TEXAS INSTRUMENTS

Application Report
SPRA509

Overflow Avoidance Techniques in Cascaded IIR Filter Implementations on the TMS320 DSP's

Aaron Kofi Aboagye *C5000 DSP Software Applications*

Abstract

DSP programmers are faced with the problem of dealing with overflows that occur as a result of some computation. A typical case is that of the implementation of recursive or infinite impulse response (IIR) digital filters. These implementations are very prone to overflows, due to the recursive naturs of the filters. In this application report, different methodologies are presented to help programmers avoid or deal with these overflow conditions. The application report also presents a simple design example that utilizes one of the techniques in Matlab™.

Contents

Figures

Introduction

One of the most important issues in digital filter implementations on fixed-point devices is the prevention or handling of overflow of results, due to the precision of the hardware. In certain cases, an occurrence of an overflow is noncatastrophic; while it is catastrophic in others. This is shown by the nature of recursive (IIR) and nonrecursive (FIR) digital filter implementations. In nonrecursive filters, an overflow in computation affects the result of that particular computation, y[n]. In contrast, an overflow in computation in a recursive filter affects that particular result, y[n], and all remaining results due to the inherent feedback in their structure.

The objective of this application report is to present different techniques that may be used to prevent overflows in implementations of IIR digital filters on the Texas Instruments (TI™) TMS320 family of digital signal processors (DSPs). This application report focuses on the TMS320C54x DSP; however, the concepts may be applied to any other DSP. Note that most of the implementations currently available for the 'C54x devices are based on the direct form II structured second order sections (biquads) and, as such, the discussion here will be with respect to second order IIR filter sections. The general solution to this problem is scaling, to ensure that overflows do not occur. However, the choice of scale factors and where and when to scale for optimal precision and efficiency is not so trivial. The main categories of scaling techniques we will examine are:

❒ fixed scaling of second order coefficients

❒ split fixed scaling of feed-forward and feed-back coefficients

❒ fixed scaling of input samples

❒ adaptive scaling of second order coefficients

We will discuss the methodologies in each category.

Second Order IIR Filter Sections

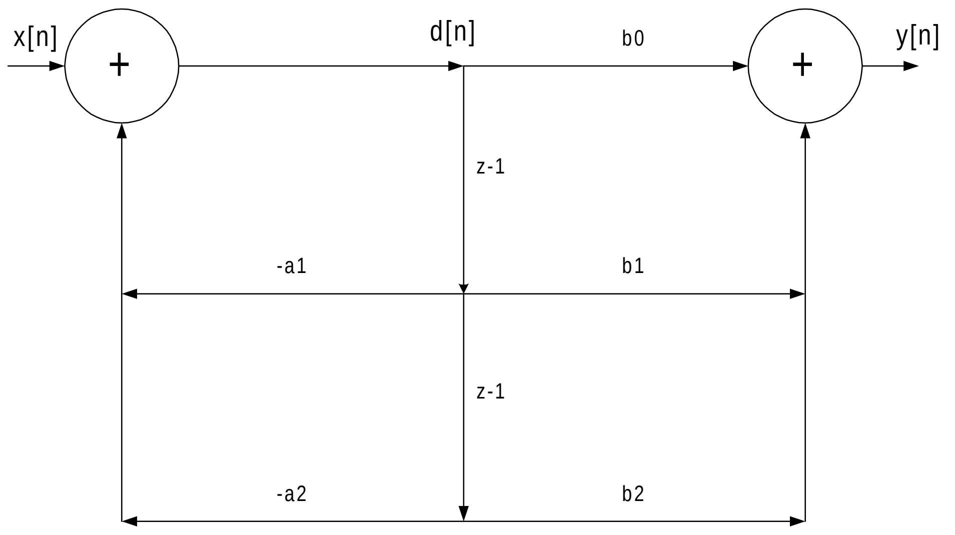
To design and implement high order IIR filters, the general transfer function is given in equation (1).

$$H(z) = \frac{b_0 + b_1 Z^{-1} + ... + b_M Z^{-M}}{1 + a_1 Z^{-1} + ... + a_N Z^{-N}} \quad (1)$$

However, when the coefficients (a_1.... a_N, b_0.... b_N) are quantized, the resulting errors can significantly alter the desired filter characteristics. It has been shown that breaking up the transfer function into lower-order sections and connecting these in cascade or parallel can reduce this sensitivity to coefficient quantization. In our case, we chose to use direct form II structured second order sections in our implementations. The transfer function of this section is given in equation (2) and the corresponding direct form II block diagram is shown in Figure 1.

$$H(z) = \frac{b_0 + b_1 Z^{-1} + b_2 Z^{-2}}{1 + a_1 Z^{-1} + a_2 Z^{-2}} \quad (2)$$

Figure 1. Direct Form II Structure Block Diagram



The difference equations used for implementation of the direct form II structure shown in Figure 1:

$$d[n] = x[n] - a_1 d[n-1] - a_2 d[n-2]$$
$$y[n] = b_0 d[n] + b_1 d[n-1] + b_2 d[n-2]$$

In certain implementations of cascaded IIR filters, only four coefficients are used instead of five as shown in Figure 1. This reduction in the number of coefficients is achieved by dividing the numerator by b2, thus making the coefficient of the z^{-2} term equal to 1 and giving the entire system a gain of b2. In doing this, a change in the number representation of the coefficients may be necessary since it is possible to end up with numerator coefficients larger than 1. The transfer function in that case is:

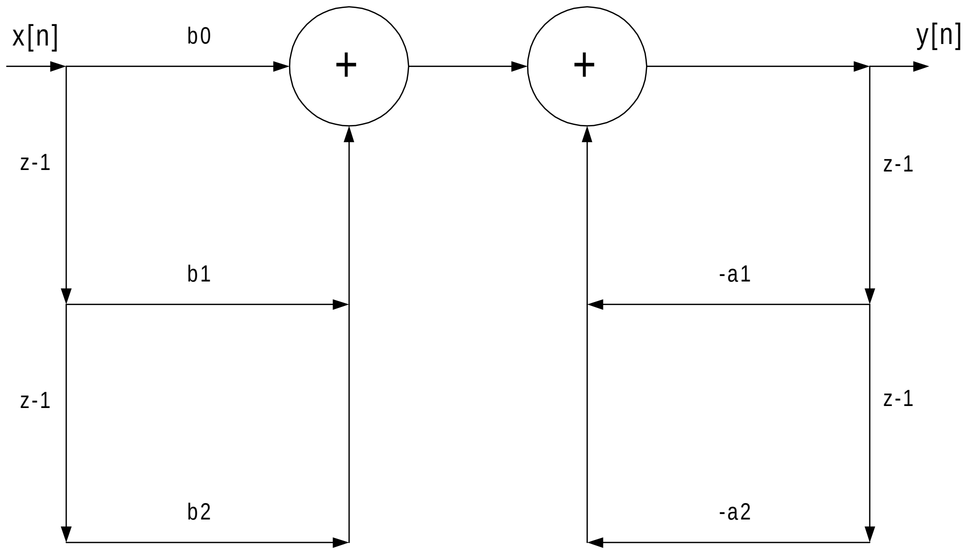
$$H(z) = b_2 * \frac{b_{01} + b_{11}Z^{-1} + Z^{-2}}{1 + a_1 Z^{-1} + a_2 Z^{-2}} \tag{3}$$

where

$b_{01} = {}^{b0}\!/_{b2}$ and $b_{11} = {}^{b1}\!/_{b2}$

Figure 2 shows the structure for a direct form I IIR filter. The main difference between the direct form I and the direct form II implementations is the amount of buffer space required. The direct form I implementation requires two delay buffers for x[n] and y[n]; the direct form II requires only one buffer for d[n]. The presence of the guard bits in the ‘C54x devices allows the entire computation of the difference equation for direct form I to be done in the accumulator and the overflow only needs to be taken care of at y[n]. This makes the direct form I structure less susceptible to overflows than the direct form II structure. However, a drawback of the direct form I structure is that its implementation consumes more cycles than the direct form II.

Figure 2. Direct Form I Structure Block Diagram



The difference equation used for implementation of the direct form I structure shown in Figure 2 is:

$$y[n] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] - a_1 y[n-1] - a_2 y[n-2]$$

The IIR functions in the TMS320C54X DSP function library, 54DSPLIB, accessible from ftp://ftp.ti.com/, feature both implementation structures. DSPLIB implements direct form II IIR filters with four and five coefficients per biquad and a direct form I IIR filter with five coefficients per biquad.

Scale Factor Computation

Each of the techniques we will discuss utilizes the same scale factor computation technique. The differences are where and when to apply the scale factor/gain. Following the convention that each fixed-point number represents a fraction, each node in the network (see Figure 1) must be constrained to have an amplitude of less than 1 to avoid overflow. If we let $w_k[n]$ denote the value of the kth node variable, $h_k[n]$ denote the impulse response from the input, and x[n] to the node variable $w_k[n]$, then:

$$\left|w_k[n]\right| = \left|\sum_{m=-\infty}^{\infty} x[n-m]h_k[m]\right| \tag{4}$$

We will present three different methods of computing G_k, the scale factor to ensure that

$$G_k * \left|wk[n]\right| < 1 \tag{5}$$

l1 Norm

The l1 norm is evaluated as:

$$\| h \|_1 = \sum_n | h[n] | \tag{6}$$

In this method of computing G_k, the signal is assured not to overflow.

Chebychev Norm

The Chebychev norm of the frequency response H(f) is evaluated as:

$$\| H \|_C = \max_F | H(f) | \tag{7}$$

This method of computing G_k only assures that the steady-state response of the system to a sine wave will not overflow.

l2 Norm

The l2 norm is evaluated as:

$$\| h \|_2 = \left[\sum_n h^2[n]\right]^{1/2} \tag{8}$$

This method of computing G_k also allows overflows but inherently uses a measure of the probability of overflow.

Scaling Techniques

Fixed Scaling of Second Order Coefficients

The first step is to identify the nodes in each second order section where overflow could occur. Looking back at Figure 1, we identified nodes d[n] and y[n] to be the potential overflow nodes.

At this stage, the transfer function for each of these nodes needs to be evaluated. The transfer function for d[n] is shown in equation (9). Note that the transfer function for y[n] is the same as that given in equation (2).

$$H_d(z) = \frac{1}{1 + a_1 Z^{-1} + a_2 Z^{-2}} = \frac{D(z)}{X(z)} \tag{9}$$

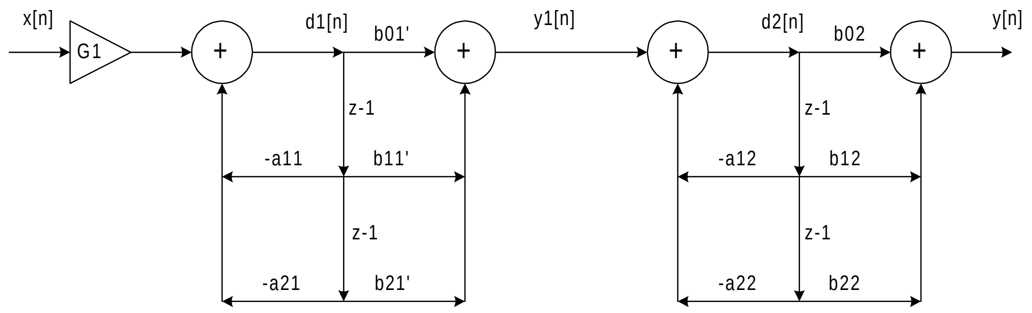
With these two transfer functions, we are now in a position to calculate the scale factor, G_k, for each node in the block diagram. Depending on the application and types of inputs, one of the three methods outlined in the previous section should be chosen for the computation of $G_{k.}$ An alternative is to evaluate all three values of G_k for each node. Then, having evaluated the different values of G_k for all nodes in the section, the highest value is chosen, even though this will be a very conservative means of scaling. At this point, G_k should be used to scale the coefficients of that section.

This process should be repeated for each subsequent second order section in the IIR filter. It is essential to reduce the risk of overscaling by ensuring that all subsequent scale factors take into consideration the scaling of the previous section. One method of achieving this is to ensure that each transfer function/impulse response evaluated should be relative to the input x[n] and not the input to that section. In this case, the scaled coefficients of preceding sections should be used in computing these transfer functions/impulse responses. Another method is to calculate the scale factor of each biquad independently, and the actual scale factor used in biquad k, G_k', is:

$$G_k' = \frac{G_k}{G_{k-1}}$$

Figure 3 shows the structure for the previous method of scale factor allocation. In this method, only the b_i coefficients are scaled. The scale factor for each section is introduced in the b_i's of the preceding section and there will be no overflow at any of the nodes. The scale factor for the first stage, G_1, is introduced as an input scale factor as shown in Figure 3 and subsequently, there will be no scaling in the last stage since the scale factor for that stage will have been introduced in the previous stage.

Figure 3. Direct Form II Scaling Structure Block Diagram



In Figure 3, the scale factor of section 2, G_2, is used to scale the b_i coefficients of section 1 as follows:

$$G_2' = \frac{G_2}{G_1}, b_{01}' = \frac{b_{01}}{G_2'}, b_{11}' = \frac{b_{02}}{G_2'}, b_{21}' = \frac{b_{21}}{G_2'}$$

Since this example only has two second order sections, the b_i coefficients of the last section are left unchanged.

In the C54x DSPLIB, you are expected to scale the input signal by G_1 prior to invoking the IIR direct form II functions. The input will still be in Q.15 but the coefficients should be kept in a number format, QN.M that ensures that the largest coefficient just fits. An indication of this number format, the number of integer bits N-1, is also passed to the function.

Split Fixed Scaling of Feedforward and Feedback Coefficients

This technique is very similar to that previously described except that the transfer function of the second order section (biquad) is split into the feedforward (FIR) and feedback (All-Pole) functions. Equation (10) shows the feedforward transfer function.

$$H_b(z) = b_0 + b_1 Z^{-1} + b_2 Z^{-2} = \frac{Y(z)}{D(z)} \qquad (10)$$

Splitting the scaling procedure into the two paths ensures that overflow does not occur in either direction during computation. Based on the application and types of signals in question, one of the 3 norms is evaluated for each of the two transfer functions, equations (9 and 10), and the corresponding coefficients, b_k's for the feedforward and a_k's for the feedback, are scaled by the calculated norms.

When scaling the feedback coefficients, ak's, it is essential to determine whether the actual norm calculated is sufficient or a negation is necessary. This is because the computation of the norm uses absolute values and, as such, the summation of coefficients of the impulse response of equation (9),

$$1 - a_1 - a_2$$

may not be less than 1 in some cases, if the absolute value computed is always used.

The workaround is:

```
if (–a1 – a2 < 0)
        Scale with $G_k$
else
        Scale with $–G_k$
end
```

This procedure is repeated independently for each biquad in the entire structure. In certain implementations, it may be desirable to scale up the output of each biquad back to the expected value prior to computing the output of the next stage. This is because this procedure tends to overscale in certain cases. However, this scaling up procedure also leads to another possibility of overflow and care should be taken if this approach is adopted.

Fixed Scaling of Input Samples

The main idea in this technique is to scale the inputs with a fixed scale factor. The scale factor, G_k, is computed using one of the scale factor computation methods outlined in the previous section for each node or, as previously discussed, all three scale factors may be evaluated for each node. The maximum scale factor is then chosen, since this is the only way to avoid overflow at each node.

There are typically two ways of applying this scale factor. The scale factor may be applied at each node/register in the network (scaling the coefficients) prior to or during the computation or the input samples may be scaled prior to any computation. In either case, this technique results in a lower signal-to-noise ratio (SNR) than the other techniques described.

Adaptive Scaling of Second Order Coefficients

This scaling technique is different from the three techniques previously discussed, in that, the scaling of coefficients is done at run time. The overflow bit is checked to determine if an overflow occurs at any node in any second order section during computation and, if so, the results of computation of that node are scaled down sufficiently to avoid the overflow. The value is also checked to see if an underflow has occurred (result too small) and, if so, the result is scaled up.

However, even though this technique provides the best means of scaling, due to the fact that the risk of overscaling is greatly minimized, the MIPS consumption is very high and does not make it a practical implementation technique on a fixed point device.

Example

An example of the fixed scaling of second order coefficients technique was developed in Matlab for a specified IIR filter using cascaded second order sections, with five coefficients per biquad. This example uses the *l1* norm to evaluate the scale factor but, as previously mentioned, any or all of the other techniques may be used. The problem is to design an Nth-order elliptic low-pass filter with the following specifications:

pass-band edge frequency, Wp = 0.2fs

pass-band ripple = 0.2dB

stop-band edge frequency, Ws = 0.3fs

stop-band ripple, Rs = -20dB

The solution will be presented in a stepwise fashion with the Matlab script interleaved when necessary.

Step 1: Determine the required order of the filter to meet specs

[N, Wn] = ellipord(0.2, 0.3, 0.2, 20)

Step 2: Determine filter coefficients

[B, A] = ellip(N, 0.2, 20, Wn);

Step 3: Break filter into second order sections

Transform transfer function into zero-pole representation:

[Z, P, K] = tf2zp(B, A)

Transform zero-pole representation into second order section coefficients,
coeff = zp2sos(Z, P, K):

```
coeff1 = coeff;       % copy of coeffs for method II
coeff2 = coeff;       % copy of coeffs for method II
coeff3 = coeff;       % copy of coeffs for no scaling
```

At this point, we have the order, N, and coefficients, coeff, of the corresponding biquads. From the results of the script, we get N = 4 and the following coefficients shown in the transfer function:

$$H(z) = \frac{0.1571 - 0.0656Z^{-1} + 0.1571Z^{-2}}{1 - 1.2109Z^{-1} + 0.4596Z^{-2}} \bullet \frac{0.6911 - 0.9895Z^{-1} + 0.6911Z^{-2}}{1 - 1.4857Z^{-1} + 0.8877Z^{-2}}$$

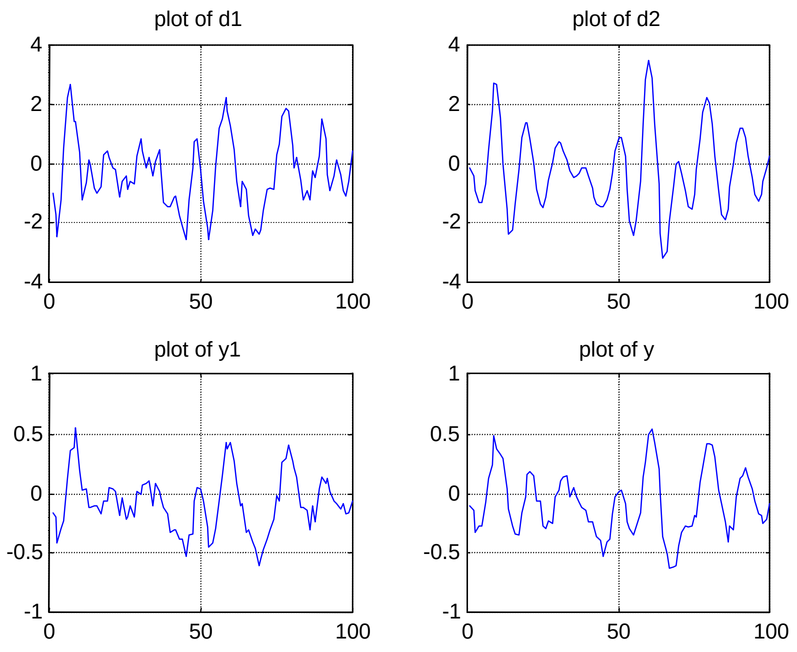
Using the same implementation as that used in the 54DSPLIB, the output in response to random input is evaluated. The Matlab script is:

```
% Inputs, Outputs, and Intermediate results
Ns = size(coeff,1);                                  % number of sections
Nx = 100;
y = zeros(Ns, Nx+2);
x = [0 0 (rand(1,Nx)-0.5)*2];
d = zeros(Ns, Nx+2);

for n = 3:Nx+2
    for i = 1:Ns
        d(i,n)=(x(n)-coeff(i,5)*d(i,n-1)-coeff(i,6)*d(i,n-2));
    y(i,n)=(coeff(i,1)*d(i,n)+coeff(i,2)*d(i,n-1)+coeff(i,3)*d(i,n-2));
    x(n) = y(i,n);
  end
end
y(2, :) = y(2, :) * K;
```

Figure 4 shows plots of all the intermediate computational results. You can see that overflows occur in both intermediate stages of the computation even though the final outputs of the stages do not overflow; therefore, scaling is needed. As previously stated, this example makes use of the *l1* norm in calculating the scale factors. Two methods of determining scale factors of all subsequent biquads following the first will be used and the results compared.

Figure 4. Results of Unscaled Computation



The Matlab script for evaluating the *l1* norms for the two methods and scaling of coefficients of the second order sections in this example is shown in step 4. Note that the implementation structure used for the two methods is the same as that shown in Figure 3.

Step 4: Calculate scale factors and scale coefficients for each section

```
H1 = zeros(Ns,159);
H2 = zeros(Ns,159);

for i=1:Ns
    % Method I impulse responses & norm evaluation
    Ha(i,:) = impz([1 0 0], coeff(i,4:6), 80)';
    Hb(i,:) = impz(coeff(i,1:3), coeff(i,4:6), 80)';

    l1_H1(i,1) = sum(abs(Ha(i,:)));        % l1_norm for first node
    l1_H1(i,2) = sum(abs(Hb(i,:)));        % l1_norm for second node
    Gk(i) = max(l1_H1(i,:));               % scale factor for section i
    l1_1(i) = Gk(i) / prod(l1);            % Gk(I) / product of previous scale factors

    % Method II impulse responses & norm evaluation
    if i == 1
        H1(i,1:80) = impz([1 0 0], coeff2(i,4:6), 80)';
        H2(i,1:80) = impz(coeff2(i,1:3), coeff2(i,4:6), 80)';
    else
        H1(i,:) = conv(impz(coeff2(i-1,1:3), coeff2(i-1,4:6), 80), impz([1 0 0], coeff2(i,4:6), 80))';
        H2(i,:) = conv(impz(coeff2(i-1,1:3), coeff2(i-1,4:6), 80), impz(coeff2(i,1:3), coeff2(i,4:6), 80))';
    end

    l1_H2(i,1) = sum(abs(H1(i,:)));        % l1_norm for first node
    l1_H2(i,2) = sum(abs(H2(i,:)));        % l1_norm for second node
    l1_2(i) = max(l1_H2(i,:));             % scale factor for section i

    % Scaling of coefficients
    if i ~= 1
        coeff(i-1,1:3) = coeff(i-1,1:3) / l1_1(i);      % Scale coeffs (bi's) - Method I
        coeff1(i-1,1:3) = coeff1(i-1,1:3) / l1_2(i);  % Scale coeffs (bi's) - Method II
    end
    coeff2(i,1:3) = coeff2(i,1:3) / l1_2(i);           % Scale coeffs (bi's) - Method II convolution
end
```

The results of the previous script provide you with the following numerical results:

```
l1_1 = [4.6801   3.8934]

l1_2 = [4.6801   2.3940]
```

The Matlab script for the actual implementation with both scale factors is:

```
% Using scaled coefficients from method I
for n = 3:Nx+2
    x(n) = x(n) / l1_1(1);
    for i = 1:Ns
        d(i,n) = x(n) - coeff(i,5)*d(i,n-1) - coeff(i,6)*d(i,n-2);
        y(i,n) = coeff(i,1)*d(i,n) + coeff(i,2)*d(i,n-1) + coeff(i,3)*d(i,n-2);
        x(n) = y(i,n);
    end
end
y_0 = y(2,:) * prod(l1_1) * K;
```

```
% Using scaled coefficients from method II
for n = 3:Nx+2
    x1(n) = x1(n) / l1_2(1);
    for i = 1:Ns
        d1(i,n) = x1(n) - coeff1(i,5)*d1(i,n-1) - coeff1(i,6)*d1(i,n-2);
        y1(i,n) = coeff1(i,1)*d1(i,n) + coeff1(i,2)*d1(i,n-1) + coeff1(i,3)*d1(i,n-2);
        x1(n) = y1(i,n);
    end
end
y_1 = y1(2,:) * prod(l1_2) * K;
```

Figure 5 and Figure 6 show the results from methods I and II, respectively. You can see that none of the results overflow. You can also observe that the output using method I is scaled more severely than the output of method II. This is because method I is based on the assumption that the input to stage 2 is in the range, $-1 < y1[n] < 1$, which is not the case.

Figure 5. Results of Scaled Computation (Method I)

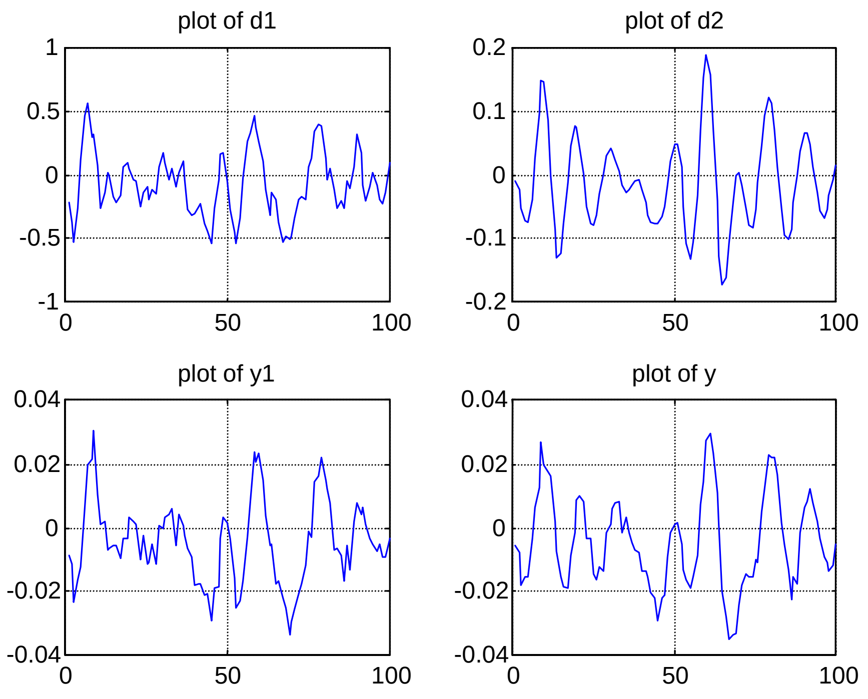


Figure 6. Results of Scaled Computation (Method II)

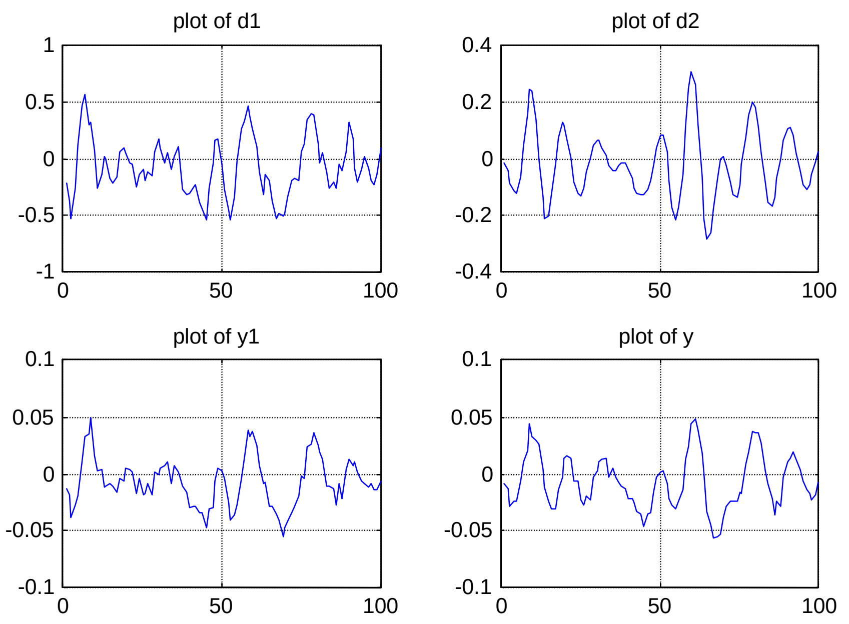
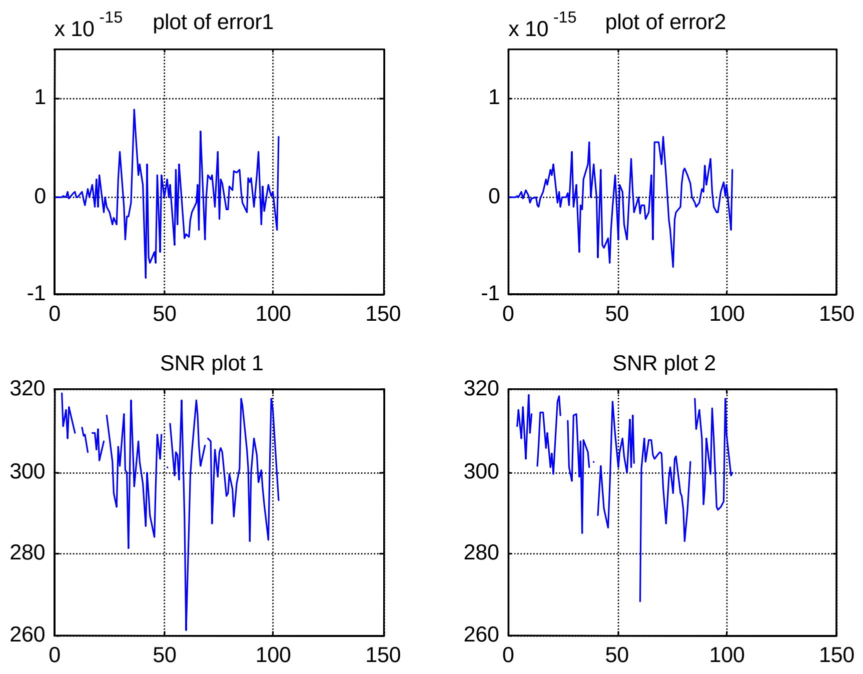


Figure 7 shows plots of the error due to scaling and the corresponding signal-to-noise ratio (SNR) for both methods. You can see that the SNR is almost the same for both implementations and the only differentiating factor is the severity of scaling in method I compared to method II.

Figure 7. Signal-to-Noise Ratio (SNR) Plots for Method I and Method II



To further illustrate the effectiveness of both techniques, the input data was changed to a constant stream of 1s, which represents the worst-case scenario. The results of this are shown in Figure 8 and Figure 9.

Figure 8. Results of Worst-Case Analysis (Method I)

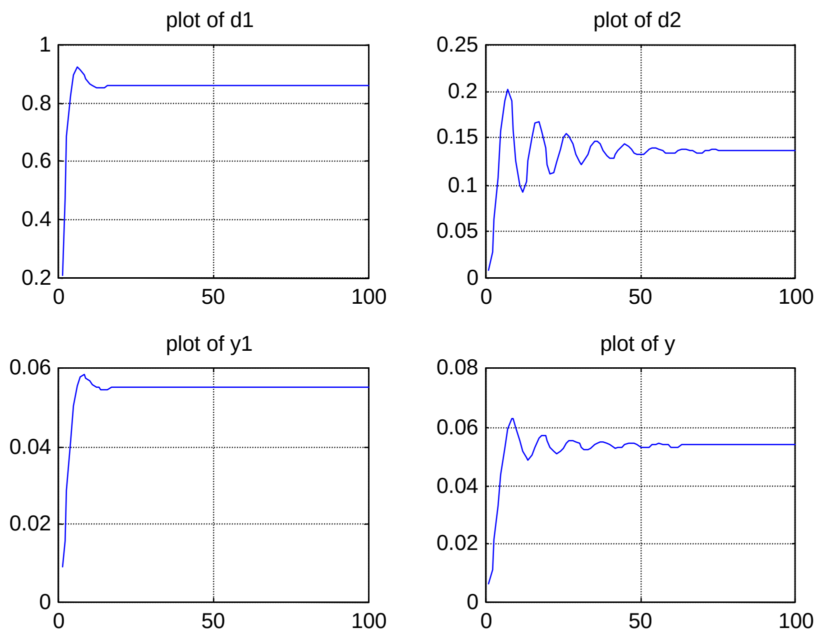
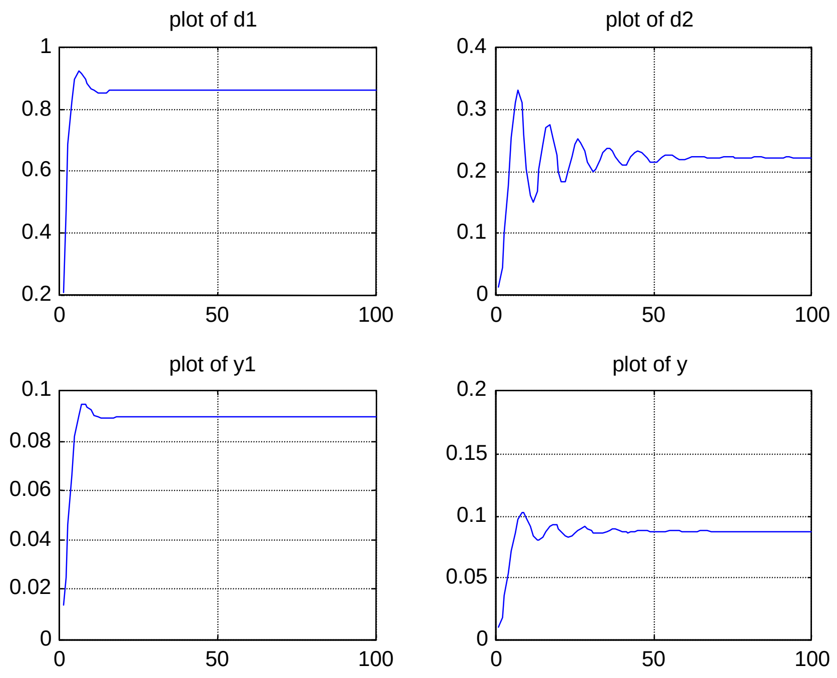


Figure 9. Results of Worst-Case Analysis (Method II)



Conclusion

This application report has presented four techniques for overflow avoidance in implementations of IIR filters on fixed-point devices. The results of the example also demonstrate one of these techniques. However, it should be noted that the choice of a scaling technique depends greatly on the application and required accuracy. It is up to you to determine whether a fixed scaling technique or an adaptive technique is adequate. If a fixed technique is to be used, the choice of scale factor will still depend on the application.

TI Contact Numbers