IEEE 802.11AC MIMO TRANSMITTER BASEBAND PROCESSING ON CUSTOMIZED VLIW PROCESSOR

Mona Aghababaeetafreshi[1], Lasse Lehtonen[2], Maliheh Soleimani[1], Mikko Valkama[1], and Jarmo Takala[2]

[1]Department of Electronics and Communications Engineering

[2]Department of Pervasive Computing

Tampere University of Technology, Korkeakoulunkatu 1, FI-33720 Tampere, Finland

Email: mona.aghababaeetafreshi@tut.fi

ABSTRACT

This paper presents a software-based implementation for the MIMO transmitter baseband processing conforming to the IEEE802.11ac standard on a DSP core with vector extensions. The transmitter is implemented in four different transmission scenarios, which include 2×2 and 4×4 MIMO configurations, yielding beyond 1Gbps transmit bit rate. The implementation is done for the frequency-domain processing and real-time operation has been achieved when running at a clock frequency of 500MHz. The proposed software solution is evaluated in terms of power consumption, number of clock cycles and memory usage. This SDR based implementation provides improved flexibility and reduced design effort compared to conventional approaches while maintaining energy consumption close to fixed-function hardware solutions.

Index Terms— OFDM, MIMO, WLAN, Software Defined Radio, Parallel Processing

1. INTRODUCTION

Due to the rapid growth and popularity of wireless handheld devices with efficient support for rich multimedia functionalities and broadband Internet access, both mobile cellular radio networks and Wireless Local Area Networks (WLAN) are evolving rapidly. While broadband wireless access is typically the driving priority, security, low power, low cost and reliability are also seen as very important aspects. Considering in particular the wireless connectivity in indoor environments, WLAN/WiFi solutions with optimized local area access for physical (PHY) and medium access control (MAC) layers are of increasing interest. This is also the main focus area of this article.

Currently, the clear majority of wireless local area connectivity is provided by IEEE WLAN/WiFi solutions whose flag-ship technology is IEEE 802.11ac [1]. In this standard, the throughput enhancements compared to legacy systems are obtained mainly through the deployment of advanced PHY layer innovations such as considerably wider transmission bandwidth through carrier aggregation, improved modulation and coding schemes and advanced deployment of multiantenna/MIMO transmission schemes. The standard utilizes transmission bandwidths up to 80MHz (mandatory) and 160MHz (optional), which is substantial improvement compared to 802.11n legacy system. Moreover, the flexibility of RF spectrum use is improved through allowing non-contiguous carrier aggregation where the total RF bandwidth can be composed of non-contiguous channels. Furthermore, multiantenna support up to 8×8 MIMO with eight spatial streams is specified, including also multiuser MIMO. The IEEE 802.11ac amendment also allows modulation orders up to 256QAM to further increase the highest achievable throughput. Overall, the instantaneous peak throughputs can reach 1Gbps [2].

In the existing literature, a clear majority of local area connectivity device implementations, in particular 802.11ac related, are fixed-function hardware based solutions. In [3], a VLSI implementation of a 4×4 MIMO-OFDM transceiver with 80MHz transmission bandwidth is described, and tailored to a single transmission scenario. In recent reports, some contributions have also been made towards the software defined radio concept. Design and implementation of the IEEE802.11 MAC layer processing on general-purpose DSP and additional accelerator systems is reported in [4]. In [5], a software defined radio implementation of 802.11 MAC with emphasis on cross-layer communications and networking is proposed and evaluated. However, as it can be seen also in [6]-[9], only selected parts of PHY or MAC layer are typically targeted while other processing still relies on dedicated hardware.

In this paper, we address the feasibility of software based implementation using VLIW processor for the real-time operation of IEEE802.11ac transmitter full PHY layer baseband processing in four different transmission scenarios which include 2×2 and 4×4 MIMO configurations. As the processing platform, stemming from the requirements for very fast processing of huge amounts of data with transmission bit rates in the order of 1Gbps, the customized VLIW processor with vector processing capabilities is used. Such a software based implementation, if found feasible, can offer more flexibility, much faster time-to-market, and highly improved possibilities to bringing in new transmission features and enhancements.

The rest of the article is organized as follows. In Section II, a detailed description of the selected transmission scenarios of 802.11ac standard is given. Then, in Section III, the employed processor and some of its main features are described. Furthermore, the software development environment and some of

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.

the employed optimization approaches are introduced. The implementation results and analysis are then provided in Section IV. Finally, the conclusions are drawn in Section V.

2. TRANSMISSION SCENARIOS

In this work, we mainly focus on the PHY layer implementation of the IEEE802.11ac standard compatible multi antenna transmitter with selected transmission modes. According to the 802.11ac standard draft version, the VHT PHY comprises of two functional entities: the PHY function and the physical layer management functions. The PHY is consisting of PHY header part and data part; the header part itself is further divided into multiple fields where L-STF, L-LTF and L-SIG are the legacy portions and VHT-SIG-A, VHT-SIG-B, VHT-STF and VHT-LTF are the very high throughput fields. For more details, refer to [1].

In order to obtain very high throughput, IEEE 802.11ac defines various core functionalities and parameters, which increase the data rate considerably. Modulation and Coding Scheme (MCS) improvements and spatial multiplexing based MIMO transmission allow VHT performance achievement. Furthermore, other solutions such as wider bandwidth, shorter GI, and higher number of spatial streams are also introduced in the amendment. Additionally, the PHY implementation plays an important role in the VHT scenario, for instance the optional usage of Low-Density Parity-Check (LDPC) encoder and Space Time Block Coding (STBC) enhance the error protection and diversity characteristics. As a result, the performance characteristics are improved compared to legacy systems; thus helping to achieve VHT targets.

Fig. 1 depicts the main structure of the implemented data part processing at the transmitter side where depending on the transmission scenario some blocks may be obsolete. The minimum time for IFFT and frequency domain processing for a single OFDM symbol is 4µs for the header part and 3.6µs for the data part when short Guard Interval (GI) is used.

In our work, we cover the implementation of four operation points (transmission modes) of the IEEE 802.11ac standard. These four operation points have some of the implementation parameters in common such as channel bandwidth and modulation scheme. The channel bandwidth is set to 80MHz, which implies 256 subcarriers (234data+14null+8pilot subcarriers). Also in all cases, 256-QAM modulation scheme is employed to map a block of eight bits into one constellation point. In this work, the operational blocks from the stream parser to IFFT are implemented in all four scenarios. It should be noted that we have assumed all the incoming bits from the LDPC encoder are already stored in the local memory, hence the required time for data transfer to the local memory is not considered. The implementation procedure of the transmitter blocks in each four scenarios will be shortly discussed in the following.

2.1. Case A: 80MHz TX/RX bandwidth, 256QAM with 3/4 coding, Short GI, 2×2 SU-MIMO

This case uses a 2×2 antenna configuration and two spatial streams, which are directly mapped to the space time streams hence removing the need for STBC coding. As defined in the IEEE 802.11ac standard, the bits received from the LDPC encoder should be directly fed to the stream parser block to be rearranged and parsed [1]. However, as tone mapping can be done more efficiently at bit-level rather than with complex numbers, in this work, this operation is placed before the stream parser and modulation blocks. Furthermore, the bits from the channel encoder are first fed to a preparation block to be rearranged for faster tone mapping and modulation. Preparation block combines the real and imaginary parts of each subcarrier, in such a way that the first outcoming 16-bit block has the real parts of the two streams, and the second 16-bit block has the imaginary parts (8 bits in each 16-bit parts are zeros). Then, the prepared streams are fed to the LDPC tone mapper, which shuffles the data subcarriers of both streams simultaneously. Next is the stream parser block after which each stream will have 234×8 bits in the following form; from left to right, the first and second 4-bit blocks present the real part of the first and second streams (first subcarrier). Respectively the third and fourth 4-bit blocks show the imaginary parts (first subcarrier) of the first and second streams and so on.

In order to obtain the most efficient performance, the stream parser and constellation mapper are merged into one function; thus the LDPC tone mapped complex numbers are parsed and mapped into the constellation points in the same function. Basically, the stream parser parses 234×8×2 coded bits per symbol into two spatial streams, i.e., each stream has 234×8 coded bits per symbol. Afterwards, a block of eight bits is mapped into one 256-QAM constellation point.

As rest of the operations, namely pilot insertion, cyclic shift diversity, spatial mapping and phase rotation are based on multiplication between the data subcarriers and coefficients, all these operations can be done at the same time. All of the possible combinations of the mentioned operations are pre-calculated and stored in a look-up table so that the operations can be executed with a single multiplication per subcarrier.

2.2. Case B: 80MHz TX/RX bandwidth, 256QAM with 3/4 coding, Short GI, 4×4 SU-MIMO

This transmission mode has four spatial streams to be mapped into a 4×4 antenna configuration and thus due to the equal number of space time and spatial streams, STBC is obsolete. As mentioned in the previous subsection, the incoming bit streams should be rearranged for faster implementation. But in this case, as four spatial streams are used, the first incoming 16 bits already feature the real part of the first subcarrier of each stream, and the second 16 bits are the imaginary part. Therefore, the bits are arranged in the desired 16-bit format. Thus, no preparation is needed and the bit streams can be directly fed into the LDPC tone mapper.

After shuffling the data bits, the stream parser rearranges the bits and allocates them to the four streams and then every 8 bit block is mapped into one constellation point in the same function. Pilot insertion, cyclic shift diversity, spatial mapping, and phase rotation, similar to the previous case, are performed by multiplication using the look up table values as coefficients.

2.3. Case C: 80MHz TX/RX bandwidth, 256QAM with 3/4 coding and STBC, Short GI, 2×2 antenna configuration with 1×1 SU-SISO transmission

As the title indicates, in this case, the number of the spatial streams is less than the number of space time streams which means STBC implementation is needed in addition to the other blocks employed in the previous cases. As defined in the IEEE 802.11ac standard, producing the even numbered space time streams includes conjugation and negation of the symbols in the odd numbered space time streams. Since conjugation is basically negating the imaginary part of a complex number, this operation can be easily done at bit level by simply inverting the sign bit. Therefore, STBC block is also moved to the preparation block in this work.

In the next phase, the STBC encoded bits are fed to the LDPC tone mapper to be shuffled. Then the stream parsing and constellation mapping are applied to both space time streams, and finally in the last stage, pilot insertion, cyclic shift, spatial mapping and phase rotation are done at once.

2.4. Case D: 80MHz TX/RX bandwidth, 256QAM with 3/4 coding and STBC, Short GI, 4×4 antenna configuration with 2×2 SU-MIMO transmission

In the last transmission mode, there are two spatial streams and a 4×4 antenna configuration, which means STBC shall be applied. Similar to the previous scenario, the STBC creates four space time streams from two spatial streams in the preparation block. Afterwards, the bit streams will be LDPC tone mapped in the next block and then go through stream parser/constellation mapper block simultaneously. As described in the previous scenarios, the final block performs pilot insertion, cyclic shift, spatial mapping and phase rotation.

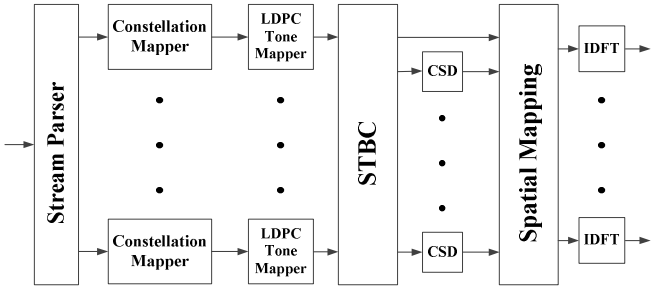

Fig. 1. Principal block diagram of transmitter baseband processing.

3. VLIW ARCHITECTURE AND IMPLEMENTATION

Due to huge amount of information processed, a customized VLIW processor with vector extensions is used as the platform to implement the transmitter baseband processing functions in this work. The adopted DSP core, Tensilica ConnX BBE32 [10], is a high performance, very small size and ultra-low power consumption DSP core that has been specifically designed for use in the cost and power sensitive baseband modem systems [10]. This DSP core is a 4-issue VLIW processor and has support for vector operations with the aid of a 16-way SIMD ALU engine and 32-way MAC SIMD engine. In addition, the processors can access wide data chunks from memory in blocks of 256 bits. Fig. 2 illustrates the general architecture of the ConnX BBE32 core.

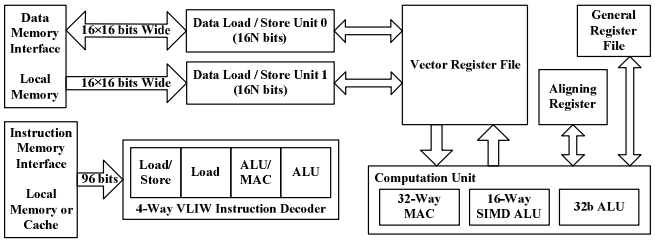

Fig. 2. ConnX BBE32 architecture.

Additionally, this DSP engine is equipped with dedicated hardware accelerator blocks to off-load computationally intensive operations such as FFT/IFFT [11]. The processor is configurable and special function units can be added to speed up the computations. We used two different processor configurations of the ones provided by the vendor: Low-Power (LP) and Performance-Maximize (PM) configurations. The LP is the baseline configuration and the PM configuration provides instruction extensions for accelerating various functions, e.g., FFT, FIR filtering, bit mapping etc. The processor has Harvard architecture with one instruction memory and two data memories.

Tensilica uses an Eclipse-based software development environment (Xtensa Xplorer), which provides a comprehensive collection of code generation and analysis tools. This tool enables software development to be carried out using C programming. However, for optimization purposes, nearly all of the software implementation in this work is done by heavily using the provided processor intrinsics. In spite of the automatic vectorization capability of the compiler, the code was vectorized manually for better performance.

As mentioned earlier, correct configuration and programming play an important role in the efficiency of the implementation, thus some optimization approaches have been applied in this work to fasten the processing. One very effective optimization approach was merging and combining the functions as much as possible. Specifically the operation blocks whose functionality involves multiplication with a constant (such as phase rotation, spatial mapping, pilot insertion and cyclic shift diversity) can be easily merged. Moreover, since it is easier to deal with bits rather than complex numbers, as many operations as possible have been implemented before the constellation mapping. For instance, although in the standard and as shown in Fig. 1, the STBC and LDPC tone mapper blocks are defined to be employed after the constellation mapping, it has been observed that such operations can be implemented more efficiently when the data is still in bits and not yet modulated to symbols.

4. RESULTS AND ANALYSIS

The described software based implementation was profiled and analyzed with the aid of the tools provided by the vendor. The results related to number of clock cycles, power, and memory usage are presented in this section.

The numbers of clock cycles were obtained with the instruction set simulator and profiling tools. In Fig. 3, the numbers of clock cycles needed to process one OFDM symbol in LP configuration are presented. PM configuration requires larger number of cycles in comparison with the LP model but the difference is only 1-2 %.

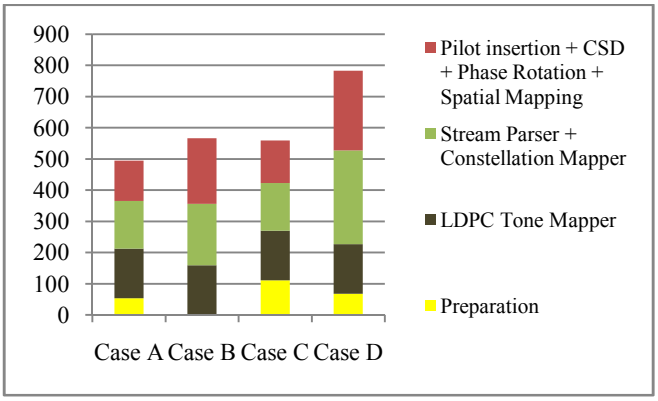

Fig. 3. LP model clock cycles results for all the cases.

As mentioned earlier, in different transmission scenarios, different blocks may operate; therefore the results are given for each block in each transmission scenario. It should be noted that for the Cases C) and D), the preparation block also includes the STBC coding operation.

As previously mentioned, the duration of an OFDM symbol is 4μs for the header part and 3.6μs for the data part when short Guard Interval (GI) is used. Thus to achieve real-time operation in the transmitter, all the processing needed to create one OFDM symbol should not take more than 3.6μs. Assuming a 500 MHz operating frequency, 3.6μs can accommodate 1800 clock cycles. Looking at the total number of clock cycles for each transmission scenario from Fig. 3, it can be concluded that the system operations can be computed in real-time in this implementation.

One of the most important evaluation criteria for the implementation is the power consumption, which is directly dependent on the memory configuration/capacity. As the vendor provides Energy Xplorer for energy consumption estimation, first energy usage is profiled and then power consumption is calculated by dividing the energy values by time. The time for each block is defined by the number of clock cycles. We have considered two common cases, which are the maximum (128k) and half (64k) of the memory capacity. The energy consumption was estimated by exploiting technology libraries for a 40nm low-power IC technology provided by the tool vendor. We also assume clock frequency of 500MHz. The monitoring time for the energy analysis is 3.6μs, and it includes both leakage and dynamic parts. Fig. 4 reveals the results related to the power consumption for the LP model, in case of full and half memory usage. The power consumption results for the PM model are not presented as there is no significant difference with the LP model. In general, the power consumption estimates are found feasible to mobile terminal scale devices.

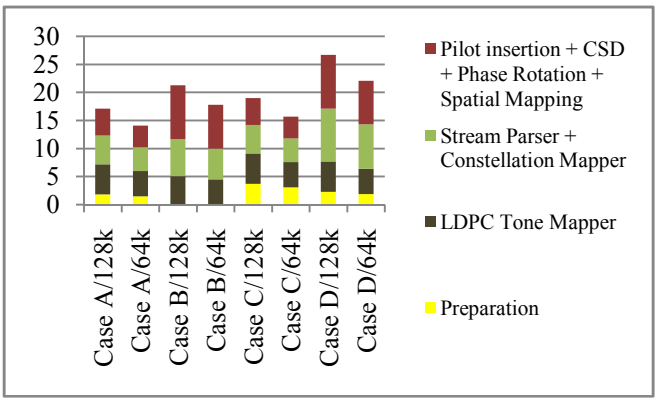

Fig. 4. Power consumption in mW for 128k and 64k memory capacities.

In order to avoid stalls and keep the pipeline full, loop unrolling was heavily exploited in some of the functional blocks such as LDPC tone mapper and the block including pilot insertion, spatial mapping, CSD and phase rotation to exploit parallelism among the instruction. As loop unrolling increases the program code size, to evaluate how much memory is needed for the developed software, instruction memory usage for each operation in each transmission scenario was measured and is presented in Table 1. Since there is no difference between PM and LP model from the memory usage point of view, only the results when using the PM model are presented. In order to get more informative realization on the effect of loop unrolling on the program code size, the code density was also calculated for all the functional blocks. The average code density over all blocks is 52.95%.

In addition to the instruction memory usage, data memory usage was evaluated and is presented in Table 1. It should be noted that the amount of used data memory does not depend on the transmission scenario except for the input buffer usage.

Table 1. Memory usage in bytes

	Case A	Case B	Case C	Case D
Instruction memory				
Preparation	184	---	376	240
LDPC Tone Mapper	1808	1808	1808	1808
Stream Parser + Constellation Mapper	432	200	428	528
Pilot insertion + CSD + Phase Rotation + Spatial Mapping	720	720	720	720
Total	3612	3664	3800	4232
Data Memory				
Local Data RAM #1	4.8 K			
Local Data RAM #2	5.128 K			
Input Buffer	468	936	468	936
Total	10.396K	10.864K	10.396K	10.864K

To achieve higher performance and faster processing, the numerical values of the cyclic shift diversity for different streams, spatial mapping and phase rotation operations were calculated and stored in a look-up table. This look-up table takes 128 and 800 bytes from the local data RAM #1 and local data RAM #2 memories, respectively.

5. CONCLUSIONS

In this paper, we developed software-based implementation of the IEEE 802.11ac transmitter full frequency-domain PHY layer baseband processing for four different multi-antenna transmission scenarios. We have evaluated the solution by profiling and analyzing the implementation using the tools provided by the vendor. We have presented the results with regards to number of clock cycles, power consumption, and memory usage. The analysis of the performance numbers clearly shows that the developed software based implementation on a DSP core can achieve real-time operation for the transmitter baseband processing assuming 500 MHz clock frequency. Furthermore, the implementation resulted in realistic power consumption and memory usage, despite of massive amount of data processing yielding beyond 1Gbps transmission bit rate in the most ambitious transmission scenario. The future work will focus on implementing the corresponding receiver chain PHY processing, which includes more complex functions such as channel state estimation and detection.

6. REFERENCES

[1] IEEE P802.11ac™ Draft Standard, version 5, January 2013.

[2] E. Perahia and R. Stacey, *Next Generation Wireless LANs*, Cambridge, NY, 2013.

[3] S. Yoshizawa and Y. Miyanaga, "VLSI Implementation of a 4×4 MIMO-OFDM transceiver with an 80-MHz channel bandwidth," in *Proc. IEEE ISCAS*, Taipei, Taiwan, 24-27 May 2009, pp. 1743-1746.

[4] S. Samadi, A. Golomohammadi, A. Jannesari, M.R. Movahedi, B. Khalaj, and S. Ghammanghami, "A Novel

Implementation of the IEEE802.11 Medium Access Control," in *Proc. Int. Symp. Intelligent Signal Process. Commun.*, Yonago, Japan, 12-15 Dec. 2006, pp.489-492.

[5] 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.

[6] K. Rounioja, and K. Puusaari, "Implementation of an HSDPA Receiver with a Customized Vector Processor," in *Proc. Int. Symp. System-on-Chip*, Tampere, Finland, 13-16 Nov. 2006, pp.1-4.

[7] W. Xu, M. Richter, M. Sauermann, F. Capar, and C. Grassmann,, "Efficient baseband implementation on an SDR platform," in *Proc. Int. Conf. ITS Telecommunications*, St. Petersburg , Russia, 23-25 Aug. 2011, pp.794,799.

[8] J. Janhunen, T. Pitkänen, M. Juntti, and O. Silvén, "Energy-efficient programmable processor implementation of LTE compliant MIMO-OFDM detector," in *Proc. IEEE ICASSP*, Kyoto, Japan, 25-30 March 2012, 3276 – 3279.

[9] S. Eberli, A. Burg, and W. Fichtner, "Implementation of a 2×2 MIMO-OFDM receiver on an application specific processor," *Microelectronics Journal*, vol. 40, no. 11, pp. 1642-1649, November 2009.

[10] Tensilica Inc., *ConnX BBE32 DSP User Guide*, USA, 2012.

[11] Tensilica Inc., *ConnX BBE32 DSP Core for Baseband Processing,* USA, 2013.