MULTIPLIER-LESS BASED PARALLEL-PIPELINED FFT ARCHITECTURES FOR WIRELESS COMMUNICATION APPLICATIONS

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

School of Engineering and Electronics, University of Edinburgh, Edinburgh, UK.

ABSTRACT

This paper proposes two novel parallel-pipelined FFT architectures based on multiplier-less implementation targeting wireless communication applications, such as IEEE 802.11 wireless baseband chip and MC-CDMA receiver. The proposed parallel-pipelined architectures have the advantages of high throughput and high power efficiency. The multiplier-less architecture uses shift and addition operations to realize complex multiplications. By combining a new commutator architecture, and a low power butterfly with this approach, the resulting power and area savings are up to 31% and 20% respectively, for 64-point and 16-point FFTs, as compared to parallel-pipelined FFTs based on Booth coded Wallace tree multipliers.

1. INTRODUCTION

The FFT processor is widely used in DSP and communication applications. It is a critical block in OFDM based communication systems, such as WLAN (IEEE 802.11) and MC-CDMA receiver. Recently, both high data processing and high power efficiency assumes more and more importance in wireless systems. Due to the nature of non-stop processing at the sample rate, the pipelined FFT appears to be the leading architecture for high performance applications. However, only one processor element (PE) in each column makes a bottleneck on the throughput of pipelined FFTs. For example, for N-point radix-4 FFT, the radix-4 PE, consisting of a radix-4 butterfly element and a complex multiplier, has to calculate $N/4$ times to complete all computation in one stage.

For recent wireless systems, such as IEEE 802.11a providing 54Mbps data rate, increased throughput requires further parallelization. It means more than one PE need to be assigned per column to the FFT. Parallel-pipelined FFTs are suitable for both high throughput and high power efficiency. In parallel-pipelined architectures, only hardware cost for PEs will be increased, the actual size of the FIFOs between stages usually remains constant for a given FFT sized N. With the increase of the FFT size, the area of FFT processor will be dominated by the FIFOs. Hence, parallel-pipelined FFTs have not significant area overhead, compared to pipelined FFT [1]. For a given throughput, parallel-pipelined FFTs can operate at lower frequency than pipelined FFTs, therefore resulting in lower power consumption. Not many researchers have explored the scope of parallel-pipelined FFTs. In [2], the perfect shuffle matrix is decomposed into the product of several matrices and a memory grouping, switching can be reduced from M^2 to $2M$, where M is the number of PEs allocated per column. For low power FFTs, a lot of work has been done such as [3, 4, 5, 6]. In [6], we proposed a multiplier-less architecture to replace the conventional complex multiplier in pipelined FFTs. Both area and power consumption for the multiplier unit are reduced. This paper explores a parallel-pipelined architecture with 4 radix-4 PEs in a column for 64-point FFT and a parallel-pipelined architecture with 2 radix-4 PEs in a column for 16-point FFT. The complex multiplication in parallel-pipelined FFTs is replaced by the minimum number of shift and addition operations based on common subexpression sharing across coefficients. Through different combinations of hybrid low power approaches, a number of designs have been successfully implemented, with up to 31% power saving and 20% area saving.

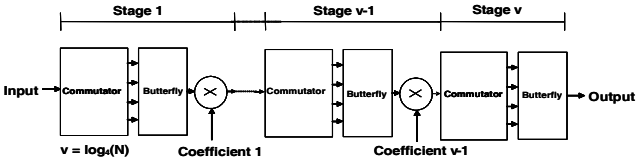

Fig. 1. N-point radix-4 pipelined FFT processor architecture

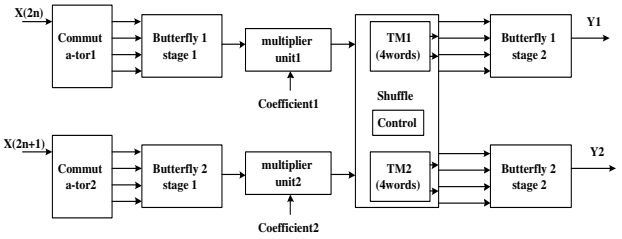

Fig. 2. Block diagram of 16-point 2-parallel-pipelined FFT architecture

2. PARALLEL-PIPELINED ARCHITECTURES

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

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

where $W_N = e^{-j2\pi/N}$. W_N is twiddle factor or coefficient. The FFT is the speed-up algorithm of DFT. For example, a N-point radix-4 pipelined FFT processor is shown in Fig.1. We explore to allocate two radix-4 PEs in each stage of the 16-point pipelined FFT. The structure is shown in Fig.2, namely 2-parallel-pipelined FFT. It can achieve double throughput, compared to the pipelined FFT at the same operation frequency. As shown in Fig.2, the input data are separated into two streams. Two commutators in stage1, each has half storage units of the commutator in pipelined FFTs, transform odd and even sequential input data to parallel data respectively. The coefficients are divided into two responding sections, in terms of even and odd. Two coefficient sections are fed

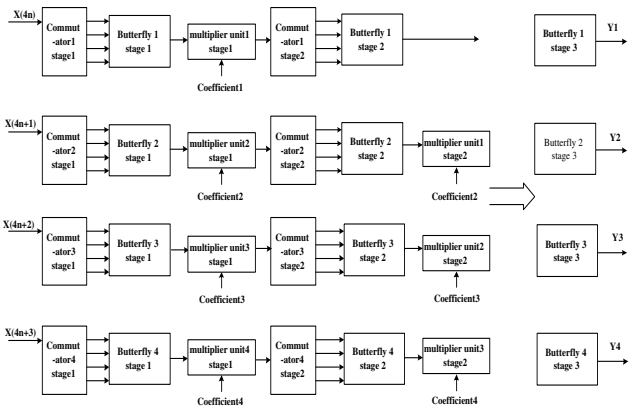

Fig. 3. Block diagram of 64-point 4-parallel-pipelined FFT architecture

into two complex multipliers, respectively. Due to the separate processing on odd and even data, a shuffle unit is needed in stage2 to implement the intersatge data shuffle. The shuffle unit is composed of two triple port SRAMs, each sized 4 words, and a addressing control unit. The 4 outputs from two triple port SRAMs are fed into each simplified butterfly unit. Each simplified butterfly unit only yields the half of all outputs.

For 64-point pipelined FFT, we employ four radix-4 PEs in each stage. The block diagram of the architecture is depicted in Fig.3, termed as 4-parallel-pipelined FFT. The architecture can achieve four times throughput, compared to the pipelined FFT. The input data are separated into four streams. There are four commutators in stage1 and stage2, respectively. Each of them has 1/4 size of the commutators in pipelined FFTs and implements the transform from sequential data to parallel data. The coefficients in each stage are divided into four sections, responding to four input streams. Only 3 complex multipliers are used in stage2, because the output of butterfly1 in stage2 is multiplied by stage 2's coefficient1, which only contains (7fff,0000). In this architecture, the number of PEs per column is same as the radix, hence, no shuffle units is needed in stage3. The multipliers' outputs in stage2 are fed into each of 4 simplified butterfly elements as shown in Fig.3. Each simplified butterfly element only yields 1/4 of all outputs.

3. LOW POWER TECHNIQUES

3.1. Multiplier-less architecture in parallel-pipelined FFTs

The multiplier-less architecture proposed in [6] can also be employed in parallel-pipelined FFTs. Taking 64-point FFT as an example, the number of coefficients for the second stage is 16. The coefficients are partitioned into 4 sections, each having 4 coefficients. Three of these sections are fed into 3 complex multiplier units respectively, since the other section only consists of 7fff0000. These coefficients are shown in Table 1. Among them, the trivial coefficients are (7fff,0000) and (0000,8000), which are the quantized 16-bit two's complement representation for (1,0) and (0,-1) respectively. In each set, two entries correspond to the cosine function (the real part, W_r) and the sine function (the imaginary part, W_i), respectively. For (7fff,0000) and (0000,8000), the complex multiplication is not necessary. Data can directly pass through the multiplier unit without any multiplication, when multiplied with (7fff,0000). Only an additional

Table 1. The coefficients for the second stage of 64-point 4-parallel-pipelined FFT

Multiplier 1	Multiplier 2	Multiplier 3
7fff,0000	7fff,0000	7fff,0000
7641,cf04	5a82,a57d	30fb,89be
5a82,a57d	0000,8000	a57d,a57d
30fb,89be	a57d,a57d	89be,30fb

unit, which swaps the real and imaginary parts of input data, and inverts the imaginary part, is needed for those data by (0000,8000) in multiplier unit2 [6]. For multiplier1, 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 coefficients. For example, a multiplication by a57d could be realized by first multiplying the data with 5a83, and then two's complementing the result. Note that a multiplication by 5a82 already exists. Therefore, the multiplication with 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 Canonical Signed-Digit (CSD) format as follow: $5a82$ (0101101010000010), 7641 $(1000 - 10 - 1001000001)$ and $30fb$ $(010 - 1000100000 - 10 - 1)$. The mixed use of Canonical Signed-Digit (CSD) and two's complement is for minimizing the number of addition/shift operations. According to the above representation, the required operations for the direct computation of the nontrivial complex multiplications for the three constants, are shown in Table 2.

Table 2. The operations required before common subexpression

operation	5a82X	7641X	30fbX
addition	5	2	2
subtraction	0	2	3
shift	6	4	4

However, if pre-computing $5X = X + X << 2$ and $65X = X + X << 6$, the multiplications can be written as follow:
$5a82X = 5X << 12 + 5X << 9 + 65X << 1$
$7641X = X << 15 + 65X << -5X << 9$
$30fbX = 65X << 8 - X << 12 - 5X$
The common subexpressions are 101 (5) and 1000001 (65). The pre-computation requires 2 additions and 2 shifts. The results of the pre-computation can be used for both the multiplication with real part (W_r) and the multiplication with imaginary part (W_i) of nontrivial coefficients. After the pre-computation, the operations required for the computations are shown in Table 3. As can be seen, the number of operations is significantly reduced, after the pre-computation.

Fig.4 shows the shift-and-addition module for the three constants in the multiplier-less1 architecture which replaces the multiplier1. 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. Firstly the input data are fed into the common subexpression block. The signal s1 indicates which constant channels will be chosen for processing the input data. Each channel carries out shift, negation and addition for the constant.

Table 3. The operations required after common subexpression

operation	5a82X	7641X	30fbX
addition	2	1	0
subtraction	0	1	2
shift	3	3	2

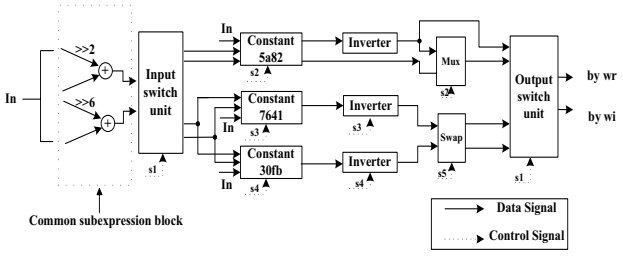

Fig. 4. Block diagram of the shift-and-addition module in multiplier-less1 unit of 64-point 4-parallel-pipeline FFT

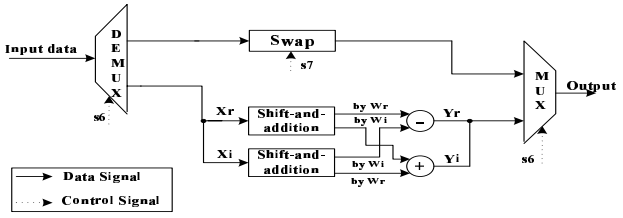

Fig. 5. Block diagram of the multiplier-less1 unit in 64-point 4-parallel-pipeline FFT

The control signal s2 indicates that the constant 7641 block outputs the product by either 7641 or 7642. Similarly, the signal s3 controls the output of constant 30fb block. The constant 5a82 block provides two products, 5a82X and a57dX. The invert 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,89be) or (7641,cf04). The output switch unit judges which couple of products are final outputs. The block diagram of the multiplier-less1 unit is depicted in Fig.5. Only those data, which multiply nontrivial complex coefficients, are fed into the shift-and-addition units, under the control of s5. Two shift-and-addition units are needed for the real part (X_r) and imaginary part (X_i) respectively. In the multiplier-less1 unit, 22 adders are used to substitute the four real multipliers in the complex multiplier unit. ROM unit storing coefficients is replaced by a FSM unit generating control signals (s1 - s5) in multiplier-less approach. Based on the above discussion, multiplier2 and multiplier3 can also be replaced by similar multiplier-less units. For multiplier3, three constants are used as shown in Table 1. The structure of multiplier-less3 is similar to those in Fig.4 and Fig.5. However, no inverter units are needed in shift-add modules and an additional controllable swap unit. In total, 20 adders are employed in this multiplier-less3 unit. For multiplier2, only the constant channel 5a82 is retained to process input data. This multiplier-less2 only contains 10 adders. Similar multiplier-less architectures can be applied to 16-point FFT and stage1's multiplier1 of 64-point FFT.

3.2. Low power commutator architecture

The commutator unit is one of the main power-consuming components in pipelined FFTs. Previous approaches to implement commutators include shift register (SR) [7], conventional dual port RAM architecture (DR) [8], and triple port RAM architecture (TR) [8]. In this paper, we employ a new architecture based on dual port RAMs, termed as IDR. The block diagram of IDR is depicted in Fig.6. IDR efficiently reduces the switching activity through maintaining the unused outputs of RAMs at their previous values. IDR also reduces the number of write operations to memory blocks with new interconnection topology. Table 4 illustrates which RAMs are enabled for write operation during each period. m_t is the butterfly operation factor for stage$t$. In radix-4 FFT, $0 < m_t < 3$. It can be seen from Table 4, in IDR architecture, each RAM block is enabled 5/3 times on average. Whereas, for DR and TR, each RAM block is enabled 4 and 10/3 times respectively [8]. Hence, our architecture is significantly more power efficient than both DR and TR, due to reduced memory access.

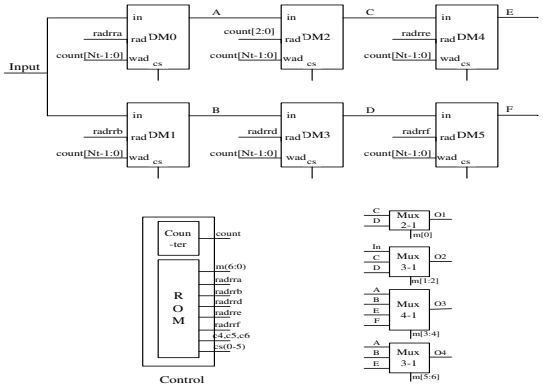

Fig. 6. Block diagram of IDR commutator

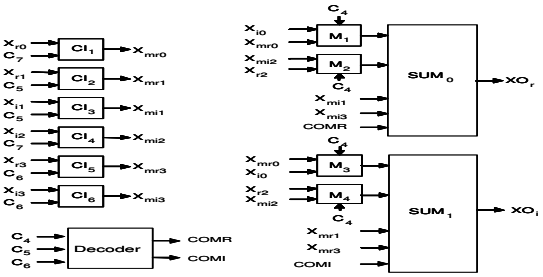

Fig. 7. Block diagram of the low power butterfly architecture

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

3.3. Low Power Butterfly

The conventional butterfly architecture consists of 6 adder/subtracters. In this paper, we proposed a low power butterfly architecture which employs two 5-input summation blocks to replace six adder/subtracters. Fig.7 shows the low power butterfly architecture (LB). Inverters (CI1 to CI6) are used to generate the normal or the one's complement form under the control of C5, C6 and C7. The signal C4 controls the four multiplexers (M1 to M4) for directing appropriate data to the inputs of the summation blocks. Two 5-input summation blocks (SUM0 and SUM1) are employed to generate the real and imaginary parts of the output respectively. An additional decoder unit is used to generate compensation for eliminating the error which results from the one's complement inversion controllable inverters .

4. SIMULATION RESULTS

Three different designs, Design I - III, and five different designs, Design I - V, have been implemented for 16-point and 64-point

parallel-pipelined FFTs. The 64-point and 16-point FFTs are synthesized at 16ns and 12ns clock cycles respectively to maximize timing slack, using Synopsys DesignCompiler targeting the UMC 0.18μ CMOS technology library. Synopsys DesignPower was used to evaluate power for 64-point and 16-point FFTs at 20ns and 14ns clock cycles respectively. Tables 5 and 6 provide information regarding to main modules for each implementation. For 16-point FFTs, all 3 designs use the same commutator architectures, where commutator1 is based on shift registers (SR), and Commutator2 utilizes a triple port SRAM based shuffle. The designs differ in their use of butterflies and multipliers. For example, Design I and II are based on Booth coded Wallace tree (wall) multipliers, whereas, Design III employs multiplier-less technique. Butterfly1 and 2 use adder/subtracter architecture (add-sub) for Design I. However, Design II and III utilize proposed low power butterfly architecture (LB). For 64-point FFTs, all designs employ 4 sum units for butterfly in stage3 (Butterfly3). The multiplier-less approach is employed in stage2 of Designs III - V and stage1 of Design V. Commutator1 is realized with dual-port RAMs (DR) for Designs I - III, and with IDR for Designs IV and V. For all designs, Commutator2 is implemented with shift registers.

Table 5. Implemented architectures for 16-point FFT

Main modules	I	II	III
Commutator1	SR	SR	SR
Butterfly1	2 X add-sub	2 X LB	2 X LB
Multiplier1	2 X wall	2 X wall	2 X mless
Commutator2	shuffle (TM)	shuffle (TM)	shuffle (TM)
Butterfly2	2 X add-sub	2 X LB	2 X LB

Table 6. Implemented architectures for 64-point FFT

Main modules	I	II	III	IV	V
Butterfly1	4 X add-sub	4X LB	4X LB	4X LB	4X LB
Butterfly2	4 X add-sub	4X LB	4X LB	4X LB	4X LB
Butterfly3	4 X sum	4X sum	4X sum	4X sum	4X sum
Commutator1	DR	DR	DR	IDR	IDR
Commutator2	SR	SR	SR	SR	SR
Multiplier1	4 X wall	4X wall	4X wall	4X wall	mless, 3 X wall
Multiplier2	3 X wall	3X wall	3X mless	3X mless	3X mless

The comparative power and area results are illustrated in Fig.8 and 9 respectively. Clearly, the best power result for 16-point FFT is achieved by Design III, which has 31% power saving, compared to Design I. For 64-point FFTs, Design V achieves the best power saving with 30%, compared to Design I. For area comparison, in 16-point FFT, Design III also has 20% area saving, as compared to Design I. For 64-point FFT, Design III has the best area result, followed by Designs IV and V.

5. CONCLUSION

The paper proposes two parallel-pipelined architectures for 64-point and 16-point FFTs. Several novel low power techniques: multiplier-less, IDR commutator and LB butterfly are presented. A number of 64-point and 16-point FFTs based on the above techniques were implemented. The impact of parameterization on power/area performance has been compared. Based on the combination of above mentioned low power techniques proposed by

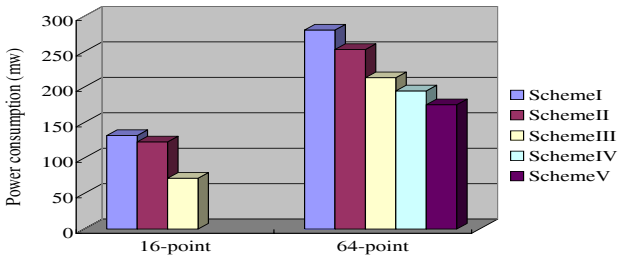

Fig. 8. Power consumption comparison

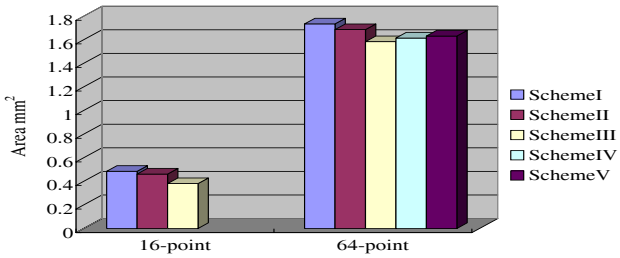

Fig. 9. Area comparison

the authors, up to 31% power saving and 20% area saving were achieved, for 64-point and 16-point parallel-pipelined FFTs.

6. REFERENCES

[1] S. Hong, S. Kim, M. C. Papefthymiou, and W. E. Stark, "Power-complexity analysis of pipelined vlsi fft architectures for low energy wireless communication applications," in *Circuits and Systems, 1999. 42nd Midwest Symposium*, Aug. 1999, vol. 1, pp. 8–11.

[2] Steve F. Gorman and Jeffrey M. Wills, "Partial column fft pipelines," *IEEE Transactions on circuits and systems*, vol. 42, no. 6, June 1995.

[3] Bevan M. Bass, "A low-power high-performance 1024-point FFT processor," *IEEE Journal of Solid-State Circuits*, vol. 34, no. 3, pp. 380–387, March 1999.

[4] L. Jia, Y. Gao, J. Isoaho, and H. Tenhunen, "A new vlsi oriented fft algorithm and implementation," in *Eleventh annual IEEE International ASIC confernece*, 1998, pp. 337–341.

[5] 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, pp. 484–493, March 2004.

[6] Wei Han, A. T. Erdogan T. Arslan, and M. Hasan, "A novel low power pipeliend FFT based on subexpression sharing for wireless LAN applications," in *IEEE Signal Processing Systems Workshop, 2004. (SIPS 2004)*, Oct. 2004, pp. 83–88.

[7] Bi Guoan and E. Jones, "A pipelined FFT processor for word sequential data," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, pp. 1982–1985, Dec. 1989.

[8] M. Hasan and T. Arslan, "A triple port ram based low power commutator architecture for a pipelined fft processor," in *Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium*, May 2003, vol. 5, pp. 353–356.