Volume 5, Issue 12, December 2015 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

High Speed Area Efficient FFT using Modified SQRT CSLA and Compressor

Ambuj Tiwari, Prof. Sonu Lal
IES College of Technology,
Bhopal, India

Abstract*— While designing fast fourier transform (FFT) cores, due to the use of multiplexers, memory, or ROMs, there is a substantial increase in power consumption and area. In order to increase speed and throughput, folding and pipelining methods have been approached by various existing designs. But the prime disadvantage of those architectures is the use of multipliers for twiddle multiplications. This present work has proposed fast fourier transform using compressors based multiplier. Both parallel and pipelining techniques have also been used in the proposed designs. Carry Select adder is known to be the fastest adder among the Conventional adder structures. This work uses an efficient Carry select adder by sharing the binary to excess-1 converter (BEC) term. After a logic simplification, we only need one XOR gate, one AND gate and one inverter gate for carry and summation operation. Through the multiplexer, we can select the correct output according to the logic states of the carry in signal. These all design and experiments were carried out on a Xilinx 14.1i Vertex-7 device family.

***Keywords*— Fast Fourier Transform (FFT), Regular 16-bit SQRT CSLA, Modified SQRT CSLA**

I. INTRODUCTION

In light of the need of fast data transmission, today flexible data exchanges industry goes up against the issue of giving the advancement that have the ability to support a blended pack of organizations running from voice correspondence with a bit rate of a couple kbps to remote media in which bit rate up to 2 Mbps [1]. The FFT is a standout amongst the most regularly utilized computerized signal preparing calculation. As of late, FFT processor has been generally utilized as a part of computerized sign preparing field connected for correspondence frameworks [2]. FFT/IFFT processors are key parts for an Orthogonal Frequency Division Multiplexing (OFDM) based remote broadband correspondence framework; it is a standout amongst the most perplexing and serious calculation module of different remote measures. Since FFT is done in the propelled region, there are a couple of procedures to execute the structure. One of the methodologies to realize the system is using ASIC (Application Specific Integrated Circuit) [3-4]. ASICs are the speediest, most diminutive, and minimum force way to deal with execute FFT into hardware. The essential issue using this strategy is unbending nature of design method included and the more attracted out time to market period for the arranged chip [5].
We have proposed diagram for executing the speedy fourier change (FFT) using adjusted SQRT CSLA and proposed 4:2 and 7:2 compressors operation in hardware keeping the goal of a power successful auxiliary designing. These arrangements are checked using distinctive hardware reproducing gadgets.

II. 4:2 AND 7:2 COMPRESSORS

To add binary numbers with minimal carry propagation we use compressor adder instead of other adder. Compressor is a digital modern circuit which is used for high speed with minimum gates which requires designing technique. This compressor becomes the essential tool for fast multiplication adding technique by keeping an eye on fast processor and lesser area [6].

- **4:2 Compressor**

5:3 compressors are capable of adding 4 bits and one carry, in turn producing a 3 bit output. The 5-3 compressor has 4 inputs A_1, A_2, A_3 and A_4 and 2 outputs Sum and Carry along with a Carry-in (Cin) and a Carry-out (Cout) as shown in Figure 1. The input Cin is the output from the previous lower significant compressor.

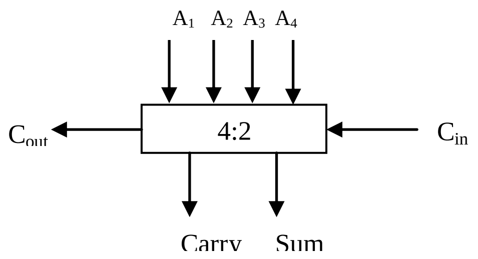

Figure 1: Block Diagram of 4:2 Compressor

The Cout is the output to the compressor in the next significant stage. The critical path is smaller in comparison with an equivalent circuit to add 5 bits using full adders and half adder. The 4-2 compressor is governed by the basic equation

$$A_1 + A_2 + A_3 + A_4 + C_{in} = Sum + 2*(Carry + Cout) \quad (1)$$

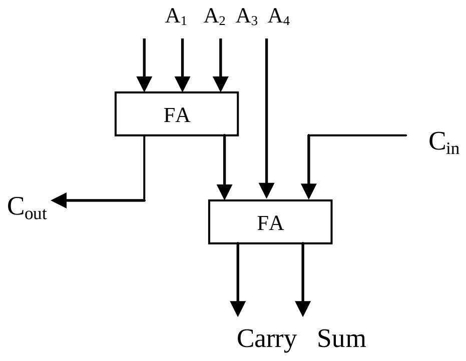

Figure 2(a): Design of 4:2 compressor using Full Adder

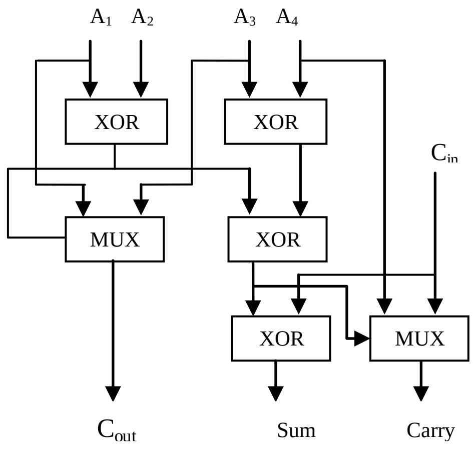

Figure 2(b): 4:2 Compressor using XOR and Multiplexer

The standard implementation of the 4-2 compressor is done using 2 Full Adder cells as shown in Figure 2(a). When the individual full Adders are broken into their constituent XOR blocks, it can be observed that the overall delay is equal to 4*XOR [7-8]. The block diagram in Figure 2(b) shows the existing architecture for the implementation of the 4-2 compressor with a delay of 3*XOR. The equations governing the outputs in the existing architecture are shown below

$$Sum = A_1 \oplus A_2 \oplus A_3 \oplus A_4 \oplus C_{in} \quad (2)$$

$$Cout = (A_1 \oplus A_2).A_3 + \overline{(A_1 \oplus A_2)}.A_1 \quad (3)$$

$$Carry = (A_1 \oplus A_2 \oplus A_3 \oplus A_4).C_{in} + \overline{(A_1 \oplus A_2 \oplus A_3 \oplus A_4)}.A_4 \quad (4)$$

Thus replacing some XOR blocks with multiplexer’s results in a significant improvement in delay. Also the MUX block at the SUM output gets the select bit before the inputs arrive and thus the transistors are already switched by the time they arrive. This minimizes the delay to a considerable extent.

- **7:2 Compressor**

Similar to its 4:2 compressor counterpart, the 7:2 compressors as shown in Figure 3, is capable of adding 7 bits of input and 2 carry’s from the previous stages, at a time [9]. In our implementation, we have designed a novel 7:2 compressor utilizing two 4:2 compressors, two full adder and one half adders. The architecture for the same has been shown in Figure 4.

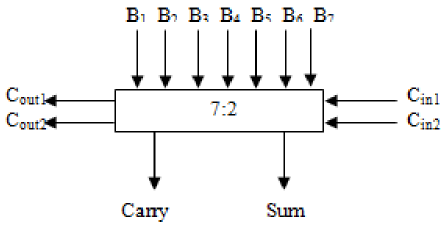

Figure 3: Block Diagram of 7:2 Compressors

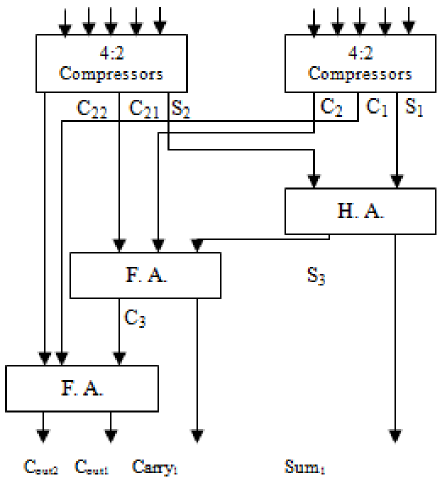

Figure 4: 7:2 Compressor using 4:2 Compressor

$$Sum1 = S_1 \oplus S_2 \qquad (5)$$
$$Carry1 = S_3 \oplus C_1 \oplus C_{21} \qquad (6)$$
$$C_{out1} = C_3 \oplus C_2 \oplus C_{22} \qquad (7)$$
$$C_{out2} = C_3C_2 + C_{22}C_2 + C_3C_{22} \qquad (8)$$

III. REGULAR 16-BIT SQRT CSLA USING RCA

The customary 16-bit SQRT CSLA is a contains two swell pass on adders and a multiplexer. Counting two n-bit numbers with a pass on select snake is done with two adders (along these lines two swell pass on adders) remembering the final objective to perform the calculation twice, one time with the assumption of the pass on being zero and the other tolerating one. After the two results are learned, the right total, and the privilege pass on, is then picked with the multiplexer once the privilege passes on is known [10].

A 16-bit pass on select snake can be made in two one of a kind sizes to be particular uniform piece size and variable square size. Swell pass on adders are the simplest and littler full adders, however their execution is confined by a pass on that must multiply from the base paramount piece to the most-basic piece. The pace of a pass on select snake can be upgraded upto 40% to 90%, by performing the increments in parallel, and reducing the most compelling pass on postponement. Figure 5 exhibits the Regular structure of 16-bit SQRT CSLA. It fuses various swell pass on adders of variable sizes which are isolated into social affairs. Pack 0 contains 2-bit RCA which contains emerge swell pass on snake which incorporates the data bits and the data pass on and results to sum [1:0] and the do. The do of the Group 0 which goes about as the decision information to mux which is in social affair 1, picks the result from the relating RCA (Cin=0) or RCA (Cin=1). Moreover the remaining social events will be picked depending upon the Cout from the past get-togethers.

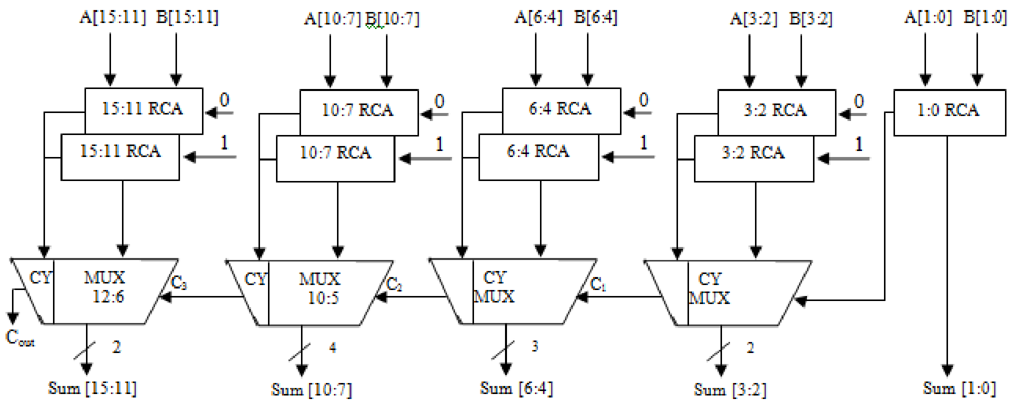

Figure 5: Block Diagram of 16-bit Regular 16-bit SQRT CSLA using RCA

- **Modified SQRT CSLA**

The Binary to excess one Converter (BEC) replaces the ripple carry adder with Cin=1, in order to reduce the area and power consumption of the regular CSLA. The modified16-bit CSLA using BEC is shown in Figure 6. The structure is again divided into five groups with different bit size RCA and BEC. The group 2 of the modified 16-bit CSLA is shown Figure 6

- **Delay and Area Methodology**

The delay and area evaluation methodology considers all gates to be made up of AND, OR, and Inverter (AOI), each having delay equal to 1 unit and area equal to 1 unit. We then add up the number of gates in the longest path of a logic block that contributes to the maximum delay. The area evaluation is done by counting the total number of AOI gates required for each logic block [11-12].

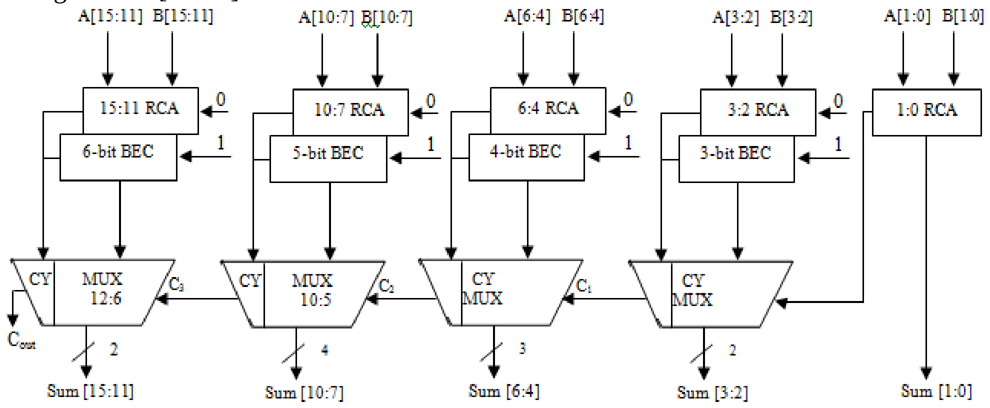

Figure 6: Block Diagram of Regular 16-bit SQRT CSLA using RCA with BEC

Table 2: Delay and Area count of the basic block of CSLA

Adder Block	Delay	Area
XOR	3	5
2:1 MUX	3	4
Half Adder	3	6
Full Adder	6	13

Table 3: Delay and Area Count of Regular SQRT CSLA and Modified SQRT CSLA Groups

Regular SQRT CSLA			Modified SQRT CSLA		
Group	Delay	Area	Group	Delay	Area
Group-1	9	19	Group-1	9	19
Group-2	11	57	Group-2	13	43
Group-3	13	87	Group-3	16	61
Group-4	16	117	Group-4	19	84
Group-5	19	147	Group-5	22	107

IV. FAST FOURIER TRANSFORM

The decimation, however, causes shuffling in data. The entire process involves $v = \log 2N$ stages of decimation. Fast Fourier Transform (FFT) is one of the most efficient ways to implement Discrete Fourier Transform (DFT) due to its reduced usage of arithmetic units. DFT is one of those primary tools that are used for the frequency analysis of discrete time signals and to represent a discrete time sequence in frequency domain using its spectrum samples. The analysis (forward) and synthesis (inverse) equations of an N point FFT are given below.

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn} \quad (8)$$

$$x[n] = \frac{1}{N} \sum_{n=0}^{N-1} X[k] W_N^{-kn} \quad (9)$$

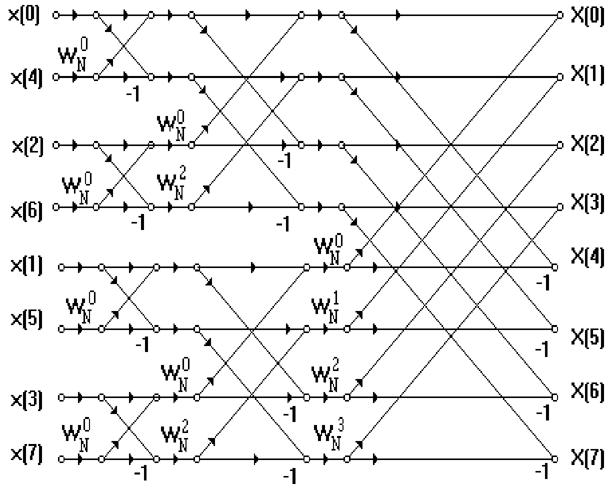

Figure 8: IFFT Signal Flow Graph.

Graphically the operation can be view using FFT flow graph shown in figure 8. From this figure, the FFT computation is accomplished in three stages. The X (0) until X (7) variable is denoted as the input value divided as even and odd terms for FFT computation and X (0) until X (7) is comes in a sequence form as the output. There are two operations to complete the computation in each stage. The upward arrow will execute addition operation while downward arrow will execute subtraction operation. The subtracted value is multiplied with twiddle factor value before being processed into the next stage. This operation is done concurrently and is known as butterfly process. Noted that each of the butterfly process is performed concurrently enable it to execute FFT computation process in a very fast technique.

V. SIMULATION RESULT

All the designing and experiment regarding algorithm that we have mentioned in this paper is being developed on Xilinx 14.1i updated version. Xilinx 14.1i has couple of the striking features such as low memory requirement, fast debugging, and low cost. The latest release of ISE^{TM} (Integrated Software Environment) design tool provides the low memory requirement approximate 27 percentage low. By the aid of that software we debug the program easily. Also included is the newest release of the chip scope Pro Serial IO Tool kit, providing simplified debugging of high-speed serial IO designs for Spratan-3 FPGAs. We functionally verified each unit presented in this paper including modified SQRT CSLA and proposed compressor based multiplier. We have been found from the results shown in Table 2 to Table 4 respectively.

Table 4: Device utilization summary (VERTEX-7) of Fast Fourier Transform (FFT)

Design	No. of slices	No. of 4 input LUTs	MCPD (ns)
FFT using Modified SQRT CSA and Compressor based Multiplier	327	576	22.269
FFT using Modified SQRT CSA and Compressor based Multiplier	171	302	16.369
FFT using Modified SQRT CSA and Proposed Compressor based Multiplier	149	261	13.965

VI. CONCLUSION

While designing various FFT cores, due to the use of multiplexers, memory, or ROMs, there is a substantial increase in power consumption and area. In order to increase speed and throughput, folding and pipelining methods have been approached by various existing designs. But the prime disadvantage of those architectures is the use of multipliers for twiddle multiplications. This present work has proposed FFT using compressors based multiplier. Both parallel and pipelining techniques have also been used in the proposed designs. A compressor adder is a logical circuit which is used to improve the computational speed of the addition of 4 or more bits at a time. The proposed design is hardware efficient as compared to other traditional methods as well as architectures that are built using only compressors.

REFERENCES

[1] Sushma R. Huddar and Sudhir Rao, Kalpana M., "Novel High Speed Vedic Mathematics Multiplier using Compressors", 978-1-4673-5090-7/13/$31.00 ©2013 IEEE.

[2] S. S. Kerur, Prakash Narchi, Jayashree C N, Harish M Kittur and Girish V A, "Implementation of Vedic multiplier for Digital Signal Processing", International Conference on VLSI, Communication & Instrumentation (ICVCI) 2011, Proceedings published by International Joural of Computer Applications® (IJCA), pp.1-6.

[3] Himanshu Thapaliyal and M.B Srinivas, "VLSI Implementation of RSA Encryption System Using Ancient Indian Vedic Mathematics", Center for VLSI and Embedded System Technologies, International Institute of Information Technology Hyderabad, India.

[4] Jagadguru Swami Sri Bharati Krishna Tirthaji Maharaja, "Vedic Mathematics: Sixteen simple Mathematical Formulae from the Veda", Delhi(2011).

[5] Sumit Vaidya and Depak Dandekar. "Delay-power perfor-mance comparison of multipliers in VLSI circuit design". International Journal of Computer Networks & Communications (IJCNC), Vol.2, No.4, July 2010.

[6] P. D. Chidgupkar and M. T. Karad, "The Implementation of Vedic Algorithms in Digital Signal Procesing", Global J. of Eng. Edu, Vol.8, No.2, 204, UICEE Published in Australia.

[7] S. Correa, L. C. Freitas, A. Klautau and J. C. W. A. Costa, "VHDL Implementation of a Flexible and Synthesizable FFT Processor", IEEE LATIN AMERICA TRANSACTIONS, VOL. 10, NO. 1, JAN. 2012.

[8] Y. Kim and L. -S. Kim, "16-bit carry-select adder with reduced area,"Electron. Lett. vol. 37, no. 10, pp. 614-615, May 2001.

[9] B. Ramkumar, H. M. Kittur, and P. M. Kannan, "ASIC implementation of modified faster carry save adder," *Eur. J. Sci. Res.*, vol. 42, no. 1, pp. 53–58, 2010.

[10] T. Y. Ceiang and M. J. Hsiao, "Carry-select adder using single ripple carry adder," *Electron. Lett.*, vol. 34, no. 22, pp. 2101–2103, Oct. 1998.

[11] Y. Kim and L.-S. Kim, "64-bit carry-select adder with reduced area," *Electron. Lett.*, vol. 37, no. 10, pp. 614–615, May 2001.

[12] J. M. Rabaey, *Digtal Integrated Circuits—A Design Perspective*. Upper Saddle River, NJ: Prentice-Hall, 2001.

[13] Y. He, C. H. Chang, and J. Gu, "An area efficient 64-bit square root carry-select adderfor low power applications,"in *Proc.IEEE Int. Symp. Circuits Syst.*, 2005, vol. 4, pp. 4082–4085.