DIGITAL ELECTRONICS: LOGIC

LAB 15 INTRO: INTRODUCTION TO DISCRETE DIGITAL LOGIC AND MEMORY

GOALS

In this lab, we will learn about the most basic elements of digital electronics, from which more complex circuits, including computers, can be constructed.

DEFINITIONS

Duty cycle – percentage of time during one cycle that a system is active (+5V in the case of digital logic)
Truth-table – table that shows all possible input combinations and the resulting outputs of digital logic components
Flip-flop - a circuit that has two stable states and can be used to store state information.
Logic gates – a physical device that implements some Boolean logic operation

DIGITAL CIRCUITS - GENERAL

In almost all experiments in the physical sciences, the signals that represent physical quantities start out as analog waveforms. To display and analyze the information contained in these signals, they most often are converted to digital data. Often this is done inside a commercial instrument such as an oscilloscope or a lock-in amplifier, which is then connected to a computer through a digital interface. In other cases, data acquisition cards are added to a computer chassis and the analog signals can be inputed directly to the computer. Scientists usually buy their data acquisition equipment rather than build it, so they often don't have to know too much about the digital circuitry that makes it work. Almost all data is eventually analyzed with a computer. We emphasize analog electronics in this course because scientists usually have to know much more about it to design and build their experiments.

Analog information can be translated into digital form by a device called an Analog-to-Digital Converter (A/D converter or ADC). A set of N bits has 2^N possible different values. You might recall this from Lab #5. If you try to represent an analog voltage by 7 bits, your uncertainty will be about 1%, since there are $2^7 = 128$ possible combinations of 7 bits. For higher accuracy you need more bits. The corresponding device that can convert digital data back into an analog waveform is called a Digital-to-Analog Converter (D/A converter or DAC).

Logic gates alone can be used to construct arbitrary combinatorial logic (they can generate any truth-table), but to create a machine that steps through a sequence of instructions like a computer does, we also need memory and a clock. The fundamental single-bit memory element of digital electronics is called a flip-flop. We will study two types, called SR (or RS) and JK. The flip-flops we have chosen are also from the TTL family. A digital clock is a repeating digital waveform used to step a digital circuit through a sequence of states. We will introduce the 555 timer chip and use it to generate a clock signal. Digital circuits able to step through a sequence of states with the aid of flip-flops and a clock are called sequential logic.

DIGITAL LOGIC STATES

The voltage in a digital circuit is allowed to be in only one of two states: HIGH or LOW. HIGH is taken to mean logical (1) or logical TRUE. LOW is taken to mean logical (0) or logical FALSE. In the TTL logic family (see Figure 1), the "ideal" HIGH and LOW voltage levels are 5 V and 0 V but any input voltage in the range 2 to 5.0 V is interpreted as HIGH, and any input voltage in the range 0 to 0.8 V as LOW. Voltages outside this range are undefined, and therefore "illegal," except if they occur briefly during transitions. If the input to a TTL circuit is a voltage in this undefined range, the response is unpredictable, with the circuit sometimes interpreting it as a "1" and sometimes as a "0." See Fig 1.

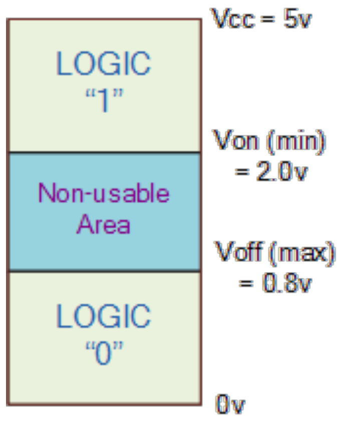


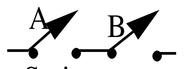
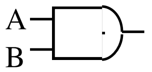
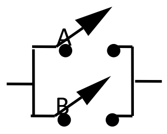
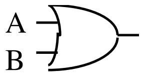
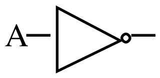
Figure 1: TTL Input Voltage Levels

DIGITAL LOGIC GATES

The flow of digital signals is controlled by transistors in various configurations depending on the logic family (see H&H 8.09 for details). For most purposes, we can imagine that the logic gates are composed of ideal switches with just two states: OPEN and CLOSED. The state of a switch is controlled by a digital signal. The switch remains closed so long as a logical (1) signal is applied. A logical (0) control signal keeps it open.

Logic signals interact by means of gates. The three fundamental gates AND, OR, and NOT, are named after the three fundamental operations of logic that they carry out. The AND and OR gates each have two inputs and one output. The output state is determined by the states of the two inputs. The NOT gate has one input and one output.

The function of each gate is defined by a truth table, which specifies the output state for each possible combination of input states. The output values of the truth tables can be understood in terms of two switches. If the switches are in series, you get the AND function. Parallel switches perform the OR operation. The most common gates are shown in Fig. 2. A small circle after a gate or at an input on the schematic symbol indicates negation (NOT).

Operation	Switches	Condition that circuit is closed	Boolean Notation	Symbol	Truth Table
AND	 Series	(A AND B are closed)	$A \bullet B$ or *AB*	 A·B	A B | A·B 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR	 Parallel	(A OR B is closed)	$A + B$	 A+B	A B | A+B 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
NOT (same as invert)	Different kind of switch	1 means open 0 means closed	NOT $A \equiv \overline{A}$	 $\overline{A}$	A | $\overline{A}$ 0 | 1 1 | 0

Compound Gates

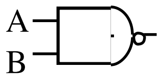
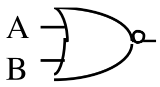
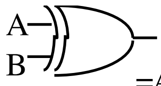
NAND		$\overline{A\cdot B}$
NOR		$\overline{A+B}$
XOR		$A \oplus B$ $=A\overline{B}+\overline{A}B$

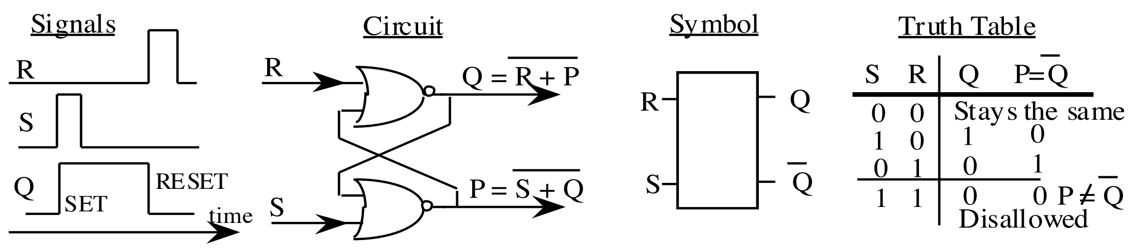
Figure 2: Digital Logic gates

MEMORY ELEMENTS AND FLIP-FLOPS

In sequential logic circuits, the output depends upon previous values of the input signals as well as their present-time values. Such circuits necessarily include memory elements that store the logic values of the earlier signals. The fundamental circuit is the RS memory element. The JK flip-flop has an RS flip-flop at its core, but it adds circuitry that synchronizes output transitions to a clock signal. Timing control by a clock is essential to most complex sequential circuits

RS Memory Circuit

The truth table shows how the circuit remembers. Suppose that it is originally in a state with Q=0 and R=S=0. A positive pulse S at the input sets it into the state Q=1, where it remains after S returns to zero. A later pulse R on the other input resets the circuit to Q=0, where it remains until the next S pulse.



S	R	Q	$P=\overline{Q}$
0	0	Stays the same	
1	0	1	0
0	1	0	1
1	1	0 Disallowed	0 $P \neq \overline{Q}$

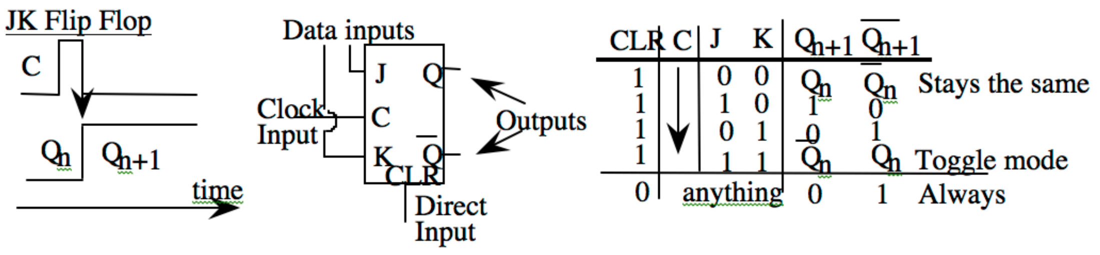
Figure 3: RS memory element.

JK Flip-Flop (TTL74107)

There are three kinds of input to the JK flip-flop

1) data inputs J and K
2) the clock C
3) the direct input CLR (clear)

There are two outputs: *Q* and its compliment.



CLR	C	J	K	Q_{n+1}	$\overline{Q}_{n+1}$	
1	↓	0	0	Q_n	$\overline{Q}_n$	Stays the same
1	↓	1	0	1	0	
1	↓	0	1	0	1	
1	↓	1	1	$\overline{Q}_n$	Q_n	Toggle mode
0	anything			0	1	Always

Figure 4: JK Flip-Flop

In the absence of a clock pulse, the output remains unchanged at the previously acquired value, Q_n, which is independent of the present-time data inputs J and K. Only on the arrival of a clock pulse, C, can the output change to a new value, Q_{n+1}. The value of Q_n depends on the J and K inputs in the way specified in the truth table. The change occurs at the falling (trailing) edge of the clock pulse, indicated by a downward arrow in the truth table in Fig. 4.
The direct input, CLR, overrides the clock and data inputs. During normal operation, CLR = 1. At the moment CLR goes to zero, the output goes to zero and remains there as long as CLR = 0.

USEFUL READINGS

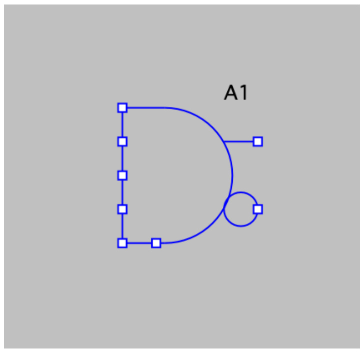
1. FC Chapter 11 (digital electronics)
2. H&H Chapter 8. Everything in this chapter is good to know about but sections 8.01, 8.02, 8.04, 8.07-8.10, 8.12, 8.16 are most relevant. Also have a look at section 5.14 on the 555 timer chip.

LAB PREP ACTIVITIES

You should review the intro material of this Lab manual carefully.

At this point you should be proficient with using LTSpice for simulating analog circuits. In this lab we will use LTSpice to simulate digital circuits. It should be noted that LTSpice is not really a digital simulator, there are better tools out there for this purpose. However, LTSpice provides the most fundamental building blocks of digital electronics and can be used to simulate simple digital circuits.

LTSpice provides models for the most fundamental logic gates including AND (logic and), OR (logic or), XOR (logic exclusive or), and INV (inverter). In LTSpice all logic gates have 5 inputs to the left and an inverting and non-inverting output to the right:

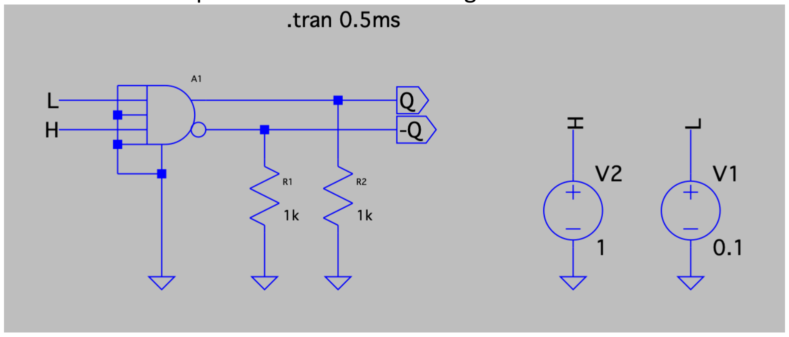


Unused inputs and outputs need to be connected with the pin at the bottom of the gate, which in turn must be connected to common ground (this indicates to LTSpice to remove those pins from the simulation). Used output pins need a path to ground. Therefore, they either have to be connected to an input pin of another element or to a resistor to ground. There is no power supply for logic gates in LTSpice.

The default logic level of LTSpice logic gates is 1V, where voltages < 0.5V are considered as logic low, and > 0.5 V als logic high. Common ground itself has no logic value!

LOGIC GATES

Step 1: Truth Tables. Open a new asc file and add create a circuit with a 2 input AND gate. Also create a voltage source for logic High (pick 1 V), one for logic Low (pick 0.1 V), and label them with H (for High) and L (for Low), respectively. Don't forget that the output pins need a load resistance (1 kΩ is a good value), and that unused pins need to be connected to the pin at the bottom of the gate.



(a) Verify the truth table for the 2 input AND gate by labeling the input pins with all possible combinations of H and L.
(b) Verify the truth table for a 2 input NAND gate by using the inverting output of the AND gate.
(c) Replace in your circuit the AND with an OR gate and verify the truth tables for a 2 input OR and NOR.
(d) Use an XOR gate to verify its truth table.
(e) Now build and test the XOR circuit of your own design using only NANDs and NORs.

MEMORY CIRCIUTS

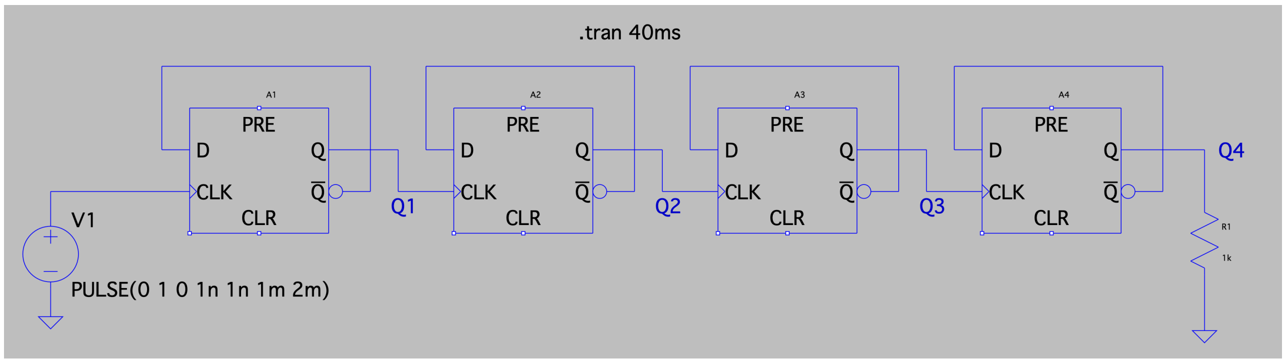
Step 2: RS memory circuit. Create an RS memory circuit from two NOR gates and save the circuit in a new asc file.

(a) Demonstrate the memory property by going through a complete memory cycle. You can do this by connecting the R and S inputs with voltage sources generating short H (i.e. 1V) pulses. You need to offset the R pulse with respect to the S pulse, otherwise you would generate an illegal input R = S = H. Run the simulation and plot the R and S pulses relative to the Q output signal. Does the behavior of your RS circuit agree with predictions?
(b) Now examine the effect of the “illegal” input R = S = H. Describe the outcomes of the illegal operation.

D FLIP-FLOP

Step 3: D Flip-Flop. The RS memory circuit is the simplest possible flip flop circuit. There are many more kinds of them. The by far most important flip flop that can be clocked is the Delay or D-flip flop. It has a single input, labeled “D” (where D stands for Data), and a clock input. The D flip flop stores the logic level applied to the D input, which appears at the Q output after one clock cycle. D-flip flops are commonly used as frequency dividers and counters.

(a) Create a circuit with a D-flip flop by selecting a DFLOP component from the LTSpice component library. Connect the CLK input to voltage pulse source and connect the negated output to the data input. Record the resulting waveforms at D and Q. How does the output frequency compare to the input frequency?
(b) Now daisy chain the flip flop with another 3 of them and record the waveforms at each Q output. Write for 16 clock cycles the logic levels at Q1…Q4 into a table. Now interpret (Q4, Q3, Q2, Q1) as 4 bit dual numbers. How does this number change with a clock cycle?



APPENDIX: BOOLEAN ALGEBRA

Fundamental laws

We imagine a logical variable, *A*, that takes on the values 0 or 1. If $A = 0$ then $\bar{A} = 1$ and if $A = 1$ then $\bar{A} = 0$. Here are some obvious identities using the AND, OR and NOT operations. Looking at these identities you can see why the ‘plus’ symbol was chosen for OR and ‘times’ (•) for AND.

OR	AND	NOT
$A+0=A$	$A\bullet 0=0$	$A+\overline{A}=1$
$A+1=1$	$A\bullet 1=A$	$A\bullet\overline{A}=0$
$A+A=A$	$A\bullet A=A$	$\overline{\overline{A}}=A$
$A+\overline{A}=1$	$A\bullet\overline{A}=0$	

Equality

Two Boolean expressions are equal if and only if their truth tables are identical.

Associative Laws

$$(A+B)+C=A+(B+C)$$
$$(AB)C=A(BC)$$

Distributive Laws

$$A(B+C)=AB+AC$$

Related identities:

$$(A+AB)=A$$
$$(A+\overline{A}B)=A+B$$
$$(A+B)\bullet(A+C)=(A+BC)$$

DeMorgan's Theorems

$$\overline{A\bullet B\bullet\ldots}=\overline{A}+\overline{B}+\ldots$$
$$\overline{A+B+\ldots}=\overline{A}\bullet\overline{B}\bullet\ldots$$

Example of Proof

Each of the above equalities is a theorem that can be proved. Let's do an example by directly comparing the truth tables for the left and right sides. We take on DeMorgan's first theorem for two variables, $\overline{AB}=\overline{A}+\overline{B}$

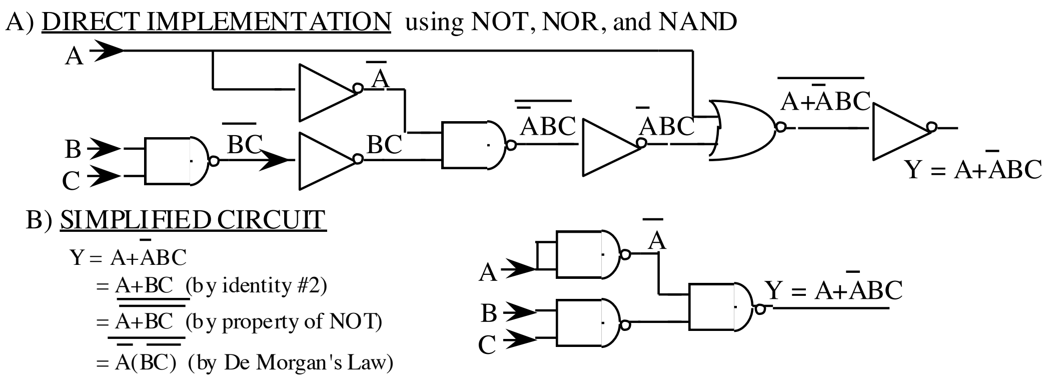
A	*B*	*AB*	$\overline{AB}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

A	*B*	$\overline{A}$	$\overline{B}$	$\overline{A}+\overline{B}$
0	0	1	1	1
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

The last columns of the truth tables are identical. Thus, the first theorem is proven for two variables.

Example of Simplification

Boolean algebra can be used to simplify logical expressions and reduce the number of gates required in a circuit. In Fig. 9.3 we show two ways to implement the expression, $Y = A + \bar{A}BC$.



$Y = A+\bar{A}BC$

$= A+BC$ (by identity #2)

$= \overline{\overline{A+BC}}$ (by property of NOT)

$= \overline{\bar{A}(\overline{BC})}$ (by De Morgan's Law)

Fig. 9.3. Boolean simplification