RFSynth: Data generation and testing platform for spectrum information systems

Hari Prasad Sankar*
ECE, UC San Diego

Raghav Subbaraman*
ECE, UC San Diego
* Equally credited authors

Tianyi Hu
JASR Systems, San Diego

Dinesh Bharadia
ECE, UC San Diego

***Abstract*—The paper presents a scalable RF data generation framework which aims to address the challenges of limited data generation/testing platforms for spectrum sensing systems. The proposed framework combines simulation and real-world data generation methods to enable large and diverse data sets for training and testing RF ML models and spectrum sensing systems. The framework includes modules for metadata generation which allows for easy experimentation. The effectiveness of the proposed framework is demonstrated through experiments including signal detection and modulation classification. This paper contributes to the development of a comprehensive framework for generating RF IQ data with ease that can significantly reduce the development and deployment time of complex wireless systems.**

***Index Terms*—Wireless data generation, RF Machine learning, spectrum sensing, wireless system testing**

I. Introduction

The advent of next-generation wireless systems, including the Internet of Things (IoT) and 5G mmWave technologies, underscores a critical challenge: the efficient utilization of dynamically available spectrum spaces. With the expansion of these technologies, failing to capitalize on these spaces can incur significant opportunistic costs. Essential to harnessing this potential is the development of systems capable of intelligently monitoring RF spectrum activity and allocating resources across diverse applications. However, the design and testing of these data-intensive spectrum information processing systems present considerable hurdles. A primary challenge lies in generating *controlled RF activities with accurate metadata*, essential for testing these systems comprehensively. Traditional methods, relying on off-the-shelf devices, offer limited control over signal transmissions, complicating the process further. Additionally, the burgeoning use of deep learning in the physical layer amplifies the demand for extensive, well-labeled datasets for algorithm tuning and RF ML model training [1]. Despite commendable efforts to compile and share such datasets [2], [3], creating customized, large-scale datasets remains a daunting task due to configurability constraints, computational limitations, and memory requirements.

Existing approaches to address these challenges primarily involve simulation environments and the use of real devices for system testing. While useful, these methods fall short in accurately representing real-world scenarios systems may encounter, constrained by computational power, device availability, and signal diversity. Industry-standard signal generators offer some capabilities but come with prohibitive costs. Moreover, testing interference management, signal detection in the presence of interference, and other advanced functionalities is hampered by the lack of control over experimental conditions. If one has to develop an end-to-end solution to tackle data availability and address testing problems faced by spectrum data processing systems, It is desired to have the following functional features:

- A comprehensive signal generation engine that provides high fidelity data,
- Precise control over data generation and transmission,
- A forensic framework that ensures repeatability and facilitates algorithm tuning,
- User-friendly access for broad applicability.

In this paper, we propose, *RFSynth*, a system that can generate IQ data and establish a controlled RF environment with aforementioned qualities. We discuss our efforts to tackle design challenges, including creating abstractions for RF data generation, accurately simulating non-idealities, signal superposition, and generating precise metadata for effective training and forensics. We validate our contributions through case studies demonstrating various simulated and real-world OTA scenarios. RFSynth is available under an open source license at https://github.com/ucsdwcsng/rfsynth.

II. Background: Testing and Data requirements for spectrum information systems

Systems that process wireless spectrum information at a fast pace and identify malicious devices that cause unintended RF transmissions or change the balance of the wireless network are the need of the hour. These concerns have crucial implications and efforts like IARPA-SCISRS and DARPA-RFMLS, which necessitate to perform blind signal analysis coupled with spectrum sensing. These software platforms require testing systems whose requirements deviate significantly from what is currently in prevalent use and data generation platforms for training the RF ML models.

In this specific context of wireless system design, the development process has been mostly dependent on the simulation environments, and testing such systems has been mostly done on real devices for performance/functional evaluations. But even with real devices, it is challenging to fully represent a scenario that the system will encounter on the field because of limits on computing, the number of wireless devices required, and signal requirements. Conventional industry-standard signal generators such as Keysight Pathwave signal generators [4], Rohde & Schwarz WinIQSim2 [5] which have such capabilities are also extremely costly. For example, if a designer wants to test a system's performance against interference management and suppression, doing so with real devices can be hard because of the lack of sufficient control over causing interference in a specific time and frequency. An extension of this problem is to blindly detect signals of interest in presence of interference of various levels. These environment sensing capability systems (ESCs) are very common in CBRS band [6] operations and testing wireless software systems that perform such functions are difficult due to the rigidity of current platforms. In addition, the data requirements for the training and testing of RF ML models are complex and hard to obtain. This is due to the fact that data from real-time devices are without labels and labeled data are not always representative of the actual OTA data [7]. The fact that all these requirements have to be scalable makes the entire process of data generation and testing these software systems complex.

We solve this problem in two phases. In the first phase, we developed a signal generation platform, Sig-Gen, which can generate IQ of complex wireless spectrum along with metadata in a simulation environment. In the next phase, we extended this platform for performing long-duration (minutes-hours) OTA testing based on some assumptions which will be explained in later sections.

III. Sig-Gen Architecture

Keeping in mind the functional specifications mentioned in the previous sections, a Sig-Gen platform was developed and it mainly satisfies the synthetic data generation requirements. It has the following functional requirements:

- Emulate complex RF scenarios easily based on user-defined configurations.
- Ground truth metadata report of the RF IQ generated.
- Comprehensive RF signal library with diverse modulations and protocol signal.
- Generate large amounts of data easily with variations in parameters for training and testing ML models.

A. Abstraction Levels

The following abstraction levels were used in the Sig-Gen to generate the IQ for a specified RF environment scenario.

- **Source**: A radio signal emitting entity that can emit one or more signals at the same time.
- **Signal**: A set of homogeneous energies with transmissions following a specific parameter set. The homogeneous aspect here refers to the signal parameters used to generate the energies.
- **Energy** : Fundamental building block of a signal. It can be interpreted to be a unit similar to the data packet.

These notions make it easier for the end user to write input configurations and are also a representative of how real-world devices operate. An example scenario and the abstractions are shown in the Figure1

For example, a single mobile phone does not emit one specific

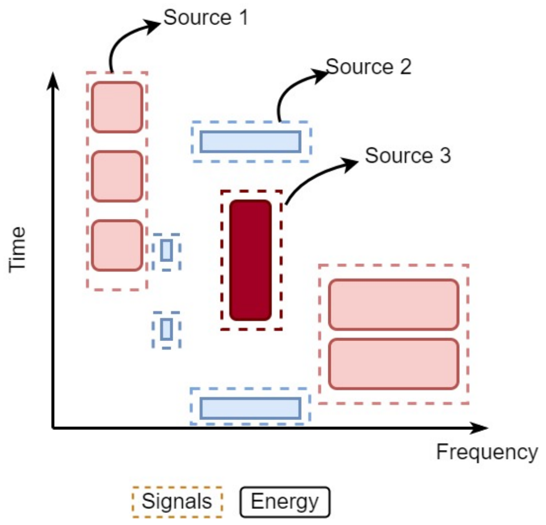


Fig. 1. Abstraction levels used in Sig-Gen. The whole set of transmission can be thought to be of from a single source and the individual energies are demarcated through solid lines and the signals comprising of energies are indicated through dashed lines.

signal. At any point, the mobile can be connected to Wi-Fi and is also attached to an LTE base-station. This means that the same device is emitting multiple signals at multiple bands at different transmission intervals. The abstractions of energies, signals, and sources is useful to characterize such a behavior.

B. Sig-Gen System generation Call flow

Generating a composite RF environment data with multiple sources is achieved by emulating multiple RF chains that contains a data source, modulator, a mixer (up-converter) and is followed by a resample operation to make the signals uniform in length. Even though [8] had developed a similar architecture, it had only focused on using a single RF chain for a single signal IQ file and did not support the placement of multiple signals across multiple bands in the same IQ. *To generate the IQ for a complete RF environment with multiple signals in a single IQ, a superposition/combining module that glues all the activities of individual sources together into a single IQ file is required.* Also, just generating the individual IQ per individual source will not be representative of what occurs in the real world. So, the hardware imperfections, and wireless channel effects are also added on top of each of the data generated before combining all the signals. Finally, Sig-Gen can be thought to have a receiver operating in a user-defined frequency capturing the signal requested by the user through the configuration.

C. RF and analog distortions

Consider the scenario where there is a single source emitting three signals at three different frequencies. The signal generation call flow in the Sig-Gen follows the procedure as shown in Figure 2. Assuming that we have extracted the necessary information from the configuration that the user has given, the Sig-Gen block will have inputs in the form that corresponds to the abstraction level as mentioned previously, i.e. in terms of energies, signals, and sources. The baseband signals are generated from the library of signals called *Atomics*, explained in the next section, and different kinds of RF imperfections are added to the signal. The RF imperfections added are IQ mismatch, frequency offset and DC offset.

Assuming that $S_k^1(n)$ is the k^{th} signal needed to be produced for the 1^{st} source as provided in the example generation flow Figure 2, the combined output in complex baseband is given by:

$$\mathrm{Y}[n] = \mathrm{S}_k^1[n] \cdot e^{-i2\pi\left(F^1_{\text{offset}}\right)t} \cdot e^{i\varphi^1_{\text{offset}}} + DC^1_{\text{offset}} \tag{1}$$

where k represents the signal index (one of the N_{sig} signals emitted by the source), F^1_{offset} is the center frequency offset and the ϕ^1_{offset} is the phase offset arising due to IQ imbalance and DC^1_{offset} is the gain added due to the DC addition in the 1^{st} source. Then, in complex baseband, we also add a per-signal channel by convolving the complex baseband signal with a channel response:

$$\mathrm{Y}[n] = \left[\mathrm{S}_k^1[n] \cdot e^{-i2\pi\left(F^1_{\text{offset}}\right)t} \cdot e^{i\varphi^1_{\text{offset}}} + DC^1_{\text{offset}}\right] * h_k^1(n) \tag{2}$$

When superposing these multiple signals and multiple sources together t form one single IQ, we first resample each signal to the same target sample rate as shown in Figure 2 and shift the center frequencies of each signal by $e^{-i2\pi F_c^l t}$. The composite signal from a single source $Y_1(n)$ is given as,

$$\mathrm{Y}_1[n] = \sum_{k=1}^{N_{\text{sig}}} \left[\mathrm{S}_k^l[n] \cdot e^{-i2\pi F_c^l t} \cdot e^{-i2\pi\left(F^l_{\text{offset}}\right)t} \cdot e^{i\varphi^l_{\text{offset}}} + DC^l_{\text{offset}}\right] * h_k^l(n) \tag{3}$$

where N$_{sig}$ is the total number of signals from a single source.The process is repeated for multiple sources and a final IQ is generated after performing superpositions across all sources. A receiver is conceived with a particular center frequency and bandwidth. The final IQ output of the Sig-Gen can be thought to be as the data that is captured by this receiver with a specified bandwidth 'BW' with a user-defined sample rate.

The output signal after all these processes can be represented as

$$\mathrm{Y}[n] = \sum_{l=1}^{N_{\text{source}}} \sum_{k=1}^{N_{\text{sig}}} \left[\mathrm{S}_k^l[n] \cdot e^{-i2\pi F_c^l t} \cdot e^{-i2\pi\left(F^l_{\text{offset}}\right)t} \cdot e^{i\varphi^l_{\text{offset}}} + DC^l_{\text{offset}}\right] * h_k^l(n) \tag{4}$$

where $h_k^l(n)$ is the configured channel for the kth signal of lth source to the receiver.

IV. Submodules in Sig-Gen

A. Atomic Library

The Sig-Gen generation module interacts with a signal library called atomic signals which consists of basic modulation signal generation classes and protocol binding signal generation classes. To construct any complex RF scenario, these atomic functions act as the fundamental building blocks. Whenever a user wants to generate data that is :

- Binding to a specific modulation, and/or
- Binding to a specific protocol

The Sig-Gen creates instances of these atomic signal classes and generates IQ data. Once the baseband modulation data is generated by using the atomic library, these signals are further processed as explained above by the Sig-Gen.

1) Signals in the Atomic Library: The atomic functions are designed to be comprehensive in terms of a variety of signals and flexible in terms of parameter space for each signal in the library. The signals in the atomic library can be broadly classified into:

- Vanilla signals which are plain modulated signals
- Protocol-adhering waveforms
- Anomalous signals

The complete list of signals in the atomic library is provided in the following Table I

TABLE I
Signals in atomic library

Vanilla Signals	**Protocol Signals**	**Anomalous Signals**
Analog: AM, FM, SSB	BLE	DS
FSK family: FSK, GFSK, CPFSK MSK, GMSK	LTE	Frequency hoppers
PSK family: PSK, DBPSK, DQPSK	WLAN 802.11	RF Emanations
QAM	5G NR	
PAM	GSM	
ASK	LoRa [9]	
APSK	Zigbee	
OFDM		

2) Traffic Types: The Traffic generation module generates the transmission start times for the energies based on some probability distributions when given a start and stop time of signals, duration of each energy, and transmissions required per second. These start times will act as the arrival instances of the energies in a particular signal.

B. Forensics framework

A persistent problem with many data generation systems is that there is no way of recording the metadata along with the data generation process. A familiar format is the sigMF format but sigMF is more suitable for captures instead of the data generation process itself. So, a forensic framework that probes the signal at certain important points along the generation pipeline is put in place. The IQ data that is generated is dumped in the form of ".32cf", a 32-bit complex float, and a "JSON" metadata file is generated along with it. An example metadata format is shown in Figure 3.

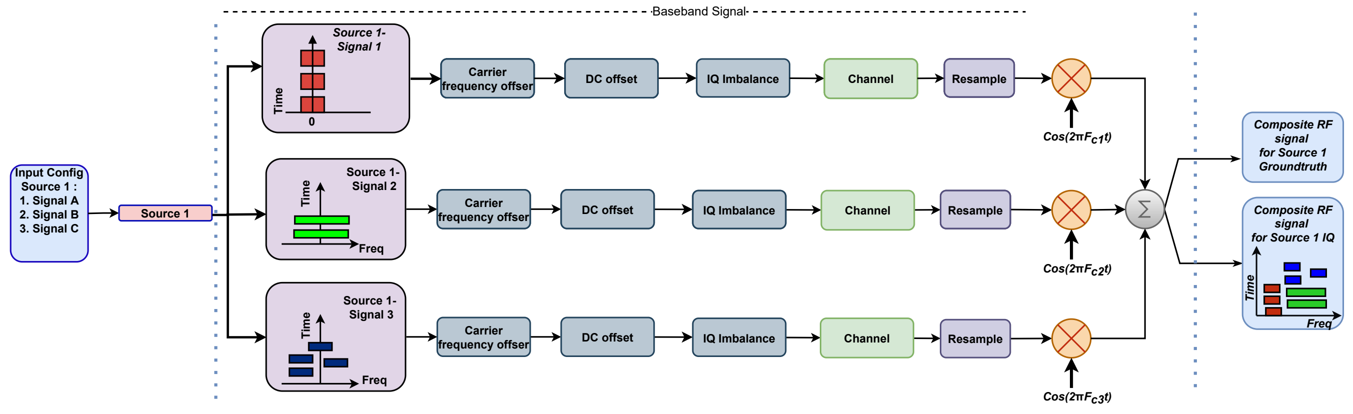


Fig. 2. Sig-Gen IQ generation flow: An example case of Sig-Gen call flow is shown where a user wants to generate RF IQ for a scenario with a single source emitting 3 signals centered around three different center frequencies. The resulting final IQ contains all these signals at user-specified times and frequencies.

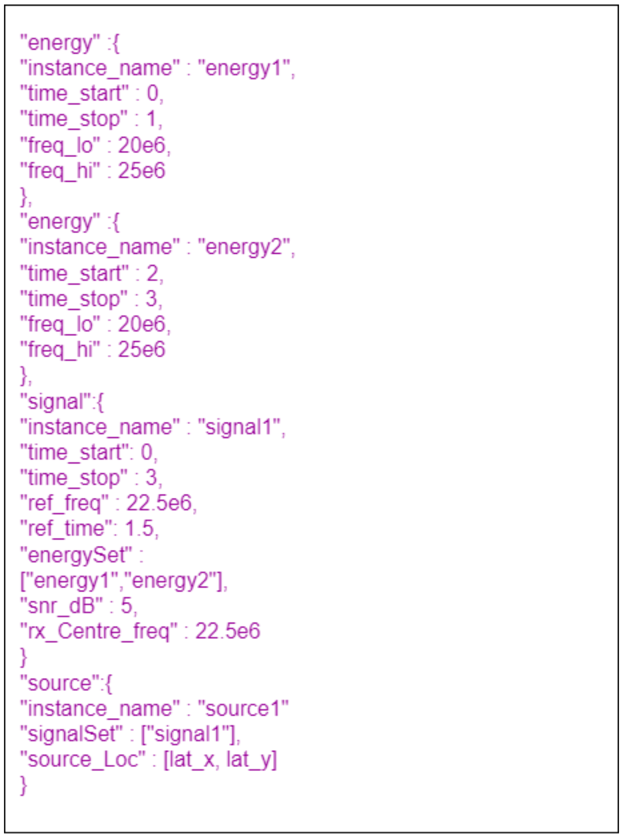
To enable this flow, a reporting mechanism is created. The mechanism is flexible enough to add new entries to the metadata field and can be plugged into any part of the signal generation process to enable the addition of any new parameter about that process for future reference or debugging purposes.

The mentioned metadata in Figure 3 is surface-level metadata. However, in certain scenarios, users would need in-depth metadata. To achieve this feature, the system also outputs the entire parameter structure that has all the parameters that were used to generate the particular signal. This way, we solve the issue of metadata at two levels, first by providing a coarse level metadata through the ".json" file which indicates the time-frequency location bounds of the signal, and fine-tuned metadata through ".mat" files for providing the modulation or protocol-specific parameters.

V. Over the air Testing and Data generation

In the previous sections, the necessity of the signal generation platform and the call flow of the signal generation were presented in detail. In the effort to scale the system further and enable real-time transmissions, Over-the-air testing is seen as the next obvious step. To achieve this, a complicated way is to generate the data through Sig-Gen and then port the data to a place where the radios are connected, and then set up the environment for transmission. This is extremely cumbersome and is a logistical trouble that the user has to undergo. To alleviate these, a unified platform that can generate both simulated data and real-time OTA data in an automated manner is conceived. The overall functional requirements that we tend to fulfill are as follows:

- Ability to transmit the signals over the air.
- Automatically queue the signals/energies to be transmitted with the appropriate radios.
- Ability to transmit for long hours.
- Have proper metadata for the OTA signals transmitted.



```
"energy" :{
"instance_name" : "energy1",
"time_start" : 0,
"time_stop" : 1,
"freq_lo" : 20e6,
"freq_hi" : 25e6
},
"energy" :{
"instance_name" : "energy2",
"time_start" : 2,
"time_stop" : 3,
"freq_lo" : 20e6,
"freq_hi" : 25e6
},
"signal":{
"instance_name" : "signal1",
"time_start": 0,
"time_stop" : 3,
"ref_freq" : 22.5e6,
"ref_time": 1.5,
"energySet" :
["energy1","energy2"],
"snr_dB" : 5,
"rx_Centre_freq" : 22.5e6
}
"source":{
"instance_name" : "source1"
"signalSet" : ["signal1"],
"source_Loc" : [lat_x, lat_y]
}
```

Fig. 3. Metadata that is generated from the Sig-Gen in JSON format. The metadata consists of all three abstraction levels specified. The energy level data with its Time-Frequency bounds are indicated by the label "energy" and each energy is provided with an instance name, here "Energy1" and "Energy2". These energies cumulatively form a signal, here "Signal1" and the the source that emits this particular "signal1" is indicated through the tag "Source", here "source1"

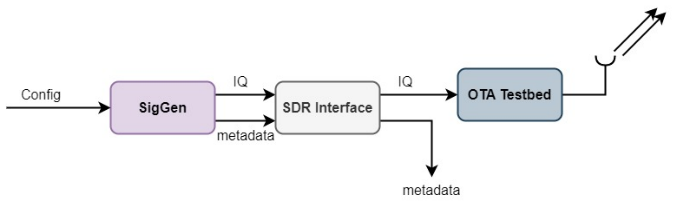


Fig. 4. RF Synth architecture: An SDR interface that can relay the IQ generated through the Sig-Gen along with metadata.

VI. RFSynth : System Architecture & design choices

The proposed RFSynth platform has a software-defined radio interface and it would act as a linking module between the Sig-Gen and the SDRs as shown in Figure4.

A. Extending Sig-Gen

To enable over-the-air transmissions, the Sig-Gen data path was modified in some places. Although the core aspects of the Sig-Gen remains the same,these modifications include,

1) Since we are already transmitting the data through the hardware and channel, the additional hardware imperfections and channel application that were done previously were removed.
2) The superposition module is also bypassed to create 'N' IQ files, where 'N' is the total number of signals one wants for a particular experiment. This is done to queue the signals as it is and send them over the air. Thus, any RF scenario that the user wants is obtained once these IQs are transmitted.
3) One of the major reasons for developing the system is to enable long hours of spectrum activity in a particular band. To enable this, many hours' worth of data cannot be produced in one shot before shipping the data for transmission to the radios due to limits in computing and memory. To overcome this, the system operates on the assumption that all the energies in the signal contain the same payload. This makes data generation a lot easier as only a small period worth of samples need to be generated and stored but it can be used for long hours of transmission. In other words, the same payload of the requested modulation or protocol signal is transmitted for long hours.
   An example comparison showing the memory consumption when the test data vector is generated without using the assumption and after using the assumption is given in the following table II.

TABLE II
Example memory usage computation with and without assumption on payload

Attributes	Without using the assumption of same payload	After the assumption of same payload
Total time of IQ to be generated	2000 (secs)	2000 (secs)
Sample Rate	100 MHz	100 MHz
Total samples to be generated	200 Giga samples	200 Giga Samples
Total file size	1600 GB	80 MB* (Assume energy duration of 100 ms)

B. Sources to radio mapping

To map the signals that are created from the configuration file automatically to the transmission radios that are set up, a source allocation algorithm is also provided as a part of the call flow. This reduces the user's effort to run tests as the user no longer needs to queue the signal manually. This becomes especially complex when the testing is done for spectrum sensing applications because of the involvement of multiple radios and multiple signal transmissions. The steps followed in the radio allocation are as follows :

1) Find the maximum number of simultaneous signals present.
2) Assert if the number of simultaneous signals is greater than the number of transmitter resources available.
3) Sort the signals according to their start times.
4) Setup a radio availability buffer that holds the earliest available times of all the radio resources and sets all its entries to be zero for the first iteration, indicating all the radios are available at from time 0 secs for transmission. Label the radios from 1 through N, N being the number of available radios.
5) Assign each signal to a radio based on the earliest availability of any of the N radios and update the resource buffer with the end time corresponding to that particular signal.
6) Assert if the start time of the next signal to be transmitted is lesser than the maximum of the end times in the buffer, indicating that an assignment cannot be made for a signal.
7) Continue the process from step 5 to step 6 until all signals are assigned.

Once the signals to transmit radio mapping are done, a schedule file of *".csv"* format is created which contains the following :

1) Location of energy IQ to be transmitted
2) Tx radio in which the signal is to be transmitted
3) Time in which the energy needs to be transmitted
4) frequency in which the energy needs to be transmitted

Once the test is triggered, first the IQ samples are generated using the configuration file which is provided to the Sig-Gen by the user and the SDR interface will loop through this schedule file and will transmit the data according to the frequency and time information as mentioned in the schedule file. This way, the whole process of transmitting signals over the air is automated end to end.

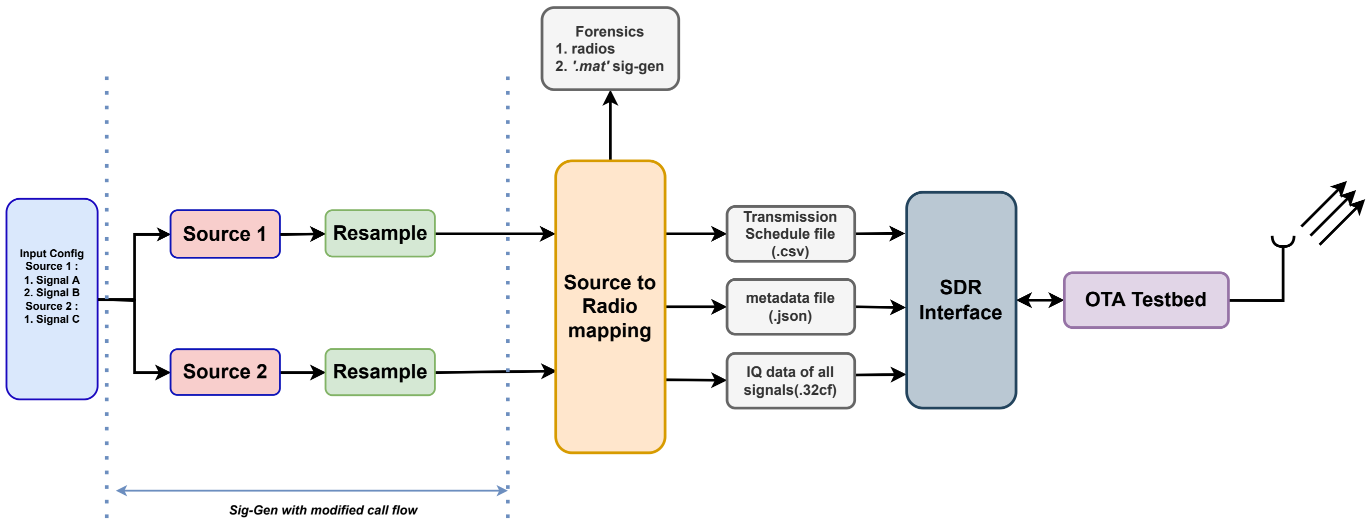


Fig. 5. RFSynth signal and metadata generation flow: As one can notice, the hardware imperfections and channel additions are removed. The source of radio mapping outputs forensic information about radios and interacts with the SDR interface for scheduling the transmission of the signals using SDRs.

C. Metadata data propagation

In the Sig-Gen, since over-the-air transmissions are not taking place, the generation of metadata is fairly straightforward once the signals to be generated are put into sources. However, when the same operation is ported to Over-the-air mode, the time at which the signals leave the antenna is difficult to predict.

To solve the problem of identifying the moment at which the signal leaves the antenna, the metadata is propagated to the SDR interface. The system then synchronizes the metadata to the POSIX time of the system when the energy was transmitted as the average precision of the POSIX clock is around 1 ms to 1 μs. At the end of transmission, a count of the number of energies to be transmitted is displayed along with metadata. This metadata will be identical in terms of all other contents except the time start and time stop metric whose values will extracted from the POSIX clock. This entire generation and transmission process is detailed in Figure 5.

Thus, combining the assumption on payloads of energy and automatic signal to transmitter radio mapping, data generation, and testing for long hours is enabled with easy data augmentation.

VII. Implementation

The data-generation system described in this work is implemented three modular parts to allow simulated and OTA data-generation, while allowing the framework to be extended with new signal types. The first part is the Sig-Gen software, that allows configurable simulated data generation with accurate metadata. Second, the SDR interface that extends Sig-Gen into RFSynth. And finally, the SDR hardware platform that performs the actual transmission.

Sig-Gen is implemented in MATLAB with a framework to extend the library with new signals. The parts of the system described in Figure 2 are implemented as self-describing classes, i.e., they can be serialized into metadata when needed. This allows abstractions, non-ideal transforms (like CFO), and data-generation parameters to be easily codified into metadata for ground to labels and forensics.

SDR Interface: For ease of use and repeatability, we use the GNURadio [10] and the UHD drivers [11] to implement the SDR interface. The interface uses artifacts generated by the Sig-Gen to control SDRs and stream data to them in a tightly time-controlled manner. Transmission time of signals and energies are *exactly* controlled using timed commands that instruct the FPGA to start and stop streams at nanosecond precision [12]. For performance reasons, the interface uses a separate thread for each SDR.

OTA Testbed is the SDR platform used to transmit generated signals. In our experiments, we use 6 USRP N320 radios attached to a server through a 100G switch. Each radio has an instantaneous bandwidth and sample rate of 100 MHz. Each transmitter is bound to the SDR interface and transmits only one signal at a time, simultaneous transmissions requires multiple radios. This platform is similar to other USRP-based testbeds like POWDER, COSMOS, and Colosseum [13]. As such, RFSynth can be deployed as a service on any of these testbeds as they provide the necessary software drivers and tools.

VIII. Experimental Scenarios

To qualitatively evaluate the advantages of the RFSynth platform, this section discusses four case studies in which the platform was used extensively. The following case studies exemplify the flexibility the platform offers.

A. Case Study 1: Recreating RF environments

A simple case study to show the capability of the system was undertaken and it was to closely recreate a spectrum band

TABLE III
PARAMETERS USED FOR COLLECTION AND RE-CREATING 2450 MHZ SPECTRUM

Parameter	Collect Param Values	Recreate Param Values
Centre Frequency	2450 MHz	baseband
Sampling rate	100 MHz	100 MHz
Capture condition	Indoor environment	Rayleigh
Hardware used	SM200C Signal hound	NA

using the RFSynth platform which was observed by the user. The main application of the feature is that it would be easier to do a quick check functionality check of the system in the recreated band than using hardware every time. This is mainly aided by the metadata that comes along with the data IQ.

To showcase this feature, a capture of the 2450 MHz spectrum was made using the signal hound SM200C [14] as shown in 6. The 2450 MHz is a heavy activity band with multiple bands of Wi-Fi, BLE, and Zigbee Co-existing in the band. It is a good band to test algorithms that tackle multi-protocol co-existence and interference.Figure 7 shows the recreated version of the spectrum using Sig-Gen alone. It can be inferred that it was possible to recreate the traffic of Wifi OFDM and Wifi Direct spread spectrum very closely along with the LO leakage. All the features including the Wi-fi OFDM, DSSS, and BLE are exactly matched with their corresponding frequencies.

B. Case Study 2: Energy detection algorithm design

The problem of detecting the presence of RF signals scales in complexity if the energy detection has to be performed blindly because of system throughput bottleneck. An interesting use of Sig-Gen and RFSynth has been in the design of blind energy detection system which was recently proposed, searchlight [15]. The main functionality of the detector is to identify the center frequency of the energy and reference time, which is the average of start and end times of the energy, and draw a bounding box. To test the sensitivity of the detector's performance when two energies are close to one another, RFSynth's agile capability of placing energies in time and frequency grid with ease was used. An example of a spectrum designed to test the performance of [15] and the output with bounding boxes are also shown in Figure 8.

C. Case study 3: RF ML dataset generation (Blind Modulation classification problem)

Providing context to the raw RF IQ data captured within a band of the spectrum requires the characterization of the signals in the desired bands. To blindly characterize signals, automatic modulation classification algorithms are run and the decision is fed back to the main system based on which the subsequent decisions are taken. Classifying the signals which are blindly captured based on modulation is a difficult problem and continuous efforts have been going on in that front. Approaches for this purpose include the use of wavelet analysis [16], cyclostationary analysis of signals [17], and also RF machine learning [18]. To train and test the machine learning models for modulation classification, a variety of data sets have also been captured and are widely used. Some examples of such datasets include RFMLS datasets, RF fingerprinting data set by [19] and Radio ML 2016.04C, 2018.1A by [20]. The following section does a comparative study of the parameter space that was varied in radio ML datasets and the flexibility that the Sig-Gen and RFSynth platforms offered to its users for generating data at scale.

Comparison of parameter space between common RF ML datasets and RFSynth-based data set generation: The prevalent dataset used for benchmarking the performance of machine learning models in automatic modulation classification is RadioML datasets. For the comparison, the radioML2016.04c was chosen and the comprehensiveness of the parameter space of the dataset is compared with the parameter space capabilities of the Sig-Gen.

A surface-level comparison was made first by seeing the different types of modulations which is supported in radioML2016.04c and the signals present in the atomic library. Further on, the number of SNR data points which has been recorded for each modulation is compared against the framework's flexibility in IV.

TABLE IV
COMPARISON BETWEEN RML 2016.04C.MULTISNR WITH RFSYNTH.
* - INCLUDES PROTOCOL SIGNALS ALSO.

Parameter	**RMLdataset**	**RFSynth capability**
Modulations	11	27*
Protocols	Not Present	Support for LTE, 5G NR, BLE, LoRA, Zigbee, GSM
SNR points	-20:18 (Not all modulations have the same points)	Arbitrarily configurable

D. Case study 4: End to End testing for spectrum information processing systems

The wireless spectrum is vast and is filled with activities by countless devices. As a result, the wireless spectrum is susceptible to various types of interference and unauthorized access, posing security threats to legitimate users and critical communication systems. To address these challenges, security measures incorporating spectrum sensing algorithms have emerged. Testing these systems is not trivial as we have to keep moving across bands, creating spectrum activities. The testing requirements for such a system are as follows :

1) Simultaneous transmissions at different bands of frequencies.
2) Testing for long hours
3) Ability to generate agile LPI signals like snugglers.
4) Provide metadata for transmissions.

The easy modular setup of Sig-Gen coupled with RFSynth was used effectively with little modification for queuing multiple generation configurations one after the other automatically at different bands of the spectrum. As mentioned in the energy

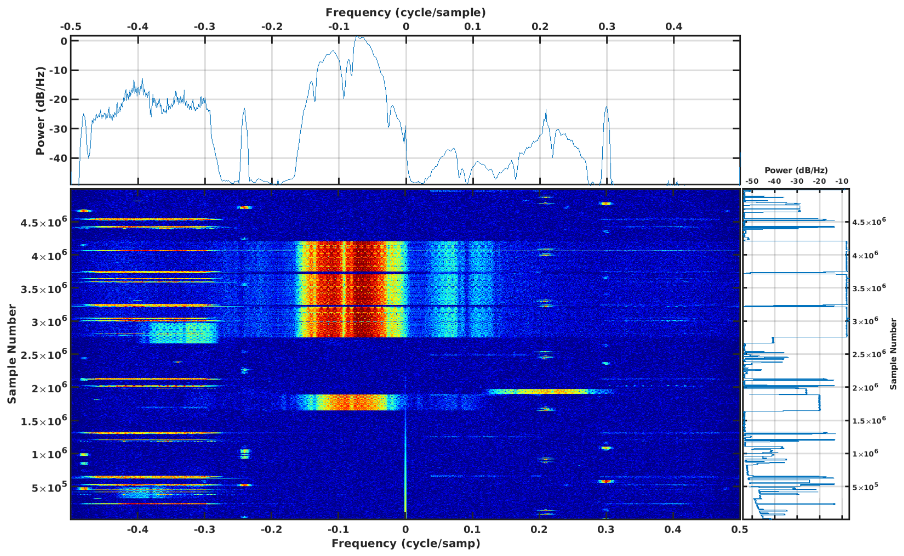


Fig. 6. The activity around centre frequency of 2.4 GHz spectrum captured through a USRP N320 radio is shown in this spectrogram. The band contains multiple signals including DSSS / OFDM Wi-Fi along BLE and Zigbee.

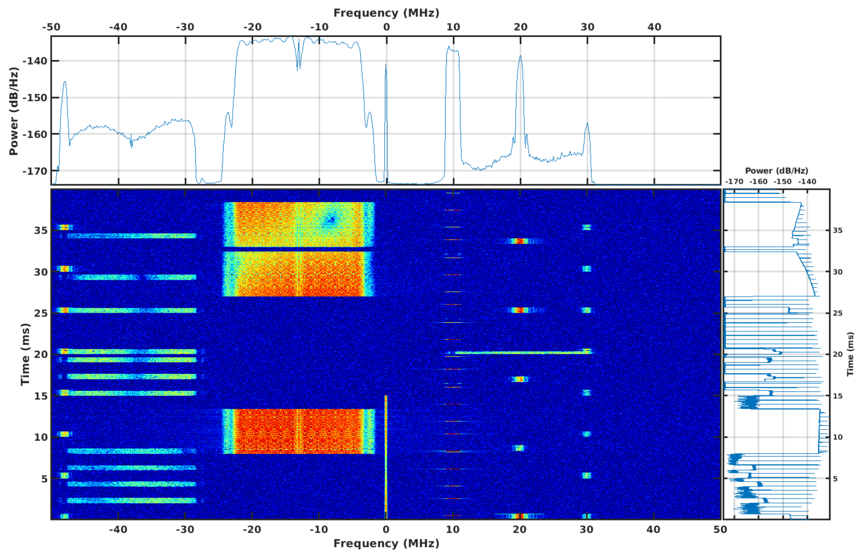


Fig. 7. The activity in 2.4 GHz band recreated using Sig-Gen platform. One can notice similarity spectrum activity across frequencies when compared to Figure 6 except for the image frequencies of the Wi-Fi. This is a baseband representation of the Figure 6.

detection case study, the ability to place signals at any time-frequency position was leveraged to create abstract signals like snugglers and transmit them. One such example is provided in the Figure 9 where anomalous snuggler type signals were created in 800 MHZ - 900 MHz LTE band and the snuggler tries to move along the LTE signal with it in the same band. The spectrum information system was tested with this data vector to see if it could identify these transmissions quickly. An example metadata for one of the tests performed using the RFSynth platform with time synchronization achieved between Tx and Rx machines using Posix time is provided below.

1) **Energy level metadata example:**

```
{
"report_type": "energy",
"instance_name": "bBiaaUMmS",
"time_start": 1684952693.346253,
"time_stop": 1684952693.348253,
"freq_lo": 2481.875,
"freq_hi": 2482.125,
"timeLength_s": 0.00200,
```

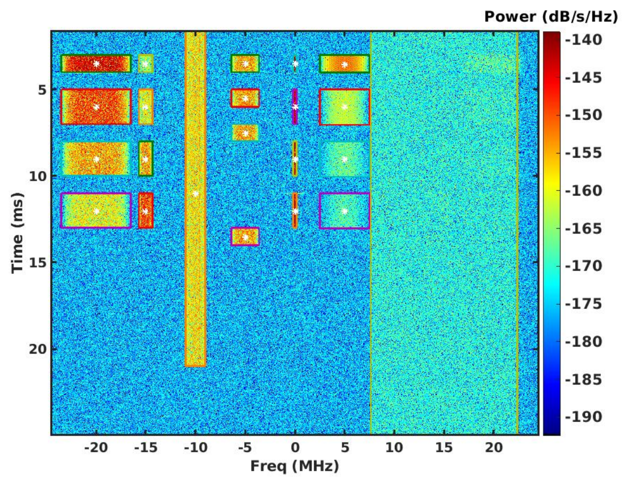


Fig. 8. Spectrogram of a baseband test vector with bounding boxes after being processed using SearchLight [15]. The center frequencies of energies detected using SearchLight are marked using white dots.

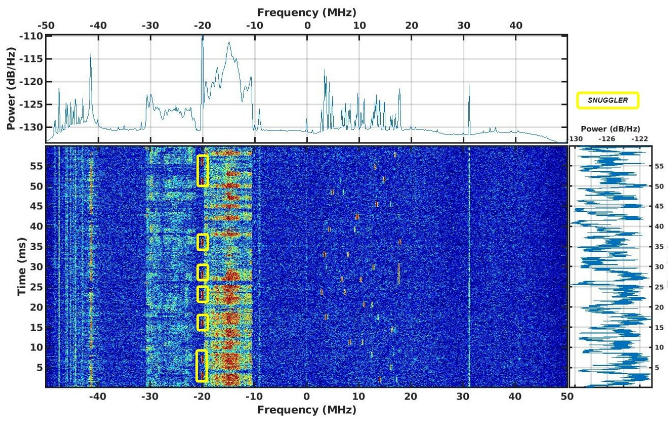


Fig. 9. Spectrogram shows an IQ capture centered around 850 MHz shwing an Over the Air snuggler type anomalous signal transmission using RFSynth. The snuggler signal marked in yellow tags alongside a common LTE activity.

```
    "bandwidth_Hz": 0.25
    }
```

2) **signal level metadata example**

```
        {
        "report_type": "signal",
        "instance_name": "bBiaaUMm",
        "protocol": "unknown",
        "modality": "single_carrier",
        "modulation": "qam16",
        "activity_type": "overt_new",
        "time_start": 1684952693,
        "time_stop": 1684952803,
        "freq_lo": 2481.875,
        "freq_hi": 2482.125,
        "rx_center_freq": {
            "rx1": 2.482E+9
        },
        "reference_time": 65.001,
        "reference_freq": 2482,
        "timeLength_s": 110.002,
        "bandwidth_Hz": 0.25
        }
```

It also contains the set of energies which are part of the signal but considering space constraints, it is omitted.

3) **source level metadata**

```
        {
        "report_type": "source",
        "device_origin": "default",
        "instance_name": "default0",
        "tx": [],
        "locationXYZ_m": [0,0,0],
        "outputSamplingRate_Hz": 1.0E+8,
        "channelModel": "IDENTITY",
        "nSignal": 1,
        "signal_set": [
        "bBiaaUMmSQmgIfLw0e"
        ]
        }
```

This case study showcases a simple example of how RFSynth can be leveraged to test spectrum sensing systems which detect anomalous RF activities. However, more complicated testing such as interference and co-existence testing can also be conducted very easily using the mechanism established in this section.

IX. Future Work

The development of Sig-Gen first and from Sig-Gen to RFSynth was a natural progression as the use case/necessity presented itself. After extensive use of the system, we found that there are certain design advancements that can be make this system more robust. The design choice of having the same payload for all energies of the signal posed bottlenecks when we wanted to generate test cases for the detection of RF Emanations which generally occur with different payloads and are always-on. This in addition to the fact that the system operates by generating all the required IQ first before initiating the transmissions increases the latency and storage. If this can be changed to an IQ streaming mechanism (procedural generation and transmission), then it can support signals that are always-on and can make the spectrum activity generation even more realistic.

X. Conclusion

In this paper, we introduce RFSynth, a comprehensive system designed to address gaps in data generation and testing platforms within wireless systems. Together with Sig-Gen, RFSynth aims to provide a robust solution to the challenges currently faced in this domain.

Specifically, our focus was on resolving the lack of a comprehensive data generation platform that integrates various signal libraries. We propose a method for generating data over-the-air (OTA) and synchronizing metadata using the RFSynth interface.

To evaluate the efficacy of our proposed solution, we conducted qualitative case studies. The results of these evaluations demonstrate the effectiveness and practical applicability of the RFSynth system in addressing the challenges encountered in wireless systems testing and data generation.

XI. Acknowledgements

This paper is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via [2021-2106240007]. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

[1] William H. Clark IV. *The Importance of Data in RF Machine Learning*. PhD thesis, Virginia Polytechnic Institute and State University, 2022.

[2] RF Data Factory. https://www.rfdatafactory.com. Last accessed on 10 Mar 2023.

[3] Signal Metadata Format (SigMF) Development Team. SigMF. https://github.com/sigmf/SigMF. Last accessed on 10 Mar 2023.

[4] keysight pathwave signal generator. https://www.keysight.com/us/en/assets/7018-01538/brochures/5989-6448.pdf. Accessed: 2023-05-21.

[5] Rohde and schwarz winiqsim2 system. https://www.rohde-schwarz.com/webhelp/RS_WinIQSIM2_Help/Content/welcome.htm. Accessed: 2023-05-21.

[6] Federal Communications Commission. ”FCC 18-149: Report and Order In the Matter of Promoting Investment in the 3550-3700 MHz Band”. https://docs.fcc.gov/public/attachments/FCC-18-149A1.pdf.

[7] Robert D. Miller, Silvija Kokalj-Filipovic, Garrett Vanhoy, and Joshua Morman. Policy based synthesis: Data generation and augmentation methods for rf machine learning. In *2019 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 1–5, 2019.

[8] Timothy James O'Shea, Tamoghna Roy, and T. Charles Clancy. Over-the-air deep learning based radio signal classification. *IEEE Journal of Selected Topics in Signal Processing*, 12(1):168–179, 2018.

[9] Zhenqiang Xu, Shuai Tong, Pengjin Xie, and Jiliang Wang. From demodulation to decoding: Toward complete lora phy understanding and implementation. *ACM Trans. Sen. Netw.*, 18(4), jan 2023.

[10] GNU Radio Website, accessed March 2024.

[11] Ettus Knowledge Base. Uhd and usrp user manual — ettus knowledge base,, 2016. [Online; accessed 22-June-2018].

[12] Tomoya Nakahama, Yoji Yamada, and Suguru Kameda. Example gnu radio implementations of phase alignment between usrp devices. *IEICE Technical Report; IEICE Tech. Rep.*, 120(238):74–81, 2020.

[13] Leonardo Bonati, Pedram Johari, Michele Polese, Salvatore D'Oro, Subhramoy Mohanti, Miead Tehrani-Moayyed, Davide Villa, Shweta Shrivastava, Chinenye Tassie, Kurt Yoder, et al. Colosseum: Large-scale wireless experimentation through hardware-in-the-loop network emulation. In *2021 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, pages 105–113. IEEE, 2021.

[14] Signal Hound. ”SM200C — 20 GHz Real-time Spectrum Analyzer with 10GbE”. https://signalhound.com/products/sm200c-20-ghz-real-time-spectrum-analyzer-with-10gbe/.

[15] Richard Bell, Kyle Watson, Tianyi Hu, Isamu Poy, fred harris, and Dinesh Bharadia. Searchlight: An accurate, sensitive, and fast radio frequency energy detection system. In *MILCOM 2023-2023 IEEE Military Communications Conference (MILCOM)*. IEEE, 2023.

[16] Cheol-Sun Park, Jun-Ho Choi, Sun-Phil Nah, Won Jang, and Dae Young Kim. Automatic modulation recognition of digital signals using wavelet features and svm. In *2008 10th International Conference on Advanced Communication Technology*, volume 1, pages 387–390, 2008.

[17] Chad M. Spooner, Apurva N. Mody, Jack Chuang, and Josh Petersen. Modulation recognition using second- and higher-order cyclostationarity. In *2017 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, pages 1–3, 2017.

[18] Timothy J O'Shea, Johnathan Corgan, and T. Charles Clancy. Convolutional radio modulation recognition networks, 2016.

[19] Samer Hanna, Samurdhi Karunaratne, and Danijela Cabric. Open set wireless transmitter authorization: Deep learning approaches and dataset considerations. *IEEE Transactions on Cognitive Communications and Networking*, 7(1):59–72, 2021.

[20] Deepsig dataset. https://www.deepsig.ai/datasets. Accessed: 2023-05-21.