EE1000 Project 3
Electric Die

Overview

In this project, we make design and build an electronic die (singular of dice). The input is a button that causes the die to cycle through 1, 2…6 and repeat until the button is released. The outputs are 7 LEDs configured as shown in Figure 1, below:

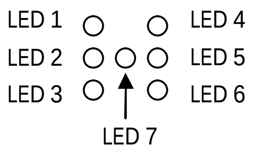

Figure 1. LED Configuration

Since the time the button is released is independent of the cycle time, the number on the die when the cycle stops will be random.

Design

Whenever we are faced with a design problem such as this, we need to follow a few simple design steps:

1. Understand the concept – First we need to find out what is required of the design and what it is supposed to do. If you understand the overview, above, you should be well on your way to completing this step.
2. Understand or create an algorithm to do the job – An algorithm is a sequence of steps or processes that when performed, will accomplish the task we desire. We'll get to that shortly.
3. if the design is too complicated, subdivide and repeat. If not, design the device using your education, skill and experience. We'll walk you through that process too, but as the semester progresses, you will take more and more responsibility for accomplishing this step.
4. Test. We design from the top down, but we test from the bottom up. Each subdivision of the design (step 3) is tested before moving on to the higher level of the design,

Approach (Algorithm)

One way to make (implement) this project is:

1. Have an oscillator that is controlled by the button (an oscillator is a device that produces an AC signal). When the button is pressed, the oscillator must produce a square wave that alternates between 0 volts and 5 Volts. When the button is released, the signal must stop at either 0 or 5 Volts, it doesn't matter which,
2. The output of the oscillator must be fed into a binary counter. A counter is a device that outputs a (binary) number, and that number is incremented each time the "clock" signal rises from 0 to 5 volts. This counter needs to be designed so that when the output reaches 6, the next output will be 1 instead of 7.
3. The output of the counter is in binary, and we need to display dice pips. We will need to decode the binary value into signals for each of the LEDs that represent the count on the die.

This algorithm can be illustrated by a block diagram (See Figure 2).

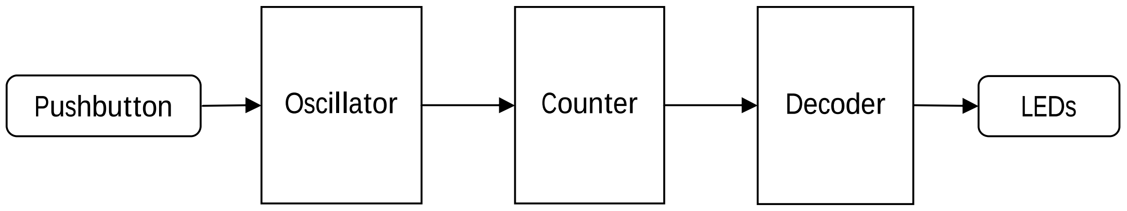

Figure 2 – Block Diagram

Note that the arrows between the blocks represent information, not wires. The connection between the oscillator and the counter is only one wire, but the connection between the counter and decoder would need 3 wires. Of course, there need to be 7 wires going to the LEDs.

We will want to subdivide this design into 3 parts: (a) the oscillator, (b) the counter and (c) the decoder.

Oscillator

Oscillators can be tricky to make, but fortunately we can purchase a device that makes it easy. The part number of this device is NE555 but most people simply call it a 555 timer. (We may actually be using an NE556, which contains two timers in one package.)

In order to operate, the NE555 needs a DC voltage supply. For historical reasons, we call that voltage Vcc. For this project, Vcc will be 5 Volts.

At its heart, the NE555 has two inputs: *threshold* and *trigger* that control two outputs: *discharge* and *output*. When the voltage on *threshold* exceeds 2/3 of Vcc, the NE555 connects *discharge* to ground until the voltage on *trigger* is below 1/3 of Vcc. While *discharge* is on, the NE555 drives *output* to 0 Volts, otherwise *output* is driven to Vcc. We can use this device to make an oscillator by connecting it as shown in Figure 3.

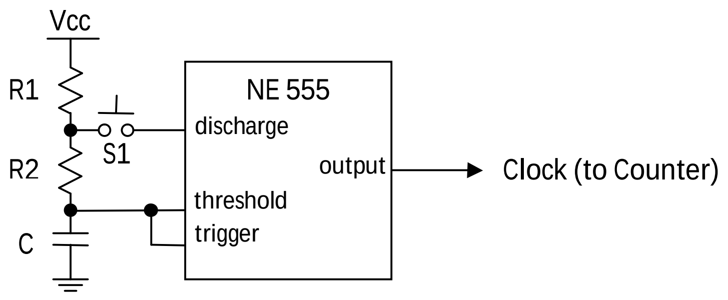

Figure 3 – Making an Oscillator with a 555 Timer

The oscillator functions when pushbutton, S1, is closed (connected). Initially, the capacitor is uncharged, so the voltage on *threshold* (and *trigger*) is 0. Current flows through R1 and R2 to charge the capacitor until its voltage is 2/3 Vcc, which causes the NE555 to connect *discharge* to ground. The capacitor discharges through R2 until the *trigger* (and *threshold*) voltage is 1/3 Vcc. The NE555 then disconnects *discharge* and the capacitor starts to charge again.

Releasing the pushbutton, S1, interrupts the discharge cycle and the oscillator stops. Figure 4 shows the pin diagram for both the NE555 and NE556 timers.

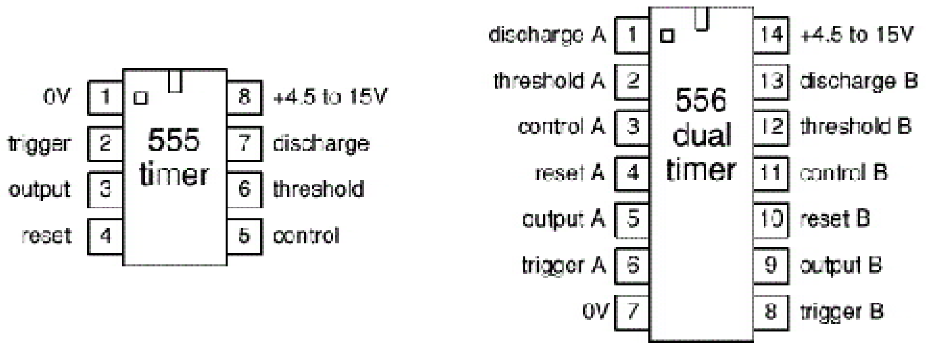

Figure 4 – 555 and 556 Timer Pin Diagrams

To operate, *reset* should be connected to Vcc and *control* should remain unconnected. In this configuration, the frequency of the oscillator can be calculated by:

$$f = \frac{1.44}{C(R1 + 2R2)}$$

You will need to find a suitable capacitor and suitable resistors, with the provision that (a) the resistors are in the range 1kΩ-1MΩ, (b) the capacitor is in the range 0.001μF to 0.1uF, and (c) the frequency is over 200Hz.

Testing: Double-check all your connections, connect the probe of the oscilloscope to the output of the NE555 and apply 5 Volts to Vcc. When the button is pressed, the scope should show square waveform that is high (5V) more than it is low (0V). When the button is released, the output should go low and stay there.

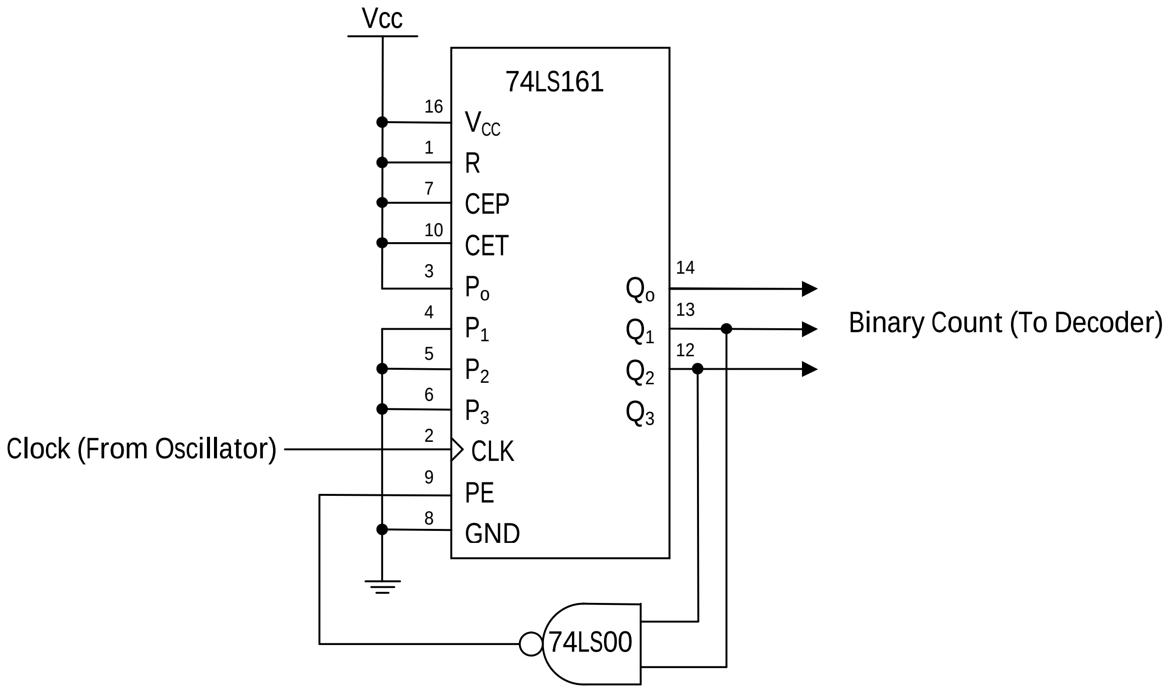

Figure 5 – Modulo 6 Counter

Counter

We can buy devices that count in binary (they are called, unimaginatively, binary counters). One such device is the 74LS161. This counter has 4 outputs, Q_3, Q_2, Q_1 and Q_0 that represent a binary number. On each rising edge of the clock, the count is cleared (if R is low), loaded to a preset value (if PE is low), or incremented. Counting up to 6 is easy because that is what the counter is designed to do, but to make it restart at 1 instead of counting to 7 we'll need to add circuitry. Figure 5 shows one way to make the counter count from 1 to 6 then repeat. When the counter reaches 6 (binary 0110), Q_2 and Q_1 are high (and Q_0 is low), so the NAND gate drives PE low. Consequently, on the next clock, the counter is loaded with 0001 (because P_3, P_2 and P_1 are low while P_0 is high) and the cycle repeats. CEP and CET are count enables that must be tied high (to Vcc). Reset (R) should also be tied high so the counter is never reset. You will study counters in more detail when you get to EE2700.

Testing: Attach your oscillator to the clock and attach the channel 1 scope probe to Q_2. Trigger on the falling edge of Q_2. Apply 5 Volts to Vcc and use the channel 2 probe to verify (one at a time) that Q_0, Q_1 and Q_2 count in binary (see input columns of Figure 6).

Decoder

Careful consideration of Figure 1 will reveal that LEDs 1 and 6 always light together, as do LEDs 2 and 5 and LEDs 3 and 4. So there need be only 4 outputs from the decoder: LED1&6, LED2&5, LED3&4 and LED7. The easiest way to understand a decoder is to draw a truth table. This table specifies exactly which LEDs turns on for each possible binary output of the counter. (See Figure 6).

Inputs			Outputs			
Q_2	Q_1	Q_0	LED1&6	LED2&5	LED3&4	LED 7
0	0	1	0	0	0	1
0	1	0	1	0	0	0
0	1	1	1	0	0	1
1	0	0	1	0	1	0
1	0	1	1	0	1	1
1	1	0	1	1	1	0

Figure 6 – Die Pip Decoder Truth Table

We can relate the outputs of the decoder to the inputs using logic equations (this process will be covered in detail in EE2700):

LED1&6 = Q_2 or Q_1
LED2&5 = Q_2 and Q_1
LED3&4 = Q_2
LED7 = Q_0

Once you have logic equations, it is normally a straightforward process to design a circuit to generate the outputs. We would use an OR gate to get LED1&6 and an AND gate to get LED2&5. LED3&4 would be connected directly to Q_2 and LED7 would be connected directly to Q_0. But there is a twist. The devices in the 74LS family can only deliver enough power to turn on an LED if their outputs are low, which means that all the outputs need to be inverted. So we'll use NAND and NOR gates instead of AND and OR gates, and we'll add inverters for the other outputs. This approach is illustrated in Figure 7.

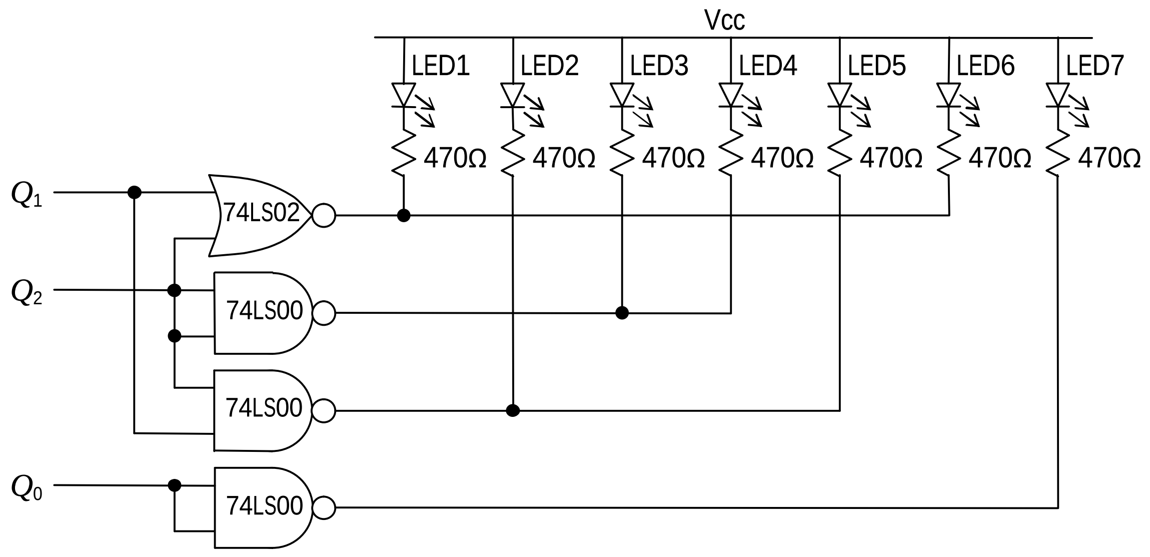

Figure 7 – Decoding Binary to Dice Pip LEDs

Note that Figure 7 shows NAND gates where inverters are expected. This is because NAND gates come four-to-a-package, so it is easier to connect the inputs of two NAND gates together and make an inverter than it is to add an inverter chip (e.g.74LS04).

Testing: Connect up the oscillator and counter and press the button. Repeat until you have seen all six possible outcomes and have verified they are correct. If necessary, replace the oscillator with the function generator and adjust the frequency to 1Hz so you can see each output as the counter cycles.