A Multi-Mode IFFT/FFT Processor For IEEE 802.11ac: Design and Implementation

Abdelmohsen Ali[a], Walaa Hamouda[b]

[a] *Department of Electrical and Computer Engineering, Concordia University, Montreal, Quebec, H3G 1M8, Canada. E-mail: ali_abde@encs.concordia.ca*
[b] *Department of Electrical and Computer Engineering, Concordia University, Montreal, Quebec, H3G 1M8, Canada. E-mail: hamouda@ece.concordia.ca*

Abstract

In this paper, we present 64/128/256/512 point inverse fast Fourier transform (IFFT)/FFT processor for single-user and multi-user multiple-input multiple-output orthogonal frequency-division multiplexing based IEEE 802.11ac wireless local area network transceiver. The multi-mode processor is developed by an eight-parallel mixed-radix architecture to efficiently produce full reconfigurability for all multi-user combinations. The proposed design not only supports the operation of IFFT/FFT for 1-8 different data streams operated by different users in case of downlink transmission, but also it provides different throughput rates to meet IEEE 802.11ac requirements at the minimum possible clock frequency. Moreover, less power is needed in our design compared with traditional software approach. The design is carefully optimized to operate by the minimum wordlengths that full fill the performance and complexity specifications. The processor is designed and implemented on Xilinx Vertix-5 field programmable gate array technology. Although the maximum clock frequency is 377.84MHz, the processor is clocked by the operating sampling rate to further reduce the power consumption. At the operation clock rate of 160 MHz, our proposed processor can calculate 512-point FFT with up to eight independent data sequences within $3.2\mu sec$ meeting IEEE 802.11ac standard requirements.

Keywords: 802.11ac, FFT, MU-MIMO, FPGA

1. Introduction

Recently, the demand for wireless communications has been increasing explosively, especially for the Wireless Local Area Networks (WLAN). Although the commercially available IEEE 802.11n-based products have a maximum gross data rate of 540 Mbps [1], this technology can not afford the required throughput for the bandwidth hungry applications such as large file transfer and uncompressed multimedia streaming over IP [2]. As a consequence of the market pressure, IEEE has offered the fifth generation WiFi networking standard, IEEE 802.11ac, to deliver data rates faster than the gigabit Ethernet.

The new generation WLAN uses Orthogonal Frequency Division Multiplexing (OFDM) technology, similar to 802.11n, to cope with severe varying channel conditions and narrowband interference. OFDM is a multi-carrier transmission method that proven as a reliable and effective transmission method. The subcarriers are not only overlapped to provide high spectral efficiency but also the Discrete Fourier Transform (DFT) and its inverse are utilized to create multiple orthogonal subcarriers [3]. To reduce the effect of Inter-Symbol Interference (ISI), a Guard Interval (GI) or a Cyclic Prefix (CP) is introduced for each OFDM symbol by copying a number of time domain samples from the symbol rear to the symbol head. In OFDM, each subcarrier carries a complex symbol that is usually modulated by the well-known Quadrature Amplitude Modulation (QAM). The advanced Multiple-Input Multiple-Output (MIMO) technique can be further employed to increase the number of spatial parallel streams so that the total throughput is extremely amplified.

The 802.11ac standard guarantees a Very High Throughput (VHT) by enhancing many features to 802.11n. To support a maximum data rate of 6.933 Gbps [4], the system has to utilize 256-QAM modulation, 160 MHz channel bandwidth, and a maximum of eight parallel transmission streams. To improve the receiver performance, the Low Density Parity Check (LDPC) code is added as an optional Forward Error Correction (FEC) mechanism to the existing Binary Convolutional Code (BCC). In addition, the downlink multi-

user MIMO (MU-MIMO) is a new feature to WLAN that has been added by 802.11ac [4]. Here, multiple independent streams can be transmitted to multiple users at the same time over the same frequency band unlike the single-user MIMO (SU-MIMO) in which multiple independent streams, over multiple independent antennas, can be transmitted to a single user at the same time. In 802.11ac, MU-MIMO system supports four users with up to four spatial streams per user with the total number of spatial streams not exceeding eight.

By considering this radical change in wireless network throughput, new challenges arise for the system implementation with respect to the computational capacity and the power consumption. Among various building blocks, the DFT processor is one of the highest computational complexity modules in the physical layer of the IEEE 802.11ac standard. Up to the authors best knowledge, current research efforts are not mature enough to provide a complete complexity analysis for the physical layer of IEEE 802.11ac. However, the transform blocks in an IEEE 802.11n transceiver, which is a similar system to 802.11ac, are the second complex modules of the whole system after the channel decoder [5]. Therefore, a special attention has to be paid for these interesting blocks. The most popular algorithm for computing the DFT is the Fast Fourier Transform (FFT) algorithm which eliminates the redundant calculations needed in computing DFT and is thus very suitable for efficient hardware implementation. Various FFT architectures such as the pipelined architecture [6] and the array architecture [7] have been proposed. In fact, there has been extensive research on the hardware implementation of the FFT algorithms. However, high-throughput multi-stream applications and especially 802.11ac were not generally given more attention. The work in [8] presents a 1-4 stream support with only 64/128 point FFT but the maximum throughput is 160 MSamples/sec which is compliant with 802.11n. Furthermore, 1-8 streams are supported by [9] for only 256-point FFT and 300MHz channel. Since special Digital Signal Processing (DSP) resources are offered by modern Field Programmable Gate Array (FPGA), a reconfigurable processor is proposed by [10]. However, this processor is inconvenient for high throughput and multiple stream applications.

According to IEEE 802.11ac standard, the minimum OFDM symbol period is $3.6\mu sec$ where the guard interval is 400nsec. Therefore, the maximum required throughput for the FFT processor at the receiver side is 1.138 GSamples/sec where 512-point with 8 simultaneous data sequences are processed. To support this requirement, a software defined architecture is presented [11]. Although that is the first recoded real-time configurable architecture for 802.11ac FFT and it effectively reduces the resources, the minimum supported clock frequency is 330 MHz which is a source of high power consumption. Also, this work does not address the parallel stream processing provided by different users at the downlink transmitter. Moreover, up to our knowledge, all the high-throughput multi-stream architectures including [11] assume fixed bus width for the processor. Indeed, the compromize problem between the system performance against the hardware complexity has to be addressed accurately to develop an optimum design.

In this paper, we present a configurable VHT multi-stream IFFT/FFT processor for both downlink and uplink transceivers. The problem of a unified bus width across the design has been tackled by applying a simulation-based optimization procedure that relies on practical Signal to Quantization Noise Ratio (SQNR) values which have been carefully designed to achieve the proper system performance. The algorithm takes into account the input statistics and all quantization error sources to provide the best combination of wordlengths across the processor. In IFFT mode, the MU-MIMO is supported by scheduling different stream units to different users based on the operating mode. Proper resource sharing between different streams is employed to reduce the overall gate count. The power has been minimized by reducing the operating clock frequency to the sampling frequency. Thus, the processor is designed in a fully pipelined architecture whose issue rate is exactly one. This guarantees that the proposed dedicated circuit-based architecture operates by the minimum acceptable clock frequency. Moreover, the unused modules are switched off by the control to further reduce the required power. We even push further to consider the challenge when FPGA implementation is the target platform.

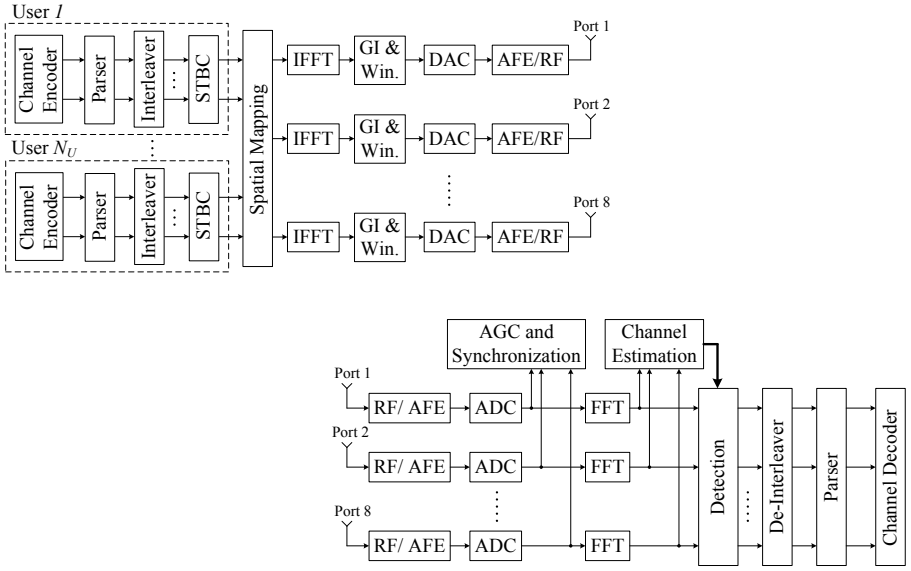

Fig. 1. Transmitter and receiver structures for 802.11ac

The paper is organized as follows. Section 2 summarizes the IEEE 802.11ac system model and provides the background for the mixed radix IFFT/FFT algorithm. In section 3, the proper architecture is selected to support the required configurations and constraints. In addition, we present the fixed point model by utilizing an empirical optimization procedure to select the proper bit-width for different variables in section 4. The implementation aspects are considered in section 5. Finally, the performance of the fixed point model and the implementation results are evaluated in section 6.

2. Background

2.1. IEEE 802.11ac System Model

Fig. 1 shows a simplified model of 802.11ac transceiver. For each user, the incoming data is randomized and channel encoded either by a BCC or an LDPC encoder. The encoded bits are parsed to multiple streams so that each stream is interleaved and modulated separately. The Space-Time Block Coding (STBC)

is an optional feature to further enhance the MIMO performance. The modulated symbols for multiple users are then precoded by the spatial multiplexer. The objective is mainly to direct each user power to the proper user receiver by means of beamforming mechanisms. The time domain OFDM symbols are synthesized by applying IFFT to each stream independently. The guard time is inserted and time domain windowing is applied to enhance the spectral properties of the OFDM signal. Finally, the analogue-converted signal is processed by the Analogue Front End (AFE) in which filtering and power amplification are required. Clearly, it is possible to have different time advances between multiple users. Therefore, the OFDM time domain constructions from different paths are not guaranteed to be time-synchronized.

At the receiver side, the received signal at each antenna is mixed, filtered, and amplified by the RF (Radio Frequency) and AFE processing. The amplification should be fully programmable to prevent the Analogue to Digital Converter (ADC) from clipping. The amplifier gain is fed by the Automatic Gain Control (AGC). After AGC has settled, the receiver should enter the synchronization mode in which timing and frequency errors are estimated and compensated. The frequency domain OFDM signal is obtained by employing FFT. The receiver needs the channel state information to decode the received frequency-domain symbols. Therefore, the MIMO channel estimation is evaluated through the transmitted training symbols and the frequency domain pilots. The received symbol vector at each sub-carrier must be decoded to separate transmitted symbols among different antennas by the detection block. The received symbols are then decoded by utilizing the channel estimates and thus the decoded symbols are demodulated with a soft demodulator which determines bit metric, the Log Likelihood Ratio (LLR), for each bit. The LLRs are delivered to perform de-interleaving and then spatial demultiplexing by the parser. Afterwards, the soft bits are decoded by the proper channel decoder to obtain the decoded data bit sequence.

2.2. Mixed Radix FFT Algorithm

Given a complex input sequence $x(n)$, an N-point DFT is defined as given by (1) and the N-point inverse DFT can be written as given by,

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

Where $W_N^{kn} \, = \, e^{-j2\pi kn/N}$ is known as the twiddle factor and $()^*$ denotes the conjugate operation. The IDFT can be obtained by simply taking the DFT to the conjugate of the input, then applying a conjugate to the result which has to be scaled by $1/N$. Due to the high computational complexity $O(N^2)$ in (1), the radix-r FFT algorithms, which have lower complexity $O(Nlog_r(N))$, have been widely adopted in DFT implementations [12]. The radix-r FFT algorithm can be easily derived from DFT by decomposing the N-point DFT into a set of recursively related r-point FFT transform, if N is a power of r. Higher radix FFT algorithm has less number of the non-trivial complex multiplications, compared with the radix-2 FFT algorithm which is the simplest form in all FFT algorithms [13]. If less multiplicative complexity is desirable, higher radix FFT algorithms should be used. However, high-radix FFT algorithms, that typically have $r > 4$, often increase the circuit complexity and are not easy to implement. The reason is that more arithmetic operations have to be involved for each radix-r stage, wider bit-widths are typically required for internal variables, and the control path needs to handle more ports for each radix-r stage.

The mixed-radix algorithm is the best in terms of the multiplicative complexity for N-point FFT when the number of DFT points is configurable [14]. This algorithm decomposes a length-N FFT into multiple radix stages to effectively reduce the number of complex multiplications. In general, if N is decomposed such that $N \, = \, r_1^{\alpha} \times r_2^{\beta}$, $r_1 \, \neq \, r_2$, and α and β are two non-zero integers, then a mixed radix decomposition is achieved. In this case, the DFT definition can be rewritten as given by (2) where we substitute for $n \, = \, r_2^{\beta} n_1 + n_2$, $k \, = \, r_1^{\alpha} k_2 + k_1$, $0 \leq n1$, $k1 < r_1^{\alpha}$, and $0 \leq n2$, $k2 < r_2^{\beta}$. The r_1^{α}-point DFT can be further decomposed to α stages of radix-r_1. Similarly, the r_2^{β}-point DFT can

be decomposed into β stages of radix-r_2. Again, the objective is to minimize the complexity and, at the same time, to support reconfigurability by allowing different numbers of α and β.

$$
\begin{aligned}
X(k) &= X(r_1^{\alpha}k_2 + k_1) \\
&= \overbrace{\sum_{n_2=0}^{r_2^{\beta}-1} \left\{ \underbrace{\left[\sum_{n_1=0}^{r_1^{\alpha}-1} x(r_2^{\beta}n_1 + n_2) W_{r_1^{\alpha}}^{k_1 n_1} \right]}_{r_1^{\alpha}\text{-point DFT}} \underbrace{W_N^{k_1 n_2}}_{\text{Twiddle factors}} \right\} W_{r_2^{\beta}}^{k_2 n_2}}^{r_2^{\beta}\text{-point DFT}}
\end{aligned} \tag{2}
$$

3. Processor Architecture

We propose a configurable 64/128/256/512-point IFFT/FFT processor to support 1-8 simultaneous data sequences for a MIMO OFDM system that is compliment with 802.11ac standard. Furthermore, the processor can process 1-8 independent data streams to support MU-MIMO. In this design, the power dissipation has been saved by employing high radix FFT, optimum bit-width selection, minimum required clock frequency, and gating the unused building blocks. The architecture supports a low power high throughput design on the account of a reasonable increase in the memory requirement when compared to low throughput designs that operate by high clock frequency. Further, the hardware complexity is minimized by optimizing the bit-width for each internal variable. The number of both complex multipliers and constant multiplications are also optimized with the specification restrictions.

There are recent researches that optimize the FFT architecture for simultaneous streams processing such as [8] and [9]. The basic assumption is that different streams are processed concurrently and there is no possibility that different streams are not time-aligned. In fact, this approach can not satisfy the requirements for MU-MIMO at the transmitter side in which separate user streams have to be processed independently. To support this feature, the processor is structured such that parallel independent units are employed to process the par-

allel independent streams. There are two degrees of freedom for the processor reconfigurability that enables the processor to function in either SU-MIMO or MU-MIMO. The number of FFT points is configurable for each stream and the number of active parallel streams can be also controlled.

Furthermore, one recent design for the FFT engine of IEEE802.11ac has been presented in [15]. Although this design is an Application Specific Integrated Circuit (ASIC) design, it is beneficial to discuss the architecture and to study its convenience to the FPGA environment. This architecture utilizes eight parallel FFT engines to process up to eight streams in parallel. To reduce the required memory at the input stage of each FFT engine, the input samples from different streams are buffered in a shared input buffer, that is implemented through registers. A scheduling technique is employed to arrange the input samples and to provide the proper samples to each FFT engine in a contention free manner and without any overflow. Despite the fact that this processor is configurable to support 64/128/256/512-point FFT processor, the design has mainly two issues: (1) It assumes a continuous processing in which no stream can be idle or delayed in time when compared to other streams (i.e, streams have to be synchronous). Again, this does not suite the MU-MIMO at the downlink transmitter. (2) The operating clock frequency is designed to be 40MHz only which implies that higher sampling rates are not supported by such a processor. For FPGA environment, storing eight complete symbols in registers is mainly a utilization issue due to the limited number of logic elements for sequential processing.

3.1. Mixed Radix Selection

The mixed radix architecture is employed to support this configurable IFFT /FFT unit for each stream. In fact, the most challenging task is to properly choose the mixed radix system including the number of required radices, the values of those radices, the number of stages for each radix, and the order of different stages. The selection criterion aims at deploying a low complexity design that is fully configurable and optimized under the umbrella of the specifications. The selection mechanism relies on couple of metrics: (1) The

architecture should be able to support all different configurations. (2) Higher radix is preferred to minimize the number of required twiddle multipliers. (3) The quantization noise sources should be kept as minimum as possible.

First, if radix-16 is involved, then each stream processing unit includes at least two radix-16 stages, one radix-4 stage, and one radix-2 stage in order to support all configurations. In this case, three twiddle multipliers are needed and 20 constant complex multipliers are required. Second, when radix-8 is employed, then each stream processing unit requires three radix-8 stages and two radix-2 stages to allow all configurations. The complexity here is represented by four twiddle multipliers and 12 constant multipliers. Last, if radix-4 is the highest radix, then each stream processing unit needs four radix-4 stages and one radix-2 stage. The number of twiddle multipliers is only four and there is no need for any constant multiplications. It is worth to mention that all these choices for the number of stages assume the minimum complexity as the basic strategy to differentiate between the choices.

When radix-8 and radix-4 systems are compared, they simply have the same number of twiddle multipliers. However, radix-8 requires an extra of 12 constant complex multipliers which add more complexity. Although utilizing radix-16 reduces the number of twiddle multipliers, it actually operates by the maximum number of constant multiplication. From the complexity perspective, 20 constant complex multipliers are a heavy weight when compared to a single complex multiplier especially when the number of bits for the constant coefficient is not too small. In addition, extremely higher radix systems are known by their complex control that amplifies the general complexity of the module. Also, when constant multiplication is involved, the system performance is influenced by the induced quantization noise from the multiplier output quantization. Based on this investigation, the conclusion is that the best practical mixed radix system for 802.11ac multi-mode IFFT/FFT processor is radix-4/radix-2 system. It does not only fit the required specifications, but it also introduces the best compromise between complexity, performance, and simple implementation.

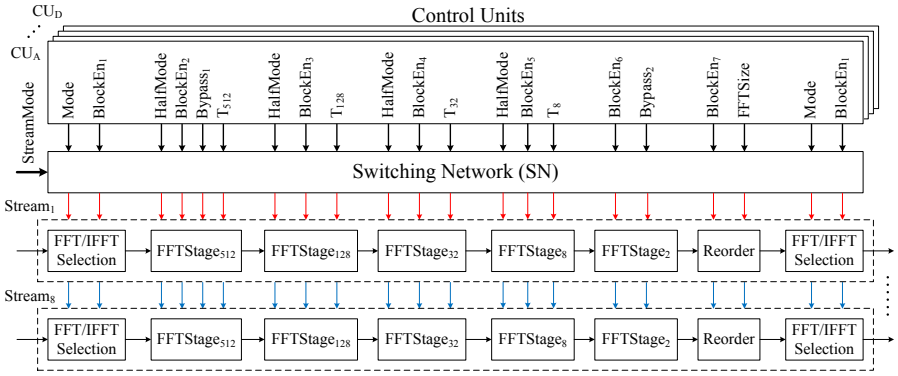

Fig. 2. Architecture for multi-stream processing with control signals

3.2. General Processor Architecture

There are eight modules to carry out the full configurable IFFT/FFT operation for a single data stream. Two of those modules are used to either apply the conjugate operation or not. If the conjugate operation is selected at both input and output, then the resulting operation is an IFFT. In addition, there are also four radix-4 stages, one radix-2 stage, and a reordering block. The first radix-4 stage is only activated if the number of FFT points is either 256 or 512. Moreover, the radix-2 stage is bypassed if the number of FFT points is either 64 or 256. The reordering block is used to shuffle the output samples so that the proper sample order is achieved. To configure a single stream processor, couple of control signals are required to decide the IFFT/FFT mode, to select the size, and to enable the corresponding stages.

Based on 802.11ac specifications, the access point can communicate with a maximum number of four users over a maximum of eight parallel streams in MU-MIMO mode where the maximum number of allowed streams per user is four. Therefore, the minimum MIMO configuration for the multi-user case is 2×2 MIMO and the maximum configuration is 4×4 MIMO. On the other hand, SU-MIMO can utilize up to eight parallel streams and hence 8×8 MIMO configuration is also allowed. Therefore, to support both SU-MIMO and MU-MIMO, eight parallel data streams are instantiated as shown in Fig. 2. However,

The control signals as well as the twiddle factors are generated by only four independent control units, namely CU_A - CU_D, that correspond to the number of active users. each control unit is responsible for deriving all the streams for one user. Therefore, a minimum of two streams and a maximum of four streams are controlled by a single control unit. The Switching Network (SN) applies the proper interconnection between the control units and the control signals for each processing stream. A demultiplexing procedure is applied by the SN to direct the control signals of different units to the proper stream.

Each control unit generates the signals that manage different modules in each stream. For all modules, there should be a common signal which acts as an enable signal. This signal is active if the module is needed to be on. Therefore, complete streams can be turned off if their individual modules are disabled. From the implementation perspective, a block enable signal is utilized inside the block as a clock enable signal for various registers. This feature is supported by current Xilinx FPGAs as a form of a power optimization approach. Typically, a logic element in Xilinx FPGA uses the clock gating concept to enable the designer to eliminate unnecessary switching activity. Thus, a logic element has a tri-state buffer controlled by the clock enable signal to either pass the clock to the register or keep it in a tri-state mode in which no power dissipation is allowed. For the IFFT/FFT selection, a mode signal is required to choose between IFFT or FFT. The first radix-4 stage needs a special bypass signal which directs the incoming data to the radix-4 stage or transfers the data to the next stage. Other than that, the first radix-4 stage has similar control signals to all other radix-4 stages. A bypass control signal is applied to radix-2 stage to either perform radix-2 processing or to deliver the input to the next stage unchanged. The reorder block needs to know the exact FFT size in order to apply the proper shuffling for the output samples.

4. Fixed Point Optimization

FFT processors are mostly implemented by fixed-point number format due to the lower hardware cost and the higher throughput of fixed-point representation compared to the floating-point. It has been formally proved [16] that if the number of fractional bits for a suitable variable or coefficient is increased by only one bit, it becomes possible to widely reduce the bit-width of several other variables and save hardware cost. Therefore, the precision analysis methods are of high importance in setting the bit-widths of fixed-point variables. In fact, increasing bit-width means reduced quantization error at the cost of area and power. On the other hand, shorter bit-width can be chosen to maintain small area and low power at the sacrifice of precision and performance.

The quantization loss due to the finite bit-width in digital systems has been classified into four categories [17] that are the roundoff noise due to arithmetic rounding operations, the coefficient quantization, the ADC quantization, and the constraint on signal level to maintain the dynamic range and prevent overflow. In recent literature, there have been various researches that have considered the first two categories. An optimization algorithm is introduced in [18] based on the SQNR analysis for Radix-2 architectures. The approximate analysis is based on widely underestimates the error by ignoring the coefficient quantization, which makes the analysis non-robust. Also, an analysis for FFT fixed-point models based on Mean Square Error (MSE) is investigated for Radix-8 FFT designs [19] and hence an optimization technique is also presented as a result of this analysis. However, the procedure assumes an overestimated quantization error to have a safe but not the optimum fixed-point model. Moreover, traditional FFT fixed point researches do not particularly elaborate on the basis by which the ADC is quantized as it is mainly an application-specific problem.

All these approaches and others such as [20] and [12] assume that the input is uniformly distributed to simplify the analysis. On the contrary, multi-path signals are received at the OFDM receiver and hence the input distribution is usually considered as Gaussian distributed. This assumption has been investi-

gated in details in [21] where the SQNR is analyzed for different received signal power and different wordlengths. However, this work aims only at introducing the proper scaling factors for different FFT stages based on the signal statistics. It does not present an optimization technique to obtain the proper number of bits for different stages.

In principle, if the processor input is the only quantized variable while other variables are left floating, then a lower bound for the quantization error, represented by $\mathrm{SQNR_{REF}}$, at the processor output can be fully determined. The ADC precision is selected based on the required dynamic range of the received signal plus noise. Theoretically, the SQNR is related to the number of allocated bits R by the relation $\mathrm{SQNR} = 6\mathrm{R} - 7.3$ dB when uniform quantization is applied [22]. The ADC signal to quantization noise ratio should account for four quantities: (1) The minimum signal to noise ratio, SNR, that is required to achieve the target uncoded system performance. (2) The PAPR, $\mathrm{R_P}$, for the received signal in order not to clip the instantaneous high power samples. (3) The ratio, $\mathrm{R_{QN}}$, between the quantization noise power and the traditional noise plus interference power. This ratio considers the amount by which the traditional noise plus interference dominates over the quantization noise power so that the detection procedure is not sensitive to the quantization noise. Enlarging this parameter will enhance the performance but will also result in higher system complexity as more bits will be involved. (4) A safety margin, SM, for the fading and uncertainties. If all parameters are defined in dBs, then the minimum signal to quantization noise ratio at the ADC output shall be $\mathrm{SNR} + \mathrm{R_P} + \mathrm{R_{QN}} + \mathrm{SM}$. This requires a number of bits for the ADC that is close to $[(\mathrm{SNR} + \mathrm{R_P} + \mathrm{R_{QN}} + \mathrm{SM} + 7.3)/6]$ bits.

The quantization noise power varies at different quantization points across the FFT stages. The amount of noise power at certain point depends on the input statistics, the number of assigned bits for this variable, and the noise sources from preceding quantizers. Therefore, in this work we rely on the 802.11ac simulation model to choose the proper wordlengths. In this model, the IFFT and FFT algorithms are manually coded according to the selected architecture. A

set of parametrized uniform quantizers are introduced to the inputs, the outputs, the twiddle factors, and the twiddle multiplication outputs for both IFFT and FFT modules. The quantizers should support the selective disabling capability and the number of bits for each variable can be fully controlled through the quantizer parameters. This model is the key step to evaluate the selected wordlength with respect to the required system performance and input signal statistics.

The SQNR at the block output due to the input quantization can be obtained for the selected input bit-width. The input has to be generated with the same distribution and same power as the integrated model. This SQNR will be the target design metric or the reference SQNR for the block because it already satisfies the required system performance. The internal variables are quantized one after each other. Different number of bits are simulated for each variable and the output SQNR is evaluated each time. The minimum number of bits that approximately satisfies the reference SQNR is chosen. Once the bit-width is selected for a certain variable, it remains fixed during other variables quantization. At the end, all internal variables are quantized with the optimum bit widths for the desired system performance, the given architecture, and real input statistics. The optimization algorithm is shown in Fig. 3 where $\mathrm{N_b}$ is the number of bits for the variable under optimization and $\mathcal{N}_{Bits}$ is the set of bits from which wordlength is chosen. In this work, we have followed the SQNR criterion defined by (3) such that K IFFT/FFT windows are involved in this definition where each IFFT/FFT window has N points. $\mathrm{Y}(m,n)$ is the floating point output at point n in window m. Similarly, $\mathrm{Y_{Fxd}}(m,n)$ is the fixed point output at point n in window m.

$$\mathrm{SQNR} \quad = \quad \frac{\sum_{m=0}^{K-1}\sum_{n=0}^{N-1}\left|\mathrm{Y}(m,n)\right|^2}{\sum_{m=0}^{K-1}\sum_{n=0}^{N-1}\left|\mathrm{Y}(m,n)-\mathrm{Y_{Fxd}}(m,n)\right|^2} \tag{3}$$

1: Evaluate the number of input bits
2: Run the standalone simulation to obtain $\mathrm{SQNR_{REF}}$
3: **repeat**
4: **for** $i = 1 : \mathrm{length}(\mathcal{N}_{Bits})$ **do**
5: $\mathrm{N_b} = \mathcal{N}_{Bits}(i)$
6: Simulate to obtain SQNR(i)
7: **end for**
8: Find the smallest k where $\mathrm{SQNR}(k) \approx \mathrm{SQNR_{REF}}$
9: Freeze $\mathrm{N_b} = \mathcal{N}_{Bits}(k)$ for the current variable
10: **until** All variables are covered
11: Report $\mathrm{N_b}$ for all variables

Fig. 3. Simulation-based fixed point optimization algorithm

5. FFT Processor Implementation

In this section, we present the hardware implementation aspects for the proposed architecture. First, when enabled, Radix-2 stage applies simple addition and subtraction to two successive inputs. The outputs are then serialized to be fed to the reorder block. Second, the traditional twiddle multiplier uses four real multipliers, one addition, and one subtraction. To reduce the complexity, the multiplier architecture presented in [20] has been utilized where only three multipliers, three adders, and two subtraction units are involved. The remaining building blocks will be discussed in details.

5.1. Radix-4 Module

To achieve the full functionality of the reconfigurable architecture, Radix-4 stages have to support two modes based on the activation of the Radix-2 stage or not. If the Radix-4 stage is designed to process M input samples, then the stage applies 4-point DFT to the whole set of M input samples in normal mode. When operating in Half mode, the Radix-4 stage applies 4-point DFT to only for $M/2$ input samples because in this case the Radix-2 stage will

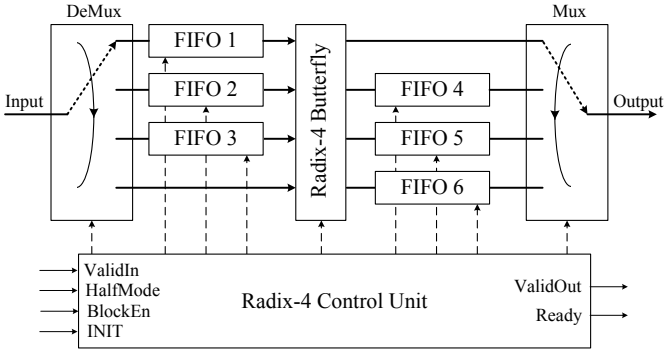

Fig. 4. Radix-4 module block diagram

be bypassed. The implementation of the radix-4 stage is fully pipelined such that a new input sample is accepted every clock cycle as long as the input is continuously valid and the processor mode is not changed. When the mode is changed or the FFT size is readjusted, each stage has to be reinitialized in order to flush the pipeline and to reset the internal control signals. Since each radix-4 stage expects a specific number of input samples, the radix-4 stage architecture should be modular enough so that the implementation supports a sufficient degree of flexibility. A configurable radix-4 stage, that can acquire a parametrized number of input samples, is designed. The hardware architecture for this design is depicted by Fig. 4.

Suppose that the expected number of input samples for a general radix-4 stage in normal mode is M every operation cycle. Since the radix-4 stage can not process the input data by the butterfly except that $3M/4$ samples are already available, the first three quarters of the input samples are buffered by FIFOs 1, 2, and 3. The demultiplexing at the input is required to properly execute this distribution process among the FIFOs and the butterfly. During the reception of the last quarter, the input samples are directed to the butterfly along with the concurrent outputs from FIFOs 1, 2, and 3. The control for the FIFOs and the butterfly is performed such that the first sample of each quarter appears together at the butterfly input. The butterfly outputs appear sequentially where four samples, representing an output group, are evaluated

each time.

As four samples can not be processed concurrently by the following stage, the first sample of each data group is issued as a valid output while the other three samples are stored simultaneously in the output buffers implemented by FIFOs 4, 5, and 6. When the butterfly finishes the input processing, the output is fed sequentially from FIFOs 4, 5, and 6 respectively with the same input order. The serialization procedure are implemented by a second set of FIFOs, rather than FIFOs 1, 2, and 3, in order to support continuous processing for the following M samples. In other words, while the second quarter of the output samples are read from FIFO 5, the first quarter of the following M samples can be written to FIFO 1 at the same time. For this reason, the FFT processor can be run by a clock frequency that is as minimum as the sampling rate since the architecture guarantees an issue rate of one. It is worth to mention that the last radix-4 stage involves a non-trivial twiddle multiplication when only radix-2 is active. The real and imaginary parts of the non-trivial twiddles are always $1/\sqrt{2}$. Therefore, to further reduce the complexity and to save one twiddle complex multiplier, this twiddle multiplication is implemented by simple shift and add logic.

5.2. Reordering Block

The reorder block architecture is shown in Fig. 5. To allow an issue rate of one, two ping-pong memory modules are used to accept a continuous stream of input samples such that the processing for two FFT windows is overlapped. The idea is that the first FFT window is placed in the first module and the following FFT window occupies the second memory module. While writing the samples into the second module linearly, the first module is read by the shuffling address. When the second module is filled, the next FFT window is written to the first memory module and the second memory is read by the interleave address. This mechanism allows a continuity for the input sequence and output sequence so that one sample is processed every clock and the reorder process never stalls. The internal control is responsible to respond to new valid inputs by updating

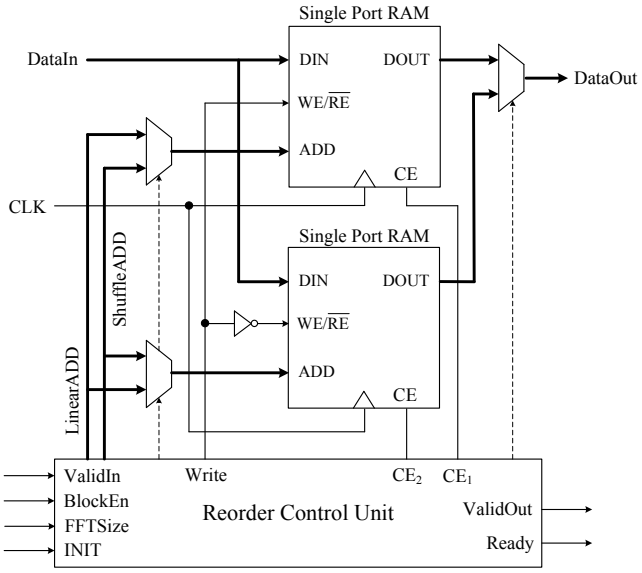

Fig. 5. Implementation for the reorder block

the linear address, the interleave address, the multiplexers control, and memory modules enables.

5.3. Twiddle Factor Generator

Twiddle factors are generated by the control units that manage different streams. Multiple streams share the same twiddle factor generation modules to reduce the hardware complexity and power consumption. A block digram of the twiddle factor generation consisting of two ROMs, two 2's complement models, two shifters, one adder, an address adjustment module, and some multiplexers is shown in Fig. 6. Only 1/8 period of cosine and sine waveforms are stored in ROM and the other period waveforms can be reconstructed by these stored values. During the reconstruction process of the full cosine and sine waveforms, the stored values may be negated or may be required in their original format based on the twiddle factor sequence number. It is left to the control, which already produces the twiddle sequence for the ROM address, to generate the control signals for the output multiplexers.

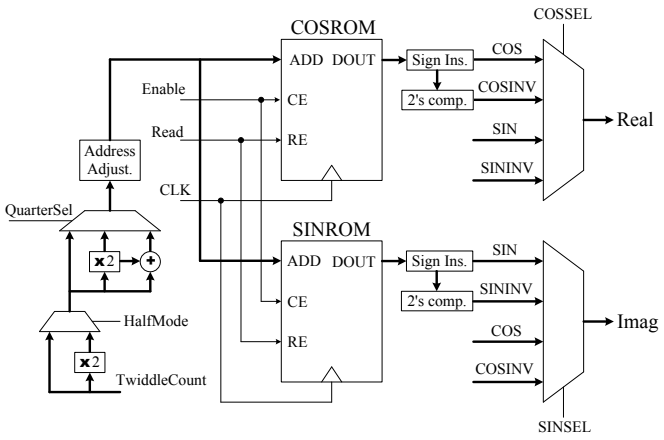

Fig. 6. Block diagram for twiddle factor generation for radix-4 stages

The twiddle factors can be divided into four groups where each group corresponds to one quarter of the full complex exponential waveform. The first group is of the trivial type which is compensated by skipping the twiddle factor multiplication. The second group is directly generated by directly accessing the tables. Since the complex exponential of third group is simply double the frequency of the complex exponential of the second group, the third group samples can be viewed as a decimated version of the first group samples with a decimation ratio of two. Therefore, the ROM address is doubled to accept a sample and drop a sample. The same concept is applied for the last group in which the decimation ratio is three. In all cases, proper sign and address adjustments are required to maintain the expected functionality. When the radix-2 is inactive, half of the normal twiddles are required. The decimation concept is applied by originally considering the up-counter value as twice as the original value that has been generated for the normal mode.

5.4. Switching Network

The switching network is the block that distributes the control signals coming from different control units to the proper streams such that the minimum number of control units is utilized for multiple-stream operation. From the

802.11ac specifications, 18 different combinations can describe all possible SU-MIMO and MU-MIMO scenarios for user/stream assignment. Among the combinations, eight possibilities are presented for a single user since one user can employ at least one stream and eight streams at most. Six combinations are allowed for two users since each user is allowed to have two streams as minimum and four streams as maximum. Recall that the multi-user case is only for MIMO systems in which at least two antennas are utilized at both the transmitter and the receiver. The remaining four combinations are divided between the three user case and four user case. The later has a unique combination in which each user utilizes two streams.

Suppose that five bits, $b_4b_3b_2b_1b_0$, are used to distinguish between the mentioned user combinations. To implement this network, a dedicated multiplexer is assigned to only one control signal for certain stream. The multiplexer inputs are hard-wired to the control units outputs such that the proper control unit is selected to manage the stream based on the user assignment code provided by the configuration bits $b_4b_3b_2b_1b_0$. This information has to provided to the processor by higher layers at the transmitter side.

6. Results

In this section, we introduce the fixed point optimization technique to the 802.11ac IFFT and FFT modules to obtain the optimum wordlengths for all IFFT and FFT variables. The final fixed point model for the IFFT/FFT processor is implemented and verified on FPGA. The implementation shows that all 802.11ac timing constrains are met by the presented architecture.

6.1. Fixed Point Simulations

To obtain the optimum wordlengths for the processor, a simulation environment that is compliant with 802.11ac specifications is coded in MATLAB where 10^4 OFDM data symbols are generated for each SNR value. The worst case scenario has been configured such that 160MHz channel is used (i.e. 512 subcarriers). Among those subcarriers, there are 468 data subcarriers that carry

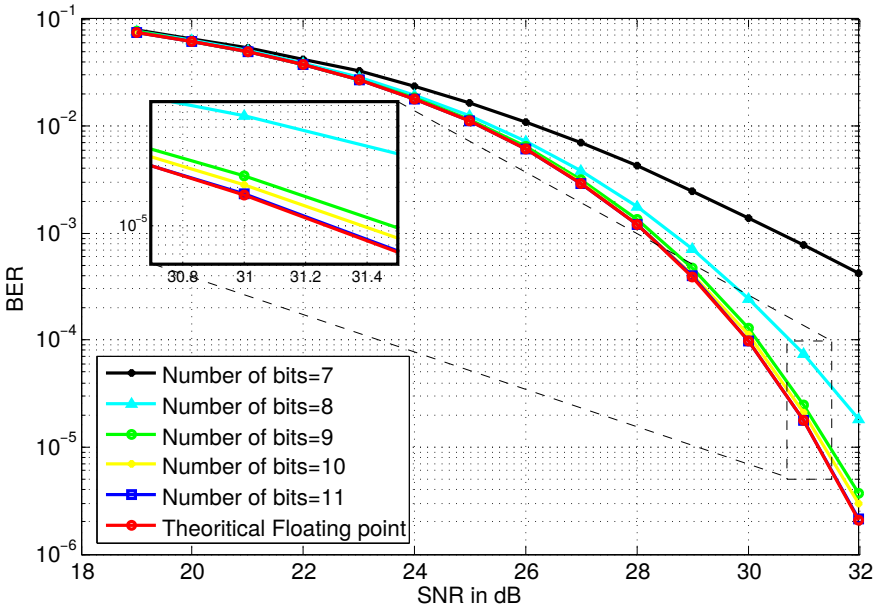

Fig. 7. System performance for different FFT input wordlengths under AWGN channel for 160MHz and 256-QAM configuration

the QAM symbols, 28 guard subcarriers including the DC-subcarrier, and 16 pilot subcarriers. 256-QAM is utilized to verify the corner case of the system. The floating point model is adapted and verified against the theoretical system performance in Additive White Gaussian Noise (AWGN) environment.

Since the uncoded performance for 256-QAM at BER $= 10^{-5}$ is achieved at SNR $\simeq$ 31.22dB, then the ADC requires at least 11 bits if we assume reasonable and practical parameters such as R_P = 7dB, R_{QN} = 20dB, and SM = 3dB. To verify this result, we run a system simulation where the wordlength of the FFT input is swept from 7 to 11 bits and AGC back-off is assumed to be 14dB. As shown in Fig. 7, the quantization noise dominates at higher SNR and more loss is obtained by decreasing the bit-width. At BER $= 10^{-5}$, the performance loss is about 0.18 and 0.02 dBs for 10 and 11 bits, respectively. In fact, this difference is not observable because of the flatness of the channel which does not contribute to the larger dynamic scale required to compensate the fading of the real channel. Therefore, 11 bits are enough for the 802.11ac FFT input

when realistic factors are considered.

A standalone simulation is applied to the FFT module only to provide wordlengths for the internal variables. The FFT input signal is assumed to be Gaussian distributed whose power is adjusted such that AGC accommodates for 14dBs back-off. When the input is the only quantized variable to 11 bits, the SQNR is measured to show a reference SQNR of 57.07dB. Following the algorithm shown in Fig. 3, The internal variables are quantized one after each other starting from former FFT stages. For each variable, the wordlength is swept from 12 to 17 bits (i.e $\mathcal{N}_{Bits}$ = [12 13 14 15 16 17]) and the SQNR is evaluated for each bit-width. The minimum wordlength that achieves an output SQNR close to the reference SQNR is selected. The SQNRs produced by different variables at different wordlengths are listed in Table 1. The selected SQNR is written in bold face to point out the corresponding bit-width and the corresponding variable. It is clear that internal quantization results in an SQNR loss of 2dBs from the reference SQNR. The overall fixed point FFT model is then simulated using the integrated 802.11ac model to investigate the overall performance loss which is found to be 0.03dB at BER = 10^{-5}. When the processor model is tested as an IFFT module by applying normalized 256-QAM modulated symbols at the input, the resulting SQNR shows 56.36dBs. In this case, the overall loss is found to be 0.05dBs at BER = 10^{-5} when we simulate the integrated model involving both IFFT and FFT modules in fixed point.

6.2. Implementation

Based on the wordlengths obtained by the fixed point optimization, the architecture of the proposed FFT processor is modelled in Verilog and functionally verified using Modelsim simulator. Test vectors have been generated by the reference fixed point MATLAB code and provided to the verilog as input files. The expected outputs are also captured into output files which are loaded by the verilog testbench to test the module functionality. Of course, the verilog has been revised till a match is obtained between the reference fixed point MATLAB code and the synthesizable verilog code. The design is then synthesised onto

Table 1. SQNR values produced by different variables with different wordlengths for FFT output

Variable	Number of bits					
	12	13	14	15	16	17
1^{st} Stage Output	51.89	**56.27**	56.80	56.95	56.99	57.04
2^{nd} Stage Output	51.95	54.17	**55.51**	55.61	55.64	55.65
3^{rd} Stage Output	52.37	54.78	**55.66**	56.07	56.09	56.09
4^{th} Stage Output	52.67	54.53	55.14	**55.65**	55.65	55.66
Twiddle factors	55.05	**55.64**	55.63	55.64	55.64	55.65
Processor Output	51.67	54.19	**55.17**	55.44	55.61	55.62

Xilinx Vertix-5 FPGA technology. We have utilized Xilinx ISE as the design software to do the required synthesis, place and route, pin assignment, and even to produce the binary files for the testing board. The translated logic is placed and routed into the target device where place-and-route simulations guarantee the processor behaviour when timing is involved.

Although it is unfair to compare different processors that have different features, it is useful to analyze each design based on another design perspective. The idea is that our design supports the MU-MIMO feature. Other processors assume simultaneous stream processing with proper time alignment which is not the case for MU-MIMO IFFT. However, the software defined processor [11] and Xilinx FFT core [23] which is used for streaming applications can support the required timing requirements. Therefore, our design is evaluated with respect to these implementations. Table 2 lists the cost and performance for our design, the software approach (shown in parentheses), and the Xilinx design (shown in brackets). The operating clock frequency is the same as the sampling frequency since the pipelined processor achieves an issue rate of one similar to the Xilinx processor. The software solution requires an operating frequency of 320MHz to account for the 160MHz channel. This processor employs a time share procedure between the process elements to reduce the complexity on the account of

Table 2. Area and performance results for the proposed IFFT/FFT processor in comparison to [11] and [23]

Streams	LUT	BRAMs	DSP units
1	2848 (1265) [3025]	7 (7)[7]	9 (20) [34]
2	5248 (2170) [6033]	13 (11) [14]	18 (40) [68]
3	7640 (3075) [9042]	19 (14) [20]	27 (60) [102]
4	9948 (3980) [12051]	24 (16) [25]	36 (80) [136]
5	12383 (4885) [15060]	30 (19) [33]	45 (100) [170]
6	14773 (5790) [18069]	36 (22) [38]	54 (120) [204]
7	17137(6695) [21078]	41 (26) [42]	63 (140) [238]
8	19424 (7600) [24047]	47 (29) [51]	72 (160) [272]

Streams	F_{max} (MHz)	Throughput (MSamples/sec)
1	377.84 (357) [413]	F_s (175) [413]
2	377.84 (368) [399]	F_s (180) [399]
3	377.84 (356) [377]	F_s (174) [377]
4	377.84 (372) [370]	F_s (182) [370]
5	377.84 (360) [363]	F_s (176) [363]
6	377.84 (373) [339]	F_s (183) [339]
7	377.84 (357) [371]	F_s (175) [371]
8	377.84 (329) [299]	F_s (161) [299]

increasing the clock frequency.

Our design has the smallest utilization for the number of DSP units. The reason is that our design first utilizes higher radix system which reduces the number of complex multiplication. Second, we have implemented each complex multiplier using only three real multipliers in addition to an increase of the logic utilization to account for extra additions. Third, we optimized the twiddle multiplication for the last radix-4 stage by utilizing simple shift and add methodology for the twiddle multiplication. Since other realizations depend on radix-2 design, they consume a large number of complex multiplication.

For the utilization of the logic elements, our design saves 6% of the corresponding look-up-tables (LUT) count for the Xilinx design for a single stream and saves 20% when eight streams are active. This reflects the amount of area reduction as a cause of the control and twiddle factor sharing. The software processor exhibits 58.3% LUT reduction on the average when compared to our design. The software solution cuts the logic utilization by iterating over the same resource twice and by employing simple control architecture. For the memory blocks, the proposed design utilizes higher memory when compared to the software defined processor because the memory is distributed among the streams which can operate simultaneously in MU-MIMO. Since the software procedure does not address the MU-MIMO case, it utilizes the memory in an efficient way as it can increase the memory data bus to account for all streams which have no memory conflicts based on the architecture. At the end, our approach still manages to have less memory capacity than the Xilinx design which has on average 2.2 Kbits increase in memory size.

It is clear from the above comparison that our design has a good compromize between area, speed, and power. However, to provide more clarity about our design and to eliminate any confusion, we have compared the presented work to the most recent designs for ASIC application such as the work presented in [15]. First, we would like to emphasize that the number of memory instances in our design is large due to the separation of the input and output buffers at every stage. Although this is not a critical issue for FPGA designs, it really requires

a second look if ASIC is the target. One option is to concatenate the real and imaginary data into a single memory instance by providing wider bus. One other limitation could be the shuffling network which consumes huge amount of wiring. This typically causes a routing issue at the back end. However, dividing the shuffling network into layers may be a good solution.

7. Conclusion

A high-throughput configurable 64/128/256/512-point IFFT/FFT processor, that is compliant to IEEE 802.11ac standard, has been proposed. In our design, eight processing units are employed to support downlink MU-MIMO feature. Based on the concept of block gating, the processor can provide low power by switching off the unused data path along with the corresponding control. Moreover, mixed-radix architecture is adopted to support a reconfigurable processor. The design is further optimized to obtain the wordlengths that satisfy the required performance at the lowest complexity.The proposed IFFT/FFT processor is implemented on Xilinx Virtex-5 FPGA. The results show that the processor meets IEEE 802.11ac standard throughputs in all operating modes at an operation clock rate that equals the sampling rate. A comprehensive comparison with other implementations is also considered.

References

[1] A. Gravalos, M. Hadjinicolaou, Q. Ni, R. Nilavalan, Spatial data stream multiplexing scheme for high-throughput WLANs, IET Communications 2 (9) (2008) 1177–1185.

[2] L. Verma, S. Lee, Proliferation of Wi-Fi: Opportunities in CE ecosystem, in: IEEE Consumer Communications and Networking Conference, 2011, pp. 213–217.

[3] R. Prasad, OFDM for Wireless Communications Systems, Artech House, London, 2004.

[4] L. Verma, M. Fakharzadeh, S. Choi, WiFi on steroids: 802.11AC and 802.11AD, IEEE Wireless Communications 20 (6) (2013) 30–35.

[5] Z. Pajouhi, S. M. Fakhraie, S. Jamali, Hardware implementation of a 802.11n MIMO OFDM transceiver, in: International Symposium on Telecommunications, 2008, pp. 414–419.

[6] K. Harikrishna, T. Rao, An efficient FFT architecture for OFDM communication systems, in: International Conf. on Advances in Recent Technologies in Communication and Computing, 2009, pp. 183–185.

[7] J. O'Brien, J. Mather, B. Holland, A 200 MIPS single-chip 1k FFT processor, in: IEEE Solid-State Circuits Conf., 1989, pp. 166–167.

[8] Y.-W. Lin, C.-Y. Lee, Design of an FFT/IFFT processor for MIMO OFDM systems, IEEE Transactions on Circuits and Systems I: Regular Papers 54 (4) (2007) 807–815.

[9] Y. Chen, Y.-W. Lin, Y.-C. Tsao, C.-Y. Lee, A 2.4-Gsample/s DVFS FFT processor for MIMO OFDM communication systems, IEEE Journal of Solid-State Circuits 43 (5) (2008) 1260–1273.

[10] I. Uzun, A. Amira, A. Bouridane, FPGA implementations of fast Fourier transforms for real-time signal and image processing, IEE Proceedings Vision, Image and Signal Processing 152 (3) (2005) 283–296.

[11] P. Wang, J. McAllister, Y. Wu, Software defined FFT architecture for IEEE 802.11ac, in: IEEE Global Conf. on Signal and Information Processing, 2013, pp. 1246–1249.

[12] J. G. Proakis, D. G. Manolakis, Digital Signal Processing: Principles, Algorithms, and Applications, 3rd Edition, Prentice Hall, India, 2003.

[13] W.-C. Yeh, C.-W. Jen, High-speed and low-power split-radix FFT, IEEE Transactions on Signal Processing 51 (3) (2003) 864–874.

[14] E. J. Kim, M. H. Sunwoo, High speed eight-parallel mixed-radix FFT processor for OFDM systems, in: IEEE International Symposium on Circuits and Systems (ISCAS), 2011, pp. 1684–1687.

[15] G. Locharla, K. Kumar, K. Mahapatra, S. Ari, Implementation of input data buffering and scheduling methodology for 8 parallel MDC FFT, in: International Symposium on VLSI Design and Test, 2015, pp. 1–6.

[16] O. Sarbishei, K. Radecka, Analysis of precision for scaling the intermediate variables in fixed-point arithmetic circuits, in: International Conf. on Computer-Aided Design (ICCAD), 2010, pp. 739–745.

[17] A. Oppenheim, C. Weinstein, Effects of finite register length in digital filtering and the fast Fourier transform, Proceedings of the IEEE 60 (8) (1972) 957–976.

[18] W.-H. Chang, T. Nguyen, On the fixed-point accuracy analysis of FFT algorithms, IEEE Transactions on Signal Processing 56 (10) (2008) 4673–4682.

[19] O. Sarbishei, K. Radecka, Analysis of mean-square-error (MSE) for fixed-point FFT units, in: IEEE International Symposium on Circuits and Systems (ISCAS), 2011, pp. 1732–1735.

[20] L. Wenqi, W. Xuan, S. Xiangran, Design of fixed-point high-performance FFT processor, in: International Conf. on Education Technology and Computer (ICETC), Vol. 5, 2010, pp. 139–143.

[21] S. Ramakrishnan, J. Balakrishnan, K. Ramasubramanian, Exploiting signal and noise statistics for fixed point FFT design optimization in OFDM systems, in: National Conf. on Comm., 2010, pp. 1–5.

[22] A. Leon-Garcia, Probability, Statistics, and Random Processes For Electrical Engineering, 3rd Edition, Prentice Hall, 2007.

[23] Xilinix INC., Xilinx LogiCORE IP fast Fourier transform v7.1, Technical Report.