Real Time Self-Interference Cancellation for Wireless Transceivers

Daniel N Taylor

Electrical & Computer Engineering

Advisor: Arun Natarajan

Date: 9-27-2018

Abstract

With the need for higher spectral efficiency and data rates the rise of simultaneous transmit-and-receive (STAR) radios is becoming increasingly valuable. While many systems employ two separate channels for transmit and receive to double the data speed, this comes at the cost of doubling the RF band. This same performance could be attained with a full duplex system with half the bandwidth. RF historically, has been incapable of full duplex due to the tremendous amount of self-interference (SI) on the receiver, but due to advances in both analog and digital cancellation, full duplex is realizable.

This project focuses on the digital back end of self-interference cancellation following the analog front end for full duplex applications implemented in hardware. Pairing software coefficient computation through cross-correlation, scaling optimization, and interpolation, with high speed FPGA processing to develop real time self-interference cancellation. Utilizing software configurable firmware on a Xilinx FPGA and a 16-bit ADC/DAC card at 1GSamp/sec to reduce the SI by more than 50dB.

Keywords: Full Duplex Radio, Self-Interference Cancellation, FPGA, DSP, FMC, ADC, Wireless transceivers, Real time.

Acronyms

ADC	Analog-to-Digital Converter
AXI	Advanced eXtensible Interface protocol
AXI4-Stream (AXIS)	AXI for high-speed streaming data
AXI-4-Lite	AXI for simple memory-mapped communication
CID	Constellation ID
CPLD	Complex Programmable Logic Device
CPU	Central Processing Unit
DAC	Digital-to-Analog Converter
DAM	Direct Access Memory
DSP	Digital Signal Processing
EEPROM	Electrically Erasable Programmable Read-Only Memory
FMC	FPGA Mezzanine Card
FPGA	Field Programmable Gate Array
IP	Intellectual Property
PCIe	Peripheral Component Interconnect Express

This paper documents the components used, IP's designed, software interaction, and the results of a self-interference canceller implemented using a Xilinx Kintex UltraScale on a PCIe card (PC821) with a mezzanine ADC/DAC from Abaco Systems. The end goal is a device that is fully capable of simultaneously transmitting and receiving through a single antenna upon the same frequency and accounting for the resulting self-interference. This is known as a Full Duplex Radio. To accomplish this, a device must be capable of fully isolating and removing the output from the input and thus passing on only the desired signal. The system has 4 channels each having a sampling rate of 1GSamp/sec at a resolution of 16bits, the frequencies dealt with in this document will be primarily in the range of 5MHz-50MHz.

Hardware

PC821:

The PC821 is an FPGA card that contains a Kintex UltraScale device and several interfaces and modules that are readily accessible through software (See figure 1). The first interconnect is the PCIe edge connect for communication with the host PC, this is how images are uploaded to the FPGA as well as show all software Read/Writes are carried out. The board has two high speed mezzanine connections, the ADC/DAC mezzanine board resides in the first of these. The PC821 also has 8GB of DDR4 memory for direct memory access (DAM). This is used when storing large buffers of data from the ADCs, which is then passed to the host PC through the PCIe interconnect for coefficient calculation. The PC821 runs off a 300MHz reference clock from the

PC, which is then passed into the clock-rst IP, for division/multiplication. The diagram for these as well as other important elements for this project are shown below in figure 2.

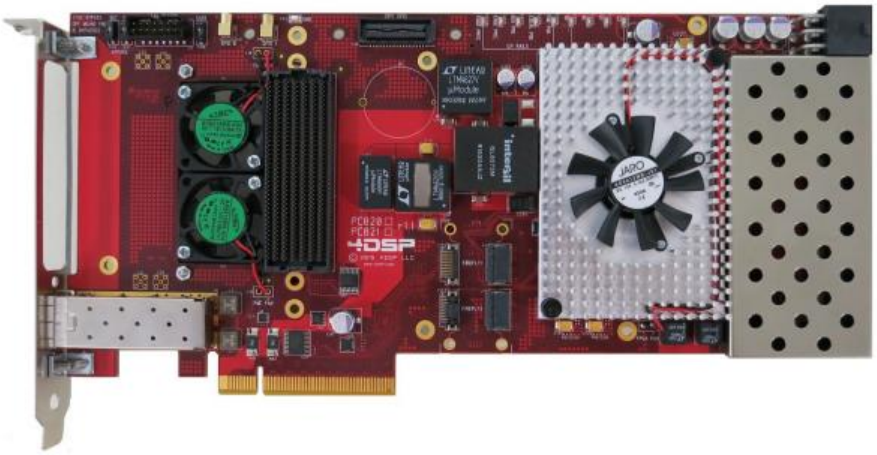

Figure 1: PC821 from Abaco Systems

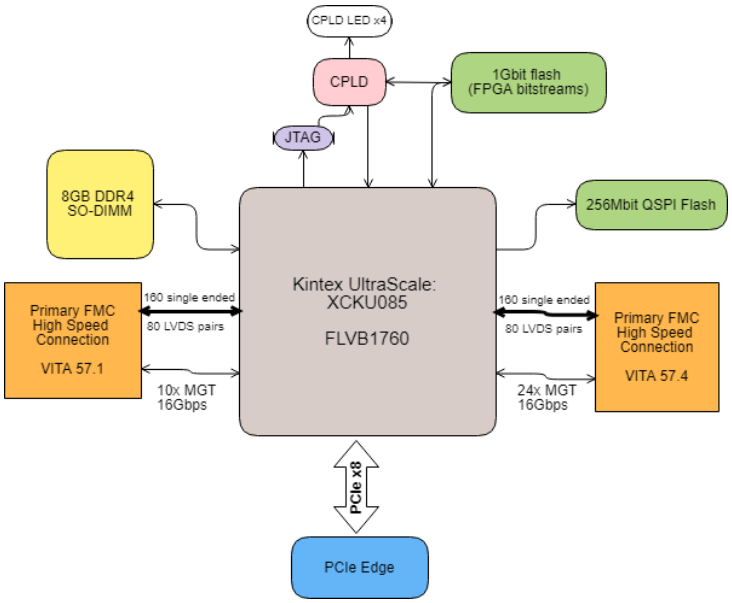

Figure 2: PC821 Diagram

FMC120:

The FMC120 is a 4 channel ADC/DAC FPGA Mezzanine Card (FMC). Each channel is capable of 16-bit resolution and simultaneously capturing/outputting at 1GSamp/sec. It can be driven

from the FPGA or from an external hardware trigger and clock. The ADC's within the FMC are Analog Devices ADS54J60 and the DACs are Texas Instruments DAC39J84.

Figure 3: FMC120

For high speed data rates the JESD204B converter interface is used within the FMC for communication between the FPGA and the ADCs/DACs. Each channel is devoted two data lines from the JESD204B to support the high sampling rates for this design. The CPLD, Voltage monitor, EEPROM, and Offset DAC within the FMC120 can all be configured thru I2C which is controlled through the AXI4-lite interface to the host. All ADCs and DACs are passed into the FMCs I/O through MMCX for high density.

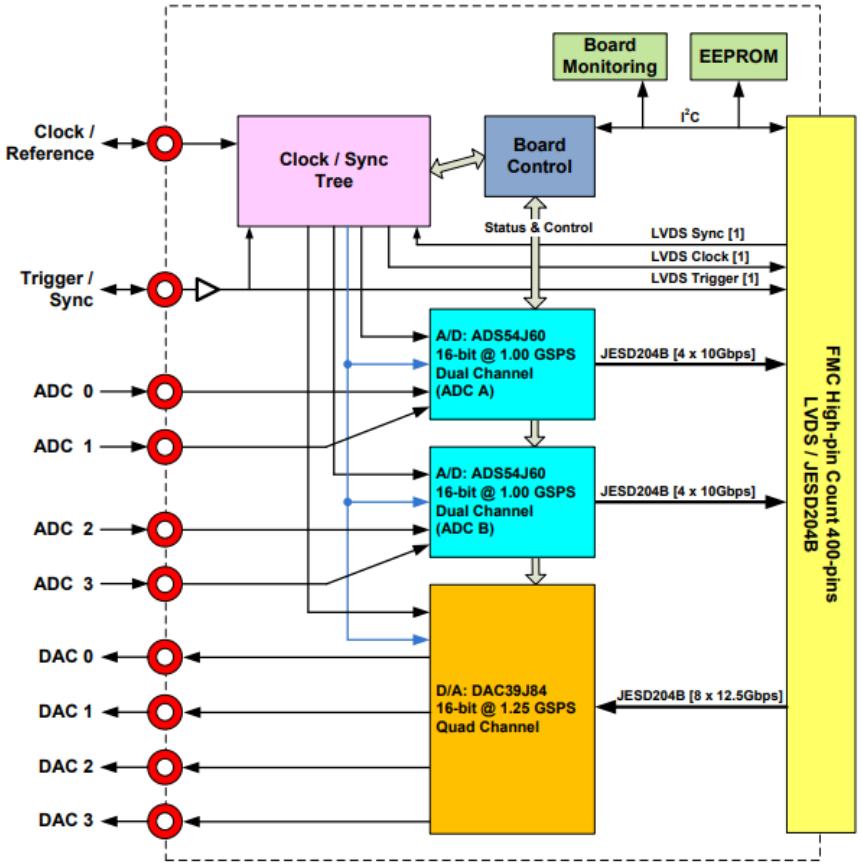

Figure 4: FMC120 Architecture

Interfaces

AXIS Stream:

The AXIS Stream interface is a protocol created by Xilinx for high speed communication between IP on the fabric. Although it contains many options for different applications, most of these are disregarded and only 4 lines are chosen. Additional lines can be used for notifying the end point of the last packet or specifics on packet. Since all the packets will be samples continuously, only a clock, data line, valid, and ready are needed.

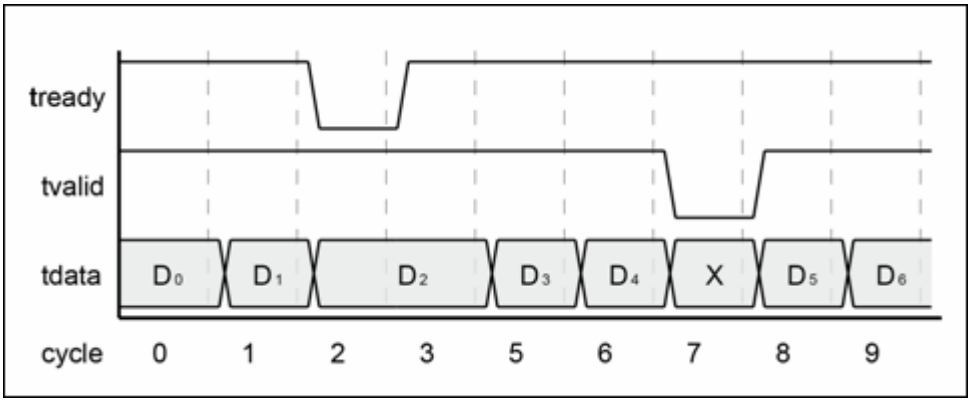

Figure 5: AXI Stream Protocol

There is a data line that in this project is 64-256bits depending on where in the design it is, a valid signal from the master to alert the slave that the data line is correct, and a ready signal from the slave to alert the master that the slave is grabbing the data. If at any point the slave is backed up and not ready for a fresh packet, it will pull the ready signal low, which the master sees and won't pass another packet until the ready signal is pulled high again. The slave in turn only pulls data from the tdata line if the valid signal is high. In this way, no addresses or specificity is needed for the interface and communication is only one-way from master to slave. The data widths are set in hardware between the master and slave and the clock speed is set by the FMC120 and the Host star.

AXI4-Lite:

The AXI4-Lite interface is a very robust and widely used communication protocol created by Xilinx for memory mapped communication between master and slave in hardware. All communication from the CPU to the fabric is done through the AXI4-Lite interface. This communication is much slower as it specifies the address, how the data will be sent, and is bi-directional. The master can write data to any addressed register or request data from any register.

For this to work there exists a ready, valid, data, and address line for both reading and writing data. In addition to this there is also read/write response lines from the slave.

Simple read and write diagrams for the protocol are shown below:

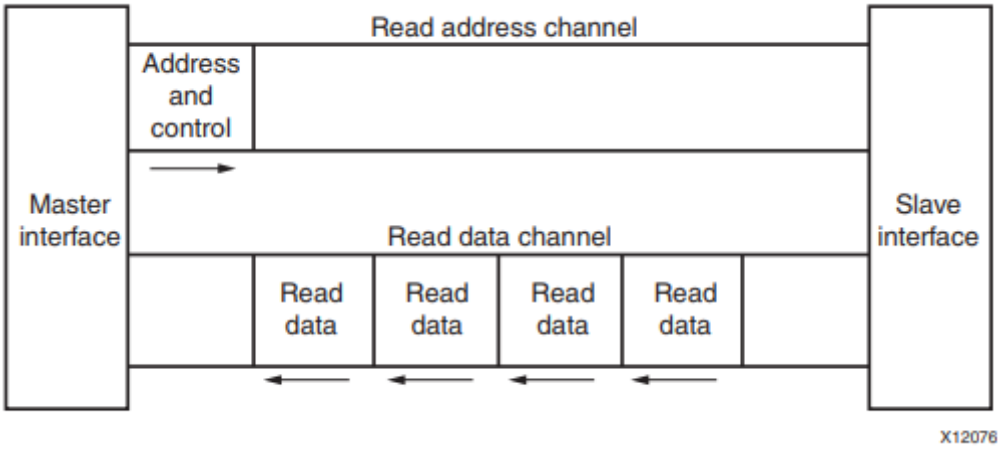

Figure 6: Channel Architecture of Reads

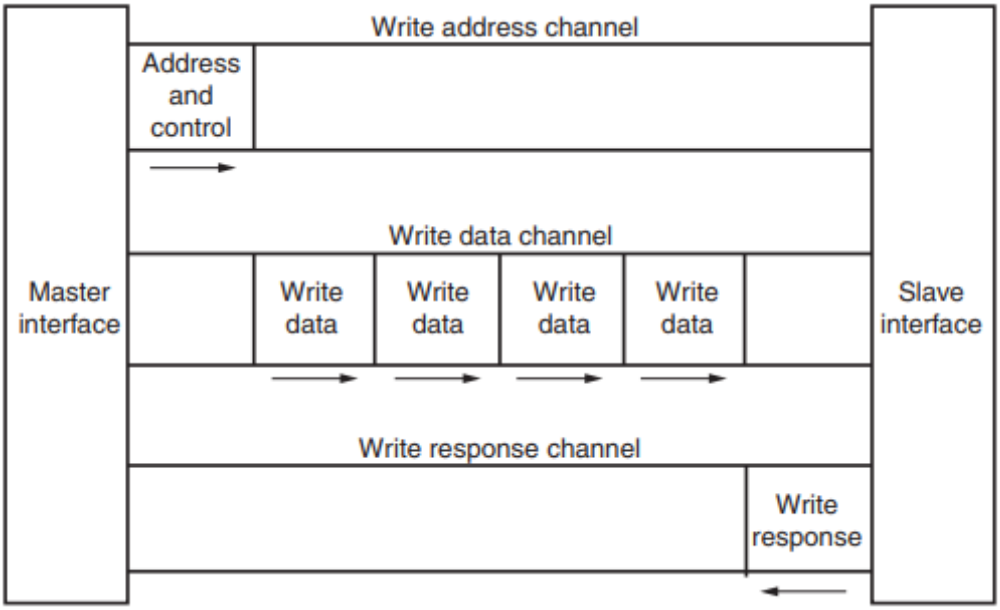

Figure 7: Channel Architecture of Writes

In this project the host IP sends out commands through the AXI4-lite interface and this is broadcasted to every IP in the design, the IP that contains the address sent responds with its ready signal and the handshake is established and data is transmitted.

PCIe Edge:

The Peripheral Component Interconnect Express (PCIe) is the high speed interface between the PC and the FPGA board. The PC821 receives its power, configuration for the fabric, and the 300MHz reference clock through PCIe. The PC821 is capable of PCIe x8, which has 8 data lanes for transmit and receive at 300MHz. Because of the speed limitation of this interface, the CPU is only able to evaluate data from buffers of data captured by hardware and not on a sample by sample basis. For this reason, large amounts of data are captured and analyzed by the CPU, which then writes the coefficients for sample by sample processing to the fabric through AXI4-lite. These values are read and applied to the AXIS streams for the highest throughput and lowest latency.

IP Blocks

The following Intellectual Property (IP) are implemented into the VHDL design and are each described in detail. The terminology used in this document refers to IPs as stars and the entire collection of stars as the constellation. These are the terms used by Abaco systems and thus is continued here. There are other stars such as data width converters and other misc IP used within the constellation, but these will not be covered as they only serve as interconnects between larger stars. The following are those that are key, configurable, and took the greater portion of the time for this project.

Host and CLKRST stars:

The host star serves as the start and origin of the commands, configuration, clock and reset signals to the system. The PCIe bus is connected here and the 300MHz reference clock is passed through the CLKRST star, which derives 5 slower clocks of 10, 50, 100, 125, and 200MHz. These clocks are passed to the host and from there distributed to the respective stars and internal memory access with in the host star. The 100MHz is the clock that is used for the command clock AXI4-lite interface. The host star has a single master command bus to read/write that is passed to a mux and both a master/slave pair of AXIS Stream interfaces for passing ADC data to the CPU and writing DAC data from the CPU.

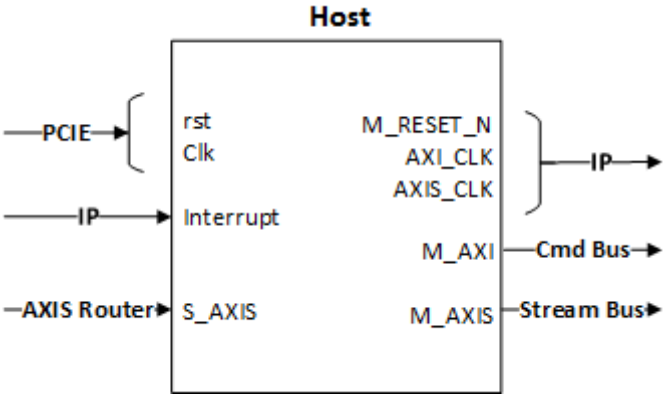

Figure 8: Host IP Interface Diagram

An interrupt signal is received from the I2C star in order to communicate with I2C hardware through AXI4-lite. This star contains many test features for debugging the status of the firmware and clocks as well as direct memory access (DMA). The DMA connections can be seen in the diagram below. The AXIS streams, both master and slave, are placed in a 256x256 FIFO which is then written/read to/from the main memory via the PCI Requester.

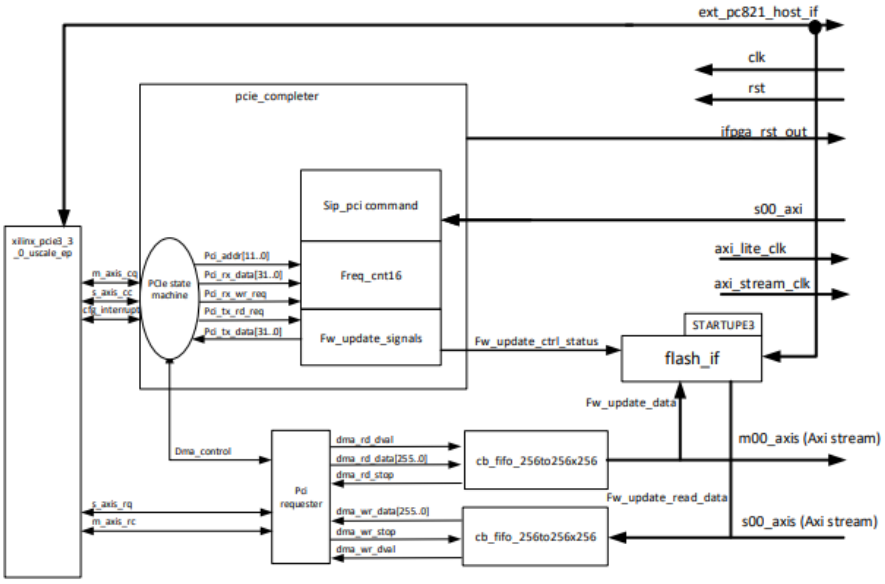

Figure 9: Host Internal Architecture

Command Mux:

The command mux simply passes the command line from the host star to however many stars need to receive it. These stars are all OR'ed together and passed back through the same interface to the host star. It should be noted that this mux is not entirely AXI4-lite compliant because if any star responds by asserting it's ready signal then a handshake could take place between the host and a star that does not have the address specified. For this reason, each star must check the address before asserting the ready signal. The command mux assumes this check is implemented for each star in the constellation and thus by simply OR'ing the responses together there should be no issue as only one star will ever respond.

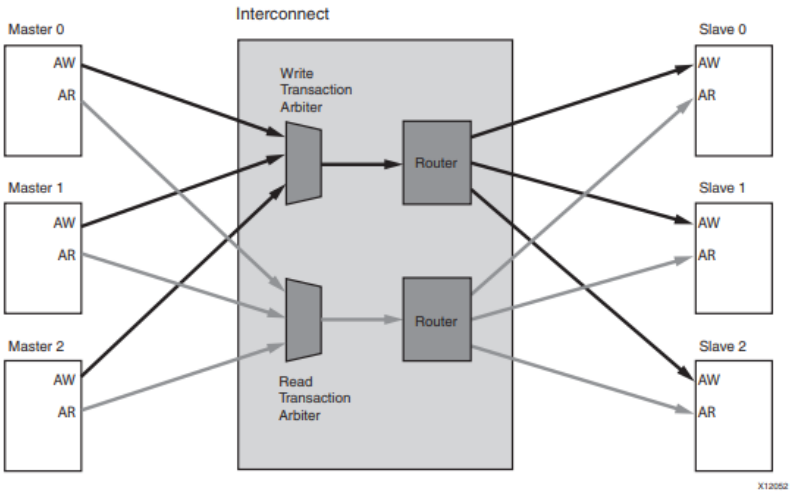

Figure 10: AXI4-Lite Command Mux Concept

FMC120:

This is the first host configurable star seen so far. The command bus comes from the command mux and the clock/rst signals from the host star. As the FMC120 star is the origin of all the ADC data, it passes an AXIS clock, reset, and trigger associated with the streams to the stars following. The output trigger needed so that capture stars can be triggered simultaneously to grab the sampled data in the same clock cycle whenever the FMC120 is ready with samples. This is a configurable software command and used when gathering the buffers of sampled data to send to the CPU.

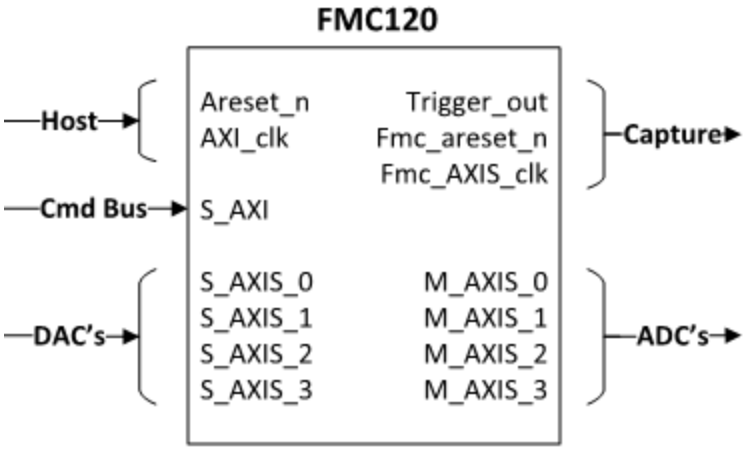

Figure 11: FMC120 IP Interface Diagram

The eight AXIS streams are the ADC/DAC connections. Each has a data width of 64bits and contains four samples of 16-bits. The AXIS stream clock has its source here and runs at 250MHz, so four samples are ready at 250MHz, which results in 1GSamp/sec.

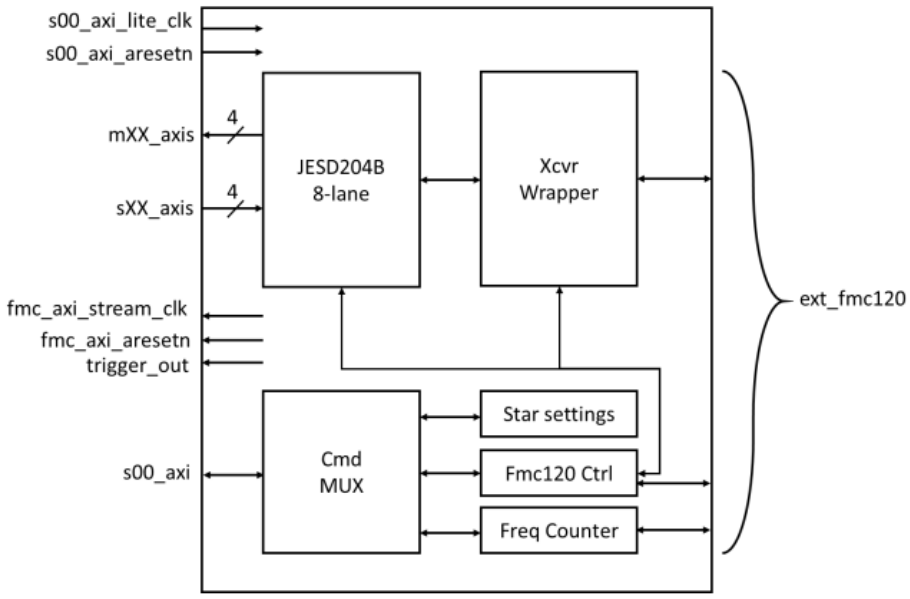

Figure 12: FMC120 IP Internal Architecture

The JESD204B converter is the interface of the streams and the star settings, control, and tests are all accessible via the AXI4-lite interface. The ext_fmc120 connects to the physical I/O's of the FMC.

AXIS Router:

Two AXIS Routers are used in this design. One for 64bit AXIS streams and another for 256bit streams. The router has an AXI4-lite interface for configuration. This can forward any input to any output(s). Any output cannot have multiple inputs, but any input can have multiple outputs. The ADC's and DAC's are tied to the 64bit router and can be configured to be sent to either the DSP block for hardware operation with the calculated coefficients or the capture block to be

stored and sent to the CPU. The DSP's outputs are tied to the input of the router so that they can also be configured to forward to the DAC's.

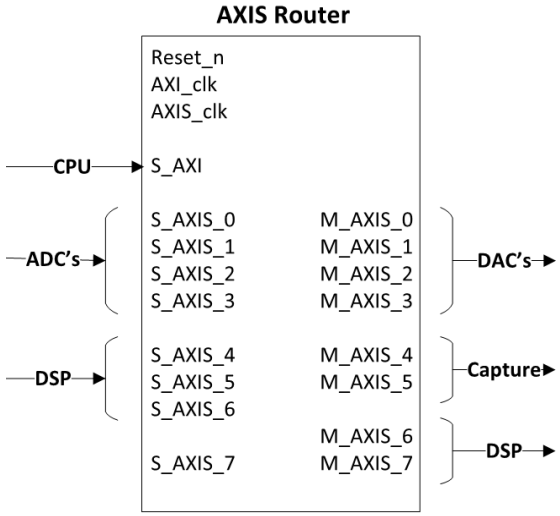

Figure 13: AXIS Router Interface Diagram

The second AXIS Router receives larger packets of samples from the capture star and can be configured to pass these buffers to the output or to the CPU through the host star.

Capture:

Each capture star receives on AXIS stream from an ADC through the AXIS router once the trigger_in is set from the FMC120 star. This stream is then passed into a 64bit-256bit convertor and these are passed into a configurable buffer that stores 256-32768k samples that can then be passed to the CPU.

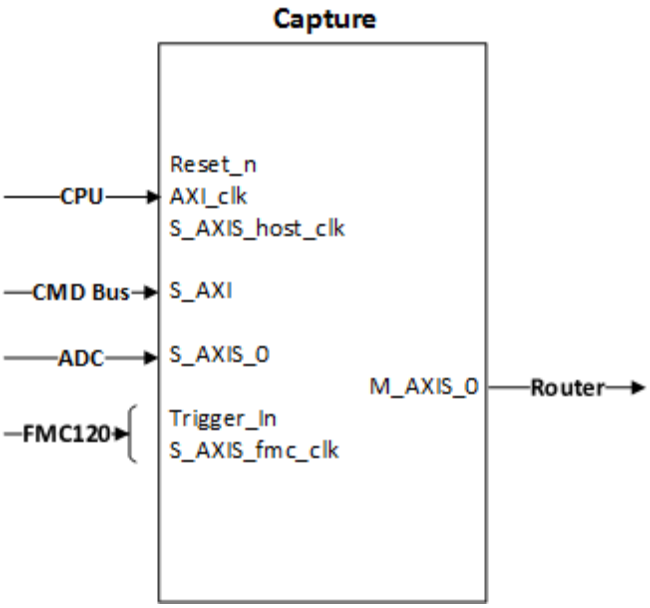

Figure 14: Capture IP Interface Diagram

The AXI4-lite clock is received from the host for the command line, but there are two AXIS stream clocks. One is from the fmc120 for the inputs and the second is from the host IP and is the clock for the buffer output.

DSP:

Since the FMC120 star passes out 4 samples from the ADC's on each clock cycle the DSP must have parallel processing to compute 4 output samples for every 4 sample packet at 250MHz. The DSP IP receives its clock, reset, and command signals just as the capture star, but has two stream inputs for the ideal and non-ideal streams to pass in and a single stream output for the residual. The other master streams are simply for debugging purposes and pass out the results along the DSP processing.

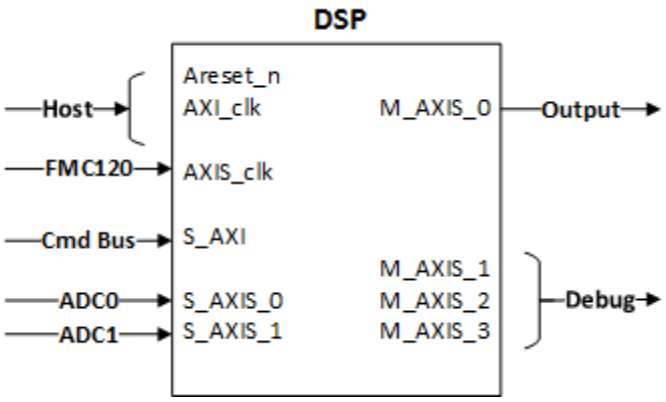

Figure 15: DSP IP Interface Diagram

The DSP star has the following values that can be written to from the host. The Control can be read and return the current status of computation as well as written to in order to start/stop/idle the process. The shift index is how much to delay the ideal stream in order to align it to the non-ideal input. The scaler value scales the non-ideal input in order to best match the ideal and minimize the residual. The Interpolation weights set the hardware weights of the current sample versus the next or previous sample. Setting Weight equal 1 would essentially ignore the weighting, while setting it to 0.5 would compute the current output sample as ½(Current sample) + ½(Next sample).

Content	Memory Offset
Control	0x00
Shift Index	0x10
Scaler	0x20
Interpolation Weight	0x30

Table 1: DSP Memory Mapping

The below diagram shows the interaction of the CPU to DSP throughout the DSP hardware. The ideal and non-ideal samples are received from the FMC120 through the router and the 4-sample packets for ideal are shifted and weighted, while the non-ideal are scaled, then the non-ideal signal are subtracted from the ideal signal.

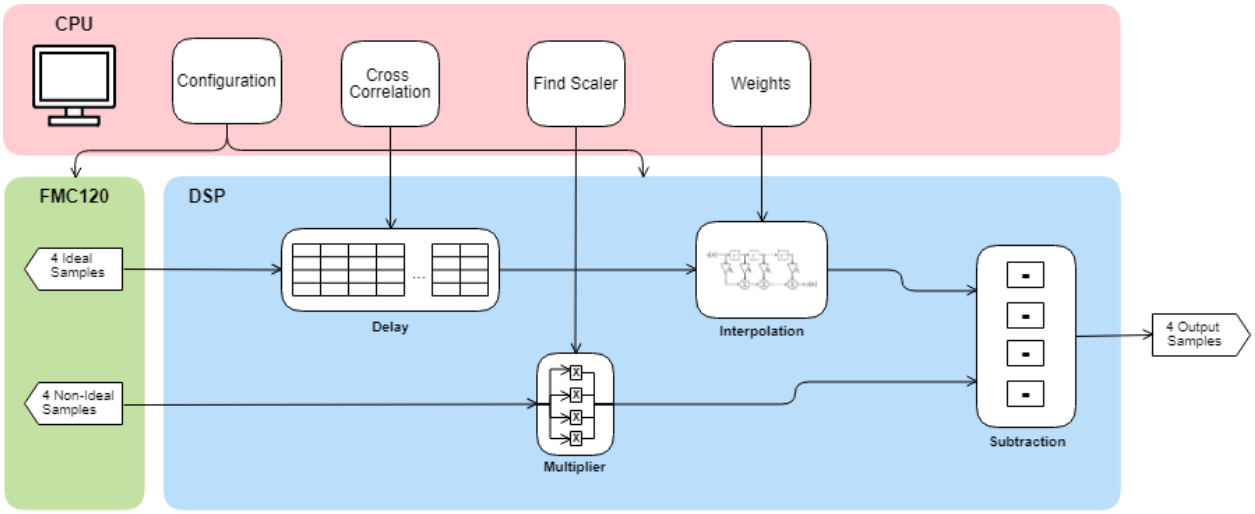

Figure 16: DSP Interfacing With Externals

The above diagram is accomplished with a latency of 7-9 clock cycles (depending on amount of precision desired) and an interval (throughput) of 1 clock cycle. The clock estimate is 3.49ns, which is within the 4ns required time with some room allowable for clock drift. The end result would be 4 samples processed every clock cycle with a 7-9 clock cycle delay from when an input is seen at the output.

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	4.00	3.49	0.50

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
9	9	1	1	function

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	4.00	3.49	0.50

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
7	7	1	1	function

Table 2: DSP Estimated Timing for 1/32 Precision (left) and 1/16 Precision (right)

The summary of the tasks per clock cycle can be seen below:

DSP Clock Cycle Pipeline Summary									
CC0	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9
Scaler & Shift Read									
Ideal & Non-Ideal AXIS Read									
	Shift Index Logic								
		Branch 0: Non-Ideal Scaling & Shifting For Precision							
		Branch 1: Non-Ideal Scaling & Shifting For Precision							
		Branch 2: Non-Ideal Scaling & Shifting For Precision							
		Branch 3: Non-Ideal Scaling & Shifting For Precision							
		Shift Index Adjust							
			Ideal Shift Register						
			Delay Out 0 Assign						
			Delay Out 1 Assign						
			Delay Out 2 Assign						
			Delay Out 3 Assign						
				Delay Out 0: Weighting (Multiply, Sum, and Shift)					
				Delay Out 1: Weighting (Multiply, Sum, and Shift)					
				Delay Out 2: Weighting (Multiply, Sum, and Shift)					
				Delay Out 3: Weighting (Multiply, Sum, and Shift)					
								Sample 0: Subtract	
								Sample 1: Subtract	
								Sample 2: Subtract	
								Sample 3: Subtract	
								Output AXIS Writes	

Figure 17: DSP Pipeline

Initially the ADC samples must be read from the inputs via the AXIS protocol, then four non-ideal samples are scaled by the value computed by software. To avoid floating point arithmetic, which would add a much larger number of clock cycles, the decimal is multiplied by 1024 such that sufficient digits of precision are gained and the result can simply be shifted to restore it to the appropriate scale. Then the value of the shift index sets the way that hardware interacts with the shift register. Since four samples are shifted into the register in a single clock cycle, four are also shifted out. The shift index will likely not be a multiple of four and conditional logic is needed for the delay out assignments. These are then weighted with 1/16 or 1/32 precision with the next or previous sample. This takes a multiplication, sum with the next or previous sample, and then a shift for division. These are subtracted from the scaled non-ideal samples in clock cycle 8, and the resulting output and debugging outputs are written to the DAC via the AXIS stream.

I2C:

This star serves as a bridge from AXI4-lite to I2C hardware. The PC821 has a local I2C bus that can communicate with both FMCs and on-board ICs such as the CPLD. On board measurements are made on startup of every software application that measures the status of test points, clock frequencies, offsets, and calibrations.

Constellation ID:

The Constellation ID (CID) IP stores the addresses for each IP along with the capabilities of each star, data path lines, and the constellation information. This IP is purely for the ease of software. Each IP can be given a global unit number and this can be used in software instead of each stars starting address and the offset from there. Instead, software can right to the offset 0x30 of global unit 3. This makes the software very clean and software can check the firmware ID and carry out different initialization routines depending on what hardware is currently implemented on the fabric.

Software

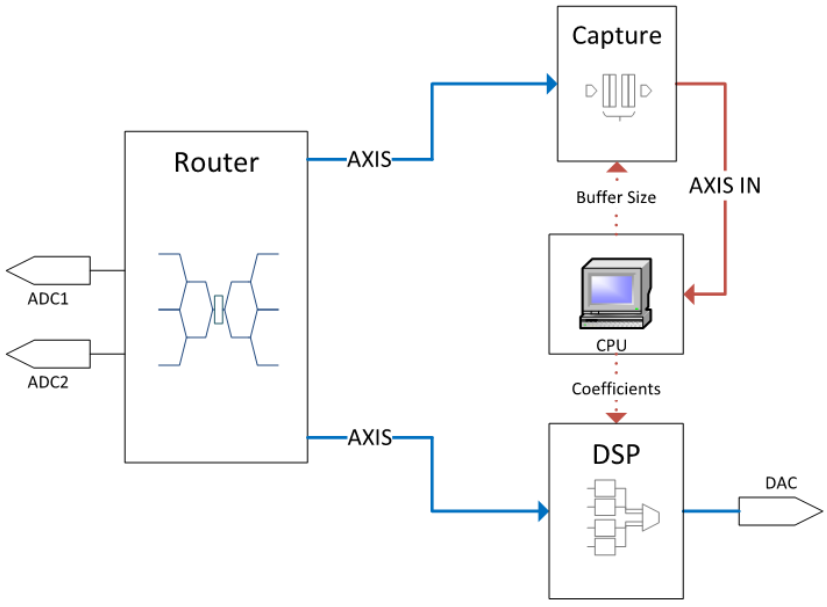

Figure 18: Software Interfacing with IP

Initialization & Configuration: Every star in the constellation must be configured and once communication is made with the CID IP, then the addresses for each IP are not needed, just the offsets. The FMC120 configuration is complex and requires a good deal of knowledge of the LMK04828 clocking tree chip, DAC39J84 for the four DACs, and the ADS54J60 for the four ADCs, all of which communicates over I2C protocol. The clocking tree has three reference clocks that can all be optimized for the sampling frequency. Each of these clocks has multipliers, dividers, and prescalers across two PLLs. The sampling frequency is limited both by the JESD204B interface IC, which has a minimum data speed of 250MHz, and the ADC itself, which has a maximum rate of 1GHz.

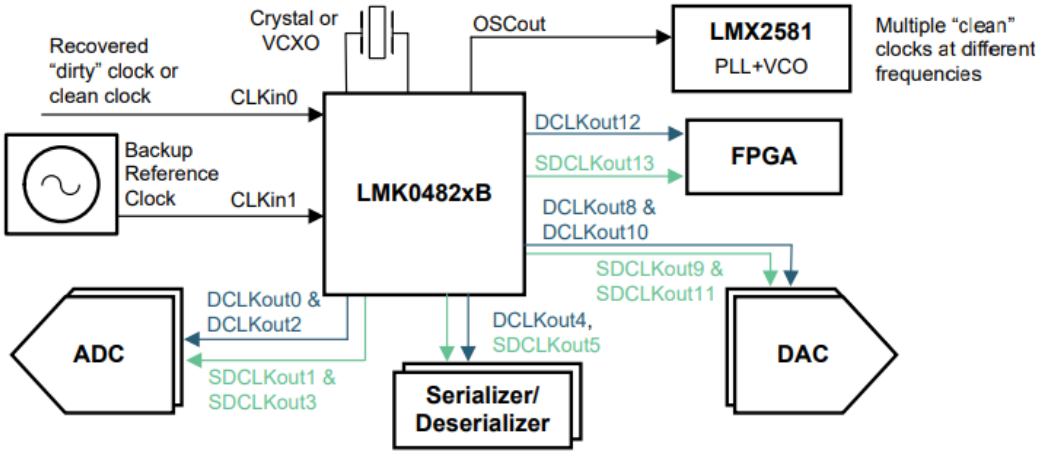

Figure 19: FMC Clocking Chip Interface

The DMA access must also be configured which is an 8GB DDR4 SO-DIMM by Micron (MTA18ASF1G72HZ). This is built into the PC821 and is used for storing the large buffers before being read by the CPU.

The remaining configurations needed are much simpler and require setting values within a number of register inside the AXIS routers to set the correct forwarding, and the buffer size for

the capture star to capture. Then the FMC120's hardware trigger can be written to so that both ADC0 and ADC1 capture the buffers of data simultaneously.

Cross Correlation: The optimal shift index to align the ideal signal to the non-ideal signal can be found by implementing a software cross correlation on the buffers of samples stored in the capture unit and passed to the CPU. The signals are then shifted across one another until the largest magnitude of the individual sums is found. Both signals are cross correlated with one another with new buffers of data and these values must be found to be equal magnitude. Once they are then this index is written to the DSP IP.

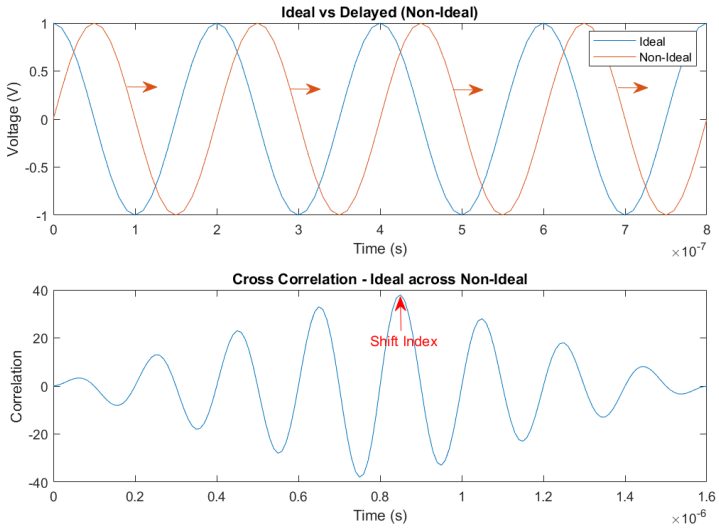

Figure 20: Cross Correlation of Sinusoid

Depending on the signal frequency the window size (or buffer size) can be varied and the number of times of the index is computed to assure the optimal index for minimum residual.

Interpolation The weighting is decided by software in order to account for any shift that the sampling speed can't account for. As the samples are set at a 1GHz speed, the worst possible

error in the shift index would be 0.5ns. The following graph shows this margin of error around a 50MHz input signal.

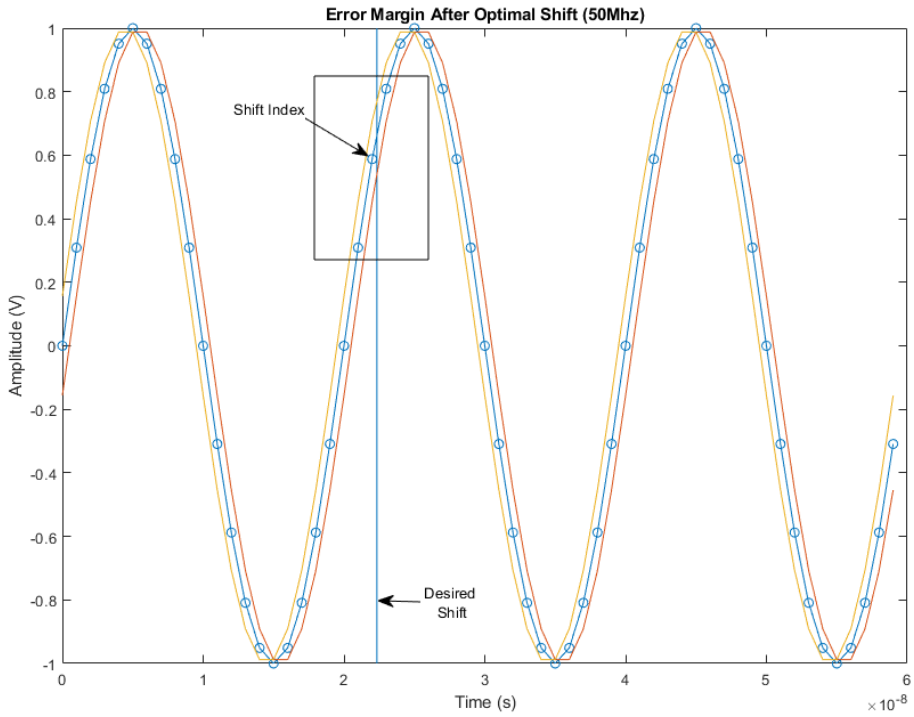

Figure 21: Error Margin Around 50MHz Input Signal

The plot below shows a zoomed in look at the above plot, but with additional points showing the computed sample with different weighting schemes. The shift index will be computed to the current sample in the plot as this will result in the smallest residual. The actual shift value can be seen to fall between the upcoming sample and the current sample, but still closer to the current sample. As a result using a weighting scheme of something like ¾ weight to the current sample and ¼ weight to the next sample would result in a much better residual.

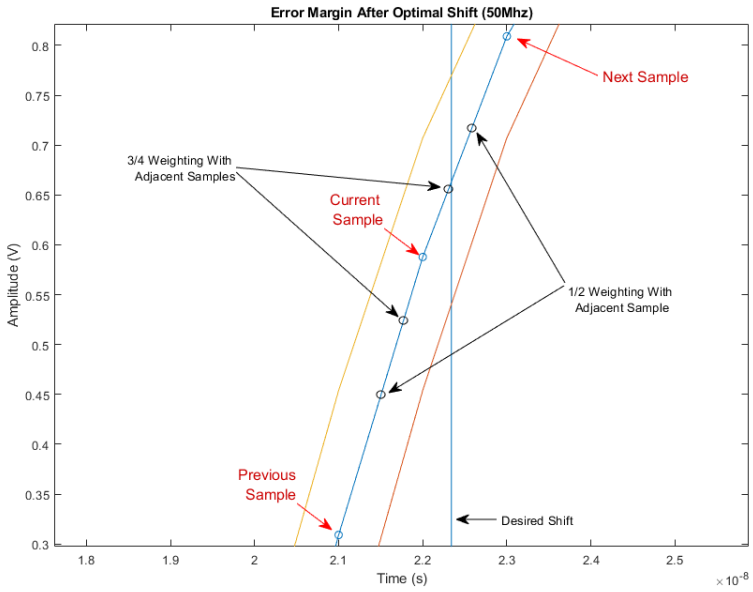

Figure 22: Zoomed In Look with Different Weightings

For maximum speed and throughput the hardware makes use of shifting for all division so that numerous multipliers/dividers and/or floating point multipliers are not needed on a per sample basis. The simplest implementation is assigning the current output as the average of the current output and the next output. The two samples can simply be summed and then shifted right by one.

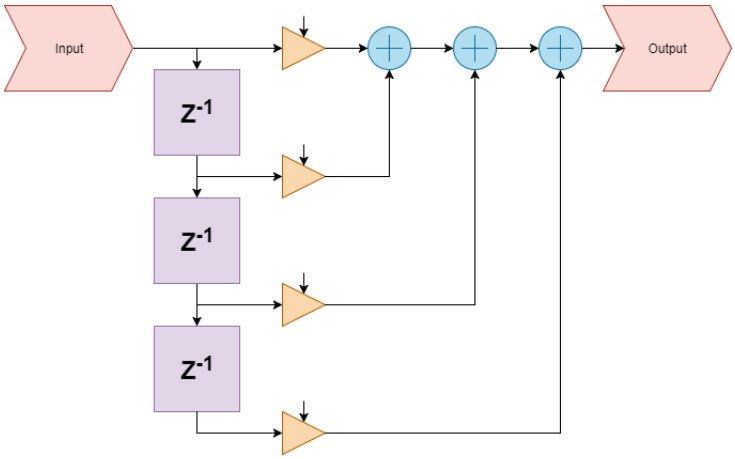

Figure 23: DSP Output Sample Weighting

As frequency increases the need for more precise interpolation is needed as the change in input signal per sample is greater. For a 50 MHz signal the below plot shows the residual for the worst possible situation, which is where the signal falls directly between the samples.

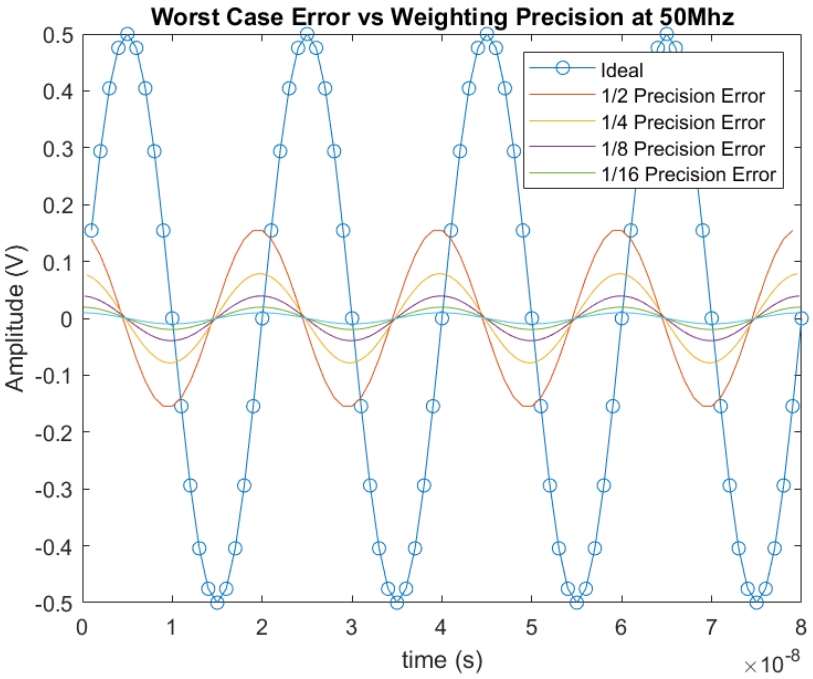

Figure 24: Worst case Error for Different Weighting Schemes

The peaks of the residuals is shown in the table below. It can be seen that for higher frequencies there is a definite need for greater than a quarter sample precision. 1/16 sample interpolation can be implemented with a single clock cycle penalty, and 1/32 sample interpolation with an additional three clock cycles.

Weighting Precision	50Mhz Worse Case Error
1/2 Sample	78mV
1/4 Sample	39mV
1/8 Sample	19mV
1/16 Sample	9.8mV
1/32 Sample	5mV

Table 3: Worst Case Errors for 50MHz:

Find Scaler: This follows the shift index computation and interpolation for the best alignment of the signals. Both signals then will be aligned on the CPU so that the scaler optimization can begin. This scaler is found by sweeping a multiplier for the non-ideal waveform such that the residual is minimized. Each digit is optimized and then the range around that digit is compared for the optimum second digit and so on and so forth for as many digits of precision are needed. This saves from running tens of thousands of comparisons to just about 10 per digit of precision.

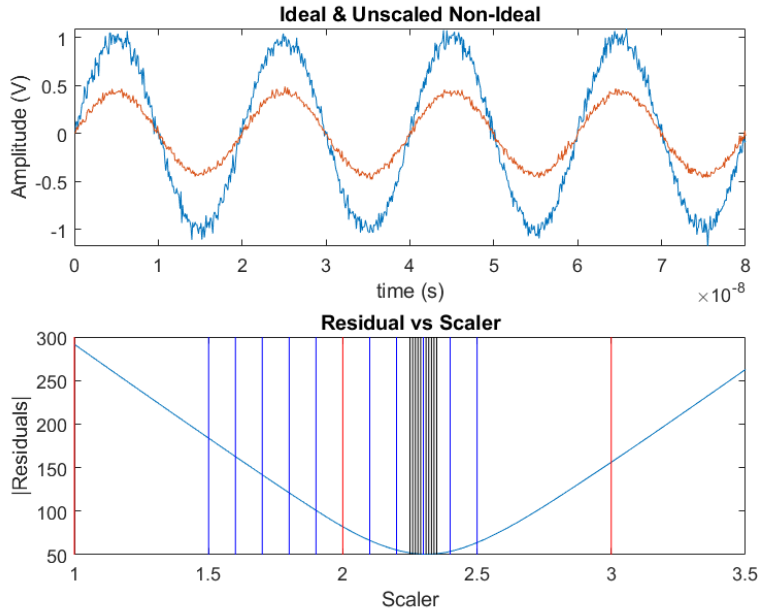

Figure 25: Find Scaler Algorithm for Minimal Residual

Results

A simple test can be run by passing a single 5MHz sine wave into the ideal input and the inverted signal passed into the non-ideal input. These can be seen as the yellow and purple waveforms, (although the purple waveform is after scaling ~0.99832), and are returned out the debug ports from the DSP IP. The red waveform is a waveform that resulted from shifting the

ideal waveform to optimally match the blue non-ideal waveform. The shifted and scaled waveforms are subtracted, given weights and the residual is passed out through the blue waveform. This reduced the $0.9V_{p\text{-}p}$ inputs by more than 70dB.

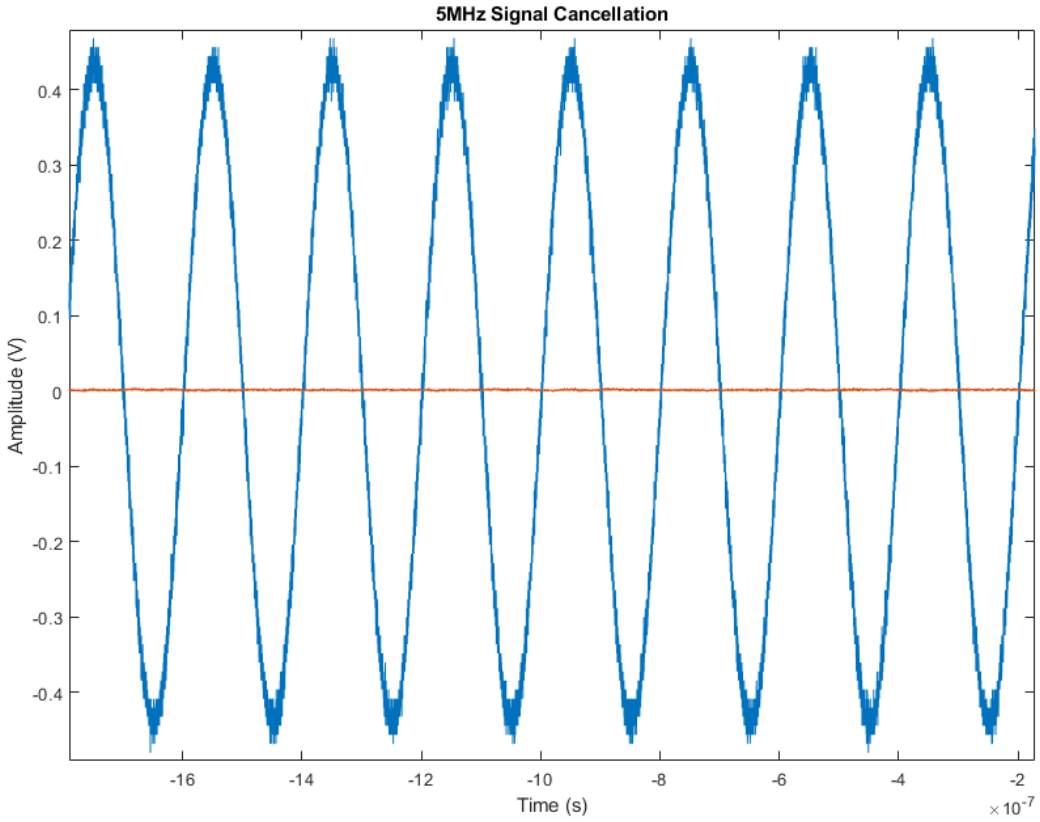

Figure 26: Single Sinusoid Cancellation

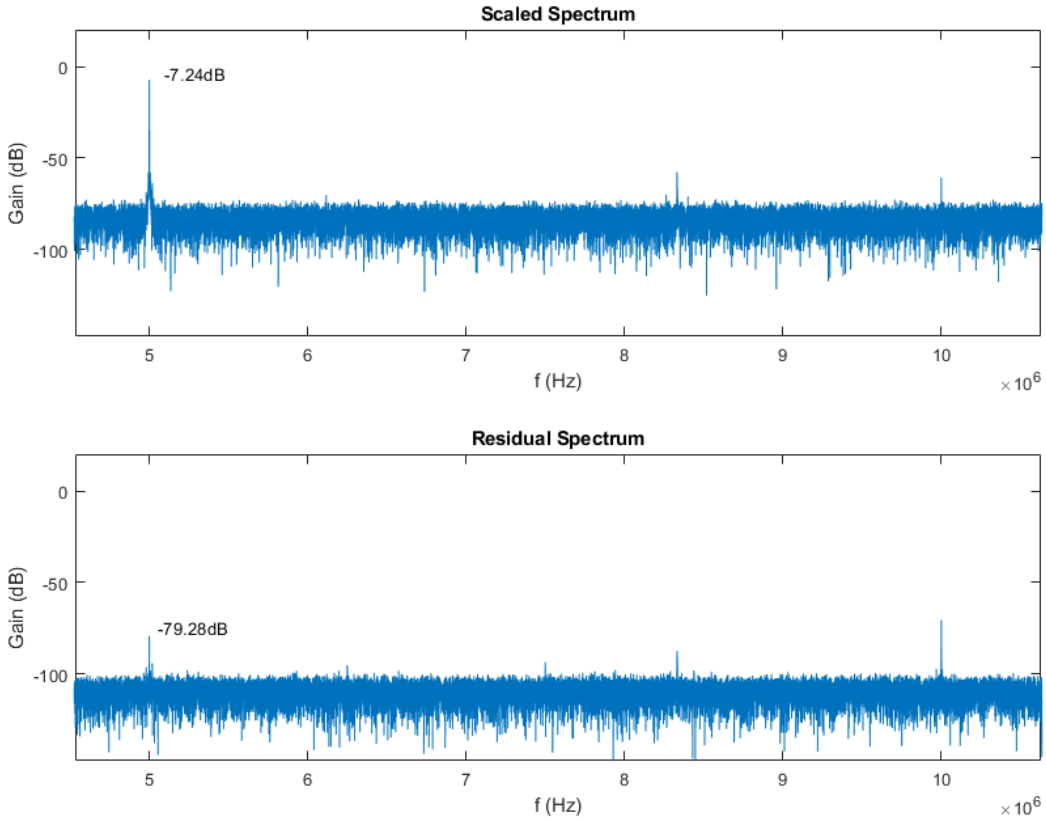

Figure 27: Single Sinusoid Cancellation

Taking the FFT of the measurement the 72dB for a 5MHz signal can be seen. Also an increase in the 2nd harmonic at 10MHz. This is caused by the zero crossings of the waveform. As the greatest change in the waveform occurs here, the largest error is always found at the zero crossings and thus the error has peaks at twice the frequency of the input.

This same results are shown below for a 50MHz input and a delayed, and attenuated 50MHz signal:

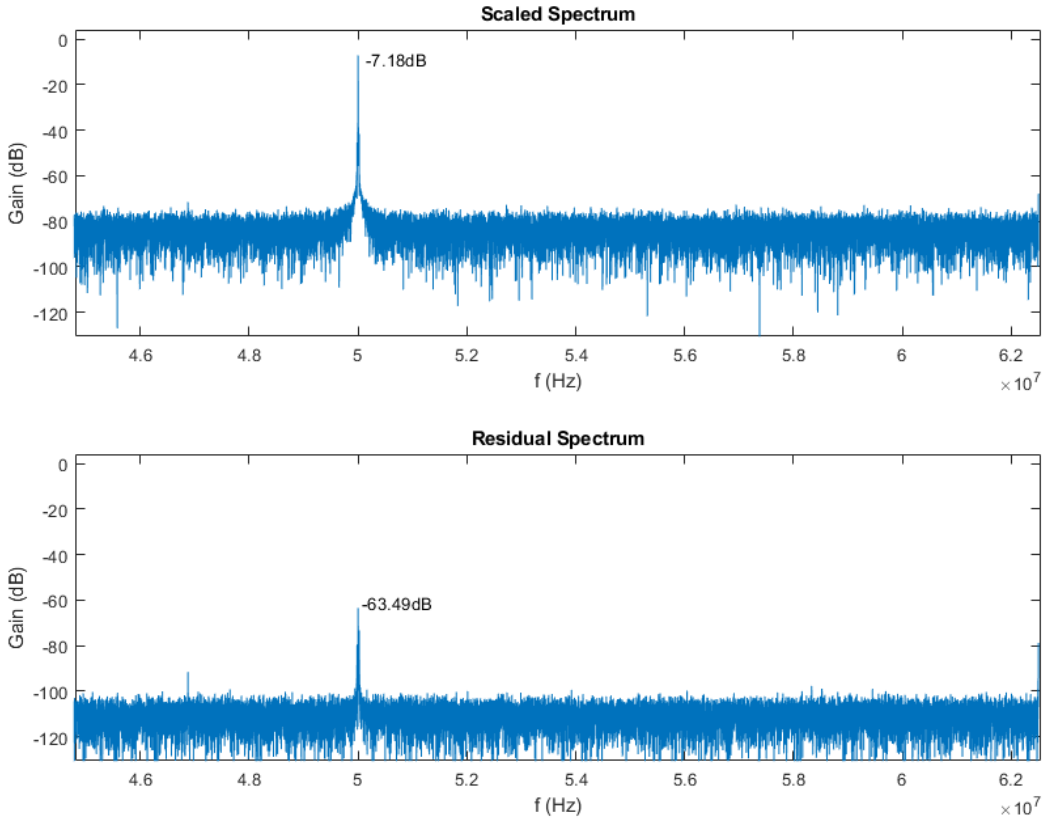

Figure 28: 50MHz Single Sinusoid Cancellation

A deterioration can be seen here for the higher frequencies as the sampling speed starts to have a greater toll on the shift precision. The result above is with implementing 1/32ns shifting precision. The result is 56.31dB of cancellation.

Next, a 900mV$_{p\text{-}p}$ signal at 50MHz is passed into both a non-ideal and ideal inputs. The non-ideal has an additional 46MHz 50mV$_{p\text{-}p}$ signal summed on top of it. The non-ideal FFT can be seen in the first graph and the FFT of the residual after DSP does its computation.

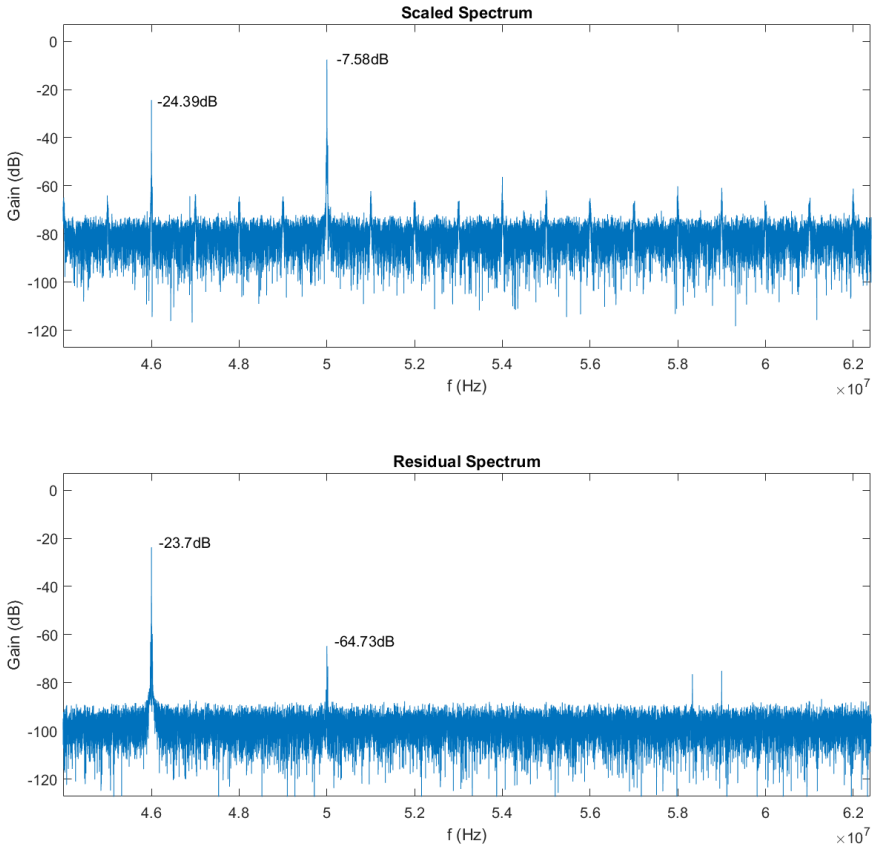

Figure 29: Two Sinusoidal Cancellation FFT

The ideal signal starts at -7.58dB and is attenuated down to -64.73, which is cancellation of 57.15dB. The non-ideal signal only has a loss of about .69dB. This same result can be seen even if the signals are very close in frequency. This same result is seem with 50MHz and 49.5MHz below:

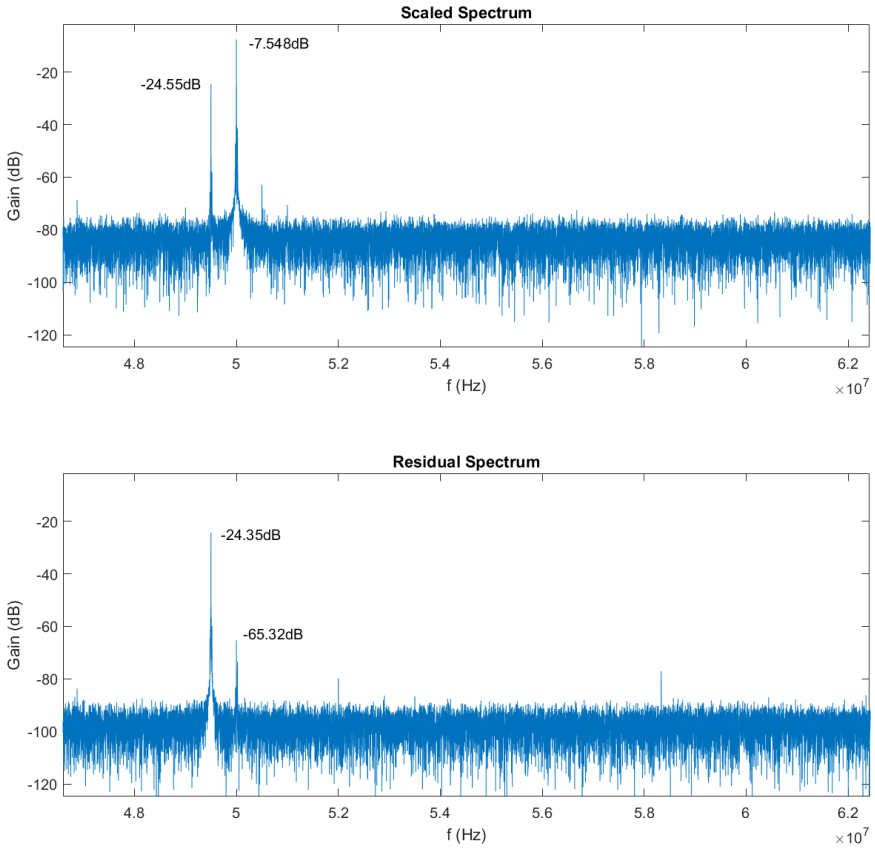

Figure 30: 50MHz Ideal Signal Cancellation with 49.5MHz Residual

Conclusion & Future Work

A good deal more work is needed to implement this in a full duplex application as this does not account for any nonlinear effects of a system. It accounts for any delay and linear scaling that a signal experiences and can reduce these by more than 70dB-55dB for the frequency range of 5MHz-50MHz. This performance decreases dramatically when nonlinearities are introduced and this would require calculating Volterra coefficients through software calculation and hardware to implement the filtering.

The end result for this project is consistently more than 55dB cancellation of the ideal from the non-ideal signal. For STAR applications with an analog front end getting 40dB already,

this could then be doubled for a 95dB+ cancellation. This design does not implement any sort of filtering that could potentially increase the cancellation additionally. A configurable FIR filter could be added through the AXIS router, so that the ADC samples can be sent through a filter or bypassed to the DSP IP. This would add an additional latency, but as the shift register is not the bottleneck for the ideal signal this would not be an issue and the same throughput could be kept. This filter could be set to filter out the second harmonic that typically is setting the noise floor for this project. Additionally, adding greater sampling speeds to keep the performance at 70db+ cancellation for frequencies higher than 5MHz would be another improvement.