Boston University
Electrical & Computer Engineering
EC 464 Senior Design Project

Second Prototype Testing Plan
3/3/2022

RF Spectrum Analyzer

By
Team 26

Isaac Yamnitsky isaacy@bu.edu
Kakit Wong kakit@bu.edu
Yana Galina yanag@bu.edu
Jaime Mohedano jaimem@bu.edu

Required Materials

Hardware:

- Xilinx RFSoC 2x2 board
- Personal Computer (x2)
    - PC 1 (Yana's laptop)
    - PC 2 (Jaime's laptop)
- RFSoC Power Cable
- USB Cable
- Ethernet Cable
- Antenna

Software:

- Jupyter Notebooks
    - PYNQ framework
    - Numpy library
    - RFSoC Studio
- Redis Software
    - HiRedis
- Frontend
    - Python 3
    - Dash

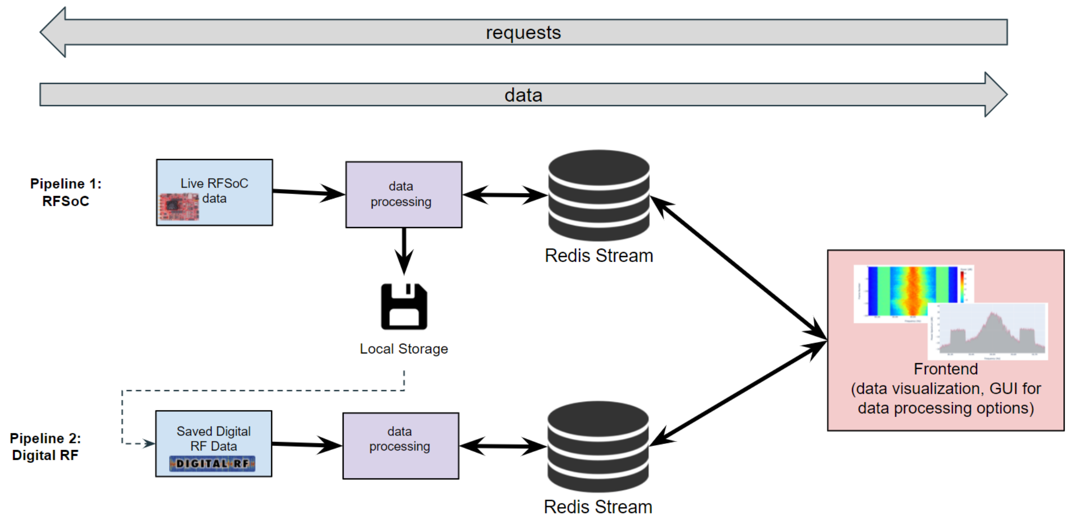

Figure 1: A block diagram of our design

Set Up

Pipeline 1:

Our setup involves two main components:

- The hardware side consists of the RFSoC board loaded with a data collection Jupyter notebook script.The board is connected to power and the internet via ethernet cable. One RX channel is fitted with an antenna. PC1 is needed to connect to the board and run the necessary scripts.When the board is powered on and running, the script takes incoming data, converts it to spectrum data using Fast Fourier Transforms, and sends that data over the internet into a Redis database running on PC2.
- The software side consists of a PC2 running the Dash web application as well as the Redis database. This web application processes data streaming into the Redis database from the board and displays it in the browser with an interactive graph format.

We will be comparing the results of the data processing script we created against the live data being displayed within the RFSoC Studio application,which come preinstalled on the board.

Pipeline 2:

The setup consists of a PC running a redis database, a back end python server, and a front end Dash web app. The back end streams requested Digital RF data into the front end to be displayed on the graphs there, through the Redis database.

Pipeline 2 had been tested and demonstrated in our first prototype, so we will be focusing on Pipeline 1 for this test.

Pre-testing Setup Procedure

Pipeline 1:

To get the board running:

- Plug the power cable and the ethernet cable into the appropriate sockets.
- Connect the board to PC1 using the USB cable
- Power on the board. Wait about 1 minute for the board to boot up.
- In a browser in PC1, go to http://192.168.3.1/lab to access the Jupyter Notebooks available to the board.
- Open the “voila_rfsoc_spectrum_analyzer.ipynb” and the “simplestream.ipynb” scripts

To get the Redis server and web application running:

- On PC2, run the command “redis-server --protected-mode no” in a terminal window
- In a separate terminal window, navigate to the RFSoC-Spectrum-Monitoring repository and run ‘conda activate rfsoc’ and ‘python frontend/app.py’
- In a browser, go to http://127.0.0.1:8050/ and navigate to the “streaming” tab

Pipeline 2:

- On PC1, open up a linux terminal and run the command “redis-server”.
- In a second terminal, navigate to the RFSoC-Spectrum-Monitoring repository and run ‘conda activate rfsoc’ and ‘python frontend/app.py’
- In a third terminal, navigate to the RFSoC-Spectrum-Monitoring repository and run ‘conda activate rfsoc’ and ‘python backend/redis_back_end.py --type drf’

Testing Procedure

Pipeline 1:

1) First, run the RFSoC Studio notebook on the board from PC1. Display the spectrum analyzer for receiver channel 0. Observe the results. This data will serve as our baseline for comparison. Stop the notebook.
2) Run the “simplestream.py” script on the board with the “channel” variable set to 0.
3) On PC2, find the “bu_rfsoc” channel within the web application, and hit “play”. Metadata for the stream should appear on the sidebar, and data should start appearing in the graphs. Toggling the “log scale” option should toggle the scale of the graphs. Hitting “pause” should pause the graphs.
4) Compare the results. The data being displayed on our web application should look similar to the data displayed on the RFSoC Studio application for channels 0.

Pipeline 2:

1) In a browser tab, navigate to http://127.0.0.1:8050/. Stay in the “Digital RF” tab. Leave the default input file “C:/Users/yanag/openradar/openradar_antennas_wb_hf/” and click “Choose input directory”. Metadata should appear in the sidebar.
2) Hit “Load data”. Data from the Digital RF file should be played on the graphs, and then stop when the complete amount of data has been played. The log scale can be toggled on and off.
3) Select a different channel and repeat step 2.

Measurable Criteria

Pipeline 1:

The criteria for successful running and output is as follows:

1. The RFSoC should successfully capture raw data, convert it to frequency domain data, pack the data into a JSON format and stream the data into the Redis server.
2. The Redis server should receive the data stream and forward the stream to the frontend.
3. The dashboard should properly display the spectrum data. The user should be able to pause and play the data, as well as toggle between the logarithmic scale for the graph.

Pipeline 2:

1) The Dash application should allow the user to to select the input Digital RF file, select between different channels, and play the appropriate data.
2) The user should be able to toggle the logarithmic scale for the graph.

Scoring

Pipeline 1:

RFSoC Channel	Frequency Domain Data Transmitted to Redis (T/F)	Data Displayed on Dashboard (T/F)	Data Matches RFSoC Studio Baseline (T/F)	Data can be played, paused. Log scale can be toggled (T/F)
RFSoC Channel 0				

Pipeline 2:

Digital RF Channel	The user can select the sample range, and number of bins (T/F)	Data played back properly (T/F)	Log scale can be toggled (T/F)
Discone			

Monopole			