IEEE 802.11ac MIMO Receiver Baseband Processing on Customized VLIW Processor

Mona AghababaeTafreshi, Lasse Lehtonen, Toni Levanen, Mikko Valkama, and Jarmo Takala

Tampere University of Technology, Tampere, Finland
mona.aghababaeetafreshi@tut.fi

Abstract— In this paper, a software-based implementation for the Multiple Input and Multiple Output (MIMO) receiver baseband processing conforming to the IEEE 802.11ac standard on a DSP core with vector extensions is presented. The implementation is carried out for different operation points including 2×2 and 4×4 MIMO configurations, yielding beyond 1Gbps transmission bit rate. This implementation mainly focuses on the frequency domain processing of the receiver. The presented solution is evaluated in terms of number of clock cycles and power consumption, and the feasibility of a real-time operation is then addressed and analyzed. If found feasible, such Software Defined Radio based solutions offer more flexibility and reduced time-to-market-cycles compared to the conventional solutions using fixed-function hardware platforms.

Keywords— *OFDM; MIMO; WLAN; Software Defined Radio; parallel processing*

I. INTRODUCTION

As wireless standards continue to evolve rapidly, the need for adaptable devices supporting different air interfaces grows. Currently, most wireless devices are implemented based on application specific fixed-function hardware platforms where most of the physical (PHY) and Medium Access Control (MAC) layer processing is still done using dedicated hardware, most notably Application-Specific Integrated Circuits (ASIC) [1]. Being implemented in silicon, such devices can offer only limited programmability and flexibility. Furthermore, adding more support for different specifications in these devices requires a larger die size and will consequently result in more power hungry devices. In contrast, a software-based solution can offer high flexibility by employing programmable and reconfigurable platforms. In addition to the lack of flexibility of ASIC implementations, the complexity and parameterization of future systems is so high that HW optimization is extremely difficult and error-prone. Using a software defined radio platform, on the other hand, the functionality can be changed by modifying the software while still maintaining high energy-efficiency compared to fixed-function hardware implementations. Such software-based implementations will enable fast scalability at the radio layer, to improve the efficiency and flexibility of RF spectrum use. Having less costs and design efforts during development, testing, and maintenance, such solutions will also clearly reduce the time-to-market cycle [1].

The extraordinary growth in the number of applications with high bandwidth requirements such as video streaming along with the increasing number of users has created an evolutionary demand to enhance the capacity of wireless networks. As a result, both mobile cellular radio networks and

Wireless Local Area Networks (WLAN) are evolving rapidly to meet these high demands. Considering in particular the wireless connectivity in indoor environments, the IEEE WLAN family provides one important technology component, in parallel to cellular mobile radio evolution. The emerging flagship amendment to IEEE 802.11TM WLAN standard with beyond 1Gbps bit rates is the IEEE 802.11ac [2].

This new amendment to IEEE 802.11TM WLAN standard is intended to meet the evolving needs for higher transmission data rates in the range of gigabits per second and to help enable new generations of data-intensive wireless applications. The IEEE 802.11ac enables multi-gigabit data throughput at 5 GHz band [2]. The IEEE 802.11ac specification adds support for 80MHz and 160MHz channel bandwidths. The 160MHz channel may be contiguous or non-contiguous, where the non-contiguous allocation provides more flexible channel assignment. Additionally, it adds higher order modulation in the form of 256 Quadrature Amplitude Modulation (QAM) which results in improved peak data rate [2]. Furthermore, by advanced deployment of multi-antenna techniques, a further increase in data rates is achieved. The Very High Throughput (VHT) physical (PHY) layer defined in [2] allows increasing the number of spatial streams up to eight streams [2]. This amendment also introduces a new technique to allow multiple users to be served simultaneously on downlink. This technique is referred to as Multi-User (MU) MIMO. MU-MIMO enables higher system capacity, more efficient spectrum use, and reduced latency [2].

The IEEE 802.11ac physical layer packet consists of PHY header part and data part. The PHY header part is divided into multiple fields where Non-HT Short Training Field (L-STF), Non-HT Long Training Field (L-LTF), Non-HT SIGNAL Field (L-SIG) are legacy portion and VHT Signal A field (VHT-SIG-A), VHT Short Training Field (VHT-STF), VHT Long Training Field (VHT-LTF) and VHT Signal B field (VHT-SIG-B) are the VHT specific fields. Fig. 1 illustrates the VHT physical layer data packet structure and the challenging timing requirements, assuming that the short guard interval (GI) is used for data symbols.

Fig. 1. Structure of VHT physical layer data packet

The majority of the previous works carried out regarding the implementation of wireless connectivity devices have focused on fixed function hardware based implementations. An example can be found in [3], where a VLSI implementation for

This work was supported by the Finnish Funding Agency for Technology and Innovation (TEKES) under the Parallel Acceleration (ParallaX) project, and Tampere University of Technology graduate school.

a 4x4 MIMO-OFDM transceiver is described which is fixed to a single operating point using 80MHz transmission bandwidth. In [4], implementation of the complete baseband processing of an IEEE 802.11a receiver on an application specific processor is described. Some contributions have been also made towards software-based solutions. However, in these works typically only parts of the PHY or MAC layer processing have been addressed. In [5], a software-defined FFT/IFFT architecture for IEEE 802.11ac is proposed based on customized soft stream processor on Field-Programmable Gate Array (FPGA). In [6], a fully programmable Software Defined Radio implementation of the IEEE 802.11 MAC that can be fully modified to develop advanced cross-layer communications and networking techniques, is presented.

In this paper, we address the feasibility of achieving a real-time operation for the IEEE 802.11ac receiver PHY layer baseband processing using a software-based implementation on a customized Very Long Instruction Word (VLIW) processor. The implementation is carried out for different transmission scenarios including 2x2 and 4x4 MIMO antenna configurations. The implemented scenarios can reach data bit rates in the order of 1Gbps. Originating from the requirements for fast processing of large amounts of data for such high data rates, a customized VLIW processor with vector processing capabilities is selected as the implementation platform. The work presented in this paper is the continuation of the transmitter implementation of the IEEE 802.11ac PHY layer baseband processing using the same platform presented in [7].

The rest of the article is organized as follows. In Section II, the implemented receiver functionalities and employed algorithms are introduced. Then, in Section III, a short description of the different implemented scenarios of the IEEE 802.11ac standard is given. In Section IV, the implementation platform and the used architecture are described. In Section V, the implementation results are presented in terms of number of clock cycles and power consumption. Finally, in Section VI, the conclusions are drawn.

II. RECEIVER PROCESSING

In this section, a brief overview of the implemented algorithms for the different functional blocks in the receiver is given.

A. SINR Estimation

To improve the link quality and performance, the SINR estimation needs to be done in the receiver. SINR estimation can be used to optimize the transmit power level and dynamically adapt the data rate. In the current implementation, we calculate the Received Channel Power Indicator (RCPI), Average Noise Power Indicator (ANPI), and the Received Signal to Noise Indicator (RSNI). The RSNI value is reported for the transmitting entity for Modulation and Coding Scheme (MCS) adaptation. The averaging of measured values is done for better stability of the system. The averaged measurements should be obtained closely in time for high correlation.

1) *RCPI measurement*: RCPI is calculated as the average power over all received R_x , $x=1,2, \dots, N_{Rx}$, receiver antennas. When receiving a Null Data Packet (NDP) for RSNI update, we calculate the RCPI over VHT-LTF symbols and VHT-SIG-B symbol. If RCPI is updated over a data packet, we

calculate the RCPI over DATA symbols. The RCPI is evaluated as the average power over all non-pilot active subcarriers. RCPI updated over a DATA packet can be written as:

$$RCPI = \frac{1}{N_{Rx}} \sum_{R_x} \text{var}(y_{R_x,t,i}) \quad (1)$$

where y represents a received DATA symbol, $t, t=1,2,\dots, N_t$, is the symbol index, $i \in I_{\text{non-pilot, active subcarriers}}$, and $\text{var}(y_{R_x,t,i})$ is the unbiased estimator of the variance, in which summation is over t and i . RCPI can be also averaged over several data packets inside a desired time window to further improve the reliability.

2) *ANPI measurement*: In the standard [8], the ANPI measurement is defined to be done during idle periods. However, as we use this value for the symbol detection, we estimate ANPI in the receiver based on the average power in the null carriers, except DC, in the STF symbols. L-STF and VHT-STF symbols are suitable for noise estimation, because they contain several zeros in the frequency domain presentation in addition to non-active carriers. Thus any changes in the subcarriers containing zeros can be considered as noise. We assume that the time and frequency synchronization accuracy while detecting STF symbols is sufficient for us to measure only noise power in the zero-valued carriers. This can be done also as a post processing step, after properly synchronizing to the received signal. ANPI can be written as:

$$ANPI = \frac{1}{N_{Rx}} \sum_{R_x} \text{var}(y_{R_x,i}) \quad (2)$$

where y represents a received L-STF or VHT-STF symbol, $i \in I_{\text{active, zero-valued pilots}}$, and the summation in $\text{var}(y_{R_x,i})$ is over i .

3) *RSNI Measurement*: Having calculated the values for RCPI and ANPI, the RSNI can be calculated according to the following:

$$RSNI = 10 \times \log_{10} \left((RCPI - ANPI) / ANPI \right) \quad (3)$$

In the above formula, the RCPI and ANPI are the power values in linear scale.

B. Channel Estimation

For detecting a WLAN 802.11ac packet, the receiver has to calculate two different channel estimates, one for the non-precoded (non-VHT) part and one for the possibly precoded part (VHT-part). First channel estimate is obtained from L-LTF symbols for detecting L-SIG and VHT-SIG-A fields. Second channel estimate is obtained, after detecting VHT-SIG-A, from VHT-LTF fields.

1) *Channel estimator for the legacy part*: After time and frequency synchronization, Cyclic Prefix (CP) removal and FFT operation, the received signal for L-LTF symbol per symbol index t , $t = [1, 2]$, per subcarrier index k , $k \in I_{\text{active, non-pilot L-LTF subcarriers}}$, can be written as (4) where $\mathbf{H}_k$ is a $(N_{Rx} \times N_{Tx})$

complex channel matrix, $\mathbf{H}_{eff,k}$ is the $(N_{Rx} \times 1)$ effective sum channel for legacy part, $\mathbf{X}_{L-LTF,k}$ is an $(N_{Tx} \times 1)$ real vector containing only the training symbol $\mathbf{x}_{L-LTF,k}$ (ones and minus ones) and $\mathbf{N}_{t,k}$ is an $(N_{Rx} \times 1)$ complex Gaussian noise vector.

$$\begin{aligned} \mathbf{Y}_{k,t} &= \mathbf{H}_k \mathbf{X}_{L-LTF,k} + \mathbf{N}_{t,k} \\ &= \mathbf{x}_{L-LTF,k} \begin{bmatrix} \frac{1}{N_{Tx}} \sum_{j=1}^{N_{Tx}} \mathbf{h}_{1,j} \\ \vdots \\ \frac{1}{N_{Tx}} \sum_{j=1}^{N_{Tx}} \mathbf{h}_{N_{Rx},j} \end{bmatrix} + \mathbf{N}_{t,k} = \mathbf{x}_{L-LTF,k} \mathbf{H}_{eff,k} + \mathbf{N}_{t,k} \end{aligned} \quad (4)$$

Now, given that we receive two L-LTF symbols in the preamble, the Least Squares (LS) channel estimator is given as:

$$\mathbf{H}_{eff,k,LS} = \frac{\mathbf{x}_{L-LTF,k}}{2} \sum_{t=1}^2 \mathbf{Y}_{k,t} \quad (5)$$

where index t , $t = [1, 2]$, indicates L-LTF symbol index over which the received signal is averaged before channel estimation. The channel estimate is obtained by directly multiplying with $\mathbf{x}_{L-LTF,k}$ because it can only have values $[-1, +1]$. With $\mathbf{H}^H$ representing the Hermitian transpose of Matrix $\mathbf{H}$, the Linear Minimum Mean Square Error (LMMSE) estimator based on LS estimate can be written as:

$$\begin{aligned} \hat{\mathbf{H}}_{eff,k,LMMSE} &= \hat{\mathbf{H}}_{eff,k,LS} \hat{\mathbf{H}}_{eff,k,LS}^H \\ &\times \left(\hat{\mathbf{H}}_{eff,k,LS} \hat{\mathbf{H}}_{eff,k,LS}^H + \frac{\sigma_n^2}{2} \mathbf{I}_{N_{Rx}} \right)^{-1} \hat{\mathbf{H}}_{eff,k,LS} \end{aligned} \quad (6)$$

2) *Channel estimator for VHT part:* After detecting VHT-SIG-A, the receiver knows how many VHT-LTF symbols it should collect for VHT channel estimation. The VHT-LTF preamble differs from the legacy part in two main ways. First, the VHT-LTF subcarriers k , $k \in \mathbf{I}_{active, non-pilot \text{ VHT-LTF subcarriers}}$, are precoded by VHT-LTF mapping matrix $\mathbf{P}$ (defined in [2]) of size $(N_{STS} \times N_{VHT-LTF})$. Secondly, the VHT-LTF symbols may be precoded by the precoder matrix $\mathbf{Q}_j$, $j \in \mathbf{I}_{active, VHT-LTF \text{ subcarriers}}$. After synchronization, CP removal and FFT operation, the received signal for VHT-LTF symbol per symbol index t , $t = [1, \dots, N_{VHT-LTF}]$, per subcarrier index k , $k \in \mathbf{I}_{active, non-pilot \text{ VHT-LTF subcarriers}}$, can be written as:

$$\begin{aligned} \mathbf{Y}_{k,t} &= \mathbf{H}_k \mathbf{Q}_k \mathbf{P}(:, t) \mathbf{x}_{VHT-LTF,k} + \mathbf{N}_{t,k} \\ &= \mathbf{H}_{eff,k} \mathbf{P}(:, t) \mathbf{x}_{VHT-LTF,k} + \mathbf{N}_{t,k} \end{aligned} \quad (7)$$

Now, in the receiver, to get effective channel estimates per Space Time Stream (STS), the received VHT-LTF symbols are weighted with the rows of $\mathbf{P}$ matrix and averaged over all VHT-LTF symbols. For presentation clarity, let us stack the received samples per subcarrier k , over all Rx antennas and VHT-LTF symbols into single column vector, given as (8) where $\otimes$ represents Kronecker tensor product.

$$\begin{aligned} \mathbf{Y}_k &= \begin{bmatrix} \mathbf{Y}_{k,1} \\ \vdots \\ \mathbf{Y}_{k,N_{VHT-LTF}} \end{bmatrix} \\ &= (\mathbf{P} \otimes \mathbf{I}_{N_{Rx}})^T \begin{bmatrix} \mathbf{h}_{eff,k}(:, 1) \\ \vdots \\ \mathbf{h}_{eff,k}(:, N_{VHT-LTF}) \end{bmatrix} \mathbf{x}_{VHT-LTF,k} + \mathbf{N}_k \end{aligned} \quad (8)$$

Now, the received training VHT-LTF training signal after decoding diversity coding is given as:

$$\begin{aligned} \tilde{\mathbf{Y}}_k &= \frac{1}{N_{VHT-LTF}} (\mathbf{P} \otimes \mathbf{I}_{N_{Rx}}) \begin{bmatrix} \mathbf{Y}_{k,1} \\ \vdots \\ \mathbf{Y}_{k,N_{VHT-LTF}} \end{bmatrix} \\ &= \mathbf{x}_{VHT-LTF,k} \begin{bmatrix} \mathbf{h}_{eff,k}(:, 1) \\ \vdots \\ \mathbf{h}_{eff,k}(:, N_{VHT-LTF}) \end{bmatrix} + \mathbf{W}_k \end{aligned} \quad (9)$$

where $\mathbf{W}_k \in \mathcal{CN}\left(0, \frac{\sigma_n^2}{N_{VHT-LTF}}\right)$. Then, the LS channel estimate can be written as:

$$\hat{\mathbf{H}}_{eff,k,LS} = \mathbf{x}_{VHT-LTF,k} \tilde{\mathbf{Y}}_k \quad (10)$$

After obtaining the LS channel estimate, it is used to calculate the LMMSE channel estimation using (6) by

replacing $\frac{\sigma_n^2}{2} \times \mathbf{I}_{N_{Rx}}$ with $\sigma_n^2 \mathbf{I}_{N_{Rx} N_{VHT-LTF}}$.

During the receiver implementation, LMMSE channel estimation proved to be one of the most time consuming operations as it involves the inversion of a 4×4 matrix according to (6). If (6) is written assuming a two antenna configuration, (11) is derived as:

$$\begin{aligned} \hat{\mathbf{H}}_{eff,k,LMMSE} &= \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \begin{bmatrix} a^* & b^* & c^* & d^* \end{bmatrix} \\ &\times \left(\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \begin{bmatrix} a^* & b^* & c^* & d^* \end{bmatrix} + \begin{bmatrix} I_N & 0 & 0 & 0 \\ 0 & I_N & 0 & 0 \\ 0 & 0 & I_N & 0 \\ 0 & 0 & 0 & I_N \end{bmatrix} \right)^{-1} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \end{aligned} \quad (11)$$

where a , b , c , d are calculated according to (10), $\hat{\mathbf{H}}_{eff,k,LS} = [a \ b \ c \ d]^T$, $\mathbf{I}_N = \sigma_n^2 \mathbf{I}_{N_{Rx} N_{VHT-LTF}}$, and the superscript $*$ represents the complex conjugate of the corresponding element. Note that now the columns of the original channel matrix are stacked on top of each other.

Now, using the Sherman-Morrison law [9], (11) can be simplified to (12) where $\text{conj}(\mathbf{H})$ represents the element wise complex conjugate of matrix $\mathbf{H}$.

$$\hat{\mathbf{H}}_{eff,k,LMMSE} = \begin{bmatrix} \frac{a(aa^* + bb^* + cc^* + dd^*)}{(\sigma_n^2 + aa^* + bb^* + cc^* + dd^*)} \\ \frac{b(aa^* + bb^* + cc^* + dd^*)}{(\sigma_n^2 + aa^* + bb^* + cc^* + dd^*)} \\ \frac{c(aa^* + bb^* + cc^* + dd^*)}{(\sigma_n^2 + aa^* + bb^* + cc^* + dd^*)} \\ \frac{d(aa^* + bb^* + cc^* + dd^*)}{(\sigma_n^2 + aa^* + bb^* + cc^* + dd^*)} \end{bmatrix} \quad (12)$$

$$= \frac{\mathbf{H}_{LS} \mathbf{H}_{LS}^T \times \text{conj}(\mathbf{H}_{LS})}{\sigma_n^2 + \mathbf{H}_{LS}^T \times \text{conj}(\mathbf{H}_{LS})}$$

Using this approach, we can avoid the complexity due to the matrix inversion in the channel estimation process. The same approach can be used in the 4x4 antenna configuration. Simplifying the computation of the matrix inversion reduced the complexity and the number of clock cycles to a great extent.

C. Pilot Based Fine Frequency Error Estimation and Correction

In order to compensate the effects of the frequency error on the received symbols, first the frequency error should be estimated. The frequency error measured using the phase angle difference per symbol index t can be given as:

$$\hat{\Theta}_{t+1} = \frac{1}{N_p} \sum_{i \in \Omega_P} \angle P_{i,t+1} - \angle P_{i,t} \quad (13)$$

where $i \in I_{\text{pilot, DATA subcarriers}}$. Having the phase angle differences, the frequency error can be calculated as:

$$\hat{F}_{error,t+1} = \hat{F}_{error,t} + \frac{F_s}{N_s + N_{GI}} \hat{\Theta}_{t+1} \quad (14)$$

where F_s is the sampling frequency, N_s is the number of subcarriers, and N_{GI} is the number of samples in the Guard Interval (GI). Once the frequency error is calculated, the received DATA symbols can be corrected using:

$$\mathbf{y} = \mathbf{y}_{\text{received}} \times e^{-j\pi \hat{F}_{error,t}} \quad (15)$$

D. LDPC Tone Demapping

When Low Density Parity Check (LDPC) encoder is used as the Forward Error Correction (FEC) method, LDPC tone mapping should be employed in the transmitter, whereas in case of Binary Convolutional Codes (BCC), BCC interleaver shall be employed. LDPC tone mapper was introduced in 802.11ac to achieve full frequency diversity from 80MHz and 160MHz bandwidths. The LDPC tone mapper maps consecutive symbols to non-consecutive subcarriers inside one OFDM symbol. In other words, the LDPC tone mapper shuffles the data subcarriers in each OFDM symbol in each spatial stream. Thus in the receiver, the LDPC tone demapper rearranges the shuffled subcarriers into their original places.

E. Stream De-parser

Stream parsing is the operation done in the transmitter to rearrange and divide the coded bits into N_{ss} spatial streams. The left-hand side in Fig. 3 illustrates how stream parsing is done for an unknown number of streams. In the receiver the N_{ss} streams are then de-parsed to form one bit stream as shown in the right-hand side of Fig. 2.

Fig. 2. Stream parsing and de-parsing process

F. Symbol Detection

To detect the symbols at the receiver, LMMSE detection is employed. The detector coefficients can be calculated using the LMMSE channel estimation derived in (6). The detector coefficients can be calculated as:

$$\mathbf{D}_{coeff} = \left(\mathbf{H}_{LMMSE}^H \mathbf{H}_{LMMSE} + \sigma_n^2 \mathbf{I}_{N_{STS}} \right)^{-1} \mathbf{H}_{LMMSE}^H \quad (16)$$

where the $\mathbf{H}_{LMMSE}^H$ is a $(N_{Rx} \times N_{STS})$ matrix. Once the coefficients are calculated, the received symbols can be detected using:

$$\hat{\mathbf{X}} = \mathbf{D}_{coeff, fixed} \mathbf{Y} \quad (17)$$

where $\mathbf{Y}$ is a $(N_{Rx} \times 1)$ matrix containing the received symbols.

G. Soft Bit Detection

In soft bit detection, for each bit position, the difference of distances to the nearest zero and one bit on the constellation is calculated. This operation is illustrated in Fig. 3. This is a sub-optimal method for reducing the complexity of the soft bit detection implementation where instead of calculating the distance to all constellations, only the nearest ones are considered.

Fig. 3. Soft bit detection process

III. TRANSMISSION SCENARIOS

In this work, we have covered four different operation points (transmission scenarios) of the IEEE 802.11ac. The transmitter

implementation for these cases was discussed in [7]. In all cases the channel bandwidth is set to 80MHz, which implies that each OFDM symbol contains 256 subcarriers including 234 data, 14 null, and 8 pilot subcarriers. Furthermore, in all cases 256QAM is selected as the modulation scheme mapping a block of 8 coded bits into one constellation point. In this implementation short GI is used implying that the duration of each OFDM DATA symbol is equal to 3.6 μ s.

Fig. 4 and Fig. 5 depict main structure of the implemented processing at the receiver. Some blocks may be obsolete in some cases depending on the scenario. It is also assumed that the incoming symbols are stored in a local memory and consequently the time required for the transfer of the data to the local memory is not considered.

Fig. 4. Overall block diagram of frequency domain receiver processing

Fig. 5. Block diagram of the receiver DATA field baseband processing

Table I briefly describes the different transmission scenarios implemented in this work and highlights the common parameters and differences in these four operation points.

TABLE I. THE SPECIFICATIONS FOR THE IMPLEMENTED TRANSMISSION SCENARIOS

	Common Parameters	Number of antennas	Number of spatial streams	STBC coding
CASE A	80MHz TX/RX bandwidth, 256QAM, 1/4 coding rate, Short GI	2	2	No
CASE B		4	4	No
CASE C		2	1	Yes
CASE D		4	2	Yes

IV. ARCHITECTURE AND IMPLEMENTATION

The implementation platform used in this work was selected by taking into consideration the requirement for fast processing of huge amounts of data, imposed by the IEEE 802.11ac support for very high data rates in the order of gigabits per second. As a result a VLIW processor with vector processing capabilities is chosen. More specifically, we have selected the Tensilica ConnX BBE32 DSP core as our processing platform in this work. This DSP core, which is specifically designed to be used in the next generation communication systems, is based on a high performance, ultra-low power, and very small size architecture [10]. The ConnX BBE32 meets the high computational requirements by supporting vector operations using a 16-way SIMD ALU and a 4-issue VLIW processing pipeline. Additionally, this core is equipped with 32 multiply-accumulate units and can access wide data chunks in blocks of

256 bits from the memory. The ConnX BBE32 block diagram can be found in Fig. 6. This DSP core uses a Harvard architecture having two data memories and one instruction memory. Moreover, to help offload the computationally intensive operations such as FFT/IFFT, the dedicated hardware accelerator blocks can be used, which are then controlled with custom instruction extensions. Tensilica uses an Eclipse based software development environment named Xtensa Xplorer, which provides a complete set of tools for code generation and profiling. Programming in C language is possible in this environment. However, we have manually optimized our code with the aid of the compiler intrinsics.

Fig. 6. Principal block diagram of ConnX BBE32

V. RESULTS AND ANALYSIS

In order to study the feasibility of the introduced solution on this platform, we have profiled and then analyzed the solution in terms of number of clock cycles and power consumption. This has been done using the profiling tools provided by the vendor. Two of the most challenging symbols to process in the IEEE 802.11ac packet structure are the VHT-LTF symbol and the DATA symbol. The reason for this is that the VHT-LTF symbol is used for channel estimation and calculating the detector coefficients which are the two most computationally intensive operations. The DATA symbol also involves heavy operations such as the soft bit detection. As a result, we have presented the power consumption and clock cycle results related to these two symbols in this section.

Table II shows the number of clock cycles needed for the different operations on one DATA symbol in the receiver. The results for the four different operation points are presented in the same table. As it can be seen from Fig. 1, the duration of one DATA symbol is 3.6 μ s when short GI is used. To achieve real-time operation in the receiver, all the processing should not exceed 3.6 μ s.

TABLE II. THE NUMBER OF CLOCK CYCLES NEEDED FOR THE PROCESSING OF A DATA SYMBOL

	Case A	Case B	Case C	Case D
RCPI Variance	185	365	185	365
RCPI, ANPI, RSNI	146	150	146	150
Remove Pilots	140	336	140	336
Symbol Detection	468	625	208	652
LDPC Tone De-Mapper	324	672	324	672
Stream Deparser	74	254	-	148
Soft Bit Detection	3193	6504	3193	6504
Frequency Error Correction	255	331	255	331
Total	4785	9237	4451	9158

Looking at the total number of cycles presented in Table II, in the first and third scenarios (scenarios using only two

antennas), an operating frequency of 1GHz is required for achieving a real-time operation. However, in the second and fourth cases (where four antennas are used) the clock frequency should be doubled.

The duration of one VHT-LTF symbol is 4 μ s. According to the total number of clock cycles presented in Table III, to achieve a real-time operation, frequencies less than 1GHz are required for the cases using two space-time streams, whereas very high frequencies may be needed for the cases with four space-time streams due to the high number of cycles consumed by the detector co-efficient calculation function.

TABLE III. THE NUMBER OF CLOCK CYCLES NEEDED FOR THE PROCESSING OF A VHT-LTF SYMBOL

	Case A	Case B	Case C	Case D
LS Channel Estimation	281	2873	281	2873
LMMSE Channel Estimation	1078	1643	1078	1643
Detector Coefficients	2630	31713	851	34005
Total	3989	36229	2210	38521

The matrix to be inverted for calculating the detector coefficients does not benefit from the special structure available for the LMMSE channel estimation. Therefore, it could not be simplified using the same methods. We continue our work to reduce the frequencies required to achieve a real-time operation by introducing instruction extensions for the bottleneck operations such as 4 $\times$ 4 unstructured matrix inversion in the symbol detector coefficient calculations. This is done by adding a customized inversion accelerator to the core.

One important criterion, which needs to be taken into consideration during the implementation, is the power consumed by the design. As the Xtenxa Xplorer tools provide the energy consumption estimation, we have calculated the power consumption using the energy numbers and dividing those by time. The time needed for each block is defined by the number of clock cycles and we have considered maximum memory capacity (128k). We have assumed a clock frequency of 500MHz and the monitoring time for the energy analysis is 3.6 μ s for the DATA part and 4 μ s for the VHT-LTF. Table IV and V present the power consumption results for different operations in all transmission cases for DATA and VHT-LTF symbols, respectively. In general the power consumption results are mostly found feasible to mobile terminal scale devices. It should be noted that the VHT-LTF operations are done only once per packet, but the data symbol operations are repeated multiple times per packet. Therefore, minimizing the power consumption per data symbol is more critical. In other words, for a VHT-LTF symbol the detector coefficient evaluation is a time critical operation and processing a data symbol is a power critical operation.

VI. CONCLUSIONS

In this paper, we have proposed a software-based implementation for the IEEE 802.11ac receiver frequency domain PHY layer baseband processing. This implementation was carried out using a customized DSP core with vector processing capabilities. The solution was developed for four different multi-antenna transmission scenarios. The implementation has been evaluated in terms of number of clock

cycles and power consumption. We presented the results for two of the symbols that require more computations, namely the DATA and VHT-LTF symbols. The analysis of the performance numbers showed that achieving a real-time operation for the IEEE 802.11ac receiver on this customized DSP platform requires very high operating frequencies. In the continuation of this work, we customize the core by adding instruction extensions for the computationally intensive operations such as matrix inversion to lower the operating frequency needed for achieving a real-time operation.

TABLE IV. POWER CONSUMPTION IN MW FOR THE PROCESSING OF A DATA SYMBOL

	Case A	Case B	Case C	Case D
RCPI Variance	3,36	6,67	3,35	6,16
RCPI,ANPI,RSNI	2,76	2,68	2,59	2,78
Remove Pilots	4,76	10,39	4,76	9,52
Symbol Detection	13	23,51	6,82	34,65
LDPC Tone De-Mapper	10,02	20,07	10,02	20,03
Stream Deparser	3,06	8,59	-	6,11
Soft Bit Detection	106,95	213,92	105,43	213,19
Frequency Error Correction	4,46	7,55	4,44	7,55
Total	148,37	293,38	140,5	299,99

TABLE V. POWER CONSUMPTION IN MW FOR THE PROCESSING OF A VHT-LTF SYMBOL

	Case A	Case B	Case C	Case D
LS Channel Estimation	8,49	96,45	8,51	85,64
LMMSE Channel Estimation	31,27	51,22	31,27	51,22
Detector Coefficients	66,39	38,4	21,55	94,03
Total	106,15	186,07	61,33	230,89

REFERENCES

- [1] NVIDIA, *NVIDIA SDR (Software Defined Radio) Technology*, USA, 2013.
- [2] IEEE P802.11ac™ Draft Standard, version 5, January 2013.
- [3] S. Yoshizawa and Y. Miyanaga, "VLSI Implementation of a 4 $\times$ 4 MIMO-OFDM transceiver with an 80-MHz channel bandwidth," in *Proc. IEEE ISCAS*, Taipei, Taiwan, 24-27 May 2009, pp. 1743-1746.
- [4] S. Eberli, A. Burg, T. Bosch, and W. Fichtner, "An IEEE 802.11a baseband receiver implementation on an application specific processor," in *Proc. 50th Midwest Symposium on Circuits and Systems*, Montreal, Canada, 5-8 Aug. 2007, pp.1324,1327.
- [5] P. Wang, J. McAllister, and Y. Wu, "Software defined FFT architecture for IEEE 802.11ac," in *Proc. Global Conference on Signal and Information Processing (GlobalSIP)*, Austin, USA, 3-5 Dec. 2013, pp.1246-1249.
- [6] J.R. Gutierrez-Agullo, B. Coll-Perales, and J. Gozalvez, "An IEEE 802.11 MAC Software Defined Radio implementation for experimental wireless communications and networking research," in *Proc. IFIP Wireless Days*, Venice, Italy, 20-22 Oct. 2010, pp.1-5.
- [7] M. Aghababaeatafeshi, L. Lehtonen, M. Soleimani, M. Valkama, and J. Takala, "IEEE 802.11ac MIMO transmitter baseband processing on customized VLIW processor," in *Proc. IEEE ICASSP*, Florence, Italy, 4-9 May 2014, pp. 7550-7554.
- [8] IEEE Std 802.11™ Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, March 2012.
- [9] M. S. Bartlett, "An inverse Matrix Adjustment Arising in Discriminant Analysis," *The Annals of Mathematical Statistics* 22, no. 1, pp. 107-111.
- [10] Tensilica Inc., *ConnX BBE32 DSP User Guide*, USA, 2012.