Jason Durand

Problem 4.1 – If V_{ref+} = 1.0V and V_{ref-} = 0V, regenerate Fig 4.1 using spice. (Design a 3-bit ideal DAC model in SPICE.) The y-axis will be voltages in decimal form.

The output of an ideal DAC is the exact analog value represented by the digital input. There is also no clock input to an ideal DAC, the output is updated as the binary input changes. Pictured below (Fig 1) is one way to model an ideal DAC in spice.

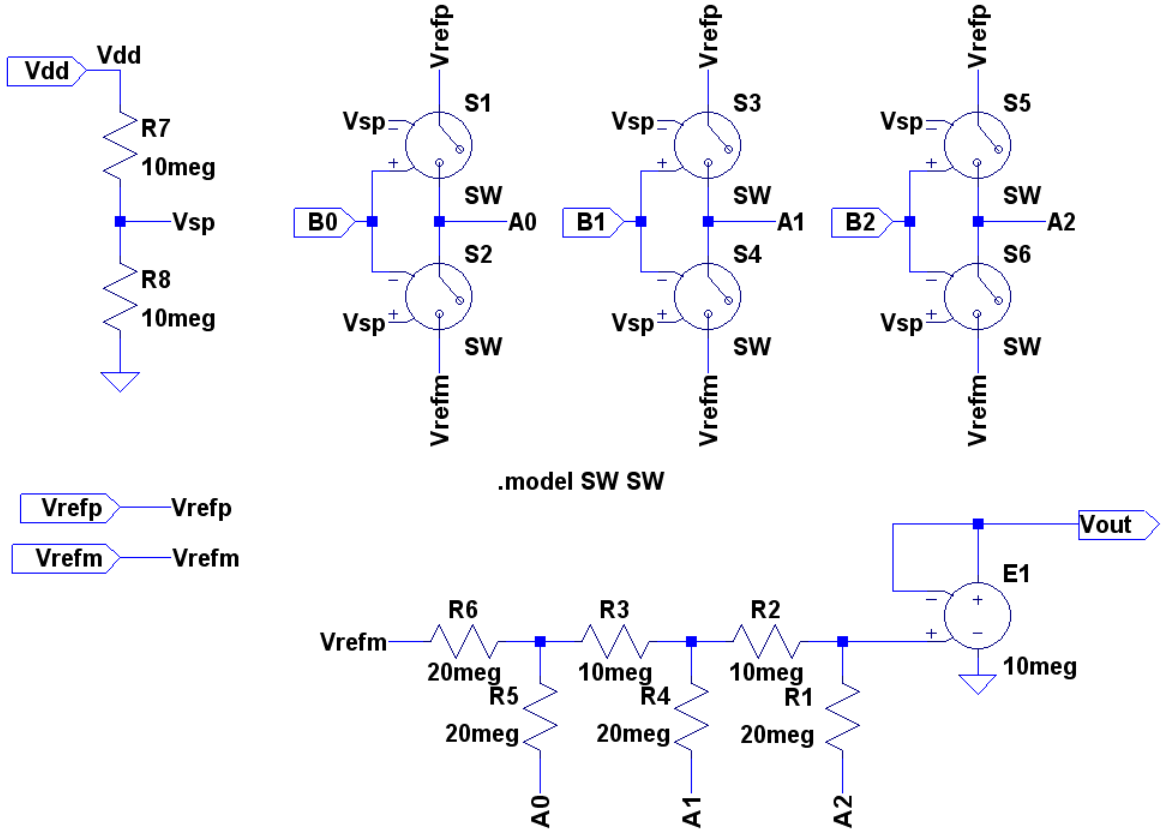


Figure 1: Schematic of Ideal 3-bit DAC

The resistive divider in the left defines an ideal switching point for the digital input, assumed to be Vdd/2. This switching point is used to clean up the (possibly) noisy input digital signals B2,B1,B0 and move them to clean digital values at the positive and negative references. These newly valued and clean digital signals are then fed to an R-2R resistor network an amplifier (voltage controlled voltage source), an architecture used in real DAC's. The circuit to the right (Fig 2), is the 3-bit DAC with a binary input of 000

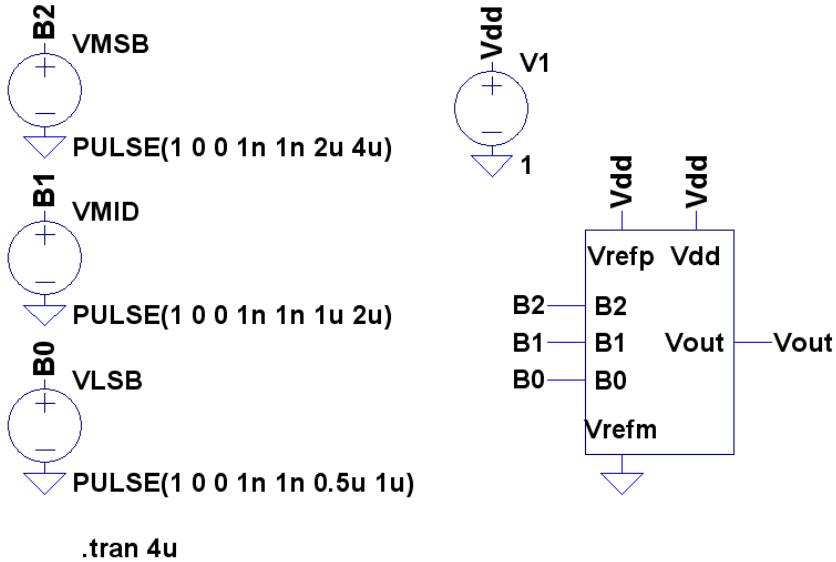


Figure 2: Ideal 3-but DAC icon and test circuit

to 111, adding 1 bit each 500ns. The output of the ideal DAC is shown in Fig 3.

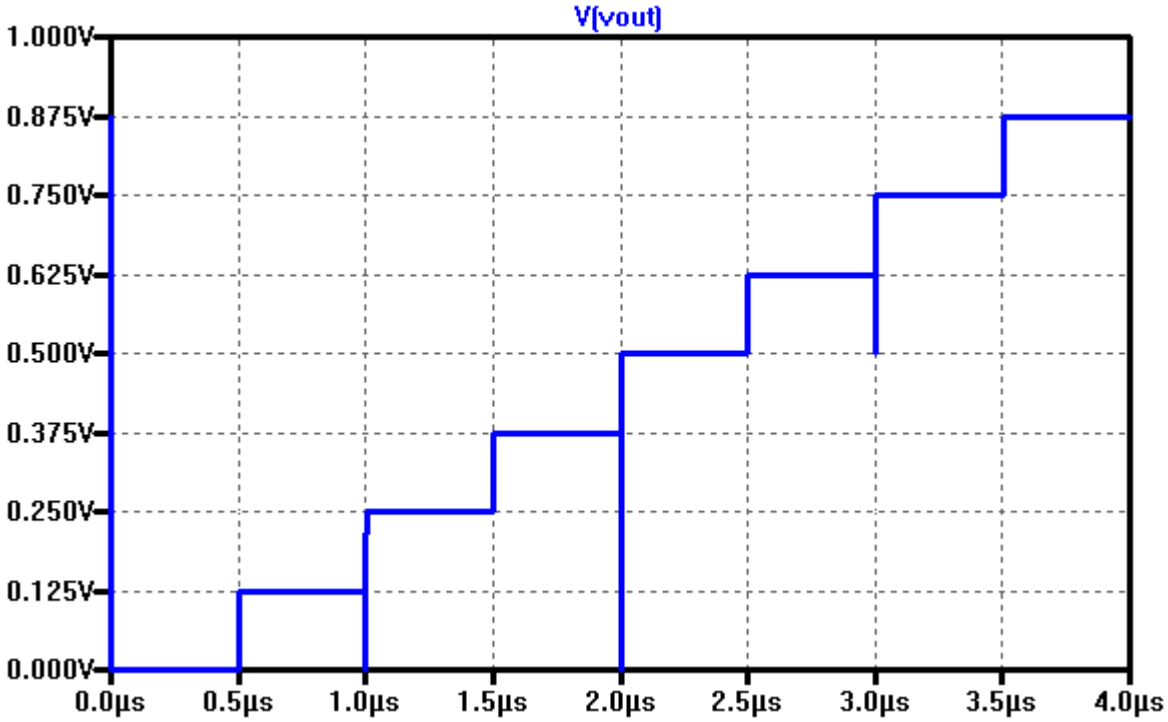


Figure 3: 3-bit DAC Output vs. Binary Input

Note that the digital input value changes every 500ns starting at t=0 with 000, ending at 3.5us with 111.

Question 4.2

If, again, $V_{REF+} = 1.0$ V and $V_{REF-} = 0$, sketch Fig. 4.1 for a 1-bit DAC. Note that the digital input code will be either a 0 or a 1 and the analog voltage out of the DAC will be either 0 or 1.0 V. Using Eq. (4.1) what is the voltage value of 1 LSB? How does this compare to the value of 1 LSB we get from the sketch? Is Eq. (4.1) valid for a 1-bit DAC? Why? The 1-bit DAC will be ubiquitous component in our noise-shaping modulators later in the book (see Fig. 7.15).

Solution

Figure 4.1 (page 120) shows the transfer characteristics of an ideal 3-bit DAC. As per the question requirements, Figure 1 below shows the transfer curve and block diagram for an ideal 1 bit DAC.

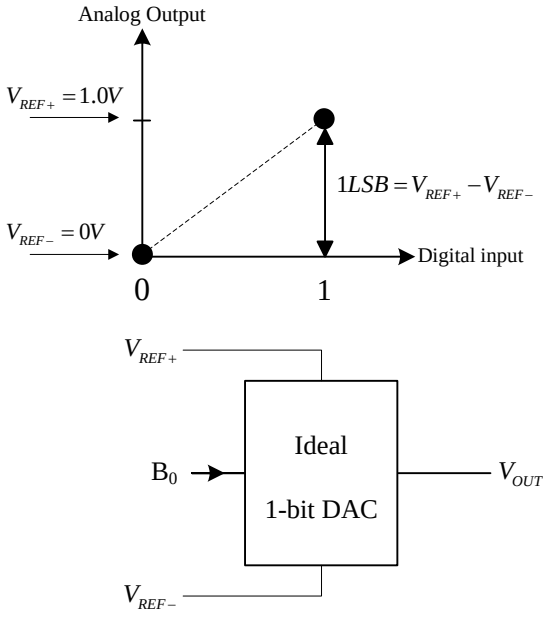


Figure1. An ideal 1-bit Digital-to-Analog converter

Now, equation 4.1 states that

$$1\ LSB = \frac{V_{REF+} - V_{REF-}}{2^N} = V_{LSB} \qquad N \geq 2$$

Thus from the equation above the value of 1LSB for an ideal 1-bit DAC is $\frac{V_{REF+} - V_{REF-}}{2^1}$ or with values of V_{REF+} and V_{REF-} as 1.0 V and 0 V respectively, 1 LSB = 0.5 V. Looking at Fig.1 above, the value of 1 LSB is $\frac{V_{REF+} - V_{REF-}}{2^0}$ or 1 LSB = 1.0 V instead. Hence from the above calculation and discussion, equation 4.1 (page 119) does not hold for 1-bit DAC case (as it is already specified in equation that $N \geq 2$). This is because 1-bit DAC is a special case and V_{OUT} final goes all the way up to V_{REF+} instead of $V_{REF+} - 1\ LSB$ in DACs with $N \geq 2$.

Tyler Hansen

4.3 – Why do the transfer curves of Fig. 4.3 show a shift of ½ LSB to the left? How do we implement this shift in Spice?

Fig. 4.3 from the book is a representation of the transfer curves produced by an ideal 3-bit ADC. This figure has been re-produced below in Figure 4.3.1.

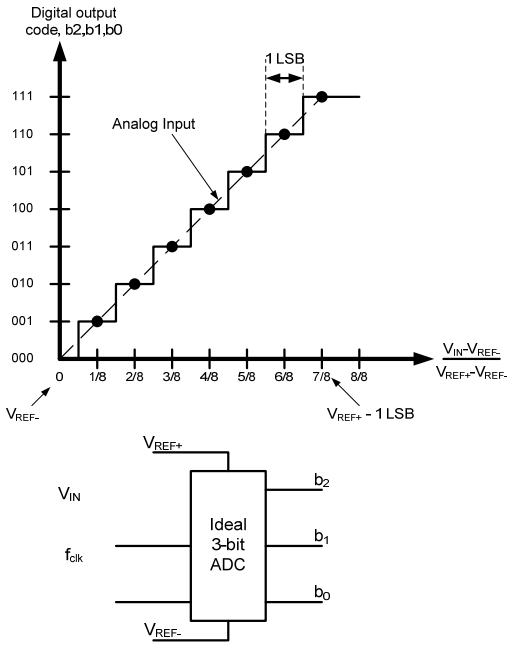


Figure 4.3.1 - An ideal 3-bit ADC

In analyzing an incoming analog signal, and ADC will start by sampling and holding the signal, then it will categorize the continuous analog input signal into different digital quantized "BINS". We shift the transfer curve to the left by ½ LSB to improve the ability of the ADC to efficiently "BIN" the incoming signal while minimizing the quantization noise.

We can think of it a different way using an example. Consider the case where the ADC in figure 4.3.1 is expected to convert an analog input voltage *slightly* lower than 1/8V into a digital representation. An example analog value of 0.1244V would most accurately be recognized as 001 instead of 000. This reduces the quantization noise that happens as a result of making the conversion from a continuous signal into quantized BINs. If we recognize the signal as 001, or effectively "0.125V", the quantization noise is 0.6 mV. On the other hand, if we didn't shift the transfer curve, anything below 0.125V would be "BIN-ed" as 000, resulting in a quantization noise of 0.1244V.

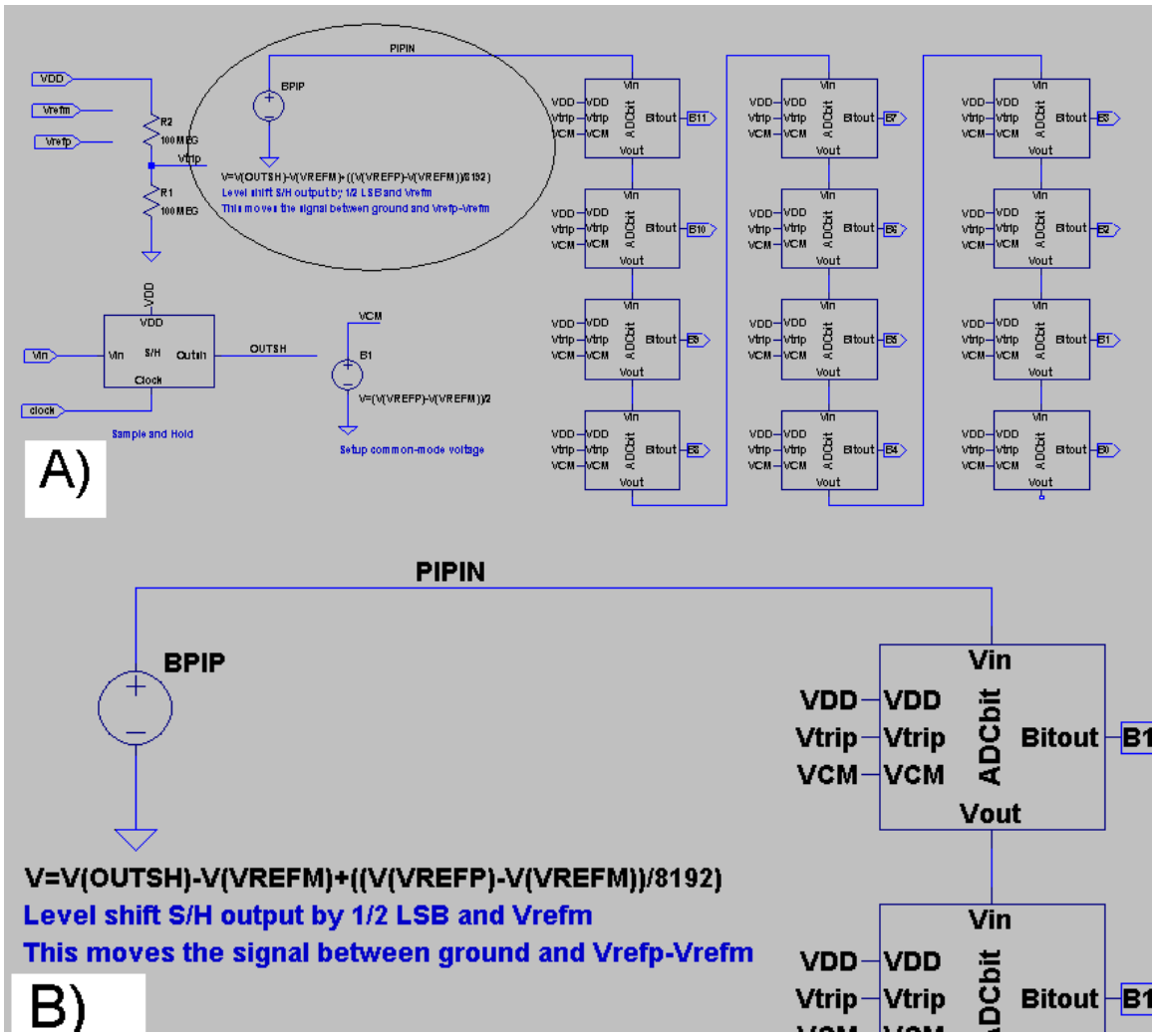
We accomplish the "1/2 LSB shift left" in Spice by using the following statement (copied from the Mixed-Signal Design Text page 123):

BPIP PIPIN 0 V = V(VOUTSH) – V(VREFM) + ((V(VREFP) – V(VREFM))/2^9)
Note that the last term of the statement…

((V(VREFP) – V(VREFM))/2^9

… is the definition of ½ LSB, or $\frac{V_{REF+} - V_{REF-}}{2^{N+1}}$. EQ – 4.3.1

At this point, I will take the liberty of copying in a screen-shot of the ideal 12-bit ADC provided in the Spice library provided at CMOS.edu in figure 4.3.2 below:

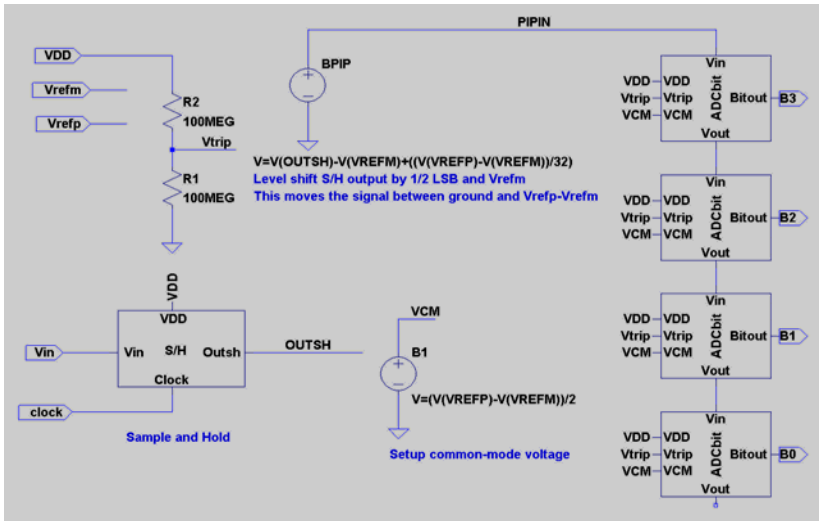


Equation 4.3.2 A) shows the sub-schematic showing the implementation of a 12-bit ADC. B) Zooms in on the implementation of the ½ LSB shift to the left, and depicts how this can be accomplished.

Note finally that in figure 4.3.2b), the ½ LSB term has a factor of 8192 in the denominator. According to equation 4.3.1 above, this should be 2^(12+1), which it is.

4.4 Use SPICE to implement 4-bit ADC and DAC. If the converters are clocked at 100 MHz (and the outputs of the ADC are connected to the inputs of the DAC), apply an input sinewave (to the ADC) that has an amplitude of 500 mV peak centered around 500 mV DC with a frequency of 5 MHz. Again, use VREF+ = 1.0 V and VREF- = 0. Show the DAC's analog output.

The following schematics are the individual ideal representation of an ADC and a DAC.



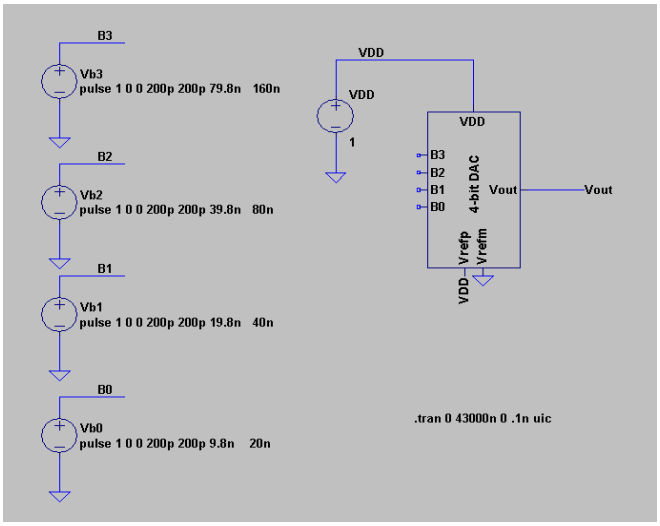


Fig. 1.a Ideal ADC circuit ***Fig. 1.b Ideal DAC circuit***

The ideal DAC has an output range that is the maximum possible: the V_{REF+} and V_{REF-} values of +1V and 0V respectively.

The SPICE representation of the DAC is:

```
B1 Vout 0 V=((v(vrefp)-
v(vrefm))/16)*(v(B3L)*8+v(B2L)*4+v(B1L)*2+v(B0L))+v(vrefm)
```

To implement the circuit with the outputs of the ADC feeding the DAC, the intermediate nodes were shorted. The circuit below has the a sinusoidal input wave of 5MHz, with the ADC and DAC being clocked at 100 MHz.

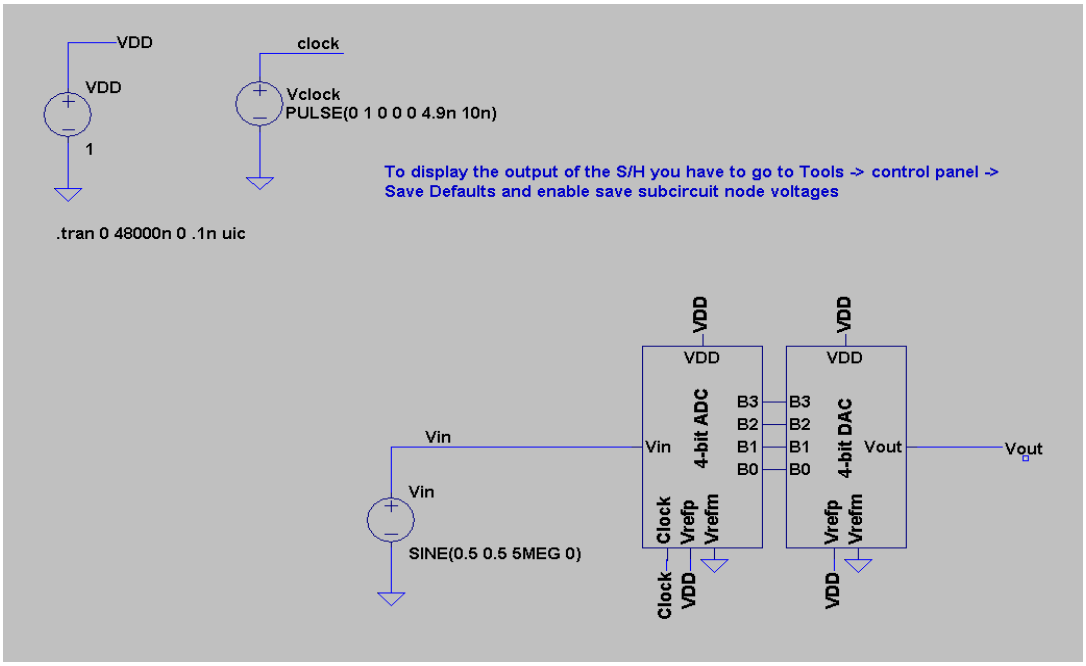


Fig. 2 Circuit Implementation

The following output waveform shows that, though tracking the input sinusoidal waveform, it is quantized. This is a characteristic of the ADC.

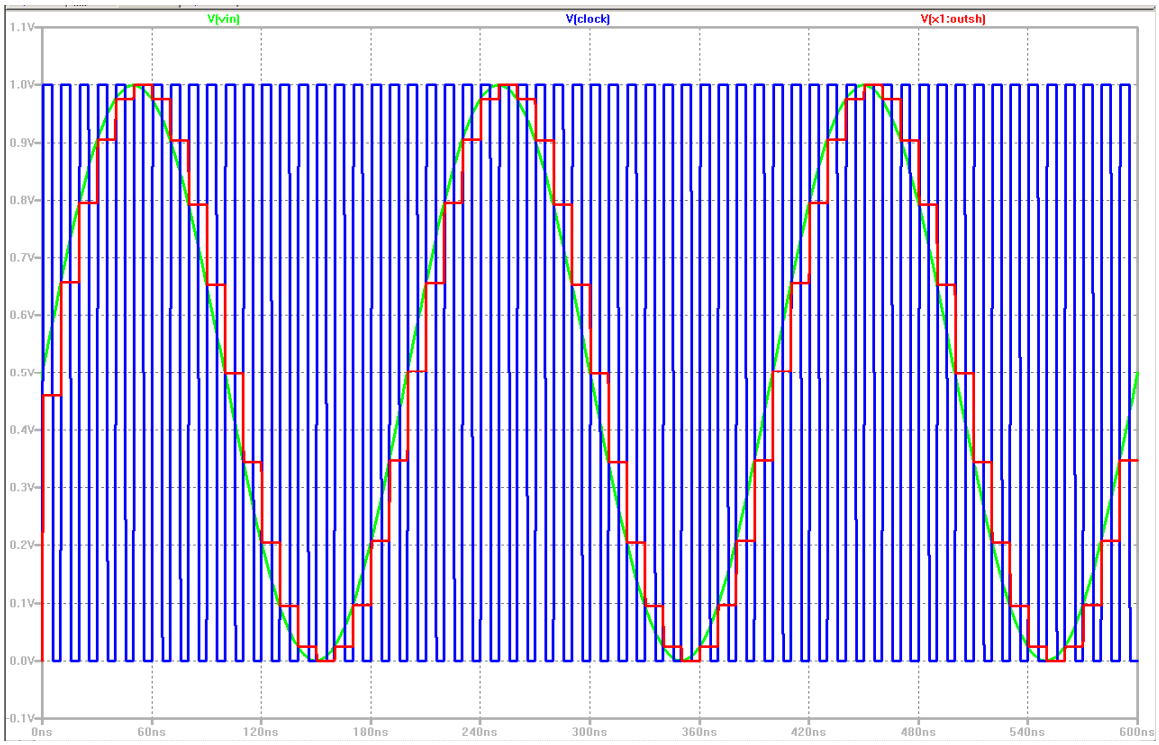


Fig. 2.a Output waveform

The reference voltage V_{CM} (also termed Common Mode coltage) can be calculated as the average of V_{REF+} and V_{REF-} values. In this example, since the V_{REF+} values are +1V and 0V, V_{CM} is 0.5V.

Another datapoint that needs to be calculated to represent the ADC is the ½ LSB value.

$$\text{½ LSB} = \frac{V_{REF+} \text{ and } V_{REF-}}{2^{N+1}}$$

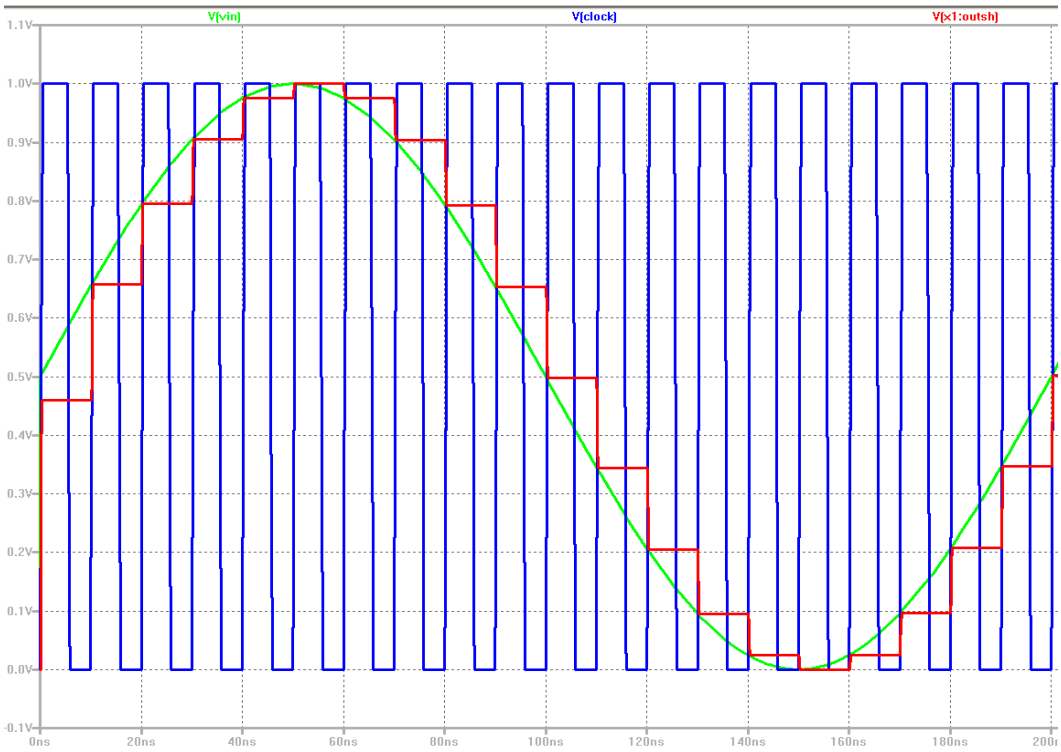


Fig. 2.b Output waveform (zoomed in)

The ideal DAC has an output range that is the maximum possible: the V_{REF+} and V_{REF-} values of +1V and 0V respectively.

4.5 Using SPICE generate the spectrums of the input and output signals in question 4.4.

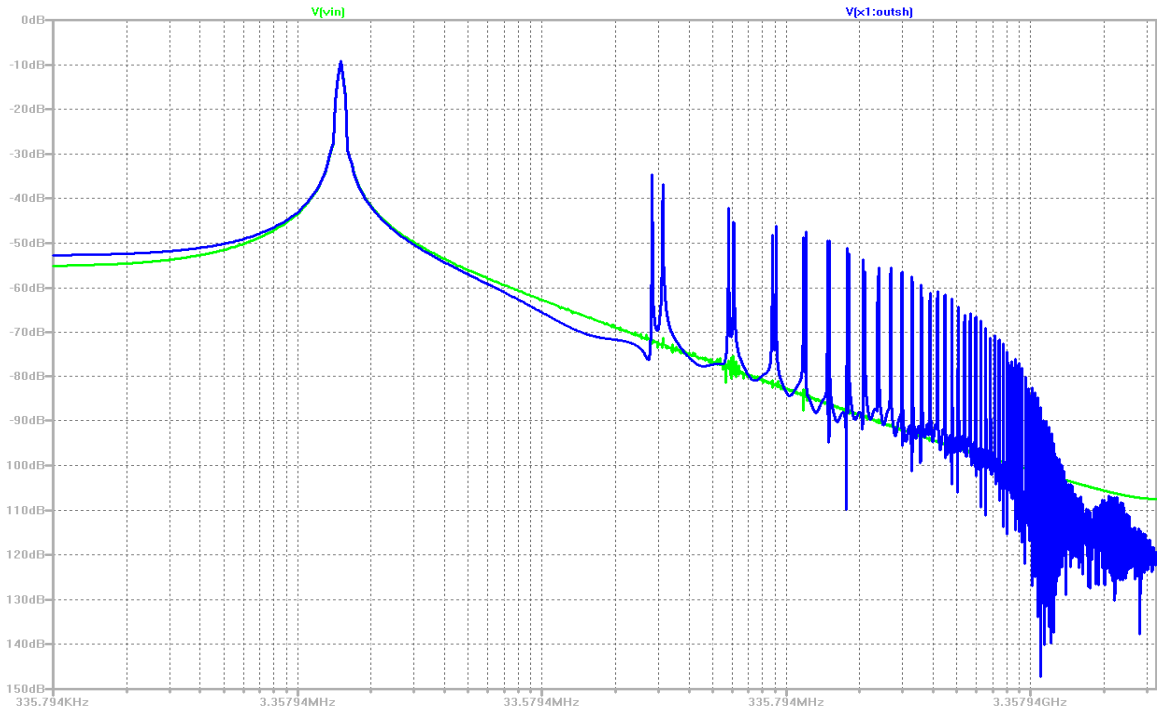


Fig. 1 Frequency spectrum(FFT) plot for q4.4

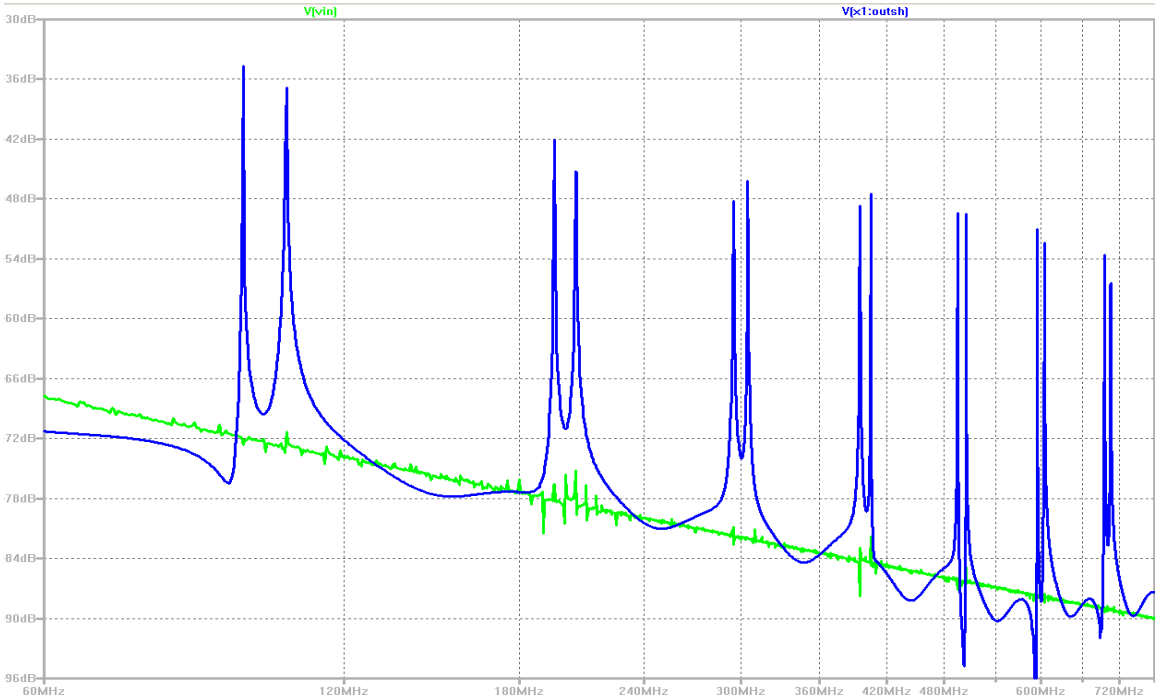


Fig. 1 Frequency spectrum(FFT) plot for q4.4 (zoomed in)

4.6) Suppose we think of the 1-bit input, 0 or 1, in fig. 4.9 as +1 or -1 (two's complement numbers). What is the output of the digital filter when the input is always 0? Is the magnitude response seen in fig. 4.10 correct? why?

The fig. 4.9 shows the implementation of a counter used as a digital averaging filter, with a 1-bit input.

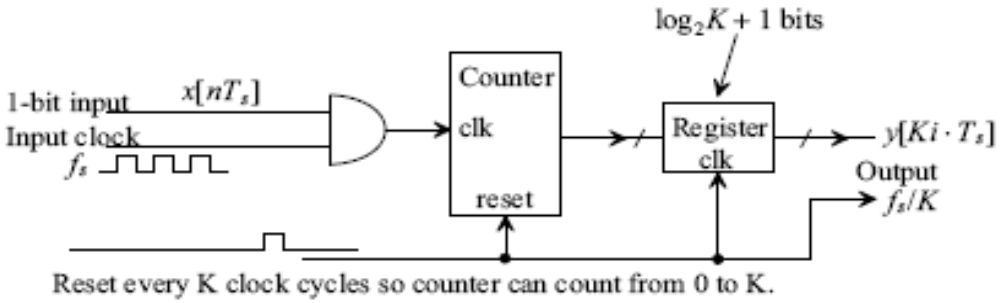


Figure 4.9 Using the counter as a digital filter.

The transfer function can be represented as:

$$H(z) = \frac{1 - z^{-K}}{1 - z^{-1}}$$

The magnitude of the frequency response for this transfer function is:

$$|H(z)| = \left| \frac{1 - e^{-j.2\pi K f/f_s}}{1 - e^{-j.2\pi f/f_s}} \right| = \left| \frac{\sin(\pi K f/f_s)}{\sin(\pi f/f_s)} \right|$$

$$\Rightarrow \quad |H(z)| = K \cdot \left| \frac{\text{sinc}(\pi K f/f_s)}{\text{sinc}(\pi f/f_s)} \right|$$

For a constant input of 0's, the counter response is going to be 0 – since the digital filter is essentially an averaging filter.

Similarly, for a constant DC input of 1's, the output would be 1 – or after K clock cycles, K. See fig. 4.10 below for the magnitude response.

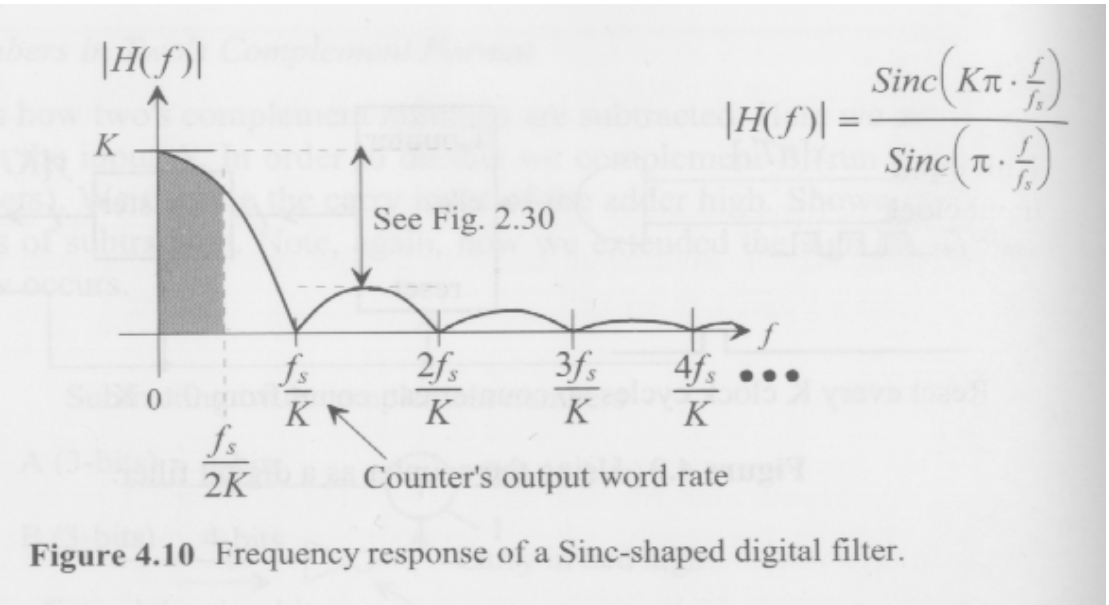


Figure 4.10 Frequency response of a Sinc-shaped digital filter.

Converting the inputs to 2's complement, 0 can be represented as 11. Hence, a stream of 0's is 11 + 11 + 11 + 11 + ….

For a sample case, K = 2, the inputs are 0 + 0

```
  0      ->    11     (2's complement)
+ 0      ->  + 11
             1 10
```

10 is the 2's complement representation of -2, which is –K for our sample case.

4.7 Suppose the 1-bit input signal seen in Fig. 4.9 is an alternating sequence of 101010... In terms of two's complement numbers, what is the output of the digital filter (what is the output of the counter)? What is the frequency of the input signal? Is the frequency response seen in Fig. 4.10 correct?

Solution:

Figs. 4.9 and 4.10 in [1] are redrawn in Figs. 1a and 1b respectively.

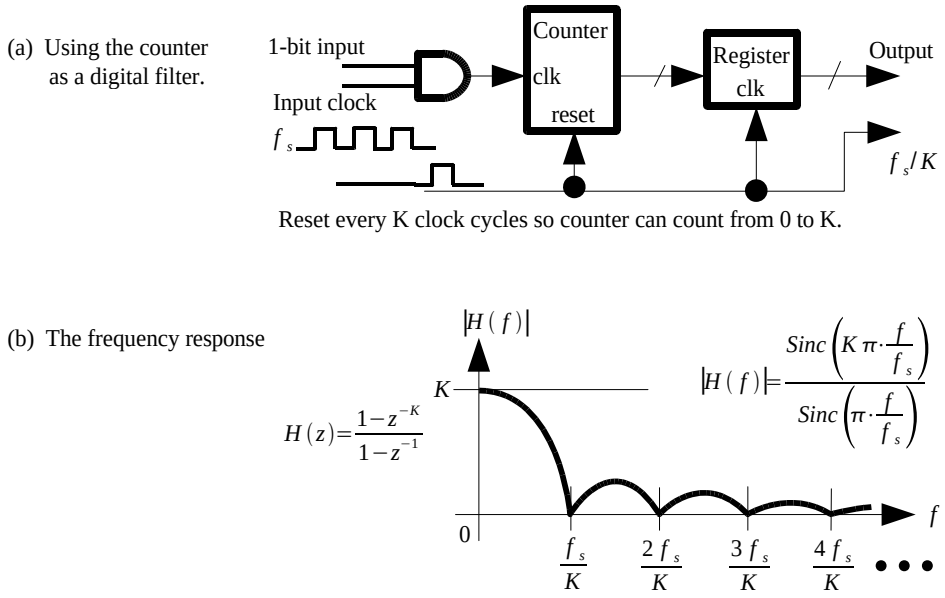


Figure 1 Using the counter as a digital filter.

If the input to the digital filter is an alternating sequence of 101010... the output will be $K/2$. Let's assume $K=8$ and the digital filter has 4-bit output then we can write the output code as

$$\text{Output} = \overbrace{\underset{(4)}{0100}}^{\text{Binary offset}} \rightarrow \overbrace{\underset{(-4)}{1100}}^{\text{Two's complement}} \tag{1}$$

And the counter repeatly output the code:

$$\underset{(0)}{0000} \rightarrow \underset{(1)}{0001} \rightarrow \underset{(2)}{0010} \rightarrow \underset{(3)}{0011} \rightarrow \underset{(4)}{0100} \tag{2}$$

Every time after generating an output code of $0100(4)$, the counter will be reset to $0000(0)$.

To understand the circuit operation, this filter is modeled using Verilog (the Verilog code is appended). Let's get started by simulating the circuit with a DC input of constant 1s. The timing diagram of this simulation is shown in Fig. 2. The register clocked in data every 8 clock cycles, or 80 ns. Every time the data is clocked in the register, the counter is reset. The register does not see the change at its input, or the counter's output value, so it's output is always $1000(8)$.

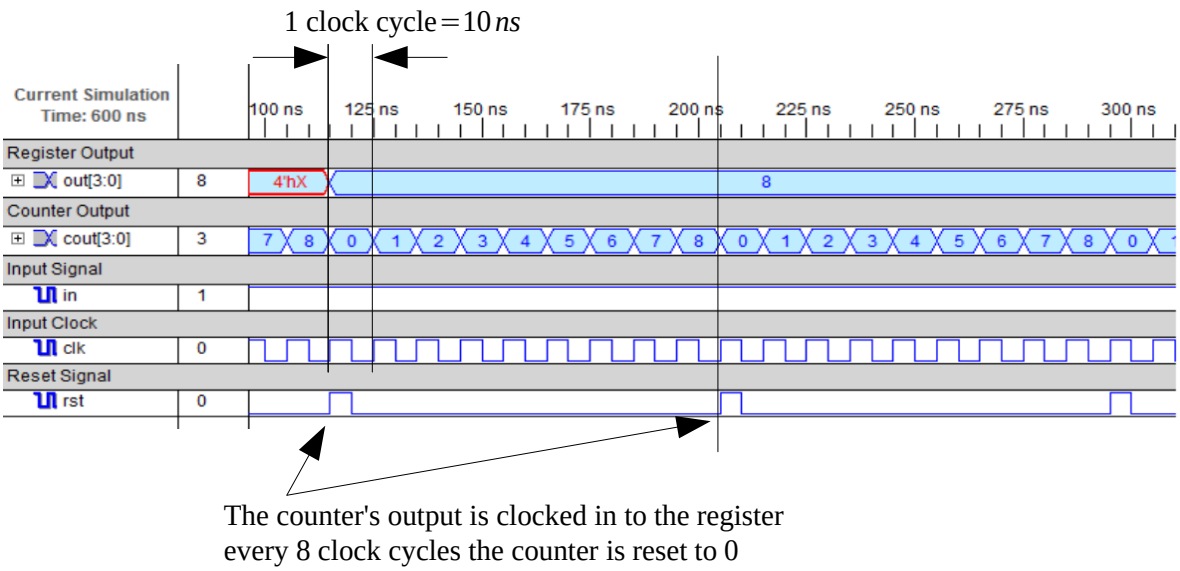


Figure 2 The outputs of the filter and the counter for a constant 1s DC input.

Now let's input a alternating sequence of 101010... The input and output timing diagram is shown in Fig. 3. The counter counts from 0 to 4 and then reset to 0. Again, the register does not see the change at the output of the counter and clocked in $0100(4)$ every time.

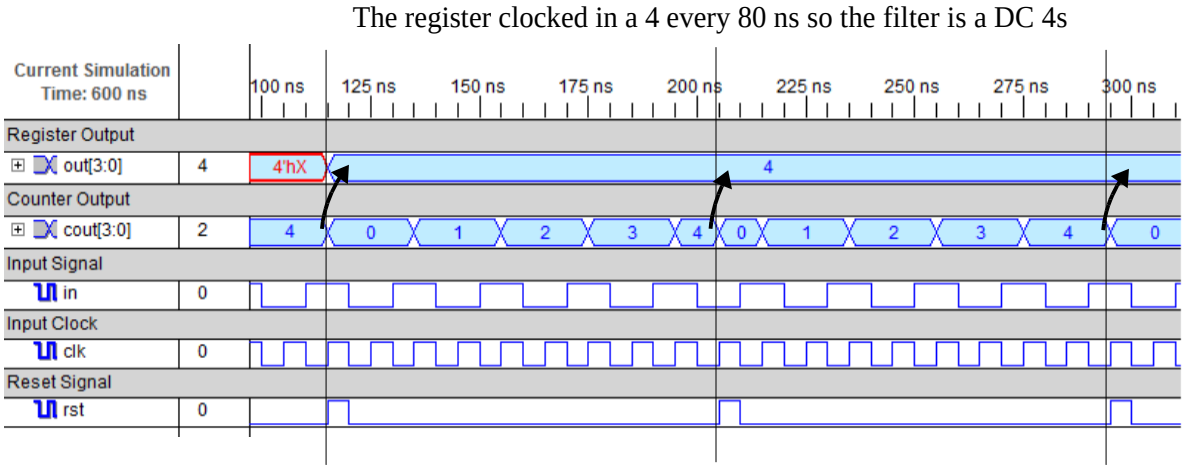


Figure 3 The output of the filter and the counter for an alternating sequence of 101010...

The input clock period is 10 ns, or $f_s = 100\,MHz$. We can see from Fig. 3 that the frequency of the input signal is $f_s/2$, or $50\,MHz$. Fig. 3a shows the input signal spectrum. Looking at this figure and the filter's frequency response in Fig. 1b, we may think that this input signal should be filtered out and the output of the digital filter is 0. However, since the counter's output is read out every 80 ns, the signal is resampled at $f_{s,new} = f_s/8$, or $12.5\,MHz$. This

sampling rate is lower than the input signal frequency, $50\,MHz$, thus aliasing occurs. Fig. 3b shows that the image originally associated with $400\,MHz$ is now moved to DC. This is why the filter outputs a constant value of 4. So the frequency response in Fig. 1b or Fig. 4.10 in [1] is correct.

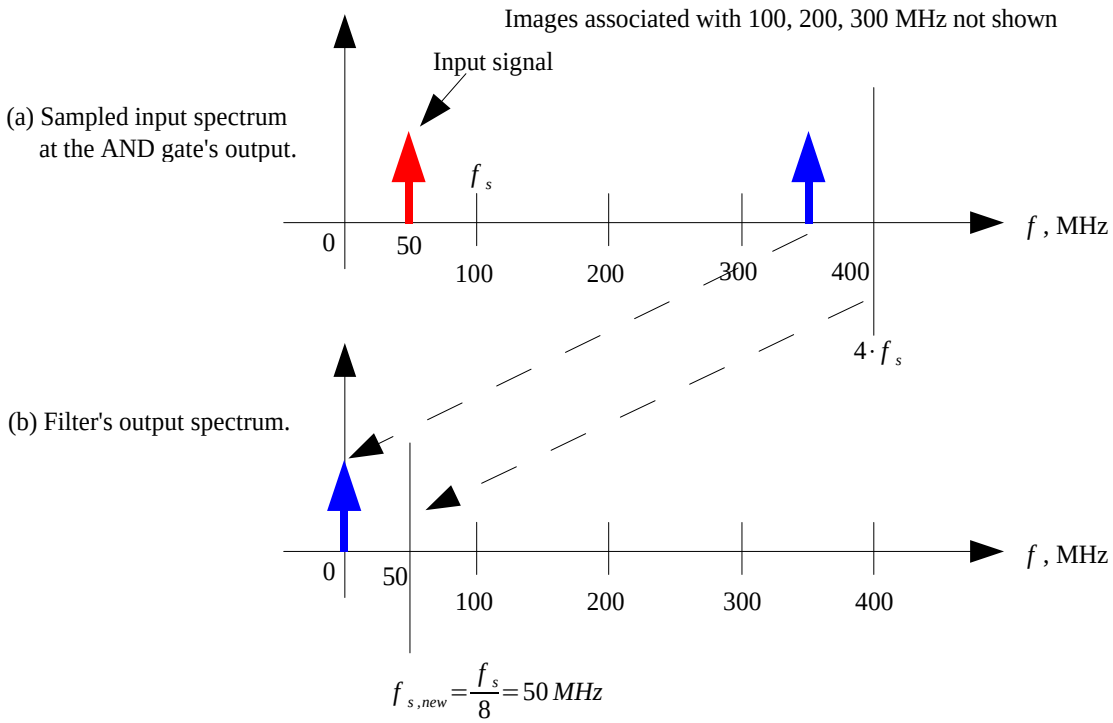


Figure 4 The input and output signal spectrum

Reference:

[1] R. J. Baker, *CMOS Mixed-signal Circuit Design, Second Edition*, Wiley-IEEE, 2009.

Verilog Code:

```
// fig_4_9.v
// Digital filter shown in Fig. 4.9
module fig_4_9(in, clk, rst, out, cout);
  input in;
  input clk;
  input rst;
  output [3:0] out;
  output [3:0] cout;

  //wire cin;
  reg [3:0] cout;
  reg [3:0] out;

  // 4-bit Counter
  always @( posedge rst, posedge clk) begin
    if (rst)
      cout <= 4'd0;
    else if(in)
      cout <= cout + 1'd1;
    else
      cout <= cout;
  end
  // Register
  always @(posedge rst) begin
    out <= cout;
  end
endmodule
```

```
// t_fig_4_9.v
// Simulate the filter
`timescale 1ns / 1ps
module t_fig_4_9;
        // Inputs
        reg in; reg clk; reg rst;
        // Outputs
        wire [3:0] out;   wire [3:0] cout;
        // Instantiate the Unit Under Test (UUT)
        fig_4_9 uut (
                .in(in),
                .clk(clk),
                .rst(rst),
                .out(out),
                .cout(cout)
          );
  parameter T = 10;
  initial begin
    clk = 1'b0;
    forever
      #(T/2) clk = ~clk;
  end
// input: 101010
/*   initial begin
    in = 1'b0;
    forever
      #(T) in = ~in;
  end*/
  // input: DC
  initial begin
    in = 1'b0;
  end
  initial begin
    rst = 1'b1;
    #(3.5*T);
    forever begin
      rst = 1'b0; #(T*8);
      rst = 1'b1; #(T/2);
      rst = 1'b0; #(T/2);
    end
  end
```

	/* initial begin #(30*T) $finish; end*/ endmodule

4.8) Repeat Ex. 4.3 for a filter with a transfer function of $\frac{1-z^{-7}}{1-z^{-1}}$. *Also plot the location of the filter's poles and zeroes in the z-plane.*

> *Ex. 4.3)*
> *Assuming an 8-bit word, sketch the implementation of the Sinc-shaped filter*
> $$\frac{1-z^{-8}}{1-z^{-1}}.$$
> *What is the output word size? Using SPICE, verify that the filter's frequency response is given by Eq. 4.13 with K=8. Assume f_s = 100MHz.*

Part I. Output word size and verification of Eq. 4.13

Following example 4.3, the output word size must be N + $\log_2$K. N = 8, K = 7. By properties of logarithms, $\log_2 7 = \log 7/\log 2 = 2.81$. Since we cannot have a non-integral number of bits we must round up to 3, so the output word size is then 8 + 3 = 11 bits. The filter is sketched in Fig. 4.8.1, which is a photocopy of Fig. 4.14 from the text, where z^{-8} has been replaced with z^{-7} in the Comb filter.

Figure 4.8.2 shows the filter's input and output when the input frequency is 7.14MHz $\left(= f_s/2K = 100MHz/2\cdot 7\right)$. Here the output magnitude is 0.224V_p with an input of 0.4V_p. The magnitude of output over input is 0.56. This is -5.04dB. This does not agree with what we expect. We expect it to be -3.9db, right? No. We really expect the output to be given by equation 4.12, approximated by Eq. 4.13. Equation 4.13 is $|H(f)| = K\cdot\left|Sinc\left(\pi\frac{Kf}{f_s}\right)\right|$.

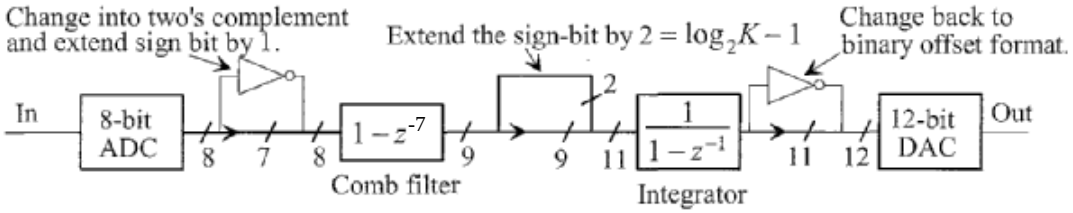


Figure 4.8. 1.Digital filter sketch for problem 4.8.

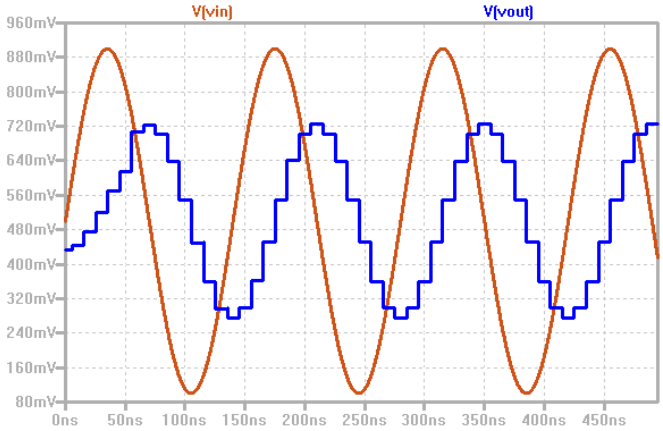


Figure 4.8. 2. Input and output of the filter in Fig. 4.8.1 at a frequency of 7.14MHz.

Using this, we find $|H(f)| = 4.48$. The difference comes by the fact that we have divided the output by 8, not 7 (extended the sign bit by three to prevent overflow). If we divide the expected output by 8 we get 4.48/8 = 0.56, or exactly what we measured in the simulations output.

Part II: finding the pole-zero z-plane plot.
We see that the transfer function can go through the following algebraic manipulation:

$$\frac{1-z^{-7}}{1-z^{-1}}\left(\frac{z^7}{z^7}\right) = \frac{z^7-1}{z^7-z^6} = \frac{z^7-1}{z^6(z-1)}.$$

The roots of polynomials of the form $z^k - 1$, for k>0, follow the form of having one root at x = 1 and have k equally spaced roots on the unit circle $x^2 + y^2 = 1$, with x being the real and y being the imaginary axes in the z-plane. If one wants to verify this, we can use a solver to find the roots of the numerator. Choosing MATLAB we would try the following code: roots([1 0 1]), using k-1 0's in between the leading and final 1's. In our example with k = 7 we would use: roots([1 0 0 0 0 0 0 1)]). The answer is:
-1.0000, -0.6235 + 0.7818i, -0.6235 - 0.7818i, 0.2225 + 0.9749i, 0.2225 - 0.9749i, 0.9010 + 0.4339i, 0.9010 - 0.4339i.

Rather than go through the trouble of plotting these by hand, however, let's use the pole-zero plot function of MATLAB with the following code,

```
%Plot pole-zeros for (1-z^-7)/(1-z^-1) = (z^7 - 1)/(z^7-z^6)
num = ([1 0 0 0 0 0 0 -1]); %numerator
den = ([1 1 0 0 0 0 0  0]); %denominator
sys = tf(num,den);          %transfer function(num/den)
%create the pole-zero plot
pzplot(sys)
axis([-1.25,1.25,-1.25,1.25])%format the axes
```

which produces the pole-zero plot of Fig. 4.8.3. The superscript 6 has been added post MATLAB processing. (Hard to see, 6 poles at origin, 1 at x=1.)

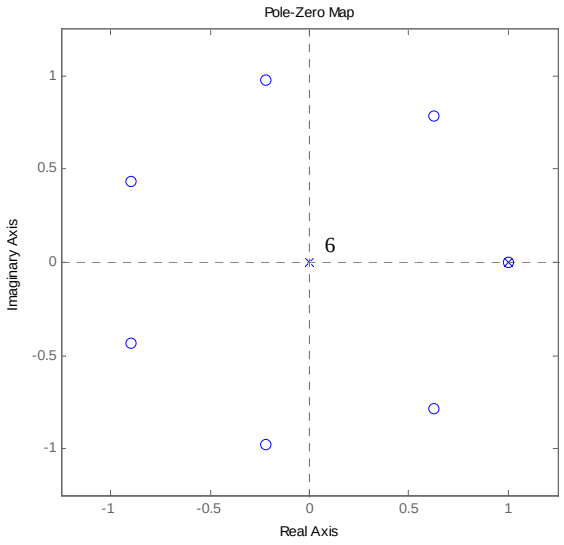
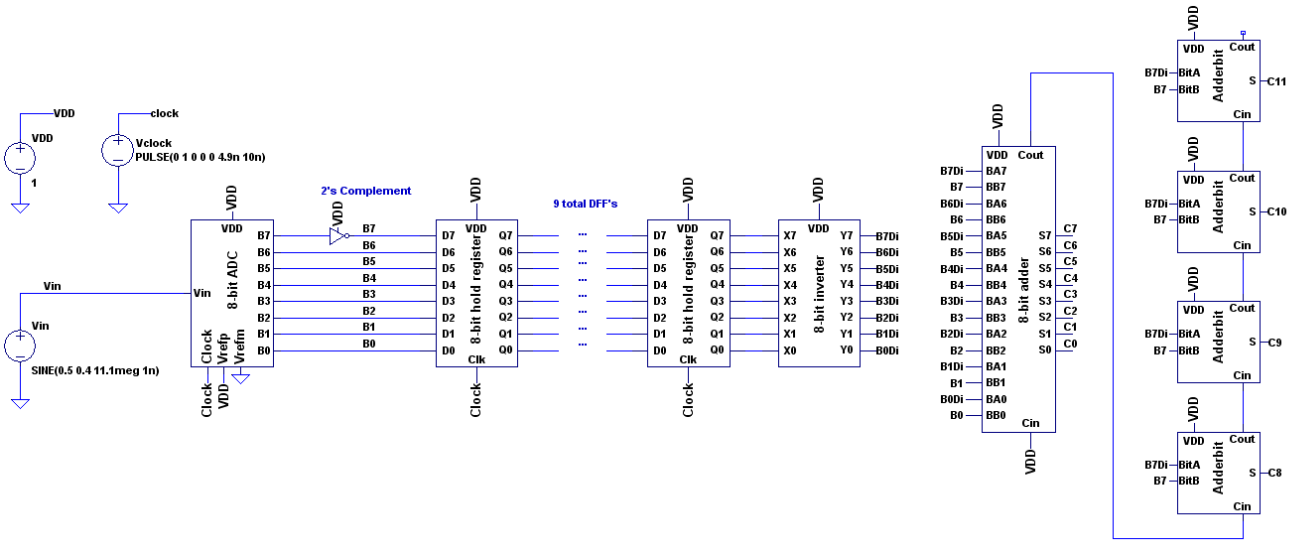


Figure 4.8. 3. Pole-zero plot for the filter in Fig. 4.8.1.

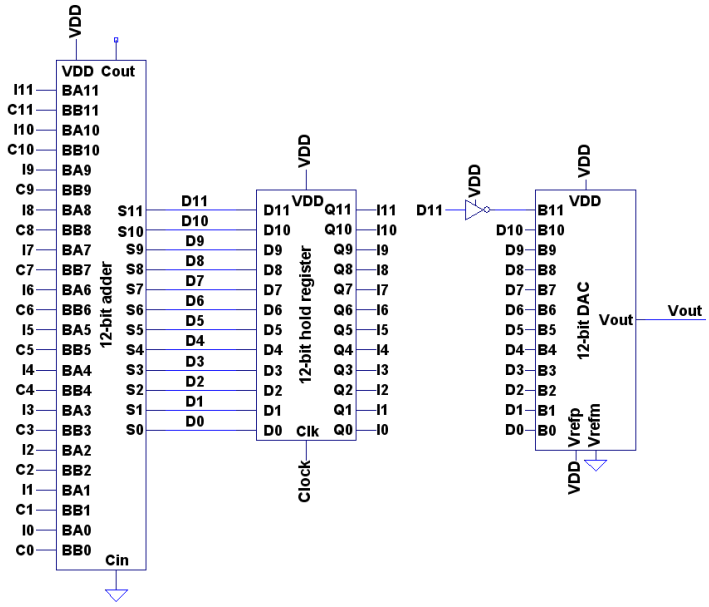
Jason Durand

Problem 4.9 – Repeat example 4.3 for a filter with a transfer function of $H(z) = \frac{1-z^{-9}}{1-z^{-1}}$.



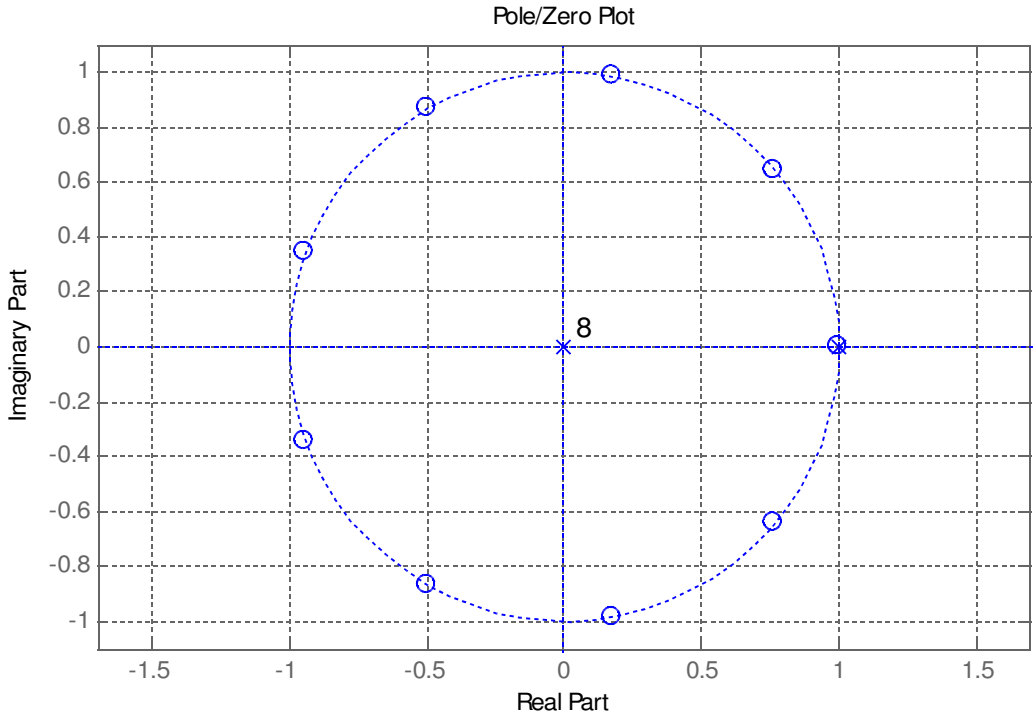
Example 4.3 has a 100Mhz sampling frequency, which will be used in this circuit also. The transfer function, $H(z) = \frac{1-z^{-9}}{1-z^{-1}}$, can be implemented easily when split into two blocks in series, a comb filter part and an integrator. The comb filter, $H(z) = 1-z^{-9}$, consists of series connected delay blocks, which are d flip-flops, and an adder. The adder sums the input signal from the ADC to the opposite of the delay block (it performs a subtraction). The schematic shows the subtraction setup, note the conversion to two's complement and the inversion with carry-in tied high to subtract two's complement numbers.

The next block implements $H(z) = \frac{1}{1-z^{-1}}$, which is the sum of the output of the previous stage (C0-C11) and a one clock cycle delay of the adder output (I0-I11).

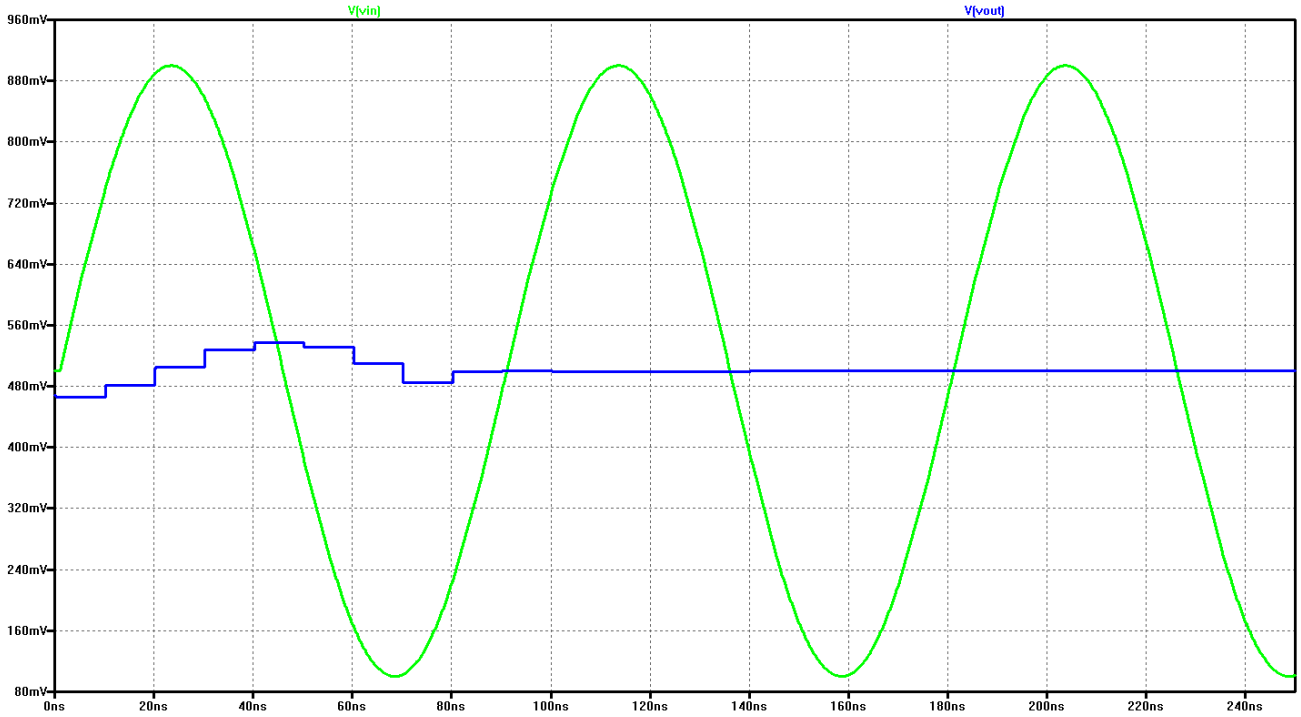


Before the integrator block is added to the filter, the word size of the digital signal must be increased by 4 bits to (or more accurately, $8 + \log_2(9) = 3.17$, which is rounded up to 4) to allow for the maximum size of the moving average sum (the comb filter). The output is finally run through a 12 Bit DAC to easily see the analog output.

Pole Zero plot of transfer function:



The first zero in the transfer function, as seen by the pole-zero plot, it at the frequency $f_s/9$, or for this problem, 11.1MHz. The figure below is an 11.1MHz sine wave and the corresponding output, confirming the location of the zero.



Que. 4.10: Repeat Ex. 4.5 if *L* is increased to 3.

Solution: Based on the question and looking at Ex. 4.5, the time domain impulse response of an averaging filter with K=8 has to be determined which is cascade 3 times or L=3. First the solution will discuss the averaging filter (Lowpass sinc response), the time domain impulse response and then cascading of the averaging (sinc response) filters. The solution of problem will be followed in the end;

Note: Solution interchanges the word Lowpass sinc response filter and averaging filter, which is similar for this discussion.

Frequency response of (Lowpass sinc response) averaging filter

The Lowpass sinc response of an averaging filter can be implemented without decimation inherent in a counter or an accumulate-and-dump. Figure 1 shows the cascading of a comb filter with an integrator block to achieve the averaging without decimation. The transfer function of the system in Figure 1 given as;

$$H(z) = \frac{1 - z^{-K}}{1 - z^{-1}}$$

The equation above is similar to the transfer function of a counter or an accumulate-and-dump system [please refer section 4.2.1 page 126]. In this problem value of K=8. Both the components are clocked with same frequency.

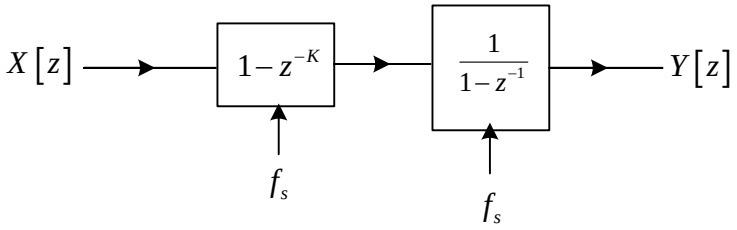


Figure 1 Block diagram implementation of Low pass sinc filter with comb filter and an integrator

Figure 2 shows the magnitude-frequency response of the system when K=8 with pole and zeroes plot. Notice that the bandwidth of the signal of interest is given by Nyquist frequency of $f_n = f_s/2K$; in this case it is given as $B = f_n = f_s/16$.

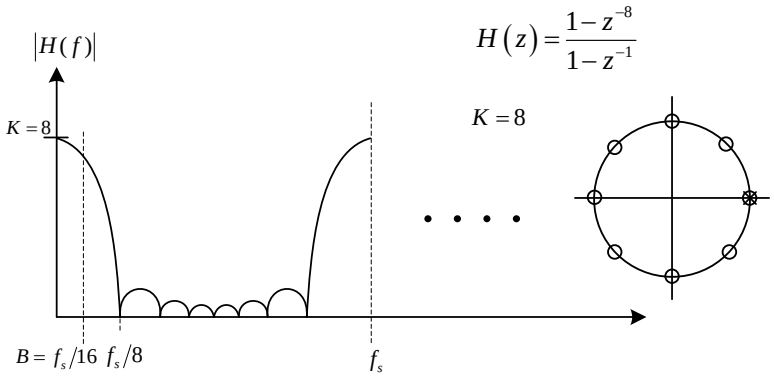


Figure 2 Lowpass Sinc-response filter with K=8

Time domain impulse response of averaging filter

The transfer function of the filter can be expanded to a sum of individual delay in terms of z as shown below

$$H(z)=\frac{1-z^{-8}}{1-z^{-1}}=1+z^{-1}+z^{-2}+z^{-3}+z^{-4}+z^{-5}+z^{-6}+z^{-7}$$

The time domain relationship is given by

$$y[nT_s]=x[nT_s]+x[(n-1)T_s]+x[(n-2)T_s]+\cdots\cdots+x[(n-7)T_s]$$

Thus the output is simply sum of the inputs over time KT_s. The time domain impulse response of this first order averaging filter is shown in Figure 3. The rectangular shape of the impulse response can explain the sinc behavior in frequency domain,

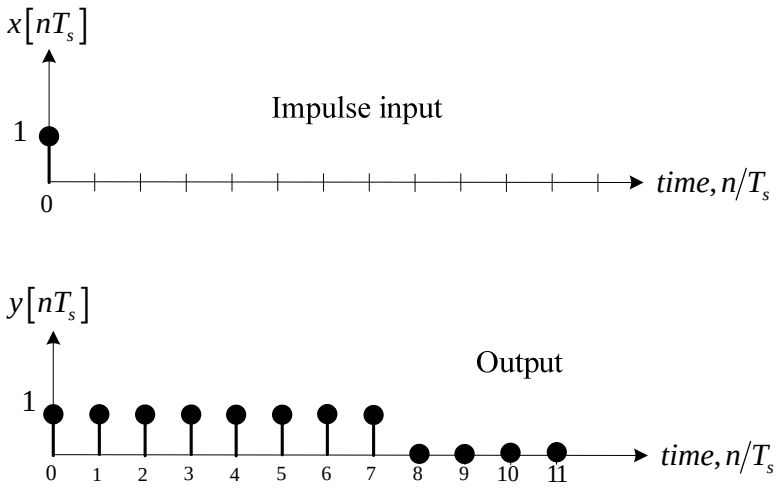


Figure 3 Impulse response of a K=8 averaging filter

Cascading Sinc (averaging) filter

The transfer function of a cascade of *L* of these sinc filters will be written as

$$H(z)=\left[\frac{1-z^{-K}}{1-z^{-1}}\right]^L$$

Frequency domain equation can be written as

$$\left|H(f)\right|=K^L\cdot\left[\frac{Sinc\left(K\pi\frac{f}{f_s}\right)}{Sinc\left(\pi\frac{f}{f_s}\right)}\right]^L$$

The block diagram of cascade of *L* such filter can be seen in figure 4.

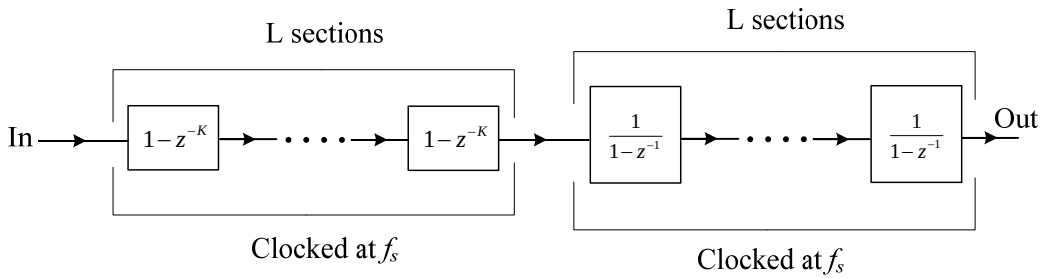


Figure 4 Block diagram of a cascaded of L sinc filter with K=8

Problem solution for *L*=3

In the problem it is asked to determine and sketch time domain impulse response of this type of sinc filter where *L*=3 for cascade. From equation of *L* cascaded filter, putting the value of *L*=3, the transfer function of filter is as shown below, where K=8 for the problem.

$$H(z)=\left[\frac{1-z^{-8}}{1-z^{-1}}\right]^3$$

Figure 5 shows the block diagram of the filter

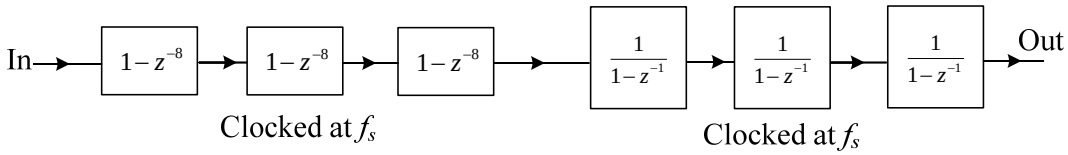


Figure 5 Block diagram of *L*=3 cascade sinc filter

The transfer function of the sinc filter as shown above can be expanded in similar way as it was done for the first order (*L*=1) filter

$$H(z)=\left[\frac{1-z^{-8}}{1-z^{-1}}\right]^3=\left[\left(1+z^{-1}\right)\left(1+z^{-2}\right)\left(1+z^{-4}\right)\right]^3$$

Further expansion gives

$$H(z)=1+3z^{-1}+6z^{-2}+10z^{-3}+15z^{-4}+21z^{-5}+\cdots\cdots+15z^{-17}+10z^{-18}+6z^{-19}+3z^{-20}+z^{-21}$$

The time domain relationship is given by

$$y[nT_s]=x[nT_s]+3x[(n-1)T_s]+6x[(n-2)T_s]+10x[(n-3)T_s]+\cdots\cdots+3x[(n-20)T_s]+x[(n-21)T_s]$$

The time domain impulse response of this third order averaging filter is shown in Figure 6.

Notice the near triangular shape of the output impulse as in case of *L*=2 (Ex. 4.5 page 133).

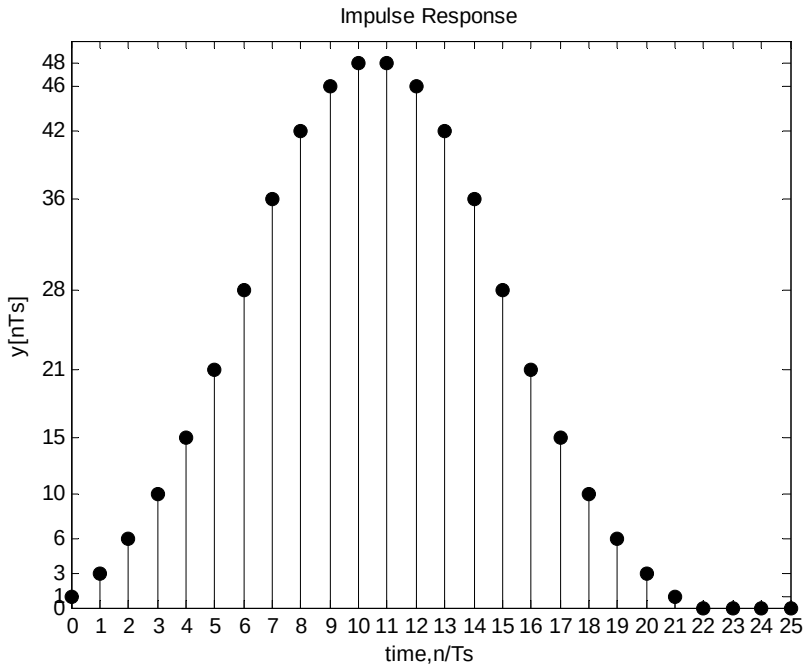


Figure 6 Impulse response of an *L*=3 filter with K=8

The filter response (L=3) last approximately thrice as long as in case of L=1, with 22 samples.

4.11 – Sketch the impulse response of the filter seen in Figure 4.19 (Tyler Hansen)

The filter in the book figure 4.19 is a high-pass filter with the transfer function seen in Figure 1. Figure 1 also shows how this transfer function is composed of an 8 delay differencer comb filter, followed by a single delay resonator. The block diagrams are also shown.

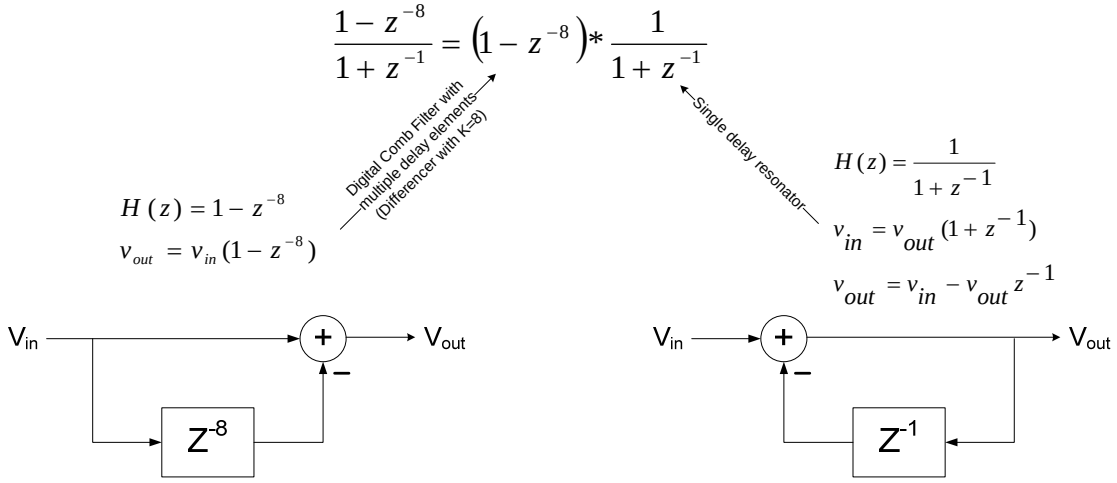


Figure 1: Transfer function breakdown of a high-pass filter, and the block diagrams of the individual components.

Figure 2 below shows the impulse response for each component of the system. Note that the time domain impulse response is defined as the result of running a single impulse at time 0 into the system and plotting the output. Since I have elected to use digital delays, the impulse response is shown at each successive clock after the input is given.

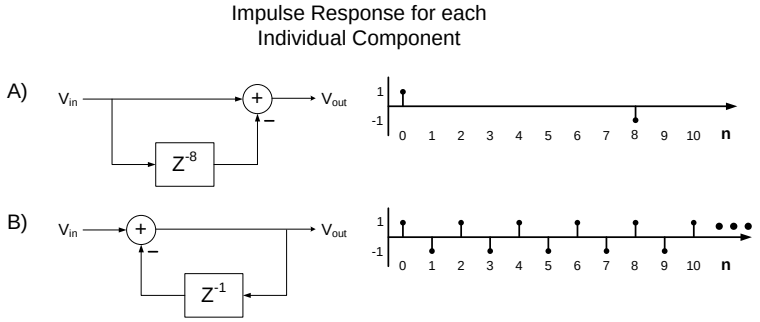


Figure 2: Impulse response given for each individual component of the high-pass filter. Namely, an 8 delay differencer, and a single delay resonator.

The impulse response of the system can be evaluated by connecting the two components of our system, and looking at the impulse response. A pulse at n=0 will pass unmolested to the output. At n=1, the input goes to zero and remains at zero throughout the evaluation (the definition of an impulse response). Also, we note that at n=1, the input signal of 1 has passed through one of the delays in the 8-stage differencer. Also, the original input of 1 is subtracted from the input due to the characteristics of the resonator.

Therefore, we can evaluate the response at n=1 with the equation 0-1 = -1. At n=2, the input remains at 0, the input pulse passes through the 2nd delay in the differencer, and the previous output is subtracted from 0. The response at n=2 is evaluated as 0-(-1) = 1.

The output continues to oscillate between 1 and negative 1 until something changes. The change takes place after 8 clock cycles when the original input of 1 has passed through all 8 of the differencer delays, and is subtracted from the input before the resonator response is considered. The 2-stage evaluation happens at n=8. The stage 1 equation is 0-1 = -1. Then, the resonator response to an input of -1 is: -1 minus the output of the system at n=7. From figure 2, we see that the output at n=7 was -1. Therefore, the equation is -1-(-1) = 0. At n=8, the output returns to zero, and there are no longer any signals in the system to cause any changes. The results of this analysis can be seen in Figure 3 below.

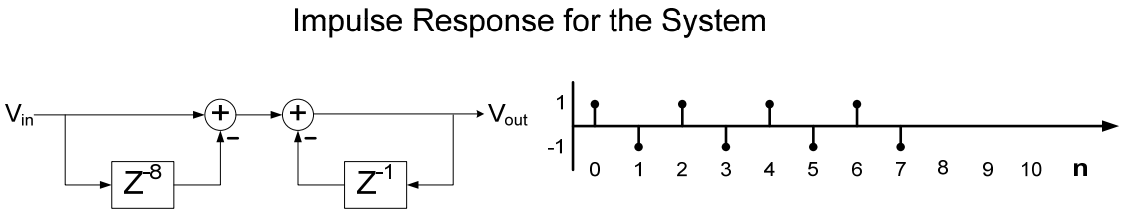


Figure 3: Impulse response of the high-pass filter.

Even though the question has been suitably answered at this point, for fun, let's see if we can confirm this result using LTspice simulations. Modifying the schematic used to produce the graph in the book figure 4.25 to reflect a resonator of $\frac{1}{1+z^{-1}}$ instead of $\frac{1}{1+z^{-2}}$ we have the schematics in Figure 4.

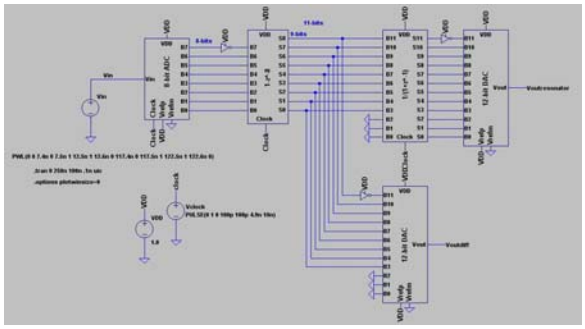


Figure 4: Book LTSpice schematic 4.25 modified to produce the transfer function in the question.

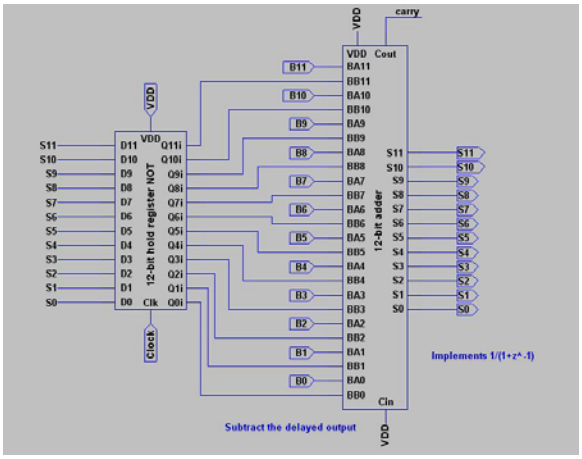


Figure 5: Schematic for the single delay resonator component of the overall high-pass filter.

The simulation is run using a clock period of 10ns. At 20ns I will give a “1” signal to the input in the form of a unity magnitude pulse of 10ns width. Then I will be able to monitor the outputs of the differencer, and the entire system to confirm my hand derivations. The simulation results are presented in Figure 6.

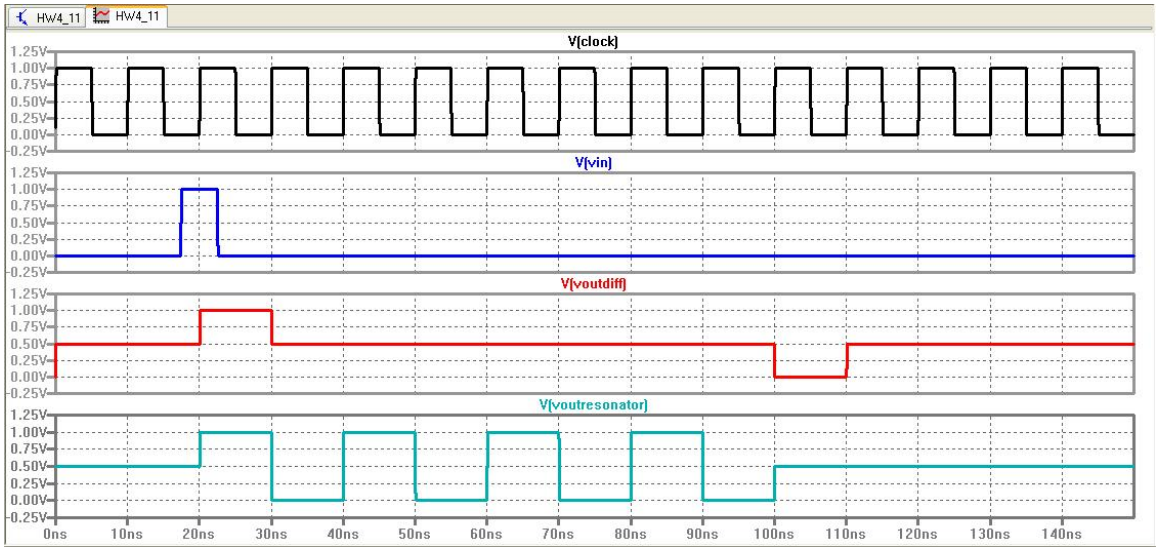


Figure 6: Waveforms from the impulse response simulation.

The simulation results are as expected! Correlate the output of the system (bottom waveform in Figure 6) with the expected results given in Figure 3. 80ns after in input pulse, the output returns to zero. The differencer is behaving, showing the expected instant output, and the negative value resurfacing 8 clock cycles later.

4.12) What are the transfer functions of the bandpass filters, indicated in Fig. 4.22, with center frequencies of $f_s/6$ and $f_s/4$? Sketch the frequency responses and the locations of their poles and zeroes in the z-plane.

For convenience to the reader, Fig. 4.22 from the text is reproduced here.

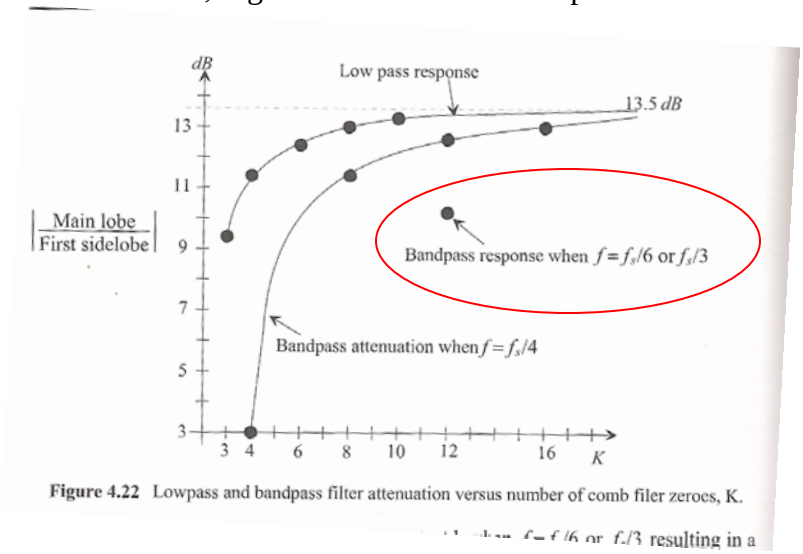


Figure 4.22 Lowpass and bandpass filter attenuation versus number of comb filer zeroes, K.

The figure indicates only one filter for each of the two listed center frequencies, namely, filters with K=12, circled in red. As the reader is likely aware of at this point, the sinc filter with K zeros acts as a lowpass filter, attenuating the magnitude of input signal from its maximum level at f=0 to its minimum (being zero) at f_s/K. Looking at the transfer function of the sinc filter in (4.12.1) we can get an understanding of why this occurs.

$$H(z) = \frac{1 - z^{-K}}{1 - z^{-1}} \tag{4.12.1}$$

Multiplying the numerator and denominator by z^K reveals that there are K equally spaced zeros around the $z = e^{j2\pi f \cdot T_s}$ unit circle, with one of them at $f = 0$, and also K-1 poles at the origin and one pole at $f = 0$. As we know, from the technique of examination of the z-plane plot, the magnitude response can be determined by the distance from the frequency in question to the zeros divided by the distance from the frequency in question to the poles. If there had not been a pole at $f = 0$ then the distance to the zero would have been zero, causing a magnitude of 0. However, we have canceled the zero at $f = 0$ by placing a pole at the same location. Now this is *the* location on the $z = e^{j2\pi f \cdot T_s}$ unit circle that has the largest distance from the zeros and hence the largest magnitude response.

Understanding this concept is the key to designing bandpass filters. Instead of deriving a transfer function that cancels a zero at $f = 0$, we want to derive one that cancels the zero at the desired bandpass frequency. An artifact of the design (at least using techniques presented in chapter 4) is that we get a second bandpass at $f_s - f$. This should not present a problem, however, as this second pass band is above the Nyquist frequency and would generally not be used (filtered out by an anti aliasing filter).

For the lowpass (Sinc) filter, the numerator is a comb filter and the denominator is a digital integrator. For the bandpass filters we still have the comb filter but the denominator is referred to as a digital resonator. The design equation for the digital resonator is

$$H_D(z) = \frac{1}{1 - 2\cos\left[2\pi \frac{f}{f_s}\right] \cdot z^{-1} + z^{-2}}. \qquad (4.12.2)$$

Plugging in $f = f_s/6$ we find that we are taking $\cos(\pi/3)$ which evaluates to 0.5. Therefore, the design equation for the digital resonator at $f = f_s/6$ is

$$H_D(z) = \frac{1}{1 - z^{-1} + z^{-2}}, \quad f = f_s/6. \qquad (4.12.3)$$

Similarly for $f = f_s/3$, the cosine term evaluates to -0.5 and the resonator equation is

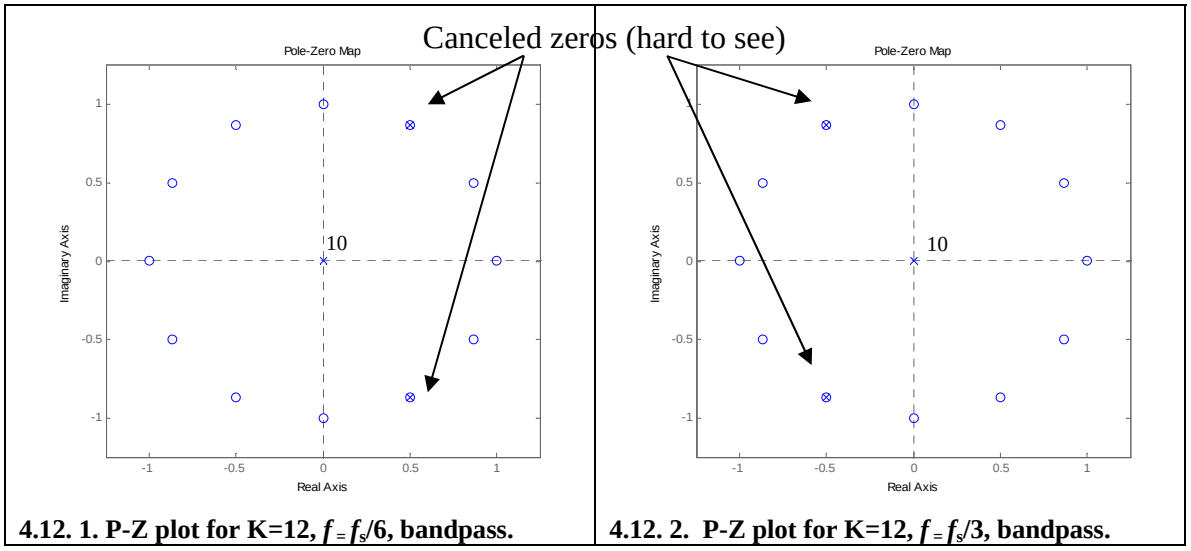
$$H_D(z) = \frac{1}{1 + z^{-1} + z^{-2}}, \quad f = f_s/3. \qquad (4.12.4)$$

As mentioned earlier, Fig. 4.22 from the text indicates that for these filters K=12. Using this, we can put together the complete transfer functions shown in (4.12.5) and (4.12.6).

$$H(z) = \frac{1 - z^{-12}}{1 - z^{-1} + z^{-2}}, \quad f = f_s/6 \qquad (4.12.5)$$

$$H(z) = \frac{1 - z^{-12}}{1 + z^{-1} + z^{-2}}, \quad f = f_s/3 \qquad (4.12.6)$$

Figures 4.12.1 and 4.12.2 show the z-plane plots. Notice the cancelling of the zeros.



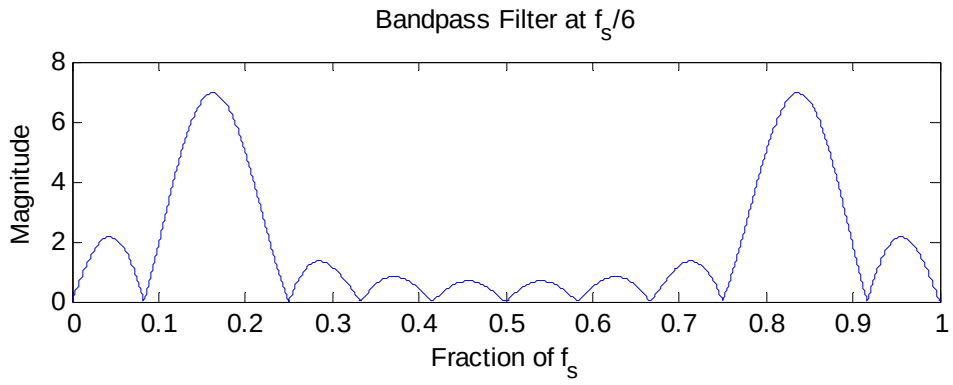
4.12. 1. P-Z plot for K=12, $f = f_s/6$, bandpass.

4.12. 2. P-Z plot for K=12, $f = f_s/3$, bandpass.

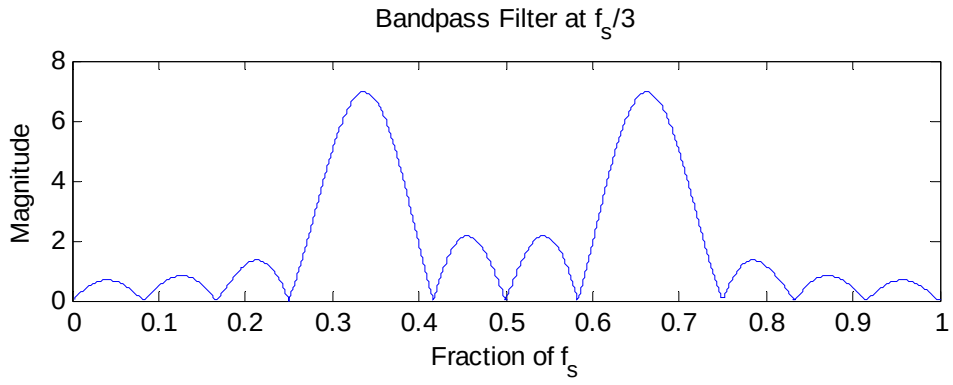
The frequency responses are shown in Figs. 4.12.3 and 4.12.4 and were produced in MATLAB with the following code:

```
f = 0:1e-4:1;
fs = 1;
z = exp(j*2*pi*f/fs);
H = abs((z.^12 - 1)./(z.^12-z.^11+z.^10));
plot(f,H);
title 'Bandpass Filter at f_s/6';
xlabel 'Fraction of f_s', ylabel 'Magnitude';

figure;
H = abs((z.^12 - 1)./(z.^12+z.^11+z.^10));
plot(f,H);
title 'Bandpass Filter at f_s/3';
xlabel 'Fraction of f_s', ylabel 'Magnitude';
```

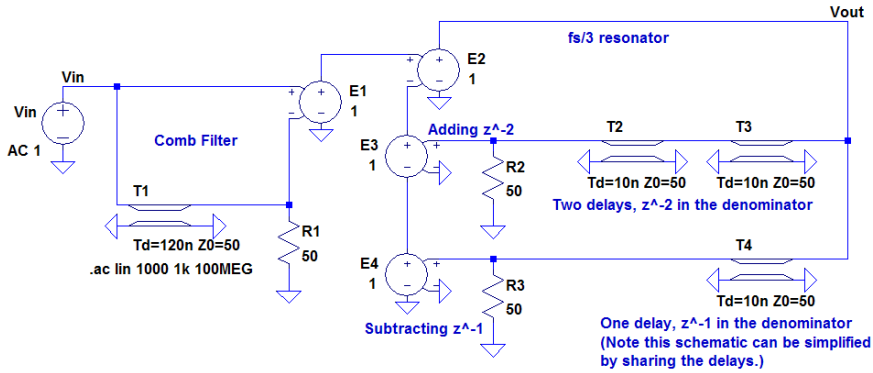


4.12. 3. $f_s/6$ bandpass filter frequency response over a fraction of the sampling frequency.

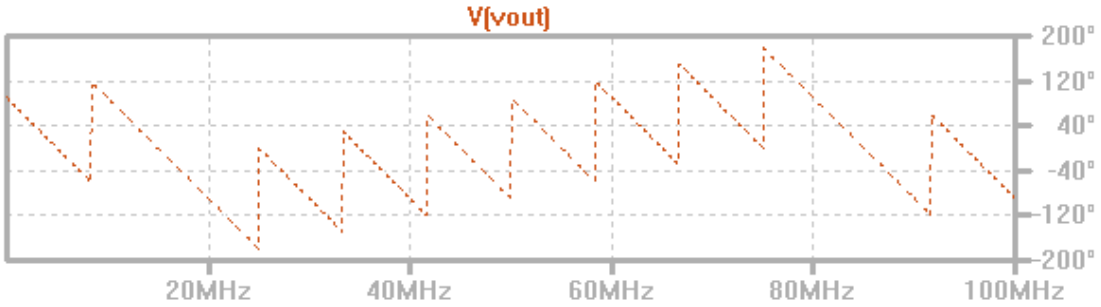


4.12. 4 $f_s/3$ bandpass filter frequency response over a fraction of the sampling frequency.

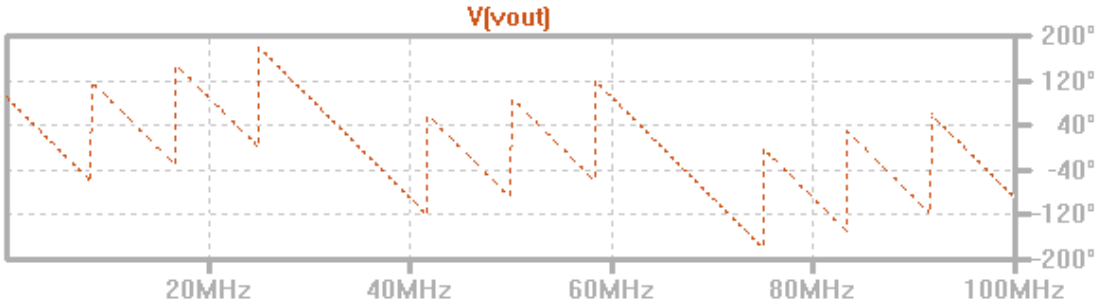
The phase was found using transmission gates for delay elements and unity gain voltage controlled voltage sources (ideal amps) for the summing junctions. These plot the magnitude as well, but I already showed that with MATLAB. The circuit used for the $f_s/3$ bandpass filter is shown if Fig. 4.12.5. and might be found in the books webpage LTSpice files for chapter 4. The phase responses are shown in Figs. 4.12.6 and 4.12.7.



4.12. 5. $f_x/3$ bandpass filter implemented with transmission gates and ideal unity gain amps; from the LTSpice files found in the books webpage.



4.12. 6. $f_x/6$ bandpass filter phase response. f_s = 100MHz.



4.12. 7 $f_x/3$ bandpass filter phase response. f_s = 100MHz.

4.13 Simulate, using an ideal 8-bit ADC on the input, and an ideal DAC on the output (calculate the size of the DAC), the operation of the digital resonator seen in Fig.23.

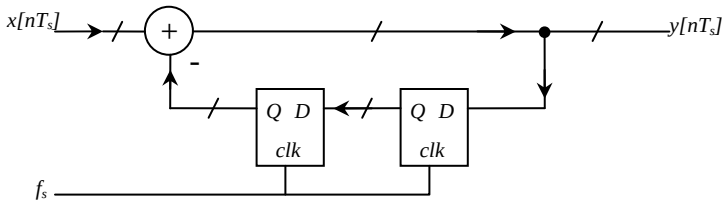
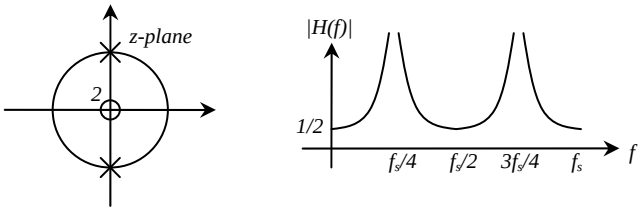


Figure 4.23 Implementation of a digital resonator.

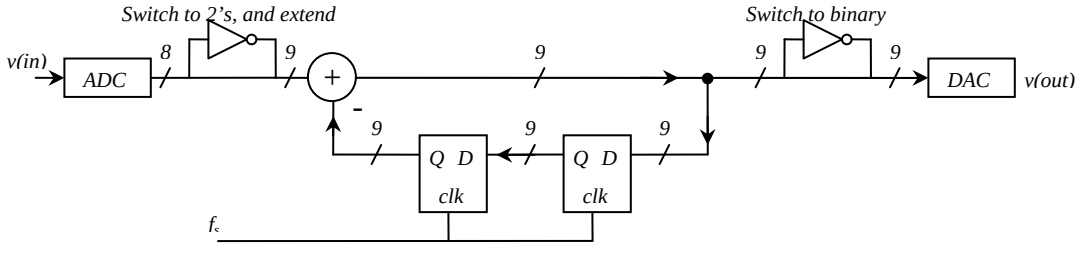
Before we simulate let's make sure we understand the operation of this circuit and can draw the frequency response. This allows us to pick the correct operating points of our simulation, and to determine if they are valid.

$$H(z) = \frac{1}{1 + z^{-2}}$$



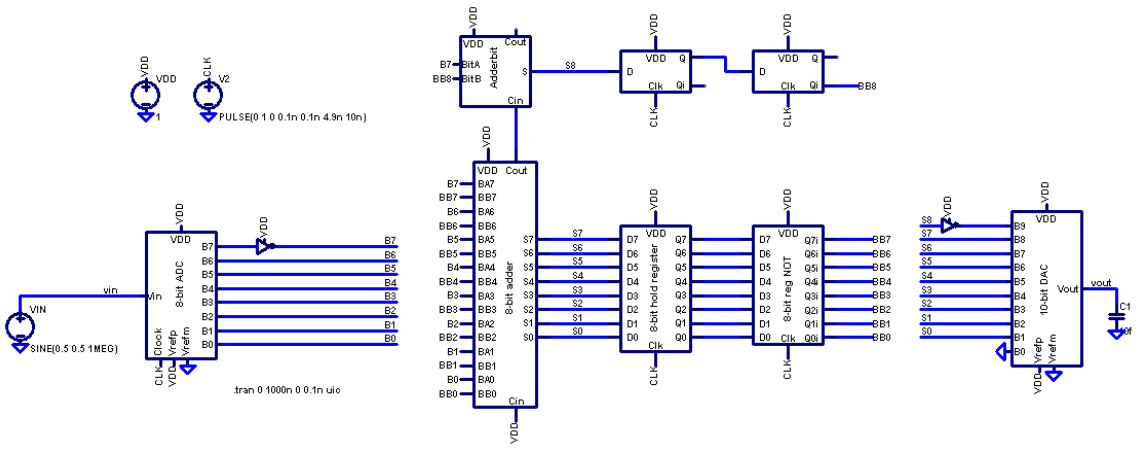
F-1 Z-Plane and frequency response of the digital resonator

We should be comfortable determining the z-plane and frequency response of the transfer function. To simulate this resonator let's determine the number of bits that are needed for each step:



F-2 Determining the size of the counters, DAC, and delay elements

F-2 shows how we implement the circuit. To prevent harmful overflow we need to extend the 8-bit to 9-bits before "adding" the two together. We also convert the output of the ADC to a 2's complement number, by inverting its' MSB. This means that we need at least a 9-bit DAC.

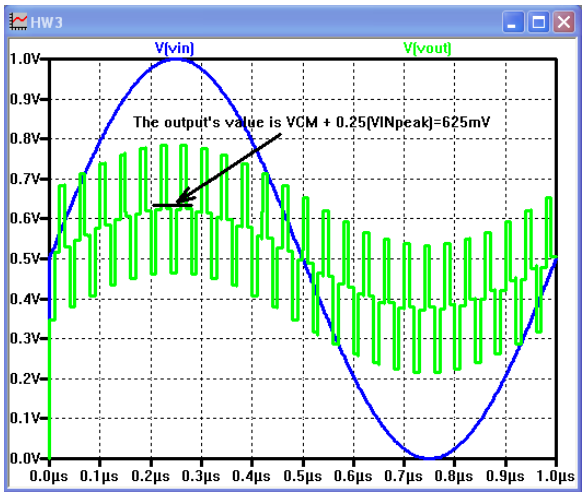


F-3 LTSPICE implementation of the resonator using ideal components

We use ideal components available at http://cmosedu.com/cmos2/book2.htm (see Ch4_MSD_LTspice.zip) to implement our resonator. There are not any 9-bit elements in our zip files, so to we create 9-bit elements and use a 10-bit DAC (with B0 tied low) to implement our resonator.

When we extend the sign bit we also are reducing our output by 0.5, so we can expect a scaled response on our output. Notice how we use an 8-bit adder with an extra adder-bit to implement a 9-bit adder. The same is true for the 8-bit registers. To implement the subtraction we invert the 9-bits being subtracted and tie the carry in bit of the adder to VDD.

To verify the functionality of our design let us simulate at ~DC (1MHz), we expect an output of 0.25 times the input peak voltage at DC from F-2:



Kaijun Li
Problem 4.14
Qualitatively explain why the desired spectrum of an input signal can't be increased by passing data through an interpolator. Using the simulations given in Ex. 4.8, verify that this is indeed the case.

Solution:
When we pass a signal through an interpolator, the signal's spectrum can't be increased, and what's happening is the sampling frequency f_s is increased.

Let us investigate Ex. 4.8, and we can prove that this is true. Figure 1 shows that the input signal generated by ideal voltage source is sampled and held. Then the output of sample-and-hold stage is fed into an up-sampler and an image removal filter.

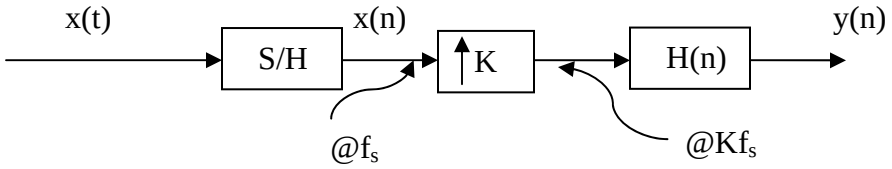


Figure 1. The block diagram of Ex. 4.8

For sample-and-hold (S/H) stage, the signal suffers a sinc-response low-pass filter which is drawn in Figure 2.

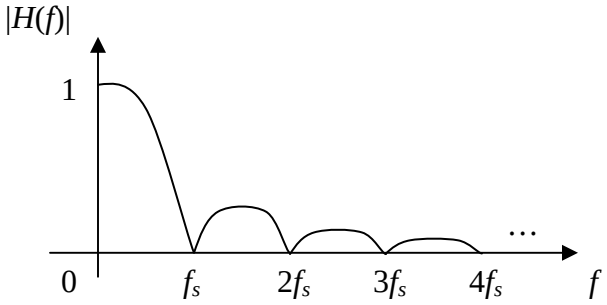


Figure 2. The frequency response of a S/H

The signal's spectrum after S/H will be repeated at every sampling frequency f_s (in Ex. 4.8, f_s is 12.5MHz) as seen in Figure 3. After the signal is passed through an up-sampler, the new sampling frequency $f_{s,new}=Kf_s$ (in Ex. 4.8, K is 8, and $f_{s,new}$ is 100MHz), and the signal's spectrum will be centered around $n\,f_{s,new}$ with n is any integer number.

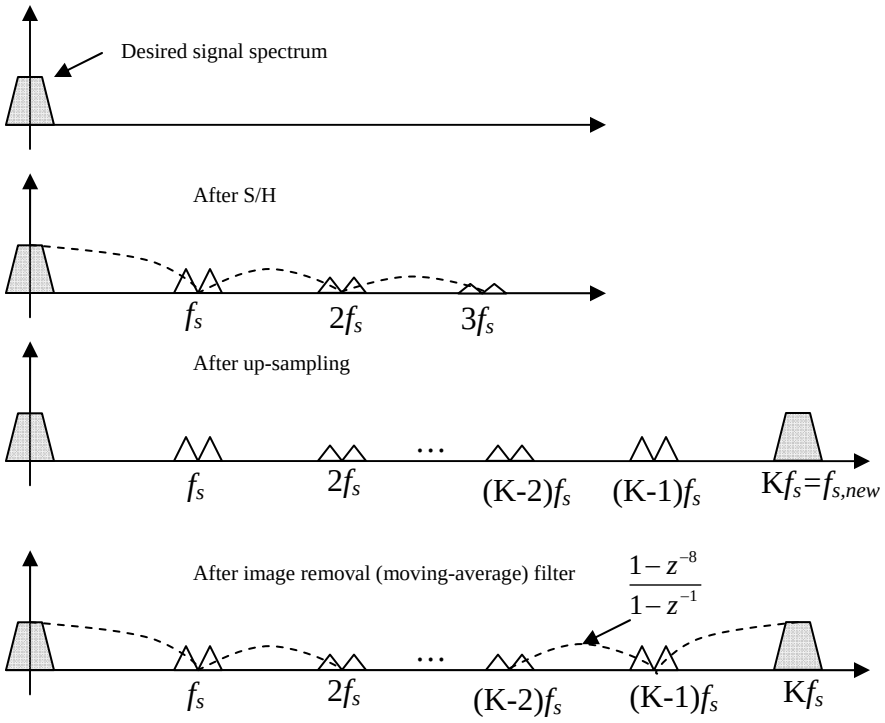


Figure 3. The spectrum of Ex. 4.8

The Figure 4.29 is re-simulated for 16us to get enough data for FFT plot in Figure 5.

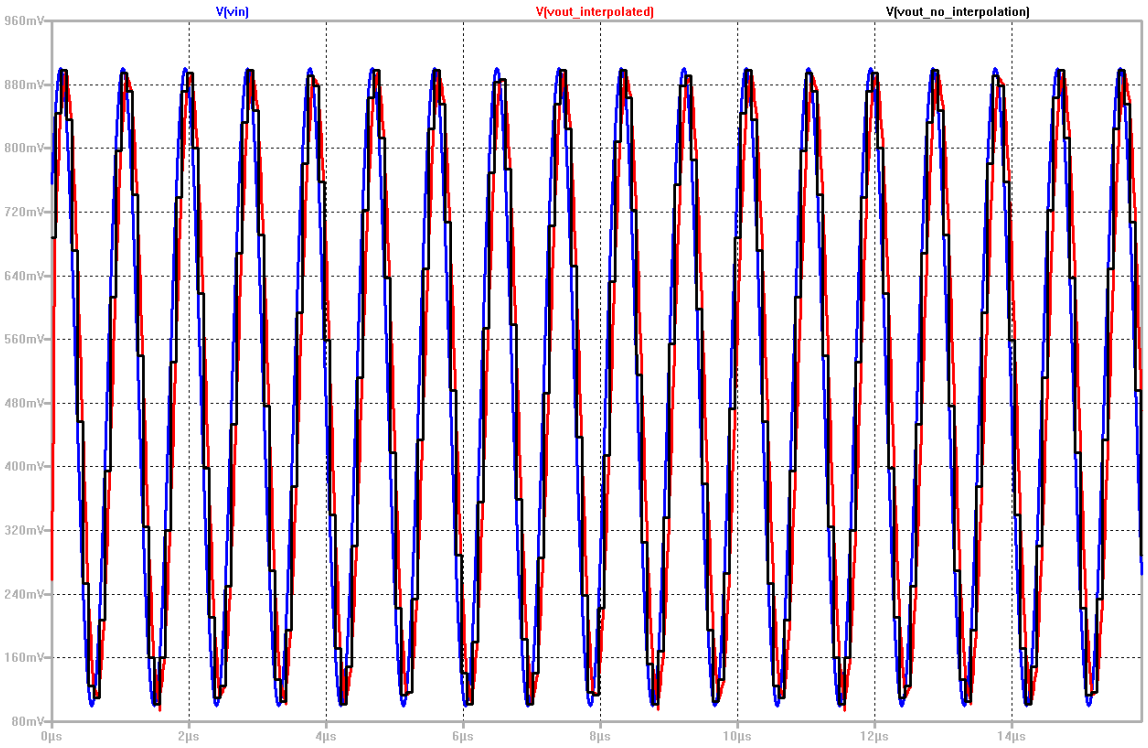


Figure 4. Time domain plot for Ex. 4.8

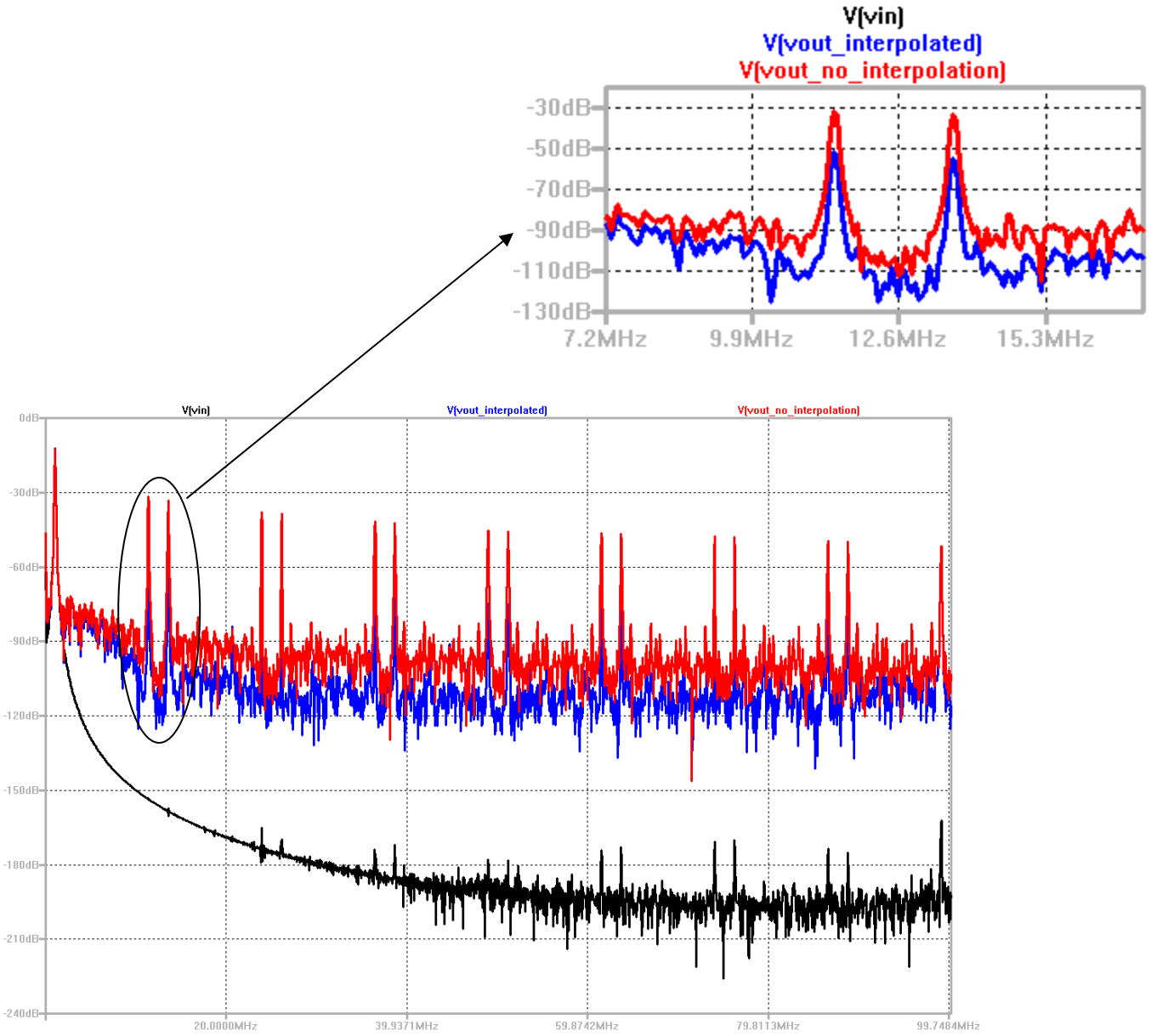


Figure 5. Frequency domain (FFT) plot for Ex. 4.8

In Figure 5, it is first noted that the signal's spectrum for vout_no_interpolation which is the S/H stage's output, is centered around n*12.5MHz, and the amplitude is attenuated as frequency goes up due to the sinc-response of S/H stage. Comparing vout_interpolated's spectrum with that of vout_no_interpolation's, the image spectrum is further reduced by the moving-average image-removal filter whose frequency response is seen in Figure 3.

4.15) ***In Fig. 4.32, which blocks serve as the AAF and which serve as the S/H?***

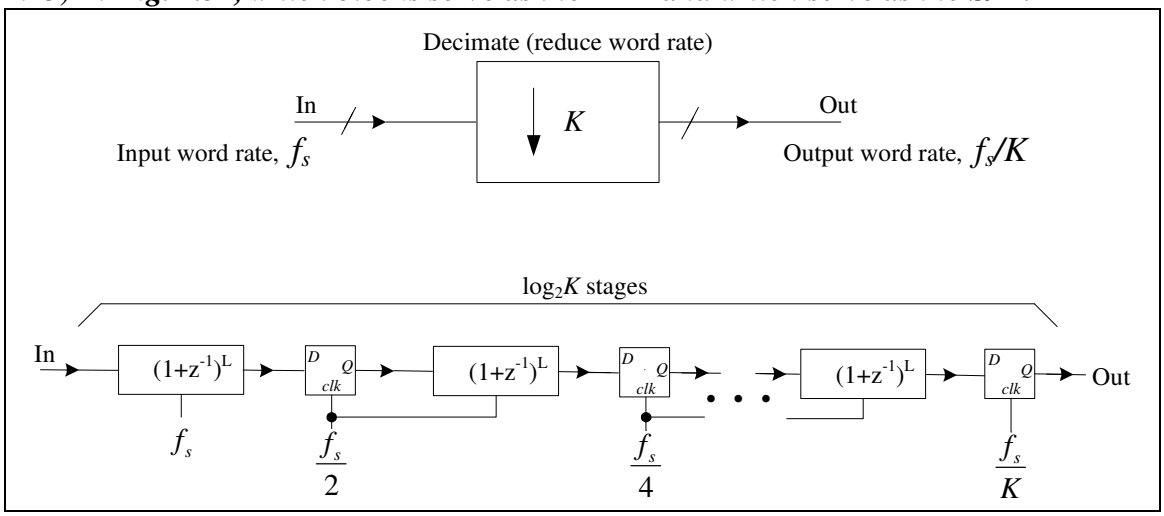


Figure 4.32: Decimation using Sinc anti-aliasing filters

Solution:

In figure 4.32, we are given a circuit that performs decimation for discrete time input signals. Decimating a digital signal is also known as down-sampling of the input signal, the sampling rate here in fig.4.32 goes from f_s to f_s/K. By performing decimation, we are effectively discarding some information, in other words we are selecting only the information we need and discarding the rest. This technique of selecting the required information and discarding the rest is done by an antialiasing digital low-pass filter which selects the only information that we want and attenuates the rest, after which the signal is resampled at a lower decimated rate by a sample and hold circuit.

From the figure above, since we are decimating K times, the effective sampling frequency of the output signal is f_s/K and the spectrum of the signal that carries the information we need should be bandwidth limited from DC to $f_s/2K$ or the new Nyquist frequency. Because whatever information that falls above this bandwidth is attenuated/discarded by the AAF.

Let's look at the first stage of the decimating sinc filter shown above. For now, say L=1,

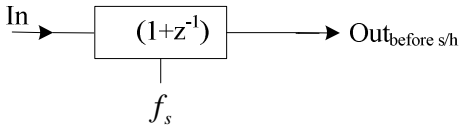


Figure 1: First stage averager

The transfer function of the averager is $(1+z^{-1})$, which can also be written as,

$$\frac{(1-z^{-2})}{(1-z^{-1})} \tag{1}$$

This function is not exactly a sinc response; it acts as a comb filter for stage 1. This response starts to behave more like a sinc response when further stages are added, which is shown below in figure 3 and 4. Since it is clocked at f_s the information below the Nyquist frequency of $f_s/2$ is passed on to the output reliably. While the sample and hold is done by the simple latch that follows the averager, this resamples the data at a lower

decimated frequency of $f_s/2$ and therefore the reliable information of this output spectrum is bandlimited from DC to $f_s/4$. The magnitude response of the AAF filter discussed above is:

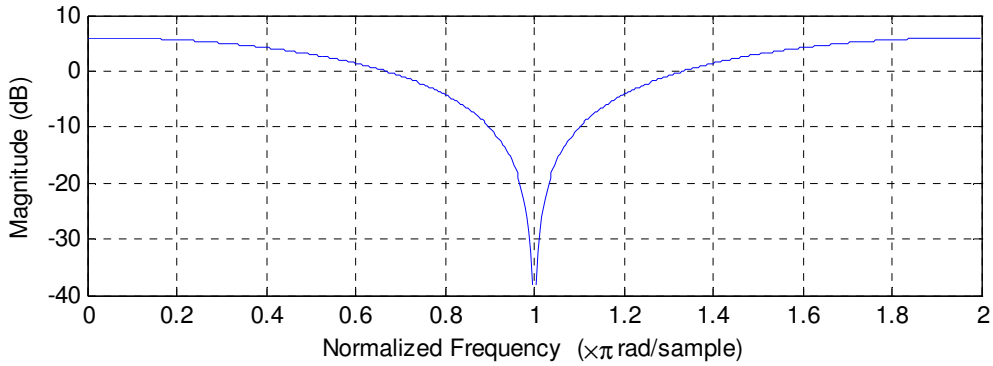


Figure 2: Magnitude response of the averager of stage 1 acts like a comb filter

Similarly for stage 2, the output after the averager of the second stage is given by,

$$(1+z^{-1})\cdot(1+z^{-2}) = (1+z^{-1}+z^{-2}+z^{-3}) = \frac{(1-z^{-4})}{(1-z^{-1})} \qquad (2)$$

which is a response of a sinc filter. The anti-aliasing filtering for the second stage is done by the averager while the d-flip flop that follows it does the sample and hold. The magnitude response is shown below:

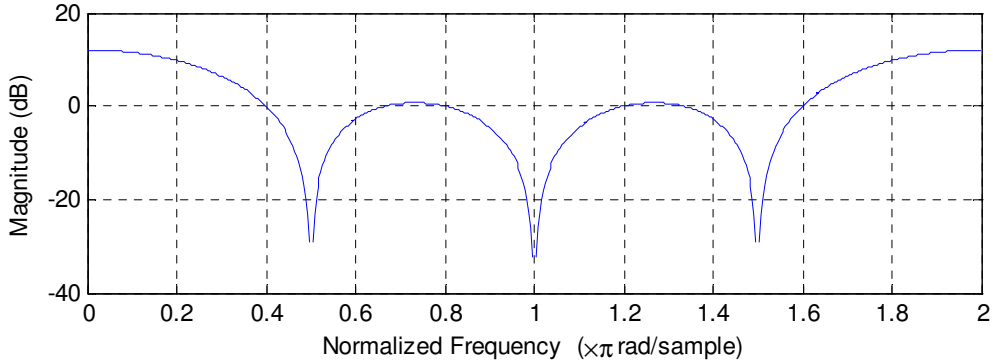


Figure 3: Magnitude response of the averager of stage 2 (Sinc response)

For the third stage, the output of the averager is

$$(1+z^{-1}+z^{-2}+z^{-3})\cdot\left(1+z^{-4}\right) = (1+z^{-1}+z^{-2}+z^{-3}+z^{-4}+z^{-5}+z^{-6}+z^{-7}) = \frac{(1-z^{-8})}{(1-z^{-1})} \qquad (3)$$

And the magnitude response is shown below

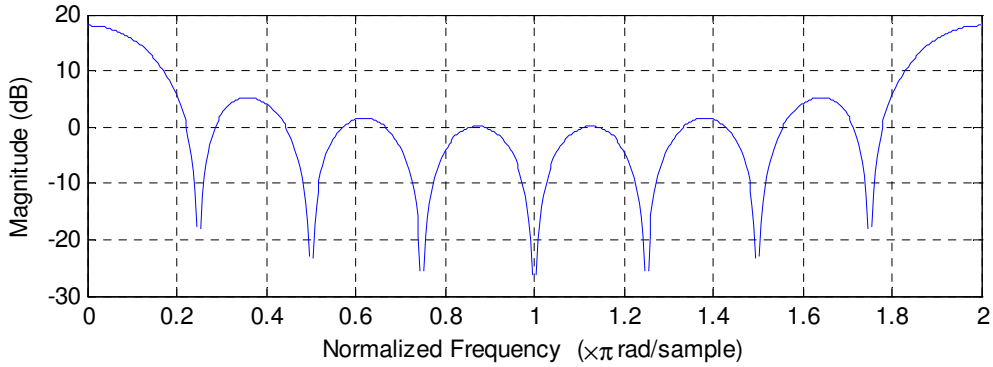


Figure 4: Magnitude response of the averager of stage 3 (Sinc response)

Summarizing the observations, the input signal is passed into the AAF or the digital averager. The averager frequency response is a sinc-response (except for the first stage), the information below the Nyquist frequency of $f_s/2$ is passed reliably to the S/H stage. The D-flip flop is clocked every $f_s/2$ to resample the signal and hold it and therefore the wanted spectrum is now bandlimited to $f_s/4$. Similarly, the second stage averager is clocked at $f_s/2$ and the S/H is clocked at $f_s/4$ so that the information below the new second stage Nyquist frequency of $f_s/8$ is passed reliably and so on. Hence, after K stages, the reliable information that is passed to the output is below $f_s/2K$, the wanted information that we need should be contained within this band-limit for proper functioning of the decimation technique.

Therefore, we can see that we are performing a anti-aliasing filtering in each stage of figure 4.32 in form of an averager and the sample-and-hold using a latch.

Solution by Jake Baker

4.16 For the FIR filter seen in Fig. 4.35 with all coefficients set to 0.25, sketch the filter's frequency response.

If all of the coefficients are set to 0.25 we can write, using Eq. (4.35),

$$H(z) = 0.25 \cdot (1 + z^{-1} + z^{-2} + z^{-3})$$

or

$$H(z) = 0.25 \cdot (1 + z^{-1} + z^{-2} + z^{-3}) \cdot \frac{1 - z^{-1}}{1 - z^{-1}} = 0.25 \cdot \frac{1 - z^{-4}}{1 - z^{-1}}$$

which is simply a lowpass Sinc filter (a moving average filter) with the frequency response discussed in Sec. 4.2.2,

$$|H(f)| = \left| \frac{Sinc\left(\pi \frac{4 \cdot f}{f_s}\right)}{Sinc\left(\pi \frac{f}{f_s}\right)} \right|$$

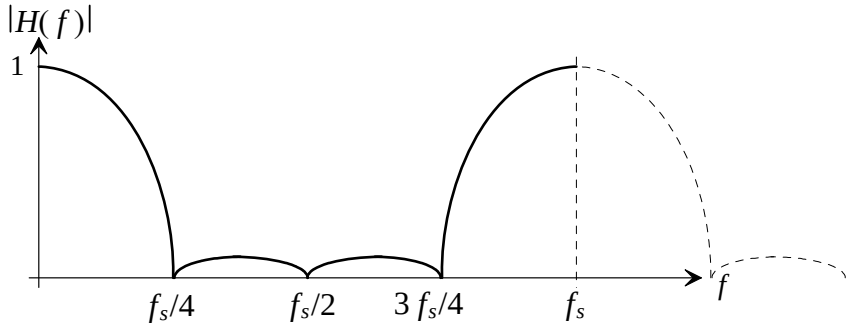
and

$$\angle H(f) = \overbrace{\frac{\pi}{2} - K\pi \cdot \frac{f}{f_s}}^{\text{Eq. (1.54), comb filter}} + \overbrace{2\pi \frac{f}{f_s} - \left(\pi \frac{f}{f_s} + \frac{\pi}{2}\right)}^{\text{Eq. (1.63), non-delaying integrator}} \quad \text{for } 0 < f < f_s/K$$

For $K = 4$ the phase response simplifies to

$$\angle H(f) = -3\pi \cdot \frac{f}{f_s} \text{ for } 0 < f < f_s/4$$

A sketch of the filter's frequency response is seen below.



4.17. For the filter seen in Fig. 4.57 determine the range of values for *a* and *b* where the filter will be stable. What is the filter's transfer function? Sketch the location of the filter's poles and zeroes.

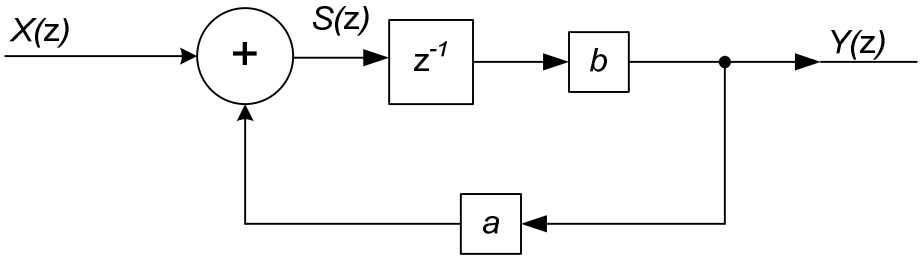


Figure 4.57 A weighted integrating filter

Sol: To find the stability of a digital filter, we need to find the location of the poles. If all the poles of the filter lie inside the unit circle then we can say that the filter is stable.

To solve this problem we derive the transfer function of the filter. In the next section we discuss the range of values of **a** and **b** for which the filter is stable. Locate the poles and zeroes of the filter in z-plane. In the next section we discuss why having a pole outside the unit circle introduces instability in a filter with given filter as example.

Section 1: Derivation of the transfer function of the filter

From Figure 4.57 we can see that **"S(z)"** is delayed by a unit sample interval, and is multiplied by a scalar **"b"** to derive the output "**Y(z)"**.

$$Y(z) = b \bullet z^{-1} \bullet S(z) \tag{1}$$

$$S(z) = X(z) + \left(a \bullet Y(z)\right) \tag{2}$$

$$Y(z) = b \bullet z^{-1} \bullet \left(X(z) + \left(a \bullet Y(z)\right)\right) \tag{3}$$

$$Y(z) = b \bullet z^{-1} \bullet X(z) + a \bullet b \bullet z^{-1} \bullet Y(z) \tag{4}$$

$$Y(z) - a \bullet b \bullet z^{-1} \bullet Y(z) = b \bullet z^{-1} \bullet X(z) \tag{5}$$

$$Y(z)\left(1 - a \bullet b \bullet z^{-1}\right) = b \bullet z^{-1} \bullet X(z) \tag{6}$$

$$Y(z) = \frac{b \bullet z^{-1} \bullet X(z)}{\left(1 - a \bullet b \bullet z^{-1}\right)} \tag{7}$$

$$\frac{Y(z)}{X(z)} = \frac{b \bullet z^{-1}}{\left(1 - a \bullet b \bullet z^{-1}\right)} \tag{8}$$

$$H(z) = \frac{b \bullet z^{-1}}{\left(1 - a \bullet b \bullet z^{-1}\right)} \tag{9}$$

$$H(z) = \frac{b/z}{(1 - a \cdot b/z)} \tag{10}$$

The transfer function of the filter is given by

$$H(z) = \frac{b}{(z - a \cdot b)} \tag{11}$$

Section 2: Range of values of a and b for the filter to be stable

Inspecting the transfer function obtained using eq. (11) we can say that the pole is located at **a.b** and there are no zeroes for this filter. Since for the filter to be stable the pole should lie inside the unit circle we have

$$|a.b| < 1 \tag{12}$$

This means if for example **a** lies between -2 and +2 then **b** is given by

$$|a| = 2 \Rightarrow 0 < |b| < 1/2 \Rightarrow -1/2 < b < 1/2 \tag{13}$$

Hence the range of values of a, b is given by

$$\text{for } |a| = x \text{ where x is an integer } 0 < |b| < (1/x) \Rightarrow (-1/x) < b < (1/x) \tag{14}$$

Section 3: Location of the Poles and zeroes of the filter

Let us consider four cases.

Case 1: **a.b > 1**, pole lies outside the unit circle, on the right half of the z-plane filter is unstable example values are a=2, b=1. Pole is at z=a.b i.e at z=2.

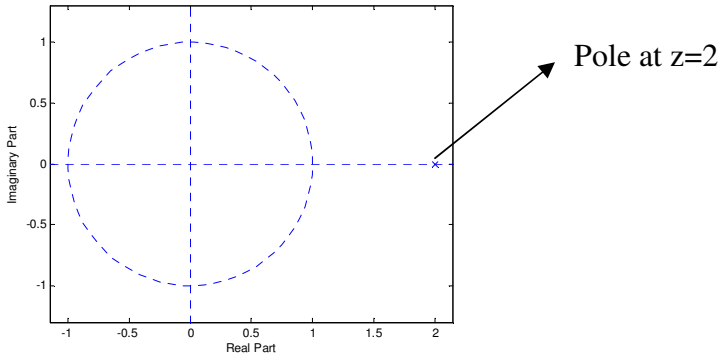


Fig. 1. Pole-zero location when a.b=2

```
%MATLAB CODE to plot poles and zeros of the filter
%a=2,b=1,a.b=2
%H(z)= b/(z-ab)
%coefficients of numerator in transfer function H(z)
y=[0 1]
%coefficients of denominator in transfer function H(z)
x=[1 -2]
```

```
%plot poles and zeroes for H(z)
zplane(y,x)
figure
```

Case 2: **a.b < -1**, pole lies outside the unit circle, on the left half of the z-pane, filter is unstable, example values of a,b are a=-2, b=1. Pole is at z=a.b i.e at z=-2.

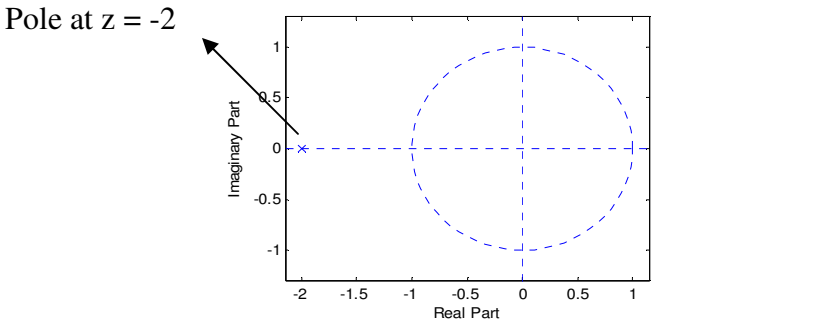


Fig. 2. Pole-zero location when a.b=2

```
%MATLAB CODE to plot poles and zeros of the filter
%a=-2,b=1 a.b=-2
%H(z)= b/(z-ab)
%coefficients of numerator in transfer function H(z)
y=[0 1]
%coefficients of denominator in transfer function H(z)
x=[1 2]
%plot poles and zeroes for H(z)
zplane(y,x)
figure
```

Case 3: **0<a.b <1**, pole lies inside the unit circle and on right half of z-plane, filter is stable, example values of a,b are a=2, b=0.25. Pole is at z=a.b i.e. at z=0.5.

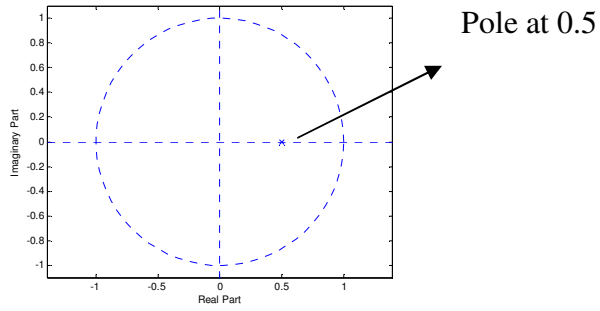


Fig. 3. Pole-zero location when a.b=0.5

```
%MATLAB CODE to plot poles and zeros of the filter
%a=2,b=0.25 a.b=0.5
%H(z)= b/(z-ab)
%coefficients of numerator in transfer function H(z)
```

```
y=[0 1]
%coefficients of denominator in transfer function H(z)
x=[1 -0.5]
%plot poles and zeroes for H(z)
zplane(y,x)
figure
```

Case 4: **-1<a.b < 0**, pole lies inside the unit circle and on left half of z-plane, filter is stable, example values of a,b are a=2, b=-0.25. Pole is at z=a.b i.e. z=-0.5.

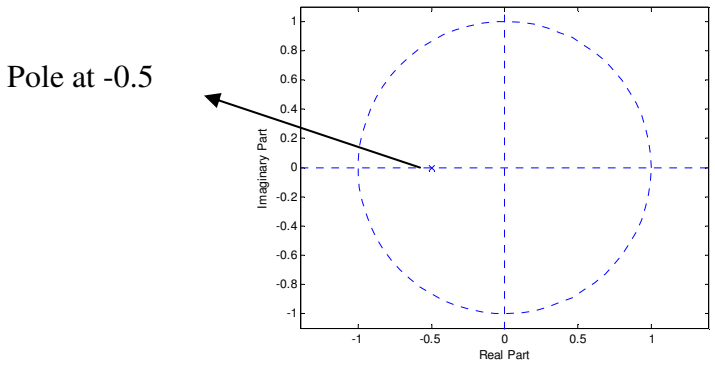


Fig. 4. Pole-zero location when a.b = -0.5

```
%MATLAB CODE to plot poles and zeros of the filter
%a=2,b=-0.25 a.b=-0.5
%H(z)= b/(z-ab)
%coefficients of numerator in transfer function H(z)
y=[0 1]
%coefficients of denominator in transfer function H(z)
x=[1 0.5]
%plot poles and zeroes for H(z)
zplane(y,x)
```

Section 4: Illustrating why the filter becomes unstable when poles lie outside the unit circle

In this section we illustrate in time domain how the pole outside the unit circle causes the filter to become unstable.

The time domain equation for the filter can be derived by considering equation (4) i.e.

$$Y(z) = b \bullet z^{-1} \bullet X(z) + a \bullet b \bullet z^{-1} \bullet Y(z)$$

Reducing this equation in to discrete time domain we have

$$y[nT_s] = b \bullet x[(n-1)T_s] + a \bullet b \bullet y[(n-1)T_s] \qquad (15)$$

For a=2, b=1 pole lies outside the unit circle, and the filter is unstable. Let us prove this by analyzing the impulse response of the filter, and using MATLAB. For a=2, b=1 the equation (15) reduces to

$$y[nT_s] = x[(n-1)T_s] + 2 \bullet y[(n-1)T_s] \tag{16}$$

Consider a unit impulse input at zero time n=0 i.e.

$$x[nT_s] = \delta[n] \Rightarrow x[nT_s] = 1 \text{ for } n=0 \text{ , } x[nT_s] = 0 \text{ for } n \neq 0 \tag{17}$$

$$y[0] = x[-T_s] + 2 \bullet y[-T_s] \tag{18}$$

$$y[T_s] = x[0] + 2 \bullet y[0] \Rightarrow y[T_s] = 1 \tag{19}$$

$$y[2T_s] = x[T_s] + 2 \bullet y[T_s] \Rightarrow y[2T_s] = 2 \tag{20}$$

$$y[3T_s] = x[2T_s] + 2 \bullet y[2T_s] \Rightarrow y[3T_s] = 4 \tag{21}$$

Similarly by inspection we can say that the n^{th} output sample is given by

$$y[nT_s] = 2^{n-1} \tag{22}$$

This is an example of **Infinite Impulse Response (IIR)** filter. The impulse response increases exponentially and is of infinite duration as n increases. A filter is said to be stable if its impulse response decays to zero as n goes to infinity. Hence the filter is unstable if **a.b>1**. Let us prove this using MATLAB.

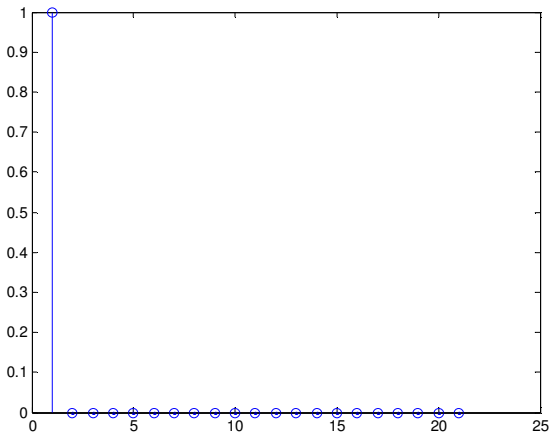


Fig. 5. The unit impulse input to the filter i.e. $x[nT_s]$

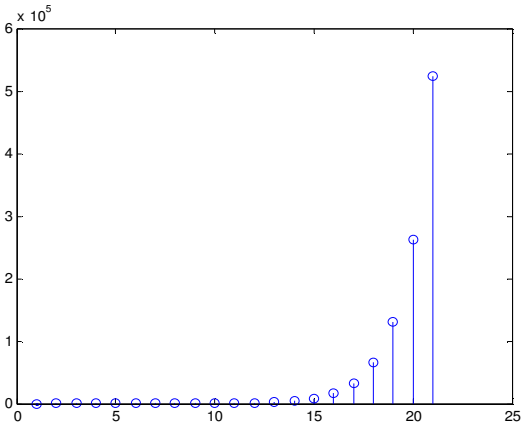


Fig. 6. The impulse response of the filter i.e. $y[nT_s]$
Note: smaller magnitudes at output appear to be zero (but they are not) because of the scale on Y-axis is big.

```
%MATLAB CODE
%unit impulse input
imp = [1; zeros(20,1)];
%plot the input
stem(imp)
figure
%plot the filters impulse response
%coefficients of numerator
b=[0 1]
%coefficients of denominator
a=[1 -2]
%impulse response of the filter
h=filter(b,a,imp)
%plot the impulse response of the filter
stem(h)
```

We see from Fig. 6 that the impulse response grows exponentially. The output at the 21st sample is around $5.5*10^5$, even though the input is a single unit impulse at time zero. Now we can say that as a finite duration input with finite amplitude causes the output to grow exponentially, the filter is unstable when $|ab| > 1$.

Summary:

1. Derived the transfer function from the block diagram of the given filter.
2. Found the range of poles and zeroes for which the filter becomes stable.
3. Sketched the plots of poles and zeroes for different values of a and b.
4. Explained why the filter becomes unstable when poles of the filter lie outside the unit circle.

4.18 Repeat Ex. 4.9 for a filter with a transfer function of

$$\frac{v_{\text{out}}}{v_{\text{in}}}=\frac{1+j\cdot\dfrac{f}{4\,MHz}}{1+j\cdot\dfrac{f}{800\,kHz}} \tag{1}$$

Solution:

This solution shows how to sketch and determine the transfer function for the digital equivalent of Eq. 1 in three steps.

Step 1: Find the coefficients for the digital filter

First, let's compare Eq. 1 with the bilinear transfer function

$$\frac{v_{\text{out}}(f)}{v_{\text{in}}(f)}=\frac{1}{G_2}\cdot\frac{1+j\cdot\dfrac{f}{f_s/(2\pi G_{3\text{D}})}}{1+j\cdot\dfrac{f}{f_s G_{1\text{D}} G_2/2\pi}} \tag{2}$$

where $G_{1\text{D}}$, $G_{2,}$ and $G_{3\text{D}}$ are the coefficients for digital implementation showing in Fig. 1.

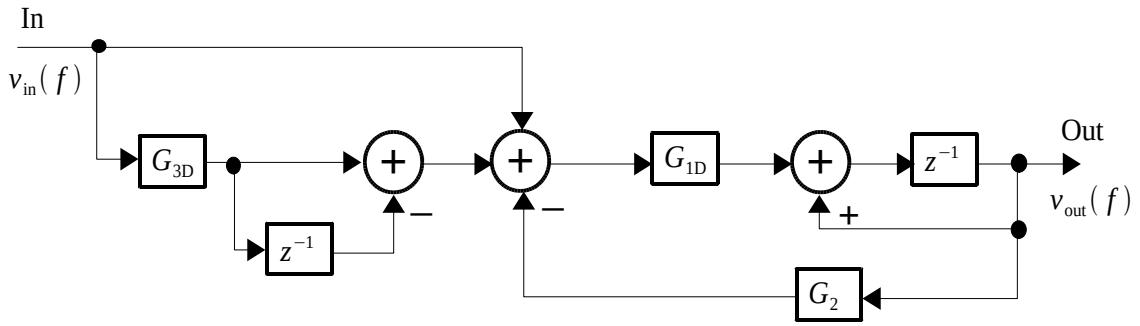


Figure 1 Digital implementation o f the bilinear transfer function.

Setting $G_2=1$, we can rewrite Eq. 2 as

$$\frac{v_{out}(f)}{v_{in}(f)}=\frac{1+j\cdot\dfrac{f}{f_s/(2\pi G_{3\text{D}})}}{1+j\cdot\dfrac{f}{f_s G_{1\text{D}}/2\pi}} \tag{3}$$

Comparing Eq. 1 with Eq. 3, we now can write down

$$4\,MHz=\frac{f_s}{2\pi G_{3\text{D}}} \tag{4}$$

$$800\,kHz=\frac{f_s G_{1\text{D}}}{2\pi} \tag{5}$$

Assuming the digital filter is clocked at 100 MHz, its coefficients are then

$$G_{1D} \approx 0.05 \quad and \quad G_{3D} \approx 4 \tag{6}$$

Step 2: Choose a filter topology

The digital filter can be implemented using any of the filter topology shown in Fig. 4.43 in [1]. Here we use the implementation shown in Fig. 2.

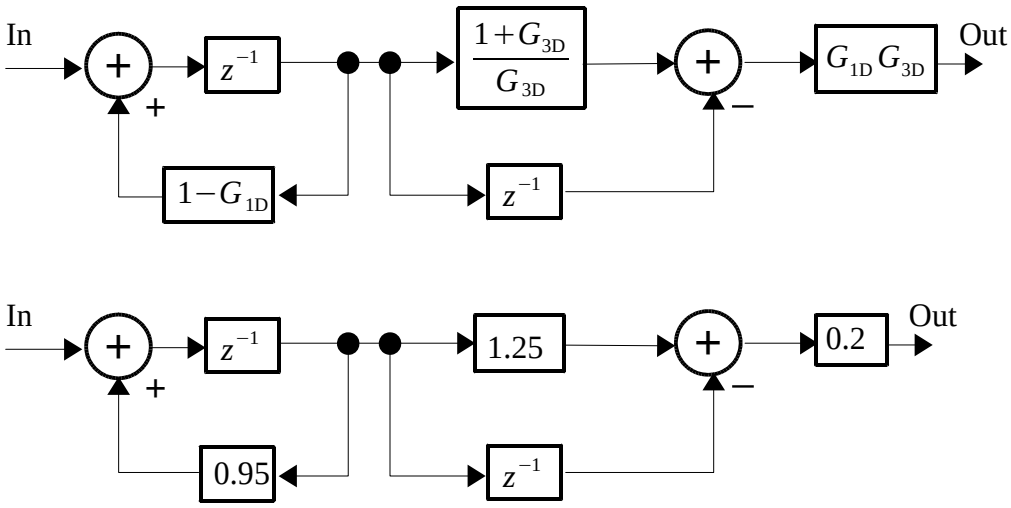


Figure 2 Digital filter implementation for Eq. 1

Step 3: Determine the z-domain transfer function

To avoid making mistake, we can label the output at each nodes as shown in Fig. 3.

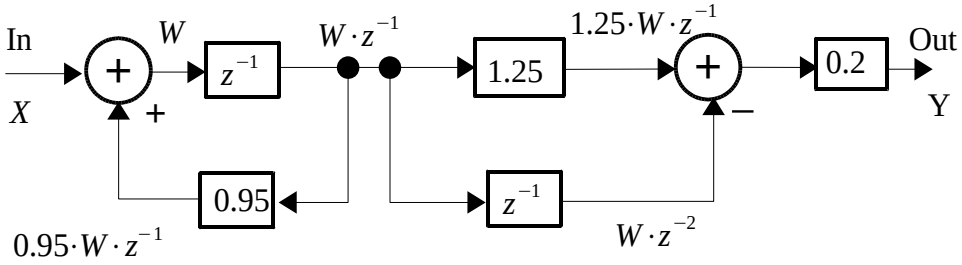


Figure 3 The digital filter for Eq. 1

From Fig. 3, in z-domain, we have

$$X + 0.95 \cdot W \cdot z^{-1} = W \tag{7}$$

$$0.2 \cdot (1.25 \cdot W\, z^{-1} - W\, z^{-2}) = Y \tag{8}$$

Rewriting Eqs. 7 and 8, we get

$$X = W \cdot (1 - 0.95\, z^{-1}) \tag{9}$$

$$Y = 0.25 \cdot W \cdot z^{-1} \cdot (1 - 0.8\, z^{-1}) \tag{10}$$

The z-domain transfer function is then

$$H(z) = \frac{Y}{X} = 0.25 \cdot \frac{z^{-1}}{1 - 0.95\, z^{-1}} \cdot (1 - 0.8\, z^{-1}) \tag{11}$$

Reference:

[1] R. J. Baker, *CMOS Mixed-signal Circuit Design, Second Edition*, Wiley-IEEE, 2009.

4.19) Repeat Question 4.18 using the canonic form of the first-order digital filter.

4.18 Repeat Ex. 4.9 for a filter with a transfer function of

$$\frac{v_{out}}{v_{in}} = \frac{1 + j\frac{f}{4MHz}}{1 + j\frac{f}{800kHz}}. \qquad (4.19.1)$$

Example 4.9) Sketch and determine the transfer function for the digital filter equivalent of the following RC circuit. Assume the digital filter is clocked at 100MHz.

Solution:
The RC circuit referred to in Ex. 4.9 does not apply to this problem. Instead, the time domain transfer function from question 4.18 replaces it. The author of this solution may not have a solid enough foundation on the use of the variables G and G_D to which this problem pertains due to the fact that chapter 3 material was not covered at the time of preparing this solution. The reader is encouraged to refer to chapter 3 or other sources for more information on this.

Following the work shown in Ex. 4.10 we begin –

Both the numerator and denominator in the transfer function of (4.19.1) are in the form $1 + j(f/f_{3dB})$. This means that $f_{3dB,zero}$ = 4MHz and $f_{3dB,pole}$ = 800kHz. Also, as stated in the problem statement of Ex. 4.9, the sampling frequency is 100MHz. Comparing (4.50) from the book to (4.19.1) and using (4.51) from the book, along with the values just defined, we can find a value for A_1:
$f_{3dB,pole} = \frac{f_s(1-A_1)}{2\pi} \Rightarrow A_1 = 1 - \frac{2\pi \cdot f_{3dB,pole}}{f_s} \Rightarrow A_1 = 1 - \frac{2\pi \cdot 800k}{100M} \Rightarrow A_1 = 0.95.$ The DC gain from equation (4.50) is $\frac{B_0 + B_1}{1 - A_1}$. From the problem statement we see that the DC gain is 1. $1 = \frac{B_0 + B_1}{1 - A_1} \Rightarrow \frac{1 - A_1}{B_0} - 1 = \frac{B_1}{B_0} = \frac{0.05}{B_0} - 1.$ From (4.52) from the book we find: $f_{3dB,zero} = \frac{f_s}{2\pi}\left(1 + \frac{B_1}{B_0}\right)$. Substituting in $\frac{0.05}{B_0} - 1$ for $\frac{B_1}{B_0}$ we find
$f_{3dB,zero} = \frac{f_s}{2\pi}\left(1 + \frac{0.05}{B_0} - 1\right) \Rightarrow B_0 = \frac{f_s \cdot 0.05}{2\pi \cdot f_{3dB,zero}} \Rightarrow B_0 = \frac{100M \cdot 0.05}{2\pi \cdot 4M} = 0.2.$ Finally, we can find B_1: $\frac{B_1}{B_0} = \frac{0.05}{B_0} - 1 \Rightarrow B_1 = 0.05 - B_0 = 0.05 - 0.2 = -0.15.$

To recap, A_1 = 0.95, B_0 = 0.2, and B_1 = -0.15. The sketch for this circuit can be seen in Fig 4.19.1.

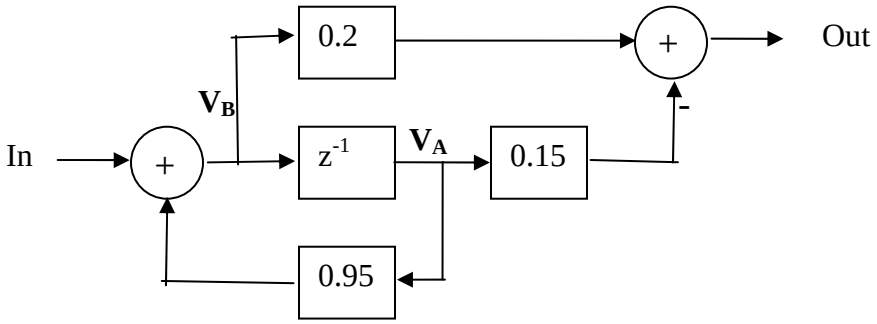


Figure 4.19.1. Canonic form of the digital version of the transfer function given in question 4.18.

The question asks for the transfer function. To find that, we can find the values at points A and B in Fig 4.19.1 and then sum them. I have included two different approaches for finding the solution. They differ in their assessment of V_B. The second approach is, perhaps, a little faster and easier in this case.

$$V_A = \left(0.95V_A + V_{In}\right) z^{-1}$$

$$V_A = V_{In}\left[\frac{z^{-1}}{1-0.95z^{-1}}\right]$$

$$V_B = V_{In} + 0.95V_A$$

$$V_{Out} = 0.2V_B - 0.15V_A$$

$$V_{Out} = 0.2\left[V_{In} + 0.95V_{In}\left(\frac{z^{-1}}{1-0.95z^{-1}}\right)\right] - 0.15V_{In}\left(\frac{z^{-1}}{1-0.95z^{-1}}\right)$$

$$\frac{V_{Out}}{V_{In}} = 0.2 + \frac{0.04z^{-1}}{1-0.95z^{-1}}$$

$$\frac{V_{Out}}{V_{In}} = \frac{0.2-0.15z^{-1}}{1-0.95z^{-1}} = 0.2\left(\frac{1-0.75z^{-1}}{1-0.95z^{-1}}\right).$$

$$V_A = \left(0.95V_A + V_{In}\right) z^{-1}$$

$$V_A = V_{In}\left[\frac{z^{-1}}{1-0.95z^{-1}}\right]$$

$$V_B = \frac{V_A}{z^{-1}} = V_{In}\left[\frac{1}{1-0.95z^{-1}}\right]$$

$$V_{Out} = 0.2V_B - 0.15V_A$$

$$V_{Out} = 0.2V_{In}\left[\frac{1}{1-0.95z^{-1}}\right] - 0.15V_{In}\left(\frac{z^{-1}}{1-0.95z^{-1}}\right)$$

$$\frac{V_{Out}}{V_{In}} = \frac{0.2-0.15z^{-1}}{1-0.95z^{-1}}$$

$$\frac{V_{Out}}{V_{In}} = 0.2\left(\frac{1-0.75z^{-1}}{1-0.95z^{-1}}\right).$$

Jason Durand

Problem 4.20 – Repeat Example 4.12 if the Q is increased to 1. Example 4.12 is a repeat of example 3.8, a lowpass filter, only for digital signals. Design with a sampling frequency of 100MHz and cutoff frequency (f_0) of 1.59MHz.

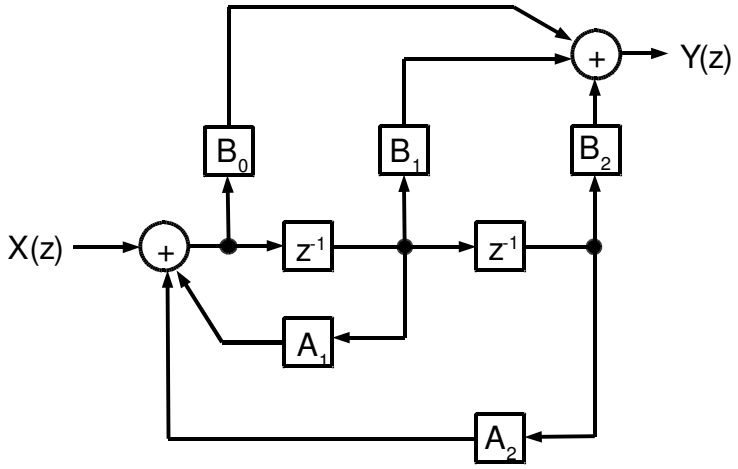


Figure 1: General digital Bi-quadratic filter

The general form of the digital bi-quad filter looks like Fig 1, with a transfer function of

$$H(z) = \frac{Y(z)}{X(z)} = \frac{B_0 z^2 + B_1 z + B_2}{z^2 - A_1 z - A_2} \quad ,$$

which turns to approximately

$$H(s) = \frac{B_0 s^2 + f_s(2B_0 + B_1)s + f_s^2(B_0 + B_1 + B_2)}{s^2 + f_s(2 - A_1)s + f_2^2(1 - A_1 - A_2)} \quad ,$$

using the simplification that $z = 1 + \frac{s}{f_s}$ for frequencies much smaller than the sampling frequency f_s. This simplification is useful, since it now looks similar to an analog LRC filter of the form

$$\frac{V_{out}}{V_i} = \frac{\frac{1}{LC}}{s^2 + s\frac{R}{L} + \frac{1}{LC}} \quad \text{, with} \quad f_0 = \frac{1}{2\pi\sqrt{LC}} \quad \text{and} \quad Q = \frac{1}{R}\sqrt{\frac{L}{C}} \quad .$$

For the digital implementation, this means that $B_0 = f_s(2B_0 + B_1) = 0$, $\frac{2\pi f_0}{Q} = f_s(2 - A_1)$ and $f_0 = \frac{f_s}{2\pi}\sqrt{1 - A_1 - A_2}$, which are used to design the filter.

For a Q of 1,

$$A_1 = 2 - \frac{2\pi f_0}{Q f_s} \approx 1.900 \quad \text{. Furthermore,} \quad A_2 = 1 - A_1 - \left(\frac{2\pi f_0}{f_s}\right)^2 \approx -0.910 \quad .$$

To normalize low frequency gain to 1, $B_2 = 1 - A_1 - A_2 = 0.01$.

The calculated values of A_1, A_2, and B_2 then plug directly into the digital filter block diagram as seen in figure 1. Since the gain values of 1.9, -0.91 and 0.01 cannot be implemented with bit shifts, clever adding can be used to come close to the gain values, such seen in figure 2.

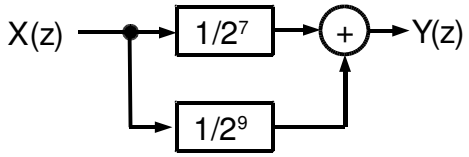


Figure 2: Simple implementation of gain = 0.01 (approximately)

Transfer function of designed filter:

$$H(s) = \frac{0.01 f_s^2}{s^2 + 0.1 f_s s + 0.01 f_s^2} \approx H(z) = \frac{0.01 z^{-2}}{1 - 1.9 z^{-1} + 0.91 z^{-2}}$$, which has frequency response as shown in figure 3. Note the peaking that occurs from Q = 1.

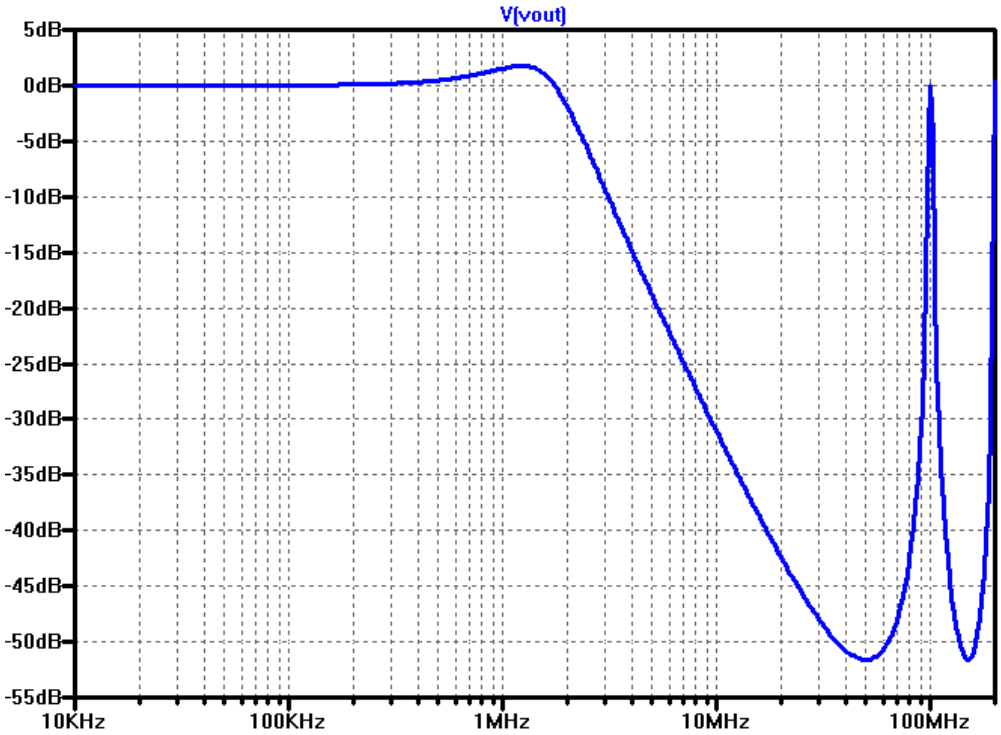


Figure 3: Frequency response of designed filter

Shantanu Gupta

Question 4.21

Show that the filter shown in Fig. 4.58 can be implemented using a single multiplier.

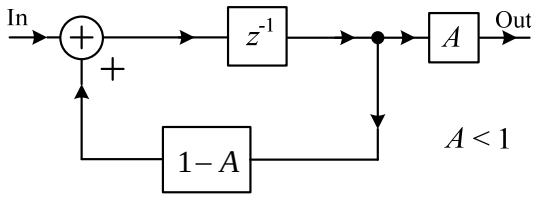


Figure 4.58 Filter used for question above (page 161)

Solution:-

Looking at the block diagram of filter above the signal at the point before multiplier block A can be thought of as $\frac{y[nT_s]}{A}$ if the output is $y[nT_s]$. Thus looking at the block diagram below the transfer function is given as:

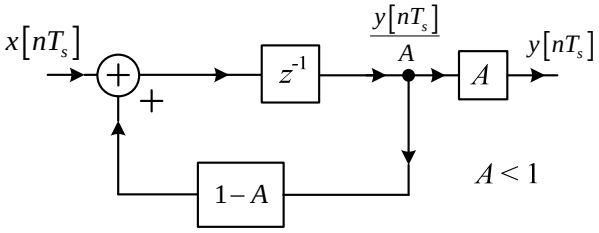


Figure 1 Filter block diagram with discrete signal

Transfer function

$$\frac{y[nT_s]}{A} = x\left[(n-1)T_s\right] + \frac{y\left[(n-1)T_s\right]}{A}(1-A)$$

Equation can be written in z-domain as; where z^{-1} is unit delay element

$$\frac{Y(z)}{A} = X(z)z^{-1} + \frac{Y(z)z^{-1}}{A}(1-A)$$

after rearranging the equation we get

$$Y(z) = z^{-1}A\left(X(z) - Y(z)\right) + Y(z)z^{-1}$$

This can equation can be represented with a single multiplier A (as per question) in block diagram shown below

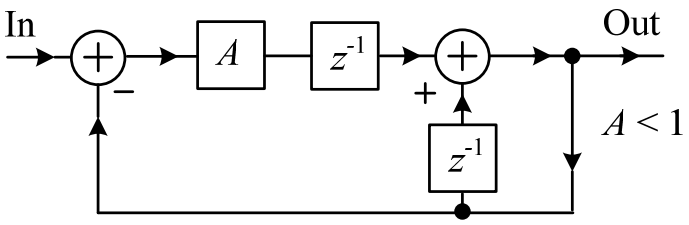


Figure 2 Filter block diagram with single multiplier

Show that if the values of A and B are restricted to 1, 0.5, 0.25, 0.125, etc. that the circuit of Fig. 4.59 can be used to implement multiplication by coefficients that aren't directly powers of two. How would a multiply by 0.75 be implemented? a multiply-by-0.9375? a multiply-by-0.5625?

We begin the solution by re-printing Fig. 4.59 from the book as solution Figure 4.22.1.

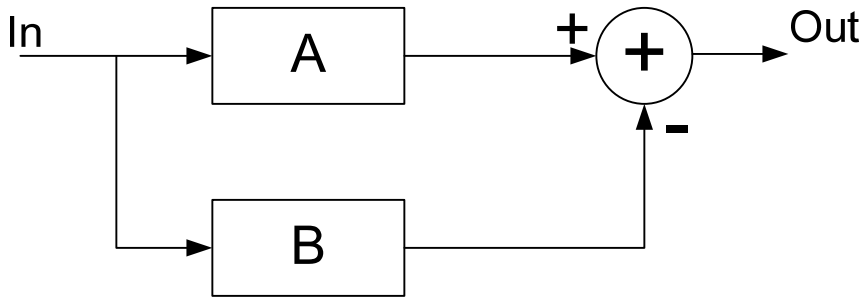


Figure 4.22.1: A simple multiplier where A and B simply shift the data.

This configuration (using dividers by shifting the data) can be used to implement multiplication by a number less than 1. Let's proceed with the discussion using examples… If we would like to implement multiplication by 0.75, we can restate our purpose by saying that we want to keep only three quarters of the signal, subtracting 25%. We can get 25% of the signal by dividing the signal by 4 (the value of B) as seen in Figure 4.22.2.

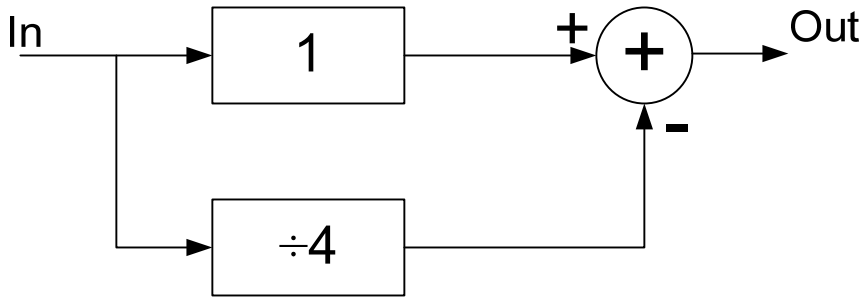


Figure 4.22.2: Simple multiplier implementing multiplication by 0.75.

Let's check the accuracy of Figure 4.22.2 by assuming that our input is 1. The output of the "divide by 4" block will be 0.25, while the output of the "divide by 1" block will simply be 1. The adder then performs the operation 1-0.25 and outputs 0.75, or the effective multiplication of the input by 0.75.

Another example is seen in Figure 4.22.3. This example implements multiplication by 0.9375. Remember, that another way to think about multiplication by 0.9375 is to think that we want to *keep* 93.75% of our signal, and *discard* 6.25%. That enables us to clearly see that we want to *subtract* 6.25% of our signal and return the result.

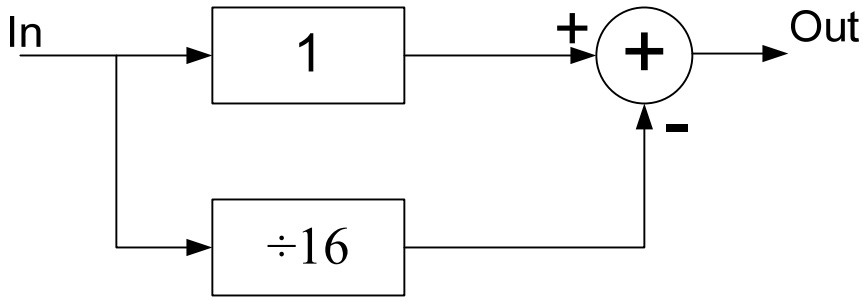


Figure 4.22.3: Simple multiplier implementing multiplication by 0.9375.

Again, let's check the accuracy of Figure 4.22.3 by testing the results with an input of 1. Block A (referring to figure 4.22.1) outputs the unmolested input signal, and block B outputs the input signal divided by 16, or 0.0625 in the case where the input signal is 1. The adder then performs the simple subtraction: 1-0.0625 and outputs the result: 0.9375.

What about the case where we want to subtract out a percentage of our signal that is not a direct multiple of 2? Figure 4.22.4 shows a method of modifying Figure 4.22.1 that allows us to get creative with the multiplication factors.

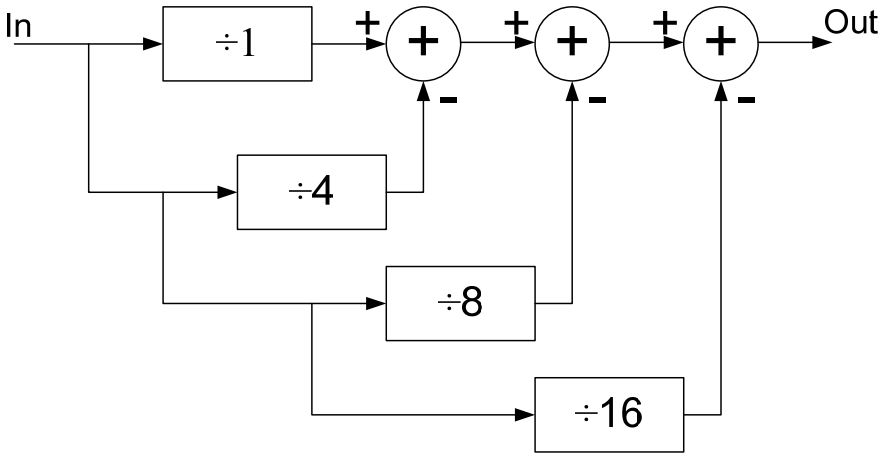


Figure 4.22.4: Simple multiplier implementing multiplication by 0.5625.

First I will claim that the circuit in Figure 4.22.4 implements multiplication by 0.5625. Let's use a test case to see if we get the result we want. Assuming an input signal of 1, we can see that subsequent subtractions of first ¼, then 1/8 and finally 1/16 are made from the pure input signal. Combining the operations of the 3 adders, we will end up with the following operation:

1-1/4-1/8-1/16 = 9/16, or 0.5625

So we see that stage by stage we are discarding a portion of our signal until we are left with the desirable percentage. As a final point, note that block A of Figure 4.22.1 will always by 1 for multiplication implementations greater than 0.5.