

A NOVEL LOW POWER PIPELINED FFT BASED ON SUBEXPRESSION SHARING FOR WIRELESS LAN APPLICATIONS

Wei Han, T. Arslan, A. T. Erdogan and M. Hasan

School of Engineering and Electronics
University of Edinburgh, Edinburgh UK

w.han@ed.ac.uk T.Arslan@ed.ac.uk

ABSTRACT

This paper proposes a novel low power multiplier-less radix-4 Single-path Delay Commutator (R4SDC) FFT processor architecture for wireless LAN (IEEE 802.11 standard) applications, where short FFTs are utilised in the implementation of the physical layer. The multiplier-less architecture uses shift and addition operations to realize complex multiplications. By combining a new commutator architecture, and low power butterfly architectures with this approach, the resulting power savings are around 19% and 35% for 64-point and 16-point radix-4 FFTs respectively, as compared to a conventional FFT architecture based on non-Booth coded Wallace tree multiplier.

1. INTRODUCTION

The FFT processor is a critical block in those fields based on orthogonal frequency division multiplexing (OFDM) technology, such as wireless LAN (IEEE 802.11) and Digital Video Broadcasting (DVB). For the portability requirement, the need of low power FFT architecture for telecommunication systems in portable form is attached more and more importance.

Due to the nature of non-stopping processing on the same clock frequency of sampling data, pipelined FFT is preferred especially for a high throughput demand or low power solution [1]. In the pipelined architectures, the commutator and the complex multiplier at each stage contribute a dominating part of the whole power consumption. A number of researchers have explored the scope of low power implementation for FFT processors. In [2], the authors combined voltage overscaling and algorithmic noise-tolerance techniques to reduce power consumption in butterfly blocks. The author of [3] presented a low power cache-memory architecture by using an algorithm offering good data locality to increase speed and energy efficiency. In [4], the authors proposed

a new radix-2/4/8 algorithm, which can effectively minimize the number of complex multiplications in pipelined FFTs. A novel ordering based low power pipelined radix-4 FFT was presented in [5]. Coefficient ordering reduces the switching activity between successive coefficients fed to the complex multiplier and hence leads to low power consumption [5]. However, as far as we know, until now, there are no explorations on the application of common subexpression sharing to FFTs. In the past, some researches used shifters and adders to replace the complex multiplication by some special constant coefficients. In [4], the authors used 12 additions to realize the complex multiplication by $\sqrt{2}/2(1 \pm i)$. The authors in [6] employed seven shift-and-add units to carry out seven multipliers in parallel, each by a constant coefficient. However, [4] and [6] do not refer to common subexpression sharing.

This paper explores the application of common subexpression sharing across coefficients to the second stage of 64-point or the first stage of 16-point FFTs, based on a popular pipelined FFT, R4SDC [7]. Complex multiplications are replaced by the minimum number of shift and addition operations. Hence, both area and power consumption for the multiplier unit are reduced. This new FFT processor architecture also employs a new commutator architecture based on dual port RAMs proposed in our recent work [8] and improved low power butterfly elements [9].

2. ALGORITHM

2.1. R4SDC FFT algorithm

The Discrete Fourier Transform (DFT) of N complex data points $x(n)$ is defined by

$$X(k) = \sum_{n=0}^{N-1} x(n)W_N^{nk} \quad k = 0, 1, \dots, N-1; \quad (1)$$

where $W_N = e^{-j(2\pi/N)}$, W_N is twiddle factor.

In [7], the authors presented the R4SDC pipelined FFT algorithm for word-sequential data. For radix r_1 , equation 1 can be written as follows:

$$X(k) = \sum_{q=0}^{N_1-1} W_N^{q1k} \sum_{p=0}^{r_1-1} x(N_1p + q_1) W_{r_1}^{pk} \quad (2)$$

The N -point DFT in (2) can be decomposed into v stages where $N = r_1 r_2 \dots r_v$. The final stage is defined by

$$\begin{aligned} X(r_1 r_2 \dots r_{v-1} m_v + r_1 r_2 \dots r_{v-2} m_{v-1} + \dots + r_1 m_2 + m_1) \\ = \sum_{q_{v-1}=0}^{r_v-1} x_{v-1}(q_{v-1}, m_{v-1}) W_{r_v}^{q_{v-1} m_v} \end{aligned} \quad (3)$$

While intermediate stages (t) are given by the recursive equation 4 below [7]:

$$x_t(q_t, m_t) = W_{N_{t-1}}^{q_t m_t} \sum_{p=0}^{r_t-1} x_{t-1}(N_{t-1} p + q_t, m_{t-1}) W_{r_t}^{p m_t} \quad (4)$$

Where, for both (3) and (4)

$$0 \leq q_i \leq N_i - 1, \quad 2 \leq i \leq v \text{ and } N_t = N / (r_1 r_2 \dots r_v)$$

$$2 \leq t \leq v - 1, \quad 0 \leq m_i \leq r_i - 1$$

When $r_1 = 4$, we use 16-point FFT whose flow graph based on the above equations can be seen in Figure 1. As can be seen, each open circle denotes a summation while the dots define the stage borders. The number inside the open circle is the value of m_1 (for the first stage) or m_2 (for the second stage). The number outside the open circle is the twiddle factor used. N -point pipelined FFT processor based on this architecture is shown in Figure 2. It achieves 75% utilization of the complex multiplier and 100% utilization of the butterfly element respectively.

2.2. Common Subexpression Sharing

Common subexpression sharing shares the subexpression among several multiplication-accumulation operations in order to reduce the total number of operations [10]. This approach is very effective for reducing the hardware cost of multiple constant multiplications, especially for the filter-like operation. For example, for a 3-tap FIR filter, the output $Y(2)$ is given as follow.

$$Y(2) = \sum_{i=0}^2 A_i \times X_{n-i} \quad (6)$$

The weights A_i are the filter coefficients. Suppose the coefficients are given as $A_0 = 00111011$, $A_1 = 00101011$, and $A_2 = 10110011$. The coefficients are represented in two's complement format. According to equation 6, $Y(2) = A_0 \times X_2 + A_1 \times X_1 + A_2 \times X_0$. Using shifts and additions to replace the multiplications, gives:

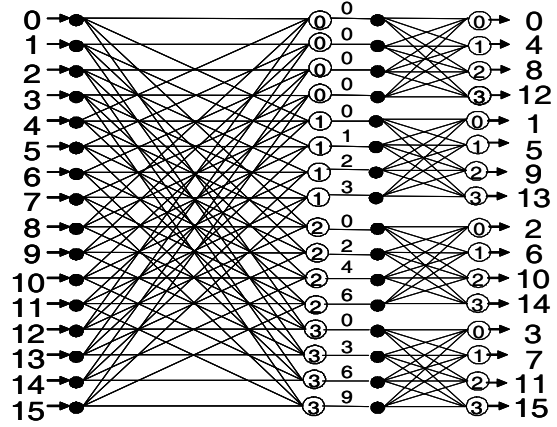


Fig.1. Signal flow graph of a radix-4 16-point FFT [5]

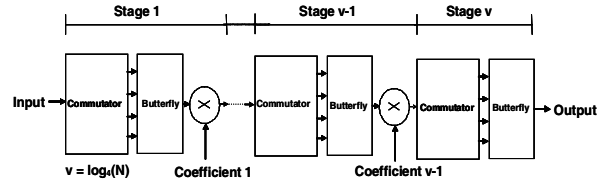


Fig.2. N -point R4SDC pipelined FFT processor architecture [9]

$Y(2) = X_2 + X_2 \ll 1 + X_2 \ll 3 + X_2 \ll 4 + X_2 \ll 5 + X_1 \ll 1 + X_1 \ll 3 + X_3 \ll 5 + X_0 + X_0 \ll 1 + X_0 \ll 4 + X_0 \ll 5 - X_0 \ll 7$. The computation requires 12 additions, 1 subtraction and 11 shifts. However, if pre-computing $X_{02} = X_0 + X_2$; $X_{12} = X_1 + X_2$; $X_{012} = X_{12} + X_0$, the output can be shown as: $Y(2) = X_{012} + X_{02} \ll 4 + X_{012} \ll 1 + X_{012} \ll 5 + X_{12} \ll 3 - X_0 \ll 7$. This computation only needs 7 additions, 1 subtraction and 5 shifts. X_{02}, X_{12}, X_{012} are the common subexpressions for this case. From the above examples, it can be shown that common subexpression sharing can reduce the number of additions and subtractions from 13 to 8 (38% reduction). In Section III, we discuss how to use common subexpression sharing in R4SDC FFT.

2.3. Canonic Signed Digit (CSD)

It is common to use the redundancy of signed digit code to replace the conventional multiplier digits, such that addition operations in a multiplication can be reduced with the increase of average shift length across the zeros in the multiplier [11]. Canonical Signed-Digit (CSD) is one widely used signed digit approach. In CSD code of a number, each bit is set to 0, 1 or -1 and no two consecutive bits are nonzero. The advantage of CSD form is that no value has more than $(N+1)/2$ nonzero bits, often fewer, and so the multiplication by a constant requires no more than that number of additions for its

implementation. An algorithm for transform from two's complement form to CSD form can be found in [11].

3. IMPLEMENTATION

3.1. Multiplier-less in R4SDC FFT

In R4SDC FFT, the conventional complex multiplier consists of four real multipliers, one adder and one subtracter. However, since the complex coefficients for all stages can be pre-computed, we can apply shift and addition operations with common subexpression sharing to those stages where the number of coefficients is limited. For example, the number of coefficients for the second stage of 64-point or the first stage of 16-point FFTs is 16. These coefficients are shown in Table1. A close observation of these coefficients reveals that seven of these are (7fff,0000), one is (0000,8000) which are the quantized representation for (1,0) and (0,-1) in 16-bit two's complement format respectively. In each set, the first entry corresponds to the cosine function (the real part, W_r) and the second one corresponds to the sine function (the imaginary part, W_i). For the trivial coefficients (7fff,0000) and (0000,8000), the complex multiplication is not necessary. Data can directly pass through the multiplier unit without any multiplication, when data is multiplied with (7fff,0000). Only an additional unit, which swaps the real and imaginary parts of input data, and inverts the imaginary part, is needed for those data by (0000,8000).

Table1: The coefficients for 16-point R4SDC FFT

Coefficient sequence m1=0,1	Original quantized coefficient	Coefficient sequence m1=2,3	Original quantized coefficient
W_0	7fff,0000	W_0	7fff,0000
W_0	7fff,0000	W_2	5a82,a57d
W_0	7fff,0000	W_4	0000,8000
W_0	7fff,0000	W_6	a57d,a57d
W_0	7fff,0000	W_0	7fff,0000
W_1	7641,cf04	W_3	30fb,89be
W_2	5a82,a57d	W_6	a57d,a57d
W_3	30fb,89be	W_9	89be,30fb

The rest of the coefficients are composed of only 6 constants (7641, 5a82, 30fb, a57d, 89be, cf04). However, one can see that only 3 of these constants (7641, 5a82 and 30fb) would be enough to implement all of the coefficients. For example, a multiplication with the constant a57d could be realized by first multiplying the data with 5a83, and then two's complementing the result. Note that a multiplication by the constant 5a82 already exists. Therefore, the multiplication with the constant

5a83 can simply be obtained by adding the data to the already existing multiplication with 5a82. The other two constants (89be and cf04) can be realized in a similar manner, using constants 7641 and 30fb respectively. 5a82 is represented by two's complement format, 7641 and 30fb are represented by CSD format as follow,

5a82	0101101010000010
7641	1000-10-1001000001
30fb	010-1000100000-10-1

The mixed use of CSD and two's complement is for minimizing the number of addition/shift operations. We can use shifters and adders based on the three constants to carry out those nontrivial complex multiplications as shown below:

$$5a82X = 5X \ll 12 + 5X \ll 9 + 65X \ll 1$$

$$7641X = X \ll 15 + 65X - 5X \ll 9$$

$$30fbX = 65X \ll 8 - X \ll 12 - 5X$$

where X means input data. The common subexpressions for the three constants are 101 (5) and 1000001 (65).

Figure 3 shows the shift-and-addition module for the three constants in 16-point FFT. The module carries out the multiplications in which the real part (X_r) or imaginary part (X_i) of input data will be multiplied with W_r and W_i respectively. The shift-and-addition module is equipped with 5 single-bit control signals $s_1 - s_5$. Firstly the input data are fed into the common subexpression block. The signal s_1 indicates which constant channels will be chosen for processing the input data. Each channel carries out shift, negation and addition for the constant. The control signal s_3 indicates that the constant 7641 block outputs the product either by 7641 or 7642. Similarly, the signals s_2 and s_4 control the outputs of constant 5a82 and 30fb blocks respectively. The invert units following the constant units either invert the outputs of the constant units or pass them without any change. The swap unit provides the appropriate swapping for input data, depending on whether the coefficient is (30fb,7641) or (7641,30fb). The demultiplex unit judges which couple of products are final outputs. Totally, 11 adders are used to compose the shift-and-addition module.

Based on the above discussion, the complex multiplication unit in 16-point radix-4 pipelined FFT can be substituted by a multiplier-less unit. The block diagram of the unit is depicted in Figure 4. Only those data, which multiply nontrivial complex coefficients, are fed into the shift-and-addition units. Two shift-and-addition units are needed for the real part (X_r) and imaginary part (X_i) respectively. There are two single-bit control signals s_6 and s_7 in the multiplier-less unit. The signal s_6 indicates that whether the input data is corresponding to a nontrivial complex coefficient or not. When the signal s_7 is asserted to logic 1 state, the real and imaginary parts of the input data are swapped, and

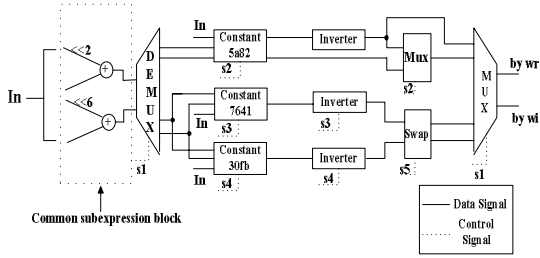


Fig.3. Block diagram of the shift-and-addition module in multiplier-less unit of 16-point R4SDC FFT

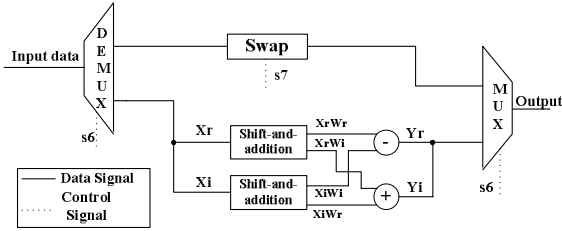


Fig.4. Block diagram of the multiplier-less unit in 16-point R4SDC FFT

the imaginary part is inverted. Here, in the multiplier-less unit, 22 adders are used to substitute the four real multipliers in the complex multiplier unit. Due to the use of multiplier-less unit, the Rom unit storing the coefficient will be replaced by a FSM unit generating control signals ($s1 - s7$).

3.2. IDR commutator

The commutator unit is one of the main power-consuming components in R4SDC FFT. Previous approaches to implement commutators include shift register architecture (SR) [7], conventional dual port RAM architecture (DR) [12], and triple port RAM architecture (TR) [12]. These architectures are based on the same interconnection topology among different FIFO elements [12]. In [8], we proposed a new architecture based on dual port RAMs, termed as IDR. IDR exploits a new interconnection topology among dual port RAM blocks. The block diagram of IDR is depicted in Figure 5. IDR efficiently reduces the switching activity through maintaining the unused outputs of RAMs at their previous values [8]. IDR also reduces the number of write operations to memory blocks. Table2 illustrates which RAMs are enabled for write operation during each period.

Table2: RAMs selected in different periods

m_t	0	1	2	3
RAMs selected	DM1 DM3	DM0, DM2, DM4	DM1, DM3, DM5	DM0 DM2

For stage t , when m_t is equal to 1, new N_{t-1} input data is processed. The first N_t data will be written into DM0.

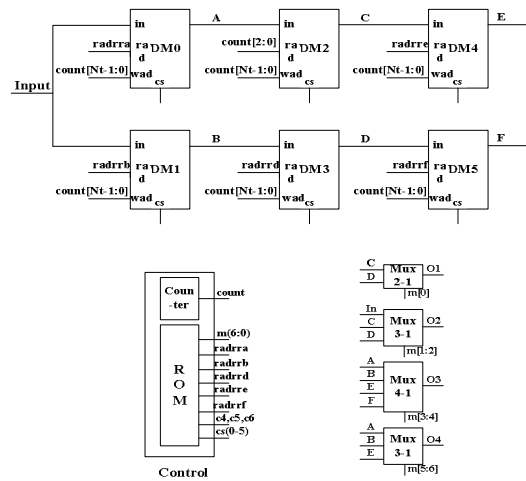


Fig.5. Proposed commutator architecture based on dual port RAMs for R4SDC

The previous N_t values stored in DM0 will be read out and written into DM2 for vacating space for the new ones. The same applies for DM2 and DM4. The other three RAMs will be disabled for write operation during this period. For $m_t = 0$ and 3, the number of RAMs enabled is two, because the previous values stored in DM2 and DM3 are no longer needed for subsequent outputs. Therefore, with IDR architecture each RAM block is enabled $5/3$ times on average, during the four periods. Whereas, for DR and TR, each RAM block is enabled 4 and $10/3$ times respectively [12]. Hence, our architecture is significantly more power efficient than both DR and TR.

3.3. Low Power Butterfly

In R4SDC FFT, the butterfly element performs the summations of Equations 4 and 5. The conventional butterfly architecture consists of 6 adder/subtractors. In [9], a low power butterfly architecture was presented. Two 4-input summation blocks were employed to replace six adder/subtractors. However, since inversions were implemented based on one's complementing (and not two's complementing), this architecture introduced a small error in the butterfly operations. In this paper, we improve this architecture by eliminating this error. Figure 6 shows the improved low power butterfly architecture. Six inverters (CI_1 to CI_6) are used to generate the normal or the one's complement form under the control of C_5 , C_6 and C_7 . The signal C_4 controls the four multiplexers (M_1 to M_4) for directing appropriate data to the inputs of the summation blocks. Two 5-input summation blocks (SUM_0 and SUM_1) are employed to generate the real and imaginary parts of the output respectively. An additional decoder unit is used to generate compensation for

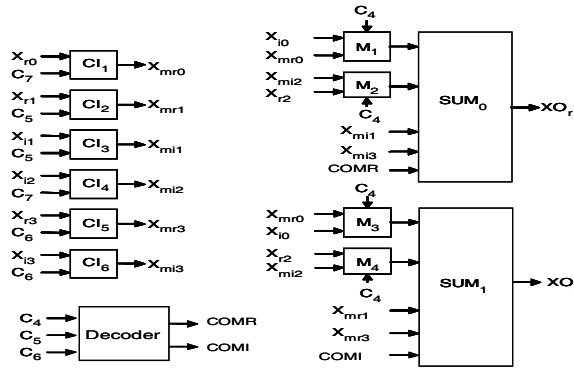


Fig.6. Block diagram of the improved low power butterfly architecture

eliminating the error due to the one's complement based inversion.

4. SIMULATION RESULTS

A total of 4 different architectures, namely conventional FFT, Scheme I, Scheme II and Scheme III, have been implemented in Verilog HDL for 64-point and 16-point R4SDC pipelined FFTs respectively. Input data used are 32 bits (one word) complex data. The 64-point and 16-point R4SDC FFTs are synthesized at 16ns and 12ns clock cycles respectively, using Synopsys DesignCompiler targeting the UMC 0.18 μ CMOS technology library. Power evaluations were carried out, using Synopsys DesignPower, at 20ns and 14ns clock cycles for 64-point and 16-point FFTs respectively. Tables 3 and 4 provide information about main modules for each implementation. For example, in conventional 16-point FFT design, the butterfly modules are based on conventional adder/subtractor architecture (add-sub), dual port RAMs (DR) and shift registers (SR) are used for Commutator1 and Commutator2 respectively. The multipliers are based on non-Booth coded Wallace tree (nbw). In Scheme I, add-sub were replaced with low power butterfly architecture (sum), as described in section 3.3. Scheme II employs an IDR based architecture for Commutator1, as well as using sum architecture for the butterfly modules. Scheme III is a modified version of Scheme II where a multiplier-less approach was employed in the multiplications, as discussed in section 3.1.

The comparative power and area results are shown in Figures 7 and 8 respectively. Clearly, for 16-point FFT the best power saving of 35% is achieved with Scheme III, followed by 23% and 15% savings with Scheme II and Scheme I respectively. Although power savings for 64-point FFT are less than for 16-point FFT, the power profile remains the same, achieving 19%, 12% and 6%

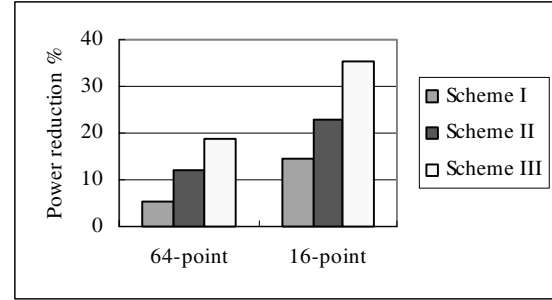


Fig.7. Power reduction of Scheme I – III relative to Conventional FFT

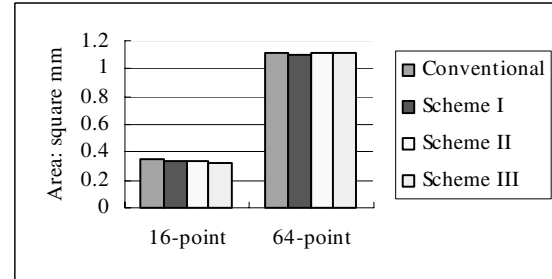


Fig.8. Area of Conventional FFT, and Scheme I – III

power reduction with Scheme III, Scheme II and Scheme I respectively. Table 5 and 6 provide power consumption of the main modules for each implementation.

Table3: Implementation schemes for 16-point FFT

Main modules	Conventional	I	II	III
Butterfly1	add-sub	sum	sum	sum
Butterfly2	add-sub	sum	sum	sum
Commutator1	DR	DR	IDR	IDR
Commutator2	SR	SR	SR	SR
Multiplier	nbw	nbw	nbw	mless

Table4: Implementation schemes for 64-point FFT

Main modules	Conventional	I	II	III
Butterfly1	add-sub	sum	sum	sum
Butterfly2	add-sub	sum	sum	sum
Butterfly3	add-sub	sum	sum	sum
Commutator1	DR	DR	IDR	IDR
Commutator2	DR	DR	IDR	IDR
Commutator3	SR	SR	SR	SR
Multiplier1	nbw	nbw	nbw	nbw
Multiplier2	nbw	nbw	nbw	mless

As can be seen, for Scheme I the low power butterfly architecture consumes only about half of the power of the conventional add-sub based butterfly architecture.

However, for 64-point FFT, stage1 and stage2 butterflies in Scheme II and Scheme III consume more power compared to the conventional butterfly. This is due to the increased switching activity caused by IDR architectures employed in Commutator1 and Commutator2 for these two schemes. However, the effect of IDR is less for 16-point FFT, as can be seen in Table5.

The multiplier-less units in stage1 of 16-point FFT and stage2 of 64-point FFT achieve about half power reduction, as compared to the complex multiplier based on non-Booth coded Wallace tree. IDR architecture performs better in stage2 (32% power saving) than stage1 (15% power saving) of 64-point FFT, as compared to DR architecture. We also compared the area for each design as shown in Figure 8. All designs have the commensurate area with each other. The application of the new techniques brings a slight reduction in the cost of area.

Table5: Power consumption analysis for 16-point R4SDC FFT

Unit: mw	Conventional	I	II	III
Butterfly1	9.530	4.621	4.701	5.132
Butterfly2	8.446	4.543	4.543	4.552
Commutator1	18.704	18.347	12.708	13.004
Commutator2	6.525	6.541	6.559	6.571
Multiplier	15.27	15.27	15.27	7.923
Total	63.243	54.072	48.626	40.908

Table6: Power consumption analysis for 64-point R4SDC FFT

Unit: mw	Conventional	I	II	III
Butterfly1	2.800	1.545	3.049	3.053
Butterfly2	2.821	1.580	3.460	3.461
Butterfly3	5.311	3.110	2.934	3.461
Commutator1	27.109	27.130	23.233	23.206
Commutator2	13.223	13.236	8.950	8.951
Commutator3	3.867	3.841	3.755	3.713
Multiplier1	9.450	9.450	9.450	9.450
Multiplier2	9.866	9.866	9.866	4.985
Total	84.808	80.114	74.643	68.948

5. CONCLUSION

This paper presents a novel multiplier-less pipelined FFT processor architecture suitable for shorter FFTs. This design approach can also be applied for the last stages of longer FFTs. The multiplier-less architecture employs the minimum number of shift and addition operations to realize the complex multiplications. This reduces the power consumption of the multiplier by half. When the architecture is utilized with low power butterfly and commutator scheme developed by the authors, it achieves 19% and 35% power savings for 64-point and 16-point

R4SDC FFTs respectively, as compared to conventional non-Booth coded Wallace tree multiplier based R4SDC FFT architecture.

11. REFERENCES

- [1] S. He and M. Torkelson, "Design and implementation of 1024-point pipeline FFT processor" *Custom Integrated Circuits Conference, 1998. Processing of the IEEE 1998*, 11-14 May 1998. pp. 131 – 134
- [2] S.R. Sridhara and N.R. Shanbhag "Low-power FFT via reduced precision redundancy" *Signal Processing Systems, 2001 IEEE Workshop* pp. 117-124 Sep. 2001
- [3] M.B. Bevan, "A Low-Power, High-Performance, 1024-point FFT Processor", *IEEE Journal of Solid-State Circuit*, Vol. 34, no. 3, pp. 308-387, March 1999
- [4] L. Jia, Y. Gao, J. Isoaho and H. Tenhunen, "A new VLSI oriented FFT algorithm and implementation", in *Proceedings of Eleventh annual IEEE International ASIC conference*, pp. 337-341, 1998
- [5] M. Hasan, T. Arslan, and J.S. Thompson, "A novel coefficient ordering based low power pipelined radix-4 FFT processor for wireless LAN application", *IEEE Transaction on Consumer Electronics*, Vol. 49, no.1, pp. 128-134, FEB. 2003.
- [6] K. Maharatna, E. Grass and U. Jagdhold "A 64-Point Fourier Transform Chip for High-Speed Wireless LAN Application Using OFDM" *IEEE Journal of Solid-State Circuit*. Vol. 39, No. 3, March 2004
- [7] G. Bi and E. V. Jones, "A pipelined FFT processor for word-sequential data", *IEEE Transactions on acoustic, speech, and signal processing*, Vol. 37, no: 12, Dec. 1989, pp.1982 - 1985
- [8] W Han, T. Arslan, A.T. Erdoga, and M. Hasan "A Low power Dual Port RAM Based Commutator for A Pipelined FFT Processor" submitted to *18th International Conference on VLSI Design*
- [9] M. Hasan *Low Power Techniques and Architectures for Multicarrier Wireless Receivers* PhD thesis, University of Edinburgh, 2003
- [10] T.S. Chang, J.I. Guo and C.W. Jen, "Hardware-Efficient DFT Designs with Cyclic Convolution and Subexpression Sharing" *IEEE Transaction on Circuit and Systems* Vol. 47, No. 9, Sep. 2000
- [11] Kai Hwang *Computer Arithmetic: principles, architecture, and design* John Wiley & Sons, Inc. 1979 pp. 149-151
- [12] Mohd. Hasan and Tughrul Arslan, "A triple port RAM based low power commutator architecture for a pipelined FFT processor" *Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium*, Vol. 5, on 25-28 May 2003, pp. V-353 - V-356