International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

IJIEMR Transactions, online available on 12th Nov 2018. Link :http://www.ijiemr.org/downloads.php?vol=Volume-07&issue=ISSUE-12

Title: **RADIX-2 FFT ARCHITECTURE TO PROCESS TWIN DATA STREAMS FOR MIMO USING A NORMAL IO ORDER**

Volume 07, Issue 12, Pages: 156–163.

Paper Authors

M.SAILA BHANU, T.GANGADHARARAO

V S M College of Engineering, Ramachandrapuram, A.P

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

RADIX-2 FFT ARCHITECTURE TO PROCESS TWIN DATA STREAMS FOR MIMO USING A NORMAL IO ORDER

[1]M.SAILA BHANU, [2]T.GANGADHARARAO
[1]Assistant professor, Dept. of ECE, V S M College of Engineering, Ramachandrapuram, A.P
[1]Assistant Professor, Dept. of ECE, V S M College of Engineering, Ramachandrapuram, A.P

Abstract. The primary objective of this project is to design a multi-way FFT reversal switch that has the probability of processing the paired data stream. Fast Fourier Transform has become ubiquitous in engineering applications. Now FFT is most widely used in many communication and signal processing systems. This project presents a proposed FFT pipeline processor for the FFT calculation of two independent data streams. The FFT architecture of the multipath delay switch is used in the proposed architecture to process an N / 2 point decimation in FFT time and an N / 2 point decimation in frequency FFT operations of odd and even samples of two Data separately to reduce the area and high throughput to achieve the best performance of the architecture using a modified booth algorithm to minimise power.

INTRODUCTION:

The Fast Fourier Transform (FFT) is one of the prominent algorithms in the digital signal processing domain. It is used to compute the discrete Fourier transform (DFT) efficiently. In order to meet the high and real-time demands of modern applications, hardware designers have always tried to implement efficient architectures for FFT calculation. In this context, pipelined hardware architectures are used to simplify the FFT operations. Transmitter operations perform FFT operations. DFT operations perform receiver operations. For the operating speed on architecture to process both DIT and DIF simultaneously. the transmitter side to implement this new A real-time pipeline FFT processor operates as a function of the single path delay feedback [1], a hardware-oriented rdix-22 algorithm is derived by integrating a twiddle factor decomposition technique into the division and conquest. The multi-mode multipath delay feedback architecture based on dynamic voltage The scaling program (DVFS) [2] uses to process the FFT processor for OFDM MIMO applications. To achieve the high throughput of the multi-mode FFT processor with flexible-variable delay (FRCMDF) feedback structure (FRCMDF) [3] for WLAN, WPAN, WMAN applications. [2] - [4] The architectures of the FFT processor do not have specific bit reversal circuits. The four independent data streams work independently using the MDC technique, but the outputs have not come parallel. [6] in these decimated dual channel delay transfer data switch units and integrated bit sequence

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

converter, returns the various radixes to process the pipeline FFT operations. [7] In this paper folding technique and register minimization techniques are used elaborate the pipelined parallel FFT operations.For different radixes, the series of data bit reversal circuits have been discussed in [8]. In this [9], the MDC and MDF methods with the organisation of the memory bank of the parallel circuit with reversed bits used. [10] Advanced architectures are used in this reorganisationThe circuit is applicable for a specific order. [11] Here SDC-SDF architectures combined to obtain only serial data transmission.

Idea of the working methodology:

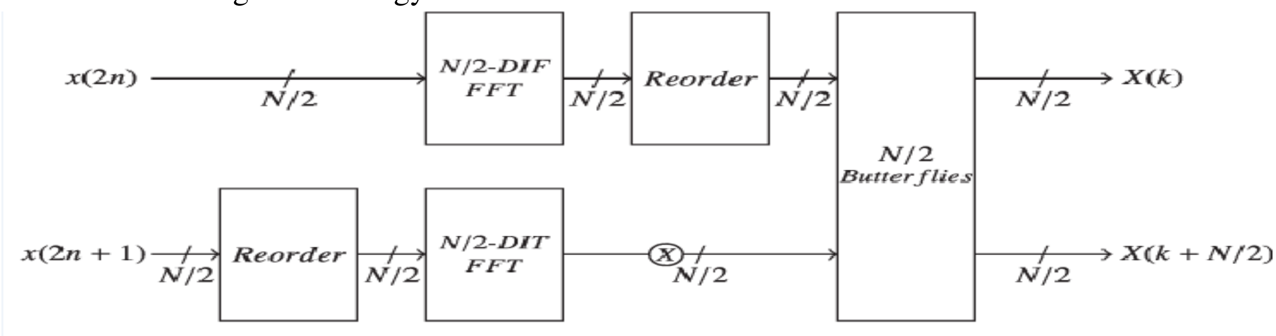


Fig. 1idea of the working methodology

The reorganisation blocks of FIG. 1 shows the odd samples N / 2 (x (2n + 1)) are rearranged before FIT DIT operation N / 2 point, and the even samples N/2 (x (2)) are rearranged after the DIF N / 2 points FFT operation. In order to calculate the N-point DIT FFT from the outputs of two N / 2-point FFT, an additional step of the butterfly operations are performed on the results of the two FFTs. Thus, the outputs generated by an additional butterfly stage are in the natural order.

TABLE I
DATA FLOW THROUGH DIFFERENT LEVELS

Level	Time —>						
L_1	$E_1(1,1)$	$O_1(1,1)$	$E_1(1,2)$	$O_1(1,2)$			
L_2		$E_1(2,1)$	$E_2(2,1)$	$E_1(2,2)$	$E_2(2,2)$		
L_3			$E_1(3,1)$	$O_1(3,1)$	$E_1(3,2)$	$O_1(3,2)$	
M_1		$E_2(1,1)$	$O_2(1,1)$	$E_2(1,2)$	$O_2(1,2)$		
M_2			$O_1(2,1)$	$O_2(2,1)$	$O_1(2,2)$	$O_2(2,2)$	
M_3				$E_2(3,1)$	$O_2(3,1)$	$E_2(3,2)$	$O_2(3,2)$

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

Table 1 shows the L1, L2, L3, M1, M2, M3 here LI, M1 are two N / 2 bits of separation of the odd rearranged bits and equal data and L2 and M2 are butterfly operations performed and L3 and M3 is the last steps of the butterfly operation to rearrange the even bits to obtain outputs from normal order.

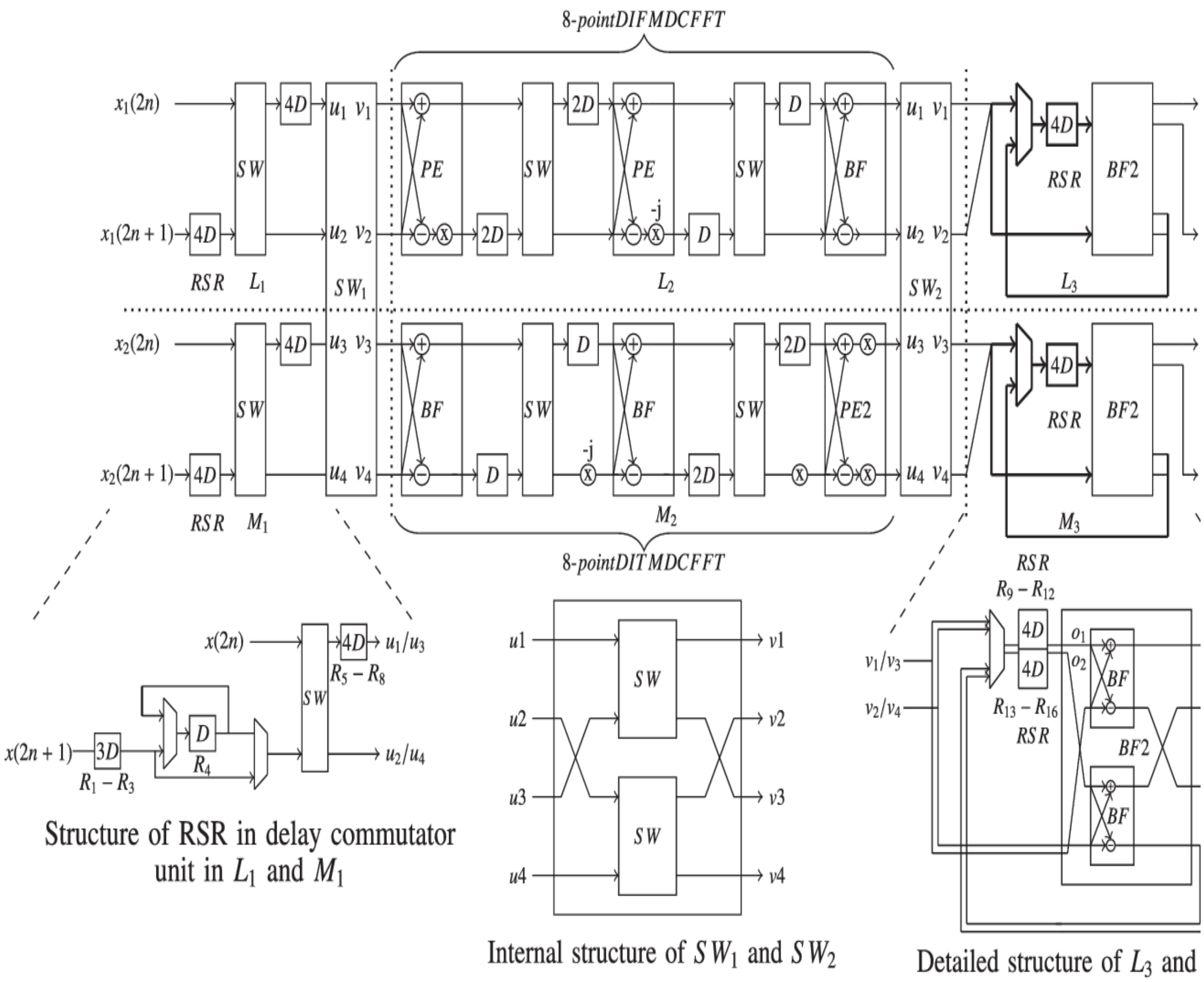


Fig2:16point radix-2 FFT architecture

The FFT architecture in the figure above is divided into six levels (L1, L2, L3, M1, M2 and M3). The registers RSR in the levels L1 and M1 reorder the odd input data and the registers RSR in the levels L3 and M3 reorder the partially processed uniform data. FIFT DIF and DIT eight-point operations are performed in L2 and M2, respectively. Data from L1 and M1 can be transmitted to L2 and M2, respectively, or vice versa using SW1. Similarly, data from L2 and M2 can be transmitted to L3 and M3, respectively, or vice versa using SW2. SW1 and SW2 have two switches (SW) for exchanging the data path and propagating the data at different levels. During the normal mode,

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

the switches (SW1 or SW2) pass the data to u1, u2, u3 and u4 to v1, v2, v3 and v4, respectively. However, during the swap mode, the switches (SW1 or SW2) pass the data to u1, u2, u3 and u4 to v3, v4, v1 and v2, respectively. SW1 is in swap mode during the first N / 2 clock cycles and is in normal mode during N / 2 + 1 to N. On the other hand, SW2 is in normal mode during the first clock cycles N / 2 and It is in swap mode during N / 2 + 1 to N. Thus, SW1 and and these switch control signals exchange at each N / 2 clock cycle.

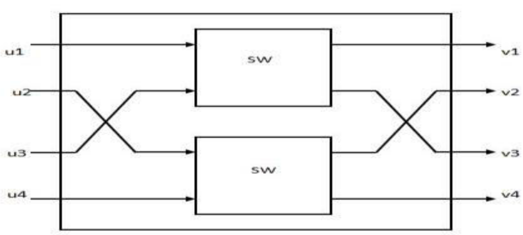


Fig3: Internal structure of SW1 and SW2

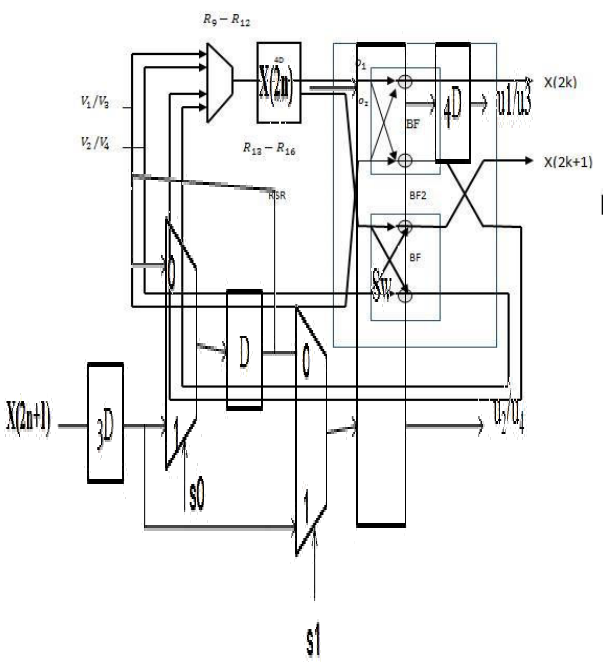


Fig4:Detailed structure of L3 and M3

SW2 are in different mode at all times and change mode for all N / 2 clock cycles. There is a data transition between L y and L y + 1 or My and My + 1 (where y may be 1 or 2), the switches (SW1 or SW2) are in normal mode, and if there is Transition Between L y and My + 1 or My and L y + 1, then the switches (SW1 or SW2) are in swap mode. Like other control signals in the design, the control signals at switches SW1 and SW2 are externally supplied

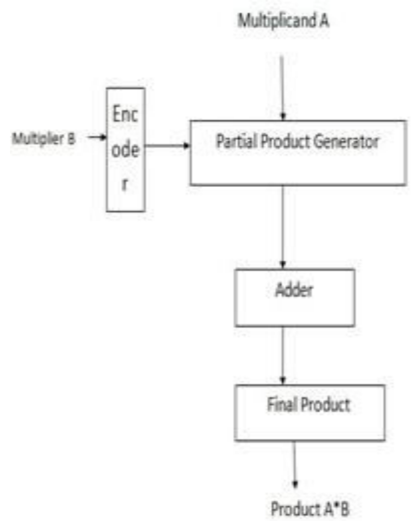


Fig5: Structure ofRSR in delay commutator unit in L1 and M1

The proposed architecture is inspired by the architecture in [7] where the N / 2 data planning records before the first butterfly unit are used to separate the odd samples from the even samples and delay them to generate x (n) and x (N + N / 2) in parallel. In the proposed architecture, these data planning registers are reused to retrieve inverted strong samples. Similarly, N / 2 data planning registers are used before the last butterfly unit to store partially processed even samples until odd samples arrive in [7] and here, these registers are reused to

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

reverse Partially processed partial samples (DFT FFT outputs). In [8], circuits that use multiplexers and shift registers for bit inversion are proposed. If N is the equal power of r, then the number of registers required to invert the data N is (√N - 1) 2. If N is the odd power of r then the number of registers required for Inverting the bit N is (√ r N - 1) (√N / r - 1), where r is the basis of the FFT algorithm. In the proposed architecture, these bit inversion circuits are incorporated into the data planning register to perform a dual role.

PROPOSED ARCHITECTURE

RADIX-8 MODIFIED BOOTH ALGORITHM: The Booth algorithm consists of repeatedly adding one of two predetermined values to a product P and then performing an arithmetic shift to the right on P.The multiplier architecture consists of two architectures, i.e., Modified Booth. By the study of different multiplier architectures, we find that Modified Booth increases the speed because it reduces the partial products by half. Also, the delay in the multiplier can be reduced by using Wallace tree. The energy consumption of the Wallace Tree Multiplier is also lower than the Booth and the array. The characteristics of the two multipliers can be combined to produce a high-speed and low-power multiplier. The modified stand-alone multiplier consists of a modified recorder (MBR). MBR has two parts, i.e., Booth Encoder (BE) and Booth Selector (BS). The operation of BE is to decode the multiplier signal, and the output is used by BS to produce the partial product. Then, the partial products are added to the Wallace tree

TABLE II

BIT REVERSAL OPERATION IN THE LEVELS L_1 AND M_1

Clk	$x(2n)$	$x(2n+1)$	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	u_3	u_4
0	x(0)	x(1)	-	-	-	-	-	-	-	-	-	-
1	x(2)	x(3)	x(1)	-	-	-	x(0)	-	-	-	-	-
2	x(4)	x(5)	x(3)	x(1)	-	-	x(2)	x(0)	-	-	-	-
3	x(6)	x(7)	x(5)	x(3)	x(1)	-	x(4)	x(2)	x(0)	-	-	-
4	x(8)	x(9)	x(7)	x(5)	x(3)	x(1)	x(6)	x(4)	x(2)	x(0)	x(0)	x(8)
5	x(10)	x(11)	x(9)	x(7)	x(5)	x(3)	x(1)	x(6)	x(4)	x(2)	x(2)	x(10)
6	x(12)	x(13)	x(11)	x(9)	x(7)	x(3)	x(5)	x(1)	x(6)	x(4)	x(4)	x(12)
7	x(14)	x(15)	x(13)	x(11)	x(9)	x(7)	x(3)	x(5)	x(1)	x(6)	x(6)	x(14)
8	-	-	x(15)	x(13)	x(11)	x(9)	x(7)	x(3)	x(5)	x(1)	x(1)	x(9)
9	-	-	-	x(15)	x(13)	x(11)	-	x(7)	x(3)	x(5)	x(5)	x(13)
10	-	-	-	-	x(15)	x(11)	-	-	x(7)	x(3)	x(3)	x(11)
11	-	-	-	-	-	x(15)	-	-	-	x(7)	x(7)	x(15)

TABLE III

BIT REVERSAL OPERATION IN THE LEVELS L_3 AND M_3

Clk	v_3	v_4	R_9	R_{10}	R_{11}	R_{12}	R_{13}	R_{14}	R_{15}	R_{16}	o_1	o_2
0	X(0)	X(4)	-	-	-	-	-	-	-	-	-	-
1	X(1)	X(5)	X(0)	-	-	-	X(4)	-	-	-	-	-
2	X(2)	X(6)	X(1)	X(0)	-	-	X(5)	X(4)	-	-	-	-
3	X(3)	X(7)	X(2)	X(1)	X(0)	-	X(6)	X(5)	X(4)	-	-	-
4	-	-	X(3)	X(2)	X(1)	X(0)	X(7)	X(6)	X(5)	X(4)	X(0)	X(8)
5	-	-	-	X(3)	X(2)	X(1)	-	X(7)	X(6)	X(5)	X(4)	X(12)
6	-	-	-	-	X(3)	X(2)	-	-	X(7)	X(6)	X(2)	X(10)
7	-	-	-	-	-	X(3)	-	-	-	X(7)	X(6)	X(14)

adders, similar to the carry-save-adder approach. The last transfer and sum output line are added by a carry look- ahead adder, the carry being stretched to the left by positioning.

Quartet value	*Signed-digit value*
0000	0
0001	+1
0010	+1
0011	+2
0100	+2
0101	+3
0110	+3
0111	+4
1000	-4
1001	-3
1010	-3
1011	-2
1100	-2
1101	-1
1110	-1
1111	0

Here we have a multiplication multiplier, 3Y, which is not immediately available. To Generate it, we must run the previous addition operation: 2Y + Y = 3Y. But we are designing a multiplier for specific purposes and then the multiplier belongs to a set of previously known numbers stored in a memory chip. We have tried to take advantage of this fact, to relieve the radix-8 bottleneck, that is, 3Y generation. In this

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

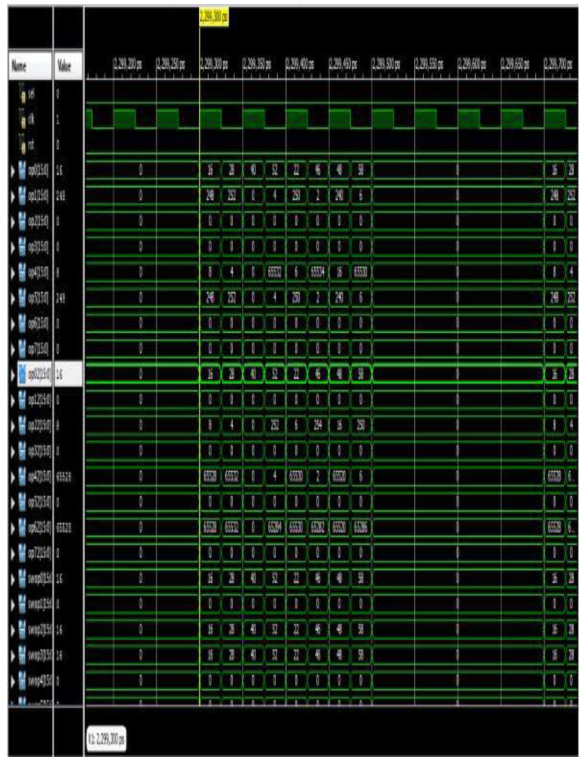
way, we try to obtain a better overall multiplication time or at least comparable to the time, we can obtain using a radix-4 architecture (with the added benefit of using fewer transistors). To generate 3Y with 21-bit words you just have to add 2Y + Y, ie add the number with the same number moved to a left position.A product formed by multiplying it with a multiplier digit when the multiplier has many digits. Partial products are calculated as intermediate steps in the calculation of larger products.The partial product generator is designed to produce the product multiplying by multiplying A by 0, 1, -1, 2, -2, -3, -4, 3, 4. Multiply by zero implies that the product is "0 ". Multiply by" 1 "means that the product remains the same as the multiplier. Multiply by "-1" means that the product is the complementary form of the number of two. Multiplying with "-2" is to move left one as this rest as per table.

SIGN EXTENSION CORRECTOR:

The Sign Extension Corrector is designed to increase the Booth multiplier capacity by multiplying not only the unsigned number but also the signed number. The principle of the sign extension that converts the signed multiplier not signed as follows. When unsign is signalled s_u = 0, it indicates the multiplication of the unsigned number and when s_u = 1, it shows the multiplication of the signed number. When a bit signal is called unsigned bit (s_u), it is indicated whether the multiplication operation is an unsigned number or number.

Sign-unsign	Type of operation
0 1	Unsigned multiplication Signed multiplication

RESULT:



CONCLUSION:

In this article, several FFT architectures are designed and simulated using VERILOG HDL. Power dissipation and fan-out conditions are calculated on different devices using the XILINX 14.5 tool. The proposed processor can simultaneously process two independent data streams and make it suitable for many real-time high-speed applications. The bit inversion circuit present in the previous drawings is eliminated by integrating two FFT processors, and the logs that are present in the architecture are reused for bit

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

reversal. As a result, it avoids the need for additional registers to reduce inverted outputs. Additionally,the proposed architecture offers a higher throughput than previous architectures.

REFERENCES

[1]J. M. Cioffi, The communications Handbook. Boca Raton, FL: CR, 1997.

[2] N. Weste and D. J. Skellern, "VLSI for OFDM," IEEE Commun. Mag., vol. 36, pp. 127–

131, Oct. 1998. [3] C.-H. Chang, C.-L. Wang, and Y.-T. Chang, "Efficient VLSI architectures for fast computation of the discrete Fourier transform and its inverse," IEEE Trans. Signal Process., vol. 48, pp. 3206–3216, Nov. 2000.

[4] S.-F. Hsiao and W.-R. Shiue, "Design of low-cost and high-throughput linear arrays for DFT computations: Algorithms, architectures, and implementations," IEEE Trans. Circuits Syst. II,

Anal. Digit. Signal Process., vol. 47, no. 11, pp. 1188–1203, Nov. 2000.

[5] W.-C. Yeh and C.-W. Jen, "High-speed and low-power split-radix FFT," IEEE Trans. Signal
Process., vol. 51, no. 3, pp. 864–874, Mar. 2003.

[6]S. He and M. Torkelson, "Designing pipeline FFT processor for OFDM (de) modulation," in

Proc. IEEE URSI Int. Symp. Signals, Syst., Electron., 1998, pp. 257–262.

[7] Y. W. Lin, H. Y. Liu, and C. Y. Lee, "A dynamic scaling FFT processor for DVB-T applications," IEEE J. Solid-State Circuits, vol. 39, no. 11, pp. 2005–2013, Nov. 2004.

[8] P. Duhamel and H. Hollmann, "Split radix FFT algorithms," Electron. Lett., vol. 20, pp. 14–16, 1984.

[9]C. Cheng and K. K. Parhi, "High-throughput VLSI architecture for FFT computation," IEEE

Trans. Circuits Syst. II, Exp. Briefs, vol. 54, no. 10, pp. 863–867, Oct. 2007.

[10]Y.-N. Chang, "An efficient VLSI architecture for normal I/O order pipeline FFT design," IEEE

Trans. Circuits Syst. II, Exp. Briefs, vol. 55, no. 12, pp. 1234–1238, Dec. 2008.

[11] M. Ayinala, M. Brown, and K. K. Parhi,

"Pipelined parallel FFT architectures via folding transformation," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 20, no. 6, pp. 1068–1081, Jun. 2012.

[12] M. Garrido, J. Grajal, M. A. Sanchez, and O.

Gustafsson, "Pipelined radix-2k feedforward FFT architectures," IEEE Trans. Very Large Scale Integr.

(VLSI) Syst., vol. 21, no. 1, pp. 23–32, Jan. 2013.

[13] K.-J. Yang, S.-H. Tsai, and G. C. H. Huang, “MDC FFT/IFFT processor with variable length for MIMO-OFDM systems,” IEEE Trans. Very Large

Scale Integr. (VLSI) Syst., vol. 21, no. 4, pp. 720– 731, Apr. 2013. [14] J. Lee, H. Lee, S.-I. Cho, and S.-S. Choi, “A high-speed, low-complexity radix-24 FFT processor for MB-OFDM UWB systems,” in Proc. IEEE Int. Symp. Circuits Syst., May 2006, pp. 210–213.