

LAB #6: LOGIC CIRCUIT CHARACTERISTICS AND SIMPLE LOGIC GATES

Updated July 30, 2003.

Objective:

To characterize BJT and CMOS *inverters*. To investigate simple *logic gates*. To compare *measured* and *simulated* logic circuits.

Components:

1 × CD4007UB MOSFET Array, 1 × 2N2222A *npn* BJT, 4 × 1N4148 *pn* diodes, 1 × 0.1 μF capacitor, and resistors: 1 × 1.0 kΩ and 3 × 10 kΩ (all 5%, ¼ W).

Instrumentation:

A bench power supply, a waveform generator (triangle/square wave), a digital voltmeter (DVM), and a dual-trace oscilloscope.

PART I – THEORETICAL BACKGROUND

As we know, one of the most convenient ways to represent information is electronically, via voltage or current signals. A common example is the voltage waveform emerging from a microphone. Since this waveform is an *electrical replica* of the sound wave applied to the microphone, which is of *mechanical nature*, the microphone output is referred to as an *analog signal*. The part of electronics dealing with the amplification and processing of analog signals is aptly called *analog electronics*. In demanding situations such as hi-fi and precision instrumentation, analog electronics often faces extremely challenging tasks. Consider, for instance, an op amp circuit that at some given instant is required to output the voltage $v_O = \pi \text{ V} = 3.1415927 \text{ V}$. Because of component non-idealities, such as the op amp's input offset error and inherent delays, the circuit might actually give $v_O = 3.1523362 \text{ V}$, thereby invalidating the significance of all digits following the “4”, and corrupting even the “4” as it is being rendered as a “5” instead of a “4”.

Digital electronics circumvents the inherent limitations of analog electronics by representing information via *binary signals*, such as a voltage that is restricted to assume only two values, $v_O = V_{OL}$ (L for *Low*) and $v_O = V_{OH}$ (H for *High*). If these values are sufficiently well separated from each other, such as $V_{OL} \cong 0 \text{ V}$ and $V_{OH} \cong 5 \text{ V}$, then *even the crudest* circuits would be capable to generate them as well as to interpret them correctly.

A binary digit, or *bit* for short, represents the tiniest piece of information possible. If we were to use it as basis for course grades, we would have only 2 possibilities, say, H for Pass, and L for Fail. A better bet would be to use 2 bits, for then we would have 4 possible combinations, say, HH for a letter grade of *A*, HL for *B*, LH for *C*, and LL for *D*. If we wanted to encode also the letter grade of *F* as well as intermediate grades such as *A-*, *B+*, *B-*, and so forth, then we would need a total of 12 combinations. This can be achieved with 4 bits, which provide 16 possible combinations, HHHH, HHHL, HHLH... LLLH, LLLL. The four remaining combinations could be used to encode something else, like *Freshman*, *Sophomore*, *Junior*, and *Senior*. Luckily enough, the number of possibilities increases *exponentially* (2^n) with the number n of bits. For instance, with $n = 10$, we can represent $2^{10} = 1024$ different possibilities, a number commonly referred to as “k” for kilo. Likewise, $2^{20} = 1,048,576$ is called “M” for Mega.

Though extremely convenient for the *processing*, *transmission*, and *storage* of information, digital electronics is not the answer to everything. *Nature is inherently analog*, so there will always be a need for analog circuits as well as competent analog engineers to design them. A popular example is offered by *digital audio*. Here, the tiny analog signals available from microphones and various musical

instrument pickups are suitably amplified and conditioned via analog circuitry. They are then converted to digital form via *analog-to-digital* (A/D) converter circuits, whence they are manipulated via suitable *digital signal processing* (DSP) circuitry for mixing as well as the creation of special effects, and finally burned into a CD for storage, transportation, and sale. At the user's end, suitable *digital-to-analog* (D/A) circuitry converts the CD's digital data back to analog form, for further amplification by power amplifiers and finally for the conversion back to sound waves via the loudspeakers. Today's tendency is thus as follows: Perform as many functions as possible in **digital form**, keeping in mind that nature presents itself predominantly in **analog form**, so that the electronic interface between real world and digital systems is analog.

Basic Logic Gates:

Figure 1 shows the three most common circuits for processing binary signals: the *Inverter*, the *NAND* gate, and the *NOR* gate. Shown immediately below are their *truth tables*, representing the output Y for all possible input combinations. Gates are made up of transistors and thus exhibit distinctive electrical characteristics. It is the intent of this lab to investigate them. The issues of greatest concern in the application of logic gates are (a) their ability to correctly produce as well as interpret binary signals in the presence of noise and other forms of disturbance, (b) how long it takes for a gate to evaluate its input combination and yield the correct output, and (c) how much power a gate takes to operate. These issues are best illustrated for the case of the *inverter*, the most basic of logic circuits. Of course, to build a complex digital system we need more sophisticated logic circuits such as a NAND and NOR gates, flip-flops, encoders/decoders, registers, and memories. However, when it comes to the study of electrical properties, the basic inverter, in spite of its simplicity, is fairly representative of most logic circuits, so we find it appropriate to investigate it in proper detail.

Noise Margins:

As a signal produced by a *transmitting gate* travels down a wire, it picks up various forms of noise, so by the time it reaches a *receiving gate* it may be appreciably contaminated. The question arises as to *how much noise* can be tolerated at the receiving end and still allow for the signal to be interpreted correctly there. A logic family's ability to function correctly in noisy environments is measured via its *noise margins*.

As shown in Fig. 2(a), an inverter is powered between V_S (typically 5 V) and ground. Circuit behavior is best visualized via the *voltage transfer curve* (VTC), representing the plot of v_O versus v_I . Figure 2(b) shows the idealized VTC. For $v_I < 0.5V_S$ the circuit gives $v_O = V_S (= V_{OH})$, and for $v_I > 0.5V_S$ it gives $v_O = 0 \text{ V} (= V_{OL})$. The VTC of a practical inverter will generally depart from this idealized shape,

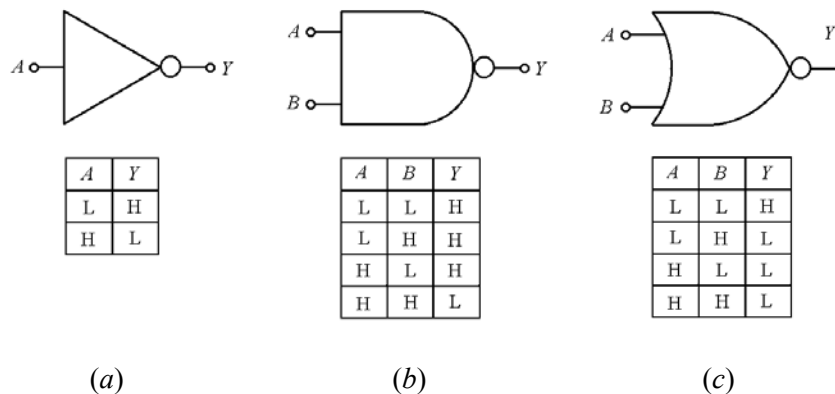


Fig. 1 – Circuit symbols and truth tables for the basic logic gates: (a) the Inverter, (b) the NAND gate, and (c) the NOR gate.

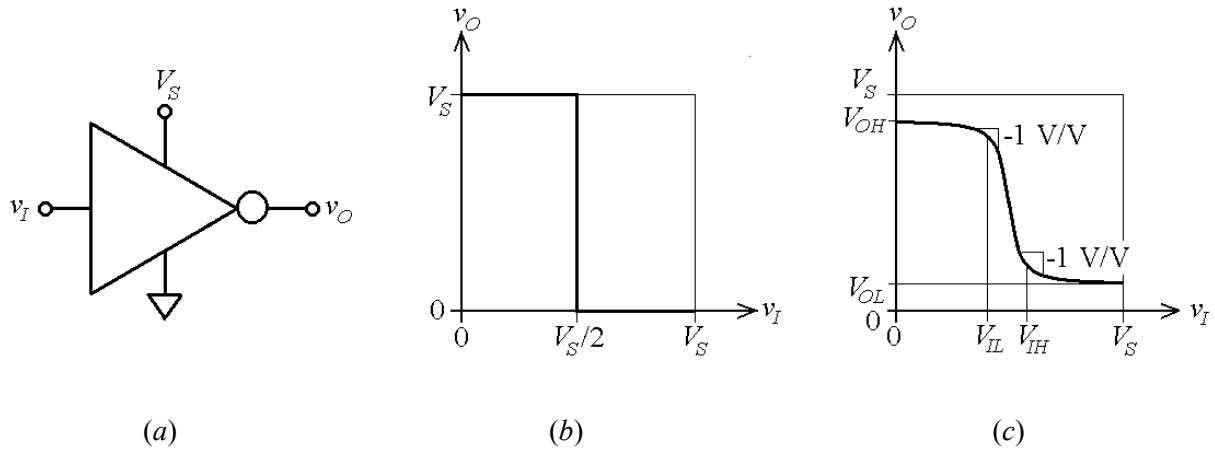


Fig. 2 – (a) Logic inverter. (b) Idealized VTC, and (c) practical VTC.

and will look more like that of Fig. 2(c). Here we observe that as long as v_I is sufficiently low (near 0V), the inverter gives $v_O = V_{OH}$ (note that V_{OH} is not necessarily V_S); as long as v_I is sufficiently high (near V_S), the inverter gives $v_O = V_{OL}$ (note that V_{OL} is not necessarily 0 V).

The departure of a practical VTC from the ideal is specified in terms of the *noise margins*, defined as

$$NM_L = V_{IL} - V_{OL} \tag{1a}$$

$$NM_H = V_{OH} - V_{IH} \tag{1b}$$

where V_{IL} and V_{IH} are the values of v_I corresponding to the points of the VTC where *slope* is -1 V/V. Physically, the noise margins represent the *maximum amount of noise* that can be tolerated on a line going from the output of a transmitting gate to the input of a receiving gate. As illustrated further in Fig. 3, any noise in excess of this margin will be amplified by the receiving gate by *more than unity*, potentially leading to erroneous circuit behavior. Clearly, the higher the noise margins, the better. For the idealized VTC of Fig. 2(b) we have $NM_L = NM_H = 0.5V_S$ ($= 2.5$ V for $V_S = 5$ V).

VTCs are readily visualized with a dual-trace oscilloscope operated in the *x-y* mode, or via

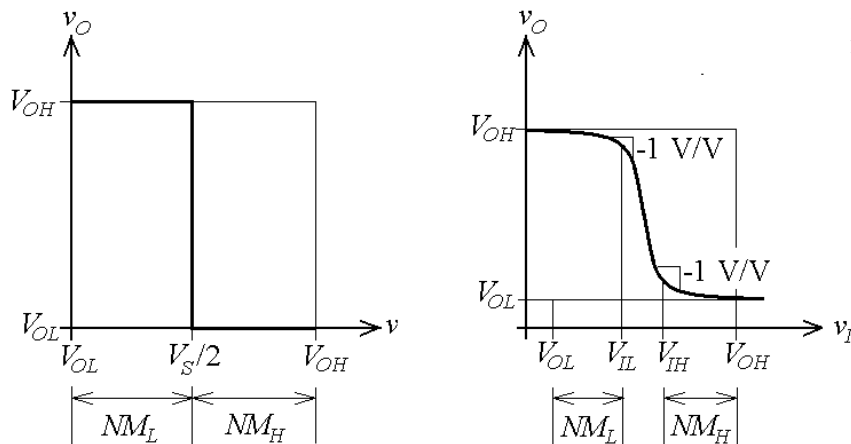


Fig. 3 – Visualizing the noise margins

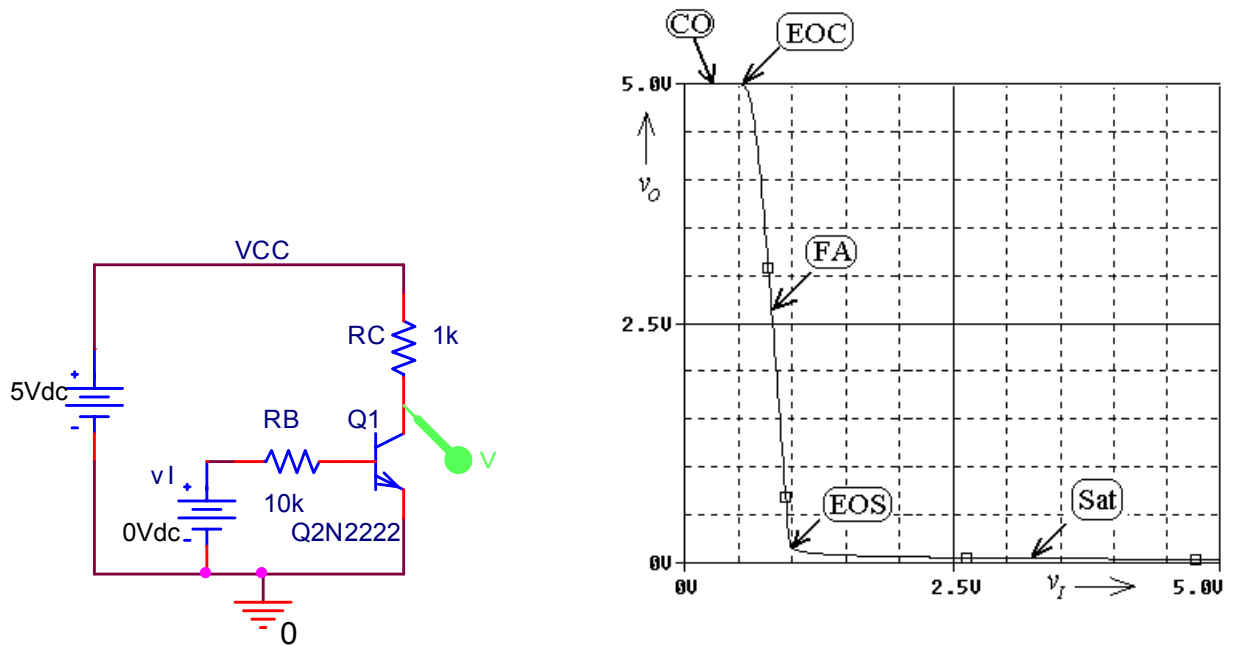


Fig. 4 – BJT inverter and its VTC.

PSpice using a DC Sweep.

Figure 4 shows a PSpice inverter based on the familiar 2N2222 BJT, along with its VTC. This circuit has been simulated using the 2N2222 model available in PSpices’s library. As we know from Lab #4, the input sweep carries the BJT from the *cutoff* (CO) region, to the *edge of conduction* (EOC), through the *forward-active* (FA) region, to the *edge of saturation* (EOS), and thence into deep *saturation* (Sat). One can use simple graphical techniques to identify the points where slope is -1 V/V, and then determine V_{IL} and V_{IH} . Moreover, one can use the cursor to find the values of V_{OL} and V_{OH} , and thus calculate the noise margins via Eq. (1).

Figure 5 shows a PSpice inverter of the CMOS type, along with its VTC. This circuit has been simulated using two homebrew MOSFETs, called respectively 301pMOSFET and 301nMOSFET. Both devices have been created by renaming and suitably editing the *PSpice Models* of two MOSFETs available in the PSpice Library. This has been done first by clicking the device to select it, then by clicking **Edit** → **PSpice Model** to change the values of its parameters. Following are the model statements for the two devices:

```
.model 301pMOSFET          PMOS(W=20u L=5u kp=20u Vto=-1.75 lambda=0.08)
.model 301nMOSFET          NMOS(W=10u L=5u kp=50u Vto=1.5 lambda=0.05)
```

As shown on the VTC itself, the input sweep carries the two MOSFETs, denoted respectively as M_p and M_n , through the regions indicated, where CO indicates the *cutoff* region, Ω the *ohmic* region, and PO the *pinch-off* region. Again, from the VTC one can readily find the noise margins using simple graphical technique.

You can simulate both inverters by downloading their appropriate files from the Web. To this end, go to <http://online.sfsu.edu/~sfranco/CoursesAndLabs/Labs/301Labs.html>, and once there, click on **PSpice Examples**. Then, follow the instructions contained in the **Readme** file.

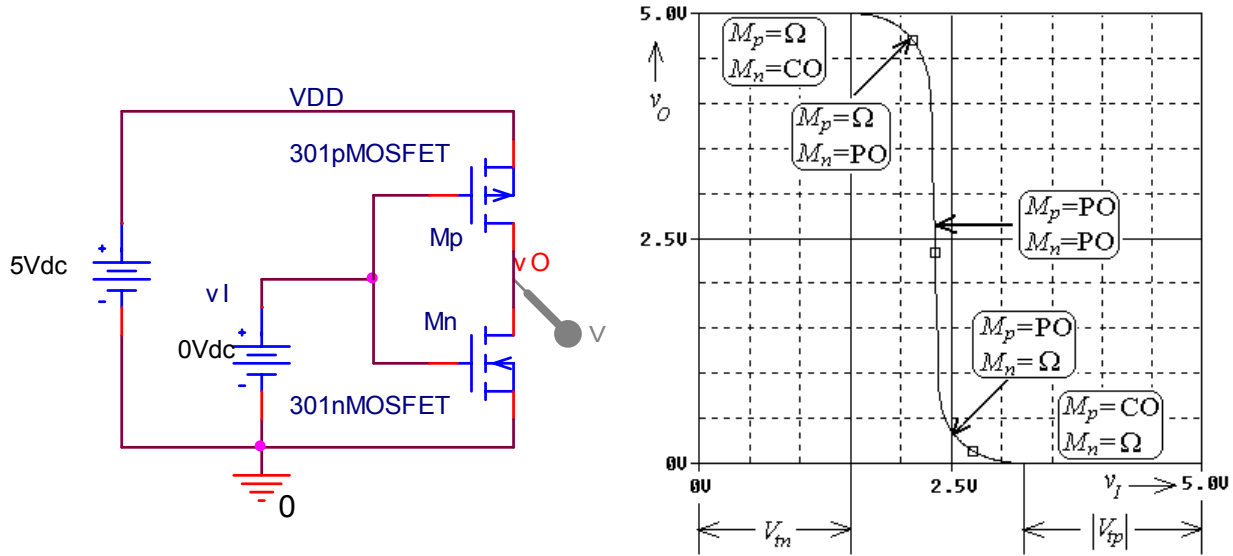


Fig. 5 – CMOS inverter and its VTC.

Propagation Delays:

An inverter's response to a sharp-edged input pulse is not instantaneous, as the circuit takes a certain amount of time to swing its output from one level to the other (see Fig. 6). The speed of response is characterized in terms of the *propagation delays* t_{PLH} and t_{PHL} . The amount of time, following the *leading* edge v_I , that it takes for v_O to swing from V_{OH} down to the transition's *midpoint*, defined as

$$V_{50\%} = \frac{V_{OL} + V_{OH}}{2} \quad (2)$$

is denoted as t_{PHL} . Likewise, the amount of time, following the *trailing* edge of v_I , that it takes for v_O to swing from V_{OL} up to $V_{50\%}$ is denoted as t_{PLH} . The two delays are not necessarily identical, so their average

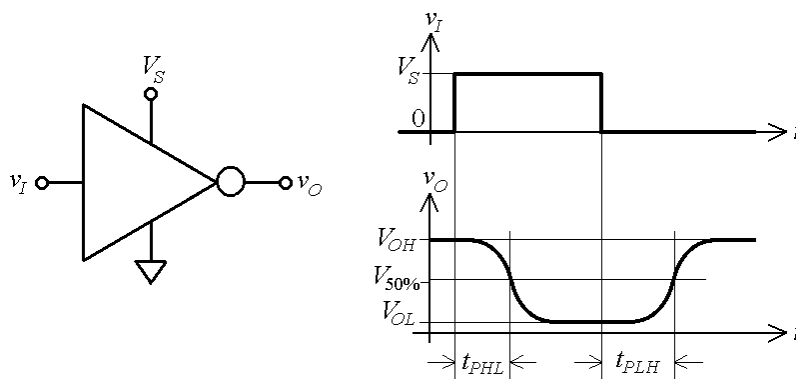


Fig. 6 – Illustrating the propagation delays.

$$t_p = \frac{t_{PLH} + t_{PHL}}{2} \quad (3)$$

is aptly called the *average propagation delay*.

Figure 7 shows the pulse response of the basic BJT inverter. One can readily measure the propagation delays using the cursor facility of PSpice. The results are $t_{PHL} \cong 75$ ns and $t_{PLH} \cong 630$ ns. These delays stem from the fact that it takes time to build up/remove charge in the *junction capacitances* of the BJT as well as the *excess minority charge* inside the base region. These functions are performed by the base current. During t_{PHL} the base current is $I_{BF} \cong (5 - 0.7)/10 = 0.43$ mA, and it flows into the BJT to buildup charge. During t_{PLH} the base current is $I_{BR} \cong (0 - 0.7)/10 = -0.07$ mA, the minus sign indicating that I_{BR} flows *out* of the BJT, thus removing the charge built up by I_{BF} . The asymmetry in the values of t_{PHL} and t_{PLH} stems in part from the different magnitudes of I_{BF} and I_{BR} .

A particularly undesirable component of t_{PLH} is the *storage time* t_S , representing the amount of time it takes for I_{BR} to remove the charge stored the base of the BJT when it was driven past the EOS, into deep saturation. It can be proved that the storage time can be calculated as

$$t_S = \tau_S \ln \frac{I_{BF} - I_{BR}}{I_{B(EOS)} - I_{BR}} \quad (4)$$

where τ_S is a characteristic parameter of the BJT called the *saturation time constant*, in ns, and $I_{B(EOS)}$ is the base current needed to bring the BJT to the edge of saturation, or $I_{B(EOS)} = I_{C(EOS)}/\beta_F$. For our circuit, $I_{C(EOS)} \cong (5 - 0.2)/1 = 4.8$ mA. Using the cursor facility of PSpice, we readily find $t_S \cong 430$ ns.

In the case of CMOS gates there are no charge storage effects. However, there are still parasitic capacitances internal to the MOSFETs that need to be charged/discharged, thus causing nonzero propagation delays.

A convenient way to find average propagation delays experimentally is to connect an *odd* number n of inverters in *ring* fashion, as depicted in Fig. 8 for the case $n = 3$. It is readily seen that the counter's period of oscillation is $T = 2nt_p$ or $T = 6t_p$ in our example. One can thus measure T with the oscilloscope, and then divide by 6 to find the experimental value of t_p .

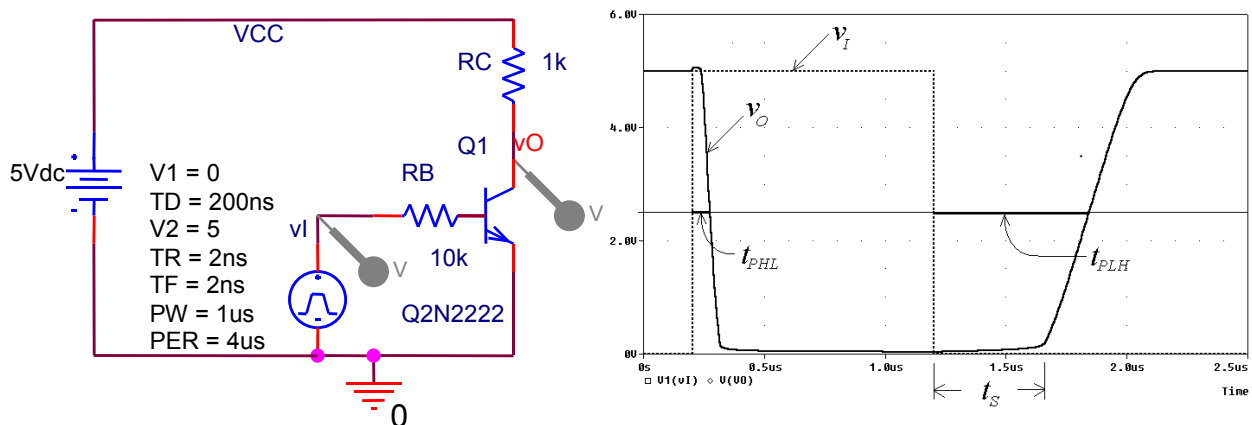


Fig. 7 – Basic BJT inverter and its pulse response.

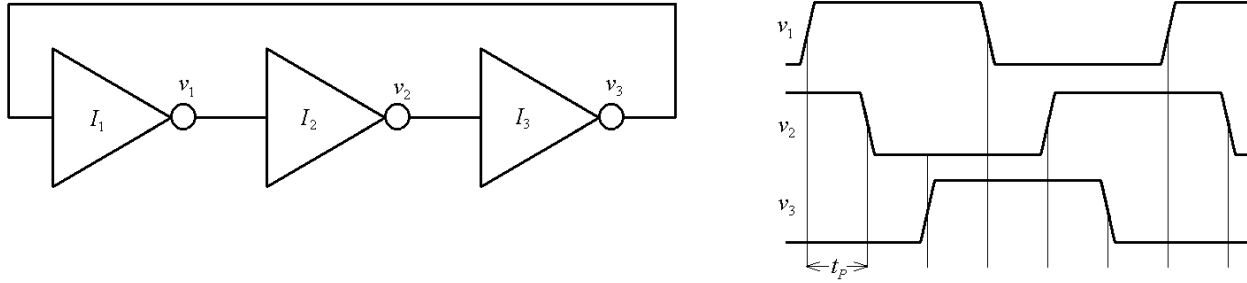


Fig. 8 – Ring oscillator and its waveforms.

PART II – EXPERIMENTAL PART

This experiment is based on the 2N2222 *n*p*n* BJT and the CD4407UB CMOSFET Array. You are urged to download their data sheets from the Web. To this end, go to <http://www.google.com> and search for “CD4007UB” and “2N2222”, or variants thereof. The pin configurations of these devices are repeated in Fig. 9 for your convenience. As stressed many times, be careful (a) to assemble your circuits with power off, (b) to keep leads short, and (b) to bypass the power supply to ground via a 0.1- μ F capacitor.

Henceforth, steps shall be identified by letters as follows: **C** for calculations, **M** for measurements, **P** for prelab work, and **S** for SPICE simulation.

BJT Inverter:

The basic inverter of Fig. 4, though useful to illustrate how a BJT performs the logic function of *inversion*, is inadequate from a practical standpoint as it offers poor NM_L and excessively long storage. It is also susceptible to *loading* when connected to similar gates, indicating a drop in the value of V_{OH} and, hence, a reduction in NM_H . The improved inverter of Fig. 10a is of the so-called DTL (Diode-Transistor Logic) type, a precursor of the more popular TTL (Transistor-Transistor Logic) type. The function of D_1 and D_2 is to shift the VTC toward the right and thus improve NM_L , while that of R_3 to help shut off the BJT during the storage time. Moreover, the base drive network has been redesigned so that with D_A

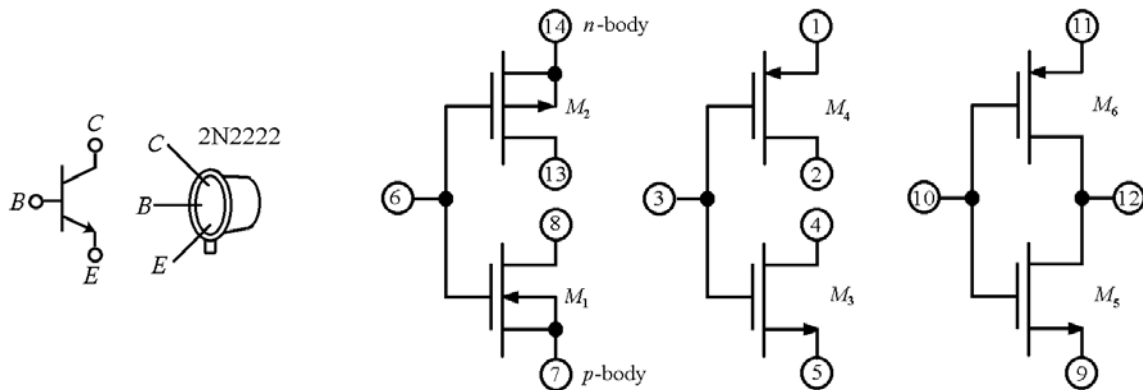


Fig. 9 – Pin configuration for the 2N2222 *n*p*n* BJT and the CD4007UB CMOS Array.

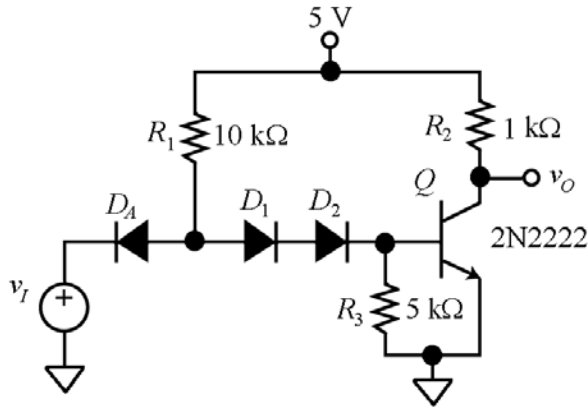


Fig. 10 – DTL-type inverter.

reverse biased it will not load a similar gate upstream.

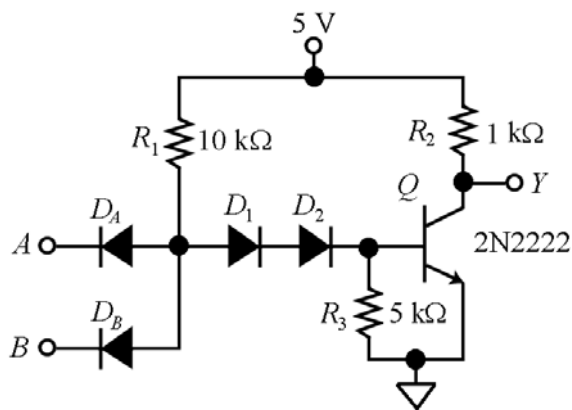
PSC1: Use PSpice to display the VTC of the BJT inverter of Fig. 10 (for the diodes and the BJT, use the models available in PSpice’s library.) Hence, use graphical techniques to determine its *noise margins*.

PSC2: Use PSpice to display the pulse response of the BJT inverter of Fig. 10, in the manner of Fig. 7. Hence, use the cursor facility of PSpice to find its *propagation delays*.

MC3: With power off, assemble the circuit of Fig. 10. Hence, apply power, display its VTC on the oscilloscope, and determine its *noise margins* graphically. Finally, compare with your findings of Step PSC1, and account for possible differences.

Note: To observe the VTC experimentally, first adjust the waveform generator so that v_I is a 100-Hz triangular wave alternating between 0 V and 5 V (make your adjustments using Ch. 1 of the oscilloscope, DC Mode, Trigger from Ch. 1). Next, switch the oscilloscope to the *x-y* Mode, set both channels to 1 V/div, DC, and adjust the offsets so that the origin of the *x-y* display (dot) is at the lower left corner of the screen. Finally, feed v_I to the *x*-axis and v_O to the *y*-axis.

M4: In the circuit of Fig. 10 adjust the waveform generator so that v_I is a 250-kHz pulse train



A	B	D_A	D_B	D_1	D_2	Q	Y
L	L						
L	H						
H	L						
H	H						

Fig. 11 – DTL-type logic gate.

alternating between 0 V and 5 V (make your adjustments using Ch. 1 of the oscilloscope, DC Mode, Trigger from Ch. 1). Hence, while monitoring v_O with Ch. 2 (1 V/div, DC), determine the *propagation delays*. How do they compare with those of Step PSC2? (For optimum visualization, you may need to vary the frequency from the suggested initial value of 200 kHz.)

A BJT Logic Gate:

MC5: With power off, add diode D_B as shown in Fig. 11. Reapply power, and while monitoring Y with the DVM, record in the adjoining table the values of Y (H or L) as well as the states (ON or CO) of the diodes and the BJT for each of the four input combinations shown ($L = 0$ V, $H = 5$ V). What logic function does your gate provide?

CMOS Inverters:

PSC6: Use PSpice to display the VTC of the CMOS inverter of Fig. 12a. Hence, use graphical techniques to determine its *noise margins*.

Note: The PSpice library does not come with models for the DC4007 MOSFETs. You are therefore to create your own models, using the values of k_n , V_{tn} , λ_n , and k_p , V_{tp} , λ_p found experimentally in Lab #5.

MC7: With power off, assemble the circuit of Fig. 12a. Next, apply power, and follow the procedure of Step MC3 to display its VTC on the oscilloscope. Finally, find its *noise margins* graphically, compare with your findings of Step PSC6, and account for possible differences.

MC8: With power off, assemble the circuit of Fig. 12b, consisting of three inverters connected in cascade, so that the composite circuit, known as a *buffered inverter*, still provides the logic function of inversion. Next, apply power, and follow the procedure of Step MC7 to display its VTC on the oscilloscope. Finally, find its *noise margins*, compare with the basic inverter of Fig. 12a, and justify the reason for the buffered inverter's *much improved noise margins*.

MC9: With power off, disconnect the waveform generator from the circuit of Fig. 12b, and connect the

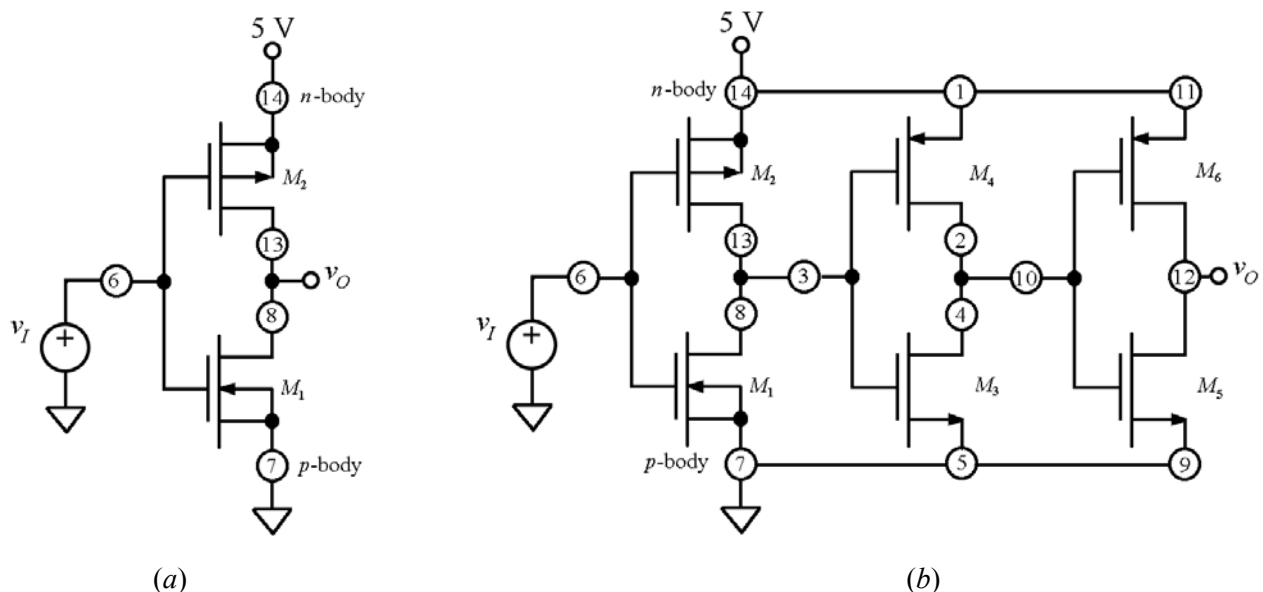


Fig. 12 – CMOS inverters: (a) basic type, and (b) buffered type.

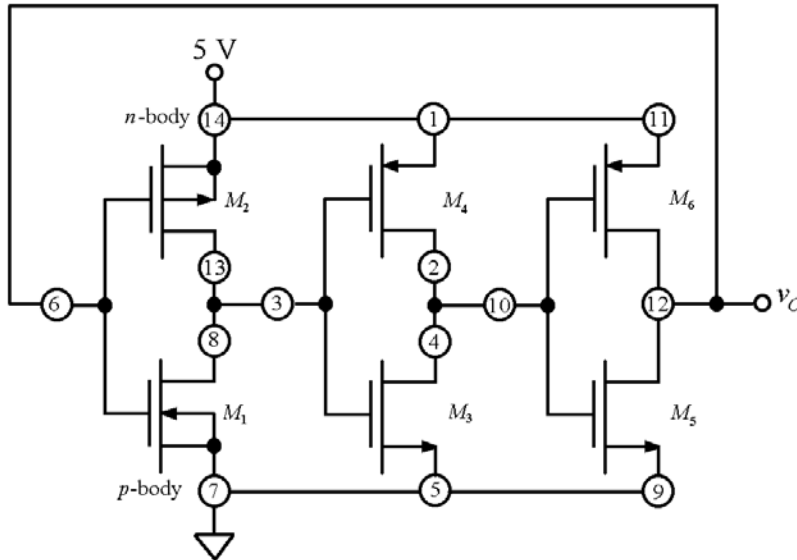
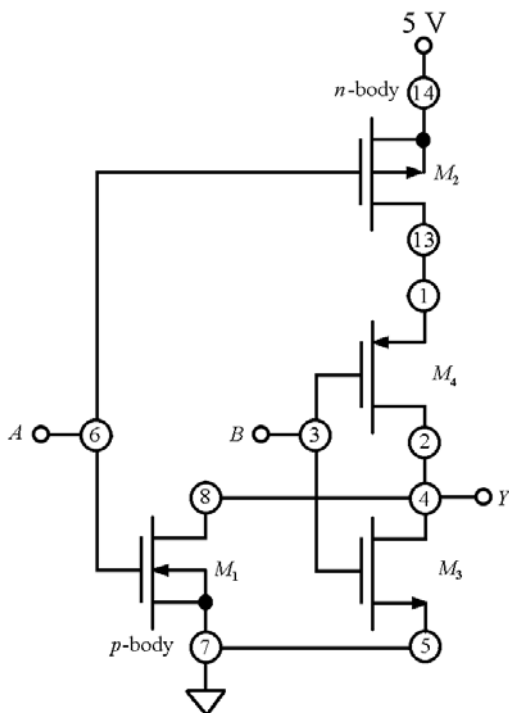


Fig. 13 – CMOS ring counter

output of the rightmost inverter back to the input of the leftmost one, to turn the circuit into a *ring counter* with $n = 3$, as shown in Fig. 13. Apply power, measure the period T of oscillation with the oscilloscope, and obtain the average propagation delay as $t_p = T/6$. How does it compare with that of the BJT inverter of Step M4, given that there are no charge storage effects in MOSFETs?

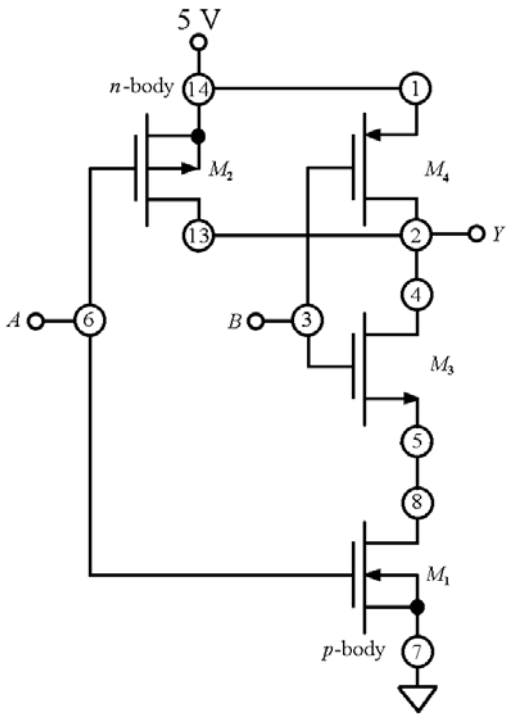
CMOS Logic Gates:

M10: With power off, assemble the circuit of Fig. 14. Apply power, and while monitoring Y with the DVM, record in the adjoining table the values of Y (H or L) as well as the states (Ω or CO) of the



A	B	M_1	M_2	M_3	M_4	Y
L	L					
L	H					
H	L					
H	H					

Fig. 14 – A CMOS logic gate.



A	B	M_1	M_2	M_3	M_4	Y
L	L					
L	H					
H	L					
H	H					

Fig. 15 – Another CMOS logic gate.

MOSFETs for each of the four input combinations shown ($L = 0\text{ V}$, $H = 5\text{ V}$). What logic function does your gate provide?

M11: Repeat Step M10, but for the CMOS gate of Fig. 15.