Noname manuscript No.
(will be inserted by the editor)

Efficient Parallelization of Polyphase Arbitrary Resampling FIR Filters for High-Speed Applications

Hannes Ramon · Haolin Li · Piet Demeester · Johan Bauwelinck · Guy Torfs

Received: date / Accepted: date

Abstract This article describes a method for increasing the sampling rate of efficient polyphase arbitrary resampling FIR filters. An FPGA proof of concept prototype of this architecture has been implemented in a Xilinx Kintex-7 FPGA which is able to convert the sampling rate of a signal from 500 MHz to 600 MHz. This article compares this new architecture with other best known efficient resampling architectures implemented on the same FPGA. The area usage on the FPGA shows that our proposed implementation is very proficient in high bandwidth applications without requiring significantly more resources on the FPGA. A theoretical calculation of the resampling error introduced on a modulated data stream is provided to evaluate the new architecture against other existing resampling architectures.

Keywords FIR · FPGA · polyphase filter · resampling · broadband

1 Introduction

In wireless communication, there is often the need to resample the data from one sampling frequency to another. One particular case is to perform digital pulse shaping with a FIR filter on the data symbols. Theoretically, this pulse shaping filter only needs to be sampled at the original symbol rate, but a higher ratio between the pulse shaping sampling frequency and the symbol rate, leads to lower requirements on the digital-to-analog conversion, especially on the analog front-end filtering of the images. When the symbol rate is already quite low, it is easier to obtain a larger upsampling factor and even perform integer resampling. However, in case of broadband signaling, the maximum sampling rate is often limited by the capabilities of the digital-to-analog converter (DAC).

With the rise of new wireless communication standards like WiGig [2], the bandwidths and symbol rates have drastically increased. These bandwidths come very close to the maximum sampling speed of commercial high-speed DACs, which means that integer upsampling is no longer possible without compromising the cost and power consumption of the DAC. Fractional resampling on the other hand is a lot more complex and is typically less hardware efficient. Research for efficient low symbol rate fractional resampling has been done in [1,3,6]. All these architectures are based on the polyphase representation of upsampling and resampling FIR filters. Reference [3] provides an FPGA implementation of a sampling rate conversion filter, but the maximum achievable output frequency stays below 300 MHz. This is hardly enough for broadband applications since e.g. WiGig provides channel bandwidths larger than 3 GHz.

This work provides an architecture which allows increasing the frequency and hence bandwidth of resampling filters while maintaining reasonable hardware efficiency. A prototype of 500 MHz to 600 MHz sampling rate conversion FIR filter has been implemented in a Xilinx Kintex-7 FPGA as proof of concept. For this implementation no optimizations depending on the Kintex-7 architecture are performed except for the general applicable speed improvements proposed in this work. This FPGA is, without optimization or dedicated structures, unable to run a resampling FIR filter with a frequency higher than 300 MHz due to the speed limitations of the DSP (multiplier) blocks.

This paper is structured as follows: Section 2 gives a brief introduction to polyphase resampling FIR filters where the standard architecture and optimizations are explained.

This work was supported by the iMinds IoT research program

H. Ramon, H. Li , P. Demeester, J. Bauwelinck, G. Torfs
are with the Dep. of Information Technology (INTEC), Ghent University - iMinds - imec, Technologiepark-Zwijnaarde 15 (iGent), B-9052 Gent, Belgium.
E-mail: hannes.ramon@ugent.be

Next, the influence of imperfect resampling on the data symbols in digital communication systems is described. The proposed speed improvements are discussed in section 4. Finally, section 5 presents the proof of concept on FPGA.

2 Polyphase resampling

2.1 A parallel FIR filter polyphase architecture

Fractional resampling involves 3 sampling frequencies: the starting frequency (F_s), the target frequency (F_t) and the intermediate frequency (F_i) given by the least common multiple of F_s and F_t. This yields a fractional resampling factor of $f = \frac{F_t}{F_s} \in \mathbb{Q}$. Remark that for an integer upsampling ratio $F_i = F_t$. First an upsampling step to F_i is performed (by padding 0's in between the original samples) followed by a low pass filter FIR filter at this frequency. Any low pass filter with the correct cut-off frequency below F_t that meets the system requirements can be chosen. However in telecommunication resampling is often performed together with pulse shaping, hence, the pulse shaping filter can be reused. Finally a decimation step converts the data at F_i to F_t. However, this intermediate sampling frequency F_i is often too high (multi-GHz) for practical implementation in FPGA or even ASIC designs. To illustrate this, assume digital data with a bandwidth of 500 MHz (F_s) and a required oversampling of $f = 1.2$, then $F_t = f \cdot F_s = 600$ MHz which yields $F_i = 3$ GHz. 3 GHz is without doubt too fast for an FPGA implementation. Implementing such high clock frequencies in ASIC would be hard and the success depends on the used technology. This problem is solved by noting that most of the operations in the FIR filter are multiply-by-zero followed by add-with-zero steps. Due to these zero operations, the upsampling process can avoid the filtering at F_i by splitting the filter in $N = F_i/F_s$ parallel filters at F_s [5]. The N filters perform their calculations on the same data but operate each at a different decimated version of the original filter at F_i. A parallel-to-serial operation on the outputs of these N parallel FIR filters reconstructs the data at the intermediate frequency. In the remainder of this article, the N parallel filters are called $filter_0$ to $filter_{N-1}$.

In the case of fractional resampling, it is not required to reconstruct the data at the intermediate frequency, but only the outputs of the output streams after decimation at F_t are of interest. This involves a demultiplexer and a state machine (Fig. 1) calculating the next selected stream ($s[k+1]$) with respect to $s[k]$ according to the following rule:

$$s[k+1] = (s[k] + D) \bmod N \text{ ; with } D = \frac{F_i}{F_t} \tag{1}$$

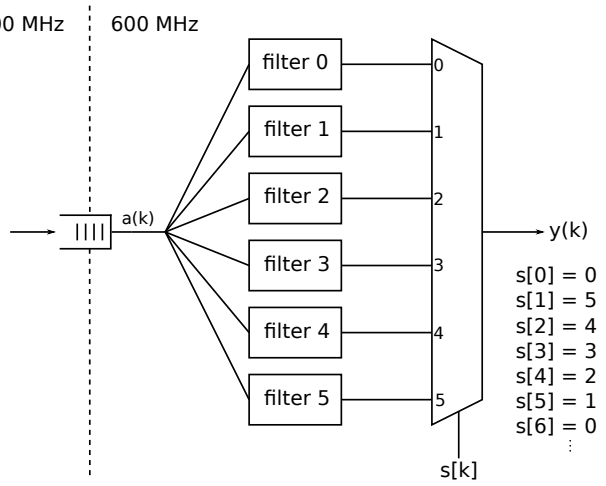


Fig. 1 Parallel polyphase resampling topology with multiplexer ($F_s = 500$ MHz, $F_t = 600$ MHz).

2.2 Hardware efficient polyphase filtering

This parallel implementation is, however, not hardware efficient as it often requires a large number of FIR filter taps (due to the high F_i). As a result of the direct decimation on the N parallel streams, only one of the N FIR filters produces a useful sample each time an output is taken. The other $N-1$ outputs are discarded and hence these tapped delay lines are wasting energy and area. By making only one tapped delay line for the FIR filter and switching the filter coefficients instead of switching the outputs, only one filter, instead of N filters, is required. The filter index $s[k]$ is then e.g. applied to a lookup table (LUT) or ROM device as address for reading the filter coefficients. Several possibilities to calculate the filter index $s[k]$ are available. The Barker structure [1] provides a general implementation allowing the use of an arbitrary F_i which makes it very powerful. Another possibility presented in this work implements the state machine of Eq. (1).

2.2.1 The Barker structure

The index calculation in [1] is performed by a modulo C accumulator followed by a floor function as displayed in Fig. 2. When the accumulator overflows, a new input sample is loaded in the shift register of the FIR filter. ϕ_Δ is given in Eq. (2) and C is an arbitrary constant. The choice of C influences the error made on the fractional interpolation. A higher C results in a better approximation of the interpolation factor f.

$$\phi_\Delta = \frac{C}{f} = \frac{CF_s}{F_t} \tag{2}$$

ϕ_Δ is in fixed point notation and hence more precise calculation leads to more resources used and harder tim-

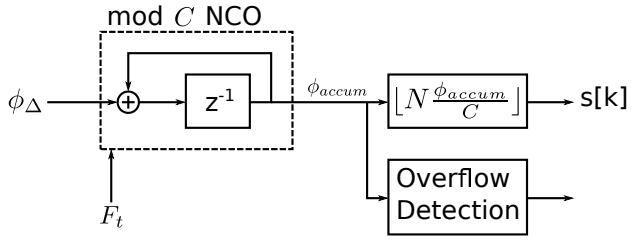


Fig. 2 The filter index $s[k]$ generator from [1] (Barker structure).

ing constraints to be met. Also, as seen in Fig. 2, a floor function is performed after the multiplication and division operations. All these operators introduce errors, especially the floor function. This function is necessary because the index is an integer. Due to this architecture, the downsampling from F_i to F_t can be temporarily off, but on average the resampling is correct. These local anomalies in the resampling causes distortion in the signal. The importance of this distortion depends on the application of the filter. However, these local anomalies can have devastating effects on the data in telecommunication applications which is discussed in section 3.

The multiplication by N and division by C can be implemented as a shift operation to improve the speed of the circuit. Furthermore, the mod C operation can be more efficiently calculated when C is a power of 2. This leads to less flexibility in N and C (as they both have to be a power of 2) and possibly introduces distortion as mentioned earlier.

Reference [3] provides an FPGA based implementation of such an efficient implementation of the architecture published in [1], but is limited to 300 MHz. This is too low when anticipating for the rise of broadband communication systems. Our proposed implementation on the other hand allows for a speed improvement of at least a factor of 2 as discussed in section 4.

2.2.2 State machine implementation

Compared to the Barker structure, a finite state machine implementation (FSM) is exact as it follows Eq. (1). Moreover, the FSM can be implemented without multiplications and additions leading to a very fast index generation circuit.

An issue with the state machine implementation arises when $D < N$ or equivalently if $f > 1$. To understand what happens in this case, the example shown in Fig. 1 is first analyzed. Remember that the input of the multiplexer consists of parallel samples running at F_i and are all simultaneously calculated with the same input data. Now, if $s[k] = 0$ then $s[k+1] = 5$ and as a result, the output signal, which is a decimated version of F_i, will consists of the samples of $filter_0$ followed by those of $filter_5$. However, both output samples are generated using the same input data. Making this more general: if in Eq. (1) $s[k+1] > s[k]$ ($s[k] + D < N$) both output samples at time step k and $k+1$ are calculated using the same data, which means the delay line is not allowed to shift or change data between step k and $k+1$. Hence, all the shift registers need to be stalled for one clock cycle. This issue makes the design more complicated but will be solved in section 4 while discussing the speed improvement techniques on the resampling architecture. In the Barker structure previously discussed, this is solved using the overflow detection.

The exact and fast characteristics of the FSM implementation are the reasons for taking this approach in this work. In fact, this FSM can be implemented as a simple lookup table (LUT).

3 Influence of resampling error in telecommunication applications

Section 2.2 explained that for high speed applications, the Barker structure needs to be implemented with a bit shift operation. By using the shift operator for the division, higher clock rates can be used, but at the cost of losing full control over the values C, N and ϕ_Δ. C and N can only be a power of 2, which is not necessary a problem for C as it can be arbitrarily chosen, but N determines the intermediate frequency. When this intermediate frequency is not equal to the common multiple of F_s and F_t, the downsampling from F_i to F_t will be fractional. In the Barker structure this results in a decimation factor D that changes over time, but having an average value $E[D] = F_i/F_t$. In normal resampling applications, this error does not pose a problem. However, when using this resampling topology in telecommunication applications where matched filters are of great importance, the non-ideal downsampling with a variable D causes intersymbol interference (ISI). This section provides a calculation of this ISI due to the non-ideal resampling of the Barker structure and evaluates this theoretical result with a simulation.

3.1 Theoretical ISI calculation

In order to fully capture all the effects of the resampling error on the ISI, the data symbols $a(k)$ are resampled from their symbol period $T_s = 1/F_s$ to $T_t = 1/F_t$ by using the intermediate sampling period $T_i = 1/F_i$. T_i is not necessarily a common divisor of T_s and T_t. The transmit pulse is represented as the continuous time $p_t(t)$ function.
Eq. (3) shows the upsampled signal at F_i [4].

$$y_{T_i}(m) = \sum_{k=0}^{\infty} a(k) p_t(mT_i - kT_s) \tag{3}$$

In order to obtain the wanted signal sampled at F_t, $y_{T_i}(m)$ needs to be downsampled. Downsampling with a constant

factor D is straightforward and results in selecting the samples at $m = nD$. However in this case the downsampling factor varies in time, and is represented at each sampling moment by $d(n)$. The resulting signal is:

$$y_{T_t}(n) = y_{T_i}\left(\sum_{n'=0}^{n} d(n')\right) = \sum_{k=0}^{\infty} a(k) p_t\left(\sum_{n'=0}^{n} d(n')T_i - kT_s\right) \quad (4)$$

The samples $y_{T_t}(n)$ are transmitted. To study the effect of the imperfect decimation of the transmit stream on ISI, the data is recovered using an ideal matched filter $p_t^*(-t)$. To remove additional sampling effects at the receiver, the sampling period at the receiver is chosen to be a common divisor of T_s, T_t and T_i. This sampling period is represented as T_s/L. The upsampling from T_t to T_s/L is assumed to be ideal in order not to introduce additional ISI. The low-pass characteristic of $p_t^*(-t)$ is reused to interpolate during the upsampling process. The resulting samples are given in Eq. (5).

$$y_{\frac{T_s}{L}}(m) = \sum_{k=0}^{\infty} a(k) T_t \sum_{n=0}^{\infty} p_t\left(\sum_{n'=0}^{n} d(n')T_i - kT_s\right) \cdot p_t^*\left(nT_t - m\frac{Ts}{L}\right) \quad (5)$$

The extra factor T_t is a by-product keeping the energy of the signal constant after the upsampling to T_s/L. The received symbols $a'(k')$ are found by decimating Eq. (5) by a constant factor L:

$$a'(k') = \sum_{k=0}^{\infty} a(k) T_t \sum_{n=0}^{\infty} p_t\left(\sum_{n'=0}^{n} d(n')T_i - kT_s\right) \cdot p_t^*\left(nT_t - k'T_s\right) \quad (6)$$

In real digital communication systems, the transmit pulse p_t is sampled in order to do the pulse shaping in the digital domain. Eq. (6) can hence be rewritten to Eq. (7) where $p(l)$ is $p_t(t)$ sampled with a sampling period T_s/L.

$$a'(k') = \sum_{k=0}^{\infty} a(k) T_t \sum_{n=0}^{\infty} p\left(\sum_{n'=0}^{n} d(n')\frac{L}{N} - kL\right) \cdot p^*\left(nL\frac{T_t}{T_s} - k'L\right) \quad (7)$$

Because T_s/L is a common divisor of all the sampling periods, $L\frac{T_t}{T_s}$ is an integer and hence Eq. (7) is a discrete formula. There are several ways to evaluate the ISI from Eq. (7): through the Error Vector Magnitude (EVM) defined in Eq. (8) or by looking at $g(k'-k)$ defined in Eq. (9).

$$\text{EVM} = 10\log_{10}\frac{\sum_{k=0}^{\infty}|a'(k)-a(k)|^2}{\sum_{k=0}^{\infty}|a(k)|^2} \quad (8)$$

$$a'(k') = \sum_{k=0}^{\infty} a(k) g(k'-k) \quad (9)$$

EVM is the error power divided by the power of the transmitted symbols expressed in dB. The EVM of a perfect communication system is $-\infty$ dB and a larger EVM leads to a more erroneous system, in this case originating from more ISI. The other method for evaluating the ISI of a system is by evaluating $g(k-k')$. $a'(k')$ is the convolution of $a(k)$ and $g(l)$. The latter can be identified from Eq. (7) in combination with Eq. (9). For an ideal communication system $g(k-k') = \delta(k-k')$, which leads to $a'(k) = a(k)$. Eq. (10) is the identified $g(k'-k)$.

$$g(k'-k) = T_t \sum_{n=0}^{\infty} p\left(\sum_{n'=0}^{n} d(n')\frac{L}{N} - kL\right) \cdot p^*\left(nL\frac{T_t}{T_s} - k'L\right) \quad (10)$$

$g(0)$ should hence be 1 and $g(l \neq 0) = 0$ and more ISI leads to larger differences in these two equalities. Eq. (8) in combination with Eq. (7) is not very practical since the transmitted data is still needed. A good estimation of the real EVM is obtained by replacing $a(k)$ with $\delta(k)$, which is the equivalent of determining the impulse response $g(l)$. The EVM then only depends on $g(l)$ and can easily be calculated.

$$\text{EVM} \approx 10\log_{10}\left(\sum_{k=0}^{\infty}|g(k)-\delta(k)|^2\right) \quad (11)$$

To quantify the effect of the fractional resampling error on ISI, a 500 MHz to 600 MHz conversion with $p_t(t)$ the root-raised cosine function (roll-off 0.1) is performed. First with the ideal $F_i = 3$ GHz ($N = 6$ and $d(n) = D = 5$), which will be used as reference, and then with $F_i = 4$ GHz ($N = 8$ and $E_n[d(n)] = D = 6 + 2/3$). The decimation factors $d(n)$ resulting from the Barker structure are a repetition of [7 6 7 7 6 7]. The calculated EVMs using Eq. (11) are presented in Table 1.

A large number of taps has been used during the calculation of the results in Table 1 to reduce truncation effects of the raised cosine filter. It is clear that resampling using an arbitrary intermediate frequency can cause some severe problems in telecommunication systems. For a 4-QAM system,

	$F_i = 3$ GHz	$F_i = 4$ GHz
EVM	-207.24	-23.16

Table 1 Calculated EVM for a 500 to 600 MHz resampling filter with $F_i = 3$ GHz and $F_i = 4$ GHz.

-23 dB EVM might be good enough, but for e.g. a 64-QAM system, -23 dB EVM leaves almost no room for noise and other non-ideal phenomena in the system. The next section confirms these results with a full system simulation.

3.2 ISI Simulation results

In order to evaluate the correctness of Eq. (11) a complete simulation of the examples from Table 1 has been performed. The Barker structure has been implemented and used for the ideal and the non-ideal resampling case. Table 2 summarizes the configuration used for the Barker structure in each case.

	$F_i = 3$ GHz	$F_i = 4$ GHz
N	6	8
D	5	-
C	6	1024
ϕ_Δ	5	853.33

Table 2 Configuration for the simulation of the 500 MHz to 600 MHz resampling filter for respectively $F_i = 3$ GHz and $F_i = 4$ GHz.

The simulated transmitter generates random data, maps it to a 64-QAM constellation and filters the symbols with the resampling filters. The spectrum of the signal after the resampling filtering is given in Figures 3 and 4. The ISI is already visible on the spectral plot for $F_i = 4$ GHz. The spectrum of this signal is more noisy and the spectrum beyond 250 MHz is only 20 dB below the wanted signal. The spectrum for $F_i = 3$ GHz is as expected very clean.

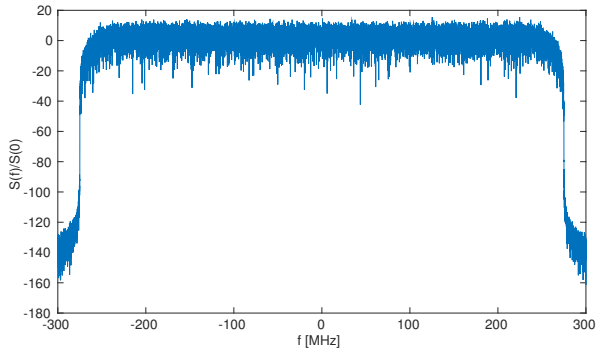


Fig. 3 Simulated spectrum of the data after the 500 MHz to 600 MHz resampling filter for $F_i = 3$ GHz.

This signal is then resampled with a matched receive filter as described in section 3.1 and demodulated with an ideal

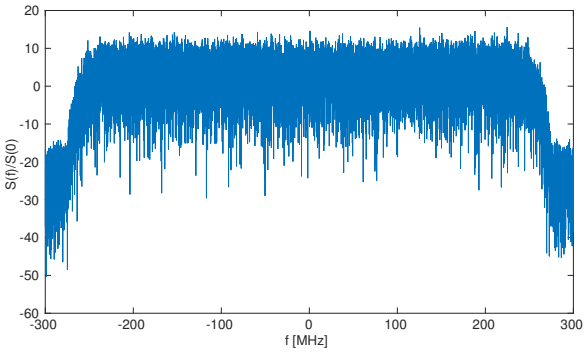


Fig. 4 Simulated spectrum of the data after the 500 MHz to 600 MHz resampling filter for $F_i = 4$ GHz.

receiver. The results of the demodulation are given in Fig. 5 with the corresponding EVMs in Table 3. The EVM results are very similar to the calculated EVM in Table 1. Fig. 5 shows that -23 dB EVM is not sufficient to realize error-less transmission using 64-QAM as little added noise will drastically increase the bit error rate (BER).

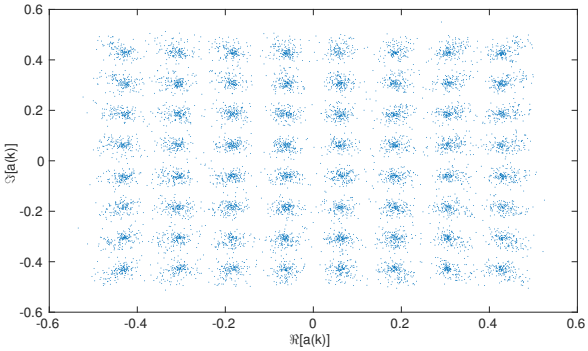


Fig. 5 Simulated 64-QAM scatter diagram of the data after demodulation in the ideal receiver for non-ideal $F_i = 4$ GHz.

	$F_i = 3$ GHz	$F_i = 4$ GHz
EVM	-152.95	-22.61

Table 3 Simulated EVM with real random data for 64-QAM for a 500 to 600 MHz resampling filter with $F_i = 3$ GHz and $F_i = 4$ GHz.

The simulation results together with the theoretical calculations prove that for telecommunication systems, ideal resampling, using a common multiple of start and end sampling frequency, needs to be performed in order to not introduce extra distortion which lowers the system performance.

4 Proposed speed improvements

In FIR filter design, the multipliers are often the largest bottleneck when increasing the clock frequency. In FIR filter structures where the coefficients do not change each clock cycle, this problem can efficiently be solved by designing multiply-by-constant multipliers. However, in polyphase resampling FIR architectures, this is not possible without consuming a lot of area, hence other more efficient solutions to increase the sampling speed are required.

This work solves this speed problem by reintroducing the parallel tapped delay line, but each tapped delay line only runs at half the output frequency F_t. If a larger reduction in frequency is needed, it is straightforward to increase the degree of parallelism, but a larger degree of parallelism also reduces the benefits obtained by combining the N parallel filters. Hence, only a parallelization of 2 will be discussed.

4.1 Two parallel tapped delay lines

To reduce the clocking frequency, the N filters in Fig. 1 are divided over two tapped delay lines $DL1$ and $DL2$. The two delay lines are clocked at half the output rate ($F_t/2$). The outputs of the two delay lines are combined with a parallel-to-serial converter to the full rate F_t where the output samples with even k are produced by $DL1$ and with odd k are $DL2$ (Fig. 6). Let us revisit Eq. (1) to see which filter coefficients go into which tapped delay line. Starting from $k = 0$ and $s[0] = 0$, the filter indices produced by an even k go into $DL1$ and the filter indices with an odd k into $DL2$. To illustrate, if $F_s = 500$ MHz and $F_t = 600$ MHz, then $N = 6$ and $D = 5$. $DL1$ contains (in order of execution) $filter_0$, $filter_4$ and $filter_2$, while $DL2$ contains $filter_5$, $filter_3$ and $filter_1$. So $filter_{s[k]}$ and $filter_{s[k+1]}$ (k even) are active at the same time. Hence, typically $DL2$ runs one sample ahead of $DL1$ (or equivalently $DL2$ is a by one sample shifted version of $DL1$) and a switching block that properly routes the data to $DL1$ and $DL2$ is needed. To illustrate, when filtering the datastream $\{0,\ldots,$ A, B, C, D, $0,\ldots, 0\}$, the content in $DL1$ is e.g. $\{0, 0,$ A, B, C$\}$ and the content in $DL2$ is $\{0,$ A, B, C, D$\}$. However, an exception occurs when $s[k+1] > s[k]$. This event implies, as mentioned earlier, that the content in $DL1$ and $DL2$ is the same. Returning to the above illustration this would e.g. mean that $DL1$ and $DL2$ both contain $\{$A, B, C, D, $0\}$.

4.2 Hardware reduction by virtually combining delay lines

The system in Fig. 6 still uses 2 tapped delay lines. To reduce the number of registers needed in the 2 delay lines, $DL1$ can be cleverly reused for $DL2$. By adding an extra register in front of $DL1$ and starting $DL2$ one register before $DL1$, $DL2$ is always one clock cycle ahead of $DL1$ which is required for the parallel configuration. Extra multiplexers are needed to give $DL2$ the possibility to have the same data when $s[k+1] > s[k]$. Fig. 7 illustrates this single delay line architecture for speed improvements. This change loses the need of the complex data switching block needed in Fig. 6. By virtually combining $DL1$ and $DL2$ in one physical delay line a large reduction of registers can be accomplished even when using two or more parallel lines. In the remainder of this article, this physical delay line virtually combining $DL1$ and $DL2$ will be referred as PDL which stands for physical delay line.

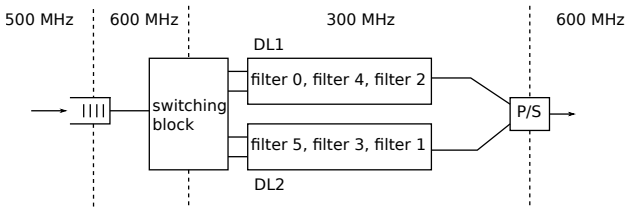


Fig. 6 Resampling FIR filter topology with two parallel lower rate filter streams with $F_s = 500$ MHz and $F_t = 600$ MHz.

As displayed on Fig. 7 the PDL requires 2 inputs (*input1* and *input2*) as it shifts by two instead of one place. This is because $DL1$ and $DL2$ only run at half the output rate, but still accept the data at the full rate. In order to fulfil this, two data samples need to be applied to PDL each clock cycle (at $F_t/2$). *Input1* holds the current input sample and *input2* holds the input sample before the sample applied at *input1* However, when $s[k+1] > s[k]$, the multiplexers make sure that the same data is routed to the $DL1$ and $DL2$ multiply-adder block. The clock cycle after this event, the virtual delay line $DL2$ needs to shift by two while $DL1$ may only shift one place. This special action makes sure that $DL2$ is again one sample ahead of $DL1$ without losing data. The solution is simple. In this case, PDL shifts only one place so $DL1$ shifts one. The shift-by-two for $DL2$ is automatically accomplished due to the shift-by-one of PDL and the multiplexers returning to the initial state. Fig. 8 shows a more detailed version of the the shift-by-one-or-two shift register from Fig. 7.

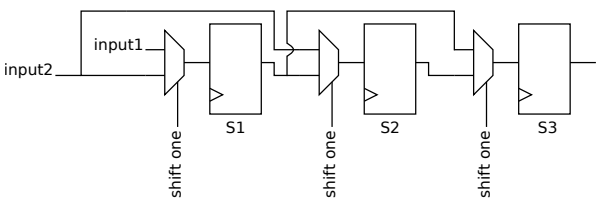


Fig. 8 The shift-by-one-or-two shift register. When the *shift one* control signal is high, the shift register shifts one place. The register shifts two places otherwise.

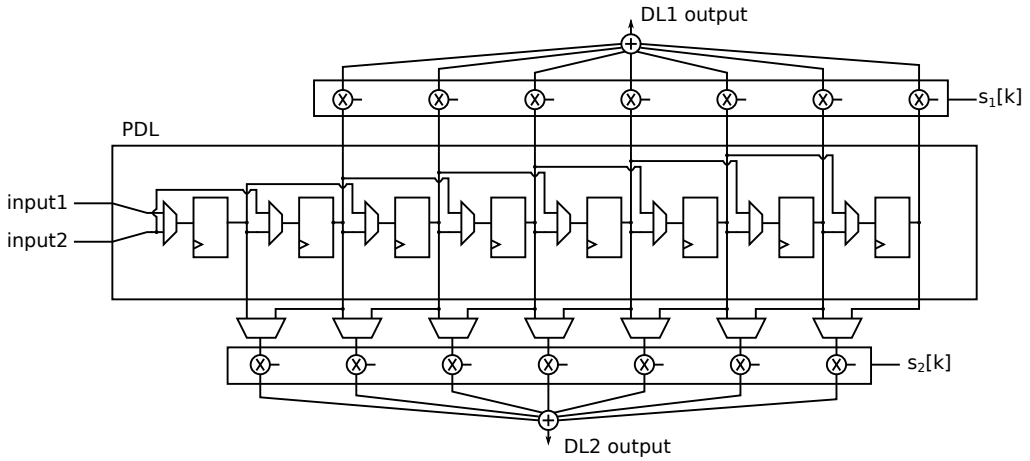


Fig. 7 The single delay line architecture for speed improvements.

4.3 Filter coefficient index generation

In order to meet the speed requirements and to not introduce extra ISI, a state machine implementation for the filter index $s[k]$ calculator has been chosen. However, this generator needs to produce (in the 2 parallel delay line case) two filter states: $s_1[k]$ for $DL1$ and $s_2[k]$ for $DL2$. A simple change to Eq. (1) needs to be made and the new formula is presented in Eq. (12).

$$\begin{aligned} s_i[k+1] &= (s_i[k] + 2D) \bmod N \; ; i = \{1, 2\} \\ s_1[0] &= 0 \\ s_2[0] &= D \end{aligned} \tag{12}$$

The difference is that 2 states are calculated in parallel and they jump with $2D$ instead of D. $s_i[k]$ in Eq. (12) is now calculated at $F_t/2$.

The state changes in Eq. (12) must be implemented as an FSM to enable high speed applications. The needed states and the transitions follow from Eq. (12), which is a cyclic FSM with a repeatable pattern.

Until now it was assumed that N is dividable by the degree of parallelism. However, when it is not dividable, this is not a problem as Eq. (12) remains valid. It will only take longer to reach the repeatable pattern for the FSM. In case of even N, the length of the repeated pattern is $N/2$. If N is odd, the length of this pattern will be N.

If the speed reduction due to the parallelization is large enough, Eq. (12) can be implemented directly with the $+$ and $\bmod$ operator in hardware instead of the FSM implementation which makes the implementation more flexible, but slower.

5 FPGA implementation and results

As a proof of concept, a 500 to 600 MHz resampling FIR filter (also viewable in the examples throughout this article) following the above described architecture has been implemented in a Xilinx Kintex-7 FPGA. This FPGA provides a DSP block for multiplication, but this block can only go up to 300 MHz without explicitly invoking these blocks and taking special precautions in mind. The goal of this design was to not explicitly invoke the DSP blocks but let the synthesiser implicitly choose them and hence only a maximal clocking frequency of 300 MHz was achievable. By choosing this approach, the full design of the filter is not limited to one FPGA or FPGA-family and hence it is easier to port to other FPGA or ASIC designs in the future.

With this design, we were able to generate a signal with 500 MHz bandwidth which was sent to a DAC running at 600 MHz to get an oversampling ratio of 1.2. In order to compare our design and architecture with the other architectures available, the Barker structure with the multiplier (BM) for coefficient index generation and the Barker shift register (BS) implementation were implemented. The Barker shift register implementation uses $N = 8$, which means that F_i is not optimal, while the presented and the Barker multiplier implementation use the exact F_i needed. Table 4 compares the three architectures in speed, resources and distortion. All the architectures in Table 4 implement a $f = 1.2$ resampling filter, but due to the speed limitations, the BS and BM use a lower clock frequency. All three implementations use 16 bits for the data and filter coefficients. The filter coefficients are stored in standard LUTs.

This work explicitly uses more area in order to overcome the speed limitations. However, Table 4 shows that the extra resources required are very reasonable. As expected, twice as much DSP blocks are needed for the multiplication, but the needed number of registers is only a factor 1.3 higher. The number of needed LUTs is a factor 2.2 higher. This

	Barker multiplier (BM)	Barker shift register (BS)	This work
LUTs	460	416	1032
Registers	1132	1245	1590
DSP blocks	21	21	42
Max speed	200 MHz	300 MHz	>600 MHz
ISI	none	high	none

Table 4 Comparison in terms of number of LUTs, registers and DSP blocks (multipliers) used, maximum clock speed and extra ISI introduced for the three structures implemented on a Xilinx Kintex 7 FPGA. All three designs perform a $f = 1.2$ resampling, but due to the speed limitations both BM and BS perform this on a slower signal.

factor 1.3 for the registers and 2.2 for the LUTs is mainly due to the 2 large accumulators in the presented structure instead of only one in the BS and BM structure. To reach high clock rates, the accumulators in the three structures are implemented using pipelining techniques, which introduces extra registers in the accumulator.

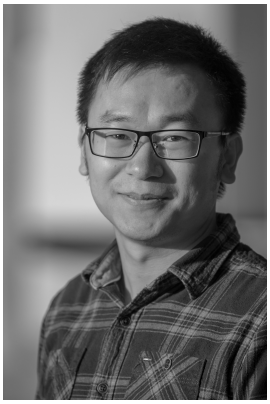
6 Conclusions

We have discussed a new resampling FIR filter architecture for polyphase resampling filters that enables higher sample rates and bandwidths for new communication standards. As a proof of concept, we implemented a 500 to 600 MHz resampling filter showing that it is possible to go to higher data rates with current FPGA technologies. This design is, taking the higher order of parallelism and higher clocking speed into account, comparable in size to other resampling architectures. The different architectures also show a difference in signal distortion. In telecommunication systems, non-ideal resampling gives rise to extra ISI introduced by the resampling filter. This work provides an architecture which is capable of performing the exact resampling operation, hence, without introducing additional ISI.

References

1. Barker, D.: Efficient Resampling Implementations [DSP Tips & Tricks]. IEEE Signal Process. Mag. **25**(4), 114–117 (2008). DOI 10.1109/msp.2008.923508
2. Hansen, C.J.: WiGiG: Multi-gigabit wireless communications in the 60 GHz band. IEEE Wireless Communications **18**(6), 6–7 (2011). DOI 10.1109/MWC.2011.6108325
3. Happi Tietche, B., Romain, O., Denby, B.: A Practical FPGA-Based Architecture for Arbitrary-Ratio Sample Rate Conversion. J Sign Process Syst **78**(2), 147–154 (2013). DOI 10.1007/s11265-013-0840-5
4. Proakis, J.G., Salehi, M.: Digital Communications, fifth edn. McGraw-Hill (2008)
5. Richard, W.D.: Efficient Parallel Real-Time Upsampling with Xilinx FPGAs. Xcell Journal **89**, 38–44 (2014)
6. Xu, Y., Wang, H., Shen, Z.: Modified Polyphase Filter for Arbitrary Sampling Rate Conversion. Wireless Communications Networking and Mobile Computing (WiCOM), 2010 6th International Conference on (2010)

Hannes Ramon received the M.S. degree in electrical engineering from Ghent University, Belgium in 2015. From 2015 on, he has been working towards a Ph.D degree in the Design group of IDLab, Dep. INTEC at the same university and associated with imec. His research interests are high-speed, high-frequency mixed signal designs for (opto-) electronic communication systems and digital signal processing.



Haolin Li received the M.S. degree in electrical engineering from Ghent University, Belgium in 2014. From 2014 on, he has been working towards a Ph.D degree in the Design group of IDLab, Dep. INTEC at the same university and associated with imec. His research focuses on high capacity wireless communication and the internet of things.

Piet Demeester is professor in the faculty of Engineering at Ghent University. He is head of the research group Internet Based Communication Networks and Services (IBCN, Ghent University) and leading the Internet Technologies Department of the strategic research centre iMinds. He is co-author of over 1000 publications in international journals or conference proceedings. He is Fellow of the IEEE.

Johan Bauwelinck received a Ph.D. degree in applied sciences, electronics from Ghent University, Belgium in 2005. Since Oct. 2009 he is a professor in the IDLab research group of the department of Information Technology (INTEC) at the same university where he is leading the Design lab since 2014. He became a guest professor at iMinds in the same year, now imec since 2016. His research focuses on high-speed, high-frequency (opto-) electronic circuits and systems, and their applications on chip and board level, including transmitter and receiver analog front-ends for wireless, wired and fiber-optic communication or instrumentation systems.

Guy Torfs received the M.S. and Ph.D. degree in electrical engineering from Ghent University, Belgium in 2007 and 2012 respectively. From 2007 on, he has been working at the Design group of IDLab, Dep. INTEC at the same university and associated with imec. His research focuses on high-speed mixed signal designs for fiber-optic and backplane communication systems, including equalization circuits and clock and data recovery systems.