Measurement Board
Turn-On Notes

Version: 1.00

Authors: Jeremy W. Webb

Email: jwwebb@ece.ucdavis.edu

Contents

List of Figures

List of Tables

Table 1: Revision History

Revision	**Date**	**Description**	**Initials**
1.00	05/22/2009	First Draft	JWW

1 Overview

This document is intended to document the turn-on and troubleshooting of the p342 Measurement Board.

2 Reference Documents

2.1 Schematics

The Measurement Board schematics are located at the following url:

http://www.ece.ucdavis.edu/vcl/vclpeople/jwwebb/measbd/docs/sch_p342_blk.pdf

2.2 Datasheets and User Guides

2.2.1 Xilinx Spartan-3A

Manufacturer: Xilinx, Inc.
Manufactuers Part Number: XC3S1400A-4FGG484C

1. **Data Sheet**
   http://www.xilinx.com/support/documentation/data_sheets/ds529.pdf
2. **User Guide**
   http://www.xilinx.com/support/documentation/user_guides/ug331.pdf
3. **Configuration Guide**
   http://www.xilinx.com/support/documentation/user_guides/ug332.pdf

2.2.2 Texas Instruments High-Speed 16-bit 1GS/s D/A Converter (DAC)

1. **DAC5682ZIRGCT Data Sheet**
   http://focus.ti.com/lit/ds/symlink/dac5682z.pdf

2.2.3 Texas Instruments Temperature Sensors and Fan Controllers

1. **AMC6821 Data Sheet**
   http://focus.ti.com/lit/ds/symlink/amc6821.pdf
2. **TMP125 Data Sheet**
   http://focus.ti.com/lit/ds/symlink/tmp125.pdf

2.2.4 Analog Devices Clock PLL IC

1. **AD9516-3BCPZ Data Sheet**
   http://www.analog.com/UploadedFiles/Data_Sheets/AD9516_3.pdf

2.2.5 UMC 1GHz VCO

1. **UMX-244-B14 Data Sheet**
   http://www.vco1.com/SCDs/scd244-a.pdf

2.2.6 Vectron VTC2 TCXO

1. **Vectron International VTC2 Series TCXO**
   http://vectron.com/products/tcxo/VTC2.pdf

2.2.7 Silicon Laboratories CP2102

1. **CP2102 Product Brief**
   http://www.silabs.com/public/documents/marcom_doc/pbrief/Microcontrollers/Interface/en/CP2102_Brief.pdf
2. **CP2102 Data Short**
   http://www.silabs.com/public/documents/tpub_doc/dshort/Microcontrollers/Interface/en/CP2102_short.pdf
3. **CP2102 Data Sheet**
   http://www.silabs.com/public/documents/tpub_doc/dsheet/Microcontrollers/Interface/en/cp2102.pdf
4. **CP2102 Evaluation Kit User Guide**
   http://www.silabs.com/public/documents/tpub_doc/evbdsheet/Microcontrollers/Interface/en/CP2102-EK.pdf
5. **CP2102 Virtual COM Port Driver**
   http://www.silabs.com/tgwWebApp/public/web_content/products/Microcontrollers/USB/en/mcu_vcp.htm

2.2.8 Future Technology Devices International Ltd. (a.k.a, FTDI Chip)

1. **FT245BL Data Sheet**
   http://www.ftdichip.com/Documents/DataSheets/DS_FT245BL.pdf
2. **FT245BM Designers Guide (DG245)**
   http://www.ftdichip.com/Documents/AppNotes/DG245_20.pdf
3. **FT245BM Bit-Bang Mode (AN232B-01)**
   http://www.ftdichip.com/Documents/AppNotes/AN232B-01_BitBang.pdf
4. **FT245BM Power Control and Pin States (AN232B-02)**
   http://www.ftdichip.com/Documents/AppNotes/AN232B-02_PinModes_20.pdf
5. **Optimizing D2XX Data Throughput (AN232B-03)**
   http://www.ftdichip.com/Documents/AppNotes/AN232B-03_D2XXDataThroughput.pdf
6. **Data Throughput, Latency & Handshaking (AN232B-04)**
   http://www.ftdichip.com/Documents/AppNotes/AN232B-04_DataLatencyFlow.pdf
7. **Debugging FT245BM Based Designs (AN232B-06)**
   http://www.ftdichip.com/Documents/AppNotes/AN232B-06_11.pdf
8. **Configuring FTDI's VCP Drivers to use Location IDs (AN232B-07)**
   http://www.ftdichip.com/Documents/AppNotes/AN232B-07_LocIDs.pdf
9. **Using the Modem Emulation Mode in FTDI's VCP Driver (AN232B-09)**
   http://www.ftdichip.com/Documents/AppNotes/AN232B-09_Modem_Emulation_Mode.pdf
10. **Advanced Driver Options (AN232B-10)**
    http://www.ftdichip.com/Documents/AppNotes/AN232B-09_Modem_Emulation_Mode.pdf
11. **Virtual COM Port Drivers**
    http://www.ftdichip.com/Drivers/VCP.htm
12. **D2XX Drivers**
    http://www.ftdichip.com/Drivers/D2XX.htm

3 Block Diagrams

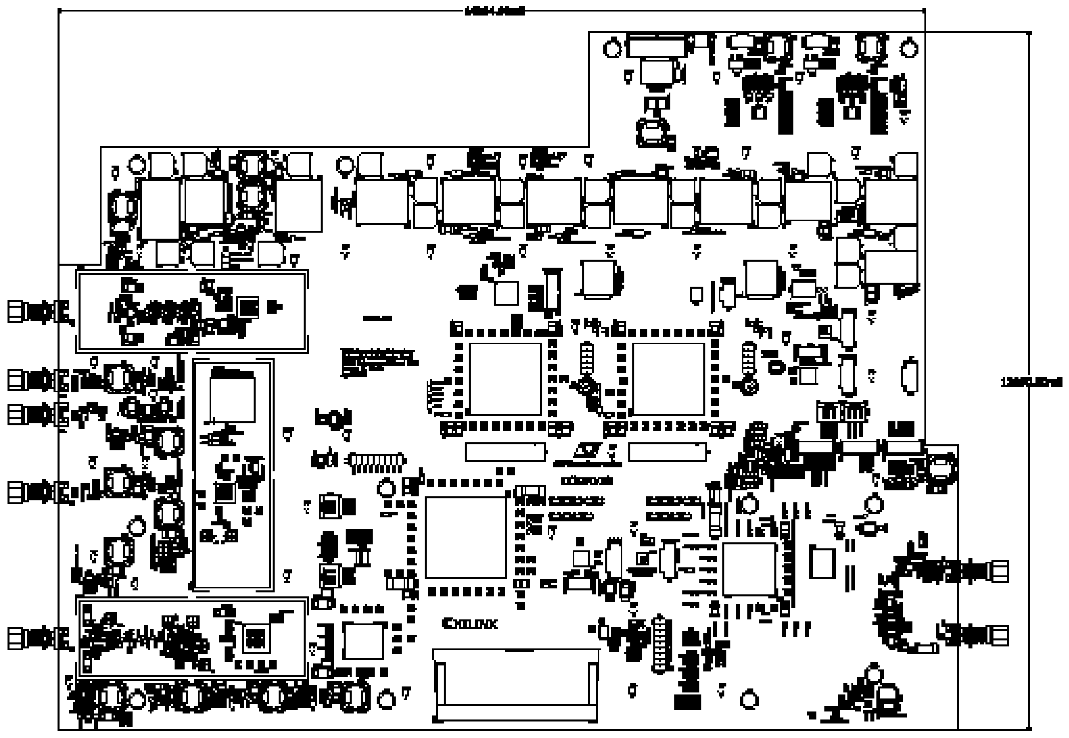

Figure 3.1: P342 Placement Block Diagram

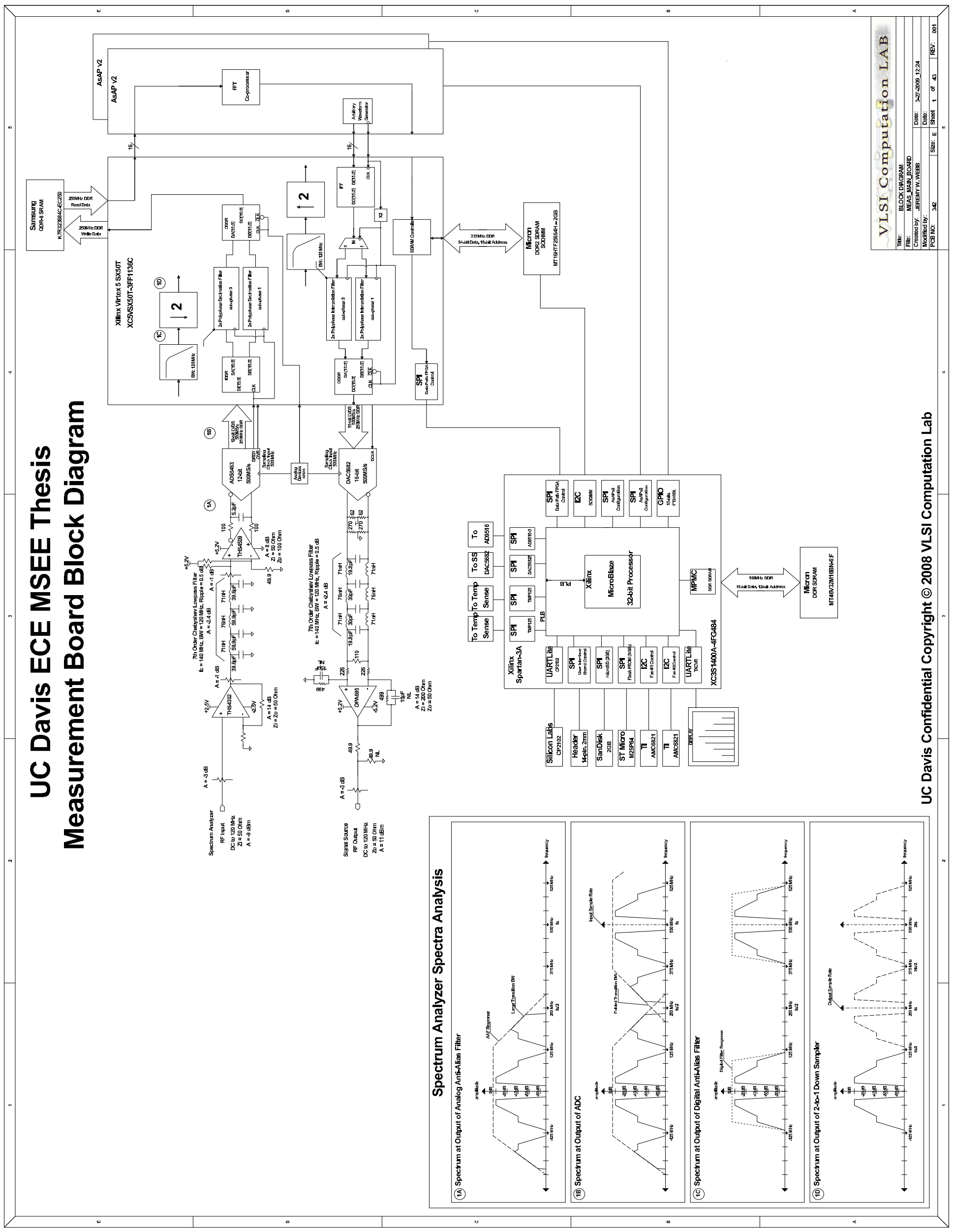
UC Davis ECE MSEE Thesis
Measurement Board Block Diagram
Spectrum Analyzer Spectra Analysis
Xilinx Virtex 5 SX50T
Xilinx Spartan-3A
MicroBlaze 32-bit Processor
AsAP v2
UC Davis Confidential Copyright © 2008 VLSI Computation Lab
VLSI Computation LAB

Figure 3.2: P342 Functional Block Diagram

4 PCB Turn-On Notes

4.1 Initial Turn-On

1. No shorts on power supplies.
   (a) Board Powers Up Properly.
   (b) All supplies within spec.
2. Control FPGA Programs via JTAG successfully.
   (a) Using Xilinx iMPACT Software.
3. SPI Configuration PROM programs via JTAG successfully.
   (a) Using Xilinx iMPACT Software.
4. Data Path FPGA Programs via JTAG successfully.
   (a) Using Xilinx iMPACT Software.
   (b) Operates at about 42° C with Fan Sink.
   (c) Operates at about 62° C without Heat Sink.
5. Control FPGA RS-232 (CP2102) interfaces functions correctly.
   (a) Operates a 115.2kBaud
   (b) Read/Write control of Control FPGA registers.
6. Control FPGA controls Data Path FPGA successfully via "ilb_master".
   (a) LEDs turn on and off.
   (b) Block RAM read registers respond correctly.
7. 1GHz VCO operating correctly.
8. 10MHz Internal/External circuits operating correctly.
9. 100MHz LVDS oscillator on both the Control FPGA and Data Path FPGA operating correctly.
10. Power CPLD Programs via JTAG Successfully.
11. Power CPLD receives 10MHz and properly distributes 312.5kHz to all DC-DC Converters.
12. Attempted to configure AD9516 Clock Generator.
    (a) Verilog HDL Driver verified in simulation.
    (b) Doesn't seem to be configured properly.
    (c) Double-check config registers to verify VCO is turned on properly.
    (d) It turns out there were a couple of problems.
        i. The SPI interface wasn't working very well at all. For example, I meant to write data 0x99 to address 0x000, but it was writing 0x20 to address 0x013. For some reason the data was shifted to the left by 5 bits. I simulated the design and it looked correct, so I just played around with the SCLK and SDIN shift registers. Eventually I was able to get it to look correct and I could read/write data reliably with a Perl Script.

ii. Once the SPI was fixed I could see clocks on all the desired outputs, but the frequencies were all off. I had set the Divider Low/High Cycles to 1 each for a divide by 2, but it was actually dividing by 4. I set them both to 0, and I got 500MHz out of my LVPECL outputs. The LVDS outputs were really off, because there are two dividers in that path. Now I'm getting the correct frequencies on all paths.

(e) I still need to tweak the differential voltages to get the correct common mode voltages. Some of them aren't where they should be.

13. Attempted to configure DAC5682Z 16-bit DAC.
    (a) Verilog HDL Driver verified in simulation.
    (b) Add read capability in hdl driver to verify functionality.

4.2 MicroBlaze Application Turn-On

1. Ported over MicroBlaze design from work project and verified that the DDR SDRAM is working correctly.
    (a) Bootloader Application successfully read out the MicroBlaze application from the SPI Configuration PROM and transfered the data to the DDR SDRAM.
        i. The Bootloader then successfully jumped to the application address 0x8c000000 and the MicroBlaze application began executing out of the DDR SDRAM.
    (b) The MicroBlaze application successfully read the DPIMAGE.bin file from the microSD card and configured the Data Path FPGA via a serial interface.
        i. The Data Path FPGA successfully programed.
        ii. The microSD card functions properly.
    (c) The MicroBlaze application successfully communicates with the Data Path FPGA via the ILB serial interface.
    (d) The MicroBlaze application successfully transfered a waveform from the microSD card to the Data Path FPGA.
    (e) For some reason the I2C Fan Controllers don't seem to be working.
        i. I get error messages pertaining to the iicWriteRegister.
            A. This error was due to an incorrect I2C address. The application was addressing Fan #1 at address 0x4C, when it was really at address 0x18. Once the application was updated the Fan Controllers started working properly.
        ii. Have some fans made up with connectors for testing the fan controllers.
    (f) AD9516 doesn't seem to be working with MicroBlaze application.
        i. Probe Serial Data and Clock to see if data is being transmitted at power-up.
            A. For some reason my custom Verilog HDL Peripheral wasn't shifting out 24 clocks, so the AD9516 wouldn't properly configure. I adjusted the shift count clock, and added a ChipScope Pro core to verify. I also discovered that the read wasn't shifting in the last bit, so I modified the shift register to clock read data in on the positive edge of sclk. Now the AD9516 is properly configured.
        ii. Create a debug command to configure AD9516 upon command.
            A. Completed. See command 30.
        iii. Create a debug command to read AD9516 register.
            A. Completed. See command 29.
        iv. Create a debug command to write AD9516 register.
            A. Completed. See command 28.
    (g) Add DAC5682Z peripheral configuration and control to current MicroBlaze application and system.mhs/mss files.
        i. Wrote SPI EDK Peripheral to control DAC5682Z via SPI Interface.
        ii. Initialization routine successfully completes.
        iii. Read/Write access correctly read/write to various registers.
        iv. Debug Commands:

   A. Command 31: write command.
   B. Command 32: read command.
   C. Command 33: initialization command.

(h) Use ChipScope Pro to verify SRAM and Block RAM pattern playback inside of Data Path FPGA.
   i. Block RAM Patterns Play back successfully in ChipScope Pro.
   ii. QDR-II SRAM Patterns Play back successfully in ChipScope Pro.

(i) The 6 temperature sensors appear to be functioning properly.
   i. Temp: 30.5° C, 27.75° C, 26.75° C, 29.25° C, 28.25° C, 29.50° C

4.3 AD9516 Clock Generation IC Turn-On

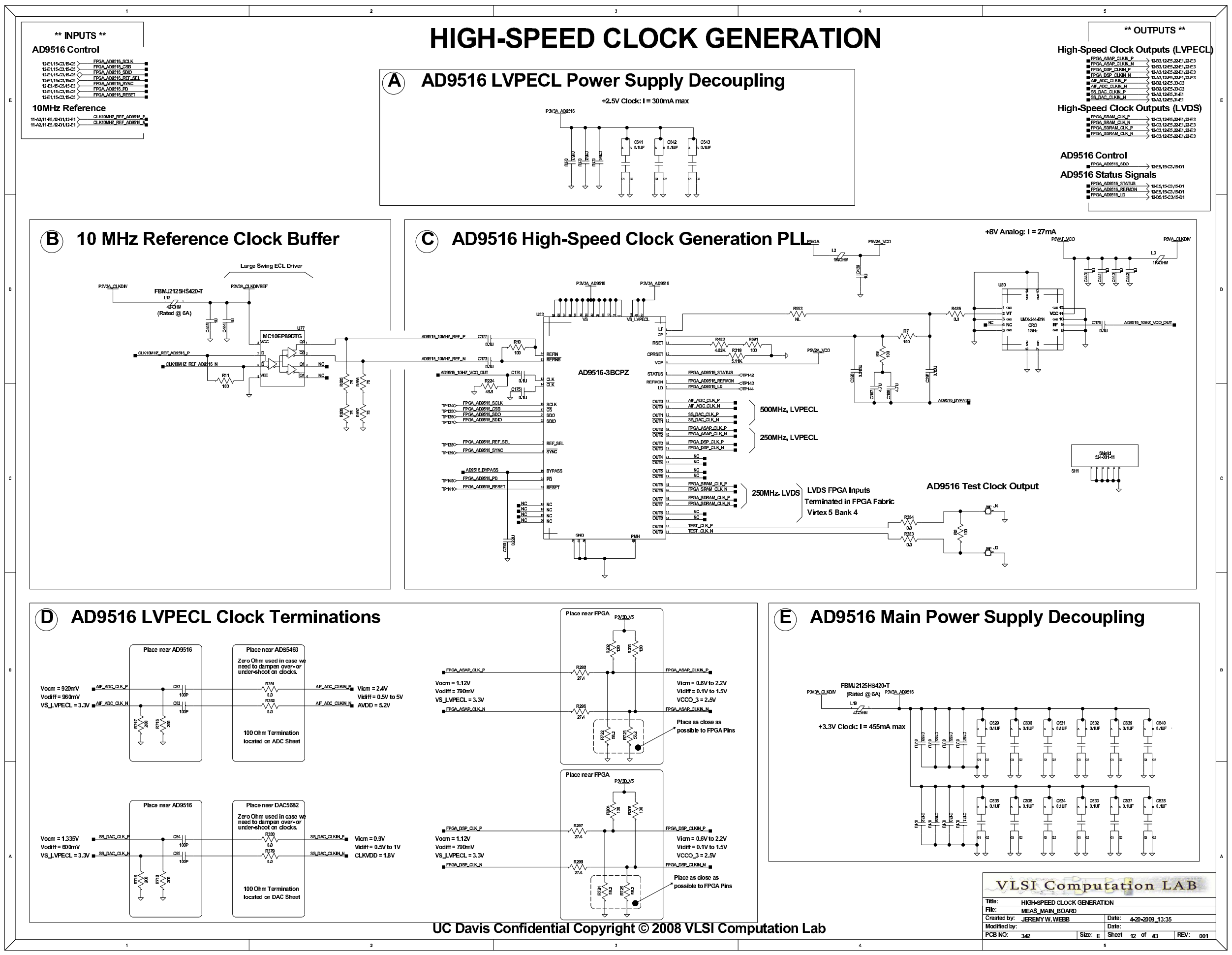
HIGH-SPEED CLOCK GENERATION
** INPUTS **
AD9516 Control
10MHz Reference
** OUTPUTS **
High-Speed Clock Outputs (LVPECL)
High-Speed Clock Outputs (LVDS)
AD9516 Control
AD9516 Status Signals
A AD9516 LVPECL Power Supply Decoupling
+2.5V Clock: I = 300mA max
B 10 MHz Reference Clock Buffer
Large Swing ECL Driver
MC10EP89DTG
C AD9516 High-Speed Clock Generation PLL
+8V Analog: I = 27mA
AD9516-3BCPZ
500MHz, LVPECL
250MHz, LVPECL
250MHz, LVDS
LVDS FPGA Inputs Terminated in FPGA Fabric Virtex 5 Bank 4
AD9516 Test Clock Output
D AD9516 LVPECL Clock Terminations
Place near AD9516
Place near AD6463
Zero Ohm used in case we need to dampen over- or under-shoot on clocks.
100 Ohm Termination located on ADC Sheet
Place near DAC5682
100 Ohm Termination located on DAC Sheet
Place near FPGA
Place as close as possible to FPGA Pins
E AD9516 Main Power Supply Decoupling
+3.3V Clock: I = 455mA max
VLSI Computation LAB
Title: HIGH-SPEED CLOCK GENERATION
File: MEAS_MAIN_BOARD
Created by: JEREMY W. WEBB
UC Davis Confidential Copyright © 2008 VLSI Computation Lab

Figure 4.1: Schematic: AD9516 Clock Generation Circuit

1. Review AD9516 differential voltage settings to optimize resulting common mode voltages on all clock outputs.
   (a) See schematics for estimated common voltages of DAC5682Z, AD9516, and Virtex-5.
      i. DAC5682Z - Measured at DAC5682Z
         A. Vicm: 0.9V (spec), 0.9V (meas)
         B. Vdiff: 0.5V to 1V (spec), 0.328V (meas)
         C. Changed Vodiff on AD9516 from 600mV to 960mV.
         D. New Measurements; Vidiff: 580mV
      ii. DAC5682Z - Measured at AD9516
         A. Vocm: 1.335V (spec), 2.09V (meas)
         B. Vodiff: 600mV (spec), 540mV (meas)
         C. Changed Vodiff on AD9516 from 600mV to 960mV.
         D. New Measurements; Vodiff: 750mV
      iii. ADS5463 - Measured at ADS5463
         A. Vicm: 2.4V (spec), 2.41V (meas)
         B. Vdiff: 0.5V to 5V (spec), 600mV (meas)
      iv. ADS5463 - Measured at AD9516
         A. Vocm: 0.920V (spec), 2.0V (meas)
         B. Vodiff: 960mV (spec), 760mV (meas)
         C. The Vocm spec is an estimated value based on Vodiff.
   (b) Verify that clocks get into Virtex-5 FPGA.
      i. The LVPECL OUT3 of the AD9516 routes successfully from the AD9516 to the Virtex-5 FPGA. OUT3 is the main clock input of the Virtex-5, and is used to generate SRAM clocks and $\frac{clk}{128}$ and $\frac{clk}{32}$.

4.4 DAC5682Z High-Speed 16-bit DAC Turn-On

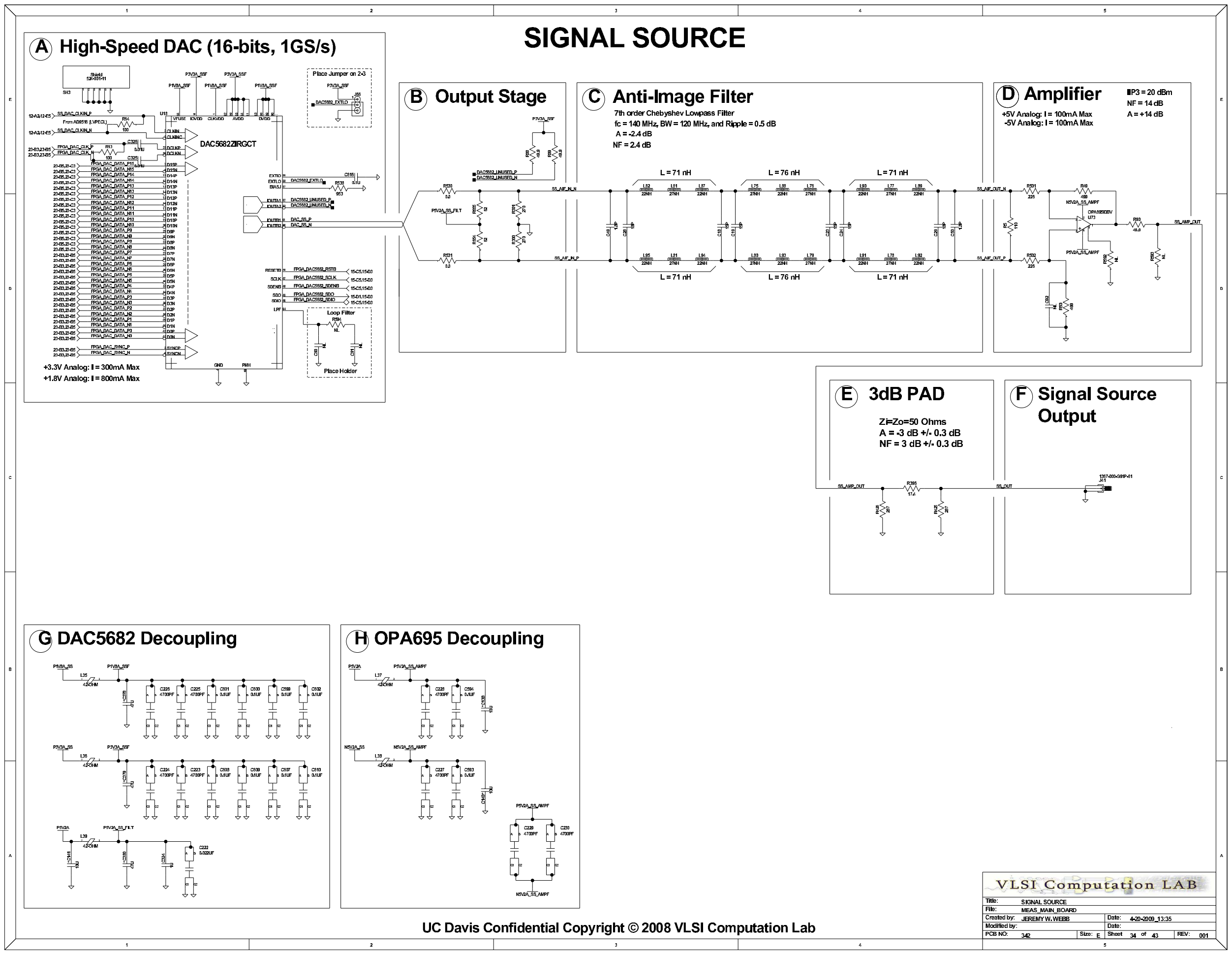

Figure 4.2: Schematic: DAC Output Circuit

1. DAC5682Z High-Speed 16-bit DAC Data Path.
   - (a) Using MicroBlaze application I could configure the DAC5682Z via SPI; however, I couldn't see any data coming out of the Channel B DAC output.
     - i. I read through my configuration settings and looked at the data sheet and discovered that my configuration settings were putting Channel B to sleep and the DAC in single-dac mode. Since my Anti-Image filter was being driven by Channel B, this explained why I wasn't seeing any data on the output. In probing the termination resistor on Channel A I was able to see data toggling. By setting the DAC5682Z to Dual DAC mode I was able to see data coming out of Channel B, but it was effectively decimated by 2. This was due to my FPGA design which was supposed to drive 8 consecutive packets of 16-bit data into the DAC rather than every other 16-bit packet of a 128-bit word out to Channel A. The DAC digital interface DDR, so it uses the 16-bit packet clocked in on the rising edge of DCLK for Channel A and the 16-bit packet clocked in on the falling edge of DCLK for Channel B.
     - ii. After reading the data sheet I discovered that when using the DAC5682Z in single-dac mode Channel A is the output. When I originally designed the schematics I was using Channel A, but during layout I changed to Channel B for layout reasons. Using Channel B meant I did not have to use vias in my analog signal path at the DAC output. Fortunately I chose to terminate the unused Channel A and used series 0 ohm resistors on the Channel B output before the Anti-Image filter. I forgot about the note in the data sheet regarding Channel A. The following PCB mod/butch was made:
       - A. Remove R88, R89, R530, and R531.
       - B. Solder a wire from pad 1 of R88 to pad 2 of R531.
       - C. Solder a wire from pad 1 of R89 to pad 2 of R530.
     - iii. This mod/butch effectively changes the DAC output from Channel B to Channel A. The wire length is approximately 3/4" long, so this shouldn't have a huge impact on the performance of my Signal Output because I'm only trying to generate signals from DC to 120MHz. If I were trying to generate signals at the full rate of the DAC, then I might be in trouble due to signal integrity effects.
   - (b) After the PCB mod/butch was made, I was able to drive data out of the DAC5682Z. I could load various patterns into either Block RAM or QDR-II SRAM and play them back to the DAC5682Z. I was able to use the DAC Sync pin to turn the DAC5682Z output on or off.
   - (c) The Signal Output waveforms driven from the OPA695 don't look too good. There's a lot of jitter and the noise floor is about -40dB. I expected the noise floor to be down at -60dB. I need to speak with Texas Instruments to see if they have any recommendations for testing the performance of this circuit, since my DAC circuit is similar to one of the TI evaluation boards.
     - i. I was able to get a nice looking waveform to come out of the DAC5682Z and the analog circuit, so I suspect to the problem lies with the resolution and sample rate of the signals being sent to the DAC5682Z. I may need to generate the signals with more precision than 16-bits, and dither the LSB bits before/after truncation occurs with a 4- to 6-bit PRBS signal.
     - ii. Another alternative for signal generation is an 8 channel DDS implementation. I'd need to design the DDS in Verilog, and this would take a week or two of design time. With a DDS I would be able to dynamically change frequency and phase without reloading a waveform into either Block RAM or QDR-II SRAM.
       - A. Leveraged a DDS from another design I created into an 8 phase DDS design. The output of the DAC5682Z didn't look much better. There could still be a timing problem with the 16-bit data bus due to the PC Board traces being routed shortest length from the Virtex-5 to the DAC5682Z. Probably need to tweak the fixed delay values in the IODELAY blocks. Measure the alignment from each data bit to the clock bit to determine the approximate setup/hold. Use one of the test patterns that provide alternating 1's and 0's.
     - iii. I implemented a routine that configures the DAC5682Z for Digital Self Test, and it passed the self test.

iv. I was also able to get a really nice looking sine wave to come out of the DAC5682Z, when I cycle through static values that are representative of a sine wave. See Figure 4.3. For example,

* R0:0000
* F0:3FFF
* R1:7FFF
* F1:3FFF
* R2:0000
* F2:C001
* R3:8001
* F3:C001

where Rx indicates rising edge of DCLK, Fx indicates falling edge of DCLK, and the x indicates the order of samples driven into the DAC5682Z with 0 being first and 3 being last. The results of this test lead me to believe that my DAC5682Z and OPA695 plus the low-pass filter are functioning properly.

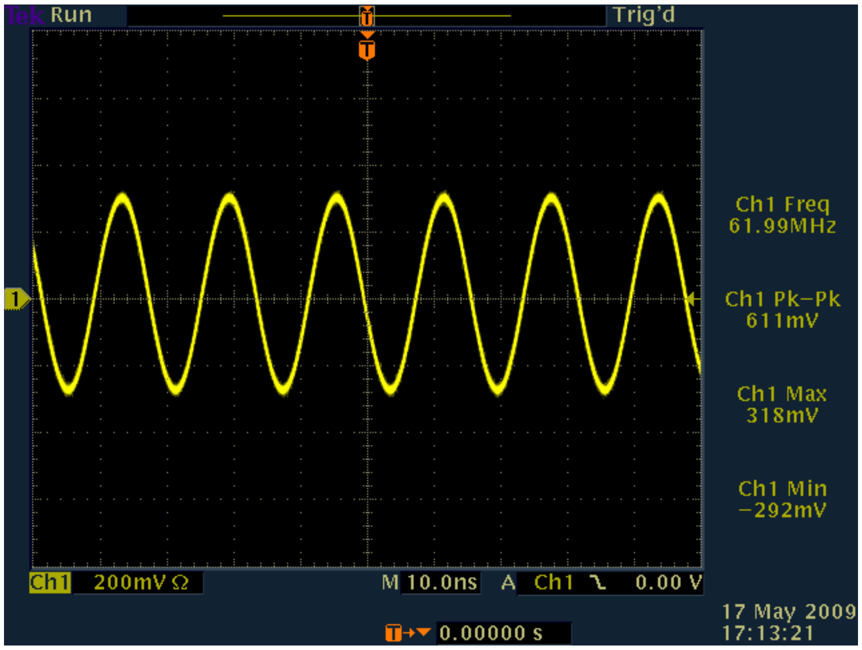

Figure 4.3: Measured: 62.5MHz Sine Wave

v. When I try to drive a DDS signal made up of 2's comp sine or cosine values I get a really bad looking waveform, which most likely means I'm having a data related problem. Especially since I get a nice looking waveform using the sequence of values listed above. I'm not sure why this is. I'm using a method similar to that described on page 40 of the DAC5682Z data sheet, except for the following:

1. I'm using 8:1 SERDES instead of a 4:1 SERDES.
2. The slow SERDES clock is 125MHz and the fast SERDES clock is 500MHz.

This gives me a data rate of 1Gb/s on the 16-bit DAC5682Z interface with a DCLK of 500MHz. My CLKIN is also 500MHz.

My setup is actually very similar to that shown in Figure 7 on page 18 of the TSW3070EVM User's Guide. My FPGA is essentially the same thing as the TSW3100 Pattern Generator, and my DAC5682Z circuit is essentially the same thing as one of the channels of the TSW3070EVM (i.e., the one with the OPA695).

vi. Today I realized that I was violating the sampling theorem. My DCLK is 500MHz, which corresponds to a data rate of 1Gb/s, and my CLKIN is 500MHz. I am effectively sampling 1Gb/s data with a 500MHz clock. There are three options for fixing this problem.

A. Set the AD9516 to bypass the VCO divider, which would send 1GHz to OUT0 and OUT1 of the AD9516. Both the DAC5682Z and the ADS5463 would receive 1GHz sampling clocks. This is a problem for the ADS5463, since it can only sample at a rate up to 500MHz. The OUT0 and OUT1 AD9516 outputs share a divider, so they must both be the same frequency.
- 28 191 c0
- 28 192 2

B. Setup the PLL in the DAC5682Z. The PLL would use an M of 4 and an N of 2. I could set the PLL to generate 2GHz, and use the VCO divide by 2 to improve the phase noise of the output. I would also need to design a loop filter for the PLL on the DAC5682Z.
- DAC5682Z Loop Filter:

  C91 (C1): 1.43uF = 1uF ‖ 0.47uF → 1.47uF
  C90 (C2): 16nF = 10nF ‖ 6.2nF → 16.2nF
  R594 (R1): 10.6 Ohms = 36 Ohm ‖ 15 Ohm → 10.5882 Ohms

  15 Ohm: ERJ-3GEYJ150V
  36 Ohm: ERJ-3GEYJ360V
  1uF: 0603X105K250G
  0.47uF: C1608Y5V1H474Z
  10nF: ECJ-1VB1H103K
  6.2nF: 06035C622JAT2A

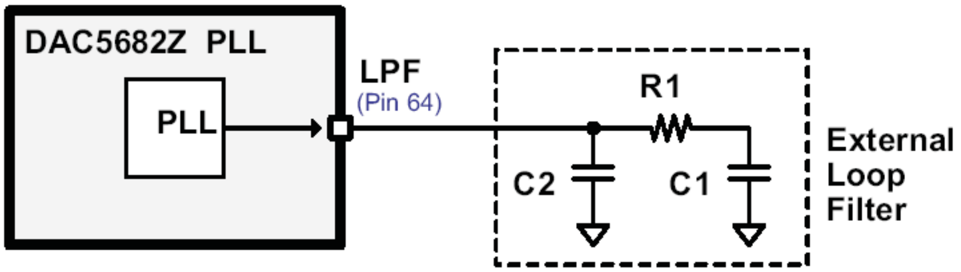

Figure 4.4: DAC5682Z Recommended External Loop Filter Topology

C. Change 128-bit bus clock from 125MHz to 62.5MHz. This also changes the DDR data rate to the DAC from 1Gb/s to 500Mb/s. The OSERDES fast clock changes from 500MHz to 250MHz. This allows the data path fpga to still be 128-bits, or 8-lanes of 16-bits. And the OSERDES are still configured as 8:1.

vii. After fixing the sampling clock of the DAC5682Z, the frequency of the static sine wave is now 125MHz. This makes sense, because I'm using 125MHz to generate the data inside the FPGA. After sending a real sine waveform to the DAC5682Z from the QDR-II SRAM the data looks better, but it still has a fair amount of jitter. I need to verify the sampling frequency at the DAC5682Z with a better oscilloscope. My scope only has 500MHz of bandwidth. I can barely see the 1GHz signal with my 1GHz active probe.

viii. The DAC5682Z has a DLL, which is driven by the DCLK. The DLL is used to skew the DCLK and Data, so that the 16-bit data is optimally sampled as it is driven into the 8 sample FIFO of the DAC5682Z. The DLL has a frequency range setting that needs to be adjusted based on the the DCLK frequency. After properly adjusting the DLL config register I was able to get the DAC5682Z DLL to lock reliably.

ix. I simulated the FPGA design to check the 16-bit DAC Data and Clock. I discovered that the DCLK was inverted. After fixing the clock polarity, the signal out of the DAC looks about the same.

x. It looks as though the DAC is sampling the data incorrectly. I tested the interface with various combinations of DCLK and CLKIN rates, but I have not been successful at getting signals of varying frequencies to be output from the DAC5682Z.

xi. Another problem I discovered in my current system architecture is the relationship between DCLK and CLKIN. At present, my DCLK was generated from an on-board 100MHz crystal oscillator, and the CLKIN was generated by the AD9516. The DCLK and CLKIN really need to be generated from a common clock, otherwise the DAC5682Z will have difficulty sampling the digital data.

   A. To fix this problem I used one of the 4 differential clock pairs from the AD9516 to the FPGA as the 100MHz main clock. Now the DCLK and CLKIN is generated from a common 1GHz VCO connected to the AD9516. The DCLK is actually generated by a PLL inside the FPGA, but the PLL's main input clock is the 100MHz clock from the AD9516.

xii. Once the DCLK and CLKIN were generated from a common clock the DAC5682Z still looks incorrect, but is approximately 20MHz which was the intended frequency of the waveform file being played back from QDR-II SRAM. I decided to simulate the waveform playback and store the 16-bit DAC Data in a Matlab data file. Plotting the waveform in the time domain revealed the incorrect waveform shown in Figure 4.5.

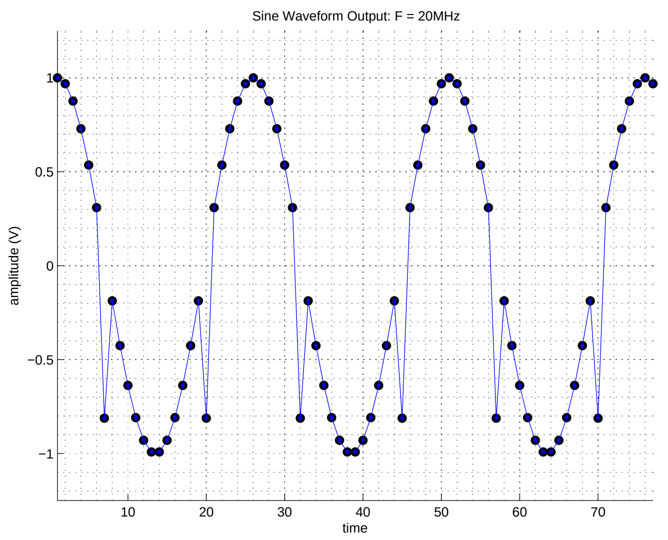

Figure 4.5: Verilog Simulation: 20MHz Sine Wave with bad sample point

xiii. The waveform shown in Figure 4.5 led me to believe the problem was with the waveform generation. The method I used to generate the 20MHz sine waveform was:
- Write Matlab script to generate desired waveform with appropriate sample rate.
- Use Matlabe script to write single cycle of waveform to a text file in signed 2's complement decimal.
- Use a Perl script to convert signed 2's complement decimal to 16-bit hexadecimal and replicate to a 128-bit boundary.
  - This is where the sample point was being corrupted:

```
Numerator (rounded to nearest integer): 2057
Quantized Data to 16-bits resolution: 0.062774658203125
Quantized Data (Hex): 808
Number of hex digits: 3
Grabbing bits -1 to 2
Quantized Data (Hex, 4-digits): 8808
```

    The hexadecimal representation was less than 3 digits, and the Perl script assumed the hexadecimal data contained 4 digits. The Perl script was grabbing data from -1 to 2 instead of 0 to 3. The sample's LSB digit was copied to the MSB position resulting in a negative value instead of a small positive number (i.e., 8808 instead of 0808). This explains the waveform shown in Figure 4.5. To fix this problem I created a Perl sub-routine to check each hexadecimal value for at least 4 digits. If the hexadecimal value contained less than 4 digits it would pad the value with the appropriate number of zeros.

xiv. Upon fixing the waveform generation Perl script I was able to get proper waveforms out of the DAC5682Z. Figure 4.6 shows the correct waveform. Figure 4.7 shows the incorrect and correct waveforms plotted on the same axes.

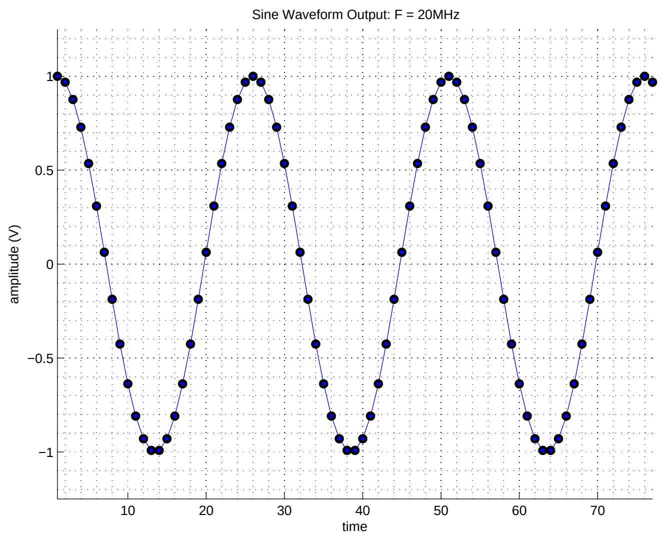

Figure 4.6: Matlab Simulation: 20MHz Sine Wave

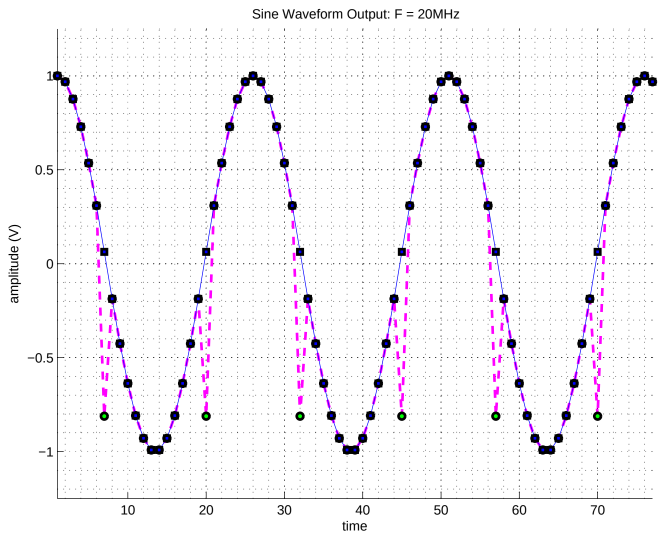

Figure 4.7: Matlab Simulation: 20MHz Sine Wave

xv. Figures 4.9 and 4.8 show oscilloscope screen shots of the signals measured at the OPA695 output.

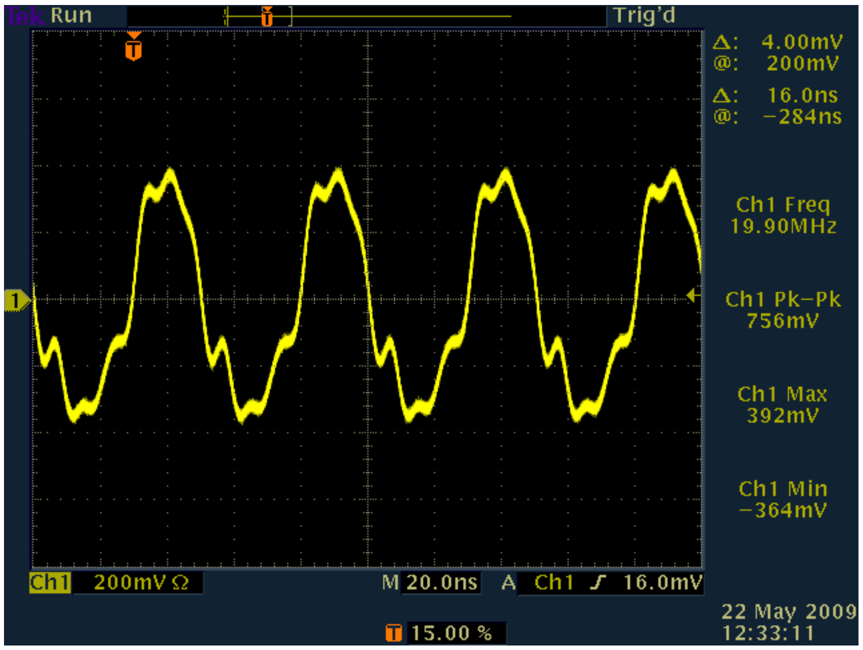

Figure 4.8: Measured: 20MHz Sine Wave

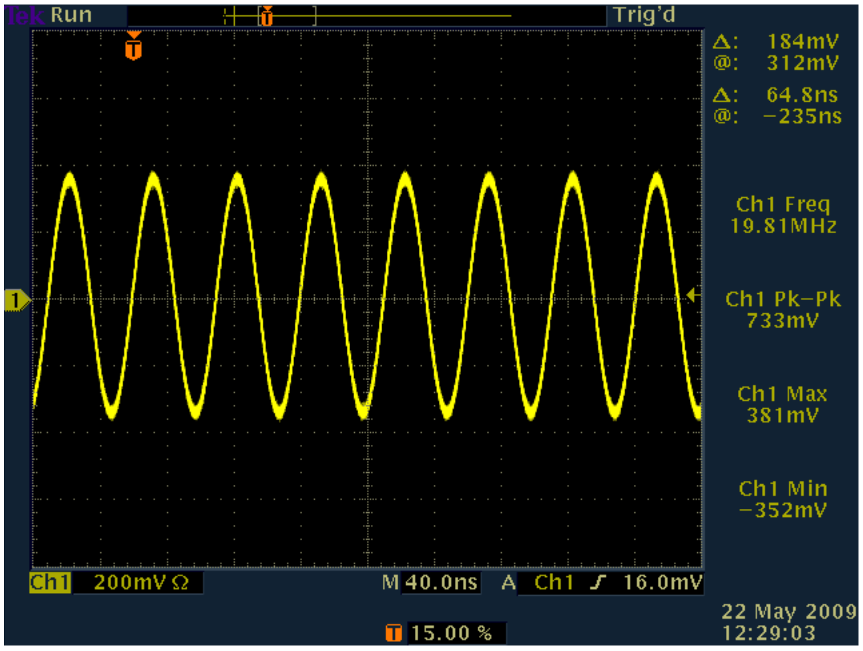

Figure 4.9: Measured: 20MHz Sine Wave

4.5 ADS5463 High-Speed 16-bit ADC Turn-On

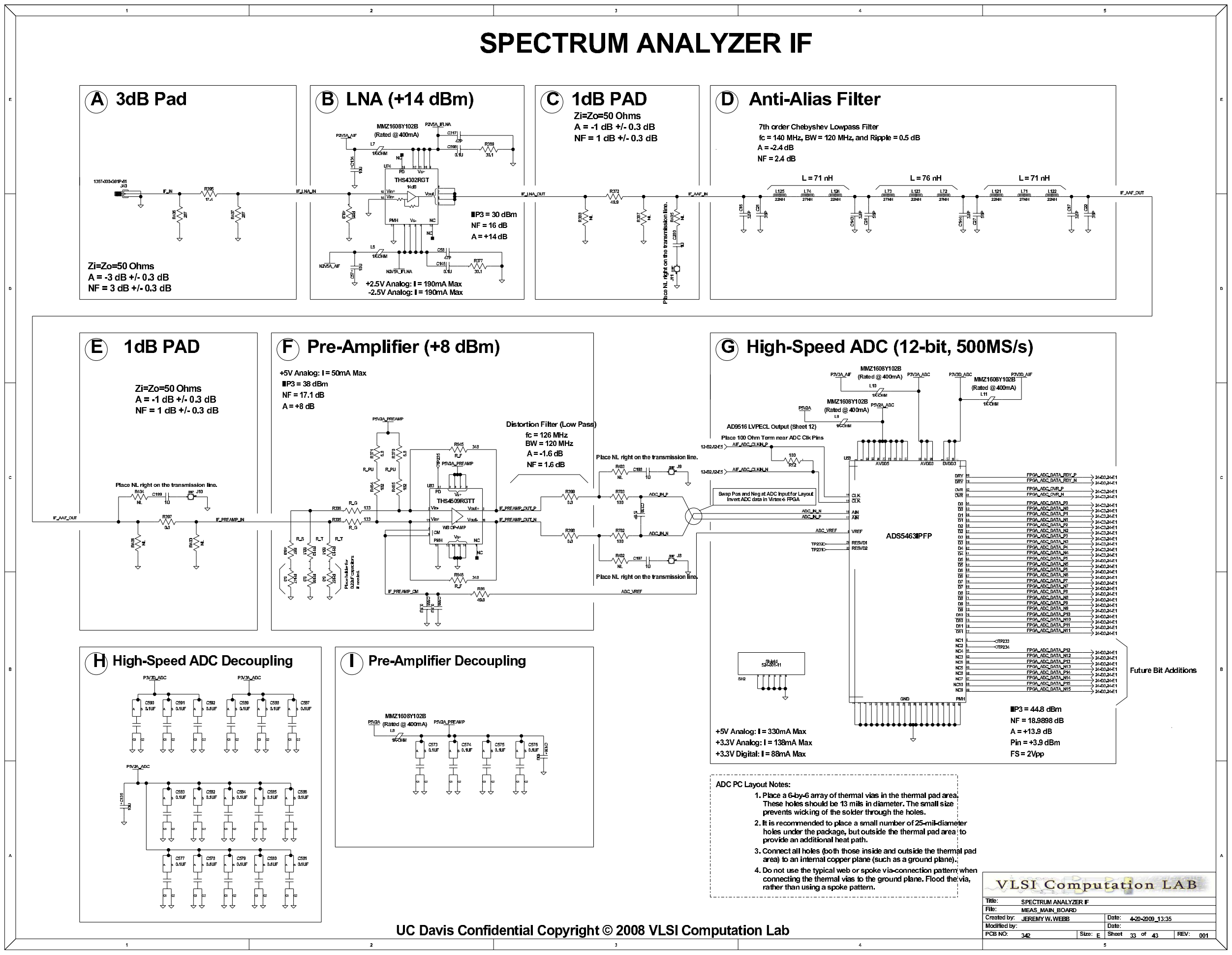
SPECTRUM ANALYZER IF
A 3dB Pad
Zi=Zo=50 Ohms
A = -3 dB +/- 0.3 dB
NF = 3 dB +/- 0.3 dB
B LNA (+14 dBm)
IIP3 = 30 dBm
NF = 16 dB
A = +14 dB
+2.5V Analog: I = 190mA Max
-2.5V Analog: I = 190mA Max
C 1dB PAD
Zi=Zo=50 Ohms
A = -1 dB +/- 0.3 dB
NF = 1 dB +/- 0.3 dB
D Anti-Alias Filter
7th order Chebyshev Lowpass Filter
fc = 140 MHz, BW = 120 MHz, and Ripple = 0.5 dB
A = -2.4 dB
NF = 2.4 dB
L = 71 nH
L = 76 nH
L = 71 nH
E 1dB PAD
Zi=Zo=50 Ohms
A = -1 dB +/- 0.3 dB
NF = 1 dB +/- 0.3 dB
F Pre-Amplifier (+8 dBm)
+5V Analog: I = 50mA Max
IIP3 = 38 dBm
NF = 17.1 dB
A = +8 dB
Distortion Filter (Low Pass)
fc = 126 MHz
BW = 120 MHz
A = -1.6 dB
NF = 1.6 dB
G High-Speed ADC (12-bit, 500MS/s)
ADS5463IPFP
Future Bit Additions
IIP3 = 44.8 dBm
NF = 18.9898 dB
A = +13.9 dB
Pin = +3.9 dBm
FS = 2Vpp
+5V Analog: I = 330mA Max
+3.3V Analog: I = 138mA Max
+3.3V Digital: I = 88mA Max
H High-Speed ADC Decoupling
I Pre-Amplifier Decoupling
ADC PC Layout Notes:
1. Place a 6-by-6 array of thermal vias in the thermal pad area. These holes should be 13 mils in diameter. The small size prevents wicking of the solder through the holes.
2. It is recommended to place a small number of 25-mil-diameter holes under the package, but outside the thermal pad area to provide an additional heat path.
3. Connect all holes (both those inside and outside the thermal pad area) to an internal copper plane (such as a ground plane).
4. Do not use the typical web or spoke via-connection pattern when connecting the thermal vias to the ground plane. Flood the via, rather than using a spoke pattern.
VLSI Computation LAB
Title: SPECTRUM ANALYZER IF
File: MEAS_MAIN_BOARD
Created by: JEREMY W. WEBB
Date: 4-20-2009_13:35
UC Davis Confidential Copyright © 2008 VLSI Computation Lab

Figure 4.10: Schematic: ADC Input Circuit

1. Now that the DAC5682Z is functioning correctly I was able to loop the DAC output back into the ADC. The signal looks good at the output of the THS4302 Op-Amp, Anti-Alias Filter, and THS4509 Differential Op-Amp. Figure 4.11 shows the signal measured at the input of the ADS5463 High-Speed 16-bit ADC.

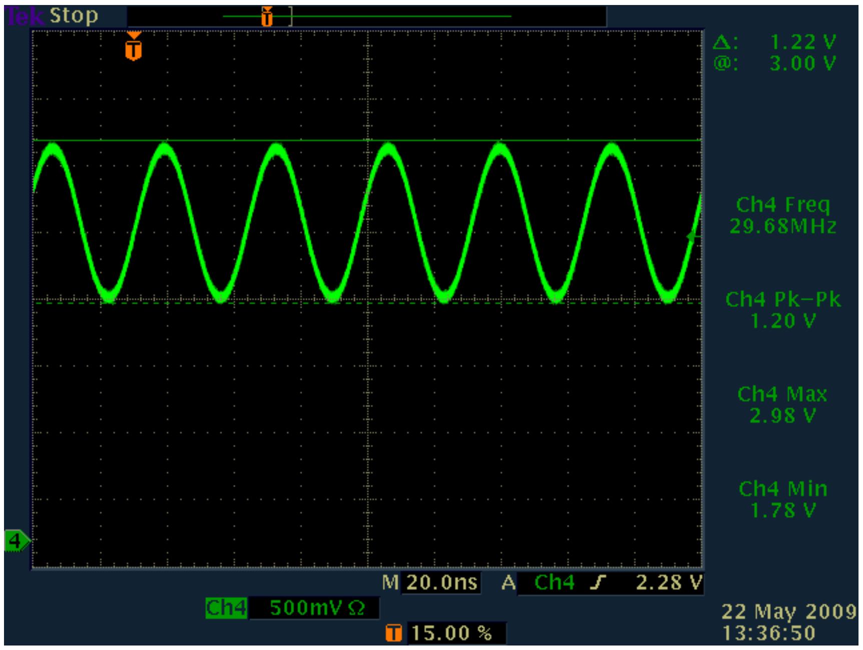

Figure 4.11: Measured: ADS5463 Input Signal 30MHz Sine Wave

2. Designed a function to initiate a waveform capture in the Data Path FPGA and transfer the data from the capture RAM into the MicroBlaze logfile. The capture RAM can hold 16,384*8 = 131,072 samples from the ADC.

   (a) The data from the ADC is presented in bipolar offset binary (BOB) coded format. The MSB bit of the 12-bit ADC data can be inverted to convert the data from BOB coded format to bipolar two's complement (BTC) coded format. See Table 4.1. Note: digital zero *0000* corresponds to Bipolar Zero (BPZ).

Table 4.1: Bipolar Offset Binary to Bipolar Two's Complement Conversion

MNEMONIC	**DIGITAL CODE**	
	BOB	**BTC**
-FS	0000	1000
	0001	1001
	0010	1010
	0011	1011
$\frac{1}{2}$-FS	0100	1100
	0101	1101
	0110	1110
BPZ - 1V$_{LSB}$	0111	1111
BPZ	1000	0000
BPZ + 1V$_{LSB}$	1001	0001
	1010	0010
	1011	0011
$\frac{1}{2}$+FS	1100	0100
	1101	0101
	1110	0110
+FS	1111	0111

   (b) Prior to converting the data from BOB to BTC, the ADC data must be inverted because the positive and negative differential ADC input signals were swapped to eliminate the need for vias in the ADC signal path. The Data Path FPGA performs the ADC data inversion and conversion from BOB to BTC in real time.

   (c) When the command 'adc' is entered at the debug prompt, 131,072 samples are transfered from the Data Path FPGA to the logfile stored in the DDR SDRAM. After the transfer is complete, the logfile can be copied from the DDR SDRAM 8MB RAM Disk to the 2GB microSD Card using the command 'logsave'.

3. Created a Matlab script to post-process the ADC capture data. The Matlab script performs the following post-processing steps:

   (a) Plot Time Domain of Sampled Waveform
   (b) Plot FFT of Sampled Waveform: 0 to 250MS/s
   (c) Low-Pass FIR Filter and Decimate-by-2 the Sampled Waveform
   (d) Plot FFT of Decimate-by-2 waveform: 0 to 125MS/s
   (e) Low-Pass FIR Filter and Decimate-by-2 the Sampled Waveform
   (f) Plot FFT of Decimate-by-2 waveform: 0 to 62.5MS/s
   (g) Low-Pass FIR Filter and Decimate-by-2 the Sampled Waveform
   (h) Plot FFT of Decimate-by-2 waveform: 0 to 31.25MS/s

4. Drove ADC with the following waveforms:

   (a) Sine Waveform @ F = 30MHz

      i. Sine Waveform @ F = 30MHz; Time Domain (See Figure 4.12).

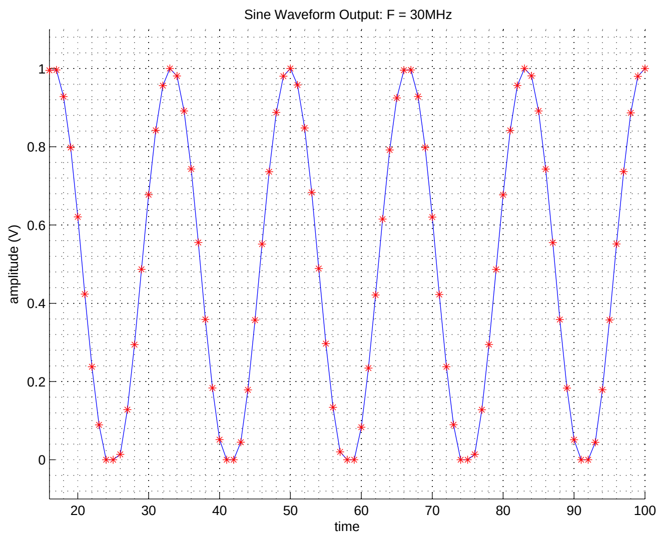

Figure 4.12: Measured: ADS5463 Digitized 30MHz Sine Wave; Time Domain

      ii. Sine Waveform @ F = 30MHz; Frequency Domain (See Figure 4.13.)

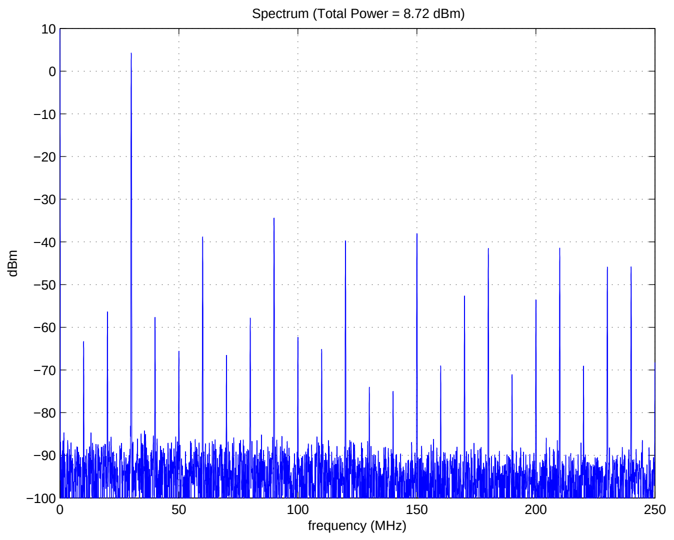

Figure 4.13: Measured: ADS5463 Digitized 30MHz Sine Wave; Frequency Domain

iii. Sine Waveform @ F = 30MHz; Decimate-by-8; Frequency Domain (See Figure 4.14.)

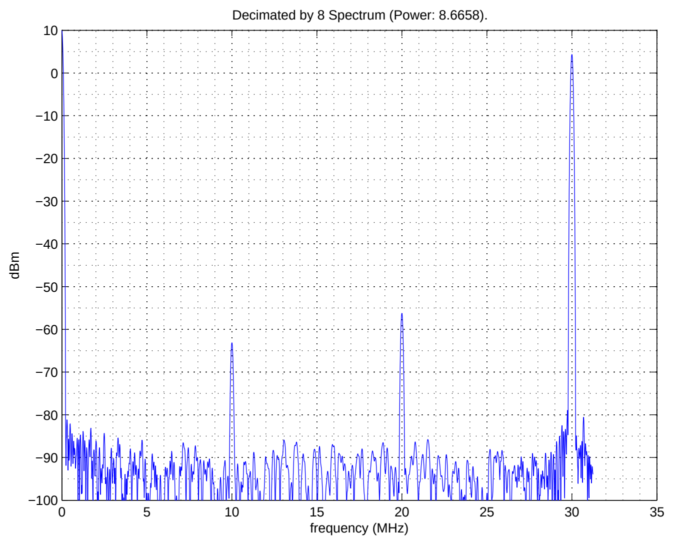

Figure 4.14: Measured: ADS5463 Digitized 30MHz Sine Wave; Decimate-by-8; Frequency Domain

(b) Sine Waveform @ F = 40MHz

i. Sine Waveform @ F = 40MHz; Time Domain (See Figure 4.15.)

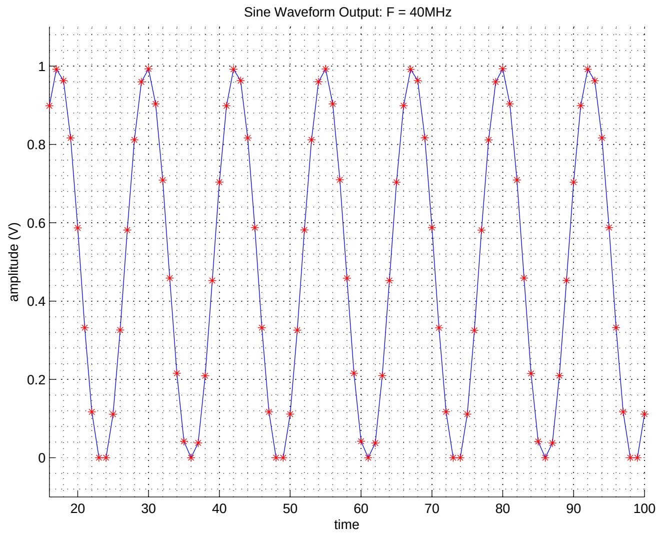

Figure 4.15: Measured: ADS5463 Digitized 40MHz Sine Wave; Time Domain

ii. Sine Waveform @ F = 40MHz; Frequency Domain (See Figure 4.16.)

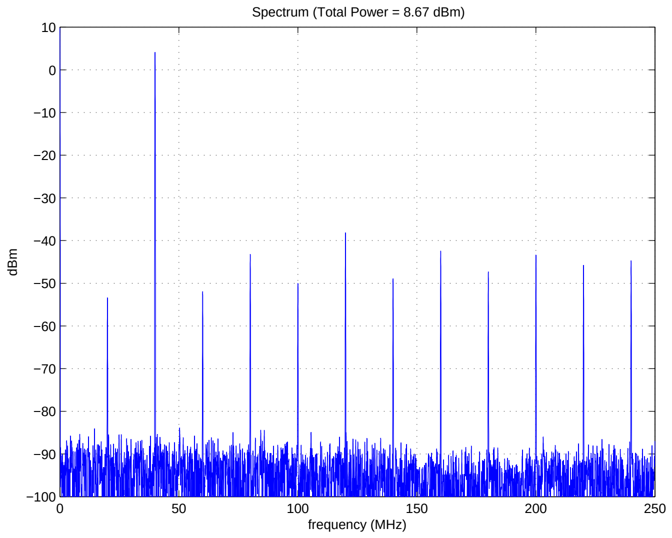

Figure 4.16: Measured: ADS5463 Digitized 40MHz Sine Wave; Frequency Domain

iii. Sine Waveform @ F = 40MHz; Decimate-by-4; Frequency Domain (See Figure 4.17.)

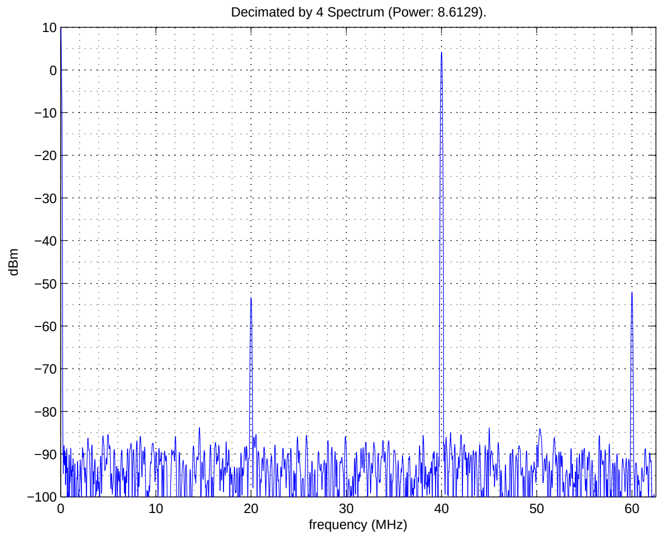

Figure 4.17: Measured: ADS5463 Digitized 40MHz Sine Wave; Decimate-by-4; Frequency Domain

(c) Sine Waveform @ F = 80MHz

i. Sine Waveform @ F = 80MHz; Time Domain (See Figure 4.18.)

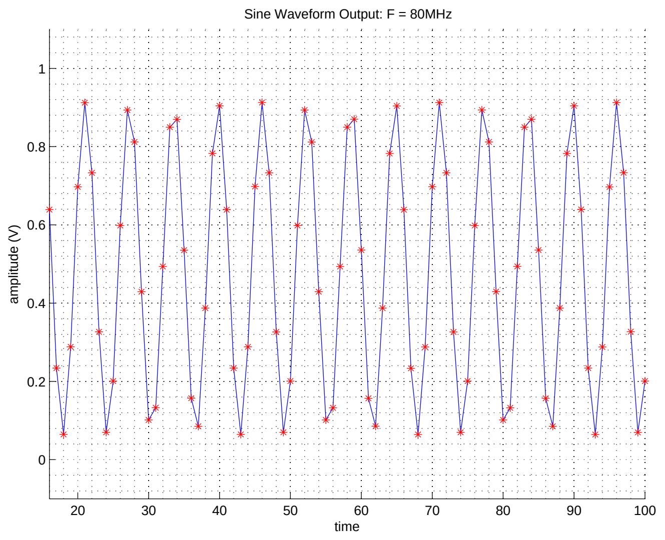

Figure 4.18: Measured: ADS5463 Digitized 80MHz Sine Wave; Time Domain

ii. Sine Waveform @ F = 80MHz; Frequency Domain (See Figure 4.19.)

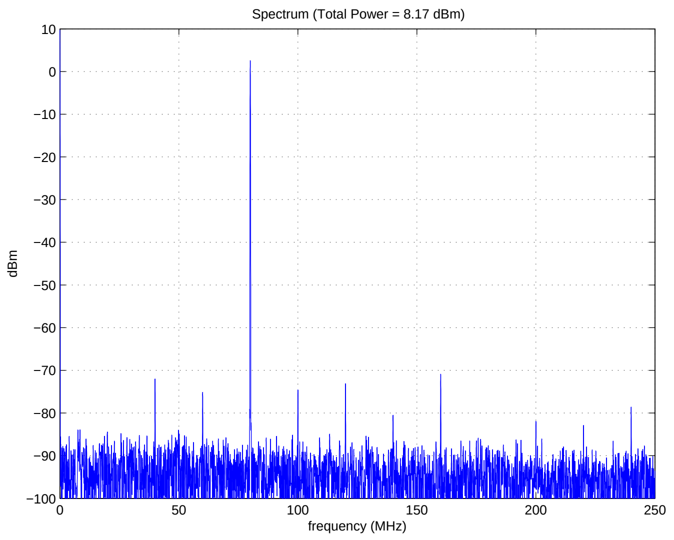

Figure 4.19: Measured: ADS5463 Digitized 80MHz Sine Wave; Frequency Domain

iii. Sine Waveform @ F = 80MHz; Decimate-by-2; Frequency Domain (See Figure 4.20.)

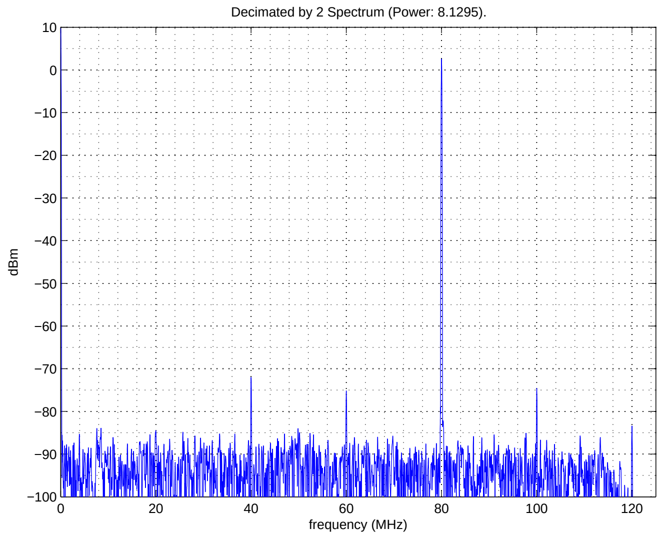

Figure 4.20: Measured: ADS5463 Digitized 80MHz Sine Wave; Decimate-by-2; Frequency Domain

(d) Sine Waveform @ F = 110MHz

i. Sine Waveform @ F = 110MHz; Time Domain (See Figure 4.21.)

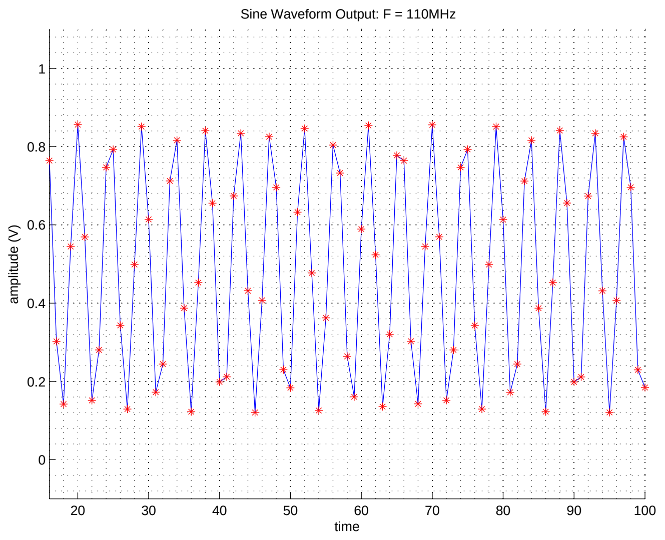

Figure 4.21: Measured: ADS5463 Digitized 110MHz Sine Wave; Time Domain

ii. Sine Waveform @ F = 110MHz; Frequency Domain (See Figure 4.22.)

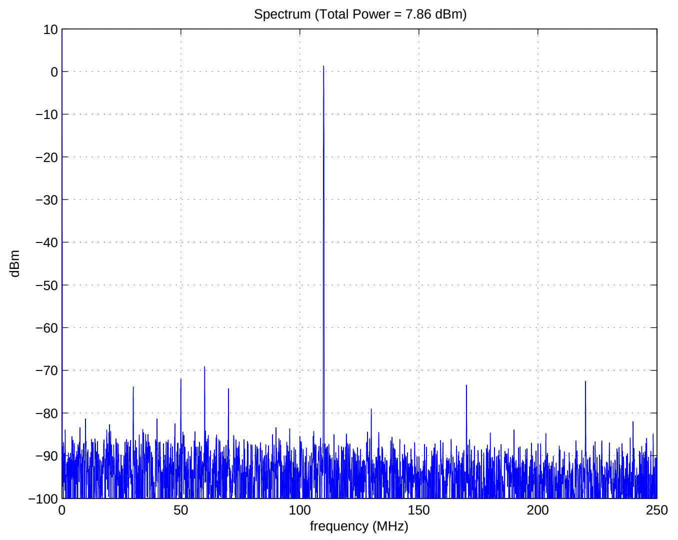

Figure 4.22: Measured: ADS5463 Digitized 110MHz Sine Wave; Frequency Domain

iii. Sine Waveform @ F = 110MHz; Decimate-by-2; Frequency Domain (See Figure 4.23.)

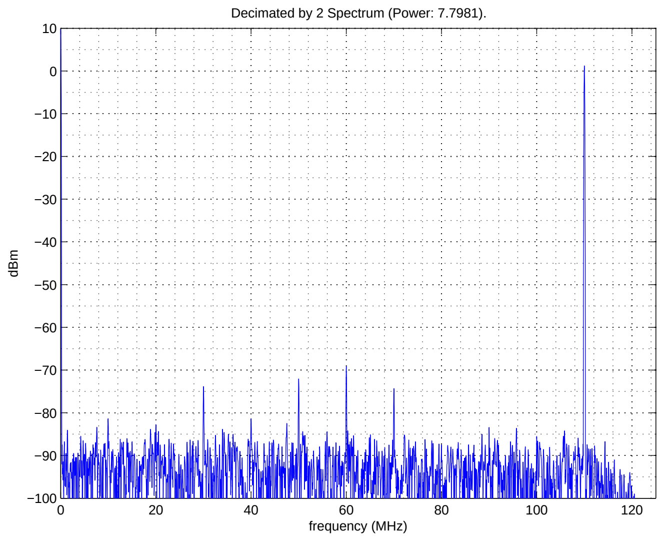

Figure 4.23: Measured: ADS5463 Digitized 110MHz Sine Wave; Decimate-by-2; Frequency Domain

(e) Waveform made up of Sine Waveforms @ F = 80MHz, 90MHz, and 100MHz

i. Sine Waveform @ F = 80MHz, 90MHz, and 100MHz; Time Domain (See Figure 4.24.)

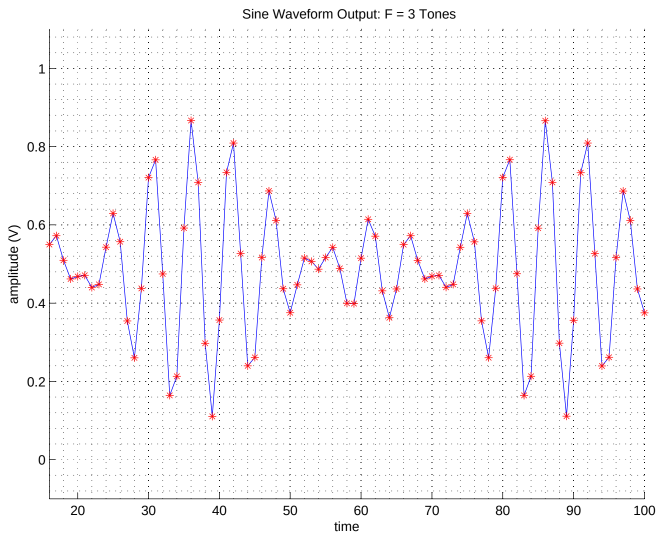

Figure 4.24: Measured: ADS5463 Digitized 80MHz, 90MHz, and 100MHz Sine Wave; Time Domain

ii. Sine Waveform @ F = 80MHz, 90MHz, and 100MHz; Frequency Domain (See Figure 4.25.)

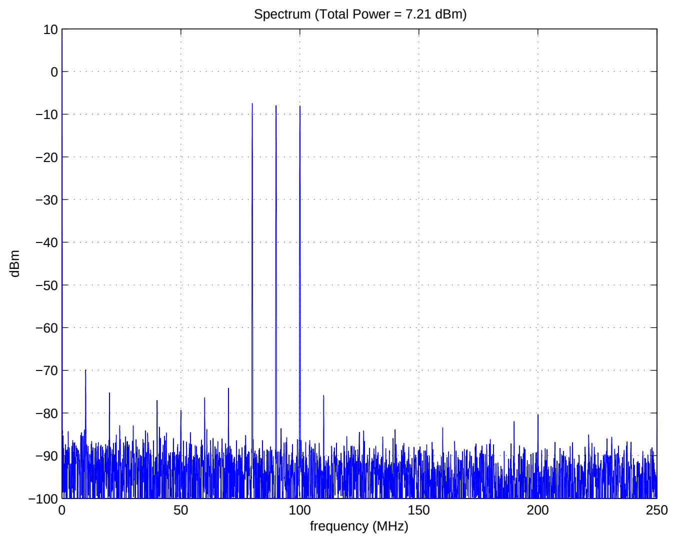

Figure 4.25: Measured: ADS5463 Digitized 80MHz, 90MHz, and 100MHz Sine Wave; Frequency Domain

iii. Sine Waveform @ F = 80MHz, 90MHz, and 100MHz; Decimate-by-2; Frequency Domain (See Figure 4.26.)

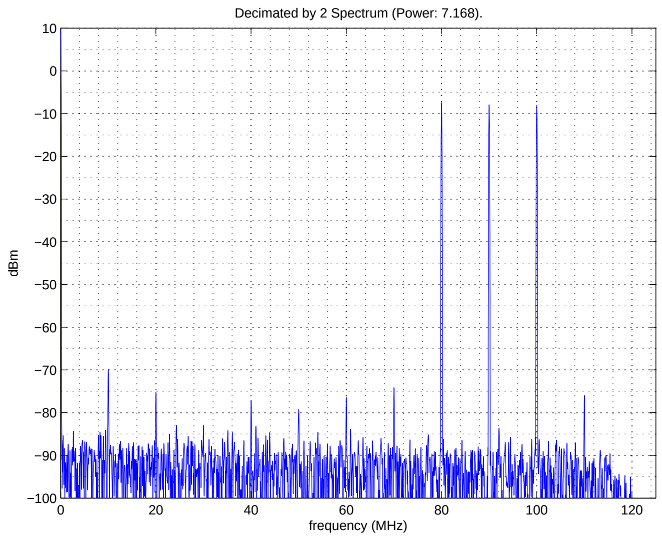

Figure 4.26: Measured: ADS5463 Digitized 80MHz, 90MHz, and 100MHz Sine Wave; Decimate-by-2; Frequency Domain

5. The spurs at the lower rates are due to signal clipping at the ADC input. I suspect that the attenuation in the Anti-Alias filter is less severe at those frequencies, so the signal at the ADC is a bit bigger. If you look at the time domain waveforms in Figures 4.12, 4.15, 4.18, and 4.21 you'll notice that the higher frequency sine waves have a smaller amplitude than the lower frequency sine waves. This was not immediately obvious to me at first glance, and after talking with a friend we determined the cause was due to signal clipping. A hard-clipped signal results in the introduction of many high frequency harmonics during the sampling process. This Wikipedia entry has a good description in the last sentence of the first paragraph of the effect I was seeing:

   - http://en.wikipedia.org/wiki/Clipping_(signal_processing)

   Also, the ADC has an over range indicator, and it was high during several of the ADC samples at the lower frequencies but never for the higher frequencies. I'm going to do some tests with an attenuator on the input to see how the spectrum of the lower frequency sine waves changes.

4.6 Trigger Output Turn-On

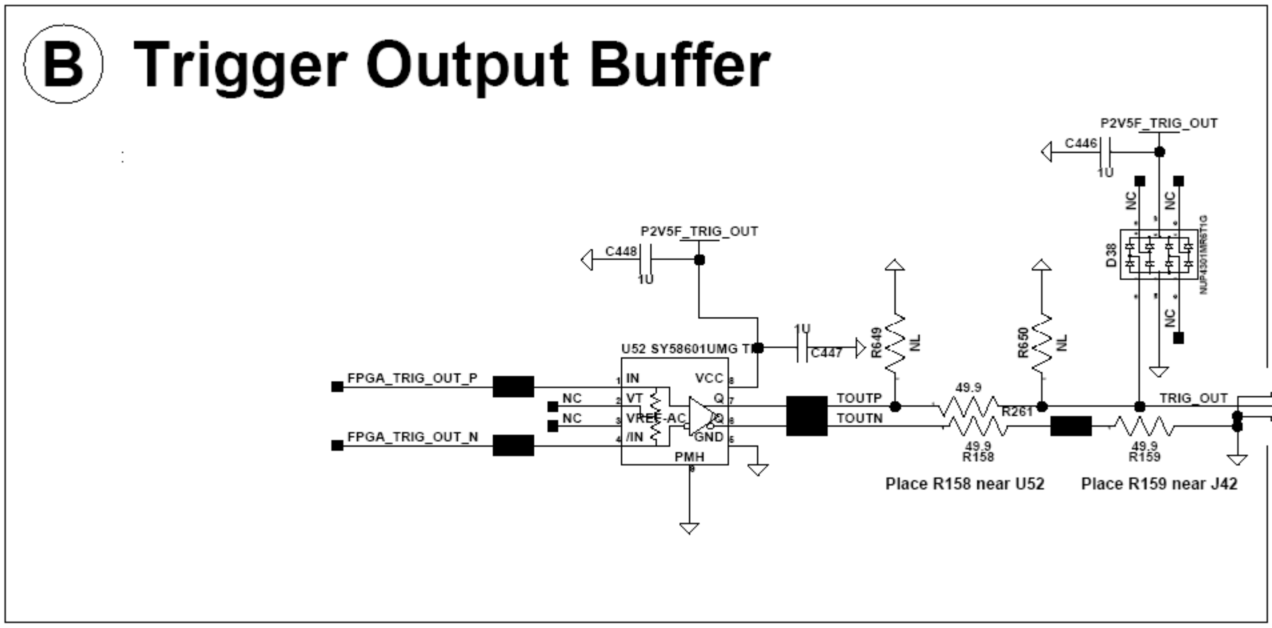

Figure 4.27: Schematic: Trigger Output Buffer

1. Take measurements of Trigger Output, Trigger Input, and Auxiliary Input.
   (a) Trigger Output Signal seems to function properly based on measurements taken with a 500MHz scope. Further investigation with a higher bandwidth scope are required to determine if any signal integrity problems exist on the Trigger Output Signal.
      i. The Pattern Trigger functions properly.
      ii. The Divided Clock Trigger functions properly at clk, $\frac{clk}{2}$, $\frac{clk}{4}$, and $\frac{clk}{8}$.

4.7 Trigger Input Turn-On

4.8 Auxiliary Input Turn-On

5 Modifications

5.1 Modifications List

The following modifications are necessary for the p342 to function properly:

M1: Load the following Data Path FPGA Configuration Resistors

R569: CUST_INIT_B

R568: CUST_CFG_DONE

R567: CUST_CCLK

R566: CUST_PROG_B

M2: DAC5682Z Channel A/B Swap:

(a) Remove R88, R89, R530, and R531.

(b) Solder a wire from pad 1 of R88 to pad 2 of R531.

(c) Solder a wire from pad 1 of R89 to pad 2 of R530.

5.2 Board Status

The status of each p342 board is listed below:

SN1.1: (a) Load resistors as described in M1. (Completed)

(b) C15 package (0508) was standing on its edge, rather than lying flat. (Fixed)

SN1.2: (a) Load resistors as described in M1. (Completed)

SN1.3: (a) Load resistors as described in M1. (Completed)

(b) J40 SMA Center Pin had a cracked solder joint at the trace landing.

i. Reflowed center pin to make good contact.

(c) U45 VDD pin 5 was not soldered down (i.e., floating).

i. Temporarily pressed down to make contact.

ii. Re-solder pin to make good connection.

SN1.4: (a) Load resistors as described in M1. (Completed)

(b) Perform DAC5682Z Channel A/B swap as described in M2.

SN1.5: (a) Load resistors as described in M1. (Completed)

SN1.6: (a) Load resistors as described in M1. (To Be Performed)

6 PCB Fabrication Problems

The following fabrication probleems were present on the p342 board:

1. Incorrect decal used for U42 (DDR2 SDRAM SODIMM).
    - Designed interposer board (p389) to fix decal.