

Introduction

Analog circuitry works with all values of voltages and currents. Digital logic attempts to work with only 1's and 0's. This is done by assigning 0 V as 0, and 5 V as a 1. These are called Logic Levels, and often called Logic Level High (H) and Logic Level Low (L). These digital signals between digital devices are generally called Transistor-Transistor Logic (TTL). Digital logic is used to store information, compute, and often for control of devices. In some of the design projects, we can expect to primarily use TTL to signal and to control electromechanical devices, such as stepper motors.

Digital Logic

We made a generalization about TTL being 0 V and 5 V. Newer logic families use 0 V for 0, and 3.3 V for 1. This is done to save energy – it takes less current, and therefore less power – to run these devices. In the future, these voltage levels may get even smaller. For the purpose of this tutorial, we will assume Logic Level 1 is 5 V.

So what happens if a digital device sees a voltage of 2.5 V? We cannot predict the outcome and we call this is as X level. If you leave a connection completely unconnected, we get logic level Z, which is called High Impedance. The Z levels are not as bad as X, but why build a circuit with unconnected inputs or outputs? If you really want to assign Z a value, 1 is generally encountered – but by in no way guaranteed.

Analog circuitry is generally faster as compared to the digital logic. The time a digital circuit takes to provide an output is called as the delay time. If you are a large digital circuit, one part of the circuit may finish its operation before another. This means that you need to wait for the correct time in order to get your circuit's output. Thus digital logic timing becomes an important issue when dealing with large digital logic circuits. We will use a certain circuit component called the Flip-Flop to make sure we wait longer than we need to for an answer and then save it. A Flip-Flop is essentially a single bit of memory.

The heart of digital logic is the gate. There are many types of digital logic gates, but the AND, OR and XOR make up the foundation of all digital logic. Each gate compares two or more bits (1's or 0's), and makes a decision as to what the output should be based on the input bits. The symbols for AND, OR and XOR gates along with their truth tables are shown in figure 1. For the most part, these symbols are uniform across the world, but you may run in to some older symbols.

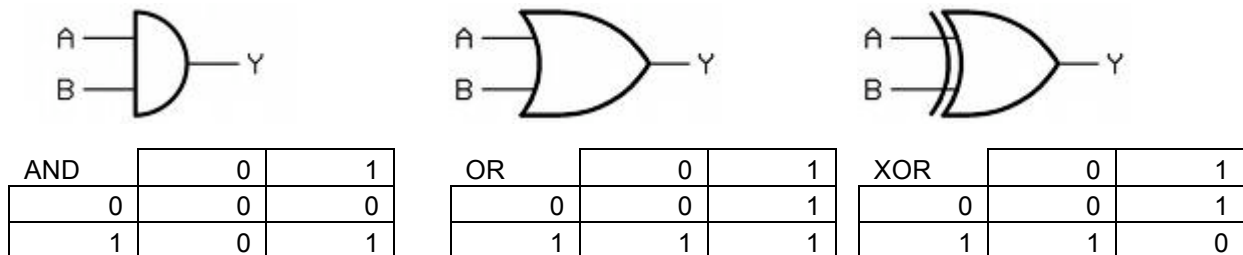


Figure 1 The symbols for AND, OR and XOR gate and their truth tables

Introduction to Digital Logic
Steven Skroch

The AND gates are true or 1 when both inputs are true (A and B must be true or 1). The OR is true if either A or B or both are true. XOR is true if A or B are true, but not if both are true.

The flip-flop also has its own symbol as shown in figure 2, but no truth table. Instead it acts the following way: If (C rising edge) then $Q=D$, else then $Q=\bar{Q}$.



Figure 2 The symbol for a flip-flop

Full Adder with Carry

We want to use digital logic to add two values. We will call these values A and B. The output will be called S. Adding in binary can be complex, and there are a lot of short cuts we could use. However, we are just beginning, so we will go with the most direct route. When adding two single bit binary numbers, $A=1$ and $B=1$, the output will be of two bits, as $01+01=10$ (binary). Our input will be of two bits, A and B, and output two bits, S and C (Carry bit). In the example above, the carry bit is 1 and the output is 0.

Table 1 Adding binary numbers

Inputs		Outputs	
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

In order to figure out what logic we need, we can build a logic table. If $A=0$ and $B=1$, or if $A=1$ and $B=0$, we want the output to be 01, as $01+00=01$. If Both A and B are 0, $S=0$. If $A=1$ and $B=1$, $C=1$ and $S=0$. Let's split this up bit by bit. If $A=0$ and $B=0$, $S=0$. If $A=1$ or $B=1$, $S=1$. If $A=1$ and $B=1$, $S=0$.

Table 2 The table for the Sum bit

S	0	1
0	0	1
1	1	0

Look familiar? It's the XOR gate.

Now lets come up with the table for the Carry bit. If $A=0$ and $B=0$, $S=0$. If $A=1$ or $B=1$, $S=0$. If $A=1$ and $B=1$, $S=1$.

Table 3 The table for the Carry bit

C	0	1
0	0	0
1	0	1

Oh, look !! It's the AND gate. We can now say the following:

$$S = A \text{ XOR } B$$

$$C = A \text{ AND } B$$

We can draw that as shown in figure 3:

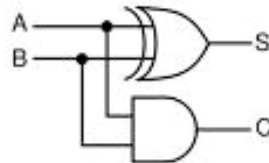


Figure 3 The binary adder with the Sum and the Carry bit

Now what if we wanted to add bigger numbers? The adder we have here, called the half adder, can only do two single bit numbers. It is, however, the building block of a full adder, capable of adding many bits together. We need to modify this to accept a carry in bit, from the adder before it. A full adder looks as shown in figure 4.

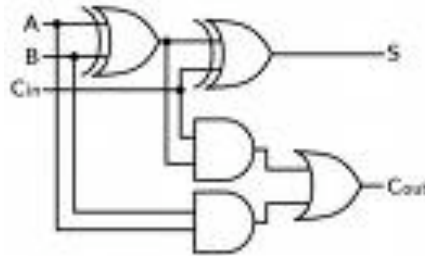


Figure 4 The full adder

The full adder is really 2 half adders with the addition of an OR gate for the Carry bit. Now recall what we said about propagation delay? S will be determined before C will. If every gate takes 1 ms to compute its output, S will arrive at 2 ms, and C will arrive at 3 ms. In order to wait that long, and then capture our output, we will need Flip-flops.

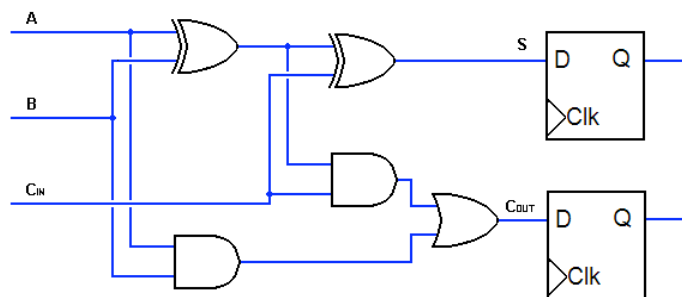


Figure 5 The full adder with flip-flops

Introduction to Digital Logic

Steven Skroch

Clock

Now we need a clock to wait the necessary time to catch our signal. A clock just flips from 0 to 1 and back to 0 again in a specified time, over and over again. Each time the clock rises from 0 to 1, the flip-flops catch the inputs, and keep them there no matter how the rest of the circuit changes.

If we chain a lot of these adders together, we will need to wait 3 ms after A, B, and C_{in} have changed before we can look for S and C_{out} . Let's make this 5 ms, just to be sure. To make a clock, we need a 555 timer. This old, reliable circuit has been used for decades for timing purposes, and is fairly easy to set up. Our final circuit will look as shown in figure 6.

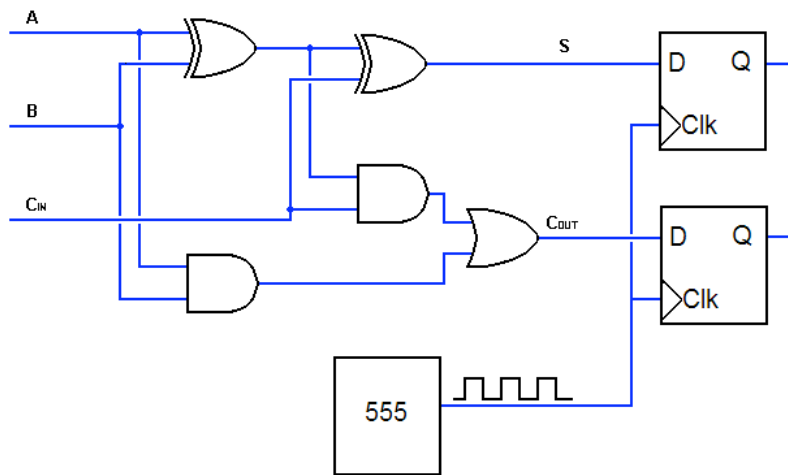


Figure 6 The full adder with flip-flops and timing circuit

Circuit Setup – Reading Schematics

The hardest part of building the circuit is reading the data sheet. There are lots of numbers, all of which you need to care about if you were making a device that were to be used in a hospital. But since our full adder isn't leaving this room, we just need to care about a few values.

Open the data sheet for the AND gate. This can be found at:

<http://focus.ti.com/lit/ds/symlink/sn74hc08.pdf>

The first thing you should see is the operating range which is 2 to 6 V. This refers to the power required to run the gate. Refer to figure 7 for pinout of AND gate.

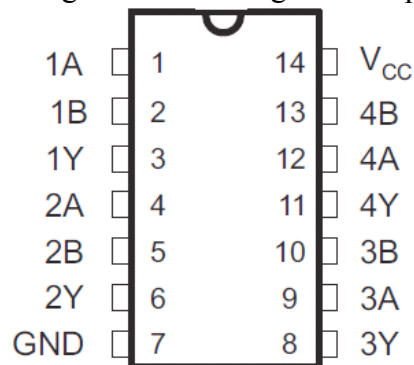


Figure 6 Pinout for the AND gate

Introduction to Digital Logic

Steven Skroch

The Vcc is always where you provide the power and GND is always grounded. A and B's are inputs, Y's are outputs. This particular chip has four AND gates on it. We will be using two AND gates. Refer to page 3 in the datasheet and look for "V_{IH}" and "V_{IL}". V_{IH} is the minimum voltage the gate has to see to be considered an input of 1. V_{IL} is the maximum voltage that the gate must see in the input side to be considered a 0.

Now scroll down to page 4 on the datasheet. Look for "V_{OH}" and "V_{OL}". V_{OH} is the minimum voltage the gate will output when giving out a value of 1. V_{OL} is the maximum voltage that the gate will output when Y=0. When connecting the gates to each other, we have to make sure that the output voltages of the first gate are greater than the input voltages of the second gate.

Also refer to page 4 on the datasheet for the timing characteristics. T_{pd} is the propagation time of the gate – how long it takes to get from a change in the inputs to a change in the outputs. 125 ns is pretty small compared to our target 5 ms clock cycle, so we don't really have to worry about it. On page 5 of the datasheet there are a bunch of timing parameters, including how long it takes to get from 0 to 1 and back again (in this case it is 6 ns). This chip is fast, at least compared to our clock.

The gates are easy to set up. No resistors needed. The flip-flop is also quite easy to set up. Now we will set up the clock using the 555 timer. Open the following data sheet:

http://semicon.njr.co.jp/njr/hp/fileDownloadMedia.do?_mediaId=407

Scroll down to page 5 on the datasheet and refer to figure 3 on the right side of the datasheet. We will build the clock in the Free Running Operation as shown in the figure 3. The pin 8 will receive the 5 V of power and pin 3 will be connected to the Clock (Clk) of our Flip-Flops. For this circuit, we will use capacitors we have on hand, and will have to use potentiometers to make up for it. The following equations specify the behavior of our 555 timer:

$$\text{High Time } T_H = 0.693 \cdot (R_A + R_B) \cdot C$$

$$\text{Low Time } T_L = 0.693 \cdot R_B \cdot C$$

These units are in seconds, ohms and farads. So you will have to convert from, say, 10 uF to 0.00001 F. Please be careful with the conversion.

Low time has only 2 variables, so we will start there. If we have a 10 uF capacitor to work with, we will have to solve the following equation: $0.005 \text{ s} = 0.693 \cdot R_B \cdot 0.00001 \text{ F}$. So, $R_B = 721.5 \text{ ohms}$.

Now we want $T_H = T_L$, which gives us an $R_A = 0$. However, the 555 won't work without R_A , so we need something there. An important lesson with the 555 is that it really has problems doing 50% duty cycles (that is, it is high half the time, or 50% of the period or duty cycle). However, we are in luck. We only care about the rising edge of the wave. So we can dispense with the high time and low time, and go on to the frequency calculation on page 5 of the 555's data sheet.

$$F = 1.44 / (R_A + 2R_B) \cdot C$$

We want $F = 1/5 \text{ ms} = 200 \text{ Hz}$. For simplicity, we will make $R_A = R_B$, and $C = 10 \text{ uF}$. So: $200 \text{ Hz} = 1.44 / (3 \cdot R) \cdot 0.00001 \text{ F}$.

$R = 240 \text{ ohms}$, but double check that. If we don't have those resistors, just take a potentiometer, and measure it using the multimeter. Keep turning the potentiometer until you get the resistance you want.

Introduction to Digital Logic

Steven Skroch

From here on we just connect everything up the right way.

Note: This is the hardest part so please ask the TA for any help. Just remember to keep your wires flat, and power everything off the same 5 V and Ground source. You will learn the most from messing up, so feel free to.