FESC Laboratory 7: TTL Full Adder

The aim of this lab is to demonstrate how logic circuits can be implementing using prototype board (also called bread board), an example is shown in figure 1. This board contains a number of sockets linked to internal wires allowing you to construct simple logic circuits. At the end of this practical you will understand:

- How to design, layout and construct circuits using breadboards.
- The impact of an IC's input and output impedance.
- Interfacing switches to logic gates and removing electrical noise
- Design and construction of simple logic circuits.

Note, in this lab you will be using the white prototype board with integrated power supply. The blue breadboard shown below will be used in PROM, having the same internal construction as the white prototype board.

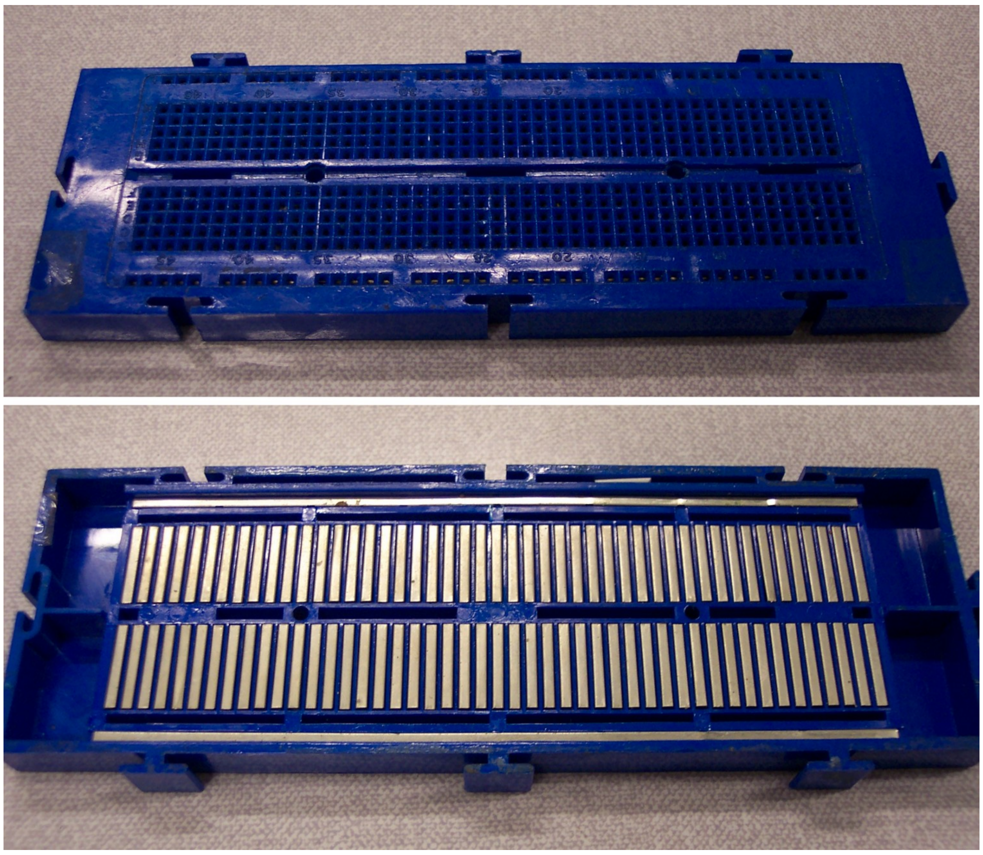


Figure 1: Prototype or bread board, top (top), bottom (bottom) :)

Breadboards contain a 0.1" matrix of holes which are designed to allow integrated circuits (IC) and other components to be easily connected. The electrical connections between these components are made via internal metal tracks embedded in this board, as shown in the bottom frame of figure 1. By convention the top and bottom horizontal tracks are used for the power supply rails, the central vertical tracks for components.

ICs should be placed along the middle gap as shown in figure 2 i.e. each row of pins is connected to a different bank of vertical metal tracks as shown in figure 1. Do not place an IC on one side of the middle gap as this will connect two pins together (a short circuit) and may damage the device.

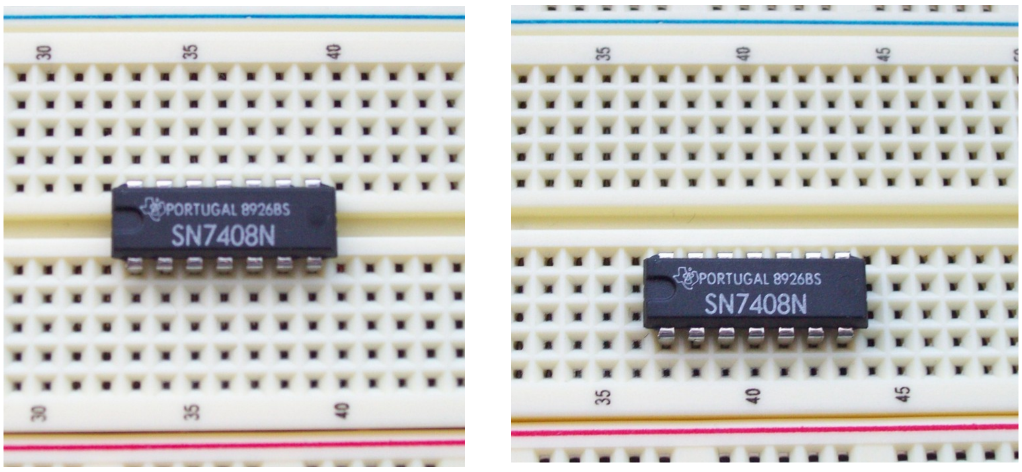


Figure 2: Placing an IC into the prototype board, correct (left), wrong (right)

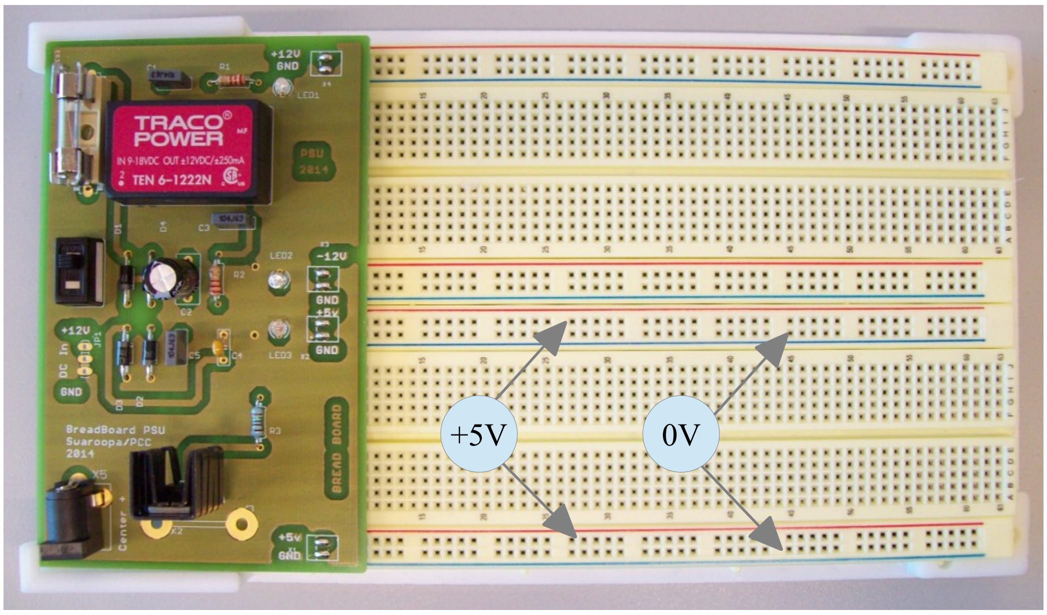


Figure 3: Top and bottom +5V 0V rails

Along each edge of the bottom bread board are to parallel rails supplying +5V (**RED** rail) and 0V (**BLUE** rail), as shown in figure 3. Components are inserted between the two banks of vertical tracks, each track having five holes. When connecting components to the supply rails use **RED** wires for +5V and **BLACK** wires for 0V (also referred to as GND). Input and output signals should use a different colour e.g. **GREEN** wires for inputs and YELLOW wires for outputs.

Note, the top bread board shown in figure 3 has +12V and –12V power rails and is normally used for analogue circuits.

Tip, when wiring up a circuit choose a colour scheme and **stick** with it as this greatly simplifies testing / debugging, which typical takes more time than construction. Keep wires short, neatness of construction is key i.e. access to ICs and pins for testing will become very import for your next module PROM.

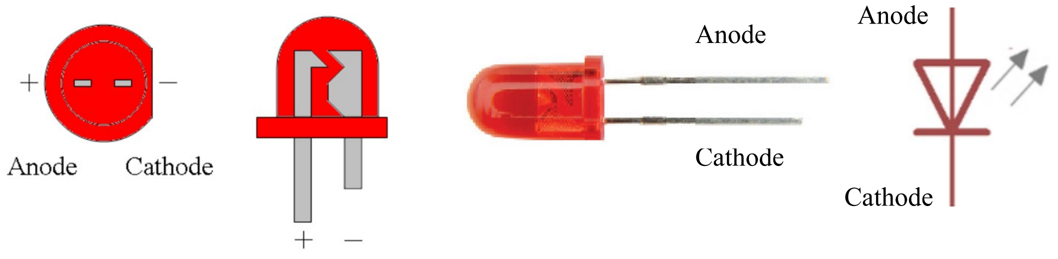


Figure 4 : Example LED IO indicators (left), LED case style / connections (right)

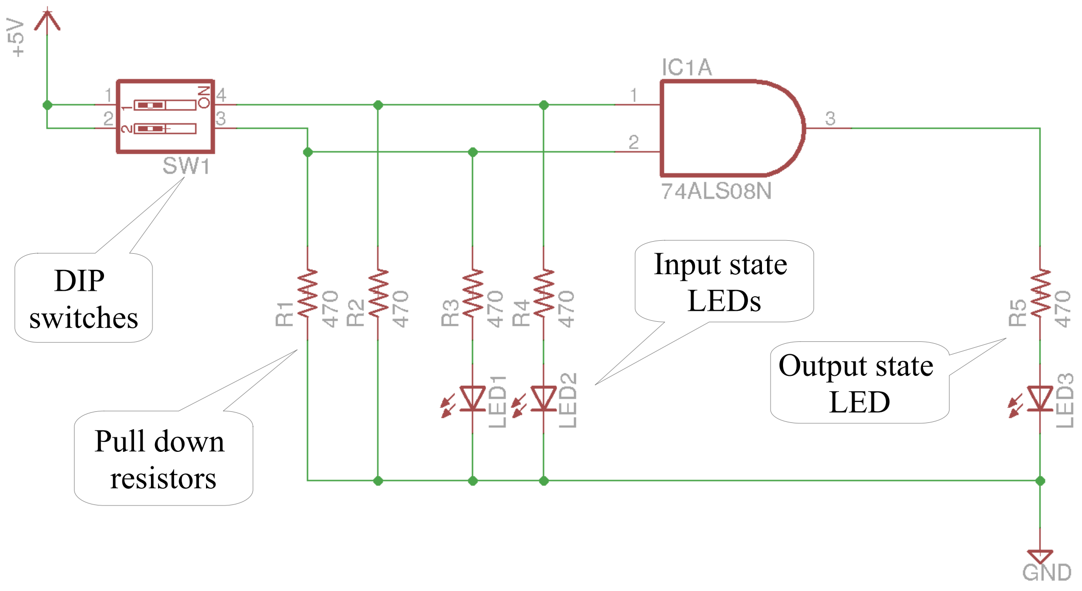


Figure 5 : Test circuit using pull down resistors

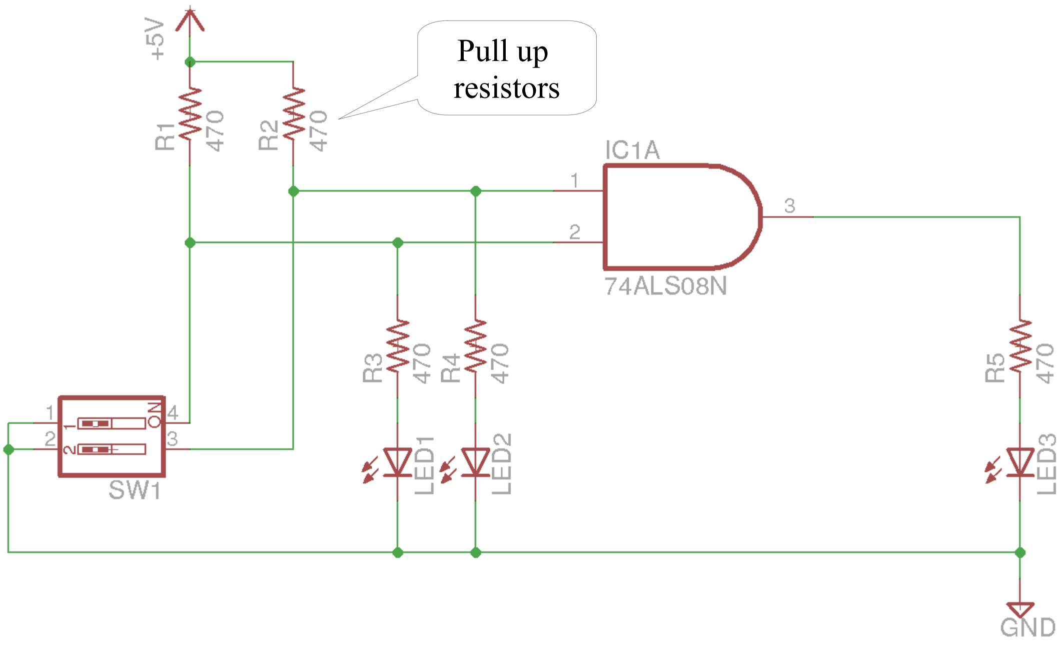


Figure 6 : Test circuit using pull up resistors

Task 1

Construct the logic test circuit shown in figure 5 using a 7408 AND gate. IC pin-outs can be found in Appendix A. All ICs are stored in the cabinet at the front of the lab, one possible layout is shown in figure 7.

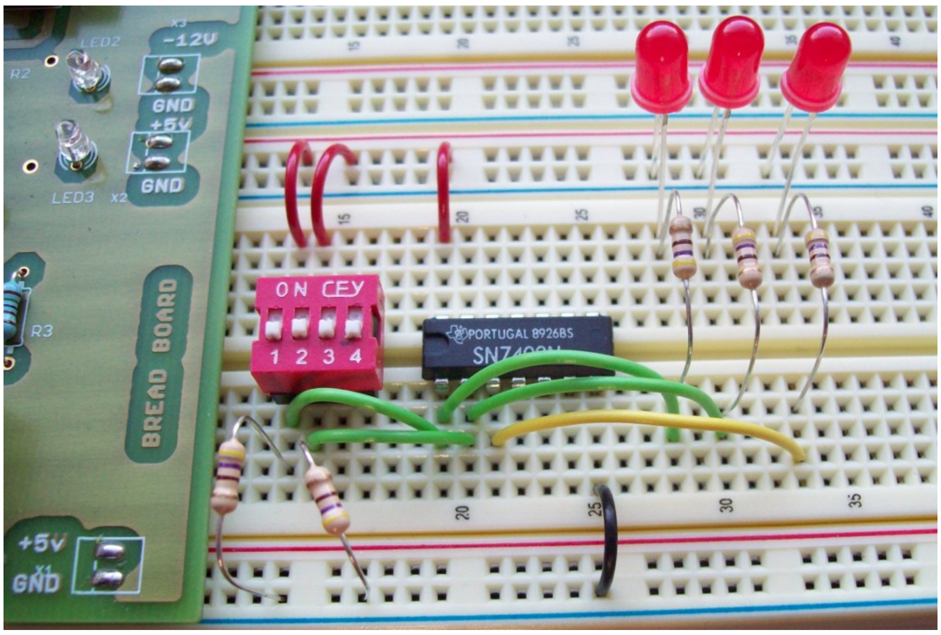


Figure 7 : Final test circuit

To visualise the state of a logic circuit's inputs and outputs, light emitting diodes (LED) can be used. These can be connected to the AND gate's inputs and outputs using a series 470Ω resistor as shown in figures 5 & 6. A LED has two pins, an anode (+) and a cathode (-). These can be identified by the LED's internal construction, the case cut-out (flat edge), or the length of its terminals as shown in figure 4. To illuminate the LED a positive voltage must be applied to the anode with respect to the cathode.

Test your implementation of the circuit shown in figure 5, apply all possible inputs, ensure that the LEDs and the logic gate are working correctly.

Always use a series resistor to limit the current flowing through the LED, otherwise the LED will be damaged / destroyed (normal < 10mA). If a 470Ω resistor is used, how much current will flow through the LED? Answer: assuming 2V is needed to forward bias the LED :

$$I = \frac{V}{R} \qquad I = \frac{5\text{V} - 2\text{V}}{470} = 6.3\text{mA}$$

This current should not exceed the specified maximum output current for the logic gate used. For the 7408 AND gate the maximum logical '1' output current is 1 mA. Therefore, drawing 6 mA from the output will cause the output voltage to reduce i.e. the logic gate's output driver has a specified power rating, P = VI, therefore, if we take more current than the device's maximum output current, the output voltage must fall i.e. P is fixed, I increases, V decreases.

Question: try measuring the logic gates output voltage using a multi-meter with the LED connected and then unconnected. Does the output voltage increase?

As this output is only used to drive the LED this is not a 'problem'. However, if this output was used to drive another logic gate the reduction in output voltage may cause the other gate to misinterpret the logic '1' as a logic '0' i.e. the output voltage could fall out of the band that defines a logic '1'.

Note, the 7408 AND gates are very robust, it is made from a silicon technology that can take a lot of abuse. More modern device e.g. the Raspberry Pi are made from significantly smaller transistors, therefore, exceeding the specified voltages or currents may cause permanent damage.

Question: what will happen if a 330Ω or 600Ω resistor is used instead of a 470Ω i.e. how will it change the LED brightness? Try replacing 470Ω resistor with a higher value. Next try connecting the output LED directly to the output of the AND gate i.e. with no series resistor. Why isn't it damaged?

Hint: have a look at figure 8 for the answer.

Question: why will connecting the LED with a 50Ω series resistor across the +5V / 0V supply rails be very bad?

IMPORTANT: when testing hardware you may only get one chance, if something is wired up incorrectly the hardware is destroyed. Remember, check, check, then check again before turning on the power supply.

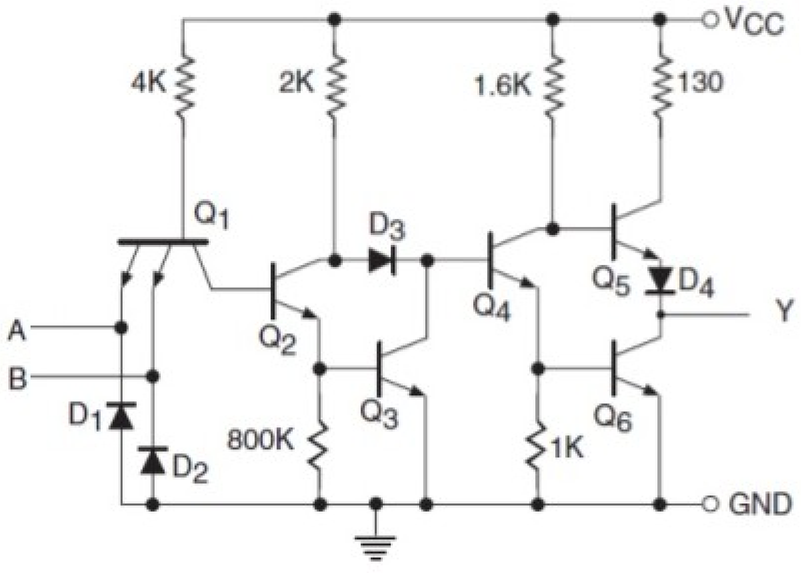
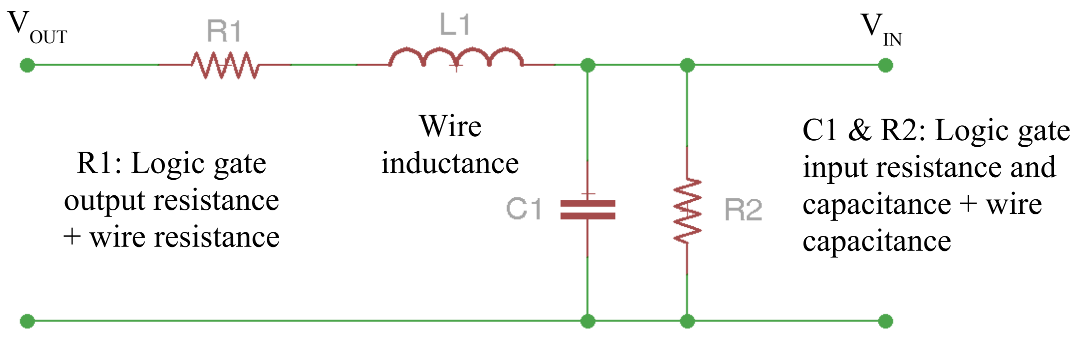


Figure 8 : 7408 AND gate transistor implementation

Question: what is the output resistance of the logic gate when driving a logic '1' or a logic '0' onto a wire connected to its output? Why will these different resistances cause the time for the output to reach a stable logic state to vary?

Hint: the wire connected to the logic gate's output can be represented by the equivalent circuit (model) shown in figure 9. The inputs of logic gates driven by this output will have capacitance associated with it internal construction e.g. 5 pF.

Note, the values of R1, L1 and C1 are very small and the value of R2 is very large, therefore, for 'slow' changing signals they are normally ignored. However, for fast changing signals they start to become very important.



V_{OUT}
R1
L1
V_{IN}
Wire inductance
R1: Logic gate output resistance + wire resistance
C1
R2
C1 & R2: Logic gate input resistance and capacitance + wire capacitance

Figure 9 : Wire model

Question: examine figure 8, to simplify analysis the 0.7V voltage drops caused by the output diode (D4) and transistor (Q5) are ignored the output resistance of the logic gates is equivalent to a 130Ω resistor tied to +5V. If each logic gate has an input resistance of 5KΩ, and the minimum logical '1' voltage threshold is 2V, how many inputs can be driven by one output?

Hint, consider the equivalent circuit shown in figure 10. This shows the 130Ω output resistor driving four 5KΩ inputs. Using the potential divider theorem what is each logic gates input voltage i.e. V_{IN} in the circuit below?

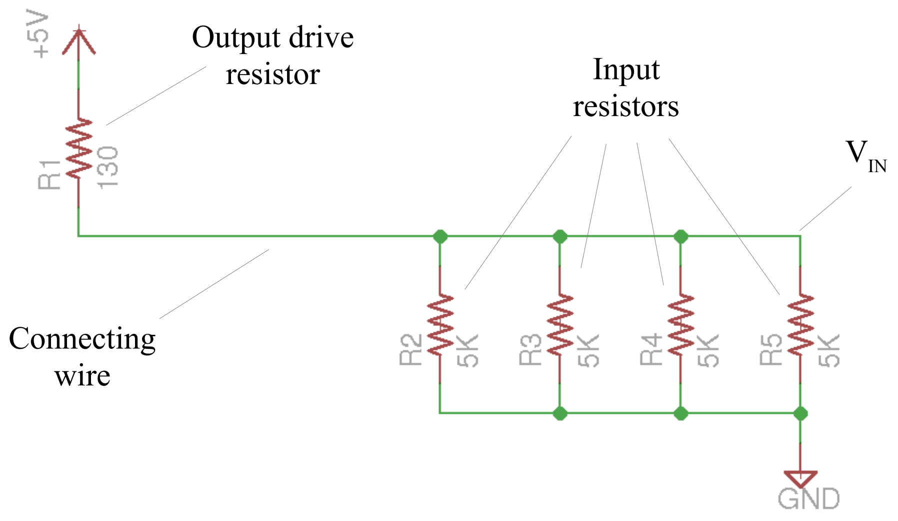


Figure 10 : Output driver equivalent circuit

Task 2

To set the input values of the AND gate in figure 5, small dual-in-line package (DIP) switches are used. These contain a number of on / off switches i.e. single pole, single throw. To ensure that an input is always connect to a valid logic value (voltage) a pull up, or pull down resistor is required i.e. an input with no voltage connect to it (floating) is not the same as one connected to 0v, or +5V. Possible implementations are shown in the original circuit digrams back in figures 5 & 6.

Tip, when placing resistors on the bread board there is a danger that the metal legs will touch i.e. when pushed together, forming a short circuit. To prevent this cut component wires to reduce their length i.e. so that they are mounted flush to the board, or push old wire sleeving onto resistor leads. Always ensure wires are cut to the correct length, minimising the amount of exposed metal to prevent short circuits.

Question: remove the pull down resistors R1 and R2. Does the output change? What does that indicate about the state of the inputs?

Note, if an input is driven by the output of another logic gate it will not require pull-up or pull-down resistors as the logic gate will drive the input to a logic '0' or logic '1'.

Question: replace the pull down resistors R1 and R2 with 47KΩ. Does the output change i.e. return to a logic '0'? What does this indicate about the input resistance of the logic gate?

Hint, if the input resistance is assumed to be 5KΩ what is the approximate equivalent resistances when a 47KΩ or a 470Ω resistor is used as the pull-down resistor i.e. placed in parallel?

Note, remember the rule of thumb for parallel resistors: if you have two resistors in parallel and one is at least 10 times larger than the other, the larger resistor can be ignored, as very little current will flow through it.

Task 3

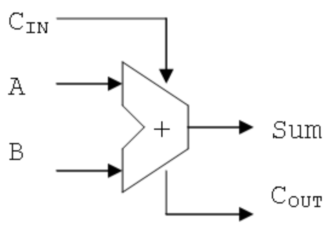
Using this bread board we will now design and construct a full adder. Complete the truth tables for this component as shown in figure 11 i.e. implement the following pseudo code description:

```
If A + B + $C_{IN}$ >= 2          Sum = A + B + $C_{IN}$ mod 2
THEN
      $C_{OUT}$ = 1
ELSE
      $C_{OUT}$ = 0
END IF
```

where A, B and C_{IN} (carry in) are binary inputs, Sum and C_{OUT} (carry out) are binary outputs. If required the final truth table is shown in Appendix B.

Task 4

From the truth table derive the Sum Of Products (SOP) Boolean equations for the outputs Sum and C_{OUT} i.e. the logical states of A, B and C_{IN} for which their associated output is high (logic '1').



A	B	Cin
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Cout

A	B	Cin
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Sum

Figure 11: Full Adder

Sum

Cin \ AB	_00_	_01_	_11_	_10_
_0_				
_1_				

Cout

Cin \ AB	_00_	_01_	_11_	_10_
_0_				
_1_				

Figure 12: K-maps

Task 5

Confirm the SOP equations for a full adder are:

$$C_{OUT} = /A.B.C_{IN} + A./B.C_{IN} + A.B./C_{IN} + A.B.C_{IN};$$
$$Sum = /A./B.C_{IN} + /A.B./C_{IN} + A./B./C_{IN} + A.B.C_{IN};$$

Note, electronic engineers use different notation to represent logic gates i.e. symbols '.', '+' , '^' and '/' to represent the logical functions AND, OR, XOR and NOT.

To minimise the required hardware identify common logic gates (functions) within these equations that can be shared i.e. used in both Sum and C_{OUT}. Using this technique what is the minimum number of logic gates required to implement these equations?

Three and four input logic OR, AND gates may be used. Sketch out the circuit diagram for these new Boolean equations.

Note, you are looking for redundant logic gates (functions) i.e. logic that will produce the same outputs. These can be at the level of redundant AND gate terms, NOT gates or within individual terms i.e. factorising out smaller shared terms using Boolean algebra.

An alternative minimisation method is to use Karnaugh maps (K-maps) as shown in figure 12. Using this technique minimise these SOP equations.

Note, terms can be grouped in 1,2,4 or 8. The larger the grouping the smaller the resultant logic circuit will be.

Task 6

Confirm the minimised Boolean equations for the full adder are:

C_{OUT} = B.C_{IN} + A.C_{IN} + A.B;
Sum = /A./B.C_{IN} + /A.B./C_{IN} + A./B./C_{IN} + A.B.C_{IN};

Karnaugh map minimisation has helped to reduce the number of gates for the `Carry` output, but was unable to help with the `Sum` output. An alternative minimisation method is to use Boolean algebra. Consider the following Boolean relationships:

Distributive Law :	A.(B+C) = A.B + A.C
	A+(B.C) = (A+B).(A+C)
Negation Law:	/(/A) = A
	A+/A = 1
	A./A = 0
De Morgan's Theorem	/(A+B) = /A . /B
	/(A.B) = /A + /B
Exclusive OR (XOR):	A^B = /A.B + A./B
Exclusive NOR (XNOR):	/(A^B) = /A./B + A.B

Using these laws of Boolean algebra minimise these equations:

C_{OUT} = B.C_{IN} + A.C_{IN} + A.B;
Sum = /A./B.C_{IN} + /A.B./C_{IN} + A./B./C_{IN} + A.B.C_{IN};

Hint, using the Distributive law factorise out common terms within the `Sum` equation. Next convert this representation into an Exclusive OR implementation. Finally using the Negation and Distributive laws convert the C_{OUT} equation into a form containing an Exclusive OR logic gate.

Task 7

Confirm the final, minimised Boolean equations for the full adder are:

C_{OUT} = B.C_{IN} + A.(B^C_{IN});
Sum = A^B^C_{IN};

Your final implementation should contain two Exclusive OR gates, two AND gates and one OR gate. If required the final circuit is shown in Appendix C.

Note, the B^C_{IN} term is correct, you could also use B+C_{IN}, as this would produce the same output for this particular Boolean equation. However, this would need an additional logic OR gate i.e. the exclusive OR gates used to produce the `B^C`$_{IN}$ term can be shared with `Sum` equation as shown in Appendix C.

Task 8

To test your construction skills next implement the full adder circuit shown in Appendix C. This requires three ICs (available from the draws at the front of the lab):

- 7408 : AND gate
- 7432 : OR gate
- 7486 : XOR gate

Note, there are **FOUR** logic gates in each IC, as shown in Appendix A.

You will need to plan the layout of your design, things to consider:

- What wire colour could should be used, RED=+5V, BLACK=0V, YELLOW=signals etc.
- There are three inputs A,B and C_{IN} these can be driven by DIP switches
- Which logic gates should be used within each IC to minimise wiring e.g. using the two XOR gates on the same side of the IC, connecting pins 3 and 4.
- The inputs to the AND gates are shared with the XOR gates, perhaps these ICs should be placed close together on the bread-board.
- There are two outputs SUM and CARRY, these can drive LEDs,

Implement your design on the white bread board (with built in power supply). Apply all possible inputs, ensure that the LEDs and the Logic gates are working correctly.

Note, remember to add the pull-down and resistors to the three switch signals and the current limiting resistors for the `Sum` and `Carry` LEDS.

There is a good chance that when you have constructed your circuit it will not work correctly i.e. it does not implement the desired truth table. This could be for a number of reasons:

1. The power supply is not connected / turned on (obvious, but common).
2. The wrong ICs have been used, very small writing, easily misread.
3. The IC is in the wrong way round, pin one indicated by the notch.
4. The IC is not wired to the power supply rails i.e. +5V (pin 14) and 0V (pin 7).
5. The +5V and 0V pins are the wrong way round i.e. pin 14 is connect to 0V.
6. The circuit has been incorrectly wired up, wires connected to the wrong pins.
7. The wire links are too short i.e. the exposed copper wire does not make contact with the internal metal tracks.
8. Component legs of exposed wires are forming accidental short circuits.
9. The LEDs have been wired up the wrong way round i.e. they will not glow when a logic '1' is applied.
10. The IC is damaged, rare, but it can happen. To test an IC use the TTL logic tester on the side bench.

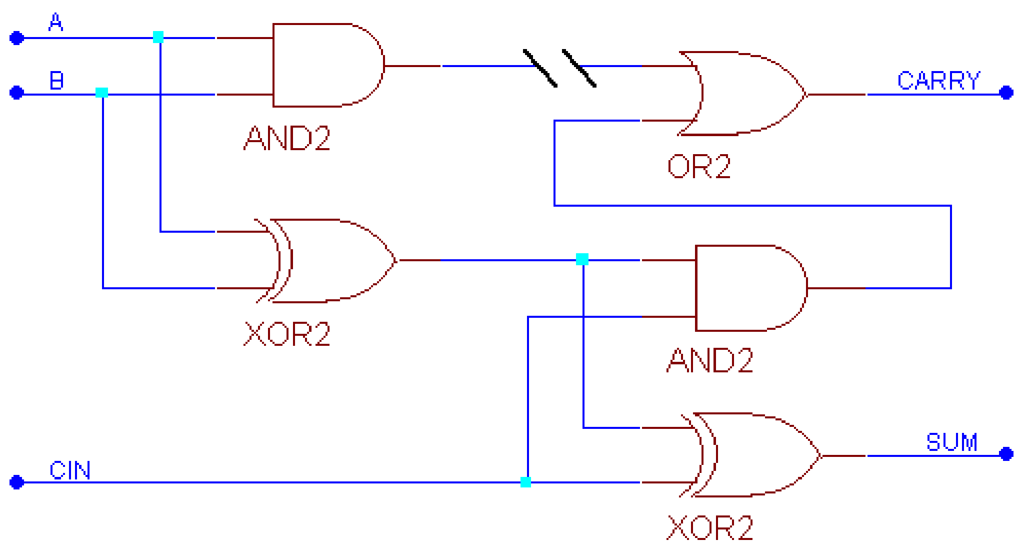


Figure 13: Open circuit wiring fault, open-circuit on OR input

Testing is the process of applying inputs and interpreting outputs. Consider the scenario where the CARRY signal is always high .i.e. independent of the input state. There could be multiple reasons for this output state e.g. one of the inputs to the OR gate is open circuit, missing wire, as shown in figure 13. This would result in the unconnected input floating high, forcing the CARRY out to always be high. To test this hypothesis the output of the AND needs to be driven to a logical 0 state i.e. input A or B must be connected to a logic 0. The output of the AND gate and the input to the OR can now be measured using a multi-meter or an oscilloscope to prove / disprove this assumption.

When testing this circuit ensure that the LEDs and the Logic gates are working correctly. Then apply an input state, trace these signals through the circuit, testing each logic gate's output to see if the circuit is functioning correctly (using multimeter or oscilloscope).

IMORTANT, you will need to develop these skills for the PROM module.

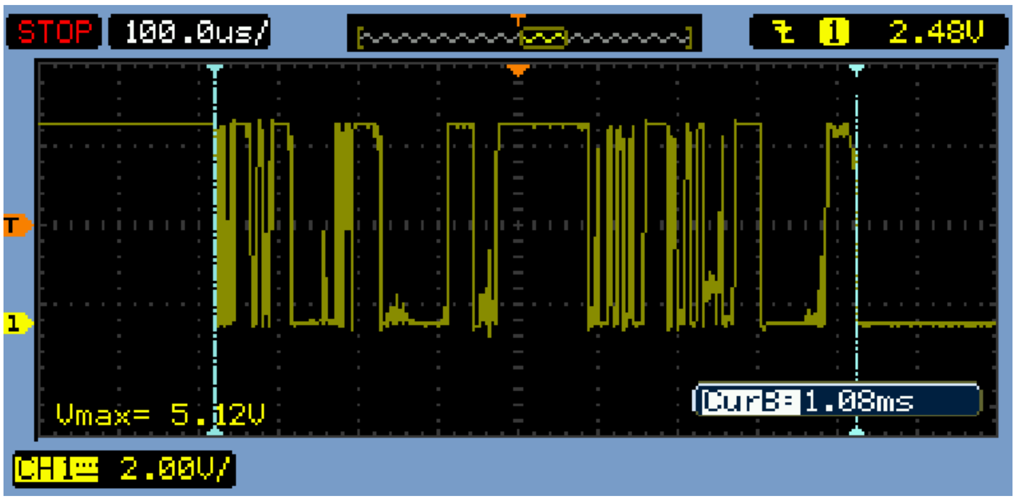


Figure 14 : Contact bounce.

Task 9

When using mechanical switches contact bounce (chatter) can cause problems, as shown in figure 14 i.e. the transition from a logic '1' to a logic '0' causes a series of input pulses. When switch contacts are either opened or closed the mechanical properties of the metal contacts i.e. mass, momentum and spring pressure, can cause them to bounce, repeatedly opening and closing the switch contacts.

Note, normally, this is not a problem e.g. driving the LEDs, but when these signals are used to drive other digital circuits these high speed pulses can be misinterpreted as a valid signals i.e. a switch being pressed multiple times.

To remove these high frequency signals the circuit shown in figure 15 can be used. This circuit contains a low pass filter (LPF) and a Schmitt trigger inverter. The cut-off frequency of the LPF is selected to remove the high frequencies associated with contact bounce, the inverter is then used to 'square' the filtered signal i.e. convert it back to a logic '1' or logic '0' voltage level.

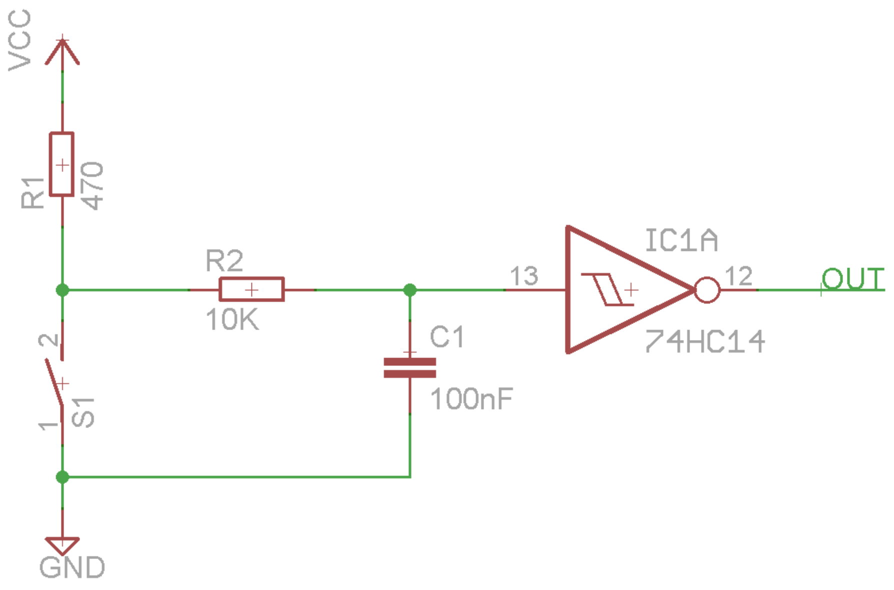
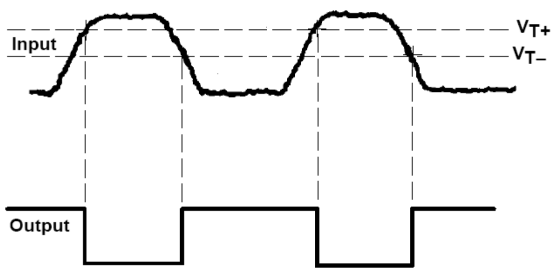


Figure 15 : Switch debounce circuit



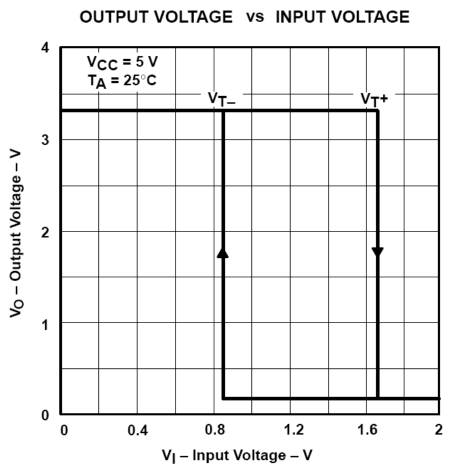
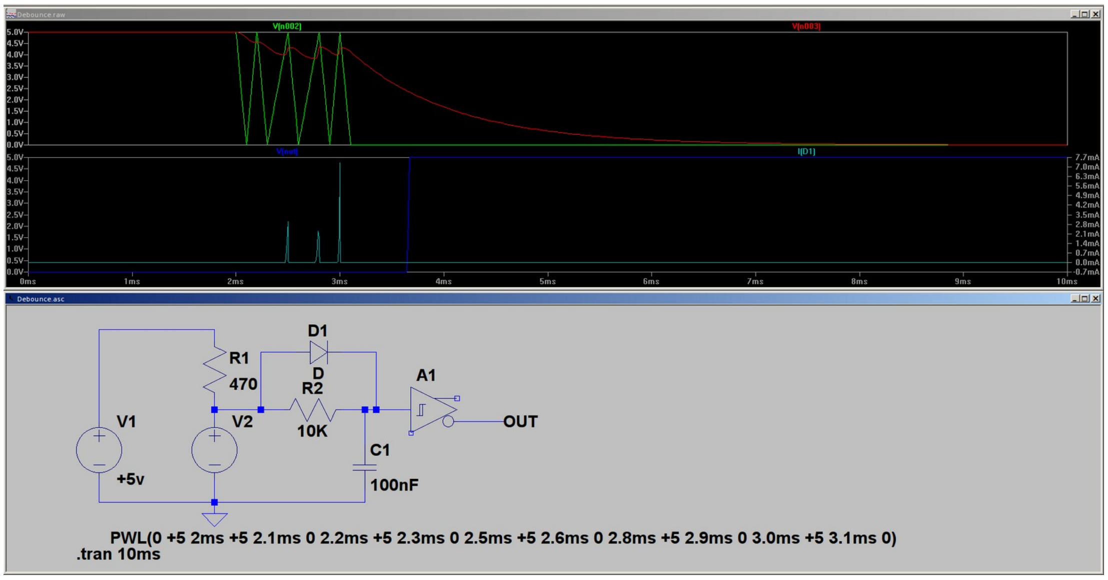


Figure 16 : 7414 Schmitt trigger

A standard inverter (7404) could be used in this circuit, but, for slow moving input signals a Schmitt trigger version (7414) helps reduce output noise by using a variable logic threshold i.e. hysteresis, as shown in figure 16. With a logic '0' input the threshold is set to V_{T+}, when the input voltage crosses this voltage the output switches to a logic '0' and the threshold is reduced to V_{T-}. Moving the threshold away from the input voltage ensures that if the input was to remain at V_{T+} small variation caused by input noise on this signal will not cause the output to oscillate i.e. switch from a logic '1' to a logic '0'. The reverse happens when switching the input from a logic '1' to a logic '0'.

Note, Schmitt trigger inputs should be used when connecting analogue signals to digital inputs e.g. the voltage on the capacitor in the LPF. Schmitt trigger inputs on their own do not remove noise, they only help prevent additional noise from being generated.

As its difficult to predict the amount of contact bounce produced by a switch we can test this circuit in a SPICE simulation, as shown in figure 17. Simulate this circuit by either drawing the circuit diagram using the CAD tools or SPICE model below.



PWL(0 +5 2ms +5 2.1ms 0 2.2ms +5 2.3ms 0 2.5ms +5 2.6ms 0 2.8ms +5 2.9ms 0 3.0ms +5 3.1ms 0)
.tran 10ms

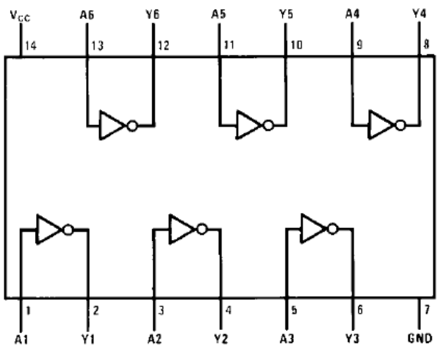
```
* DEBOUNCE CIRCUIT

.lib C:\apps\LTspiceIV\lib\cmp\standard.dio

R1 1 2 470
R2 3 2 10K
C1 3 0 100nF
D1 2 3 1N4148

A1 3 0 0 0 0 OUT 0 0 SCHMITT Vhigh=+5

V1 1 0 +5v
V2 2 0 PWL(0 +5 2ms +5 2.1ms 0 2.2ms +5 2.3ms 0 2.5ms +5 2.6ms 0
2.8ms +5 2.9ms 0 3.0ms +5 3.1ms 0)

.tran 10ms
.end
```

Figure 17 : 7414 Schmitt trigger SPICE simulation

The switches contact bounce is simulated by the piecewise linear voltage `V2`. The circuit shown in figure 17 differs slightly from that shown in figure 15, with the additional of diode `D1`. Can you see why this component was added.

Hint, what is the default input voltage?

Appendix A: Logic Gates

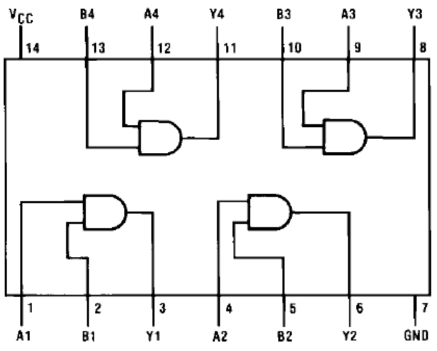


$Y = \overline{A}$

Input	Output
A	Y
L	H
H	L

H = HIGH Logic Level
L = LOW Logic Level

7404 : NOT Gate

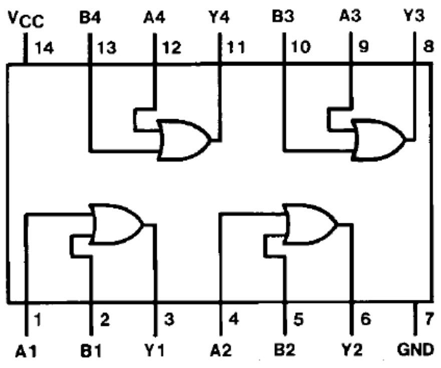


Y = AB

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

H = High Logic Level
L = Low Logic Level

7408 : AND Gate

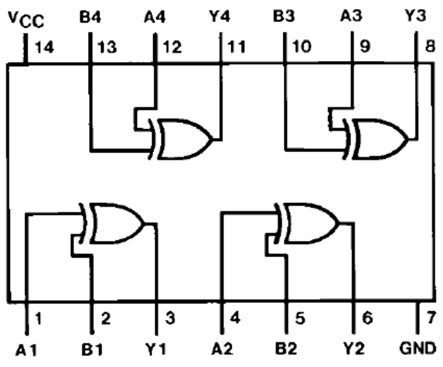


Y = A + B

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

H = HIGH Logic Level
L = LOW Logic Level

7432 : OR Gate



$Y = A \oplus B = \overline{A}B + A\overline{B}$

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

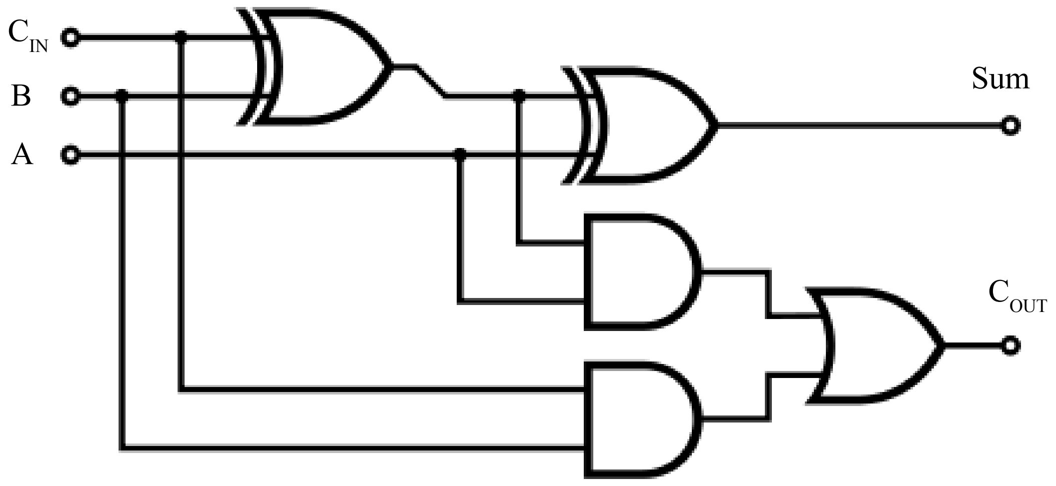
H = HIGH Logic Level
L = LOW Logic Level

7486 : XOR Gate

Appendix B: Full Adder Truth Table

A	B	Cin	Carry	Sum
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

Appendix C: Full Adder Circuit Diagram



```
COUT  = B.CIN + A.(B^CIN);
                              Sum  = A^B^CIN;
```