Low Cost FPGA based Implementation of a DRFM System

Author: *:*

Misha Mesarcik

msrmic004@myuct.ac.za

Friday 15th September, 2017

Presentation Outline

Project Description

Introduction

Background to Study

Introduction

Background to Study

- What is Digital Radio Frequency Memory?
  - Digitize and store incoming RF input signals
  - Time delay
  - Frequency shift
  - Scale Amplitude
  - Retransmit
- Used to deceive radars in ECM.

Introduction

Background to Study

- What is Digital Radio Frequency Memory?
  - Digitize and store incoming RF input signals
  - Time delay
  - Frequency shift
  - Scale Amplitude
  - Retransmit
- Used to deceive radars in ECM.

Motivations for Low Cost DRFM

Introduction

Background to Study

- What is Digital Radio Frequency Memory?
  - Digitize and store incoming RF input signals
  - Time delay
  - Frequency shift
  - Scale Amplitude
  - Retransmit
- Used to deceive radars in ECM.

Motivations for Low Cost DRFM

- DRFM systems are currently very expensive.
- Could be integrated into the PCL group's systems

Introduction

Archetypal DRFM Architecture

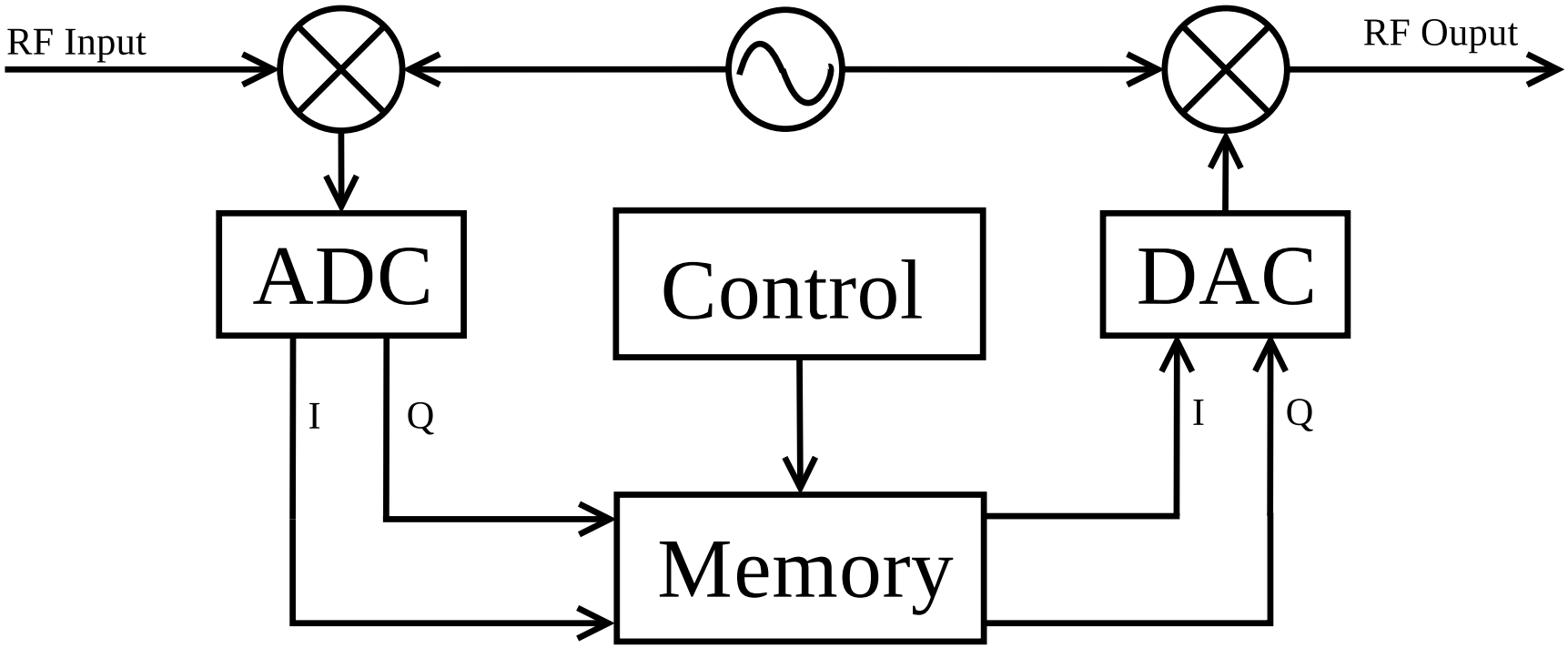


Figure: DRFM Architecture

System Overview

System Overview

System break down

System Overview

System break down

On the FPGA there are the following subsystems

System Overview

System break down

On the FPGA there are the following subsystems

- Interfacing

System Overview

System break down

On the FPGA there are the following subsystems

- Interfacing
- Peripherals

System Overview

System break down

On the FPGA there are the following subsystems

- Interfacing
- Peripherals
- Digital Signal Processing (DSP)

System Overview

System break down

System Overview

System break down

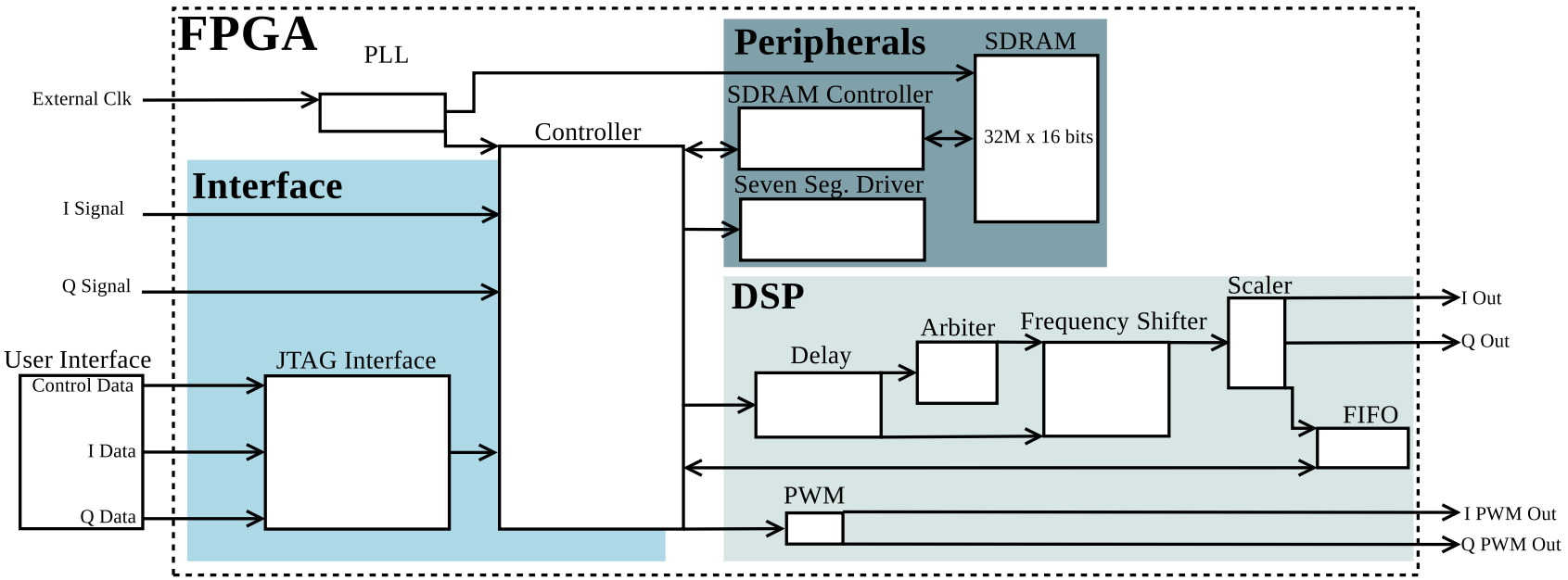


Figure: DRFM System Overview

System Overview

JTAG Interface

System Overview

JTAG Interface

- JTAG based interface
- Implements Altera's Virtual JTAG Interface IP Core
- Allows for both real time control and SDRAM Injection

System Overview

JTAG Interface

- JTAG based interface
- Implements Altera's Virtual JTAG Interface IP Core
- Allows for both real time control and SDRAM Injection

User Interface

System Overview

JTAG Interface

- JTAG based interface
- Implements Altera's Virtual JTAG Interface IP Core
- Allows for both real time control and SDRAM Injection

User Interface

- Built on PyQt and the Altera TCL scripting API
- Allowed for fine grain control over
  - Frequency Shift (32 bit resolution)
  - Time Delay (10 bit resolution)
  - Amplitude Scaling (16 bit resolution)
- As well as facilitating I/Q Data injection into SDRAM

System Overview

User Interface

Control Information	Control Flag Position	Control Data Length
Doppler Shift	33	32
Time Delay	44	10
Amplitude Scale	61	16

System Overview

External Peripherals

System Overview

External Peripherals

- Interfacing with SDRAM on the DE10-lite Development Board
- Seven Segment Display
- PWM output

System Overview

Signal Model

System Overview

Signal Model

- I/Q data was either received from RF front end or from SDRAM injection.
- Done to minimize sampling frequency criteria
- Both I and Q channels were 16 bits wide

System Overview

Signal Model

- I/Q data was either received from RF front end or from SDRAM injection.
- Done to minimize sampling frequency criteria
- Both I and Q channels were 16 bits wide

$$s_{rx}(n) = v_{rx}(n) + j\mathbb{H}\{v_{rx}(n)\} = I(n) + jQ(n)$$

System Overview

Digital Signal Processing

System Overview

Digital Signal Processing

- Order of operations was important
- Time Delay
  - Worked by changing index of a RAM
  - $s_{delay}(k) = s_{rx}(n-k), \quad n \geq k$

System Overview

Digital Signal Processing

- Frequency Shift
  - As I/Q data was inject, it simplified the frequency shift operation
  - Naturally a frequency shift is represented by a complex exponential representation: $s_s(n) = s_{rx}(n)e^{j2\pi f_s n}$
  - Doesn't translate well to digital systems, so:

$$\begin{aligned} s_s(n) &= [I(n) + jQ(n)][cos(2\pi f_s n) + jsin(2\pi f_s n)] \\ &= [I(n)cos(2\pi f_s n) - Q(n)sin(2\pi f_s n)] \\ &\quad + j[I(n)sin(2\pi f_s n) + Q(n)cos(2\pi f_s n)] \\ &= I_s(n) + jQ_s(n) \end{aligned}$$

System Overview

Digital Signal Processing

- Amplitude Scaling
  - As UI sends a 16 bit amplitude control word need to somehow scale with it
  - This value needs to be scaled so it is between 1 and close to zero

$$s_{scaled}(n) = 2^{-p} s_{rx}(n), \quad 0 \leq p \leq 15$$

Detailed Design

Detailed Design

User Interface

Table: Control Word Composition

Detailed Design

User Interface

- Always be available to send control data
- Should be able to quickly change to SDRAM Injection
- Using PyQt/Altera TCL Scripting API

Table: Control Word Composition

Detailed Design

User Interface

- Always be available to send control data
- Should be able to quickly change to SDRAM Injection
- Using PyQt/Altera TCL Scripting API

Table: Control Word Composition

UI Flowchart

Detailed Design

User Interface

- Always be available to send control data
- Should be able to quickly change to SDRAM Injection
- Using PyQt/Altera TCL Scripting API

Table: Control Word Composition

UI Flowchart

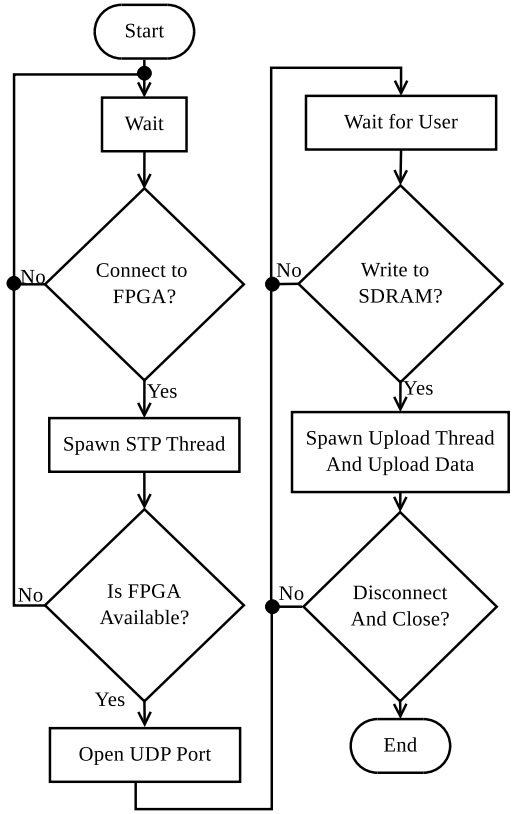


Figure: Flow chart showing the User Interfacing Structure

Detailed Design

User Interface

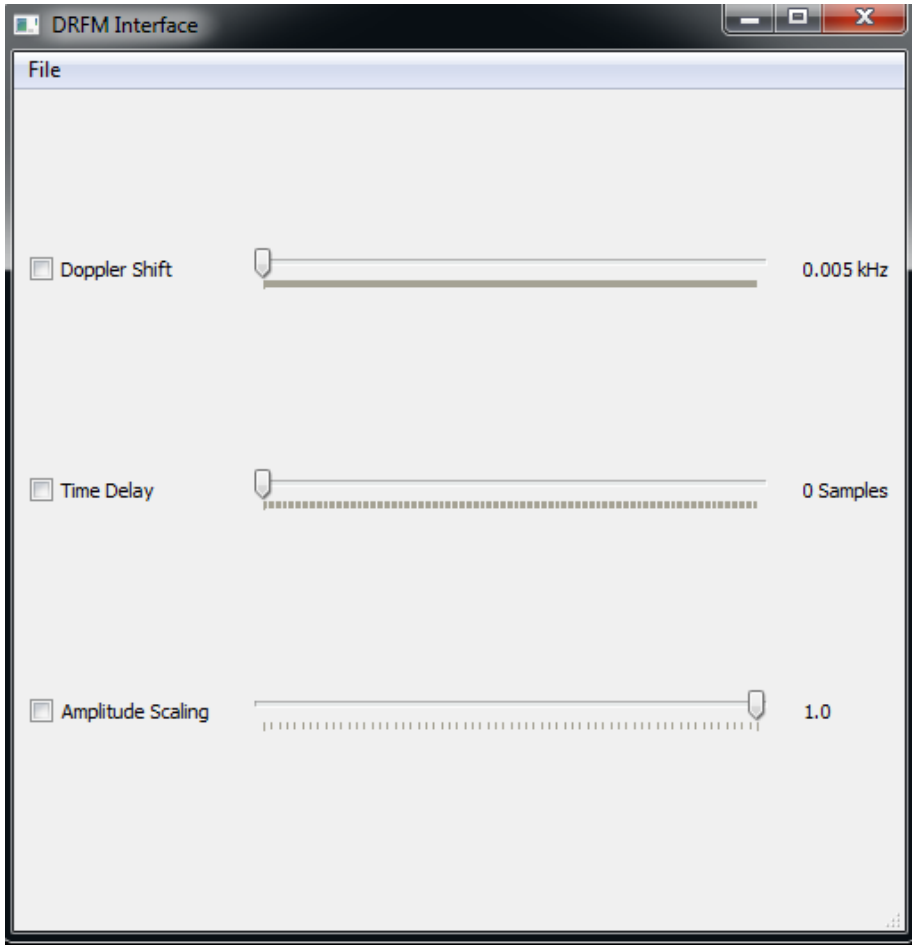


Figure: Functional UI

Detailed Design

JTAG Interface

Detailed Design

JTAG Interface

- Two modes of operation
  - Write Data (TDI line routed to SDRAM Controller)
  - Data Capture Mode (TDI line shifted and captured)
  - Determined by UI and by SW0 on Development Board

Detailed Design

JTAG Interface

- Two modes of operation
  - Write Data (TDI line routed to SDRAM Controller)
  - Data Capture Mode (TDI line shifted and captured)
  - Determined by UI and by SW0 on Development Board

JTAG Physical Lines

Detailed Design

JTAG Interface

- Two modes of operation
  - Write Data (TDI line routed to SDRAM Controller)
  - Data Capture Mode (TDI line shifted and captured)
  - Determined by UI and by SW0 on Development Board

JTAG Physical Lines

Pin Name	Description
tck	Test Clock Input
tdi	Test Data Input
tdo	Test Data Output
ir_in	Instruction Input
cdr	Capture Data Register
sdr	Shift Data Register
udr	Update Data Register

Figure: Table showing JTAG lines used

Detailed Design

Controller Module

- Arbitrates between SDRAM and RAM
  - Dependant on `Read_DataValid` and `Read_WaitRequest`
- Also performs changes in assignment for reading and writing from SDRAM

Detailed Design

SDRAM Subsystem

- Uses Altera's SDRAM Controller IP Core
- Uses Altera's PLL IP Core

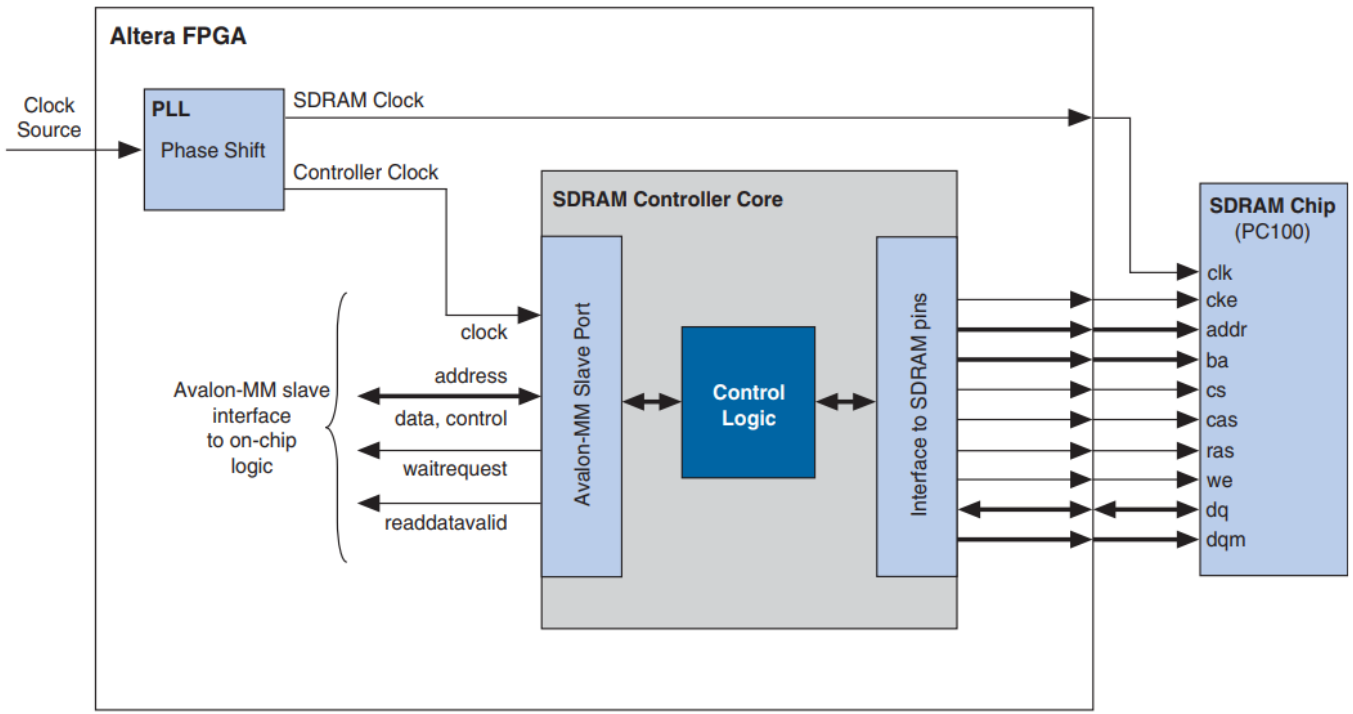


Figure: Block Diagram of the SDRAM interfacing subsystem

Detailed Design

DSP Chain

- Delay module
  - Integrated into the controller module
  - Takes data from SDRAM
  - Writes it to a 2048 address wide RAM
  - Implements a state machine for keeping track of the indexing

Detailed Design

DSP Chain

Detailed Design

DSP Chain

- Arbiter
  - As data in SDRAM was 16 bits per address
  - Each I/Q were 16 bits wide each
  - Needed both for Frequency shifting operation
  - Used FSM to arbiter RAM data so that both I/Q data streams were available at the same time

Detailed Design

DSP Chain

- Arbiter
  - As data in SDRAM was 16 bits per address
  - Each I/Q were 16 bits wide each
  - Needed both for Frequency shifting operation
  - Used FSM to arbiter RAM data so that both I/Q data streams were available at the same time

Arbiter State machine

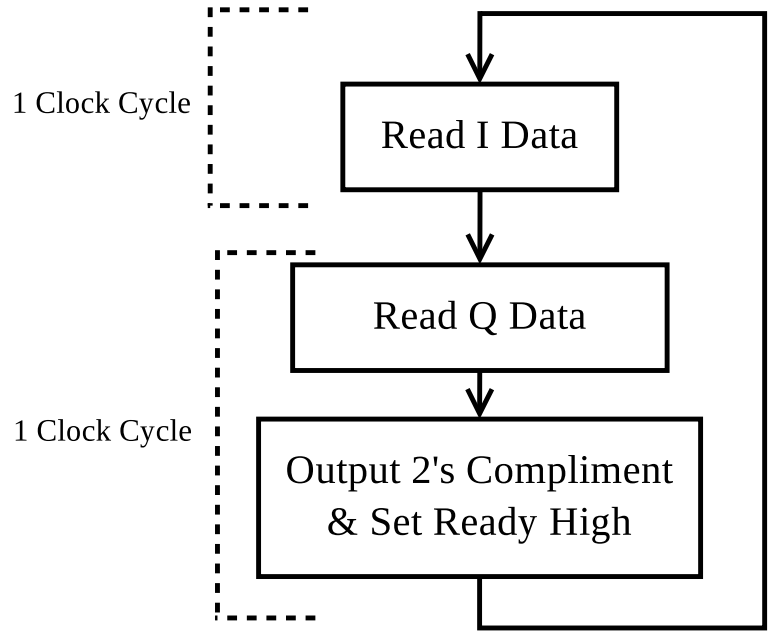


Figure: State Diagram of the Arbiter

Detailed Design

Frequency Shifter

Detailed Design

Frequency Shifter

- NCO
  - Required to perform frequency shifting as per equation shown previously
  - Look up table that translates input frequency to a periodic waveform
  - Used a 4086 address dual port ram
  - Address counters were 1024 addresses out of phase to get both sine and cosine
  - Initialized with MATLAB script

Detailed Design

Frequency Shifter

- NCO
  - Required to perform frequency shifting as per equation shown previously
  - Look up table that translates input frequency to a periodic waveform
  - Used a 4086 address dual port ram
  - Address counters were 1024 addresses out of phase to get both sine and cosine
  - Initialized with MATLAB script

NCO Block Diagram

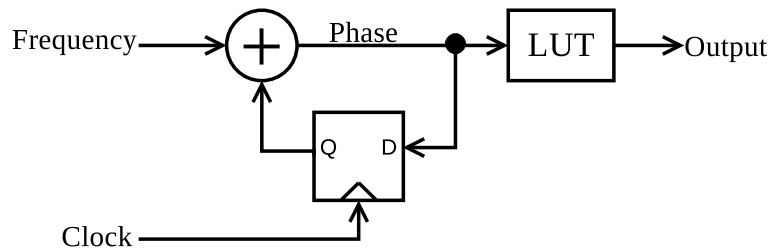


Figure: Diagram of NCO Operation

$$f_{shift} = f_{slider} \frac{2^{32}}{100MHz}$$

Detailed Design

Frequency Shifter

Detailed Design

Frequency Shifter

- Arithmetic Operations
  - 4 multiplies, 1 add, 1 subtract
  - All checked overflows for the 32 bit 2's complement result
  - Maximum negative value 0x8000 0000
  - Maximum positive value 0x7FFF FFFF

Detailed Design

Frequency Shifter

- Arithmetic Operations
  - 4 multiplies, 1 add, 1 subtract
  - All checked overflows for the 32 bit 2's complement result
  - Maximum negative value 0x8000 0000
  - Maximum positive value 0x7FFF FFFF

Arithmetic Unit

Detailed Design

Frequency Shifter

- Arithmetic Operations
  - 4 multiplies, 1 add, 1 subtract
  - All checked overflows for the 32 bit 2's complement result
  - Maximum negative value 0x8000 0000
  - Maximum positive value 0x7FFF FFFF

Arithmetic Unit

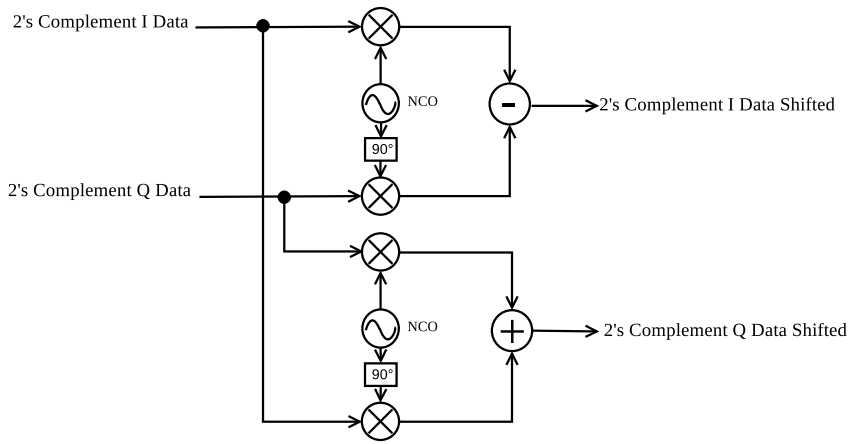


Figure: Diagram of Arithmetic Operations

Detailed Design

DSP Chain

- Amplitude Scaling Module
  - Received unsigned, time delayed, frequency shifted 32 bit I/Q data
  - Received amplitude scaling control word
  - Applied non circular shifts to get amplitude scale between 1 and 3.05×10^{-5}
  - Did multiplication and checked overflows

Results and Conclusions

Results and Conclusions

User Interface

- Works well but some latency
- Cannot receive data from FGPA

Time Delay

Frequency Shifting

Results and Conclusions

User Interface

- Works well but some latency
- Cannot receive data from FGPA

Time Delay

- Impossible to see in real time.

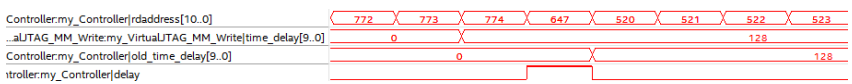


Figure: Signal Tap Delay Result

Frequency Shifting

Results and Conclusions

User Interface

- Works well but some latency
- Cannot receive data from FGPA

Time Delay

- Impossible to see in real time.

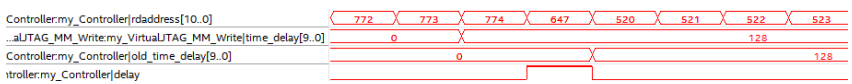
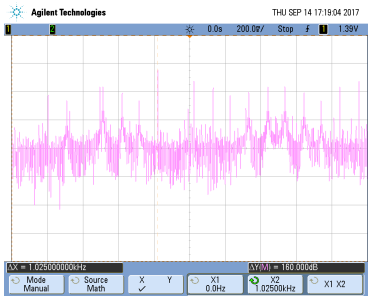


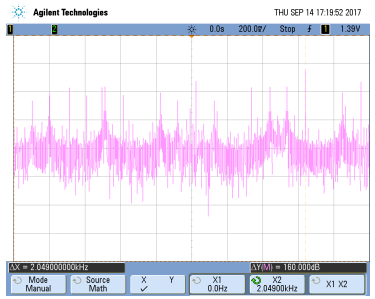
Figure: Signal Tap Delay Result

Frequency Shifting

- Injected sum of 5 sinusoids.
- Noise possibly from PWM or from concatenation of vector



(a) Oscilloscope Print of a 1kHz frequency shift

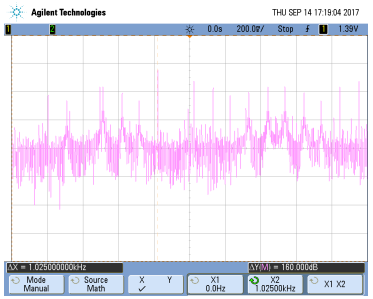


(b) Oscilloscope Print of a 2kHz frequency shift

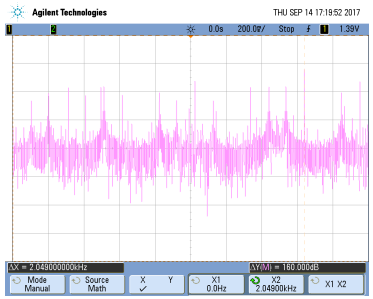
Detailed Design

Frequency Shifting

- Injected sum of 5 sinusoids.
- Noise possibly from PWM or from concatenation of vector



(a) Oscilloscope Print of a 1kHz frequency shift



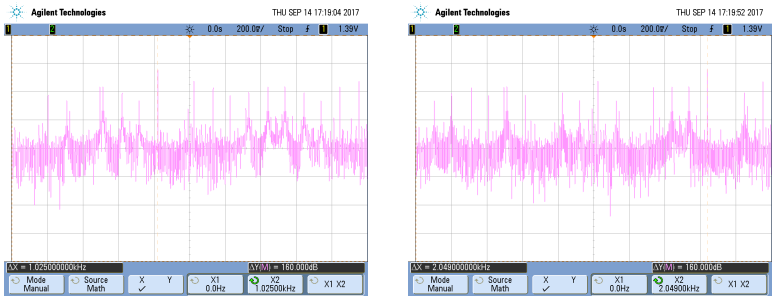
(b) Oscilloscope Print of a 2kHz frequency shift

Amplitude Scaling

Detailed Design

Frequency Shifting

- Injected sum of 5 sinusoids.
- Noise possibly from PWM or from concatenation of vector

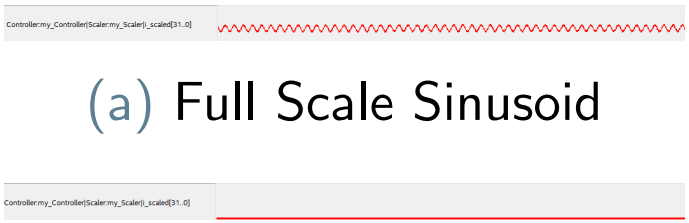


(a) Oscilloscope Print of a 1kHz frequency shift

(b) Oscilloscope Print of a 2kHz frequency shift

Amplitude Scaling

- Very easy to see
- Effective.



(a) Full Scale Sinusoid

(b) Fully Attenuated Sinusoid

Figure: Figures Showing the Measured Amplitude Scaling

Recommendations

Recommendations

Potential Future Work

Recommendations

Potential Future Work

- Increased Delay Functionality
- Improved Frequency Measurement
- Integration into a RF Front-end
- Extension to Multiple Targets

Demo

Questions?