**CS 152A / EE M116L**

# Introductory Digital Lab Design

**LAB #1**

**Laboratory Familiarization: The Real World**

**April 15, 2014**

**Grade:**

---

**Tianheng Tu**

**Name1: Nathan Tung**

**SID1: 004-059-195**

**Name2: Mark Iskandar**

**SID2: 704-050-889**

## INTRODUCTION

The purpose of Lab 1 is to explore the various devices that will be used throughout this digital design lab. These devices include large equipment like oscilloscopes as well as small electronics like the 74LS series chips, 74HC00 chip, 555 timer, 7-segment display, etc. In general, all the exercises of Lab 1 led us to familiarize ourselves with the breadboard, the devices described above, and the inner workings of the chips.

A brief summary of what each exercise involved follows. In Exercise 1, we examined the NAND gates in both TTL (Transistor-Transistor Logic) and CMOS (Complementary Metal-Oxide Semiconductor) chips to determine what ranges of inputs are acceptable and produce valid voltages. Exercise 2 dealt with the 555 timer, for which we utilized resistors, capacitors, and the potentiometer to generate an oscillating voltage. A glitch is found in Exercise 3 that has to do with different counter types. We then experiment with asynchronous and synchronous counters in Exercises 4 and 5, respectively, and therefore check the changing outputted periods of the different significant bits. In Exercise 6, we construct a ring oscillator, or a chain of an odd number of inverter gates, in order to chart the varying periods of oscillations as we add more and more inverters. Exercise 7 asks us to design an SR gated latch using NAND gates. Finally, in Exercise 8, we utilize Karnaugh Maps, VHDL (VHSIC Hardware Description Language), and the FPGA board in order to create a 7-segment display that is capable of producing all possible numbers (0-9) via four binary switches.

In this report, we will be showing how different outputs arise from different inputs in order to demonstrate the intended effect of each circuit. Answering the specific questions for each exercise may further illuminate the idea and application behind the circuits we have constructed.

## EXERCISES

Exercise 1

Using the NAND gates inside both TTL (Transistor-Transistor Logic) and CMOS (Complementary Metal-Oxide Semiconductor) chips, we test different voltages as input in order to determine whether the experimental output values fall within the theoretical output ranges.

Our data looks something like the following, with units in volts:

TTL (74LS00)

| **Input1** | **Input2** | **Output** |
|---|---|---|
| 2 | 0 | 4.5 |
| 0 | 0 | 4.5 |
| 0 | 2 | 4.5 |
| 2 | 2 | 0 |

CMOS (74HC00)

| Input1 | Input2 | Output |
|---|---|---|
| 0 | 0 | 5 |
| 0 | 5 | 5 |
| 5 | 0 | 5 |
| 5 | 5 | 0 |

**For each experiment, do your output values satisfy the range specified in the datasheets?**
As expected, our output values do satisfy the theoretical range. In fact, a binary 0 is represented by a voltage very close to 0 V (probably around 0.1 or 0.2 V), whereas a binary 1 is represented by a voltage of approximately 4.5 or 5 V, depending on the chip, as shown in the tables above.

**What potential issues could you come across when interfacing TTL and CMOS components?**
We attempted both TTL-CMOS (that is, using TTL outputs as CMOS inputs) and the opposite CMOS-TTL interfaces. TTL-CMOS seems to work just as anticipated. On the other hand, the CMOS-TTL interface gave unexpected, incorrect output values. A problem occurs in the latter case because the output voltage from CMOS is too high for TTL to recognize as input.

Exercise 2

Clocks are important in circuits for producing an oscillation between high and low states that can coordinate actions within the circuit. We can use a 555 timer, along with two 10k resistors and a 10nF capacitor, in order to build a makeshift clock.

Later, we replace one of the resistors with a variable-resistance potentiometer in order to alter the duty cycle of the clock. The duty cycle, or the percentage of a signal's period where it is in a high state, is roughly set to 60% for the rest of this lab (specifically Exercises 2, 3, 4, and 5). In our tests, it was relatively easy to read and interpret the uniform and straight voltage outputs on the oscilloscope display.

**What is the frequency of oscillation?**
Time/Div = 0.5 ms
Period = 2.4 grid lines per cycle
Therefore, the frequency is about 2.4/0.5 ms = 4.8 kHz

**What is the duty cycle of your clock?**
Duty Cycle = t1/(t1+t2) = 8/(8+4) = 2/3 = 66%

**What happens as you turn the left potentiometer knob?**
We discovered that turning the potentiometer knob clockwise increases the duty cycle. At its lowest, the duty cycle is 0%. When the knob is pointing at the 9 o'clock position, the duty cycle reaches 50%. At any point after this, the duty cycle surpasses a 50% duty cycle by an exceedingly greater amount.

Exercise 3

Using Vcc and the clock we constructed above as the input for four connected NAND gates from a TTL (74LS00) chip, we can create a "glitch" output. The first three NAND gates simply take in as input the output of the previous NAND gate, which effectively does nothing but adds delay to the circuit. The last gate NANDs the delayed CLK (clock) and the initial CLK to produce the output GLIT.

**Why is it not possible to measure the width accurately enough?**
We can tell that the output levels do indeed oscillate, but they are not at all uniform or straight. In fact, they are slanted and squiggly, and therefore impossible to measure accurately.

**Why does this glitch occur, and what is the lesson learned from this exercise?**
The glitch occurs when we try to match up the initial CLK input with a CLK input delayed by the sequence of NANDs – combining these inputs from different times does not make sense! The resulting edges are misaligned and that is essentially why the output lines are not uniform as they usually are.

Exercise 4

Using the same clock and the 74LS393 chip, we wire up an asynchronous counter, or a ripple counter. A single ripple counter is capable of storing a bit (and thus counting from zero to one) before overflowing. These can be wired in series for larger counters.

As the lab manual explains, asynchronous counters increment count for every falling edge of the clock such that output bits have twice the period of the next most significant bit. Our test demonstrated just that, with pin 2D producing the longest period and pin 1A with the shortest period.

Exercise 5

In order to create a synchronous counter, we use either the ripple counter's 2D pin (or the most significant bit) from above or another clock as an input into two 74LS163 chips, wired according to the diagram in the manual. By inputting the clock for all counter circuits, we allow all output bits to maintain state in parallel.

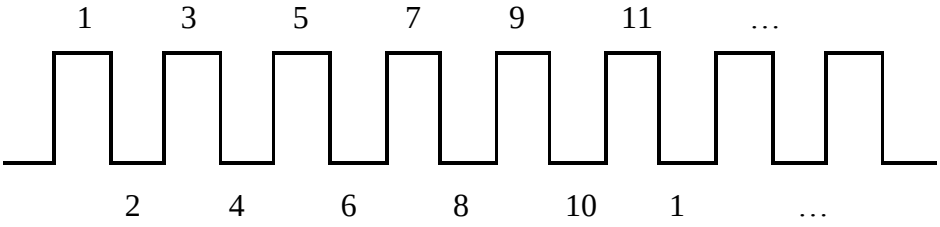
**Explain how RCO output and T input work?**
According to our wiring, RCO (ripple carry out) from the first chip is passed into the second chip's T input. RCO is the overflow value of the first counter; if some overflow is achieved, T gets the message and later prompts the counter to reset bits that are less significant. Our synchronous counter is dependent on this mechanism to keep track of larger counts.

**How many Flip-Flops and how much logic is required to implement each counter?**
The 4-bit synchronous counter that we modeled uses four (JK) flip-flops. Since an AND gate is required for each bit apart from the first and last, such a design will require two AND gates at a bare minimum.

Exercise 6

Now we will build a ring oscillator, or an odd number of inverters connected in series, using the 74LS04 chip, which contains six inverter gates for us to use. We connect the output from an inverter gate to the input from the next inverter gate, and so on. Eventually, the final output from the last inverter loops around and becomes the input of the first inverter gate. We attempt this for two ring oscillators of 11 and 17 inverters in each, using two or three chips as necessary.

**Using a timing diagram, explain why this circuit oscillates.**
As explained above, an odd amount of inverters is necessary for our ring oscillator to actually oscillate. If an even amount of inverters is used, then the final output that is fed back into the ring of inverters is the same as the original input, causing no oscillation. On the other hand, an odd number of inverters will flip the output continuously, albeit at a delay.




**What is the frequency of oscillation?**
Time/Div = 1 $\mu$s with 10x magnification
Period = 1.2 grid lines per cycle
Therefore, the frequency is about 0.12 $\mu$s (or 8.33 MHz)

**What should the period of oscillation be with 17 inverters in the ring?**
With more inverters, the delay should increase, causing the frequency to increase as well.
We find it to be:
Time/Div = 1 $\mu$s with 10x magnification
Period = 1.9 grid lines per cycle
Therefore, the frequency is about 0.19 $\mu$s (or 5.26 MHz)

**What are the new values of minimum and maximum voltage?**
The minimum voltage in both cases is very close to 0 V. The new maximum voltage (that is, in the case of 17 inverters) may have been slightly higher. However, both old and new values of the maximum voltage measured at approximately 4 V.

Exercise 7

We construct a gated SR latch using four NAND gates from both the TTL and CMOS chips. Since the pin positioning in these chips are identical, it is easy to replace the entire chip on the breadboard and examine any resulting differences. The circuit itself is wired using the clock as the enable bit, which feeds into two NAND gates with R (reset) and S (set). The outputs of these two NAND gates are fed into another set of NAND gates which cross over, using one NAND gate's output as input for the other NAND gate.

When the gated SR latch is enabled, it operates like a normal SR latch as shown in the Prelab. However, when it is disabled (or when enable bit E=0), the latch remembers its last enabled state.

A summary truth table of the enabled gated SR latch is shown below:

TTL

| **S** | **R** | $\boldsymbol{Q}^+$ | $\overline{\boldsymbol{Q}}^+$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | CLK |

CMOS

| **S** | **R** | $\boldsymbol{Q}^+$ | $\overline{\boldsymbol{Q}}^+$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 1 | CLK | $CLK^{-1}$ |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 |

Exercise 8

This last exercise involved programming in VHDL code in order to display all digits from 0 through 9 by flipping through four binary switches on the FPGA board. The first step was designing a truth table relating each digit to be displayed with each of the 7-segment parts that would be lit up. Next, Karnaugh maps were drawn for each of the seven segments. By extracting the Sum of Products equation for each segment, we were able to code this design into VHDL while linking the variables to their physical hardware (switches, segment displays, etc.) on the FPGA.

The7-segment display indeed showed the values corresponding to the Boolean values of the switches, demonstrating that our program was successful.

## CONCLUSION

This report addresses the background behind the equipment and circuits that were involved and follows the thought process that ultimately led us to design and build these circuits. We also take a look at the outcomes of these eight exercises and scrutinize them within a digital design context to gain insight for future labs.

We faced challenges in both hardware and software. In terms of the hardware and equipment, it was often difficult to figure out how to best wire the circuits on the breadboard and how to easily pinpoint errors in the way our circuits are wired. To remedy this, we would draw diagrams with numbered pins all while verifying that the chips are working and powered on. On software side, it was hard to debug minor errors in VHDL due to the syntactical sensitivity of the programming language. Our solution was to unit test the syntax until we found the erroneous line.

Overall, Lab 1 was a valuable experience in laying down a foundation for future labs.

Lab Contribution:
Nathan Tung: 50%
Mark Iskandar: 50%