

Linköping University Post Print

**Twiddle factor memory switching activity
analysis of radix-2² and equivalent FFT
algorithms**

Fahad Qureshi and Oscar Gustafsson

N.B.: When citing this work, cite the original article.

©2010 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Fahad Qureshi and Oscar Gustafsson, Twiddle factor memory switching activity analysis of radix-2² and equivalent FFT algorithms, 2010, The IEEE International Symposium on Circuits and Systems (ISCAS) , Paris, 2010, 4145-4148.

Postprint available at: Linköping University Electronic Press

<http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-65911>

Twiddle Factor Memory Switching Activity Analysis of Radix- 2^2 and Equivalent FFT Algorithms

Fahad Qureshi and Oscar Gustafsson
 Department of Electrical Engineering, Linköping University
 SE-581 83 Linköping, Sweden
 E-mail: {fahadq, oscarg}@isy.liu.se

Abstract—In this paper, we propose equivalent radix- 2^2 algorithms and evaluate them based on twiddle factor switching activity for a single delay feedback pipelined FFT architecture. These equivalent pipeline FFT algorithms have the same number of complex multipliers with the same resolution as the radix- 2^2 . It is shown that the twiddle factor switching activity of the equivalent algorithms is reduced with up to 40% for some of the equivalent algorithms derived for $N = 256$.

I. INTRODUCTION

Computation of the discrete Fourier transform (DFT) and inverse DFT is used in e.g. orthogonal frequency-division multiplexing (OFDM) communication systems and spectrometers. An N -point DFT can be expressed as

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^k, \quad k = 0, 1, \dots, N-1 \quad (1)$$

where $W_N = e^{-j\frac{2\pi}{N}}$ is the twiddle factor, the N :th primitive root of unity with its exponent being evaluated modulo N , n is the time index, and k is the frequency index. Various methods for efficiently computing (1) have been the subject of a large body of published literature. They are commonly referred to as fast Fourier transform (FFT) algorithms. Also, many different architectures to efficiently map the FFT algorithm to hardware have been proposed [1].

A commonly used architecture for transforms of length $N = b^r$ is the pipelined FFT [2]. The pipeline architecture is characterized by continuous processing of input data. In addition, the pipeline architecture is highly regular, making it straightforward to automatically generate FFTs of various lengths. In a pipeline architecture the complex multiplier is one of the most power-consuming unit.

Figure 1 outlines the architecture of a Radix- 2^i single-path delay feedback (SDF) decimation in frequency (DIF) pipeline FFT architecture of length N . This architecture is generic while the required ranges of each complex twiddle factor multiplier is outlined in Table I for varying numbers of i . For the twiddle factor multipliers with small ranges special methods have been proposed. Especially, one can note that for a W_4 multiplier the possible coefficients are $\{\pm 1, \pm j\}$ and, hence, this can be simply solved by optionally interchanging real and imaginary parts and possibly negate (or replace the

TABLE I
 MULTIPLICATION RESOLUTION AT DIFFERENT STAGES FOR VARIOUS FFT ALGORITHMS ($N = 256$).

Radix	Stage number						
	1	2	3	4	5	6	7
2	W_{256}	W_{128}	W_{64}	W_{32}	W_{16}	W_8	W_4
2^2 [3]	W_4	W_{256}	W_4	W_{64}	W_4	W_{16}	W_4
2^3 [4]	W_4	W_8	W_{256}	W_8	W_8	W_{32}	W_4
2^4 [5]	W_4	W_8	W_{16}	W_{256}	W_4	W_8	W_{16}
2^5 [6]	W_4	W_8	W_{16}	W_{32}	W_{256}	W_4	W_8
2^6 [6]	W_4	W_8	W_{16}	W_{32}	W_{64}	W_{256}	W_4

addition with a subtraction in the subsequent stage). In [5], [8] twiddle factor multiplier for $\{W_8, W_{16}, \text{ and } W_{32}\}$ using constant multiplication were proposed. However, a common way to solve the twiddle factor multiplication is to use a general complex multiplier and precompute the twiddle factors and store in a memory.

In integrated circuits, low power design is always desirable. In digital CMOS circuits, dynamic power is the dominating part of the total power consumption which can be approximated by [9]

$$P_{dyn} = \frac{1}{2} V_{DD}^2 f_c C_L \alpha \quad (2)$$

where V_{DD} is the supply voltage, f_c is the clock frequency, C_L is the load capacitance and α is the switching activity. In this work we focus on the switching activity and how to reduce the switching activity between two successive coefficients fed to the complex multiplier.

In [11]–[14], methods for reducing the size of the coefficient memory has been proposed. In [10], [15], methods for reducing the switching activity between successive twiddle factor coefficients have been proposed. However, these methods comes with a hardware overhead. In this work we focus on the algorithms derived from radix- 2^i having same memory complexity as the standard radix- 2^2 algorithm, i.e., the same resolution of the twiddle factors. However, as will be seen, the twiddle factor memory switching activity differs between the different algorithms.

The rest of the paper is organized as follows. In next section, the radix- 2^2 algorithm and equivalent algorithms derived from radix- 2^i are presented for $N = 256$. Then in Section III, some

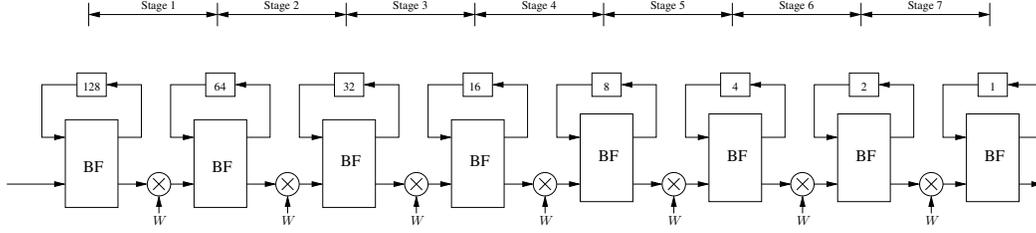


Fig. 1. The $R2^i$ single-path delay feedback (SDF) decimation in frequency (DIF) pipeline FFT architecture ($N = 256$) with twiddle factor stages as used in Table I.

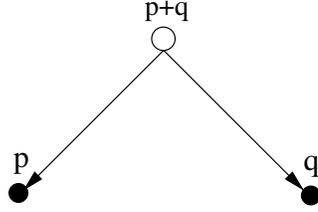


Fig. 2. Illustration of binary tree corresponding to (3).

results are presented and, finally, some conclusions are given in Section IV.

II. RADIX 2^2 FFT AND ITS EQUIVALENT ALGORITHMS

The Cooley-Tukey FFT algorithm can be expressed as

$$X[Qk_1 + k_2] = \sum_{n_1=0}^{P-1} \left[\left(\sum_{n_2=0}^{Q-1} x[n_1 + Pn_2] W_Q^{n_2 k_2} \right) W_M^{n_1 k_2} \right] W_P^{n_1 k_1} \quad (3)$$

$0 \leq n_1, k_1 \leq P - 1; 0 \leq n_2, k_2 \leq Q - 1$

In this algorithm, N, P and Q are considered to be powers of 2, i.e., $N = 2^{p+q}$, $P = 2^p$ and $Q = 2^q$ where p and q are positive integers. Here, the N -point DFT is decomposed into the Q P -point and P Q -point DFTs. Between these DFTs we have twiddle factor multiplications. Typically, the P and Q -point DFTs are again divided into smaller DFTs. An efficient representation of algorithms of this type is the binary tree representation [7]. An example of a binary tree is shown in Fig. 2 corresponding to (3). The left branch corresponds to the $P = 2^p$ -point DFT and the right branch to the $Q = 2^q$ -point DFT. The resolution of the interconnecting twiddle factor is $N = 2^{p+q}$, i.e., a W_N multiplier is required. A radix- 2^2 decimation in frequency algorithm is shown in Fig. 3. In the remainder of this section we will present the radix- 2^2 algorithm and other algorithms having the same intermediate node values as the radix- 2^2 algorithm, but different binary trees. The naming of the resulting algorithms are shown in Table II.

A. Case I

The radix- 2^2 algorithm have identical structure to radix-2 and are computationally identical to radix-4. In a pipeline FFT architecture, a structural advantage over the other algorithms

TABLE II
BINARY TREE ALGORITHMS .

Case	Figure	Comments
I	Fig. 3	Radix- 2^2
II	Fig. 4(a)	Radix 2^2 & Modified Radix- 2^4
III	Fig. 4(b)	Modified Radix- 2^4
IV	Fig. 4(c)	Modified Radix- 2^6
V	Fig. 4(d)	Modified Radix- 2^6

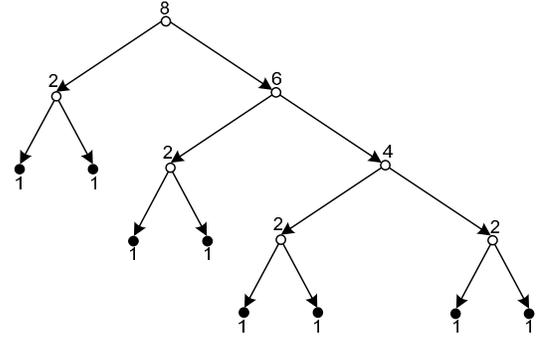


Fig. 3. Binary tree representation of Radix- 2^2 DIF algorithm.

that the non-trivial multiplication operations are after every other stage. Figure 3 represents the binary tree diagram, each node corresponding to the twiddle factor multiplication. Twiddle factors are indexed by the n and k , the linear index map equations and sequences of required n and k to determine the index. Twiddle factors with indices are tabulated in Table III.

B. Case II

In this algorithm, the 256-point DFT is decomposed based on the radix- 2^2 [3] for the first stages, then the modified radix- 2^4 [5] is applied to the remaining stages. The radix- $(2^2 \& M \cdot 2^4)$ algorithm is characterized that it has same twiddle factor complex multiplier as the radix- 2^2 for the W_N multiplier. For instance consider the 256-point FFT, corresponding twiddle factors and indices with n and k sequences are shown in Tables III and IV, respectively. The binary tree representation of the algorithm is shown in Fig. 4(a).

C. Case III

This algorithm is considered as a balanced binary-tree decomposition [7] shown in Fig. 4(b). It can also be seen

TABLE III
TWIDDLE FACTOR EQUATIONS FOR ALL CASES.

Case	1	2	3
I	$W_{256}^{n_3(k_1+2k_2)}$ $n = 128n_1 + 64n_2 + n_3$ $k = k_1 + 2k_2 + 4k_3$ $\{n_1, n_2 = 0, 1, n_3 = 0 \sim 63\}$ $\{k_1, k_2 = 0, 1, k_3 = 0 \sim 63\}$	$W_{64}^{n_3(k_1+2k_2)}$ $n = 32n_1 + 16n_2 + n_3$ $k = k_1 + 2k_2 + 4k_3$ $\{n_1, n_2 = 0, 1, n_3 = 0 \sim 15\}$ $\{k_1, k_2 = 0, 1, k_3 = 0 \sim 15\}$	$W_{16}^{n_3(k_1+2k_2)}$ $n = 8n_1 + 4n_2 + n_3$ $k = k_1 + 2k_2 + 4k_3$ $\{n_1, n_2 = 0, 1, n_3 = 0 \sim 3\}$ $\{k_1, k_2 = 0, 1, k_3 = 0 \sim 3\}$
II	$W_{256}^{n_3(k_1+2k_2)}$ $n = 128n_1 + 64n_2 + n_3$ $k = k_1 + 2k_2 + 4k_3$ $\{n_1, n_2 = 0, 1, n_3 = 0 \sim 63\}$ $\{k_1, k_2 = 0, 1, k_3 = 0 \sim 63\}$	$W_{16}^{n_3(2n_3+n_4)(k_1+2k_2)}$ $n = 32n_1 + 16n_2 + 8n_3 + 4n_4 + n_5$ $k = k_1 + 2k_2 + 4k_3 + 8k_4 + 16k_5$ $\{n_1, n_2, n_3, n_4 = 0, 1, n_5 = 0 \sim 3\}$ $\{k_1, k_2, k_3, k_4 = 0, 1, k_5 = 0 \sim 3\}$	$W_{64}^{n_5(k_1+2k_2+4k_3+8k_4)}$ $n = 32n_1 + 16n_2 + 8n_3 + 4n_4 + n_5$ $k = k_1 + 2k_2 + 4k_3 + 8k_4 + 16k_5$ $\{n_1, n_2, n_3, n_4 = 0, 1, n_5 = 0 \sim 3\}$ $\{k_1, k_2, k_3, k_4 = 0, 1, k_5 = 0 \sim 3\}$
III	$W_{16}^{n_3(2n_3+n_4)(k_1+2k_2)}$ $n = 32n_1 + 16n_2 + 8n_3 + 4n_4 + n_5$ $k = k_1 + 2k_2 + 4k_3 + 8k_4 + 16k_5$ $\{n_1, n_2, n_3, n_4 = 0, 1, n_5 = 0 \sim 15\}$ $\{k_1, k_2, k_3, k_4 = 0, 1, k_5 = 0 \sim 15\}$	$W_{256}^{n_5(k_1+2k_2+4k_3+8k_4)}$ $n = 32n_1 + 16n_2 + 8n_3 + 4n_4 + n_5$ $k = k_1 + 2k_2 + 4k_3 + 8k_4 + 16k_5$ $\{n_1, n_2, n_3, n_4 = 0, 1, n_5 = 0 \sim 15\}$ $\{k_1, k_2, k_3, k_4 = 0, 1, k_5 = 0 \sim 15\}$	$W_{16}^{(2n_3+n_4)(k_1+2k_2)}$ $n = 32n_1 + 16n_2 + 8n_3 + 4n_4 + n_5$ $k = k_1 + 2k_2 + 4k_3 + 8k_4 + 16k_5$ $\{n_1, n_2, n_3, n_4 = 0, 1, n_5 = 0\}$ $\{k_1, k_2, k_3, k_4 = 0, 1, k_5 = 0\}$
IV	$W_{64}^{(8n_3+4n_4+2n_5+n_6)(k_1+2k_2)}$	$W_{16}^{(2n_5+n_6)(k_3+2k_4)}$	$W_{256}^{n_7(k_1+2k_2+4k_3+8k_4+16k_5+32k_6)}$
V	$W_{16}^{(2n_3+n_4)(k_1+2k_2)}$	$W_{64}^{(2n_5+n_6)(k_3+2k_4)}$	$W_{256}^{n_7(k_1+2k_2+4k_3+8k_4+16k_5+32k_6)}$
	$n = 128n_1 + 64n_2 + 32n_3 + 16n_4 + 8n_5 + 4n_6 + n_7$ $k = k_1 + 2k_2 + 4k_3 + 8k_4 + 16k_5 + 32k_6 + 64k_7$ $\{n_1, n_2, n_3, n_4, n_5, n_6 = 0, 1, n_7 = 0 \sim 3\}$ $\{k_1, k_2, k_3, k_4, k_5, k_6 = 0, 1, k_7 = 0 \sim 3\}$		

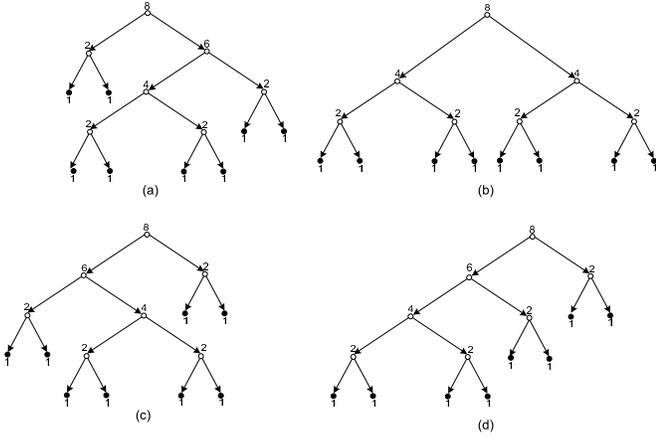


Fig. 4. Binary tree representation of radix- 2^2 equivalent algorithms .

as applying a modified radix- 2^4 algorithm for the first stages, followed by a radix- 2^2 algorithm for the remaining stages. It is worth noticing that this algorithm is not exactly equivalent to radix- 2^2 as the W_{64} -multiplier is replaced by a W_{16} -multiplier. This should in general lead to lower memory requirements. The Twiddle factors with required index sequences are tabulated in Tables III and IV.

TABLE IV
MULTIPLICATION AT DIFFERENT STAGES FOR ALL CASES.

Case	Stage number						
	1	2	3	4	5	6	7
I	W_4	W_{256}	W_4	W_{64}	W_4	W_{16}	W_4
II	W_4	W_{256}	W_4	W_{16}	W_4	W_{64}	W_4
III	W_4	W_{16}	W_4	W_{256}	W_4	W_{16}	W_4
IV	W_4	W_{64}	W_4	W_{16}	W_4	W_{256}	W_4
V	W_4	W_{16}	W_4	W_{64}	W_4	W_{256}	W_4

D. Cases IV and V

In these algorithm, 256-point DFT is decomposed to get the modified radix- 2^6 algorithm [6]. There are two decomposition which have same complexity as radix- 2^2 algorithm. Figures 4(c) and 4(d) show the binary tree representation in which the difference is only the position of the W_{64} and W_{16} twiddle factors. Twiddle factors of all stages for both cases are shown in Table IV. The sequences for k and n are important to determine the index. As seen in Table III, the sequences and ranges with linear index map equations are tabulated. It should be noted that Case V corresponds to a radix- 2^2 decimation in time (DIT) algorithm.

III. RESULTS

The switching activity between successive coefficient fed to the complex multiplier is defined in terms of Hamming

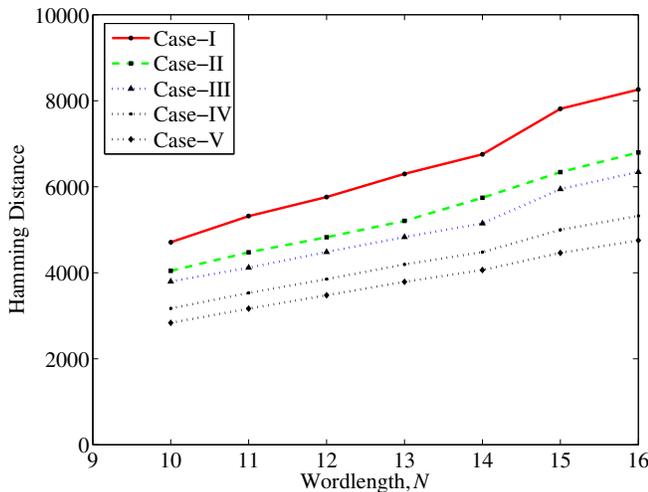


Fig. 5. Twiddle factor memory switching activity for varying wordlengths.

distance for each coefficient transition. The Hamming distance is defined the number of 1's of the XOR operation between two binary coefficient. Twiddle factors can be precomputed and store in look-up tables instead of calculating in real time. In pipelined SDF architecture, in each cycle these stored coefficients are fed to the complex multiplier. The sequence of the stored coefficient affects the switching activity.

All the coefficient sequence of the complex multiplier having in both radix-2² and equivalent algorithms are encoded with different word lengths and two's complement representation. The reading sequence is then simulated to obtain the resulting switching activity. The results for the different algorithms are shown in Table V, where it can be seen that the different algorithms have significantly different twiddle factor switching activity. The algorithm with the lowest total switching activity is Case V.

Results using different wordlengths are shown in Fig. 5. These results confirms that for $N = 256$ Case V provides the lowest twiddle factor switching activity.

In [5], the W_{16} is implemented through the use of a dedicated constant multiplier. Hence, it is for this case of interest to know how often the multiplier coefficient is changed. The results in Table VI shows that the number of coefficient changes can be significantly reduced using an equivalent algorithm.

TABLE V
SWITCHING ACTIVITY OF RADIX-2² AND EQUIVALENT ALGORITHMS
(16-BITS)

Twiddle factor	Case I	Case II	Case III	Case IV	Case V
W_{16}	3088	760	178+3088	760	178
W_{64}	2701	3566	-	661	672
W_{256}	2471	2471	3077	3901	3901
Total	8260	6797	6343	5322	4751
Reduction	-	17.7%	23.2%	35.6%	42.5%

TABLE VI
NUMBER OF OUTPUT CHANGE IN W_{16} MULTIPLIER

Case I	Case II	Case III	Case IV	Case V
192	48	12, 192	48	12

IV. CONCLUSIONS

In this work, we discuss the different equivalent algorithms of Radix-2² having the same implementation complexity but with possibly less switching activity between subsequent twiddle factor coefficients. It is shown that the twiddle factor switching activity of the equivalent algorithms can be reduced with more than 40% for some of the equivalent algorithms. Even though the corresponding effect on the power consumption should be evaluated, one would expect a significant reduction.

REFERENCES

- [1] L. Wanhammar, *DSP Integrated Circuits*, Academic Press, 1999.
- [2] E. H. Wold and A. M. Despain, "Pipeline and parallel-pipeline FFT processors for VLSI implementations," *IEEE Trans. Comp.*, vol. 33, no. 5, pp. 414–426, May 1984.
- [3] S. He and M. Torkelson, "A new approach to pipeline FFT processor," in *Proc. IEEE Parallel Processing Symp.*, 1996, pp. 766–770.
- [4] S. He and M. Torkelson, "Designing pipeline FFT processor for OFDM(de)Modulation," in *Proc. IEEE URSI Int. Symp. Sig. Elect.*, 1998, pp. 257–262.
- [5] J.-E. Oh, and M.-S. Lim, "New radix-2 to the 4th power pipeline FFT processor," *IEICE Trans. Electron.*, vol. E88-C, no. 8, pp. 694–697, Aug. 2005.
- [6] A. Cortes, I. Velez and J. F. Sevilano, "Radix r^k FFTs: Matricial Representation and SDC/SDF Pipeline Implementation," *IEEE Trans. on Signal Processing*, vol. 57, no. 7, pp. 2824–2839, July 2009.
- [7] Hyun-Yong Lee, and In-Cheol Park, "Balanced Binary-Tree Decomposition for Area-Efficient Pipelined FFT Processing," *IEEE Trans. on Circuits and Systems-I*, vol. 54, no. 4, pp. 889–900, April 2009.
- [8] F. Qureshi and O. Gustafsson, "Low-complexity reconfigurable complex constant multiplication for FFTs," in *Proc. IEEE Int. Symp. Circuits Syst.*, Taipei, Taiwan, May 24–27, 2009.
- [9] K. Johansson, O. Gustafsson, and L. Wanhammar, "Switching activity estimation for shift-and-add based constant multipliers," in *Proc. IEEE Int. Symp. Circuits Syst.*, Seattle, WA, USA, May. 18-21, 2008.
- [10] J. Ming Wu and Y. Chun Fan, "Coefficient Ordering Based Pipelined FFT/IFFT with Minimum Switching Activity for Low Power WiMAX Communication System," in *Proc. IEEE Tenth Int. Symp. Consumer Electronics*, 2006, pp. 1–4.
- [11] Seungbeom Lee, Duk-bai Kim and Sin-Chong Park, "Power-efficient design of memory based FFT processor with new addressing scheme," in *Proc. Int. Symp. Communications and Information Technology*, 26–29 Oct. 2004, pp. 678–681.
- [12] F. Qureshi and O. Gustafsson, "Analysis of Twiddle Factor Memory Complexity of Radix-2ⁱ Pipelined FFTs," in *Proc. Asilomar Conf. Signals Syst. Comp.*, Pacific Grove, CA, Nov. 1-4, 2009.
- [13] H. Cho, M. Kim, D. Kim, and J. Kim "R²SDF FFT implementation with coefficient memory reduction scheme," in *Proc. Vehicular Technology Conf.*, 2006.
- [14] M. Hasan and T. Arslan, "Scheme for reducing size of coefficient memory in FFT processor," *Electronics Letters*, vol. 38, no. 4, pp. 163–164, Feb. 2007.
- [15] K. Masselos, S. Theoharis, P. K. Merakos, T. Stouraitis and C. E. Goutis, "A novel methodology for power consumption reduction in a class of DSP algorithms," in *Proc. IEEE Int. Symp. Circuits Syst.*, 1998, vol. VI, pp. 199–202.