DOI: 10.48047/IJIEMR/V11/I01/18

Title: **USING ADDITION SCHEMES TO IMPROVE THE PERFORMANCE OF OPTIMIZED RADIX-2 AND RADIX-4 FFT BUTTERFLIES**

volume 11, Issue 01, Pages: 113-119

Paper Authors **Mr. D. Nethaji, Mr.A Bala Raju**

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

USING ADDITION SCHEMES TO IMPROVE THE PERFORMANCE OF OPTIMIZED RADIX-2 AND RADIX-4 FFT BUTTERFLIES

Mr. D. Nethaji, DE&CE Branch,Dept of Electronics and Communication Engineering,Mahatma Gandhi Institute of Technology, Telangana, India,

Mr.A Bala Raju, Assistant Professor, Dept of Electronics and Communication Engineering,Mahatma Gandhi Institute of Technology, Telangana, India.

ABSTRACT:

In FFT computation, the butterflies play a central role, since they allow the calculation of complex terms. Therefore, the optimization of the butterfly can contribute for the power reduction in FFT architectures. In this paper we exploit different addition schemes in order to improve the efficiency of 16 bitwidth radix-2 and radix-4 FFT butterflies. Combinations of simultaneous addition of three and seven operands are inserted in the structures of the butterflies in order to produce powerefficient structures. The used additions schemes include Carry Save Adder (CSA), and adder compressors. The radix-2 and radix-4 butterflies were implemented in hardware description language and synthesized to 45nm Nangate Open Cell Library using Cadence RTL Compiler. The main results show that both radix-2 and radix-4 butterflies, with CSA, are more efficient when compared with the same structures with other adder circuits.

Keywords- FFT; Radix-2 butterfly; Radix-4 butterfly; Adders compressors; Carry Save Adders

1. INTRODUCTION

Fast Fourier Transform (FFT) is the largely implementation of the Discrete Fourier Transform (DFT), because this algorithm needs less computation due its recursive operator named butterfly [1].

A radix-2 butterfly, with Decimation in Time (DIT), based on three multipliers, was presented in [2], whose architecture uses only 3-2 adder compressor [3] in its internal structure. In this work we show that the use of Carry Save adder - CSA [4] is more efficient for this butterfly. In [5] the structures of the radix-2 of [2] were optimized with the reduction of two adders/subtractors. In [5], the radix-4 butterfly was optimized using part of the optimized radix-2 butterfly.

In this work we are able to identify what is the best addition scheme for both radix-2 and radix-4 butterflies. While in the radix-2, 3-2 adder compressors or CSA can be used, the radix-4 butterfly enables the use of 7-2 adder compressors or CSA tree schemes. The 7-2 adder compressor was implemented with the combination of different structures based on CSA, 3-2, 4-2, and 5-2 adder compressors.

To implement N-bit, CSA, 3-2, and 7-2 adder compressors, it is necessary a recombination line of partial Carry and Sum terms. To make the recombination of these terms, is used a cascade of half and full-adders circuits, in a Ripple Carry (RCA) form. In this work, efficient adders schemes such as Carry Look Ahead (CLA) [6], Kogge-Stone [7] adders, and efficient 3-2 adder compressor [3] are exploited inside the recombination line of addition. The best of the adders, in terms of delay and power dissipation, is used inside the addition recombination line of CSA and adder compressors.

The radix-2 and radix-4 butterflies, with different adder schemes, were implemented in VHDL, with 16-bit width, and synthesized to 45nm Nangate Open Cell Library [8], using Cadence synthesis tool. Area, delay, and power values for the dedicated butterflies are presented. The main results show that both butterflies with CSA are the most power efficient structures. The main contribution of this work is the exploration of different addition schemes in the optimized radix-2 and radix4 butterflies previously proposed in literature in order to provide power efficiency improvement. The rest of the paper is organized as follow.

2. LITERATURE REVIEW

Fast Fourier Transform (FFT) is the largely implementation of the Discrete Fourier Transform (DFT), because this algorithm needs less computation due its recursive operator named butterfly. This operator performs the calculation of complex terms, which involves multiplication of input data by appropriate coefficients [1]. In [2], a radix-2 butterfly structure, with Decimation in Time (DIT), based on four multipliers, and three other structures based on three multipliers, for the calculation of the real and imaginary parts are presented. A new structure, also based on three multipliers, was presented in [3]. Among the structures of [2], and [3], the original structure (named structure A), based on 4 multipliers, presented the best power consumption result.

However, one of the three multiplierbased structures (named structure B) proved to be potentially power consumption reduction, mainly when two levels of pipeline was used, because it enabled the reduction of one multiplier, and because the large reduction of the critical path. In this work, we optimize the radix-2 butterfly structure B from [2] by changing the positions of the operands and the twiddle factors inside the architecture, what enables the reduction of two adder/subtractor circuits. According to [4], radix-4 algorithms have a computational advantage over radix-2 algorithms because one radix-4 butterfly does the work of four radix-2 butterflies, and the radix-4 butterfly requires only three complex multipliers compared to four complex multipliers of four radix-2 butterflies. In this work, the focus is to reduce the number of real multipliers in the radix-4 DIT butterfly, whose original structure is composed of 12 real multipliers to compute one complex multiplication. As will be presented later, when using part of the proposed optimized radix-2 butterfly, the radix-4 butterfly has reduced three real multipliers inside its structure.

Most of the low power implementations of FFT from the literature try to optimize the entire architecture by using techniques such as pipelining, reusing the butterflies in sequential and semi-parallel architectures, or even reordering the twiddle factors such as in [9]-[11]. However, neither mentioned work proposed to reduce the number of multipliers in the butterfly operator from the FFT architecture. The reduction of the number of multipliers in the radix-2 butterfly was presented in [12], where only two multipliers are used in the butterfly. However, the butterfly was operated at two clock cycles, i.e., the multiplier was reused for the butterfly calculation. In [5] the radix-2 and radix-4 butterflies operate at one clock cycle, but they do not exploit different addition scheme

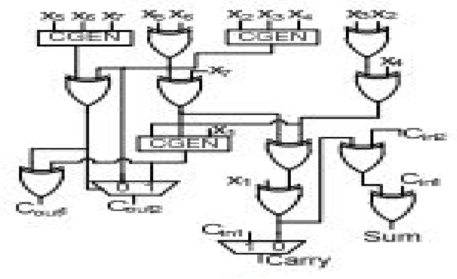
International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

in order to improve the power efficiency of the butterflies, as in this work.

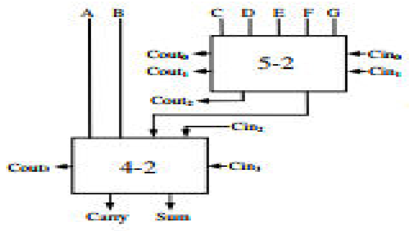
3. RADIX-2 AND RADIX-4 DIT BUTTERFLIES OVERVIEW

The FFT X(k) of a signal x(n) can be computed using (1), where WN is named twiddle factor, i is the imaginary component and N is the number of points of the FFT. The FFT has a hierarchical computation and the butterfly plays a central role in this computation [1]. In this work, radix-2 and radix-4 butterflies are exploited.

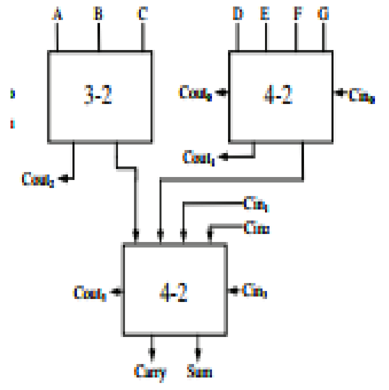
$$X(k) = \sum_{n=0}^{N-1} x(n).W_N^{k.n} \qquad W_N = e^{-j2\pi/N} \qquad 0 \le k \le N-1 \;\&\; 0 \le n \le N-1 \qquad (1)$$



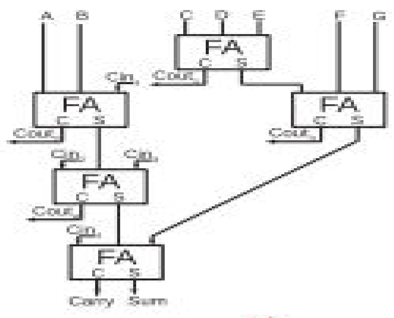
(a)



(b)



(c)



(d)

Fig. 1. Structure of 7-2 adder compressor (a) based on (b) based on 5-2 and 4-2 adder compressors, (c) based on 3-2 and 4-2 adder compressors and (d) Structure of CSA overturned-stairs tree for seven operands .

4. ADDITION SCHEMES FOR THREE AND SEVEN OPERANDS

The used additions schemes consider more than two operands additions at a time. As in the radix-2 and radix-4 butterflies three and seven operands additions are allowed, respectively, thus different addition schemes are exploited in this work.

A.Addition Schemes for Three Operands

For the addition of three operands, CSA and 3-2 adder compressor are used. The basic idea of the CSA is that three numbers can be reduced to two, in a 3-2 compressor, by doing the addition while keeping the carries and the sum separate, as shown in Fig. 1(a) [3]. The CSA is very fast because it simply outputs the carry bits instead of propagating them to the left.

The 3-2 compressor uses one multiplexer (MUX) to generate the Carry output and two Exclusive-OR (EXOR) gates to generate the Sum term, as presented in Fig. 1(b) [3]. The final sum result is given according to S = Sum + 2Carry . The critical path of the 3-2 adder compressor is given by the two EXOR gates of the Sum term calculation [3].

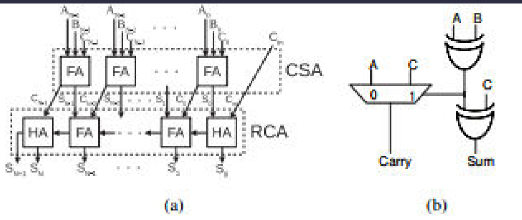


Fig. 2. Internal structure of: (a) N-bit CSA (b) 3-2 adder compressor .

B.Addition Schemes for Seven Operands

For the addition of seven operands, we have exploited different schemes, such as 7-2 adder compressors, and a tree of addition based on CSA scheme. The 7-2 adder compressor proposed in [14] is presented in Fig. 2(a). In this structure, the critical path is given by 6 EXOR gates. All the inputs and output Sum have the same weight. On the other hand, while the output Carry and Cout1 are weighted one bit order higher, the output Cout2 has two bit order higher than the inputs. Besides the EXOR gates and MUX, it also uses a carry generator module (CGEN). A scheme of CSA overturned-stairs tree for seven operands [15] is also used in this work, as presented in Fig. 1(d). This structure presents a mix of the Wallace tree and the balanced tree, what can provide a good relationship between power and delay values.

C.Addition Schemes for the Recombination Line of CSA and Adder Compressors

To implement an N-bit CSA, or an N-bit adder compressor, we have to use a recombination of partial Carry and Sum terms. This recombination is made from a cascade of half and full-adders circuits in a RCA form, as marked in the dotted lines of the example in Fig. 3, that shows an 8-bit addition using 4-2 adder compressor. The line of recombination is a bottleneck in the compressor performance, since the partial carries produced by the actual block are propagated to the next blocks.

For the RCA recombination line, four structures were used: 1- RCA composed of Half Adder (HA) and Full Adder (FA), as in Fig. 3, 2- RCA composed of HA and 3-2 adder compressor, 3- RCA based on Carry Look Ahead adder, and finally 4- RCA based on Kogge-Stone adder.

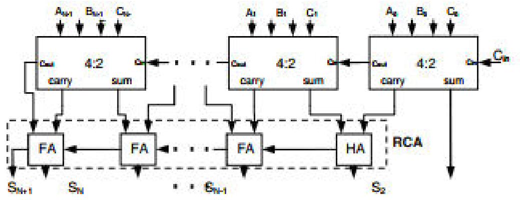


Fig. 3. Example of 8-bit addition using 4-2 adder compressors

The best results for the addition of three operands, in terms of number of cells and cell area, is the structure named 3-2/RCA_3-2, and the best results for power is the structure named CSA_FA/RCA_3-2. In fact, although the CSA structure presents more area, and consequently more leakage power, the aspect of presenting less dynamic power contributes for the slightly reduction in total power in this structure. From now, we will be using in the radix-2 butterfly, the CSA_FA/RCA_3-2 instance.

Table III shows the synthesis results of the 16-bit width addition of seven operands, where five structures were used with 7-2 adder compressor: 1- with the structure of [14], and shown in Fig. 4(a) (named 7-2_[14]); 2- with one 5-2 and one 4-2 adder compressor (named 7-2_5-2,4-2); 3- with two 4-2 and one 3-2 adder compressor (named 7-2_4-2,3-2); 4- with CSA using 3-2 adder compressors (named 7-2_CSA_3-2), and finally 5- with CSA using FA (named 7-2_CSA_FA). For all structures, the delay was set to 1100ps.

Table III shows that the best results, for all parameters, is the structure named 7-

2_CSA_FA. This occurs because the good relationship between speed performance and power dissipation presented by the CSA overturned-stairs tree. From now, we will use in the radix-4 butterfly, the addition of seven operands with CSA using FA, and the recombination line with HA and 3-2 adder compressors.

Instance	Number of Cells	Cells Area (μm^2)	Leakage Power (μW)	Dynamic Power (μW)	Total Power (μW)
7-2_5-2,4-2	774	804	10,82	292,29	303,11
7-2_4-2,3-2	741	744	9,52	256,32	265,84
7-2_CSA_3-2	713	730	8,96	237,63	246,59
7-2_[14]	668	713	8,36	218,38	226,75
7-2_CSA_FA	621	663	7,08	202,03	209,12

Table I. Results for the 16-bit width addition of seven operands with different adders structures

5. RESULTS

Optimized Radix 4:

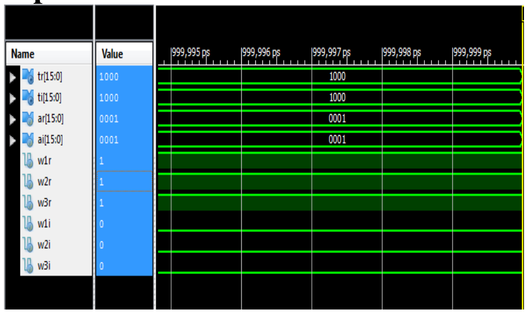


Fig 4: Radix2 using 3*2 Compressor

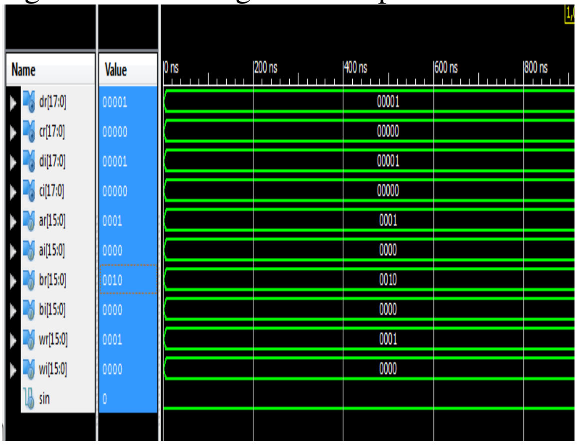


Fig 5:Optimized radix 2

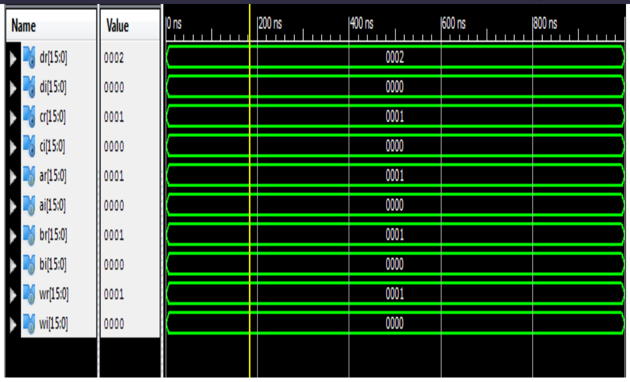
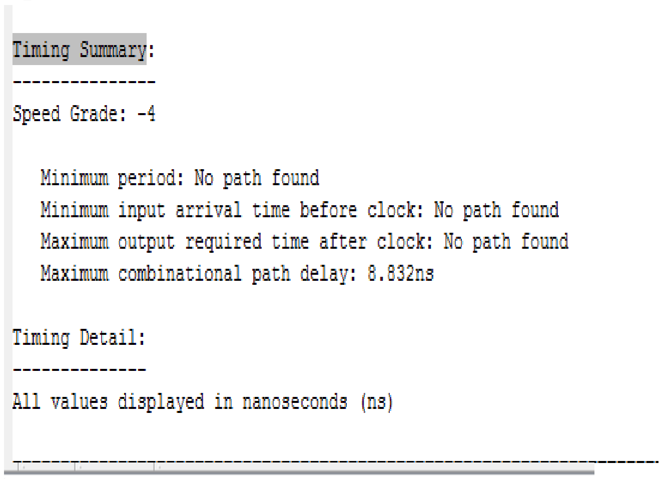


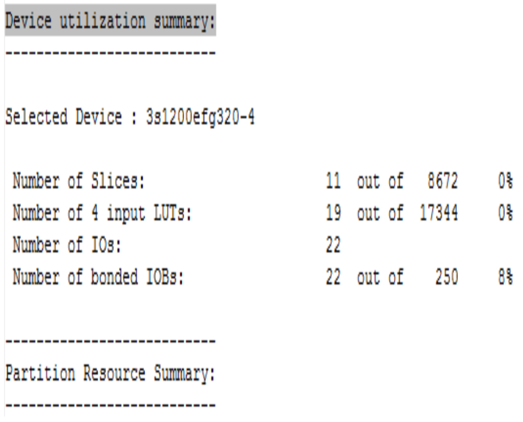
Fig 6:Synthesis reports
Optimized radix 4



```
Timing Summary:
---------------
Speed Grade: -4

   Minimum period: No path found
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: No path found
   Maximum combinational path delay: 8.832ns

Timing Detail:
--------------
All values displayed in nanoseconds (ns)
```



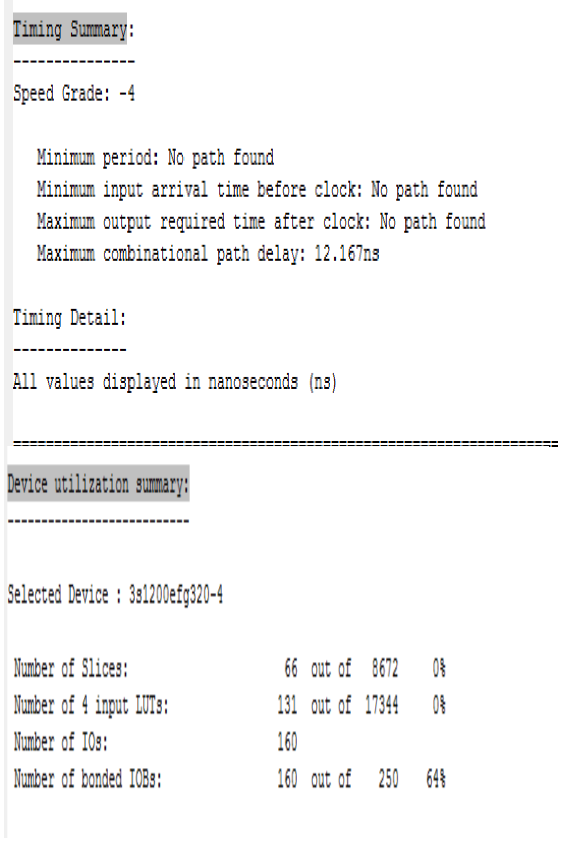
```
Device utilization summary:
---------------------------

Selected Device : 3s1200efg320-4

 Number of Slices:                      11  out of   8672     0%
 Number of 4 input LUTs:                19  out of  17344     0%
 Number of IOs:                         22
 Number of bonded IOBs:                 22  out of    250     8%

---------------------------
Partition Resource Summary:
---------------------------
```

Optimized radix 2:



```
Timing Summary:
---------------
Speed Grade: -4

   Minimum period: No path found
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: No path found
   Maximum combinational path delay: 12.167ns

Timing Detail:
--------------
All values displayed in nanoseconds (ns)

=========================================================================
Device utilization summary:
---------------------------

Selected Device : 3s1200efg320-4

 Number of Slices:                      66  out of   8672     0%
 Number of 4 input LUTs:               131  out of  17344     0%
 Number of IOs:                        160
 Number of bonded IOBs:                160  out of    250    64%
```

6.CONCLUSION

This paper presented different addition schemes for radix-2 and radix-4 butterflies. The main results showed that the best adder scheme for the recombination line of CSA and adder compressors is that uses internally HA and 3-2 adder compressor. This efficient recombination line was used in the CSA and adder compressors for three and seven operands additions. As could be presented, the radix-2 and radix-4 butterflies from the literature were improved by using a CSA with FA, and one line of pipeline. These results prove the efficiency of the CSA for both butterflies.

7. REFERENCES

[1] J. Cooley and J. Tukey, "An algorithm for the machine calculation of the complex fourier series", Math. Computation, vol.19, pp.297-301, 1965.

[2] M. Fonseca, E. Costa, and J. Martins, "Design of Power Efficient Butterflies from Radix-2 DIT FFT Using Adder Compressors with a New XOR Gate Topology," Analog Integrated Circuits and Signal Processing, vol. 73, pp. 945-954, 2012.

[3] S. Veeramachaneni, et. al., "Novel Architectures for High-Speed and Low-Power 3-2, 4-2 and 5-2 Compressors," In VLSI Design, 2007. Held jointly with 6th International Conference on Embedded Systems, 20th International Conference on, 2007, pp. 324–329.

[4] T. Kim, W. Jao, and S. Tjiang, "Arithmetic Optimization using CarrySave-Adders, " In 35th DAC, pp. 433–438, 1998.

[5] R. Neuenfeld, M. Fonseca, E. Costa, "Design of Optimized Radix-2 and Radix-4 Butterflies from FFT with Decimation in Time, "Proceedings of 7th IEEE Latin American Symposium on Circuits & Systems (LASCAS), pp. 1-4, 2016.

[6] B. Parhami, Computer Aritmethic - Algorithms and Hardware Desings: Oxford University Press. 2000.

[7] P. M. Kogge, H. S. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence equations," IEEE, 1973.

[8] NanGate 45 nm Open Cell Library, www.nangate.com/?page_id=22.

[9] P. Sophy, P. Srinivasan, J. Raja, and M. Avinash, "Analysis and design of low power radix-4 FFT processor using pipelined architecture," In IEEE International Conference on Computing and Communications Technologies (ICCCT), 2015, pp. 227-232.

[10] A. Luz, E. Costa, and S. Ghissoni, " Reducing the Hamming distance of encoded FFT twiddle factors using improved heuristic algorithms, " In IEEE Fourth Latin

American Symposium on Circuits and Systems (LASCAS), 2013, pp. 1-4.

[11] E. Costa, J. Monteiro, and S. Bampi, “Gray Encoded Arithmetic Operators Applied to FFT and FIR Dedicated Datapaths,” In 12th International Conference on Very Large Scale Integration (VLSI-SoC), 2003, pp. 307-312.

[12] T. Takala and K. Punkka, “Butterfly Unit Supporting Radix-4 and Radix-2 FFT,” Proceedings of the 2005 International TICSP Workshop on Spectral Methods and Multirate Signal Processing, SMMSP 2005, vol. 30, pp. 47-54, 2005.

[13] T. Kim, W. Jao and S Tjiang, “Arithmetic Optimization using CarrySave adders, ” In 35th DAC Conference, pp. 433-438,1998.

[14] Rouholamini, M. Kavehie, O. Mirbaha, A., Jasbi, S. and Navi, K, “A New Design for 7:2 Compressors, ” IEEE/ACS Intl. Conf. on Computer Systems and Applications, 2007.

[15] Earle, J. G. "Latched carry-save adder." IBM Technical Disclosure Bulletin 7.10 (1965): 909-910.