Efficient and Expandable Interpolating FIR Filter Design and Implementation

A. CHOREVAS, D. REISIS
Physics Department
University of Athens
Panepistimiolopis, Physics Building IV & V
GREECE

Abstract: - This work describes the design and implementation of an interpolating digital filter that is placed at the input of a Digital to Analog Converter (DAC) to relax the characteristics of the analog filter following the DAC. The work presents techniques of parallelizing the filter computations leading to the design and implementation of efficient architectures with respect to throughput and VLSI area. Furthermore, this design involves techniques which facilitate expansion with respect to the filter length (taps) and internal and external accuracy as well as adaptivity to various sampling rates.

Key-Words: - digital filter, interpolation, DAC, FIR, parallel. CSCC'99 Proceedings, Pages:6791-6795

1 Introduction

Digital to Analog Converters require output low-pass analog filters to eliminate the undesired high frequencies produced by the conversion. Other systems, such as the modulator part of a telecommunication system, use the output analog filters also as shapers for the produced pulses [1]. Most often, the researchers designing DACs, telecom systems, etc. require analog filters with characteristics which lead to expensive realizations [2,3].

An attractive solution to the problems imposed by the analog filter realization is to increase the input rate of the DAC in conjunction with an interpolating DAC-input digital filter. This can relax the analog filter's requirements which then can be easily implemented [2,4].

In digital audio reproduction systems, such as CD players, an efficient digital filter design can further improve the system's performance as it can eliminate the need of the analog filter. This can be achieved by increasing the sampling rate. For example an x128 increase in the basic rate of 44.1 Ksamples /sec will produce a 5.6 Msamples/sec output. Any commercial audio amplifier will act as a low-pass filter for such high frequencies so there is no need to use a distinct analog filter that will affect the quality of the signal. A second use in audio frequency systems is in sigma-delta modulation converters [2].

Another application field for the interpolating filters is the modems of communication systems. Here, the digital filter not only relaxes the constraints of the antialiasing analog filter but also implements part or all of the pulse shaping transfer function that is required for the best performance of the modem [1].

The numerical interpolation starts with inserting zeros between the real values (real samples). This higher rate signal feeds a Finite Impulse Response (FIR) filter. This method can preserve the values of the initial samples by appropriate selection of the impulse response of the filter [2]. Typically, the impulse response of the FIR filter is a function with a sin(x)/x form.

An interpolating filter can be implemented in a microprocessor specially designed for Digital Signal Processing (DSP processor) or in specific hardware (VLSI or FPGA). In a FIR filter only the repetitive execution of Multiply/Accumulate (MAC) instructions is required. So a custom VLSI design that embeds a DSP-core is inefficient as most of the processors capabilities are not used. Furthermore, there are designs were one DSP processor cannot handle the full set of computations to accommodate the desired rate. Another drawback of DSP processors is that when the required accuracy (word length) exceeds the standard number representation of the processor, the number of operations increase dramatically as the MAC operation has to be

performed involving number fragmenting techniques[6]. This leads to reduced throughput.

Direct VLSI implementation of a FIR filter (interpolating or not) typically involves replication of the basic multiplication unit in order to achieve the maximum throughput[4]. These designs require larger area for both the additional multipliers and for the storage of the intermediate results.

This work proposes a direct VLSI implementation which is well suited for optimizing VLSI area with respect to sampling rate, word length and filter's length. The design uses a regular structure and makes extensive use of paralelization and pipelining in both word and bit level in order to achieve the best performance.

Our test-case consists of a x128 oversampling filter for audio frequencies. The filter is working with input words of 16 bits, coefficient words of 18 bits, internal (partial result) and output words of 32 bits, input sampling rate 44.1 KHZ and 16 real input samples. These values produce a 5.6 MHZ output.

This work is organized as follows: Section 2 describes the FIR filter design, section 3 presents the results in conjunction with some applications, and finally section 4 concludes the work.

2 Interpolating FIR filter design

The data come with an input rate Fin while the filter produces data with output rate Fout. Let us call the rate Fout/Fin as “oversampling” rate and denote it with the letter V. The interpolation process is accomplished in two steps. In the first step (V-1) zero values are inserted between two consecutive input samples. In the second step the data are passed through a FIR filter with impulse response of type sin(x)/x.

2.1 Input data processing

Let us start with K consecutive values of input X and denote them

$$X_m, X_{m-1}, \ldots\ldots , X_{m-(K-1)}$$

where m denotes time instances which correspond to the input rate Fin. Let us now insert (V-1) zeros between any two consecutive values of these. The sequence becomes:

X_m,0,0,...0, X_{m-1}, 0,0,...0, ... , $X_{m-(K-1)}$, 0,0,...0

The last sequence is now rewritten using n to denote time instances which correspond to the output rate Fout.

X_n,0,0,...0, X_{n-V}, 0,0,...0, ... , $X_{n-V(K-1)}$, 0,0,...0

It is clear that most of the data that feed the FIR filter is zero.

2.2 FIR filter Design

The FIR filter calculates the output y at time n according to the equation :

$$y_n = \sum_{i=0}^{M-1} b_i \cdot x_{n-i} \qquad (1)$$

where x_i is the input of the filter at time i, M is the number of taps (or filter coefficients) and b_i is the i-th coefficient of the filter ($0 \le i < M$). Most of the x_i involved in equation (1) are zero. Let K represent the number of input samples used (as in the above paragraph). Then M is connected with K through the following equation:

$$M = V \cdot K \qquad (2)$$

When the values of V and M are high, commercial chips usually cascade filters with lower interpolating rates in order to reduce the complexity. Cascading filters is far from being the optimal solution due to the increased noise it produces.

For the test case considered in this work the coefficients are produced from the sampling of function sin(x) / x. A window function is also used [5] eventhough the differences between various window functions are small for this application.

The coefficients are digitized at the required accuracy and represented as fixed-point values. Simulations showed that representing the coefficients with 18 bits give excellent results in out test-case. An extra bit is used in the representation to compensate for the possible "overshoot" produced by the interpolation, and further to improve negative-number handling within the multiplier units. The same procedure is followed to produce the coefficients of any function which can be realized by the filter.

2.3 Implementation

When high rates are employed in FIR filtering, distributed computation is required (fig.1). This leads to designs that need extra registers to store the intermediate results. The present work deals with filters that not only perform in high rates but also have a large number of taps. In our test-case M=2048 so 2K words are used only to store the coefficients. A design facilitating distributed computations requires another 2K words to store the intermediate results.

The symmetry of the coefficients can lead to "folded" designs [4] that use half the storage and

calculate half of the multiplications, but the final VLSI area remains high for the applications considered in this work. Moreover, for interpolating filters with V > 2, the "folded" designs are not very efficient as the main factor that allows the reduction of the computational power is the presence of a large number of zero input values.

From equations (1) and (2) it is clear that the calculations for each output value Y require only K multiplications. So it is more efficient, with respect to VLSI area, to perform the calculations directly for each output Y instead of distributing the calculations using architectures such as these presented in fig. 1. Serial implementation of equation (1) results in the minimal area but leads to low throughput designs.

The block diagram of our design is presented in fig.2. It consists of S stages (S<K). Each stage employs P=K/S multiplications for each output Y and uses a separate multiply/add unit. Each stage keeps P of the K input samples (X_m) in a cyclic shift register. These P values feed the first input of the multiplier. The multiplier is working in a rate Fw which is P times greater than the output rate Fout. A ROM in each stage keeps the respective coefficients and feeds the second input of the multiplier at the working rate Fw. The adder unit is located at the output of the multiplier. The control circuit instructs the adder to accumulate P of the produced values and add them to the result Y from the stage on the right. The Yin of the rightmost stage has always zero value. The complete architecture requires only S registers to hold the intermediate results while keeping all the benefits of the paralelization.

The multiplier is a fully bit parallel array multiplier [fig. 3]. The maximum speed is determined by the time needed for a simple addition. Bitwise pipelining in the addition can be easily employed to achieve the maximum throughput [7].

The regularity of the design makes it easily implementable to multi-chip solutions when off the self components are used. In our test case the filter has been implemented in FPGA chips from XILINX [8] and ALTERA [9].

3 Results and applications

A VHDL description with the appropriate utilities has been produced in order to ensure the proper function of the FIR filter under various values for parameters V, K and the word length. The VHDL description makes the design portable to any VLSI platform.

The test-case of this work is a DAC for commercial CD players. It consists of an interpolating filter with he characteristics that are described in sections 1 and 2, and a parallel DAC. A FPGA chip (e.g. the EPF10K130V from ALTERA) includes the interpolating filter and the interface to the digital output of the CD player. The filter has 32-bit wide words output. The Most Significant Bits (MSB) of the filter's output feed the actual Digital to Analog Converter which has a tree structure (fig. 4). The design of the DAC is still in progress. Expandability and adaptivity have been tested so far by timing simulations.

Another application area is modems. The design presented in this work can work in Discrete Multitone (DMT) transceiver in order to improve the analog output. DMT is used in Asymmetric Digital Subscriber Line (ADSL) systems [3] (fig. 5).

4 Conclusion

This paper has presented techniques to design and implement FIR filters for improved high speed modem and audio reproduction applications. The resulting architecture has been shown to be efficiently expandable with respect to filter length as well as data and coefficient word length. Further, it can accommodate high throughput requirements and use optimal storing area.

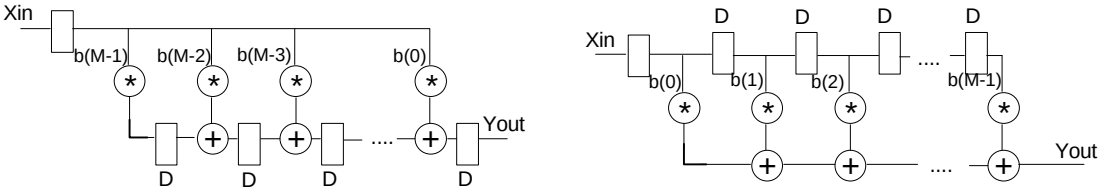

Fig. 1: FIR filters using distributed computations.

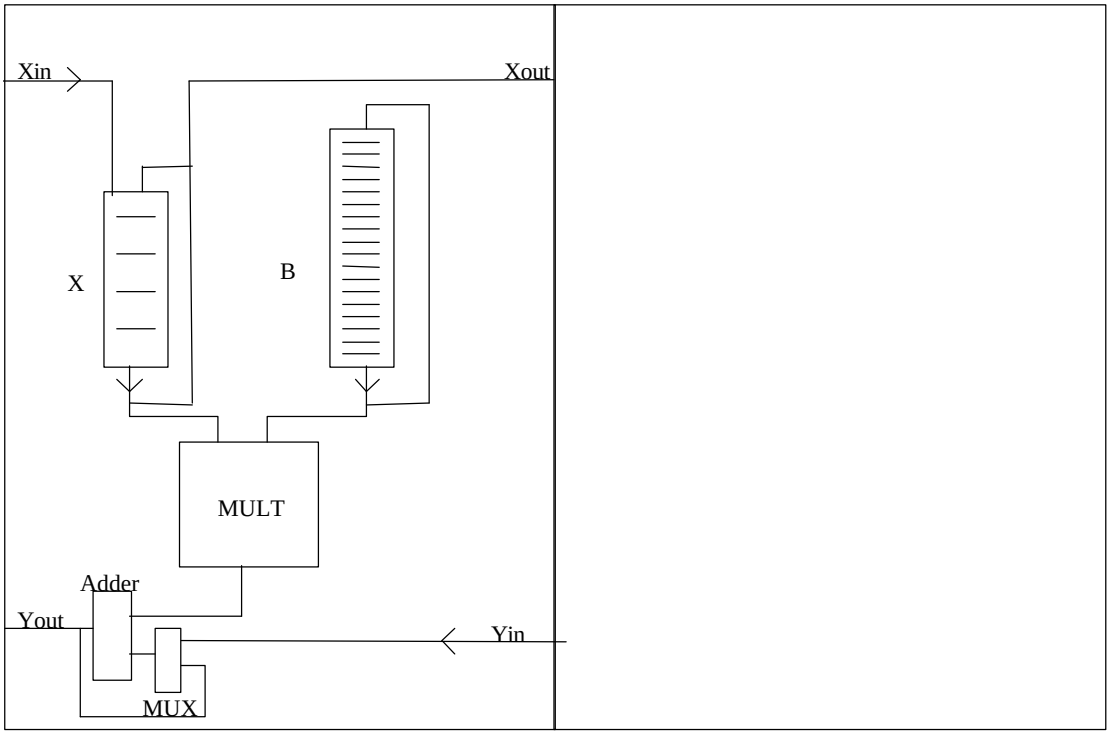

Fig. 2: Block diagram of the interpolating FIR filter.

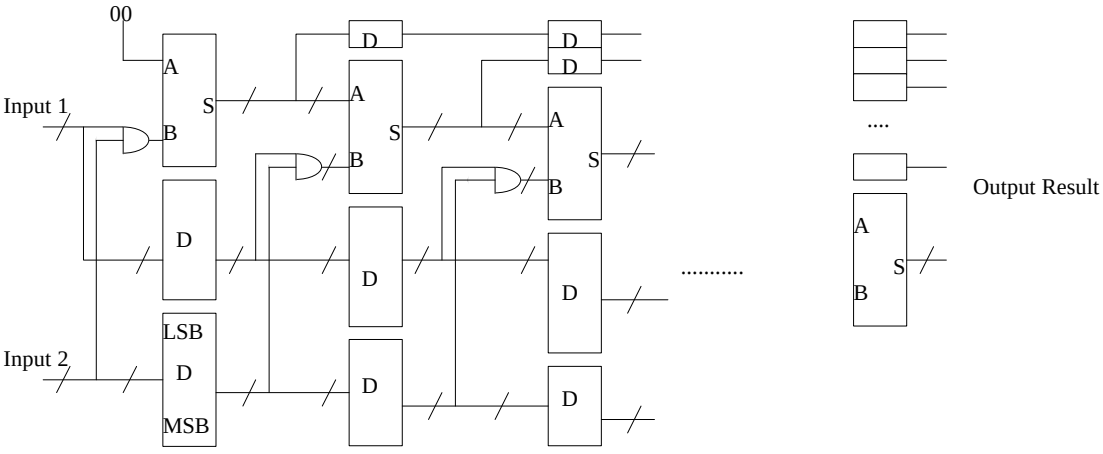

Fig. 3: Pipelined multiplier.

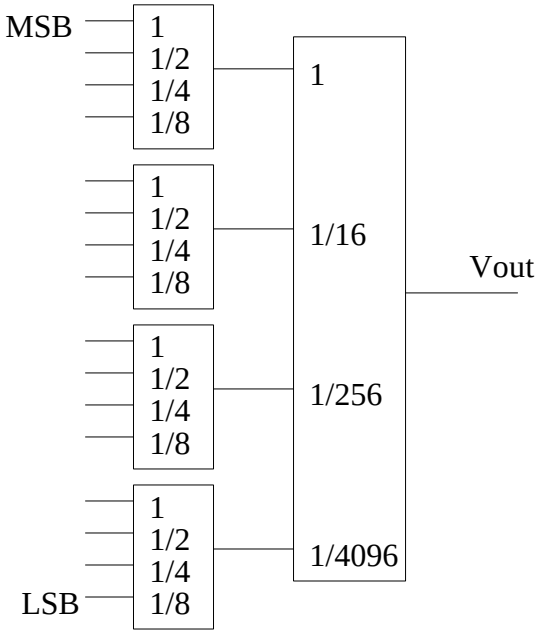

Fig. 4: Block diagram of a parallel DAC using tree-structure.

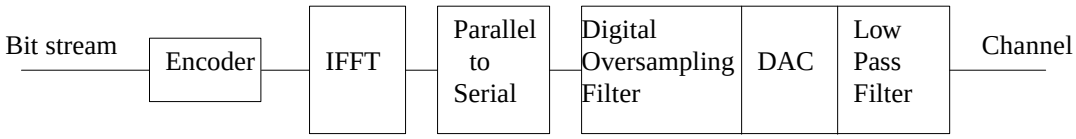

Fig. 5: Block diagram of a DMT transmitter.

References:

[1] J. Proakis, *"Digital Communications"*, McGraw Hill, 1995.

[2] K. Pohlmann, *"Principles of Digital Audio"* 2nd Edition, Howard Sams & Company, 1989.

[3] J. Chow, J. Tu and J. Cioffi, "A discrete multitone transceiver system for HDSL applications*", IEEE J. Select. Areas Commun.*, vol. 9, no. 6, 1991, pp.895-908.

[4] A. Chorevas, D. Reisis, E. Metaxakis, "An Efficient Digital FIR Filter Design for 64 QAM", *Proceedings of IEEE ICECS 1996*, Vol.2, 1996, pp.900-903.

[5] A. Oppenheim and R. Schufer, *"Digital Signal Processing"*, Prentice Hall, 1975.

[6] K. Hwang, *"Computer Arithmetic"*, Jhon Wiley & Sons, 1979.

[7] F. T. Leighton, *"Parallel Algorithms and Architectures: Arrays-Trees-Hypercubes"*, Morgan Caufmann, 1992.

[8] XILINX, *"4000 databook"*, 1997

[9] ALTERA, *"FLEX 10K databook"*, 1998