**Stoplight with LED Counter**

Machine Project Documentation

for the course on

Database Design

(DIGIDES)

Submitted by:

Amadora, Angelo John

Aquino, Kurt Neil

Choy, Matthew Seaver

Ramos, Luis Angelo

Dr. Joel Ilao

Professor

August 19, 2014

1. **Introduction**
2. **Abstract**

Digital Designing is the organization and logic design of digital systems. This course presents a structured design philosophy, emphasizing hardwired and microprogrammed control, boolean algebra, hardware building blocks, circuit synthesis, as well as microprogramming. Digital systems have such a prominent role in everyday life that we refer to the present technological period as the digital age. Digital systems are used in communication, business transactions, traffic control, spacecraft guidance, medical treatment, weather monitoring, the Internet, and many other commercial, industrial, and scientific enterprises.

One characteristic of digital systems is their ability to represent and manipulate discrete elements of information. Any set that is restricted to a finite number of elements contains discrete information. This is what coins the term “Logic Design”.

For this course’s final output, students are required to imitate, produce, or invent their very own digital system. Using lessons learned during lectures, the objective of this project is to prove that the students have understood the basic concepts of logic analysis, manipulation, as well as application. From flashing LEDs to making sensors, there are no limits as to what the students may be able to do for their machine outputs.

For our group’s case, we decided to implement one of the more common applications of digital systems, traffic control, to be more specific, a stoplight system with an LED counter.

As most traffic light systems are being controlled with built-in processors, so that they may be remotely operated from a station or whatnot, our own implementation will only make use of simple logic gates and binary manipulation. Through the use of data analysis, logic representation, and basic circuitry, our group aims to apply the lessons taught in this course and represent it by using the digital system that we wish to simulate.

For this project, Our group will be using a seven segment LED display, and multiple LEDs (green, yellow, and red) as our outputs, a function generator as our input, being the one that produces the pulses as a timer function for our system, and finally, flip flops as well as multiple logic gates for the data manipulation.

1. **Objective**

To further explain the desired output of our group’s project, we wish to implement a single traffic light system with a fully working counter. The seven segment LED display will count down from nine to four (9-4) with the green LED lit, from three to zero (3-0) with yellow, and finally, from five to zero (5-0) with red. After a single cycle (16 counts), our system will then loop again and restart the count.

1. **Methodology**
2. **Discussion of the System**

Before any actual implementation should be done for any digital system, one should first discuss as to why you need the system. As mentioned earlier, we wish to imitate one of the systems being used for traffic control, a stop light. Next is how you wish to present the outputs, which was discussed in this paper’s objective; and finally how will you implement the circuit system itself.

For starters, since our group’s project relies on sixteen (16) counts per cycle, we will have to implement a system that will use four bits as inputs; although we may still be able to use othe timer circuits such a 555 relay or a 4017 counter for the system, this will require other parts that have not been introduced in this course, and for the reason that we also wish to keep our system simple and easy to understand so that others, especially beginners, may easily implement the same or similar systems as well. Given that circuitry only produces and accepts 0’s and 1’s as data, we will have apply that into out system accourdingly. For every set of counts, our system will display different outputs.

Next is how to manipulate the output pattern of the seven segement LED display. There are two diferent types of seven segment LED displays, one that has four inputs and one that has seven. For the four input display, it accepts binary logic to determine its output. Since it relies on the data being received by its inputs, it will only display specific patterns. 0 and 1, 0 to 3, 0 to 7, 0 to 9 and A to F, for one, two, three, and four inputs respectively. This corresponds to the concept that 2 raised to the number of inputs will result to the number of outputs (e.g. 2^4 = 16). Our group’s system requires a 4 bit input, but we will not rely on the 7 segment display which requires 4 as mentioned earlier, as it displays a fixed sequence of patterns only. For our system, we used the 7 segment display which requires 7 inputs. Each of its inputs corresponds to a single segment; this allows us to manipulate which segments should only be lit at a given time, sequence, or cycle, depending on the implementation.

1. **Discussion of Parts**

For this part of the paper, we will only discuss the less common parts being used, from a beginners point of view. That being said, we will not discuss how the logic gates work, the LEDs, as well as the seven segment LED display. To be specific we will only discuss the functionality of function generators, the one that provides the pulses for the clock/timing of the system, and JK Flip flops, as the component that we will make use of to provide our system with the 4 bit sequence (16 count output).

**Function Generator**

Function generators are important for an electrical circuit. It produces different kinds of signals, and it can be used as a signal source. The four most common waves or signals it produces are sine, triangular, square, sawtooth waves. All of these waves can have their ranges adjusted. This allows function generators to have an extremely wide variety of different waves it can produce. Function generators have the ability of producing two different waveforms simultaneously. It can also phase lock into a certain external signal source which allows the function generator to be constant.

**JK Flip Flops**

In the field of electronics, the flip-flop or so called a “latch” is a kind of circuit that can be used to store state information. The flip-flop is also a bistable multivibrator, meaning that it is able to deliver the information from one state to the other. Usually, people implement a multivibrator whenever the circuit requires a timed interval. The flip-flop is mainly used as data storage elements for the circuit. The data they store is often called a “state” and the circuit they are implemented on is described as Sequential Logic. When a flip-flop is used in a finite -state machine meaning that the machine has an expected output based on the current input and the previous input. The flip-flop acts as a memory device for the circuit.

The JK flip-flop is widely used among all the other flip-flops that are currently made. The JK flip – flop is also considered to be a universal flip – flop. The JK flip – flop works exactly the same as an SR flip – flop with the same “set” and “reset” inputs. The SR and the JK flip – flop have the same use, however, their main difference is their input states. The JK has no invalid or forbidden input states unlike the SR flip – flop.

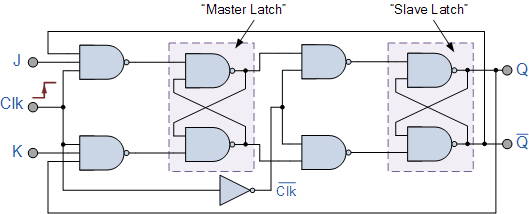


Figure 1. JK Flip flop component

1. **Implementation**

In our group’s case, this will be the truth table that will determine which segments of the counter and which LEDs should only be lit at a given sequence:

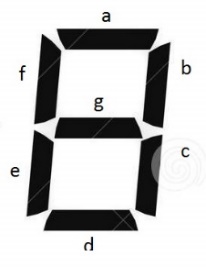


Figure 2. 7 segment display and its segments.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | green | yellow | red | a | b | c | d | e | f | g |
| 0 | **0** | **0** | **0** | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 0 | **0** | **0** | **1** | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | **0** | **1** | **0** | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | **0** | **1** | **1** | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | **1** | **0** | **0** | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | **1** | **0** | **1** | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | **1** | **1** | **0** | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0 | **1** | **1** | **1** | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | **0** | **0** | **0** | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | **0** | **0** | **1** | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | **0** | **1** | **0** | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | **0** | **1** | **1** | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | **1** | **0** | **0** | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | **1** | **0** | **1** | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | **1** | **1** | **0** | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | **1** | **1** | **1** | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

Table 1. The truth table of the 7 segment display and LEDs

And the state transition is represented as follows:

Diagram 1. State transition diagram of the system

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Circuit Input | Current State | | | | Next State | | | | Circuit Output | | |
| 1 | **A** | **B** | **C** | **D** | A | B | C | D | green | yellow | red |
| 1 | **0** | **0** | **0** | **0** | 0 | **0** | **0** | **1** | 1 | 0 | 0 |
| 1 | **0** | **0** | **0** | **1** | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | **0** | **0** | **1** | **0** | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | **0** | **0** | **1** | **1** | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | **0** | **1** | **0** | **0** | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | **0** | **1** | **0** | **1** | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | **0** | **1** | **1** | **0** | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | **0** | **1** | **1** | **1** | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | **1** | **0** | **0** | **0** | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | **1** | **0** | **0** | **1** | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | **1** | **0** | **1** | **0** | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | **1** | **0** | **1** | **1** | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | **1** | **1** | **0** | **0** | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | **1** | **1** | **0** | **1** | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | **1** | **1** | **1** | **0** | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | **1** | **1** | **1** | **1** | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Table 2. State Transition Table for the LEDs

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Circuit Input | Current State | | | | Next State | | | | Circuit Output | | | | | | |
| 1 | A | B | C | D | A | B | C | D | a | b | c | d | e | f | g |
| 1 | **0** | **0** | **0** | **0** | 0 | **0** | **0** | **1** | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | **0** | **0** | **0** | **1** | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | **0** | **0** | **1** | **0** | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | **0** | **0** | **1** | **1** | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | **0** | **1** | **0** | **0** | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | **0** | **1** | **0** | **1** | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | **0** | **1** | **1** | **0** | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | **0** | **1** | **1** | **1** | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | **1** | **0** | **0** | **0** | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | **1** | **0** | **0** | **1** | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | **1** | **0** | **1** | **0** | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | **1** | **0** | **1** | **1** | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | **1** | **1** | **0** | **0** | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | **1** | **1** | **0** | **1** | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | **1** | **1** | **1** | **0** | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | **1** | **1** | **1** | **1** | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

Table 2. State Transition Table for the 7 segment display

Given the data from the truth table, we derived the minterm list of every output (each segment and the LEDs) and simplified it using a Karnaugh map. With the state transition diagram, we were able to determine how many JK flip flops we must use, as well as establish the manipulation needed to control its output into a 4 bit sequence. Given a funtion genrator, JK or D flip flops, as well as a few logic gates, we were able to set up a simple system which produces the binary numbers needed as inputs for the 7 segment display. In a sense we used an input, the function generator, to produce an output, with the JK/D flip flop system, and used it as an input for the 7 segment display, as well as the LEDs, to produce an output.

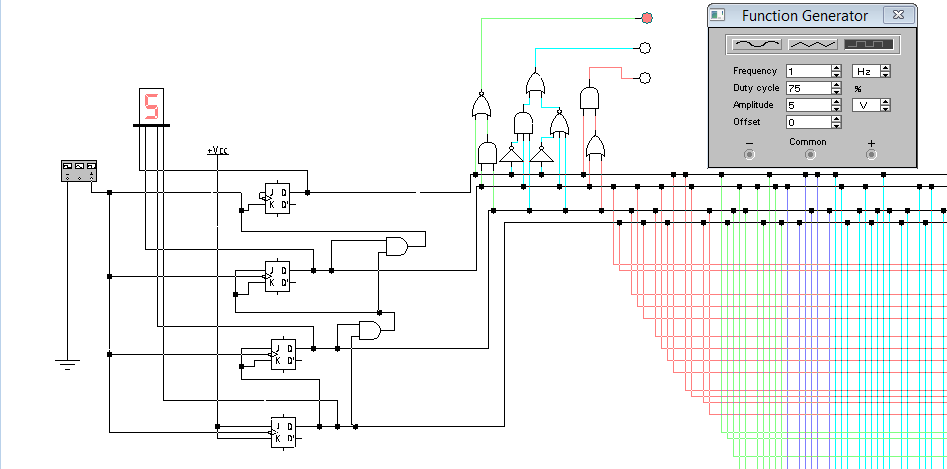


Figure 3. 4 bit Output Generator + LED Stoplight

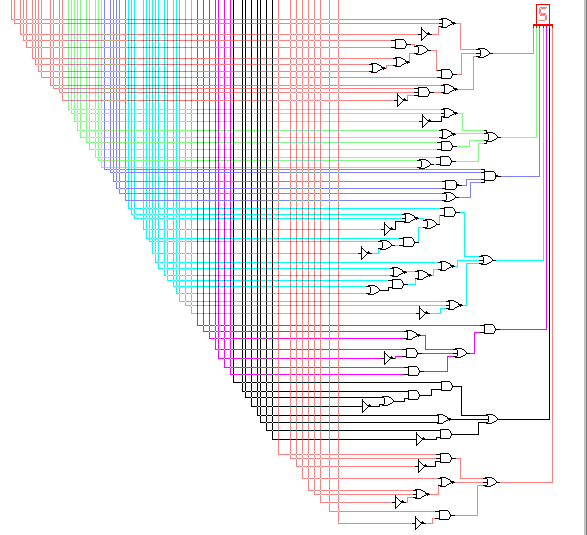


Figure 4. 7 Segment Display Connections + Binary manipulation implementation

1. **Data**

These are the Karnaugh maps of each output as derived from the truth table above, as well as their corresponding sum of products:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AB\CD | 00 | 01 | 11 | 10 |
| 00 | 1 | 1 | 1 | 1 |
| 01 | 1 | 1 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AB\CD | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 1 | 1 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 1 | 1 | 0 | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AB\CD | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 1 | 1 | 1 | 1 |
| 10 | 0 | 0 | 1 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AB\CD | 00 | 01 | 11 | 10 |
| 00 | 1 | 1 | 1 | 1 |
| 01 | 1 | 0 | 1 | 1 |
| 11 | 1 | 1 | 1 | 0 |
| 10 | 0 | 1 | 0 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AB\CD | 00 | 01 | 11 | 10 |
| 00 | 1 | 1 | 0 | 1 |
| 01 | 0 | 1 | 1 | 1 |
| 11 | 1 | 1 | 1 | 1 |
| 10 | 1 | 1 | 1 | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AB\CD | 00 | 01 | 11 | 10 |
| 00 | 1 | 1 | 1 | 1 |
| 01 | 1 | 1 | 0 | 1 |
| 11 | 1 | 0 | 1 | 0 |
| 10 | 1 | 1 | 1 | 1 |

**c** = A’C’ + AC + B’ + D’

**Red** = AB + AC

**b** = A’CD’ + B’C’ +BD +AB + AD

**a** = B’CD’ + AB’C + ABD + AC’D + A’B’ + A’D’ + A’C

**Yellow** = A’BC + AB’C’

**Green** = A’B’ + A’C’

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AB\CD | 00 | 01 | 11 | 10 |
| 00 | 1 | 1 | 1 | 0 |
| 01 | 1 | 0 | 1 | 1 |
| 11 | 1 | 1 | 1 | 0 |
| 10 | 0 | 1 | 0 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AB\CD | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 1 | 0 |
| 01 | 0 | 0 | 1 | 0 |
| 11 | 0 | 1 | 1 | 0 |
| 10 | 0 | 1 | 0 | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AB\CD | 00 | 01 | 11 | 10 |
| 00 | 1 | 1 | 1 | 1 |
| 01 | 1 | 0 | 1 | 1 |
| 11 | 1 | 1 | 1 | 0 |
| 10 | 0 | 1 | 0 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AB\CD | 00 | 01 | 11 | 10 |
| 00 | 1 | 1 | 1 | 0 |
| 01 | 1 | 1 | 1 | 1 |
| 11 | 1 | 1 | 0 | 0 |
| 10 | 0 | 0 | 1 | 1 |

**f** = ACD + AB’C + A’C’ +B’D

**e** = B’C’D + A’CD + AC’D

**d** = AB’CD’ + A’C’D’ + B’C’D + A’CD + A’BC + AB’C +ABD

**g** = AB’C + A’C’ + A’D + A’B + B’C

1. **Data Analysis**

Even with the use of Karnaugh maps, the sums of products of each output will still require quite a lot of AND and OR gates. This is where we applied boolean algebra in order to further simplify the sums of products in terms of the number of gates being used. The fewer number of gates being used, the better for every system, as it lessens the connections being applied, optimizes its functionality, and removes any redundancy in the process. As for our system, since we did not use any microcontrollers or programmable components, we heavily relied on logic gates for the manipulation, as well as the control, of the flow of data. Here are the simplified equations for each output:

* Green = (A + (BC))’
* Yellow = A’BC + (A’ + B + C)’
* Red = A(B + C)
* a = (B + C’ +D)’ + A(BD + (C + (B + D)’)’) + (A + (BDC’)’)’
* b = (A + C’ + D)’ + (B + C)’ + BD + A(B + D)
* c = ((A + C) (AC)’ BD)’
* d = A((B + C’ + D)’ + B(C’ + D)) + (A + ((C + D)’ + C(D + B)’)’ + (B + C + D’)’
* e = D((C + (B’ + A)’)’ + A’C)
* f = AB’C + (A + (C’ + D + B)’)’ + BC’

Though it may look even more complicated than the previous equations, this actualy simplifies the number of gates being used, as well as the number of connections it requires from the 4 bit output generator (JK flip flop + function generator circuitry). However, no matter how simplified this is, there are still a considerable amount of gates being used. The only way to actually lessen or even remove the use of logic gates completely is by using Arduino, a programmable circuit that you may apply to your system, or a microcontroller; both of which automates the entire process of binary manipulation, but on the other hand, this defeats the purpose of applying of what we have learned throughout the period of this course.

1. **Recommendations**

This project, all in all, is already quite logically simple. It uses the very basics of binary logic and manipulation, boolean algebra, as well as circuitry. The only way for those who wish to implement the same or a similar system and make it even simpler is by using programmable components such as Arduino or microcontrollers. Depite the price, it optimizes the entire system, leaving you with more space for improvement, modifications, and functionality for your entire system.

To even simplify the entire system, you may use timer circuits such as a 555 relay and a 4017 counter to directly provide your system with a clock. This further optimizes your system as it uses less components and provides you with the inputs you need directly. Rather than making a seperate system and use it to provide another system with the inputs it needs, it is more efficient to use circuits and components that does it for you. With further modifications, you may even have the option of adding a delay switch in order to implement another stoplight and counter.

1. **Conclusion**

The project we chose to implement presents the very foundation of digital systems and logic design. Nothing overly complicated was implemented, and the functionality of the entire system itself is easy to understand. It uses basic components, and applies boolean and binary logic taught early in this course to produce the desired output. Though our group could have applied the later parts of cirucitry in this course, we chose to stick with the appication of binary logic, programmable logic arrays to be specific, in our system as it is the one that is mostly taught early on in most, if not all, logic oriented courses. On the other hand, with further study and research, as well as the application of a larger variety of circuits and components, we may be able to create more complex systems.

The project we chose may look complex (visually), but it is actually quite simple; but it presents the very core of how most systems work nowadays. With the 7 segment LED display, you may not only display patterns of numbers, but of letters and other symbols as well. With the logic we applied to turning on the LEDs of the green, yellow and red light, you may be able to implement a 24 x 6 LED matrix to display whatever you want.

The application of digital systems are never ending; from displays, to traffic control, from sensors, to medical equipment, circuitry has been an important factor for the advancement of technology, and most importantly in our everyday lives. Given an idea, the right amount of data, components, proper implementation and application, anyone can make any system. And though not everyone outside from this field may become interested in this topic, it would not hurt to learn, even a little, to understand how most technological systems work.

Even outside the field of circuitry, logic design can be applied in mathematics, programming, and even artificial intellience, but that’s for another topic to discuss.

1. **References**

The J-K flip-flop. (n.d.). Retrieved August 18, 2014, from <http://www.allaboutcircuits.com/vol_4/chpt_10/6.html>

Thomas, T. (2012, November 12). Function Generator Working with Block Diagram. Retrieved August 18, 2014, from <http://www.mycircuits9.com/2012/11/function-generator-working-with-block.html>

7-segment Display and Driving a 7-segment Display. (n.d.). Retrieved August 18, 2014, from <http://www.electronics-tutorials.ws/blog/7-segment-display-tutorial.html>

Mano, M., & Ciletti, M. (n.d.). Digital Design, 5th Edition by M. Morris Mano and Michael Ciletti.pdf - Google Drive. Retrieved August 18, 2014, from <https://docs.google.com/file/d/0B9LJy8vattSMeWxOMDl1Sk43Sjg/edit?pli=1>