**Caesars Cipher Design Report**

|  |  |
| --- | --- |
| **Topic:** | *Topic 8* |
| **Date:** | *April 28th, 2023* |
| **Revision:** | *1.3* |
| **Milestone Summary:** | |  |  |  | | --- | --- | --- | | **User Story / Task** | **Hours Worked** | **Hours Remaining** | | *User can enter a phrase and receive the encoded phrase* | *7* | *0* | | *Use the switches to set different key values* | *3* | *0* | | *Enter an encoded phrase, and with the right key, get the correct message decoded* | *6* | *0* | | *Display the key being used to the 7-segment display* | *3* | *0* | | *Connect the C program to GPIO pins* | *5* | *0* | | *Create the circuit in VHDL* | *2* | *0* | | *Port the circuits to Verilog* | *2* | *0* | | *Create circuits using a behavioral model* | *3* | *0* | |  |  |  | |  |  |  | |  |  |  | |  |  |  | |  |  |  | |
| **GIT URL:** | *https://github.com/hydrenoid/CaesarsCipher.git* |

**Design Documentation**

**General Technical Approach:**

*The Caesars Cipher algorithm is very simple, it takes in a phrase and encodes it using a certain key, and returns an encoded version of the phrase. Then to get the correct decoded phrase you must enter the encoded phrase and the key that was used to encode it and you will get your original phrase back. It does this by taking each character in the phrase and mapping it to an integer value and, depending on the key, add a certain number to it mapping it to a different character, to get the original character back you simply subtract the number specified by the key. In this project the user will enter their phrase into the terminal running on the ARM processor, choosing whether to encode or decode. Then they will set the key by using the first four switches, the key will be displayed on the first two 7 segment displays (using a BCD decoder to display the correct numbers). Then the program will map each character in the phrase to a number, turn that number to binary, and pass the number to the FPGA where the Caesars Cipher algorithm (which will just be a parallel adder/subtractor) using 8 GPIO pins, as well as another to select whether encoding or decoding. Then the program will read the output pins from the FPGA to get the encoded value, map the new number to its new character, and build the new phrase.*

**System Design:**

*Diagram

Description automatically generated*

**Application Design:**

*Diagram

Description automatically generated*

**Digital Logic Design:**

*7-segment Decoder*

*Truth Table:*

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Decimal Digit | A | B | C | D |  | a | b | c | d | e | f | g |
| 0 | 0 | 0 | 0 | 0 |  | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |  | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 |  | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 3 | 0 | 0 | 1 | 1 |  | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 4 | 0 | 1 | 0 | 0 |  | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 |  | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 6 | 0 | 1 | 1 | 0 |  | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 7 | 0 | 1 | 1 | 1 |  | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 8 | 1 | 0 | 0 | 0 |  | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 9 | 1 | 0 | 0 | 1 |  | 1 | 1 | 1 | 1 | 0 | 1 | 1 |

*Boolean Expressions:*

*a = A + C + BD + ~B~D*

*b = ~B + ~C~D + CD*

*c = B + ~C + D*

*d = ~B~D + C~D + B~CD + ~BC + A*

*e = ~B~D + C~D*

*f = A + ~C~D + B~C + B~D*

*g = ~BC + C~D + B~C + A*

*Caesars Cipher*

*Final Circuit Diagrams:*

*Diagram, engineering drawing

Description automatically generated*

*Diagram

Description automatically generated*

**VLSI Design:**

*Structural VHDL Code:*

*Text

Description automatically generated*

*Behavioral VHDL Code:*

*Text

Description automatically generated*

*Structural Verilog Code:*

*Table

Description automatically generated with low confidence*

*Behavioral Verilog Code:*

*Text

Description automatically generated*

*BCD 7-Segment decoder VHDL Code:*

*Table

Description automatically generated with medium confidence*

*Computer Image Cipher Circuit:*

*Graphical user interface, text, application

Description automatically generated*

*Code added to end of computer image:*

**

*Schematic:*

*See Repository.*

**Risks and Issues:**

*Some issues that I have already delt with were 1) getting the application to set the GPIO pins with the correct binary number as many of the binary numbers were themselves smaller than 6 bits, so I had to create an algorithm to create a binary array in the correct order and 2) deciding how many pins I was going to need depending on the mapping values I was using. Originally I wanted to do ascii values and I might move back to that but I wanted to limit the number of pins that I was using to six for now so I had to create my own map for the characters and functions to map them. For risks in the future it will mostly be making sure that the circuit diagrams for the Cipher are built correctly and making sure that there are no instances where it fails. Along with that the biggest step will be finding a way to easily manipulate the GPIO pins inside of the register as that will be the majority of what the program does reading and writing to that register.*

*After some talking with Professor Mark, I have decided to switch to using the ascii table. To do this I have extended the circuit to use 8 bits input and to have 8 bits for the output to hold the larger numbers. On top of that when the phrase is entered, all of the characters will be sent to their lower case version to make sure not to go over the top of the ascii table.*

*As progress has continued the largest obstacle as of right now will be connecting the C program running on the ARM processor to access the pins from the circuits. As of right now all of the circuits are completed and tested, the program is simulating correctly and the only step left is to connect to those pins.*

*As of the final release the project has been completed in full, the hardest and most confusing section was connecting my circuits to the computer image provided with the board, the issue was to set the GPIO pins your output needs to be in a tristate, to do this I had to change the output from reg to trireg, and also add an input called enable (connected to a switch on the board) that when turned on will connect the cipher circuit to the GPIO pins. Without this the circuit will not compile and the circuit will now work if the enable pin is not set to on.*

**Other Documentation:**

*Pin Assignments:*

*Table

Description automatically generated*