Hardware Design Decisions

IoT Interface Project

Jacob Hazelbaker

May 25, 2018

Contents

1 Introduction **2**

2 Design 1: iCEstick as a Hub **2**

3 Design 2: Manual I/O Switch **3**

4 Design 3: iCEstick With I/O Switch **4**

5 Creating a Universal Chip Programmer **5**

6 Conclusion **5**

1 Introduction

There are a wide array of programmable chips and a plethora of tools designed to read, write, and analyze such chips. Continuously moving the programmable chip from one tool to another can be quite tedious, especially for chips which have many I/O pins. Therefore, the first goal of this document is to identify a set of readily available hardware components which can be assembled together so that it will no longer be necessary to unplug one tool so that another tool can access the programmable chip. The second goal of this document is to implement a device, such as the iCEstick, into the design which would allow the user to create custom tools for interacting with the programmable chip.

2 Design 1: iCEstick as a Hub

The iCEstick by Lattice Semiconductor is an FPGA board which is enclosed in the form factor of a typical USB stick. Priced at $25, the iCEstick is an inexpensive device which could be programmed to perform a wide array of tasks through its 16 I/O ports. [4] The Raspberry Pi 3 Model B was also considered, especially since it includes 24 GPIO pins. [3] So too was the Arduino Uno considered as a viable option for a device which could allow the user to create a custom tool for interacting with the programmable chip. The Arduino Uno can readily be used read and write to programmable chips, as was demonstrated by Ben Eater. [1] However, the iCEstick provides the functionality needed at a slightly reduced cost to the Raspberry Pi 3 Model B and the Arduinno Uno.

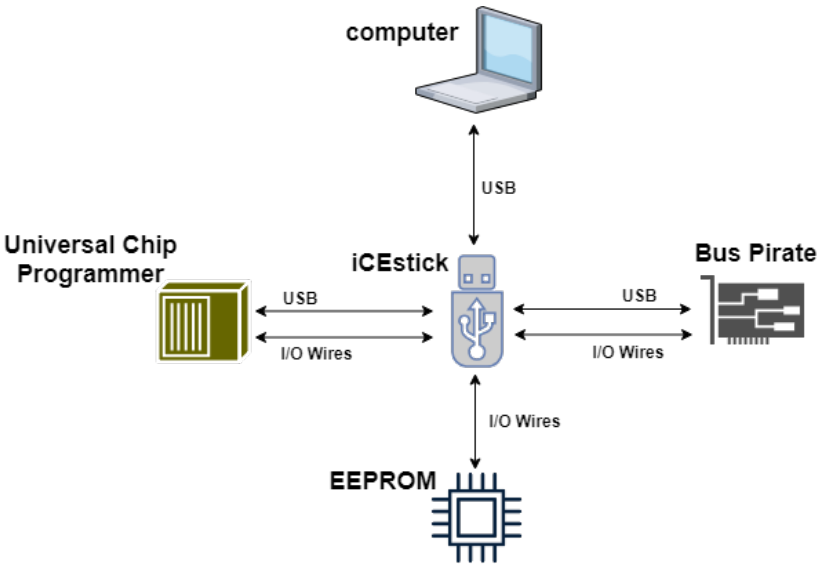

Figure 1: Utilizing an iCEstick to Control I/O With EEPROM

The hardware design diagram in Figure 1 illustrates how an iCEstick could be used to control all of the I/O to and from the programmable chip. Note that in this design even the USB connections to the computer are first routed through the iCEstick. While this hardware layout is simple, it presents several problems:

- The various software running on the computer to control the Universal Chip Programmer and the Bus Pirate might crash or cease to function correctly since all connections are first routed through the iCEstick.
- The iCEstick only has 16 I/O ports, whereas some programmable chips have 48 pins or more.
- Some chips require up to 22 Volts, yet the I/O pins on the iCEstick are only rated for 3.3V. [4]

Design 1 would therefore be very limited in the kind of chips it would be capable of reading and writing to. While this design represents a minimalist solution, it would not be a good design decision for the IoT Interface Project.

3 Design 2: Manual I/O Switch

Another minimalist solution is to cut out the iCEstick entirely and focus solely upon solving the issue of having to frequently plug and unplug I/O wires to the programmable chip. This can be accomplished by incorporating a manual I/O switch.

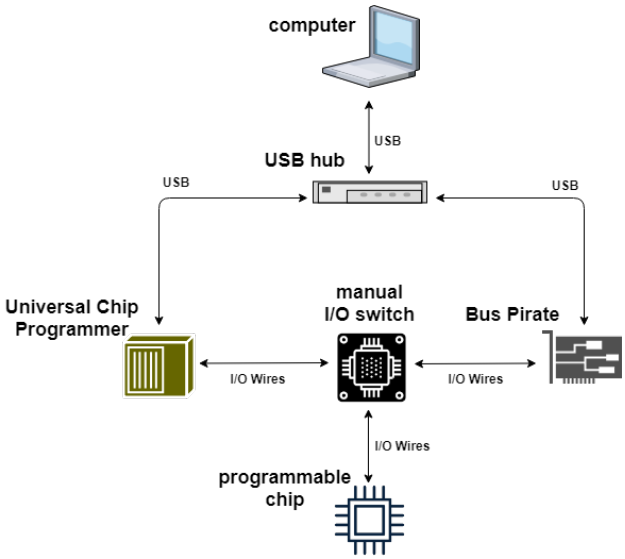

Figure 2: Negating Need to Switch I/O Wires Via Use of a Switch

The hardware design depicted in Figure 2 would effectively prevent the Bus Pirate and the Universal Chip Programmer from interfering with one another, potentially damaging the programmable chip. Also, the inclusion of a USB hub enables the user to have all of the devices connected at once. This design, however, does not include a device, such as the iCEStick, which would easily allow the user to create a custom tool to interact with the programmable chip.

4 Design 3: iCEstick With I/O Switch

The use of a USB hub and a I/O switch would make the final product would meet the first design requirement of negating the need to plug and unplug wires to use different tools on the programmable chip. Including an iCEstick would satisfy the second design requirement of providing a platform by which the user may create their own custom tool to interact with the programmable chip. The hardware design chosen is therefore Design 3, depicted in Figure 3.

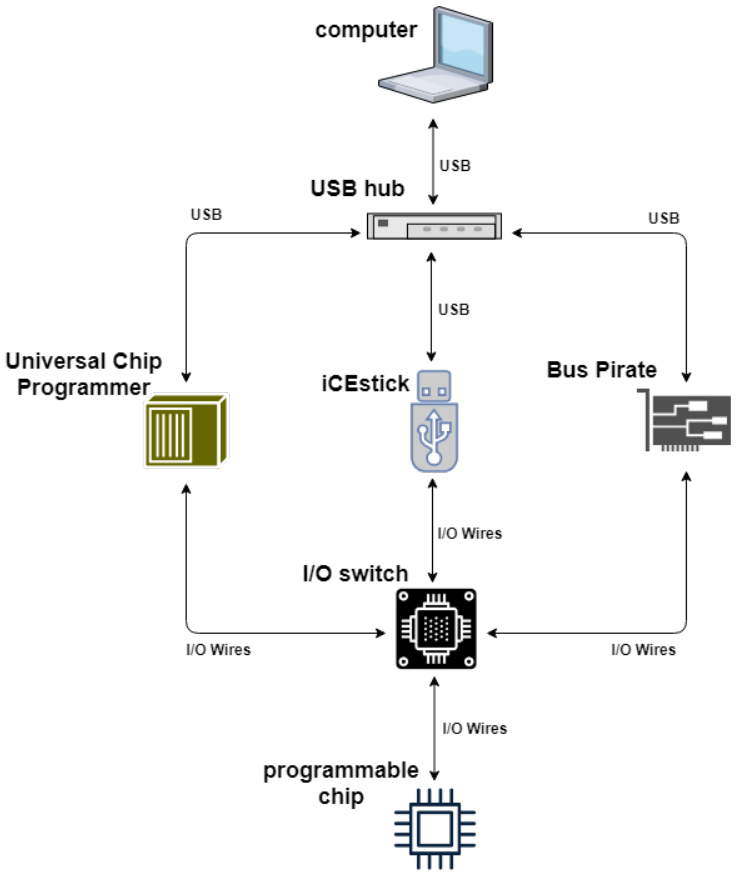

Figure 3: Design Providing Easy Wire Management and Custom FPGA

5 Creating a Universal Chip Programmer

The possibility of creating a universal chip programmer from scratch was also explored. One benefit of this approach would be that more of the finished product could be represented on a single PCB. Moreover, if the finished product were to be reproduced, then it would arguably cost less than a design which would require purchasing one of the universal chip programmers which are currently on the market

After researching some of the design requirements for a universal chip programmer, however, it has been decided not to pursue this route. Even the renowned Data IO universal chip programmer produced in the late 1980's was astonishingly complex, including 12 bit DACs to precisely control voltage and current output. [2]

Developing and maintaining the software for a universal chip programmer would also prove to be quite challenging. The creator of such a device would need to program the pinout for each model of chip supported. Modern universal chip programmers support thousands of different models of chips, such as the Xeltec SuperPro 6100 which supports over 100,000 different chips. [5] It has therefore been decided to use an existing universal chip programmer, instead of designing one from scratch.

6 Conclusion

It is my recommendation that the IoT Interface Project utilize "Design 3" depicted in Figure 3. This would enable the user to access the programmable chip with various tools without the need to plug and unplug I/O wires continuously. This design would also provide a platform by which the user may develop customer tools via the use of an iCEstick FPGA.

References

[1] Ben Eater. *Build an Arduino EEPROM programmer*. URL: `https://youtu.be/K88pgWhEb1M`. accessed: 05.24.2018.

[2] EEvblog. *EEVblog 1060 - $35,000 DataIO Unisite Universal Programmer Teardown!* URL: `https://youtu.be/9MIzbTpdt80`. accessed: 05.24.2018.

[3] Raspberry Pi Foundation. *GPIO*. URL: `https://www.raspberrypi.org/documentation/usage/gpio/README.md`. accessed: 05.24.2018.

[4] Lattice Semiconductor. *iCEstick Evaluation Kit*. URL: `http://www.latticesemi.com/icestick#_0A295861592D47E0AF833894914CF0FC`. accessed: 05.24.2018.

[5] Xeltec. *SuperPro 6100*. URL: `https://www.xeltek.com/universal-programmers/superpro-6100-universal-ic-chip-device-programmer/`. accessed: 05.24.2018.