Microelectronics Journal 51 (2016) 112–118

Contents lists available at ScienceDirect

Microelectronics Journal

journal homepage: www.elsevier.com/locate/mejo

Direct computation for high performance interpolation filter

Ratshih S. Abd El-Azeem [a], Magdy A. El-Moursy [a,b,*], Amin M. Nassar [c], Ahmed Gharib [d,e], Nahla T. Abou-El-Kheir [f], Moataz S. El-Kharashi [g]

[a] Microelectronics Department, Electronic Research Institute, Cairo, Egypt
[b] Mentor Graphics Corporation, Cairo, Egypt
[c] Electronics Department, Cairo University, Cairo, Egypt
[d] Institute for Electronics Engineering, Friedrich-Alexander-University of Erlangen–Nuremberg, 91058 Erlangen, Germany
[e] EESY-IC GmbH, 90449 Nürnberg, Germany
[f] University of Ottawa, Ottawa, Canada
[g] Department of Electron Devices, University College Southeast Norway, Norway

ARTICLE INFO

Article history:
Received 15 May 2015
Received in revised form
18 October 2015
Accepted 20 January 2016
Available online 25 March 2016

Keywords:
Digital interpolation
Interpolation factor
Digital low pass filter
FIR
CSHM
DSP

ABSTRACT

A Computational Filter (*CF*) that employs a sample calculation functional block is presented. *CF* significantly reduces the hardware requirements to realize an interpolation filter[2]. The Computational Filter is compared with advanced Finite Impulse Response (*CFIR*). *CFIR* programmable filter is implemented using a Computation Sharing Multiplication (CSHM) technique to optimize the conventional design. The proposed *CF* significantly reduces the implementation area as compared to *CFIR*. The maximum average error for wide range of interpolation factors from 8 to 256 is 2.59% for *CF* and 0.02% for *CFIR*. Xilinx Virtex5 FPGA XC5VTX240T device is used to compare the proposed design and the advanced *CFIR* for interpolation factor of 8, 16, 32, and 64. The Computational Filter average reduction in hardware implementation is 78.8%, 89.5%, 94.5%, and 97.1% for interpolation factor of 8, 16, 32, and 64, respectively. The maximum operating frequency for the *CFIR* is 1 MHz. The maximum operating frequency is 0.92 MHz, 0.57 MHz, 0.32 MHz, and 0.2 MHz for interpolation factor of 8, 16, 32, and 64, respectively.

1. Introduction

Many applications in digital signal processing (DSP) nowadays need high-performance and low-power VLSI systems. Digital interpolation filters are widely used in DSP applications such as Delta-Sigma based digital to analog converters (Δ-$\sum$ DAC). Conventional interpolation filters use up-sampling and digital low pass filter (LPF) to obtain high precision [1–3]. The interpolation process is used to increase the number of samples of the input signal by an interpolation factor (L). This process is performed by inserting ($L-1$) uniformly spaced zero values between the sampled data followed by low pass filter [4]. Digital low pass filter requires large number of components such as multipliers, adders, and delay units [5]. Discrete Fourier Transform (DFT) and Fast Fourier Transform (FFT) are used to reduce the cost-needed to implement the filter [6,7]. This reduction leads to high performance as well as low power design. Low pass filter algorithm use either Finite Impulse Response (FIR) [8] or Infinite Impulse Response (IIR) [9]. For interpolation, the FIR filter is appropriate algorithm due to its stability. For low power design a transposed direct form (TDF) FIR filter realization is proposed in [10]. Digital data could be interpolated using different interpolation techniques. Nonlinear interpolation needs storing input samples according to the polynomial order [11]. That requires large Memory and hardware to achieve high accuracy. On the other hand, linear interpolation requires only two samples that reduces the hardware [12]. As shown in the paper a dramatic reduction in hardware could be achieved with insignificant scarify in accuracy.

In this paper a complete design for a new interpolation filter using direct computation *CF* is presented. The design is multiplier free. A direct sample calculation is used to achieve significant hardware implementation reduction. Also a comparison between a *CF* technique and an advanced FIR using CSHM (*CFIR*) design [13] is presented. *CF* is shown in Fig. 1(a) and *CFIR* as shown in Fig. 1(b).

The paper is organized as follows. In Section 2, the Computational Filter (*CF*) technique is presented. In Section 3, an enhanced FIR design using CSHM (*CFIR*) is described. In Section 4, simulation results and comparison between the two techniques are provided.

* Corresponding author.
E-mail addresses: eng_ratshih@yahoo.com (R.S. Abd El-Azeem), magdy_el-moursy@mentor.com (M.A. El-Moursy), amin.nassar@yahoo.com (A.M. Nassar), ahmed.radwan@eesy-ic.com (A. Gharib), nabou081@uottawa.ca (N.T. Abou-El-Kheir), moataz@ieee.org.com (M.S. El-Kharashi).

http://dx.doi.org/10.1016/j.mejo.2016.01.006
0026-2692/

In Section 5 FPGA implementation is presented. Conclusion is summarized in Section 6.

2. Computational Filter

In this section, the proposed *CF* is presented. The operation of the *CF* is described in Section 2.1. The technique requires determining the majority and the minority samples which is presented in Section 2.2. The filter implementation is discussed in Section 2.3.

2.1. Computational Filter operation

The *CF* calculates output samples $Y(n)$ to avoid using digital low pass filter to reduce hardware implementation. Computational filter perform linear interpolation that needs two successive samples of the input data sequence $X(m)$ and $X(m+1)$. The main blocks of the *CF* are illustrated in Fig. 2.

The difference between two successive integer input samples D is determined in step calculation block,

$$D = X(m+1) - X(m). \tag{1}$$

Assuming $X(m)$ is integer, $Y(n)$ is expected to be an integer possessing the same precision of $X(m)$. The proposed technique is directly extended to fixed point representation. For linear interpolation, D and the interpolation factor L are used to get the step that given by

$$\text{Step} = (D/L) \tag{2}$$

Assuming fixed point representation and normalizing all bits to the available integer numbers of bits, the output samples are calculated. The step could be fraction number, but the output is integer number. In the ideal case the output sequence should be incremented by the step, so the ceiling M_1 or the floor M of the step are used [12],

$$M = floor(D/L), \tag{3}$$

$$M_1 = M + 1. \tag{4}$$

The number of output samples which are incremented by floor of the step M is r. The number of output samples which are incremented by ceiling of the step M_1 is s, where,

$$r + s = L, \tag{5}$$

$$M*r + M_1*s = D. \tag{6}$$

Given D and L, r and s are determined by solving (5) and (6),

$$r = M_1*L - D, \tag{7}$$

$$s = D - M*L. \tag{8}$$

Eqs. (3), (4), (7), and (8) are the equations needed to realize the Computational Filter.

The output of the step calculation block are M, M_1, D and Zero Check. To guarantee the uniform distribution of the calculated samples, the output samples are distributed based on two shapes *shape*1 and *shape*2 as shown in Table 1. The distribution depends on the value of r and s. The majority output samples are the maximum of r and s. The minority output samples are the minimum of r and s,

$$\mathrm{M_{maj}} = \mathrm{Max}(r, s), \tag{9}$$

$$\mathrm{M_{min}} = \mathrm{Min}(r, s). \tag{10}$$

The outputs of "Calculate Step" block are used to determine M_{maj} and M_{min} through "Calculate M_{maj} and M_{min}" block, M_{maj} and

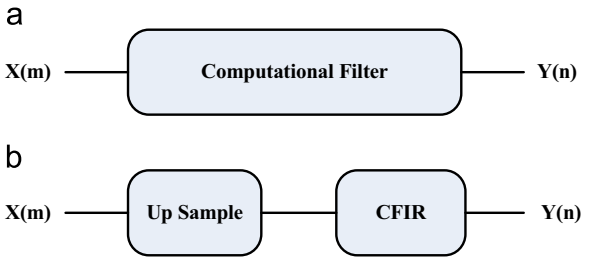

Fig. 1. (a) Multiplier free technique and (b) up-sample and *CFIR* filter.

Table 1
The distribution of the minority and majority in the output samples (a) shape1 (b) shape2.

(a)				
Sample index	***0***	***1***	***2***	***L***
Output samples	y_{maj}	y_{min}	y_{maj}	y_{min}
(b)				
Sample index	***0***	***1***	***2***	***L***
Output samples	y_{maj}	y_{maj}	y_{maj}	y_{maj}

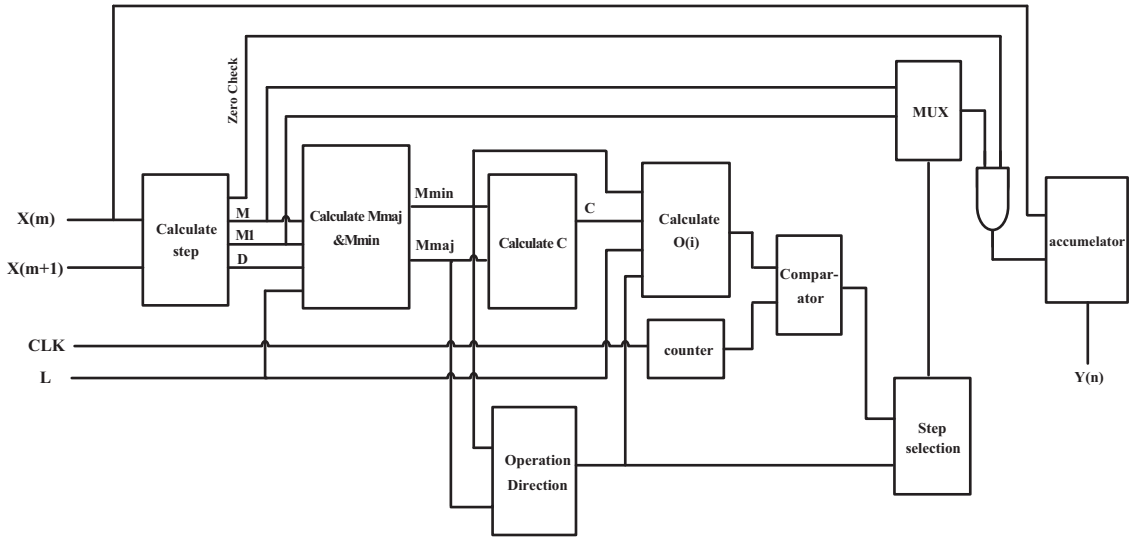

Fig. 2. The architecture of Computational Filter.