Design of pipelined parallel FFT architectures Using folding transformation

RAGAVI.V,
M.E (VLSI Design)
Srinivasan Engineering College
ragavimevlsi@gmail.com[1]

K.RENUKA
Assistant Professor (ECE)
Srinivasan Engineering College
renudev20@gmail.com[2]

Abstract - In this paper, a novel approach to develop parallel pipelined architectures for the Fast Fourier transform (FFT) is presented. The folding transformation and register minimization techniques are proposed for designing FFT architectures. Novel parallel-pipelined 128-point radix-2^4 FFT architecture for the computation of complex and real valued fast Fourier transform are derived. For Complex valued Fast Fourier Transform (CFFT), the proposed architecture takes benefit of underutilized hardware in the serial architecture to derive L-parallel architectures not including the increment of hardware complexity by a factor of L. In addition to, the new parallel-pipelined architecture for the computation of Real-valued Fast Fourier Transform (RFFT) is presented. To reduce the hardware complexity, the proposed architecture exploits redundancy in the computation of FFT samples. A comparison is shown between the proposed design and the previous architectures.

Index Terms – Fast Fourier Transform (FFT), folding, radix-2^4, register minimization.

I. INTRODUCTION

DFT is one of the most important tools in the field of digital signal processing. Several Fast Fourier Transform (FFT) algorithms have been developed over the years due to its computational complexity. FFT plays a critical role in modern digital communications such as Digital Video Broadcasting (DVB) and Orthogonal Frequency Division Multiplexing (OFDM) systems. The design of pipelined architectures for computation of FFT of complex valued signals (CFFT) has been carried out. Different algorithms have been developed to reduce the computational complexity, of which Cooley-Tukey radix-2 FFT [1] is very popular.

Algorithms such as radix-4 [2], split-radix [3] and radix-2^2 [4] have been developed based on the basic radix-2 FFT approach. The one of the most classical approaches for pipelined

implementation of radix-2 FFT is Radix-2 multi-path delay commutator (R2MDC) [5]. A standard usage of the storage buffer in R2MDC leads to the Radix-2 Single-path delay feedback (R2SDF) [6] architecture with reduced memory.

The architectures are developed for a specific-point FFT in [7] and [8], whereas hypercube theory is used to derive the architectures in [9].The method of developing these architectures from the algorithms is not well established.

In additional, most of these hardware architectures are not fully utilized and require high hardware complexity. In the period of high speed digital communications, the high throughput and low power designs are essential to meet the speed and power requirements while keeping the hardware overhead to a minimum. In this paper, a new approach to design the architecture from the FFT flow graphs is presented. Folding transformation [10] and register minimization techniques [11], [12] are used to derive several known FFT architectures.

If the input samples are real then the spectrum is symmetric and approximately half of the operations are redundant. The applications such as speech, audio, image, radar, and biomedical signal processing, a specialized hardware implementation is best suitable to meet the real-time constraints. The implantable or portable device saves power by using this type of implementation which is a key limitation. Few pipelined architectures for real valued signals have been proposed [13] based on the Brunn algorithm. However, these are not widely used. Different algorithms such as doubling algorithm, packing algorithm have been proposed for computation of RFFT. These approaches are based on removing the redundancies of the CFFT while the input is real. RFFT is calculated using the CFFT architecture in an efficient manner [14].

In the folding transformation, many butterflies in the same column can be mapped to one butterfly unit. If the FFT size is N, a folding factor of N/2 leads to 2-parallel architecture and in another design, a folding factor of N/4 leads to design 4-parallel architectures in which four samples are processed in the same clock cycle. Various folding sets lead to a family of FFT architectures. Alternatively, known FFT architectures can also be described by the proposed methodology by selecting the appropriate folding set. To reduce latency and the number of storage elements, folding sets are designed. The prior FFT architectures were derived in an informal way, and their derivations were not explained in a systematic way. This is the effort to simplify the design of FFT architectures for arbitrary level of parallelism in an efficient manner by

means of the folding transformation. In this paper, the prior design architectures are explained by constructing the specific folding sets. Then new architecture is derived for radix and levels of parallelism and for either Decimation-In-Time (DIT) or Decimation-In-Frequency (DIF) flow graphs. The new architecture achieves full hardware utilization. It may be noted that all prior parallel FFT architectures did not achieve full hardware utilization. The new real FFT architecture is also presented based on higher radices.

In [15], the parallel-pipelined architectures for the computation of RFFT based on radix-2^2 and radix-2^3 algorithms have been proposed. The real FFT architectures are not fully utilized. This drawback is removed by proposed methodology. The novel parallel-pipelined FFT architectures for the real-valued signals with full hardware utilization based on radix-2^4 algorithm is presented. It combines the advantages of radix-2^n algorithms, which requires fewer complex multipliers when compared to radix-2 algorithm, with the reduction of operations using redundancy.

This paper is organized as follows. The folding transformation and register minimization based FFT architectures design is presented in Section II. The proposed architecture for complex FFT is explained in Section III. The proposed architecture for real FFT is explained in Section IV. In Section V, the proposed architecture is compared with the previous approaches and some conclusions are drawn in Section VI.

II. FFT ARCHITECTURES DESIGN TECHNIQUES

In this section, the folding transformation method and register minimization to derive several known FFT architectures is illustrated in general. The process is described using an 8-point radix-2 DIF FFT as an example. It can be extended to other radices in a similar fashion. Fig. 1 shows the flow graph of a radix-2 8-point DIF FFT. The graph is divided into three stages and each of them consists of a set of butterflies and multipliers. The twiddle factor in between the stages indicates a multiplication by W_N^k, where W_N denotes the N^{th} root of unity, among its exponent evaluated modulo N.

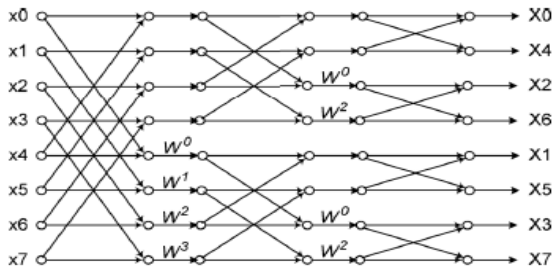

Fig.1 Flow graph of a radix-2 8-point DIF FFT.

This algorithm can be represented as a data flow graph (DFG) as shown in Fig. 2. The nodes in the DFG represent tasks or computations. In this case, all the nodes represent the butterfly computations of the radix-2 FFT algorithm. Assume nodes A and B have the multiplier operation on the bottom edge of the butterfly. The folding transformation is used on the DFG in Fig. 2 to derive a pipelined architecture.

To transform the DFG, a folding set is required which is an ordered set of operations executed by the same functional unit. Each folding set contains K entries some of which may be null operations is called the folding factor, i.e., the number of operations folded into a single functional unit. The operation in the j^{th} position within the folding set (where j goes from 0 to K-1) is executed by the functional unit during the time partition. The term j is the folding order, i.e., the time instance to which the node is scheduled to be executed in hardware.

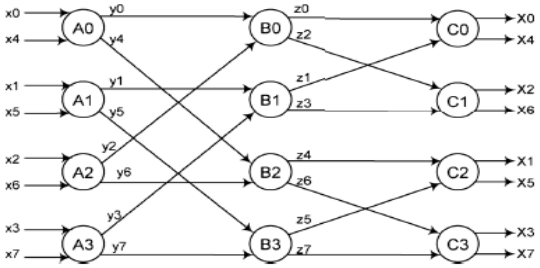

Fig. 2 DFG of a radix-2 8-point DIF FFT.

For example, consider the folding set A = {□, □, □, □, A0, A1, A2, A3} for K=8. The operation $A0$ belongs to the folding set A with the folding order 4. The functional unit executes the operations A0, A1, A2, A3 at the respective time instances and will be idle during the null operations. The systematic folding techniques are used to derive the 8-point FFT architecture. Consider an edge e connecting the nodes U and V with w (e) delays. The folding equation (1) for the edge e is

$$D_F(U \rightarrow V) = K\, w(e) - P_U + v - u \qquad (1)$$

where P_U is the number of pipeline stages in the hardware unit which executes the node U [10]. By using folding sets, folding equations are derived with negative delays (w/o pipeline) and non negative delays (with pipeline or retiming). Consider folding of the DFG in Fig.2 with the folding sets

A = {□, □, □, □, A0, A1, A2, A3}
B = {B2, B3, □, □, □, □, B0, B1}
C = {C1, C2, C3, □, □, □, □, C0}.

Assume that the butterfly operations do not have any pipeline stages, i.e, $P_A=0$, $P_B=0$, $P_C=0$.Retiming and/or pipelining can be used to either satisfy $D_F U \rightarrow V) \geq 0$ or determine that the folding sets are not feasible [10]. The negative delays on some edges can be observed. The equations are

$D_F(A0 \rightarrow B0) = 2$ $\qquad$ $D_F(B0 \rightarrow C0) = 1$

$D_F(A0 \rightarrow B2) = -4$ $D_F(B0 \rightarrow C1) = -6$
$D_F(A1 \rightarrow B1) = 2$ $D_F(B1 \rightarrow C0) = 0$
$D_F(A1 \rightarrow B1) = -4$ $D_F(B1 \rightarrow C1) = -7$
$D_F(A2 \rightarrow B0) = 0$ $D_F(B2 \rightarrow C2) = 1$
$D_F(A2 \rightarrow B2) = -6$ $D_F(B2 \rightarrow C3) = 2$
$D_F(A3 \rightarrow B1) = 0$ $D_F(B3 \rightarrow C2) = 0$
$D_F(A3 \rightarrow B3) = -6$ $D_F(B3 \rightarrow C3) = 1$ (2)

The DFG can be pipelined is shown to ensure that folded hardware has non-negative number of delays. The folded delays for the pipelined DFG are

$D_F(A0 \rightarrow B0) = 2$ $D_F(B0 \rightarrow C0) = 1$
$D_F(A0 \rightarrow B2) = 4$ $D_F(B0 \rightarrow C1) = 2$
$D_F(A1 \rightarrow B1) = 2$ $D_F(B1 \rightarrow C0) = 0$
$D_F(A1 \rightarrow B1) = 4$ $D_F(B1 \rightarrow C1) = 1$
$D_F(A2 \rightarrow B0) = 0$ $D_F(B2 \rightarrow C2) = 1$
$D_F(A2 \rightarrow B2) = 2$ $D_F(B2 \rightarrow C3) = 2$
$D_F(A3 \rightarrow B1) = 0$ $D_F(B3 \rightarrow C2) = 0$
$D_F(A3 \rightarrow B3) = 2$ $D_F(B3 \rightarrow C3) = 1$ (3)

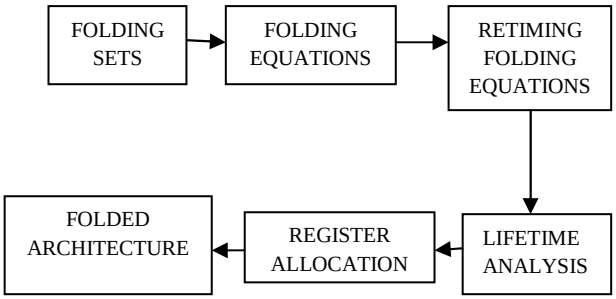

Fig. 3 Block diagram of FFT design techniques

The technique for minimizing register is lifetime analysis [12] which analyzes the time for when a data is produced (T_{input}) and when a data finally is consumed (T_{output}).

$$T_{input} = u + P_U \quad (4)$$
$$T_{output} = u + P_U + \max_v \{D_F(U \rightarrow V)\} \quad (5)$$

where u is the folding order of U and P_U is the number of pipelining stages in the functional unit that executes u. From (3) the 24 registers are required to implement the folded architecture. Lifetime analysis technique is used to design the folded architecture with minimum possible registers. For example, in the current 8-point FFT design, consider the variables y0, y1,. . . y7, i.e., the outputs at the nodes A0,A1,A2,A3 respectively. It takes 16 registers to synthesize these edges in the folded architecture. The linear lifetime table and lifetime chart for these variables is shown in Fig. 4 and Fig. 5. From the lifetime chart, it can be seen that the folded architecture requires 4 registers as opposed to 16 registers in a straightforward implementation. The next step is to perform forward-backward register allocation.

NODE	Tinput→ Toutput
yo	4→6
y1	5→7
y2	-
y3	-
y4	4→8
y5	5→9
y6	6→8
y7	7→9

Fig.4 Linear lifetime table

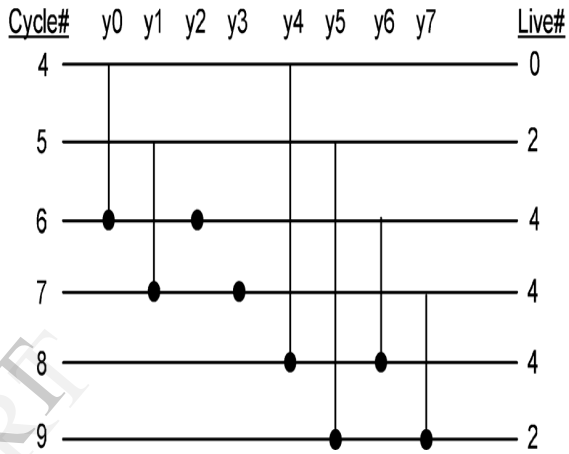

Fig.5 Linear lifetime chart

	I/P	R1	R2	R3	R4
4	y0,y4				
5	y1,y5	y4		y0	
6	(y2),y6	y5	y4	y1	(y0)
7	(y3),y7	y6	y5	y4	(y1)
8		y7	(y6)	y5	(y4)
9			(y7)		(y5)

Fig. 6 Register allocation table.

From the allocation table in Fig.6 and the folding equations, the final architecture in Fig. 7 can be synthesized and can be derived by minimizing the registers on all variables at once. The hardware utilization is only 50% in the derived architecture. This can also be observed from the folding sets where half of the time null operations are being executed, i.e., hardware is idle. The pipelined parallel FFT architectures are presented by using this methodology.

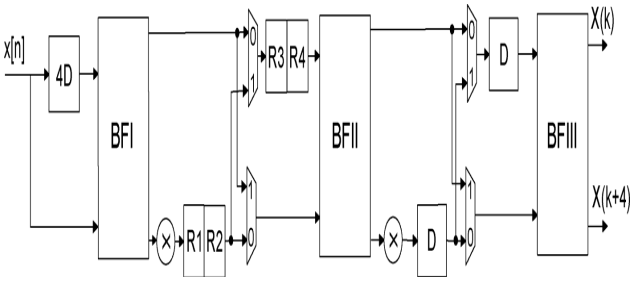

Fig. 7 Folded architecture.

III. PROPOSED ARCHITECTURES WITH COMPLEX INPUTS (CFFT)

The proposed approach can be described using folding methodology [10].The 4-parallel 128-point FFT architecture can be derived using the following folding sets.

A = {A0, A1, A2, A3} A' = {A'0, A'1, A'2, A'3}
B = {B3, B0, B1, B2} B' = {B'3, B'0, B'1, B'2}
C = {C1, C2, C3, C0} C' = {C'1, C'2, C'3, C'0}
D = {D1, D2, D3, D0} D' = {D'1, D'2, D'3, D'1}

The folded architecture can be derived by writing the folding equation [10] for the edges in the flow graph. The register minimization techniques and the forward and backward register allocation scheme [12] are applied to derive the architecture and then the final architecture can be derived. The 128-point FFT flow graph is based on radix-2^4 algorithm which is decimated in time. To achieve the high throughput requirement with low hardware cost, both the proposed pipelining method and radix-2^n algorithms are exploited in this design.

The proposed 4-parallel 128-point FFT architecture is shown in fig.8. It consists of two parallel data paths processing two input samples. Each data path consists of seven butterfly units, four constant and two full complex multipliers, delay elements and multiplexers. The function of delay elements and switches is to store and reorder the input data until the other available data is received for the butterfly operation. The four output data values generated after the first stage are multiplied by constant twiddle factors ($W^1_8 = e^{ij2\pi/8}$, $W^3_8 = e^{ij2\pi 3/8}$). These twiddle factors can be implemented efficiently using Canonic Signed Digit (CSD) approach. The outputs after the third stage are multiplied by the nontrivial twiddle factor.

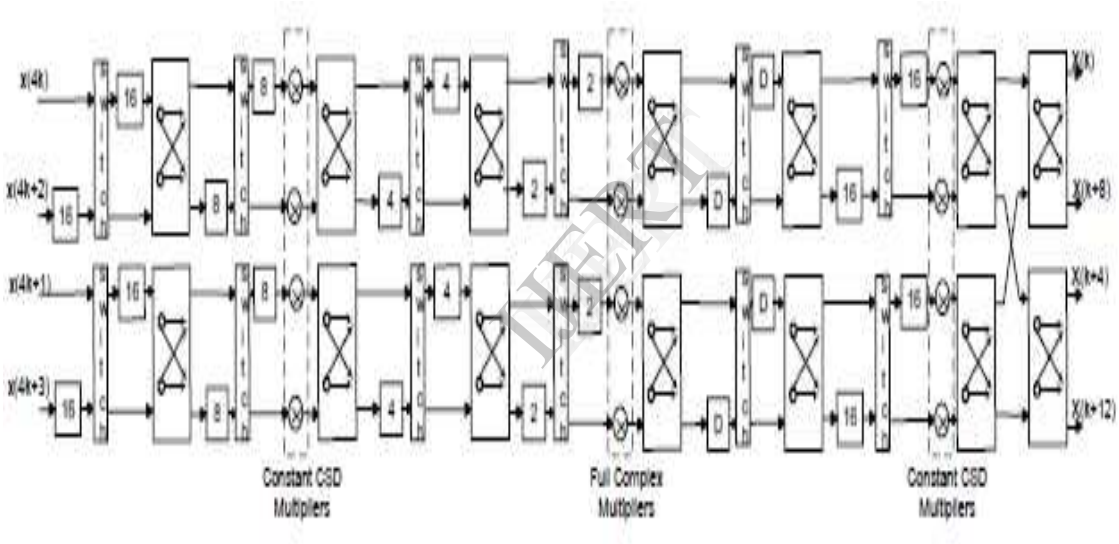

Fig.8 Proposed 128 point CFFT architecture based on radix-2^4 algorithm

Another constant multiplier stage is required before the sixth butterfly stage. The CSD complex constant multiplier processes the multiplication of twiddle factors W^8, W^{16}, W^{24}, W^{48}. These twiddle factors correspond to $cos\ (\pi/8)$, $sin\ (\pi/8)$, and $cos(\pi/4)$.

IV. PROPOSED ARCHITECTURE WITH REAL INPUTS (RFFT)

The proposed radix-2^4 4-parallel architecture is explained using N = 128 point FFT is shown in fig.9. Further, the folding sets can be modified to derive L-parallel architectures of any N-point RFFT.

The radix-2^4 algorithm is described in detail in [8]. We can modify the flow graph similar to the other radices. The advantage of radix-2^4 algorithm is that it needs only one full multiplier every four stages. To derive the 4-parallel architecture divides the nodes into two groups. The nodes in the same group are processed by the same computation unit. Consider the following folding sets.

A = {A0, A2, A4, A6} A' = {A1, A3, A5, A7}
B = {B1, B3, B0, B2} B' = {B5, B7, B4, B6}
C = {C2, C1, C3, C0} C' = {C6, C5, C7, C4}
D = {D3, D0, D2, D1} D' = {D7, D4, D6, D5}

The mapping of nodes to different butterfly structures can be different in the case of 4-parallel architecture. The nodes *{B*4,..., *B*7*}* can be implemented with only a complex multiplier instead of BFIV structure, as these nodes consists of only complex multiplication operation.

Three different butterfly structures are necessary to handle the real and complex data paths. Similar to radix-2^3 architectures, complex multipliers need to operate on samples computed at different time instances.

V. COMPARISON AND ANALYSIS

A.) Complex FFT:

A comparison is made between the previous pipelined architectures and the proposed ones for the case of computing an *N*-point complex FFT in Table 1. The comparison is made in terms of required number of complex multipliers, adders, delay elements, twiddle factors and throughput. The proposed architectures can process 4 samples in parallel, thus achieving a higher performance than previous designs. When compared to some previous architecture, the proposed design doubles the throughput and halves the latency.

B.) Real FFT:

The Table 2 shows the hardware complexity and the throughput of the previous architectures and the proposed ones for computing an *N*-point Real FFT. The hardware complexity of the architectures depends on the required number of multipliers, adders and delay elements. The performance is represented by throughput. The number of multiplier required in the radix-2^4 architecture is less compared to the previous designs. The proposed RFFT architecture leads to low hardware complexity.

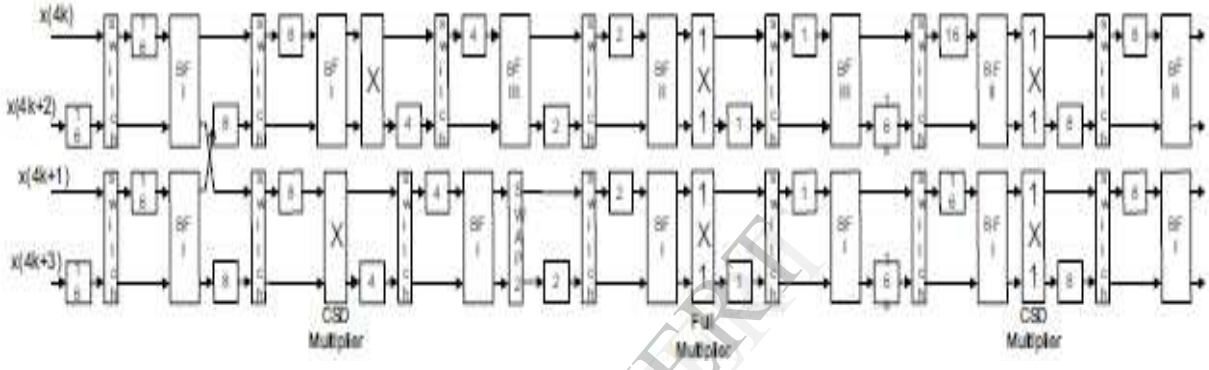

Fig.9 Proposed 128-point RFFT architecture based on radix-2^4 algorithm

Table : 1 Comparison of Pipelined Hardware Architectures for the Computation of N -Point CFFT

ARCHITECTURE	# MULTIPLIERS	# ADDERS	# DELAYS	THROUGHPUT
R2MDC	$2(\log_4 N-1)$	$4\log_4 N$	3N/2-2	1
R2SDF	$2(\log_4 N-1)$	$4\log_4 N$	N-1	1
R2^2SDF	$(\log_4 N-1)$	$4\log_4 N$	N-1	1
R2^3SDF	$(\log_8 N-1)$	$4\log_4 N$	N-1	1
Radix-2 (4-parallel)	$4(\log_4 N-1)$	$8\log_4 N$	2N-4	4
Radix-2^2 (4-parallel)	$3(\log_4 N-1)$	$8\log_4 N$	2N-4	4
Radix-2^3 (4-parallel)	$\log_8 N-1$	$4\log_4 N$	3N/2-2	2
Radix-2^4 (4-parallel)	$2(\log_{16} N-1)$	$4\log_2 N$	N-4	4

ARCHITECTURE	#MULTIPLIERS	#ADDERS	#DELAYS	THROUGHPUT
R2MDC	$2(\log_4 N-1)$	$4\log_2 N$	$2(3N/2-2)$	1
R2SDF	$2(\log_4 N-1)$	$4\log_2 N$	$2(N-1)$	1
$R2^2SDF$	$(\log_4 N-1)$	$4\log_2 N$	$2(N-1)$	1
$R2^3SDF$	$(\log_8 N-1)$	$4\log_2 N$	$2(N-1)$	1
Radix-2^3(4-parallel)	$2(\log_8 N-1)$	$4\log_2 N-2$	$< 2N$	4
Radix-2^4(4-parallel)	$2(\log_{16} N-1)$	$4\log_2 N-2$	$< 2N$	4

Table : 2 Comparison of Pipelined Hardware Architectures for the Computation of N -Point RFFT

VI. CONCLUSION

A novel four parallel 128-point radix-2^4 FFT architecture has been developed using proposed method. The hardware costs of delay elements and complex adders and the number of complex multipliers is reduced using higher radix FFT algorithm by using proposed approach. The throughput can be further increased by adding more pipeline stages which is possible due to the feed-forward nature of the design. The power consumption can also be reduced and leads to low hardware complexity in proposed architectures compared to previous architectures. The simulation can be done by using Modelsim software. A generalized approach to design efficient architectures for the computation of RFFT is also proposed. The approach can be extended to radix-2^5 and higher radix algorithms. Further higher parallel architectures can be developed using the proposed approach.

REFERENCES

[1] J. Cooley and J. Tuckey, "An Algorithm for the Machine Calculation of the Complex fourier series", *Math. Comput*, vol. 19, pp. 297–301, Apr. 1965.

[2] J.A.C. Bingham, "Multicarrier modulation for data transmission: an idea whose time has come," IEEE Communication Magazine, vol. 28, no. 5, pp. 5-14, May 1990.

[3] P. Duhamel, "Implementation of split-radix FFT algorithms for complex, real, and real-symmetric data", *IEEE Trans. Acoust., Speech,Signal Process.*, vol. 34, no. 2, pp. 285–295, Apr. 1986.

[4] S. He and M. Torkelson, "A new approach to pipeline FFT processor", in *Proc. of IPPS*, 1996, pp. 766–770.

[5] L. R. Rabiner and B. Gold, *"Theory and Application of Digital Signal Processing"*, Englewood Cliffs, NJ: Prentice-Hall, 1975.

[6] E. H. Wold and A. M. Despain, "Pipeline and parallel-pipeline FFT processors for VLSI implementation", *IEEE Trans. Comput.*, vol.C-33, no. 5, pp. 414–426, May 1984.

[7] Y. W. Lin, H. Y. Liu, and C. Y. Lee, "A 1-GS/s FFT/IFFT processor for UWB applications", *IEEE J. Solid-State Circuits*, vol. 40, no. 8, pp. 1726–1735, Aug. 2005.

[8] J. Lee, H. Lee, S. I. Cho, S. S. Choi, "A High-Speed two parallel radix-2^4 FFT/IFFT processor for MB-OFDM UWB systems", *IEEE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, pp. 1206-1211, April 2008.

[9] M. Garrido, "Efficient hardware architectures for the computation of the FFT and other related signal processing algorithms in real time", Ph.D. dissertation, Dept. Signal, Syst., Radiocommun., Univ. Politecnica Madrid, Madrid, Spain, 2009.

[10] K. K. Parhi, C. Y. Wang, and A. P. Brown, "Synthesis of control circuits in folded pipelined DSP architectures," *IEEE J. Solid-State Circuits*, vol. 27, no. 1, pp. 29–43, Jan. 1992.

[11] K. K. Parhi, "Systematic synthesis of DSP data format converters using lifetime analysis and forward-backward register allocation," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 39, no. 7, pp. 423–440, Jul. 1992.

[12] K. K. Parhi, "Calculation of minimum number of registers in arbitrary life time chart," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 41, no. 6, pp. 434–436, Jun. 1995.

[13] Y. Wu, "New FFT structures based on the Bruun algorithm," *IEEE Trans. Acoust., Speech Signal Process.*, vol. 38, no. 1, pp. 188–191, Jan. 1990.

[14] W.W. Smith and J. M. Smith, *Handbook of Real-Time Fast Fourier Transforms*. Piscataway, NJ: Wiley-IEEE Press, 1995.

[15] M. Ayinala, M. Brown, K.K. Parhi, "Pipelined parallel FFT architectures via folding transformation", *IEEE Transactions on VLSI Systems*, pp. 1068-1081, vol.20, no. 6, June 2012.