Design Plan Draft - Lycan

Introduction

Purpose

The Lycan device is a universal hardware interface meant for debugging and reverse-engineering ICs with common digital interfaces such as UART, SPI, and JTAG. This project is aimed at people interested in hardware reverse engineering, professional engineers working on embedded systems, and students learning computer engineering. Currently, if any of these groups are trying to work with hardware interfaces like UARTs, they need to purchase separate interfaces to work with each protocol. The Lycan device will enable users to have a single USB device that is able to reconfigure into any set of digital communication interfaces required for their device.

Domain and Prior Art

While there are many devices that allow for reading and writing different digital interfaces, there are none that are reconfigurable to allow interfaces to be mapped to different pins while being able to communicate with said interfaces at high speeds. Additionally, our device will have a scripting API using the industry-standard Python scripting language.

Digilent Analog Discovery [1]

- Allows both reading and writing over its digital GPIO pins.
- Protocol analysis is done in software by parsing the raw waveforms
- Scriptable only through an embedded Javascript runtime with a poorly documented API.

Bus Pirate [2]

- Able to reconfigure its pins to read and write different protocols, but the protocols are bit-banged by an MCU and limited speed (<1 MHz in many cases).
- Designed to be used programmatically through a VT100 terminal interface with custom commands rather than through a standard scripting language.
- Limited number of I/O pins - only 8 are available on the Bus Pirate while Lycan aims to have at least 16 I/Os usable by peripherals.

Impact & Risk Assessment

We believe that Lycan will be beneficial to society by reducing the need for many separate hardware interfaces when doing embedded development. This will reduce the potential waste produced by manufacturing those single-purpose interfaces and lower the barrier to entry for embedded development. The scripting capabilities will also allow reverse engineers to automate the initial discovery process, enabling them to understand unknown devices quicker. As with all security research, there is a discussion to be had whether it is advantageous to make hardware reverse engineering more efficient and accessible, since any benefit that is used for white-hat research can be used maliciously as well.

Statement of Work

The goal of the project is to create a Lycan device that supports running up to 8 communication peripheral instances at a time that can be reconfigured at runtime to connect to any pin of the device. Lycan will run at a clock speed of 100 MHz to use the maximum bandwidth of a 5 Gbps USB link. Additionally, we will design a Python API for interfacing with Lycan and a GUI to expose the functionality of the API in an intuitive format. Finally, we will create a custom PCB for Lycan that will integrate voltage translation to allow Lycan to interface with DUT logic levels from 1.2V-5V and analog measurement to enable automatic detection of whether pins are inputs or outputs.

Details on the specifics of each task are documented in our Github repository in planning.md

Task	Description	Team Member(s)	Deadline
USB FIFO Protocol Manager	Circuit that abstracts read/write requests from the USB bus into control signals for the FT601	Gilon	Milestone 2
Bus Arbitrator	Circuit to decide which of 8 peripherals can send/receive data over USB at any given time.	Adam	Milestone 2
Generic Peripheral Wrapper	Common functionality for every peripheral, including I/O multiplexing and TX/RX FIFOs	Matthew H.	Milestone 2
Partial Reconfiguration Flow	Set up partial reconfiguration for our project (software process and	Matthew H.	Milestone 3 & 4
UART Peripheral		Andres	Milestone 3 & 4

SPI Peripheral		Andres	Next Semester
JTAG Peripheral		Adam, Gilon, Andres	Next Semester
I2C Peripheral		Adam, Gilon, Andres	Next Semester
API - Receive Functionality	Receive data from peripherals using Python	Matthew S.	Milestone 2
API - Transmit Functionality	Transmit data through peripherals using Python	Matthew S.	Milestone 2
API - Configuration	Reconfigure Lycan's peripherals using Python	Matthew S.	Next Semester
GUI Application	Expose API functionality through a simple GUI	Matthew S.	Milestone 4
Custom PCB	Integrate all circuitry needed for Lycan into a single board	Matthew H., Matthew S.	Depends on progress of main Lycan RTL - Next Semester

Milestones

1. Design Revision - Week 8	● Design Plan Draft ● Researching protocols and drivers
2. Pre-Alpha Build - Week 10-11	● Design major hardware blocks (peripheral wrapper, USB interface, bus arbitration) ● Initial API working with FTDI (TX/RX)
3. Design Prototype - Week 13-14	● Integrate and verify hardware architecture built for pre-alpha ● Start implementing partial reconfiguration ● Communicate with PC through an API to configure Lycan ● UART peripheral mostly functional
4. Prototype Presentation - Week 14-15	● UART peripheral fully verified and integrated ● Continue integrating partial

	reconfiguration ● Work on simple GUI prototype

Deliverable Artifacts

We are planning to distribute the following items through our Github repository under an Open Source license. This will enable us to easily make software revisions and accept contributions from any Lycan users who want to improve the project. Our plan is to create a finalized “1.0” package by the end of the project containing hardware and software designs which can be used indefinitely into the future as a complete tool.

Hardware

- **Lycan -** RTL source code written in SystemVerilog that encompasses the sampling and protocol decoding portions of the projects. Can be used on an Artix-7 development board with a FTDI FT601 USB to FIFO IC development board.
- **Custom PCB** - A custom PCB will be created to replace the Xilinx development board, and will integrate all necessary components (FPGA, FTDI USB interface IC, I/O pins, etc.). This board will be designed with high speed signals (100+ MHz) in mind. To improve upon the development boards, our custom PCB will add voltage translation and analog measurement circuitry.

Software

- **Lycan API -** A Python API that communicates with Lycan hardware to configure peripherals and transmit/receive data from the peripherals. This API will be distributed as a Python package for users to install through pip.

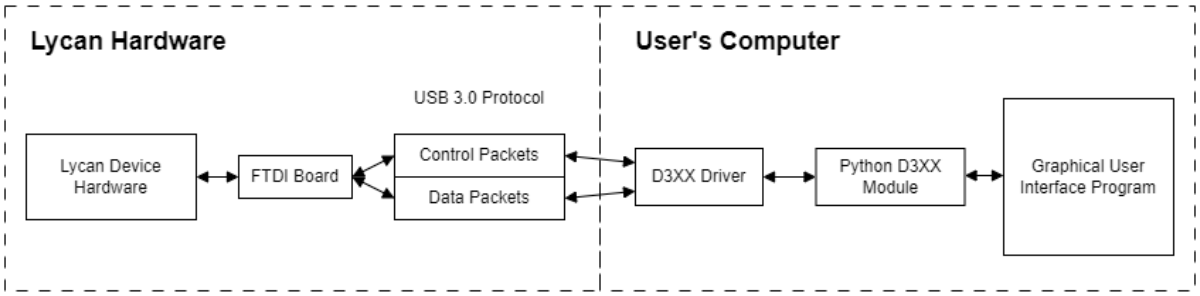
- **Lycan GUI** - A GUI application that exposes the functionality of the Lycan API in a more approachable format, such as choosing which pins should be configured as a specific peripheral and viewing the data that has been transmitted/received by a specific peripheral. This will likely be distributed as a script or portable executable from our repository's Releases page.

Documentation

- **User Manual -** A document compiling the necessary information for end users on how to operate Lycan. Information such as hardware specifications, user setup, and best practices will appear. An API reference will also be included in the manual. The user manual will be formatted into a PDF file that can be easily distributed online on our repository.

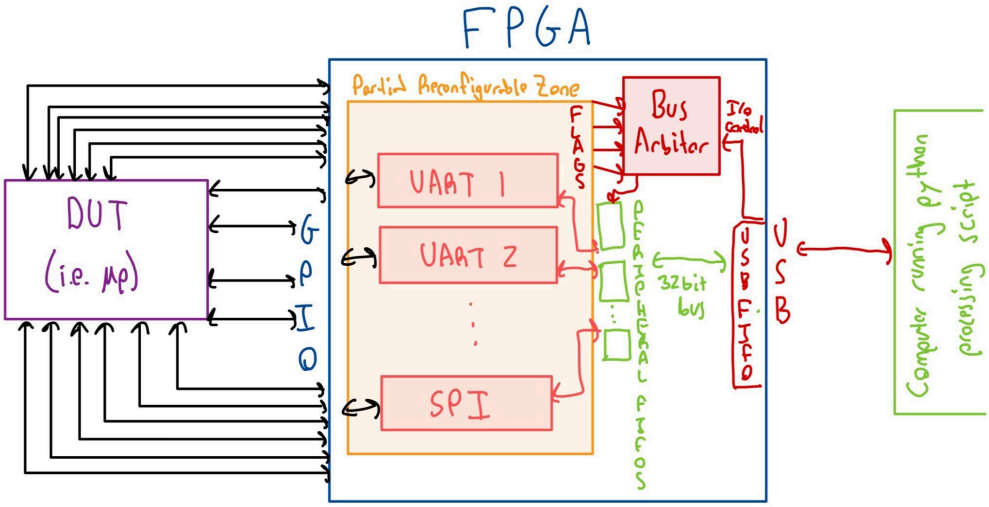
Mockups

Interfaces



The interface between hardware and software will primarily occur over a USB bus that connects Lycan's FTDI FT601 USB-to-FIFO interface IC to a host computer. In order to allow the Lycan API and GUI to speak with the Lycan, we will utilize the D3XX driver [3]. This driver is supplied by FTDI for its SuperSpeed USB 3.0 ICs. The driver has a Python wrapper module, called ftd3xx [4], which will be used to easily integrate the USB communication into the software application.

Systems



The above schematic depicts a high-level systems overview of Lycan. Key features include a “black-box” design under test. This is connected to the FPGA, which contains sections for partial reconfiguration that can be set to instantiate different communication peripherals at runtime. The data transmitted and received by the peripherals are buffered with FIFOs so that we do not have to constantly stream data from every peripheral. A bus arbiter circuit decides which peripheral is allowed to send its data back to the host computer over USB at a given time. The data from peripherals is interpreted by a Python script running on the host computer.

Networking

We are utilizing USB for communication between the host computer and Lycan. The data bus of the FTDI USB interface is 32 bits wide, with separate FIFOs for packets being received and transmitted. We have designed a simple protocol for communicating with Lycan that will utilize 32 bit packets sent over the USB bus. There are three types of packets to consider: peripheral data from Lycan to the host, peripheral data from the host to Lycan, and configuration data from the host to Lycan.

Data from Host to Lycan (Transmit Data)

Name	Bit Field	Notes
Peripheral Address	31-29	Supports up to 8 peripherals
Configuration Flag	28	0 = data to transmit
Number of Valid Bytes	27-26	1-3 bytes in the “transmit data” field of this packet
Reserved	25-24	
Data to Transmit	23-0	

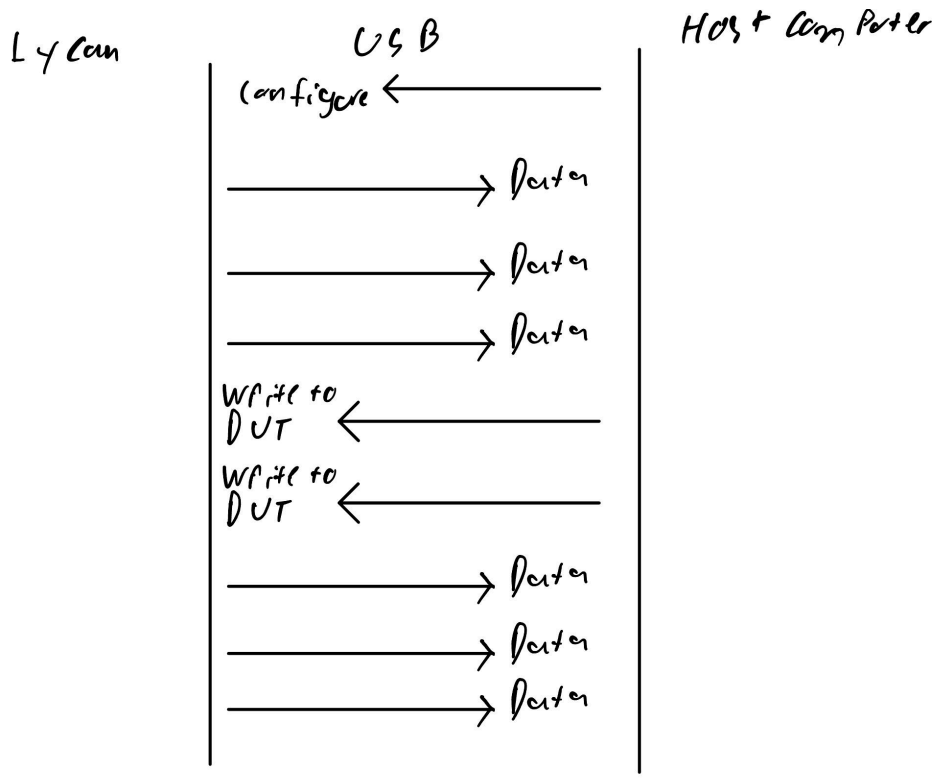
Data from Lycan to Host (Receive Data)

Name	Bit Field	Notes
Peripheral Address	31-29	Supports up to 8 peripherals
Number of Valid Bytes	28-27	1-3 bytes in the “receive data” field of this packet
Reserved	26-24	
Received Data	23-0	

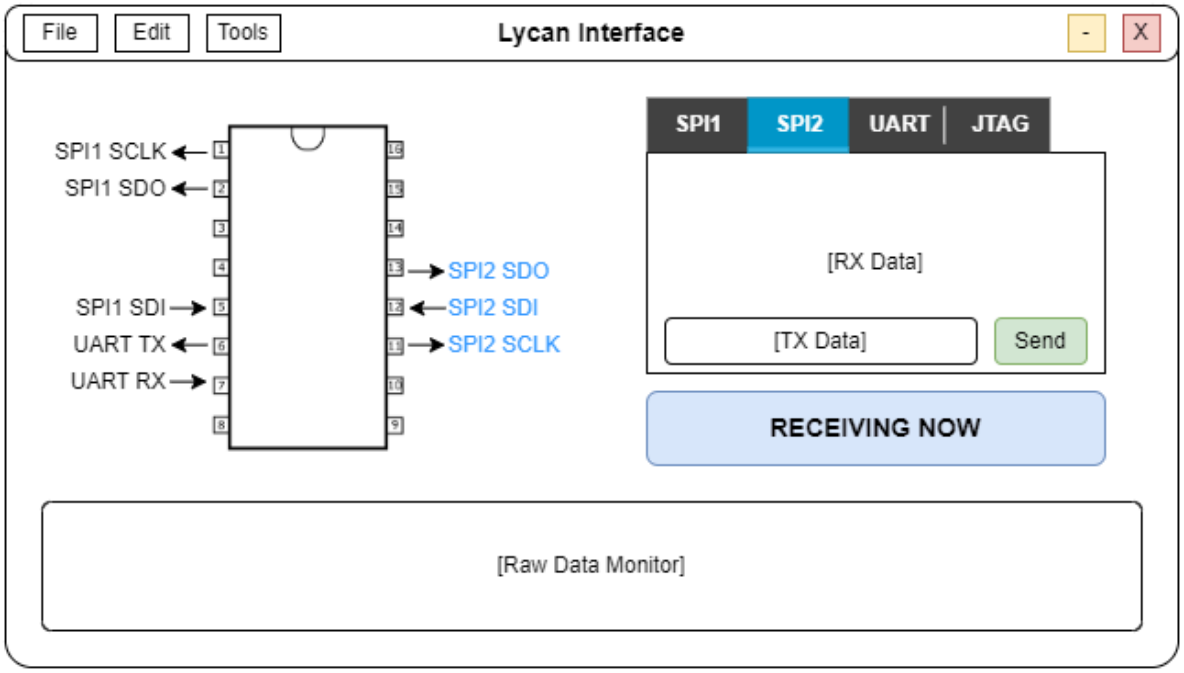
Configuration Command from Host to Lycan

Name	Bit Field	Notes
Peripheral Address	31-29	Supports up to 8 peripherals
Configuration Flag	28	1 = configuration command
Read/Write	27	Read/write values from a configuration register
Configuration Register Address	26-24	Peripheral dependent configuration register address
New value of register	23-0	Ignored for reads. The bottom bits of this field are used for registers smaller than 24 bits.

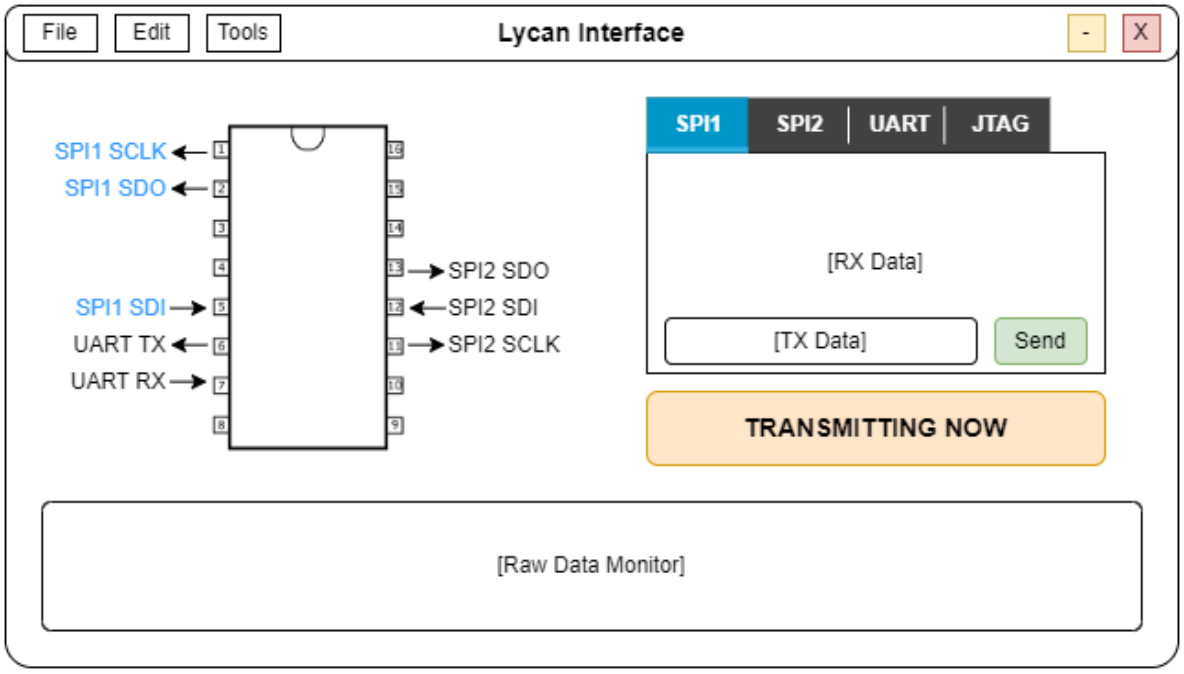
Below is an example utilization of the USB bus. A session will be initiated with configuration data being sent to Lycan. This tells Lycan what peripherals (UARTs, SPIs, etc.) to use, and the various config values for each peripherals (baud rate, stop/parity bits, etc.). Once configured, the USB bus will be used for streaming data between Lycan and the host. Data streaming is asynchronous, which means data will be sent to the host as soon as it is available. The host may also send more configuration data to change the configuration of a specific peripheral, or to reconfigure Lycan to have different peripherals.



Storyboards



The user interface will be an abstraction over the hardware peripherals that makes reading and writing to the DUT through different protocols straightforward. The application has a single page, with a graphic of the DUT and its pins. Some of the pins are assigned to peripherals. For example, there are three pins assigned to the SPI2 peripheral. This peripheral is then selected in the tabbed pane on the right. In this pane, the data being received is displayed in a console. Similar to a terminal window, the user can type in data and hit send to transmit it through that peripheral to the DUT.

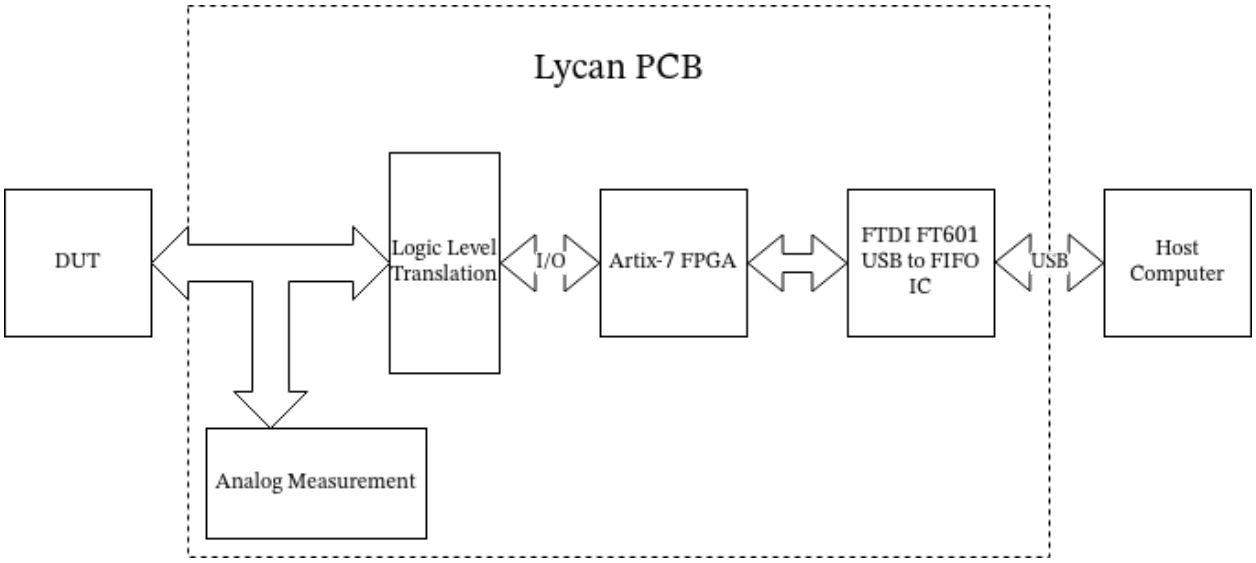


This storyboard is similar to the first, except that SPI1 is now selected, and the peripheral is transmitting at the current moment (shown by the big orange “LED” section).

Draft Schematics

Our custom board will contain four major components: an FPGA implementing the Lycan functionality, an FTDI USB-to-FIFO interface IC, voltage translation for I/Os, and analog

measurement circuitry to measure the voltages and resistances of DUT pins. A block diagram can be seen below. Currently, the main focus is on getting the FPGA and software components to a functional state, so a custom PCB design will be a low priority until we have a proof-of-concept completed on development boards.



References

[1] Digilent, "Analog Discovery 3," digilent.com. https://digilent.com/shop/analog-discovery-3/ (accessed Oct. 6, 2024).

[2] Where Labs, LLC, "Bus Pirate 5 Hardware," buspirate.com. https://hardware.buspirate.com (accessed Oct. 6, 2024).

[3] Future Technology Devices International Limited, "D3XX Drivers," ftdichip.com. https://ftdichip.com/drivers/d3xx-drivers/ (accessed Oct. 6, 2024).

[4] Future Technology Devices International Ltd, "FTD3XX," pypi.org. https://pypi.org/project/ftd3xx/ (accessed Oct. 6, 2024).