Power and Area Minimization of Reconfigurable FFT Processors: A 3GPP-LTE Example

Chia-Hsiang Yang, *Member, IEEE*, Tsung-Han Yu, *Student Member, IEEE*, and Dejan Marković, *Member, IEEE*

***Abstract*—This paper presents a design methodology for power and area minimization of flexible FFT processors. The methodology is based on the power-area tradeoff space obtained by adjusting algorithm, architecture, and circuit variables. Radix factorization is the main technique for achieving high energy efficiency with flexibility, followed by architecture parallelism and delay line circuits. The flexibility is provided by reconfigurable processing units that support radix-2/4/8/16 factorizations. As a proof of concept, a 128- to 2048-point FFT processor for 3GPP-LTE standard has been implemented in a 65-nm CMOS process. The processor designed for minimum power-area product is integrated in 1.25×1.1 mm^2 and dissipates 4.05 mW at 0.45 V for the 20 MHz LTE bandwidth. The energy dissipation ranging from 2.5 to 103.7 nJ/FFT for 128 to 2048 points makes it the lowest energy flexible FFT.**

***Index Terms*—Fast Fourier transform (FFT), CMOS digital integrated circuits, reconfigurable architecture, power and area minimization.**

I. Introduction

FAST FOURIER TRANSFORM (FFT) is an important digital signal processing (DSP) technique to analyze the phase and frequency components of a time-domain signal. FFT processors have been widely used in various applications such as communications, image, and biomedical signal processing. For example, high-performance and low-power FFT processing is indispensable in orthogonal frequency-division multiplexing (OFDM) systems. Applications are now changing towards increasing diversity of features and standards that need to be supported on a single device. This change in applications greatly emphasizes the need for flexibility. At the same time, maintaining high levels of energy efficiency is of crucial importance for mobile terminals. We therefore investigate energy efficiency of *flexible* FFTs that be configured to a variety of FFT sizes and sampling rates.

FFT architectures have been extensively studied. Traditional architectures include memory-based [1], pipelined [2], array [3], and cached-memory architecture [4]. Advanced circuit techniques such as design for minimum-energy operation [5], [8], dynamic voltage and frequency scaling (DVFS) [6], asynchronous logic [7], and deep pipelining [8] have also been used to enhance energy efficiency of FFT processors. The benefits of radix factorization for reduced hardware cost of custom FFTs have been largely unexplored. A ring-structured multiprocessor architecture was proposed in [9] to utilize mixed radix. A mixed-radix (radix 4 and radix 8) multipath delay feedback (MRMDF) architecture and indexed-scaling pipelined architecture were introduced in [10] and [11], respectively. A variable-length FFT processor that integrates two radix-2 stages and three radix-2^3 stages for FFT sizes 512, 1024 and 2048 was proposed in [12]. Prior work optimized various aspects of the FFT processors, but explored limited set of parameters. A systematic design methodology that integrates parallelism, radix factorization, and memory parameters for flexible FFTs has not been thoroughly investigated.

We propose an FFT design methodology that jointly considers algorithm, architecture, and circuit parameters. We contribute with insights on how to use FFT radix structure for highly energy- and area-efficient implementations. Hundreds of architectures for 128- to 2048-point FFT exist by varying the degree of parallelism and radix factorization, as will be explained in this paper. Apart from parallelism and radix, delay buffers need to be efficiently implemented. Memory size partition and memory elements for delay lines of different lengths are evaluated. Our approach provides a cross-layered FFT design methodology to jointly optimize above parameters. For illustration, we will design for minimum power-area product (PAP). We will show an FFT processor that achieves the lowest energy per FFT operation, comparable area and much fewer processing cycles as compared to prior work.

This paper is organized as follows. Section II gives a brief review of FFT operation, FFT radix structure and possible hardware architectures. Estimation of power and area and the use of FFT design techniques are discussed in Section III. As a proof of concept, Section IV presents a chip implementation of 3GPP-LTE compliant FFT (128 to 2048 points). Chip measurements indicate over a $2\times$ better energy efficiency than prior work. Section V concludes the paper.

II. FFT Algorithms and Architectures

The N-point discrete Fourier transform (DFT) of an input sequence is defined as

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk} \tag{1}$$

where $k = 0, 1, 2, \ldots, N-1$ and $W_N = e^{-j2\pi/N}$ is known as the twiddle factor. Direct implementation of (1) requires N^2

Manuscript received August 17, 2011; revised October 25, 2011; accepted November 04, 2011. Date of publication December 23, 2011; date of current version February 23, 2012. This paper was approved by Associate Editor Stefan Rusu.

C.-H. Yang is with the Electronics Engineering Department, National Chiao Tung University, Hsinchu 30010, Taiwan (e-mail: chy@nctu.edu.tw).

T.-H. Yu and D. Marković are with the Department of Electrical Engineering, University of California at Los Angeles, Los Angeles, CA 90095 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSSC.2011.2176163

complex multiplications and $N\,(N-1)$ complex additions. FFT algorithms efficiently compute the DFT by decomposing the input sequence into smaller-size DFTs [13]. The Cooley-Tukey algorithm [14] leverages the divide-and-conquer technique to recursively partition a DFT of size $N = M \times L$ into many smaller DFTs of sizes M and L. For $N = M \times L$ and $k = Mp + q$, where $0 \le p \le L-1$ and $0 \le q \le M-1$, the N-point FFT can be represented in a two-dimensional form as in (2):

$$X[k] = \underbrace{\sum_{l=0}^{L-1} \left(\underbrace{\sum_{m=0}^{M-1} x[l,m] W_M^{mq}}_{M-pt\,\mathrm{DFT}} \underbrace{W_N^{lq}}_{\text{twiddle factor}} \right) W_L^{lp}}_{L-pt\,\mathrm{DFT}}. \tag{2}$$

The calculation of $X[k]$ is hence decomposed into three steps: 1) compute M-point DFT, 2) multiply the outputs by twiddle factors, and 3) compute L-point DFT. Due to decomposition, the number of complex multiplications is reduced from N^2 to $N(M+L+1)$. The number of additions is reduced from $N(N-1)$ to $N(M+L-2)$.

The most common FFT algorithm is the radix-2 Cooley-Tukey algorithm, which recursively splits an N-point FFT into two segments of size $N/2$. Using the symmetry ($W_N^{k+N/2} = -W_N^k$) and periodicity ($W_N^{k+N} = W_N^k$) of the twiddle factors, DFT can be computed efficiently. The split-radix FFT algorithm [15] mixes radix-2 and radix-4 factorizations, yielding an algorithm with fewer additions and multiplications than the radix-2 FFT. However, the split-radix algorithm has an irregular signal-flow structure and it can only be applied when N is a multiple of 4. For the case where M and L are relatively prime, the DFT can be computed more efficiently using the prime-factor algorithm (PFA) [16]. The PFA decomposes M and L into smaller recursive sections without using the twiddle factors. The computation of the PFA has fewer multiplications but a complex re-indexing is required. For a power-of-prime size, the Winograd Fourier Transform Algorithm (WFTA) [17] performs the decomposition efficiently using cyclic-convolution techniques. Since the cyclic-convolution operations often have coefficients of 1, 0, or -1, the WFTA requires fewer multiplications at the cost of more additions. The disadvantage of PFA and WFTA is that they only work for specific FFT sizes.

Inspired by the split-radix algorithm, various radix factorizations have been proposed to implement efficient FFTs. For example, a split-radix FFT with minimal multiplicative complexity is proposed in [18]. High-speed low-power hardware implementations are presented in [19], [20]. As mentioned before, prior work utilized radix factorization with limited success. The hardware complexity is minimized by only reducing the number of complex (full) multiplications for various radix FFTs. Also, the use of high-radix is commonly believed to be more area-efficient than low-radix algorithms [6], [10] due to the use of fewer complex multiplications. The complexity analysis that considers only the asymptotic trend $O(N \log_r N)$ may lead to an inefficient architecture for dedicated designs that consider the cost of constant multipliers and adders, and also account for the signal wordlengths. This work presents an optimal-radix design methodology by considering the cost of all arithmetic elements. We start from radix-2 architecture since mixed-radix structures are built using radix-2 butterflies.

A. Radix-2-Butterfly Based Architecture

The radix-2 FFT algorithm decomposes the N-point DFT into 2-point DFTs recursively. It requires $(N/2)\log_2 N$ multiplications and $N \log_2 N$ additions, leading to a significant saving for a large N compared to direct-mapped DFT. The atomic module of radix-2 FFT is the butterfly operation shown in Fig. 1(a). Decimation-in-time (DIT) and decimation-in-frequency (DIF) butterfly operations have the same structure and only differ in the placement of the W_N multiplier. In this work, DIF structure is adopted, but the proposed design methodology is also applicable for DIT. Besides radix-2, an N-point DFT can also be decomposed into r-point DFTs. This is known as radix-r FFT algorithm. There are $\log_r N$ stages and N/r butterflies per stage. Each butterfly requires $r-1$ complex multiplications. When $r = 2^k$, radix-r butterfly can be further decomposed by cascading k radix-2 stages, known as radix-2^k algorithm [21]. The total number of complex multiplications of radix-r FFT is $[N(r-1)/r][(\log_r N) - 1]$ considering that the twiddle factors of the last stage are always equal to unity. As we will describe later, all multipliers internal to the radix-r butterfly ($r = 2^k$) can be implemented as *constant* multipliers in order to reduce area and power. The number of complex additions is $N \log_2 N$ regardless of r.

Due to its regular structure, the FFT can be realized using radix-r ($r = 2^k$) butterflies, delay lines, and complex multipliers. Radix-r can be realized using several architectures considering the tradeoff between silicon area and execution time. The memory-based time-multiplexed architecture [1] has only one radix-r butterfly and $r-1$ complex multipliers. The inputs, outputs, and interim results are read from and written back to memory for complete FFT operation. This architecture has the lowest area and longest execution time. Another extreme is the direct-mapped fully parallel architecture, which requires $[N(r-1)/r][(\log_r N) - 1]$ complex multipliers. Between these two extreme cases, there is a pipelined architecture which provides a balanced tradeoff between area and execution time. Two major types of pipelined architectures are multi-path delay commutator (MDC) and single-path delay feedback (SDF). For higher-radix algorithms, the SDF architecture is preferred since it requires less memory and fewer complex multipliers than the MDC architecture [21].

The radix-2 SDF architecture for 2^k-point FFT is shown in Fig. 1(b). In each stage, the required $N/2$ butterfly operations are time-multiplexed onto one butterfly operator. Delay buffers are used to support the time-multiplexing. The inter-stage multipliers are used to multiply stage outputs by twiddle factors W_N^{kn}. The radix-2 SDF architecture requires $\log_2 N - 1$ inter-stage complex multipliers, $2 \log_2 N$ complex adders, and $N-1$ delay buffers.

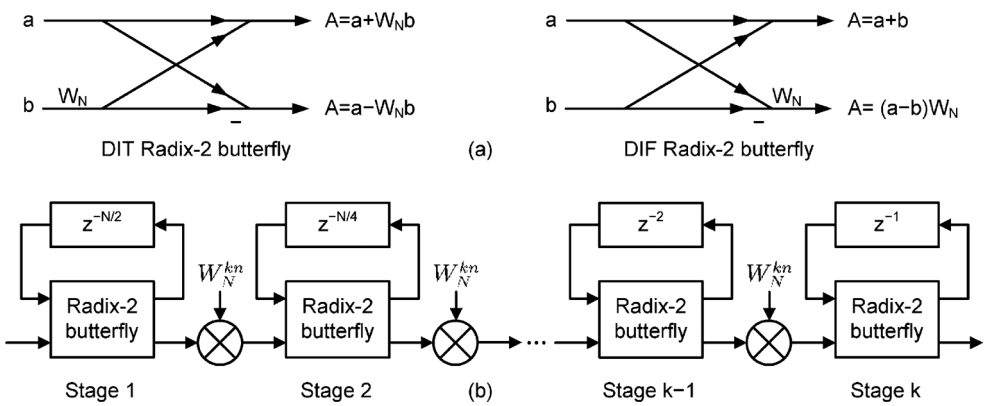


Fig. 1. (a) Signal-flow graph of radix-2 butterfly for decimation in time (DIT) and decimation in frequency (DIF). (b) Radix-2^k single-path delay feedback (SDF) architecture. Output of stage k does not need twiddle-factor multiplication.

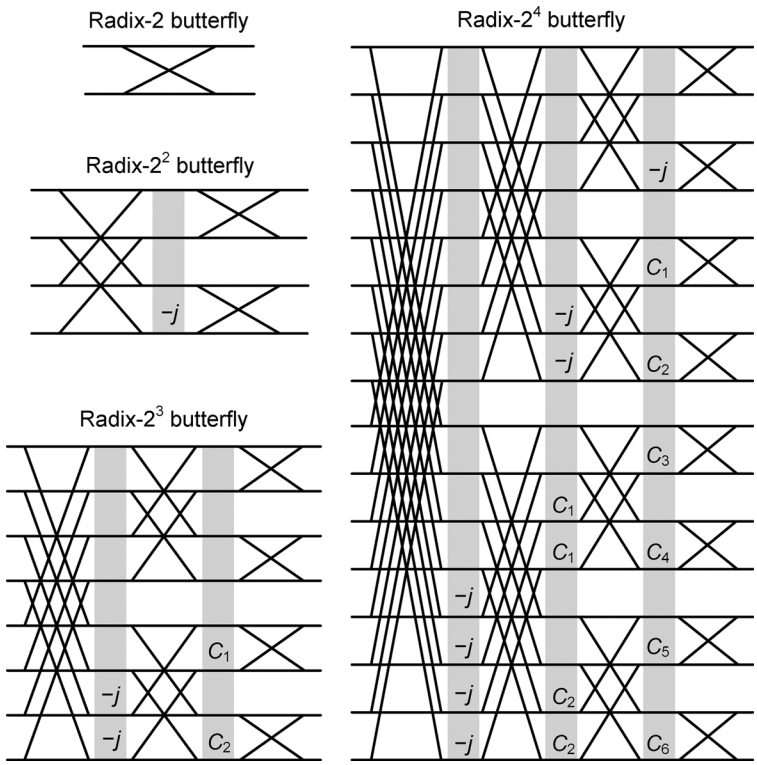


Fig. 2. Radix-$2/2^2/2^3/2^4$ butterfly operations. Shaded stripes indicate constant multipliers.

B. Reconfigurable Architecture

Based on the pipelined SDF architecture, a reconfigurable FFT architecture can be implemented by cascading several radix-2^k stages in order to accommodate different FFT sizes. The signal-flow graphs for radix-2 to radix-2^4 butterflies are shown in Fig. 2. The minus signs in the butterfly modules are omitted for simplicity. The highlighted inter-stage multiplications are implemented as constant (complex) multipliers as opposed to using full (complex) multipliers. The radix-2^k can be realized by cascading several atomic processing units (PUs) as in Table I. The PUs are shown in Fig. 3. Looking at the cost of the constant multipliers (shown in brackets in Fig. 3) in terms of the number of equivalent adders, we discovered that radices beyond 2^4 are impractical, because the increasing number and complexity of required constant multipliers makes them no longer advantageous over full multipliers. In addition, full multipliers need extra ROMs to store the coefficients as opposed to locally computed coefficients of constant multipliers. Therefore, radix-2 to radix-2^4 is the proper level of granularity for mixed-radix FFT implementations.

As shown in Fig. 3(a), each PU contains a basic butterfly module and a set of *constant* multipliers. The butterfly module is initialized to the data-switch mode until the delay buffers are loaded by the valid inputs and then switched to the butterfly mode for FFT operation. The required constant multipliers for PUs 1–4 are shown in Fig. 3(b). Only half the twiddle factors (dark-filled circles) are generated in the PUs, the other half (gray-filled circles) are created using the symmetry property. The PUs with lower index can be deduced from the PUs with

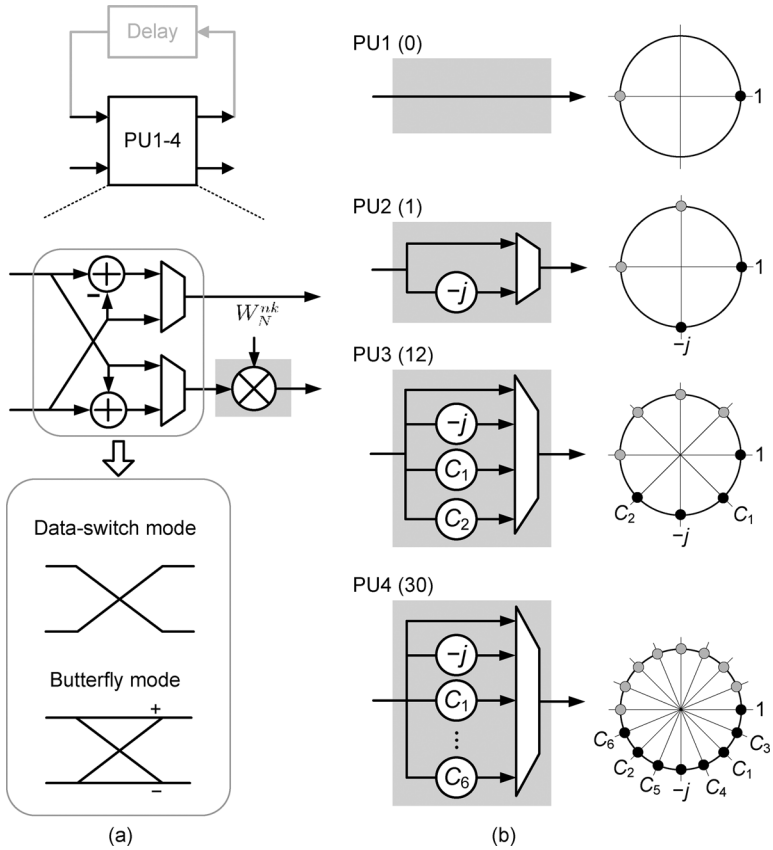


Fig. 3. Reconfigurable processing units (PUs) support radix-2 to radix-2^4 factorizations. (a) PU1-4 architecture. Each PU can operate in a data-switch or a butterfly mode. (b) The constant multipliers associated with PU1-4. The numbers in brackets next to each PU indicate relative cost (in terms of the number of real-valued adders) of its constant multiplier.

TABLE I
COST OF RADIX IMPLEMENTATIONS WITH RECONFIGURABLE PROCESSING UNITS (PUS)

Type	Implementation	Butterfly adders	Constant multipliers (equivalent adders)
Radix-2	PU1	4	0
Radix-2^2	PU2+PU1	8	1
Radix-2^3	PU2+PU3+PU1	12	13
Radix-2^4	PU2+PU3+PU4+PU1	16	43

higher index. For example, PU4 can serve as PU3, PU2 or PU1. This "back-compatibility" is useful for reconfigurable designs. All the intra-stage multipliers inside the PUs for a 2^k-point FFT $(k \leq 4)$ are constant multipliers. Full multipliers are only used for the inter-stage twiddle factors. Since the inter-stage full multipliers cost more than the intra-stage constant multipliers, radix factorization should minimize the number of full multipliers.

III. POWER AND AREA MINIMIZATION

We propose a systematic methodology to explore FFT power-area tradeoff. FFT realizations are systematically explored in three steps. First, architecture parallelism combined with FFT decomposition is used to explore the power-area space through voltage scaling. Next, radix factorization is explored for a given FFT size. The third step consists of memory partition, selection of memory cells and logic operators. To support multiple FFT sizes, optimal mapping of processing units that considers all required FFT sizes is performed.

A. Parallel Architecture With FFT Decomposition

Parallelism is an effective technique to increase throughput [10] or to reduce power consumption [22], [23] of an FFT processor. For a fixed throughput, scaled voltage and a lower clock frequency improve the energy efficiency of a parallel architecture. Since time-multiplexing is inherently applied to SDF architecture, parallelism is used to improve its energy efficiency and

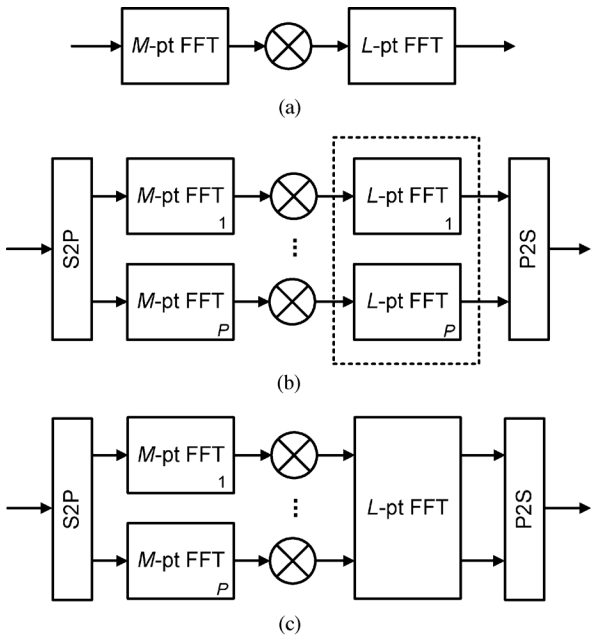


Fig. 4. (a) Reference N-point ($N = M \times L$) FFT architecture. (b) A P-way parallel architecture ($P \neq L$) requires P replicas of the L-point FFT. (c) When $P = L$, the L-point FFT can be shared across the P streams, which leads to a reduced hardware area.

adjust the design point in the area-energy-delay plane [24]. An area-efficient parallel architecture is possible by leveraging FFT decomposition. As shown in Fig. 4(a), an N-point FFT is decomposed into M-point FFT and L-point FFT. Straightforward P-way parallel architecture, Fig. 4(b), requires P M-point FFTs and P L-point FFTs, increasing the area by a factor of P (neglecting the overhead of the serial-to-parallel and parallel-to-serial blocks). When $P = L$, the L single-input SDF FFTs can be combined into a single L-input *parallel* FFT, as in Fig. 4(c), to reduce area. This architecture simplification is possible since the M-point FFTs can be computed first and combined into the L-point ($L = N/M$) output stage to compute an N-point FFT. The architecture in Fig. 4(c) contains

$$L[(\log_2 M) - 1] + L + L[(\log_2 L) - 1]/2 = L[(\log_2 NM) - 1]/2 \text{ complex multipliers}$$

and

$$L(2\log_2 M) + L\log_2 L = L(\log_2 NM) \text{ complex adders.}$$

Besides the reduced arithmetic complexity, effective implementation of memory is also required. $L - 1$ delay buffers used for scheduling of the L-point FFT in Fig. 4(b) can be removed since in Fig. 4(c) the inputs of L paths are available and aligned in time. The total number of delay buffers is reduced from $L(N - 1)$ to $L(N - 1) - L(L - 1) = L(N - L)$.

Finally, the hardware complexity of the M-point FFT can be reduced. Combining L streams in Fig. 4(b) necessitates zero-padding of data and the length of delay buffers in the M-point FFTs to be a multiple of L. When $P = L$, as in Fig. 4(c), the length-L delay buffers in each of the M-point FFTs can be replaced by length-1 buffers at L times lower rate to match the delay. This reduces the number of delay buffers in Fig. 4(c) from $L(N - L)$ to $L(N - L)/L = N - L = L(M - 1)$, which is the minimal number of delay buffers required for an L-path M-point FFT.

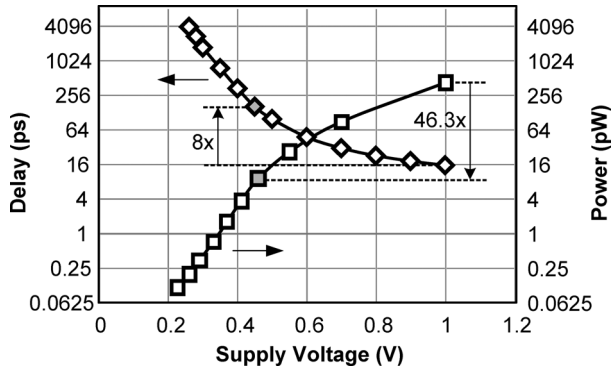


Fig. 5. Simulated delay (left y-axis) and power (right y-axis) vs. supply voltage for a FO-4 inverter in 65 nm CMOS is used to predict the speed-power tradeoff in FFT architectures. The reported power assumes maximum clock rate for a given voltage.

B. Estimation of Area and Power

To evaluate many architecture and circuit realizations, high-level area and power models of the FFT building blocks are developed. Area is estimated as the total relative area of multipliers, adders, and memory. The area of the L-path SDF architecture in Fig. 4(c) is estimated as

$$\text{Area} = A_{\text{mult}} \cdot L[(\log_2 NM) - 1]/2 + A_{\text{add}} \cdot L(\log_2 NM) + A_{\text{mem}} \cdot L(M - 1) \quad (3)$$

where A_{mult}, A_{add}, and A_{mem} represent the area of multiplier, adder, and delay buffer, respectively. These three parameters can be estimated from synthesis. Without loss of generality, we use 12-bit arithmetic and DFF-based delay buffers in our analysis.

Power is estimated from the total area, switching activity, operating frequency and voltage. It considers both switching (P_{sw}) and leakage (P_{leak}) components. Fig. 5 shows FO4 inverter delay and power vs. supply voltage in the typical-typical (TT) corner in a standard-V_T 65-nm CMOS technology. The delay-voltage curve is used to predict the amount of voltage scaling for varying performance requirements. For example, if a 20 MHz design is required to run at 0.45 V, it has to be synthesized for 160 MHz at 1 V, as shown in Fig. 5. A ratio $P_{sw}/P_{\text{leak}} = 13.5$ is estimated from a FO4 inverter chain at 0.45 V and 20 MHz. By lowering V_{DD} from 1 V to 0.45 V and frequency from 160 MHz to 20 MHz, the power is reduced by 46.3×.

Since circuits operate at the same voltage and frequency the switching power P_{sw} of the processing units can be estimated as the product of per-cycle utilization rate α and active area A_{total} as given by

$$P_{sw} = \alpha C V^2 f \propto \alpha A_{\text{total}} \quad (4)$$

where A_{total} is the area cost. Inactive blocks are disabled by using clock gating or wired-AND circuits. The two complex

Factor	CSD realization	Adders
0.7071	$2^{-1}+2^{-3}+2^{-4}+2^{-6}+2^{-8}$	4
0.9238	$2^{0}-2^{-3}+2^{-5}+2^{-6}+2^{-9}$	4
0.3828	$2^{-2}+2^{-3}+2^{-7}$	2

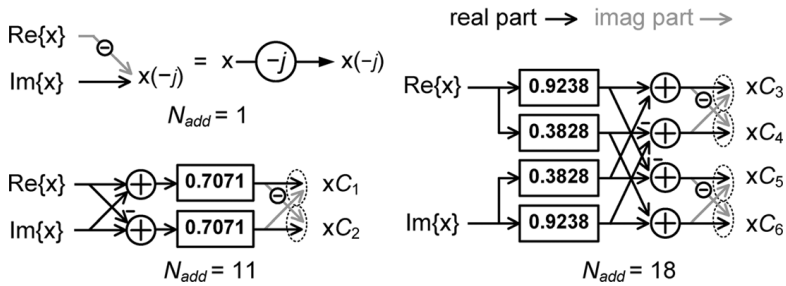


Fig. 6. Constant multipliers use canonic signed digit (CSD) representation for area-efficient implementation. Only three generated values (top table) are required to implement all constant multipliers required by the PUs from Fig. 3, resulting in a total of 30 real-valued adders.

adders in the butterfly module of PUs (Fig. 3) have utilization rate of 1/2 because the periods for the data-switch and butterfly modes are the same, resulting in two real additions per cycle, on average. According to the signal-flow graph in Fig. 2, the utilization rate of each constant multiplier is 1/4, 1/8, and 1/16 per clock cycle for PU2, PU3, and PU4, respectively. The utilization rate of the inter-stage full multipliers depends on the preceding PUs and it is accounted for in our analysis.

C. Mixed-Radix Implementation

As mentioned in Section II.A, higher-radix structures can be made more area efficient by judiciously replacing full multipliers with constant multipliers. These constant multipliers are implemented using canonic signed digit (CSD) representation [25], as shown in Fig. 6. A 10-bit twiddle factors are assumed. Each of the constant factors requires no more than 4 additions, which leads to a large area reduction. The area of constant multipliers is minimized using the symmetry property of twiddle factors and sharing of common sub-expressions. Only 30 adders are need for all twiddle-factor multiplications shown in Fig. 3. The cost of implementing radix-2 to radix-2^4 operations, which will be used in radix factorizations, is summarized in Table I.

To minimize the hardware cost of inter-stage full multipliers (between radix-2^k blocks), full complex multipliers are implemented by using 3 real multipliers and 5 real adders (rather than 4 real multipliers and 2 adders) [26] as follows:

$$(a+jb)(c+jd) = [c(a-b)+b(c-d)] + j[d(a+b)+b(c-d)].$$

The equivalent number of adders in a full real multiplier is estimated as the wordlength of twiddle factors. Since the number of butterfly adders is the same in all radix structures, only the equivalent adders for complex multiplications are used for quick comparison of different FFT architectures. The number of equivalent adders for 4- to 16-point FFTs with twiddle factor (TF) wordlengths 8b–16b is summarized in Table II. For example, a 16-point FFT can be implemented using two radix-2^2 stages, one adder per stage (Table I), along with an inter-stage full multiplier (using three multipliers and five adders). For 8-bit twiddle factors, the number of equivalent real-valued adders is thus $2+(3\times 8+5)=31$. FFTs up to 16 points are considered, because this is the adequate level of granularity for radix factorization, as discussed earlier. The choice of radix depends on the FFT size and twiddle-factor wordlength. Radix structures with minimum area are highlighted. Radix-2^2 architecture needs only one adder for 4-point FFT, but the radix-2 architecture needs 29–53 adders due to the use of full multipliers. Using higher radix makes sense here, because radix-2^2 exactly matches 4 FFT points. As the FFT size increases, a larger number of constant multipliers required to support higher radix diminishes the area advantage over full multipliers. For 16 points, the radix-2^4 architecture needs 37–49 adders while the radix-2 architecture needs 87–159 adders (a 2.53–3.24× larger area). The area saving for the 16-point FFT is not as large as that for the 4-point FFT. The wordlength of twiddle factors also affects the area. Radix-2^2 architecture is the most area efficient for twiddle factors below 14 bits, else radix-2^4 should be used. In summary, radix-2 is the least area efficient. Higher radix (up to 2^4) is generally better unless wordlength of twiddle factors is small (less than 14 bits).

Using the area and power models from Section III.B, we can evaluate area-power tradeoff for various radix factorizations and degrees of parallelism. As an example, Fig. 7 shows the area-power tradeoff for 2048-point FFT implemented using architecture from Fig. 4(c). Partitioning with $M=256$ and $L=P=8$ gives the lowest PAP for 2048 points. It achieves a $5\times$ lower PAP than $M=2048, L=1$ design. Next, since $M>2^4$, we have to examine radix factorization of 256-point FFT for further area and power reduction. Radix structure of 256-point FFT is shown in Fig. 8. The architecture with eight radix-2 stages (A1) occupies the largest area, as expected from prior analysis. Architecture A11 (consisting of one radix-2^2 stage and two radix- 2^3 stages) is the most area efficient. The architecture with highest radix, i.e., cascading two radix-2^4 stages, (A14) has the lowest PAP. A $6.7\times$ and $1.72\times$ PAP reduction are achieved from radix-2 (A1) to radix-2^4 (A14) and from radix-2^2 (A10) to radix-2^4 (A14), respectively, due to radix factorization. Taking

TABLE II
NUMBER OF EQUIVALENT ADDERS REQUIRED TO IMPLEMENT FFT

FFT size	4 points					8 points					16 points				
TF bits	8	10	12	14	16	8	10	12	14	16	8	10	12	14	16
Radix-2	29	35	41	47	53	58	70	82	94	106	87	105	123	141	159
Radix-2^2	1	1	1	1	1	-	-	-	-	-	31	37	43	49	55
Radix-2^3	-	-	-	-	-	11	13	13	13	15	-	-	-	-	-
Radix-2^4	-	-	-	-	-	-	-	-	-	-	37	43	43	43	49

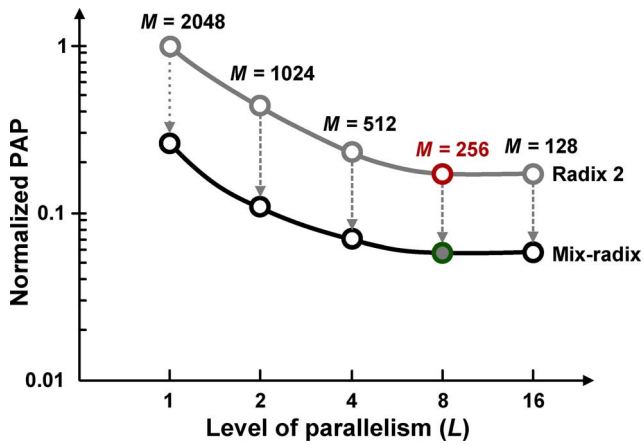
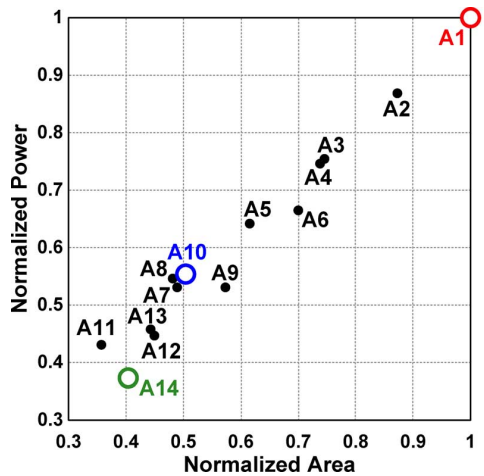


Fig. 7. Power-area product (PAP) for varying levels of parallelism for the architecture in Fig. 4(c). Radix factorization (the mix-radix curve) and parallelism have comparable impact on PAP.



Arch	Radix stages 2	2^2	2^3	2^4
A1	8			
A2	6	1		
A3	4	2		
A4	5		1	
A5	3	1	1	
A6	4			1
A7	1	2	1	
A8	2		2	
A9	2	1		1
A10		4		
A11		1	2	
A12		2		1
A13	1		1	1
A14				2

Fig. 8. Power and area (normalized to radix-2 architecture) for feasible radix factorizations of 256-point FFT. Architecture A14 has the lowest power-area product.

delay buffers into consideration, as will be discussed next, a $3.4\times$ reduction in power-area product is achieved from architecture A1 to A14. An overall $17\times$ PAP reduction compared to the direct mapped architecture ($M = 2048$, $L = 1$) is achieved through FFT decomposition $(5\times)$ and radix factorization $(3.4\times)$.

D. Delay-Buffer Architecture and Memory Bit-Cell

After FFT decomposition and radix factorization, the next step is the power-area optimization of delay buffers. Three options are considered for delay buffer implementation [21]: 1) DFF-based shift register, 2) register file (RF)-based, and 3) SRAM-based delay buffer. Architecture and memory cells for various delay lengths have to be evaluated. To illustrate our methodology and support the example in Section IV, we assume delay lengths up to 1024.

We start by comparing RF-based and SRAM-based delay buffers, which have different memory cells and peripheral circuits. A straightforward implementation is a dual-port (DP) RF/SRAM-based architecture, as shown in Fig. 9(a). If the RF/SRAM memory size is N, the output of the dual-port RF/SRAM is read after $N - 1$ cycles, so an overall N-cycle delay is achieved. Area and power of RF and SRAM designs are evaluated using commercial memory compilers for the target 65-nm technology. For 32-bit (complex-valued input) delay buffers, RF-based design is superior since it consumes 41–49% of power with a 66–82% silicon area compared to the SRAM-based counterpart.

Next, we compare RF-based and DFF-based designs. RF-based delay buffers are more area efficient due to the compact bitcell structure but the peripheral circuits contribute a considerable area overhead for small-size memories. However, RF-based designs cannot be operated at very low voltage due to cell read margin and sense amplifier operation. Since DFF-based designs can operate at a low voltage, they are more energy efficient despite the area disadvantage. In addition to voltage scaling, the power consumption of the DFF-based buffers can be reduced using a pointer-based architecture, Fig. 9(b). Instead of shifting the data, a shift-delay-line chooses the corresponding DFF to read and write. The remaining DFFs are clock gated when they are not activated by the shift-delay-line, eliminating significant dynamic power. To evaluate the tradeoff between RF-based and DFF-based delay buffers, power and area for the delay lines of interest are shown in Fig. 10. For the delay buffers longer than 256, the RF-based design at 0.9 V has a lower PAP compared to the DFF-based

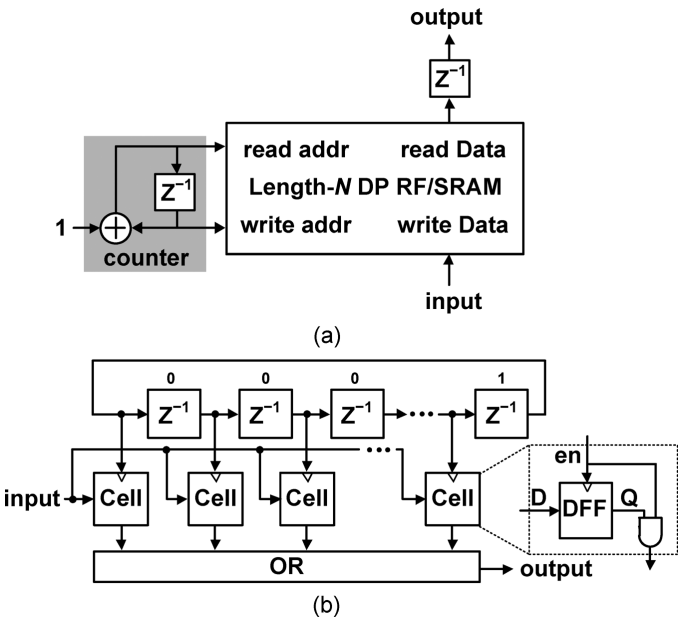


Fig. 9. (a) Dual-port RF-based delay buffer and (b) pointer-based DFF-based delay buffer.

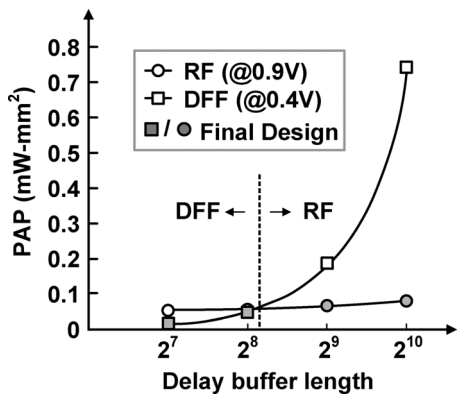


Fig. 10. Power-area product of delay buffers: D flip-flops (DFF) are used for delay lengths of 128 and 256, register files (RF) are used for delay lengths of 512 and 1024.

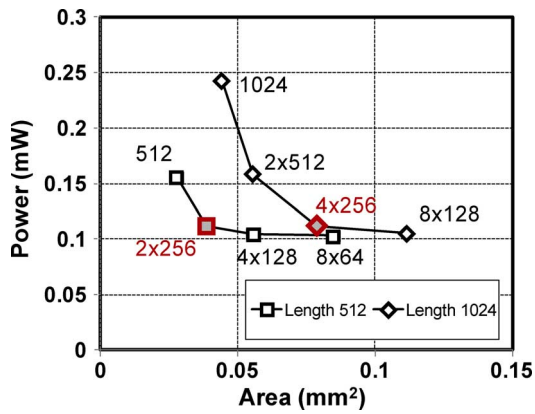


Fig. 11. Power-area tradeoff for feasible memory partitions. Partitions 2×256 for length 512 and 4×256 for length 1024 delay buffers yield minimum power-area product.

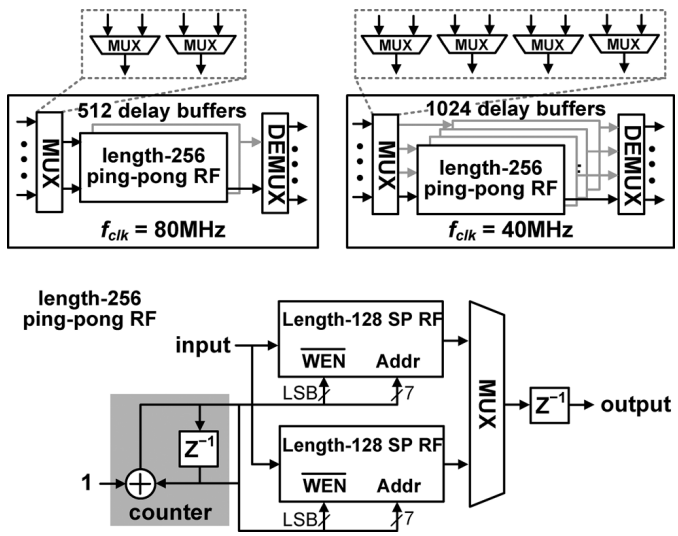


Fig. 12. Memory architecture for length 512 and 1024 delay buffers (top), length-256 ping-pong register file (bottom).

design operated at 0.4 V, for a 20 MHz sampling rate. Therefore, the delay buffers of length 512 and 1024 are implemented using RFs.

The power of RF is further reduced via memory partitioning. Fig. 11 shows the possible memory partition schemes and the optimal partition for lengths 512 and 1024. The architecture with lower power is chosen if multiple designs have the same PAP. The designs with two and four 256×32b RF banks are chosen for the delay buffers of length 512 and 1024, respectively.

Finally, one length-256 dual-port (DP) RF block can be replaced with two length-128 single-port (SP) RF blocks. This reduced PAP by $1.59\times$ and $1.72\times$, respectively, for delay buffers of length 512 and 1024. The final memory structure for the delay buffers is shown in Fig. 12. Eight 128×32b and four 128×32b RFs are used to implement length 1024 and 512 delay buffers. Input/output of two SP RF modules are written/read alternatively to create adequate delays. For the remaining delay buffers, DFF-based registers with aggressive voltage scaling are used.

E. Area-Efficient Twiddle-Factor Generator

Twiddle factors are generated in an area-efficient way by trigonometric approximation instead of fetching coefficients from ROMs. The trigonometric approximation is realized by a first-order linear approximation [27], as follows. First, by means of symmetric sine/cosine values, angles $[0, \pi/4)$ can be used to construct the whole sine/cosine space. Second, sine values can be approximated using piecewise-linear approximation, as given by

$$\sin(2\pi\alpha) \approx \begin{cases} 25/4\alpha & 0 \le \alpha < 1/16 \\ 41\alpha/8 + 9/128 & 1/16 \le \alpha \le 1/8 \end{cases}.$$

Third, $\cos(2\pi\alpha) = \sqrt{1 - \sin^2(2\pi\alpha)}$ for $0 \le \sin(2\pi\alpha) \le 0.7071$ is approximated by another linear approximation as shown in the equation at the bottom of

TABLE III
3GPP-LTE DESIGN SPECIFICATIONS

Parameter	Configuration					
FFT size	128	256	512	1024	1536	2048
BW (MHz)	1.25	2.5	5	10	15	20
Parallelism	8	8	8	8	6	8
Factorization	16	4x8	8x8	8x16	16x16	16x16

this page. Based on these approximations, only *shift* and *add* operations are needed to generate the twiddle factors (TFs). The overall number of intra-stage TF generators is $(L-1)+(F-1)$, where F is the number of PU stages of in the M-point FFT block. Since TFs for the multipliers in the same stage of the L M-point FFT blocks can be shared, only $F-1$ unique TF generators are necessary. In addition, there are $L-1$ TF generators for the multipliers between M-point and L-point FFTs.

IV. CHIP IMPLEMENTATION

A reconfigurable FFT processor for the OFDM-based 3GPP long term evolution (LTE) standard is designed and implemented to demonstrate the proposed design methodology [28]. The processor supports complex FFTs with 128 to 2048 points for the bandwidths from 1.25 to 20 MHz [29], as shown in Table III.

A. Architecture and Chip Implementation

Starting from FFT decomposition and architecture parallelism, the maximum-size 2048-point FFT is decomposed into $M=256$, $L=8$ to achieve minimum PAP for the bandwidth of 20 MHz. The 8-path 256-point SDF FFT architecture is shown in Fig. 13. It can support 16 to 256 points by reconfiguring the data-path inter-connection between the PUs. To support 1536 points, a 6-point FFT module is constructed by sharing hardware resources with the 8-point FFT. The optimal PU configuration for each FFT size is chosen from minimum PAP. The optimal radix factorizations for supported FFT sizes are listed in Table III. An overall $17\times$ PAP reduction can be achieved for 2048 points through FFT decomposition and radix factorization compared to the baseline radix-2 architecture.

The overall memory size for the register files is $(1024+512)\times 32\text{b}=48$ kb. The feedback delays (64/32, 32/16, 16/8) are reconfigured by switching between different delay-buffer modules. The area difference between 128 and 2048 points is less than $4\times$ since half of the 256-pt module is used to support 16 points and they share the same 8-point FFT in the second stage. Clock gating disables unused PUs and delay buffers for energy saving. The twiddle factors for 1536 points are calculated using the same TF generator as for FFTs with a power-of-2 size.

Fig. 14 is the chip micrograph of an OFDM-based MIMO decoder with highlight on the FFT processor. Chip features are summarized in Table IV. The chip is fabricated in a standard-V_T 65-nm CMOS process. The total chip area with I/O pads is 6.86 mm^2. The area of the reconfigurable FFT processor (including RF memory bank) is 1.375 mm^2. The core supply voltage is tunable between 0.2 V and 1.0 V, and the I/O pads are supplied from 1.0/2.5 V. Cross-coupled level shifters are inserted between voltage domains and the core-I/O boundary. The supply voltage of the FFT core and RF bank is 0.45 V and 0.9 V, respectively, for the 20 MHz bandwidth. To support up to eight antennas for the LTE-Advanced standard, the maximum I/O frequency is 160 MHz. The input is de-multiplexed into the 8-path FFT processor, with core frequency of 20 MHz, and multiplexed back for output. Clock frequencies of 40 MHz and 80 MHz are used for memory access, as shown in Fig. 12. The numerical performance of the FFT processor for input SNR of 25 dB, Fig. 15, justifies the choice of the 12-bit input wordlength since the output SNR flattens beyond 12 bits.

B. Test Setup

Chip testing is performed with the aid of an IBOB FPGA board [30] for pattern generation and data analysis. Test vectors are stored on the FPGA, which stimulates the custom ASIC board over two high-speed Z-DOK+ connectors. The outputs of the ASIC are captured into the block RAMs on the FPGA board for analysis. The I/O interface between the client PC and the FPGA board is built on the BEE Platform Studio environment and controlled through an integrated Simulink/Matlab interface [31]. The FPGA testing approach is favorable due to the real-time operation and low cost as compared to traditional pattern generation and logic analysis systems.

$$\cos(2\pi\alpha)\approx\begin{cases}1-\sin(2\pi\alpha)/16 & 0\le\sin(2\pi\alpha)<1/8\\ 65/64-3\sin(2\pi\alpha)/16 & 1/8\le\sin(2\pi\alpha)<1/4\\ 17/16-3\sin(2\pi\alpha)/8 & 1/4\le\sin(2\pi\alpha)<1/2\\ 5/4-3\sin(2\pi\alpha)/4 & \text{otherwise}\end{cases}$$

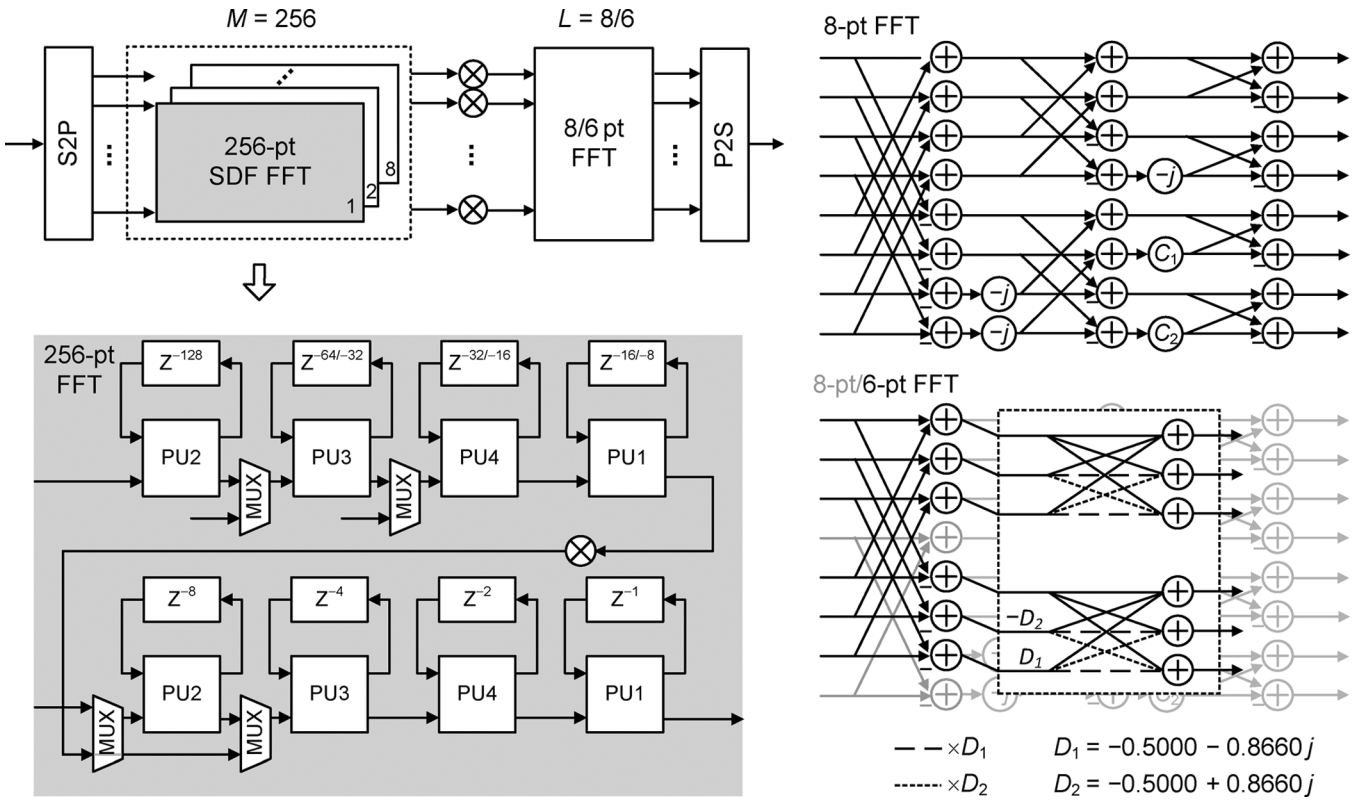


Fig. 13. Reconfigurable 2048/1536-point FFT architecture. The first-stage pipelined 256-point FFT is reconfigurable to support 16-256 points (in powers of 2). The second stage parallel FFT supports 8 or 6 points. The overall FFT meets the 3GPP-LTE standard specification (128, 256, 512, 1024, 1536, 2048 points).

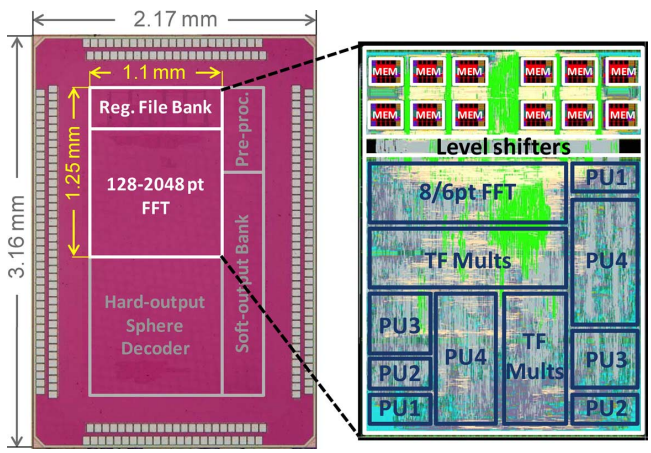


Fig. 14. Die photo of LTE-compliant MIMO sphere decoder with the reconfigurable FFT processor (highlighted) from Fig. 13.

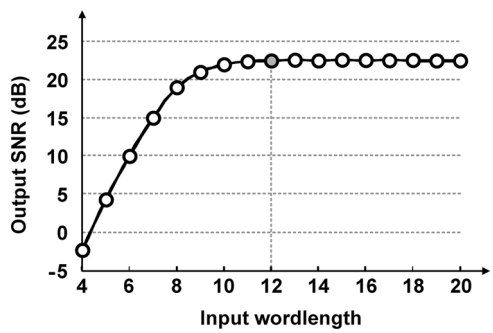


Fig. 15. Numerical performance of the FFT processor for input SNR = 25 dB. Output SNR remains constant for input wordlength greater than 12 bits.

C. Chip Measurements and Comparison to Prior Work

The *normalized Energy/FFT* [4], [21] as in (5) is used to compare the energy of the proposed FFT with prior work.

$$\text{Normalized Energy/FFT} = \frac{\text{Power} \times T_{\text{Clock}} \times N_{\text{FFT}}}{L_{\min}/65\text{ nm}} \quad (5)$$

where N_{FFT} is the number of clock cycles required to perform an FFT, T_{Clock} is the clock period, and $L_{\min}$ is the minimum channel length. The area is evaluated by the *Normalized area* [4], [12]:

$$\text{Normalized Area} = \frac{\text{Active Area}}{(L_{\min}/65\text{ nm})^2}. \quad (6)$$

Fig. 16 compares variable-length FFTs [5], [9] in terms of normalized energy and area. The power and area for real-valued FFT [5] is multiplied by a factor of 2 for fair comparison with complex-valued FFTs. Our work is used as baseline for the energy and area normalizations. As shown in Fig. 16(a), this work achieves the highest energy efficiency (lowest energy). Although the design target was to minimize the power-area product instead of energy, the energy of this work is still lower than the sub-threshold design from [5] while operating at a higher voltage (0.45 V vs. 0.35 V) and with a higher sample rate (20 MHz vs. 10 kHz). In [5], the energy is minimized by voltage scaling and circuit design for sub-threshold operation. In our work, we use radix factorization and chip synthesis tools. Further improvements in energy of our work would be possible with similar circuit-level techniques. This comparison reveals

TABLE IV
CHIP SUMMARY

Process	1P9M 65nm CMOS
FFT area (including RF bank)	1.25 x 1.1 mm^2
FFT gate count	1,100K
Clock frequency	1.25-20MHz
Memory	48kb Register File
FFT power	4.05mW (4 ant), 8.55mW (8 ant)
FFT size	128, 256, 512, 1024, 1536, 2048
Energy/FFT	2.5, 6.2, 13.9, 39.9, 58.7, 103.7nJ

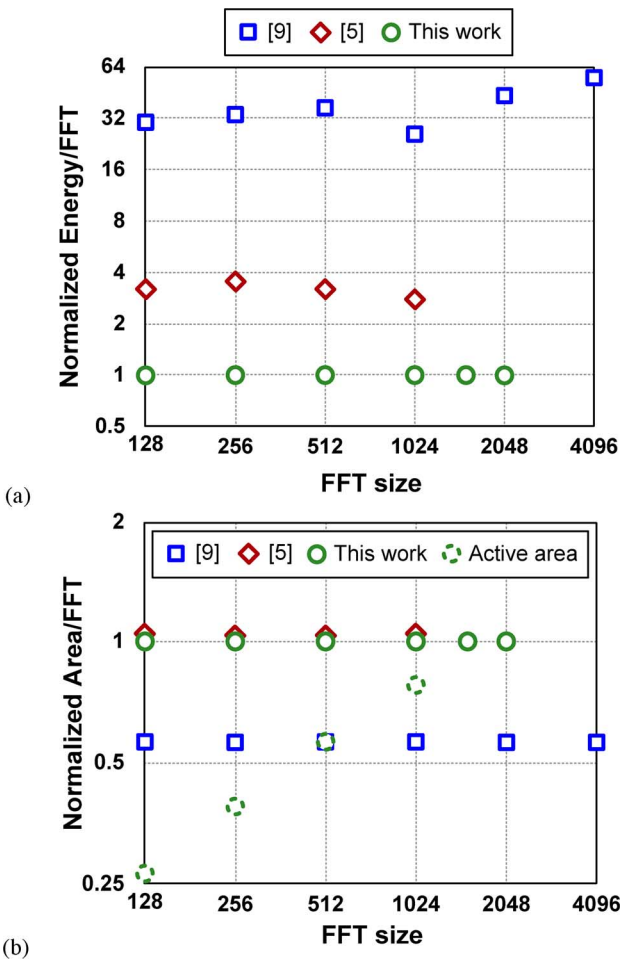


Fig. 16. (a) Energy and (b) area comparison with prior work. This work is used as baseline for energy and area normalization.

that radix factorization has higher impact than circuit-level techniques. The energy efficiency of [9] is not comparable with other ASICs due to the processor-based architecture. The area comparison is shown in Fig. 16(b). Active area indicates only the blocks that are actually used for the required FFT configuration. The chip in [9] supports larger FFT size (4096 points) with less area due to the processor-based architecture at the cost of longer latency. For example, it takes 5280 clock cycles to compute a 1024-point FFF as compared to 1024 cycles in this work.

V. CONCLUSION

An FFT processor design methodology yielding optimal power-area tradeoff is explored by examining feasible parallel architectures and radix factorizations. The use of constant multipliers for intra-stage twiddle factors enables substantial area and power savings compared to the use of full multipliers. Radix factors up to 16 should be used. Radices beyond 16 are ineffective due to a large number of constant multipliers required. Short delay line buffers (up to 256) are best realized in D-flip-flops, medium buffers (length 512 and 1024) are the most energy and area efficient when realized with register files. Twiddle factors are generated using trigonometric approximations as opposed to ROM memories. The proposed synthesis-based methodology is robust to process scaling and can quickly port across technologies. The methodology can be further refined with circuit-level customizations if that is available to the designer. It is also applicable to digital filters and general DSP architecture optimization.

As a demonstration, a 128- to 2048-point FFT processor for the 3GPP-LTE standard has been implemented in a 65-nm CMOS technology. The chip designed for minimum power-area product consumes 2.5 to 103.7 nJ/FFT for 128 to 2048 points and 1.25 to 20 MHz bandwidth (4.05 mW worst-case power consumption), making it the lowest energy flexible FFT processor.

ACKNOWLEDGMENT

The authors acknowledge the support of the Focus Center for Circuit & System Solutions (C2S2), one of six research centers funded under the Focus Center Research Program, a Semiconductor Research Corporation Program; DARPA HEALICs program; Cadence and Synopsys software; and H. Chen for help with testing.

REFERENCES

[1] S. Magar, S. Shen, G. Luikuo, M. Fleming, and R. Aguilar, "An application specific DSP chip set for 100 MHz data rates," in *Proc. Int. Conf. Acoustics, Speech, and Signal Processing*, Apr. 1988, vol. 4, pp. 1989–1992.

[2] S. He and M. Torkelson, "Design and implementation of a 1024-point pipeline FFT processor," in *Proc. IEEE Custom Integrated Circuits Conf. (CICC'98)*, May 1998, pp. 131–134.

[3] J. O'Brien, J. Mather, and B. Holland, "A 200 MIPS single-chip 1k FFT processor," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 1989, pp. 166–167.

[4] B. M. Baas, "A low-power high-performance, 1024-point FFT processor," *IEEE J. Solid-State Circuits*, vol. 34, no. 3, pp. 380–387, Mar. 1999.

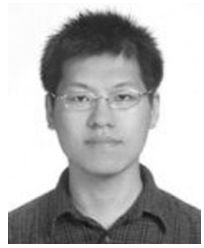
[5] A. Wang and A. Chandrakasan, "A 180-mV subthreshold FFT processor using a minimum energy design methodology," *IEEE J. Solid-State Circuits*, vol. 40, no. 1, pp. 310–319, Jan. 2005.

[6] Y. Chen, Y.-W. Lin, Y.-C. Tsao, and C.-Y. Lee, "A 2.4-Gsample/s DVFS FFT processor for MIMO OFDM communication systems," *IEEE J. Solid-State Circuits*, vol. 43, no. 5, pp. 1260–1273, May 2008.

[7] K.-S. Chong, B.-H. Gwee, and J. S. Chang, "Energy-efficient synchronous-logic and asynchronous-logic FFT/IFFT processors," *IEEE J. Solid-State Circuits*, vol. 42, no. 9, pp. 2034–2045, Sep. 2007.

[8] M. Seok, D. Jeon, C. Chakrabarti, D. Blaauw, and D. Sylvester, "A 0.27 V 30 MHz 17.7 nJ/transform 1024-pt complex FFT core with super-pipelining," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2011, pp. 342–343.

[9] G. Zhong, F. Xu, and A. N. Willson, "A power-scalable reconfigurable FFT/IFFT IC based on multi-processor ring," *IEEE J. Solid-State Circuits*, vol. 41, no. 2, pp. 483–495, Feb. 2006.
[10] Y.-W. Lin, H.-Y. Liu, and C.-Y. Lee, "A 1-GS/s FFT/IFFT processor for UWB applications," *IEEE J. Solid-State Circuits*, vol. 40, no. 8, pp. 1726–1753, Aug. 2005.
[11] Y. Chen, Y.-C. Tsao, and C.-Y. Lee, "An indexed-scaling pipelined FFT processor for OFDM-based WPAN applications," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 55, no. 2, pp. 146–150, Feb. 2008.
[12] Y.-T. Lin, P.-Y. Tsai, and T.-D. Chiueh, "Low-power variable-length fast Fourier transform processor," *IEE Proc.—Comput. Digit. Tech.*, vol. 152, no. 4, pp. 499–506, Jul. 2005.
[13] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, 3rd ed. Englewood Cliffs, NJ: Prentice Hall, 1996.
[14] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comp.*, vol. 19, pp. 297–301, 1965.
[15] P. Duhamel and H. Hollmann, "Split-radix FFT algorithm," *Electron. Lett.*, vol. 20, pp. 14–16, 1984.
[16] I. J. Good, "The interaction algorithm and practical Fourier analysis: An addendum," *J. R. Statist. Soc.*, no. 2, pp. 372–375, 1960.
[17] S. Winograd, "On computing the discrete Fourier transform," *Math. Comp.*, vol. 32, pp. 175–199, 1978.
[18] P. Duhamel, "Algorithms meeting the lower bounds on the multiplicative complexity of length-2^n DFT's and their connection with practical algorithms," *IEEE Trans. Acoustic, Speech, Signal Process.*, vol. 38, no. 9, pp. 1504–1511, 1990.
[19] W.-C. Yeh and C.-W. Jen, "High-speed and low-power split-radix FFT," *IEEE Trans. Signal Process.*, vol. 51, no. 3, pp. 864–874, 2003.
[20] T. J. Ding, J. V. McCanny, and Y. Hu, "Rapid design of application specific FFT cores," *IEEE Trans. Signal Process.*, vol. 47, no. 5, pp. 1371–1381, 2003.
[21] T.-D. Chiueh and P.-Y. Tsai, *OFDM Baseband Receiver Design for Wireless Communications*. New York: Wiley, 2007.
[22] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *IEEE J. Solid-State Circuits*, vol. 27, no. 4, pp. 473–484, Apr. 1992.
[23] K. K. Parhi, *VLSI Digital Signal Processing Systems*. New York: Wiley, 1999.
[24] D. Marković, B. Nikolić, and R. W. Brodersen, "Power and area minimization for multidimensinal signal processing," *IEEE J. Solid-State Circuits*, vol. 42, no. 4, pp. 922–923, Apr. 2007.
[25] S. W. Reitwiesner, "Binary arithmetic," *Advances in Computers*, pp. 231–308, 1966.
[26] A. Wenzler and E. Luder, "New structures for complex multipliers and their noise analysis," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'95)*, May 1995, vol. 2, pp. 1432–1435.
[27] K.-H. Lin, "Design of a Baseband Receiver for DVB-T Standard," M.S. thesis, National Taiwan University, Taipei, Taiwan, Sep. 2004.
[28] C.-H. Yang, T.-H. Yu, and D. Marković, "A 5.8 mW 3GPP-LTE compliant 8×8 MIMO sphere decoder chip with soft-outputs," in *Proc. Int. Symp. VLSI Circuits (VLSI'10)*, Jun. 2010, pp. 209–210.
[29] J. Zyren, "Overview of the 3GPP long term evolution physical layer," *Freescale White Paper*, 2007.
[30] IBOB: Interconnect Break-Out Board. [Online]. Available: http://casper.ssl.berkeley.edu/wiki/index.php/IBOB
[31] D. Marković, C. Chang, B. Richards, H. So, B. Nikolić, and R. W. Brodersen, "ASIC design and verification in an FPGA environment," in *Proc. IEEE Custom Integrated Circuits Conf. (CICC'07)*, Sep. 2007, pp. 737–740.



Chia-Hsiang Yang (S'07–M'10) received the B.S. and M.S. degrees from the National Taiwan University, Taiwan, in 2002 and 2004, respectively, all in electrical engineering. He received the Ph.D. degree from the Department of Electrical Engineering of the University of California, Los Angeles, in 2010.

He then joined the faculty of the Electronics Engineering Department at the National Chiao Tung University, Taiwan, as an Assistant Professor. His current research is focused on energy-efficient integrated circuits and architectures for biomedical and communication signal processing.

Tsung-Han Yu (S'10) received the B.S. and M.S. degrees in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2005 and 2007, respectively. He is currently working toward the Ph.D. degree in the Department of Electrical Engineering, University of California, Los Angeles.

His research is focused on digital integrated circuits and architectures for communication signal processing. Now he is working on development of algorithm and circuit architecture on wideband spectrum sensing for cognitive radios.

Dejan Marković (S'96–M'06) received the Dipl. Ing. degree from the University of Belgrade, Serbia, in 1998, and the M.S. and Ph.D. degrees from the University of California, Berkeley, in 2000 and 2006, respectively, all in electrical engineering.

In 2006, he joined the faculty of the Electrical Engineering Department at the University of California, Los Angeles, as an Assistant Professor. Since 2009, he has been affiliated with the Biomedical Engineering Interdepartmental Program at UCLA as a co-chair of the Neuroengineering field. He is also a director of the Integrated Circuits track within the UCLA Master of Science in Engineering Online Program. His current research is focused on integrated circuits for emerging radio and healthcare systems, programmable ICs, design with post-CMOS devices, optimization methods and CAD flows.

Dr. Marković was awarded the CalVIEW Fellow Award in 2001 and 2002 for excellence in teaching and mentoring of industry engineers through the UC Berkeley distance learning program. In 2004, he was a co-recipient of the Best Paper Award at the IEEE International Symposium on Quality Electronic Design. In recognition of the impact of his Ph.D. work, he was awarded 2007 David J. Sakrison Memorial Prize at UC Berkeley. He received an NSF CAREER Award in 2009. In 2010, he was a co-recipient of ISSCC Jack Raper Award for Outstanding Technology Directions and a winner of the DAC/ISSCC Student Design Contest.