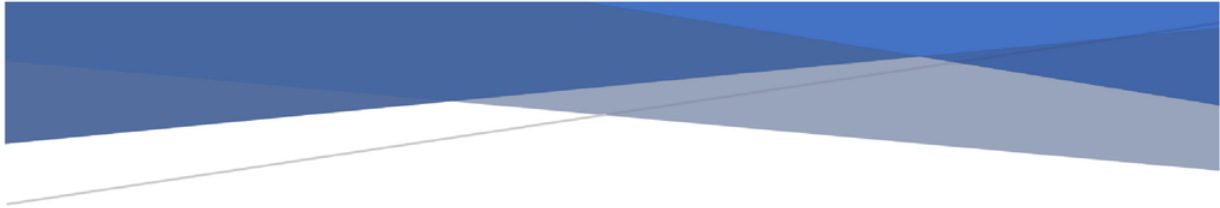


Ass 2 EPT final

by Michael Lewis

FILE	5293_MICHAEL_LEWIS_ASS_2_EPT_FINAL_158487_240109377.DOCX		
TIME SUBMITTED	03-MAY-2018 01:44PM (UTC+0100)	WORD COUNT	6663
SUBMISSION ID	88238598	CHARACTER COUNT	37786



ELECTRONIC PRODUCTION AND TEST

Assignment 2

Michael Lewis

Contents

Table Of figures.....	3
1. Introduction	5
2. Research.....	6
IC fabrication faults	6
Stuck at fault model	7
Stuck-at fault example	8
ESD dangers and prevention.....	9
ESD (Electro-Static-discharge) susceptibility	11
3. Design of assignment requirements	12
Derivation of testing truth table	12
Multisim testing	16
4. Analysis of multisim results	18
Fault testing on Multisim circuit	18
Test 1.....	19
Test 2.....	20
Test 3.....	21
5. ATE alternative using LFSR	22
A next state	23
B next state	23
C next state	23
D next state	24
E next state	24
F next state.....	24
Shift register pattern generator.....	26
6. Practical build and test on LabVIEW	28
LabView correct test set	31
Test Result.....	34

LabVIEW fault testing.....	34
Test 1 – Input A SA0	34
Test 2 – Input D SA0	35
Test 3 – Input F SA0.....	35
LabVIEW fault testing results	36
7. Conclusion.....	37
Recommendations	38
8. Appendices and references	39
Bibliography	39
Appendix	40
AND gate 2 input.....	40
NAND gate 3 input 4023	40
NOR gate 2 input.....	41
OR gate 2input	41
XOR gate 2 input	42

Table Of figures

Figure 1 - Fault model equivalent circuit	7
Figure 2 - Basic AND gate & truth table	8
Figure 3 - AND gate S@1 & truth table	8
Figure 4 - AND gate S@0 and truth table	8
Figure 5 - ESD component damage	9
Figure 6 - ESD sources	10
Figure 7- ESD protected workbench	10
Figure 8 - Earthing wrist strap/foot strap	11
Figure 9- Component ESD sensitivity	11
Figure 10- 3 input nand gate	12
Figure 11- 2 input and gate	12
Figure 12- 2 input nor gate	12
Figure 13 - 2 input or gate.....	12
Figure 14 - Derived test table.....	14
Figure 15 - Reduced test table	15
Figure 16 - Further reduced test table	15
Figure 17- Multisim testing no faults	16
Figure 18- Multisim no faults output waveform.....	16
Figure 19 - Logic Analyser output/ test table comparison.....	17
Figure 20- Fault testing tests 1, 2 & 3	18
Figure 21- Test 1 circuit.....	19
Figure 22 - Test 1 waveform	19
Figure 23 - Test 2 circuit.....	20
Figure 24 - Test 2 waveform	20
Figure 25- Test 3 waveform	21
Figure 26- LFSR present/next state table.....	22
Figure 27- ATE alternative summary of expressions.....	25
Figure 28 - Shift register Multisim circuit.....	26
Figure 29- Shift register waveform output	27
Figure 30 - HEX to DEC conversion	28
Figure 31 - LabVIEW front panel part 1	28
Figure 32- LabView front panel part 2	30
Figure 33- User interface screen.....	30

Figure 34 - Labview tests 0-2	31
Figure 35- LabVIEW tests 3 - 5	32
Figure 36- LabVIEW tests 6-7	33
Figure 37- LabVIEW fault testing, test 1	34
Figure 38- LabVIEW fault testing - test 2	35
Figure 39- LabVIEW fault testing - test 3	35

*Good list
of
contents*

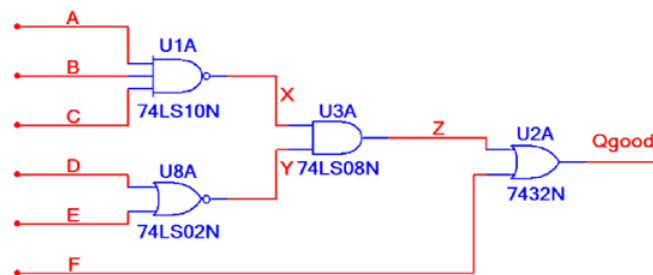
1. Introduction

The aims of this assignment are as follows:

- Investigate 'single stuck at' fault model and IC fabrication faults which will be used to test a PCB designed for a control process in industry
- Investigate the cause and dangers of electrostatic discharge (ESD) and methods used to prevent this in electronic design and manufacturing processes
- Implement the previously researched single stuck at fault modelling technique on a simple logic circuit and derive a minimal list of tests to carry out, to fully test the logic circuit

The objective that will be used to achieve the aims of this assignment are:

- Research into IC fabrication faults and the use of SSF model to ensure them minimum number of tests to fully test the design.
- Research ESD and prevention methods used within industry and electronic manufacturing processes
- Produce the following combinational logic circuit on Multisim:



- Use the researched SSF model to detect various faults within the circuit
- Derive a minimum number of test vectors to test all gates and possible faults
- Design a 6-bit pattern generator that tests the circuit using the minimum number of test vectors
- Obtain suitable results from simulation and critically analyse
- Built the circuit on FPGA prototype board and use 2 of 3 following methods to generate the minimum number of test vectors:
 - NI Multisim and NI FPGA board and
 - NI LabVIEW and the NI ELVIS prototyping board or
 - NI Multisim PLD mode with FPGA.

*Good
introduction
which shows
your
understanding
of the tasks.*

2. Research

IC fabrication faults

The increasing complexity of modern IC's and the high-precision processes used to manufacture them, allows various fabrication faults to occur when producing these devices. A quote from the IEEE journal explains it best: *"imperfections in the fabrication process result in yield-reducing manufacturing defects, whose severity grows proportionally with the size and density of the chip."* (ISRAEL KOREN, 1998, p. 1819)

Some of the main categories of IC fabrication faults are listed below:

- Gate-level stuck at fault model
 - This is a fault that occurs when a logic gate has an input stuck at 1 or 0, which effects the logical output of the device. This is sometimes referred to as the line-stuck-at-fault.
- Transistor level stuck at fault
 - This fault describes a transistor at a switch level, either being stuck-open or stuck-closed.
- Bridging fault model
 - This fault model describes various opens or shorts between components/connections within the circuit. Opens tend to behave like line-stuck at faults, but resistive opens effect the propagation delay of the logic path. A short between two wires is commonly referred to as a bridge fault. (Chang, 2008)
- Delay fault model
 - This is the fault model that describes the resistive opens/shorts, which may mean signal propagate so slow they fall short of their specified timing, therefore effecting any following logic.

(Charles F. Hawkins, 1994)

(Andrew Grochowski, 1997)

(Kabisatpathy. P, 2005)

Stuck at fault model

The stuck at fault model can be divided into smaller categories that explain different kinds of stuck at fault:

- SA1 (Stuck-At-1) fault – This fault occurs when a signal in a logic circuit is stuck at a logic 1
- SA0 (Stuck-At-0) fault – This fault occurs when a signal in a logic circuit is stuck at a logic 0
- Bridging fault – This fault occurs when two or more signals are shorted together, forming a bridge between the two separate signals
- Feedback bridge fault – Occurs when an input and output are shorted together

(O. A. Petlin, n.d.)

Figure 1 shows stuck open and stuck short fault on various basic components.

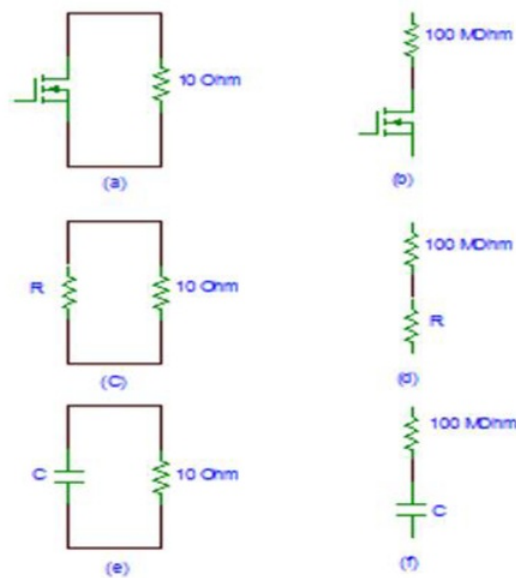


Figure 1 - Fault model equivalent circuit

(R. H. Khade, 2014)

- a) MOSFET stuck-short
- b) MOSFET stuck-open
- c) Resistor stuck short
- d) Resistor stuck open
- e) Capacitor stuck-short
- f) Capacitor stuck-open

Stuck-at fault example

A simple example will be shown on a 2 input and gate and the effects of a stuck at 1 and stuck at 0 fault. Figure 2 shows a standard 2 input AND gate with the resulting truth table.

Figure 3 shows what happens if the input node A is stuck at a logic 1. As shown by the truth table, having an input signal stuck at 1 causes the AND gate to output twice. The same outcome would be achieved if node B was stuck at 1. If both A and B were stuck at 1, the AND gate would always output a 1.

Figure 4 shows the AND gate having its A input node stuck at 0. The resulting truth table shows that this causes the AND gate to always output logic 0. The same would be true if both nodes A and B were stuck at 0.

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1



Figure 2 - Basic AND gate & truth table

A	B	Y
1	0	0
1	1	1
1	0	0
1	1	1



Figure 3 - AND gate S@1 & truth table

A	B	Y
0	0	0
0	1	0
0	0	0
0	1	0



Figure 4 - AND gate S@0 and truth table

ESD dangers and prevention

Electro-static discharge (ESD) danger is the risk of discharging built-up static electricity into electronic devices. Texas instruments explain the cause of this phenomenon: *“Typically, it is created by insulator surfaces rubbing together or pulling apart. One surface gains electrons, while the other surface loses electrons.”* (Cline, 2014).

The negative effects of ESD not only effect manufacturing throughput, they can also cause internal damage to components that can sometimes go undetected by factory test equipment. Some examples of this damage are shown in Figure 5. The white arrows and circles show the damage to the microscopic traces inside of IC’s. These are only a couple of examples, other damage that can be caused is; oxide punch through, junction burnout, and metallization burnout. (Cline, 2014)

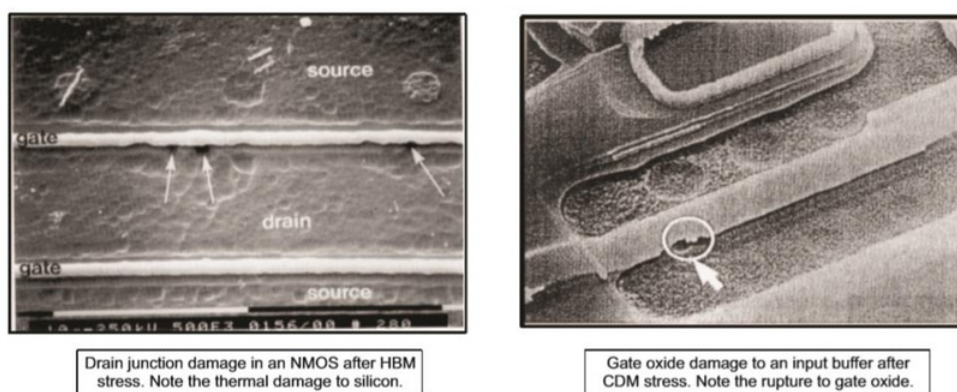


Figure 5 - ESD component damage

(Cline, 2014)

The relatively tiny damage that ESD can have a massive impact on the system that contains the damaged IC. Therefore, control of this phenomenon is essential where the risk of ESD damage is present. To be able to control the risk, the conditions that create ESD must be understood.

The table in Figure 6 shows some sources of ESD and the typical charge that can be achieved from a source. The table also shows how the level of relative humidity in an area can affect the amount of charge that can be built up. Although much less charge is built up when the relative humidity (RH) is high compared with a low RH, the voltage level can still be hazardous to components at 1500V, as described in the table.

Examples of Static Generation - Typical Voltage Levels		
Means of Generation	10-25% RH	65-90% RH
Walking Across Carpet	35,000V	1,500V
Walking Across Vinyl Tile	12,000V	250V
Worker at a Bench	6,000V	100V
Poly Bag Picked up from Bench	20,000V	1,200V
Chair with Urethane Foam	18,000V	1,500V

Figure 6 - ESD sources

(ESD Association, 2018)

One of the most common and cost-effective methods of controlling ESD is to have an effective earthing system for employees, equipment and workspaces. The theory behind this is the ESD is not allowed to build up on people/workspaces/equipment because the charge always has a low impedance path to earth.

Figure 7 shows a typical workbench setup to protect from ESD. Figure 8 shows the employees foot-strap and wrist-strap. The employee is earthed through the earth strap, the workbench is also earthed as well as having an earth mat for the employee to stand on. As most footwear does not create a low impedance path to earth, employees working in ESD sensitive areas wear foot-straps that ensure a path from the employee's body to the ESD protected foot mat in the figure below.

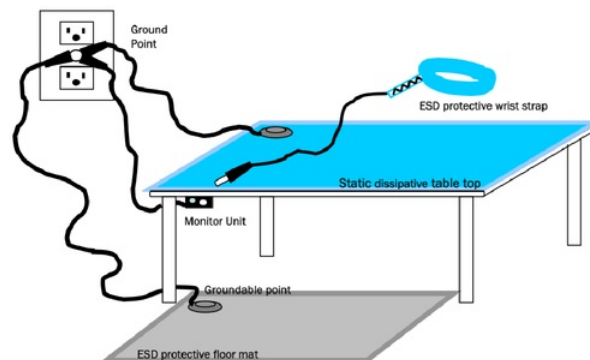


Figure 7- ESD protected workbench

(Mini-Circuits, 2015)



Figure 8 - Earthing wrist strap/foot strap

(ZiLOG Inc., 2001)

ESD (Electro-Static-discharge) susceptibility

Different electronic components can withstand more or less ESD than others. The table in Figure 9 shows some devices and their typical sensitivity to ESD. Knowing this helps find the correct solution to eliminate the risk of ESD depending on what components are being handled.

Device type	ESD withstand voltage sensitivity (V) HBM
MR heads, RF FETs	10 - 100
Power MOSFETs PIN diodes, laser diodes	100 - 300V
Pre - 1990 VLSI	400 - 1000V
Modern VSLI	1000 - 3000V
HCMOS	1500 - 3000V
CMOS B Series	2000 - 5000 V
Linear MOS	800 - 4000 V
Small geometry older bipolar	600 - 6000 V
Small geometry modern bipolar	2000 - 8000 V
Power bipolar	7000 - 25000 V
Film resistor	1000 - 5000 V

Figure 9- Component ESD sensitivity

(Electrostatic Solutions Ltd , 2012)

*Good
amount of
research
and relevant
sources.
Examples
are included
and
explanations
are logical.*

3. Design of assignment requirements

Derivation of testing truth table

The truth table for each gate in the combinational logic circuit must be sourced and used to create the full testing table. All information and images of the gates were taken from the manufacturer's datasheet. The datasheets used to get the truth tables use the terminology: H (Logic '1'), L (Logic '0') and X (Don't care).

Inputs			Output
A	B	C	Y
X	X	L	H
X	L	X	H
L	X	X	H
H	H	H	L

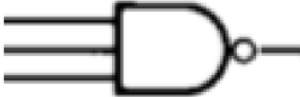


Figure 10- 3 input NAND gate

Inputs		Output
A	B	Y
L	L	L
L	H	L
H	L	L
H	H	H



Figure 11- 2 input and gate

Inputs		Output
A	B	Y
L	L	H
L	H	L
H	L	L
H	H	L



Figure 12- 2 input nor gate

Inputs		Output
A	B	Y
L	L	L
L	H	H
H	L	H
H	H	H



Figure 13 - 2 input or gate

*These are the fault models of the gates. How were these derived?
There is a formula for this.*

The test table derived from the logic gate truth tables is shown in Figure 14.

The numbers in red show where the truth table inputs for the logic gates have been extracted, the blue numbers show how the values for Q good are carried through the circuit. To be more specific, the following will detail which letters and test numbers correspond to which gate:

- 3 input and gate = A, B, C, Q good: Test numbers 1, 2, 3, 4
- 2 input nor gate = D, E, Q good: Test numbers: 5, 6, 7
- 2 input and gate = X, Y, Q good: Test numbers: 8, 9, 10
- 2 input or gate = F, Z, Q good: Test numbers: 11, 12, 13

As X, Y and Z are not inputs to the circuit, they are signals generated by the circuit and they are not included in the HEX values column. Instead only inputs A to F and Q good are used to generate the HEX value. The faults column shows what conditions for each test would cause the output to be incorrect.

If all the inputs for the gates used are added up, the total number is 16. However, not all the rows from the truth tables must be entered into the derived test table. For example; The two input AND gate have 4 states from the truth table, but only 3 states need to be put into the derived test table without repeating tests. The A = H, B = H, Y = L can be omitted from the derived test table because A and B have already been tested HIGH independently, and the output result will be the same. This is true for some of the other gates too. It is for this reason that the first derived test table has 13 tests instead of the expected 16 from the logic truth tables, see Figure 14.

Any tests that share the same hex value means that they are effectively the same test. For example, test 1 and 9 both have a hex value of 70, therefore one of them can be removed. 5, 8 and 13 also shares the same hex value and 2 of these can be removed. By doing this, the total number of tests has been reduced from 13 to 10, shown in the table in Figure 15.

The numbers in the table marked with a * means that all or at least one number must be a logic 0 to achieve a correct result. The numbers marked ** means that all or at least one must be a logic 1 for a correct result. As the numbers marked with * and ** can be modified if the rules mentioned are followed, the table can potentially be compressed by changing the marked numbers to hopefully match other tests, making that test redundant.

Following the above procedure, the total number of tests has been further reduced from 10 to 8, see Figure 16. The further reduced test table is shown in. The process of reducing the test table makes the procedure for testing of the circuit much less complicated, as well as reduces the time the testing will take.

Test No	A	B	C	D	E	F	Q good	HEX	X	Y	Z	Fault
1	1	1	1	0	0	0	0	70	0	1	0	A+B+C SA0 X SA1
2	0	1	1	0	0	0	1	31	1	1	1	A SA1
3	1	0	1	0	0	0	1	51	1	1	1	B SA1
4	1	1	0	0	0	0	1	61	1	1	1	C SA1
5	0* 1	0* 1	0*	0	0	0	1	01	1	1	1	D+E SA1 X+Y SA0, Z SA0
6	0*	0*	0*	1	0	0	0	08	1	0	0	D SA0
7	0*	0*	0*	0	1	0	0	04	1	0	0	E SA0
8	0*	0*	0*	0	0	0	1	01	1	1	1	X+Y SA0
9	1	1	1	0	0	0	0	70	0	1	0	X SA1
10	0*	0*	0*	1**	1** 0	0	0	0C	1	0	0	Y SA1
11	1	1	1	1**	1**	0	0	7C	0*	0*	0	F+Z SA1
12	1	1	1	1**	1**	1	1	7F	0*	0*	0	F SA0
13	0*	0*	0*	0	0	0	1	01	1	1	1	Z SA0

Figure 14 - Derived test table

Good table derived from the above results

Test No	A	B	C	D	E	F	Q good	HEX	X	Y	Z	Fault
1	1	1	1	0	0	0	0	70	0	1	0	A+B+C SA0 X SA1
2	0	1	1	0	0	0	1	31	1	1	1	A SA1
3	1	0	1	0	0	0	1	51	1	1	1	B SA1
4	1	1	0	0	0	0	1	61	1	1	1	C SA1
5	0* 1	0* 1	0*	0	0	0	1	01	1	1	1	D+E SA1 X+Y SA0, Z SA0
6	0*	0*	0*	1	0	0	0	08	1	0	0	D SA0
7	0*	0*	0*	0	1	0	0	04	1	0	0	E SA0
8	0*	0*	0*	1**	1** 0	0	0	0C	1	0	0	Y SA1
9	1	1	1	1**	1**	0	0	7C	0*	0*	0	F+Z SA1
10	1	1	1	1**	1**	1	1	7F	0*	0*	0	F SA0

Figure 15 - Reduced test table

Test No	A	B	C	D	E	F	Q good	HEX	X	Y	Z	Fault
1	1	1	1	0	0	0	0	70	0	1	0	A+B+C SA0 X SA1
2	0	1	1	0	0	0	1	31	1	1	1	A SA1
3	1	0	1	0	0	0	1	51	1	1	1	B SA1
4	1	1	0	0	0	0	1	61	1	1	1	C SA1
5	0*	0*	0*	1	0	0	0	08	1	0	0	D SA0
6	0*	0*	0*	0	1	0	0	04	1	0	0	E SA0
7	1	1	1	1**	1**	0	0	7C	0*	0*	0	F+Z SA1
8	1	1	1	1**	1**	1	1	7F	0*	0*	0	F SA0

Figure 16 - Further reduced test table

Good
explanat
ion and
table
reductio
n
method
s. The

Multisim testing

The logic circuit was built on Multisim to allow the system to be tested with a logic analyser. To fully test the circuit, the logic values shown in the further reduced table have to be fed into the logic circuit. The circuit will then generate an output 'q', this is compared with a generated value of 'Qgood' which is the known good output. Any discrepancies between the logic circuit output 'q' and the generated 'Qgood' will show as a positive output from the XOR gate that 'q' and 'Qgood' feed in to. If the XOR gate outputs a 1, the signals do not match and there is a fault in the circuit.

There are many ways to feed the values from the reduced table into the logic circuit, but in this instance the HEX values from the reduced table have been entered into a 32-bit word generator. See Figure 17 for the Multisim circuit.

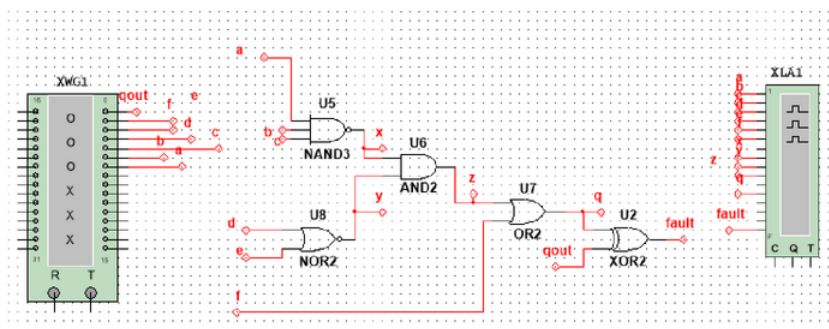


Figure 17- Multisim testing no faults

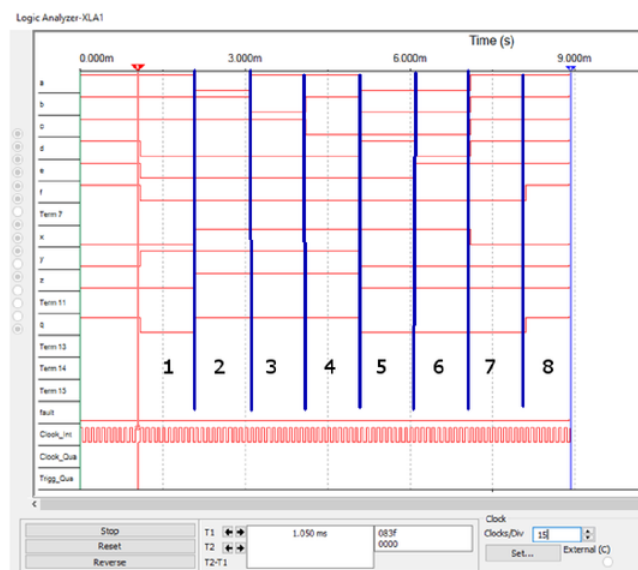


Figure 18- Multisim no faults output waveform

Some
ingenuity
shown her
with the use of
the word

Figure 18 shows the logic analyser output for the logic circuit. The screenshot has been annotated to show which columns correspond to the test numbers on the further reduced test table. As there has been no faults detected, the logic analyser output should match exactly.

Studying the output waveform gives the following results:

Test	A	B	C	D	E	F	X	Y	Z	Qgood
1	1	1	1	0	0	0	0	1	0	0
2	0	1	1	0	0	0	1	1	1	1
3	1	0	1	0	0	0	1	1	1	1
4	1	1	0	0	0	0	1	1	1	1
5	0	0	0	1	0	0	1	0	0	0
6	0	0	0	0	1	0	1	0	0	0
7	1	1	1	1	1	0	0	0	0	0
8	1	1	1	1	1	1	0	0	0	1

Figure 19 - Logic Analyser output/ test table comparison

The table derived from the logic analyser output matches the derived test table exactly and proves the Multisim circuit functions as specified.

*A very good
analysis here and
some ingenuity
demonstrated.*

4. Analysis of Multisim results

Fault testing on Multisim circuit

To confidently confirm that the Multisim functions correctly under all circumstances, the circuit must be tested under fault conditions. The faults that will cause the logic circuit to output incorrectly are shown in all derived test tables, see Figure 16. Only the tests from the most reduced test table are going to be performed as the other derived tables contain redundant tests.

Three tests from the further reduced table in Figure 16 are going to be performed. Figure 20 shows the three test that have been taken from the further reduced test table.

From the table in Figure 20, the tests to be performed will be:

1. X stuck-at 1
2. A stuck at 1
3. B stuck at 1

For the first test, X stuck at 1 was chosen arbitrarily; A or B or C stuck at 0 would have also caused the system to output incorrectly. But for this example, testing one condition for each will be sufficient.

Test No	A	B	C	D	E	F	Q good	HEX	X	Y	Z	Fault
1	1	1	1	0	0	0	0	70	0	1	0	A+B+C SA0 X SA1
2	0	1	1	0	0	0	1	31	1	1	1	A SA1
3	1	0	1	0	0	0	1	51	1	1	1	B SA1

Figure 20- Fault testing tests 1, 2 & 3

Test 1, 2 and 3 waveform outputs are shown in Figure 22, Figure 24 and Figure 25 respectively. In the first two test the Multisim circuit is provided to show how the fault have been simulated. All three waveforms have been annotated to show the signal that has been 'stuck-at' a particular logic level. The detected fault has also been annotated to show when it occurs. All the faults that were put onto the circuit were correctly detected as expected.

As the further reduced testing table only contains 8 entries, only 5 more tests would have the circuit fully tested (except test 1 requires further testing). This is not in the assignment as only 3 tests have been asked for.

The circuit is functioning exactly as detailed in the initial specification so far.

Test 1

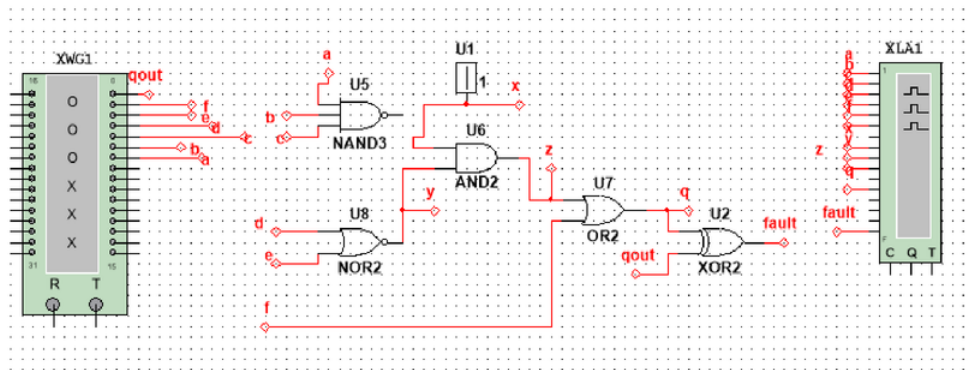


Figure 21- Test 1 circuit

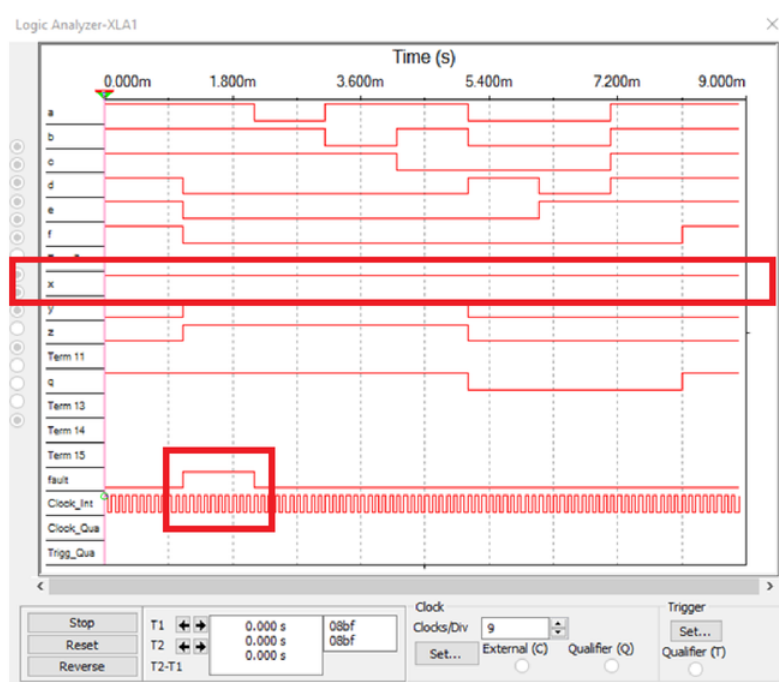


Figure 22 - Test 1 waveform

Test 2

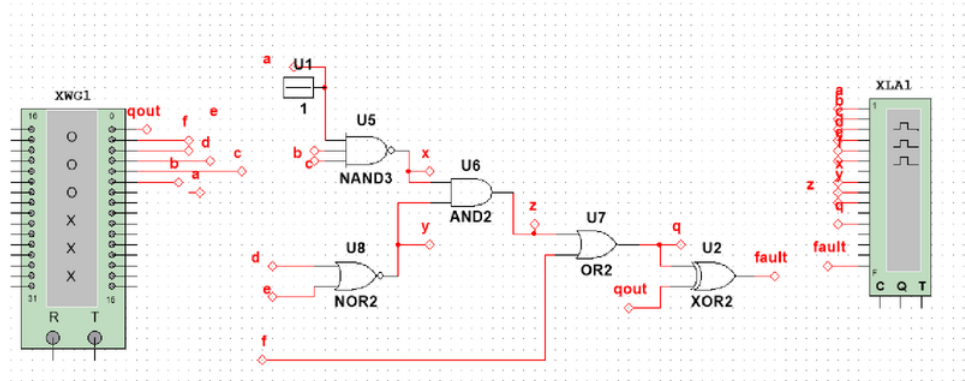


Figure 23 - Test 2 circuit

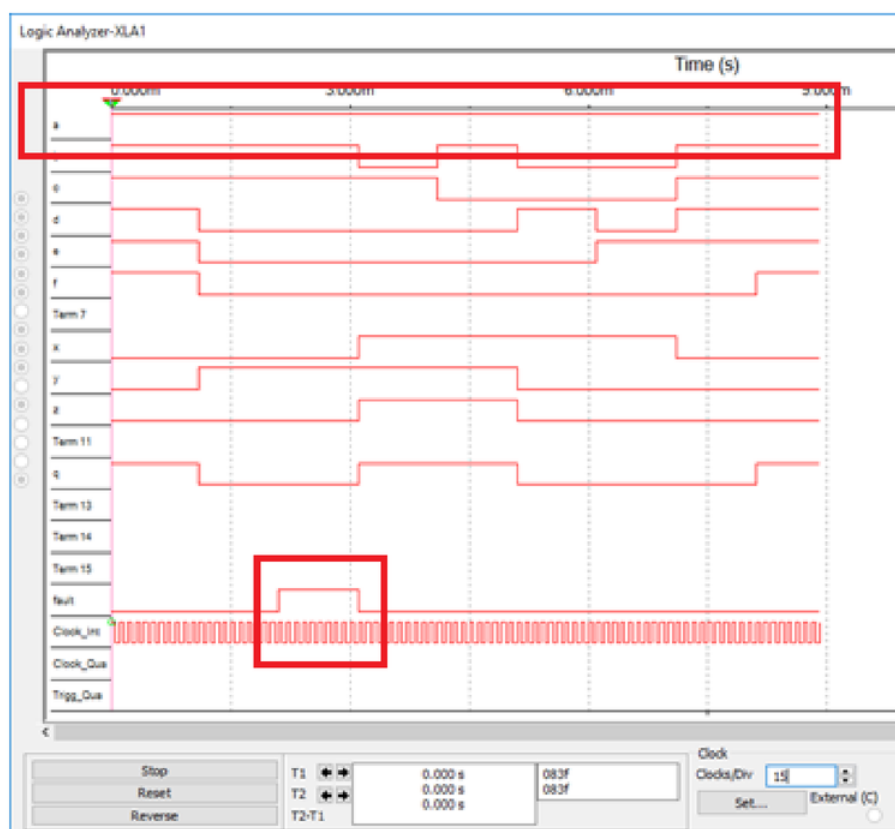


Figure 24 - Test 2 waveform

Test 3

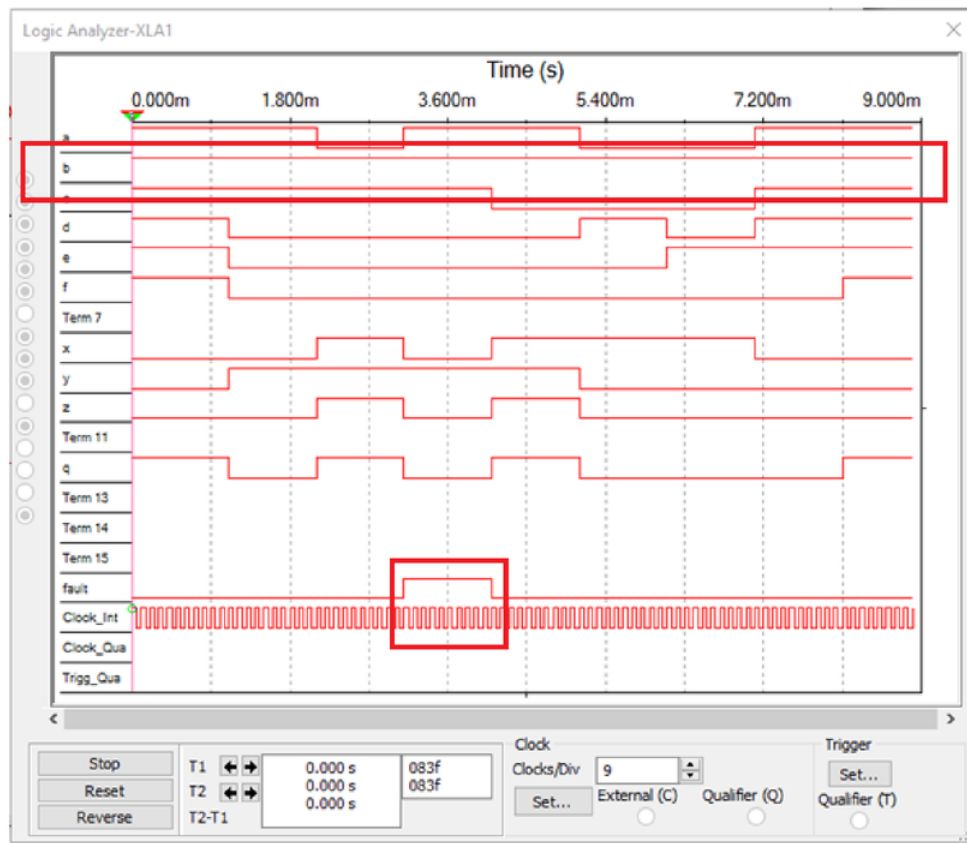


Figure 25- Test 3 waveform

*Excellent
analysis
demonstrated
here with
reference to the
table.*

5. ATE alternative using LFSR

Shift register can be used to test the circuitry as an alternative to the methods used so far. Karnaugh maps were developed from the reduced truth table which is a different method of testing. The LFSR will not be implemented physically or in simulation, instead the LFSR will only be designed to show that alternative methods of testing are available.

From the further reduced test table, a present/next state table was created. Only A to F signals are going to be generated, Qgood will not be included. See Figure 26 for the present/next state table.

The table has been annotated to show how the present states have been converted into the next state. The state directly underneath the present state is the next state.

PRESENT STATE						NEXT STATE					
A	B	C	D	E	F	A+	B+	C+	D+	E+	F+
1	1	1	0	0	0	0	1	1	0	0	0
0	1	1	0	0	0	1	0	1	0	0	0
1	0	1	0	0	0	1	1	0	0	0	0
1	1	0	0	0	0	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	1	0
0	0	0	0	1	0	1	1	1	1	1	0
1	1	1	1	1	0	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	0	0	0

Figure 26- LFSR present/next state table

From the present/next state diagram a Karnaugh map for each signal (A to F) will have to be created, totalling 6 6-input Karnaugh maps. The groupings on the Karnaugh map provides a compressed Boolean expression for the signals next state.

The following pages show each Karnaugh map and the resulting Boolean expression derived from them. Only the logic 1s have been entered onto the maps to make it easier to read.

All the next state expressions derived have been entered into a summary table, shown in Figure 27.

A next state

	ABC							
DEF	000	001	011	010	100	101	111	110
000			1			1		
001								
011								
010	1							
100								
101								
111							1	
110							1	

$A.B.C.D.E + A.NB.C.ND.NE.NF + NA.B.C.ND.NE.NF + NA.NB.NC.ND.E.NF$

B next state

	ABC							
DEF	000	001	011	010	100	101	111	110
000						1	1	
001								
011								
010	1							
100								
101								
111							1	
110							1	

$A.C.ND.NE.NF + A.B.C.D.E + NA.NB.NC.ND.E.NF$

C next state

	ABC							
DEF	000	001	011	010	100	101	111	110
000			1				1	
001								
011								
010	1							
100								
101								
111							1	
110							1	

$B.C.ND.NE.NF + A.B.C.D.E + NA.NB.NC.ND.E.NF$

D next state

	ABC							
DEF	000	001	011	010	100	101	111	110
000								1
001								
011								
010	1							
100								
101								
111								
110							1	

$NA.NB.NC.ND.E.F + A.B.NC.ND.NE.NF + A.B.C.D.E.NF$

E next state

	ABC							
DEF	000	001	011	010	100	101	111	110
000								
001								
011								
010	1							
100	1							
101								
111								
110							1	

$NA.NB.NC.ND.E.NF + NA.NB.NC.D.NE.NF + A.B.C.D.E.NF$

F next state

	ABC							
DEF	000	001	011	010	100	101	111	110
000								
001								
011								
010								
100								
101								
111								
110							1	

$A.B.C.D.E.NF$

State	Next state expression
A	A.B.C.D.E + A.NB.C.ND.NE.NF + NA.B.C.ND.NE.NF + NA.NB.NC.ND.E.NF
B	A.C.ND.NE.NF + A.B.C.D.E + NA.NB.NC.ND.E.NF
C	B.C.ND.NE.NF + A.B.C.D.E + NA.NB.NC.ND.E.NF
D	NA.NB.NC.ND.E.F + A.B.NC.ND.NE.NF + A.B.C.D.E.NF
E	NA.NB.NC.ND.E.NF + NA.NB.NC.D.NE.NF + A.B.C.D.E.NF
F	A.B.C.D.E.NF

Figure 27- ATE alternative summary of expressions

*Good
work and
good
attention
to detail.*

Shift register pattern generator

The Multisim circuit in Figure 28 shows the pattern generator built in multisim using shift register method. The circuit successfully generates the same test vector values as the previous pattern generator circuit, as shown in the logic analyser output in Figure 29. As intensive circuit testing or physical construction was not asked for, the running of the circuit and logic analyser output is sufficient proof that this method is equally effective.

The Z1, Z2 and Z3 outputs have not been derived from a karnaugh map in the same way the other output signals were derived. Instead these represent common expressions throughout the circuit, and save time by labelling the common expression for use throughout the logic circuit. This is why some of the A-F contain the inputs Z1, Z2 or Z3.

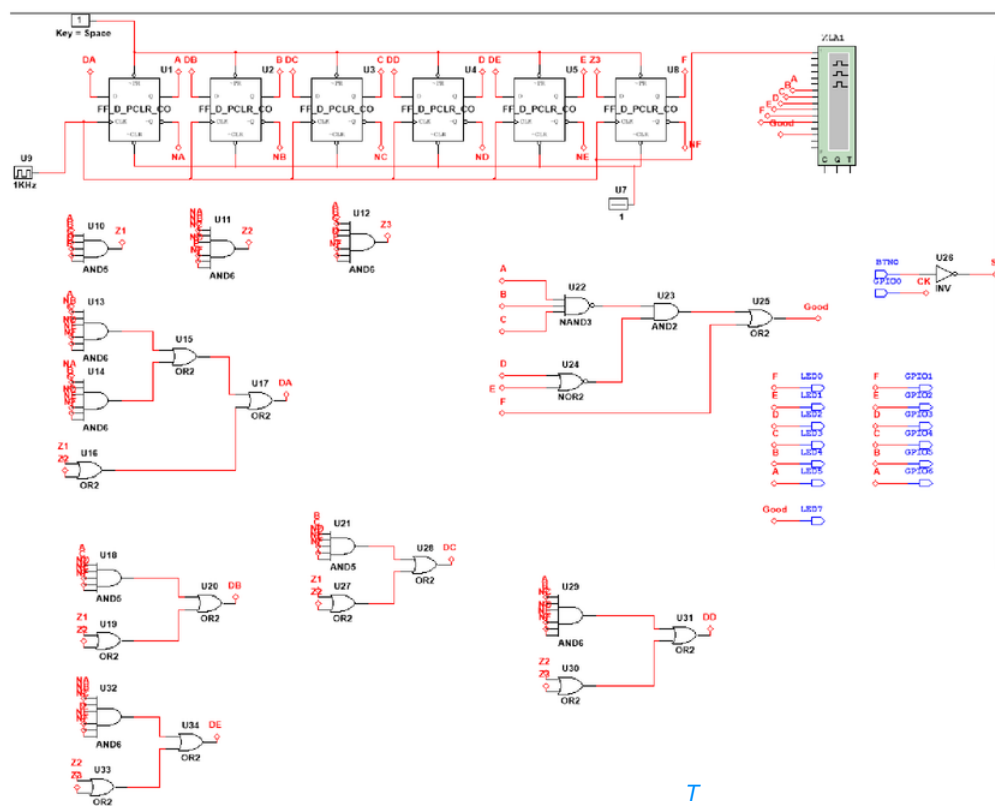


Figure 28 - Shift register Multisim circuit

T
h
e
r
e
s
u
l
t
a
n
t
c
o
n

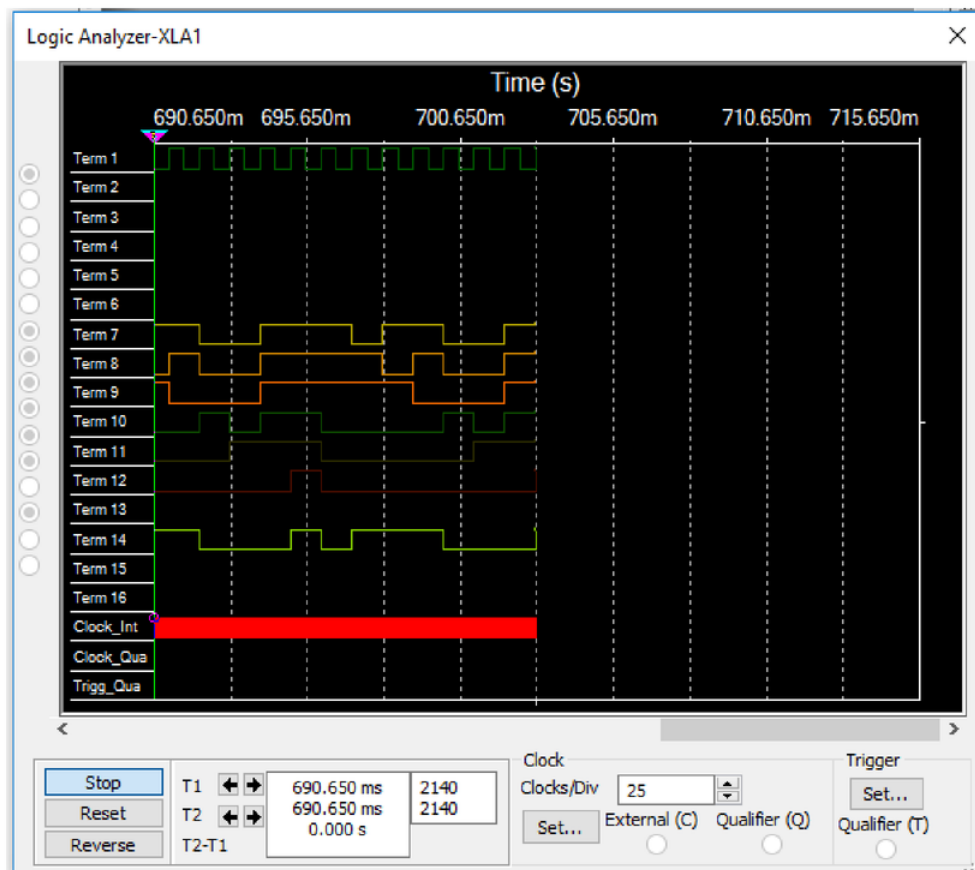


Figure 29- Shift register waveform output

Good analysis. The EX-OR should be included here and its function explained.

6. Practical build and test on LabVIEW

The LabVIEW circuit will output 7 bits to the logic circuit, the 7 bits correspond to the HEX values used in the further reduced test table. The bits being generated are: A to F, and Qgood. The A to F signals feed into the physical logic circuit, the Qgood signal is the correct sequence and is used to compare the output of the logic circuit to the good sequence, and output if a fault is detected. The fault can be detected because the Logic circuit Q output and generated Qgood are fed into an XOR gate; if the 2 values do not match, the XOR gate will output HIGH.

The HEX values must be converted to decimal as LabVIEW will convert the decimal into binary to be sent along the 7 output lines. See Figure 30 for conversion table.

HEX	70	31	51	61	08	04	7C	7F
DEC	112	49	81	97	8	4	124	127

Figure 30 - HEX to DEC conversion

The first part of the LabVIEW front panel is shown in Figure 31. The main components that make up the LabVIEW program have been highlighted with red boxes and number, to ease the explanation of how the program operates.

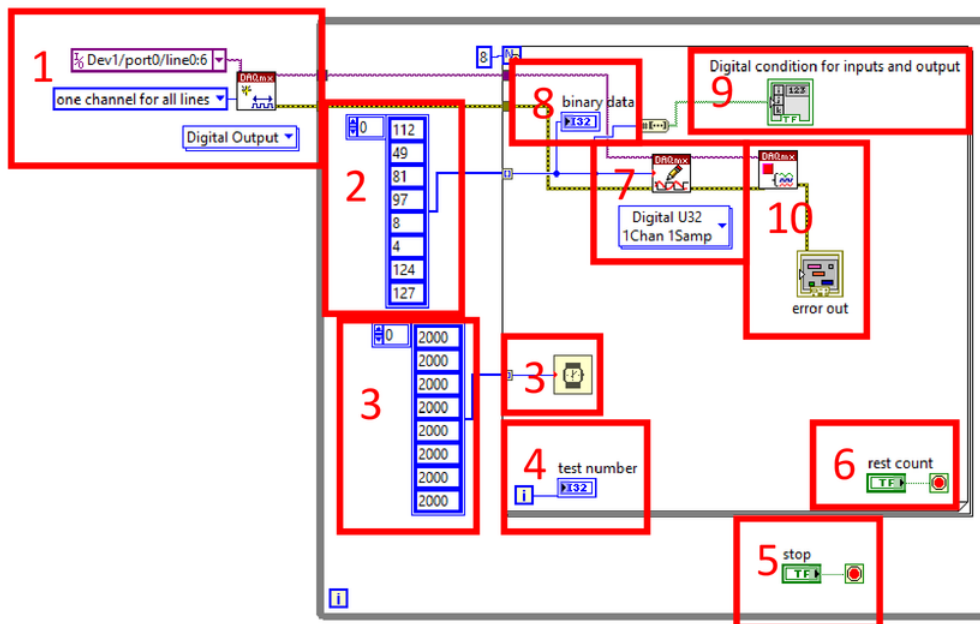


Figure 31 - LabVIEW front panel part 1

Below is a numbered list which explained each circled component in Figure 31.

1 – This component sets up the output lines from the LabVIEW board. This is part of the DAQMX design tools which allow data to be read and written from the prototyping board. Lines 0 to 6 are being used on port 0. Dev1 refers to the device number, as we only have 1 board connected its device 1.

2 – This is the array of decimal values that were converted from HEX. These are the values which will be output down the 7 output lines using components that will be explained shortly.

3 – An array of values used for timing. The wrist watch icon represents a delay, the numbers being fed into this component are a decimal value representing milliseconds. As all values are 2000, each stage of the program will delay for 2000ms.

4 – This is used to display the test number on the screen that the user will see (user interface) when the program is running (see Figure 33)

5 – This allows the program to be stopped by pressing a button on the user interface. (See Figure 33)

6 – This is used to reset the 'test number' count by pressing a button on the user interface. (See Figure 33)

7 – This component is part of the DAQMX design tools. Its function is to write the decimal values in box 2 to the output lines in **binary**.

8 – Shows the value of the 'binary data' on the user interface screen, displayed on screen as a **decimal** value. (See Figure 33)

9 – This component is used to display each binary bit (A to F and Q) in the form of LEDs on the user interface screen. (See Figure 33)

10 – These components are necessary for the LabVIEW Program to function correctly. They are a 'stop' component from the DAQMX tools and an error output component.

What about the function of the FOR loop?

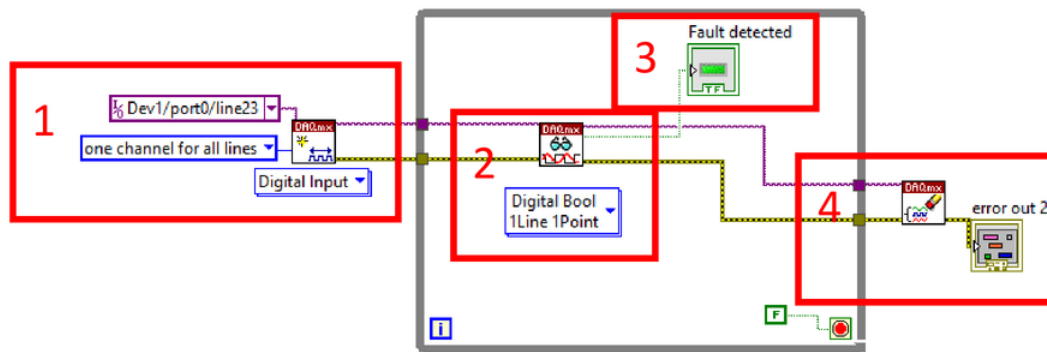


Figure 32- LabVIEW front panel part 2

Figure 32 shows the second part of the LabVIEW program. The program will be explained in the same way as part 1, using a numbered list referring to the figures annotations:

1 – Component from DAQMX design tools, allows an input to the development board. Only one input is being setup in this case to allow the XOR gate to feed in and display a fault on screen.

2 – Another component from the DAQMX design tools, takes the input from the previous component and controls a separate 'fault light' component that is displayed on the user interface screen.

3 – This is the light component that is placed on the user interface screen. Controlled through the previous 2 DAQMX components, fed from the XOR gate. (See Figure 33)

4 - These components are necessary for the LabVIEW Program to function correctly, as explained previously.

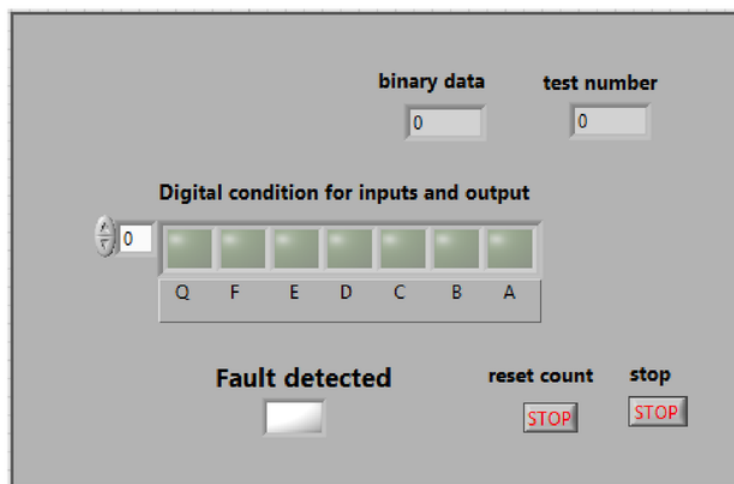


Figure 33- User interface screen

LabVIEW correct test set

The LabVIEW design will be run in conjunction with the physical logic circuit. If all has been designed correctly, the program should run through all 8 tests, displaying the correct corresponding binary number, without the fault light coming on.

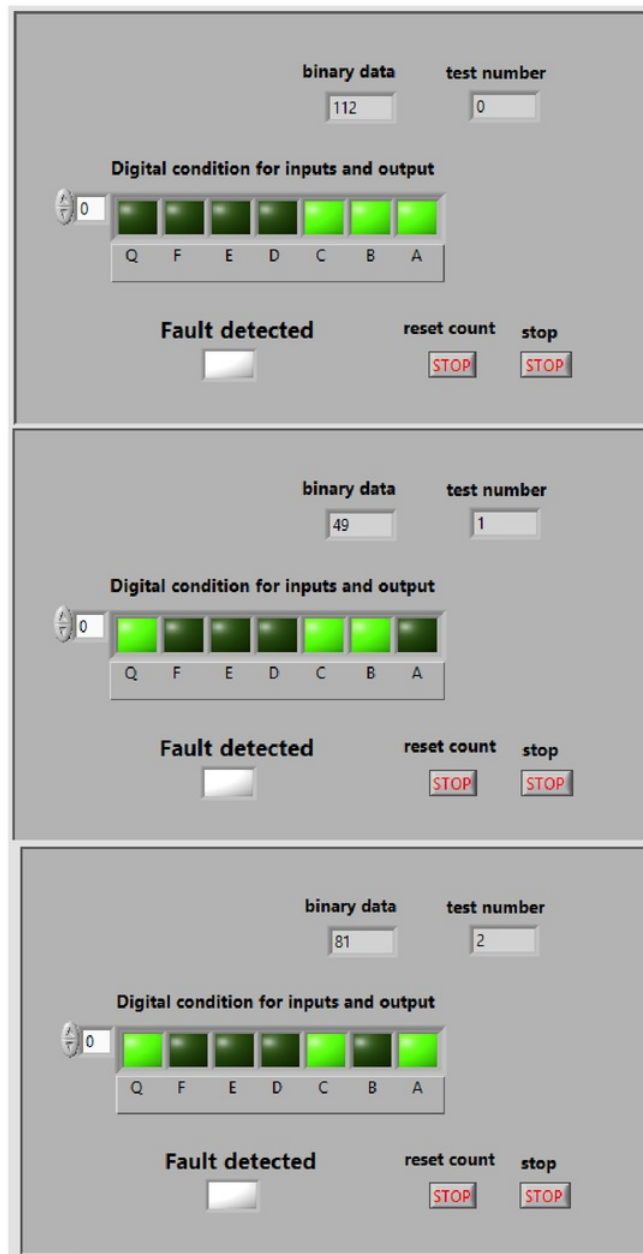


Figure 34 - LabVIEW tests 0-2

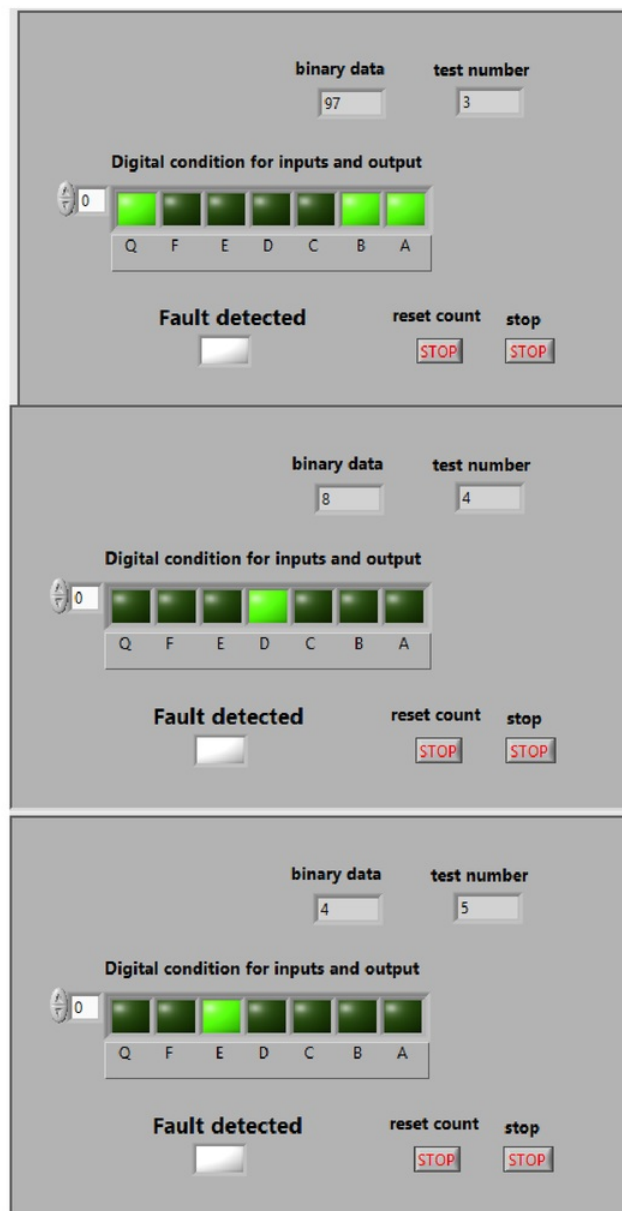


Figure 35- LabVIEW tests 3 - 5

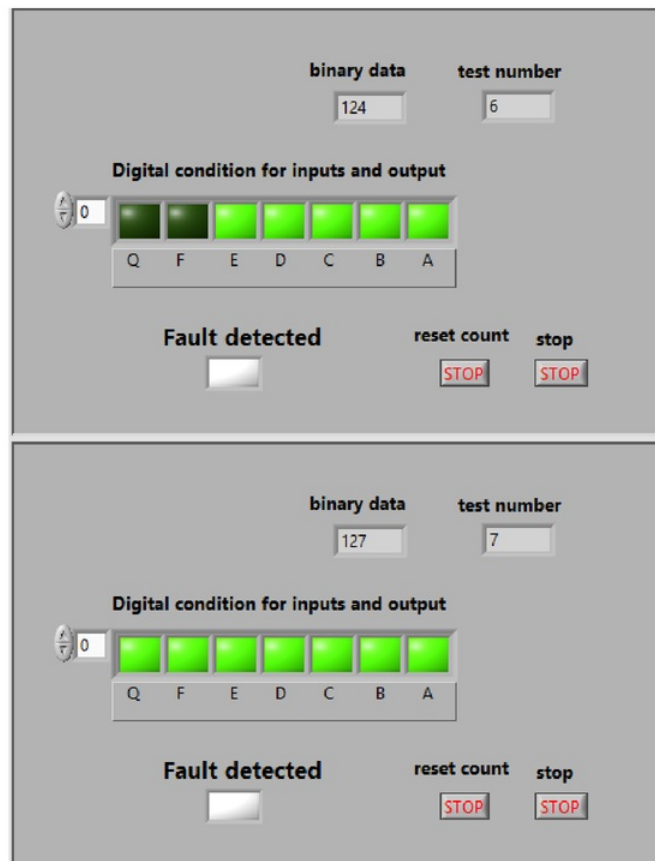


Figure 36- LabVIEW tests 6-7

Test Result

The LabVIEW program functions exactly as expected. All 8 tests ran correctly displaying the correct test number and binary data. The fault didn't indicate during the tests.

Now that the program is known to be functioning correctly, faults must be forced to ensure the software detects these faults as it should.

LabVIEW fault testing

To test the LabVIEW circuits fault detection capabilities a total of 3 tests will be forced onto the circuit. Stuck-at faults, that were researched earlier, will be the method used to put the circuit into fault. For a SA1 fault, the corresponding signal wire will be connected to the positive voltage rail. For the SA0 fault, the signal wire will be connected to the 0VDC rail.

The specific tests to be carried out are: Input A SA0, input D SA0 & input F SA0.

Test 1 – Input A SA0

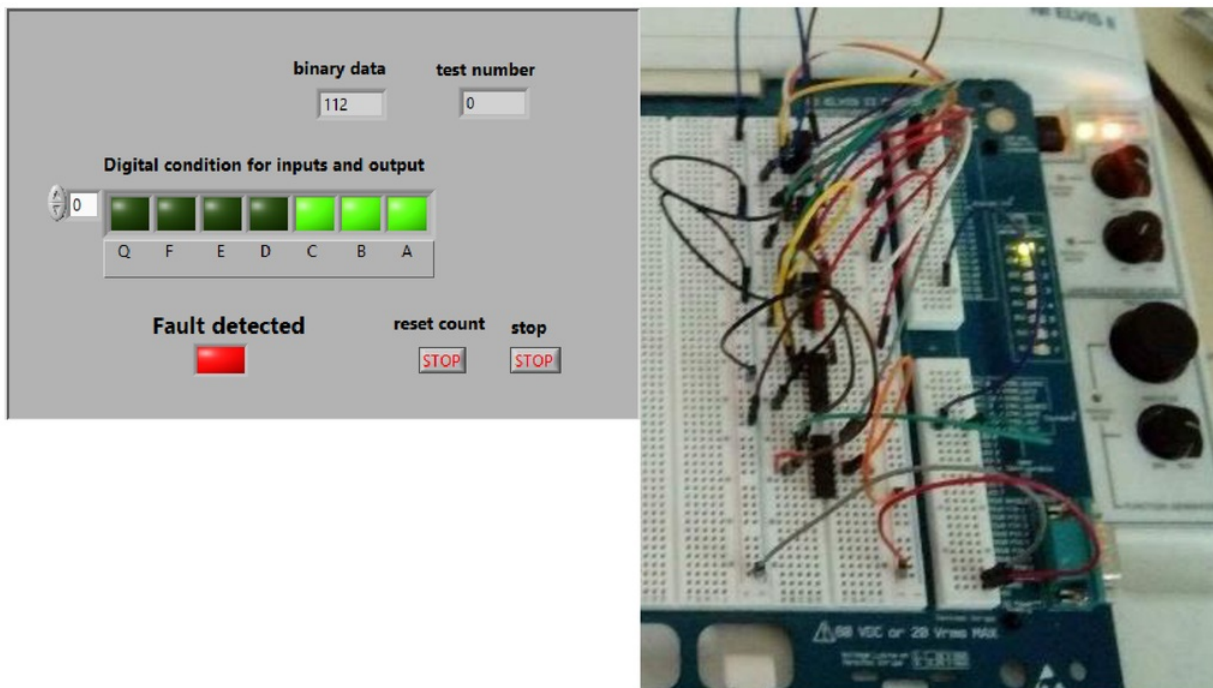


Figure 37- LabVIEW fault testing, test 1

Test 2 – Input D SA0

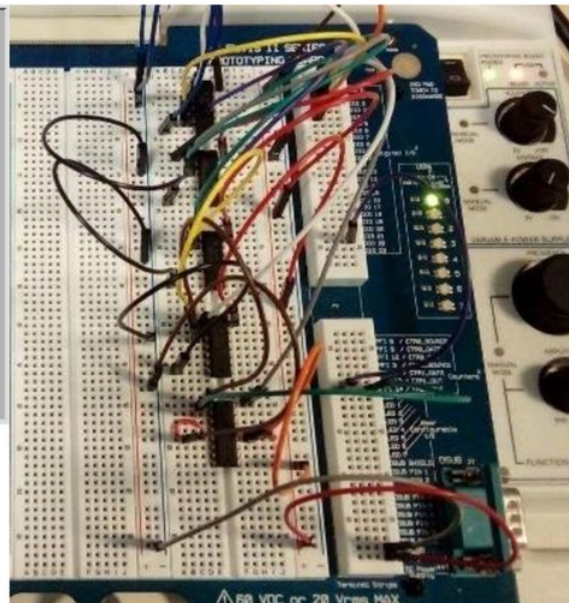
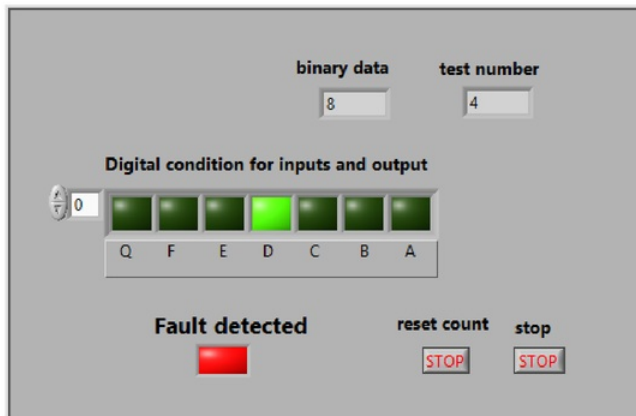


Figure 38- LabVIEW fault testing - test 2

Test 3 – Input F SA0

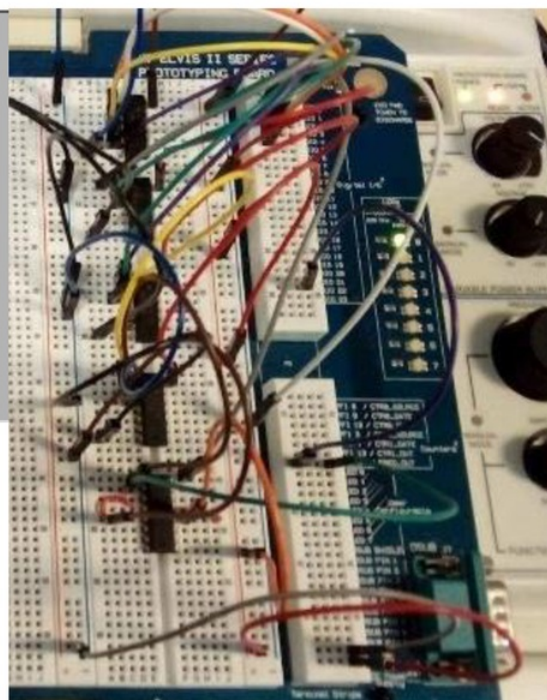
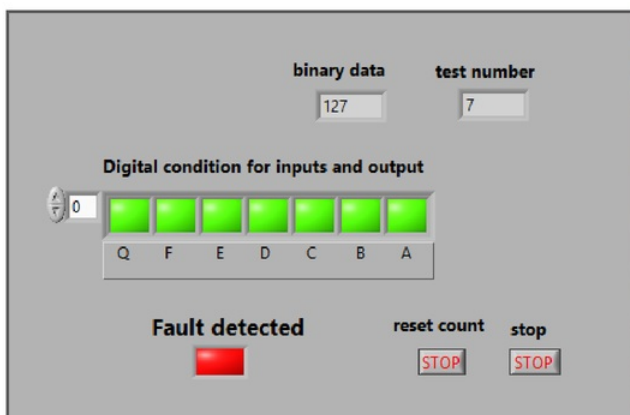


Figure 39- LabVIEW fault testing - test 3

LabVIEW fault testing results

Proved with the evidence provided, the LabVIEW circuit correctly detected all three stuck-at faults that were put onto the circuit.

The 'fault detected' light illuminated on the user interface screen to prompt the user that the circuit is faulty.

Running the correct circuit and faulty test were all observed in the college Lab by the lecturer.

The LabVIEW circuit meets all criteria set out in the introduction, and functions exactly as specified.

Very good analysis here using a good understanding of LabVIEW with DAX

7. Conclusion

The research section of this assignment was essential to completing the project and adhering to the initial specification. Many topics that were researched were put into practice in the simulation, building and most notably the stuck-at fault model and testing of IC faults. The ESD research allowed steps to be taken to reduce the risk of damaging components when building the physical logic circuit. This information is incredibly useful when building or working with ESD sensitive equipment and components.

The design of assignment requirements section shows how a very large number of initial test vectors can be reduced to a minimal number of test vectors. Various methods were used to reduce the table to its final 8 test vectors:

The first reduction method was to omit any rows from the truth table that were unnecessary i.e. for certain gates bits were retested that did not affect the logic gates output. Specifically, the two input AND gate have 4 states from the truth table, but only 3 states need to be put into the derived test table without repeating tests.

Converting the truth table for the logic circuit into HEX values that represent each test, duplicate tests were easily spotted as they had matching HEX values. Duplicates were removed so none of the HEX values matched in the table.

The reduced test table was marked with * and **, which indicates that the binary values can change as the rules that were specified are followed. One * indicates only 1 marked bit must be logic 0. Two * indicates at least 1 of the bits must be a logic 1. Using these rules, the bits were modified to make more matching HEX values. The duplicates that were created are then removed as they have been made redundant.

Using these steps, the initial test table was reduced to a relatively small 8 test vectors.

The testing of the Multisim circuit built upon the knowledge gained throughout the research. 3 stuck at faults were put onto the circuit, the resulting logic analyser waveforms were analysed, annotated and explained which helped solidify some of the theory from the research. The HEX values on the derived test table proved useful again as these HEX values were directly entered into a word generator in Multisim, which allowed a rapid and efficient test of the circuit.

An alternative method to testing the logic circuit was explored using shift registers and creating Karnaugh maps to produce next state equation for each of the input bits to the logic circuit. This 6 bit pattern generator circuit was only designed and built on Multisim without any testing of the circuit.

The point of this was to investigate other methods of testing that would achieve the same outcome. Although the shift register circuit was just as effective as using a word generator, much more time is needed for this method as the derivation of the Boolean expressions and the construction of the logic circuit. For this reason, it's the authors opinion that the best method in this example was the word generator. The testing was effective and has the advantage of a quick setup time. But, the use of a word generator or similar equipment is not always available so exploring a different method of testing is advantageous.

The LabVIEW program was another method of generating a pattern to test the logic circuit. Due to the fundamental differences in LabVIEW and Multisim, the pattern generator constructed on LabVIEW was much more compact and contained far less components than its Multisim equivalent. The LabVIEW program also allowed the test vector 'binary data' values to be easily changed, simply by modifying one of the numbers held in the 'binary data' array. The process for changing ONE test vector value in the Multisim circuit (i.e. change test 1 from HEX 70 to HEX 2F) means a complete redesign of the Karnaugh maps and the resulting logic circuit. The circuit in the assignment only had 6 Boolean equations to be derived from the 6 input bits, so the redesign would only have a relatively small-time impact. If the circuit had say 20 bits, the redesign of the circuit to change one test vector would be substantial. Therefore, it's the authors opinion that LabVIEW is the best software to use for this type of circuit. Not only because modifying the circuit is simplified, but the circuit is far less complex.

Overall, all tasks carried out met the criteria set-out in the initial specification. All tests passed successfully.

Recommendations

Generally, only 3 tests were asked to be carried out on each circuit, all of which passed correctly. Claiming that the circuit has a 100% test coverage may sound deceiving as the further reduced test table had remaining tests that had not been carried out! As only 3 tests were asked for, this extensive testing was outside the scope of the assignment and would have required more time to carry out. The stuck at fault model achieved an 100% test coverage rate, **based on the number of tests carried out**. If time would have allowed, it would have been helpful to truly test ALL the test vectors that had been derived, using the stuck at fault model. If that was the case, one could confidently state that the fault coverage for the circuit was 100%.

*Very
good
conclusion*

8. Appendices and references

Bibliography

Andrew Grochowski, D. B. T. R. V. a. K. L., 1997. Integrated Circuit Testing for Quality Assurance in Manufacturing: History, Current Status, and Future Trends. *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS*, 44(8), pp. 610-633.

Chang, Y.-W., 2008. *Electronic design automation*, Taipei: Department of Electrical Engineering National Taiwan University.

Charles F. Hawkins, J. M. S. A. W. R. a. F. J. F., 1994. Defect Classes - An Overdue Paradigm for CMOS IC Testing. *INTERNATIONAL TEST CONFERENCE*, 18(3), pp. 413-425.

Cline, T. D. a. R., 2014. *Electrostatic Discharge (ESD)*, Dallas: Texas Instruments Incorporated.

Electrostatic Solutions Ltd , 2012. *ESD Sensitivity of electronic devices*. [Online] Available at: www.electrostatics.net/ESD_Guide/technical/ESD_sensitivity.htm [Accessed 25 04 2018].

ESD Association, 2018. *Fundamentals of Electrostatic Discharge*, Rome: ESD Association.

ISRAEL KOREN, F. I. a. Z. K., 1998. Defect Tolerance in VLSI Circuits: Techniques and Yield Analysis. *PROCEEDINGS OF THE IEEE*, 86(9), pp. 1819-1836.

Kabisatpathy. P, B. A. S. S., 2005. FAULT AND FAULT MODELLING. In: *Fault diagnosis of analog intergrated circuits*. Verlag US: Springer, pp. 23-39.

Mini-Circuits, 2015. *The Prevention and Control of Electrostatic Discharge (ESD)*. [Online] Available at: <https://ww2.minicircuits.com/app/AN40-005.pdf> [Accessed 22 4 2018].

O. A. Petlin, S. B. F., n.d. *Detecting Fabrication Faults in C-elements*, Manchester: University of Manchester.

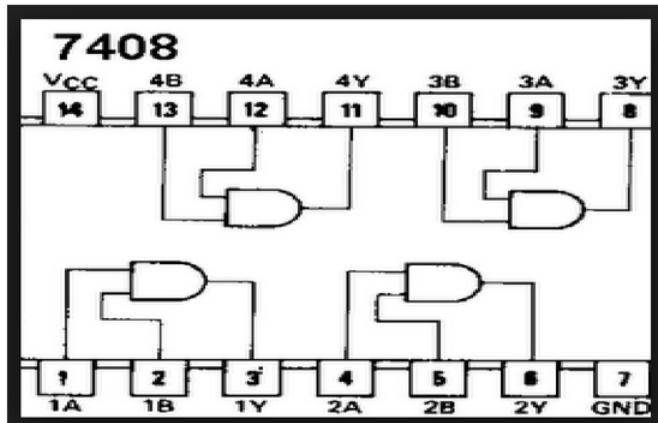
R. H. Khade, D. C., 2014. A REVIEW OF METHODOLOGIES FOR TESTING AND LOCATING FAULTS IN INTEGRATED CIRCUITS. *International Journal of Emerging Trends & Technology in Computer Science*, 3(6), pp. 239-241.

ZiLOG Inc., 2001. *Preventing Electrostatic Discharge*, Austin: ZiLOG Inc..

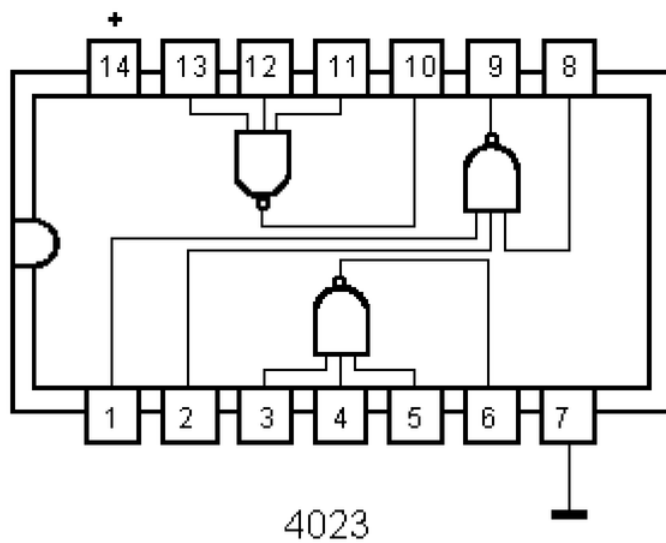
Appendix

The references for the datasheets have been omitted because multiple sources are available that supply these parts.

AND gate 2 input

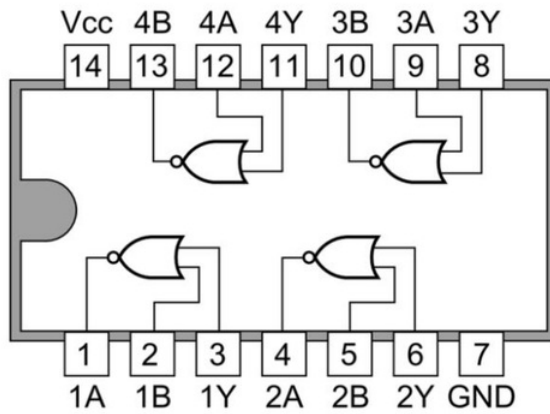


NAND gate 3 input 4023



NOR gate 2 input

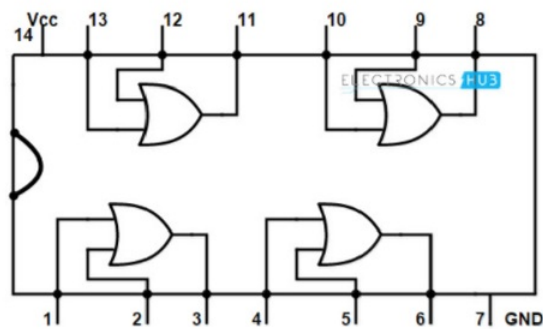
7402 Quad 2-input NOR Gates



OR gate 2input

7432 Quad 2-input OR Gate IC

The 74 series IC is used as Quad 2 input OR gate. The IC diagram of 7432 is shown below.

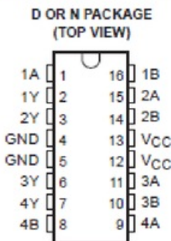


XOR gate 2 input

74AC11086
QUADRUPL 2-INPUT EXCLUSIVE-OR GATE

SCAS081A – NOVEMBER 1989 – REVISED APRIL 1996

- Center-Pin V_{CC} and GND Configurations Minimize High-Speed Switching Noise
- EPIC™ (Enhanced-Performance Implanted CMOS) 1- μ m Process
- 500-mA Typical Latch-Up Immunity at 125°C
- Package Options Include Plastic Small-Outline (D) Packages and Standard Plastic 300-mil DIPs (N)



description

This device contains four independent 2-input exclusive-OR gates. It performs the Boolean function $Y = A \oplus B = \overline{A}B + A\overline{B}$ in positive logic.

A common application is as a true/complement element. If one of the inputs is low, the other input is reproduced in true form at the output. If one of the inputs is high, the signal on the other input is reproduced inverted at the output.

The 74AC11086 is characterized for operation from -40°C to 85°C.

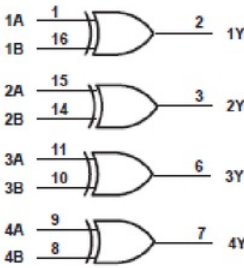
FUNCTION TABLE
(each gate)

INPUTS		OUTPUT
A	B	Y
L	L	L
L	H	H
H	L	H
H	H	L

74AC11086
QUADRUPL 2-INPUT EXCLUSIVE-OR GATE

SCAS081A – NOVEMBER 1989 – REVISED APRIL 1996

logic diagram (positive logic)



Go
od
rel
ev
ant
ap
pe
ndi
x.

Ass 2 EPT final

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

Instructor

84/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

Text Comment. Good list of contents

PAGE 6

Text Comment. Good introduction which shows your understanding of the tasks.

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

Text Comment. Good amount of research and relevant sources. Examples are included and explanations are logical.

PAGE 13

Text Comment. These are the fault models of the gates. How were these derived? There is a formula for this.

PAGE 14

PAGE 15

Text Comment. Good table derived from the above resultant models.

PAGE 16

Text Comment. Good explanation and table reduction methods. The correct conditions are identified for elimination from the model.

PAGE 17

Text Comment. Some ingenuity shown here with the use of the word generator.

PAGE 18

Text Comment. A very good analysis here and some ingenuity demonstrated.

PAGE 19

PAGE 20

PAGE 21

PAGE 22

Text Comment. Excellent analysis demonstrated here with reference to the table.

PAGE 23

PAGE 24

PAGE 25

PAGE 26

Text Comment. Good work and good attention to detail.

PAGE 27

Text Comment. The resultant conditions for the LFSR feedback are the result of the k-map reduction method.

PAGE 28

Text Comment. Good analysis. The EX-OR should be included here and its function explained.

PAGE 29

PAGE 30

Text Comment. What about the function of the FOR loop?

PAGE 31

PAGE 32

PAGE 33

PAGE 34

PAGE 35

PAGE 36

PAGE 37

Text Comment. Very good analysis here using a good understanding of Labview with DAQMX

PAGE 38

PAGE 39

Text Comment. Very good conclusion

PAGE 40

Text Comment. References are good with relevant sources

PAGE 41

PAGE 42

PAGE 43

Text Comment. Good relevant appendix.

GRADING FORM: CSG-RUBIC

MICHAEL LEWIS

0

RESEARCH-INFORMED LITERATURE

IC fabrication faults clearly explained with examples.



4/5 Electrostatic discharge research is extensive with good examples and sources for support. 4/5 SAF is explained very well with examples. 4/5 Good research with good references

12/15

KNOWLEDGE AND UNDERSTANDING



Test vector design is good with appropriate methods used. Very good understanding demonstrated. 8/10 LabVIEW test vector design is excellent with detailed explanation. 9/10 CAD design and methods used are innovative and explained in detail. 9/10 Only minor elements left out of the report. Very good explanations and ingenuity used.

26/30

ANALYSIS



CAD analysis is detailed and correct. Good use of the word generator which shows innovation. 8.5/10 Comparison to theory is very good and explained in a logical manner. 3.5/5 Analysis is excellent. Only minor elements missing in parts such as EX-OR gate function in pattern generator circuit.

12/15

PRACTICAL APPLICATION AND DEP.



Prototyping method of CUT is demonstrated with good logical explanations of procedure. 8.5/10 LabVIEW testing of CUT is excellent. 13.5/15 A very good application of all aspects of the tasks with some ingenuity added.

22/25

SKILLS FOR PROFESSIONAL PRAC.

Application of software is very professional. 4/5



Report is very good and follows a logical sequence.
4/5 Conclusion includes good arguments and
recommendations. 4/5 Very good report

12/15