High-Performance Low-Power FFT Cores

Wei Han, Ahmet T. Erdogan, Tughrul Arslan, and Mohd. Hasan

Recently, the power consumption of integrated circuits has been attracting increasing attention. Many techniques have been studied to improve the power efficiency of digital signal processing units such as fast Fourier transform (FFT) processors, which are popularly employed in both traditional research fields, such as satellite communications, and thriving consumer electronics, such as wireless communications. This paper presents solutions based on parallel architectures for high throughput and power efficient FFT cores. Different combinations of hybrid low-power techniques are exploited to reduce power consumption, such as multiplierless units which replace the complex multipliers in FFTs, low-power commutators based on an advanced interconnection, and parallel-pipelined architectures. A number of FFT cores are implemented and evaluated for their power/area performance. The results show that up to 38% and 55% power savings can be achieved by the proposed pipelined FFTs and parallel-pipelined FFTs respectively, compared to the conventional pipelined FFT processor architectures.

Keywords: Low power, FFT, multiplierless architecture, parallel FFT architecture, pipelined FFT architecture, energy efficient.

Manuscript received June 25, 2007; revised Feb. 16, 2008.

Wei Han (phone: + 44 (0) 131 6505592, email: w.han@ed.ac.uk), Ahmet T. Erdogan (email: Ahmet.Erdogan@ee.ed.ac.uk), and Tughrul Arslan (email: T.Arslan@ed.ac.uk) are with the School of Engineering and Electronics, University of Edinburgh, Edinburgh, UK.

Mohd. Hasan (phone: + 0091 571 2721148, email: m_hasan786@rediffmail.com) is with the Department of Electronics Engineering, AMU, Aligarh, India

I. Introduction

The FFT processor is widely used in mobile systems for image and signal processing applications. It is a main module of OFDM-based systems, such as the MC-CDMA receiver and WLAN chips. The requirement for low-power FFT architectures for telecommunication systems in portable form is becoming more and more important. Due to the characteristic of non-stop processing at sample rate, the pipelined FFT is the leading architecture for high throughput or low-power solutions [1]. In pipelined architectures, power consumption is dominated by the commutator and the complex multiplier at each stage.

Researchers have proposed a number of low-power techniques for FFT processors. In [2], a cache-memory-based architecture was presented, which uses an algorithm that offers good data locality to increase speed and energy efficiency. In [3], the authors proposed a new radix-2/4/8 algorithm, which can effectively minimize the number of complex multiplications in pipelined FFTs. An ordering-based pipelined radix-4 FFT was presented in [4]. Coefficient ordering reduces the switching activity between successive coefficients fed to the complex multiplier, which leads to lower power consumption. The power of a pipelined FFT processor is dominated by the size of storage blocks. Therefore, the authors of [5] and [6] proposed progressive word length instead of fixed word length, using a shorter word length for stages in which the word length's impact on size is significant and a longer word length for stages in which the word length's impact on precision is significant. In [7], the authors proposed a low-power FFT architecture based on multirate signal processing and asynchronous circuit technology. The communication is localized, and the sharing of the global memory is eliminated. To reduce the number of operations in FFTs and thus reduce power consumption, some researchers

use shifters and adders to replace the complex multiplications by some special constant coefficients. The authors of [8] employ seven shift-and-add units to carry out seven multiplications in parallel, each by a constant coefficient. In [9] and [10], we proposed a multiplierless architecture based on common subexpression sharing, which replaces the complex multipliers in FFTs, and a low-power commutator architecture, which reduces the number of memory accesses. In addition to the pipelined FFT, the parallel-pipelined FFT is a good solution for applications requiring high throughput and high power efficiency. This paper implements a number of FFT IP cores based on both pipelined and parallel-pipelined architectures, through different combinations of hybrid low-power architectures.

II. Pipelined FFTs and Parallel-Pipelined FFTs

The discrete Fourier transform (DFT) of N complex data points, $x(n)$, is defined by

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

where W_N is the twiddle factor, $W_N = e^{-j(2\pi/N)}$. The FFT is a fast algorithm for computing DFT. In [11], the authors presented a radix-4 single-path delay commutator (R4SDC) pipelined FFT algorithm for word-sequential data. For radix r_1, (1) can be rewritten as

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

The N-point DFT in (2) can be decomposed into v stages, where $N = r_1 r_2 \cdots r_v$, and the final stage is defined as

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

The intermediate stages (t) are given as in [11] by the following recursive equation:

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

where, for both (3) and (4), the following conditions apply:

$$\begin{aligned} &0 \le q_i \le N_i - 1, \quad 2 \le i \le v, \quad \text{and} \quad N_t = N/(r_1 r_2 \cdots r_v) \\ &2 \le t \le v-1, \quad 0 \le m_i \le r_i - 1. \end{aligned} \tag{5}$$

Based on the preceding equations, the flow graph of a 16-point FFT with $r_1 = 4$ can be seen in Fig. 1, where 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). An N-point pipelined FFT processor based on the R4SDC architecture is shown in Fig. 2. It achieves 75% utilization of the complex multiplier and 100% utilization of the butterfly element [11], [12]. The parallel-pipelined architectures for 16-point and 64-point FFTs are shown in Figs. 3 and 4, respectively.

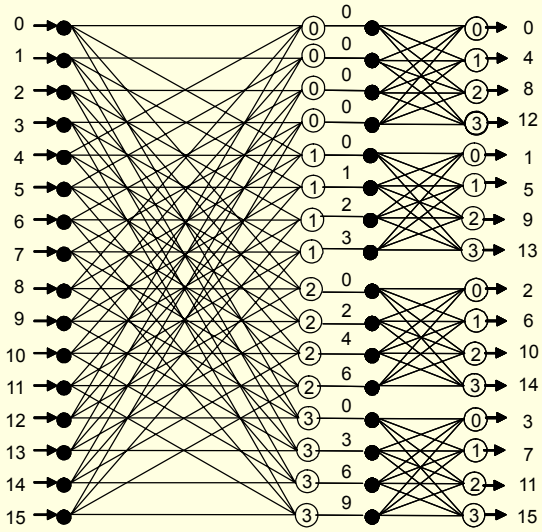


Fig. 1. Signal flow graph of a radix-4 16-point FFT.

III. Low-Power Techniques

In this paper, three low-power techniques are employed to reduce the power consumption in different FFT architectures. The first technique involves the use of a parallel-pipelined architecture at a lower frequency to meet the given throughput. The second technique replaces the complex multiplier with a minimum number of adders and shifters by using both two's complement and canonical signed-digit (CSD) representations. The third technique proposes a low-power architecture of the commutator (IDR) with a minimal number of memory write operations. These techniques are described in more detail in the following sub-sections.

1. Parallel-Pipelined Architectures

The pipelined FFT is viewed as the leading architecture for real time applications. However, the use of only one processor element (PE) in each stage limits the throughput of pipelined FFTs. Therefore, an increased throughput requires further parallelization.

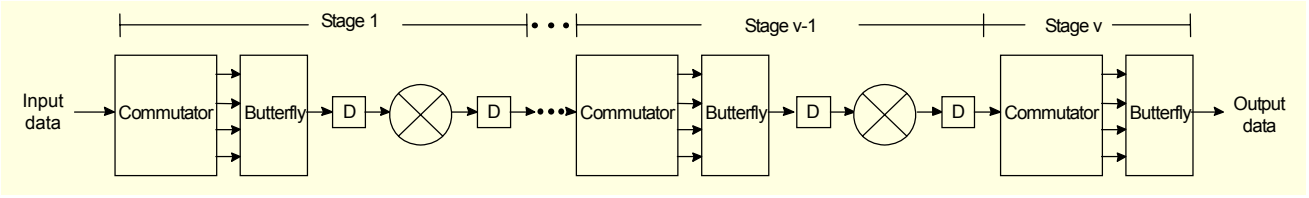


Fig. 2. N-point R4SDC pipelined FFT processor architecture.

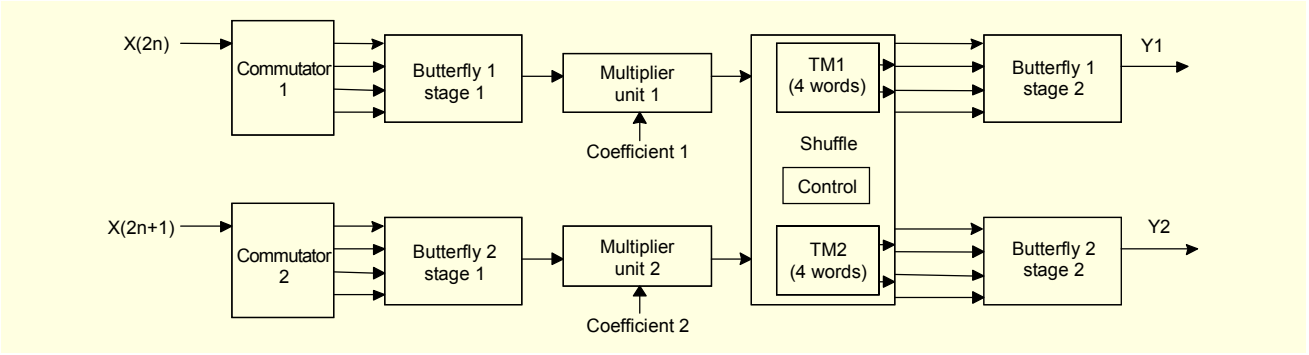


Fig. 3. Block diagram of 16-point 2-parallel-pipelined FFT architecture.

This means more PEs need to be used to complete the computations required in each stage of the FFT. Parallel-pipelined FFTs provide a suitable solution for both high throughput and high power efficiency applications. In parallel-pipelined architectures, only the number of PEs is increased, and the sizes of FIFOs remain the same. Hence, the area overhead of the parallel-pipelined architectures is not significant. For a given throughput, parallel-pipelined FFTs can operate at lower frequencies than the pipelined FFTs, resulting in lower power consumption. In a 16-point 2-parallel-pipelined FFT, 2 PEs are allocated to each stage of the FFT, doubling the throughput. This architecture is shown in Fig. 3, where the input data is separated into two streams, namely $X(2n)$ and $X(2n + 1)$. Two commutators in stage 1, each having half of the storage units of the original commutator, provide the data to the butterfly units. The coefficients are divided into even (coefficient 1) and odd (coefficient 2) sections according to even and odd input data streams. These coefficients are then fed into the corresponding complex multipliers. Due to the separate processing for odd and even data, a shuffle unit is needed in stage 2 to implement the interstage data shuffle. The shuffle unit is composed of two triple port SRAMs (TM1 and TM2) and an addressing control unit. The four outputs from the two triple port SRAMs are then fed into the simplified butterfly units in stage 2.

Similarly, in a 64-point 4-parallel-pipelined FFT, four PEs are allocated to each stage of the FFT, resulting in a 400% increase of the throughput rate. The input data is separated into four streams, namely, $X(4n)$, $X(4n + 1)$, $X(4n + 2)$, and $X(4n + 3)$. There are four commutators in both stage 1 and stage 2. Each of the commutators has 1/4 size of the commutators in corresponding stages of the R4SDC pipelined FFT. The coefficients in each stage are divided into four sections. Coefficients 1 to 4 correspond to the four input streams. Only 3 complex multipliers are used in stage 2, since the coefficient set for the first butterfly consists of only (7fff, 0000), which can easily be implemented without a multiplier. In this architecture, since the number of PEs per stage is equal to four, no shuffle unit is needed. The outputs of the multiplier units in stage 2 are fed into each of the four simplified butterfly elements in stage 3, as shown in Fig. 4.

2. Multiplierless Architecture

In FFTs, the conventional complex multiplier consists of four real multipliers, one adder and one subtractor. However, since the complex coefficients for all stages can be pre-computed, we can use shift and add operations with common subexpression sharing for those stages that have few coefficients. In [9], we proposed a multiplierless architecture to substitute the complex multiplier. For example, the number of coefficients used in the second stage of a 64-point FFT or the first stage of a 16-point FFT is 16. These coefficients are shown in Table 1. A close observation of these coefficients reveals that seven of them are (7fff, 0000), and one of them is (0000, 8000). These are the quantized representations of (1, 0) and (0, -1) in a 16-bit fractional two's complement format, respectively. In each set, the first entry corresponds to the cosine function (the real part,

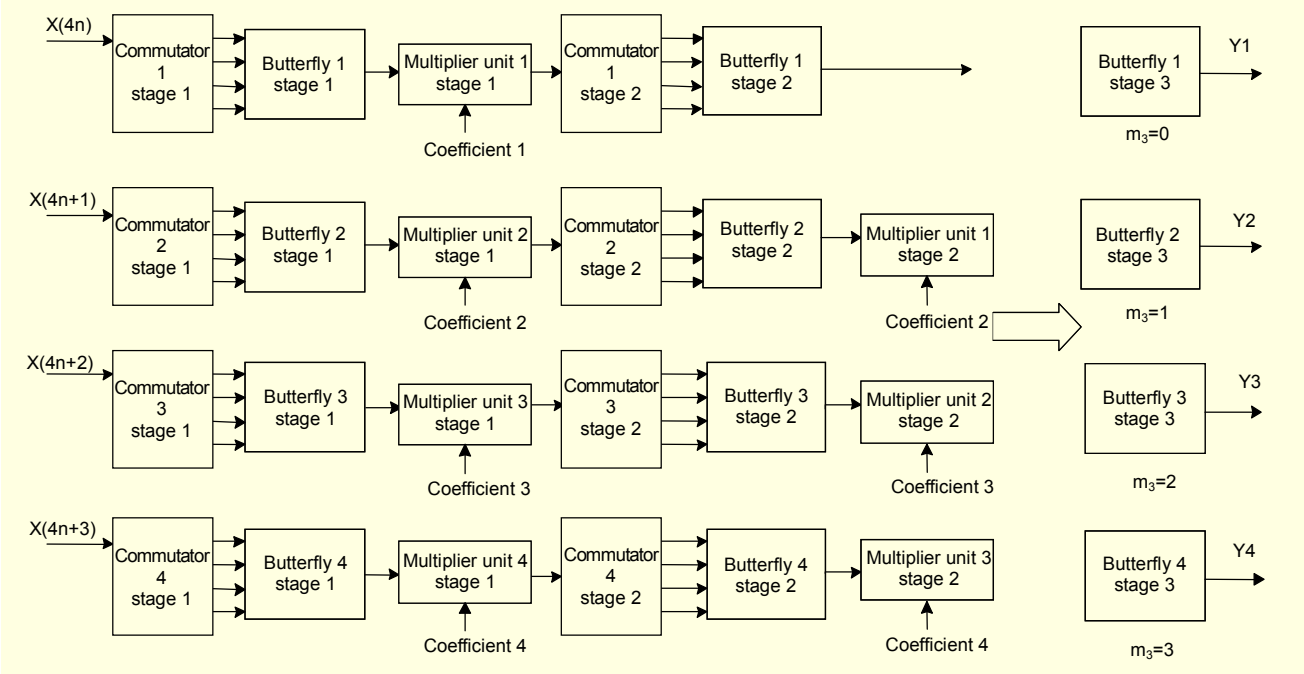


Fig. 4. Block diagram of 64-point 4-parallel-pipelined FFT architecture.

Table 1. The coefficients for 16-point R4SDC FFT.

Coefficient sequence m_1=0,1	Original quantized coefficient	Coefficient sequence m_1=2,3	Original quantized coefficient
W0	7fff, 0000	W0	7fff, 0000
W0	7fff, 0000	W2	5a82, a57d
W0	7fff, 0000	W4	0000, 8000
W0	7fff, 0000	W6	a57d, a57d
W0	7fff, 0000	W0	7fff, 0000
W1	7641, cf04	W3	30fb, 89be
W2	5a82, a57d	W6	a57d, a57d
W3	30fb, 89be	W9	89be, 30fb

W_r) and the second entry corresponds to the sine function (the imaginary part, W_i). For trivial coefficients, such as (7fff, 0000) and (0000, 8000), complex multiplication is unnecessary. For example, the data can directly pass through the multiplier unit without any multiplication, when it is multiplied with the coefficient set (7fff, 0000). Similarly, only an additional unit, which swaps the real and imaginary parts of the input data and inverts the imaginary part, is needed for that data which is multiplied by (0000, 8000). The rest of the coefficients are nontrivial coefficients. They are composed of only 6 constants (7641, 5a82, 30fb, a57d, 89be, and cf04). However, 89be, a57d, and cf04 are the one's complements of 7641, 5a82, and 30fb, respectively. Hence, only multiplications with these constants (7641, 5a82, and 30fb) would be required to implement all multiplications with these nontrivial 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 is already available. 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. The constant 5a82 is represented by two's complement format, and 7641 and 30fb are represented by CSD format as follows: 5a82 (0101101010000010), 7641 (1000-10-1001000001), and 30fb (010-1000100000-10-1). The mixed use of CSD and two's complement minimizes the number of addition/shift operations. We can use the shift-add based implementation of multiplications with the three constants to carry out those nontrivial complex multiplications. According to the previous representation, these multiplications with the three constants are given by:

$$\begin{aligned} 5a82X &= X << 1 + X << 7 + X << 9 + X << 11 + X << 12 + X << 14, \\ 7641X &= X + X << 6 - X << 9 - X << 11 + X << 15, \\ 30fbX &= -X - X << 2 + X << 8 - X << 12 + X << 14, \end{aligned} \quad (6)$$

where X represents the input data. In the previous equations,

Table 2. Operations required before common subexpression sharing.

Operation	5a82X	7641X	30fbX
Addition	5	2	2
Subtraction	0	2	3
Shift	6	4	4

Table 3. Operations required after common subexpression sharing.

Operation	5a82X	7641X	30fbX
Addition	2	1	0
Subtraction	0	1	2
Shift	3	3	2

the number of operations required for the computation of 5a82X, 7641X, and 30fbX are shown in Table 2. In all nontrivial coefficient multiplications, the proportions of the multiplications referring to 5a82, 7641, and 30fb are 50%, 25%, and 25%, respectively. Hence, the average operations for a nontrivial coefficient multiplication are 4.5 additions, 2.5 subtractions, and 7 shifts.

However, if $5X = X + X << 2$ and $65X = X + X << 6$ are precomputed, (6) can be rewritten as

$$\begin{aligned} 5a82X &= 5X << 12 + 5X << 9 + 65X << 1, \\ 7641X &= X << 15 + 65X << - 5X << 9, \\ 30fbX &= 65X << 8 - X << 12 - 5X. \end{aligned} \quad (7)$$

The common subexpressions for the three constants are 101 and 1000001. Pre-computation requires 2 additions and 2 shifts. In (7), the operations required for the computations of 5a82X, 7641X, and 30fbX are shown in Table 3. The results of the pre-computation can be used for both multiplications with the real part (W_r) and the imaginary part (W_i) of nontrivial coefficients. Therefore, after the common subexpression sharing, the average operations for a nontrivial coefficient multiplication are 3.5 additions, 1.5 subtractions, and 6 shifts, including the operations in pre-computation. Therefore, the operation saving of additions/subtractions is 28.8%, and that of shift is 14.3%.

Figure 5 shows the shift-and-add module for the three constants in a 16-point FFT. The module carries out the multiplications in which the real part (X_r) or the imaginary part (X_i) of input data is multiplied by W_r and W_i, respectively. The shift-and-add module is equipped with five single-bit control signals, s_1–s_5. First, the input data is 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 operations for a given 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 controllable invert units following the constant units either invert the outputs of the constant units or pass them unchanged. The swap unit provides the appropriate swapping for input data, depending on whether the coefficient is (30fb, 7641) or (7641, 30fb). The output switch unit selects the final outputs. Only 11 adders are used in the shift-and-add module.

Based on the above discussion, the complex multiplication unit in a 16-point radix-4 pipelined FFT can be substituted by a multiplierless unit. The block diagram of the unit is depicted in Fig. 6. Only data that has to be multiplied with nontrivial complex coefficients is fed into the shift-and-add units. Two shift-and-add units are needed for both the real part (X_r) and the imaginary part (X_i). There are two single-bit control signals, s_6 and s_7, in the multiplierless unit. Signal s_6 indicates whether the input data corresponds to a nontrivial complex coefficient. When signal s_7 is asserted to logic 1 state, the real and imaginary parts of the input data are swapped, and the imaginary part is inverted. Otherwise, the swap unit passes the

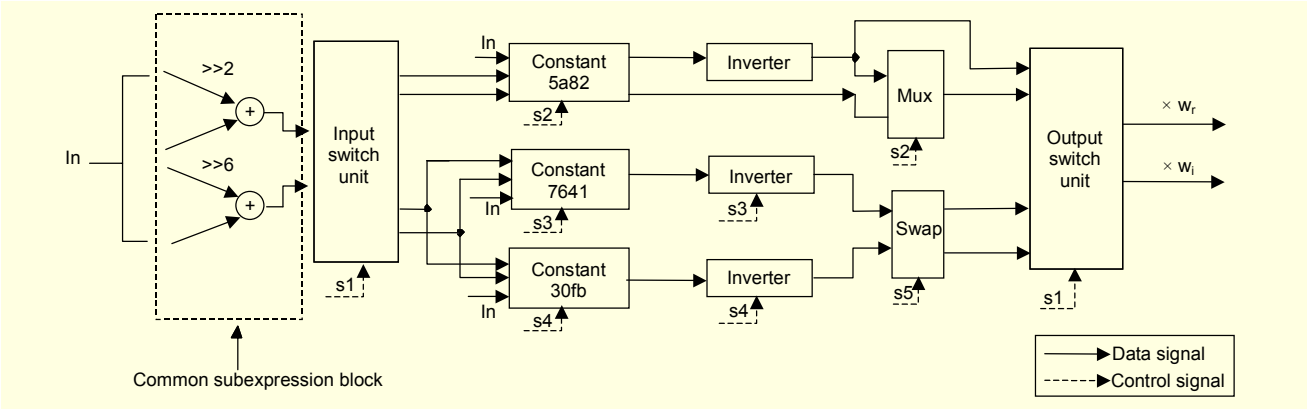


Fig. 5. Block diagram of the shift-and-add module in multiplierless unit of the 16-point R4SDC FFT.

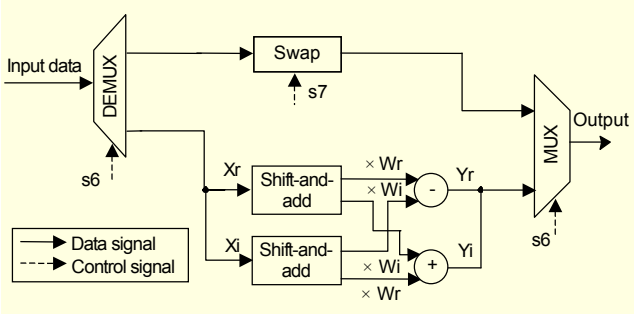


Fig. 6. Block diagram of the multiplierless unit in the 16-point R4SDC FFT.

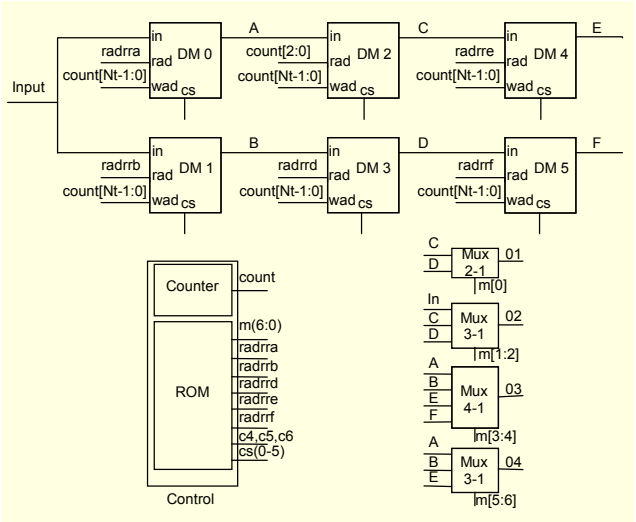


Fig. 7. IDR commutator architecture based on dual port RAMs for R4SDC.

input data unchanged. Here, in the multiplierless unit, 22 adders are used to substitute the four real multipliers in the complex multiplier unit. Due to the use of the multiplierless unit, the ROM unit storing the coefficients is replaced by an FSM unit for generating the control signals (s_1–s_7). The multiplierless approach can also be used for 16-point 2-parallel-pipelined FFTs and 64-point 4-parallel-pipelined FFTs. Similar multiplierless architectures such as the one shown in Fig. 6 are employed in place of complex multipliers in those FFTs.

3. IDR Commutator

The commutator unit is one of the main power-consuming components in the R4SDC FFT. Previous approaches to implementing commutators include shift register (SR) architecture [11], conventional dual-port RAM (DR) architecture [11], and triple-port RAM (TR) architecture [13]. These architectures are based on the same interconnection topology among different FIFO elements [13]. In [10], we proposed a new architecture based on dual-port RAMs, termed as IDR. IDR exploits a new interconnection topology among dual-port RAM blocks. Figure 7 shows a block diagram of IDR. IDR efficiently reduces the switching activity by maintaining the unused outputs of RAMs at their previous values. It also reduces the number of write operations to memory blocks. Table 4 shows which RAMs are enabled for write operation during each period. For example, for stage t, when m_t is equal to 1, new N_t-1 input data is processed. The first N_t data is written into DM0. The previous N_t values stored in DM0 will be read out and written into DM2 to free space for the new ones. The same applies to DM2 and DM4.

Table 4. RAMs selected in different periods.

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

The other three RAMs are 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 an average of 5/3 times during the four periods, whereas for DR and TR, each RAM block is enabled 4 and 10/3 times, respectively [13]. Hence, our architecture is significantly more power efficient than both DR and TR.

4. Low-Power Butterfly

In the R4SDC FFT, the butterfly element performs the summations of (2). The conventional butterfly architecture consists of 6 adders/subtractors. In [14], a low-power butterfly architecture was presented. Two 4-input summation blocks were employed to replace six adder/subtractors. However, since the inversions were implemented based on one's complement (and not two's complement), the architecture introduced a small error in the butterfly operations. In this paper, we improve the architecture by eliminating this error. Figure 8 shows the improved low-power butterfly architecture. Six inverters (Con_1 to Con_6) are used to generate the normal form or the one's complement form under the control of C_5, C_6, and C_7. Signal C_4 controls the four multiplexers (Mux_1 to Mux_4) to direct 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 eliminate the error caused by the one's-complement-based inversion.

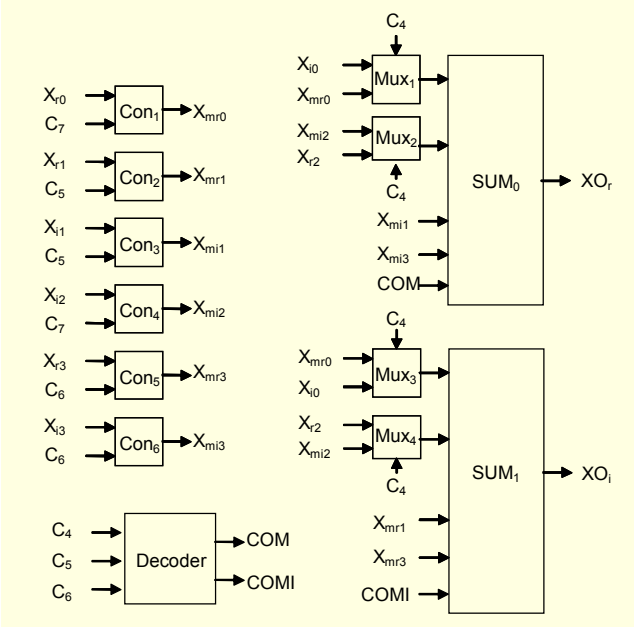


Fig. 8. Block diagram of improved low-power butterfly architecture.

IV. Results

A total of five different architectures (namely, the conventional scheme and schemes I to IV) were implemented for 64-point and 16-point R4SDC pipelined FFTs. The conventional FFT and schemes I to III were implemented for 32-point FFTs. Pipelined architectures were used in the conventional approach and in schemes I to III for three FFTs of different sizes. Parallel-pipelined architectures were employed in scheme IV. The input data used was 32 bits long. The 64-point pipelined FFTs were synthesized at 16 ns clock cycles. The 32-point and 16-point pipelined FFTs were synthesized at 12 ns clock cycles. Power evaluations were carried out, at 20 ns for the 64-point FFTs and at 14 ns for the 32-point and 16-point FFTs. For the parallel pipelined FFTs, the 16-point 2-parallel-pipelined FFT was synthesized at 24 ns clock cycles and power evaluated at 28 ns clock cycles. The 64-point 4-parallel-pipelined FFT was synthesized at 60 ns clock cycles and power evaluated at 80 ns clock cycles. All designs were synthesized by the Synopsys DesignCompiler targeting the UMC 0.18 μ CMOS technology library, and power was evaluated by Synopsys DesignPower. Tables 5 to 7 provide information about the main modules for each implementation. According to the R4SDC pipelined architecture, for 32-point and 64-point FFTs, there are three stages, each containing one butterfly module and one commutator module. Multiplier 1 and multipiler 2 represent the multiplier units used after commutator 1 and commutator 2, respectively. There are only two stages and one multiplier unit for 16-point FFTs. For all

Table 5. Implementation schemes for 16-point FFT.

Main modules	Conv.	I	II	III	IV
Butterfly 1	add-sub	sum	sum	sum	2X sum
Butterfly 2	add-sub	sum	sum	sum	2X simplified
Commutator 1	DR	DR	IDR	IDR	SR
Commutator 2	SR	SR	SR	SR	shuffle (TM)
Multiplier	NBW	NBW	NBW	mless	2X mless

Table 6. Implementation schemes for the 32-point FFT.

Main modules	Conv.	I	II	III
Butterfly 1	add-sub	sum	sum	sum
Butterfly 2	add-sub	sum	sum	sum
Butterfly 3	add-sub	add-sub	add-sub	add-sub
Commutator 1	DR	DR	IDR	IDR
Commutator 2	DR	SR	SR	SR
Commutator 3	SR	SR	SR	SR
Multiplier 1	NBW	NBW	NBW	NBW
Multiplier 2	NBW	NBW	NBW	mless

Table 7. Implementation schemes for the 64-point FFT.

Main modules	Conv.	I	II	III	IV
Butterfly 1	add-sub	sum	sum	sum	4X sum
Butterfly 2	add-sub	sum	sum	sum	4X sum
Butterfly 3	add-sub	sum	sum	sum	4X simplified
Commutator 1	DR	DR	IDR	IDR	IDR
Commutator 2	DR	DR	IDR	IDR	SR
Commutator 3	SR	SR	SR	SR	NA
Multiplier 1	NBW	NBW	NBW	NBW	mless, 3X NBW
Multiplier 2	NBW	NBW	NBW	mless	3X mless

different size FFTs, in the conventional approach, the butterfly modules in all stages are based on the conventional adder/subtractor architecture (add-sub). The commutator in the final stage is based on an SR, whereas the commutators in other stages are based on DRs. The multipliers are based on a non-Booth coded Wallace tree (NBW) architecture. The only difference between the conventional approach and scheme I is that the add-subs were replaced with a low-power butterfly architecture (sum) as described in section III.4, except the third stage of the 32-point FFT, where radix-2 butterflies were used. Scheme II differs from scheme I, as the IDR architecture was employed in stage 1 of all FFTs and stage 2 of the 64-point FFT. Scheme III is a modified version of scheme II, where a

Table 8. Power consumption analysis for 16-point R4SDC FFT.

	Power consumption (mW)				
	Conv.	I	II	III	IV
Butterfly 1	9.53	4.62	4.70	4.52	3.70
Butterfly 2	8.45	4.54	4.54	4.55	2.99
Commutator 1	18.70	18.3	12.71	12.48	10.13
Commutator 2	6.53	6.54	6.56	6.55	2.83
Multiplier	15.27	15.27	15.27	8.06	8.55
Total	63.24	54.07	48.63	39.41	30.20

Table 9. Power consumption analysis for 32-point R4SDC FFT.

	Power consumption (mW)			
	Conv.	I	II	III
Butterfly 1	10.66	5.61	4.87	4.62
Butterfly 2	6.423	3.54	3.54	3.53
Butterfly 3	2.365	2.37	2.37	2.36
Commutator 1	26.57	27.13	18.99	18.26
Commutator 2	10.52	10.55	10.54	10.54
Commutator 3	1.381	1.38	1.38	1.37
Multiplier 1	14.99	14.99	14.99	13.68
Multiplier 2	15.08	15.08	15.08	3.07
Total	98.18	90.81	82.17	65.34

Table 10. Power consumption analysis for 64-point R4SDC FFT.

	Power consumption (mW)				
	Conv.	I	II	III	IV
Butterfly 1	2.80	1.55	3.05	1.45	3.56
Butterfly 2	2.82	1.58	3.46	2.97	3.24
Butterfly 3	5.31	3.11	2.93	2.93	1.11
Commutator 1	27.11	27.13	23.23	22.64	9.65
Commutator 2	13.22	13.24	8.95	8.46	3.43
Commutator 3	3.87	3.84	3.76	3.70	NA
Multiplier 1	9.45	9.45	9.45	8.86	10.07
Multiplier 2	9.87	9.87	9.87	4.98	5.29
Total	84.81	80.11	74.64	64.80	40.59

multiplierless approach (mless) was used for the multiplier unit in the final stage, as discussed in section III.2. For scheme IV of the 16-point FFT, the parallel-pipelined architecture was employed, where two sum-based butterfly modules were used in stage 1 and two simplified butterfly modules were used in stage 2. Commutator 1 was based on SR, commutator 2 utilized a triple-port SRAM-based shuffle, and two multiplierless units

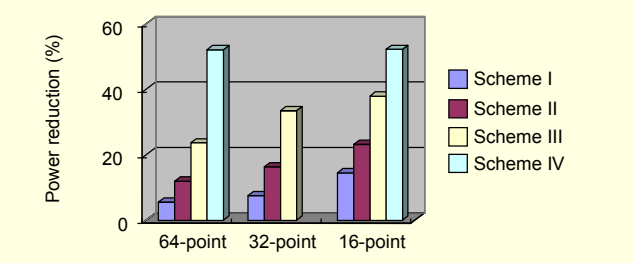


Fig. 9. Power reduction of schemes I to IV relative to conventional FFT.

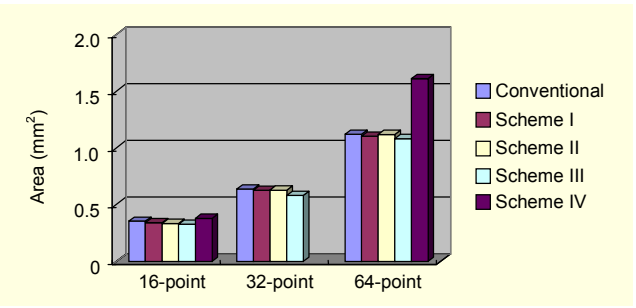


Fig. 10. Area comparison.

were utilized to replace complex multipliers. For scheme IV of the 64-point FFT, IDR and SR were used for commutator 1 and commutator 2. Three multiplierless units were applied in stage 2, while one multiplierless unit and three NBW-based complex multipliers were applied in stage 1.

The comparative power and area results are shown in Figs. 9 and 10, respectively. Clearly, for 16-point FFTs, the best power saving of 52% is achieved with scheme IV, followed by 38%, 23%, and 15% savings with schemes III, II, and I, respectively. The power profiles for 64-point and 32-point FFTs remain the same, achieving 52%, 24%, 12%, and 6% power reduction for 64-point FFTs from scheme IV to scheme I, and 34%, 16%, and 8% for 32-point FFTs from scheme III to scheme I respectively. Tables 8 to 10 show the power consumption of the main modules for each implementation. For most pipeline-based FFTs, the low-power butterfly consumes only about half the power of the conventional add-sub-based butterfly architecture. However, in schemes II and III of the 64-point FFT, sum-based butterflies in some stages consume more power than conventional butterflies. This is due to the increased switching activity caused by IDR architectures, on the input data of butterfly modules. However, IDR has less influence on both 16-point and 32-point FFTs. For all sizes of FFTs, the multiplierless units achieve power savings of about 50% or more than the complex multiplier based on the non-Booth coded Wallace tree. For commutators, IDR architectures perform better in stage 2 (32% power saving) than stage 1 (15% power saving) of 64-point FFTs, as compared to DR architecture. For pipelined FFTs,

the proposed techniques bring an area reduction to some extent. From all of the results, it is obvious that parallel-pipelined architectures can achieve the best power savings with the area penalty to some extent. For 16-point FFTs, the parallel-pipelined architecture only has 7% area overhead. However, for 64-point FFTs, the area penalty of the parallel-pipelined architecture is 47%. Hence, scheme III has the best trade off between power saving and area increase for various sizes of FFTs.

V. Conclusion

In this paper, a number of high-performance FFT cores based on combinations of hybrid low-power techniques were presented. These low-power techniques are parallel-pipelined architectures, the multiplierless architecture, IDR commutator architecture, and the low-power butterfly architecture. The impact of parameterization on power/area performance has been studied. Based on the combination of the proposed low-power techniques, up to 52% power saving is achieved.

References

[1] S. He and M. Torkelson, "Design and Implementation of 1024-Point Pipeline FFT Processor," *Custom Integrated Circuits Conference, Processing of the IEEE,* May 1998, pp. 131-134.

[2] M.B. Bevan, "A Low-Power, High-Performance, 1024-Point FFT Processor," *IEEE Journal of Solid-State Circuit*, vol. 34, no. 3, Mar. 1999, pp. 308-387.

[3] L. Jia, Y. Gao, J. Isoaho, and H. Tenhunen, "A New VLSI-Oriented FFT Algorithm and Implementation," *Proc. of Eleventh Annual IEEE Int'l ASIC Conference*, 1998, pp. 337-341.

[4] 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 Trans. on Consumer Electronics*, vol. 49, no. 1, Feb. 2003, pp. 128-134.

[5] W. Li, Y. Ma, and L. Wanhammar, "Word Length Estimation for Memory Efficient Pipeline FFT/IFFT Processor," *Int'l Conf. on Signal Processing Applications & Technology (ICSPAT)*, Nov. 1999.

[6] S. Johansson, S. He, and P. Nilsson, "Wordlength Optimization of a Pipelined FFT Processor," *42nd Midwest Symp. on Circuits and Systems*, vol. 1, 1999, pp. 501-503.

[7] 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.

[8] 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, Mar. 2004.

[9] W. Han, A.T. Erdogan T. Arslan, and M. Hasan, "A Novel Low Power Pipelined FFT Based on Subexpression Sharing for Wireless LAN Applications," *IEEE Signal Processing Systems Workshop (SIPS)*, Oct. 2004, pp. 83-88.

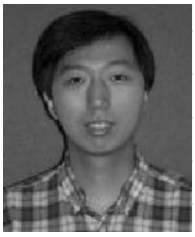
[10] W. Han, A.T. Erdogan, T. Arslan, and M. Hasan, "Low Power Commutator for Pipelined FFT Processors," *IEEE Int'l Symp. on Circuit and Systems (ISCAS*), vol. 5, Kobe, Japan, May 2005, pp. 5274-5277.

[11] G. Bi and E.V. Jones, "A Pipelined FFT Processor for Word-Sequential Data," *IEEE Trans. on Acoustic, Speech, and Signal Processing*, vol. 37, no. 12, Dec. 1989, pp. 1982-1985.

[12] K. Hwang, *Computer Arithmetic: Principles, Architecture, and Design*, John Wiley & Sons, Inc., 1979, pp. 149-151.

[13] M. Hasan and T. Arslan, "A Triple-Port RAM-Based Low Power Commutator Architecture for a Pipelined FFT Processor," *Proc. of the 2003 Int'l Symp. on Circuits and Systems (ISCAS '03),* vol. 5, May 2003, pp. V-353 – V-356.

[14] M. Hasan, *Low Power Techniques and Architectures for Multicarrier Wireless Receivers* PhD thesis, University of Edinburgh, UK, 2003.



Wei Han obtained his BS degree in microelectronics from Nankai University, China. Currently, he is a PhD student with the School of Engineering and Electronics, University of Edinburgh, United Kingdom. He is interested in low-power multi-processor systems.

Ahmet T. Erdogan is a research fellow with the System Level Integration Group in the School of Engineering and Electronics, University of Edinburgh, United Kingdom. He is also involved in teaching and distance learning tutoring activities in VLSI design and IP block authoring at the Institute for System Level Integration, Livingston, UK. He has MSc and PhD degrees in electronic engineering from the University of Cardiff, UK. His research interests include low-power VLSI design, low-power DSPs, and macro IPs.

Tughrul Arslan holds the Chair of Integrated Electronic Systems in the School of Engineering and Electronics of the University of Edinburgh, and is also a co-founder and the Chief Technical Officer of Spiral Gateway Ltd. He is a member of the Integrated Micro and Nano Systems (IMNS) Institute and leads the System Level Integration group (SLIg) in the university. His research interests include low-power system design, integrated micro and nano systems, secure and long life wireless sensor networks, autonomous

systems, system-on-chip (SoC) architectures, evolvable intelligent systems, multi-objective optimisation, and autonomous self-adaptive behaviour in general. He has published more than 300 articles and is the inventor of a number of patents in these areas. He was an associate editor for *IEEE Transactions on Circuits and Systems I* and currently serves as an associate editor for *IEEE Transactions on Circuits and Systems II*. He also sits on the editorial board of *IET Proceedings on Computers and Digital Techniques*. He is a member of the IEEE CAS Committee on VLSI Systems and Applications and is involved in the organisation of numerous conferences. Recently, he was the general chair for the IEEE NASA/ESA conference on Adaptive Hardware and Systems (AHS) and the ECSIS Bio-inspired, Learning, and Intelligent Systems for Security Symposium (BLISS). He has been invited as a keynote speaker to a number of international conferences.

Mohd. Hasan received the BSc Engg degree in electronics engineering in 1990 from AMU, Aligarh, India. Subsequently, he completed his MTech in integrated electronics and circuits from the Indian Institute of Technology, Delhi. He joined the Electronics Engineering Department of AMU, Aligarh, India, as a lecturer in 1992. He became a reader in 1997. He completed his PhD in low-power architectures for signal processing and telecommunications with the School of Engineering and Electronics, University of Edinburgh, UK, in 2004. His research interests include low-power IC design for wireless communications and reconfigurable systems on FPGAs.