An Architecture for Real-Time Arbitrary and Variable Sampling Rate Conversion With Application to the Processing of Harmonic Signals

Fco. Javier Galindo Guarch, Philippe Baudrenghien, and Juan Manuel Moreno Arostegui

***Abstract*— The paper presents a new solution for sampling rate conversion and processing of harmonic signals with known but possibly varying fundamental frequency. This problem is commonly found in particle accelerators, for tracking the beam signals whose revolution frequency varies during the acceleration ramp. It is also common among many other fields such as speech and music processing, removal of mechanical noises, filtering of biomedical recordings, active crack imaging, etc. The key element in the proposed solution is a new architecture for a Farrow-based resampler, in which the resampling ratio can take any value and can be modified continuously to follow the signal fundamental frequency. The combination of two complementary resamplers creates a processing region where signal synchronous processing is performed. The resampler architecture is optimized for modern FPGA features. It decouples the processing and sampling clocks, and uses a single processing (hardware) clock whose frequency remains fixed. The functional model was migrated to Xilinx System Generator and the overall performance is evaluated with an application that filters a periodic signal whose frequency follows a known linear ramp in the presence of additive white noise.**

***Index Terms*— Resampler, Farrow structure, arbitrary sampling rate conversion, particle accelerator, harmonic signals, signal synchronous processing, FPGA.**

I. Introduction

CIRCULAR particle accelerators transfer energy to a beam by successive small momentum kicks. RF power injected into electromagnetic cavities induces electric fields which accelerate the particles at each passage. An increase in energy causes an increase in particle velocity. The effect is very small in relativistic machines, where the energy increase does not result in a significant change in velocity as it saturates at the speed of light, but is significant in smaller machines.

Manuscript received May 15, 2019; revised September 3, 2019, October 19, 2019, and December 13, 2019; accepted December 14, 2019. Date of publication January 3, 2020; date of current version May 1, 2020. This work was supported by CERN as a part of the LHC Injectors Upgrade. This article was recommended by Associate Editor G. Jovanovic Dolecek. *(Corresponding author: Fco. Javier Galindo Guarch.)*

F. J. Galindo Guarch is with the Department of Electronic Engineering, Universitat Politècnica de Catalunya, 08034 Barcelona, Spain, and also with the CERN Beams Department, 1211 Geneva, Switzerland (e-mail: francisco.javier.galindo@upc.edu).

P. Baudrenghien is with the CERN Beams Department, 1211 Geneva, Switzerland.

J. M. Moreno Arostegui is with the Integrated Smart Sensors and Health Technologies Group, Universitat Politècnica de Catalunya, 08034 Barcelona, Spain.

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSI.2019.2960686

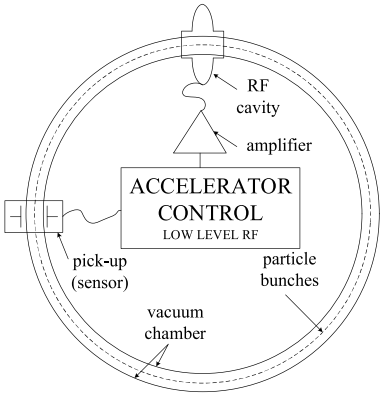


Fig. 1. Sketch of a circular particle accelerator: Accelerating cavity, RF pick-up sensor and the processing block.

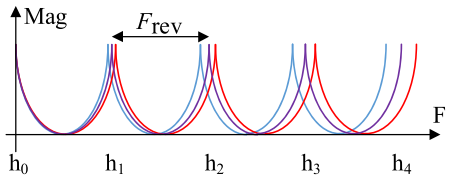


Fig. 2. Homothetic transformation. Position and tone spacing change during acceleration ramp proportionally to the revolution frequency increase (we call it Homothetic transformation because the change in spectrum is equivalent to a dilation of the frequency axis).

In modern circular machines, the particles travel in a vacuum chamber and the trajectory is not changed during the acceleration (Fig. 1). The change in particle velocity therefore results in a change in revolution period [1]. The Low Level RF system (LLRF) uses beam based signals (azimuth and transverse position) acquired with an electromagnetic pick-up sensor (Fig. 1) [2]. In return, after processing, it generates the drive for the RF amplifier powering the cavity. The processing is intended to keep the beam in the center of the vacuum chamber and to minimize the effect of RF noise. When a particle bunch passes through the pick-up, it generates a short pulse. In the time domain this signal is a train of delta pulses spaced by the revolution period, as the bunch crosses turn after turn. The signal spectrum is also a train of deltas at multiples of the revolution frequency [3].

During the acceleration ramp the revolution frequency increases and the spacing of the spectral lines changes. This is depicted in Fig. 2. The trace in blue shows the harmonic

content of the pick-up signal at the beginning of the RF ramp, while the traces in purple and red correspond to increased energy levels, with higher revolution frequencies.

This has severe implications on the LLRF system processing these signals. The digital processing, for instance filtering, needs to adjust its frequency response in real time to the changing signal spectrum during the acceleration ramp. When digital electronics was first introduced in particle accelerators in the mid-eighties, the obvious solution was to use a sampling clock which was a multiple of the revolution frequency [4]. The processing algorithm then needs not to be changed during the acceleration. However, modern FPGAs are intended for use with a fixed clock, implying adaptation of the algorithm as the acceleration proceeds.

This work presents a solution for the processing of pseudo-periodic signals whose period changes slowly and in a known manner, by tuning the sampling rate of the processed signal. It uses a single *fixed clock and avoids the real-time reconfiguration* of the processing elements. The treatment of a frequency-variant signal in real time is made possible, using any static algorithm. The solution is based on a Farrow-inspired arbitrary ratio resampler, for real time applications. The architecture is targeted for FPGA implementation.

The paper is structured as follows: Section II presents the state of the art for signal processing of pseudo-periodic signals. Section III is a review of existing rate conversion methods, with a focus on our processing requirements and implementation problems. Section IV presents the new proposed solution. Section V describes the implementation of our resampler in detail. Section VI presents a proof of principle of our proposal, and section VII summarizes the main results.

II. State of the Art in Signal Synchronous Processing

In the previous section, the beam signal was modeled as a succession of delta pulses. Due to the non-zero bunch length, the revolution frequency harmonics (nF_{rev}) are not all equal in amplitude, A_n and phase[1], φ_n. As the revolution frequency varies very slowly compared with the beam dynamics, the signal from a given bunch is

$$x(t) = \sum_{n=0}^{N} A_n \cdot \mathrm{e}^{i(2\pi n F_{\mathrm{rev}} t + \varphi_n)} \tag{1}$$

where N, the number of harmonics to be considered, depends on the intended application. For example, if we want to compensate for the beam-induced transient in a given cavity we must consider only the harmonics falling in the cavity bandwidth. In all cases the signal bandwidth must be limited before sampling to avoid aliasing.

Let us consider a simple process trying to enhance the pick-up signal out of white additive measurement noise. The required comb filtering contains harmonically related pass-bands located on the revolution frequency harmonics. Its frequency response would follow the signal spectrum shown on Fig. 2. Recall that the revolution frequency varies with time in our system.

In particle accelerators the historic solution was to clock the hardware with a multiple of the revolution frequency [4], [5]. The treatment therefore inherits beam synchronism by design. We call this method *Signal Synchronous Processing (SSP)*. There is no need for frequency tracking and reconfiguration of the processing. The real frequency F[Hz] of the signal sampled using a clock at f_{s}[Hz] is mapped to a normalized[2] digital frequency Ω [radian/sample]. Consider now a frequency sweep with an excursion Δ (2). If the sampling clock f_{s} is swept in proportion, the digital frequency Ω remains constant

$$\Omega = 2\pi \frac{F(1+\Delta)}{f_{\mathrm{s}}(1+\Delta)} = 2\pi \frac{F}{f_{\mathrm{s}}} = \text{constant} \tag{2}$$

If the processing algorithm has been designed at that digital frequency Ω, there is no need for reconfiguration. The swept sampling clock locks the processing on the spectral content of the beam signal.

This method has been widely used for compensation of beam-induced voltage in accelerator cavities and for longitudinal [5] and transverse bunch-by-bunch dampers [6]. Frequency modulation of the sampling and processing clock brings two problems however. First, a sweeping clock is not optimal for modern hardware. Second, the swept sampling clock modulates any absolute processing in the signal (such as compensation for sensor response) which is not intended to track the instantaneous frequency.

Another classic method is Adaptive Noise Cancelling [7]. The filtering of electrocardiogram using the Least Mean Square (LMS) algorithm is an early example. More sophisticated algorithms such as the Recursive Prediction Error (RPE) and Recursive Maximum Likelihood (RML) have been proposed for comb filtering [8], [9]. The later did not require knowledge of the instantaneous fundamental frequency. It can be estimated by the algorithm. The computational load of these algorithms scales linearly with the number of harmonics when using the simple LMS, and quadratic for the more sophisticated RPE and RML algorithms. Their performance degrades if the number of harmonics present in the signal is not modeled correctly [8]. The solution proposed in the present paper requires the knowledge of the fundamental frequency. But it is much lighter in terms of calculations as it will be independent of the number of harmonics.

Variable Fractional Delay (VFD) filters have been proposed to implement a comb adapted to any fundamental frequency (not necessarily a sub-multiple of the sampling frequency) [10]. The coefficients of the filter are then changed to track the fundamental frequency. A side effect is the displacement of the poles with the delay that can possibly make the Infinite Impulse Response filter (IIR) unstable. In the proposed method we do not reconfigure the filter coefficients as function of the changing signal frequency. The tracking is the responsibility of the resamplers while the comb filter has fixed coefficients, and therefore fixed pole locations.

[1]The envelope of the revolution frequency harmonics is the Fourier Transform of the longitudinal bunch profile.

[2]The normalized frequency Ω denotes a digital frequency in radians per sample, relative to a sampling clock at f_{s} (in Hertz) acquiring f_{s} samples per second.

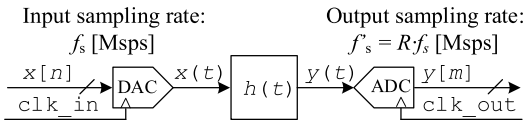


Fig. 3. Sampling Rate Conversion with Analog reconstruction.

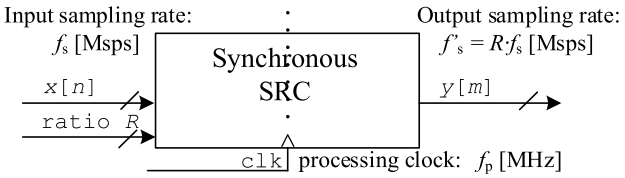


Fig. 4. Synchronous sampling rate conversion architecture: One single clock is present, being synchronous to input and output sampling rates.

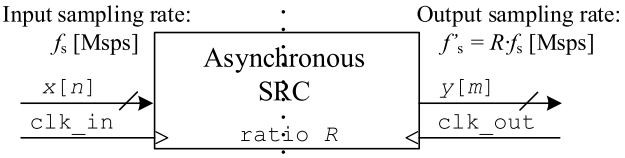


Fig. 5. Asynchronous sampling rate conversion architecture: The ratio R between input and output clocks can be any real number.

III. Sampling Rate Conversion Architectures

The ideal system for the presented application has two clock domains. One is the input fixed frequency clock for the processing algorithms that do not depend on the fundamental frequency of the signal, and the other is an output sweeping clock for the SSP. Such a system has been installed in the CERN SPS in 2002 [11]. There, digital data transfer and sampling frequency adaptation are solved by analog reconstruction of the signal after the first clock domain, and subsequent new sampling by the second domain [12]. This is depicted in Fig. 3.

Today rate conversion can be done with all-digital resamplers. Discrete Sampling Rate Conversion (SRC) is an operation transforming the sequence $x[n]$ acquired at a given sampling rate f_s, into a new sequence $y[m]$, which approximates the acquisition of the original time-continuous signal at another rate f'_s[13]. These discrete resamplers can be classified in two families: Synchronous SRC (SSRC) and Asynchronous SRC (ASRC) [14]. In the first one the timing reference is derived from a single clock, as depicted in Fig. 4. The relation between sampling rates is dictated by the resampling ratio R which is a fixed rational number

ASRC addresses a different philosophy. In this case there are two clocks and the output clock is asynchronous with respect to the input (Fig. 5). The resampling ratio, relation between the clocks, now adopts any value and can vary in operation with time. Architectures and optimizations in the literature are in general abstracted from the hardware which will be used for implementation [15]-[18], focusing on the properties of the resampling filters [19]-[22]. There are digital architectures with arbitrary resampling ratios which can be either synchronous [23]-[24], or asynchronous [25]-[27] depending on the implementation.

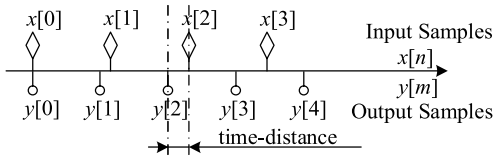


Fig. 6. Time-distance between input and output samples.

Resampling algorithms and solutions are usually classified according to the desired conversion ratio R. Integer ratios are well suited for implementation with Cascaded Comb filters or Cascaded Integrator Comb (CIC) architectures which exploit factorization of the transfer function. Fractional ratios are more suited for polyphase implementations where the periodic interpolation/decimation processes benefit from the optimization of the clocking and processing architecture. Subsequent optimizations of the polyphase architecture have led to architectures based on time multiplexing of the data-path, pre-stored filter coefficients, computation of the coefficients online, or a deeper optimization, the so-called Farrow-based architecture and its evolutions [16], [18], [28], [29].

This last architecture translates the interpolation operation to filtering of input samples to produce new outputs. The Farrow-based resampler includes a (Farrow-based) VFD filter [15], and a time-distance algorithmic engine. The latter computes the time-distance [30] between an output sample and the nearest input sample, that is plus or minus half an input clock period. The VFD filters the input samples to produce new resampled data, closely approximated to the original time-continuous signal at the calculated time-distance (Fig. 6).

The Farrow architecture requires limited or no hardware modifications when handling different resampling ratios [31] and allows for efficient implementations. It is the most common solution for arbitrary SRC. Static ratios, as in Software Defined Radio where the sampling ratio of generic RF front-ends is adapted to different standards [29], [32] are implemented generally with Farrow-based SSRCs. These are well suited for either Application Specific Integrated Circuits (ASICs) or FPGAs, as the output clock has fixed frequency derived from the system clock. When the ratio is irrational, SSRC is implemented with rational approximations validated according to the application. These solutions are not valid for us as the ratio supports only a discrete set of values, fixed during operation, and if other values are needed re-synthesis of the architecture is required [30], [33]-[37].

For resampling ratios varying in operation, as in digital audio [38], digital video [39] and digital communications synchronization [40]-[42], Farrow-based ASRC in ASICs or soft processors [43]-[46] are used. ASRC is well suited for technologies where the clocking architecture can be optimized for arbitrary swept clocks. It can also be implemented in FPGAs but an output swept clock-domain limits the use of internal hardware resources, as clock managers, intended for stable clocks. None of these available solutions solve our problem, which is the need for an FPGA arbitrary and sweeping ratio resampler, reconfigurable in real time, and using a fixed processing clock to best profit from all the FPGA features [47].

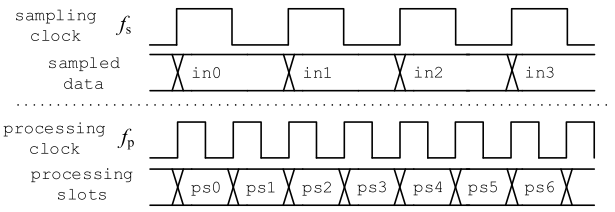


Fig. 7. Decoupling of sampling and processing clocks, samples vs processing slots in a system with $f_p = 2 \cdot f_s$.

IV. The Proposed Method

A. The Decoupling of Processing and Sampling Clocks

In discrete-time processing systems, the sampling clock f_s is the rate at which the data is acquired. In one sampling period T_s, one data sample is digitized. On the other hand, the processing clock f_p defines the rate at which the system is able to perform an elementary operation. In one processing period T_p, one processing slot is available. For simple systems the sampling and processing clocks are identical. However, when the processing requirements are more demanding, this paradigm limits exploitation of the hardware capabilities, the use of new devices, and the implementation of more complex algorithms.

Processing and sampling clocks need not be equal. The processing clock can be faster. This is the case in the solution presented here (Fig. 7). The sampling clock, at the top of the image, represents data acquired at f_s, while the processing clock, bottom, runs at a frequency f_p that is the double of f_s.

This decoupling idea is not new [36], [48]. It is widely used to implement time multiplexing of hardware resources within digital systems, for interleaving or serializing data. The presented resampler uses this decoupling scheme with a different flavor. It uses a fixed-frequency processing clock for the hardware. At the same time, the data-path has a variable and arbitrary sampling period. This is achieved by abstracting the data-path samples from the hardware. The sampling period of the data is decoupled from the hardware processing clock. In [36] a similar solution is used for SRC in a multichannel system, however the architecture is limited to a fixed conversion ratio, and its modification implies re-synthesis of the architecture. This solution is constrained by technological limitations, as the software tool only allows for a single clock domain. It is therefore not valid for us, as it does not handle a real-time variable ratio.

A consequence of the decoupled scheme is the use of a single hardware clock. Traditionally the resampler input port, or clock domain, uses a hardware processing clock at the input sampling rate. At the output port, a second hardware clock runs at the new output sampling rate, derived from the input or system clock in FPGA SSRC. The new architecture now uses the same hardware processing clock rate in both input and output ports, or hardware clock domains. Fig. 4 remains valid but the processing clock f_p is also used in the decoupled interfaces of the data-path, which contains data virtually sampled at a rate f_s at the input and $\mathtt{R} \cdot f_s$ at the output. Note that, when the processing clock is faster than the rate at which the data is acquired, there are more processing slots than samples available. By exploiting carefully this characteristic, the resampler can solve the problem of the different volume of data samples at the input and output ports. In the case of up-sampling for instance, there will be more samples at the output. In all cases the processing clock f_p must be larger than f_s and f_s'. If the maximum resampling ratio is for instance $R = 2$, a simple choice for the processing clock is

$$f_p = 2 \cdot f_s \tag{3}$$

This decoupled architecture satisfies the needs for a real-time variable and arbitrary rate, the use of a fixed processing clock at the input and output ports, and the use of a variable sampling rate at the input and/or output ports.

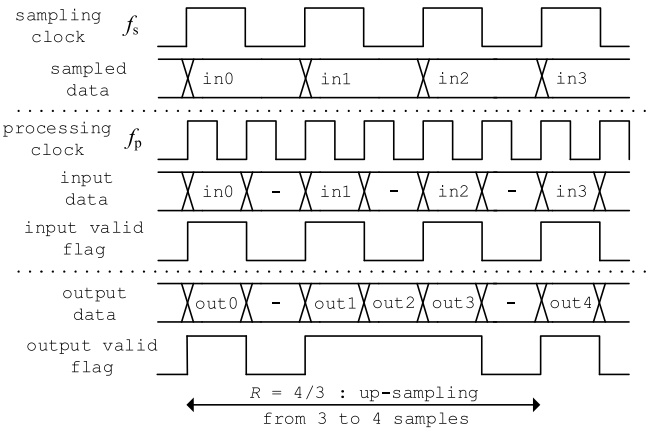


Fig. 8. Decoupling of sampling and processing clocks, valid signal and relation between the original and resampled signals.

B. The Valid Flag

The decoupled architecture can be implemented, adding little extra complexity to the system. An extra hardware line in the data-path, one bit wide, is used to indicate which processing slots contain valid data, and which ones are void slots. The decoupling therefore requires only this extra valid line and some glue logic. The activation rate, *ar*, of the valid line is a function of the relation between the sampling period and the period of the processing clock

$$f_s = ar \cdot f_p \tag{4}$$

Additional processing after the resampler (filtering for example) needs only to process slots flagged as valid. When the flag is not active, the processing can remain idle or discard the results. This solution satisfies the requirements and implements a variable ratio resampler with very little complexity.

Fig. 8 depicts the concepts presented so far. The top trace shows data sampled at f_s. The following two traces show the input and output ports of a resampler configured with a ratio $\mathtt{R} = 4/3$ (up-sampling) and a processing clock $f_p = 2 \cdot f_s$. The first of these shows the data flow for the input data-path containing the data bus and the valid signal. The valid line flags one data out of two processing cycles, $ar_{in} = (1/2)$.

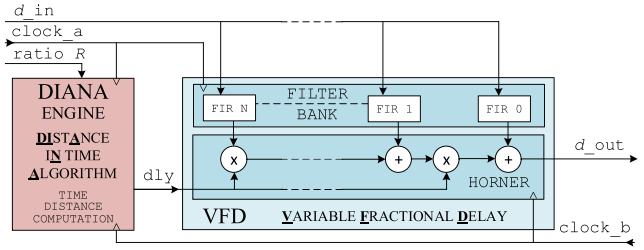


Fig. 9. Resampler architecture based on a Farrow VFD filter, and an algorithmic engine that implements the DIANA algorithm for computing time distances.

The bottom trace shows the data flow at the output port. The activation rate of the valid signal is now four out of six processing cycles

$$ar_{\text{out}} = R \cdot ar_{\text{in}} = (4/3) \cdot (1/2) = (2/3) \tag{5}$$

The sampling rate of the data-path at the output port is

$$\begin{aligned} f_{\text{s\_out}} &= R \cdot f_{\text{s\_in}} = (4\ /\ 3) \cdot f_{\text{s\_in}} \\ &= (4/3) \cdot ar_{\text{in}} \cdot f_{\text{p}} = (2\ /\ 3) \cdot f_{\text{p}} \end{aligned} \tag{6}$$

C. Farrow-Based Resampling Architecture

The resampler uses an interpolating filter implemented with a hardware efficient Farrow-based VFD filter [15]. The resulting architecture does not need polynomial re-computation of the interpolating coefficients for each output sample. Instead, it exploits the reordering of the filtering arithmetic operations within a Farrow architecture. The VFD hardware internals are depicted in Fig. 9. It is composed of a bank of FIR sub-filters, which pre-processes the available input samples, d_{in} in (7), to generate intermediate filtered data, $\text{FIR}_n(d_{\text{in}})$. These filters FIR_n use coefficients that are static and pre-computed offline. A new output d_{out} is computed by combining the intermediate data, weighted by powers of the delay value dly

$$\begin{aligned} d_{\text{out}}(d_{\text{in}}, dly) &= (dly)^0 \cdot \text{FIR}_0(d_{\text{in}}) + (dly)^1 \cdot \text{FIR}_1\,(d_{\text{in}}) \\ &\quad + \cdots + (dly)^{\text{N}} \cdot \text{FIR}_{\text{N}}(d_{\text{in}}) \\ &= \text{FIR}_0(d_{\text{in}}) + dly \cdot (\text{FIR}_1(d_{\text{in}}) \\ &\quad + dly \cdot (\text{FIR}_2(d_{\text{in}}) + dly \cdot (\cdots))) \end{aligned} \tag{7}$$

The Horner's rule [49] in (7) efficiently reorders the arithmetic to optimize the hardware implementation. ASRC implementations of this architecture employ at least two clock domains, marked `clock_a` and `clock_b` in Fig. 9. Our goal is to use a single processing clock, and the decoupled data-path in the FPGA. This combination makes it possible to have different sampling rates in data-paths, using a single processing clock, and to vary rates in real-time. The problem is now to synchronize the data-path in the different elements of the VFD, and the delay in the time-distance engine. References [33] and [36] have faced similar problems, however their solutions require an external signal at the output sampling rate. The present solution presents a data-path which is capable of this arbitrary real time modification of the ratio with a single system clock.

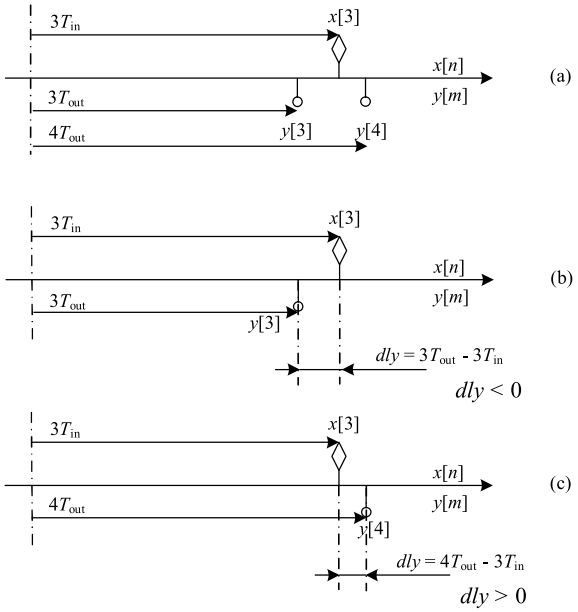


Fig. 10. (a) Absolute time position for input sample $x[3]$, and output samples $y[3]$ and $y[4]$. (b) Delay computation for output $y[3]$. (c) Delay computation for output $y[4]$.

A Farrow resampler contains a time-distance algorithmic engine, that calculates the delay for the VFD. This delay is the time difference between available input samples and desired interpolated output. For static resampling ratios, it is a periodic sequence, making the implementation easy with a Numerically Controlled Oscillator (NCO) [16], [37]. In [25] a solution for a variable rate is presented, but it is not valid for us as it does not address a decoupled data-path (where data propagates in the hardware based on the input sampling rate; this concept is presented in section V). The solution presented in [36] uses such a data-path, but in addition to the processing clock it requires an external signal driving the output rate. This adds complexity that we want to avoid.

For our real time variable and arbitrary ratio implementation, with fixed processing clock and decoupled data-path, a new algorithm has been developed, the DIstAnce iN time Algorithm (DIANA). The architecture and the algorithm focus on these adaptive processing requirements rather than optimization of the sampling rate conversion performance.

The delay generated by the VFD is a physical time difference in seconds. Fig. 10(a) shows several cases: The diamonds mark the input samples, $x[n]$, with T_{in} the input sampling time. The circles represent output samples $y[m]$, with T_{out} the output sampling time. Fig. 10(b) depicts the delay computation for the output sample $y[3]$, considering $x[3]$ as the input sample in the middle tap of the interpolating filter [22]. The delay value is the time difference between the two discrete samples

$$dly = t_{y[3]} - t_{x[3]} < 0 \tag{8}$$

In this case, looking "backwards", the delay value is a negative number. Fig. 10(c) depicts the delay computation for the output sample $y[4]$. In this case, the delay value is positive,

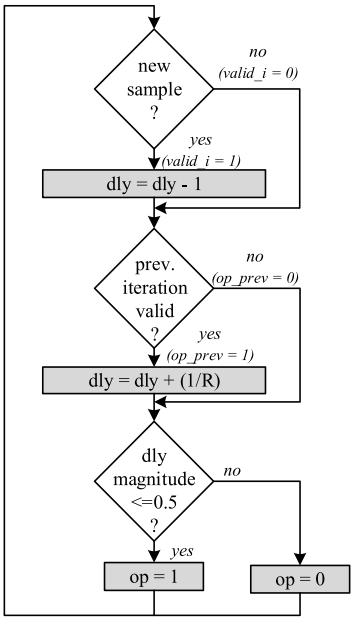


Fig. 11. DIANA algorithm for time-distance computation.

looking "forward"

$$dly = t_{y[4]} - t_{x[3]} > 0 \tag{9}$$

From now on, we will normalize the delay to the input sampling frequency f_s. The VFD filter will accept delay values in the range $[-0.5, +0.5]$. After normalization, a delay equal to one output sample will be $1/R$ with R the resampling ratio. The DIANA algorithm is shown in Fig. 11. The input variables are the `valid_i` signal of the data-path, which indicates when a new input sample is available, and the `op_prev` signal which keeps track of whether a valid output sample was calculated during the previous iteration. In case a new input sample is received, the delay variable *dly* is decremented by one unit. If the previous iteration has produced a valid output sample, the delay must be incremented by $1/R$ according to the time position of the next output sample. When the delay magnitude is less than or equal to 0.5, a new output sample can be computed. This is indicated by the output signal `op` forwarded to the VFD together with the delay value. The algorithm is initialized with $dly = 1$.

The architecture assumes that each new sample arriving at the resampler is directly inserted into the VFD filter bank. The algorithm iterates with the processing clock f_p. The algorithm can handle the delay computation as long as the processing clock frequency f_p is higher than both input and output sampling clocks (f_s and f_s'). This architecture handles changes in the ratio, in real time, by just updating the R variable. Of course, the change in ratio must be very slow compared to the processing period $1/f_p$ and to the resampler latency. A proof of principle presented in section VI demonstrates this feature.

The VFD of Fig. 9 has been designed using the weighted-least-square (WLS) method [22]. The delay signal `dly` takes values from -0.5 to 0.5. For each value we want the response from `d_in` to `d_out` to approximate an exact delay. Let $H[\Omega, p]$ be this transfer function, with Ω the normalized angular frequency in radian/sample and p the delay, the WLS computes the filter coefficients that minimize the cost function

$$J = \int_0^{\alpha\pi} \int_{-0.5}^{0.5} \left| H(\Omega, p) - e^{-j\Omega p} \right|^2 \mathrm{d}p \, \mathrm{d}\Omega \tag{10}$$

The parameter α is a fixed number, smaller than one, that specifies the passband. The number of filters and the filter length must be chosen to achieve the required precision in the passband. In our design, we set $\alpha = 0.6$, and used six filters containing fifteen taps each. The resulting maximum square error for all values of p, in the passband is 10^{-9}. This was found to be good enough for our application. The number of taps has a big effect. We have reduced it to seven, without changing other parameters, which resulted in a maximum square error of $3{\cdot}10^{-5}$.

V. Implementation of the Resampler

A. The High Level Architecture

The high-level architecture of the resampler is presented in Fig. 12. From a functional point of view, only four signals interface with the architecture. At the input the data-path port, `d_in_str`, feeds the available data samples, a control port, `T_out_n`, lets the resampler know the relation between output and input rates ($1/R$), and finally a clock port, `clk`, drives the hardware. At the output, only the data-path port, `d_out_str`, is present.

Looking at the internals of the resampler, it is composed of three functional blocks. The first entity, in red in Fig. 12, hosts the DIANA engine, algorithm presented in Fig. 11, for the computation of the time shift (`dly` signal). It also controls when an output data sample can be computed (`op` signal). The second block implements the VFD filter, in blue in Fig. 12. This hosts the bank of FIR filters to process the available samples, and the Horner chain of adders and multipliers combining these filter outputs as outlined in Fig. 9. Finally, the third block, in green in Fig. 12, contains the "Control Logic and Synchronization Memories". This block handles the communication and synchronization between the entities of the resampler and the interface ports.

B. The Interface Ports

The data-path port `d_in_str` in the input interface of the resampler is composed of two signals (Fig. 12), a data bus `data_i`, which can be implemented with any generic bit width depending on the application, and a valid signal `valid_i` one bit wide. The data bus contains the quantized samples provided to the resampler. The valid signal indicates when the data bus contains valid information. The control port of the resampler, `T_out_n`, is also a bus signal with a width depending on the application performance (precision of the resampling ratio). Finally, the clock port `clk`, receives the processing clock. The output interface follows the same philosophy. The data-path port `d_out_str` is composed of two signals, a bus containing the data words `data_os`, and a valid signal `valid_o` indicating valid samples. Again, the processing clock is clk and the sampling period is defined by the activation ratio of the valid signal.

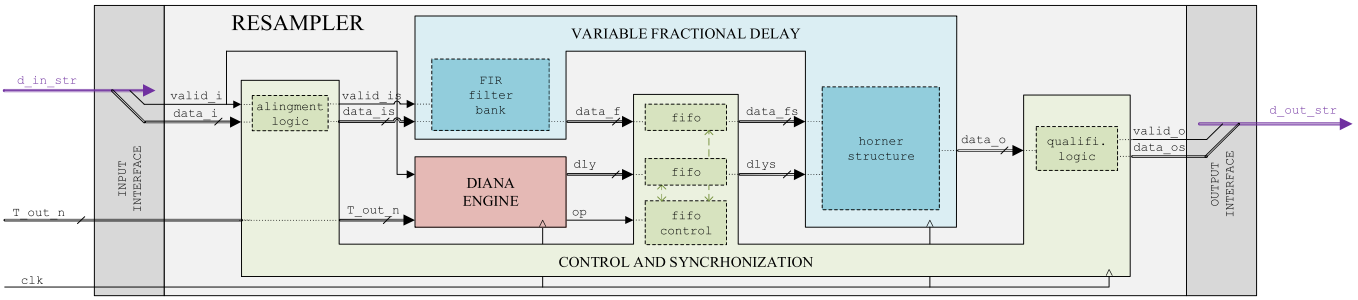


Fig. 12. Functional architecture of the resampler. Data-path decoupled architecture with a VFD filter bank, a DIANA engine and the Control Logic and Synchronization Memories.

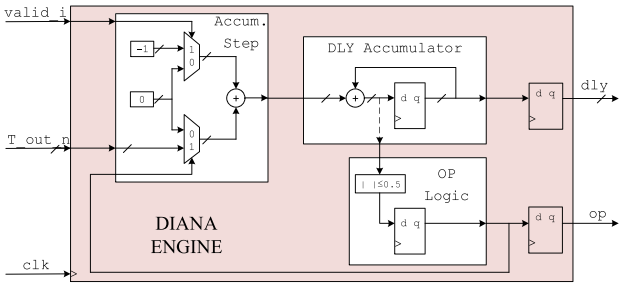


Fig. 13. Implementation of the DIANA engine.

C. The DIANA Engine

This functional unit, depicted in Fig. 13, has three input ports. The first one is the valid signal, `valid_i`, indicating the validity of the data present in the engine input data-path. This indicates a new input sample arriving at the resampler. The second input port is a bus signal, `T_out_n` *(equal to 1/R in normalized units)*, which provides the information about the resampling ratio. Finally, a clock input port receives the processing clock.

At the output, the entity has two ports. The first is the computed delay, `dly`, implemented by means of a bus signal whose width is dependent on the performance (desired precision) of the resampler. The second port contains the `op` signal, which indicates that a new output sample can be computed using the delay value present in the delay bus. The DIANA algorithm of Fig. 11 is implemented by means of an accumulator (Fig. 13).

Two multiplexers evaluate the accumulator step at each iteration. One is controlled by the `valid_i` signal, while the other is driven by feedback logic evaluating the op signal from the last iteration. The output of the multiplexor is accumulated by a dedicated DSP block. The `op` signal is activated when a comparator evaluating the magnitude of the accumulator indicates a value less than or equal to 0.5 (half an input sampling period). The `T_out_n` signal is evaluated at each iteration, for real-time variable and arbitrary ratio operation of the resampler. In case of a fixed ratio resampler, this signal could be hard-wired for resource optimization.

D. The Variable Fractional Delay

Introduced in Fig. 9, the VFD unit implements two functional elements: A bank of FIR filters and the Horner chain of adders and multipliers. The interfacing signals of the entity are as depicted in Fig. 14. In the filter-bank, the `valid_is` signal enables the Tap delay line registers of the bank, when valid data is present in the data-path. The data signal `data_is` feeds the filters of the bank with the new samples in parallel.

In our evaluation system, the bank of filters is composed of $B = 6$ filters with fifteen taps each. The outputs of the filter bank are the B buses `data_f`, containing the pre-filtered available samples.

In the Horner structure, the `data_fs` signals represent B buses containing the outputs of the filter bank, and `dlys` contains the delay value used for the current output sample. These signals need synchronization among themselves, hence the postfix "s". FIFO memories are used for this (in green in Fig. 14). The memories are part of the "Control Logic and Synchronization Memories" entity.

E. The Control Logic and Synchronization Memory

This block contains glue logic and synchronization memories (synchronization is understood as aligning, at the level of processing-clock cycles, different propagating signals) used to manage signals between the different functional blocks. It comprises all the registers, FIFOs, and elements used for synchronization purposes. These are spread within the resampler, but it is still possible to group its signals according to the functional blocks interfaced to, as shown in Fig. 15. Signals related to the data-path alignment at the input of the FIR filter bank are located at the top. The synchronization of delay and data-path by means of FIFOs is shown in the middle. The signals for qualifying the resampler output are placed at the bottom.

All these signals propagate through three different paths, depicted schematically in Fig. 16. The first, in blue, hosts the data-path from the input to the output interface. The second, depicted in red, contains the signals related to the resampling ratio, fed at the input as `T_out_n` and internally translated to delay. Finally, the third path, in green, groups the internal control signals. Alignment between these signal types is required at certain interfaces between the entities. The first interface is at the input of the resampler, where the data-path `d_in_str` and the resampling ratio `T_out_n` must arrive aligned.

The second interface lies at the input of the bank of filters; it requires alignment between the data-path, signals `data_is` and `valid_is`, the computed delay signal `dly`, and the

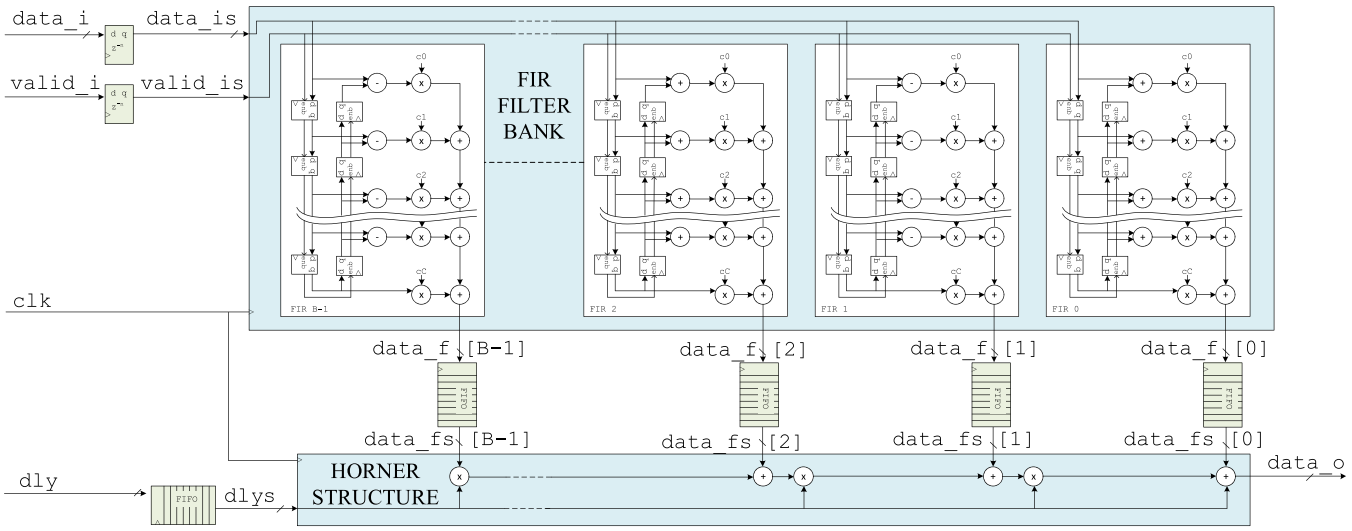


Fig. 14. Architectural view of the VFD filter. In blue, the FIR bank of filters and the Horner architecture. In green, the Synchronization Memories, not part of the entity.

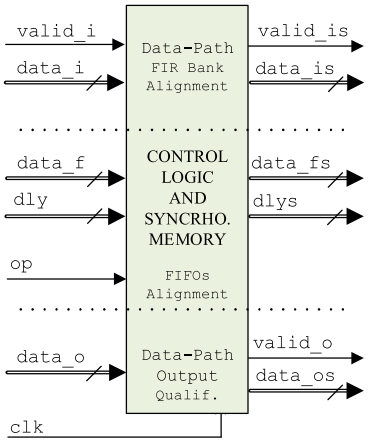


Fig. 15. Input and output signals of the Control Logic and Synchronization Memories for the variable and arbitrary rate resampler architecture.

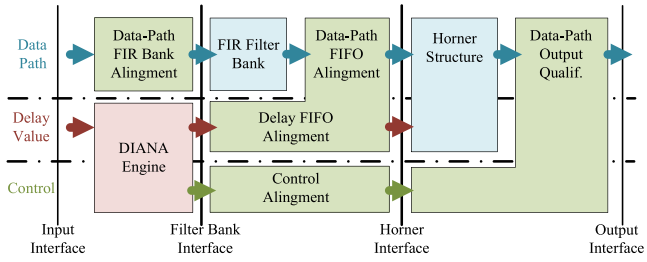


Fig. 16. Synchronization interfaces of the resampler (vertical lines), and signal paths (horizontal arrows).

op control signal. The third interface, at the input of the Horner structure, requires alignment between the data-path buses data_fs filtered by the bank of filters, the delay value dlys, and the internal control signals. Finally, the fourth interface, at the output of the resampler, requires alignment between the data-path data_os, and the valid_o signal qualifying the output stream.

These paths and interfaces inside the resampler, depicted in Fig. 16, have different and unrelated propagation times and latencies. The solutions adopted in [33] or [36] are less sophisticated as the resampling ratio does not vary in operation. The propagation in the data-path is constant for a given ratio, and when changed, re-synthesis of the architecture is needed. That is avoided in the presented solution. The problem solved by the proposed architecture is hence to align and synchronize the arrival of the signals to the interfaces.

In the input interface, the alignment is an external requirement imposed by the resampler to the system hosting it. The synchronization, in the filter bank interface, is a trivial problem. Only glue logic (registers) is used to align the data-path to the output signals of the DIANA engine. This unit has been designed with a latency of three cycles, thus three registers are inserted in the data-path between data_i and data_is, and between valid_i and valid_is. At this point, those signals, the delay value dly, and the control signal op are aligned at processing-clock cycle level, hence ready to be fed to the VFD. The DIANA engine also makes a decision about processing of a new output sample (algorithm in Fig. 11), indicated by op. If possible, and depending on valid_is, the last filtered sample or the new sample ready in the Filter Bank interface will be used to compute a new output. In any case, the current delay value needs also to be synchronized to the input of the Horner structure. The used sample and the propagation of the delay to the Horner interface will require different synchronization mechanisms, depending on the case. The possible scenarios are listed in table I.

In case a new sample needs to be filtered and a new output computed (scenario 4), the alignment mechanism needs to estimate the latency of the sample through the filter bank. This is the time the data-path takes to propagate from the filter bank interface to the outputs of the filters. Then it needs to synchronize the delay propagation with the same latency.

This estimation is a trivial problem in a non-decoupled filtering architecture, or with a fixed resampling ratio. The signal

TABLE I
SCENARIOS, CONTROL SIGNALS AND ACTIONS TO BE DONE

Scenario	valid_is	op	Actions of the VFD
1	0	0	Do nothing
2	0	1	Process new output
3	1	0	Propagate data-path to filter bank
4	1	1	Propagate data-path & process output

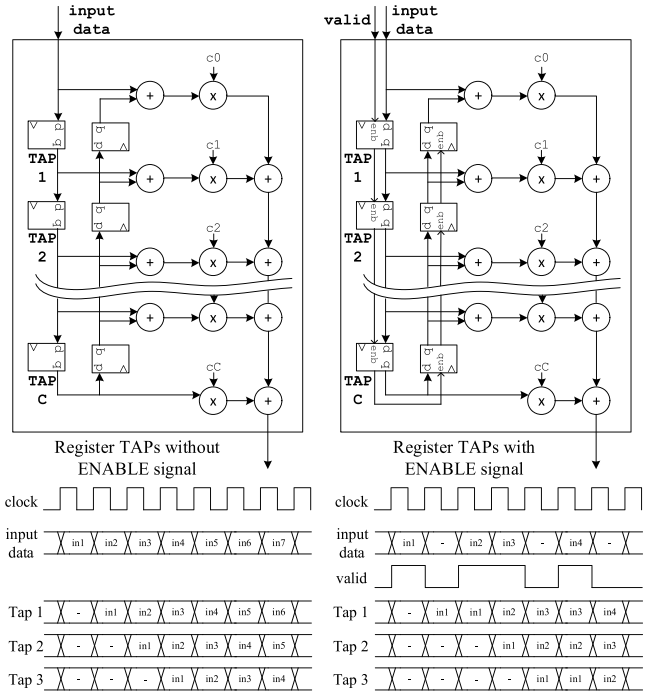


Fig. 17. Tapped delay line data propagation through a classic non-decoupled filter on the left and the decoupled data-path on the right.

advances one register in each clock cycle. This is depicted in the left half of Fig. 17. There, latency is a function of the hardware architectural implementation of the filter (there is no enable signal). This makes the solution for the synchronization of the delay signal easy; the filtering latency can be mimicked by adding the same fixed number of pipeline registers to the delay path. This is not possible for our filtering architecture using a decoupled data-path. The filtering latency varies with the sampling rate. This variation is a direct consequence of the decoupling and the valid signal: The data advances one register further in the filter's tapped delay line only when the enables are active (`valid` signal in Fig. 17 on the right). The data-path propagation through the taps is now driven by the `valid` signal, and the latency (in processing clock cycles) is dependent on its activation rate. The filtering latency, for a given processing clock, is therefore a function of the sampling period, as stated in expression (4). As our architecture must be generic (a resampler accepting different input rates and thus having different filtering latencies), the delay synchronization problem cannot be solved anymore by adding a fixed number of pipeline registers to the delay path.

A second related problem appears in the filter bank due to this driven propagation through the taps. Filtering of one sample requires several samples marked valid (depending on the length of the filter). The active `valid` signal accompanying the initial sample triggers the filtering in the first tap. Then the upcoming samples with the `valid` signal active drive the sample through the output. The pattern of valid samples (and valid flags) can present bursts within the data-path processing slots. In that case, the filter produces consecutive outputs in consecutive processing slots. These bursts need to be handled properly by the synchronization mechanism, otherwise corruption may occur. This is the case if a single filtered sample needs to be re-used twice in the Horner structure, to produce two output samples, with two delay values (scenario 2 followed by scenario 4 at the interface 2). If a burst arrives at that moment, the filter will pop new data, which will remain for one single slot at the output port. This violates the delay-sample combination, since the sample is available for only one of the two processing slots required.

To cope with the propagation based on upcoming samples, and the variable filtering latency, the solution adopted is to use FIFO memories for synchronization. Two banks are used: One stores the delay value `dly`, the `op` and the `valid_is` control signals in the filter bank interface, and the second stores the outputs of the filter bank `data_f` at the output of the filters. The two banks are depicted in Fig. 18 in dark green. The first bank (control signals) compensates the propagation of the data-path through the filter bank. It acts to handle the propagation latency of the delay and control signals (current scenario) between the output of the DIANA engine, the filter bank interface, and the input of the Horner structure (Horner interface).

The second memory bank stores the valid processed outputs of the filter bank. This makes the controlled extraction of filtered samples from the memory possible, reusing them when needed. The scenario previously presented in the filter bank interface can now be reproduced in the Horner interface, by combining the two memory banks. The delay and filtered sample signals are thus fed aligned to the Horner structure, and a new output can be produced. The `Not Empty FIFOs` block in Fig. 18 generates an active signal when both FIFOs contain data; the `Filter Bank Ready` block latches an active signal after filtering of the first sample in the FIR bank.

The propagation between Horner interface and output interface, through the Horner structure, is deterministic; only arithmetic operators implemented in DSP macros are present, and no enable register is used in the pipeline. The latency can thus be estimated and mimicked: The op control signal, depicted in Fig. 18, is propagated in parallel to the data-path with pipeline registers reproducing the delay. At the output of the Horner structure, the synchronized `op` signal qualifies the result of the resampler (signals `data_os` and `valid_o`).

F. Performance and Limitations

The resampling ratio R can take any value but, as mentioned before, the processing clock must be larger than both input and output rates.

Resampling is an interpolation process. Its performance degrades with the frequency of the input signal. The design of the VFD (number of FIR filters and number of taps per

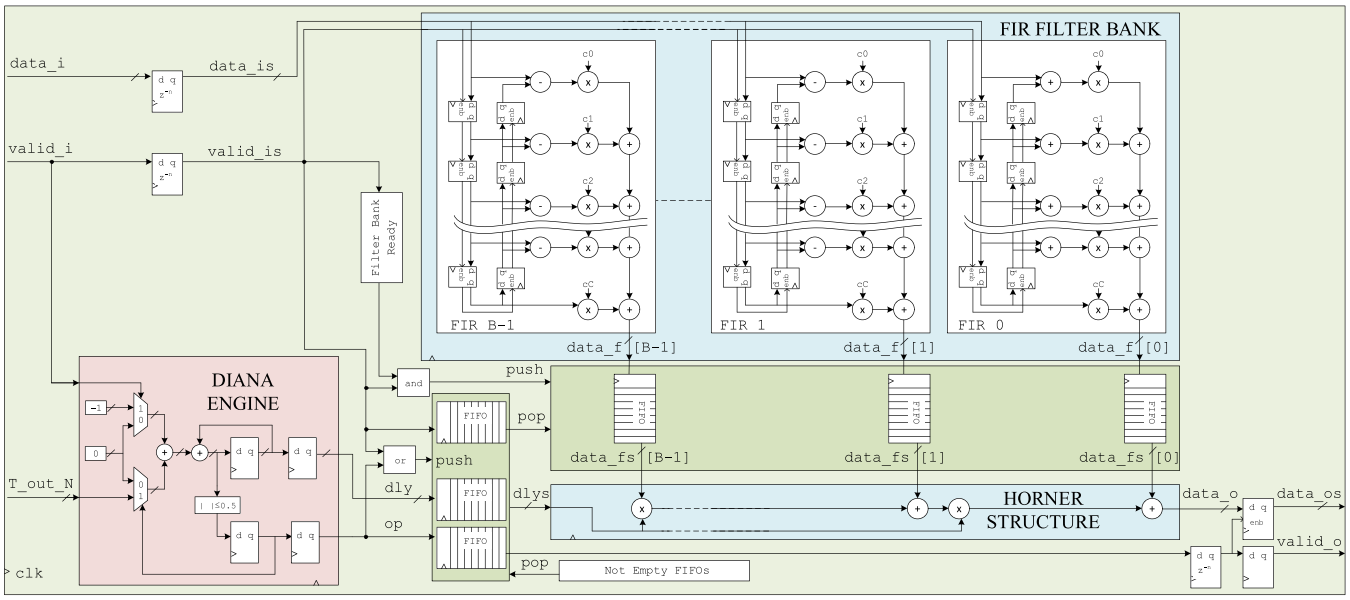


Fig. 18. Detail of the Control Logic, in the light green region, and the Synchronization Memories, in the dark green regions. In the red region the DIANA engine, and in the blue region the VFD filter.

FIR) depends on the demanded precision and passband [22]. We have required a maximum square error of 10^{-9} in the band extending from DC to 60% of the input Nyquist rate.

The delay *dly* is calculated from the instantaneous value of the ratio *R* and used to combine the FIR outputs in the Horner polynomial. This indicates that *R*, which can be modified in real time, should not change significantly during a time interval equal to the latency of the FIR filters.

VI. Validation of the Resampler

This section presents a proof of principle implementation validating the architecture presented in the paper. It solves a classic problem, the enhancement of a periodic signal out of added white noise, found also in the field of particle accelerators. Two resamplers are used in combination with a filter to demonstrate how to implement a SSP system solution.

A. The Test Bench

The presented resampling architecture has first been validated with MATLAB simulations in order to study the resampling algorithm regardless of any implementation details. The results have shown that the algorithm behaves as expected. A functional Simulink model was then used for architectural exploration and its verification. The algorithmic operations and control structures were grouped by functionality, and translated to system level blocks. These were integrated into a functional system level solution. These simulations included the latency of the hardware blocks mapping the data-path and the control structures. No quantification or precision constraints were incorporated at this stage. The functional model was migrated to Xilinx System Generator primitives for hardware verification. The simulation of the architecture and its internal signals with a processing clock cycle accuracy is made possible by the hardware primitives. Finally, the resampling architecture has been integrated into a simulation test bench scenario mimicking a potential real application of the resampler. System level simulations of the proof of concept solution are presented here.

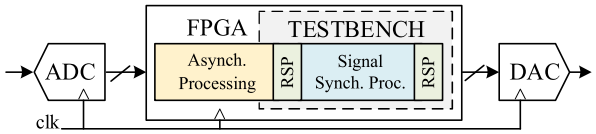


Fig. 19. System level architecture of the application reproduced in the test bench implementation. RSP stands for Resampler.

B. The Application

The objective of the processing in this application is the enhancement of the beam signal out of white noise. For this it uses a periodic comb filter inserted between two resamplers (Fig. 19). The signal consists of a short Gaussian pulse periodic at the beam fundamental frequency plus broadband (white) measurement noise. The frequency of this repetitive pulse changes with time in a known fashion.

The test-bench contains only the FPGA blocks responsible for the resampling and the SSP.

C. The Data-Path

The test-bench architecture hosts, in-between the resamplers, a decoupled data-path SSP region. The first resampler increases the sampling rate of the discrete signal. The resampling ratio is modified in real-time proportionally to the instantaneous revolution frequency of the beam. This operation tunes the discrete representation of the sweeping spectral components of the processed signal to fixed normalized frequencies according to (2). These fixed frequencies are the

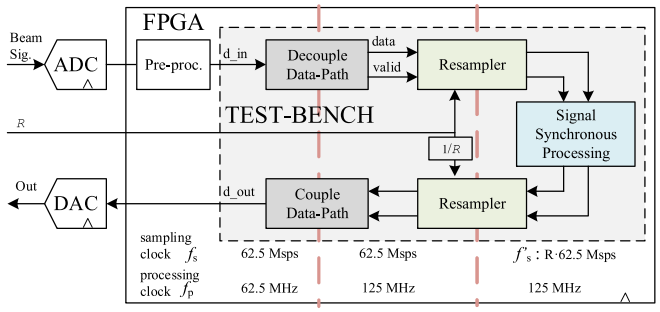


Fig. 20. Data Flow in the proof of concept implementation. In light grey the region of the FPGA emulated in the test bench.

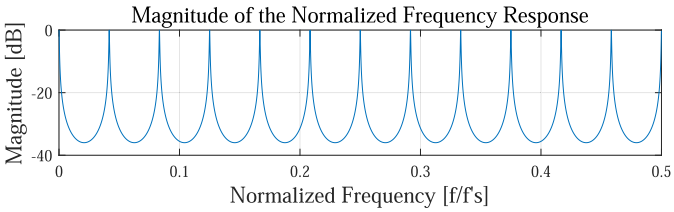


Fig. 21. Magnitude of the filter frequency response normalized to the SSP sampling frequency f'_s.

resonances of the static filter. Finally, the filtered signal is brought back to the original sampling rate by the second resampler. The ratio is also modified in real-time in this resampler with the inverse value of the input ratio.

The data-path is decoupled before the first resampler and coupled back after the second. The decoupling at the input consists of doubling the processing clock. It is also necessary to add a valid signal to the data-path with an activation ratio of $ar = (1/2)$. The coupling performs the complementary transformation: It divides the data-path processing clock by two. This merges sampling and processing clocks at the same rate. The valid line is not necessary thereafter. The resulting data flow is depicted in Fig. 20. The resamplers and SSP filter are implemented in System Generator primitives. The stimulus and other auxiliary system blocks are implemented using Simulink blocks for simplicity.

D. The Input Signal

The input stimulus to the test-bench is the digitized signal fed to the FPGA blocks responsible for the SSP. It is represented by `d_in` in Fig. 20. The output corresponds to the signal `d_out`. The stimulus signal is sampled at $f_s = 62.5$ Msps. The simulation lasts for 500 ms.

At the beginning of the simulation the fundamental frequency is $F_{rev} = 2.6875$ MHz ($\Omega_{rev} = 2\pi \cdot 0.043$ [radian/sample]). At the end of the simulation it has increased by 18.6%, to $F_{rev} = 3.1875$ MHz ($\Omega_{rev} = 2\pi \cdot 0.051$ [radian/sample]). The pulse spectrum contains a DC component, the fundamental frequency and harmonics up to 6 F_{rev}, so that it does not extend beyond the passband used in designing the VFD (section IV.C). A white noise of $\sigma_{RMS} = 0.3$ is added.

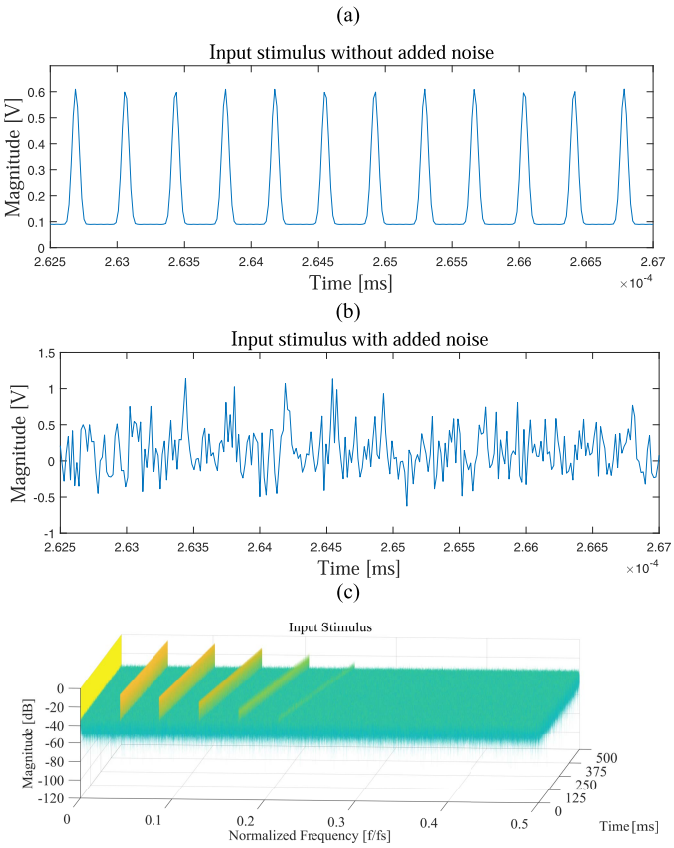


Fig. 22. Stimulus: Time waveform without (a) and with added noise (b) and spectrogram (c). Digital frequencies are normalized to the input sampling rate.

E. The Filters in the SSP

The processing only considers data with `valid_o` flag activated, thus performing SSP at a rate $f'_s(6)$. It implements a periodic comb filter

$$H(z) = \frac{1-a}{1-a\,z^{-N}} \tag{11}$$

with $a = 31/32$ and $N = 24$. The normalized frequency response peaks are located at $\Omega_{peak\_k} = 2\pi \cdot k/24$ [radian/sample]. There are 24 peaks ($k \in [0, 23]$) from DC to the resampling frequency. Fig. 21 depicts the magnitude response of the filter normalized to the SSP sampling frequency f'_s.

After tuning the spectral content of the signal to the filter frequency response (up-sampling with the first resampler), the signal harmonics will be located on the response peaks, while the white noise will be attenuated.

F. The Resampling Ratio in the SSP

The up-sampling ratio, R signal in Fig. 20, tracks in real-time the known instantaneous fundamental frequency F_{rev}. As a computation example, at the start of the simulation, the needed sampling rate which tunes the fundamental frequency to the first filter peak is

$$f'_{s\_init} = 2\pi \cdot F_{rev\_init}/\Omega_{peak\_1} = 2.6875/(1/24) = 64.5 \text{ Msps} \tag{12}$$

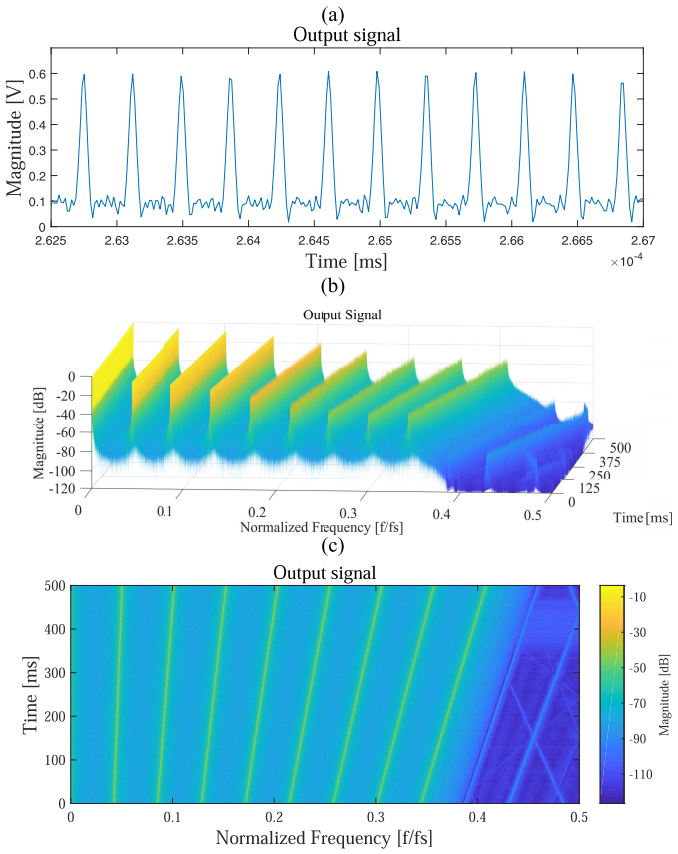


Fig. 23. Time domain (a), spectrogram (b) and top view of the spectrogram (c) at the output of the processing architecture. Digital frequencies are normalized to the input sampling rate.

The resampling ratio is therefore

$$R_{\text{init}} = f'_{\text{s\_init}}/f_{\text{s}} = 64.5 \ / \ 62.5 = 1.032 \tag{13}$$

At the end of the simulation, the needed sampling rate is

$$f'_{\text{s\_end}} = 2\pi \cdot F_{\text{rev\_end}}/\Omega_{\text{peak\_1}} = 3.1875/(1/24) = 76.5\text{Msps} \tag{14}$$

The resampling ratio is then

$$R_{\text{end}} = f'_{\text{s\_end}}/f_{\text{s}} = 76.53 \ / \ 62.5 = 1.224 \tag{15}$$

The resampling ratio is larger than one, resulting in up-sampling in the first stage. After the comb the data will be down-sampled with the risk of aliasing. A Low-Pass filter with design sampling frequency of 64.5 Msps, $F_{\text{pass}} = 0.34$ [radian/sample] and $F_{\text{stop}} = 0.37$ [radian/sample] is added after the comb to prevent this.

G. Results of the Test Bench

Fig. 22 shows the time waveform and spectrogram of the input stimulus signal. In the time domain the Gaussian pulse shows distortion caused by the truncation of its spectrum after the harmonic at 6 F_{rev}. In the frequency plot, the x axis shows the normalized frequency contents of the signal spectrum, the y axis shows the simulation time. The DC component, the fundamental frequency and the harmonics are clearly visible above the floor caused by the added white noise.

Fig. 23 depicts the resulting output signal in both time and frequency domain, after up-sampling, filtering and down-sampling. The white noise has been significantly reduced while the spectral components of the periodic signal are unaffected. Fig. 23(c) shows a color-coded 2D plot of the spectrogram of Fig. 23(b). The signal components are unaffected while the noise is reduced by 36 dB. The noise reduction depends on the attenuation of the comb in the stop-bands (Fig. 21). The effect of the anti-aliasing filter is clearly visible. The spectral components on harmonics seven and eight come from quantization noise in the fixed-coefficients comb filter and can be reduced by using more bits.

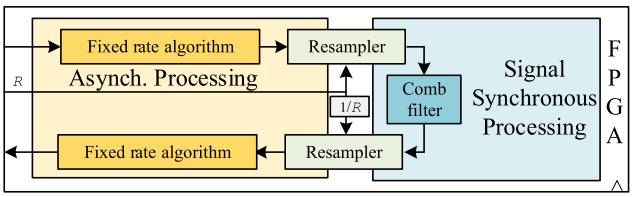


Fig. 24. Data Flow with SSP and Asynchronous (fixed sampling rate) algorithms.

TABLE II
FPGA RESOURCE UTILIZATION AFTER PAR

Resource	Avail.	Used	%	Avail.	Used	%
Slice Registers	94400	4079	4.32	484800	4079	0.84
LUT as Logic	47200	2240	4.74	242400	2238	0.92
LUT as Memories	19000	97	0.51	112800	97	0.08
Block Ram Tiles	105	3.5	3.33	600	3.5	0.58
DSP blocks	180	53	29.4	1920	53	2.76

H. Fixed-Rate Processing

The FPGA where the presented architecture is implemented can, in addition, perform any other fixed-rate processing requiring a fixed sampling clock instead of a sweeping clock. These algorithms can include filtering with a response unrelated to the signal fundamental frequency, for example the compensation of amplifier response in particle accelerators LLRF. There is no restriction in performing this side processing before or after the SSP region in the data-path. A processing data-path hosting both types of filtering is presented in Fig. 24.

I. Implementation Results

The implementation of the testbench was successfully performed with a target 125 MHz processing clock for two devices, a high-grade FPGA, Xilinx Kintex-7 XCKU040-1FFVA1156C and small device Xilinx Artix-7 XC7A75T-2FGG484C. The timing report shows that the maximum achievable clock for the Artix device is 149.9 MHz and

200.8 MHz for the Kintex. The PAR synthesizer strategy was "Optimization of Area".

Table II shows resource utilization results after the PAR. The used glue logic and memory requirements are very limited and negligible for both devices. The DSP blocks are mainly consumed by the bank of filters of the VFD. The data-paths at the input and output of the resampler are 16 bits wide, with 15 fractional part bits. The filter coefficients are 18 bits wide, with 16 fractional part bits. The data-path within the resampler is extended to 24 bits.

VII. Conclusion

The paper has presented a new architectural solution for a Synchronous Sampling Rate Converter (SSRC) with a real time variable and arbitrary ratio. The design is a Farrow-based resampler. It has two main parts: A DIstAnce iN time Algorithm engine (DIANA) calculates the delay between the present available input samples and the desired interpolated outputs. It forwards this result to the Variable Fractional Delay (VFD) consisting of a bank of fixed-coefficient FIRs filtering the input samples. The FIR outputs are then combined after multiplication by powers of the delay calculated by the DIANA engine. The architecture is optimized so that the resampling ratio can be changed continuously. It uses a single fixed-frequency processing clock, thanks to a decoupled data-path that avoids the need for a swept clock as the resampling ratio changes. This makes it perfectly suited for implementation in an FPGA.

This architecture was then demonstrated as a solution for the processing of periodic signals with known but possibly varying fundamental frequency. Two complementary resamplers are combined to create a processing region between them where the sampling period can be swept and a fixed coefficient comb filter is inserted. The resampling ratio is set so that the fundamental frequency of the input signal falls on the first resonance of the comb. The comb then enhances the harmonics and rejects the broadband noise. The second resampler, with the inverse ratio, restores the signal spectrum to the original sampling rate. Instead of adapting the processing to the spectral content of the signal, this method modifies the sampling rate to tune the spectral representation of the signal to a predefined normalized frequency in which the processing is performed with a fixed coefficient filter. This avoids the need for reconfiguration of the processing elements in real time.

The proposed hardware will be implemented on a uTCA platform and evaluated in the CERN Super Proton Synchrotron machine for the compensation of beam induced voltage in a Radio-Frequency cavity during the acceleration ramp. A second paper is being prepared presenting these results. The authors believe that the method can also be applied to other fields for the processing of pseudo-periodic signals.

Future work needs to evaluate the real-time behavior of the architecture. Of prime importance is the tracking capability of the resamplers. How fast can the fundamental frequency vary? This is not an issue for the intended application in particle accelerators as the rate of change is slow, but it may be important for other fields. Frequency tracking is the responsibility of the VFD that includes Finite Impulse Response filters with few taps, resulting in a small latency and suggesting a good tracking capability. This must still be evaluated. Other optimization methods for the VFD coefficients such as the Offset Window method [50], [51] need to be studied. Also, other similar VFD architectures, such as the modified or transposed Farrow [18], could offer some optimization in the hardware implementation.

Acknowledgment

The authors would like to thank CERN colleagues in BE-RF-FB section, plus K. Smith's group at BNL, T. Mastoridis at California Polytechnic State University, C. Rivetta at SLAC and F. Tamura from J-PARC for the many fruitful discussions. Special thanks to A. Butterworth and R. Borner at CERN BE-RF for proofreading the paper.

References

[1] H. Wiedemann, "Circular accelerators," in *Particle Accelerator Physics*, 4th ed. Cham, Switzerland: Springer, 2015, pp. 59–79.

[2] P. Baudrenghien, "Low level RF systems for synchrotrons: Part I and II," in *Proc. CERN Accel. School, Radio Freq. Eng.*, Seeheim, Germany, May 2000, pp. 146–209. [Online]. Available: https://cds.cern.ch/record/386544

[3] U.S. Particle Accelerator School, University of California, Santa Cruz, CA, USA. (Jan. 2008). *Microwave Measurement and Beam Instrumentation Laboratory*. Accessed: Oct. 12, 2018. [Online]. Available: https://uspas.fnal.gov/materials/08UCSC/UCSC-Microwave-Measurements.shtml

[4] D. Boussard and G. Lambert, "Reduction of the apparent impedance of wide band accelerating cavities by RF feedback," *IEEE Trans. Nucl. Sci.*, vol. NS-30, no. 4, pp. 2239–2241, Aug. 1983.

[5] D. Boussard, "Control of cavities with high beam loading," *IEEE Trans. Nucl. Sci.*, vol. NS-32, no. 5, pp. 1852–1856, Oct. 1985.

[6] W. Hofle, "Towards a transverse feedback system and damper for the SPS in the LHC era," *Part. Accel.*, vol. 58, pp. 269–279, Apr. 1997. [Online]. Available: http://cds.cern.ch/record/326868

[7] B. Widrow *et al.*, "Adaptive noise cancelling: Principles and applications," *Proc. IEEE*, vol. 63, no. 12, pp. 1692–1716, Dec. 1975.

[8] A. Nehorai and B. Porat, "Adaptive comb filtering for harmonic signal enhancement," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-34, no. 5, pp. 1124–1138, Oct. 1986.

[9] M. Niedáwiecki and M. Meller, "Generalized adaptive comb filters/smoothers and their application to the identification of quasi-periodically varying systems and signals," *Automatica*, vol. 49, no. 6, pp. 1601–1613, Jun. 2013.

[10] S. C. Pei, Y. D. Huang, S. H. Li, and J. J. Shyu, "Design of variable comb filter using FIR variable fractional delay element," *Signal Process.*, vol. 92, no. 10, pp. 2409–2421, Oct. 2012.

[11] M. Benedikt, P. Collier, V. Mertens, J. Poole, and K. Schindl. (Dec. 2004). *LHC Design Report. CERN*. vol. 3, p. 130. [Online]. Available: http://cds.cern.ch/record/823808

[12] R. E. Crochiere and L. R. Rabiner, "Sampling rate conversion—An analog interpretation," in *Multirate Digital Signal Process*. Upper Saddle River, NJ, USA: Prentice-Hall, 1983, pp. 22–29.

[13] D. Manolakis and V. Ingle, *Applied Digital Signal Processing*. New York, NY, USA: Cambridge Univ. Press, 2011, pp. 706–727.

[14] R. Adams and T. Kwan, "Theory and VLSI architectures for asynchronous sample-rate converters," *J. Audio Eng. Soc.*, vol. 41, nos. 7–8, pp. 539–555, Jul. 1993.

[15] T. I. Laakso, V. Valimaki, M. Karjalainen, and U. K. Laine, "Splitting the unit delay," *IEEE Signal Process. Mag.*, vol. 13, no. 1, pp. 30–60, Jan. 1996.

[16] F. Harris, *Multirate Signal Processing for Communication Systems*. Upper Saddle River, NY, USA: Prentice-Hall, 2004.

[17] T. Ramstad, "Digital methods for conversion between arbitrary sampling frequencies," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 32, no. 3, pp. 577–591, Jun. 1984.

[18] A. Franck, "Efficient algorithms for arbitrary sample rate conversion with application to wave field synthesis," Ph.D. dissertation, Tech. Univ. Ilmenau, Ilmenau, Germany, 2012. [Online]. Available: http://publica.fraunhofer.de/documents/N-219863.html

[19] J. Vesma and T. Saramaki, "Polynomial-based interpolation filters—Part I: Filter synthesis," *Circuits, Syst. Signal Process.*, vol. 26, no. 2, pp. 115–146, Apr. 2007.

[20] M. Blok, "Fractional delay filter design for sample rate conversion," in *Proc. Federated Conf. Comput. Sci. Inf. Syst. (FedCSIS)*, Wroclaw, Poland, Sep. 2012, pp. 701–706.

[21] H. Johansson and O. Gustafsson, "Linear-phase FIR interpolation, decimation, and mth-band filters utilizing the farrow structure," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 10, pp. 2197–2207, Oct. 2005.

[22] T.-B. Deng and Y. Lian, "Weighted-least-squares design of variable fractional-delay FIR filters using coefficient symmetry," *IEEE Trans. Signal Process.*, vol. 54, no. 8, pp. 3023–3038, Aug. 2006.

[23] B. H. Tietche, O. Romain, and B. Denby, "A practical FPGA-based architecture for arbitrary-ratio sample rate conversion," *J. Signal Proces. Syst.*, vol. 78, no. 2, pp. 147–154, Feb. 2015.

[24] H. Ramon, H. Li, P. Demeester, J. Bauwelinck, and G. Torfs, "Efficient parallelization of polyphase arbitrary resampling FIR filters for high-speed applications," *J. Signal Proces. Syst.*, vol. 90, no. 3, pp. 295–303, Mar. 2018.

[25] M. Blok and P. Drózda, "Variable ratio sample rate conversion based on fractional delay filter," *Arch. Acoust.*, vol. 39, no. 2, pp. 231–242, 2014.

[26] G. Evangelista, "Design of digital systems for arbitrary sampling rate conversion," *Signal Process.*, vol. 83, no. 2, pp. 377–387, Feb. 2003.

[27] A. Chinaev, P. Thune, and G. Enzner, "Low-rate Farrow structure with discrete-lowpass and polynomial support for audio resampling," in *Proc. 26th Eur. Signal Proces. Conf. (EUSIPCO)*, Rome, Italy, Sep. 2018, pp. 475–479.

[28] C. W. Farrow, "A continuously variable digital delay element," in *Proc. IEEE Int. Symp. Circuits Syst.*, Espoo, FI, Finland, Jun. 1988, pp. 2641–2645.

[29] T. Hentschel, *Sample Rate Conversion in Software Configurable Radios*. Norwood, MA, USA: Artech House, 2002.

[30] J. P. Long and J. A. Torres, "High throughput Farrow re-samplers utilizing reduced complexity FIR filters," in *Proc. IEEE Military Commun. Conf. (MILCOM)*, Orlando, FL, USA, Oct. 2012, pp. 1–6.

[31] T. Hentschel and G. Fettweis, "Sample rate conversion for software radio," *IEEE Commun. Mag.*, vol. 38, no. 8, pp. 142–150, Aug. 2000.

[32] F. Sheikh and S. Masud, "Efficient sample rate conversion for multi-standard software defined radios," in *Proc. Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Honolulu, HI, USA, Apr. 2007, pp. II329–II332.

[33] B. Markovic and J. Certic, "FPGA realization of Farrow structure for sampling rate change," *Serbian J. Elect. Eng.*, vol. 13, no. 1, pp. 83–93, 2016.

[34] A. Tkacenko, "Variable sample rate conversion techniques for the advanced receiver," *NASA Interplanetary Netw. Prog. Rep.*, vol. 42, p. 168, Feb. 2007. [Online]. Available: https://ipnpr.jpl.nasa.gov/progress_report/42-168/168A.pdf

[35] M. A. Siddiqi, N. Samad, S. Masud, and F. Sheikh, "FPGA-based implementation of efficient sample rate conversion for software defined radios," in *Proc. 10th IEEE Int. Conf. Comput. Informat. Technol.*, Bradford, U.K., Jun. 2010, pp. 2387–2390.

[36] Altera Corporation, San Jose, CA, USA. (Aug. 2010). *AN 623: Using the DSP Builder Advanced Blockset to Implement Resampling Filters*. Accessed: Feb. 10, 2019. https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/an/an623.pdf

[37] U. Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays*, 3rd ed. Berlin, Germany: Springer-Verlag, 2007, pp. 290–296.

[38] R. Adams and T. Kwan, "A stereo asynchronous digital sample-rate converter for digital audio," *IEEE J. Solid-State Circuits*, vol. 29, no. 4, pp. 481–488, Apr. 1994.

[39] Analog Devices. *ADV7180 Datasheet*. Accessed: Feb. 14, 2019. [Online]. Available: http://www.analog.com/en/products/adv7180.html

[40] F. M. Gardner, "Interpolation in digital modems. I. Fundamentals," *IEEE Trans. Commun.*, vol. 41, no. 3, pp. 501–507, Mar. 1993.

[41] F. Takahata, M. Yasunaga, Y. Hirata, T. Ohsawa, and J. Namiki, "A PSK group modem for satellite communications," *IEEE J. Sel. Areas Commun.*, vol. SAC-5, no. 4, pp. 648–661, May 1987.

[42] G. Maalouli and D. R. Stephens, "Joint, fractional resampler with delay equalization for high synchronization accuracy with a reduced number of samples per symbol," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Montreal, QC, Canada, May 2004, pp. V349–V352.

[43] P. Beckmann and T. Stilson, "An efficient asynchronous sampling-rate conversion algorithm for multi-channel audio applications," *Audio Eng. Soc. Conv.*, vol. 119, pp. 1–15, Oct. 2005.

[44] H. Zhang and C. Zhu, "A filter structure for arbitrary re-sampling ratio conversion of a discrete signal," *Information*, vol. 8, no. 2, p. 53, May 2017.

[45] K. Rajamani, Y.-S. Lai, and C. W. Furrow, "An efficient algorithm for sample rate conversion from CD to DAT," *IEEE Signal Process. Lett.*, vol. 7, no. 10, pp. 288–290, Oct. 2000.

[46] S. Park, G. Hillman, and R. Robles, "A novel structure for real-time digital sample-rate converters with finite precision error analysis," in *Proc. Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Toronto, ON, Canada, Apr. 1991, pp. 3613–3616.

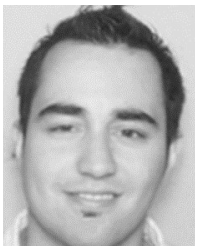
[47] F. J. G. Guarch, J. M. M. Arostegui, and P. Baudrenghien, "Compensation of transient beam loading in ramping synchrotrons using a fixed frequency processing clock," in *Proc. J. Phys., Conf. Ser., Beam Instrum.*, vol. 1067, Sep. 2018, Art. no. 072033.

[48] A. DeHon *et al.*, "Design patterns for reconfigurable computing," in *Proc. 12th Annu. IEEE Symp. Field-Program. Custom Comput. Mach.*, Napa, CA, USA, Apr. 2004, pp. 13–23.

[49] D. E. Knuth, "Evaluation of polynomials," in *The Art Computer Programming*, 3rd ed. Boston, MA, USA: Addison-Wesley, 1997, pp. 485–486.

[50] A. Yardim, G. D. Cain, and P. Henry, "Optimal two-term offset windowing for fractional delay," *IET Electron. Lett.*, vol. 32, no. 6, pp. 526–527, Mar. 1996.

[51] M. Blok, "Fractional delay filter design with extracted window offsetting," in *Proc. 19th Int. Conf. Mixed Des. Integr. Circuits Syst. (MIXDES)*, Warsaw, Poland, 2012, pp. 489–494.



Fco. Javier Galindo Guarch received the M.Sc. degree in telecommunications engineering from the Universidad de Zaragoza, Zaragoza, Spain, in 2007, and the M.Sc. degree in electronic engineering from the Department of Electronic Engineering, Universitat Politècnica de Catalunya, Barcelona, Spain, in 2015, where he is currently pursuing the Ph.D. degree in collaboration with CERN. His research interests include microelectronics and its application to low level RF systems for particle accelerators.

Philippe Baudrenghien received the Ph.D. degree in electrical engineering from Stanford University in 1984. Since 1985, he has been with the European Centre for Particle Research (CERN) Beams Department, Geneva, Switzerland, in the Radio-Frequency Group. He has worked on the SPS, Linac4, and LHC accelerators. His contributions are in the field of accelerator physics (upgrade of existing and design of new accelerators, namely HL-LHC and FCC) and in the field of electronics (design of feedback systems for the control of accelerator beam).

Juan Manuel Moreno Arostegui received the Ph.D. degree in telecommunications engineering from the Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, in 1994. He is currently an Associate Professor with the Department of Electronic Engineering, UPC. His research interests include analog and digital VLSI design, neural networks models, architectures for programmable devices and systems on chip, bio-inspired computing techniques, and electronic systems for e-health infrastructures.