

Giaan Nguyen

Lab 4: Discrete Fourier Transform on FPGAs

ECEN 689-600: FPGA Information Processing Systems
Wednesday, March 3, 2021

Decimation-in-Time (DIT) Simulation

We consider computing the 8-point discrete Fourier transform (DFT) by means of implementing the radix-2 fast Fourier transform (FFT) algorithm on an FPGA. The three signals of interest are a sinusoidal wave (signal 1), a square wave (signal 2), and a triangular wave (signal 3). For reference, their actual DFTs are outlined in Figure 1; ideally, we want our values to be within 0.1 error.

| 8x3 complex double | | | |
|--------------------|-------------------|------------------|-------------------|
| | 1 | 2 | 3 |
| 1 | 0.5537 + 0.0000i | 4.0000 + 0.0000i | 4.0000 + 0.0000i |
| 2 | 2.3946 - 2.0970i | 1.0000 - 2.4142i | -1.7071 + 0.0000i |
| 3 | -1.3867 + 0.9156i | 0.0000 + 0.0000i | 0.0000 + 0.0000i |
| 4 | -0.8810 + 0.2804i | 1.0000 - 0.4142i | -0.2929 + 0.0000i |
| 5 | -0.8076 + 0.0000i | 0.0000 + 0.0000i | 0.0000 + 0.0000i |
| 6 | -0.8810 - 0.2804i | 1.0000 + 0.4142i | -0.2929 + 0.0000i |
| 7 | -1.3867 - 0.9156i | 0.0000 + 0.0000i | 0.0000 + 0.0000i |
| 8 | 2.3946 + 2.0970i | 1.0000 + 2.4142i | -1.7071 + 0.0000i |

Figure 1. The 8-point DFTs of a sine wave (column 1), square wave (column 2), and triangular wave (column 3).

First, we simulate the decimation-in-time (DIT) FFT on signal 1. Figure 2 shows the real and imaginary parts of the input and output, read in reverse order. We can confirm that the values are within 0.1 error of signal 1's DFT from Figure 1.

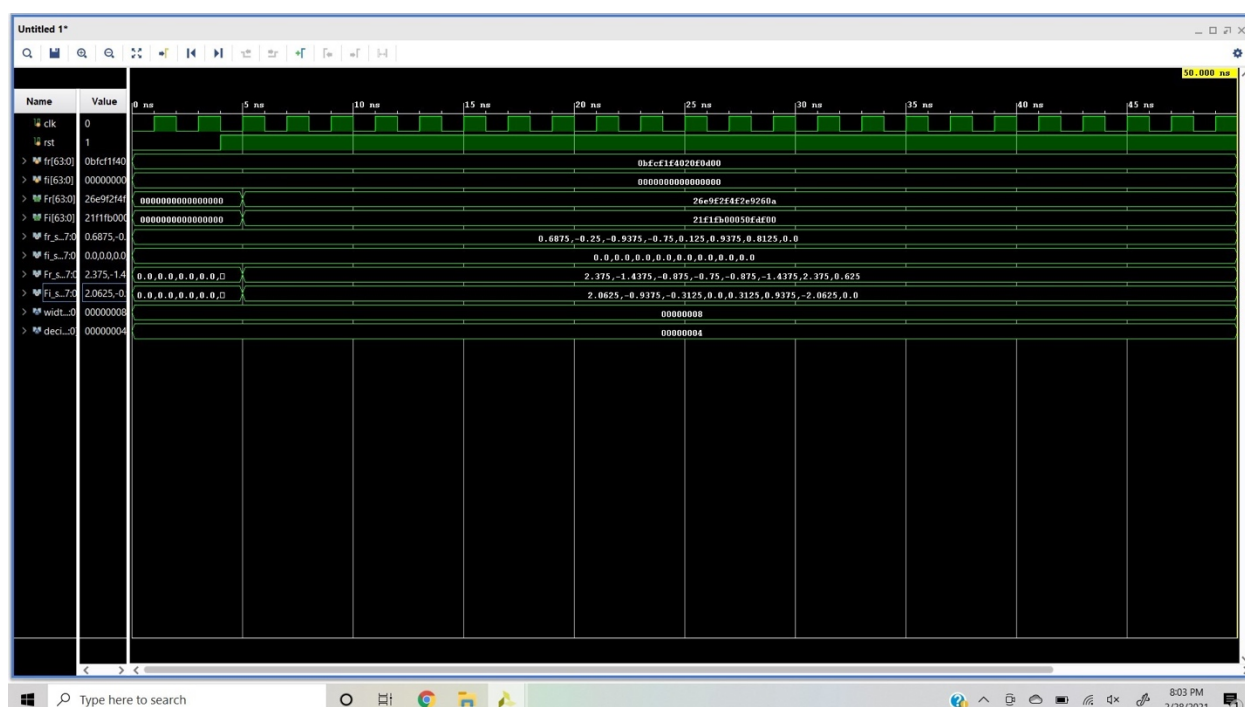
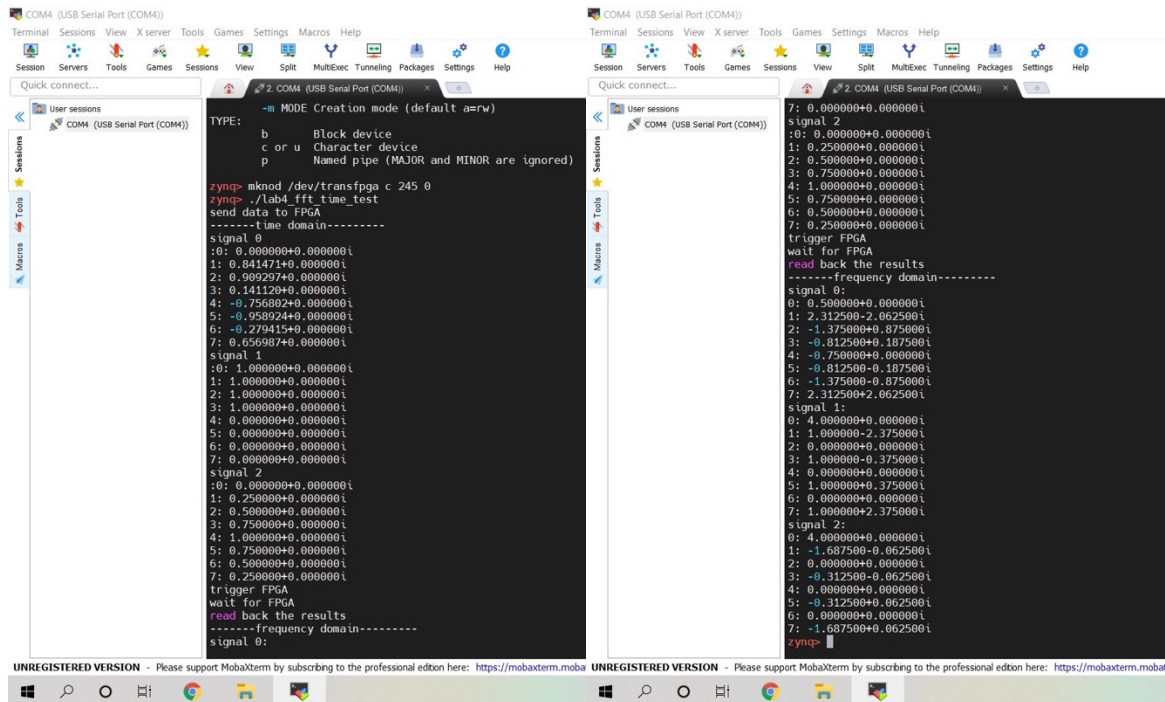


Figure 2. Simulation of the decimation-in-time FFT on a sine wave, with input and output sequences in reverse order.

DIT FPGA Outputs

Figure 3 shows the terminal outputs for the DIT-FFT, showing both the real sequences for signals 1, 2, and 3, and their DFTs. We can verify for each signal that the DFT errors fall within 0.1 for each signal when compared to Figure 1.



```
-m MODE Creation mode (default a=rw)
TYPE:
b      Block device
c or u Character device
p      Named pipe (MAJOR and MINOR are ignored)

zynq> mknod /dev/transfpga c 245 0
zynq> ./lab4_fft_time_test
send data to FPGA
-----time domain-----
signal 0:
0: 0.000000+0.000000i
1: 0.841471+0.000000i
2: 0.909297+0.000000i
3: 0.141120+0.000000i
4: -0.756802+0.000000i
5: -0.958924+0.000000i
6: -0.279415+0.000000i
7: 0.656987+0.000000i
signal 1:
0: 1.000000+0.000000i
1: 1.000000+0.000000i
2: 1.000000+0.000000i
3: 1.000000+0.000000i
4: 0.000000+0.000000i
5: 0.000000+0.000000i
6: 0.000000+0.000000i
7: 0.000000+0.000000i
signal 2:
0: 0.000000+0.000000i
1: 0.250000+0.000000i
2: 0.500000+0.000000i
3: 0.750000+0.000000i
4: 1.000000+0.000000i
5: 0.750000+0.000000i
6: 0.500000+0.000000i
7: 0.250000+0.000000i
trigger FPGA
wait for FPGA
read back the results
-----frequency domain-----
signal 0:
0: 0.000000+0.000000i
1: 2.312500-2.062500i
2: -1.375000+0.875000i
3: -0.812500+0.187500i
4: -0.750000+0.000000i
5: -0.812500-0.187500i
6: -1.375000-0.875000i
7: 2.312500+2.062500i
signal 1:
0: 4.000000+0.000000i
1: 1.000000-2.375000i
2: 0.000000+0.000000i
3: 1.000000-0.375000i
4: 0.000000+0.000000i
5: 1.000000+0.375000i
6: 0.000000+0.000000i
7: 1.000000+2.375000i
signal 2:
0: 4.000000+0.000000i
1: -1.687500-0.062500i
2: 0.000000+0.000000i
3: -0.312500-0.062500i
4: 0.000000+0.000000i
5: -0.312500+0.062500i
6: 0.000000+0.000000i
7: -1.687500+0.062500i
zynq>
```

Figure 3. Generated DFTs via DIT-FFT of a sine wave, square wave, and triangular wave. The real signals are shown on the left, and their DFTs are displayed on the right.

DIT Implementation Summary

Figure 4 shows the power, timing, and utilization summaries for the implementation of the DIT-FFT.

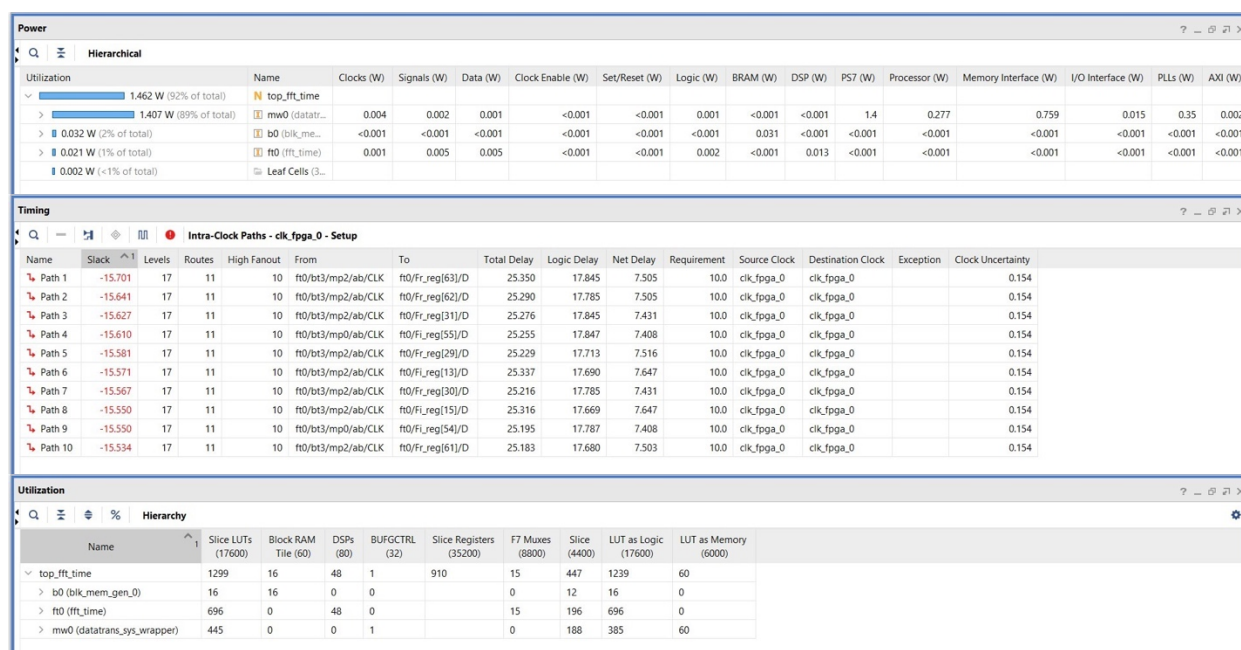


Figure 4. Power (top), timing (middle), and utilization (bottom) summaries for the DIT-FFT.

Decimation-in-Frequency (DIF) Simulation

Figure 5 shows the results for simulating the decimation-in-frequency (DIF) FFT on signal 1. We can confirm that the DFT errors are within 0.1 from the actual values displayed in Figure 1. There are some minor differences between the DIF and the DIT results (such as the index-6 real DFT value being -0.875 for the DIT, but -0.8125 for the DIF). The two methods largely generate similar, if not same, simulation outputs that fall within the desired 0.1 error.

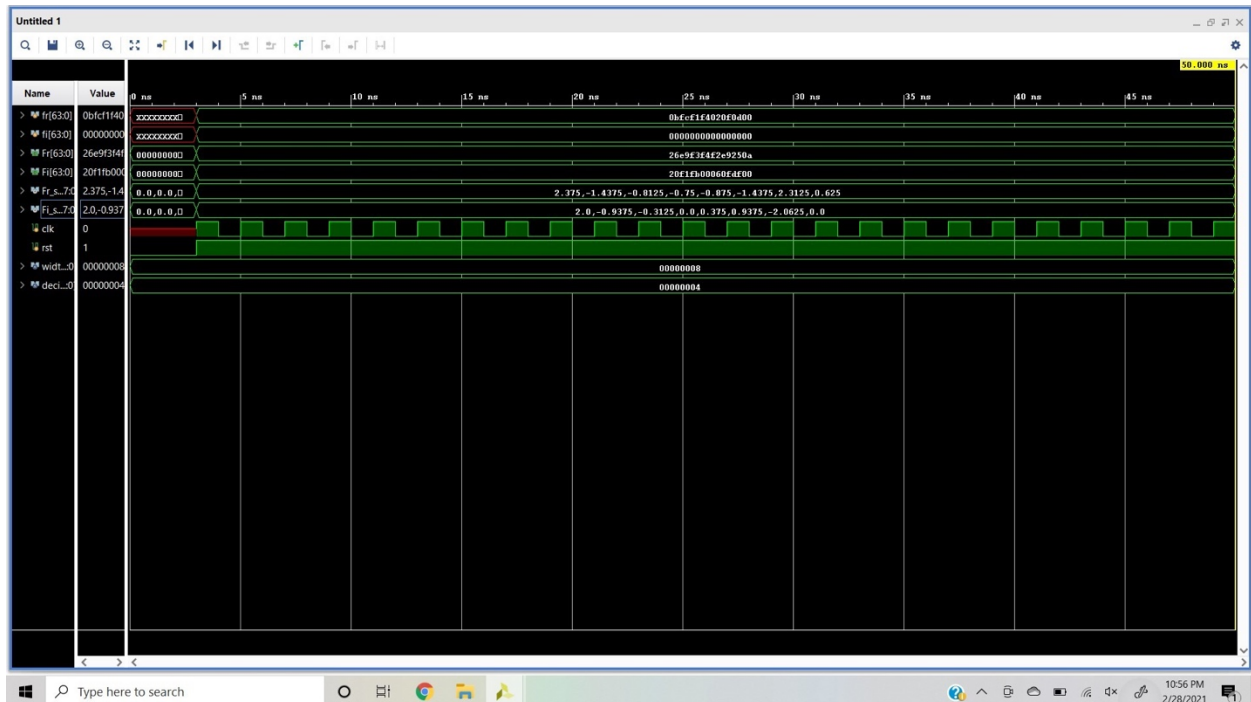


Figure 5. Simulation of the decimation-in-frequency FFT on a sine wave, with input and output sequences in reverse order.

DIF FPGA Outputs

Figure 6 shows the terminal outputs for the DIF-FFT, showing both the real sequences for signals 1,2, and 3, and their DFTs. We can verify for each signal that the DFT errors fall within 0.1 for each signal when compared to Figure 1.

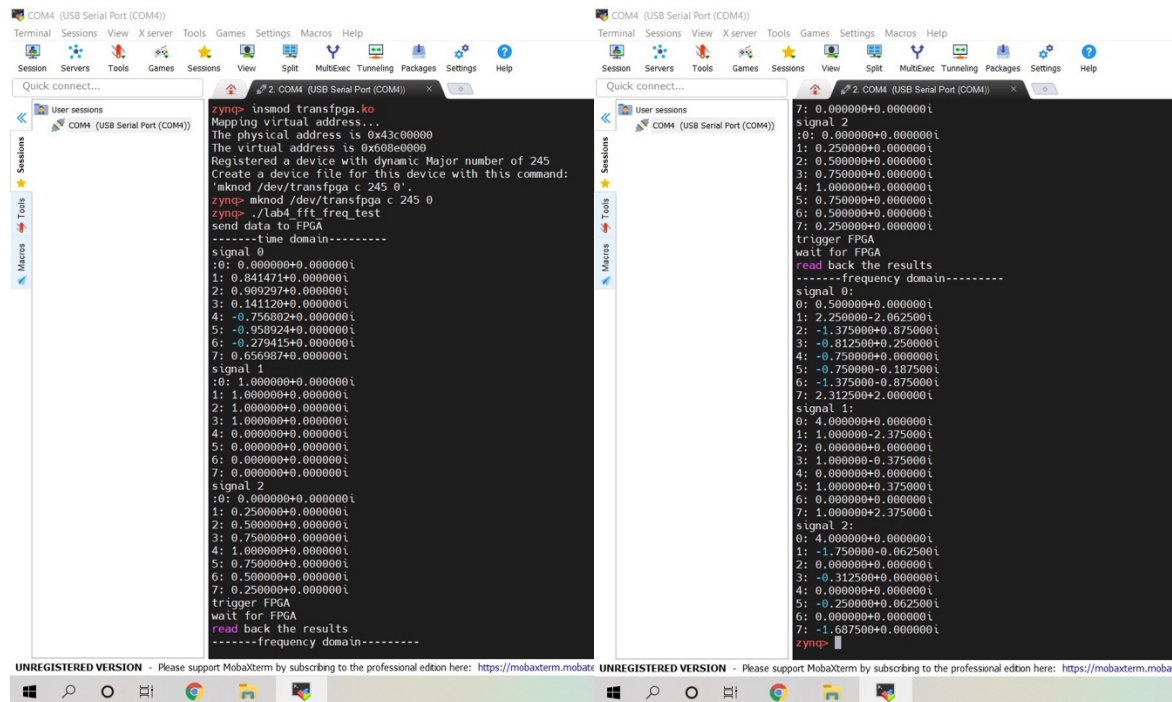


Figure 6. Generated DFTs via DIF-FFT of a sine wave, square wave, and triangular wave. The real signals are shown on the left, and their DFTs are displayed on the right.

DIF Implementation Summary

Figure 7 shows the power, timing, and utilization summaries for the implementation of the DIF-FFT.

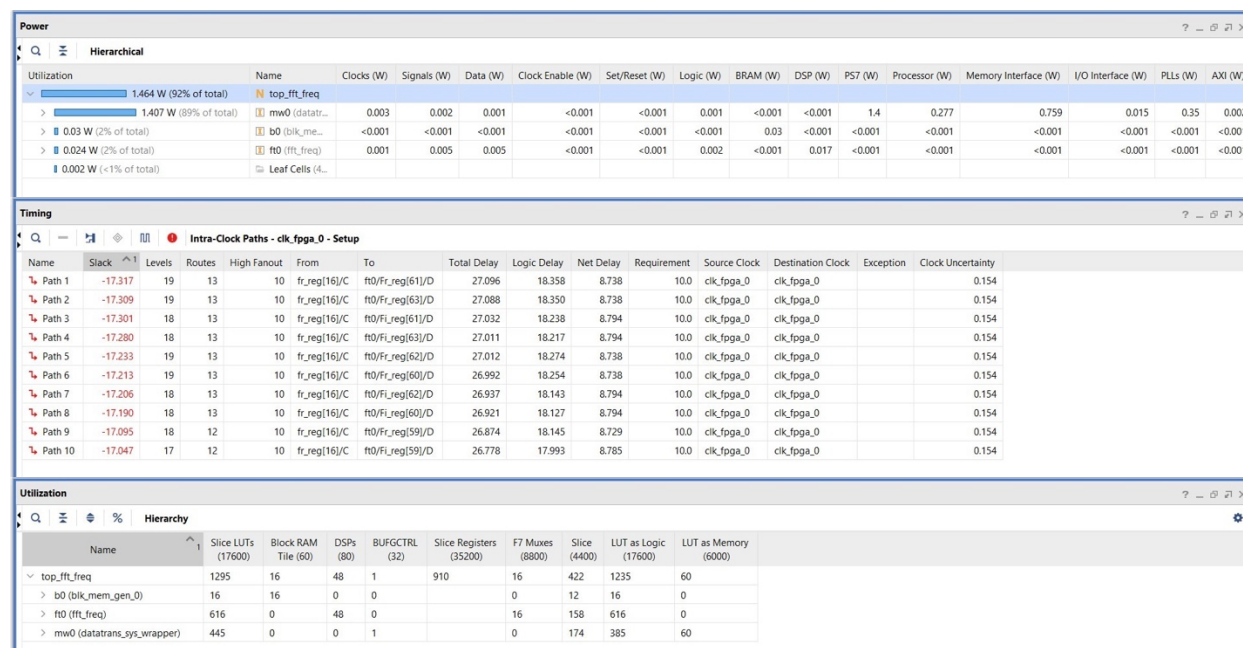


Figure 7. Power (top), timing (middle), and utilization (bottom) summaries for the DIF-FFT.

Pipelined DIF Simulation

We pipeline the DIF-FFT design by adding shift registers at the end of each stage. Figure 8 shows the results for simulating the pipelined DIF-FFT on signals 1,2, and 3 in that order. We can confirm that the DFT errors are within 0.1 from the actual values displayed in Figure 1. As expected, the pipelined and non-pipelined versions generate the same results, as seen in Figures 5 and 8. We can expect the difference to play out in the implementation summary.

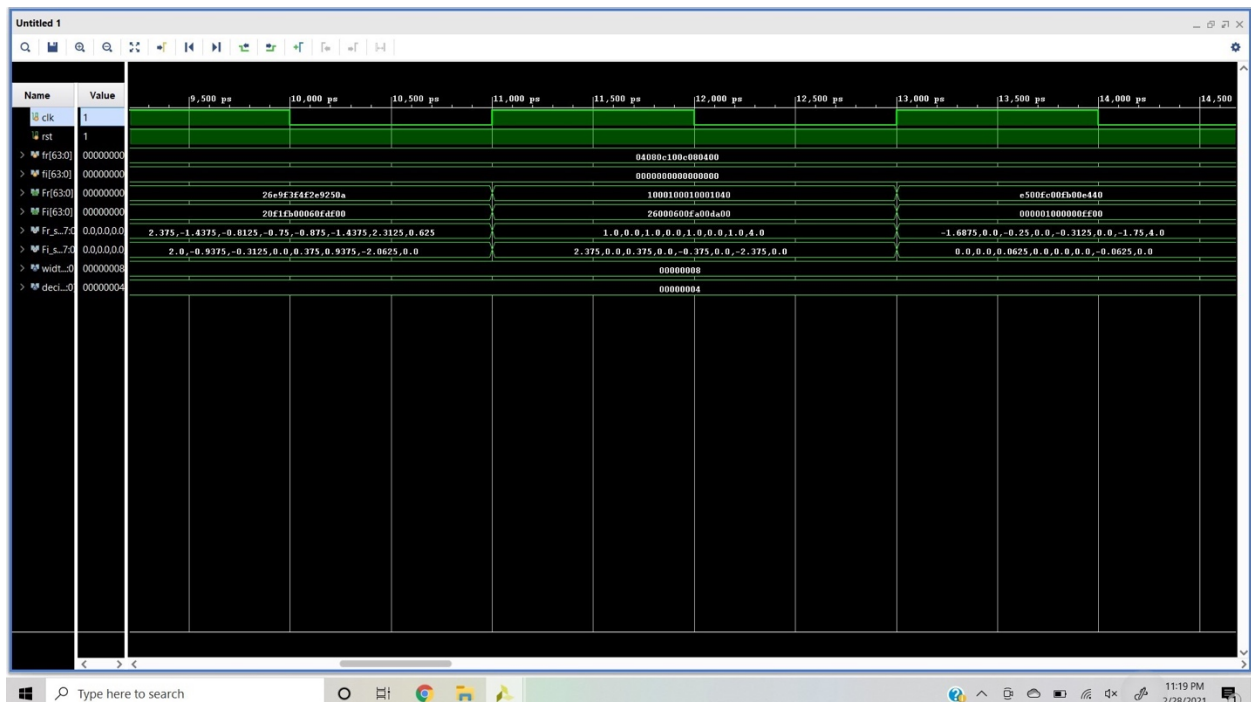


Figure 8. Simulation of the pipelined DIF-FFT on a sine wave, followed by square wave and triangular wave, with input and output sequences in reverse order.

Pipelined DIF FPGA Outputs

Figure 9 shows the terminal outputs for the pipelined DIF-FFT, showing both the real sequences for signals 1,2, and 3, and their DFTs. As expected, the values are the same as those displayed in Figure 6. We can verify for each signal that the DFT errors fall within 0.1 for each signal when compared to Figure 1.

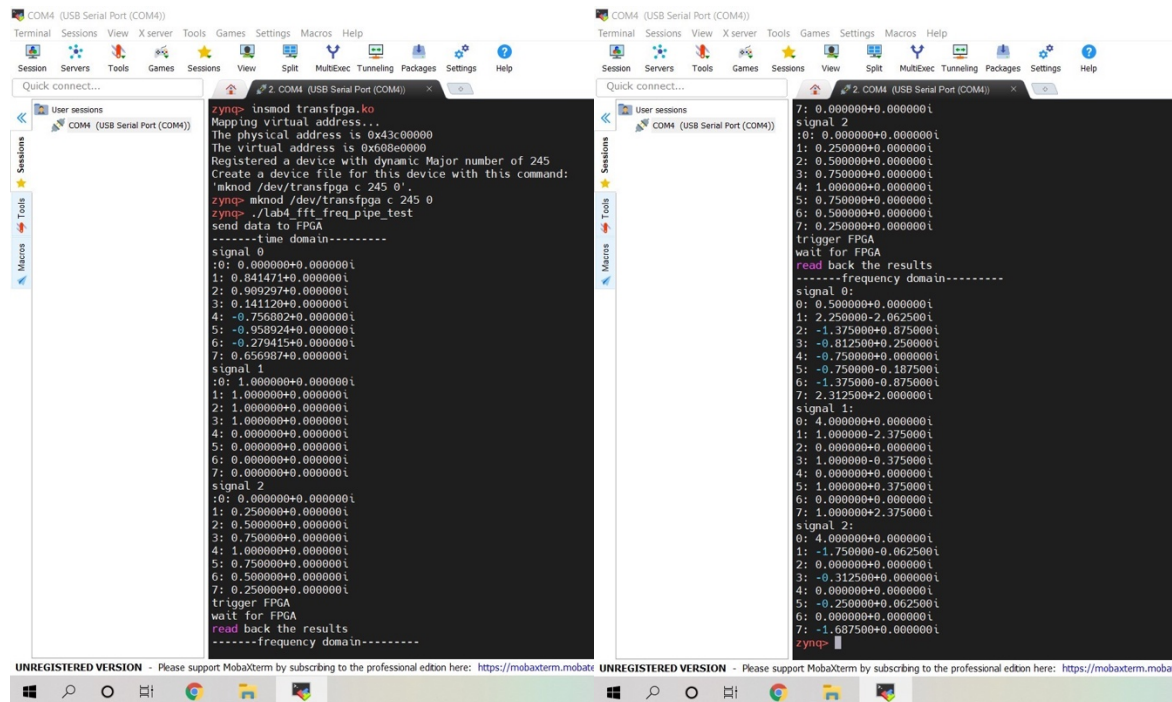


Figure 9. Generated DFTs via DIF-FFT of a sine wave, square wave, and triangular wave. The real signals are shown on the left, and their DFTs are displayed on the right.

Pipelined DIF Implementation Summary

Figure 10 shows the power, timing, and utilization summaries for the implementation of the pipelined DIF-FFT.

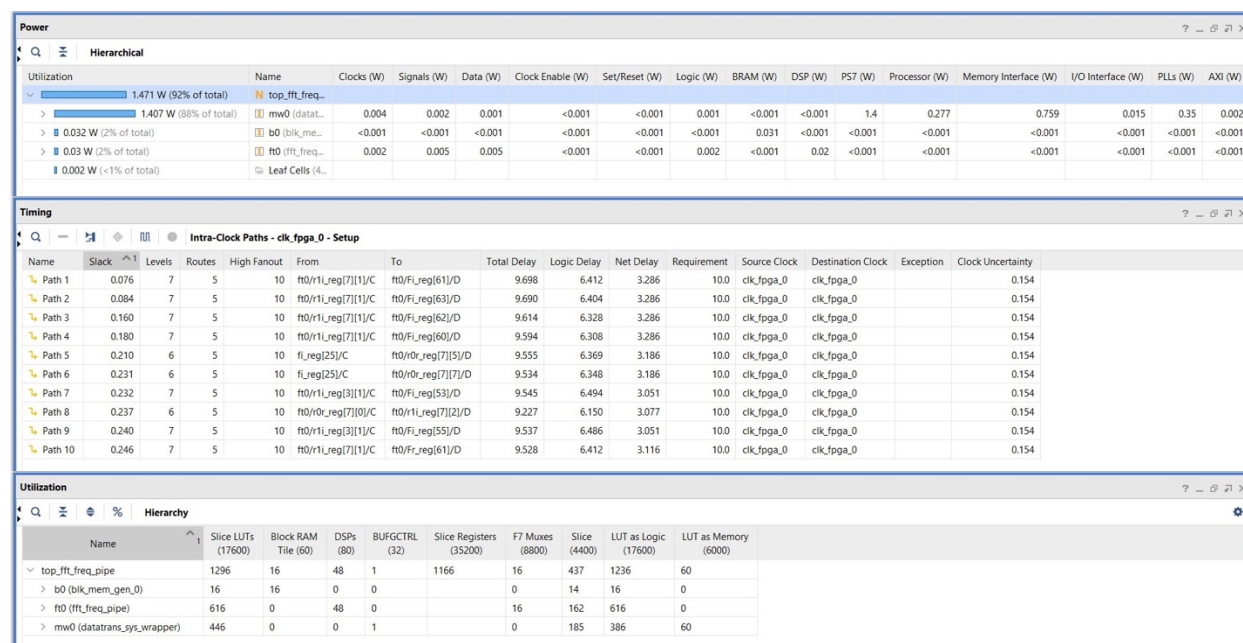


Figure 10. Power (top), timing (middle), and utilization (bottom) summaries for the pipelined DIF-FFT.

As we can see, when comparing Figures 4, 7, and 10, pipelining is done to reduce the clock period of the critical path such that the implemented design operates at a higher frequency. There is little difference in power, with DIT minimally edging out the others in least consumption. In terms of utilization, both versions of the DIF uses less slice LUTs than the DIT.

Questions

1. Please indicate one way to reduce the hardware utilization while designing FFT.

Instead of computing multiple butterflies during the same clock cycle as done in this lab, we can apply folding to the design. That is, we can utilize one butterfly per stage; that is, given three stages, we can use three butterfly modules and multiplexers to apply a delay for when each butterfly is computed.

We can expect a larger delay in generating outputs in comparison to the original design. For instance, for DIT, the second module would have to wait until the first module computes its first two butterflies before the second module even start computing its first butterfly. Similarly, the third module would have to wait until the second module finishes computing its first and third butterflies.

That said, when one clock cycle is one nanosecond, a larger delay is still fast relative to our human perception/experience of time.