

To the Graduate Council:

I am submitting herewith a thesis written by Zachary William Pannell entitled “Design of a Highly Constrained Test System for a 12-bit, 16-channel Wilkinson ADC.” I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree Master of Science, with a major in Electrical Engineering.

Benjamin Blalock

Major Professor

We have read this thesis
and recommend its acceptance:

Syed Islam

Jeremy Holleman

Accepted for the Council:

Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original Signatures are on file with official student records.)

Design of a Highly Constrained Test System for a 12-bit,
16-channel Wilkinson ADC

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Zachary William Pannell

December 2009

ABSTRACT

Outer space is a very harsh environment that can cause electronics to not operate as they were originally intended. Aside from the extreme amount of radiation found in space, temperatures can also change very dramatically in a relatively small time frame. In order to test electronics that will be used in this environment, they first need to be tested on Earth under replicated conditions. Vanderbilt University designed a dewar that allows devices to be tested at these extreme temperatures while being radiated. For this thesis, a test setup that met all of the dewar's constraints was designed that would allow a 12-bit, 16-channel analog-to-digital converter to be tested while inside.

Contents

1	Introduction	2
1.1	Remote Sensor Interface (RSI)	3
1.2	Analog-to-Digital Converters	3
1.2.1	ADC Parameters	4
1.2.2	Wilkinson ADC	5
2	FPGA	7
2.1	Actel A54SX32	7
2.2	Timing Characteristics	8
2.3	Verilog Code	10
2.4	Simulation	20
2.4.1	Pin Placement	20
2.4.2	Simulation Results	22
3	Board Design	24
3.1	Vanderbilt Dewar Constraints	24
3.2	Daughterboard	25
3.2.1	ADC	25
3.2.2	Clock Signals	31
3.2.3	LVDS Transmitter	33
3.2.4	Passive Components	35
3.3	Motherboard	36
3.3.1	Regulator Circuits	38
3.3.2	Channel Inputs	40
3.3.3	Line Drivers	40
3.3.4	Data Collection	41
3.4	EAGLE Design	41
3.4.1	Daughterboard	42
3.4.2	Motherboard	46

4	Testing and Results	49
4.1	Test Setup	49
4.2	ADC Output Results	51
5	Conclusions	56
5.1	Summary	56
5.2	Original Contributions	57
	Bibliography	59
A	FPGA Control Code	62
B	FPGA Testbench Code	70
	Vita	73

List of Tables

2.1	State Names and Values	10
2.2	FPGA Timing Delays	11
3.1	RSI Pads Bonded Out(1)	27
3.2	RSI Pads Bonded Out (2)	28
3.3	RSI Pads Bonded Out (3)	29
3.4	LVDS Transmitter Connections	35
3.5	References	39

List of Figures

1.1	ADC Block Diagram	6
2.1	ADC Timing Diagrams	9
2.2	FPGA I/O Pin Placement	21
2.3	Simulation Results of FPGA Cycling from Start of Control Signals to End of Third Channel's Output	23
2.4	Simulation for Comparing with ADC Timing Diagram	23
3.1	Block Diagram of Daughterboard	26
3.2	RSI Chip Layout	26
3.3	RSI Bonding Diagram	32
3.4	PGA Package Pins Utilized	32
3.5	Female SMA Receptacle	33
3.6	Placement of 100 Ω Termination Resistor [11]	35
3.7	Bypass Capacitor Schematics	36
3.8	Block Diagram of Motherboard	37
3.9	Simplest LM317 Voltage Regulator Schematic	39
3.10	Regulator Circuit	39
3.11	Low-pass Filter	40
3.12	PGA Package Drawings	43
3.13	PGA Package Pins Color-coded by Function	43
3.14	PCB Layer Comparisons	44
3.15	Daughterboard Layout	44
3.16	Daughterboard Schematic	45
3.17	Motherboard Layout	47
3.18	Motherboard Schematic	48
4.1	Block Diagram of Test System	51
4.2	Charge Channel Data at -120°C	53
4.3	Charge Channel Data at -90°C	53

4.4	Charge Channel Data at -50°C	54
4.5	Charge Channel Data at $+25^{\circ}\text{C}$	54
4.6	Charge Channel Data at $+70^{\circ}\text{C}$	55
4.7	Charge Channel Data at $+125^{\circ}\text{C}$	55

Chapter 1

Introduction

Electronics used in space applications are frequently exposed to wide variations in temperature. The moon, for example, has temperature extremes ranging from $+123^{\circ}\text{C}$ in the daytime to -180°C at night. In addition to these wide temperature swings, the devices are subjected to a wide array of energetic particles that occur naturally in space, such as solar flares, charged particles trapped in Van Allen belts, heavy ions, and galactic solar flares. All of this is added on top of the total ionizing dose (TID) effects from electromagnetic radiation [9].

For this thesis, a test setup was designed to test a 12-bit, 16-channel analog-to-digital converter (ADC) inside a dewar designed by Vanderbilt University. The dewar allows the device under test to be tested at -180°C under radiation conditions at the same time. In the past, chips were tested under these conditions separately. Several factors that made the design challenging are discussed throughout the thesis.

1.1 Remote Sensor Interface (RSI)

The RSI ASIC is a mixed-signal ASIC that translates analog sensor data to digital form for back end processing. It is loosely based on the front end multi-chip-modules of the BAE Systems Remote Health Node (RHN) which was developed in 1997-1998 for the now-defunct NASA X-33 space plane program. The RSI includes 16 channels: 12 low-speed “universal” channels, 2 high-speed channels, and 2 charge amplifier channels.

The RSI is designed for operation across a wide temperature range, -180°C to $+125^{\circ}\text{C}$. Radiation hardened by design circuitry is incorporated within the design where possible and appropriate, particularly in the digital circuits.

Incorporated within the low-speed and high-speed channels are programmable Wheatstone bridges. Each channel of this type has a 5-pin interface, whereas the charge amplifier channels only use a 2-pin interface. The Wheatstone bridges are programmed through the use of on-chip shift registers.

Clocking for the RSI includes a very high-speed clock associated with the counter for the ADC. To support the 4x oversampling of the maximum 5 kHz sensors, an 85.12 MHz clock is required for ideal operation [2].

1.2 Analog-to-Digital Converters

An analog-to-digital converter is a circuit that changes analog signals to digital representations of that signal. Since we live in an analog world, ADCs are crucial in translating this analog data in order for products to perform calculations in the digital or discrete time domain [3].

1.2.1 ADC Parameters

One of the most important characteristics of an ADC is its resolution – the number of discrete values the ADC can supply over its analog input voltage range. Because these values are stored in a binary form, the resolution of an ADC is most often represented in output bits. An ADC with a resolution of n bits will have 2^n quantization levels [10]. For instance, if an ADC is said to have 3-bit resolution, it will have 2^3 , or 8, quantization levels over the full input voltage range. If the full-scale measurement range is from 0 to 5 Volts, the ADC must be able to take any voltage between this range and place it in 1 of the 8 quantization levels. The size of the quantization levels is also referred to as the step width. The ideal step width is defined in Equation 1.1 [3].

$$V_{idealstepwidth} = \frac{1}{2^n} \times V_{REF} = 1 \text{ LSB} \quad (1.1)$$

Using Equation 1.1 with the 3-bit, 5 V example ADC results in an ideal step width of 0.625 V.

Differential Nonlinearity (DNL) is defined as the deviation in step width from the value of 1 LSB [10]. The (DNL) of the ADC is derived plugging Equation 1.1 into Equation 1.2 below.

$$\text{DNL} = \text{Actual step width} - \text{Ideal step width} \quad (1.2)$$

Integral nonlinearity (INL) is the deviation of the ADC transfer function from the ideal straight line. When spanning a converter over its full-scale range, its INL determines the deviation of its transfer characteristic from the expected ideal straight line. Best-straight-line INL is the deviation of any code from a straight line calculated to minimize the worst-case INL errors [10].

Offset error comes about when there is a difference between the value of the first step transition and the ideal value of $\pm 1/2$ LSBs. Gain error, however, is the difference in the slope of a straight

line drawn through the transfer characteristic and the slope of 1 of an ideal ADC [3].

Signal-to-noise ratio (SNR) is a measure of the broadband and spurious noise that is introduced into the signal by the sampling and analog-to-digital conversion processes. It is given by the ratio expressed in dB of the rms amplitude of the ADC output fundamental tone to the rms amplitude of the output noise, where noise is defined as the sum of all frequencies in the Nyquist band excluding DC, fundamental, harmonics, and spurious components [10].

1.2.2 Wilkinson ADC

The ADC contained within the RSI is a 12-bit, 16-channel Wilkinson ADC. Wilkinson ADCs function by comparing the input voltage with the voltage produced by a capacitor that charges until its voltage is equal to the amplitude of the input signal, which is determined by a comparator. At the point where the capacitor begins linearly discharging and producing a ramp voltage, a gate pulse is initiated and remains on until discharging is complete. The time that this gate pulse is on is proportional to the amplitude of the input pulse. There is also a clock counting the pulse length and storing that value in a register, which is the ADC output.

The ADC also includes radiation aware power management (RAPM) circuitry that generates 2.5 V internally from the 3.3 V digital supply. This 2.5 V reference powers the digital back-end of the ADC. The RAPM requires multi-phase clocking that is generated internally from a 166.25 kHz source clock [2].

The ADC portion of the RSI chip has several input/output ports as seen in Figure 1.1.

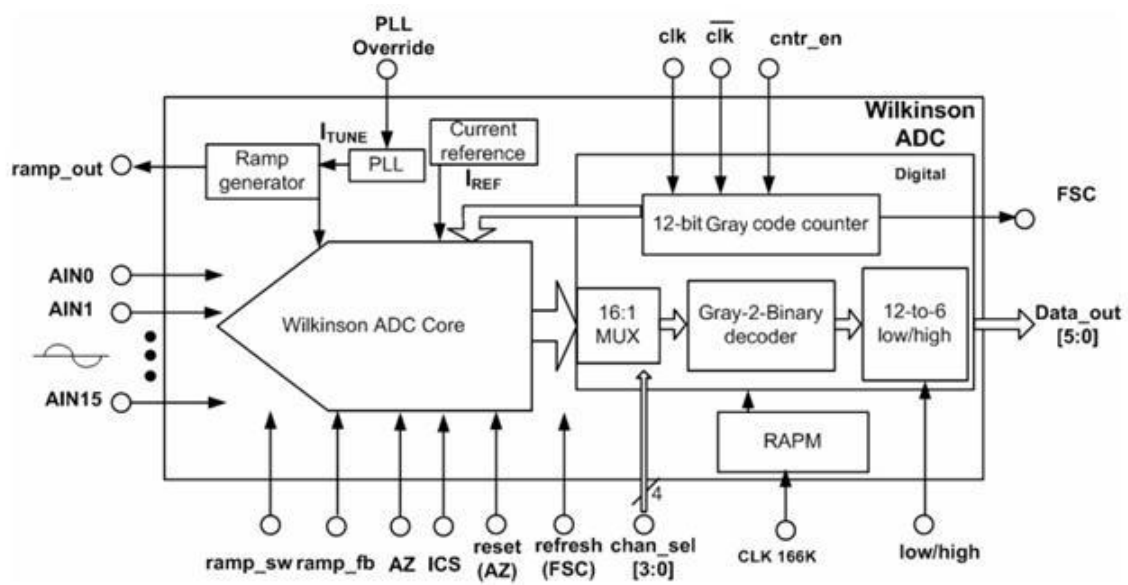


Figure 1.1: ADC Block Diagram

Chapter 2

FPGA

In order to test the ADC, a field-programmable gate array (FPGA) was utilized. FPGAs are integrated circuits that are designed to be programmed by a customer using a hardware description language (HDL). FPGAs contain programmable logic components, commonly referred to as logic blocks, and a hierarchy of reconfigurable interconnects that allow the blocks to be wired together. Verilog, one of many HDLs, was used to provide all of the necessary control signals to test the ADC.

2.1 Actel A54SX32

To produce the most accurate results, the FPGA needed to be as close to the ADC testchip as possible. This meant that the FPGA and RSI had to be placed on the same board. One ramification of this is that FPGAs that can function properly at -180°C are very hard to find and extremely expensive if found. Previous research by the NASA Jet Propulsion Laboratory (JPL) proved that the Actel A54SX32 FPGA works in such cold temperatures. Upon researching this Actel FPGA, it was learned that although it is an FPGA, it is not reprogrammable – meaning that if the chip is

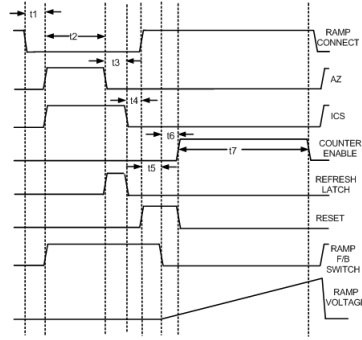
not programmed correctly on the first try, there is no way to reprogram it in order to reuse it. In addition, the only way to test the programmed chip is by soldering it to the testboard and verifying system functionality.

The Actel SX architecture features two types of logic modules that are optimized for fast and efficient mapping of synthesized logic functions – the combinatorial cell (C-cell) and the register cell (R-cell). The routing and interconnect resources are in the metal layers above the logic modules allowing the entire floor of the device to be spanned with an uninterrupted grid of fine-grained, synthesis-friendly logic modules, which reduces the distance signals have to travel between logic modules. The SX family also contains a hardwired, constantly loaded clock network that has been tuned to provide fast clock propagation with minimal clock skew [1].

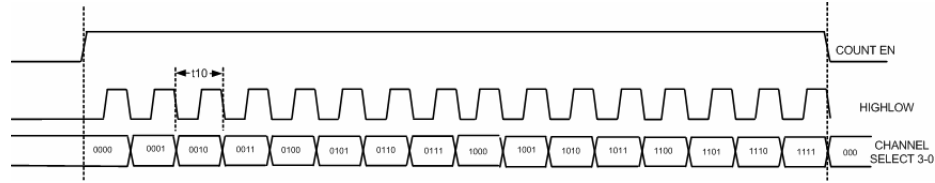
FPGAs in the Actel SX family are implemented on a high-voltage twin-well CMOS process using $0.35\ \mu\text{m}$ design rules. The metal-to-metal antifuse is made up of a combination of amorphous silicon and dielectric material with barrier metals and has a programmed (“ON” state) resistance of $25\ \Omega$ with a capacitance of $1.0\ \text{fF}$ for low signal impedance. Each I/O on the SX device can be configured as an input, an output, a tristate output, or a bidirectional pin. The SX family supports $3.3\ \text{V}$ operation and is designed to tolerate $5\ \text{V}$ inputs [1].

2.2 Timing Characteristics

Figures 2.1(a) and 2.1(b) show the timing characteristics of the ADC that the FPGA will need to provide. The conversion of the analog signal is mainly divided into two timing phases – the sample and autozero phase and the ramp, or conversion, phase. During the first phase, the input signal is sampled by the comparator. At the same time, the ICS and AZ signals control the removal of the output offset by autozeroing. The ramp generator is reset during this phase, allowing the integrating



(a) Main ADC Control Signal Timing Diagram



(b) Timing Diagram for Reading Data

Figure 2.1: ADC Timing Diagrams

capacitor to completely discharge. During the conversion phase, the autozero and input sampling switch is opened. The ramp generator output is then connected to the input of the comparator using the RAMP CONNECT signal. The ramp feedback (RAMP F/B) switch is opened to allow the integration capacitor to begin charging, and the COUNTER ENABLE signal is pulled high to start the counter. At the end of this phase, the ramp output voltage reaches 1.2 V and the counter reaches a count of 4,096, or 2^{12} . This marks the end of the conversion cycle. The individual channel comparators trip during the conversion cycle depending on their respective analog inputs, and the corresponding counter output state is stored in its respective channel's latch. During the next autozero cycle, each channel latch transfers data to a second set of latches using the REFRESH LATCH signal, after which the first set of latches are reset using the RESET signal in order to prepare for the next conversion. The data from the second set of latches is read out during the next conversion using the CHANNEL SELECT signals and HIGHLOW signal that alternates the

output from the 6 least significant bits (LSBs) to the 6 most significant bits (MSBs) [6].

In order for the FPGA to interface with the ADC properly, it had to be able to take a small set of inputs as well, the first of which is a clock. The clock signal oscillates between a high and a low state in order to coordinate when actions are to occur. The FPGA must also be able to accurately read the outputs from the 6 ADC pins and concatenate them properly in order to represent the full 12-bit output.

2.3 Verilog Code

The purpose of this section is to walk the reader through the Verilog code sent to the FPGA in order to provide a better understanding of its internal workings.

Table 2.1 shows the names given to each state of the state machine and their assigned binary values. Each of these states is defined in the same manner that state “ST_6” is defined below.

Table 2.1: State Names and Values

State Name	Binary Value
ST_0	0000
ST_1	0001
ST_2	0010
ST_3	0011
ST_4	0100
ST_5	0101
ST_6	0110
ST_7	0111
ST_CH_SEL_00	1000
ST_CH_SEL_01	1001
ST_CH_SEL_10	1010
ST_CH_SEL_11	1011
ST_CH_SEL_20	1100
ST_CH_SEL_21	1101
ST_CH_SEL_END	1110
ST_NEXT_CYCLE	1111

```
'define ST_6 6'b000110
```

This statement is in place for both debugging purposes as well as to let the FPGA know when to change to the next state.

The next step in the initial setup was defining the time delays that the FPGA had to replicate as shown in Figure 2.1(a). Table 2.2 gives the approximate time delays when an 83.33 MHz clock is used.

The values for these time delays were calculated using Equation 2.1 below.

$$(\text{End of State} - \text{Start of State}) \times \frac{1}{\text{Clock Frequency}} = \text{Delay (s)} \quad (2.1)$$

All time delays are defined in the same manner as the COUNT_RMP_ICS_CLOSE and COUNT_RMP_START are defined below.

```
'define COUNT_RMP_ICS_CLOSE 16'd145
```

```
'define COUNT_RMP_START      16'd153
```

Equation 2.2 uses these 2 signals as an example on how to calculate the difference in the two times giving a 96 ns time delay, which corresponds with time t6 shown in the ADC timing diagram in

Table 2.2: FPGA Timing Delays

Parameter	Time
t1	48 ns
t2	1.5 μ s
t3	96 ns
t4	48 ns
t5	48 ns
t6	96 ns
t7	12.12 μ s

Figure 2.1(a).

$$(153 - 145) \times \frac{1}{83.33 \text{ MHz}} = 96 \text{ ns} \quad (2.2)$$

The next few lines in the Verilog code define the signals that are the actual output from the FPGA itself. Every output signal is defined here in a manner similar to that seen in the example using the Reset signal seen below.

```
output   Reset;
```

The reg definitions are next in the code. The “reg” command stores both value and drive strength and is beneficial when a case statement is required. All internal signals and some external to the FPGA are defined like the ADC_out signal below.

```
reg [11:0] ADC_out;
```

The inputs are defined next using the input statement. The command for defining the Clock signal as an input is shown below.

```
input   Clock;
```

The lines below tell the FPGA to map the input ports Chip_enable and SOC to reg definitions Enable and Start, respectively.

```
always @(posedge Clock)
begin
    #CQ Enable = Chip_enable;
end
always @(posedge Clock)
begin
    #CQ Start = SOC;
end
```

The next 7 lines tell the FPGA to begin churning through the state machine as long as both signals Enable and Start are high.

```

always @(posedge Clock)
begin
    if(Enable == 1)
        begin
            if(Start == 1)
                begin
                    case (State)

```

In the code for state ST_1 several values are initialized to zero to replicate time t1 in the ADC timing diagram. Once the state counter counts through 4 clock cycles, it moves on to state ST_2.

```

‘ST_1:
begin
    #‘CQ S1_rmp_ctrl          = 1'b0;
    #‘CQ S2_comp_rmp         = 1'b0;
    #‘CQ S3_comp_inp         = 1'b0;
    #‘CQ S4_comp_fb          = 1'b0;
    #‘CQ Count_enable        = 1'b0;
    #‘CQ Refresh_latch       = 1'b0;
    #‘CQ High_low            = 1'b0;
    #‘CQ Ch_select           = 4'b0000;
    #‘CQ Reset               = 1'b0;
    #‘CQ Start_cnt_counter   = 1'b0;
    if (Count_4095 == ‘COUNT_RMP_INP_SWO)
        begin
            #‘CQ State = ‘ST_2;
        end
    else
        begin
            #‘CQ State = ‘ST_1;
        end
end

```

Once in state ST_2, time t2 in the ADC timing diagram, the AZ, ICS, and RAMP F/B SWITCH signals are set high. After 125 clock cycles, state ST_3 begins. The ADC datasheet contains limits for each timing parameter. For clarification, these limits are shown in the states' comments.

```

‘ST_2:
begin
    #‘CQ S1_rmp_ctrl = 1'b1;
    #‘CQ S3_comp_inp = 1'b1;

```

```

#CQ S4_comp_fb = 1'b1;
if (Count_4095 == 'COUNT_AZ_INP_SWCC) //1.5 us MAXIMUM
  begin
    #CQ State = 'ST_3;
  end
else
  begin
    #CQ State = 'ST_2;
  end
end
end

```

State ST_3 corresponds to time t3 in the ADC timing diagram in which the autozero signal, AZ, is switched to low and REFRESH LATCH is switched to high. This state lasts for the previously defined 8 clock cycles until moving to the next state.

```

'ST_3:
begin
  #CQ S4_comp_fb = 1'b0;
  #CQ Refresh_latch = 1'b1;
  if (Count_4095 == 'COUNT_AZ_INP_SWCO) //94 ns MINIMUM
    begin
      #CQ State = 'ST_4;
    end
  else
    begin
      #CQ State = 'ST_3;
    end
  end
end
end

```

The only signal that changes state during time t4, or state ST_4, is the ICS signal. This signal is switched low and the 4 clock cycle period begins until moving to the next state.

```

'ST_4:
begin
  #CQ S3_comp_inp = 1'b0;
  if (Count_4095 == 'COUNT_INP_COMP_OPEN) //47 ns MINIMUM
    begin
      #CQ State = 'ST_5;
    end
  else
    begin

```

```

        #CQ State = 'ST_4;
    end
end

```

In time t5, state ST_5 in the Verilog code, the REFRESH LATCH signal is switched low and the RAMP CONNECT signal is finally switched high.

```

'ST_5:
begin
    #CQ S2_comp_rmp    = 1'b1;
    #CQ Refresh_latch = 1'b0;
    if (Count_4095 == 'COUNT_RMP_ICS_CLOSE) //47ns MINIMUM
        begin
            #CQ State = 'ST_6;
        end
    else
        begin
            #CQ State = 'ST_5;
        end
    end
end

```

After 4 clock cycles, state ST_6 begins. As seen in time t6 in the ADC timing diagram, the RESET signal is switched high, while the RAMP F/B SWITCH signal is set to a low state. This state is also where the ramp voltage is initialized and begins its ramp up to the reference voltage. Because of this, an internal counter that counts 4,096 clock ticks is started after 8 clock cycles and the next state is initiated.

```

'ST_6:
begin
    #CQ Reset          = 1'b1;
    #CQ S1_rmp_ctrl    = 1'b0;
    if (Count_4095 == 'COUNT_RMP_START) //94ns MINIMUM
        begin
            #CQ State = 'ST_7;
            #CQ Start_cnt_counter = 1'b1;
        end
    else
        begin
            #CQ State = 'ST_6;
        end
    end
end

```

```

    end
end

```

In time t7 on the ADC timing diagram, state ST_7 in the code, the COUNTER ENABLE signal is set to high and the RESET signal is switched low. After waiting for a while, the FPGA begins cycling through the ADC's channels in order to output the data.

```

'ST_7:
begin
    #CQ Count_enable = 1'b1;
    #CQ Reset        = 1'b0;
    if (Count_4095 == 'COUNT_CH_SEL_START) //48.12us MAXIMUM
        begin
            #CQ State = 'ST_CH_SEL_00;
        end
    else
        begin
            #CQ State = 'ST_7;
        end
    end
end

```

The next few parts of the code that occur are all within time t7 on the timing diagram. This is when the ADC cycles through each channel and outputs its 12-bit data 6 bits at a time. In the first section, state ST_CH_SEL_X0, signal HIGHLOW is set low and the 4 CHANNEL SELECT bits are configured to select the proper channel. The ADC outputs the lowest 6 bits and they are stored within the FPGA. After 84 clock cycles, the state is changed.

```

'ST_CH_SEL_20:
begin
    #CQ High_low      = 1'b0;
    #CQ Ch_select     = 4'b1110;
    #CQ ADC_store_2[5:0] = ADC_in[5:0];
    if (Count_4095 == 'COUNT_CH_SEL_20) //1us
        begin
            #CQ State = 'ST_CH_SEL_21;
        end
    else
        begin

```

```

        #CQ State = 'ST_CH_SEL_20;
    end
end

```

In state ST_CH_SEL_X1, signal HIGHLOW is set high and the ADC outputs the highest 6 bits from the same channel. These 6 bits are then stored and concatenated with the lowest 6 bits in order to give a 12-bit output. After another 84 clock cycles, signal Chgdet is then inverted to let the data gathering software know it is time to read the FPGA's output. The process is repeated until the final channel has been converted.

```

'ST_CH_SEL_21:
begin
    #CQ High_low = 1'b1;
    #CQ ADC_store_2[11:6] = ADC_in[5:0];
    #CQ ADC_out = ADC_store_2;
    if (Count_4095 == 'COUNT_CH_SEL_21) //1us
        begin
            #CQ State = 'ST_CH_SEL_END;
            #CQ Chgdet = ~Chgdet;
            #CQ High_low = 1'b0;
        end
    else
        begin
            #CQ State = 'ST_CH_SEL_21;
        end
    end
end

```

State ST_CH_SEL_END then resets the Start_cnt.counter, RAMP CONNECT, and COUNT ENABLE signals to be low.

```

'ST_CH_SEL_END:
begin
    if (Count_cntr4095 >= 12'd4095)
        begin
            #CQ Start_cnt_counter = 1'b0;
            #CQ State = 'ST_NEXT_CYCLE;
            #CQ S2_comp_rmp = 1'b0;
            #CQ Count_enable = 1'b0;
        end
    end

```

```

end

'ST_NEXT_CYCLE:
begin
  if (Count_4095 == 'COUNT_CH_SEL_END)
    begin
      #CQ State = 'ST_0;
    end
  else
    begin
      #CQ State = 'ST_NEXT_CYCLE;
    end
  end
end
'ST_0:
begin
  #CQ Ch_select = 4'b0000;
  #CQ State = 'ST_1;
end

```

One other critical part of the FPGA code is the RAPM clock generation – based on a simple clock divider. If the clock used is 83.33 MHz and the RAPM clock needs to be 166.66 kHz, then for every 500 cycles of the main clock, the RAPM clock would have one cycle. This is accomplished by using a simple counter and inverting the Clk_166k signal every 250 clock cycles as seen below.

```

always @(posedge Clock)
begin
  if (Enable == 1'b1)
    begin
      if (Count_rapm == 10'h0FA)
        begin
          Clk_166k = ~ Clk_166k;
          Count_rapm = 10'h000;
        end
      else
        begin
          Count_rapm = Count_rapm + 1;
        end
      end
    end
  else
    begin
      Clk_166k = 1'b0;
      Count_rapm = 10'h000;
    end
  end
end

```

```

    end
end

```

In order to simulate the code, a testbench is written that acts as the ADC's outputs (the FPGA's inputs). The plotted outputs are mapped and defined using the proper statements. The statement shown below is used for clock generation. This provides a period of 12 ns, a frequency of 83.33 MHz, with a duty cycle of 0.5 and offset of 0. When the “#” command is used, it means to wait the time period set by the timescale. For instance, in the case below, #7, means to wait 7 ns.

```

`timescale 1ns/1ps
parameter PERIOD = 12;
parameter real DUTY_CYCLE = 0.5;
parameter OFFSET = 0;

initial    // Clock process for Clock
begin
    #OFFSET;
    forever
    begin
        Clock = 1'b0;
        #(PERIOD-(PERIOD*DUTY_CYCLE)) Clock = 1'b1;
        #(PERIOD*DUTY_CYCLE);
    end
end
end

```

After defining the clock, the testbench begins to simulate the 6 bits that are output by the ADC. The FPGA is first initialized by setting the Chip_enable and SOC signals, and then several ADC inputs are sent to it in order to make sure it is concatenating the highest and lowest 6 bits properly to form the 12-bit output. A sample is shown below.

```

#100;
    Chip_enable = 1'b1;
    SOC = 1'b1;
    ADC_in = 6'b101010;
#15400

```

```

    ADC_in = 6'b010101;
#1113
    ADC_in = 6'b011001;
#1008
    ADC_in = 6'b110010;
#1113
    ADC_in = 6'b101011;
#1008
    ADC_in = 6'b000011;
#1113
    ADC_in = 6'b000000;

```

2.4 Simulation

The code is then uploaded as a project into Libero Integrated Design Environment (IDE), Actel's comprehensive software toolset for its FPGAs. Libero manages the entire design flow from synthesis and simulation to pin placement all the way to program file generation.

2.4.1 Pin Placement

The design utilizes 320 of a total 2880 cells inside the chip, only 12%, with a total of 35 I/O cells used. Since the Libero software allows users to select pin locations, it is important to consider the placement of the FPGA on the PCB with respect to ease of routing its I/O signal paths. All of the ADC output bits are grouped together to minimize difference in delay of the signals while the rest of the signals are specifically ordered on the corner of the chip so that they will line up properly with the RSI chip. The 3 reference voltages and the special pins used for boundary-scan testing are locked in place per Actel's requirements and thus, cannot be moved to a different location. Figure 2.2 shows the final FPGA pin placement.

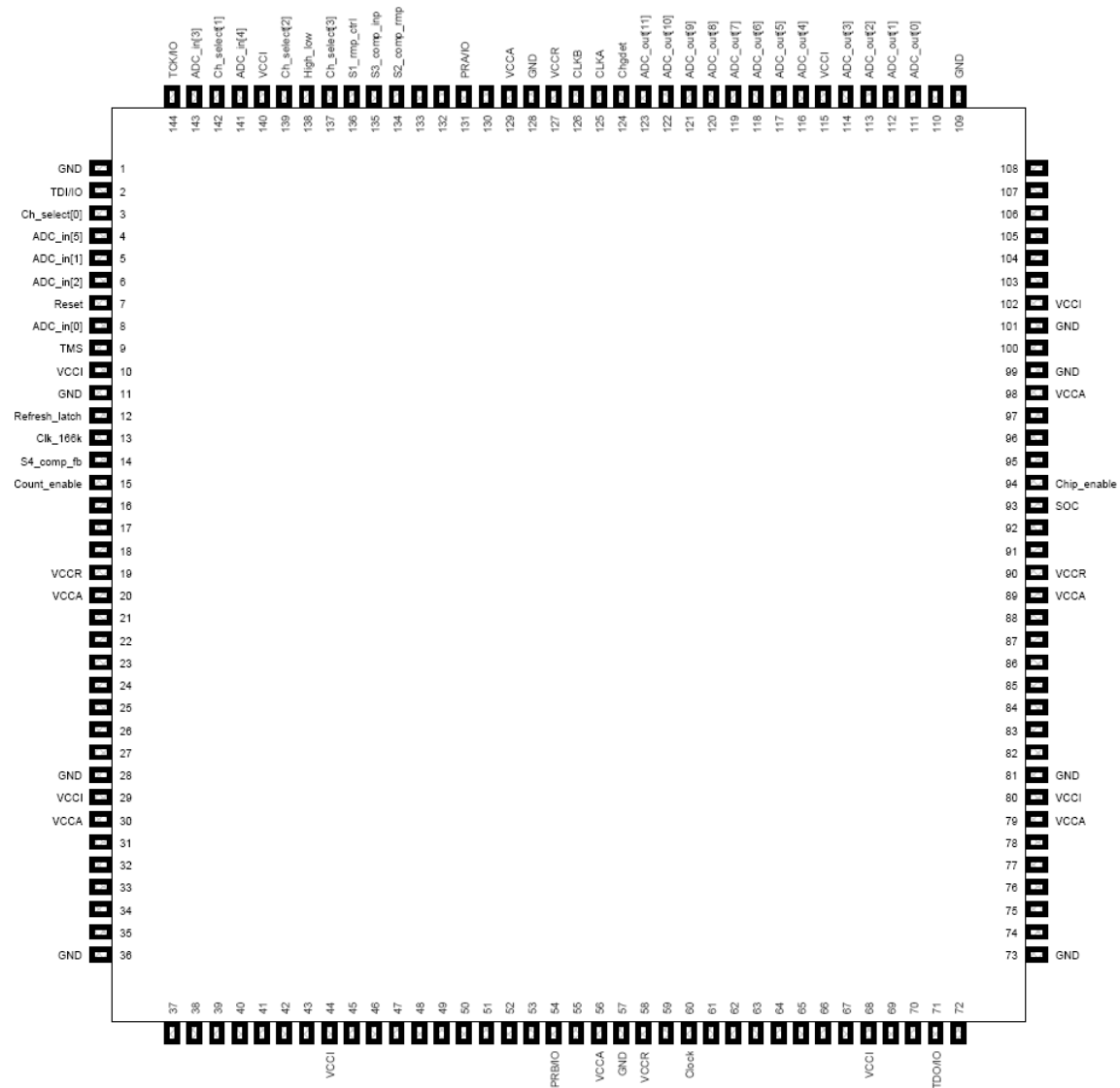


Figure 2.2: FPGA I/O Pin Placement

2.4.2 Simulation Results

Post-layout simulation is done after settling on the pinout for the chip. The main point of this simulation is to take into account the timing delays associated with placing the pins in their respective locations. Figure 2.3 shows the FPGA running through an entire cycle from the beginning of the ADC control signals to the end of the third channel's output. Figure 2.4 shows the waveform that can be compared with the ADC timing diagram to ensure the accuracy of the relative signal timing.

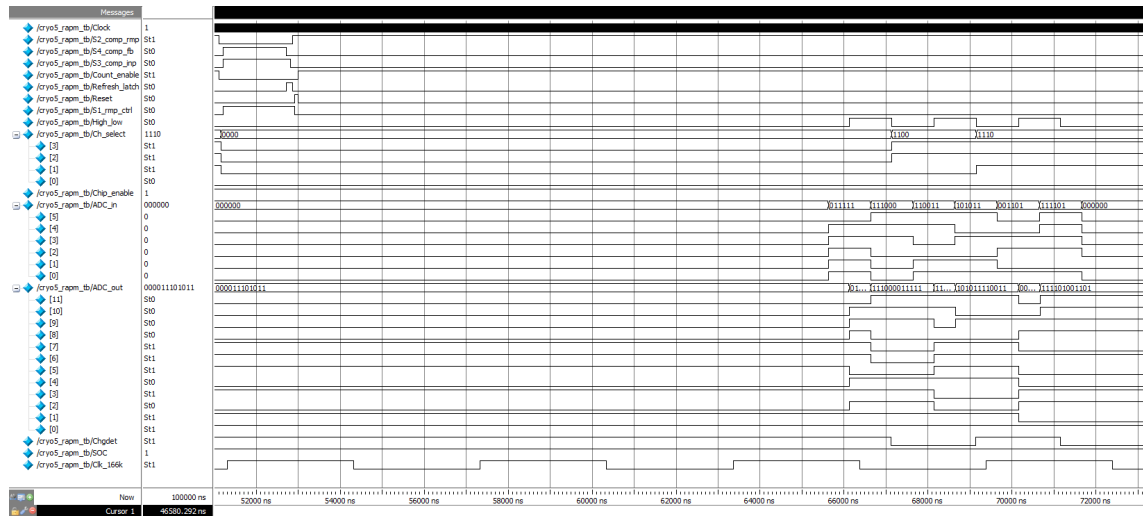


Figure 2.3: Simulation Results of FPGA Cycling from Start of Control Signals to End of Third Channel's Output

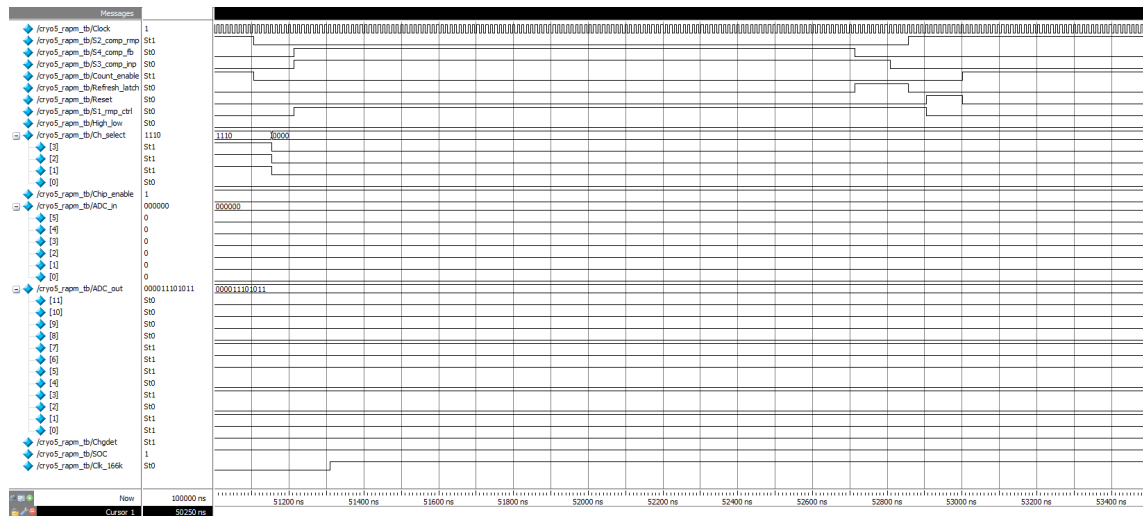


Figure 2.4: Simulation for Comparing with ADC Timing Diagram

Chapter 3

Board Design

3.1 Vanderbilt Dewar Constraints

While the Vanderbilt dewar is an amazing piece of equipment, it does not allow much flexibility in terms of designing equipment to be placed inside for testing. The following constraints are considered during all aspects of this ADC testboard design.

- The testboard placed inside the dewar must be no more than 4.7" by 4.5".
- The testchip must be placed in an exact location close to the middle of the board in order to maximize beam exposure.
- Only radiation tolerant devices can be placed within a 2 inch diameter of the center of the testchip.
- A relatively large hole must be drilled in the testboard in order to place a cold finger on the back of the testchip.
- The I/O interface that is used can only support 25-pin MDM cables and SMA cables.

3.2 Daughterboard

In order to test the system, all of the critical components, basically everything but the supply voltage regulators and a few minor parts, are placed on the same board that is designed to go into the Vanderbilt dewar. This includes the RSI chip, the FPGA, clocking components, a Low-voltage differential signaling (LVDS) transmitter for clock I/O, and several passive devices. Figure 3.1 is a block diagram of the daughterboard. Each individual block is discussed in more detail later in this chapter.

3.2.1 ADC

As stated before, the RSI contains much more than just the ADC. Because of this, there are nearly 250 pads on the entire chip shown in Figure 3.2. Since the objective is to only test the ADC, only pads related to its operation are bonded out during chip packaging. Tables 3.1 through 3.3 show all of the pads that are bonded out to ensure ADC functionality as well as a brief description of each signal's function.

Several of the signals are those that have been previously discussed that interface with the FPGA. Many others are for supply voltages and electrostatic discharge (ESD) protection of the channels and the ADC.

There still are a few signals though whose functionality has not yet been clarified. The 2.5 V output from the RAPM regulator is PAD_REGR_OUT_ADC. It is used to identify when rail collapse has occurred due to onset of latch-up from ion beam irradiation, as the regulator output will drop to 0 V for a short period and restart itself. An external 2 nF capacitor is needed from this pin to ground. PAD_MPC_SCN_OUT_GLBL is the manufacturing test data output of the scan string for checking the flip-flops that comprise the multi-phase clock generation circuitry.

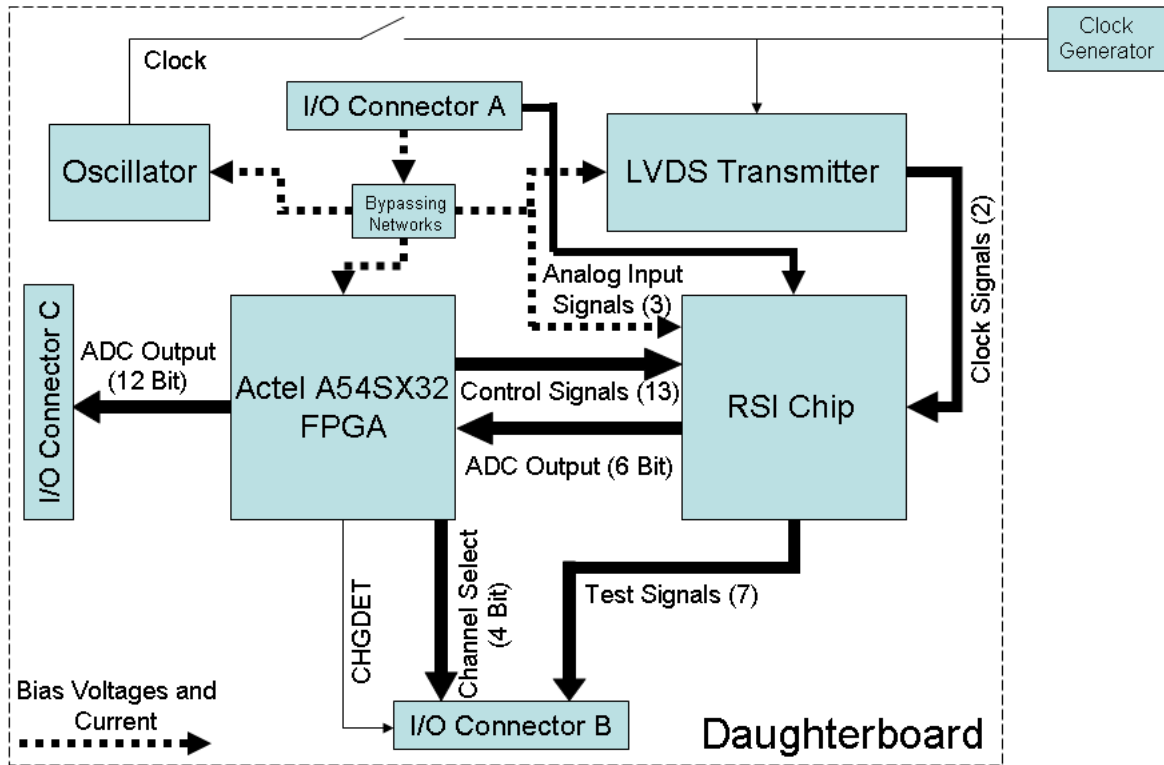


Figure 3.1: Block Diagram of Daughterboard

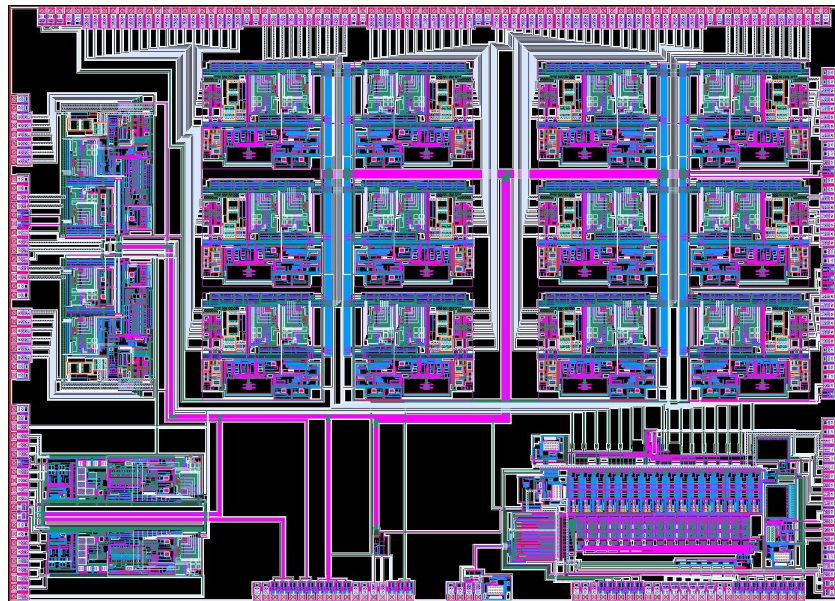


Figure 3.2: RSI Chip Layout

Table 3.1: RSI Pads Bonded Out(1)

Pad	Signal Name	Signal Description
1	PAD_ESD_GND7_LSC	ESD ground
2	PAD_ESD_VCC7_LSC	ESD supply
12	PAD_ESD_GND7_LSC	ESD ground
13	PAD_ESD_VCC7_LSC	ESD supply
14	PAD_6PHASE_DVSS_3P3V	6 phase clock ground
15	PAD_6PHASE_DVDD_3P3V	6 phase clock supply
19	PAD_ESD_GND3_ADC	ESD ground
20	PAD_ESD_VCC3_ADC	ESD supply
21	PAD_CLK_LVDS_ADC	Differential high-speed clock for the ADC counter
22	PAD_CLKBAR_LVDS_ADC	Differential high-speed clock for the ADC counter
23	PAD_ESD_GND2_ADC	ESD ground
24	PAD_ESD_VCC2_ADC	ESD supply
25	PAD_REGR_OUT_ADC	RAPM 2.5 V output to external 2 nf capacitor and monitor for regulator shut down
26	PAD_AZ_ADC	Triggers auto-zero of the comparator
27	PAD_COUNT_EN_ADC	Starts the counter following auto-zero
28	PAD_CLK_IN_166K_ADC	Source of the three-phase RAPM clock
29	PAD_REFRESH_ADC	Updates latch data to register during auto-zero
30	PAD_RESET_ADC	Clears all latch contents
31	PAD_ESD_GND2_ADC	ESD ground
32	PAD_ESD_VCC2_ADC	ESD supply
33	PAD_DATAOUT_0_ADC	Digital data output bit 0 Least Significant Bit (LSB) from ADC
34	PAD_DATAOUT_1_ADC	Digital data output bit 1 from ADC
35	PAD_DATAOUT_2_ADC	Digital data output bit 2 from ADC
36	PAD_DATAOUT_3_ADC	Digital data output bit 3 from ADC
37	PAD_DATAOUT_4_ADC	Digital data output bit 4 from ADC
38	PAD_DATAOUT_5_ADC	Digital data output bit 5 Most Significant Bit (MSB) from ADC
39	PAD_ESD_GND2_ADC	ESD ground
40	PAD_ESD_VCC2_ADC	ESD supply
41	PAD_HIGH_LOW_ADC	Selects between MSB (11:6) and LSB (5:0) of the ADC digital output
42	PAD_CHAN_SEL_0_ADC	Select Least Significant Bit (LSB) 0 for digital 12-bit channel output from the ADC

Table 3.2: RSI Pads Bonded Out (2)

Pad	Signal Name	Signal Description
43	PAD.CHAN_SEL_1_ADC	Select Bit 1 for 12-bit digital channel output from the ADC
44	PAD.CHAN_SEL_2_ADC	Select Bit 2 for 12-bit digital channel output from the ADC
45	PAD.CHAN_SEL_3_ADC	Select Most Significant Bit (MSB) 0 for 12-bit digital channel output from the ADC
46	PAD.RAMP_SW_ADC	Gates the ramp generator into the comparator for each channel
47	PAD.ICS_ADC	Gates the channel inputs to the comparator for each channel
48	PAD.RAMP_FB_ADC	Control signal to charge vs. discharge the ramp generator capacitor
49	PAD.MPC_SCN_OUT_GLBL	Manufacturing test signal
50	PAD.FSC_ADC	Full Scale Count test pin goes high when counter reaches maximum count value
51	PAD.DVSS_3P3V_ADC	Digital ADC ground
52	PAD.DVDD_3P3V_ADC	Digital ADC supply
53	PAD.ESD_GND1_ADC	ESD ground
54	PAD.ESD_VCC1_ADC	ESD supply
55	PAD.PLL_CCOMP_ADC	Output to external 1 nF capacitor from the PLL temperature compensation circuit
56	PAD.DVDD_PLL_3P3V_ADC	PLL supply
57	PAD.DVSS_PLL_3P3V_ADC	PLL ground
58	PAD.PLL_OVERRIDE_ADC	Bypass for the temperature compensation PLL circuit
59	PAD.AVDD_RAMP_3P3V_ADC	Ramp supply
60	PAD.AVSS_RAMP_3P3V_ADC	Ramp ground
61	PAD.RAMP_OUT_ADC	Output of ramp generator
62	PAD.COMP_5U_IOUT_ADC	Current source monitor output
63	PAD.AVDD_ANA_3P3V_ADC	Analog ADC supply
64	PAD.AVDD_ANA_3P3V_ADC	Analog ADC supply
65	PAD.AVSS_ANA_3P3V_ADC	Analog ADC ground
66	PAD.AVSS_ANA_3P3V_ADC	Analog ADC ground
67	PAD.ESD_GND1_ADC	ESD ground
68	PAD.ESD_VCC1_ADC	ESD supply
69	PAD.MPC_SCN_IN_GLBL	Manufacturing test signal
70	PAD.ANLG_OUT_LSC_CH1	Analog output signal from low speed channel 1: test point prior to ADC input

Table 3.3: RSI Pads Bonded Out (3)

Pad	Signal Name	Signal Description
71	PAD_ESD_GND1_LSC	ESD ground
72	PAD_ESD_VCC1_LSC	ESD supply
94	PAD_SHFT_REG_TM_GLBL	Manufacturing test signal
95	PAD_PBUS_DVDD_3P3V	Digital supply
96	PAD_ESD_GND3_LSC	ESD ground
97	PAD_ESD_VCC3_LSC	ESD supply
203	PAD_ESD_GND1_HSC	ESD ground
204	PAD_ESD_VCC1_HSC	ESD supply
209	PAD_ANLG_OUT_HSC_CH1	Analog output signal from high-speed channel 1: test point prior to ADC input
215	PAD_ESD_GND1_HSC	ESD ground
216	PAD_ESD_VCC1_HSC	ESD supply
225	PAD_ESD_GND_CHC	ESD ground
226	PAD_ESD_VCC_CHC	ESD supply
227	PAD_ANLG_OUT_CHC_CH1	Analog output signal from charge amp channel 1: test point prior to ADC input
238	PAD_ESD_GND_CHC	ESD ground
239	PAD_ESD_VCC_CHC	ESD supply

In normal operation, this signal was not utilized and is left unconnected. PAD_FSC_ADC is another test pin; it is set to high when the 12-bit Gray code counter reaches its maximum value. PAD_PLL_CCOMP_ADC is an output signal that is used by the ADC temperature compensation circuit. An external 1 nF capacitor is tied from this pin to ground. PAD_PLL_OVERRIDE_ADC is a signal that allows the temperature compensating phase locked loop (PLL) to be bypassed and replaced with a constant current source reference circuit. For normal operation with the compensation activated, this input is tied to ground. To bypass the PLL, this pin is tied to logic high. PAD_RAMP_OUT_ADC is the output of the ramp generator within the ADC. It is bonded out for debugging purposes, but it is important to note that any external connections could have an effect on the ramp signal's characteristics, thus not giving the "true" signal that the ADC would otherwise see. PAD_COMP_5U_IOUT_ADC is another simple output test signal that monitors the internal current reference. PAD_MPC_SCN_IN_GLBL is the manufacturing test data input for

scanning through the flip-flops that comprise the multi-phase clock generation circuitry. Under normal operation, this signal is not used and tied to ground. PAD_SHFT_REG_TM_GLBL is also a manufacturing test signal that selects the scan input for each shift register in place of the parallel data. The test mode is entered when the signal is held at logic high. When held low, normal operation occurs with registers accepting data from their functional data inputs. It is important that this signal is tied to a known potential and not left floating [2].

This leaves the 3 analog “out” signals – one from each channel type: PAD_ANLG_OUT_LSC_CH1, PAD_ANLG_OUT_HSC_CH1, and PAD_ANLG_OUT_CHC_CH1. Each of these signals is the analog output of the channels prior to going into the ADC. In order to eliminate the loading effects of the channel, a focused ion beam (FIB) is used to virtually “cut” the metal that connects to the channel. This, in turn, makes the output act as an input allowing an external signal to be brought in and connected to its respective channel on the ADC.

When all of these signals are taken into consideration, there are a total of 74 pads to be bonded out to the package. There are several factors to keep in mind when selecting a package – I/O pins, cavity size (the area in which the chip is placed), ease of bonding, external package size, and package type. Cavity size is very important in this case because the RSI is 14 mm x 10 mm, large by most standards. An additional 1 mm needs to be added to each side to allow room for bonding. This means that a cavity size of 16 mm x 12 mm (0.63 in x 0.472 in) is necessary. Another constraint that complicates the selection of the ADC package with enough I/O and large enough cavity is the fact that the ADC is concentrated in the lower-right corner of the chip. Because of this, a majority of the pads bonded out are highly concentrated in this area. Generally, it is not a good practice to route highly inductive bond wires over long lengths across the chip; therefore, it is important to select a package with a large number of bonding pads on the lower and right

side. These constraints lead to the selection of a 145-pin pin grid array (PGA) package. This is a necessary, though non-ideal, choice because the internal wiring from the bond pads to the I/O pins is highly inductive and provides many opportunities to negatively affect signal integrity. Figures 3.3 and 3.4 below show the bonding diagram for the RSI in the selected package as well as a diagram showing which I/O pins of the package are used (shown in black). It is described later in this chapter how having all of these I/O pins heavily concentrated on 2 sides complicates printed circuit board (PCB) signal routing.

To swap out testchips efficiently, or just in case something goes awry, a socket that fit the package is needed. As previously stated, the dewar uses a cold finger to make thermal contact with the back of the package to bring the temperature down, meaning that the socket itself has to have a cutout that allows the cold finger to pass through. The chosen socket is manufactured with a 0.64 square inch square cutout.

3.2.2 Clock Signals

As previously stated, the clock signals also need to be brought directly to the motherboard in order to avoid issues that are associated with running high-speed digital signals over long distances. There are actually two methods that can be used to provide the clock signals.

The first method simply involves providing external clock signals via a clock generator and sending them to the board. A special type of cable has to be used in this method. Simple ribbon cable will not provide the clock signal fidelity needed. Coaxial radio frequency (RF) cables with SubMiniature version A (SMA) connectors are utilized. These connectors have a $50\ \Omega$ impedance and typically offer excellent electrical performance from DC to 18 GHz. Female receptacles such as the one shown below in Figure 3.5 are needed on the board to bring these signals in to the ADC

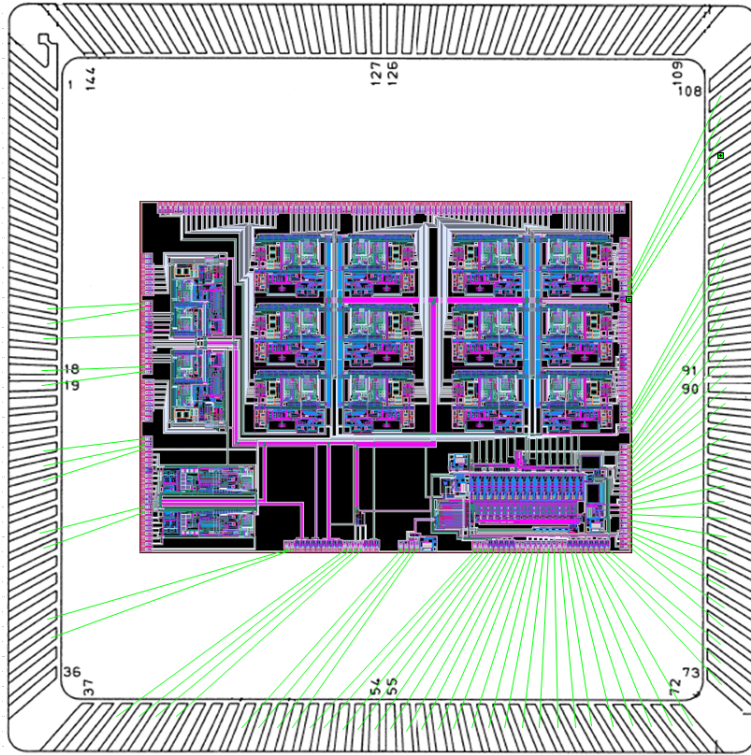


Figure 3.3: RSI Bonding Diagram

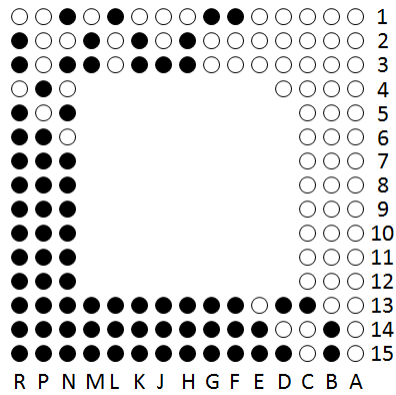


Figure 3.4: PGA Package Pins Utilized

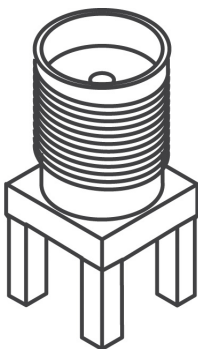


Figure 3.5: Female SMA Receptacle

test board.

At the time of design, it was unsure whether the dewar design would allow SMA connectors, so an alternative clocking mechanism was considered. The second method of providing the clock was to use an on-board oscillator. Using the piezoelectric effect, quartz crystals convert mechanical vibrations to electrical current at a particular frequency in order to serve as the primary element in oscillator circuits. In piezoelectric materials the application of mechanical pressure along an axis will result in the creation of an electrical charge along an axis at right angles to the first. In some materials though, the imposition of an electric field on the ends of an axis will result in a mechanical deflection along an axis at right angles to the first. Because of this, quartz is well suited in terms of chemical, electrical, and mechanical properties, to be used as a frequency control device [4].

3.2.3 LVDS Transmitter

As mentioned before, the board needs an LVDS transmitter to send the clock signals to the LVDS receiver already embedded in the RSI. In addition to LVDS being able to deliver high data rates and consume considerably less power than alternate signaling schemes, it brings the following benefits:

- Low-voltage power supply compatibility

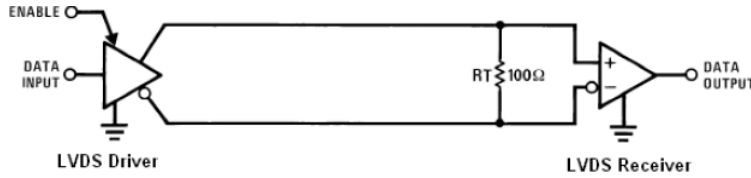
- Robust transmission signals
- High noise rejection
- Low noise generation
- Ability to be integrated into system-level ICs

LVDS is differential, meaning that it uses two signal lines to convey information. The small penalty to pay for this is the requirement of two PCB traces for the signal, but there is a very large gain in noise tolerance due to common-mode rejection. Because the signal-to-noise rejection is improved, the signal swing can be dropped to only a few hundred millivolts, enabling faster data transfer rates because the rise time is now significantly shorter [8].

One of the major requirements in selecting a particular LVDS transmitter for this application is its low-temperature capability. Because commercial LVDS transmitters that are capable of going to down to temperatures of -180°C are not widely available, it was decided to use a low-temperature transmitter previously designed by Chandradevi Ulaganathan of the Integrated Circuits and Systems Laboratory here at UT. Although this is not the best option in terms of area on the PCB, it is designed and proven to work with a similar setup. Because the signal's output voltage swing is reduced to 350 mV when using this transmitter, it allows slower edge rates and thereby reduces transient current spikes that develop from clock switching on the board. This transmitter uses only seven external connections (shown in Table 3.4 below). The 700 μA bias current is generated by placing a voltage from one of the regulators (discussed later in this chapter) across a potentiometer and series resistor of known value. The potentiometer is then adjusted until the current is calculated to be 700 μA using Ohm's law. When using this LVDS transmitter, it is important to place a 100 Ω termination resistor as close to the receiver as possible, as shown in Figure 3.6.

Table 3.4: LVDS Transmitter Connections

Pin Name	Connection
VSUB	Apply DC 0 V
VSS_LVDS_ESD	Apply DC 0 V
VREF_BGR	Apply DC 1.2 V
LVDS_OUT_N	Signal output (-)
LVDS_OUT_P	Signal output (+)
VDD_LVDS_ESD	Apply DC 3.3 V
VIN_CLK	Signal input
ISRC_700U	Provide 700 μ A bias current

Figure 3.6: Placement of 100 Ω Termination Resistor [11]

This termination resistor is used to match the characteristic impedance of the trace and it converts the driver's output to the receiver's input voltage [11].

3.2.4 Passive Components

Several passive devices are added to the board for the system to work as desired. As previously stated, PAD_REGR_OUT_ADC and PAD_PLL_CCOMP_ADC require a 2 nF and 1.2 nF capacitor tied from their respective pins to ground. The recently mentioned 100 Ω LVDS resistor needs to be connected between PAD_CLK_LVDS_ADC and PAD_CLKBAR_LVDS_ADC. The single-ended clock signal is connected to the daughterboard via SMA cable. For impedance matching of these cables, a 50 Ω termination resistor is required between the single-ended clock input and ground. After proper termination, the single-ended clock signal travels to the FPGA and the LVDS transmitter. The datasheet for the oscillator also recommends that a 0.01 μ F bypass capacitor is placed between VDD and ground as close to the oscillator as possible.

In addition to these devices, several bypass capacitor networks are placed on the board. Figure 3.7(a) shows the bypass capacitor network used for each of the 3 FPGA supplies. In order to cut down on the equivalent series resistance (ESR) of the bypass network, as well as reject a wider range of frequencies, two different ceramic capacitor values, $0.1\ \mu\text{F}$ and $0.01\ \mu\text{F}$, are placed in parallel connected between the supply voltage and ground. The bypass configuration for the rest of the supplies is similar with one key difference. As seen in Figure 3.7(b), the supply voltage passes through a surface mount $10\ \Omega$ series resistor before propagation throughout the rest of the daughterboard. By measuring the voltage drop across this resistor and using Ohm's law, the individual supply currents are calculated if necessary.

3.3 Motherboard

In addition to the daughterboard that is designed to go inside the Vanderbilt dewar, a motherboard was designed to provide voltages and bias current, the input channel signals, and to act as an interface for data collection. Figure 3.8 is a block diagram of the daughterboard. Each individual block is discussed in more detail later in this chapter.

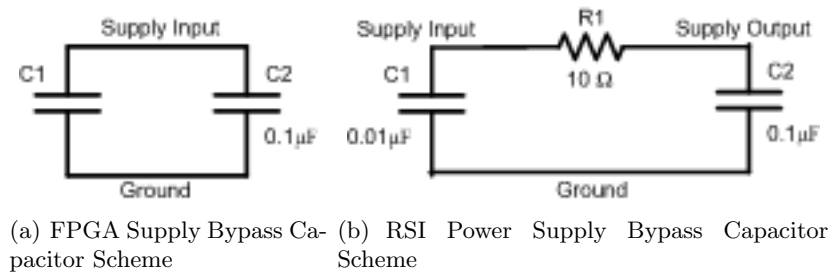


Figure 3.7: Supply Bypassing Schemes

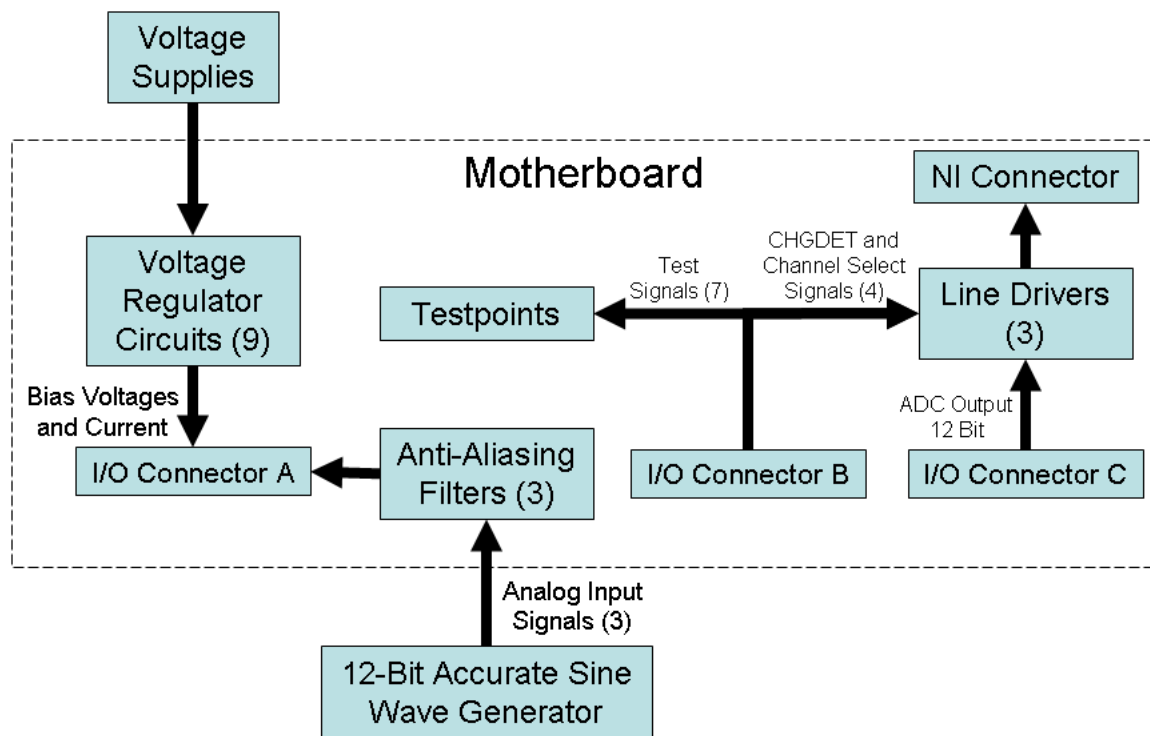


Figure 3.8: Block Diagram of Motherboard

3.3.1 Regulator Circuits

The supply voltage circuits are designed using the LM317T voltage regulator. Voltage regulators maintain a constant voltage level, regardless of input voltage (within a specified range). This particular voltage regulator is equipped with full overload protection and can output as low as 1.2 V. Figure 3.9 shows the simplest usable configuration for the regulator while Equation 3.1 shows how to determine the output voltage.

$$V_{OUT} = V_{REF} \left(1 + \frac{R2}{R1} \right) + I_{ADJ}R2 \quad (3.1)$$

A wide range of output voltages can be achieved with the adjustment of the variable resistor ($R2$). A nominal 1.25 V reference voltage (V_{REF}) is developed internally by the LM317T. The manufacturer recommends the use of a 0.1 μ F ceramic or 1 μ F solid tantalum input bypass capacitor. In addition to this, the adjustment terminal is bypassed to ground in order to improve ripple rejection. A 10 μ F bypass capacitor at the adjustment terminal provides 80 dB ripple rejection (with respect to the input) at any output level. A 10 μ F capacitor is also placed at the output which ensures stability in the feedback circuit. It is sometimes necessary to add protection diodes to prevent the capacitors from discharging through low current points into the regulator [7]. To prevent these huge current spikes from damaging the components, a protection diode is placed across the regulator's output and input terminals as well as the adjustment and output terminals. The final regulator circuit is shown in Figure 3.10 below. Table 3.5 provides the supply voltage values generated by each of the regulator circuits.

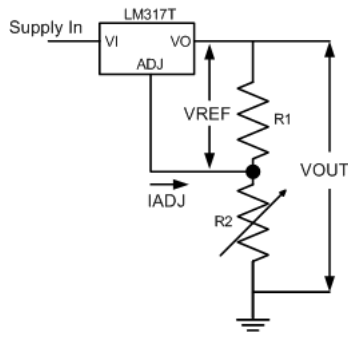


Figure 3.9: Simplest LM317 Voltage Regulator Schematic

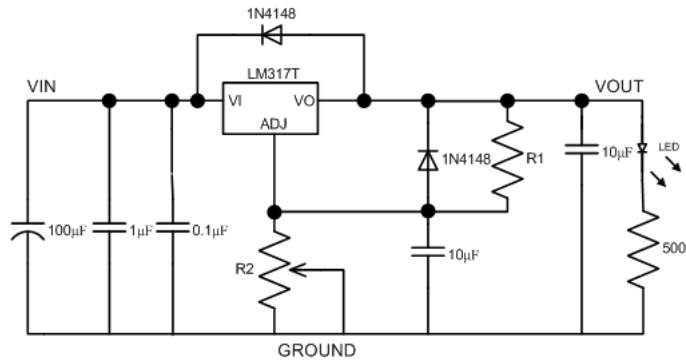


Figure 3.10: Regulator Circuit

Table 3.5: References

Signal	Output
DIG_3P3V_ADC	3.3 V
VCCR	5 V
VCCI	3.3 V
VCCA	3.3 V
VDD_BUFFER	5 V
VREF_BGR	2.4 V
VDD_ESD	3.3 V
ANA_3P3V_ADC	3.3 V
DIG_3P3V_PLL	3.3 V
ISRC_700U	700 μ A

3.3.2 Channel Inputs

The input channels employ the same SMA connections that are on the daughterboard and require the same $50\ \Omega$ termination resistor. In addition to this, anti-aliasing filters are added to each input channel. High frequency noise and interference are removed from the signal path before the analog-to-digital conversion using these filters; therefore, the digital output code of the conversion does not include the unwanted aliased harmonic information. In this case, a simple passive low-pass filter (Figure 3.11) is designed with a single resistor and capacitor, with the cutoff frequency of this filter calculated using Equation 3.2 [5].

$$f_{cutoff} = \frac{1}{2\pi RC} \quad (3.2)$$

3.3.3 Line Drivers

To collect the output data from the ADC, the outputs are brought out to the motherboard and connected to the data collection hardware. These signals; however, are run over a very long cable, and if not conditioned properly, could result in faulty information. Line drivers, amplifiers that improve signal strength at the source by driving the input to the transmission line with a higher than normal signal level, are used. This has two benefits – not only does it increase the signal

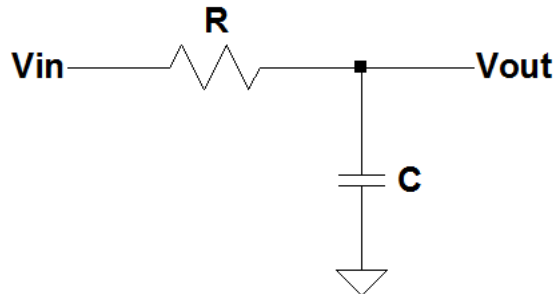


Figure 3.11: Low-pass Filter

quality, but it also steps up the ADC's output digital voltage swing from 3.3 to 5 V, which is the signal level read by the data collection system. The Texas Instruments SN74AC244 octal buffer/line driver accomplishes this task. Aside from the inputs, these buffers have pins for Vcc (the signal's new voltage level), ground, and two output-enable inputs. Vcc is connected to a 5 V source. In order for the device to pass noninverted data straight through from the inputs to the outputs, the output-enable pins are tied to ground. These buffers are commercially available in a small outline package (SOP).

3.3.4 Data Collection

The final critical component on the motherboard is the National Instruments (NI) 68-pin connector that connects to the data acquisition system. The signals connected to this include ADC output bits 0 through 11, channel select bits 0 through 3, and the CHGDET signal. These are all input signals and cannot, in any way, control the ADC.

3.4 EAGLE Design

Both PCBs are designed using Easily Applicable Graphical Layout Editor (EAGLE). EAGLE comes pre-loaded with several component libraries. Development in EAGLE is first done on the schematic level where components are selected based on type and PCB footprint. After completion, a board file is generated that places each component and its footprint. It is then up to the user to decide the board design features such as component placement, signal routing paths, copper plane pours, and number of layers.

One of the benefits of EAGLE is that it allows the design of custom components on both the schematic and footprint levels. Components such as the 145-pin PGA package, the 25-pin MDM

connectors, and the Actel FPGA are not preloaded into the EAGLE libraries. In order to design these, mechanical drawings such as the one for the 145-pin PGA (Figure 3.12(a)) are replicated using the part editor. Perhaps the two most important features to correctly draw are the pin locations so that the component is correctly soldered to the board and part perimeter which ensures that other devices are not located in the same spot. Figure 3.12(b) shows the PCB footprint for the 145-pin PGA designed in EAGLE.

3.4.1 Daughterboard

The pins on the 145-pin package used for the ADC test system are highlighted in Figure 3.13 below with the signal pins shown in blue and the VDD and ground pins shown in red. Initially, the plan was to design a 6-layer board with only the top layer used for digital signals; however, this can not be done because routing to the inner ring of pins is not possible, mainly due to the hole in the board and the fact that having two very thin signal lines running between pins would more than likely result in switching noise and crosstalk. This means that the board needs at least one extra layer for digital signals, giving a total of 7. However, the cost of 8-layer boards, is very close to the same price as 7-layer boards, thus bringing the total layer count to 8. Figures 3.14(a) and 3.14(b) below show the comparison of the board stack-ups for both the 6 and 8-layer varieties with the additional layers for both the digital signals and supplies. Figures 3.15 and 3.16 show the final daughterboard layout and schematic in the EAGLE PCB editor.

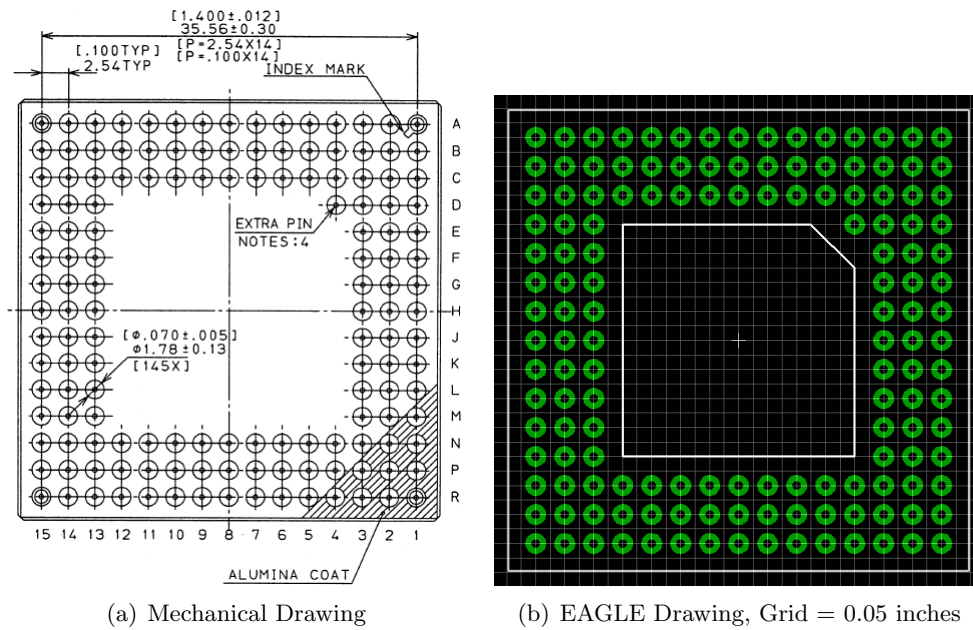


Figure 3.12: PGA Package Drawings

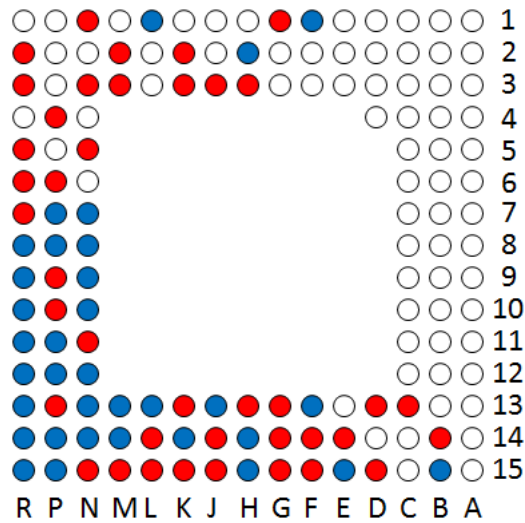


Figure 3.13: PGA Package Pins Color-coded by Function

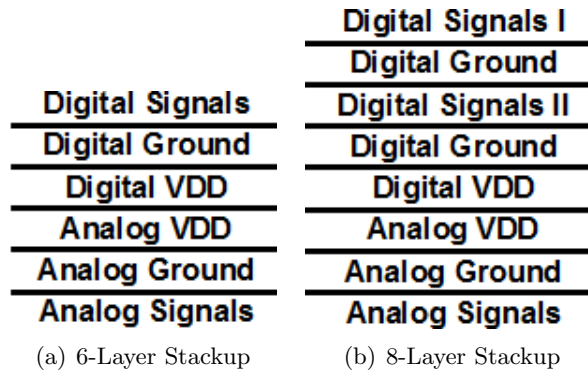


Figure 3.14: PCB Layer Comparisons

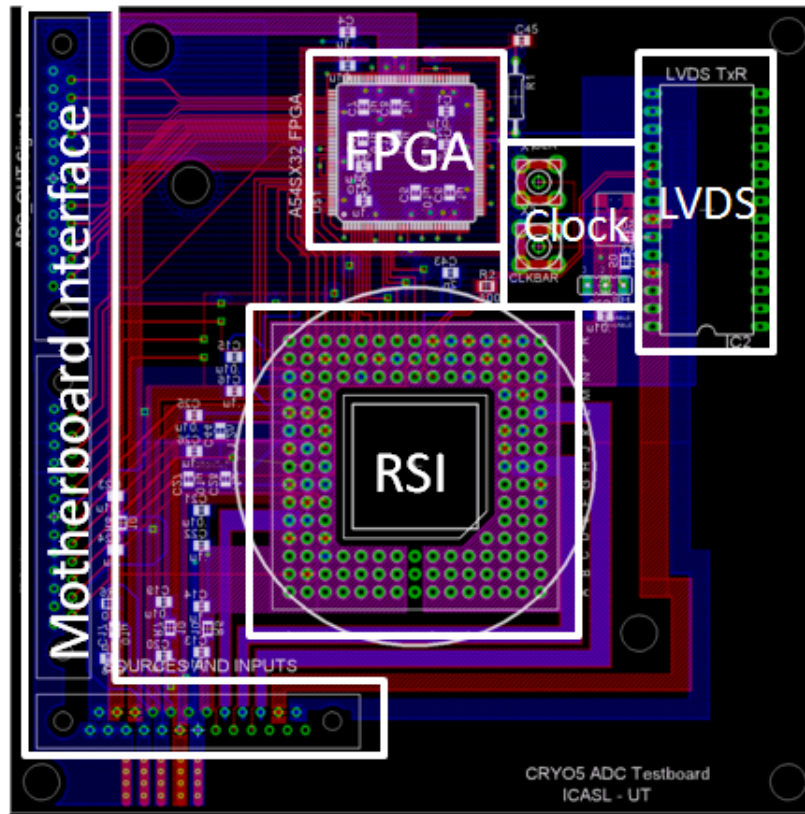


Figure 3.15: Daughterboard Layout

3.4.2 Motherboard

Since the motherboard does not have the size and location constraints that are placed on the daughterboard, its design is much simpler. Compared to the daughterboard, the motherboard also has far less signal integrity issues, allowing a 4-layer board stackup with signals being routed on the top and bottom layers, using inner layers 2 and 3 as ground and VDD planes, respectively. In addition to the board containing the supplies and elements necessary for proper data acquisition, the motherboard also contains several signals used for testing/debugging purposes. The three previously mentioned manufacturing test signals are all brought out so that they are externally tied to the proper voltage level. Other signals brought out to the motherboard include the full scale count signal, which goes high once the counter reaches maximum value, the RAPM supply that drops from 2.5 V to 0 V in the event of a latch-up event, and the ramp output. In addition to these six signals, the PLL override control signal is also brought out to the motherboard. There is an option, via a jumper, to either enable or disable the PLL, thereby bypassing the temperature compensation scheme for the ramp generator. The final motherboard design is depicted in Figures 3.17 and 3.18.

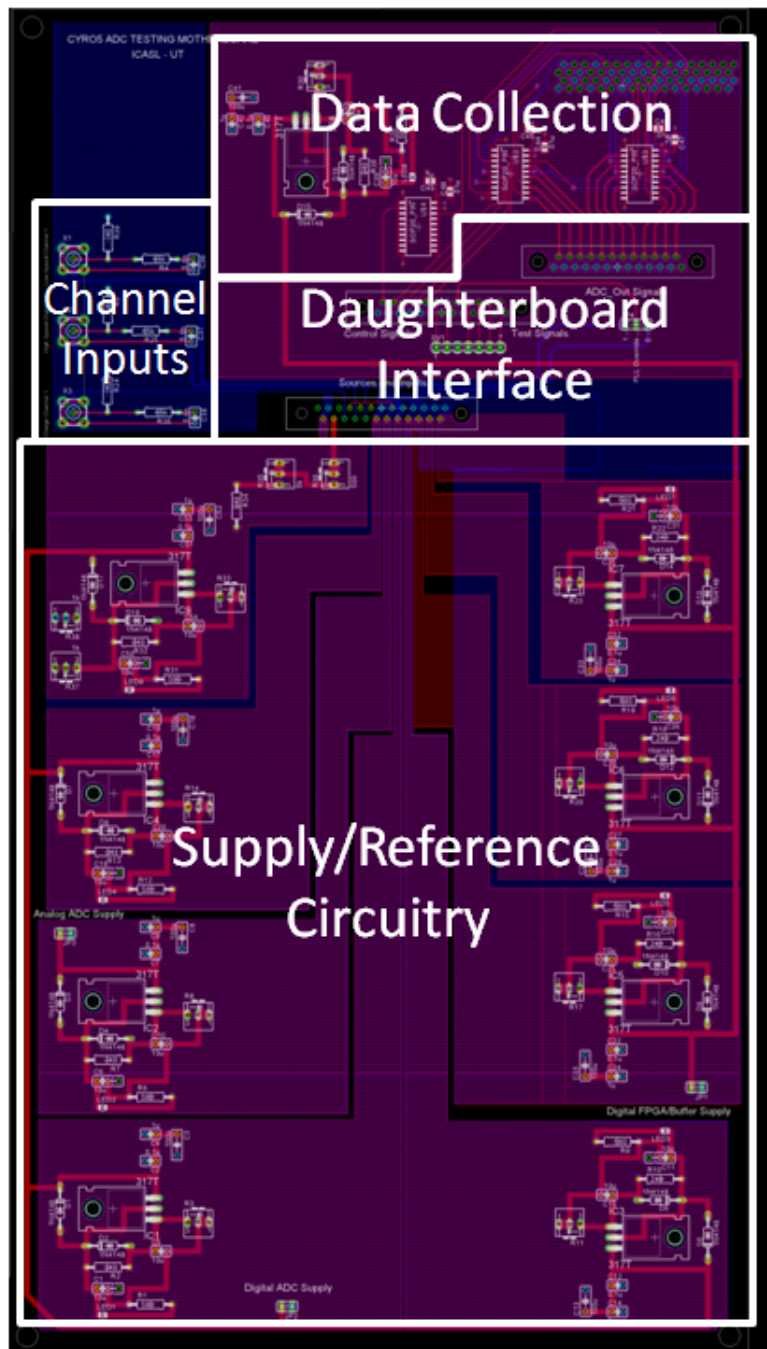


Figure 3.17: Motherboard Layout

Chapter 4

Testing and Results

4.1 Test Setup

In order to test the boards, the motherboard was first left unconnected from the daughterboard. The supply voltages were then checked by powering the board and measuring the proper output voltages from the regulators. The output voltages were then adjusted to proper voltage level by adjusting the value of the potentiometers in the voltage regulator circuits.

After confirming the supply levels, the daughterboard was placed inside the temperature chamber in the UT ICASL during preliminary testing. The ADC output signals, as well as the control and test signals, were connected using the 25-pin MDM cables. However, the supply lines and analog input signals were connected by running approximately 2-foot long wires from the motherboard to daughterboard.

The clock signals were provided by a LeCroy clock generator with 1 ns rise and fall times. A test sine wave was generated by an Audio Precision SYS-2022 waveform generator. It is important that the sine wave generator can accurately output a 12-bit waveform in order to fully test the

ADC. Voltage levels were monitored using standard multimeters and the ADC output data was processed using LabView on an external computer.

In order to calculate certain ADC parameters and evaluate its performance, it is important to capture the output data properly. A National Instruments Data Acquisition (DAQ) card is used to obtain and store the ADC's output data. The 68-pin NI connector on the motherboard connects the DAQ card to collect the ADC's outputs to store and process them using LabView.

The PCI 6534 DAQ card has 32 individually configurable I/O ports with a maximum data transfer rate of 20 MHz. The card has several different methods of data transfer, including the change-detection method where the digital data is acquired every time the state of a certain line changes. This signal links with the CHGDET signal output by the FPGA to let the DAQ system know when it is acceptable to read data.

LabView is a useful Graphical User Interface (GUI) tool that assists the computer interface with the hardware and instruments. The LabView ADC data collection software was designed by Neena Nambiar and modified by Jeremy Brantley of the UT ICASL in order to support the testing of multiple channels. In the system designed for ADC testing, LabView is used to accomplish the following tasks:

- The Audio Precision sine wave generator's output amplitude and frequency are controlled with LabView using a General Purpose Interface Bus (GPIB) card.
- The ADC data is retrieved from the testboard and transferred to the computer using the DAQ card.
- The ADC data is further processed and analyzed.

Figure 4.1 depicts a block diagram of the entire test system.

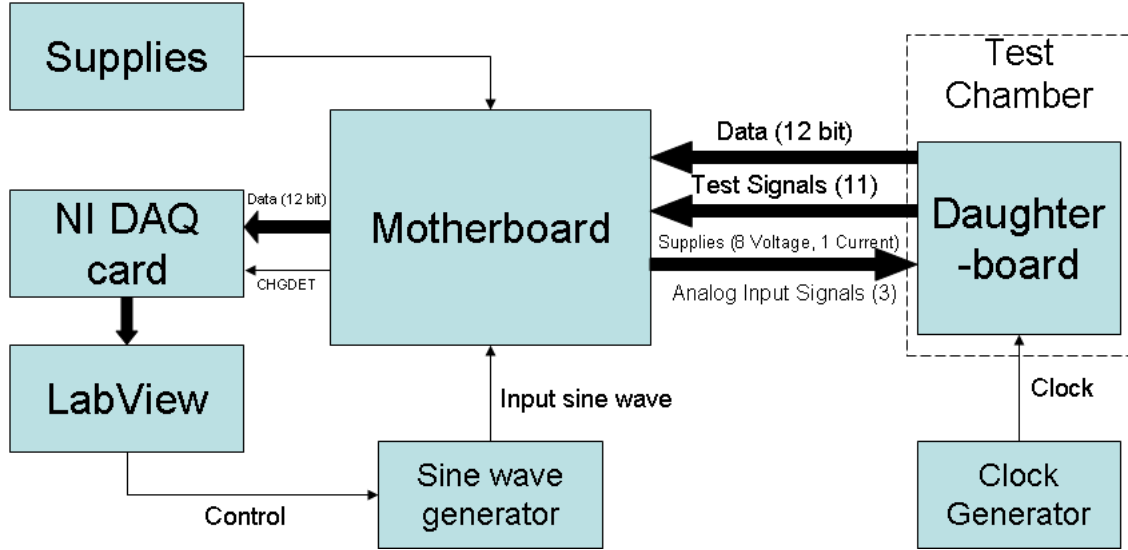


Figure 4.1: Block Diagram of Test System

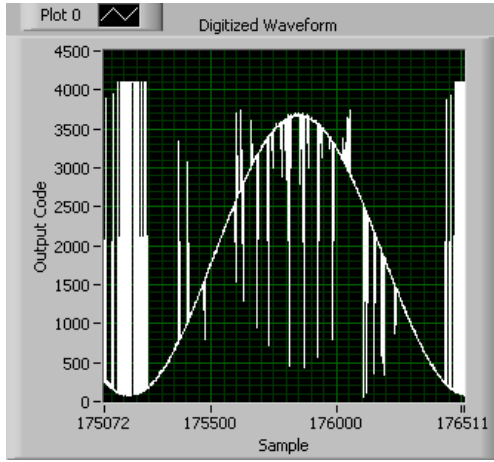
4.2 ADC Output Results

Due to difficulties in insulating the test chamber, preliminary testing data could only be obtained from -120°C to $+125^{\circ}\text{C}$. Data points were obtained for each of the three channels at temperatures of -90°C , -50°C , $+25^{\circ}\text{C}$, $+70^{\circ}\text{C}$, and $+125^{\circ}\text{C}$. A charge channel data point was also taken at -120°C . Since all of the channels had similar output characteristics, only the charge channel data is shown below in Figures 4.2 through 4.7.

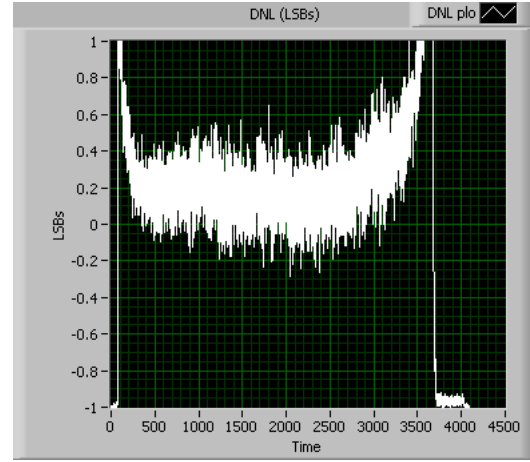
The plots show that the sine wave is much more accurately replicated at increased temperatures than at the lower temperatures. In the lower temperatures, there are many missing codes that are indicated by the spikes on the replicated sine wave. One of the most likely reasons this happens is that there are a few passive devices on the daughterboard that may change value enough at the lower temperatures to affect the way that the ramp and comparator circuitry operate. Another issue may be the PGA package itself. Since its internal bond wires contribute crosstalk, the issues that are brought on by this may be amplified at lower temperatures causing several missing bits.

Another issue could be the possibility of the FPGA not functioning properly as the temperature drops. When NASA tested it, their timing constraints may not have been as tight as the ones needed to control the ADC, thus working properly at colder temperatures for their application, but not this one. There is also a sporadic ground bounce issue brought on by the board design itself. This could be another source of error, but it is likely that this would cause the data for all temperature ranges to look similar to that of the extremely cold temperatures.

The DNL plots themselves look very good as temperature increases. For the most part, they stay within the ± 0.5 LSB range that is necessary to ensure accurate analog-to-digital conversion. At the lower temperature, however, the DNL does increase, but is still within an acceptable range.

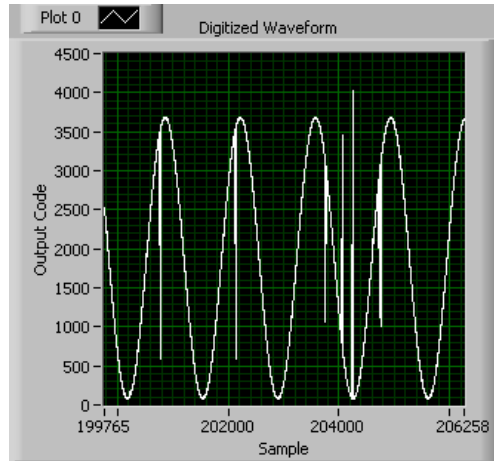


(a) Digitized Waveform

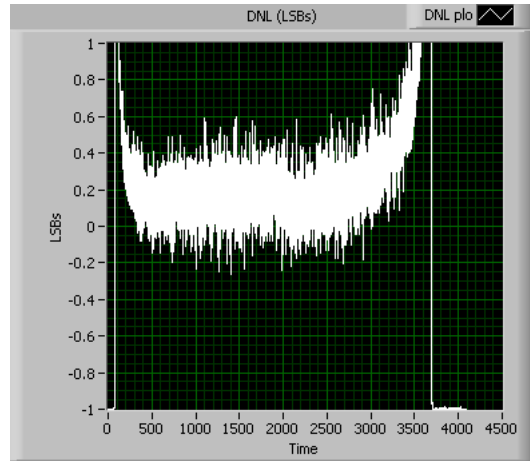


(b) DNL Plot

Figure 4.2: Charge Channel Data at -120°C

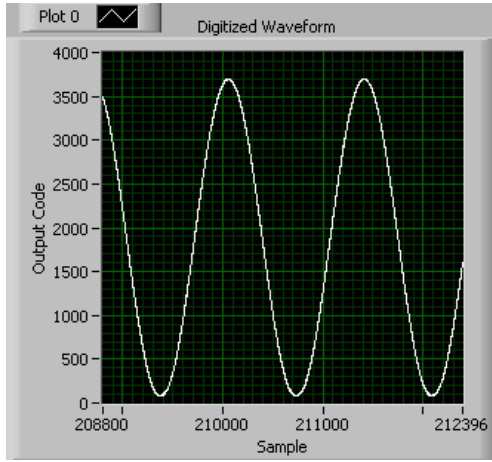


(a) Digitized Waveform

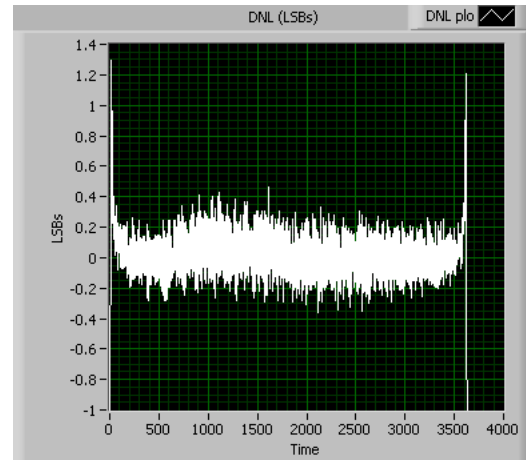


(b) DNL Plot

Figure 4.3: Charge Channel Data at -90°C

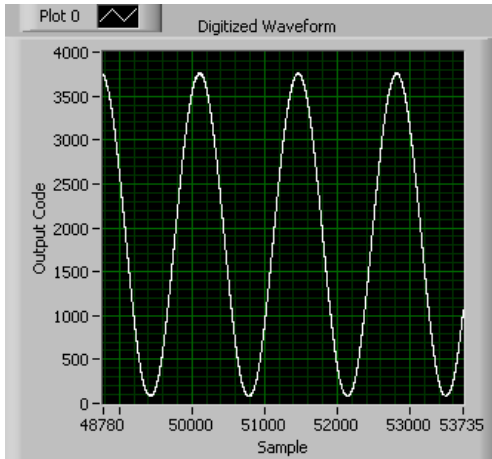


(a) Digitized Waveform

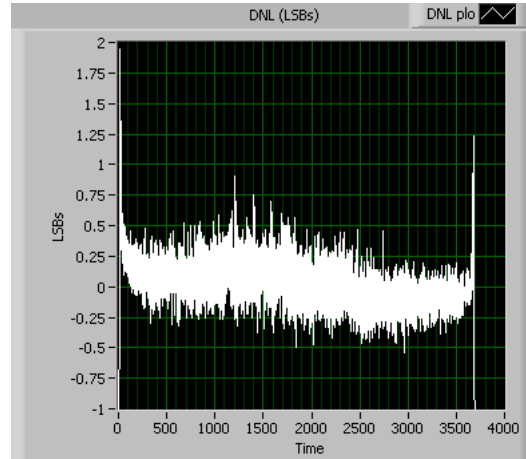


(b) DNL Plot

Figure 4.4: Charge Channel Data at -50°C

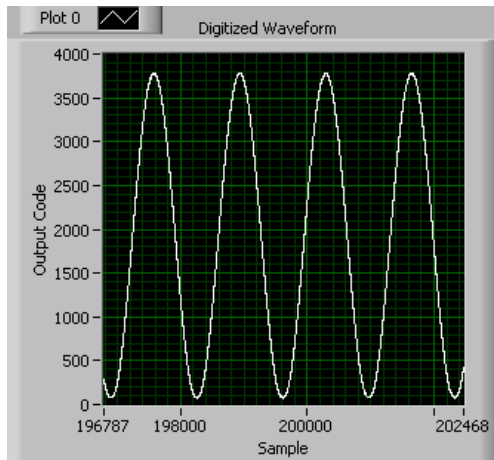


(a) Digitized Waveform

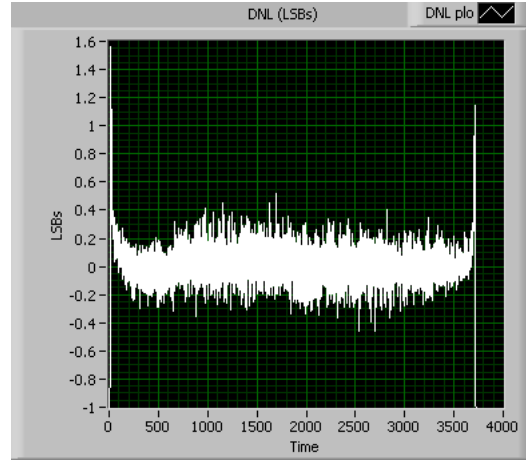


(b) DNL Plot

Figure 4.5: Charge Channel Data at $+25^{\circ}\text{C}$

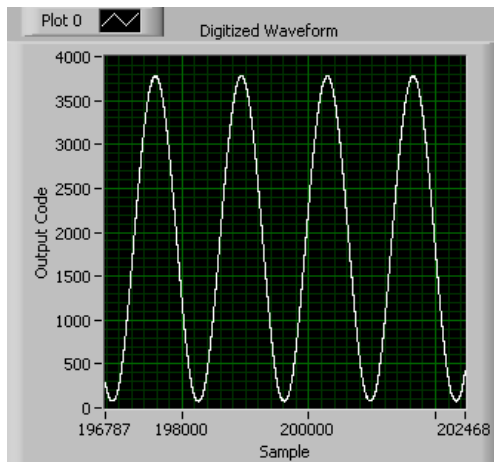


(a) Digitized Waveform

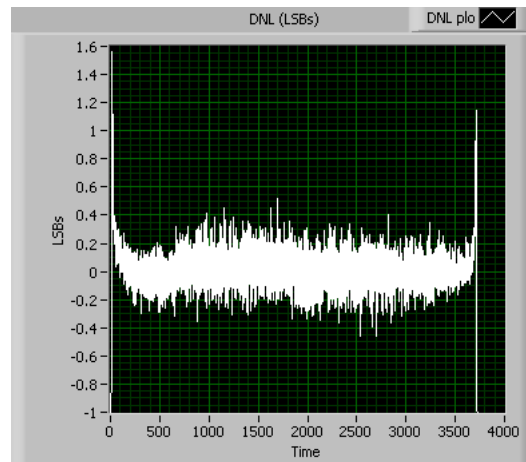


(b) DNL Plot

Figure 4.6: Charge Channel Data at +70 °C



(a) Digitized Waveform



(b) DNL Plot

Figure 4.7: Charge Channel Data at +125 °C

Chapter 5

Conclusions

5.1 Summary

The research done for this thesis was performed in order to test the irradiation performance at cryogenic temperature of a Silicon-Germanium (SiGe) Bipolar Complementary Metal Oxide Semiconductor (BiCMOS) 12-bit, 16-channel Wilkinson ADC while inside a dewar designed by Vanderbilt University. There were several obstacles to overcome in the design process. Several of these obstacles were set in place by the design of the Vanderbilt dewar, such as the placement of the test chip, the maximum size of the board, access for the cold finger, and the cabling setup. Other design constraints included the package that was selected for the test chip, and the FPGA temperature ratings.

The first phase of the research involved finding an FPGA that could meet the necessary requirements. After searching for an FPGA that was rated to operate at -180°C , no reasonable solution was found. The NASA Jet Propulsion Laboratory recommended the Actel A54SX32 FPGA, stating that they had proved it to be functional at -180°C . Aside from the fact that there was no previous

experience with Actel FPGAs to leverage, this FPGA was found to be one-time programmable, thus providing an additional constraint. This meant that if the FPGA was soldered to the testboard and it proved to not function correctly, the testboard could possibly be destroyed in the removal process, and the FPGA code would have to be redesigned.

The next step was to design a daughterboard to be placed inside the dewar while taking into account all of the associated design constraints. The main components for the daughterboard were the testchip, the FPGA, the LVDS transmitter, and the 25-pin MDM connectors. It was decided that the best approach was to use an 8-layer board with the layers optimally partitioned. A motherboard was also designed to be placed outside the dewar. This board contained the necessary supporting supply voltages and bias current as well as the circuits and components necessary for data acquisition.

The test setup consisted of the daughterboard and motherboard, a LeCroy clock generator, and an Audio Precision SYS-2022 waveform generator to provide a high fidelity test signal. The daughterboard was then placed inside the test chamber while leaving the motherboard on the outside. They were then connected using the approximately 2-foot long 25-pin MDM cables. The motherboard also interfaced with the National Instruments DAQ card in order to analyze the data using LabView. Both a digitized sine wave and DNL data were taken for temperatures ranging from -120°C to $+125^{\circ}\text{C}$. As the temperature decreased, it was much more difficult to replicate the input sine wave, causing the DNL data to worsen.

5.2 Original Contributions

- Programming and demonstration of the Actel A54SX32 FPGA for digital control of a 12-bit, 16-channel Wilkinson ADC suitable for temperatures ranging from -180°C to $+125^{\circ}\text{C}$.

- An 8-layer daughterboard was designed to meet all limiting constraints to fit inside a dewar built by Vanderbilt University in order to obtain radiation data at -180°C .
- A 4.5 inch by 4.5 inch ADC testboard that contained a large testchip, an FPGA, an LVDS transmitter, as well as several other smaller components.

Bibliography

- [1] Actel. (2006, June). SX Family FPGAs.
- [2] BAE Systems. (2009, September 23). Remote Electronics Unit Sensor Interface (RSI) ASIC Functional Requirements Specification.
- [3] Baker, J. R. (2008). *CMOS Circuit Design, Layout, and Simulation*. Hoboken: John Wiley & Sons.
- [4] FOX Electronics. (2004). *Quartz Crystal Theory of Operation*. Retrieved September 2009, from <http://www.foxonline.com/>
- [5] Microchip Technology. (1999). Anti-Aliasing, Analog Filters for Data Acquisition Systems.
- [6] Nambiar, N., Ulaganathan, C., Chen, S., Hale, M., Antonacci, A., Blalock, B. J., et al. (2008). SiGe BiCMOS 12-bit 8-Channel Low Power Wilkinson ADC. *Midwestern Symposium on Circuits and Systems*. Knoxville.
- [7] National Semiconductor. (2006, June). LM117/LM317A/LM317 3-Terminal Adjustable Regulator.
- [8] National Semiconductor. (2004, Spring). LVDS Owner's Manual.
- [9] Ramachandran, V., Alles, M. L., Reed, R. A., Black, J. D., Pellish, J. A., & Walker, D. G. (2008). In situ Cryogenic SEE Testing of High-Speed SiGe BiCMOS Devices. *International Planetary Probe Workshop*. Atlanta.
- [10] UT-ICASL. (2009, February 2). 16-Channel Wilkinson ADC (CRYO-4) Datasheet.
- [11] UT-ICASL. (2008, June 6). LVDS Transmitter (CRYO-4).

Appendix

Appendix A

FPGA Control Code

```
'timescale 1ns / 1ps

module cryo5_adc(
    Clock ,
    S1_rmp_ctrl ,
    S2_comp_rmp ,
    S3_comp_inp ,
    S4_comp_fb ,
    High_low ,
    Ch_select ,
    Reset ,
    Refresh_latch ,
    Count_enable ,
    Chip_enable ,
    ADC_in ,
    ADC_out ,
    Chgdet ,
    SOC ,
    Clk_166k
);

'define CQ 0

// *** STATE DEFINITIONS *** \\

'define ST_0          6'b000000
'define ST_1          6'b000001
'define ST_2          6'b000010
'define ST_3          6'b000011
'define ST_4          6'b000100
'define ST_5          6'b000101
'define ST_6          6'b000110
'define ST_7          6'b000111
'define ST_CH_SEL_00  6'b001000
'define ST_CH_SEL_01  6'b001001
'define ST_CH_SEL_10  6'b001010
'define ST_CH_SEL_11  6'b001011
'define ST_CH_SEL_20  6'b001100
'define ST_CH_SEL_21  6'b001101
'define ST_CH_SEL_END 6'b001110
'define ST_NEXT_CYCLE 6'b001111

// *** DEFINE TIME DELAYS *** \\

'define COUNT_RMP_INP_SWO 16'h0004
'define COUNT_AZ_INP_SWCC 16'd129                                     //16'h000A
                                                                    //16'h0081
```

```

`define COUNT_AZ_INP_SWCO    16'd137                //16'h0090
`define COUNT_INP_COMP_OPEN  16'd141                //16'h0096
`define COUNT_RMP_IC5_CLOSE  16'd145                //16'h009B
`define COUNT_RMP_START      16'd153                //16'h00AA
`define COUNT_CH_SEL_START    16'd1163              //16'd4163        //16'h0188
`define COUNT_CH_SEL_00       16'd1247              //16'd4247        //16'h023C
`define COUNT_CH_SEL_01       16'd1331              //16'd4331        //16'h02F0
`define COUNT_CH_SEL_10       16'd1415              //16'd4415        //16'h03A4
`define COUNT_CH_SEL_11       16'd1499              //16'd4499        //16'h0458
`define COUNT_CH_SEL_20       16'd1583              //16'd4583        //16'h040C
`define COUNT_CH_SEL_21       16'd1667              //16'd4667        //16'h05C0
`define COUNT_CH_SEL_END      16'd4252

```

```
// *** DEFINE OUTPUTS *** \\
```

```

output S1_rmp_ctrl;
output S2_comp_rmp;
output S3_comp_inp;
output S4_comp_fb;
output High_low;
output [3:0] Ch_select;
output Reset;
output Refresh_latch;
output Count_enable;
output [11:0] ADC_out;
output Chgdet;
output Clk_166k;
//output [5:0] State;

```

```
// *** REG DEFINITIONS *** \\
```

```

reg S1_rmp_ctrl;
reg S2_comp_rmp;
reg S3_comp_inp;
reg S4_comp_fb;
reg Count_enable;
reg Refresh_latch;
reg Chgdet;
reg High_low;
reg Reset;
reg [3:0] Ch_select;
reg [11:0] ADC_store_0;
reg [11:0] ADC_store_1;
reg [11:0] ADC_store_2;
reg [11:0] ADC_out;
reg Start_cnt_counter;
reg [15:0] Count_4095;
reg [11:0] Count_cntr4095;
reg Enable;
reg Start;
reg [5:0] State;
reg [9:0] Count_rapm = 10'b0000000000;
reg Clk_166k = 0;
//reg Start_state_counter;

```

```
// *** DEFINE INPUTS *** \\
```

```

input Clock;
input Chip_enable;
input SOC;
input [5:0] ADC_in;

```

```

always @(posedge Clock)
begin

```

```

    #CQ Enable = Chip_enable;
end
always @(posedge Clock)
begin
    #CQ Start = SOC;
end
always @(posedge Clock)
begin
    if(Enable == 1)
        begin
            if(Start == 1)
                begin
                    case (State)
                        'ST_1:
                            begin
                                #CQ S1_rmp_ctrl      = 1'b0;
                                #CQ S2_comp_rmp      = 1'b0;
                                #CQ S3_comp_inp      = 1'b0;
                                #CQ S4_comp_fb       = 1'b0;
                                #CQ Count_enable     = 1'b0;
                                #CQ Refresh_latch    = 1'b0;
                                #CQ High_low        = 1'b0;
                                #CQ Ch_select       = 4'b0000;
                                #CQ Reset          = 1'b0;
                                #CQ Start_cnt_counter = 1'b0;
                                if (Count_4095 == 'COUNT_RMP_INP_SWO)
                                    begin
                                        #CQ State = 'ST_2;
                                    end
                                else
                                    begin
                                        #CQ State = 'ST_1;
                                    end
                                end
                            end
                        'ST_2:
                            begin
                                #CQ S1_rmp_ctrl = 1'b1;
                                #CQ S3_comp_inp = 1'b1;
                                #CQ S4_comp_fb = 1'b1;
                                if (Count_4095 == 'COUNT_AZ_INP_SWCC) //1.5us MAXIMUM
                                    begin
                                        #CQ State = 'ST_3;
                                    end
                                else
                                    begin
                                        #CQ State = 'ST_2;
                                    end
                                end
                            end
                        'ST_3:
                            begin
                                #CQ S4_comp_fb = 1'b0;
                                #CQ Refresh_latch = 1'b1;
                                if (Count_4095 == 'COUNT_AZ_INP_SWCO) //94ns MINIMUM
                                    begin
                                        #CQ State = 'ST_4;
                                    end
                                else
                                    begin
                                        #CQ State = 'ST_3;
                                    end
                                end
                            end
                        'ST_4:
                            begin
                                #CQ S3_comp_inp = 1'b0;
                                if (Count_4095 == 'COUNT_INP_COMP_OPEN) //47ns MINIMUM

```

```

        begin
            #CQ State = 'ST_5;
        end
    else
        begin
            #CQ State = 'ST_4;
        end
    end
'ST_5:
    begin
        #CQ S2_comp_rmp    = 1'b1;
        #CQ Refresh_latch = 1'b0;
        if (Count_4095 == 'COUNT_RMP_ICS_CLOSE) //47ns MINIMUM
            begin
                #CQ State = 'ST_6;
            end
        else
            begin
                #CQ State = 'ST_5;
            end
        end
    end
'ST_6:
    begin
        #CQ Reset          = 1'b1;
        #CQ S1_rmp_ctrl    = 1'b0;
        if (Count_4095 == 'COUNT_RMP_START) //94ns MINIMUM
            begin
                #CQ State = 'ST_7;
                #CQ Start_cnt_counter = 1'b1;
            end
        else
            begin
                #CQ State = 'ST_6;
            end
        end
    end
'ST_7:
    begin
        #CQ Count_enable = 1'b1;
        #CQ Reset        = 1'b0;
        if (Count_4095 == 'COUNT_CH_SEL_START) //48.12us MAXIMUM
            begin
                #CQ State = 'ST_CH_SEL_00;
            end
        else
            begin
                #CQ State = 'ST_7;
            end
        end
    end
'ST_CH_SEL_00:
    begin
        #CQ High_low          = 1'b0;
        #CQ Ch_select         = 4'b0000;
        #CQ ADC_store_0[5:0] = ADC_in[5:0];
        if (Count_4095 == 'COUNT_CH_SEL_00) //1us
            begin
                #CQ State = 'ST_CH_SEL_01;
            end
        else
            begin
                #CQ State = 'ST_CH_SEL_00;
            end
        end
    end
'ST_CH_SEL_01:
    begin
        #CQ High_low = 1'b1;

```

```

#CQ ADC_store_0[11:6] = ADC.in[5:0];
#CQ ADC_out = ADC_store_0;
if (Count_4095 == 'COUNT_CH_SEL_01') //1us
begin
#CQ State = 'ST_CH_SEL_10;
#CQ Chgdet = ~Chgdet;
end
else
begin
#CQ State = 'ST_CH_SEL_01;
end
end
'ST_CH_SEL_10:
begin
#CQ High_low = 1'b0;
#CQ Ch_select = 4'b1100;
#CQ ADC_store_1[5:0] = ADC.in[5:0];
if (Count_4095 == 'COUNT_CH_SEL_10') //1us
begin
#CQ State = 'ST_CH_SEL_11;
end
else
begin
#CQ State = 'ST_CH_SEL_10;
end
end
'ST_CH_SEL_11:
begin
#CQ High_low = 1'b1;
#CQ ADC_store_1[11:6] = ADC.in[5:0];
#CQ ADC_out = ADC_store_1;
if (Count_4095 == 'COUNT_CH_SEL_11') //1us
begin
#CQ State = 'ST_CH_SEL_20;
#CQ Chgdet = ~Chgdet;
end
else
begin
#CQ State = 'ST_CH_SEL_11;
end
end
'ST_CH_SEL_20:
begin
#CQ High_low = 1'b0;
#CQ Ch_select = 4'b1110;
#CQ ADC_store_2[5:0] = ADC.in[5:0];
if (Count_4095 == 'COUNT_CH_SEL_20') //1us
begin
#CQ State = 'ST_CH_SEL_21;
end
else
begin
#CQ State = 'ST_CH_SEL_20;
end
end
'ST_CH_SEL_21:
begin
#CQ High_low = 1'b1;
#CQ ADC_store_2[11:6] = ADC.in[5:0];
#CQ ADC_out = ADC_store_2;
if (Count_4095 == 'COUNT_CH_SEL_21') //1us
begin
#CQ State = 'ST_CH_SEL_END;
#CQ Chgdet = ~Chgdet;
#CQ High_low = 1'b0;

```

```

        end
    else
        begin
            #CQ State = 'ST.CH.SEL_21;
        end
    end
'ST.CH.SEL_END:
    begin
        if (Count_cntr4095 >= 12'd4095)
            begin
                #CQ Start_cnt_counter = 1'b0;
                #CQ State = 'ST.NEXT_CYCLE;
                #CQ S2_comp_rmp = 1'b0;
                #CQ Count_enable = 1'b0;
            end
        end
'ST.NEXT_CYCLE:
        begin
            if (Count_4095 == 'COUNT.CH.SEL_END)
                begin
                    #CQ State = 'ST_0;
                end
            else
                begin
                    #CQ State = 'ST.NEXT_CYCLE;
                end
            end
        end
'ST_0:
        begin
            #CQ Ch_select = 4'b0000;
            #CQ State = 'ST_1;
        end
    default:
        begin
            #CQ S1_rmp_ctrl = 1'b0;
            #CQ S2_comp_rmp = 1'b1;
            #CQ S3_comp_inp = 1'b0;
            #CQ S4_comp_fb = 1'b0;
            #CQ Count_enable = 1'b0;
            #CQ Refresh_latch = 1'b0;
            #CQ High_low = 1'b0;
            #CQ Ch_select = 4'b0000;
            #CQ Reset = 1'b0;
            #CQ State = 'ST_1;
            #CQ Start_cnt_counter = 1'b0;
        end
    endcase
end
else
    begin
        #CQ State = 'ST_1;
    end
end
else
    begin
        #CQ S1_rmp_ctrl = 1'b0;
        #CQ S2_comp_rmp = 1'b0;
        #CQ S3_comp_inp = 1'b0;
        #CQ S4_comp_fb = 1'b0;
        #CQ Chgdet = 1'b0;
        #CQ Count_enable = 1'b0;
        #CQ Refresh_latch = 1'b0;
        #CQ High_low = 1'b0;
        #CQ Ch_select = 4'b0000;
        #CQ Reset = 1'b0;
    end
end

```

```

        #CQ State = 'ST_1;
        #CQ Start_cnt_counter = 1'b0;
    end
end

always @(posedge Clock)
begin
    if(Enable == 1)
        begin
            if(Start == 1)
                begin
                    if (Count_4095 >= 'COUNT.CH.SEL.END)
                        begin
                            #CQ Count_4095 = 16'h0000;
                        end
                    else
                        begin
                            #CQ Count_4095 = Count_4095 + 1'b1;
                        end
                    end
                end
            else
                begin
                    #CQ Count_4095 = 16'h0000;
                end
            end
        end
    else
        begin
            #CQ Count_4095 = 16'h0000;
        end
    end
end

always @(posedge Clock)
begin
    if(Enable == 1)
        begin
            if(Start == 1)
                begin
                    if(Start_cnt_counter == 1)
                        begin
                            #CQ Count_cntr4095 = Count_cntr4095 + 1'b1;
                        end
                    else
                        begin
                            #CQ Count_cntr4095 = 12'h000;
                        end
                    end
                end
            else
                begin
                    #CQ Count_cntr4095 = 12'h000;
                end
            end
        end
    else
        begin
            #CQ Count_cntr4095 = 12'h000;
        end
    end
end

always @(posedge Clock)
begin
    if (Enable == 1'b1)
        begin
            if (Count_rapm == 10'h0FA)
                begin
                    Clk_166k = ~ Clk_166k;
                    Count_rapm = 10'h000;
                end
            end
        end
    end
end

```

```

        end
    else
        begin
            Count_rapm = Count_rapm + 1;
        end
        end
    else
        begin
            Clk_166k = 1'b0;
            Count_rapm = 10'h000;
        end
    end
end
endmodule

```

Appendix B

FPGA Testbench Code

```
'timescale 1ns/1ps

module cryo5_rapm_tb;
  reg Clock = 1'b0;
  wire S1_rmp_ctrl;
  wire S2_comp_rmp;
  wire S3_comp_inp;
  wire S4_comp_fb;
  wire High_low;
  wire [3:0] Ch_select;
  wire Reset;
  wire Refresh_latch;
  wire Count_enable;
  reg Chip_enable = 1'b0;
  reg [5:0] ADC_in = 6'b000000;
  wire [11:0] ADC_out;
  wire Chgdet;
  reg SOC = 1'b0;
  wire Clk_166k;
  // wire [5:0] State;
  // wire [15:0] Count_4095;
  // wire [11:0] Count_cntr4095;
  // wire Start_cnt_counter;
  // wire [9:0] Count_rapm;
  // wire Start_state_counter;

  parameter PERIOD = 12;
  parameter real DUTY_CYCLE = 0.5;
  parameter OFFSET = 0;

  initial // Clock process for Clock
  begin
    #OFFSET;
    forever
      begin
        Clock = 1'b0;
        #(PERIOD-(PERIOD*DUTY_CYCLE)) Clock = 1'b1;
        #(PERIOD*DUTY_CYCLE);
      end
  end

  cryo5_adc UUT (
    .Clock(Clock),
    .S1_rmp_ctrl(S1_rmp_ctrl),
    .S2_comp_rmp(S2_comp_rmp),
    .S3_comp_inp(S3_comp_inp),
```

```

.S4_comp_fb(S4_comp_fb),
.High_low(High_low),
.Ch_select(Ch_select),
.Reset(Reset),
.Refresh_latch(Refresh_latch),
.Count_enable(Count_enable),
.Chip_enable(Chip_enable),
.ADC_in(ADC_in),
.ADC_out(ADC_out),
.Chgdet(Chgdet),
.SOC(SOC),
.Clk_166k(Clk_166k));
// .State(State);
// .Count_4095(Count_4095),
// .Count_cntr4095(Count_cntr4095),
// .Start_cnt_counter(Start_cnt_counter),
// .Count_rapm(Count_rapm));

initial begin
// ----- Current Time: 869ns
#100;
Chip_enable = 1'b1;
SOC = 1'b1;
ADC_in = 6'b101010;
#15400
ADC_in = 6'b010101;
#1113
ADC_in = 6'b011001;
#1008
ADC_in = 6'b110010;
#1113
ADC_in = 6'b101011;
#1008
ADC_in = 6'b000011;
#1113
ADC_in = 6'b000000;
#44770
ADC_in = 6'b011111;
#1008
ADC_in = 6'b111000;
#1008
ADC_in = 6'b110011;
#1008
ADC_in = 6'b101011;
#1008
ADC_in = 6'b001101;
#1008
ADC_in = 6'b111101;
#1008
ADC_in = 6'b000000;
#44770
ADC_in = 6'b110101;
#1008
ADC_in = 6'b011101;
#1008
ADC_in = 6'b111001;
#1008
ADC_in = 6'b011011;
#1008
ADC_in = 6'b111100;
#1008
ADC_in = 6'b001010;
#1008
ADC_in = 6'b000000;
// -----

```

```
end  
endmodule
```

Vita

Zachary William Pannell was born in Knoxville, Tennessee on February 24, 1986. He graduated from Roane County High School in Kingston, Tennessee in 2004 and then went on to the University of Tennessee to pursue his Bachelor of Science in Electrical Engineering graduating in May 2008. He continued his education at the University of Tennessee where he is currently pursuing his Master of Science degree in Electrical Engineering. Over the course of his studies he also participated in several internships at the Y12 National Security Complex in Oak Ridge, Tennessee and at the NASA Jet Propulsion Laboratory in Pasadena, California.