Real-Time Downsampling in Digital Storage Oscilloscopes With Multichannel Architectures

Ettore Napoli, *Senior Member, IEEE*, Efstratios Zacharelos, Mauro D'Arco, *Senior Member, IEEE*, and Antonio Giuseppe Maria Strollo, *Senior Member, IEEE*

***Abstract*— Digital Storage Oscilloscopes (DSOs) conjugate high performance with large number of features and flexibility. The basic structure, based on fast Analog to Digital Converter (ADC) and memory, is augmented with several components for channel matching and bandwidth improvement, and processors that provide visualization, frequency processing, jitter and stability measurement, etc. Unfortunately, fine resolution in sample rate selection is not available, such that for several applications the user must run complex measurement procedures that require data download and offline processing. The paper proposes a dedicated digital circuit that offers fine control of the time-base by real time downsampling the input stream at an almost arbitrary sampling rate. The proposed circuit implements a series of operations involving real-time data filtering, defragmentation and packing, which are not considered by alternative approaches, like those based on polyphase filters, that offer very limited choices for the sampling rate. The circuit is designed to work in conjunction with the highest performance DSOs that use a multichannel architecture. Design rules for circuit design are provided together with implementation results in 14nm FinFET technology. When designed for an architecture with 64 channels with 8bit input samples the circuit works in real time with a sampling rate of 220 GSps running at 3.42 GHz, with a silicon footprint of 0.17 mm^2 and a power dissipation of 0.85W.**

***Index Terms*— Digital circuits, signal sampling, oscilloscopes, digital filters, interpolation, parallel processing, data storage, storage management, resampling, digital storage oscilloscope.**

I. Introduction

STATE of the art circuit design and test requires state of the art measurement systems that match or overcome the speed and the bandwidth of the designed circuits [1]. Digital Storage Oscilloscopes (DSOs) are the backbone of circuit testing and need to provide always higher bandwidth and resolution to allow the measurement of newly designed circuits [2], [3].

The scientific literature routinely reports the advancements in DSO technologies. Most of them leverage on the capabilities of digital circuits, to extend and/or improve the functionalities of this general-purpose instrument. Various contributions deal with the very complex software structure of the instruments that are equipped with complete operating systems and a series of dedicated software modules, [4]–[6], while other research activities are oriented to the whole circuital structure of the DSO, [7]–[23]. Many papers study and improve the front-end of the DSO as this is the portion that determines the sampling rate and the bandwidth, [7]–[11]. Other contributions propose various sampling methods and real-time or off-line processing that, exploiting the peculiarities of the signals and the front-end, allow an extension of the bandwidth or an improvement of the resolution, [12]–[19]. The scientific literature also provides complete DSO circuits either implemented with FPGA, microcontrollers, or general purpose processors, [20]–[22], or analyzes the possible structures in terms of performance, [23].

An up-to-date single acquisition channel of high-end DSOs offers up to 33 GHz input electrical bandwidth and 100 GSps maximum sample rate (Tektronix DPO/MSO 73304DX oscilloscopes offer 33 GHz bandwidth and 100 GS/s maximum sample rate [24], while Teledyne-Lecroy WaveMaster 830Zi-B offers 30 GHz Bandwidth, and 80 GS/s maximum sample Rate [25]). Patented technologies that exploit a combination of channel resources, such as Asynchronous Time Interleaving (ATI) or Digital Bandwidth Interleaving (DBI), attain input bandwidth up to 70 GHz and sample rates up to 160 GSps (ATI), or bandwidth in excess of 100 GHz and sample rates over 240 GSps (DBI) [26]–[29].

The focus of this paper is on the traditional time interleaving (TI) technology, which is the pioneering technology exploiting channel resources combination; its bandwidth performance is limited by the baseband nature of the digitizing system. DBI and ATI technologies introduce heterodyning to support bandwidth extensions: DBI uses asymmetrical channels requiring synchronization and massive processing that impact complexity and cost of the DSO, whereas ATI deploys a compact and low noise architecture whose bandwidth performance is intermediate between TI and DBI technologies.

In time-interleaved structures, [30]–[37], in simple terms, L (8 to 48) ADCs acquire the input signal at the same rate (up to 5 GSps), with a time-offset equal to $1/L$ of the sample period, so that the aggregate throughput is L times the sample rate of the individual ADCs. The general structure of the processing chain, for a single channel, of a DSO is shown in Fig. 1. Each ADC in a time-interleaved structure is complemented with an ASIC to carry out real-time operations

Manuscript received September 18, 2020; revised March 2, 2021; accepted July 28, 2021. Date of publication August 13, 2021; date of current version September 30, 2021. This article was recommended by Associate Editor I. Kale. *(Corresponding author: Ettore Napoli.)*

The authors are with the Department of Electrical and Information Technology Engineering, University of Napoli Federico II, 80125 Naples, Italy (e-mail: ettore.napoli@unina.it).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSI.2021.3102386.

Digital Object Identifier 10.1109/TCSI.2021.3102386

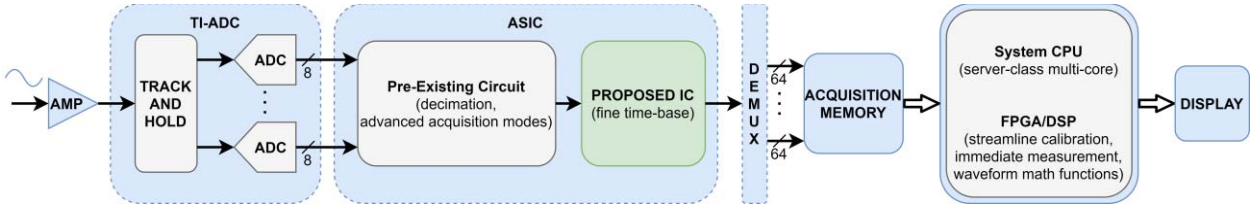

Fig. 1. Typical structure of the acquisition channel of a high-performance DSO. A given DSO could have different bus width toward the acquisition memory and/or the DEMUX positioned before the proposed IC. Capabilities and complexity of each element is widely changing depending on the brand and the performance.

on the sample stream. The ASIC implements the time-base of the DSO that allows a (coarse) selection of the sample rate through real-time decimation, i.e. by downsampling by an integer factor the sample stream at the highest sample rate. In more advanced solutions, it even includes several hardware machines that enable a variety of real-time functions and support alternative acquisition modes such as peak detection, high-resolution, averaging, etc. Generally, the ASIC is in charge of critical processing operations that need real-time constraints but affords only a small part of all processing required in DSOs, that involves streamline calibration, display operations, and waveform math functions ranging from immediate measurements to more complex algorithms like FFTs. Such additional processing is carried out by means of server-class multi-core processors, or FPGA circuits (advanced trigger detection) integrated in the instrument. The typical processing chain often includes a DEMUX that interfaces the ASIC with the acquisition memory and balances between the fast rate of ADCs and the ASIC with the slower access rate of the acquisition memory. Namely, the DEMUX retrieves the samples from ASIC, collects them into bunches, and writes single bunches with memory accesses at a slower rate.

Hereinafter, the attention is paid to the limits of the time-base system implemented at the very heart of the digitizing system in DSOs, namely at circuit level by the ASIC. The main intent of the work is to improve the time-base that at present offers only a discrete set of sample rates rather than continuously variable control. Note that fine control of the time-base has always been a critical issue. Manufacturers of analog oscilloscopes allowed the user to dispose of a limited set of calibrated settings or use a continuously variable control (accepting uncalibrated speeds), whereas, in early DSO models, they offered the possibility of using an external frequency variable clock to drive the time-base circuit within limited frequency ranges, warning the user that the overall system accuracy was no more granted by the specifications declared in the DSO documentation. The reader interested to further investigate time-base issues in DSOs is addressed to [38]–[44].

As previously stated, modern DSOs use a digital circuit deployed between ADC and acquisition memory to seamlessly decimate the digitized signal. Decimation consists in singling out one among N consecutive samples from the ADC sample stream, where N is the decimation factor. In this way the ADC operates at the fixed maximum frequency, whichever the sample rate selected by the user. The rationale for this widely adopted design choice is substantially related to a couple of benefits. Specifically, the use of a fixed reference frequency simplifies buffer design and memory access rate regulation needed for the data transfer from ADC to acquisition memory. Also, the jitter performance is made independent of the chosen sample rate. This approach however prevents an optimal usage of the memory resources and control of the time interval capture, since it does limit the possible choices of the sample rate, which are typically constrained to a few values per decade. In other terms, the decimation logic adopted in ASIC appears restrictive when the user needs real-time control of the time interval capture in waveform analysis. For instance, rigid sample rate selection prevents setting coherent sampling in the analysis of steady alternate waveforms, as well as precludes fine control of the frequency span and resolution in FFT analyses, which claim for sample rate adaptation through off-line processing. On the counterpart, the decimation logic is rewarding in terms of sampling jitter containment: as a matter of fact, the performance granted by time base systems operating with variable frequency clocks is by far inferior to that of systems with fixed frequency clocks; at the state-of-the-art, the latter can assure 100 fs rms (typical) timing jitter, i.e. 0.2% of a clock period at 5 GHz. This is the typical sample clock jitter specified by the Oscilloscopes Teledyne-Lecroy Wavemaster 8 Zi B 4–30 GHz, [25].

A proposal to enable an almost arbitrary selection of the sample rate in DSOs is therefore presented in this paper. It consists in real-time downsampling the ADC output stream by an arbitrary non-integer factor, and it is implemented by means of a digital circuit that improves the processing capabilities of the ASIC. It should be noted that the proposed circuit is conceived as an add-on for the existing processing machines. It does not perform by itself the streamline calibration operation aimed at mitigating the static interleaving mismatches, which is carried out by other DSP machines.

The proposal is inspired by a previous research work that led to the realization of a digital circuit capable of downsampling by a non-integer factor the stream of an individual channel at a maximum clock frequency up to a few GHz [45]–[47]. Such circuit uses a finite impulse response filter with dynamically varying coefficients and an effective strategy to occasionally discard samples for reducing the sample rate. The circuit proposed in this paper is however completely different being oriented to a system based on many interleaved ADCs (channels from now on). The multi-channel architecture needed the circuital implementation of a novel algorithm, as will be shown in the following, that requires the complex phases of

defragmenting and packing, placed after the not trivial linear interpolation. In addition, the proposed circuit allows a sample rate that overcomes the needs of the high-performance DSOs available to date.

Sample rate selection could be implemented exploiting conventional resampling techniques such as interpolation or polyphase filters that counteract aliasing effects by means of explicit or implicit low pass filtering operations. Unfortunately, the direct hardware implementation of both approaches is only effective if the resampling factor is fixed, whereas it becomes cumbersome or impossible if the resampling factor is selected ad lib, since different sample rate selections require different circuit layouts. More in detail polyphase structures represent a standard solution for fixed sample rate conversion and would finely match the parallel structure of time interleaved ADCs, but they are unfeasible for the considered application, where the user can make arbitrary selections of the sample rate, each one requiring a different structure. Alternatively, programmable components, such as FPGAs, could be used to adapt the polyphase structure to the user required sample rate but they offer insufficient speed and put limits on the selectable sample rates, excluding all the settings demanding polyphase structures that exceed the available resources. Artificial intelligence-based approaches have also been successfully adopted to cope with resampling tasks [48], [49]. Unfortunately, their typical computational burden requires massive resources and, most importantly, the paradigm underneath their operation principles lacks the determinism which is required in measurement instruments.

The proposed circuit exploits instead original approaches that allow: parallelizing the serial computation presented in [45]–[47]; distinguishing valid from dummy samples in the raw bunches collecting the time-interleaved ADCs outputs; defragmenting the raw bunches and compacting valid and invalid samples in separate bunches of fixed size; discarding the invalid bunches while saving the valid ones in the acquisition memory.

The proposed circuit has real-time performance, hence produces no impact on the capture rate, differently from software post-processing, which involves detrimental blind intervals. Moreover, it can grant the jitter performance of the internal clock to all the selectable sample rates.

The paper is organized as follows. In Section II the resampling algorithm is presented while Sections III and IV describe the proposed circuit and detail its performances.

II. Resampling Algorithm

In the proposed system, the resampling algorithm receives, at any clock tick, a bunch of L samples, each one on 8 bits, where the clock rate is the one of the time-interleaved ADCs, i.e., f_{ck} (period T_{ck}). The output is a version of the input signal resampled at a sample rate $f_s = C\ f_{ck}$, where the resampling factor C is an arbitrary (within the limits of the numeric precision detailed in Appendix II) fractional value in the interval [1/2, 1) selected by the user, [45]. It is produced as a sequence of valid bunches, which are saved in the acquisition memory, and invalid bunches that are instead discarded, thus lowering, as required, the sample rate.

It is worth noticing that resampling is usually performed after low-pass filtering/interpolation operations aimed at counteracting the aliasing effects. For a digital input spectrum $X(\upsilon)$, where υ is the frequency in hertz normalized to f_{ck}, the resampled version, $Z(\upsilon)$, characterized by sample rate Cf_{ck}, can analytically be described as:

$$Z(\upsilon) = \sum_{p,q=-\infty}^{\infty} \frac{1}{C} sinc^2\left(\frac{\upsilon}{C}\right) X\left(\frac{\upsilon - p}{C} - q\right) \quad (1)$$

where $\frac{1}{C} sinc^2\left(\frac{\upsilon}{C}\right)$ accounts for the low-pass filter effect due to the linear interpolation. From (1) the alias-free version of the resampled signal is obtained using $p = q = 0$, whereas all the combinations satisfying $|p - Cq| < C$ single out and permit quantifying the alias contributions that fall within the bandwidth of the signal. Equation (1) permits to quantify the typical strength of the alias contributions and show that they are generally comparable to the quantization noise floor of an 8 bit digitizer. Furthermore, it can even be shown that the effective number of bits of the digitizer (ENOB) improves with respect to the nominal number of bits when the input frequencies are substantially lower than the analog bandwidth of the digitizer. The improvement is an effect of the low-pass filter implicit in the interpolation that restricts the acquired bandwidth and thus the quantization noise. In fact, the low-pass filter due to linear interpolation is characterized by a 3 dB attenuation bandwidth that is only 64% times f_{ck}. But, as the input frequency approaches the upper limit of the Nyquist bandwidth, i.e. for C approaching 1, the performance of the linear interpolator decreases, such that the use of the resampling approach could be unsuitable for the measurement goal according to the requirements in terms of ENOB. A more detailed analysis of linear interpolators performance is given in [47].

Fine sample rate selection can be straightforwardly extended by combining the hard decimation (by integer value) capability of the ASIC with the operation of the proposed circuit.

The algorithm is illustrated in the following by firstly recalling the single channel algorithm presented in [45]–[47], and then detailing the operations of the multichannel algorithm that develop through three stages named: interpolation, defragmentation, and packing.

A. Single Channel Algorithm

The single-channel algorithm reads one sample at its input and writes one sample at its output, as illustrated in [45]–[47] where it is shown that the algorithm and the architecture are feasible, but the maximum clock frequency is limited to a few GHz, hence the sampling rate is in the GSps range.

The circuit implementation of the single channel algorithm is shown in Fig. 2. It is a FIR filter with time variable coefficients (vcFIR) that processes the digital signal $x(n)$ deriving from the ADC, and produces the output, $y(n)$. The linear interpolation provides a good compromise between performances, which enlist both accuracy of the output results, costs of circuitry resources, and feasibility of the resampling scheme, [47]. Therefore, the value $y(n)$ is the linear

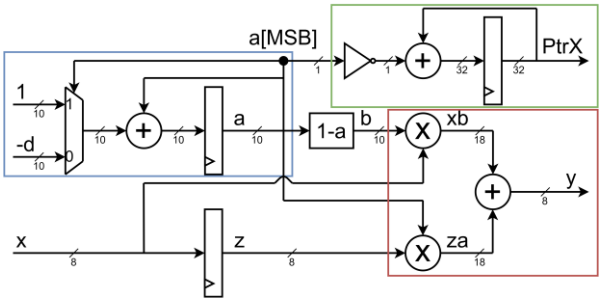

Fig. 2. Single channel architecture proposed in [47]. Blue rectangle: updating of a signal. Green rectangle: updating of the memory pointer. Red rectangle: linear interpolation (FIR block).

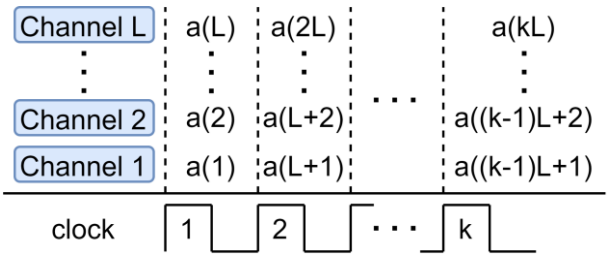

Fig. 3. Parallel channels generating the a-sequence. The figure shows the behavior of L channels for k clock cycles. Each channel must calculate $a(L+i)$ from $a(i)$ without waiting for the calculation of $a(L+i-1)$.

interpolation of the samples $x(n)$ and $x(n-1)$:

$$y(n) = (1 - a(n))\, x(n) + a(n)\, x(n-1) \tag{2}$$

where $a(n)$ is a time-varying coefficient, normally in the range [0, 1). This coefficient quantifies in T_{ck} units the time delay of the system clock tick that follows the sampling instant required by the selected sample rate f_s. It is computed recursively at every clock cycle defining:

$$d = C^{-1} - 1 \tag{3}$$

(C is the chosen resampling factor) and decrementing $a(n)$:

$$a(n) = a(n-1) - d \tag{4}$$

but, if the accumulated subtractions have produced a negative value for $a(n-1)$, the decrement is skipped, and the current coefficient is obtained incrementing by 1 the negative value:

$$a(n) = a(n-1) + 1 \tag{5}$$

thus restoring $a(n)$ to its normal [0, 1) range and connotation as a time delay. This case is referred to as an exception. In fact, negative values for the coefficient a, occur anytime a couple of consecutive system clock ticks fall within the same sampling interval, i.e. anytime that a clock tick expected after the sampling instant leads instead that sampling instant, such that a negative delay is returned by the recursive formula. All the samples, $y(n)$, returned by the circuit in the presence of a negative coefficient, are invalid and not saved in the acquisition memory, thus creating a lower rate stream and saving memory, while the rest of them form the valid downsampled signal. Specifically, the circuit manages the acquisition memory by means of a pointer that is always incremented by one, except for the cases that follow a negative a coefficient (exceptions). The invalid samples calculated using a negative a coefficient, are thus overwritten, whereas the valid samples are stored in consecutive locations in the memory.

B. Multichannel Algorithm

The multichannel architecture receives a bunch of L samples deriving from the ADCs at the system frequency f_{ck} and produces a bunch of L interpolated samples at the same rate. To cope with the lower rate of the output stream, the multichannel architecture provides, with a certain pace, an invalid bunch that is not saved into the memory.

In order to implement the downsampling operation in the multichannel architecture, one vcFIR filter for each of the L channels, is necessary. Furthermore, suitable solutions capable of considering both the time varying character of the coefficients and the need for disposing of the invalid samples must be defined. These solutions should update in real-time the coefficients of the vcFIR filters deployed on each individual channel and dedicated to the interpolation task, defragment each bunch singling out the invalid samples, assemble valid and invalid bunches in a sequence, such that discarding the invalid bunches permits downsampling the input bunched stream.

A diagram of the proposed algorithm can be seen in the actual circuital implementation shown in Fig. 7. The multichannel algorithm is composed by i) interpolation: operates in parallel on a bunch of samples and produces a set of interpolated valid and invalid samples; ii) defragmentation: operates on the set of samples sweeping on one side the invalid samples; iii) gluing: counts the number of valid samples in a set and provides this information to the next phase; iv) packing: operates on three sets of defragmented samples after gluing to pick the valid samples and generate a set of all-valid or all-invalid samples.

1) Interpolation: In the multichannel version an interpolated sample is calculated following the principle adopted in the single channel system, thus calculating the value of the time-varying coefficient $a(n)$ and using (2) to calculate the output, $y(n)$. The discrete time variable n stems the samples characterized by sampling rate $L \cdot f_{ck}$ (sampling period, T_{ck}/L), where f_{ck} is the sampling rate (T_{ck} sampling period) of each channel.

Each channel is responsible for producing the correct value $a(n)$ which follows the value $a(n-1)$ calculated by the adjacent channel, as explained in Fig. 3, and combining the samples $x(n-1)$ and $x(n)$. But, since parallelism is of the essence, each channel cannot wait to pick-up the value $a(n-1)$ produced by the adjacent channel to determine the required value $a(n)$. Therefore, each channel must be configured to work as an independent channel.

To this end, the initial set of coefficients for the discrete time instants ranging from 1 up to L, i.e. $\{a(1), \ldots, a(L)\}$, needed by the L channels to elaborate the first bunch of samples, is calculated in advance according to the settings on the desired C value made by the user. A detailed explanation on how to calculate the initial set of coefficients is in the Appendix I.

Once the initial set of coefficients is configured and given in input, the interpolation algorithm is responsible for carrying on and calculating the next sets for the elaboration of the following bunches. Each channel must produce a value for the coefficient a that is L steps ahead just exploiting the current value it holds. Using the discrete time variable k to stem the samples characterized by the sampling rate f_{ck} of each channel, and the parameter $l \in [1, \ldots, L]$ to address the generic channel one has:

$$a\left(kL+l\right) = a\left(\left(k-1\right)L+l\right) - \left(L-m\right)d + m \tag{6}$$

where d is the decrement to be applied defined in II-B ($0< d \leq 1$), and m is the number of exceptions occurring in the interval delimited by $\left((k-1)L+l\right)\frac{T_{ck}}{L}$ and $\left(kL+l\right)\frac{T_{ck}}{L}$. The applicability of (6) relies on the capability of recognizing the number of exceptions, m, in the sub-sequence made up of the L values following $a\left((k-1)L+l\right)$.

The approach implemented by the algorithm considers that the average number of exceptions in each bunch is given by $L\left(1-C\right)$, that in general has both integer and fractional parts: for instance, for an architecture with $L=4$ channels using a resampling factor $C=0.70$, one has 1.2 exceptions per bunch on average. Exceptions detected at run time in an L-subsequence will be on some occurrences equal to:

$$M = \lceil L\left(1-C\right)\rceil \tag{7}$$

and on the remaining ones equal to $M-1$, such that the variable m can take only two values for any selection of the resampling rate, namely $\{M-1, M\}$. Intuitively, if the coefficient $a((k-1)L+l)$ evaluated by the l-th channel, approaches zero, it is understandable that the next exception is close enough so that there is enough space for M exceptions; on the other hand, if $a((k-1)L+l)$ is close to one, the next exception is far enough so that there is only room for M - 1 exceptions. To recognize the number of exceptions in a deterministic manner and calculate the coefficient $a(kL+l)$, a threshold is introduced:

$$TH = \left[L-1-\left(M-1\right)\right]d - \left(M-1\right) \tag{8}$$

which quantifies the total decrement that is applied to the coefficient $a((k-1)L+l)$ through $L-1$ subsequent steps, in presence of $M-1$ exceptions. If the coefficient $a((k-1)L+l)$ is below the threshold there are M exceptions through the following L steps, otherwise the L-th step is not an exception, and only $M-1$ exceptions are present through the following L steps. More in detail, the algorithm calculates beforehand, according to the settings made by the user, two quantities:

$$inc\_m = -\left(L-M+1\right)d + M - 1 \tag{9}$$

$$inc\_M = -\left(L-M\right)d + M \tag{10}$$

and, as required by (6), if $a((k-1)L+l) \geq TH$ adds to $a((k-1)L+l)$ the quantity inc_m, otherwise inc_M is added. A numerical example shown in the Appendix I is provided to clarify the coefficients update process of the parallel interpolation algorithm.

With the correct values of $a\left(kL+l\right)$, each channel calculates the interpolated sample, $y\left(kL+l\right)$, using (2). The couple of consecutive samples, $x\left(kL+l\right)$ and $x\left(kL+l-1\right)$ needed in input to the interpolation is normally fetched from the bunch of x samples, $\{x\left(kL+1\right), \ldots, x\left(kL+L\right)\}$, that arrive from the ADCs. The only exception is for channel one that requires $x\left(kL+1\right)$ and the L-th sample of the previous bunch $x\left(\left(k-1\right)L+L\right)$. The algorithm thus requires storing the L-th sample of each input bunch to process the following bunch.

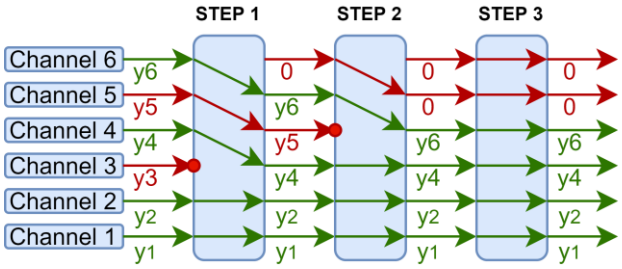

Fig. 4. Defragmentation algorithm. The interpolation channels produce scrambled valid (green) and invalid (red) samples. Defragmentation separates them so that invalid samples are at the top and valid at the bottom.

The algorithm produces at each clock tick a structured array, in which each sample of the output bunch, $\{y\left(kL+1\right), \ldots, y\left(kL+L\right)\}$, is labeled with a Boolean flag, set true if the sample is invalid and as such to be discarded at the memorization stage. The structured array is hence processed by the defragmentation and packing algorithms.

2) Defragmentation: The structured array is processed by the defragmentation algorithm that sweeps the invalid samples at the top of the array, leaving at the bottom the valid ones.

The defragmentation algorithm consists of so many identical steps as the maximum number of exceptions that could appear in L channels for an arbitrary selection of the resampling factor C in [1/2, 1), which is equal to $\lceil L/2 \rceil$ (rounding to the upper integer). At each step, the lowest location hosting an invalid sample is detected, and all the samples that are above the aforementioned location are shifted down, while the topmost location, which is freed up by the downward shift, is marked as invalid, thus keeping constant the number of invalid samples in the array; the example given in Fig. 4 clarifies the described approach. Let us assume an architecture with $L=6$ channels. The maximum number of exceptions that could be present is $\lceil L/2 \rceil = 3$, consequently, a structure with 3 stages is required for defragmentation. Fig. 4 considers an array that contains four valid samples, shown in green, and two invalid samples, shown in red. The array elements are enumerated from bottom to top correspondingly to the interpolation channels. During the first step the invalid sample that derives from channel 3 is identified and the data above channel 3 (channels 4 to 6) are shifted down while the location corresponding to channel 6 is marked as invalid (and filled with zero, as these data are not used). At the second stage the process similarly addresses the sample in the fourth location of the buffer. The last stage leaves the samples unchanged; it is necessary in case the resampling factor selected by the user produces 3 exceptions in a bunch.

The following packing stage requires the knowledge of the number of valid samples in each bunch. Counting the valid

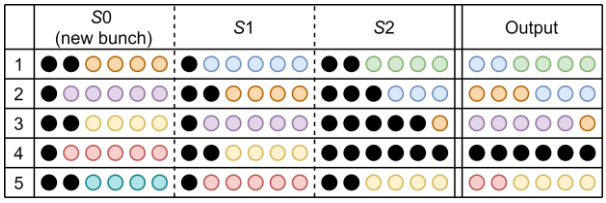

Fig. 5. Example of the Packing algorithm. S2, S1 and S0 are combined to form an output of L valid (colored), or L invalid (black) consecutive samples. The first column indicates the clock ticks.

samples in the bunches is straightforward from the algorithmic point of view but requires an additional circuital stage that in Section III is named 'Glue'.

3) Packing: Packing is obtained implementing a queue that manages three consecutive defragmented bunches and arranges, at each clock tick, one output, containing a bunch of L valid or L invalid samples and a related Boolean flag; the released bunch is substituted by a new bunch from the defragmentation unit according to the logic of the queue, while the two partially elaborated bunches move forward through the queue.

The operation principle of the packing algorithm can be illustrated as follows. The bunches in the queue are named $S0$, $S1$ and $S2$, where $S0$ is the bunch that just entered, and $S2$ is the bunch to be released. $S2$ might contain valid samples or not. In the first case, the output bunch is formed taking the valid samples available in $S2$ and as many valid samples as needed to complete a valid bunch made up of L samples from the bunch $S1$, and if $S1$ can provide only a part of them, the rest is taken from $S0$. In the second case, the same $S2$ bunch is returned as output marked as invalid.

For the sake of clarity more details related to the operation of the algorithm are given in Fig. 5, where an architecture with $L = 6$ channels and a resampling factor $C = 0.75$ are considered. Each defragmented bunch carries some valid samples, highlighted with colored circles, and some invalid samples, highlighted with black circles. The graphic shows the bunches in the queue for 5 consecutive clock ticks, enumerated from 1 to 5 in the leftmost column. Row 1 in Fig. 5 shows bunch $S0$ with 4 valid samples, $S1$ with 5 valid samples, and $S2$ with 4 valid samples, and makes clear that at the first clock tick the output is composed using the 4 valid samples of the oldest bunch in the queue, $S2$, and 2 out of the 5 valid samples in $S1$. At the next clock tick, illustrated by row 2, the bunches $S0$, $S1$, and $S2$, are updated such that they contain 5, 4, and 3 valid samples, respectively. Notice that apart from the incoming bunch, $S0$, bunch $S1$ contains the valid samples that were in $S0$ and have not contributed to the valid output at the previous step, and bunch $S2$ contains the valid samples that were in $S1$ and have not been used to generate the valid output, because in excess of need. Hence, at the second step, the output is composed by 3 samples from $S2$ and 3 samples from $S1$. At the third step, illustrated by row 3, the bunches $S0$, $S1$, and $S2$, contain 4, 5, and 1 valid sample, respectively. A valid output is assembled taking 1 sample from $S2$ and 5 samples from $S1$. At this step all the samples of $S1$ are required, so that at the next step bunch $S2$, will have no valid

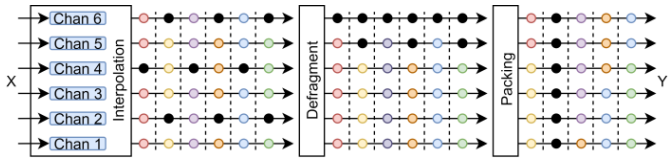

Fig. 6. Resampling algorithm. Example of defrag and packing steps. Colored circles correspond to valid samples. Black circles to invalid samples. Six consecutive outputs can be observed for each step of the algorithm.

samples. At the fourth step, illustrated with row 4, $S0$, $S1$, and $S2$, contain 5, 4, and 0 valid samples, such that, bunch $S2$ with no valid samples is returned marked as an invalid bunch of samples.

The selection of a resampling factor C in the interval [1/2, 1) assures that at least one every other bunch is valid, and that a queue managing three bunches in the packing stage is sufficient to assemble at each clock tick an output bunch with L valid or invalid samples, whichever the number of channels in the architecture. This can be proved observing that each defragmented bunch arriving to the packing stage as bunch $S0$, carries at least $\lfloor L/2 \rfloor$ valid samples. The most critical scenario happens when there is the minimum number of valid samples in the queue, which consists in having bunch $S2$ with no valid samples and only one residual valid sample in $S1$, that assures that $S0$ still contains at least $\lfloor L/2 \rfloor$ valid samples. The algorithm then returns a bunch marked as invalid and loads a new bunch $S0$ with at least $\lfloor L/2 \rfloor$ valid samples. It can be shown that the number of valid samples in the queue is thus $L+1$, which is sufficient to output a valid bunch and fall again in the most critical scenario. The result is obvious in case L is even, since bunch $S2$ has one residual valid sample, and bunches $S1$ and $S0$ each have $L/2$ valid samples; it can also be shown in case L is odd, by observing that the minimum number of valid samples in the two consecutive bunches, namely $S1$ and $S0$, must be expressed as $\lfloor 2L/2 \rfloor = L$.

Fig. 6 presents a summarizing schematic including the described algorithms. A six-channels architecture and a resampling factor equal to 0.75 are chosen. Notice how the bunches are firstly defragmented and then packed to produce L valid bunches 75% of the time, and L invalid bunches data 25% of the time; no sample is lost during the process.

III. Proposed Circuit

A digital circuit that implements the proposed algorithm has been designed. The HDL code and the simulation environment for the proposed circuits is available on CodeOcean to boost scientific reproducibility, [50]. The architecture of the circuit is in Fig. 7. It is composed of: Interpolation (composed by L time varying FIR filters); Defragmentation (changes the order of the samples to separate valid and invalid samples); Glue (calculates the number of valid samples in a defragmented block of samples); Pack (collects the samples to obtain bunches of L valid or invalid samples).

A. Interpolation Block

The first part of the circuit, the interpolation block, is shown in Fig. 8. It is composed by the register that stores the x_L sample and L vcFIR circuits.

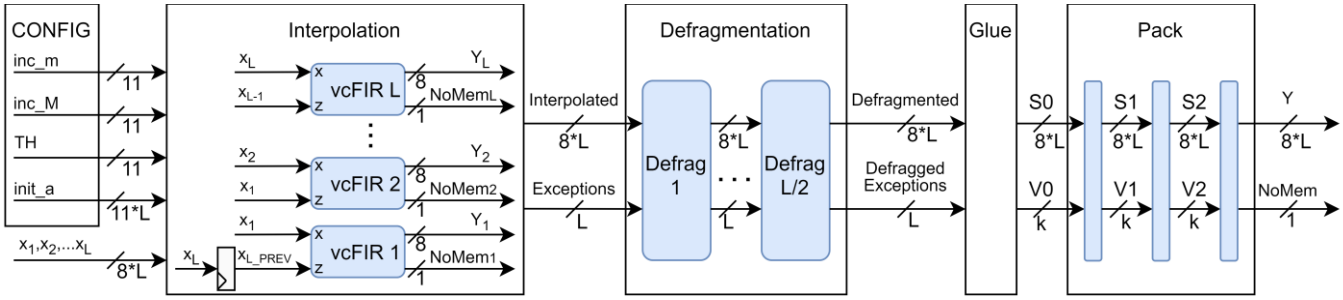

Fig. 7. Architecture of the proposed circuit. The signals in the CONFIG box are calculated (as in Appendix I) and driven to the vcFIR during the configuration stage. The bunches arriving from the ADCs are composed by L samples, $\{x_1, \ldots, x_L\}$ that are forwarded to the vcFIR circuit in couples. The register in the interpolation block stores the x_L sample needed by vcFIR1 to process the subsequent bunch. In the interpolation block, the input signal is processed by L parallel vcFIR filters; In the defragmentation block valid and invalid samples are divided; In the glue block, the defragmented exceptions are converted to the actual number of valid samples; In the pack block, valid samples are combined to form the output.

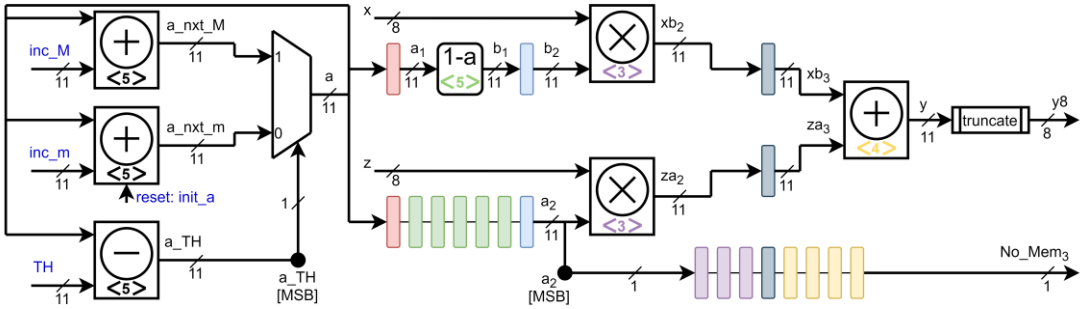

Fig. 8. Variable coefficient FIR (vcFIR) for one channel in Fig. 7. The circuit is replicated L times in the interpolation block of Fig. 7; Colored rectangles correspond to pipeline registers (same color for the same level); Rectangles containing arithmetic operators and a number between angled brackets indicate that the operator contains as many pipeline levels as dictated by the number.

The structure of the vcFIR is shown in Fig. 8. It is similar to the single channel architecture of Fig. 2 presented in [46], [47] but features many differences. An important difference is the way a is updated. In the single channel architecture d is subtracted from a, in every clock cycle, until a gets negative. At that moment, a mux exploits the MSB of signal a to determine the sign of a, and instead of yet another subtraction, a unitary addition restores a to positive values.

The multi-channel version follows the same principle, but since now a needs to be calculated L steps ahead, instead of choosing between d and 1, the multiplexer chooses between summing inc_m or inc_M, depending on the most significant bit of a_TH (that is signal a minus the TH value), which corresponds to its sign (see Section II.B.1). The circuit has also been changed duplicating the adders to calculate in advance the two possible results (loop split technique). In this way the critical path includes only one adder instead of passing through the subtractor and the adder.

Another key difference with the circuit in Fig. 2, is the use of pipeline registers, that allow higher speed notwithstanding the more complex structure. In the multi-channel version, as it can be seen in Fig. 8, a total of twenty pipeline levels has been introduced. Pipeline levels are also inside the adders (4 or 5 levels) and the multipliers (3 levels). The pipeline registers are depicted with colored rectangles in Fig. 8. Signals are named according to the pipeline stage they are in. For instance, signals between the red and blue lines of registers, carry the subscript '1'. Signals between the blue and dark blue registers carry the subscript '2', and after that, '3'. After a, the signal b is calculated by subtracting a from 1:

$$b_1 = 1 - a_1 \tag{11}$$

And the circuit performs the multiplications:

$$za_2 = a_2 \cdot z \tag{12}$$

$$xb_2 = b_2 \cdot x \tag{13}$$

Finally, the two products are summed and truncated to form an 8 bits output sample:

$$y8 = za_3 + xb_3 \quad \text{(truncated to 8 bits)} \tag{14}$$

At the same time, the most significant bit of a, is carried to the output (No_Mem_3), to dictate invalid samples.

The pipelined adders and the subtractor at the input of the circuit in Fig. 8, allow higher working frequencies. However, they create a subtle problem. If they are reset to zero, they provide the wrong a_TH, a_nxt_m, and a_nxt_M signals to the MUX for as many clock cycles as the number of pipeline levels. The MUX will then generate the wrong sequence for signal a, thus jeopardizing the functionality of the entire system. The solution of this problem is to reset the pipeline registers of the adder that generates a_nxt_m to the actual sequence of values that the normal operation would generate (since a_TH is zero, the MUX continuously selects a_nxt_m whilst the pipeline is being filled, so there is no need to initialize a_nxt_M). Thus, additional values for $init\_a$ are

calculated in advance following the algorithm detailed in Section II and in the Appendix I ($L{\cdot}5$ values, where 5 is the number of pipeline levels) and forwarded to the pipeline registers. Since the loop of the a signal, due to the presence of the pipeline, is now looking further ahead in the sequence of the values of a, equations (7), (8), (9), and (10) that calculate the signals M, TH, inc_M, and inc_m, are modified accordingly. The modification is straightforward and consists in substituting L with $L{\cdot}5$ (5 is the number of pipeline levels) in the cited equations.

B. Defragmentation Block

The L samples, named block of samples, and the L exception flag bits that stem out of the interpolation block, feed the 'Defragmentation' block that is composed by $\lceil L/2 \rceil$ equal 'Defrag' stages. Each 'Defrag' stage must locate the first exception and shift by one position all the samples above that position (see Section II.B.2).

A 'Defrag' stage is composed by: a pipeline register for the input block of samples; a pipeline register for the exceptions; several 2 to 1 MUX that allow the conditional shift by one of both the exceptions and the block of samples; a thermometric code combinatorial circuit that receives the exception as input and produces a thermometric code L-bit signal named *thermo* composed of zeros up to the position of the first exception and ones from this position upwards. The *thermo* signal feeds the MUX to generate a partially defragged signal that goes to the next stage for further defragmentation. In terms of speed and silicon area, these components, at least for the implementation with less than 32 channels, are not critical for the circuit.

C. Glue Block

After the defragmentation, follows the 'Glue' block, where the defragged exceptions signal is processed to provide the signal *V0* that indicates the number of valid samples in the defragged block of samples. *V0* is a k-bit signal where:

$$k = \lceil log_2(L+1) \rceil \tag{15}$$

The 'Glue' block is composed by a pipeline register on signal *defragged* that produces the signal *S0* and a pipeline register on signal *defragged_exceptions* that is followed by a priority encoder that produces signal *V0*. In terms of speed and silicon footprint, these components, at least for the implementation with less than 32 channels, are not critical for the circuit.

D. Pack Block

The circuital implementation of the Packing algorithm can be seen in Fig. 9. The inputs to this circuit, are the defragged samples and the *V0* signal described in Section III-C. The outputs are the final block of samples Y (composed by L valid or L invalid samples) and a 1-bit signal, *NoMem*, that turns high to indicate an invalid block of samples.

The 'Pack' circuit is divided in three stages as shown by the dashed lines in 9. As discussed in Section II.B.3), the objective

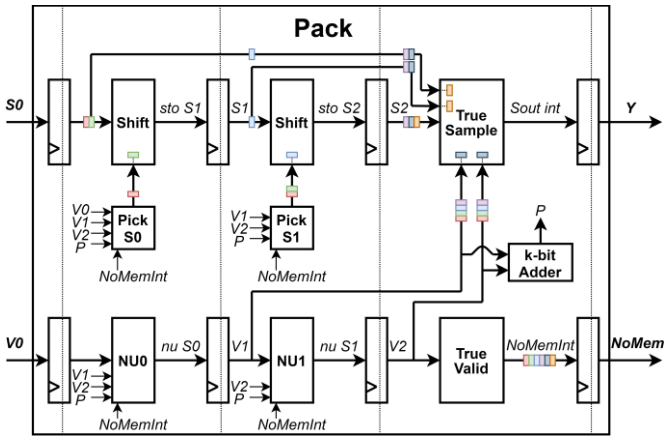

Fig. 9. Pack circuit. Colored rectangles indicate pipeline levels. The top portion (from $S0$ to Y) processes the defragged bunches to generate the output. The bottom portion (from $V0$ to $NoMem$) calculates the number of valid samples in the processed bunches and determines the invalid outputs.

is to combine the valid samples of the three blocks of samples (*S0*, *S1* and *S2*), to create the output block of samples.

The top portion of the 'Pack' circuit processes the block of samples *S0*, *S1*, and *S2*. It generates *S1* using *S0*, generates *S2* using *S1*, and produces the output.

The bottom portion processes the *V0* signal (indicates how many valid samples are in *S0*) and its derived versions (*V1* and *V2* that tell how many valid samples are in *S1* and *S2*, respectively) to determine how many valid samples to pick from *S0*, *S1*, and *S2* to produce the output. It also checks if *S2* contains any valid samples to decide whether the output block of samples is valid or not.

The three stages are separated by registers that are necessary to avoid combinatorial loops. The 'Pack' circuit also contains six additional pipeline levels (shown in Fig. 9 with colored rectangles) to increase the working frequency.

The 'True_Valid' circuit is a NOR gate on *V2* that, if $V2 = 0$, asserts the *NoMemInt* signal and defines the output as invalid.

The 'Pick S1' and 'Pick S0' blocks generate the k-bit signals *PickS1* and *PickS0* that indicate how many samples to extract from *S0* and *S1* to compose the output. The two blocks exploit the flag signal P that is asserted if $(L - V2) < V1$ and is implemented with a k-bit adder and a comparator.

Let us firstly detail the circuital composition of 'Pick S1'. It is composed by a mux, and one k-bit adder. The mux is driven by two control lines *NoMemInt* and the signal, P.

- If $\{NoMemInt, P\} = \{1, -\}$, $PickS1 = 0$.
- If $\{NoMemInt, P\} = \{0,0\}$, $PickS1 = V1$ since all the valid *S1* samples contribute to the output.
- If $\{NoMemInt, P\} = \{0,1\}$, $PickS1 = (n - V2)$ (thus the need of the k-bit adder). In this case only some of the *S1* samples contribute to the output.

The block 'Pick S0' is very similar to 'Pick S1'. Its mux is driven by the same signals as 'Pick S1'.

- If $\{NoMemInt, P\} = \{1, -\}$ or $\{0,1\}$, $PickS0 = 0$.

- If $\{NoMemInt,\ P\} = \{0,0\}$, $PickS0 = n\text{-}V2\text{-}V1$. The circuit will also use some of the valid $S0$ samples to generate the output.

The blocks 'Shift0' and 'Shift1' are combinational shifters (shift from 0 to n bit) that right-shift $S0$ and $S1$ signals as many times as $PickS0$, $PickS1$. This eliminates the samples that contribute to the output and in the next clock cycle, $S1$ and $S2$ will only contain the unused valid samples.

The blocks 'NU0', 'NU1' determine the valid samples in $S1$, $S2$, defining the k-bit signals $nxV1$, $nxV2$. These blocks are very similar to 'PickS0' from the circuital point of view. They are composed by a mux driven by $\{NoMemInt,P\}$, and a k-bit adder.

The mux in the 'NU0' block operates as follows:

- If $\{NoMemInt,\ P\} = \{1,\ -\}$ or $\{0,1\}$ $nxV1 = V0$. No samples picked from $S0$.
- If $\{NoMemInt,\ P\} = \{0,0\}$ $nxV1 = V0{+}V1\text{-}V2\text{-}L$.

The mux in the 'NU1' block operates as follows:

- If $\{NoMemInt,\ P\} = \{1,\ -\}$ $nxV2 = V1$. No samples have been picked from $S1$.
- If $\{NoMemInt,\ P\} = \{0,0\}$ $nxV2 = 0$. All the valid samples in $S1$ have been picked.
- If $\{NoMemInt,\ P\} = \{0,1\}$ $nxV2 = V1\text{-}(L\text{-}V2)$.

The last remaining block is the 'True Sample' that is responsible for producing the final output signal. The block is composed by two combinational shifters, one k-bit adder, and one OR gate. The shifters operate on S1 (shifted by $V2$ positions) and on S0 (shifted by $V2{+}V1$ positions, thus the need for the adder). The OR gate combines the shifted $S0$, shifted $S1$ and $S2$ to create the output sample. It can be demonstrated that the shifters, no matter the value of P, succeed in producing a valid output if $NoMemInt = 0$.

In conclusion, the circuital implementation of the 'Pack' block, notwithstanding the complexity of the algorithm and the large final silicon footprint of the circuit, is broken down to six k-bit adders (depending on the optimization this number can change), muxes, and some wide input combinational shifters. Since k is slowly increasing with the number of channels, L, it is expected that the performance limit is due to the shifters whose complexity linearly increases with L).

IV. Implementation Results

The circuit has been synthesized targeting a commercial standard-cell library in 14 nm FinFET technology provided by Global Foundry. Physical synthesis is performed by using Cadence Genus (that also includes parasitic estimates from the subsequent Place&Route phase). The circuit is synthesized according to the imposed timing constraints. The considered technology corner is the typical one with 0.8 V supply voltage and regular V_{TH}. The simulations, with delay and switching activity annotation, have been conducted with Cadence NCSIM. Power dissipation is computed by simulating the final netlist with 10^4 input vectors from an asynchronously sampled sinusoid to obtain the switching activity.

TABLE I

Throughput. ASIC Implementation Results in 14nm FinFET GF Technology. Different Number of Channels Are Considered

Number of Channels	Frequency [GHz]	Throughput [GSps]
Proposed Circuit		
8	5.26	42.1
16	4.55	72.8
32	4.20	134.4
48	3.82	183.4
64	3.42	218.9
Previous Art [47]		
1	5.26	5.26

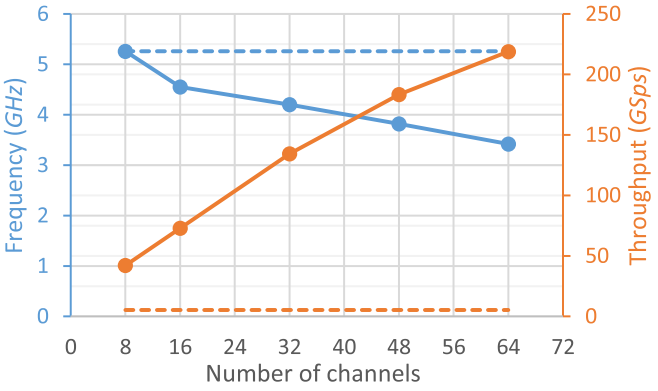

Fig. 10. Frequency and throughput vs number of channels. Dashed lines mark the results for the single channel (5.26 GSps) architecture proposed in [47].

A. Clock Frequency and Throughput

Architectures featuring different numbers of channels in the interpolating block, have been simulated and synthesized. The maximum clock frequency and the sample throughput is reported in Table I. Five versions of the proposed circuit are reported whose number of channels ranges from 8 to 64.

With reference to the clock frequency, it is worth noting that the proposed circuit, notwithstanding the largely increased complexity when compared with [47], reaches the same working frequency of 5.26 GHz when implemented with 8 channels. As the number of channels is increased, certain computations impose a greater workload to specific parts of the circuit, resulting in reduced circuit speed. As an example, the maximum working frequency for the 64 channels implementation of the circuit is reduced to 3.42 GHz.

However, as it can be observed in Fig. 10, even though the highest working frequency is decreased, the throughput is increased almost linearly when the number of channels is increased (around 3.3 GSps increase for each additional channel starting from the result obtained with 8 channels). That is, the more channels are implemented, the more samples are processed per unit time. As it is easily gathered, it takes around 56 channels to reach the 200 GSps milestone, which is far ahead of the state of the art. To match the state-of-the-art results of 100 GSps around 24 channels are needed. When compared with the implementation proposed in [47], whose

TABLE II

SILICON AREA OCCUPATION. ASIC IMPLEMENTATION RESULTS IN 14NM FINFET GF TECHNOLOGY. DIFFERENT NUMBER OF CHANNELS ARE CONSIDERED

Number of Channels	Total Area [μm^2]	Area per channel [μm^2]	Area breakdown Percentage [%]		
Proposed Circuit			Interpolation	Defrag	Glue + Pack
8	12713	1589	80.7	3.9	15.0
16	27371	1711	76.1	8.1	15.4
32	67354	2105	68.0	15.1	16.8
48	117148	2441	62.5	21.2	16.2
64	170385	2662	58.0	26.8	14.8
Previous Art [47]					
1	683	683	-	-	-

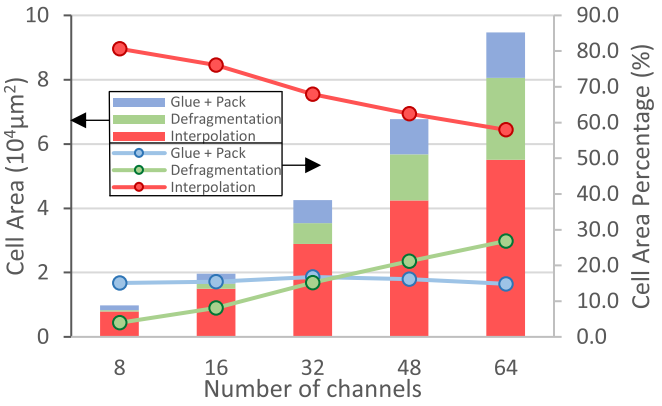

Fig. 11. Breakdown of the silicon area occupation vs number of channels. Absolute and percentage values for the Interpolation, Defragmentation, and Glue summed with Pack blocks are shown.

TABLE III

COMBINATIONAL AND SEQUENTIAL LOGIC BREAKDOWN. ASIC IMPLEMENTATION RESULTS IN 14NM FINFET GF TECHNOLOGY

Number of Channels	# combinational cells / area percentage	# Sequential Elements (FF)
Proposed Circuit		
8	9116 / 29.3%	6048
16	16490 / 28.4%	12018
32	39709 / 32.2%	25096
48	62975 / 30.9%	39816
64	85061 / 29.4%	56103
Previous Art [47]		
1	785 / 43.0%	307

throughput was limited to 5.26 GSps, the proposed circuit shows largely improved performance and design flexibility.

B. Area and Power Results

Silicon area occupation for the five implementations of the circuit considered in the paper is reported in Table II. The total area increases slightly faster than linearly with the number of channels. This is because the proposed circuit is composed by L vcFIR circuits whose structure is not dependent on the number of channels, and the Defragmentation, Glue, and Pack blocks that indeed depend on, and slowly increase in terms of complexity, with L. A breakdown of the area occupation together with a graph showing the area percentage for the various blocks is reported in Fig. 11. The Interpolation block is about 80% of the total area for the 8 channels version of the circuit while only occupies 58% of the circuit when moving to the 64 channels version due to the increased complexity of the remaining blocks. Among these, the area required by the Glue block is negligible. With reference to the Defragmentation, and Pack blocks, the largest area increase is due to the Defragmentation block whose area increases with the square of L (note that the Defragmentation block is composed by a number of Defrag stages that linearly depends on L while the dimension of each defrag stage also increases linearly with L).

The comparison with the area of the circuit proposed in [47] is not simple since the actual proposal includes many more blocks and channels. Table II shows the area per channel of the proposed circuit (ranges from 2.5 to 4 times larger than [47]). An interesting comparison is between the circuit in [47] and the vcFIR circuit of Fig. 8 (since both carry out similar processing) that shows that the increased pipelining and the changes in the structure result in about 30% area increase with no penalty on the working frequency.

The breakdown of combinational and sequential logic composition for the proposed circuit is in Table III. The use of pipelining and the design for high working frequency result in a massive use of flip-flops with a percentage of area occupation for sequential elements that is around 70% and is not dependent on the number of channels. When comparing with [47], that uses 57% of the area for flip-flops, the proposed circuit shows a higher share due to pipelining.

Power dissipation results follow the same trends as the area occupation, as seen in Table IV. The largest circuit with 64 channels dissipates less than 1 mW. The most power-hungry component is always the Interpolation block, even if its weight decreases with the number of channels. With reference to the rest of the circuit, when it comes to power, the defragmentation is again the most critical block since its power requirement increases with the square of the number of channels.

C. Circuit Scalability With Number of Input Bits

Conventional high-performance oscilloscopes are based on 8 bits ADC and aim at high sample rate and bandwidth since

TABLE IV

POWER DISSIPATION. ASIC IMPLEMENTATION RESULTS IN 14NM FINFET GF TECHNOLOGY. DIFFERENT NUMBER OF CHANNELS ARE CONSIDERED

# Channels	Leakage [μW]	Dynamic [μW/MHz]	Power at max freq. [mW]	Power breakdown [mW]		
Proposed Circuit				Interpolation	Defrag	Glue+Pack
8	10.85	26.25	138.15	112.55	6.11	25.14
16	21.17	52.31	237.79	183.65	20.88	44.58
32	45.95	113.29	476.05	328.45	78.40	93.13
48	76.75	183.23	699.44	436.53	158.72	127.45
64	106.39	248.99	852.79	499.44	236.13	134.87
Previous Art [47]						
1	0.665	1.40	7	-	-	-

TABLE V

ASIC IMPLEMENTATION RESULTS IN 14NM FINFET GF TECHNOLOGY. DIFFERENT NUMBER OF CHANNELS AND BITS PER SAMPLE ARE CONSIDERED. PERCENTAGES INDICATE THE WORSENING OF RESULTS WITH RESPECT TO 8-BIT SAMPLES

# Channels / #bits per sample	Total Area [μm^2]	# Sequential Elements (FF)	Frequency [GHz] / Throughput [GSps]	Power at max freq. [mW]
4 / 12	8171	3806	5.49 / 22.0	96.77
8 / 8	12713	6048	5.26 / 42.1	138.15
8 / 12	16869 (+32.7%)	7456 (+23.3%)	4.85 / 38.8 (-7.8%)	165.99 (+20.2%)
16 / 8	27371	12018	4.55 / 72.7	237.79
16 / 12	35609 (+30.1%)	14579 (+21.3%)	3.92 / 62.7 (-13.7%)	262.11 (+10.2%)
64 / 8	170385	56103	3.42 / 219.2	852.79
64 / 12	210517 (+23.6%)	72304 (+28.9%)	2.78 / 177.8 (-18.9%)	956.21 (+12.1%)

these parameters are of utmost importance in most applications. For those applications that benefit from measurements with higher dynamic range (e.g., ultrasound range finder), the market provides oscilloscopes that use 12 bit ADC, [51], that provide sample rate up to 20GSps. For a larger number of input bits oscilloscopes are not available and the user needs to move towards digitizers with a sample rate up to 250 MSps, [52], [53].

The circuit presented in the paper is easily scalable with the number of input bits being mostly composed of blocks that grow linearly or logarithmically with the number of input bits. The only exceptions are the binary multipliers in the vcFIR blocks (exploited for the linear interpolation), but these can be easily pipelined to counteract the impact of the number of input bits on the speed. Please note that the interested reader can use the parametrizable HDL code available at [50] to adapt the circuit to a particular application.

Synthesis results using 12 bits input samples are shown in Table V. The table shows the results for 4, 8, 16, and 64 channels at 12 bit compared, when possible, against the 8 bit version of the circuit. The 12bit circuits, yet at 4 channels, provide a throughput higher than what is needed by the high-end oscilloscope at 12 bits available today. In general, moving from 8 to 12 bits (50% increase of the number of bits) results in about 30% area increase, a reduction of the maximum clock frequency that is below 20% (even if, as said before, speed can be tailored to the application by careful pipelining and retiming) and a power increase that ranges from 10% to 20%.

The synthesis results, that can also be further tailored to a given application requiring certain speed or power, allow to state that the circuit is tolerant to the increase of the number of input bits.

V. CONCLUSION

A digital circuit able to downsample the output stream of digitizers made up of several time interleaved ADCs, like those adopted in modern high-performance DSOs, at an almost arbitrary sample rate, has been proposed and demonstrated in 14nm FinFET technology. The circuit implements an additional option for the DSOs, that allows optimal usage of memory resources, and is useful for specific applications requiring fine selection of the sampling rate. The proposed circuit has no impact on the capture rate and jitter performance of DSO.

APPENDIX A
INITIALIZATION SIGNALS

An example clarifies the configuration stage and the update of the time varying coefficient represented by signal a. Let us assume a four-channel architecture, i.e. $L = 4$ and a chosen resampling factor $C = 0.8$. During the configuration stage II-B is used to calculate $d = C^{-1} - 1 = 0.25$.

Recalling that the current value of signal a is obtained by decrementing by d the previous one, unless this one is negative, in which case it is instead obtained incrementing by 1 the negative value, one can state that the first set of values, namely $a(kL + l)$ related to $k = 0$ with l ranging from 1 up to L, are:

$$a(1) = 0$$
$$a(2) = a(1) - d = -0.25$$
$$a(3) = a(2) + 1 = 0.75$$
$$a(4) = a(3) - d = 0.50$$

In order to update the values $a(kL + l)$ related to $k > 0$, the circuit needs the parameter $M = \lceil L\,(1 - C) \rceil = 1$, the

threshold $TH = [\mathrm{L} - 1 - (\mathrm{M} - 1)]\,\mathrm{d} - (\mathrm{M} - 1) = 0.75$, the increments $inc\_m = -(L - M + 1)\,d + M - 1 = -1$ and $inc\_M = -(L - M)\,d + M = 0.25$ calculated using (7), (8), (9), and (10). These increments are managed according to a conditional instruction, identical for all channels, namely:

if $a(kL + l) - TH \geq 0$
then $a(kL + l + L) = a(kL + l) + inc\_m$
else $\{a(kL + l) - TH < 0\}$
$a(kL + l) = a(kL + l) + inc\_M$

For instance, at time instant $k = 1$, for channels $l = 1,\ldots, L$, one obtains:

$$a(5) = 0.25, \text{ obtained using } a(5) = a(1) + inc\_M$$
$$a(6) = 0, \text{ obtained using } a(6) = a(2) + inc\_M$$
$$a(7) = -0.25, \text{ obtained using } a(7) = a(3) + inc\_m$$
$$a(8) = 0.75, \text{ obtained using } a(8) = a(4) + inc\_M.$$

After $a(kL + l)$, $l = 1, \ldots, L$ are calculated, the corresponding outputs, $y(kL + l)$, are determined interpolating the input samples $x(kL + l - 1)$ and $x(kL + l)$. Each output is labelled by means of a flag that turns high every time the corresponding $a(kL + l)$ is negative. The structured array, formed by $y(kL + l)$, $l = 1, \ldots, L$ and the related flags, is driven to the defragmentation unit and to the pack unit as explained in Section II.

Appendix B
Resolution of the Resampling Factor

The resolution of the resampling factor, C, depends on the numerical representation of the configuration signals within the circuit. All the configuration signals are calculated exploiting an internal parameter defined as:

$$E = -d = 1 - C^{-1} \tag{A1}$$

The circuit proposed in the paper represents the parameter E, which is in the range $[-1,0)$, using an 8 bits signal that represents the fractional part of the number (and implies the leading ones). The representation of E with 8 bits means that there are 256 possible E values thus 256 possible C values in the [0.5,1) range. Such values are not evenly spaced. The best resolution is obtained for C close to 0.5 and is 0.97×10^{-3} while the worst resolution is obtained for C close to 1 and is 3.9×10^{-3}.

A detailed derivation of the resolution of the resampling factor C in the general case is given in the following. Assuming an n-bit E signal, its resolution is constant and equal to 2^{-n}, while its range is $\left[-1, -2^{-n}\right]$. The actual resampling factor, C, is:

$$C = (1 - E)^{-1} \tag{A2}$$

and: $C \in \left[1/2, 1/\left(1 + 2^{-n}\right)\right] \sim \left[1/2, 1 - 2^{-n}\right]$. Thus, the proposed circuit allows the user to choose between 256 different values for the resampling factor. Due to the non-linear dependence between E and C the resolution of C, that is the step between two consecutive possible values, is not constant. It can be calculated as a function of the resolution of E:

$$|d(C)| = \left|\frac{d(C)}{d(E)} d(E)\right| = \left|\frac{d(1 - E)^{-1}}{d(E)}\right| d(E) = \frac{2^{-n}}{(1 - E)^2} \tag{A3}$$

- For E values close to 0, (resampling rates, C close to 1), the resolution of C, is larger:

$$LargestC_{STEP} = d(C)\,|_{E \to 0} = 2^{-n} \tag{A4}$$

- For E values close to -1, (resampling rates, C close to 0.5), the resolution of C, is relatively small:

$$SmallestC_{STEP} = d(C)\,|_{E=-1} = \frac{1}{4} 2^{-n}. \tag{A5}$$

A larger number of bits for the representation of E, results in a very fine resolution of the resampling factor, but it is in trade off with the complexity of the system.

References

[1] M. Krauss, H. Thieme, H.-G. Schniek, and E. Wittig, "Fully-integrated 5 V CMOS system for a 20 M sample/s sampling oscilloscope," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 1996, pp. 384–385, doi: 10.1109/ISSCC.1996.488727.

[2] S. Naboicheck and S. Ems, "5 GHz sampling oscilloscope front-end based on heterojunction bipolar transistors (HBT)," in *Proc. 15th Annu. GaAs IC Symp.*, Oct. 1993, pp. 155–158, doi: 10.1109/GAAS.1993.394480.

[3] J. Chou, J. Conway, G. Sefler, G. Valley, and B. Jalali, "150 GS/s real-time oscilloscope using a photonic front end," in *Proc. Int. Topical Meeting Microw. Photon. Jointly Held Asia–Pacific Microw. Photon. Conf.*, Sep. 2008, pp. 35–38, doi: 10.1109/MWP.2008.4666628.

[4] H. Huawei and L. Xiaohong, "Design and implementation of network communication module for non-OS DSO," in *Proc. 2nd Int. Conf. Comput. Eng. Technol.*, Apr. 2010, pp. V7-150–V7-153, doi: 10.1109/ICCET.2010.5485301.

[5] P. Z. Csurcsia, A. Banovics, and I. Kollar, "Digital oscilloscope displays results together with confidence bounds," in *Proc. IEEE Int. Symp. Intell. Signal Process.*, Aug. 2009, pp. 81–86, doi: 10.1109/WISP.2009.5286576.

[6] X. Zhuang, Y. Zhao, and L. Wang, "The research and application of sine interpolation in digital storage oscilloscope," in *Proc. IEEE Circuits Syst. Int. Conf. Test. Diagnosis*, Apr. 2009, pp. 1–3, doi: 10.1109/CAS-ICTD.2009.4960897.

[7] J. Xiaochang and W. Jie, "An analog front end design for GSPS oscilloscope," in *Proc. 14th IEEE Int. Conf. Electron. Meas. Instrum. (ICEMI)*, Nov. 2019, pp. 286–291, doi: 10.1109/ICEMI46757.2019.9101869.

[8] Y. Kuojun, P. Zhixiang, S. Jiali, and Y. Peng, "A fast baseline and trigger level calibration method in digital oscilloscope," in *Proc. IEEE Int. Instrum. Meas. Technol. Conf. (I2MTC)*, May 2019, pp. 1–6, doi: 10.1109/I2MTC.2019.8827158.

[9] K. Park and J. Park, "20 ps resolution time-to-digital converter for digital storage oscilloscopes," in *Proc. IEEE Nucl. Sci. Symp. Conf. Rec.*, vol. 2, Sep. 1998, pp. 876–881, doi: 10.1109/NSSMIC.1998.774310.

[10] Q. Duyu, T. Shulin, and P. Huiqing, "Design and realization of compensation for one type of broadband acquisition channel," in *Proc. IEEE 10th Int. Conf. Electron. Meas. Instrum.*, Aug. 2011, pp. 212–216, doi: 10.1109/ICEMI.2011.6037890.

[11] Q. Duyu, T. Shulin, Z. Hao, and P. Huiqing, "Study on signal conditioning technology in 2GHz broadband digital oscilloscope," in *Proc. IEEE 11th Int. Conf. Electron. Meas. Instrum.*, Aug. 2013, pp. 342–346, doi: 10.1109/ICEMI.2013.6743069.

[12] D. Qiu, S. Tian, T. Feng, Z. Hao, and H. Wuhuang, "Design and implementation of 500GSPS random equivalent sampling," in *Proc. 12th IEEE Int. Conf. Electron. Meas. Instrum. (ICEMI)*, Jul. 2015, pp. 953–957, doi: 10.1109/ICEMI.2015.7494363.

[13] Y. Wanyu, Z. Yijiu, Y. Zhonghao, and L. Dan, "Design of random equivalent sampling control module based on FPGA," in *Proc. 14th IEEE Int. Conf. Electron. Meas. Instrum. (ICEMI)*, Nov. 2019, pp. 1027–1032, doi: 10.1109/ICEMI46757.2019.9101501.

[14] M. D'Apuzzo, M. D'Arco, and M. Vadursi, "A proposal for DSO bandwidth extension through synchronous time interleaving," in *Proc. IEEE Int. Instrum. Meas. Technol. Conf. Proc.*, May 2016, pp. 1–5, doi: 10.1109/I2MTC.2016.7520494.

[15] M. Vertregt, W. Rey, M. Boonen, J. Verhaegh, and W. Wiertsema, "A 0.4 W mixed-signal digital storage oscilloscope processor with Moire prevention, embedded 393 kb RAM and 50 M sample/s 8b ADC," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 1998, pp. 114–115, doi: 10.1109/ISSCC.1998.672396.

[16] Y. Zhao, X. Zhuang, and L. Wang, "The research and application of random sampling in digital storage oscilloscope," in *Proc. IEEE Circuits Syst. Int. Conf. Test. Diagnosis*, Apr. 2009, pp. 1–3, doi: 10.1109/CAS-ICTD.2009.4960896.

[17] M. D'Arco, M. Genovese, E. Napoli, and M. Vadursi, "Design and implementation of a preprocessing circuit for bandpass signals acquisition," *IEEE Trans. Instrum. Meas.*, vol. 63, no. 2, pp. 287–294, Feb. 2014, doi: 10.1109/TIM.2013.2278564.

[18] M. D'Apuzzo and M. D'Arco, "A wide-band DSO architecture based on three time interleaved channels," *J. Instrum.*, vol. 11, no. 8, Aug. 2016, Art. no. P08003, doi: 10.1088/1748-0221/11/08/P08003.

[19] M. D'Apuzzo and M. D'Arco, "Sampling and time–interleaving strategies to extend high speed digitizers bandwidth," *Measurement*, vol. 111, pp. 389–396, Dec. 2017, doi: 10.1016/j.measurement.2017.08.001.

[20] Z. Zou, J. Wu, G. Yang, Y. Bie, Y. Wang, and M. Zhou, "Design of simple oscilloscope and spectrum analysis system," in *Proc. IEEE 4th Inf. Technol. Mechatronics Eng. Conf. (ITOEC)*, Dec. 2018, pp. 1662–1666, doi: 10.1109/ITOEC.2018.8740670.

[21] H. Zhiqiang, F. Guonan, and Z. Jingzhi, "Design and implement of the digital storage oscilloscope card based on VHDL," in *Proc. IEEE 10th Int. Conf. Electron. Meas. Instrum.*, Aug. 2011, pp. 346–349, doi: 10.1109/ICEMI.2011.6037747.

[22] W. Zibin and C. Changling, "Design of WBDSO control circuit with state machine," in *Proc. Int. Conf. Commun., Circuits Syst.*, May 2005, p. 1180, doi: 10.1109/ICCCAS.2005.1495318.

[23] F. Attivissimo, A. Di Nisio, N. Giaquinto, and M. Savino, "Measuring time base distortion in analog-memory sampling digitizers," *IEEE Trans. Instrum. Meas.*, vol. 57, no. 1, pp. 55–62, Jan. 2008, doi: 10.1109/TIM.2007.909600.

[24] *Tektronix, Digital and Mixed Signal Oscilloscopes—MSO/DPO70000 Series Datasheet*, document Lit. Num. 55W-23446-34, Apr. 2020. [Online]. Available: www.tek.com

[25] (Aug. 7, 2021). *Teledyne-Lecroy Wavemaster 8 Zi-B Oscilloscopes 4-30GHz DataSheet, Lit. Num. wm8zi-b-ds-21may19*. [Online]. Available: http://cdn.teledynelecroy.com/files/pdf/wavemaster-8zi-b-datasheet.pdf

[26] P. J. Pupalaikis *et al.*, "Technologies for very high bandwidth real-time oscilloscopes," in *Proc. IEEE Bipolar/BiCMOS Circuits Technol. Meeting (BCTM)*, Sep. 2014, pp. 128–135, doi: 10.1109/BCTM.2014.6981299.

[27] J. Pickerd, *DSP in High Performance Oscilloscope*, document Lit. Num. 55W-17589-0, Tektronix White Paper, 2004.

[28] P. Monsurrò, A. Trifiletti, L. Angrisani, and M. D'Arco, "Streamline calibration modelling for a comprehensive design of ATI-based digitizers," *Measurement*, vol. 125, pp. 386–393, Sep. 2018, doi: 10.1016/j.measurement.2018.04.099.

[29] J. Song, S. Tian, Y.-H. Hu, and P. Ye, "Digital iterative harmonic rejection and image cancellation for LPF-less frequency-interleaved analog-to-digital converters," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 66, no. 12, pp. 2072–2076, Dec. 2019, doi: 10.1109/TCSII.2019.2895341.

[30] G. Kim *et al.*, "A 161-mW 56-Gb/s ADC-based discrete multitone wireline receiver data-path in 14-nm FinFET," *IEEE J. Solid-State Circuits*, vol. 55, no. 1, pp. 38–48, Jan. 2020, doi: 10.1109/JSSC.2019.2938414.

[31] Hulidan and W. Jinwei, "DSO-based signal reconstruction of time-interleaved," in *Proc. Int. Conf. Commun., Circuits Syst.*, May 2008, pp. 843–846, doi: 10.1109/ICCCAS.2008.4657901.

[32] B. Xu, Y. Zhou, and Y. Chiu, "A 23-mW 24-GS/s 6-bit voltage-time hybrid time-interleaved ADC in 28-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 52, no. 4, pp. 1091–1100, Apr. 2017, doi: 10.1109/JSSC.2016.2642204.

[33] S. Cai, E. Z. Tabasy, A. Shafik, S. Kiran, S. Hoyos, and S. Palermo, "A 25 GS/s 6b TI two-stage multi-bit search ADC with soft-decision selection algorithm in 65 nm CMOS," *IEEE J. Solid-State Circuits*, vol. 52, no. 8, pp. 2168–2179, Aug. 2017.

[34] D.-S. Jo, B.-R.-S. Sung, M.-J. Seo, W.-C. Kim, and S.-T. Ryu, "A 40-nm CMOS 7-b 32-GS/s SAR ADC with background channel mismatch calibration," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 67, no. 4, pp. 610–614, Apr. 2020, doi: 10.1109/TCSII.2019.2916913.

[35] K. Sun, G. Wang, Q. Zhang, S. Elahmadi, and P. Gui, "A 56-GS/s 8-bit time-interleaved ADC with ENOB and BW enhancement techniques in 28-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 54, no. 3, pp. 821–833, Mar. 2019, doi: 10.1109/JSSC.2018.2884352.

[36] J.-W. Nam, M. Hassanpourghadi, A. Zhang, and M. S.-W. Chen, "A 12-bit 1.6, 3.2, and 6.4 GS/s 4-b/Cycle time-interleaved SAR ADC with dual reference shifting and interpolation," *IEEE J. Solid-State Circuits*, vol. 53, no. 6, pp. 1765–1779, Jun. 2018.

[37] M. Pisati *et al.*, "A 243-mW 1.25–56-Gb/s continuous range PAM-4 42.5-dB IL ADC/DAC-based transceiver in 7-nm FinFET," *IEEE J. Solid-State Circuits*, vol. 55, no. 1, pp. 6–18, Jan. 2020, doi: 10.1109/JSSC.2019.2936307.

[38] S. Gori and C. Narduzzi, "Application of a phase measurement algorithm to digitizing oscilloscope characterization," *IEEE Trans. Instrum. Meas.*, vol. 49, no. 6, pp. 1211–1215, Dec. 2000.

[39] S. A. Khan, A. K. Agarwala, D. T. Shahani, and M. M. Alam, "Advance oscilloscope triggering," *IEEE Trans. Instrum. Meas.*, vol. 56, no. 3, pp. 944–953, Jun. 2007.

[40] R. Lapuh, B. Pinter, B. Voljc, Z. Svetik, and M. Lindic, "Digital oscilloscope calibration using asynchronously sampled signal estimation," *IEEE Trans. Instrum. Meas.*, vol. 60, no. 7, pp. 2570–2577, Jul. 2011.

[41] C. M. Wang, P. D. Hale, K. J. Coakley, and T. S. Clement, "Uncertainty of oscilloscope timebase distortion estimate," *IEEE Trans. Instrum. Meas.*, vol. 51, no. 1, pp. 53–58, Feb. 2002.

[42] C. M. Wang, P. D. Hale, and K. J. Coakley, "Least-squares estimation of time-base distortion of sampling oscilloscopes," *IEEE Trans. Instrum. Meas.*, vol. 48, no. 6, pp. 1324–1332, Dec. 1999.

[43] V. G. Ivchenko, A. N. Kalashnikov, R. E. Challis, and B. R. Hayes-Gill, "High-speed digitizing of repetitive waveforms using accurate interleaved sampling," *IEEE Trans. Instrum. Meas.*, vol. 56, no. 4, pp. 1322–1328, Aug. 2007.

[44] Y. J. Zhao, Y. H. Hu, and H. J. Wang, "Enhanced random equivalent sampling based on compressed sensing," *IEEE Trans. Instrum. Meas.*, vol. 61, no. 3, pp. 579–586, Mar. 2012.

[45] M. D'Arco, E. Napoli, and L. Angrisani, "A time base option for arbitrary selection of sample rate in digital storage oscilloscopes," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 6, pp. 3936–3948, Jun. 2020, doi: 10.1109/TIM.2019.2939765.

[46] M. D'Arco, E. Napoli, and E. Zacharelos, "Digital circuit for the arbitrary selection of sample rate in digital storage oscilloscopes," in *Proc. Int. Conf. Appl. Electron. Pervading Ind., Environ. Soc.*, Pisa, Italy, Sep. 2019, pp. 183–189, doi: 10.1007/978-3-030-37277-4_21.

[47] M. D'Arco, E. Napoli, and E. Zacharelos, "Digital circuit for seamless resampling ADC output streams," *Sensors*, vol. 20, no. 6, p. 1619, Mar. 2020, doi: 10.3390/s20061619.

[48] S. Xu, X. Zou, B. Ma, J. Chen, L. Yu, and W. Zou, "Deep-learning-powered photonic analog-to-digital conversion," *Light, Sci. Appl.*, vol. 8, no. 1, pp. 1–11, Jul. 2019.

[49] M. Imani, E. R. Dougherty, and U. Braga-Neto, "Boolean Kalman filter and smoother under model uncertainty," *Automatica*, vol. 111, Jan. 2020, Art. no. 108609.

[50] E. Napoli. *Real-Time Downsampling in Digital Storage Oscilloscopes With Multichannel Architectures*. Accessed: Feb. 2021, doi: 10.24433/CO.3482395.v1. [Online]. Available: https://codeocean.com/

[51] (Dec. 20, 2018). *Teledyne-Lecroy WavePro HD Oscilloscopes 2.5-8GHz DataSheet*. [Online]. Available: http://cdn.teledynelecroy.com/files/pdf/waveprohd-datasheet.pdf

[52] *National Instruments PXIe-5622 150MS/s, 16-Bit PXI IF Digitizer*, document Lit. Num. 375023F-01, Dec. 2017. [Online]. Available: https://www.ni.com/pdf/manuals/375023f.pdf

[53] (Jan. 26, 2021). *Spectrum Instrumentation DN2.822-04 250MS/s Digitizer*. [Online]. Available: https://spectrum-instrumentation.com/sites/default/files/download/dn2_82x_datasheet_english.pdf

Ettore Napoli (Senior Member, IEEE) received the Ph.D. degree (Hons.) in electronic engineering in 1995, the Ph.D. degree in electronic engineering in 1999, and the Ph.D. degree (Hons.) in physics in 2009.

He was a Research Associate with the Engineering Department, University of Cambridge, U.K., in 2004. Since 2020, he has been a Full Professor with the University of Napoli Federico II. He has authored or coauthored more than 100 articles published in international journals and conferences. His research interests include modeling and design of power semiconductor devices and VLSI circuit design.

Efstratios Zacharelos received the B.S. degree in physics and the M.S. degree in electronic physics from Aristotle University, Thessaloniki, Greece, in 2016 and 2019, respectively. He is currently pursuing the Ph.D. degree in electronic engineering with Federico II University, Naples, Italy.

Mauro D'Arco (Senior Member, IEEE) received the Ph.D. degree (Hons.) in electronic engineering in 1999, and the Ph.D. degree in electro-technical engineering in 2003. He was the Coordinator of a task force at DE-STI-ECE, CERN, where he has been Unpaid Associate from 2010 to 2011 and a Visiting Scientist at TE-MSC-MM in 2014. He is currently an Associate Professor with the University of Napoli Federico II. His research interests include the models for digital-to-analog converters, the arbitrary waveform generators, and innovative acquisition modes.

Antonio Giuseppe Maria Strollo (Senior Member, IEEE) received the M.S. (Hons.) and Ph.D. degrees in electronic engineering from the University of Napoli Federico II, Italy. Since 2002, he has been a Full Professor with the University of Napoli Federico II, where he has been the Head of the Department of Electronic and Telecommunication Engineering from 2005 to 2008. He has published more than 140 articles on international journals and conferences. His research interest includes design and analysis of digital VLSI circuits. From 2009 to 2012, he was an Associate Editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS. He is currently an Associate Editor of *Integration, the VLSI Journal*.