Design of Parallel-Form Digital Filters

Vikas Paliwal and Dr. Chung-Kuan Cheng (Course Professor)

***Abstract*— Recent advancements in field programmable gate array (FPGA) technology offer significant advantages in terms of reduction of word lengths used in calculations. This necessitates determining the bit lengths of each of the calculated variables and coefficients so that a simpler implementation can be realized as compared to fixed word length digital filters. Parallel-form digital filters are preferred in variety of scenarios due to their direct relation with poles and accuracy in terms of placing the poles very close to the unity circle. This study addresses the need for calculating the necessary bit widths needed for realizing a desired level of accuracy in parallel from digital filters based on analysis and simulations and study of similar research efforts towards direct-form digital filters. We provide the word lengths needed to avoid the overflow and loss of accuracy in realizing the filters. Finally we show the accuracy offered by choosing sufficiently wide word lengths as opposed to fixed word length designs.**

I. Introduction

DIGITAL filters are used in wide variety of applications due to their simplicity and flexibility in implementation. Their behavior can easily be manipulated based on their mathematical formulation. Recent advancements in field programmable gate arrays (FPGAs) over traditional digital signal processing (DSP) offer accurate and fast design choices for digital filters. However, these advancement also pose several new design challenges in choosing the bit-lengths used in representation of calculated variables and coefficients. This is needed to make the fullest use of the design flexibility offered by variable bit widths in FPGAs.

Fixed point calculations are of greater importance in signal processing applications. On the other hand, floating point calculations - although come with greater precision, accuracy and a wider dynamic range - require complex hardware implementation and are not as cost effective as fixed-point processors. While the dynamic range of fixed point numbers can be addressed by use of more bits in representation, but an optimal design requires using only sufficient number of bits for the sake of representation of variables [1].

In this work, fixed-point parallel-form digital filter design is presented. Minimum number of bits needed to avoid overflows and achieve a desired level of accuracy is presented. This is based on existing research on direct form filters [4], [7] and signal processing theory [2]. We further illustrate by means of simulations and analysis that desired accuracy can be attained by using the bounds presented in this work. Extension of the parallel-filter problem to generic pole placements is also presented.

II. Fixed-point arithmetic

Fixed point representation of numbers has a fixed number of decimal points before and after the radix point, thereby making the radix point location as fixed. Compared to the floating numbers, whose radix location varies depending on the required level of precision, fixed point representation has static design. This results in reduced dynamic range of numbers [7] that can be represented by fixed point numbers. Figure 1 shows the fixed point number representation with 16 bits. 11 bits are used for fractional part, 4 bits for the integer part and 1 bit for the sign. This results in a dynamic range of numbers from $-2^4 = -16$ to $2^4 - 2^{-11} = 15.99951171875$.

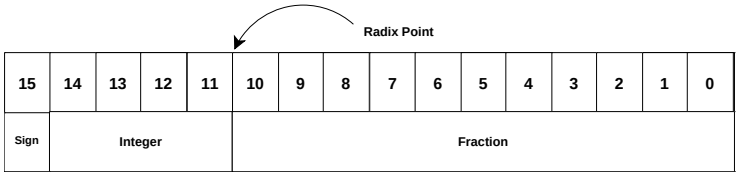

Fig. 1. Fixed point number representation

Fixed point arithmetic in digital signal processors (DSPs) involves using a constant register lengths for carrying out mathematical operations and hence a single fixed point is used throughout the calculations. FPGAs, in contrast to DSPs, allow separate fixed point format for each of the signals and hence offer greater savings in widths of numbers and simplicity of implementation. Figure 2 illustrates how for a single multiplication of a number with a fixed coefficient, FPGAs require fewer number of bits. It is clear that for multiplication of an 8-bit number with a 3-bit coefficient, savings can be achieved in number of bits used for representing both the coefficient and result. Clearly, FPGAs offer a greater scope of optimizing the hardware design and hence lesser power consumption.

Even though FPGA based design is more optimized, it poses a new requirement for designing the length of each of signals [4], [6]. To harness the full potential of flexibility in choosing the bit lengths of signals, it is mandatory to determine the relation between the bit-length and the level of accuracy required. In this reference, this study seeks to give estimate for minimum number of bits required to achieve a desired level of accuracy and avoid the overflow in signal operations. In brief, with FPGAs, every system needs to be optimized uniquely based on its behavior to achieve desired stability and precision.

III. Parallel Form Digital Filters

Digital filters perform the filtering operations on incoming samples based on mathematical relations. Such mathematical operations need to be implemented in the hardware in an optimal fashion. Filters can be represented in several forms - cascade, parallel, direct among others. The actual hardware implementation of a filter is closely tied to the representation used for the filter. In general, the question of choosing a

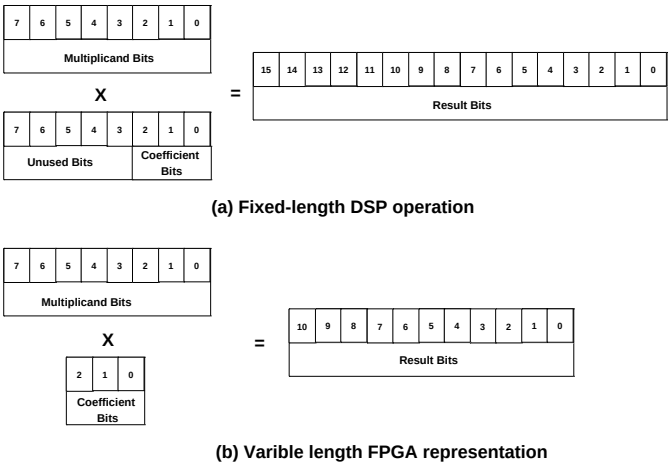

Fig. 2. Number representation in DSPs and FPGAs

suitable form can be resolved by the requirements of a particular scenario. Each of the representations offer advantages in particular situations. Further, it is well known from signal processing theory that performance of a filter is closely tied to its pole/zero locations and their proximity to the unity circle in the z-plane. Thus accuracy of mathematical operations is key to the performance of the filter. The infinite impulse response (IIR) filters [8] have non-zero response to an impulse function and in the most general form can be represented in an auto-regressive moving average form as,

$$y[n] = \sum_{m=1}^{N} a_m y[n-m] + \sum_{m=0}^{N} b_m x[n-m]. \quad (1)$$

The z-transform for such an IIR filter is,

$$H(z) = \frac{b_0 + b_1 z^{-1} \cdots b_j z^{-j} \cdots + b_M z^{-M}}{1 + a_1 z^{-1} \cdots a_k z^{-k} \cdots + b_N z^{-N}}. \quad (2)$$

Parallel form of digital filters translates to a conversion to partial fraction terms of the type,

$$H(z) = \sum_{m=1}^{N} \frac{R_m^{'}}{1 - p_m z^{-1}}. \quad (3)$$

Such parallel form filters are fairly easy to implement when the order of pole location is more important than the precise pole location. In these cases, choosing binary poles of the form, $p_m = (1 - \frac{1}{2^{k_m}})$, is very easy to realize in hardware as it only involves addition, subtraction, multiplication with residues and bit shifts. This eliminates the need for the costly multiplications in the loop sections and even costlier divisions. However if precise pole locations are indeed needed, multiplications need to be performed in the feedback loop as discussed in later sections. Figure 3 shows a realization of the parallel-form digital filter with binary poles and $R_i = R_i^{'}(1-p_i)$. This design only involves addition and bit-shifting and is therefore very easy to implement in hardware.

Despite the advantages offered by parallel form digital filters, the filter response precision is impacted by finite word length effects. Compared to a floating point representation with a large dynamic range, fixed point implementation results

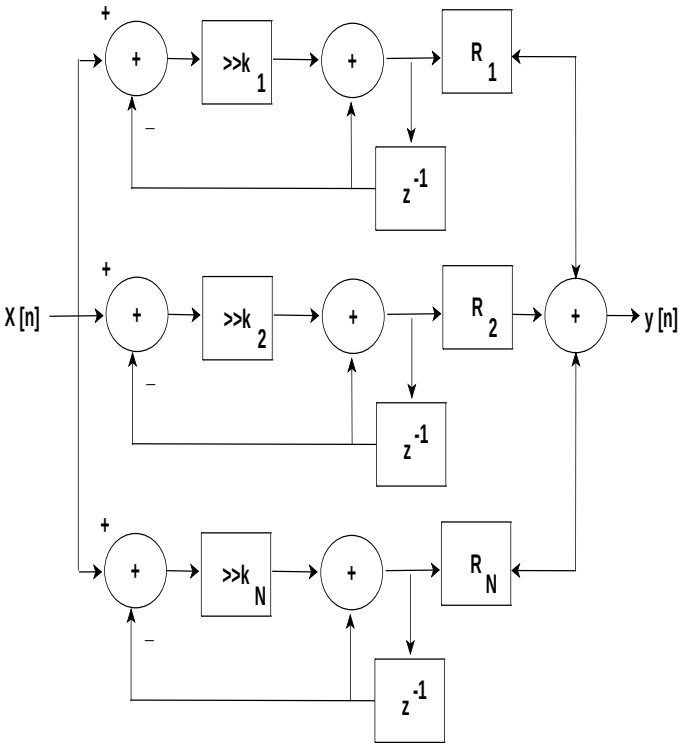

Fig. 3. Parallel-form digital filter with binary poles

in truncation errors and possible overflow. While overflow problems are directly related to the precision used in choosing the representation, truncation errors get compounded over and over again by being fed to the filter and are therefore difficult to model. As an example for truncations, in our filter example in Figure 3, each right shift by k bits results in loss of information by same amount.

IV. Residue bit width calculations

The residue values of the parallel-form filter need to be represented in suitable form. This entails choosing the width of the integer part as well as the fractional part in fixed point notation. While the integer part can be trivially chosen as $(\lceil log_2 R_m \rceil + 1)$. The fractional part needs to be derived based on the filter characteristics and desired level of precision in the the zeroes of the filter [4]. If the system is tolerant to larger variations in the zeroes of the filter, we can do with away with fewer bits and vice versa. In a more rigorous form, for instance, a two-pole filter,

$$H(z) = \frac{R_1}{1 - p_1 z^{-1}} + \frac{R_2}{1 - p_2 z^{-1}} \quad (4)$$

has a zero at

$$\frac{R_1 p_2 + R_2 p_1}{R_1 + R_2}. \quad (5)$$

For a permissible perturbation δ in the zero location, assuming other coefficients and poles being accurate, the perturbation in residue R_1, ΔR_1, is related as,

$$\delta = \frac{(R_1 + R_2)(\Delta R_1 p_1) - (\Delta R_1)(R_1 p_2 + R_2 p_1)}{(R_1 + R_2)^2} \quad (6)$$

$$\Rightarrow \Delta R_1 = \frac{\delta(R_1+R_2)^2}{R_2(p_2-p_1)} \tag{7}$$

So, for a permissible perturbation δ in the zeroes of the filter, the minimum number of fractional bits required to achieve the desired accuracy would be,

$$\lceil -log_2(\frac{\delta(R_1+R_2)^2}{R_2(p_2-p_1)}\rceil + 1 \tag{8}$$

Similar analysis can be developed to calculate the fraction width with an n-pole transfer function.

V. Overflow avoidance

Each of the calculated intermediate signals, need to have enough number of bits in the integer part of representation so that overflow is prevented. The problem with intermediate signals is that they are dependent on the filter response as well. Knowing the upper bound on the values attained by the signals requires an understanding of the filter performance. In this regard, known results from DSP theory [3] on the upper bounds of the intermediate and output signals of filters come handy. The idea behind estimating the upper bound primarily comes from seeing the transfer function from the input signal, $x[n]$, to the target signal, $v[n]$, i.e. $h_{v,x}[n]$ and calculating the upper bound achieved by this transfer function if perpetually fed with the maximum possible value of input signal, x_{max}.

$$|v_{max}| \leq \sum_{k=0}^{\infty} h_{v,x}[k]|x_{max}|. \tag{9}$$

Based on this relation and the value of maximum possible signal value, the number of bits required for representing the integer part is simply $\lceil log_2|v_{max}|\rceil + 1$. The procedure for calculating the upper bound in parallel form filters is very simplified because of representation in pole-residue forms. This is due to the fact that the inverse z-transform of parallel form filter in eqn. (3) can be very easily obtained as opposed to a direct form representation. For instance, the output signal in parallel form

$$h[n] = Z^{-1}\{H(z)\} = \sum_{m=1}^{N} R_m p_m u[n] \tag{10}$$

and the upper bound on the output signal is simply,

$$|y_{max}| \leq \sum_{k=0}^{\infty} h[k]|x_{max}| \tag{11}$$

$$\Rightarrow |y_{max}| \leq \sum_{k=0}^{\infty}\sum_{m=1}^{N} R_m p_m u[k]|x_{max}| \tag{12}$$

$$\Rightarrow |y_{max}| \leq |\sum_{m=1}^{N} \frac{R_m}{1-p_m}||x_{max}| \tag{13}$$

So, the number of bits required to avoid overflow in the output signal is simply, $\lceil log_2|\sum_{m=1}^{N}\frac{R_m}{1-p_m}||x_{max}|\rceil + 1$. This result was verified with a single pole IIR filter and it was confirmed that the bound of $\lceil log_2|\frac{R_1}{1-p_1}||x_{max}|\rceil + 1$ is necessary and sufficient to avoid overflow under all circumstances.

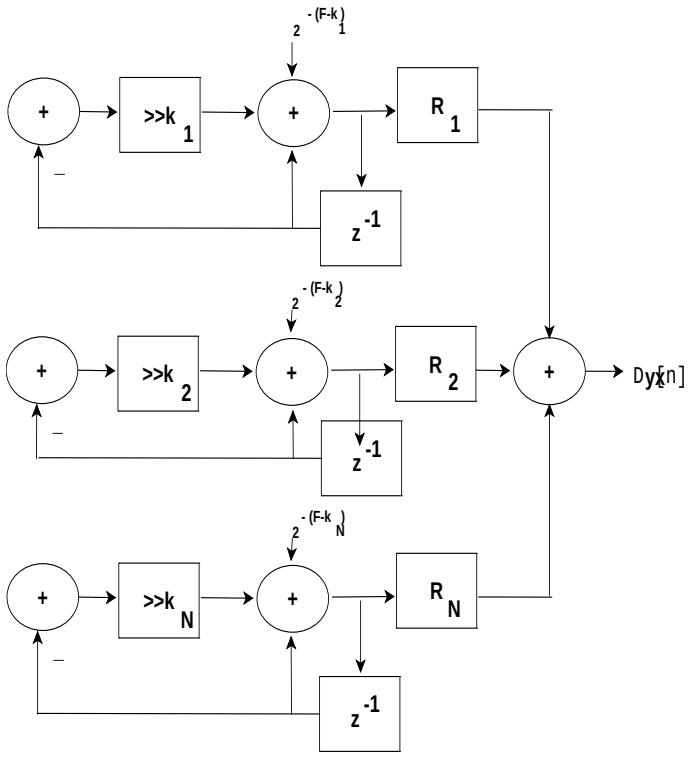

Fig. 4. Modeling the truncation errors

VI. Truncation Errors

It can be understood that right bit shifts used to achieve division operation in the filter relations in eqn. (3) result in truncation errors which keep propagating in the filter. It is therefore imperative to develop a better understanding of the relation between the desired accuracy and the number of bits used in representing the fractional parts of the signal. Therefore the primary source of error in filter realization comes from truncations due to bit shifts.

In an attempt to model the total truncation error in the filter, the impact of truncation in each of the branch needs to be considered. This is different from direct form [4], where all the errors are lumped at a single point. In this case the errors are distributed in each of the branches. For parallel form, the contributions from each of the branches need to be accumulated to get the overall impact of truncations. An approach analogical to the one with overflow avoidance can be followed where the impulse response from each of the branches contributing to the truncation errors is calculated and added together.

It can be understood that the maximum possible truncation errors in each of the bit shift blocks can easily be calculated based on the situation when all the bits that are lost due to shifting are 1's. The positional value of all such bits in a fixed point representation is related to the number of bits shifted as well the number of bits used in representing the fractional part of the fixed-point representation. So, if F bits are used in representing the fractional part and k_i bits are omitted due to right shift in the i-th branch, the maximum possible truncation error in output in branch i, y_i that we can have is,

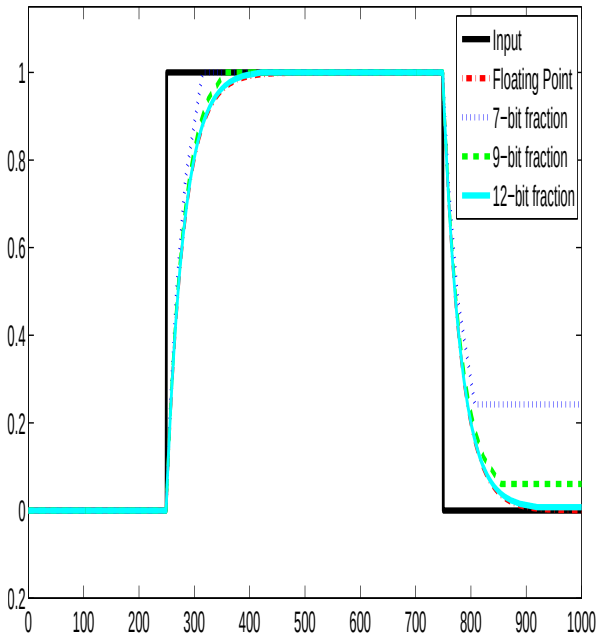

Fig. 5. Impact of number of fractional bits on the filter accuracy

$$\Delta y_i[n] = 2^{-(F-k_i)} \tag{14}$$

This maximum possible error can be fed after each of the shift operators as shown in 3. The total permissible error can be obtained by looking at the transfer function from these error input branches to the output signal. The transfer function from i-th branch to the output signal is simply,

$$H_{y,e_i}(z) = \frac{R_i(1-p_i)^{-1}}{1-p_i z^{-1}} \tag{15}$$

Using the bound in eqn. 9, the maximum possible error that we can have in the system is,

$$|\delta y_{max}| = \sum_{m=1}^{N} R_m 2^{-(F-2k_m)} \tag{16}$$

and the minimum number of bits used to represent the fractional part in the output to ensure that truncation error losses do not adversely impact the filter behavior is simply,

$$\lceil |log_2(\delta y_{max})| \rceil + 1 \tag{17}$$

We attempt to use these bounds for a simple case of single pole IIR filter with $R_1 = 1$ and $p_1 = (1 - \frac{1}{2^5})$. For a desired The bounds as shown in eqn. 17, indicate at using at least 11 bits if permissible error level is at 2%. It can be seen in Figure 5 that using 12 bits for representing the fractional part helps us nearly match the equivalent floating point representation which has a much larger dynamic range for representing values than a 12-bit fixed point representation. If the tolerable error levels are increased to 10%, the bounds indicate that only 9 bits can be sufficient to represent the fractional part. If we further reduce the number of bits in fractional part, the truncation errors become huge and Figure 5 shows that the filter behavior is largely inaccurate when insufficient number of bits are used in representing the fractional parts.

VII. Future Work: Non-binary Poles

When exact precision is required in pole placement in the parallel form filter, the bit shift operation in the loop needs to be replaced by multiplications. Essentially, the ARMA relation for generic pole placement translates to,

$$y[n] = \sum_{m=1}^{N} (R_m x[n] + p_m y[n-1]) \tag{18}$$

which needs greater precision in doing the multiplication with the pole values due to their proximity to the unity circle. The multiplier poles can also be represented in fixed point format and the multiplications can be made faster using specialized dedicated hardware accelerator such as carry-save adder (CSA) based multiplications to accelerate the multiplication process [5]. The required precision can be calculated in a manner analogous to the determining the fractional part as discussed in previous section. This work is being continued to work with optimized, effective design for filters with generic pole placements and metrics related to hardware complexity and power consumption will be collected.

VIII. Conclusion

In this work, a particular class of IIR digital filters in parallel form with binary poles was considered. The study presented details on choosing sufficient number of bits to accomplish (a) Precision in residue value representations (b) Overflow avoidance in signal value calculations (c) Bounding the truncation errors in final output to a desired value. Analytical results are presented to work out the necessary details for filter design and some fixed-point simulations are compared against floating point simulations to validate the bound on desired precision. Extension of such approaches to a more generic class of filters will be the focus of future research efforts to derive similar word length requirements.

References

[1] R. Yates, *Practical Considerations in Fixed-Point FIR Filter Implementations*, Digital Signal Labs White paper, November 2006.
[2] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, Prentice-Hall, New Jersey, 1996.
[3] J. C. Doyle, B. A. Francis, and A. R. Tannenbaum, *Feedback Control Theory*, Macmillan, New York, 1992.
[4] J. Carletta, R. Veillette, F. Krach and, Z. Fang, *Determining Appropriate Precisions for Signals in Fixed-point IIR Filters*, DAC 2003, June 2-6,2003, Anaheim, Califomia, USA.
[5] Andrew G. Dempster and, Malcolm D. Macleod, *Use of Minimum-Adder Multiplier Blocks in FIR Digital Filters*, IEEE Transactions on Circuits and Systems-11; Analog and Digital Signal Processing, Vol. 42, No. 9, September 1995.
[6] N. Karaboga1, B. Cetinkaya1, *Efficient Design of Fixed Point Digital FIR Filters by Using Differential Evolution Algorithm*, Volume 3512 Computational Intelligence and Bioinspired Systems.
[7] B. Wu, J. Zhu, F. Najm, *An Analytical Approach for Dynamic Range Estimation*, DAC, pp. 472-477, Design Automation Conference, 41st Conference on (DAC'04), 2004.
[8] A. V. Oppenheim, R. W. Schafer, *Discrete-Time Signal Processing*, Prentice Hall, 2001.