# Saturn Development Activities

This document describes the development steps required

# Project Overview

“Saturn” is a new FPGA radio project which will couple a Xilinx FPGA for the ADC/DAC and immediate DSP functions, together with a Raspberry Pi4 compute module for the data handling, protocol code and local SDR application. It will be possible to use Saturn in several ways. Other radios achieve each of these individually, but Saturn will be able to implement all of them.

|  |  |
| --- | --- |
| Diagram  Description automatically generated | The processor module is used simply to move data using protocol 1 – like the Red Pitaya’s processor. Low demand on the processor. Wired ethernet connection to PC running Thetis or other app. |
| Diagram  Description automatically generated | The processor module is used simply to move data using protocol 2. This will have higher demand because the data rate is higher. Wired ethernet connection to PC running Thetis or other app. |
| Diagram  Description automatically generated | The processor executes an SDR app such as Pihpsdr or linhpsdr. No PC required, and high quality display outputs are available. |
| Diagram  Description automatically generated | The processor execute Pihpsdr and has an attached 7” RPi touchscreen display. Possibly single ADC or 14 bit ADC version, with Apollo-like RF module. This could be a lower cost small radio, but there may be no market for it now. |

Saturn uses a Raspberry Pi4 Compute Module as its local processor. The unique element to the Pi4 CM is a single lane PCI Express interface. This provides the connection to the FPGA for both low bandwidth data settings and high bandwidth data transfer.

Saturn is the project name for the board accommodating ADC/DAC, FPGA and Raspberry Pi. It does not (yet, at least!) describe radios using it. Saturn does have some moons, which might be appropriate for that purpose when the time comes.

# Development Required

To make Saturn happen there are several aspects:

1. The FPGA itself – being coded using Vivado and making substantial use of Xilinx pre-defined IP modules. We have all the functionality in the design.
2. The PCB. Prototype 2 is fully working, but we need to replace the codec because of obsolescence.
3. A “protocol 2 app” for the Raspberry pi; this will accept and send protocol 2 packets to a remote SDR client.
4. Subsequently, a port of Pihpsdr will be needed to provide local processing.
5. Utility code for the Raspberry pi. This is for functions such as the boot loader, to program the FPGA configuration memory a simple GUI app has already been created.
6. Potentially some kind of thin clint app
7. Documentation is always required!

# Things to Do

1. There is still some driver code that is untested, but not needed for the “P2” app. If made a “native” interface into Pihpsdr it would be needed – eg setting Alex filters individually.
2. The DC spike is much reduced – comparable with band noise on 14MHz – but there is still a spike.
3. Should the P2 app be command line, or should it be a GUI app with a “stop” button and minimal display of status?
4. Do we need a P1 app? (I assume “no” but it could be done, relatively easily)
5. Do we need any other apps? The “audio test” one seems useful for testing microphones etc. If we add a new codec, then having some more settings (and even simple equalisation) may become possible.
6. If we add an ATU, then the interfaces into Pihpsdr will need to be added (they are quite simple – essentially “push” frequency and antenna changes out as serial CAT messages)
7. For Thetis/PC users, how would connections to the ATU and Andromeda be handled? Is there a recognised way to interface serial through ethernet, so that the PC thinks it is seeing a serial port but the messages travel to the pi by ethernet?
8. We will need to test with an Andromeda front panel, for those that want that. I assume there will also be a non-Andromeda variant.
9. We/someone could look into a “thin client” version, with remote UI; but that is potentially a substantial task.