**Digital Circuits Lab**

**Experiment 10**

**CENG 2112.01**

Submitted by

Yevgeniy Potapov

2244991

Computer Science

University of Houston-Clear Lake

Houston, Texas 77058

May 15, 2023

Table of Contents

[Abstract 3](#_Toc134789214)

[Introduction 3](#_Toc134789215)

[Requirement 3](#_Toc134789216)

[Prelab 3](#_Toc134789217)

[Implementation 9](#_Toc134789218)

[1.0 3-bit Up Counter 9](#_Toc134789219)

[1.1 Multisim Simulation 9](#_Toc134789220)

[2.0 4-bit Sequence Counter 10](#_Toc134789221)

[2.1 Multisim Simulation 11](#_Toc134789222)

[3.0 3-bit Up Counter 13](#_Toc134789223)

[3.1 Multisim Simulation 13](#_Toc134789224)

[Results 14](#_Toc134789225)

# Abstract

A final demonstration of circuit logic is performed. A 3-bit Up counter was modeled in Multisim, and demonstrated an Up Counter behavior. A 4-bit Sequential Counter was modeled in Multisim, and demonstrated the sequence ““9,1,2,8,3,74,6,5,0””. A 22 bit Up Counter was modeled in Multisim and stops at the student ID number.

# Introduction

Sequential Circuits use memory to store p[previous data and change outputs. Using JK flip flops; this experiment shows the function of sequence circuits to model counters.

# Requirement

For understanding the topic material, an understanding of multiplexers is needed to formulate the circuit. A TTL data book, or an online resource (A.K.A. Google) will be used to acquire part numbers. Multisim will be used to model the circuit in software.

# Prelab

3-bit Up Counter

1. Prelab #1 – Design a 3-bit up counter. Implement your design using JK flip-flops (74LS112) and any necessary gates. The output of the circuit will be the states of the flip-flops and an output Z = AB (Note: A and B).

3-bit Up Counter Truth Table

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Qc | Qb | Qa | Q+c | Q+b | Q+a | J1 | J2 | J3 | K1 | K2 | K3 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | X | X | X |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | X | X | X | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | X | 1 | X | 0 | X |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | X | X | X | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | X | 0 | 1 | 0 | X | X |
| 1 | 0 | 1 | 1 | 1 | 0 | X | 1 | X | 0 | X | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | X | X | 1 | 0 | 0 | X |
| 1 | 1 | 1 | 0 | 0 | 0 | X | X | X | 1 | 1 | 1 |

Up counter K-maps

**CB\A – > J1 = QaQb**

|  |  |  |
| --- | --- | --- |
|  | **0** | **1** |
| **00** | 0 | 0 |
| **01** | 0 | 1 |
| **11** | X | X |
| **10** | X | X |

**CB\A – > J2 = Qa**

|  |  |  |
| --- | --- | --- |
|  | **0** | **1** |
| **00** | 0 | 1 |
| **01** | x | x |
| **11** | x | x |
| **10** | 0 | 1 |

**CB\A – > J3 = Qa’**

|  |  |  |
| --- | --- | --- |
|  | **0** | **1** |
| **00** | 1 | x |
| **01** | 1 | x |
| **11** | 1 | x |
| **10** | 1 | x |

**CB\A – > k1 = QaQb**

|  |  |  |
| --- | --- | --- |
|  | **0** | **1** |
| **00** | x | X |
| **01** | X | x |
| **11** | 0 | 1 |
| **10** | 0 | 0 |

**CB\A – > k2 = QaQb**

|  |  |  |
| --- | --- | --- |
|  | **0** | **1** |
| **00** | x | X |
| **01** | 0 | 1 |
| **11** | 0 | 1 |
| **10** | x | x |

**CB\A – > k3 = Qa**

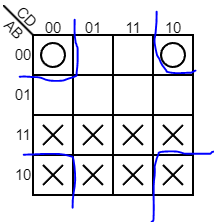
|  |  |  |
| --- | --- | --- |
|  | **0** | **1** |
| **00** | x | 1 |
| **01** | x | 1 |
| **11** | x | 1 |
| **10** | x | 1 |

4-bit Sequence Counter

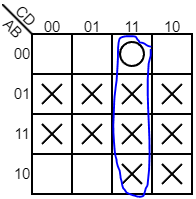
Design a 4-bit counter that produce the following sequence: 0,9,1,2,8,3,7,4,6,5,0, … Implement your design using JK flip-flops (74LS112) and any necessary gates.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| QA | QB | QC | QD | Q+A | Q+B | Q+C | Q+D | J1 | J2 | J3 | J4 | K1 | K2 | K3 | K4 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | X | X | X | X |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | X | X | X | X | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | X | 0 | X | X | 1 | X |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | X | X | X | X | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | X | 1 | 0 | X | 0 | X | X |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | X | 0 | X | X | 1 | X | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | X | X | 1 | X | 0 | 1 | X |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | X | X | X | X | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | X | 0 | 1 | 1 | 1 | X | X | X |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | X | 0 | 0 | X | 1 | X | X | 0 |
| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

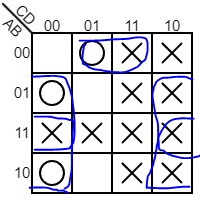
J1 = B’D’



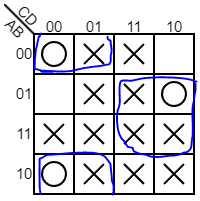
J2 = CD =



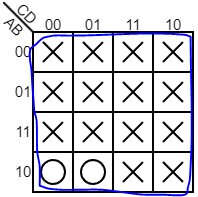
J3 = BD’+AD’+A’B’D = QbQd’+QaQd’+Qa’Qb’Qd



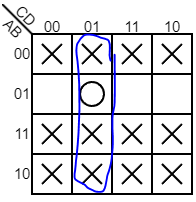
J4 = B’C’+BC = Qb’Qc’+QbQc



K1 = 1 = VCC



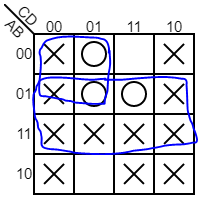
K2 = C’D = Qc’Qd



K3 = B+D’ = Qb+Qd’



K4 = B + A’C’



# Implementation

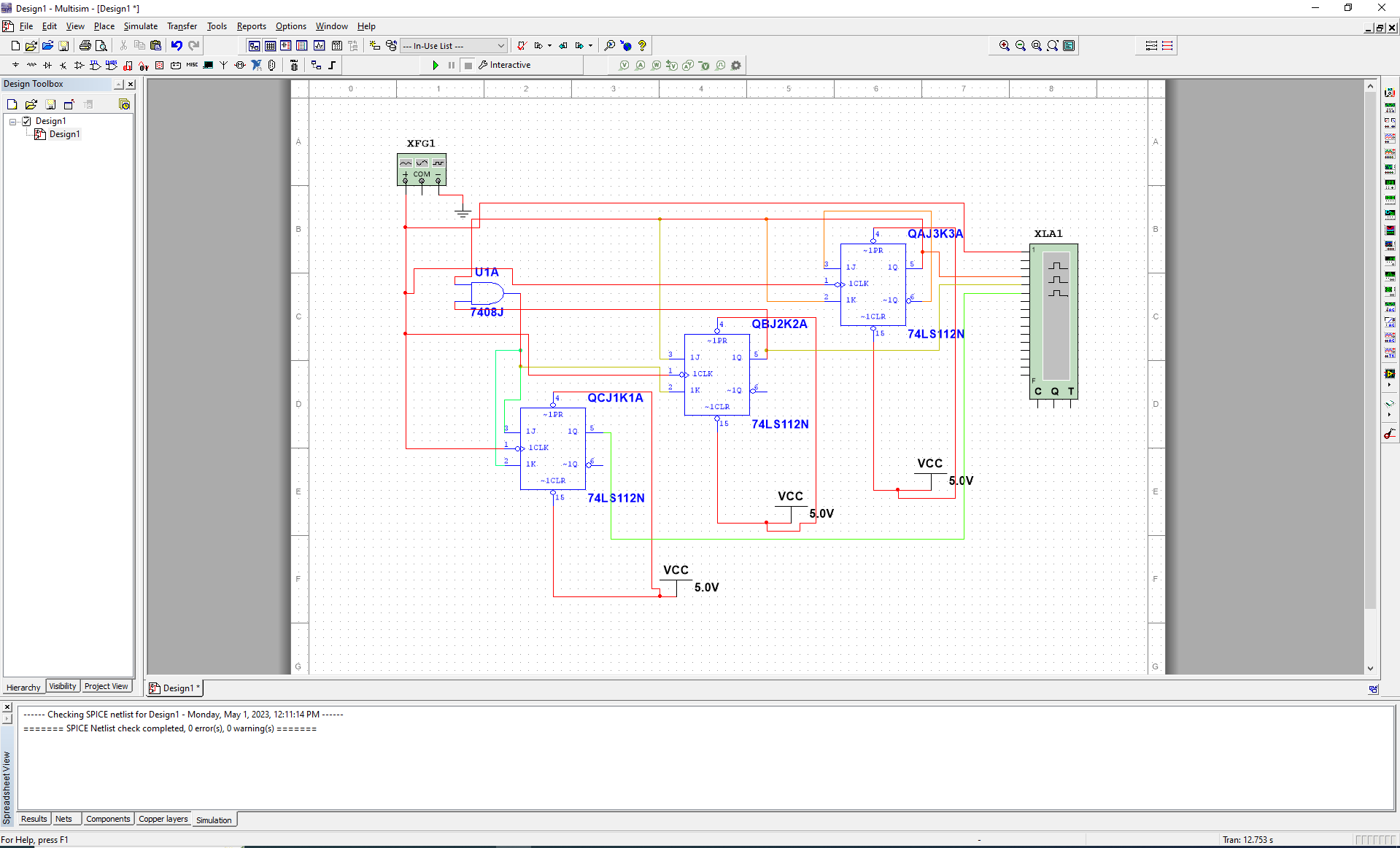
## 1.0 3-bit Up Counter

The resulting circuit from the prelab was modeled into Multisim. Multisim used Function Generator at Square wave, 200 Hz, 50% duty cycle, 10 Amp, 0 offset.

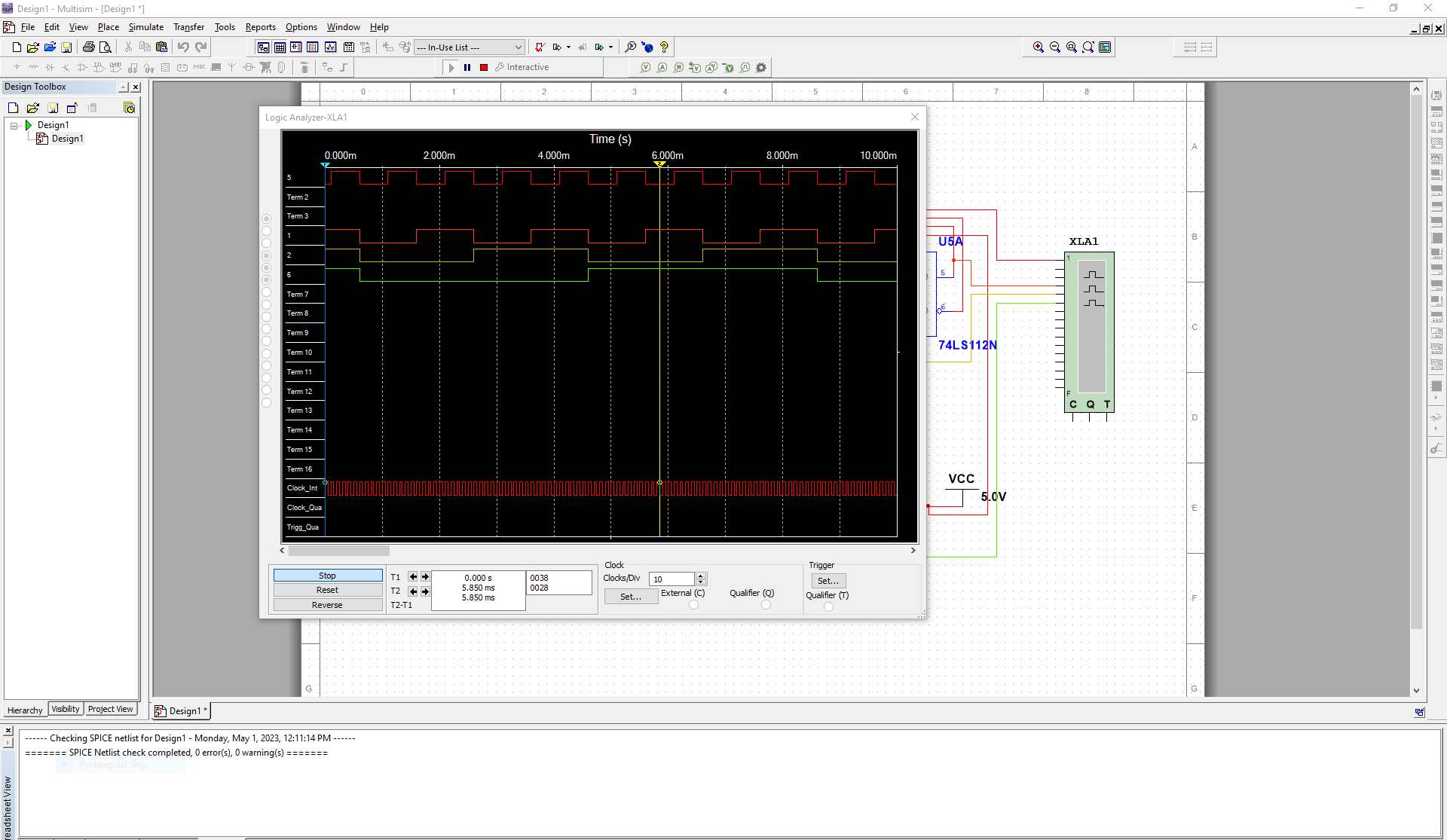
### Multisim Simulation

Multisim Software was used to model the circuit. After opening Multisim, the parts were loaded, word generator, analyzer, Function Generator at Square wave, 200 Hz, 50% duty cycle, 10 Amp, 0 offset, and 3 JK flip flops connected according to the prelab.

The circuit from the prelab was wired using the Multisim software. The word generator wires into each of the Q output of each flip flop, and the clock from the function generator. The clock is Red, and the lowest bit is Orange.



The resulting waveform is shown below.



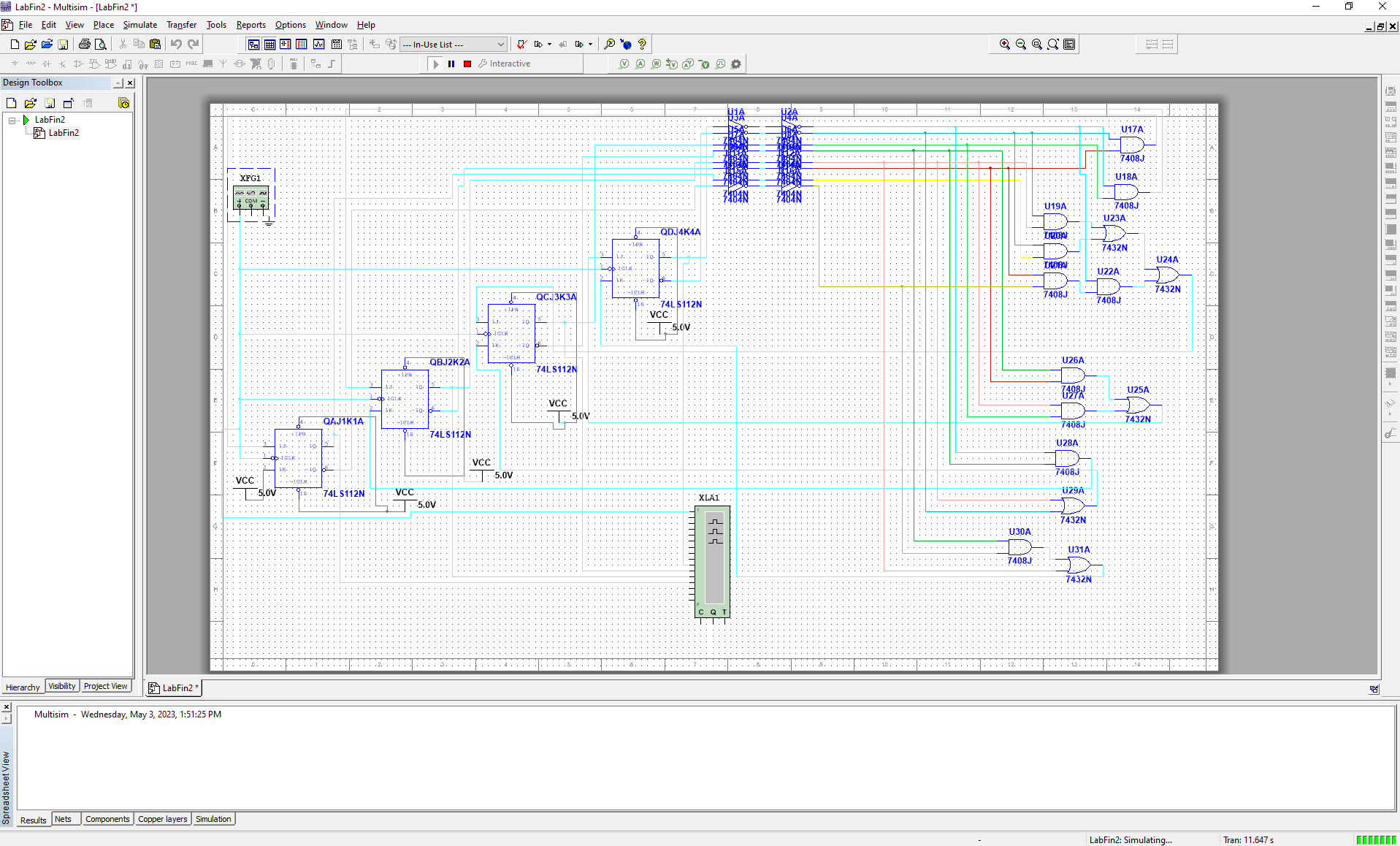
## 2.0 4-bit Sequence Counter

The resulting circuit from the prelab was modeled into Multisim. Multisim used Function Generator at Square wave, 200 Hz, 50% duty cycle, 10 Amp, 0 offset.

### Multisim Simulation

Multisim Software was used to model the circuit. After opening Multisim, the parts were loaded, word generator, analyzer, Function Generator at Square wave, 200 Hz, 50% duty cycle, 10 Amp, 0 offset, and 4 JK flip flops connected according to the prelab. There is a set of double inverters in the top right corner used only for cabling, and are not necessary for the function of the circuit.

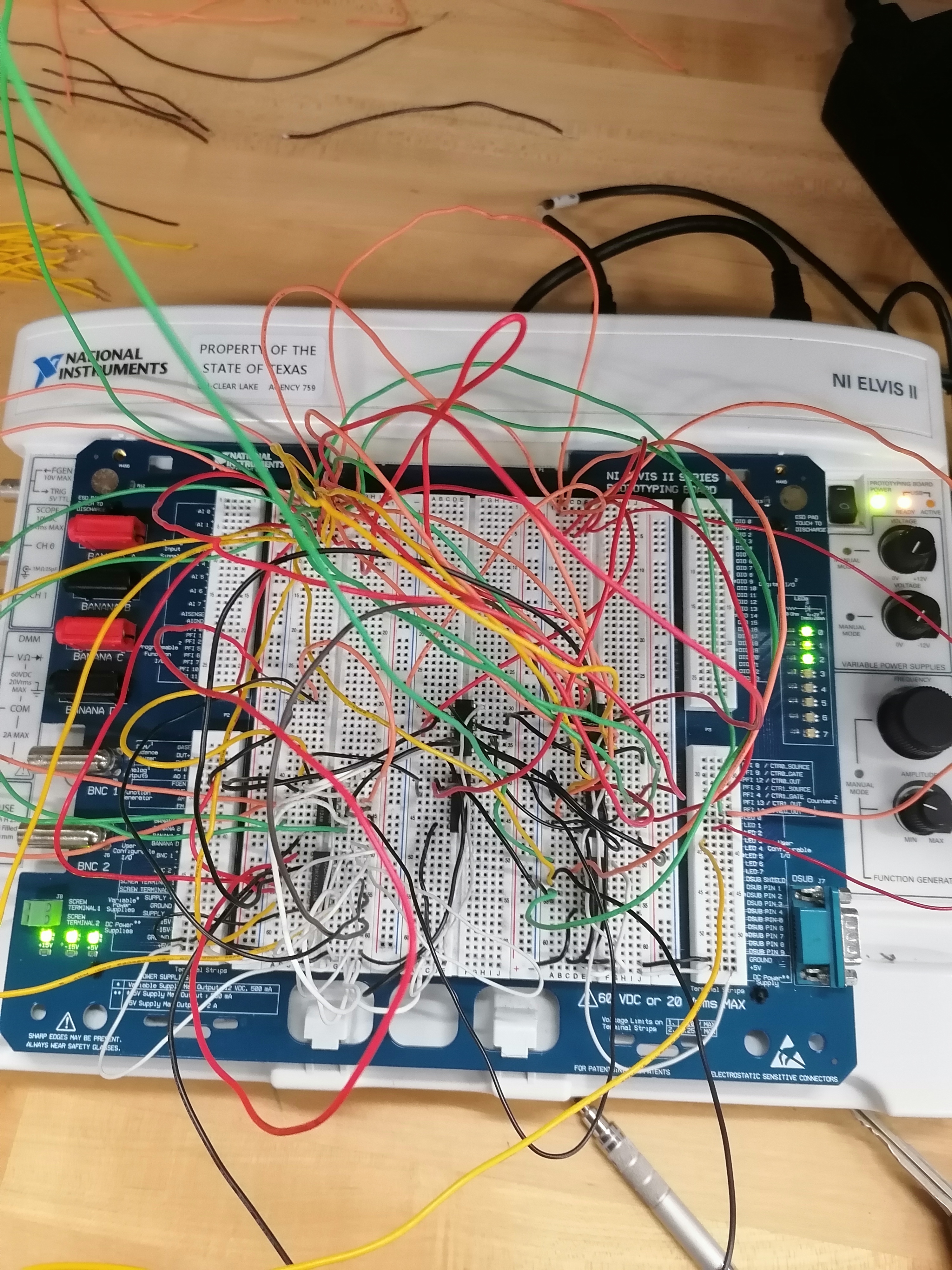
The circuit from the prelab was wired using the Multisim software. The word generator wires into each of the Q output of each flip flop, and the clock from the function generator. The clock is Blue, and the lowest bit is Blue.



The resulting waveform is shown below.



The NiElvis Model shown below



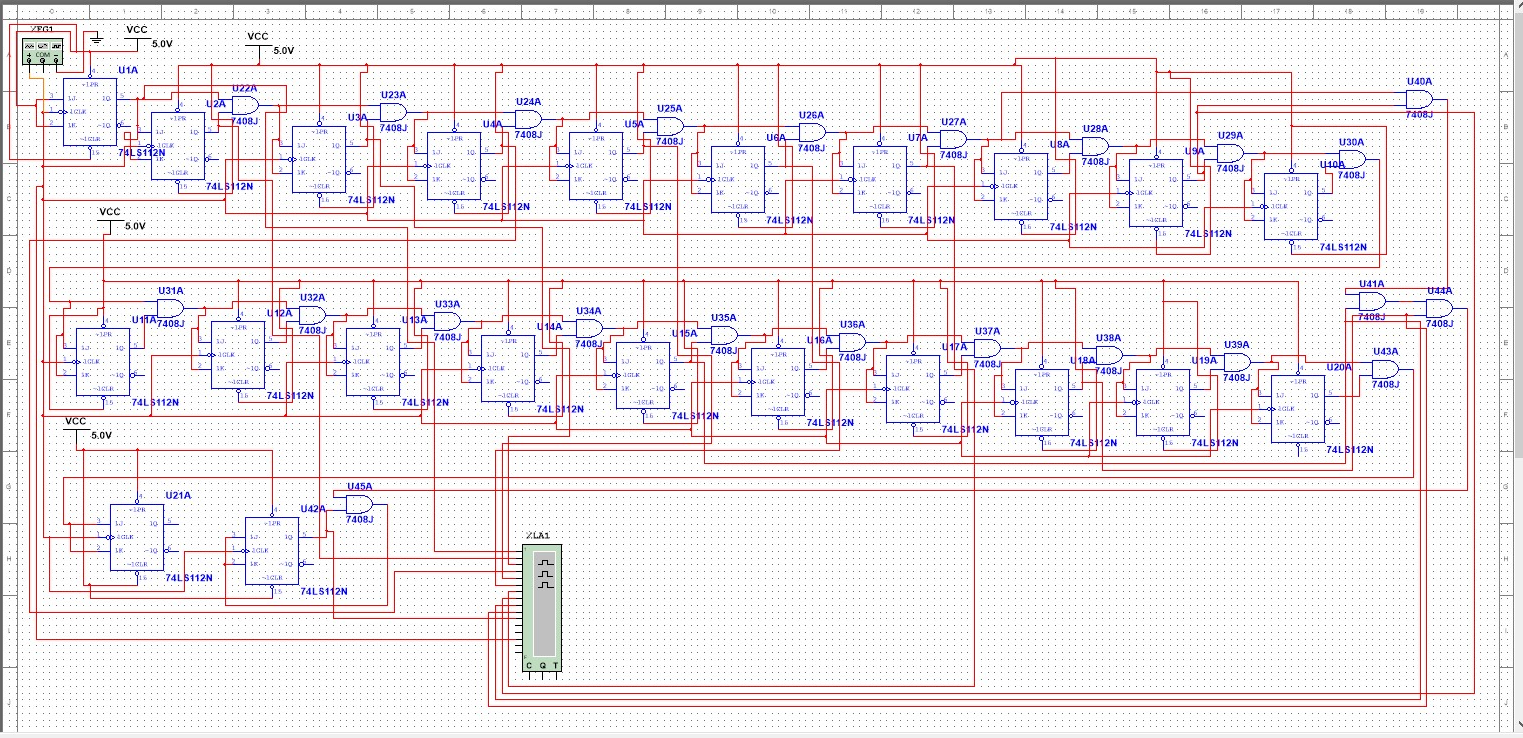
## 3.0 3-bit Up Counter

Multisim used Function Generator at Square wave, 200 Hz, 50% duty cycle, 10 Amp, 0 offset.

### Multisim Simulation

Multisim Software was used to model the circuit. After opening Multisim, the parts were loaded, word generator, analyzer, Function Generator at Square wave, 200 Hz, 50% duty cycle, 10 Amp, 0 offset, and 22 JK flip flops.

Using the circuit logic from Part 1, each previous Q output of a JK flip flop was AND’d to the next J and K inputs to form the Up Counter action, until the final flip flop, which uses the Student ID number to stop the counter and reset to 0.



# Results

3-bit Up Counter

The resulting waveform from the Multisim schematic shows the functioning Up Counter. The initial Output Waves Orange Yellow Green(lowest bit Orange) show the mystery condition, in which the state of the flip flops is unknown. After that point, we see the waveform show ascending 3-bit binary numbers. The first is 0, with all outputs low, then we see 001, with Orange being high, and it progresses to 7, when all 3 waves are high, and after that all 3 waves drop down to 0 again.

4-bit Sequence Counter

The resulting waveform from the Multisim schematic shows the functioning Sequence Counter. The sequence needed is “9,1,2,8,3,74,6,5,0”, in repetition. The Selected Area, between the Yellow and Blue vertical prongs, display this behavior. The 4 output waves BLUE GREEN RED YELLOW(Blue lowest) start with the waves High Low Low High, which demonstrates a 9 as a binary number. The following number is High only on blue, which shows a binary 1. Following this, a 2 is shown. This continues until we see the end of the sequence as 0 right before the Blue Vertical Prong.