```
#*******************************************************************
#
# turnon_notes.txt module
#
#*******************************************************************
#
# VCL Confidential Copyright © 2009 UC Davis, ECE Department
#
#*******************************************************************
#
# created on:   04/28/2009
# created by:   jwwebb
# last edit on: $DateTime: $
# last edit by: $Author: $
# revision:     $Revision: $
# comments:     Generated.
#
# board name:           MSEE Thesis Measurement Board
# board number:         P342
# board revision:       001
#
#*******************************************************************

#−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
# T U R N  O N  N O T E S:
#−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
1.  No shorts on power supplies.
        − Board Powers Up Properly.
        − All supplies within spec.

2.  Control FPGA Programs via JTAG successfully.
        − Using Xilinx iMPACT Software.

3.  SPI Configuration PROM programs via JTAG successfully.
        − Using Xilinx iMPACT Software.

4.  Data Path FPGA Programs via JTAG successfully.
        − Using Xilinx iMPACT Software.
        − Operates at about 42 degrees C with Fan Sink.
        − Operates at about 62 degrees C without Heat Sink.

5.  Control FPGA RS−232 (CP2102) interfaces functions correctly.
        − Operates a 115.2kBaud
        − Read/Write control of Control FPGA registers.

6.  Control FPGA controls Data Path FPGA successfully via "ilb_master".
        − LEDs turn on and off.
        − 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
```

- Verilog HDL Driver verified in simulation.
- Doesn’t seem to be configured properly.
- Double−check config registers to verify VCO is turned on properly.
- It turns out there were a couple of problems.
  - a. 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 & 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.
  - b. 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.
- 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.
    - Verilog HDL Driver verified in simulation.
    - Add read capability in hdl driver to verify functionality.
      - a. See item 14.

14. Ported over MicroBlaze design from work project and verified that the DDR SDRAM is working correctly.
    - Bootloader Application successfully read out the MicroBlaze application from the SPI Configuration PROM and transfered the data to the DDR SDRAM.
      - a. The Bootloader then successfully jumped to the application address 0x8c000000 and the MicroBlaze application began executing out of the DDR SDRAM.
    - The MicroBlaze application successfully read the DPIMAGE.bin file from the microSD card and configured the Data Path FPGA via a serial interface.
      - a. The Data Path FPGA successfully programed.
      - b. The microSD card functions properly.
    - The MicroBlaze application successfully communicates with the Data Path FPGA via the ILB serial interface.
    - The MicroBlaze application successfully transfered a waveform from the microSD card to the Data Path FPGA.
    - For some reason the I2C Fan Controllers don’t seem to be working.
      - a. I get error messages pertaining to the iicWriteRegister.
        - i. 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.
      - b. Have some fans made up with connectors for testing the fan controllers.
    - AD9516 doesn’t seem to be working with MicroBlaze application.
      - a. Probe Serial Data and Clock to see if data is being transmitted at power−up.
        - i. 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.

```
            b. Create a debug command to configure AD9516 upon command.
                i. Completed. See command 30.
            c. Create a debug command to read AD9516 register.
                i. Completed. See command 29.
            d. Create a debug command to write AD9516 register.
                i. Completed. See command 28.
        - Add DAC5682Z peripheral configuration and control to current
          MicroBlaze application and system.mhs/mss files.
            a. Wrote SPI EDK Peripheral to control DAC5682Z via SPI Interface.
            b. Initialization routine successfully completes.
            c. Read/Write access correctly read/write to various registers.
            d. Debug Commands:
                i.   Command 31: write command.
                ii.  Command 32: read command.
                iii. Command 33: initialization command.
        - Use ChipScope Pro to verify SRAM and Block RAM pattern
          playback inside of Data Path FPGA.
            a. Block RAM Patterns Play back successfully in ChipScope Pro.
            b. QDR-II SRAM Patterns Play back successfully in ChipScope Pro.
        - The 6 temperature sensors appear to be functioning properly.
            a. Temp: 30.5 C, 27.75 C, 26.75 C, 29.25 C, 28.25 C, 29.50 C

15. DAC5682Z High-Speed 16-bit DAC Data Path.
        - 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.
            a. 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.
            b. 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:
                i.   Remove R88, R89, R530, and R531.
                ii.  Solder a wire from pad 1 of R88 to pad 2 of R531.
                iii. Solder a wire from pad 1 of R89 to pad 2 of R530.
               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.
```

- 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.
- 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.
  a. 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.
  b. 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.
     i. 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.
- I implemented a routine that configures the DAC5682Z for Digital Self Test, and it passed the self test. 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. 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.

  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).
       − Today a light when on, and 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.
                       i.  28 191 c0
                       ii. 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

                                       R1
                       LPF >−−−−−−−−/\/\/\/\−−−−
                       Pin 64   |              |
                                |              |
                              −−−−−  C2     −−−−−−  C1
                              −−−−−         −−−−−−
                                |              |
                                |              |
                               −−−            −−−
                               \ /            \ /
                               GND            GND

               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.
          − 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.
          − 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.
          − 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.
          − 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.
          − 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.

16. 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
                        − Vicm: 0.9V (spec), 0.9V (meas)
                        − Vdiff: 0.5V to 1V (spec), 0.328V (meas)
                        a. Changed Vodiff on AD9516 from 600mV to 960mV.
                        b. New Measurements; Vidiff:  580mV
                ii.  DAC5682Z − Measured at AD9516
                        − Vocm: 1.335V (spec), 2.09V (meas)
                        − Vodiff: 600mV (spec), 540mV (meas)
                        a. Changed Vodiff on AD9516 from 600mV to 960mV.
                        b. New Measurements; Vodiff: 750mV
                iii. ADS5463 − Measured at ADS5463
                        − Vicm: 2.4V (spec), 2.41V (meas)
                        − Vdiff: 0.5V to 5V (spec), 600mV (meas)
                iv.  ADS5463 − Measured at AD9516
                        − Vocm: 0.920V (spec), 2.0V (meas)
                        − Vodiff: 960mV (spec), 760mV (meas)
                        a. 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 clk/128 and clk/32.

17. Take measurements of Trigger Output, Trigger Input, and Auxiliary Input.
        - 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.
                a. The Pattern Trigger functions properly.
                b. The Divided Clock Trigger functions properly at clk,
                   clk/2, clk/4, and clk/8.

#-----------------------------------------------------------
# P C B  M O D I F I C A T I O N S:
#-----------------------------------------------------------
1. Load the following Data Path FPGA Configuration Resistors:
        - R569; CUST_INIT_B
        - R568; CUST_CFG_DONE
        - R566; CUST_PROG_B
        - R567; CUST_CCLK
2. DAC5682Z Channel A/B swap:
        - Remove R88, R89, R530, and R531.
        - Solder a wire from pad 1 of R88 to pad 2 of R531.
        - Solder a wire from pad 1 of R89 to pad 2 of R530.

#------------------------------------------------------------
# P C B  A S S E M B L Y   S T A T U S / P R O B L E M S:
#------------------------------------------------------------

SN1.1:
        1. Load Resistors Listed Above. (Completed)
        2. C15 package (0508) was standing on its edge, rather
           than lying flat.
SN1.2:
        1. Load Resistors Listed Above. (Completed)
SN1.3:
        1. Load Resistors Listed Above. (Completed)
        2. J40 SMA Connector Center Pin cracked solder joint at
           trace landing.
                a. Reflowed center pin to make good contact.
        3. U45 VDD Pin 5 was not soldered down (i.e., floating).
                a. Temporarily pressed pin down to make contact.
                b. Re-solder joint to make good connection.
SN1.4:
        1. Load Resistors Listed Above. (Completed)
        2. Perform DAC5682Z Channel A/B swap as described
           above. (Completed).
SN1.5:
        1. Load Resistors Listed Above. (Completed)


#----------------------------------------------------------
# P C B  F A B R I C A T I O N   P R O B L E M S:
#----------------------------------------------------------

1. Incorrect decal used for U42.
        a. Designed interposer board to fix decal.
```