19

Reduced Logic and Low-Power FFT Architectures for Embedded Systems

Erdal Oruklu, Jafar Saniie and Xin Xiao
Illinois Institute of Technology
USA

1. Introduction

Discrete Fourier Transform (DFT) is one of the core operations in digital signal processing and communication systems. Many fundamental algorithms can be realized by DFT, such as convolution, spectrum estimation, and correlation. Furthermore, DFT is widely used in standard embedded system applications such as wireless communication protocols requiring Orthogonal Frequency Division Multiplexing (Wey et al., 2007), and radar image processing using Synthetic Aperture Radar (Fanucci et al., 1999). In practice, DFT is difficult to implement directly due to its computational complexity. To reduce the degree of computation, Cooley and Tukey proposed the well-known Fast Fourier Transform (FFT) algorithm, which reduces the calculation of N-point DFT from $O(N^2)$ to $O(N/2 log_2 N)$. (Proakis & Manolakis, 2006). Nevertheless, for embedded systems, in particular portable devices; efficient hardware realization of FFT with small area, low-power dissipation and real-time computation is a significant challenge. The challenge is even more pronounced when FFTs with large transform lengths (>1024 points) need to be realized in embedded hardware. Therefore, the objective of this research is to investigate hardware efficient FFT architectures, emphasizing compact, low-power embedded realizations.

As VLSI technology evolves, different architectures have been proposed for improving the performance and efficiency of the FFT hardware. Pipelined architectures are widely used in FFT realization (Li & Wanhammar, 1999; He & Torkelson, 1996; Hopkinson & Butler, 1992; Yang et al., 2006) due to their speed advantages. Higher radix (Hopkinson & Butler, 1992; Yang et al., 2006) and multi-butterfly (Bouguezel et al., 2004; X. Li et al., 2007) structures can also improve the performance of the FFT processor significantly, but these structures require substantially more hardware resources. Alternatively, shared memory based schemes with a single butterfly calculation unit (Cohen, 1976; Ma, 1994, 1999; Ma & Wanhammar, 2000; Wang et al., 2007) are preferred in many embedded FFT processors since they require least amount of hardware resources. Furthermore, "in-place" addressing strategy is a practical choice to minimize the amount of data memory. With "in-place" strategy, the two outputs of the butterfly unit can be written back to the same memory locations of the two inputs, and replace the old data. For in-place FFT processing, two data read and two data write operations occur at every clock cycle. Multiple memory banks and conflict-free addressing logic are required to realize four data accesses in one clock cycle. Consequently, a typical FFT processor is composed of three major components: i) butterfly calculation units, ii) conflict free address generators for both data and coefficient accesses and iii) multi-bank memory units.

In this study, several techniques are developed for reducing the hardware logic and power requirements for these three components:

1. In order to optimize the conflict free addressing logic, a modified butterfly structure with input/output exchange circuits is presented in Section 2.
2. CORDIC based FFT algorithms are presented for multiplier-less and coefficient memory-less implementation of the butterfly unit in Section 3.
3. Memory bank partitioning and bitline segmentation techniques are presented for dynamic power reduction of data memory accesses. Furthermore, a special coefficient memory addressing logic which reduces the switching activity is proposed in Section 4.

Case studies with ASIC and FPGA synthesis results demonstrate the performance gains and feasibility of these FFT implementations on embedded systems.

2. Hardware efficient realization of fast Fourier transform

There is an ongoing interest in hardware efficient FFT architectures. Cohen (Cohen, 1976) introduced a simplified control logic for FFT address generation, which is composed of parity checks, barrel shifters and counters based on the fact that two data addresses of every butterfly operations differ in their parity. Ma (Ma, 1999) proposed a method to realize the radix-2 addressing logic which reduces the address generation delay by avoiding parity check (XOR operations), but barrel shifters are still needed. Furthermore, Ma's approach is not "in-place", so more registers and related control logic are needed to buffer the interim data to avoid the memory conflict. Yang (Yang et al., 2006) proposed a locally pipelined radix-16 FFT realized by two radix-2 deep feedback (R2SD^{2}F) butterflies. This architecture can improve the throughput of the FFT processing and reduce the complex multipliers and adders compared to other pipelined methods, but it needs extra memory and there is significantly more coefficient access due to radix-16 implementation. Li (X. Li et al., 2007) proposed a mixed radix FFT architecture, which contains one radix-2 butterfly and one radix-4 butterfly. The two butterflies share the multipliers, which reduce the hardware consumption, but the address generation is based on XOR logic, and similar to Cohen's design. Next section describes in detail addressing schemes that emphasize reduced hardware.

2.1 Conflict-free addressing for FFT

The N-point discrete Fourier transform is defined by

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \qquad k = 0,1,...,N-1, \quad W_N^{nk} = e^{-j\frac{2\pi}{N}nk} \tag{1}$$

Fig. 1 shows the signal flow graph of 16-point decimation-in-frequency (DIF) radix-2 FFT (Proakis & Manolakis, 2006). FFT algorithm is composed of butterfly calculation units:

$$x_{m+1}(p) = x_m(p) + x_m(q) \tag{2}$$

$$x_{m+1}(q) = [x_m(p) - x_m(q)] W_N^r \tag{3}$$

Equations (2), (3) describe the radix-2 butterfly calculation at Stage m as shown in Fig. 2. Parallel and "in-place" butterfly operation using two memory banks of two-port memory

units requires that the two inputs of any butterfly are read from different banks of memory and the two outputs are written to the same address locations as the inputs. As shown in Fig. 1, in the conventional FFT addressing scheme, only the butterflies in the first stage satisfy this requirement. Two inputs and two outputs of butterfly operations in all other stages are originating from and sinking to the same memory bank. Therefore, a special addressing scheme is required to prevent the conflicting addresses.

Cohen (Cohen, 1976) used parity check to separate the data into two memory banks. Fig. 3 is the signal flow graph of Cohen's approach and it shows that inputs and outputs of any butterfly stage utilize separate memory banks. The addresses of butterfly operations are "in-place" located. The drawback of Cohen's method is the address generation delay. In order to reduce the delay of the address generation, Ma (Ma, 1999) proposed an alternative addressing scheme which avoids using parity check. The signal flow graph of Ma's scheme is shown in Fig. 4. In Ma's scheme, two inputs of a butterfly unit originate from two separate memory banks but two outputs of the butterfly unit utilize the same memory bank. The inputs and outputs of a butterfly unit are not "in-place". Therefore, extra registers and related control logic are needed to buffer the outputs of the butterfly until next butterfly calculation is finished in order to realize the "in place" operation. Compared to Cohen's approach which uses both parity check and barrel shifters, Ma's method needs only barrel shifters and avoids parity check, resulting in a reduced address generation delay. However, Ma's approach consumes more hardware resources to realize the "in-place" operation.

In the following section, a hardware efficient FFT engine with reduced critical path delay is proposed. Addressing logic is reduced by using a butterfly structure which modifies the conventional one by adding exchange circuits at the input and output of the butterfly (Xiao, et al., 2008]. With this butterfly structure, the two inputs and two outputs of any butterfly can be exchanged; hence all data addresses in FFT processing can be reordered. Using this flexible input and output ordering, addressing logic is designed to be "in-place" and it does not need barrel shifters.

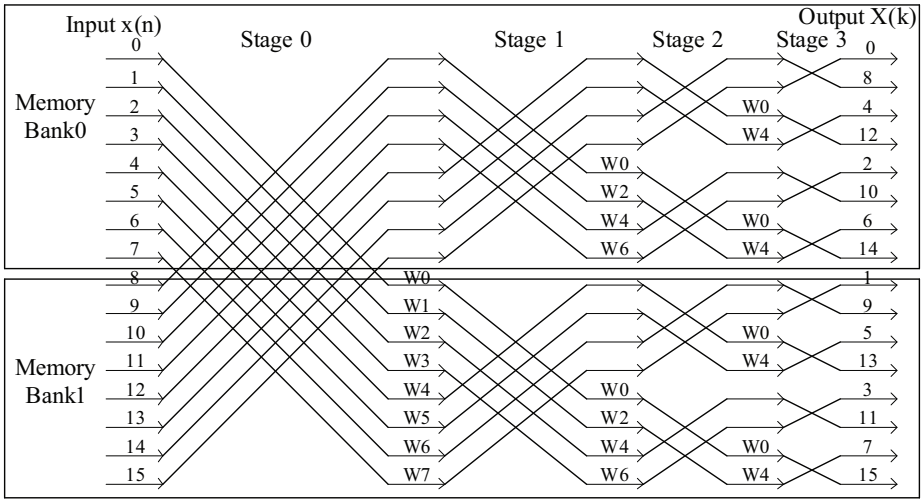


Fig. 1. Signal flow graph of 16-point FFT

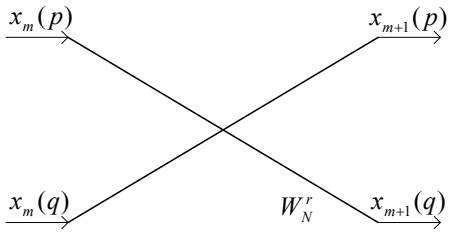


Fig. 2. Butterfly unit at stage m

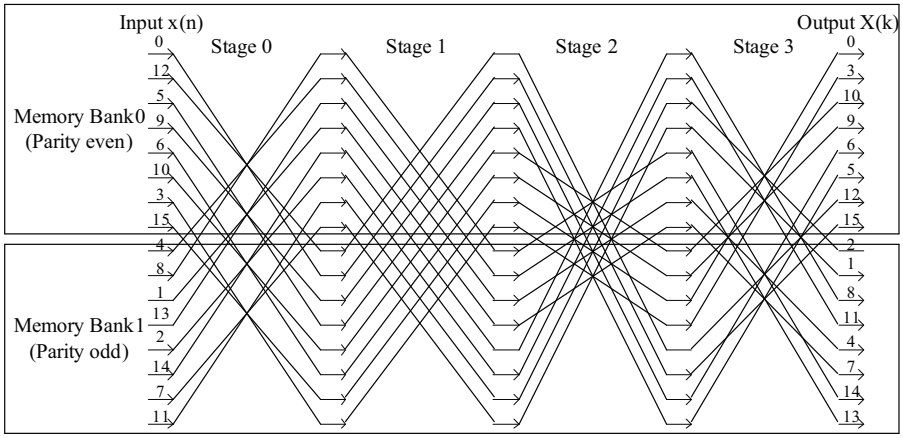


Fig. 3. Signal flow graph of 16-point FFT using Cohen's method (Cohen, 1976)

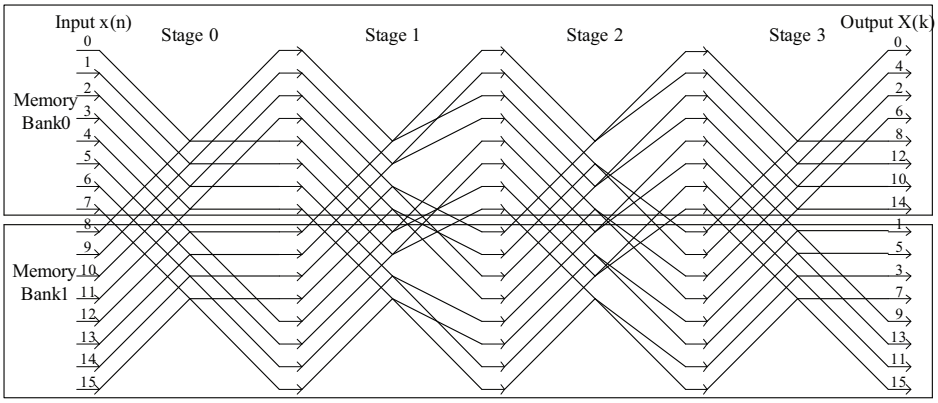


Fig. 4. Signal flow graph of 16-point FFT using Ma's method (Ma, 1999)

2.2 Reduced address generation logic with the modified butterfly FFT (mbFFT)

This addressing scheme is based on a **m**odified **b**utterfly **FFT** (mbFFT) structure, which is shown in Fig. 5. The main difference between the modified butterfly structure and the conventional one is the addition of two exchange circuits that are placed at both the input and the output of the butterfly unit. Each exchange circuit is composed of two (2:1) multiplexers; when the exchange control signal *C1* or *C2* is *1*, the data will be exchanged, otherwise they keep their locations.

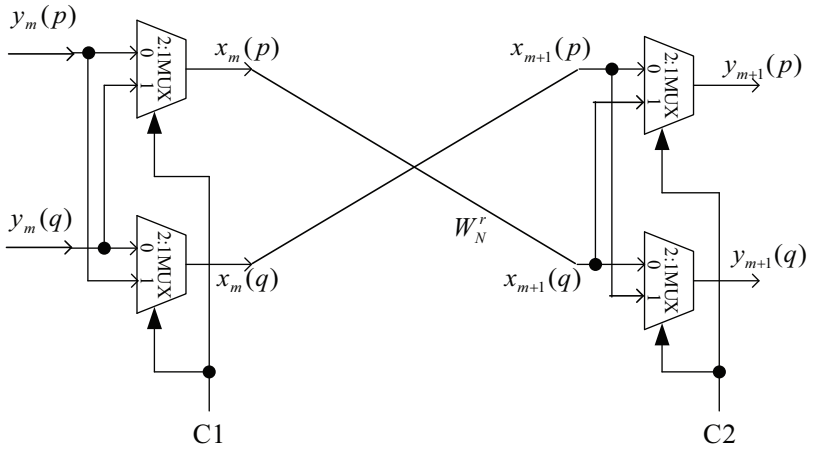


Fig. 5. Modified butterfly structure

Equation (4) shows the function:

$$
\begin{aligned}
&\textit{If C1=1:} && x_m(p) = y_m(q), \quad x_m(q) = y_m(p); \\
&\textit{Else:} && x_m(p) = y_m(p), \quad x_m(q) = y_m(q); \\
&\textit{If C2=1:} && y_{m+1}(p) = x_{m+1}(q), \quad y_{m+1}(q) = x_{m+1}(p); \\
&\textit{else:} && y_{m+1}(p) = x_{m+1}(p), \quad y_{m+1}(q) = x_{m+1}(q);
\end{aligned} \tag{4}
$$

Based on this butterfly structure, all data within the FFT processing can be reordered by setting the different values of the exchange control signals *C1* and *C2*. The control signals are chosen such that the input data always originate from two separate memory banks and output data are written to the same memory location in order to achieve in-place operation.

2.2.1 *16*-point mbFFT implementation

For 16-point mbFFT, the signal flow graph is shown in Fig. 6. In the figure, the butterfly inputs or outputs indicated by broken lines denote that the data have been exchanged. Fig. 7 shows the complete address generation architecture and components for 16-point FFT implementation. The address generation logic is composed of a 5-bit counter *D*, three

inverters, a 3-bit shifter, three (2:1) multiplexers, two (4:1) multiplexers, four multi-bit (2:1) multiplexers and delay elements. *Stage Counter* S indicates which stage of FFT is currently in progress and controls the two (4:1) multiplexers to generate the correct exchange control signals *C1* and *C2* for the butterfly operation. The 3-bit shifter shifts one bit at each stage and it controls three (2:1) multiplexers to generate the correct *M1* address. Since this technique is "in-place", the addresses for read and write are same with the exception of a delay introduced for compensating the butterfly computation time. Table I presents the counter values (control logic) which are used to generate the addresses for *M0* and *M1* memory banks.

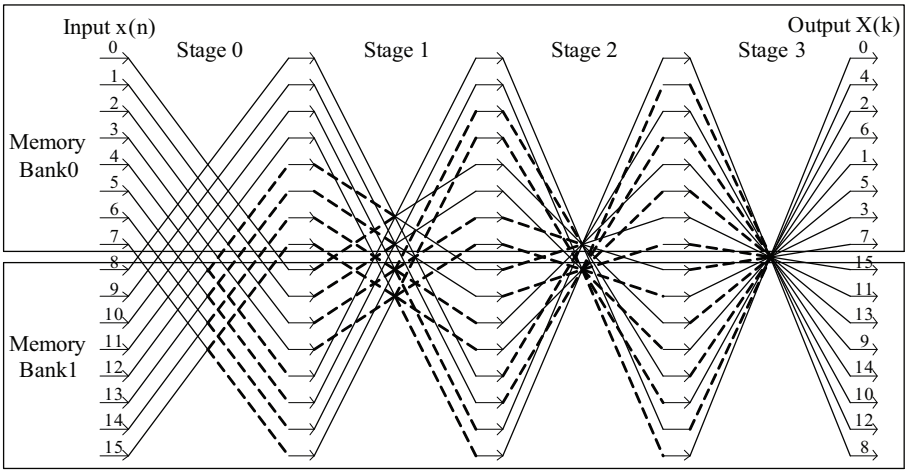


Fig. 6. Signal flow graph of 16-point mbFFT

Counter $B(b_2b_1b_0)$	Counter $\overline{B}(\overline{b}_2\overline{b}_1\overline{b}_0)$	Stage 0 (exchange control signal: C1=0,C2=b_2)		Stage 1 (exchange control signal: C1= b_2,C2= b_1)		Stage 2 (exchange control signal: C1= b_1,C2= b_0)		Stage 3 (exchange control signal: C1= b_0,C2=0)	
		Bank0 address $b_2b_1b_0$	Bank1 address $b_2b_1b_0$	Bank0 address $b_2b_1b_0$	Bank1 address $\overline{b}_2b_1b_0$	Bank0 address $b_2b_1b_0$	Bank1 address $\overline{b}_2\overline{b}_1b_0$	Bank0 address $b_2b_1b_0$	Bank1 address $\overline{b}_2\overline{b}_1\overline{b}_0$
000	111	000	000	000	100	000	110	000	111
001	110	001	001	001	101	001	111	001	110
010	101	010	010	010	110	010	100	010	101
011	100	011	011	011	111	011	101	011	100
100	011	100	100	100	000	100	010	100	011
101	010	101	101	101	001	101	011	101	010
110	001	110	110	110	010	110	000	110	001
111	000	111	111	111	011	111	001	111	000

Table 1. Address generation table for the 16-point mbFFT

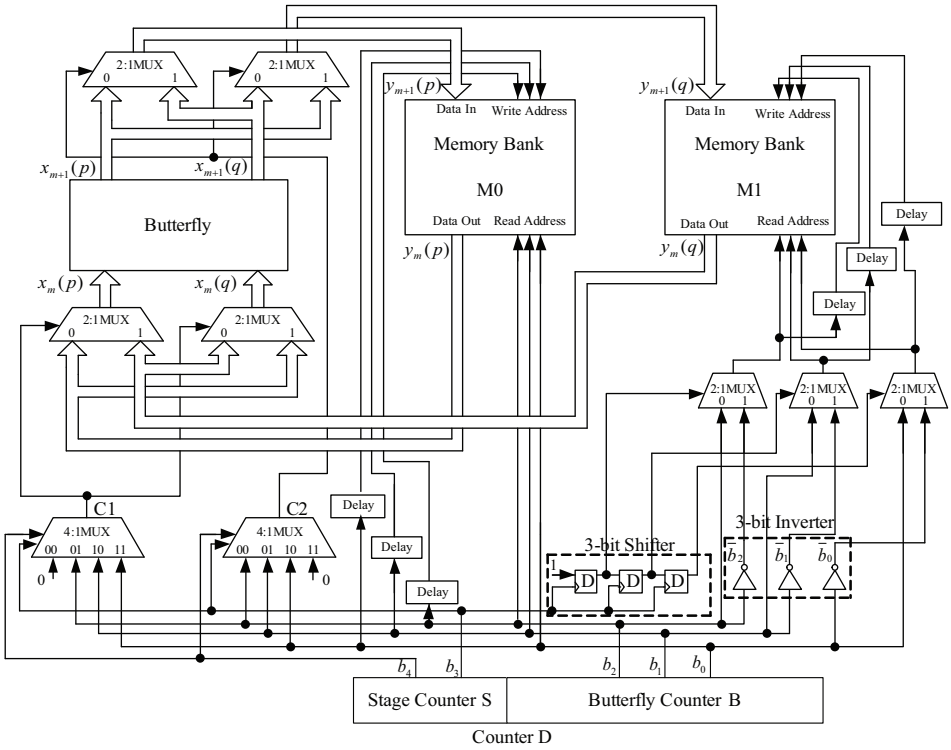


Fig. 7. Address generation circuits for 16-point mbFFT

2.2.2 *N*-point mbFFT implementation

In order to generalize the addressing scheme for $N = 2^n$ - point FFT, the necessary circuit components of the addressing and control logic can be listed as follows:

- (n-1)-bit *Butterfly Counter* $B = b_{n-2}b_{n-3}...b_1b_0$,
- (n-1) inverters which generate the complement of the *Butterfly Counter* $\overline{B} = \overline{b}_{n-2}\overline{b}_{n-3}...\overline{b}_1\overline{b}_0$ from counter B ,
- $\lceil \log_2 n \rceil$ - bit *Stage Counter* $S = (n-1),...,2,1,0$.
- *Two memory banks, Bank 0 (M0) and Bank 1 (M1).*

In practice, *Stage Counter S* and *Butterfly Counter B* can be combined to a single *counter D*, where *B* is the least significant (n-1) bits of *counter D*, and *S* is the most significant $\lceil \log_2 n \rceil$ bits of *counter D*. At any time, the read and write addresses of *M0* is exactly same as the value of *Butterfly Counter B*. For *M1*, the read and write address at Stage s is $\overline{b}_{n-2}\overline{b}_{n-3}...\overline{b}_{n-s-1}b_{n-s-2}...b_1b_0$, which is a combination of counters B and $\overline{B}$. The exchange control signal *C1* is equal to b_{n-s-1} (assume $b_{n-1} \equiv 0$), and *C2* is equal to b_{n-s-2} (assume $b_{-1} \equiv 0$). The address of twiddle factors at stage s is given by $b_{n-s-2}b_{n-s-3}...b_0 0...0$ (s '0's).

2.3 VLSI synthesis results

The mbFFT architecture is synthesized using TSMC CMOS 0.18μm technology. Synthesis is performed with Cadence Build Gates and Encounter tools. The synthesis results for 16-point FFT with 32-bit complex number input show a maximum clock frequency of 280MHz with 0.665mm² area and 0.645mW total power consumption for the complete FFT operation including butterfly unit, address generation unit, and memory circuits.

In order to compare different FFT addressing methods, the logic complexity can be evaluated similar to (Ma, 1999), based on gate counts. The sizes of some basic circuits and gates are listed in Table 2. Estimated gate count comparison for 1024-point FFT of 32-bit complex data (16-bit each for the real part and imaginary part) is shown in the Table 3. In terms of area, mbFFT scheme requires 24% fewer number of transistors. This reduction is mainly due to the difference in logic complexity of the multiplexers and barrel shifters. Based on the gate counts in Table 2 (and confirmed by synthesis results), *r*-input (r:1) multiplexer is approximately 4 times smaller than *(r-1)* barrel shifter in terms of area.

The delay of address generation for both read and write operations in the mbFFT addressing scheme is determined by two stages of multiplexers, where the first stage uses an *r*-input (r:1) multiplexer and the second stage uses a *2*-input (2:1) multiplexer for a 2^r-point FFT operation (see Fig 7). In (Ma, 1999), worst-case address generation delay is dominated by an *(r-1)*-bit barrel shifter and a (2:1)-multiplexer. An *(r-1)*-bit barrel shifter requires $\lceil \log_2(r-1) \rceil$ stages of (2:1) multiplexers in the critical path. Cohen's address generation method (Cohen, 1976) uses an *r*-bit parity check unit, an *(r-1)*-bit barrel shifter, and two (2:1) multiplexers in the critical path. Standard cell synthesis results in Table 4 show that the proposed mbFFT address generation scheme is faster compared to (Cohen, 1976) and (Ma, 1999) for large FFTs, due to the complex wiring and parasitic capacitances in barrel shifters and elimination of the parity-check operation.

Compared to a pipelined FFT architecture such as R2SD²F given in (Yang et al., 2006), the shared memory architectures such as mbFFT offer significantly reduced hardware cost and power consumption at the expense of (slower) throughput. R2SD²F requires log_4N-1 multipliers, $2log_4N$ adders and $10log_4N$ multiplexers for the butterfly operations in an *N*-point FFT. In contrast, only one multiplier, two adders and four multiplexers are used in the mbFFT architecture datapath. The latency (total clock cycles) of a pipelined FFT architecture is faster by a factor of $\frac{1}{2}\log_2 N$. However, the maximum achievable clock frequency would be less than the mbFFT design due the increased complexity of the R2SD²F datapath and address generation. Hence, for embedded applications, the proposed reduced logic, shared memory FFT approach with modified butterfly units presents a more viable solution.

Types of Gates and Circuits	No. of. Transistors
2-Input XOR	10
2-1 Multiplexer	6
10-1 Multiplexer	42
1-bit Register/Latch	10
9-bit Counter	182
13-bit Counter	270
9-bit Barrel Shifter	152
10-bit Barrel Shifter	168

Table 2. Transistor counts for CMOS cells (Ma, 1999)

Design Schemes	Components		Transistor Counts
	Quantity	Type	
Proposed mbFFT Design	1	13-bit Counter	1562
	9	Inverters	
	1	9-bit Shifter	
	9	1-bit 2:1 Multiplexer	
	2	1-bit 10:1 Multiplexer	
	4	32-bit 2:1 Multiplexer	
	2	9-bit Latches	
(Ma, 1999)	1	13-bit Counter	2066
	2	9-bit Barrel Shifters	
	4	9-bit Latches	
	2	32-bit Latches	
	2	9-bit 2:1 Multiplexers	
	2	32-bit 2:1 Multiplexers	
(Cohen, 1976)	1	13-bit Counter	1924
	1	9-bit Counter	
	2	9-bit Latch	
	2	10-bit Barrel Shifter	
	2	9-bit 2:1 Multiplexer	
	4	32-bit 2:1Multiplexer	
	1	9-bit Address Parity Generator	

Table 3. Address generation logic comparison for 1024-point FFT with 32-bit complex data

FFT size $=2^n$	Proposed mbFFT	(Ma, 1999)	(Cohen,1976)
n=4	1.28 ns	1.28 ns	1.82 ns
n=8	1.40 ns	1.53 ns	2.50 ns
n=10	1.47 ns	1.71 ns	2.61 ns
n=16	1.59 ns	1.85 ns	2.87 ns

Table 4. Delay comparison of address generation circuits

3. Multiplierless FFT architectures using CORDIC algorithm

In FFT processors, butterfly operation is the most computationally demanding stage. Traditionally, a butterfly unit is composed of complex adders and multipliers. A complex multiplier can be very large and it is usually the speed bottleneck in the pipeline of the FFT processor. The Coordinate Rotation Digital Computer (CORDIC) (Volder, 1959) algorithm is an alternative method to realize the butterfly operation without using any dedicated multiplier hardware. CORDIC algorithm is versatile and hardware efficient since it requires only add and shift operations, making it suitable for the butterfly operations in FFT (Despain, 1974). Instead of storing actual twiddle factors in a ROM, the CORDIC-based FFT processor needs to store only the twiddle factor *angles* in a ROM for the butterfly operation. In recent years, several CORDIC-based FFT designs have been proposed for different applications (Abdullah et al., 2009; Lin & Wu, 2005; Jiang, 2007; Garrido & Grajal, 2007). In (Abdullah et al., 2009), non-recursive CORDIC-based FFT was proposed by replacing the

twiddle factors in FFT architecture by non-iterative CORDIC micro-rotations. It reduces the ROM size, however, it does not eliminate it completely. (Lin & Wu, 2005) proposed a "mixed-scaling-rotation" CORDIC algorithm to reduce the total iterations, but it increases the hardware complexity. (Jiang, 2007) introduced Distributed Arithmetic (DA) to the CORDIC-based FFT algorithms, but the DA look-up tables are costly in implementation. (Garrido & Grajal, 2007) proposed a memory-less CORDIC algorithm to reduce the memory requirements for a CORDIC-based FFT processor by using only shift operations for multiplication.

Conventionally, a CORDIC-based FFT processor needs a dedicated memory bank to store the necessary twiddle factor angles for the rotation. In our earlier work (Xiao et al., 2010), a modified CORDIC algorithm for FFT processors is proposed which eliminates the need for storing the twiddle factor angles. The algorithm generates the twiddle factor angles successively by an accumulator. With this approach, memory requirements of an FFT processor can be reduced by more than 20%. Memory reduction improves with the increasing radix size. Furthermore, the angle generation circuit consumes less power consumption than angle memory accesses. Hence, the dynamic power consumption of the FFT processor can be reduced by as much as 15%. Since the critical path is not modified with the CORDIC angle calculation, system throughput does not change.

In the following sections, CORDIC algorithm fundamentals and the design of the proposed memory efficient CORDIC-based FFT processor are described.

3.1 CORDIC algorithm

CORDIC algorithm was proposed by J.E. Volder (Volder, 1959). It is an iterative algorithm to calculate the rotation of a vector by using only additions and shifts. Fig. 8 shows an example for rotation of a vector V_i.

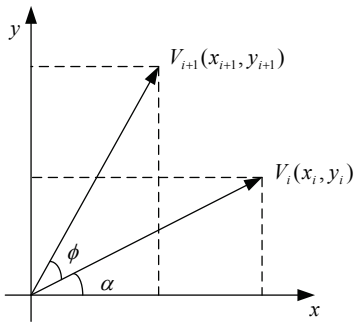


Fig. 8. Rotate vector $V_i(x_i, y_i)$ to $V_{i+1}(x_{i+1}, y_{i+1})$

The following equations illustrate the steps for calculating the rotation:

$$\begin{aligned} x_{i+1} &= r\cos(\alpha+\phi) = r(\cos\alpha\cos\phi - \sin\alpha\sin\phi) \\ &= x_i\cos\phi - y_i\sin\phi \end{aligned} \tag{5}$$

$$\begin{aligned} y_{i+1} &= r\sin(\alpha+\phi) = r(\sin\alpha\cos\phi + \cos\alpha\sin\phi) \\ &= y_i\cos\phi + x_i\sin\phi \end{aligned} \tag{6}$$

If each rotate angle ϕ is equal to $\arctan 2^{-i}$, then:

$$x_{i+1} = \cos\phi(x_i - y_i \cdot 2^{-i}) \tag{7}$$

$$y_{i+1} = \cos\phi(y_i + x_i \cdot 2^{-i}) \tag{8}$$

Since $\phi = \arctan 2^{-i}$, $\cos\phi$ can be simplified to a constant with fixed number of iterations:

$$x_{i+1} = K_i(x_i - y_i \cdot d_i \cdot 2^{-i}) \tag{9}$$

$$y_{i+1} = K_i(y_i + x_i \cdot d_i \cdot 2^{-i}) \tag{10}$$

where $K_i = \cos(\arctan(2^{-i}))$ and $d_i = \pm 1$. Product of K_i's can be represented by the K factor which can be applied as a single constant multiplication either at the beginning or end of the iterations. Then, (9) and (10) can be simplified to:

$$x_{i+1} = x_i - y_i \cdot d_i \cdot 2^{-i} \tag{10}$$

$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i} \tag{11}$$

The direction of each rotation is defined by d_i and the sequence of all d_i 's determines the final vector. d_i is given as:

$$d_i = \begin{cases} -1 & \text{if } z_i < 0 \\ +1 & \text{if } z_i \geq 0 \end{cases} \tag{12}$$

where z_i is called angle accumulator and given by

$$z_{i+1} = (z_i - d_i \cdot \arctan 2^{-i}) \tag{13}$$

All operations described through equations (10)-(13) can be realized with only additions and shifts; therefore, CORDIC algorithm does not require dedicated multipliers. CORDIC algorithm is often realized by pipeline structures, leading to high processing speed. Fig. 9 shows the basic structure of a pipelined CORDIC unit.

As shown in equation (1), the key operation of FFT is $x(n) \cdot W_N^{nk}$, ($W_N^{nk} = e^{-j\frac{2\pi}{N}nk}$). This is equivalent to "Rotate $x(n)$ by angle $-\frac{2\pi}{N}nk$ " operation which can be realized easily by the CORDIC algorithm. Without any complex multiplications, CORDIC-based butterfly can be fast. An FFT processor needs to store the twiddle factors in memory. CORDIC-based FFT doesn't have twiddle factors but needs a memory bank to store the rotation angles. For radix-2, N-point, m-bit FFT, $\frac{mN}{2}$ bits memory needed to store $\frac{N}{2}$ angles. In the next section, a new CORDIC based FFT design which does not require any twiddle factor or angle memory units is presented. This design uses a single accumulator for generating all the necessary angles instantly and does not have any precision loss.

3.2 Reduced memory CORDIC based FFT

Although several multi-bank addressing schemes have been used to realize parallel and pipelined FFT processing (Ma, 1999; Xiao et al., 2008), these methods are not suitable for the reduced memory CORDIC FFT. In these schemes, the twiddle factor angles are not in regular increasing order (see Table 5), resulting in a more complex design for angle generators. As shown in Table 6, using a special addressing scheme first proposed in (Xiao et al., 2009), the twiddle factor angles follow a regular, increasing order, which can be

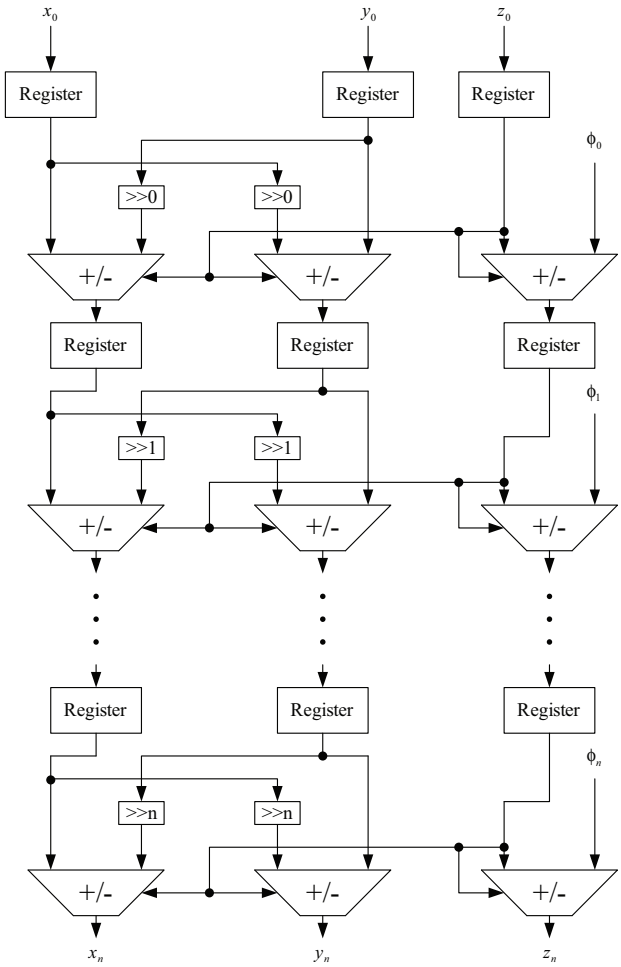


Fig. 9. Basic structure of a pipelined CORDIC unit

generated by a simple accumulator. Table 6 shows the address generation table of the 16-point radix-2 FFT. It can be seen that twiddle factor angles are sequentially increasing, and every angle is a multiple of the basic angle $2\pi/N$, which is $\pi/8$ for 16-point FFT. For different FFT stages, the angles increase always one step per clock cycle. Hence, an angle

generator circuit composed of an accumulator, and an output latch can realize this function, as shown in Fig. 10. Control signal for the latch that enables or disables the accumulator output is simple and it is based on the current FFT butterfly stage and RAM address bits $b_2b_1b_0$ (see Table 6).

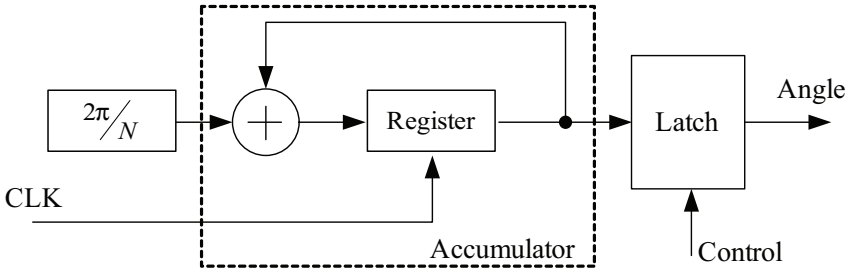


Fig. 10. Angle generator for the CORDIC based FFT

Butterfly Counter B(b2b1b0)	Stage 0		Stage 1		Stage 2		Stage 3	
	RAM address b0b2b1	Twiddle factor angle	RAM address b1b0b2	Twiddle factor angle	RAM address b2b1b0	Twiddle factor angle	RAM address b0b2b1	Twiddle factor angle
000	000	0	000	0	000	0	000	0
001	100	$4\pi/8$	010	$4\pi/8$	001	$4\pi/8$	100	0
010	001	$\pi/8$	100	0	010	0	001	0
011	101	$5\pi/8$	110	$4\pi/8$	011	$4\pi/8$	101	0
100	010	$2\pi/8$	001	$2\pi/8$	100	0	010	0
101	110	$6\pi/8$	011	$6\pi/8$	101	$4\pi/8$	110	0
110	011	$3\pi/8$	101	$2\pi/8$	110	0	011	0
111	111	$7\pi/8$	111	$6\pi/8$	111	$4\pi/8$	111	0

Table 5. Address generation table of Ma's (Ma, 1999) design for 16-point radix-2 FFT

Fig. 11 shows the architecture of the proposed *no-twiddle-factor-memory* design for radix-2 FFT. Four registers and eight 2-to-1 multiplexers are used. Registers are needed before and after the butterfly unit to buffer the intermediate data in order to group two sequential butterfly operations together. Therefore, the conflict-free "in-place" data accessing can be realized. This register-buffer design can be extended to any radix FFTs. For radix-2, the

structure can be simplified by using just 4 registers, but for radix-r FFT, $2 \times r^2$ registers are needed. Fig. 12 shows the structure for radix-*r* FFT.

Butterfly Counter B(b2b1b0)	Stage 0		Stage 1		Stage 2		Stage 3	
	RAM address b2b1b0	Twiddle factor angle	RAM address b0b2b1	Twiddle factor angle	RAM address b1b0b2	Twiddle factor angle	RAM address b2b1b0	Twiddle factor angle
000	000	0	000	0	000	0	000	0
001	001	$\pi/8$	100	0	010	0	001	0
010	010	$2\pi/8$	001	$2\pi/8$	100	0	010	0
011	011	$3\pi/8$	101	$2\pi/8$	110	0	011	0
100	100	$4\pi/8$	010	$4\pi/8$	001	$4\pi/8$	100	0
101	101	$5\pi/8$	110	$4\pi/8$	011	$4\pi/8$	101	0
110	110	$6\pi/8$	011	$6\pi/8$	101	$4\pi/8$	110	0
111	111	$7\pi/8$	111	$6\pi/8$	111	$4\pi/8$	111	0

Table 6. Address generation table for 16-point radix-2 FFT with the proposed angle generator

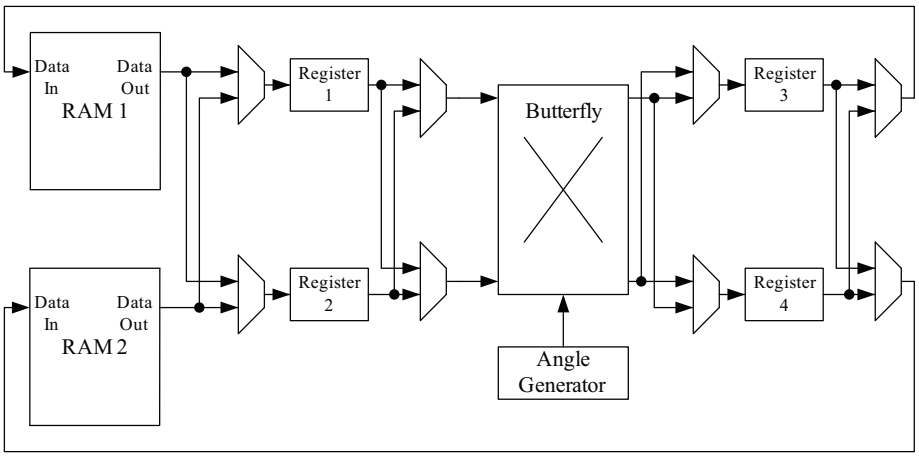


Fig. 11. Radix-2 FFT processor with no-twiddle-factor-memory

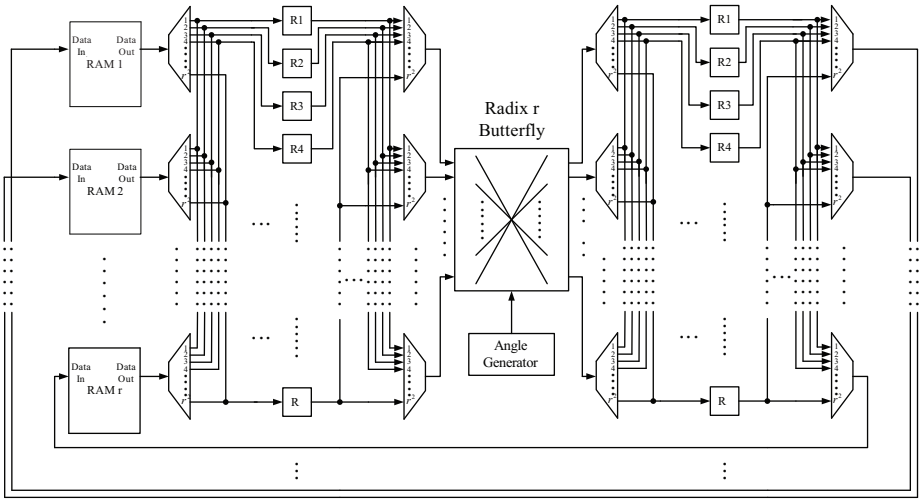


Fig. 12. Proposed radix-r CORDIC-based FFT

For an $N = 2^n$-point FFT, the addressing and control logic are composed of several components: An $(n-1)$-bit butterfly counter $B = b_{n-2}b_{n-3}...b_1b_0$ will provide the address sequences and the control logic of the angle generator. In stage S, the memory address is given by $b_{s-1}b_{s-2}...b_1b_0b_{n-2}b_{n-3}...b_s$, which is rotate right S bits of butterfly counter B. Meanwhile, the control logic of the latch of the angle generator is determined by the sequence of the pattern; $b_{n-2}b_{n-3}...b_s0...0$ (S "0"s).

For radix-2, $N = 2^n$-point, m-bit FFT, (each data is $2m$-bit complex number; m-bit each for the real part and imaginary part) by using the proposed angle generator, $\frac{5mN}{2}$ bits memory required by the conventional CORDIC can be reduced to $\frac{4mN}{2}$ which corresponds to 20% reduction. For higher radix FFT, the reduction is even more significant. For radix-r FFT, the saving is $\frac{(r-1)mN}{r}$ bits out of $\frac{(3r-1)mN}{r}$, which converges to 33.3% reduction.

Due to finite wordlength, as the accumulator operates, the precision loss will accumulate as well. In order to address this issue, more bits (wider wordlength) can be used for the fundamental angle $2\pi/N$ and the accumulator logic. For example, for 1024-point FFT, the accumulator is extended from 16 bits to 21 bits and no precision loss is observed compared to a conventional angle-stored CORDIC FFT processor.

3.3 FPGA synthesis results

The proposed reduced memory CORDIC based FFT designs for both radix-2 and radix-4 FFT algorithms have been realized by Verilog-HDL and implemented on an FPGA chip (STRATIX-III EP3SE50C2). Synthesis results shown in Table 7 show that these designs can reduce memory usage for FFT processors without any tangible increase in the number of logic elements used when compared against the conventional CORDIC implementation (i.e.,

angles are stored in memory). Furthermore, dynamic power consumption is reduced (up to 15%) with no delay penalties. The synthesis results match with the theoretical analysis.

		Radix-2		Radix-4	
		Proposed CORDIC FFT (angle generator)	Conventional CORDIC FFT (angles stored)	Proposed CORDIC FFT (angle generator)	Conventional CORDIC FFT (angles stored)
256-point FFT	Total logic elements	1,427 (19-bit accum.)	1,386	5,892 (20-bit accum.)	5,763
	Total memory	8,672	10,720	8,728	11,800
	Dynamic Power	136.87 mW	156.22mW	437.53 mW	495.06 mW
1024-point FFT	Total logic elements	1,773 (21-bit accum.)	1,718	5,991 (22-bit accum.)	5,797
	Total memory	33,248	41,440	33,304	45,592
	Dynamic Power	135.07 mW	175.98 mW	439.40 mW	496.64 mW
4096-point FFT	Total logic elements	1,809 (23-bit accum.)	1,757	5,993 (24-bit accum.)	5,863
	Total memory bits	131,552	164,320	131,608	180,760
	Dynamic Power	212.78 mW	242.85 mW	501.11 mW	571.72 mW

Table 7. FPGA implementation results for Radix-2 and Radix-4 FFT

4. Low-power FFT addressing schemes

For embedded applications, power dissipation is often a crucial design goal. (Ma & Wanhammar, 1999) proposed a new addressing logic to improve the memory accessing speed and to reduce the power consumption. (Hasan et al., 2003) designed a new coefficient ordering method to reduce the power consumption of radix-4 short-length FFTs. Gate-level algorithms have also been proposed (Zainal at al., 2009; Saponara, 2003) to reduce the FFT processor's power consumption by lower supply voltage techniques and/or voltage scaling. Power consumption of FFT processors can be significantly reduced by optimizing both data and coefficient memory accesses. Dynamic power consumption in CMOS circuits can be characterized by the following equation:

$$P_{dynamic} = \alpha \cdot C_{total} \cdot V_{DD}^2 \cdot f \tag{14}$$

where a is the switching activity, V_{DD} is the supply voltage, f is the frequency and C_{total} is the total switching capacitance charging and discharging in the circuit. In particular,

architectural techniques can reduce two parameters in (14), C_{total} and α. These techniques are discussed next: First, a multi-bank memory structure is proposed for data memory accesses, resulting in reduced overall capacitance load on the SRAM bit-lines. Second, a new butterfly calculation order reduces the memory access frequency for twiddle factors and minimizes the switching activity.

4.1 Memory bank partitioning

Since FFT operation largely consists of data and twiddle factor memory accesses, it is desirable to reduce the power dissipation caused by memory accesses. Memory bank partitioning and bitline segmentation is an important technique to reduce the power dissipation in SRAMs. The bitlines (each read and write port is associated with one bitline) in the SRAM logic are a significant source of energy dissipation due to the large capacitive load. This capacitance has two components, wire capacitance of the bitlines and the diffusion capacitance of each pass transistor connecting bitline to bitcells. Hence, the capacitive load increases linearly with the components attached to the bitline i.e., the number of words or size of the memory. In order to reduce this large capacitive load, the data memory can be partitioned into four memory banks instead of two. As a result, the capacitive loading in each memory bank is lowered since the bitline wire length and the number of pass transistors connected to the bitline is now only one fourth of the original bitline. The first two memory banks, *bank0* and *bank1* are accessed by the upper leg of the butterfly structure, and *bank2* and *bank3* are accessed by the lower leg of the butterfly (see Fig. 13). The most significant bit (MSB) of the addresses determine which two memory banks will be accessed; the remaining two memory banks will be inactive. Multi-bank memory structure has been proposed before (Ma & Wanhammar, 2000), but a major advantage of the proposed addressing scheme is that the memory bank switching occurs only once in the middle of a stage. In the first half of the stage, same two memory banks are used and in the second half of the stage, the other two memory banks are accessed. There is no precharging and discharging of bitlines in the inactive memory banks.

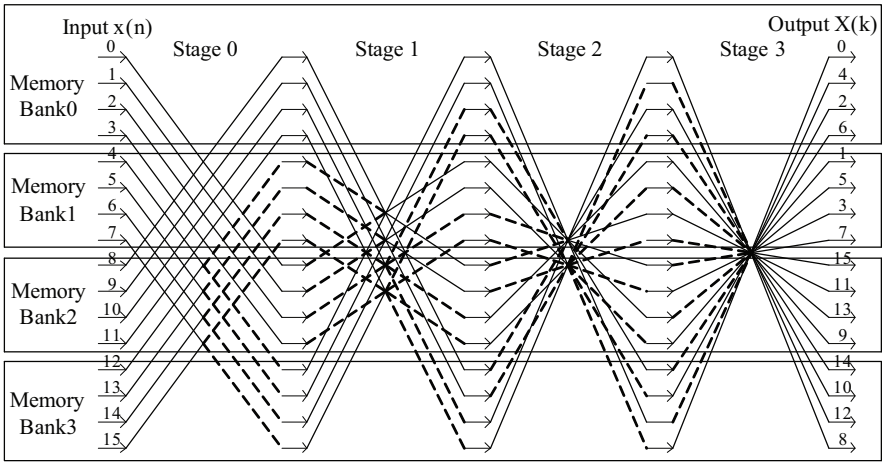


Fig. 13. Signal flow graph of 16-point FFT using memory partitioning

4.2 Reordering coefficient access sequence

The mbFFT architecture (see Section 2.2) can be used to generate the addressing scheme for reducing twiddle factor memory accesses and switching activity power. The twiddle factor access sequence is optimized for minimizing data bus changes. For all butterfly stages, the twiddle factor addresses are ordered in such a way that the twiddle factors at the same address are grouped together and accessed sequentially. This way, the twiddle factor ROM is not accessed every clock cycle. Reordering of the coefficient access sequences is shown in Table 8 and Table 9. For example, in *stage 1* in Table 9, only 8 accesses are needed instead of 16, and in *stage 2,* only 4 accesses instead of 8 and so on.

Counter $B(b_2b_1b_0)$	Stage 0			Stage 1		
	Bank 0,1 address $b_2b_1b_0$	Twiddle factor address b_1b_0	Bank 2,3 address $b_2b_1b_0$	Bank 0,1 address $b_2b_0b_1$	Twiddle Factor address b_10	Bank 2,3 address $\bar{b}_2b_0b_1$
000	000	00	000	000	00	100
001	001	01	001	010	00	110
010	010	10	010	001	10	101
011	011	11	011	011	10	111
100	100	00	100	100	00	000
101	101	01	101	110	00	010
110	110	10	110	101	10	001
111	111	11	111	111	10	011

Stage 2			Stage 3		
Bank0,1 address $b_2b_1b_0$	Twiddle factor address 00	Bank2,3 address $\bar{b}_2\bar{b}_1b_0$	Bank0,1 address $b_2b_1b_0$	Twiddle factor address b_00	Bank2,3 address $\bar{b}_2\bar{b}_1\bar{b}_0$
000	00	110	000	00	111
001	00	111	001	00	110
010	00	100	010	00	101
011	00	101	011	00	100
100	00	010	100	00	011
101	00	011	101	00	010
110	00	000	110	00	001
111	00	001	111	00	000

Table 8. Address generation table for the 16-point, reduced memory access FFT

Counter $B(b_3b_2b_1b_0)$	Stage 0			Stage 1		
	Bank 0,1 address $b_3b_2b_1b_0$	Twiddle factor address $b_2b_1b_0$	Bank 2,3 address $b_3b_2b_1b_0$	Bank 0,1 address $b_3b_0b_2b_1$	Twiddle factor Address b_2b_10	Bank 2,3 address $\bar{b}_3b_0b_2b_1$
0000	0000	000	0000	0000	000	1000
0001	0001	001	0001	0100	000	1100
0010	0010	010	0010	0001	010	1001
0011	0011	011	0011	0101	010	1101
0100	0100	100	0100	0010	100	1010
0101	0101	101	0101	0110	100	1110
0110	0110	110	0110	0011	110	1011
0111	0111	111	0111	0111	110	1111
1000	1000	000	1000	1000	000	0000
1001	1001	001	1001	1100	000	0100
1010	1010	010	1010	1001	010	0001
1011	1011	011	1011	1101	010	0101
1100	1100	100	1100	1010	100	0010
1101	1101	101	1101	1110	100	0110
1110	1110	110	1110	1011	110	0011
1111	1111	111	1111	1111	110	0111

Stage 2			Stage 3			Stage 4		
Bank0,1 address $b_3b_1b_0b_2$	Twiddle factor address b_200	Bank2,3 address $\bar{b}_3\bar{b}_1b_0b_2$	Bank0,1 address $b_3b_2b_1b_0$	Twiddle factor address 000	Bank2,3 address $\bar{b}_3\bar{b}_2\bar{b}_1b_0$	Bank0,1 address $b_3b_0b_2b_1$	Twiddle factor Address 000	Bank2,3 address $\bar{b}_3\bar{b}_0\bar{b}_2\bar{b}_1$
0000	000	1100	0000	000	1110	0000	000	1111
0010	000	1110	0001	000	1111	0100	000	1011
0100	000	1000	0010	000	1100	0001	000	1110
0110	000	1010	0011	000	1101	0101	000	1010
0001	100	1101	0100	000	1010	0010	000	1101
0011	100	1111	0101	000	1011	0110	000	1001
0101	100	1001	0110	000	1000	0011	000	1100
0111	100	1011	0111	000	1001	0111	000	1000
1000	000	0100	1000	000	0110	1000	000	0111
1010	000	0110	1001	000	0111	1100	000	0011
1100	000	0000	1010	000	0100	1001	000	0110
1110	000	0010	1011	000	0101	1101	000	0010
1001	100	0101	1100	000	0010	1010	000	0101
1011	100	0111	1101	000	0011	1110	000	0001
1101	100	0001	1110	000	0000	1011	000	0100
1111	100	0011	1111	000	0001	1111	000	0000

Table 9. Address generation table for the 32-point, reduced memory access FFT

Equations (15) and (16) show the twiddle factor memory access frequency for shared memory methods (Xiao et al., 2008) and the proposed reduced memory access method for $N = 2^n$ point FFT.

Conventional method: $$\frac{N}{2} \times (n-2) + 2 = \frac{N}{2}\left(\left(\log_2 N\right) - 2\right) + 2 \tag{15}$$

Reduced memory access method: $$\sum_{i=2}^{n-1} 2^i + 2 = 2^n - 2 = N - 2 \tag{16}$$

Table 10 shows the twiddle factor memory access frequency for different FFT lengths. As FFT length increases, the power saving also scales up.

4.3 Implementation

To implement an $N = 2^n$ -point FFT with reduced coefficient memory accesses, an *(n-1)*-bit *Butterfly Counter* $B = b_{n-2}b_{n-3}...b_1b_0$, and a $\lceil \log_2 n \rceil$ -bit *Stage Counter* $S = (n-1), \ldots, 2,1,0$ is needed. In addition, one *(n-2)*-bit barrel shifter is used: Assume $RR(x_u x_{u-1} x_{u-2} \ldots x_1 x_0, v)$ indicates rotate-right counter $x_u x_{u-1} x_{u-2} \ldots x_1 x_0$ by v bit. At *stage s,* the read and write addresses of the upper legs of the butterfly is $A_u = RR(b_{n-3} \ldots b_1 b_0, s) = a_{n-3}a_{n-4} \ldots a_1 a_0$, and b_{n-2} decides if *bank0* or *bank1* will be accessed.

	16-point FFT	32-point FFT	64-point FFT	128-point FFT	256-point FFT	512-point FFT	1024-point FFT	2048-point FFT	4096-point FFT	8192-point FFT
Conventional FFT design	18	50	130	322	770	1794	4098	9218	20482	45058
Reduced memory access FFT design	14	30	62	126	254	510	1022	2046	4094	8190
Reduction	22%	40%	52%	61%	67%	72%	75%	78%	80%	82%

Table 10. Reduction in twiddle factor memory access frequency

For example, for the 32-point FFT shown in Table 9, at *stage 2*, the address of the upper legs of the butterfly is $RR(b_2b_1b_0, 2) = b_1b_0b_2$, and when b_3=0, memory *bank0* will be accessed, when b_3=*1*, memory *bank1* will be accessed. For the read and write addresses of the lower legs of the butterfly, *(n-2)* inverters are needed. The address is given by $\bar{a}_{n-3}\bar{a}_{n-4} \ldots \bar{a}_{n-s-1}a_{n-s-2} \ldots a_1a_0$, and b_{r-2} decides if *bank2* or *bank3* will be accessed at *stage 0*. At stage 0, when $b_{n-2} = 0$, *bank2* will be accessed. When $b_{n-2} = 1$, *bank3* will be accessed. For other stages $b_{n-2} = 0$ means *bank3* will be accessed, $b_{n-2} = 1$ means *bank4* will be accessed. The address of twiddle factors is given by $a_{n-s-3} \ldots a_0 0 \ldots 0$ (S '0's). Fig 14 shows the components of the address generation logic using mbFFT and four memory banks.

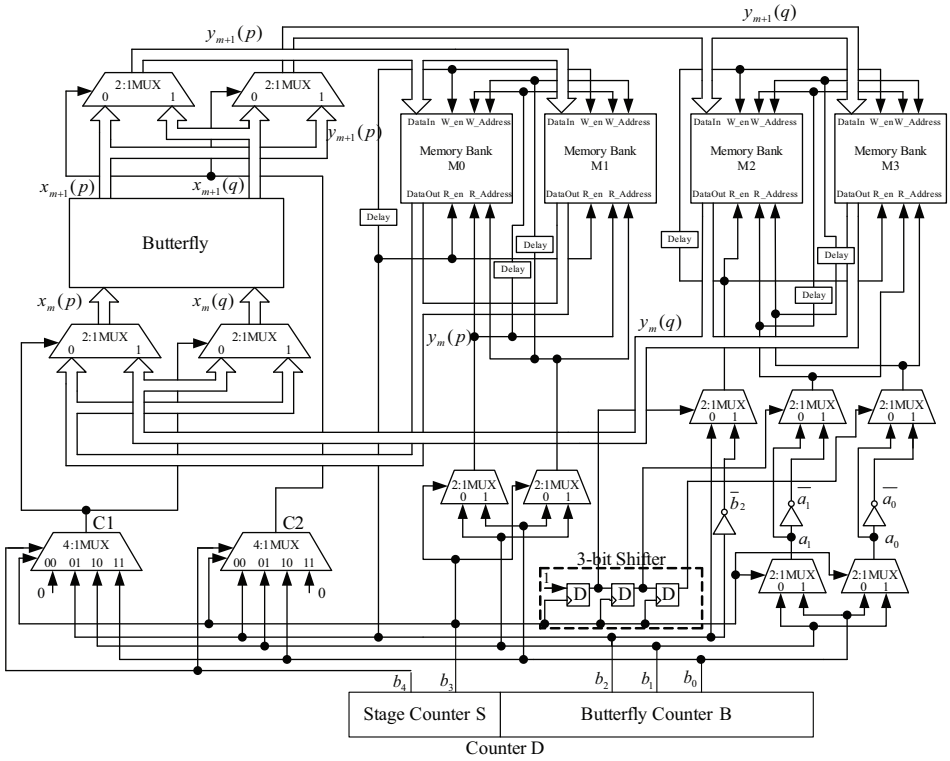


Fig. 14. Address generation circuits for low-power 16-point FFT using mbFFT and four memory banks

	Shared memory design (Xiao et al., 2008)			Power optimized design		
	Total power	Dynamic power	Static power	Total power	Dynamic power	Static power
512 point FFT	653.14mw	203.13mw	450.00mw	635.47mw	185.47mw	450.00mw
1024 point FFT	715.79mw	265.79mw	450.00mw	676.79mw	226.78mw	450.00mw
2048point FFT	840.49mw	390.49mw	450.00mw	764.31mw	314.31mw	450.00mw
4096 point FFT	1089.33mw	639.33mw	450.00mw	939.25mw	489.24mw	450.00mw
8192 point FFT	1595.13mw	1145.13mw	450.00mw	1289.17mw	839.17mw	450.00mw

Table 11. FPGA synthesis results – Reduction in dynamic power

4.4 FPGA synthesis results

The low-power FFT algorithm is implemented on an FPGA chip (ALTERA STRATIX EP1S25F780C5) with FFT length up to 8192 points as shown in Table 11. The synthesis results demonstrate that dynamic power reduction grows with the transform size, making this architecture ideal for applications requiring long FFT operations.

5. Conclusion

This study focused on hardware efficient and low-power realization of FFT algorithms. Recent novel techniques have been discussed and presented to realize conflict-free memory addressing of FFT. Proposed methods reorder the data and coefficient address sequences in order to achieve significant logic reduction (24% less transistors) and delay improvements within FFT processors. Multiplierless implementation of FFT is shown using a CORDIC algorithm that does not need any coefficient angle memory, resulting in 33% memory and 15% power reduction. Finally, optimization of FFT dynamic power consumption is presented through memory partitioning and reducing coefficient memory access frequency (26% power reduction for 8192 point-FFT).

6. References

Abdullah, S. S.; Nam, H.; McDermot, M. & Abraham, J. A. (2009). A High Throughput FFT Processor with No Multipliers. *IEEE International Conference on Computer Design*, pp. 485-490, 2009.

Bouguezel, S.; Ahmad, M. O. & Swamy, M. N. S. (2004). A New Radix-2/8 FFT Algorithm for Length-Q X 2^m DFTs. *IEEE Transactions on Circuits and Systems I*, vol. 51, no. 9, pp. 1723-1732, September 2004.

Cohen, D. (1976). Simplified Control of FFT Hardware. *IEEE Transactions on Acoustics, Speech, Signal Processing*, vol. 24, pp. 577–579, December 1976.

Despain, A. M. (1974). Fourier Transform Computers Using CORDIC Iterations. *IEEE Transactions on Computers*, vol. c-23, no.10, pp. 993-1001, October 1974.

Fanucci, L.; Forliti, M. & Gronchi, F. (1999). Single-Chip Mixed-Radix FFT Processor for Real-Time On-Board SAR Processing. *6th IEEE International Conference on Electronics, Circuits and Systems, ICECS '99*, vol. 2, pp. 1135-1138, September 1999.

Garrido, M. & Grajal, J. (2007). Efficient Memory-Less CORDIC for FFT Computation. *IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 2, no. 2, pp. 113-116, April 2007.

Hasan, M.; Arslan, T. & Thompson, J. S. (2003). A Novel Coefficient Ordering Based Low Power Pipelined Radix-4 FFT Processor for Wireless LAN Applications, *IEEE Transactions on Consumer Electronics*, vol. 49, no.1, pp. 128-134, February 2003.

He, S. S. & Torkelson, M. (1996). A New Approach to Pipeline FFT Processor. *Proceedings of 10th International Parallel Processing Symposium*, pp. 766-770, April 1996.

Hopkinson, T. M. & Butler, G. M. (1992). A Pipelined, High-Precision FFT Architecture. *Proceedings of the 35th Midwest Symposium Circuits and Systems*, vol. 2, pp. 835-838, August 1992.

Jiang, R. M. (2007). An Area-Efficient FFT Architecture for OFDM Digital Video Broadcasting. *IEEE Transactions on Consumer Electronics*, vol. 53, no. 4, pp. 1322-1326, 2007.

Li, W. D. & Wanhammar, L. (1999). A Pipeline FFT Processor. *Proceedings of IEEE Workshop on Signal Processing Syst*ems, pp. 654-662, October 1999.

Li, X.; Lai, Z. & Cui, J. (2007). A Low Power and Small Area FFT Processor for OFDM Demodulator. *IEEE Transactions on Consumer Electronics,* vol. 53, no. 2, pp. 274-277, May 2007.

Lin, C. & Wu, A. (2005). Mixed-Scaling-Rotation CORDIC (MSR-CORDIC) Algorithm and Architecture for High-Performance Vector Rotational DSP Applications. *IEEE Transactions on Circuits and Systems I*, vol. 52, no. 11, pp. 2385-2396, 2005.

Ma, Y. (1994). A Fast Address Generation Scheme for FFT Processors*, Chinese Journal Computers*, vol. 17, no. 7, pp. 505-512, July 1994.

Ma, Y. (1999). An Effective Memory Addressing Scheme for FFT Processors. *IEEE Transactions on Signal Processing*, vol. 47, no. 3, pp. 907–911, March 1999.

Ma, Y. & Wanhammar, L. (1999). A Coefficient Access Control for Low Power FFT Processors. *IEEE 42nd Midwest Symposium on Circuits and Systems*, vol.1, pp. 512-514, Aug. 1999.

Ma, Y. & Wanhammar, L. (2000). A Hardware Efficient Control of Memory Addressing for High-Performance FFT Processors. *IEEE Transactions on Signal Processing,* vol. 48, no. 3, pp. 917-921, March 2000.

Proakis, J. G.; & Manolakis, D. G. (2006). *Digital Signal Processing Principles, Algorithms, and Applications,* Prentice Hall, ISBN 978-0131873742.

Saponara, S.; Serafini, L. & Fanucci, L. (2003). Low-Power FFT/IFFT VLSI Macro Cell for Scalable Broadband VDSL Modem. *The 3rd IEEE International Workshop on System-on-Chip for Real-Time Applications,* pp.161-166, June 2003.

Volder, J. (1959). The CORDIC Trigonometric Computing Technique. *IEEE Transactions on Electronic Computers,* vol. EC-8, no. 8, pp. 330-334, September 1959.

Wang, Y.; Tang, Y.; Jiang, Y.; Chung, J.; Song, S. & Lim, M. (2007). Novel Memory Reference Reduction Methods for FFT Implementations on DSP Processors. *IEEE Transactions on Signal Processing,* vol. 55, no. 5, pp. 2338-2349, May 2007.

Wey, C.; Lin, S. & Tang, W. (2007). Efficient Memory-Based FFT Processors For OFDM Applications. *IEEE International Conference on Electro- Information Technology,* pp.345 - 350, May 2007.

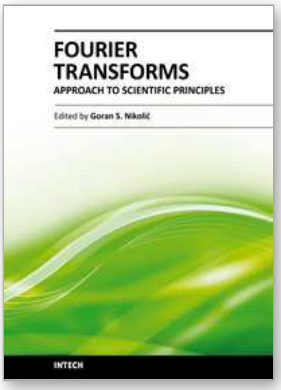
Xiao, X.; Oruklu, E. & Saniie, J. (2008). An Efficient FFT Engine with Reduced Addressing Logic. *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 55, no. 11, pp.1149-1153, November 2008.

Xiao, X.; Oruklu, E. & Saniie, J. (2009). Fast Memory Addressing Scheme for Radix-4 FFT Implementation. *IEEE International Conference on Electro/Information Technology, EIT 2009*, pp. 437-440, June 2009.

Xiao, X.; Oruklu, E. & Saniie, J. (2010). Reduced Memory Architecture for CORDIC-based FFT. *IEEE International Symposium on Circuits and Systems (ISCAS),* June 2010.

Yang, L.; Zhang, K.; Liu, H.; Huang, J. & Huang, S. (2006). An Efficient Locally Pipelined FFT Processor. *IEEE Transactions on Circuits and Systems II, Exp. Briefs,* vol. 53, issue 7, pp. 585-589, July 2006.

Zainal, M. S.; Yoshizawa, S. & Miyanaga, Y. (2009). Low Power FFT Design for Wireless Communication Systems. *International Symposium on Intelligent Signal Processing and Communications Systems ISPACS 2008*, pp. 1-4, February 2009.



Fourier Transforms - Approach to Scientific Principles

Edited by Prof. Goran Nikolic

ISBN 978-953-307-231-9
Hard cover, 468 pages
Publisher InTech
Published online 11, April, 2011
Published in print edition April, 2011

This book aims to provide information about Fourier transform to those needing to use infrared spectroscopy, by explaining the fundamental aspects of the Fourier transform, and techniques for analyzing infrared data obtained for a wide number of materials. It summarizes the theory, instrumentation, methodology, techniques and application of FTIR spectroscopy, and improves the performance and quality of FTIR spectrophotometers.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Erdal Oruklu, Jafar Saniie and Xin Xiao (2011). Reduced Logic and Low-Power FFT Architectures for Embedded Systems, Fourier Transforms - Approach to Scientific Principles, Prof. Goran Nikolic (Ed.), ISBN: 978-953-307-231-9, InTech, Available from: http://www.intechopen.com/books/fourier-transforms-approach-to-scientific-principles/reduced-logic-and-low-power-fft-architectures-for-embedded-systems

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821