Direct Generation of Upsampled FIR Filter Response

A Simple Extension to Filters with Distributed Arithmetic

Zeynep Kaya
Dept. of Electrical&Electronics Eng.
Bilecik Seyh Edebali University
Bilecik, Turkey
zeynep.kaya@bilecik.edu.tr

Erol Seke
Dept. of Electrical&Electronics Eng.
Eskisehir Osmangazi University
Eskisehir, Turkey
eseke@ogu.edu.tr

***Abstract*—A memory based upsampling/interpolating FIR filter modification/extension to distributed arithmetic (DA) based FIR filters is proposed that can be used for any filter coefficient set. Use of minimum or no multiplier is a desired design property when signal processing is performed using FPGAs since multipliers are scarce/expensive resources within FPGAs whereas registers and such are abundant. Upsampling a digital stream is usually performed by inserting zeros between original samples followed by a low pass filter to reject images. Compared to basic distributed arithmetic based filter designs where partial products/sums are stored in memory blocks, our design stores interpolation values. These samples are output sequentially using a simple counter, eliminating zero insertions and saving circuit elements. As an example FIR filter, we have designed a raised-cosine band-limiting filter with example roll-off factor and upsampling values. Successful implementation using VHDL+FPGA with ease has proven that the approach is a simple and effective compared to input upsampling.**

Keywords—FIR; FPGA; upsamling; raised-cosine;

I. INTRODUCTION

Digital signal processing is an important component of digital communication systems and filters are one of the most common elements in digital signal processing. Increasing use of FPGAs in digital systems and SDR designs make it necessary to develop signal processing/filtering techniques and optimize digital component use in terms of available primitives within FPGA architecture. Arithmetic elements like adders and multipliers are vastly needed complex primitives and reducing the number of them in digital designs is probably the main objective of the design optimizations, along with the throughput. In this paper, we propose an extension to distributed arithmetic (DA) based FIR filter approach that reduces the number of taps at the input for upsampled filter response. It requires no multipliers and minimum count of adders, instead, our extension uses storage elements that are available in high counts in FPGA architectures.

Studies on use of LUTs instead of multipliers are quite common [1-4]. Memory based multiplier-free filter architectures are commonly based on distributed arithmetic (DA) structure proposed very earlier [5]. In DA, partial products within direct-form FIR filter convolution (1) are placed in a storage area and called back with the time-skewed input bits. Although equivalent in the generated circuit, we approach the problem differently and treat the filter as a linear combination of multiple single bit filters, which makes the DA and proposed extension easily understandable. The actual filter impulse response is stored in the memory which is shared among the bits of the input samples. The advantages of this approach are;

- Simple
- Keeps “no multipliers, single accumulator” of DA
- Easy to generate upsampled filter response
- Saves logic elements
- Readily used in parallel filtering of a single input

II. IMPULSE RESPONSE OF A FIR FILTER

The response of Finite Impulse Response filters to a signal with finite duration, as the name implies, diminishes in finite duration determined by the designed filter length, mimicking Infinite Impulse Response (IIR) filter counterpart in that duration and becomes zero afterwards. Finite response which is an approach that simplifies initial design duration completely eliminates the stability problem which, for IIR filters, should be analyzed very carefully.

The response of a FIR filter to an impulse input can be written as a sequence of finite duration of N samples

$$c_i,\ i = 0,1,...,N-1. \tag{1}$$

For linear and time invariant FIR filters, the general response of a filter to a discrete sample sequence can be written in a form of convolution as

$$y[n] = \sum_{i=0}^{N-1} c_i x[n-i] \tag{2}$$

where $x[.]$ is the sequence of input samples.

Expressing input samples in the form of the linear sum of bit values as

978-1-4799-3303-7/14/$31.00 ©2014 IEEE

$$x[n]=\sum_{b=0}^{B-1}x_b[n]2^b \tag{3}$$

where B is the number of bits and $x_b[n]$ is the b^{th} bit value of the n^{th} sample, (2) becomes

$$y[n]=\sum_{b=0}^{B-1}2^b\sum_{i=0}^{N-1}c_i x_b[n-i] \tag{4}$$

.

That is, $y[n]$ can be obtained by a weighted sum of responses of filters given as

$$y_b[n]=\sum_{i=0}^{N-1}c_i x_b[n-i]\ ,\ b=0...B-1 \tag{5}$$

where $y_b[n]$ is the response of the filter to single bit input stream. $y_b[n]$, depending on the design requirements, is B' bits where B' is not necessarily be the same with B. This is the approach of DA.

Fig. 1 illustrates a filter response to a single bit input stream. Here, $y_b[n,t]$ is the possible continuous impulse response to a single bit with finite duration T_F. Possible continuous output would be the sum of these responses, whereas discrete output samples are the sums of individual responses at sample times. Note that when viewed in this manner, output sample rate does not need to be the same as the sample rate of the input $(1/T_b)$ and can be at any rate. This is the upsampled response approach used here in this paper. The upsampling filter response can be formulated by

$$y_b[nK+k]=\sum_{i=0}^{N-1}c_{iK+k}x_b[n-i] \tag{6}$$

where K is the upsampling ratio and k is the index of the interpolated samples.

In the illustrative example given in Fig. 1, it is obvious that the output waveform (discrete or continuous) within 2<t<3 is determined only by 3 input samples; $x_b[n]$, $x_b[n+1]$ and $x_b[n+2]$, so the number of possible waveforms is limited by 2^N where N is 3 in this example. Therefore, these waveforms can be stored and the appropriate one can played out from memory when needed.

In some applications, the output of the filter response, consequently the impulse response of the filter, needs to be upsampled, that is, for each input samples, multiple output samples are generated. Given an upsampling ratio K and $n=\text{integer}(k/K)$, (5) can be rewritten as

$$y'_b[k]=\sum_{i=0}^{NK-1}c_i x'_b[k-i]\ ,\ b=0...B-1 \tag{7}$$

.

It is easily seen that the number of operations (circuit components) increases by K.

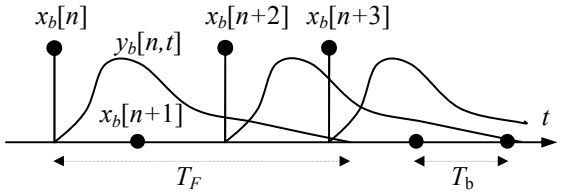

Fig. 1. Individual responses to bits in "101100…" b^{th} bit stream.

One of the applications that use upsampling is the use of band-limiting filters on binary stream before or after the baseband modulation in communications systems. Raised-cosine filter [7,8] is one of the filters used for this purpose due to its features that need not be listed here. Our example design uses raised-cosine filter for demonstration purposes and its discretized impulse response for roll-off factor of 0.3 and oversampling ratio of 5 is given in Fig. 2. Although actual filter response is infinite, it is truncated to 51 samples as the rest of the samples are too small, a customary approximation for FIR filters. Further modifications and approximations are sometimes necessary when additional hardware limitations apply during implementations.

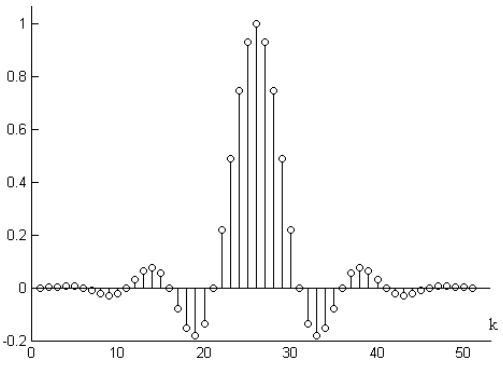

Fig. 2. Impulse response of the raised-cosine filter used in example design.

Upsampling FIR filter output for a single bit stream illustrated in Fig. 1 is generated as follows;

Samples of the possible waveforms are pre-stored in an addressable memory. Since the filter is N-tap (output depends only on previous $N-1$ input bits and the current bit) there exist only $M=2^N$ possible waveforms, K samples each (subparts from impulse response illustrated in Fig. 2), where $K\geq 1$. Initial addresses of partial waveforms are addressed via a serial-in parallel-out shift register. The rest of the samples

are played out by addressing the low address bits through a simple counter counting 0 to $(K-1)$ within T_b. Given the possible waveforms

$$W_m[\text{reminder}(k/K)], m = 0...M-1 \quad (7)$$

.

the mechanism is simply illustrated in Fig. 3.

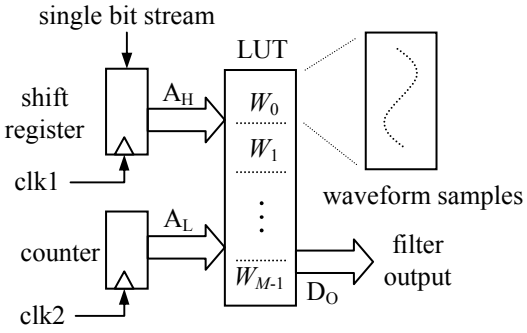

Fig. 3. Single bit FIR generator for the method illustrated in Fig. 1.

It is possible to improve storage efficiency and have some simplifications by choosing K such that $K = 2^R$ where R is a positive integer, but not necessary.

For an NK-tap filter with input upsampling of K, the number of input registers/selectors would be NK, whereas it is N when output upsampling is employed using waveform playing approach.

The waveform samples W_m are actually K samples extracted from the middle of the $M = 2^N$ possible filter responses as illustrated in Fig. 4. In Fig. 4, all possible filter responses for the input bit streams 00000 through 11111 are displayed for the $N = 5$ tap raised-cosine FIR filter previously mentioned (Fig. 2).

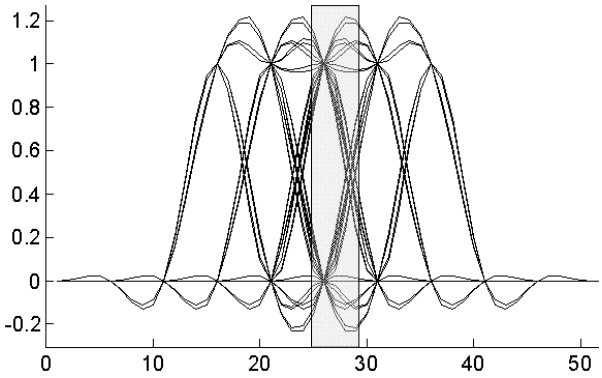

Fig. 4. All possible responses for the filter given in Fig.2 given input streams 00000 through 11111. Initial and final states are assumed 00000. However, marked values of the waveforms are independent of the initial conditions of the filter.

For clarity, three of the waveforms in Fig.4 are given separately in Fig. 5, representing the input streams 10000, 01000 and 11000. When the previous shift register content was 10000, the current content can be either 01000 or 11000. The waveforms that correspond to these inputs follow the waveform corresponding 10000 as expected and required.

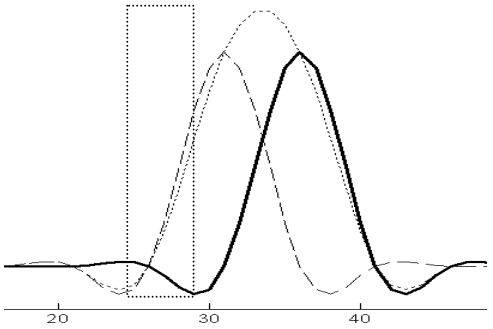

Fig. 5. 5-tap raised-cosine filter response to input bit streams 10000, 01000 and 11000.

III. EXTENSION IN DA FIR FILTER

Equipped with the upsampling extension, the memory in DA filter no longer contains only partial products but patches of the FIR filter responses to single bit stream given in Fig. 4. Waveform samples are identical with the ones in single input bit filter illustrated in Fig. 3, but time shared among input bits. Therefore, the frequency of bit shifting clock is B times the frequency of sample selection counter clock (clkL in Fig. 6). Consequently, responses of the bit filters needs to be accumulated for the whole response per sample input per waveform sample, making the clock frequency at the accumulator the same as the bit shifting clock. Accumulator register needs to be cleared at the end of every such cycle, and the calculation repeats for the new value of the counter. The design is shown in Fig. 6.

In our test designs, instead of the long bit shifter at the input stage (shown in Fig. 6), we used a $N \times B$ 2 dimensional shift+rotate register capable of both shifting vertically and rotating horizontally, generating identical result.

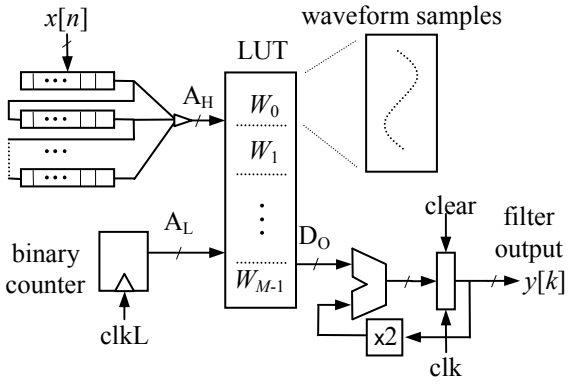

Fig. 6. Proposed upsampling FIR filter structure with one adder.

978-1-4799-3303-7/14/$31.00 ©2014 IEEE

Multiplication by 2 in the output stage represents 2^b term given in (4). Although the output stage looks somewhat complicated in schematics, it is very straightforward in hardware description languages (HDL) and requires an adder and a resettable register. Multiplication by 2 does not use any resource since it is just a shift in wires. In order to relate the clock frequencies, let the input sample frequency be f_x. The counter clock frequency then is Kf_x and bit shift frequency is BKf_x. Consequently, the accumulator frequency is BKf_x and it is cleared at a rate of Kf_x.

Algorithmic flow of the proposed design is as follows;

```
for each sample x[n]
  shift down x[n] into first B-bit of SR
  for AL=0 to K-1
    using AL as sample number address
    for b=0 to B-1
      using AH as memory start address
      Do=retrieve word from memory
      ytemp=2x ytemp+Do
    next b
    y=ytemp (output the acc content)
    ytemp=0 (clear accumulator)
  next AL (start processing next bit)
next sample
```

IV. CONCLUSION

We applied random binary stream to the raised-cosine FIR filter designed using VHDL+FPGA+DAC set and analyzed the output to see if the desired filter output is generated. The reason for testing the filter with a binary signal is one of the features of the raised-cosine filter; it generates the exact values at the sample points when upsampled, therefore it is trivial to verify the test results. Two example portions of the binary stream and generated filter output corresponding to this inputs is given in Fig. 7. We compared this output with the MATLAB results and confirmed that they match exactly within the quantization error bounds.

Parallel filtering of a single input stream is quite economic using this approach. In that case, memory block should have wider storage to accommodate coefficients of two filters in parallel. Two accumulator blocks are required for these memory outputs as illustrated in Fig. 8.

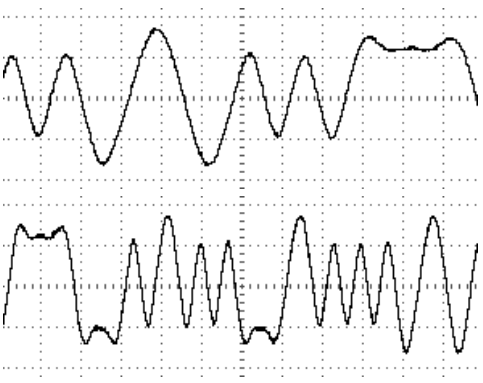

Fig. 7. Output signals generated when the input streams are a) 10100110010101111 b) . 1111000101101010001101010100 1100. The filter's roll-of factor is 0.5 and output is upsampled 5 times.

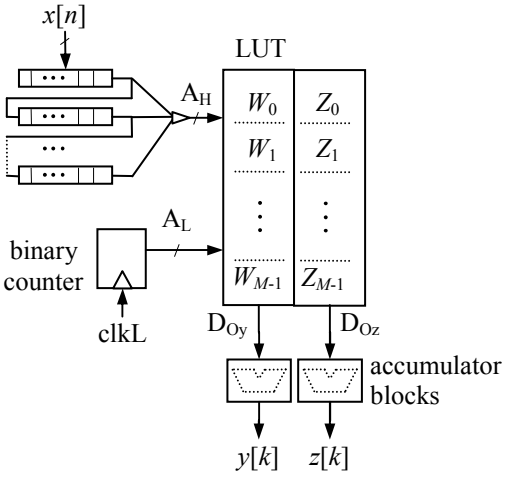

Fig. 8. Two parallel filters for a single input stream.

V. REFERENCES

[1] Peled, A. and Bedei L., "A New Hardware Realization of Digital Filters", IEEE Trans. Acoustics, Speech and Signal Proc., Vol 22. p456-462 no. 6, 1974.

[2] Mirzaei, S., Hosangadi, A. and Kastner, R. "FPGA Implementation of High Speed FIR Filters", Proc. IEEE Int'l Conf. On Computer Design, p308-313, San Jose, 2007.

[3] Meher, P. K., Chandrasekaran, S., and Amira, A. "FPGA Realization of FIR Filters by Efficient and Flexible Systolization Using Distributed Arithmetic", IEEE Trans. Signal Proc., Vol. 56, no 7, p3009-3017, 2008.

[4] Zhao, Q. and Tadokoro, Y. "A simple design of FIR filters with powers-of-two coefficients", IEEE Trans. Circuits Syst., Vol. 35, pp. 566-570, 1988

[5] L. Mintzer, "FIR Filters with Field-Programmable Gate Arrays", Journal of VLSI Signal Processing, 1993, vol.6, pp.19-127

[6] Presti, L. L. "FIR Design of Raised-Cosine Filters", Proc. of IEEE European Conf. on Area Comm., Stockolm, 1988.

[7] Presti, L. L., Mondin, M. "Design of Optimal FIR Raised-Cosine Filters", IEEE Electronics Letters, Vol. 25 (7), p467-468, 1989.