Reducing Area and Power in FFT Architectures Using Twiddle Factor Decomposition Approach

Sidinei Ghissoni, Eduardo Costa, Ricardo Reis
{sghissoni, reis}@inf.ufrgs.br, ecosta@ucpel.tche.br
UFRGS, UCPEL, UNIPAMPA

Abstract

This paper presents efficient method for reducing area and power for multiplication part of radix-2 FFT (Fast Fourier Transform) architecture based on the decomposition of its twiddle factors. The proposed approach consists on the decomposition of the original real and imaginary coefficients into other less complex ones so that the architecture can be implemented with less area, what leads to the reduction of its power consumption. The method is based on the use of Constant Matrix Multiplication (CMM) and gate level approaches. A control unit is responsible for selecting the correct constant to be used after the decomposition. The proposed architectures were synthesized using SYNOPSYS Design Compiler for the UMC130nm technology. The results show that reductions up to 24% in area and 15% in power could be achieved when compared with the solutions of state of the art.

1. Introduction

Fast Fourier Transform is the largely used implementation of the Discrete Fourier Transform (DFT) used in Digital Signal Processing (DSP) applications, such as audio and video process, wireless communication, and it is also found in modules of WLAN chips. The various existing FFT algorithms use the partitioning of the set of input samples into sequences of half of the length of the original sequence. This is done recursively until there are only sequences of length 2, where the input samples cannot be more partitioned. This algorithm is named radix-2 FFT [1]. The partitioning of the input sequence into more than two subsequences leads to higher radices of the FFT algorithm, what leads to the increase of arithmetic operations.

The mainly complexity of designing FFT is to compute the calculations of complex terms that involves the multiplication of input data by appropriate coefficients named twiddle factors [2]. As the number of used coefficients increases as higher the number of points in the FFT, we propose in this work the reduction of the complexity of the butterfly implementation by decomposing the original coefficients into less complex ones that can be easily implemented by using only shifts, and sharing the partial coefficients as much as possible.

The decomposed coefficients are shared by using Constant Matrix Multiplication (CMM) approach [3] in order to allow further implementation of an efficient FFT. In this method each operation of each set of constant can be implemented using addition/subtraction and shift operations rather than using a general multiplier [4]. Moreover, the controlling of multiplexers, that enable the reuse of the decomposed constants by adder/subtractors, as much as possible, enables the reduction of complexity of the butterfly. As the use of CMM generates a tree of adder/subtraction and shift operations, we were able to adjust our approach in order to optimize the circuits by considering the use of more efficient Carry Save Adders - CSA at gate-level metrics, where half adders (Has) and full adders (FAs) for addition and subtraction operations under unsigned and signed input models can be used. By properly exploiting the use of CSA, the hardware implementation can be significantly optimized, because the opportunity for sharing the common sub-expression is largely increased.

This paper begins with the concepts in section 2, as well as the use of a gate level metric in CSA. In Section 3, the proposed methodology is introduced. In Section 4 some experimental results are presented, and finally, in Section 5 presents some conclusions of this work and ideas for future work.

2. The Fast Fourier Transform

The FFT has a hierarchical computation and the butterfly plays a central role in this computation [4]. For the radix-2 FFT algorithm with decimation in time, the butterfly allows the calculation of complex terms according to the Fig. 1. Where the A and B are the sequence even and odd of the points of a signal *x(n)* and *W* is named twiddle factor.

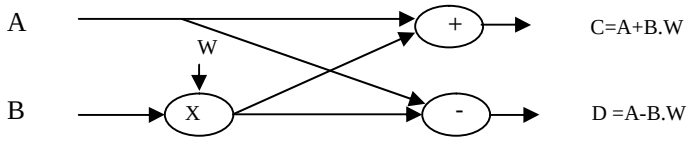

Figure 1. Radix-2 DIT butterfly structure.

According to the structure shown in Fig. 1, the butterfly is composed by addition, subtraction and multiplication, where these operations involve complex numbers. In this work, we implement the butterfly structure using the decomposition of the real and imaginary coefficients.

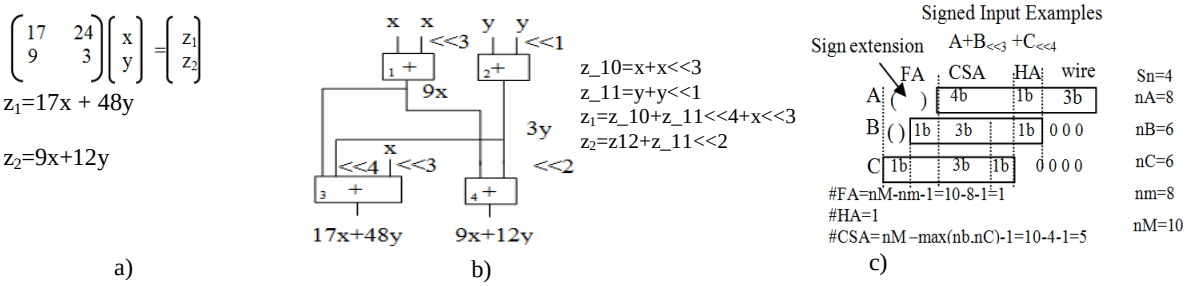

Figure 2. CMM and Gate Level representation examples: a) Linear transforms representation; b) with subexpression sharing using CSA; c) computation of the cost of A+B<<4+C<<3 operations under signed numbers using gate level metric[5].

2.1. CMM and Gate Level Approaches

It is presented in Fig. 2(a) a matrix representation, and their linear transform representation using subexpression sharing, where both the number of adders and the critical path can be reduced when using Carry Save Adders. The application of the gate level metric in FFTs using the more efficient CSA [5], where additions of three input operands at a time are taken into account, as can be observed in Fig. 2(b). As stated in [5], the cost function of an operation at the gate-level depends on: i) the type of operation (addition or subtraction); ii) the shifted input in a subtraction (minuend or subtrahend); iii) the number of shifts at the inputs; iv) the position of the operation in the architecture (influences the number of bits); and v) the range and type of numbers considered (unsigned or signed).

For example, in Fig. 2(c), is showed the cost computation of the block marked as 3 in Fig. 2(b), i.e., A+B<<4+C<<3 operations under signed numbers, using gate level metric [5], where Sn indicates the number of shifts; nA, nB and nC indicate the number of bits of inputs variables; nm and nM are the minimum and maximum number of bits respectively of the following relation: [min(nA+SA, nB+SB, nC+SC)] and [max(nA+SA, nB+SB, nC+SC)].

2.2. Related Work

Several solutions have been proposed the optimization of the multiplier in FFTs by using Constant Matrix Multiplication (CMM), gate level and pipelined implementations. However, only a few of them have been used decomposition of the constants as in this work. For this task, the work introduced in [2] proposes the optimization of the twiddle factors of a 32-point FFT using trigonometric identity, where the adders are replaced by multiplexers that determine the correct coefficient to be used. In the work of [4], different combinations of hybrid low-power techniques and use MCM for the sharing of various multipliers that are located at the same stage of the hybrid architectures are presented. However, the combination of CMM and gate level approaches has not been used as we have done in this work. Moreover, the presented results are limited to 32-point FFT.

In previous works [5], [6] we have combined the use of both approaches with CSA in order to reduce the complexity of the FFT. However, we had not presented an automatic algorithm that decomposes the constants efficiently as we have proposed in this work. As will be presented, the choice of the best set of decomposed coefficients combined with the use of CMM and gate level approaches enable the realization of larger efficient area and power FFTs.

3. The Proposed approach

For operation of an *N*-point FFT architecture, *N*/2 twiddle factors are needed. Using as example the implementation of a 32-point FFT, whose twiddle factors are defined as: 0, ±1, ±0.195, ±0.381, ±0.707, ±0.555, ±0.831, ±0.9213 and ±0.9803. The main idea consists on finding the set of constants that can be manipulated efficiently without affecting the structure and with smallest area through the decomposed constants. Fig. 3 shows the developed pseudo code for the decomposition of the constants.

Firstly, the method selects the possible constants that will be decomposed by the *constant_decomposition* function among all the twiddle factors of the 32-point FFT. This function reads the original constants, makes the difference between the maximum and minimum values of the constants (0.9803 – 0.195 = 0.7853) and chooses the constants that are close to theses value (0.707, 0.555, 0.831). These constants will be decomposed through the *decomposition_value function*.

```
1.for all the constant for decomposition{Constant_decomposition}{
2.   for all constants of input_constant{(0-N)}
3.    {Constant_value=decomposition_value(Input_constant)
4.       if (constant_value =input_constant) {
5.           save intermediate_constant;}
6.       else  {
7. Constant_value=decomposition_value((Input_Constant-Constant_value));
8.     save intermediate_constant;}}
```

Figure 3: Pseudo code for the decomposition of the constants.

Each selected constant is decomposed by the *decomposition_value* function that uses heuristic-based algorithm. Firstly the function sets the least value constant input. Independent of what twiddle factor is used for 32-points FFT, the first selected constant is ever the one that presents the less value (0.195). The second constant is chosen by the difference between the largest constant value and the constant that will be used for decomposition. For example, for 32-point FFT, using 0.707 as base for decomposition, thus the second constant result will be given by: (0.9803-0.707=0.273).

From this new constant it is checked if it is possible to find the one of the others needed constants by using only shifting. If not, the process is repeated using the found constants (0.195, 0.273, 0.707). These constants are named *intermediate_constant*. Following the same procedure, the constant 0.1855 is found by the difference between 0.381 and 0.195. After it is tried again to find a new constant by only shifting the constant 0.1855, where it is observed that the constant 0.555 can be found by multiplying 0.1855 by 3, and it is chosen as a new *intermediate_constant*. In terms of hardware cost it represents: (0.555 = 0.1855 * 2 (= 0.1855 << 1) + 0.1855). The new subset is added (0.381, 0.195, 0.1855, 0.707) and the process repeats until all the needed constants have been found. The main steps of the proposed algorithm can be seen in the third column of Tab 1.

The reduction of area is realized by *cost_function*, presented in (1), where the cost of each constant decomposed is determined. The constant with the less cost is chosen as base for decomposition. In (1), *intermediate_constant* means the new temporary constants achieved by the steps of the algorithm, *n* is the amount of new generated constant, and *shift_const* (*shif*$_i$) is the value of shifting for each constant. This method enables more flexibility, since it is possible to obtain new set of constants to be implemented with a larger amount of shifts what can leads to FFT implementations with less area.

$$cost\_function_i = \sum_{n=0}^{n-1} \left(intermediate\_constant_i * (1 - shif_i)\right) \quad (1)$$

The Tab.1 shows the cost values of the constants (0.707, 0.555, 0.831), where the constant 0.707 presents the less cost according to the *cost_function* in (1). By using the constant 0.707 as base for decomposition, the new set of decomposed constants (0.053; 0.124; 0.149; 0.1855; 0.195) is selected.

Tab.1 – COST FUNCTION FOR THE CHOICE OF SET CONSTANTS

Constants	**Cost Eq. (2)**	**Steps for the choice of New_constant using 0.707 as base for decomposition**	**New selected constants**
0.555	0.597	1° 0.1950	0.053; 0.124; 0.149; 0.1855; 0.195
0.707	**0.415** (0.053*(1-2) + 0.124*(1-0)+ 0.149*(1-0)+ 0.1855*(1-1)+ 0.195*(1-0))	2° 0.273=0.983-0.707 3° 0.1855=0.381-0.1950 4° 0.555=3*0.1855 5° 0.124=0.831-0.707 6° 0.149= 0.273-0.124 7° 0.214=0.921-0.707	
0.831	0.526	8° 4*0.0535=0.214	

The decomposition of the constant 0.707 results in five new constants: 0.053, 0.124, 0.149, 0.1855 and 0.195, whose representation in 16 bit-width are: 1735, 4062, 4881, 6078 and 6389 respectively by using Q15 format. These five new constants are controlled by a system which is responsible to indicate the right constants. Figure 4 shows the use of CMM in the implementation of constant decomposition (0.707), where adder/subtrators were implemented withe gate-level seen in section 2.1 and the control system of the correct twiddle factor is inserided.

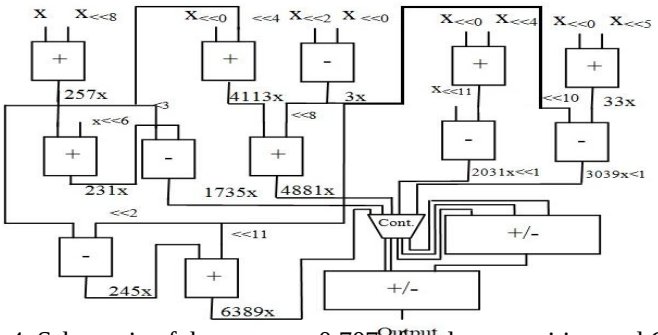

Figure 4: Schematic of the constant 0.707 after decomposition and CMM.

4. Architecture and Synthesis Result

This section shows the main results obtained with the 16 and 32 bit-width radix-2 DIT FFT architectures, which were implemented in hardware description language - VHDL. The results from the synthesized FFT architectures and comparisons with related work are presented. The logic synthesis was performed with the Design Compiler from Synopsys targeting the UMC 0.13um CMOS technology. Power was evaluated by Synopsys Design when 5000 input random vectors are applied. All the simulations are limited by the path delay of the circuits and the same type of adder and multiplexers were used to assure a fair comparison among solutions from the literature.

Tab.2 – RESULTS OF LOGIC SYNTHESIS FOR RADIX-2 DIT FFT ARCHITECTURE PROPOSED

Circuits	**Freq. max (MHZ)**	**The Proposed Solution**		**Solution [2]**	
		Area (nm2)	***Power (uW)***	***Area (nm2)***	***Power (uW)***
FFT 16P,16b	105	7319	7.1	9506	8.3
FFT 32P,16b	100	9891	8.4	10716	9,1

In order to show the impact on the complexity reduction of the butterfly by using our method, we have implemented the fully architecture of the 16 and 32-point FFT. The architectures were implemented in hardware description language VHDL, and the adders/subtractors components were implemented by taking into account the gate-level metric [6].

The Tab 2 presents reductions of up of 24% and 15% on area and power respectively, when compared with the solution of [2] for 16-point. However, for 32-point FFT the reduction was of up of 9% and 10% on area and power respectively. These better results presented by our solution are due to decrease in the number of components adders and multiplexers achieved with the decomposition of the constants.

Another aspect to be observed is that since our method always produces solutions with less complexity (in terms of less number of adders and multiplexers), thus our solutions can be synthesized faster for a range of different number of points for the 16-bit FFT architectures. By using MatLab tool on PC Pentium operating on 2.4GHz with 2GM of RAM memory it was possible to obtain the needed adders and multiplexers for the 16-bit 16-point FFT only 0.7s was enough. It proves the simplicity of the method that generates solutions with little computational effort.

5. Conclusions

The method for the decomposition of Twiddle factors introduced in the paper decreases area and power of FFT architectures. The results show the efficiency of the proposed approach by reducing the necessary number of adders and multiplexers in the multiplier part of the butterfly. The results also indicate that the power of the FFT architectures with our method can be further reduced using pipeline between the stages of the FFT. Although this work has been focused the implementation of radix-2 architectures, our method can be naturally applied to other configurations of FFT (radix-4, radix-8, split-radix).

6. References

[1] J. W. Cooley and J. W. Tukey, “An Algorithm for the Machine Calculation of Complex Fourier Series. Mathematics of Computation”, [S.l.], v.19, n.90, p.297–301, 1965.

[2] F. Qureshi, and O. Gustafsson. “Low-complexity reconfigurable complex constant multiplication for FFTs”, *Circuits and Systems. ISCAS* 2009. IEEE International Symposium on, pp. 1137-1140, May 2009.

[3] J.-E. Oh and M.-S. Lim, “New radix-2 to the 4th power pipeline FFT processor,”IEICE Trans. Electron, vol. E88-C, no. 8, pp. 1740–1764, Aug. 2005.

[4] W. Han, T. Arslan, A. T. Erdogan and M. Hasan, “High-performance low-power FFT cores,” ETRI Journal, vol. 30, no. 3, pp. 451–460, June 2008.

[5] S. Ghissoni, E. Costa, C.Lazzari, L.Aksoy, J. Monteiro.and R. Reis,. “Radix-2 Decimation in Time (DIT) Implementation Based on a Matrix-Multiple Constant Multiplication Approach”. In: *17th IEEE ICECS*, Athens (2010), pp. 859–862, Dec 2010.

[6] S. Ghissoni, E. Costa, J. Monteiro and R. Reis, “Combination of Constant Matrix Multiplication and Gate-Level Approaches for Area and Power Efficient Hybrid Radix-2 DIT FFT Realization In: *18th IEEE ICECS*, Beirute (2011), pp. 567–570, Dec 2011.