[redacted]

DSP Mini-Project: IIR Filter Design

By [redacted]

For [redacted]

Algorithm and System Implementation - [redacted]

Contents

Chapter 1: Design Decisions

The design of the infinite impulse response filter will consist of 4 second-order sections cascaded sequentially. A normalised Denominator-Numerator TDL structure must also be implemented as instructed by the specification. The initial cascaded structure that can be designed from the given specifications is a cascaded 4 stage filter with each stage being a second order D-N (TLD) structure as shown in the diagram below.

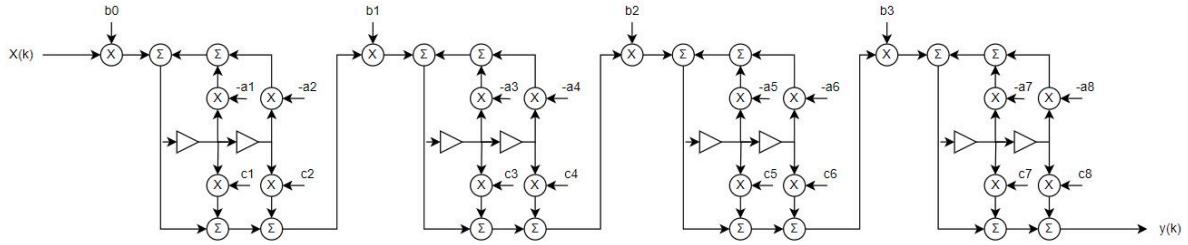

Figure 1; Four cascaded normalised DN-(TDL) sections.

Filter Structure

When cascading DN (TDL) sections it is often the case that the internal signals and the output are driven up to very high levels due to the denominator coming first. Whereas computing the numerator first can be expressed as an attenuation, which would decrease the level of the output. If we can compute the numerator before the denominator, then the possible maximum signal level is sure to be decreased as the signal is attenuated before computing the gain. Therefore it can be said that the N-D (TDL) form is better than the D-N (TDL) form, due to the limiting of signal levels internally and at the output. The design of the infinite impulse response filter could incorporate ND cascades while maintaining a D-N configuration, then the resulting design would still gain from the benefits of decreased signal growth in ND cascades. It is better to treat cascaded DN sections as D-ND-ND-ND-N, this improves the pole-zero pairing and allows us to achieve lower internal signal growth while maintaining an overall function of a D-N (TDL) filter. Therefore a structure of four cascaded normalised N-D(TDL) blocks can be represented instead as shown below figure 2.

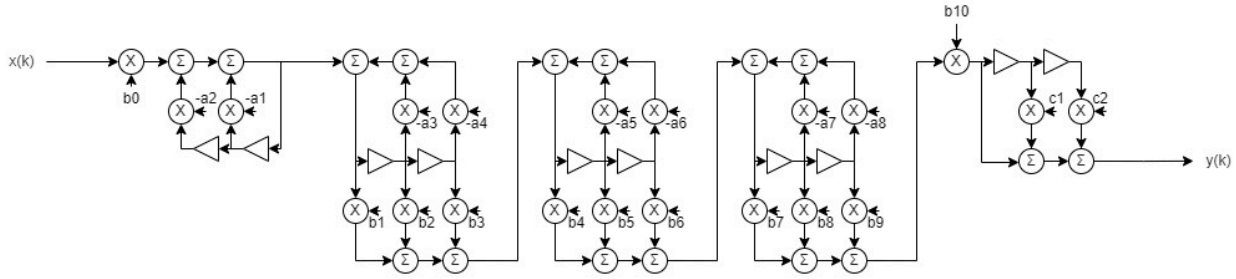

Figure 2: D-ND-ND-ND-N structure

Where the D block at the beginning and the N block at the end are the denominator and numerator sections of a normalized N(TDL)-D(TDL) structure. The intermediate ND blocks are implemented using a reversed order structure that allows the four delayers to be merged. This reversed order structure saves on components and allows the structure to be modeled and simulated more efficiently. There is an increase in the number of adders with this structure

however the monetary and energetic cost of the delayer components are much greater than that of the adders.

Chapter 2: Stage Gain Analysis

When analysing the gain of each stage we must place nodes at the appropriate points in the filter structure model and deduce the transfer function at each node.

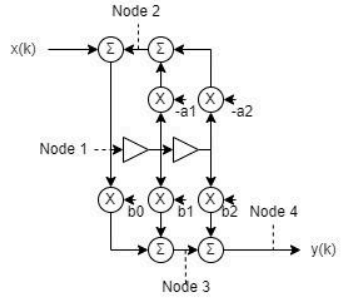

Figure 3; Reversed order normalised ND(TDL) block with nodes labeled.

The signal level at nodes 2 and 3 are not so important if the intermediate results are being clipped, which they will be as in the simulink model each adder block is quantising the output. The transfer functions for the final node of both D and N blocks must be calculated, the structure of the D and N block are shown below in figures 4 and 5 respectively. In order to obtain the transfer function of the output of each structure, the signals can be traced manually. Otherwise the numerator coefficients of the D block are given by the matlab variable previously defined as den1, with the numerator or b given the value of 1 and the coefficients of the N block are given by the variable num1.

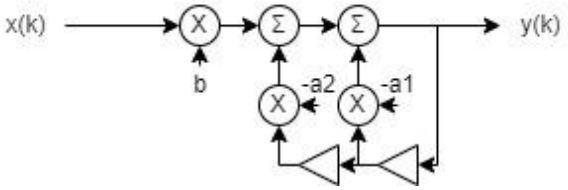

Figure 4; D block structure

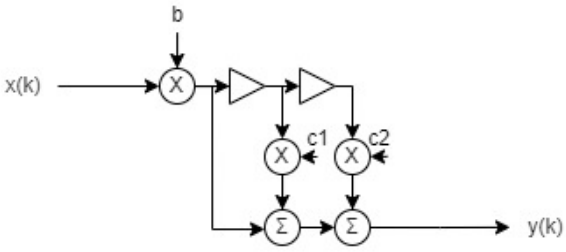

Figure 5; N block structure

The transfer function of the D block will be $H_1(z) = \frac{b}{1+a_1 z^{-1}+a_2 z^{-2}}$ and the transfer function of the N block will be $H_5(z) = 1 + c_1 z^{-1} + c_2 z^{-2}$.

Therefore the numerator and denominator coefficients of each block in the cascade can be stored to variables in MATLAB using the following lines of code:

```
numd = [1   0          0];      dend = [1 -1.2       0.294975];
num2 = [1   0.712824   1];      den2 = [1 -0.413853 0.505521];
num3 = [1   0.05026    1];      den3 = [1 -0.355163 0.82    ];
num4 = [1 -1.995649    1];      den4 = [1 -1.951062 0.9604  ];
numn = [1   0         -1];      denn = [1   0         0       ];
```

The L1 norm for the final node, node 4 of each stage has been calculated using the following MATLAB code:

```
L1_D=sum(abs(impz(numd,dend)))
L1_2=sum(abs(impz(num2,den2)))
L1_3=sum(abs(impz(num3,den3)))
L1_4=sum(abs(impz(num4,den4)))
L1_N=sum(abs(impz(numn,denn)))
```

Where the L1 norm of each stage is calculated by taking the sum of the magnitude of the impulse response of each stage. The results show the worst case maximum signal level of each stage given by the L1 norm of the final node at the output. This also allows me to skip the calculation of the L2 and L infinity norms to save on time. As the absolute worst case signal level, to be conservative, it is a good choice to design the filter according to its worst case scenario as to not cut any corners and provide a fully functional product in all scenarios.
The L1 norms of the output of each stage are as follows:

```
L1_D = 10.5285;
L1_2 = 4.6361;
L1_3 = 4.4380;
L1_4 = 3.0758;
L1_N = 2;
```

It is reassuring to see that the maximum worst-case signal level decreases gradually from stage to stage, this would decrease the maximum signal level that can be reached at the final result.
The stage scale factor can be calculated using the following formula:

$$b_s = \left(max\, L_1 \prod_{n=1}^{s-1} b_n \right)^{-1}$$

Which can be written in MATLAB for each stage as:

```
b1= (L1_D*1)^-1                          Result:
b2= (L1_D*b1)^-1                         b1 = 0.0950
b3= (L1_D*b1*b2)^-1                      b2 = 1
b4= (L1_D*b1*b2*b3)^-1                   b3 = 1
b5= (L1_D*b1*b2*b3*b4)^-1                b4 = 1
                                         b5 = 1
```

The simulink model can be adjusted and simulated with these stage gains preceding each block, the resultant output signal and fft have greatly decreased.

Filter section ordering

The set of coefficients that are to be used in the four section of the filter are given in the specification as the following:

Numerator Coefficients			Denominator Coefficients		
1	0	1	1	-1.2	0.294975
1	0.712824	1	1	-0.413853	0.505521
1	0.505026	1	1	-0.355163	0.82

If we take the coefficients in this order, forming the transfer function for the first stage using the first row of numerator coefficients with the first row of denominator coefficients and continuing so. Then the resulting transfer functions of each stage are:

$$H_1 = \frac{1+z^{-2}}{1-1.2z^{-1}+0.294975z^{-2}}$$

$$H_2 = \frac{1+0.712824z^{-1}+z^{-2}}{1-0.413853z^{-1}+0.505521z^{-2}}$$

$$H_3 = \frac{1+0.05026z^{-1}+z^{-2}}{1-0.355163z^{-1}+0.82z^{-2}}$$

$$H_4 = \frac{1-1.995649z^{-1}+z^{-2}}{1-1.951062z^{-1}+0.9604z^{-2}}$$

The order in which the stages are set, depend on the resulting and internal signal growth of any of the possible combinations. The section with poles at the greatest distance from the unit circle will give the lowest signal gains. Systems with poles that are close to the unit circle tend to be unstable, and these stages should therefore be left until last to limit the level of internal signal growth.

The zeroes are given by the numerator of the transfer function of each stage. As it is possible to choose any set of numerator and denominator coefficients, we must consider the options of choosing which coefficients to pair as well as the ordering of the stages. To do this we must take into consideration the distance of each possible zero pair, to each of the four pairs of poles.

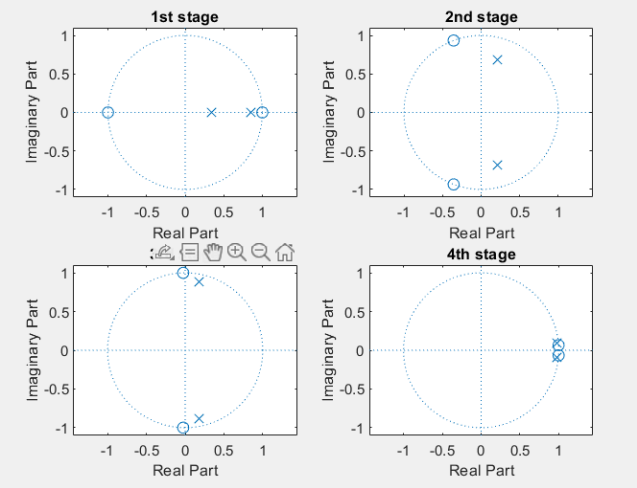

Figure 6: PZP plots for each stage.

Plotted above in figure 6, are the pole-zero-patterns of each stage presented in four subplots, it appears that this order is most acceptable. The distance of the poles from the unit circle increases gradually from stage to stage and each pair of poles and zeros are paired with their closest counterpart. A function could be written in MATLAB to calculate the radii of each pole pair as well as its closest zero pair; this would ensure that the pole-zero pairing is optimal. However, with the current coefficients it appears that the pairing is ideal, the first pole pair have the lowest radii and are paired with the closest possible zeros. The final stage's pole zero pairing is one that has its poles closest to the unit circle; the zeros are also positioned as close as possible.

Chapter 3: Bit-true behavioral filter model

The structure of the Simulink model is identical to that of the cascaded filter architecture outlined in chapter 1. The Simulink blocks that this filter structure makes use of are the sum, gain and delay. Each section of the cascaded structure is built separately and broken up into subsystems, scopes and the ‘ToWorkspace’ block are used to output the data from the simulation. Each section is built using its equivalent components in the Simulink model library, and the parameters of each block are double checked to make sure that the proper quantisation and datatype is being used at each stage in the computation. The addition of gain scaling greatly decreases the signal level at the output by a factor of about 11. The simulink model is tested against an identical floating point model and is compared to evaluate any noise introduced from quantisation.

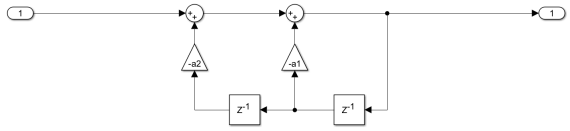

Figure 7; Simulink model for D block.

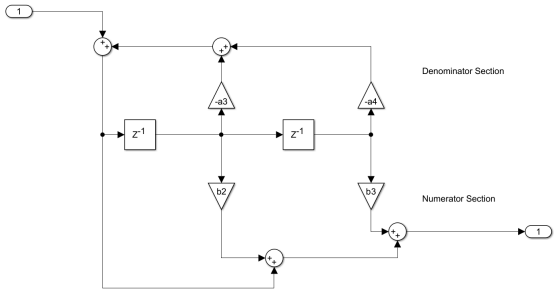

Figure 8; DN block subsystem built in simulink.

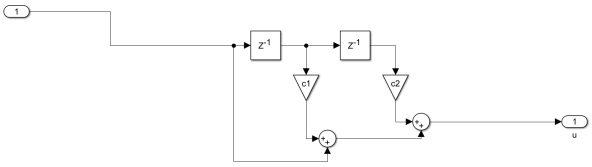

Figure 9; N block model in Simulink shown on the left.

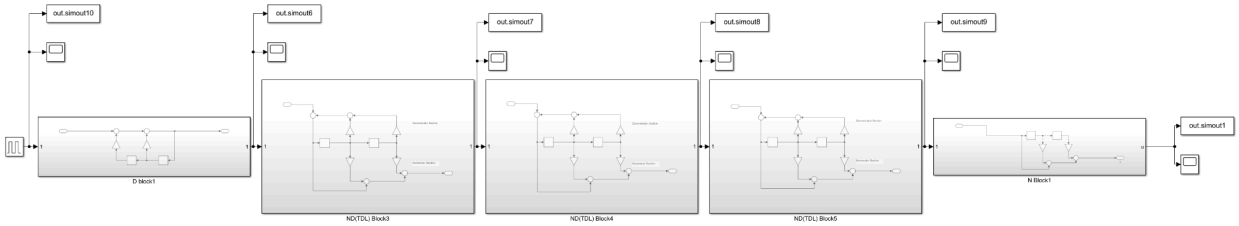

Figure 10; Simulink model of the total cascaded filter structure.

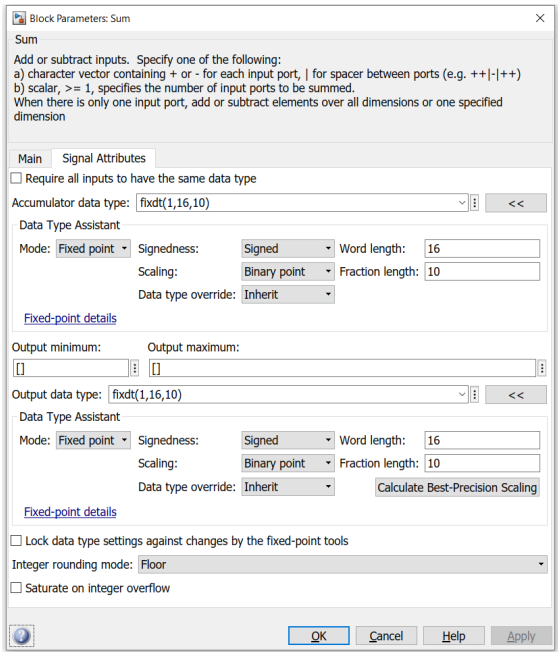

Figure 11; Block parameters of a Sum block in Simulink.

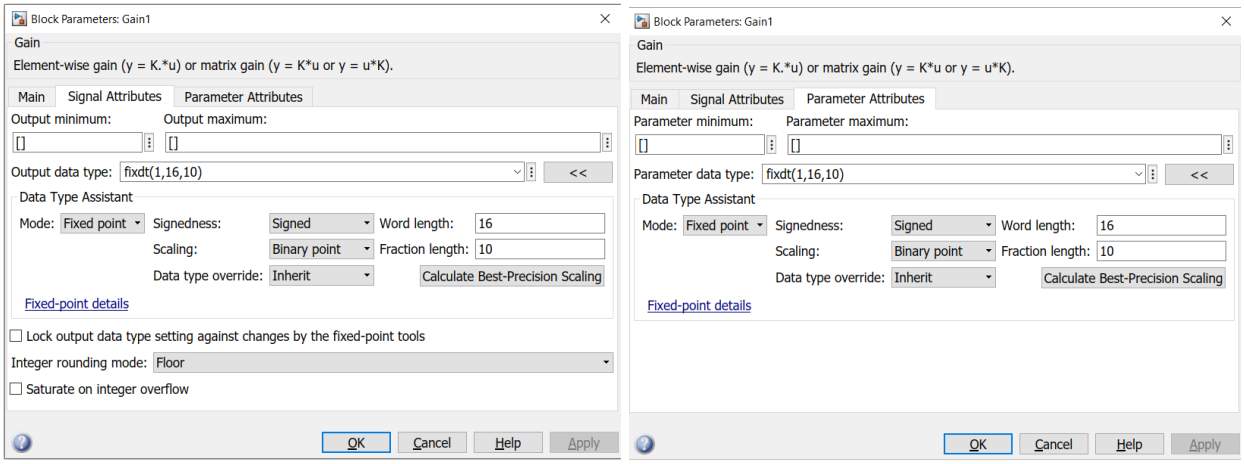

Figure 11 and 12; Signal and parameter attributes of a Gain block in Simulink.

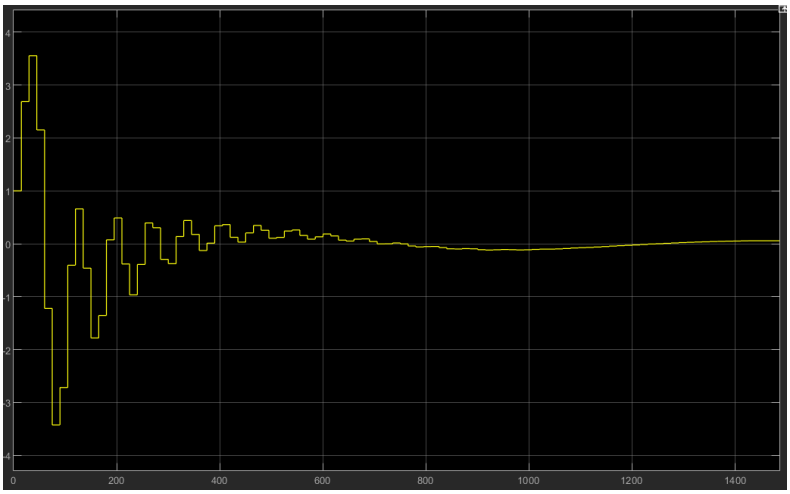

Figure 13; Filter output without stage gain scaling.

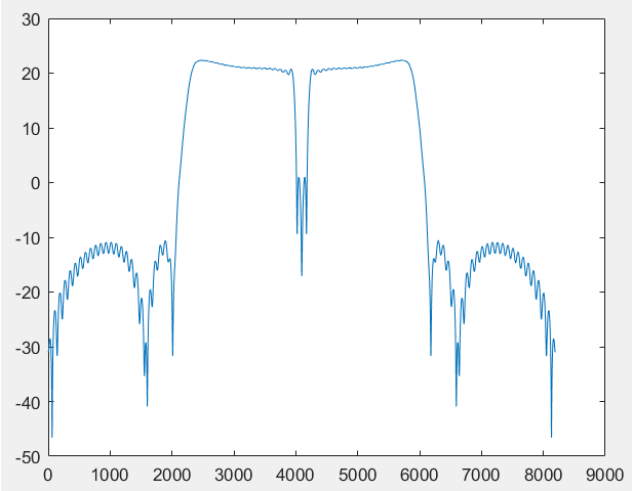

Figure 14; FFT plot of filter without stage gain scaling.

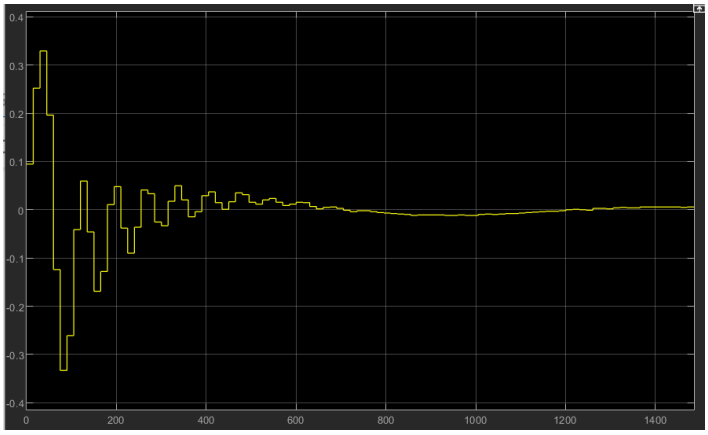

Figure 15; Filter output with stage gain scaling.

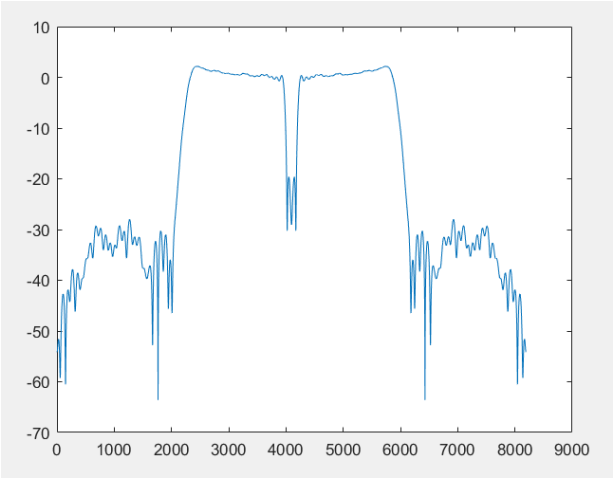

Figure 16; FFT plot of filter with stage gain scaling.

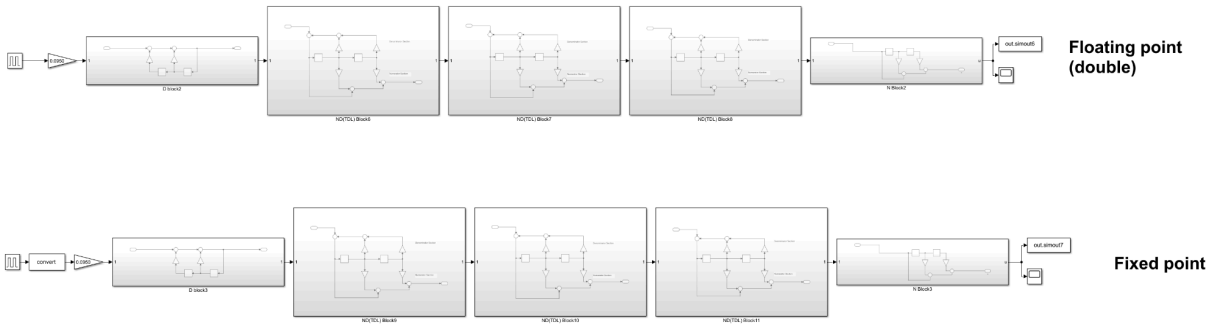

Figure 17; Separate models for floating point and fixed point data types.

Chapter 4: Data and Control Processors

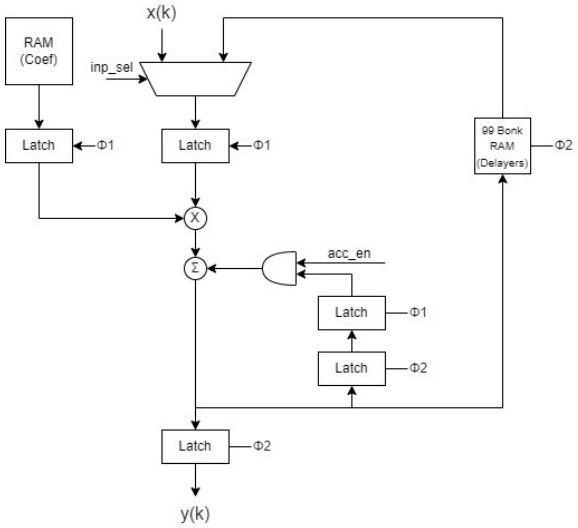

Figure 18; General purpose IIR TDL processor structure.

A single multiplier and adder are used to compute the results. Either the input signal x(k) or a delayed input sample from the RAM comes through the multiplexer and is stored in a latch, the appropriate coefficient is fetched from the RAM to be multiplied and accumulated with a delayed result depending on the value of the accumulator enable signal ‘acc_en’. The result from the adder is stored in the latch as a partial product. The result of the addition is stored in the two latches clocked on Φ2 & Φ1. This partial sum is stored and summed with the next state multiplied by a coefficient. This loop will stop when the last coefficient is used, the accumulator is cleared and the final value which is a sum of all the partial products produced from the multiplier and adder will then be stored in the final latch that is clocked on Φ2.

Chapter 5: Conclusion

The task was to produce a filter with an infinite impulse response, composed of four second order sections cascaded in sequence. Through analysis of the stage gains and pole zero pattern plots, the structure is adjusted to limit the possible signal growth. Initially starting with four DN sections, then using an ND cascade with a D block and N block at the beginning and end. Then each ND section was restructured to be in the reverse order so that the model can save on expensive delayer components. Unfortunately, it was not possible to complete both models of the filter due to time constraints and the limited man-power in our group. With only one person working on both building the models and writing up the report, a processor architecture was not built to compare with the operation of the cascaded structure.

A lot was learned from this exercise, ensuring the operation of a system such as this one requires in-depth analysis, multiple equivalent models to compare with and a thorough investigation of all possible methods that could improve quality of results. If there was more time to also determine the L_2 and L_∞ norms at each node for every stage, then there could be more appropriate stage gain scaling. With a processor architecture level model either built in Simulink or coded in VHDL, the models could be compared to ensure that they are bit-true. With consistent fixed point arithmetic, the output of both models should be identical. This verification of operation across both models of the same filter would give a great amount of confidence in the operation of the design.