

A. Hati, M. Ghosh and B.C. Sarkar (Physics Department, Burdwan University Burdwan-713104, West Bengal, India)

References

- 1 LEE, T.H., and BULZACHELLI, J.F.: 'A 155-MHz clock recovery delay and phase locked loop', *IEEE J. Solid-State Circuits*, 1992, **27**, (12), pp. 1736–1745
- 2 HOGGE, C.R.: 'A self correcting clock recovery circuit', *J. Lightwave Technol.*, 1985, **LT-3**, (6), pp. 1312–1314
- 3 SARKAR, B.C., and HATI, A.: 'PLL-based frequency synthesiser without using the frequency divider', *IEE Proc., Circuits Devices Syst.*, **148**, (5), pp. 255–260

Scheme for reducing size of coefficient memory in FFT processor

M. Hasan and T. Arslan

Long fast Fourier transforms (FFTs) are required in applications such as orthogonal frequency division multiplexing, radars and sonars. It is highly desirable to reduce the size and power requirements of the FFT so as to realise single chip long FFT-based systems targeting portable applications. Presented here is 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.

Introduction: The area and power consumption are of prime importance in portable telecommunication applications. Fast Fourier transform (FFT) has to be computed in many of these applications. Long FFTs are common in orthogonal frequency division multiplexing (OFDM), radars and sonars, etc. Hence, it is very important to minimise FFT hardware for its further area and power efficient realisation. 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* [1]. Ma and Wanhammar [2] and Chang and Parhi [3] 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*. In this Letter we present 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 FFTs in different types of realisations compared to the *Others* memory-based techniques.

Scheme description: The FFT coefficients are expressed as follows:

$$W_k = \exp(-j2\pi k/N)$$

where k varies from 0 to $N/2 - 1$ giving rise to $N/2$ 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 2's complement form are given in Table 1. The partitioning of coefficients into blocks (as shown in Table 1) is applicable to all FFT lengths. *Others* proposed to divide the memory into two identical blocks, namely A and B. 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. In *Our* technique, the memory is partitioned into four blocks (Block I to Block IV) rather than two, as shown 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.

Table 1: Description of various memory organisation schemes

Address	Coefficient set (real, imaginary)	16-bit quantised coefficient values	Our scheme	Others schemes
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	Block II	Block B
0101	.55, -.83	471c, 9592		
0110	.38, -.92	30fb, 89be		
0111	.19, -.98	18f9, 8276		
1000	0.0, -1.0	0000, 8000	Block III	Block B
1001	-.19, -.19	e706, 8276		
1010	-.38, -.92	cf04, 89be		
1011	-.55, -.83	b8e3, 9592		
1100	-.71, -.71	a57d, a57d	Block IV	Block B
1101	-.83, -.55	9592, b8e3		
1110	-.92, -.38	89be, cf04		
1111	-.98, -.19	8276, e706		

Let the complex coefficient values in terms of the real and imaginary parts be represented as in (1):

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

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 (2):

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

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 1 bit less than the conventional 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 are given by the following equation:

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

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

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 (2): $R_1 \rightarrow \sim I_1$, $I_1 \rightarrow \sim R_1$ and $g \rightarrow (N/8 - 1 - g)$. The symbol \sim here corresponds to a complement operation. This can also be verified from Table 1. The resulting equation is as follows:

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

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 2'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 (2) as follows:

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

When the coefficient memory address generator proceeds to generate the addresses in the third block, the corresponding first block addresses are obtained as follows:

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

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

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

Similarly for the fourth block, the corresponding addresses in the first block are given by the following equation.

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

To understand the above scheme, let us consider an example. In Table 1, the real and imaginary coefficient values for a 32-point FFT (n equals 4) corresponding to index '0' of the second block are 0.55 and -0.83 , 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 2'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 1 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 since the coefficient delay in a typical FFT implementation is half that of the data delay [4]. Moreover, the access time of memory in our case is much lower on account of its smaller size.

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

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

[n] equals number of equivalent NAND gates

Table 3: Comparison of two approaches in terms of power for 16-bit coefficients

FFT size	Conv scheme [μ W]	Others scheme [μ W]	Our scheme [μ W]	% reduction of Our scheme over Others schemes
64	183	228	283	-24
128	222	261	355	-36
256	286	322	406	-26
512	372	383	474	-24
1 K	714	485	482	+0.62
2 K	867	934	623	+33
4 K	3403	1121	797	+29
8 K	5296	3331	1353	+59

Simulation results: The scheme has been implemented in register transfer level Verilog hardware description language for different FFT lengths and then synthesised using Cadence BuildGates with 0.35 μ Alcatel MTC 45000 CMOS technology library. Power evaluation was carried out using Synopsys DesignPower for the circuit netlist. Simulations were carried out for 1 million clock cycles using a supply voltage of 3.3 V and a clock frequency of 10 MHz. 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 continues increasing as FFT length increases since the additional hardware to implement our scheme remains almost fixed. *Our* scheme proves to be beneficial for 1 K points onwards, both in terms of area and power. The savings in area range from 48 to 54%, whereas the power savings vary from 0.62 to 59% for longer FFTs (1 K points \geq).

Conclusion: We have presented a coefficient memory reduction technique that is more power and area efficient than existing

approaches in the literature. This reduction is more significant for longer FFTs which are crucial in OFDM applications. The scheme 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.

© IEE 2002

6 November 2001

Electronics Letters Online No: 20020117

DOI: 10.1049/el:20020117

M. Hasan and T. Arslan (Department of Electrical Engineering, The University of Edinburgh, Edinburgh EH9 3JL, United Kingdom)

E-mail: Mohammad.Hasan@ee.ed.ac.uk

References

- 1 COHEN, D.: 'Simplified control of FFT hardware', *IEEE Trans. Acoust. Speech Signal Process.*, 1976, **ASSP-24**, pp. 577-579
- 2 MA, Y., and WANHAMMAR, L.: 'Hardware efficient control of memory addressing for high performance FFT processors', *IEEE Trans. Signal Process.*, 2000, **48**, (3), pp. 917-921
- 3 CHANG, Y., and PARHI, K.K.: 'Efficient FFT implementation using digital-serial arithmetic'. IEEE Workshop on Signal Processing Systems, Taipei, Taiwan, ROC, 1999, pp. 645-653
- 4 MA, Y.: 'An effective memory addressing scheme for FFT processors', *IEEE Trans. Signal Process.*, 1999, **47**, pp. 907-911

High bit rate and programmable multiwavelength generator based on Raman soliton effect in DSF

M. Kato, K. Kurokawa, K. Fujiura, T. Kurihara and K. Okamoto

A wavelength tunable multiwavelength pulse source with clock rate tunability is described. A tuning range of 32 nm and multiwavelength generation were achieved at a 10 GHz clock rate and a quasi-320 GHz OTDM signal. The device consists of a programmable PLC multiplexer and employs amplitude-to-wavelength conversion based on the Raman soliton effect.

Introduction: Widely tunable picosecond pulse sources at a GHz clock rate are emerging as crucial components for ultrafast wavelength routing networks with the ability to provide 1:1 communication and 1:N multicasting. These light sources provide flexibility to WDM applications that support wavelength routers and converters, transponders, optical coding-decoding and optical switching. Multiwavelength generation from a single supercontinuum (SC) source at a GHz level that meets these requirements has been demonstrated [1]. An SC source emits broad white light that depends on the dispersion characteristics of SC fibres; however, for multicasting, it does not always use the generated wavelength most efficiently owing to the inclusion of unnecessary wavelengths. Another multiwavelength pulse generation technique utilising Raman soliton, using the polarisation dependent dispersion difference in birefringent fibre, has recently been demonstrated [2]. Wavelength tunable soliton generation based on Raman soliton effect has already been reported that uses a femtosecond pulse train at a MHz repetition rate in optical fibres [2-5]. In this Letter, we describe a new high bit rate and programmable multiwavelength picosecond pulse generator based on Raman soliton. We have demonstrated a tuning range of 32 nm and multiwavelength generation at a 10 GHz clock rate and a quasi-320 Gbit/s OTDM signal.

Operational principle of generator: Fig. 1a shows the configuration of a programmable multiwavelength generator. We used a 4 ps, 10 GHz fibre laser at 1555 nm as a seed pulse source. The pulse train was coupled into a 1 km-long highly-nonlinear fibre (HNLF) and a 1.1 km-long nonzero dispersion-shifted fibre (NZ-DSF). These fibres enabled us to realise adiabatic soliton compression to 1.1 ps at 300 mW. The compressed pulse train was coupled into a programmable PLC multiplexer consisting of thermo-optic switches [6] as amplitude controllers and splitters, delay lines with a relative delay