Table of contents

Figure list

Tabular list

1 Introduction

In this practical exercise, the objective is to design and assess two variants of a digital low-pass filter utilizing the Xilinx System Generator (or Model Composer) workflow. This system integrates Matlab-Simulink with Xilinx Vivado, enabling swift prototyping of DSP systems on FPGAs through a model-based architectural method. The equation from which the designs are constructed is shown below:

$$y(n) = \sum_{i=0}^{17} a_i x(n-i) \tag{1}$$

The first design is without pipelining or optimization, while the second one can include techniques such as pipelining, register-retiming, and data broadcast, improving critical path delay but increasing area complexity. For these two designs the appropriate fixed-point word lengths for both the input (gateway in blocks) and the filter coefficients has to be found. These designs will be compared to a software-based filter with a focus on decreasing the signal to error rate (SER). After pinpointing the most effective word lengths and coefficients that result in the lowest SER for both designs, metrics such as FPGA resource consumption, critical path delay, peak frequency, and throughput will be examined. This examination will be done using HDL netlist generation, targeting the Artix-7 FPGA series.

2 Design Description

2.1 Unoptimized Design Description

As mentioned in the introduction, the unoptimized design does not implement pipelining or any form of optimization. The design focuses only on using the basic HDL elements available in the Simulink library browser. The architecture uses components like delays, which manage timing and data synchronization; CMults, responsible for multiplication operations; and Addsubs, which handle addition. Furthermore, it also includes gateway inputs and outputs, connections to and from the workspace, and the Vitis Model Composer Hub. The design features 18 delays, 17 CMults, and 16 Addsubs. The first and final delays signify the beginning and the end of the operation (Figure: 1).

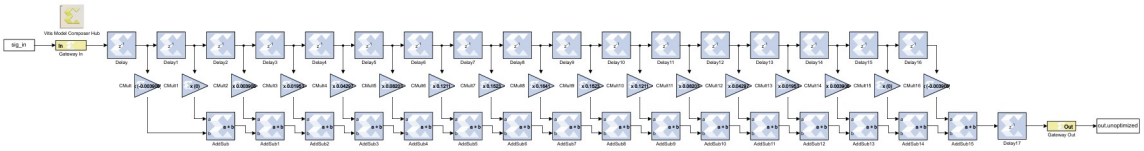


Figure 1: Block Diagram of Unoptimized Design

2.2 Optimized Design Description

During the system optimization process, many methods were explored. However, for this particular task, the folding technique was utilize. This decision was made because the values are symmetrically reflected around the central value, as shown in (Figure: 2).

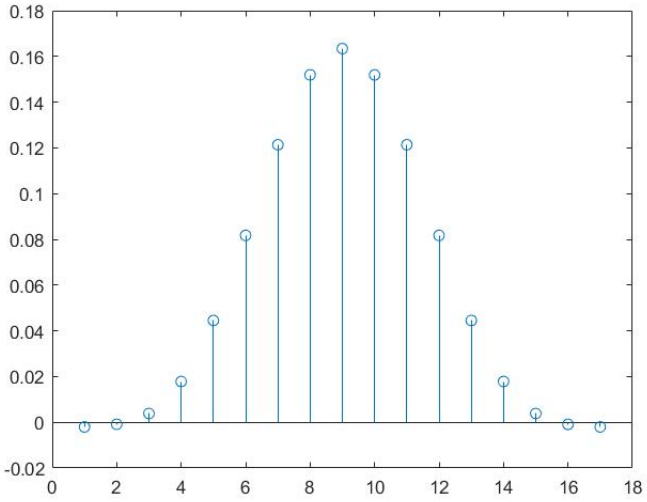


Figure 2: Values Mirrored Around the Center

The optimized version is displayed below (Figure: 3). By implementing folding technique, the design reduced its resource usage from 17 CMults to 9, while the count of Addsub units and delays remained unchanged.

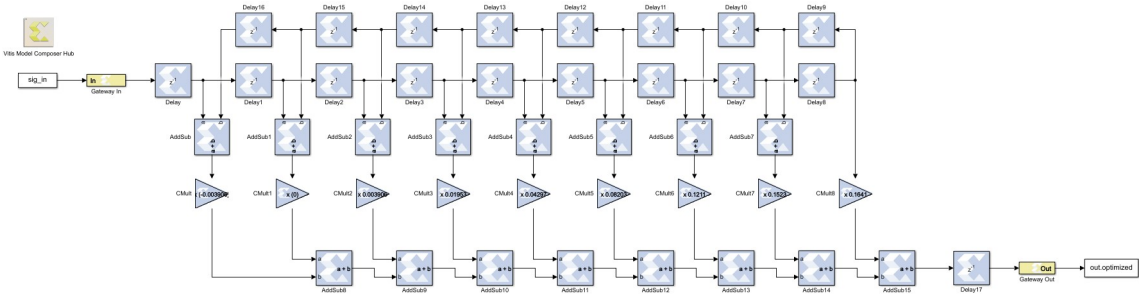


Figure 3: Block Diagram of Optimized Design

3 Simulation Results

In the simulation results section of the report, the objective is to determine the fixed-point word lengths for signals (W and D) and fixed-point word lengths for filter coefficients (Wc and Dc). The aim is then to compare the software filter with both the unoptimized and optimized designs to decide which one aligns most accurately with the software filter. To understand the comparison, here is the original signal compared to the software filter.

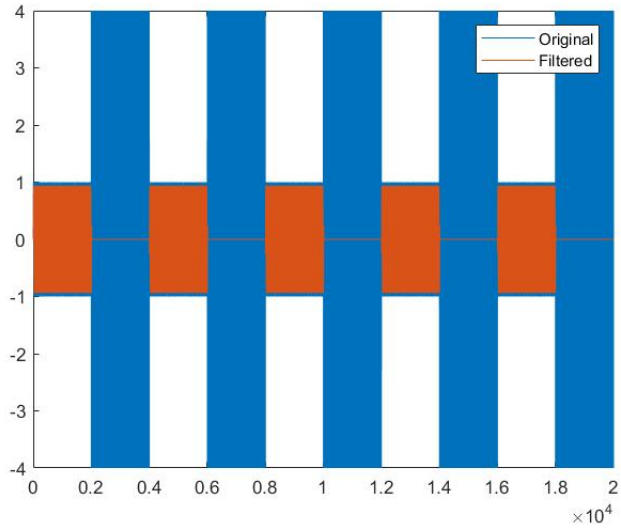


Figure 4: Original vs Software

To determine which variations in word lengths and filter coefficients provide the lowest error rate, the equation for the signal to error ratio will be used.

$$\text{Error Energy} = \sum |\text{Hardware} - \text{Software}|^2 \tag{2}$$

$$\text{Signal Energy} = \sum |\text{Software}|^2 \tag{3}$$

$$\text{SER} = 10 \cdot \log\left(\frac{\text{Error Energy}}{\text{Signal Energy}}\right) \tag{4}$$

In the equation, "Hardware" refers to both the unoptimized and optimized filters, depending on which one we want to calculate first. The absolute value is taken in these formulas to ensure that all differences, regardless of their initial sign, contribute positively to the computed energy. The sum is taken of the result because "Hardware" and "Software" consist of a [1:20000] matrix.

3.1 Unoptimized Simulation Results

Before determining the fixed-point word lengths for signals and the fixed-point word lengths for filter coefficients in the unoptimized design, it's essential to check if the sofwtare signal and unoptimized signals align for an accurate comparison. The standard values for word lengths and coefficients, set by the computer when placing blocks in Simulink, were W = 16, D = 12, Wc = 16, and Dc = 14. For simplicity, the plot displays from 1:500 samples, reducing the time needed to zoom in on the signals. Below, the plot shows both signals in their raw form (Figure: 5).

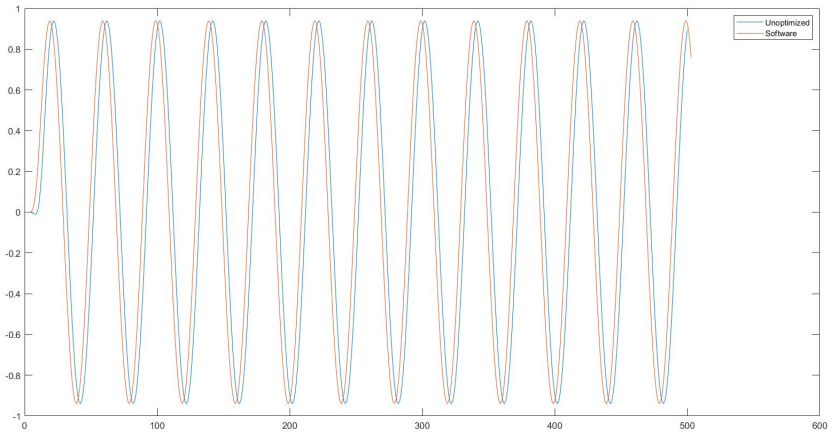


Figure 5: Unoptimized vs Software Without Padding

In the figure, an offset is observed between the unoptimized and software signals, making a direct comparison of the two inaccurate in their current state. To address this issue, a technique called padding can be employed. By adding zeros before and after the signal, the two signals can be aligned, ensuring they operate on the same matrix scale.

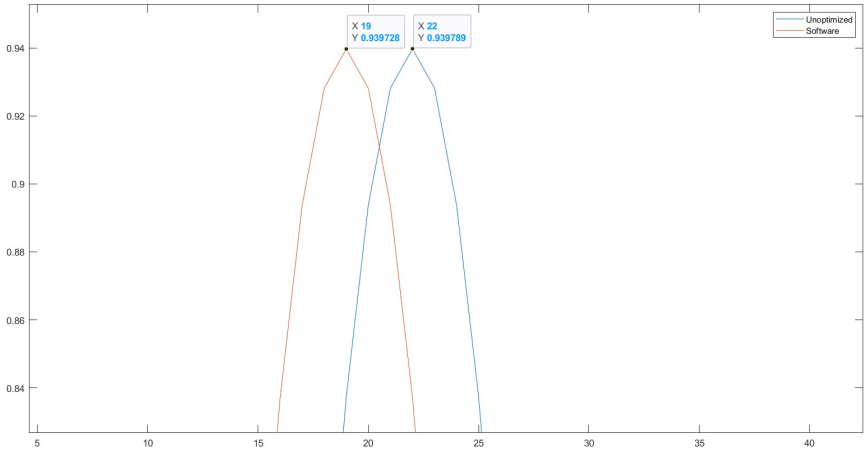


Figure 6: Zoomed in Unoptimized vs Software Without Padding

In this situation, there is an observed offset of 3 between the software and the unoptimized filter in (Figure: 6). By introducing 3 zeros at the beginning of the software filter, the issue can be resolved. In theory, a corresponding adjustment would typically require adding 3 zeros to the end of the unoptimized filter. However, due to 1 latency from the workspace where a value is already incorporated into the unoptimized filter, only 2 zeros need to be added to its end.

Under the results of padding (Figure: 7) is shown, where the two signals are aligned. This means that the testing of the best values for the fixed-point word lengths and the fixed-point word lengths for filter coefficients can begin.

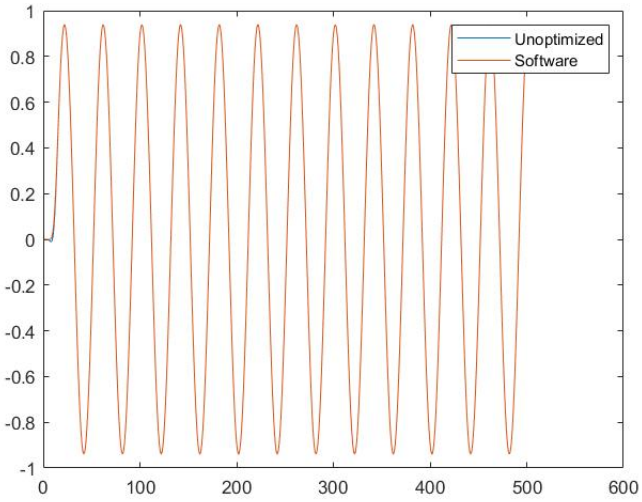


Figure 7: Unoptimized vs Software With Padding

To determine the optimal values for the fixed-point word lengths and the fixed-point word lengths for filter coefficients, equation SER is used (Equation: 1). The values of W and D are identified by setting the Wc and Dc to Simulink's standard values as an assumption to test for the most accurate W and D. Specifically, Wc is set to 16, and Dc is defined as Wc-2. A test was preformed to see whether Wc-4 would impact the SER differently, but no effect was observed, leading to the choice of using Wc-2. To find the lowest SER, the value for W was tested from 5 to 32, with D equaling W-4. The reason for D = W-4 is based on the highest amplitude of the signal being 4 and -4. In the second complement, 3 bits are required to represent 4 and 4 bits for -4. Therefore, W needs to exceed 4 to obtain a valid value. The upper boundary is set at 32 since, theoretically, the accuracy of the word length plateaus after a specific value. Hence, committing more resources beyond this point would not result in a notable increase in accuracy. Below, the plot illustrates the value of W ranging from 5 to 32, indicating that the W which results in the lowest SER is 16 and it plateaus after 22 (Figure: 8).

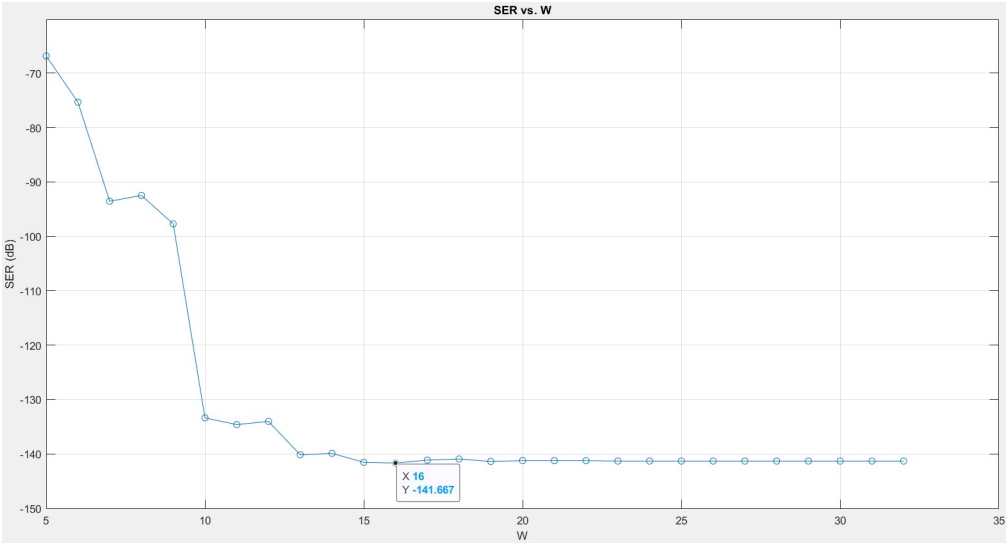


Figure 8: SER(DB) vs W Unoptimized Plot

The next step involves determining the optimal fixed-point word lengths for filter coefficients that result in the lowest SER. To accomplish this, the same technique that was used to determine the

fixed-point word lengths is used. From the discovery of the best fixed-point word lengths, it was found that the ideal fixed-point word length is 16, with D being W-4. Using the same method with using W = 16 and D = W-4, values for Wc ranging from 5 to 32 and Dc = Wc-2 were plotted to determine the optimal value. The plot shows that a W value of 19 results in the lowest SER. However, considering the negligible difference in output (less than 0.000001) and the increased resource usage, the decision was made to set it to 16. You can also see that the graph plateaus after 23 (Figure: 9).

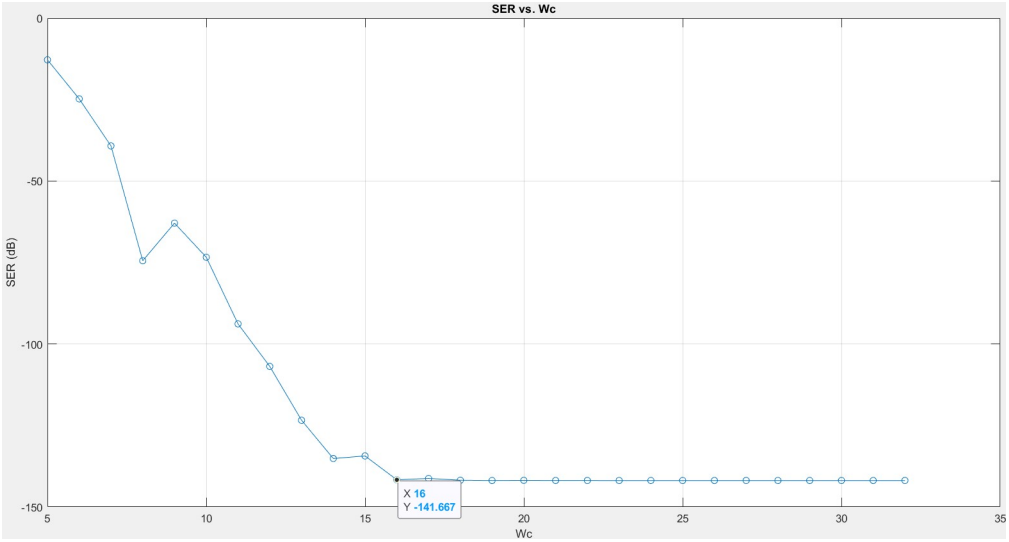


Figure 9: SER(DB) vs Wc Unoptimized Plot

After determining the fixed-point word lengths for signals and the optimal fixed-point word lengths for filter coefficients that results in the lowest SER, a comparison can be made between the unoptimized and the software version. Below is a plot comparing the two, with a low difference of 0.000001, indicating that the unoptimized version is an effective filter (Figure: 7). The calculated SER for the unoptimized version is -141.6667, indicating a very low error rate.

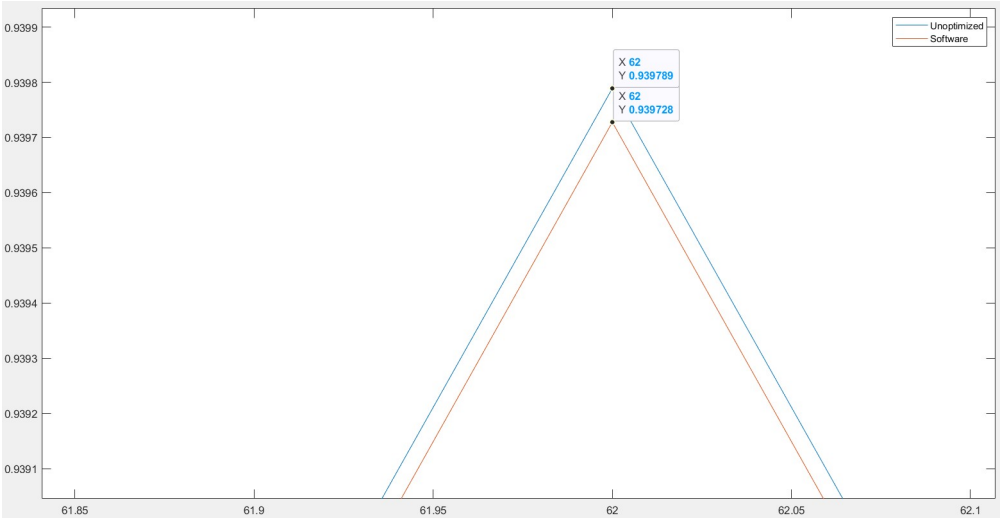


Figure 10: Zoomed in Unoptimized vs Software With Padding

3.2 Optimized Simulation Results

For the optimized version, the same process used for the unoptimized version was followed. Both the optimized and software versions were first padded to ensure alignment, and then the ideal W was determined, with the best value for Wc assumed to be 16. The main difference was that using D = W-4 resulted in a poor SER and low pass filtering. Therefore, it was adjusted to D = W-1 to achieve a filter performance comparable to the software version. The value for Dc remained unchanged at Dc = W-2. Below, the plot shows the value of W ranging from 5 to 32. It reveals that the optimal W for the lowest SER is 17, and it plateaus after 20 (Figure: 11).

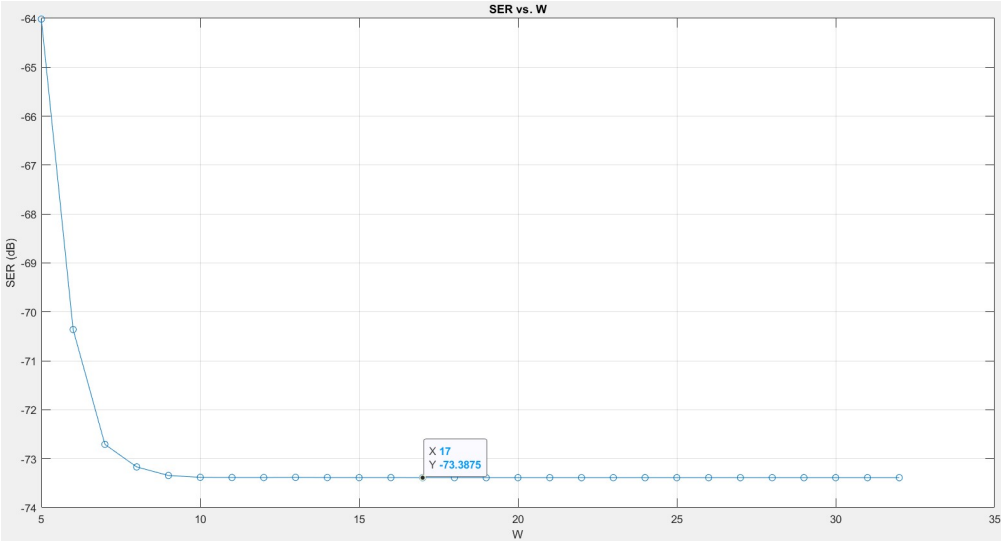


Figure 11: SER(DB) vs W Optimized Plot

Next, to determine the optimal fixed-point word lengths for filter coefficients Wc for the optimized version, W is set to 17. The same plotting method as previously described is then used to identify the value. Showing that the optimal Wc for the lowest SER is 10, and it plateaus after 16 (Figure: 12).

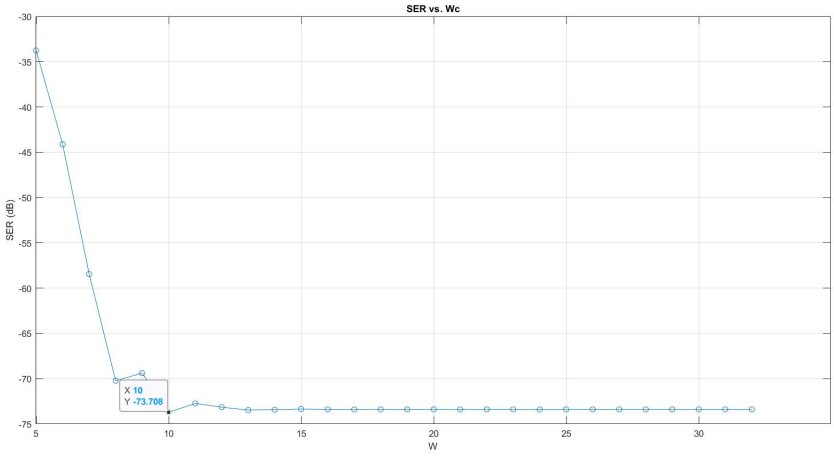


Figure 12: SER(DB) vs Wc Optimized Plot

Below is the plot comparing the software signal to the optimized version using the best fixed-point word lengths and filter coefficients (Figure: 13). The graph indicates that they are well-aligned and appear quite similar.

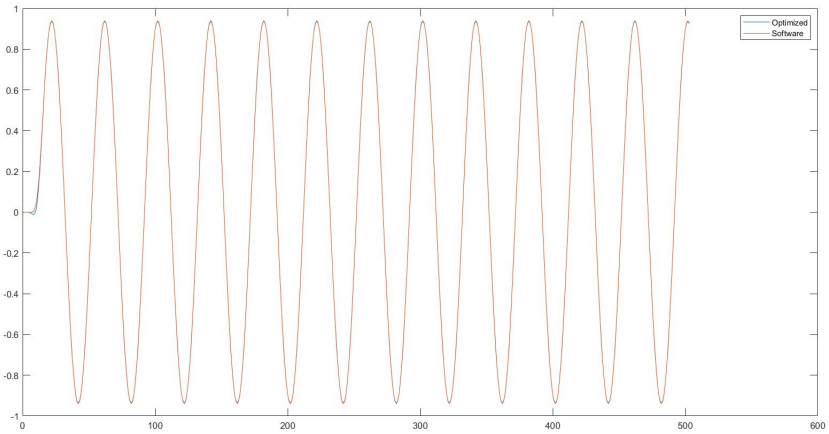


Figure 13: Optimized vs Software With Padding

Below is a detailed plot comparing the software and optimized versions (Figure: 14). The difference between them is 0.002228, suggesting that the optimized version is a competent filter with a low SER of -73.7080. When comparing the optimized SER to the unoptimized version's SER of -141.6667, it's logical because the optimized version is the folded version of the unoptimized, resulting in half the SER.

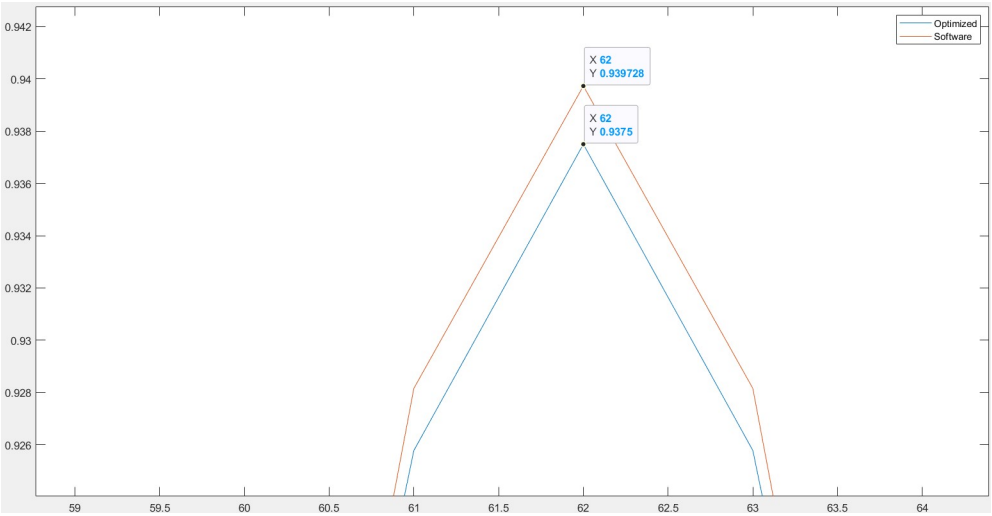


Figure 14: Zoomed in Optimized vs Software With Padding

4 Post Synthesis Results

In the post-synthesis results of the report, the goal is to evaluate the resource usage and time outcomes for various designs. The aim is to then compare them to determine which design is the most resource-efficient and effective.

4.1 Unoptimized Post Synthesis Results

Below, the post-synthesis details of the resources for the unoptimized version are presented, showing its use of 1640 LUTs and 288 registers (Figure: 15).

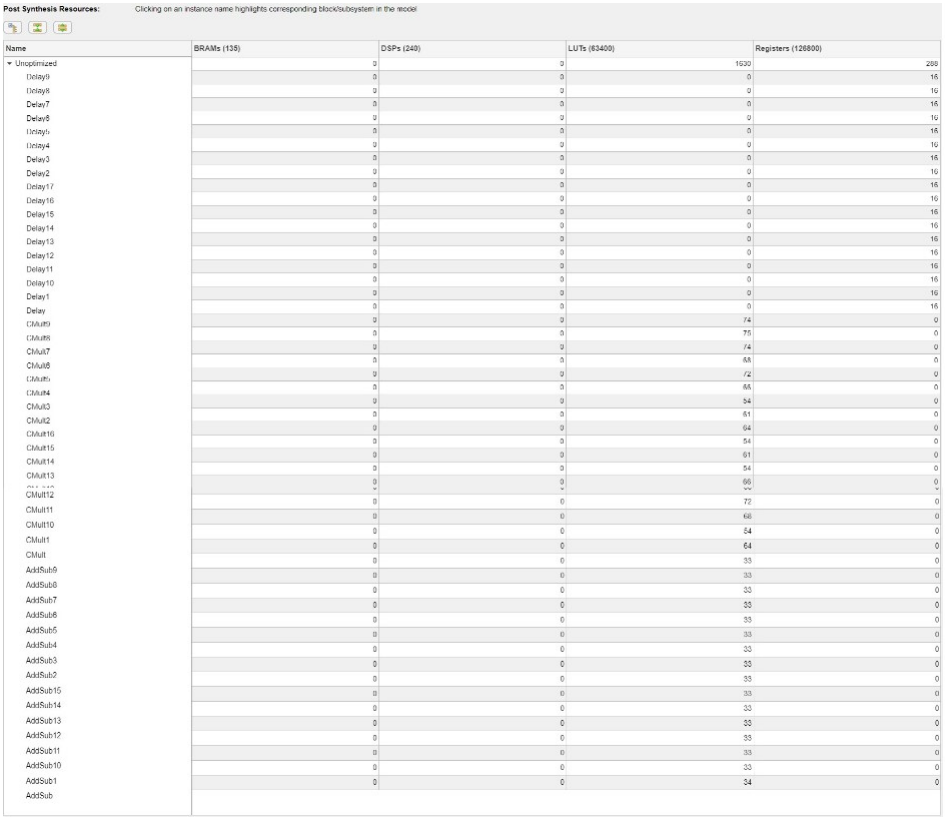
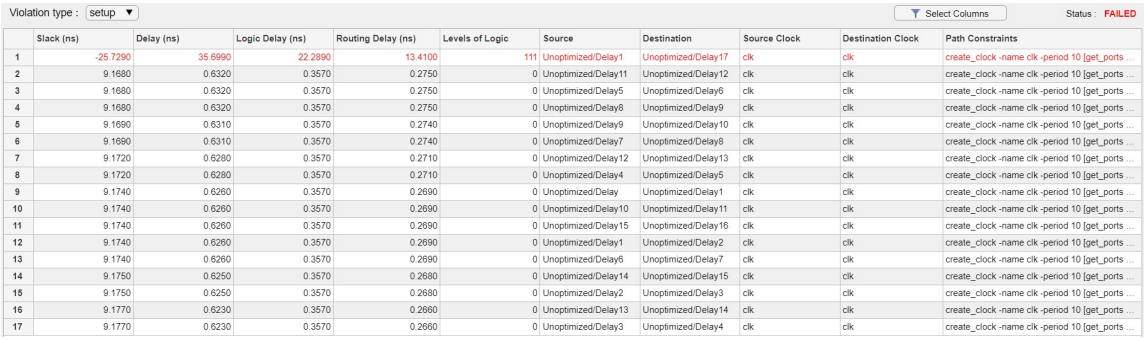


Figure 15: Unotimized Post Synthesis Resources

Under, the post-synthesis timing for the unoptimized version is displayed. It reveals that the system fails at a 10 ns runtime because it doesn't have adequate time, resulting in a slack of –25.7290 (Figure: 16). This suggests that we need to add this slack to the 10 ns runtime to provide the system sufficient time to operate. Due to resource constraints on the FPGA clock, achieving 100% accuracy isn't possible. For simplicity, the following test was ran at a 36 ns runtime instead of 35.7290 ns. Here, it is shown that the system passed, having an additional slack of 0.2710 to spare (Figure: 17).



	Slack (ns)	Delay (ns)	Logic Delay (ns)	Routing Delay (ns)	Levels of Logic	Source	Destination	Source Clock	Destination Clock	Path Constraints
1	-25.7290	35.6990	22.2890	13.4100	111	Unoptimized/Delay1	Unoptimized/Delay17	clk	clk	create_clock -name clk -period 10 [get_ports ...
2	9.1680	0.6320	0.3570	0.2750	0	Unoptimized/Delay11	Unoptimized/Delay12	clk	clk	create_clock -name clk -period 10 [get_ports ...
3	9.1680	0.6320	0.3570	0.2750	0	Unoptimized/Delay5	Unoptimized/Delay6	clk	clk	create_clock -name clk -period 10 [get_ports ...
4	9.1680	0.6320	0.3570	0.2750	0	Unoptimized/Delay8	Unoptimized/Delay9	clk	clk	create_clock -name clk -period 10 [get_ports ...
5	9.1690	0.6310	0.3570	0.2740	0	Unoptimized/Delay9	Unoptimized/Delay10	clk	clk	create_clock -name clk -period 10 [get_ports ...
6	9.1690	0.6310	0.3570	0.2740	0	Unoptimized/Delay7	Unoptimized/Delay8	clk	clk	create_clock -name clk -period 10 [get_ports ...
7	9.1720	0.6280	0.3570	0.2710	0	Unoptimized/Delay12	Unoptimized/Delay13	clk	clk	create_clock -name clk -period 10 [get_ports ...
8	9.1720	0.6280	0.3570	0.2710	0	Unoptimized/Delay4	Unoptimized/Delay5	clk	clk	create_clock -name clk -period 10 [get_ports ...
9	9.1740	0.6260	0.3570	0.2690	0	Unoptimized/Delay	Unoptimized/Delay1	clk	clk	create_clock -name clk -period 10 [get_ports ...
10	9.1740	0.6260	0.3570	0.2690	0	Unoptimized/Delay10	Unoptimized/Delay11	clk	clk	create_clock -name clk -period 10 [get_ports ...
11	9.1740	0.6260	0.3570	0.2690	0	Unoptimized/Delay15	Unoptimized/Delay16	clk	clk	create_clock -name clk -period 10 [get_ports ...
12	9.1740	0.6260	0.3570	0.2690	0	Unoptimized/Delay1	Unoptimized/Delay2	clk	clk	create_clock -name clk -period 10 [get_ports ...
13	9.1740	0.6260	0.3570	0.2690	0	Unoptimized/Delay6	Unoptimized/Delay7	clk	clk	create_clock -name clk -period 10 [get_ports ...
14	9.1750	0.6250	0.3570	0.2680	0	Unoptimized/Delay14	Unoptimized/Delay15	clk	clk	create_clock -name clk -period 10 [get_ports ...
15	9.1750	0.6250	0.3570	0.2680	0	Unoptimized/Delay2	Unoptimized/Delay3	clk	clk	create_clock -name clk -period 10 [get_ports ...
16	9.1770	0.6230	0.3570	0.2660	0	Unoptimized/Delay13	Unoptimized/Delay14	clk	clk	create_clock -name clk -period 10 [get_ports ...
17	9.1770	0.6230	0.3570	0.2660	0	Unoptimized/Delay3	Unoptimized/Delay4	clk	clk	create_clock -name clk -period 10 [get_ports ...

Figure 16: Unotimized Post Synthesis Time Failed

Violation type : setup ▼ Select Columns Status : PASSED

	Slack (ns)	Delay (ns)	Logic Delay (ns)	Routing Delay (ns)	Levels of Logic	Source	Destination	Source Clock	Destination Clock	Path Constraints
1	0.2710	35.6990	22.2890	13.4100	111	Unoptimized/Delay1	Unoptimized/Delay17	clk	clk	create_clock -name clk -period 36 [get_ports ...
2	35.1680	0.6320	0.3570	0.2750	0	Unoptimized/Delay11	Unoptimized/Delay12	clk	clk	create_clock -name clk -period 36 [get_ports ...
3	35.1680	0.6320	0.3570	0.2750	0	Unoptimized/Delay5	Unoptimized/Delay6	clk	clk	create_clock -name clk -period 36 [get_ports ...
4	35.1680	0.6320	0.3570	0.2750	0	Unoptimized/Delay8	Unoptimized/Delay9	clk	clk	create_clock -name clk -period 36 [get_ports ...
5	35.1690	0.6310	0.3570	0.2740	0	Unoptimized/Delay9	Unoptimized/Delay10	clk	clk	create_clock -name clk -period 36 [get_ports ...
6	35.1690	0.6310	0.3570	0.2740	0	Unoptimized/Delay7	Unoptimized/Delay8	clk	clk	create_clock -name clk -period 36 [get_ports ...
7	35.1720	0.6280	0.3570	0.2710	0	Unoptimized/Delay12	Unoptimized/Delay13	clk	clk	create_clock -name clk -period 36 [get_ports ...
8	35.1720	0.6280	0.3570	0.2710	0	Unoptimized/Delay4	Unoptimized/Delay5	clk	clk	create_clock -name clk -period 36 [get_ports ...
9	35.1740	0.6260	0.3570	0.2690	0	Unoptimized/Delay	Unoptimized/Delay1	clk	clk	create_clock -name clk -period 36 [get_ports ...
10	35.1740	0.6260	0.3570	0.2690	0	Unoptimized/Delay10	Unoptimized/Delay11	clk	clk	create_clock -name clk -period 36 [get_ports ...
11	35.1740	0.6260	0.3570	0.2690	0	Unoptimized/Delay15	Unoptimized/Delay16	clk	clk	create_clock -name clk -period 36 [get_ports ...
12	35.1740	0.6260	0.3570	0.2690	0	Unoptimized/Delay1	Unoptimized/Delay2	clk	clk	create_clock -name clk -period 36 [get_ports ...
13	35.1740	0.6260	0.3570	0.2690	0	Unoptimized/Delay6	Unoptimized/Delay7	clk	clk	create_clock -name clk -period 36 [get_ports ...
14	35.1750	0.6250	0.3570	0.2680	0	Unoptimized/Delay14	Unoptimized/Delay15	clk	clk	create_clock -name clk -period 36 [get_ports ...
15	35.1750	0.6250	0.3570	0.2680	0	Unoptimized/Delay2	Unoptimized/Delay3	clk	clk	create_clock -name clk -period 36 [get_ports ...
16	35.1770	0.6230	0.3570	0.2660	0	Unoptimized/Delay13	Unoptimized/Delay14	clk	clk	create_clock -name clk -period 36 [get_ports ...
17	35.1770	0.6230	0.3570	0.2660	0	Unoptimized/Delay3	Unoptimized/Delay4	clk	clk	create_clock -name clk -period 36 [get_ports ...

Figure 17: Unotimized Post Synthesis Time Passed

4.2 Optimized Post Synthesis Results

Under, the post-synthesis details of the resources for the Optimized version are displayed, showing its use of 669 LUTs and 299 registers (Figure: 18).

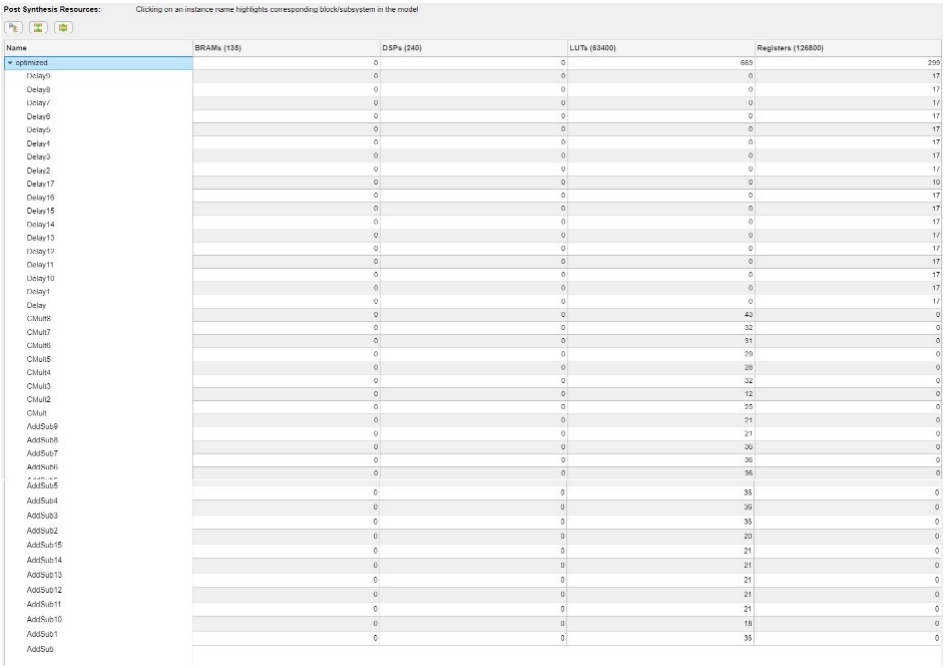


Figure 18: Otimized Post Synthesis Resources

Below, the post-synthesis timing for the optimized version is presented. It shows that the system fails at a 10 ns runtime due to insufficient time, leading to a slack of -11.5660 (Figure: 19). This indicates the need to add this slack to the 10 ns runtime to allow the system enough operation time. Given the resource limits of the FPGA clock, reaching 100% accuracy is not possible. For simplicity, the following test was ran at a 22 ns runtime instead of 21.5660 ns. In this instance, the system successfully passed with an extra slack of 0.4340 available (Figure: 20).

Violation type : setup ▼ Select Columns Status : FAILED

	Slack (ns)	Delay (ns)	Logic Delay (ns)	Routing Delay (ns)	Levels of Logic	Source	Destination	Source Clock	Destination Clock	Path Constraints
1	-11.5660	21.5360	11.2950	10.2410	53	optimized/Delay16	optimized/Delay17	clk	clk	create_clock -name clk -period 10 [get_ports ...
2	9.1790	0.6210	0.3570	0.2640	0	optimized/Delay8	optimized/Delay9	clk	clk	create_clock -name clk -period 10 [get_ports ...
3	9.2010	0.5990	0.3570	0.2420	0	optimized/Delay	optimized/Delay1	clk	clk	create_clock -name clk -period 10 [get_ports ...
4	9.2010	0.5990	0.3570	0.2420	0	optimized/Delay1	optimized/Delay2	clk	clk	create_clock -name clk -period 10 [get_ports ...
5	9.2010	0.5990	0.3570	0.2420	0	optimized/Delay2	optimized/Delay3	clk	clk	create_clock -name clk -period 10 [get_ports ...
6	9.2010	0.5990	0.3570	0.2420	0	optimized/Delay3	optimized/Delay4	clk	clk	create_clock -name clk -period 10 [get_ports ...
7	9.2010	0.5990	0.3570	0.2420	0	optimized/Delay4	optimized/Delay5	clk	clk	create_clock -name clk -period 10 [get_ports ...
8	9.2010	0.5990	0.3570	0.2420	0	optimized/Delay5	optimized/Delay6	clk	clk	create_clock -name clk -period 10 [get_ports ...
9	9.2010	0.5990	0.3570	0.2420	0	optimized/Delay6	optimized/Delay7	clk	clk	create_clock -name clk -period 10 [get_ports ...
10	9.2010	0.5990	0.3570	0.2420	0	optimized/Delay7	optimized/Delay8	clk	clk	create_clock -name clk -period 10 [get_ports ...
11	9.2070	0.5930	0.3570	0.2360	0	optimized/Delay9	optimized/Delay10	clk	clk	create_clock -name clk -period 10 [get_ports ...
12	9.2070	0.5930	0.3570	0.2360	0	optimized/Delay10	optimized/Delay11	clk	clk	create_clock -name clk -period 10 [get_ports ...
13	9.2070	0.5930	0.3570	0.2360	0	optimized/Delay11	optimized/Delay12	clk	clk	create_clock -name clk -period 10 [get_ports ...
14	9.2070	0.5930	0.3570	0.2360	0	optimized/Delay12	optimized/Delay13	clk	clk	create_clock -name clk -period 10 [get_ports ...
15	9.2070	0.5930	0.3570	0.2360	0	optimized/Delay13	optimized/Delay14	clk	clk	create_clock -name clk -period 10 [get_ports ...
16	9.2070	0.5930	0.3570	0.2360	0	optimized/Delay14	optimized/Delay15	clk	clk	create_clock -name clk -period 10 [get_ports ...
17	9.2070	0.5930	0.3570	0.2360	0	optimized/Delay15	optimized/Delay16	clk	clk	create_clock -name clk -period 10 [get_ports ...

Figure 19: Otimized Post Synthesis Time Failed

Violation type : setup ▼ Select Columns Status : PASSED

	Slack (ns)	Delay (ns)	Logic Delay (ns)	Routing Delay (ns)	Levels of Logic	Source	Destination	Source Clock	Destination Clock	Path Constraints
1	0.4340	21.5360	11.2950	10.2410	53	optimized/Delay16	optimized/Delay17	clk	clk	create_clock -name clk -period 22 [get_ports ...
2	21.1790	0.6210	0.3570	0.2640	0	optimized/Delay8	optimized/Delay9	clk	clk	create_clock -name clk -period 22 [get_ports ...
3	21.2010	0.5990	0.3570	0.2420	0	optimized/Delay	optimized/Delay1	clk	clk	create_clock -name clk -period 22 [get_ports ...
4	21.2010	0.5990	0.3570	0.2420	0	optimized/Delay1	optimized/Delay2	clk	clk	create_clock -name clk -period 22 [get_ports ...
5	21.2010	0.5990	0.3570	0.2420	0	optimized/Delay2	optimized/Delay3	clk	clk	create_clock -name clk -period 22 [get_ports ...
6	21.2010	0.5990	0.3570	0.2420	0	optimized/Delay3	optimized/Delay4	clk	clk	create_clock -name clk -period 22 [get_ports ...
7	21.2010	0.5990	0.3570	0.2420	0	optimized/Delay4	optimized/Delay5	clk	clk	create_clock -name clk -period 22 [get_ports ...
8	21.2010	0.5990	0.3570	0.2420	0	optimized/Delay5	optimized/Delay6	clk	clk	create_clock -name clk -period 22 [get_ports ...
9	21.2010	0.5990	0.3570	0.2420	0	optimized/Delay6	optimized/Delay7	clk	clk	create_clock -name clk -period 22 [get_ports ...
10	21.2010	0.5990	0.3570	0.2420	0	optimized/Delay7	optimized/Delay8	clk	clk	create_clock -name clk -period 22 [get_ports ...
11	21.2070	0.5930	0.3570	0.2360	0	optimized/Delay9	optimized/Delay10	clk	clk	create_clock -name clk -period 22 [get_ports ...
12	21.2070	0.5930	0.3570	0.2360	0	optimized/Delay10	optimized/Delay11	clk	clk	create_clock -name clk -period 22 [get_ports ...
13	21.2070	0.5930	0.3570	0.2360	0	optimized/Delay11	optimized/Delay12	clk	clk	create_clock -name clk -period 22 [get_ports ...
14	21.2070	0.5930	0.3570	0.2360	0	optimized/Delay12	optimized/Delay13	clk	clk	create_clock -name clk -period 22 [get_ports ...
15	21.2070	0.5930	0.3570	0.2360	0	optimized/Delay13	optimized/Delay14	clk	clk	create_clock -name clk -period 22 [get_ports ...
16	21.2070	0.5930	0.3570	0.2360	0	optimized/Delay14	optimized/Delay15	clk	clk	create_clock -name clk -period 22 [get_ports ...
17	21.2070	0.5930	0.3570	0.2360	0	optimized/Delay15	optimized/Delay16	clk	clk	create_clock -name clk -period 22 [get_ports ...

Figure 20: Otimized Post Synthesis Time Passed

4.3 Comparing Unoptimized and Optimized

The Critical path delay is measured in seconds and represents the lengthiest combinational delay in the design between two registers. This delay determines the maximum clock rate. The critical path delay is derived using the formula:

$$F_{\text{samp}} = KF_{\text{clk}}$$

where F_{CLK} signifies the maximum clock rate governed by the critical path delay, K stands for the degree of parallel processing, and F_{samp} represents the throughput. The Critical path delay can be found in Figure: 17) and (Figure: 20). The maximum frequency can be determined by taking the inverse of the Critical path delay and since the desings does not use parallel processing, throughput equals the maximum frequency.

Table 1: Comparison between Unoptimized and Optimized

	Unoptimized	Optimized
LUTs	1640	669
Registers	288	299
Critical Path Delay	35.7290 ns	21.5660 ns
Maximum Frequency	28 MHz	46.4 MHz
Throughput	28 MHz	46.4 MHz

From the comparison, it's clear that the optimized design uses fewer resources, reducing the look-up tables from 1640 to 669. While there is a slight increase in resistors from 288 to 299, this is acceptable given the reduced space used on the FPGA. The time has also decreased from 35.7290 ns to 21.5660 ns, a reduction of 39.63%. This represents a significant increase in speed, maximum Frequency and throughput, making the optimized design far superior to the unoptimized one.

5 Discussion and Conclusion

In this practical, I effectively created two versions of a digital low-pass filter. The first design is a non-pipelined and unoptimized version, while the second design utilizes the folding technique. I have determined the the fixed-point word lengths for signals and the fixed-point word lengths for filter coefficients that minimize the Symbol Error Rate (SER) for both designs. Finally, I carried out resource and speed test to compare the two designs, ultimately determining that the optimized design was the superior option.

During this practical exercise, I acquired a deeper knowledge of employing MATLAB for crafting filter designs using Simulink. I learned how to configure systems in Simulink and identify strategies to enhance system efficiency, focusing on resource conservation and speed improvement. Additionally, I implemented a for-loop for SER calculation, allowing for efficient determination of optimal fixed-point word lengths for signals and filter coefficients, saving considerable time in the process.

During this learning process, I encountered several challenges. My initial version of MATLAB wasn't compatible with Vivado. Additionally, there was a mismatch between the software and hardware signals, requiring me to pad them for alignment. I also had to find an efficient method to determine the optimal fixed-point word lengths for both signals and filter coefficients to minimize the SER. Despite these setbacks, they provided valuable learning experiences. While there's potential to further optimize and speed up the system, I choice a simpler approach given this is an introductory phase to the project. In the project, I intend to explore improvements such as pipelining, register-retiming, and data broadcasting.

Overall, this practical was immensely beneficial, since much of the knowledge acquired will be applied in the project. Additionally, it served as a good refresher on using MATLAB and Simulink.

Marking Criteria

The pracs in this course are marked to a specific criteria. This means you must demonstrate sufficient understanding and functionality and the marking will be done according to the rubric provided below. You must attempt each prac (i.e., a report must be received by the due date) to pass the course. All designs VHDL code used **must be your own work.** You are NOT permitted to use other VHDL code sources, unless directed to. Plagiarism is unacceptable and please read and understand the School Statement on Misconduct, available on the ITEE website at: http://www.itee.uq.edu.au/itee-student-misconduct-including-plagiarism .

Marks	**Criteria**
Simulation	
0	Not sufficient results are provided
1	Results shown only for the un-optimised design – working correctly
2	Results shown only for the optimised design – working correctly
3	Results shown for both designs - working correctly
4	Both designs are working correctly, and a comparison is provided
Report	
0	No evidence of content or work
1	Some content, insufficient explanation of circuit
2	Reasonable content, some explanation of circuit (both approaches are explained)
3	Good content, reasonable explanation of circuit (both approaches are explained)
4	Excellent content, good explanation of circuit (both approaches are explained)
Oral assessment	
0	No knowledge of the design
1	Very little knowledge of the design
2	Reasonable knowledge of the design
3	Good knowledge of the design.
4	Excellent knowledge of the design.
Total (12):	**Marker Initials:** **Date:**