Implementation of a Nonlinear Self-Interference Canceller using High-level Synthesis

Sakari Lahti, Pablo Pascual Campo, Vesa Lampu, Lauri Anttila, Mikko Valkama, Timo D. Hämäläinen
Faculty of Information Technology and Communications, Tampere University, Tampere, Finland
sakari.lahti@tuni.fi, pablo.pascualcampo@tuni.fi, vesa.lampu@tuni.fi,
lauri.anttila@tuni.fi, mikko.valkama@tuni.fi, timo.hamalainen@tuni.fi

***Abstract*—High-level synthesis (HLS) aims to improve the productivity of digital logic design over traditional register-transfer level (RTL) methods. This paper shows that HLS can replace RTL when implementing a complex data path oriented signal processing algorithm under strict throughput constraints. Our system is a nonlinear spline-based Hammerstein self-interference (SI) canceller for full-duplex transceiver capable of achieving high SI suppression, while maintaining low computational complexity. The achieved suppression of the SI is superb 45 dB, while consuming 29 026 of the available LUTs, 17 992 of registers, and 655 of the DSP slices on Kintex-7 XC7K410T FPGA. Our paper also compares the usability of two commercial HLS tools that were used in this work.**

***Index Terms*—FPGA implementation, full-duplex, high-level synthesis, self-interference cancellation, software-defined radio.**

I. Introduction

The increasing complexity of digital systems has driven the industry to raise the design abstraction level to maintain productivity. The register-transfer level (RTL) has dominated as the principal design methodology for several decades. However, the RTL languages are rather low-level and require specialized knowledge. Therefore, RTL has low productivity and the engineers are in short supply compared to modern software methodologies.

High-level synthesis (HLS) increases the abstraction level by using higher-level languages such as C++ and SystemC to describe digital systems [1]–[3]. However, it has struggled to gain widespread adoption in the industry due to perceived worse quality of results in the past [4]. It is therefore important to research further, whether HLS is mature enough to replace RTL. This paper adds to that body of knowledge.

We use HLS to design a nonlinear spline-based Hammerstein self-interference (SI) canceller integrated in full-duplex (FD) transceivers [5]. The algorithm is computationally heavy, data path driven, and benefits from a highly parallel implementation on a field programmable gate array (FPGA). We employ two different commercial HLS tools and compare their design flows and usability.

The rest of the paper is structured as follows: Section II describes the theoretical background of the canceller algorithm. Section III discusses the user experience with the HLS tools, along with the required code transformations from the algorithm to a source code suitable for efficient HLS. Section IV shows the results of the final canceller implementation and Section V contains our conclusions.

II. Hammerstein Modeling of an FD Transmitter

A. The SI Problem in Full-Duplex Devices

FD technology aims at providing enhanced spectral efficiency, which is particularly sought in 5G wireless communications and beyond. FD does this by the simultaneous transmission and reception of useful information in time and frequency [6]. However, the SI problem arises, which is the unwanted transmit signal that is leaked in the RX chain of the FD transceiver. The SI should be identified and removed from the received signal [7].

Usually, FD transceivers implement a canceller (typically analog + digital) that suppresses the SI present in the system [8]. In this paper, we particularly focus on the HLS implementation of the digital canceller based on [9]. The final goal of the design is to provide sufficient levels of SI cancellation while maintaining low complexity, feasible to be implemented in real-time on a FPGA.

Fig. 1 shows the scheme of a complete FD transceiver, including the TX chain at the top and the RX chain at the bottom. The analog (RF canceller) and digital (SI regeneration and subtraction) cancellation stages are also presented. The digital cancellation stage estimates the transmit signal after the nonlinear effects of the power amplifier (PA) and the memory effects of the SI channel. After the SI signal is estimated, it is directly removed from the received signal so that only the useful information remains. The cancellation stage can be seen as

$$z[n] = \underbrace{r[n] + \tilde{x}[n]}_{d[n]} - \hat{x}[n] = r[n] + e[n], \tag{1}$$

where $d[n]$ is the overall received signal, $r[n]$ is the useful signal, $z[n]$ is the final signal, $\tilde{x}[n]$ is the SI signal, $\hat{x}[n]$ is the estimated SI signal, and $e[n]$ represents the residual error after the cancellation, which is ideally zero.

We need to accurately construct the modeled SI signal $\hat{x}[n]$ so that it lies as close as possible to $\tilde{x}[n]$. This modeling of unknown system is described in the following subsection.

B. The Hammerstein Modeling Algorithm

Behavioral modeling refers to the estimation of the transfer function of an a-priori unknown system or device, so that its behavior can be replicated in the digital domain. The RF modeling of the system is therefore simplified, requiring only

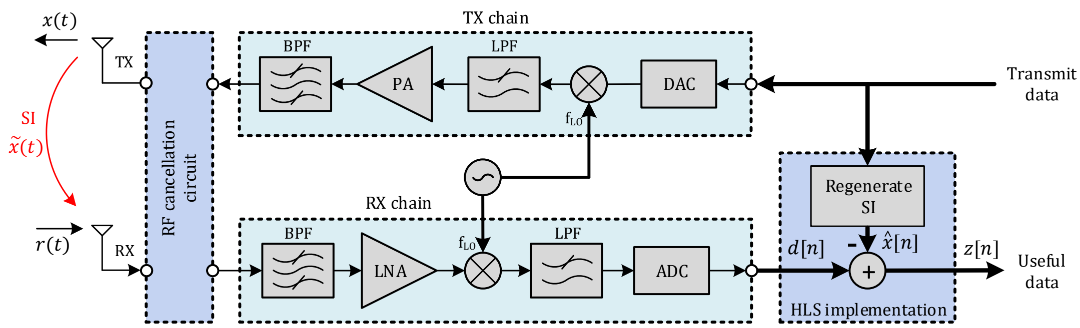

Fig. 1: An illustration of a FD transceiver. Several elements of the TX and RX chain are omitted for the sake of simplicity. Note that 'RF cancellation circuit' refers to the analog canceller, and the SI regeneration and subtraction constitute the digital canceller.

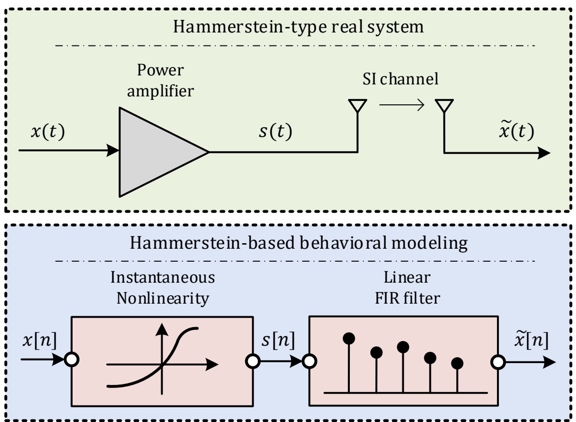

Fig. 2: A Hammerstein structure, where a nonlinear block is cascaded with a linear memory filter. The top diagram presents the real system, and the bottom diagram depicts the behavioral modeling scheme.

a mathematical formulation that comprises the relation of its input and output [10]. To obtain this relation, the system input signal and the signal to be modeled are required. In our case, the input signal is $x[n]$, and the signal to be modeled is $\tilde{x}[n]$, contained in the received signal $d[n]$.

Estimating the SI signal $\tilde{x}[n]$ is equivalent to modeling the TX chain plus the SI channel of the FD transceiver. The components along this path that cause the most distortion are the PA and the SI channel, the former introducing an instantaneous nonlinearity and the latter introducing memory effects. This type of a cascaded structure is known as a Hammerstein system, and, in general, the best approach to model such a real system is to use a Hammerstein model [11]. This idea is depicted in Fig. 2.

We adopted a spline-interpolated look-up-table (LUT) to model the nonlinearity and a finite impulse response filter for the memory filter, briefly described within the next lines. Spline interpolation divides the input range into different regions according to two local variables called index and abscissa [12], which can be defined respectively as

$$i_n = \lfloor |x[n]| \rfloor + 1, \tag{2}$$

$$u_n = |x[n]| - (i_n - 1). \tag{3}$$

After the input magnitude is evaluated, the spline interpolation can be implemented as

$$s[n] = x[n]\, \mathbf{\Phi}_n (\mathbf{1} + \mathbf{q}_n), \tag{4}$$

where $\mathbf{1} \in \mathbb{R}^{Q\times 1}$ is a vector of all ones, $\mathbf{q}_n \in \mathbb{C}^{Q\times 1}$ is a vector containing the PA model coefficients, $\mathbf{u}_n = \begin{bmatrix} u_n^2 & u_n & 1 \end{bmatrix}^T$, and $\mathbf{\Phi}_n \in {}^{1\times Q}$ reads

$$\mathbf{\Phi}_n = \begin{bmatrix} 0 & \cdots & 0 & \mathbf{u}_n^T \mathbf{C} & 0 & \cdots & 0 \end{bmatrix}, \tag{5}$$

with the term $\mathbf{u}_n^T \mathbf{C}$ indexed in the i_nth position of the vector, $\mathbf{C}$ is the second order B-spline matrix [9], and Q is the LUT size. Finally, the effects of the memory filter are added as

$$\hat{x}[n] = \mathbf{w}_n^H \mathbf{s}_n, \tag{6}$$

where $\mathbf{w}_n \in \mathbb{C}^{M\times 1}$ contains the filter coefficients and $\mathbf{s}_n \in \mathbb{C}^{M\times 1}$ is the signal regression of $s[n]$.

As presented in eq. (1), the error signal can be extracted as $e[n] = \tilde{x}[n] - \hat{x}[n]$, and the problem now becomes minimizing this magnitude. One way to do so is to use the steepest gradient descent adaptation rule [9]. With this approach, the learning rules for the PA model coefficients $\mathbf{q}_n$ and the linear filter $\mathbf{w}_n$ can be expressed as

$$\hat{\mathbf{w}}_{n+1} = \mathbf{w}_n + \mu_w[n] e^*[n] \mathbf{s}_n, \tag{7}$$

$$\hat{\mathbf{q}}_{n+1} = \mathbf{q}_n + \mu_q[n] e[n] \mathbf{\Sigma}_n^T \mathbf{X}_n^* \mathbf{w}_n, \tag{8}$$

where μ_w and μ_q are the learning rates, $\mathbf{X}_n$ is a diagonal matrix containing the signal regression of the input $x[n]$, $\mathbf{\Sigma}_n = \begin{bmatrix} \mathbf{\Phi}_n^T & \mathbf{\Phi}_{n-1}^T & \cdots & \mathbf{\Phi}_{n-M_{\mathrm{SP}}+1}^T \end{bmatrix}^T$, and M_{SP} refers to the number of memory taps considered.

A pseudocode example of the algorithm is shown in Alg. 1. One iteration of this algorithm reads and outputs one sample and updates the coefficients according to the learning rules.

III. HLS Implementation of the Digital Canceller

A. *Target platform*

The canceller was implemented in an existing FD capable testbed. The testbed included a National Instruments (NI) universal software radio peripheral (USRP), with an integrated FPGA device and a RF front-end, acting as a transceiver. The FPGA was a Xilinx Kintex-7 family device XC7K410T.

Data: $x[n]$, $d[n]$
Result: $\hat{x}[n]$
Initialize: $\mathbf{w}_n \leftarrow \bar{0}$, $\mathbf{q}_n \leftarrow \bar{0}$;
$\mu_w \leftarrow 1^{-2}$, $\mu_q \leftarrow 1^{-3}$;
for *All the samples in x[n]* **do**
Calculate i_n, u_n, $\mathbf{u}_n$ as in (2), (3);
$s[n] \leftarrow x[n]\,\mathbf{\Phi}_n(1 + \mathbf{q}_n)$;
$\hat{x}[n] \leftarrow \mathbf{w}_n^H \mathbf{s}_n$;
$e[n] \leftarrow d[n] - \hat{x}[n]$;
$\hat{\mathbf{w}}_{n+1} \leftarrow \mathbf{w}_n + \mu_w[n]e^*[n]\mathbf{s}_n$;
$\hat{\mathbf{q}}_{n+1} \leftarrow \mathbf{q}_n + \mu_q[n]e[n]\mathbf{\Sigma}_n^T \mathbf{X}_n^* \mathbf{w}_n$;
end

Algorithm 1: Pseudocode of the spline-based Hammerstein SI canceller.

NI's LabVIEW Communications version 2.0 was used in the testbed for the control of the USRP, and to develop the transceiver code to be run real-time on the FPGA. Additionally, the LabVIEW FPGA tools were used to integrate the canceller VHDL code.

The target operation frequency of the digital canceller was 60 MHz, with the requirement that one sample is processed every clock cycle without strict latency restrictions. The 60 MHz target is sufficient for narrow band signals used with the transceiver system.

B. The Used HLS Tools

We used two commercial HLS tools to create synthesizable RTL code for the digital canceller. We cannot disclose the names of both tools due to license agreements. We therefore call the first one Tool A. The second tool was Mentor Catapult HLS [13].

Both tools accept C/C++, and SystemC as input languages and produce synthesizable VHDL and Verilog code as output. They both include a built-in library for fixed-point C++ data types and use source code pragmas and graphic user interface directives to guide the synthesis process. Both tools also support automatic RTL testbench generation from a C/C++ source and perform equivalence checking between the simulation behavior of the C/C++ source of the design and the produced RTL.

C. Code Transformations and Tool Capabilities

1) Optimizations common to both tools: The starting point was a floating point MATLAB implementation of the canceller algorithm described in Section II. The code was first rewritten as C++ line-for-line as closely as possible. However, a direct conversion from an algorithm targeted for CPU is rarely suitable for HLS. Various code transformations are required to create an efficient hardware implementation that takes into account the limitations and advantages of FPGAs. The first such transformation was to convert the floating point types to fixed-point types, as floating point arithmetic would require an excessive amount of resources on FPGA.

Equation (2) requires determining the absolute value of the complex variable x, which necessitates calculating a square root. This is a complex operation, which can be simplified by using an approximate formula [9]:

$$\begin{aligned}|x[n]| = {} & \alpha \max\{|\mathrm{Re}\{x[n]\}|, |\mathrm{Im}\{x[n]\}|\} \\ & +\beta \min\{|\mathrm{Re}\{x[n]\}|, |\mathrm{Im}\{x[n]\}|\},\end{aligned} \tag{9}$$

where α and β are chosen based on selected approximation criteria, such as minimum RMS error for zero-mean signals [14].

The estimation of the new nonlinear coefficient $\hat{\mathbf{q}}_{n+1}$ is complex due to the large size of $\mathbf{\Sigma}_n$ and $\mathbf{w}_n$. However, the most significant memory taps in $\mathbf{w}_n$ are around time instant n. We therefore excluded some high-order memory taps in the filter coefficient without having a significant loss in cancellation performance. Consequently, the time dimension (i.e. number of rows) in $\mathbf{\Sigma}_n$ and $\mathbf{w}_n$ was reduced, only considering five filter taps in the update. With this approximation, the number of real multiplications was reduced by 87%. Furthermore, all multiplications with known zero elements in the matrices were removed.

To achieve the required throughput of one sample per clock cycle, all the loops in the algorithm were completely unrolled and the main loop was pipelined with an initiation interval of 1 clock cycle. All the arrays were mapped to registers on the FPGA to maintain fast accesses.

2) Tool A: We first utilized Tool A. All the variables involved in arithmetic operations used the same fixed-point precision of 30 bits, since Tool A uses the C++ standard library's "complex.h" class to handle complex numbers and the class requires that operands of arithmetic operations have the same type.

Synthesis revealed that the algorithm used more DSP48 slices on the FPGA than available. One DSP48 slice can handle a 25×18 bit multiplication, but our data were 30-bit complex values as noted above. Thus, two DSP48 slices were used per multiplication operation.

To reduce the number of multiplications, we lowered the total number of taps of the linear filter from 50 to 28, enabling the canceller to fit on the FPGA. Lowering the number of taps reduced the cancellation result by only about 1 dB in simulations. While this quality reduction is acceptable, to allow the algorithm to work on other devices in different environments, the small number of taps is undesirable.

A further problem with Tool A was that it was unable to schedule the algorithm with the requirement of one output sample per clock cycle. This was due to the coefficients $\mathbf{q}_n$ and $\mathbf{w}_n$ creating a data feedback dependency: They are updated at the end of the multi-cycle pipeline but are required at the beginning of the pipeline for the next sample, forcing the next input to wait for the coefficient update. An example of this situation is shown in Fig. 3a. The problem can be solved by adding registers to the data feedback path (Fig. 3b), balancing the latency with the forward data path. This changes the numerical properties of the algorithm slightly, but poses no degradation of the cancellation performance. However, despite adding feedback registers, Tool A still failed to schedule the algorithm.

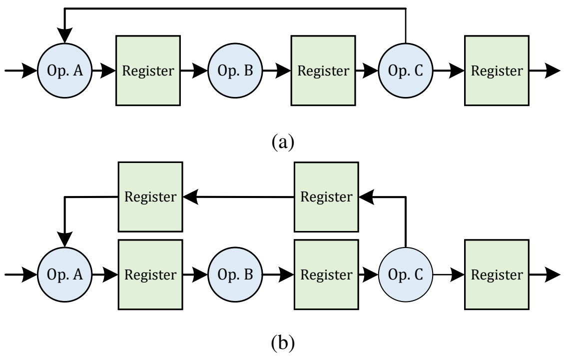

Fig. 3: Data feedback problem in the pipeline (a). Fixing the problem with feedback registers (b).

TABLE I: Resource usage of the canceller on the Kintex-7 XC7K410T FPGA

	DSP48s	**LUTs**	**Registers**	**Total slices**
Total	1 540	254 200	508 400	63 550
Used	655	29 026	17 992	12 323
Percentage	43%	11%	4%	19%

3) Catapult HLS: We next used Catapult HLS to synthesize the canceller making the appropriate changes to the C++ source code. Catapult HLS contains a custom class for complex numbers, which allows arithmetic operations between fixed-point numbers of different bit widths. The fixed-point types were now always chosen in accordance to the DSP48 slices, where the multiplicants have a maximum size of 18 and 25 bits. This enabled reducing the bit widths of most variables, resulting in a significantly lower DSP48 usage. This, in turn, allowed increasing the number of linear filter taps back to 50. Catapult HLS was also able to perform scheduling after adding the two delay registers to the data feedback path.

However, the canceller was unstable, as running it even for a short time on FPGA made the output to saturate. Analysis showed that the problem was caused by the default rounding mode used by the HLS tool, which is truncation. By changing the rounding mode of key variables to convergent rounding, the saturation problem vanished.

IV. Results

The resource usage of the digital canceller is shown in Table I. With only 43% of the available DSP48s used, there is still a considerable amount of space left for other operations on the FPGA, which highlights the relatively low complexity of the optimized implementation.

The algorithm was tested independently in the testbed to better validate its functionality. The transmitter and receiver chains were connected together with a wire and a 40 dB attenuator, which allows the SI signal to leak into the receiver attenuated. The signal used in the measurements was a widely used OFDM signal with a bandwidth of 10 MHz, which is suitable to reveal the nonlinear behavior of the transceiver system. The output power of the transmitter was adjusted to +16 dBm to ensure a heavily nonlinear behavior of the transmitter PA. The center frequency was 2.45 GHz, and the transmit signal was frequency shifted by -15 MHz.

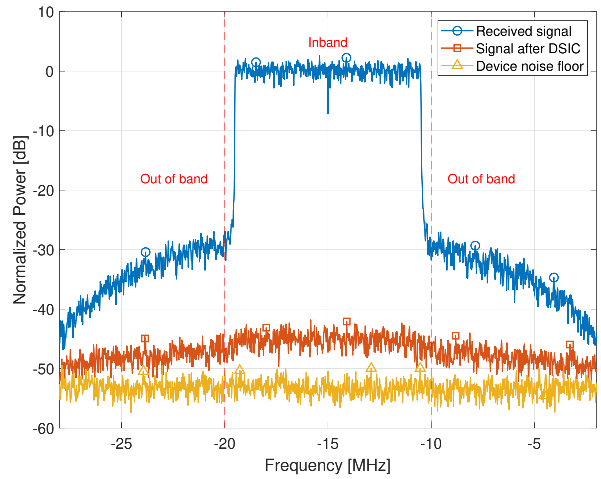

Fig. 4: The power spectral density of the received signal, signal after the digital canceller stage, and the receiver noise floor.

Fig. 4 illustrates the inband cancellation capabilities of the algorithm. It shows the power spectral densities of the received signal, the signal after the digital canceller, and the device noise floor. The signals are scaled so that the received signal inband power lies at 0 dB. It can be seen that the inband cancellation of the SI is around 45 dB, even with considerable amount of nonlinearity distorting the received signal. With less nonlinear distortion, it is possible to reach inband cancellations of up to 47 dB. Furthermore, the signal after the canceller can reach within 1 dB of the device noise floor if the transmit power is reduced or the analog cancellation of the SI is increased. For reference, [15]–[17], have demonstrated digital SI cancellations of 44 dB, 43 dB and 35 dB, respectively.

V. Conclusion

This study underlined that even though HLS can provide an expedient stepping stone from algorithm to RTL description, the process still requires understanding hardware design principles. At this point, it is unrealistic to assume that an efficient hardware implementation can be done without explicitly considering the properties of the target platform. Our experiences with the different HLS tools also show that even though they often boast similar properties on the surface, the choice of the tool can make a significant difference between the success and failure of a project.

Our real-time HLS implementation of the digital canceller achieved a 45 dB inband cancellation of the SI signal, which is on par with state-of-the-art. The FPGA resource usage was moderate with the DSP48 slices being the most heavily utilized component type by 43% of the total available. We conclude that HLS methodology can be used to realize a complex data path oriented signal processing algorithm under strict throughput constraints, without resorting to RTL techniques.

REFERENCES

[1] P. Coussy, D. D. Gajski, M. Meredith, and A. Takach, "An introduction to high-level synthesis," *IEEE Design Test of Computers*, vol. 26, no. 4, pp. 8–17, July 2009.

[2] H. Ren, "A brief introduction on contemporary high-level synthesis," in *2014 IEEE International Conference on IC Design Technology*, May 2014, pp. 1–4.

[3] J. Cong, B. Liu, S. Neuendorffer, J. Noguera, K. Vissers, and Z. Zhang, "High-level synthesis for FPGAs: From prototyping to deployment," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 4, pp. 473–491, April 2011.

[4] G. Martin and G. Smith, "High-level synthesis: Past, present, and future," *IEEE Design Test of Computers*, vol. 26, no. 4, pp. 18–25, July 2009.

[5] D. Bharadia, E. McMilin, and S. Katti, "Full duplex radios," in *ACM SIGCOMM computer communication review*, vol. 43, no. 4. ACM, 2013, pp. 375–386.

[6] D. Korpi, "Full-duplex wireless: Self-interference modeling, digital cancellation, and system studies," Ph.D. dissertation, Tampere University of Technology, Dec. 2017.

[7] A. Sabharwal, P. Schniter, D. Guo, D. W. Bliss, S. Rangarajan, and R. Wichman, "In-band full-duplex wireless: Challenges and opportunities," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 9, pp. 1637–1652, Jun 2014.

[8] M. Z. Waheed, P. P. Campo, D. Korpi, A. Kiayani, L. Anttila, and M. Valkama, "Digital cancellation of passive intermodulation in FDD transceivers," in *2018 52nd Asilomar Conference on Signals, Systems, and Computers*. IEEE, Oct 2018, pp. 1375–1381.

[9] P. Pascual Campo, D. Korpi, L. Anttila, and M. Valkama, "Nonlinear digital cancellation in full-duplex devices using spline-based Hammerstein model," in *IEEE Global Communications Conference (GLOBECOM)*, Dec. 2018.

[10] F. M. Ghannouchi and O. Hammi, "Behavioral modeling and predistortion," *IEEE Microwave magazine*, vol. 10, no. 7, pp. 52–64, Nov 2009.

[11] L. Ding, "Digital predistortion of power amplifiers for wireless applications," Ph.D. dissertation, Georgia Institute of Technology, 2004.

[12] M. Scarpiniti, D. Comminiello, R. Parisi, and A. Uncini, "Hammerstein uniform cubic spline adaptive filters: Learning and convergence properties," *Signal Processing*, vol. 100, pp. 112–123, 2014.

[13] "Catapult high-level synthesis." [Online]. Available: https://www.mentor.com/hls-lp/catapult-high-level-synthesis/

[14] R. G. Lyons, *Understanding digital signal processing*. Pearson Education India, 2004.

[15] Y. Kurzo, A. Burg, and A. Balatsoukas-Stimming, "Design and Implementation of a Neural Network Aided Self-Interference Cancellation Scheme for Full-Duplex Radios," in *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, Oct 2018, pp. 589–593.

[16] M. Chung, M. S. Sim, J. Kim, D. K. Kim, and C.-B. Chae, "Prototyping real-time full duplex radios," *IEEE Communications Magazine*, vol. 53, no. 9, pp. 56–63, Sep 2015.

[17] M. Aghababaeetafreshi, D. Korpi, M. Koskela, P. Jääskeläinen, M. Valkama, and J. Takala, "Software defined radio implementation of a digital self-interference cancellation method for inband full-duplex radio using mobile processors," *Journal of Signal Processing Systems*, vol. 90, no. 10, pp. 1297–1309, Oct 2018.