

FFT COEFFICIENT MEMORY REDUCTION TECHNIQUE FOR OFDM APPLICATIONS

M. Hasan and T. Arslan

Department of Electronics and Electrical Engineering
The University of Edinburgh
The King's Buildings, Mayfield Road
Edinburgh, EH9 3JL, U.K.

ABSTRACT

There is a strong need to implement long FFT's in applications like orthogonal frequency division multiplexing (OFDM), radars and sonars etc. It is highly desirable to reduce the size and power requirements of the FFT so as to realize single chip long FFT based systems targeting portable applications. This paper presents a novel technique to reduce the coefficient memory almost by a factor of four by exploiting the relationships among the coefficient values thereby significantly reducing the area and power requirements of the hardware.

1. INTRODUCTION

Fast fourier transform (FFT) has to be computed in numerous portable Telecommunication applications where area and power consumption are of prime importance. Long FFT's are common in OFDM, radars and sonars etc. Hence, it is very important to minimize FFT hardware for its further area and power efficient realisation[1]. The number of coefficients in a radix-2 FFT is equal to $N/2$, where 'N' is the length of the FFT. The coefficient memory required to store these coefficients will thus require $N/2$ locations where each location stores the real and the imaginary part of the coefficient in a conventional implementation, here we term it *Conv* [2]. Ma[3] and Chang[4] proposed that the size of the coefficient memory can be cut to half ($N/4$) by exploiting the correlation among the coefficient values, here we term it *Others*. This paper presents a

novel technique to reduce the size of the coefficient memory by almost a factor of four ($(N/8)+1$). We provide results demonstrating that by adopting *Our* technique the savings in area and power will be substantial for long FFT's in different types of realisations compared to the *Others* memory based techniques. The power and area savings are in the range of 48-54% and 0.62-59% respectively for long length FFT processors ($1K \geq$).

2. DESIGN TECHNIQUE

The N-point discrete Fourier Transform (DFT) is defined by

$$X_k = \sum_{m=0}^{N-1} x_m \cdot W_N^{mk}$$

where $W_N = \exp(-j2\pi/N)$ and $k = 0, 1, \dots, N-1$. The radix-2 FFT [5] is an efficient way to compute an N-point Discrete Fourier Transform (DFT). The basic operations in an FFT are multiplication of the complex data inputs (x_m) by the FFT coefficients ($\exp(-j2\pi k/N)$) at each stage in the signal flowgraph followed by their summation or subtraction and associated data and coefficient address generation. There are $N/2$ FFT coefficients for an N-point FFT where 'N' indicates the number of data points or the length of the FFT. The values of the coefficients for a 32-point FFT in 16-bit 2's complement format are given in Table 1. The partitioning of coefficients into blocks (as depicted in Table 1) is applicable to all FFT lengths. *Others* proposed to divide the memory into two identical blocks namely A and B. Block A corresponds to the

Table 1: Description of the various memory partitioning Schemes.

Address	Coefficient set (real,Imag)	16-bit quantised coefficient values	Our scheme	Others scheme
0000	1.0, 0.0	7fff,0000	Block I	Block A
0001	.98, -.19	7d89,e706		
0010	.92, -.38	7641,cf04		
0011	.83, -.55	6a6d,b8e3		
0100	.71, -.71	5a82,a57d		
0101	.55, -.83	471c,9592		
0110	.38, -.92	30fb,89be	Block II	Block B
0111	.19, -.98	18f9,8276		
1000	0.0, -1.0	0000,8000	Block III	
1001	-.19, -.98	e706,8276		
1010	-.38, -.92	cf04,89be		
1011	-.55, -.83	b8e3,9592	Block IV	
1100	-.71, -.71	a57d,a57d		
1101	-.83, -.55	9592,b8e3		
1110	-.92, -.38	89be,cf04		
1111	-.98, -.19	8276,e706		

low value of the most significant bit (MSB) whereas block B is addressed by its high value. All the remaining three bits are identical for both the blocks. It is clear that the coefficient values in block B can be generated from those in block A by interchanging the real and imaginary parts of the coefficients and by also complementing the real part before its assignment to the imaginary part corresponding to block B. Hence, only block A needs to be stored and all the entries of block B can be generated by adopting the above-mentioned method. In *Our* technique, the memory is partitioned into four blocks (Block I to Block IV) rather than two as depicted again in Table 1. The memory size in *Our* scheme is reduced to $((N/8)+1)$ locations (Only Block I is needed) from the $N/4$ locations proposed by *Others* schemes (Only Block A is needed). Using *Our* scheme, there is a need to store only coefficient values in block I and the rest of the coefficient values in other blocks and their corresponding first block addresses can be generated by following the general procedure given below. This procedure can be explained with the help of a 32-point FFT example given in Table 1.

Let the complex coefficient values in terms of the real and imaginary parts be represented as in

equation (1).

$$Z_{bg} = R_{bg} + j I_{bg} \quad (1)$$

Where 'b' indicates the memory block number and 'g' is an index which points to the coefficient values within individual blocks. The first coefficient value in each block has an index 'g' equal to zero.

The first block coefficient values are obtained from (1) by replacing 'b' with '1' as shown in equation (2).

$$Z_{1g} = R_{1g} + j I_{1g} \quad 0 \leq g \leq N/8 \quad (2)$$

Let the coefficient memory address generated and the actual block address be represented by an n-bit array 'A_m' and an (n-1) bit array 'A_{bg}' respectively. The coefficient memory block address is always one bit less than the conventional coefficient memory address as the block size is limited to $((N/8)+1)$ instead of $N/2$. The corresponding addresses of the coefficient values in the first block as per definition are given by the following equation.

$$A_{1g}[n-2 : 0] = A_m[n-2 : 0]$$

Where $n = \log_2(N/2)$, $0 \leq g \leq N/8$,
 $0 \leq m \leq N/8$

The second block coefficient values can be obtained in terms of the first block coefficient values by performing the following substitution in the right-hand side of equation (2) : $R_1 \rightarrow \sim I_1$, $I_1 \rightarrow \sim R_1$ and 'g' $\rightarrow (N/8-1-g)$. The symbol '~' here corresponds to a complement operation. This can also be verified from Table I. The resulting equation is as follows.

$$Z_{2g} = [\sim I_{1(N/8-1-g)}] + j [\sim R_{1(N/8-1-g)}]$$

Where, $0 \leq g \leq ((N/8)-2)$

When the coefficient memory address generator proceeds to generate the address in the second block, the corresponding address in the first block (Only Block I is stored) are obtained by taking the two's complement of the coefficient memory address as follows.

$$A_{2g} = \sim A_m[n-2:0] + 1 \quad 0 \leq g \leq ((N/8)-2)$$

$$((N/8)+1) \leq m \leq ((N/4)-1)$$

Similarly, the coefficient values in the third block are obtained from the first block coefficient values using equation (2) as follows.

$$Z_{3g} = [I_{1g}] + j [\sim R_{1g}] \quad 0 \leq g \leq N/8$$

Similarly, when the coefficient memory address generator proceeds to generate the address in the third block, the corresponding address in the first block are given by the following equation.

$$A_{3g} = A_m[n-2:0] \quad 0 \leq g \leq N/8 \\ N/4 \leq m \leq 3N/8$$

Similarly, the fourth block coefficient values are obtained in terms of the first block coefficient values again using equation (2) as follows.

$$Z_{4g} = [\sim R_{I(N/8-1-g)}] + j [I_{I(N/8-1-g)}] \\ \text{Where, } 0 \leq g \leq ((N/8)-2)$$

When the coefficient memory address generator proceeds to generate the address in the fourth block, the corresponding address in the first block are given by the following equation.

$$A_{4g} = \sim A_m[n-2:0] + 1 \quad 0 \leq g \leq ((N/8)-2) \\ ((3N/8+1) \leq m \leq (N/2)-1)$$

In order to understand the above scheme, consider the following example.

In Table I, the real and imaginary coefficient values for a 32-point FFT (n equals 4) corresponding to index '0' of the second block are .55 and -.83 respectively. The coefficient address $A_5[2:0]$ for these values is 101. These coefficient values can be generated by moving to index 3 of the first block and then by complementing and interchanging the real and imaginary values stored at this address. The appropriate first block address $A_{20}(011)$ corresponding to index 3 is generated by taking the Two's complement of the coefficient memory address $A_5[2:0]$.

The blocks can be easily identified with the help of the higher two bits along with the all zero combination of the remaining bits. Hence, only block I needs to be stored and the remaining blocks

can be generated by designing an additional hardware for implementing the above-mentioned procedure. The delay introduced by the additional block is not of much consequence because the coefficient delay in a typical FFT implementation is half that of the data delay [6]. Moreover, the access time of memory in our case is much lower on account of its smaller size. The same concept can be extended to longer FFT lengths.

3. RESULTS

Coefficient memories corresponding to different FFT lengths are synthesized from the RTL (Register Transfer level) Verilog description using Cadence *BuildGates* with 0.35u Alcatel MTC 45000 CMOS technology library. The power based gate level simulations were carried out at a clock frequency of 10 MHz and at a supply voltage of 3.3V for one million clock cycles using Synopsys *DesignPower*. The same procedure was followed for the *Conv* and *Others* approaches and the comparative results in terms of area and power for the different FFT lengths are given for all the schemes in Table 2 and Table 3 respectively. It is clear from Table 2 that the savings in area continuously increases with increase in FFT lengths. This is because the additional hardware to implement *Our* scheme remains almost fixed. *Our* scheme proves to be beneficial for 1K-points onwards both in terms of area and power. The savings in area ranges from 48% to 54% whereas the power savings varies from 0.62% to 59% for longer FFT's (1K-points onwards). The percentage savings in terms of area and power for long FFT's are illustrated in figure 1.

4. CONCLUSION

This paper has presented a coefficient memory reduction technique that is more power and area efficient than the other memory based schemes in the literature. This reduction is more significant for longer FFT's which are crucial in OFDM applications. It will lead to the design of more power and area efficient long length FFT processors. The power and area issues are extremely important in portable applications.

Table 2: Comparison of the two approaches in terms of area for 16-bit coefficients

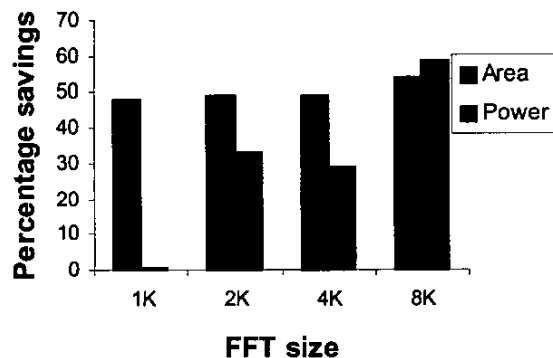
FFT size	Conv. Scheme [n]	Others Scheme [n]	Our Scheme [n]	% reduction over <i>Others</i> Scheme
64	259	212	179	16
128	520	343	256	25
256	1012	605	383	37
512	2222	1070	652	39
1K	4452	2232	1168	48
2K	8729	4449	2272	49
4K	19986	8526	4391	49
8K	34890	18366	8378	54

[n] equals number of equivalent nand gates

Table 3: Percentage savings in power of *Our* scheme over *Others* schemes for long FFT's.

FFT size	Conv. Scheme (uW)	Others Schemes (uW)	Our Scheme (uW)	% reduction over <i>Others</i> schemes
64	183	228	283	-24
128	222	261	355	-36
256	286	322	406	-26
512	372	383	474	-24
1K	714	485	482	+0.62
2K	867	934	623	+33
4K	3403	1121	797	+29
8K	5296	3331	1353	+59

Figure 1: Percentage savings in area and power of *Our* scheme over *Others* schemes for long FFT's.



5. REFERENCES

- [1] T. Arslan, A.T. Erdogan and D.H. Horrocks, Low Power Design For DSP: Methodologies and Techniques, Microelectronics Journal, Vol. 27, No. 8, pp.731-744, November 1996.
- [2] D.Cohen, "Simplified Control of FFT hardware", IEEE transactions on acoustics, speech and signal processing, vol. ASSP-24, pp. 577-579, Dec. 1976.
- [3] Y.Ma and L. Wanhammar, "A Hardware efficient control of memory addressing for high performance FFT processors", IEEE transactions on signal processing, vol. 48, no. 3, pp. 917-921, March, 2000.
- [4] Y. Chang and Keshab K. Parhi, "Efficient FFT implementation using digit-serial arithmetic", IEEE Workshop on signal processing systems, pp. 645-653, 1999.
- [5] L.R. Rabiner, B. Gold, "Theory and Application of Digital signal processing", Englewood Cliffs, NJ : Prentice-Hall, 1975.
- [6] Y.Ma, "An effective memory addressing scheme for FFT processors", IEEE transactions on signal processing, vol. 47, no. 3, pp. 907-911, March, 1999.