Reusing Smaller Optimized FFT Blocks for the Realization of Larger Power-Efficient Radix-2 FFTs

Sidinei Ghissoni
Federal University of Pampa - UNIPAMPA, Alegrete-RS, Brazil
sidineighissoni@unipampa.edu.br

Eduardo Costa
Catholic University of Pelotas - UCPel, Pelotas-RS, Brazil
eduardo.costa@ucpel.edu.br

Ricardo Reis
PGMICRO –Federal University of Rio Grande do Sul - UFRGS, Porto Alegre-RS, Brazil
reis@inf.ufrgs.br

***Abstract*— This paper reports the reuse of smaller optimized FFT (Fast Fourier Transform) blocks for the realization of larger power efficient radix-2 FFTs. The smaller FFT blocks use Constant Matrix Multiplication (CMM) method along its stages that are implemented with Carry Save Adders (CSA). The use of CMM at gate level enables the replacement of the multiplication operations by addition/subtractions and shifts for each stage of the real and imaginary parts of the FFT butterflies. The larger FFT is obtained through the composition of the optimized smaller FFT modules. Through a control unit, the partial decomposition of coefficients allows the computation of all coefficients necessary for the larger FFTs. The use of pipeline into the stages of the FFT enabled gains in both power and performance when compared with the previous non-pipelined solution. Moreover, the results showed that, when using pipeline, our solution is more delay and power efficient when compared with prominent works from the literature.**

Keywords — CMM; decomposition of coefficients; radix-2 FFT

I. INTRODUCTION

Fast Fourier Transform (FFT) is a largely used implementation of the Discrete Fourier Transform (DFT), because the FFT algorithm needs less computation due its recursive operator named butterfly. This operator performs the calculation of complex terms, which involves a multiplication of input data by appropriate coefficients. The radix-2 butterfly with decimation in time [1] is presented in (1) and (2), where the N^2 multiplications obtained in the direct DFT are reduced to $Nlog_2N$ multiplications. In (1) and (2), $W_N = e^{-j\frac{2\pi k}{N}}$ is the trigonometric coefficient (also known as twiddle factor) and G and H are even and odd points. All the other algorithms that were developed to reduce the computational complexity of the butterfly are based on [1].

$$X(K) = G(k) + W_N H(K) \quad (1)$$

$$X(K + N/2) = G(k) - W_N H(K) \quad (2)$$

The complexity of the design of FFT architectures is mainly related to the multiplication of the inputs by a large number of coefficients. The multiplication of a set of constants by a variable, *i.e.*, the Multiple Constant Multiplication (MCM) method has enabled significant impact on the design of Digital Signal Processing (DSP) area. In the MCM operation, each constant is implemented using only addition/subtraction and shift operations rather than using a general multiplier for each constant.

A large amount of algorithms has been proposed to optimize the multiplier block in digital filters by using the MCM approach [2]. However, only a few of them have been applied to FFT using a matrix, as in this work. Therefore, we propose to optimize a radix-2 with decimation time (DIT) butterfly by using the CMM (Constant Matrix Multiplication) approach along the stages in order to allow further implementation of an efficient FFT architecture. Firstly, the algorithm of [2] is used for the generation of the tree of adder/subtraction and shift operations. After that, the algorithm presented in [3] was adjusted in order to optimize the circuits by considering the use of more efficient Carry Save Adders - CSA [4], using gate-level metrics. By properly exploiting the use of CSA, the hardware implementation can be significantly optimized, because the opportunity of sharing common subexpressions is increased.

We introduce FFT architectures based on half adders (HAs) and full adders (FAs) for addition and subtraction operations under unsigned and signed input models. Moreover, the controlling of multiplexers between the stages of the FFT enables the reduction of complexity of the radix-2 butterfly of the FFT with decimation in time, by reusing the architecture as much as possible. In this work, we present variations of 16 and 32-bit width FFTs. Smaller optimized 16 to 128-point FFTs are used in order to generate larger 64 to 2048-point FFTs. The architectures were described in VHDL, and the syntheses were performed with the Synopsys Design Compiler tool for the XFAB 180nm technology.

The main contribution of this work is the reuse of smaller FFT blocks aiming at the design of power-efficient FFTs of larger sizes. This contribution has enabled us to increase the comparisons of our solutions of larger FFTs with prominent works from the literature, in terms of area, power and delay. Moreover, for the first time, we present a pipelined solution

9781467394192/15/$31.00 ©2015 IEEE

that enabled the increase of both power efficiency and performance of the FFTs, when compared with our previous solution.

This paper shows in the next section an overview of prominent FFT solutions proposed in literature. In Section III, the CMM approach with CSA, as well as the decomposition of coefficients approach are presented. In Section IV we present the scheme for the proposed large FFT solution. The main results of the work and comparisons with the literature are presented in Section V. Finally, Section VI presents some conclusions of this work.

II. RELATED WORK

In the last years, a reasonable amount of researches has been presented with the objective of increasing the efficiency in area of FFT architectures. Solutions such as multiplierless implementation [5], and implementation based on multirate signal processing and asynchronous circuit technology [6] are proposed in order to implement efficient fixed-point FFT architectures, where the multiplications realized in the butterflies are replaced by additions, subtractions and shifts. The mentioned methods minimize the number of additions and subtractions, while achieving a specified level of accuracy.

Parallel architectures for high throughput and power efficient FFT cores are proposed in [7]. The implementation of a FFT with reduced number of points is presented in [8], by optimizing the twiddle factors using trigonometric identity, where the adders are replaced by multiplexers. The use of CMM for FFT is presented in [3], where the multiplication operations are realized by adder, subtraction and shifting operation. However, the use of decomposition of the twiddle factors was not taken into account as in our work.

In the same scope of our work, a system level modeling for configurable FFT architecture for system-on-chip design is proposed in [9]. In this method, customized reconfigurable IP cores that use different fixed butterflies are included.

In terms of the use of pipelined solutions, a high throughput energy efficient parallel FFT architecture based on Cooley-Tukey algorithm is proposed in [10]. Multiple pipeline FFT processors, using time-multiplexing, are used to perform FFT computation tasks in parallel. This design realizes high performance using task-level parallelism and avoids complex routing. Furthermore, the method reduces the memory power consumption, by periodic memory activation (PMA). In the same scope, the work of [11] presents a pipeline FFT architecture, which supports FFT lengths of power-of-two multiple of three. The architecture is basically single delay feedback structure followed by radix-3 computation unit. The proposed architecture is memory optimal for N-point transform, where only N - 1 memory locations are needed.

An efficient folded pipelined architecture is proposed in [12], where folded architecture, and folding transformations are used to design FFT architectures with reduced number of functional units. In the folding transformation, many butterflies in the same column can be mapped to one butterfly unit.

In a previous work [4], we have implemented a 32-point FFT by reusing only an 8-point FFT architecture with CMM approach. The reuse of smaller blocks to implement FFTs architectures, based on CMM associated with gate level metric, was presented in [14], where the coefficients were decomposed into the different stages of the FFTs. The reordering of coefficients for the increase of performance of large FFTs architectures was presented in [15].

In this work, the reuse of different sizes of smaller FFTs blocks is increased in order to implement larger FFT architectures. In this way, we could prove the efficiency of the proposed method of twiddle factors decomposition for the design of larger power efficient FFTs. Moreover, the use of pipeline between the stages of the blocks of FFT, has enabled us to improve the performance of our solution. Therefore, with the combination of maximizing the reuse of the optimized smaller blocks of FFT, and the use of pipeline, we were able to increase the comparisons of our larger FFTs with others solutions from the literature, as will be presented later.

III. CMM APPROACH AND DECOMPOSITION OF TWIDDLE FACTORS

In this section, the decomposition of trigonometric coefficients as well as an example of the CMM representation for the implementation of the FFT architecture are presented. Note that both strategies are used in the smaller FFT blocks that are reused for the implementation of larger FFT instances.

A. CMM Representation with CSA

The CMM method, also called CMVM (Constant Matrix-Vector Multiplication) [3], consists on the multiplication of an input vector by a matrix with constant coefficients. The matrix of constants specifies how the outputs are obtained from the linear transformations of the inputs.

In our implementation the resultant terms of each butterfly were used for the decomposition of coefficients [4] and CMM representation [3], where the results are composed by adders/subtractors and shifts with a smaller amount of components. An example of CMM is presented in Fig. 1(a), where a matrix representation (3), and their linear transform representation using subexpression sharing are shown. Fig. 1(b) shows the subexpression sharing with Carry Save Adder - CSA.

Note that the implementation with subexpression sharing and CSA, Fig. 1(b), requires one adder less than the implementation of Fig. 1(a), what may implies on less area, due to the use of gate level approach [4], and less power, due to the smaller critical path. With the use of gate level approach it is possible to decrease the number of addition blocks, because the implementation of coefficients with shifting is allowed. The use of gate level enables the use of CSA and Ripple Carry Adders (RCA), what is important to reduce area and critical path of the implemented addition block. It is also important to highlight that the implementation with shifting does not require any additional hardware, since only wires are needed.

$$\begin{pmatrix} 17 & 24 \\ 9 & 6 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}$$

$$z_1 = 17x + 24y$$
$$z2 = 9x + 6y \quad (3)$$

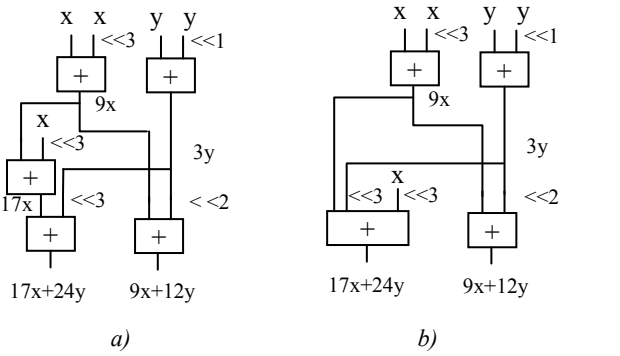

Fig. 1. Implementation of the linear transforms of (3): a) schematic with subexpresion sharing; b) subexpression sharing using Carry Save Adder [14].

B. Decomposition of Twiddle Factors

The decomposition of twiddle factors is based on the algorithm presented in [4]. This algorithm checks what the best constants could be implemented in order to obtain the news constants of the each stages of the parallel FFT, but that still attends all the coefficients of large FFTs. The pseudo-code of the algorithm is presented in Fig. 2.

The function **decomposition_value** presents a heuristic that first sets the least constant input. The following the function determines the other constants from the difference between the largest constant and the constant used as basis. After that, the new constant is determined by the difference between the constant used as basis, and the previous two ones. From this task, two results are presented, where the first one is the result of function **constant_value** that is the value of **new_constant** when subjected to shifts and sums/ subtractions, and the second one is the **new_constant**.

```
1.for all the constant for decomposition{Constant_decomposition}
2.  {
3.    for all constants of input_constant{(0-n)} –where n is the number of
coefficientes
4.    {
5.    Constant_value=decomposition_value(Input_constant)
6.       if (constant_value =input_constant) {
7.           save new_constant;
8.       }
9.       else  {
10.          Constant_value=decomposition_value((Input_Constant-
Constant_value));
11.      save new_constant;
12.    }
13.   }
14.  }
```

Fig. 2. Pseudo-code of the algorithm for the decomposition of constant [4].

Using as example the implementation of a parallel 32-point FFT, whose twiddle factors of the last stage are defined as: 0, ±1, ±0.195, ±0.381, ±0.707, ±0.555, ±0.831, ±0.9213 and ±0.9803. After the process of decomposition of the coefficients, a new set of constants (0.053; 0.124; 0.149; 0.1855; 0.195) is used for implementing this stage [4].

IV. THE SCHEME FOR IMPLEMENTING LARGE FFT ARCHITECTURES

In this work, the method proposed in [4] is extended for the implementation of large FFT architectures. We have reused different sizes of smaller FFTs blocks in order to implement larger FFT architectures on fixed point by using Q31 and Q15 representations [13].

First of all, the scheme consists on defining the number of points of the FFT. After that, the parallel FFT architecture that will be used is defined. In the next step, the method for decomposition of twiddle factors analyze what are the twiddle factors that can be reused among all the butterflies of the FFT used in parallel as basis. The algorithm first selects the constants that will be decomposed for each stage based on the cost of implementation. After that, the implementation is realized in two main steps: i) the first step applies the CMM method to the stages with no constants selected for decomposition; ii) the second step is the implementation of the new values obtained from the constant decomposition, where CMM or MCM can be used depending on the number of selected values. The final architecture is implemented with both ripple carry adder (RCA) and carry save adder (CSA) blocks using gate-level metric [4]. A control system based on multiplexers is inserted in order to allow the selection of the constants. The flowchart of Fig. 3, summarizes the implementation of large FFT by using CMM and decomposition of coefficients.

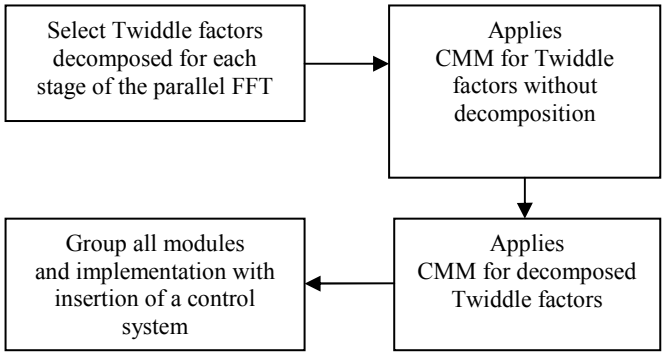

Fig. 3. Flowchart of the algorithm for decomposition of Twiddle factors and implementation of large FFTs.

Fig. 4 shows the schematic of an example for the realization of 128-point FFT, based on 32-point FFTs. For the operation of the 128-point FFT, four different stages of smaller 32-point FFT architectures are used. The blocks A1-A4, B1-B4, C1-C4, and D1-D4 represent parallel blocks of each stage of the FFT. Note that the blocks are separated by stages of pipeline, represented by four barriers of buffers (BF1-BF4). Each parallel block presents 25% of coefficients

needed for the total operation of the large FFT. The blocks are responsible for real and imaginary operations, where the odd blocks (1,3) are responsible for the real operations and the even blocks (2,4) realize the imaginary operations. After the partial values have been calculated, they are stored in the respective pipeline barrier, and they will be reused according to the reuse of the next stage. Finally, when the operations are finished, the final results are sent to the output registers.

Fig. 5 shows all the outputs of the decomposed coefficients of A1 block defined as: 0, ±1, ±0.195, ±0.381, ±0.707, ±0.555, ±0.831, ±0.9213 and ±0.9803 implemented with CMM [4] and quantized with Q15 format. The number of coefficients for each block depends on the achieved sharing by the CMM [4].

As an example, it can be observed that the twiddle factor 0.707 (46298 in Q15 format) is calculated from the decomposed constants [3473 + (12156 + 12779) + (8126 + 9764)].

Note that for the implementation of 128-point FFT, firstly its 64 different coefficients are distributed among 16 blocks of 32-point FFT that are used as basis in both real and imaginary blocks. After that, the repetitive coefficients are eliminated, and each stage of 32-point FFT will receive the remaining coefficients that will be reused for the implementation of the 128-point FFT.

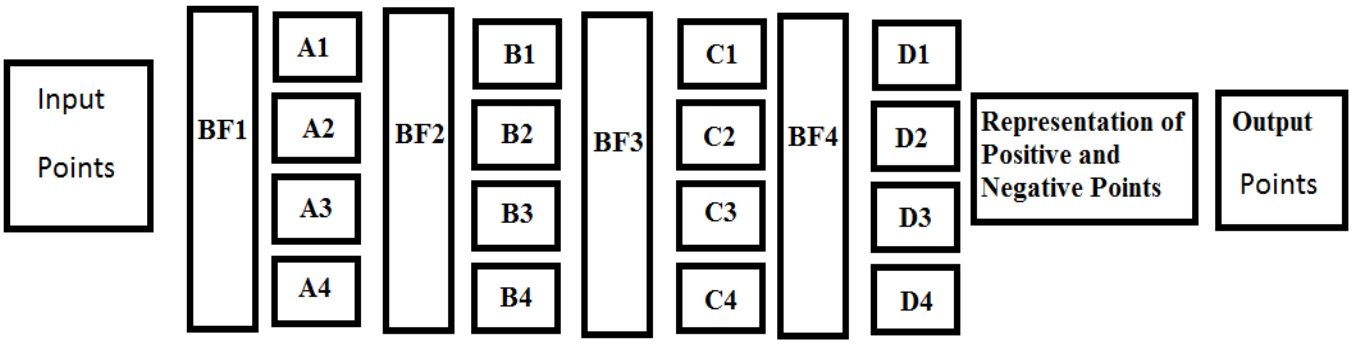

Fig. 4. Schematic of 128-point FFT architecture with four stages of pipeline using smaller parallel blocks.

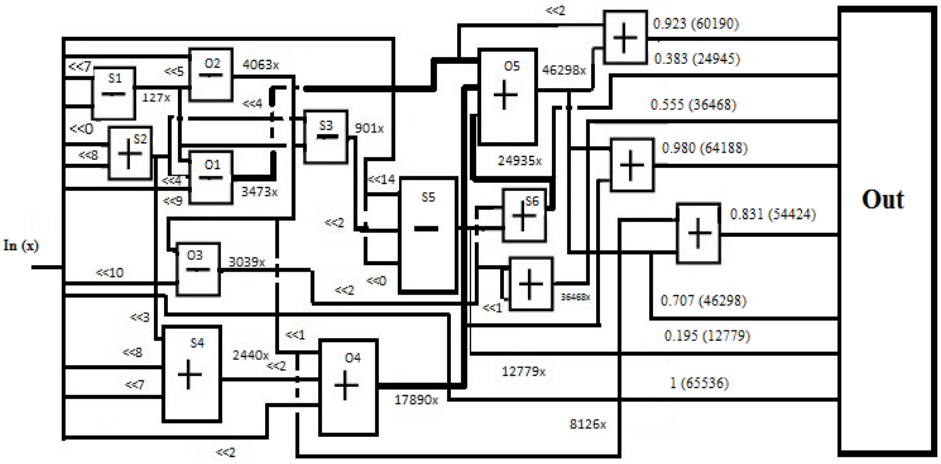

Figure 5. Schematic of the constant decomposition for 8 coefficients of A1 Block of the Fig. 4. The constants are quantized for 16 bit-width associated with CMM approach.[14]

V. Results

This section shows the main results of area, delay and power obtained with the radix-2 DIT FFT, ranging from 64 to 2048 points, whose, implementations were realized by reusing smaller blocks of 16, 32, 64 and 128 point-FFT described in hardware description language - VHDL. For the fair comparisons, we implemented the works from the literature, based on their descriptions presented on the papers. Logic syntheses were performed with the Synopsys Design Compiler tool for the XFAB 180nm technology. The power was also evaluated by the Design Power tool of Synopsys, where for the simulations 3,000 random vectors were applied for the inputs.

A. Results for the Pipelined and Non-Pipelined FFT Architectures

For this analysis the logic synthesis was restricted the limitations of the critical path of faster architecture for same large FFT/basis. Thus it is possible verified the advantages and disadvantages of method. Table I presents the results of our method for both pipelined and non-pipelined solutions. For the presented results, the logic synthesis constraint was set to the critical path of the faster architecture in terms of the relation large FFT/basis. This was done in order to verify the advantages and disadvantages of the proposed method.

As can be observed, the pipelined solution achieves, in average, gains of up to 2% in power and of up 7% in delay, when compared to the non-pipelined one, at the cost of an increase in area of up to 3%. In fact, the increase in area in the pipelined solution is justified by the insertion of registers. In fact, the pipelined solution consumes less power due to the smaller critical path enabled by the barriers of registers. This also enables the reduction of glitching activity produced at each stage of the smaller parallel FFT blocks. It is also important to mention that the decomposition of coefficients technique enables a higher correlation between the coefficients, since the original coefficients are decomposed into more optimized ones, what contributes for the reduction of the switching activity into each FFT block.

B. Comparisons with the Literature

Table II presents area, delay and power results of the proposed pipelined solution and prominent works from the literature.

As can be observed in Table II, our pipelined solutions are more delay and power efficient for all the architectures. When compared with [9], reductions of up to 24% in power consumption and 38% in area are achieved. It occurs because we have used CMM approach into each FFT block, what reduces the number of coefficients used. Moreover, in our solution, the size of the blocks are not fixed, since they are implemented at logic level, and therefore the size of the block is given according to the maximum wide of bits needed in the word.

TABLE I. RESULTS FOR THE PIPELINED AND NON-PIPELINED FFT ARCHITECTURES

	Pipelined Solution			**Non-Pipelined Solution**		
FFT/Basis	***Area (um2)***	***Power (uW)***	***Delay (ps)***	***Area (um2)***	***Power (uW)***	***Delay (ps)***
64/16	223711	245.5	18.1	217962	246.7	19.4
128/32	376722	472.1	25.2	375292	473.8	27.3
256/32	120019	1453.1	41.3	1187002	1460.9	42.5
512/32	2274107	2701.8	43.6	2255303	2775.7	44.7
1024/32	4319901	5215.2	61.2	4285077	5273.8	63.1
2048/32	8370022	10101.2	73.2	8141646	10200.3	74.9
256/64	1321471	1153.2	75.2	1238463	1168.7	76.9
512/64	2431216	2203.5	77.1	2353080	2220.6	78.4
1024/64	4612127	4076.2	82.5	4470853	4108.0	83.8
2048/64	8613225	7492.3	84.3	8494621	7517.7	85.9
512/128	2251127	1701.1	86.2	2229234	1732.0	87.5
1024/128	4519206	3610.5	89.1	4353199	3686.2	91.6
2048/128	8492659	6509.5	90.8	8405204	6572.8	92.3

TABLE II. RESULTS FOR THE FFT ARCHITECTURES FOR THE PROPOSED METHOD

	Proposed Pipelined Method			Ahmadinia [9]			Macleod [5]			Han [7]		
FFT/Basis	***Area (um2)***	***Power (uW)***	***Delay (ps)***	***Area (um2)***	***Power (uW)***	***Delay (ps)***	***Area (um2)***	***Power (uW)***	***Delay (ps)***	***Area (um2)***	***Power (uW)***	***Delay (ps)***
64/16	223711	245.5	18.1	2619990	291.1	19.1	270577	288.6	19.1	223881	251.3	19.2
128/32	376722	472.1	25.2	458870	562.6	27.1	476792	567.5	27.3	376611	482.1	27.4
256/32	120019	1453.1	41.3	1663562	1763.2	42.2	1755673	1781.7	42.4	1199910	1512.1	42.7
512/32	2274107	2701.8	43.6	2593599	3219.8	45,1	2706364	3638.4	44.6	2277608	2882.3	45.1
1024/32	4319901	5215.2	61.2	5056391	6223.2	62.1	5356347	7032.1	61.4	4318124	5322.2	62.9
2048/32	8370022	101001.2	73.2	9851393	11924.2	76.3	10177058	13474.3	74.1	8347617	10881.7	74.7
256/64	1321471	1153.2	75.2	1510926	1367.4	77.4	1461387	1545.2	76.2	1317057	1181.1	76.3
512/64	2431216	2203.5	77.1	2917820	2620.3	78.1	2894289	2960.9	77.1	2438143	2361.4	77.9
1024/64	4612127	4076.2	82.5	5588567	4970.7	83.1	5499150	5616.9	83.1	4612650	4203.4	83.1
2048/64	8613225	7492.3	84.3	10703224	9246.8	86.2	10788170	10448.9	85.7	8531974	7619.4	84.2
512/128	2251127	1701.1	86.2	2608204	2009.2	88.1	2853420	2270.2	87.2	2257136	1789.3	86.6
1024/128	4519206	3610.5	89.1	5702691	4460.2	92.2	5528563	5040.0	91.7	449238	3736.4	89.7
2048/128	8492659	6509.5	90.8	11347026	8084.6	93.3	10926766	9135.6	92.4	8468196	6619.2	90.9

As can be also observed in Table II, with our solution, reductions of up to 31% in power consumption and 19% in area are achieved, when compared with [5]. In fact, as in our solution, in [5] the multiplications are also replaced by only additions, subtractions and shifting. However, our method decomposes the coefficients before applying CMM approach, what increases the opportunity of finding the more optimized coefficients. Therefore, the decomposition of the coefficients into more suitable ones by selecting the most indicated intermediate constants, enables our FFTs to be implemented with less area and less power consumption, when compared with the architectures of literature. It is enforced when comparing our solution with [7], where our solution is slightly more delay and more power efficient, with gains close up to 8% and 7% in delay and power respectively. It occurs because our solution uses adders and subtractors at gate level that guarantees the less number of half adders and full adders used. As in [7], the pipelined strategy is also used in our work, and therefore area values are almost the same. In the case where our solution presents more area than [7] (in our worst case, the area was approximately 1% higher) it occurs due to the fact that the sharing of coefficients depends on the amount of parallel blocks used. The increase of number of points of the FFTs demands major decomposition of coefficients into the architecture, what consequently generates higher area.

The previous results presented in Table I and II can be summarized, in Fig. 5, where we can see the variations of number of points and the size of the smaller blocks of FFTs used as basis for the implementation of larger FFTs. As can be observed in Fig. 5(a), (b) and (c), our solution is more efficient in area and power respectively, when compared with the solutions of the literature [5] and [9], mainly for 256-point FFT that used smaller blocks of 32-point FFT for its implementation. The main advantage of our method is that the decomposition of coefficients allows that, when implemented with CMM, lesser adder and subtractors components are needed due to the fact that the decomposition method put in order of importance the implementation with shifting. It was observed in Table I and Table II that the result of delay in our non-pipelined FFT architecture was also higher than [5] and [7] for most of the cases. This is due to the fact that our implementations use more components in series than the solution of [5], what causes an increase in the critical path of the architecture.

However, when the pipeline strategy is taken into account, our solution is more efficient than the works from the literature. It can be enforced when comparing our solutions with the work of [7]. As in [7] the pipeline strategy is used, therefore its solution presented less delay value than our non-pipelined FFTs, for all structures, as observed in Fig. 5(c). However, when the pipelined strategies are compared, our solutions present higher performance and less power consumption than [7].

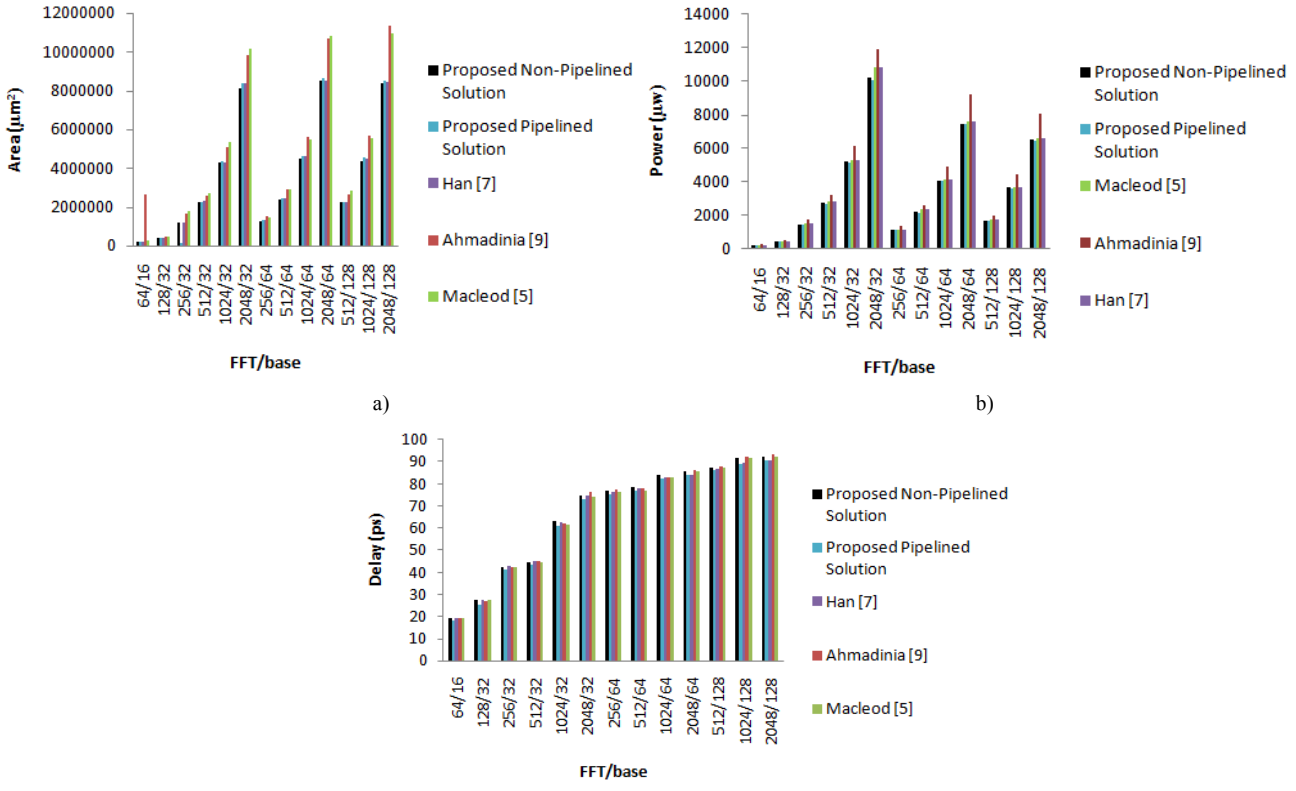

FIG. 5. RESULTS COMPARISONS FOR THE PROPOSED FFTS. A) AREA; B) POWER; C)DELAY.

VI. CONCLUSIONS AND FUTURE WORKS

In this work the design of large FFT architectures were presented based on the reuse of smaller optimized FFT blocks. The smaller FFTs used CMM and gate level metric approaches. The pipeline strategy was also taken into account. The main results showed that the combination of the optimized smaller FFT blocks and the use of pipeline contributed to the realization of large power efficient FFTs. The obtained results showed that our pipelined FFT solutions are more performance and power efficient when compared with prominent solutions from the literature. As future works we intend to improve the use of pipeline for increasing the gains in performance of the proposed FFTs. We also intend to synthesize our architectures in a more recent technology in order to verify the influence of leakage power on the proposed FFT solutions.

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] L. Aksoy, E. Costa, P. Flores and Monteiro J."'Exact and Approximate Algorithms for the Optimization of Area and Delay in Multiple Constant Multiplications", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 27(6), 1013-1026, May 2007.

[3] L. Aksoy, E. Costa, P. Flores, and J. Monteiro, "Optimization Algorithms for the Multiplierless Realization of Linear Transforms," in ACM Transactions on Design Automation of Electronic Systems, vol. 17, no, 1, Jan. 2012, pp. 1-27.

[4] S. Ghissoni et al. Radix-2 Decimation in Time (DIT) Implementation Based on a Matrix-Multiple Constant Multiplication Approach. In: 17th IEEE ICECS. 2010.

[5] M D Macleod, " Multiplierless Implementation of Rotators and FFTs ", EURASIP Journal on Applied Signal Processing, 2005:17 (2005) pp 2903-2910, Sept 2005.

[6] K. Stevens and B. Suter, "A Mathematical Approach to a Low Power FFT Architecture," IEEE Int'l Symp. on Circuits and Systems, vol. 2, 1998, pp. 21-24

[7] 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.

[8] 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.

[9] A. Ahmadinia, B.Ahmad, T. Arslan, "System Level Modelling of Reconfigurable FFT Architecture for System-on-Chip Design," Second NASAIESA Conference on Adaptive Hardware and Systems, pp. 169-175, 2007.

[10] R. Chen, N. Park, V. K. Prasanna."High throughput energy efficient parallelFFTarchitecture on FPGAs. High Performance Extreme Computing Conference (HPEC), 2013 IEEE, pp. 1-6, 2013.

[11] Inkeun Cho;Patyk, T.;Guevorkian, D.;Takala, J. Bhattacharyya, S. "Pipelined FFT for Wireless Communications Supporting 128-2048 / 1536 -Point Transforms". Global Conference on Signal and Information Processing (GlobalSIP), 2013, pp. 1242 – 1245, 2013.

[12] N.S. Shymna and A. R.Krishna. An Efficient Folded Pipelined Architecture For Fast Fourier Transform Using Cordie Algorithm,

Advanced Communication Control and Computing Technologies (ICACCCT), 2014 International Conference on, pp. 462 – 467, 2014.

[13] Woon-Seng Gan and S. M. Kuo, Embedded Signal Processing with the Micro Signal Architecture, Wiley-IEEE Press; 1° edition 2007.

[14] Ghissoni, S.; Costa, E.; Monteiro, J.; Reis, R., "Efficient area and power multiplication part of FFT based on twiddle factor decomposition," Electronics, Circuits and Systems (ICECS), 2012 19th IEEE International Conference on , vol., no., pp.657,660, 9-12 Dec. 2012

[15] Ghissoni, S.; da Costa, E.A.C.; Goncalves da Luz, A., "Implementation of power efficient multicore FFT datapaths by reordering the twiddle factors," Very Large Scale Integration (VLSI-SoC), 2014 22nd International Conference on , vol., no., pp.1,6, 6-8 Oct. 2014