Image Steganography with FPGA

Aithu Snehith, Nakka Chakradhar

Introduction to FPGA

A field-programmable gate array (FPGA) is an integrated circuit designed to be configured by a customer or a designer after manufacturing hence the term "field-programmable". The FPGA configuration is generally specified using a hardware description language (HDL)
Generally there are licensed tool-chains to program and work with FPGAs. Here we are exploring icoprog - an opensource FPGA bitstream convertor for FPGAs.

Introduction to Steganography

Steganography is the practice of concealing a file(any information) within another file.

Project Scope

This project is useful for hiding information in any file. The scope of the project is implementing steganography on FPGAs for hiding information which includes any type of files like image files, audio files, video, apk etc., in any other file.

For this project we're hiding a message string in an image file.

Methodology - LSB insertion

The LSB is the lowest significant bit in the byte value of the character.

The LSB based image steganography hides the secret(byte) in the least significant bits of pixel values of the cover image.

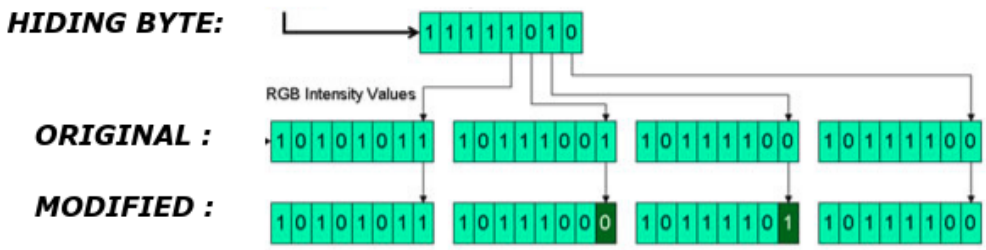

Figure: Overview of LSB Insertion

Rundown on Algorithm

Here is a quick rundown on the algorithm we're implementing here

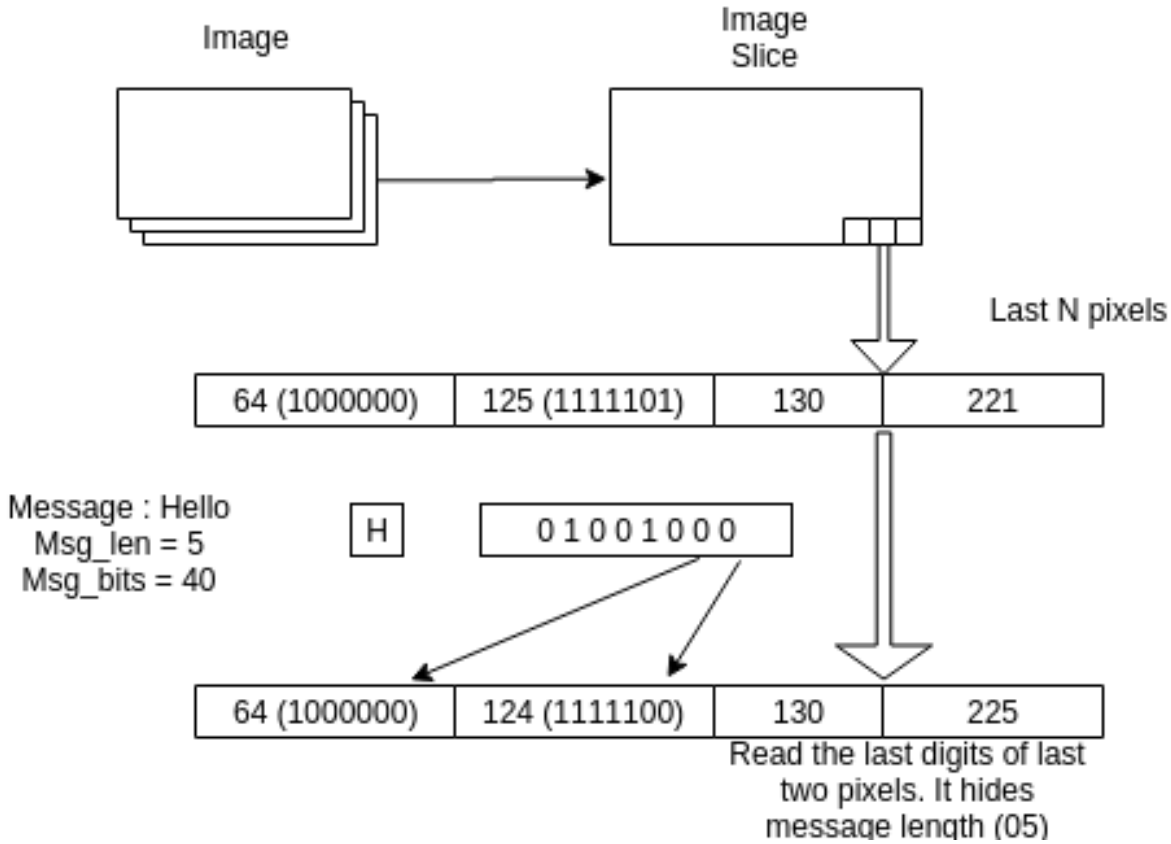

Figure: Overview of LSB Insertion

Hardware Connections

A quick look on the connections we're making between Raspi, FTDI and IcoBoard

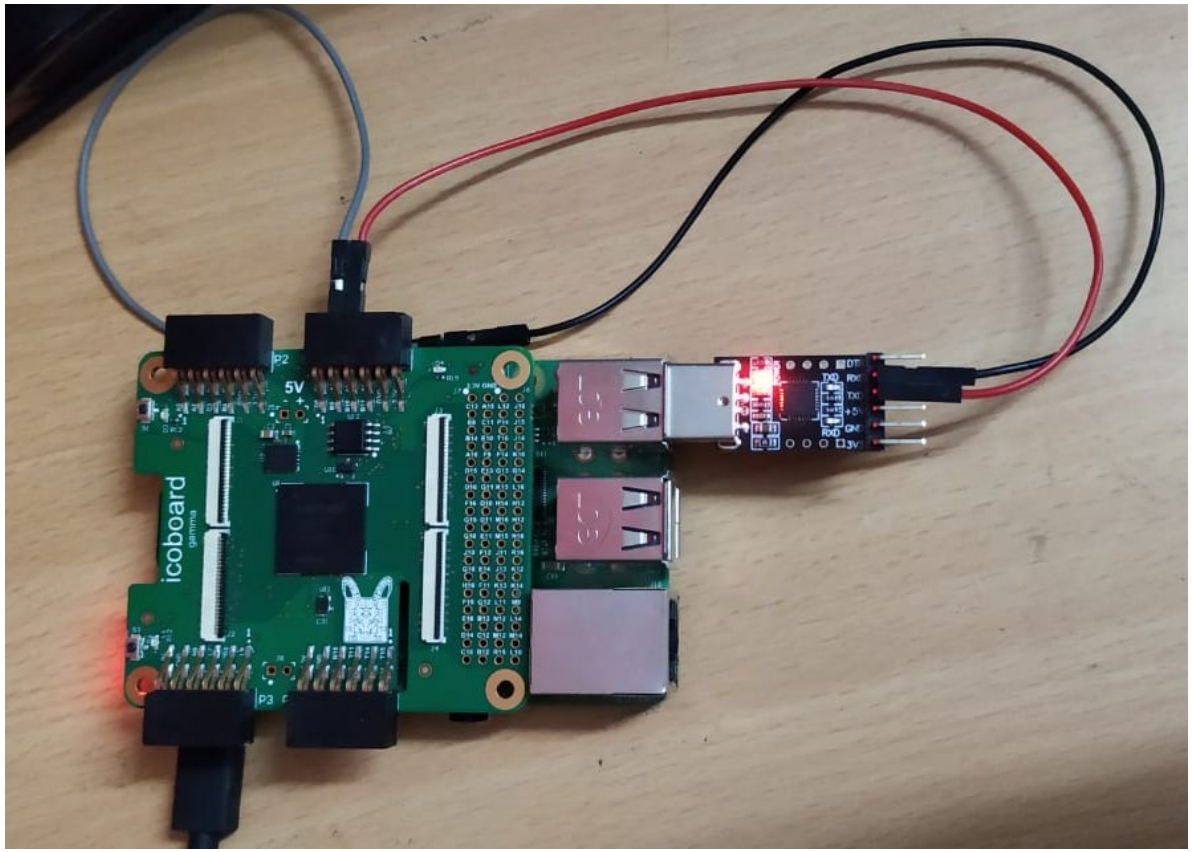

Figure: Overview of LSB Insertion

Installation of open-source tools

We would require 4 tools primarily to work with FPGAs

- Icestorm
- Arachne-PNR
- Yosys
- Icoprog

These are all available for Linus systems. We'd be primarily dealing with Ubuntu based systems

Installation of Dependencies

For the above 4 tools to work, first we need to install the following dependencies

- sudo apt install build-essential clang bison flex
- sudo apt install libreadline-dev gawk tcl-dev libffi-dev
- sudo apt install git mercurial graphviz xdot pkg-config
- sudo apt install python python3 libftdi-dev qt5-default
- sudo apt install python3-dev libboost-all-dev cmake

run the command as it is on a terminal session

Icestorm

- git clone https://github.com/cliffordwolf/icestorm.git icestorm
- cd icestorm
- make -j$(nproc)
- sudo make install

Arachne-PNR

- git clone https://github.com/cseed/arachne-pnr.git arachne-pnr
- cd arachne-pnr
- make -j$(nproc)
- sudo make install

Yosys

- git clone https://github.com/cliffordwolf/yosys.git yosys
- cd yosys
- make -j$(nproc)
- sudo make install

Icoprog Installation

These steps must be done on Raspi as these are to burn the .bin to IcoBoard

- cd $HOME
- git clone git://git.drogon.net/wiringPi
- cd wiringPi && ./build

- cd $HOME
- sudo apt-get install subversion
- svn co http://svn.clifford.at/handicraft/2015/icoprog
- cd icoprog && make install

UART in Verilog

We are deploying two verilog modules for this purpose

- UART protocol to send out the encoded image
- Module to hide any given text in the pixel matrix of the image

Contents of each file

Lets dive into the code for better understanding

- v_gen.py - generates the required .v file by accepting image and message string as inputs
- steg.pcf - a pin configuration file. We can change what pins act Rx and Tx for the UART
- regen_img.py - receives input from IcoBoard and spits out image mod.png with string hidden inside
- decrypt.py - decrypts and reconstruct the hidden message from the image mod.png

How to execute - Laptop

Here are the steps for execution

- run the command python v_gen.py. The command asks you to enter message and image name. Enter accordingly
- run *make* in terminal to generate verilog file. It is suggested to do it on a laptop or system as the compilation takes a lot of time on Raspberry pi.
- If run on a laptop, copy the files onto Raspberry Pi by some sorts.
- If Raspi is on a local network, copy the files from laptop with the following command - *scp steg.bin* pi@ip_addr:*path*
- run *icoprog* $-$ *p* $<$ *steg.bin* on Raspi to flash the bin file
- Run regen_img.py on Raspi to receive and save the encrypted image
- Run decrypt.py to decrypt and display the message

How to execute - Raspberry

Here are the steps for execution

- run the command python v_gen.py. The command asks you to enter message and image name. Enter accordingly
- run *make* in terminal to generate verilog file
- run $icoprog - p < steg.bin$ on Raspi to flash the bin file
- Run regen_img.py on Raspi to receive and save the encrypted image
- Run decrypt.py to decrypt and display the message

TL:DR ? Shorter way of execution

Here are the steps for execution

- Run sh build.sh on Laptop to build the bin file and push it to raspberry pi on the local network
- Run execute_nd_decrypt.sh to decrypt and display the message

Changes After LSB Insertion

Comparing two images before and after hiding text shows that there is no significant visual changes in the image

Original Image

Image with text hidden

Figure: Comaprision after LSB Insertion

Alternative Method

Although this is not a viable approach, another alternative way to solve this problem is to deploy a PicoRV32 as an SoC onto IcoBoard and do the computations. Once PicoRV32 is

implimented, we can run RISC-V GCC compatible C codes. So instead of generating Verilog files with image matrices, we generate C codes and compile and flash onto icoboard. The compilation takes time so it's adviced to run it on a Laptop

Alternative Method

Another advantage to this method is once the processing is done, PicoRV32 has comm modules pre-defined which we can use by configuring PMOD pins. One notable example is UART. Not just

that but once PicoRV32 is implemented, Raspi assumes access to one of the registers on the IcoBoard which enables console output. So we can use printf statements in the above mentioned C code for debugging or Raw communication purposes.

References

- ZipCPUs implementation of UART - https://github.com/ZipCPU/icozip/tree/master/rtl
- CliffordWolf Icotools - https://github.com/cliffordwolf/icotools
- Pramode's Blog posts on working with IcoBoards and UART - http://pramode.in/2016/10/05/fpga-programming-with-foss-tools/
- Google search for FTDI functionality