

2017196

by Jack Rogers

Submission date: 28-Feb-2023 11:39PM (UTC+0000)

Submission ID: 198188964

File name: 16489_Jack_Rogers_2017196_181628_877698946.pdf (2.37M)

Word count: 4669

Character count: 31180

Course: HND Electrical and Electronic Engineering

Module: SGEE5007 Virtual Instrumentation and Control

Portfolio: Magnitude Comparator

Name: Jack Rogers

University number: 2017196

Date submitted: 28/02/2023

Table of Contents

1. Element 1	6
1.1. Introduction	6
1.2 Research.....	7
1.2.1 Digital design methods using VHDL and IP integration.....	7
1.2.2 NI DAQmx data acquisition used in LabVIEW.	9
1.2.3 Digital Magnitude Comparators.....	10
1.2.4 Analogue to Digital Converters.	11
1.2.5 Load Cells (namely LCKD-1Kg).....	11
1.2.6 Instrumentation and signal conditioning.....	12
1.3 Design Requirements.....	13
1.3.1 Half Adder	13
1.3.2 Full Adder	14
1.3.3 4-Bit Adder	15
1.4 Implementation	16
1.4.1 4 Bit Adder	16
1.4.2 Combining 4-Bit adder and 4-Bit Magnitude Comparator.....	17
1.4.3 8-Bit Magnitude Comparator Design	18
1.5 Testing and Analysis.....	19
1.5.1 4-Bit Adder and Comparator FPGA Testing	19
1.5.2 Test Bench for 4-Bit Adder	22
1.5.3 Test bench 4-bit mag comp	23

1.5.4 8-Bit Magnitude Comparator PLD	24
1.5.3 Testing 8-Bit Magnitude Comparator -FPGA	24
1.6 Evaluation	25
2. Element 2	26
2.1 Introduction	26
2.2 Design Requirements	26
2.3 Implementation	27
2.3.1 10-Bit ADC Prototype	28
2.3.2 12-Bit ADC Prototype	30
2.4 Testing and Analysis	31
2.4.1 Element 2 Test results	31
Evaluation of results	32
2.5 Evaluation	33
3. Element 3	34
3.1 Design Requirements	34
3.2 Implementation	35
3.3 Testing and Analysis	36
4. References	37
Bibliography	37
5. Appendix	39
Unity gain amplifier	39
Files for Element 3 block diagram	46

Good comprehensive table of contents

Figure 1 - Example of VHDL Code	7
Figure 2 - Example of IP Integration.....	8
Figure 3 - Example of DAQmx	9
Figure 4 - Example of 4bit Magnitude Comparator	10
Figure 5 - Logic Diagram for Half Adder	13
Figure 6 - Logic Diagram for Full Adder	14
Figure 7 - Truth Table for 4-Bit Adder	15
Figure 8 - Logic Diagram for 4-Bit Adder.....	15
Figure 9 - Hierarchy for 4bit Adder	16
Figure 10 – 4-bit adder code	16
Figure 11 -IP for 4-Bit magnitude Comparator	17
Figure 12 – 4-bit Magnitude Comparator Multisim.....	17
Figure 13 – Hierarchy for 8-Bit Magnitude Comparator.....	18
Figure 14 –Multisim Circuit for 8-bit Magnitude Comparator.....	18
Figure 15 - A equal to B.....	19
Figure 16 - A Less than B	20
Figure 17 - A Greater than B	21
Figure 18 - 4-Bit Adder Code for testing	22
Figure 19 - Simulation for 4-Bit Adder	22
Figure 20 - Code for 4-Bit Magnitude Comparator testing.....	23
Figure 21 - Simulation for 4-Bit magnitude Comparator	23

Figure 22 -PLD Circuit for 8-Bit Mag Comparator	24
Figure 23 - Test Results for 8-Bit mag Comp - FPGA.....	24
Figure 24 - Instrumentation Amplifier Circuit.....	27
Figure 25 - Graph of Results from Instrument Amplifier and Load Cell.....	28
Figure 26 - Block Diagram for 10-Bit ADC	28
Figure 27 - Test 1 for 10-Bit ADC.....	29
Figure 28 - Test 2 for 10-Bit ADC.....	29
Figure 29 - Test 1 for 12-Bit ADC.....	30
Figure 30 - Test 2 for 12-Bit ADC.....	30
Figure 31 - 700grms Less than Input.....	31
Figure 32 - 700grms Equal to the Input	31
Figure 33 - 700grms Greater than the Input.....	32
Figure 34 - Final Circuit (Element 3).....	34
Figure 35 - Configure Write to Measurement File.....	35
Figure 36 - LabView Files Generated	35
Figure 37 - Magnitude Comparator Excel Graph	36

Good detailed list of figures

Table 1 - Truth table for Half Adder	13
Table 2 - Truth Table for Full Adder	14
Table 3 - Comparison Table for Load Cell Operation	27
Table 4 - Table of Results for 8-Bit Mag.....	32

Good use of tables

1. Element 1

1.1. Introduction

Element 1's goal is to use VHDL and IP integration to conduct research, design, and implement a combinational logic system. Software programmes Vivado and NI Multisim is needed to finish the design, functional testing and the build. Once each test has been successfully run using the tools provided by its package, it will be downloaded onto an FPGA board for physical testing. The effectiveness of the method will next be demonstrated by conducting an analysis on both circuits to make sure they are operating as intended.

Introduction demonstrates an understanding of requirements

1.2 Research

1.2.1 VHDL and IP integration.

VHDL is a hardware description language used to design digital systems. It is used by designers to write and simulate digital logic designs and programmable logic devices. VHDL can also be used for the design of integrated circuits and other digital systems.

```
library ieee;
use ieee.std_logic_1164.all;
entity decoder_3x8 is
port (a : in std_logic_vector(0 to 2) ;
      d : out std_logic_vector (0 to 7));
end decoder_3x8;
architecture decoder_arch of decoder_3x8 is
begin
  process (a)
  begin
    case a is
      when "000" => d <= "00000001";
      when "001" => d <= "00000010";
      when "010" => d <= "00000100";
      when "011" => d <= "00001000";
      when "100" => d <= "00010000";
      when "101" => d <= "00100000";
      when "110" => d <= "01000000";
      when "111" => d <= "10000000";
      when others => d <= "00000000";
    end case;
  end process;
end decoder_arch;
```

Relevant example

Figure 1 - Example of VHDL Code

VHDL is a text-based language used to represent the behavior and structure of digital systems. It is mainly used for the design of digital integrated circuits, but it can also be used for the design of hardware description languages (HDLs). VHDL allows designers to create, simulate, and test digital systems before they are physically implemented. (Land, 2022)

Good significant points

IP integration is a technology used to integrate intellectual property (IP) into digital designs. IP integration is an important part of the design process, as it helps to reduce design time and cost.

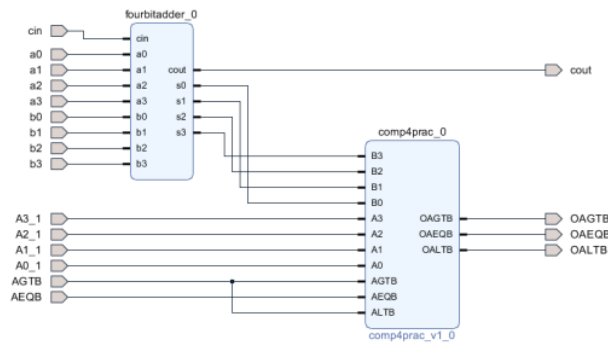


Figure 2 - Example of IP Integration

Good relevant example

IP integration allows designers to use pre-existing IP blocks to quickly create complex designs. By using pre-existing IP blocks, designers can avoid the need to create custom designs, which can be time-consuming and expensive. IP integration also provides designers with access to advanced functionalities and architectures that may not be available in custom designs. (IEEE, 2023)

Good explanation of IP block design with recognition of design possibilities

VHDL and IP integration are often used together to create digital designs. By combining the two technologies, designers can create complex digital designs in a shorter amount of time. VHDL is used to create the design logic and IP integration is used to add pre-existing IP blocks to the digital design.

Designers can also use VHDL and IP integration together to create digital designs that include both hardware and software components. By combining the two technologies, designers can create digital designs that are both powerful and cost-effective.

VHDL and IP integration are both powerful technologies that can be used to create sophisticated digital designs. By combining the two technologies, designers can reduce design time and cost while creating powerful digital designs.

There are good findings with this topic area, and supported with relevant diagram and ref sources.

1.2.2 NI DAQmx used in LabVIEW.

National Instruments (NI) DAQmx is an application used to acquire, analyze, monitor and control data. It is the software platform that runs on National Instruments hardware. NI DAQmx is used to acquire, analyze and control data from sensors, instruments and other sources. It is built on a modular architecture that allows it to be used in a variety of applications. (CORP., 2023)

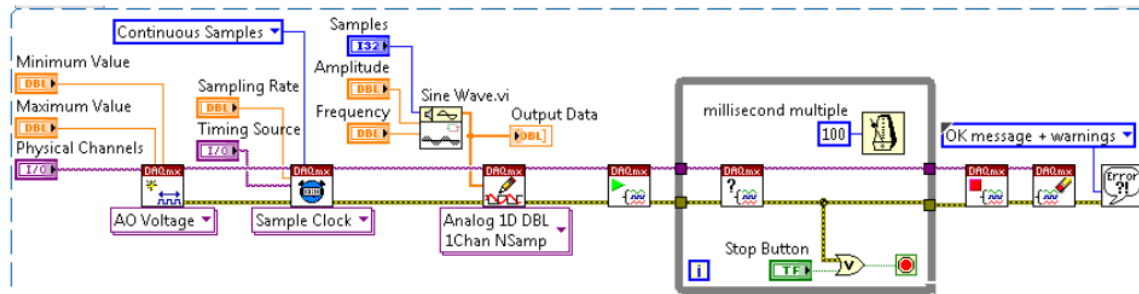


Figure 3 - Example of DAQmx

NI DAQmx is used in LabVIEW, a graphical programming language developed by National Instruments. The LabVIEW platform is used to develop applications for data acquisition, instrument control and analysis. It provides a graphical programming language for the development of user interfaces and software applications. LabVIEW also provides the ability to rapidly build custom systems for data acquisition and control.

NI DAQmx is used with LabVIEW to acquire data from various devices and sensors. It is used to control sensors and instruments, and to analyse the acquired data. It can be used for everything from basic data logging to complex control systems. (Corp, 2023)

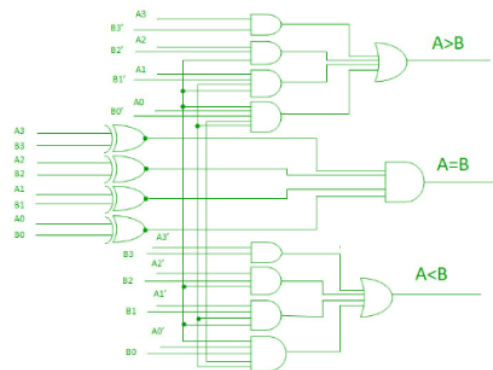
NI DAQmx is a powerful and flexible tool for data acquisition and control. It is used in LabVIEW to create custom systems for data acquisition, instrument control and analysis. It provides a wide variety of features and functions for data acquisition, instrument control and analysis. It is used to acquire data from a variety of sources, including sensors, instruments, and other sources. It is used to control sensors and instruments, and to analyze the acquired data. Additionally, it provides support for multiple types of output, including analog, digital, and motion control.

A good general description of LabVIEW and the DAQ practical application. The image is a relevant example, however, the block functions of the DAQ should have been identified and discussed

1.2.3 Digital Magnitude Comparators.

There is no restriction to 2 digital inputs

Digital magnitude comparators are used to compare two digital inputs and output the result as a single bit that indicates which of the two inputs has the greater magnitude. The output of a digital magnitude comparator is a single bit that can be either high (1) or low (0). Generally, a high output indicates that the first input has the greater magnitude, while a low output indicates that the second input has the greater magnitude. Figure 4 shows an example of a 'cascading' multiple comparators together, in this case a 4bit compactor. (Pandey, 2022)



The example shows the 3 possible conditions. The explanation should identify the 3 conditions.

Figure 4 - Example of 4bit

Magnitude Comparator

Digital magnitude comparators are used in a variety of applications, including: digital signal processing, logic circuits, arithmetic logic units, and computer-aided design (CAD). In digital signal processing, digital magnitude comparators are used to compare two digital signals to determine which one has the higher magnitude. This can be used to detect the presence of certain frequency components in a signal. In logic circuits, digital magnitude comparators are used to compare two variables and output a Boolean result indicating which one is greater. This can be used to determine if a certain condition is met or not. In arithmetic logic units, digital magnitude comparators are used to compare two numbers and output the result indicating which one is greater. This can be used to compare the results of two calculations and determine which one is greater. (ElectronicsTutorials, 2023)

A broad application of digital comparators is explained and supported with example and ref source. This indicates a good study of the topic area.

Digital magnitude comparators are useful for a variety of applications, as they can be used to quickly and accurately compare two digital signals, variables, numbers, or designs. They are easy to implement, either in hardware or in software, and can be used in a variety of contexts.

1.2.4 Analogue to Digital Converters.

Analogue to Digital Converters (ADCs) are devices used to convert analogue signals into digital signals. The purpose of this conversion is to enable the digital signals to be processed by a computer or other electronic device. The ADC typically uses a combination of analogue and digital circuitry to convert the analogue signal into a series of digital values.

Analogue signals are generated by physical sources such as sensors, microphones and motors. These signals are continuous and can be represented by a waveform. Digital signals, on the other hand, are discrete and represented by a series of numbers. The ADC converts the analogue signal into a numeric representation by taking samples of the signal at regular intervals. The samples are then converted into digital values by quantizing the data into a range of discrete values. (AspenCore, 2023)

In conclusion, Analogue to Digital Converters are devices used to convert analogue signals into digital signals. They are used in a variety of applications and come in a variety of sizes and configurations. The accuracy and resolution of the ADC is determined by its sampling rate, resolution and dynamic range. *A definition of ADC's is included, however, the different methods are not covered*

1.2.5 Load Cells (namely LCKD-1Kg)

Load cells are transducer devices that are used for measuring the weight of an object or a system. They are commonly used in industrial and scientific settings to measure force or tension. The LCKD-1Kg load cell is a single point, single bridge force transducer. It is constructed using a metal alloy with strain gauges bonded to the surface. The strain gauges measure the amount of strain, or deformation, that is caused by the applied force. This strain is converted into an electrical signal that can be read and interpreted by a computer or other device. The LCKD-1Kg load cell can be connected to a variety of devices, such as a digital display, computer, or PLC system. The output of the load cell is typically in the form of a voltage signal, which can then be read by the connected device. (inc., 2023) *A general description and application example with relevant source, that needs a supporting image with identifiable parts.*

1.2.6 Instrumentation and signal conditioning.

Instrumentation and signal conditioning are two interrelated concepts in measuring and monitoring physical processes. Instrumentation is the use of instrumentation to measure physical quantities such as pressure, temperature, flow, and level. Signal conditioning is the process of manipulating an electrical signal in order to prepare it for further processing. Instrumentation and signal conditioning can be used to measure and monitor a wide range of physical processes, from industrial operations to medical diagnostics. (HBK, 2023)

Good points included with relevant source

Instrumentation is the use of instruments to measure physical parameters. The most used instruments are sensors, transducers, and meters. Sensors are devices that measure a physical quantity and convert it into an electrical signal. These signals can then be amplified, filtered, and processed to obtain the desired output. Examples of sensors include thermocouples, strain gauges, and pressure transducers. Transducers are devices that convert one form of energy into another. Examples of transducers include ultrasonic transducers and piezoelectric transducers. Meters are devices that measure the magnitude of an electrical signal. Examples of meters include voltmeters and ammeters. (Systems, 2019)

Instrumentation and signal conditioning are used in many fields, from industrial operations to medical diagnostics. In industrial operations, instrumentation and signal conditioning are used to monitor and control processes such as temperature, pressure, flow, and level. In medical diagnostics, instrumentation and signal conditioning are used to measure and analyse physiological parameters such as heart rate and blood pressure.

There is repetition here

In summary, instrumentation and signal conditioning are two interrelated concepts in measuring and monitoring physical processes. Instrumentation is the use of instruments to measure physical parameters, while signal conditioning is the process of manipulating an electrical signal in order to prepare it for further processing. Instrumentation and signal conditioning are used in many fields, from industrial operations to medical diagnostics.

This is a repetition of above

The content includes repetition, and would have benefitted from a block diagram to illustrate a process with signal conditioning

1.3 Design Requirements

A 4-bit adder is created in this task with the Vivado software from Xilinx using the VHDL language. To create the design of the 4-bit adder, the procedure is divided into phases. A 4-bit adder consists of 4 complete adders, with 2 half adders inside each full adder.

This demonstrates an understanding of the hierarchy for the VHDL

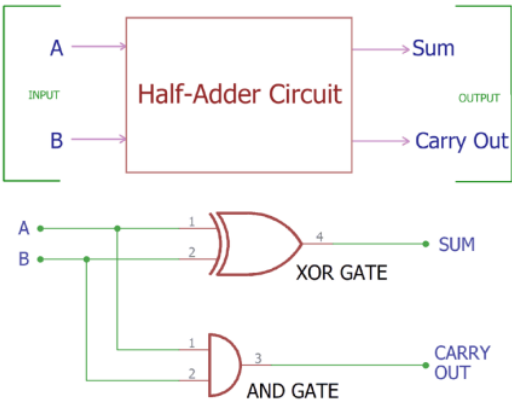
1.3.1 Half Adder

Table 1 - Truth table for Half Adder

Inputs		Outputs	
A	B	S	C
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

A half adder is a digital circuit that can add two binary digits (0s and 1s). It has two inputs (A and B) and two outputs (S and C). The output S is the sum of the two inputs, and the output C is the carry out. A half adder can be represented by a truth table, which shows the relationship between the inputs and outputs of the circuit. 'Table 1' is a truth table for Half adder. The truth table shows that if both inputs are 0, the sum and carry outputs will be 0. If only one input is 1, the sum output will be 1 and the carry output will be 0. If both inputs are 1, the sum output will be 0 and the carry output will be 1.

A half adder can also be represented by a logic diagram, which shows the logic gates used to implement the circuit.



Good example

Figure 5 - Logic Diagram for Half Adder

Image should include a ref source

1.3.2 Full Adder

Table 2 - Truth Table for Full Adder

A	B	Carry IN	Sum	Carry OUT
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

A full bit adder is a digital logic circuit that adds two binary numbers, often denoted as A and B, together to produce a sum and a carry. The sum is a binary number representing the result of the addition of the two binary numbers, while the carry is a single bit representing the carry-out of the addition. The circuit typically consists of two logic gates, typically an XOR gate and an AND gate, and two carry-in bits, one for each of the two binary numbers being added. The XOR gate takes the two binary numbers as inputs and produces a sum. The AND gate takes the two binary numbers as inputs and produces a carry-out. The two carry-in bits represent the carry-in from the previous bit-adder, if any.

Good explanation with example

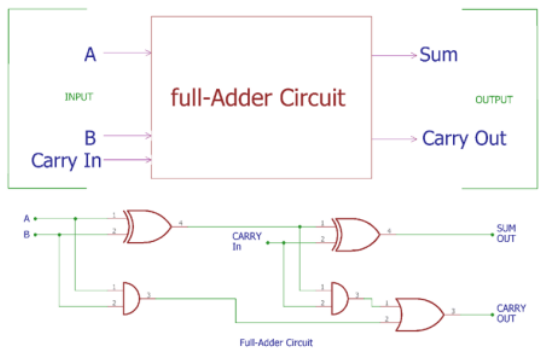


Figure 6 - Logic Diagram for Full Adder

The image should include a ref source

1.3.3 4-Bit Adder

Cin	A				B				Sum				Carry
	A3	A2	A1	A0	B3	B2	B1	B0	S3	S2	S1	S0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	1	0	0	1	0	0
0	0	0	1	0	0	0	1	0	0	1	0	0	0
0	0	0	1	1	0	0	1	1	0	1	1	0	0
0	0	1	0	0	0	1	0	0	1	0	0	0	0
0	0	1	0	1	0	1	0	1	1	0	1	0	0
0	0	1	1	0	0	1	1	0	1	1	0	0	0
0	0	1	1	1	0	1	1	1	1	1	1	0	0
0	1	0	0	0	1	0	0	0	0	0	0	0	1
0	1	0	0	1	1	0	0	1	0	0	1	0	1
0	1	0	1	0	1	0	1	0	0	1	0	0	1
0	1	0	1	1	1	0	1	1	0	1	1	0	1
0	1	1	0	0	1	1	0	0	1	0	0	0	1
0	1	1	0	1	1	1	0	1	1	0	1	0	1
0	1	1	1	0	1	1	1	0	1	1	0	0	1
0	1	1	1	1	1	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 7 - Truth Table for 4-Bit Adder

A 4 bit adder is a digital circuit that performs the addition of two 4-bit binary numbers. It is composed of four full adders, each of which adds two bits at a time. It can also be used to add two single-bit numbers. The 4-bit adder takes two 4-bit binary numbers (A3, A2, A1, A0 and B3, B2, B1, B0) as its inputs and produces a 4-bit binary number (C3, C2, C1, C0) as its internal outputs. Each full adder adds two bits and the output is the sum of those two bits, along with a carry bit, which is passed to the next full adder. The sum of the two 4-bit numbers is then outputted as a 4-bit binary number. This is shown in 'Figure 7'.

Even though there are inaccuracies in the explanation, the description of the function is correct, and agrees with the truth table.

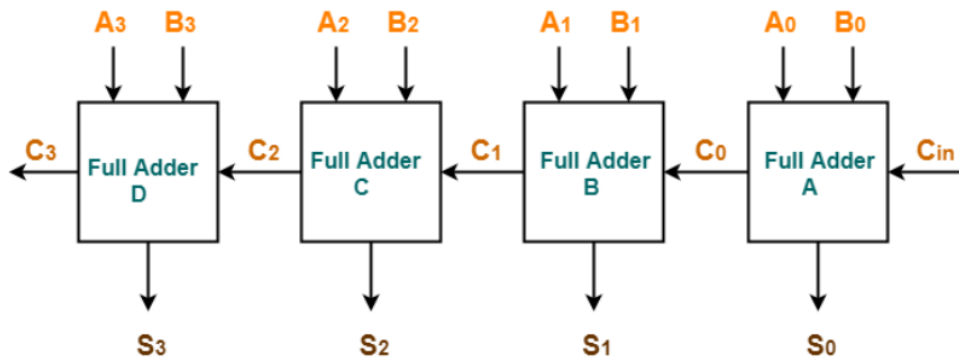


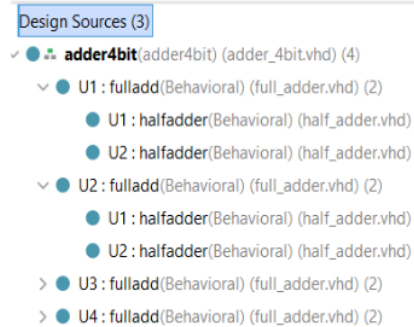
Figure 8 - Logic Diagram for 4-Bit Adder

Good block diagram that is crucial for the VHDL design. This demonstrates a good understanding of how to start the design procedure

1.4 Implementation

1.4.1 4 Bit Adder

A 4 Bit Adder was then created, using 4 full adders, and each of which is regarded as "1 Bit." The 4-bit adder, as shown in the design source hierarchy, is made up of 4 complete adders, each of which contains 2 half adders.



Good description with relevant hierarchical image

Figure 9 - Hierarchy for 4bit Adder

In order for each '1 bit' (full adder) to work a Vivado program was written (Figure 10).

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  entity adder4bit is
4  port(
5    cin : in STD_LOGIC;
6    a0 : in STD_LOGIC;
7    a1 : in STD_LOGIC;
8    a2 : in STD_LOGIC;
9    a3 : in STD_LOGIC;
10
11    b0 : in STD_LOGIC;
12    b1 : in STD_LOGIC;
13    b2 : in STD_LOGIC;
14    b3 : in STD_LOGIC;
15
16    cout : out STD_LOGIC;
17    s0 : out STD_LOGIC;
18    s1 : out STD_LOGIC;
19    s2 : out STD_LOGIC;
20    s3 : out STD_LOGIC
21  );
22  end adder4bit;
```

```
24  architecture adder4bit of adder4bit is
25  component fulladd
26  port (
27    a : in STD_LOGIC;
28    b : in STD_LOGIC;
29    cin : in STD_LOGIC;
30    cout : out STD_LOGIC;
31    s : out STD_LOGIC
32  );
33  end component;
34  signal c1 : STD_LOGIC;
35  signal c2 : STD_LOGIC;
36  signal c3 : STD_LOGIC;
37  begin
38  U1 : fulladd
39  port map(
40    a => a0, b => b0, cin => cin, cout => c1,
41    s => s0);
42  U2 : fulladd
43  port map(
44    a => a1, b => b1, cin => c1, cout => c2,
45    s => s1);
46  U3 : fulladd
47  port map(
48    a => a2, b => b2, cin => c2, cout => c3,
49    s => s2);
50  U4 : fulladd
51  port map(
52    a => a3, b => b3, cin => c3, cout => cout,
53    s => s3);
54  end adder4bit;
```

VHDL program is listed

Figure 10 – 4-bit adder code

This requires an explanation of entity and architecture

1.4.2 Combining 4-Bit adder and 4-Bit Magnitude Comparator

Initially, IP integration (Figure 11) was used to combine a 4-bit adder and also the 4-bit magnitude comparator. To join the two IP Integrated circuits into a single "IP Wrapper," a new project was started. The 4 bit adder will be fed by the [VHDL board](#) by several inputs when the circuit is put together.

A circuit was then constructed on Multisim (figure 12).

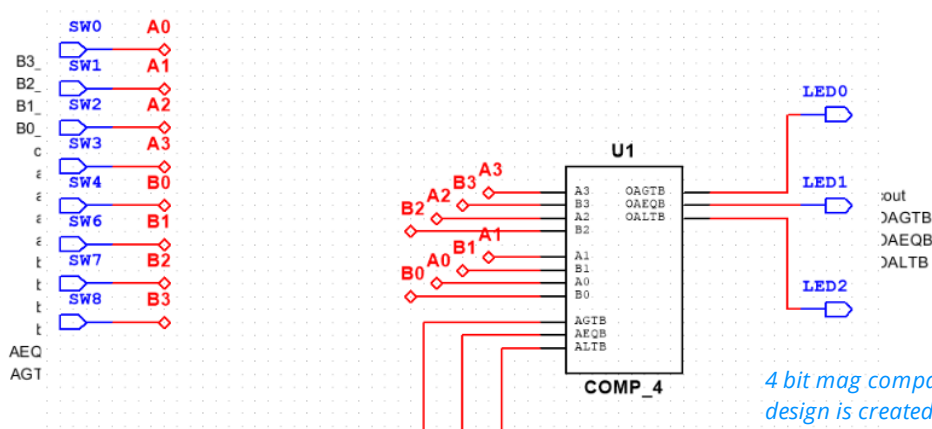


Figure 11 -IP for 4-Bit magnitude Comparator

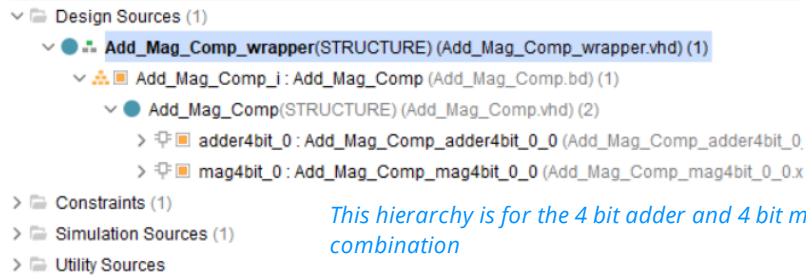
4 bit mag comparator Msim design is created. This should be explained that the VHDL is auto generated, and the IP created from the VHDL

Figure 12 – 4-bit Magnitude Comparator Multisim

1.4.3 8-Bit Magnitude Comparator Design

[Inaccurate heading here](#)

Figure 13 shows the hierarchy of the 8-bit magnitude comparator in Vivado.

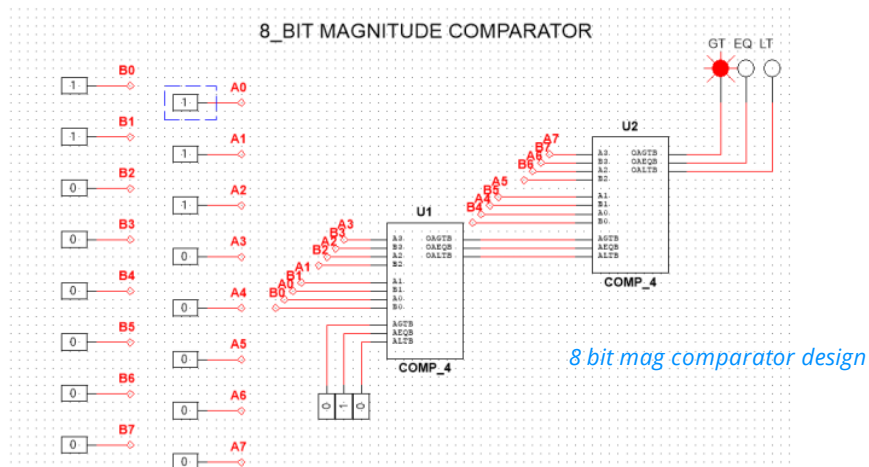


This hierarchy is for the 4 bit adder and 4 bit mag comparator combination

The combination IP design should be included here. This is what is downloaded onto the FPGA board

Figure 13 – Hierarchy for 8-Bit Magnitude Comparator

The circuit shown in Figure 14 was created in Multisim using the cascading of two 4-bit comparators. Binary inputs were fed to the comparators using toggle switches. LEDs linked to the second comparator output provide a visual display of the condition after three outputs (AGTB, AEQB, and ALTB) from comparator 1 were fed into comparator 2, which was used to compare the outputs with the inputs.



8 bit mag comparator design

Figure 14 – Multisim Circuit for 8-bit Magnitude Comparator

1.5 Testing and Analysis

1.5.1 4-Bit Adder and Comparator FPGA Testing

Test 1: A equal to B

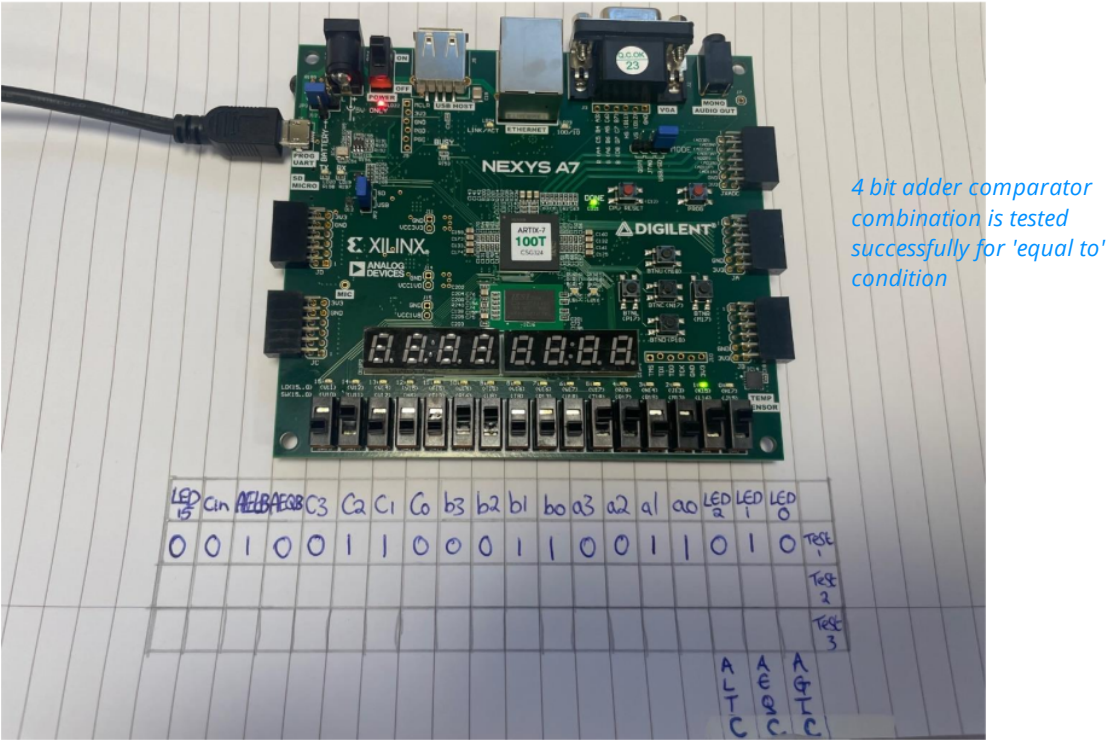


Figure 15 - A equal to B

Test 2: A Less than B



Figure 16 - A Less than B

4 bit adder combination successfully for all 'than' condition

LED ₅ Cin	A[3:0]	B[3:0]	C[3:0]	C ₂	C ₁	C ₀	b ₃	b ₂	b ₁	b ₀	a ₃	a ₂	a ₁	a ₀	LED ₂	LED ₁	LED ₀	
0	0	1	0	0	1	1	0	0	0	1	1	0	0	1	1	0	1	0
0	0	1	0	0	1	1	0	0	0	1	1	0	0	1	0	1	0	0
0	0	1	0	0	1	1	0	0	0	1	1	0	1	1	0	0	1	0

Test 1
Test 2
Test 3

A L T C
A E B C
A G T C

[illegible]

The 3 test condition have been tested successfully. There should be an explanation of the test procedure and a table to clarify results.

1.5.2 Test Bench for 4-Bit Adder

```
○ a0 <= '1';  
○ a1 <= '0';  
○ a2 <= '0';  
○ a3 <= '0';  
○ b0 <= '1';  
○ b1 <= '0';  
○ b2 <= '0';  
○ b3 <= '0';  
○ cin <= '0';  
○ wait for 10 ns;  
  
○ a0 <= '1';  
○ a1 <= '1';  
○ a2 <= '1';  
○ a3 <= '0';  
○ b0 <= '0';  
○ b1 <= '0';  
○ b2 <= '1';  
○ b3 <= '0';  
○ cin <= '0';  
○ wait for 10 ns;  
  
○ a0 <= '1';  
○ a1 <= '1';  
○ a2 <= '1';  
○ a3 <= '0';  
○ b0 <= '1';  
○ b1 <= '0';  
○ b2 <= '1';  
○ b3 <= '1';  
○ cin <= '0';  
○ wait for 10 ns;
```

Good range of values in the test data.

Figure 18 - 4-Bit Adder Code for testing

The range of test data values drives the adder over the 4 bit threshold, and causes the 'cout' carry bit to the high condition.

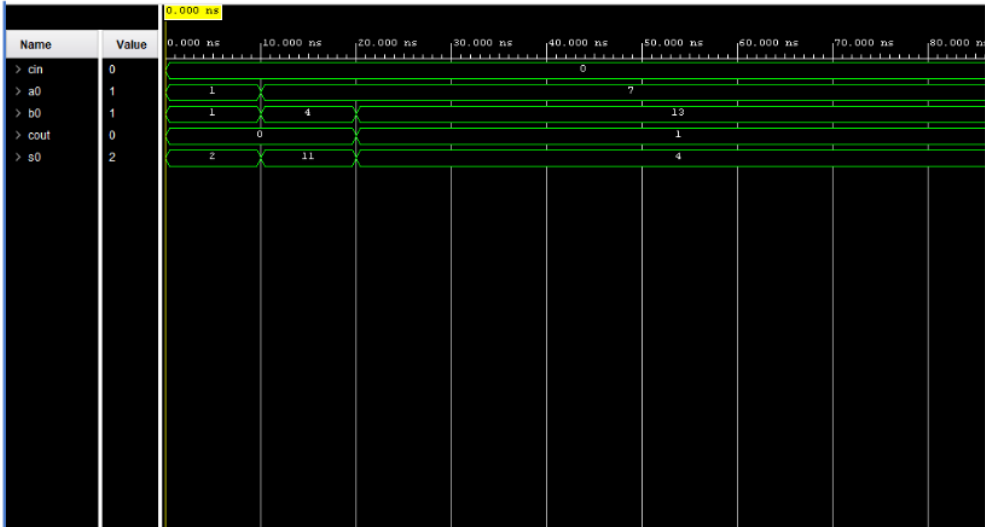


Figure 19 - Simulation for 4-Bit Adder

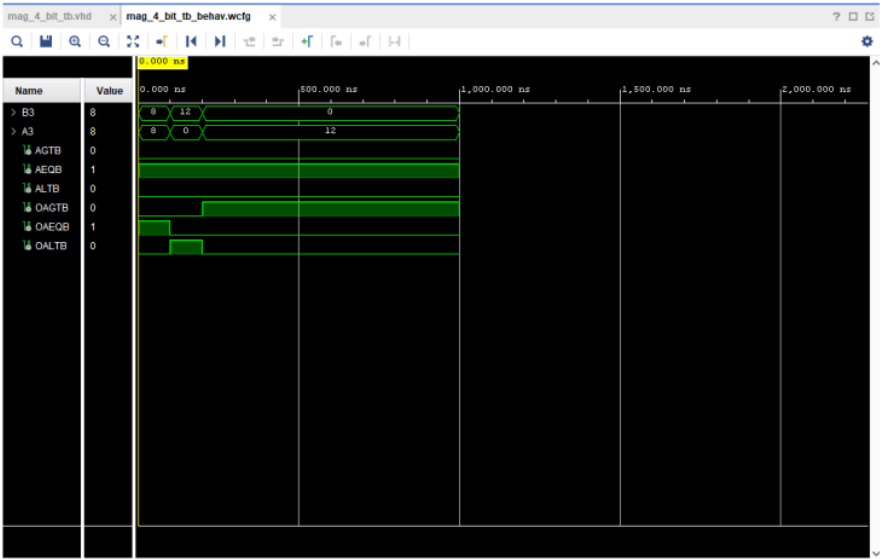
1.5.3 Test bench 4-bit mag comp

The code for the 4-Bit magnitude comparator was copied from generated VHDL from Multisim.

```
mag_4_bit_tb.vhd x mag_4_bit_tb_behav.wcfg x
W:/SGEE5007 VICS/4_bit_mag/4_bit_mag.srscs/sim_1/newmag_4_bit_tb.vhd
63
64 -- Put initialisation code here
65 B3 <= '1';
66 B2 <= '0';
67 B1 <= '0';
68 B0 <= '0';
69 A3 <= '1';
70 A2 <= '0';
71 A1 <= '0';
72 A0 <= '0';
73 AGTB <= '0';
74 AEQB <= '1';
75 ALTB <= '0';
76 wait for 100ns;
77 B3 <= '1';
78 B2 <= '1';
79 B1 <= '0';
80 B0 <= '0';
81 A3 <= '0';
82 A2 <= '0';
83 A1 <= '0';
84 A0 <= '0';
85 AGTB <= '0';
86 AEQB <= '1';
87 ALTB <= '0';
88 wait for 100ns;
89 B3 <= '0';
90 B2 <= '0';
91 B1 <= '0';
92 B0 <= '0';
93 A3 <= '1';
94 A2 <= '1';
95 A1 <= '0';
96 A0 <= '0';
```

Test data set to test all 3 conditions

Figure 20 - Code for 4-Bit Magnitude Comparator testing



Good result from test data

Figure 21 - Simulation for 4-Bit magnitude Comparator

Good use of the virtual bus test method

1.5.4 8-Bit Magnitude Comparator PLD

Below in figure 22 is a PLD circuit for the 8-Bit magnitude comparator that was created in multisim.

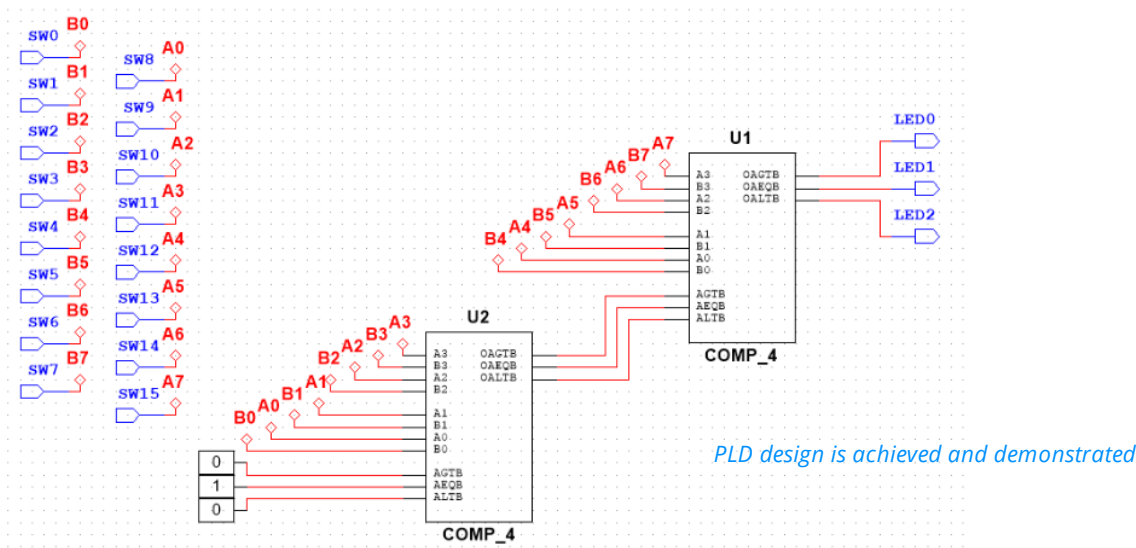
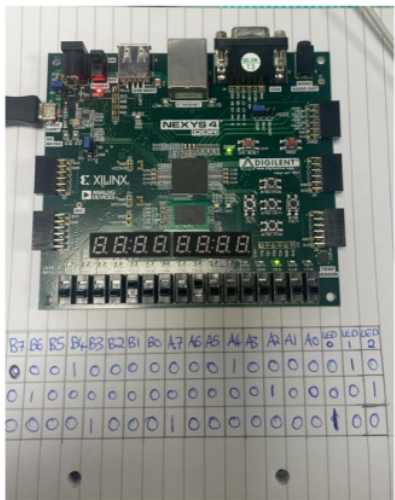


Figure 22 -PLD Circuit for 8-Bit Mag Comparator

1.5.3 Testing 8-Bit Magnitude Comparator -FPGA

Test Results



By looking at figure 23, this shows an example of a test carried out for 'A greater than B'. Two similar tests were then done for 'A equal to B' and 'A Less than B'. After these tests were completed, the results were then put into a table and as can be seen from the image each test passed.

There are 2 test conditions missing here!

Figure 23 - Test Results for 8-Bit mag Comp - FPGA

1.6 Evaluation

The first research that was conducted laid a solid foundation for this element's fundamental ideas. In order to successfully complete the design and build phases, it was crucial to have knowledge of digital magnitude comparators and digital design techniques utilising VHDL and IP integration. The foundation for element 2 was also supplied by additional investigation into ADCs, signal conditioning, and load cells. Along with this, understanding of truth tables and logic circuits was expanded, starting with the half-adder and moving on to more advanced circuits like the 4-bit magnitude comparator.

The first circuit created with VHDL was the half-adder, which was then simulated to confirm its functionality. Afterwards, a full adder was built using a duplicate of this architecture, which was tested and examined. Afterwards, four full adders were cascaded to produce a 4-bit adder, which was tested. The FPGA board was then used to download the programme and do physical testing.

Comparator

The circuit was demonstrated to be functional after testing. An 8-bit adder was made by cascading two 4-bit adders and tested and examined in Multisim. For use during functional testing on the FPGA board, this was later transformed into PLD mode. On the board, the outputs for AGTB, ALTB, and AEQB were allocated. These were tested, and they were found to be operational.

Developed using

In conclusion, the hierarchy's 8-bit magnitude comparator was successfully created, constructed, and tested, proving that all design, construction, and testing phases had been successful. A wide variety of fundamental concepts have been learned during this element, and element 2 will build on these concepts.

The evaluation covers significant points relevant for the next stage, and indicates an understanding of the tasks carried out.

2. Element 2

2.1 Introduction

This could be worded more effectively

The focus of Element 2 is on designing and implementing an electronic circuit to transform into digital 8-bit signals from analogue signals, which can also be summarised using the "less than," "equal to," and "greater than" comparison formats. The load cell circuit will be designed using the knowledge gained from the conducted research/theory in the first element. This signal will be compared to a standard 0V-5V signal that can be adjusted by a slider/potentiometer.

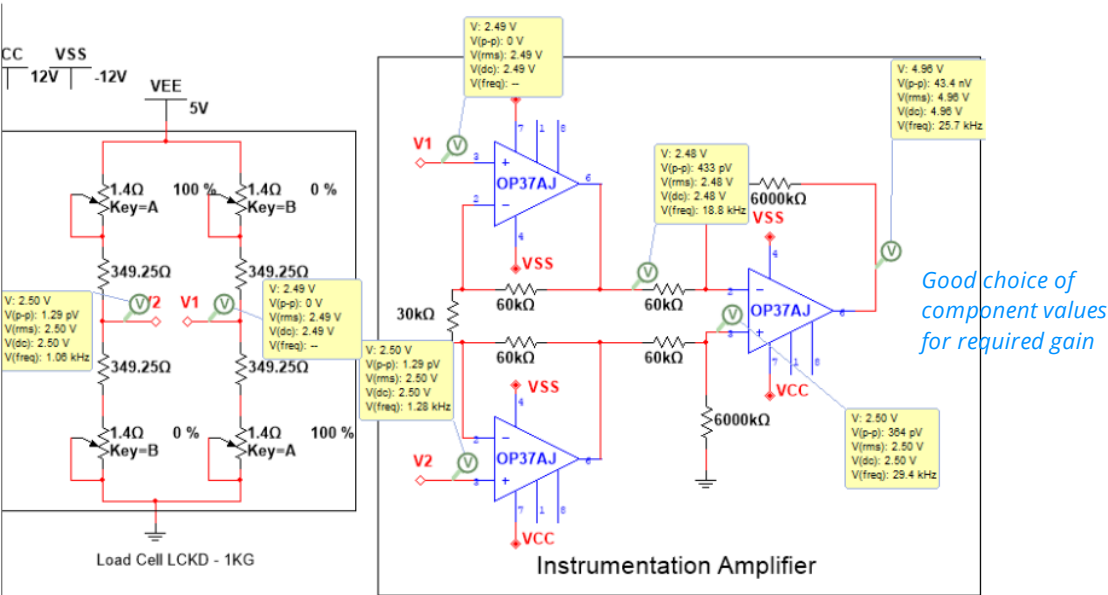
The intro only covers part of the requirements

2.2 Design Requirements

Signal conditioning must be used to produce the desired output of 0 to 5 volts. A unity gain amplifier is needed since a small load on the load cell will result in a low output voltage. The potential difference between V1 and V2 must also be measured using a differential op-amp; together, these two circuits will produce the instrument amplifier needed for the circuit to operate properly.

Explanation is good

2.3 Implementation



V: 4.96 V
V(p-p): 71.2 nV
V(rms): 4.96 V
V(dc): 4.96 V
V(freq): 25.0 kHz

Table 3 - Comparison Table for Load Cell Operation

Vrms=4.96V for load of 100% to 0%.

Instrument amplifier

Buffer gain = $(1 + (2 * 60k) / 30k) = 5$

Diff amp gain = $6000k / 60k = 100$

Gain required:

At 1KG and 5V excitation $V2 - V1 = 10mV$

At 1KG $V_{out} = 5V$

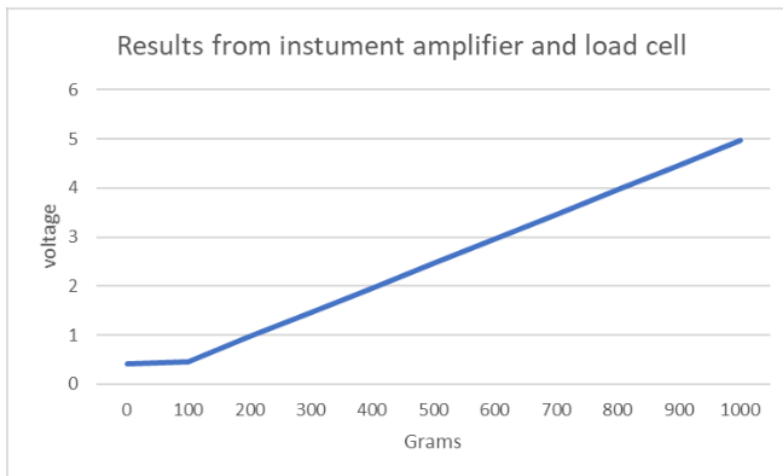
Gain $G = 5 / 0.01 = 500$

% Resistance	Grams	Voltage
50	0	0.41
55	100	0.46
60	200	0.96
65	300	1.46
70	400	1.96
75	500	2.46
80	600	2.96
85	700	3.46
90	800	3.96
95	900	4.46
100	1000	4.96

Calc proves the required gain from the component values.

Good table of results

Overall gain= diff amp gain * buffer gain= 5×100=500



The linear graph is good and illustrates the trend.

Figure 25 - Graph of Results from Instrument Amplifier and Load Cell

2.3.1 10-Bit ADC Prototype

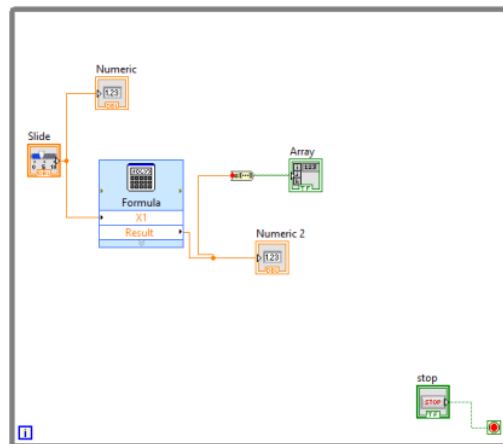


Figure 26 - Block Diagram for 10-Bit ADC

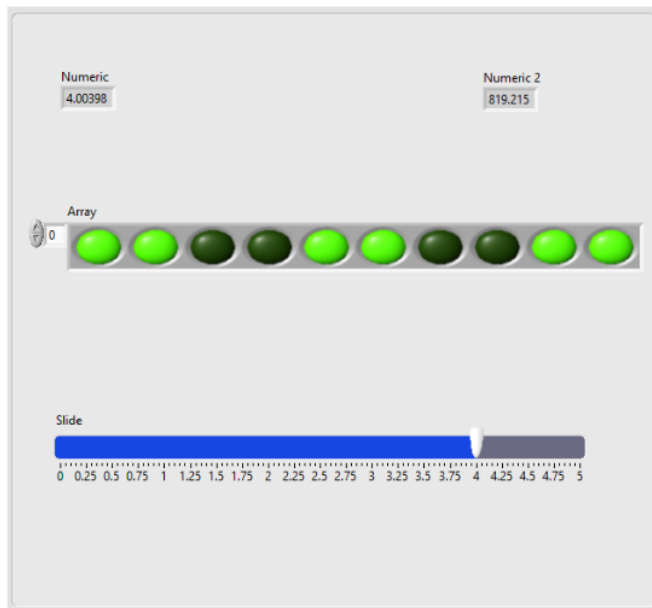
In figure 26, this shows a block diagram of a 10-Bit ADC that was constructed in LabView.



10 bit ADC prototype is implemented and tested

Figure 27 - Test 1 for 10-Bit ADC

Figure 27 shows an image of a test at 2.5v.



10 bit ADC second value tested

Figure 28 - Test 2 for 10-Bit ADC

Figure 28 shows an image of a test at 4v.

2.3.2 12-Bit ADC Prototype

After the 10-bit ADC a similar circuit and tests were carried out for the 12-Bit ADC.

The visual results from the 12-Bit ADC prototype can be shown in figures 29 & 30.



Figure 29 - Test 1 for 12-Bit ADC

Figure 29 shows the voltage at 1.5v.

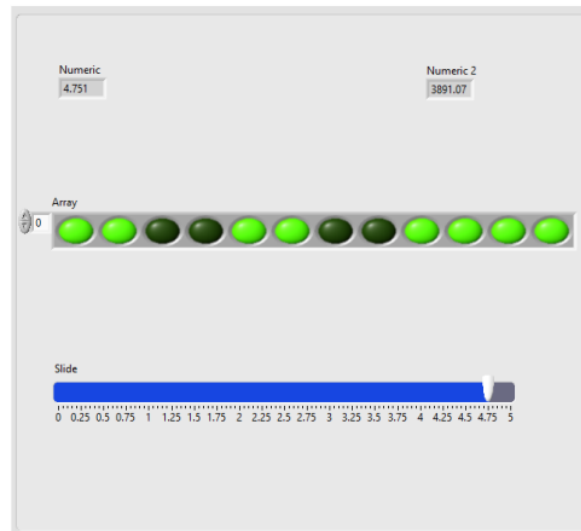


Figure 30 - Test 2 for 12-Bit ADC

Figure 30 shows the voltage at 4.75v.

Good tests for 12 bit ADC

The LabVIEW program development explanation is missing! This is a significant part of this element. The interfacing method and mapping is missing.

2.4 Testing and Analysis

2.4.1 Element 2 Test results

This should be identified with a heading

Set point at 700grms



Set point at 700grms is lower than the input value from the sensor and therefore the indicator is showing the condition for this state.

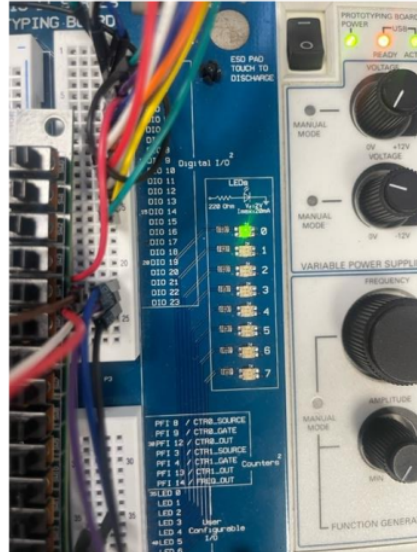


Figure 31 - 700grms Less than Input

Good evidence of test, with good result indicated in front panel

Set point at 700grms



Set point at 700grms is equal to the input value from the sensor and therefore the indicator is showing the condition for this state.

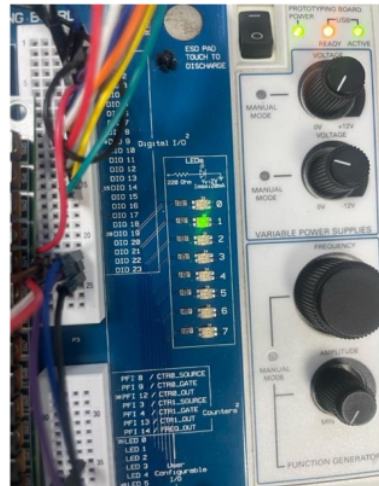


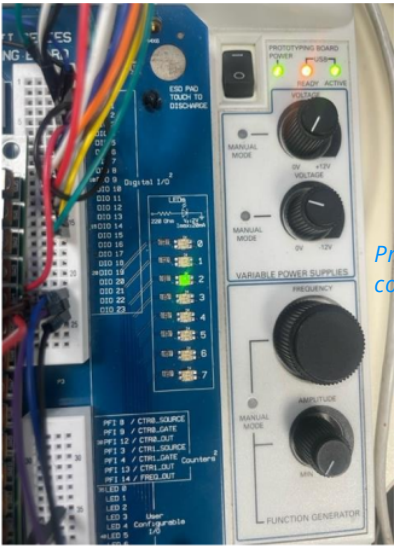
Figure 32 - 700grms Equal to the Input

Second condition test successful with good clear evidence

Set point at 700grms



Set point at 700grms is greater than the input value from the sensor and therefore the indicator is showing the condition for this state.



Practical DAQ with condition indicated

Third condition test result is successful with evidence

Figure 33 - 700grms Greater than the Input

Evaluation of results

Table 4 - Table of Results for 8-Bit Mag

Test No	State	LabVIEW Result	8 bit Mag Result
1	>	Sensor > SP	PASS
2	=	Sensor = SP	PASS
3	<	Sensor < SP	PASS

Good use of table to clarify test results

2.5 Evaluation

The tests indicated the interface with the 8 bit Magnitude Comparator was functioning in real time. The results taken from the front-end HMI prove the interface comms with the FPGA. Three states were tested ($\leq$, $=$, $\geq$) and all gave the required results.

This sets up the system in a reliable state for element 3 data logging and the test can be proven by using the same set point values. The virtual power supply has been set up in this case to simulate the sensor values. The values are input into the FPGA for comparison, and the 8 bit Magnitude comparator designed in element 1 automatically carried out the analysis of the 16 bits. The result is in the form of the three states.

The 16 bit referred to here has not been explained and therefore has no relevance

Evaluation needs to include the more significant parts of the element, such as the LabVIEW program structure, and the shift left 8 bits method for sending the 16 bits to the 8 bit mag comparator,

3. Element 3

3.1 Design Requirements

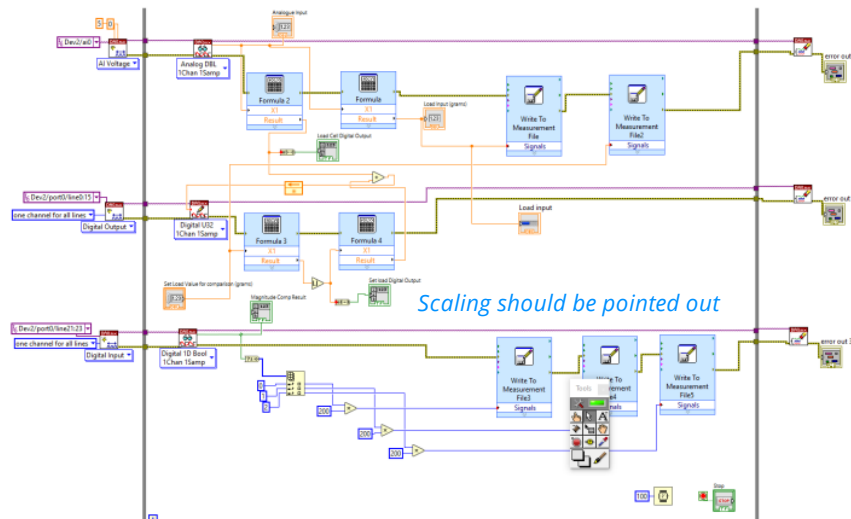


Figure 34 - Final Circuit (Element 3)

The program needs to be explained!

3.2 Implementation

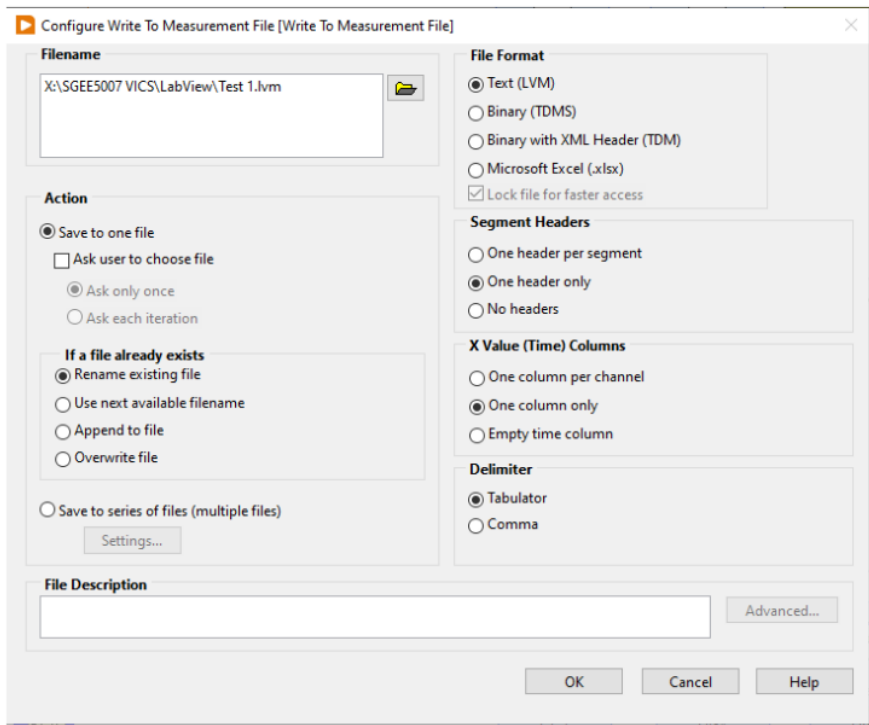


Figure 35 - Configure Write to Measurement File

For each of the flies in the circuit there are test files generated in Labview.

 Test 1.lvm	09/02/2023 17:55	LVM File	3 KB
 Test 2.lvm	09/02/2023 17:55	LVM File	2 KB
 Test 3.lvm	09/02/2023 17:55	LVM File	2 KB
 Test 4.lvm	09/02/2023 17:55	LVM File	2 KB
 Test 5.lvm	09/02/2023 17:55	LVM File	2 KB

Figure 36 - LabView Files Generated

write files are evident here, however, they are not explained in the program

3.3 Testing and Analysis

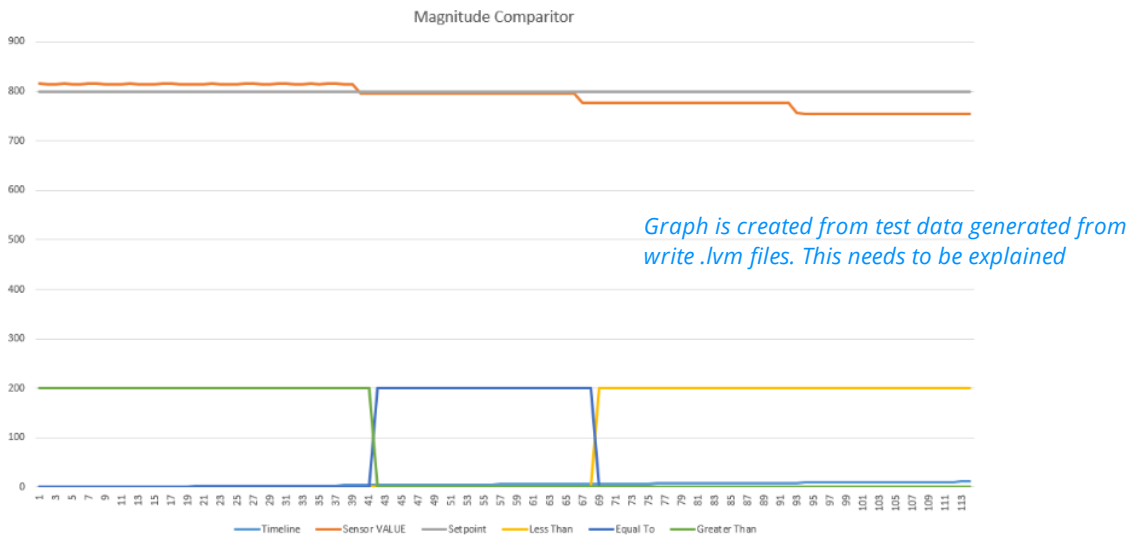


Figure 37 - Magnitude Comparitor Excel Graph

No evaluation for this element!

4. References

Bibliography

AspenCore, 2023. *Analogue to Digital Converter*. [Online]

Available at: <https://www.electronics-tutorials.ws/combination/analogue-to-digital-converter.html>

[Accessed 21 02 2023].

CORP., N. I., 2023. *NI-DAQmx*. [Online]

Available at: <https://www.ni.com/docs/en-US/bundle/ni-daqmx/page/daqhelp/nidaqoverview.html#:~:text=What%20is%20DAQmx%3F,CVI%2C%20to%20program%20your%20devices.>

[Accessed 08 02 2023].

Corp, N., 2023. *NI-DAQmx and LabVIEW Compatibility*. [Online]

Available at: <https://www.ni.com/en-gb/support/documentation/compatibility/16/ni-daqmx-and-labview-compatibility.html>

[Accessed 16 02 2023].

ElectronicsTutorials, 2023. *Digital Comparator*. [Online]

Available at: https://www.electronics-tutorials.ws/combination/comb_8.html

[Accessed 13 02 2023].

HBK, F. -, ©., 2023. *hbm.com*. [Online]

Available at: <https://www.hbm.com/en/7339/what-is-a-signal-conditioner/>

[Accessed 17 02 2023].

IEEE, 2023. *ieeexplore.ieee.org*. [Online]

Available at: <https://ieeexplore.ieee.org/document/1434582>

[Accessed 10 02 2023].

inc., O. E., 2023. *Introduction to Load Cells*. [Online]

Available at: <https://www.omega.co.uk/prodinfo/load-cells.html>

[Accessed 20 02 2023].

Land, N., 2022. *Nandland.com*. [Online].

Land, N., 2022. *Nandland.com*. [Online]

Available at: <https://nandland.com/introduction-to-vhdl-for-beginners-with-code-examples/>

[Accessed 15 02 2023].

Pandey, H., 2022. *Magnitude Comparator in Digital Logic*. [Online]

Available at: <https://www.geeksforgeeks.org/magnitude-comparator-in-digital-logic/>

[Accessed 09 02 2023].

Systems, J. T., 2019. *WHAT IS INSTRUMENTATION?*. [Online]

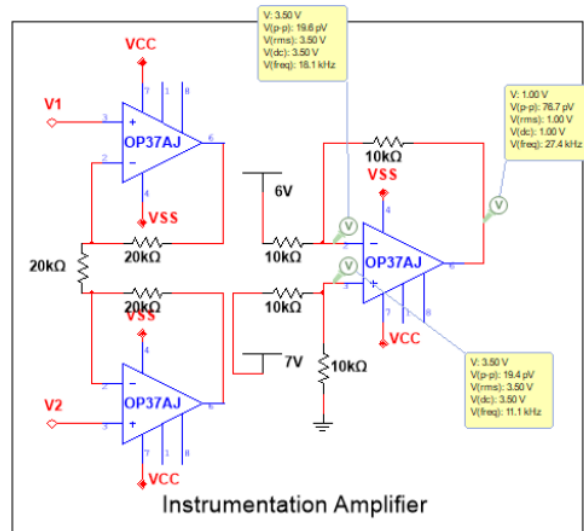
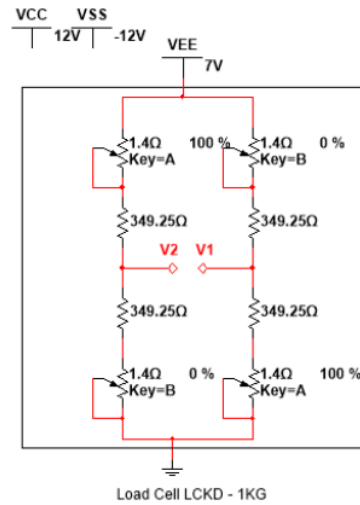
Available at: <https://jmttest.com/what-is-instrumentation/>

[Accessed 18 02 2023].

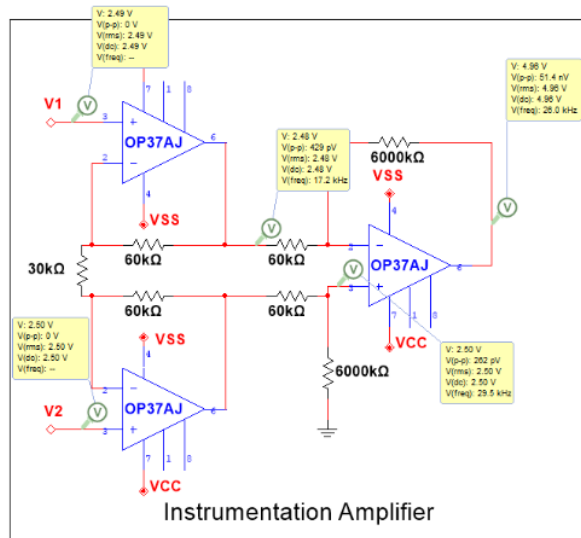
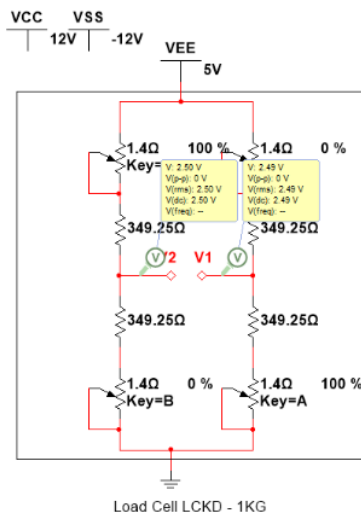
Appropriate number of references used

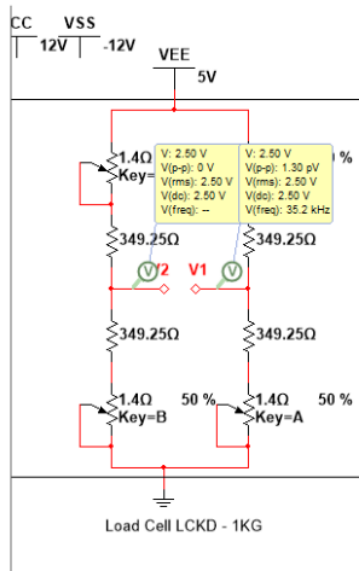
5. Appendix

Unity gain amplifier



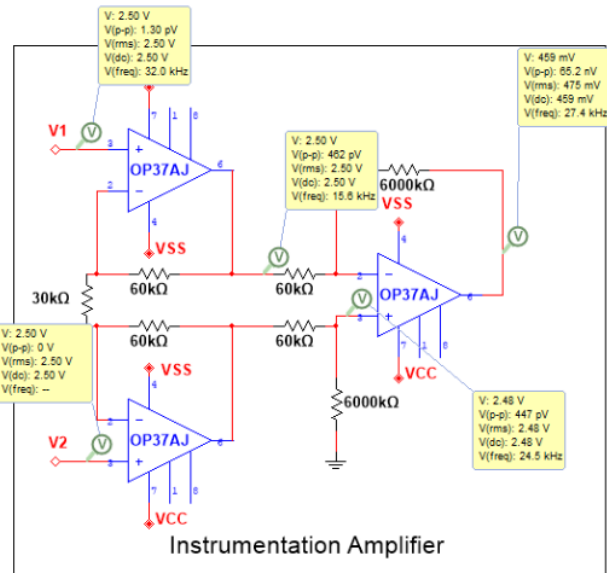
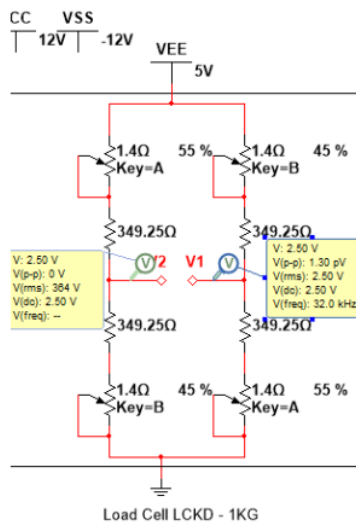
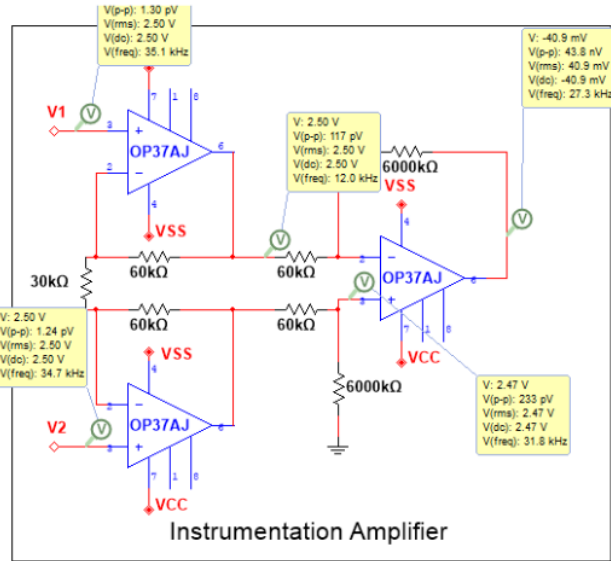
Load cell with instrument amplifier.





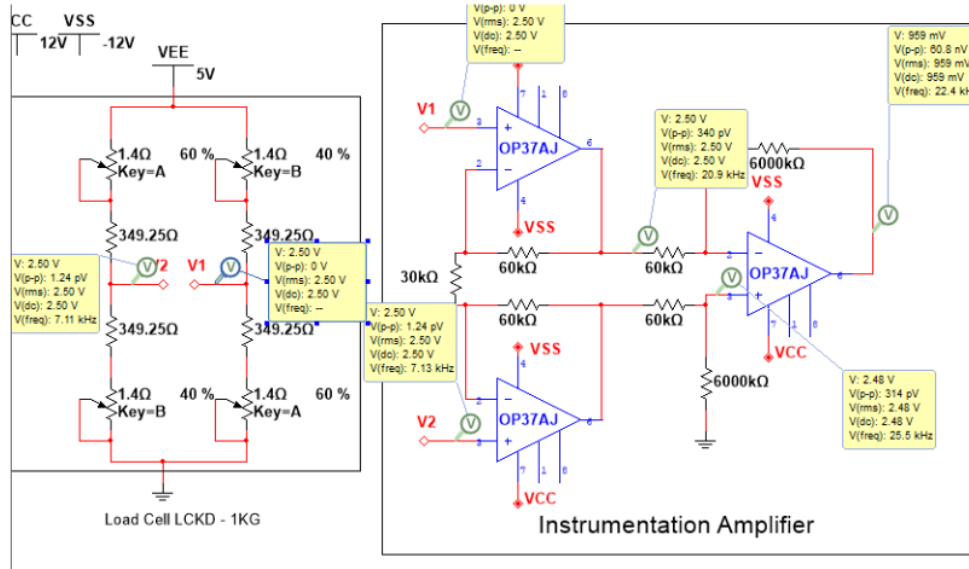
V: -40.9 mV
V(p-p): 21.4 nV
V(rms): 40.9 mV
V(do): -40.9 mV
V(freq): 27.3 kHz

Vrms=0.041V for load of 50% to 50%.



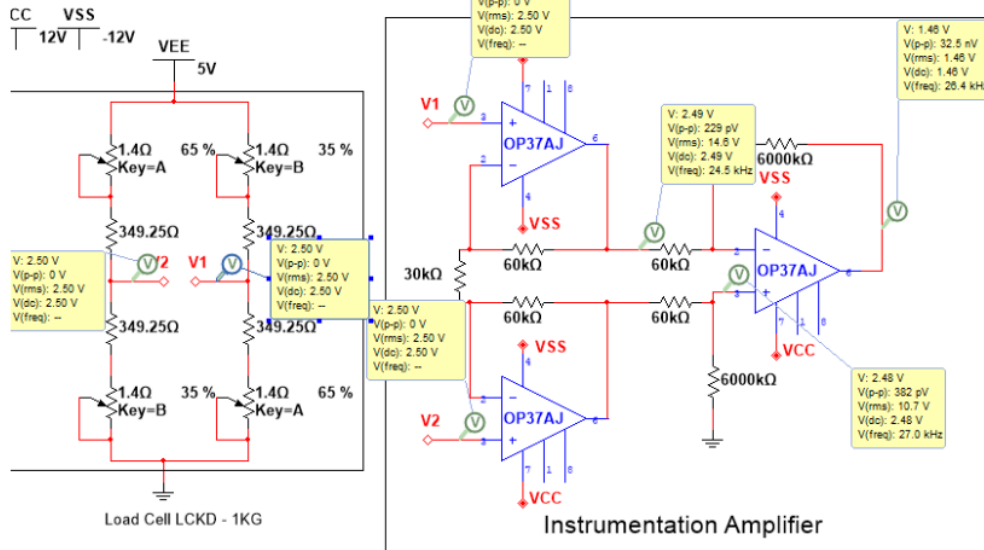
V: 459 mV
V(p-p): 77.8 nV
V(rms): 483 mV
V(dc): 459 mV
V(freq): 27.0 kHz

Vrms=0.459V for load of 55% to 45%.



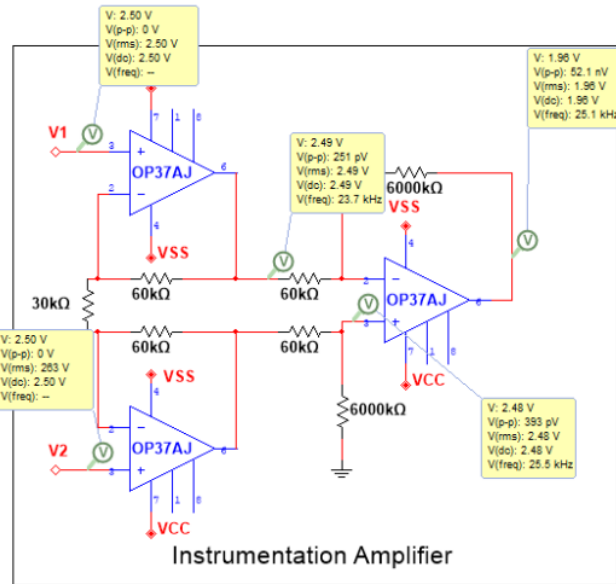
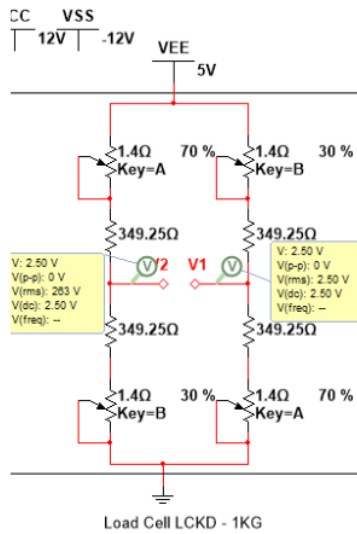
V: 959 mV
V(p-p): 54.7 nV
V(rms): 959 mV
V(dc): 959 mV
V(freq): 22.2 kHz

Vrms=0.959V for load of 60% to 40%.



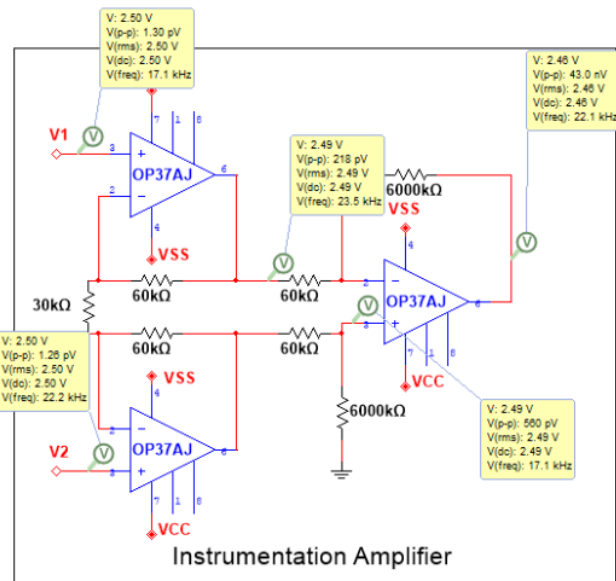
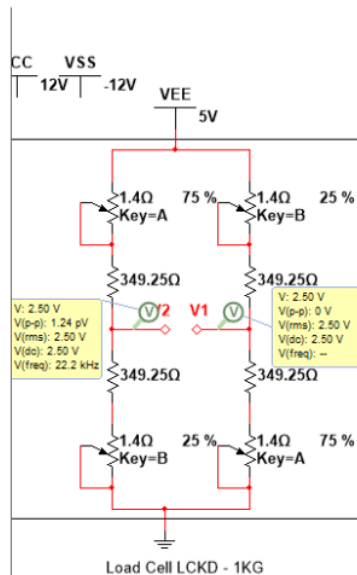
V: 1.46 V
V(p-p): 57.4 nV
V(rms): 1.46 V
V(dc): 1.46 V
V(freq): 25.9 kHz

Vrms=1.46V for load of 65% to 35%.



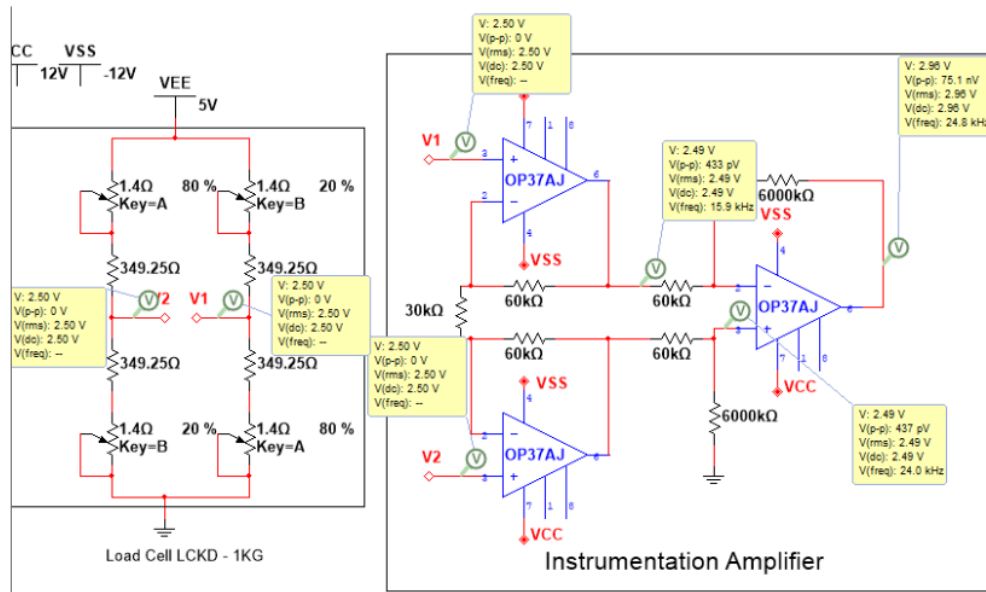
V: 1.96 V
V(p-p): 82.2 nV
V(rms): 1.96 V
V(dc): 1.96 V
V(freq): 25.0 kHz

Vrms=1.96V for load of 70% to 30%.



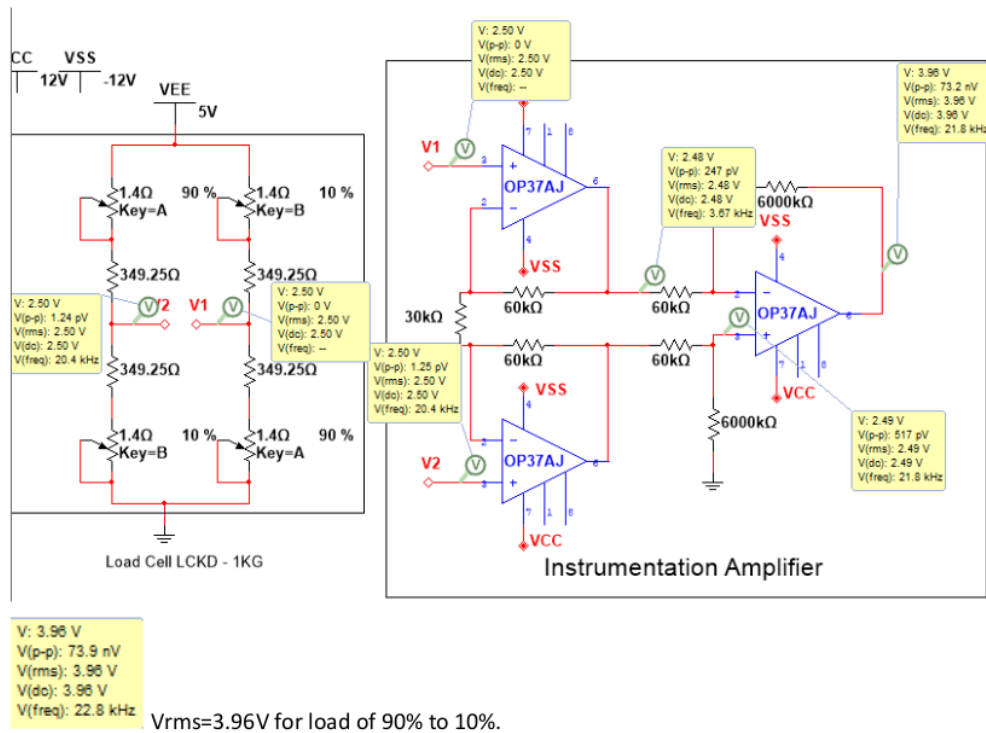
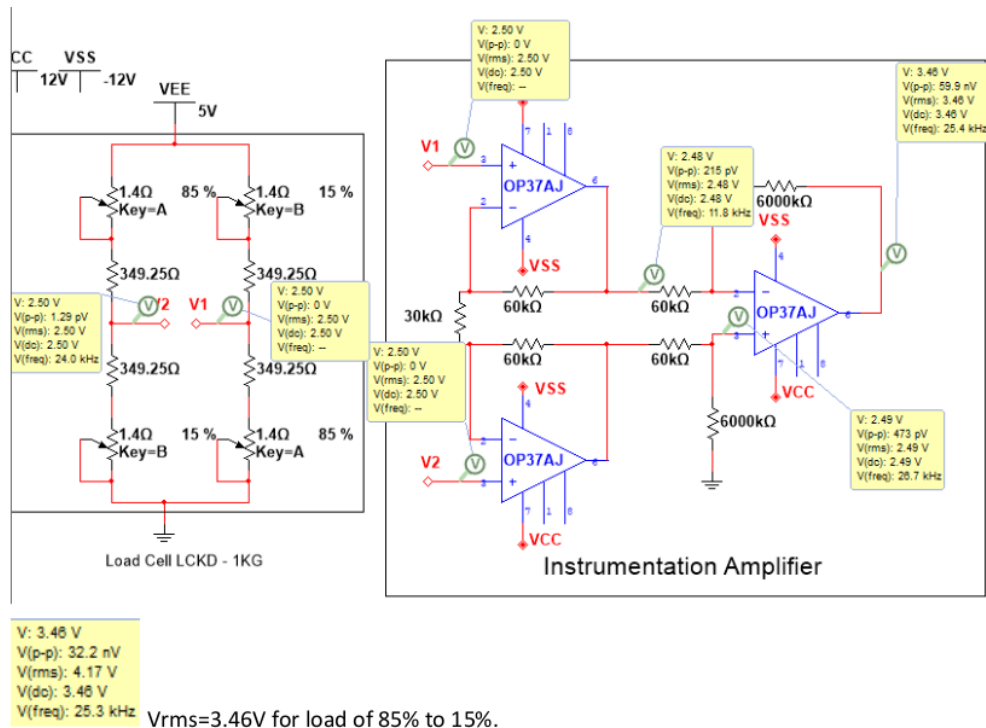
V: 2.46 V
V(p-p): 62.8 nV
V(rms): 3.33 V
V(dc): 2.46 V
V(freq): 22.0 kHz

Vrms=2.46V for load of 75% to 25%.



V: 2.96 V
V(p-p): 79.7 nV
V(rms): 2.96 V
V(dc): 2.96 V
V(freq): 25.1 kHz

Vrms=2.96V for load of 80% to 20%.





Files for Element 3 block diagram

Configure Write To Measurement File [Write To Measurement File]

Filename
X:\SGEE5007 VICS\LabView\Test 1.lvm

Action
☒ Save to one file
☐ Ask user to choose file
☒ Ask only once
☐ Ask each iteration
If a file already exists
☒ Rename existing file
☐ Use next available filename
☐ Append to file
☐ Overwrite file
☐ Save to series of files (multiple files)
Settings...

File Format
☒ Text (LVM)
☐ Binary (TDMS)
☐ Binary with XML Header (TDM)
☐ Microsoft Excel (.xlsx)
☒ Lock file for faster access

Segment Headers
☐ One header per segment
☒ One header only
☐ No headers

X Value (Time) Columns
☐ One column per channel
☒ One column only
☐ Empty time column

Delimiter
☒ Tabulator
☐ Comma

File Description
Advanced...

OK Cancel Help

X:\SGEE5007 VICS\LabView\Test 1.lvm - Notepad++

	X_Value	Comment
22	0.000000	815.140546
23	0.101717	814.915104
24	0.202644	815.043928
25	0.302712	815.301576
26	0.402764	814.625250
27	0.502082	814.689662
28	0.601018	815.172752
29	0.702156	815.462606
30	0.800751	814.915104
31	0.901020	814.947310
32	1.001775	814.754074
33	1.104605	815.108340
34	1.201608	814.979516
35	1.301946	814.947310
36	1.403206	814.979516
37	1.502801	815.301576
38	1.603798	815.204958
39	1.705087	814.850692
40	1.803978	815.043928
41	1.904208	815.043928
42	2.005608	814.818486
43	2.105767	815.140546
44	2.203896	814.850692
45	2.304416	814.882898
46	2.405061	814.882898
47	2.505396	815.108340
48	2.604439	815.172752
49	2.704058	814.915104
50	2.804032	814.721868
51	2.905923	815.172752
52	3.005660	815.237164
53	3.106580	814.721868

Normal text file length: 2,841 lines: 137 Ln: 22 Col: 28 Pos: 432 Windows (CR LF) UTF-8 INS

Test 1 Sample

Configure Write To Measurement File [Write To Measurement File2]

Filename
X:\SGEE5007 VICS\LabView\Test 2.lvm

Action
☒ Save to one file
☐ Ask user to choose file
☒ Ask only once
☐ Ask each iteration

If a file already exists
☒ Rename existing file
☐ Use next available filename
☐ Append to file
☐ Overwrite file

☐ Save to series of files (multiple files)
Settings...

File Format
☒ Text (LVM)
☐ Binary (TDMS)
☐ Binary with XML Header (TDM)
☐ Microsoft Excel (.xlsx)
☒ Lock file for faster access

Segment Headers
☐ One header per segment
☒ One header only
☐ No headers

X Value (Time) Columns
☐ One column per channel
☐ One column only
☒ Empty time column

Delimiter
☒ Tabulator
☐ Comma

File Description
Advanced...

OK Cancel Help

X:\SGEE5007 VICS\LabView\Test 2.lvm - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

Test 1.lvm Test 2.lvm Test 3.lvm Test 4.lvm Test 5.lvm

	X_Value	Comment
22		
23	800.000000	
24	800.000000	
25	800.000000	
26	800.000000	
27	800.000000	
28	800.000000	
29	800.000000	
30	800.000000	
31	800.000000	
32	800.000000	
33	800.000000	
34	800.000000	
35	800.000000	
36	800.000000	
37	800.000000	
38	800.000000	
39	800.000000	
40	800.000000	
41	800.000000	
42	800.000000	
43	800.000000	
44	800.000000	
45	800.000000	
46	800.000000	
47	800.000000	
48	800.000000	
49	800.000000	
50	800.000000	
51	800.000000	
52	800.000000	
53	800.000000	
54	800.000000	

Normal text file length: 1,918 lines: 137 Ln: 1 Col: 1 Pos: 1 Windows (CR LF) UTF-8 INS

Test 2 Sample

Configure Write To Measurement File [Write To Measurement File3]

Filename
X:\SGEE5007 VICS\LabView\Test 3.lvm

Action
☒ Save to one file
☐ Ask user to choose file
☒ Ask only once
☐ Ask each iteration

If a file already exists
☒ Rename existing file
☐ Use next available filename
☐ Append to file
☐ Overwrite file

☐ Save to series of files (multiple files)
Settings...

File Format
☒ Text (LVM)
☐ Binary (TDMS)
☐ Binary with XML Header (TDM)
☐ Microsoft Excel (.xlsx)
☒ Lock file for faster access

Segment Headers
☐ One header per segment
☒ One header only
☐ No headers

X Value (Time) Columns
☐ One column per channel
☐ One column only
☒ Empty time column

Delimiter
☒ Tabulator
☐ Comma

File Description
Advanced...

OK Cancel Help

X:\SGEE5007 VICS\LabView\Test 3.lvm - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

Test 3.lvm Test 4.lvm Test 5.lvm

	X_Value	Untitled	Comment
22	0.000000		
23	0.000000		
24	0.000000		
25	0.000000		
26	0.000000		
27	0.000000		
28	0.000000		
29	0.000000		
30	0.000000		
31	0.000000		
32	0.000000		
33	0.000000		
34	0.000000		
35	0.000000		
36	0.000000		
37	0.000000		
38	0.000000		
39	0.000000		
40	0.000000		
41	0.000000		
42	0.000000		
43	0.000000		
44	0.000000		
45	0.000000		
46	0.000000		
47	0.000000		
48	0.000000		
49	0.000000		
50	0.000000		
51	0.000000		
52	0.000000		
53	0.000000		
54	0.000000		

Normal text file length: 1,782 lines: 137 Ln: 1 Col: 1 Pos: 1 Windows (CR LF) UTF-8 INS

Test 3 Sample

Configure Write To Measurement File [Write To Measurement File4]

Filename
X:\SGEE5007 VICS\LabView\Test 4.lvm

Action
☒ Save to one file
☐ Ask user to choose file
☒ Ask only once
☐ Ask each iteration

If a file already exists
☒ Rename existing file
☐ Use next available filename
☐ Append to file
☐ Overwrite file

☐ Save to series of files (multiple files)
Settings...

File Format
☒ Text (LVM)
☐ Binary (TDMS)
☐ Binary with XML Header (TDM)
☐ Microsoft Excel (.xlsx)
☒ Lock file for faster access

Segment Headers
☐ One header per segment
☒ One header only
☐ No headers

X Value (Time) Columns
☐ One column per channel
☐ One column only
☒ Empty time column

Delimiter
☒ Tabulator
☐ Comma

File Description
Advanced...

OK Cancel Help

X:\SGEE5007 VICS\LabView\Test 4.lvm - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

Test 4.lvm Test 5.lvm

```
22 X_Value Untitled Comment
23 0.000000
24 0.000000
25 0.000000
26 0.000000
27 0.000000
28 0.000000
29 0.000000
30 0.000000
31 0.000000
32 0.000000
33 0.000000
34 0.000000
35 0.000000
36 0.000000
37 0.000000
38 0.000000
39 0.000000
40 0.000000
41 0.000000
42 0.000000
43 0.000000
44 0.000000
45 0.000000
46 0.000000
47 0.000000
48 0.000000
49 0.000000
50 0.000000
51 0.000000
52 0.000000
53 0.000000
54 0.000000
```

Normal text file length: 1,744 lines: 137 Ln: 1 Col: 1 Pos: 1 Windows (CR LF) UTF-8 INS

Test 4 Sample

Configure Write To Measurement File [Write To Measurement File5]

Filename
X:\SGEE5007 VICS\LabView\Test 5.lvm

Action
☒ Save to one file
☐ Ask user to choose file
☒ Ask only once
☐ Ask each iteration

If a file already exists
☒ Rename existing file
☐ Use next available filename
☐ Append to file
☐ Overwrite file

☐ Save to series of files (multiple files)
Settings...

File Format
☒ Text (LVM)
☐ Binary (TDMS)
☐ Binary with XML Header (TDM)
☐ Microsoft Excel (xlsx)
☒ Lock file for faster access

Segment Headers
☐ One header per segment
☒ One header only
☐ No headers

X Value (Time) Columns
☐ One column per channel
☐ One column only
☒ Empty time column

Delimiter
☒ Tabulator
☐ Comma

File Description
Advanced...

OK Cancel Help

X:\SGEE5007 VICS\LabView\Test 5.lvm - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

Test 5.lvm [3]

	X_Value	Untitled	Comment
22	200.000000		
23	200.000000		
24	200.000000		
25	200.000000		
26	200.000000		
27	200.000000		
28	200.000000		
29	200.000000		
30	200.000000		
31	200.000000		
32	200.000000		
33	200.000000		
34	200.000000		
35	200.000000		
36	200.000000		
37	200.000000		
38	200.000000		
39	200.000000		
40	200.000000		
41	200.000000		
42	200.000000		
43	200.000000		
44	200.000000		
45	200.000000		
46	200.000000		
47	200.000000		
48	200.000000		
49	200.000000		
50	200.000000		
51	200.000000		
52	200.000000		
53	200.000000		
54	200.000000		

Normal text file length: 1,772 lines: 137 Ln: 1 Col: 1 Pos: 1 Windows (CR LF) UTF-8 INS

Test 5 Sample

Good set of test data from .lvm text files

0	815.140546	800	0	0	200
0.101717	814.915104	800	0	0	200
0.202644	815.043928	800	0	0	200
0.302712	815.301576	800	0	0	200
0.402764	814.62525	800	0	0	200
0.502082	814.689662	800	0	0	200
0.601018	815.172752	800	0	0	200
0.702156	815.462606	800	0	0	200
0.800751	814.915104	800	0	0	200
0.90102	814.94731	800	0	0	200
1.001775	814.754074	800	0	0	200
1.104605	815.10834	800	0	0	200
1.201608	814.979516	800	0	0	200
1.301946	814.94731	800	0	0	200
1.403206	814.979516	800	0	0	200
1.502801	815.301576	800	0	0	200
1.603798	815.204958	800	0	0	200
1.705087	814.850692	800	0	0	200
1.803978	815.043928	800	0	0	200
1.904208	815.043928	800	0	0	200
2.005608	814.818486	800	0	0	200
2.105767	815.140546	800	0	0	200
2.203856	814.850692	800	0	0	200
2.304416	814.882898	800	0	0	200
2.405061	814.882898	800	0	0	200
2.505396	815.10834	800	0	0	200
2.604439	815.172752	800	0	0	200
2.704058	814.915104	800	0	0	200
2.804032	814.721868	800	0	0	200
2.905923	815.172752	800	0	0	200
3.00566	815.237164	800	0	0	200
3.106889	814.721868	800	0	0	200
3.205563	814.754074	800	0	0	200
3.305051	815.204958	800	0	0	200
3.406976	815.043928	800	0	0	200
3.507746	815.333782	800	0	0	200
3.607144	815.204958	800	0	0	200

FINAL GRADE

GENERAL COMMENTS

Instructor

43/100

PAGE 1

PAGE 2

PAGE 3

Text Comment. Good comprehensive table of contents

PAGE 4

PAGE 5

Text Comment. Good detailed list of figures

Text Comment. Good use of tables

PAGE 6

Text Comment. Introduction demonstrates an understanding of requirements

PAGE 7

Text Comment. Relevant example

Text Comment. Good significant points

PAGE 8

Text Comment. Good relevant example

Text Comment. Good explanation of IP block design with recognition of design possibilities

Text Comment. There are good findings with this topic area, and supported with relevant diagram and ref sources.

PAGE 9

Text Comment. A good general description of LabVIEW and the DAQ practical application. The image is a relevant example, however, the block functions of the DAQ should have been identified and discussed

PAGE 10

Text Comment. There is no restriction to 2 digital inputs

Text Comment. The example shows the 3 possible conditions. The explanation should identify the 3 conditions.

Text Comment. A broad application of digital comparators is explained and supported with example and ref source. This indicates a good study of the topic area.

PAGE 11

Text Comment. A definition of ADC's is included, however, the different methods are not covered

Text Comment. A general description and application example with relevant source, that needs a supporting image with identifiable parts.

PAGE 12

Text Comment. Good points included with relevant source

Text Comment. There is repetition here

Text Comment. This is a repetition of above

Text Comment. The content includes repetition, and would have benefitted from a block diagram to illustrate a process with signal conditioning

PAGE 13

Text Comment. This demonstrates an understanding of the hierarchy for the VHDL

Text Comment. Good example

Text Comment. Image should include a ref source

PAGE 14

Text Comment. Good explanation with example

Text Comment. The image should include a ref source

PAGE 15

Text Comment. This part is not accurate

Text Comment. Even though there are inaccuracies in the explanation, the description of the function is correct, and agrees with the truth table.

Text Comment. Good block diagram that is crucial for the VHDL design. This demonstrates a good understanding of how to start the design procedure

PAGE 16

Text Comment. Good description with relevant hierarchical image

Text Comment. VHDL program is listed

Text Comment. This requires an explanation of entity and architecture

PAGE 17

Text Comment. FPGA

Text Comment. 4 bit mag comparator Msim design is created. This should be explained that the VHDL is auto generated, and the IP created from the VHDL

PAGE 18

Text Comment. Inaccurate heading here

Text Comment. This hierarchy is for the 4 bit adder and 4 bit mag comparator combination

Text Comment. The combination IP design should be included here. This is what is downloaded onto the FPGA board

Text Comment. 8 bit mag comparator design

PAGE 19

Text Comment. 4 bit adder comparator combination is tested successfully for 'equal to' condition

PAGE 20

Text Comment. 4 bit adder comparator combination is tested successfully for 'less than' condition

PAGE 21

Text Comment. 4 bit adder comparator combination is tested successfully for 'greater than' condition

Text Comment. The 3 test condition have been tested successfully. There should be an explanation of the test procedure and a table to clarify results.

PAGE 22

Text Comment. Good range of values in the test data.

Text Comment. The range of test data values drives the adder over the 4 bit threshold, and causes the 'cout' carry bit to the high condition.

PAGE 23

Text Comment. Test data set to test all 3 conditions

Text Comment. Good result from test data

Text Comment. Good use of the virtual bus test method

PAGE 24

Text Comment. PLD design is achieved and demonstrated

Text Comment. There are 2 test conditions missing here!

Text Comment. Comparator

Text Comment. Developed using

Text Comment. The evaluation covers significant points relevant for the next stage, and indicates an understanding of the tasks carried out.

Text Comment. This could be worded more effectively

Text Comment. The intro only covers part of the requirements

Text Comment. Explanation is good

Text Comment. Good choice of component values for required gain

Text Comment. Calc proves the required gain from the component values.

Text Comment. Good table of results

Text Comment. The linear graph is good and illustrates the trend.

Text Comment. 10 bit ADC prototype is implemented and tested

Text Comment. 10 bit ADC second value tested

Text Comment. Good tests for 12 bit ADC

Text Comment. The LabVIEW program development explanation is missing! This is a significant part of this element. The interfacing method and mapping is missing.

Text Comment. This should be identified with a heading

Text Comment. Good evidence of test, with good result indicated in front panel

Text Comment. Second condition test successful with good clear evidence

PAGE 32

Text Comment. Practical DAQ with condition indicated

Text Comment. Third condition test result is successful with evidence

Text Comment. Good use of table to clarify test results

PAGE 33

Text Comment. The 16 bit referred to here has not been explained and therefore has no relevance

Text Comment. Evaluation needs to include the more significant parts of the element, such as the LabVIEW program structure, and the shift left 8 bits method for sending the 16 bits to the 8 bit mag comparator,

PAGE 34

Text Comment. Scaling should be pointed out

Text Comment. The program needs to be explained!

PAGE 35

Text Comment. write files are evident here, however, they are not explained in the program

PAGE 36

Text Comment. Graph is created from test data generated from write .lvm files. This needs to be explained

Text Comment. No evaluation for this element!

PAGE 37

PAGE 38

Text Comment. Appropriate number of references used

PAGE 39

PAGE 40

PAGE 41

PAGE 42

PAGE 43

PAGE 44

PAGE 45

Text Comment. Full set of test results included

PAGE 46

PAGE 47

PAGE 48

PAGE 49

PAGE 50

Text Comment. Good set of test data from .lvm text files

PAGE 51

GRADING FORM: CSG-RUBIC

JACK ROGERS

0

RESEARCH-INFORMED LITERATURE



Research includes good points and indicate a good level of study around the topic areas. A good number of ref sources have been used and the content is evenly spread across the topic areas. There is repetition in some areas and content that would have benefitted from example diagrams.

5/10

KNOWLEDGE AND UNDERSTANDING



Element 1 includes significant parts of the requirements. The VHDL program for the 4 bit adder is present, however, this not explained. The auto generated VHDL for 8 bit mag comparator is not explained. There are IP designs missing, and test procedures are not explained. 4/10 Element 2 instrument amplifier has been implemented correctly and has correct compoenet values. 8 bit ADC prototype is missing and the LabVIEW program is not included that represents a significant part of the element. 4/10 Element 3 includes the LabVIEW program with no explanations. There is evidence of the practical configuration with the data logging. 4/10

12/30

ANALYSIS



Element 1 test results are only part achieved. Two of the 8 bit comparator tests are missing. However the Vivado test results are included. There is a reasonable evaluation. 3/5 Element 2 test results have 8 bit ADC missing. Tests have the required range of values. Main tests with 8 bit comparator are included. The evaluation does not include enough relevant points. 5/10 Element 3 test results

12/25

with graph are evident, no explanations are included and no evaluation given. 4/10

PRACTICAL APPLICATION AND DEP.



Element 1 design and build has been carried out and there is evidence included. There are good explanations in some parts. However, the practical testing for the 4 bit adder and comparator combination has no explanation. There are practical tests for the 8 bit comparator missing. 2.5/5 Element 2 interface mapping and LabVIEW program with explanation is missing. The 10 and 12 bit ADC, and the inst amp are included. Main part of program is missing from the element content. 1.5/5 Element 3 data logging has been implemented , no explanations have been given for the additions to the program from element 2. There are test results with sample data included as evidence. 3/10

7/20

SKILLS FOR PROFESSIONAL PRAC.



Element 1 content is content is wrong sequence. The structure of the element is disjointed in parts. There are good points to the element. 3/5 Element 2 has significant parts missing and this upsets the sequence of tasks. 2/5 Element 3 has limited content due to the parts missing, and the sequence is hard to follow. 2/5

7/15