Parallel Processing Intensive Digital Front-End for IEEE 802.11ac Receiver

Mona AghababaeeTafreshi, Juha Yli-Kaakinen, Toni Levanen, Ville Korhonen, Pekka Jääskeläinen,
Markku Renfors, Mikko Valkama, and Jarmo Takala
Tampere University of Technology, P.O. Box 553, FI-33720 Tampere, Finland
email: mona.aghababaeetafreshi@tut.fi

***Abstract*—Modern computing platforms offer increasing levels of parallelism for fast execution of different signal processing tasks. In this paper, we develop and elaborate on a digital front-end concept for an IEEE 802.11ac receiver with 80 MHz bandwidth where parallel processing is adopted in multiple ways. First, the inherent structure of the 802.11ac waveform is utilized such that it is divided, through time-domain digital filtering and decimation, to two parallel 40 MHz signals that can be processed further in parallel using smaller-size FFTs and, e.g, legacy 802.11n digital receiver chains. This filtering task is very challenging, as the latency and the cyclic prefix budget of the receiver cannot be compromised, and because the number of unused subcarriers in the middle of the 80 MHz signal is only three, thus necessitating very narrow transition bandwidth in the deployed filters. Both linear and circular filtering based multirate channelization architectures are developed and reported, together with the corresponding filter coefficient optimization. Also, full radio link performance simulations with commonly adopted indoor WiFi channel profiles are provided, verifying that the channelization does not degrade the overall link performance. Then, both C and OpenCL software implementations of the processing are developed and simulated for comparison purposes on an Intel CPU, to demonstrate that the parallelism provided by the OpenCL will result in substantially faster realization. Furthermore, we provide complete software implementation results in terms of time, number of clock cycles, power, and energy consumption on the ARM Mali GPU with half precision floating-point arithmetic along with the ARM Cortex A7 CPU.**

***Keywords*—WLAN, IEEE 802.11ac, Multirate Filtering, Digital Front-End, Graphics Processing Units, Open Computing Language, Parallel Processing.**

I. Introduction

Software-based implementations of radio transceiver digital front-end (DFE) and baseband (BB) processing stages are receiving increasing interest, due to substantially enhanced re-configurability and reduced time-to-market cycles, when compared to classical fixed-function digital hardware implementations [1][2]. Modern platforms have increased parallel computation capabilities due to the limits of improving performance by means of increasing the clock frequency. Multi-core processors and graphics processing units (GPU) along with programming standards such as OpenCL enable software developers to explicitly utilize the parallelism for faster processing [3].

In this paper, we address the DFE processing of the flagship WLAN/WiFi technology, namely IEEE 802.11ac [4], where the basic radio access is based on 80 MHz instantaneous bandwidth. Interestingly, this 80 MHz access waveform is composed by essentially aggregating two 40 MHz sub-signals [4], stemming from the legacy IEEE 802.11n access bandwidth, with three null subcarriers (approximately 1 MHz) inbetween. In the digital front-end concept proposed in this paper, this overall 80 MHz signal is divided to two 40 MHz sub-signals, through carefully optimized time-domain filtering, which in turn can then be processed forward in parallel, with two smaller-size FFTs and corresponding frequency-domain processing. This overall receiver principle, assuming also wideband I/Q downconversion from RF to baseband, is depicted at conceptual level in Fig. 1. This can be also extended to a 160 MHz signal being divided into four 40 MHz sub-signals from which each can be received by a modified 40 MHz 802.11n receiver. However, this filtering task is far from trivial, as the cyclic prefix (CP) budget of the overall wireless link, including filtering in the devices, should not be compromised, since the latency requirements of the 802.11ac receiver are very tight [4], and because the small spectral gap of around 1 MHz calls for very narrow transition bandwidth in the filter optimization. Hence, in this paper, we first address this channelization filter optimization task, and report both linear digital filtering and circular digital filtering based multirate solutions with different characteristics and tradeoffs related to latency, filtering performance, and CP budget. Essentially, the circular filtering based solution slightly increases the latency but does not compromise the CP budget at all, being implemented after the CP removal, just prior to the parallel FFT units. We also provide full radio link simulation results, with commonly adopted WiFi indoor channel models, to verify that the overall channelization filtering does not degrade the link performance.

Then, related to the actual software-based processing implementations, we have developed both C and OpenCL-based solutions on the Intel® Core™ i7-4800MQ CPU [5] to demonstrate that the explicit parallelism provided by the OpenCL framework will result in substantially faster execution. We also provide complete software implementation results using the Odroid XU3 [6]. The Odroid XU3 is based on the Samsung Exynos 5 Octa, powered by ARM Cortex™-A15 quad core and Cortex™-A7 quad core CPUs, which employs the ARM® big.LITTLE™ technology [7][8][9]. This technology creates a multicore processor which couples relatively slower processor cores with more powerful ones. The XU3 also features the ARM® Mali™-T628 MP6 GPU [10] with half precision floating-point arithmetic. Different filter designs are implemented and assessed in terms of execution time, number of clock cycles, power, and energy consumption.

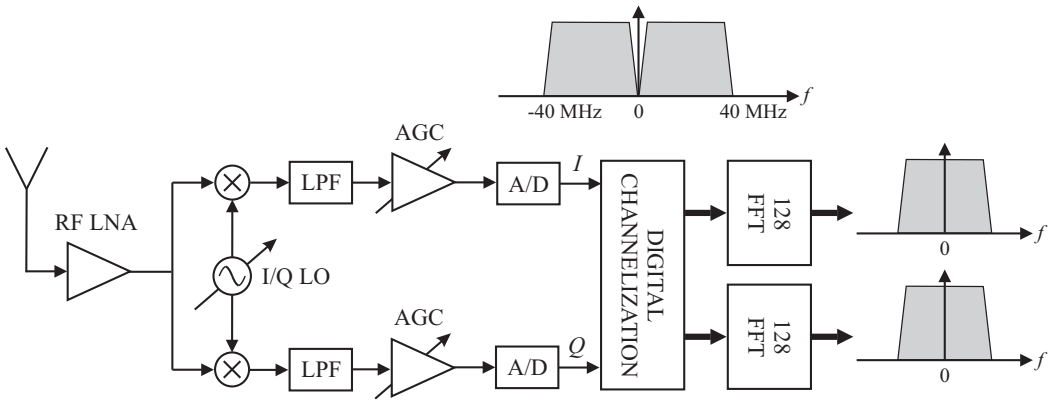

Fig. 1. The overall receiver principle with digital channelization filtering yielding two 40 MHz sub-signals.

The rest of the paper is structured as follows. First, in Section II, the channelization filtering architectures based on linear and cyclic half-band multirate filters, together with corresponding filter optimization, are described. Then, in Section III, we provide comprehensive link performance evaluations, with and without channelization filtering, to verify and demonstrate that the optimized filtering solutions reported in Section II do not essentially degrade the link performance in any way. In Section IV, the software implementation and OpenCL kernel designs are described. Finally, the results from the GPU and the two CPUs are reported in Section V, and Section VI concludes the work.

II. Channelization Filter Architectures for IEEE 802.11ac

In this work, 80 MHz access bandwidth in IEEE 802.11ac system consisting of 256 subcarriers is considered. 242 subcarriers out of the total 256 are active (234 for data and 8 for pilots). Three subcarriers around DC (subcarriers $-1, 0, 1$) are zero and both the negative and positive frequency components contain 121 transmission subcarriers (subcarriers $\pm k$ for $k = 2, 3, \ldots, 122$) [4]. In the IEEE 802.11 standards [11], the total multicarrier symbol duration is defined as 4 μs; 20 percent of this duration (800 ns) is the guard interval which carries the cyclic prefix of the signal. For FFT size of $L = 256$ this corresponds to the cyclic prefix of 64 samples. As described already in the Introduction, the goal is to divide the 80 MHz IEEE 802.11ac signal sampled at the Nyquist rate into two 40 MHz-wide signals using linear filtering such that the positive frequency components are separated into one signal and negative frequency components into a second. These are then processed further, in parallel, with two 128 point FFTs and subsequent subcarrier level processing.

A. Polyphase Halfband Filters

The problem stated above can be solved either using finite-impulse response (FIR) or infinite-impulse response (IIR) *analytical filters* [12], [13]. For FIR case, the analytical filter requiring minimum number of multiplier values can be derived with the aid of halfband filters. The transfer function of the halfband lowpass-highpass FIR filter pair can be realized efficiently as a parallel connection of *Type II* (odd-order symmetric) FIR transfer function $H(z^2)$ and a delay of M

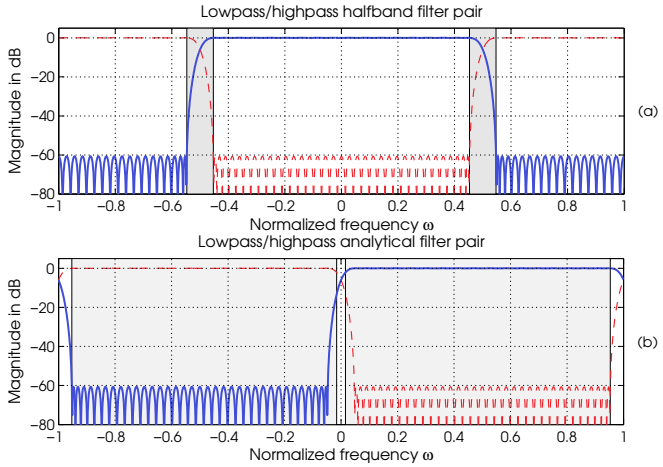

Fig. 2. Magnitude responses of the (a) halfband and (b) analytical filter pairs. In (a), the gray areas indicate the transition bands of the prototype filter pair. In (b), the gray areas indicate the 40 MHz sub-bands containing active subcarriers.

as expressed in [14].

$$G(z) = H(z^2) \pm 1/2z^{-M}. \quad (1)$$

Here, M is an odd integer such that the order of the overall transfer function $G(z)$ is $N = 2M$. The lowpass (highpass) filter is realized using the above transfer function with the plus (minus) sign.

The analytical filter is obtained from $H(z)$ by multiplying the impulse response values $h(n)$ by j^{-n}. This corresponds to shifting the frequency response of the filter by $\pi/2$. The parallel connection of the resulting Hilbert transformer (more precisely, the approximation of it) and a delay can be used for forming analytical signals, that is, for separating the positive and negative frequency components as desired. Fig. 2(a) and Fig. 2(b) show the magnitude response of the lowpass-highpass halfband and analytical filter pairs, respectively. The active subcarriers in Fig. 2(b) are denoted by the gray area. The resulting output signals can be decimated by two, if desired, by sharing the input samples into these two branch filters such that odd samples go to one branch and even samples to another. In this case, the branch filters [$H(z)$ and $1/2z^{-M/2}$] work at the output sample rate, that is, at the half of the input rate.

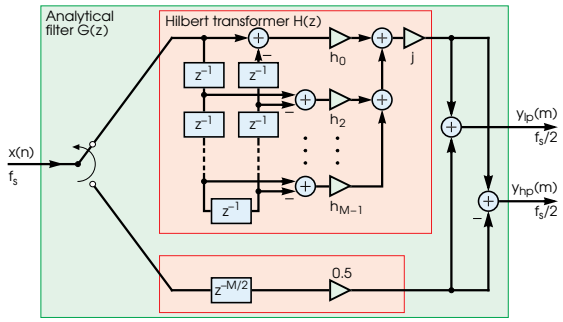

Fig. 3. Efficient processing structure of the decimating analytical filter realizing both the lowpass and highpass outputs $y_{\text{lp}}(m)$ and $y_{\text{hp}}(m)$, respectively.

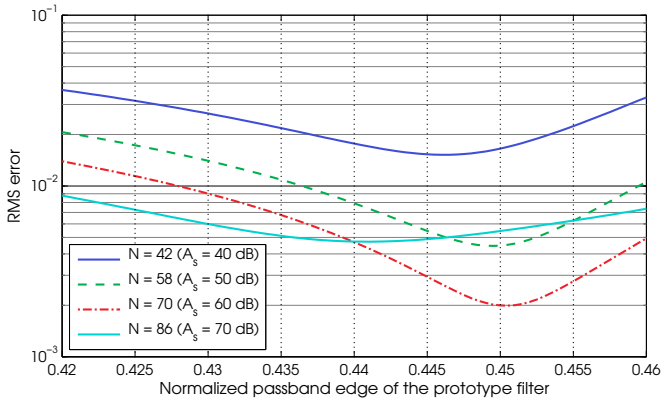

Fig. 4. RMS error between the received and the transmitted symbols as a function of passband edge for halfband FIR filters of order 42, 58, 70, and 86.

A more detailed structure of this analytical filter is shown in Fig. 3. A pair of these filters is required for filtering both the real and imaginary parts of the input signal.

When decimating the resulting lowpass and highpass filtered signals, the residue of the active negative (positive) subcarriers alias above positive (negative) subcarriers, i.e., subcarriers $-k$ for $k = 2, 3, \ldots, 122$ alias above subcarriers $128 - k$ for $k = 2, 3, \ldots, 122$. Consequently, the stopband edge of the lowpass analytical filter has to be $\omega_s = (128 - 122)/128\pi = 0.046875\pi$ to prevent aliasing into positive active subcarriers. Correspondingly, the passband and stopband edges of the prototype halfband filter are $\omega_p = 1/2\pi - (128 - 122)/128\pi = 0.453125\pi$ and $\omega_p = \pi - 0.53125\pi$ as the passband and stopband edges of the prototype halfband filter are located symmetrically around $\pi/2$ as $\omega_s = \pi - \omega_p$ for $\omega_p < \pi/2$ [12]. The gray areas in Fig. 2(a) denote the transition bands of the prototype filter pair.

The magnitude of the aliasing components is defined by the stopband attenuation of the prototype filter. Due to the properties of the prototype halfband filters, the order of the transfer function is restricted to be $N = 2 + 4k$, where k is an integer [12]. The performance of the analytical filter (parallel connection of Hilbert transformer and delay) is evaluated by measuring the root-mean-square (RMS) error of the received channelized signals as a function of the passband edge and filter length. In this simulation, the frequency response of the channelization filter is equalized per channel and 16-QAM subcarrier modulation with 1000 symbols are used. As can be seen from Fig. 4, the best RMS error performance is obtained using the filter of order $N = 70$ for which the passband edge is located at $\omega_p = 0.4505\pi$. The difference between the derived passband edge ($\omega_p = 0.453125\pi$) and the value obtained by simulation can be explained by the distribution of the zeros at the stopband. By optimizing the locations of the zeros in z-domain, their contribution to the attenuation at the exact carrier frequencies can maximized if desired. However, in real environment the difference would be negligible due to, e.g., the possible carrier-frequency offset.

B. Cyclic Polyphase Halfband Filters

The previous linear digital filtering based channelization increases the effective time dispersion of the received signal, and thus partially compromises the CP budget of the receiver. A straightforward way to tackle the increase in the overall impulse response length is to perform the channelization processing using cyclic convolution instead of linear convolution (conventional FIR filter). The basic idea is to carry out the linear convolution block-wise for the received data and then cyclically add the last N samples from the resulting N+L samples long sequence to the beginning of the block as depicted in Fig. 5. As a consequence, due to duality of cyclic convolution in time-domain and multiplication in frequency-domain, the effect of the channelization filter can be exactly equalized. Furthermore, as the cyclic convolution processing can be carried out *after removing the CP*, this solution does not contribute in any way to the effective time dispersion in the signal.

In this case, only the FFT size and the computational complexity restrict the length of the channelization filter. The computational complexity of cyclic realization is approximately 25 percent lower for the same filter order since the CP can excluded before channelization. It should be pointed out that the same polyphase filter channelization architecture can be used for both the linear and cyclic convolution.

III. 802.11AC LINK PERFORMANCE EVALUATIONS

In order to verify that the overall channelization filtering does not degrade the 802.11ac link performance, extensive link simulations are carried out. Standardized WLAN/WiFi channel models D and F [15], [16], are used to simulate the link performance of the two proposed channelization architectures in the case of frequency selective fading channel. Table I shows the delay spread and cluster parameter values of these channel models. These two channel models can be considered to represent the environments with little-to-moderate frequency selectivity, as it is typically the case in indoor offices and houses (channel model D), and moderate-to-large frequency selectivity, common in large indoor spaces such as airport and conference centers (channel model F). The symbol error rate (SER) and error vector magnitude (EVM) performance of the channelization architectures are evaluated in two cases. In the first case, the performance is evaluated as a function of signal-to-noise ratio (SNR) whereas in the second case as a function of co-channel signal-to-interference ratio (SIR). For SNR simulation, the SER and EVM evaluation is carried out with both the perfect timing synchronization as well

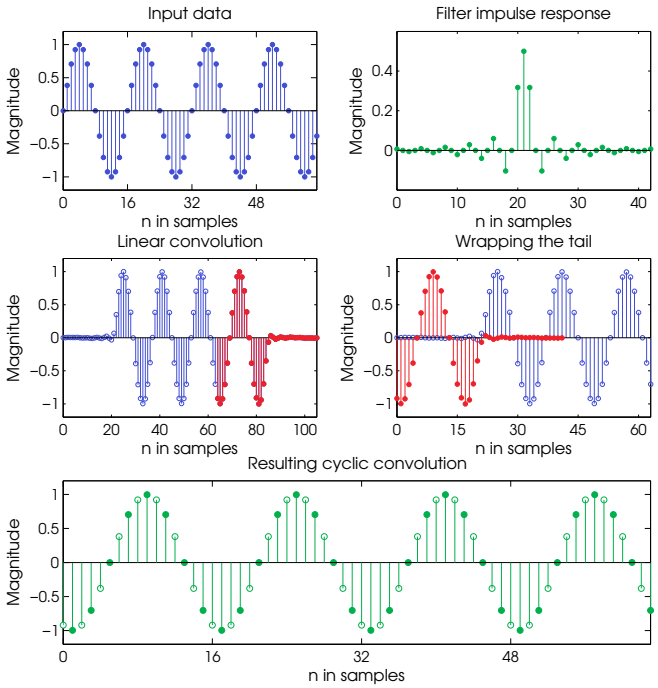

Fig. 5. Illustration of cyclic convolution using linear halfband filter.

as an example timing synchronization error of 8 samples. Three different prototype filters are used for the linear filter channelization case. The stopband attenuations of these filters are 40 dB, 50 dB, and 60 dB. For circular filter case, only one prototype filter is used with the stopband attenuation of 40 dB. In addition, SER and EVM performance is evaluated in the case with no channelization for reference purposes.

In the case of SIR simulations, the co-channel interference is a complex exponential having a random frequency inside the negative frequency band and the error functions are evaluated over the positive active subcarriers. In this case, only the perfect time-synchronization case is simulated. In all simulations, the number of random channel instances is 1000 whereas the number of 16-QAM modulated OFDM symbols is equal to 100.

The simulated SER and EVM as a function of SNR are shown in Fig. 6 whereas the corresponding SIR results are shown in 7. As can be seen from these figures, in the case of SNR simulation, the performance of the circular filter architecture is approximately the same as with no channelization. In the case of SIR simulation, the linear filter architecture slightly outperforms the circular filter with Channel model D, whereas in the case of Channel F, the circular architecture results in considerably better SER and EVM values. This is because the circular filtering based channelization architecture does not reduce the CP budget of the receiver in any way. However, as the adjacent channel rejection in IEEE 802.11 RF front-ends should be at least 40 dB, the SIR performance of the circular filter also with little-to-moderate frequency selectivity of the Channel D can be considered to clearly meet the requirements.

TABLE I. DELAY SPREADS AND CLUSTER PARAMETERS OF INDOOR TGN AND TGAC SPATIAL CHANNEL MODELS [15], [16]

Model	Scenario	RMS delay spread	Number of clusters	Taps/cluster
D	Indoor typical office	50 ns	3	16,7,4
F	Large indoor space	150 ns	6	15,12,7,3,2,2

IV. SOFTWARE IMPLEMENTATION

Three different platforms were employed for the implementation of the channelization task described in the previous sections. Firstly, C and OpenCL implementations were carried out on the Intel® Core™ i7-4800MQ CPU which has 4 cores and runs at base frequency of 2.7GHz and turbo frequency up to 3.7GHz. This step was done with the purpose of demonstrating the speedup achieved as a result of using OpenCL compared to C. Then, to take advantage of the parallel computing ability of GPUs, the implementation was additionally carried out on the ARM® Mali™-T628 MP6 GPU [10]. Mali-T628 is a part of the Samsung Exynos 5 Octa (Exynos 5422) mobile System on Chip (SoC). Mali-T628 offers scalability from one to eight cores and runs at a frequency of 600 MHz. This GPU also provides support for half precision floating point arithmetic. Half-precision floating numbers are defined by the IEEE 754 standard to have 16 bits consisting of five bits for the exponent, 10 bits for the fraction and one bit for the sign [17]. The usage of half floats could possibly reduce the execution time, power, and energy consumption to some extent. Exynos 5422 is also equipped with two CPUs using the big.LITTLE heterogeneous computing architecture [9]. The two CPUs are ARM® Cortex®- A15™ and A7™ [7][8]. A15 and A7 are quad core CPUs and can run at up to 2.1GHz and 1.5GHz, respectively. The ARM big.LITTLE architecture aims at achieving high performance while improving the power efficiency by coupling a performance driven ”big” core with a power efficiency driven ”LITTLE” core. Thus, these CPUs were also taken into consideration for the implementation. In this work, we have used ODROID-XU3 [6] to utilize the Samsung Exynos 5422 SoC for the implementation.

Various approaches were considered to implement the channelization filter in OpenCL, designed carefully to utilize the available parallelism. Two of the approaches which proved to be most efficient are described in the following. Different filter designs and software implementations are considered in each solution. The first solution focuses on a halfband filter design with higher length and lower number of required arithmetic operations, due to the zero coefficients, compared to the second solution. The second one, however, uses a non-halfband filter design with shorter length and utilizes vector operations. These implementations are carried out for both linear and cyclic filter designs.

A. Halfband Filter Without Vectorization

First approach implements the linear and cyclic halfband filters described in Section II. This implementation takes advantage of the fact that every other coefficient in the filter design is zero (as illustrated in Fig. 5), thus reducing the number of required multiplications by a factor of two. Moreover, having symmetric coefficients helps simplify the implementation further by first subtracting the pair of samples having the same coefficient values and then multiplying the resulting difference

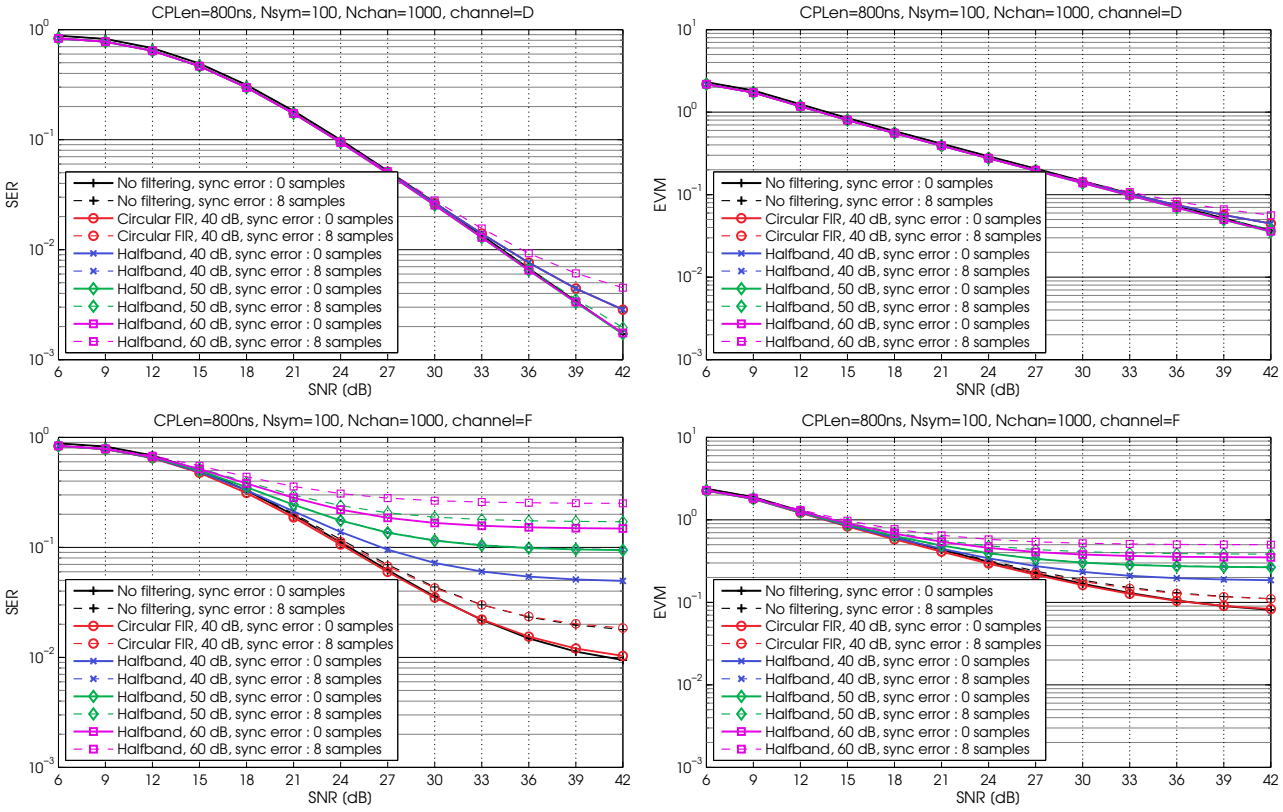

Fig. 6. SER and EVM as a function of SNR for conventional and circular halfband filters with channel models D and F.

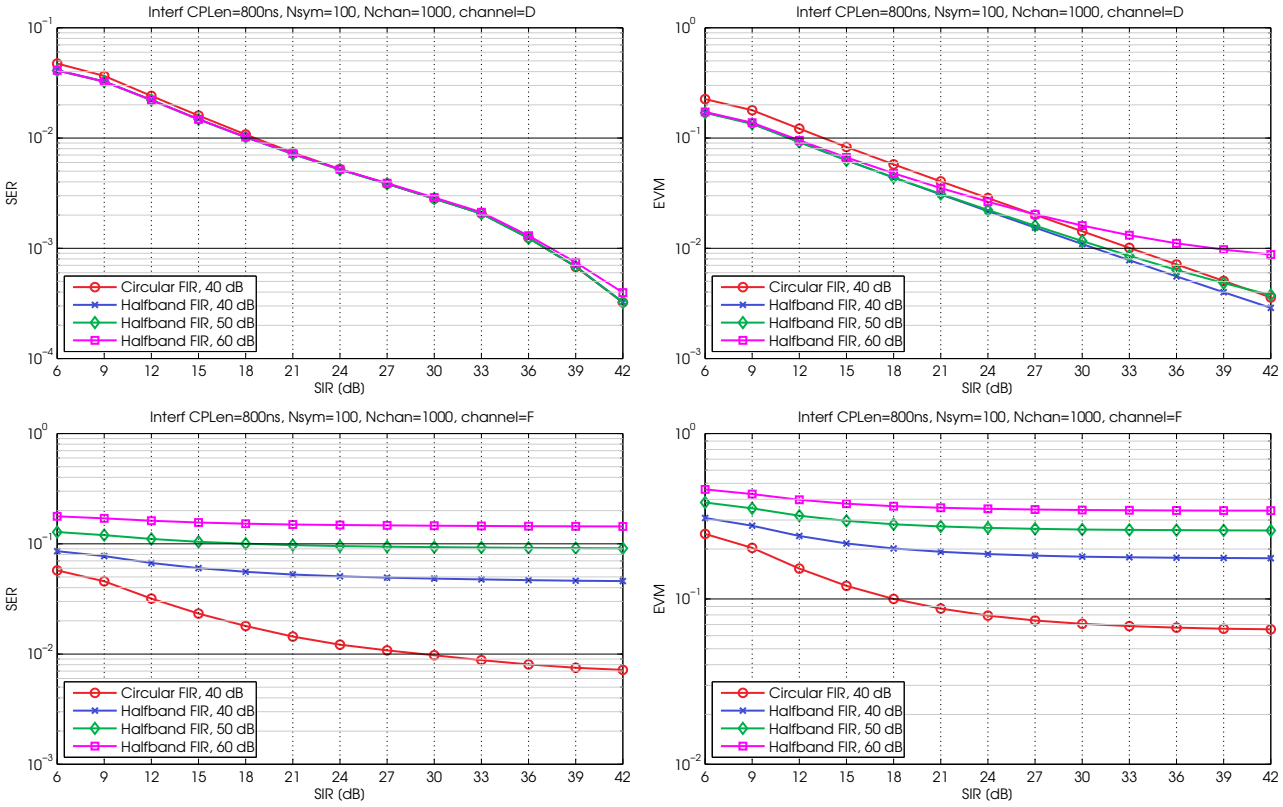

Fig. 7. SER and EVM as a function of SIR for conventional and circular halfband filters with channel models D and F.

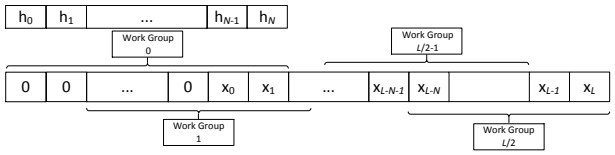

Fig. 8. The structure for the implemented halfband kernel, x denotes input samples, h denotes filter coeficients, L is the number of input samples, and N is filter order.

only once. Also both the lowpass and highpass filters can be realized at the same cost. In this implementation, it is assumed that all the input samples of one OFDM symbol and the filter coefficients are fed to the kernels input buffers. Fig. 8 illustrates the distribution of the computations among work groups, having L and N representing the number of subcarriers in one OFDM symbol and the filter order, respectively. To start the parallel computations, it is assumed that $N+L-1$ samples are stored in the input buffer, $N-1$ of which are zeros, added at the beginning of the input stream. As shown in Fig. 8, in this implementation, $L/2$ work groups work simultaneously to multiply the vector coefficients with the input samples and do the summations. Consequently, each work group produces one lowpass and one highpass output at the same time with other work groups.

B. Non-halfband Filter with Vectorization

In software based solutions, an important aspect to consider aside from the number of arithmetic operations is the memory accesses. In case of a halfband filter implementation, having zero coefficients, every other sample is skipped, which reduces the number of required multiplications. However, having these erratic memory accesses in the halfband filter could be less efficient than executing all multiplications instead of half. Thus, in the second approach, a non-halfband filter is considered for the channelization. As the cores support Single Instruction Multiple Data (SIMD) operations, an efficient implementation for the filter could be carried out using the OpenCL vector operations. The highest number of allowed vector components in OpenCL is 16. For this reason, the optimum designed filter should have a length that is multiple of 16. In general, the realization of odd-order (even length) two-channel FIR filter bank is not desirable. This is due to the reason that the resulting polyphase branch filters are non-symmetric filters, resulting in a quadruple complexity compared with the original half-band design. Therefore, the filter length is chosen to be $16n-1$. Then the length has been increased to $16n$ by padding one zero at the end of the impulse response.

In this kernel design, input samples and filter coefficients should be in the form of vectors of length 16. Fig. 9 depicts the arrangement of work groups and work items in this implementation. S is the number of subcarriers in one OFDM symbol plus N zeros added for filtering. $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{S/16}$ are vectors of length 16 containing S samples altogether. As it is illustrated in Fig. 9, each work group operates on a number of vectors. Then inside each work group, each work item, according to its work item number, carries out the processing related to a part of the vectors corresponding to that work group. This processing includes the multiplication of the data samples by coefficient values and the final summation.

V. Results and Analysis

To evaluate the performance enhancement achieved by exploiting parallelism using OpenCL, we have measured the execution time and number of clock cycles consumed when executing the filters both using C and OpenCL. This preliminary step was carried out on the Intel® Core™ i7. Then to study the advantages and disadvantages of different multicore platforms, the channelization filter was additionally implemented on the ARM® Mali™-T628 and the ARM® Cortex®- A7™ CPU. In addition to time and number of clock cycles, power and energy consumption were measured on the ARM platforms using the sensors available on the Odroid XU3. Most importantly, the performance improvements obtained by the application of half precision floating point arithmetic on Mali was investigated and is presented in this section. In all the measurements, the number of input samples is equal to the FFT size plus the CP length and the filter order, all multiplied by two as all the samples are in complex form.

A. Execution Time

Fig. 10 shows the execution times in milliseconds for running linear and circular filters using halfband and non-halfband implementations on the different platforms introduced in Section IV. Firstly, Fig. 10 shows that the halfband filter is executed approximately 80% faster when using OpenCL rather than C. Furthermore, it can be seen that among the OpenCL implementations, the Intel Core i7 consumes the least time. The second fastest platform is the Mali GPU, and the slowest is the ARM A7 CPU. This can be explained by the Intel CPU having the highest clock frequency, up to 3.7GHz which is six times higher than Mali's and two times higher than A7's. Another important observation from the implementation results is the amount of speedup gained by using half precision floats on the Mali GPU. The results show that the application of half precision floats has lowered the execution time by at least 55% which exceeds the expected linear speedup of two. This could be explained by the fact that taking up less space for the data results in more cache hits and less memory transfers, thus causing the faster execution. As it can be seen from Fig. 10, there is less difference between the linear and circular filtering solutions in non-halfband implementations as the designed non-halfband linear and circular filters have the same filter length. However, with the halfband implementation, the circular design requires a higher filter length, thus resulting in relatively slower execution.

The latency restrictions for this channelization process originate from the duration of the defined short interframe space (SIFS) in the IEEE 802.11ac amendment. As this channelization task is carried out for 80 and 160 MHz bandwidths which are only available in 5GHz carrier, the available SIFS time is equal to $16\mu s$. The lowest possible execution time realized on the platforms used in this work is $6.02\mu s$. Taking into consideration the other related required processing, such as MAC processing, the filtering can fit in the time frame. However, to have better margins for the rest of the required processing, it is beneficial to still reduce the execution time further. Mali is a small mobile GPU and employing a faster, larger GPU can result in lower execution times for the filtering that can, more easily, meet the real time requirements. Thus,

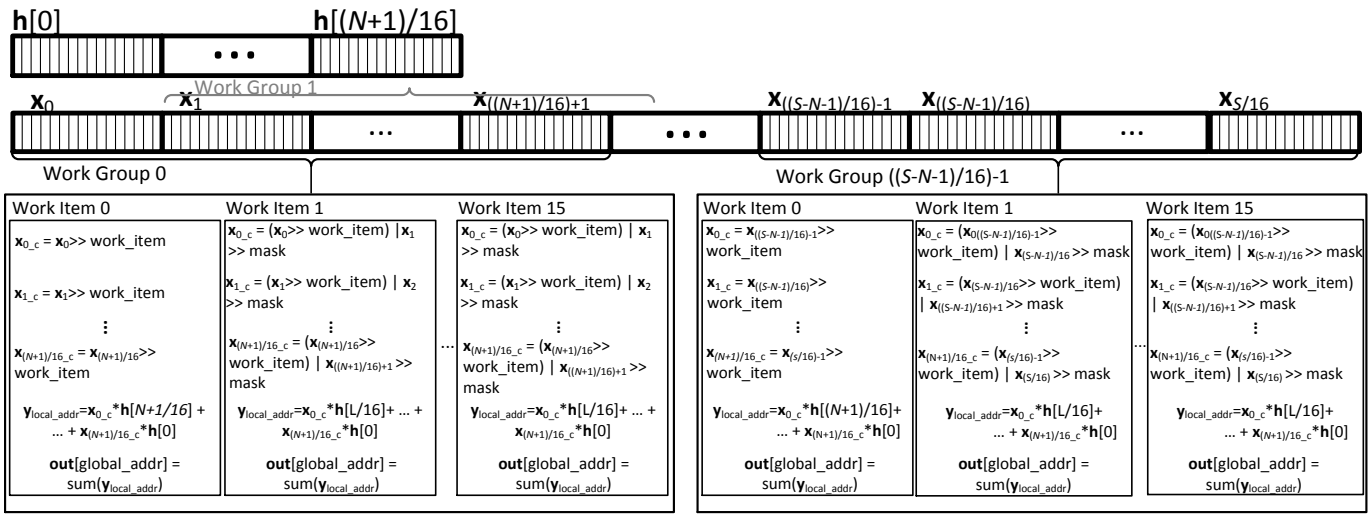

Fig. 9. The structure for the implemented non-halfband kernel, **x** denotes input sample vectors, **h** are the vectors containing filter coefficients, N is the filter order, and S is the number of input samples plus N zeros.

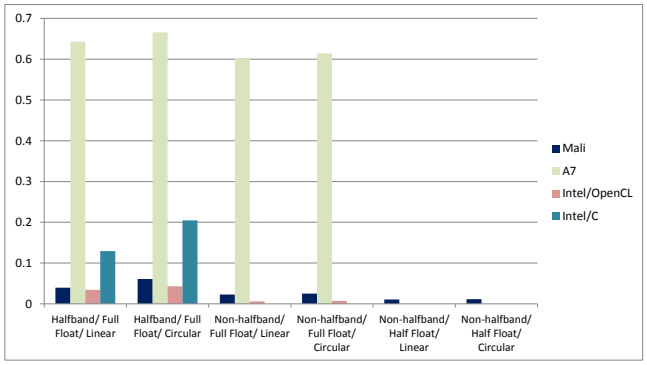

Fig. 10. Execution time in miliseconds consumed by linear and circular digital filtering using halfband and non-halfband designs on different platfroms.

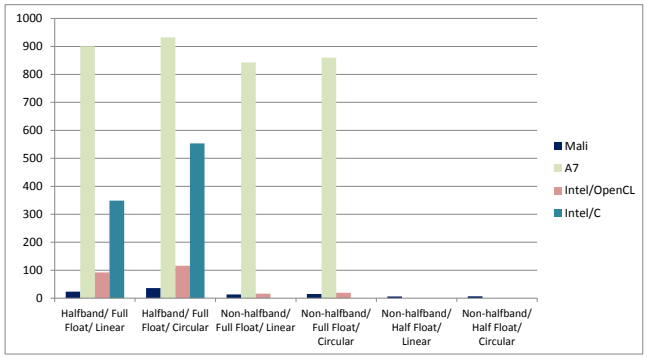

Fig. 11. Number of clock cycles consumed by linear and circular digital filtering using halfband and non-halfband designs on different platfroms.

this could be also considered as a suitable implementation candidate e.g. in an access point setting using larger GPUs.

B. Number of Clock Cycles

To calculate the number of clock cycles required for each filter implementation, the nominal frequency of the platforms was assumed. The clock frequencies considered for the Intel CPU, ARM CPU, and the Mali GPU were 2.7GHz, 1.4GHz, and 600 MHz, respectively. The calculated number of clock cycles are presented in Fig. 11. Similar to the execution times presented in the previous section, these numbers, most importantly, verify the great advantage of using half precision floats over the full precision floats.

C. Power

The Odroid is equipped with four separated current sensors to measure the power consumption of Big CPU (A15), Little CPU (A7), GPU and DRAM in real time. In this work, the measurements are carried out in a way that 200 samples are taken from the sensors in intervals of 100ms and then averaged over a 20s time period to assess the average power consumption. To achieve more precise measurements, the kernels were run in high number of iterations to keep the cores active with kernel executions during the whole 20s. We did not have any tools available to measure the power consumption of the Intel CPU. The power consumed by Mali and A7 in different scenarios are presented in Fig. 12. It can be seen that the relatively lower power, lower performance Little CPU, A7, consumes less power than the GPU. Moreover, the application of half precision floating points has reduced the power consumption by approximately 33%.

D. Energy

While it is important to evaluate power consumption for heating matters, energy consumption, specifically in mobile applications, plays a very important role, as it translates to battery life. Fig. 13 illustrates the calculated energy consumption in different implementations by both the GPU and the CPU. As it is shown in this figure, employing half precision floating points has resulted in almost 60% reduced energy consumption in comparison with the case with full precision floating points. This is due to the reason that kernel execution with half floats is carried out in more than half of the time and with almost half power as the full floats. Although A7 is a low power CPU, the much lower kernel execution times on Mali has resulted

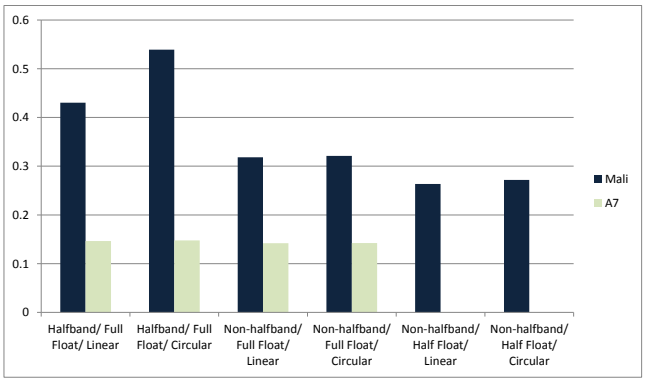

Fig. 12. Power in watts consumed by linear and circular digital filtering using halfband and non-halfband designs on different platfroms.

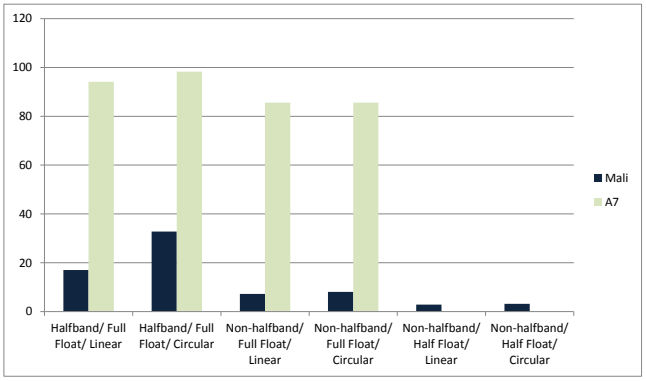

Fig. 13. Energy in µJ consumed by linear and circular digital filtering using halfband and non-halfband designs on different platfroms.

in overall lower energy consumption by the GPU.

VI. Conclusion

In this paper, we addressed the digital front-end processing of the IEEE 802.11ac receiver, targeting software-based processing implementation with substantially increased level of parallelism for fast execution. First, the overall 80 MHz received waveform is divided to two 40 MHz-wide signals through time-domain digital filtering so that the two 40 MHz signals can be then processed in parallel. We have optimized the channelization filter realizations and reported the results for both the linear and circular digital filtering. Then, the overall 802.11ac radio link was simulated, incorporating the developed channelization filter architectures, with two different WLAN/WiFi channel models. The SER and EVM performance of the channelization architectures were evaluated, showing that the link performance is not degraded by these filtering solutions. Finally, actual software implementations were carried out for linear and circular digital filtering using both halfband and non-halfband designs on different platforms, namely the Intel® Core™ i7-4800MQ CPU, ARM® Cortex®- A7™, and ARM® Mali™-T628 MP6 GPU. All filter designs were evaluated in terms of execution time and number of clock cycles on all three platforms, and power and energy consumption on the Mali GPU and A7 CPU. Comparing the OpenCL and C implementations revealed that exploiting parallelism using OpenCL yields a five times faster execution. The results also demonstrated that the high performance Intel CPU and the Mali GPU executed the filtering tasks much faster. Moreover, while the power efficient ARM A7 consumes less power than Mali, having very short execution times resulted in Mali consuming much lower energy. Taking advantage of half precision floating points on Mali reduces the execution time, number of clock cycles, power, and energy to a great extent. The measured execution times also showed that the designs can marginally meet the latency requirements for the IEEE 802.11ac. However, the filtering can more easily satisfy the restrictions by employing higher performance GPUs or CPUs.

Acknowledgment

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

References

[1] W. Tuttlebee (Ed.), *Software Defined Radio: Baseband Technologies for 3G Handsets and Basestations*. 1^{st}ed. West Sussex: Wiley, 2004.

[2] E. Grayver, *Implementing Software Defined Radio*. New York: Springer, 2013.

[3] *The OpenCL specification*, The Khronos Group Inc., 2011. [Online]. Available: https://www.khronos.org/registry/cl/specs/opencl-1.1.pdf

[4] *IEEE Standard for Information Technology Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks Specific Requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications – Amendment 4: Enhancements for Very High Throughput for Operation in Bands below 6 GHz*, IEEE Standard 802.11ac-2013, Dec. 2013.

[5] *Intel® Core™ i7 Processor Family for LGA2011 Socket*, Intel Corporation, 2014.

[6] Hardkernel co., Ltd. ODROID-XU3. Available: http://www.hardkernel.com/main/products/prdt_info.php?g_code=G140448267127&tab_idx=1

[7] *Cortex-A15 Technical Reference Manual*, ARM, 2011. [Online]. Available: http://infocenter.arm.com/help/topic/com.arm.doc.ddi0438c/DDI0438C_cortex_a15_r2p0_trm.pdf

[8] *Cortex-A7 MPCore Technical Reference Manual*, ARM, 2011, 2012. [Online]. Available: http://infocenter.arm.com/help/topic/com.arm.doc.ddi0464d/DDI0464D_cortex_a7_mpcore_r0p3_trm.pdf

[9] *big.LITTLE Technology: The Future of Mobile, Making very high performance available in a mobile envelope without sacrificing energy efficiency*, ARM, 2013.

[10] *The ARM® Mali™ Family of Graphics Processors*, ARM, 2013.

[11] *IEEE Standard for Information Technology Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks Specific Requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Standard 802.11-2012, 2012.

[12] H. W. Schüßler and P. Steffen, "Halfband filters and Hilbert transformers," *Circuits, Syst., Signal Process.*, vol. 17, no. 2, pp. 137–164, 1998.

[13] R. Ansari, "IIR discrete-time Hilbert transformers," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, pp. 1116–1119, Aug. 1987.

[14] T. Saramäki, "Finite impulse response filter design," in *Handbook for Digital Signal Processing*, S. K. Mitra and J. F. Kaiser, Eds. New York: John Wiley and Sons, 1993, ch. 4, pp. 155–277.

[15] *TGn Channel Models*, IEEE Standard 802.11-03/940r4, 2004. [Online]. Available at: https://mentor.ieee.org/802.11/dcn/03/11-03-0940-04-000n-tgn-channel-models.doc

[16] *TGac Channel Model Addendum*, IEEE Standard 802.11-09/0308r12, Dec. 2010. [Online]. Available at: https://mentor.ieee.org/802.11/dcn/09/11-09-0308-12-00ac-tgac-channel-model-addendum-document.doc

[17] *IEEE standard for floating-point arithmetic*, IEEE standard 754-2008, Aug.29, 2008.