TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol. 12, No. 8, August 2014, pp. 6224 ~ 6229
DOI: 10.11591/telkomnika.v12i8.5224

Impact of FFT Algorithm Selection on Switching Activity and Coefficient Memory Size

Imran Ali Qureshi*[1], Fahad Qureshi[2]
[1]School of Information and Electronics, Beijing Institute of Technology, Beijing, P.R China
[2]Mehran University of Engineering and Technology, Jamshoro Pakistan
*Corresponding author, e-mail: i.ali1225@yahoo.com

Abstract

The binary tree decomposition allows for obtaining a large number of algorithms that can be used to calculate the fast Fourier transform. This paper analyzes the differences among these algorithms in terms of switching activity, which is related to the power consumption of the circuit, and size of the coefficient memories, which is related to the area of the circuit.Experimental results show the most efficient algorithms in term of area and power consumption. Furthermore, the paper shows the importance of a proper algorithm selection, since efficient algorithms can lead to savings of upto 45% in terms of the coefficient memory and even greater than 50% in terms of switching activity with respect to other less efficient ones.

Keywords*: switching activity, twiddle factor, binary tree, pipelined architecture, coefficient memory*

1. Introduction

The discrete Fourier transform (DFT) is one of the most important algorithms in the field of digital signal processing. It transforms a signal in the time domain into the frequency domain. The DFT is used in multiple applications, including spectrum analysis and orthogonal frequency division multiplexer (OFDM). There are various efficient methods for computing DFT however the Fast Fourier Transform (FFT) based on the Cooley-Tukey algorithm [1] is mostly used as it reduces the number of operations from $O(N^2)$ for the DFT to O(N log N) for the FFT. To efficiently implement the FFT algorithms, various hardware architectures have been proposed, such as in-place architectures [2], pipelined architectures [3-7]. Among these, the pipelined architecture are preferred, as this architecture provides high performance with an acceptable hardware cost. In addition, pipelined hardware architectures are highly regular, thereby automatically generating FFTs of various lengths.

A pipelined FFT architecture consists of a series of stages in which additions and twiddle factor multiplications are calculated [8, 9]. However, the distribution of the twiddle factors among the different stages of the FFT depends on the algorithm that is chosen [9, 10]. In [10], all the possible radix-2 FFT algorithms based on binary tree were presented. These FFT algorithms include all the different distributions of twiddle factor between the stages of the FFT.

A twiddle factor multiplication is actually a rotation, i.e, a multiplication by a complex number with a magnitude equal to one, so only the phase of the input data is affected. Different methods have been suggested in the literature for twiddle factor multipliers. For instance the coefficients {±1,±j} for a W4 multiplier can be easily solved by interchanging real and imaginary parts or changing the sign of the inputs. In [11, 12] twiddle factor multiplier for {W_8,W_{16}, and W_{32}} using constant multiplications were proposed, whereas in [13] a method to increase the accuracy of the rotations based on scaling the coefficients has been presented. The use of general complex multiplier is still the most common way to calculate the twiddle factor multiplication which precomputes and stores the twiddle factors in memory.

In this work we analyze the impact of the FFT algorithm selection on the switching activity (SA) and on the size of the coefficient memory. The switching activity [14], α, is the number of 0→ 1 transitions between successive coefficient that are read from the coefficient memory. The SA is related to the power consumption of the circuit, being able to approximate the dynamic power [15] by

Received November 28, 2013; Revised February 19, 2014; Accepted March 6, 2014

$$P_{dynamic} = \alpha f_{clk} C_L V^2{}_{dd} \quad (1)$$

Where f_{clk} is the clock frequency , V_{dd} is the supply voltage, and C_L is the load capacitance. On the other hand, the memory size is an indicator of the area of the circuit. Thus, each algorithm presents a tradeoff between area and power consumption, which allows for selecting the most efficient algorithm.

The paper is organized as follows. In the next section we briefly review the radix-2 FFT algorithms based the on binary tree decomposition and the FFT architecture that is used for calculating the algorithms. Section III discusses the switching activity of the coefficient memories. Section IV presents experimental results on the switching activity and the total coefficient memory size. In section V conclusions are given.

2. The Binary Tree Representation of FFT Algorithms

2.1. Algorithms

A binary tree can be used to represent the Cooley-Tukey FFT algorithm. This representation is based on the decomposition of an N-point FFT into two smaller FFTs. As a result we can break the N-point FFT into N1-point and N2-point FFTs that are named as inner and outer FFTs, respectively. In a binary tree, each node has leaves which corresponds to the inner and outer FFTs, wherein the left leaf corresponds to the N1 FFTs of N2-points and the right leaf to the N2 FFTs of N1- points. The root node is assigned the power of two weight of the total FFT length and the subnodes are assigned the power of two weight for the inner and outer FFTs, respectively. As a consequence, the weight of a node is the sum of the weights of its subnodes. When the tree is complete, all leaves correspond to butterfly and nodes represent the twiddle factor multiplication. Figure 1 shows the possible decomposition of a node with weight 6, i.e., a 2^6 = 64-point FFT.

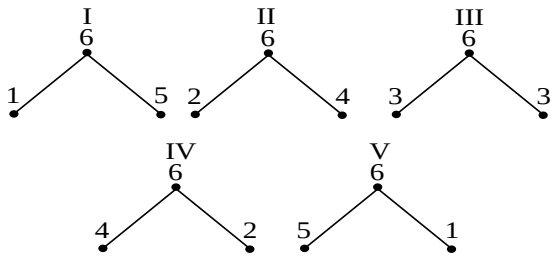


Figure 1. Decomposed Algorithms for 64 Point FFTs

With this representation, a number of possible radix-2 FFT algorithms were generated in [10]. The generated algorithms have a different twiddle factor multiplier for each stage but have the same number of butterfly operations. The number of radix-2 2^n-point FFT algorithms generated using binary trees was shown in [10] to be.

2.2. Architecture

Single delay feedback (SDF) pipelined FFT architecture shown in Figure 2. In this architecture samples enter in natural order and outputs are delivered in bit-reversed order [16]. The number of stages is calculated as n = $\log_2$(N). Each stage consists of a radix-2 butterfly, a complex multiplier, buffers for storing data and memories for the twiddle factor coefficients. Any FFT algorithm generated by the binary tree decomposition can be mapped on the SDF architecture.

$$\frac{(2(n-1))!}{n!(n-1)!} \quad (2)$$

These algorithms only differ in the contents of the twiddle factor coefficient memories, i.e. M1, M2, . . . , Mn−1. As the throughput of SDF architecture is one sample per clock cycle, which means that a coefficient is fetched from the memory every clock cycle.

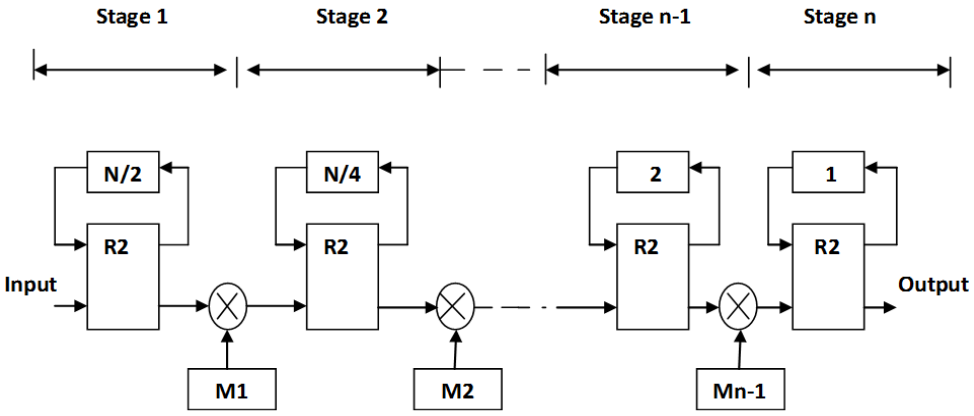


Figure 2. Single Delay Feedback Pipelined FFT Hardware Architecture

The generation of the coefficients that are used for the rotations is detailed in Figure 3. First, φ is generated from the index, being φ a number in the range [0,N − 1] that defines a rotation by:

$$e^{-j\frac{2\Pi}{N}\phi} \quad (3)$$

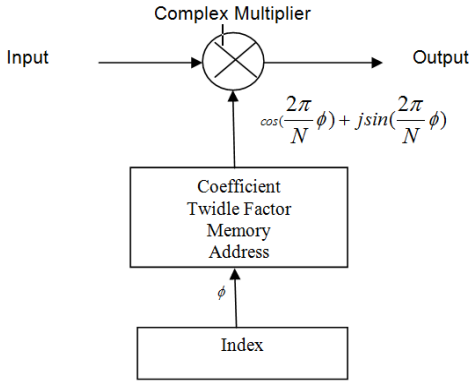


Figure 3. Twidle Factor Multiplication

The generation of φ from the index is explained in detail in [10]. Once φ is obtained, its value is used to read the address of a twiddle factor memory in which the corresponding coefficients, i.e., $\cos(\frac{2\pi}{N}\phi)$ and $\sin(\frac{2\pi}{N}\phi)$, are stored. These coefficients are input to the complex multiplier in order to calculate rotation of input sample In order to reduce the size of the coefficient memory, it is a common practice to utilize the quarter symmetry property of the coefficients [17]. This means that only twiddle factors corresponding to angles in the range of $0 \le \phi \le \pi / 2$ need to be considered. Multiplication for other values can be obtained by appropriate

swapping of real and imaginary outputs and an appropriate sign or by rotating any outstanding $\pi/2$, π, or $3\pi/2$ rad in the following stage, similar to what is done in radix-2^2 FFTs.

3. Switching Activity

The switching activity, α, in (1) is the average number of 0 → 1 and 1 → 0 transitions per clock cycle. In a twiddle factor multiplication, the Switching Activity between successive coefficients is defined in terms of Hamming distance for each coefficient transition. In pipelined SDF architecture the twiddle factor are precomputed and stored in look-up tables and are fed to the complex multiplier in each cycle. The sequence of the coefficient fed to the complex multiplier affects the SA.The algorithms generated by the binary tree have either same twiddle factor complexity with different position of twiddle factor or different twiddle factor complexity. In both cases the twiddle factor memory and switching activity differ between these algorithms.

4. Experimental Results

We consider all the coefficient sequences of the complex multipliers resulting from the FFT algorithms based on binary trees. These are initially represented with 16-bit word length (real and imaginary). The access sequence is then computed to obtain the resulting activity.

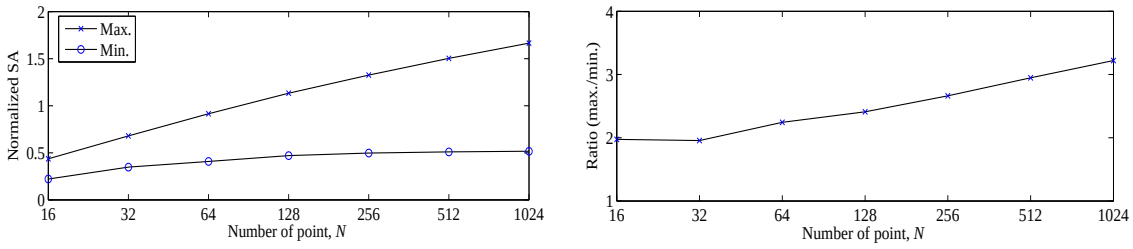
The switching activity is then normalized as:

$$NSA = \frac{SA}{2b.N} \tag{4}$$

Where SA is the switching activity, N is the FFT length and b is the word length of the coefficients. The normalized SA for the algorithms with varying FFT lengths is shown in Figure 4, illustrating the best and the worst case. On the other hand, the normalized SA as a function of the wordlength is shown in Figure 5. As can be noted in Figure 4(a), for a fixed N the normalized SA is almost constant with the number of bits, b. This shows that the switching activity is proportional to the size of the FFT. Likewise, for a fixed b the normalized SA is constant with the FFT size, as shown in Figure 5(a). Therefore, the switching activity is also proportional to the wordlength.

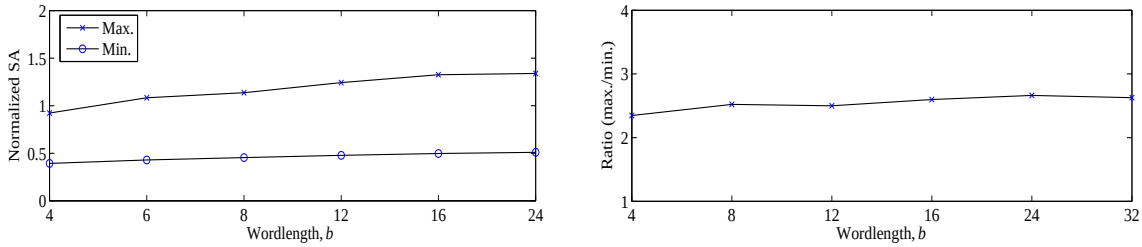
Furthermore, Figure 4(b) and 5(b) show the ratios between the algorithms with the maximum and minimum switching activities. It can be noted that the ratio between the maximum and the minimum switching activity increases with the length of the FFT. Furthermore, the ratios are bigger than 2 in most cases, which leads to savings of more than 50% in the switching activity.

The algorithms with the lowest switching activity are summarized in Table 1 for different FFT lengths. The algorithms shown in the table turn out to be the radix-2^3 decimation in time (DIT) algorithm in all the cases. As mentioned above, we calculate the SA from the W16, due to this fact radix- 2^3 DIT have minimum SA. Similarly, one can expect that if the twiddle factors from W_2^k and higher resolution multipliers was considered, the best algorithm would be a radix-2^{k-1} DIT algorithm.



(a) Normalized switching activity (b) Ratio between the maximum and minimum SA

Figure 4. Normalized Switching Activity of Coefficient Memories of the Different Algorithms of the FFT Size, N, for 16 bit Coefficients

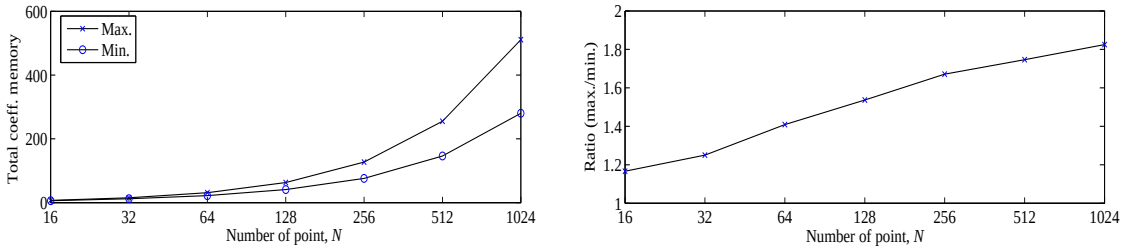


(a) Normalized switching activity (b) Ratio between the maximum and minimum SA

Figure 5. Normalized Switching Activity of the Coefficient Memories of the Different Algorithms as a Function of the Word Length of the Coefficients, b, for a 256 Point FFT

Table 1. Twidle Factor Distribution of the FFT Algorithms with the Minimum SA

Number of Points, N	Stage Number								
	1	2	3	4	5	6	7	8	9
16	W_{16}	W_8	W_4						
32	W_4	W_{32}	W_8	W_4					
64	W_8	W_4	W_{64}	W_8	W_4				
128	W_{16}	W_8	W_4	W_{128}	W_8	W_4			
256	W_4	W_{32}	W_8	W_4	W_{256}	W_8	W_4		
512	W_8	W_4	W_{64}	W_8	W_4	W_{512}	W_8	W_4	
1024	W_{16}	W_8	W_4	W_{128}	W_8	W_4	W_{1024}	W_8	W_4



(a) Total coefficient memory (b) Ration between the maximum and the minimum total coefficient memory

Figure 6. Total Coefficient Memory of the Different Algorithms as a Function of the FFT Size, N, for 16 bit Coefficients

With respect to the size of the coefficient memory, Figure 6 shows the maximum and minimum total coefficient memory size that the different algorithms require. As happened with the switching activity, it can be noted that there is a big difference between the memories required by different algorithms. The savings vary from a 14% for a 16-point FFT to 45% for a 1024-point FFT.

Finally, Figure 7 shows the tradeoff between coefficient memory and switching activity for all the algorithms based on binary trees that can be used to calculate a 256-point FFT. It can be noted that there are big differences among the algorithms both in switching activity and memory size. Furthermore, the algorithms with both low switching activity and memory size can be determined from Figure 7. These are also same algorithms having the lowest switching activity.

5. Conclusion

This paper shows that both the power consumption due to switching activity and the total size of the coefficient memories in FFT hardware architectures are directly related to the FFT algorithm that is used. It has been shown that the switching activity is proportional to both the FFT size and the number of bits of the coefficients. Finally, by comparing the results of all the possible algorithms that can be generated using the binary tree decompositions it can be noted that selecting an efficient algorithm can lead to significant reductions in terms of switching activity and memory size.

References

[1] J Cooley, J Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Comp.*, 1965; 19: 297–301.
[2] SC Moon, IC Park. Area-efficient memory-based architecture for fft processing. *Proc. IEEE Int. Symp. Circuits and Systems*. 2003; 5: V–101– V–104.
[3] S He, M Torkelson. Design and implementation of a 1024-point pipeline FFT processor. *Proc. IEEE Custom Integrated Circuits Conf.*, 1998: 131–134.
[4] H Groginsky, G Works. A pipeline fast Fourier transform. *IEEE Trans. on Computers*. 1970; C-19(11): 1015-1019.
[5] W Han, AT Erdogan, T Arslan, M Hasan. High-performance low-power FFT cores. *ETRI Journal*. 2008; 30(3): 451–460.
[6] M Garrido, KK Parhi, J Grajal. A pipelined FFT architecture for real-valued signals. *IEEE Trans. Circuits Syst. I*. 2009; 56(12): 2634–2643.
[7] MAS´anchez, M Garrido, MLL´opez, J Grajal. Implementing FFT-based digital channelized receivers on FPGA platforms. *IEEETrans. Aerosp. Electron. Syst.,* 2008; 44(4): 1567–1585.
[8] L Wanhammar, *DSP Integrated Circuits*. Academic Press. 1999.
[9] M Garrido. Efficient hardware architectures for the computation of the FFT and other related signal processing algorithms in real time. Ph.D. dissertation, Universidad Polit´ecnica de Madrid. 2009
[10] F Qureshi, O Gustafsson. Generation of all radix-2 fast Fourier transform algorithms using binary trees. *Proc. European Conf. Circuit Theory Design*. 2011.
[11] S Lee, Y Jung, J Kim. Low complexity pipeline FFT processor for MIMO-OFDM systems. *IEICE Electronics Express*. 2007; 4(23): 750–754.
[12] F Qureshi, O Gustafsson. Low-complexity constant multiplication based on trigonometric identities with applications to FFTs. *IEICETrans. Fundamentals*. 2011; E94-A,(11).
[13] M Garrido, O Gustafsson, J Grajal. Accurate rotations based on coefficient scaling. *IEEE Trans. Circuits Syst. II*, 2011.
[14] JM Wu, YC Fan. Coefficient ordering based pipelined FFT/IFFT with minimum switching activity for low power wimax communication system. *Proc. IEEE Tenth Intern. Symp. Consumer Electronics.* 2006: 1–4.
[15] F Najm. A survey of power estimation techniques in VLSI circuits. *IEEE Trans. Very Large Scale Integration (VLSI) Systems,*. 1994; 2(4): 446–455.
[16] M Garrido, J Grajal, O Gustafsson. Optimum circuits for bit reversal. *IEEE Trans. Circuits Syst. II*. 2011.
[17] HY Lee, IC Park. Balanced binary-tree decomposition for areaefficient pipelined FFT processing. *IEEE Trans. Circuits and Syst. I*. 2007; 54(4): 889–900.