

1809799

by Thomas Grennan

FILE	11301_THOMAS_GRENNAN_1809799_169043_823655272.DOCX (10.51M)		
TIME SUBMITTED	20-MAY-2021 11:47AM (UTC+0100)	WORD COUNT	10962
SUBMISSION ID	154067821	CHARACTER COUNT	53342

HND Electrical Electronic Engineering

SGEE5007 Virtual Instrumentation and
Control Systems

NI LabView Magnitude Comparator

Thomas Grennan: 1809799

Date: 20/05/2021

Word Count: 5346

Contents

Table of Figures.....	4
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	8
1.2.3 Digital Magnitude Comparators.....	11
1.2.4 Analogue to Digital Converters	13
1.2.5 Load Cells (LCKD-1Kg)	15
1.2.6 Instrumentation and Signal Conditioning	16
1.3. Design Requirements	17
1.3.1 Half Adder	17
1.3.2 Full Adder	18
1.3.3 4-Bit Adder	18
1.4. Implementation	19
1.4.1 Half Adder	19
1.4.2 Full Adder	20
1.4.3 4 Bit Adder	21
1.4.4 4-Bit Adder, Testing and Analysis	23
1.4.5 4-Bit Magnitude Comparator	24
1.4.6 Combining 4-Bit adder and 4-bit magnitude comparator	26
1.4.7 8-Bit Magnitude Comparator Design	27
1.5. Testing and Analysis.....	28
1.5.1 4- Bit Adder Testing and Analysis.....	28
1.5.2 4-Bit Comparator, VHDL Simulation Test and Analysis.....	30
Test 1 - A less than B	31
Test 2 – A equal to B	31
Test 3 – A greater than B	32
1.5.3 4-Bit Adder and Comparator FPGA Testing	34
Test 1: A equal to B	34
Test 2: A greater than B	35
Test 3: A less than B	36
1.5.4 8-Bit Magnitude Comparator Simulation Test (NI Multisim).....	37
Test 1 - AGTB.....	37
Test 2 - AEQB.....	37
Test 3 - ALTB.....	38
1.5.5 8-Bit Magnitude Comparator PLD.....	39

1.5.6 8-Bit Magnitude Comparator test - FPGA	40
Test 1 – A Equal to B	40
Test 2 – A greater than B	41
Test 3 – A less than B	42
1.6 Evaluation	43
2. Element 2	44
2.1 Introduction	44
2.2 Design Requirements	45
2.3 ADC Prototypes	49
2.3.1 10 Bit ADC Prototype	49
2.3.2 12 Bit ADC Prototype	53
2.3.3 8 Bit ADC Prototype	55
2.4 8 Bit Comparator Implementation	59
2.4.1 Magnitude Comparator Setup	60
2.5 Overall LabVIEW Program	61
2.6 Testing and Analysis	63
2.7 Evaluation	65
3. Element 3	66
3.1 Testing	68
3.1.1 200g Test	68
3.1.2 500g Test	69
3.1.3 700g test	71
3.2 Element 3 Evaluation	72
4. Overall Conclusions and Evaluation	72
5. References	73
6. Appendix	75
6.1 Element 2	75
6.2 Element 3	81

*Good
table
of
contents
but
there
are
parts
missing*

Table of Figures

Figure 1 – VHDL Code Example	7
Figure 2 – IP Integrator Example	8
Figure 3 – DAQmx	8
Figure 4 – DAQmx Add Icon	9
Figure 5 – DAQmx Start Task Icon	9
Figure 6 – DAQmx Digital Input Icon	9
Figure 7 – DAQmx Digital Output Icon	9
Figure 8 – DAQmx Digital Read Icon	10
Figure 9 – DAQmx While Loop Icon	10
Figure 10 – DAQmx Clear Task Icon	10
Figure 11 – DAQmx Timing Block Icon	10
Figure 12 – 1 bit digital comparator	11
Figure 13 – Digital Comparator Truth Table	11
Figure 14 – 4bit Magnitude Comparator	12
Figure 15 – 8bit Magnitude Comparator	12
Figure 16 – ADC Block Diagram.....	13
Figure 17 – ADC process digagram	13
Figure 18 – SAR ADC.....	14
Figure 19 – Pros and Cons of ADCs	14
Figure 20 – Strain Gauge Load Cell diagram	15
Figure 21 – Omega LCKD-1Kg.....	15
Figure 22 – Instrumentation and Signal Conditioning Process.....	16
Figure 23 – Signal Conditioning – Signal vs Modified	16
Figure 24 – Half Adder and Truth Table	17
Figure 25 – Full adder and Truth table.....	18
Figure 26 – Full Adder	18
Figure 27 – Half Adder Hierarchy.....	19
Figure 28 – Half Adder Code	19
Figure 29 – Full Adder Hierarchy.....	20
Figure 30 – Full Adder Code.....	20
Figure 31 – 4bit Adder Hierarchy.....	21
Figure 32 – 4-bit adder code	22
Figure 33 – 4-Bit adder test bench.....	23
Figure 34 – 4-Bit Set Code.....	23
Figure 35 – 4-Bit Magnitude Comparator.....	24
Figure 36 – 4-Bit Magnitude Comparator IP	24
Figure 37 – 4-Bit Magnitude Comparator Code.....	25
Figure 38 – 4-Bit Magnitude Comparator IP	26
Figure 39 – 4-bit Magnitude Comparator Multisim.....	26
Figure 40 – 8-Bit Magnitude Comparator Hierarchy	27
Figure 41 – 8-bit Magnitude Comparator Multisim.....	27
Figure 42 – 4-Bit adder virtual bus testing.....	28
Figure 43 – Test Results for 4-Bit Adder Virtual Bus	29
Figure 44 – 4-Bit Magnitude Comparator TB code	30
Figure 45 – 4-Bit Comparator Virtual Bus Test - ALTB	31
Figure 46 – 4-Bit Comparator Virtual Bus Test - AEQB	31
Figure 47 – 4-Bit Comparator Virtual Bus Test - ALTB	32
Figure 48 – 4-Bit Adder Virtual Bus Testing Table	33
Figure 49 – 4-Bit Adder and Comparator FPGA Testing and Evidence AEQB	34
Figure 50 – 4-Bit Adder and Comparator FPGA Testing and Evidence AGTB	35

Figure 51 - 4-Bit Adder and Comparator FPGA Testing and Evidence ALTB	36
Figure 52 – 8-Bit Magnitude Comparator Simulation Test AGTB	37
Figure 53 - 8-Bit Magnitude Comparator Simulation Test AEQB	37
Figure 54 - 8-Bit Magnitude Comparator Simulation Test ALTB.....	38
Figure 55 – 8-Bit Magnitude Comparator Simulation Test Evidence	38
Figure 56 – 8-Bit Magnitude Comparator PLD Circuit	39
Figure 57 – 8-Bit Magnitude Comparator FPGA Test and Evidence - AEQB	40
Figure 58 - 8-Bit Magnitude Comparator FPGA Test and Evidence - AGTB	41
Figure 59 - 8-Bit Magnitude Comparator FPGA Test and Evidence - ALTB	42

Good list of figures for element 1. There are no figures listed for element 2 and 3

1. Element 1

1.1. Introduction

The purpose of Element 1 is the research, design and implementation of a combinational logic system using VHDL and IP integration. The design, build and functional testing will be completed with the use of software packages Vivado and NI Multisim.

Using the VHDL programming language a 4-bit adder and magnitude comparator will be designed, built and tested, an 8-bit magnitude comparator will also be designed, built and tested.

Testing will be carried out using their respective packages testing facilities, once each test is successful, they will be downloaded onto an FPGA board for physical testing. Both circuits will then be analysed to ensure that they are working as initially designed in order to prove the success of the process.

*Reasonable
introduction*

1.2 Research

1.2.1 Digital design methods using VHDL and IP integration

VHDL (Very High-Speed Integrated Circuit Hardware Description Language) is one of the widely used hardware description languages, this is often the programming language choice of engineers using FPGA and ASIC devices with it being a programming language approved by the IEEE. By using this language engineers have the ability to interface between LabVIEW and Nexys FPGA boards which can be easily integrated together.

VHDL is widely used for text-based models which describe logic circuits such as adders and comparators, this is due to its ability to monitor complex circuits for simulation and synthesis (Pellerin, n.d.). An example of this text is seen in figure 1.

```
reg1: process (rst, clk)
begin
  if rst = '1' then
    q_reg <= (others => '0');
    q_i <= (others => '0');
  elsif rising_edge(clk) then
    if s_1 = '1' then
      q_i(0) <= q_i(7);
      loop1: for i in 6 downto 0 loop
        q_i(i + 1) <= q_i(i);
      end loop loop1;
      q_reg <= y;
    else
      q_i <= q_reg;
      q_reg <= y;
    end if;
  end if;
end process reg1;
```

Figure 1 – VHDL Code Example

This is a favourable language due to its functionality, On Vivado the circuit can be designed within the software programme, the model can be simulated using a “Virtual Bus” to ensure the design is functioning as intended before physically testing it on hardware, any problems can be corrected within Vivado. (Dekker, 2014)

Intellectual Properties (IP) design has been developed for FPGA boards, IPs are blocks which contain a full electronic system of logic circuits such as adders and comparators which then seamlessly works as a finished circuit, these blocks can be used for various different circuits once created and also reduce the complexity of the circuit by minimising the amount of connections needed reducing design efforts and maximise productivity using Vivado and other IP integrator GUIs. (Xilinx, 2018), An example of an IP is displayed in Figure 2.

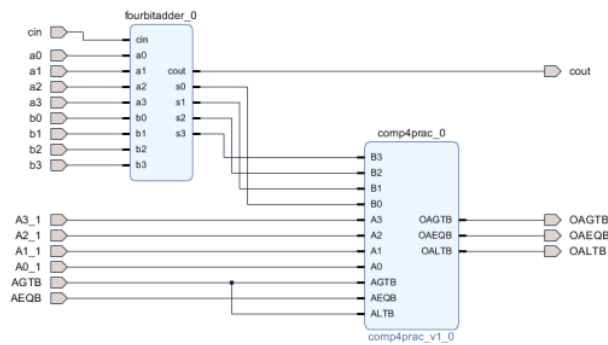


Figure 2 – IP Integrator Example

Excellent work here with good relevant sources to support your research

1.2.2 NI DAQmx data acquisition used in LabVIEW

NI DAQmx is a National Instruments driver which controls all aspects of a data acquisition (DAQ) system, this includes signal conditioning, it covers from low-level configuration up to high-level programming for various operating software, device control and LabVIEW (National Instruments, 2019)

NI DAQmx is National Instruments current model of data acquisition driver, it is built upon National Instruments original NI-DAQ, which was upgraded to DAQmx due to architecture changes. DAQmx offers an entirely new driver architecture and application programming interface (API), DAQmx offers a more user-friendly interface to increase ease of use and improve performance over the original DAQ. (National Instruments, 2019)

DAQmx offers a visual programming language (Figure 3).

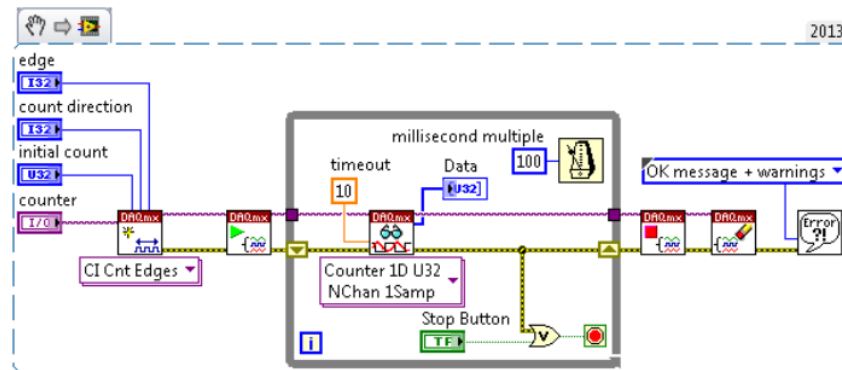


Figure 3 – DAQmx

Add

The add icon is used to add the sum of two values “X” and “Y” to give an output of “X+Y”.



Figure 4 – DAQmx Add Icon

Start Task

The start task function runs the task which carries out the process depending on what functions are down the line. Much like a start button it will only start the process when commanded, ensuring that the process will not run when not required.

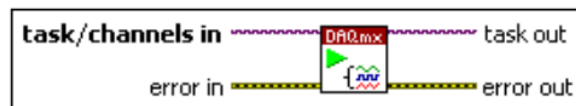


Figure 5 – DAQmx Start Task Icon

Digital Input

Digital inputs are used to measure inputs sent to the system; the input icon has parameters than can be altered to achieve the correct “Task Out” by generating single, dual and multiple channels by linking out the line input.

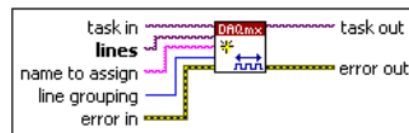


Figure 6 – DAQmx Digital Input Icon

Digital Output

Much like the digital input, the Digital Output works in the opposite was to the input by giving single, dual or multiple channel outputs by modifying the icon parameters.

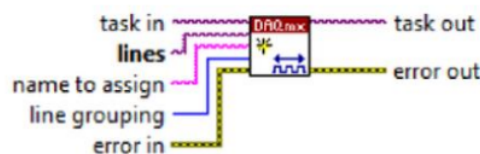


Figure 7 – DAQmx Digital Output Icon

Digital Read

The digital read icon reads a Boolean signal from multiple channels which have one digital line.



Figure 8 – DAQmx Digital Read Icon

While Loop

The “While Loop” allows the program to carry out its task as long as this task is within the constraints of the loop, this will stop when it meets a specific condition not within the loop, such as a stop button. Point 3 on figure 9 shows where the loop can end the process by exiting the loop depending on the state of the button, Point 1 refers to the constraints of the loop and point 2 is the loop count.

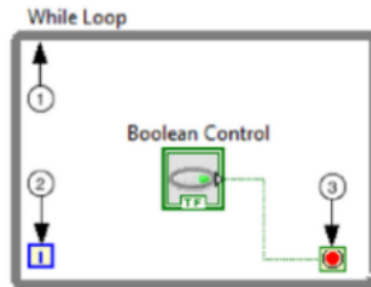


Figure 9 – DAQmx While Loop Icon

Clear Task

The clear task icon clears the existing task, once it has been cleared it must be recreated, this is used to free up memory.

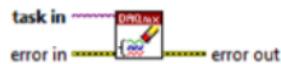


Figure 10 – DAQmx Clear Task Icon

Timing Block

There are various different timing blocks within DAQmx, Figure 11 shows the timing block used to synchronise the tasks within a sequence in the while loop shown in figure 9.



Figure 11 – DAQmx Timing Block Icon

*Good work with relevant
sources referenced to
support research*

1.2.3 Digital Magnitude Comparators

A digital comparator is a highly useful combinational logic circuit, this circuit is used to compare the value of binary values using logic gates. Figure 12 shows a 1-bit digital comparator circuit.

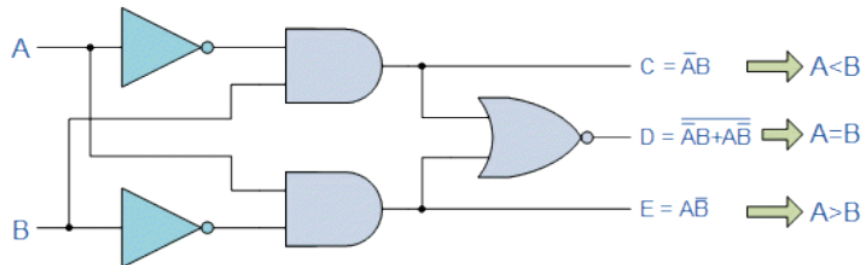


Figure 12 – 1-bit digital comparator

Comparator circuits are constructed with NOT, AND and NOR logic gates, in the arrangement that they are connected the output will give out a value which is dependent on the input values. (Tutorials, 2018)

Digital magnitude comparators are used to compare binary values, these circuits have three output terminals, $A > B$, $A = B$ and $A < B$. These comparators compare a set of variables "Set A" against another set of numbers "Set B". The results of these values can be seen in figure 13 which depicts the basic truth table for these circuits.

Inputs		Outputs		
B	A	$A > B$	$A = B$	$A < B$
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

Figure 13 – Digital Comparator Truth Table

This principle can be made more complex by "cascading" multiple comparators together to make 4-bit (Figure 14), 8-bit and up to n-bit comparators.

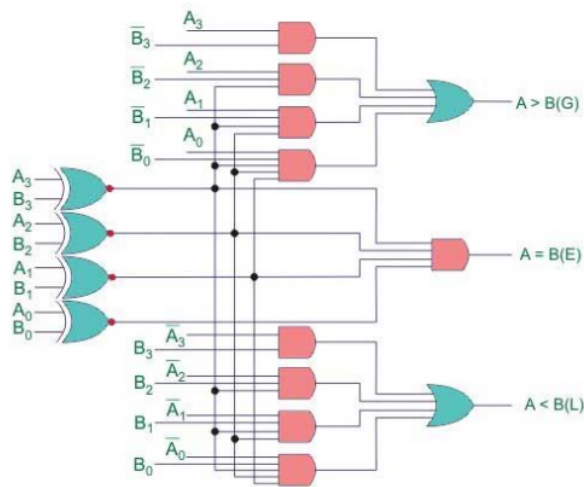


Figure 14 – 4bit Magnitude Comparator

Figure 15 shows two 4-bit magnitude comparators cascaded together in order to achieve an 8-bit magnitude comparator.

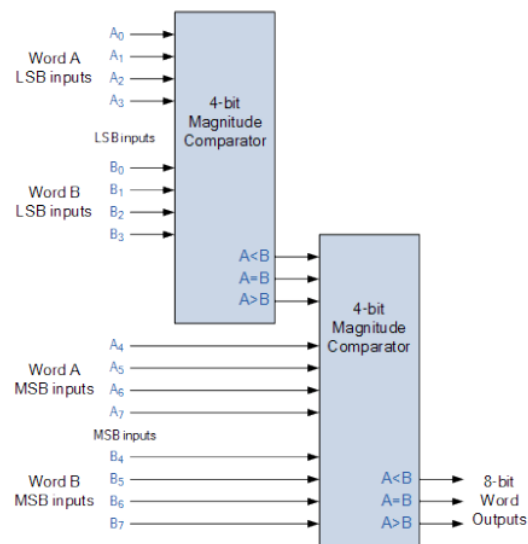


Figure 15 – 8bit Magnitude Comparator

Very good effort here and demonstrates a clear understanding of the principle of operation

1.2.4 Analogue to Digital Converters

An Analogue to Digital converter, also known as an ADC is a useful electronic device that converts an analog signal into a digital binary signal, these devices can take light, audio and temperature signals and turn them into a signal that microcontrollers such as Arduino and Raspberry Pis can understand. ADCs are comparable by their resolution value, their resolution value controls the accuracy of the output signal, the higher the resolution value means the higher the samples per second which in turn gives a higher accuracy output as there are more “bits” per analogue signal.



Very good point to make about resolution which can affect the speed of the overall process.

Figure 16 – ADC Block Diagram

The

process diagram of the ADC is shown in figure XX.



Figure 17 – ADC process diagram

There are five main types of ADCs: Dual Slope, Flash, Pipeline, Successive Approximation (SAR) and Delta-Sigma. These are compared in figure 19. Two of the wider used ADCs are the SAR and Delta-Sigma.

The SAR (Figure 18) offers a great balance between resolution and speed whilst having the ability to handle a sample rate up to 10MHz with a resolution up to 18bits. SAR designs offer reliability and stability at a relatively low cost per unit. They also are widely used due to their ability to be multiplexed using multiple channels with one ADC. (Dewesoft, 2020)

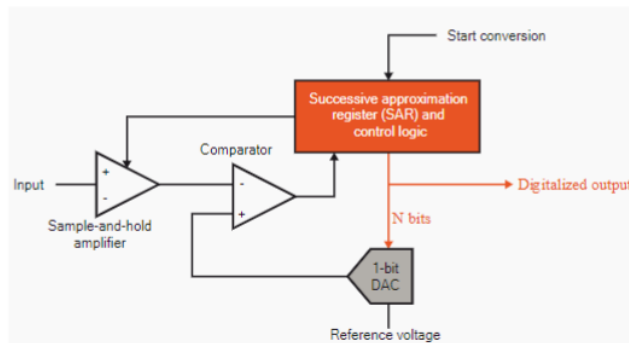


Figure 18 – SAR ADC

SAR ADCs have limited amplitude axis resolution which restricts their use in applications in applications such as noise, shock and vibration. The Delta-Sigma ADC is a newer design of ADC, this design has been built upon improving the amplitude axis resolution and high frequency quantization noise which is seen in the SAR ADC. (Dewesoft, 2020)

ADC Type	Pros	Cons	Max Resolution	Max Sample Rate
Dual Slope	Inexpensive	Low speed	20 bits	100 Hz
Flash	Very fast	Low bit resolution	12 bits	10 GHz
Pipeline	Very fast	Limited resolution	16 bits	1 GHz
SAR	Good speed/resolution ratio	No inherent anti-aliasing protection	18 bits	10 MHz
Delta-sigma ($\Delta\Sigma$)	High dynamic performance, inherent anti-aliasing protection	Hysteresis on unnatural signals	32 bits	1 MHz

Good use of evaluation table

Figure 19 – Pros and Cons of ADCs

1.2.5 Load Cells (LCKD-1Kg)

A load cell is an electronic transducer that converts forces of pressure, torque and compression into an electrical output signal which is relative to the force applied, this means that the as the force applied to the loadcell increases so does the output, this is because the output is proportional to the input.

There are various types of load cells such as strain-gauge, pneumatic and hydraulic. Among these types of load cells. The most commonly used is the strain gauge, this load cell has four strain gauges which are connected within the “Wheatstone Bridge” arrangement (Figure 20), when a force is applied to the cell it causes a deformation to the strain gauge which is then measured as a change in electrical signal as the voltage will fluctuate under load. (HBM, 2019)

The strain gauge load cell has 4 strain gauges within them, two compression gauges and two tension gauges, these are connected with compensation values (Omega, 2020). Without a load on the load cell each strain gauge will have equal resistance, when a load is applied the strain gauge resistance will differ causing the voltage to change, this output can then be scaled and converted into useable data.

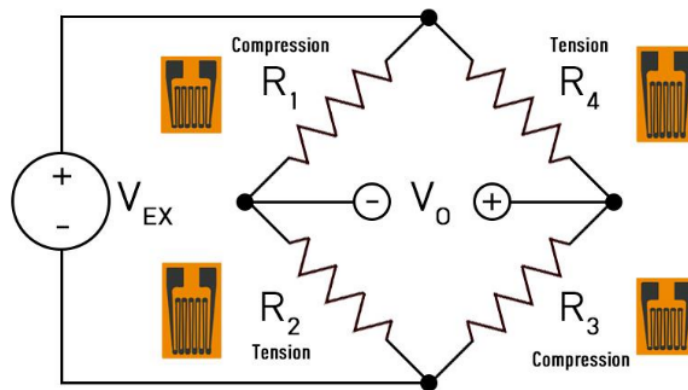


Figure 20 – Strain Gauge Load Cell diagram

Omega are a renowned manufacturer of Load Cells, they manufacture the LCKD-1kg which is a compression load cell with a load capacity of 1kg or 2.2lb. Omega manufacture loadcells from 1kg up to 450kg in standard and metric models housed in a stainless steel with high performance strain gauges to ensure linearity and stability. (Omega, 2018)



Figure 21 – Omega LCKD-1Kg

Good work

1.2.6 Instrumentation and Signal Conditioning

Analog signals will often need to be scaled before they can become digital signals, this is done through the process of signal conditioning. Signal conditioning is the manipulation of a signal in order to prepare it for the next stage of its process, whereby it will be converted to digital with the use of an ADC. (Instruments, 2019), Figure 22 displays the process of signal conditioning.

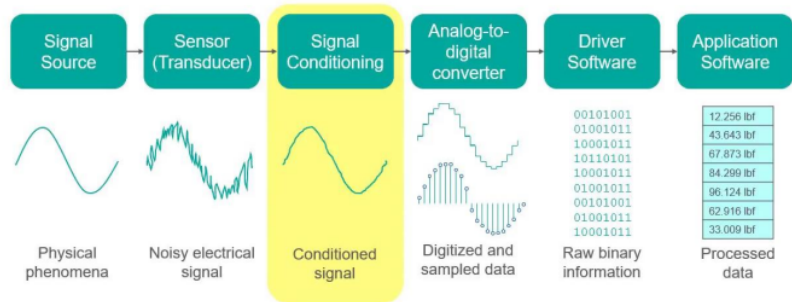


Figure 22 – Instrumentation and Signal Conditioning Process

Signal conditioning is a data acquisition process whereby a “signal conditioner” instrument is used to provide precise measurements. A signal conditioner will convert an input-signal into an output-signal that can be used. The purpose of this instrument is to amplify and convert the input signal into a useful data that is compatible for data acquisition (HBM, 2019), Essentially a signal conditioner takes a physical signal and refines it to enable application software to be able to use a refined signal.

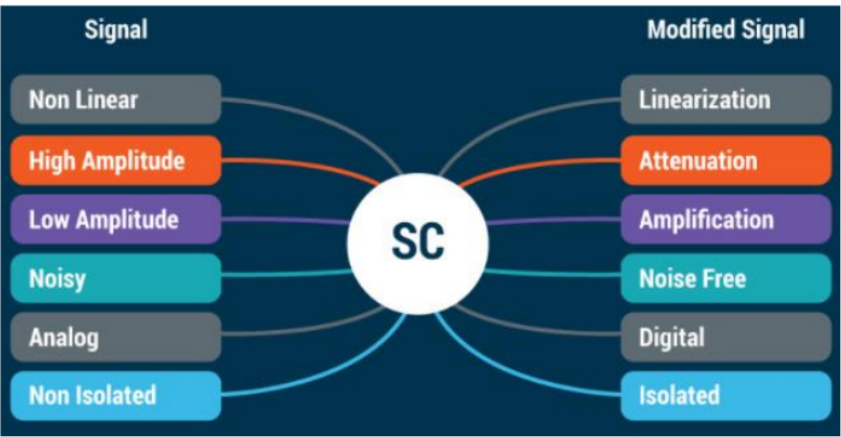


Figure 23 – Signal Conditioning – Signal vs Modified

Many instrumentation devices and analog sensors such as temperature and flow sensors will require signal conditioning before they can be effectively and accurately used.

Research overall is very good and a good number of relevant sources used to support your findings.

1.3 Design Requirements

In this task a 4-bit adder is designed using the VHDL language in Xilinx's Vivado programme. The process of designing the 4-bit adder is split into sections in order to make the process manageable. A 4-bit adder consists of 4 full adders, within each full adder is 2 half adders.

1.3.1 Half Adder

The first process of the design process is to construct a half adder, a half adder is a combinational logic circuit which adds the two signals at its input terminal, this circuit is an XOR gate in parallel with an AND gate, it then produces a sum bit (S) and a carry bit (C) depending on the input value at the terminals. The half-adder is the simplest form of the adder circuit. (circuitstoday, 2017)

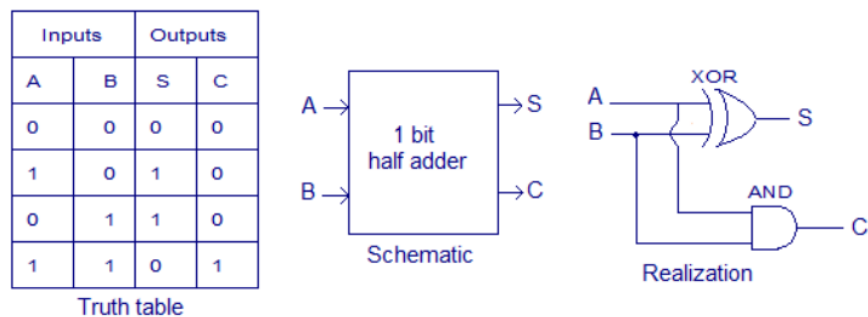


Figure 24 – Half Adder and Truth Table

1.3.2 Full Adder

A full adder is constructed by duplicating the half adder and cascading them whilst adding an OR gate. This adder has 3 inputs (A, B and Carry In) and two outputs (Sum and Carry Out). Full adders are designed to take 8bits or 1 byte whilst cascading the carry bit from one adder to the next. (geeksforgeeks, 2019)

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

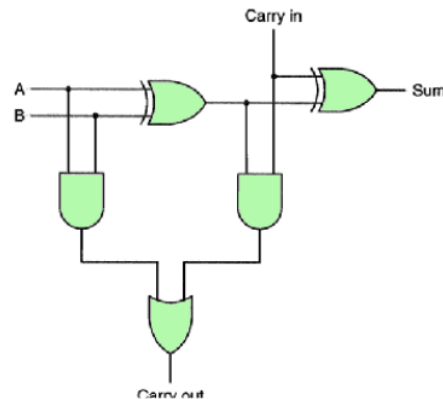


Figure 25 – Full adder and Truth table

1.3.3 4-Bit Adder

A 4-Bit adder is constructed by cascading four full adders (Figure 26). In this circuit the carry-out output feeds from the current adder into the next one adders carry-in terminal in the cascade. Each full adder will also give a sum-out signal.

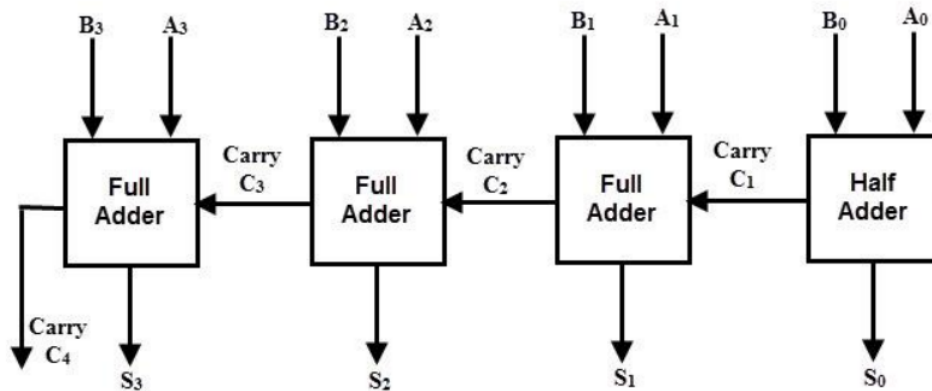


Figure 26 – Full Adder

Components for the 4 bit adder design have been clearly identified with good supporting images.

1.4 [Implementation](#)

1.4.1 [Half Adder](#)

Using Xilinx's Vivado Software, the code for the half-adder was written as a design source, this is done utilising the VHDL language. This is designed referring to the truth table in figure 24. The hierarchy for the half adder is seen in figure 27, along with the code in figure 28.

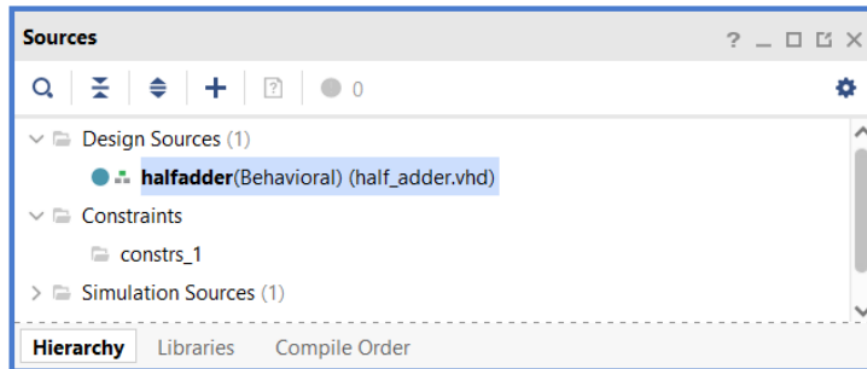


Figure 27 – Half Adder Hierarchy

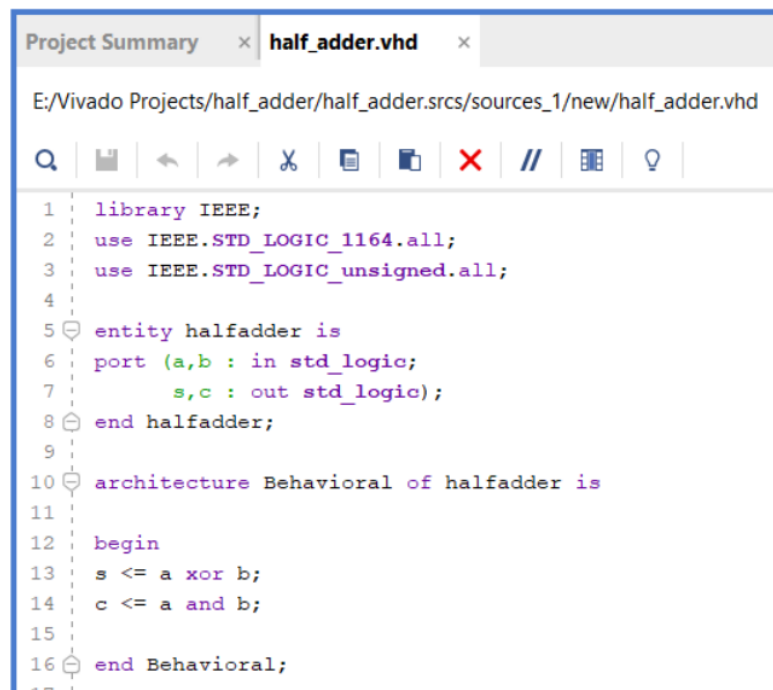


Figure 28 – Half Adder Code

1.4.2 Full Adder

Utilising the code previously written for the half-adder, a full adder can be designed. This is done by duplicating the half adder code and writing a code to connect them, this is shown in the full-adders hierarchy in figure 29.

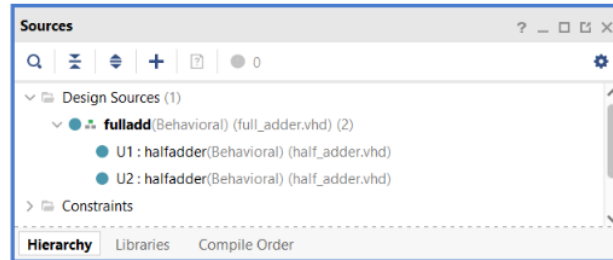


Figure 29 – Full Adder Hierarchy

Figure 30 shows the code that is written to utilise the two half adders together, half adder 1 is referenced as U1, half-adder 2 as U2.

```
full_adder.vhd
E:/Vivado Projects/full_adder/full_adder.srcs/sources_1/new/full_adder.vhd

1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  entity fulladd is
4  port(
5    a : in STD_LOGIC;
6    b : in STD_LOGIC;
7    cin : in STD_LOGIC;
8    cout : out STD_LOGIC;
9    s : out STD_LOGIC
10 );
11 end fulladd;
12 architecture Behavioral of fulladd is
13 component halfadder
14 port (
15   a : in STD_LOGIC;
16   b : in STD_LOGIC;
17   c : out STD_LOGIC;
18   s : out STD_LOGIC
19 );
20 end component;
21
22 signal c1 : STD_LOGIC;
23 signal c2 : STD_LOGIC;
24 signal s1 : STD_LOGIC;
25 begin
26 U1 : halfadder
27 port map(
28   a => a, b => b, c => c1, s => s1);
29 U2 : halfadder
30 port map(
31   a => s1, b => cin, c => c2, s => s);
32 cout <= c2 or c1;
33 end Behavioral;
34
```

Figure 30 – Full Adder Code

1.4.3 4 Bit Adder

The 4 Bit Adder is then designed, this adder utilises 4 full adders, each full adder is seen as "1 Bit". As seen in the design source hierarchy it shows that the 4-bit adder is constructed from 4 full adders which each contain 2 half adders. (Figure 31)

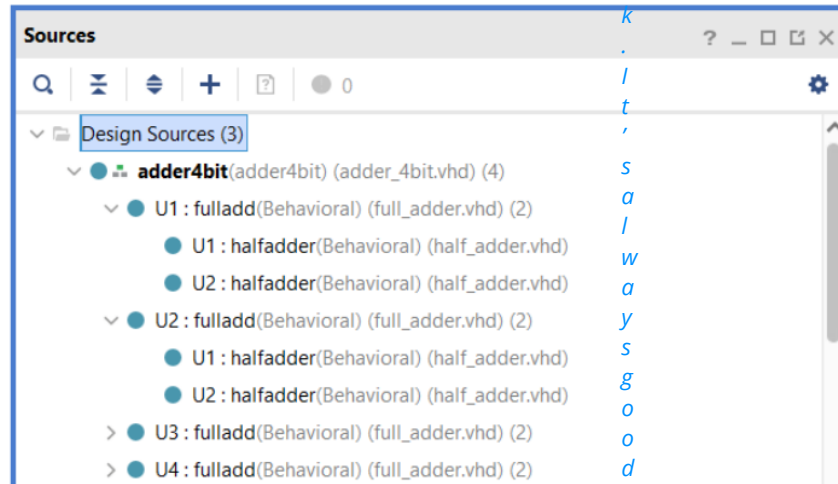


Figure 31 – 4bit Adder Hierarchy

Hierarchical and inheritance has been identified in the design

A Vivado program was written in order for each full adder (1 Bit) to work as a 4-Bit adder (Figure 32)

```
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;
55 ⊞

```

VHDL code is correct and neatly presented with the 4 full adder components showing in the image

Figure 32 – 4-bit adder code

1.4.4 4-Bit Adder, Testing and Analysis

Using Vivado's software the 4-bit adder was simulated. Firstly, a test bench was used in order to generate the code needed, the code's initialisation specification was written to the test bench in order for it to perform the simulation. A testbench automatically generates a template file of a VHDL entity's simulation.

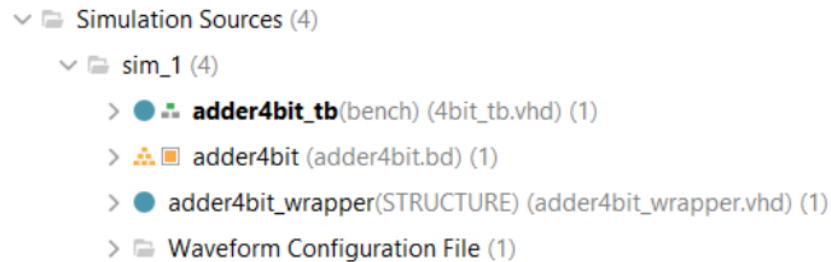


Figure 33 – 4-Bit adder test bench

From the written code it would be expected that the 4-bit adder will add the numerical values of "Set A" and "Set B" to give a numerical output.

```
64 |      -- Put initialisation code here
65 |      a0 <= '1';
66 |      a1 <= '0';
67 |      a2 <= '0';
68 |      a3 <= '0';
69 |      b0 <= '1';
70 |      b1 <= '0';
71 |      b2 <= '0';
72 |      b3 <= '0';
73 |      cin <= '0';
74 |      wait for 10 ns;
75 |
76 |      a0 <= '1';
77 |      a1 <= '1';
78 |      a2 <= '1';
79 |      a3 <= '0';
80 |      b0 <= '0';
81 |      b1 <= '0';
82 |      b2 <= '1';
83 |      b3 <= '0';
84 |      cin <= '0';
85 |      wait for 10 ns;
-- |
```

Test data is correct with a good range of values

Figure 34 – 4-Bit Set Code

1.4.5 4-Bit Magnitude Comparator

A 4-Bit magnitude comparator is constructed of an array of logic gates, a simplified version of this is shown in figure 35.

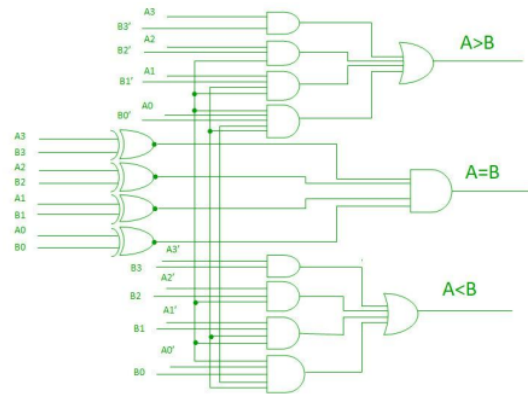
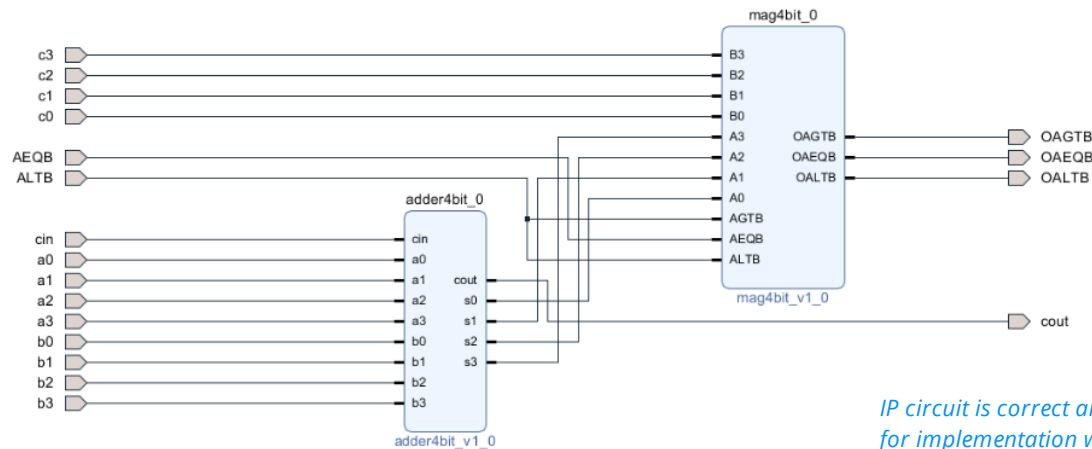


Figure 35 – 4-Bit Magnitude Comparator

These comparators take two sets of 4-bit inputs (A0-A3 and B0-B3), these two sets are compared with each other by the logic gates in order to give an output, these outputs will be A_Greater_Than_B (AGTB), A_Less_Than_B (ALTB) and A_Equal_To_B (AETB). The IP block diagram for this circuit is displayed in figure 36.



IP circuit is correct and is ready for implementation with FPGA. All IO are set.

Figure 36 – 4-Bit Magnitude Comparator IP

This circuit was then deciphered and a code was written in Vivado for the VHDL boards. This was written, this was written in order to give an output for each possible scenario of inputs using “if commands”. “elsif” commands are also used, this is a conditional statement used after an if statement that performs a function if it is “true” (Figure 37)

```

25  architecture BEHAVIORAL of COMP_4_NI is
26  begin
27  PROCESS (A0, B0, A1, A2, A3, B1, B2, B3, AGTB, AEQB, ALTB)
28  BEGIN
29  if A3>B3 then OAGTB<='1';
30  OAEQB<='0';
31  OALTB<='0';
32  elsif A3<B3 then OAGTB<='0';
33  OAEQB<='0';
34  OALTB<='1';
35  elsif A3=B3 then
36  if A2>B2 then OAGTB<='1';
37  OAEQB<='0';
38  OALTB<='0';
39  elsif A2<B2 then OAGTB<='0';
40  OAEQB<='0';
41  OALTB<='1';
42  elsif A2=B2 then
43  if A1>B1 then OAGTB<='1';
44  OAEQB<='0';
45  OALTB<='0';
46  elsif A1<B1 then OAGTB<='0';
47  OAEQB<='0';
48  OALTB<='1';
49  elsif A1=B1 then
50  if A0>B0 then OAGTB<='1';
51  OAEQB<='0';
52  OALTB<='0';
53  elsif A0<B0 then OAGTB<='0';
54  OAEQB<='0';
55  OALTB<='1';
56  elsif A0=B0 then
57  if AGTB='1' and AEQB='0' and ALTB='0' then OAGTB<='1';
58  OAEQB<='0';
59  OALTB<='0';
60  elsif ALTB='1' and AEQB='0' and AGTB='1' then OAGTB<='0';
61  OAEQB<='0';
62  OALTB<='0';
63  elsif AEQB='1' and AGTB='0' and ALTB='0' then OAGTB<='0';
64  OAEQB<='1';
65  OALTB<='0';
66  elsif ALTB='1' and AEQB='0' and AGTB='0' then OAGTB<='0';
67  OAEQB<='0';
68  OALTB<='1';
69  elsif AEQB='1' then OAGTB<='0';
70  OAEQB<='1';
71  OALTB<='0';
72  elsif ALTB='0' and AEQB='0' and AGTB='0' then OAGTB<='1';
73  OAEQB<='0';
74  OALTB<='1';
75
76  end if;
77  END PROCESS;
78  end BEHAVIORAL;

```

VHDL is correct

Figure 37 – 4-Bit Magnitude Comparator Code

1.4.6 Combining 4-Bit adder and 4-bit magnitude comparator

Firstly the 4-bit adder and 4-bit magnitude comparator were IP integrated (Figure 38), this enables them to be combined effectively. A new project was created in order to connect the two IP integrated circuits into 1 "IP Wrapper". When the circuit is combined various inputs from the VHDL board will feed the 4-bit adder, the adder will then give outputs to feed the comparator circuits Set A (Signals from the VHDL board will also be connected straight to the comparator's Set B). The values sent to set B are compared with the values of the two other sets sum, an output of AETB, AEQB or AELB is given accordingly.

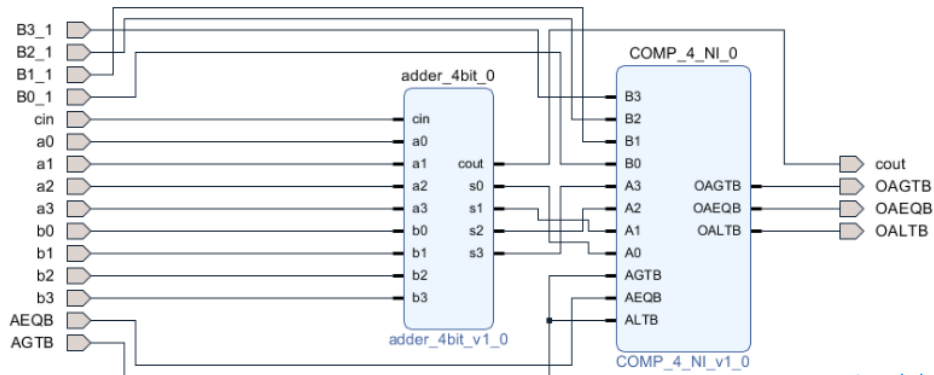


Figure 38 – 4-Bit Magnitude Comparator IP

Good design

This circuit was then created in NI Multisim (Figure 39)

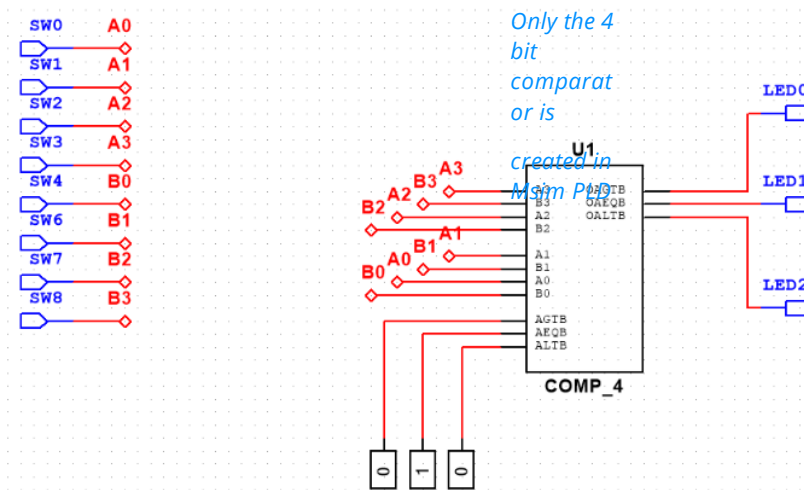


Figure 39 – 4-bit Magnitude Comparator Multisim

1.4.7 8-Bit Magnitude Comparator Design

Using the Multisim package an 8-Bit magnitude comparator was constructed, this is constructed by cascading two 4-bit comparators. Figure 40 shows the hierarchy of the 8-bit magnitude comparator in Vivado.

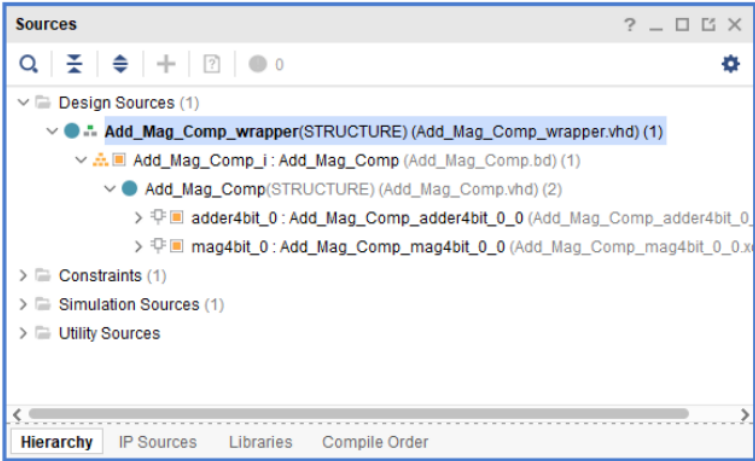


Figure 40 – 8-Bit Magnitude Comparator Hierarchy

Using the principle of cascading two 4-bit comparators a circuit was built in Multisim (Figure 41), Toggle switches were used to feed binary inputs to the comparators. Three outputs (AGTB,AEQB and ALTB) were fed from comparator 1 into 2, these were then compared with the inputs fed into comparator 2, a visual output of the state is given by LEDs being connected to the second comparator output

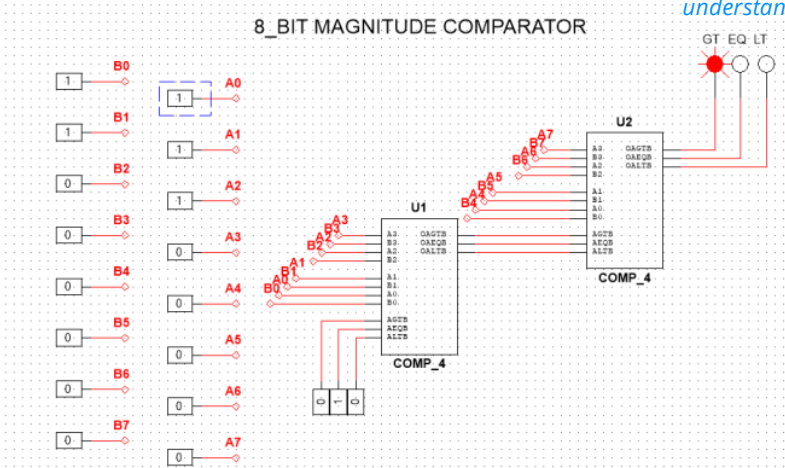


Figure 41 – 8-bit Magnitude Comparator Multisim

8 bit comparator explanation is good and clearly demonstrates the understanding.

1.5 [Testing and Analysis](#)

The testing stage was then undertaken, each circuit for each of the adders and comparators will be tested respectively to ensure that they are functioning as designed.

1.5.1 [4- Bit Adder Testing and Analysis](#)

Figure 42 shows that the 4-bit adder virtual bus test is working as expected by adding the two binary inputs together and producing an output, for this test inputs were inserted into the virtual bus before any I/O values were assigned. Tests were then undertaken by changing the input values of 'A' and 'B' in order to get numerical outputs, this test was undertaken to ensure that the numerical output matched the sum of the inputs.

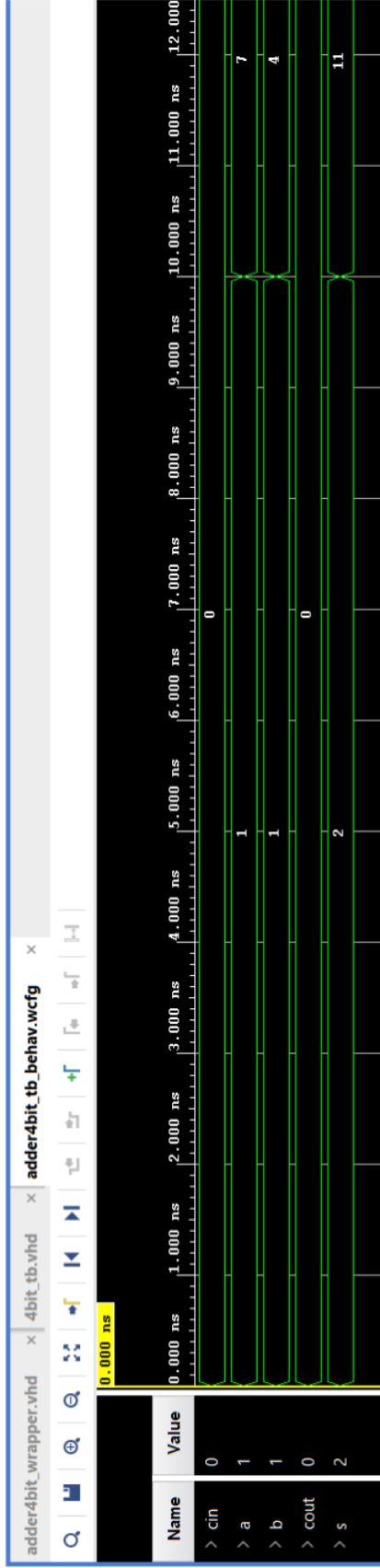


Figure 42 – 4-Bit adder virtual bus testing

Good test result

Figure 43 shows the comparison between the expected results and actual results from the Vivado program, as they are identical it proves that the program is performing as intended.

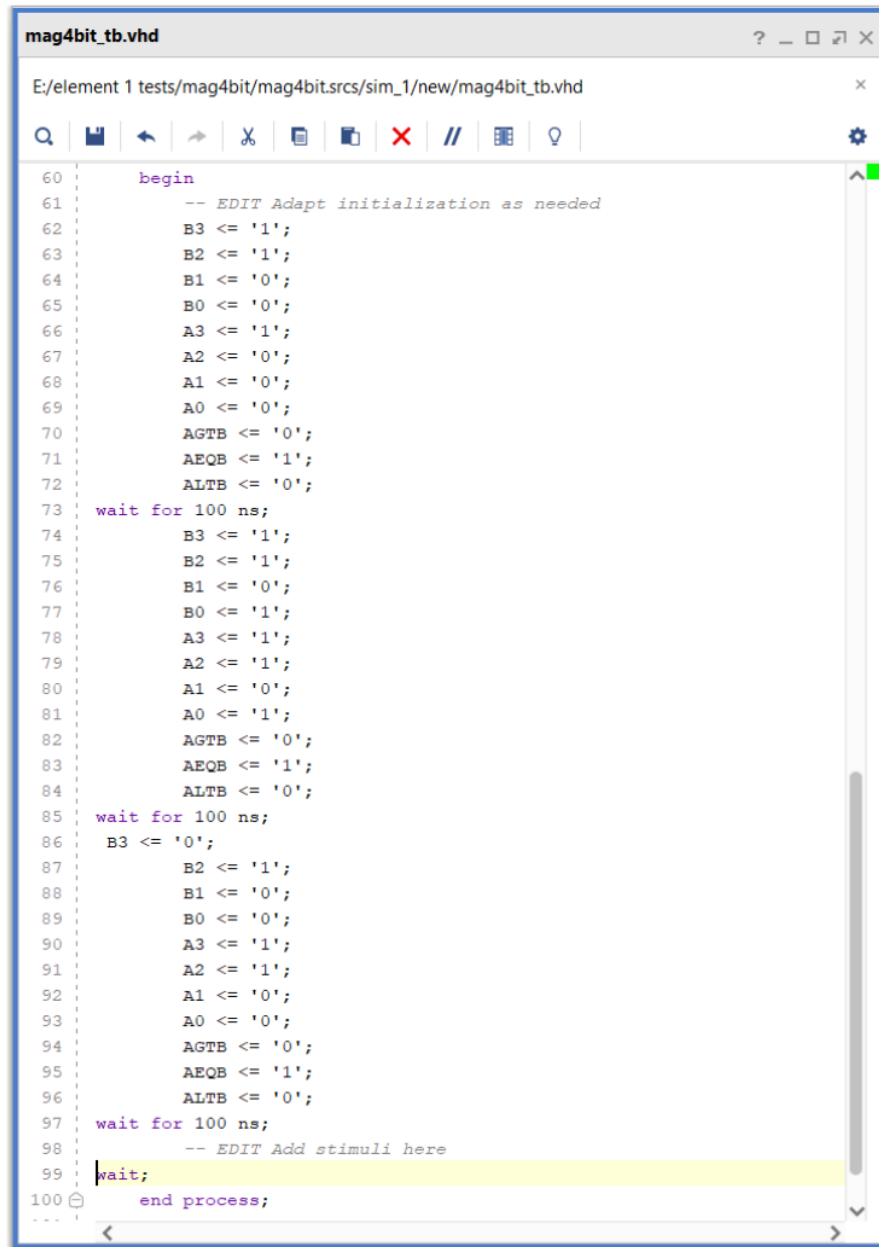
Set A			Set B			Expected Result			Actual Result		
Binary	Decimal	C In	Binary	Decimal	C In	Binary	Decimal	C Out	Binary	Decimal	C Out
0001	1	0	0001	1	0	2	0010	0	2	0010	0
0111	7	0	0100	4	0	11	1011	0	11	1011	0

Figure 43 – Test Results for 4-Bit Adder Virtual Bus

Good use of results table

1.5.2 4-Bit Comparator, VHDL Simulation Test and Analysis

A VHDL testbench program was then created, Using the testbench website the initialisation code was then generated, this code will produce the simulations values (Figure 44)



```
mag4bit_tb.vhd
E:\element 1 tests\mag4bit\mag4bit.srcs\sim_1\new\mag4bit_tb.vhd

60     begin
61         -- EDIT Adapt initialization as needed
62         B3 <= '1';
63         B2 <= '1';
64         B1 <= '0';
65         B0 <= '0';
66         A3 <= '1';
67         A2 <= '0';
68         A1 <= '0';
69         A0 <= '0';
70         AGTB <= '0';
71         AEQB <= '1';
72         ALTB <= '0';
73     wait for 100 ns;
74         B3 <= '1';
75         B2 <= '1';
76         B1 <= '0';
77         B0 <= '1';
78         A3 <= '1';
79         A2 <= '1';
80         A1 <= '0';
81         A0 <= '1';
82         AGTB <= '0';
83         AEQB <= '1';
84         ALTB <= '0';
85     wait for 100 ns;
86         B3 <= '0';
87         B2 <= '1';
88         B1 <= '0';
89         B0 <= '0';
90         A3 <= '1';
91         A2 <= '1';
92         A1 <= '0';
93         A0 <= '0';
94         AGTB <= '0';
95         AEQB <= '1';
96         ALTB <= '0';
97     wait for 100 ns;
98         -- EDIT Add stimuli here
99     wait;
100 end process;
```

Figure 44 – 4-Bit Magnitude Comparator TB code

Using the virtual bus method on Vivado, three tests were undertaken from the testbench program to ensure that it is operating correctly.

Test 1 - [A less than B](#)

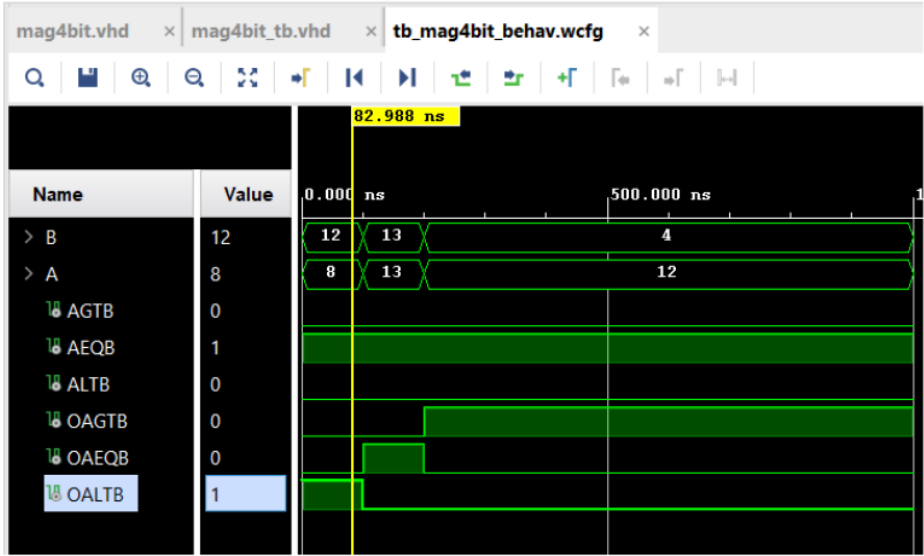


Figure 45 – 4-Bit Comparator Virtual Bus Test - ALTB

Test 2 – [A equal to B](#)

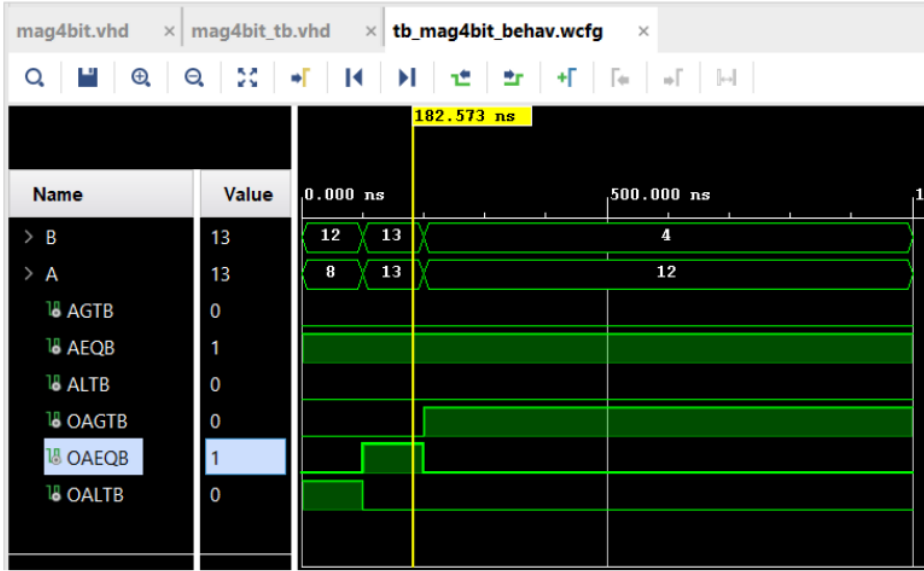


Figure 46 – 4-Bit Comparator Virtual Bus Test - AEQB

Test 3 – [A greater than B](#)

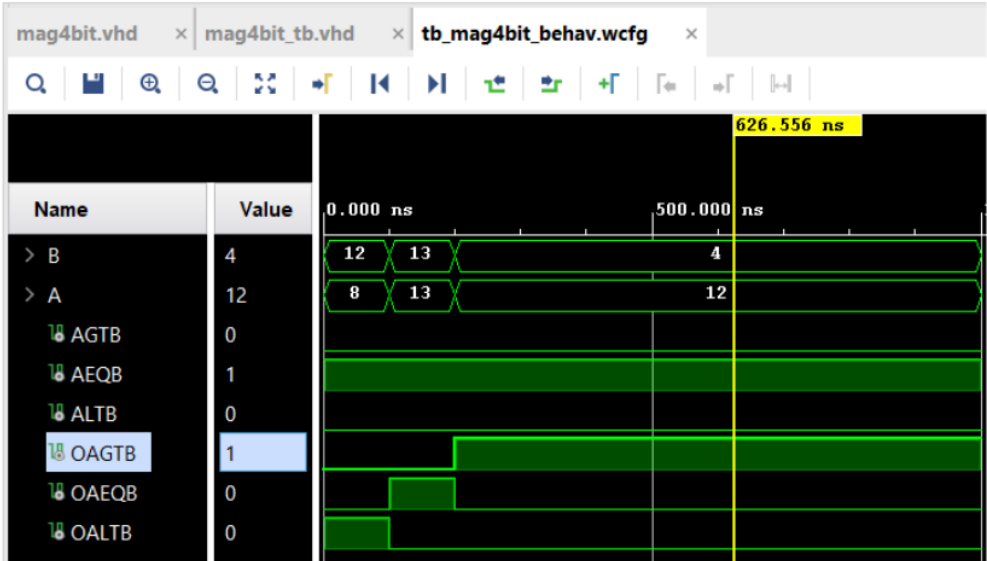


Figure 47 - 4-Bit Comparator Virtual Bus Test - ALTb

Very good set of results

All results were then entered into a table where the readings from the tests will be compared to the results that would be expected from the circuits operation. As seen below in figure 48 the expected output matches the actual output proving that the circuit is operational.

	Set A		Set B		Inputs			Outputs	
	Binary	Decimal	Binary	Decimal	AGTB	AETB	ALTB	Expected Output	Actual Output
Test 1	1000	8	1100	12	0	0	1	OALTB	OALTB
Test 2	1101	13	1101	13	0	1	0	OAEQB	OEQTB
Test 3	1100	10	0100	4	1	0	0	OAGTB	OAGTB

Figure 48 – 4-Bit Adder Virtual Bus Testing Table

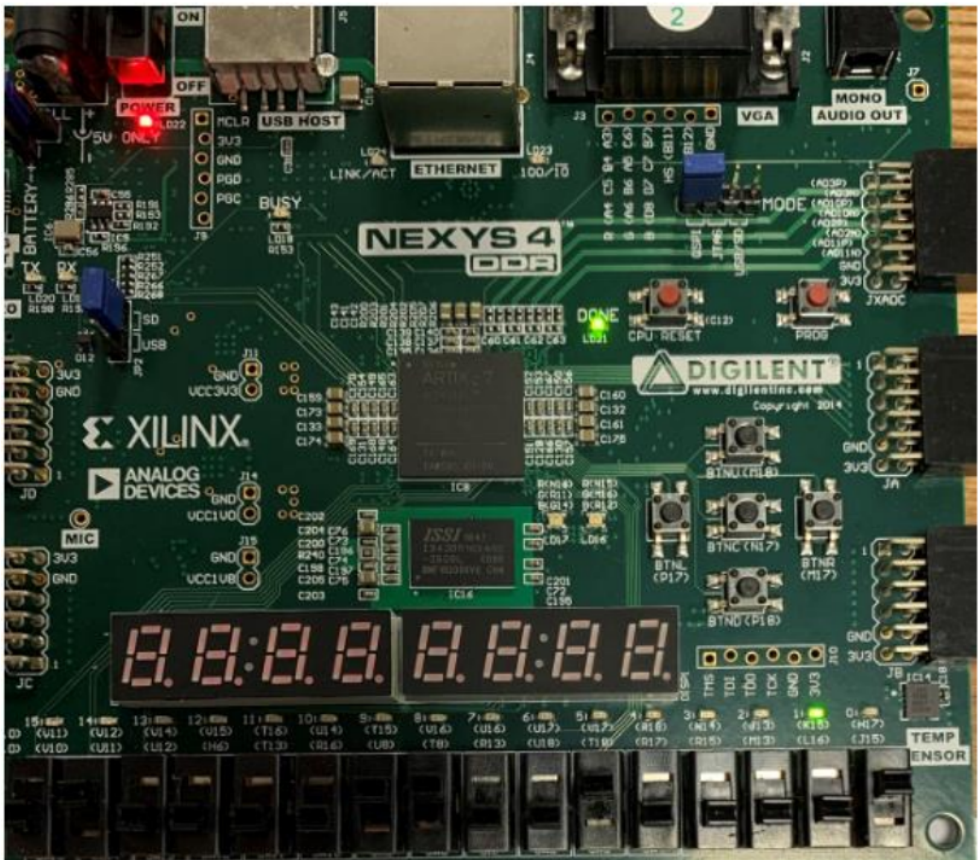
After the simulation tests were proven to be accurate, the program was downloaded onto the Nexys Diligent FPGA board in order for practical testing. The same tests as above will be carried out, one test for A greater than B, A equal to B and A less than B, each possible will be output will be assigned to an individual LED on the FPGA board. The binary values of AO-A3, BO-B3 and CO-C3 will be set using the boards switches.

Good use of table of results

1.5.3 4-Bit Adder and Comparator FPGA Testing

Test 1: A equal to B

Test 1 is A equal to B, for this test the switches on the FPGA board were set manually. Set A were set to a decimal value of 1, set B to a decimal value of 2. The set value was set to a decimal value of 3. Therefore, the sum of A and B will be equal to the set value and shall illuminate LED 1 which is assigned to the output “AEQB”



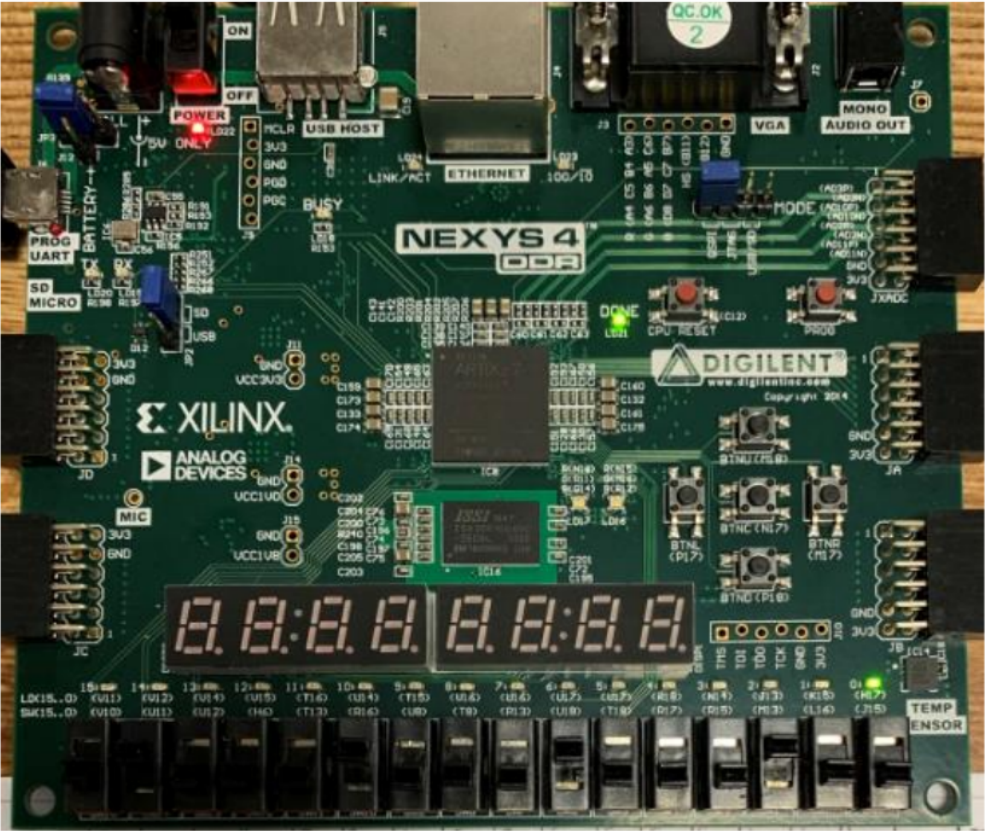
Test 1 – A equal to B																		
				Set Value				Sum of B				Sum of A				AGTB	AEQB	ALTB
S15	S14	S13	S12	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0			
Cin	ALTB	AEQB	N/A	C3	C2	C1	C0	B3	B2	B1	B0	A3	A2	A1	A0			
Binary Value				0	0	1	1	0	0	1	0	0	0	0	1			
Decimal Value				3				2				1						
Sum of A + B = 3								Set Value of C = 3								A Equal to B		

Figure 49 - 4-Bit Adder and Comparator FPGA Testing and Evidence AEQB

As seen visually on the FPGA board and from the values in the table below it is seen that outcome from the test correlates with the theory in the statement above proving that the test was successful

Test 2: A greater than B

Test 2 is A greater than B. Set A were set to a decimal value of 4, set B to a decimal value of 4. The set value was set to a decimal value of 4. Therefore, the sum of A and B will greater than the set value of 4 and shall illuminate LED 0 which is assigned to the output “AGTB”



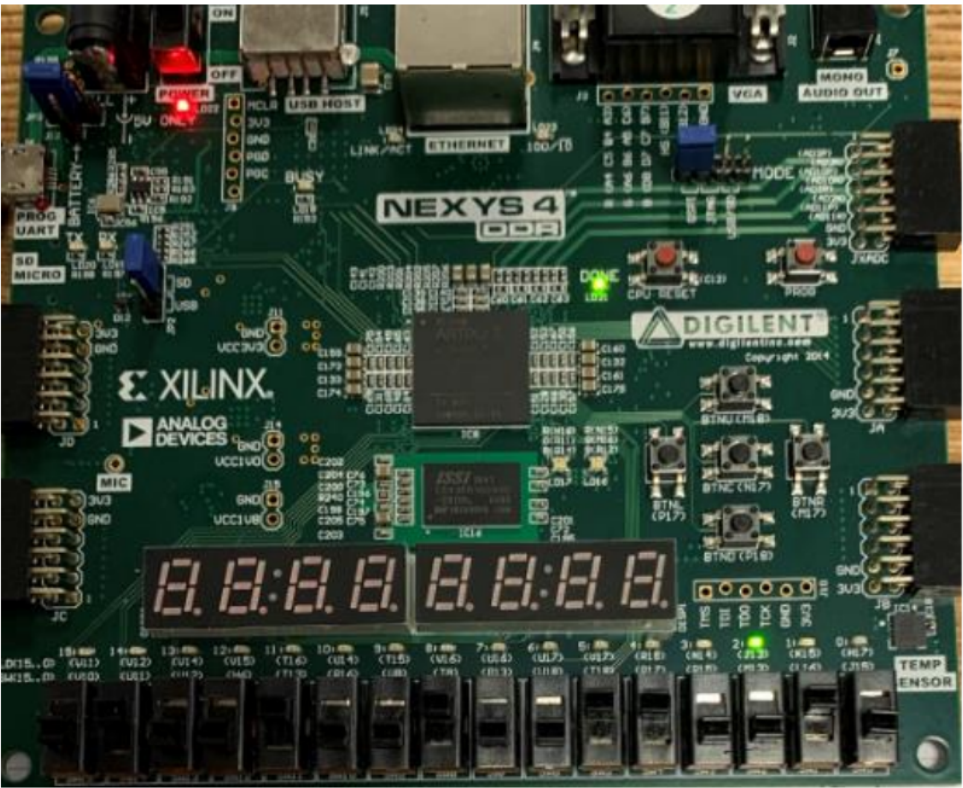
Test 2 – A Greater than B																		
				Set Value				Sum of B				Sum of A				AGTB	AEQB	ALTB
S15	S14	S13	S12	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0			
Cin	ALTB	AEQB	N/A	C3	C2	C1	C0	B3	B2	B1	B0	A3	A2	A1	A0			
Binary Value				0	1	0	0	0	1	0	0	0	1	0	0			
Decimal Value				4				4				4						
Sum of A + B = 8								Set Value of C = 4								A Greater than B		

Figure 50 - 4-Bit Adder and Comparator FPGA Testing and Evidence AGTB

As seen visually on the FPGA board and from the values in the table below it is seen that outcome from the test correlates with the theory in the statement above proving that the test was successful.

Test 3: A less than B

Test 3 is A less than B. Set A were set to a decimal value of 2, set B to a decimal value of 3. The set value was set to a decimal value of 9. Therefore, the sum of A and B will less than the set value of 9 and shall illuminate LED 2 which is assigned to the output “ALTB”



Test 3 – A Less than B																		
				Set Value				Sum of B				Sum of A				AGTB	AEQB	ALTB
S15	S14	S13	S12	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0			
Cin	ALTB	AEQB	N/A	C3	C2	C1	C0	B3	B2	B1	B0	A3	A2	A1	A0			
Binary Value				1	0	0	1	0	0	1	1	0	0	1	0			
Decimal Value				9				3				2						
Sum of A + B = 5								Set Value of C = 9								A Less than B		

Figure 51 - 4-Bit Adder and Comparator FPGA Testing and Evidence ALT B

Excellent set of results

As seen visually on the FPGA board and from the values in the table below it is seen that outcome from the test correlates with the theory in the statement above proving that the test was successful

The outcome of the virtual bus tests and FPGA tests it proves that the 4-Bit adder is functioning as intended.

1.5.4 8-Bit Magnitude Comparator Simulation Test (NI Multisim)

Referring to section 1.4.7 Preliminary tests were performed on Multisim in order to prove that the circuit was functionally operating before it was converted to PLD mode for the Nexys board.

Test 1 - AGTB

A = Binary 7, B = Binary 2 therefore, AGTB.

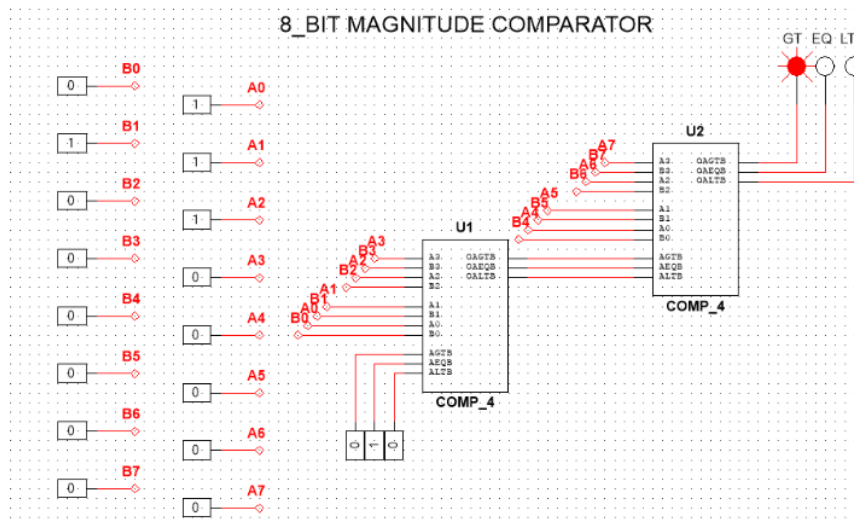


Figure 52 – 8-Bit Magnitude Comparator Simulation Test AGTB

Test 2 - AEQB

A = Binary 7, B = Binary 7 therefore AEQB

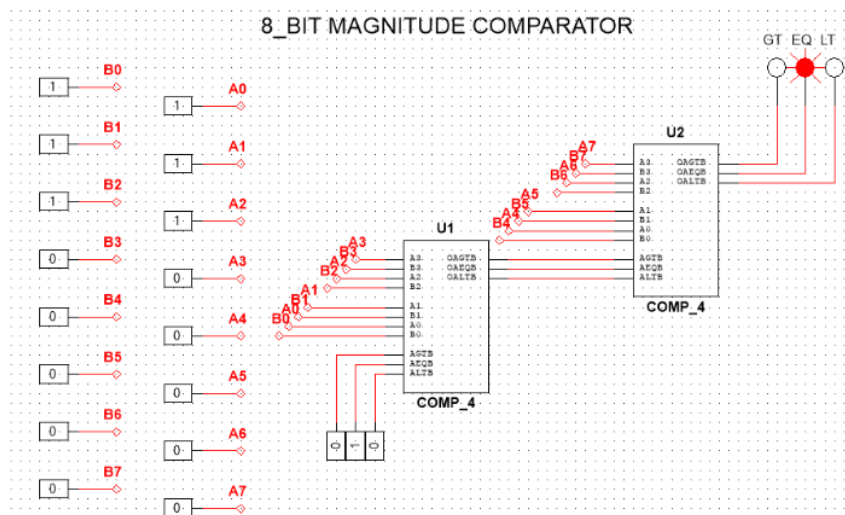


Figure 53 - 8-Bit Magnitude Comparator Simulation Test AEQB

Test 3 - ALTB

A= Binary 2, B – Binary 7 therefore ALTB

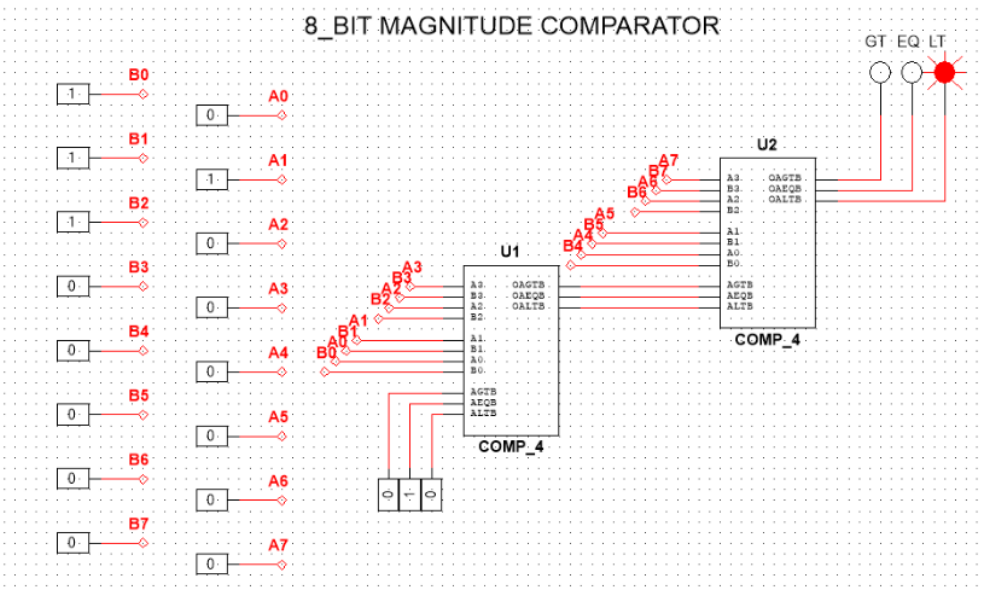


Figure 54 - 8-Bit Magnitude Comparator Simulation Test ALTB

Test no.	Set A		Set B		Expected Result	Actual Result
	Binary	Decimal	Binary	Decimal		
1	00000111	7	00000010	2	AGTB	AGTB
2	00000111	7	00000111	7	AEQB	AEQB
3	00000010	2	00000111	7	ALTB	ALTB

Figure 55 – 8-Bit Magnitude Comparator Simulation Test Evidence

As seen in figure 55, the actual results match the Multisim results proving that the test was successful.

Good work and proves the 8 bit design before PLD design and implementation onto FPGA

1.5.5 8-Bit Magnitude Comparator PLD

The 8-bit magnitude comparator was then exported to PLD mode through Multisim, this allows the circuit to be controlled from a programmable logic device such as the Nexys board. The inputs to the circuit will be controlled by rocker switches and the device output are the boards LEDs.

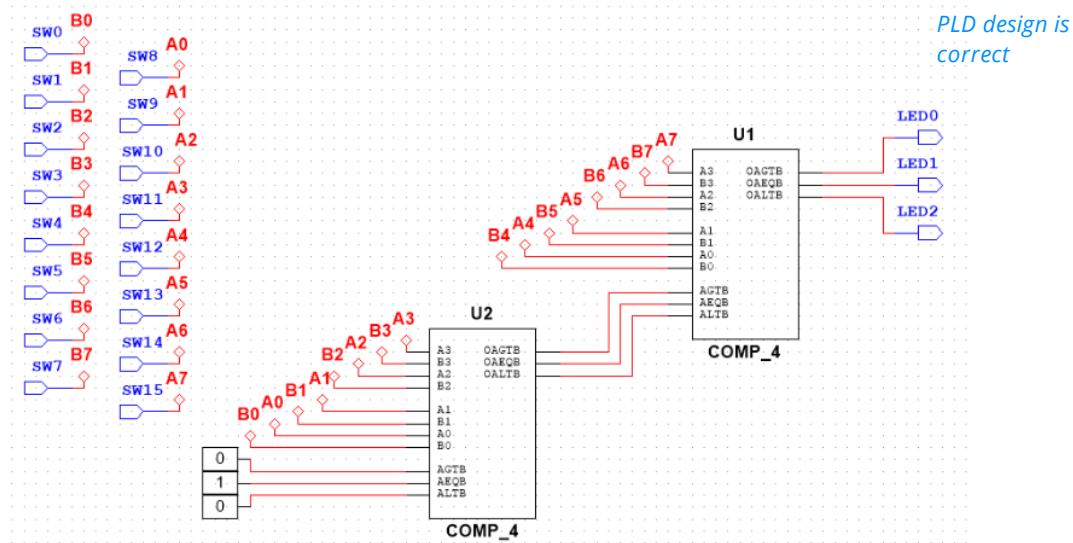
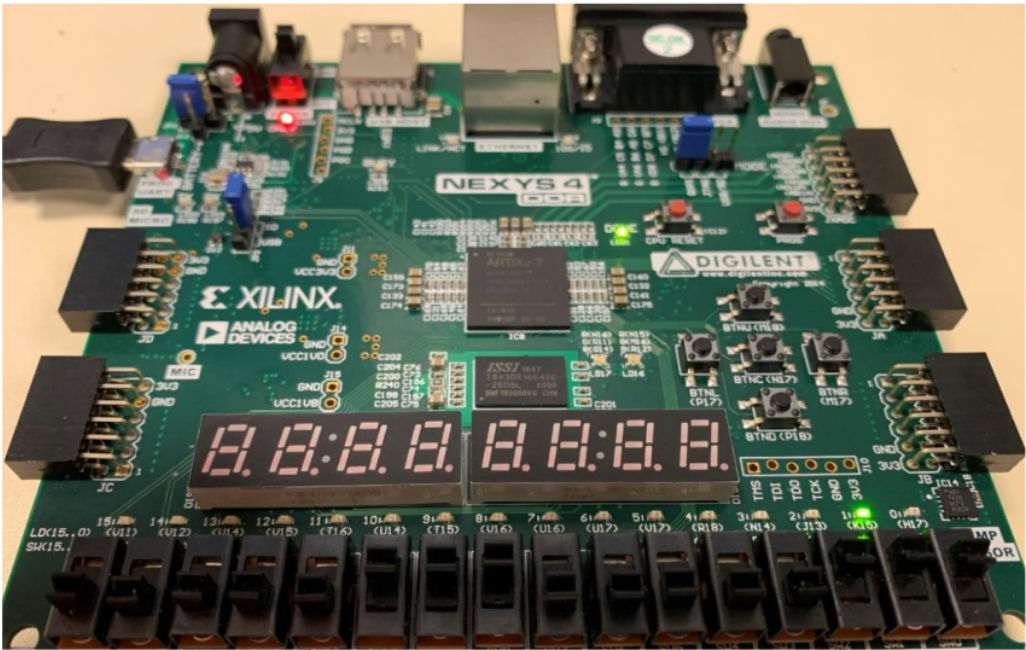


Figure 56 – 8-Bit Magnitude Comparator PLD Circuit

1.5.6 8-Bit Magnitude Comparator test - FPGA.

Test 1 – A Equal to B

Test 1 is “A equal to B”, for this test the switches for set ‘A’ were set to binary 7, in order to achieve the output “AEQB” set ‘B’ was also set to binary 7.



B								A								Status LED		
S15	S14	S13	S12	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0	LED 0	LED 1	LED 2
0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1	Red	Green	Red
Digital Sum = 7								Digital Sum = 7								AGTB	AEQB	ALTB
A is Equal to B																		

Figure 57 – 8-Bit Magnitude Comparator FPGA Test and Evidence - AEQB

As seen in figure 57 set ‘A’ and set ‘B’ both had binary 7 values which drove the output for LED 1 on, proving that this section of the code is operational.

Test 2 – A greater than B

Test 2 is “A greater than B”, for this test the switches for set ‘A’ were set to binary 128, in order to achieve the output “AGTB” set ‘B’ was also set to binary 48.

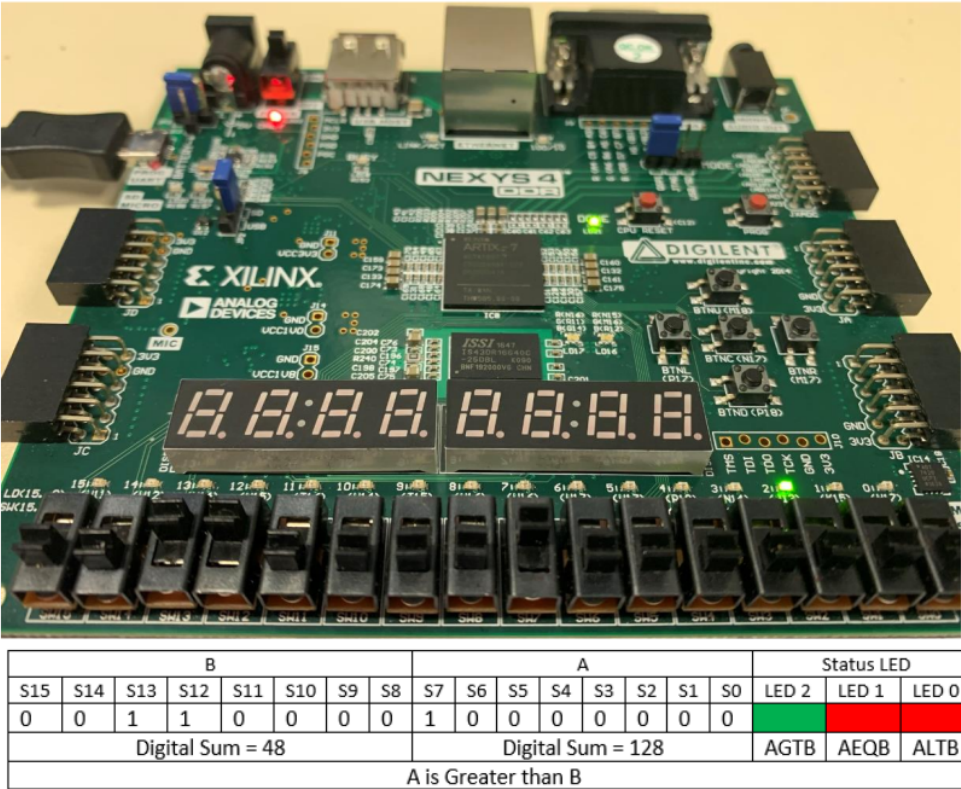
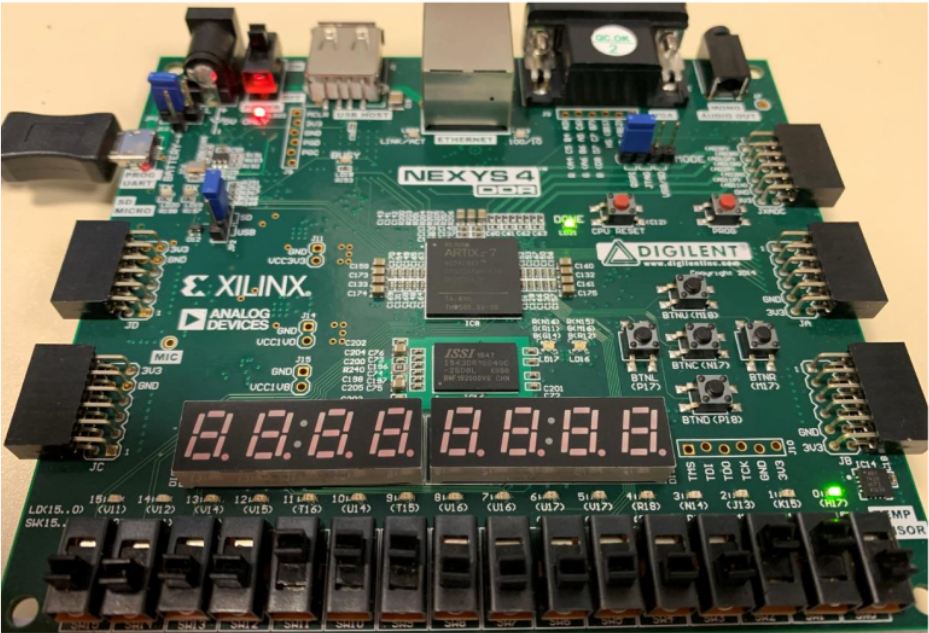


Figure 58 - 8-Bit Magnitude Comparator FPGA Test and Evidence - AGTB

As seen in figure 58 set ‘A’ had a binary value of 128 and set ‘B’ had a value of 48, with this arrangement of values which drove the output for LED 2 on, proving that this section of the code is operational.

Test 3 – A less than B

Test 3 is “A less than B”, for this test the switches for set ‘A’ were set to binary 6, in order to achieve the output “ALTB” set ‘B’ was set to binary 14.



B								A								Status LED		
S15	S14	S13	S12	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0	LED 2	LED 1	LED 0
0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	0			
Digital Sum = 14								Digital Sum = 6								AGTB	AEQB	ALTB
A is less than B																		

Figure 59 - 8-Bit Magnitude Comparator FPGA Test and Evidence - ALT B

As seen in figure 59 set ‘A’ had a binary value of 6 and set ‘B’ had a value of 14, with this arrangement of values which drove the output for LED 0 which is assigned to ALT B on, proving that this section of the code is operational.

In conclusion the outcome of the Multisim, PLD and FPGA tests proves that the 8-Bit adder is functioning as intended, which then concludes that all design and testing for element 1 is successful.

Very good set of results and demonstrates a good working knowledge of the system and its capabilities

1.6 Evaluation

The research undertaken at the start of this element gave an insightful under-pinning knowledge on some of the key principles of this element. Knowledge and understanding of digital magnitude comparators and digital design methods using VHDL and IP integration gave useful information that was built upon during the design and build stage, Further research into ADCs, Signal Conditioning and Load cells have also provided a base where element 2 can be built upon.

Knowledge of logic circuits and truth tables was also built upon from basic stages with the half-adder increasing to more complex circuits such as the 8-bit magnitude comparator.

The half-adder was the first circuit designed using the VHDL language, as this was the simplest circuit it greatly benefited as VHDL is a new programming language. The half adder was designed and simulated within Vivado to prove the circuit is operational, using the principles learnt in the research stage this was then duplicated to create a full adder, the architecture of the half adder was duplicated with the VHDL language within Vivado, successful testing of this circuit was undertaken within Vivado.

A 4-Bit adder was created by cascading four full adders within Vivado, using Vivado's bitstream generation the program was proven to be functional after testing and analysis was performed. This program was then downloaded onto the Nexys FPGA board for physical testing of the functions, after testing and analysing it was proved that the circuit was operational.

Error here. There is no 8 bit adder

Two 4-Bit adders were then cascaded together in order to create an 8-bit adder. An 8bit magnitude comparator was created in Multisim, this was tested within Multisim to ensure correct operation, this was then converted into PLD mode for functional testing on the FPGA board, using the toggle switches on the board binary signals were used to give various state inputs to the board, three outputs were assigned on the board to give outputs for AGTB, ALTB and AEQB, all three states were tested and proved functional.

In evaluation the 8-bit magnitude comparator was successfully designed, built and tested within the Vivado hierarchy, in turn proving that all design, build and testing stages beforehand were successful, throughout this element great knowledge has been learnt on a range of fundamental principles which will be built upon in element 2.

*Some minor
errors in the
evaluation
but overall
there are
good points
made*

2. Element 2

2.1 Introduction

Element 2 is focused around the design and implementation of an electronic circuit with the purpose of converting analogue signals into digital 8-bit signals, these will then be compared in a “greater than”, “equal to” or “less than” format. The research and theory undertaken on load cells in element 1 will be used in the design of a load cell circuit which will provide one of the analogue signals to be compared, this will be compared to a standard 0V-5V signal that can be altered using a potentiometer or slider.

Load cells are widely used in industry to determine how much physical weight is applied to a given area, the resistance change within the load cell will determine the weight that is applied, a typical diagram of a load cell is seen in figure 60.

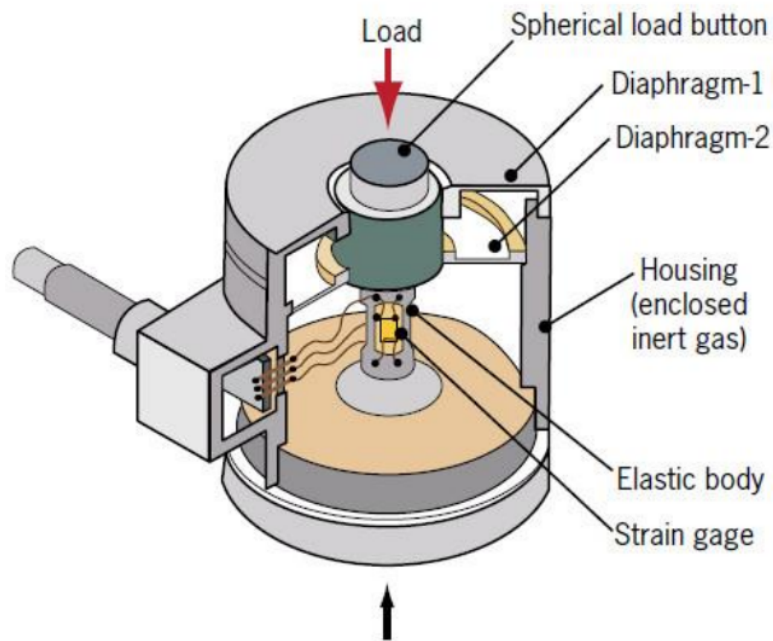


Figure 60 – Load Cell Diagram

2.2 Design Requirements

As per specification, the load cell is required to produce an output of 2mV/V excitation over the full-load applied, in this application it will be between 0 and 1000grams, or 1kg, it will be designed with a supply voltage of 5VDC, the full component specification is seen below in table 1:

Load Cell Component Specification	
Excitation Voltage	5VDC (Max Voltage 7VDC)
Output	2 _{mV} /V
Unstressed Resistance	350Ω
Unloaded Resistance Values	50% on all Resistors (Equal)
Full Loaded Values	Diagonally Unequal, 100% and 0% in opposing corner.

Table 1 – Load Cell Component Specification

If the full load is applied to the load cell it will produce an output of 10mV due to the specification, this is seen in the calculation:

$$5V \times 2mV/V = 10mV$$

The internal circuit inside the load cell is of the Wheatstone bridge form, this operates with an unstressed resistance of 350Ω as seen in the specification table, this circuit operates upon the principle of null deflection, whereby when all resistances are equal (under no load), no current will flow through the circuit. Under normal conditions, when the bridge is loaded and unbalanced current can flow through the galvanometer. The opposing effect occurs when the bridge is in a balanced condition under no load where no current flows through the galvanometer. This condition can be achieved by adjusting the known resistance and variable resistance (byjus, 2021). The load is measured by the resistance difference between resistor pair 1 and resistor pair 2, with this resistance difference creating a potential difference on the output.

In order to achieve the specified output of 0V to 5V signal conditioning has to be implemented. When the load cell is under a small load, the output voltage will be small, therefore, a unity gain amplifier is required. A differential op-amp is also required to measure the potential difference between V1 and V2, these two circuits will create an instrument amplifier required for the circuit's full operation, the Multisim models of these circuits are seen in figure 61.

*Unity gain is to test diff amp only.
Amp needs a high gain for a small
load to range from 0-5V.*

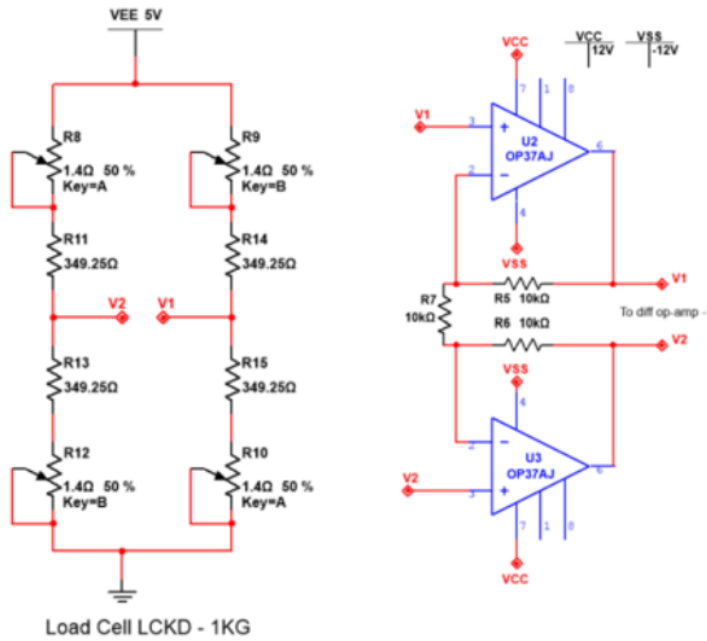


Figure 61 – Loadcell and Voltage Buffer Multisim Models

The gain for the op-amp is calculated using the equation:

$$Op - amp Gain = 1 + \frac{R_5 + R_6}{R_7}$$

In this case:

$$Op - amp Gain = 1 + \frac{20K + 20K}{10K} = 5$$

Due to the gain value of 5, it is not possible for the op-amp to operate at unity gain, as R_1 and R_2 are of equal resistance, it holds the gain of each op-amp at the same value, in turn holding the voltage difference between V1 and V2 equal. The unity gain differential op-amp is seen in figure 62.

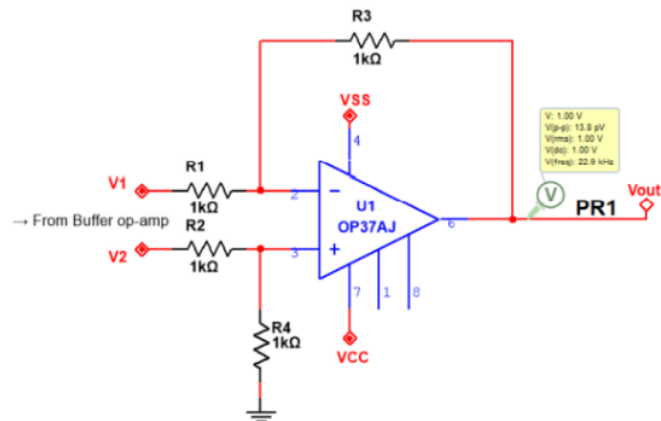


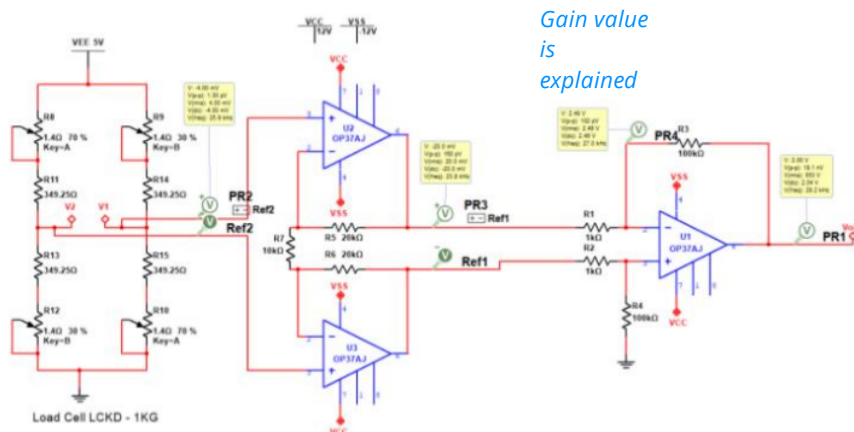
Figure 62 – Unity gain differential op-amp

The differential amplifier gain was then calculated using the equation:

$$\text{Differential Amplifier Gain} = \frac{10K}{100} = 100$$

The unity gain differential op-amp can be connected to the load cell circuit and the buffer circuit, in order to create the full circuit. In order to confirm that the correct calculations have been derived, as the load cell produces a V_{OUT} of 10mV, a gain of 500 is required to meet the specification of 5V, in order to meet the specification, the instrument amplifier is designed to have a gain of 500 as $5V/10_mV = 500$. This gain is achieved by multiplying the buffer gain and differential amplifier gain together, this being $5 \times 100 = 500$.

In order for the voltage buffer to have a gain of 5, the parallel resistors R_5 and R_6 are increased in value by a factor of 2, each of the three individual circuits can then be connected to produce the overall circuit as seen in figure 63.



Gain value
is
explained

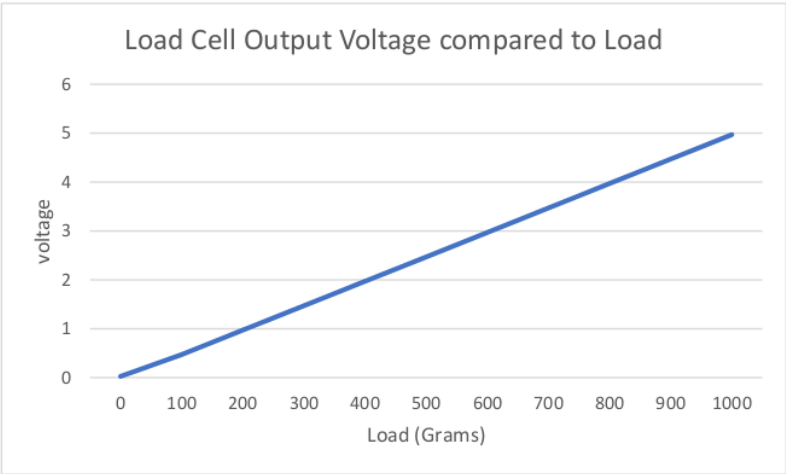
Figure 63 – Multisim model of the Loadcell, Voltage Buffer and Unity gain differential op-amp circuit

The system was then tested to prove its operational reliability, the test was carried out by increasing the load from 0 to 1000 grams in 100-gram steps, this is performed by altering the variable resistance values by 5%, with the top-left and bottom-right resistors being increased by this rate and the opposing resistors were reduced by 5%. The evidence of the Multisim circuit operating is seen in appendix 6.1 with the results table seen in table 2.

Comparison Table for Load Cell Operation in Multisim		
% Resistance Balance	Load Cell Load (Grams)	Differential op-amp output (Voltage)
50 Ω - 50 Ω	0	0.026
55 Ω -45 Ω	100	0.473
60 Ω -40 Ω	200	0.973
65 Ω -35 Ω	300	1.47
70 Ω -30 Ω	400	1.97
75 Ω -25 Ω	500	2.47
80 Ω -20 Ω	600	2.97
85 Ω -15 Ω	700	3.47
90 Ω -10 Ω	800	3.97
95 Ω -5 Ω	900	4.47
100 Ω -0 Ω	1000	4.97

Table 2 - Comparison Table for Load Cell Operation in Multisim

It is clearly seen in table 2 that the differential op-amp voltage starts off at 0.026V, this is due to the leakage voltage within the circuit being amplified by a large gain, therefore this is the systems 0 as expected at a load of 0 grams. The circuit is proved to be operational as a linear response of output voltage against the increase in load, a voltage increases of 0.5V per 100 grams is seen, with a full load voltage seen at 4.97V at 1000 grams as expected.



Linear graph is correct

Figure 64 – Linear response of output voltage vs load

A graph was plotted in excel to visually show the linear response of voltage against load, which is seen by the steadily increasing gradient, this is expected for this system. This shows that the signal conditioning is correct and the gain has made the scale of 0 – 5V and proves the scale to be linear which satisfies the specification, this is seen in figure 64.

2.3 ADC Prototypes

Analog to Digital converter prototypes were then designed using the LabVIEW package. These are designed so that the analogue signals are referred to a constant voltage ranging from 0 to 5VDC. The analogue conversion for this is centred around the input voltage which is set using a slider within the LabVIEW GUI, showing a simple but effective application of an ADC.

2.3.1 10 Bit ADC Prototype

Figure 65 shows the block diagram view of the LabVIEW program, this application uses block diagram programming, this application has various different block diagrams with different functions, as seen in figure 65 it shows the block diagram for the 10-Bit ADC, this system uses Inputs and Outputs like most other logic controllers.

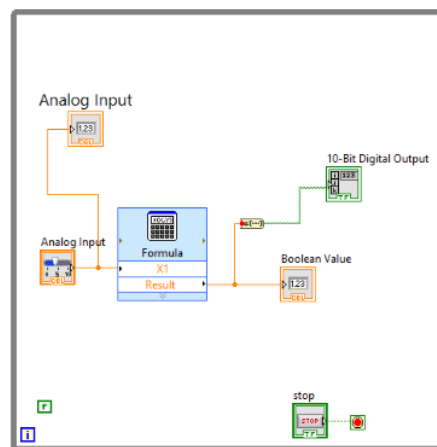


Figure 65 – 10-Bit ADC Block Diagram

The block diagrams are then implemented on the “panel view”, this is LabVIEW’s graphical user interface, much like HMI screen seen in PLCs, this is seen in figure 66.

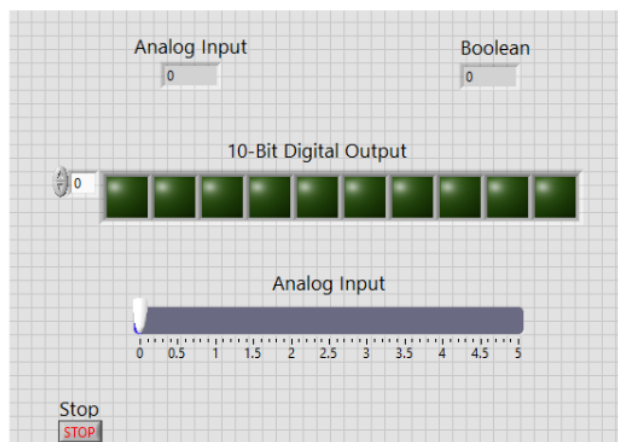


Figure 66 – 10-Bit ADC Panel View

The formula block in the block diagram panel was then set up for the 10-Bit ADC, this is done using the equation:

$$\frac{X1}{Max V_{ref}} \times 2^n$$

Here, $Max V_{ref}$ is the maximum scale of the ADC, this is 5V for this application. 2^n is used to calculate the number of bits. For this system the equation is set up as seen below:

$$\frac{X1}{5} \times 2^{10}$$

This equation was then entered in the formula configuration block, as seen in figure 67.

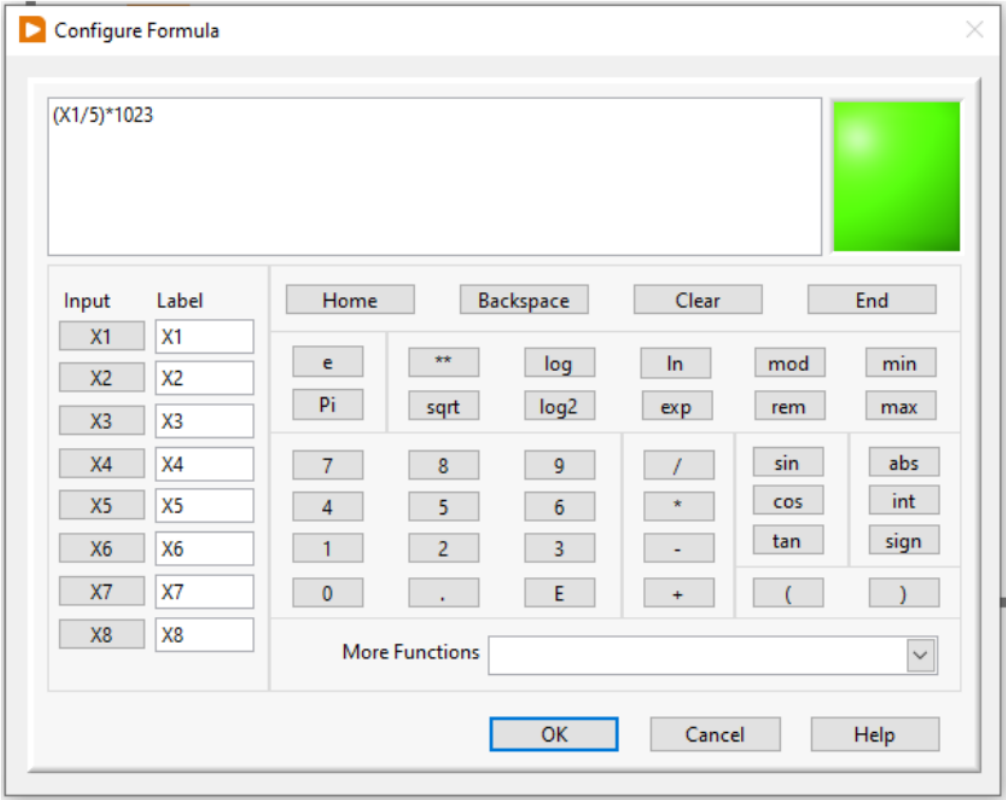


Figure 67 – 10-Bit Formula Block Equation

As both the block diagram and panel view are configured, the system can be tested with various values to prove its functionality. The analog input was set at 2.5V out of 5V, as this is half of the maximum input value, it is expected that the output will be half of the maximum output of 1023. This is mathematically proven using the equation seen below:

$$(2.5/5) * 1023 = 511.5$$

Therefore, it is predicted the Boolean output will display 511.5 with the 10-bit digital output displaying a response like table 3. Within LabVIEW, 511.5 will be rounded up to the nearest value of 512.

1	2	4	8	16	32	64	128	256	512
0	0	0	0	0	0	0	0	0	1

Table 3 – Expected Boolean output of decimal 512

As seen in figure 68, the LabVIEW response has simulated an identical response, proving the reliability and accuracy of the program functionality.

Good test
result for 10
bit ADC

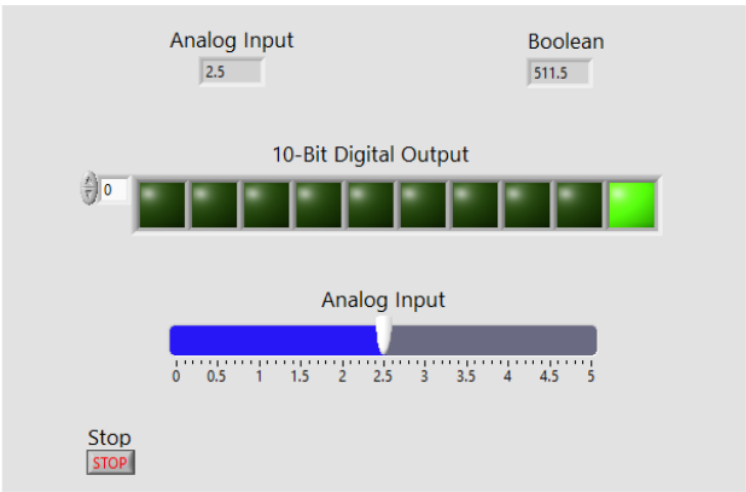


Figure 68 – 10-bit Boolean Output at 512 (2.5V)

The system was then tested again at an analog input of 5V. Using the same calculation used in the first test, the 10-Bit output and Boolean value can be predicted:

$$(5/5) * 1023 = 1023$$

Therefore, it is predicted the Boolean output will display 1023 with the 10-bit digital output displaying a response like table 4. Within LabVIEW, as this value is a whole value that would be expected of this system it does not need to be rounded.

1	2	4	8	16	32	64	128	256	512
1	1	1	1	1	1	1	1	1	1

Table 4 - Expected Boolean output of decimal 1023

As seen in figure 69, the LabVIEW response has simulated an identical response, further proving the reliability and accuracy of the program functionality.

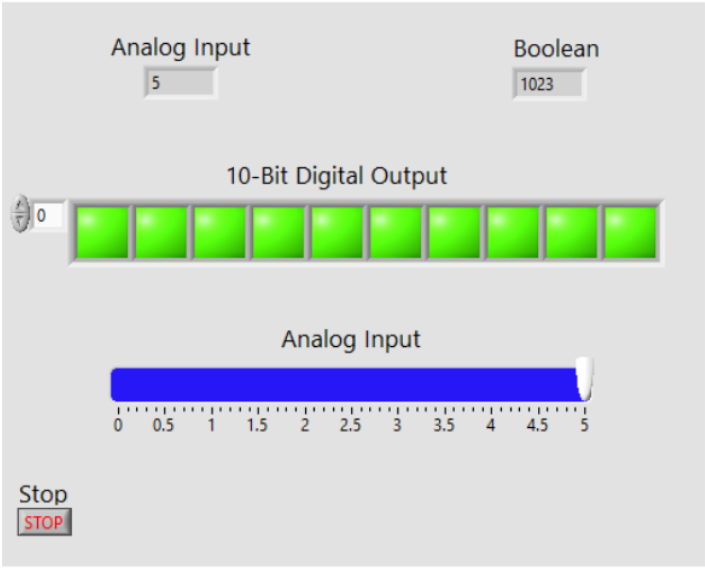


Figure 69 – 10-bit Boolean Output at 1023 (5V)

Test using front panel view shows the correct operation

2.3.2 12 Bit ADC Prototype

The system was then upgraded to perform as a 12-bit ADC. Two extra LEDs were added to display a 12-Bit digital output and the formula block equation was modified for the 12-Bit ADC; these modifications are seen in figure 70; this equation is seen below:

$$\frac{X1}{5} \times 2^{12} = 4096$$

As seen in the equation above, there is a maximum Boolean output of 4096 - 1 for the 12-Bit ADC this leaves a maximum Boolean output of 4095.

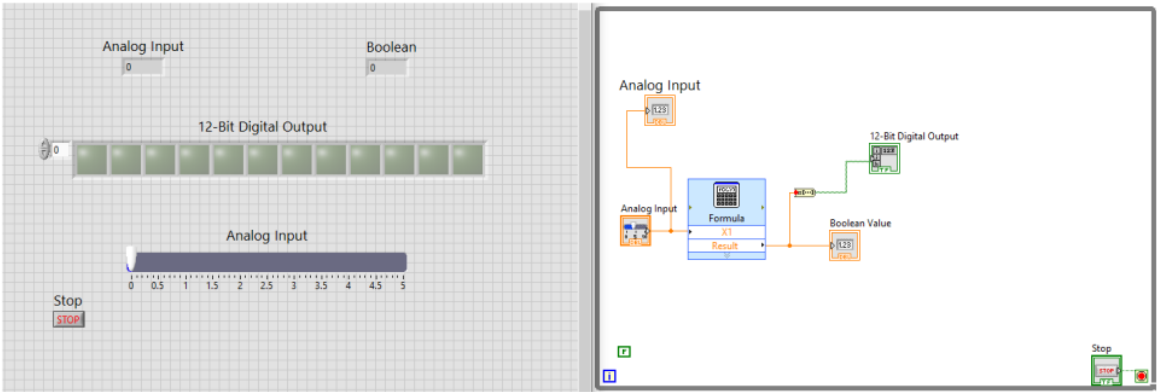


Figure 70 – 12-Bit ADC Prototype Block Diagram and Front Panel

The same tests as the 10-Bit ADC can be replicated, the only difference will be the larger Boolean value and 2 extra bits. The system was tested firstly at an analog input of 2.5V. Using the same calculation used in the 10-bit test using a value of 4095 instead of 1023. The 12-Bit output and Boolean value can be predicted:

$$(2.5/5) * 4095 = 2048$$

Therefore, it is predicted the Boolean output will display 2048 with the 12-bit digital output displaying a response like table 5. Within LabVIEW, as this value is a whole value that would be expected of this system it does not need to be rounded.

1	2	4	8	16	32	64	128	256	512	1024	2048
0	0	0	0	0	0	0	0	0	0	0	1

Table 5 - Expected Boolean output of decimal 2048

As seen in figure 71, the LabVIEW response has simulated an identical response in terms of the Boolean output LEDs when compared to table 5, the Boolean value shows a value of 2047.5 as predicted. LabVIEW has rounded this value up to 2048 for the 12-bit digital output LEDs as expected.

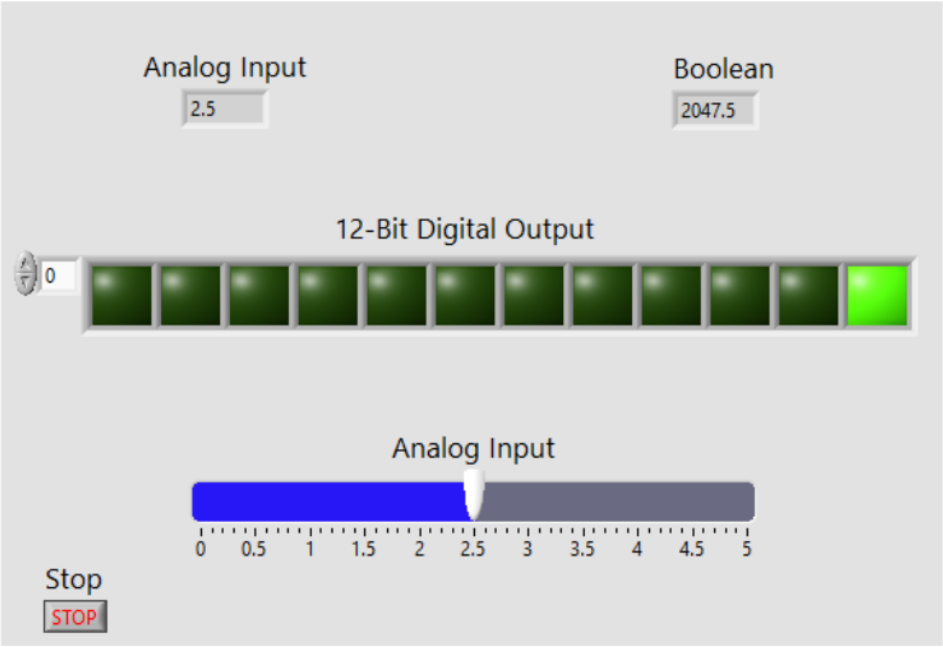


Figure 71 – 12-bit Boolean output at 2048 (2.5V)

Test is correct showing the scale at half way point

This test is then repeated for another value, this time 5 was chosen, from this it is expected to have the largest Boolean output of 4096, as confirmed by the equation below:

$$(5/5) * 4096 = 4096$$

Therefore, it is predicted the Boolean output will display 4096 with the 12-bit digital output displaying a response like table 6. Within LabVIEW, as this value is a whole value that would be expected of this system it does not need to be rounded.

1	2	4	8	16	32	64	128	256	512	1024	2048
1	1	1	1	1	1	1	1	1	1	1	1

Table 6 - Expected Boolean output of decimal 4096

As seen in figure 72, the LabVIEW response has simulated an identical response in terms of the Boolean output LEDs when compared to table 6.

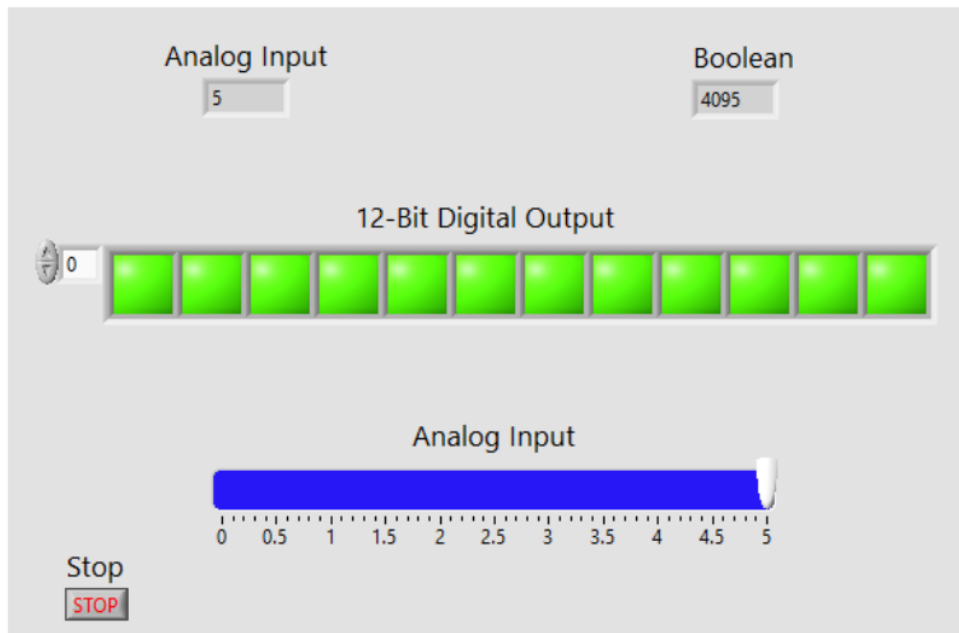


Figure 72 - 12-bit Boolean output at 4096 (5V)

Good test at full scale with limits set to prevent going over the max value

2.3.3 8 Bit ADC Prototype

An 8-Bit ADC was then designed in LabVIEW, as seen by figure 73 which displays the block diagram of the system. This system has the ability to take an analog input from the load cell through the instrument amplifier seen in figure 73. The output from the 8-Bit ADC will drive signals to a magnitude comparator, this will compare values to the reference value to the ADC output and will produce an “equal to”, “less than” or “greater than” response depending on these values.

The analog voltage enters the system at Dev1/ai0 and has a range between 0 and 5V, this is read by a data read block which reads the voltage level, this is then fed to both the analog input display which is displayed on the GUI and to the formula block. The formula block completes a calculation with the analog voltage input as its variable. The output of the formula block is fed into result blocks which will display the results on the GUI. There are also blocks seen for the errors

As this is an 8-Bit system, the equation used in the formula block is seen below:

$$\frac{X1}{5} \times 2^8$$

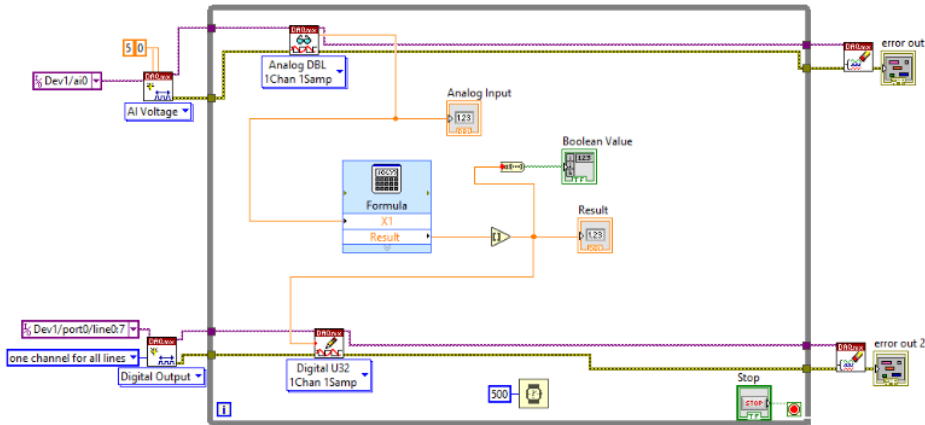


Figure 73 – 8-bit ADC Block Diagram

Figure 74 shows that this has been entered into the formula block, as $2^8 - 1 = 255$

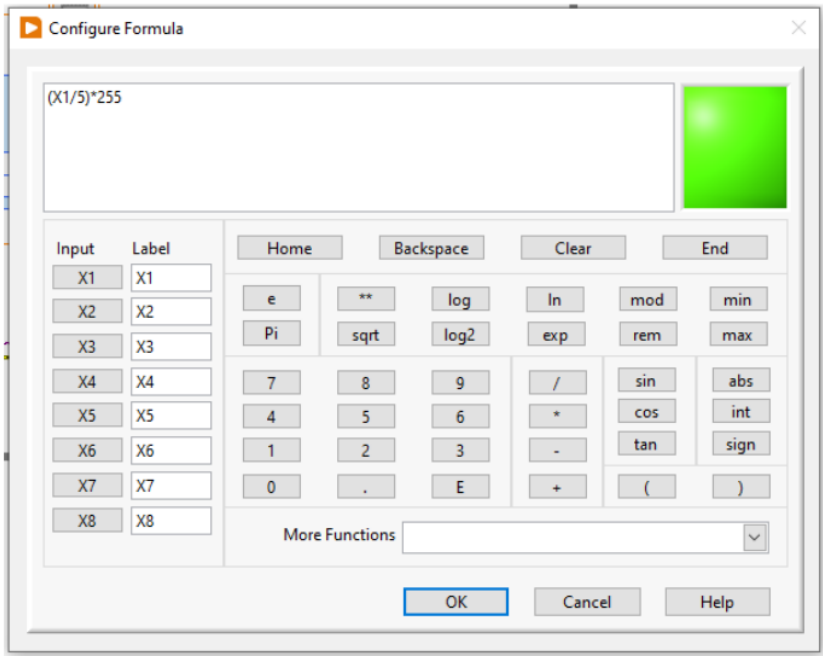


Figure 74 – 8-bit ADC Formula Block

The circuit can then be simulated in LabVIEW, the analog input is set using the variable power supply, this was tested with three different analog inputs to prove the functionality. The first input signal was set at 1V, the predicted value Boolean value is calculated below:

$$(1/5) * 255 = 51$$

From this calculated value, it is predicted that the binary output LEDs will display the same response as table 7.

1	2	4	8	16	32	64	128
1	1	0	0	1	1	0	0

Table 7 - Expected Boolean output of decimal 51

Figure 75 displays the result of the block diagram applied with a 1V signal, a minor error of 830μA is seen which is negligible for the reconverted signal. It is clearly seen that the LED response is identical to the predicted response seen in table 7. It is also seen that the Boolean result of 51 matches the calculated value proving the systems accuracy.

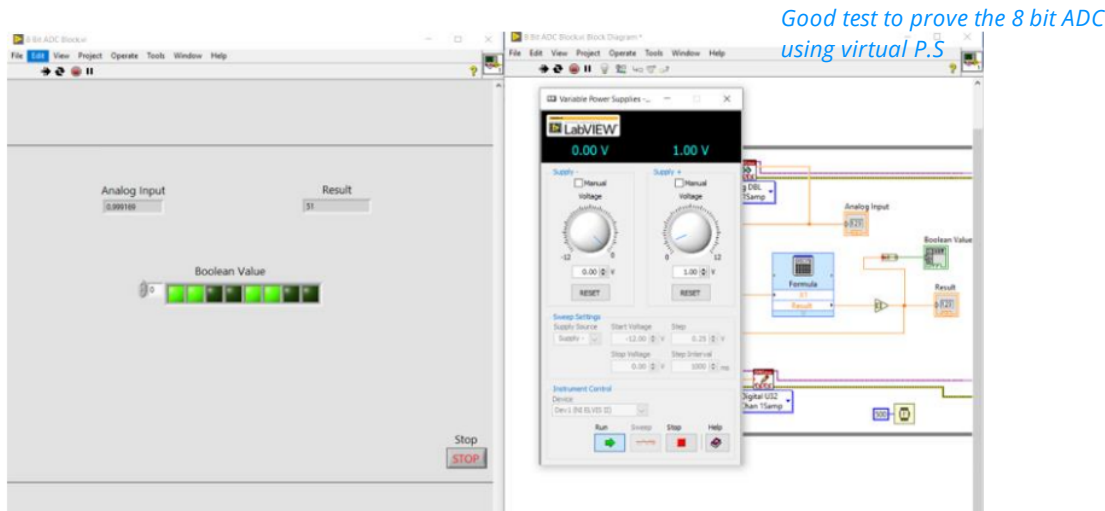


Figure 75 - 8-bit Boolean output at 51 (1V)

The system was then tested again with a different value, 2.5V was used. As this is half of the maximum voltage it is expected that the Boolean result will have a decimal value of 127.5, however, LabVIEW will round this figure up or down. This value is inserted into the equation below to confirm the prediction:

$$(2.5/5) * 255 = 127.5$$

From this calculated value, it is predicted that the binary output LEDs will display the same response as table 8, therefore it is expected that LabVIEW will round the value down.

1	2	4	8	16	32	64	128
1	1	1	1	1	1	1	0

Table 8 - Expected Boolean output of decimal 127

Figure 76 displays the result of the block diagram applied with a 2.5V signal, a minor error of 5.5mA is seen which is negligible for the reconverted signal. It is clearly seen that the LED response is identical to the predicted response seen in table 8. It is also seen that the Boolean result of 127 matches the predicted and calculated values proving the systems accuracy.

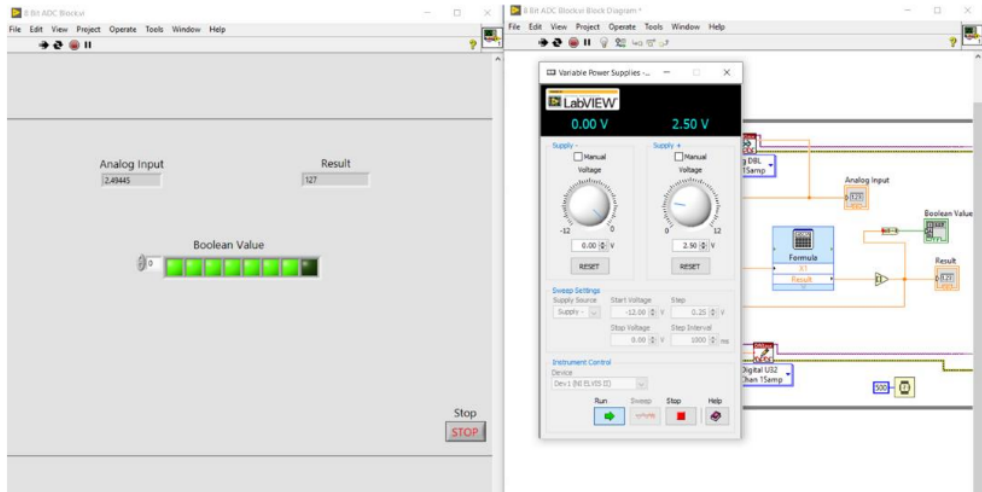


Figure 76 - 8-bit Boolean output at 127 (2.5V)

The final simulated test for the 8-bit ADC was performed at an input value of 4.99V as the variable power supply couldn't not settle at 5V, as this is the upper-limit of the input voltage it is expected that a decimal value of 255 will be produced. This is confirmed by the calculation seen below:

$$(4.99/5) * 255 = 254.5$$

The 0.01 error has caused the predicted decimal value to fall by 0.5, this error will be ignored as LabVIEW will round this up to the nearest Boolean value at 255. The predicted response is seen in table 9.

1	2	4	8	16	32	64	128
1	1	1	1	1	1	1	1

Table 9 - Expected Boolean output of decimal 255

The results of the block diagram being applied a voltage of 4.99V is seen in figure 77, it is clearly seen that the predicted decimal value of 255 has been produced by LabVIEW, along with the LEDs providing a binary signal that matches the predicted response in table 9.

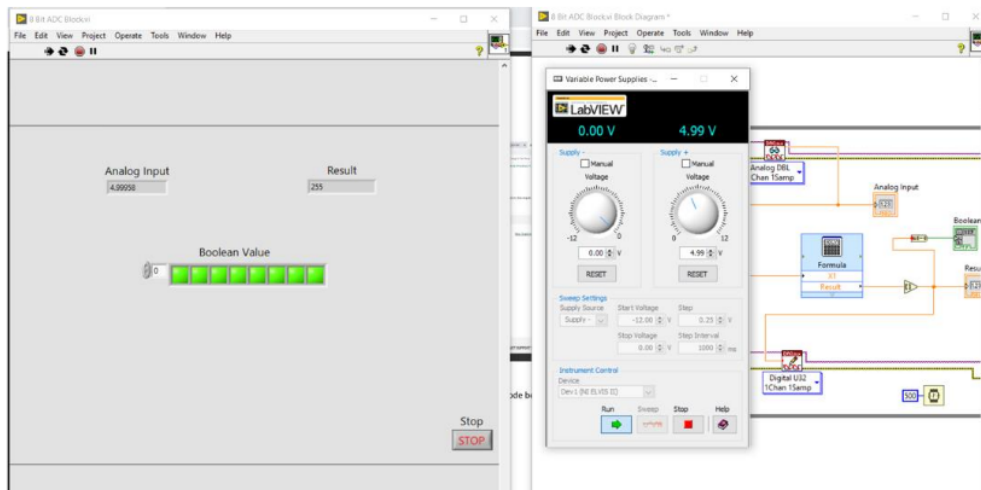


Figure 77 - 8-bit Boolean output at 255 (4.99V)

A good range of tests including max value test

2.4 8 Bit Comparator Implementation

The 8-Bit magnitude comparator designed in element 1 was then re-configured, this was modified to add connectors for Digilent's Peripheral Module Interface, commonly known as "Pmod Interface", this is an open standard developed by Digilent to allow for Multisim to be used with their Nexys 4 DDR FPGA board. The reconfigured circuit is seen in figure 78. Input set A are connected to the JA connector, Input set B are connected to the JB connector, with the three comparator outputs being connected to connector JC. The Pmod pin assignments are seen in figure 79.

8-BIT MAGNITUDE COMPARATOR

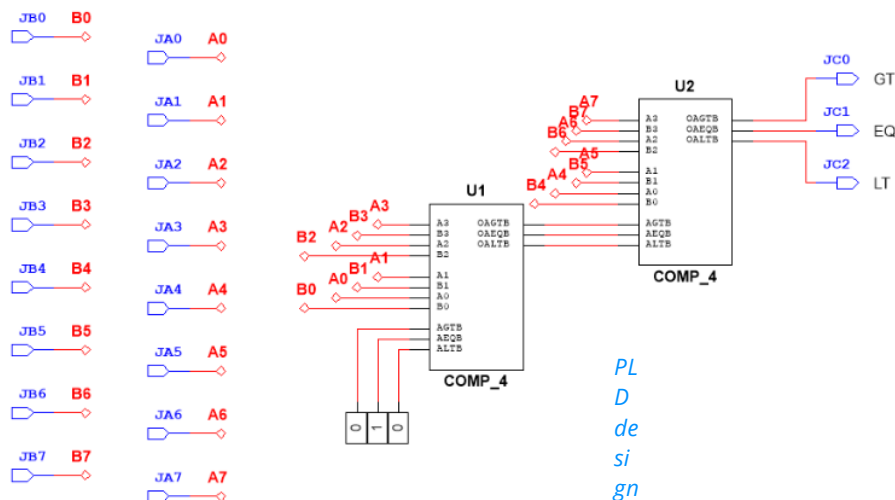


Figure 78 – 8-Bit Magnitude Comparator Multisim Model

PLD design is correct and p



Table 5. Nexys4 DDR Pmod pin assignments.

Pmod JA	Pmod JB	Pmod JC	Pmod JD	Pmod XDAC
JA1: C17	JB1: D14	JC1: K1	JD1: H4	JXADC1: A13 (AD3P)
JA2: D18	JB2: F16	JC2: F6	JD2: H1	JXADC2: A15 (AD10P)
JA3: E18	JB3: G16	JC3: J2	JD3: G1	JXADC3: B16 (AD2P)
JA4: G17	JB4: H14	JC4: G6	JD4: G3	JXADC4: B18 (AD11P)
JA7: D17	JB7: E16	JC7: E7	JD7: H2	JXADC7: A14 (AD3N)
JA8: E17	JB8: F13	JC8: J3	JD8: G4	JXADC8: A16 (AD10N)
JA9: F18	JB9: G13	JC9: J4	JD9: G2	JXADC9: B17 (AD2N)
JA10: G18	JB10: H16	JC10: E6	JD10: F3	JXADC10: A18 (AD11N)

Figure 79 – Pmod Pin Assignments of the Nexys 4 DDR

2.4.1 Magnitude Comparator Setup

The Nexys 4DDR board was then set up onto the NI Elvis II board, the signals from the Nexys board are wired into the Elvis II. The Nexys 4DDR contains the magnitude comparator code. The potentiometer is used to set the scaled value, this set up is seen in figure 80.



Good image of the setup

Figure 80 – Nexys 4 DDR connected to the NI Elvis II Board

2.5 Overall LabVIEW Program

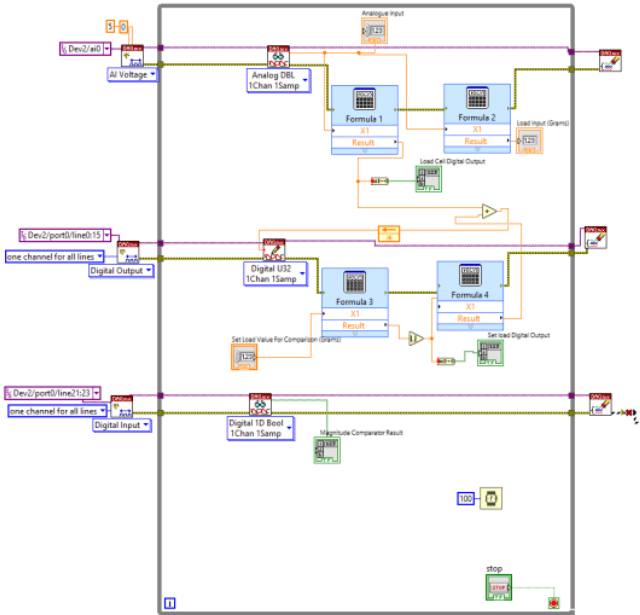
Good use of table for formula blocks

The final LabVIEW program designed is seen in figure 81, this system has been built-up by making additions to the 8-Bit ADC system. This system now contains four formula blocks which are required for the systems operation, these are explained in table 10.

Explanation of Formula Blocks		
Formula Blocks	Block Equation	Equation Function
Formula Block 1	$(X1/5) * 256$	This formula block converts the analog input signal into a digital value. The input voltage is divided by the maximum input voltage, this is then multiplied by 256 to obtain an 8-bit signal representing the analogue voltage input.
Formula Block 2	$(X1/5) * 1000$	Converts the analog voltage input into a weight value in grams as a digital signal. The analog voltage is divided by the maximum input voltage and is then multiplied by 1000 to obtain the grams value.
Formula Block 3	$(X1/1000) * 256$	Converts the reference setpoint into a digital value from grams. The set value in grams is divided by the max gram value of 1000 and multiplied by 256 to produce an 8-bit signal.
Formula Block 4	$X1*256$	Converts an 8-Bit signal into 16-Bit, the Boolean value is multiplied by 256 to produce this.

Table 10 – Explanation of System Formula Blocks

Other changes include the digital output block has been altered to allow for the use of up to 16 outputs, this is required by the system as two bytes of data are compared; therefore 16 bits are required. An extra digital input block is added to read the outputs from the 8-Bit magnitude comparator for 'equal to', 'greater than' and 'less than' signals.



G-code is correct

Figure 81 – Final LabVIEW Block Diagram

Figure 82 shows the front panel design for the final LabVIEW program.

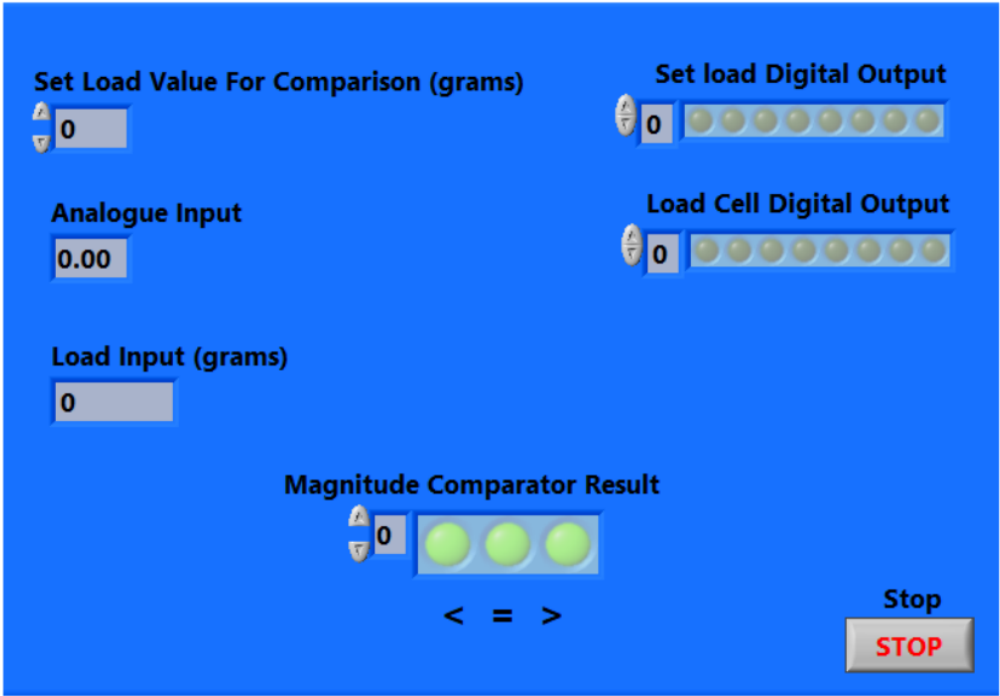


Figure 82 – Panel View of final LabVIEW program

Good design of front panel with arrays included

2.6 Testing and Analysis

As the GUI has been configured, the final testing section of element 2 is undertaken. This system was configured so that the 16-Bit words are sent from the NI ELVIS II board to the Nexys 4 DDR board, here the signals are read and processed then transmitted back as a 3-Bit signal. For this test, the potentiometer will be altered to produce three values, one equal to the set load value, one less than the set load value and the final greater than the set load value. The three LEDs seen labelled “Magnitude Comparator Result” are assigned to a different condition, the left-hand being less than, middle being equal to and the right-hand being greater than.

The first test that was performed is the less than test, here the load input was set to lower than the set load value of 400 grams. As seen in figure 83, when the load input was set to a lower value than the set load it shows that the magnitude comparator’s ‘less than’ LED is illuminated

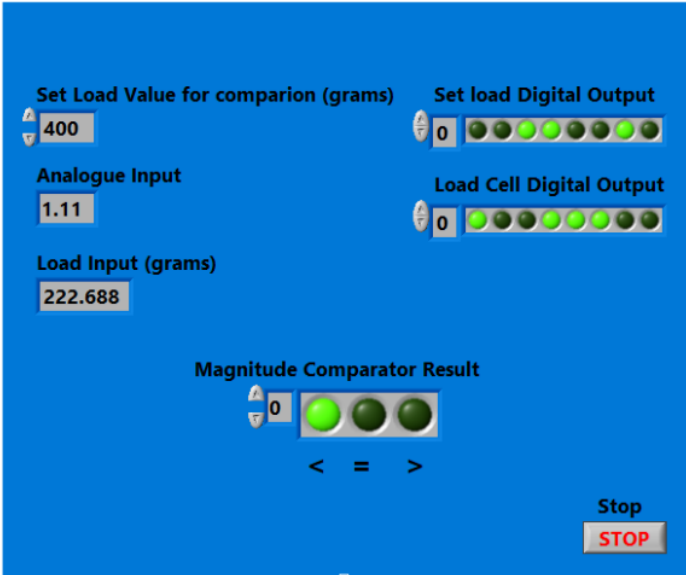


Figure 83 – LabVIEW Magnitude Comparator Test – Less than result

For the second test, the load input was set to the same value as the set load value for comparison, here it is seen that that magnitude comparator result is in the ‘equal to’ condition, this is seen in figure 84. There is some error on the load input of around 12.5%, this is seen as the set-load value is set at 400 grams and the load input is set at 412.484 grams and the equal to LED is illuminated. This error is due to the ADCs resolution, if this was increased to 10 or 12-Bit it would reduce this error as there will be far more quantization levels in turn reducing the amplitude between each quantization step producing a more accurate response.

Exp
lan
ati
onA
doe
s I
not
exp
lai h
n e
hor
w p
theo
bits
are
chif

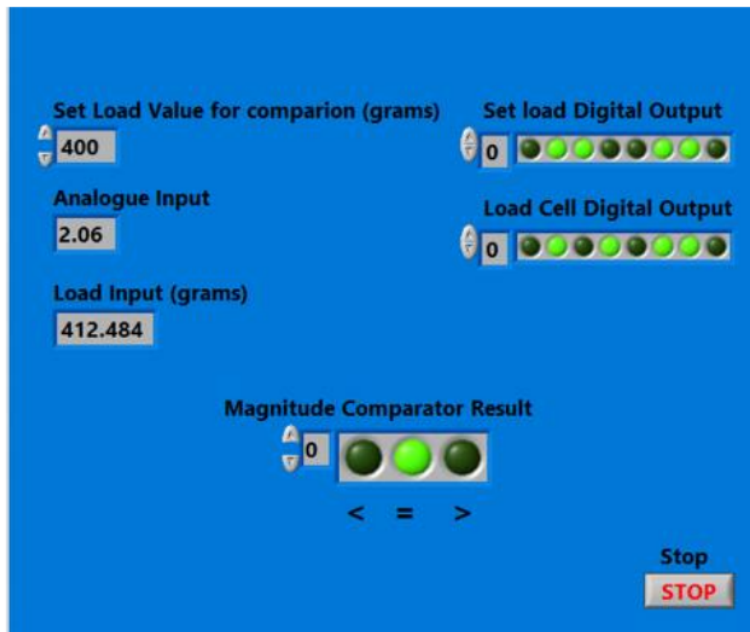


Figure 84 - LabVIEW Magnitude Comparator Test – Equal to result

The final test was to set the potentiometer to set a signal that is greater than the load value for comparison. It is seen that that load input was set to 989 grams, this is far greater than the set load value of 400 grams, therefore the correct condition is active, this seen in figure 85.

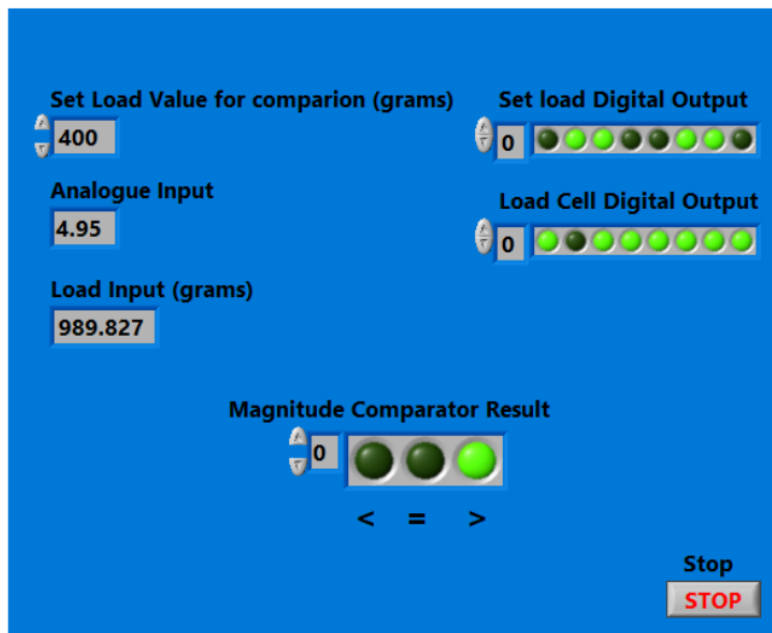


Figure 85 - LabVIEW Magnitude Comparator Test – Greater than result

Tests indicate correct operation of system

2.7 Evaluation

In evaluation, Element 2 has proved to be highly successful. The calculations and Multisim design of the instrumentation amplifier was highly successful as it operated to specification producing a linear response as the load cell is excited with increasing weights, this is seen in the linear response graph in figure 64, this proved the accuracy of the signal conditioning and that the ADC range was correctly configured with a maximum voltage of 5V.

Extensive computational skills were developed in LabVIEW, from designing the block diagrams for the 12-Bit, 10-Bit and 8-Bit ADC prototypes, from this it was proved that the LabVIEW program had the capability to take an analog input and condition it converting it into a binary and decimal value. Knowledge was also obtained on the design of GUIs with the design of the front panel within LabVIEW. The formula blocks were found to be a highly useful method of converting the analog signals into binary signals to be easily displayed on the front panel.

The useability of LabVIEW was also proven through the ease of altering pre-existing programs to perform new tasks. This was seen as the 12-bit prototype program was designed, this was then modified to produce the 10-bit, 8-bit and final overall circuit. Multisim's functionality was also seen, using the Pmod interface it was easy to connect between the Nexys 4 DDR board and the NI Elvis II board, allowing for seamless operation, Multisim also performed well with the design of the magnitude comparator which combined two bytes of data to be transmitted to the Nexys board.

Minor error here. LabVIEW combines the 16 bit word for the

The virtual analogue signal had minor inaccuracies due to the ADC resolution at 8-bit, this is due to the larger quantization levels present at 8-bit, meaning the amplitude between each step is larger, turning producing more error. As a recommendation, the ADC could be upgraded to 10-bit or 12-bit, reducing the quantization error by producing more quantization levels at lower amplitudes.

However, the final test of the integration between LabVIEW and Multisim was highly successful, operating as expected. Proving that using LabVIEW has the capability to produce scaled digital outputs from using input signals and conditioning programs. As a further recommendation a load cell could be constructed based upon the Multisim design. All communications between the boards were flawless with no issues raised

Evaluation covers good points but there are some confusion

issues with the role of Msim and LabVIEW.

3. Element 3

The previous program from Element 2 was then modified so it is functional to have the capability of recording and storing data, the modifications consisted of the addition of data recording blocks, these have been added to record 5 values in the system.

The time line and load value in grams are logged using block 1, this operates for the whole test analysis period, Block 2 is utilised by logging the test value and blocks 3 to 5 scale and log the comparator condition values. *Good explanation*

The values that are recorded are the “Load Input (grams)”, “Set Load Comparison (grams)”, “Load Input Under set load”, “Load Input Over set load” and “Load input equal to set load”. The comparator value is recorded through these reading blocks, it reads the binary signals at its input terminal and multiplies this value by 200, this helps to differentiate the less than, equal to and greater than traces from the analogue input and set-point values on the data log. This circuit will operate by producing the data in text files which will be imported into excel to produce graphs. *Good explanation*

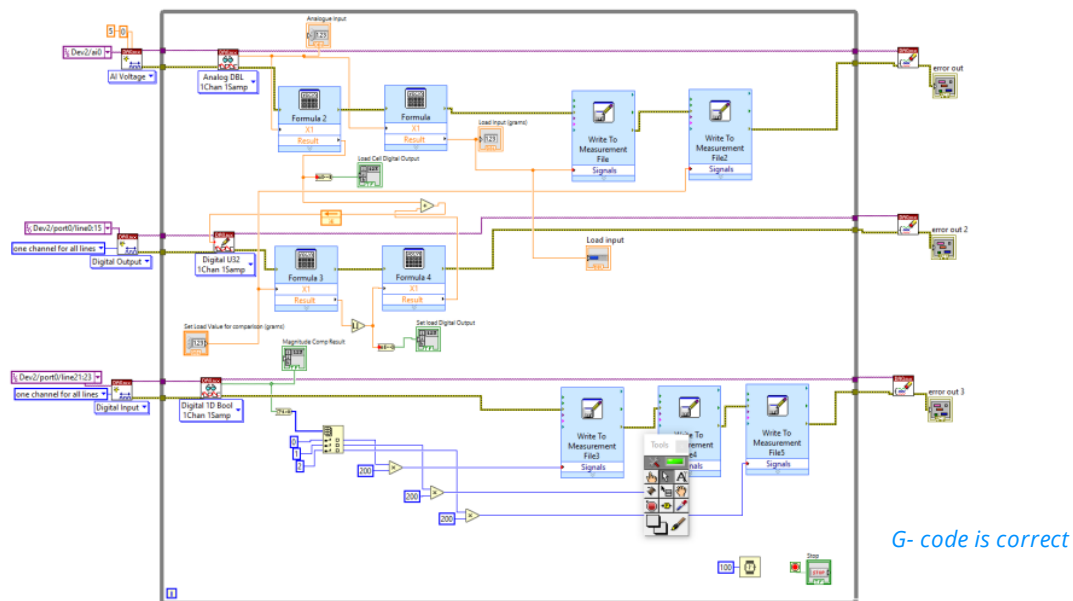


Figure 86 – Final Circuit

Figure 87 displays the configuration file for one of the “Configure Write to Measurement” files, this is configured so that the file format is a LabVIEW Measurement file (LVM), this is required as it allows for data to be converted from the LVM text file into an excel spreadsheet. The configuration settings are common to each block, other than the analogue input block which requires its X Value (Time) Column to be set to “one column only” rather than “empty time column” like the other write files are set to. All other configuration files are seen in appendix 6.2.

Configure Write To Measurement File [Write To Measurement File]

Filename
H:\LabVIEW Data\test1.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

Figure 87 – Write to Measurement file configuration

Good example of using the write text file configuration

3.1 Testing

The circuit seen in figure 86 was then tested to prove its functionality. Its functionality was tested over three set-points, the set-points chosen for testing were at 200g, 500g and 700g. For these tests, the magnitude comparator's function will be tested by comparing an analogue input value to a set-value. From this, evidence of functionality will be depicted through images of the set-point log excel graph which has been produced showing LabVIEW's data logging functions and through the magnitude comparator operation within LabVIEW's front panel.

3.1.1 200g Test

The first test undertaken was at 200g, figure 88 displays the trend graph over a range of 16 seconds, this was produced in excel, whereas figure 89 displays the magnitude comparator results from LabVIEW's front panel.

As seen in figure 88, up until 6 seconds the "Less Than" trendline is in its high state, this is due to the comparator deducing that the analogue input (orange trendline) is a lesser value than the set-point. At 6 seconds it is seen that the yellow "Less Than" trendline falls to 0 and the blue "Equal To" trendline settles at 200g, this has occurred as the set-point is now equal to the analogue input. At around 11 seconds the analogue input has increased to a greater value than the set-point, this has caused the blue "equal to" trend line to fall and the green "Greater Than" trendline to reach its high state. This graph proves the functionality of not only the circuit, but also LabVIEW's ability to export data to other software packages.



Figure 88 – Trendline graph for 200g

*Good test
for 200g*

Figure 89 displays the magnitude comparator results for the 200g test, the “Set Load Value for comparison (grams)” is always set at 200g for this test. When the load input was set at 143.005, the magnitude comparator registered that this is less than the set-point and produced a “less-than” output indicated by the left-hand LED illuminating, it is also seen that the “set-load” binary counter is displaying a larger output than the “load cell digital” counter.

The load input was then increased to 200g, the magnitude comparator indicated that this is equal to the set-point as the centre LED illuminated as expected for an “equal to” condition, here, the binary counters match which also indicates an equal-to signal.

The final 200g test was to increase the load input to a value greater than the set-point, this was increased to 356.402g and it is clearly seen that the magnitude comparator has registered that it is a “greater than” value as the “greater than” LED is illuminated, here the “load cell digital” counter displays a greater binary value than the “set load” counter.

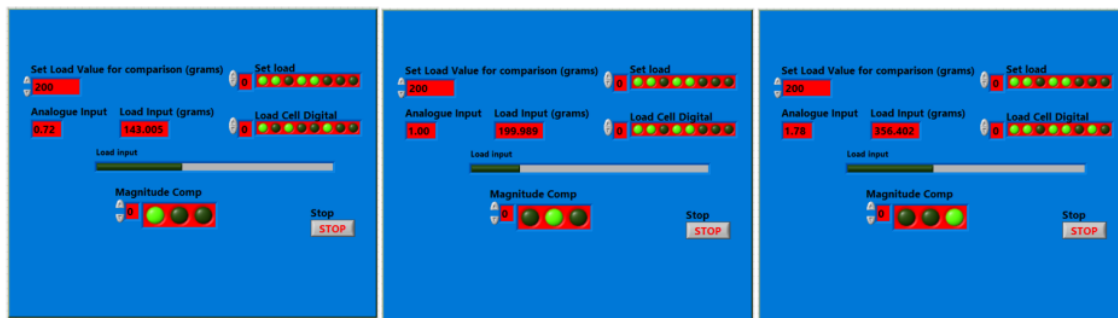


Figure 89 – Magnitude comparator results for 200g

*Good set of
test results
for 200g*

3.1.2 500g Test

The comparator set-point was then increased to 500g. Figure 90 displays the excel trendline graph. The configuration for this test is identical to the 200g test with the analogue input displayed by the orange trendline, the set point in grey, the “less than” trendline seen in yellow, “equal to” in blue and “greater than” in green.

As seen in figure 90, the test was started with the analogue input value being set at a lesser value than the 500g set point, this is confirmed as the yellow “Less Than” trendline is in its high state. The set point was then increased to 500g and the “Less Than” trendline entered its zero state as the blue “Equal To” trendline entered its state proving that the analogue input is equal to its set load value. Finally, the analogue input was increased to over 500g causing the green “Greater Than” trendline to enter its high state.

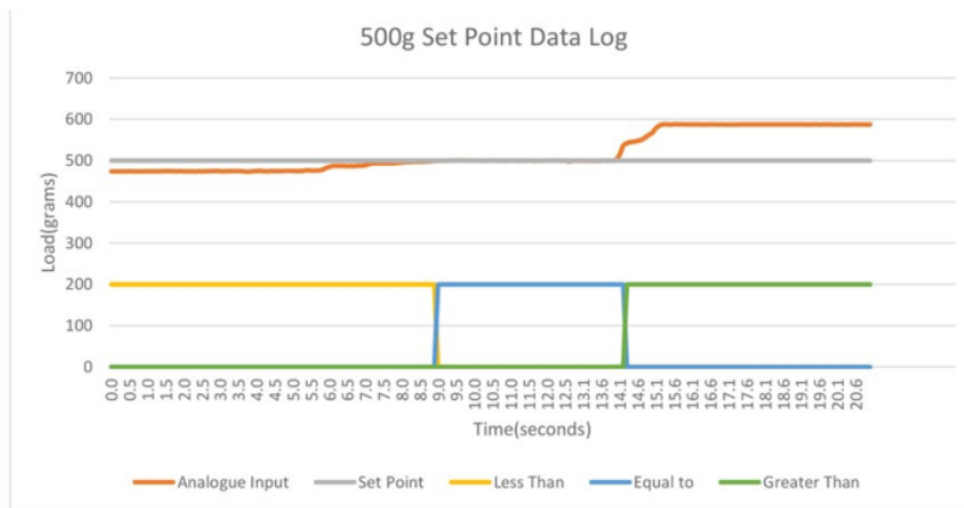


Figure 91 - Trendline graph for 500g

Figure 90 displays the LabVIEW front panel magnitude comparator results, these performed correctly just like the 200g test. It is seen that when the analogue input is lower than the set-load value the "Less Than" lamp illuminates. When the analogue input is equal to the load value the "Equal To" lamp illuminates and when the analogue input is set to a greater value than the set load the "Greater Than" lamp illuminates. All binary outputs are also seen to be correctly functioning.

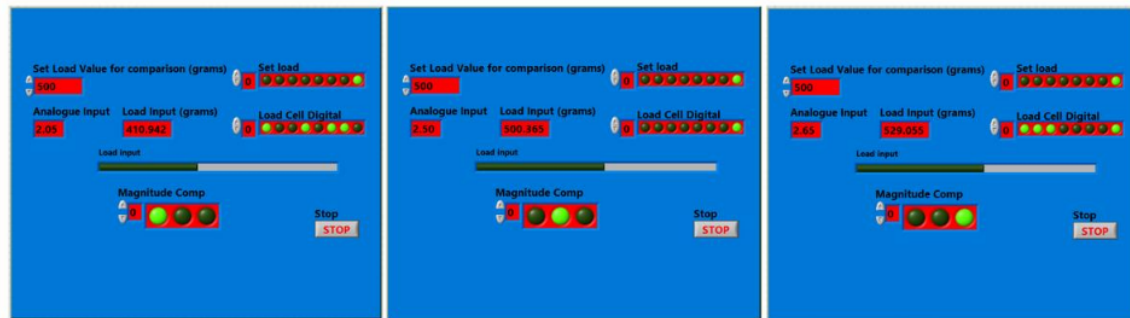


Figure 90 - Magnitude comparator results for 500g

Good set of
results for 500g

3.1.3 700g test

The comparator set-point was then increased to 700g. Figure 92 displays the excel trendline graph. The configuration for this test is identical to the 200g and 500g test with the analogue input displayed by the orange trendline, the set point in grey, the “less than” trendline seen in yellow, “equal to” in blue and “greater than” in green.

As seen in figure 92, the test was started with the analogue input value being set at a lesser value than the 700g set point, this is confirmed as the yellow “Less Than” trendline is in its high state. The set point was then increased to 700g and the “Less Than” trendline entered its zero state as the blue “Equal To” trendline entered its high state proving that the analogue input is equal to its set load value. Finally, the analogue input was increased to over 700g causing the green “Greater Than” trendline to enter its high state.

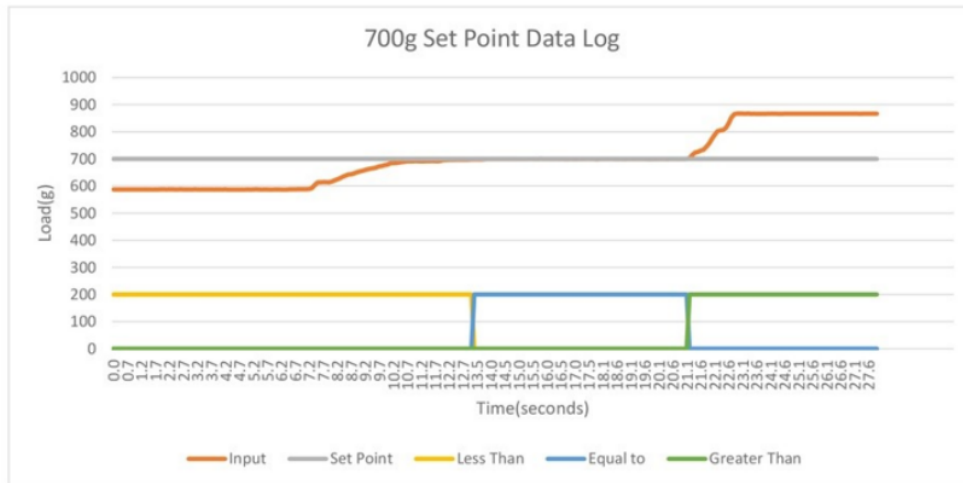


Figure 92 - Trendline graph for 700g

Figure 93 displays the LabVIEW front panel magnitude comparator results, these performed correctly just like the 200g and 500g test. It is seen that when the analogue input is lower than the set-load value the “Less Than” lamp illuminates. When the analogue input is equal to the load value the “Equal To” lamp illuminates and when the analogue input is set to a greater value than the set load the “Greater Than” lamp illuminates. All binary outputs are also seen to be correctly functioning.

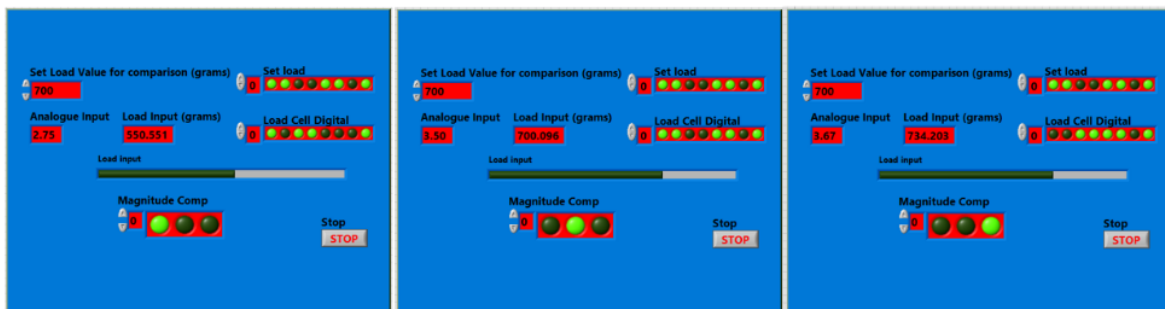


Figure 93 - Magnitude comparator results for 700g

Good
set of
results
for
700g

3.2 Element 3 Evaluation

In evaluation, element 3 has been hugely successful. The goal of designing a software-based system which has the functionality of comparing an input to a pre-determined set-point using digital design techniques was achieved, this was successfully developed with the functionality to export this data to an excel database to monitor and record the data in a graphical format, this was then functionally tested using the Nexys 4 DDR board. The reliability of the system was stress-tested by introducing 9 tests over three different set-points to prove the systems reliability. The use of LabVIEW allowed for a graphical language to be used to design a complex system with endless capability. The system could be improved as an overall by increasing the systems resolution to 12-bit, instead of 8-bit. As seen in element 2, when the ADC resolution was increased it produced far more accurate results, this is due to the quantization level amplitude being vastly reduced as the resolution increases, in turn reducing the error seen between the analogue input and the binary output, it would also eliminate the error seen whereby the "Equal To" magnitude comparator LED illuminates when the analogue input is not exactly equal to the set load. This modification would greatly increase the systems accuracy.

4. Overall Conclusions and Evaluation

In conclusion, the overall portfolio has been successful. The research section undertaken in element 1 provided the vital underpinning knowledge into key areas within the topic, key principles of the design of digital systems using VHDL, NI DAQmx, the design and operation of digital magnitude comparators, analogue to digital converters instrumentation and signal conditioning , and load cells, with each section providing the relevant knowledge needed for the further tasks.

Key design principles were understood and developed during the design of various logic circuits such as the 4-bit adder, half-adder and full-adders, further more complex circuits such as the 4-bit comparator were also designed. Digital design techniques using software packages was also learnt, this was implemented using Vivado and Multisim where these circuits, along with the 8-bit magnitude comparator were tested and simulated. The use of Xilinx's Nexys 4DDR boards the link between software and hardware were linked whereby the user's software was functionally tested.

Element 2 provided a greater depth of detail to the topics already researched in element 1, particularly focused on Load cells and signal conditioning, this element greatly strengthened areas within the design of electronic circuits, this is seen by the various circuits designed such as the Load Cell, Voltage Buffer and Unity Gain differential operational amplifier, these were then successfully implemented together as a functioning circuit. The system was then functionally tested and analysed, the analysis results proved that the calculations undertaken and design theory were correct. Element 2 provided knew knowledge of programming using the LabVIEW package for the design of systems using their graphical language, graphical user interfaces were also designed using LabVIEW's front panel section. Successful systems were designed within this package such as the 10-bit and 12-bit ADCs, these were tested to prove their functionality. Once proven functional, the 8-Bit ADC was designed. Element 2 concluded with the Multisim design of an 8-bit magnitude comparator which was implemented onto the Nexys 4DDR board in conjunction with the 8-bit ADC on the NI ELVIS II board using Digilent's Pmod Interface, the overall system was then successfully tested.

Element 3 was highly successful, throughout this element it was understood how data can be transferred from LabVIEW into excel in order for data to be stored, graphed and analysed. Modifications were made to the LabVIEW circuit from element 2. Analysis skills were strongly developed from analysing the trendline graphs for the system and in turn successfully proving that the system was correctly configured, as proved by the LabVIEW magnitude comparator. As a whole, the topics learnt within this portfolio have provided valuable skills in design, mathematical, analytical and software skills which are highly applicable to industry, especially with the emergence of industry 4.0.

Good evaluation on

Overall evaluation is reasonable

5. References

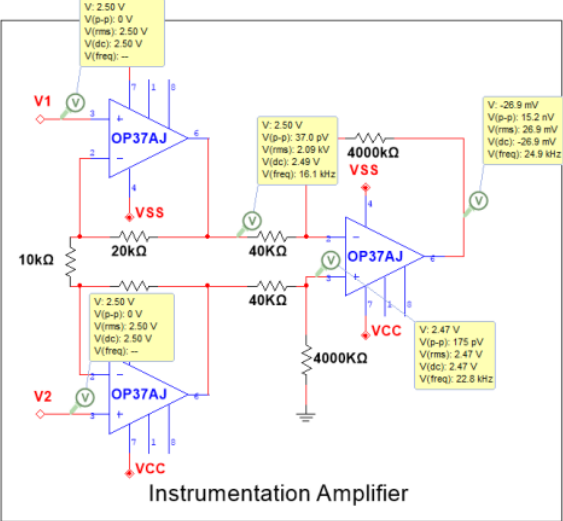
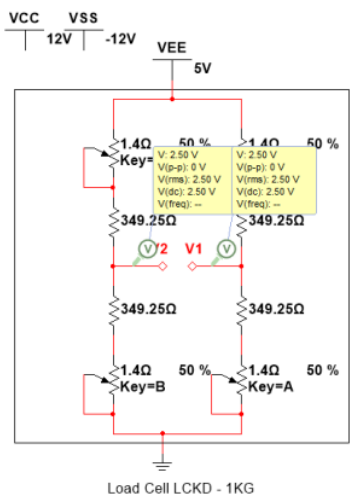
- byjus, 2021. *Wheatstone Bridge*. [Online]
Available at: <https://byjus.com/physics/wheatstone-bridge/#:~:text=The%20Wheatstone%20bridge%20works%20on,current%20flows%20through%20the%20galvanometer.>
[Accessed 03 05 2021].
- circuitstoday, 2017. *Half Adder*. [Online]
Available at: <https://www.circuitstoday.com/half-adder/#:~:text=Half%20adder%20is%20a%20combinational,AND%20of%20A%20and%20B.>
[Accessed 12 2020].
- Dekker, R., 2014. *What's the Difference Between VHDL, Verilog, and SystemVerilog?*. [Online]
[Accessed 2020].
- Dewesoft, 2020. *What Is ADC Converter (Analog-to-Digital Converter)?*. [Online]
Available at: <https://dewesoft.com/dag/what-is-adc-converter>
[Accessed 06 12 2020].
- geeksforgeeks, 2019. *Full Adder in Digital Logic*. [Online]
Available at: <https://www.geeksforgeeks.org/full-adder-in-digital-logic/#:~:text=Full%20Adder%20is%20the%20adder,as%20S%20which%20is%20SUM.>
[Accessed 12 2020].
- HBM, 2019. *Signal Conditioners: Helping to Provide Precise Measurements*. [Online]
Available at: <https://www.hbm.com/en/7339/what-is-a-signal-conditioner/>
[Accessed 12 2020].
- HBM, 2019. *The Working Principle of a Compression Load Cell*. [Online]
Available at: <https://www.hbm.com/en/7325/the-working-principle-of-a-compression-load-cell/>
[Accessed 12 2020].
- Instruments, N., 2019. *What is Signal Conditioning?*. [Online]
Available at: <https://www.ni.com/en-gb/innovations/white-papers/09/what-is-signal-conditioning-.html>
[Accessed 12 2020].
- National Instruments, 2019. *What is NI-DAQmx?*. [Online]
Available at: <https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z000000P8baSAC&l=en-GB>
[Accessed 2020 12 2020].
- Omega, 2018. *Subminiature Industrial Compression Load Cell*. [Online]
Available at: <http://www.farnell.com/datasheets/2340205.pdf>
[Accessed 12 2020].
- Omega, 2020. *Load Cells & Force Sensors*. [Online]
Available at: <https://www.omega.com/en-us/resources/load-cells>
[Accessed 12 2020].
- Pellerin, D., n.d. *An Introduction to VHDL*. [Online]
Available at: http://www.uco.es/~ff1mumui/h_intro.htm
[Accessed 2020].
- Tutorials, E., 2018. *Digital Comparator*. [Online]
Available at: https://www.electronics-tutorials.ws/combinational/comb_8.html
[Accessed 06 12 2020].

Xilinx, 2018. *Intellectual Property*. [Online]
Available at: <https://www.xilinx.com/products/intellectual-property.html>
[Accessed 12 2020].

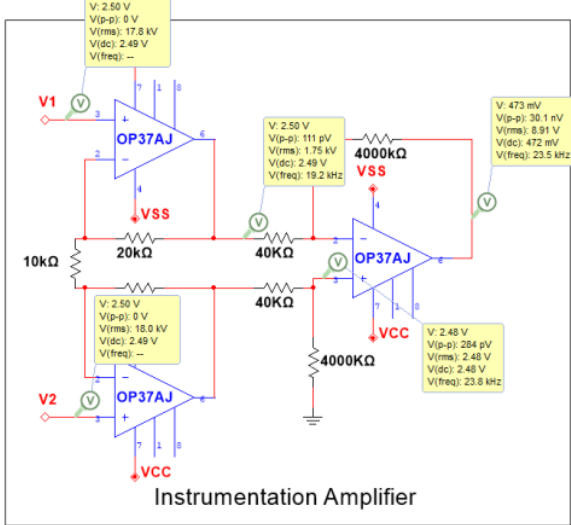
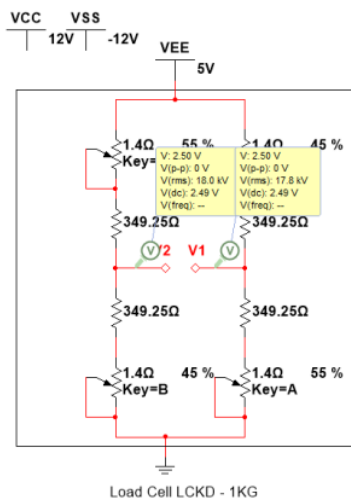
*Good number of
refs used*

6. Appendix
6.1 Element 2

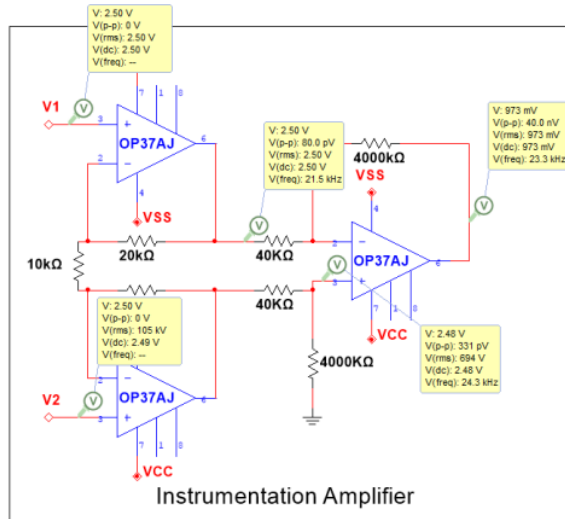
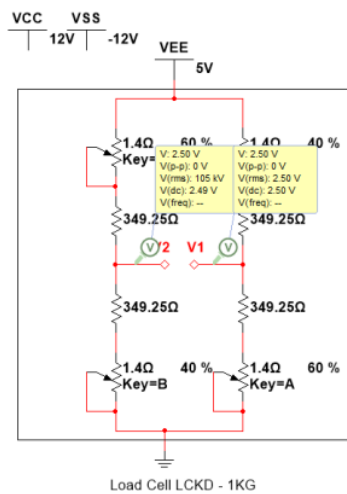
50Ω-50Ω – 0 Grams



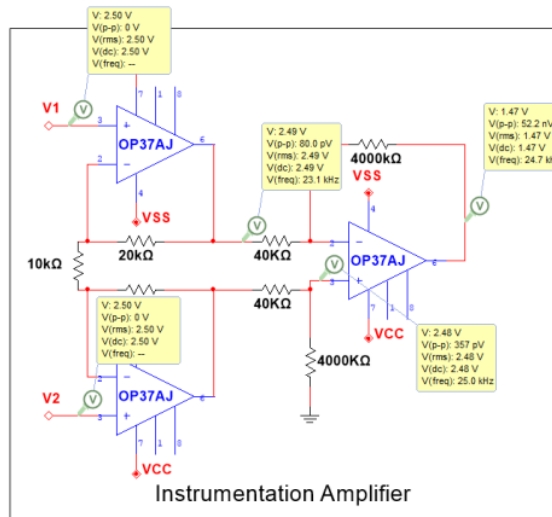
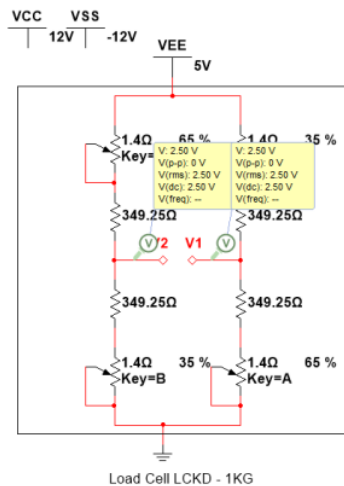
55Ω -45 Ω – 100 Grams



60-40



65-35



VCC VSS
12V -12V
VEE
5V

1.4kΩ 70 %
Key=

349.25Ω

V=2.50 V
V(p-p): 1.22 pV
V(rms): 29.9 kV
V(dc): 2.49 V
V(freq): 19.6 kHz

1.4kΩ 30 %
V=2.50 V
V(p-p): 0 V
V(rms): 27.8 kV
V(dc): 2.48 V
V(freq): --

349.25Ω

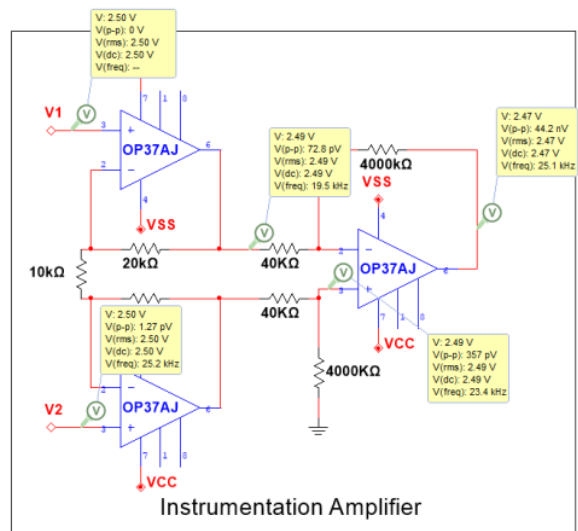
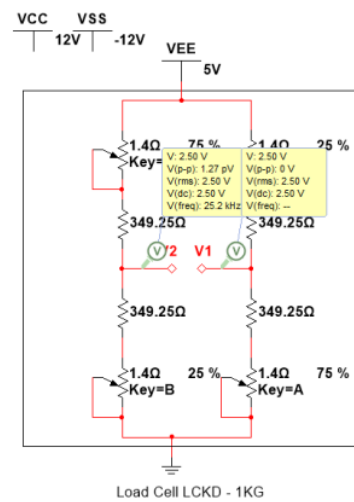
V1

349.25Ω

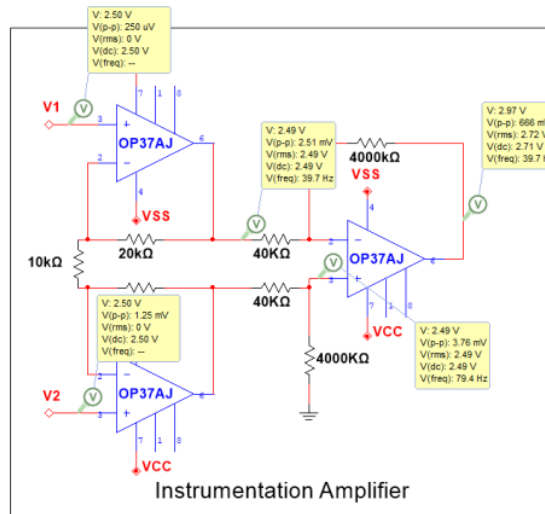
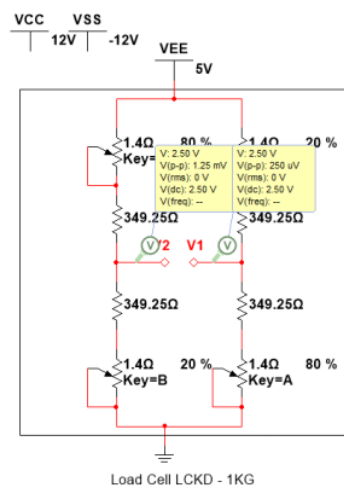
1.4kΩ 30 %
Key=B

1.4kΩ 70 %
Key=A

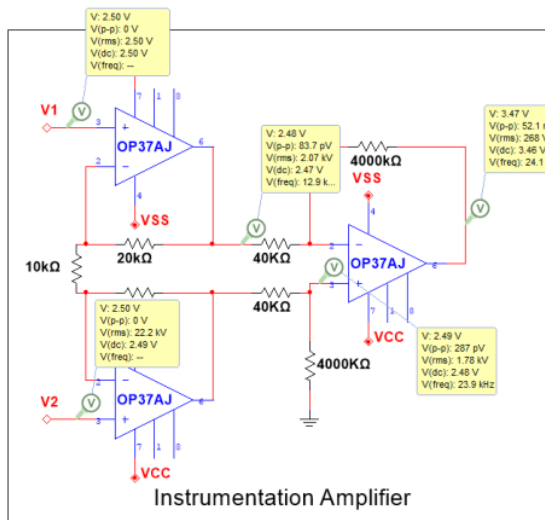
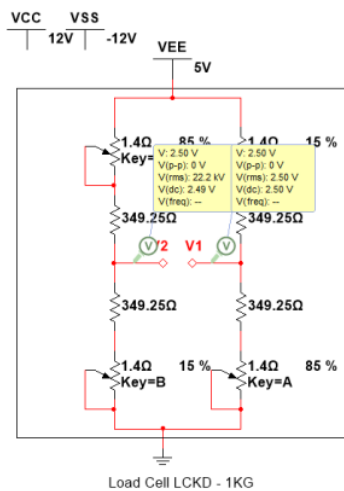
Load Cell LCKD - 1KG



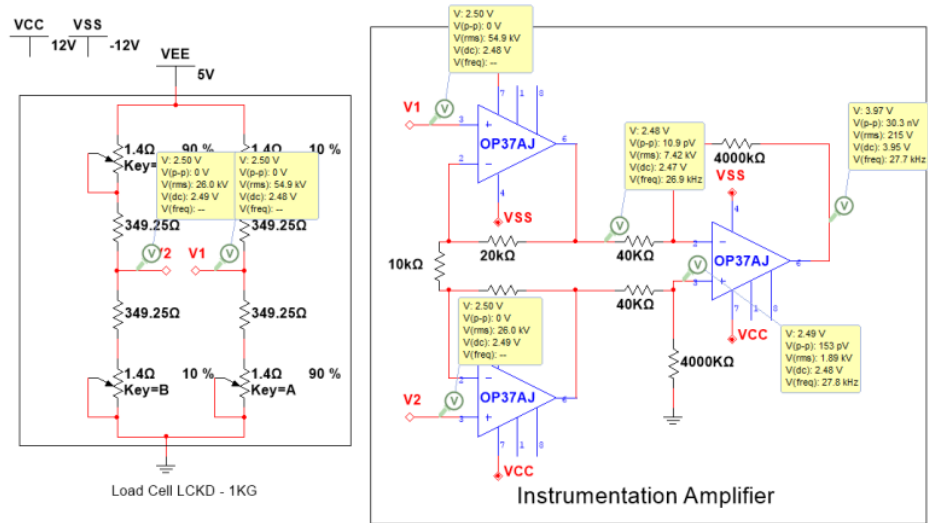
80-20



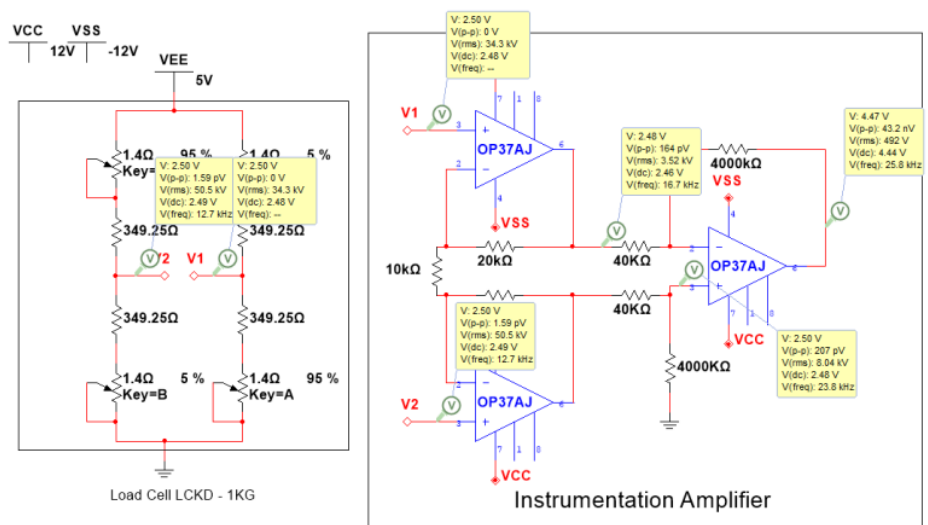
85-15

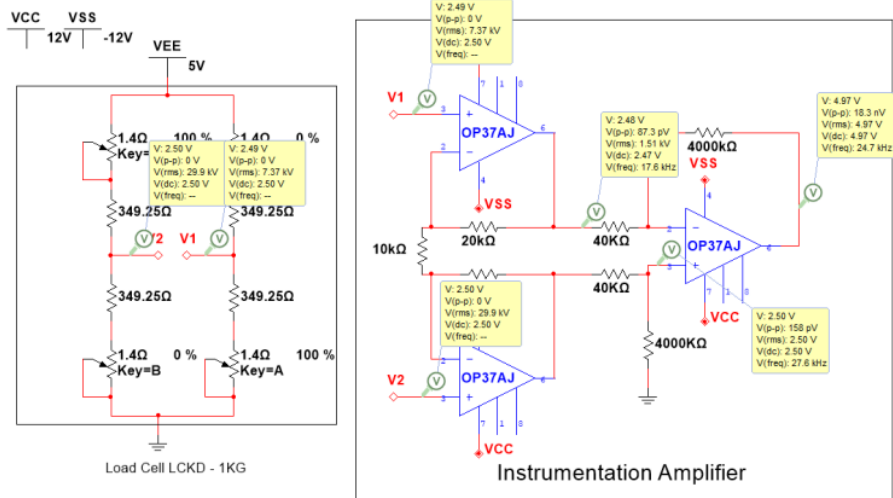


90-10



95-5





*Instrument
amp
conditions
are fully
covered here*

6.2 Element 3

Write to Measurement configuration File 2

Configure Write To Measurement File [Write To Measurement File2]

Filename
H:\LabVIEW Data\test2.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

Write to Measurement configuration File 3

Configure Write To Measurement File [Write To Measurement File3]

Filename
H:\LabVIEW Data\test3.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

Write to Measurement configuration File 4

The dialog box is titled "Configure Write To Measurement File [Write To Measurement File4]". It contains several sections:

- Filename:** A text box containing "H:\LabVIEW Data\test4.lvm" with a folder icon to its right.
- Action:** A group box containing:
 - ☒ 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)A "Settings..." button is located below the "Save to series of files" option.
- File Format:** A group box containing:
 - ☒ Text (LVM)
 - ☐ Binary (TDMS)
 - ☐ Binary with XML Header (TDM)
 - ☐ Microsoft Excel (.xlsx)
 - ☒ Lock file for faster access
- Segment Headers:** A group box containing:
 - ☐ One header per segment
 - ☒ One header only
 - ☐ No headers
- X Value (Time) Columns:** A group box containing:
 - ☐ One column per channel
 - ☐ One column only
 - ☒ Empty time column
- Delimiter:** A group box containing:
 - ☒ Tabulator
 - ☐ Comma
- File Description:** A text box with an "Advanced..." button to its right.

At the bottom are "OK", "Cancel", and "Help" buttons.

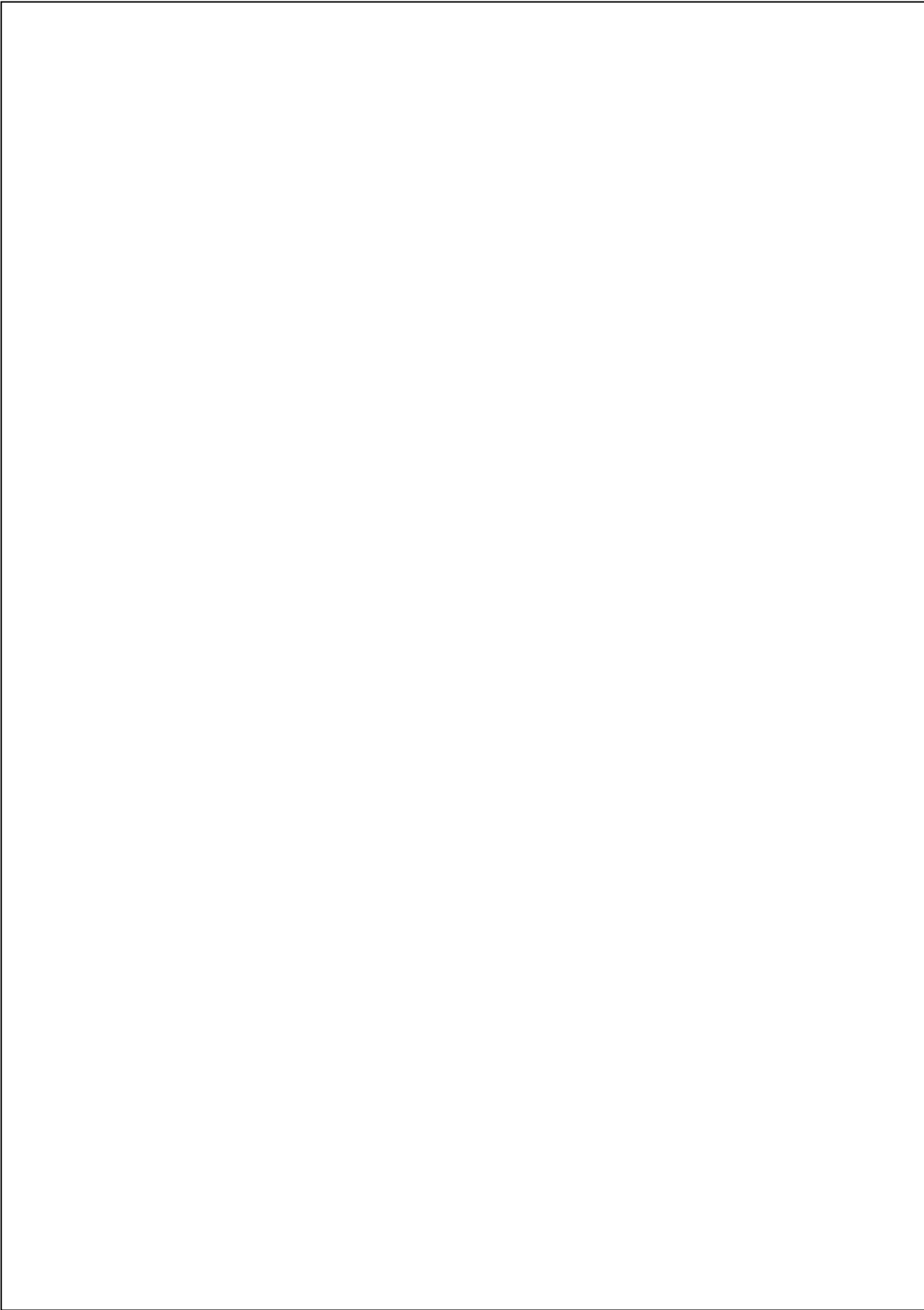
Write to Measurement configuration File 5

The dialog box is titled "Configure Write To Measurement File [Write To Measurement File5]". It contains several sections:

- Filename:** A text box containing "H:\LabVIEW Data\test5.lvm" with a folder icon to its right.
- Action:** A group box containing:
 - ☒ 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)A "Settings..." button is located below the "Save to series of files" option.
- File Format:** A group box containing:
 - ☒ Text (LVM)
 - ☐ Binary (TDMS)
 - ☐ Binary with XML Header (TDM)
 - ☐ Microsoft Excel (.xlsx)
 - ☒ Lock file for faster access
- Segment Headers:** A group box containing:
 - ☐ One header per segment
 - ☒ One header only
 - ☐ No headers
- X Value (Time) Columns:** A group box containing:
 - ☐ One column per channel
 - ☐ One column only
 - ☒ Empty time column
- Delimiter:** A group box containing:
 - ☒ Tabulator
 - ☐ Comma
- File Description:** A text box with an "Advanced..." button to its right.

At the bottom are "OK", "Cancel", and "Help" buttons.

*Full set of text
file
configurations
are present*



1809799

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

Instructor

71 /100

PAGE 1

PAGE 2

PAGE 3

Text Comment. Good table of contents but there are parts missing

PAGE 4

PAGE 5

Text Comment. Good list of figures for element 1. There are no figures listed for element 2 and 3

PAGE 6

Text Comment. Reasonable introduction

PAGE 7

PAGE 8

Text Comment. Excellent work here with good relevant sources to support your research

PAGE 9

PAGE 10

Text Comment. Good work with relevant sources referenced to support research

PAGE 11

PAGE 12

Text Comment. Very good effort here and demonstrates a clear understanding of the principle of operation

PAGE 13

Text Comment. Very good point to make about resolution which can affect the speed of the overall process.

PAGE 14

Text Comment. Good use of evaluation table

PAGE 15

Text Comment. Good work

PAGE 16

Text Comment. Research overall is very good and a good number of relevant sources used to support your findings.

PAGE 17

PAGE 18

Text Comment. Components for the 4 bit adder design have been clearly identified with good supporting images.

PAGE 19

PAGE 20

PAGE 21

Text Comment. Good work. It's always good practice to get all components working individually first, which you have clearly achieved here.

Text Comment. Hierarchical and inheritance has been identified in the design

PAGE 22

Text Comment. VHDL code is correct and neatly presented with the 4 full adder components showing in the image

PAGE 23

Text Comment. Test data is correct with a good range of values

PAGE 24

Text Comment. IP circuit is correct and is ready for implementation with FPGA. All IO are set.

PAGE 25

Text Comment. VHDL is correct

PAGE 26

Text Comment. Good design

Text Comment. Only the 4 bit comparator is created in Msim PLD

PAGE 27

Text Comment. 8 bit comparator explanation is good and clearly demonstrates the understanding.

PAGE 28

Text Comment. Good test result

PAGE 29

Text Comment. Good use of results table

PAGE 30

PAGE 31

PAGE 32

Text Comment. Very good set of results

PAGE 33

Text Comment. Good use of table of results

PAGE 34

PAGE 35

Text Comment. Excellent set of results

Text Comment. Good work and proves the 8 bit design before PLD design and implementation onto FPGA

Text Comment. PLD design is correct

Text Comment. Very good set of results and demonstrates a good working knowledge of the system and its capabilities

Text Comment. Error here. There is no 8 bit adder

Text Comment. Some minor errors in the evaluation but overall there are good points made

Text Comment. Unity gain is to test diff amp only. Amp needs a high gain for a small load to range from 0-5V.

Text Comment. Gain value is explained

Text Comment. Good table of values

Text Comment. Linear graph is correct

PAGE 49

PAGE 50

PAGE 51

Text Comment. Good test result for 10 bit ADC

PAGE 52

Text Comment. Test using front panel view shows the correct operation

PAGE 53

PAGE 54

Text Comment. Test is correct showing the scale at half way point

PAGE 55

Text Comment. Good test at full scale with limits set to prevent going over the max value

PAGE 56

PAGE 57

Text Comment. Good test to prove the 8 bit ADC using virtual P.S

PAGE 58

PAGE 59

Text Comment. A good range of tests including max value test

Text Comment. PLD design is correct and Pmod port on fpga shows a working knowledge.

PAGE 60

Text Comment. Good image of the setup

Text Comment. Good use of table for formula blocks

Text Comment. G-code is correct

Text Comment. Good design of front panel with arrays included

Text Comment. Explanation does not explain how the bits are shifted right and added for the fpga comparator

Text Comment. All other points are explained

Text Comment. Tests indicate correct operation of system

Text Comment. Minor error here. LabVIEW combines the 16 bit word for the fpga

Text Comment. Evaluation covers good points but there are some confusion issues with the role of Msim and LabVIEW.

Text Comment. Good explanation

Text Comment. Good explanation

Text Comment. G- code is correct

Text Comment. Good example of using the write text file configuration

Text Comment. Good test for 200g

Text Comment. Good set of test results for 200g

Text Comment. Good set of results for 500g

Text Comment. Good set of results for 700g

Text Comment. Good evaluation

Text Comment. Overall evaluation is reasonable

Text Comment. Good number of refs used

Text Comment. Instrument amp conditions are fully covered here

Text Comment. Full set of text file configurations are present

THOMAS GRENNAN

0

RESEARCH-INFORMED LITERATURE



Research section has covered all requirements in good detail. Good relevant sources used.

7/10

KNOWLEDGE AND UNDERSTANDING



E1-Design and use of magnitude comparator with digital inputs has been created with good explanations. There are some minor errors / confusions in the explanations . Test methods used are good and followed a logical sequence. 7/10 E2- The LabVIEW design for the analogue load cell input is explained and the design is a good effort. The 16 bit word operation for the fpga is not explained. The shift right and add parts are missing and this is the key to how the data bits are organised for the fpga. The design is operational but needs to be explained. 6.5/10 E3-Dat logging has ben fully explained and a good knowledge of this section is demonstrated. 7.5/10

21/30

ANALYSIS



E1-Test and results are very comprehensive . The evaluation has minor errors / confusions but does cover the main issues. 3.5/5 E2-The results from LabVIEW are comprehensive and detailed. There are some minor confused points made in the evaluation. 6.5/10 E3-Results from the data logging section are very detailed and a good evaluation for this section. 8/10

18/25

PRACTICAL APPLICATION AND DEP.



E1-Prototypes for the comparator have been designed using logical methods. PLD design methods used were successfully implemented using the fpga dev board. 3.5/5 E2-Design and build of LabVIEW interface for fpga followed a good development sequence. The dev sections were tested at designated stages to prove design. The data was shifted right and added successfully. 3.5/5 E3-Data logging using LabVIEW methods has ben designed and tested for operational requirements. 8/10

15/20

SKILLS FOR PROFESSIONAL PRAC.



Report content was good and had a good sequence. The list of figures only listed for E1 and the table of contents has parts missing. 9/15

10/15