Master thesis and internship[BR]- Master's thesis : Design and Implementation of a Low-Cost Multi-channel Coherent Software Defined Radio Receiver for Educational and Amateur Radio Use[BR]- Integration internship

Auteur : Rouma, Alexandre
Promoteur(s) : Redouté, Jean-Michel
Faculté : Faculté des Sciences appliquées
Diplôme : Master en ingénieur civil en aérospatiale, à finalité spécialisée en "aerospace engineering"
Année académique : 2024-2025
URI/URL : http://hdl.handle.net/2268.2/23363

Design and Implementation of a Low-Cost Multi-channel Coherent Software Defined Radio Receiver for Educational and Amateur Radio Use

ALEXANDRE ROUMA

UNIVERSITY OF LIÈGE
SCHOOL OF ENGINEERING AND COMPUTER SCIENCE

MASTER'S THESIS COMPLETED IN ORDER TO OBTAIN THE DEGREE OF:

MASTER OF SCIENCE IN AEROSPACE ENGINEERING

Abstract: Current coherent software defined radio receivers, although on a path towards affordability for amateur radio and educational use, still have major down sides such as low bandwidth, hard to calibrate architectures, low channel count or high prices. Reviewing the state of the art in the field reveals that reusing mass produced commodity hardware by working around its intrinsic limitations is a viable path towards an affordable, high channel count, medium bandwidth coherent receiver. This masters thesis investigates the design of such a receiver using tuner ICs originally meant for low cost digital television receivers and a recently released high channel count ADC. Due to each tuner having its own local oscillator and phase locked loop, an onboard phase reference is added with the option of switching it into the receive path for auto-calibration. Before the full receiver is assembled, various subsystems are tested in advance to make sure they will perform adequately. The results shows that the phase reference splitter required for auto-calibration is able to maintain under 1% phase difference between its outputs over the entire frequency range of the receiver. It is also demonstrated that the power supply has the required accuracy and low ripple noise. Some mistakes were found in the design such as an incorrect capacitor in the power supply, an incorrect resistors in the tuner supporting circuitry and incorrect supply sequencing for the clock generator but all issues were easily resolved. Single channel reception was demonstrated showing that each channel on its own is a capable receiver with good sensitivity. Due to time constraints, mutli-channel coherent reception has not yet been demonstrated, but the currently obtained results do not indicate anything that would preclude the full system from working as intented.

THESIS SUPERVISOR: PR. JEAN-MICHEL REDOUTÉ

ACADEMIC YEAR 2024 - 2025

Contents

1 Introduction

This report describes the process of designing and testing a low cost 8 channel coherent SDR receiver for use by hobbyists and education institutions in applications such as radio astronomy, beamforming experiments and satellite reception.

First, clear goals are set in order for the designed hardware to be a useful addition to the currently available software defined radio tool set. This is done by requiring that the device designed for this thesis project has at least equal or better performance to to the current cost leader. Additionally, the total price of the device must lower than the current cost leader.

Following this, the state of the art in terms of affordable coherent receivers is reviewed. A huge emphasis is set on "low-cost" as, like will be demonstrated, getting a receivers with many coherent channels is not difficult is a high budget is available. Four different devices each corresponding to a specific RF architecture are reviewed before being compared to each other.

With a clear idea of the goals and the current implementation, a hardware design is drafted and described in detail. The design process is broken up into a series of subsystems, the design and relation of which will be detailed sequentially. For each subsystem, both the schematic of the circuit and the PCB layout are described and discussed.

Next, since the hardware design will use an FPGA, the internal logic design is drafted and described. Again, the description of the design is done on a per module basis describing the relation of each module to the other through its interfaces. In the case of modules implementing a finite state machine, state diagrams will be draw up and discussed.

The FPGA design will include a soft CPU core, therefore its firmware will also be described and discussed. The description of said firmware will start by describing the initialisation procedure followed by the runtime command handling.

Finally, with the hardware, firmware and software discussed, the tests done on each system will be detailed and discussed. Specifically, the discussion will be broken up in 3 to 4 parts. First, if a prototype outside of the full receiver prototype has been created, its design is discussed. Next, the testing methodology and expected results are described. If any issue was discovered during testing, a dedicated subsection will be included to discuss them and the remediation measures that were taken. Finally, the results of the test after issues were corrected are exposed and discussed.

1.1 Goals

As will be exemplified in the state of the art review, current educational and/or amateur radio grade coherent receivers are relatively expensive, only support a low number of channels and, in some cases, are not actually truly coherent out of the box. This has contributed to a situation where coherent reception is still relatively uncommon in amateur radio, and thus, means that no developments are made on the technology in the amateur space. Historically, amateur radio has been on the forefront of radio communications innovation, but without the right tools, this cannot be extended to coherent reception.

Parameter	Minimum	Maximum	Description
Channels	6	-	At least more channels than the current cost-per-channel leader.
Frequency range	144 MHz	440 MHz	Must cover at least 2-meter and 70-centimeter bands.
Bandwidth	2.4 MHz	-	Must have at least the same bandwidth as a RTL-SDR.
Bit-depth	8 bits	-	Must have at least the same bitt depth as the RTL-SDR.
Price	-	$499	Must be cheaper than the current cost-per-channel leader.

Table 1: Goals set for this thesis project in terms of the receiver's performance and cost.

In order for a new coherent receiver design to be successful in the amateur radio and educational space, several conditions have to be met. First, it must support a frequendcy range which allows experimentation on amateur radio bands such as the 2-meter band and the 70-centimeter band. Second, it must have at least as much bandwidth as low-end SDR receivers such as the RTL-SDR. Indeed, anything less than that makes browsing for signals in the area increasingly cumbersome as the user is only able to see a tiny swath of bandwidth at a time. Finally, and most importantly, it must be affordable. As will be shown later, current coherent receivers, even witth only two channels easily cost over $400, and models with even more channels easily get into the tens of thousands of dollars. With a clear understanding of what would make a good amateur radio or educational coherent receiver, it is now possible to put these traits into numbers as is done in table 1.

2 Background

This section investigates the state of the art in the domain of coherent software defined radio receivers, with an emphasis on those designed to be low-cost enough for educational and amateur radio use. Since there exist many different architectures of software defined radios which all have their advantages and disadvantages indepedently of their coherent capabilities, an example was chosen for each of the common SDR receiver architecture.

The LimeSDR was chosen as an example of receiver using an integrated IQ transceiver RFIC, the AMD RFSoC was chosen as an example of a direct sampling integrated transceiver, the USRP x310 was chosen as an example of a discrete IQ transceiver, and finally the KrakenSDR was chosen as an example of a discrete Low-IF receiver.

Finally, all the devices mentioned above and some more are compared to each other and conclusions are drawn to be used in the design of the receiver described in this report.

2.1 LimeSDR

Figure 1: Picture of the LimeSDR Software Defined Radio receiver.

The LimeSDR[18] as shown in figure 1 is a Software Defined Radio tranceiver based on, and designed as the main development board for, the LMS7002[15] RF Integrated Circuit (RFIC). It is capable of receiving and transmitting two 51 MHz channels coherently. A block diagram of the LimeSDR is shown in figure 2.

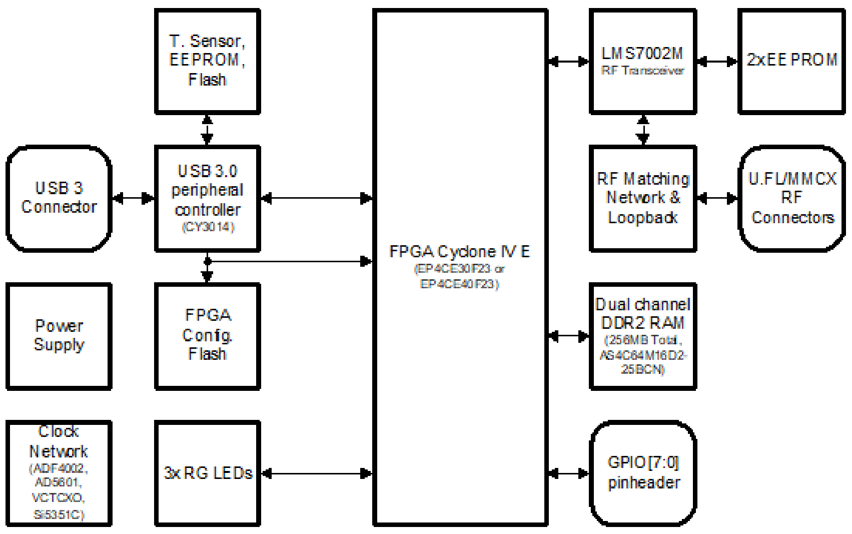

T. Sensor, EEPROM, Flash
USB 3 Connector
USB 3.0 peripheral controller (CY3014)
Power Supply
FPGA Config. Flash
Clock Network (ADF4002, AD5601, VCTCXO, Si5351C)
3x RG LEDs
FPGA Cyclone IV E (EP4CE30F23 or EP4CE40F23)
LMS7002M RF Transceiver
2xEEPROM
RF Matching Network & Loopback
U.FL/MMCX RF Connectors
Dual channel DDR2 RAM (256MB Total, AS4C64M16D2-25BCN)
GPIO[7:0] pinheader

Figure 2: Block diagram of the LimeSDR Software Defined Radio receiver.

At the heart of the LimeSDR lies the LMS7002 RFIC. As shown in figure 3, RF entering the RFIC first passes through variable amplifiers before heaving to quadrature mixers. Following this, the now quadrature IF passes through another set of amplifiers and then passed through programmable low pass filters which set the receive bandwidth. Finally, the filtered IF passes through one more stage of variable amplifier before finally being sampled. The transmit path is essentially the same concept except in reverse. The fact this design is coherent comes from that fact that both channel's quadrature mixers are fed from the same local oscillator.

Both the transmit and receive Local Oscillators (LOs) are generated on-chip from a selectable reference input and can be tuned from 100 kHz to 3.8 GHz. The LMS7002 also features a built-in 8051 microcontroller capable of fully controlling the state of the transceiver outside of the actual baseband data streams. Those datastream go through a separate programmable Digital Signal Processor called the Transceiver Signal Processor (TSP).

In conclusion, this RFIC contains every part needed to create a dual channel coherent software defined radio up to the actual signal demodulation. This level of integration allows the LimeSDR to be not only remarkably compact, but most importantly to be relatively cheap at only $350 compared to a discrete design like the USRP x310 described later.

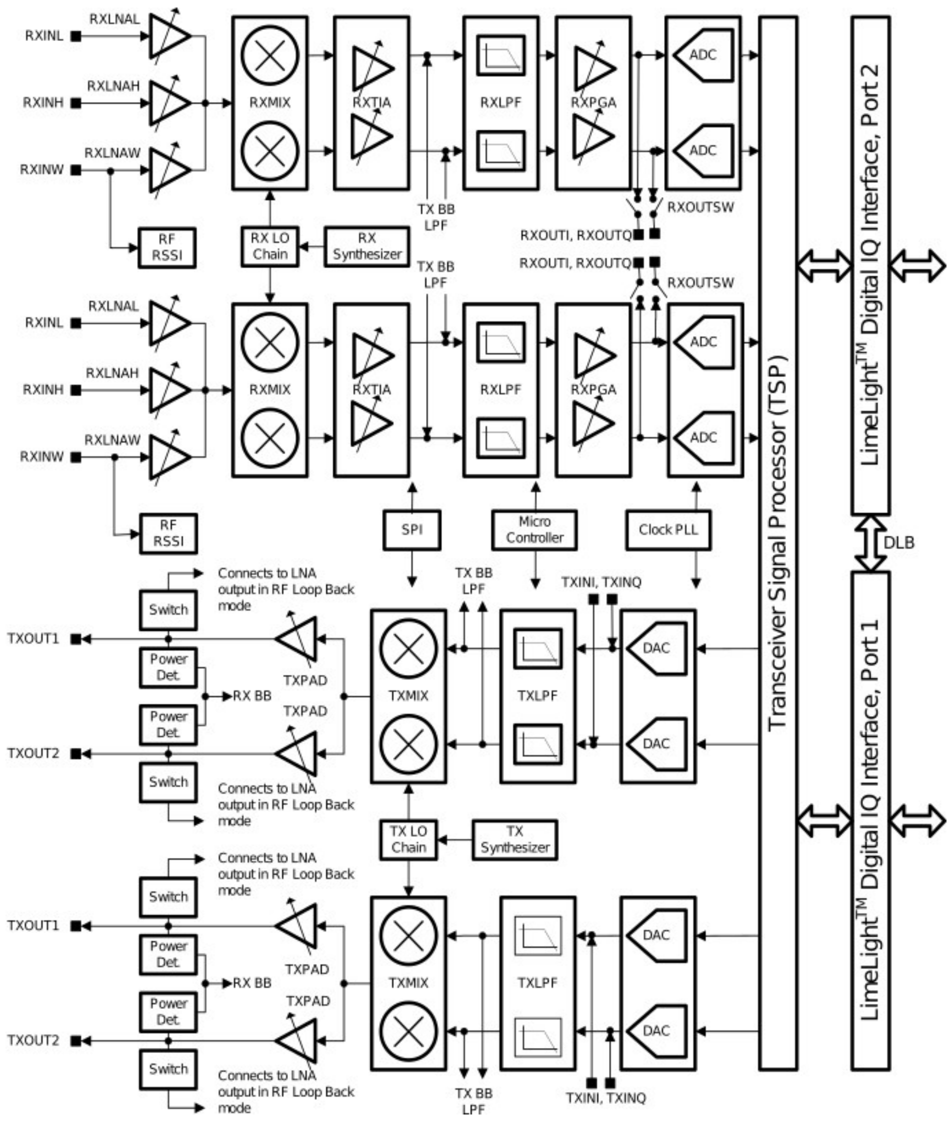


Figure 3: Block diagram of the LMS7002 RF Tranceiver IC.

In order to be processed, the digital baseband then needs to be transfered to the host computer. For this, the LimeSDR uses an Altera Cyclone IV and a Cypress FX3 USB 3.0 interface IC. The FPGA is responsible for acting as glue logic between the USB IC and the LMS7002 RFIC as well as responding to control commands from the PC. While the Cypress FX3 is easy to integrate into designs for streaming data from a device to a PC, it has well known reliability issues[19] which have, by extension, been plaguing the LimeSDR since its release. Finally, clocking of the entire board is achieved via a SiLabs Si5351c clock generator IC using a TCXO as its reference.

While this design is relatively cheap and simple, it has one major downside: The RFIC costs over $80 and only has two receive channels. While two channel MIMO is generally enough for diversity reception[29], more directional or complex antenna patterns are not possible. Adding more channel would require adding more of the same RFIC and using an external phase reference signal as will be done later with much cheaper RFIC in this thesis.

2.2 AMD RFSoC

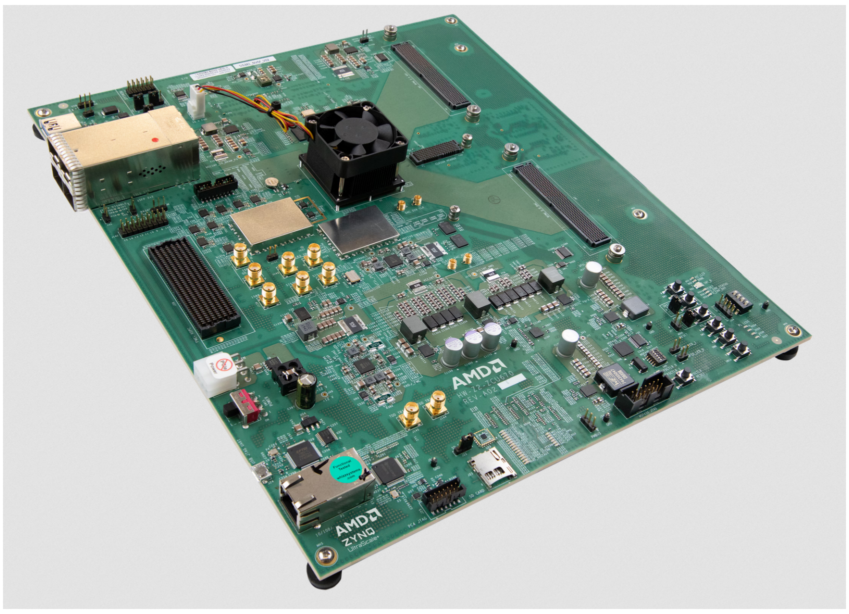


Figure 4: Picture of the AMD RFSoC development board from AMD's marketting material.

The RFSoC[5] series of RFICs shown in figure 4 are a relatively new Software Defined Radio platform developed by AMD. They feature up to 8 receive and transmit channel with a frequency range of DC to 5 GHz and a bandwidth of up to 5 GHz. This extremely high bandwidth is due to its novel direct RF sampling architecture. A block diagram of the latest RFSoC generation is shown in figure 5.

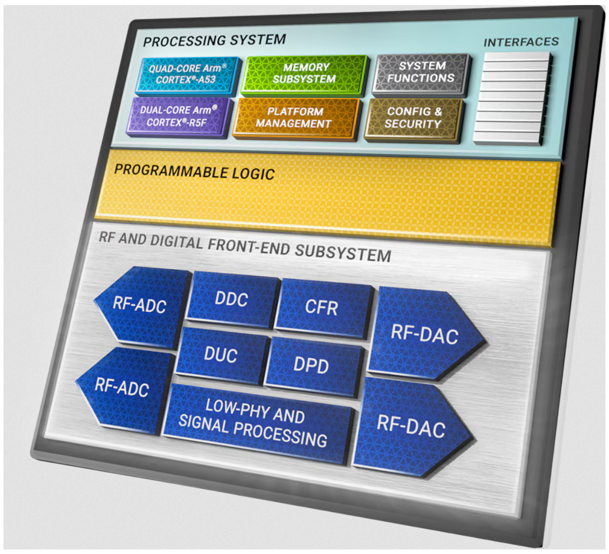


Figure 5: Block diagram of the AMD RFSoC chip as shown in AMD's marketing material.

As shown in figure 5, the AMD RFSoC integrates ultra high speed Analog to Digital and Digital to Analog converters as well as a programmable logic area, dedicated DSP and an entire hexa-core ARM SoC. As described previously, this chip uses direct RF sampling instead of analog mixers and local oscillators. This has only recently become possible for higher frequency bands as it requires extremely fast ADCs. The integrated ADCs are capable of running at up to 10 GHz. The RF signal is then down converted using Digital Down Conversion (DDC) and processed through the integrated DSP and programmable logic, after which the ARM CPU can interact with the data.

Such a high level of integration allows the RFSoC to have amazing performance per dollar in terms of bandwidth and flexibility, but it is so complex that the price is at the very upper end of what is acceptable for amateur and educational use.

2.3 USRP x310

Figure 6: Ettus Research USRP x310 modular SDR transceiver.

The Ettus Research USRP x310[9] as shown in figure 6 is a modular Software Defined Radio receiver platform. Unlike the previously described devices, this design is based on discrete data converters connected through a standard daughterboard interface to user replaceable frontends. Particularly of relevance is the TwinRX[8] daughterboard which instead of providing analog IQ signals, provides one channel on the I branch and another on the Q branch. This allows, as its name indicates, to receive from two inputs coherently instead of only having one input. Since the USRP x310 has space for up to two daughterboards, a single device is able to receive on up to four channels coherently.

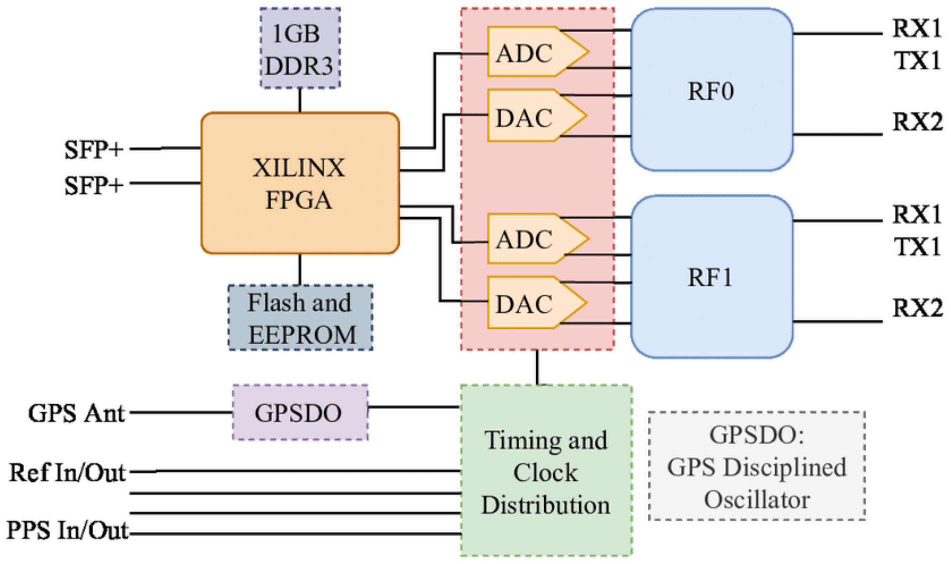


Figure 7: Block diagram of the USRP x310 modular SDR transceiver.

As shown in figure 7, the USRP x310 uses an FPGA to package up the samples from the data converters and send them over a standard SFP+ interface. While Ethernet over SFP+ is a relatively simple protocol to implement within an FPGA, SFP+ requires a dedicated network card in the host computer and expensive cables making it quite expensive and inflexible. The receiver possesses a frequency reference input and output as well as an integrated GPSDO socket for frequency synchronisation between devices.

2.4 KrakenSDR

Figure 8: The KrakenSDR 5-channel coherent Software Defined Radio receiver.

The KrakenSDR[13] shown in figure 8 as is a 5 channel Software Defined Radio receiver based on the RTL-SDR[24]. In order to truly understand it, it is first therefore necessary to understand the RTL-SDR, for which a block diagram is shown in figure 9. The RTL-SDR is based on the Realtek *RTL2832u* USB TV dongle chipset. Its original use was as a DVB-T TV dongle demodulator and USB interface, but reverse engineering efforts revealed it featured a mode in which IQ samples going to the built-in OFDM demodulator would instead be sent over USB directly[23], making it useful as an SDR receiver. The *RTL2832u* however is only half of the receiver as it still requires a frontend to give it an analog IF signal in either Low-IF or IQ format. For this, most modern RTL-SDRs use the Rafael Micro *R820T2* tuner. This tuner uses a simple heterodyne architecture with tracking filters for image rejection and include built in variable amplifiers for gain control. It also includes a fractional PLL synthesizer to generate its own LO internally.

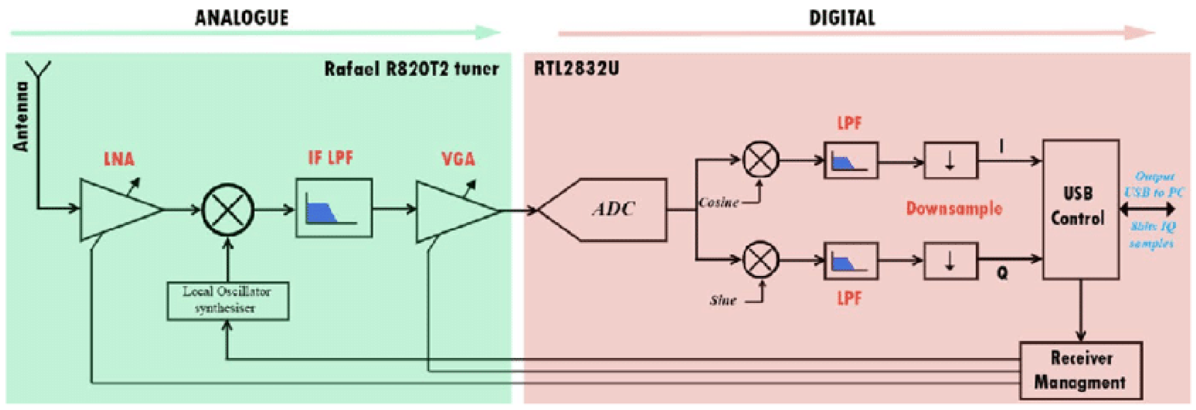
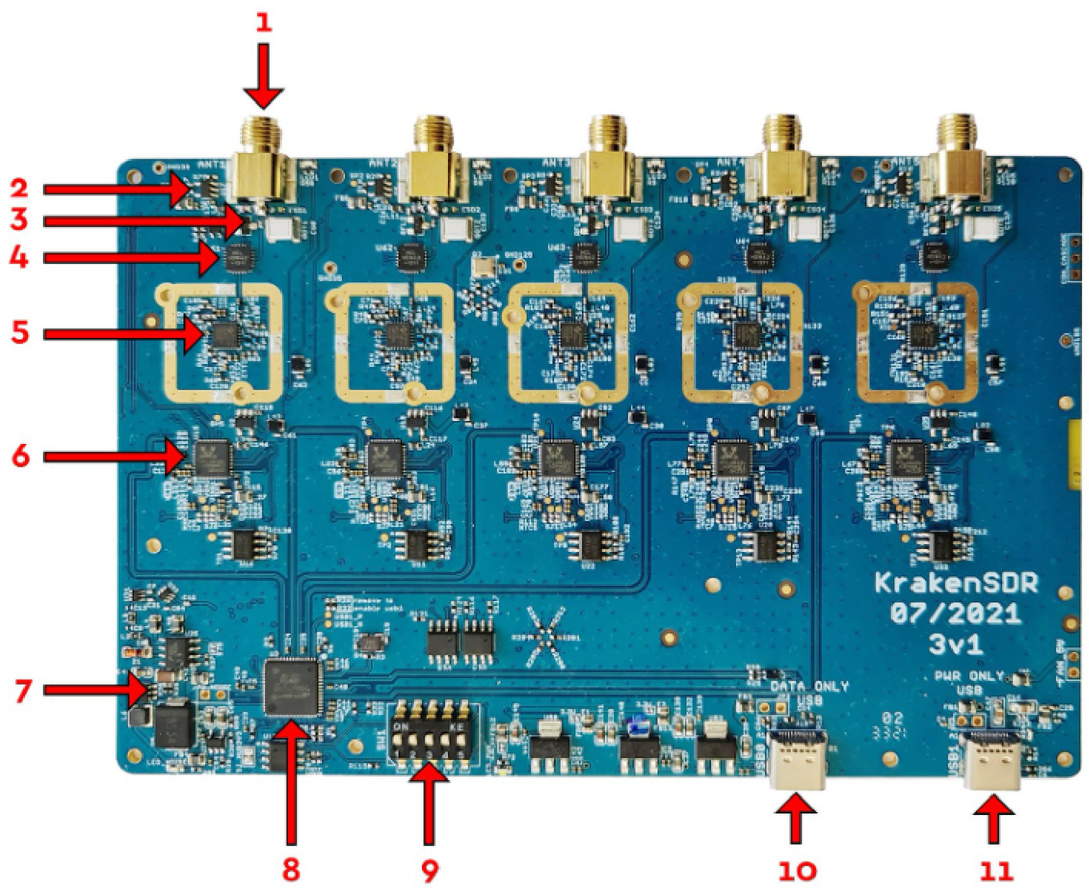


Figure 9: Block diagram of the RTL-SDR Software Defined Radio receiver. [34]

As shown in figure 10, the KrakenSDR is simply 5 RTL-SDRs sharing a common clock and communicating back to a computer through an integrated USB hub. No block diagram of the hardware is available from the manufacturer so an annotated diagram they provide as has been included here instead.



1. SMA Antenna inputs	**5.** R820T2 tuner	**9.** Individual tuner on/off DIP switched
2. Bias Tee	**6.** RTL2832U ADC	**10.** USB Type-C DATA
3. ESD protection	**7.** Noise source	**11.** USB Type-C PWR
4. Noise calibration switches	**8.** USB Hub	

Figure 10: Annotated picture of the KrakenSDR PCB. [12]

Unlike the previously mentioned devices, the KrakenSDR is not inherently coherent. Indeed, while the five RTL-SDR are running from the same reference clock, the phase of the various PLLs used to drive the ADCs and the mixers in the tuners will take on random phase offsets relative to one another every time they unlock. In order to achieve coherence, the device features a noise source that can be switched into all of its RF inputs. The PC software is then responsible for synchronizing the channels both in time and frequency. Time synchronisation is especially difficult as buffers of samples may be lost while going over USB and buffers have no inherent guarantee to line up with one another.

2.5 Comparison and Conclusions

In order to compare these coherent receiver implementations, it is first necessary to define the measure by which to rate both them and the hardware designed for this project. Indeed, as exposed in the previous examples, there is no shortages of variations for bandwidth, frequency coverage and channel count. For amateur and educational applications as targeted by this project, bandwidth and frequency coverage are secondary, within reason, to channel count. This is because if the use of the device is experimenting with beamforming and radio direction finding applications, bandwidth and frequency range are simple pass/fail requirements depending on signal the user wishes to receives and do not impact the beamforming or RDF performance in anyway. As such, channel count will be one of the factors used to rate the devices.

Since as mentioned previously the goal of this project is to create something of value to amateurs and educational institution, price must also be a factor in judging the performance. Indeed, if price is of no concern, there is already no shortage of high end offering from manufacturers like Per Vices Corporation and their 16 channel Cyan SDR costing over $200,000[25]. As such, the final index used to rate the receivers will be $i = \frac{p}{c}$ the price in dollar per number of channels. Table 2 shows the specifications, architecutre, price and rating index for every receiver investigated for this project. The table is sorted from lowest (best) to highest (worst) index.

Device	Architecture	Freq. Range [MHz]	B.W. [MHz]	Chan. [-]	Price [$]	Index [$/Ch.]
KerberosSDR	RFIC (LIF)	25 - 1750	2.4	4	299	74.75
KrakenSDR	RFIC (LIF)	25 - 1750	2.4	5	499	99.8
LimeSDR	RFIC (IQ)	0.1 - 3800	61.44	2	450	225
BladeRF 2.0	RFIC (IQ)	70 - 6000	61.44	2	540	270
USRP B210	RFIC (IQ)	70 - 6000	61.44	2	2160	1080
AMD RFSoC	RFIC (Dir.)	0 - 5000	5000	8	16995	2125.375
P.V. Crimson	Discrete (IQ)	0 - 6000	325	4	17000	4250
USRP x310	Discrete (LIF)	10 - 6000	80	4	19480	4870
P.V. Cyan	Discrete (IQ)	0 - 18000	1000	16	208950	13059.375

Table 2: Specifications of various coherent SDR receivers sorted by the index corresponding to price per number of channels.

Inspecting table 2 reveals several notable points. First, very high end devices are, as expected, very badly rated in this comparison, costing in the worst case over 170 times more per channel then the best rated receiver. This can be explained by the fact that these receivers are targeted at very different markets like Signals Inteligence (SIGINT) and Spectrum Monitoring [26] which are happy to pay a lot more per device for higher bandwidths and frequency ranges.

Second, all the better rated devices use RFICs instead of descrete receivers. This is trivial to explain by the cost advantages or integration and mass production. Discrete receivers, IQ or Low-IF all require considerably more components since the different units are split between several integrated circuits. While this usually allows for higher performance, it significantly increases the bill of material as well as production costs.

Finally, both the best and second best rated receivers are based on the RTL-SDR. This does however mean that they are also both affected by the limitations of the RTL-SDR, supporting a bandwidth of only 2.4 MHz, over 25 times less than the next best rated receivers. The reason for this dominance in the rating is explained by the fact that they benefit from cheap consumer grade hardware repurposed for SDR use, as opposed to more expensive RFICs like those used in the LimeSDR and BladeRF which are designer for commercial telecom infrastructure [16].

3 Hardware

As was demonstrated when reviewing the state of the art in affordable coherent SDR receivers, re-using commodity components outside of their original purpose and working around their limitations is key to achieve an affordable receiver design. The design created for this thesis is as shown in figure 11 and an overview of said design is provided below.

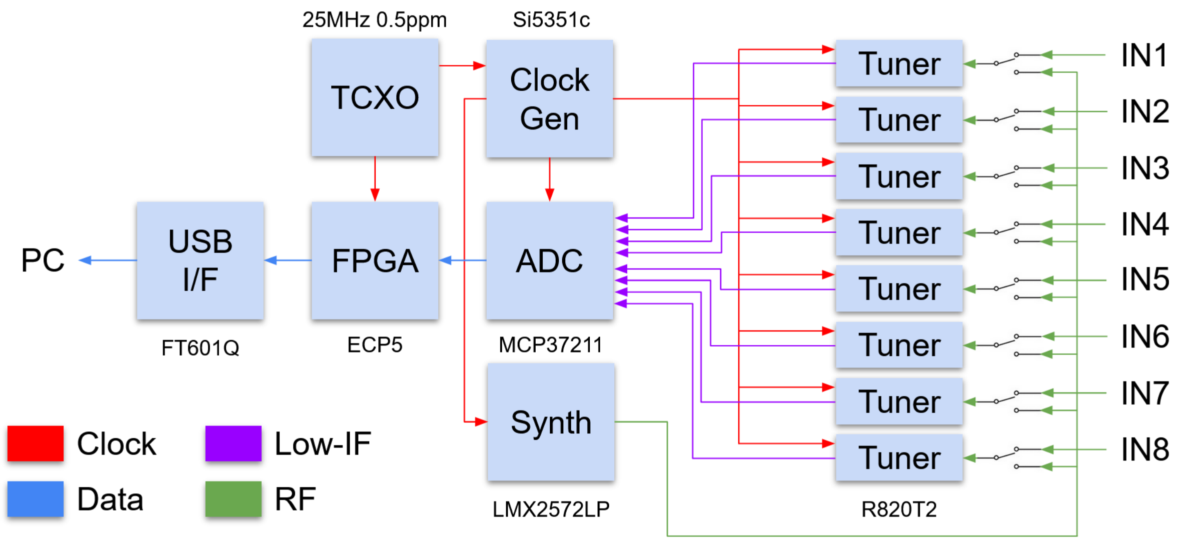


Figure 11: Architectural overview of the receiver hardware. Signal conditioning components are ommited for clarity.

Starting from the RF side, antenna input connect through Single-Pole Double-Throw RF switches to directly to individual tuners. The RF switches are used to switch out the received signal with a locally generated phase reference signal generated by a LMX2472LP single chip RF synthesizer IC. The tuners are Rafael Micro R820T2. These were originally designed as TV tuners for USB DVB-T dongles but happen to be very capable general purpose tuners. Due to behind massproduced commodity hardware, their price is extremely low at only around \$3 per unit. This explains why the phase reference is required. Indeed, these tuners come with their own integrated local oscillators and phase locked loop with no option to feed an external oscillator instead. Therefore, the reference is there so that a signal of known phase can be applied to each tuner's input and calibrate out the differences in phase that each PLL ended up with.

Next, the tuners output the downconverted and bandpassed signal as a Low-IF centered around 4.57 MHz. This IF passes through signal circuits conditioning not shown on the diagram before being sampled by an 8-channel 12-bit 25 $\mathrm{Ms^{-1}}$ ADC. This recently released low-cost ADC from Microchip is another part that allows the receiver to be quite cheap for its performance. The digitized IF is then sent over to an FPGA for processing.

Nowadays, many low cost FPGAs are available such as Intel/Altera's MAX10 and AMD/Xilinx's Spartan series. However, in order to further reduce costs like licensing, the Lattice *ECP5* was chosen due to the fact that a fully Free and Open Source toolchain exists for it. The FPGA is, in fine, supposed to do some basic signal processing and calibration on the IFs before packaging them into fixed length packets and forwarding them to the USB fifo. As will be described later, the onboard DSP has not yet been implemented and the samples are instead immediately packed as it.

The USB FIFO used for this design is an FTDI *FT601* capable of up to 3.2 $\mathrm{Gbit\,s^{-1}}$, well over the maximum data rate produced by the ADC. As will be explained later, this chip was chosen for its extraordinary low cost and to avoid other ICs like the Cypress FX3 which have a bad reputation in terms of stability, and are also much more expensive.

Finally, none of these subsystems could run without a clock, so a *Si5351c* clock generator is used to take the output of the single 25 MHz precision Temperature Controlled Oscillator and generate clocks for the ADC, synthesizer and tuners. Not shown on the diagram are the specialized splitter circuits used to distribute the the clocks and phase references adequately to the tuners.

With the rough design covered, an important part of actually implementing the design into a circuitboard is the EDA tool that will be used. For this thesis project, the Free and Open Source KiCad EDA software was chosen. This software has been in development for three decades is already used in industry. Its free and open source nature avoids the layer count or other annoying restrictions of the free versions of proprietary EDA tools such as Eagle or Altium Designer require.

3.1 Frontend

The frontend of the receiver in this design encompasses the ESD protection circuit, the phase reference injection, and down conversion to a low-IF using a tuner IC.

3.1.1 Input Selection

In order to calibrate the phase of each tuner it is necessary to add a Single Pole Double Throw (SPDT) switch in line with the input which allows selecting between the antenna port and the reference signal. Due to the fact that the synthesizer generating the reference signal is turned off when not in use, low insertion loss is to be preferred over high isolation. Indeed, keeping the insertion loss of the switch to a minimum is essential in keeping the noise figure of the front end low.

The switch chosen for this design is the *MASWSS0136* from Macom. It features a very low insertion loss of only 0.4 dB with a moderate isolation of 27 dB. It was chosen specifically for its low cost given its insertion loss. The circuit of the input selection circuit is as shown in figure 12.

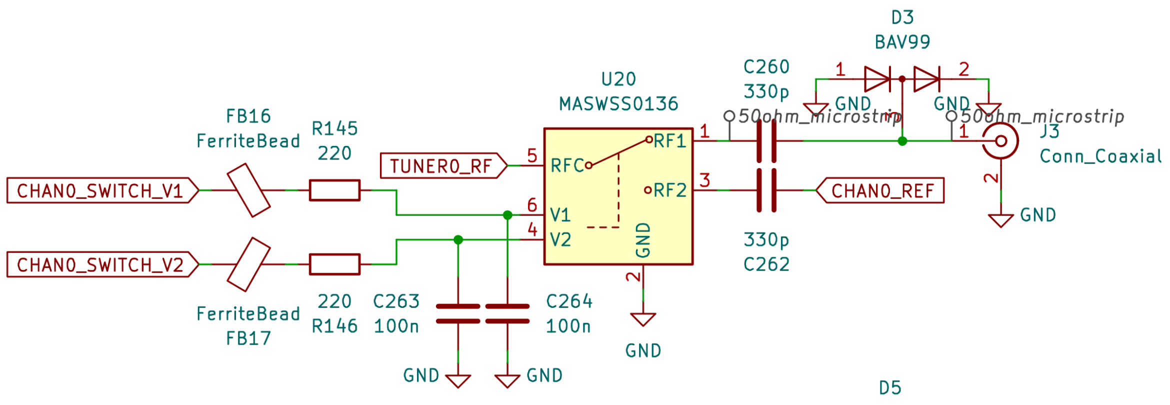


Figure 12: Schematic of the input selection and protection circuit.

Starting at the input, the signal coming from the antenna connector is connected through a *BAV99* low capacitance double diode to ground. This is done for ESD protection purposes as the RF switch IC does not have any built-in ESD protection. Connecting a diode directly to the RF port is simple but has two minor disadvantages. First, if a device such as a LNA imparts a DC offset onto its output and is connected to the antenna port of this device, the diode will short out the DC offset. To remedy this, an additional DC blocking capacitor could be added directly after the antenna connector. Second, although low, the parasitic capacitance of the diode will cause an impedance mismatch and thus induces reflections and losses in the frontend. The BAV99 is specified to have a maximum parasitic capacitance of 1.5 pF. At the upper end of the operating frequency range of the receiver, that is 1725 MHz, this creates a parallel reactance of 61 Ω.

Assuming a $Z_0 = 50\,\Omega$ source and RF switch input, it is possible to compute the losses due to impedance mismatch. The impedance at the receiver's input connector will the the 50 Ω of the RF switch input in parallel with the capacitor's 61 Ω reactange. This gives a total impedance of $Z_L = 30.677 + 24.347i\Omega$. The reflection coefficient is then computed as $\Gamma = \frac{Z_L - Z_0}{Z_L + Z_0} \approx -0.1360 + 0.3428i$. Finally, since the power reflected from the impedance change is given by $P_{\text{in}}|\Gamma|^2$, one can conclude that the power transmitted through the impedance change is equal to

$P_{\text{in}}(1 - |\Gamma|^2)$, and thus an insertion loss of $10\log_{10}\left(1 - |\Gamma|^2\right) = -0.635\text{dB}$ is therefore to expect at 1750 MHz.

After ESD protection the signal continues through a DC blocking capacitor to the switch. All ports of the switch have a DC blocking capacitor as the interal workings of the switch is such that a DC offset will appear on its ports. The manufacturer does go into detail about this but very clearly warns about the requirement for DC blocking capacitors. The exact value of these capacitors is not crucial but the choice is explained later in the Tuner subsection.

Finally, the state of the switch is controlled through two input pins `V1` and `V2`. In order to isolate any noise coming from the FPGA's GPIOs from the sensitive RF signals, the control signals are first run through a ferrite bead and a RC low pass filter. This low pass filter was designed to have a time constant of 50 µs which was chosen relatively arbitrarily to be short enough for switching to be fast without allowing too much signal noise to leak through at 25 MHz, that is the lower end of the receive frequency range.

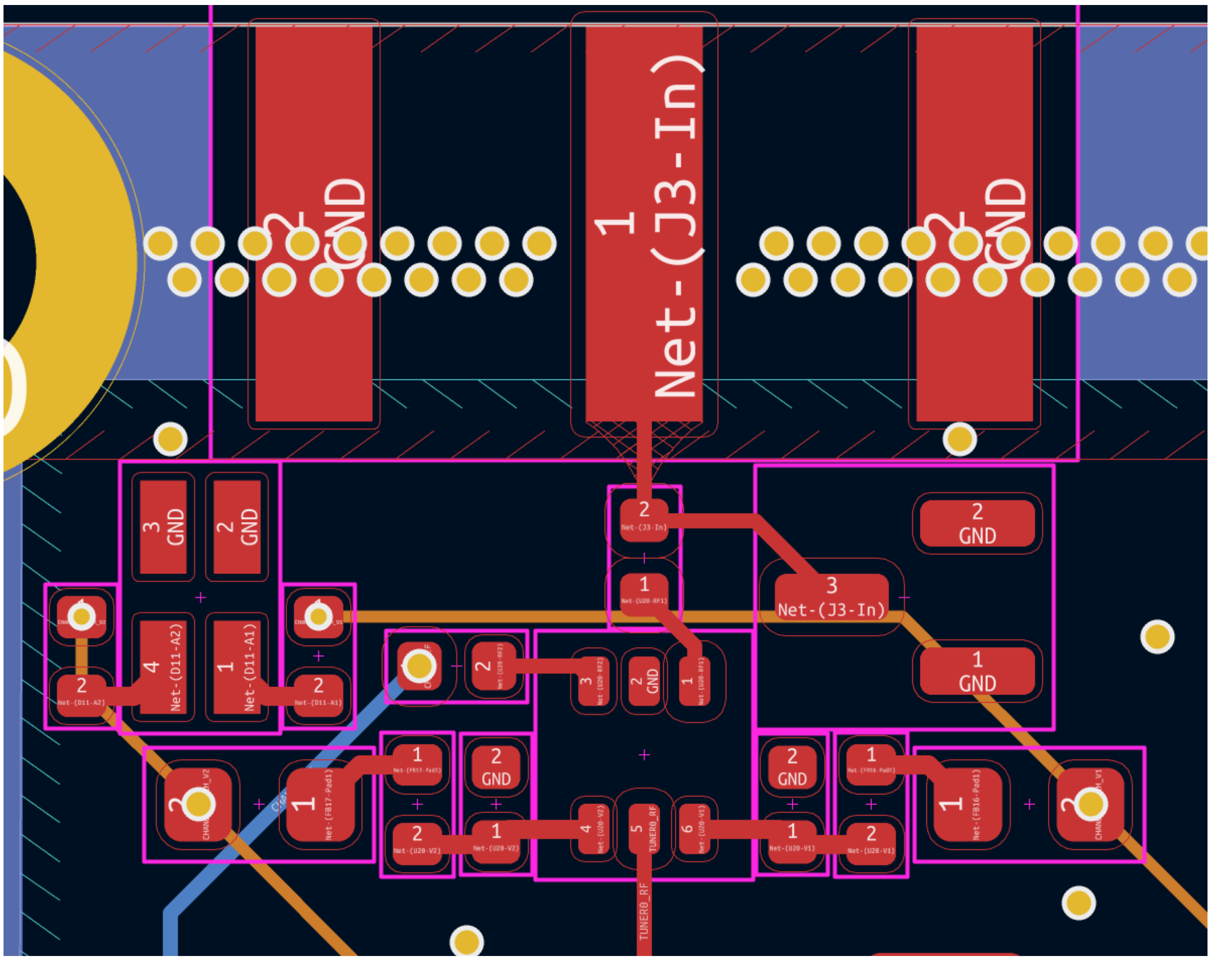
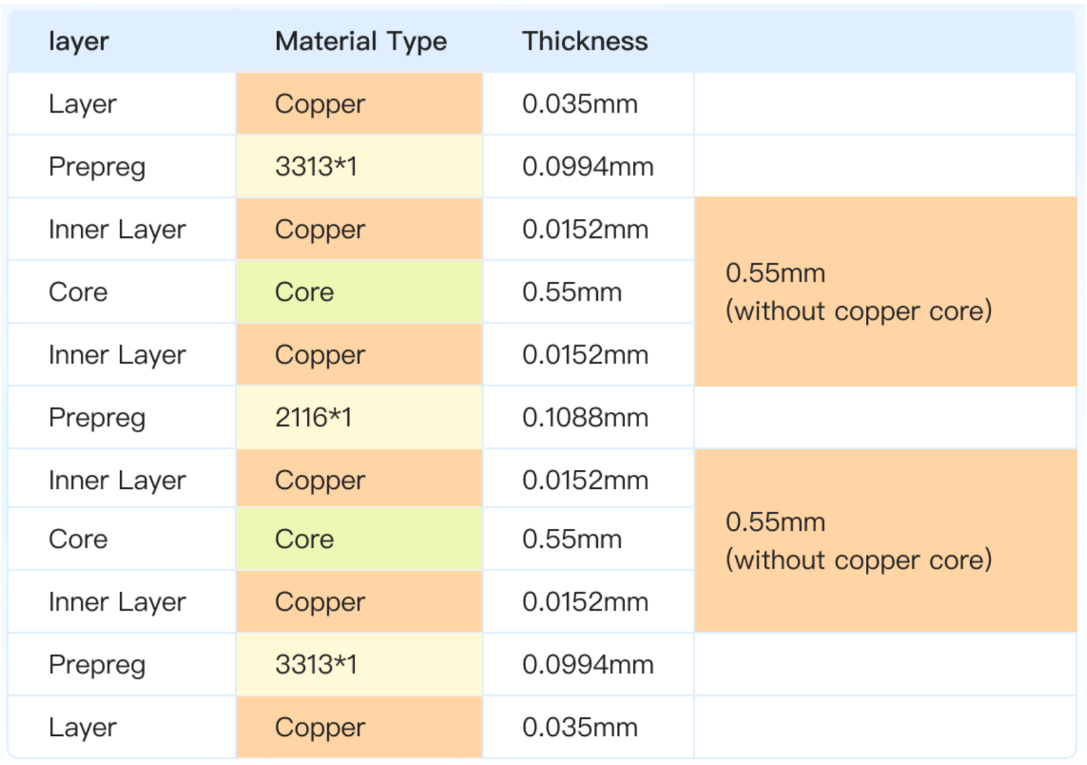


Figure 13: PCB layout of the input selection and protection circuit. Layer 1 is represented in red, layer 3 in orange and layer 6 in blue.

The PCB layout of the input selection and protection circuit is as shown in figure 13. all RF traces were layed out on the first layer and kept as short as possible to reduces losses and keep the PCB compact. Additionally, the second layer is kept a ground plane under the entire section in order to provide the ground required for the micro-strip traces and to properly isolate them from other signals running in the area.

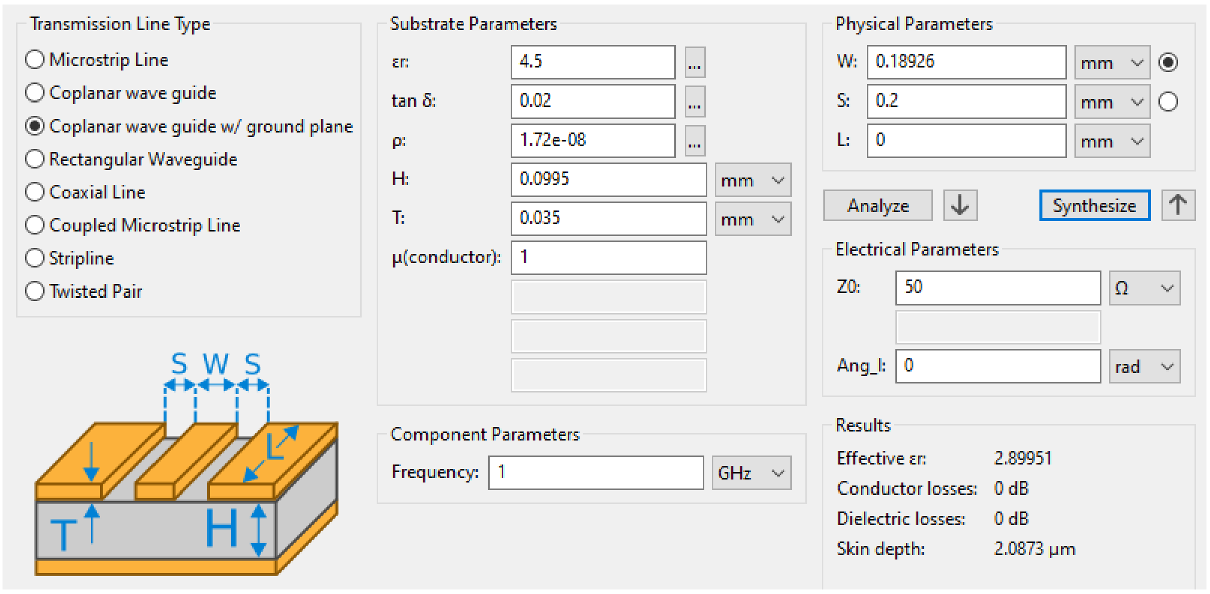
In order for the microstrip traces to be matched to both the input and the RFICs, their width must be calculates using the parameter given by the PCB manufacturer, which for this project was JLCPCB. The exact stackup used was JLC06161H-3313 using FR-4 substrate. All the physical parameters of this stackup such as layer thicknesses and dielectric indices are indicated on the manufacturer's website[10] and the stackup is shown in figure **??**.



layer	Material Type	Thickness	
Layer	Copper	0.035mm	
Prepreg	3313*1	0.0994mm	
Inner Layer	Copper	0.0152mm	0.55mm (without copper core)
Core	Core	0.55mm	
Inner Layer	Copper	0.0152mm	
Prepreg	2116*1	0.1088mm	
Inner Layer	Copper	0.0152mm	0.55mm (without copper core)
Core	Core	0.55mm	
Inner Layer	Copper	0.0152mm	
Prepreg	3313*1	0.0994mm	
Layer	Copper	0.035mm	

Figure 14: Stackup specification as shown on the PCB manufacturer's website[10].

The values for the first and second layer as well as the dielectric between them can then be entered in KiCad's integrated micro-strip calculator aiming for a trace impedance of 50 Ω and a microstrip type of coaxial waveguide. The results of the calculation are as shown in figure 15. The calculator shows a required width of 0.189 mm. Since the stackup is symmetric, the results are also valid for traces run on the under side of the PCB such as the phase reference signals.



Transmission Line Type
- Microstrip Line
- Coplanar wave guide
- [✓ Coplanar wave guide w/ ground plane]
- Rectangular Waveguide
- Coaxial Line
- Coupled Microstrip Line
- Stripline
- Twisted Pair

Substrate Parameters

εr:	4.5	
tan δ:	0.02	
ρ:	1.72e-08	
H:	0.0995	mm
T:	0.035	mm
μ(conductor):	1	

Component Parameters

Frequency:	1	GHz

Physical Parameters

W:	0.18926	mm	●
S:	0.2	mm	○
L:	0	mm	

Electrical Parameters

Z0:	50	Ω
Ang_l:	0	rad

Results

Effective εr:	2.89951
Conductor losses:	0 dB
Dielectric losses:	0 dB
Skin depth:	2.0873 μm

Figure 15: Interface of KiCad's Micro-strip calculator with the stackup information and desired settings filled out and the results visible.

As shown in figure 13, the protection diode was placed as close as possible to the input in order to reduce the impedance to it and thus better protect the circuit in case of en ESD event. After going throught the DC blocking capacitor mentioned previously, the signal reaches the RF switch IC. The control signals for this switch arrive from the internal 3rd layer passing under the ground plane and then go through the filters described in the explanation of the schematic. These signals and filters are layed out orthogonally to the RF traces arriving to and leaving from the switch helping with isolation. The phase reference signal arrived from the bottom layer and goes directly into the DC blocking capacitor and switch.

3.1.2 Tuner

The job of the tuner is to take the un-conditioned RF signal from the antenna, extract a certain frequency range from it and down-convert it to an Intermediate-Frequency (IF). In the past, this would have required multiple chips, hundreds of passive components and would have required manual tuning or "Alignement". Thanks to advances in RF technology and the need for mass production of radio/television enabled devices, such a circuit is now available in single ICs.

For this project, the *R820T2* from *Rafael Micro* was selected. It features an official frequency range of 42 MHz to 1002 MHz, but has been confirmed by many amateur projects to work fine from 25 MHz to 1750 MHz. The tuner outputs a 2 kΩ impedance differential Low-IF where the passband is located between 1 MHz and 9 MHz. This results in a usable bandwidth of 8 MHz which, for its original purpose, was enough to receive the widest standards DVB-T signals. This chip has been the basis of many of the best selling amateur Software Defined Radio devices in the past 15 years. This is due to its incredible performance for its price. In large quantities, this chip can be bought for less than 2 dollar a piece, a price unmatcheable by discrete circuitry. Notable devices using this chip are the RTL-SDR[24] and Airspy [2].

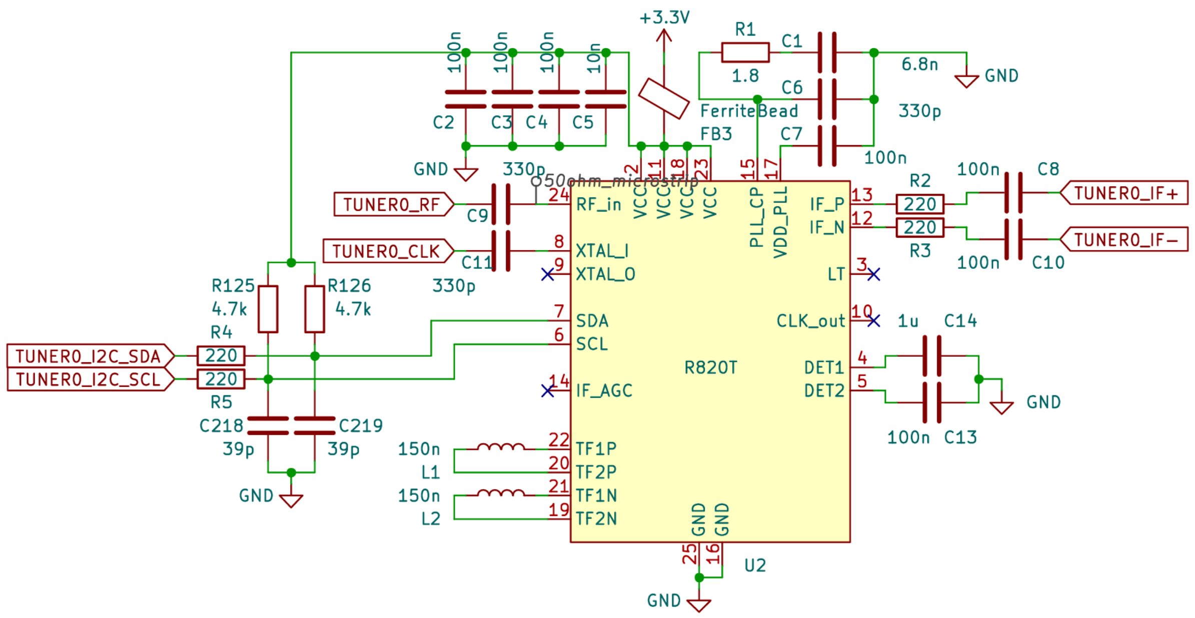


Figure 16: Schematic of the tuner sections.

As shown in figure 16, the R820T2 requires very little external circuitry to work. First, its single 3.3 V supply rail is filtered using a ferrite bead as well as 100 nF capacitors on each of it's power input pins. The R820T2 also exposes two internally generated voltages for external decoupling. The first is the `PLL_CP` pin which is the voltage generated by an internal charge pump used to power the integrated Voltage Controlled Oscillator [3]. The second is `VDD_PLL` which is the power supply for the PLL. The manufacturer gives the exact decoupling circuits for each of these pins so design was not necessary or even recommended in the first place.

The RF and reference clock are fed to the tuner via 330 pF DC blocking capacitors. The value of these capacitors was again specified in the datasheet but can be explained by the requirement of having a low impedance at the lowest operation frequency. Using the traditional capacitor impedance formula, it can be calculated to be only 20 Ω at 25 MHz. While it would be possible to simply use a higher value capacitor, it was decided to keep the manufacturer recommended value since such a small impedance change is not worth straying away from the recommended value which may cause issues with the DC offset output from the switch IC such as damage to the input of the tuner caused by the charging current of the DC blocking capacitor at power on. On the IF side, the DC blocking capacitor and protection resistors were also specified by the manufacturer. Since the IF is not used outside of its official specs, there is no need to recompute these values.

Next, some additional passive components are required. First are the two 150 nH inductors. These serve as the basis of the internal tracking input filters of the tuners. Their value are obviously specified by the manufacturer as they have to work with the internal immutable circuitry of the IC to resonate at the correct frequency. Second are the decoupling capacitors connected to the `DET1` and `DET2` pins. These serve as low pass filters for the power detectors built into the R820T2 for use by the intergrated Automatic Gain Control. Since this feature will not be used and the effect of leaving them unpopulated is not known, they were included with their manufacturer recommended value.

In order to reduce noise going into the tuner from the digital section and as per the manufacturer's specifications, RC low pass filters are included on the I2C pins of the tuner. The values specified by the manufacturer were used as is but it is possible to confirm that they are sensible. Knowing the 3 dB corner frequency of a RC filter is given by $F_{3\text{dB}} = \frac{1}{\pi RC}$, one can conclude that $F_{3\text{dB}} = 18.55\,\text{MHz}$, which is both well over the maximum operating frequency of the I2C bus of 100 kHz, but also well below the lower end of the receive frequency range of 25 MHz ensuring that the I2C communications won't be disrupted and that the noise coming from these lines will be attenuated in the receive frequency range.

Finally, the Low-IF output if the tuner passes through some 220 Ω protection resistors and DC blocking capacitors before going off to the rest of the receiver circuit. The value of the DC blocking capacitors are much higher at 100 nF than those on the RF side due to the much lower frequency of 1 MHz at which they are to operate. Their exact value is again not critical so the very commonly used value of 100 nF was chosen. This creates a series impedance of 1.591 Ω which is negligeable compared to the 2 kΩ differential impedance of the IF.

3.2 Clocks

The description of the clock generation hardware will be split into two section. First, the generation of the clock signals will be discussed, followed by splitter circuit used to feed the 8 tuners.

3.2.1 Generation

As shown in the overview, three subsystems of the receiver require separate clock signals as listed below.

1. **Tuners:** Each tuner require a fixed 16 MHz CMOS clock.
2. **Synthetize:** The phase reference synthesizer requires a configurable 5 to 125 MHz LVDS clock.
3. **ADC:** The ADC requires a LVDS sampling clock equal to the desired sample rate. For this design, this means 200MHz.

A very popular clock generator that fits these requirements is Silicon Labs's *Si5351C-B* [28]. This clock generator features 8 CMOS or 4 LVDS outputs that can range in frequency from 25KHz to 200MHz. This chip requires a 25 or 27MHz input clock, so this design includes an onboard 25MHz Temperature Controlled Oscillator (TCXO) with 2ppm frequency stability which will be discussed later in this section. The *Si5351C-B* also features a 10MHz reference input which can be used as the input clock for its PLL in order to use an external high precision reference such as a Oven-Controlled Oscillator (OCXO), GPS Disciplined Oscillator (GPSDO), Rubidium standard or other atomic clock. The circuit for this clock generator is shown in figure 17.

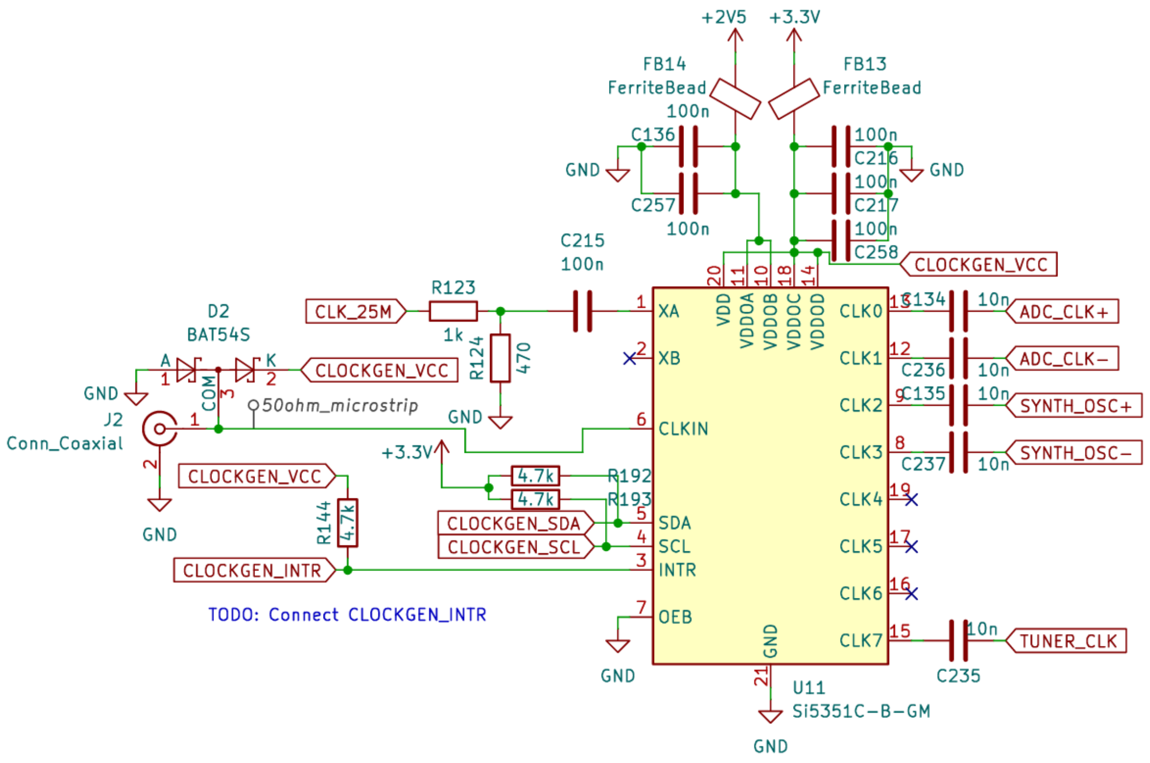


Figure 17: Schematic of the clock generation section of the receiver.

Power for the *Si5351C-B*'s power rails first pass through ferrite beads to create a high impedance for high frequency components in the power supply. This is done because any interference on the power rails of the clock generator will translate directly to phase noise in its generated clocks. Power is then distributed to each of the power pins in parallel with 100 nF decoupling capacitor.

The *Si5351C-B* requires a 3.3 V for VDD. The output buffer power inputs *VDDOx*'s voltages depend on what is connected to them. VDDOA covers outputs 0 and 1, VDD0B covers outputs 2 and 3 and so on. The unused output pins *CLK4*, *CLK5*, *CLK6* as well as the output going to the tuners *CLK7* are all configured as CMOS and correspond to *VDDOC* and *VDDOD* which are therefore connected to 3.3 V. On the other hand, output pins *CLK0*, *CLK1*, *CLK2*, *CLK3* are connected to the ADC and phase reference synthesizer and are configured as LVDS which requires a 2.5 V supply according to the *Si5351C-B*'s datasheet [28].

The oscillator input *XA* of the *Si5351C-B* expects a maximum voltage of 1.3 V. As such, a resistor divider required to reduce the 3.3 V of the 25 MHz TXCO. This can be done with a division ratio of around 3 and was achieved using 1 kΩ and 470 Ω resistors. These values were chosen because they were already required elsewhere in the circuit which avoids adding different component values to the Bill of Material. A 100 nF DC blocking capacitor is also added as per the manufacturer's recommendations.

The 10 MHz reference input is CMOS compatible is this directly connected to the input connector. For Electrostatic Discharge (ESD) protection, some reverse-biased diodes are connected from this input to ground and 3.3 V.

Finally, the I2C pins are pulled up to 3.3 V by 4.7 kΩ resistors and so is the *INTR* pin which didn't end up being used. The signals go straight to the FPGA for controlling the chip.

The layout of the clock generation section of the receiver is as shown in figure 18.

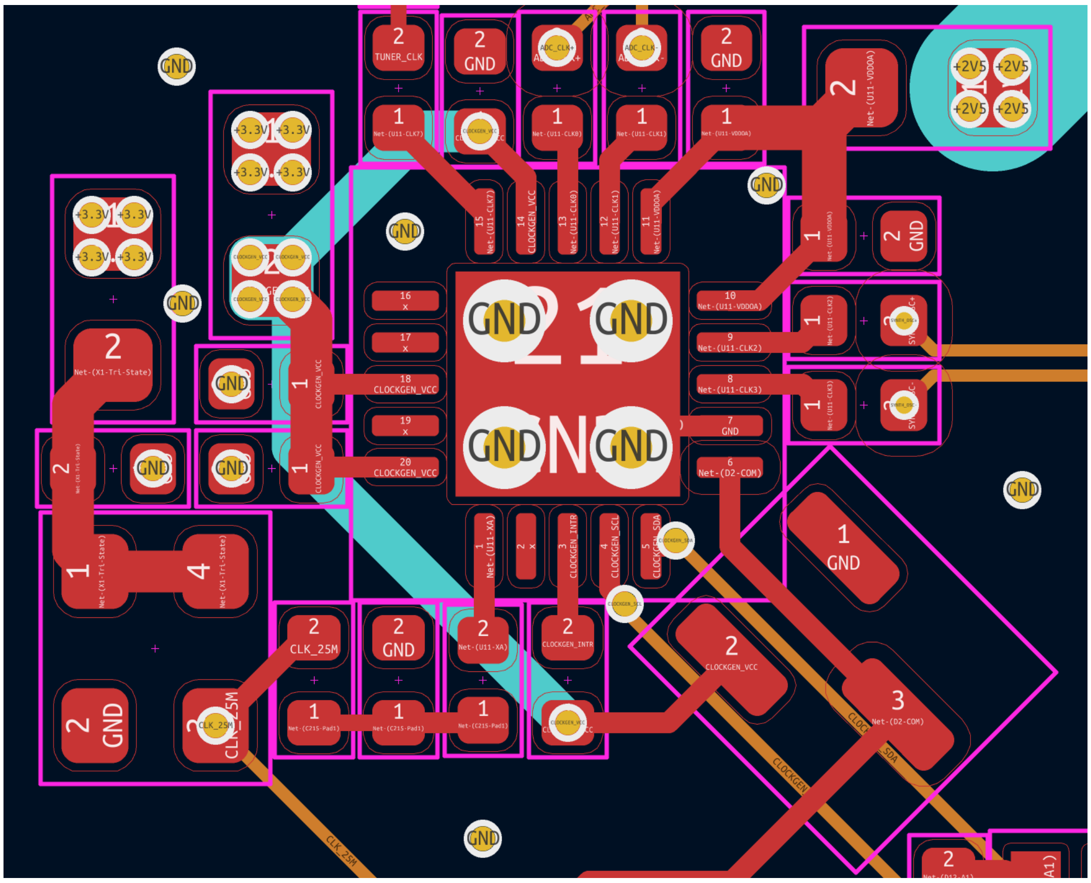


Figure 18: PCB layout of the clock generation section of the receiver. Red corresponds to the first layer, orange to the third and cyan to the forth. Layer 1 is represented in red, layer 3 in orange and layer 4 in cyan.

As can be seen in figure 18, decoupling capacitors are always placed directly in front, and as close as possible to their respective power pins. This is done in order to minimize the impedance to the decoupling capacitor which helps reduce power rail ripple. Four large vias are placed on the ground pad of the *Si5351C-B*. This is done both to help with thermal dissipation, but also to ensure a very low impedance path to ground.

Most signals are routed directly on the top (red) layer, with power rails on the 4th and 5th (not visible in the figure 18) and inter-section signals on the 3rd layer. Layers 2 and 6 are used as dedicated ground layer, and all unused areas of layers 3 are also connected to ground.

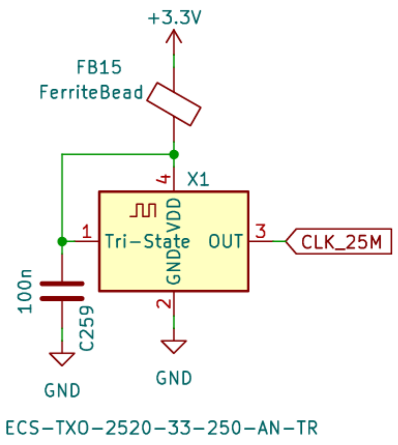


Figure 19: Schematic of the Temperature Controlled Oscillator used as the 25 MHz source of the clock generator.

As mentioned earlier, a 25 MHz CMOS clock is necessary for the *Si5351C-B* to operate. This is provided by a 2 ppm TXCO as shown in figure 19. Just like the clock generator, a ferrite bead and decoupling capacitor are placed at the power input of the TCXO in order to attenuate high frequency components from the power rail and improve phase noise performance.

3.2.2 Distribution

Finally, the generated clock must be split out to each tuner. Contrary to what one would expect, matching the phase of these clock signals to each tuner is not actually necessary. Indeed, any phase shift to these signals will be calibrated out anyway when compensating for the random phase offset of each of the tuner's PLL. However, in order to ease debugging, best efforts were made to still keep these signals in phase. Clock splitting is implemented using a simple resistive network as shown in figure 20.

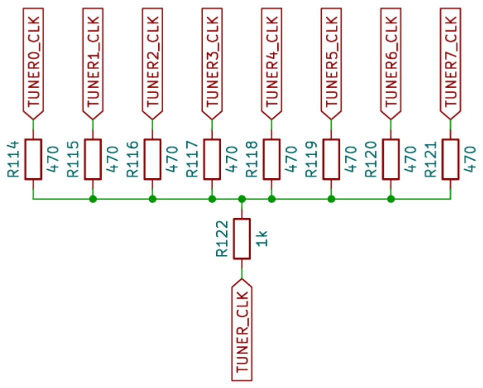


Figure 20: Schematic of the resistive clock splitter used to feed all 8 tuners from a single output of the clock generator chip.

This circuit is implemented in such a way to accomplish two things. First, the level of the clock signal must be dropped from the 3.3 V LVCMOS output of the clock generator down to around 150 mV. This is because the tuners were not originally meant to be driven by a regular clock, they are designed to resonate using an external crystal and have a built-in oscillator amplifier. Driving them at 3.3 V could potentially lead to damage. Second, each tuner must have a relatively high impedance to the next. This is done to prevent unwanted interactions between the tuners such as signal and noise leakage. The exact values of this splitter were tentative as no specifications are available on the crystal port of the tuner. These values were suggested by Mr. Youssef Touil from Airspy SA. who has significant experience using the R820T2 chip. The layout of this splitter is shown in figure 21

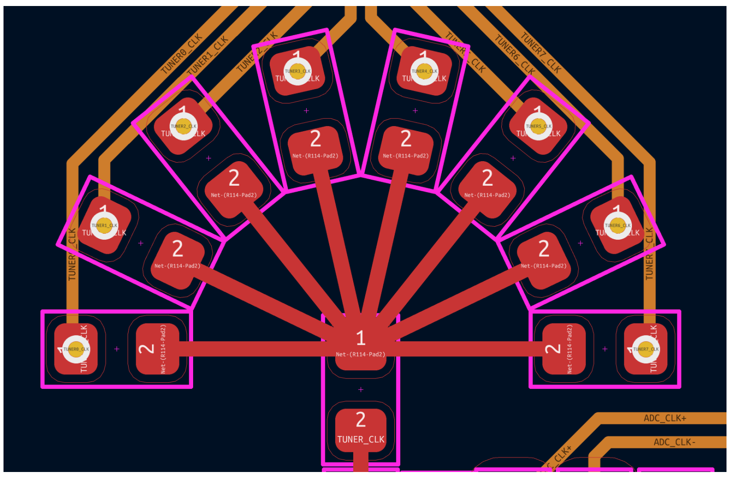


Figure 21: PCB layout of the resistive network used to split the tuner clock into 8 in-phase clock signals. Layer 1 is represented in red and layer 3 in orange.

The network is layed out in such a way that it is fully symetric from the clock source to each tuner and thus won't shift the phase of the clock. Of course, the wave length of the 16 MHz clock, equal to $\lambda = \frac{c}{F} = 18.737\,\text{m}$ is extremely large compared to the circuit which is in the order of a few millimeters. This means that any asymetry in this section would have a minimal effect, but it is still good practice to use a symetric layout as that has effecttively no cost.

Finally, after being split, the clock signals have to be routed to their respective tuner ICs. As shown in figure 22, the clock signals are run on the third layer, first groupe together and then spread out shortly before reaching the tuners. Before each tuner, length matching sections are implemented to ensure the phase of the clock reaching each tuner is identical. The geometry of these length matching sections is automatically computed and generated by KiCad given a desired length, which in this case was simply the length of the longest trace before length matching.

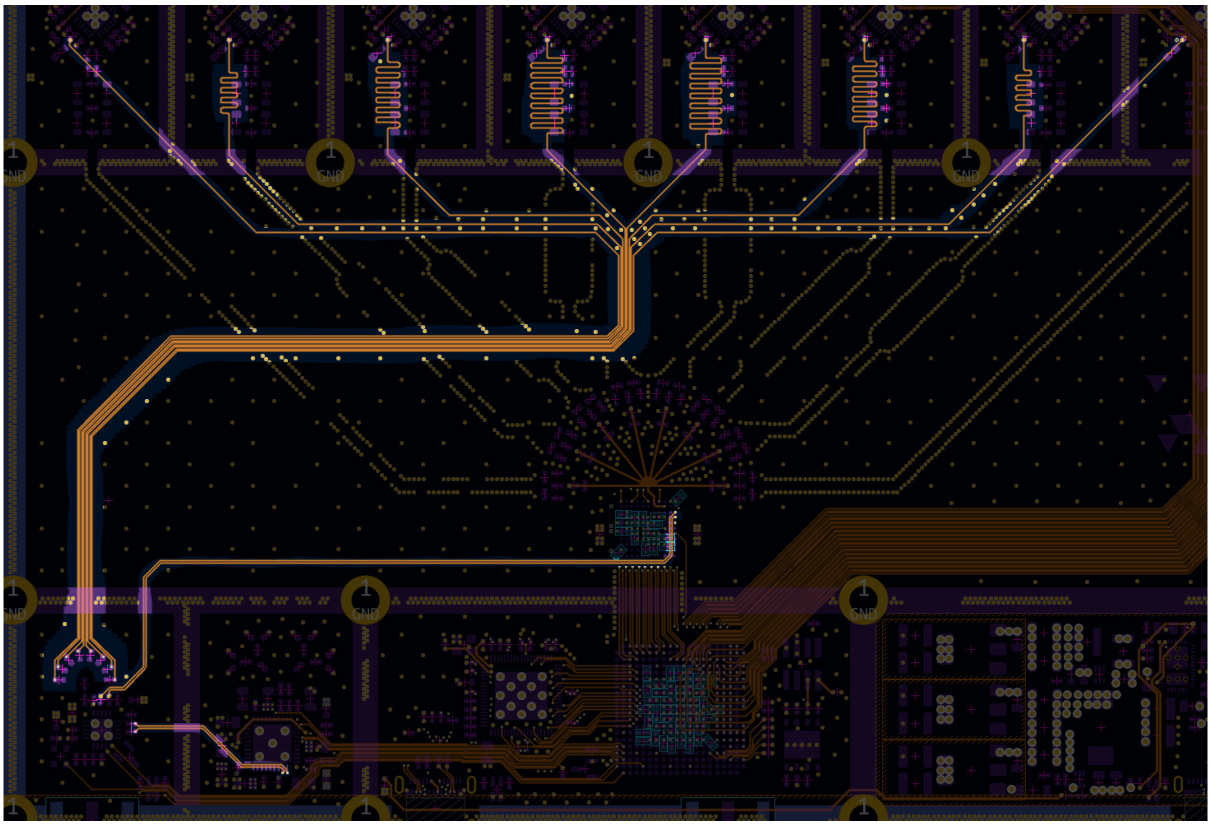


Figure 22: Routing of every clock signal generated by the Si5351c clock generator. The signals in question are highlighted.

3.3 Phase Reference

The description of the phase reference hardware will be split into two section. First, the generation of the refrence signal will be discussed, followed by the design and analysis of its distribution circuit.

3.3.1 Generation

As described earlier, each tuner in this design uses its own internal synthetizer generated from their clock input as their local oscillator. Unfortunately, this means that even while sharing the same input clock, all tuners will only be frequency locked and not phase locked. Any time their integrated PLL unlocks, for example when tuning to another frequency or when powering them off while the receiver is idle, their relative phase is randomized. As such, a phase reference signal is generated and distributed to each tuner. The purpose of this signal is to provide knowm phase reference that can be used to infer the phase offset of each tuner, and thus make the entire system truely coherent.

There are relatively few requirements on this reference signal. It must of course have relatively low phase noise to avoid requiring impractically long calibration periods, but most importantly, it should cover the entire desired frequency range of the receiver.

As such, Texas Instrument's LMX2472LP was chosen for its low price and fitting the requirements outlined above. This chip features two parallel output of which only one will be used. It's able to cover a frequency range of 12.5 MHz to 2 GHz which almost perfectly fits the receiver's frequency range. It is controlled via a Serial Peripheral Interface (SPI) bus and only requires a single clock input anywhere from 5 to 125 MHz.

The phase refence circuit is as shown in figure 23.

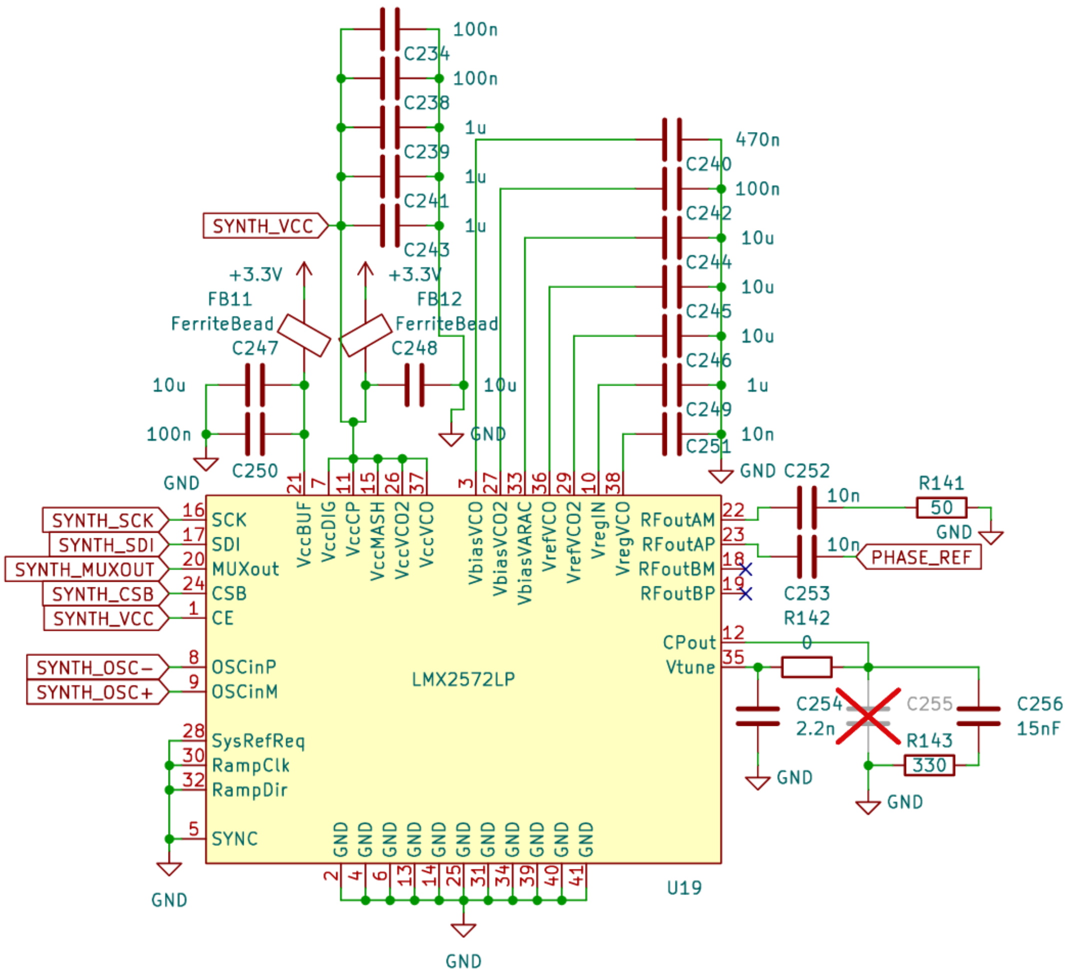


Figure 23: Schematic of the phase reference synthesizer.

Just like for the clock generation, power for the *LMX2572LP*'s power rails first pass through ferrite beads to create a high impedance for high frequency components in the power supply in order to attenuate high frequency component in the power supply. Power is then distributed to each of the power pins in parallel with decoupling capacitors of which the characteristics were specified by Texas Instruments [30]. As per the manufacturer's specification, separate ferrite beads were used for the output buffer power pins in order to insolate them better from the rest of the chip's power rails.

The *LMX2572LP* provides dedicated pins for decoupling various internal signals. The required values of the capacitors on these pins were also directly given in the datasheet.

The output of the synthesizer being 100 Ω differential, it is necessary to adapt it to 50 Ω single ended. For this, DC blocking capacitor are first added to each side of the differential output, after which the positive side is connected to ground through a 50 Ω resistor and the negative side goes out to the distribution network. This looks to the chip as if there was a 100 Ω differential load on its output while being extremely wide band and cost effective, unlike a balun for example. Whether the positive or negative side of the output is connected to the grounded resistor does not matter since only the relative phase between tuners is of concern. For this project, the negative side was chosen as this made routing easier on the PCB.

Another important part of the phase reference circuit is the loop filter composed of C254, C255, C256, R142 and R143. The manufacturer does not give particular guidance on designing this filter and asks to use their software to design it. Fortunately, the schematic for the evaluation board of the *LMX2572LP* is available so the values for these parts was taken from it. As can be seen in the schematic, both C255 and R142 were not needed and thus were replaced respectively with nothing (Do Not Populate) and a 0 Ω jumper.

The PCB layout of the phase reference section is as shown in figure 24.

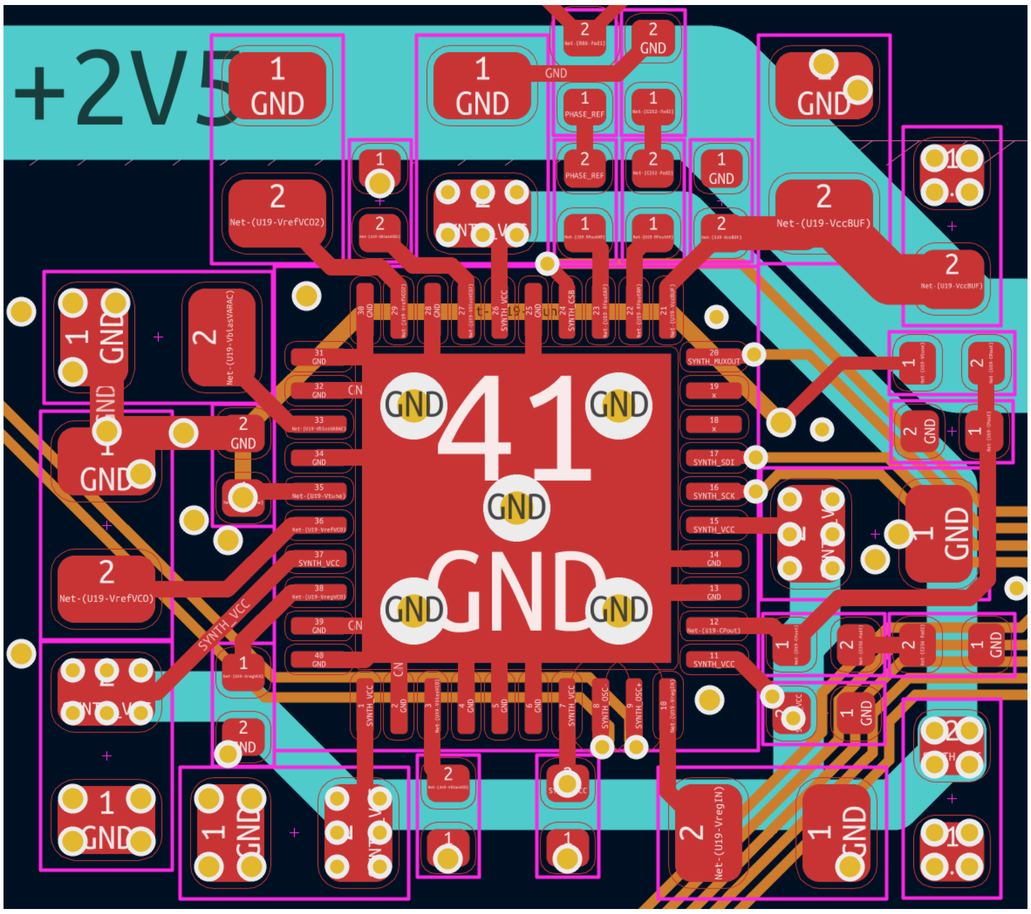


Figure 24: PCB layout of the phase reference generation circuit. Layer 1 is represented in red, layer 3 in orange and layer 4 in cyan.

3.3.2 Distribution

The most critical property of the phase reference signal is that it must arrive at each frontend with exactly the same phase. Indeed, any phase offset in this reference would directly translates into a phase error because the phase calibration algorithm assumes the references have the same phase on every channel. Like the tuner clocks, distribution of the phase reference consists in first splitting it and then routing it to the frontends.

To split the reference, both an active or passive splitter could be used. Since the tuners are extremely sensitive, losing power in the phase reference is not only desirable, but mandatory to avoid damaging the tuners. Thus, a passive resistive splitter is chosen. In order to design this splitter, lets first start by inspecting the two most common three port networks, that is the *Delta* configuration as shown in figure 25 and the *Star* configuration as shown in figure 26.

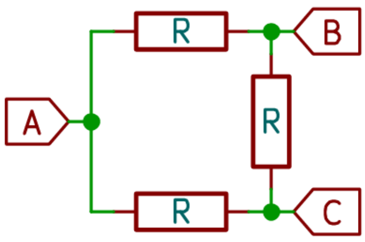


Figure 25: Three-port "Delta" configuration network.

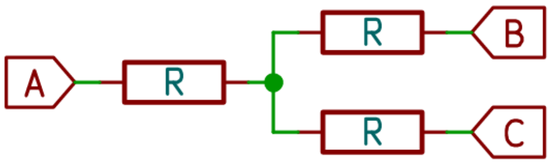


Figure 26: Three-port "Star" configuration network.

While both of these networks can achieve the goal of splitting the signal between multiple ports, one has to consider how the circuit will evolve when trying to stack this circuit recursively in order to get an 8-way splitter. In the case of the *Delta* network, one would end up with $3 * (1 + 2 + 4) = 21$ resistors with no obviously way to simplify the resistors. However, when using the *Star* network, one can notice that the output resistor of one stage and the output resistor of the other are in series and can thus be added up to $2R$. This reduces the number of required resistors from 21 to only $1+2+4+8 = 15$, making the *Star* network 29% more spare/cost efficient that the *Delta* network.

Going back to the signle stage *Star* network, since the impedance on each port assuming a $50\,\Omega$ termination on the two other must be $50\,\Omega$, we can conclude that all three resistors must be equal by symmetry. Solving for R is shown in equation 1.

$$R_{\mathrm{A}} = R + \cfrac{1}{\cfrac{1}{R+50} + \cfrac{1}{R+50}} = \frac{3R+50}{2} = 50 \iff R = \frac{50}{3} \approx 16.667\,\Omega \tag{1}$$

Knowing the value of R, it is now possible to build the full 8-way splitter using both R and $2R$ resistors as explained above. The schematic for this final 8-way splitter is as shown in figure 27.

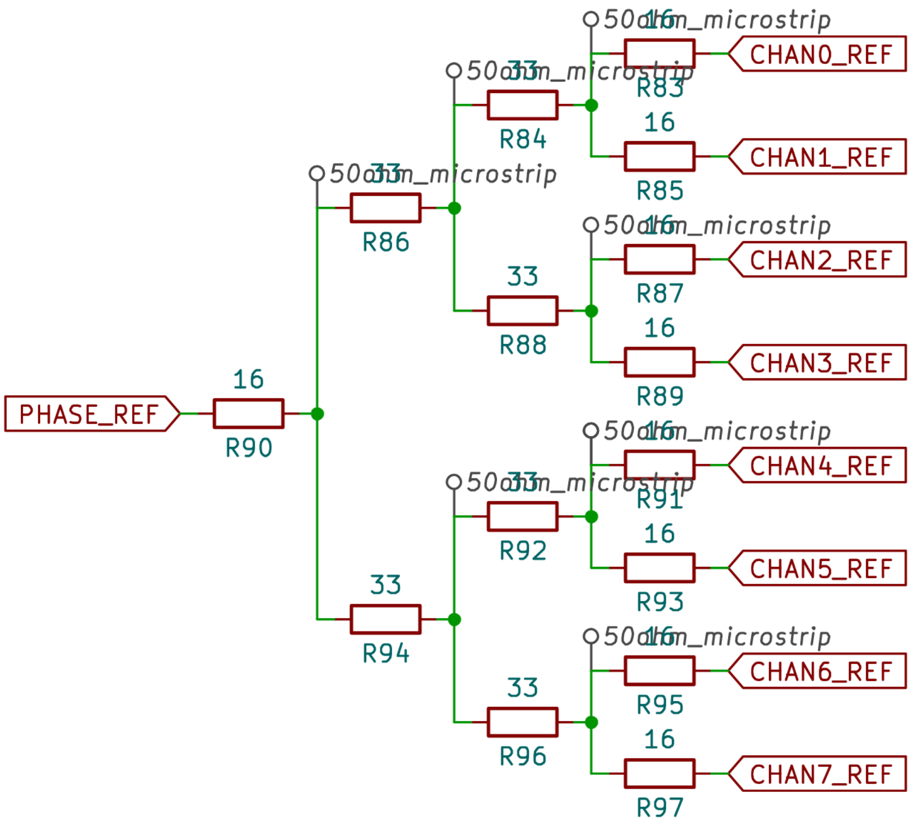


Figure 27: Circuit of the 8-way resistive splitter used to split the phase reference to the 8 frontends.

Of course, the circuit is only one part of ensuring that each output receives the same phase. The second part is the physical layout of the components. Indeed, the phase reference frequency can be as high as 1750 MHz which corresponds to a wavelength of approximately 17 cm. This corresponds to a phase offset of over 2° per millimeter, meaning that the physical layout of the circuit will be crucial to ensuring phase matching on all ports. As shown in figure 28, each branch of the splitter is made to be recursively symetric with the last, ensuring that all paths are perfectly of equal length.

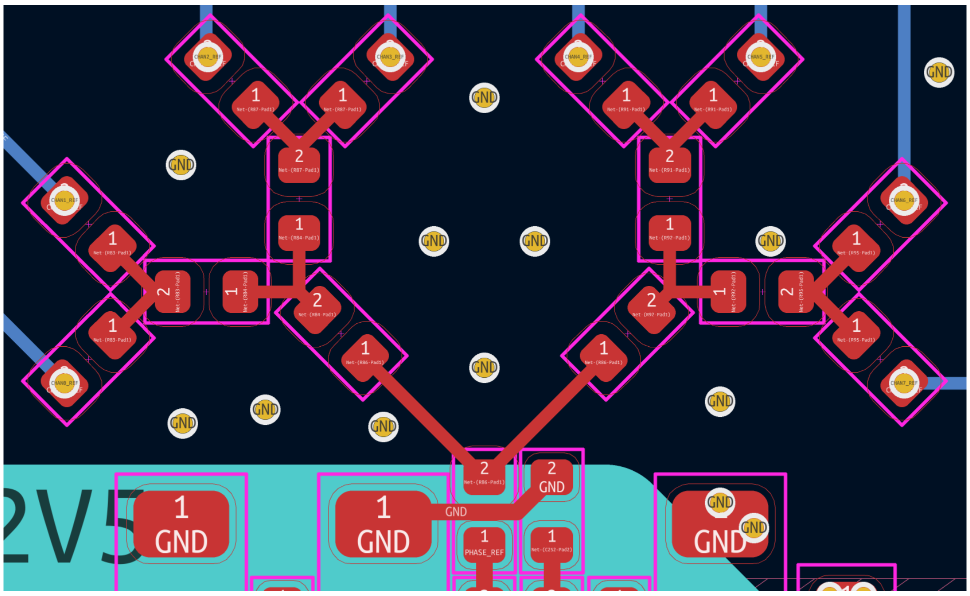


Figure 28: PCB layout of the 8-way resistive splitter used to split the phase reference to the 8 frontends. Layer 1 is represented in red, layer 4 in cyan and layer 6 in blue.

Finally, just like the tuner clocks, the phase references must be routed to the frontends making sure that all they all arrive to the front ends with the exact same phase. This is done by running the signals on the bottom layer of the PCB and adding length matching sections so that all traces end up the same length as shown in figure 29

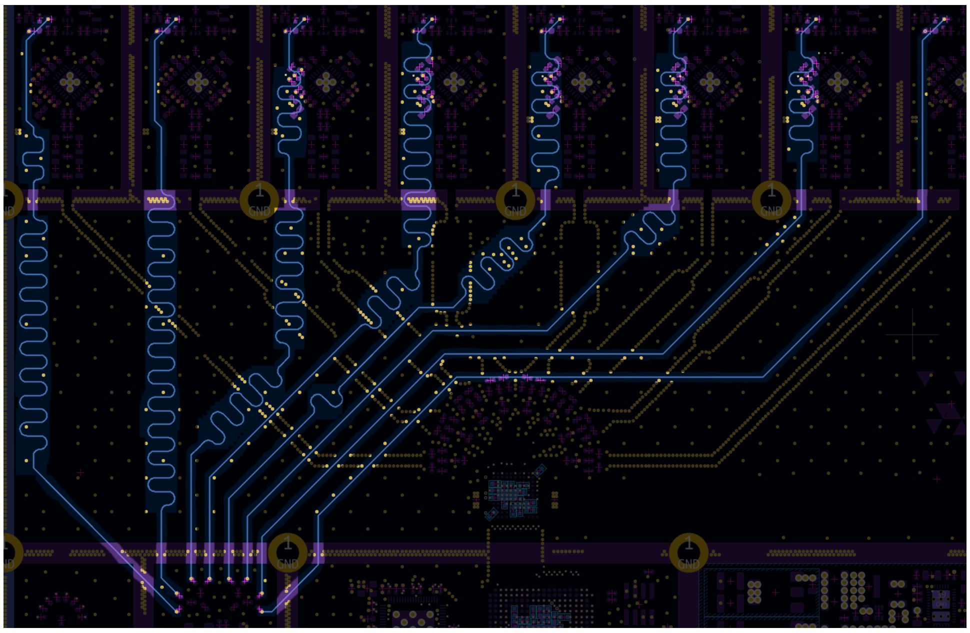


Figure 29: PCB layout of the traces going from the 8-way splitter to each frontend.

3.4 ADC

The Analog to Digital Converter (ADC) is the one chip that this entire design was built around. The MCP37211-200 [17] from Microchip features 8 channels multiplexed to a central 12bit 200MS/s core allowing a samplerate per channel of up to 25 MHz. Due to this chip's intended use in radar and sonar applications, it features a built in Digitial Signal Processor capable of time-shifting the samples of each channel such that they are virtual sampled at the same time. This allows to simulate a 8 channel non-multiplexed ADC but comes at the cost that the resampling filters use only properly time shift the samples within a certain frequency rangek, specifically up to approximately 9 MHz.

What truely sets this chip apart from the alternative is it's extremely low cost of only $27 in single quantities. This is what allows this receiver design to be viable in the first place. As discussed in the Background section, previous designs had to make due with either multiple very cheap ADCs which required external time synchronisation and calibration, or expensive multi-channel ADCs that were often too high performance for hobby or educational level hardware.

3.4.1 Anti-Aliasing Filter Design

First, the IF signals to be sampled must be adequately low-pass filtered to avoid out-of-band components appearing in the final passband. The design of this filter is based on three variables. First, the sampling rate of the ADC which in this case is 25 MHz per channel, second, the desired alias-free passband, which the manufacturer specifies as 0.57 MHz to 8.57 MHz and third, the desired minimum attenuation of these out of band signals.

The sampling frequency being significantly higher than the upper passband frequency as well as the way signals over $F_s/2$ fold back means that the filter doesn't need to be as sharp as one would expect. Indeed, the frequency F_d at which a signal of frequency F_a in nyquist zone n appears in the digital signal after sampling at sampling rate F_s is given by equation 2.

$$F_\mathrm{d} = \pm\left[(n-1)F_\mathrm{s} - F_\mathrm{a}\right]\begin{cases} + \text{ if } n \text{ even} \\ - \text{ if } n \text{ odd}\end{cases} \qquad F_\mathrm{a} \in \left[\frac{n-1}{2}F_\mathrm{s}, \frac{n}{2}F_\mathrm{s}\right] \qquad n \in \mathbb{Z}_0^+ \tag{2}$$

The frequency of signals in the second nyquist zone ($n = 2$) necessary to reach the 9 MHz upper limit of the passband is therefore trivially computed assuming $F_\mathrm{d} = 8.57$ MHz and $F_\mathrm{s} = 25$ MHz giving $F_\mathrm{a} = 16.43$ MHz which is the frequency at which the anti-aliasing filter will be expected to provide the desired minimal out of band attenuation.

The required out of band attenuation can be deduced from the tuner and ADC specification. Ideally, one should base the antialiasing filter specifications on the specifications of the filters higher up the RF chain. Unfortunately, the manufacturer of the tuner does not provide any such data. However, as previously described, the tuner is originally meant for DVB-T usage, and us thus, reports the minimum adjacent channel attenuation that it is capable of. This value can be used along with the ADC's maxmimum Signal to Noise ration to compute the attenuation at which an out of band signal will be below the ADC's noise floor. The tuner reports a minimum adjacent channel rejection of -47dBc, and the ADC supports an SNR of 71.33dBFS. Therefore, the anti-aliasing filter must provide at the very least $71.33 - 47 = 24.33$ dB of attenuation atthe previously computed 16.43 MHz point.

With all the parameters of the requirements of the filter in hand, it is now possible to select a filter type. Before that however, one has to chose between an RC and LC filter. In terms of cost, an LC circuit is naturally more expensive than a RC filter as inductors, especially precise ones, are more expensive. However, they feature twice as fast amplitude drop off and no passband attenuation, therefore a LC filter is chosen for this use case. There exist many types of LC low-pass filters, a comparison of which is shown in table 3.

Type	Roll-off rate	Passband ripple	Group delay variations	Ultimate attenuation
Bessel	Very low	None	Very low	Unbounded
Butterworth	Low	None	Small	Unbounded
Legendre	Low	None	Moderate	Unbounded
Chebyshev	Moderate	Present	Moderate	Unbounded
Elliptic	High	Present	Large	Bounded

Table 3: Qualitative comparison of common LC low-pass filter types at a given order sorted by roll-off rate from slowest to fastest as described in [?].

From this comparison, it is possible to understand that a fast roll off comes with numerous trade off. Firstly, filter types with a higher roll-off rate have passband ripple. This, if kept within reasonable bounds is not problematic. The second issue are group delay variations. In the case of audio processing, this is generally not problematic as the human ear cannot hear phase information, however, for a modulated digital signal, a nonlinear phase will cause distortion and thus increase the bit error rate. The final investigated parameter is ultimate attenuation, and it is shown that it is bounded for elliptic filters, which means that the filter has an attenuation value that it will not exceed no matter the input frequency.

As such, a Chebychev filter is chosen due to its higher roll off rate while featuring only moderate group delay variations and an unbounded ultimate attenuation. From here, it is now possible to determine the order necessary to achieve the required attenation. For this, a filter calculator tool by Marki Microwave[?] was used to compute filter designs of increasing order. The tool allows to specify a few more parameters then previously established, such as impedance and maximum allowable passband ripple. For the impedance, the same impedance as the tuner of 2 kΩ differential preferable. However, since the filter design tool is meant for single-ended filters, the impedance is specified as1 kΩ, after which the components will be mirroed to form a differential filter. For the passband ripple, a standard 0.1 dB value is used. One important thing to note is the filter is configured to limit its component choices to standard 5% precision values, ensuring manufacturability with off-the-shelf components. The results from the tool are as shown in table 4.

Order	Attenuation at 16.43 MHz
1	−0.278 dB
2	−1.021 dB
3	−12.529 dB
4	−20.882 dB
5	−32.575 dB
6	−42.042 dB

Table 4: Results of the filter calculator tool for different filter orders

These results show that in order to achieve the previously described required attenuation, a filter order of at least 5 is necessary. Going with an order higher than 5 would only add more components and cost to the design, so 5 is chosen as the filter order for this design. The component values generated by the calculator tool, after mirroring the filter circuit to create a differential circuit are as shown in figure 30.

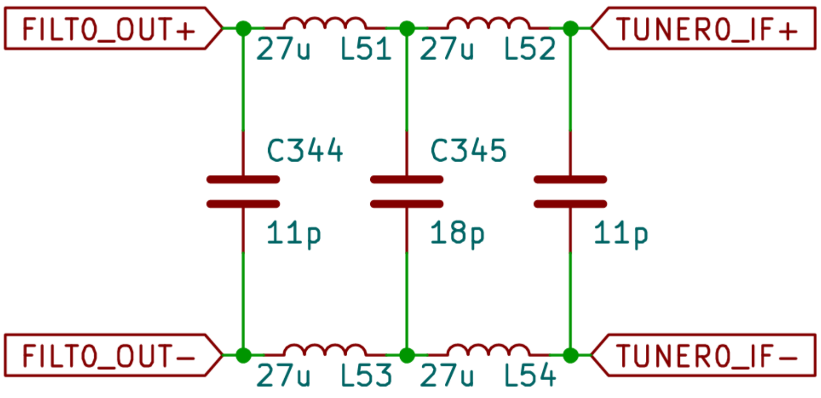


Figure 30: Circuit of the 5th order Chebyshev low-pass filters designed for use as anti-aliasing filters in front of the ADC.

In order to verify that the values generated by the tool result in a filter with the desired response, the circuit is first simulated using the LTspice[?]. This yields the filter response shown in figure 31.

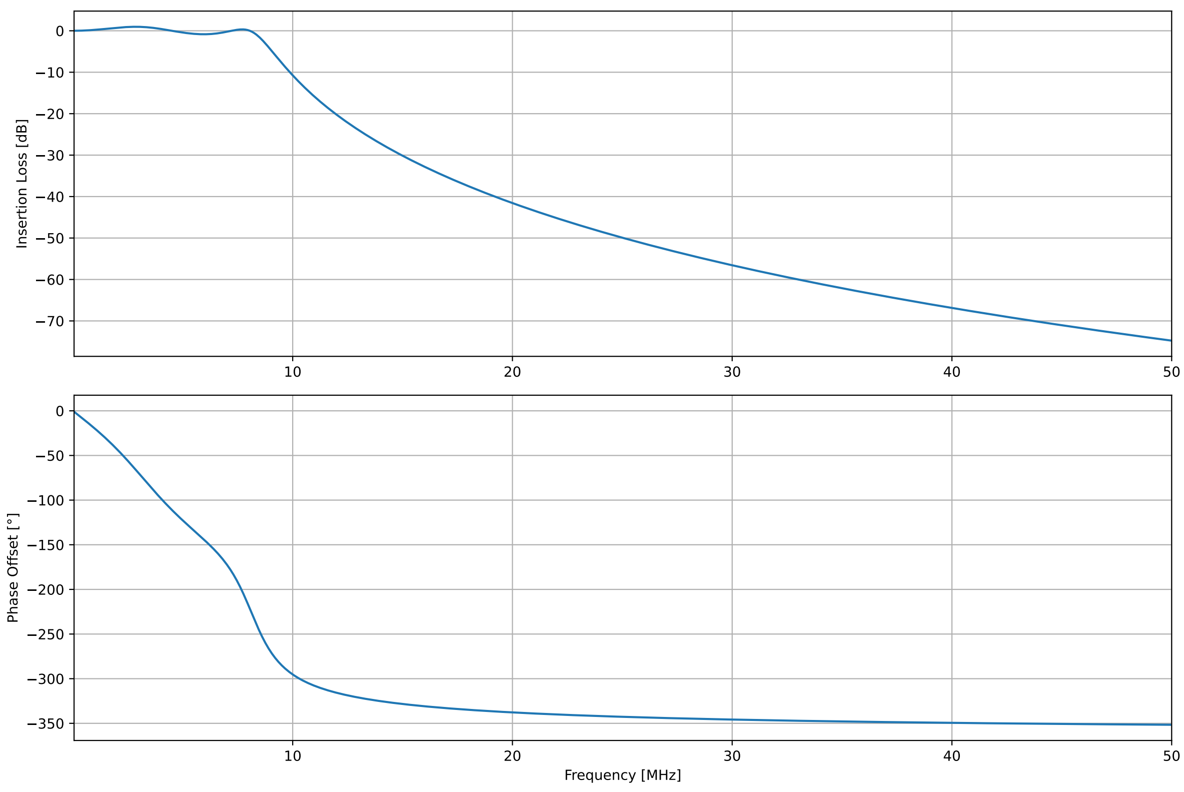


Figure 31: Bode plot of the anti-aliasing filter as computed by the LTspice circuit simulator software.

The simulation results show that the attenuation is even slightly higher that what the Marki Microwave filter design tool. This is almost certainly because of slight manual selection errors when selecting the position of the marker for 16.43 MHz. Next, the passband ripple is clearly over the 0.1dB value configured in the tool. This is likely due to the fact that the tool had to chose standard component values which are slightly off the theoretically ideal values. Finally. the 3dB corner frequency is computed to be 8.754 MHz. This is slightly different from the 8.57 MHz corner frequency that the tool was instructed to use, but this is again explained by the tool being forced to chose standard component values.

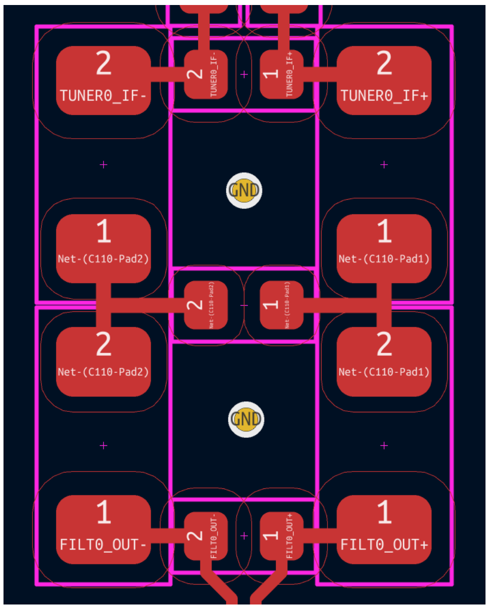


Figure 32: PCB layout of the anti-aliasing filter.

Finally, the layout of the anti-aliasing filter is as shown in figure 32. The anti-aliasing filter is layed out with the shunt capacitors in between the inductors in order to reduce the total length of the filter. The anti-aliasing filter are placed inside of the tuner compartment in order to reduce cross-talk between channels as much as physically possible.

3.4.2 Input Driver

Coupling the output of the anti-aliasing filter to the ADC requires two main steps. First, the ADC requires a precise common mode offset on its input that it generates by itself. Second, a kickback filter is required in order to minimize the amount of ADC sampling noise that makes it back to the tuner through the IF filter.The circuit create to drive the ADC's input is as shown in figure 33.

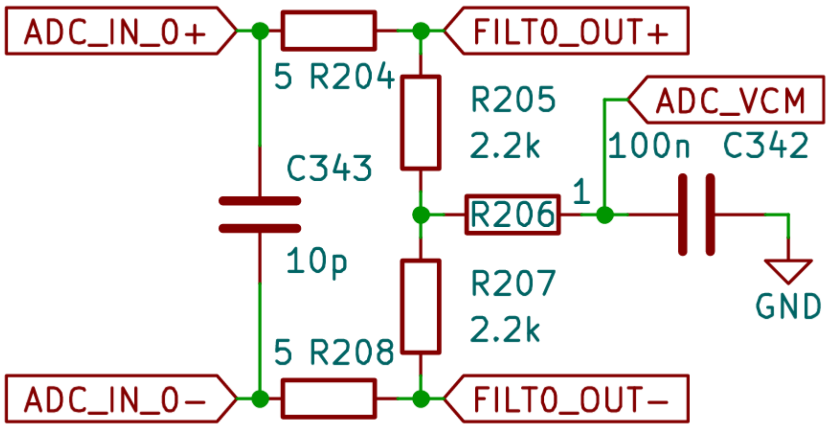


Figure 33: Driver circuit for the ADC.

Traditionally, this is accomplished either by using a dedicated ADC driver amplifier, or by using a transformer. An ADC driver amplifier would be relatively expensive as 8 separate units would be required. Transformers would have the same issue, with the added problem they they take up a significant amount of space on the PCB. Therefore, a fully passive driving circuit is designed. Starting with the kickback filtering capacitor, its value was chosen to be as large as possible without significantly reducing the bandwidth of the ADC's input. 5 Ω resistors are added in series with each branch as per the manufacturer's recommendation to reduce ringing which would be caused by the high transient currents when the sampling switches close. Then, for DC offset injection, a resistor divider between the positive and negative branches. The value of the resistors is chosen such that the total impedance as seen by the the tuner is close to 2 kΩ.

3.4.3 Power Conditioning

A clean power supply is paramount to ensuring a low noise and spur free digitized signal. This is achieved in a two step process. First, both power rails of the ADC are generated for it alone by linear voltage regulators which drastically isolates the ADC from low frequency noise due to the closed-loop nature of linear voltage regulators. Secondly, power to the analog power pins of the ADC passes through ferrite beads which in turn isolates it from high frequency noise and decoupling capacitors are placed on all power pins. The circuit of the power conditioning section after the linear regulators is as shown in figure 34.

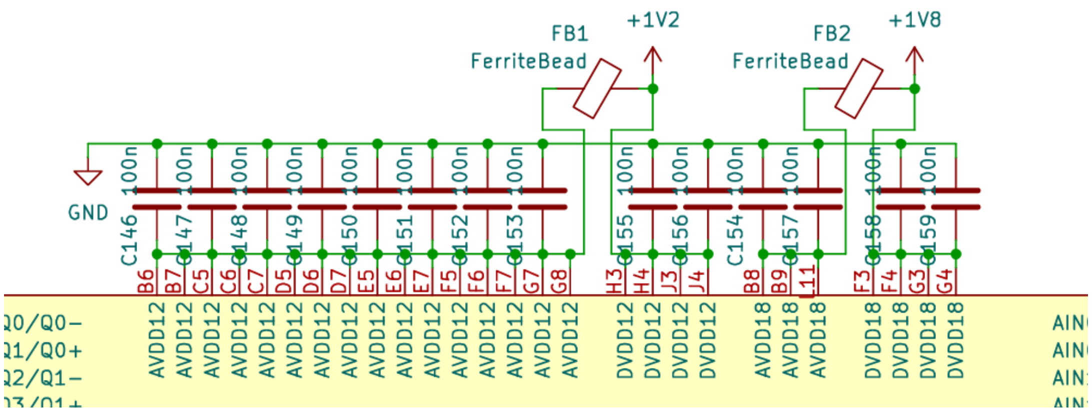


Figure 34: Power conditioning circuit for the ADC.

The layout of the ADC's power circuitry is as shown in figure 35. The power rails arrive on the forth layer and connect directly to the digital supply pins as well as to one side of the ferrite beads. After passing through the ferrite beads, the supply rails make their way to the analog supply pins also via the fourth layer. All decoupling capacitors are placed on the back side of the PCB so that their pads can be behind, or as close as possible to the BGA pads ensuring that their path to the pads is as low impedance as possible. This does mean that during manufacturing, an adhesive must be used to keep the capacitor fastened to the board during the reflow soldering process as they will be hanging upside down.

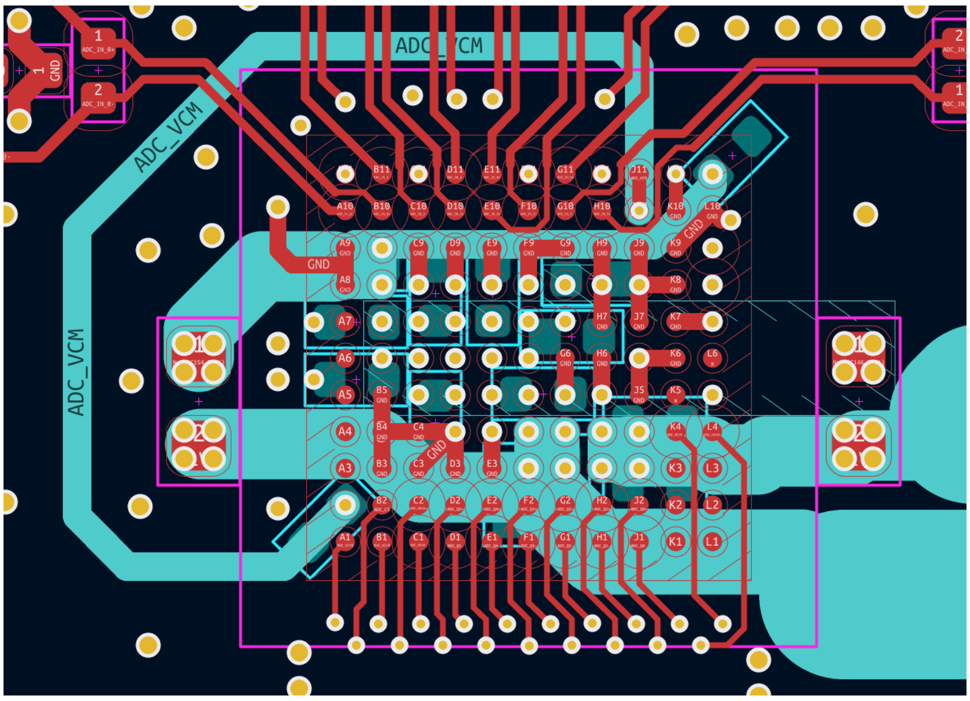


Figure 35: PCB layout of the ADC and its associated power conditioning components.

3.5 FPGA

In order to process samples from the ADC and interface with the USB FIFO, some sort of Digital Signal Processor (DSP) is required. In the past, the solution would generally have been to use a specialized DSP chip such as Analog Device's *SHARC*[6], Texas Instrument's *TMS320*[32] or NXP's *VSPA*[20]. Nowadays, with the advent of low-cost, high LUT count FPGAs, such an approach has become obsolete.

Serving as the only processing chip on the receiver is Lattice's *LFE5U-85F*[14], a low cost 85k LUT FPGA. This FPGA was chosen for its extremely low cost and, most importantly, the fact that a Free and Open Source toolchain[27] exists for it. This makes developing designs for this FPGA cheap since no licensing costs are required for the development tools. It is important to note that the 85k LUT variation is likely over-specified, but it was chosen as this is a prototype board and running out of space would have been highly problematic.

3.5.1 Configuration Memory

In order for the FPGA to load its design at startup, an external configuration flash memory is required. Indeed, most FPGAs do not come with non-volatile configuration memory and require either an external flash or a micro-controller to program it after a power interruption or hard reset. In this case, since the microcontroller will be a soft core inside of the FPGA, a configuration flash is chosen.

The FPGA's manual lists the minimum recommended flash sizes for each of the FPGA variants and does not go over 64 Mbit. However, most development boards for this chip, including Lattice's own evaluation board use a 128 Mbit flash chip. As such, since this board is a prototype, it is chosen to also use a 128 Mbit flash chip.

Winbond's *W25Q128JVP* was chosen as it is one of the most widely used and available SPI flash memory chips for the ECP5 line of FPGAs, being used by Lattice's evaluation board as well as third party development boards such as OneBitSquared's OrangeCrab[22] which was used for early development of the firmware for this project before the PCBs were manufactured. The circuit for the configuration memory is as shown in figure 36.

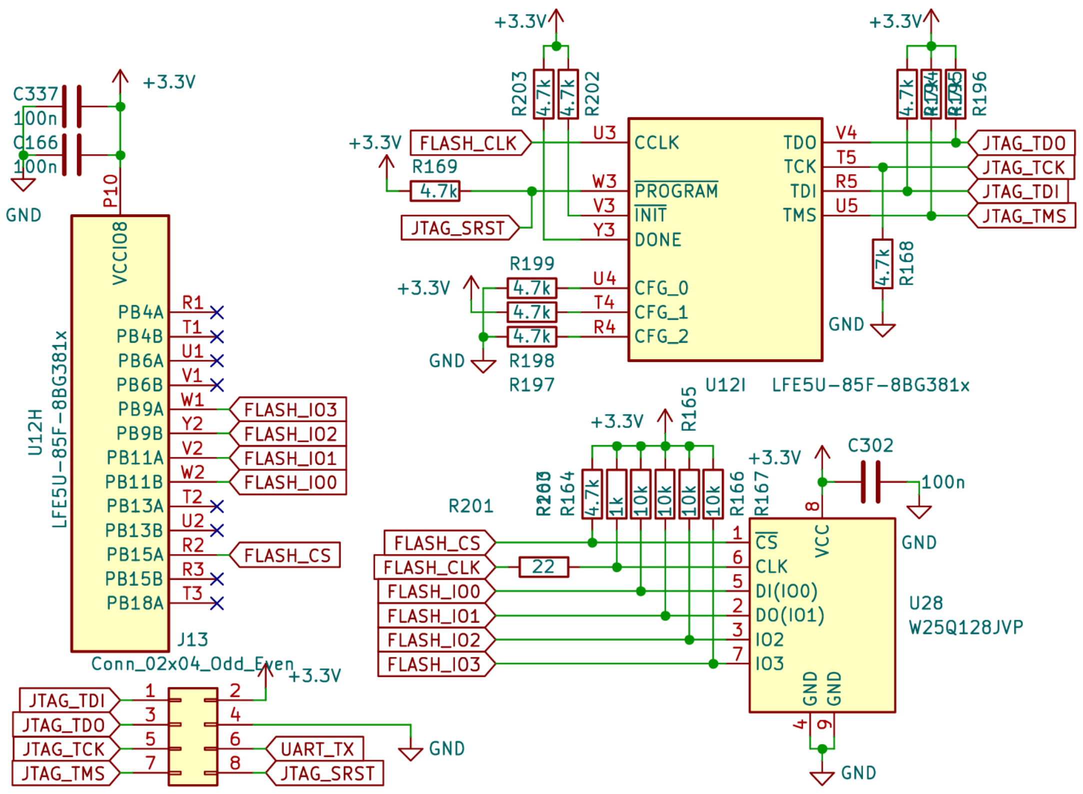


Figure 36: Circuit of the FPGA configuration memory and JTAG debug port.

The configuration flash chip is directly connected to the FPGA through a QSPI interface. The data and clock lines are all pulled up to ensure they remain in a safe state before the FPGA enters configuration mode where they would otherwise be floating. The power to the flash chip is as usual decoupled with a 100 nF capacitor.

Next, the configuration pins `CFG_0` through `CFG_2` are tried in such a way to encode `0b010` which sets the FPGA in SPI master mode. This mode is the one that enables the FPGA to talk to an internal flash chip to load its configuration at power on or after a reset. The `INIT` and `DONE` pins are unused in SPI master mode and are thus pulled high as per the manufacturer's recommendations.

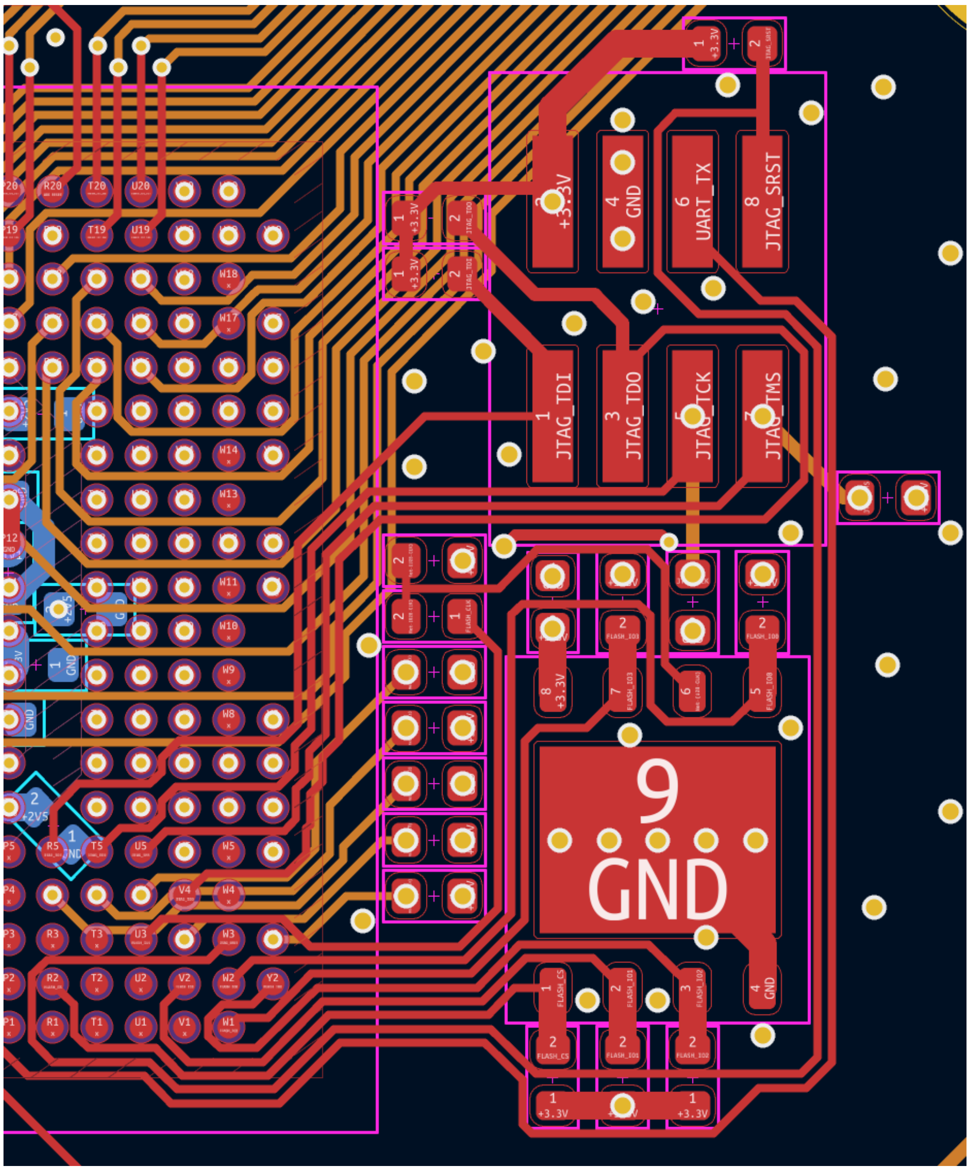


Figure 37: PCB layout of the FPGA configuration memory and JTAG debug port. Layer 1 is represented in red, layer 3 in orange and layer 6 in blue.

As shown in figure 37, the decoupling capacitor of the flash chip is placed immediately in front of its power pin. All the various pull ups and pull downs are placed whereever made sense from a layout perspective as their exact location is not important. All signals are run on the first and third layers.

3.5.2 JTAG Programming and Debugging Port

In order to program the FPGA and flash memory, as well showing debug messages through a UART interface, a JTAG port is added to the PCB. JTAG allows not only to program the FPGA and flash memory but also to probe any IO pin under the package making it useful for testing of PCBs during manufacturing. The circuit of the JTAG port is also shown in figure 36.

The pull ups and pull downs are used to keep the jtag port in a safe state when no debugger is connected. Indeed, `TCK` is pulled low as a rising edge changes the state of the JTAG state machine and `TMS` is pulled high because keeping TMS high and clocking the input resets the JTAG state machine. The other pins are pulled high as per the manufacturer's recommendation. Due to how trivial this part of the circuit is, the layout will not be discussed.

3.5.3 The FPGA itself and 1PPS input

Outside of the configuration memory, the FPGA requires very little external circuitry. As shown in figure 38, all power pins are as usual decoupled using 100 nF capacitors. Additionally, an SMA connector with ESD protection diodes is connected directly to a pin of the FPGA to use as a One Pulse-Per-Second (1PSS) input for GPS time synchronisation of the samples.

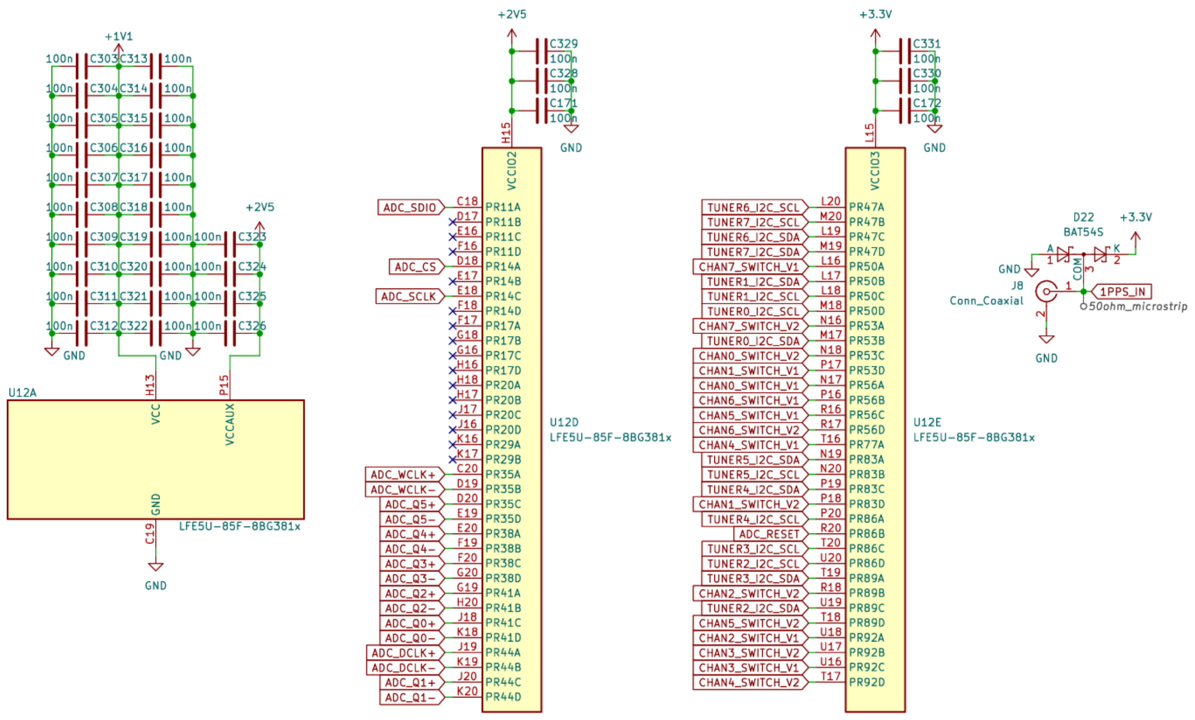


Figure 38: Partial circuit of the FPGA section showing the connections to the ADC, tuners and the 1PPS input.

Using this 1PPS would allow multiple distributed receivers to work together to perform Time Difference of Arrival analysis, allowing for wide area localisation of signals where the angle of arrival may be modified by multipath propagation. This technique is already in use through open networks such as the KiwiSDR network[11].

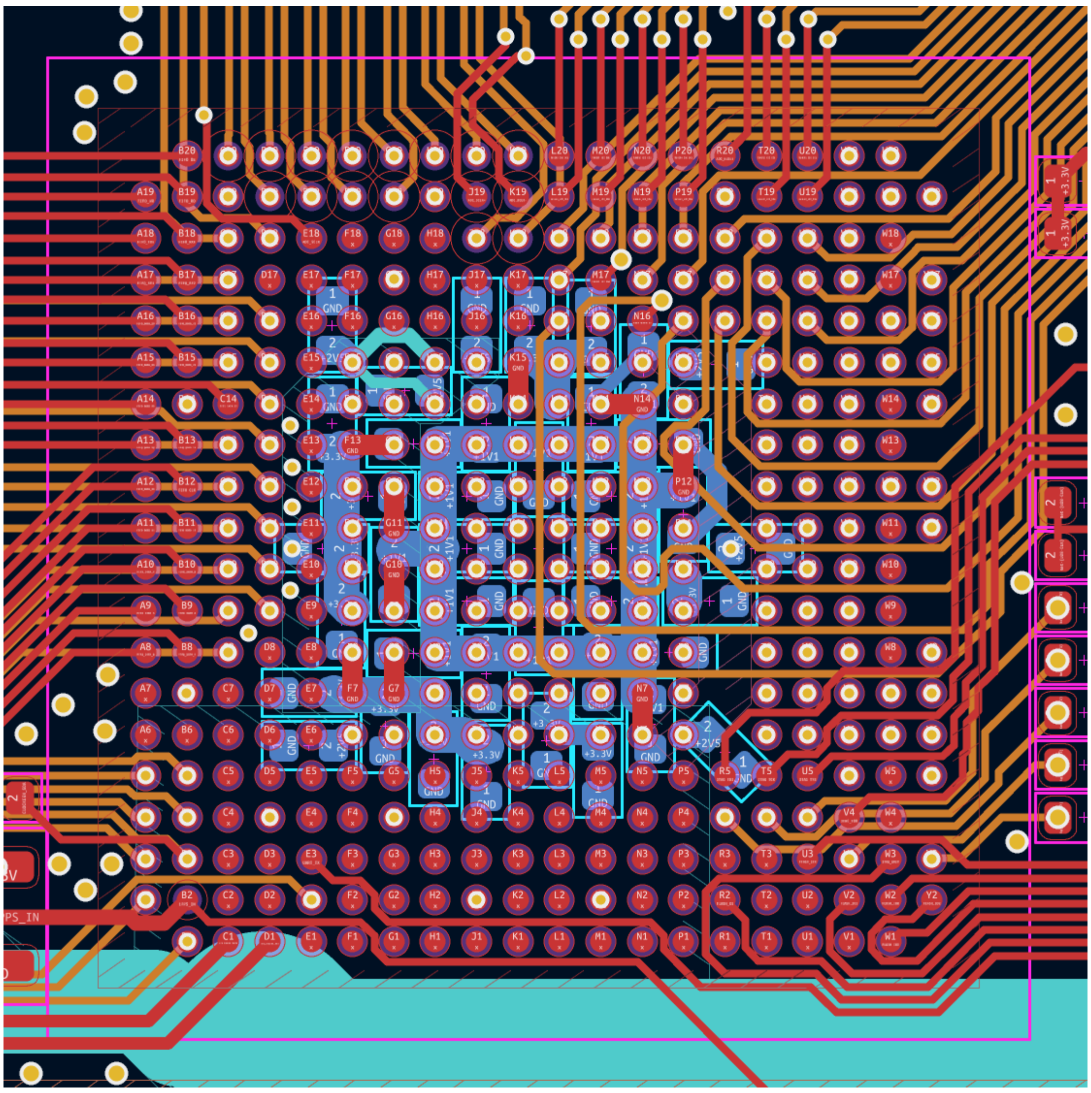


Figure 39: PCB layout of the under side of the FPGA showing the decoupling capacitors and via-on-pad technology. Layer 1 is represented in red, layer 3 in orange, layer 4 in cyan and layer 6 in blue.

One of the more challenging things about using an FPGA is the fact most use a BGA package. While common in industrially produced devices, they require specialized equipment to solder and, as shown in figure 39 require special layout considerations. First of all, the top layer ground plane fill must be excluded under the BGA to avoid disrupting the shape of the pads and creating thermal inconsistencies during reflow soldering. Next, power must be routed to each power pin with the lowest impedance possible. To do this, large polygons are used to carry 3.3 V on layer 5 and 1.1 V on layer 4. Additional traces are routed on layer 4 to carry the 2.5 V power rail. To ensure they have the lowest impedance path possible to their respective pins, the decoupling capacitors are mounted on the under side of the board directly, or as close as possible to, the via leading to the BGA pad.

Finally, all IO signals must be fanned out. Traditionally, this would have been done by placing vias in between pads of the BGA and connecting each via to its corresponding pad. This had the advantage of not requiring any specialized PCB manufacturing process but wasted space and added additional resistance and inductance to every pin. Nowadays, most PCB manufacturers, including JLCpcb, support *Via-on-Pad* technology which consists in plugging the vias with an epoxy material then plating over the vias to create a smooth pad surface that components can be soldered to normally. Via-on-Pad remedies the issues previously mentioned and makes routing significantly cleaner and easier.

3.6 USB Interface

3.6.1 USB FIFO

While it would have been possible to directly interface to USB using the FPGA, this would have been extremely time consuming and requires using an FPGA with a very high number of logic blocks. Therefore, it was decided to instead use a dedicated USB interface IC. Before selecting an interface IC, it is first necessary to know how much data will need to be moved around.

It is known that the ADC runs at 200 MHz and has a bit depth of 12 bits. The amount of data that has to be sent back to the PC is thus trivially computed as $2 \cdot 10^8 \cdot 12 = 2.4\,\mathrm{Gbit\,s^{-1}}$. This not only confirms that, as described previously, 1 gigabit ethernet would not have been the adequate interface, but also means that USB 3.0 will have to be used

Two of the most prominent ICs used for interfacing FPGAs with FPGAs are Cypress' FX3 and FTDI's FT601. Both of these ICs are capable of transporting up to $3.2\,\mathrm{Gbit\,s^{-1}}$ but the FX3 has several disavantages. First, it requires proprietary tools to program it's interface controller. While this is useful to adapt the chip to different use cases, it adds a level of complexity to the design that the FT601 does simply does not require. Second, and most importantly, the FX3 is widely known to be unreliable with some PC USB controllers [19]. As such, the FT601 is chosen to interface the FPGA with USB 3.0.

The FT601's interface consists in a simple 32bit wide parallel interface. It provides signals such as "TX Empty" and "RX Full" which tell the FPGA the state of its built in RX and TX fifos. It also provide its own selectable 66 MHz or 100 MHz clock which the FPGA must use to receive to send data. The schematic of the USB FIFO section is as shown in figure 40.

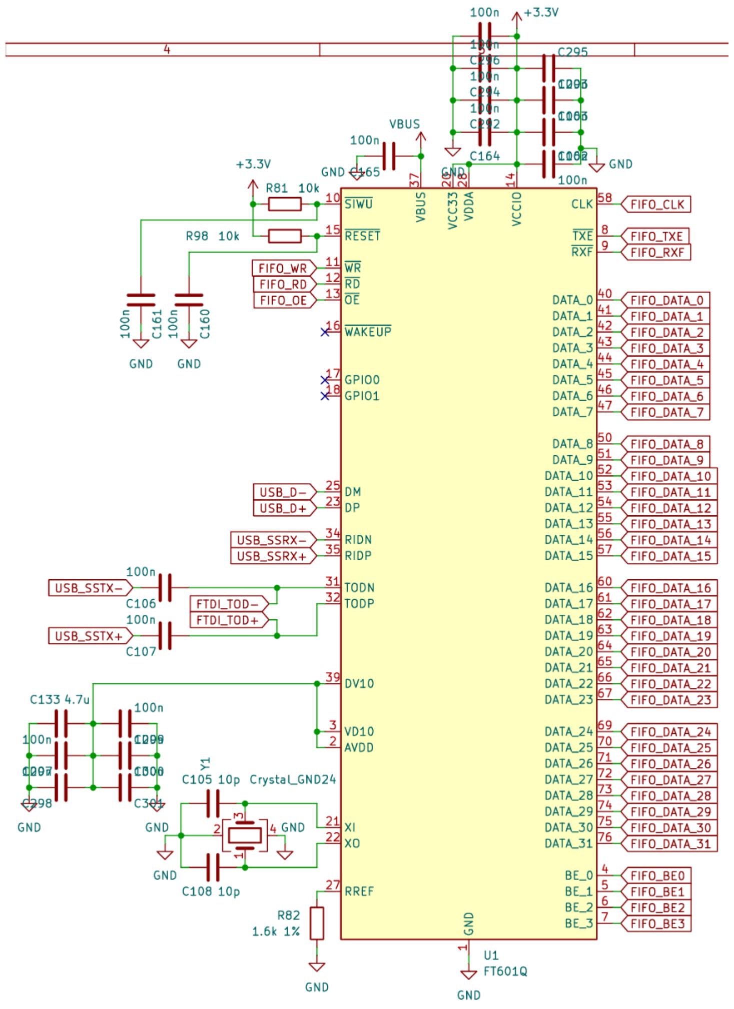


Figure 40: Schematic of the USB FIFO section.

Starting with the power rails, each power input input receives its own decoupling capacitor of which the value recommended by the manufacturer of the IC is used. The chip also contains a built-in Low 1 V Low DropOut regulator who's output is also decoupled and routed back to the 1 V input pins. The `SIWU` and `RESET` pins are unused so they are tied to VCC using standard 10 kΩ resistors. Next, the 30 MHz crystal is connected to the chip in parallel with loading capacitors specified by the manufacturer of the crystal. Although it would have been helpful to use the clock generator to drive the FT601 instead of a crystal, the manufacturer specifically states that this is not possible. Finally, a precision 1.6 kΩ resistor is connected between the dedicated reference resistance pin `RREF` and ground. This is used by the FT601 in combination with its internal current source to set the reference voltage for its USB 2.0 interface. Finally, all FIFO signals are connected directly to the FPGA.

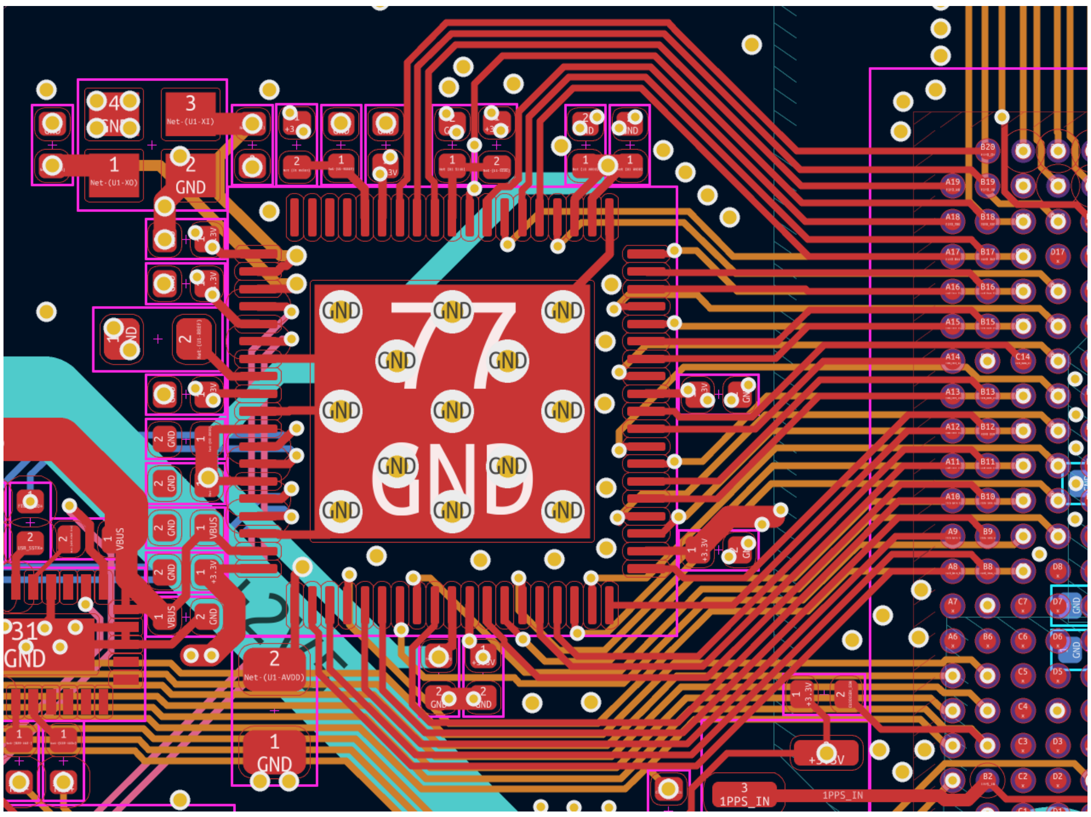


Figure 41: PCB layout of the USB fifo section and its connections to the FPGA. Layer 1 is represented in red, layer 3 in orange, layer 4 in cyan and layer 6 in blue.

As shown on figure 41, the decoupling capacitors are as usual placed as close as possible to their respective pin in order to minimize the impedance of the trace. The 3.3 V power rail for the chip arrives from layer 5 and the internally generated 1 V is routed on layer 4. A dozen large vias are placed on the ground pad of the FT601. This is done both to guarantee a very low impedance path to ground, but also for thermal management to ensure that the IC is able to dissipate its heat through the ground layers of the PCB. The 30 MHz crystal and its loading capacitors are also placed as closed as possible to the IC to reduce unwanted emissions, and to reduce the emissions even further, its connections are routed through the third layer and shielded by ground vias.

Since the FIFO runs at a relatively low clock frequency of 100 MHz, matching the length of all of its signals is not critical. There was however an attempt to keep the length similar. Due to the high number of signals that had to be routed, having a perfect length match would not have been possible anyway. The largest length difference is between data bits 0 and 29 at 11.31 mm. At the 100 MHz frequency of the clock, this corresponds approximately to a phase difference of 1.3° which is negligible.

3.6.2 USB Type C circuity

In recent years, the USB Type C connector has slowly imposed itself as the industry standard allowing for every devices, no matter the manufacturer, to be inter-compatible between cables, chargers and power supplies. As such, this connector type was chosen for both the data port.

While older USB connectors could be directly connected to the FIFO IC, USB Type C is reversible and thus requires a bit more logic in order to work properly no matter the plug orientation. For this, a *HD3SS3220* USB-C SuperSpeed Mux and Flip Detection IC from Texas Instruments is added between the Type-C connector and the FIFO IC. This chip detects the orientation of the plug and selects the appropriate pairs for RX and TX. Such an multiplexer is not needed on the USB 2.0 signals of the Type-C connector as the USB Type-C cable only contains one pair for USB 2.0. Therefore, the top and bottom pins of the connect can simply be shorted together without another unconnected pair in the cable interfering with the impedance.

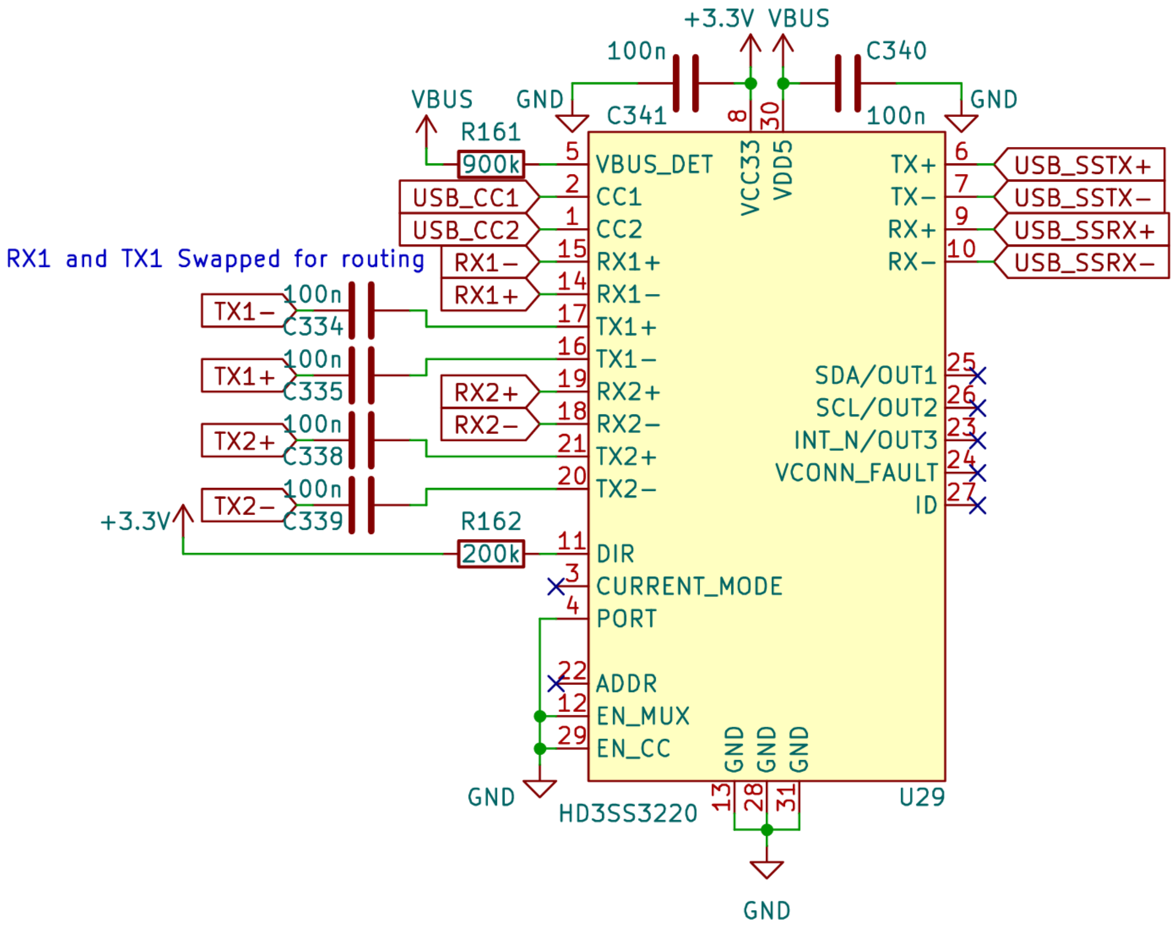


Figure 42: Schematic of the USB Type-C SuperSpeed pair multiplexer circuit.

The circuit for the multiplexer is as shown in figure 42. Just like every other component in this circuit, the power inputs of the multiplexers are decoupled with 100 nF capacitors. The transmit pairs going to the USB Type-C connector go through DC blocking capacitor as per the USB 3.0 standard. The receive pairs do not have decoupling capacitors on this board as the standard specifies the transmitter on the PC side should have implemented them. The `EN_MUX` and `EN_CC` enable signals are tied to ground to enable the chip. An important thing to note is that USB SuperSpeed has the helpful features that the polarity of the pairs are autodetected on device connection. This means that PCB designers are free to swap the polarity of the pairs in any way their like to make routing easier, and this was made use of in this design.

The `VBUS_DETECT` pin is pulled up to the 5 V rail of the data USB connector so the chip can detect when it has been plugged or unplugged. The resistor used for this pull-up has a rather unusual value of 900 kΩ but the manufacturer of the chip specifically request this value. Finally, the I2C control bus of the chip is unused and thus left floating.

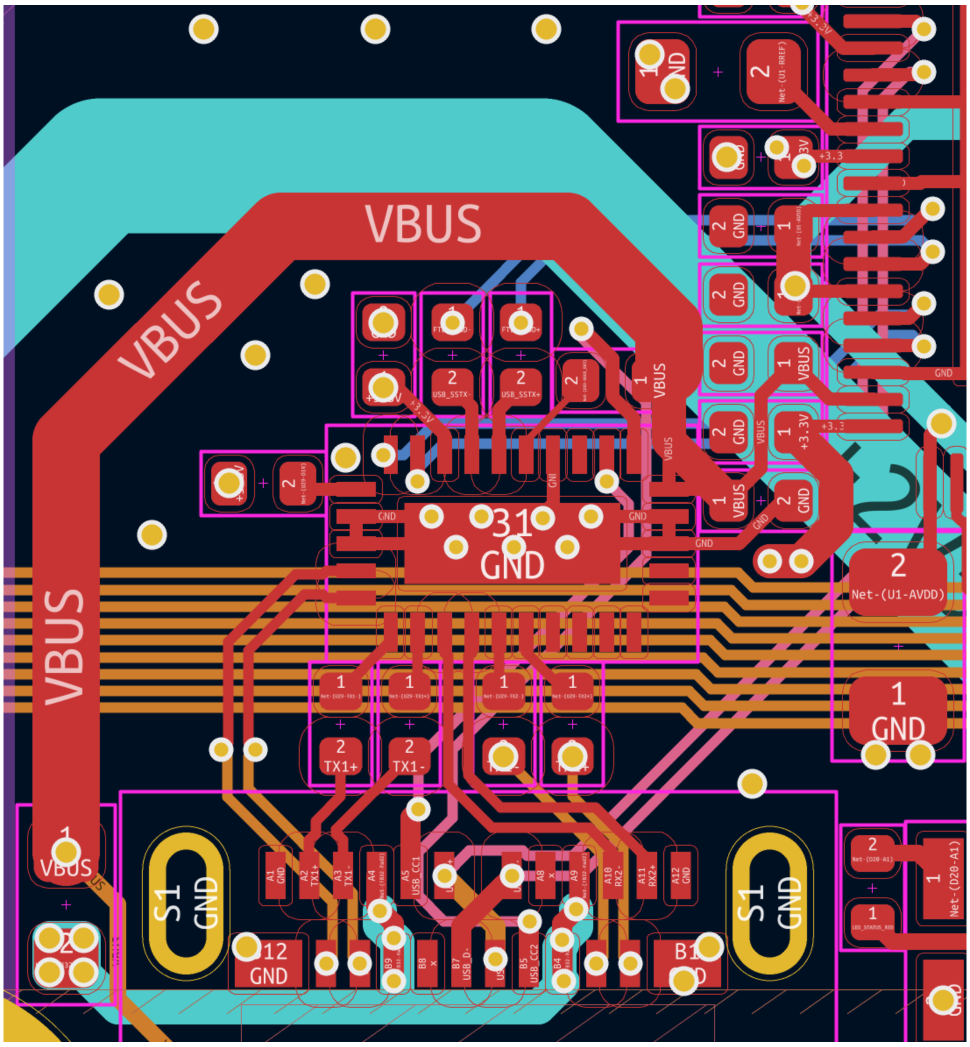


Figure 43: PCB layout of the USB Type-C SuperSpeed pair multiplexer circuit. Layer 1 is represented in red, layer 3 in orange, layer 4 in cyan, layer 5 in pink and layer 6 in blue.

As can be seen on figure 43, the decoupling capacitors are as usual places immediately in front of their associated power pin. As for the USB SuperSpeed pairs, they are kept as short as possible and care is taken to make sure that each side of the pair is of the same length. This is done because any mismatch in the length of the sides of the pairs would make them unbalanced which would in turn cause impedance and signal integrity issues. Thankfully, since USB SuperSpeed only uses one pair in each direction and does not use a separate clock signal, the lengths of the pairs do not have to be matched to each other.

In this case, the pairs coming from the connector are routed directly to the multiplexers through the top layer. Thankfully, the designers of the chip made it so that the layout of the pins on the chip closely match that of a standard USB Type-C connector allowing for easy routing between the two. The pairs then leave the multiplexer and are routed to the USB FIFO chip through the bottom layer. This was done in case there was a mistake when joining RX and TX to their respective ports on the FIFO. Indeed, the datasheet was not very clear on whether the names of the pins were from the point of view of the computer or of the FIFO. Routing them on the bottoms would have allowed for swapping them over if necessary. Thankfully, the way they were routed was correct and no rework ended up being necessary.

3.7 Power Supply

This section discusses the power supply circuit of the receiver. First, the power requirements are assessed by tabulating the requirements of all the active components in the circuit. Next the design of the power regulation circuitry is discussed. Finally, the USB Type-C power port implementation is discussed.

3.7.1 Power Requirements

Before the power supply can be designed, the power requirements of the entire device must first be assessed. As such, table 5 lists the required voltages and currents all significant IC on the device. It is important to note that for the FPGA, the current consumption was taken as a worst case scenario. The actual current consumption depends heavily on the internal design and clock frequencies used meaning it could not be accurately estimated before hand. The datahseet of the FPGA does not even list the maximum current so other designs such as the OrangeCrab FPGA developement board[22] were used as a reference.

Chip	Count	1.1 V Rail	1.2 V Rail	1.8 V Rail	2.5 V Rail	3.3 V Rail
R820T2	8	-	-	-	-	200 mA
MASWSS0136	8	-	-	-	-	5 µA
MCP37211	1	-	532 mA	116 mA	-	-
Si5351c	1	-	-	-	-	70 mA
LMX2572LP	1	-	-	-	-	70 mA
HD3SS3220	1	-	-	-	-	1 mA
FT601Q	1	-	-	-	-	185 mA
LFE5U-85F	1	<2 A	-	-	<500 mA	<500 mA
W25Q128JVP	1	-	-	-	-	25 mA
LEDs	22	-	-	-	-	5 mA
Total	-	2 A	532 mA	116 mA	500 mA	2.561 A

Table 5: Power requirement of every IC on the device.

This summary shows that 3.3 V rail, being the main logic supply rail, will be quire heavily loaded at over 2.5 A. The 1.1 V rail powering the FPGA will also be quite heavily loaded at 1 A. The three remaining rails are however much lower current peaking at 531 mA for the 1.2 V rail.

3.7.2 Power Regulation

There exist two main technologies for voltage regulation: Linear and Switching. Both have their advantages and disadvantages which must be carefully investigated before choosing which option to go with. Linear regulators are cheap, compact, simple and generate a very clean output, but have the huge disadvantage of having to drop the voltage difference between their input and output resistively. This makes them extremely inefficient for high voltage drops which in turns means that they generate a lot of heat that has to be dissipated. Switching regulators on the other hand, are more expensive, require external components and generates dirtier power, but are much more efficient as their transistors operate in switching mode and thus do not dissipate much power.

As with most things in engineering, it is therefore concluded that the best solution is not one or the other, but a combination of the two. As such, a switching regulator will be used to drop the 12 V to 15 V coming from the power port to the 3.3 V main bus. Then, for the low current, noise sensitive voltage rails such as 1.2 V and 1.8 V for the ADC as well as ‘2.5 V‘ for the clock generator, synthesizer and auxiliary power of the FPGA. Since the main core voltage of the FPGA is 1.1 V and requires up to 2 A, a second switching regulator is used to drop the 3.3 V rail down again.

With this rough design in mind, it is now possible to compute the current requirement for each regulator. First of all, since linear regulators drop voltage resistively, we know that the input current will be equal to the output current. For the 1.1 V rail, a worst case efficiency e of 80% can be assumed and the input current can be computed using equation **??**.

$$P_{\text{out}} = P_{\text{in}}e, \quad P_{\text{in}} = V_{\text{in}}I_{\text{in}}, \quad V_{\text{out}}I_{\text{out}} \iff I_{\text{in}} = \frac{V_{\text{out}}I_{\text{out}}}{V_{\text{in}}e} \tag{3}$$

This gives an input current to the 1.1 V switching regulator of $\frac{1.1 \cdot 2}{3.3 \cdot 0.8} = 0.833$ A. Adding this up with the input currents of the three linear regulators and the current requirement of the 3.3 V rail, it can be concluded that the 3.3 V switching regulator will need to be able to output $2.561 + 0.532 + 0.116 + 0.5 + 0.833 = 4.542$ A.

The a portion of the circuit for the power regulation section is as shown in 44. Only the 2.5 V linear regulator is shown as the 1.8 V and 1.2 V regulator circuits are indentical with just the regulator part number changing between them.

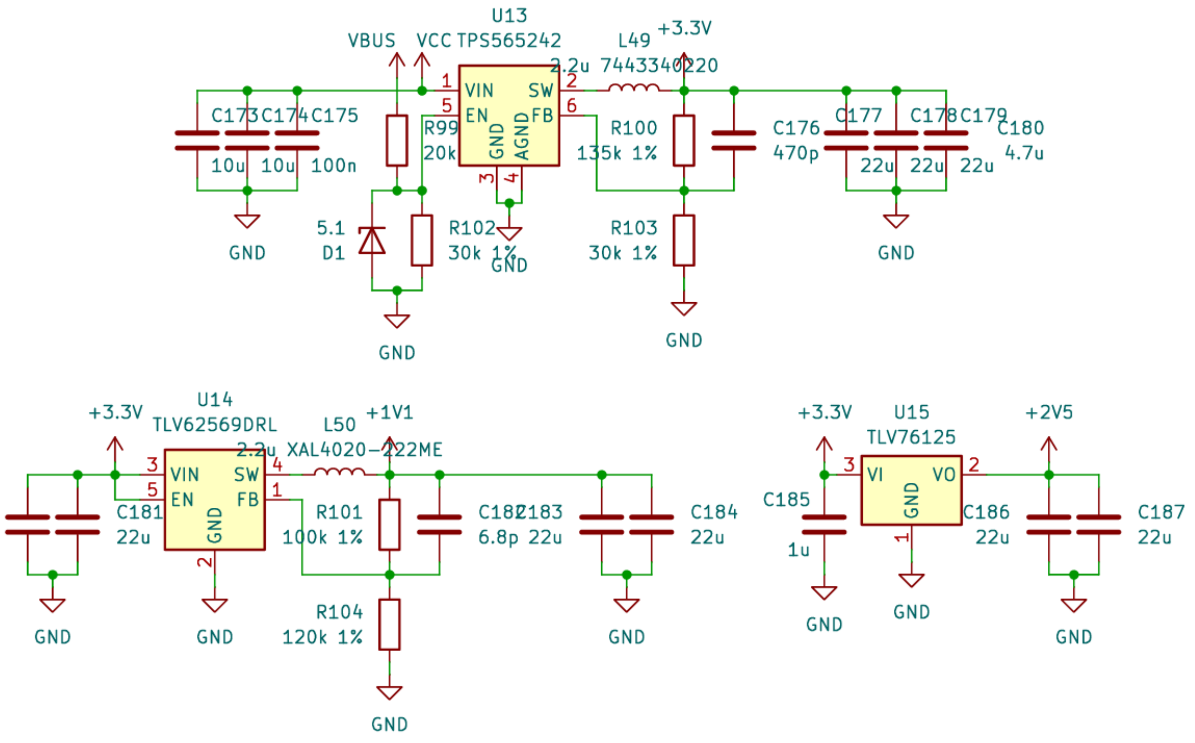


Figure 44: Circuit of the power supply section omitting the 1.2 V and 1.8 V linear regulators.

For the 1.1 V rail, Texas Instrument's *TLV62569* is chosen due to its low cost, high availability and fact that it contains a built in switching mosfet simplifying the design. While it supports a maximum output current of 2 A which, on the surface, does not leave any head room in current draw, one has to remember that the 2 A estimation was an absolute worst case scenario estimation obtained by looking at the current ratings on development board designs using that same FPGA. For the 3.3 V rail, Texas Instrument's *TPS565242* is chosen for the same reasons as the previous regulator. It features a maximum output current rating of 5 A which is around 450 mA higher than the expected worst case current draw, leaving a bit of headroom for current spikes and variations in parts.

The exact recommended schematic from the manufacturer was followed which specified all capacitors and inductors for the desired output voltage. Said voltage is determined using a feedback resistor divider with an additional feed-forward capacitor in parallel with the high side resistor. This capacitor serves to increase the loop gain at high frequency which reduces the output ripple. The value of this capacitor is again given by the manufacturer and cannot be computed manually as the details of the control loop are not public.

In the case of the 3.3 V switching regulator, the enable pin is controlled through the VBUS power rail of the USB data connector. This is done because the USB mux chip requires VBUS to come up before the 3.3 V rail. Control is implemented through a resistor divider in conjunction with a 5.1 V zener diode used to clamp the voltage at the regulator's enable pin in case of an over-voltage or ESD event on the USB data connector.

As for the linear regulators, Texas Instrument's *TLV761xx* series was chosen for it's high accuracy of 1% which is required for the ADC and the FPGA's 2.5 V supply rail. Unlike older 1117 series linear regulators, these devices are compatible with Low-ESR Multi-Layer Ceramic Capacitors (MLCCs) without risking instability making them ideal in applications where low ripple is essential. The only required components are input and output capacitors.

For the output capacitance, the manufacturer recommends a value between 1 µF and 220 µF. Since the exact dynamics of the current draw of the circuit cannot be known, it is not possible to compute the output capacitance as a function of the desired ripple. Therefore, the largest value that fit on the PCB was chosen which ended up being 44 µF split between two 22 µF capacitors. These capacitor values were chosen because were already required for the output capacitance of the switching regulators, avoiding the need for another different part.

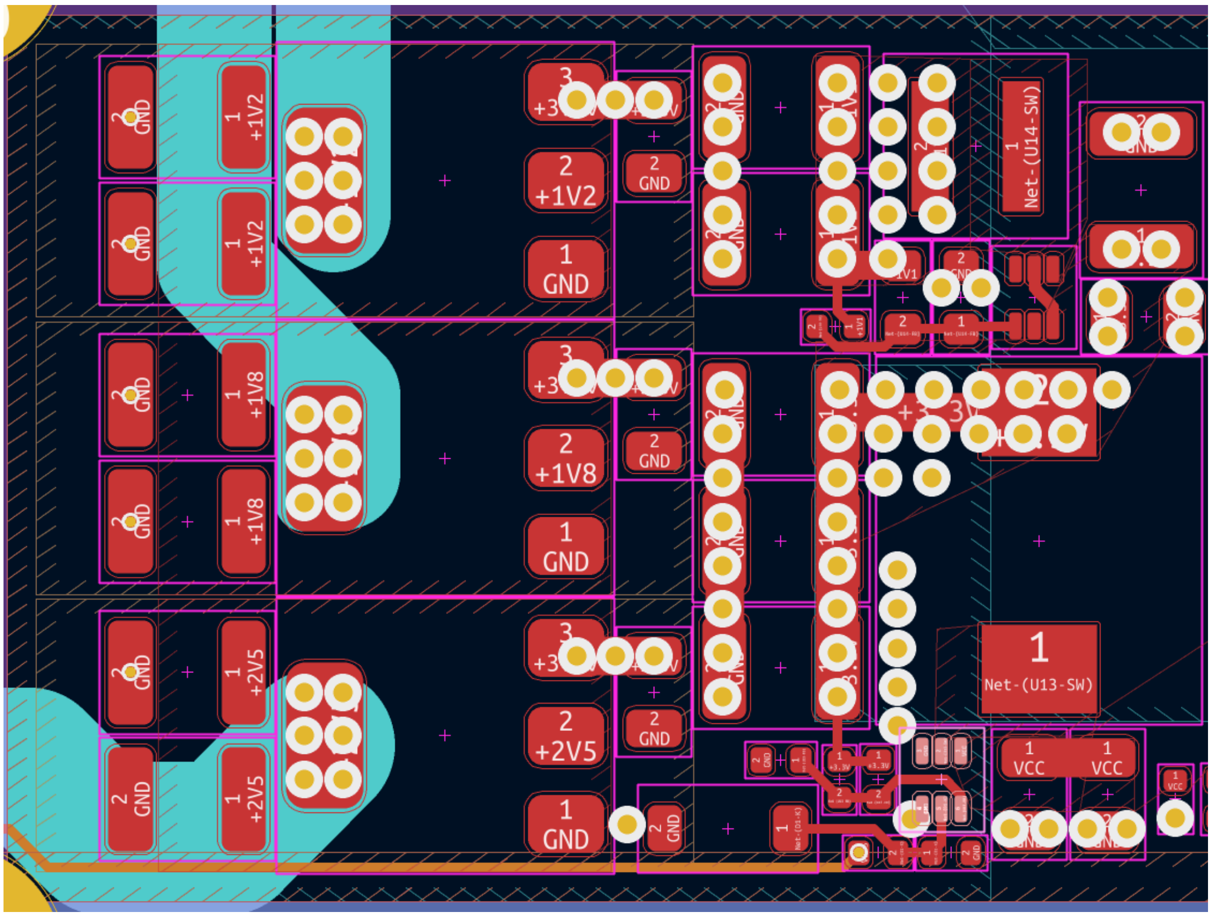


Figure 45: Layout of the entire power regulation section. Layer 1 is represented in red, layer 3 in orange and layer 4 in cyan.

The layout of the power regulation section is as shown in 45. All capacitors are placed as close as possible to the regulator pins and connects are made with copper polygons that are as thick as possible. The first layer is used for connections between components of the power supply, while internal layers are used as ground planes, heat sinks and to send power out of the regulator section. Specifically, layer 4 contains a large polygon tied to the 1.1 V volt rail as well as traces feeding 1.2 V and 1.8 V to the ADC, the trace feeding 2.5 V to the FPGA and the large polygon carrying 3.3 V to the tuners. Layer 5 is dedicated to the main 3.3 V power rail. All other layers are tied to ground, or, in the case of the polygons under the linear regulators, tied to the voltage rail of the regulators to serve as heat sinks.A high number of vias are used to create low impedance paths to the power traces and polygons and to help with thermal management.

3.7.3 USB Type-C Power Delivery

As mentioned in the Power Requirements section 3.7.1, this device requires a maximum of 15 W to operate meaning that it definitely will not be able to operate on bus power alone. The simplest way to provide power to the board would therefore be to include a standard power jack, but this has several downsides. Indeed, this means the device would have to be used with a dedicated power adapter making it hard to use in a mobile application. The need for a dedicated power adapter also raises ecological concerns. Indeed, why require a power adapter when everyone likely already has a charger or laptop power supply. This is where USB Type-C Power delivery comes into play.

USB Type-C Power Delivery is a standard[33] introduced in 2012 which allows a Downstream Facing Port (the official name for a USB Host port) to supply more voltage and/or current than a tradition USB port on request from the device. Some modern USB Type-C chargers allow to source up to 240 W provided that compatible cables are used, compared to the measly 4.5 W that a USB 3.0 Type A port can provide.

In order for a compliant Downstream Facing Port (DFP) to supply this greater power, proper signaling and negotiation must occur on the CC1 and CC2 pins of the USB Type-C connector. This can be done with a microcontroller or a dedicated IC. Since a microcontroller would also require an external switch, it was decided to go with a dedicated Upstream Facing Port (UFP) power delivery controller IC, specifically, the *TPS25730D* from Texas Instruments [31]. This chip an be configured simply by applying certain voltages to its configuration pins. The circuit for this section of the device is as shown in figure 46.

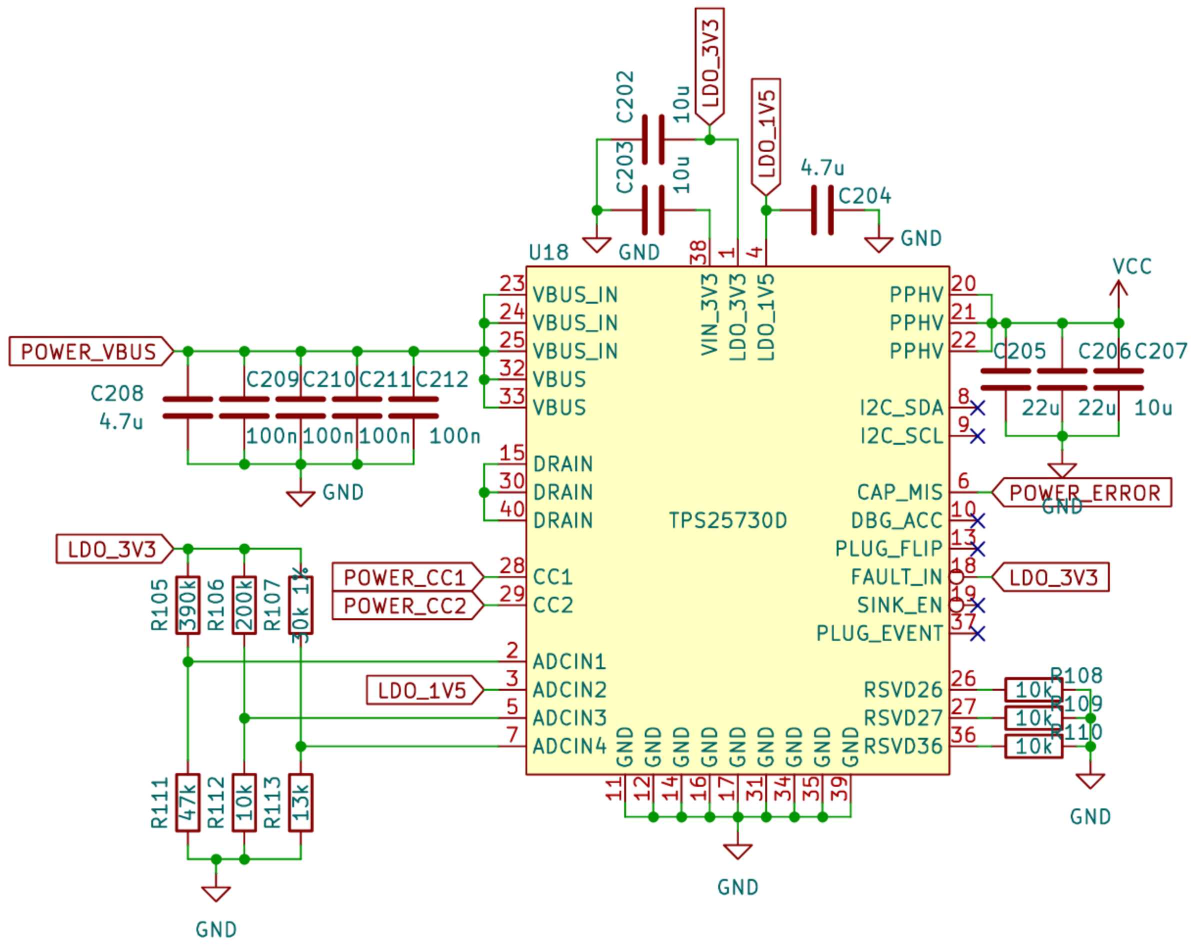


Figure 46: Schematic of the USB Type-C Power Delivery negotiation circuit.

As usual, the power going into the chip is generously decoupled with one 100 nF capacitor per power pin as well as an additional 4.7 µF capacitor as per the manufacturer's specifications. At the output, two 22 nF plus one 10 nF capacitors are used as per the manufacturer's specifications. The *TPS25730D* has built-in 3.3 V and 1.5 V Low Drop-Out regulators to both power its own circuitry and use in creating the control voltages to apply to its configuration pins. These are again decoupled as per the manufacturer's specifications.

The configuration pins are tied to resistive dividers in order to select a minimum 12 V at 1.5 A minimum which results in 18 W of available power. Selecting the value of these resistors must be done carefully in order to get the correct voltage without having to use unusual resistor values or risking the control voltage to stray out of the acceptable region for regular precision 5% resistors. For this, a small python script was written to uses a list of all 5% resistor decimal numbers, that is to say the portion of the value without the trailing zeros. This is because standard 5% resistor values have two digits follow by an arbitrary number of zeros, similar to how scientific number notation works. Knowing there are 24 different possible decimal values for 5% resistor, and assuming two orders of magnitude are checked, making the search space $2 \cdot 24 \times 2 \cdot 24 = 2304$. This is quite small so simple bruteforcing of every combination is good enough to compute the best possible match. The algorithm therefore tries every possible resistor divider using standard values and returns the one that ends up the closest to the desired voltage. Additionally, the script verified that no matter where in the 5% precision region the resistors are the resulting voltage is within the acceptable range for the desired configuration.

Finally, a few unused pins are tied to ground or 3.3 V depending on their exact purpose and the `CAP_MIS` pin is routed to an indicator LED. This pin indicates whether or not the USB Type-C Power Delivery negotiation was sucessful. In the case it isn't, the integrated switching mosfet is not activated and the `CAP_MIS` pin is toggled at a frequency of 1 Hz indicating to the user that the power supply is not compatible.

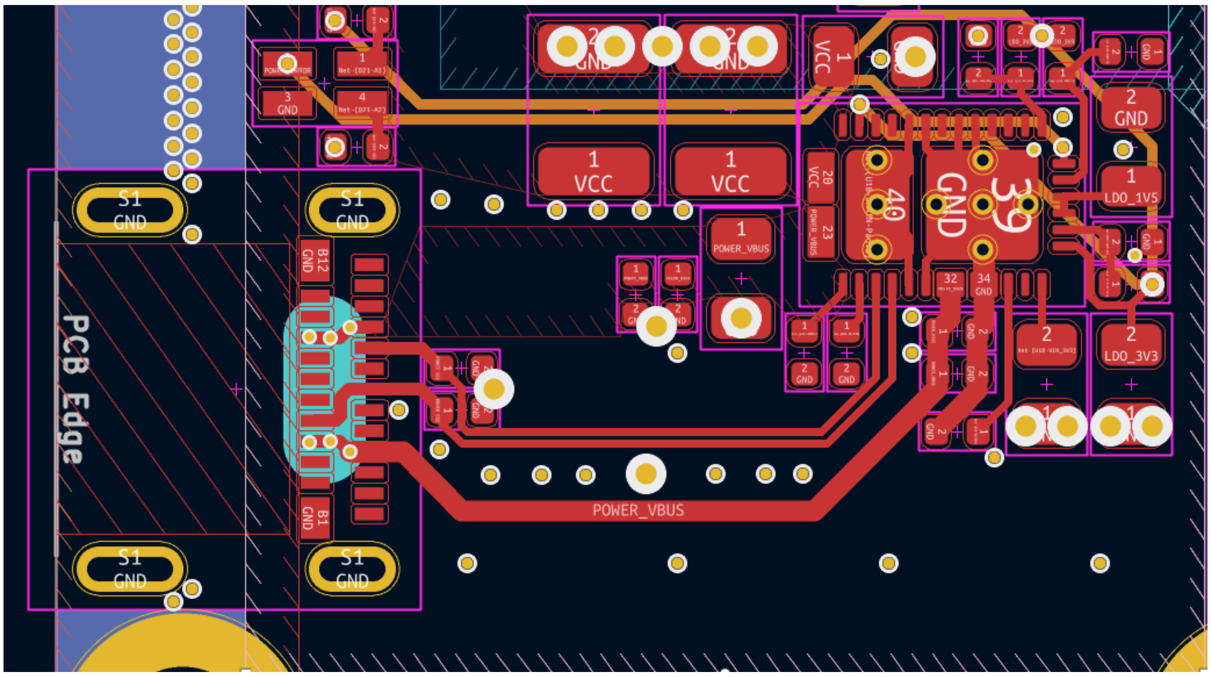


Figure 47: PCB layout of the USB Type-C Power Delivery negotiation circuit. Layer 1 is represented in red and layer 3 in orange.

As shown in figure 47, most signals are routed on the top layer with the exception of the traces going to the status LED. VCC was routed using copper polygons to ensure the lowest possible trace resistance. The decoupling capacitors are as usual located as close as possible to the IC's pins ensuring the lowest possible impedance. Since the IC contains an internal switching mosfet, thermal vias are added on both the ground and drain pads to ensure its able to sink heat into the PCB efficiently.

3.8 LED Indicators

In order to communicate basic status of the device to the user, mutliple bi-color LED indicator were added to the design. The purpose of each of the led is explained in later subsections, but since they all operate at the same voltage, the design of the driving circuit will be explained only once in this section.

The LEDs chosen for this design are the Kingbright APHBM2012. They were chosen due to low cost, but most importantly, their small size, allowing to waste as little space as possible on the PCB. The red side features a forward voltage of 1.75 V and the green side 1.9 V. Both sides accept a maximum forward current of 30 mA. However, since these LEDs will be viewed directly and not through a light pipe or enclosure, it is decided to limit the current to a low value of 5 mA. This value was chosen from experience to allow the LEDs to be visible without being obnoxiously bright indoors. Were this design be put in a case that uses light pipes, it may be necessary to raise the current to 10 mA.

For simplicity, and since the voltage being used to power the indicator LEDs is constant, a simple resistor is used to limit the current flowing through them as shown in figure 48.

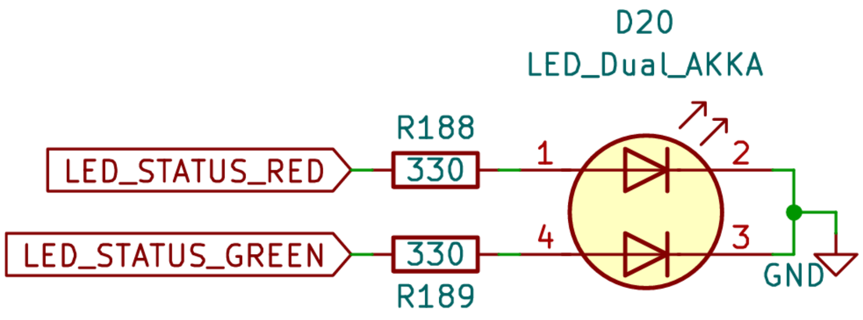


Figure 48: Circuit used for the indicator LEDs.

3.8.1 Frontend State

On each channel, an LED indicator was added in parallel with the SPDT switch that selects either the antenna or the reference. When the LED is green, this indicates that the frontend is on receive mode, while when the LED turns red, it indicate the frontend is in calibration mode. This is useful both for debugging purposes but also for the user to immediately known when the device is ready to receive a signal of interest.

3.8.2 Power Supply State

The USB-C Power Delivery IC provides a status signal to check whether or not the power delibery negotiation failed. It is therefore decided to connect this pin to the red side of the LED and connect the output of the 3.3v switching convert to the green side of the same LED. Therefore, when the device is powered on properly, the LED lights up green and if the power negotiation failed, it lights up red, indicating to the user that the issue must be resolved before the device will operate.

3.8.3 10MHz Reference Input State

The final LED is added next to the 10MHz reference input. It is controlled directly by the CPU inside of the FPGA depending on the option selected by the user and feedback from the clock generation IC. When the input is not select, the LED is off, when the input is selected, the LED lights up green, and if the clock generator IC reports an issue with the reference, it lights up red, again indicating to the user that an issue must be resolved.

4 Firmware

The firmware of the receive is split into two parts. First is the FPGA logic code written in Verilog which defines the internal circuitry of the FPGA. This is where all of the sample processing is done and where the 6502 soft CPU core is implemented. The FPGA design also contains various interfaces such as SPI, I2C and UART which are memory mapped into the 6502's address space. This is discussed in section 4.1.

The second part of the firmware is the code running on the 6502 soft core. This code is responsible for configuring hardware external to the FPGA such as the ADC, clock generator, tuners and synthesizer after which it is in charge of responding to commands from the host computer and do general house keeping tasks. This code is discussed in section 4.2.

4.1 Logic Design

The FPGA logic design was implemented using Verilog HDL and the Open Source Yosys[?] synthesis toolkit for the ECP5 series of FPGA. The choice to go with an open source toolkit rather than the manufacturer's official tools was motivated by better familiarity from previous projects and lesser licensing requirements were the design to become an actual product.

This does however come with some downsides. Since the manufacturer does not publish information on the bitstream format used by their FPGAs, support for the ECP5 in Yosys is entirely based on the reverse engineering efforts of the Trellis project[27]. As such, the bitstream generation may be less efficient than the official tools. Also, it was noticed that, in certain cases, the timing verifications were over-estimated the maximum operating frequency by a few percent, leading to unstable behavior.

Since the Yosys suite does not include a build system, standard Unix Makefiles were written in order automated the build process. Said build process takes place inside a custom Docker container which allows for simple installation of the environment.

4.1.1 Soft Core

Due to time constraints, and the debugging nightmare that could ensue from going with a custom solution, it was decided to use a pre-written CPU core. Many different CPU cores are available under permissive Open Source licenses on platforms such as OpenCores[21], so it is first necessary to select an architecture.

The requirements for the CPU are quite relaxed. Indeed, it does not need to perform any compute intensive task as any DSP will be taken care off directly in the FPGA. It is only responsible for basic housekeeping tasks and communication with the host PC. Another consequence of this low performance requirement is that it will not need much memory to accomplish its task, meaning that a small address bus size is permissible.

Since cost is one of the main concerns, reducing the size of the FPGA is paramount. Therefore, the CPU core has to be extremely simple to not use many logic units. Therefore, the MOS 6502[1] architecture is selected as the architecture for the soft core. The MOS 6502 features a 8 bit data bus and ALU and a 16 bit address bus, allowing for up to 64 kB of address space. The verilog module implementation selected for this project is from Arlet Ottens[7] and is available in the `fpga/src/soft6502/` directory. The actual implemention of the CPU core is outside of the scope of this prokect but its interface is as shown in table 6.

Name	Size	Direction	Description
`clk`	1	Input	Global clock for the CPU and bus.
`reset`	1	Input	Synchronous CPU reset signal.
`AB`	16	Output	Address bus.
`DI`	8	Input	Data bus from Devices to CPU.
`DO`	8	Output	Data bus from CPU to Devices.
`WE`	1	Output	Write enable signal.
`IRQ`	1	Input	Interrupt Request which triggers an interrupt request on rising edge.
`NMI`	1	Input	Non Maskable Interrupt input which triggers an NM interrupt on rising edge.
`RDY`	1	Input	Bus ready signal indicating the CPU is allowed to access the bus.

Table 6: Signals of the CPU verilog module

The entire system bus, that is the collections of the address bus, in and out data busses and write enable is connected directly to the memory mapper described later in this section. Both of the interrupt signals are currently unused in the design, but they could later be connected to the USB subsystem to more efficiently notify the core of data being ready for it handle. The ready signal is held at 1 as the CPU is the only bus master in the design. The reset signal is held high for 16 clock cycles at startup by trivial logic which will not be covered in this report for conciseness. Finally, the clock input is connected directly to the FPGA's input pin wired to the 25 MHz TCXO described earlier.

4.1.2 Memory Mapper

The memory mapper is responsible for connecting all the peripherals that the CPU needs to have access to via the CPU's memory bus. This includes not only SPI and I2C hardware, but also Random Access Memory (RAM). The memory mapper is implemented in `fpga/src/mem_map/mem_map.v` and its interface of is as shown in table 7.

Name	Size	Direction	Description
		CPU Interface	
`clk`	1	Input	Global clock for the CPU and bus.
`addr`	16	Input	CPU address bus.
`dataOut`	8	Output	Data output to the CPU.
		Peripheral Interfaces	
`memDataOut`	8	Input	Data coming from the RAM.
`memEn`	1	Output	Bus enable signal for the RAM.
`gpio0DataOut`	8	Input	Data coming from the GPIO0 module.
`gpio0En`	1	Output	Bus enable signal for the GPIO0 module.
		⋮	
`gpio3DataOut`	8	Input	Data coming from the GPIO3 module.
`gpio3En`	1	Output	Bus enable signal for the GPIO3 module.
`spi0DataOut`	8	Input	Data coming from the SPI0 module.
`spi0En`	1	Output	Bus enable signal for the SPI0 module.
`spi1DataOut`	8	Input	Data coming from the SPI1 module.
`spi1En`	1	Output	Bus enable signal for the SPI1 module.
`uartDataOut`	8	Input	Data coming from the UART module.
`uartEn`	1	Output	Bus enable signal for the UART module.
`i2c0DataOut`	8	Input	Data coming from the I2C0 module.
`i2c0En`	1	Output	Bus enable signal for the I2C0 module.
		⋮	
`i2c7DataOut`	8	Input	Data coming from the I2C7 module.
`i2c7En`	1	Output	Bus enable signal for the I2C7 module.
`fifoDataOut`	8	Input	Data coming from the USB FIFO module's CPU interface.
`fifoEn`	1	Output	Bus enable signal for the USB FIFO module's CPU interface.

Table 7: Signals of the memory mapper verilog module

The first notable property of the memory mapper is that it does not handle the CPU's data output bus, the write enable signal or even the address going to each device. Indeed, since only the CPU is the only module that needs to control these signals, there is no need to switch them depending on address. Instead, each device connected to the memory mapper takes an enable signal which is set when it should be paying attention to the bus. The CPU's data output and write enable signals are then routed as-is to every device. The address bus on is also routed as is with the exception that it is truncated to only as many least significant bits as the particular device requires.

The memory mappers only function is thus to control these enable lines and then to multiplex the data output of each device into the data input of the CPU depending on the address that the CPU is trying to access. For this, a certain memory range was allocated to each device manually. This memory mapping is as shown in table 49.

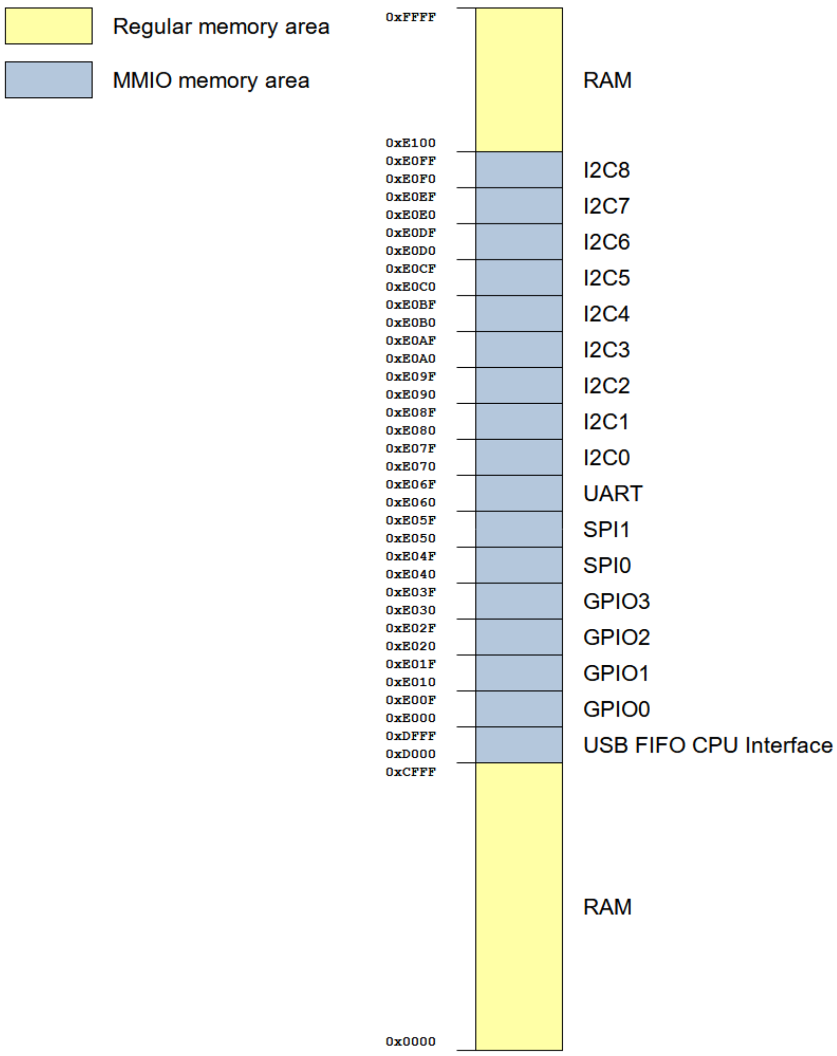


Figure 49: Diagram representing the address space as viewed by the CPU. Regular memory areas are colored yellow and Memory Mapped IO (MMIO) areas are colored blue.

This mapping was designed in such a way that only unsigned equalities on a certain number of most significant bit is sufficient to determine if a particular device is to be access. For example, the `fifoEn` line is high if `addr[15:11] == 0xD`, and the `gpio0En` line is high if `addr[15:4] == 0xE000`. If this was not the case, it would be necessary, for each enable line, to check whether the address is higher than one value and lower than another, which would naturally be a lot less efficient than a single equality comparison. This kind of mapping does have the disadvantage that unless the target peripheral happens to need a power of two amount of bytes, some memory will be wasted. Additionally, it is important to node that, while 16 bytes are given for most MMIO devices, this is only done to compare as few signals as possible, and most MMIO devices only need 4 or 8 bytes of address space. In that case, the registers of the device will simply wrap around multiple times within its allocated address space. The enable line for the memory is finally computed as the inverse of every other enable line ored together, ensuring that RAM is accessed only outside of MMIO devices.

One the enable lines are computed, the next step is to multiplex the data output of each memory mapped device. One difficulty in doing this is ensuring that there are no non-clocked logical loops. Indeed, the CPU module is written in such a way that changes in the data base during a read can cause non-clocked changes to the address bus. Using a purely combinational multiplexer in this case would mean that the selected device could change outside of clock edge, which would in turn cause the address to change, and so on. Therefore, each enable line is clocked into a synchronous version which is then used as the control signal for that data multiplexer. This ensures that no change in the address bus outside of a clock edge can create a change in data output to the CPU.

4.1.3 Double-Data-Rate to Single-Data-Rate Converter

Since the ADC outputs the samples in Double-Data-Rate (DDR) mode, it is first necessary to convert the signals it outputs to single data rate by deinterleaving the signals being sent on the rising and falling edges of the clock. This task is performed by the `DDRToSDR` module which is implemented in `fpga/src/adc/ddr_to_sdr.v`. The interface of that module is as described in table 8.

Name	Size	Direction	Description
Clock			
`dclk`	1	Input	Clock coming from the ADC.
Double Data Rate Interface			
`ddr_wclk`	1	Input	Word Clock / Overload signal coming from the ADC.
`ddr_q`	6	Input	Half width data bus from the ADC.
Single Data Rate Interface			
`sdr_wclk`	1	Output	Deinterleaved word clock signal.
`sdr_ovr`	1	Output	Deinterleaved Overload signal.
`sdr_q`	12	Output	Full width data bus.

Table 8: Signals of the DDR to SDR verilog module.

The double data rate interface is connected directly to the ADC and the single data rate interface goes off to the deinterleaver described later. The main difficulty in designing such a module is making sure timing requirements are met. Indeed the ADC clock runs at 200 MHz meaning that any downstream circuitry needs to be optimized to meet the setup and hold time requirements. The module is thus written to be as simple as possible, mearly transfering data between registers.

The module consists in two process, one that triggers on the rising edge of the clock, and the other that triggers on the falling edge of the same clock. The rising edge process simply saves the value of the half-width data bus and the `ddr_wclk` signal to registers. The falling edge process saves the two previously mentioned signal directly to half of the output registers, and copies the value from the temporary falling edge register to the remaining half of the output registers. This is done to ensure that all signals have the same maximum setup time, and does mean that circuitry using the output of this module must trigger off the falling edge of the clock.

4.1.4 Deinterleaver

Even after the data from the ADC has been put into a single data rate format, the clock running at 200 MHz is still problematic. Therefore, in order to reduce the clock frequency at which further logic has to run, the samples are deinterleaved and a new, lower frequency clock is generated. This task is accomplished by the `ADCDeinterleave` module implemented in `fpga/src/adc/adc_deinterleave.v`, the interface of which is described in table 9.

Name	Size	Direction	Description
Interleaved Interface			
`adc_dclk`	1	Input	Clock coming from the ADC.
`adc_wclk`	1	Input	Word Clock signal coming from the ADC.
`adcr_q`	12	Input	Full width data bus from the ADC.
Deinterleaved Interface			
`clk`	1	Output	One-Eighth rate clock output.
`chan0`	12	Output	Sample value for channel 0.
`chan1`	12	Output	Sample value for channel 1.
`chan2`	12	Output	Sample value for channel 2.
`chan3`	12	Output	Sample value for channel 3.
`chan4`	12	Output	Sample value for channel 4.
`chan5`	12	Output	Sample value for channel 5.
`chan6`	12	Output	Sample value for channel 6.
`chan7`	12	Output	Sample value for channel 7.

Table 9: Signals of the DDR to SDR verilog module.

To understand how the deinterleaver works, it is first important to understand how the ADC's `wclk` signal is defined. Firstly, as mentioned in the section about the DDR to SDR module, the `wclk` signal, despite its name, encodes not only the Word Clock but also the Overload signal, with one being transmitted on the rising edge of the clock, and the other on the falling edge. Therefore, for the remainder of this section, the `wclk` signal is only the Word Clock part. This signal is defined by the manufacturer as being high while the channel 0 sample is being sent.

Knowing this, the process of the deinterleaver module can be described as follows. The module's only states are a 3 bit counter indicating which channel is currently being receiver, as well as 8 staging registers. On every rising edge of the clock, the module writes the sample received from the ADC to the staging register corresponding to the current channel counter index. The way this counter is updated depends on the stage of the `wclk` signal. If it is high, the channel counter is set to 1 as that would be what the next channel is. Otherwise, the channel counter is simply incremented by 1. When the channel counter is equal to 0, all of the staging registers are copied to the output registers and the output clock register is set to 0, creating the falling edge. Finally, when the channel counter is equal to 4, the output clock register is set to 1, creating the rising edge. This, in summary, has the effect of having the channel sample values be updated on the falling edge of the generated clock, so that they are valid and stable during the rising edge of said clock.

4.1.5 Sample Frame Packer

The sample frame packer module is responsible for taking the stream of 12bit samples and packing them into discrete 1024 byte long packets as well as prepending a frame counter. The module contains a built-in double buffer which stores the packet in words of 32bit for easier access by the USB FIFO controller. This counter increments by one every time the module generates a packet and is used to detect and quantify packet drops on the PC side. The interface of this module is as described in table 10.

Name	Size	Direction	Description
Sample Interface			
`samp_clk`	1	Input	Sample clock from the DSP.
`samp_valid`	1	Input	Indicates whether the provided sample is to be read.
`samp_value`	1	Input	Value of the sample to be read.
Buffer Interface			
`buffer_clk`	1	Input	Clock used for reading out the latest buffer.
`buffer_addr`	10	Input	Address into the buffer.
`buffer_flush`	1	Input	Help high for once cycle to flush the currently read buffer
`buffer_avl`	1	Output	Indicates that a buffer is available for reading.
`buffer_out`	32	Output	Data output of the latest buffer.

Table 10: Signals of the USB FIFO controller module.

The sample frame packer is actually composed of three subprocess as shown in figure 50, two of which run on the sample clock, and one runs on the buffer clock. This is done in order to increase the maximum usable sample clock. Indeed, pipelining the circuitry in this way means that less logic is required at each stages and thus results in a higher maximum operating frequency. The first process takes the samples packs them into 32 bit words. Additionally, it injects the frame counter as the first of every 256 words. The second process takes these words and writes them to one of two buffers depending on the state of their associated flag which indicated whether they are empty or not. This is done to allow for latency without having to drop samples. One can think of this system as a two element FIFO where each element is an entire buffer. The reason for not using a regular FIFO is that packets cannot be partially read or written. They must always be transfered as one complete entity or there is a risk of serious performance degradation resulting from the USB chip having to split the sample packets between USB packets. The third and final process is the one on the buffer clock side which selects the buffer that has data ready, handles flush requests from the downstream logic and indicates to downstream logic whether or not a buffer is available.

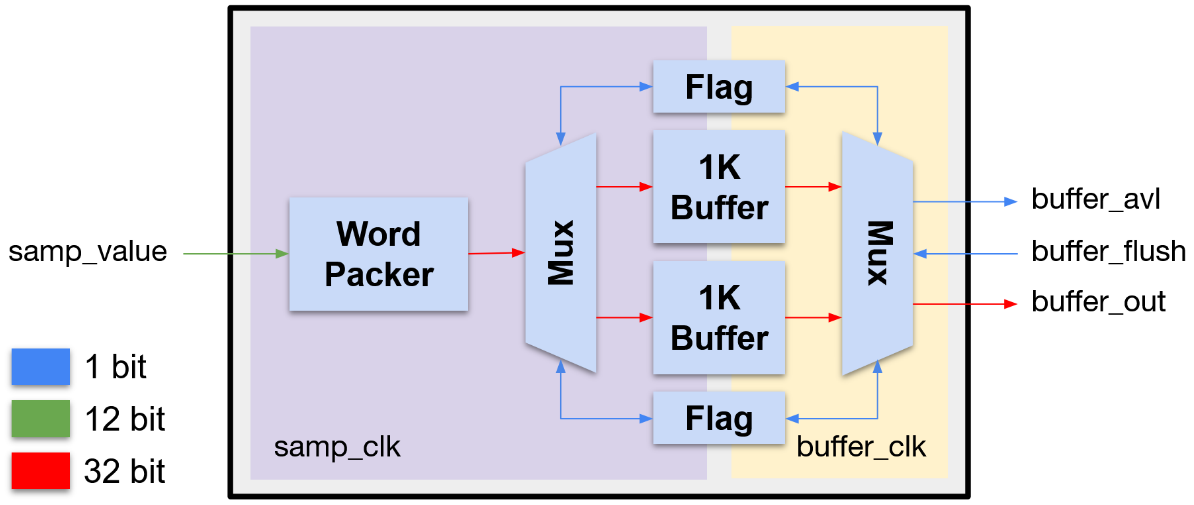


Figure 50: Simplified block diagram of the sample frame packer showing the different clock domains and bus sizes. Some signals are omitted for clarity.

The sample packer is based on a two states. One counting the number of nibbles, that is words of 4 bits, have been output to the buffer, and another used as temporary storage for partial words. Since the size of samples is 12 bits, this nibble counter is incremented by 3 on every sample. The value of this counter is used to determine which portion of the temporary word have to be updated with the latest sample, as well as if and when to output that word for the rest of the pipeline to handle.

Next, the words head to the second process which fills up the buffers depending on their state. To do this, it uses a word counter which gets incremented every time a word is written. When the counter gets to the end, the process checks whether or not the buffer not currently being written to is empty, in which case it switches to that buffer for the next words. If the other buffer is not empty, the word counter is reset and the buffer is dropped. The state of each buffer is indicated by its associated flag module. These flags have a set signal in one clock domain and a reset signal in the other and allow the status of the buffer to be read on both clock domains.

The buffers, in this case, are dedicated dual-port FPGA block RAM. They are instantiated by inference using Yosys's defined dual port memory pattern inside of the `SDPRAM` module located in `fpga/src/dsp/sdpram.v`. This dual port memory has a write port in one clock domain and a read port in the other, allowing to pass entire buffers across clock domains.

The process on the sample clock side uses the set signal on each buffer it finishes writing to indicate that the buffer is now full. If the buffer has to be dropped because the other buffer was not empty, the full flag is however not set. This has the affect that only one of the two buffers has the full flag set, allowing to use one of them to control the multiplexer on the . Finally, the buffer clock side process simply waits for a flush command and routes it to the currently read buffer.

4.1.6 USB FIFO Controller

The USB FIFO controller module is, as its name implies, responsible for controlling the *FT601* USB FIFO IC. It continuously checks whether samples are available to send, or data from the PC is available to receivers, arbitrates the direction of the FIFO and moves data to its intended destination. This controller's implementation is split between multiple files located in `fpga/src/usb_fifo`. The top level module file `usb_fifo.v` connects all of its submodules together, `rxram.v`, `txram.v` and `tdpram.v` contain the various memories and buffers used to store data and act as registers, `mmio.v` describes the interface to the CPU and `usb_fifo_sm.v` contains the main state-machine arbitrating the entire operation. The interface of the USB FIFO controller is as shown in table 11

Name	Size	Direction	Description
FIFO Interface			
`fifo_clk`	1	Input	Clock for the FIFO bus coming from the FT601.
`fifo_oe`	1	Output	FT601's Output Enable signal indicating that it may output to the bus.
`fifo_we`	1	Output	FT601's Write Enable signal indicating that the FPGA wants to write.
`fifo_re`	1	Output	FT601's Read Enable signal indicating that the FPGA wants to read.
`fifo_rxf`	1	Input	Signal from the FT601 indicating that data is available to receive.
`fifo_txe`	1	Input	Signal from the FT601 indicating that space is available to send.
`fifo_be`	4	Bidir.	Byte enable signal going to and from the FIFO.
`fifo_data`	32	Bidir.	Data bus going to and from the FIFO.
CPU Interface			
`cpu_clk`	1	Input	Global clock for the CPU and its data bus.
`cpu_addr`	12	Input	12 least significant bits of the CPU's address bus.
`cpu_data_in`	8	Input	Data bus going from the CPU to the controller.
`cpu_data_out`	8	Output	Data bus going from the controller to the CPU.
`cpu_rw`	1	Input	Signal indicating whether the CPU intends to read or write.
`cpu_en`	1	Input	Signal indicating to the controller that it is selected for access.
Sample Interface			
`samp_clk`	1	Input	Sample clock from the DSP.
`samp_valid`	1	Input	Indicates whether the provided sample is to be read.
`samp_value`	1	Input	Value of the sample to be read.

Table 11: Signals of the USB FIFO controller module.

The FIFO interface is, as its name suggests, connected through the FPGA's IO pins to the FT601 USB FIFO IC. Currently, the `fifo_be` outgoing signal is hard-coded to be `0xF` indicating that all 4 bytes of the 32bit words being exchanged are valid, which means that all buffer operations must be on a multiple of 4 bytes. This was done to simplify the controller state machine. The CPU interface signals are connected to the previously described memory mapper. Finally, the sample interface is connected directly to the ADC. In the future this would be connected to the output of the DSP. A block diagram of the controller is shown in figure 51.

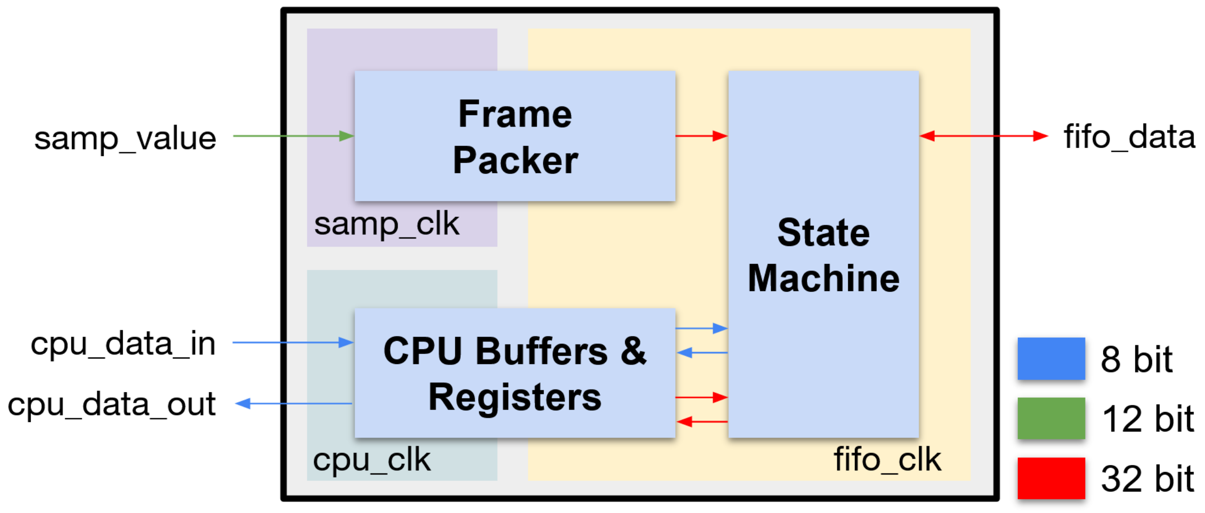


Figure 51: Block diagram of the USB FIFO controller module showing the clock domains and flow of data. Some signals are omitted for clarity.

As can be seen in the block diagram, the controller operates over three different clock domains. The first is the sample clock domain in which samples sent to the controller first pass through the previously described Sample Frame Packer block in order to be packed into discrete 1024 byte packets. The controller's state machine than read back the packets as 32bit words within the FIFO clock domain. The final clock domain is the CPU's clock domain which interfaces with the controller state machine over a series of dual port RAM blocks acting as registers and buffers for received command packets.

One important thing to note is, while the design implements four 8bit registers for the state machine and CPU to communicate commands and status with, only one of the register is currently in use and acts a command buffer empty/full flag. The other three registers are reserved for future use.

Before being used by the state machine, the incoming signal from the FIFO are synchronized on `fifo_clk`'s rising edge. This is done because the FIFO changes its output signals, and requires the controller to change its output signals, on the falling edge of the clock. It is therefore necessary to sample those incoming logic signals on the falling edge so that the setup and hold times required by the FPGA's internal logic are respected. On the sample interface, some preprocessing is also required. Indeed, these 12bit samples first need to be packed into discrete 1024 byte long packets for transfer over USB 3.0. This is achieved by the Sample Frame Packer module described earlier. The main state machine for the controller is described in figure 52.

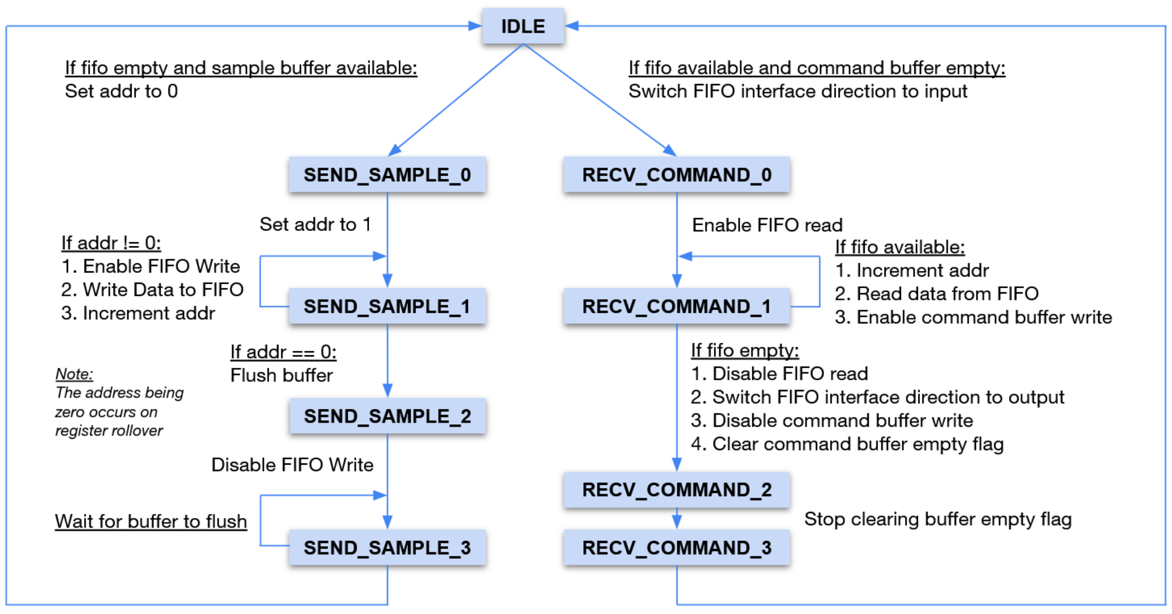


Figure 52: Finite State Machine diagram of the USB FIFO controller. The path from the idle state to itself when none of the start conditions are fufilled is implied and omitted from the diagram for clarity.

The default state of the controller is, of course, the idle state. In this state, the controller continuously checks whether it is able to send sample packets, or whether it is able to receive command packets. If neither of these conditions are met, it remains waiting in the idle state.

If a sample packet is available to be sent, and the USB FIFO has space for it, the controller starts by setting the sample packer buffer address register to 0. This has to be done before anything else because the FPGA's block RAM has a one clock cycle latency between setting the address and getting corresponding data. It is important to note that the address register is 8 bits wide, so will rollover after sending 256 words, or 1024 byte, the size of a sample packet. The controller now transitions to the `SEND_SAMPLES_0` state. This state is only used as the delay to wait for the data out of the ram. In this state, it sets the address to 1 and switches to the `SEND_SAMPLES_1` state. This is the main sample sending state. In this state, if the address has rolled over, indicating that all 256 words have been sent, the buffer is flushed and the controller moves to the `SEND_SAMPLES_2` state. Otherwise, the controller enables the write enable line to the FIFO, outputs a word of data from the packet, increments the address, and stays on the current state. On the `SEND_SAMPLES_2` state however, the controller disables the FIFO's write enable line and goes to the final `SEND_SAMPLES_3` state. This state is only used to wait for the buffer to actually flush. As described in the section describing the buffer flag module, this take a few clock cycles to occur, after which the controller switches back to the idle state.

If the USB FIFO has data available and the command packet buffer is empty, the controller first switches the FIFO's interface signaling direction to input, after which it transitions to the `RECV_COMMAND_0` state. In that state, the controller enables the FIFO's read enable line and switches to the `RECV_COMMAND_1` state. When in that state, while the FIFO has data available, it increments the command packet buffer address, writes the data word from the FIFO into the command buffer, enables the command buffer's write enable signal, and remains in the `RECV_COMMAND_1` state. If however, the USB FIFO no longer has any data available, the controller disables the FIFO's read enable line, switches the FIFO interface signal direction to output, disable the command packet buffer's write enable line, clears the command command buffer empty flag and switches to the `RECV_COMMAND_2` state. This state only serves as a delay to then disable the signal clearly the buffer empty flag. The final `RECV_COMMAND_3` state was used during development but is no longer in use, so it switches immediately back to the idle state.

4.2 CPU Code

This section describes the behavior and some implementation details of the code running on the soft core within the FPGA. The description of this code is broken down by phase of execution from start up to responding to commands from the host computer.

4.2.1 Initialization

As soon as the soft core exists reset, it executes the reset handler which brings it to the `main()` function. There, the first task accomplished by the firmware is to initialize the UART block such that debugging logging is able to be displayed on an external terminal. This is done by calling the `uart_init()` function which loads the UART's config register with the stop bit and word length information and calls the `uart_set_baudrate()` which computers the clock divider value in order to achieve the desired baud rate.

Following this, the code waits for exactly 1 ms. This is done because the UART interface on the PC side requires a bit of time to recognise that the transmit line has gone high and is getting ready to send characters. Ommitting this leads to the first transmitted character being corrupted. The code then prints firmware version information to the UART.

The next step in the initialize process is to initialize all four GPIO blocks. This is done by calling `gpio_init()` which in turn writes the direction and state masks to the GPIO module's registers. The initial configuration for the GPIOs is such that all outputs going to the RF switches are off and that all chip select lines to external ICs are de-asserted.

Next comes the time to initialize the two SPI modules. This follows a very similar process to the uart where the `init_spi()` function loads the SPI module's configuration registers with the clock polarity and phase information, then computes the required divider for the requested baud rate and writes it to the divider register of the SPI module being initialized. The process is exactly the same for the I2C hardware so it will not be repeated.

Now that all communication interfaces are up and running, hardware external to the FPGA can start being initialized. The first hardware to initialize is naturally the clock generator, without which no other external IC can function. This is done by calling the `lmx2572lp_init()` function which simply loads all of the chip's registers using the values generated by Skywork's Clock Builder Pro software.

Following this, the synthesizer is initialized by calling `lmx2572lp_init` which issues a soft reset command and configures the synthesizer to enable register read back for status validation. This is then immediately followed by a 10 ms sleep in order for the ADC to finish its start up procedure which , as per its specification, takes 2^{20} clock cycles to complete, which when running from the 200 MHz takes just over 5.2 ms. The sleep time being almost double the required amount is simply to ensure 100% that the startup of the ADC is complete.

Once the ADC is up, it is possible to start configuring it. First, the firmware writes to the channel enable register in order to select all 8 channels for sampling. Next, the integrated Delay Locked Loop is configured and enabled to add a phase shift of 45° to the clock. This is done in order to ensure setup and hold times for the FPGA are respected. Finally, the ADC is commanded to go into standby in order to reduce power draw when not actively receiving. The last thing the initialization section of the code does is print to the uart that it is ready and flush the USB command buffer.

4.2.2 Command Loop

Following initialization, the soft core goes into a loop calling the `process_commands()` function indefinitely which, as its name suggests, takes care of processing commands coming over USB from the host PC. The first thing the function does is loop until the `usb_available()` function returns true, indicating that the USB receive buffer contains a command packet.

From there, the packet is decoded to extract its length and the command ID. The code than executes different procedures depending on the command ID. The currently implemented commands are as described below.

1. `SET_RF_SWITCHES`: This command takes exactly two argument bytes which describe the state of all control signals going to the RF switches. Uppon receipt of the command, the two bytes are extracted and the GPIO states are set accordingly.
2. `MCP37211_WRITE_REG`: This command takes 3 argument bytes which encode the register address and register value to write to the ADC. The first and second bytes contain the register address in little-endian

format and the third contains the register value. The command handler extracts these values then calls `mcp37211_write_reg()`.

3. `MCP37211_READ_REG`: This command takes 2 argument butes which only encode the register address to read from the ADC. Since a way to send command responses back the computer has not yet been implemented, it simply calls `mcp37211_read_reg()` and prints the result to the debug UART.
4. `LMX2572LP_WRITE_REG`: This command works on the same principle as `MCP37211_WRITE_REG` except this time the register address is a single byte long and the register value is two bytes storing a 16bit value in little-endian format. When executed, the command calls `lmx2572lp_write_reg()` with the arguments decoded from the command packet.
5. `SET_LNA_GAIN`: This command takes a single argument byte and calls `r820t_set_lna_gain()` in order to set the LNA gain of the tuners.
6. `SET_VGA_GAIN`: This command takes a single argument byte and calls `r820t_set_vga_gain()` in order to set the VGA gain of the tuners.
7. `SET_MIXER_GAIN`: This command takes a single argument byte and calls `r820t_set_mixer_gain()` in order to set the mixer gain of the tuners.
8. `SET_FREQUENCY`: This command sets the frequency that the tuners are tuned to and, in a finished version, would also trigger recalibration of the phase offsets of each channel.
9. `TUNER_WRITE_REG`: As its name implies, this command works similarly to the other register writing commands except this one is for writing to the tuners.
10. `TUNER_READ_REG`: Again this command works similarly to the others and allows to read a certain number of registers from the tuners. However, again since no system to send responses back to the host has been implemented, the result is simplyy displaued in the terminal.
11. `SEL_CHANNEL`: This final command allows to select which ADC channel's samples are sent back to the host. Currently, only single channel reception has been implemented so this command allows to select which one gets sent back.

Unfortunately, the tuner control commands outside of the commands to read and write registers are not currently functional. This is because the driver for the tuners currently does not work properly when compiled for the 6502 CPU. Since time contraints do not allow to write an entirely new driver from scratch or debug the existing one, the driver was moved to the host PC and interacts with the tuner through the `CMD_TUNER_WRITE_REG` command.

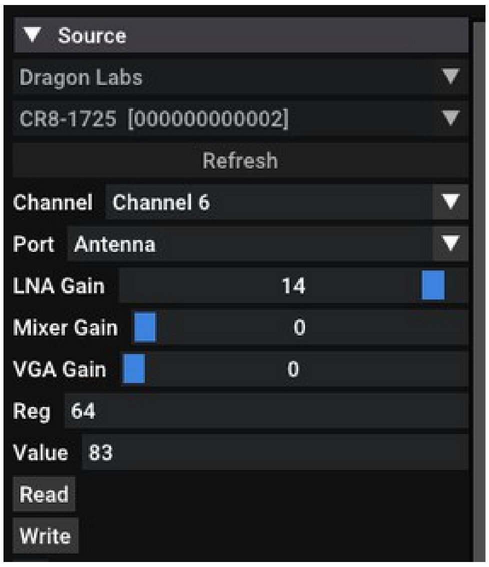
Finally, after the command has executed and the command packet data is no longer required, the code calls the `usb_flush()` function which releases the command packet buffer so a new command packet can be received. This cycle repeats indefinitely until the unit loses power.

5 Host Software

Control and visualization of the received signals is implemented via a source module for the SDR++[4] software directly. In the future, having a separate device would be desirable, but direct development of a source module is ideal for fast prototyping and testing.

Device enumeration and sample streaming is implemented using the libusb cross-platform USB library. The streaming code runs in a separate thread from which it queue buffers for the PC's USB hardware to fill with packets coming from the receiver. When a packet is receiver, the frame counter is extracted and used to keep a running count of lost sample packets. Next the samples are unpacked from 12bit signed integer to 32bit floating point. These samples are then passed through a double buffer to the processing thread.

The processing thread runs direct digital down conversion as already implemented inside SDR++ to convert from a Low IF signal with the center at 4.57 MHz to a Zero-IF IQ digital signal which the rest of the software expects. The source menu for controlling the hardware is as shown in figure 53



Dragon Labs

CR8-1725 [000000000002]

Channel Channel 6

Port Antenna

LNA Gain 14

Mixer Gain 0

VGA Gain 0

Reg 64

Value 83

Figure 53: Screen capture of the menu which the receiver's source module exposes to the user.

The top drop down widget shows the selected source module with the second drop down widget showing the selected device which was automatically detected during enumeration. For debugging purposes, the source module allows the user not only to select any channel from the ADC, switch between antenna and synthesizer inputs, adjust the tuner's gain but also to write to then ADC's registers directly.

6 Testing

This section of the report details the testing methodology and results for both the subsystem prototypes and the finished receiver.

6.1 Clock Generator

Before any other IC, with the exception of the USB FIFO, can be used, it it necessary to configure the *Si5351c* clock generator to output the required clocks. After this, the clocks can be checked to ensure that they are clea, at the correct frequency and of correct amplitude.

6.1.1 Methodology

Testing the clock simply involves programming the *Si5351c* with the configuration generated by Skywork's Clock Builder Pro software and checking each of the clock outputs with an oscilloscope.

6.1.2 Issues

After configuring the clock generator, it was immediately apparent that it was not behaving properly. Often, none, or only some of the clock outputs would start up, and when they did start up, they would be missing cycles or be extremely noisy and unstable. This unstable behavior is shown in figure 54.

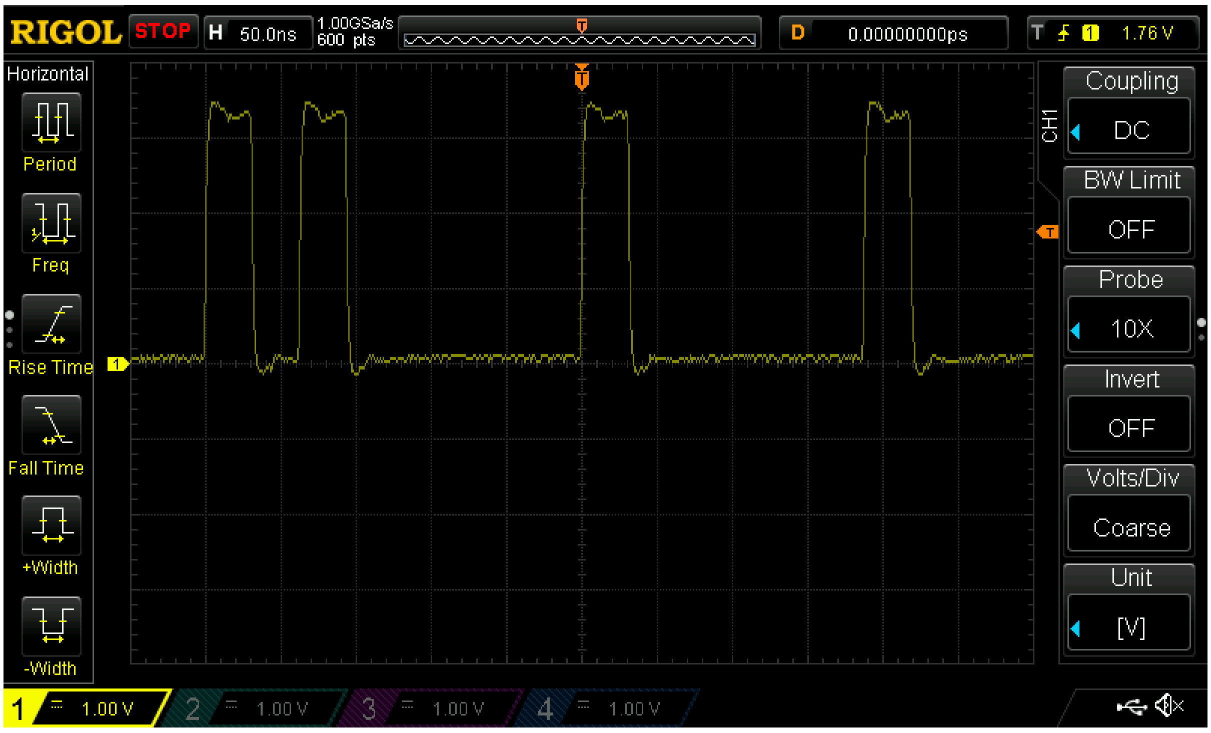


Figure 54: Screen capture of an oscilloscope showing the erratic behavior of the tuner, specifically only certain clock pulses being output.

While experimenting with the receiver board, it was noticed that the misbehavior of the part was randomized, not on reset of the part, but on power cycle of the entire board. This pointed towards a power related issue. As such, the voltages coming into the chip were measured and were all clean and within specification. However, when examining transient behavior during power on of the board, a problem was identified. Figure 55 shows the voltage of both the 2.5 V and 3.3 V rails going into the clock generator chip.

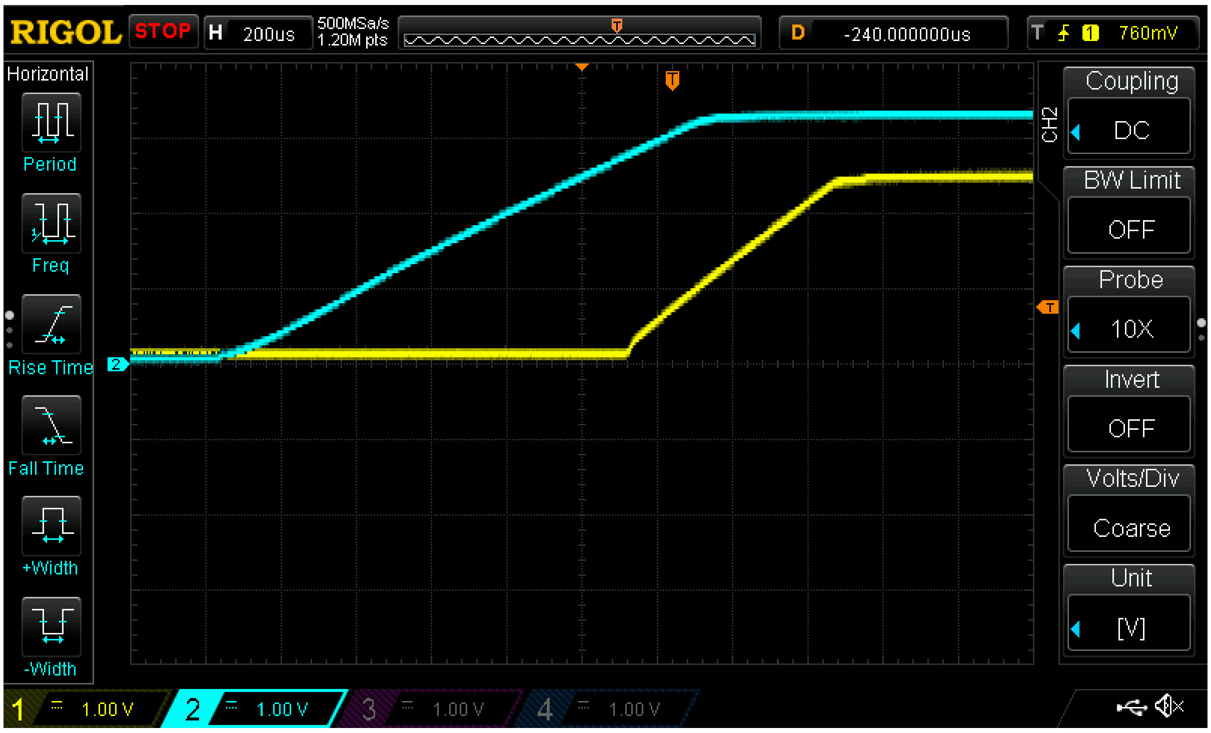


Figure 55: Screen capture of an oscilloscope showing the 2.5 V rail in yellow and 3.3 V rail in blue going into the clock generator chip right as the board is powered up.

This measurement reveals the fact that, unlike most linear voltage regulators, the ones used for the 2.5 V rail possess internal circuitry to delay start up until their input has reached a certain voltage. This is extremely problematic, because the *Si5351c*'s datasheet specifies that the IO voltages rails must come up either at the same time or before the core supply rail. Due to the 2.5 V linear regulator delaying its start up, the 2.5 V rail would come up very slightly after the core power rail and thus violated the requirements of the chip, leading to undefined behavior.

There two ways of fixing this issue. The first would be to swap the linear voltage regulator to one that does not exhibit this behavior. This is not a good option as this behavior is often not even documented in datasheets, and thus cannot be relied upon. The second solution would be to simply switch from using 2.5 V to 3.3 V for the IO voltage rail of the clock generator. This would ensure that, by definition, the IO and core rails come up at the same time, but would mean that an attenuator is required in the clock output going to the ADC.

After consideration, it was decided to go with the second option. In order to replace the 2.5 V rail with the 3.3 V rail, the FB14 ferrite bead was removed and a wire was run between the output of ferrite bead FB13. Next, the DC blocking capacitors C134 and C135 were temporarily removed and in their place was added a Pi attenuator. This was done by first soldering the input shunt resistor across the input pads of the original DC blocking capacitor then soldering two resistors vertically on top of the input shunt resistor. Next, the original capactiors were soldered back vertically on what was originally their output pads. Finally, the output shunt resistor was then soldered on top of the two vertical capacitors and thin wire was added to link the vertical resistor vertical capacitor. This dubious bodge is shown in figure 56.

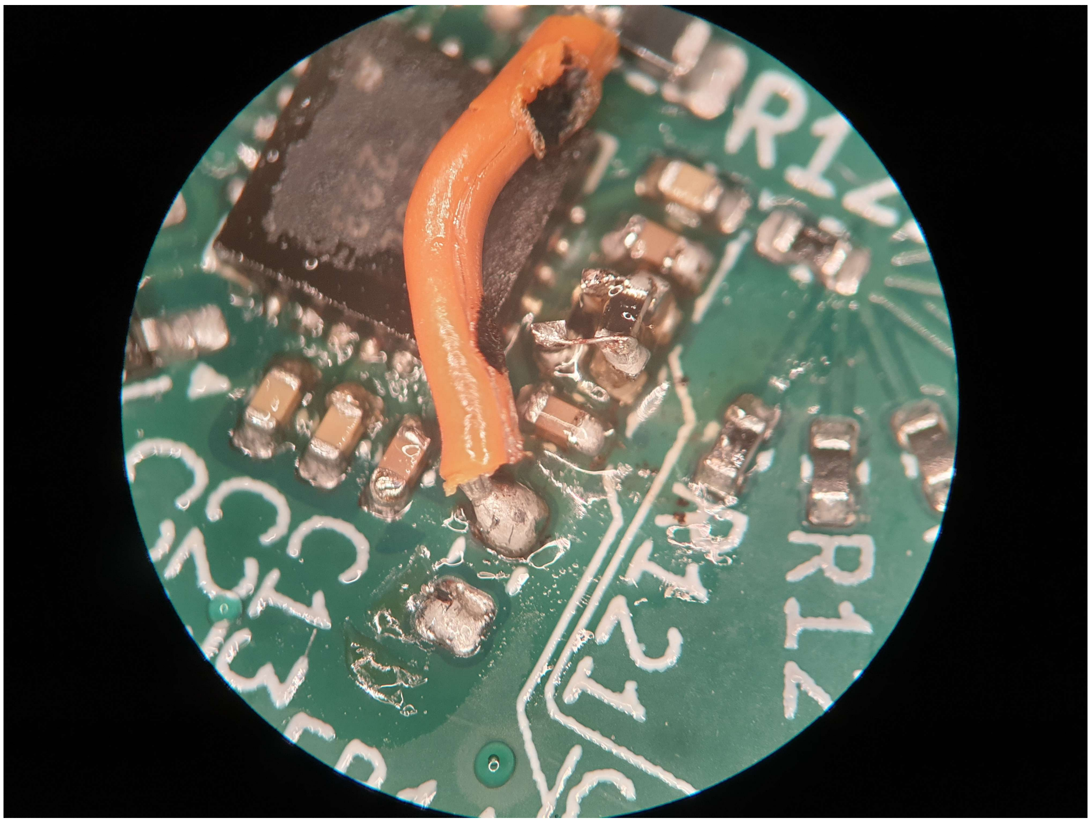


Figure 56: Microscope picture of the rework done to the clock generator section of the PCB.

6.2 Phase Reference Splitter

Due to how critical the reference splitter is to the proper function of the receiver, a prototype was built to verify its performance prior to ordering the final PCBs.

6.2.1 Prototype

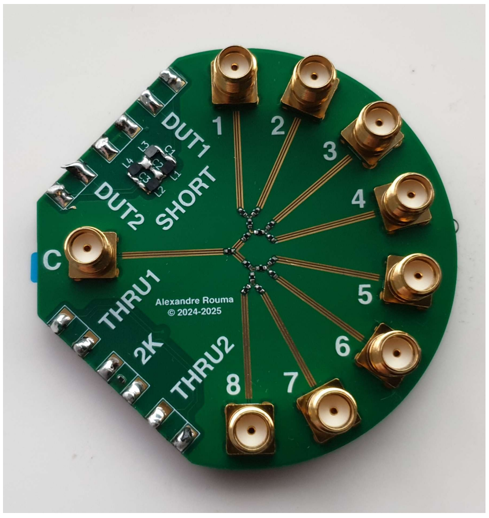


Figure 57: Picture of the phase reference splitter and anti-aliasing filter prototype PCB. The anti-aliasing filter is visible between the DUT1 and DUT2 markings. The phase reference splitter is placed right in the center of the PCB where its connections to the SMAs are visible as golden traces. A VNA calibration kit is also included in the from of the SHORT, 2K and THRU terminals.

As can be seen in figure 57, the outputs of the splitter circuit are routed to SMA connectors via equal length microstrip lines. This ensures that the connects to the splitter will not create different phase shifts between the outputs. This prototype was built using the exact same PCB stackup and layout as the one on the final PCB in order to ensure that the results are accurate.

6.2.2 Methodology

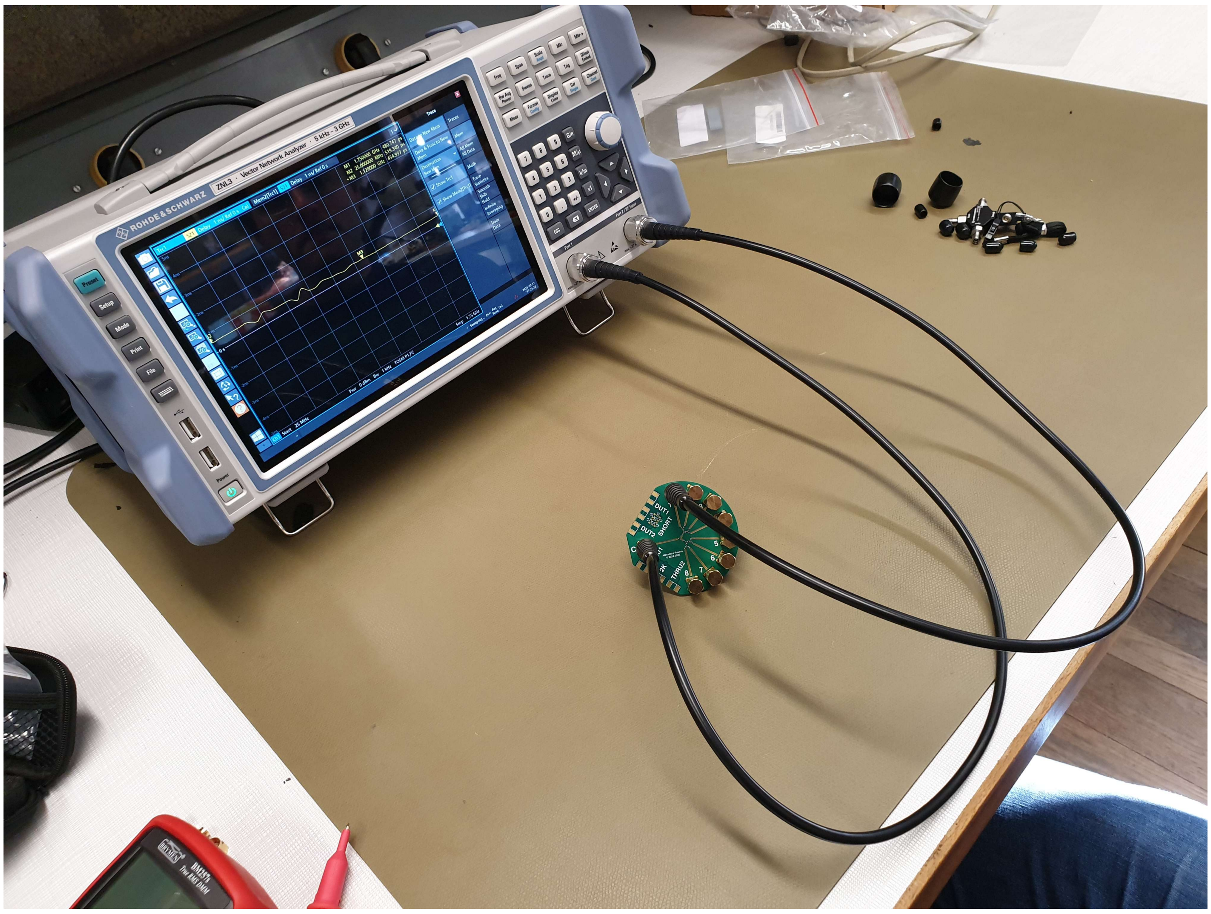


Figure 58: Picture of the test setup used to validate the performance of the phase reference splitter..

The experimental set up is as shown in figure 58. A Vector Network Analyzer (VNA) is used to measure the phase shift and insertion loss between the input and each output. This is done by connecting port 1 of the VNA to the input of the splitter, connecting port 2 to the output under test and connecting 50 Ω terminations on all output. Terminations are required because reflections from the paths not being measured would change the response of the splitter. The VNA is configured to measure oven the entire intended operating frequency range of the receiver, that is 25 MHz to 1.75 GHz.

6.2.3 Results

The VNA measurement results are as shown in figure 59. Starting with the insertion loss, it is as expected rather high. This is normal of resistive splitter and, as mentioned in the hardware design section, is desired for the design as the synthesizer's output must be attenuated before reaching the extremely sensitive input of the tuners. In terms of phase offset, it is quite linear with slight deviations at around 350 MHz and 650 MHz. Since this effect is present equally on all 8 channels, and thus has no impact on the use case of the splitter, its exact cause will not be investigated.

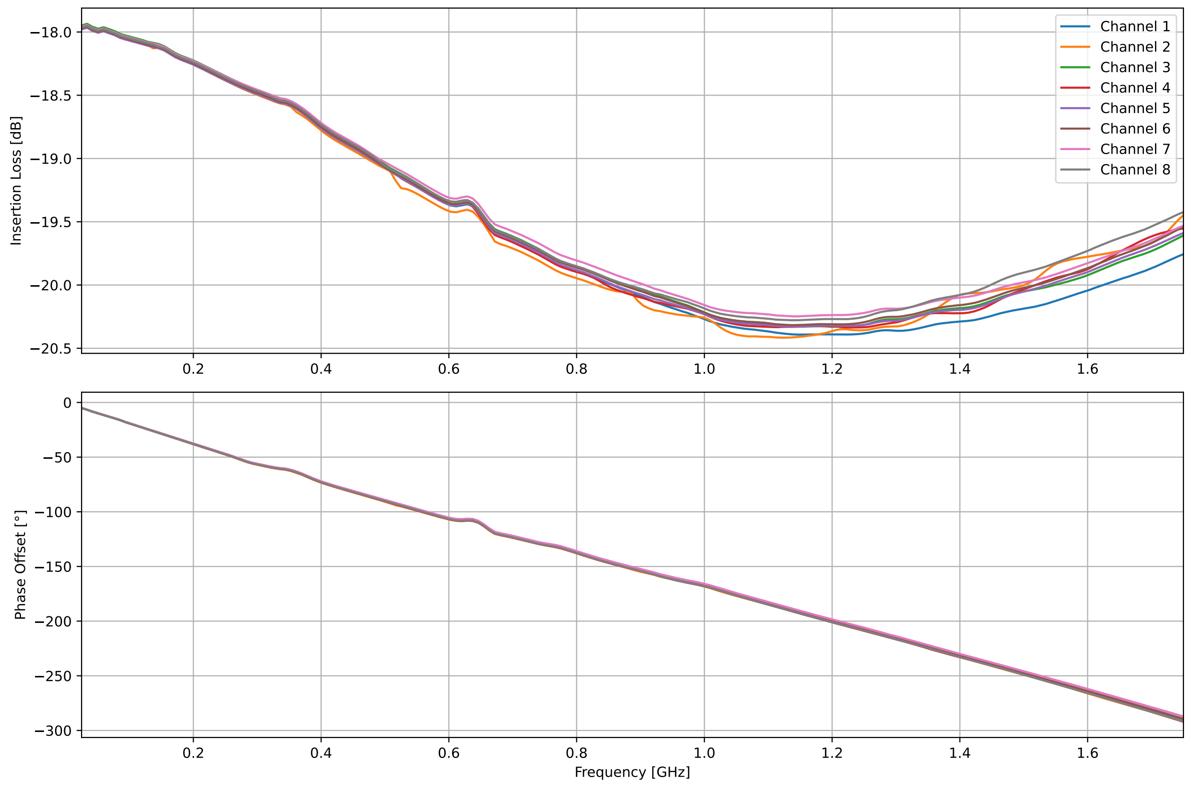


Figure 59: Bode plot showing the response from the input to each output of the phase reference splitter versus frequency.

It is then possible to compute the difference in phase offset between each channel and a reference channel. In this case channel 1 is used as the reference against which channels 2 through 8 are compared. The result of this comparison is as shown in figure 60. These results reveal a phase offset difference that naturally grows with frequency. Although the difference of some channels appear to move in groups, this appears to be a coincidence as they are not geometrically correlated. The worse of the channels ends up with a phase offset difference of a little over 3° compared to channel 1. Since the target was less than 1% phase error over the entire frequency range, this test can be considered successful.

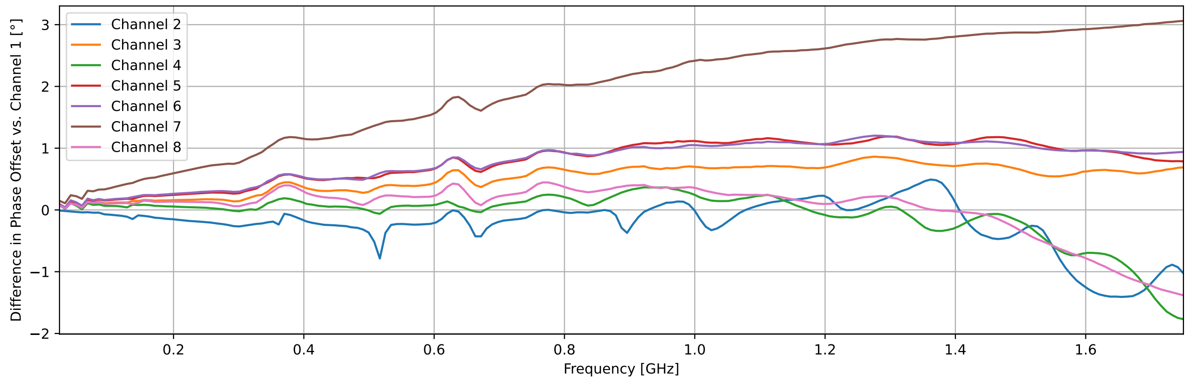


Figure 60: Difference in phase offset between channel 1 and channels 2 through 8 versus frequency.

6.3 Anti-Aliasing Filter

Due to the importance of this filter for the image rejection performance of the receiver, a prototype was built in order to validate the simulation results. This section details the custom tools designed to test the filter, the test methodology

6.3.1 Custom tooling

As previously discussed, the high nominal impedance of the filter of over 2 kΩ means that a regular Vector Network Analyzer (VNA) with a nominal impedance of 50 Ω would likely struggle to get an accurate measurement of the filter's characteristics. Additionally, the filter is balanced while the VNA's ports are unbalanced. It is therefore not possible to directly connect a VNA to this filter to measure its characteristics.

As such, a custom high-impedance differential probe was designed and built. As shown in figure 61, the probe simply consists of a Mini-Circuits T36-1 36:1 impedance matching transformer connected to an SMA connector on the low impedance side, and a pair of solder pads on the other. The ideal impedance ratio to use would have obviously been 1:40 to obtain a 2 kΩ impedance but no such transformer was available. The 1:36 impedance ratio transformer brings the impedance up to 1.8 kΩ, which should be close enough to 2 kΩ for the VNA to be calibrated successfully and return useful results. Solder pads were chosen on the high impedance side for two reasons. First, there are no off the shelf 2 kΩ connectors on the market anyway, and second, the probes only have to be connected a handful of times for experimentation.

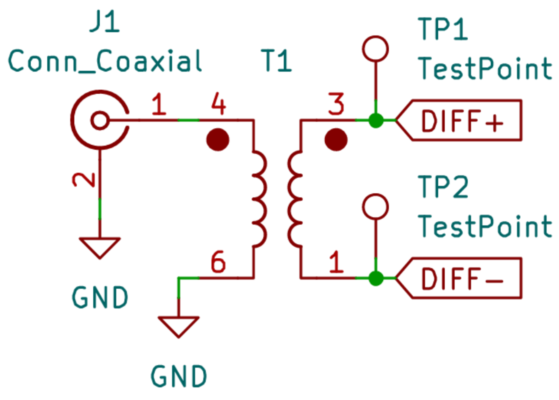


Figure 61: Schematic of the high-impedance probe used for taking measurements of the anti-aliasing filter prototype.

For cost reasons, this very simple circuit was layed out on a 2 layer PCB of which only the top layer is actually used, the back only serving as a ground plane. A picture of the high-impedance probe is shown in figure 62.

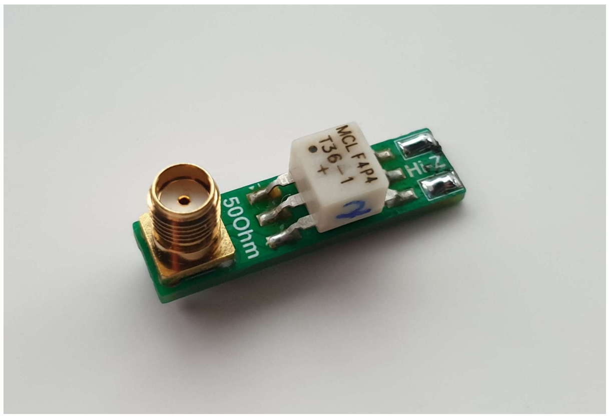


Figure 62: Close-up of the custom designed high impedance probed designed to be soldered on to the side of the device under test and connected to the VNA.

6.3.2 Prototype

The prototype board is as visible in figure 63. The filter is copied exactly as it is on the final PCB design and traces are run from it to the solder pads on the side of the PCB. Next, in order for the measurements to be possible, it is also necessary to provide a calibration set with which to calibrate the VNA before measurement. This simply consists in a a short circuit reference, an open circuit reference, a 2 kΩ reference, an isolation reference and a pass-through reference.

6.3.3 Methodology

The test setup for the anti-aliasing filter is as shown in figure 63. Before any measurement is made, the VNA is configured for a frequency range of 5 kHz to 50 MHz and is calibrated using the aforementioned calibration set. Next, one port of the VNA is connected to the input of the filter, and the other port is connected to the output. The response of the filter is finally measured and saved.

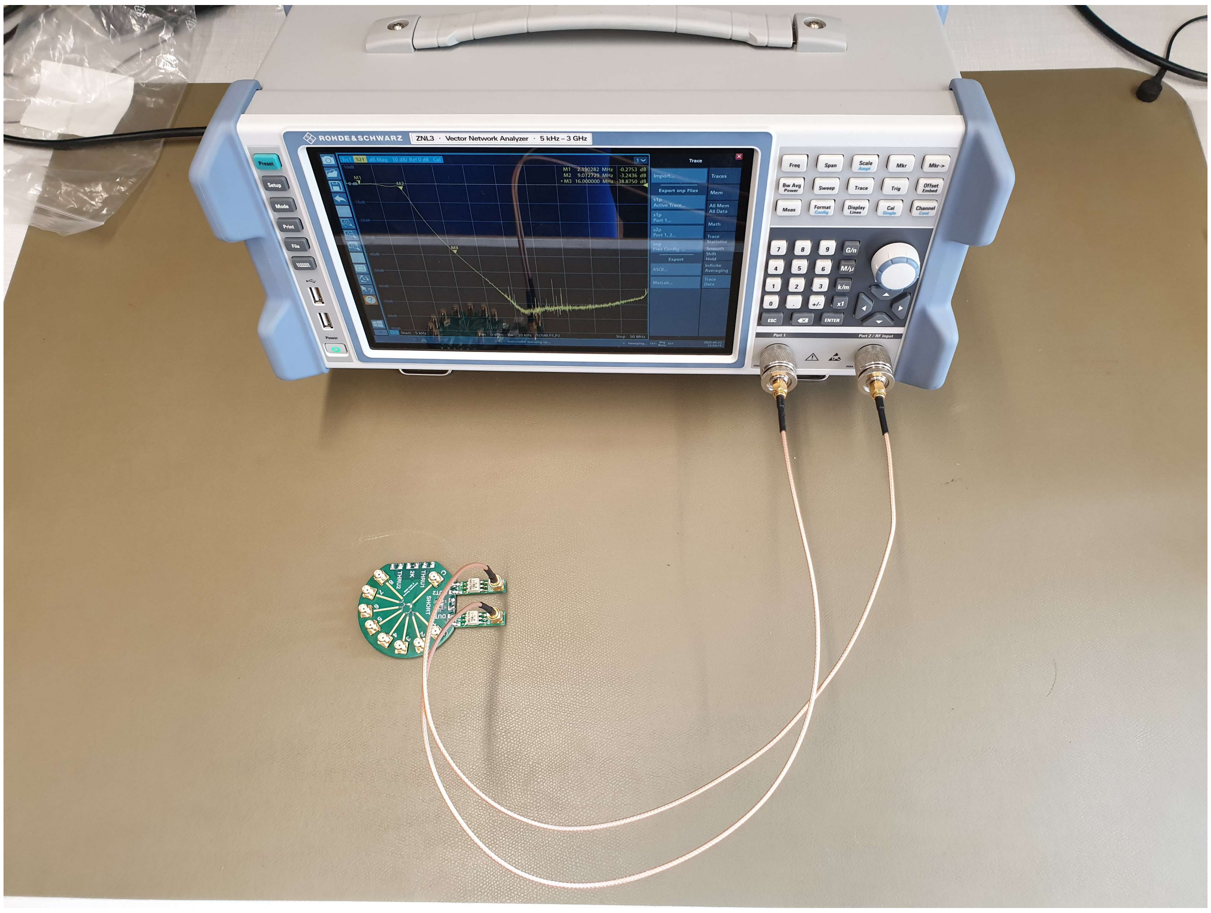


Figure 63: Picture of the test setup used to validate the performance of the anti-aliasing filter used in front of the ADC. The high-impedance probes are soldered to the DUT1 and DUT2 terminals.

6.3.4 Results

The measurement results taken with the VNA are as shown in figure 64 and compared with the simulation described earlier in this report. Firstly, the flattering out of the response after around 25 MHz is not actually a feature of the response, but merely a hardware limitation of the VNA where its dynamic range is not high enough to measure the very high attenuation of the filter at high frequencies. Similarly the jump in phase between 15 MHz and 20 MHz is simply the phase rolling over. The more concerning difference between simulation and experimental results however is the fact the that staring at around 12 MHz, the real filter's response continues to drop quite fast while the simulated filter tapers off. This yields an attenuation at 16.43 MHz of −40.669 dB versions the simulated result of 33.873 dB. Although this means the filter is better than expected, this is quite difficult to explain. One possible reason for this is the very slight impedance missmatch between the probe and the filter. Indeed, the filter is designed to have an impedance of 2 kΩ while the probe has an impedance of 1.8 kΩ.

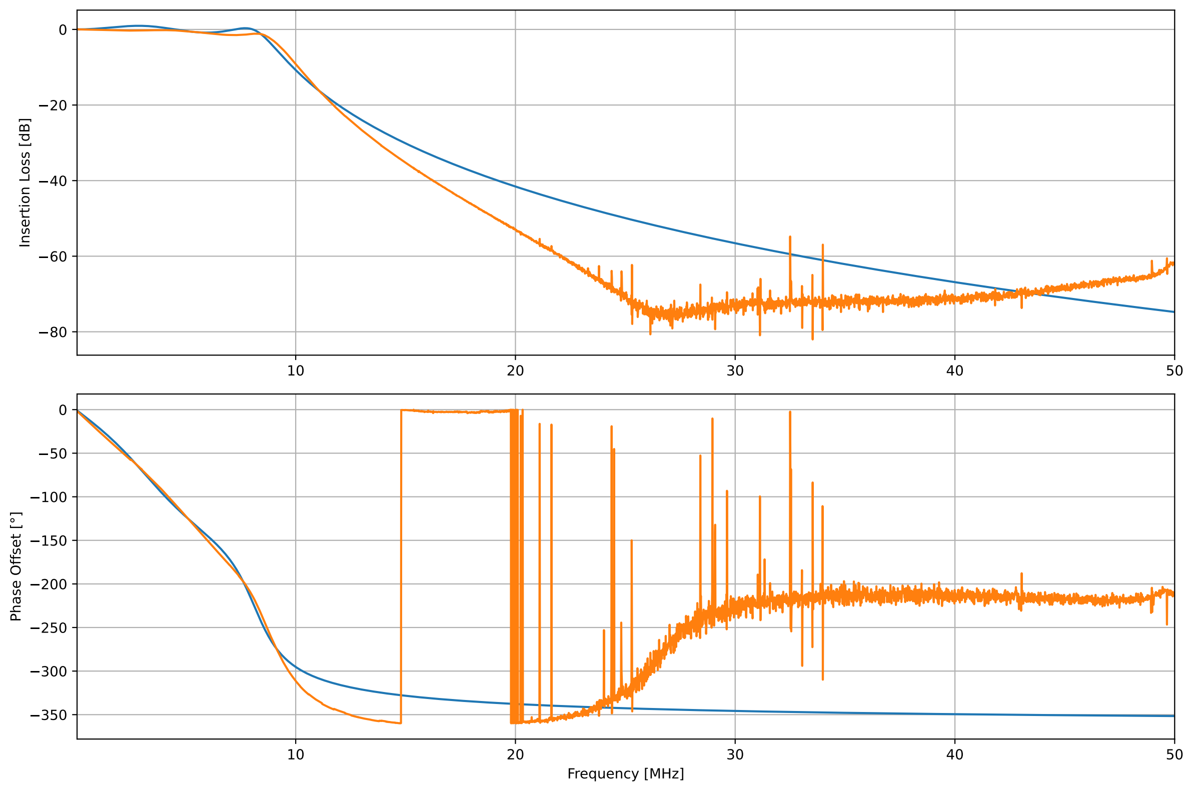


Figure 64: Comparison of the experimental results and simulation for the anti-aliasing filter.

6.4 Power Supply

The power supply being such an important part of the design, a prototype was built to be validated before the main PCBs were ordered. This section details the prototype, the testing setup used to validate it and the results of the tests conducted on it.

6.4.1 Prototype

Since, outside of routing the power traces, the layout of the power supply fits within a single layer, it was chosen to use a two layer PCB for the prototype, dramatically lowering its cost. A picture of the prototype is shown in figure 65. The prototype has one specific difference with the final PCB in that there is a jumper to allow directly powering the input of the 3.3 V switching regulator. This was done so that the power regulation circuit can still be tested separately of the USB Type-C Power Delivery negotiation circuit if said circuit were to not be functional

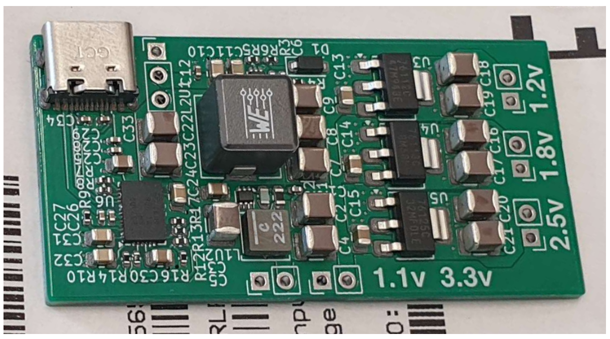


Figure 65: Picture of the power supply prototype taken before reflowing.

6.4.2 Methodology

The test setup is as shown in figure 66. The jumper between the type-C circuitry and the power input and a regulated bench power supply is connected in its place and configured to output 15 V. A Rigol DL3021 electronic load is then connected to each output rail set to sink the maximum current rating of said rail. Ideally, there should be one electronic load per rail and all loads should be applied at once, but only one electronic load is available at the law. Therefore, it is decided to instead check all the rails derived from 3.3 V separately, and then test the 3.3 V rail by drawing a current equal to that of its expected external load plus that of the downstream regulators. While under load, the voltage of each rail is measured by the electronic load using two additional sense wire soldered directly to the back of the headers on the prototype in order to avoid voltage drop over the wires affecting the measurement. The ripple is then measured using an oscilloscope configured for a bandwidth of 20 MHz as ripple in that range risks ending up in the IF and is too low in frequency to be removed properly by ferrite beads. Finally, one should note that when testing the voltages other than 3.3 V, the measured efficiency will be that of both the 3.3 V regulator and the downstream regulator combined and is thus only provided in order to rule out any major anomalous behavior.

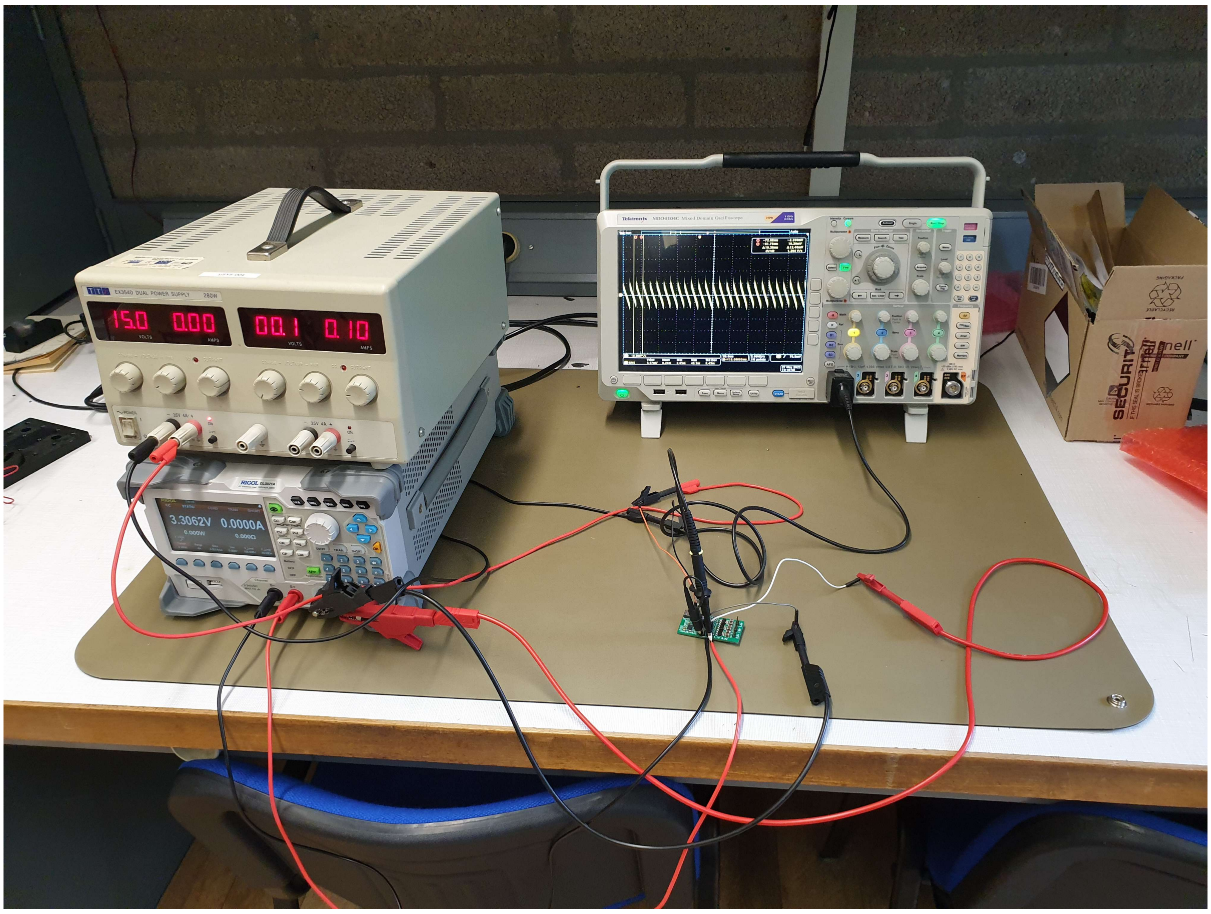


Figure 66: Picture of the test setup for the power supply prototype. The bench power supply and electronic load are situated one on top of the other on the left of the picture. The prototype is in the center and behind it is the oscilloscope.

6.4.3 Issues

At the first power on, two issues were immediately revealed. First, the output of the 3.3 V regulator exhibited extremely high ripple of over 200 mV RMS. Looking close at the ripple revealed that the voltage rapidly overshoots the target voltage before going back down. Second, the inductor of the 3.3 V regulator was getting extremely hot, indicating a higher than normal current through it. Unfortunately, a screen capture of the anomalous behavior was not taken before the issue was rectified.

The first thing that was suspected was a dead switching regulator. The part was swapped for a new one but the issue remained the same. Next, due to how the voltage overshot the target output voltage, it was suspected that there may be a solder blob shorting the inductor. Indeed, the inductor used has its pins under it which are not visible from the outside after being soldered. Removing the inductor revealed no solder bridge.

Finally, a closer reading of the datasheet of the 3.3 V switching regulator revealed the issue. The same 6.8 pF feed-forward capacitor had been used in the feedback circuit for the 3.3 V regulator that was used on the 1.1 V regulator. This is much smaller than the 470 pF value recommended in the datasheet. Knowing that this capacitor effectively increases the gain of the feedback circuit at high frequency completely explains the anomalous overshooting behavior. Indeed, when the switching controller enables the mosfet to start increasing the output voltage, the low loop gain causes it to overshoot the target voltage significantly, at which point the mosfet shuts off, and the output capacitor discharge slowly before reaching the point where the cycle repeats. Swithing the feed-forward capacitor to the correct value fully eliminated the issue and the final test results are as shown in the next subsection.

6.4.4 Results

Once the power supply section was fixed, it was then possible to get proper measurements of ripple, accuracy and efficiency. These results are layed out for each voltage rail at different load levels in table 12.

Starting with the 1.1 V rail, the highest deviation from the target voltage is at 100% load where the output voltageof 1.083 is 1.54% off target. This is thankfully perfectly acceptable as the FPGA recommends its 1.1 V rail be between 1.045 V and 1.155 V. In terms of ripple, one immediately notices the very high amount at 25% loading. This can be explained by a power saving feature of the switching converter chip used where under light load conditions switches from Pulse Width Modulation to a proprietary waveform which, while increasing efficiency up to 95% in ideal conditions, creates much higher ripple. Thankfully, even in that mode the ripple doesn't exceed the FPGA's supply requirements.

Onto the 1.2 V rail, the maximum deviation from target voltage also occurs at 100% loading where the voltage drops down to 1.196 V, only 0.33% off target. This is naturally perfectly acceptable as the ADC requires the 1.2 V rail to be between 1.14 V and 1.26 V. The measured ripple is slightly higher than expected at 3.35 mV RMS when under 25% loading conditions. This is again explained by the same power saving feature that the upstream 3.3 V regulator activates when under the light load of the 1.2 V rail only. Efficiency for this rail is, as expected for a linear regulator quite poor, dropping as low as 31.334% under full load conditions.

The 1.8 V rail is a similar story to the 1.2 V rail. The maximum deviation from target occurs at 100% loading where the voltage drops to 1.794 V which is again 0.33% off target. This is also perfectly acceptable as the ADC requires the 1.8 V rail to be between 1.71 V and 1.89 V. The same observation of high ripple ripple can be made and this time to an even larger extent as the 1.8 V rail only requires 116 mA at full load, way less than the 5 A current rating of the upstream 3.3 V regulator. The efficiency is very poor here again going as low as 43.256%.

The 2.5 V rail's highest deviation occurs also at 100% load where the voltage drops to 2.493 V which is only 0.28% off target. This is again perfectly fine as the FPGA only requires the 2.5 V rail to be between 2.375 V and 2.625 V. This time, thanks to the higher current at which the rail is tested, the ripple figure, especially from 50% load are very low going down to 0.95 mV RMS under load. This rail also has a higher efficiency than the two other linear regulators. This can be explained by the fact the less voltage has to be dropped over the regulator to each 2.5 V from the upstream 3.3 V rail.
Finally, the 3.3 V rail is also very precise only dropping down to 3.272 V which is 0.84% off from target. This is again well within range of all the circuits in the design. Ripple this time is consistently low outside of the no load condition likely because 25% load corresponding to 1.13 A is higher than the regulator's threshold for power saving mode. Efficiency is, as expected for a switching regulator, quite high at up to 92.5% at no load and as low as 85.4% at full load.

1.1 V Rail					
Load [%]	0	25	50	75	100
Voltage [V]	1.093	1.087	1.086	1.084	1.083
Ripple [mV RMS]	1.185	11.5	1.69	1.85	1.9
Efficiency [%]	-	90.567	80.45	77.428	68.762
1.2 V Rail					
Load [%]	0	25	50	75	100
Voltage [V]	1.203	1.2	1.199	1.197	1.196
Ripple [mV RMS]	0.695	3.35	2.55	1.37	1.43
Efficiency [%]	-	31.433	31.83	31.808	31.334
1.8 V Rail					
Load [%]	0	25	50	75	100
Voltage [V]	1.797	1.796	1.795	1.794	1.794
Ripple [mV RMS]	0.682	2.47	3.92	5.72	3.02
Efficiency [%]	-	43.258	45.934	46.702	46.981
2.5 V Rail					
Load [%]	0	25	50	75	100
Voltage [V]	2.499	2.496	2.496	2.494	2.493
Ripple [mV RMS]	0.681	3.63	0.95	0.95	0.95
Efficiency [%]	-	64.719	66.196	66.365	66.268
3.3 V Rail					
Load [%]	0	25	50	75	100
Voltage [V]	3.306	3.298	3.29	3.282	3.272
Ripple [mV RMS]	5.57	2.16	2.29	2.52	2.7
Efficiency [%]	-	92.507	90.565	88.718	85.41

Table 12: Measured voltage, ripple and efficiency of each of the power supply's rail at 5 different load levels.

6.5 Single Tuner Operation

In order to make sure that the tuners work as expected, an attempt was first made to try out each tuner individually.

6.5.1 Methodology

Testing the tuners was done by powering them up, tuning to various frequency bands and sampling their output with the ADC. The samples are then transfered in real time over USB to the host PC where the SDR++ software was used to inspect and demodulate the received signals in order to get a rough idea of whether the device is operating properly or not. Following this, a calibrated signal generator in conjuction with attenuators were used in order to determine the sensitivity of the receiver at various frequencies. The procedure for measuring the receiver's sensitivity is as follows:

1. Set the center frequency of the receiver and signal generator to the beginning of the frequency range.
2. Set the gain of the receiver to its maximum value.
3. Set the output power of the signal generator to its minimum value.
4. Enable the output of the signal generator.
5. Monitor the received signal on the FFT plot of the SDR++ software and increase the output power of the signal generator until a signal to noise ratio of 10 dB is achieved. The Minimum Discernible Signal (MDS) at that frequency and with the bandwidth equal to the width of a FFT bin is then $P_{\text{out}} + G_{\text{att}} - 10\,\text{dB}$.
6. Increment the frequency.
7. Go back to step 4.

In order for the results to be comparable with other devices on the market, it is then necessary to convert from the MDS per the bandwidth of a FFT bin, to the MDS for a standard bandwidth of 500 Hz. To understand how to do this, it is first important to understand the fact that the the MDS, by definition, also represents the noise floor of the receiver per the bandwidth in which it was measured. As such, converting from one bandwidth to the other is as simple as converting the MDS to dBm/Hz and then multiplying by the desired bandwidth, or more simply $MDS_{2,\mathrm{dB}} = MDS_{1,\mathrm{dB}} + 10\log_{10}\left(\frac{B_2}{B_1}\right)$.

The FFT in SDR++ is configured to compute 65536 bins with a total bandwidth of 8 MHz. Therfore the width of a bin is $\frac{8000000}{65536} \approx 122.07\,\mathrm{Hz}$. The correction factor to add to the data is therefore $10\log_{10}\left(\frac{500}{122.07}\right) \approx 6.124\,\mathrm{dB}$.

6.5.2 Issues

The first time the tuners were powered up, it was quickly noticeable that something was terribly wrong. Changing the center frequency of the receiver would result in the received signal being heavily frequency moduled smearing them all over the received bandwidth. Additionally, the receiver's PLL would constantly unlock and refuse tune to most frequencies. This erroneous behavior is shown in figure 67.

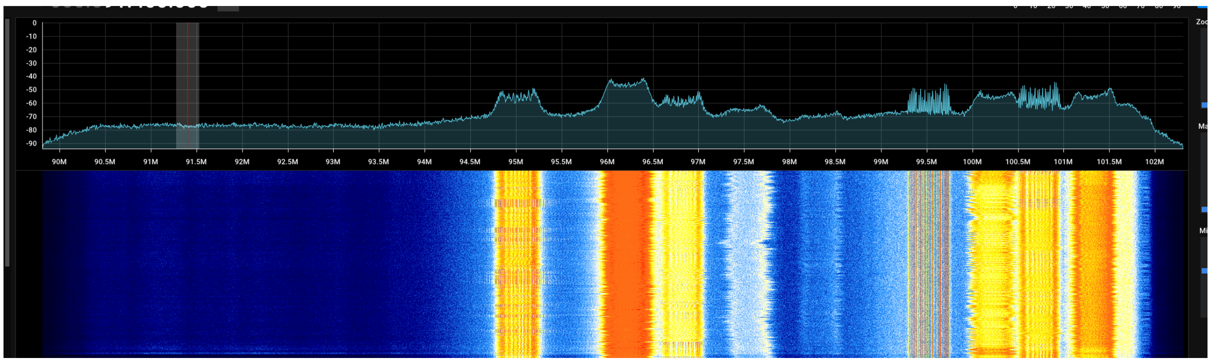


Figure 67: Spectrogram and FFT plot of broadcast FM signals being received with the tuners misbehaving. The phase noise issue is evident by how smeared the signals are on the spectrum. The rectangle in the top left is the marker for the demodulator's passband and can be ignored for this test.

After hours of debugging, the issue revealed itself to be a simple mistake in the tuner circuit schematic. While adding one of the resistors for the PLL loop filter, a "k" was mistakenly ommited in the value, meaning that the resistors that was soldered at those spots were $1.8\,\Omega$ instead of $1.8\,\mathrm{k}\Omega$. Swapping the resistors on one of the channels made it immediately behave as expected.

6.5.3 Results

Once the PLL instability issues were resolved, the tuner started receiving signals as expected. A spectrogram of the same broadcast FM frequency band received with the device is shown in figure 68. All signals now appear clear and sharp and changing the center frequency of the tuner works as expected instead of only working at some specific frequencies.

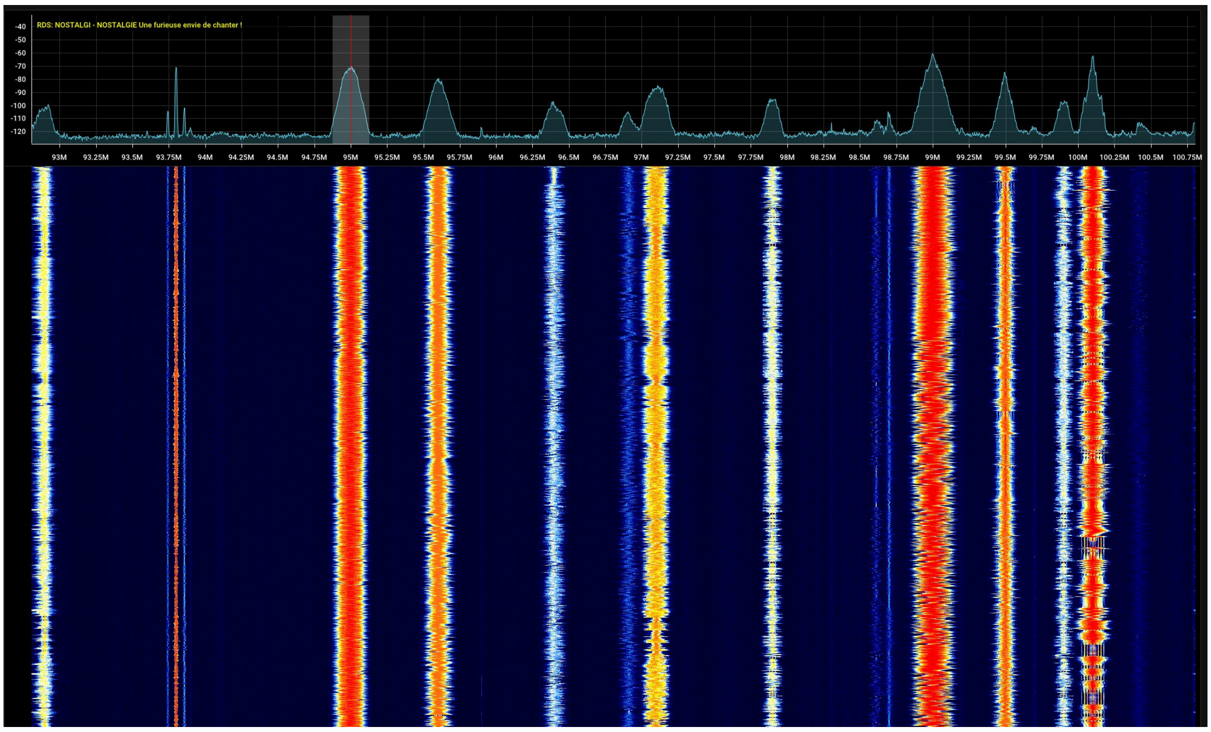


Figure 68: Spectrogram and FFT plot of broadcast FM signals being received after the PLL issues were resolved. Compared to figure 67, the signals are clearly no longer smeared.

Now that the tuners are known to be operating as expected, the performance of the entire frontend can be measured. The result of the minimum discernible signal (MDS) measurements is as shown in figure 69. The MDS starts out at around -127dBm before quickly dropping to -138dBm at 200 MHz. after around 600 MHz, the MDS increases again before ultimately reaching -127dBm at 1.75 GHz.

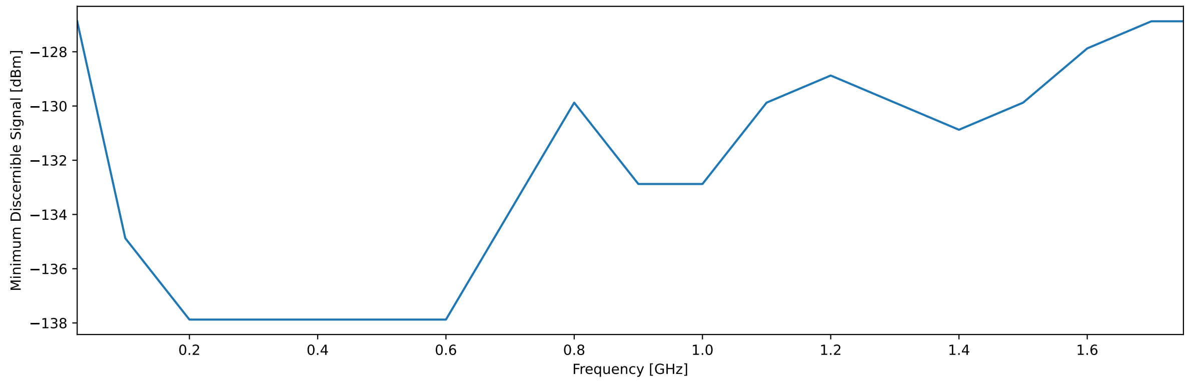


Figure 69: Minimum discernible signal measurement over the entire supported frequency range of the receiver for a bandwidth of 500 Hz at 100 MHz intervals.

The MDS figures of -138dBm between 200 MHz and 600 MHz is rather close to what other SDRs based on the R820T2 achieve. The Airspy R2 for example features an MDS of -140dBm over its entire frequency range. At the low and high end of the frequency range of this receiver however, the performance is over 13dB worse than the Airspy R2. Multiple factors could explain this discrepancy. In the case of the high frequency range, the ESD protection diode is, as mentioned in the hardware design section, significantly altering the input impedance of the receiver due to its parasitic capacitance. Finally, it is important to take these MDS results with a grain of salt. All measurements were taken manually by visually identifying when the test signal had reached 10dB above the noise floor. An error of around ±1dB can be expected.

7 Conclusion

This thesis investigated the design of a low cost coherent software defined radio receiver. First, goals were set in order to adequately direct the design process. These goals were based on the the current market leaders in terms of affordable coherent SDRs and the general needs of an education and hobbyist audiance.

Following that, a detailed review of the state of the art concerning low-cost coherent software defined radio receivers was undertaken, highlighting the various architectures and technologies used to achieve their designed puropose. After comparing these different designs to each other, it was concluded that the solution to making an affordable coherent software defined radio receiver is to reused mass-produced hardware that was originally meant for other purposes and work around their limitations.

A design was then drafted that uses 8 extremely low cost Rafael Micro R820T2 tuners originally meant to receive digital television, along with an onboard phase reference to obtain a coherent receiver without the use of expensive specialty ICs. The design was meant especially cost effective due to a recently released low-cost 8 channel ADC from Microchip. Adding up the cost of all components and prototype PCBs gives a total of 709€, that is only 354.50€ = $405 per completed receiver board. This is well under the price of the current cost leader in the market at $499, therefore the cost aspect of this project can be considered a success.

Following that, a detailed design description of each of the receiver's subsystem was undertaken. Starting with the frontend, the RF switches used to select between the antenna input and phase reference were chosen to prioritise low insertion loss over isolation, due to the fact that the phase reference will be shut off when not in use anyway. When designing the clock subsystem, the *Si5351c* clock generator was chosen for its low cost and for being able to supply every clock necessary for the design. For distribution, a simple resistive network was selected. Continuing on to the phase reference, a special phase-balanced splitter designed in order to create 8 in-phase copy of the reference, one for each frontend. A resistive splitter design was used because high insertion loss was a desired properly to avoid overloading the tuners and a high bandwidth was required. Moving on to the ADC subsystem, an anti-aliasing filter was designed to prevent out of band signals from ending up in the final passband of the receiver. The impedance of this filter was chosen to be 2 kΩ differential to match the tuner's output. A passive ADC input driver was selected and designed to reduce cost and complexity. To process the data coming from the ADC, a Lattice ECP4 FPGA was selected due to the existant of a free and open source toolchain for it. After processing, a FTDI USB FIFO chip was used to forward the samples to a host computer. This USB chip was selected because of the higher cost and poor reputation of alternatives such as the Cypress FX3. Finally, the power supply was designed to use both switching and linear voltage regulators in order to provide clean power to the ADC without the need to dissipate an immense amount of power as heat.

After the hardware design comes the firmware design. A verilog 6502 soft core was chosen as the CPU to run command processing and housekeeping tasks on the receiver. This choice was motivated by the simplicity of its design leading to very low LUT usage inside of the FPGA. To allow the CPU core to access peripherals and memory, a memory mapping unit was developed and optimised for low LUT usage. Then, the modules for interfacing with the ADC were developed. Specifically, a module to convert from double data rate to single data rate, followed by another module to de-interleave the samples. These module had to be carefully designed to ensure they could run at the very high clock rate of the ADC, but allowed subsequent circuits downstream to run at a much lower frequency. Next, a frame packer module was designed to pack samples into 1024 byte packets for transfer over USB. Finally a controller for the USB FIFO IC was designed based on a finite state machine and a few buffers.

Finally, various subsystems and the device itself were subject to testing. Starting with the clock generator, testing simply involved verifying that it was outputting a clear and stable clock. Unfortunately, this was not the case and the issue was diagnosed as the power rail sequencing. Fixing this issue involved modifying the circuit so that it would run entirely off the 3.3 V rail and adding an inline attenuator on the differential clock going to the ADC. Next the phase reference splitter was tested by using a VNA and comparing the response betwween the input and each output. This showed that even at up to 1750 MHz, the phase difference between channels was less than 1%. Moving on to the anti-aliasing filter, testing it required designing custom high impedance probe so that a 50 Ω VNA could adequately measure the performance of a filter with a 2 kΩ impedance. Measuring this filter revealed that its amplittude response tapered off less than the simulated filter, making it actually better than expected at its intended use case. Next, the power supply section was tested by loading eachb rail to its maximum design current. This showed that the ripple was satisfyably low and that all voltages were well within the required accuracies. Finally, the entire receive chain was tested at once by connecting an antenna to one of the tuners and attempting to receive broadcast FM signals. This revealed a mistake in the schematic where a "k" was ommited from the resistor value resulting in 1.8 Ω resistors being put in place of 1.8 kΩ resistors in the loop filter circuit of the tuners. When in that condition, the PLL would fail to lock and signals would be extremely smeared in the frequency domain. Fixing the issue immediatel caused the receiver to behave perfectly.

Unfortunately, due to time constraint, full multichannel coherent operation could not yet be achieved. This is not due to any design flaw of the receiver, but simply because problems in the various subsystem cost precious debugging time leaving none for implementing the necessary components for coherent operation. It is hoped that coherent operation will be demonstrated shortly after the deadline of this thesis.

8 Acknowledgements

Many people supported me during this project. Their names and contributions are listed in alphabetical order of last names below:

Morgan Diepart: Thanks to Morgan Diepart for his feedback on the design of the PCB, his help ordering the components, his help using the test equipment at the lab to make the measurements that were essential to the validation of the design as well as his help troubleshooting the clock generator issues faced during testing.

Samuel Dricot: Thanks to Samuel Dricot for assembling the PCBs and replacing chips that had failed.

Pieter Ibelings: Thanks to Pieter Ibelings (RFspace Inc.) for donating the R820T2 tuner ICs which are difficult to source in low quantities.

Jean-Michel Redouté: Thanks to Jean-Michel Redouté for all his feedback and for being my advisor even though I am not an electrical engineering student.

Youssef Touil: Thanks to Youssef Touil (Airspy SA.) for his invaluable guidance on using the R820T2 tuner IC.

I would also like to thank all of the Microsys staff for being welcoming and guiding me around the lab.

9 Statement on AI use

No AI was ever used at any point during, or for any purpose relating to this masters thesis.

References

[1] 6502.org. 6502 Microprocessor. http://www.6502.org/, 2025.

[2] Airspy. Airspy R2 SDR Receiver. https://airspy.com/airspy-r2/, 2025.

[3] Airspy. R820T Datasheet. https://rtl-sdr.com/wp-content/uploads/2013/04/R820T_datasheet-Non_R-20111130_unlocked.pdf, 2025.

[4] Alexandre Rouma. SDR++. https://www.sdrpp.org/, 2025.

[5] AMD. AMD Zynq™ UltraScale+™ RFSoCs. https://www.amd.com/en/products/adaptive-socs-and-fpgas/soc/zynq-ultrascale-plus-rfsoc.html#tabs-b3ecea84f1-item-922eeefd59-tab, 2025.

[6] Analog Devices. SHARC Processor Architectural Overview. https://www.analog.com/en/lp/001/sharc-processor-architectural-overview.html, 2025.

[7] Arlet Ottens. A Verilog HDL model of the MOS 6502 CPU . https://github.com/Arlet/verilog-6502, 2025.

[8] Ettus Research. TwinRX Daughterboard. https://www.ettus.com/all-products/twinrx/, 2025.

[9] Ettus Research. USRP X310. https://www.ettus.com/all-products/x310-kit/, 2025.

[10] JLCPCB. 6 Layer PCB Prototype Stackup. https://jlcpcb.com/6-layer-pcb, 2025.

[11] KiwiSDR. KiwiSDR TDoA. http://kiwisdr.com/ks/using_Kiwi.html, 2025.

[12] KrakenRF. Diagram taken from the manufacturer's wiki. https://github.com/krakenrf/krakensdr_docs/wiki/, 2023.

[13] KrakenRF. KrakenSDR - A Coherent Receiver for Radio Direction Finding. https://www.krakenrf.com/, 2025.

[14] Lattice. ECP5 / ECP5-G FPGA Family. https://www.latticesemi.com/Products/FPGAandCPLD/ECP5, 2025.

[15] Lime Microsystems. FPRF MIMO Transceiver IC With Integrated Microcontroller. https://cdn.sanity.io/files/yv2p7ubm/production/dc7fec4c445b008b9a1d35ea5a16a2850e923fe2.pdf, 2019.

[16] Lime Microsystems. Systems. https://limemicro.com/systems/, 2025.

[17] Microchip. MCP37211-200. https://www.microchip.com/en-us/product/mcp37211-200, 2025.

[18] MyriadRF. LimeSDR-USB. https://wiki.myriadrf.org/LimeSDR-USB, 2019.

[19] MyriadRF Forum. Discussion on the manufacturer's forum about USB issues. https://discourse.myriadrf.org/t/just-trying-to-get-started-having-no-luck/1127, 2017.

[20] NXP. Layerscape® Access LA9310 Programmable Baseband Processor. https://www.nxp.com/products/processors-and-microcontrollers/arm-processors/layerscape-processors/layerscape-access-la9310-programmable-baseband-processor:LA9310, 2025.

[21] Open Cores. Open Cores. https://opencores.org/, 2025.

[22] OrangeCrab FPGA. OrangeCrab FPGA. https://orangecrab-fpga.github.io/orangecrab-hardware/, 2025.

[23] Osmocom. History and discovery of RTL-SDR. https://osmocom.org/projects/rtl-sdr/wiki/Rtl-sdr#history_and_discovery_of_rtlsdr, 2025.

[24] Osmocom. RTL-SDR Osmocom project page. https://osmocom.org/projects/rtl-sdr/wiki/Rtl-sdr, 2025.

[25] Per Vices Corporation. Per Vices SDR hardware pricing. https://www.pervices.com/pricing/, 2025.

[26] Per Vices Corporation. Spectrum Monitoring. https://www.pervices.com/spectrum-monitoring/, 2025.

[27] Project Trellis. Project Trellis. https://prjtrellis.readthedocs.io/en/latest/, 2025.

[28] Skyworks. Si5351C-B-GM LVCMOS 8 Output Clock Generator. https://www.skyworksinc.com/en/Products/Timing/CMOS-Clock-Generators/si5351c-b-gm, 2025.

[29] S. N. Tatarinskiy, M. V. Kavun, and D. N. Trembach. Diversity reception system. In *2006 16th International Crimean Microwave and Telecommunication Technology*, volume 2, pages 1014–1014, 2006.

[30] Texas Instruments. LMX2572LP 2-GHz Low Power Wideband RF Synthesizer With FSK Modulation. https://www.ti.com/lit/ds/symlink/lmx2572lp.pdf, 2025.

[31] Texas Instruments. Sink-only USB Type-C® and USB Power Delivery (PD) controller with no firmware development required. https://www.ti.com/product/TPS25730, 2025.

[32] Texas Instruments. The TMS320 Family of Digital Signal Processors. https://www.ti.com/lit/an/spra396/spra396.pdf, 2025.

[33] USB Implementers Forum. USB Charger (USB Power Delivery). https://www.usb.org/usb-charger-pd, 2025.

[34] Yagoub, Reda and Benaissa, Mohamed and Benadda, Belkacem. Diagram taken from: Nearby Carrier Detection Based on Low Cost RTL-SDR Front End. https://www.researchgate.net/publication/333007039_Nearby_Carrier_Detection_Based_on_Low_Cost_RTL-SDR_Front_End, 2019.