

A Practical FPGA-Based Architecture for Arbitrary-Ratio Sample Rate Conversion

Brunel Happi Tietche · Olivier Romain · Bruce Denby

Received: 12 April 2013 / Revised: 19 July 2013 / Accepted: 6 August 2013 / Published online: 18 September 2013
© The Author(s) 2013. This article is published with open access at Springerlink.com

Abstract Many digital systems for telecommunications are implemented via the Software Defined Radio technique today. In such systems, digitally implemented modules to interface analog-to-digital converters with the rest of the system working at a different clock rate can be required. When implementing these modules, generated spurious harmonics and limited hardware resource problems can be critical factors in embedded applications. The article describes a Field-Programmable Gate Array (FPGA) circuit for arbitrary-ratio re-sampling of signals in the Low Frequency to Very High Frequency bands, intended for Software Defined Radio applications. The proposed resampler allows to control Spurious Free Dynamic Range while providing a simple, practical interface between the input and output clock domains that requires no additional clock, thus making it appropriate for FPGA clock-limited designs. Both up-sampling and down-sampling variants are presented. Resource utilization for FPGA implementations is also discussed.

Keywords Arbitrary Ratio · FPGA · SFDR · Re-sampling · Software Defined Radio

1 Introduction

It is often the case in radio communication systems and Software Defined Radio (SDR), that the initial sampling frequency is not convenient in some subsequent processing stage. Re-sampling, or sample rate conversion, SRC, is the operation of changing the sampling rate of a given digital signal. In most

cases, the required rate modification is not an integer multiple; this is what we refer to as arbitrary ratio re-sampling. Field-Programmable Gate Arrays (FPGAs) are often the technology of choice in such systems, however, FPGAs suffer from certain limitations such as the range of frequencies (<500 MHz) and circuit complexity which can practically be exploited. In the design of such systems, Spurious Free Dynamic Range, or SFDR, defined as the ratio of a signal to its largest spurious harmonic, is a natural quality criterion, as spurious harmonics may negatively influence the output of a processing chain and thus must be handled carefully.

The paper describes an FPGA-based, arbitrary ratio re-sampler for SDR applications that allows the designer to control SFDR while providing a simple solution requiring no additional clock, thus making it appropriate for FPGA clock-limited designs. The proposed SFDR-based approach provides estimates of resource utilization that reliably match actual resource utilization figures obtained after the FPGA circuit has been implemented, thus making it a valuable tool for design prototyping.

2 Related Work

In this section, we present a state of the art of existing re-sampling methods related to our work. Since the first polynomial interpolation structures proposed by Farrow in 1988 [1], the past decades have seen the arrival of numerous SRC architectures suitable for SDR that are amenable to an FPGA implementation. These may be grouped into four classes: Farrow structures-based architectures [2, 3]; Cascaded Integrator–Comb, or CIC-based [4–9]; polyphase FIR filters-based [10–13]; and hybrid methods [14–16]. Table 1 summarizes the methods listed above with their characteristics.

Unfortunately, none of these methods proposes a clear way of avoiding the use of a third clock with a frequency that is usually a multiple of the input or the output clock frequency,

B. Happi Tietche (✉) · B. Denby
Pierre and Marie Curie University/SIGMA laboratory, ESPCI,
Paris, France
e-mail: brunel.happi_tietche@etu.upmc.fr

B. Happi Tietche · O. Romain
ETIS laboratory, UMR8051, ENSEA/Cergy-Pontoise University,
Cergy-Pontoise, France

Table 1 Classification of some existing re-sampling methods.

Class	References	Characteristics	Third clock needed?
Farrow structure-based	[1–3]	Can be implemented without using memory resources ([3])	Yes
CIC-based	[4–9]	Don't need multiplication	Yes
Polyphase FIR filter-based	Dick and Harris [10]	-Can perform high SFDRs	Yes
		-More complex implementation	
	Jorgovanovic et al. [11]	-Support drift of the output clock average frequency	Yes (To perform high SFDRs)
		-Can perform high SFDRs	
Barker [12]	-should perform high SFDRs -FPGA implementation not treated	No	
Barker modified [13]	-should perform high SFDRs -FPGA implementation not treated	Yes	
Hybrid types	[14–16]	Perform better SFDR than the Farrow and CIC classes	Yes

as it is often proposed to manage the interface between the two clocks domains. This creates a problem in the context of VHF applications; indeed, the frequency of such a clock could quickly reach the limits of what the FPGA can support, or what a standard quartz oscillator can generate. By using a re-sampling structure similar to that of Barker [12], however, while at the same time properly interfacing the two clock domains, the classical FIR filter technique should in fact be able to provide practical re-sampler designs that do not include a third clock.

In this paper, we present the detailed implementation of such a structure, a re-sampler that is similar to the Barker structure, but which is based on a classical rather than polyphased FIR implementation, and which is distinct from the other re-sampling structures studied in the fact that it conveniently interfaces the two clock domains without the use of a third clock.

3 The Proposed Method

3.1 Theoretical Consideration

Let $H(n)$ be the discrete impulse response of the digital reconstruction filter to be used in our re-sampling process. There is an equivalent continuous impulse response $H_C(t)$ such that if we sample $H_C(t)$, we obtain $H(n)$. Indeed, from the value of the ratio F_{IN}/F_{OUT} , it is possible to precisely calculate the time position of each output sample from the input samples time positions. In other words, it is possible to calculate the output time instants from the fractional interval μ_k , the fraction of one input sampling period between an output sample k and the input sample m_k that immediately precedes it (Fig. 1). Using this time instant, along with $H_C(t)$, we can determine the impulse response coefficients which must weight the input

samples (Fig. 1). In this way, the filtering process can generate an output sample at any desired output instant.

In what follows, let B be the one-sided bandwidth of the input signal and β be the ratio $2.B/F_{IN}$. Two questions now remain, namely:

- Which type of filter shall we choose, and with which parameters?
- How shall we determine the weighting coefficients at each output instant?

As for the first question, the user can of course implement any type of lowpass FIR, such as sine cardinal (Sinc) filters windowed with different window types [17–22]; frequency sampling-based filters [23]; or filters designed with optimization methods [24–26]. The SFDR value obtained with our re-sampling process may be derived from the spectral width of the

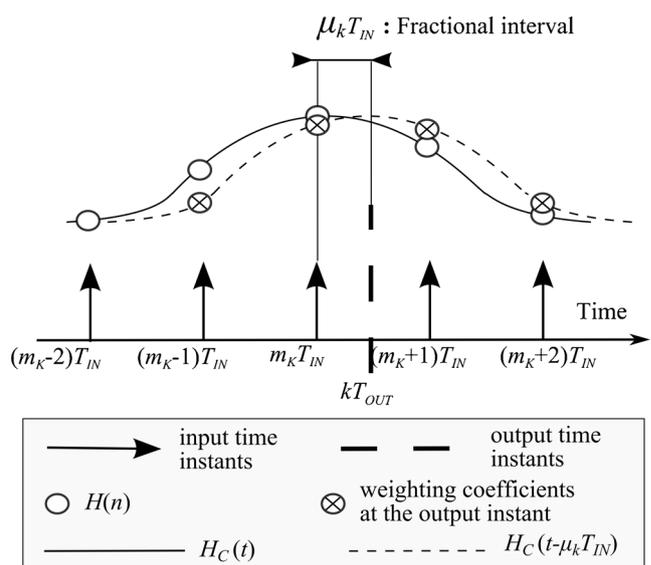


Figure 1 $H(n)$ and $H_C(t)$. The weighting coefficients at the output instant are obtained after $H_C(t)$ is delayed by $\mu_k T_{IN}$.

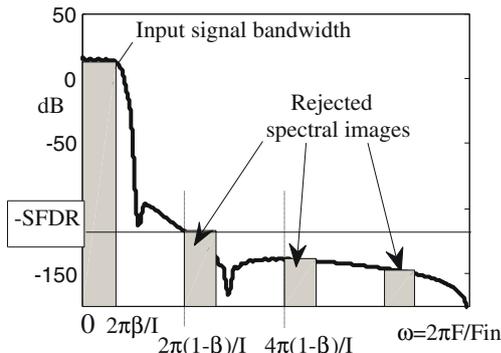


Figure 2 Approximate SFDR value. The curve represents the spectrum of the filter’s impulse response in dB.

the continuous-time version of the filter’s impulse response, $H_C(t)$. To approximate SFDR, we may up-sample $H(n)$ by a large factor I to obtain a new discrete response, $H_I(n)$ that better approximates $H_C(t)$. (The necessary interpolation can be easily made with mathematical software package, with I of the order of 10,000.) An approximate value of the SFDR is then given by the maximal amplitude of $H_I(n)$ above a frequency of $2\pi(1-\beta)/I$ (see Fig. 2). Thus, a filter can be considered to fulfill the SFDR requirement for a given application when, for a given order, the estimated SFDR is superior or equal to the minimum required SFDR, where we shall include plus a 10 dB margin for practical considerations.

For the second question, we first recall that $H_I(n)$ is used to calculate the filter weighting coefficients at each output instant by the nearest neighbor technique. Each coefficient chosen will be the $H_I(n)$ sample value nearest to the ideal value that we would have from $H_C(t-\mu_k T_{IN})$. Indeed, we must keep the stair-step error noise below the required SFDR, so that $I > 0.5 \times 10^{SFDR/20}$ is satisfied.

As the signal is quantized with a limited number of bits b , it is sometimes necessary that the SFDR be kept below the noise

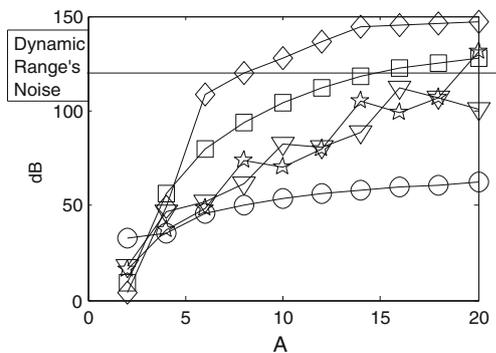


Figure 3 Estimated SFDR given by different popular filters. $A < 21$. \circ Rectangular, \star Parks-McClellan, \square Hann, ∇ Least-Square, \diamond Blackman Harris. The horizontal line “Dynamic Range Noise” indicates the upper limit due to the precision of the signal and the coefficients. The line, of which the level depends on that precision, acts as a maximum SFDR that the implemented re-samplers can reach. The 10 dB of margin mentioned above are not reported in this figure.

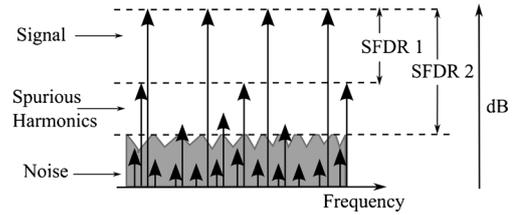


Figure 4 Illustration of the SFDR measured after implementation. The four largest arrows represent the 4 carriers that compose the signal. The other arrows represent the spurious harmonics. The grey represents the dynamic range noise. SFDR 1 is the SFDR measured when the harmonics are above the dynamic range noise. So, it is indeed the difference between the signal and the harmonics. SFDR 2 is the SFDR measured when the harmonics are inferior to and merged with the noise floor. It is the difference between the signal and the noise floor.

related to the dynamic range of the b -bit coded signal. This consideration leads to:

$$I > 2^{b-1} [10] \tag{1}$$

Since the coefficient calculation is made offline, there is no restriction on the duration of the processing. We can thus achieve the needed interpolation factor with a mathematical software package. Then, from $H_I(n)$, we obtain the samples corresponding to the output sampling instants.

As we wished to ensure that the theoretical SFDRs would correspond well to those obtained after implementation, we determined a set of SFDR values using the theoretical approach described above, and then confirmed these using actual experiments. Figure 3 shows the SFDR that we estimated with the process illustrated in Fig. 2 for $\beta=0.5$ for some popular filters as a function of A . The dynamic range (i.e., quantization) noise is represented on the figure by the horizontal line, which indicates the limit above which the spurious harmonics are lower than the dynamic range noise, as shown in Fig. 4. Above this

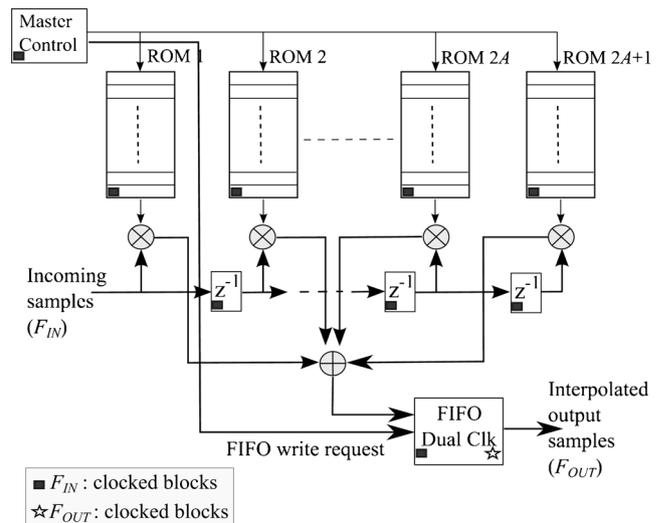
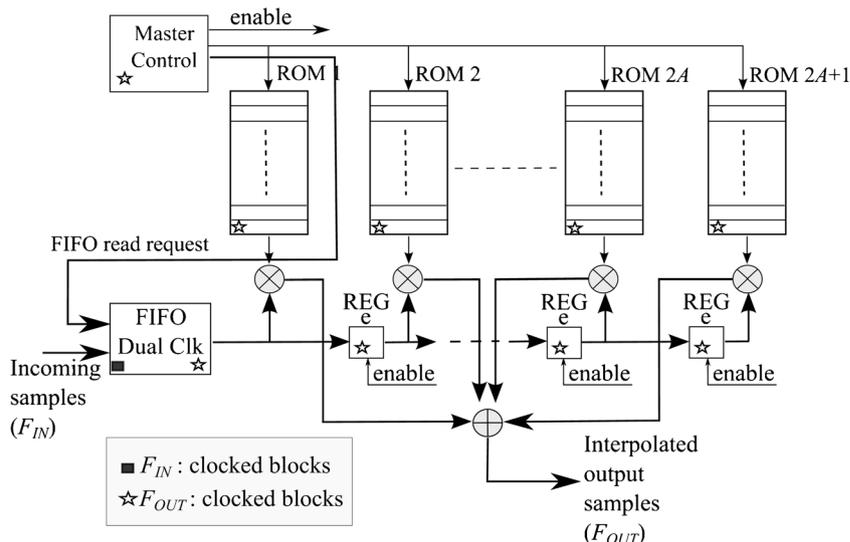


Figure 5 Arbitrary down-sampler architecture.

Figure 6 Arbitrary up-sampler architecture.



line, the SFDR measured after implementation is the difference between the signal and the dynamic range noise.

3.2 Hardware Architecture

Let Q and N be respectively the numerator and denominator of the simplified version of the fraction F_{IN}/F_{OUT} . Let A be an integer so that $(2A + 1)$ is the reconstruction filter order.

3.2.1 Down – Sampler

The down-sampler system is summarized in Fig. 5. The coefficient values are stored in $(2A + 1)$ Read Only Memories, ROMs, each of which has N memory locations. We recall from Fig. 1 that each μ_k is associated with one output sample; we thus know precisely where output samples are located with regard to the input samples during one period of μ_k . Thus, each ROM stage contains a coefficient value that weights an input sample at a given output time instant within this period, in order to calculate an output sample. Stated another way, at the i^{th} instant in a period, the i^{th} location of ROMs 1 to $(A + 1)$ weights the $(A + 1)$ input samples to the left of the output sample being calculated, and the i^{th} stage of ROMs $(A + 2)$ to $(2A + 1)$ weights

the A input samples to the right. The weighted input samples are then summed with a parallel adder to give the output sample.

We note that the output sample calculation is entirely synchronized to F_{IN} . The “dual clock” FIFO makes the output samples formerly calculated at F_{IN} pass to the F_{OUT} clock domain. We call D_{EP} the depth of this FIFO. In order to tackle the arbitrary ratio problem, the “Master Control” block, which is the core of the structure, precisely controls the incrementation of the ROM indices when a new output sample is to be calculated using the input samples that arrive in the registers.

Master Control also avoids FIFO overflow. Indeed Fig. 5 shows that if a data word is stored in the FIFO on every rising edge of the F_{IN} clock, the FIFO will overflow because $F_{IN} > F_{OUT}$. To avoid the overflow, Master Control ensures that only valid output samples enter the FIFO via the “FIFO Write Request” signal. To fulfill these two roles, Master Control keeps in memory a simplified “down-sampling map” of the output samples with regard to the inputs samples. This map is a binary vector of length Q . A map element is 1 if the input time immediately precedes an output time; otherwise, it is 0. Master Control uses the map to increment the ROM indices at the appropriate time and assign the “FIFO Write Request” signal. Thus, even though the write of the FIFO is clocked on F_{IN} ,

Table 2 Resource estimation.

Memory in bits for down-sampling	$(2A + 1) \times b \times N + Q + (2b + x) \times D_{EP} (*)$
Memory in bits for up-sampling	$((2A + 1) \times b + 1) \times N + b \times D_{EP}$
Multipliers $b \times b$ bits	$2A + 1$
Adders	1 parallel adder $(2A + 1) \times 2b$ bits

* $2^{x-1} < 2A < 2^x$

Table 3 FPGA-based Down-sampler resources.

ALUTs	298/143.520 (<1 %)
Dedicated logic Registers	560/143.520 (<1 %)
Total Registers	560
Memory bits	54.707/9.383.040 (<1 %)
DSP block 9 bit	26/768 (3 %)
Maximal F_{IN}	120.83 MHz
Maximal F_{OUT}	146.09 MHz

Table 4 Comparison between theory and experimental results: Down-sampler.

Memory in bits	$((2 \times 6 + 1) \times 12) \times 256 + 435 + (2 \times 12 + 4) \times 512 = 54707$
Multipliers 12x12 bits	13
Parallel adder 13x24 bits	1

actually it is written at a rate equivalent to it F_{OUT} . We can easily determine the appropriate map configuration with a mathematical software package such as Matlab.

3.2.2 Up-Sampler

The structure of the up-sampler is similar to that of the down-sampler (Fig. 6). The FIFO retains its role of interface between the two clock domains. Master Control manages the reading of the FIFO in order to avoid underflow. Indeed, using an “up-sampling map”, Master Control ensures that the ROM indices are incremented and the FIFO read at the correct instants.

The binary map for up-sampling is of course not identical to the one for down-sampling. Whereas for down-sampling the map is synchronized on the input times, for up-sampling it is synchronized on the output times. When a map element is set to 1, for up-sampling we must increment ROM indices. Master Control also enables registers “REG e” which are registers with an “enable” input. When “enable” is set low, the “REG e” output keeps its previous value. Otherwise, it works like a simple D Flip-Flop register; this is useful when we calculate two or more output samples from the same input samples.

3.2.3 Pre-Implementation Resource Estimation

To summarize, the resampler uses $(2A + 1) \times N + 1 + D_{EP}$ memory locations, namely $(2A + 1) \times N$ for the filter’s coefficients, 1 for the map and D_{EP} for the dual clock FIFO. It is easy to

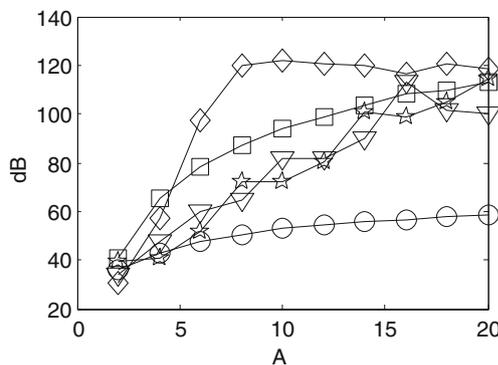


Figure 7 Experimental SFDR results for down-sampling when $A \leq 20$. \circ Rectangular, \star Parks-McClellan, \square Hann, ∇ Least Square, \diamond Blackman Harris. The 19-bit noise floor, 116 dB, explains why the obtained results are upper limited.

Table 5 Down-sampler tests.

	Carrier frequencies (MHz)	F_{IN} (MHz)	F_{OUT} (MHz)	F_{OUT}/F_{IN}
Signal 1	$1 + C \times 2.2$ (*)	87	51.2	0.588506
Signal 2	$1.95 + C \times 2.2$ (*)	87	51.2	0.588506
Signal 3	$0.8 + C \times 2.3$ (*)	86	74	0.860465
Signal 4	$0.3 + C \times 2.3$ (*)	86	74	0.860465

* $C = \{0, 1, \dots, 9\}$

estimate the resource utilization before the implementation phase of the design flow. Indeed, D_{EP} can take on any value set by the designer. Now, once A is chosen from Fig. 3, D_{EP} set, and N and Q determined, an estimation of the required resources can be obtained using Table 2 where input samples and filter coefficients are b -bit words.

Before concluding this section, we stress that in Table 2, the “down-sampling” or “up-sampling” map is considered as a single memory element. Also, multipliers of $b \times b$ -bit size and parallel adders can of course be implemented in many different ways (DSP blocks or logic elements), depending on the FPGA type and the implementation chosen by the designer.

4 Case Study

This section details the experimental tests made on our re-sampling system as well as the comparison of the obtained results to the theoretical expectations.

4.1 Down-Sampler

We propose to implement a down-sampler to process the VHF Band II generally used for FM broadcasting, which extends over the range 87.5–108 MHz, by re-sampling the band 87–108.75 MHz. For this example we choose to down-sample from 87 MHz to 51.2 MHz. The input samples are 12-bit words. As the input signal bandwidth is 21.75 MHz, we have $F_{IN} = 4$. B

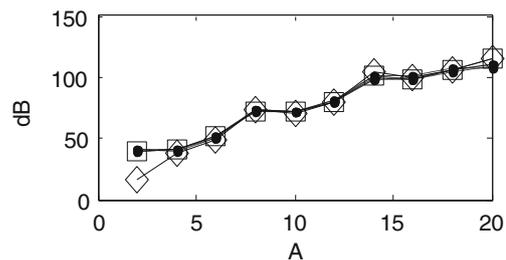


Figure 8 Experimental and theoretical SFDR results for a Parks-McClellan filter, and for down-sampling when $A \leq 20$. \square Experimental FPGA results, \diamond theoretical SFDR, \bullet simulation results. The 19-bit noise floor, 116 dB, explains why the obtained results are upper limited by that level.

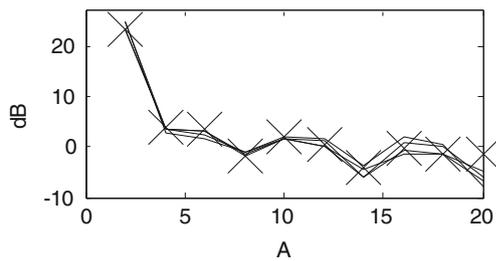


Figure 9 Errors between the SFDRs results reported in Fig. 8 and the theoretical SFDRs reported in Fig. 3, for a Parks-McClellan filter. X error of the experimental FPGA results, — error of the simulation results.

(where B is the one-sided bandwidth of the input signal) so $\beta = 0.5$. As $\beta = 0.5$, this test example is in the same conditions as the predictions of Fig. 3. We choose to set the SFDR below the noise floor: $\text{SFDR} \geq 72$ dB for 12 bits. Thus, expression (1) leads to $I_2 > 2,048$. According to Fig. 3, we choose $A = 6$ with a Blackman-Harris window. Clearly $N = 256$, and we also choose to set D_{EP} to 512, for this example.

4.1.1 FPGA Implementation

The design is implemented on an Altera Stratix II EP2S180F 1020C5 FPGA chip using a Stratix II DSP development board. Complete hardware design is done in Altera Quartus II v9.1, with synthesis and implementation. Table 3 summarizes the hardware resources used.

4.1.2 Comparison Between Theory and Experimental Results

The comparison will be based upon two points: the SFDR and the amount of resources used. Concerning the former, the measured SFDR is 73.5 dB, and spurious harmonics are kept below the noise floor, as expected. Then, according to Table 2, the down-sampler resources are given in Table 4.

According to the FPGA type and the design chosen by the user, the multipliers are implemented differently. In our case, we implemented the multipliers using DSP blocks only. In Quartus 18×18 element DSP blocks are synthesized to replace 12×12 element DSP blocks, which are not available in the Stratix II. One 18×18 DSP block is made from two 9×9 blocks. So, the 13 predicted element DSP blocks of Table 4 should be synthesized as 26 9×9 blocks. Furthermore, Quartus implements parallel adders on Stratix II using ALUTs and logic registers. Taking in account these considerations, if we compare Table 4 to the implementation results given in Table 3, we see that implemented down-sampler uses the memory and multipliers resources as expected. The resources used by the parallel adder and any other part of the down-sampler are reported in Table 3 in the boxes ALUTs and registers.

Two series of tests were performed on the down-sampling system to confirm the theoretically predicted SFDR results of Fig. 3 for the case $A \leq 20$. The first series consisted in measuring the SFDR generated by an input test signal (which is the sum of 4 sine signals) for different FPGA re-sampler implementations (i.e., different filter types, with $A \leq 20$). Fig. 7 shows the results.

These experimental results may then be compared to the theoretical predictions. Before doing this, however, in order to assure ourselves that the results of Fig. 7 are not the result of a fortuitous choice of carrier frequencies of F_{IN} and F_{OUT} , in our second series of tests, we built a Matlab test bench that simulates precisely our down-sampler and thus gives results identical to what the hardware down-sampler would give.

This test bench determined the SFDR produced by our down-sampler for 4 different choices of input signals, and for 2 different F_{OUT}/F_{IN} ratio values, with $\beta = 0.5$. Each input signal here is a sum of ten sine signals with carrier frequencies chosen so that $F_{IN} = 4B$. The input signal and the weighting values are 19-bits words and I_2 is set according to (1). Table 5 summarizes the tests carried out.

The second series of tests were also performed for different filter types. In Fig. 8, the results of the two series of tests in the case of the Parks-McClellan filter are summarized, where each set of curves presents is compared to the corresponding theoretical result that we saw in Fig. 3. As it is explained at the end of Section 3.1, this theoretical result is upper bounded by an SFDR value imposed by the noise floor, which is approximately 116 dB for 19-bit. For each tested filter, the errors between the SFDRs results and the theoretical SFDRs were calculated too. Fig. 9 shows these errors for the Parks-McClellan filter. For the other filter types, as the case of Fig. 9, the calculated errors reveal that the results given by the down-sampling system are globally positive as these errors rarely go down below -10 dB. Also, it justifies the choice of a 10 dB-margin, as mentioned in Section 3.1. The hardware experimental results are thus in reasonable accord with theory.

4.2 Up-Sampler

Similar tests as those performed in Section 4.1 on a down-sampler were performed on an up-sampler. The results, which for space reasons are not included here, demonstrate that the proposed system is applicable for up-sampling as well.

5 Conclusion

We have presented a practical FPGA re-sampling architecture suitable for SDR implementations, which takes as its starting

point the Barker structure [12]. Compared to other SRC systems suitable for FPGA designs, ours does not require a third clock at a multiple of F_{IN} or F_{OUT} , as is the case with the majority of the methods cited in Section 2, and is thus suitable even for very high sampling frequencies. The SFDR requirement of the targeted application is incorporated directly into the design process via a simple calculational procedure. It is furthermore shown in Table 4 that the expected resource utilization estimated in the design phase closely matches actual resource utilization after hardware implementation. Although not intended for applications in which input and output clocks are completely decoupled (see for example [10, 11]), the proposed structure provides a simple, practical alternative to other techniques of re-sampling, for the use cases treated.

Open Access This article is distributed under the terms of the Creative Commons Attribution License which permits any use, distribution, and reproduction in any medium, provided the original author(s) and the source are credited.

References

- Farrow, C. W. (1988). *A continuously variable digital delay element* (3rd ed., pp. 2641–2645). Espoo: IEEE International Symposium on Circuits and Systems.
- Erup, L., Gardner, F., & Harris, R. (1993). Interpolation in digital modems – Part II: Implementation and performance. *IEEE Transactions on Communications*, 41(6), 998–1008.
- Quang, D. X. (2010). *Digital resampling and timing recovery in QAM systems*. Saskatchewan: Master of Science thesis, Saskatchewan University, Electrical Engineering Department.
- Saramaki, T., & Ritonieni, T. (1997). *A modified comb filter structure for decimation* (4th ed., pp. 2353–2356). Hong Kong: Proc. IEEE International Symposium Circuits and Systems, ISCAS'97.
- Jovanovic-Dolecek, G., & Mitra, S. K. (2005). A new two-stage sharpened comb decimator. *IEEE Transactions on Circuits and Systems I*, 52(7), 1414–1420.
- Nerurkar, S. B., & Abed, K. H. (2006). Low-power decimator design using approximated linear-phase N-band IIR filter. *IEEE Transactions on Signal Processing*, 54(4), 1550–1553.
- Abu-Al-Saud, W. A., & Stuber, G. L. (2003). Modified CIC filter for sample rate conversion in software radio systems. *IEEE Signal Processing Letters*, 10(5), 152–154.
- Jovanovic-Dolecek, G., & Mitra, S. K. (2006). *Stepped triangular CIC filter for rational sample rate conversion* (pp. 916–919). Singapore: IEEE Asia Pacific Conference on Circuits and Systems, APCCAS 2006.
- Jovanovic-Dolecek, G. (2007). *Modified CIC filter for rational sample rate conversion* (pp. 252–255). Sydney: IEEE International Symposium on Communications and Information Technologies, ISCIT'07.
- Dick, C., & Harris, F. (2004). *Options for arbitrary re-samplers in FPGA-based modulators* (1st ed., pp. 777–781). Pacific Grove: Conference Record of the 38th IEEE Asilomar Conference on Signals, Systems and Computers.
- Jorgovanovic, M., Pajic, M., Kvascev, G., & Popovic, J. (2008). *FPGA design of arbitrary down-sampler* (pp. 391–394). Nis: 26th IEEE International Conference on Microelectronics, MIEL 2008.
- Barker, D. E. W. (2008). Efficient re-sampling implementations [DSP Tips & Tricks]. *IEEE Signal Processing Magazine*, 25(4), 114–117.
- Xu, Y.-J., Wang, H.-Y., & Shen, Z. (2010). *Modified polyphase filter for arbitrary sampling rate conversion* (pp. 1–4). Chengdu: 6th IEEE International Conference Wireless Communications Networking and Mobile Computing, WiCOM.
- Babic, D., & Renfors, M. (2005). Power efficient structure for conversion between arbitrary sampling rates. *IEEE Signal Processing Letters*, 12(1), 1–4.
- Ramstad, T. (1984). Digital methods for conversion between arbitrary sampling frequencies. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(3), 577–591.
- Evangelista, G. (2003). Design of digital systems for arbitrary sampling rate conversion. *Signal Processing*, 83(2), 377–387.
- Kaiser J. F. (1974). Nonrecursive digital filter design using the I_0 -sinh window function. In *Proc. 1974 I.E. International Symposium on Circuits and Systems* (pp.20–23). April.
- Oppenheim, A. V., & Schaffer, R. W. (1999). *Discrete-time signal processing* (pp. 468–471). Upper Saddle River: Prentice-Hall.
- Oppenheim, A.V., & Schaffer, R.W. (1989). *Discrete-Time Signal Processing* (pp. 447–448). Prentice-Hall.
- Nuttall, A. H. (1981). Some Windows with Very Good Sidelobe Behavior. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29(1), 84–91.
- Harris, J. F. (1978). On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform. *Proceedings of the IEEE*, 66(1), 51–83.
- Tukey, J. W. (1968). *An introduction to the calculations of numerical spectrum analysis* (pp. 25–46). New York: Spectral Analysis of Time Series. Wiley.
- Mitra, S. K. (1998). *Digital signal processing: A computer based approach* (Firstth ed., pp. 462–468). New York: McGraw-Hill.
- Parks, T. W., & Burrus, C. (1987). *Digital filter design* (pp. 54–83). Canada: John Wiley & Sons.
- Programs for Digital Signal Processing* (Algorithm 5.1). IEEE Press, New York, 1979.
- Selesnick, I.W., Lang, M., & Burrus, C.S. (1995). Constrained Least Square Design of FIR Filters without Specified Transition Bands, In *Proc. IEEE International Conference on Acoustic, Speech and Signal Processing, ICASSP-95* (vol. 2, pp. 1260–1263). May.



Brunel Happi Tietche was born in Cameroon. He received the engineering degree in electronic systems from SUPELEC engineering school, France, in 2009. Since 2010, he has been working toward the Ph.D. degree at the Pierre and Marie Curie University, Paris, France. His research interests include signal processing systems for broadcasting radio standards, software radio systems and signal processing.



Olivier Romain holds an engineering degree from ENS Cachan, MS from Louis Pasteur University, and PhD from Pierre and Marie Curie University, Paris, all in Electronics. Since 2012, he is head of the architecture department at ETIS laboratory. He is full professor of electrical engineering at Cergy-Pontoise University. His research interests are in system on chip for broadcast and biomedical applications.



Bruce Denby holds a BS from Caltech, MS from Rutgers University, and PhD from the University of California at Santa Barbara, all in Physics. Since 1995 he has been full professor of electrical engineering in France, currently at the Pierre and Marie Curie University in Paris. His research interests are in signal processing and wireless communications.