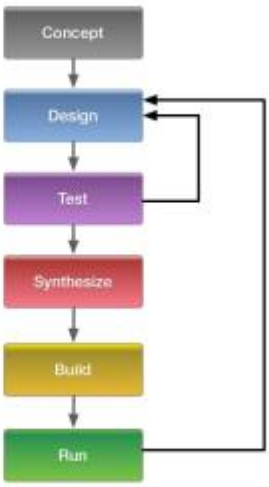
Step 2

Fair warning. You are about to enter a battle zone. This is not easy! It is fraught with peril and can be frustrating. Things worth doing are rarely easy. Some boards are easier to work with than others. Step 2 is where you sort out your particular development environment and document the steps required for someone else to duplicate it.

At all stages, we are going to share our work and help each other out. Radio functions are not mysteries. They are well-studied and can be mastered. Persistence reveals results!

Learn FPGA Design Flow with a Frequency Divider Circuit

Adapted from Chapter 2 of "Make: FPGAs"



Above is a diagram of how things like FPGA design almost always happen.

We begin with a **concept**.
We **design** our implementation.
We **test** the implementation, use what we learn to adjust the design, and then test the result.
We may do this many times!

Once we're satisfied with the result, we **synthesize**, **build**, and **run**.

When we run the completed design, we **test it again**.

It is highly likely that we may find that we need to make additional adjustments to the design, and we have to go back to the **design-and-test** loop. This is normal!

Hardware Hacking Hello World

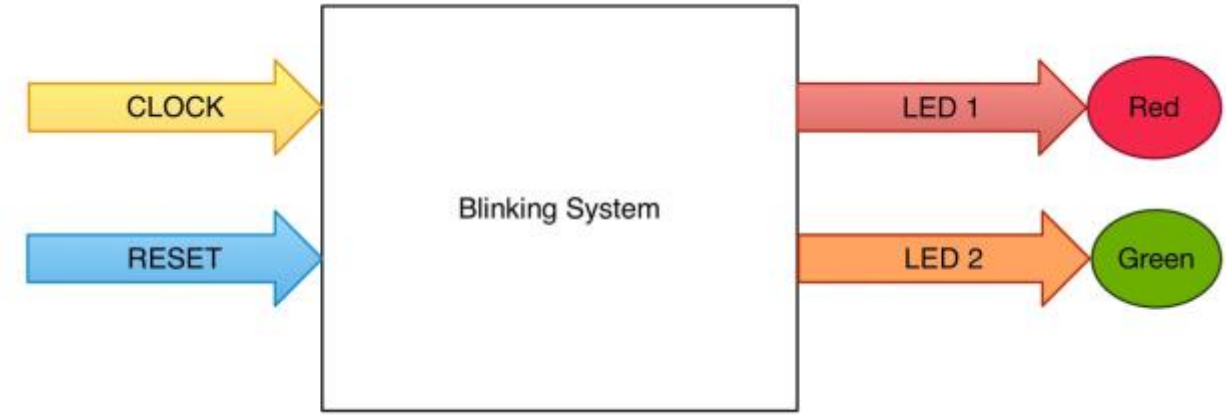
You may already be familiar with the Hello World concept, where the phrase “Hello World” is printed out on a screen or readout. In the software world, “Hello World” is often the first real test that the system can be programmed and is working.

On the hardware side, when you want to test that the system can be programmed and is working, the iconic “Hello World” is blinking an LED.

Extending Hello World

For this step, we’re going to extend the basic “Hello World” of blinking an LED. We are going to combine blinking an LED with dividing down a fast signal to get a slower one. We will also have a signal that overrides the basic behavior.

Here’s a block diagram of what we’re doing.



Our inputs are system clock (CLOCK) and reset (RESET). When documenting work like this in drawings, inputs generally come in from the left and flow to the right. Our outputs are signals that blink two LEDs (LED 1 and LED 2). Outputs also flow from left to right. Blocks are used to abstract functions or collections of functions. High level block diagrams are the simplest and most abstracted form of documentation. Each block can be replaced with a more detailed, or lower level, block diagram.

We are going to make the LEDs blink at different rates. The output blinking rates we are going to produce with the Blinking System are much lower than the system clock. If we used the system clock directly to blink LEDs, the LEDs would blink at a rate much faster than what we could physically see. We have to divide down the system clock rate to get LED blink rates that can be seen with human eyes. Being able to control the LED behavior with a reset signal is an important concept. Signals from user interface and sensors need to be successfully incorporated into the system design all along the process. We’re beginning with a very simple

reset signal, but this control signal concept will be extended to numerous signals for a working radio.

Select Your FPGA Board

In order to do this step with physical hardware, **you will have to choose some sort of evaluation board or development module or testbed. You need an FPGA that can be programmed.**

For Phase 4 Ground, we are going to be using a variety of SDRs, and even make our own!

Do you have an SDR with an FPGA that Vivado can talk to? Great! If you need one, then this is your chance to go get one!

You can tell whoever you need to that Michelle said it was ok!

Vivado talks to mid-scale and up Xilinx parts like Ultrascale, Virtex-7, Kintex-7, Artix-7, and Zynq-7000 series. If you have a board with any of these parts, then Vivado is what you need. It's what you installed in Step 1.

If you want to use smaller Xilinx parts, like the Spartan-3, Spartan-6, Virtex-4, Virtex-5, and Virtex-6 families, then you will need the (discontinued) Xilinx ISE. Xilinx ISE is in "sustaining" mode, which means no new versions will be released, but you can still install it and use it.

We strongly recommend Vivado. It's the current version of the toolchain from Xilinx. The chips supported by Vivado are what we're going to be dealing with.

Different boards may need different versions of Vivado. That's ok. Install whatever is called out for your board. Don't fight it, just install it.

For example, let's look at the Red Pitaya.

http://pavel-demin.github.io/red-pitaya-notes/led-blinker/

These are a set of notes to get the Red Pitaya cooperating with Vivado, and also blinking an LED. Do you have a Red Pitaya? Then the link above is a great start!

If you're using something like a USRP x310, then according to the Ettus website, Xilinx Vivado 2015.2 Design Suite is what you'll need. Don't bang your head against a wall. If something blows up, back off and double-check.

Getting things set up for development can be hard. Tribal lore, unclear directions, things that change out from under you – all of this and more is considered to be part of the embedded development landscape.

This is not an excuse. Difficult or badly designed environments should not be normal or put up with without complaint. However, dealing with the innards of an FPGA is not the same as firing up a word processor and printing off a document. With great power comes great responsibility and almost always a steep learning curve. A good attitude (and sympathetic co-conspirators) is irreplaceable!

Get an LED Blinking

This section will be updated as people document their recipes.

What Did We Accomplish?

We chose a board.
We got it working with Vivado.
We blinked an LED.
We implemented an LED **Blinking System**.

- We learned how to divide down the system clock to get useful human-rate signals.
- We added a control signal.

Next Step: Learn about Concurrency with a Digital Clock