

An FPGA Learning Experience: SPI Interface to Max10 FPGA

ARRL and TAPR 38th Annual Digital Communications Conference

Gregory Raven KF5N

September 21 2019



Gregory Raven, first licensed novice WD5HUV in 1978.

KF5N since 1980.

35 Year Motorolan, entire career designing Two-way FM Radios. Mostly interested in Ham Radio experimenting and science. Residing in Florida, all of my antennas have been blown down by hurricanes or struck by lightning.

NOT a professionally trained digital hardware engineer! I am a very "Analog-RF" engineer. Eventually I decided learning digital was a good idea, and I attended the TAPR conference in Maryland in 2011. Slowly going up the learning curve since then ...



The Golden Age of Single Board Computers

My early introduction to "digital" was via Arduino, and later Beaglebone and Raspberry Pi. The so-called "Single Board Computer" or "SBC".

I spent a couple of years working with SBCs. This was greatly accelerated when I got my hands on "Exploring Beaglebone, Tools and Techniques for Building with Embedded Linux". Great book!

Looking back, I think TAPR St Petersburg 2016 was essentially an SBC-themed conference! Obviously there is a huge opportunity for SBC in Amateur Radio.



Prerequisites to begin FPGA?

- Digital Hardware experience (but not that much!)
- Work with SBCs for a while. Finish some projects.
- C or similar programming. You did this working with SBCs?
- Discretionary time to expend. FPGA is not rocket science, but it is time consuming!
- Optional: Projects with some flavor of RTOS. Helps understanding of why FPGA is great! (ESP32, FreeRTOS, \$10)



The State of “Hobbyist” FPGA

“Hobbyist” FPGA is miniscule compared to SBC:

- Dozens, maybe hundreds of SBC books. FPGA: 4
- Fair representation on hackster.io and hackaday.io.
- ARRL bookstore: lots of Arduino, RPi, nothing on FPGA.

As usual, Amateur Radio experimenters were pioneers in new technology, having used FPGA to create early state-of-the-art “Software Defined Radio” transceivers. This goes back to at least the early 2000s, and perhaps years earlier than that. Does anyone know the first Amateur Radio application of FPGA?



FPGA Boards?

What is out there to work with?

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

Currently, most FPGA are “development boards” intended for industrial or academic applications.

There is a little bit of good news! Will return to this topic later ...



FPGAs in the Silicon Ecosystem

Boiling it down to fundamental capabilities:

INSERT ASIC FPGA SOC diagram here.



What is exciting about FPGAs?

“3-D Printer for Electronics?”

HARDWARE not software!

This gives you the power to create multiple specifically targeted digital machines which can work independently of one another. This is the capability that SDR uses to crunch DSP math very efficiently while handling data flow simultaneously. A list of FPGA applications from Wikipedia:

https://en.wikipedia.org/wiki/Field-programmable_gate_array



How Do You Begin?

For FPGA this is not as clear as SBCs. An SBC beginner I would say to get an Arduino UNO and a good book or website and get going. Later graduate to embedded Linux and RPi, Beaglebone, or equivalent.

So I will give an example here, not correct for everyone ... Lectures by Bruce Land of Cornell University using PIC32:

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

This course uses a PIC32 microcontroller board. However, the material in the lectures is generic enough to apply to any board which can be programmed in C. Even better, a second course covers FPGA!

https://www.youtube.com/playlist?list=PLKcjQ_UFkrd7Uc0VMm39A6VdMbWWq-e_c

and the matching website:

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



Online MOOC Style Courses?

I tried:

- Coursera
- Udemy
- Hackster.io

I didn't get much out of the above. They didn't really follow through to a practical application. Hopefully some new courses on FPGA will appear, and I'm sure they will. So far my experience with FPGA MOOCs is underwhelming.



Choosing a Development Board

As mentioned earlier, the choice is limited compared to SBCs. Most are targeted at industrial/academic application. Many are expensive!

The Cornell FPGA course uses an "SOC" style device, and the board cost is \$250. For someone who is not sure about FPGA, that is too much!

Also, an "SOC" is too much for a beginner. This has two Cortex A9 processors, runs its own flavor of embedded Linux, and it looks overwhelming!

The same company making the DE1-SOC also makes others:

<https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=163>



What I Selected

Terasic DE10-Lite is a cost-effective Altera MAX 10 based FPGA board. The board utilizes the maximum capacity MAX 10 FPGA, which has around 50K logic elements(LEs) and on-die analog-to-digital converter (ADC). It features on-board USB-Blaster, SDRAM, accelerometer, VGA output, 2x20 GPIO expansion connector, and an Arduino UNO R3 expansion connector in a compact size. The kit provides the perfect system-level prototyping solution for industrial, automotive, consumer, and many other market applications.

The DE10-Lite kit also contains lots of reference designs and software utilities for users to easily develop their applications based on these design resources.

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



What I Selected

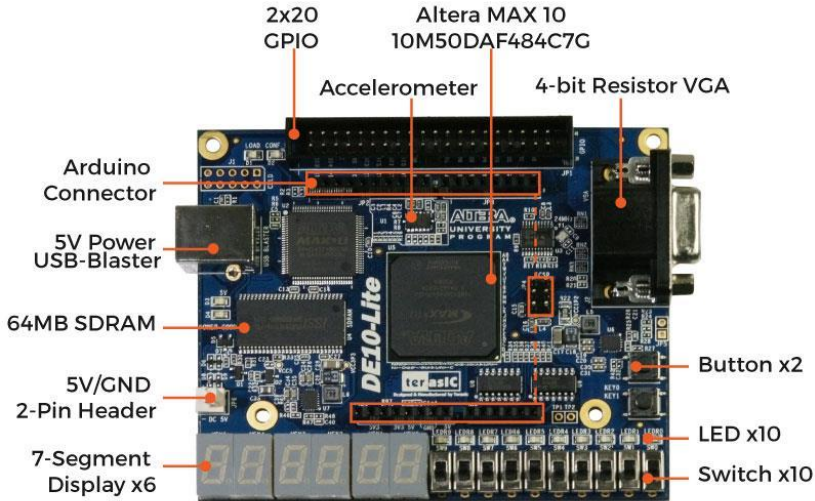


Figure: Terasic DE10-Lite \$85



Wait!!! Don't Put it in the Cart and Purchase

This is not a moment for “instant gratification”! You have some work to do first.

Before you take the plunge, you **MUST** install the tools required to make this thing work! I chose an Intel (formerly Altera) based board (MAX10), so this means installing “Quartus”. Xilinx devices will require “Vivado”.

But to do this, you should get a “free” account at the Intel site:

<https://www.intel.com/content/www/us/en/products/programmable.html>

Go here for “Quartus Prime Lite”:

<http://fpgasoftware.intel.com/?edition=lite>

Make sure you can install and run this large beast!



Development Platforms

I am a huge fan of Linux and I almost always use a distribution of Ubuntu. Quartus has a version for Windows, however, I have not tested it.

Regarding Ubuntu, I don't recommend running Quartus in Ubuntu directly. The simulation tool, Vsim, will not run.

Instead, use a Virtual Box to create a "virtual machine". I experimented with Centos 6 and 7. Centos 6 was the easiest to configure. Even so, there were a couple of tricks to get everything working smoothly.

If you go the route of a virtual machine, you will need to work out permissions with USB devices, as the DE10-Lite uses the so-called "USB Blaster" mechanism for programming.

Quartus is a significant user of desktop real estate. I don't think anything less than full HD (1920x1080) is going to be satisfactory. I found FHD to be marginal, and I bought a new 27 inch "QHD" display, which is 2560x1440 pixels. Much better!



Project Inspiration: JTAG Interface to DE10-Lite

I had abandoned the SOC with embedded Linux as too complex, however, I still wanted a means of “talking” to the FPGA, preferably from my Ubuntu box.

After a bunch of searching, I found this:

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

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! This met my requirement that I be able to control the FPGA remotely from a desktop (or SBC) computer. 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. The interface is a bit clunky to use. It requires a TCL Server, and a running instance of the Quartus development tool!



Project Inspiration: JTAG Interface to DE10-Lite

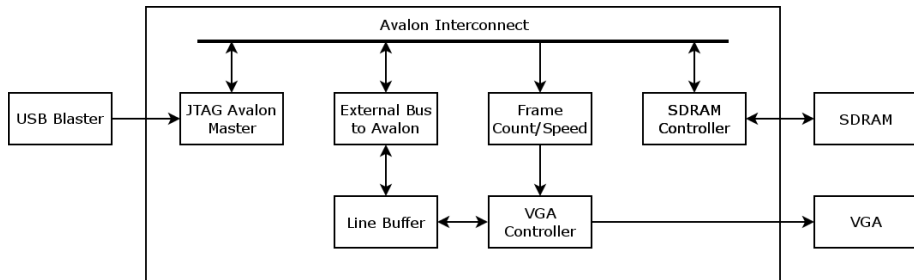


Figure: Play_gif JTAG Interface System Diagram

Not shown is a running instance of Quartus and TCL/JTAG server required to make this system function.



IP and Platform Designer

FPGA Jargon: “Intellectual Property” or “IP”

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 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.



IP and Platform Designer

File Edit System Generate View Tools Help

IP Catalog System Contents Address Map Interconnect Requirements

System: system Path: clk_0

Use Connections Name Description Export Clock Base End IRQ Tags

<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	clk_0	Clock Source		clk_0	exported			
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	clk_in	Clock Input	clk					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	clk_in_reset	Reset Input	reset					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	clk_out	Reset Output	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	clk_reset	Reset Output	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	altpll_0	ALTPLL Intel FPGA IP						
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	inc1k_interface	Clock Input	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	inc1k_interface_reset	Reset Input	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	pll_slave	Avalon Memory Mapped Slave	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	c0	Clock Output	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	c1	Clock Output	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	c2	Clock Output	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	jtag_master	JTAG to Avalon Master Bridge	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	clk_in	Clock Input	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	clk_reset	Reset Input	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	master_reset	Avalon Memory Mapped Master	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	sdram_controller	SDRAM Controller Intel FPGA IP	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	clk_in	Clock Input	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	reset	Reset Input	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	s1	Avalon Memory Mapped Slave	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	wire	Conduit	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	bridge_0	External Bus to Avalon Bridge	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	clk_in	Clock Input	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	reset	Reset Input	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	avalon_master	Avalon Memory Mapped Master	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	external_interface	Conduit	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	clock_bridge	Clock Bridge	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	in_clk	Clock Input	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	out_clk	Clock Output	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	framecount	PIO (Parallel I/O) Intel FPGA IP	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	clk_in	Clock Input	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	reset	Reset Input	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	s1	Avalon Memory Mapped Slave	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	external_connection	Conduit	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	imagecount	PIO (Parallel I/O) Intel FPGA IP	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	clk_in	Clock Input	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	reset	Reset Input	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	s1	Avalon Memory Mapped Slave	clk_0					
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	external_connection	Conduit	clk_0					

Current filter:

Messages

Type	Path	Message
1 Info Message		
1	system.sdram_controller	SDRAM Controller will only be supported in Quartus Prime Standard Edition in the future release

0 Errors, 0 Warnings

Generate HDL... Finish

IP and Platform Designer

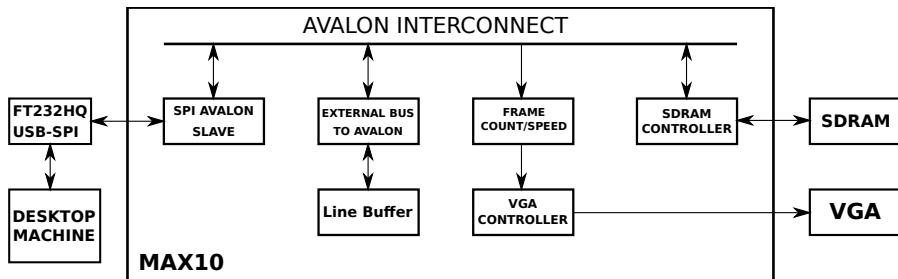


Figure: Play_gif with SPI System Diagram

Replacing the JTAG with a SPI interface. Drop in the included SPI IP!



DE10-Lite + FTDI USB to SPI

Here is the rapid-prototype breadboard hook-up:

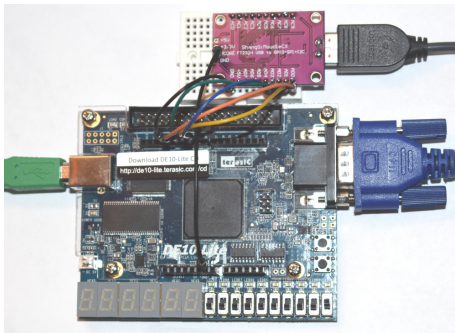


Figure: DE10-Lite Connected to FTDI SPI Breakout

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

