

High-Performance Low-Power Digital Linear Interpolation Filter

Magdy El-Moursy¹, Member IEEE and Ahmed G. Radwan²

¹ Mentor Graphics Corporation, Cairo, Egypt
Electronics Research Institute, Cairo, Egypt

² Institute for Electronics Engineering, University of Erlangen, Nuremberg, Germany

Abstract

A new technique to implement digital interpolation filter is presented in this paper. The technique employs a sample calculation functional block which reduces the hardware required to realize the filter by orders of magnitude. The filter is realized with 80X160 μm^2 using 65 nm CMOS technology. Over Sampling Rate of up to 256 is achieved for 16-bit digital data sampled at 705,600 bps. The filter dissipates 77.68 mW when operating at frequency of 833.3 MHz.

Keywords: Digital interpolation, Oversampling, Digital Low Pass Filters, DAC, Delta-Sigma Modulators, FFT, FIR, DSP

1. Introduction

Interpolation filter is used in many digital signal processing (DSP) applications. Delta-Sigma based Digital-to-Analog converters (Δ - Σ DAC) employ an up-sample and interpolation filter to obtain high precision by speeding up the conversion processing over less number of bits [1]-[3]. Shown in Fig. 1(a) is up-sampling and interpolation filter which is performed by inserting zeros (known as zero-stuffing) to the in-between over-sampled data samples followed by a digital Low Pass Filter (LPF). This technique is widely used to oversample/interpolate digital signal [2]-[4]. Digital low pass filters (LPF) require large computational blocks. Discrete Fourier Transform (DFT) and Fast Fourier Transform (FFT) are used to reduce the computational cost of the filter. Yet, FFT requires tens, if not hundreds, of multipliers to be implemented [5]-[10]. In addition to the high cost in terms of hardware, stability is another factor which adds to the complexity of using DFT [11]-[13]. In this paper a new multiplier-free technique, as shown in Fig. 1(b), to implement digital interpolation filter is presented. The technique could be used to interpolate digital data represented in either fixed point or floating point number format. The presented technique uses Finite Impulse Response (FIR) and requires much less hardware to realize the interpolation filter.

Oversampled digital data could be interpolated using different interpolation techniques. Hold interpolation is the simplest of all. The up-samples hold the value of the slow samples in the region between input samples. Up-sampling in Δ - Σ modulators is based on reducing the difference between the samples which makes hold interpolation less attractive (for Δ - Σ modulators). Non-linear interpolation requires storing input samples over long period to determine the over-sampled sequence. However, linear interpolation requires less hardware since only two samples are sufficient to determine the output sequence. Linear interpolation is sufficient when high Over-Sampling Rate (OSR) is used. Since widely used in DSP, linear interpolation is adopted in the proposed interpolation filter.

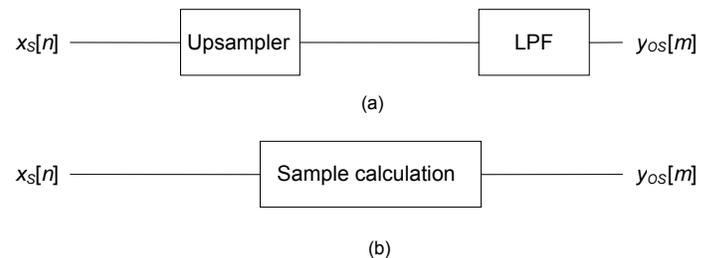


Fig. 1. Digital up-sample and interpolation filter. (a) conventional technique with LPF. (b) sample calculation technique.

The paper is organized as follows. In section 2, the theory behind the sample calculation of the interpolation filter is described. Some circuit implementation issues are described in section 3. In section 4, simulation results are presented. Some conclusions are provided in section 5.

2. Sample Calculation Techniques

Sample calculation interpolation technique is presented in this section. In section 2.1, sample calculation interpolation technique for fixed point number representation is described.

The technique is extended to include floating point numbers in section 2.2. In section 2.3, the technique to evenly distribute the calculated samples is provided.

2.1 Fixed-Point Number Representation

Fixed point number representation is assumed to represent the input data samples $x_s[n]$, however, the methodology could be generalized to include both fixed point and floating point number representation as described in section II.B. Fixed point number representation dedicates k bits out of the total number of bits l to the integer part and $(l-k)$ bits to the fraction part. Without loss of generality, the decimal point is shifted right $(l-k)$ bits to convert the number to integer $x[n]$ to simplify the sample calculation process. The decimal point of the output $y[m]$ is shifted back left $(l-k)$ bits to maintain the format of the number representation of the oversampled output $y_{os}[m]$.

Every two successive samples of the integer input data sequence $x[n]$ and $x[n+1]$ are used to determine the oversampled and interpolated output $y[m]$. The output $y[m]$ (referred to by $y[n,j]$) is produced at the OSR , where $m=(n,j)$ is two dimensional index shown in Fig. 2, j is index changing from 0 to N_s-1 , and $N_s=OSR$. In sample calculation interpolation filter, the value of the output sequence $y[n,j]$ is determined by direct calculation of the samples to avoid using digital LPF which is needed in the conventional techniques. Since $x[n]$ is integer, $y[n,j]$ is expected to be an integer possessing the same precision (number of bits l) of $x[n]$.

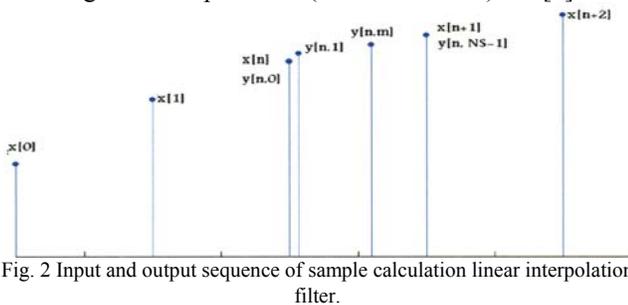


Fig. 2 Input and output sequence of sample calculation linear interpolation filter.

The difference between two successive integer (fixed point number with shifted decimal point) input samples $x[n]$ and $x[n+1]$, Δ is integer as well,

$$\Delta = x[n+1] - x[n] \quad (1)$$

For ideal linear interpolation, the ideal output $y_{id}[n,j]$ is determined by,

$$y_{id}[n,j] = x[n] + j \cdot Step \quad (2)$$

given

$$Step = \frac{\Delta}{N_s} \quad (3)$$

Since $y[n,j]$ is integer and $Step$ could be fraction, (2) could not be used to determine $y[n,j]$. The output $y[n,j]$ could be either

$$y[n,j] = \lfloor y_{id}[n,j] \rfloor \text{ or } \lceil y_{id}[n,j] \rceil \quad (4)$$

Note that for perfect linear interpolation,

$$y[n,0] = x[n] \quad (5)$$

$$y[n,N_s-1] = x[n+1] \quad (6)$$

In order to satisfy (3)-(6), non-ideal interpolation is used. For DSP applications such as $(\Delta-\Sigma)$ DAC, non-ideal interpolation is tolerable since the reduction in hardware complexity could be orders of magnitude. Furthermore, quantization error, always exists in DSP. Moreover, for high OSR , non-ideal interpolation is acceptable.

The number of output sequence samples $y[n,j]$ in-between two successive input samples $x[n]$ and $x[n+1]$ equals N_s . In the ideal case each output sequence should be incremented by $Step = \Delta/N_s$. Rather than using fraction number to represent the output samples either the *ceiling* or the *floor* of the $Step$ is used. In order to ensure that all output sequence samples are integer the output samples are divided into two sets of outputs $y_c[n,j]$ and $y_f[n,j]$, where

$$y_f[n,j] = y[n,j-1] + N \quad (7)$$

$$y_c[n,j] = y[n,j-1] + N_1 \quad (8)$$

$$N = \lfloor \frac{\Delta}{N_s} \rfloor \quad (9)$$

$$N_1 = \lceil \frac{\Delta}{N_s} \rceil = N + 1 \quad (10)$$

The output sequence

$$y[n,j] = \begin{cases} y_c[n,j] & \text{for } j = 0_{y_c} \\ y_f[n,j] & \text{for } j = 0_{y_f} \end{cases} \quad (11)$$

The set of indices of the output samples which take the value $y_c[n,j]$, 0_{y_c} , and which take the value $y_f[n,j]$, 0_{y_f} is determined in section II.C.

In order to satisfy (5)-(8), the number of samples which take the value $y_f[n,j]$, p and the number of samples which take the value $y_c[n,j]$, q are to be determined to satisfy,

$$p + q = N_s \quad (12)$$

$$N \cdot p + N_1 \cdot q = \Delta \quad (13)$$

Given Δ and N_s , p and q are determined by solving (12) and (13) simultaneously,

$$p = N_1 \cdot N_s - \Delta \quad (14)$$

$$q = \Delta - N \cdot N_s \quad (15)$$

In order to reduce computational complexity in most DSP applications, *modulo-2 OSR* is used. Since only *floor* and *ceiling* of division by N_s are to be determined, simple r -bit left shift, where $r = \log_2 N_s$, is sufficient to realize (9) and (10). Also multiplication in (14) and (15) is realized by simple r -bit shift right, significantly reducing the hardware needed to calculate the samples and simplify the implementation of the samples calculation technique. Note that, no multipliers or divisors are needed to implement the filter.

2.2 Floating-Point Number Representation

In section II.A fixed point number representation was assumed for the input sequence $x[n]$. The presented technique

is generalized to include floating point numbers in this section. It is to be noted that equations (9), (10), (14), and (15) are the equations needed to realize the filter. The output samples are determined by simple addition in (7) and (8). Simple floating point addition/subtraction is needed to realize the filter. The output samples are determined by simple addition in (7) and (8). In addition, multiplication and division by N_s are to be implemented. Under the assumption of having *OSR modulo-2*, multiplication and division is performed on floating point numbers by simple addition and subtraction of the exponent of the number, respectively. Without lose of accuracy, in case of underflow or overflow in the exponent, the mantissa of the floating point number is treated like the integer number in section II.A. The technique could be used for both fixed point and floating point number representation without lose in accuracy. Once again multiplication and division are completely eliminated.

2.3 Uniform Distributed of Calculated Samples

It is shown in section II.A that the output sequence is divided into two sets of output samples; the samples which are incremented by N , $y[n,j]$ and the samples which are incremented by N_1 , $y_c[n,j]$. The number of output samples, which take the value $y[n,j]$, p and the number of output samples, which take the value $y_c[n,j]$, q are provided in (14) and (15) respectively. In order to obtain linear interpolation, the samples which take the values $y[n,j]$ and $y_c[n,j]$ are evenly distributed along the output samples sequence. Depending on solving (14) and (15), if $p \neq q$; whether $p > q$ or $p < q$, the majority of $y[n,j]$ take either the value of $y[n,j]$ or $y_c[n,j]$, respectively. Among the N_s output samples, majority samples y_{Maj} appears N_{Maj} times in the output samples and minority samples y_{Min} appears N_{Min} times in the output samples, where

$$\begin{aligned}
 N_{Maj} &= \text{Max}(p, q), & (16) \\
 N_{Min} &= \text{Min}(p, q), & (17) \\
 \text{if } N_{Maj} &= p, \quad y_{Maj} = y[n,j], & (18) \\
 \text{and } N_{Min} &= q, \quad y_{Min} = y_c[n,j], & (19) \\
 \text{if } N_{Maj} &= q, \quad y_{Maj} = y_c[n,j], & (20) \\
 \text{and } N_{Min} &= p, \quad y_{Min} = y[n,j], & (21)
 \end{aligned}$$

The distribution of the samples y_{Maj} and y_{Min} is based on one of two patterns *Patn1* or *Patn2* as shown in Fig. 3.

Sample index	0	1	2	<i>OSR-1</i>
Output sample	y_{Maj}	y_{Min}	y_{Maj}	y_{Min}

(a)

Sample index	0	1	2	<i>OSR-1</i>
Output sample	y_{Maj}	y_{Maj}	y_{Maj}	y_{Maj}

(b)

Fig. 3. The distribution of the minority and majority samples

in the output sequence a) pattern *Patn1* b) pattern *Patn2*.

Pattern *Patn1* is used when $N_{Min} \geq N_{Maj}/2$. Minority sample is assumed to exist after each majority sample in this pattern. The difference between the number of majority samples and minority samples, $D = N_{Maj} - N_{Min}$ is determined. A number of minority samples equals $D/2$ is replaced in the pattern *Patn1* with majority samples to satisfy (12), (16), and (17). The special case of $p=q$ is included in this patterns when $N_{Min}=N_{Maj}$. In order to evenly distribute the majority and minority samples, the index of the minority samples which are replaced with majority samples, $O_{y_{Maj}}(i)$ is determined by,

$$\begin{aligned}
 O_{y_{Maj}}(i) &= 2 * \left\lfloor \frac{N_s}{2} (C + 1) \right\rfloor - 1 & (22) \\
 \text{where,} \\
 C &= D/2
 \end{aligned}$$

$$i = 1, 2, 3, 4, 5, \dots, C$$

For $N_{Min} < N_{Maj}/2$, pattern *Patn2* is used to distribute the samples. In *Patn2* all samples are assumed to be majority samples. A number of majority samples equals N_{Min} is replaced in the pattern with minority samples to satisfy (12), (16), and (17). In order to evenly distribute the majority and minority samples, the index of the majority samples which are replaced with minority samples, $O_{y_{Min}}(i)$ is determined by,

$$\begin{aligned}
 O_{y_{Min}}(i) &= 2 * \left\lfloor \frac{N_s}{2(N_{Min} + 1)} \right\rfloor - 1 & (23) \\
 \text{where,} \\
 i &= 1, 2, 3, 4, 5, \dots, N_{Min}
 \end{aligned}$$

This distribution technique guarantees even distribution for the minority samples along the majority samples. The interpolated sequence is closest to linear under the digital nature of the output sequence. The linearity of the interpolation filter is demonstrated in section IV.

3. Circuit Implementation

The presented technique reduces the hardware required to realize the interpolation filter. Different techniques were considered to achieve further reduction in transistor count and power dissipation. A special case for the input sequence is to have no difference between two successive input samples ($\Delta = 0$). In this case, all output samples $y_{os}[m]$ take the value of the input sample $x_s[n]$ reducing the power dissipation by eliminating the activity in the computation block. Also, division in (22) and (23) is implemented by iterative addition. N_s is divided by two first (before performing the division) to reduce the number of iterations of the division block.

In order to satisfy (12), D must be an even number. C is realized by single shift right operation. Binary shift and addition were also sufficient to realize (9), (14), and (15). Division used in calculating $O(i)$ in (22) and (23) is done using addition and subtraction operation to calculate the floor value of the quotient. Some simulation results are included in

section IV.

4. Simulation Results

Some simulation results are provided in this section. In subsection IV.A, the interpolation algorithm is demonstrated. Simulation results for circuit implementation are summarized in subsection IV.B.

4.1 Interpolation Algorithm

The sample calculation interpolation algorithm is implemented using Matlab7.0. *OSR* of 64 is assumed. The over-sampled and interpolated output sequence is determined to demonstrate the accuracy of linear interpolation. Different input sequences are assumed. The output sequence is shown in Fig. 4 (a), (b), (c), and (d) for $\Delta = 0$, $N_{min} > N_{maj}/2$, $N_{min} < N_{maj}/2$, and $p < q$, respectively. Under the digital nature of the output sequence, highly linear and accurate interpolation is achieved as shown in the figure for different cases.

In Fig. 4(a), $\Delta = 0$, consequently no interpolation is needed and the new samples have the same value of the original samples. In this case, both $x[n]$ and $x[n+1] = 100$, $p = 64$, $q = 0$, and consequently p is assigned to N_{Maj} and q to N_{Min} . In Fig 4(b), $p > q$; the number of samples incremented by N is greater than that incremented by N_I . Also, $N_{Min} \geq N_{Maj}/2$, $Patn1$ is used to calculate the index of the output samples $U_{y[n]}$. For $x[n] = 50$ and $x[n+1] = 20$, $p = 34$, $q = 30$, p is assigned to N_{Maj} and q to N_{Min} . In Fig 4(c), $p > q$, $N_{Min} < N_{Maj}/2$, $Patn2$ is used to calculate the index of the output samples $U_{y[n]}$. In this case $x[n] = 50$, $x[n+1] = 120$, $p = 58$, and $q = 6$. Consequently p is assigned to N_{Maj} and q to N_{Min} . In Fig. 4(d), $p < q$; the number of samples incremented by N_I is greater than that incremented by N , in this case $x[n] = 20$, $x[n+1] = 70$, $p = 14$ and $q = 50$. Consequently q is assigned to N_{Maj} and p to N_{Min} . High linearity is demonstrated in different cases in Fig. 4.

4.2 Interpolation Filter

The new interpolation filter is implemented using 65 nm CMOS technology. The sample rate of compact disc digital audio system of 44.1 KHz is used. For 16-bit data sample, input data sequence of 705,600 bps is assumed. The *OSR* is determined for the required performance. Linear interpolation with *OSR* up to 256 is achieved with the implemented technique. The filter occupies an area of $80 \times 160 \mu m^2$. SPICE simulation is used to estimate the power dissipation of the circuit. The filter dissipates 77.68 mW when operating at 833.3 MHz. The filter specifications are summarized in Table 1.

5. Conclusions

In A new technique to implement digital interpolation filter is presented. The technique employs a sample calculation functional block, avoiding using multipliers in the filter and reducing the hardware to implement the filter. The filter is realized with $80 \times 160 \mu m^2$ using 65 nm CMOS technology. Over Sampling Rate of up to 256 is achieved for 16-bit digital data sampled at

705,600 bps. The filter dissipates 77.68 mW when operating at frequency of 833.3 MHz. The presented sample calculation technique reduces the hardware required to realize the filter by orders of magnitude while achieving higher Over Sampling Rate as compared to FIR technique.

References

- [1] Bernard Sklar, *Digital Communication Fundamentals and Application; Second Edition*, Prentice Hall, 2004.
- [2] Tzu-Chiek Kue; Kwentus, A.; Willson, A.N., "A Programmable Interpolation filter for Digital Communications Applications", *IEEE International Symposium on Circuits and Systems*, Vol. 2, pp.97-100, May 1998.
- [3] Udo Zolzer, *Digital Audio Signal Processing*, John Wiley & Sons, 1997.
- [4] Richard G. Lyons, *Understanding Digital Signal Processing; Second Edition*, Pearson Education, 2004.
- [5] John G. Proakis and Dimitris G. Manolakis, *Digital Signal Processing, Principles, Algorithms, and Applications; Fourth Edition*, Prentice Hall, 2007.
- [6] C. J. Pan, "A Low Power Digital Filter for Decimation and Interpolation using Approximation processing", *IEEE International Solid-State Circuits Conference*, pp 102-103, 439, February 1997.
- [7] H.K. Kwan, "High-Order Tunable Passive Digital Filters", *IEEE International Symposium in Circuits and Systems*, Vol. 2, pp II.700-II.703, May 2002.
- [8] I.R. Khan, M. Okuda, and R. Ohba, "New Designs of Frequency Selective FIR Digital Filters", *IEEE International Symposium on Circuits and Systems*, Vol. 4, pp. IV.185-IV.188, May 2003.
- [9] M. Vollmer and H. Kopmann, "A Novel Approach to an IIR Digital Filter Bank with Approximately Linear Phase", *IEEE International Symposium on Circuits and Systems*, Vol. 2, pp.II.512-II.515, May 2002.
- [10] M. Bhattacharya and T. Saramaki, "Allpass Structures for Multiplierless Realization of Recursive Digital filter", *IEEE International Symposium on Circuits and Systems*, Vol. 4, pp. IV.237-IV.240, May 2003.
- [11] Steven r.Norsworthy, Richard Schreier, Gabor c. Temes, "Delta-Sigma Data Converters Theory, Design, Simulation." *IEEE Press*, 1997.
- [12] Robert S.Balog, "Topics In DSP: Interpolation & Delta Sigma Quantization", *Prentice Hall*, 1996.
- [13] Peter Kiss, Jesus Arias, Dandan Li, and Vito Bocuzzi, "Stable High-Order Delta-Sigma Digital-to-Analog Converters", *IEEE transactions on Circuits and Systems*, Vol. 51, no.1, Jan 2004.

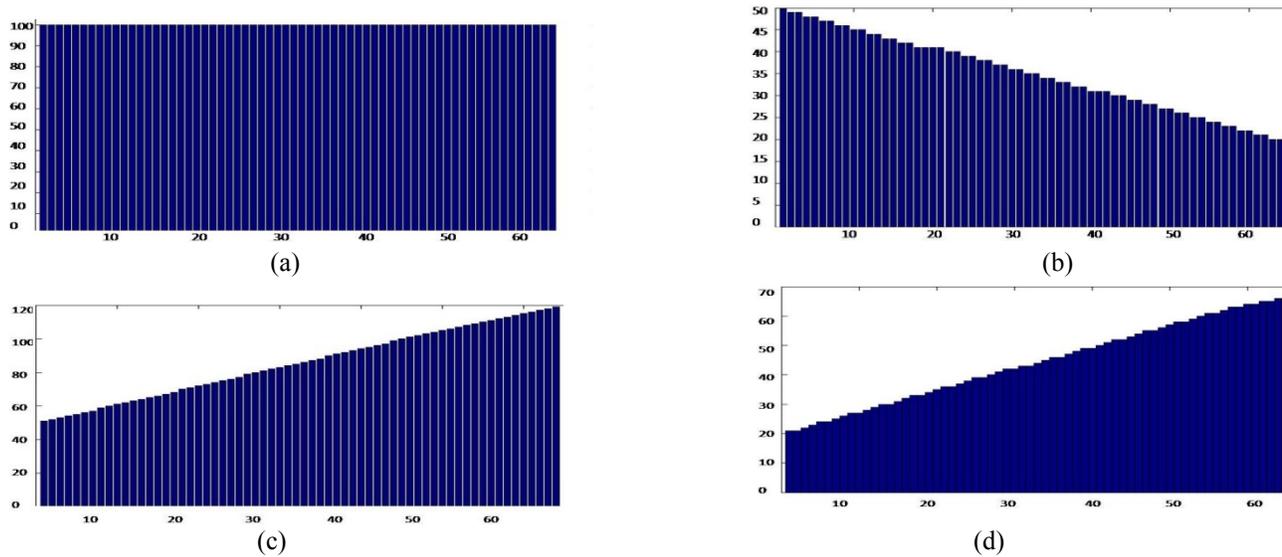


Fig. 4, Matlab simulation results for the interpolation filter. a) $\Delta=0$, no interpolation required, all output samples are equal. b) $p>q$, $N_{Min} > N_{Maj}/2$; *Patn1* is used. c) $p>q$, $N_{Min} < N_{Maj}/2$; *Patn2* is used. d) $p<q$.

Magdy A. El-Moursy was born in Cairo, Egypt in 1974. He received the B.S. degree in electronics and communications engineering (with honors) and the Master's degree in computer networks from Cairo University, Cairo, Egypt, in 1996 and 2000, respectively, and the Master's and the Ph.D. degrees in electrical engineering in the area of high-performance VLSI/IC design from University of Rochester, Rochester, NY, USA, in 2002 and 2004, respectively. In summer of 2003, he was with STMicroelectronics, Advanced System Technology, San Diego, CA, USA. Between September 2004 and September 2006 he was a Senior Design Engineer at Portland Technology Development, Intel Corporation, Hillsboro, OR, USA. During September 2006 and February 2008 he was assistant professor in the Information Engineering and Technology Department of the German University in Cairo (GUC), Cairo, Egypt. Dr. El-Moursy is currently Staff Engineer in the Mentor Graphics Corporation, Cairo, Egypt. His research interest is in Networks-on-Chip, interconnect design and related circuit level issues in high performance VLSI circuits, clock distribution network design, and low power design. He is the author of more than 30 papers, four book chapters, and one book in the fields of high speed and low power CMOS design techniques and high speed interconnect.

Ahmed Abdellatif was born in Cairo, Egypt, in 1986. He received his B.Sc. from the German university in Cairo, Egypt in 2008, M.Sc. from the University of Ulm, Germany in 2010. From 2008 to 2009, he worked as a teaching assistant in the German university in Cairo. From 2009 to 2010, he worked as a student scientist in university of Ulm, on amplifiers for Retinal implants. Currently, he is working on his Ph.D. in Friedrich-Alexander University Erlangen-Nuremberg, where his main research point is designing MMIC's for broadband operation up to 100 GHz for radar and medical applications.