TABLE 1

DEFAULT LEVELS OF LOGIC TRAINER INPUTS

INPUT	LEVEL	COMMENTS
All A	LOW	
All B	LOW	
C_{IN}	HIGH	No carry in
All CL	LOW	
All D	LOW	
G1, G5, G6, G10	LOW	Unused inputs may be disconnected
G2-4, G7-9, G11-15	HIGH	Unused inputs may be disconnected
GX, GY	LOW	Zero output
Register IN	LOW	Shift in zeros
All J, K	HIGH	Flip-flops count
All L	LOW	Lights normally off
m1, m0	LOW, HIGH	Addition
M1, M0	HIGH, HIGH	Parallel load
All R	LOW	Inactive
Register R	HIGH	Inactive
All S	LOW	Inactive
All X	LOW	
All Y	LOW	

TABLE 2

MODES OF LSI CIRCUITS ON LOGIC TRAINER

(i) AND-OR-SELECT ($Z_i = GX \cdot X_i + GY \cdot Y_i$)

GX	GY	Function
0	0	$Z_i = 0$
0	1	$Z_i = Y_i$
1	0	$Z_i = X_i$
1	1	$Z_i = X_i + Y_i$

(ii) ARITHMETIC

m1	m0	Function (for positive integers)
0	0	S = A plus C_{IN}
0	1	S = A plus B plus C_{IN}
1	0	S = A minus B minus 1 plus C_{IN}
1	1	S = A minus 1 plus C_{IN}

(iii) REGISTER

M1	M0	Function
0	0	No change
0	1	Shift Q0 towards Q3; Q0 <- IN(-1)
1	0	Shift Q3 towards Q0; Q3 <- IN(4)
1	1	Load (Qi <- Di)

1.1 Introduction to Logic Trainer

The trainer uses a small external power supply and can only be turned on or off at the mains switch. The front panel has screen-printed symbols on it for what is inside so please exercise care with the panel.

Interconnections are made between tapered sockets on the front panel, using wires with tapered plugs at each end. The plugs are 'piggy-back' ones, to allow 'daisy-chaining' of inputs. Please do not force them in; a gentle push with a slight twist produces an adequate connection.

It is best not to alter any connections unless the power is off. Check all circuits before turning the power on. Particularly check that two different outputs are not connected, as this can destroy circuits.

To save on the number of connections during wiring, each input has a weak connection (47 kΩ resistor) to either a high or a low level. The level chosen is shown in Table 1 (page 1); the significance of each choice will become apparent later. It is safe to leave any input (or output) disconnected. In that case the input assumes the value of the weak connection. If the input is connected to an output, the weak connection to the input is overridden by the level of the output source.

1.2 Verifying some simple features of the trainer.

1.2.1 Lamp Monitors L0 - L5

The electrical inputs to these stay LOW if disconnected (Table 1) and all six lamps should then be off.

(i) Connect the input to L0 to one of the two sockets labelled HIGH. Lamp L0 should now go on.

(ii) Now connect the input of lamp L0 instead to one of the two sockets labelled LOW ; the lamp should be off.

(iii) Similarly, lamps L1 - L5 come on for a HIGH input and are off for a LOW input.

1.2.2 Logic-level toggle switches (T0 - T5)

These switches allow you to create logic levels for individual signals or groups of signals.

When a switch is down the switch output T is LOW and $\overline{T}$ is HIGH. With the switch up T is HIGH and $\overline{T}$ is LOW. There is also an intermediate position when the switch is exactly upright. In this position, for T0, T1, T2 and T3, T and $\overline{T}$ are both LOW while, for T4 and T5, T and $\overline{T}$ are both HIGH. Note that the switch changeovers are not clean in that the switch bounces as it makes and breaks contacts.

(i) Connect L0 to T0, L1 to T0, L2 to T1, L3 to T1, L4 to T2, L5 to T2. Turn on the power and check that the switches work correctly (including the intermediate position).

(ii) The other three toggle switches are similar.

1.2.3 Push-button switches (P1 - P3)

These switches produce logic-level transitions when the button is pressed and when it is released; they are suitable for producing single steps of counters, registers, etc..

Output P, which is normally LOW when the switch is not pressed, goes HIGH when the button is depressed, then goes LOW when it is released. Similarly P goes from HIGH to LOW to HIGH again.

(i) Connect L0 to P1, L1 to P1, L2 to P2. Turn on the power and check that the push-buttons work correctly.

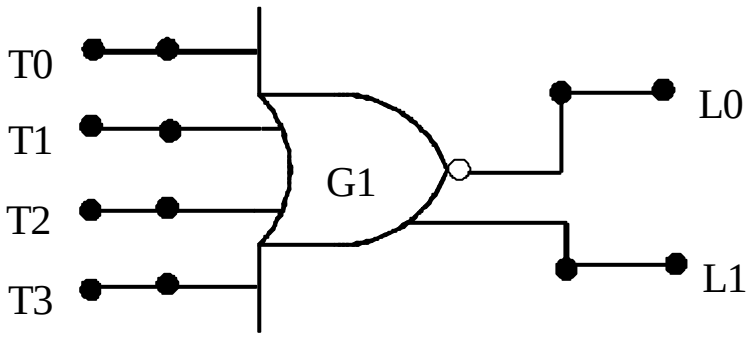
1.3 Verifying gate functions

The trainer has 15 gates G1 - G15 synthesised from available gates. They differ from those usually seen in that they have two outputs rather than just one. Each gate has four inputs on the left and two outputs on the right, where each output of that gate produces an electrical signal which is the inverse of the other.

However, before continuing, we are going to put a different interpretation on signals. So far, we have considered electrical behaviour by considering signal voltage levels, which may be LOW or HIGH. From now on we will regard the signals as representing logic levels, which may be either of the two elements 0 and 1 of Boolean algebra. We will use what is called positive logic, which is the conventional way of doing it, where 0 corresponds to LOW and 1 corresponds to HIGH. This signal can be used to represent a binary digit, or bit, whose value may be 0 or 1, or we can interpret it as truth value where 1 means True and 0 means False.

1.3.1 Gate G1

Connect the following circuit:



(i) Note the levels (0 or 1) of the two outputs for all 16 possible input-level combinations and check that the following table is correct:

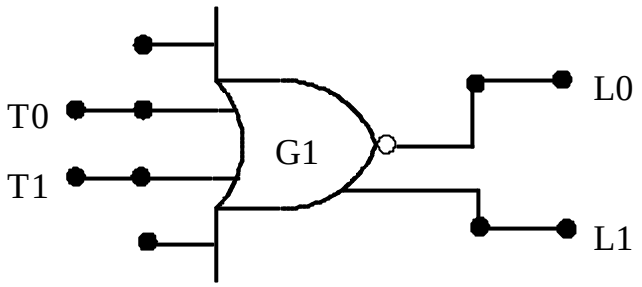
T0	T1	T2	T3	Upper output (inverted)	Lower output (non-inverted)
0	0	0	0	1	0
0	0	0	1	0	1
0	0	1	0	0	1
0	0	1	1	0	1
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	0	1
1	0	0	0	0	1
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	0	1
1	1	0	0	0	1
1	1	0	1	0	1
1	1	1	0	0	1
1	1	1	1	0	1

(ii) The lower output produces an OR function. If any one or more of the inputs is 1 then the output is 1 as well. Only when all inputs are 0 is the output 0.

(iii) The upper output is the opposite and produces the NOR function. Only when all inputs are 0 is the output 1.

1.3.2 Disconnected inputs

Normally it is bad circuit design practice to leave inputs disconnected. However, the presence of weak connections on the inputs in the logic trainer allows us to create gates with smaller numbers of inputs by leaving other inputs (apparently) disconnected and allows us to build circuits with fewer external connections. By looking at the logic trainer from the underside you can see the weak connections in the form of resistor connections from gate inputs to either the ground or power supply voltage rails. Connect the following circuit:

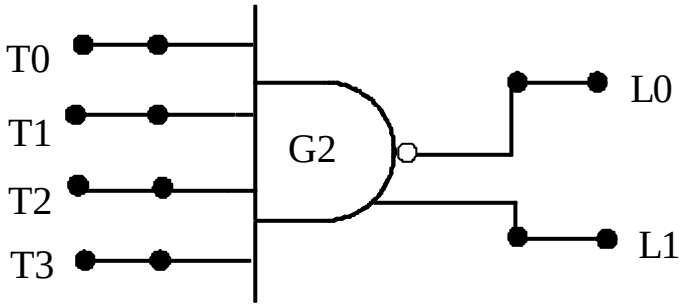


(i) Note the level (0 or 1) of the two outputs for all 4 possible input-level combinations of T0 and T1.

(ii) Verify that the circuit behaviour is identical to the circuit in Section 1.4 if and only if the two disconnected inputs have value 0. This occurs because there is a weak connection made to a 0 underneath the circuit board. It can be overridden by connecting the input directly to another gate output, switch or logic level on the board. Weak connections for gates are listed in Table 1. An alternative name for weak connection is default connection.

1.3.3 Gate G2

Connect the following circuit.



(i) Produce a table similar to the previous table for gate G1 in Section 1.4.

(ii) The lower gate output produces an AND function. The output is 1 only when all inputs are 1.

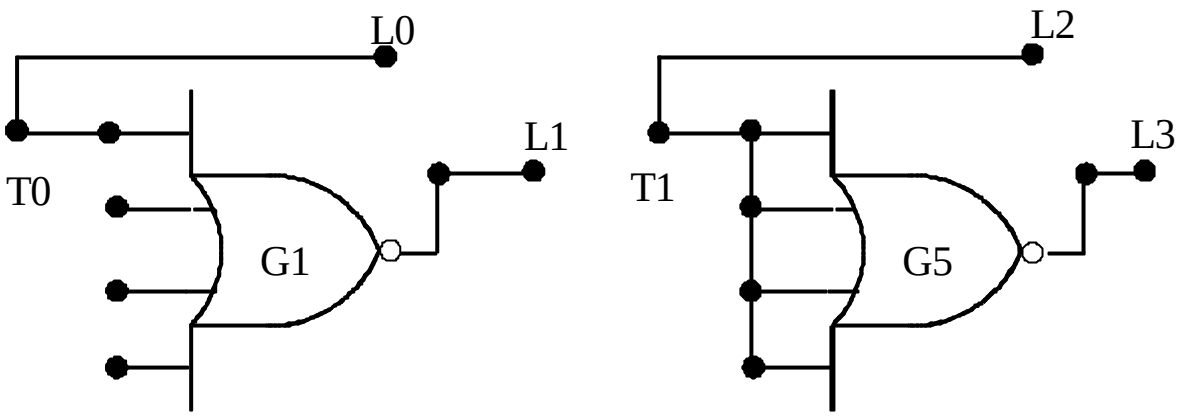
(iii) The upper output produces the NAND function, which is the opposite of AND.

1.3.4 Disconnected inputs

Carry out the same procedure as in 1.5, except that G1 is replaced by G2.

1.5 Inverter

Set up the two circuits below.

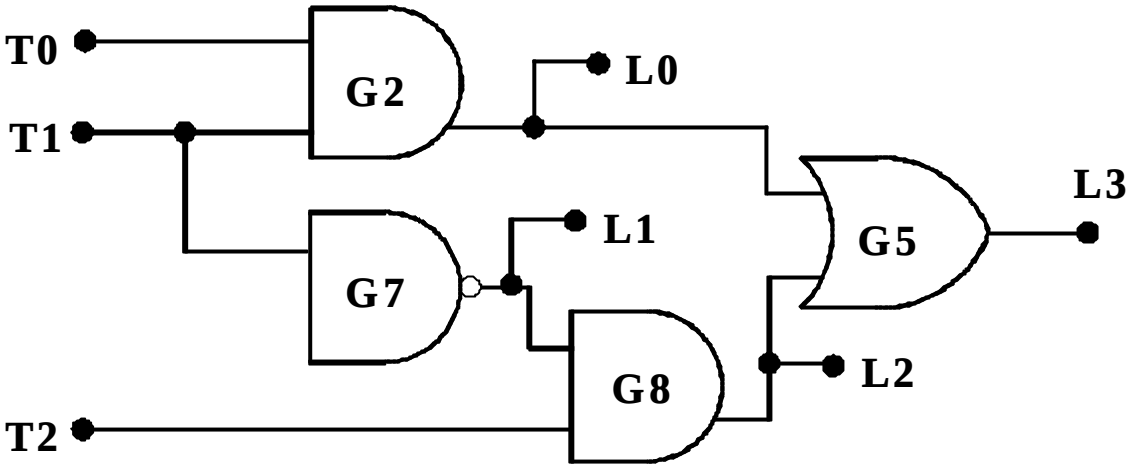


(i) Verify that each of these circuits inverts the input. Explain why in each case.

Similar results are obtained if G1, G5 are replaced by G2, G3 respectively.

2.1 Combining gates

(i) Construct the following circuit:



(ii) Draw the truth table shown below in your laboratory book:

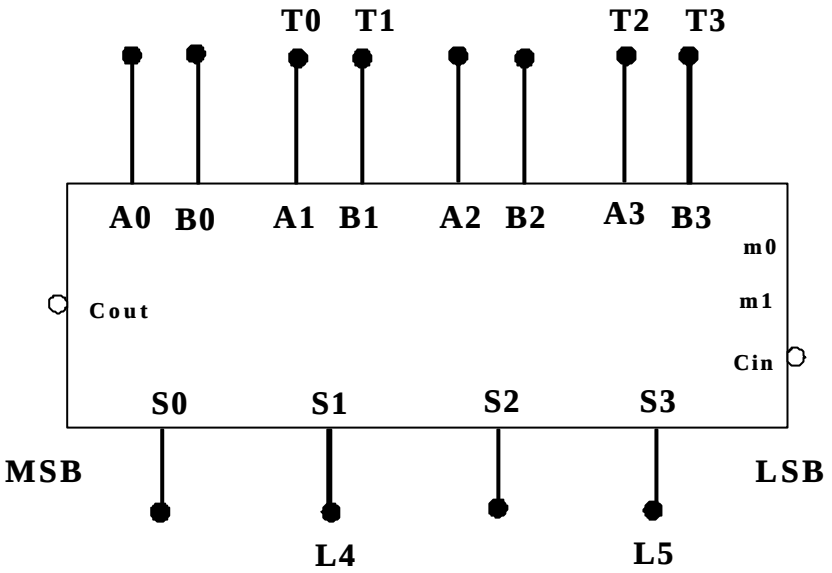
T0	T1	T2	Observed outputs			
			G2 (=L0)	G7 (=L1)	G8 (=L2)	G5 (=L3)
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

(iii) Fill in the above table by observing the outputs of gates G2, G7, G8 and G5 for each possible combination of the inputs, T0, T1 and T2.

3.1 Four-bit adder and exclusive OR gates

The modular arithmetic circuit of the logic trainer is capable of doing four different arithmetic operations depending on the mode inputs m0 and m1 as indicated in Table 1 at the front. We will use the adder in its default mode to implement exclusive OR gates.

Set up the following circuit:



(i) Verify that output S1 (L4) is the exclusive OR of T0 and T1 and that S3 (L5) is the exclusive OR of T2 and T3. Verify that the they do not interfere with each other as well.

(ii) Verify that an exclusive OR gate output is 1 if both inputs are different.

(iii) Verify that the output of the exclusive OR is 1 approximately the sum of its inputs.

(iv) Verify that the output of the exclusive OR is 1 if the number of 1's on the inputs is odd.

Add an inverter to the output of the S3 to make it into an exclusive NOR gate.

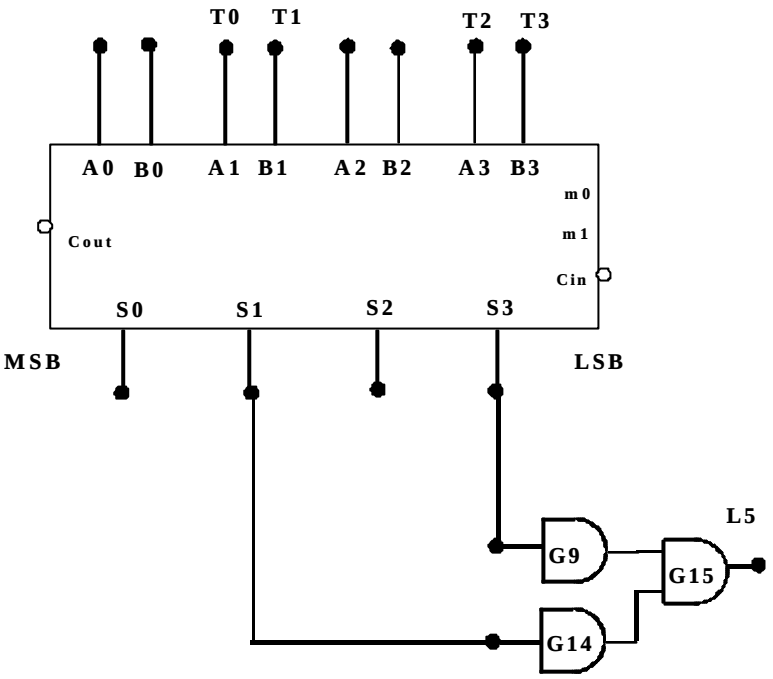
(i) Verify that the output of the exclusive NOR gate is 1 if both inputs are the same.

(ii) Verify that the output of the exclusive NOR is 1 approximately the difference of its inputs.

(iii) Verify that the output of the exclusive NOR is 1 if the number of 1's on the inputs is even.

3.2 Two-bit comparator

Add another inverter to the output of S1 and join the two outputs together using an AND gate to make a two-bit comparator. Verify that the output is 1 only if T0 = T1 and T2 = T3.



3.3 Half-adder

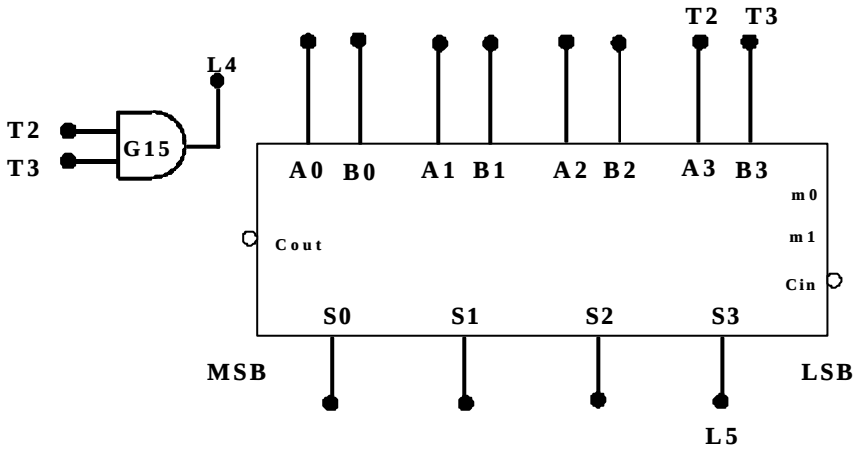
When two one-bit numbers A, B are added, in general, the result requires two output bits called the sum S and carry (Cout) as follows:

A	B	Cout	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

It is clear that

$$\text{Cout} = A \cdot B \qquad \text{and} \qquad S = A \oplus B$$

Connect up the circuit below and verify that performs the half-adder function with inputs A=T2 and B=T3, and outputs sum = L5 and carry = L4.



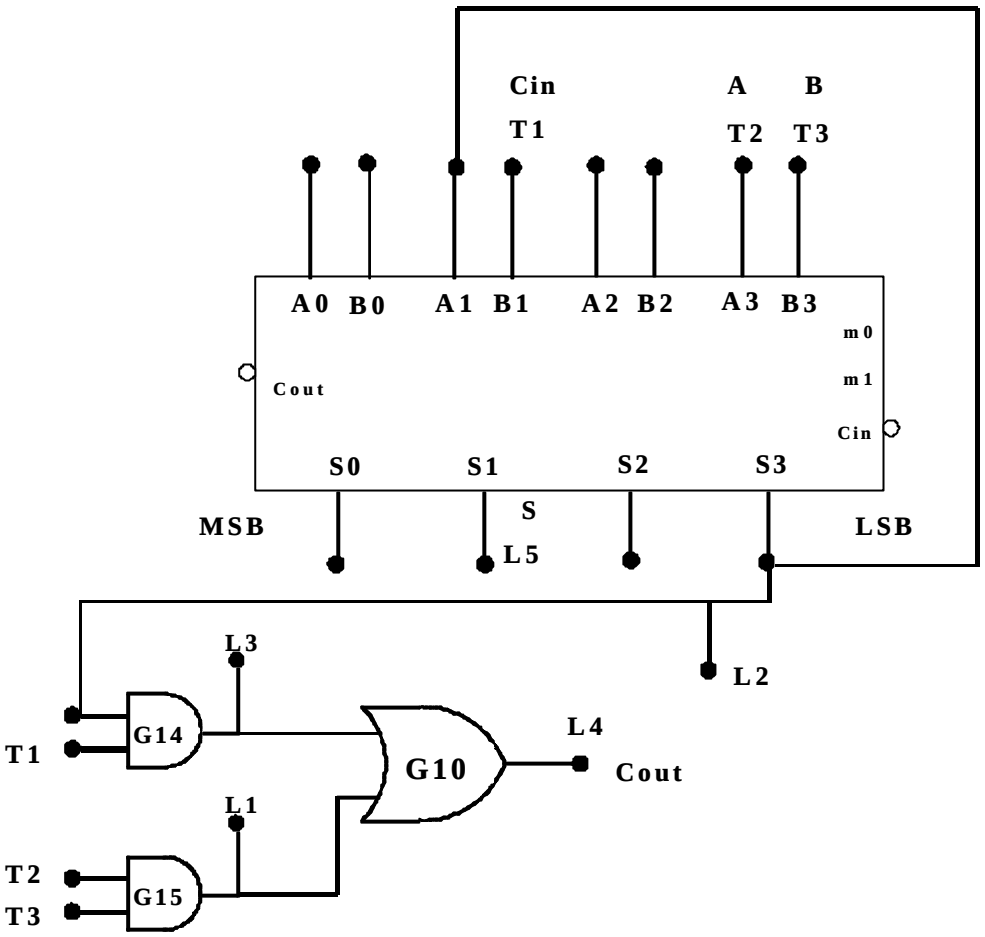
3.4 Full adder

For multi-bit numbers, the half-adder may be used for adding the least-significant bits but the other bits require a circuit that includes a carry-in (Cin) as well, to produce a sum bit S and carry-out bit (Cout).

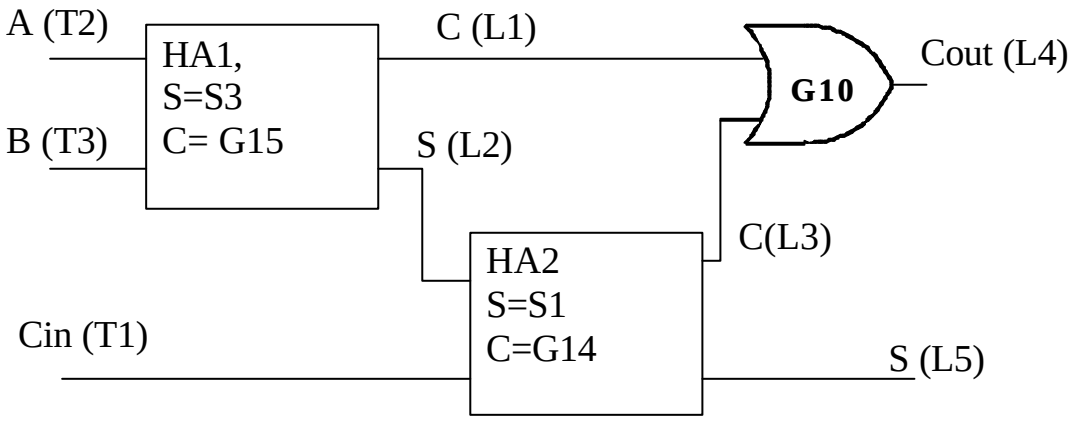
A	B	Cin	Cout	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

The functions S and Cout can be implemented in many different ways.

One way is using two half adders and an OR gate. Construct the following circuit.



The structure of the circuit is shown below.

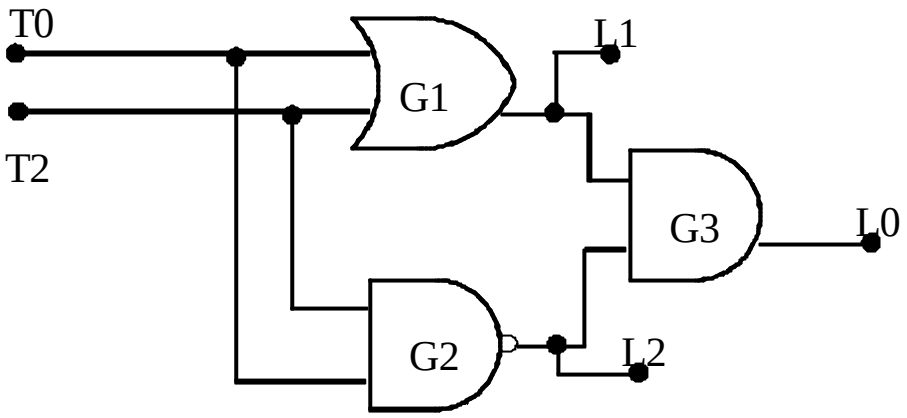


Verify the truth table for the full adder. Examine the output at intermediate points in the circuit (L1, L2, L3) to get a better understanding of how it works.

4.1 Exclusive OR gate from AND and OR gates

The logic trainer board does not have any Exclusive-OR (XOR) gates directly available. One way of obtaining an XOR function is as follows.

Construct the following circuit:



(i) Experimentally derive the truth table for L0 in terms of T0 and T2 by changing T0 and T2 and observing L0. Is this truth table consistent with the definition of an Exclusive-OR gate?

(ii) If G2 were absent then this circuit would behave like an OR gate. Observe L1, for what values of T0 and T2 is it HIGH? Now observe L2, for what values of T0 and T2 is it HIGH? Deduce what the effect of putting G2 into the circuit is (hint: the exclusive-OR gate excludes one of the input combinations of a normal OR gate, which is it?).

(iii) How would you modify the above circuit to produce the Exclusive-NOR function (XNOR) (only one simple change is needed) ?

4.2 Design exercise - Elevator-door control

In many cases the specification of a circuit is not done in such explicit terms as in the previous experiments. Instead, the operation and function of the circuit is specified using verbal descriptions as is encountered in everyday life. The important task in this case is to be able to ***interpret the specification*** *in such a way as to be able successfully design a circuit that meets the specification.*

In this exercise you are to design, build and test a logic circuit that is part of a larger circuit which controls the doors of an elevator. The **description** of the circuit function is as follows.

When the **elevator stops** at a floor the **doors open** and a **timer** is started that will remain on for 10 seconds to allow passengers to get on and off. There is an **electric-eye** detector that ensures that the doors do not accidentally close on anyone entering or leaving. After 10 seconds, if there is no one in the doorway, the **doors will close** if a passenger in the elevator has pressed a button for another floor or someone on another floor has requested the elevator. The doors can be closed prior to the timer expiry, provided that the electric eye does not detect someone in the doorway, by pressing the **"doors close" button** inside the elevator (pressing "doors open" inside the elevator cancels the "doors close" signal). In all other cases the elevator doors remain open. There is also a signal that indicates that the elevator is in **motion** (after the doors have closed) which prohibits the opening of doors regardless of the other signals.

Your circuit will have **input signals** for the expiry of the timer, for the presence of someone in the door via the electric-eye, for someone pushing a button on another floor, for the "doors close" signal inside the elevator and for indicating that the elevator is in motion. It will **output a signal** if the doors are to close. All signals are assumed to be active HIGH, meaning that a Logic 1 (HIGH) indicates that the condition specified is true.

(i) Draw a **block diagram** of the elevator circuit showing the circuit inputs and outputs.

(ii) **Design and build a circuit** which performs the operations required by the specifications. Simulate the input signals using the **switches T0 - T4** and use **L0** to indicate the "doors close" output signal. Draw the curcuit in you practical book.

(iii) Test your circuit for different input signal combinations. Record the test results in your laboratory book.