A New Architecture for Decimating FIR Filter

Tanee Demeechai and Siwaruk Siwamogsatham
National Electronics and Computer Technology Center, Pathumthani, Thailand
{tanee.demeechai, siwaruk.siwamogsatham}@nectec.or.th

Abstract

A new architecture for decimating finite impulse response filters is proposed. The architecture is based on using a number of accumulators; each one accumulates a partial sum corresponding to a unique set of D filter coefficients into the filter output, where D is the decimation factor. In the new decimating filter, the accumulated result of an accumulator is passed to another accumulator once for each period of D input samples, except for that of the last accumulator whereby the filter output is obtained. The size of each accumulator can be minimized, depending on the filter coefficients. A demonstrative FPGA implementation shows that this architecture is more favorable than the widely used polyphase architecture, as it requires much less area at similar power consumption.

Keywords: *decimating FIR filter, signal processing, polyphase filter*

1. Introduction

Decimating filters have found their use in various applications [1]. In this paper, we are dealing with an application of a decimating finite impulse response (FIR) filter in prototyping a multi-standard radio on a field-programmable gate array (FPGA)-based software-defined radio platform. An illustrative multi-standard radio is to provide communication based on IEEE 802.11n [2] or IEEE 802.15.4 [3] using a common radio-frequency (RF) front-end connected to an FPGA, where the RF front-end performs frequency up/down conversion and signal-type conversion from analog type to digital type and vice-versa. In treating the difference on channelization of the two standards, it is desirable to simplify the RF front-end by designing it to accommodate only the communication mode of largest bandwidth, i.e., the 40-MHz mode of the IEEE 802.11n, and letting other modes be accommodated with assists from digital signal processing (DSP). Therefore, the RF front-end employs a direct-conversion (zero-IF) architecture where the complex lowpass signal is sampled at 40 Msamples/s. A decimating FIR filter becomes demanded in DSP when receiving the IEEE 802.15.4 signal, which has bandwidth much lower than 40 MHz, through this RF front-end. This decimating FIR filter is required to be linear-phase, having frequency response matched to the spectrum of the desired signal, and accommodating the receiver jamming resistance requirement of the IEEE 802.15.4 standard.

Filter architectures relevant to our purpose are that of [4] and [5], while other architectures [6-8] focus on only decimation, i.e., they are not designed for implementing a filter with arbitrary impulse response. A widely used decimating FIR filter is based on the polyphase decomposed FIR filter architectures [4]. For a decimating filter with impulse response of length equal to N times the input-sampling duration and a decimation factor equal to D, the polyphase filter requires D parallel FIR filters operating at the output-sampling clock rate for a computational effort of scaling N successive input samples and summing the N scaled values once every D input samples. In this paper, we propose a new architecture for decimating FIR filter that contains equivalent computational effort to that of the polyphase

filter but using computational units of more reuse. Hence, achieving less gate-count requirement and less static power consumption but similar dynamic power consumption compared to the polyphase filter may be expected from the new architecture.

2. New Filter Architecture

In one point of view, the new filter architecture is related with a decimating FIR filter architecture of [5], where summing the N scaled values for an output sample is done using a single accumulator. Hence, the architecture of [5] will be described in detail first. Let L denote the smallest integer that is divisible by D and not less than N. The architecture of [5] as depicted in Figure 1 uses $M = L/D$ accumulators in computing the output samples according to a round-robin schedule, while it requires no more memory elements to store any other input-related values.

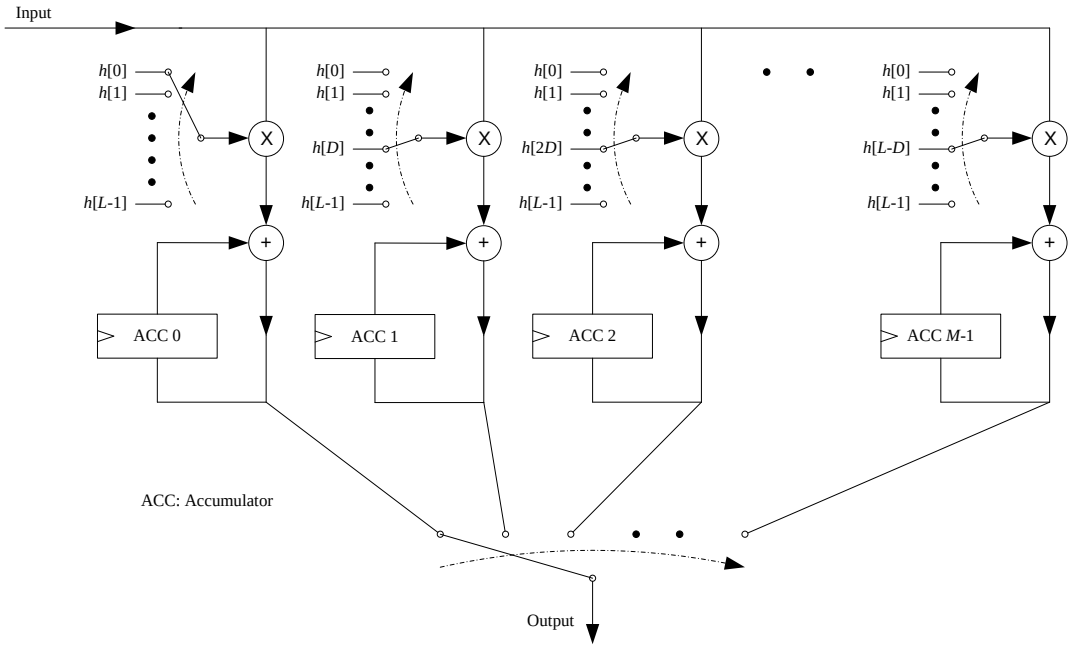

Figure 1. Decimating FIR Filter Architecture of [5]

To understand the underlining schedule, first consider that the filter output sample at (input-sampling) time instant nD is given by

$$y[n] = \sum_{k=0}^{L-1} h[k]x[nD-k], \tag{1}$$

where $x[\cdot]$, $y[\cdot]$, and $h[\cdot]$ are respectively the input signal, the output signal, and the filter impulse response (subjected to zero padding if $L>N$). This merely shows that $y[n]$ is obtained by summing the values of $x[k]$'s for k from nD-L+1 to nD, which are scaled according to the fixed filter impulse response. Accordingly the summing for $y[m]$ using a single accumulator is started by first accumulating $h[L-1]x[mD-MD+1]$ and stopped

after $h[0]x[mD]$ has been accumulated. In addition, when $y[m]$ has been output, the underlining accumulator can be reset and then used in another summing that first accumulates $h[L-1]x[mD+1]$ and so on until $h[0]x[mD+MD]$ has been accumulated, i.e., the summing for $y[m+M]$. Namely, the output samples that are equally spaced by M output-samples are obtained by regularly operating a single accumulator. Hence, to regularly obtain all the filter output samples, it requires M accumulators to operate regularly in parallel with equally shifted timing of D input-samples. It can then be noted that all selector switches shown in Fig. 1 are operated in a round-robin fashion. The scaling selectors switch once every new input sample, while the output selector switches once every new output sample.

Similarly to the architecture of [5], the new architecture also employs M accumulators to compute all the filter output samples. However, each accumulator in the new architecture accumulates only a partial sum corresponding to a unique set of D filter coefficients into the filter output. In this regard, the accumulated result of an accumulator is passed to another accumulator once for each period of D input samples, except for that of the last accumulator whereby the filter output is obtained. Accordingly, the new architecture may be shown as in Figure 2, where the selector switches are operated as follows. Firstly, the scaling selectors are operated in a round-robin fashion, they switch once every new input sample, and they are synchronized, i.e., they switch to their top lines at the same time. Secondly, the binary selectors switch to their top lines only when a scaling selector switches to its bottom line - the same time when a filter output sample is available at the filter output port. As an illustration, the accumulator contents are shown against time for the case that $M = D = 3$ and $x[n]=0$ when $n < 0$ in Table 1, from which it should be noted that ACC 0 contains the required filter output value when the binary selectors switch to their top lines.

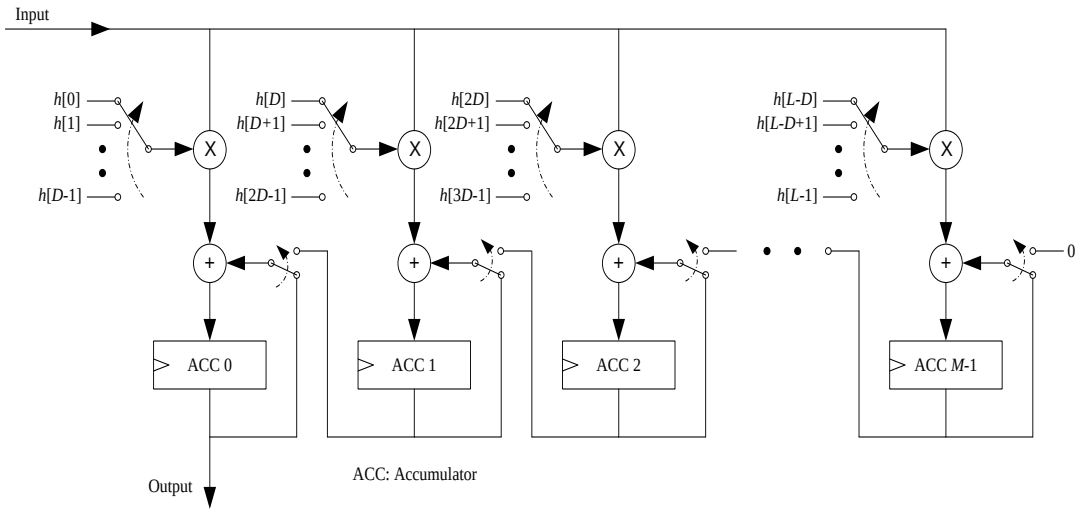

Figure 2. New Decimating FIR Filter Architecture

The new architecture should be a better choice of implementation compared to either the architecture of [5] or the polyphase architecture, due to the following reasons. In comparison with the architecture of [5], it is clear that the new architecture requires less

resource for the selector switches. In addition, it requires in most cases smaller accumulators and associated adders as measured in term of word size. Denote the word size in bits of the input sample as W. Then it may be noted that the word size of each accumulator in [5] would be $W+\log_2\left(\sum_{k=1}^{L-1}|h[k]|\right)$ bits, while the word size of the m-th accumulator of the new architecture (denoted as ACC m in Fig. 2) would be $W+\log_2\left(\sum_{k=mD}^{L-1}|h[k]|\right)$ bits. It should be clear that the new architecture is generally better than that of [5].

Table 1. Accumulator Contents versus Input-sampling Time Instants for *M*=*D*=3 (shaded rows are when the binary selectors switch to their top lines.)

n	ACC 0	ACC 1	ACC 2
0_+	0	0	0
1_+	$h[0]x[0]$	$h[3]x[0]$	$h[6]x[0]$
2_+	$h[2]x[1]$ $+h[3]x[0]$	$h[5]x[1]$ $+h[6]x[0]$	$h[8]x[1]$
3_+	$\sum_{k=1}^{3}h[k]x[3-k]$	$\sum_{k=4}^{6}h[k]x[6-k]$	$h[7]x[2]$ $+h[8]x[1]$
4_+	$\sum_{k=0}^{3}h[k]x[3-k]$	$\sum_{k=3}^{6}h[k]x[6-k]$	$\sum_{k=6}^{8}h[k]x[9-k]$
5_+	$\sum_{k=2}^{6}h[k]x[6-k]$	$\sum_{k=5}^{8}h[k]x[9-k]$	$h[8]x[4]$
6_+	$\sum_{k=1}^{6}h[k]x[6-k]$	$\sum_{k=4}^{8}h[k]x[9-k]$	$h[7]x[5]$ $+h[8]x[4]$
7_+	$\sum_{k=0}^{6}h[k]x[6-k]$	$\sum_{k=3}^{8}h[k]x[9-k]$	$\sum_{k=6}^{8}h[k]x[12-k]$
8_+	$\sum_{k=2}^{8}h[k]x[9-k]$	$\sum_{k=5}^{8}h[k]x[12-k]$	$h[8]x[7]$
9_+	$\sum_{k=1}^{8}h[k]x[9-k]$	$\sum_{k=4}^{8}h[k]x[12-k]$	$h[7]x[8]$ $+h[8]x[7]$
10_+	$\sum_{k=0}^{8}h[k]x[9-k]$	$\sum_{k=3}^{8}h[k]x[12-k]$	$\sum_{k=6}^{8}h[k]x[15-k]$
11_+	$\sum_{k=2}^{8}h[k]x[12-k]$	$\sum_{k=5}^{8}h[k]x[15-k]$	$h[8]x[10]$
12_+	$\sum_{k=1}^{8}h[k]x[12-k]$	$\sum_{k=4}^{8}h[k]x[15-k]$	$h[7]x[11]$ $+h[8]x[10]$
13_+	$\sum_{k=0}^{8}h[k]x[12-k]$	$\sum_{k=3}^{8}h[k]x[15-k]$	$\sum_{k=6}^{8}h[k]x[18-k]$

In comparison with the polyphase architecture, it should be noted that the new architecture contains equivalent computational effort to that of the polyphase architecture, i.e., scaling N successive input samples and summing the N scaled values once every D input samples. Hence, the two architectures should achieve similar dynamic power consumption. In addition, because the adders in the new architecture have much more degree of being reused, the new architecture should achieve less area requirement and less static power consumption. However, based on numerical results comparing them about an FPGA implementation described in Section 3, it is found that the new architecture utilizes less area with similar power consumption.

3. Numerical Results

We have obtained numerical results about FPGA implementation of the new architecture, based on tools from Xilinx [9]. The underlining filter is supposed to work at input sampling frequency of 40 Msamples/s, the input word size is 10 bits, and $D = 5$. The filter impulse response ($N = 42$) is [1 2 4 6 8 10 12 14 16 18 20 22 24 25 27 28 29 30 30 31 31 31 31 30 30 29 28 27 25 24 22 20 18 16 14 12 10 8 6 4 2 1], designed for use when receiving the IEEE 802.15.4 signal from an RF front-end for the 40-MHz mode of the IEEE 802.11n. Each selectable scaler, shown in Fig. 2 as comprising of a multiplier and a selector switch, is implemented by connecting five fixed scalers with a multiplexer. Each fixed scaler is enabled only when needed to pass the multiplexer. Each binary selector switch is implemented by a multiplexer. According to the mentioned impulse response, there are nine accumulators and the word sizes of ACC 0, ACC 1, .., and ACC 8 are respectively 20, 20, 20, 20, 19, 19, 18, 16, and 12 bits.

For comparison, we have also obtained numerical results of the polyphase architecture. In the polyphase filter, the five decimated input signals required to feed the five component filters are obtained by using a multi-bit serial-in-parallel-out (SIPO) shift register. Each component filter also employs a multi-bit SIPO shift register as in a direct-form FIR filter, whereas the outputs of the shift registers of all the component filters are used in the computation of each output sample. The computation of each output sample exploits the fact that many samples of the impulse response have the same value, i.e., input samples that are to be equally scaled according to the impulse response are added before being scaled.

By using Xilinx ISE 10.1 design platform targeting the XC5VLX30-3FF324 device with default settings and a filter description based on VHDL [10], the slice utilization of each implementation is obtained automatically from the Map step, and the estimation of power consumption is obtained using the XPower Analyzer with post-route simulation data. As shown in Table 2, the numbers of slices utilized by the new and polyphase architectures are respectively 291 and 345. The new architecture requires lesser number of slices by about 15%.

Table 2. FPGA Implementation Results

Power consumption (mW)	New architecture			Polyphase architecture		
	Zero input	Sawtooth input	Random input	Zero input	Sawtooth input	Random input
Static	300.81	301.33	301.51	300.87	301.29	301.45
Dynamic	8.88	17.07	19.86	9.89	16.41	18.89
Total	309.69	318.40	321.37	310.76	317.70	320.34
Slice utilization	291 slices			345 slices		

Actually, we expected that the new architecture would require very much lesser number of slices, because in computing each output sample about D nontrivial additions are done by a single adder in the new architecture while each addition in the polyphase architecture requires a dedicated adder. Note that, in this demonstrative case, the polyphase architecture requires equivalently 41 dedicated two-input adders and 18 fixed scalers as computational components, while the new architecture requires 9 two-input adders (with average size slightly higher than that of the polyphase architecture) and 18 fixed scalers. However, the whole computation per output sample in the polyphase architecture including 41 additions and 18 scalings is performed by a single large combinatorial circuit that could benefit most from a powerful combinatorial-circuit

optimization of the Xilinx ISE platform. Therefore, instead of multiply less slice utilization, the 15% less slice utilization of the new architecture could be reasonable.

The power consumption results are also shown in Table 2 for three types of filter input: zero signal, sawtooth signal, and white noise. The zero signal is supposed to cause minimum filter activity, while the white noise, as it is generated by a fully-spanning uniform random number generator, is supposed to cause maximum filter activity. It may be noted from the results that both the static and dynamic power consumptions of the new architecture are smaller than that of the polyphase architecture when the input signal is zero. This is because when the filter is at minimum activity the power consumption is highly governed by the slice utilization. However, when the input type is changed from zero to non-zero, the dynamic power consumption of the new architecture is increased at faster pace than that of the polyphase architecture. The static power consumption is also similarly increased with the filter activity, because the dynamic power consumption increases the junction temperature. However, the overall power consumptions of the two architectures do not appear to be significantly different, because (i) the overall power consumption seems to be dominated by the static power consumption, (ii) the static power consumptions of both architectures are similar, and (iii) the dynamic power consumptions of both architectures are not much different.

Given the slice utilization results, the above static power consumption results could be reasonable. In addition, consistent with the reporting results, we have also expected that both architectures have similar dynamic power consumptions. Note that, although the combinatorial-circuit optimization of the Xilinx ISE platform could significantly reduce the slice utilization of the polyphase architecture, the computational burden per output sample must still be maintained to that including 41 additions (done by equivalently 41 adders with intermediate results propagating within the combinatorial circuit) and 18 scalings (done by equivalently 18 fixed scalers). Note also that the computational burden per output sample of the new architecture includes 41 additions (done by 9 adders with some intermediate results stored in registers) and at most 40 scalings (but done by 18 fixed scalers). As a fixed scaler is much computationally simpler than a variable scaler (i.e., a multiplier), the computational burden per output sample of the new architecture should be only slightly higher than that of the polyphase architecture for nontrivial input signals. Considering that the total number of logical transitions within a circuit during a time interval is proportional to the computational burden of the circuit during that time interval and realizing that dynamic energy dissipation is proportional to the total number of logical transitions, it is reasonable that the dynamic power consumption of the new architecture is only slightly higher than that of the polyphase architecture.

Although the obtained numerical results do not show that the new architecture is better than the polyphase architecture in both power consumption and area utilization features, the new architecture still appears to utilize less area with similar power consumption.

4. Conclusion

This paper proposes a new architecture for decimating FIR filters, where the computational units consist of a number of linked accumulators. Compared to the widely used polyphase architecture, the new architecture embodies computational units with much higher degree of being reused in performing the same computational effort. A demonstrative FPGA implementation shows that the new architecture utilizes less area with similar power consumption.

References

[1] P. P. Vaidyanathan, "Multirate Digital Filters, Filter Banks, Polyphase Networks, and Applications: A Tutorial", Proc. IEEE, Vol. 78, pp. 56-93, **(1990)**.

[2] IEEE Standard for Information Technology -- Telecommunications and Information Exchange between Systems -- Local and Metropolitan Area Networks -- Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput, IEEE Standard 802.11n-2009 **(2009)**.

[3] IEEE Standard for Information Technology -- Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs), IEEE Standard 802.15.4-2006, **(2006)**.

[4] M. Bellanger, G. Bonnerot and M. Coudreuse, "Digital filtering by polyphase network: Application to sample rate alteration and filter banks", IEEE Trans. Acoust. Speech Signal Proc., Vol. 24, pp. 109-114, **(1976)**.

[5] K. Kawamoto, T. Kengaku, E. Teraoka, T. Oga and H. Ishida, "Decimating digital finite impulse response filter", US patent 5191547, **(1993)**.

[6] B. P. Brandt and B. A. Wooley, "A low-power, area-efficient digital filter for decimation and interpolation", IEEE Journal of Solid-State Circuits, Vol. 29, pp. 679-687, **(1994)**.

[7] E. B. Hogenauer, "An economical class of digital filters for decimation and interpolation, IEEE Trans. on Acoustics, Speech and Signal Proc., Vol. 29, pp. 155-162, **(1981)**.

[8] H. Aboushady, Y. Dumonteix, M. M. Louerat and H. Mehrez, "Efficient polyphase decomposition of comb decimation filters in ΣΔ analog-to-digital converters", IEEE Trans. on Circuits & Systems – II, Vol. 48, pp. 898-903, **(2001)**.

[9] www.xilinx.com.

[10] B. Cohen, "VHDL coding styles and methodologies", Springer, **(1999)**.