Boston University
Electrical & Computer Engineering
EC 464 Senior Design Project

Final Prototype Testing Plan
4/4/2022

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
- Frontend
    - Python 3
    - Dash

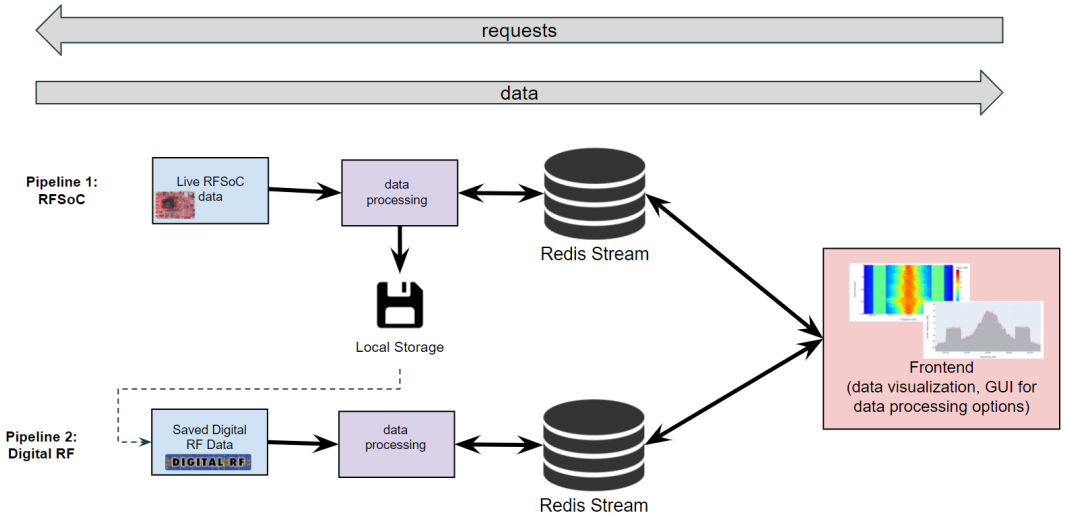

Figure 1: A block diagram of our design

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 connected to a TX channel with a cable. 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. There is also a second script which waits for requests for data from the frontend, and responds by dumping raw data into the Redis server.
- 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.

Pipeline 2:

This 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.

Integration of Pipelines 1 and 2:

We are able to download data from the board, store it in the DigitalRF playback, and play it back as described by Pipeline 2. This setup requires all of the aforementioned components of Pipelines 1 and 2.

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. When prompted for a password, enter “xilinx”.
- Open the “simplestream.ipynb” and “commands.ipynb” scripts

- Set the redis server “host” values in both scripts to match the public IP address of PC1.

To get the Redis server and web application running:

- On PC2, run the command “redis-server --protected-mode no” in a terminal window
- Make sure the redis server configurations (host and port values) in front_end/config.yaml and back_end/config.yaml match the IP address of the redis server (should be ‘localhost’ in this case).
- 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 --protected-mode no”.
- Make sure the redis server configurations (host and port values) in front_end/config.yaml and back_end/config.yaml match the IP address of the redis server (should be ‘localhost’ in this case).
- In a second terminal, navigate to the RFSoC-Spectrum-Monitoring repository and run ‘conda activate rfsoc’ and ‘python front_end/app.py’
- In a third terminal, navigate to the RFSoC-Spectrum-Monitoring repository and run ‘conda activate rfsoc’ and ‘python back_end/drf_back_end.py’

Testing Procedure

Test 1: Digital RF playback:

1) In a browser tab, navigate to http://127.0.0.1:8050/. Stay in the “Digital RF” tab.
2) Click the “Select Digital RF Data” button. Type in “C:/Users/yanag/openradar/openradar_antennas_wb_hf/” under “Digital RF Directory”. Hit “Select”. Choose between the different channels, select the sample range to play back, the number of Fast Fourier Transform bins, and modulus and integration values.
3) Hit “Load data”. Data from the DigitalRF file should be playable. Metadata should appear under the “Metadata” accordion tab. The user should be able to play, pause, and rewind the data in the “Digital RF Options” accordion tab.
4) The user should be able to modify graph parameters under the “Spectrum graph settings” and “Spectrogram graph settings” tabs. This includes toggling the scales

between linear and logarithmic, changing the y-axis limits, tracking min and max points (for the spectrum graph), and changing the color scale (for the spectrogram graph).
5) Repeat Steps 2-4 with different values.

Test 2: Live Streaming:

1) Run the "simplestream.py" script on the board with the "channel" variable set to 1. This is the RX channel in loopback with the TX channel. Set the TX channel to output a gain of 0.8 at 600 MHz.
2) On PC2, open the Web application and navigate to the "Streaming" tab. Under the "Stream options" accordion tab, find the "bu_rfsoc" channel, and hit "play". Metadata for the stream should appear on the sidebar under the "Metadata" accordion tab, and data should start appearing in the graphs. The same graph options as described in Test 1 Step 4 should work as well.
3) Confirm that the graph data "makes sense" and is accurate. There should be a noticeable peak at 600 MHz, with noise elsewhere.

Test 3: Downloading data from board and playing it back:

1) Run the "commands.py" script on the board with the "channel" variable set to 1. This is the RX channel in loopback with the TX channel. Set the TX channel to output a gain of 0.8 at 750 MHz.
2) On PC2, open the Web application and navigate to the "Streaming" tab. Under the "Download Options" accordion tab, click the "download data from board" button. A modal form should appear. Under "choose board", find the "bu_rfsoc" channel. Select a time range of 2 seconds, and click "Download Data".
3) After a few seconds, the browser should start downloading a zip file with the requested data. Extract the zip file, and play back the Digital RF data contained within as described by Test 1. The data displayed should have a clear peak at 750 MHz.

Measurable Criteria

Test 1:

1) The Dash application should allow the user to to select the input Digital RF file, select between different channels, select the sample range, and integration values.
2) Metadata should be clearly displayed under the appropriate tab.
3) The user should be able to pause, play, and rewind the data back to the beginning.

4) The user should be able to toggle the graph scales between linear and logarithmic, change the y-axis limits, track minimum and maximum points (for the spectrum graph), and change the color scale (for the spectrogram graph).

Test 2:

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 frontend should display the board under "stream_names" when the board is active.
4. The user should be able to play and pause the live stream.
5. New data from the board should be displayed at a rate of greater than 1 frame per second.
6. The graphs should display accurate data. For example, they should show a clear peak at 600 MHz.
7. Metadata for the stream should appear on the sidebar.
8. The same interactive graph options as described in Test 1 (changing graph scaling, tracking points, changing color scheme) should be functional.

Test 3:

1) The RFSoC should be able to respond to a request for data from the front end and send the appropriate data to the front end through Redis.
2) The front end should display the board in the request options form when the board is active.
3) The user should be able to select a duration of time for data, and choose the name of the download file.
4) The front end should successfully convert the data from the board into a Digital RF file.
5) This digital RF file should be downloaded into the user's file system in the format of a zip file.
6) This digital RF file should be able to be successfully played back by the Digital RF player as described in Test 1.
7) The graphs should display accurate data when playing back the downloaded data. For example, they should show a clear peak at 750 MHz.

Scoring

Test1:

	T/F
User can select a Digital RF Directory	
Channels and sample ranges appropriate for the file appear in the selection form	
Metadata appears after a request has been processed	
User can pause data	
User can play data	
User can rewind data back to beginning	
User can toggle graph between linear and logarithmic scales	
User can change y-axis limits of graph	
User can track minimum and maximum points on the spectrum graph	
User can change colors scheme of spectrogram graph	

Test 2:

	T/F
User can find “bu_rfsoc” under “Stream Options” tab	
User can pause and play data	
Graph frames are updated at rate greater than 1 per second	
The graphs depict accurate data (eg a peak at	

600 MHz)	
Metadata should appear on the sidebar	
Interactive graph options as described in Test 1 criteria work properly	

Test 3:

	T/F
User can find “bu_rfsoc” in the download request form	
A zip file is downloaded after several seconds	
After zip file extraction, the Digital RF Playback feature of the web application is able to read and process the data	
The graphs depict the data accurately (eg there should be a clear peak at 750 MHz)	