Active Noise Cancellation Design on Hardware

Hsin-Ling Lu, Hsiang-Yang Fan, Daniel Yu, Yu-Fen Huang, Chieh-Jui Hsu
uniqname: hsinling, seanfan, dddaniel, yufen, jerrihsu

1 Introduction and Motivation

With voice assistants like Siri, Alexa, and so on seamlessly integrating into our daily lives, the way we interact with technology is changed and user experiences are significantly enhanced. Reducing the background noise to enhance the accuracy of speech recognition becomes a critical issue we need to concern. Therefore, our project focuses on removing the noise from a noisy audio signal in digital voice processors. First, we aim to develop a noise suppression technique that is effective and tailored to different kind of noise. This ensures that our noise reduction technique is practical for real-life applications.

By concentrating on the internal modification of voice processor algorithms, our approach provides the flexibility to adapt to the unique characteristics of different noise profiles. The final system contributes to a robustness and adaptability of digital voice processors in diverse environments.

2 Algorithm/Datapath Description

The datapath is shown in Figure 1. The input signal undergoes initial processing through a Hanning Window, sized at 64 with half-overlapping, preparing it for subsequent analysis. Following this, a Fast Fourier Transform (FFT) is applied to transform the data into the frequency domain.

We divide the noise into 64 sections, which is equal to the window size. The corresponding spectral magnitude from FFT is collected during non-speech activity to build up the initial noise filter. The estimated speech signal is then conceptualized as subtracting the magnitude of the estimated noise from the magnitude of the noisy speech signal. Introducing the effect of half-wave rectification biases down the magnitude spectrum at each frequency by the noise bias determined at that specific frequency.

The subsequent step involves utilizing the spectral subtraction filter, calculated in the previous stage, to attenuate the noise in conjunction with the noisy speech signal. This process effectively reduces the noise, providing additional noise attenuation and resulting in a more accurate estimate of the speech spectrum. Finally, to revert back to the time domain, an Inverse Fast Fourier Transform (IFFT) is applied, yielding the noise-reduced speech signal as the output.

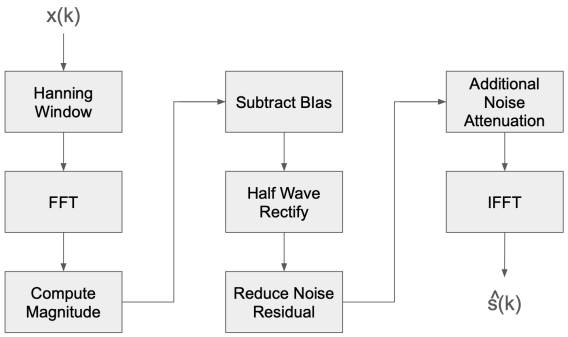


Figure 1: High Level Algorithm

3 VLSI Hardware Architecture

3.1 High-Level Architecture Overivew

In our project, we use (32, 24) fixed-point quantization in A-D converter and throughout the whole RTL implementation. The high-level architecture overiew is shown in Figure 2. As you can see, the input data undergoes a sequence of processing stages within our system. It begins by passing through a Hanning window, followed by FFT, CORDIC vectoring, Noise Cancellation, CORDIC rotation, and finally, IFFT.

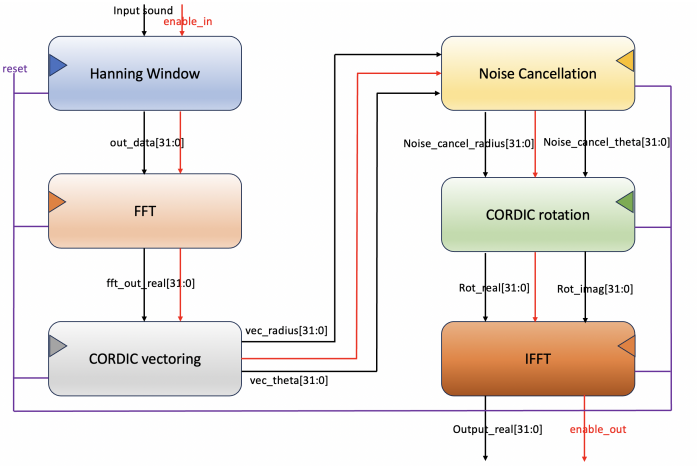


Figure 2: High Level Architecture

3.2 Hanning Window

First, we split the input into several windows with size 64 samples, and each window overlaps 32 samples. Then we apply a 64 point Hanning Filter on each window to reduce spectral leakage of the input. Spectral leakage occurs when the signal is not periodic within the window, it can introduce artifacts in the frequency domain and lead to the spreading of signal energy across multiple frequency bins. This spreading can make it challenging to accurately identify the true frequency components of the signal.

The Hanning window is a type of tapering or smoothing window that helps reduce spectral leakage. Figure 3 is the hanning window function with window size = 64. As we can see, it tapers the signal toward the edges of the window, gradually reducing the amplitude of the signal. The shape of the Hanning window is designed to minimize the impact of discontinuities at the edges of the window, which can contribute to spectral leakage. Therefore, applying hanning window can reduce the spectral leakage when converting to frequency domain.

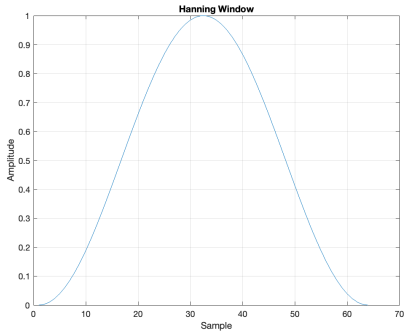


Figure 3: Hanning window with window size = 64.

3.3 64-Point FFT/IFFT

To have a more efficient utilization of area and to reduce the clock latency, we introduced the radix-2^2 single-path delay feedback (SDF) architecture to pipeline the 64-point FFT/IFFT into three stages. The radix-2^2 FFT algorithm has the same multiplicative complexity as radix-4 FFT algorithm but retains the butterfly structure of radix-2 algorithm. That makes the radix-2^2 SDF FFT algorithm the best choice for the VLSI implementation. The Figure 4 shows the data flow of SDF unit and Figure 5 shows the whole SDF architecture for 64-point FFT.

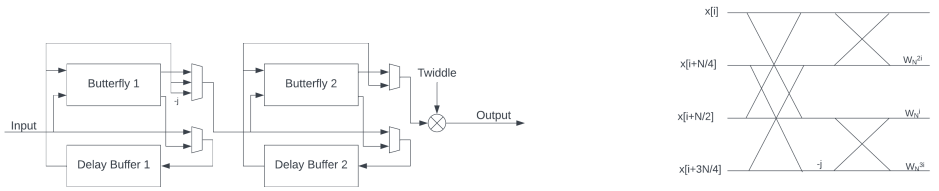


Figure 4: Radix-2^2 SDF unit

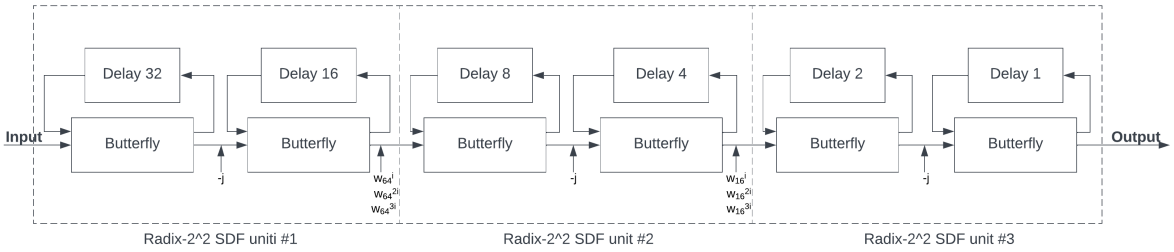


Figure 5: Radix-2^2 SDF architecture for 64-point FFT/IFFT

3.4 CORDIC

We implemented a 5-stage pipeline CORDIC in vectoring mode and another 5-stage pipeline CORDIC in rotation mode. There are 30 iterations in both of the CORDIC circuits so that we can achieve the result more precisely. The first CORDIC circuit uses the vectoring mode to turn the complex input signal into polar form. The signed imaginary part is used to choose whether it should do addition or subtraction during the operation.

The second CORDIC circuit uses the rotation mode to turn the polar form into the real and imaginary parts. The signed angle is used to choose whether it should be added or subtracted so that the final result can be performed correctly.

The Figure below is the VLSI implementation of the CORDIC circuit, the fixed point that our group implemented is (32, 24). Through the calculation, it performs 30 iterations in both the vectoring and rotation modes.

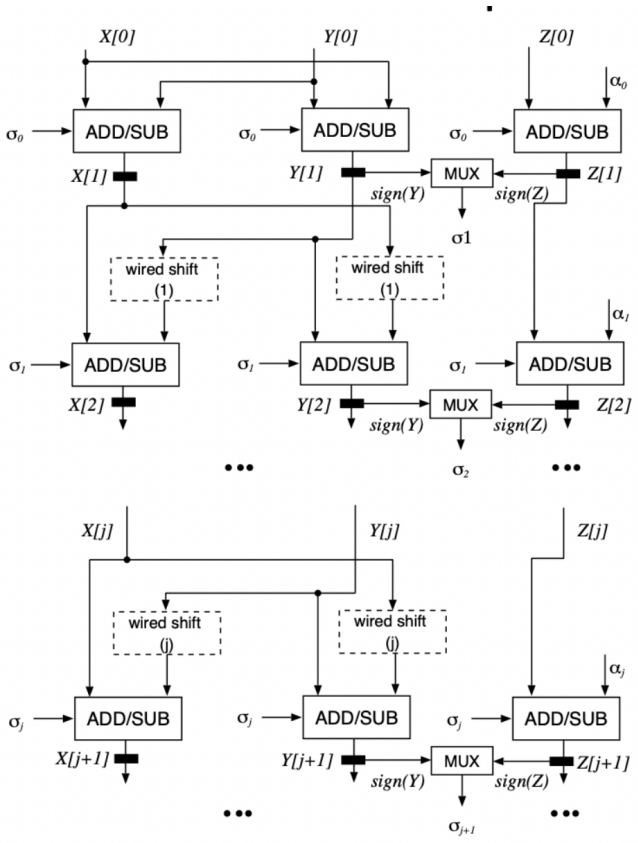


Figure 6: VLSI implementation of CORDIC

3.5 Noise Cancellation

In this architecture, we take the first 64 points of the window to be the corresponding noise threshold used in further calculation. For every 64 point of the input data afterwards, we subtract the chosen noise threshold from the input data then compare the subtraction result with the noise threshold. If the result is greater than the noise threshold, the input signal will be determined as a speech signal and the output of the noise cancellation architecture will stay the same as the original input. On the other hand, if the subtraction result is lesser than the noise threshold, the output of the noise cancellation architecture will be 0.

4 Simulation Results

4.1 Hanning Window

The Hanning window is implemented by constructing a Look-Up-Table (LUT) through Matlab, and subsequently convert these values into an RTL design.

4.2 64-Point FFT/IFFT

From the architecture above, our RTL implementation of the FFT consists of three SDF units and a sorting function. The SDF unit is structured with durations of 51 cycles, 15 cycles, and 5 cycles. These SDF units and the sorting function contribute to a cumulative delay of 136 cycles, achieving a throughput of 1 output per cycle. The architectural details are visually represented in Figure 7. The mean square error compared with Matlab's FFT output for real part and imaginary part are 4.86e-11 and 1.09e-10, respectively.

The IFFT implementation in our system is straightforward, leveraging our existing FFT module. By taking the negative conjugate of the output from the noise cancellation module and feeding it into our FFT module, we effectively achieve the IFFT operation. The addition of an output delay of this module introduces a cumulative delay of 137 cycles, maintaining a throughput of 1 output per cycle. This approach streamlines the IFFT process, capitalizing on the established FFT module to efficiently reconstruct the original signal with the desired characteristics. The architectural result shows in Figure 8. The mean square error compared with Matlab's IFFT output is 1.29e-14.

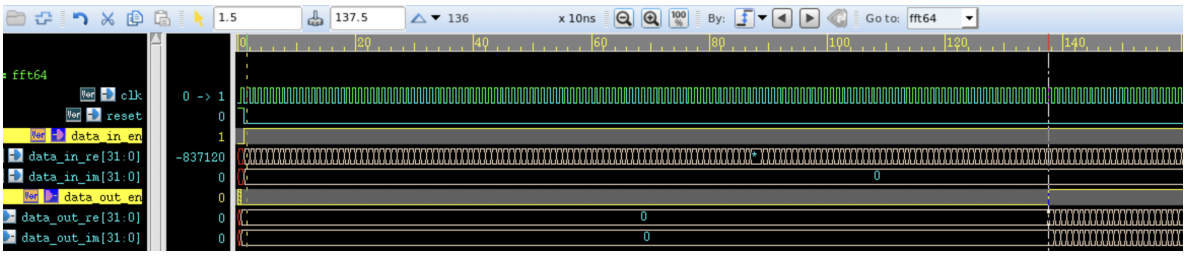


Figure 7: RTL simulation result of FFT

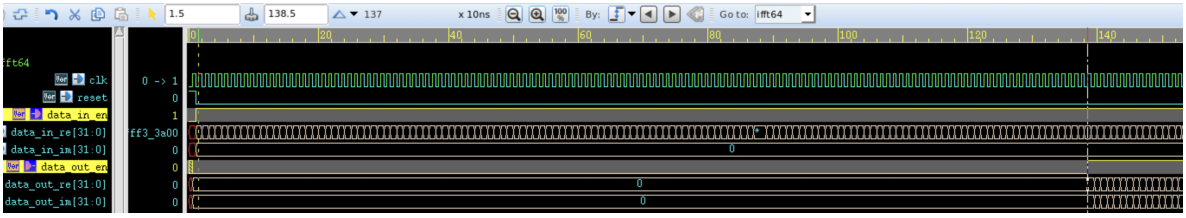


Figure 8: RTL simulation result of IFFT

4.3 CORDIC

Figure 9 shows that both of our vectoring mode CORDIC and rotation mode CORDIC take 5 cycles to finish calculation. To examine the correctness of the CORDIC outputs, we first used the input signal from HW3 to test our result. After that, we used a speech audio as the input of the CORDIC and compared the output with MatLab simulation results to check whether overflow occurred during calculation or not. Figure 10 and Figure 11 are the magnitude of the input signal and the rectangular form of the signal after doing noise cancellation respectively.

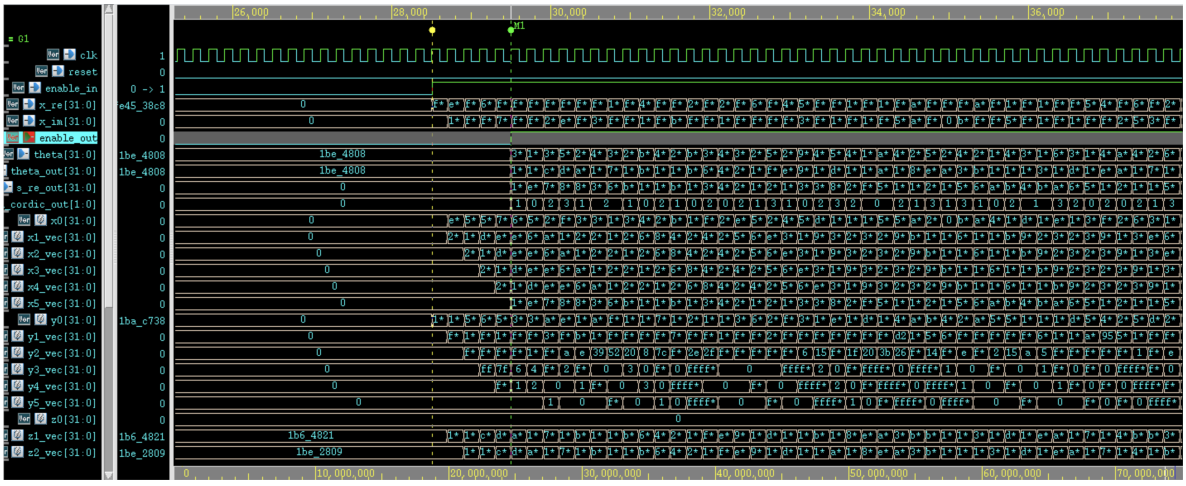


Figure 9: RTL simulation result of CORDIC

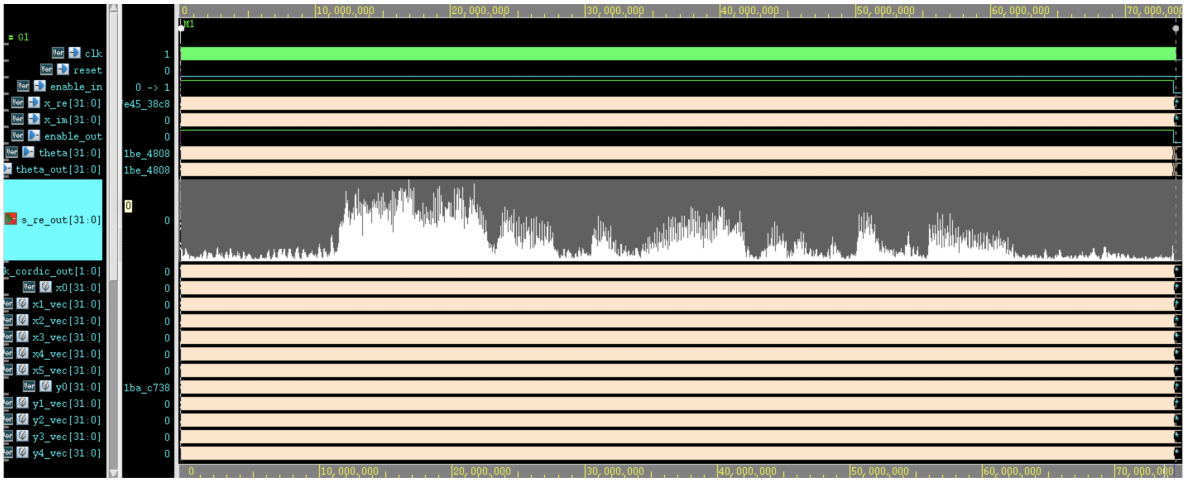


Figure 10: RTL simulation result of CORDIC in vectoring mode

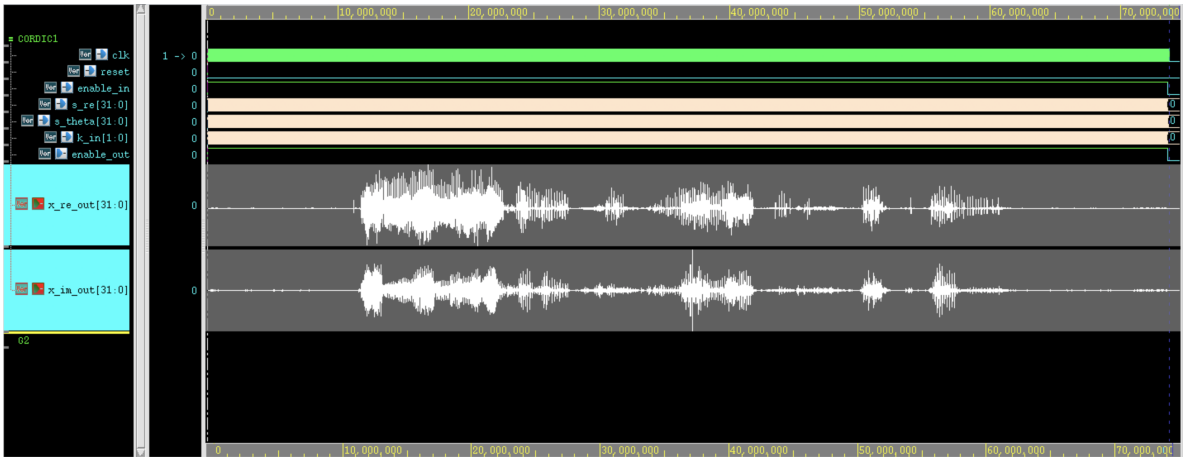


Figure 11: RTL simulation result of CORDIC in rotating mode

4.4 Noise Cancellation

The architecture takes 1 cycle to finish noise cancellation. We examined its correctness by comparing its output with MatLab simulation results. The results of RTL simulation are shown in Figure 12 and Figure 13.

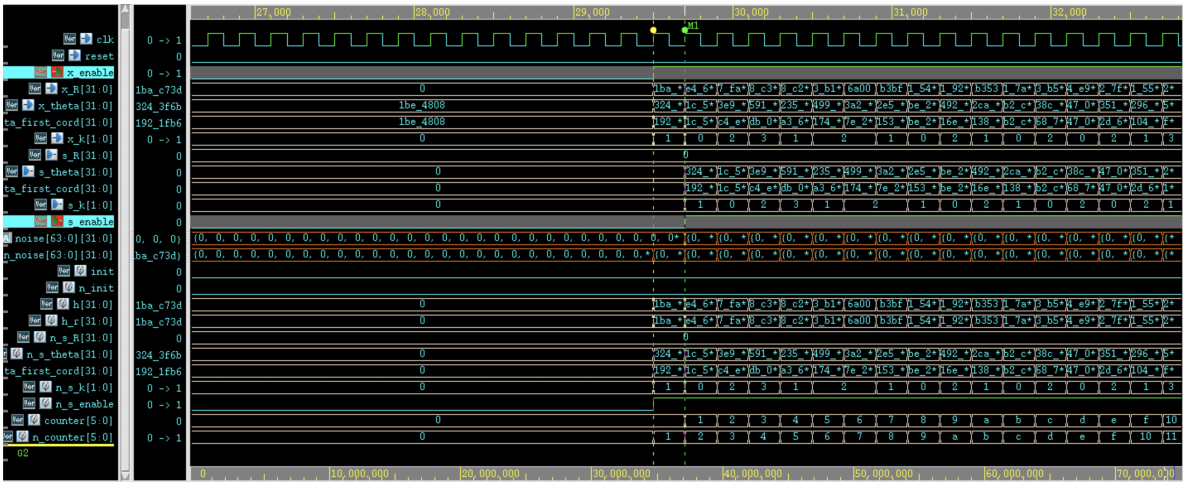


Figure 12: RTL simulation result of Noise Cancellation (digital waveform)

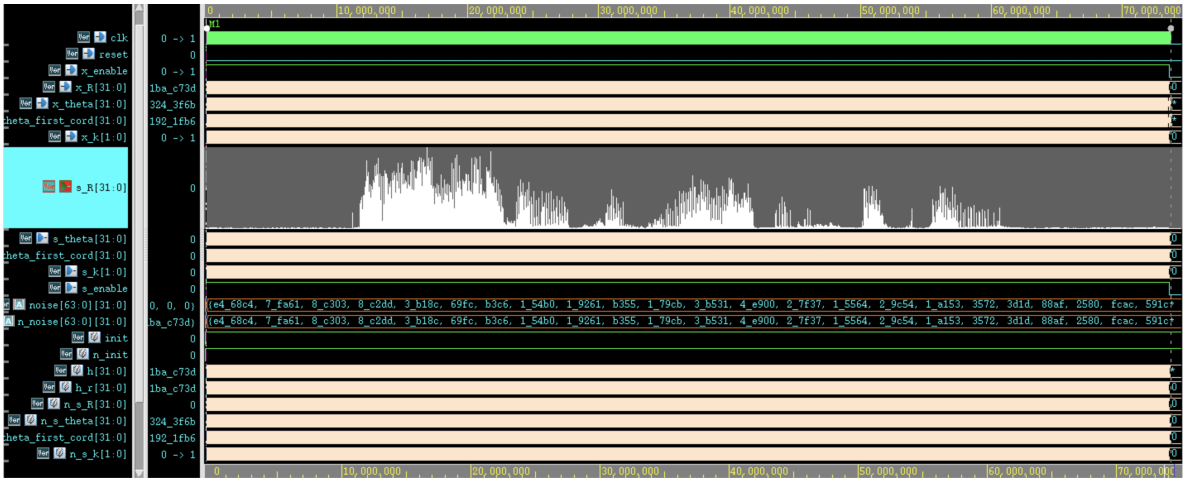


Figure 13: RTL simulation result of Noise Cancellation (analog waveform)

5 Synthesis Result

The minimum clock period, estimated power and estimated area of our design are 50ns, 351070nm^2, and 5.0044mW, respectively. The screenshot of those reports are shown in Figure 14 and Figure 15. Although we successfully synthesized our code, we were unable to meet hold time violations with the original synth.tcl and constraints.tcl. Hence, we have adjusted both files to relax the constraints. The constraints we used for synthesis are set_max_fanout = 6, clock_uncertainty = clock_period*0.001, clock_transition = 0.08, clock_latency = 0.1. For other constraints, please refer to our attached files.

clock core_clk (rise edge)	50.00
clock network delay (ideal)	0.10
clock uncertainty	-0.05
F2/T1/CORDIC/vec_stage1/R_7473/CLK (DFFX1_RVT)	0.00
library setup time	-0.05
data required time	50.00
data required time	50.00
data arrival time	-49.62
slack (MET)	0.39

Max timing report

clock core_clk (rise edge)	0.00
clock network delay (ideal)	0.10
clock uncertainty	0.05
F2/F1/SU1/R_6924/CLK (DFFX1_RVT)	0.00
library hold time	-0.01
data required time	0.14
data required time	0.14
data arrival time	-0.22
slack (MET)	0.09

Min timing report

Figure 14: Timing report

```
Number of ports:                        4160
Number of nets:                       107924
Number of cells:                       95110
Number of combinational cells:         61104
Number of sequential cells:            27480
Number of macros/black boxes:              0
Number of buf/inv:                      7760
Number of references:                      4

Combinational area:             166130.120424
Buf/Inv area:                    10255.472887
Noncombinational area:          184940.339994
Macro/Black Box area:                0.000000
Net Interconnect area:          190341.222853

Total cell area:                351070.460419
Total area:                     541411.683271
```

Area report

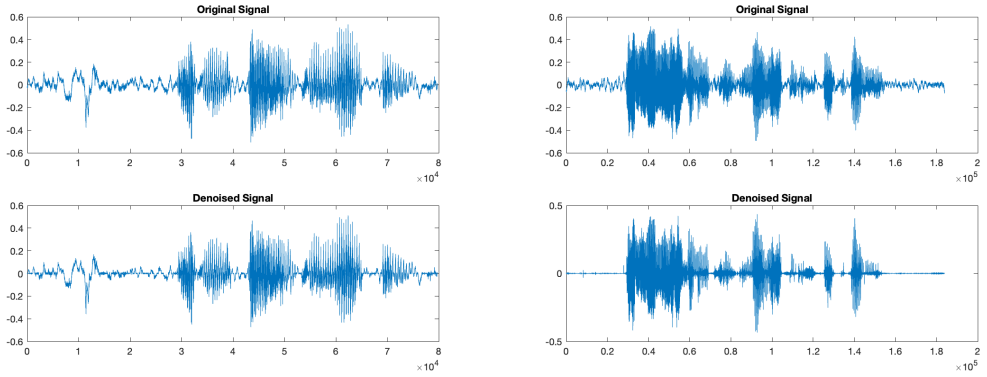
```
Attributes
----------
i - Including register clock pin internal power


  Cell Internal Power  =   4.8140 mW   (96%)
  Net Switching Power  = 190.3755 uW    (4%)
                         ---------
Total Dynamic Power    =   5.0044 mW  (100%)

Cell Leakage Power     =   9.0174 mW
```

Power report

Figure 15: Area and Power report

6 Conclusion

From the results shown in the previous section, we were able to cancel the background noise of the input audio with our RTL design. The overall throughput of our system is 1, and the total latency is 285 clock cycles. We also analyzed our results on Matlab which is shown in Figure 16. In addition, we tried different audio files to confirm its general applicability. Apparently, we can reduce the background noise to some extent from the original noisy audio signal.



p232_083.wav p232_188.wav

Figure 16: Matlab output

7 Contribution of Each Team Member

All team members particiate in final verification and report writing.
Hsin-Ling Lu: Implement FFT/IFFT module including verification, synthesis, and analysis.
Hsiang-Yang Fan: Implement CORDIC module including verification and synthesis.
Daniel Yu: Implement noise cancellation module including verification, synthesis and integration of all modules.
Yu-Fen Huang: Write Matlab files that produce input file for RTL and process ouput file from RTL. Implement FFT/IFFT module including verification and synthesis.
Chieh-Jui Hsu: Implement Hanning and CORDIC modules including verification and synthesis.

References

[1] Firdauzi, Anugerah, et al. "Design and implementation of real time noise cancellation system based on spectral subtraction method." Procedia Technology 11 (2013): 1003-1010.

[2] Boll, Steven. "Suppression of acoustic noise in speech using spectral subtraction." IEEE Transactions on acoustics, speech, and signal processing 27.2 (1979): 113-120.

[3] Algnabi, Yazan Samir, et al. "Novel architecture of pipeline Radix 2^2 SDF FFT Based on digit-slicing technique." 2012 10th IEEE International Conference on Semiconductor Electronics (ICSE). IEEE, 2012.

[4] Algnabi, Yazan Samir, et al. "FPGA Implementation of Pipeline Digit-Slicing Multiplier-Less Radix 2 Power of 2 DIF SDF Butterfly for Fast Fourier Transform Structure" (2011). European Conference on Antennas and Propagation (EUCAP2011). Pp. 4168-4172.