Regular Article

Asynchronous Sampling Rate Conversions in Digital Communications Systems

Quang X. Duong[1], Eric R. Pelet[2], J. Eric Salt[1], Ha H. Nguyen[1]

[1]Electrical and Computer Engineering, University of Saskatchewan, Saskatoon, SK, Canada
[2]TRLabs, 111-116 Research Drive, Saskatoon, SK, Canada

Correspondence: Ha Hoang Nguyen, ha.nguyen@usask.ca

Manuscript communication: received 1 August 2011, revised 5 October 2011

Abstract– **The problem of resampling digital signals at an output sampling rate that is incommensurate with the input sampling rate is the topic of this paper. This problem is often encountered in practice, as for example in the multiplexing of video signals from different sources for the purpose of distribution. There are basically two approaches to resample the signals. Both approaches are thoroughly described and practical circuits for hardware implementation are provided. A comparison of the two circuits presented in the paper shows that one circuit requires a division to compute the new sampling times. This time scaling operation adds complexity to the implementation with no performance advantage over the other circuit, and makes the "division free" circuit the preferred one for resampling.**

Keywords– **Asynchronous resampling, sampling rate conversion, cable modem, interpolation, phase locked loop.**

This work was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

1 Introduction

Today's TV networks offer a variety of programs such as news, sport, movies or TV shows. The broadcasting of different video sources over the same network is made possible by gathering the content from the sources at a main distribution center called the headend. The programs may be delivered to the headend over a cable feed or satellite link or may be played from a magnetic tape or optical disk [1].

At the headend the incoming signals are received and processed for transmissions over the distribution network. The nature of the system calls for a sampling rate conversion of the received signals. The reason is that the incoming signals, normally Quadrature Amplitude Modulation (QAM) signals, are received at different sampling rates, and the processing which combines the QAM signals for transmissions over the network requires that all signals be at the same sampling rate.

The incoming QAM signals originate from different transmitters, and thus have different data rates. The pulse shaping filter in the receiver, which normally has a square-root raised cosine frequency response, is designed to run at an integer multiple of the rate at which the data is received, i.e., the input data rate. The sampling rate of the incoming signal, referred to in the sequel as the input sampling rate, is then an integer multiple of the input data rate. The received signals must then be resampled at a common sampling rate, i.e., the output sampling rate, so they can be combined together, and transmitted over the distribution network.

The problem of sampling rate conversion is illustrated in Figure 1, which shows N incoming signals, denoted by $x_1[m_1], x_2[m_2], \ldots, x_N[m_N]$, and their respective input sampling clocks denoted by $clk_{\text{in\_1}}, clk_{\text{in\_2}}, \ldots, clk_{\text{in\_}N}$ with the sampling rates $F_{\text{in\_1}}, F_{\text{in\_2}}, \ldots, F_{\text{in\_}N}$. The sampling rate converter resamples the input signals at the sampling rate F_{out}. The output signals $y_1[k], y_2[k], \ldots, y_N[k]$ are then combined to produce $z[k]$, which is then passed to a digital to analog converter (DAC) to produce $z(t)$.

The block of interest in this paper is the sampling rate converter. Resampling the QAM signals at an output sampling rate that is incommensurate with the input sampling rates is not straightforward. There are basically two approaches to this problem. One approach takes the input sampling rate as a reference to control the digital resampling. The other method uses the output sampling rate as the reference. The thrust of this paper is to describe and provide circuits for both methods. It is shown that one method leads to less circuit complexities with no performance loss.

The asynchronous resampling of digital signals is far from being a new topic, however, to the best of the authors' knowledge, the subtle differences between the two methods of resampling have not been reported in the open literature. As sampling rate conversion is of paramount importance in digital systems, it is believed there is interest in presenting both methods in parallel, highlight their differences, and provide practical circuits for hardware implementation.

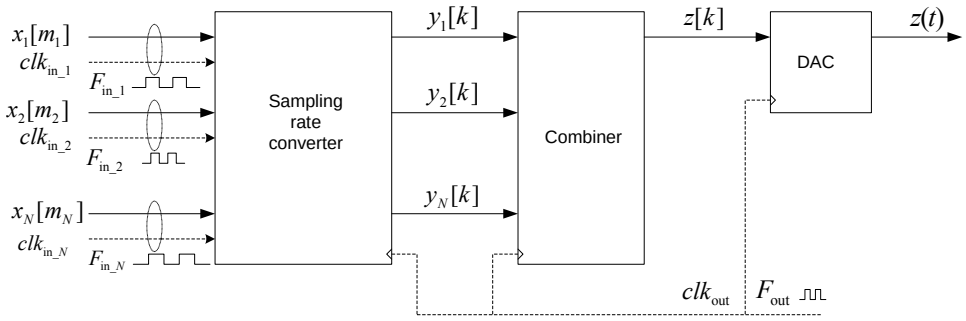


Figure 1: Asynchronous sampling rate conversion.

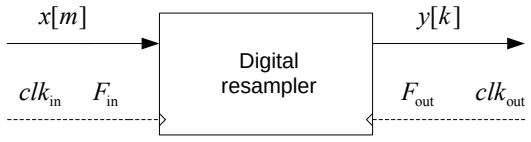


Figure 2: Model of resampling.

The paper is organized as follows. After briefly reviewing the theory of resampling in Section 2, the operations of resampling in hardware is described in Section 3. Circuits for both methods are provided in Section 4. Verification of the circuit operations is the object of Section 5. Conclusions are given in Section 6.

2 Principle of Resampling

Resampling consists of taking a sequence of samples, $x[m] \equiv x(mT_{\text{in}})$, at sampling rate, $F_{\text{in}} = 1/T_{\text{in}}$, and generating a new sequence, $y[k] \equiv y(kT_{\text{out}})$ by means of interplation between the samples of $x[m]$, as described by the block diagram in Figure 2. Inside the resampler shown in Figure 2 there is an interpolator whose purpose is to interpolate between the input samples to produce the output interpolants. By interpolants, it is understood the samples $y[k]$. In this section, the mathematical model for digital interpolation given in [2], [3] is reviewed.

To resample at $F_{\text{out}} = 1/T_{\text{out}}$, the interpolants are computed at times $kT_{\text{out}} = (m_k + \mu_k)T_{\text{in}}$ using $I_2 - I_1 + 1$ input samples, $x[m_k - I_1], x[m_k - I_1 + 1], \ldots, x[m_k - I_2]$ as follows:

$$y(kT_{\text{out}}) = \sum_{i=I_1}^{I_2} x[(m_k - i)T_{\text{in}})]h_I[(i + \mu_k)T_{\text{in}}], \quad (1)$$

where $m_k = \text{int}[kT_{\text{out}}/T_{\text{in}}]$ is the basepoint index, $\mu_k = kT_{\text{out}}/T_{\text{in}} - m_k$ is the fractional interval, $h_I(t)$ is the interpolating function, and int$[\cdot]$ is the integer part of a real number.

The simplest interpolator is the linear interpolator. It only uses 2 samples (i.e., $I_1 = -1, I_2 = 0$) and the interpolating equation (1) reduces to

$$y[k] = x[m_k](1 - \mu_k) + x[m_k + 1]\mu_k, \quad (2)$$

where T_{in} has been set to 1 with no loss of generality. The computation of $y[k]$ using (2) is illustrated in Figure 3. Clearly, $y[k]$ is the point on the line passing through $x[m_k]$ and $x[m_k + 1]$ at a distance of μ_k from $x[m_k]$.

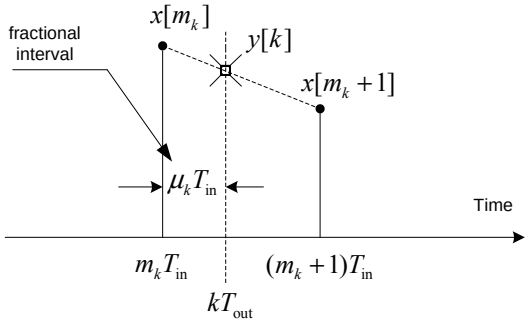


Figure 3: Linear interpolation.

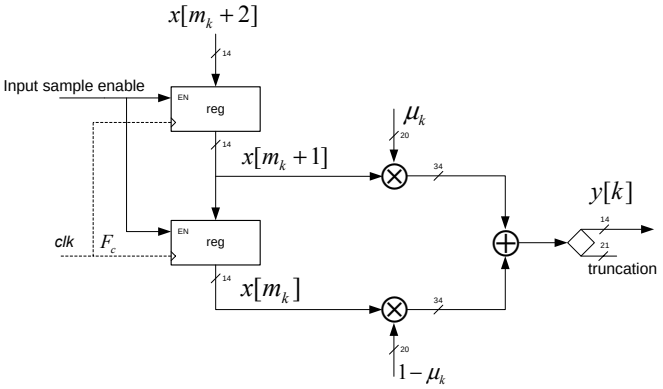


Figure 4: Hardware design of a linear interpolator.

Equation (2) is implemented in hardware with a 2-tap filter as shown in Figure 4, where the blocks labeled "reg" are registers holding the input samples used in the interpolation.

3 Operations of Resampling

3.1 Resampling Operations in Hardware

The key operation of a resampler is the generation of the basepoint index m_k and fractional intervals μ_k. This part presents the hardware circuitry required to generate m_k and μ_k. These circuits are the essential blocks in asynchronous resampling rate converter.

Basically, two time base generators are needed to compute the sampling times, kT_{out}. One generator is clocked by the input clock at frequency, F_{in} (units of samples/second) and the other generator is clocked by

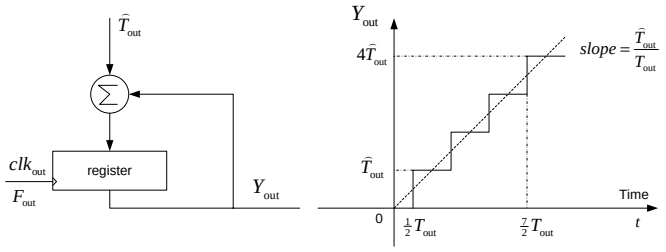


Figure 5: Output clock time base generator.

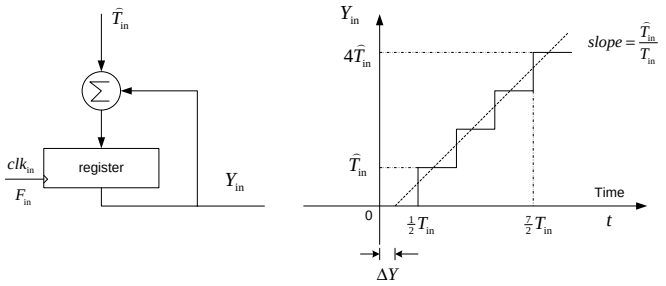


Figure 6: Input clock time base generator.

the output clock at frequency, F_{out}. Let $T_{\text{in}} = 1/F_{\text{in}}$ and $T_{\text{out}} = 1/F_{\text{out}}$ be their respective sampling periods (units of second/sample). Figure 5 shows the circuit for the time base generator clocked at F_{out}. This generator is referred to as the output clock time base. The output Y_{out} is produced by incrementing the content of the register, or accumulator, by $\hat{T}_{\text{out}}$ at every clock cycle, F_{out}. In essence, Y_{out} is the quantized value of time t with step size, $\hat{T}_{\text{out}}$, where $\hat{T}_{\text{out}}$ is a close estimate of T_{out}. Y_{out} is given by

$$Y_{\text{out}} = \hat{T}_{\text{out}} \times \text{int}\left[\frac{t}{T_{\text{out}}}\right] = \frac{\hat{T}_{\text{out}}}{T_{\text{out}}} t + n_{\text{out}}, \tag{3}$$

where t is time, $n_{\text{out}} = -\hat{T}_{\text{out}} \times \text{frac}[t/T_{\text{out}}]$ is the quantization noise, and frac$[\cdot]$ is the fractional part of a real number.

The circuit for the input clock time base generator is very similar and shown in Figure 6. It is clocked by F_{in}, its output is denoted by Y_{in} and its input by $\hat{T}_{\text{in}}$, where $\hat{T}_{\text{in}}$ is a close estimate of T_{in}. Similarly, Y_{in} is the quantized value of time t with step size $\hat{T}_{\text{in}}$ and is given by

$$Y_{\text{in}} = \frac{\hat{T}_{\text{in}}}{T_{\text{in}}}(t - \Delta Y) + n_{\text{in}}, \tag{4}$$

where ΔY represents the timing offset between input and output clocks and $n_{\text{in}} = -\hat{T}_{\text{in}} \times \text{frac}[(t - \Delta Y)/T_{\text{in}}]$ is the quantization noise.

3.2 Synchronizing the Time Bases

Since both Y_{in} and Y_{out} represent quantized values of time t with different step sizes, they are synchronized if and only if

$$Y_{\text{out}} - Y_{\text{in}} = n_{\text{out}} - n_{\text{in}}. \tag{5}$$

This happens when the slope of the output clock time base ramp, Y_{out}, (see Figure 5) is equal to the slope of the input clock time base ramp, Y_{in} (see Figure 6). Both

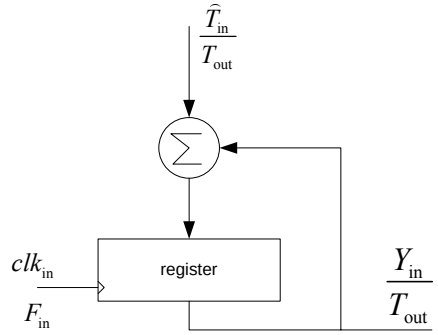


Figure 7: Input clock time base scaled by output rate.

slopes can be made equal by modifying the step size of one of the time base generators by an appropriate amount. The right amount is estimated by means of a phase lock loop (PLL). In other words, the PLL can either estimate $\hat{T}_{\text{in}}$ or $\hat{T}_{\text{out}}$ so that $\hat{T}_{\text{in}}/T_{\text{in}} = \hat{T}_{\text{out}}/T_{\text{out}}$ after the PLL has converged. The timing offset, ΔY, is removed in the process of finding the right step size.

There are two ways to establish synchronization. Either Y_{out} is taken as the reference to estimate $\hat{T}_{\text{in}}$, or Y_{in} serves as the reference to estimate $\hat{T}_{\text{out}}$. In this paper, circuits are given for both methods in Section IV. The sampling rate converter circuit which uses the output ramp as the reference is referred to as a "resampler with input clock time base". The circuit which uses the input ramp as the reference is referred to as a "resampler with output clock time base".

4 Circuit Description

4.1 Resampler with Input Clock Time Base

The output ramp, Y_{out} is the reference to construct the ramp, Y_{in} which in this method is then used to compute m_k and μ_k. The PLL is set up as follows.

Output ramp Y_{out} is normalized by fixing the input of the accumulator (i.e., step size) in Figure 5 to $\hat{T}_{\text{out}} = 1$. This causes the output, Y_{out}, to be scaled by $1/T_{\text{out}}$. Note that by normalizing the output the circuit no longer needs $\hat{T}_{\text{out}}$. As shown in Figure 7, in the process of normalizing the output clock time base generator, the output of the input clock time base generator gets normalized by T_{out}. Its input, denoted by $\hat{T}_{\text{in}}/T_{\text{out}}$, is the estimate produced by the PLL to synchronize input and output time bases.

A block diagram of the resampler with input clock time base is shown in Figure 8. It has three inputs, $x[n]$, clk_{in} and clk_{out} and one output, $y[k]$. N copies of this circuit are required to process N input signals, as illustrated in Figure 1. In Figure 8 a clock at frequency $F_{\text{c}} = MF_{\text{in}}$ is synthesized from clk_{in} using a built-in PLL in the FPGA. M must be chosen large enough so $F_{\text{c}} > F_{\text{out}}$. The block labeled "Output clock time base" in Figure 8 is the time base generator shown in Figure 5, where the input has been set to 1 (normalization by T_{out}). The block labeled "Input clock time base" is the circuit in Figure 7 where the input clock is at frequency $F_{\text{c}} = MF_{\text{in}}$. Its input, denoted by step adjustment in

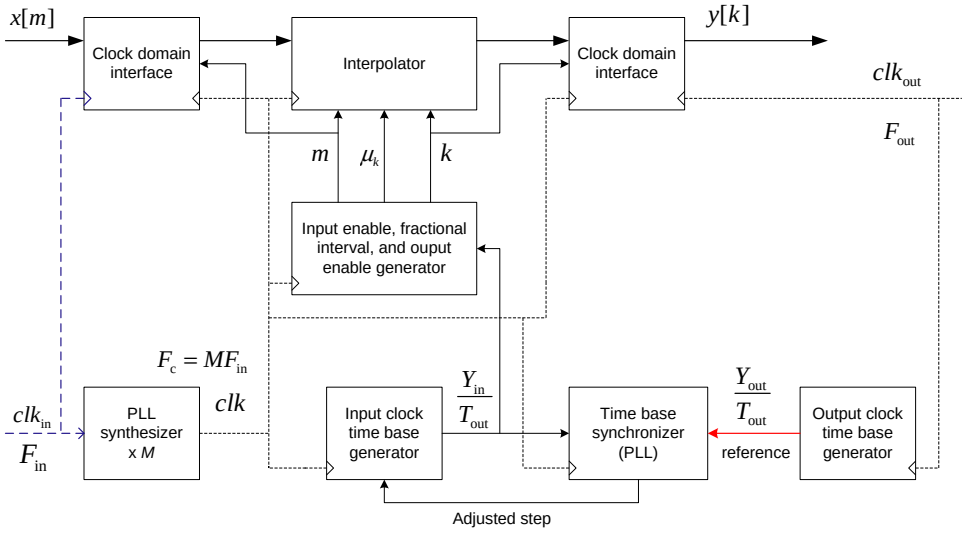


Figure 8: Resampler with input clock time base.

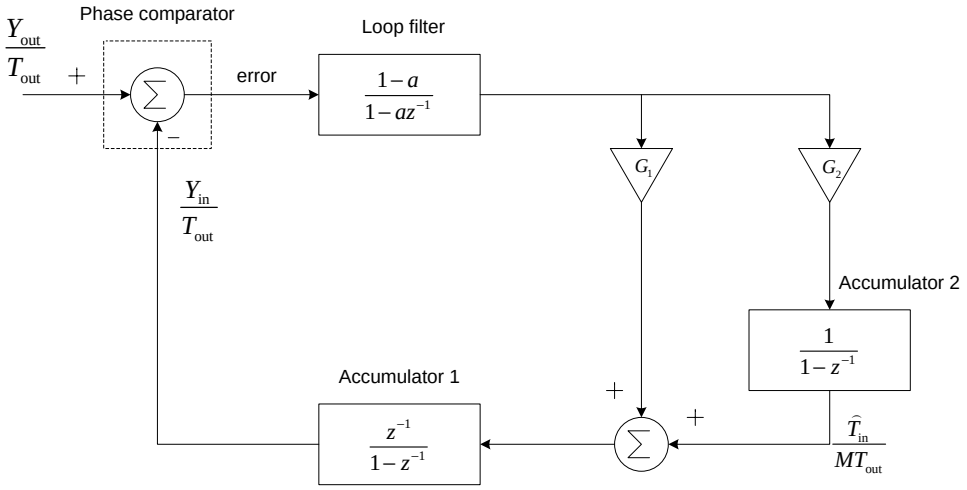


Figure 9: Setting up the PLL.

Figure 8, is generated by the "Time base synchronizer" block. This block is the PLL that is set up to estimate $\hat{T}_{\text{in}}/T_{\text{out}}$. A model for the PLL is shown in Figure 9, where the phase comparator is simply the difference between input and output ramps, Accumulator 1 is the "Output clock time base" block in Figure 8, and Accumulator 2 is an additional accumulator that is needed to hold $\hat{T}_{\text{in}}/T_{\text{out}}$. Accumulator 2 builds up during the acquisition phase to settle to $\hat{T}_{\text{in}}/T_{\text{out}}$. The PLL operates as follows [4]. If the error is positive, then the step size of Accumulator 1 is increased, which causes $Y_{\text{in}}/T_{\text{out}}$ to become larger and the error smaller at the next pass through the loop. In steady state, the two timebases are synchronized. They are both quantized representations of t/T_{out}, one with a step size of 1, and the other with a step size of $\hat{T}_{\text{in}}/T_{\text{out}}$.

The presence of different clocks inside the circuit in Figure 8 calls for a special structure, which allows samples to be written and read using different physical clocks. The average writing and reading rates are the same but the access to the samples inside the structure, represented by the block labeled "Clock domain interface" in Figure 8, occurs at different instants of time. Such a structure is implemented with a circular buffer and logic to control the access of the buffer. Two of these blocks are needed, one to interface the clocking of the input samples and the interpolator and the other one to interface the output of the interpolator and the clocking of the output samples.

The last block which remains to be described in Figure 8 is the "Basepoint index and fractional interval generator" block. This module uses the reconstructed input ramp, $Y_{\text{in}}/T_{\text{out}}$, to produce m_k and μ_k. A new pair of values for m_k and μ_k is computed at a positive clock edge of F_c every time the integer part of $Y_{\text{in}}/T_{\text{out}}$ increases by 1. We say that $Y_{\text{in}}/T_{\text{out}}$ has crossed an integer number and an integer cross-over was detected. An output interpolant is generated by computing its basepoint index m_k and fractional interval μ_k as follows. In the case of the linear interpolator, $m_k + 1$ is the index of the last received sample and m_k is the index of the previous received sample, as shown in Figure 4. The new interpolants are generated at positive edges of F_{c}. Most of the time, especially if M is large, this clock edge does not coincide with a positive clock edge of clk_{in}. The situation with $M = 4$ is depicted in Figure 11,

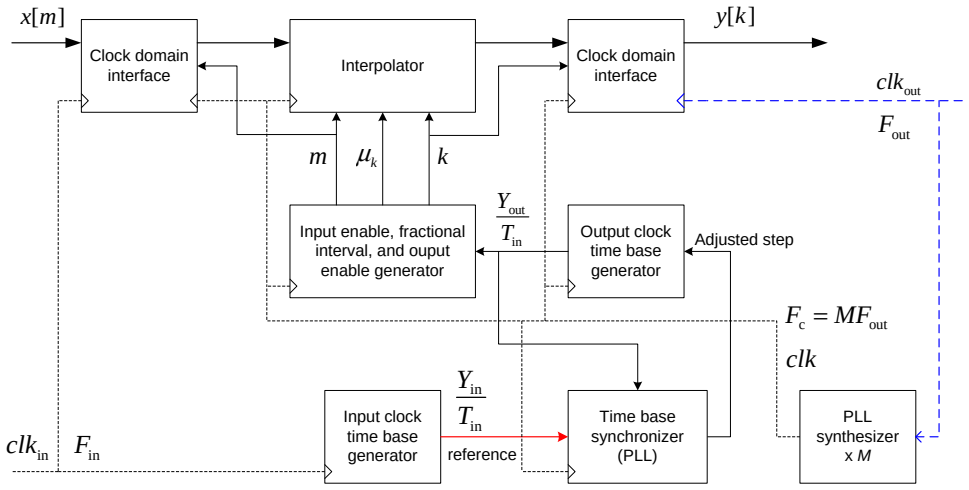


Figure 10: Resampler with output clock time base.

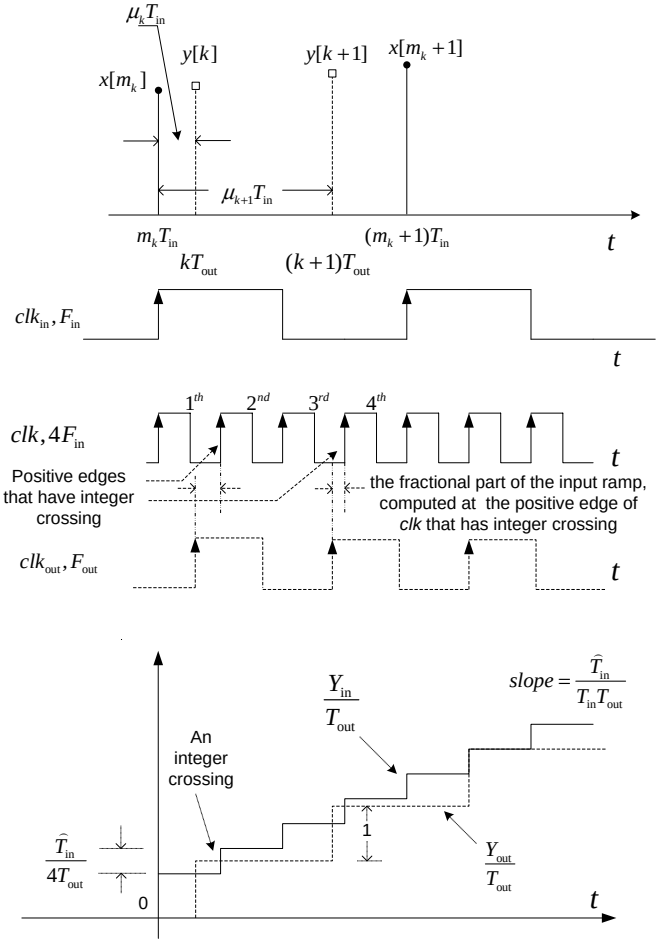


Figure 11: Timing relation in input clock time base resampling.

where the top graph shows the position of $y[k]$ with respect to the input samples and the bottom graph shows all three clocks.

The computation for μ_k depends on which clock edge of F_c the integer cross-over was detected. Since F_c is M times faster than F_{in}, M clock cycles of F_c occur between clock cycles of F_{in}. Suppose that between input sampling times $m_k T_{in}$ and $(m_k + 1)T_{in}$, an integer cross-over is detected at the l_k^{th} positive clock edge of F_c. Then μ_k is given by

$$\mu_k = \frac{l_k}{M} - \frac{T_{out}}{\hat{T}_{in}} \times \text{frac}\left[\frac{Y_{in}}{T_{out}}\right]. \quad (6)$$

In Figure 11, we show the case for $M = 4$, $l_k = 1$, $l_{k+1} = 3$.

There is a division in (6) which needs explanation, as it is not required in the second circuit, and causes additional complexity in the calculation of μ_k. The fractional part, frac $\left[\frac{Y_{in}}{T_{out}}\right]$, which is used to compute the time that separates the new interpolant from the input sample, is expressed as a fraction of the output clock cycle. This time must be converted in units of fraction of the input clock cycle to yield μ_k. This is achieved in (6) by dividing frac$[Y_{in}/T_{out}]$ by $\hat{T}_{in}/T_{out}$.

4.2 Resampling with Output Clock Time Base

In this method, Y_{in} is used as the reference to construct Y_{out}, which is then used to compute m_k and μ_k. The circuit is shown in Figure 10. Similarly the clock, clk, at frequency $F_c = M \times F_{out}$ is synthesized. Typically, M will be small, even equal to 1 in which case no PLL-synthesizer is needed, since the output sampling rate is normally much higher than the input rates.In this circuit, the step size of the "Input clock time base" block is set to 1 and the step size of the output clock time base is determined with the PLL. The main difference is that a new interpolant is produced at every positive edge of the resampling clock F_{out}. In this case, whenever an integer cross-over in the reconstructed output ramp is detected, a new input sample is loaded into the interpolator. The fractional interval is simply given by

$$\mu_k = \text{frac}\left[\frac{Y_{out}}{T_{in}}\right]. \quad (7)$$

In this case there is no division since the fractional part is already expressed as a fraction of the input clock cycle. Recall that in this method the step size of the output clock time base generator is adjusted to reproduce the ramp of the input clock, and therefore its content is directly given in the input clock domain.

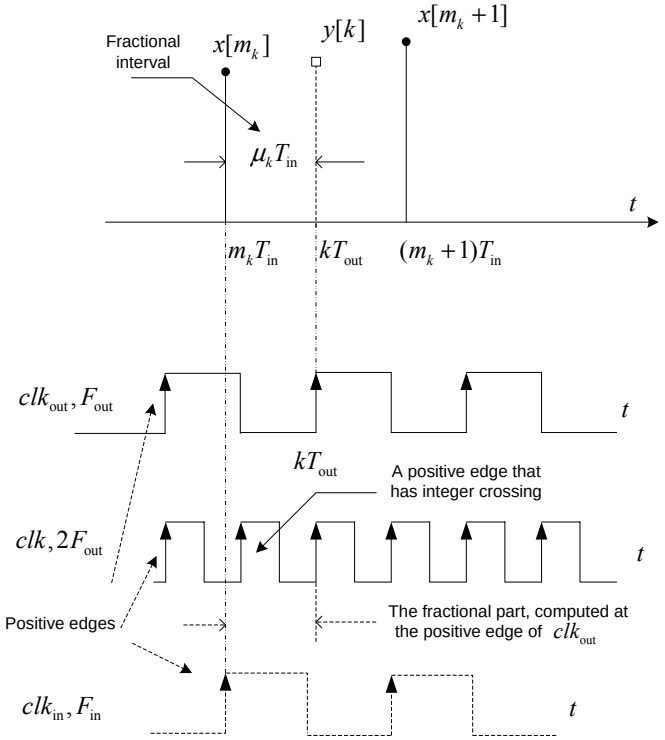


Figure 12: Timing relation in output clock time base resampling.

The timing relation for the case $F_c = 2F_{out}$ is depicted in Figure 12.

As internal PLL synthesizers are readily available in FPGA, any of the two methods can be applied independently on the relation between input and output sampling rates. Since the second circuit leads to less hardware complexity by avoiding a division, the second method, namely resampling with output clock time base, is much preferred.

5 Verification

Both resampler circuits were simulated in Matlab/Simulink software. The input signal $x[m]$ was generated with 4 samples per symbol using a random BPSK sequence and a raised cosine pulse shaping filter with a roll-off factor equal to 0.25.

The first simulation is set up as follows. A resampler with input clock time base is simulated using the Simulink software to resample the input signal from input rate F_{in} to output rate $F_{out} = (4/3)F_{in}$. The circuit is clocked by system clock clk at rate $F_c = 4F_{in}$ (i.e., $M = 4$). The PLL (see Figure 9) was set up with small loop gains to reduce the effect of timing jitter, which is the random fluctuation in the timing [5].

Figure 13 shows the evolution of the step size, which converges to $1/3$ and matches to the theoretical value, i.e., the input $\hat{T}_{in}/(MT_{out})$ of the accumulator. In steady state, the step size is fluctuated around $1/3$ as shown in Figure 14.

Figure 15 shows the convergence of the input clock time base Y_{in} to the output clock time base Y_{out} (as the reference). Note that the output time T_{out} is normalized

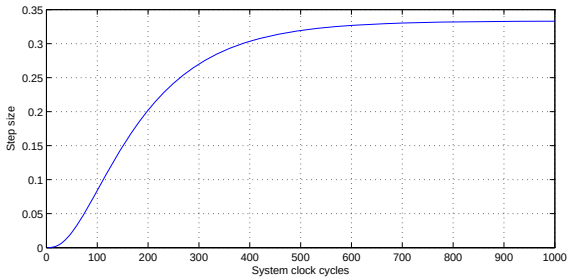


Figure 13: Evolution of the step size.

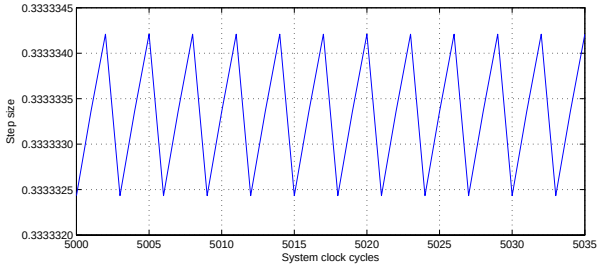


Figure 14: The step size in steady state.

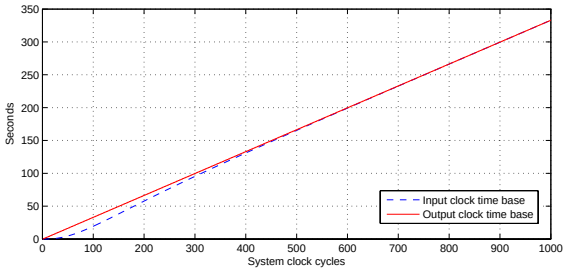


Figure 15: Variation of the input clock time base and the output clock time base.

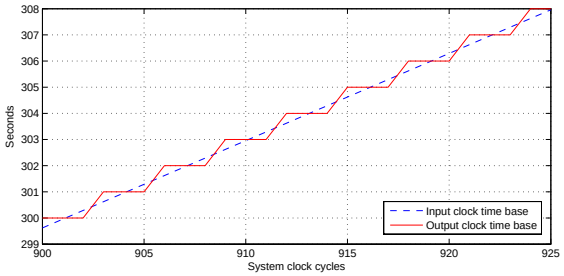


Figure 16: The input clock time base synchronizes with the output clock time base.

to 1 second. Figure 16 shows the two ramps which are synchronized after the step size has been converged.

Figure 17 shows sets of input samples (marked with a circle) and output interpolants (marked with an asterisk) after the circuit has found the correct step size and the two time bases have synchronized. The corresponding fractional intervals (marked with an asterisk) are shown in Figure 18. Cubic interpolator is used in this simulation.

The second simulation is set up to show the Power Spectral Densities (PSDs) of the resampled signals. The PSDs of the resampled signals were estimated using

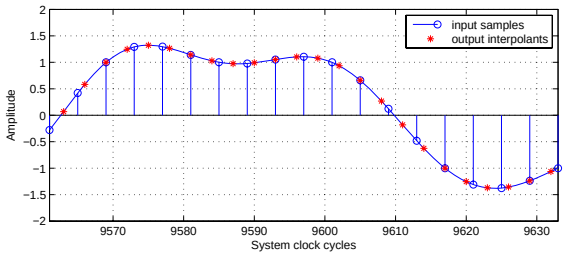


Figure 17: Input samples and output interpolants.

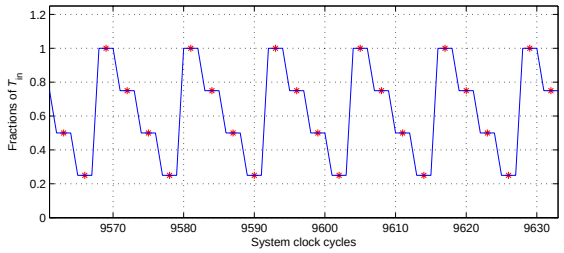


Figure 18: Fractional intervals.

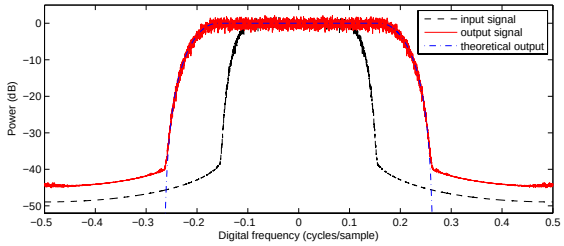


Figure 19: Resampler with input clock time base - PSDs for conversion rate ≈ 0.59.

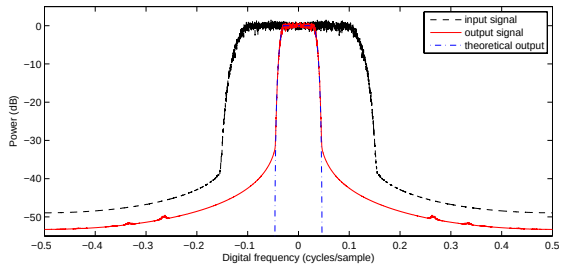


Figure 20: Resampler with input clock time base - PSDs for conversion rate ≈ 3.33.

Welch's 50% overlapping method with blocks of length 2^{13} samples intervals [6, Ch. 8]. Both resampler circuits were simulated with two different conversion rates, $10/3 \approx 3.33$ and $10/17 \approx 0.59$ of the input rate, where the input rate was $F_{\text{in}} = 4/T$ with $1/T$ being the symbol rate. The interpolator was a third order Farrow filter. The PSD estimates obtained from the resampler with input clock time base are plotted in Figure 19 and Figure 20 (solid curve) and the PSD estimates obtained from the resampler with output clock time base are plotted in Figure 21 and Figure 22 (solid curve).

Theoretical curves (dashed curves) are also plotted in all the graphs of Figures 19, 20, 21, and 22. The equation for the theoretical curves is given in (8), which is the

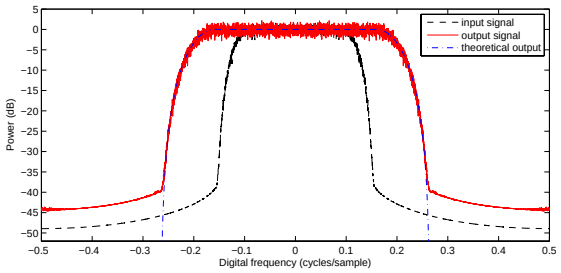


Figure 21: Resampler with output clock time base - PSDs for conversion rate ≈ 0.59.

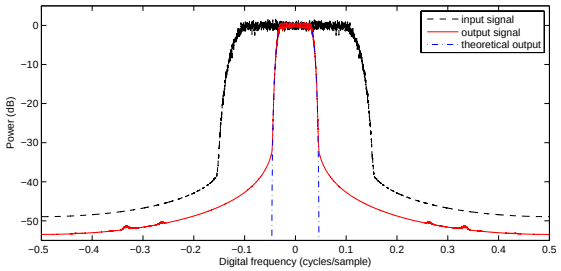


Figure 22: Resampler with output clock time base - PSDs for conversion rate ≈ 3.33.

frequency response of the raised cosine pulse [7, Ch. 9, p. 560].

$$H(f) = \begin{cases} 1, & |f| \leq \frac{1-\beta}{2T} \\ \frac{1}{2}\left[1 + \cos\left(\frac{\pi T}{\beta}\left[|f| - \frac{1-\beta}{2T}\right]\right)\right], & \frac{1-\beta}{2T} \leq |f| \leq \frac{1+\beta}{2T} \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

In Equation (8), T is the symbol period and β is the roll-off factor, which is a real number between 0 and 1. The roll-off factor β determines the excess bandwidth of the filter. The excess bandwidth is the bandwidth occupied beyond the Nyquist bandwidth of $1/(2T)$.

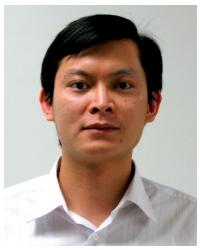
The close agreement between the experimental and theoretical curves strongly suggests that the circuits work properly. From the plots, the performances of both circuits appear to be identical. However, the resampler using output clock time base requires no division for the computation of μ_k and therefore is the preferred one.

6 Conclusion

Two resampler circuits were described and simulated in the Matlab/Simulink software. Simulations show that both circuits offer similar performances but one of them, namely the resampler circuit which uses the output clock time base has a hardware implementation advantage over the other one. It does not require any division to compute the position of the new samples with respect to the incoming samples. This is a significant advantage in FPGA implementation.

References

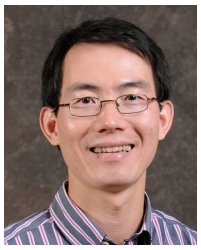
[1] L. Harte, *Introduction to Cable Television (CATV): Analog and Digital Television and Modems*, 2nd ed. Althos, 2007.
[2] F. M. Gardner, "Interpolation in digital modems.I. Fundamentals," *IEEE Transactions on Communications*, vol. 41, no. 3, pp. 501–507, Mar. 1993.
[3] L. Erup, F. M. Gardner, and R. A. Harris, "Interpolation in digital modems. II. Implementation and performance," *IEEE Transactions on Communications*, vol. 41, no. 6, pp. 998–1008, June 1993.
[4] F. M. Gardner, *Phaselock Techniques*, 3rd ed. Wiley-Interscience, 2005.
[5] E. R. Pelet, "Synchronization in all-digital QAM receivers," Ph.D. dissertation, University of Saskatchewan, Saskatoon, Saskatchewan, Canada, 2009.
[6] M. H. Hayes, *Statistical Digital Signal Processing and Modeling*. John Wiley & Sons, 1996.
[7] J. G. Proakis, *Digital Communications*, 4th ed. Mc Graw-Hill, 2006.



Duong Xuan Quang received the B.Eng. degree in Electronics and Telecommunications from Hanoi University of Technology, Hanoi, Vietnam in 2005. He worked at Vietnam Internet Network Information Center for two years and Ericsson Vietnam for one year before coming to Canada to pursue his graduate studies. Under the supervision of Prof. Ha Nguyen and Dr. Eric Pelet, he received the M.Sc. degree in Electrical and Computer Engineering from the University of Saskatchewan, Saskatoon, SK, Canada in 2010. He was with Vecima Networks, Saskatoon, SK, Canada in 2010 prior to joining Fortinet Technologies, Burnaby, BC, Canada, where he is currently working in FortiGuard Labs.

Eric R. Pelet received the degree "Diplome d'ingénieur" from the Ecole Nationale Supérieure des Téécommunications de Bretagne (Telecom Bretagne), Brest, France, in 1994. After working in the electronic industry for seven years, Eric Pelet joined the Electrical and Computing Engineering Department of the University of Saskchewan, Saskatoon, in 2001 to obtain M.Sc. degree in 2003 and Ph.D. degree in 2009. Eric Pelet received several awards and scholarships during his study at the University of Saskatchewan. Following graduation, Eric Pelet was nominated by both TRLabs, Saskatoon, Canada, and Vecima Networkds, Saskatoon, Canada to apply for an NSERC Industrial Research & Fellowship, which he successfully obtained, to devise digital cable receivers for next generation cable systems. Email address: eric.pelet@vecima.com.

J. Eric Salt received B.Sc. and Ph.D. degrees in electrical engineering from the University of Saskatchewan, Saskatoon, SK, Canada, in 1974 and 1987, respectively, and the M.Eng. degree in electrical engineering from Carleton University, Ottawa, ON, Canada, in 1978. He joined the Department of Electrical Engineering, University of Saskatchewan, in 1985, where he presently holds the rank of Professor. His current research interests are in DSP algorithms and architectures for implementing CATV headend standards in FPGAs.



Ha H. Nguyen received the B.Eng. degree from the Hanoi University of Technology (HUT), Hanoi, Vietnam, in 1995, the M.Eng. degree from the Asian Institute of Technology (AIT), Bangkok, Thailand, in 1997, and the Ph.D. degree from the University of Manitoba, Winnipeg, MB, Canada, in 2001, all in electrical engineering. He joined the Department of Electrical and Computer Engineering, University of Saskatchewan, Saskatoon, SK, Canada, in 2001, and became a full Professor in 2007. He holds adjunct appointments at the Department of Electrical and Computer Engineering, University of Manitoba, Winnipeg, MB, Canada, and TRLabs, Saskatoon, SK, Canada, and was a Senior Visiting Fellow in the School of Electrical Engineering and Telecommunications, University of New South Wales, Sydney, Australia during October 2007-June 2008. His research interests include spread spectrum systems, error-control coding and diversity techniques in wireless communications. Dr. Nguyen was an Associate Editor for the IEEE Transactions on Wireless Communications during 2007-2011. He currently serves as an Associate Editor for the IEEE Transactions on Vehicular Technology and the IEEE Wireless Communications Letters. He was a Co-chair for the Multiple Antenna Systems and Space-Time Processing Track, IEEE Vehicular Technology Conferences (Fall 2010, Ottawa, ON, Canada and Fall 2012, Quebec, QC, Canada). He is a coauthor, with Ed Shwedyk, of the textbook "A First Course in Digital Communications" (published by Cambridge University Press).