Frequenz
64 (2010)
11–12

Hardware Optimized Sample Rate Conversion for Software Defined Radio

Carina Schmidt-Knorreck, Raymond Knopp, and Renaud Pacalet

Abstract – The evolution towards applications with increasing functionalities leads to the need of high flexible systems that support a high number of different standards while decreasing the required hardware space. Therefore a high configurable platform being able to handle a multitude of standards is needed. One main issue is the tradeoff between performance and space consumption. We present a generic, flexible, fractional and hardware optimized SRC architecture in the context of SDR, providing one architecture to process up to 8 different complex channels. The solution is based on bandlimited interpolation and allows processing by supporting a 1Hz resolution of the sampling rates.

Index Terms – SDR, SRC, Hardware Architecture, Open Platforms for Multistandard Support, Baseband Processing, HW Accelerators

1 Introduction

In the past years, the number of different standards used in wireless communications (GSM, OFDM, WCDMA etc) has grown rapidly. New products deliver more functionality to the users and merge already existing ones in only one device. This evolution leads to the need of high flexible systems that support a high number of different standards while decreasing the required hardware space. Developing these systems is a challenging task as the needed architecture differs from standard to standard; due to different carrier frequencies, channel bandwidths and modulation schemes. The solution can be found in the concept of Software Defined Radio (SDR). By definition, SDR is a reconfigurable radio communication system that can be tuned to any frequency band and that can handle all the modulation schemes in a wide frequency range [6].

The digital baseband processing platform being developed by Eurecom and Telecom ParisTech is a generic prototype architecture for SDR applications [3]. Its multimodal design supports almost all existing standards and allows an easy adaptation of future technologies without changing the HW/SW architecture. The partitioning between HW and SW follows a general cost-and-complexity versus speed trade-off. One of the main objectives in design is to find the most flexible solution with highest performance and minimal space consumption. Furthermore the design has to meet the throughput and the latency requirements of the computationally most intensive task. The relevant parameters may differ between the standards and are managed by the software part that synchronizes the different processes on the platform.

One of the critical processing blocks in terms of performance and space consumption is the preprocessor that connects the external radio front end with the entire platform. In the context of this paper we focus on its most demanding element – the fractional Sample Rate Converter (SRC). First the different requirements are listed in section 2, before a short overview of the preprocessor is given in section 3. Different existing solutions are examined in section 4 and evaluated in terms of performance and space consumption. An architecture based on one of those algorithms is presented in section 5 and the performance results are part of section 6. At the end possible extensions of the filter design are summed up in section 7.

2 Requirements

The requirements can be divided into two different groups. First the non-functional requirements due to the platform design, and second, the functional requirements related to sample rate conversion. From the platform perspective, the general architecture of the preprocessor is the same like for the other blocks for the digital baseband processing (Frontend Processor, Channel Decoder, etc). The standardized IP shell that can be seen in Figure 1 allows the reuse of most of the control and communication logic and an easy upgrading or replacement of the block in the future. In contrast to the DMA, the VCI Interface and the Micro-Controller, the architecture of the IP Core and the Memory Subsystem (MSS) depends on the functionality of each block.

Another requirement results from the target architecture, a Xilinx Virtex 5 LX330FF1760 FPGA. During synthesis, so called Extreme DSPs (DSP48E slices) are invoked whenever possible. These elements speed up the design significantly and are not only used by the preprocessor but also by the frontend processor for instance. In order to minimize the required hardware space, the SRC has to be as simple as possible. This

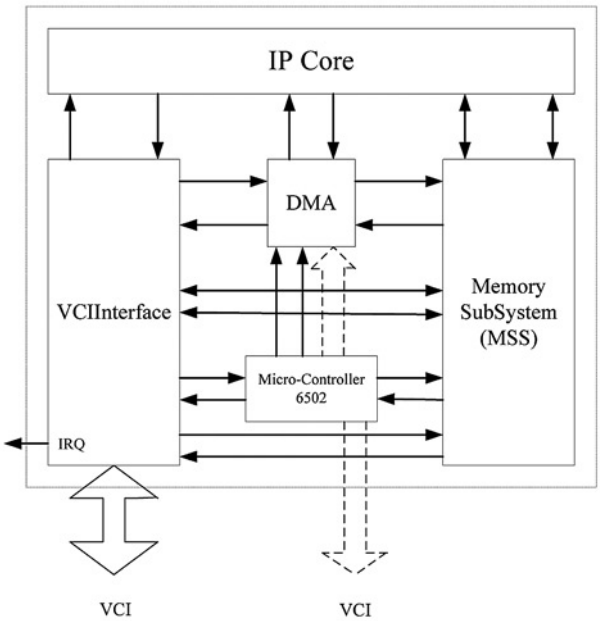

Fig. 1: Standardized IP Shell

task is not that simple as due to the high bandwidth of the signal coming from the A/D converters the data rate will be very high. This leads easily to a higher hardware complexity and a higher power consumption and thus results in a high number of DSP48E slices and accordingly to a cost intensive application. Therefore a generic design performing fractional up- and downsampling using the same underlying architecture is preferred. The whole design has to be configurable using a set of parameters that is handled by the SW control of the platform. The same SRC module shall be used for three different modes: only reception (RX), only transmission (TX) and reception and transmission at the same time. From the platform perspective, the different channels processed by the SRC are executed in parallel. Because of that is important that the SRC is able to process samples continuously, without loosing time when switching between two channels. To decrease the jitter one fixed master clock with low jitter is provided to the preprocessor. Thus the difference between the sampling rates has to be managed by the SRC itself. Performance has to be high and aliasing should be avoided.

3 Preprocessor Block

The preprocessor connects the external radio front end with the digital baseband processing platform. It has to ensure among others that the incoming and outgoing sample streams are modified in order to provide the data format and the sampling rate of the connected device as well as the sampling rate of the processed standard.

Apart from the interface with the external A/D and D/A converters, the preprocessor possesses a basic signal processing unit, a dual-port swing buffer, timing functions for framing and resynchronization and a sample synchronous interrupt generator. The signal processing unit is responsible for filtering, sample rate conversion and carrier frequency adjustment. Figure 2 gives an overview about the architecture of the preprocessor.

The incoming and outgoing samples are stored in FIFOs (interface A/D, D/A converters) and in the memory subsystem (interface entire platform). Up to 8 complex channels can be processed in parallel, each possessing a different set of parameters. The switch between two channels occurs after a fixed number of generated output samples and can be handled flexibly. The continuous processing of the preprocessor has to be guaranteed in order to to avoid a huge delay. This requires a control structure that stores / loads intermediate values into / from local RAMs. The necessary operations are performed in parallel to the main processing. The minimum number of samples processed per channel results from the time needed to perform a context switch.

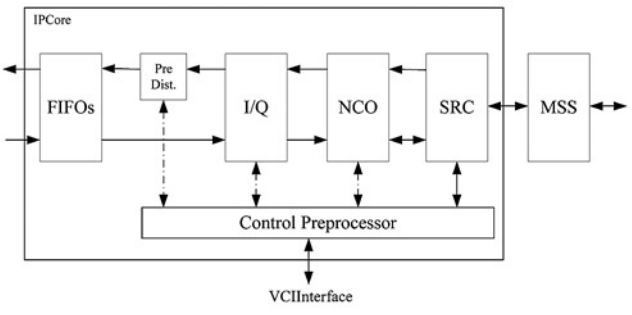

Fig. 2: Preprocessor Architecture

4 Sample Rate Conversion from a Hardware Perspective

4.1 Comparison of Existing Solutions

In the architecture of SDR systems, SRCs are one of the critical and most demanding elements [1]. Compared to the other blocks on the baseband processing platform, the SRC is the most DSP heavy block that requires around 50 % of the available DSP48E slices. In the past years, lots of different solutions for efficient SRC have already been presented, like for instance in [2], [5] or [9]. [12] has shown, that a solution based on polynomial interpolation in combination with CIC filters is the most efficient one in terms of performance. But the question in this context is, if the required space consumption justifies the obtained performance. For an efficient implementation three factors must be considered: the length of the lowpass filter that is used to avoid aliasing and thus the number of multiplications, the amount of computation involved in the calculation of each new filter coefficient and the control structure.

The simplest SRC solution is the sample-and-hold method. The analog signal is sampled and the obtained value is hold afterwards. This solution is easy to implement and does not need many resources. But the obtained performance is too low to use it in SDR applications. Alternatively one could think about first increasing the sampling rate by zero insertion and then to decimate it. No fine interpolation stage is necessary; all effective filtering can be done in one filter at a low sampling rate. Unfortunately this solution requires a high space consumption as interpolation and decimation stage are not merged and as the filter grows for higher decimation factors. Fixing the size of the filter would be no feasible solution as aliasing effects occur and as inherent redundancy is removed. A combination of a lowpass filter and A/D, D/A converters like presented in [10] is not appropriate for our design, either. This is due to the limited resources on the board and the choice of one single master clock to reduce the jitter and the phase noise respectively.

Remains the approach of interpolation in combination with the window method where an interpolation filter to calculate missing output samples or filter coefficients is combined with a lowpass filter to avoid aliasing ([8]). The performance depends on the number of stored filter coefficients and on the interpolation method. Using a higher order polynomial requires a higher computational effort and thus a higher space consumption. Calculating the filter coefficients in a recursive way would be an option. But to do so, the latency increases and further resources to calculate them are needed. To interpolate, different solutions are possible. The easiest solution is the nearest-neighbor interpolation. The corresponding hardware structure is quite simple but the efficiency is very low. Using linear interpolation leads to a better performance with low space consumption. Good filter characteristics are possible but the filter coefficients need to be precomputed and stored in memory. A generalization of linear interpolation is the polynomial interpolation, which possesses a high efficiency ([7]). Compared to linear interpolation, the calculation of the interpolating polynomial is computationally expensive. Another disadvantage is Runge's phenomenon, which shows that the interpolation polynomial may oscillate wildly between the data points for higher order ones. The most hardware consuming interpolation methods are the Spline interpolation which avoids Runge's phenomenon by using piecewise defined polynomials and the Whittaker-Shannon interpolation which is only working for infinite signals. As a mixture between linear interpolation and Whittaker-Shannon interpolation, [13] has introduced the concept of bandlimited interpolation. This method is easily

implementable and can highly be optimized by minimizing the required hardware space. The same algorithm can be used for upsampling and downsampling. The filter coefficients have to be precomputed and stored in the memory subsystem.

4.2 Presentation of the chosen Algorithm

The solution that has been chosen is the above mentioned bandlimited interpolation where the filter is a combination of a FIR lowpass filter and a linear interpolation filter. To obtain the result, the incoming samples are shifted under a windowed sinc function. This represents a linear combination of the signal samples or of the corresponding filter coefficients, respectively. The number of multiplications depends on the size of the lowpass filter and can be decreased by applying a polyphase filter structure. As it will be shown, the linear interpolation requires only three additional multiplications. For more detailed information please refer to [13].

Having a look at the downsampling case (similar derivation for the upsampling case), the sampling rate at the input, denoted $F_1 = \frac{1}{T_1}$, is decreased to a sampling rate $F_2 = \frac{1}{T_2}$. Having a sampled signal $x(nT_1)$ at the input of the SRC, its corresponding analog signal can be computed by

$$x(t) = \sum_n x(nT_1)g(t - nT_1) \tag{1}$$

where $g(t)$ is a sinc function multiplied with the Kaiser window. To get the same signal at a different sampling rate this analog signal is expressed from its samples with timing T_2, namely

$$x(kT_2) = \sum_n x(nT_1)g(kT_2 - nT_1) \tag{2}$$

T_1 and T_2have no specific relationship a priori. The sampling rate of the filter is T_3, so that the real filter function has to be expressed as $g(nT_3)$. For the downsampling case, the following term is valid: $T_1 = MT_3$ with M as the oversampling factor for the sampled representation of the basis waveform $g(t)$.

$$x(kT_2) = \sum_n x(nT_1)g(kT_2 - nMT_3) \tag{3}$$

As not all necessary filter coefficients are available right at the beginning a parameter k' has to be defined such that $k'T_3 \leq kT_2 < (k'+1)T_3$, or equivalently $k' = \left\lfloor k\frac{T_2}{T_3} \right\rfloor$. k' represents an existing filter coefficient while k represents a filter coefficient that has to be computed via linear interpolation. So (3) can be approximated as

$$x(kT_2) \approx \sum_n x(nT_1) \begin{bmatrix} (1-\alpha_k)g((k'-nM)T_3) \\ +\alpha_k g((k'-nM+1)T_3) \end{bmatrix} \tag{4}$$

with $\alpha_k = k\frac{T_2}{T_3} - k'$.

To optimize the filter structure and to minimize the necessary hardware resources it is useful to have a polyphase representation of (4). One first has to define

$$k' = k''M + l_k \tag{5}$$

where $l_k = k' \bmod M$ and

$$g_{l_k}(n) = g((nM + l_k)T_3), l_k = 0, 1, ..., M-1 \tag{6}$$

with $g_{l_k}(n)$as the polyphase representation of the prototype filter $g(t)$ and Mas the number of FIR filters processing in parallel. The approximation in (4) can now be written more simply as

$$x(kT_2) \approx \sum_n x(nT_1) \begin{bmatrix} (1-\alpha_k)g_{l_k}(k''-n) + \alpha_k g_{(l_k+1) \bmod M} \\ (k''-n+I(l_k = M-1)) \end{bmatrix} \tag{7}$$

with $I(.)$as the unit-valued indicator function.

5 SRC Architecture

A simplified overview of the resulting hardware structure can be seen in Figure 3.

The input samples are complex while the filter coefficients of the FIR filter are real. The filter coefficients are stored in the MSS and are first read in and then stored in local registers before processing. The connection between these coefficients and the FIR filters is established by the module *Select Coefficients*, which provides only the coefficients that are needed for the computation of the current output sample. The information about which filter to use is computed by the *Interpolation Control (IPC)* which also calculates the interpolation factors. All modules are working with the same clock. The difference between the sampling rate of the input samples and the sampling rate of the output samples has been realized using the module *Sample Computation* which signals when a sample has to be written in the *Registerbank* and when an output sample has to be computed.

Hardware resources are saved in different ways: It has been shown in the previous section, that for the computation of one output sample, two filters of the polyphase filter bank are needed. Thus, only four filters of the filter bank are implemented, two for the real and two for the imaginary part. Apart from that the filter coefficients are highly optimized. In Figure 4 some ideal filter coefficients and their distribution over a polyphase filter bank using four filters is shown. Per filter processing, two consecutive filters are used. Normally the input samples of the two filters are the same, except for the case when the fourth and the first filter are used. In this case, the input samples of filter one have to be right shifted by one sample. Considering an implementation in hardware this would introduce a complexity that can be avoided. Having a look at the filter coefficients one can see that the coefficients of filter one are composed of one 1 and zeros otherwise. Thus it is sufficient to change only the value of two filter coefficients to produce a shift of the coefficients to the left. These two values can be hard coded in the architecture. Furthermore a periodicity can be observed that reduces the number of coefficients that have to be stored in memory by almost 71 %.

8 complex channels can be processed at the same time (4 in RX and 4 in TX). For the purpose of not loosing any information when switching between them, the context of all SRC registers has to be stored in two local RAMs. To do so, two register sets are described. When a switch has to be performed, the old values are stored starting from a given address in the local RAM before the next channel to be processed is loaded.

Another critical issue is the handling of the different sampling frequencies. In the system of the SRC one can distinguish between three different time domains: the time distance between the input samples (T1), the time distance between the output samples (T2) and the time distance between the filter coefficients. The relation of these times depends on the mode. When upsampling, more output samples are computed than input samples exist while for

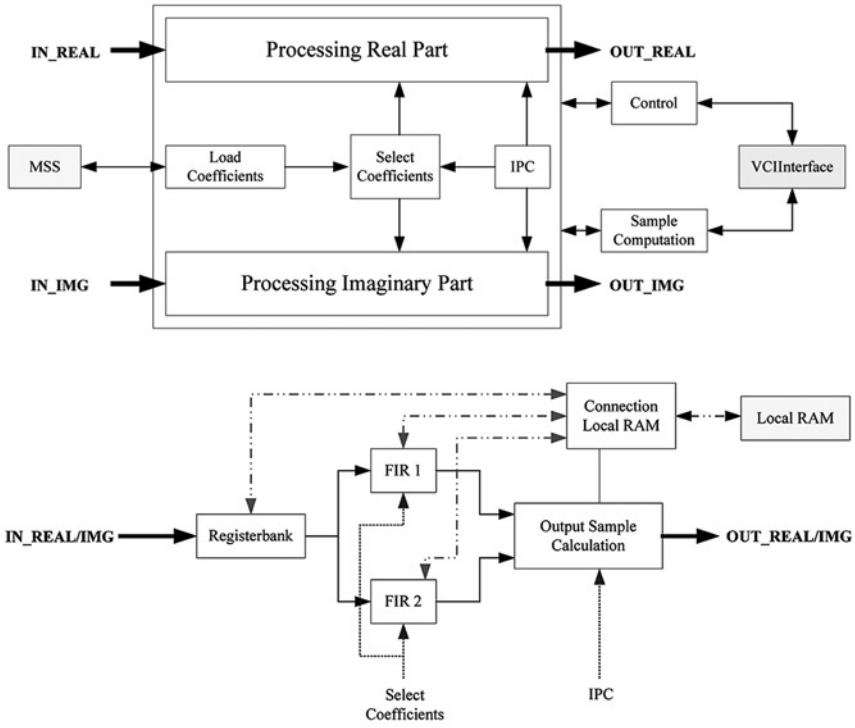

Fig. 3: Global Architecture of the SRC

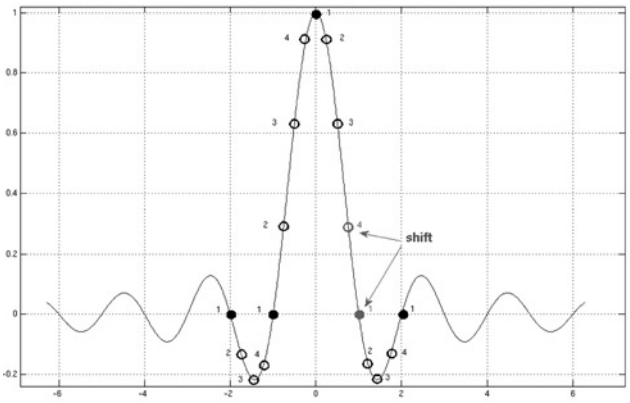

Fig. 4: Filter Coefficient Distribution for an Ideal Lowpass Filter

downsampling the relation is the other way around. All time values are represented using 70 bit, which results from the required 1Hz resolution of the sampling frequencies for the frequency range 3 MHz $\leq$ f $\leq$ 61.44 MHz and from the assumption that latest after 1000 computed output samples, T2 is a multiple of T1 (corresponds to a counter reinitialization in *Sample Computation*).

The maximum possible size of the filter is M = 8 filters in the polyphase filter bank, each with 19 filter coefficients a 16 bit. The bit size could be increased to 25 bit without increasing the number of DSP48E slices, but simulations have shown, that the performance gain is negligible. Possible degrees of freedom are the calculation of the filter coefficients or M, which could be easily increased to 16 or 32 bit. For all other parameters the space consumption would increase significantly.

Fig. 5: Channel Scheduling

5.1 Example: Processing of Four Channels Simultaneously

In this example four different channels are processed in parallel. The distribution could be 2 channels in TX and 2 channels in RX but one could also imagine 4 channels in TX or RX. The channels are processed one after another. If one of them is executed for the first time, the modules of the SRC have to be synchronized: first the filter coefficients are loaded and stored in local registers. Then the blocks *IPC* and *Registerbank* are enabled till they produce their first result. Afterwards the whole SRC is activated. In Figure 5 the scheduling is described. When for instance CH2 is executed, the context of the previous channel, CH1, is stored in a local RAM and the context of the next channel, CH3, is loaded. As soon as the load process is finished a channel switch can be performed and CH3 is executed without any delay.

6 Results

6.1 Performance Results

The maximum possible SINR is limited by the A/D and D/A converter performance. In TX / RX the resolution is 14 / 12 bit which corresponds to a maximum SINR of 86.04 dB / 74 dB [4]. Tests with different sinusoidal and sweep signals have shown that this performance can easily be reached with a filter size of 8 times 19 filter coefficients if no interpolation is needed. In case of interpolation, the SINR depends on the ratio between the sampling frequencies, the oversampling factor of the input signal and the input signal itself. For an

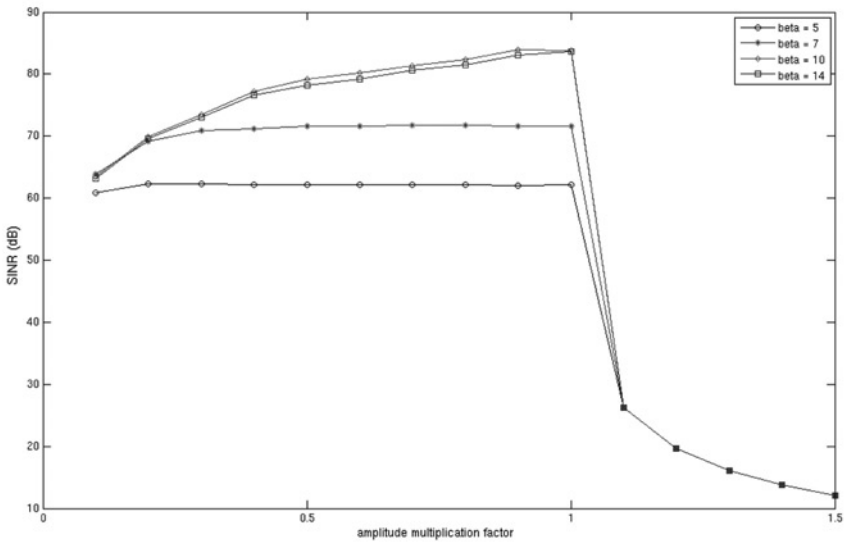

Fig. 6: SINR Performance for Changing Beta

exemplary sinusoidal signal defined as:
$y(x) = \frac{1}{4}\sin(x') + \sin\left(\frac{x'}{3}\right) + \sin\left(\frac{x'}{2}\right) + \cos(x')$
with $x' = 2\pi fx$ the following results are obtained:

Table 1:

Mode	factor	SINR
upsampling	1.45	80.12 dB
upsampling	2	86 dB
downsampling	4.3	73.8 dB
downsampling	5	74 dB

When a white Gaussian noise signal is used as input signal, the SINR can be obtained by evaluating the Power Spectral Density (PDS). Before processing the SRC, the input signal has to be lowpass filtered and oversampled. The SINR of the resulting signal has to be higher than the limiting factor by the A/D and D/A converters. Upsampling by a factor of 2.5 for instance results in a SINR of around 82 dB while the SINR is very close to the maximum if no interpolation is needed.

Apart from the number of filters in the polyphase filter bank the only adjustable parameter is the calculation of the filter coefficients. Different SINR values for the same basis waveform are obtained by changing the β parameter of the Kaiser window. Figure 6 shows how the SINR changes for different values of β. The input signal is a simple sinusoid.

6.2 Synthesis Results

In the following, synthesis results for a filter size of 8 x 19 filter coefficients are given. The target architecture is a Xilinx Virtex 5 LX330FF1760 (speed grade -2). For a reference clock frequency of 250MHz, the frequency of the SRC is 167.73 MHz. The results were obtained using the tool Precision Synthesis, Version 2009a.

Table 2:

Resource	Used	Avail	Utilization
Function Generators	31801	207350	15.34 %
CLB	7951	51840	15.34 %
DFFs	20017	209760	9.54 %
Block RAMs	4	288	1.39 %
DSP48Es	82	192	42.71 %

7 Extensions of the Current Design

The performance of the proposed SRC is the higher, the higher the sampling frequency of the incoming signal samples and the smaller the ratio between the sampling rates. For higher ratios, different extensions of the current design are possible. When upsampling, one common way is to add CIC filters to upsample the integer part before the fractional one [14]. Thanks to the proposed SRC the final size of the resulting architecture would still be suitable for the target technology. Alternatively the presented SRC could be processed several times, but the introduced delay would be too high so that the continuous processing of the incoming and outgoing samples is no longer guaranteed.

When downsampling one possible solution is a registerbank in which each register is addressable. Thanks to this structure a small program could be used to write the signal samples for the next processing into the registerbank. The latency of such a system would be smaller than the latency of a system where the samples are shifted one after another into the registerbank till the next output sample can be computed.

8 Conclusion

In this article we have proposed a simple reconfigurable architecture for a fractional sample rate converter, that is able to deal with almost all existing wireless communication standards. An analysis of existing solution has been accomplished from a hardware point of view. Our main concern was to find the algorithm with the highest performance but given a limited space consumption. As a result, a solution based on bandlimited interpolation in combination with Whittaker-Shannon interpolation has been presented.

The obtained algorithm performs resampling at a frequency of 167.73 MHz and supports a 1Hz resolution of the sampling frequencies. Furthermore possible extensions to the current filter design have been presented. Determining in detail their performance differences is part of our future work.

References

[1] Wajih A. Abu-Al-Saud, Gordon L. Stueber, "Efficient Sample Rate Conversion for Software Radio Systems," IEEE Transactions of Signal Processing, Vol 54, No. 3, March 2006.

[2] Ronald E. Crochiere, Lawrence R. Rabiner, "Interpolation and Decimation of Digital Signals," Proceedings of the IEEE, Vol. 69, No.3, March 1981.

[3] Najam-ul-Islam Muhammad, Rizwan Rasheed, Renaud Pacalet, Raymond Knopp and Karim Khalfallah, "Flexible Baseband Architecture for Future Wireless Systems", 11th EUROMICRO Conference on Digital System Design Architectures, 2008.

[4] Allen Gersho, Robert M. Gray, "Vector quantization and signal compression", Springer International Series in Engineering and Computer Science.

[5] Tim Hentschel and Gerhard Fettweis, "Sample Rate Conversion for Software Radio", IEEE Communications Magazine, 38(8):142 - 150, August 2000.

[6] J.Mitola, "The Software Radio Architecture", IEEE Communications Magazine, vol. 33, no 5, May 1995, pp.26 - 38.

[7] Alan V. Oppenheimer, Ronald W. Schaefer, John R. Ruck, "Discrete-Time Signal Processing", Second Edition, Prentice Hall.

[8] John G. Proakis, Dimitris G. Manolakis, "Digital Signal Processing; Principles, Algorithms and Applications", Fourth Edition, Prentice Hall.

[9] Tor A. Ramstad, "Sample-Rate conversion for arbitrary ratios", IEEE International Conference on Acoustics, Speech and Signal Processing, 1982.

[10] Tapio Saramäki, Tapani Ritoniemi, "An efficient Approach for Conversion between arbitrary Sampling Frequencies", IEEE International Symposium on Circuits and Systems, 1996, ISACS '96.

[11] Ali Shahed hagh ghadam, Markku Renfors, "Farrow Structure Interpolators Based on Even Order Shaped Lagrange Polynomials", Proceedings of the 3rd International Symposium on Image and Signals Processing and Analysis, 2003.

[12] Faheem Sheikh, Shahid Masud, "Sample rate conversion filter design for multi-standard software radios", Digital Signal Processing, doi:10.1016/j.dsp.2009.04.014, 2009.

[13] Julius O. Smith, Phil Gossett, "A Flexible Sampling-Rate Conversion Method", IEEE International Conference on ICASSP, 1984.

[14] Tianqi Wang, Cheng Li, "Sample Rate Conversion Technology in Software Defined Radio", Canadian Conference on Electrical and Computer Engineering, 2006. CCECE '06.

First Author
Carina Schmidt-Knorreck
Mobile Communications Department, Eurecom
2229 route des Crêtes, F-06904 Sophia-Antipolis Cedex – France
E-mail: Carina.Knorreck@eurecom.fr

Other authors
Raymond Knopp
Mobile Communications Department, Eurecom
2229 route des Crêtes, F-06904 Sophia-Antipolis Cedex – France
E-mail: Raymond.Knopp@eurecom.fr

Renaud Pacalet
Laboratoire SoC, Telecom ParisTech
2229 route des Crêtes, F-06904 Sophia-Antipolis Cedex – France
E-mail: Renaud.Pacalet@TELECOM-ParisTech.fr

* Eurecom's Research is supported in part by its industrial partners: Swisscom, Thales, SFR, Orange France, Hitachi Europe, STMicroelectronics, Bouygues Telecom, Sharp, Cisco, BMW Group. The research work leading to this paper has also been partially supported by the European Commission under the ICT research network of excellence NEWCOM++ and by the project PLATA.