final report….word > pdf

VIDEO with demos

PUT both names

1. The description of the project

2. Background theory

3. Schematics, diagrams, etc

4. Circuit operation

5. Simulation results with annotations.

6. Encountered problems and how they were solved

7. Any information regarding the project, that might be interesting

8. Conclusions

9. video?

X. Working zip file of your final project. Your project should run in Quartus when unzipped

without any errors/missing files. (quarus .v for encryptor and decryptor, quatus symbol for both also)

**1. description**

This is 1 of 2 circuits.

It is designed to input 200 unencrypted 16-bit words and output 200 16-bit words that encrypt the original input.

In each word, the input represents a 7-bit ASCII character that the sender wants to transmit, starting with the LSB of the 16-bit word. The remaining bits are 0.

The 16-bit word is encrypted on each rising clock edge.

For encryption, the 16-bit word just adds { 2^15 + 1957 } = \*\*34725\*\* to the input, producing an output. In binary, the key is 16'b 1000 0111 1010 0101

for example, 'A' input = ASCII 65 (decimal), so input is [ 0000 0000 0100 0001 ]

and the input is encrypted to (decimal) 34725 + 65 = 34790, so the output is [ 1000 0111 1110 0110 ]

the process continues until all 200 words have been encrypted

if the message is less than 200 words, the NULL character will be encrypted

The test bench stimulates this circuit by acting as the buffer for an arbitrary plain text input stream.

It is what the sender used to input his message.

Using these 200 unencrypted 16-bit words as stimuli, the input is changed between clock edges.

As the stimuli provide input to the DUT and output is recorded by the test bench, this represents the point at which the receiver gets the encrypted message.

Since the transmission was encrypted, the received data will not be coherent unless the receiver has the proper KEY.

See "20200429 decryptor #UNLV" here: https://www.edaplayground.com/x/2j7c for the circuit 2 of 2 that is able receive and decrypt a message from this circuit.

\* The sender and receiver (both parties) should each have an encryptor and decryptor circuit so they can both transmit and receive.

\* The encryption method is customizable to virtually anything, but both parties must use the same encrypt and decrypt method.

This is 2 of 2 circuits.

It is designed to input 200 encrypted 16-bit words and output 200 decrypted 16-bit words, revealing the message from the sender.

In each word, the received input represents a character.

Since the encrypted input was an unencrypted ASCII character with a key of { 2^15 + 1957 } = \*\*34725\*\* added to it:

input [ 1000 0111 1110 0110 ] is decrypted by subtracting decimal 34725 or binary 16'b 1000 0111 1010 0101

this produces the output of [ 0000 0000 0100 0001 ], revealing the character 'A'

this process continues until all the characters have been decrypted

The test bench begins by stimulating this circuit with 200 encrypted 16-bit words that would have come from the sending circuit.

As the stimuli produce output, the receiver can easily convert the binary ASCII numbers to characters and read the message.

See "20200429 encryptor #UNLV" here: https://www.edaplayground.com/x/3hWW for the circuit 1 of 2 that is able to encrypt and transmit a message to this circuit

\* The sender and receiver (both parties) should each have an encryptor and decryptor circuit so they can both transmit and receive

\* The encryption method is customizable to virtually anything, but both parties must use the same encrypt and decrypt method

\* This code is synthesizable at the RTL level and can be targeted to ASICs or FPGAs

\* Icarus Verilog 0.9.6, YOSIS ABC produces map and netlist/cell library

\* Logic synthesis on front-end and place/route on back-end were successful

This project works as follows:

1) the "transmitter" inputs a message up to 200 characters.

here, just change "original.txt"

in verilog, I will assume the inputs as 7-bit ASCII > [16:0] or use some I/O features

see the ASCII conversion in "TOasciiRaw.txt"

2) the message is transferred from a [200][7] array to a [200][16] array

see "TOasciiEXP.txt"

3) the unencrypted [200][16] array from step 2 is encrypted to a new [200][16] array

here, the array is encrypted by making the MSB of each row = 1 and the year UNLV was founded ( 2^15 + 1957 )

see "TOcrypt.txt"

4) at some point, the [200][16] encrypted array is transmitted to the "receiver"

the receiver hopefully has the circuitry to unlock the message

in this case, the receiver just needs a circuit to group x200 16-bit binary numbers and process them

all the receiver has to do is:

- get the decimal digit of the 16-bit number (or just use binary subtraction)

- subtract KEY from the decimal number

- find what ASCII character the result corresponds to

- output the result

- read the message

if "TOdecrypt.txt" matches "original.txt" then the message was successfully transmitted

in practice, the message can be transmitted multiple times and the receiver can see if all transmission are the same

in real life, the receiver will not have access to the original message

5) The verilog test bench will serve as the regular module stimulator that it is

for simplicity, a section of the test bench will serve as the input

also, the test bench will fulfill the decryption and output

the main module will have similar roles since it is stimulated by the test bench

if applied, the transmitter and receiver would have their own circuits

driving the cycle is a FSM

The encryption scheme is very weak for simplicity, and because we are short on time.

It can be anything you want as long as there is an algorithm or look up table to decrypt.

If decryption can't be done, then it defeats the purpose.

If someone can decrypt the message, then that also defeats the purpose.

But this scheme can be expanded or modified in any way.

This started to get very involved as soon as work began. There are some big numbers to work with.

Implementing on a circuit without any I/O or ability to demonstrate (during virus outbreak of 2020) makes it all theoretical.

C++ was a good way to get the idea in writing and provides a some what easy transition into verilog.

At least you can use some of the existing functions to generate some of the 1000's of inputs you will need.

The array index of C++ being reversed from the usual MSB...LSB can be negated if big endiness is always assumed.

That is what I used here, and what will be used in Verilog.

**2. Background and theory**

circuits involved

how they operate

control methods

I/O

**3. Schematics, diagrams, ect**

anything pictures or graphs

\* encrytor input (raw)

\* encryptor output(encrypted)

\* decrytor input is same as encryptor output, just note it

\* decrytpor output with verification

\* SM chart, both circuits

\* State graph if you want w/ state table

\*quarus and waves

\*quatus symbol files x2 (code screen shot if you want

**4. Circuit operation**

describe the verilog

INPUT

OUTPUT

state changes

abort stops all

2 circuits

memory

clock cycles

internal variables

**5. Simulation results with annotations.**

**6. Encountered problems and how they were solved**

ASCII input and output is hard to realize and very tedious to obtain manually. To overcome this obstacle and not have to manually code thousands of entries, we used simple programs to generate the segments of code that we needed.

The computer program also helped guide the design, get a working model rapidly, and verify circuit output.

**7. Any information regarding the project, that might be interesting**

Working with large amounts of data is very resource intensive

the combination possibilities increase exponentially when the data size increases linearly

Uses unlv founding date of 1957 to encrypt device

**8. Conclusions**

As lab partners throughout the semester, completed all 12 labs together. The first few labs were very basic and we did not know much about sequential circuits. The basics are important in logic design because each and every topic is interrelated. Circuitry has to be properly arranged to function as desired. It is notable that the designer needs to have skills with both combinational and sequential circuits. With this project, we were able to apply the majority of principles that we learned throughout the semester. Working with encryption forced us to handle much larger data sets, but we found that if you can handle 2 in a row, then 200 is the same thing. …..