

Distributed Arithmetic based Split-Radix FFT

Sunil P. Joshi · Roy Paily

Received: 17 June 2011 / Accepted: 31 May 2013 / Published online: 22 June 2013
© Springer Science+Business Media New York 2013

Abstract In this paper we have designed a Split-radix type FFT unit without using multipliers. All the complex multiplications required for this type of FFT are implemented using Distributed Arithmetic (DA) technique. A method is incorporated to overcome the result overflow problem introduced by DA method. Proposed FFT architecture is implemented in 180 nm CMOS technology at a supply voltage of 1.8 V.

Keywords Fast fourier transforms (FFT) · Split radix FFT (SRFFT) · Distributed arithmetic (DA) · ASIC implementation

1 Introduction

Fast Fourier Transform (FFT) is a very common operation used for various signal processing units. It finds applications in a wide range of communications, radar, image, speech processing and analysis. It is important to have architectures which perform FFT quickly and consume less power especially for wireless standards used in handheld devices. Since, twiddle factors in FFT/IFFT are orthogonal to each other, they are used at receiver/transmitter in an OFDM implementation. According to the European digital video/audio broadcasting (DVB-T/DAB) standards, an OFDM system may require FFT lengths ranging from 256 to 8192 point. Wireless local area network (WLAN) and

HIPERLAN/2 systems require high-speed and low-power FFT/IFFT design [20, 25]. The fourth-generation cellular phone and the forthcoming new WLAN systems may also incorporate OFDM system to deliver higher bandwidth [16]. Hence, it is important to design high-performance but low-power FFT for these applications.

In 1964, Cooley-Tukey [6] used a divide and conquer approach to reduce the computational complexity of FFT from N^2 to $N \log_2 N$. This method is considered as a breakthrough in the development of high speed and low complexity FFT algorithm. Good's mapping [9] is used to divide transforms into two different lengths FFT's N_1 and N_2 , such that $N = N_1 \cdot N_2$, where both N_1 and N_2 are co-primes. Subsequent to this mapping, fast convolution schemes were used by Winograd in 1974 [27], to solve nesting of the various multiplications. This algorithm was known as Winograd Fourier Transform Algorithm (WFTA). It requires less number of multiplication but its structure is complicated and hence takes more time for execution [15, 19]. In Prime Factor Algorithm (PFA) [13], after dividing transform into two different prime length transform, multi dimensional DFT's are calculated by row-column method. This requires more multiplication operations but with fewer additions and has a simpler structure compared to WFTA. High radix algorithms are also developed for efficient calculation of FFT. These algorithms reduces overall arithmetic operations in FFT, but increases the number of operations and complexity of each butterfly. Various implementations are reported with high radix algorithm in [2, 4, 14, 23, 28]. Among them, radix-4 algorithm is very popular due to its lesser complexity. In 1984, P. Duhamel and H. Hollmann [8], presented the split-radix FFT (SRFFT) algorithm. It calculates the even parts using the radix-2 algorithm and the odd parts using the radix-4 algorithm. This mixed-radix approach helped to achieve lower number of multiplications and additions. The

S. P. Joshi · R. Paily (✉)
EEE Department, IIT Guwahati,
Guwahati, Assam, India
e-mail: roypaily@iitg.ernet.in

S. P. Joshi
e-mail: sunil.josh88@gmail.com

resulting butterfly has simple structure and the details will be presented in subsequent sections. Radix 2/4/8 algorithm has less complexity than SRFFT, but it cannot be used for calculating FFT for all powers of 2^n .

Every butterfly has two main operations i.e., complex multiplication and addition. Complex multiplication decides the speed, hardware cost and power consumption. Usually there are three conventional ways to tackle the complex multiplication [17]: Booth-Wallace multiplier, CORDIC multiplier, and CSD multiplier. CORDIC multiplier and Booth multiplier usually have large area cost. Canonical Signed Digit (CSD) multiplier is used by Maharatna and Jagdhold [14]. They have used a shuffled-CSD based structures by putting together all the constants required for multipliers. It is not easy to handle the constant twiddle factors in CSD arithmetic and it results in large area cost. Distributed Arithmetic (DA), along with Modulo Arithmetic, are computation algorithms that perform multiplication with look-up table based schemes. The commonly encountered form of computation in digital signal processing is a sum of products and it can be executed most efficiently by DA. It was invented by S. Zohar in 1968 [32, 33]. In 1974, Abraham Peled and Bede Liu [21], presented the detailed explanation of DA method for IIR digital filter realization. DA is basically a bit-serial computational operation that forms an inner (dot) product of a pair of vectors in a single direct step. The advantage of DA is its efficiency of mechanization [26]. Since twiddle factors in any FFT algorithm are fixed for specific N-point FFT, DA can be used to replace complex multiplication in FFT.

Various DA based multipliers are reported in literature [1–3, 18, 23]. In [11], multiplier is implemented with DA using a guard bit for overflow control. CORDIC based multiplier reported in [28] has achieved overflow control by shifting input data to the right by 2 bits thereby scaling data by 4. In this paper a new methodology is presented for controlling the overflow. The advantage of the proposed method is that it does not require the pre-scaling of input data. This method eliminates the extra shifting operation and associated hardware compared to [28]. Since the data stored in ROM is pre-scaled by 2, an additional one bit shift is required after finishing multiplication in order to get back the original result as in [28]. To achieve better precision, we have incorporated same word length for both operands of multiplier.

Apart from the overflow consideration of DA based multipliers, another contribution of this paper is the implementation of a 256-point Split-Radix FFT (SRFFT) using these multipliers. A DA based SRFFT implementation has not been reported in the literature so far. Combining the advantages of SRFFT algorithm and DA based multiplication, the resulting architecture is expected to be efficient in terms of area, power and speed. There have been reports on

various DA based FFT architectures in literature. DA has been used to reduce the complexity and to improve the performance of basic radix-2 FFT structures [14, 23, 28]. In [22] and [24], prime factor fourier transform algorithm was implemented. In [22], basic arithmetic operation in PFFT butterfly is replaced with DA. In [24], the matrix equation of the DFT is reordered using DA. In [12] and [17], radix-8 FFT butterfly is used to implement 8K and 64 point FFT. In [17], a DA based twiddle multiplier is implemented for butterfly operation. However the issues related to the overflow of result was not addressed. In [12], DA is used to implement the CORDIC-based complex multipliers for 8-point FFTs. In [23], radix-2 FFT butterfly is implemented using DA based complex multiplication.

As a summary, this paper addresses the implementation aspects DA based multiplier and 256-point SRFFT. This paper is further arranged as follows. Section 2 explains the split radix algorithm and the corresponding butterfly diagram briefly. Section 3 presents how a complex multiplication can be substituted with DA operations. Sections 4 and 5 gives detailed architectures for DA based complex multiplier and pipelined SRFFT. Results are compared with other architectures in Section 6 and finally, the paper is concluded.

2 Split-Radix FFT Algorithm

This section presents briefly the SRFFT algorithm and its butterfly structure. General equation of discrete fourier transform is given as

$$X(k) = \sum_{n=1}^{N-1} x_n W_N^{nk} \quad (1)$$

where $W_N^{nk} = e^{j2\pi nk/N}$ Radix-2 algorithm calculates odd and even components of $X(k)$ using decimation in frequency. Corresponding equations for even and odd components are given by the equations

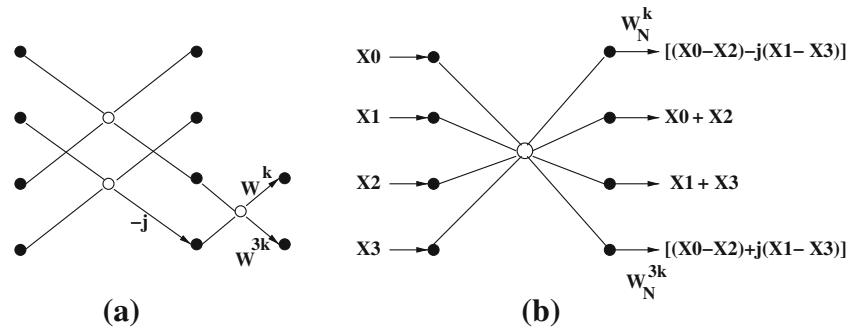
$$X_{2k} = \sum_{n=1}^{(N/2)-1} (x_n + x_{n+(N/2)}) W_{N/2}^{nk} \quad (2)$$

$$X_{2k+1} = \sum_{n=1}^{(N/2)-1} (x_n - x_{n+(N/2)}) W_N^n \cdot W_{N/2}^{nk} \quad (3)$$

Split-radix algorithm given in [8] further decomposes odd ones and calculates X_{4k+1} and X_{4k+3} . Corresponding equations for even and odd parts for an SRFFT are given in (4) to (6).

$$X_{2k} = \sum_{n=1}^{(N/2)-1} (x_n + x_{n+(N/2)}) W_{N/2}^{nk} \quad (4)$$

Figure 1 a SRFFT butterfly, b SRFFT butterfly with permuted outputs.



$$X_{4k+1} = \sum_{n=0}^{(N/4)-1} (x_n - j \cdot x_{n+(N/4)} - x_{n+(N/2)} + j \cdot x_{n+(3N/4)}) W_N^n \cdot W_{N/4}^{nk} \quad (5)$$

$$X_{4k+3} = \sum_{n=0}^{(N/4)-1} (x_n + j \cdot x_{n+(N/4)} - x_{n+(N/2)} - j \cdot x_{n+(3N/4)}) W_N^{3n} \cdot W_{N/4}^{nk} \quad (6)$$

Calculation of $X(k)$ from above equations leads to two different types of butterfly diagrams as shown in Fig. 1 [7].

The difference between part (a) and (b) is in the number of stages required to finish the butterfly operation. Compared to part (a), part (b) completes the operation in a single step and is also a symmetrical structure from a hardware point of view. Complete signal flow graph (SFG) for an 8 point FFT is shown in Fig. 2. The number of stages required for N -point FFT is $\log_2 N$. Each processing element (PE) shown in Fig. 2 has four complex inputs and generates four complex outputs. The detailed diagram of a PE is shown in Fig. 3. It consists of 3 complex adders and 3 subtractors. Similar SFG is used for implementing the proposed 256-point FFT.

3 Distributed Arithmetic Method for Complex Multiplication

In this section, complex multiplication operation using DA is explained. DA can be used to implement multiplication operation if either the multiplicand or the multiplier value is fixed. It stores the possible combinations of fixed operand in ROM and suitable combination is added and shifted with respect to bits of other operand. The method for DA based complex multiplication can be summarized as

$$Z_R + jZ_I = (B_R + jB_I) * (T_R + jT_I) \quad (7)$$

$$Z_R = B_R T_R - B_I T_I \quad (8)$$

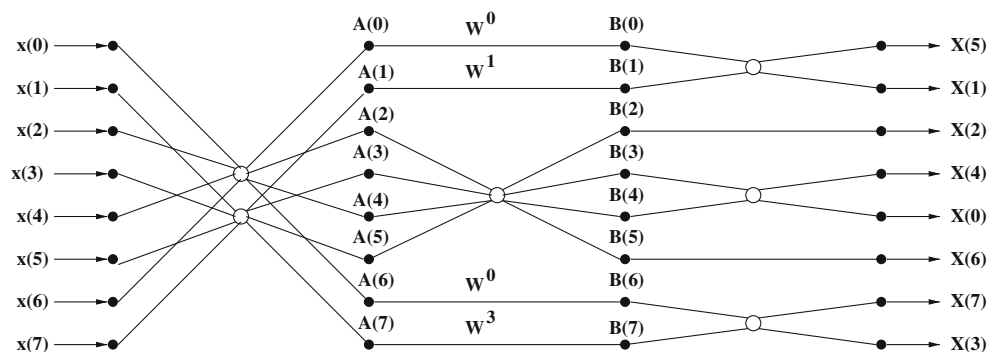
$$Z_I = B_R T_I + B_I T_R \quad (9)$$

(6) and (9) show that 4 real multiplications and 2 real additions are required to compute Z_R and Z_I . But these equations can be considered as one ‘multiply and accumulate’ operation as explained in [1].

$$y = \sum_{k=1}^K C_k x_k \quad (10)$$

Let, C_k are fixed coefficients and x_k are the input words. If x_k is M -bit fractional number in 2’s complement form then

Figure 2 Signal flow graph of 8-point SRFFT.



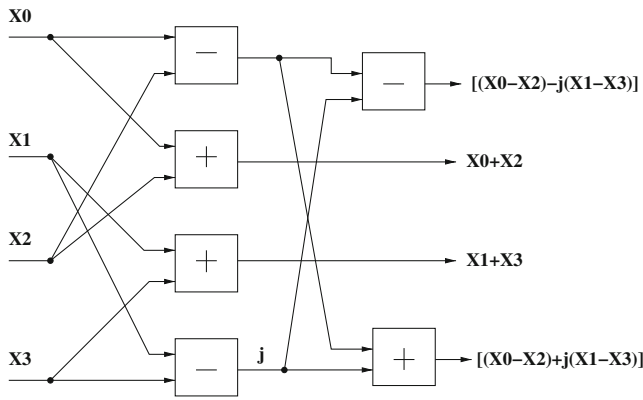


Figure 3 Processing element for butterfly.

it can be expressed in following form

$$x_k = -b_{k0} + \sum_{m=1}^{M-1} b_{km}2^{-m} \tag{11}$$

where b_{km} are binary bits which are either 0 or 1. Combining (10) and (11),

$$y = \sum_{k=1}^K \left(-b_{k0} + \sum_{m=1}^{M-1} b_{km}2^{-m} \right) C_k \tag{12}$$

it can be rearranged as

$$y = -\sum_{k=1}^K b_{k0}C_k + \sum_{m=1}^{M-1} \left[\sum_{k=1}^K C_k b_{km}2^{-m} \right] \tag{13}$$

In (13), the part

$$\sum_{k=1}^K C_k b_{km} \tag{14}$$

Figure 4 DA based complex multiplier.

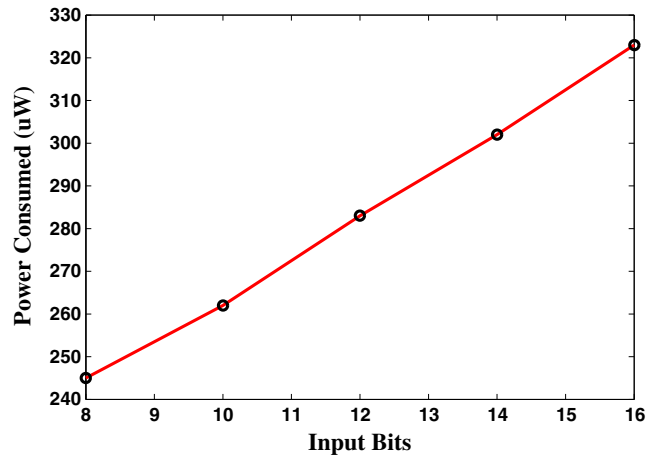
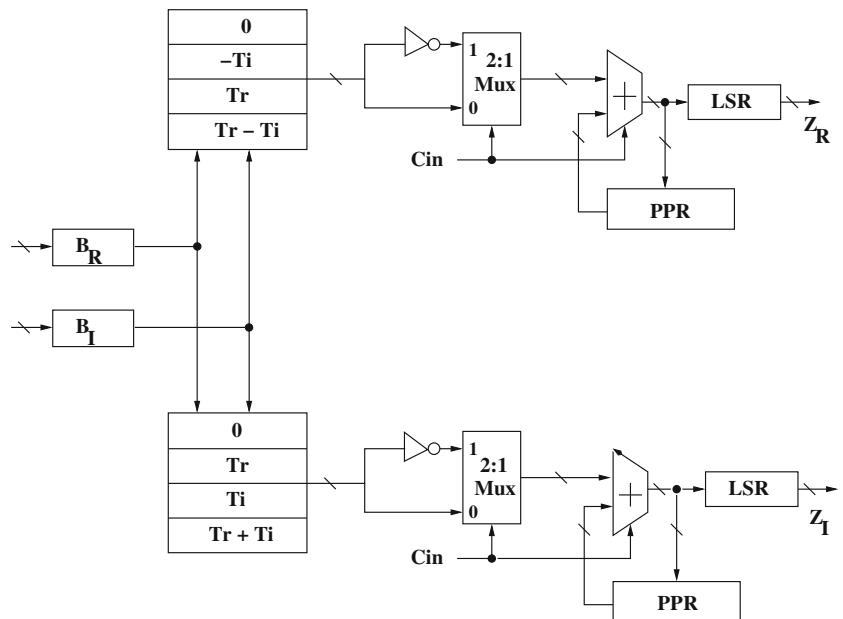


Figure 5 Power vs input bits.

is very important. Depending on binary bits b_{km} of K input words, the fixed coefficients C_k are added together. The two twiddle factors T_R and T_I of (8) and (9) are constant and B_R and B_I are two incoming words. Depending on the bits of B_R and B_I , twiddle factors are added. There can be only four combination of bits and the precalculated values for these combinations can be stored in ROM for both Z_R and Z_I . These incoming binary bits can be used to address ROM locations and ROM output is fed to adder.

4 Proposed Architecture Of DA Based Complex Multiplier

The detailed architecture for complex multiplier is shown in Fig. 4. The real and imaginary parts of incoming words, B_R

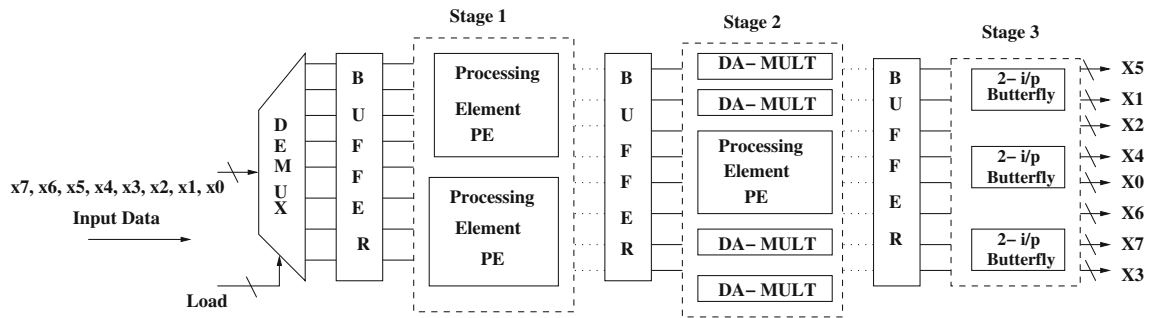


Figure 6 The block level representation of proposed SRFFT.

and B_I are stored in two 16 bits wide parallel in serial out register. Shifting is carried out starting from LSB to MSB. Each output bit of these two registers are used as address lines of the ROMs. The ROM stores precalculated outcomes for both Z_R and Z_I as per (8) and (9). The size of each ROM is 4×16 . One of the input to the 2:1 MUX is directly fed from the output of ROM and the other input to MUX is inverted. Input and output bit width for MUX is also 16 bits. The select line of MUX is ‘cin’ signal and it remains as ‘0’ till the MSB arrives at output. As explained in (13), for a 2’s complement fractional number, the MSB is sign bit and its value is required to perform subtraction at the end. If select line ‘cin’ of Mux is 1, it selects inverted output from ROM and it is added to the value stored in the partial product register (PPR). The PPR is a 16 bit wide ‘parallel in parallel out’ register which also performs 1-bit right shift operation. Finally the output is taken from the left shift register. Both T_R and T_I are in 16 bit 2’s complement fractional number format and the ROM for Z_R contains $T_R + T_I$. This addition may cause an overflow and in order to solve this issue, all twiddle factors are pre-scaled and stored in ROM. Pre-scaling is achieved by shifting one bit to the right and thereby the overflow is avoided in all possible cases. Nevertheless, to ensure proper result from complex multiplier, the result of the final addition should be shifted by one bit

to the left. Since, the combinational delay for multiplier is given by $t_{mult} = t_{read} + t_{MUX} + t_{add}$, the latency of this multiplier is 17 cycles which is equal to bit width of incoming words and additional shift is required to for up scaling the final result. Thus total time required for multiplication is $t_{max} = 17 * t_{mult}$.

A frequently stated disadvantage of DA is its apparent slowness because of the inherent bit-serial nature [26]. The total number of cycles required to complete the multiplication is proportional to the input bits. Bit widths also determine the power consumed in the multiplier, therefore we have investigated the power consumption for different input bits. Figure 5 shows that the power consumption increases almost linearly with approximately $10\mu W$ increase in power per every bit.

5 Proposed SRFFT Architecture

Figure 6 shows architecture of proposed SRFFT unit. Due to space constraints, an 8-point FFT is shown instead of 256-point. It consists of 3 stages with buffers between the states. Each stage has PE and DA based complex multipliers. The implemented FFT receives all inputs in serial fashion and therefore a ‘demux’ is needed before ‘stage1’.

Figure 7 Detailed view of processing element.

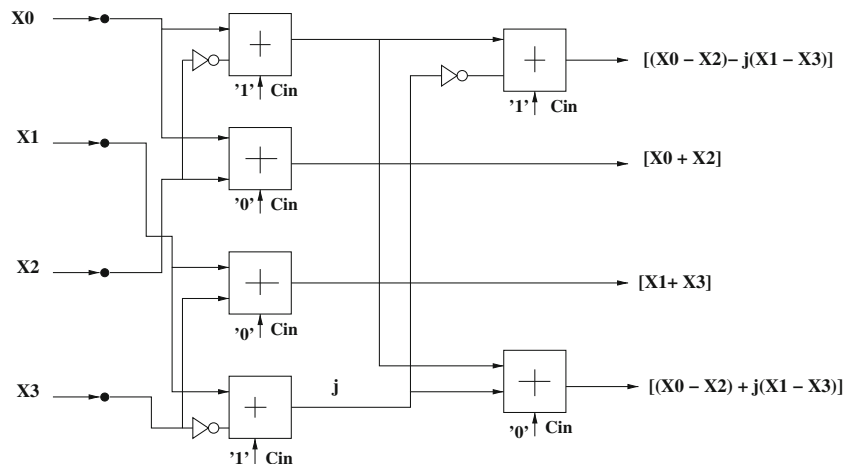


Table 1 Timing and power comparison of Multiplier with other implementations.

Parameters	Power (mW)	Area (mm ²)	Delay (ns)	Technology (μm)	Frequency (MHz)	Bit Width (Width)
Proposed	0.323	0.0127	8	0.18	125	16 × 16
[18]	70	0.5	4	0.8	250	12 × 16
[1]	66	3.59	16	0.5	40	16 × 10

Since the proposed FFT is of 256 points, a bit width of 16 is used for ROM to achieve more precision and the proposed architecture has 16 × 16 bit width. In the last stage, a 2 input butterfly unit is needed as in Cooley-Tukey's FFT algorithm. It performs complex addition and complex subtraction operation of two inputs. Since there are buffers between stages, the maximum operating frequency of operation of FFT depends on the speeds of individual stages. The detailed arrangement of adders in PE is shown in Fig. 7. Each adder and inverter shown in figure represents operation for both real and imaginary parts and therefore there are total 12 adders and 6 inverters. The width of input/output adders as well as inverters is 16 bits. The total delay of PE can be given as $t_{PE} = 2t_{inv} + 2t_{add}$. Thus total delay for whole SRFFT can be either $td1 = t_{setup} + t_{PE}$ or $td2 = t_{mul} + t_{setup}$ whichever is greater. Typically setup delays are negligible compared to multiplier delays and PE delays and therefore SRFFT can be operated at multiplier frequency.

6 Experimental Results

In this section, results of the DA based multiplier are presented. Subsequently for results for SRFFT using DA based multiplier are given. DA based multiplier is implemented in 0.18 μm CMOS technology. The functional verification of HDL coded multiplier is carried out in Xilinx ISE simulator. The proposed multiplier is coded in Verilog language. Synthesis of multiplier is carried out in design compiler using

Faraday library. The area taken by design is 0.012752 mm². The delay of design is found to be 8 ns. Power of multiplier is calculated by generating SAIF (Switching Activity Interchange Format) in design compiler. This file is used in VCS, along with main design of multiplier and testbench to calculate switching at every specified node. Information of these switching activities are stored in backward 'saif' file. This file is again used in design compiler along with generated netlist and the power is determined. The clock speed used for power estimation is 125 MHz and voltage used is 1.8 V. The power estimated for 125 MHz operation is 323 μW. Since the multiplier proposed is basically an extension of the DA technique, we have compared the results obtained with other DA based implementations in Table 1. The proposed multiplier has achieved the least power required at higher bit widths compared to other DA based multipliers reported in [1] and [18].

The Proposed SRFFT architecture is implemented for 256 point FFT in 0.18 μm CMOS technology operating at 1.8 V. The core area taken by design is 11 mm². The critical path delay of SRFFT is found to be 8 ns which is same as that of the multiplier. When the SRFFT is operated at a clock speed of 20 MHz, the power consumed is 70 mW. Seventeen clock cycles are required to complete one multiplier operation and there are six stages of multiplication and therefore it takes 102 cycles. Eight extra cycles are required to process the data through the 8 buffers. Therefore the proposed architecture calculates 256 pt FFT in 110 cycles and the total time required for the calculation is 5.5 μs. The simulation results showed that the power consumed is 21 mW and 401 mW when the SRFFT is operated at clock speeds of 5 MHz and 100 MHz respectively. The comparison of proposed SRFFT unit with other implementations is carried out in Table 2. The SRFFT operated at 5 MHz clock has achieved the lowest power consumption even at higher bit widths when compared to other reported FFTs. Therefore the proposed design can be used for WLAN applications. The layout of proposed architecture with Input Output (IO) pads, power pads and bond pads is given in Fig. 8. Layout is

Table 2 Timing and power comparison of Proposed SRFFT with other implementations.

Parameters	FFT Size	Algorithm	Power (mW)	Area (mm ²)	Delay (us)	Technology (μm)	Frequency (MHz)	Bit Width
Proposed	256	SRFFT	70	11	5.5	0.18	20	16 × 16
Proposed	256	SRFFT	21	11	22	0.18	5	16 × 16
Proposed	256	SRFFT	401	11	1.1	0.18	100	16 × 16
[30]	64	SRFFT	507	3.45	–	0.35	100	12 × 12
[29]	8K	Radix 2/4	190	–	1197	0.25	12	8 × 8
[10]	2K/4K/8K	Radix 2/4/8	25.8	2.84	1025	0.18	20	13 × 10
[31]	2K	Radix 2/4/8	640	21.45	22.5	0.35	45	12 × 16
[5]	256	Radix 16	162.7	–	–	0.18	33	10 × 10

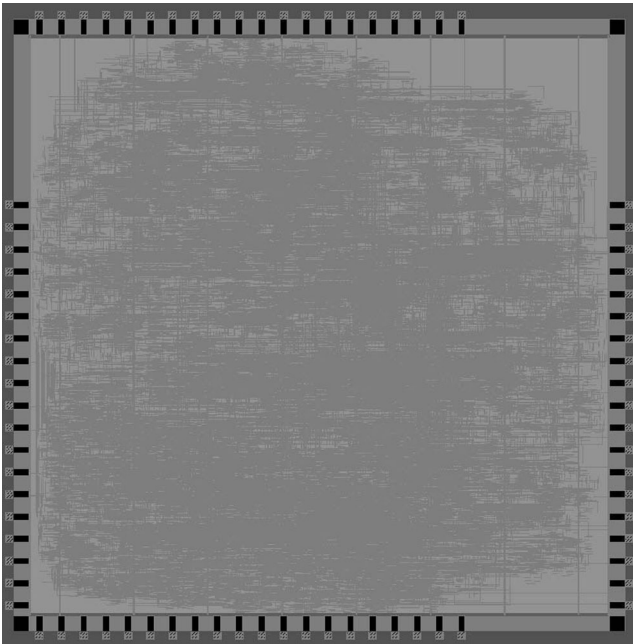


Figure 8 Layout of DA based SRFFT.

carried out in Cadence using SOC Encounter. The chip area required for this design is 5 mm × 3 mm.

7 Conclusion

In this paper, a new overflow technique was proposed for the DA based complex multiplier. The proposed architecture avoided the pre-scaling of input data; instead, it pre-scaled the data stored in ROM. Thus additional hardware required for pre-scaling input data was removed. When compared with other DA based multipliers, the proposed multiplier has achieved lower power and area at higher bit widths. The same multiplier was used to implement a pipelined architecture for 256 point SRFFT. To the best of our knowledge there are no reports available on DA based SRFFT implementation. Pipelining technique was applied to maintain the speed of individual stages of SRFFT equal to that of the multiplier. Thereby SRFFT can be operated at the speed of the multiplier with a latency of 110 cycles. The SRFFT operated at 5 MHz clock has achieved the lowest power consumption even at higher bit widths when compared to other reported FFTs. Therefore the proposed design is highly desirable for WLAN applications.

Acknowledgments This work is carried out using the Synopsys and the Cadence tools provided by the SMDP II project at IIT Guwahati.

References

- Berkeman, A., Öwall, V., Torkelson, M. (2000). A low logic depth complex multiplier using distributed arithmetic. *IEEE Journal of Solid-State Circuits*, 35, 656–659.
- Perez-Pascual, A., Sanaloni, T., Valls, J. (2002). FPGA-based radix-4 butterflies for HIPERLAN/2. In *IEEE international symposium on circuits and systems* (pp. 277–280). Scottsdale.
- Chang, Y.N., & Parhi, K.K. (2000). High-performance digit-serial complex multiplier. *IEEE Transactions on Circuits and Systems*, 47, 570–572.
- Chen, T., Sunanda, T.G., Jin, J. (1999). COBRA: a 100-MOPS single-chip programmable and expandable FFT. *IEEE Transactions Very Large Scale Integrative (VLSI) Systems*, 7, 174–182.
- Fan, C.-P., Lee, M.-S., Su, G.-A. (2006). A low multiplier and multiplication cost 256-point FFT implementation with simplified radix-16 SDF architecture. In *IEEE asia pacific conference on circuits and systems* (pp. 1935–1938).
- Cooley, J., & Tukey, J. (1965). An algorithm for the machine calculation of complex fourier series. *Mathematical Comparative*, 19, 297–301.
- Duhamel, P., & Hollmann, H. (1985). Implementation of split-radix FFT algorithms for complex, real and real-symmetric data. *IEEE International Conference on Acoustics, Speech and Signal Processing*, 10, 784–787.
- Duhamel, P., & Hollmann, H. (1984). Split radix FFT algorithm. *Electronics Letters*, 20, 14–16.
- Good, I. (1958). The interaction algorithm and practical fourier analysis. *Journal of the Royal Statistical Society, B-20*, 361–372.
- Xiao, H., Pan, A., Chen, Y., Zeng, X. (2008). Low-cost reconfigurable VLSI architecture for fast fourier transform. *IEEE Transactions on Consumer Electronics*, 54, 1617–1622.
- He, S., & Torkelson, T. (1995). A pipelined bit-serial complex multiplier using distributed arithmetic. *International Symposium on Circuits and Systems*, 3, 2313–2316.
- Jiang, R.M. (2007). An area-efficient FFT architecture for OFDM digital video broadcasting. *IEEE Transactions on Consumer Electronics*, 53, 1322–1326.
- Kolba, D., & Parks, T. (1977). A prime factor algorithm using high-speed convolution. *IEEE Transaction Acoustics Speech Signal Process*, ASSP-25, 281–294.
- Maharatna, G., & Jagdhold, U. (2004). A 64-point fourier transform chip for high-speed wireless LAN application using OFDM. *IEEE Journal Solid-State Circuits*, 39, 484–493.
- Morris, L. (1978). A comparative study of time efficient FFT and WFTA programs for general purpose computers. *IEEE Transaction Acoustics Speech Signal Process*, ASSP-26, 141–150.
- Engels, M., Eberle, W., Gyselinckx, B. (1998). Design of a 100 Mbps wireless local area network. In *Proceedings IEEE URSI international symposium signals and systems* (pp. 253–256).
- Jiang, M., Yang, B., Huang, R., Zhang, T.Y., Wang, Y.Y. (2007). Multiplierless fast fourier transform architecture. *Electronics Letters*, 43, 191–192.
- Karlsson, M., Vesterbacka, M., Wanhammar, L. (1997). *Design and implementation of a complex multiplier using distributed arithmetic* (pp. 222–231).
- Nawab, H., & McClellan, J. (1979). Bounds on the minimum number of data transfers in WFTA and FFT programs. *IEEE Transaction Acoustics Speech Signal Process*, ASSP-27, 394–398.
- van Nee, R., & Prasad, R. (2000). *OFDM for wireless multimedia communications*. Norwell: Archtech House.

21. Peled, A., & Liu, B. (1973). A new approach to the realization of nonrecursive digital filters. *IEEE Transaction Audio and Electroacoustics*, 21, 477–485.
22. Chow, P., Vranesic, Z., Yen, J.L. (1983). A pipelined distributed arithmetic PFFT processor. *IEEE Transactions on Computers*, 32, 198–206.
23. Shaditalab, M., Bois, G., Sawan, M. (1998). Self-sorting radix-2 FFT on FPGAs using parallel pipelined distributed arithmetic blocks. In *IEEE symposium on FPGAs for custom computing machines* (pp. 137–138).
24. Siu, W., & Chen, C. (1983). New realization technique of high-speed discrete fourier transform described by distributed arithmetic. *IEE Proceedings*, 130, 177–181.
25. Weste, N., & Skellern, D.J. (1998). VLSI for OFDM. *IEEE Communication Magazine*, 36, 127–131.
26. White, S.A. (1989). Applications of distributed arithmetic to digital signal processing: a tutorial review. *IEEE ASSP Magazine*, 6, 4–19.
27. Winograd, S. (1976). On computing the discrete fourier transform. *Proceedings of the National Academy of Sciences of the United States of America*, 73, 1005–1006.
28. Hui, W., Ding, T., McCanny, V. (1996). A 64-point fourier transform chip for video motion compensation using phase correlation. *IEEE Journal of Solid-State Circuits*, 31, 1751–1761.
29. Li, X., Lai, Z., Cui, J. (2007). A low power and small area FFT processor for OFDM demodulator. *IEEE Transactions on Consumer Electronics*, 53, 274–277.
30. Yeh, W.C., & Jen, C.W. (2003). High-speed and low-power split-radix FFT. *IEEE Transactions on Signal Processing*, 51, 864–874.
31. Lin, Y.T., Tsai, P.Y., Chiueh, T.D. (2005). Low-power variable-length fast fourier transform processor. *IEEE Proceedings Computers and Digital Techniques*, 152, 499–506.
32. Zohar, S. (1973). The counting recursive digital filter. *IEEE Transaction on Computers*, C-22, 338–347.
33. Zohar, S. (1973). New hardware realization of nonrecursive digital filters. *IEEE Transaction on Computers*, C-22, 328–338.



Sunil Joshi received his B.E degree in Electronics Engineering from Nagpur University, India in 2009. He obtained the M.Tech. from the Department of Electronics and Electrical Engineering, Indian Institute of Technology Guwahati in 2011. His area of interest is VLSI system design.



Roy P. Paily received the B.Tech degree in Electronics and Communication Engineering from College of Engineering, Trivandrum, India in 1990. He obtained the M.Tech. and Ph.D. from Indian Institute of Technology, Kanpur and Indian Institute of Technology, Madras in 1996 and 2004, respectively, in the area of Semiconductor Devices. Presently he is working as an Associate Professor in the Department of Electronics and Electrical Engineering, Indian Institute of Technology, Guwahati where he joined in 2004 as an Assistant Professor. His current research interests are VLSI Circuits/System Design and MEMS.