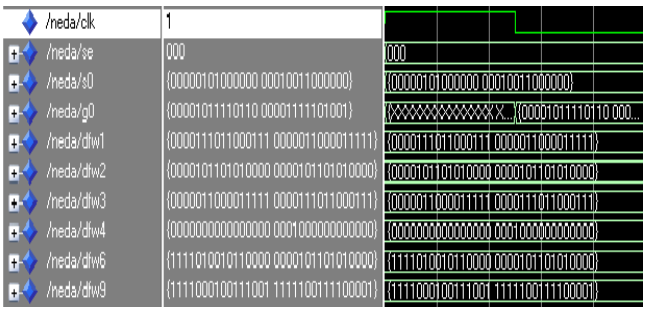

Figure 5: Simulation result of NEDA Architecture.

4.3 Simulation of 16-point Radix 4 FFT

The output from the Radix-4 butterfly block is obtained by binary addition of the four states. The first four output value from the Radix-4 butterfly block is given directly to second stage Radix-4 butterfly block since the twiddle factor is one. Meanwhile the other output values from the Radix-4 butterfly block is given to the NEDA block where addition and shifting operation between these output values and twiddle factor take place. These outputs from the NEDA block is given to the second stage Radix-4 butterfly block from which FFT values for the given DFT value is obtained. Simulation result of 16-Point Radix-4 FFT using NEDA is shown in figure 6.

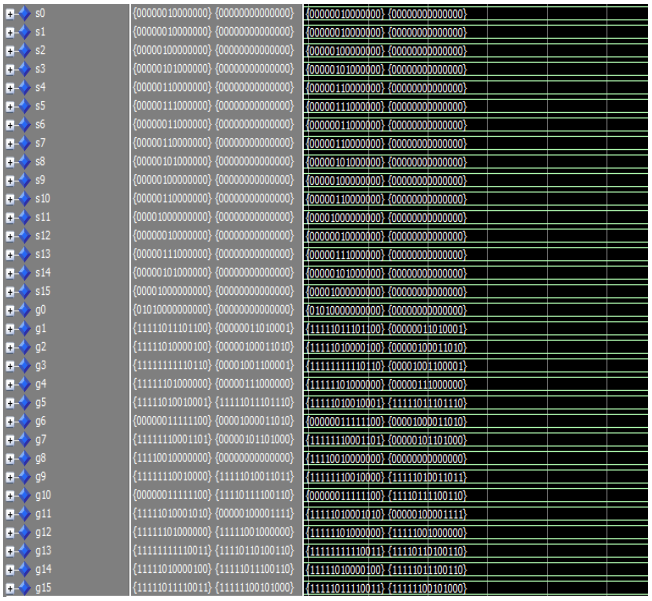

Figure 6: Simulation result of 16-Point Radix-4 FFT using NEDA

4.4 Synthesis Report

From the synthesis report of 16 point Radix-4 FFT, the architecture uses no multiplier and ROM thus making the system more reliable with minimum hardware utilization.

```
Number of errors:      0
Number of warnings:    0
Slice Logic Utilization:
  Number of Slice Registers:                1,089 out of 948,480
    Number used as Flip Flops:                  0
    Number used as Latches:                     0
    Number used as Latch-thrus:                 0
    Number used as AND/OR logics:           1,089
  Number of Slice LUTs:                     4,125 out of 474,240
    Number used as logic:                   4,113 out of 474,240
      Number using O6 output only:          2,515
      Number using O5 output only:             55
      Number using O5 and O6:               1,543
      Number used as ROM:                       0
    Number used as Memory:                      0 out of 132,480
    Number used exclusively as route-thrus:    12
      Number with same-slice register load:     0
      Number with same-slice carry load:       12
      Number with other load:                   0
```

Figure 7: Synthesis report of 16-point Radix-4 FFT using NEDA

5. Conclusion

Radix-4 complex 16 point FFT core using NEDA is designed. It is ROM less and multiplier less method. This design is efficient in terms of hardware and power consumption. Fast Fourier transform (FFT) is an efficient algorithm to compute the discrete Fourier transform and its inverse. The proposed design is simulated by using ModelSim and synthesized by Xilinx ISE. The synthesis results show that the computation for calculating the 16 point FFT is efficient in terms of area and power. Due to the extensive use of multipliers, memory or ROMs, there is a substantial increase in the power consumption and area for FFT cores. The use of FFT core reduces the number of computations in calculating a transform, thereby increasing speed and throughput. The present architecture of 16-point radix-4 complex FFT core using NEDA uses no multiplier and ROM thus making the system more reliable with minimum hardware utilization. The area and power can be further reduced by changing the adder structure in NEDA Block. Thus an efficient 16-point Radix-4 FFT can be designed by modifying the adder circuit in the architecture.

References

[1] Abhishek Mankar, Ansuman Diptisankar Das and N Prasad, "FPGA Implementation of 16-Point Radix-4 Complex FFT Core Using NEDA,"IEEE trans on Computing, Dec 2013

[2] Pooja Choudhary, and Dr. Abhijit Karmakar, "CORDIC Based Implementation of Fast Fourier Transform," Proc. Intl. Conf.Computer and Comm. Tech., Sept. 2011.

[3] Jayshankar, "Efficient Computation of the DFT of a 2N –Point Real Sequence using FFT with CORDIC based Butterflies," Proc. IEEE TENCON 2008,Nov. 2008

[4] M. Rawski, M. Vojtynski, T. Wojciechowski, and P.Majkowski, "Distributed Arithmetic Based Implementation of Fourier Transform Targeted at FPGA Architectures,"Proc. Intl. Conf. Mixed Design,, Jun. 2007. Wendi Pan, Ahmed Shams, and Magdy A. Bayoumi,"NEDA: A NEw Distributed Arithmetic Architecture and its Application to One Dimensional Discrete Cosine Transform," Proc. IEEE Workshop on Signal Processing Syst., Oct. 1999.

[5] Stanley A. White, "Applications of Distributed Arithmetic to Digital Signal Processing: A Tutorial Review," IEEE ASSP Magazine, vol. 6, no. 3, Jul.1989.