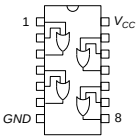
Physics 217A LAB 1 Spring 2016

Basic TTL Gates
Serial and Parallel Binary Numbers
Vocabulary

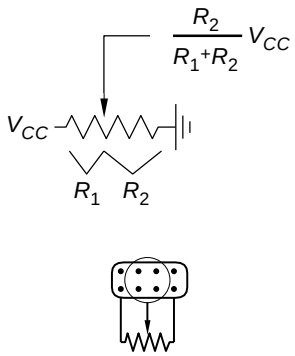


0. You will need about ten short (1–2") connecting wires in both red and black, and at least twelve longer 'signal' wires in other colors. For this class you must adopt two consistent practices: (A) color code your wires using red = +5 V (power, V_{CC}), black = 0 V (ground), and other colors for signal wires; and (B) always place chips with pin 1 up (and to the left). Please note that our color code is *not* that used for (120 V, 60 Hz) household wiring; there black is power and white is ground.

1. Starting in the upper right, press in the following TTL chips: 7400 (quad NAND), 7402 (quad NOR), 7404 (hex INVERTER), 7408 (quad AND), 7432 (quad OR), 7486 (quad XOR). Record in your notebook the exact chip number of each chip used. (I.e., your chip might be a 74ALS00 or an "equivalent" military chip: /30001.) Note that you will lose points if you do not follow this practice in this and all future labs (i.e., always include a 'Parts List' in your notebook). In this particular lab the answers to the below questions will depend on exactly what chip you're using—so here the grader needs to know. Additionally, since you must return each chip to its proper drawer at the end of the lab, a recorded translation between the part number on the chip and the label on the drawer will make finding the proper drawer fast. The 'pin-out' diagrams for these chips may be found on the following pages. Connect power (red, V_{CC}) and ground (black) "at the corners." Pick out one of the two-input gates to investigate. ("Two-input gate": all of the gates on these chips—except the 7404—have exactly two inputs; so use any chip except the 7404 for the following tests.) "Fan out" (i.e., branch) a "side-board" data-switch output to an LED and to a gate input. Do the same for the other gate input using an adjacent side-board data-switch and LED. Display the gate output on a third LED. Check all combinations of gate inputs and record the results in a truth table for the gate. Totally disconnect a chip input and investigate how the chip responds to its remaining input. In your report fill in the blank: "an open (i.e., unconnected) input is treated by the gate as if it were: [ ]" (e.g., as H or L). Move your circuit over to a gate on a different chip and record the truth table for this new gate. Check out and record the truth table for a third gate.

2. Take the wire connecting the output of your gate (whose inputs are still controlled by the side-board switches) to a LED and, instead, connect it to the input of some other gate. (No other connections: just output to one input. This is obviously the usual thing to do: connect an output to an input. Note: an output may control several inputs [this is called 'fan-out'] but an input can only be touched by one output. The disaster that results from failure to follow this rule is called 'wired-OR'.) Using a DMM find (and record) the current that flows through this output-to-input wire when the output is high and when the output is low. (Surprised?) Similarly, using a DMM find the voltage (relative to ground) of this output-to-input wire in both output states. (When you record the results of a DMM measurement, you should always draw the circuit schematic (or schematic fragment) that displays exactly how the measurement was made. For this class measurement errors need not be recorded, but get in the habit of recording every digit displayed by the DMM.)

3. The pot or potentiometer consists of a sweeper arm (denoted by the arrow) that touches a resistive slab at an adjustable location, dividing the resistive slab into two 'resistors'. If the ends of the pot are connected to power and ground (see right), the sweeper samples the voltage in between the the ends; the voltage divider equation could be applied to give the sweeper voltage, but in practice neither R_1 or R_2 is exactly known (although they could perhaps be estimated based on how far the pot had been turned). Pots are labeled with the total slab resistance $R_1 + R_2$. On the protoboard, the sweeper is connected to the central four holes. Never connect power and ground between the sweeper and an end! Never draw appreciable current from a sweeper!



Wire up your right-hand (1 kΩ) pot with ends at +5 V and ground. Set the voltmeter to read the voltage on the sweeper, and connect the sweeper's output to an inverter's input. (You have now made a voltage divider, and you can adjust the sweeper voltage by turning the pot's knob.) Examine the inverter output with the scope. Find the range of input voltages considered by the inverter to be high (i.e., produce a low output). Find the range of input voltages considered by the inverter to be low (i.e., produce a high output). Compare to the values recorded in the book for the particular type of TTL chip you are testing. (See HH p. 796)

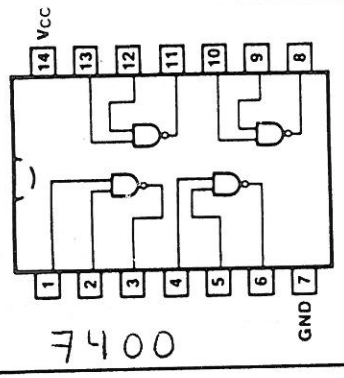
4. Daisy-chain all 6 inverters (i.e., connect output of inverter N to input of inverter $N + 1$ making a chain). Convert 4 additional gates to inverters (report how you did this) and add them to the chain. Drive the chain with the function generator (on TTL!). (A TTL pulse output, like that produced by the function generator, is often called a "clock". The verb "clock" means to drive a circuit with such a signal.) Use the scope to display both the input and the chain output. Sketch the results in your notebook. (Note: when I ask you to make a measurement with a scope, please sketch the scope trace in your notebook and record the *scales* for both axes and the location of ground on the vertical axis. Draw the circuit and show exactly how the scope was connected to the circuit.) Measure the time delay when clocking at maximum speed and compare to typical TTL gate delays. XOR the clock input and chain output. Display the resulting XOR output on the scope and sketch the resulting scope trace. If the clock and chain output were synchronized, the XOR output would be flat. (Why?) The "glitch" in XOR output results from gate delay. Disconnect the added XOR circuit before proceeding to the next part.

5. Disconnect the chain input from the function generator and instead drive the chain with the 9^{th} inverter output. What does the chain output look like in the scope? (Of course, sketch the results in your notebook.) Explain!

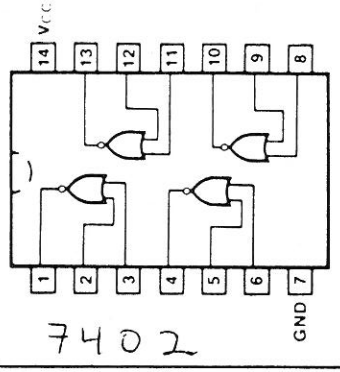
6. Design a majority-rule voting system for a committee of three members. Each member is provided with a yes/no switch; an LED lights if and only if the vote passes. Diagram your circuit in your notebook, wire it up, and show your instructor that it works.

7. Remove the simple gate chips. Wire-up the 7483 (ADDER) using the supplied pin out. This chip adds the 4-bit binary number $A_4A_3A_2A_1$ to the 4-bit binary number $B_4B_3B_2B_1$ to produce a 4-bit sum $\Sigma_4\Sigma_3\Sigma_2\Sigma_1$ and C_{OUT}. (What should you do with C_{IN}?) The A inputs should be controlled by consecutive side-board data switches; the B inputs will fill out the remaining side-board data switches. Display the 4-bit output and C_{OUT} on consecutive LEDs. Check-out the 7483's operation. Record (in both base 10 and binary) at least three correct 4-bit binary sums $A + B = \Sigma$. For these parallel binary numbers, different wires carry different bits of the number.

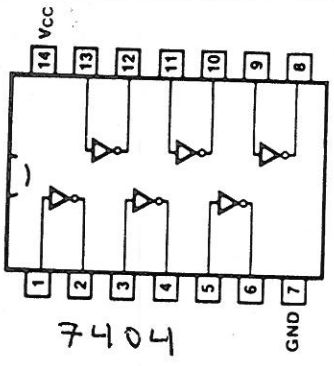
8. Go to room 118 and examine the output of a terminal on a scope. (Connect pin 7 to ground; find the signal on pin 2 for this RS-232 plug.) Hold down a key and sketch the waveform produced. Repeat this for at least 3 different keys. In addition to the usual specifications required for scope sketches, you should label the MSB and LSB of the ASCII integer and the 0/1 logic value of every bit in between. Are there any additional bits transmitted that are not part of the ASCII integer (e.g., extra bits at the start or end of the transmission)? Describe exactly how the signal displayed by the scope can be translated into the ASCII integer for the key pressed (find ASCII codes in Table 14.5 HH p.1040) . Does the MSB of the ASCII integer lead or lag the LSB? Find the voltages that correspond to logical 1 and logical 0. (Surprised?) Figure the baud rate (bits per second) from the scope and compare to the set value (9600 baud).



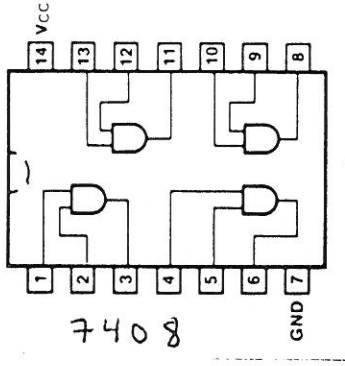
7400



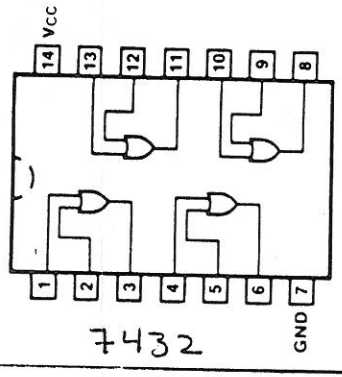
7402



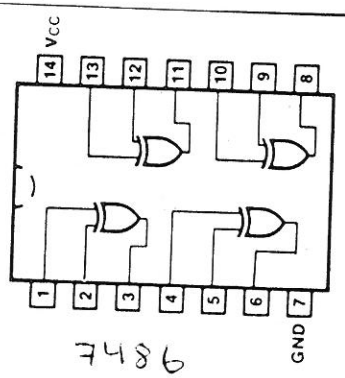
7404



7408



7432



7486

is represented in ASCII code as follows:

1010011 1110100 1100001 1110010 1110100
S t a r t

Table 1.2 ASCII Code

character	ASCII code $A_6A_5A_4A_3A_2A_1A_0$	character	ASCII code $A_6A_5A_4A_3A_2A_1A_0$	character	ASCII code $A_6A_5A_4A_3A_2A_1A_0$
space	0 1 0 0 0 0 0	@	1 0 0 0 0 0 0	`	1 1 0 0 0 0 0
!	0 1 0 0 0 0 1	A	1 0 0 0 0 0 1	a	1 1 0 0 0 0 1
"	0 1 0 0 0 1 0	B	1 0 0 0 0 1 0	b	1 1 0 0 0 1 0
#	0 1 0 0 0 1 1	C	1 0 0 0 0 1 1	c	1 1 0 0 0 1 1
$	0 1 0 0 1 0 0	D	1 0 0 0 1 0 0	d	1 1 0 0 1 0 0
%	0 1 0 0 1 0 1	E	1 0 0 0 1 0 1	e	1 1 0 0 1 0 1
&	0 1 0 0 1 1 0	F	1 0 0 0 1 1 0	f	1 1 0 0 1 1 0
'	0 1 0 0 1 1 1	G	1 0 0 0 1 1 1	g	1 1 0 0 1 1 1
(	0 1 0 1 0 0 0	H	1 0 0 1 0 0 0	h	1 1 0 1 0 0 0
)	0 1 0 1 0 0 1	I	1 0 0 1 0 0 1	i	1 1 0 1 0 0 1
*	0 1 0 1 0 1 0	J	1 0 0 1 0 1 0	j	1 1 0 1 0 1 0
+	0 1 0 1 0 1 1	K	1 0 0 1 0 1 1	k	1 1 0 1 0 1 1
,	0 1 0 1 1 0 0	L	1 0 0 1 1 0 0	l	1 1 0 1 1 0 0
-	0 1 0 1 1 0 1	M	1 0 0 1 1 0 1	m	1 1 0 1 1 0 1
.	0 1 0 1 1 1 0	N	1 0 0 1 1 1 0	n	1 1 0 1 1 1 0
/	0 1 0 1 1 1 1	O	1 0 0 1 1 1 1	o	1 1 0 1 1 1 1
0	0 1 1 0 0 0 0	P	1 0 1 0 0 0 0	p	1 1 1 0 0 0 0
1	0 1 1 0 0 0 1	Q	1 0 1 0 0 0 1	q	1 1 1 0 0 0 1
2	0 1 1 0 0 1 0	R	1 0 1 0 0 1 0	r	1 1 1 0 0 1 0
3	0 1 1 0 0 1 1	S	1 0 1 0 0 1 1	s	1 1 1 0 0 1 1
4	0 1 1 0 1 0 0	T	1 0 1 0 1 0 0	t	1 1 1 0 1 0 0
5	0 1 1 0 1 0 1	U	1 0 1 0 1 0 1	u	1 1 1 0 1 0 1
6	0 1 1 0 1 1 0	V	1 0 1 0 1 1 0	v	1 1 1 0 1 1 0
7	0 1 1 0 1 1 1	W	1 0 1 0 1 1 1	w	1 1 1 0 1 1 1
8	0 1 1 1 0 0 0	X	1 0 1 1 0 0 0	x	1 1 1 1 0 0 0
9	0 1 1 1 0 0 1	Y	1 0 1 1 0 0 1	y	1 1 1 1 0 0 1
:	0 1 1 1 0 1 0	Z	1 0 1 1 0 1 0	z	1 1 1 1 0 1 0
;	0 1 1 1 0 1 1	[	1 0 1 1 0 1 1	{	1 1 1 1 0 1 1
<	0 1 1 1 1 0 0	\	1 0 1 1 1 0 0	|	1 1 1 1 1 0 0
=	0 1 1 1 1 0 1	]	1 0 1 1 1 0 1	}	1 1 1 1 1 0 1
>	0 1 1 1 1 1 0	^	1 0 1 1 1 1 0	~	1 1 1 1 1 1 0
?	0 1 1 1 1 1 1	_	1 0 1 1 1 1 1	delete	1 1 1 1 1 1 1

SN54LS83A/SN74LS83A

4-BIT BINARY FULL ADDER WITH FAST CARRY

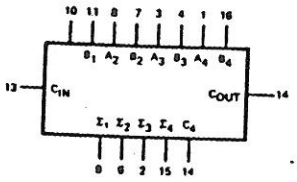
DESCRIPTION — The SN54LS83A/SN74LS83A is a high-speed 4-Bit Binary Full Adder with internal carry lookahead. It accepts two 4-bit binary words ($A_1 - A_4$, $B_1 - B_4$) and a Carry Input (C_{IN}). It generates the binary Sum outputs ($\Sigma_1 - \Sigma_4$) and the Carry Output (C_{OUT}) from the most significant bit. The LS83 operates with either active HIGH or active LOW operands (positive or negative logic). The SN54LS283/SN74LS283 is recommended for new designs since it is identical in function with this device and features standard corner power pins.

PIN NAMES

		LOADING (Note a)	
		HIGH	LOW
$A_1 - A_4$	Operand A Inputs	1.0 U.L.	0.5 U.L.
$B_1 - B_4$	Operand B Inputs	1.0 U.L.	0.5 U.L.
C_{IN}	Carry Input	0.5 U.L.	0.25 U.L.
$\Sigma_1 - \Sigma_4$	Sum Outputs (Note b)	10 U.L.	5(2.5) U.L.
C_{OUT}	Carry Output (Note b)	10 U.L.	5(2.5) U.L.

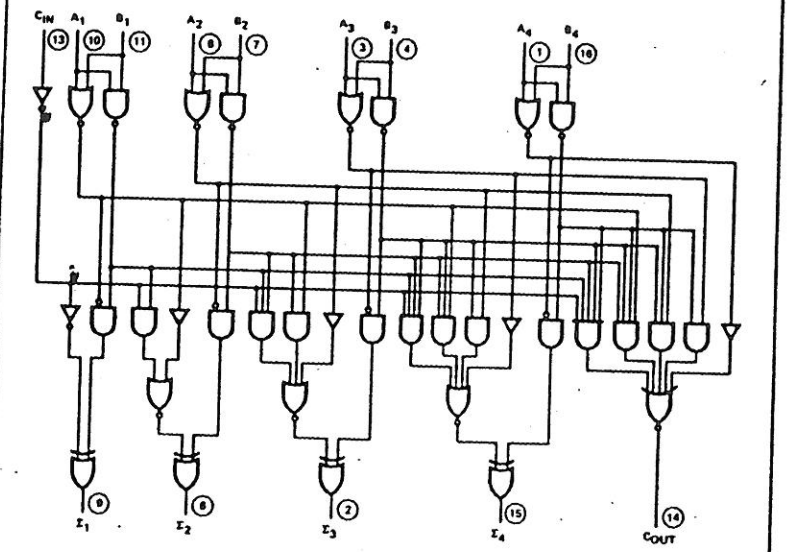
NOTES:
a. 1 TTL Unit Load (U.L.) = 40 μA HIGH/1.6 mA LOW.
b. The Output LOW drive factor is 2.5 U.L. for Military (54) and 5 U.L. for commercial (74) Temperature Ranges.

LOGIC SYMBOL



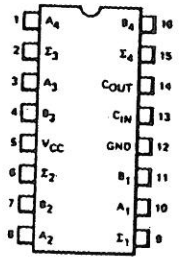
V_{CC} = Pin 5
GND = Pin 12

LOGIC DIAGRAM



V_{CC} = Pin 5
GND = Pin 12
○ = Pin Numbers

CONNECTION DIAGRAM DIP (TOP VIEW)



NOTE:
The Flatpak version has the same pinouts (Connection Diagram) as the Dual In-Line Package.

FUNCTIONAL DESCRIPTION — The LS83A adds two 4-bit binary words (A plus B) plus the incoming carry. The binary sum appears on the sum outputs ($\Sigma_1 - \Sigma_4$) and outgoing carry (C_{OUT}) outputs.

$$C_{IN}+(A_1+B_1)+2(A_2+B_2)+4(A_3+B_3)+8(A_4+B_4) = \Sigma_1+2\Sigma_2+4\Sigma_3+8\Sigma_4+16C_{OUT}$$

Where: (+) = plus

Due to the symmetry of the binary add function the LS83A can be used with either all inputs and outputs active HIGH (positive logic) or with all inputs and outputs active LOW (negative logic). Note that with active HIGH inputs, Carry In can not be left open, but must be held LOW when no carry in is intended.

Example:

	C_{IN}	A_1	A_2	A_3	A_4	B_1	B_2	B_3	B_4	Σ_1	Σ_2	Σ_3	Σ_4	C_{OUT}	
logic levels	L	L	H	L	H	H	L	L	H	H	H	L	L	H	
Active HIGH	0	0	1	0	1	1	0	0	1	1	1	0	0	1	(10+9=19)
Active LOW	1	1	0	1	0	0	1	1	0	0	0	1	1	0	(carry+5+6=12)

Interchanging inputs of equal weight does not affect the operation, thus C_{IN}, A_1, B_1, can be arbitrarily assigned to pins 10