An FPGA Learning Experience: SPI interface to Max10 FPGA

Gregory Raven, KF5N
greg.electricity@gmail.com

July 28, 2019

Abstract

A story of learning FPGA technology with the evolution of a simple project.

1 Introduction

This is the "Golden Age" of SBCs (Single Board Computers). Walk into your favorite brick-and-mortar bookstore (not many remain), and check out the magazine shelf. You will find several magazines, or "book-a-zines" dedicated to "Arduino" or "Raspberry Pi".

Or look in the magazine "Make", or use their "Makers' Guide to Boards":

https://makezine.com/comparison/boards/

Using the above webpage, you can select a "Type" of board from three categories:

- Microcontroller
- Single Board Computer
- FPGA

While there are dozens of boards listed, there are only three shown for "FPGA". This list is not entirely accurate, as more FPGA boards exist, however, it gives an idea of the relative scarcity of FPGA boards.

There are a ton of resources online, and dozens of books and magazines on SBCs which you can find at the local bookstore. The ARRL store sells quite of few SBC themed books as well. Do a search for FPGA. Nothing!

I think it is fair to say that FPGA technology has not made it very far into the ranks of hobbyists. Amateur radio operators have certainly been pioneers in this, with numerous "Software Defined Radio" projects going back to the era when the devices were really expensive. FPGA is an amazing technology, a sort of "3D printer of digital electronics". Perhaps there are other applications of FPGA within the amateur radio community in addition to SDR?

I wanted to experiment with this FPGA stuff! The current crop of devices and boards has lowered the cost of entry. This is my story of a first project in FPGA. Hopefully some of the perceived barriers can be lowered, and other hams can be encouraged to try working with this fascinating technology.

2 Where to Start

I am already familiar with the most common "Single Board Computers", the Arduino, the BeagleBone, and the Raspberry Pi. It a good thing to have some experience developing with a SBC before attempting to tackle FPGA. I have a couple of BeagleBone projects which are in use at my home on a daily basis. Very practical!

One of the ways I went up the learning curve on SBCs was this lecture series by Bruce Land of Cornell University:

https://www.youtube.com/watch?v=FYy6JN0vpg0&list=PLKcjQ_UFkrd4z2qoFuJ1jtVhCSuxxCTpk

This course uses a PIC32 SBC. However, the material in the lectures is generic enough to apply to any SBC. Even better, a second course covers FPGA!

https://www.youtube.com/playlist?list=PLKcjQ_UFkrd7UcOVMm39A6VdMbWWq-e_c

and the matching website:

http://people.ece.cornell.edu/land/courses/ece5760/

This material gave me a pretty good idea of what was involved in FPGA work. I didn't use the same development board as used in this course, however, I used another Intel FPGA based board. So the development tools and general flow of working with the FPGA are similar.

You will need to use a variety of resources to answer questions and solve problems as you go up the FPGA learning curve. Nothing about this is super difficult. It seemed to me that most of the effort is in handling large quantity of details. You will need to spend a lot of hours absorbing this stuff. I think that FPGAs are at least a little bit harder than SBCs.

3 Most Significant Source of Learning: Intel

You will need to get an account at your FPGA vendor's website. I used an Intel based board, and the account was free. On the Intel site, you will have access to large amounts of learning resources for FPGA. Be prepared to spend many hours watching videos and studying documentation, whatever vendor you have chosen. Intel has good material, and you can learn a lot! A recommended starting point is the video series "Become an FPGA Designer in 4 Hours". The 4 hours part is perhaps a bit optimistic, but it is good material:

https://www.intel.com/content/www/us/en/programmable/support/training/course/odswbecome.html

4 Other Resources for Going Up the Learning Curve

You will need to go up the learning curve on Verilog (or VHDL). Here are a couple of inexpensive books (< $20) which will get you going:

https://www.amazon.com/gp/product/1075968437/ref=ppx_yo_dt_b_asin_title_o00_s03?ie=UTF8&psc=1

http://www.lulu.com/shop/george-self/exploring-digital-logic/paperback/product-22747579.html

5 Github Repository for this Project

The documentation and code for this project is located in this git repository:

https://github.com/Greg-R/rotorcommand1

6 FPGA and Verilog Book Commentary

"Designing Video Game Hardware in Verilog" Steven Hugg, first printing 2018

A very practical introduction to digital hardware using the early history of video games as a means of illustrating the technology. The book includes significant introduction to Verilog and its relationship to the physical circuitry.

However, it is not primarily an FPGA book! The reader is encouraged to use a web-based Verilog simulator:

http://8bitworkshop.com

There is an example FPGA which uses the iCE40HX-1K iCEstick. This is a low-cost device ($40 on Amazon). Development can proceed with the official Lattice tool chain, or with an open-source system known as “IceStorm”:

http://www.clifford.at/icestorm

“Programming FPGAs, Getting Started in Verilog” Simon Monk, 2017 McGraw-Hill Education

“Exploring Digital Logic” George Self.

http://www.lulu.com/shop/george-self/exploring-digital-logic/paperback/product-22747579.html

“Designing Digital Systems with SystemVerilog” Brent E. Nelson, Brigham Young University

“FPGAs for Dummies” eBook by Intel.

https://plan.seek.intel.com/PSG_WW_NC_LPCD_FR_2018_FPGAforDummiesbook

7 Low Cost FPGA Development Boards

Maker's Guide to Boards:

https://makezine.com/comparison/boards/

This web page is very useful. Note the selector for "Type":

- Microcontroller
- Single Board Computer
- FPGA

TinyFPGA $38.95:

https://www.sparkfun.com/products/14829

IceStick. $40:

https://www.amazon.com/LATTICE-SEMICONDUCTOR-ICE40HX1K-STICK-EVN-Evaluation-iCE40HX1K/dp/B00R3QU9K0/ref=sr_1_3?crid=2HWV0KVHYP7NO&keywords=ice+stick+fpga&qid=1564252268&s=gateway&sprefix=icestick+fpga%2Caps%2C145&sr=8-3

Sipeed TANG PriMER FPGA Dev. Board $17.90:

https://www.seeedstudio.com/Sipeed-TANG-PriMER-FPGA-Development-Board-p-2881.html

Terasic DE10-Lite $85:

https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=218&No=1021

Terasic has an Amazon store front. The above link is to their website.

8 Choosing a Development Board

This was a not a straightforward process!

The first FPGA development board I purchased was the Numato Lab "Elbert V2" ($29.95) This board has the Xilinx XC3S50A Spartan 3A FPGA device containing 1584 logic cells and 54 KB RAM. The board has a nice selection of peripherals which can be driven by the FPGA:

- 16 MB Flash Memory
- USB 2.0 interface for Flash programming
- 8 LEDs
- 6 push buttons
- DIP switch
- VGA connector

- Audio connector
- SD card adapter
- 3x 7 segment LED
- 39 IOs

Note that the USB capability of this board is for Flash programming only. It is not an interface to the FPGA.

The reason for choosing the Elbert V2 was the book “Programming FPGAs" by Simon Monk. The book features the Elbert, along with the “Mojo” and “Papilio” boards.

It is interesting to note that a quick search on Amazon shows only three “hobbyist” style books for FPGA. Contrast this to Arduino/Raspberry-Pi, where you will find dozens, maybe hundreds!

If you have a specific application, then it will be straightforward to decide if the development board meets your requirements. For a person who is only interested in getting going in FPGA, then more peripheral devices are better than less. Also note that the example above has 39 IOs, so you can add your own. If the IO is in the format of the common Arduino, then adding peripherals may be quite easy to accomplish.

I was able to install the development software for the Elbert and work through a few of the projects in the book. However, I found the development tools lacking in one area. I was interested in using a more modern version of Verilog, called “SystemVerilog”. So that was a bit of a problem, and a search for another board began.

My motivation for SystemVerilog was to try some of the new features added, including those used for “testbenches” (simulation). That is only an expression of my particular interest, as the Verilog used in the Xilinx tool for this device is fine and can be used for maximum benefit with this device. This does, however, indicate that the development environment which mates up with the FPGA device is as important as the device itself. So before you choose a board, be sure to download and install the development tools first. Look at the documentation and decide if you will be comfortable with that particular development tool capabilities.

So looking around a bit more, I found this board after viewing Bruce Land’s great lecture series on youtube:

https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=165&No=836

The DE1-SOC is used in Cornell’s “ECE 5760 Advanced Microcontroller Design and system-on-chip” course.

The board requires the "Quartus" development tool, which met my requirement for SystemVerilog. However, this board is more complex than desired. This board uses an advanced ”System On Chip” (SOC) which is a combination of microcontroller (dual-core ARM Cortex A9) and FPGA in a single device. Price is $249, which seemed a little steep for a project which might not work out. The added complexity of the ARM processors, having to deal with an unfamiliar distribution of Linux, and the extra work involved seemed like too much. Indeed this board is immensely capable, so maybe I will come back to this one in the future!

Fortunately the same company, Terasic, makes board which are better suited to a beginner. Perusing their site, I found this board, the DE10-Lite ($85):

https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=234&No=1021

This looked like a good choice, and it was purchased. Later when doing internet searches, there was plenty of example projects already available for this board. It seems to be at least moderately popular in the academic world, and some courses have been based on this board. The MAX10 FPGA is widely supported, although it is not included in the latest version of the Quartus development tool (latest version supported 2018).

A good feature of this board is the Arduino compatible expansion header. In general, the DE10-Lite has been easy to work with, and seems to be robust. I've been using it for many months and it is still alive!

9 FTDI SPI Interface and Julia Programming Language

The inspiration for this project came from a a search at github.com on DE10-Lite:

https://github.com/hildebrandmw/de10lite-hdl

This is a nice collection of work done with the DE10-Lite board for academic purposes. What really got my attention is the project play_gif:

https://github.com/hildebrandmw/de10lite-hdl/tree/master/projects/play_gif

The interesting feature of this project is the usage of the USB to load data (animated GIF image file) to the FPGA. So this is a data pipeline from a Linux desktop to the FPGA which is built into the board!

But it is a bit clunky to use! The interface is via "JTAG", which is typically used as a debugging interface. It is not specifically intended for mass data transfer, but in this case it was pressed into service.

It also requires a "TCL Server", and a running instance of the Quartus development tool! Not exactly what I was looking for, but I got the demo to work easily!

What it does is conceptually simple. The FPGA part of the project implements a VGA[1] interface to the connector on the DE10-Lite board. The image loaded from the desktop computer is sliced into its constituent "frames". Another interesting aspect of the project is the interface to the SDRAM of the DE10-Lite which is a 64MB external part on the board. The sliced-up image is loaded into the SDRAM, and then another control module pages the VGA output through the memory. Thus you see the animated GIF displayed on the monitor. Really you are seeing in a very direct manner the data loaded into the SDRAM!

9.1 IP and Platform Designer

First, a little bit of FPGA jargon. "Intellectual Property" (IP) in the context of semiconductor devices is a block of circuitry which has been heavily engineered and refined to perform some particular function. It could be patented or otherwise protected from duplication by competitors. Due to the way integrated circuits are manufactured, blocks of "IP" can be added to the silicon and be expected to perform to the IP

[1] VGA is a relatively simple video standard which seems to be common on many FPGA development boards.

owner's specifications. Typically IP can be included as part of a design kit, or it can be paid for with a license fee.

IP is good because it can reduce engineering design effort, improve performance, and enhance quality. The trade-off is license fee cost, and you don't necessarily get exactly what you want.

In our case, we are given a whole bunch of IP for free that we can experiment with! This is bundled into "Platform Designer" which is a tool-within-a-tool in the Quartus design suite.

To do justice to this there should be an entire section on "Platform Designer". I will summarize here. There are excellent video Platform Designer tutorials which you can access if you register for a free account at the Intel web site.

"Platform Designer" is a building-block system. You get a library of IP, along with a mechanism to hook them together. The design is bundled into a "Qsys" file. Let's have a look at the Qsys part of the play_gif project:

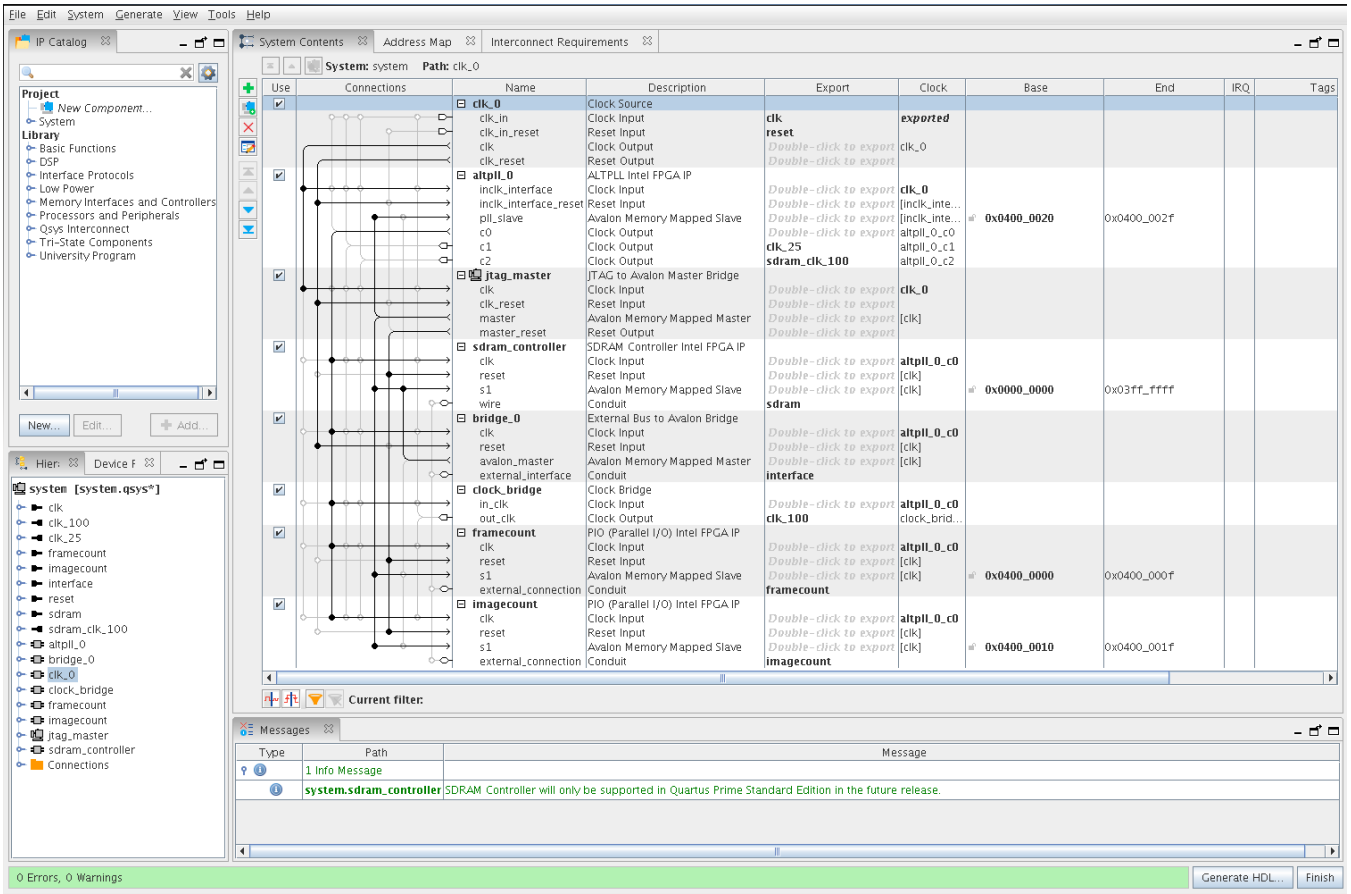

Figure 1: Play_gif project Qsys file

The upper left corner of the GUI is the library of IP. Some of the categories:

- Basic Functions
- ESP

- Memory Interfaces
- Processors and Peripherals (including a “Nios” processor)
- Memory Interfaces and Controllers

There is a sort of “sub-library” called “University Program” which includes:

- Audio and Video
- Bridges
- Clocks
- Communication
- Generic IO
- Memory

The project used this IP:

1. ALTPLL Intel FPGA IP
2. JTAG to Avalon Master Bridge
3. DRAM Controller Intel FPGA IP
4. External Bus to Avalon Bridge
5. Clock Bridge
6. (Parallel IO) Intel FPGA IP

The above IP can be seen in the column “Name”. There are two instantiations of the Parallel IO. In the column “Connections” can be seen the graphical interconnections between the IP blocks. Connections are made by simply clicking on the circles at the intersections of the “wires” between the IP. Thus an entire system can be assembled using this GUI. No writing of Verilog required! The project does include some hand-written Verilog. This “Qsys” design is “dropped in” to the project as a Verilog module.

Here is what the system looks like:

What this diagram does not show is the requirement for a running Quartus and a TCL/JTAG server program.

The “Avalon Interconnect” deserves some explanation. This is a system bus used in the MAX10 FPGA. From the Avalon Interface specification:

> Avalon® interfaces simplify system design by allowing you to easily connect components in Intel® FPGA. The Avalon interface family defines interfaces appropriate for streaming high-speed data, reading and writing registers and memory, and controlling off-chip devices. Components available in Platform Designer incorporate these standard interfaces. Additionally, you can incorporate Avalon interfaces in custom components, enhancing the interoperability of designs.

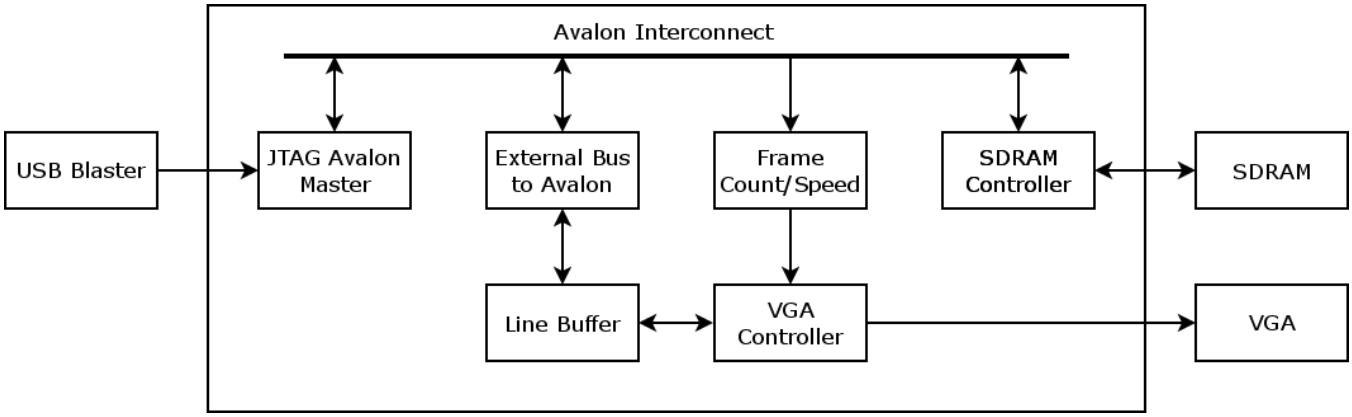

Figure 2: Play_gif System Diagram

It is an internal bus standard used to connect Avalon bus masters and slaves. So it is a single-click process to connect Avalon components in Platform Designer. It is really amazing what you get for such little effort!

This project is interesting, and shows a path to communication between a desktop computer and the FPGA. However, the JTAG + Quartus + TCL/JTAG Server is cumbersome. A SPI to Avalon bus IP component is listed in the catalog. What if the JTAG and stuff could be replaced by something simpler?

So that is what evolved into my "introductory FPGA project". The revised system diagram:

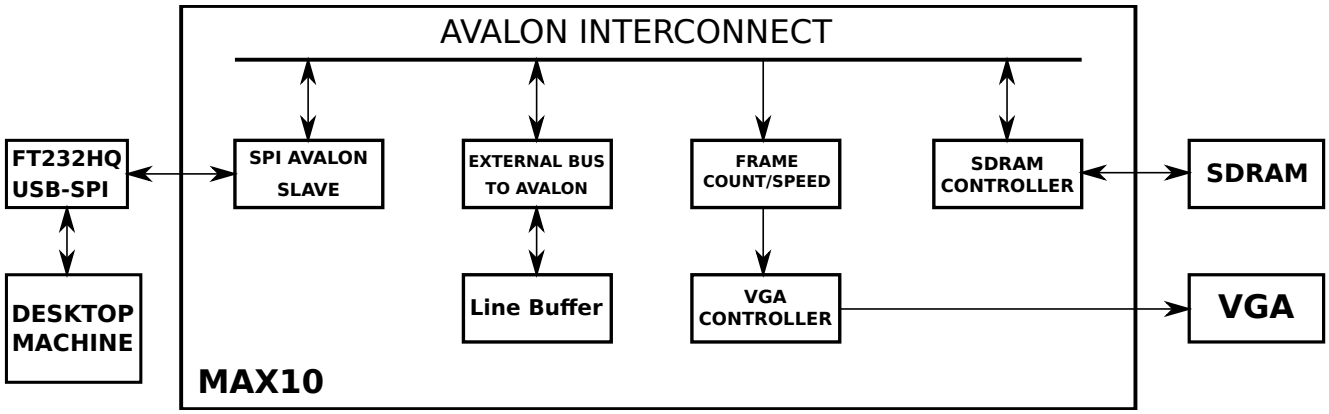

Figure 3: Play_gif with SPI System Diagram

The significant change on the FPGA is the swapping of the JTAG Avalon bus master with the SPI-Avalon slave. This was not entirely a drop-in replacement, as there were changes to reset and clock connections in addition to swapping JTAG to SPI components. But it is easy, and the swapping can be done in a couple of minutes.

External to the DE10-Lite board, there is a new USB to SPI adapter board. This board is based on the FT232H chip by FTDI. This can be bought from eBay for about $10. Search for "ft232 spi" and you will find several options. I recommend one with headers to allow it to be plugged into a common breadboard.

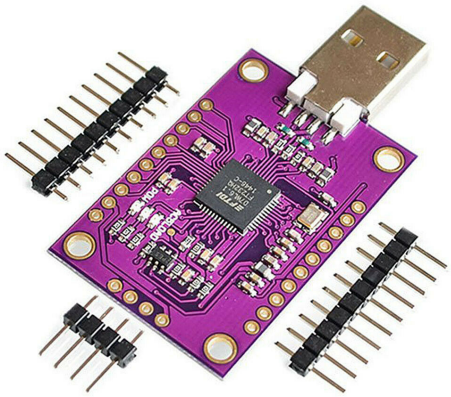

Figure 4: FTDI FT232H USB to SPI Break-out Board

Other changes required for SPI:

- The ports on the QSYS module changed (JTAG -> SPI), thus the Verilog module in which it is instantiated required minor changes. This was done with the text editor feature of Quartus.
- Another change is required to the FPGA pins. The new SPI bus must be routed to some easily accessible header on the DE10-Lite board. Since the board has an Arduino compatible header, and this header has a standard set of four pins for SPI, those pins were used. The details can be seen in the DE10-Lite manual provided by Terasic. The "Pin Planner" tool was used to make the changes.
- The "Synopsys Design Constraints" (.sdc) file was updated to incorporate the SPI bus.

Here is the crazy breadboard hook-up:

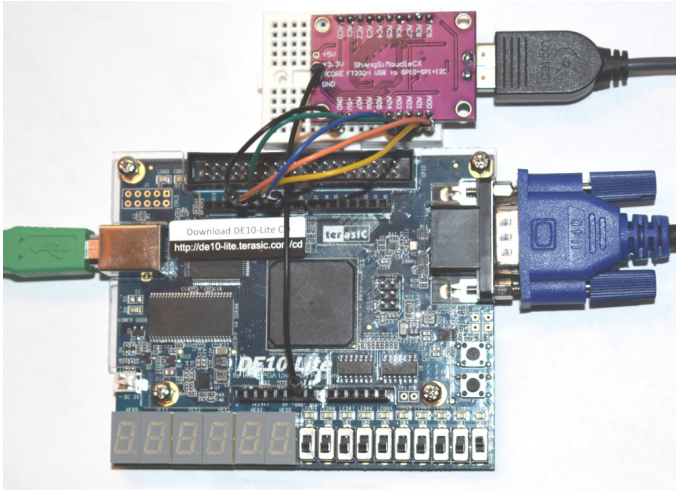

Figure 5: DE10-Lite Connected to FTDI SPI Breakout

In spite of the length of the breadboard jumper wires, the SPI interface performed remarkably well right up to the FTDI limit of 30 MHz.