Journal of Signal Processing Systems (2018) 90:1297–1309
https://doi.org/10.1007/s11265-017-1312-0

Software Defined Radio Implementation of a Digital Self-interference Cancellation Method for Inband Full-Duplex Radio Using Mobile Processors

Mona Aghababaeetafreshi[1] · Dani Korpi[1] · Matias Koskela[1] · Pekka Jääskeläinen[1] · Mikko Valkama[1] · Jarmo Takala[1]

Received: 21 April 2017 / Revised: 3 September 2017 / Accepted: 16 November 2017 / Published online: 30 November 2017

Abstract
New means to improve spectral efficiency and flexibility in radio spectrum use are in high demand due to congestion of the available spectral resources. Systems deploying inband full-duplex transmission aim at providing higher spectral efficiency by concurrent transmission and reception at the same frequency. Potentially doubling system throughput, full-duplex communications is considered as an enabler technology for the upcoming 5G networks. However, system performance is degraded due to the strong self-interference (SI) caused by overlapping of high power transmit signal with the received signal of interest. Furthermore, due to commonly existing radio frequency imperfections, advanced techniques capable of mitigating nonlinear SI are required. This article presents a real-time software-defined implementation of a digital SI canceller for full-duplex transceivers, potentially applicable even in mobile-scale devices. Recently, software-defined radio has gained a lot of interest due to its higher flexibility, scalability, and shorter time-to-market cycles compared to traditional fixed-function hardware designs. Moreover, as the performance enhancements achieved by increasing the clock frequency is reaching its limits, the current trend is towards multi-core processors. Since contemporary mobile phones already contain powerful massively parallel GPUs and CPUs, feasibility of a real-time implementation on mobile processors is studied. The reported results show that by adopting the presented solution, it is possible to achieve sufficient SI cancellation under time varying coupling channel conditions. Additionally, the possibility of carrying out such advanced processing in a real-time fashion on the selected platforms is investigated, and the implementation is evaluated in terms of execution time, power, and energy consumption.

Keywords 5G · Full-duplex · Self-interference cancellation · GPU · OpenCL

1 Introduction

In full-duplex communications, transmission and reception are carried out using the same spectral and temporal resources. Since this simultaneous use of bandwidth for both transmission and reception can theoretically increase the throughput by a factor of two, inband full-duplex communication is considered a promising enabler for the future 5G networks [17]. However, deployment of such systems is extremely challenging due to the strong self-interference (SI) produced as a result of the transmit signal coupling to the receiver. Thus, a crucial step toward achieving the promised gain in throughput by full-duplex transmission is to effectively attenuate the SI signal [3]. This is a complicated task, as it is not possible to simply subtract the known transmit signal from the received waveform to obtain the signal of interest. The reason behind this is the linear and nonlinear distortion of the signal while propagating from transmitter to the receiver due to transceiver analogue imperfections [16, 24]. This makes efficient suppression of the SI signal the main obstacle in realizing full-duplex systems.

The analogue imperfections make SI cancellation even more challenging on the mobile side compared to the base station side, since typically low cost components

✉ Mona Aghababaeetafreshi
mona.aghababaeetafreshi@tut.fi

[1] Tampere University of Technology, Korkeakoulunkatu 1, 33720 Tampere, Finland

are employed in mobile devices. Furthermore, processing resources and power consumption are limiting factors in hand-held devices. Thus, effective yet less complex solutions are required for the user equipment side in order to exploit the full potential of full-duplex systems. An SI cancellation method designed for a mobile station should also take the continuously changing environment around the device into consideration. Thus, an adaptive solution should be developed to keep track of the time-varying SI channel.

Currently, advanced processor architectures take advantage of parallel processing to achieve higher performance, since performance enhancement through increasing the processors' operating frequency in a fixed power envelope has reached technical challenges [9, 20]. Thus, we take advantage of the parallel processing capabilities of multi-core GPUs and CPUs. Moreover, to better utilize the parallel resources of the processors, Open Computing Language (OpenCL) is used. OpenCL is a programming standard for heterogeneous platforms, enabling efficient access to the available parallel resources [20].

In this work, a software based implementation for the entire digital canceller, which includes an orthogonalization procedure, together with a parameter learning algorithm is introduced. Such software defined radio (SDR) implementation provides more programmability, lower expenses, less design efforts, and thus shorter time-to-market cycles compared to traditional fixed-function approaches [13, 30]. To demonstrate the feasibility of an SDR implementation of the digital canceller, we worked with commercial off-the-shelf (COTS) low-cost components, which highlight the true advantages of a software-based solution. The implementation is carried out on four multicore platforms, which are suitable for both the network and user equipment side. The employed platforms are Qualcomm® Adreno™ 430, ARM® Mali™-T628 MP6, ARM® Cortex®-A15, and Intel® Core™ i7-4800MQ. The core i7 desktop CPU was mainly used for comparison purposes, while the rest of the processors are the main target of the study.

Using the measured signals from an actual full-duplex prototype, we demonstrate that sufficient real-time suppression of the SI signal is feasible using the proposed implementation. Furthermore, the implemented canceller is evaluated in terms of execution time, delay, power, and energy consumption. This article is a continuation of the work presented in [2].

The rest of the paper is organized as follows. Section 2 introduces some of the related work, existing in the literature. Section 3 describes the digital SI cancellation method adopted in this work. Section 4 explains the implementation of digital canceller blocks, and presents the selected platforms. Then, in Section 5, the implementation results are shown and analyzed. Finally, conclusions are drawn in Section 6.

2 Related Work

As mentioned in the previous section, sufficient cancellation of the SI signal is the main challenge in achieving a system operating effectively in full-duplex mode. This topic has been researched extensively and various techniques have been introduced in the literature.

In [21], a novel RF canceller architecture is described which cancels both the direct antenna coupling and multipath effects, while [3] proposes an all digital cancellation method. Duarte and Sabharwal [7] uses three different methods for SI suppression with both analogue and digital cancellation. Like [7], the proposed methods in literature typically include different stages of cancellation such as, propagation, analogue, and digital domain cancellation [16, 29]. Some studies assume that only the base station functions in full-duplex mode, while the mobile equipment remains operating in half-duplex mode. An example of which can be found in the work presented in [10].

Some contributions toward actual prototypes capable of full duplex communications can be found in the literature, such as the ones described in [6, 8, 16], and [27]. However, there are very few existing articles on real-time implementation of digital SI cancellation.The work in [22] implements parts of the digital cancellation method proposed in [23] on an FPGA. However, no contributions regarding a software based implementation of digital SI cancellation targeted for a mobile-scale device, as the one reported here, can be found in the literature. Especially this work uses COTS elements and eliminates the need for custom hardware design and additional hardware components.

Some contributions with similar arithmetic computations and implementation techniques, used for digital predistortion design, can be found in the literature. These works, found in [12, 25, 26], also study parallel processing on mobile-scale multicore processors, in which evaluations of the achieved performance are also reported. However, experimental measurements of power, or energy consumption are not carried out.

3 Digital Self-interference Cancellation

In order to reduce the SI signal to a level not interfering with the desired received signal decoding, both radio freqncy (RF) and digital domain cancellation are required. The former prevents the analogue-to-digital converter and the receiver low-noise amplifier (LNA) from saturating. However, further suppression of the SI signal should be carried out in the digital domain to improve system performance. The overall structure of the full-duplex

transceiver, including both the RF and digital cancellation is shown in Fig. 1. In this section, we address the latter by first introducing a model for the SI signal.

3.1 Self-interference Modeling

The transmitter and receiver paths contain numerous non-ideal components which distort the transmitted signal in linear and nonlinear ways. The transceiver impairments include nonlinear distortion by power amplifiers, phase-noise of the local oscillator, quantization noise from the analogue-to-digital converter, and in-phase/quadrature (I/Q) imbalance of the transmitter and receiver. Since the transmitter power amplifier (PA) is usually the most significant source of nonlinearity, we model the transmit signal by adopting the parallel Hammerstein (PH) model, commonly used for a highly nonlinear PA. Denoting the PA input by $x_{PA,in}$, the PA output, using the PH model, can be written as [23]:

$$x_{PA,out} = \sum_{\substack{p=1 \\ p\ odd}}^{P} \sum_{k=0}^{K-1} h_p^{PA}(k) u_p(x_{PA,in}(n-k)), \tag{1}$$

where P represents the highest nonlinearity order of the PA model, K is the memory length of the PA, h_p^{PA} is the pth-order model for the PA memory, and $u_p(x_{PA,in}(n))$ is computed as $|x_{PA,in}(n)|^{p-1}x_{PA,in}(n)$ and produces the pth-order basis function.

Now with the transmitter PA as the most prominent cause of nonlinear distortion, the whole SI channel can be effectively modelled using (1). Thus, the received signal at the digital canceller input, with respect to the original transmitted signal $x(n)$, can be expressed as:

$$r_x(n) = \sum_{\substack{p=1 \\ p\ odd}}^{P} \sum_{l=0}^{L-1} h_p(l) u_p(x(n-l)) + z(n), \tag{2}$$

where L is the memory length of the effective SI channel, $h_p(l)$ contains the coefficients for the effective pth-order SI channel, and $z(n)$ represents the noise and possible modeling mismatch. After estimating the unknown SI channel coefficients, denoted here by $\hat{h}_p(l)$, the signal at the output of the digital canceler can be written as:

$$e(n) = r_x(n) - \sum_{\substack{p=1 \\ p\ odd}}^{P} \sum_{l=0}^{L-1} \hat{h}_p(l) u_p(x(n-l)). \tag{3}$$

Looking at Eqs. 2 and 3, with accurate estimation of the SI channel coefficients, only noise should remain after digital cancellation, meaning that $e(n) \approx z(n)$. Furthermore, the estimated coefficients need to be updated, as the surrounding environment of a mobile device changes over time. The method used for the estimation should also have low computational complexity in order to be suitable for mobile-scale processing resources. Taking the aforementioned requirements into account, we have adopted the LMS based solution proposed in [23].

3.2 Orthogonalization

Since the different basis functions, mentioned in the previous section, are functions of the same transmit signal, they tend to be somewhat correlated. This will result in slow convergence of the LMS-based coefficient estimation. To alleviate this problem, the basis functions are orthogonalized using the method proposed in [23], which is briefly described here.

The basis functions are orthogonalized using a whitening transformation matrix. This matrix can be generated by eigendecomposition of the covariance matrix $\mathbf{\Sigma}$. Defining the instantaneous basis function vector as:

$$\mathbf{u}(n) = \begin{bmatrix} u_1(x(n)) & u_3(x(n)) & \cdots & u_p(x(n)) \end{bmatrix}^T, \tag{4}$$

with $u_p(x(n)) = |x(n)|^{p-1}x(n)$, the covariance matrix of basis functions across different nonlinearity orders can be defined as:

$$\mathbf{\Sigma} = \mathbb{E}[\mathbf{u}(n)\mathbf{u}(n)^H]. \tag{5}$$

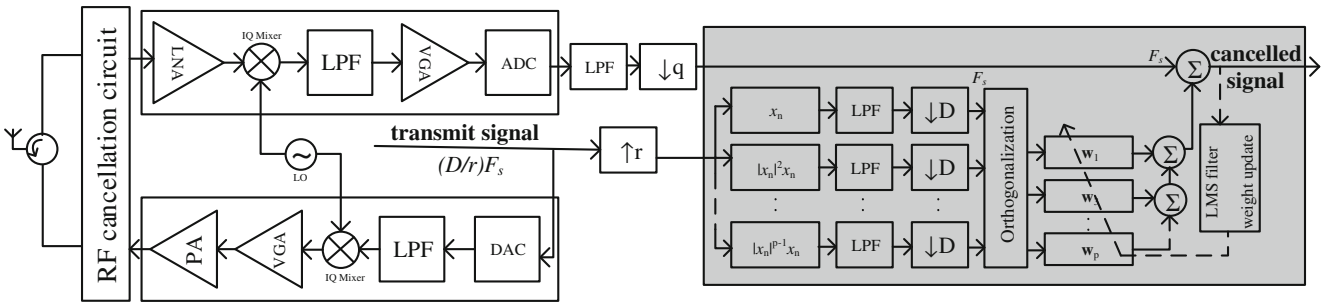


Figure 1 Overall structure of a full-duplex transceiver, where the grey part is implemented in software.

Having $\mathbf{\Sigma} = \mathbf{V}\mathbf{D}\mathbf{V}^H$, where diagonal matrix $\mathbf{D}$ contains the eigenvalues of Σ, and matrix $\mathbf{V}$ consists of the eigenvectors, the transformation matrix $\mathbf{T}$ can be written as:

$$\mathbf{T} = \mathbf{D}^{-\frac{1}{2}}\mathbf{V}^H. \tag{6}$$

Using the transformation matrix $\mathbf{T}$, the orthogonalized basis functions can be calculated by:

$$\tilde{\mathbf{u}}(n) = \mathbf{T}\mathbf{u}(n). \tag{7}$$

Now (3) can be re-written using the orthogonalized basis functions as follows:

$$e(n) = r_x(n) - \sum_{\substack{p=1 \\ p\ odd}}^{P} \sum_{l=0}^{L-1} \hat{h}_{p,ort}(l)\tilde{u}_p(x(n-l)), \tag{8}$$

where $\tilde{u}_p(x(n))$ are the orthogonalized basis functions using matrix $\mathbf{T}$, and $\hat{h}_{p,ort}(l)$ represents the corresponding SI channel estimates. Adopting vector notations, Eq. 8 can be expressed as:

$$e(n) = r_x(n) - \mathbf{w}^H\mathbf{u}_{ort}(n), \tag{9}$$

where

$$\mathbf{w} = \Big[\hat{h}_{1,ort}(0), \hat{h}_{3,ort}(0), \cdots, \hat{h}_{P,ort}(0), \cdots \\ \hat{h}_{1,ort}(L-1), \hat{h}_{3,ort}(L-1), \cdots, \hat{h}_{P,ort}(L-1)\Big]^T, \tag{10}$$

and

$$\mathbf{u}_{ort}(n) = \Big[\tilde{\mathbf{u}}(n)^T, \tilde{\mathbf{u}}(n-1)^T, \cdots, \tilde{\mathbf{u}}(n-L+1)^T\Big]^T. \tag{11}$$

It is worth mentioning that the covariance matrix $\mathbf{\Sigma}$ depends only on the statistical properties of the original transmit signal, and consequently it is not time varying. Therefore, we can assume that the transformation matrix $\mathbf{T}$ is computed and known beforehand.

3.3 LMS Parameter Learning

In this step, the effective SI channel coefficients are estimated using the decorrelated basis functions. This is carried out using an LMS-based algorithm with specific step-sizes for the different nonlinear terms [31]. Both pre-cursor and post-cursor taps are considered for a precise memory model of the SI channel. The original learning algorithm proposed in [23] is modified so that the estimated weights are not updated with every sample but only after a block of N samples are processed. This computing-friendly LMS-based approach is described in Algorithm 1, where $\tilde{\mathbf{u}}$ is a vector containing the orthogonalized basis functions calculated in Eq. 7, $\mathbf{w}$ contains the corresponding SI channel coefficients, $r_x(n)$ is the received signal, $e(n)$ represents the cancelled signal, and L_{pre} and L_{post} are the amounts of pre-cursor and post-cursor taps, respectively. Furthermore, $\boldsymbol{\mu}$ contains the step sizes, and N controls how often $\mathbf{w}$ is updated.

Algorithm 1 LMS-based adaptive nonlinear digital cancellation.

Initialize:
  $\mathbf{w} \leftarrow [0 \cdots 0]$
  $n \leftarrow L_{post}$
while transmitting **do**
  $\mathbf{u_{ort}}(n) = \Big[\tilde{\mathbf{u}}(n+L_{pre})^T \cdots \tilde{\mathbf{u}}(n-L_{post})^T\Big]^T$
  $e(n) = r_x(n) - \mathbf{w}(n)^H\mathbf{u_{ort}}(n)$
  if $(n \mod N == 0)$ **then**
    $\mathbf{w}(n+1) \leftarrow \mathbf{w}(n) + \boldsymbol{\mu}e^*(n)\mathbf{u_{ort}}(n)$
  end if
  $n \leftarrow n+1$
end while

4 Implementation

4.1 Implemented Blocks

In this section, the blocks, implemented in software, for the digital SI canceller, shown in Fig. 2 are described in short.

Basis Functions Calculation The first step is to calculate the nonlinear transformations of the original transmit signal. The pth-order basis function is computed for each sample as $u_p(n) = |x(n)|^{p-1}x(n)$. In this implementation, highest considered nonlinearity order is $P = 3$.

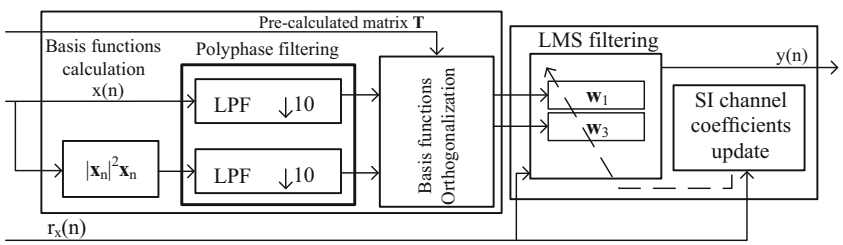


Figure 2 Implemented blocks for a third-order digital SI canceller, shown also in the grey part in Fig. 1.

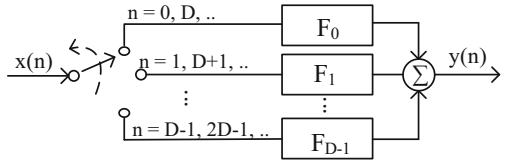


Figure 3 Functional structure of a polyphase filter with decimation factor D, where $y_(n)$ represents the signal samples after downsampling and filtering $x(n)$.

Polyphase Filtering As shown in Fig. 1, the transmit signal is oversampled before generating the basis functions. Thus the calculated basis functions can be resampled to the final cancellation signal sample rate. Assuming a decimation factor equal to D, only every D-th sample is kept after appropriate lowpass filtering.

To eliminate the unnecessary computations, we have designed a polyphase filter to perform the resampling task. This results in a more efficient implementation as the filtering is not performed on all original signal samples. An illustration of the adopted polyphase filter with downsampling factor D can be seen in Fig. 3, where $F_0, \cdots, F_{D-1}$ are sub-filters of length M. The total length for the polyphase filter is equal to $M \times D$. This work employs a polyphase filter with total length of 20, having downsampling factor $D = 10$, and sub-filter length $M = 2$.

The OpenCL implementation for the polyphase filter was carried out with both vector and scalar data types. With careful re-arrangement of the filter coefficients, the data loads can be carried out in a more efficient way. Figure 4 illustrates an example implementation and workload distribution for the polyphase filter. In this figure, the data and the coefficients are loaded as vectors of length four into vectors $\mathbf{x}$ and $\mathbf{p}$, respectively. After multiplication and summation, each work-item produces one output sample $\mathbf{y}[n]$. In Fig. 4, k denotes the polyphase filter length, $k = M \times D$, number of work groups is represented by n, and a local size of 16 is assumed for a clearer presentation.

Computing Orthogonalization Matrix This step is done according to Eqs. 4 to 6. However, as mentioned in the previous section the transformation matrix depends only on the statistical properties of the transmit signal, and does not change over time. Thus, we have assumed that the transformation matrix $\mathbf{T}$ is precomputed to reduce complexity and unnecessary computations. Having nonlinearity order $P = 3$, $\mathbf{T}$ is a 2×2 matrix.

Basis Function Orthogonalization After going through the polyphase filter, the basis functions are orthogonalized using the precomputed matrix $\mathbf{T}$, according to Eq. 7. This helps the LMS learning process to converge faster.

LMS Filtering The orthogonalized basis functions are filtered with the SI channel coefficient estimates. The filter length, i.e., the SI channel memory, is defined as $L = (L_{pre} + L_{post} + 1) \times \left(\frac{P+1}{2}\right)$. Then the filtered results are subtracted from the received signal to produce the cancelled signal. This corresponds to the computations from line 6 in Algorithm 1. The SI channel coefficients are updated after a block of N samples are processed using the *SI channel coefficients update* kernel.

SI Channel Coefficients Update Having the cancelled signal samples and step sizes $\boldsymbol{\mu}$, the SI channel estimates are updated as described in lines 7-10 in Algorithm 1. The selected step size is equal to 0.01 and 0.001 for the

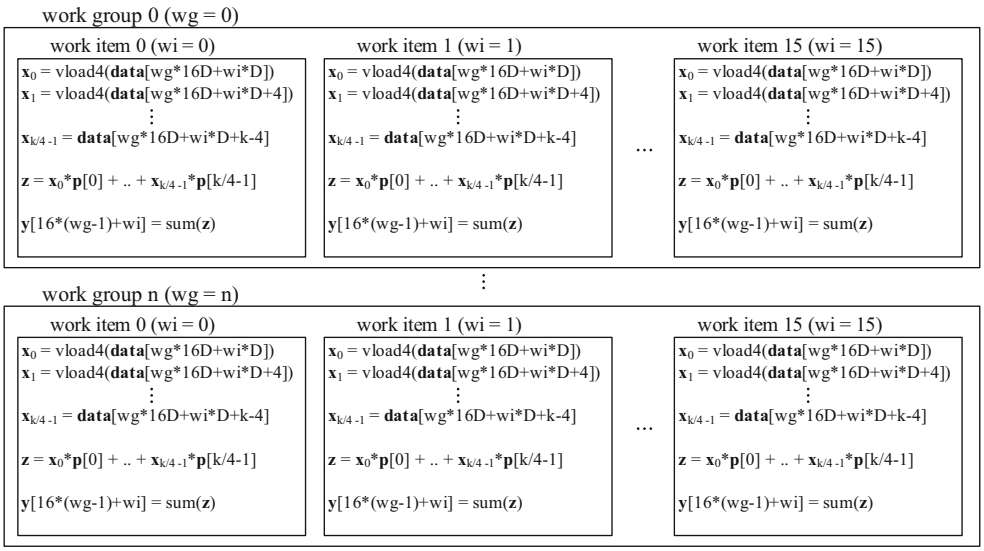


Figure 4 OpenCL kernel structure and workload distribution for the polyphase filter.

linear and third order terms, respectively. To reduce the computations, this step is also modified such that the coefficients are only updated after processing every N sample. This is done so that the LMS filter kernel would not have to wait for updated coefficients after processing every single sample. Less frequent updating of the coefficients reduces the dependency of the two kernels, the *LMS filter* and *SI channel coefficients update* kernels, and helps to increase parallelism, having larger blocks of input samples for the LMS filter kernel.

4.2 Platforms

In this work, three mobile scale multi-core processors and one desktop CPU are selected as the processing platforms. These are commercial off-the-shelf products that are currently employed in some of the available devices in the market. These platforms are briefly introduced in the following.

Qualcomm® Adreno™ 430 Adreno 430 is a mobile GPU by Qualcomm, and is available in the Snapdragon 810 System on Chip (SoC). This GPU is designed for mobile-scale devices and can run at 500 MHz, 600 MHz, or 650 MHz clock frequency [28]. Very little information about Adreno's architecture is publicly available, but it seems that it can approximately support 200 floating point operations in one clock cycle. To run the digital canceller blocks on Adreno 430, a commercial Android phone was used.

ARM® Mali™-T628 MP6 Similar to Adreno, Mali is a mobile-scale GPU and runs at a 600 MHz clock frequency [5]. Mali-T628 is a part of the Samsung Exynos 5 Octa (Exynos 5422) SoC. This GPU can scale from one to eight cores. Each core can handle up to eight floating point operations per cycle [15]. In this work, Odroid XU3 board [14] was used to access Mali.

ARM® Cortex®-A15 The Cortex-A15 MPCore is a low power multicore processor that can have one to four cores [4]. This multicore processor can be found, for example in the Exynos 5 Octa (Exynos 5422) SoC. It runs at 1.4 GHz clock frequency. Each of the four cores has one NEON (advanced Single Instruction Multiple Data instruction set) and vector floating point unit. The same Odroid XU3 board was used for implementing the digital canceller on A15 CPU.

Intel® Core™ i7-4800MQ Unlike the other three processing units mentioned above, the Intel Core i7 is a desktop CPU. This processor has four cores and can run at up to 3.7 GHz [18].

5 Evaluation and Analysis

In this section, the implementation results of the digital canceller blocks introduced in the previous section are presented. First, using the data from an actual full-duplex prototype system, described in [23] and [24], we demonstrate that the presented digital cancellation implementation can efficiently suppress the self-interference signal. Then, we evaluate this solution in terms of execution time, power, and energy consumption to study the feasibility of such software-based implementation using the four aforementioned COTS processors.

Software Tailoring To optimize the implementation, the kernels are tailored for each platform. Having a scalar or vector based implementation, the different possible vector lengths, and workload distribution between the OpenCL work-items are the factors that greatly affect the execution time of each processing task.

Different kernel designs on Mali showed that employing floating point vectors of length four yields the best results. Running the kernels on A15, different vector lengths and in some cases the scalar based implementation show similar results. However, execution of the kernels is fastest when the workload is distributed such that there are two work groups. The kernels designed for the Core i7 use vectors of length 16, and in most cases perform more efficiently when the processing is divided among eight work groups. Similar to Mali, Adreno achieves higher performance when using vectors of length four. Furthermore, workload should be spread among four work groups. The implementation results presented in the following section are obtained having designed the most efficient kernel implementation for each platform.

5.1 Digital SI Canceller Performance

After the sampled data, collected from the real full-duplex prototype system, was processed on the platforms, the cancelled signal was used to plot Fig. 5. Input buffers of 10, 1280 and 2560 samples are considered, which means that after downsampling by a factor of 10 and orthogonalization, one sample or blocks of 128 and 256 samples are processed before updating the canceller coefficients. When creating the plots in Fig. 5, L_{pre} and L_{post} are set to 8 and 7, respectively. Having total channel length $L = (L_{pre} +$

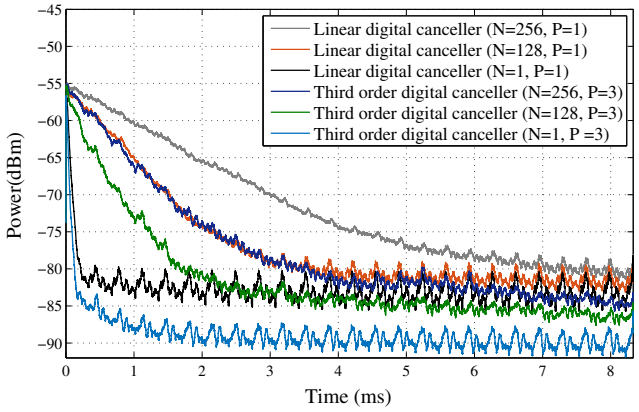


Figure 5 The instantaneous power of the SI signal, averaged over 1000 samples, of linear ($P = 1$) and third order ($P = 3$) digital canceller output signal, implemented on the Adreno 430, with respect to time, for $N = 1$, $N = 128$, and $N = 256$.

$L_{post} + 1) \times \left(\frac{P+1}{2}\right)$, L is equal to 16 for the linear canceller and 32 for the third order canceller.

This figure shows that the implemented canceller is capable of sufficient suppression of the SI signal, close to the receiver noise floor (-90 dBm). Allowing more time to converge will result in almost perfect SI cancellation. Being able to cancel the third order nonlinear SI, the third order canceller shows superior performance compared to the linear one. Comparing the curves in Fig. 5, it can be seen that less frequent updating of the SI channel coefficients has resulted in slower convergence of the LMS-learning algorithm. However,the difference is relatively small, especially after the initial learning phase, indicating that less frequent updating of the channel coefficients is a feasible option for controlling the computational complexity of the digital canceller.

5.2 Execution Time Analysis

In this section, the execution times related to the different building blocks of the SI digital canceller running on four different platforms, introduced in Section 4.2, are reported. A key factor in using OpenCL and multicore platforms with single instruction, multiple data (SIMD) or single program, multiple data (SPMD) optimized hardware is being able to take advantage of the available data parallelism. High performance can be achieved when parallel elements of the processor are utilized efficiently and the work load is distributed properly between these elements. We add to the inherent parallelism of the algorithms by increasing the amount of data processed in each kernel call. As a result, the processing time for each signal sample is decreased. Furthermore, vector lengths and workload distribution are adjusted for each implemented block on each platform so that kernel executions are carried out more efficiently.

The execution times for each digital canceller block implemented on the four platforms are presented in Tables 1, 2, 3 and 4. It should be noted that the reported times do not include data transfer, as SoC design can be easily made so that the processing unit sees the same memory as the radio hardware. The tables show the result using different buffer sizes for both linear and third order cancellers.

As the buffer lengths increase, the processing time related to one data sample decreases. In many cases, the processing time is approximately reduced by a factor of two, when the buffer size is doubled. This is the case for the "orthogonalization" and "weight update" kernels. However, the two filtering kernels, "polyphase" and "LMS", achieve

Table 1 Execution times of one signal sample for different kernels with respect to buffer lengths when implemented on *Adreno 430* for both the linear and third order canceller.

Buffer size	2560		5120		10240		20480	
Nonlinearity order	$P = 1$	$P = 3$	$P = 1$	$P = 3$	$P = 1$	$P = 3$	$P = 1$	$P = 3$
Basis functions	–	1,89	–	1,50	–	1,37	–	1,21
Polyphase	23	44,10	16	30,5	13,75	26,50	12,21	22,75
Orthogonalization	11	18	5,50	11,50	2,75	5,75	2,25	4,75
LMS filter	23	32,76	17	23,28	14,25	20,05	12,75	18,32
Weight update	11	11	5,50	5,50	2,75	2,85	1,38	1,27
Total [ns]	68	107,75	44	72,28	33,50	56,52	28,59	48,30
Rate [MHz]	14,71	9,29	22,73	13,84	29,85	17,69	34,98	20,70

Table 2 Execution times of one signal sample for different kernels with respect to buffer lengths when implemented on *Cortex A15* for both the linear and third order canceller

Buffer size	2560		5120		10240		20480	
Nonlinearity order	$P = 1$	$P = 3$	$P = 1$	$P = 3$	$P = 1$	$P = 3$	$P = 1$	$P = 3$
Basis functions	–	34,76	–	17,96	–	10,25	–	5,85
Polyphase	312,50	622,50	207,03	411,52	122,07	242,49	77,63	152,94
Orthogonalization	320,31	328,12	164,02	164,06	82,03	84,96	44,92	45,41
LMS filter	398,43	476,56	222,65	306,64	134,76	214,84	92,77	167,96
Weight update	265,62	242,18	142,57	125	83	81,05	42,96	40,52
Total [ns]	1296,8	1704,1	736,27	1025,2	421,86	633,59	258,28	412,68
Rate [MHz]	0,77	0,59	1,36	0,97	2,37	1,57	3,87	2,41

lower speed-up due to their inherent lack of parallelism, stemming from the summation step of convolution in the filters. The size of the buffers fed to the first block are chosen as powers-of-two multiplied by $D = 10$, which is the downsampling factor.

Moreover, the "basis functions" kernel's execution speed does not scale linearly with the buffer size. This can be explained by the input buffer size of this kernel which is ten times bigger than that of the "orthogonalization" and "weight update" kernels, which are executed after downsampling. This larger amount of data could saturate the available parallel resources of the cores, resulting in a slower speed-up. The effect of increasing the buffer size on the overall achieved performance is illustrated in Fig. 6. With longer buffers, the production rate improves less as the processing resources reach saturation.

The presented results show that Mali and A15 are only capable of processing the signal at rates lower than 15 MHz, even with large input data buffers. However, linear digital cancellation can be carried out on the Adreno 430 GPU and the Core-i7 CPU at rates over 20 MHz, having buffer sizes of 5120 samples. Furthermore, the Core-i7 and the Adreno 430 can perform third order digital cancellation for a 20 MHz waveform with buffer size of 5120, and 20480 samples, respectively.

Comparing the linear and third order cancellers in Tables 1–4, it can be seen that the polyphase filtering in the third order canceller takes approximately twice as much time as in the linear one. This is due to the fact that in case of a third order canceller, two filtering kernels are employed for both the linear and third order basis functions. The calculation of third order basis functions, carried out by the "basis functions" kernel is redundant in the linear canceller. The rest of the implemented blocks require equal or slightly more time for the third order canceller, as they only differ in a few multiplications and/or additions.

Table 3 Execution times of one signal sample for different kernels with respect to buffer lengths when implemented on *Mali-T628* for both the linear and third order canceller.

Buffer size	2560		5120		10240		20480	
Nonlinearity order	$P = 1$	$P = 3$	$P = 1$	$P = 3$	$P = 1$	$P = 3$	$P = 1$	$P = 3$
Basis functions	–	3,41	–	2,49	–	2,21	–	2,02
Polyphase	46,14	90,11	39,66	77,68	36,08	71,09	33,2	64,85
Orthogonalization	19,33	40,97	10,36	21,62	5,43	12,31	2,46	6,7
LMS filter	65,79	87,16	44,52	72,32	37,83	55,51	28,48	50,41
Weight update	30,51	25,19	16,08	16,84	6,15	8,44	3,43	4,54
Total [ns]	161,77	246,84	110,62	190,95	85,49	149,56	67,57	128,52
Rate [MHz]	6,18	4,05	9,04	5,23	11,70	6,68	14,80	7,78

Table 4 Execution times of one signal sample for different kernels with respect to buffer lengths when implemented on *Core i7* for both the linear and third order canceller.

Buffer size	2560		5120		10240		20480	
Nonlinearity order	$P=1$	$P=3$	$P=1$	$P=3$	$P=1$	$P=3$	$P=1$	$P=3$
Basis functions	–	1,92	–	0,66	–	0,55	–	0,48
Polyphase	20,78	39,86	12,02	22,34	9,64	17,97	6,12	11,38
Orthogonalization	5,93	7,42	3,71	4,45	2,22	2,59	1,29	2,22
LMS filter	23,75	22,26	12	12,90	9,64	10,02	6,30	6,49
Weight update	5,93	7,42	3,71	3,71	1,85	1,85	0,92	0,92
Total [ns]	56,39	78,88	31,44	44,06	23,35	32,98	14,63	21,49
Rate [MHz]	17,73	12,67	31,80	22,69	42,82	30,32	68,35	46,53

5.3 Delay Analysis

As discussed previously, to add to the available parallelism of the algorithm and utilize the parallel resources of the processors more efficiently, we increase the amount of data processed in each kernel, having longer input buffers. The disadvantage of this approach are longer delays for the system as larger blocks of data must be processed in each kernel call. The overall delays related to different buffer sizes for each platform are listed in Table 5. This delay is calculated as:

$$\begin{aligned} overall\ delay = {} & T_{basisfunctions} \times buffer\ size \\ & + T_{polyphase} \times \frac{buffer\ size}{D} \\ & + T_{orthogonalization} \times \frac{buffer\ size}{D} \\ & + T_{LMS\ filter} \times \frac{buffer\ size}{D} \\ & + T_{weight\ update} \times \frac{buffer\ size}{D}, \end{aligned} \tag{12}$$

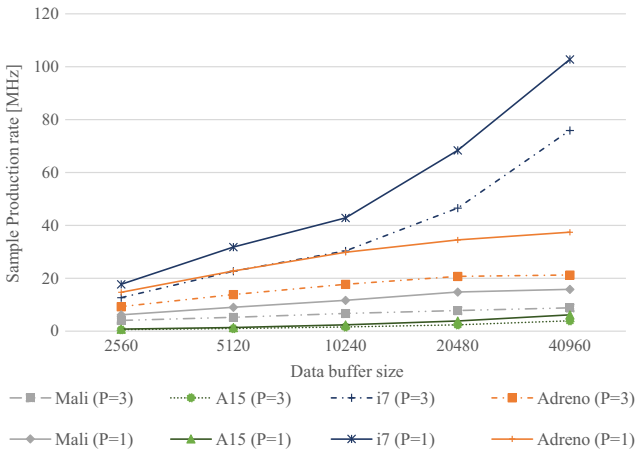


Figure 6 Sample production rate increase with regards to buffer size on the four platforms for both linear and third order cancellers.

where T_{kernel} is the processing time for one signal sample of "kernel", and D is the downsampling factor.

The calculated overall delay is equal to 25,6 μs for a third order SI canceller implemented on Core i7 and 70,5 μs on Adreno 430 with input buffer sizes of 5120 and 10240, respectively. These delays can be considered more than reasonable, when compared, e.g., to the inherent receiver processing latency of LTE user equipment (UE) which is, in minimum, 1 ms due to the downlink reference symbol structure as well as the adopted codeword mapping and interleaving processing. Furthermore, the specifications [11] allow an additional processing time of 3 ms for sending downlink hybrid ARQ (HARQ) acknowledgement within uplink L1/L2 control signaling. Thus, a balance can be achieved in the delay and sample production rate trade-off for a real application.

5.4 Power Consumption Analysis

Power measurement is not possible on Adreno 430 and Core i7, as no tools are provided for this purpose on the employed platforms. However, the Odroid XU3 board is equipped with sensors which allow measuring the power consumed by the Mali GPU, the DRAM, and both A7 and A15 CPUs. It is possible to probe the sensor data at discrete time instances. Thus, to provide a more reliable power consumption estimate, we take 200 samples of the sensor data in intervals of 100 ms. As the kernels are very small, they should be repeatedly run during this 20s interval. This keeps the processor cores occupied by the intended kernel. Then, the data from the sensors is averaged over the 20s period. However, any program running in the background, such as the operating system could partly account for the CPU/GPU power consumption. Thus, the processors idle power, i.e., power consumption while not running any kernels, are computed and subtracted from the measured results.

Figure 7 shows the average power measured when running the kernels for processing 5120 signal samples, as

Table 5 Overall delay in microseconds for different buffer lengths on all four platforms.

Buffer length	2560		5120		10240		20480	
Nonlinearity order	P = 1	P = 3	P = 1	P = 3	P = 1	P = 3	P = 1	P = 3
Mali	41,41	71,04	56,63	109,24	87,54176	173,51	42,08	300,44
A15	331,99	516,34	376,97	607,65	431,9846	743,26	42,08	952,07
i7	14,43	24,61	16,09	25,6	23,9104	38,83	42,08	52,85
Adreno	17,40	31,93	22,52	43,91	34,304	70,50	59,38	121,22

the consumed power by the GPU and CPU does not change significantly with different buffer lengths. It can be seen that there is very little or no difference in power consumption between the linear and third order canceller. As basis functions calculation is not required in the linear cases, only the measurements from the third order canceller are visible in the figure. The bars labelled as "total" correspond to the average power measured when running the complete digital canceller chain, which is slightly higher than the average power of all implemented blocks.

Comparing the results from the two processing platforms, it can be seen that A15 uses approximately 20 times more power compared to Mali when executing the same kernels. This can be explained by the higher clock frequency of the CPU (1.4 GHz compared to 600 MHZ), as well as the extra hardware on the CPU chip dedicated to the more general purpose computing. Increasing parallelism saves power by reducing the clock frequency for the same throughput. This reduces the switching activity, and more importantly the voltage which has quadratic effects to the power [1].

Mali consumes roughly an average of 104 mW running the third order digital canceller blocks with input buffer of 5120 samples. This can be considered negligible compared to the power consumption of e.g., an LTE receiver, which according to [19] is close to a couple of watts.

5.5 Energy Consumption Analysis

To better evaluate the feasibility of the proposed solution, it is also important to investigate the energy consumption of the implemented canceller. Since battery life depends on energy consumption, it is especially critical in hand-held devices. Furthermore, energy consumption comparison leads to fairer analysis compared to power, as we normalize the execution time.

We have used the measured average powers and the delays when processing 5120 samples for each kernel, and calculated the energy consumption. The results are shown in Fig. 8, in which the missing bars correspond to linear cases, where basis function calculation is redundant. As delay increases with longer buffers and power consumption remains the same, it can be concluded that energy consumption increases with longer buffers.

Using higher power and slower execution of tasks has resulted in higher energy consumption by A15 compared to Mali. Total energy used by the third order canceller,

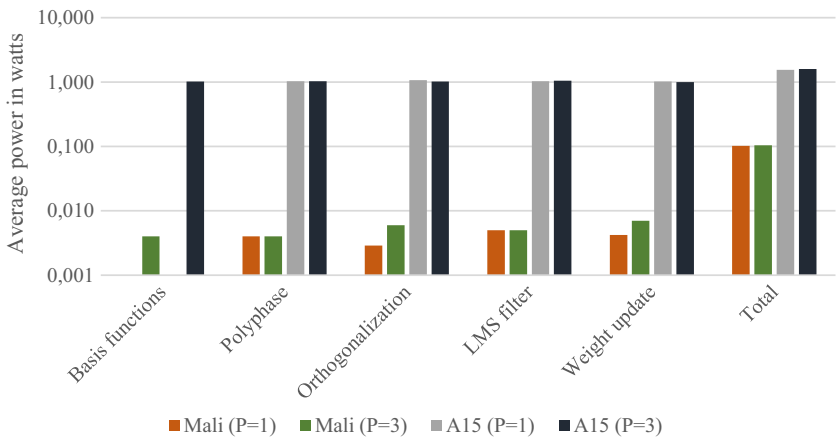


Figure 7 Consumed power by Mali and A15 running the linear and third order digital canceller kernels with input buffer length of 5120.

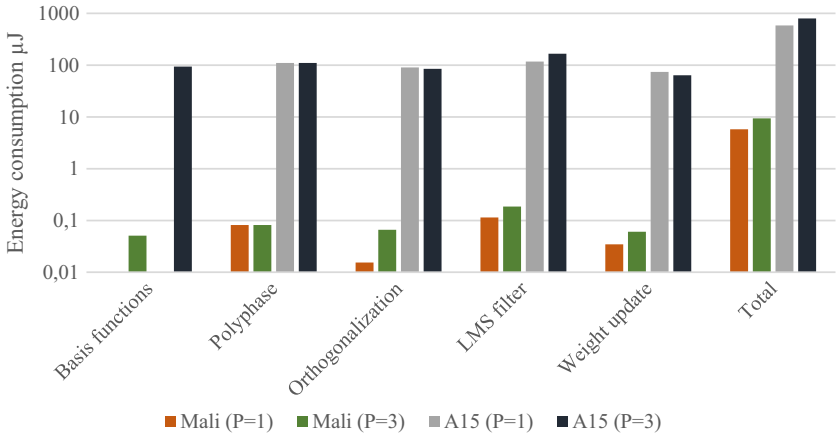


Figure 8 Consumed energy by Mali and A15 running the linear and third order digital canceller kernels with input buffer length of 5120.

implemented on Mali, and processing 5120 signal samples is approximately 9 μJ.

6 Conclusion

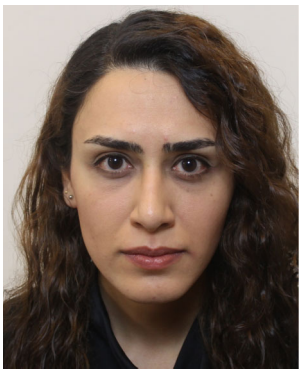
In this paper, we proposed a software-based implementation of a nonlinear digital SI canceller for full-duplex transceivers, using an adaptive cancellation algorithm, suitable for mobile-scale devices. To demonstrate the feasibility of a real-time SDR implementation, general-purpose low cost COTS processing platforms were selected, reducing the design time and costs compared to custom hardware design. The implementation was carried out on multicore processors and software tailoring was done using OpenCL to achieve high performance. The ability of the designed canceller to sufficiently suppress the SI signal was shown using the data from a real full-duplex RF test-bench. Then the implementation was evaluated in terms of execution time, delay, power, and energy consumption to investigate the feasibility of a real-time digital canceller suitable for hand-held devices. The results showed that the Qualcomm Adreno 430, a mobile-scale GPU, and the Intel Core i7, a desktop CPU, can run the proposed digital canceller with the required sample rate for, e.g., a 20 MHz LTE band. However, there is a trade-off between the achievable SI cancellation rate and the system delay, as longer data buffers are required for high sample production rates. The results also showed that, although the delay is shorter with a real-time linear SI canceller, it converges much slower and may not reach sufficient SI cancellation levels. As a proof of suitability to mobile platforms, also power and energy consumption of the implemented digital canceller were measured on Exynos 5422 SoC, and the Mali-T628 GPU showed more promising results compared to the Cortex-A15 for a mobile-scale device. It can be concluded that a real-time programmable implementation of a nonlinear digital canceller can be realized using the Adreno GPU, on the user equipment side, and the Core i7 CPU on the base station side. In the continuation of this work, we aim at adopting a platform which would allow dividing the workload between the CPU and one or more GPUs, and as a result achieving higher sample production rates with shorter delays. Furthermore, another interesting topic for future work is to use OpenCL to program an FPGA for digital SI cancellation and compare performance results of GPU and multicore processors in terms of time, power, and energy consumption.

Acknowledgements This work was supported by Tampere University of Technology graduate school, and the Academy of Finland via projects "In-Band Full-Duplex Radio Technology: Realizing Next Generation Wireless Transmission" (304147) and "Making Programmable Logic Feasible in the Cloud." (297548).

References

1. CMOS (1997). Power consumption and C_{pd} calculation. http://www.ti.com/lit/an/scaa035b/scaa035b.pdf. Last Accessed 08 Apr 2017.
2. AghababaeeTafreshi, M., Koskela, M., Korpi, D., Jääskeläinen, P., Valkama, M., Takala, J. (2016). Software defined radio implementation of adaptive nonlinear digital self-interference cancellation for mobile inband full-duplex radio. In *IEEE Global conference on signal and information processing*.
3. Ahmed, E., & Eltawil, A.M. (2015). All-digital self-interference cancellation technique for full-duplex systems. *IEEE Transactions on Wireless Communications*, *14*(7), 3519–3532. https://doi.org/10.1109/TWC.2015.2407876.
4. ARM Ltd (2011). ARM® Cortex® -A15 MPCore™ Processor. https://static.docs.arm.com/ddi0438/i/DDI0438.pdf. Last Accessed 08 Apr 2017.
5. ARM Ltd (2013). The ARM®, Mali™ Family of Graphics Processors. http://malideveloper.arm.com/downloads/events/2013/GDC/0319-11. Last Accessed 08 Apr 2017.
6. Duarte, M., Dick, C., Sabharwal, A. (2012). Experiment-driven characterization of full-duplex wireless systems. *IEEE Transactions on Wireless Communications*, *11*(12), 4296–4307. https://doi.org/10.1109/TWC.2012.102612.111278.

7. Duarte, M., & Sabharwal, A. (2010). Full-duplex wireless communications using off-the-shelf radios: feasibility and first results. In *Conference record of the forty fourth asilomar conference on signals, systems and computers* (pp 1558–1562). https://doi.org/10.1109/ACSSC.2010.5757799.
8. Duarte, M., Sabharwal, A., Aggarwal, V., Jana, R., Ramakrishnan, K.K., Rice, C.W., Shankaranarayanan, N.K. (2014). Design and characterization of a full-duplex multiantenna system for WiFi networks. *IEEE Transactions on Vehicular Technology*, *63*(3), 1160–1177. https://doi.org/10.1109/TVT.2013.2284712.
9. El-Rewini, H., & Abd-El-Barr, M. (2005). *Advanced computer architecture and parallel processing*. Wiley.
10. Everett, E., Duarte, M., Dick, C., Sabharwal, A. (2011). Empowering full-duplex wireless communication by exploiting directional diversity. In *Conference record of the forty fifth asilomar conference on signals, systems and computers* (pp. 2002–2006). https://doi.org/10.1109/ACSSC.2011.6190376.
11. 3rd Generation Partnership Project (2017). Technical Specification Group Radio Access Network; Requirements for further advancements for Evolved Universal Terrestrial Radio Access (E-UTRA) (LTE-Advanced) (Release 14). http://www.3gpp.org/ftp//Specs/archive/36_series/36.913/36913-e00.zip. Last Accessed 19 Aug 2017.
12. Ghazi, A., Boutellier, J., Anttila, L., Juntti, M., Valkama, M. (2015). Data-parallel implementation of reconfigurable digital predistortion on a mobile GPU. In *2015 49th Asilomar conference on signals, systems and computers* (pp. 186–191). https://doi.org/10.1109/ACSSC.2015.7421110.
13. Grayver, E. (2013). *Implementing software defined radio,* 1 edn. Springer.
14. Hardkernel co., Ltd. (2013). ODROID-XU3. http://www.hardkernel.com/main/products/prdt_info.php?g_code=G140448267127. Last Accessed 08 Apr 2017.
15. Harris, P. (2014). The Mali GPU: an abstract machine. https://community.arm.com/groups/arm-mali-graphics/blog/2014/03/12/the-mali-gpu-an-abstract-machine-part-3--the-shader-core. Last Accessed 08 Apr 2017.
16. Heino, M., Korpi, D., Huusari, T., Antonio-Rodriguez, E., Venkatasubramanian, S., Riihonen, T., Anttila, L., Icheln, C., Haneda, K., Wichman, R., Valkama, M. (2015). Recent advances in antenna design and interference cancellation algorithms for in-band full duplex relays. *IEEE Communications Magazine*, *53*(5), 91–101.
17. Hong, S., Brand, J., Choi, J.I., Jain, M., Mehlman, J., Katti, S., Levis, P. (2014). Applications of self-interference cancellation in 5G and beyond. *IEEE Communications Magazine*, *52*(2), 114–121.
18. Intel Corporation (2014). Intel® Core™ i7 Processor Family for LGA2011 Socket. http://www.intel.com/content/www/us/en/processors/core/4th-gen-core-i7-lga2011-datasheet-vol-1.html. Last Accessed 08 Apr 2017.
19. Jensen, A.R., Lauridsen, M., Mogensen, P., Jensen, P. (2012). LTE UE power consumption model: For system level energy and performance optimization. In *IEEE Vehicular technology conference (VTC Fall)* (pp. 1–5). https://doi.org/10.1109/VTCFall.2012.6399281.
20. Khronos OpenCL Working Group (2015). The OpenCL Specification, version 2.0. https://www.khronos.org/registry/cl/specs/opencl-2.0.pdf. Last Accessed 08 Apr 2017.
21. Kolodziej, K.E., McMichael, J.G., Perry, B.T. (2016). Multitap rf canceller for in-band full-duplex wireless communications. *IEEE Transactions on Wireless Communications*, *15*(6), 4321–4334. https://doi.org/10.1109/TWC.2016.2539169.
22. Korpi, D., AghababaeeTafreshi, M., Piilila, M., Anttila, L., Valkama, M. (2016). Advanced architectures for self-interference cancellation in full-duplex radios: algorithms and measurements. In *2016 50th Asilomar conference on signals, systems and computers* (pp. 1553–1557). https://doi.org/10.1109/ACSSC.2016.7869639.
23. Korpi, D., Choi, Y.S., Huusari, T., Anttila, L., Talwar, S., Valkama, M. (2015). Adaptive nonlinear digital self-interference cancellation for mobile inband full-duplex radio: algorithms and rf measurements. In *IEEE global communications conference* (pp 1–7). https://doi.org/10.1109/GLOCOM.2015.7417188.
24. Korpi, D., Tamminen, J., Turunen, M., Huusari, T., Choi, Y.S., Anttila, L., Talwar, S., Valkama, M. (2016). Full-duplex mobile device: pushing the limits. *IEEE Communications Magazine*, *54*(9), 80–87. https://doi.org/10.1109/MCOM.2016.7565192.
25. Li, K., Ghazi, A., Boutellier, J., Abdelaziz, M., Anttila, L., Juntti, M., Valkama, M., Cavallaro, J.R. (2015). Mobile GPU accelerated digital predistortion on a software-defined mobile transmitter. In *2015 IEEE Global conference on signal and information processing (GlobalSIP)* (pp. 756–760) https://doi.org/10.1109/GlobalSIP.2015.7418298.
26. Li, K., Ghazi, A., Tarver, C., Boutellier, J., Abdelaziz, M., Anttila, L., Juntti, M., Valkama, M., Cavallaro, J.R. (2017). Parallel digital predistortion design on mobile GPU and embedded multicore CPU for mobile transmitters. *Journal of Signal Processing Systems*. https://doi.org/10.1007/s11265-017-1233-y.
27. Mikhael, M., van Liempd, B., Craninckx, J., Guindi, R., Debaillie, B. (2014). An in-band full-duplex transceiver prototype with an in-system automated tuning for rf self-interference cancellation. In *1st International conference on 5G for ubiquitous connectivity* (pp. 110–115). https://doi.org/10.4108/icst.5gu.2014.258118.
28. Qualcomm Technologies (2015). Snapdragon 810 processor product brief. https://www.qualcomm.com/media/documents/files/snapdragon-810-processor-product-brief.pdf. Last Accessed 08 Apr 2017.
29. Sabharwal, A., Schniter, P., Guo, D., Bliss, D.W., Rangarajan, S., Wichman, R. (2014). In-band full-duplex wireless: challenges and opportunities. *IEEE Journal on Selected Areas in Communications*, *32*(9), 1637–1652. https://doi.org/10.1109/JSAC.2014.2330193.
30. Tuttlebee, W. (Ed.) (2004). *Software defined radio: baseband technologies for 3G handsets and basestations*, 1st edn. Hoboken: Wiley.
31. Widrow, B., McCool, J.M., Larimore, M.G., Johnson, C.R. (1976). Stationary and nonstationary learning characteristics of the lms adaptive filter. *Proceedings of the IEEE*, *64*(8), 1151–1162. https://doi.org/10.1109/PROC.1976.10286.



Mona Aghababaeetafreshi received her M.Sc. degree (with distinction) in information technology from Tampere University of Technology (TUT), Finland, in 2013, where she is currently pursuing her Ph.D. degree. She is a researcher in the Laboratory of Electronics and Communications Engineering, TUT. Her main research interests are in the area of software defined radio.

Dani Korpi received his B.Sc. and M.Sc. degrees (Hons.) in communications engineering from Tampere University of Technology, Finland, in 2012 and 2014, respectively. He is currently a researcher in the Laboratory of Electronics and Communications Engineering at the same university, pursuing his D.Sc. (Tech.) degree in communications engineering. His main research interest is the study and development of inband full-duplex radios, with a focus on analyzing the RF impairments.

Matias Koskela received his Bachelor and Master degrees with honors from Tampere University of Technology in 2014 and in 2015, respectively. His research interests include software and hardware parallelism and optimization especially in real-time rendering.

Pekka Jääskeläinen (Dr. Tech.) has worked with processor customization and programming tools since 2002. In addition to publishing over 60 papers in journals and conferences, he leads the open source development work of the TTA-based Co-design Environment (TCE), a toolset for rapid customization of co-processors and the Portable Computing Language (pocl), an open source OpenCL implementation. His current research interests include programmable heterogeneous platform customization methodologies and runtime/compiler techniques for enhancing performance portability of parallel programs.

Mikko Valkama (S'00–M'01–SM'15) was born in Pirkkala, Finland, on November 27, 1975. He received the M.Sc. and Ph.D. Degrees (both with honors) in electrical engineering (EE) from Tampere University of Technology (TUT), Finland, in 2000 and 2001, respectively. In 2002, he received the Best Ph.D. Thesis -award by the Finnish Academy of Science and Letters for his dissertation entitled "Advanced I/Q signal processing for wideband receivers: Models and algorithms". In 2003, he was working as a visiting post-doc research fellow with the Communications Systems and Signal Processing Institute at SDSU, San Diego, CA. Currently, he is a Full Professor and Laboratory Head at the Laboratory of Electronics and Communications Engineering at TUT, Finland. His general research interests include radio communications, communications signal processing, estimation and detection techniques, signal processing algorithms for flexible radios, cognitive radio, full-duplex radio, radio localization, and 5G mobile radio networks.

Jarmo Takala received his M.Sc. (hons) degree in Electrical Engineering and Dr.Tech. degree in Information Technology from Tampere University of Technology, Tampere, Finland (TUT) in 1987 and 1999, respectively. From 1992 to 1995, he was a Research Scientist at VTT-Automation, Tampere, Finland. Between 1995 and 1996, he was a Senior Research Engineer at Nokia Research Center, Tampere, Finland. From 1996 to 1999, he was a Researcher at TUT. Since 2000, he has been Professor in Computer Engineering at TUT and currently Vice President at TUT. Dr. Takala is Co-Editor-in-Chief for Springer Journal on Signal Processing Systems. During 2007-2011 he was Associate Editor and Area Editor for IEEE Transactions on Signal Processing and in 2012-2013 he was the Chair of IEEE Signal Processing Society's Design and Implementation of Signal Processing Systems Technical Committee. His research interests include circuit techniques, parallel architectures, and design methodologies for digital signal processing systems.