UPduino Documentation

Release 0.1

Venkat Rangan

Sep 13, 2020

Contents

CHAPTER 1

TinyVision.ai

1.1 UPDuino v3.0: PCB Design Files, Designs, Documentation

The UPDuino v3.0 is a small, low-cost FPGA board. The board features an on-board FPGA programmer, flash and LED with _all_ FPGA pins brought out to easy to use 0.1" header pins for fast prototyping.

The tinyVision.ai UPduino v3.0 Board Features:

- Lattice UltraPlus ICE40UP5K FPGA with 5.3K LUTs, 1Mb SPRAM, 120Kb DPRAM, 8 Multipliers
- FTDI FT232H USB to SPI Device
- _ALL_ 32 FPGA GPIO on 0.1" headers
- _ALL_ FTDI pins brought to test points
- 4MB SPI Flash
- RGB LED
- On board 3.3V and 1.2V Regulators, can supply 3.3V to your project
- Open source schematic and layout using KiCAD design tools
- Integrated into the open source APIO toolchain

Please see the wiki page for the changes that were implemented from v2.1. Some salient features are:

- 4 layer board with a solid ground plane, proper layout and decoupling for good signal integrity and FPGA operation
- Access to on-board 12MHz oscillator using a jumper (short R16)
- _All_ FPGA pins including LED driver pins are brought to 0.1" headers
- qSPI flash capability
- tinyFPGA bootloader compatible

Please fill out the survey to suggest improvements to this board. We really appreciate the feedback and will make improvements as business permits!

Useful links:

- osresearch: large collection of very useful code and a good overview.
- UPduino FPGA tutorial using APIO
- A very detailed blog on implementing a RISCV in the FPGA

1.1.1 Introduction

The UPduino 3.0 is a highly capable device. We will get started with setting it up, and the "hello world" of this FPGA, getting an LED to blink!

First Steps

First things first, after ensuring your board is functioning and shows up as a USB device, the tools need to get installed. There are two paths to do:

- APIO or icetools (For MacOS, Linux, and Windows)
- Lattice Radiant (Linux and Windows only)

To install these tools, please go to the "Tool Installation" document!

1.1.2 Tool Installation

The UPduino can use an FPGA image generated from either the open source icestorm/apio toolchain or the Lattice Radiant tools.

icestorm Tool Installation

You can follow the instructions to install the icestorm toolchain from various links on the web. Some are listed below for reference: - https://github.com/FPGAwars/toolchain-icestorm/wiki

APIO installation

APIO is a powerful open source ecosystem for FPGAs. To install it, use pip, and go: `pip install apio`

As the UPduino is fairly new, however, this release does not include the software to include the UPduino 3. Thus, you will need to manually configure it. Dowload the newest commit of APIO from here: https://github.com/FPGAwars/apio

To find out where pip installed apio on your system, go: `pip show apio`

Go there, open the apio folder, and replace its contents (but not the folder itself) with the contents of the "apio" folder in the downloaded commit of the apio directory. **Note:** If you would like to be able to easily update it, as apio is frequently updated due to its open sourse nature, you can use Git.

Note, this process of replacement is limited to pip. If you use a different package manager, refer to where it stores its downloaded and installed packages.

Lattice Radiant Installation

You can follow instructions from teh Lattice website to install Radiant. Note that this tool only support Linux and Windows.

1.1.3 First Steps

1. When you receive your UPduino, make sure it works properly before you proceed further! A simple way to do this is to plug the UPduino into a standard micro USB cable attached to a standard USB power supply such as a computer or a phone charger.

- You should see the green LED (D1) light up and also the 3 color LED go through a Red, Blue Green sequence.
- Also, if you are on a computer, you should see a new USB device called the "UPduino 3.0" show up in your list of USB devices.
- The board shows up as a serial port (COMxx on windows and /dev/ttyxx on Linux and Mac).

2. Download the toolchain of choice: Lattice Radiant and/or icestorm/apio.
3. Download the git repository for the UPduino and go the RTL/blink_led directory.
4. Test your toolchain installation:

- apio/icestorm toolchain: - Type in "make" and this should create a bin file to be uploaded to the UPduino. - For Windows, you will need to install Zadig and go through the process of switching the UPduino to the libusbk driver so that iceprog can see this. An alternative is to install the iceprog tool for windows and use that for programming instead of the icestorm version, some instructions are on [this forum](https://forum.1bitsquared.com/t/official-win10-instructions-missing/73). - Program the UPduino and ensure the green (D1) LED lights up and the 3 color LED starts cycling through its sequence.

5. Fiddle with the code!

1.1.4 Specifications

The UPduino supports the following features: - Lattice iCE40 UP5K UG48 FPGA

1.1.5 Blinking an LED

Blinking an LED on the UPduino 3 is its "hello world"

Getting Started

The sample code to blink an LED is built-in to this repository!

https://github.com/tinyvision-ai-inc/UPduino-v3.0

Go to RTL > blink_led folder, to see the example.

Running the Code

The code can easily be run! First, make sure the UPduino is plugged in.

If you want to run the code using icetools (iceprog, specifically), type: `make`

Then: `iceprog rgb_blink.bin`

You should be the LED blink!

If you want to use apio, first type: `apio init --board upduino3`

Then: `apio verify` to make sure your code works, and then finally:

`apio build` and `apio upload`

The LED should blink now!

Making Changes

So the LED is (hopefully) blinking now. If you want to change the code or learn how it works, open the rgb_blink.v file

.v stands for Verilog, a low-level programming language

The code runs asynchroniously, meaning that multiple lines can run at once on the board.

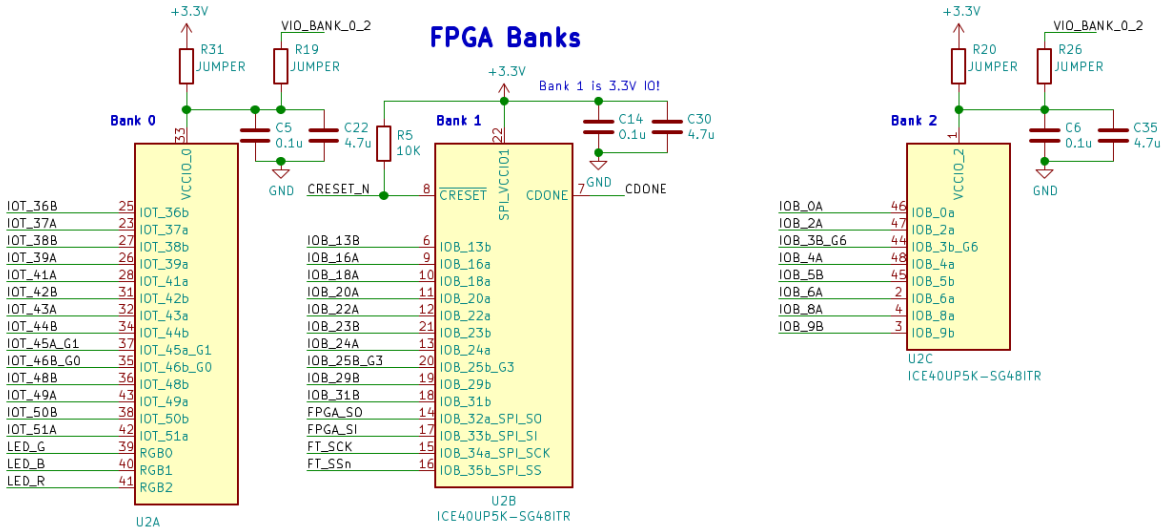
Anyways, **tinker around with the code!** Some things to do:

- Make it blink a different color - Maybe white? Yellow? A random hex value?
- Make it blink in a different interval
- Make it go fast
- How about slow?
- Can the LED stay on a solid color?
- Try different brightness values - How bright can it go? How many levels are there? - Can you do a "breathing" effect, where the LED eases in and out of brightness?

There are endless possibilities with this board! Getting an LED to blink is just the start...

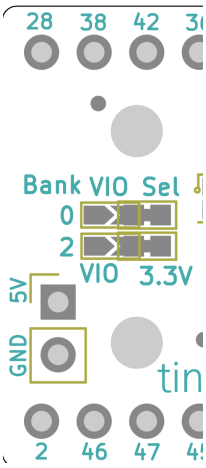
1.1.6 How to connect the two banks in the FPGA to a voltage other than 3.3V?

The FPGA on the UPduino v3 has 3 banks hooked up as follows:



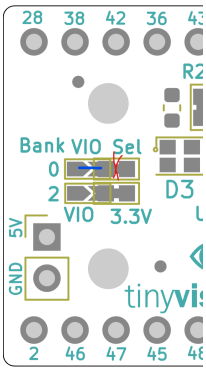
Bank 1 is connected to the 3.3V supply and cannot be modified since it is hooked up the the Flash and the FTDI parts, both of which are 3.3V devices.

Bank 0 and Bank 2 however, can be changed to source/sink voltages at any arbitrary voltage within the Lattice FPGA IO specification by the following scheme in this part of the board:



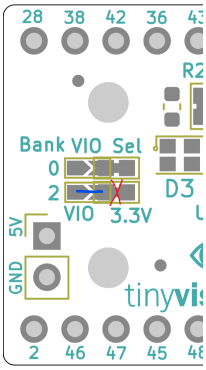
Bank 0:

Cut the trace for R31 (shorted on the board) and solder across R19.



Bank 2:

Cut the trace for R20 (shorted on the board) and solder across R26.

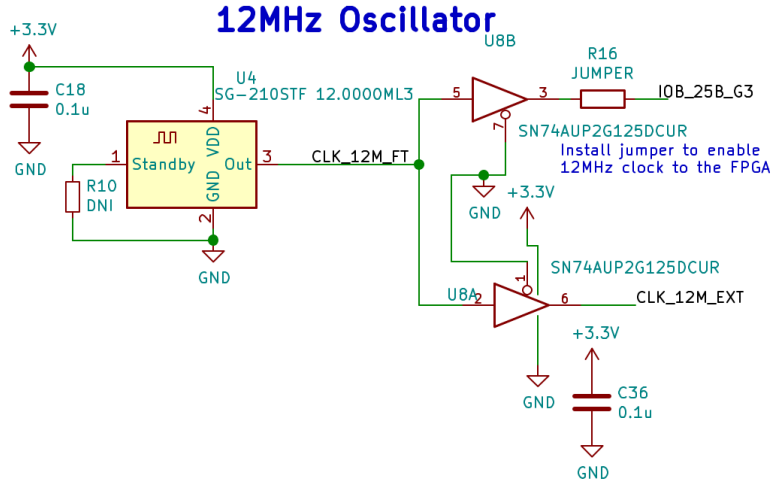


1.1.7 How to use the oscillator options on the UPduino?

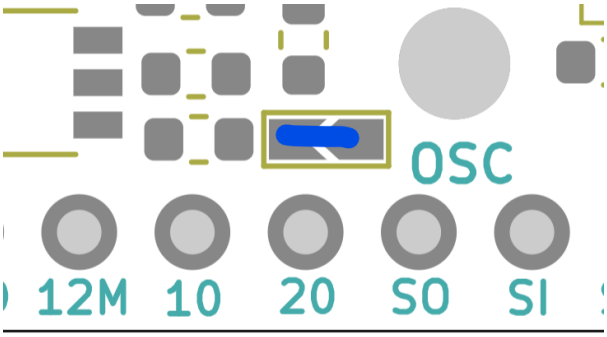
The UPduino has an on-board oscillator that generates 12MHz. This clock is generated by an oscillator and distributed to the FTDI, an external pin and also to a global buffer on the FPGA via an optional jumper.

The pin IOB_25B_G3 was chosen specifically as it is already in a bank thats forced to be at 3.3V levels. If any other global capable IO were to be used, this would force that IO bank to not be capable of the full flexibility of IO voltages.

The schematic portion is captured below:



The 12MHz can be routed to the FPGA directly on the board to preserve signal quality and also to minimize external connection by shorting R16. Note that this marked as OSC on the silkscreen for clarity.



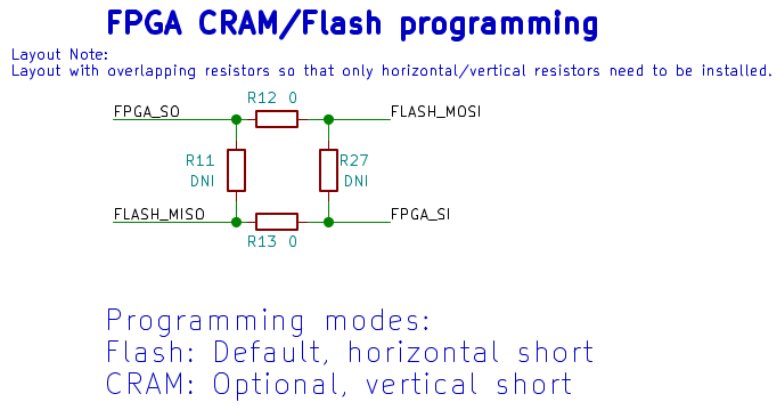
1.1.8 How to program the FPGA CRAM?

The FPGA on the UPduino can be programmed by either programming the flash and letting the FPGA reconfigure itself after a reset (default) or else, program the FPGA under direct control of the FTDI part (CRAM programming).

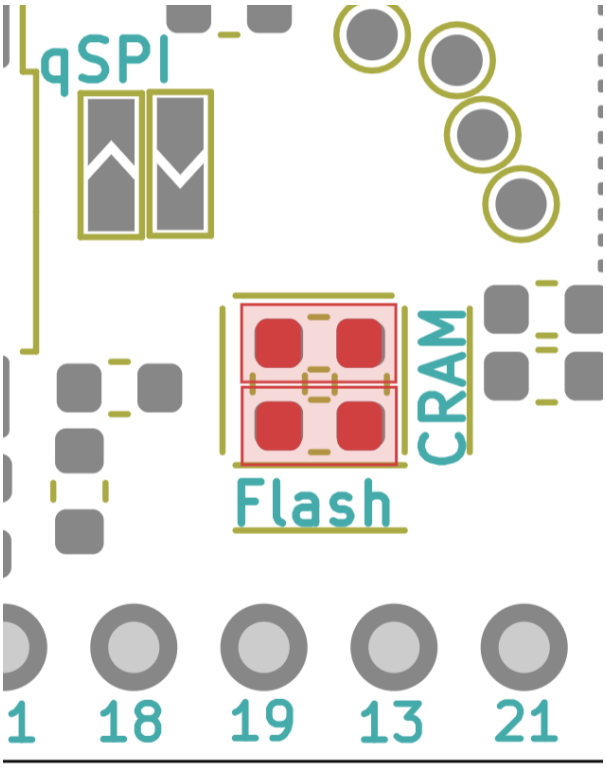
The CRAM in the FPGA is volatile and so will not survive a power down. However, the programming is extremely fast (65ms) ass compared to programming the Flash followed by a reset (>30 seconds). In applications where the FPGA may need to be reconfigured quickly and often such as under control of a processor, it makes sense to use the CRAM mode.

CRAM Programming Mode

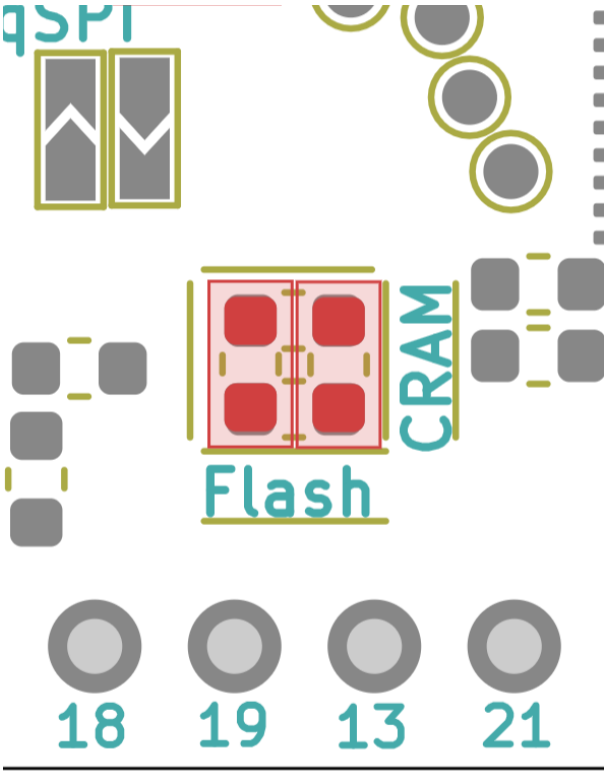
As shown below, the default option is for R12 and R13 to be installed while leaving out R11/R27. To program the CRAM, the user must remove R12/R13 and install R11/R27.



The default (flash) board layout is shown below:



Removing R12/R13 and installing R11/R27 would look like the following layout:

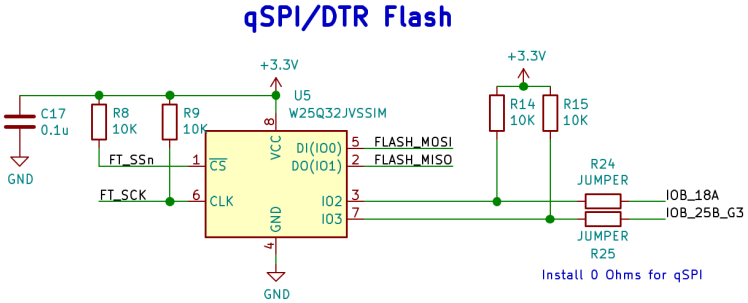


1.1.9 How to enable qSPI flash for _much_ higher flash throughput (up to 8x!)

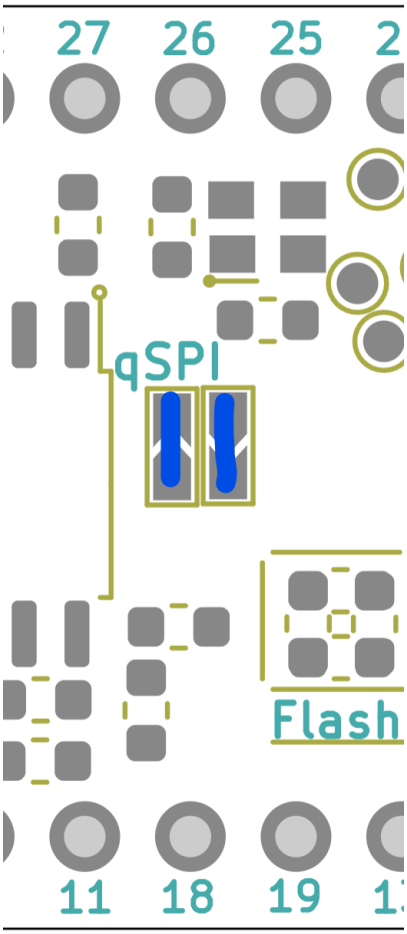
The UPduino flash is a qSPI/DTR capable device. ie. it is capable of operating with four IO's instead of a single IO and also using both edges of the clock, effectively giving 8x the bandwidth on the SPI bus. In an actual use case, this bandwidth increase will only happen for burst read/writes.

Note that this is not the default as the primary requirement was for the UPDuino to be 100% backward compatible. If qSPI mode were made the default, some of the designs that used the two extra pins required for qSPI would not work.

The layout makes special arrangements to allow for this mode of operation of the flash by shorting R24/R25 in the schematic below:



The corresponding portion of the layout is labelled qSPI for clarity and should be shorted as shown.



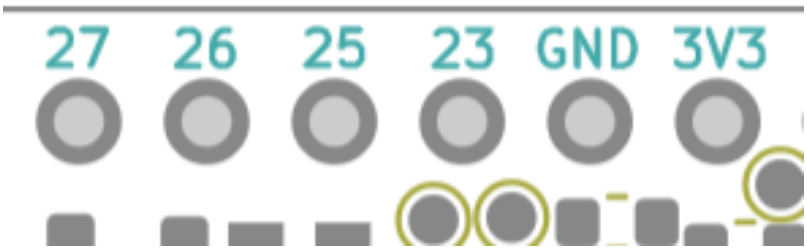
1.1.10 How to enable the tinyFPGA bootloader support in the UPduino?

1.1.11 How to map a RISCV processor into the UPduino?

1.1.12 How to use the UPduino as an OpenOCD debugger?

1.1.13 How to connect a PMOD device to the UPduino?

The UPduino pinout is setup specifically so that connecting to a 3.3V PMOD device is very easy. The pins in the following region of the UPduino are laid out per the PMOD specification allow you to interface directly to any single PMOD peripheral.



1.1.14 How to add a slave select to the FPGA from the FTDI