

**EE457: Digital IC Design**

**Fall Semester 2016**

**Final Project Report Cover Sheet**

**Due 12/20/16, 6:00PM**

Enigma Game

**PROJECT TITLE: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Murtaza Yaqoob, Steven Eisinger

Student Names:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Enigma

Group Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Five points will be deducted for not following the directions***

|  |  |  |
| --- | --- | --- |
| Check for completion | Topics | GRADES |
|  | Executive Summary | /5 |
|  | Introduction/Background | /5 |
|  | Circuit Schematics/Symbols | /10 |
|  | Working Electric Layouts | /20 |
|  | IRSIM Logic Simulations | /10 |
|  | Measurements of power/delay/chip area/timing/number of transistors. | /10 |
|  | LTSPICE code/extractions | /10 |
|  | Measurements in LTSPICE | /15 |
|  | Conclusion and References | /5 |
|  | Presentation | …………/10 |
|  | TOTAL | **/100** |

X

We want the chip fabricated and test it through Independent Study. YES\_\_\_\_, NO\_\_\_\_\_

Note: Label your figures and tables. Reference them in your text and discuss your results.

Executive Summary

The name of our project is inspired by the Enigma machine, a cypher machine used by the Germans in World War II. The Enigma machine worked by mapping different letters to each other to produce an encrypted message, which could be sent over the air to another location where it would be decrypted into the original message. This prevented allied forces from being able to understand German communications, unless they could correctly guess the encryption ‘key’ for that day which would correctly decrypt the message. In light of the Enigma machine, our ‘Enigma Game’ requires two players, one to input a ‘key’, and another who must guess it. The rules of the game are as follows:

1. Player 1 and Player 2 agree on a turn limit, which will be the amount of opportunities Player 2 gets to make a ‘guess’.
2. Player 1 inputs a ‘key’ into the designated Player 1 location, consisting of 4 unique extended ASCII or UTF-8 characters (both of which can be represented with 8-bits).
3. Player 2 inputs a guess as to what the key is (which has the same limitations and encoding as the key) into the designated Player 2 location.
4. The ‘Enigma Game’ chip lets Player 2 know whether the characters they guessed exist somewhere in the key and if the character is in the correct location.
5. Player 2 continues to guess and receive a result from the chip until they guess the correct key (meaning all characters in the guess exist in the key and are in the correct place, as determined by the Enigma Game chip), resulting in a win, or the predetermined turn limit is reached, resulting in a loss.

The game chip must only provide two outputs for each character in the guess: the ‘match’ bit which denotes that the character matches one of the characters in the key, and the ‘placement’ bit which denotes that the character is in the correct place. If we limit the key to four characters, the chip can be constructed with a total of eight 8-bit inputs (four for the key and four for the guess), sixteen special comparators which can check for a character match, and a 4-bit OR gate to check if there is a match anywhere in the key.

Physical interaction with the game chip would be possible by connecting a keyboard to a decoder which would act as an interface for the character inputs, and lightbulbs or LEDs could be connected to the placement and match outputs for each character so that they are understandable by the players. In order to simulate interactivity with the game chip, we will use the Python scripting language to interface with the Spice model of the device.

Application-Specific Integrated Circuits (ASIC) such as this one historically have been used in cars and PDAs. Due to the complexity of video games and toys and the specialized nature of ASICs, it’s easier and cheaper for companies to program generic EEPROM chips to run electronic toys, or CPUs to run modern video games; this limits the use of a chip like this in modernity.

Introduction

The Enigma Game is basically a password guessing game that lets you know if your entry is correct. As with a password, the order in which you type in a character matters. For simplicity, we will assume that all characters in a **KEY** are unique (meaning no repeats). Constructing a game such as this seems daunting, but is very doable if you split it up into smaller parts. There are many references to inputs and figures in this report, so for readability, all input names will be in **BOLD** and figures will be referenced in *Italics*.

There are 8 8-bit input ‘busses’, one for each character of both the **KEY** and the **GUESS**,because an extended ASCII or UTF-8 encoded character is represented with 8 bits. We need to know two things for each **GUESS** character: does this character match the corresponding character in the **KEY**, and if not, does it match any other character in the **KEY**? We’ll call these 2 outputs **PLACE** (for when the **GUESS** character matches a **KEY** character in the correct location) and **MATCH** (for when the **GUESS** character matches a **KEY** character in a different location). We’ll say we want **PLACE** to be **HIGH** when the **KEY** and **GUESS** in the same position matches, and **LOW** otherwise. Similarly, we’ll say **MATCH** should be **HIGH** when the **GUESS** character matches a **KEY** character in a different location. This is illustrated in *Figure 1*.

|  |  |
| --- | --- |
| Condition | Output |
| **GUESS A == KEY A** | **PLACE = HIGH** |
| **GUESS A != KEY A** | **PLACE = LOW** |
| **GUESS A == KEY B, C,** or **D** | **MATCH = HIGH** |
| **GUESS A != KEY B, C,** or **D** | **MATCH = LOW** |

**C**

**D**

**B**

**A**

**KEY**

check if equal

check if equal

check if equal

check if equal

**A**

**GUESS**

*Figure 1. Checking equality of* ***KEY*** *and* ***GUESS*** *characters.*

The diagram on the left of *Figure 1* shows how we intend to compare each character and what we expect in our outputs based in the results. Note *Figure 1* uses C-like notation for comparison and assignments, which will be present throughout this article. This type of comparison is done for each of the **GUESS** characters (**GUESS A, GUESS B, GUESS C,** and **GUESS D**).

We can represent **PLACE** and **MATCH** as switching functions using logic gates. Each **GUESS** character will have its own **PLACE** and **MATCH** bit, meaning we will have 4 of each.

Design and Implementation

Part 1: Blah

Blah

Part 2: Blah

Blah

Part 3: Blah

Blah

Conclusion

Blah

References

Blah