Open and Configurable Channelization for Future Software-Defined Radio

Ignacio Milani, Carlos Zerbini, Guillermo Riva
Grupo de Investigacion y Transferencia en Electronica Avanzada (GInTEA)
Universidad Tecnologica Nacional, Facultad Regional Cordoba (UTN-FRC)
Cordoba, Argentina
ignaciomilani@gmail.com, carloszerbini@gmail.com, guilriva@gmail.com

***Abstract*—Nowadays, signal demodulation chains used in Software-Defined Radio (SDR) receivers involve analog and digital sections. For economical reasons, the latter tends to gain ground over the analog one, and typically involves a channelization module based on Digital Down-Converters (DDCs). Given the current trend towards all-digital receivers based on direct RF sampling, the channelization architecture will become a critical issue, requiring open and flexible designs to cope with its challenges. As contribution, we evaluate two open alternatives for DDC's implementation on field-programmable gate arrays (FPGAs), one implemented in high-level language and the other one through hardware description language (HDL). Together, they provide flexibility for parameterization, as well as hardware-level optimization.**

***Index Terms*—All-digital receiver, SDR, channelization, DDC, FPGA**

I. Introduction

DDCs are a critical factor for the evolution of SDR towards direct RF sampling architectures. In addition, the flexibility of FPGAs makes these devices the natural choice for implementing DDCs. Multiple designs are available for FPGAs; however, based on our research, most of them are proprietary [1] or poorly documented [2], are valuable personal projects which require further improvements [3] or are oriented towards education [4]. In this work, we review current options for implementation of DDCs in FPGAs, and propose two complimentary choices that help approaching their high- and low-level challenges. In addition, we provide analysis of resource consumption and scalability on FPGAs.

II. Channelization in SDR

A. General approaches for channelization

Typically, SDR systems capture a relatively wide spectrum band depending on its sampling frequency, and rely on *channelizers* to extract one or multiple zones of interest for subsequent baseband processing. Most popular channelization techniques are, namely, a) Digital Down Conversion (DDC), b) Frequency Domain Filtering (FDF), and c) Polyphase FFT Filter Banks (PFFB) [5].

DDCs base on mixing and subsequent decimation and filtering to extract the channel of interest. DDCs are very flexible for adapting to variable channel widths and shifts, but they are not suited to massive input channels. FDF, meanwhile, begins by implementing the FFT of the input signal, and then implements mixing, decimation and filtering by simply extracting the adequate bins in frequency domain. As drawback, they require computation of the FFT on input signal, and inverse FFT must be applied to selected bins in order to perform baseband processing. The PFFB channelizer, finally, assumes equally-spaced channels, and on this basis decomposes the decimating lowpass filter onto an extremely efficient polyphase filter, followed by a resource-demanding FFT block.

In this work, we will focus on channelizers based on DDC techniques, due the advantages of flexibility in selecting both the carrier frequency and channel bandwidth for the mid-sized SDR receiver. For increased flexibility, the FDF approach could be considered, while PFFB would be the choice for hundreds of redundant channels.

B. Digital Down-Converters (DDCs)

Depending on the SDR generation [6], the DDC can be used to downconvert two already demodulated and digitized orthogonal signals (Gen1, e.g., BladeRF, LimeSDR, or Ettus B200), or to downconvert a still modulated signal at intermediate frequency (IF) onto a complex baseband signal, (Gens. 2 and 3, e.g., analog-IF and direct sampling such as Panoradio and Pi-radio). Current generation of direct-sampling SDRs tends to minimize analog front-end due to its inherent issues regarding linearity, DC offsets, I/Q path mismatches, etc., putting the DDC in charge of these tasks.

Fig. 1 shows a typical Gen. 2/3 structure, where the DDC involves three main elements: 1) Numerically Controlled Oscillator (NCO) generating two orthogonal sinusoids at IF or RF, 2) two mixers for getting the orthogonal I and Q baseband signals, and 3) a downsampling stage to adapt sample rates, and low-pass filtering to remove the involved aliasing. Wideband DDCs, i.e., those with moderate downsampling ratio, can be implemented through mixers followed by FIR filters in software. Narrowband DDCs, meanwhile, require faster filtering schemes implemented on DSPs or FPGAs, such as the Hogenauer or Cascade Integrator Comb (CIC) filters, and combinations of CIC and FIR in multiple stages [4] [7].

C. CIC filters

As side effect of downsampling in DDCs, aliasing arises, requiring digital low-pass filtering. When high conversion

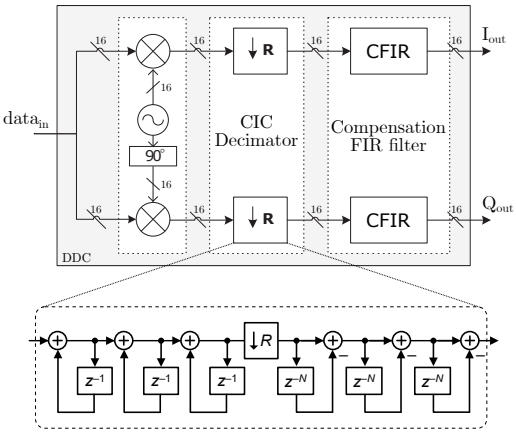

Fig. 1. Digital Down-Converter architecture.

ratios are required, i.e., above 8, CIC filters are the best choice since they only require adders in contrast to FIR filters which also require multipliers. In addition, improvements such as polyphase techniques add to their efficiency. However, since CIC filter essentially implements moving average, its frequency response follows a *sinc* shape that suffers a steep droop which must be corrected through a compensating FIR filter. As shown in Fig. 1, CIC filters involve an *integrator section* which is a recursive accumulator running at input frequency f_{in}, followed by *downsampling* by a factor R, and the *differentiator (i.e., comb) section* running at rate f_{out} which subtracts the D delayed result. Since the comb runs R times slower than the integrator, the delay chain is shortened to $N = D/R$ length.

When downsampling by R, multiple aliases fold at baseband which are minimized by cascading Q CIC stages. In addition, as consequence of the integrating nature of the filter, it has gain D^Q, so a final gain correction is needed and special care has to be taken to avoid overflow. In general, $N = D/R$ affects the zeros of the frequency response, while Q affects its shape. For increasing R, changes in frequency response are negligible as long as $N = D/R$ is kept constant, which is very convenient for variable-bandwidth SDRs. Further analysis of CIC filters can be found in [4] and [8].

III. Two open DDC alternatives

A. High-level implementation

For our first approach, we based on a DDC example offered by MathWorks [9]. The design is aimed at a typical application, running at input clock rate 122.88 MHz, standard for 5G radios, and presents an overall downsampling ratio of 64, giving an output sample rate of 1.92 Msps. This output rate is the typical sampling rate used by 4G and 5G receivers for cell search and Master Information Block (MIB) recovery, which is its essential control information.

In this case, as shown in Fig. 2, the DDC consists of NCO, mixer and subsequent decimating filter chain. Since the overall decimation ratio is 64, multiple decimating stages are required. Thus, the chain consists of a CIC decimator filter (R=8, N=1, Q=3), CIC gain correction for normalization of large CIC gain; Compensation FIR filter to flatten CIC passband and provide additional decimation by two, efficient half-band FIR decimator by two, and a final FIR decimator.

B. Low-level implementation

To evaluate the effects of hardware-level parameterization and optimization, HDL implementations of DDC were surveyed. DDC/DUC compilers and cores are offered by FPGA providers; however they are closed and sometimes deprecated designs [1].

Some SDR platforms, such as NI Ettus, have available FPGA HDL code [2]; however it is not documented, which makes its reuse or modification very difficult. Among open choices, an interesting project aiming at FPGA-based SDR cores is the one by Tsoeunyane et al. [10], however it involves excessive complexity for our current goals. In addition, a number of open, partial implementations exist which can be reused, however they require careful review for correctness and completion for its effective application. In particular, we based on the work by Tavares [3], improving it and adding the compensation FIR filter according to the guidelines in [11] and [12]. As seen in Fig. 3, the first step is the complex mixer which receives both the real, modulated signal and two orthogonal sinusoids from the NCO. Efficient NCOs are commonly implemented as CORDIC cores [7]. The next step is the downsampling filter, formed by the decimating CIC (R=8, N=1, Q=3), gain correction stage, and final decimating FIR filter to compensate for the steep magnitude droop of the CIC. This approach was designed for an input clock rate of 100 MHz, resulting in an output sample rate of 12.5 Msps.

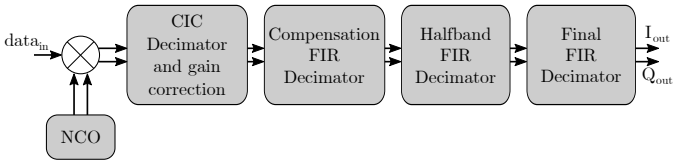

Fig. 2. Block diagram of high-level DDC implementation.

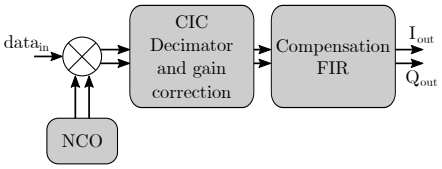

Fig. 3. Block diagram of low-level DDC implementation.

IV. Results

A. High-level implementation

In first place, the filtering chain is designed in Matlab according to the requirements $f_{passband} = 540$ (36 15-kHz LTE subcarriers, enough for our goals), $f_{stopband} = 700\ kHz$, passband ripple = 0.1, and stopband attenuation =

TABLE I
HIGH-LEVEL DDC: RESOURCE UTILIZATION (ONE INSTANCE)

Module	Slice LUT	Slice Register	Slice	LUT as Logic	LUT as Memory	DSPs
NCO/Mix	171	209	69	170	1	4
CIC_Dec	309	391	102	309	0	0
Comp_FIR	162	333	85	159	2	2
HB_Dec	301	546	156	299	2	4
Final_Dec	987	2745	648	987	0	6
Top level	2119	4849	1179	2082	37	16

TABLE II
LOW-LEVEL DDC: RESOURCE UTILIZATION (ONE INSTANCE)

Module	Slice LUTs	Slice Register	Slice	LUT as Logic	LUT as Memory	DSPs
NCO/Mix	71	70	43	71	0	2
CIC_Dec	282	432	106	182	0	0
Comp_FIR	604	416	226	588	8	5
Top level	962	926	324	946	16	12

60 dB. Fixed-point behaviour is evaluated against the reference design. Then, the NCO for the mixing stage is designed according to desired frequency resolution and spurious-free dynamic range (SFDR), generating a ROM-based lookup table. Finally, all the components are integrated in a Modelsim block used for simulation of the DDC and subsequent generation of HDL code. The design is verified both by HDL simulation and proof-of-concept on a Xilinx Zynq Z7020-based Zedboard platform, by applying a RAM-stored test tone and monitoring the output by means of an Integrated Logic Analyzer (ILA). Resource utilization is shown in Table I. Most resources are consumed by the final FIR decimator which needs to give final shape to the frequency response; however it runs at the lowers rate of the chain. As shown, the CIC needs no DSP blocks at all. It it worth to note that top-level requirements are slightly smaller than the sum for modules, since slice reuse occurs.

B. Low-level implementation

In this case, utilization reported in Table II is in general lower since this design downsamples by R=8 against R=64 for the high-level design. As a consequence, only one CIC stage and compensation FIR is required. NCO requires far less slices since this implementation does not impose frequency resolution and SFDR requirements as the high-level does. On the other hand, the compensation filter in this case requires 15 coefficients against 7 coefficients in the high-level case, therefore requiring considerable higher resources (i.e., 604 LUTs against 162 LUTs, etc.). This can be attributed to the single-stage architecture.

TABLE III
REPORT UTILIZATION FOR FOUR INSTANCES (BOTH APPROACHES)

DDC	Slice LUTs	Slice Register	Slice	LUT as Logic	LUT as Memory	DSPs
High-L	8515	19381	4764	8367	148	64
Low-L	3855	3704	1380	3791	64	48

We finally replicated both designs four times to appreciate scalability for multi-channel SDRs. As seen in Table III, both designs scale as expected, supporting multiple channels with very low percentage of available resources on a typical cost-optimized FPGA device as the Zynq Z7020.

Regarding timing results, $f_{clk} = 122,88$ MHz were required to the high-level design and $f_{clk} = 100$ MHz to the low-level one, and both are met by the designs. Worst Negative Slack (WNS) for the high-level implementation was 1.294 ns while WNS for the hardware-level was 6.591 ns, showing a safe margin.

V. CONCLUSIONS

In this work, we propose and evaluate two open alternatives for DDC's implementation on FPGAs, one implemented in high-level language and the other one through HDL. Together, they provide flexibility for parameterization, as well as hardware-level optimization, both valuable features in research and development of current and future SDR systems. As future work, further optimization as well as FDF/PFFB options are considered.

ACKNOWLEDGMENTS

This research work was supported by Programa Estimulo a las Vocaciones Científicas (EVC), Consejo Interuniversitario Nacional (CIN), and PID CCUTNCO0007833, Monitoreo de Calidad de Servicio en Redes de Comunicaciones Moviles, Secretaría de Ciencia y Tecnología (SCyT), UTN-FRC.

REFERENCES

[1] Xilinx Corp., 2022. "DDC/DUC Compiler". [Online]. Available: https://www.xilinx.com/products/intellectual-property/duc_ddc_compiler.html.
[2] Ettus Research, 2022. "USRP Hardware Driver and USRP Manual". [Online]. Available: https://files.ettus.com/manual/page_images.html#images_building_xilinx.
[3] Digital down converter - IE309E Unicamp (2022). [Online]. Available: https://github.com/danielot/ddc.
[4] U. Mayer-Baese, Digital Signal Processing with Field Programmable Gate Arrays, Springer, 2014.
[5] L. Pucker, "Channelization techniques for software defined radio, " in Proc. of SDR forum conference, 2003, pp. 1–6.
[6] R. W. Stewart, K. W. Barlee, D. S. W. Atkinson, L. H. Crockett, Software Defined Radio using MATLAB and Simulink and the RTL-SDR. Strathlclyde Academic Media, 2015.
[7] A. Abinaya and M. Maheswari, " A survey on digital down converter architecture for next generation wireless applications, " IOP Conf. Series: Materials Science and Engineering, vol. 872, pp. 12-37, 2020.
[8] R. Lyons, 2022. "A beginner's guide to cascaded integrator-comb (cic) filters". [Online]. Available: https://www.dsprelated.com/showarticle/1337.php
[9] The MathWorks, Inc., 2022. "Implement Digital Downconverter for FPGA". [Online]. Available: https://la.mathworks.com/help/dsp/ug/hdl-implementation-of-digital-down-converter-for-LTE.html.
[10] L. Tsoeunyane, S. Winberg, and M. Inggs, "Software-defined radio fpga cores: Building towards a domain-specific language, " International Journal of Reconfigurable Computing, vol. 2017, pp. 1–28, 07, 2017.
[11] Intel Corp., 2022. "Understanding CIC Compensation Filters". [Online]. Available: http://www.altera.com/literature/an/an455.pdf.
[12] Octave/MATLAB code for generating compensation FIR coefficients (2022). [Online]. Available: https://github.com/ericgineer/CIC_Octave_Matlab.