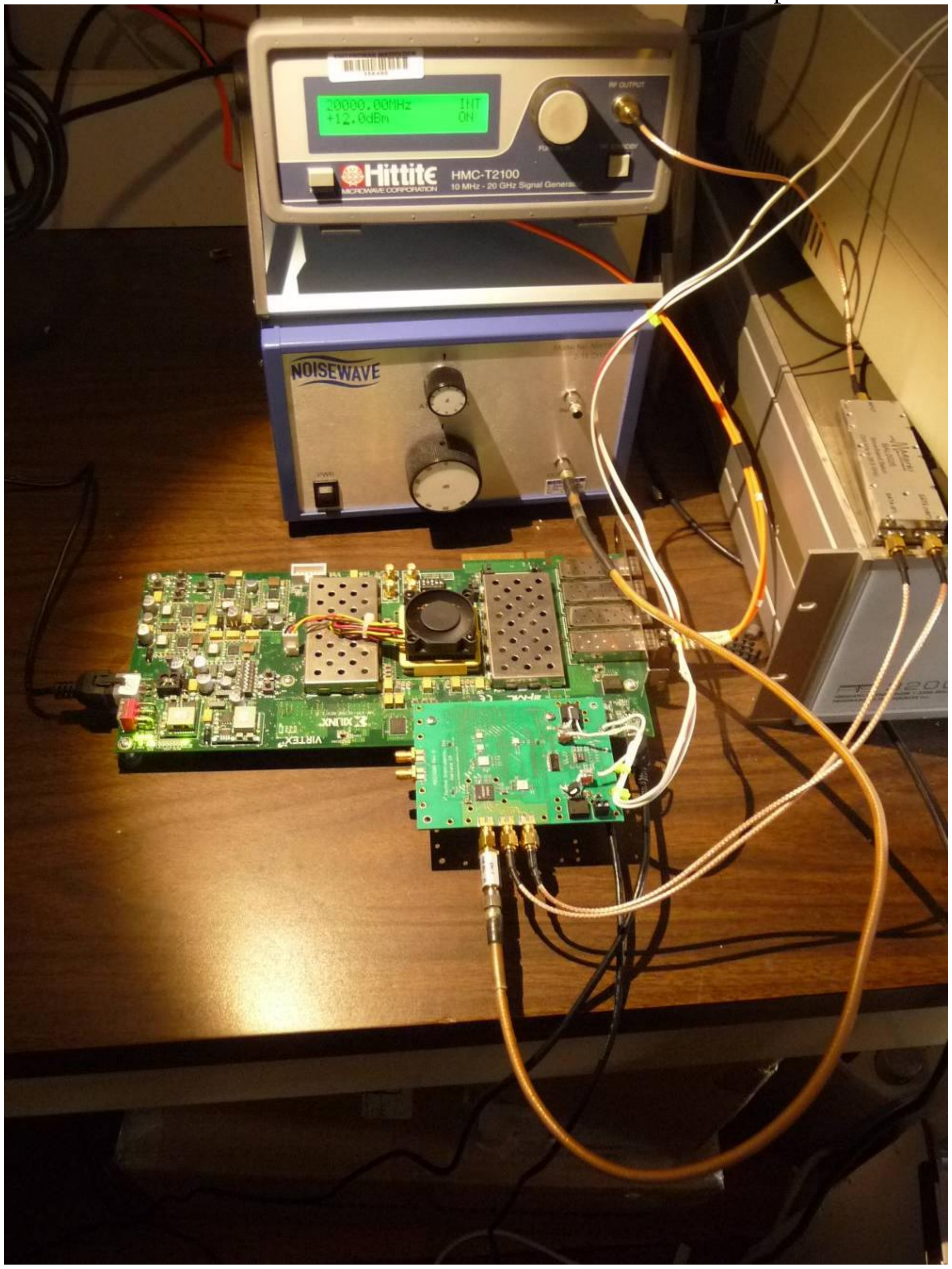
ADC1X26G Test Report

Rick Raffanti
Techne Instruments, Inc
rikraf@techneinstruments.com

Jonathan Weintroub
jweintroub@cfa.harvard.edu

October 30, 2014

This report summarizes the performance of the first assembled unit of the ADC1X26G ADC board interfaced to the VC709 Virtex7 development board.



System Overview

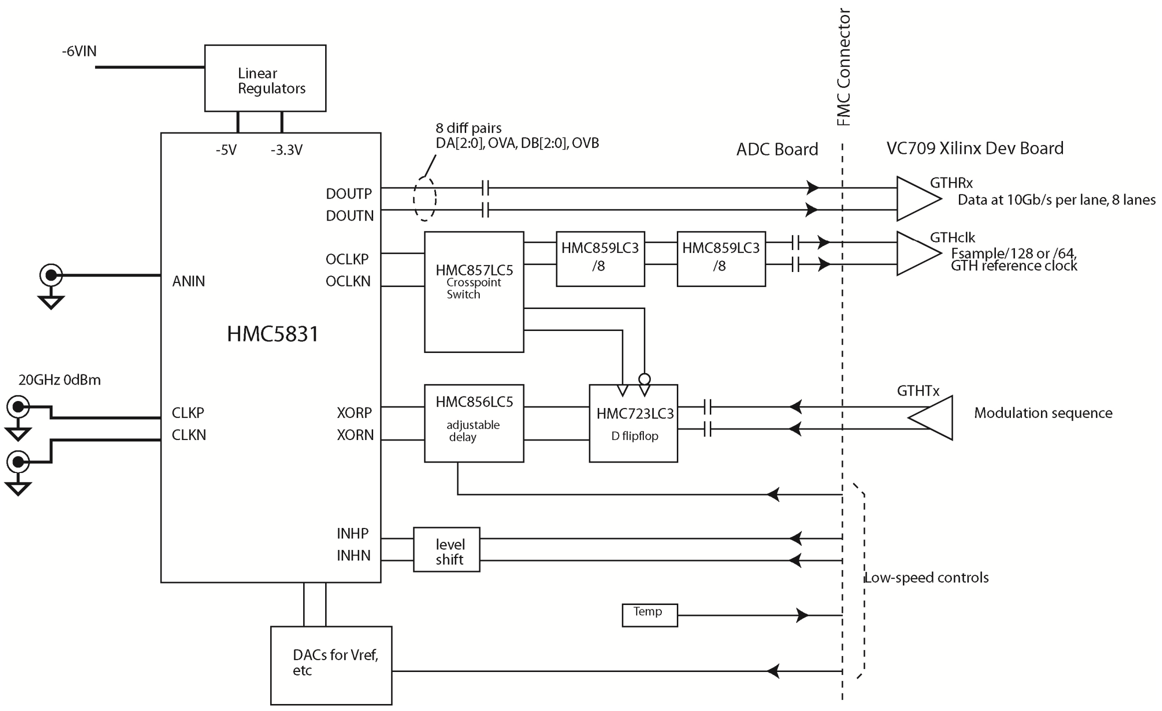
The ADC1X26G system consists of a Hittite HMC5913 26GSPS, 3.5 bit ADC and support circuitry interfaced to a VC709 FPGA development board from Xilinx. The VC709 board carries a XC7VX690T FPGA which contains 80 GTH Multi-Gigibit Serial Transceivers. The GTH Transceiver is a sophisticated subsystem capable of transmitting and receiving serial data at rates up to 13.1 Gb/s per channel (though the -2 speed FPGA on the VC709 is limited to 11Gb/s operation). The transmitter and receiver each have several functional blocks that are critical for reliable reception of data from the Hittite ADC.

The Hittite ADC is a "3 plus" bit resolution part, outputting 3 bits plus an overrange bit, so encoding a total of 10 analog levels. The digital output is demultiplexed into two 4-bit paths, with each path transmitting data at half the sample rate, or 10Gb/s for a 20GSPS conversion rate. Eight GTH Receiver channels are required to receive the data.

The ADC has an XOR input, which allows the user to multiply the digital data out of the chip by a modulating signal, to ensure there are enough data transitions for successful clock recovery. An INHIBIT input to the ADC forces the ADC data to zero, causing the modulation sequence to appear at all ADC output pairs.

ADC Board Overview

A block diagram of the ADC board design is shown here:



A critical requirement for achieving error-free data transmission from the ADC to the FPGA is the use of a modulation sequence with which the eight serial data streams are

XOR-ed internal to the ADC chip. The GTH receivers detect the clock embedded in the data stream by sensing the data edges; for this clock detection system to achieve and retain lock, the data edges must occur sufficiently often. One GTH transmitter is used to provide this modulation sequence in a very flexible manner.

The HMC723 D flipflop is provided to synchronize the incoming modulation stream to the Fsample/2 clock provided by the ADC. This relaxes the timing requirement of the modulation stream from the 20GHz clock domain of the ADC to the 10GHz Fsample/2 domain. A variable delay chip, HMC856, is provided between the DFF and the ADC chip to adjust the timing of the synchronized sequence to meet the setup and hold requirements of the ADC's internal registers.

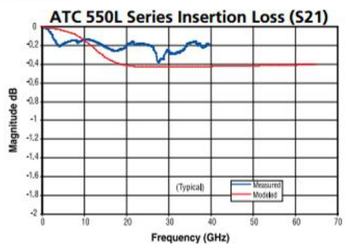
A pair of HMC859 divide-by-eight chips create the 156.25 MHz reference clock necessary for data reception in the GTH Rxs from the 10GHz clock output from the ADC (in all of this discussion and all of the testing I used a 20 GHz sample clock generated by the Hittite HMC T2100 generator seen in the photo).

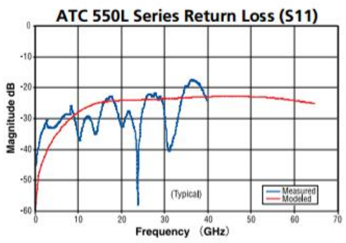
DC power is supplied from the +12v available on the FMC connector. Two DC/DC switching converters convert +12 to the -5.0 and -3.3v supply rails required by the ADC and other parts.

D/A converters are provided to set the ADC's REFT and REFB levels, and thus the conversion range. The signal input of the ADC is single-ended, dc-coupled, and is buffered internally by an emitter-follower stage, so the signal applied to the ADC core is offset by the Vbe of this stage. The DACs set the top and bottom of the reference ladder to the desired input range offset by this value. The nominal settings are REFT = -900mv, REFB = -(900 + 256) mv, and thus the nominal input span is 256mv, or 1mv per LSB.

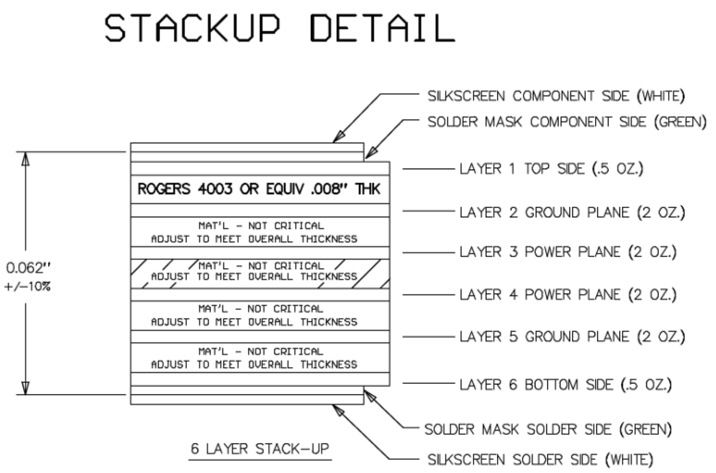
The board sits piggyback on the VC709 and connects via a Samtec SAMTEC ASP-134488-01 FMC-HPC (FPGA Mezzanine Connector- High Pin Count) connector.

Serial data output to or input from the FPGA are at “Current Mode Logic” (CML) levels below ground and must therefore be AC-coupled to the FPGA. A long time constant is desirable here to provide a long averaging time for low dc disparity. 100nF “ultra-wideband” capacitors fro American Technical Ceramics are used to provide a 5us (100nf * 50 ohm half-circuit termination resistance) time constant with low losses at RF frequencies:





All critical signal pairs are run on the top layer of the PCB, whose dielectric is Rogers 4003 RF material. Multiple 2 oz ground planes are employed to conduct heat away from the high-power parts (primarily the ADC chip). Filled thermal vias under the ADC chip conduct heat to a bottom-surface copper fill.


STACKUP DETAIL
SILKSCREEN COMPONENT SIDE (WHITE)
SOLDER MASK COMPONENT SIDE (GREEN)
LAYER 1 TOP SIDE (.5 OZ.)
ROGERS 4003 OR EQUIV .008" THK
LAYER 2 GROUND PLANE (2 OZ.)
MAT'L - NOT CRITICAL ADJUST TO MEET OVERALL THICKNESS
LAYER 3 POWER PLANE (2 OZ.)
0.062" +/-10%
MAT'L - NOT CRITICAL ADJUST TO MEET OVERALL THICKNESS
LAYER 4 POWER PLANE (2 OZ.)
MAT'L - NOT CRITICAL ADJUST TO MEET OVERALL THICKNESS
LAYER 5 GROUND PLANE (2 OZ.)
MAT'L - NOT CRITICAL ADJUST TO MEET OVERALL THICKNESS
LAYER 6 BOTTOM SIDE (.5 OZ.)
SOLDER MASK SOLDER SIDE (GREEN)
6 LAYER STACK-UP
SILKSCREEN SOLDER SIDE (WHITE)

Data Alignment

Successful data reception by the GTH Receivers requires several of the GTH subsystems. The alignment sequence is as follows:

- Assert INHIBIT to force ADC data to 0
- Send pseudo-random bit sequence (PRBS) to modulation port. Enable PRBS pattern detection in all receivers. This sequence is generated in the GTH transmitter; dedicated logic in the GTH receiver detects errors in the received sequence without regard to word- or channel-alignment. This pattern is only used to set the two delays.
- Perform a 2D scan of delays. The delay through the GTH transmitter is varied using the PPM Controller of the Tx Phase Interpolator, to find a zone where the data input to the synchronizing flipflop meets setup and hold requirements. The delay of the HMC856LC5 variable delay chip is also varied to find a zone where the data output by the flipflop meets the setup requirements of the ADC input register. Find a zone in that 2D space where no PRBS7 errors. Set delays in the middle of this zone.
- Send modulating sequence to modulation port. The modulating sequence contains both the "commas" necessary for symbol alignment (alignment of the parallel data on word, in this case 40-bit, boundaries) and the "channel bondng sequence" necessary to align all of the lanes to each other
- Turn on 8B/10B encoders and decoders. These are necessary to do the comma detection and channel bonding.
- Enable comma alignment and to align parallel output of receivers to word boundary; freeze alignment.
- Initiate channel bonding algorithm (aligning the 8 lanes to each other); freeze channel bond alignment.
- Turn off 8B/10B encoders and decoders.

At this point the data are phase-adjusted and word-aligned, and the modulating sequence will be seen identically at the eight receiver outputs. The received sequences are XORed by the demodulation sequence and the result is all zeroes in all channels. The ADC inhibit is now de-asserted and the ADC data is accurately received.

Block diagrams of a GTH receiver and transmitter channel are shown here with the functions that are important for the application highlighted.

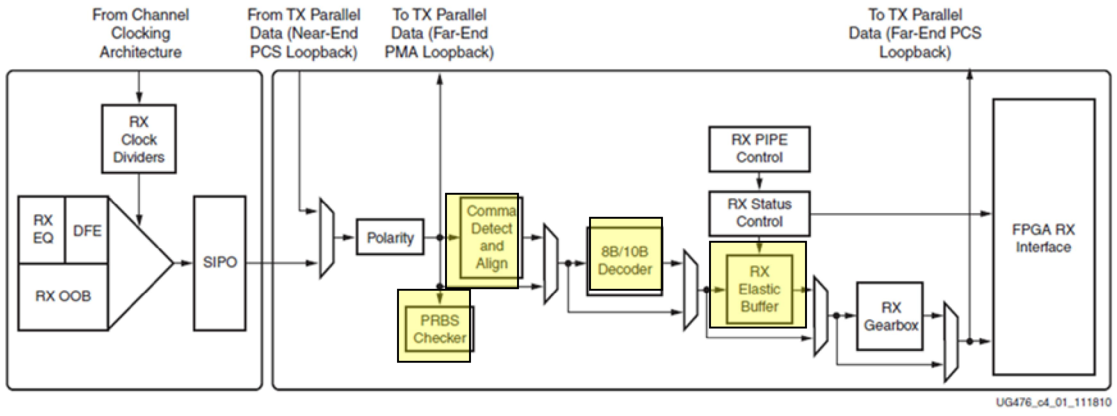


Figure 4-1: **GTX/GTH Transceiver RX Block Diagram**

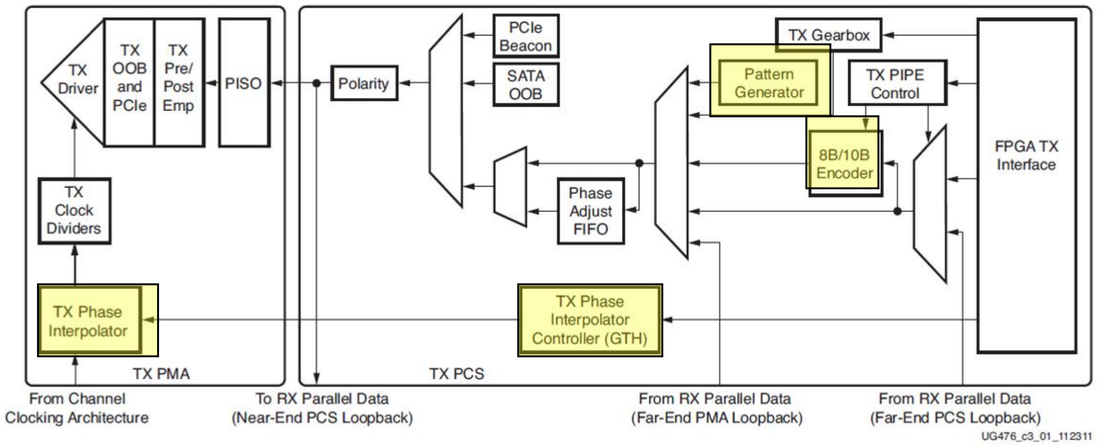
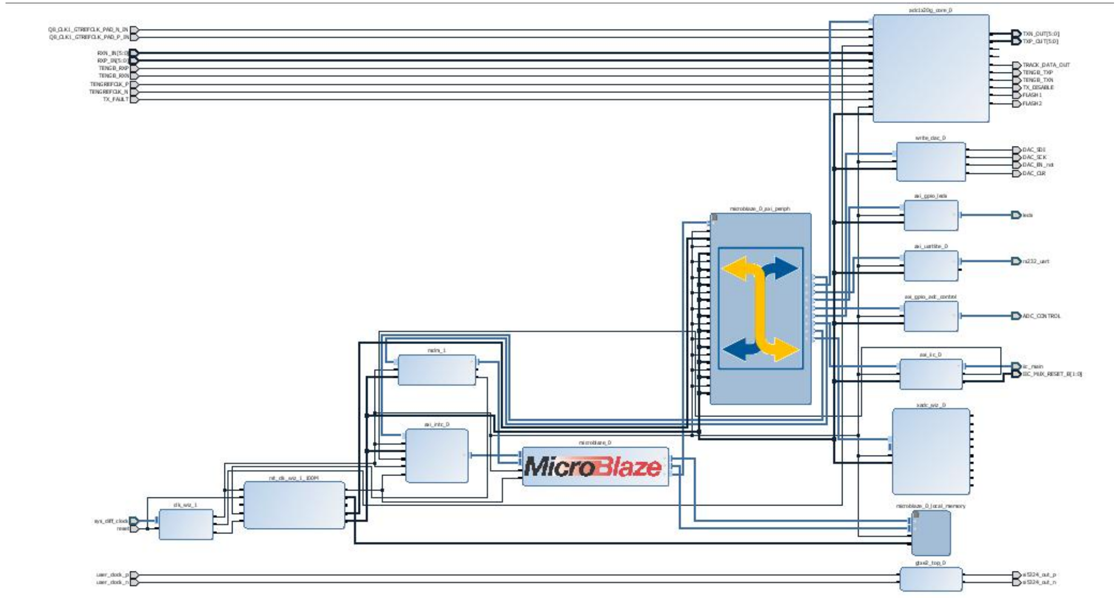


Figure 3-1: **GTX/GTH Transceiver TX Block Diagram**

Firmware Design

The firmware was developed in Xilinx's Vivado design environment (v2014.2) and comprises a MicroBlaze soft processor, many standard MicroBlaze peripherals, and a custom peripheral which includes the six GTH receivers, one GTH transmitter, modulation and demodulation sources, and the 10Gb/s Ethernet components necessary for high-speed data output.

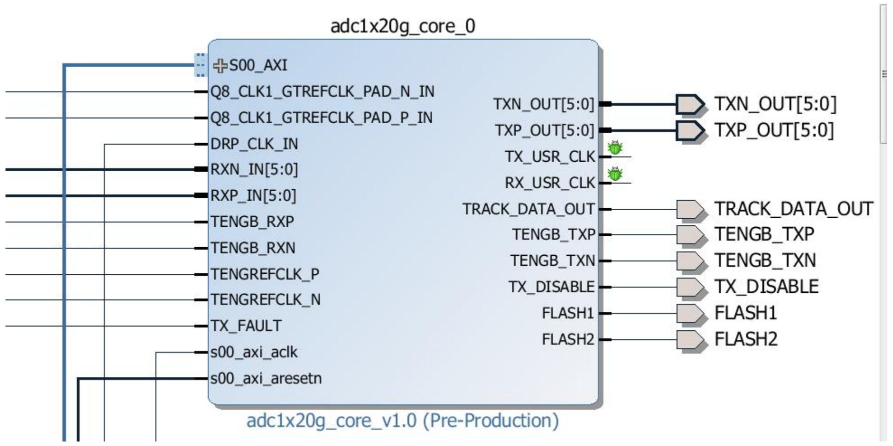


MicroBlaze system diagram

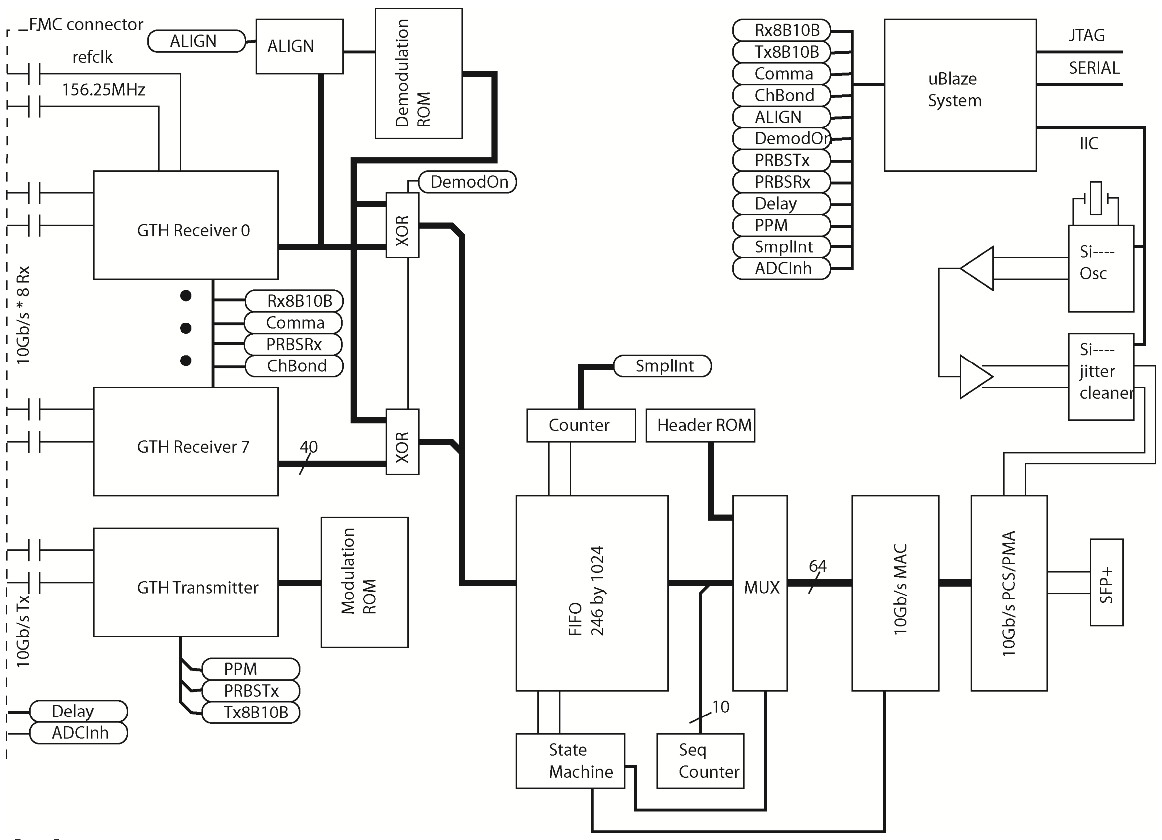
Standard peripherals employed in the design include:

- DRAM controller
- Interrupt controller
- UART
- IIC interface
- General Purpose IO module

The custom peripheral contains the GTH transmitter and receivers used to modulate and acquire ADC data, the GTH transmitter for 10Gb/s Ethernet output, and all support logic.



the custom core containing GTH Rxs and Txs



inside the custom core

The six GTH receivers used to receive the ADC data have several controls accessible by the MicroBlaze processor:

- 8B/10B decoder enable
- Comma detect enable
- Channel bond enable
- PRBS sequence
- PRBS error counter

The GTH transmitter used to produce the modulating sequence also has several processor-accessible controls:

- 8B/10B encoder enable
- PRBS generator enable
- PPM controller controls

The GTH channels operate at a parallel data width of 40 bits; for a 20GSPS sample rate and 10Gb/s per-channel bit rate the parallel input/output data rate is 250MHz (the design is compiled for a specific sample rate). The modulating sequence is stored in a 40-by-16 ROM and is thus 640bits long. One of four such modulating sequences (generated at compile time) is selected under processor control.

Data from the eight Rx channels are buffered in a FIFO. An interval counter whose terminal count is settable under processor control initiates the writing of a series of samples into the FIFO. The FIFO data is read out under the control of a state machine which loads data into the 10Gb/s Ethernet MAC (Media Access Controller), which drives the Ethernet PCS/PMA layers. Appropriate headers are added to the data for Ethernet, Internet Protocol, and UDP protocol (packets are written to hard-coded MAC and IP address and UDP port). Clock for the 10Gb/s Ethernet infrastructure is provided by the VC709’s dedicated clock generation hardware (156.25MHz, independent of the ADC clock structure).

A demodulation ROM, identical to the modulation ROM, is synchronized to the received data stream (with ADC inhibited). The ROM’s output is XORed with all received data streams to recover the unmodulated ADC data.

Integrated Logic Analyzer (formerly known as ChipScope) cores are included to display the six 40-bit data paths at the parallel outputs of the six GTH Rx channels. These are very useful for system debug.

Choice of Modulating Sequence

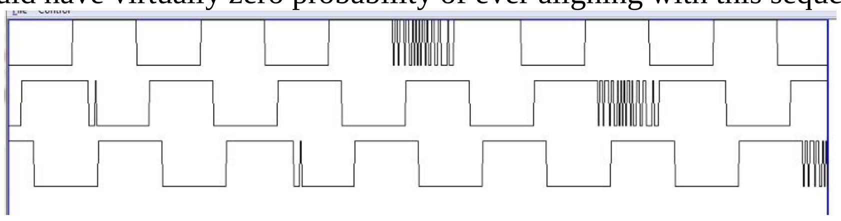
The modulating sequence with which the ADC data is XORed must meet two criteria:

- The sequence must have low dc disparity (must have approximately the same number of 1s and 0s) so that, when the ADC output is inhibited, the ac-coupled sequence will not contain an appreciable DC component that would degrade the data eye.
- The sequence must result in a sufficient frequency of data transitions in the modulated data streams such that the GTH receiver clock recovery system can function properly.

Additionally, the modulation sequence may have the word- and channel-alignment codes built-in to permit the Rx logic to do word- and channel-alignment. Note that the PRBS sequence is only used for adjustment of the phase of the modulation data and is not used as the modulation sequence itself.

The second requirement demands that the chosen modulation sequence have an extremely low probability of aligning with the ADC output data; if the two sequences were to align for a sufficiently long period, the receivers would see no clock transitions and would fall out of lock. This could cause one or more lanes to lose alignment with others. The maximum transition-free interval over which the receiver clock recovery system will stay in lock is not specified in the Xilinx documentation.

I experimented with several sequences and chose a slightly modified 250MHz square wave as my sequence. The 640-bit sequence contains eight cycles of this waveform, plus one instance of the comma (for word alignment) and one instance of the channel-bond sequence (for lane-to-lane alignment). A few cycles of the modulating sequence can be seen below. The single spike positive is one bit of the comma; the extended chattering area is the channel-bond sequence. I believe that an RF input high-passed at say 500MHz would have virtually zero probability of ever aligning with this sequence.



MicroBlaze Software

The MicroBlaze soft processor receives control commands from a host via a serial link. The processor interacts with the hardware, setting registers as needed. Additionally, the processor can do the entire data alignment process when issued the appropriate command by the host.

Host Interface

A host computer can interact with the system using a simple serial port interface. A sequence of 10 ASCII characters, terminated by the character "X" allows writing to one of 10 registers:

"ddddddddaX" writes the hex value 0xdddddddd to register a, 0<=a<=9.

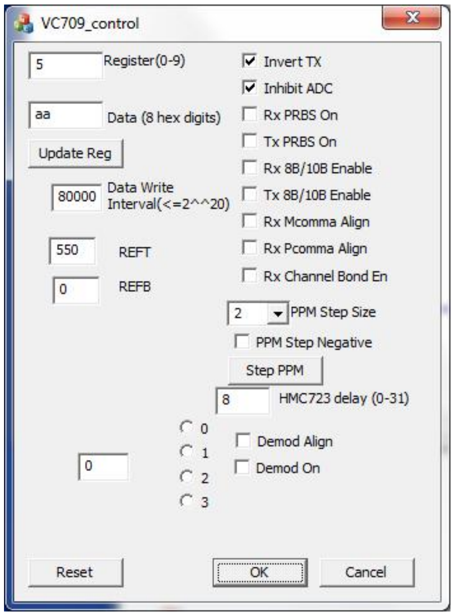
(the hex value uses the characters 0-9 and A-F only, no lower-case)

The registers are as follows

reg 0 to 2:	adc1x20g core control		
reg 3:	unused		
reg 4:	unused		
reg 5:	LEDs (six bits- two have been used as clock liveness indicators in this version)		
reg 6	ADC board low-speed controls:		
		bits 4:0	HMC856 delay control, 0-31, approx 3ps/step
		bit 5:	ADC_inhibit = 1 to force ADC output to 0
reg 7	DAC REFT		0 to 1023
reg 8	DAC REFB		0 to 1023
reg 9	Command register:		
	data=1: Sweep PPM Delay		
	data=2: Sweep HMC856 Delay		
	data=3: Adjust PPM delay		
	data=4: Do complete data alignment		
	data=5: Read die temperature		

Control and display GUIs

I wrote a Visual C++ GUI to facilitate control of the system.


VC709_control

5 Register(0-9)

aa Data (8 hex digits)

Update Reg

80000 Data Write Interval(<=2^^20)

550 REFT

0 REFB

0

○ 0
○ 1
○ 2
○ 3

☑ Invert TX
☑ Inhibit ADC
☐ Rx PRBS On
☐ Tx PRBS On
☐ Rx 8B/10B Enable
☐ Tx 8B/10B Enable
☐ Rx Mcomma Align
☐ Rx Pcomma Align
☐ Rx Channel Bond En

2 PPM Step Size

☐ PPM Step Negative

Step PPM

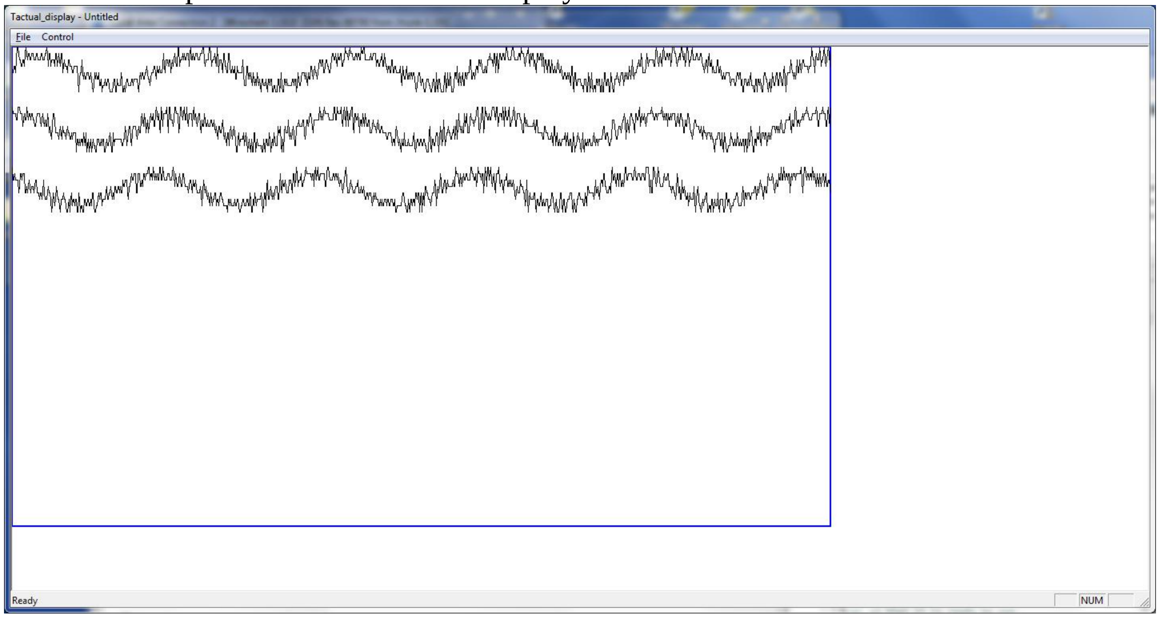
8 HMC723 delay (0-31)

☐ Demod Align
☐ Demod On

Reset OK Cancel

The various controls to the custom GTH peripheral can be adjusted by hand using this control GUI.

I also wrote a piece of VC++ code to display the data and store files.



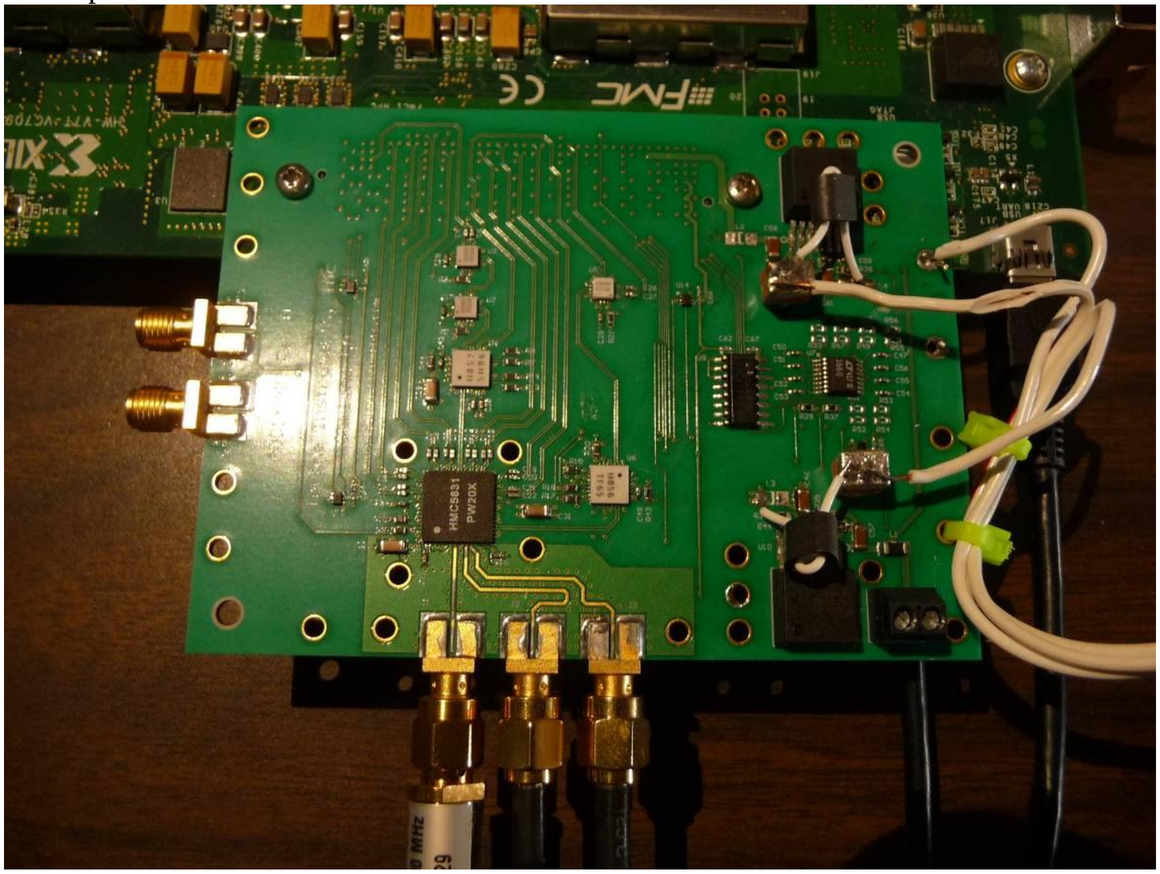
Data output display. Signal is a 100MHz sine wave phase-locked to the generator, with about 1 LSB rms of added wideband noise.

Design Problems

I made two mistakes in this board design:

- First, I made a pinout error on the FMC connector: the two Overrange pairs were connected to non-GTH FPGA pairs, so the overrange bit is unavailable, changing the converter from a 3.5-bit to a 3-bit one.
- Second, the switching converters that I selected to produce -5.0v and -3.3v from the +12v available on the FMC connector were much too noisy without additional linear post-regulators. I bypassed these and brought -5.0, -3.3 in from a pair of lab supplies.

These problems will be addressed in a board revision.



The ADC1X26G Board. Power enters on the right, differential clock and single-ended data at the bottom. SMAs at the left are unused.

Test Results

For all tests I drove the clock input with the Hittite HMC-T2100 generator, via a Marki BAL0026 balun. Generator amplitude was set to +12 dBm. The balun has an insertion loss of 6dB, and the RG316 cables used to connect generator to balun and balun to board probably contribute a few dB at 20GHz (I have no way to measure this signal level, but RG316 is not very good above a few GHz). I drove the RF input with a combination of noise from the NoiseWave NW18G-MI generator (which specifies a +-0.5dB response from 2 to 10 GHz, and +-2dB from 2 to 18 GHz) and a PTS3200 RF generator to provide lower frequency sine waves. The two signal sources were combined by a power combiner internal to the noise generator. For some tests I used a VLF-7200 low-pass filter from Minicircuits (-2dB at 8 GHz) on the input. I used some better RG142 for the connection from generator to board.

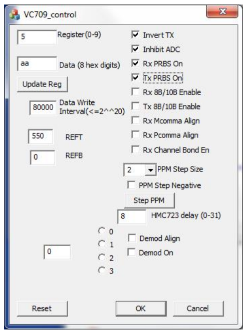
Marki wideband balun for the clock input

Manual Data Alignment

It's instructive to step the system through the various phases of the data alignment process by hand, observing the system response in the Integrated Logic Analyzer display, in order to more clearly understand the process.

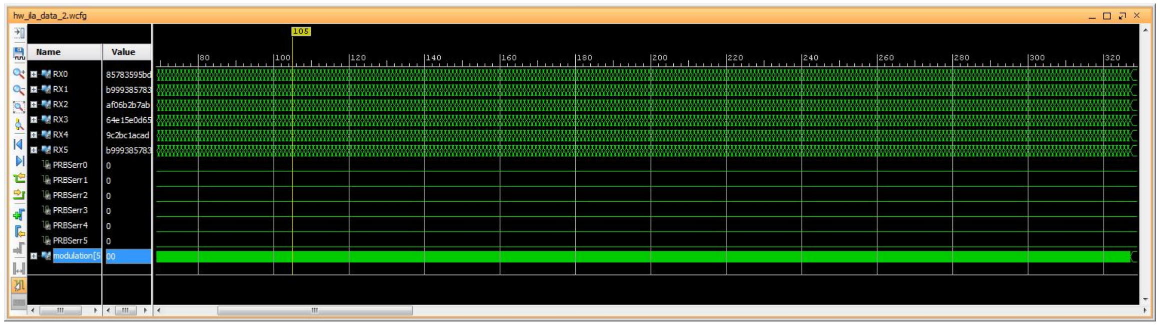
Step 1: Adjust clock phases with PRBS code stream

Inhibit ADC and turn on PRBS generation/detection in order to adjust the two delay parameters to center bit sampling in the data eye.



VC709_control

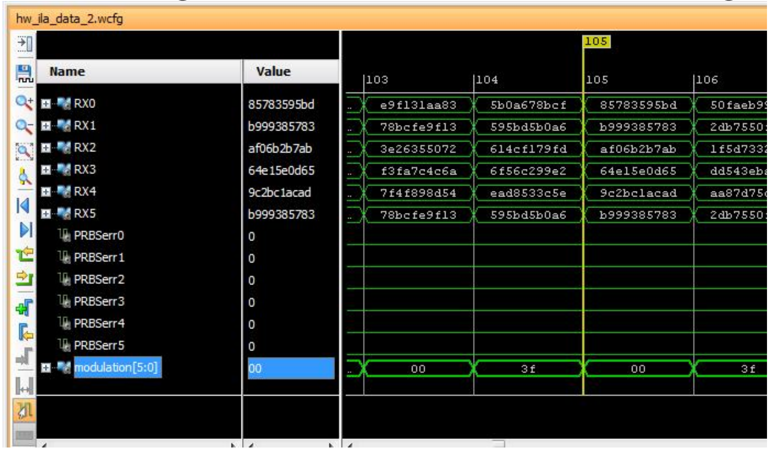
- Register(0-9): 5
- Data (8 hex digits): aa
- Data Write Interval(<=2^^20): 80000
- REFT: 550
- REFB: 0
- ✓ Invert TX
- ✓ Inhibit ADC
- ✓ Rx PRBS On
- ✓ Tx PRBS On
- ☐ Rx 8B/10B Enable
- ☐ Tx 8B/10B Enable
- ☐ Rx Mcomma Align
- ☐ Rx Pcomma Align
- ☐ Rx Channel Bond En
- PPM Step Size: 2
- ☐ PPM Step Negative
- HMC723 delay (0-31): 8
- 0
- ○ 0 ○ 1 ○ 2 ○ 3
- ☐ Demod Align
- ☐ Demod On



```
Name           Value
RX0            85783595bd
RX1            b999385783
RX2            af06b2b7ab
RX3            64e15e0d65
RX4            9c2bc1acad
RX5            b999385783
PRBSerr0       0
PRBSerr1       0
PRBSerr2       0
PRBSerr3       0
PRBSerr4       0
PRBSerr5       0
modulation[5:0] 00
```

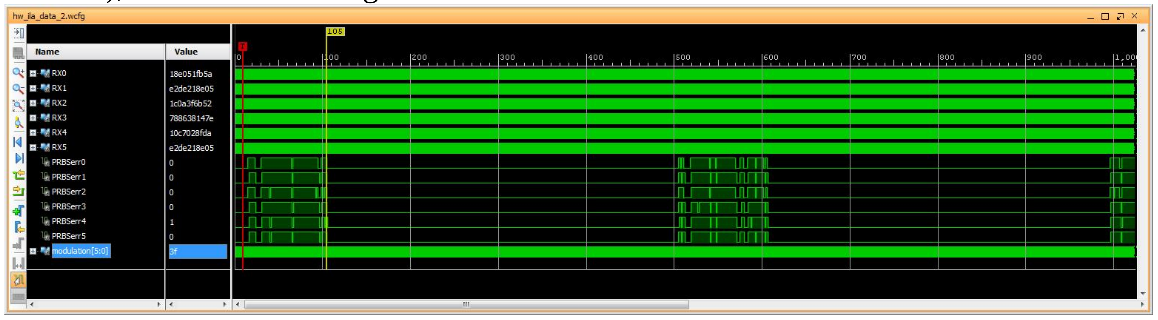
The screenshot above shows a 1k-sample (4us) time slice. The top six waveforms are the six 40-bit data streams; the next six are the PRBS error flags (one per channel) and the last is the modulation sequence which will be XORed with the data when it is aligned. Only the six LSBs of the modulation sequence are shown. The PRBS error flags are not being asserted, meaning for this initial power-on state the data streams are all being sampled without errors.

Zooming in on a smaller window of data we can see that the PRBS sequences are all different, meaning that the data streams are neither word-aligned nor channel-aligned.



```
Name            Value        |103        |104        |105        |106
RX0             85783595bd    e9f131aa83  5b0a678bcf  85783595bd  50faeb9
RX1             b999385783    78bcfe9f13  595bd5b0a6  b999385783  2db7550
RX2             af06b2b7ab    3e26355072  614cf179fd  af06b2b7ab  1f5d733
RX3             64e15e0d65    f3fa7c4c6a  6f56c299e2  64e15e0d65  dd543eb
RX4             9c2bc1acad    7f4f898d54  ead8533c5e  9c2bc1acad  aa87d75
RX5             b999385783    78bcfe9f13  595bd5b0a6  b999385783  2db7550
PRBSerr0        0
PRBSerr1        0
PRBSerr2        0
PRBSerr3        0
PRBSerr4        0
PRBSerr5        0
modulation[5:0] 00            00          3f          00          3f
```

Now we can step the PPM controller through a series of steps in the positive direction until the PRBS error flags become active, to find the edge of the data eye. After 24 steps (actually 48 steps of the native step size of the PPM controller, as I have the PPM step size set to 2), all PRBS error flags become active:



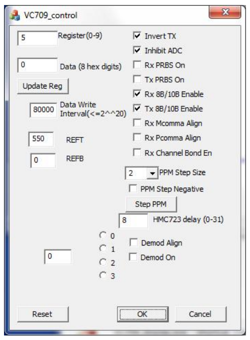
```
Name            Value
RX0             18e051fb5a
RX1             e2de218e05
RX2             1c0a3f6b52
RX3             788638147e
RX4             10c7028fda
RX5             e2de218e05
PRBSerr0        0
PRBSerr1        0
PRBSerr2        0
PRBSerr3        0
PRBSerr4        1
PRBSerr5        0
modulation[5:0] 3f
```

We can now step the PPM controller negative until we pass through the error-free zone of phase settings to another error-full zone, and we find that the width of the error-free zone is about 28 steps; we then step positive half this amount to center the phase in the error-free zone. I have found this behavior to completely stable and repeatable: after a reset, the PPM controller must be stepped positive 24 +- 1 steps to get to the first error zone.

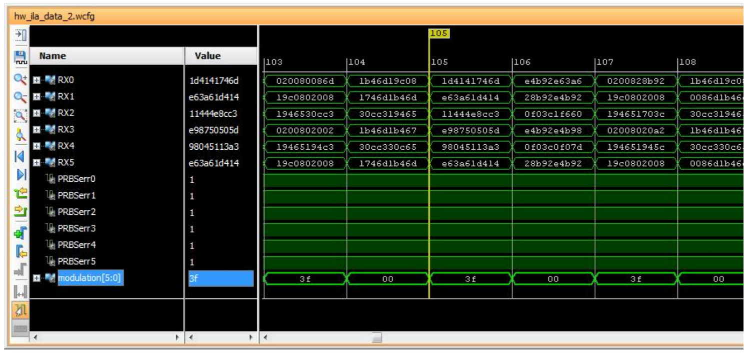
We can repeat this process with the setting of the HMC856 delay chip. This chip has a 0 to 31 range with a nominal step size of 3ps. Remarkably, the only settings which produce any errors are 0, 16 and 17; for all other settings the PRBS codes are received error-free. I have left this control set at 8 for all tests.

Step 2: word-align all channels

Now, we turn off the PRBS generation/detection and turn on the 8B/10B encoders and decoders. Now the modulation sequence is flowing through the ADC chip and is replicated on all channels.

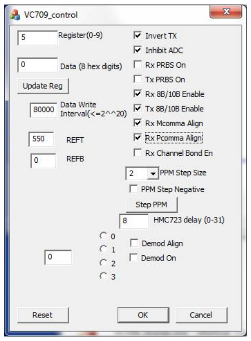


```
VC709_control
5      Register(0-9)                 [✓] Invert TX
                                     [✓] Inhibit ADC
0      Data (8 hex digits)           [ ] Rx PRBS On
Update Reg                           [ ] Tx PRBS On
                                     [✓] Rx 8B/10B Enable
80000  Data Write                    [✓] Tx 8B/10B Enable
       Interval(<=2^^20)             [ ] Rx Mcomma Align
550    REFT                          [ ] Rx Pcomma Align
0      REFB                          [ ] Rx Channel Bond En
                              2  PPM Step Size
                              [ ] PPM Step Negative
                              Step PPM
                              8      HMC723 delay (0-31)
              ( ) 0      [ ] Demod Align
0             ( ) 1      [ ] Demod On
              ( ) 2
              ( ) 3
Reset                         OK        Cancel
```

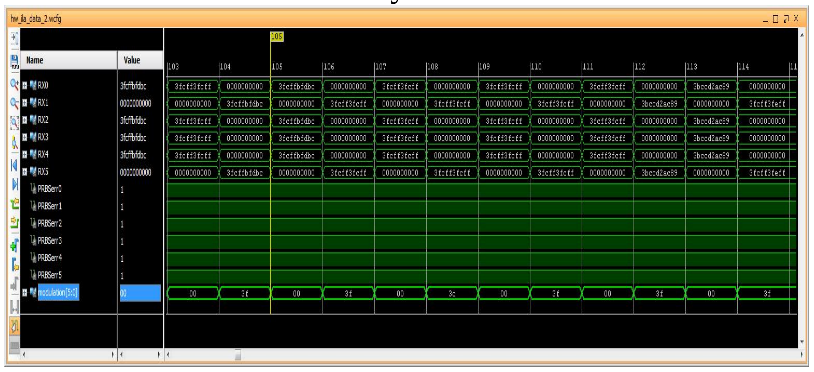


```
hw_ila_data_2.wcfg
                                   105
Name             Value         |103        |104        |105        |106        |107        |108
RX0              1d4141746d    020080086d  1b46d19c08  1d4141746d  e4b92e63a6  0200828b92  1b46d19c0
RX1              e63a61d414    19c0802008  1746d1b46d  e63a61d414  28b92e4b92  19c0802008  0086d1b46
RX2              11444e8cc3    1946530cc3  30cc319465  11444e8cc3  0f03c1f660  194651703c  30cc31946
RX3              e98750505d    0200802002  1b46d1b467  e98750505d  e4b92e4b98  02008020a2  1b46d1b46
RX4              98045113a3    19465194c3  30cc330c65  98045113a3  0f03c0f07d  194651945c  30cc330c6
RX5              e63a61d414    19c0802008  1746d1b46d  e63a61d414  28b92e4b92  19c0802008  0086d1b46
PRBSerr0         1
PRBSerr1         1
PRBSerr2         1
PRBSerr3         1
PRBSerr4         1
PRBSerr5         1
modulation[5:0]  3f            3f          00          3f          00          3f          00
```

The data streams are still un-aligned. We now turn on comma alignment. The commas are special characters inserted in the data stream to indicate the word boundaries of the data. There is both a positive and a negative comma because the 8B/10B encoding scheme allows two possible codes for each encoded symbol:



```
VC709_control
5      Register(0-9)                 [✓] Invert TX
                                     [✓] Inhibit ADC
0      Data (8 hex digits)           [ ] Rx PRBS On
Update Reg                           [ ] Tx PRBS On
                                     [✓] Rx 8B/10B Enable
80000  Data Write                    [✓] Tx 8B/10B Enable
       Interval(<=2^^20)             [✓] Rx Mcomma Align
550    REFT                          [✓] Rx Pcomma Align
0      REFB                          [ ] Rx Channel Bond En
                              2  PPM Step Size
                              [ ] PPM Step Negative
                              Step PPM
                              8      HMC723 delay (0-31)
              ( ) 0      [ ] Demod Align
0             ( ) 1      [ ] Demod On
              ( ) 2
              ( ) 3
Reset                         OK        Cancel
```

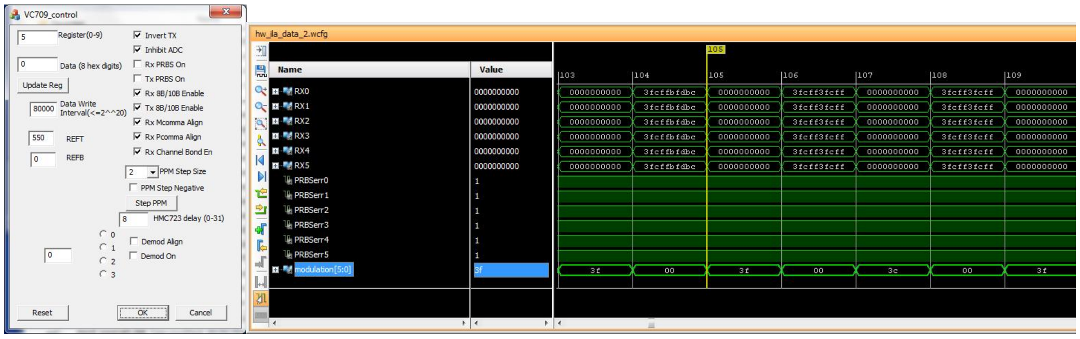


```
hw_ila_data_2.wcfg
                                   105
Name             Value         |103        |104        |105        |106        |107        |108        |109        |110        |111        |112        |113        |114
RX0              3fcffbfdbc    3fcff3fcff  0000000000  3fcffbfdbc  0000000000  3fcff3fcff  0000000000  3fcff3fcff  0000000000  3fcff3fcff  0000000000  3bccd2ac89  0000000000
RX1              0000000000    0000000000  3fcffbfdbc  0000000000  3fcff3fcff  0000000000  3fcff3fcff  0000000000  3fcff3fcff  0000000000  3bccd2ac89  0000000000  3fcff3feff
RX2              3fcffbfdbc    3fcff3fcff  0000000000  3fcffbfdbc  0000000000  3fcff3fcff  0000000000  3fcff3fcff  0000000000  3fcff3fcff  0000000000  3bccd2ac89  0000000000
RX3              3fcffbfdbc    3fcff3fcff  0000000000  3fcffbfdbc  0000000000  3fcff3fcff  0000000000  3fcff3fcff  0000000000  3fcff3fcff  0000000000  3bccd2ac89  0000000000
RX4              3fcffbfdbc    3fcff3fcff  0000000000  3fcffbfdbc  0000000000  3fcff3fcff  0000000000  3fcff3fcff  0000000000  3fcff3fcff  0000000000  3bccd2ac89  0000000000
RX5              0000000000    0000000000  3fcffbfdbc  0000000000  3fcff3fcff  0000000000  3fcff3fcff  0000000000  3fcff3fcff  0000000000  3bccd2ac89  0000000000  3fcff3feff
PRBSerr0         1
PRBSerr1         1
PRBSerr2         1
PRBSerr3         1
PRBSerr4         1
PRBSerr5         1
modulation[5:0]  00            00          3f          00          3f          00          3c          00          3f          00          3f          00          3f
```

Now all six channels are outputting the same sequence of 40-bit words, but the channels are not aligned to each other.

Step 3: Channel-bond the six channels

Now we turn channel-bonding on, which causes the receiver logic to look for a unique channel-bond sequence in the data streams and to adjust pointers in the receive FIFOs to line up all channels.


VC709_control

5 Register(0-9)
0 Data (8 hex digits)
Update Reg
80000 Data Write Interval(<=2^^20)
550 REFT
0 REFB
0

[✓] Invert TX
[✓] Inhibit ADC
[ ] Rx PRBS On
[ ] Tx PRBS On
[✓] Rx 8B/10B Enable
[✓] Tx 8B/10B Enable
[✓] Rx Mcomma Align
[✓] Rx Pcomma Align
[✓] Rx Channel Bond En
2 PPM Step Size
[ ] PPM Step Negative
Step PPM
8 HMC723 delay (0-31)
[ ] Demod Align
[ ] Demod On

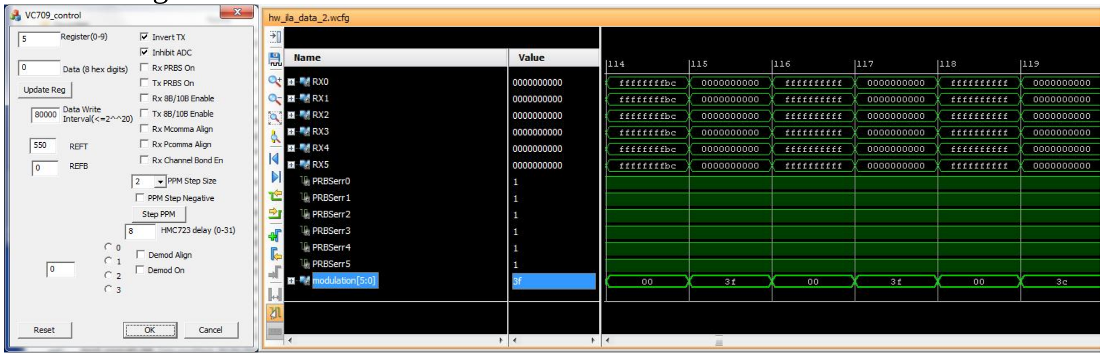
hw_ila_data_2.wcfg

```
Name             Value       |103        |104        |105        |106        |107        |108        |109
RX0              0000000000  0000000000  3fcffbfdbc  0000000000  3fcff3fcff  0000000000  3fcff3fcff  0000000000
RX1              0000000000  0000000000  3fcffbfdbc  0000000000  3fcff3fcff  0000000000  3fcff3fcff  0000000000
RX2              0000000000  0000000000  3fcffbfdbc  0000000000  3fcff3fcff  0000000000  3fcff3fcff  0000000000
RX3              0000000000  0000000000  3fcffbfdbc  0000000000  3fcff3fcff  0000000000  3fcff3fcff  0000000000
RX4              0000000000  0000000000  3fcffbfdbc  0000000000  3fcff3fcff  0000000000  3fcff3fcff  0000000000
RX5              0000000000  0000000000  3fcffbfdbc  0000000000  3fcff3fcff  0000000000  3fcff3fcff  0000000000
PRBSerr0         1
PRBSerr1         1
PRBSerr2         1
PRBSerr3         1
PRBSerr4         1
PRBSerr5         1
modulation[5:0]  3f          3f          00          3f          00          3c          00          3f
```

We see now that all channels are producing identical aligned data streams.

Step 4: Turn off channel-bonding, comma-detection, and encoders/decoders.

Now that the channels are aligned, we can freeze the alignment by turning off all the alignment functions, so that random patterns in the ADC data do not cause the data to become mis-aligned.


VC709_control

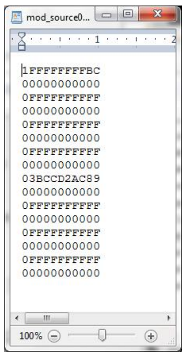
5 Register(0-9)
0 Data (8 hex digits)
Update Reg
80000 Data Write Interval(<=2^^20)
550 REFT
0 REFB
0

[✓] Invert TX
[✓] Inhibit ADC
[ ] Rx PRBS On
[ ] Tx PRBS On
[ ] Rx 8B/10B Enable
[ ] Tx 8B/10B Enable
[ ] Rx Mcomma Align
[ ] Rx Pcomma Align
[ ] Rx Channel Bond En
2 PPM Step Size
[ ] PPM Step Negative
Step PPM
8 HMC723 delay (0-31)
[ ] Demod Align
[ ] Demod On

hw_ila_data_2.wcfg

```
Name             Value       |114        |115        |116        |117        |118        |119
RX0              0000000000  ffffffffbc  0000000000  ffffffffff  0000000000  ffffffffff  0000000000
RX1              0000000000  ffffffffbc  0000000000  ffffffffff  0000000000  ffffffffff  0000000000
RX2              0000000000  ffffffffbc  0000000000  ffffffffff  0000000000  ffffffffff  0000000000
RX3              0000000000  ffffffffbc  0000000000  ffffffffff  0000000000  ffffffffff  0000000000
RX4              0000000000  ffffffffbc  0000000000  ffffffffff  0000000000  ffffffffff  0000000000
RX5              0000000000  ffffffffbc  0000000000  ffffffffff  0000000000  ffffffffff  0000000000
PRBSerr0         1
PRBSerr1         1
PRBSerr2         1
PRBSerr3         1
PRBSerr4         1
PRBSerr5         1
modulation[5:0]  3f          00          3f          00          3f          00          3c
```

Now we see the full 40-bit aligned data, and we can see that the sequences match those in the modulation source file for the selected modulation ROM:

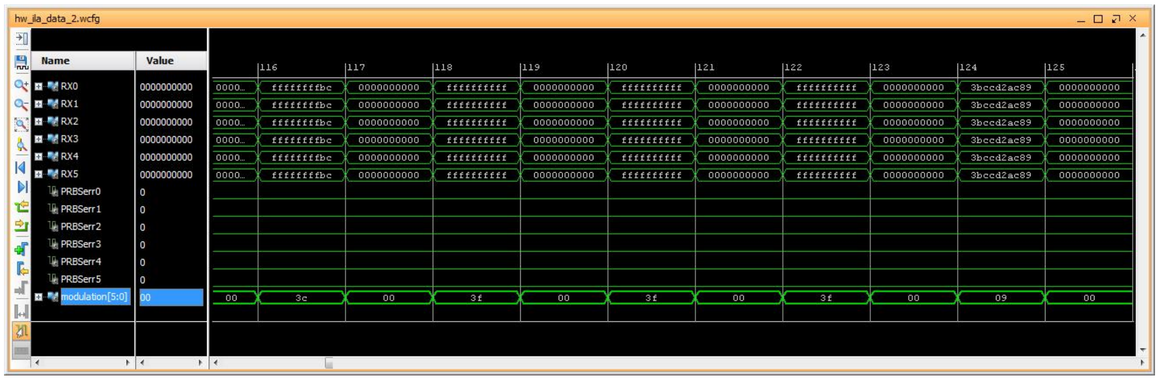

mod_source0...

```
1FFFFFFFFBC
00000000000
0FFFFFFFFFF
00000000000
0FFFFFFFFFF
00000000000
0FFFFFFFFFF
00000000000
03BCCD2AC89
00000000000
0FFFFFFFFFF
00000000000
0FFFFFFFFFF
00000000000
0FFFFFFFFFF
00000000000
```

The ROM entries have an extra 4 bits (the leftmost character) which indicate to the logic which words contain the special control character used as a comma.

Step 5: Align the demodulation generator to the received data streams.

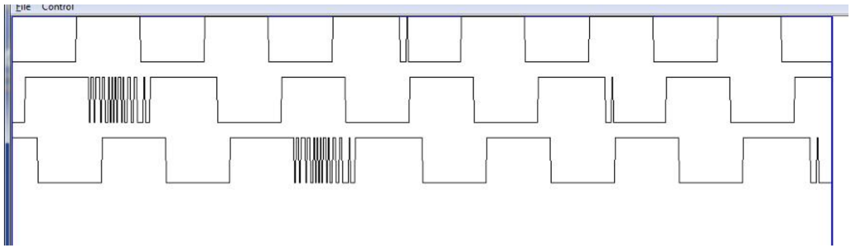
The "Demod Align" checkbox is checked and unchecked. This causes the address counter for the demodulation ROM to sync up with the data received:


hw_ila_data_2.wcfg

Name	Value		116	117	118	119	120	121	122	123	124	125
RX0	0000000000	0000...	ffffffffbc	0000000000	ffffffffff	0000000000	ffffffffff	0000000000	ffffffffff	0000000000	3bccd2ac89	0000000000
RX1	0000000000	0000...	ffffffffbc	0000000000	ffffffffff	0000000000	ffffffffff	0000000000	ffffffffff	0000000000	3bccd2ac89	0000000000
RX2	0000000000	0000...	ffffffffbc	0000000000	ffffffffff	0000000000	ffffffffff	0000000000	ffffffffff	0000000000	3bccd2ac89	0000000000
RX3	0000000000	0000...	ffffffffbc	0000000000	ffffffffff	0000000000	ffffffffff	0000000000	ffffffffff	0000000000	3bccd2ac89	0000000000
RX4	0000000000	0000...	ffffffffbc	0000000000	ffffffffff	0000000000	ffffffffff	0000000000	ffffffffff	0000000000	3bccd2ac89	0000000000
RX5	0000000000	0000...	ffffffffbc	0000000000	ffffffffff	0000000000	ffffffffff	0000000000	ffffffffff	0000000000	3bccd2ac89	0000000000
PRBSerr0	0											
PRBSerr1	0											
PRBSerr2	0											
PRBSerr3	0											
PRBSerr4	0											
PRBSerr5	0											
modulation[5:0]	00	00	3c	00	3f	00	3f	00	3f	00	09	00

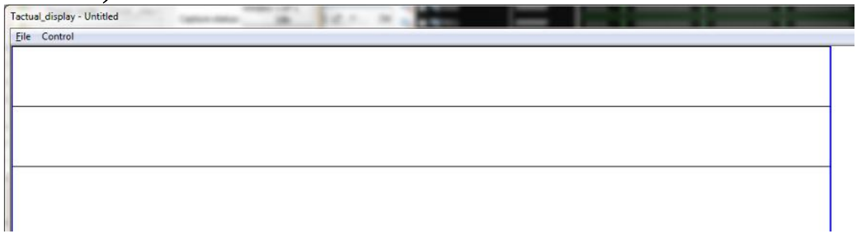
The six LSBs at the bottom match those of all data channels; in fact the demodulation stream is a full 40-bits in width, but only six LSBs are displayed.

If at this point we capture data, we will see just the modulation sequence (though we have sync'ed the demodulation source, we have not yet enabled the demodulation XOR function:



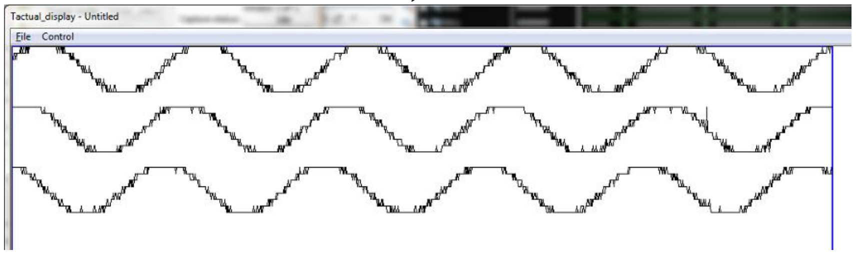
The data capture GUI displays three rows of 512 samples each. The modulation sequence of 640 bits is repeated over and over. This sequence is a 250MHz squarewave with one comma and one channel-bond sequence added. The displayed data alternates between 000 and 111 since all channels are identical.

Now we can turn on the demodulation function (checkbox at the bottom), and we see that the ADC output is all zeroes, as it should be (the demodulation cancels the modulation)



The output display has "infinite persistence" (it adds up all the signals displayed) and is updated once per second. A single bit error is plainly evident during this test, and I never saw one.

Now we can uninhibit the ADC, and see data



The data being converted here is a 100MHz sinewave adjusted to full-scale amplitude. The signal generator is phase-locked to the same source as the 20GHz clock generator.

Automated Alignment

The MicroBlaze can do the phase alignment automatically, scanning the phase setting through a range of delays, finding the edges of the error-free zones, and centering the phase. We can also use the processor to scan through an arbitrary range of delay settings, reporting the results via the serial link. Here is a typical output from a scan through 100*2 steps of the PPM controller after a system reset:

```
800000000X                          //Assert RESET
000000000X                          /De-assert RESET
000000019X                          //Sweep the PPM controller
//MicroBlaze response follows
Stepping PPM through 100 steps of 2 each
Step 0      0       0       0       0       0       0
Step 1      0       0       0       0       0       0
Step 2      0       0       0       0       0       0
Step 3      0       0       0       0       0       0
Step 4      0       0       0       0       0       0
Step 5      0       0       0       0       0       0
Step 6      0       0       0       0       0       0
Step 7      0       0       0       0       0       0
Step 8      0       0       0       0       0       0
Step 9      0       0       0       0       0       0
Step 10     0       0       0       0       0       0
Step 11     0       0       0       0       0       0
Step 12     0       0       0       0       0       0
Step 13     0       0       0       0       0       0
Step 14     0       0       0       0       0       0
Step 15     0       0       0       0       0       0
Step 16     0       0       0       0       0       0
Step 17     0       0       0       0       0       0
Step 18     0       0       0       0       0       0
Step 19     0       0       0       0       0       0
Step 20     0       0       0       0       0       0
Step 21     0       0       0       0       0       0
Step 22     0       0       0       0       0       0
Step 23     730     1255    1992    2102    2542    3016
Step 24     58683   65535   65535   65535   65535   65535
Step 25     65535   65535   65535   65535   65535   65535
Step 26     65535   65535   65535   65535   65535   65535
Step 27     65535   65535   65535   65535   65535   65535
Step 28     65535   65535   65535   65535   65535   65535
Step 29     65535   65535   65535   65535   65535   65535
Step 30     21641   31656   48217   61009   65535   65535
Step 31     482     1059    1200    1125    1242    1521
Step 32     15      13      15      15      12      13
Step 33     0       0       0       0       0       0
Step 34     0       0       0       0       0       0
Step 35     0       0       0       0       0       0
Step 36     0       0       0       0       0       0
Step 37     0       0       0       0       0       0
Step 38     0       0       0       0       0       0
Step 39     0       0       0       0       0       0
Step 40     0       0       0       0       0       0
Step 41     0       0       0       0       0       0
Step 42     0       0       0       0       0       0
Step 43     0       0       0       0       0       0
Step 44     0       0       0       0       0       0
Step 45     0       0       0       0       0       0
Step 46     0       0       0       0       0       0
Step 47     0       0       0       0       0       0
Step 48     0       0       0       0       0       0
Step 49     0       0       0       0       0       0
Step 50     0       0       0       0       0       0
Step 51     0       0       0       0       0       0
Step 52     0       0       0       0       0       0
Step 53     0       0       0       0       0       0
Step 54     1       2       30      19      43      41
Step 55     11613   17089   21831   23425   28773   34745
Step 56     65535   65535   65535   65535   65535   65535
Step 57     65535   65535   65535   65535   65535   65535
Step 58     65535   65535   65535   65535   65535   65535
Step 59     65535   65535   65535   65535   65535   65535
Step 60     65535   65535   65535   65535   65535   65535
Step 61     65535   65535   65535   65535   65535   65535
Step 62     10960   19796   33150   36818   41810   52888
Step 63     27      95      149     142     151     221
Step 64     0       0       0       0       0       0
Step 65     0       0       0       0       0       0
Step 66     0       0       0       0       0       0
Step 67     0       0       0       0       0       0
Step 68     0       0       0       0       0       0
Step 69     0       0       0       0       0       0
Step 70     0       0       0       0       0       0
Step 71     0       0       0       0       0       0
Step 72     0       0       0       0       0       0
Step 73     0       0       0       0       0       0
Step 74     0       0       0       0       0       0
Step 75     0       0       0       0       0       0
Step 76     0       0       0       0       0       0
Step 77     0       0       0       0       0       0
Step 78     0       0       0       0       0       0
Step 79     0       0       0       0       0       0
Step 80     0       0       0       0       0       0
Step 81     0       0       0       0       0       0
Step 82     0       0       0       0       0       0
Step 83     0       0       0       0       0       0
Step 84     0       0       0       0       0       0
Step 85     0       0       0       0       0       0
Step 86     0       2       2       1       2       1
Step 87     531     900     1095    1359    1819    1877
Step 88     53667   65535   65535   65535   65535   65535
Step 89     65535   65535   65535   65535   65535   65535
Step 90     65535   65535   65535   65535   65535   65535
Step 91     65535   65535   65535   65535   65535   65535
Step 92     65535   65535   65535   65535   65535   65535
Step 93     65535   65535   65535   65535   65535   65535
Step 94     20475   37708   61380   65535   65535   65535
Step 95     220     593     960     874     1096    1380
Step 96     0       0       0       0       0       0
Step 97     0       0       0       0       0       0
Step 98     0       0       0       0       0       0
Step 99     0       0       0       0       0       0
```

The output above reports the number of PRBS errors counted over a short time interval for each of the 6 channels, for each step of the PPM controller. From this output we see that the period of the noisy zones is about 32 steps, each of which is two native PPM controller steps. The transmitter output is being applied to the D flipflop, which is being clocked at 10GHz; thus we see that the native step size of the PPM controller is about 100ps/64 = 1.56ps/step. We can also see that the quiet zones are 21/32 = 65% of a bit time.

We can repeat this with the HMC856 delay setting (which adjusts the phase between the DFF output and the ADC clock)

```
000000029X                                  //The command to sweep the 856 delay
//The MicroBlaze response
Stepping HMC856 through 32 steps
Step 0      6884    14570   23436   34943   48536   59851
Step 1      0       0       0       0       0       0
Step 2      0       0       0       0       0       0
Step 3      0       0       0       0       0       0
Step 4      0       0       0       0       0       0
Step 5      0       0       0       0       0       0
Step 6      0       0       0       0       0       0
Step 7      0       0       0       0       0       0
Step 8      0       0       0       0       0       0
Step 9      0       0       0       0       0       0
Step 10     0       0       0       0       0       0
Step 11     0       0       0       0       0       0
Step 12     0       0       0       0       0       0
Step 13     0       0       0       0       0       0
Step 14     0       0       0       0       0       0
Step 15     0       0       0       0       0       0
Step 16     0       0       0       65535   65535   65535
Step 17     0       0       0       52634   65535   65535
Step 18     0       0       0       0       0       0
Step 19     0       0       0       0       0       0
Step 20     0       0       0       0       0       0
Step 21     0       0       0       0       0       0
Step 22     0       0       0       0       0       0
Step 23     0       0       0       0       0       0
Step 24     0       0       0       0       0       0
Step 25     0       0       0       0       0       0
Step 26     0       0       0       0       0       0
Step 27     0       0       0       0       0       0
Step 28     0       0       0       0       0       0
Step 29     0       0       0       0       0       0
Step 30     0       0       0       0       0       0
Step 31     0       0       0       0       0       0
```

This sweep through all 32 settings shows that there are only 3 codes which produce errors: 0, 16, and 17. The nominal step size is 3 ps, so the 16 steps between the noisy zones are in agreement with the 20GHz clock frequency of the ADC's internal registers. I found that this behavior was extremely stable, and left the delay setting at 8 for all tests.

The processor can also be commanded to sweep the PPM delay through the error-free zone, find the edge, reverse direction to find the other edge, then back up to the middle of the error-free zone. A typical response is seen here:

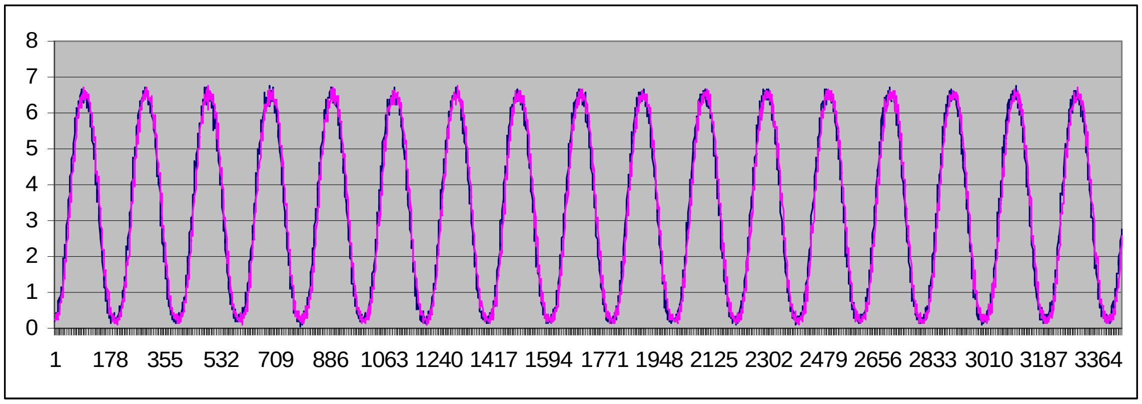
```
000000039X                                  //The command to adjust PPM delay
//The MicroBlaze response
Adjusting PPM delay
Sweeping positive
Step 0      0       0       0       0       0       0
Step 1      0       0       0       0       0       0
Step 2      0       0       0       0       0       0
Step 3      0       0       0       0       0       0
Step 4      0       0       0       0       0       0
Step 5      0       0       0       0       0       0
Step 6      0       0       0       0       0       0
Step 7      0       0       0       0       0       0
Step 8      0       0       0       0       0       0
Step 9      0       0       0       0       0       0
Step 10     0       0       0       0       0       0
Step 11     0       0       0       0       0       0
Step 12     0       0       0       0       0       0
Step 13     0       0       0       0       0       0
Step 14     0       0       0       0       0       0
Step 15     0       0       0       0       0       0
Step 16     0       0       3       3       7       6
state = 1
Sweeping negative
Step 0      159     1170    2554    3242    4302    5830
Step 1      6606    19445   35937   49476   62294   65535
Step 2      574     715     2410    4742    6968    8433
Step 3      0       0       7       36      42      37
Step 4      0       0       0       0       0       0
state = 2
Sweeping negative
```

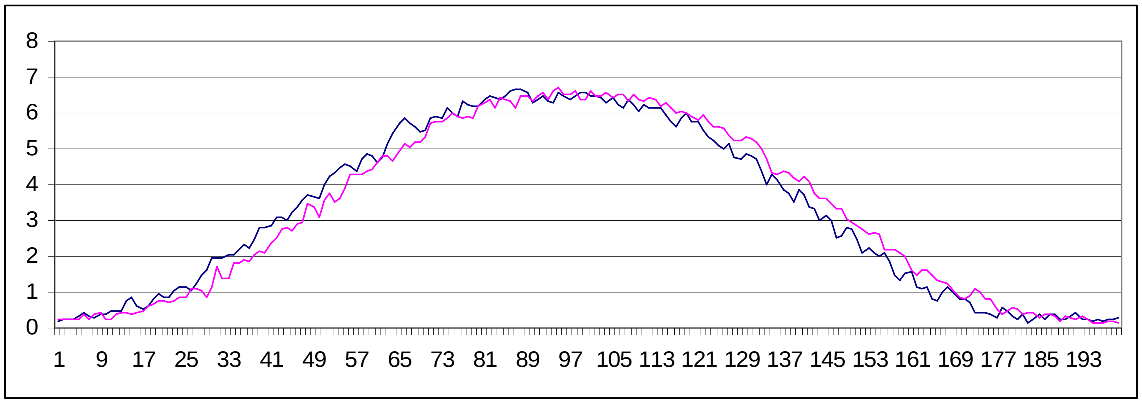
```
Step 0      0     0     0     0     0     0
Step 1      0     0     0     0     0     0
Step 2      0     0     0     0     0     0
Step 3      0     0     0     0     0     0
Step 4      0     0     0     0     0     0
Step 5      0     0     0     0     0     0
Step 6      0     0     0     0     0     0
Step 7      0     0     0     0     0     0
Step 8      0     0     0     0     0     0
Step 9      0     0     0     0     0     0
Step 10     0     0     0     0     0     0
Step 11     0     0     0     0     0     0
Step 12     0     0     0     0     0     0
Step 13     0     0     0     0     0     0
Step 14     0     0     0     0     0     0
Step 15     0     0     0     0     0     0
Step 16     0     0     0     0     0     0
Step 17     0     0     0     0     0     0
Step 18     0     0     0     0     0     0
Step 19     0     0     0     0     0     0
Step 20     0     0     0     0     0     0
Step 21     72    66    74    105   111   243
state = 3
```

This process is found to be completely stable and repeatable. Note that this command can be invoked during data acquisition without any expectation that the data alignment will be lost. We are just varying the modulation phase until a few errors are detected, then reversing. There is no reason to fear that any receiver will lose lock. No data will be acquired during this process (which takes on the order of 100us if the serial output is suppressed) but the data will still be aligned when the process is complete.

The processor can do the rest of the data alignment in response to a single command, stepping through the process in about 100us. After power on, the entire process of data alignment consists of sending the command to "Adjust PPM Controller" and then the command to "Align Data". After this, the "Adjust PPM Controller" command can be sent to measure and adjust for any shift in clock phase, perhaps due to system warmup or other factors.

To test the long-term stability of the data alignment I aligned the data, applied a 100MHz, full-scale, sinewave plus 0.8LSB rms of wideband noise and allowed the system to run for 6 hours (I found that my run time was limited by the fact that the design was compiled under the "hardware evaluation" license for the 10Gb/s Ethernet MAC core; this license causes the core to stop functioning after 8 or so hours. I'll recompile with a full license when I have access to it). The 100MHz signal was phase-locked to the 20GHz clock source, and the FPGA sample interval (the time between time records being grabbed by the FPGA) was set to 320us (any multiple of the 10ns signal period would do; this causes the signal to stay stationary with respect to the sample interval) . I allowed the two generators to come to thermal equilibrium for several hours before starting the test. I took data sets at the beginning and end of the test, taking 50 waveforms and averaging them at each test point. I then plotted the two waveforms to see any data shift.





Six-hour interface test, waveforms averaged at beginning and end

The waveform appears to have shifted about 200ps over the period of the test, which is likely an analog drift somewhere in the system. The data samples are assembled in sets of 80 samples internal to the FPGA, so it's unlikely to find a shift of a few samples resulting from any sort of interface glitch.

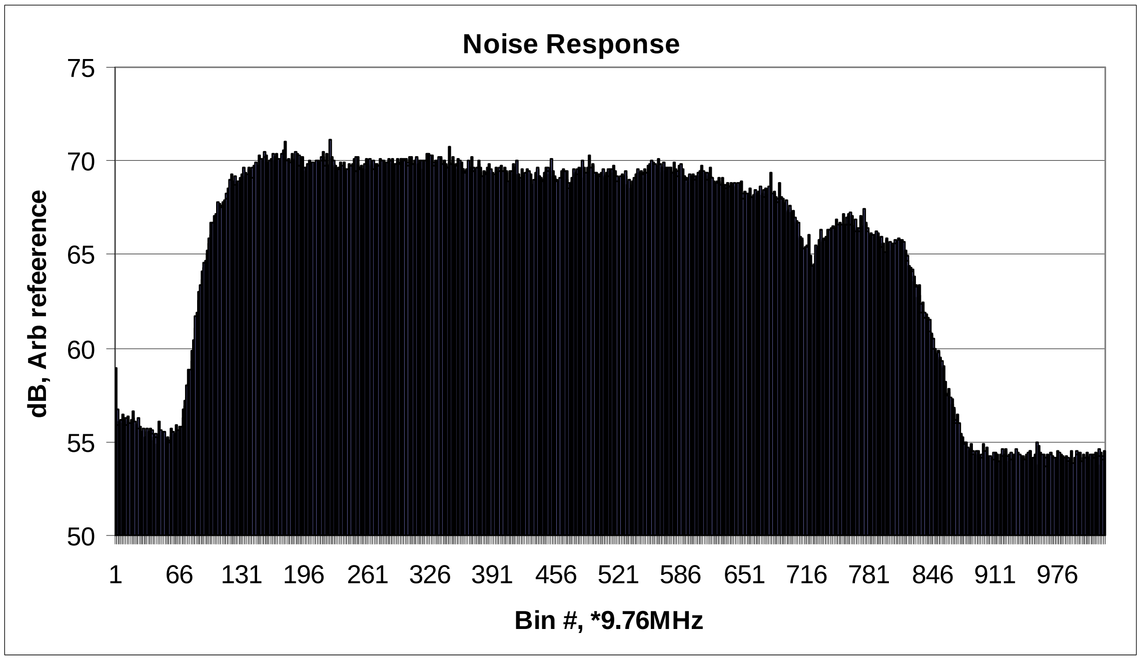
Data ordering

The data come out of the ADC demultiplexed into two 4-bit busses designated X and Y. From the datasheet I believed that the ordering of these two sets of samples (whether X[n] comes before or after Y[n]) was indeterminate and would need to be determined from examining the sampled data, either by applying a periodic signal, or by analyzing wide-band, but low-pass filtered, noise. But in fact, in all tests the X data is found to precede the Y data. This may possibly not be the case with other HMC5913 chips.

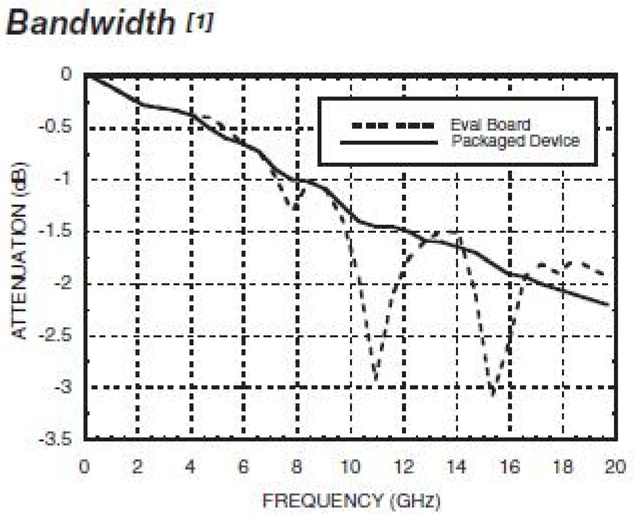
ADC dynamic tests

Noise tests

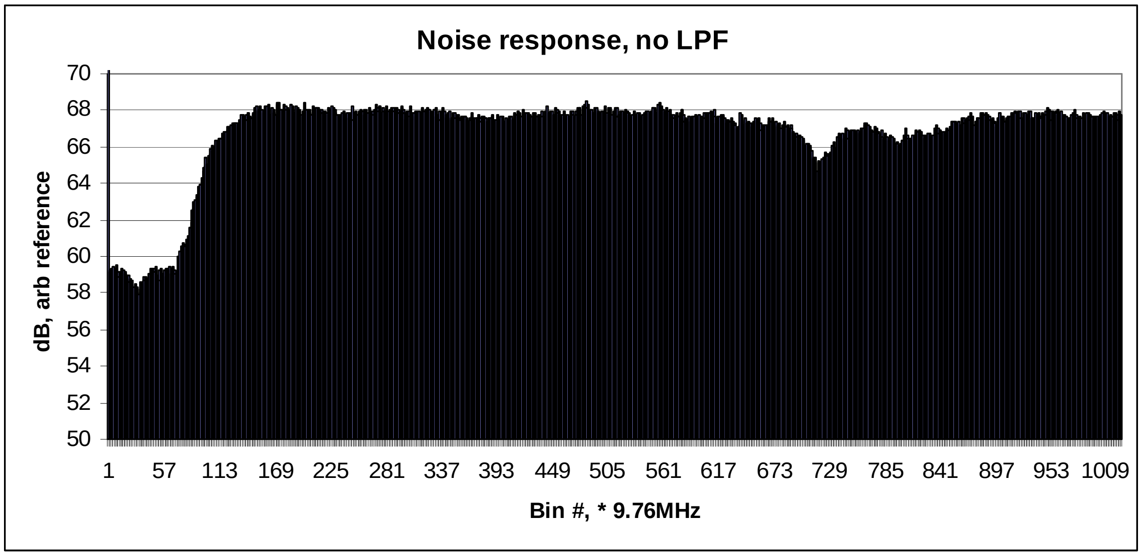
I connected the noise generator to the ADC board via the VLF-7200 LPF and adjusted the noise level to about 1.6 LSBs rms, as determined by capturing and analyzing a few waveforms. I acquired data, computed a 2048-point power spectrum, and averaged the results for 1000 such records.



The general shape of the response (the 1.3 to 8GHz passband) is that of the noise source and the LPF, with the noise in the stopband determined by the quantization noise of a 3-bit ADC. I don't know the source of the dip around 7.5GHz, though I note that the Hittite datasheet response for shows a similar dip (which they attribute to the PCB):



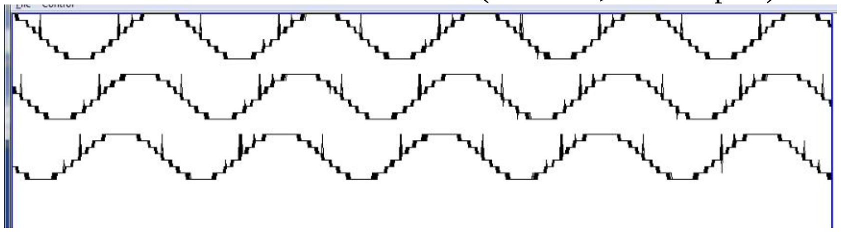
Here is the same test, with the LPF removed (again having adjusted the noise level to about 1.6 LSBs rms)

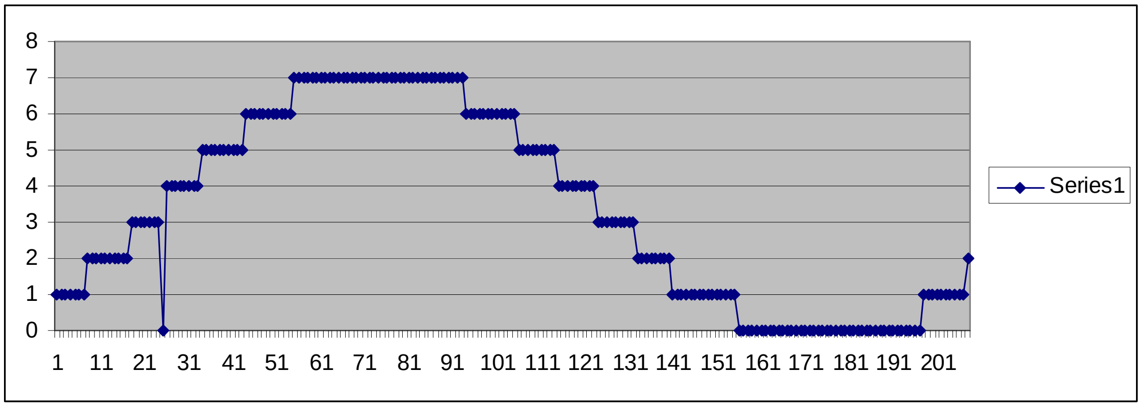


We can expect that the excess noise above Fsample/2 (the generator has a bandwidth to 18GHz) has aliased into the low frequency bins.

Sine wave tests

With the noise source attenuator set to a high value and a full-scale sinewave applied via the LPF, we can see the time waveforms displayed in the GUI. Since the 100MHz source is phase-locked to the generator, and the sample interval set to a multiple of 10ns, the waveforms stack up, and individual errors can be seen in the converted data. Here is an accumulation of about 50 waveforms (thus 150,000 samples):



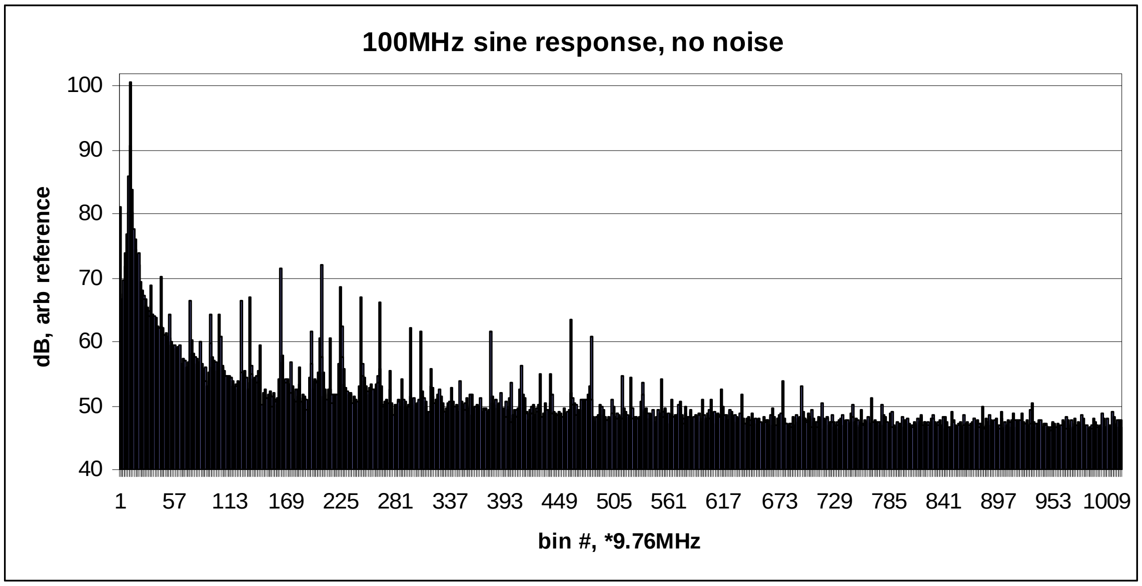


sparkle code

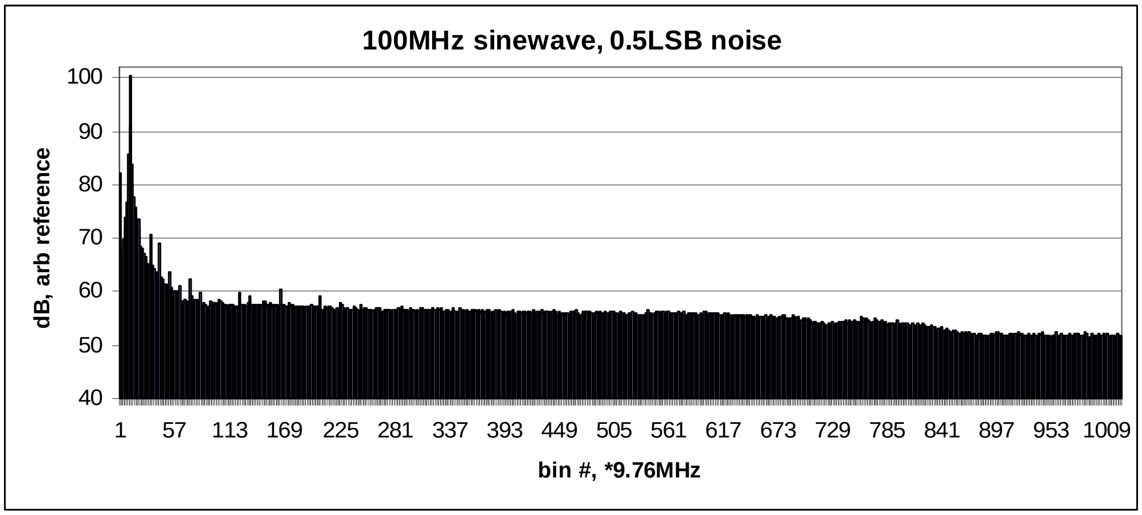
I believe the glitches are "sparkle codes" often seen in flash converters; Hittite reports that others have seen these. The frequency of occurrence is exaggerated in the above display due to the infinite persistence; inspection of the waveforms indicate that they occur about once in 3000 points for this waveform, typically at the 011 to 100 or 100 to

011 code transition (but not always). I haven't found any way to change the prevalence of these codes, but am still trying.

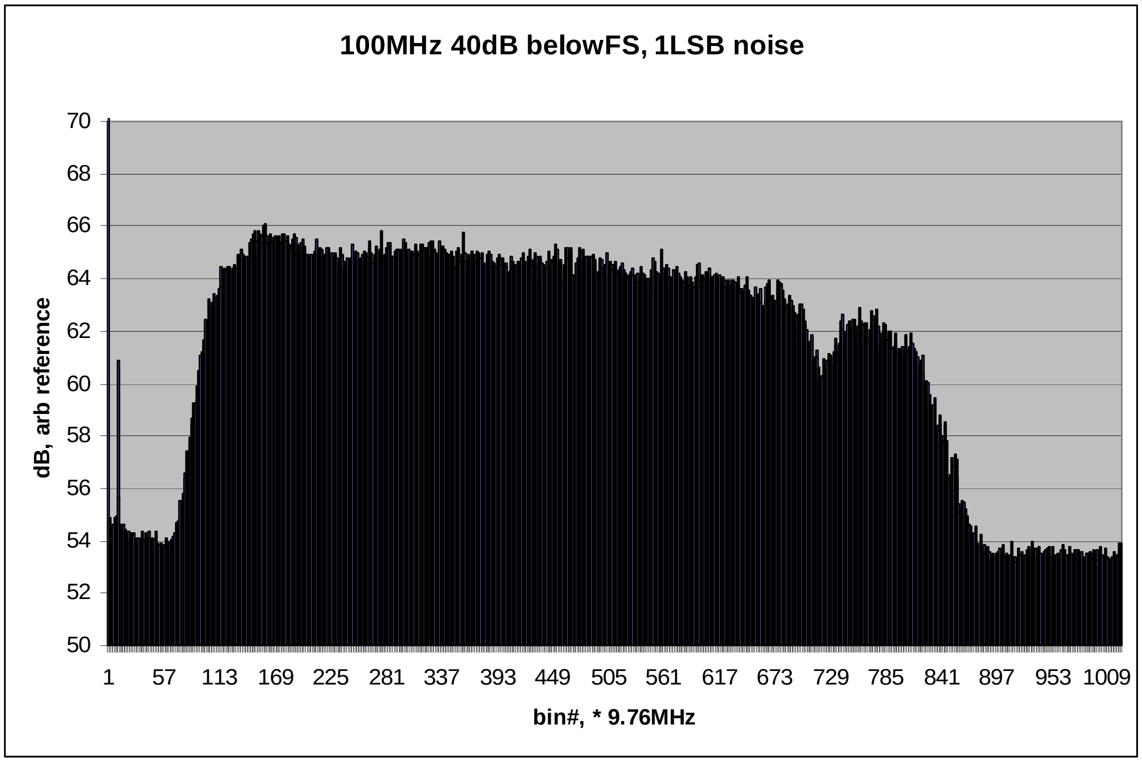
A power spectrum (2048 point) of the above waveform looks like this. Largest spur in this test is at 2.01GHz and is about 29dB down from the fundamental, consistent with HMC5913 datasheet expectations.



When we add 0.5LSB of noise (again with the 7.2GHz LPF in place) the spurs are suppressed:



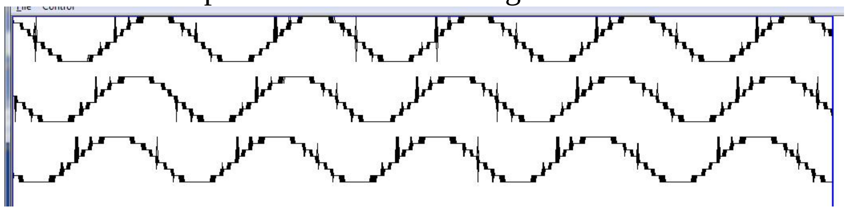
With the LPF, and the 100MHz buried in noise (switched in a 40dB attenuator in the 100MHz path):



Thermal test

To demonstrate stability of the interface in the case of widely varying FPGA power dissipation (which could occur in response to changes in applied data, and which could cause the phase of the modulating signal to shift for thermal reasons, and so degrade data reception), I allowed the system to come to equilibrium, aligned data using the automated process, and then blocked the FPGA fan. I read FPGA temperature via the MicroBlaze and monitored the signal reception (of a noiseless 100MHz sinewave) as the FPGA heated up. I swept the PPM controller to find the edges of the error-free zones before and after warmup to see how much phase shift occurs.

Over the course of this test (about 10 minutes) the die temperature rose from 43C to 66C. The data reception continued unchanged:



The logfile below shows the results of the PPM sweeps. The initial PPM scan shows that the phase setting is 17 steps away from the noisy edge. After the temperature had risen to 66C, a repeat scan shows that the phase setting is now 19 steps away from the noisy edge, indicating a shift of just a few steps, still a robust setting for the phase controller. The automatic adjustment routine could also be invoked at any time, with just the loss of about 100us of data, to re-center the phase setting.

```
Temperature 43
000000039X                    //The "Sweep PPM Phase" command
```

```
Adjusting PPM delay
Sweeping positive
Step 0      0       0       0       0       0       0
Step 1      0       0       0       0       0       0
Step 2      0       0       0       0       0       0
Step 3      0       0       0       0       0       0
Step 4      0       0       0       0       0       0
Step 5      0       0       0       0       0       0
Step 6      0       0       0       0       0       0
Step 7      0       0       0       0       0       0
Step 8      0       0       0       0       0       0
Step 9      0       0       0       0       0       0
Step 10     0       0       0       0       0       0
Step 11     0       0       0       0       0       0
Step 12     0       0       0       0       0       0
Step 13     0       0       0       0       0       0
Step 14     0       0       0       0       0       0
Step 15     0       0       0       0       0       0
Step 16     0       0       0       0       0       0
Step 17     10      36      47      95      132     237
state = 1
Sweeping negative
Step 0      229     496     927     1354    1686    2145
Step 1      302     686     1279    1912    2501    3211
Step 2      200     443     865     1168    1433    1838
Step 3      59      142     247     341     447     482
Step 4      0       0       0       0       3       4
Step 5      0       0       0       0       0       0
state = 2
Sweeping negative
Step 0      0       0       0       0       0       0
Step 1      0       0       0       0       0       0
Step 2      0       0       0       0       0       0
Step 3      0       0       0       0       0       0
Step 4      0       0       0       0       0       0
Step 5      0       0       0       0       0       0
Step 6      0       0       0       0       0       0
Step 7      0       0       0       0       0       0
Step 8      0       0       0       0       0       0
Step 9      0       0       0       0       0       0
Step 10     0       0       0       0       0       0
Step 11     0       0       0       0       0       0
Step 12     0       0       0       0       0       0
Step 13     0       0       0       0       0       0
Step 14     0       0       0       0       0       0
Step 15     0       0       0       0       0       0
Step 16     0       0       0       0       0       0
Step 17     0       0       0       0       0       0
Step 18     0       0       0       0       0       0
Step 19     0       0       0       0       0       0
Step 20     0       0       0       0       0       0
Step 21     0       0       0       0       0       0
Step 22     0       0       0       0       0       0
Step 23     8       54      112     126     161     199
state = 3
000000059X          //The "Report Temperature" command
Temperature 44
000000059X
Temperature 47
000000059X
Temperature 50
000000059X
Temperature 56
000000059X
Temperature 59
00000059X
000000059X
Temperature 62
000000059X
Temperature 63
```

```
000000059X
Temperature 64
000000059X
Temperature 65
000000059X
Temperature 66
000000039X
Adjusting PPM delay
Sweeping positive
Step 0      0     0     0     0     0     0
Step 1      0     0     0     0     0     0
Step 2      0     0     0     0     0     0
Step 3      0     0     0     0     0     0
Step 4      0     0     0     0     0     0
Step 5      0     0     0     0     0     0
Step 6      0     0     0     0     0     0
Step 7      0     0     0     0     0     0
Step 8      0     0     0     0     0     0
Step 9      0     0     0     0     0     0
Step 10     0     0     0     0     0     0
Step 11     0     0     0     0     0     0
Step 12     0     0     0     0     0     0
Step 13     0     0     0     0     0     0
Step 14     0     0     0     0     0     0
Step 15     0     0     0     0     0     0
Step 16     0     0     0     0     0     0
Step 17     0     0     0     0     0     0
Step 18     0     0     0     0     0     0
Step 19     0     7     15    22    28    28
state = 1
Sweeping negative
Step 0      80    322   455   533   591   703
Step 1      323   848   1158  1558  1992  2279
Step 2      86    197   402   621   718   964
Step 3      2     4     14    10    13    15
Step 4      0     0     0     0     0     0
state = 2
Sweeping negative
Step 0      0     0     0     0     0     0
Step 1      0     0     0     0     0     0
Step 2      0     0     0     0     0     0
Step 3      0     0     0     0     0     0
Step 4      0     0     0     0     0     0
Step 5      0     0     0     0     0     0
Step 6      0     0     0     0     0     0
Step 7      0     0     0     0     0     0
Step 8      0     0     0     0     0     0
Step 9      0     0     0     0     0     0
Step 10     0     0     0     0     0     0
Step 11     0     0     0     0     0     0
Step 12     0     0     0     0     0     0
Step 13     0     0     0     0     0     0
Step 14     0     0     0     0     0     0
Step 15     0     0     0     0     0     0
Step 16     0     0     0     0     0     0
Step 17     0     0     0     0     0     0
Step 18     0     0     0     0     0     0
Step 19     0     0     0     0     0     0
Step 20     0     0     0     0     0     0
Step 21     0     0     0     0     0     0
Step 22     0     0     0     0     0     0
Step 23     1     1     1     0     0     4
state = 3
```

Conclusions

The interface via the FMC connector and GTH receivers using the GTH transmitter as a modulation source is extremely robust, and the ADC performs in accordance with datasheet expectations. Sparkle codes occur with a surprising frequency, but don't seem to degrade dynamic performance appreciably

Next steps

The PCB needs to be revised to fix the two problems referred to above. In compiling the custom GTH core I kept all ten of the available GTH receiver cores, to insure that whatever pairs are chosen for the OVERRANGE bits would be workable. I will incorporate linear regulators to provide -5 and -3.3 (post-regulators to the switchers).

The robustness of this interface makes me wonder if the D flipflop, delay chip, and crosspoint switch are even necessary. Parts cost for these totals about $400, which may be immaterial, but the power dissipation totals 1.3W.

I'd like to test the system at higher clock rates- the ADC is specified at 26GSPS. The FPGA on the VC709 is a -2 device, whose GTH transceivers are rated at 11.3 Gb/s, rather than the maximum 13.1Gb/s of the -3 device. So it would be interesting to test at 22GHz. And it may well work at higher rates, since the transceivers are expecting to operate over lossy channels, and our signals are very high quality. The HMC-T2100 generator which I used in these tests stops at 20GHz.

The ADC chip dissipates greater than 4W and gets very hot. In all my testing I saw no problems thermally, but device life would likely be enhanced if the chip were kept cooler. The ADC chip has a bottom-surface power pad which is soldered to the PCB and connected with thermal vias to a bottom-surface copper pour. This surface is at -5v, but could be electrically insulated but thermally coupled to a machined heatsink (which might be incorporated into an overall chassis for the two-board combination).

The balun is expensive ($1600) and unnecessarily wideband; I was told that Marki would be making a cheaper board-mount unit that we could put on the board.