1. **Mu2e Calorimeter Waveform Digitizer Prototype Electronics**

The Mu2e Calorimeter Waveform Digitizer Prototype Electronics (Cal\_WFD\_Proto) subsystem is the prototype version of an electronic printed circuit board that digitizes data from the Calorimeter Front End Electronics, and then sends it upstream to the Mu2e Trigger and Data Acquisition (DAQ). Specifically, the Cal\_WFD\_Proto digitizes, serializes, and sends data out of a fiber optic transceiver to the DAQ.

The Cal\_WFD\_Proto is comprised of two subsystems: Data Acquisition and Readout Controller. Data acquisition converts the analog signals to digital, zero-suppresses, adds metadata, and combines the channels into a single block of data. The Readout Controller serializes, translates the data into the correct protocol, and sends the data out the optical transceivers to the DAQ.

* 1. **Requirements**

Create a hardware software development platform that incorporates the Texas Instruments ADS58C48 Analog to Digital Converter (ADC), a Waveform Digitizer (WFD), and a Readout Controller into a single printed circuit board. The Cal\_WFD\_Proto will aid the development of VHDL coding, Slow Controls coding, and an understanding of what is needed to be done to have a functional production WFD and Readout Controller.

* 1. **Technical Design**
     1. **Hardware:**

Figure 1.2.1 shows the block diagram of the Cal\_WFD\_Proto. The Cal\_WFD\_Proto is an electronic printed circuit board that measures 25.4 cm wide x 25.4 cm high (10 in wide x 10 in high).

 Figure .2.1 – Block Diagram of Cal\_WFD\_Proto Board.

**Analog Inputs:**

The Cal\_WFD\_Proto has eight, differentially-ended, analog channels that are digitized. Each channel has two SMA connectors that provide the differentially-ended, analog signal to the board. The analog channels design was based on the Texas Instruments (TI) ADS58C48EVM, Evaluation Module.

The analog signal’s differential input voltage should be two volts peak-peak. Each channel goes through a Texas Instruments (TI) PGA870, programmable-gain amplifier that can be configured (via a dipswitch) for gains between -11.5 dB to +20 dB, in 0.5-dB steps. The signal path for each channel continues through a band-pass filter to the TI ADS58C48 Analog to Digital Converter. The TI ADS58C48 ADC is a quad channel, 200 MSPS, high performance, 11-bit Analog to Digital Converter. The digitized data goes into a Xilinx Spartan-6 Field Programmable Gate Array (FPGA).

**ADC EVM Connector:**

A Samtec, high speed ground plane socket is mounted on the bottom side of the Cal\_WFD\_Proto board to allow the TI ADS58C48EVM evaluation module to be used. The ADS58C48 has an additional four channels which also goes to the same Xilinx Spartan-6 FPA as the analog inputs. Using the ADS58C48EVM allows us to compare the signal chain in the analog inputs with this board.

**External Trigger Input:**

Digitizing the analog signals coming from the ADC, is started by a differentially-ended, external trigger clock. The external trigger clock is connected to the Cal\_WFD\_Proto using 2 SMA connectors. The first rising edge of the external trigger clock initiates the digitization of analog signals.

**Optical Transceivers:**

Optical transceivers on the board take the outputted, serialized data from the FGPA and convert it to an optical signal to be passed up to the Data Acquisition System (DAQ). Two optical transceivers allow for redundancy in the data path to the DAQ. In the event one of the transceivers fails, the other can continue sending data without having to repair or swap out the Cal\_WFD\_Proto board.

The other role for the optical transceivers is for slow controls communication. Slow controls communication will allow the Cal\_WFD\_Proto board to be configured by the DAQ remotely and be interrogated to monitor system voltages and other environmental issues.

The optical transceivers are a class 1, multimode, 850nm device. The data rate for the transceivers is 1.25 Gbps up to 2.5 Gbps, using gigabit Ethernet protocol with 8b/10b encoding.

**Mobile Low Power Dual Data Rate (LPDDR) SDRAM:**

A Micron, MT46H64M16LFBF, 1Gb, mobile, lpddr, sdram is attached to the FPGA. It can store up to one second of digitized data that can be buffered, if needed, before sent up to the DAQ. The mobile LPDDR SDRAM operates at 200 Mhz.

**Test Points:**

Test points and a couple of leds are connected to the FPGA. The leds can be used to indicate proper operation or other feedback. The test points help aid debugging the Cal\_WFD\_Proto board.

**CAN Transceiver Module:**

The CAN Transceiver module is a redundant communications path for the slow controls in the event that the fiber optics fail or the FPGA programming becomes corrupt. The CAN module is connected to a 16 bit flash based microcontroller that implements the CAN protocol. An external PC or controller communicates with the microcontroller through the CAN Interface.

**Temperature Sensor:**

A Texas Instruments LM82 Local Digital Temperature Sensor is attached to the microcontroller via a two-wire serial interface. The temperature sensor measures the board temperature. The two-wire interface supports both I2C and SMBus Interfaces. The I2C interface is used to send temperature information to the microcontroller.

**Microcontroller:**

A general purpose, 16 bit, flash based, microcontroller is connected to the FPGA. The microprocessor communicates with the FPGA to provide environmental data (Temperature) as well as configuration data (slow controls) through the CAN Interface. The microcontroller uses ANSI C as the programming language. It uses a 10 Mhz discreet crystal as the clock for operation.

**Field Programmable Gate Array (FPGA):**

The Xilinx Spartan-6 Field Programmable Gate Array (FPGA) has a few responsibilities:

* Translation, Zero-Suppression, Serializing, and outputting serial data to the optical transceivers.
* Default method for configuring the Cal\_WFD\_Proto board.
* Communicates with microcontroller for environmental data (temperature) and as a secondary path for slow controls.

The FPGA uses the VHDL programming language to implement the Waveform Digitizer and the Readout Controller. Separate clocks drive various aspects of the FPGA. One clock is used for the gigabit transceivers on the FPGA. Another clock is used for the main system clock. A third clock is provided for redundancy and to allow other parts of the FPGA to be driven from a different clock domain.

**Clocks:**

Clocks are generated on the Cal\_WFD\_Proto board. All clocks are differentially-ended and a/c coupled to each device. Four sets of clocks are generated on the board:

* ADC clocks – Clocks that drive the ADCs. This clock is generated by an ECS, Inc., ECS-LVDS33-2000-BN Oscillator. The oscillator generates a 200 Mhz, Low Voltage Differential Signal (LVDS) clock signal. An IDT, ICS854S006I, low skew, 1 to 6, differential-to-LVDS fanout buffer cleans up the clock signal before it drives the ADCs. The clock signal has at least 1ps RMS Jitter or better specification.
* FPGA Global Clocks –Clocks that drive the FPGA . One of the clock is the main system (global) clock for the FPGA. The other can be used for other parts of the FPGA, if needed. The generated clock signal is also an ECS, Inc., ECS-LVDS33-2000-BN Oscillator. The oscillator generates a 200 Mhz, LVDS clock signal. An IDT, ICS854S006I, low skew, 1 to 6, differential-to-LVDS fanout buffer cleans up the clock signal before it drives the FPGA. The clock signal has at least 1ps RMS Jitter or better specification.
* Multi-Gigabit Transceiver (MGT) Clocks – Clocks that drive the MGTs on the FPGA. The MGT serializes the data and sends it to the optical transceivers. Each MGT is driven from a separate clock.

The clocks are generated by an ECS, Inc., ECS-LVDS33-1250-BN Oscillator. The oscillator generates a 125 Mhz LVDS clock signal. A Micrel SY58606U, precision, 1 to 2 Current Mode Logic (CML) fanout buffer converts the logic from LVDS to CML, and cleans up the signal to at least 150 fs RMS Jitter or better.

* High Speed Clock – Clock to drive other potential high-speed logic in the FPGA. This clock uses an ECS, Inc., ECS-LVDS33-2000-BN Oscillator. The oscillator generates a 200 Mhz, LVDS clock signal. Another Micrel SY58606U, precision, 1 to 2 Current Mode Logic (CML) fanout buffer converts the logic from LVDS to CML, and cleans up the signal to at least 150 fs RMS Jitter or better.

**Power Distribution:**

Input voltage to the Cal\_WFD\_Proto board can be anywhere from 6 volts dc to 12 volts dc. Two, 15 amp, pico fuses on the board protects the board from any over-current faults. Each voltage has an led that indicates when the correct voltage is present. The TI Power Module, PMP6776 & the ADS58C48EVM were used as a reference when designing this power distribution circuit. Power supply sequencing is based on Xilinx recommendations for 7 series FPGAs. Remote sensing is used to provide more accurate output voltages at the point of load. Regulated power is generated locally, on the Cal\_WFD\_Proto board, from nine separate point-of-load power supplies:

* 1.2 Volts DC @ 6 Amps

The VCCINT voltage for the FPGA.

* 3.3 Volts DC @ 8 Amps

The VCCO, VCCAUX voltages for FPGA. Also JTAG, Microcontroller, CAN, & SFP voltages.

* 1.8 Volts DC @ 6 Amps

The LPDDR voltage, VCCO\_4.

* 1.8 Volts DC @ 1 Amp

The ADC analog voltage, for channels one through four.

* 1.8 Volts DC @ 1 Amp

The ADC digital voltage, for channels one through four.

* 1.8 Volts DC @ 1 Amp

The ADC analog voltage, for channels five through eight.

* 1.8 Volts DC @ 1 Amp

The ADC digital voltage, for channels five through eight.

* 1.2 Volts DC @ 6 Amps

The digital voltage for the MGTs on the FGPA.

* 5.0 Volts DC @ 6 Amp

The analog voltage for the programmable-gain amplifiers.

* + 1. **Manufactured Printed Circuit Board:**

Figure 1.2.2 shows the front side, manufactured Cal\_WFD\_Proto board without any components installed.

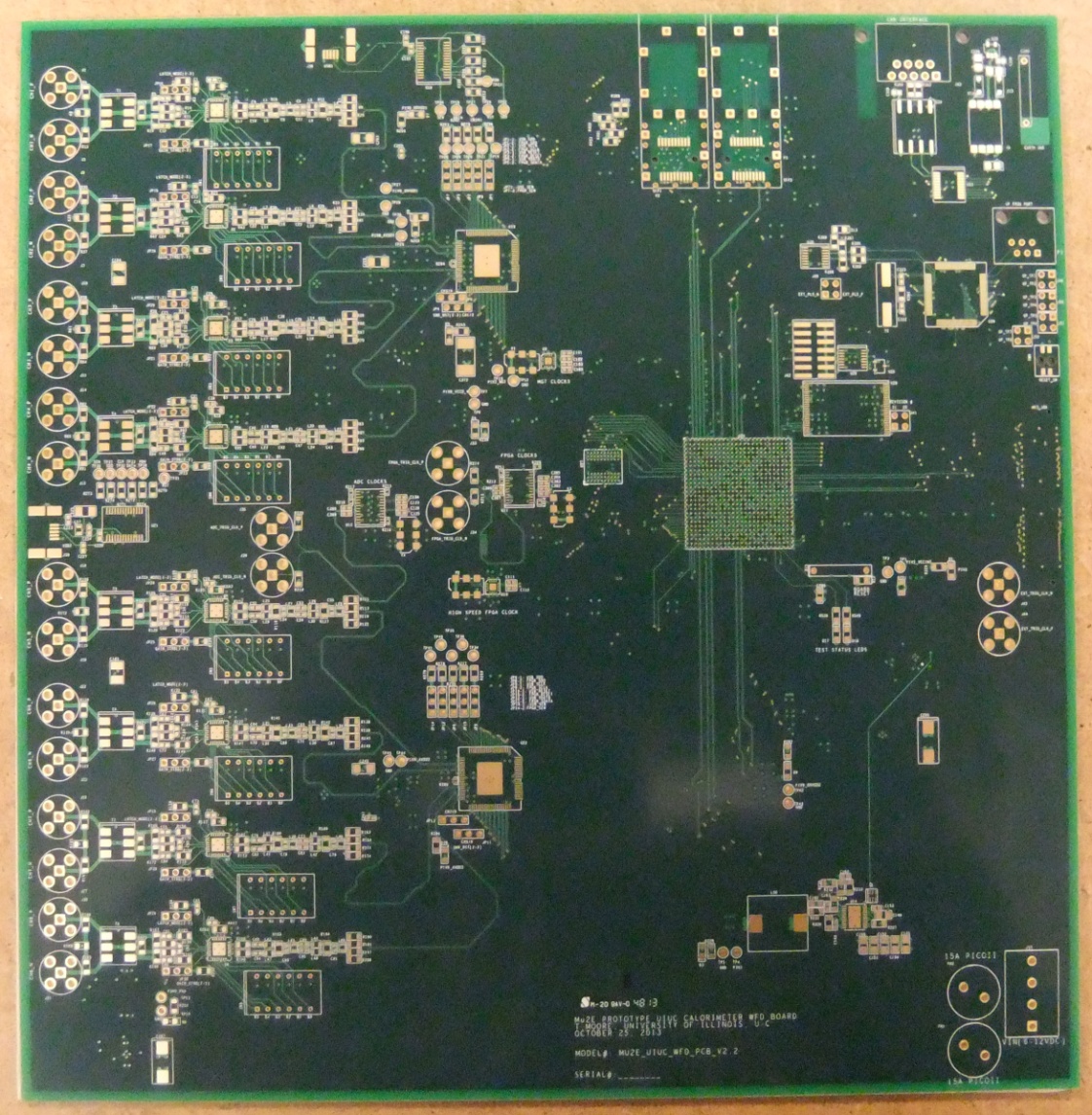


Figure 1.2.2 - Bare Manufactured PCB

* + 1. **FPGA Firmware:**

The FPGA firmware is written in the VHDL programming language using Xilinx ISE, Version 14.4. Simulations are done using Xilinx ISIM Version 14.4. Programming the Xilinx Spartan-6 is through a JTAG connector.

Firmware will be written in three stages:

* Stage 1: Write VHDL code to acquire, combine, serialize, and output the data through the fiber optic transceivers.
* Stage 2: Write VHDL code to receive the acquired data and compare it to the transmitted data to verify communication without any errors.
* Stage 3: Write VHDL code for the slow controls functionality.
  + - 1. **Stage 1: State Machine:**

The state machine is the controlling mechanism in the code and determines what tasks are to be executed in what order. The first revision of the state machine is used as a starting point for the code, with the assumption the code will be modified as new features get included.

There are thirteen states in the current version of the state machine: The code goes back to the Idle state after all 13 states are complete.

State 1 – Idle

Clear counters and control signals. Wait for External Trigger to start reading ADCs.

State 2 – Load ADC FIFOS

Acquire data from ADCs and store each channel into it’s own fifo.

State 3 – Load Channel 1 ADC Data

Move ADC channel #1 data into fifo that holds all channel data.

State 4 – Load Channel 2 ADC Data

Move ADC channel #2 data into fifo that holds all channel data.

State 5 – Load Channel 3 ADC Data

Move ADC channel #3 data into fifo that holds all channel data.

State 6 – Load Channel 4 ADC Data

Move ADC channel #4 data into fifo that holds all channel data.

State 7 – Load Channel 5 ADC Data

Move ADC channel #5 data into fifo that holds all channel data.

State 8 – Load Channel 6 ADC Data

Move ADC channel #6 data into fifo that holds all channel data.

State 9 – Load Channel 7 ADC Data

Move ADC channel #7 data into fifo that holds all channel data.

State 10 – Load Channel 8 ADC Data

Move ADC channel #8 data into fifo that holds all channel data.

State 11 – Send ‘Start Frame’ Character

Send 0x0b to optical transceiver to indicate the start of the data being transmitted.

State 12 – Send All 8 Channels of ADC Data

Send all 8 channels of data to optical transceiver.

State 13 – Send ‘End Frame’ Character

Send 0x0b to optical transceiver to indicate the end of the data being transmitted.

* + - 1. **Stage 2: State Machine:**

The 2nd stage state machine adds receiving the data from the optical transceiver and comparing it to the sent data. The state machine has 16 states. The code goes back to the Idle state after all 16 states are complete.

State 1 – Idle

Clear counters and control signals. Wait for External Trigger to start reading ADCs.

State 2 – Load ADC FIFOS

Acquire data from ADCs and store each channel into it’s own fifo.

State 3 – Load Channel 1 ADC Data

Move ADC channel #1 data into fifo, or SDRAM, that holds all channel data.

State 4 – Load Channel 2 ADC Data

Move ADC channel #2 data into fifo, or SDRAM, that holds all channel data.

State 5 – Load Channel 3 ADC Data

Move ADC channel #3 data into fifo, or SDRAM, that holds all channel data.

State 6 – Load Channel 4 ADC Data

Move ADC channel #4 data into fifo, or SDRAM, that holds all channel data.

State 7 – Load Channel 5 ADC Data

Move ADC channel #5 data into fifo, or SDRAM, that holds all channel data.

State 8 – Load Channel 6 ADC Data

Move ADC channel #6 data into fifo, or SDRAM, that holds all channel data.

State 9 – Load Channel 7 ADC Data

Move ADC channel #7 data into fifo, or SDRAM, that holds all channel data.

State 10 – Load Channel 8 ADC Data

Move ADC channel #8 data into fifo, or SDRAM, that holds all channel data.

State 11 – Copy ADC Data For Comparison

Copy all adc data into fifo, or SDRAM, for comparing transmitted data to received data.

State 12 – Start ‘Comparing Data’ Process

Start the process that compares the received data with the transmitted data and counts the errors.

State 13 – Send ‘Start Frame’ Character

Send 0x0b to optical transceiver to indicate the start of the data being transmitted.

State 14 – Send All 8 Channels of ADC Data

Send all 8 channels of data to optical transceiver.

State 15 – Send ‘End Frame’ Character

Send 0x0b to optical transceiver to indicate the end of the data being transmitted.

State 16 – Stop ‘Comparing Data’ Process

Stop the process that compares the received data with the transmitted data and output error count.

* + - 1. **Stage 3: Slow Controls:**

The 3rd stage will be to implement the slow controls. Both VHDL and C programs will need to be written; VHDL code for the FPGA and C code for the microcontroller. The slow control actions that we will use are:

* Read/Write Configuration Data:

1. Programmable Gains

2. ADCs

3. FPGAs

* Read Board Info:

1. Serial Number

2. Board Part Number

3. Board Version Number

4. Institution

5. Date of Manufacture

6. Contact Info

7. Board Temperature

* Read Firmware Info:

1. FPGA Part Number

2. FPGA Firmware Version Number

3. FPGA Firmware Design Date