Scientific Journal of Impact Factor (SJIF): 4.72
e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 4, Issue 10, October -2017

Performance Efficient MDC based Pipelined FFT Processor for MIMO-OFDM

P. Pushpalatha[1], K. Ganesh[2]

[1]*Department of ECE, UCEK (A), JNTUK, Kakinada*
[2]*Department of ECE, UCEK (A), JNTUK, Kakinada*

Abstract *—This paper presents a pipelined radix-2 FFT processor, which is efficient in terms of its delay, power and area, to process two independent data streams concurrently. This processor generates outputs in natural order without using bit reversal circuits. The shift registers, which are used to delay the data samples, perform bit reversal operation also. This FFT uses two N/2-point multipath delay commutator FFT architectures to process even and odd samples of two data streams separately. This FFT processor uses Kogge-Stone adders and Dadda multipliers in its butterfly processing units. The Kogge-stone adder is a parallel prefix form of carry look ahead adder and widely used high performance adder in the industries of the present day. Since the adder used generates the carry signal in O(log$_2$n) time, it is widely considered to be the fastest adder design possible. The Dadda multiplier is based on row reduction of partial products by compressing columns using less full adders and half adders compared to the conventional multipliers. Thus, the Dadda multiplier's speed is more and takes less hardware. Therefore, by using the Kogge-Stone adders and Dadda multipliers in butterfly processing units, the proposed FFT processor has less delay, less area and consumes less power compared to the conventional FFT architectures.*

Keywords*-Fast Fourier transform (FFT); ripple carry adder; Kogge-Stone adder; Dadda multiplier; multipath delay commutator (MDC); bit reversal; performance efficiency*

I. INTRODUCTION

The fast Fourier transform (FFT) algorithm has been widely used in many discrete-time signal processing systems. Many advanced communication systems including digital audio broadcasting (DAB), digital video broadcasting (DVB), wireless networks all require a core FFT module to process the orthogonal frequency division multiplexing (OFDM) function. Therefore, designing of an efficient dedicated FFT circuit is a very important issue. Pipeline FFT design is a popular FFT design used in the field of MIMO-OFDM. This approach allocates the individual dedicated data-path for each FFT processing stage to achieve a high level of parallel processing. So it is good for high-throughput real-time FFT applications. As for pipeline schemes, single-path delay feedback (SDF) and multipath delay commutator (MDC) are the two most popular architectures [1].

SDF-based architectures provide memory feedback paths to manage some butterfly outputs during each stage. Additionally, SDF techniques allow the initial FFT output sample to be generated instantly after the final FFT input sample has been processed. MDC-based architectures in comparison replace feedback data paths with feed forward data paths with commutators as switching operations, where each stage forwards its obtained output to the next without any feedback occurring. Sansaloni*et al.*[2]suggested that MDC could save more area than SDF in FFT with multiple streams, and Fu implemented a four-stream MDC FFT/IFFT processor in which the area was 75% that of conventional designs [3].

Many applications such as MIMO-OFDM, image processing, signal processing and so on require multiple FFT operations simultaneously. Most of the FFT architectures produce outputs in bit reversal order which further require bit reversal circuits to generate normal order outputs. In [4]-[6], the architectures process multiple independent data streams using single FFT, so they take more time to generate outputs of all data streams one by one and extra bit reversal circuits are required to generate normal order outputs. Therefore, overall delay is more in these architectures. There exists a combined SDC-SDF FFT which gives normal order output but its throughput is low. In [7], a new FFT architecture is proposed for processing two data streams in normal input-output order. Its speed more compared to previous FFT processors. However, its performance can be improve even more.

The fundamental operations involved in any Digital systems are addition and multiplication. Fast and accurate operation of digital system depends on the performance of adders and multipliers. Improving the speed by reduction in area is the main area of research in VLSI system design. In this paper, the proposed FFT processor usesKogg-Stone (KS) adder and Dadda multipliers in the butterfly processing units. KS adder is fastest adder when compared to other conventional tree adders due to the number of reduced stages. KS adder implementation is the most straightforward, and also it has one of the shortest critical paths of all tree adders. It is the most common architecture for high-performance adders in industry.The Dadda multiplier is based on column compression of partial products using full and half adders. There are three stages in multiplication process. First stage contains partial product tree. That tree is compressed to two rows in second stage. Carry propagation adder (CPA) combines these two rows in the last stage.

II. RADIX-2 MDC FFT ARCHITECTURE TO PROCESS TWO DATA STREAMS

A radix-2 pipelined MDC FFT architecture [7] is designed to process two independent data streams concurrently with its outputs in normal order. To generate normal output order, the processor does not use any additional bit reversal circuits. It uses shift registers in its architecture for bit reversal operation. But the shift registers actual purpose is to delay samples to forward correct samples to butterfly processing elements (BPEs). Thus, the shift registers perform dual role. The delay commutator units disassociates the even and odd samples of two data streams and then they are forwarded to two N/2-point DIF and DIT MDC FFTs by using switches. The outputs of the two N/2-point FFTs are forwarded to two-parallel butterfly units to generate N-point FFT outputs in natural order. This architecture has high throughput and less hardware compared to conventional FFT architectures.

2.1 Operation of the FFT Processor

For the purpose of simplicity, an eight-point FFT architecture is shown in Figure 1. The FFT architecture is divided into eight blocks named as A1, A2, A3, B1, B2, B3, SWT1 and SWT2. The two input data streams are divided into even and odd data streams in A1/B1 blocks which are delay commutator blocks. The even data samples are forwarded to A2 block which is N/2-point DIF MDC FFT. The odd data samples are forwarded to B2 block which is N/2-point DIT MDC FFT. To get the final outputs of N-point FFT operation, the outputs of two N/2-point FFTs are processed in A3/B3 blocks which contain two-parallel butterfly units. Thus the final output comes in natural order. By using switches (SWT1 and SWT2) the samples are forwarded from one block to another block. If SWT1 is in normal mode, the SWT2 is in swap mode and vice versa. They reverse their modes for every N/2 clock cycles. Reordering shift registers (RSRs) [8] in A1 and B1 are used for bit reversing the odd samples and then they are forwarded to N/2 point DIT FFT block. RSRs in A3 and B3 are used for bit reversing the outputs of N/2 point DIF FFT block.

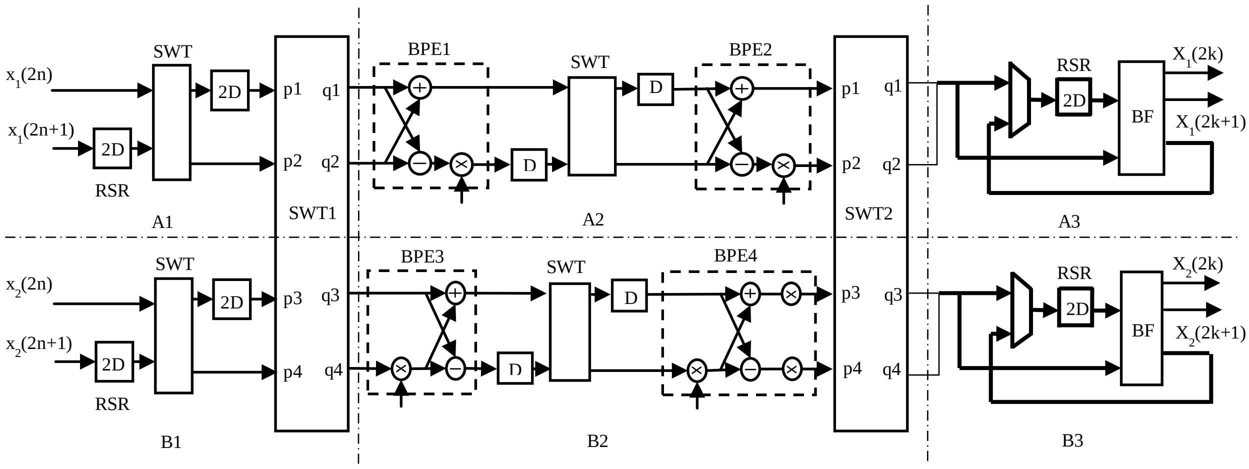

Figure 1. Eight-point radix-2 pipelined FFT architecture with inputs and outputs in normal order

Data propagation of two data streams through different blocks to process eight-point FFT operations is explained below.

1. In the first four clock cycles, the SWT1 and SWT2 are in swap mode and normal mode respectively. The first four samples of x_1 are loaded into the registers of A1 block. In the next four clock cycles, the SWT1 and SWT2 are in normal mode and swap mode respectively. The even samples of x_1 are forwarded from A1 to A2 to perform four-point DIF FFT operation. Meanwhile, the first four samples of x_2 are loaded into the registers of B1 block.
2. After four clock cycles, the SWT1 and SWT2 are in swap mode and normal mode respectively. The outputs of A2 are forwarded to A3 and stored in the shift registers of A3. At the same time, the odd samples of x_1 are bit reversed in A1 and then forwarded to B2 to perform four-point DIT FFT operation.The even samples of x_2 are forwarded from B1 to A2 to perform four-point DIF FFT operation.
3. After four clock cycles, the SWT1 and SWT2 are in normal mode and swap mode respectively.The outputs of B2 are forwarded to A3 and then combined with stored outputs of A2 blockto perform butterfly operations. Thus,eight-point FFT output of $x_1$is generated with natural order. Simultaneously, the outputs of A2 are forwarded to B3 and stored in the shift registers of B3 block.The odd samples of x_2 are bit reversed in B1 and then forwarded to B2 to perform four-point DIT FFT operation.
4. In the next four clock cycles, the SWT1 and SWT2 are in swap mode and normal mode respectively. The outputs of B2 are forwarded to B3 and then combined with stored outputs of A2 block to perform butterfly operations. Thus, eight-point FFT output of $x_2$is generated with natural order.

This architecture requires less registers and its throughput is more compared to single input stream FFT architectures [4]-[6].

III. PERFORMANCE EFFICIENT FFT WITH KS ADDERS AND DADDA MULTIPLIERS

The main aim of this paper is to improve FFT performance by decreasing the delay, reducing the area and power efficiently. The adders and multipliers in the butterfly processing units (BPEs) of the FFT architecture decides the performance of the FFT processor. To achieve high performance, the FFT architecture usesKogge-Stone adders and Dadda multipliers in BPEs. The Kogge-Stone (KS) adder is one of the fundamental tree adders that are widely used in high-performance processors. The Dadda multiplier is a fast tree multiplier which requires less half adders and full adders in its architecture. In this paper, the performance of the proposed FFT architecture, which uses the KS adder and Dadda multipliers, is compared with the performance of FFT, which uses Ripple Carry (RC) adders and Dadda multipliers, and also with the performance of the FFT architecture [7] which uses RC adders and array multipliers. The operations of KS adder and Dadda multiplier are discussed in sub sections.

3.1.Kogge-Stone Adder

The Kogge-Stone adder is a parallel prefix form of carry look-ahead adder. The KS adder concept was first developed by Peter M. Kogge and Harold S. Stone [9]. It is the fastest parallel prefix adder which generates carry signals in O(log$_2$n) time, where n is the size of adder input. It takes more area, but has a lower fan-out at each stage, which increases performance for typical CMOS process nodes. Tree graph of an 8x8 bit KS adder is shown in Figure 2.

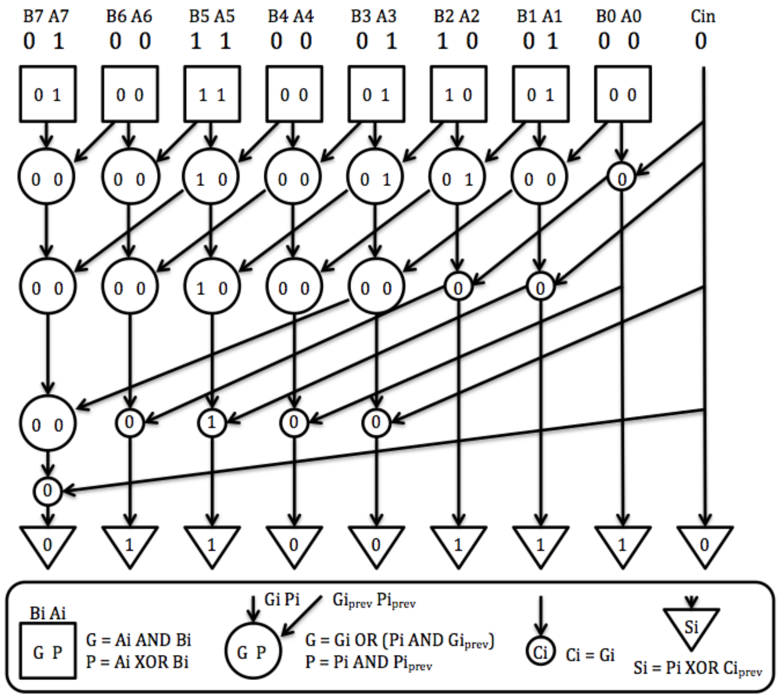

Figure 2. 8x8-bit Kogge-Stone parallel prefix adder tree graph

Algorithm of KS adder contains three computational stages. They are

1. Pre-processing stage
2. Carry generation network stage
3. Post processing stage

3.1.1. Pre-processing Stage: Pre-processing is the first stage where the generate and propagate signals of all the input pairs of signals A and B are generated separately for each bit. The logical equations of the propagate and generate signals are given by the following equations:

$$P_i = A_i \text{ XOR } B_i \quad (1)$$

$$G_i = A_i \text{AND } B_i \quad (2)$$

3.1.2. Carry Generation Network Stage: At this stage the carries of all the bits are generated separately for each bit. They are divided into smaller pieces and this overall process is carried out in parallel for all the bits. Carry generate and carry propagate bits are used as intermediate signals and their logical equations are given as follows:

$$CP_{i:j} = P_{i:k+1} \text{ and } P_{k:j} \quad (3)$$

$$CG_{i:j} = G_{i:k+1} \text{ OR } (P_{i:k+1} \text{ AND} G_{k:j}) \quad (4)$$

3.1.3. Post Processing Stage: This is the last stage of the KS adder which is common for all types of adders, i.e., calculation of summation of the bits, given by the logical equations:

$$C_{i-1} = (P_i \text{ AND } C_{in}) \text{ OR } G_i \qquad (5)$$

$$S_i = P_i \text{ XOR } C_{i-1} \qquad (6)$$

3.2.Dadda Multiplier

Dadda multiplier is an efficient column compression multiplier.This multiplier was invented by computer scientist LuigiDadda [10].The multiplication processoccur in three stages. In the first stage, the partial products matrix is generated by using N^2 AND gates. The second stagecontains a set of row reduction levels where the partial productmatrix is compressed to two rows. The row reduction is processed by placing adders at maximum heights of matrix in optimal manner. This makes the Dadda multiplier less costly inthe reduction stage. The partialproducts which are not covered under adders are forwarded tonext level unchanged. In the last stage, a carry propagatingadder is used to add the two rows so that the final multiplicationresult will come.

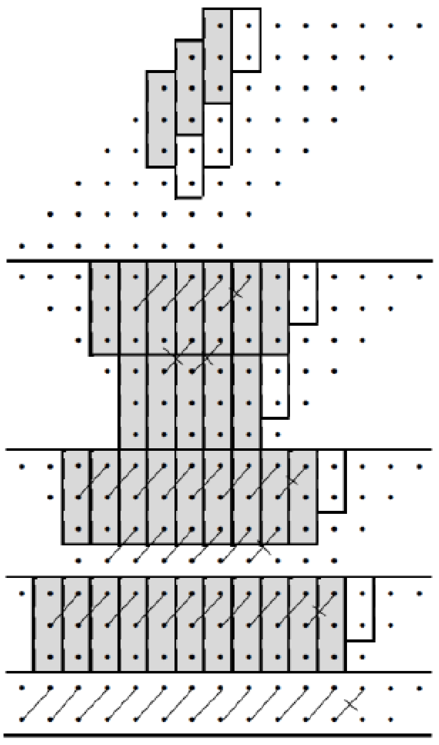

Figure 3. Algorithm for 8x8 bit Dadda multiplier

The reduction process for a Dadda multiplier is developedusing the following recursive algorithm [11].

1. Let $d_1 = 2$ and $d_{j+1} = \lfloor 1.5 * d_j \rfloor$, where j = 1, 2, 3, and so on and $d_j$is the height of the partial product matrix for the level j. Repeat until the largest j^{th} level height does not exceeds the original partial product matrix maximum height.
2. In the j^{th}level from the end, place the full and half adders as needed to achieve a reduced matrix. The columns with more than $d_j$dots and the columns, which receive carries from previous level full and half adders, are compressed.
3. Repeat the second step by letting $j = j - 1$, until a matrix generated with a height of two. This should occur only when j = 1.

Application of this recursive algorithm producesthe dotdiagram for the 8x8-bit Dadda multiplier shown in Figure 3. A Dadda multiplier [11] requires $N^2 - 4N + 3$ number of full adders and $N - 1$ half adders, where N is the size of the operands. The size of final carry propagation adder is $2N-2$. The major advantages oftheDadda multiplier is that it requires less number of adders compared to that of other conventional multipliers. This makes the Dadda multiplier an efficient multiplier architecture for FFT processors.

IV. RESULTS

The proposed FFT processor, which uses KS adders and Dadda multipliers in its BPEs, was implemented in Verilog HDL. The design was synthesized on Xilinx Vertex-5 FPGA with XC5VLX110T device, FF1136 package, -1 speed with a word length of 8 bits.The corresponding simulation results of area, delay and power for different FFTsizes (*N*)are shown in Table 1, Table 2 and Table 3. The simulation results of proposed KD FFT are compared with those of RD FFT and FFT in [7]. Here, KD FFT means FFT with KS adders and Dadda multipliers. RD FFT means FFT with RC adders and Dadda multipliers whereas FFT in [7] uses RC adders and array multipliers.

Table 1. Simulation Results of FPGA for Area

FFT Size (N)	**Architecture**	**Area**		
		Slice Registers	**Slice LUTS**	**Occupied Slices**
8	FFT [7]	64	549	393
	RD FFT	64	534	329
	KD FFT	64	514	278
16	FFT [7]	128	1761	1039
	RD FFT	128	1722	955
	KD FFT	128	1640	880

From the simulation results of area, observe that the proposed KD FFT uses less number of look-up tables (LUTs) and less occupied slices compared to that of the other two FFT architectures. This is due to the fact that the KS adder and Dadda multiplier uses less hardware compared to the other conventional adders and multipliers. Therefore, the proposed FFT processor with KS adders and Dadda multipliers is efficient in terms of area.

Table 2. Simulation Results of FPGA for Delay

Architecture	**Delay (ns)**			
	N=8	**N=16**	**N=32**	**N=64**
FFT [7]	19.662	25.886	31.055	36.903
RD FFT	11.762	14.981	17.496	20.722
KD FFT	9.896	13.339	16.327	19.957

From the simulation results of delay, observe that the proposed KD FFT has less delay for different FFT sizes when compared with that of other two FFT architectures. This is due to fact that KS adder generates carry signal faster than other conventional adders and the Dadda multiplier uses less adders so the time taken to compute multiplication is reduced. Therefore, by using these faster adders and multipliers, the proposed FFT processor is efficient in terms of delay.

Table 3. Simulation Results of FPGA for Power

Architecture	**Power (mW)**	
	N=8	**N=16**
FFT [7]	1051.90	1060.97
RD FFT	1048.93	1054.73
KDFFT	1047.24	1053.52

From the simulation results of power, it is found that the proposed KD FFT consumes less power compared to that ofRD FFT and FFT in [7]. For smaller FFT sizes, there is no much variation in the power consumption. But for higher FFT sizes, the power consumption of proposed FFT will be reduced more to that of existing FFT.

The simulation results of area, delay and power, shows that the proposed FFT's performance improved compared to the existing and conventional FFTs.

V. CONCLUSION

In this paper, a performance efficient FFT processor is implemented which uses Kogge-Stone adders and Dadda multipliers in the butterfly processing elements to improve the performance of the FFT operation. The Kogge-Stone adder is a faster adder as it generates carry signals in less time. Thus, the adder is used in high-performance processors. The Dadda multiplier is faster as it uses less adders compared to other conventional multipliers. So by using these adders and multipliers, the overall FFT architecture uses less adders and also faster. The architecture processes two data streams simultaneously so its speed is more compared to single data stream FFT architectures. Thus, this architecture is useful for high speed real time MIMO applications. Simulation results shows that the proposed KD FFT processor has less area, less delay and consumes less power when compared to that of RD FFT and existing FFT architectures. Therefore, the implemented FFT processor is efficient in terms of its hardware and speed and power consumption. Thus, the FFT processor is useful for the applications which require high performance.

REFERENCES

[1] S. He and M. Torkelson, “A new approach to pipeline FFT processor,” in *Proc. 10th Int. Parallel Process. Symp.*, 1996, pp. 766–770.

[2] T. Sansaloni, A. Perex-Pascual, V. Torres, and J. Valls, “Efficient pipeline FFT processors for WLAN MIMO-OFDM systems,” *Electron. Lett.*, vol. 41, no. 19, pp. 1043–1044, Sep. 2005.

[3] B. Fu and P. Ampadu, “An area efficient FFT/IFFT processor for MIMO-OFDM WLAN 802.11n,” *J. Signal Process. Syst.*, vol. 56, no. 1, pp. 59–68, Jul. 2009.

[4] Y. Chen, Y.-W. Lin, Y.-C. Tsao, and C.-Y. Lee, “A 2.4-Gsample/s DVFS FFT processor for MIMO OFDM communication systems,” *IEEE. Solid-State Circuits*, vol. 43, no. 5, pp. 1260–1273, May 2008.

[5] S.-N. Tang, C.-H. Liao, and T.-Y. Chang, “An area- and energy efficient multimode FFT processor for WPAN/WLAN/WMAN systems,” *IEEE J. Solid-State Circuits*, vol. 47, no. 6, pp. 1419–1435, Jul. 2012.

[6] P. P. Boopal, M. Garrido, and O. Gustafsson, “A reconfigurable FFT architecture for variable-length and multi-streaming OFDM standards,” in *Proc. IEEE ISCAS*, pp. 2066–2070, May 2013.

[7] A. X. Glittas, M. sellathurai, and G. Lakshminarayanan, “A Normal I/O Order Radix-2 FFT Architecture to process Twin Data Streams for MIMO,” in *IEEE Trans.Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 6, pp. 2402–2406, Jun. 2016.

[8] M. Garrido, J. Grajal, and O. Gustafsson, “Optimum circuits for bit reversal,” *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 58, no. 10, pp. 657–661, Oct. 2011.

[9] P. Kogge and H. Stone, “A parallel algorithm for the efficient solution of a general class of recurrence relations,” *IEEE Trans. Computers,* vol. C- 22, no. 8 , pp. 786-793, Aug. 1973.

[10] L. Dadda, “Some Schemes for Parallel Multipliers,” *Alta Frequenza*, vol. 34, pp. 349-356, 1965.

[11] Townsend, Whitney J., Earl E. Swartzlander Jr, and Jacob A. Abraham, "A comparison of Dadda and Wallace multiplier delays,” Optical Science and Technology, SPIE's 48th Annual Meeting on International Society for Optics and Photonics, Dec. 2003.