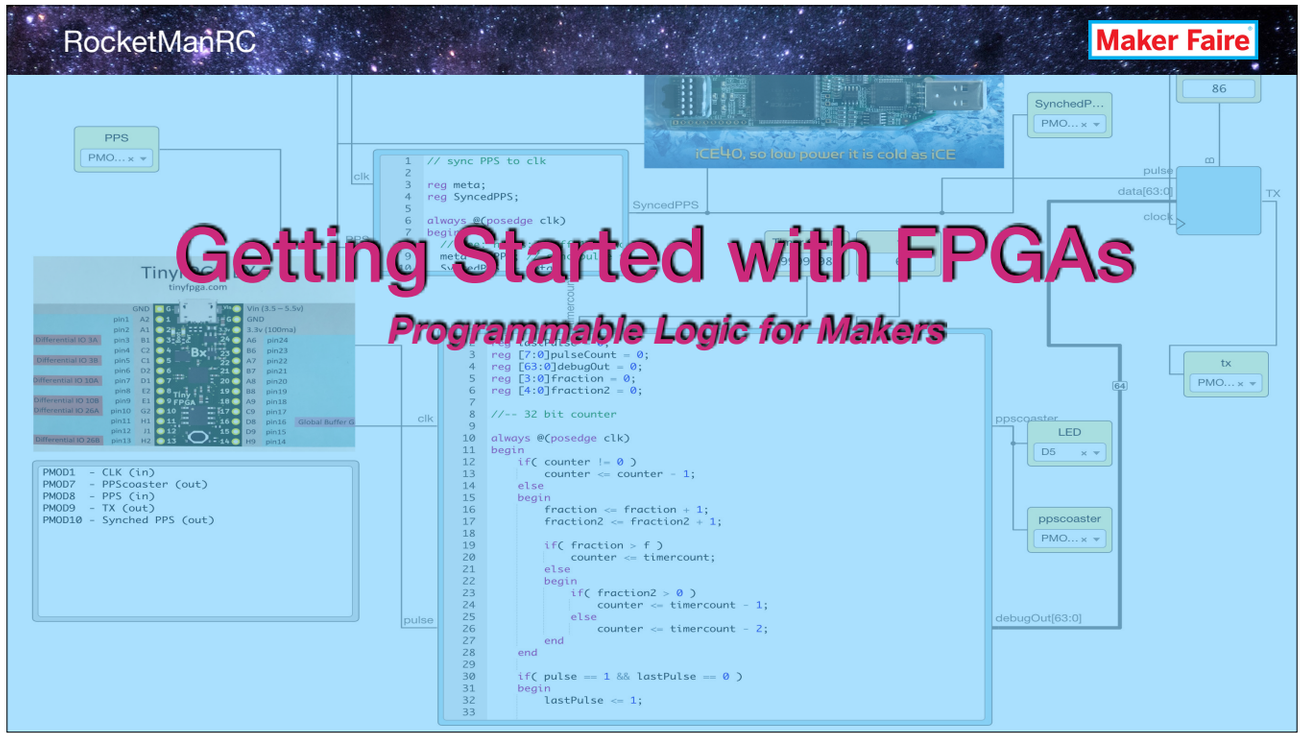

RocketManRC
Maker Faire
iCE40, so low power it is cold as iCE
Getting Started with FPGAs
Programmable Logic for Makers
PPS
PMO...
clk
SyncedPPS
SyncedP...
86
pulse
data[63:0]
clock
TX
tx
64
LED
D5
ppscoaster
debugOut[63:0]
tinyfpga.com
PMOD1 - CLK (in)
PMOD7 - PPScoaster (out)
PMOD8 - PPS (in)
PMOD9 - TX (out)
PMOD10 - Synched PPS (out)

Presentation Objective

In this presentation I will show you how to get started with FPGAs and how I managed to accomplish that using the open source icestudio and IceStorm development tools and the TinyFPGA and iCEstick development boards. This is a powerful solution for any electronics Maker!

Rick MacDonald

http://www.rocketmanrc.com

About Me

- I had a 40 year career in several industries including Aerospace a couple of times.
- My technical area of expertise is embedded systems, both hardware and software.
- I live in Atlantic Canada where we also have unpredictable weather like here!
- For the past year and a half I have been privileged to be a full time Maker, but I have always been an electronics hobbyist.
- I like to say that I am on “permanent sabbatical”!

Background

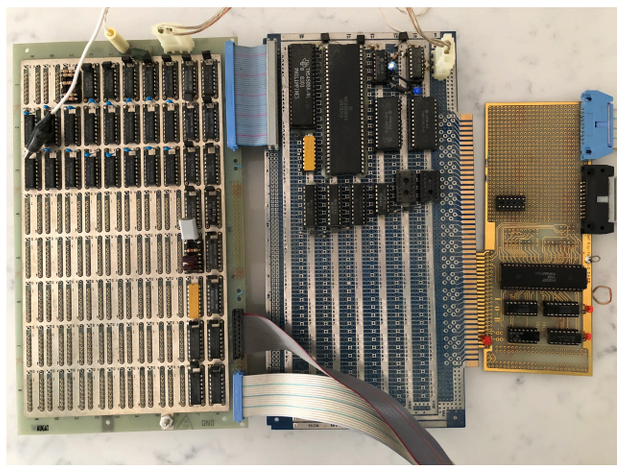
- Last year I started a project that I call OpenPPS the intent of which was to build a highly precise timing device that could be used for applications like time synchronization, frequency counting and pulse width measurement.
- I originally hoped to do this with just an EPS32 board and a GPS module but from the beginning I wasn't happy with the performance results.
- I started using the iCEstick FPGA Evaluation Board (by Lattice) in a very simple way to try and diagnose the performance issues.
- I ended up using the iCEstick and also the TinyFPGA to replace all the timing logic in order to get the performance I wanted.

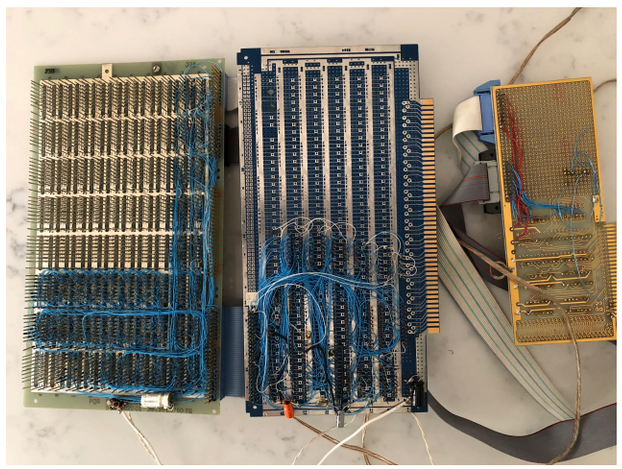
I hope everyone knows what the ESP32 is?

What is an FPGA?

- FPGA stands for “Field Programmable Gate Array”.
- It is essentially a circuit board on a chip that you program instead of changing wires.
- FPGAs have been available since the early 1990s but have never been very accessible to hobbyists.
- I saw the TinyFPGA at the Maker Faire last year which made me very interested in learning how to program one. I was starting from scratch!

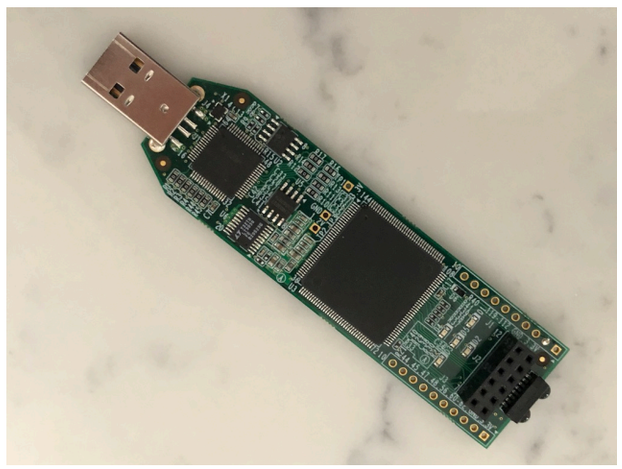
The old way for electronics projects…

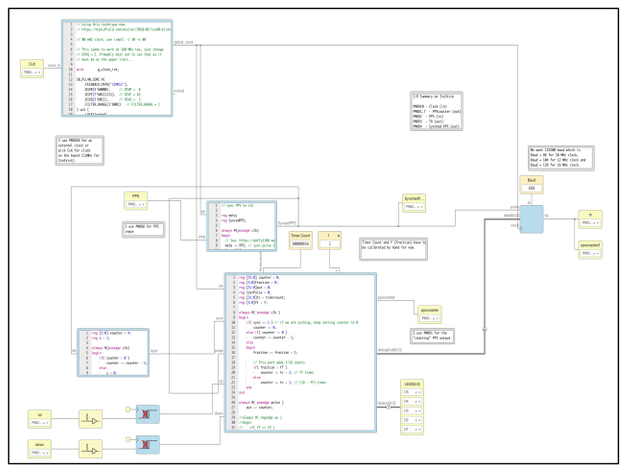




I build this in the 1980s. It is a home brew 68000 based computer.

The new way!





The iCEstick - available from Digi-Key of course!

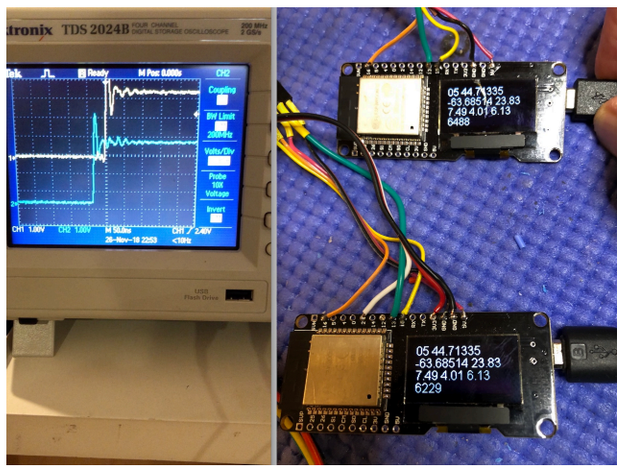
Could even put a processor on this FPGA!

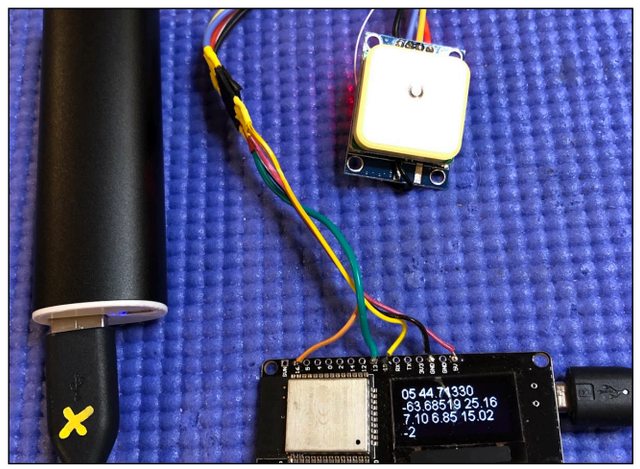
OpenPPS Project

- A GPS module provides a digital pulse-per-second (PPS) that is synchronized with the atomic clocks in the GPS satellites. The pulse starts at the beginning of the second.
- The estimated accuracy compared to the atomic clocks in the satellites can be as good as 10 ns (spec is 40).
- The ESP32 has four 64-bit counters with selectable clock rates up to 40 MHz.
- I wanted to synchronize an ESP32 counter to the PPS using an interrupt on an I/O pin to calibrate it and then after that just use the counter to generate the pulses.
- A simple concept but alas a major fail!

I have another talk about this at 5 pm

PPS from two GPS and ESP32 Setup





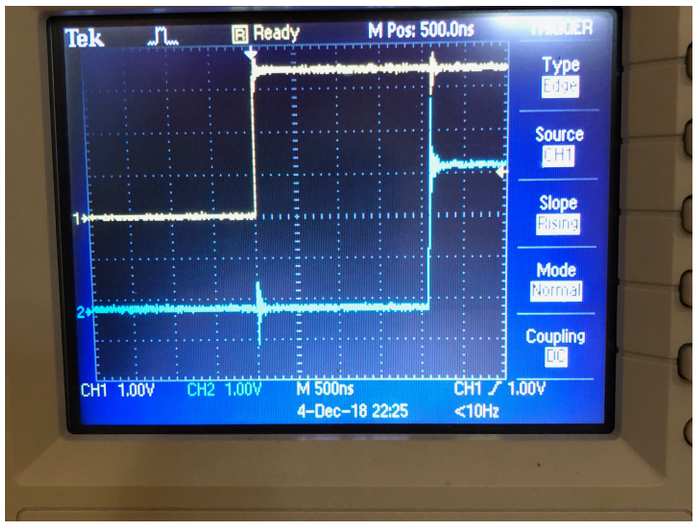
The PPS from two GPS stay within 50 ns of each other even during very degraded satellite reception.

The Initial Problem

- I had a very difficult time calibrating the ESP32 counter and was getting very inconsistent results.
- Calibrating means measuring the number of counts between one PPS and the next. I used a 10 MHz clock so each count was 100 ns.
- I was expecting to calibrate to at least 10 counts (1 us).
- I hadn't given much thought to the ESP32's interrupt handling as it has a 240 MHz CPU clock.
- By toggling an ESP32 I/O pin in the interrupt handler and using my trusty old Tektronix oscilloscope I could see that the interrupt response was at least 3 us and varied!

Tried to diagnose the problem with the oscilloscope.

ESP32 Interrupt Response

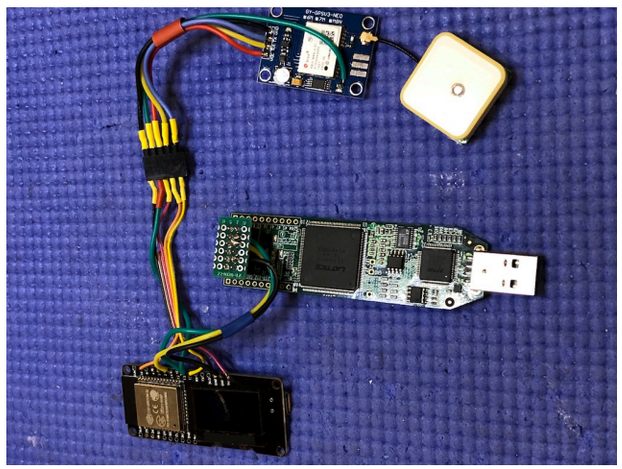


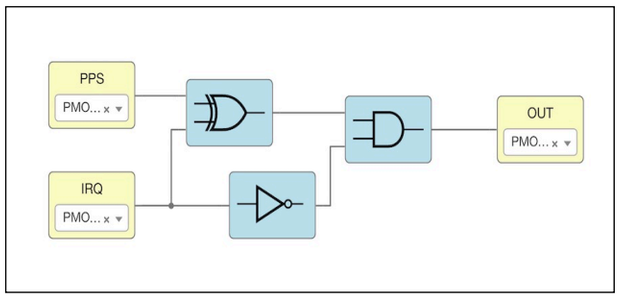
2.2 us average but it jumps around a lot

Investigating

- I really wanted to measure the ESP32's interrupt response time and variation (jitter) and see if I could do something to improve it.
- To measure using the 'scope I needed to create a pulse that started at the beginning of PPS and ended when the ESP32 toggled the output port at the beginning of the interrupt handler.
- It seemed that I was going to have to build some kind of digital logic circuit to do that.
- I had been experimenting with the iCEstick and IceStudio (running the examples and blinking LEDs) and the iCEstick was sitting on my bench.
- In 5 minutes I had the test circuit implemented!

iCEstick with ESP32 and Logic in IceStudio

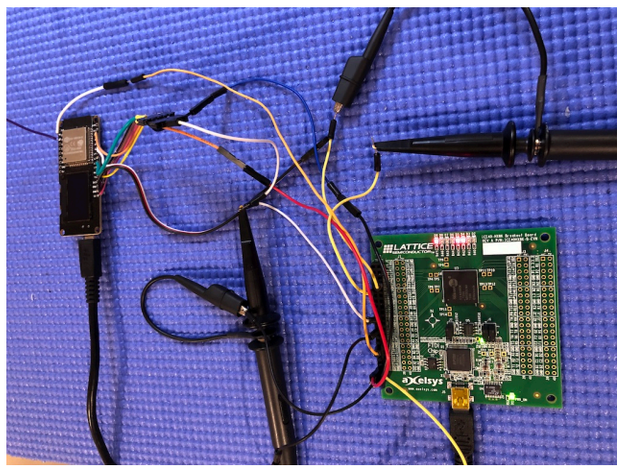


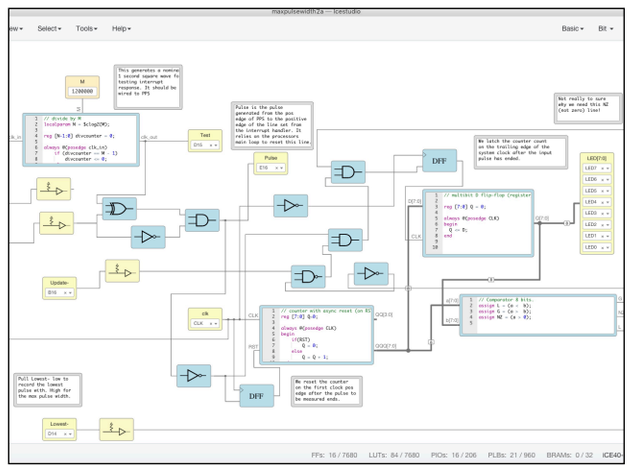


Pulse Width Measurement

- Measuring pulse width on the ‘scope is easy but measuring the variation isn’t.
- I tried to set a trigger on maximum pulse width exceeded and this sort of worked.
- Why not use the iCEstick to measure the pulse width and variation too?
- The iCEstick has 5 LEDs arranged in a star shape which is not the greatest way to display measurement data.
- I also had the Lattice iCE40-HX8K Breakout Board which has 8 LEDs in a row. Perfect!

iCE40-HX8K Breakout Board and icestudio code



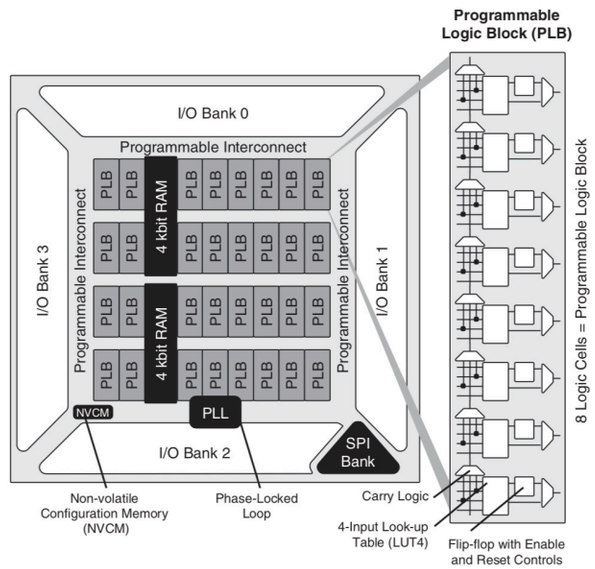


The FPGA code in icestudio starting to get a bit complex!

iCE40 & IceStorm

- Why Lattice iCE40 FPGAs? Open Source development tools and quite a good selection of supported boards so far!
- IceStorm is a set of tools that turns Verilog source code into a bitstream to load into the iCE40.
- Created by Clifford Wolf an Electronics Engineer and Professor in Vienna. First released in 2015.
- Developed by reverse engineering the iCE40 bitstream generated from the Lattice iCEcube development tool.
- The iCE40 was chosen because “It has a very minimalistic architecture with a very regular structure. There are not many different kinds of tiles or special function units.” There are also quite a good selection of devices available.

iCE40 Architecture



Essentially an array of programmable logic blocks that are configured by “programming”.

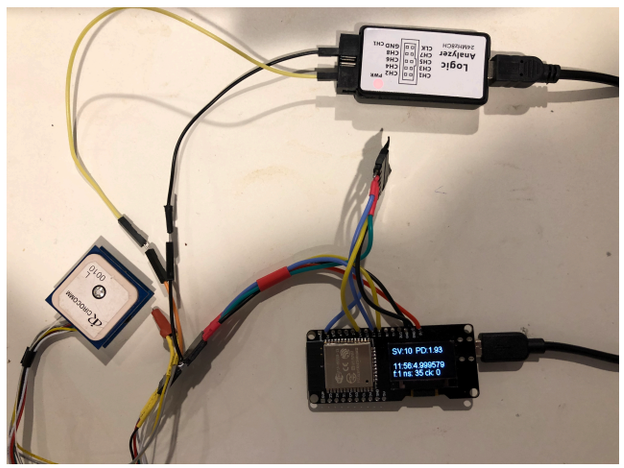
icestudio

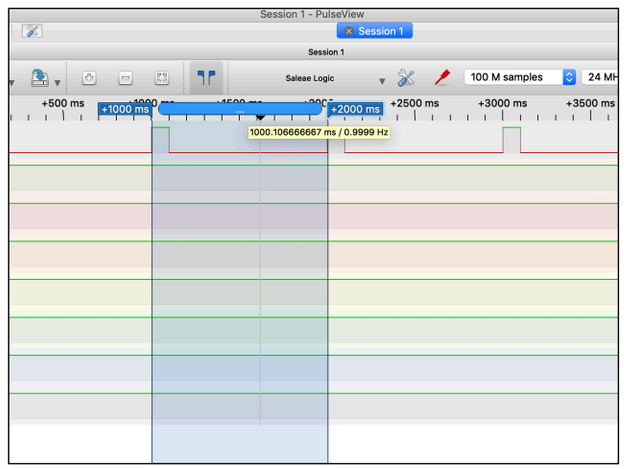
- Having to first learn Verilog to get started developing is a big mountain to climb!
- Icestudio is a graphical dataflow based development tool that builds on top of IceStorm.
- Simple logic elements can be graphically placed on a canvas and “wired” together.
- You can inspect the Verilog code for every existing logic element.
- Code blocks can be added for Verilog code making it easy to get started with Verilog.
- All of the IceStorm tools are run “under the hood”.

icestudio example

I am going to quickly show how to use icestudio and a $10 logic analyzer to create the test setup

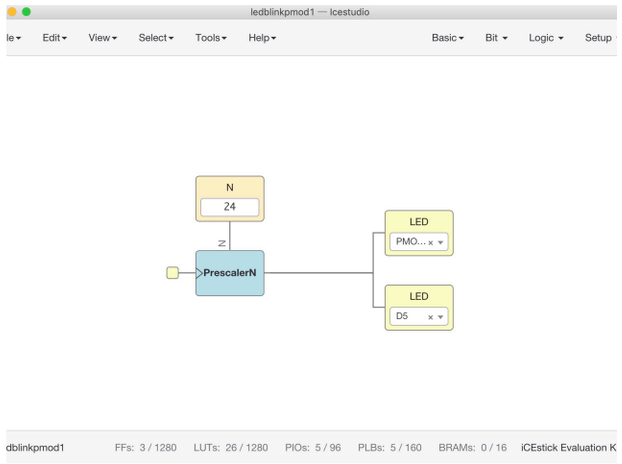
Logic Analyzer and PPS

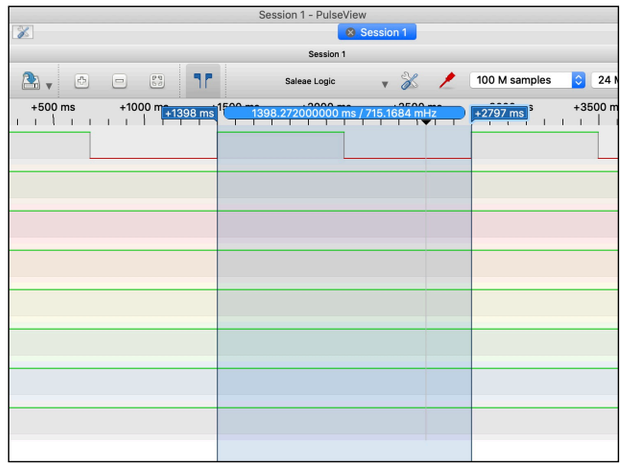




Why it is not exactly 1 second is due to the clock in the logic analyzer. But it’s good enough for this.

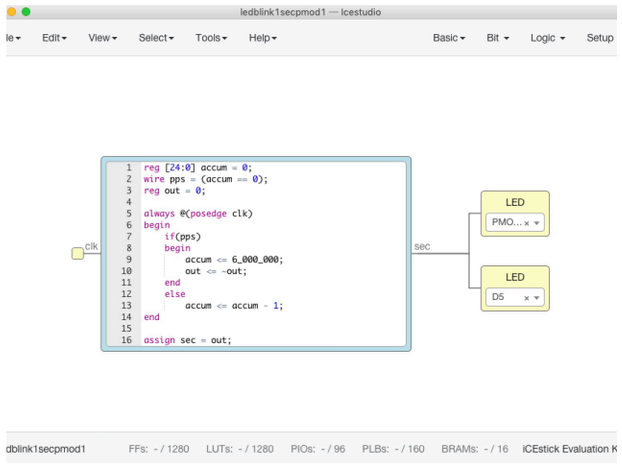
Let’s replace the GPS with a pulse generator in the iCEstick



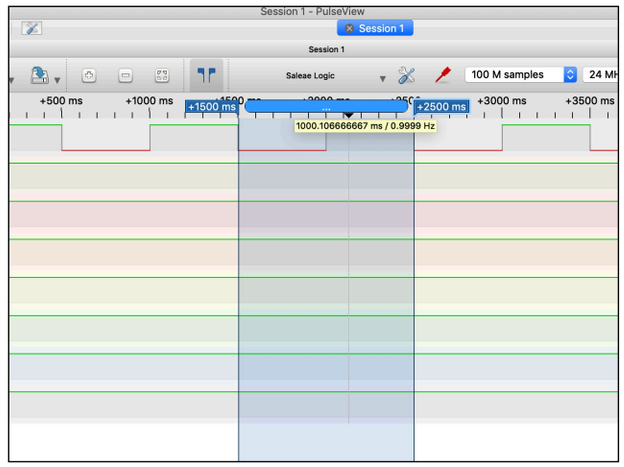


We don’t really need the GPS to measure the ESP32 response time so we create a test pulse.

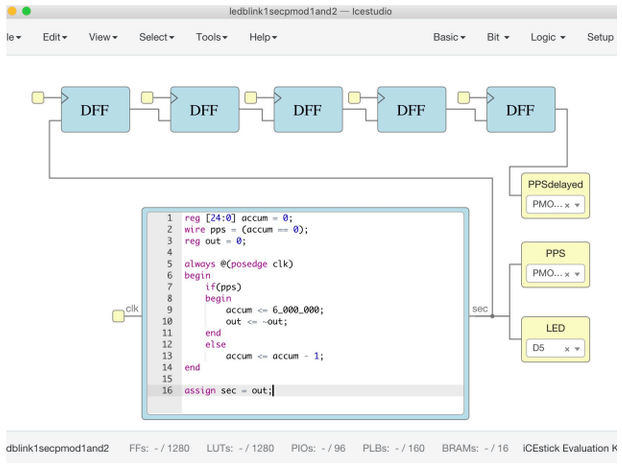
Make it closer to 1 second with some Verilog

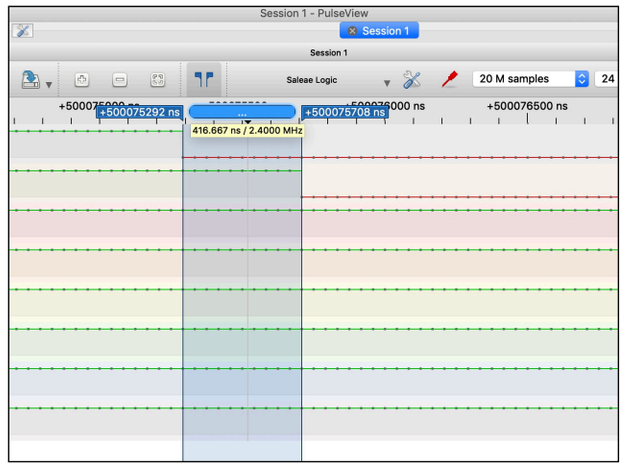



```
1  reg [24:0] accum = 0;
2  wire pps = (accum == 0);
3  reg out = 0;
4
5  always @(posedge clk)
6  begin
7      if(pps)
8      begin
9          accum <= 6_000_000;
10         out <= ~out;
11     end
12     else
13         accum <= accum - 1;
14 end
15
16 assign sec = out;
```



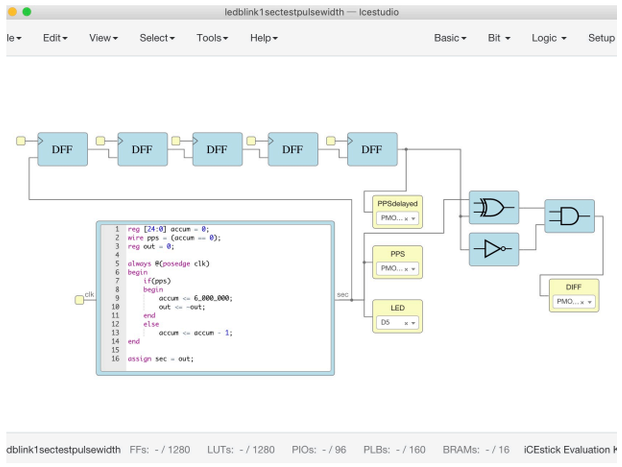
Add a delayed pulse output with some flip-flops

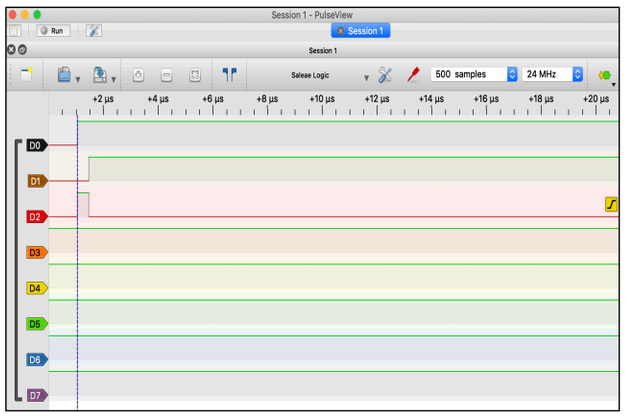




This simulates the ESP32 interrupt response so we know we have our setup correct.

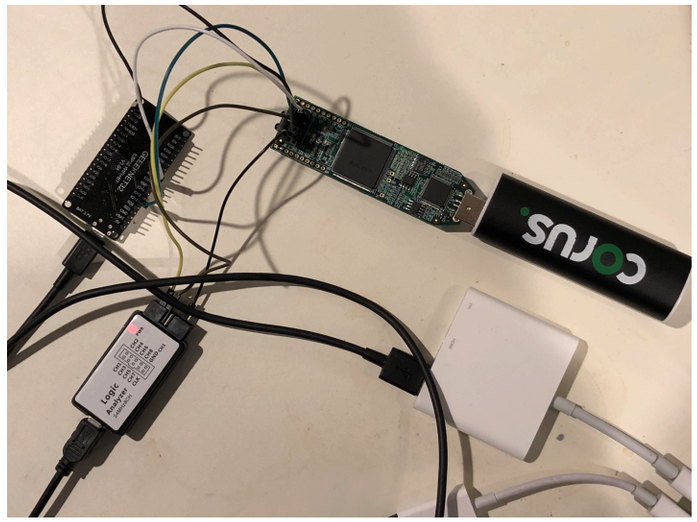
Create a single pulse from the difference





That’s the simple logic I showed you before.

Add the ESP32

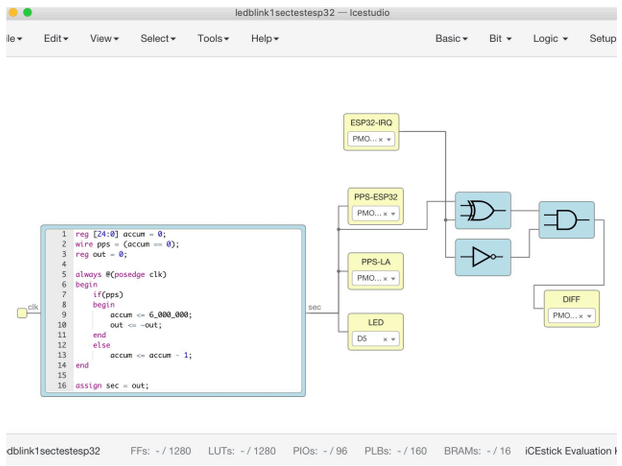


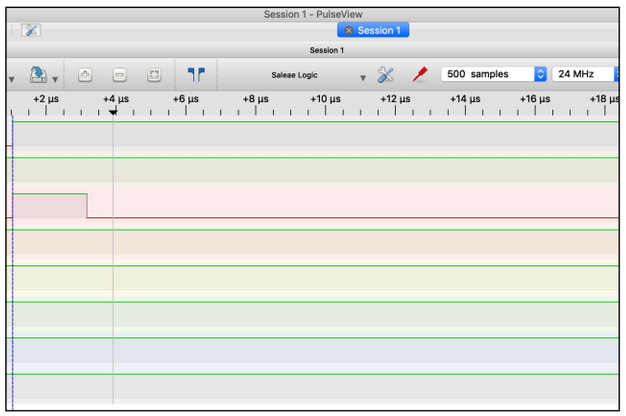
A bit of code

```
// PPS – Digital Event Interrupt
// Enters on rising edge
//=========================================
void IRAM_ATTR handleInterrupt()
{
  REG_WRITE( GPIO_OUT_W1TS_REG, BIT25 );  // NOTE if IRQpin changed have to edit this!

  irqCount++;
}

void setupPPS()
{
  pinMode( irqPin, OUTPUT ); // pin25 will be used as an output for testing PPS interrupt response

  // PPS setup
  pinMode(interruptPin, INPUT_PULLUP);                                      // sets pin high
  attachInterrupt(digitalPinToInterrupt(interruptPin), handleInterrupt, RISING); // attaches pin to interrupt on Rising Edge
}

void setup()
{
  setupPPS();
}

void loop()
{
  static uint32_t lastIrqCount = irqCount;

  // Set the IRQ pin low again when PPS goes low
  if( (irqCount != lastIrqCount) && !digitalRead( interruptPin ) )
  {
    lastIrqCount = irqCount;

    digitalWrite( irqPin, 0 );
  }
}
```

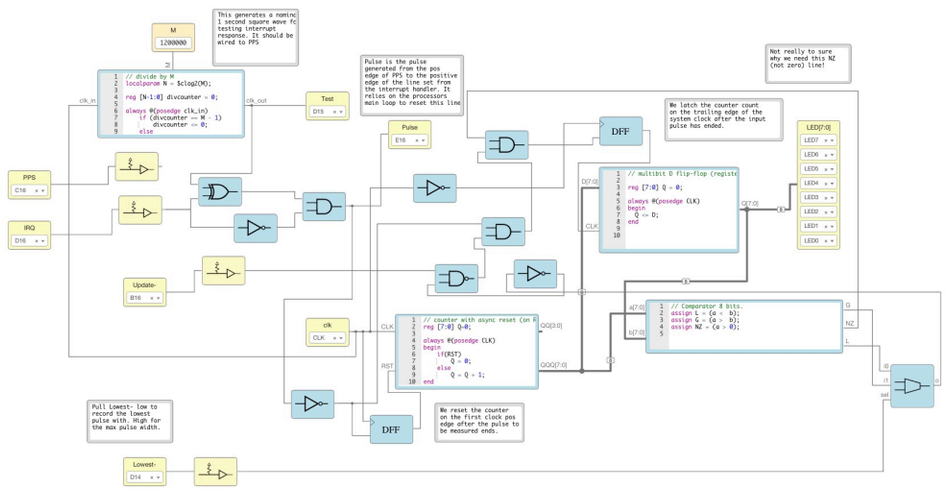
Some simple test code for the ESP32.

Measure the ESP32 interrupt response!

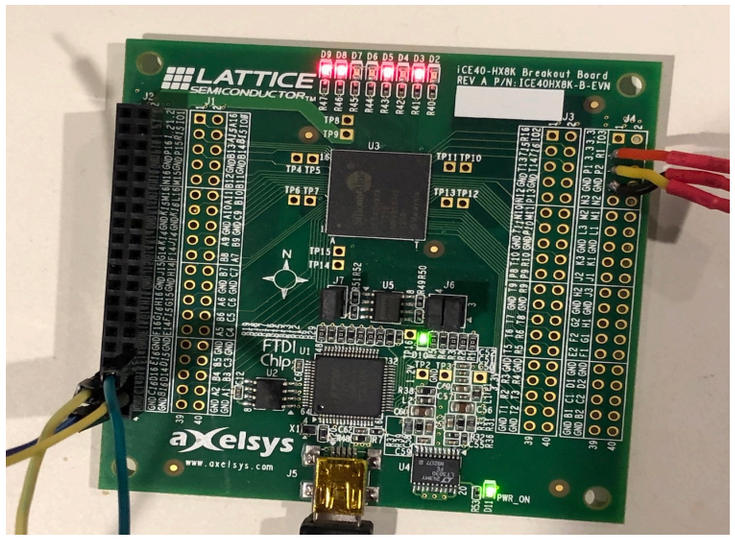




Substitute the iCE40-HX8K for the iCEstick and logic analyzer



Max interrupt response, 83 counts = 6.9 us!



A note about FPGA “hell”

- This is something that you really need to know about when interfacing digital logic to external signals.
- FPGA “hell” is when everything should work but it doesn’t!
- In my case I was missing GPS pulses which ended up being due to metastability which happens when clocks aren’t sychronized (in this case the FPGA clock and the GPS).

Conclusions

- Implementing digital logic with FPGAs is a challenge but a powerful tool.
- Icestudio and IceStorm are a gentle way to get into it.
- Transitioning to Verilog can be learn by doing.
- There is lots of sample Verilog code available to use and to learn from.
- Boards based on the iCE40 are readily available and affordable.

Thanks to Digi-Key for sponsoring this stage!

Questions?