Edge CA-CFAR Data Reduction for Bandwidth-Efficient Real-Time Wideband Spectrum Sensing on Low-Cost SDRs

Yunsu Bae , Hajung Lee , Hyojun Park , Won-ho Jang , Byung-Jun Jang *

Posted Date: 22 May 2026

doi: 10.20944/preprints202605.1524.v1

Keywords: spectrum sensing; software-defined radio; CA-CFAR; FPGA; GPU; wideband monitoring; real-time processing; edge computing

 doi:10.20944/preprints202605.1524.v1

Article

Edge CA-CFAR Data Reduction for Bandwidth-Efficient Real-Time Wideband Spectrum Sensing on Low-Cost SDRs

Yunsu Bae [1], Hajung Lee [1], Hyojun Park [1], Won-ho Jang [2] and Byung-Jun Jang [1,*]

[1] Department of Electrical Engineering, Kookmin University, Seoul 02707, Republic of Korea

[2] SUNI Corporation, #516, #517, Incubation Center, 55, Hanyangdaehak-ro Gyeonggi-do 15588, Republic of Korea

* Correspondence: bjjang@kookmin.ac.kr

Abstract

Real-time wideband radio frequency (RF) spectrum monitoring is critical for unmanned aerial vehicle (UAV) detection and RF surveillance. Low-cost software-defined radio (SDR) networks are attractive but constrained by limited instantaneous bandwidth per node, I/Q data transfer bottlenecks over USB 2.0, and multi-node computational overhead. This paper proposes a bandwidth-efficient FPGA-GPU heterogeneous architecture addressing these limitations. A lightweight cell-averaging constant false alarm rate (CA-CFAR) IP core with O(1) complexity is deployed on the edge FPGA of each SDR node, forwarding only signal-containing intervals to reduce data transfer volume proportionally to the target duty cycle. Spectra from multiple nodes are stitched into a wideband view and processed in real time via a GPU-accelerated pipeline. The CA-CFAR IP occupies 16.3% of available LUTs with no BRAM and a fixed 10-cycle latency at 100 MHz. Experiments on a five SDR testbed demonstrate an 88% data transfer reduction at a 10% duty cycle, 370 μs end-to-end latency, 96.26% detection probability at −83.16 dBm (SNR ≈ 13 dB), and a 4.5× ~ 6.0× GPU speedup over CPU processing. These results confirm the practicality of high-performance wideband RF monitoring on resource-constrained SDR platforms.

Keywords: spectrum sensing; software-defined radio; CA-CFAR; FPGA; GPU; wideband monitoring; real-time processing; edge computing

1. Introduction

The rapid proliferation of wireless technologies in autonomous driving, aviation, defense and communications has led to a substantial increase in the use of the radio frequency spectrum. At the same time, research on spectrum sensing, which enables real-time monitoring and identification of diverse radio frequency (RF) signals in complex electromagnetic environments, has accelerated significantly [1–3]. Such real-time RF detection capability has become a core component of modern surveillance architectures, ranging from civilian security systems to military electronic warfare platforms [4]. In particular, unmanned aerial vehicles (UAVs) targeting urban areas and major national security facilities have recently emerged as a significant threat. Consequently, the importance of comprehensive, gap-free wideband spectrum coverage has increased as these noncooperative targets adopt increasingly sophisticated evasion strategies [5–7].

Noncooperative targets often conceal their emissions by using frequency hopping and short pulse waveforms to avoid detection. In a congested spectrum containing mixed wireless communication and radar emissions, the ability to detect meaningful signals in real time directly determines the reliability of an RF surveillance system. However, single-node surveillance systems suffer from two fundamental limitations: non-line-of-sight (NLoS) obstruction caused by surrounding terrain and the physical constraint on instantaneous bandwidth. To overcome these

 doi:10.20944/preprints202605.1524.v1

issues, distributed sensor networks that deploy multiple sensors across different spatial locations have emerged as an effective alternative [8,9]. As a result, modern RF surveillance architectures must jointly satisfy wideband coverage, real-time detection performance and cost effectiveness, given the need to operate multiple sensor nodes.

Conventional systems have typically relied on expensive RF front-ends with wide instantaneous bandwidth and dedicated high-speed signal processors [10,11]. However, this hardware-centric approach entails substantial deployment cost and power consumption, both of which severely limit scalability in distributed surveillance networks. To mitigate these drawbacks, low-cost software-defined radio (SDR)-based systems have been actively investigated [12–14]. Nevertheless, SDR-based platforms are constrained not only by ADC/DAC bandwidth but also by the transfer rate limits of host interfaces such as USB 2.0 and Ethernet when large volumes of raw I/Q data are processed [15,16]. When multiple sensor nodes operate simultaneously, the data generated by each node can reach tens to hundreds of megabytes per second, and transmitting such data in raw form to a central host causes network saturation and severe transfer delays. These bandwidth constraints fundamentally limit real-time operation regardless of processing capability.

Prior studies have addressed this challenge by adopting either GPU-based parallel processing or FPGA-based preprocessing in isolation [17–22]. However, GPU-based approaches can only process data after it reaches the host and therefore do not remove the upstream transfer overhead, limiting scalability. By contrast, FPGA-based approaches can partially reduce the data transfer burden, but they face structural limitations when multichannel synchronization, wideband spectrum stitching and high-resolution signal processing must all be performed concurrently under constrained on-chip memory and computational resources. Consequently, either approach alone has practical limitations in satisfying the performance requirements of multi-SDR, real-time wideband detection systems.

Accordingly, this paper proposes a real-time wideband RF spectrum sensing system that combines edge-side data reduction with high-speed parallel host processing to eliminate both the data transfer overhead and the computational bottleneck. As illustrated in Figure 1b, the proposed system performs cell-averaging constant false alarm rate (CA-CFAR)-based preprocessing on the FPGA connected to each SDR node to suppress noise-only samples before transmission and then uses GPU parallelism to process spectra from multiple channels jointly. This hybrid approach delivers low-latency wideband processing performance that is difficult to achieve using either technology independently. The main contributions of this work are as follows:

- Edge-side data reduction via a resource-efficient O(1)-complexity FPGA CA-CFAR: A high-speed lightweight CA-CFAR IP core is designed for the Zynq-7010 FPGA in each SDR node. It uses in-place recursive summation and arithmetic right shift division to maintain O(1) computational complexity. The design occupies 16.3% of the available lookup tables (LUTs) and consumes no block RAM (BRAM) resources, while enabling real-time operation with a fixed latency of 10 clock cycles. Using this edge-based data reduction technique, 88% of host-bound data is removed in a 10% duty cycle environment, alleviating the transfer throughput limitation.
- Multithreaded software synchronization and parallel multi-SDR control: To compensate for the narrow instantaneous bandwidth of a single low-cost SDR, a multithreaded software architecture is implemented for stable parallel operation of multiple SDR nodes. Rather than relying on expensive external clock synchronization equipment, the host scheduler controls the sample-level data streams assigned to each node through a software-based synchronization scheme. A protocol data unit (PDU)-based scheduling structure reduces data loss, and GPU-accelerated computation at the PDU level enables low-latency signal processing.

To validate the proposed system, a custom hardware testbed was constructed and subjected to end-to-end evaluation. Experimental results demonstrate that the system achieves up to 88% reduction in data transfer volume relative to conventional approaches, an end-to-end latency of 370 μs, a detection probability exceeding 95% (Pd = 96.26%) at −83.16 dBm received power (noise floor −96 dBm) and a 4.5× ~ 6.0× processing speedup over CPU-based computation. These results

 doi:10.20944/preprints202605.1524.v1

demonstrate that the proposed architecture can support real-time wideband RF monitoring on resource-constrained SDR platforms.

The remainder of this paper is organized as follows. Section 2 presents the proposed system architecture from two complementary perspectives. Section 3 describes the hardware and software implementation of each component. Section 4 reports the experimental setup and quantitative performance evaluation. Section 5 concludes the paper and outlines future research directions.

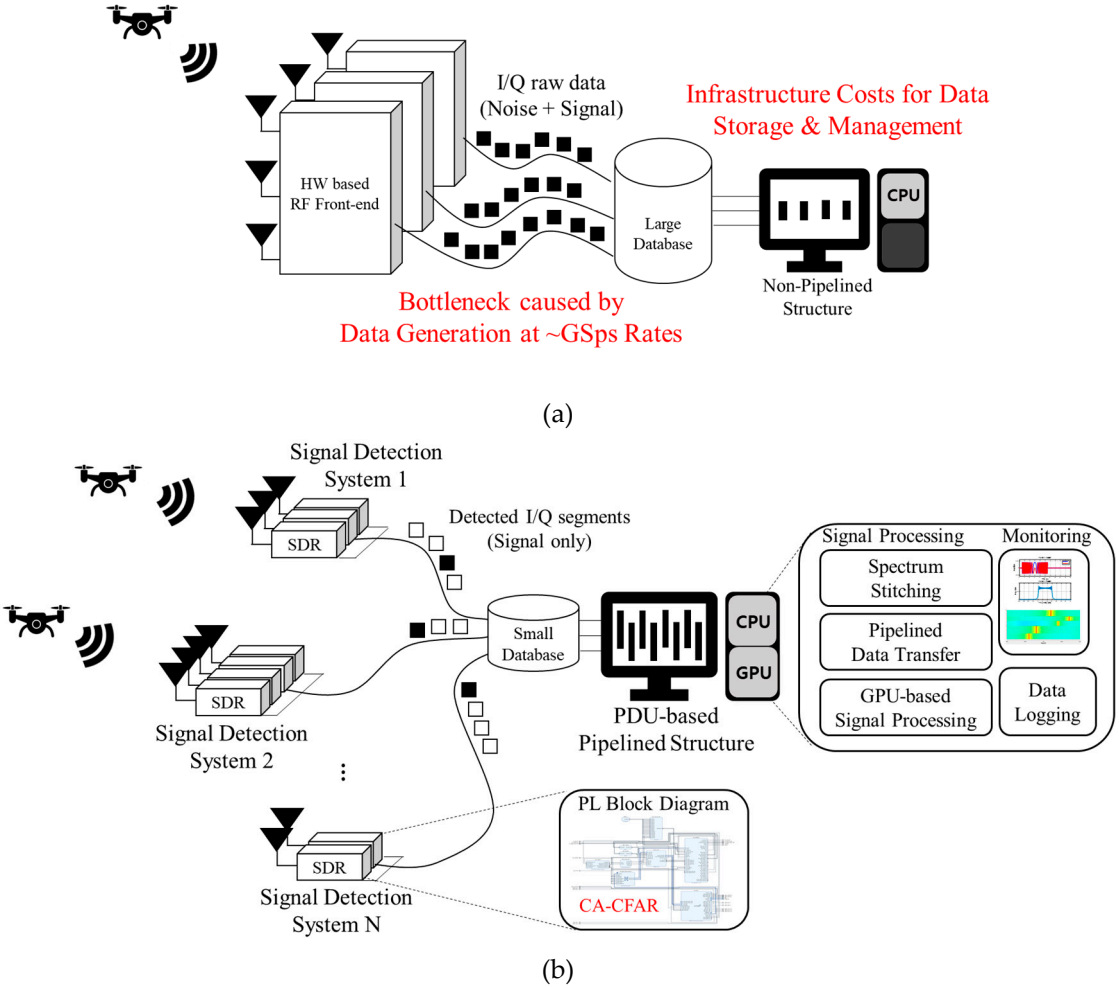

Figure 1. Comparison of spectrum sensing architectures: (a) Conventional high-cost hardware-centric system suffering from data bottlenecks; (b) Proposed FPGA-GPU acceleration system employing edge-based data reduction.

2. Proposed System Architecture

The proposed RF spectrum sensing system consists of two principal processing stages: (1) an RF front-end and preprocessing stage that collects signals from multiple SDR nodes and performs FPGA-based data reduction; and (2) a host-side acceleration stage responsible for data aggregation, pipeline control and high-speed GPU parallel computation.

2.1. FPGA-Based CA-CFAR Design for SDR Nodes

In electronic warfare (EW) and signal intelligence (SIGINT) environments, the parameters of RF signals emitted by a target are unknown a priori. Accordingly, the spectrum detector must identify unknown signals without prior knowledge of waveform, modulation scheme or bandwidth. Because conventional matched filter-based pulse compression requires a replica of the transmitted signal, it cannot be applied in this scenario [23]. Therefore, this work adopts instantaneous power-based detection of received I/Q data to determine whether a signal is present [24].

 doi:10.20944/preprints202605.1524.v1

In practice, RF signals emitted by a target are received by the SDR antenna and RF transceiver, converted into digital samples by the ADC and forwarded to the FPGA. Since the proposed system operates multiple SDR nodes in parallel for wide-area and wideband coverage, the resulting I/Q data volume creates severe pressure on the limited interface bandwidth of low-cost SDR equipment. To ensure real-time operation, an edge-based data reduction process that forwards only meaningful signal intervals before host transmission is essential. A fixed threshold is not suitable because detection reliability varies with the noise level. Therefore, the CA-CFAR algorithm [25], which adaptively adjusts the threshold according to the surrounding noise power, is applied in the time domain within the FPGA.

Time domain CA-CFAR determines the detection threshold by estimating the average power of reference cells located on both sides of the cell under test (CUT). Guard cells are placed adjacent to the CUT to prevent self-masking, which occurs when the pulse edge leaks into the reference cells. With $N/2$ reference cells on each side of the CUT, the estimated average noise power P_n divided by the number of total reference cells N is:

$$P_n = \left(\frac{1}{N}\right) \times \left[\sum_{i=1}^{N/2} y[n-G-i] + \sum_{i=1}^{N/2} y[n+G+i]\right] \quad (1)$$

where $y[n]$ denotes the instantaneous power of sample n, N is the number of reference cells, and G is the guard cell length. Assuming additive white Gaussian noise (AWGN), the target false alarm probability P_{fa} is defined as the integral of the probability density function above threshold T. The threshold constant K and the final threshold T are given by:

$$K = N \times \left(P_{fa}^{-1/N} - 1\right) \quad (2)$$

$$T = K \times P_n \quad (3)$$

This threshold enables precise time-domain detection of pulse start and end times without prior knowledge of signal parameters. Figure 2 illustrates an example of the time domain CA-CFAR algorithm.

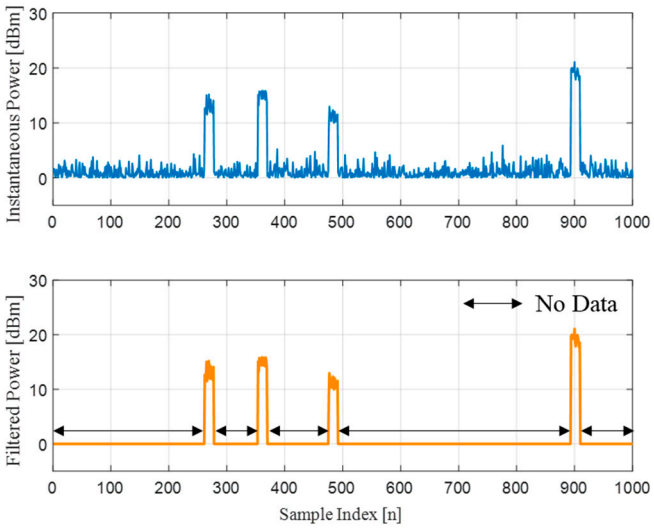

Figure 2. Example of the CA-CFAR algorithm in the time domain. The upper panel shows the instantaneous received power; the lower panel shows the CA-CFAR filtered output, where noise-only intervals are discarded.

In the proposed system, a CA-CFAR intellectual property (IP) core is deployed in the FPGA region of each SDR. The deployed CA-CFAR IP removes noise samples and outputs only samples corresponding to detected pulses. Directly porting a conventional software-based CA-CFAR

 doi:10.20944/preprints202605.1524.v1

implementation onto an FPGA would result in inefficient resource usage, higher complexity and excessive memory consumption [26,27]. Conventional CA-CFAR uses a large adder tree to sum the reference cell data in a sliding window, requires a hardware-intensive divider to compute the average noise power, and consumes substantial on-chip BRAM resources for delay line implementation. As a result, resource consumption and latency increase rapidly with window size.

In contrast, the proposed CA-CFAR architecture is designed to use the limited hardware resources of the low-cost SDR FPGA as efficiently as possible. While a conventional adder tree recomputes the sum of all N elements for each window shift, yielding $O(N)$ complexity, the proposed architecture uses the following in-place recursive update rule:

$$s[n] = s[n-1] + x_{new}[n] - x_{old}[n] \tag{4}$$

where $s[n]$ is the total power sum of the reference cells computed in the previous clock cycle, $x_{new}[n]$ is the power of the newly entered reference cell and $x_{old}[n]$ is the power of the oldest departing reference cell. This reduces the hardware complexity to a constant $O(1)$, since only two add/subtract operations are performed per clock cycle.

To compute the average noise power without a hardware divider, the number of reference cells is fixed to 2^M, allowing the division operation to be replaced by:

$$P_n = \frac{1}{2^M} \times \sum_{i=1}^{2^M} P_i \tag{5}$$

with a simple arithmetic right shift operation:

$$P_n = \sum_{i=1}^{2^M} P_i \gg M \tag{6}$$

where 2^M is the total number of reference cells and P_i is the power of the i-th reference cell within the window. When a received sample arrives at the CUT, the sample is immediately discarded if it is classified as noise. Only samples with an active detection flag are forwarded to the host PC, reducing the total transferred data volume to a level proportional to the target duty cycle and thereby alleviating the transfer load on conventional low-cost SDR interfaces. Specifically, when the CA-CFAR threshold comparison classifies the CUT as a noise-only sample, the detection flag register is explicitly set to the inactive state; when the sample is subsequently passed to the output stage, the register state is checked and inactive interval samples are discarded immediately without transmission.

In conventional FPGA-based signal processors, BRAM is primarily used to implement the sliding window delay line; however, BRAM access inevitably introduces memory read/write latency. To avoid this latency penalty, the proposed CA-CFAR uses no BRAM. The sliding window register chain is implemented entirely with lookup table (LUT)-based shift registers, which are the fundamental logic elements of the FPGA, rather than BRAM. This minimizes routing latency, enables high-speed clock operation and preserves BRAM resources for other processing tasks in the system. Figure 3 compares the conventional and proposed CA-CFAR architectures.

 doi:10.20944/preprints202605.1524.v1

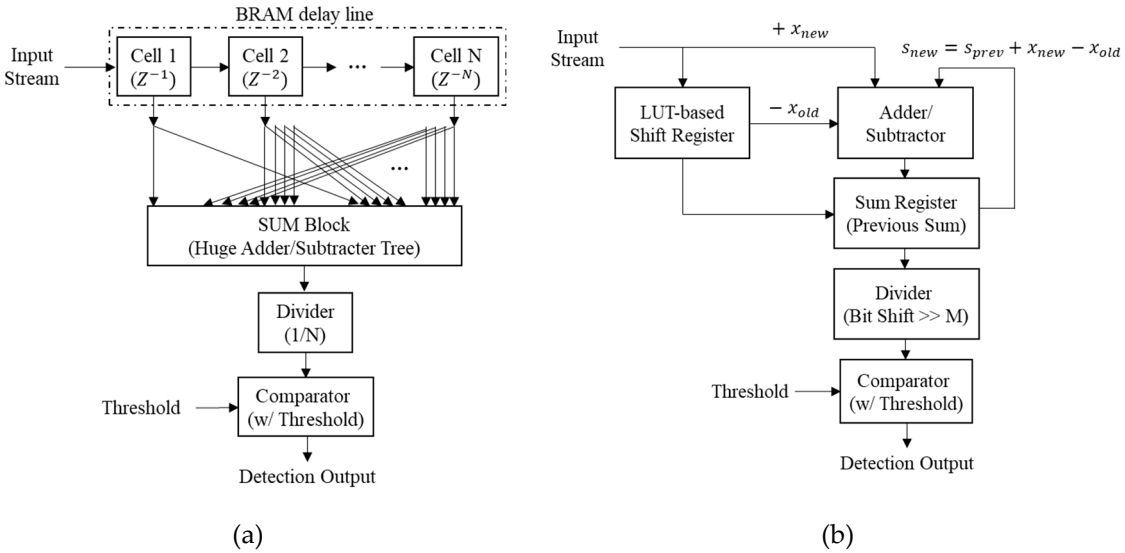

Figure 3. Comparison of CA-CFAR algorithm architectures: (a) Conventional CA-CFAR using an adder tree, divider and BRAM delay line; (b) Proposed CA-CFAR using in-place recursive summation, bit shift division and LUT-based shift registers.

2.2. Host and GPU-Based Integrated Signal Processing Architecture

In an RF detection system that uses multiple distributed SDR nodes, the host must provide a management framework that can collect large volumes of data from each node without delay and reliably control all devices. For this purpose, the host software in the proposed system is designed as a multithreaded architecture. As shown in Figure 4, functions such as SDR device control, I/Q data reception and data processing are each assigned to independent threads and executed in parallel. This architecture prevents temporary delay at individual nodes from interrupting overall system operation. To minimize data loss, the dedicated reception thread is assigned the highest OS-level priority, which helps maintain stable streaming in multi-node environments.

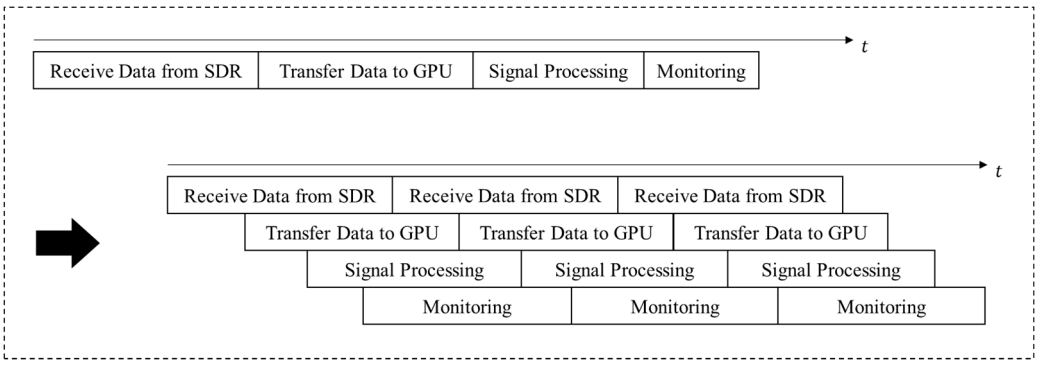

Figure 4. Pipelined multithreaded architecture. Each SDR node is assigned dedicated receive, transfer, signal processing and monitoring threads operating in parallel.

Given that raw I/Q data flowing from the SDR to the host is a simple sample stream lacking inherent metadata, parallel processing by multiple threads makes it difficult to preserve source identity and sample ordering. To solve this data management problem, the proposed system introduces a PDU-based communication and scheduling structure. A PDU consists of a header and a payload. The header contains metadata such as center frequency, bandwidth, SDR identifier and timestamp. The payload stores valid I/Q samples passing the FPGA CA-CFAR threshold, grouped into FFT length chunks and immediately encapsulated by the reception thread. A data storage thread communicates directly with the reception thread to save data in binary, MAT, or CSV format, while

 doi:10.20944/preprints202605.1524.v1

metadata is stored in separate JSONL files. For display purposes, PDUs produced by the reception thread are loaded into a priority-queue-based memory buffer, and the scheduler sorts them by metadata before forwarding them to the signal processing thread.

Using this data management framework, the proposed system performs spectrum stitching to overcome the physical bandwidth limitation of a single low-cost SDR system [28]. As shown in Figure 5, spectrum stitching synchronizes and merges data from nodes assigned to adjacent frequency bands, generating a single wideband virtual spectrum map. In electronic warfare and signal collection environments where unknown signals must be detected without prior knowledge of the frequency band or modulation scheme, this approach provides wideband coverage comparable to that of a dedicated wideband receiver, using only a small number of low-cost devices.

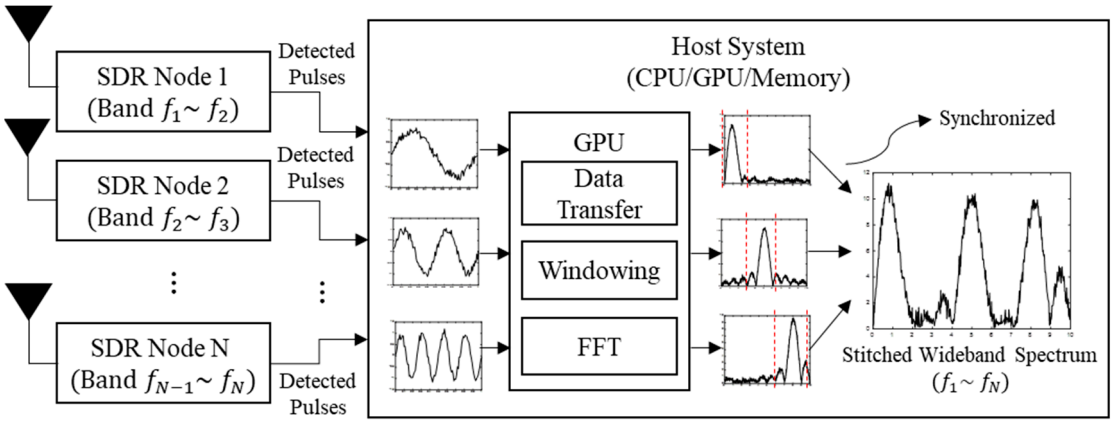

Figure 5. Conceptual diagram of spectrum stitching. Multiple SDR nodes each covering a subband (Band f_1~f_2, f_2~f_3, ...) are synchronized and stitched into a single wideband spectrum at the host GPU.

To perform spectrum stitching and signal processing efficiently, the PDUs sorted by header are subjected to computationally intensive operations such as high-resolution FFT. If a CPU alone performs this task for data from multiple nodes, the resulting computational load limits real-time operation [29]. Therefore, the signal processing thread offloads the input PDUs to the GPU. The GPU performs windowing, FFT, magnitude computation and spectrum stitching through parallel real-time processing. The resulting spectrum is then rendered directly in the GPU's memory.

3. System Implementation

The proposed system was implemented using low-cost ADALM-Pluto SDR units [15,16] as shown in Figure 6. This section details the practical realization of the architecture presented in Section 2 across both the hardware and software layers. By achieving high-performance spectrum sensing on accessible hardware, the system underscores the efficiency of the proposed FPGA and GPU processing pipelines, making the architecture suitable for scalable deployment in distributed RF sensing scenarios.

 doi:10.20944/preprints202605.1524.v1

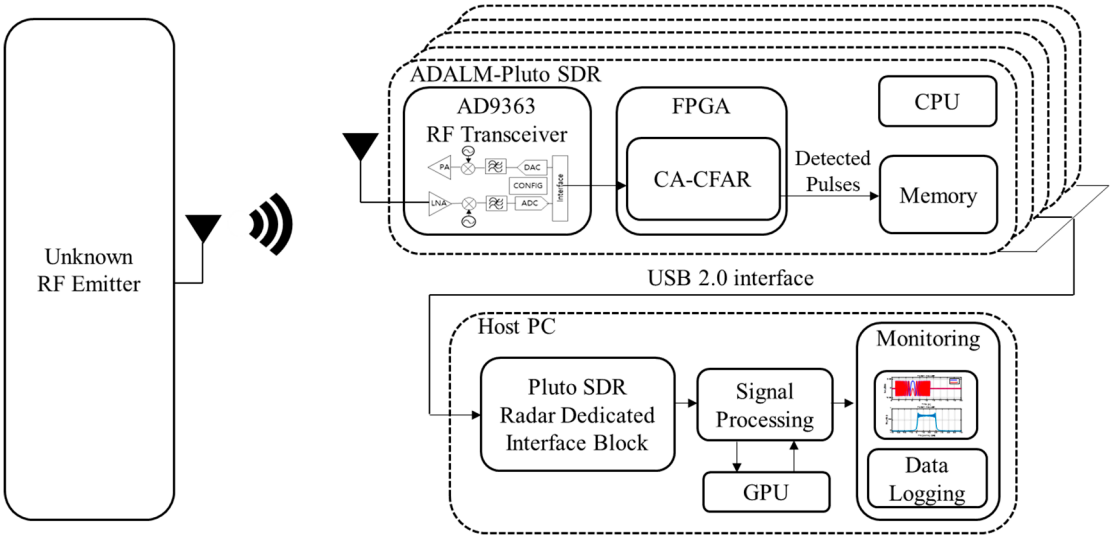

Figure 6. Block diagram of the proposed system hardware and software implementation. Each ADALM-Pluto SDR node features an AD9363 RF transceiver, FPGA integrated with the proposed CA-CFAR IP and connects to the host PC via USB 2.0.

3.1. CA-CFAR Implementation

Table 1 summarizes the CA-CFAR parameters used for RF signal detection, determined based on the actual RF operating environment. The number of reference cells used to estimate the noise power around the target signal was fixed at a power of two ($64 = 2^6$) to enable arithmetic right shift computation. The guard cell length was set to 12 to prevent target signal energy from leaking into adjacent cells and distorting the noise estimate. The target false alarm probability P_{fa} was set to 10^{-4}, with a corresponding scaling factor K of 9.9063 derived from $P_{fa} = (1 + K/N)^{-N}$ with $N = 64$.

Table 1. Parameters of the proposed CA-CFAR algorithm.

Parameter	Ref. Cell	Guard Cell	P_{fa}	K
Proposed CA-CFAR IP	64	12	10^{-4}	9.9063

To validate the proposed high-speed lightweight CA-CFAR algorithm, the CA-CFAR IP was integrated into the FPGA region of the Zynq-7010 SoC embedded in the ADALM-Pluto SDR. Development was performed using Xilinx Vitis High-Level Synthesis (HLS) and Vivado [30]. The proposed lightweight CA-CFAR algorithm was synthesized into register transfer level (RTL) code and packaged as a single IP core, which was then inserted into the receive data path of the ADALM-Pluto SDR FPGA. Figure 7 shows the IP block diagram of the ADALM-Pluto SDR FPGA region with the proposed CA-CFAR IP inserted.

Table 2 compares the hardware resource utilization of the proposed high-speed lightweight CA-CFAR IP, synthesized with Vitis HLS, against the conventional CA-CFAR implementation and the available resources of the Zynq-7010 SoC-based ADALM-Pluto SDR. As shown in Table 2, the proposed algorithm yields a compact implementation suitable for low-cost FPGA environments such as the ADALM-Pluto SDR (Zynq-7010), where firmware for basic USB communication, filters and interfaces already occupies more than half of the available resources. The conventional CA-CFAR structure consumes 8 DSP blocks and 2 BRAM blocks and incurs 487 clock cycles. By contrast, the proposed architecture uses no BRAM and only 4 DSP blocks—the scarcest on-chip resources—and occupies 16.3% of the available LUTs and 11.6% of the available FFs. Remarkably, the proposed design achieves nearly a 49-fold reduction in latency, requiring only 10 clock cycles compared to the 487 cycles of the conventional approach. Data reduction is thus achievable with a fixed, minimal latency even in resource-constrained FPGA environments without resource exhaustion or timing

 doi:10.20944/preprints202605.1524.v1

constraint violations. At a clock frequency of 100 MHz, 10 clock cycles correspond to 100 ns of latency, which is sufficiently low for real-time RF signal detection and display.

Table 2. Hardware resource utilization of the proposed CA-CFAR IP vs. total resources of the ADALM-Pluto SDR (Zynq-7010).

	Latency (clocks)	BRAM	DSP	FF	LUT
Pluto SDR (Total)	-	60	80	35,200	17,600
Conventional CA-CFAR	487	2	8	967	1,825
Proposed CA-CFAR	10	0	4	4,068	2,861

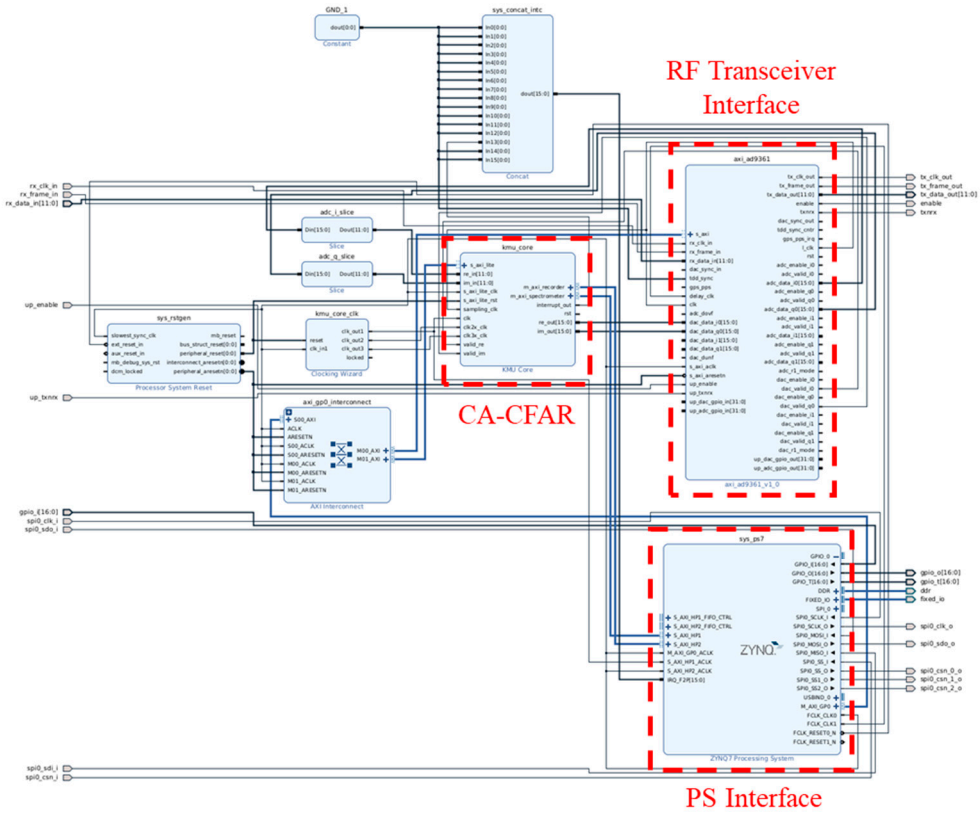

Figure 7. IP block diagram of the ADALM-Pluto SDR FPGA region with the proposed CA-CFAR IP inserted into the receive data path between the AD9363 transceiver and the USB output interface.

3.2. Multithreaded Operation and GPU Signal Processing Implementation

The multithreaded operation scheme is implemented in C/C++. The signal processing thread uses the CUDA API to exchange data with the GPU. Pinned memory and asynchronous transfers are employed to minimize latency caused by CPU-GPU memory copies. Pinned memory refers to a fixed memory region that is not subject to OS page replacement; by keeping the physical address of the data fixed during transfer, it allows the DMA controller to access memory directly without intermediate buffering. In contrast, conventional pageable memory requires the OS to copy data to an internal staging buffer before initiating DMA transfer; pinned allocation eliminates this intermediate copy step entirely, reducing both transfer latency and CPU involvement. Asynchronous transfer allows the CPU to proceed immediately to the next task after issuing a GPU transfer command, without blocking until the transfer completes. Together, these techniques significantly reduce memory copy latency and enable near-continuous RF monitoring.

 doi:10.20944/preprints202605.1524.v1

The GPU performs windowing and FFT on input data arranged in PDU-length units sorted by the PDU header. The maximum sampling frequency of the ADALM-Pluto SDR is 61.44 MSps, yielding a frequency resolution of:

$$\Delta f = \frac{f_s}{N} = \frac{61.44\ MHz}{1024} = 60kHz \tag{7}$$

where f_s is the sampling frequency and N is the FFT size. After the FFT, spectrum stitching is performed. To align data from multiple devices accurately along the frequency axis, the proposed system uses a software-based method that minimizes timing synchronization errors among the SDRs. Communication between the SDR and the host is structured such that the host issues a streaming command and the SDR initiates streaming data transmission immediately upon receipt. The host schedules data streaming requests at sample period intervals to achieve temporal synchronization among all controllable SDRs. When the number of operating nodes exceeds the available host thread count, transmission cycles for node groups are scheduled at sample intervals to maintain data continuity. The time axis resolution T_{res} is determined by the reciprocal of the sampling rate:

$$T_{res} = \frac{1}{f_s} = \frac{1}{61.44MHz} \approx 16.276ns \tag{8}$$

The resulting time resolution of 16.276 ns establishes the temporal precision with which signal events can be localized at the sample level within each PDU window. Equations (7) and (8) together define the time–frequency sensing granularity of the proposed pipeline: $\Delta f = 60kHz$ for channel-level discrimination across the stitched wideband spectrum, and T_{res} = 16.276 ns for fine-grained temporal localization of spectral events. Both resolutions are determined solely by the hardware sampling parameters f_s and N, and therefore remain consistent regardless of the number of stitched nodes or the specific frequency band under observation.

4. Experimental Results and Discussion

4.1. Testbed and Experimental Setup

Figure 8 shows the block diagram of the experimental setup. The high-speed lightweight CA-CFAR algorithm deployed in the FPGA was first verified to accurately identify and output only valid RF signal intervals. Experiments were conducted by transmitting linear frequency modulated (LFM) pulse signals from a signal generator and receiving them with a self-built hardware testbed using five ADALM-Pluto SDR units.

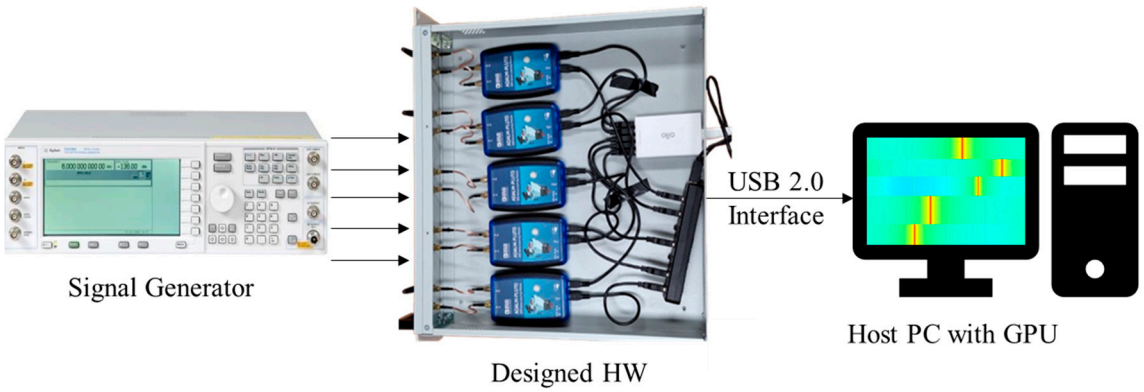

Figure 8. Block diagram of the experimental setup, showing signal flow and system components.

 doi:10.20944/preprints202605.1524.v1

4.2. Wideband Coverage and Data Reduction

Figure 9 shows the real-time wideband spectrogram output of the proposed system. The five ADALM-Pluto SDR nodes were configured with center frequencies spanning the C-band from 5.725 GHz to 5.825 GHz—a band predominantly occupied by drone control and video transmission signals. Each node covered 20 MHz, providing a combined coverage of 100 MHz. To confirm that the proposed system detects only valid signals while excluding noise, five pulses with arbitrary pulse widths, bandwidths, center frequencies and power levels sufficiently above the detection threshold were transmitted.

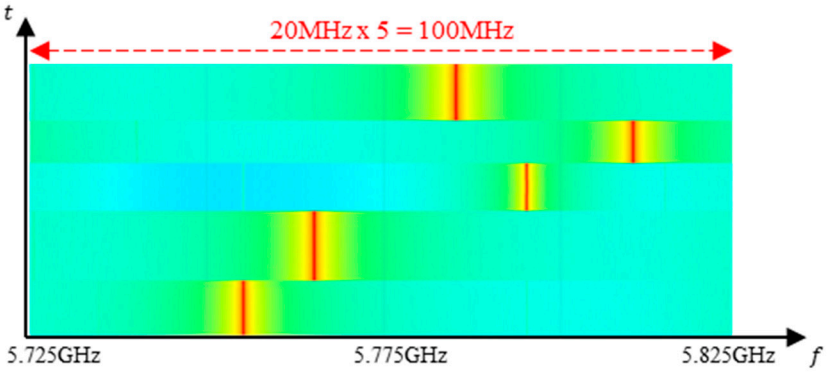

Figure 9. Real-time wideband spectrogram of the proposed system covering 100 MHz (5.725–5.825 GHz) using five ADALM-Pluto SDR nodes. Only detected pulses are displayed; noise-only intervals are discarded by the edge CA-CFAR IP.

The experimental results confirmed that, after passing through the CA-CFAR IP, signal output was activated only during pulse intervals whose energy exceeded the detection threshold, while samples from noise-only intervals were suppressed at the SDR stage. For each detected event, pulse descriptor words (PDWs)—including arrival time and pulse width—are accessible through the internal PDU metadata.

4.3. CA-CFAR Detection Performance

To evaluate the efficiency of the proposed method quantitatively, the data reduction effect was quantified by comparing binary file sizes recorded on the host with pulse detection enabled and disabled. When data were collected for one second in a 10% duty cycle RF signal environment, the file size with pulse detection enabled was 88% smaller than that obtained with detection disabled. Figure 10 compares the required data rate per SDR node with the USB 2.0 interface bandwidth limit. Selective data transmission at the SDR stage thus reduces host interface utilization well below the physical limit, validating the edge reduction approach.

CA-CFAR detection performance was characterized by measuring detection probability as a function of received signal power. The experiment was conducted at room temperature using a signal generator connected via an SMA coaxial cable. The input signal was configured as a pulse train with a pulse repetition interval (PRI) of 100 μs and a duty cycle of 10% (pulse width of 10 μs) and detection performance was measured while the signal power was varied. The FFT size was set to 1024 for signal processing and the average noise floor was measured to be −96 dBm. As shown in Figure 11, the detection probability increases gradually with received power, reaching approximately 50% at −86.1 dBm. For signal power above −83.16 dBm, the detection probability first exceeds 95%, achieving Pd = 96.26%, and approaches 99% above −82.4 dBm. The proposed hardware architecture thus achieves a minimum detectable SNR of approximately 13 dB, providing reliable detection performance under practical low-SNR conditions.

 doi:10.20944/preprints202605.1524.v1

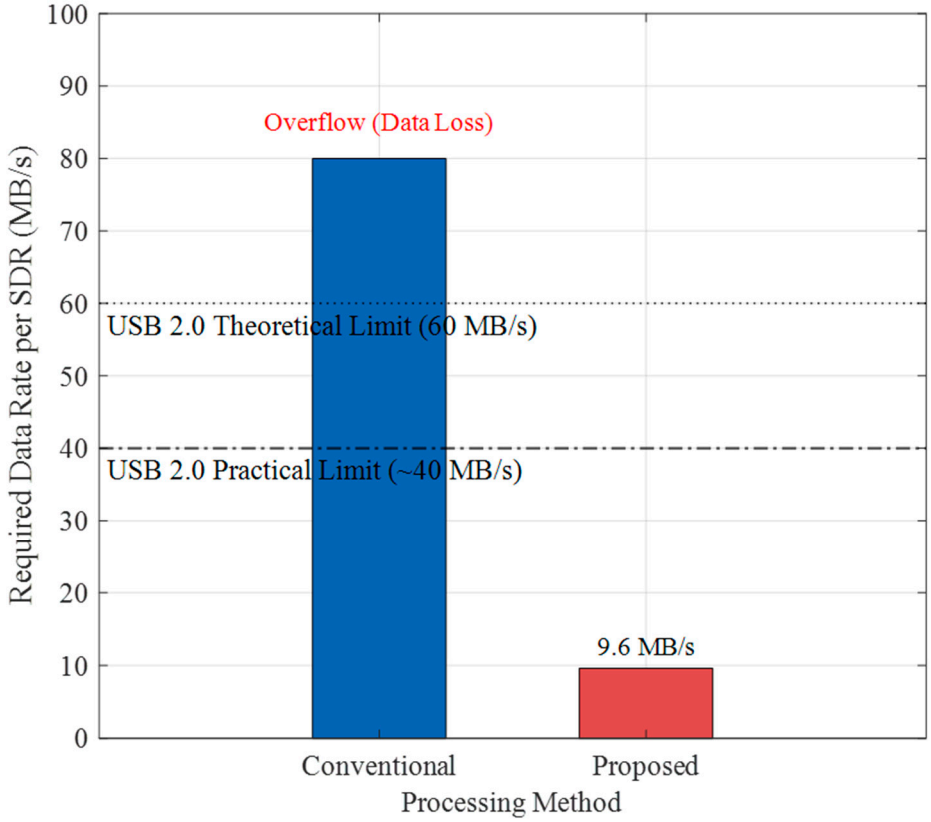

Figure 10. Comparison of the required data rate per SDR node under the USB 2.0 interface bottleneck limit at a 10% signal duty cycle. The conventional approach exceeds the practical limit of approximately 40 MB/s whereas the proposed edge CA-CFAR reduces the rate to well below the threshold.

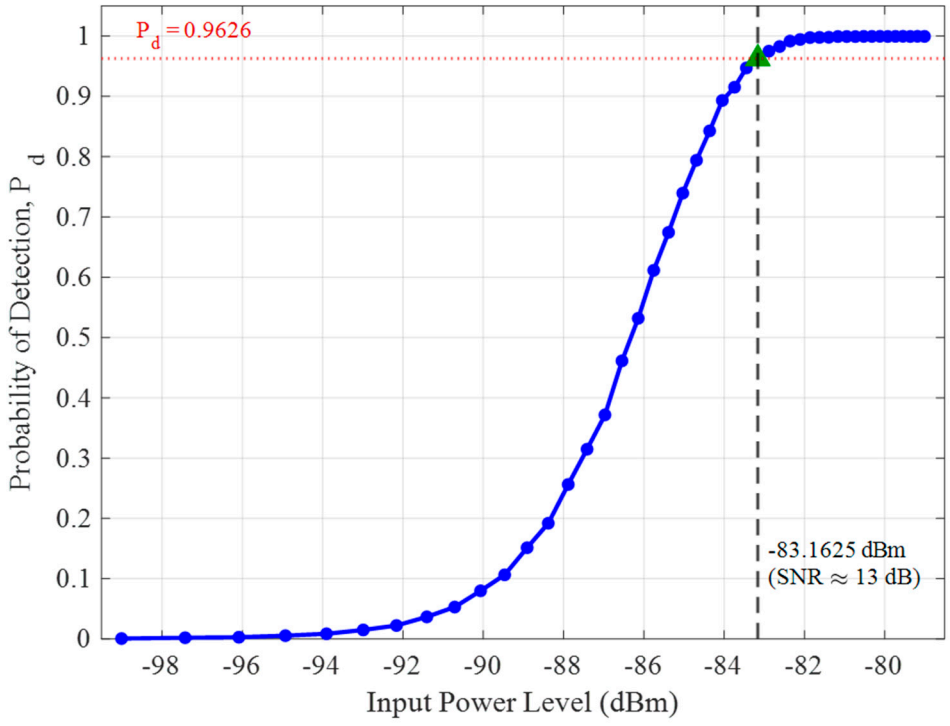

Figure 11. Measured probability of detection (P_d) versus input power level. At a noise floor of −96 dBm, P_d first exceeds 95% (Pd = 96.26%) at −83.16 dBm, indicating a minimum detectable SNR of approximately 13 dB.

 doi:10.20944/preprints202605.1524.v1

The detection–false alarm trade-off of the proposed CA-CFAR detector was characterized across its complete operating range using receiver operating characteristic (ROC) curves generated by Monte Carlo simulation. The simulation replicates the exact fixed-point integer pipeline of the FPGA IP core: 16-bit signed I/Q samples, power metric $I^2 + Q^2$ implementation using 4 DSP blocks, accumulation over $N_{total} = 64$ reference cells with exact division by a 6-bit right shift and Q8 threshold multiplication ($K_{num} = 2536, \gg 8$). Each ROC curve is traced by sweeping the threshold factor K over a logarithmic grid and recording the empirical P_{fa} and P_d from $M = 2 \times 10^6$ independent trials per operating point, ensuring a statistical resolution of approximately 5×10^{-7} in P_{fa}.

The threshold factor $K = 9.9063$ implemented in the hardware was derived from the standard CA-CFAR false alarm formula $P_{fa} = (1 + K/N)^{-N}$ with $N = 64$ and $P_{fa} \approx 10^{-4}$. While the theoretical formula assumes continuous precision, the hardware-exact simulation identifies that $K_{fp} = 9.9063$ yields an empirical P_{fa} of 1.24×10^{-4}. A slightly higher calibrated constant, $K_{emp} = 10.15$, is found to achieve $P_{fa} = 10^{-4}$ precisely. The small deviation of the empirical P_{fa} (1.24×10^{-4} $vs.$ 10^{-4}) arises from fixed-point quantization of the Q8 multiplier and is well within the acceptable false alarm tolerance of the system.

Figure 12 presents the resulting ROC curves for SNR values from 7, 9, 11, 13 dB under the Swerling 0 (nonfluctuating) target model. At the hardware-implemented operating point ($K_{fp} = 9.9063$, $P_{fa} = 1.24 \times 10^{-4}$), the simulation yields $P_d = 0.972$ at SNR = 13 dB. This is in direct agreement with the C-simulation result $P_d > 0.95$ of Pd = 96.26% at −83.16 dBm (noise floor −96 dBm, SNR ≈ 13 dB) reported in Section 4.3. This agreement, obtained without free fitting parameters, validates both the adopted signal model and the fixed-point power metric implementation.

The ROC curves further elucidate the minimum detectable SNR of approximately 13 dB observed in the hardware experiments. At the design threshold, a 2 dB increase in SNR from 11 dB to 13 dB raises P_d sharply from 0.75 to 0.97. This steep transition is characteristic of a non-fluctuating (Swerling 0) target in Gaussian noise and is entirely consistent with the steep onset of detection observed above −83.16 dBm in Figure 11. Together, the simulation and hardware results confirm that the detector achieves $P_{fa} \approx 10^{-4}$ and $P_d > 0.95$ at SNR = 13 dB, successfully meeting the design objective with hardware-exact precision.

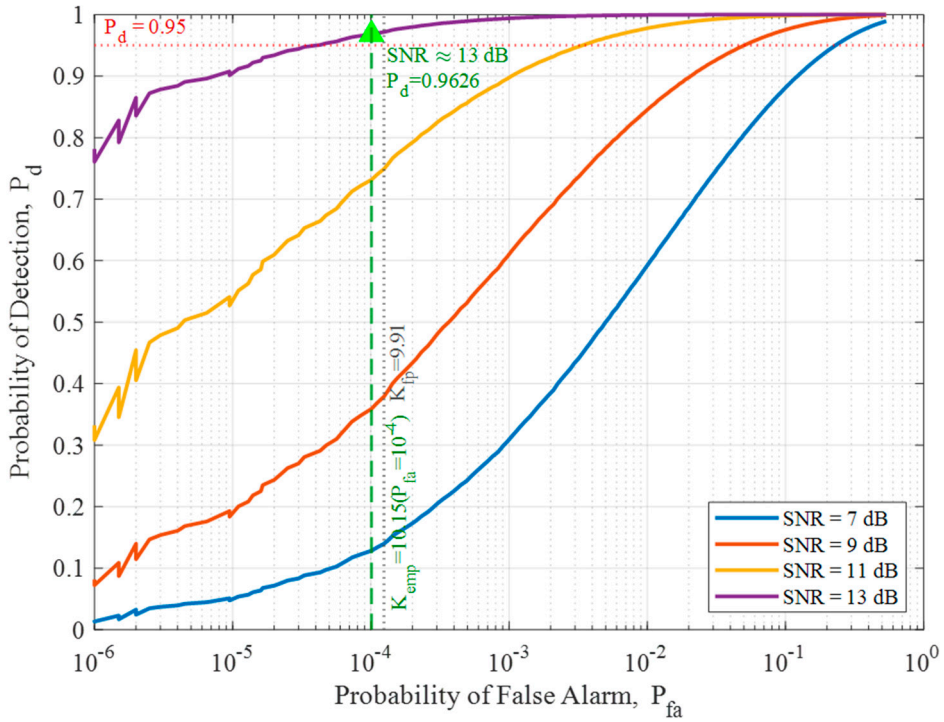

Figure 12. ROC curves of the proposed CA-CFAR detector ($N_{total} = 64$, $G = 12$) for SNR = 7, 9, 11, 13 dB under the Swerling 0 target model. Fixed-point Monte Carlo simulation ($M = 2 \times 106$ trials, hardware exact

 doi:10.20944/preprints202605.1524.v1

arithmetic). The green dashed line marks the calibrated threshold K_{emp} ($P_{fa} = 10^{-4}$); the grey dotted line shows the design value $K_{fp} = 9.9063$. The green triangle (△) marks the validated operating point at K_{emp}, SNR ≈ 13 dB (−83.16 dBm; noise floor −96 dBm).

Having validated the detection performance of the FPGA CA-CFAR stage at both the analytically derived K_{fp} and empirically calibrated K_{emp} thresholds, we now evaluate the end-to-end system latency of the proposed GPU-accelerated processing pipeline. Since the proposed multi-SDR based distributed RF surveillance system must process large-scale wideband spectrum data collected from multiple nodes in real time, minimizing the system's end-to-end latency is essential. To verify end-to-end latency, the computational latency of conventional CPU-only processing was quantitatively compared with that of the proposed GPU-accelerated pipeline. Memory copies between the CPU and GPU were performed using pinned memory and asynchronous transfers to minimize latency. Figure 13 shows processing latency as a function of the number of 1024-point PDUs, representing the cumulative received data volume from multiple nodes. The host PC was equipped with an Intel i7-12700, 16 GB of RAM and an NVIDIA GeForce RTX 3050 GPU. Performance evaluation was conducted on the actual hardware pipeline, including windowing, 1024-point FFT, magnitude computation and rendering. GPU kernel compilation overhead was eliminated through GPU pre-warmup.

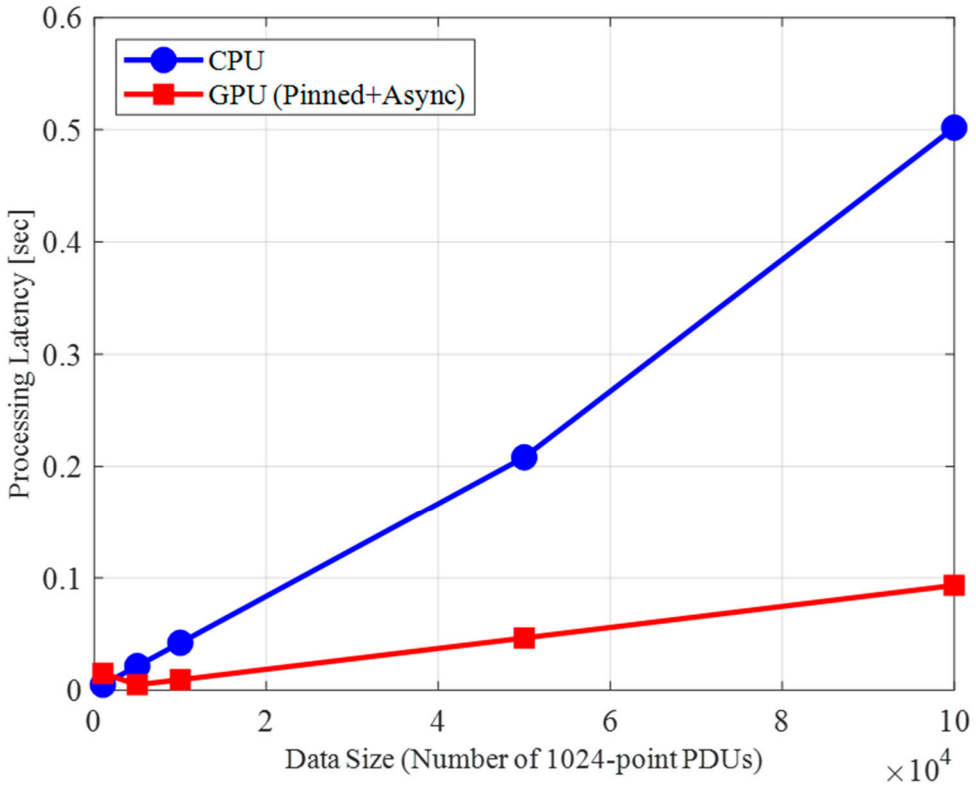

Figure 13. CPU computation time versus GPU (pinned + async) computation time as a function of the number of 1024-point PDUs. At 100,000 PDUs (~819 MB), the GPU pipeline achieves a 4.5× ~ 6.0× speedup over CPU-only processing.

4.4. GPU Acceleration and Real-Time Latency

At 1,000 PDUs (small-scale), GPU processing was approximately 2.5 × slower than CPU processing because of memory copy overhead. However, at 100,000 PDUs (819 MB of I/Q raw data), the GPU-accelerated pipeline achieved 4.5× ~ 6.0× speedup in latency, as the large data volume at this scale fully amortizes the memory copy overhead incurred at smaller PDU counts. This result was

 doi:10.20944/preprints202605.1524.v1

obtained using only a single low-cost GPU; the advantage is expected to increase further in multi-node scenarios with larger data volumes. The key factor behind this performance improvement is the pipeline structure that minimizes data movement: unlike typical GPU usage, the proposed system performs all operations from windowing and FFT through monitoring display entirely within GPU VRAM.

Real-time processing is critically important for RF signal detection systems, and the end-to-end latency of the designed system is a key performance indicator. Accordingly, the end-to-end latency from signal acquisition to visualization was measured. Latency analysis was performed by separating hardware-imposed constraints from software computation time. The experiment was conducted in 1024-sample units optimized for the minimum USB Bulk Transfer unit of 1 micro-frame (125 μs), with both PDU size and FFT size set to 1024. The measurement assumed that only the proposed system was running on the host PC, isolating pure hardware and computational latency from nondeterministic OS scheduling overhead. The GPU signal processing and rendering components in Table 3 were profiled separately using CUDA timing events in the full display-connected system; the provided duration_check.py reports these two stages as a combined measurement.

Table 3. Components of end-to-end latency for a single 1024-sample PDU.

Stage	Component	Duration (ms)
Data Acquisition	Analog Buffer Filling	0.017
Data Transfer	USB Packaging & Transfer	0.125
Signal Processing	GPU Signal Processing	0.112
Visualization	Rendering	0.116
Total	End-to-End Latency	0.370

The total end-to-end latency for one PDU is 0.370 ms. At the maximum sampling rate of 61.44 MSps, the data acquisition stage contributes only 0.017 ms. Under the USB 2.0 bandwidth constraint, the maximum lossless sampling rate is 15 MSps, at which the end-to-end latency increases to 0.421 ms—still within sub-millisecond bounds. This low-latency behavior makes the system suitable for real-time electronic warfare and RF surveillance applications, including wideband monitoring and immediate response to fast-moving targets. This sub-millisecond result is a direct consequence of the latency optimizations achieved through PDU scheduling and the GPU-accelerated processing pipeline design. The proposed system mitigates both the SDR-host transfer overhead and the internal computational bottleneck. High-performance, reliable wideband detection is thus achievable even in low-cost SDR network environments.

5. Conclusions

This paper proposed a bandwidth-efficient real-time wideband RF spectrum sensing system based on multiple low-cost SDRs and FPGA-GPU heterogeneous acceleration, with performance experimentally verified on a custom hardware testbed. The main contributions of this work can be summarized as follows.

First, a lightweight CA-CFAR IP with $O(1)$ complexity was implemented on the FPGA of low-cost SDRs to perform edge-side selective data reduction, achieving 88% reduction in transmitted data volume in a 10% duty cycle environment and reducing host-interface bandwidth demand.

Second, five ADALM-Pluto SDRs were operated in parallel via spectrum stitching using software-based sample-level synchronization without external synchronization equipment, achieving 100 MHz wideband detection coverage in the 5.8 GHz C-band.

Third, integrating PDU-based multithread scheduling with a GPU-accelerated pipeline achieved a 4.5× ~ 6.0× processing speedup over CPU-only processing and a total end-to-end latency of 370 μs for the entire system.

In the CA-CFAR detection performance evaluation, a stable detection probability exceeding 95% (Pd = 96.26%) was achieved for signal power above −83.16 dBm, demonstrating highly reliable

 doi:10.20944/preprints202605.1524.v1

detection under a minimum detectable SNR of approximately 13 dB. These results demonstrate that the proposed architecture can successfully support real-time wideband RF monitoring even on resource-constrained SDR platforms, proving its technical feasibility for practical surveillance applications such as UAV detection. Future work will focus on integrating RF signal processing with an FPGA-based deep learning inference engine on the RFSoC platform to extend the system into an intelligent real-time signal processor with pulse parameter estimation and signal classification capabilities.

Author Contributions: Conceptualization, Y.B.; methodology, B.J.; software, Y.B.; validation, H.P.; formal analysis, H.P.; investigation, Y.B.; resources, W.J.; data curation, H.L.; writing—original draft preparation, Y.B.; writing—review and editing, B.J.; visualization, Y.B.; supervision, B.J.; project administration, B.J.; funding acquisition, W.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2026-25523162). The APC was funded by the same grant.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The source code supporting the results reported in this study is openly available on GitHub at https://github.com/baeyoonsoo/spectrum_sensing_system, including the FPGA CA-CFAR IP core (Vitis HLS), GPU-accelerated processing pipeline, Monte Carlo ROC simulation, and end-to-end latency measurement script. Note that duration_check.py measures GPU signal processing and rendering as a combined stage and approximates the rendering step with a GPU buffer copy for hardware-independent execution; the individual Table 3 values were profiled using CUDA timing events in the full display-connected system. The C-simulation output data for Figure 11 (pd_vs_power_fig11.csv) are included in the GitHub repository cited above. The CPU vs. GPU latency benchmark (CPU_vs_GPU_computing_time.m, Figure 13) uses randomly generated input data without a fixed seed; absolute timing values therefore vary across runs due to data randomness and OS scheduling, but the GPU speedup over CPU consistently falls in the 4.5× ~ 6.0× range at large PDU counts. Raw RF measurement data from the physical hardware testbed are not publicly available as the hardware testbed noise environment cannot be exactly reproduced; these data are available from the corresponding author upon reasonable request.

Acknowledgments: This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2026-25523162).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Yucek, T.; Arslan, H. A survey of spectrum sensing algorithms for cognitive radio applications. IEEE Commun. Surv. Tutor. 2009, 11, 116–130.
2. Yang, Z.; Luomei, Y.; Liu, Z.; Liu, Z.; Xu, F. ZoomSpec: A physics-guided coarse-to-fine framework for wideband spectrum sensing. arXiv 2026, arXiv:2604.13568. https://doi.org/10.48550/arXiv.2604.13568
3. Axell, E.; Leus, G.; Larsson, E.G.; Poor, H.V. Spectrum sensing for cognitive radio: state-of-the-art and recent advances. IEEE Signal Process. Mag. 2012, 29, 101–116. https://doi.org/10.1109/MSP.2012.2183771
4. Xiao, Q. A conceptual architecture of cognitive electronic warfare system. In Proceedings of the 10th International Conference on Advanced Cognitive Technologies and Applications (COGNITIVE), Barcelona, Spain, 2018; pp. 38–42.
5. Guvenc, I.; Koohifar, F.; Singh, S.; Sichitiu, M.L.; Matolak, D. Detection, tracking and interdiction for amateur drones. IEEE Commun. Mag. 2018, 56, 75–81. https://doi.org/10.1109/MCOM.2018.1700455

 doi:10.20944/preprints202605.1524.v1

6. Salari, A.A.; Chen, H.; Kim, I. Wideband signal detection by employing differential sampling rates. In Proceedings of the IEEE International Conference on Communications (ICC), Ottawa, ON, Canada, 2012; pp. 4556–4560. https://doi.org/10.1109/NAECON.2011.6183094
7. Alam, S.S.; Chakma, A.; Rahman, M.H.; Bin Mofidul, R.; Alam, M.M.; Utama, I.B.K.Y.; Jang, Y.M. RF-enabled deep-learning-assisted drone detection and identification: an end-to-end approach. Sensors 2023, 23, 4202. https://doi.org/10.3390/s23094202
8. Molina-Tenorio, Y.; Prieto-Guerrero, A.; Aguilar-Gonzalez, R. Real-time implementation of multiband spectrum sensing using SDR technology. Sensors 2021, 21, 3506. https://doi.org/10.3390/s21103506
9. Nasser, A.; Al Haj Hassan, H.; Abou Chaaya, J.; Mansour, A.; Yao, K.-C. Spectrum sensing for cognitive radio: recent advances and future challenge. Sensors 2021, 21, 2408. https://doi.org/10.3390/s21072408
10. Jeon, J.; Seo, B.-S.; Ju, Y.; Lim, K.-C.; Lee, S.-J. Development of a real-time spectrum analyzer for radar pulse signals using Xilinx RFSoC. J. Korean Inst. Electromagn. Eng. Sci. 2023, 34, 69–78. https://doi.org/10.5515/KJKIEES.2023.34.1.69
11. Cha, M.; Choi, H.; Kim, S.; Moon, B.; Kim, J.; Lee, J. Development of a digital receiver for detecting radar signals. J. Korea Inst. Mil. Sci. Technol. 2019, 22, 332–340. https://doi.org/10.9766/KIMST.2019.22.3.332
12. Chiper, F.-L.; Martian, A.; Vladeanu, C.; Marghescu, I.; Craciunescu, R.; Fratu, O. Drone detection and defense systems: survey and a software-defined radio-based solution. *Sensors* 2022, *22*, 1453. https://doi.org/10.3390/s22041453
13. Ferreira, L.S.; Santos, J.F.C.M.; Carvalho, N.B. Wideband monitoring system of drone emissions based on SDR technology with RFNoC architecture. Drones 2026, 10, 117. https://doi.org/10.3390/drones10020117
14. Sharma, S.K.; Bogale, T.E.; Chatzinotas, S.; Ottersten, B.; Le, L.B.; Wang, X. Cognitive radio techniques under practical imperfections: a survey. IEEE Commun. Surv. Tutor. 2015, 17, 1858–1884. https://doi.org/10.1109/COMST.2015.2452414
15. Han, S.; Park, H.; Bae, Y.; Lee, H.; Cho, Y.-K.; Choi, W.; Kim, H.-R.; Jang, B.-J. Application of low-cost Pluto SDR as a radar target simulator. J. Korean Inst. Electromagn. Eng. Sci. 2026, 37, 117–120. https://doi.org/10.5515/KJKIEES.2026.37.1.117
16. Wu, Z.; Qaragoez, Y.; Volskiy, V.; Huangfu, J.; Ran, L.; Schreurs, D. A joint design of radar sensing, wireless power transfer, and communication based on reconfigurable software defined radio. Electronics 2022, 11, 4050. https://doi.org/10.3390/electronics11234050
17. Venter, C.J.; Grobler, H.; AlMalki, K.A. Implementation of the CA-CFAR algorithm for pulsed-Doppler radar on a GPU architecture. In Proceedings of the 2011 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT), Amman, Jordan, 2011; pp. 1–6. https://doi.org/10.1109/AEECT.2011.6132514
18. Andrews, G.S.; Shake, T.H. Real-time, GPU-accelerated processing of digital radar signal data. In Proceedings of the 2012 IEEE Radar Conference, Atlanta, GA, USA, 2012; pp. 244–249.
19. Dong, H.; Zheng, C.; Tian, W. Research on parallel architecture design of radar real-time signal processing based on CPU-GPU heterogeneous platform. IET Conf. Proc. 2020, 52–57. https://doi.org/10.1049/icp.2021.0781
20. Liu, G.; Yue, N.; Wang, S. A GPU-based real-time processing system for frequency division multiple-input-multiple-output radar. IET Radar Sonar Navig. 2023, 17, 1524–1537. https://doi.org/10.1049/rsn2.12439
21. Zhao, X.; Liu, P.; Wang, B.; Jin, Y. GPU-accelerated signal processing for passive bistatic radar. Remote Sens. 2023, 15, 5421.
22. Rupniewski, M.; Mazurek, G.; Gambrych, J.; Nalecz, M.; Karolewski, R. A real-time embedded heterogeneous GPU/FPGA parallel system for radar signal processing. In Proceedings of the 2016 IEEE International Conferences on Ubiquitous Intelligence and Computing (UIC/ATC/ScalCom), Toulouse, France, 2016. https://doi.org/10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0182
23. Richards, M.A. Fundamentals of Radar Signal Processing, 2nd ed.; McGraw-Hill Education: New York, NY, USA, 2014.
24. Urkowitz, H. Energy detection of unknown deterministic signals. Proc. IEEE 1967, 55, 523–531. https://doi.org/10.1109/PROC.1967.5573

 doi:10.20944/preprints202605.1524.v1

25. Rohling, H. Radar CFAR thresholding in clutter and multiple target situations. IEEE Trans. Aerosp. Electron. Syst. 1983, AES-19, 608–621. https://doi.org/10.1109/TAES.1983.309350
26. Sim, Y.; Heo, J.; Jung, Y.; Lee, S.; Jung, Y. FPGA implementation of efficient CFAR algorithm for radar systems. Sensors 2023, 23, 954. https://doi.org/10.3390/s23020954.
27. Wang, Y.; Liu, Q.; Fathy, A.E. CW and pulse-Doppler radar processing based on FPGA for human sensing applications. IEEE Trans. Geosci. Remote Sens. 2013, 51, 3097–3107. https://doi.org/10.1109/TGRS.2012.2217975
28. Uvaydov, D.; D'Oro, S.; Restuccia, F.; Melodia, T. Stitching the spectrum: semantic spectrum segmentation with wideband signal stitching. In Proceedings of the IEEE INFOCOM 2024, Vancouver, BC, Canada, 2024; pp. 1–10. https://doi.org/10.48550/arXiv.2402.03465
29. Perdana, R.S.; Sitohang, B.; Suksmono, A.B. A survey of graphics processing unit (GPU) utilization for radar signal and data processing system. In Proceedings of the 2017 6th International Conference on Electrical Engineering and Informatics (ICEEI), Langkawi, Malaysia, 25–27 November 2017. https://doi.org/10.1109/ICEEI.2017.8312430
30. Fisne, A.; Bahceci, M.U.; Dursun, M.; Aydogmus, S.; Cetintepe, C. High-Level Synthesis based radar hardware implementation and performance comparison. In Proceedings of the 2022 Innovations in Intelligent Systems and Applications Conference (ASYU), Antalya, Turkey, 2022; pp. 1–6. https://doi.org/10.1109/ASYU56188.2022.9925373