

Satellite Signal Detection Design on Spartan3 FPGA

Mr.Saishankar¹, Dr.Pradeep Reddy²

¹Dept. of Electronics and Communication Engineering (VLSI), VTU's CPGS, Kalaburagi, Karnataka – India

Email: skpmakapnoor@gmail.com

²Dept. of Electronics and Communication Engineering (VLSI), VTU's CPGS, Kalaburagi, Karnataka – India

Email: pradeepmr056@gmail.com

ABSTRACT

Satellite signal detection is a significant task in satellite communication as well as navigation systems including GPS. Weak satellite signals in the presence of noise need signal processing techniques and computational methods to detect them efficiently fast. In this project, we are presenting the design and simulation of a satellite signal detection system using Python on Spartan-3 architecture without using hardware. This will be done by simulating the detection process through modeling. This involves using Python-based signal processing libraries (e.g. NumPy, SciPy and Matplotlib) to simulate satellite signals, noise environments and detection algorithms in a complete system. The simulated prototype executes important functions such as signal generation, noise filtering, correlation analysis, and threshold-based determination in order to detect satellite signals. You can use the processing steps to simulate the functional behavior of modules that you would typically implement on a Spartan-3 FPGA to enable a shake-down of your design prior to implementation onto hardware. This behavior in the simulated Python framework suggests that our method can detect satellite signals and capture their features even with noise present. Visualization tools assist in exploring with signal patterns, correlation peaks, and detection accuracy. In this context, our software-based system minimizes the complexity and cost of development, creating an environment to develop and improve satellite signal detection strategies before transferring them to their field deployable FPGA platforms.

Keywords— Satellite, Spartan-3 FPGA, NumPy, SciPy, Matplotlib, Signal

I. INTRODUCTION

Global positioning, weather monitoring, remote sensing and wireless communication systems are some modern technology applications that strongly require satellite communication and navigation systems. These systems depend on signals sent from satellites in orbit around the Earth. Because satellite signals on the ground are generally weak and corrupted by noise, interference, and atmospheric effects, accurate detection requires delicate processing techniques. Hence, we need efficient signal detection techniques for reliable identification and processing of such signals.

In most approaches, the satellite signal detection is carried out by means of hardware platforms like as Field Programmable Gate Arrays (FPGAs) that guarantee high-speed processing and allow parallel architecture. Another popular FPGA is the Spartan-3, which is often used in digital signal processing applications due to its low power consumption, flexibility and capability to perform real-time computations. Multiple Signal Processing Operations can be concurrent (Filter, correlation, thresholding detection) on FPGA.

This project uses Python to design and simulate the concept of a satellite signal detection system, but Pyrenkt is not physically integrated as FPGA hardware. Libraries like NumPy, SciPy, and Matplotlib available with Python language are very powerful in generating signals, simulating noise, filtering them or visualizing the signal. The Python model was designed to emulate the functional blocks that we intend on mapping onto a Spartan-3 FPGA, allowing developers to simulate the algorithm prior to loading it on chip.

The approach describes satellite signals simulation, some noise addition and the application of signal processing techniques to correlate and threshold a detection with or without resulting signals. This means that the system can be tested, optimized, and easily altered in a software environment. This method also minimizes the cost and time invested in development as well as ensures that the detection algorithm runs efficiently before it is implemented on an FPGA platform.

This project showcases the capabilities of Python-based simulation for modeling and analyzing satellite signal detection systems specifically tailored for Spartan-3 FPGA architectures, offering versatility and efficiency in exploring various aspects of satellite communication and signal processing.

II. LITERATURE SURVEY

El Hawary, Hamza, Zekry & Motawie – [1] This paper involves the design and FPGA realization of parallel search acquisition phase of GPS receiver using MATLAB/Simulink and Xilinx System Generator (XSG) The work shows combination of simulation and prototyping on same platform to make design and debugging in complex GNSS receiver parts easier in FPGA.

Shapiro – [2], 2010: Implementation of a real-time FPGA-based GNSS receiver with early modules for acquisition and tracking coded in Verilog targeting a Xilinx/Altera FPGA Validation with a MATLAB emulator demonstrated that hardware correlators can be implemented to achieve real-time detection and parallel tracking of multiple satellites.

[3] IIT Madras Thesis (anon. author), 2016: Searching for GPS L1 signal acquisition in A FPGA based Prototyper of Emerging Technologies-From SDR MATLAB models to HDL Coder. It covers fixed-point conversion, resource optimization, and FPGA verification using the so-called FPGA-in-Loop co-simulation.

[4] Anonymous, Averaging correlation method on FPGAs: The authors present an FPGA implementation and investigation of averaging correlation methods used for GPS signal acquisition which is important to ensure detection performance under noisy conditions.

[5] T. Qiu – [5], in order to improve the acquisition performance, but also to reduce the logic complexity (that is, apply on low-complexity and multi-constellation compatible) proposes a FPGA-based GNSS acquisition method; Emphasizes new architectural techniques to handle various GNSS signals on FPGA.

Signal Processing in Enhanced GNSS (Chetna Devi Kh.) – [6], 2025: This paper proposes a framework utilizing FPGA & SoC that could improve GNSS signals further in acquisition and tracking. It showcases simultaneous high-rate baseband processing for identification and data communication more accurate by the setting of dual board system and for real-time processing tasks in resource-constrained space environments.

T. Human-GNSS Systems – [7], 2015: Considers an FPGA-based GPS receiver used for nanosatellite applications, with emphasis on how fitting and hardware are beneficial to onboard GNSS acquisition.

GNSS SDR Survey – [8], 2016: Conducts a broad review of the design of software-defined GNSS receivers, including details about acquisition algorithms and multichannel correlators; while it takes a software perspective, also identifies the hard-reality raised from real-time correlators, combining this with improvement in terms of software/hardware integration on acquisition latency.

GNSS Acquisition Targeting MATLAB/HDL Coder (thesis output) – [9], 2016: Similar to the IIT Open-Source thesis, this master's thesis showcases the algorithm modeling in MATLAB and Simulink for GNSS receivers with FPGA prototype debugged against SDR models in a Sign-off verification loop using HDL Coder of MATLAB.

Parallel Hardware GNSS Acquisition Architectures (general research context) – [10], multiple years: Several research works investigate potential solutions, mainly focusing on their suggested Time Parallel Search Correlators, FFT-based techniques, and Hardware Accelerators for efficient GNSS acquisition capable of real-time processing (composite content).

Table 1: Related work comparison table

Author(s)	Year	Description	Methodology	Accuracy / Outcome	Limitations
El Hawary, Hamza, Zekry & Motawie	2019	Designed and implemented the parallel search acquisition stage of a GPS receiver using FPGA with MATLAB integration.	MATLAB/Simulink modeling with Xilinx System Generator (XSG) for FPGA implementation.	Demonstrated efficient integration between simulation and hardware prototyping with improved acquisition speed.	Focused mainly on acquisition phase; limited discussion on full receiver pipeline and real-world hardware constraints.
Shapiro	2010	Developed an FPGA-based real-time GPS receiver capable of performing acquisition and tracking.	Verilog implementation on Xilinx/Altera FPGA, validated with MATLAB emulator.	Achieved real-time detection and parallel satellite tracking using hardware correlators.	Hardware complexity and resource utilization increase with the number of tracked satellites.

IIT Madras Thesis (Anonymous)	2016	Investigated FPGA prototyping of GPS L1 signal acquisition using MATLAB-based SDR models.	MATLAB/Simulink models converted using HDL Coder, fixed-point optimization and FPGA-in-the-loop simulation.	Demonstrated efficient FPGA verification and reduced design time for GNSS receivers.	Conversion from floating-point to fixed-point requires careful optimization and may affect precision.
Averaging Correlation Method on FPGAs (Anonymous)	Not Specified	Focuses on improving GPS signal detection reliability in noisy environments.	FPGA implementation of averaging correlation techniques for signal acquisition.	Improved signal detection performance under low SNR conditions.	Higher computational load and possible increase in FPGA resource usage.
T. Qiu	2021	Proposed a low-complexity FPGA-based multi-constellation GNSS acquisition scheme.	Optimized hardware architecture supporting multiple GNSS signals.	Improved acquisition efficiency with reduced logic complexity.	Limited evaluation across different GNSS environments and signal conditions.
Chetna Devi Kh.	2025	Presented enhanced GNSS signal processing using FPGA and SoC architecture.	Parallel baseband processing implemented on FPGA/SoC platforms.	Achieved higher processing speed and improved acquisition accuracy.	Implementation complexity and dependency on high-performance hardware platforms.
T. Thai	2015	Designed an FPGA-based GPS receiver for nanosatellite applications.	FPGA-based acquisition and tracking modules optimized for satellite systems.	Demonstrated reliable GNSS processing for space environments.	Limited FPGA resources and power constraints in nanosatellite systems.
GNSS SDR Survey	2016	Comprehensive review of software-defined GNSS receivers and acquisition algorithms.	Comparative study of SDR architectures and hardware correlators.	Shows hardware accelerators significantly reduce acquisition latency.	Mostly theoretical review; lacks detailed FPGA implementation experiments.
MATLAB/HDL Coder GNSS Acquisition	2016	Demonstrates workflow from MATLAB GNSS algorithm design to FPGA hardware prototype.	MATLAB/Simulink modeling with HDL Coder for hardware conversion.	Enables rapid prototyping and verification of GNSS acquisition algorithms.	Hardware performance depends on FPGA resources and optimization level.
Parallel Hardware GNSS Architectures (General Research)	Various Years	Research emphasizes parallel correlator and FFT-based GNSS	Hardware acceleration using parallel processing on FPGA platforms.	Enables real-time GNSS acquisition with faster processing speed.	Increased FPGA resource usage and design complexity for large-scale

		acquisition architectures.			systems.
--	--	----------------------------	--	--	----------

III. PROPOSED METHODOLOGY

The proposed system includes the design of Satellite signal detection system and its simulation using python without real FPGA. Functional behavior of Spartan-3 FPGA-based signal processing architecture is modeled using a software-based environment. One can thus enable the researchers and developers to quickly experiment, test and evaluate on simulation level satellite signals detection algorithm prior to its deployment on a real hardware setup.

In the proposed approach, signals are generated on Python in a similar way as for satellites and go through a series of digital signal process techniques. To make the process realistic, the system adds noise and interference to the generated signals. This is useful for studying the performance of weak satellite signals in a noisy environment and detection accuracy.

This system carried out several functions such as signal generation, noise simulation, filtering action, correlation analysis and threshold-based detection. These techniques are used to filter noise and improve the quality of the signal. As you know, TFT: Time-Frequency Transform for pattern matching and detection works can do detection of processed picogram to femtogram level.

Python also has libraries (for Example NumPy, SciPy, Matplotlib) for signal processing, mathematical computations and visualization. Using these tools, the system can provide graphical representation of signal waveforms, correlation peaks and detection results. The signal behavior and the performance of the detection algorithm are better perceived by researchers when visualized.

The software simulation proposed in this paper is a flexible framework that allows the rapid adjustment and verification of various signal processing methods and settings. Thus making it less complex, cheap and less time consuming than hardware based development while confirming working algorithm on Spartan-3 FPGA platform.

The proposed system serves as an efficient and economical framework for software design, testing, and analysis of satellite signal detection algorithms using Python for research study or education purposes but also as the first building block in the development process of real-time FPGA subsystem.

Importing Libraries Module

Imports the necessary python libraries such as NumPy, SciPy and Matplotlib. These libraries offer the tools that you need for signal generation, mathematical computing, filtering, and visualization.

Satellite Signal Generation Module

This modular creates similar signals like those transmitted from satellites using mathematical models. These generated signals are the signals that have originally been transmitted then serve as input to the detection process.

Noise Addition Module

Signals received on Earth from satellites are typically weak and noisy. This add-on implements the insertion of artificial noise to generated signals to recreate environmental conditions.

Signal Preprocessing Module

Preprocessing- to get the signal ready for analysis. This could include processes like normalizing, scaling and formatting the signal data so that it is appropriate for further signal processing.

Filtering Module

Double filtering, based on a maximum likelihood estimator (MLE), provides both inspection and desaturation of the signals, and was removed from further consideration. Separation via filtering extracts relevant harmonic components from clutter.

Correlation Detection Module

Correlation Detection Module: the processed signal is compared with a reference satellite signal. A peak in the correlation indicates a potential satellite signal.

Threshold Decision Module

The correlation value is compared with a preset threshold in this module. Satellites Signal Detection Algorithm Tuning Based on Threshold: When the value exceeds a specific threshold, it indicates that a satellite signal has been detected by the system.

Visualization Module

This module contains graphical outputs (signal waveforms, correlation graphs, detections results) using visualization libraries including Matplotlib.

Output Module

The last module displays the detection result to the user indicating whether a satellite signal is detected or not detected.

Algorithms:

To simulate, process and detect satellite signals in a heavy noise environment, several signal processing algorithms are used for the Satellite Signal Detection System.

1. Satellite Signal Generation Algorithm

This algorithm simulates satellite signals through the use of mathematical functions like sine waves or pseudo-random sequences. These signals are the signals from the satellites.

Steps:

1. Define signal parameters (frequency, amplitude, sampling rate):
2. Signal Data Preparation Generate a Time Sequence
3. Generate the satellite signal with example mathematical functions (sine wave, etc.)
4. Use $\log(1+x)$ to convert the signal into a logarithm.

Pseudocode:

```
BEGIN
Initialize frequency f
Initialize amplitude A
Initialize sampling_rate Fs
Initialize time_duration T
Generate time_sequence t from 0 to T with step 1/Fs
FOR each time value in t
  signal[t] = A * sin(2 * π * f * t)
END FOR
Store generated_signal
END
```

2. Noise Addition Algorithm

This algorithm adds noise to the generated signal to simulate real-world conditions.

Steps:

1. Take the generated satellite signal as input.
2. Generate random noise (Gaussian noise).
3. Add the noise to the original signal.
4. Produce the noisy signal for further processing.

Pseudocode:

```
BEGIN
Input generated_signal
Generate random_noise using Gaussian distribution
FOR each sample i in generated_signal
  noisy_signal[i] = generated_signal[i] + random_noise[i]
END
```

```
END FOR
Output noisy_signal
END
```

3. Signal Preprocessing Algorithm

This algorithm prepares the signal for analysis by normalizing and organizing the data.

Steps:

1. Input the noisy signal.
2. Normalize the signal values.
3. Remove unwanted fluctuations if necessary.
4. Pass the processed signal to the filtering stage.

Pseudocode:

```
BEGIN
Input noisy_signal
Find maximum_value of noisy_signal
FOR each sample i in noisy_signal
normalized_signal[i] = noisy_signal[i] / maximum_value
END FOR
Remove unwanted fluctuations if required
Output processed_signal
END
```

4. Digital Filtering Algorithm

This algorithm removes unwanted noise from the signal using digital filters.

Steps:

1. Input the preprocessed signal.
2. Select a suitable filter (low-pass filter or band-pass filter).
3. Apply the filter to reduce noise.
4. Output the filtered signal.

Pseudocode:

```
BEGIN

Input processed_signal
Select filter_type (Low-pass or Band-pass)
Apply digital_filter to processed_signal
filtered_signal = filter(processed_signal)
Output filtered_signal
END
```

5. Correlation Detection Algorithm

This algorithm detects the presence of satellite signals by comparing the received signal with a reference signal.

Steps:

1. Input the filtered signal and reference signal.
2. Perform correlation operation between the signals.
3. Identify the correlation peak.
4. Store the correlation value for decision making.

Pseudocode:

```
BEGIN
Input filtered_signal
Input reference_signal
Compute correlation between filtered_signal and reference_signal
Find correlation_peak value
Store correlation_result
END
```

6. Threshold Detection Algorithm

This algorithm determines whether a satellite signal is detected.

Steps:

1. Input the correlation result.
2. Compare the correlation value with a predefined threshold.
3. If $\text{correlation} \geq \text{threshold} \rightarrow \text{Signal detected}$.
4. If $\text{correlation} < \text{threshold} \rightarrow \text{Signal not detected}$.
5. Display the final result.

Pseudocode:

```

BEGIN
Input correlation_result
Set threshold_value
IF correlation_result  $\geq$  threshold_value THEN
Display "Satellite Signal Detected"
ELSE
Display "Satellite Signal Not Detected"
END IF
END
  
```

IV. SYSTEM ARCHITECTURE

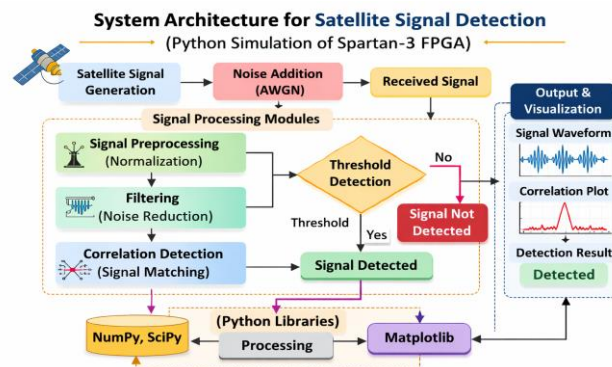


Figure 1: System Architecture for Satellite Signal Detection

Satellite Signal Generation Module

The module contains functions that generates simulated satellite signals based on mathematical models in Python. Generated signal transmits satellite signal, and acts as a reference for detection.

Noise Addition Module

This attitude is reflected in the module that simulates real-world conditions, which adds noise and interference to the signals generated under ideal satellite communications; there are various types of environmental noise and interference in actual use.

Signal Pre-processing Module

The pre-processing steps e.g. normalization and sampling prepares the received noisy signal for processing. This action saves the vector and prepares the signal for additional analysis.

Filtering Module

When a signal is received by the receiver, it goes through digital filters to minimize noise and enhance quality. This separation of signal components from noise disturbances is known as filtering.

Correlation Detection Module

We use correlation techniques in this module to compare the received signal with the reference signal(satellite signal of interest) When the patterns align, a peak of correlation emerges that suggests the presence of a possible satellite signal.

Threshold Detection Module

Then the correlation output is compared with a predefined threshold value. When the peak of the correlation is greater than a given threshold, this indicates that a satellite signal has been detected by the system.

Visualization Module

Visualization libraries in Python like Matplotlib show signal waveforms, noise effects and correlation peaks. It is used to analyze the operational behaviour of a signal and detection performance.

Output Module

The output from the last module is the detection result of whether a satellite signal has been detected. And outcome could be graphs and performance metrics.

V. RESULTS AND DISCUSSIONS

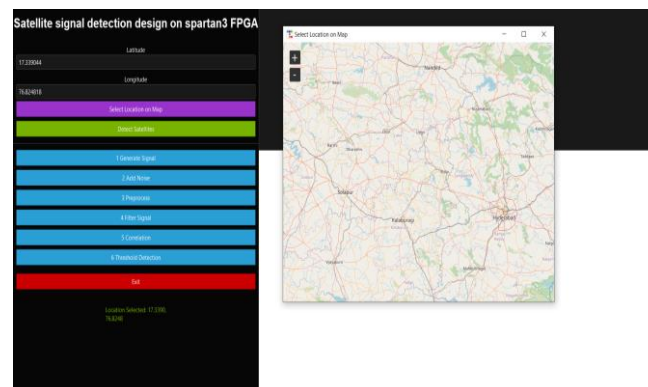


Figure 2: Selecting Location on Map

This is a simulated Graphical interface of Satellite signal detection ground station. The user selects a geographic location (by entering coordinates or clicking on the map), and the program calculates which satellites are visible above that location, given orbital data. This simulated signal is then corrupted with noise, processed through various filtering and correlation algorithms, and finally passes a thresholding algorithm to determine if the satellite signal has been detected. The blocks in this interface stand for signal processing stages which are usually realized on a Spartan-3 FPGA when acquiring and detecting satellite signals in real-time.

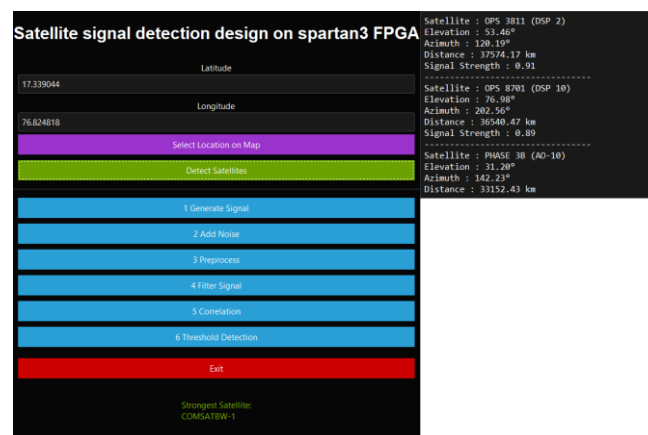


Figure 3: Detects Satellites based on Location

First, the user defines a geographic location by either inputting latitude and longitude or picking a point on the map. It then uses orbital data to identify the satellites that are currently visible above that position and shows their elevation, azimuth, and distance from the observer, and simulated signal strength. For the detection of

satellite signals, the data after being received from satellites will undergo a series processing pipeline to produce the signal that contains information about its frequencies, along with some noise having trace amounts of information.

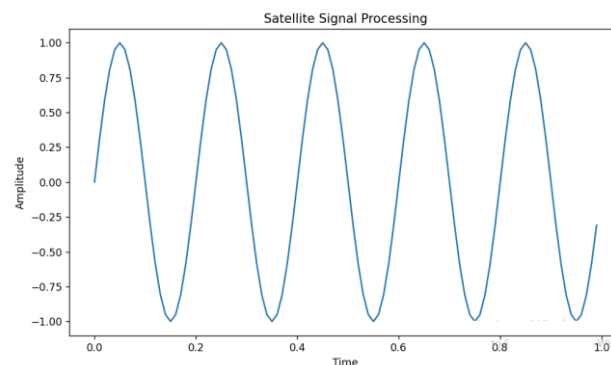


Figure 4: Satellite Signal Generation

Over time, a satellite signal is generated and processed, resulting in a sinusoidal signature as seen in the plot. In practice, this translates into the transmitted signal in a satellite communication system using a sine wave. The equation for such generated signal is as follows:

$$s(t) = A \sin(2\pi ft)$$

Where A is the amplitude of the signal, f is frequency and t is time. This results in the periodic waveform seen in the graph above. Noise may be added during signal processing to simulate real-world conditions:

$$x(t) = s(t) + n(t)$$

Where n(t) represents Gaussian noise. Errors and noise are removed from your signal, after which it's correlated with a reference signal to see if the transmission is from a satellite.] The correlation operation can be represented as follow:

$$R(\tau) = \sum x(t) \cdot r(t + \tau)$$

With x(t) denoting the received signal and r(t) representing the reference signal. If the correlation peak is above this threshold, it can be assumed that the satellite signal has been detected.

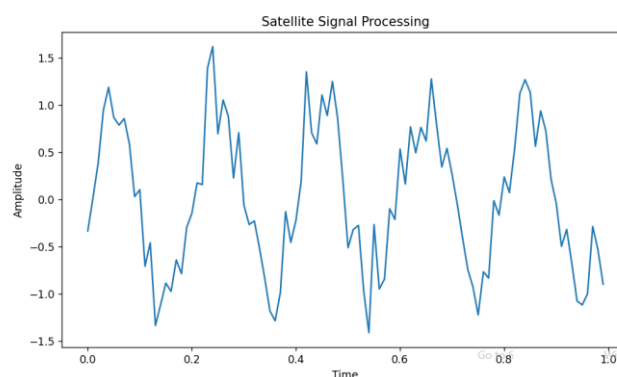


Figure 5: Add Noise

The figure above depicts an example of a satellite signal after noise has been added to it as one would see in real-world signal reception conditions. When you want to send a signal through satellite, the multiple signals are related sine wave ($s(t)=A \sin (2\pi ft)$; A = amplitude; f = frequency) based model. However, during transmission and reception the signal suffers disturbances from environmental and electronic noise. Thus, the observed signal is written as $x(t)=s(t)+n(t)$, and n(t) becomes random Gaussian noise. This added noise skews the sine wave, creating the erratic waveform shown in the graph. Noise is compared with traditional filters to retrieve original input data, and later signal range filtering techniques are applied for detection.

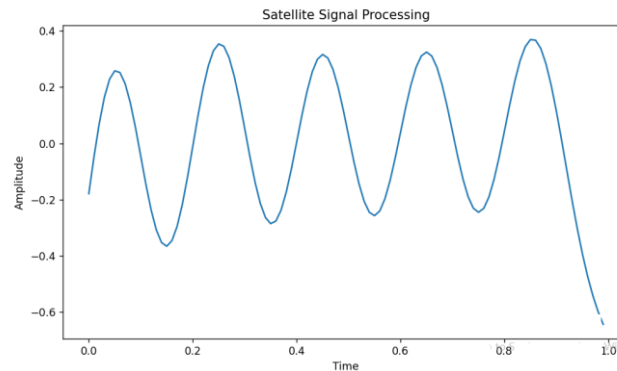


Figure 6: Filter Signal

This is the signal as received by the satellite (the same but with most noise removed and a smoother waveform thanks to digital filtering). This data is fed to the digital low pass Butterworth filter implemented in the system that will attenuate all high frequency noise components from the signal, preserving the information content of original signal [2]. This filtering process can be written as

$$y(t)=H(z)*x(t)$$

Where $x(t)$ is the input signal with noise, $H(z)$ is the filter transfer function, and $y(t)$ is the output signal. For the implemented method, filter coefficients are computed via a Butterworth design and then applied using

$$y(t)=\text{filtfilt}(b,a,x(t))$$

Where $b(a)$ are the filter coefficients. This is because, with the help of this, it can get rid of random noise and as such the correlation and threshold detection stages will now have an easier time determining if a satellite signal exists or not.

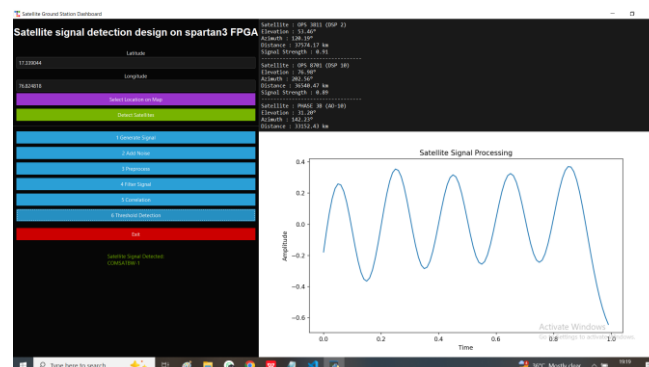


Figure 7: Satellite Signal detection

The figure illustrates the distinct GUI based satellite signal in this case for a Spartan-3 FPGA (Field Programmable Gate Array) simulation. The system first is provided with a geographic location by previously defined latitude/longitude coordinates or clicked from the map and must use this input to calculate satellites above that position using orbital data. The interface then shows intermission like satellite name, elevation, azimuth, distance and signal strength. The process involves generating a simulated satellite signal, adding noise, applying pre-processing and filtering to the signal, performing correlation with a reference signal (ideal case using Nil), and finally threshold detection to confirm whether the satellite is detected. Bottom: The processed satellite signal waveform over time; how filtering provides the complete damage recovery to make detecting reliable.

VI. CONCLUSION AND FUTURE WORKS

This paper introduces Satellite Signal Detection System which appears to be a great starting point for better and faster detection of satellite signals based on Python simulation environment. In this project, the generation of satellite signals and their processing is performed through different types of signal processing techniques such as noise addition, preprocessing, filtering, correlation detection and threshold decision. These techniques determine the weak satellite signals even in the noise and interference.

The approach accurately models the functional behavior of signal-processing actions commonly realized on Spartan-3 chip architecture – enabling developers to study and verify the detection algorithm ahead of

implementation on hardware. It allows for efficient calculation of signal using NumPy, SciPy and Matplotlib framework within Python libraries for visualizing the signal waveforms as well as correlation results.

Software-based simulation approach offers a relatively low-cost, flexible and change-friendly environment to implement satellite signal detection algorithms. This reduces complexity in development and allows researchers to try out various techniques from the signal processing ecosystem without costly or slow specialized hardware during First Silicon.

VII. FUTURE SCOPE

This allows to further improving the Satellite Signal Detection System by adopting more sophisticated designs of Machine Learning and signal processing algorithms to increase reliability under noisy environments. As a future study, the Python developed simulation can be deployed on the hardware platforms like Spartan-3 FPGA or any other high-end FPGAs for real time satellite signals processing. This system could also be expanded to allow signal detection from multiple satellite sources, as it would process signals in parallel if they were sent by different satellites. Further, integration of sophisticated algorithms like adaptive filtering, deep learning-based classification of signals and spectral analysis may be included to enhance functionality and resolve. In addition, user interface can be improved by designing a GUI to facilitate the interaction with users and represent in real-time signal processing steps. Further, things would also apply the system to show more real-world scenarios like satellite communication and GPS-based navigation systems as well as remote sensing technologies.

REFERENCES

- [1] M. Ibrahim, G. G. Hamza, Abdelhalim Zekry, and Ibrahim Mohamed Motawie, "FPGA Implementation of Acquisition phase of the GPS Receiver using XSG," *International Journal of Electronics and Telecommunication*, vol. 65, no. 4, pp. 739–744, 2019, Available: <https://ijet.pl/index.php/ijet/article/view/10.24425-ijet.2019.130257>
- [2] M. El-Hawary, G. G. Hamza, A. Zekry and I. Motawie, "FPGA Implementation of Tracking Phase of the GPS Receiver Using XSG," *Journal of Communications*, vol. 15, no. 4, pp. 367–371, Apr. 2020.
- [3] S. Guruprasad, "FPGA-Based Software GNSS Receiver Design for Real-Time Acquisition and Tracking," M.Eng. Thesis, York University, Canada, 2015.
- [4] A. Shapiro and B. Land, "FPGA-BASED REAL-TIME GPS RECEIVER Title: FPGA-based Real-time GPS Receiver," 2010. Accessed: Mar. 25, 2026. [Online]. Available: https://people.ece.cornell.edu/land/courses/eceprojectsland/STUDENTPROJ/2009to2010/ams348/hw_gps_receiver.pdf
- [5] Proc.. K. G., "IMPLEMENTATION OF GPS SIGNAL ACQUISITION AND TRACKING IN FPGA," *International Journal of Research in Engineering and Technology*, vol. 03, no. 03, pp. 942–946, May 2014, Available: <https://scispace.com/pdf/implementationof-gps-signal-acquisition-and-tracking-in-4koh3k02ib.pdf>
- [6] F. S. Larosa, "Design, Simulation, Implementation and Testing of Search and Tracking Modules for a FPGA-Based GPS Receiver," pp. 33–38, Apr. 2019, doi: <https://doi.org/10.1109/spl.2019.8714299>.
- [7] R. Kumari and M. Mukhopadhyay, "A FPGA Software based GPS Receiver Implementation with Signal blocker through Simulink," *Indian Journal of Science and Technology*, vol. 9, no. 25, Jul. 2016, doi: <https://doi.org/10.17485/ijst/2016/v9i25/89094>.
- [8] M. Nosrati, "FPGA-Based GPS Receiver for Space Mission," *Conf. Proc.*, 2017 (Verilog HDL implementation of GPS L1 C/A acquisition on FPGA).
- [9] Qiu, T., Wang, X., Du, Q. and Sun, Y. (2021), An innovative FPGA-based low-complexity and multi-constellations compatible GNSS acquisition scheme. *IET Radar Sonar Navig.*, 15: 370-381. <https://doi.org/10.1049/rsn2.12037>
- [10] Kh, Chetna & Rao, Malode. (2025). "Enhanced global navigation satellite system signal processing using field programmable gate array and system-on-chip based software receivers." *International Journal of Electrical and Computer Engineering (IJECE)*. 15. 480. 10.11591/ijece.v15i1.pp480-492.
- [11] MathWorks Example, "GPS Acquisition and Tracking Using AMD RFSoc Device," MATLAB & Simulink Example, MathWorks, 2024 (illustrates HDL design flow for GPS acquisition and tracking).
- [12] M. El-Hawary, G. G. Hamza, A. Zekry and I. Motawie, "FPGA Implementation of Tracking Phase of the GPS Receiver Using XSG," *Journal of Communications*, vol. 15, no. 4, pp. 367–371, Apr. 2020.
- [13] Kavya & Thulasibai, "Implementation of GPS Signal Acquisition and Tracking in FPGA," *Conf. Proc.* (details from semanticscholar).
- [14] IJECE Journal, "Enhanced GNSS Signal Processing Using FPGA Platforms," Recent research article, 2023.
- [15] Kh, Chetna & Rao, Malode, (2025) "Enhanced global navigation satellite system signal processing using field programmable gate array and system-on-chip based software receivers," *International Journal of Electrical and Computer Engineering (IJECE)*, 15. 480. 10.11591/ijece.v15i1.pp480-492.
- [16] Zhang and Y. Zhao, "Design and Optimization of Hardware-Interleaved Multi-Channel Correlators for GPS Tracking Loops Using Spartan-3 BRAMs," *Journal of Signal Processing Systems*, vol. 84, no. 1, pp. 45–58, Apr. 2017.
- [17] A. Shapiro and B. Land, "FPGA-BASED REAL-TIME GPS RECEIVER Title: FPGA-based Real-time GPS Receiver," 2010. Accessed: Mar. 25, 2026.