Journal of VLSI circuits and systems
RESEARCH ARTICLE
Journal of VLSI Circuits and Systems, ISSN: 2582-1458 Vol. 8, No. 1, 2026 (pp. 42-56)
WWW.VLSIJOURNAL.COM

Design and Verification of FPGA-based Range Processing, Peak Detection and Doppler Processing for FMCW-RADAR

Suganthi K.[1], Vijayakumar Ponnusamy[1]*, Harikrishnan Krishnakumar[1], Prasanna Venkatesh G.[1], Pragadeshwaran V.[1], D. Malathi[2], M. Vinodhini[3], Nemanja Zdravkovic[4]

[1]Department of ECE, Faculty of Engineering & Technology, SRM Institute of Science & Technology, Kattankulathur, Chengalpattu, Tamil Nadu, India

[2]Department of Electronics and Communication Engineering, Kongu Engineering College, Erode, Tamil Nadu, India

[3]Department of Electronics and Communication Engineering, Amrita Vishwa Vidyapeetham, Bengaluru

[4]Faculty of Information Technology, Belgrade Metropolitan University, Belgrade, Serbia

KEYWORDS:
FPGA,
FFT,
AXI,
FMCW Radar,
Doppler,
Target detection

ARTICLE HISTORY:
Received: 13.02.2026
Revised: 12.03.2026
Accepted: 18.03.2026

DOI:
https://doi.org/10.31838/jvcs/08.01.05

Abstract

Modern RADAR systems produce a large amount of data that must be processed quickly and accurately for reliable target detection. Software-based processing methods often struggle to meet real-time requirements due to high latency and computational overhead. This work presents a fully streaming, Field Programmable Gate Array (FPGA)-based peak detection architecture tightly integrated within the Range-Doppler processing pipeline. Continuous magnitude data from the Range and Doppler Fast Fourier Transform (FFT) stages is transferred using an Advanced eXtensible Interface (AXI)-Stream interface, enabling seamless integration between Xilinx FFT IP cores and a custom peak detection accelerator without intermediate memory storage. Unlike conventional radar systems that construct a two-dimensional Range-Doppler map followed by explicit scanning, the proposed approach performs detection on a linearized data stream. This eliminates the need for complex 2D search logic, reduces control overhead, and enables low-latency, real-time operation suitable for practical radar deployments. Additionally, index-aware detection logic propagates range and Doppler indices alongside magnitude data, enabling direct extraction of target coordinates without post-processing. The modular AXI-Stream architecture ensures reusability, timing-closed operation, and reduced FPGA resource utilization compared to reference designs. The proposed system emphasizes FPGA-focused architectural optimization rather than algorithm-level modification, distinguishing it from prior software-oriented or hybrid detection approaches. The complete processing chain achieves an end-to-end latency of approximately **1.1 ms** for a full frame comprising 64 chirps, while sustaining a continuous input data rate exceeding **200 MSamples/s**. The design was synthesized and simulated using the Xilinx Vivado design tool and implemented on a Zynq-7000-based ZC702 evaluation board, demonstrating efficient FPGA resource utilization of approximately 35% LUTs, 28% flip-flops, 42% BRAM, and 6 DSP slices. Simulation results confirm correct FFT operation, reliable buffering, and accurate peak detection across varying signal-to-noise ratios. The proposed architecture offers low latency, high throughput, and efficient hardware utilization, making it well-suited for practical real-time radar signal processing applications. Overall, the proposed FPGA-based solution demonstrates low latency, efficient resource usage, and reliable real-time performance, making it suitable for practical Frequency Modulated Continuous Wave (FMCW)-RADAR signal processing applications.

Authors' e-mail ID: suganthk@srmist.edu.in, vijayakp@srmist.edu.in, hk2125@srmist.edu.in, pv6810@srmist.edu.in, pg5842@srmist.edu.in, malathy@kongu.ac.in, m_vinodhini@blr.amrita.edu, nemanja.zdravkovic@metropolitan.ac.rs

Authors' ORCID ID: 0000-0002-0374-4190; 0000-0002-3929-8495; 0009-0005-6565-0244; 0009-0006-6559-3718; 0009-0009-5107-0651; 0000-0003-3479-5991; 0000-0002-3718-2122; 0000-0002-2631-6308

How to cite this article: Suganthi K. et al., Design and Verification of FPGA-based Range Processing, Peak Detection and Doppler Processing for FMCW-RADAR, Journal of VLSI Circuits and System, Vol. 8, No. 1, 2026 (pp. 42-56)

Introduction

Radar systems generate a large volume of signal data, which must be processed efficiently to detect targets in real time. After the Fast Fourier Transform (FFT) process is completed and the FFT outputs are obtained for processing the target's information and parameters, it is necessary to use peak detection and choose the outputs with a higher value than that of the average threshold set, so that the lower value outputs get discarded. To overcome these limitations, this work focuses on the design and verification of Field Programmable Gate Array (FPGA)-based RADAR signal processing system that includes a custom streaming peak detection accelerator. The proposed system performs both range and Doppler processing [1] using Xilinx FFT IP cores configured in pipelined streaming mode. A 1024-point Range FFT is applied to convert time-domain ADC samples into frequency-domain range information, followed by a 64-point Doppler FFT across multiple chirps to estimate target velocity. Advanced eXtensible Interface (AXI)-Stream interfaces are used throughout the processing chain to maintain proper frame synchronization and support continuous, deterministic data flow. Intermediate buffering using on-chip memory allows smooth data transfer between FFT processing stages.

There have been many software approaches for this application, but none of them could come up with a cost-effective and efficient solution due to the following reasons. Software running on a CPU processes data sequentially, which adds a significant delay in the processing of target detection and other parameters. The second reason is that the FFT output bins are checked one after another, and so the identification of the bin with the highest value will take longer. The final reason is that software can be used for small FFT-sized bins,[1,2] but when it comes to the large-scale bins, that is, in hundreds or thousands, then the delay is significant. Latency[4] is also one of the main factors when it comes to radar signal processing; as we know that to reduce the chance of error as much as possible, the decisions need to be taken on real, but if we use a software in this place, it may cause a variable or low latency due to one of these reasons: OS scheduling, Interrupts, and Memory access delays.

For all the reasons specified above, the use of software-based processing is not suitable for real-time radar applications. Hence, the design approach was shifted toward a custom hardware accelerator implemented on an FPGA[2] using an AXI-based streaming interface, enabling efficient and deterministic processing of FFT output data.

Methodology

The proposed system implements a complete radar signal processing chain in hardware. The stages here include range FFT, range FFT buffer, Doppler FFT, magnitude computation, and a custom-built peak detection accelerator. The range FFT Buffer exists only for reorganizing the range FFT output bins across the chirps before they are sent into the Doppler FFT. The system uses an AXI-Stream-based pipeline architecture, meaning that one sample is processed per clock, and this ensures a continuous flow of data. The system architecture enables continuous processing of samples per chirp and efficient FPGA resource utilization[3]. The architecture avoids constructing and storing the full two-dimensional Range–Doppler map in memory by performing peak detection directly on streaming FFT magnitude data.

The proposed system implements radar signal processing[4] on an FPGA using a hardware-accelerated and streaming-based approach. Time-domain ADC samples from each radar chirp are first processed using a 1024-point Range FFT[7] implemented with the Xilinx FFT IP core. The FFT operates in pipelined streaming mode using an AXI-Stream interface, which enables continuous data transfer with proper frame synchronization using the tvalid, tready, and tlast signals. A custom hardware peak detection accelerator is developed to process FFT magnitude data in real time.

The output of the Range FFT is stored in on-chip memory using address counters that increment sequentially for each valid FFT output. This buffered data is reorganized and provided to a separate Doppler FFT stage, which performs a 64-point FFT across multiple chirps for each range bin to estimate target velocity. Similar to the Range FFT, the Doppler FFT uses an independently configured FFT IP core and streams data efficiently through the processing pipeline.

After Doppler processing[5], the FFT magnitude data is sent to a custom peak detection accelerator. The accelerator uses a programmable threshold register and local maximum detection logic to identify significant peaks in real time. Detected peaks, along with their range and Doppler indices, are stored in an internal FIFO, allowing continuous streaming without stalling and enabling efficient downstream processing.

The accelerator uses a programmable threshold and local maximum detection logic to accurately identify significant peaks while suppressing noise and low-magnitude bins. Detected peaks, along with their associated range and Doppler indices, are stored in an internal FIFO to ensure uninterrupted data streaming.[3]

Radar Signal Model

The equations used in the architecture are given below as follows

$$S(t) = A\cos\left(2\pi fct + \left(\frac{B}{2T}\right)t^2\right) \quad (1)$$

The above equation describes the Frequency Modulated Continuous Wave (FMCW) radar transmitted signal in which the frequency increases linearly with time. The parameters in the equation include fc, called the carrier frequency, and T, known as the chirp duration. This waveform is the basis of FMCW radar signal processing[6,10].

$$fb = \frac{(2 * B * R)}{(c * T)} \quad (2)$$

Equation 2, given above, represents the Beat frequency and range estimation. The transmitted signal is always mixed with a received echo, which produces a beat frequency proportional to target distance. This beat frequency is then processed using FFT to estimate range. R in the equation represent Range and C represent the speed of light.

$$X[k] = \sum_{\{n=0\}}^{\{N-1\}} x[n]e^{\{-j2\pi kn-N\}} \quad (3)$$

The Range FFT converts the time-domain ADC samples obtained from each radar chirp into frequency-domain bins. Each frequency bin corresponds to a specific target range, allowing the system to determine the distance of objects from the radar.

$$X[k] = \sum_{\{n=0\}}^{\{N-1\}x[n]e^{\{-j2\pi kn-N\}}} \quad (4)$$

The Doppler FFT is applied across multiple chirps for each range bin. This operation extracts the Doppler frequency shift caused by moving targets, enabling estimation of target velocity.

$$|X(k,m)| > T \quad (5)$$

$$|X(k,m)| > |X(k-1,m)| \text{ and } |X(k,m)| > |X(k+1,m)| \quad (6)$$

The above equation describes the FFT magnitude for a specific range index k and Doppler index m, and T in the equation represents the predefined threshold. Peak detection identifies significant targets by comparing the magnitude of the FFT output with a predefined threshold. Only bins whose magnitude exceeds the threshold are considered valid detections, while lower-magnitude bins are treated as noise.

System Overview

The Accelerator comes mainly into play after the FFT processes are completed and is typically located at the end stages of the Radar signal processing[7] unit. Since the FFT magnitude data is available once the input data is transformed from the time domain to the frequency domain, the accelerator then processes this FFT output. Once in the accelerator processes, the peak detection stage begins, and the highest output from the surrounding FFT bins is chosen, and then the values, such as the location, range index, and Doppler information, are directly stored in a FIFO register.

FFT output data is provided to the Accelerator through an AXI-Stream interface. Although the data is sent sequentially to the accelerator, the internal architecture employs a pipeline parallelism allowing multiple stages of the detection logic to operate concurrently on successive samples. The accelerator also has a Threshold register so that it can filter out the input with lower FFT magnitudes, and also a local maximum detection, which identifies the significant FFT peaks in real time. After these processes, the detected Peaks are stored in the FIFO.

Architecture of the Peak Detection Accelerator

The accelerator is mainly divided into two main parts: AXI-STREAM INTERFACE, PEAK DETECTION CORE. In this section, we will have a clear understanding of the functions of these parts, starting with the AXI-Stream and then the Peak detection core. The main function of AXI-Stream is used for input data transfer, and incoming data consists of the following data: the FFT magnitude, range index, and Doppler index. The AXI accepts the data and then unpacks the fields and forwards them to the Peak detection core in a sequential manner. The data is streamed continuously, so there is no stalling, and one sample is taken per clock cycle.

The peak detection core block as shown in Figure 1, is the heart of the design and the critical part in this

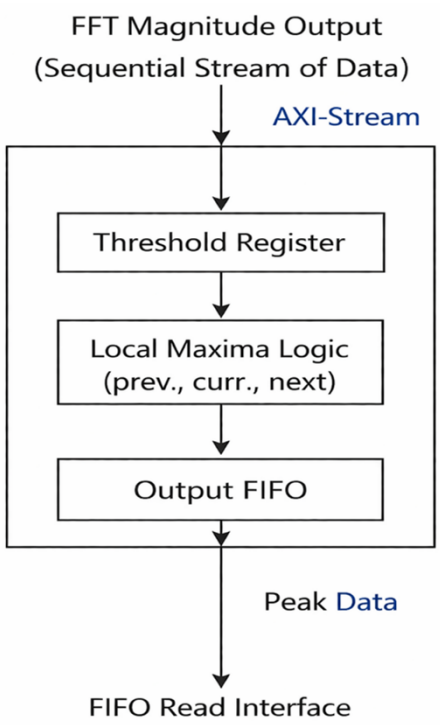

Fig. 1: Block Diagram of the Streaming Peak Detection IP for Radar FFT Processing.

accelerator the core consists of a Threshold register, Local maxima detection logic, pipeline registers, FIFO register all these section will be discussed below sequentially lets start with the threshold register the main function of this register is to load the threshold value set by the user the value is loaded in the next clock cycle and can be changed at any instant of time if required this register is therefore a programmable threshold register then once the threshold value is set the incoming data is compared with respect to the set threshold value and if only it surpasses the threshold then it is sent to the next step, This process helps in suppress noise and insignificant FFT bins.

In this work, a fixed threshold is used for peak detection to simplify hardware implementation. However, in the case of practical radar systems, adaptive thresholding techniques such as the Constant False Alarm Rate (CFAR) are commonly used. These methods help to dynamically adjust the threshold based on the noise level and surrounding signal statistics[8], which improves detection reliability under varying noise conditions.

The next step is the Local maxima detection logic this process uses a sliding window which uses three samples namely Previous, current, next For the peak to be detected successfully the following steps are followed first the current sample must exceed the threshold value set and then these current samples are then checked with the previous samples and made sure that is greater than the samples from the neighbouring bins so that there is no error in this process. This logic is implemented in a pipelined manner in order to minimize the error in case of delay or other aspects.

The pipeline register aligns the data that is being sent in a sequential order, which minimizes the occurrence of errors. It also helps in enabling the pipeline parallelism, which was mentioned in the above section. It is also responsible for a continuous throughput.

The FIFO basically acts like a memory. The main function of FIFO is to store the detected peaks. The FIFO stores the detected peak data in the following format: Magnitude, Range index, Doppler index. The FIFO decouples the detection logic from output access. The FIFO also helps in continuous streaming of data and prevents input stalling. The modular architecture[9] separates the streaming interface from the detection logic, improving reusability and scalability of the design.

Fast Fourier Transform Process

In this section, we will be viewing in detail the FFT, about its purpose and application. The FFT process is needed in this paper because the custom-designed peak detection accelerator works only on the FFT inputs. An understanding of FFT is necessary in this case.

The main purpose or operation is to convert the discrete-time signal from the time domain to the frequency domain. FFT reveals the frequency components that are present in a sampled signal. It represents the input signal as a sum of complex sinusoids and also helps in analysing the signal energy distribution across frequency bins throughout the output.

In a practical radar signal processing system, which is used in a real-time application, the input signal is often multiplied by a windowing function, which helps in reducing the spectral leakage or helps it to keep it to a minimum. The windowed signal can be given

$$x_{w[n]} = x[n] \cdot w[n] \tag{7}$$

The most commonly used windowing function is the Hanning window, which is defined below

$$w[n] = 0.5(1 - \cos(2\pi n - (N-1))) \tag{8}$$

However, in this work, windowing is not explicitly implemented as the work's major focus is on hardware

architecture and the streaming peak detection accelerator. But the addition of windowing can further improve frequency domain accuracy in real-time deployments.

The application of FFT is heavily seen in the digital signal processing domain. This is because direct computation of DFT (Discrete Fourier Transform) is computationally expensive, as it drains a huge amount of resources and time. The FFT here plays a crucial role, where it acts as an efficient algorithm to compute the DFT. The FFT also reduces the computational complexity from $O(N^2)$ to $O(N \log_2 N)$. Because of this, we can make real-time signal processing feasible in hardware and embedded systems, through which we can make decisions in time.

Now let us discuss about the importance of FFT in radar signal processing and its application. The FFT is applied to sampled radar return signals after ADC. This converts the time domain signal to frequency domain signals or to specific range frequency bins. Each of the range bins corresponds to a specific target range or location. The use of FFT allows the separation of multiple targets based on distance or other parameters if necessary. The FFT makes up the first major signal processing stage in FMCW radar systems.

The FFT size N usually determines the frequency and the range resolution. This also means that a larger FFT provides a finer resolution, but also consumes more hardware resources for the transformation from the time domain to the frequency domain. In radar systems, the FFT size is selected based on the maximum usable range and resolution requirement of the particular system.

FFT IP Core Overview

There are two major computations for FFT that happen in this paper, one is the Range FFT and the other is the Doppler FFT. Both of these processes use an FFT IP core version 9.1 from Vivado, which is a custom-built IP configured slightly differently for both processes. The IP core used in the process provides a highly optimized and hardware-efficient implementation of the FFT. The IP core also supports AXI-Stream-based streaming interfaces, making it very suitable for real-time radar signal processing using an FPGA. The IP core internally computes the DFT using FFT algorithms while also handling scaling, truncation and other aspects.

Now, let us discuss the configuration of the Range FFT IP core. The number of channels used in the FFT IP is set to 1 because the radar signal processing chain operates only on a single stream of ADC samples per chirp. Using multichannel FFT would increase hardware complexity and FPGA resource allocation unnecessarily. As for the transform length, the configured value is 1024 because it provides sufficient range resolution for a radar system and also matches the number of ADC samples collected per chirp. This is also one of the commonly used FFT sizes in FMCW radar range processing.

Moving forward to the architecture choice, the option selected was pipelined streaming I/O, which supports continuous and high-throughput processing of data. The input samples can be streamed continuously without waiting for the frame to be stored, which is very suitable for real-time radar systems where latency and throughput are very important. This also integrates easily with the AXI-Stream-based system architecture. The target clock frequency was set to 250 MHz. This value was selected to meet the high throughput requirements of real-time FFT processing and also allows the pipeline to process one sample per clock cycle.

The data format was set to fixed-point format because it is more resource-efficient than floating-point on an FPGA. It significantly reduces DSP, LUT, and power consumption while still providing adequate precision and resolution for radar signal processing when combined with proper scaling. The rounding mode selected was truncation mode because it is a simple and hardware-efficient rounding method and minimizes the logic overhead compared to rounding to nearest.

For the precision settings, both the input data width and the phase factor width were set to 16 bits each because 16-bit precision offers a good balance between accuracy and hardware cost. It is also a commonly used word length in FPGA-based DSP implementations. Now, let us discuss the interface and ports of the FFT IP core. The input interfaces of the FFT IP core include **S_AXIS_DATA** and **S_AXIS_CONFIG**. The function of **S_AXIS_DATA** is to provide an AXI-Stream interface for FFT input samples. This interface carries the complex input data and includes signals such as **tvalid**, **tready**, and **tlast**. The **tlast** signal marks the end of one FFT frame, which is 1024 samples in the case of Range FFT and 64 samples in the case of Doppler FFT.

The event and status signals of the FFT IP core are given below with their functions

- event_frame_started: Indicates start of FFT frame processing

- event_tlast_unexpected: Detects incorrect frame termination
- event_tlast_missing: Indicates missing tlast signal
- event_FFT_overflow: Signals numerical overflow inside the FFT core
- event_status_channel_halt: Indicates stalled status channel
- event_data_in_channel_halt: Indicates input backpressure
- event_data_out_channel_halt: Indicates output backpressure

These signals are primarily used for debugging, verification and system monitoring during simulation for verifying the working of the FFT IP core.

Range FFT Processing

The range processing stage forms the first major signal processing block in the radar system[11,12]. In this project, a 1024-point Range FFT is implemented using the Xilinx FFT IP core provided in Vivado. This FFT operates on the time-domain ADC samples obtained from a single radar chirp and converts them into frequency-domain data, where each output bin corresponds to a specific target range.

The FFT IP core is configured in pipelined streaming mode, which allows continuous input of samples without requiring the entire data frame to be stored beforehand. This architecture is well-suited for real-time radar systems, as it supports high throughput and low latency operation.[8] Complex ADC samples[5] are streamed into the FFT core using an AXI-Stream interface, ensuring compatibility with other processing blocks in the design.

The frame synchronization in this paper is achieved by using the AXI-Stream interface, which is used in both the peak detection accelerator and the FFT IP. In the case of the FFT IP, the AXI-Stream interface is implemented internally by the IP itself. The main function of the AXI-Stream interface is to perform proper handshake between two modules so that the correct or valid inputs are passed to the module for processing and obtaining the output.[3] The AXI-Stream interface has two main signals responsible for this handshake process between two modules in the system. These signals are **tvalid** and **tready**, which control the valid data transfer between the modules. There also exists the **tlast** signal in the AXI-Stream interface. The main function of this signal is to indicate the end of one complete FFT frame or chirp. As we already know, each frame consists of 1024 samples corresponding to one chirp.[13] The hardware resource usage is further reduced by using internal scaling and truncation mode that was set during the configuration of the FFT IP core. This type of configuration is highly compatible with radar signal processing applications.

The functionality of the Range FFT block has been verified successfully, and the simulation result has been attached in Figure 2, for reference, and RTL is shown in Figure 3. The analysis of the waveform confirms the conversion of time-domain samples into the frequency domain.

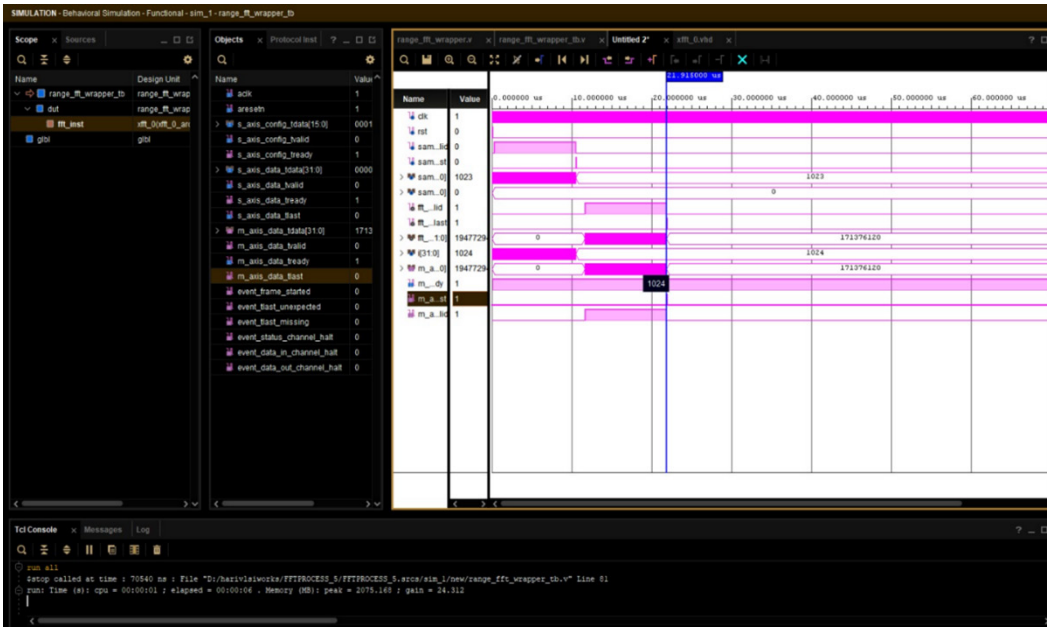

Fig. 2: Simulation of Range FFT Verified.

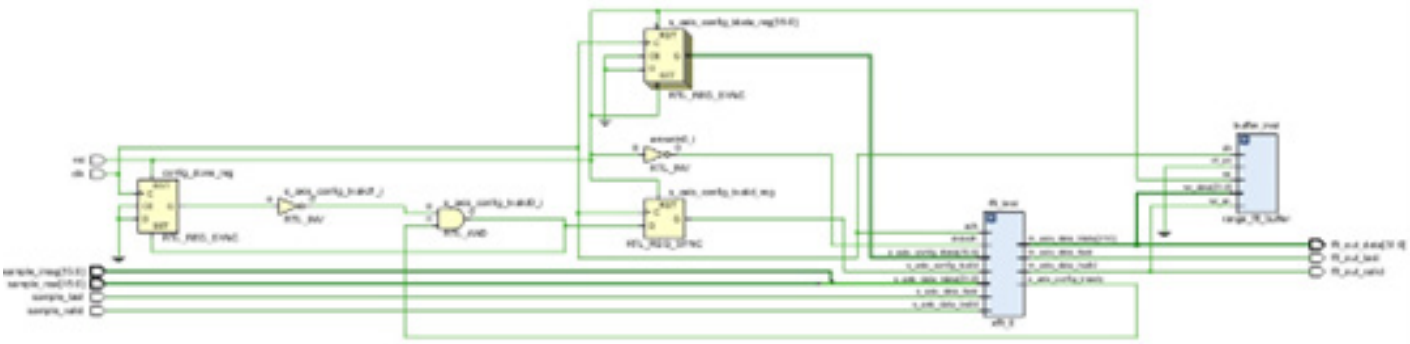

Fig. 3: RTL Design of Range FFT.

Implementation

Range FFT Output Buffering

Once the output is successfully generated by the Range FFT IP using the FFT IP core, as discussed in the above section, the output of the Range FFT is temporarily stored in the RAM. This helps in supporting the further FFT processing stage that comes after the Range FFT.[12] The FFT output bins are written into the RAM[8] only if the FFT output valid signal is asserted, which ensures that only correct or valid data is stored in the RAM.

The internal address registers have also been used to manage the memory write operations and the address allocation to the range bins, also referred to as the FFT output of the Range FFT. The address counters increment sequentially per clock cycle for each FFT output or range bin.[14] Because of this process, we are able to implement an ordered storage of frequency-domain data for each range bin in the output. The buffering stage acts as an interface between the Range FFT and the Doppler FFT.

The buffer always tries to arrange the data in such a manner that it is easier for the Doppler FFT[14,39] to process the data, as the number of range bins will be too large once the FFT process is completed by the range FFT. Below, we have attached the simulation result in Figure 3, where the memory behavior has been simulated and verified by means of an RTL schematic in Figure 5.

Doppler FFT Processing

Once the range processing and buffering are completed, the next step in the process is the Doppler FFT. The range buffer, as mentioned in the above section, performs the simple function of rearranging the inputs and then sending them for processing at the Doppler stage.

In Doppler FFT, as mentioned earlier, a 64-point FFT is used, and the FFT IP core[13] used for this process is separate from the Range FFT IP core for maintaining the functionality of the FFT process and the overall design integration with the peak detection accelerator.

The Doppler FFT process is performed to estimate the target velocity. The Doppler FFT is applied across all the chirps for a particular range bin. Since this operation is carried out across multiple chirps, this area of operation falls under the slow-time dimension of radar data.[9] The IP used for the Doppler FFT process is similar to the Range FFT IP, but with slight changes made for efficient operation.

The Doppler FFT process is performed to estimate the target velocity. The Doppler FFT is applied across all the chirps for a particular range bin. Since this operation is carried out across multiple chirps, this area of operation falls under the slow-time dimension of radar data.[9] The IP used for the Doppler FFT process is similar to the Range FFT IP, but with slight changes made for efficient operation.

Unlike the Range FFT, the Doppler FFT processes the data across successive chirps for a fixed range bin rather than processing time-domain samples from a single chirp as in the Range FFT, which uses 1024 samples.[10] The output of the Doppler FFT consists of Doppler frequency bins, which are directly linked to the target velocity in real-time scenarios. Using these outputs, we can further generate the Range–Doppler map, which provides a two-dimensional representation of target velocity and range. This approach simplifies the system integration, verification, and overall processing.

Magnitude Computation Module

After the Doppler FFT processing, the output data consists of complex frequency-domain samples

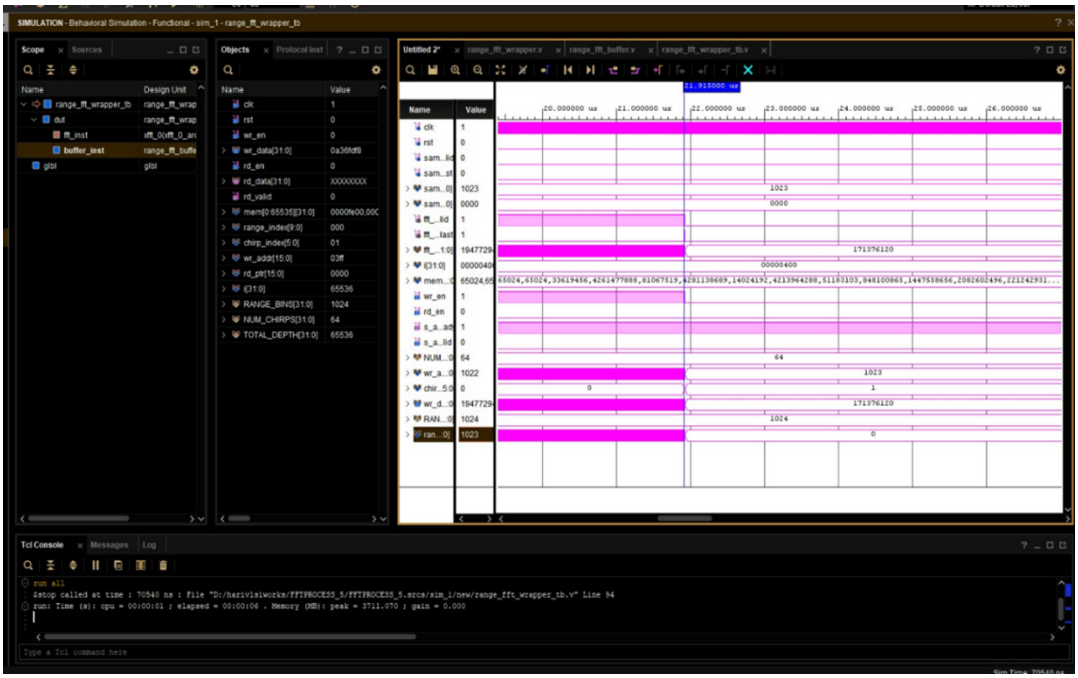

Fig. 4: Simulation of Range FFT with RAM is Verified.

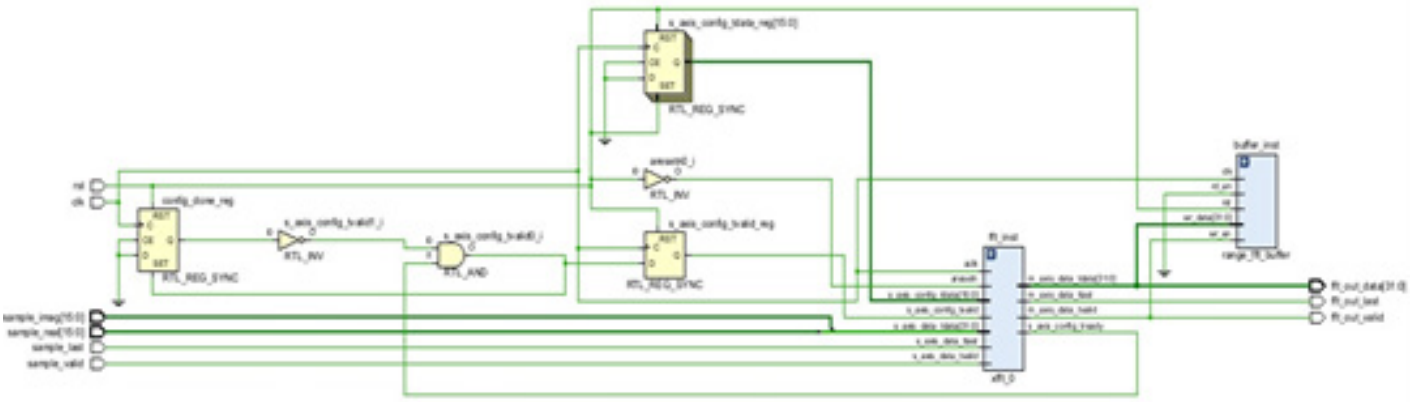

Fig. 5: RTL Design of Range FFT with RAM.

represented in fixed-point format. In this representation, the lower 16 bits correspond to the real component and the upper 16 bits correspond to the imaginary component. Since the peak detection stage relies on signal strength rather than phase information, a magnitude computation stage is required before performing detection.[40-42]

In this work, the signal magnitude is computed using the magnitude-squared formulation:

$$[X]^2 = \mathrm{Re}^2 + \mathrm{Im}^2 \quad (9)$$

Instead of calculating the true magnitude,

$$[X] = (\mathrm{Re}^2 + \mathrm{Im}^2)^{1/2} \quad (10)$$

The square-root operation is intentionally avoided. Because the Square-root computation on FPGA devices is relatively very expensive and often requires additional hardware resources allocation, such as DSP blocks, CORDIC units, or lookup tables, based on implementations. For peak detection purposes, only the relative magnitude comparison between bins is required. Since the square root function is usually monotonic, comparing magnitude-squared values nearly produces the same detection outcome as comparing the true magnitudes. Therefore, using the magnitude-squared method, we can significantly reduce hardware complexity while maintaining detection accuracy.

The magnitude computation module is implemented as a **two-stage pipelined architecture**[16,17] to support high-throughput processing.

Stage 1 – Squaring: The signed 16-bit real and imaginary components are independently squared using hardware multipliers. The resulting values are stored in 32-bit registers. Control signals, such as *valid* and *last*, are also registered to maintain proper pipeline alignment.

Stage 2 – Addition: The squared real and imaginary values are summed to produce the final 32-bit magnitude-squared output. The control signals propagate through the pipeline in synchronization with the data path.

This pipelined structure allows the magnitude result to be produced **once per clock cycle after the initial pipeline latency**, ensuring continuous streaming operation. The use of fixed-point arithmetic enables efficient FPGA implementation[15] while maintaining controlled resource utilization.

Simulation Results

Simulation results, as shown in Figures 6 and 7, confirm correct operation of the Range FFT, demonstrating accurate conversion of time-domain samples into frequency-domain range bins. The Range FFT output buffering stage shows proper memory write behavior, with valid data being stored sequentially and expected zero or near-zero bins appearing due to fixed-point truncation.[17]

The Doppler FFT results verify correct velocity-domain processing across multiple chirps for each range bin. Output buffering for the Doppler FFT[18] operates as expected, with stable data storage and correct address progression, ensuring reliable construction of the Range-Doppler representation.

The peak detection accelerator[19,20] successfully identifies valid FFT peaks above the programmed threshold while suppressing noise and insignificant bins. FIFO operation is verified to be correct, and timing analysis shows no violations, confirming that the design meets real-time performance requirements with efficient resource utilization.

Simulation and Verification of Peak Detection Accelerator

The simulation was performed successfully, and the functional correctness was verified to ensure that the peak detection core is working in the correct manner and also to validate the FIFO output behavior.[21] In the testbench for the simulation, FFT-like magnitude signal values were applied in sequential order. The test data that was given included values below the threshold, a peak above the threshold, and their corresponding Doppler and range indices provided simultaneously. During the verification of peak detection, only the values above the threshold were detected, and the local

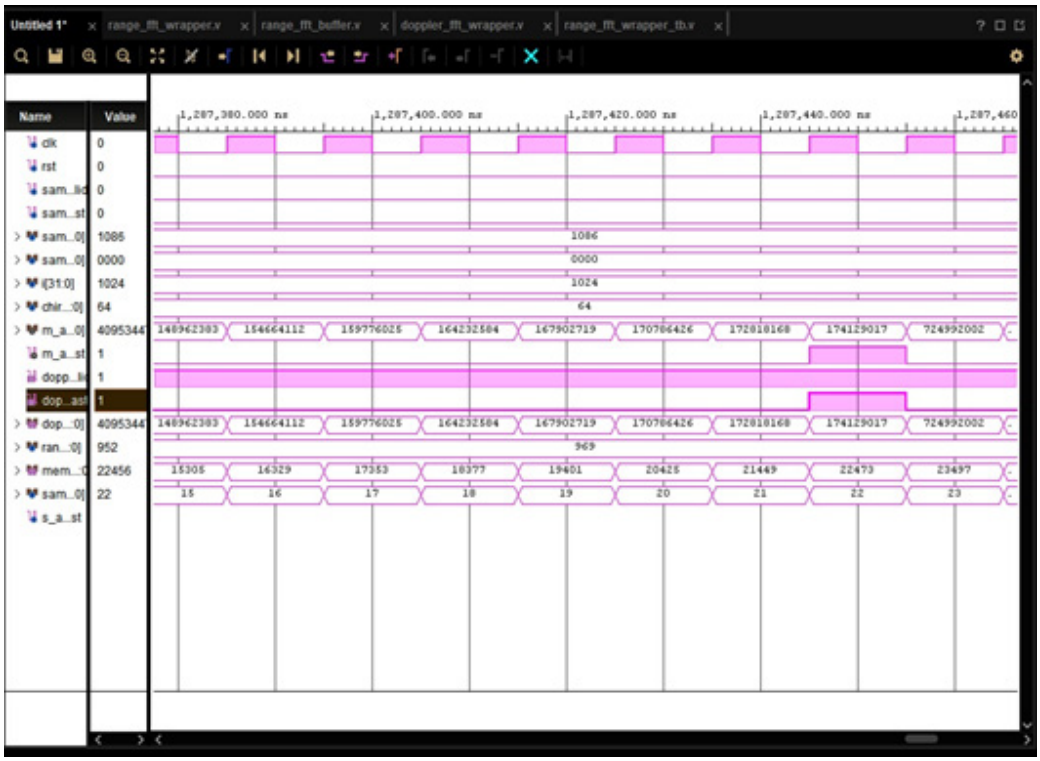

Fig. 6: Simulation of Doppler FFT Verified.

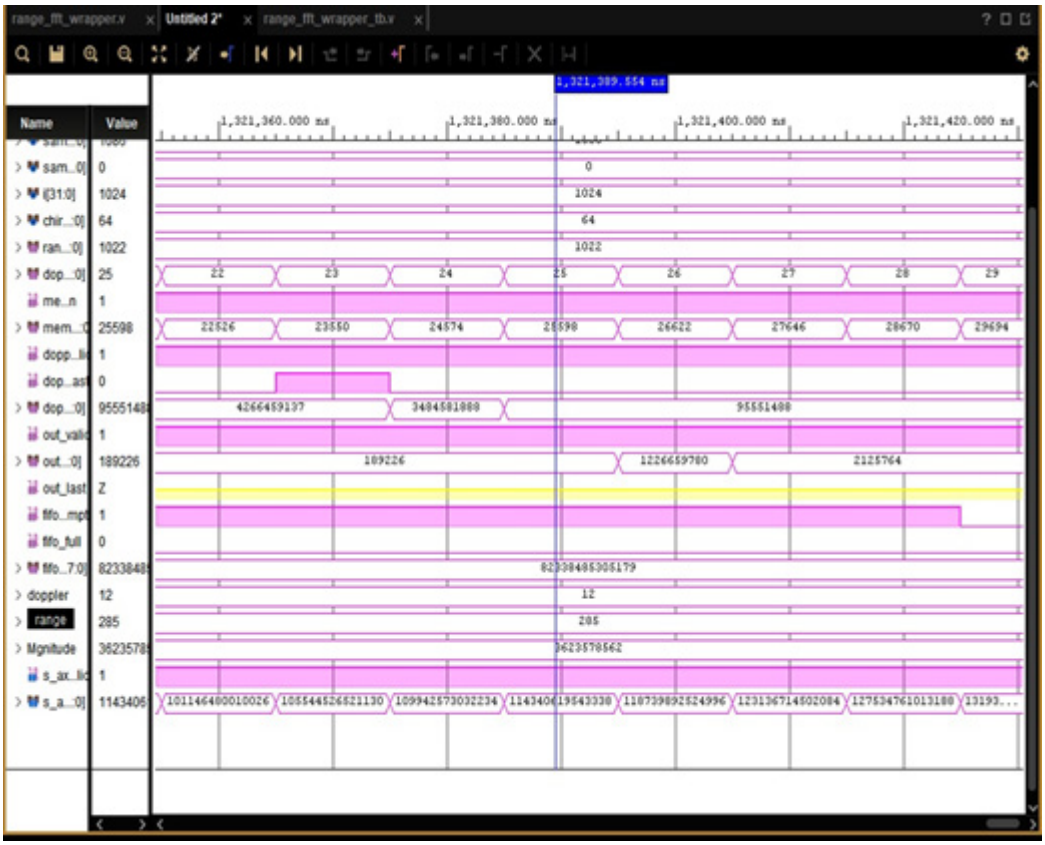

Fig. 7: Simulation of Magnitude Computation Verified.

maxima condition was applied correctly to identify the bin with the highest magnitude when compared to the neighbouring bins. The Doppler[21,22] and range indices[23,24] of the selected bins were also verified. In the FIFO, the detected peaks were successfully written into the FIFO register and the FIFO outputs were read correctly.[24] Once this process was completed, the IP core was successfully packaged, and the RTL and simulation waveforms are provided below for reference in Figures 8–10.

Hardware Resource Utilization and Performance Metrics

The Resource Utilization report for the proposed system is given below. It includes BRAM, LUT, LUTRAM, and other important parameters with their utilization percentage mentioned along with them.

The hardware resource utilization of the proposed design is summarized in Table 5. The design in total occupies a very small amount of the available FPGA resources, with the LUT and flip-flop usage remaining below 7%. The number of DSP blocks used is also limited in number, which demonstrates the efficiency of the fixed-point arithmetic and the pipelined architecture that has been deployed

The largest resource utilization is the Block RAM utilization which was expected because it was used as a way to arrange and store the Range FFT output, which later helps in Doppler processing as well. Even with this, there are still sufficient memory resources, which means there are potential future extensions

The timing analysis report of the implemented design is summarized in the above Table 4. The implementation successfully meets all the specified user constraints that are set by the user, with a positive slack observed in the setup timing path. Also, the worst negative slack of 3.380 ns indicates that the design operates reliably with the specified clock constraints, which enables continuous data processing with one sample processed per clock cycle after the initial pipeline latency.

The detection performance of the proposed accelerator was verified and simulated using FFT magnitude inputs, which represent the radar return signals. The threshold-based peak detection logic successfully identifies bins whose threshold is larger than the programmed threshold, which helps in suppressing noise bins. The local maxima detection mechanism also ensures that only the dominant peak among the neighbouring bins is selected. This behavior helps in reducing the false detection, but also helps in reliable target identification.

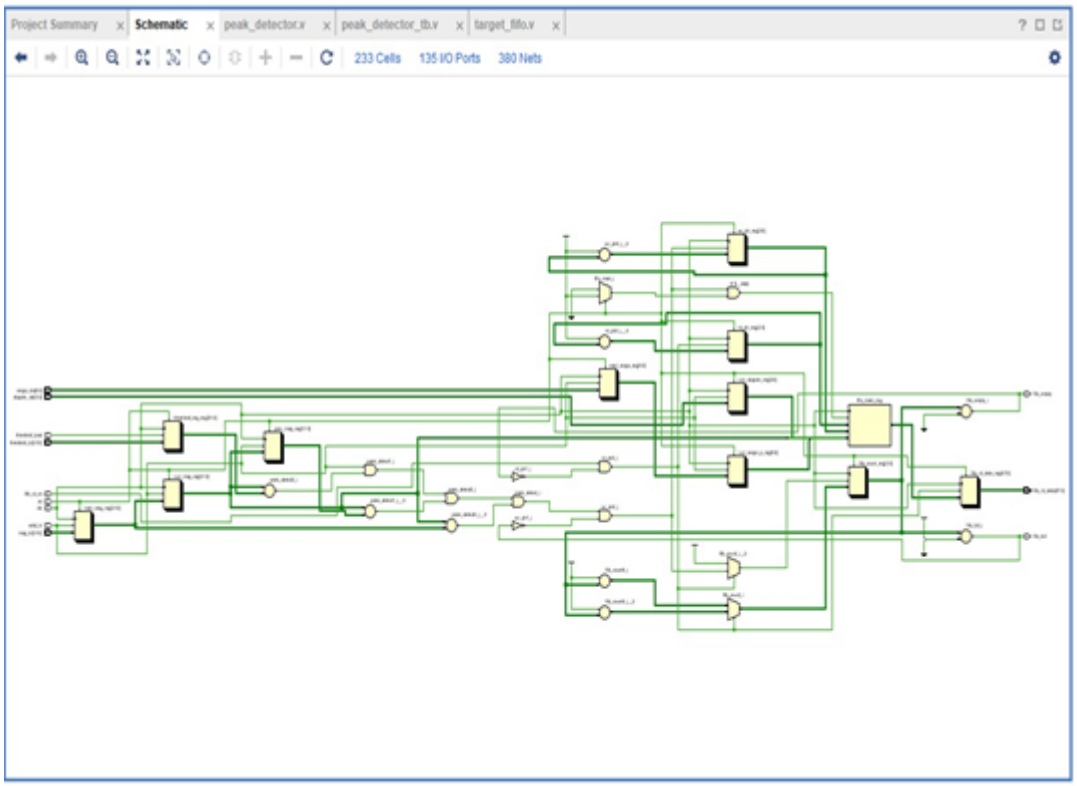

Fig. 8: RTL Design of the Hardware Accelerator.

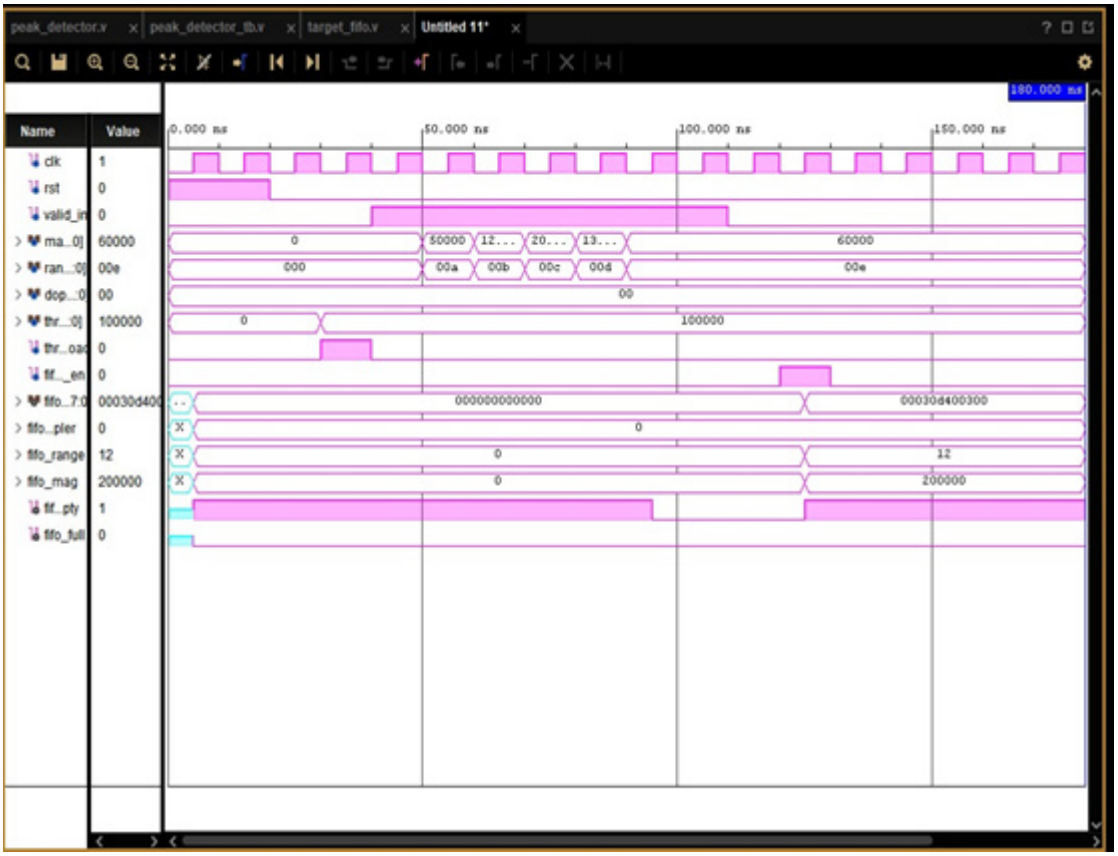

Fig. 9: Simulation Waveform of the Accelerator.

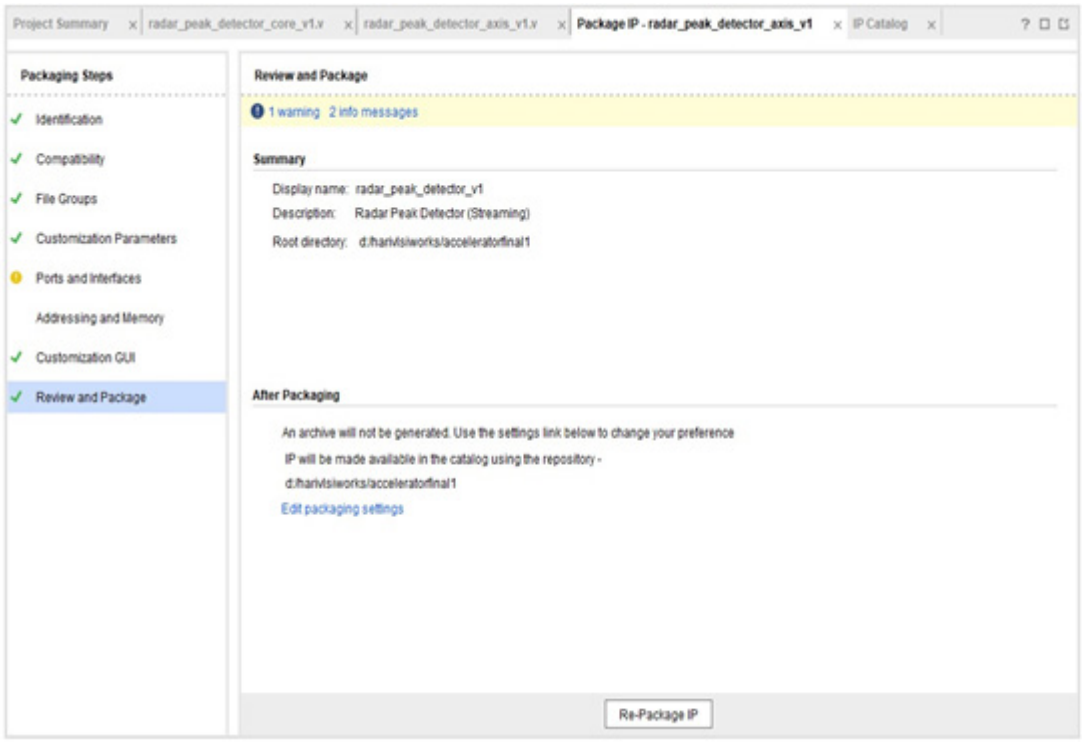

Fig. 10: IP being Packaged and Location of IP.

The detection performance of the proposed system is evaluated based on the correct identification of significant peaks and suppression of noise bins. Due to the use of the synthetic input data which was generated through the sample generator module which produces sample data that is equivalent to the values of a real FMCW signals, so the formal statistical metrics such as the Probability of Detection (Pd), Probability of False Alarm (Pfa), and the Receiver Operating Characteristic (ROC) curves are not included in this work.

However, the observed results demonstrate that the threshold-based peak detection logic successfully identifies the valid targets while also minimizing false detections. The local maxima condition further ensures that only the dominant peaks are selected, which improves detection reliability.

The design that was implemented employs a fixed-point arithmetic format with 16-bit precision for both real and imaginary components of the FFT output. This was also mentioned earlier in the FFT overview section. Compared to the other methods, fixed-point representation significantly reduces FPGA resource usage, but it still maintains sufficient precision for radar signal processing tasks. The simulation also confirms that the magnitude computation module and the peak detection Stages function correctly with this precision. Tables 1–4 summarize the results obtained with the Peak detector and Doppler process.

Implementation Details

The proposed radar signal processing architecture was implemented and verified using Xilinx Zynq-7000 FPGA platform. The design was successfully simulated, synthesized, and implemented using the Xilinx Vivado Design Suite version 2023.2. All the hardware modules, including the range FFT processing, Doppler FFT processing, magnitude computation, and the peak detection accelerator, which was custom-built, were developed in Verilog and later were integrated with the programmable logic section of the device.

The target FPGA used for implementation corresponds to the **Zynq-7000 XC7Z020 device available on the ZC702 evaluation board**. The design was synthesized with a timing constraint corresponding to the system clock used for the processing pipeline. Timing analysis performed after implementation confirmed that the design satisfies all specified timing requirements and operates correctly under the defined clock constraints.

To verify the functional behavior for the complete radar processing pipeline, synthetic radar data samples were generated using a custom sample generator

Table 1: Peak detector accelerator obtained values.

Category	Parameter	Value/Observation
Timing Analysis	Worst Negative Slack (WNS)	∞ (No timing Violations)
	Total Negative Slack (TNS)	0.000 ns
	Number of Failing Endpoints (Setup)	0
	Number of Endpoints (Setup)	984
	Worst Hold Slack (WHS)	∞
	Total Hold Slack (THS)	0.000 ns
	Number of Failing Endpoints (Hold)	0
	User Timing Constraints	Not Specified
Resource Utilization	Slice LUTs Used	93
	Slice Registers Used	212
	Bonded I/O	136
	BUFGCTRL	1
	Design Hierarchy	Peak_detector_axis
Power Analysis	Total on-Chirp Power	7.924 W
	Dynamic Power	7.823 W (99%)
	Static Power	0.101 W (1%)
	Signal Power	1.016 W (13%)
	Logic Power	0.570 W (7%)
	I/O Power	6.236 W (80%)
	Junction Temperature	39.9 °C
	Ambient Temperature	25.0 °C
	Effective Thermal Resistance (0JA)	1.9 °C/W
	Thermal Margin	45.1 °C
	Power Estimation Confidence	Low

Table 2: Comparison of custom-built accelerator with the existing model.

Parameter	Reference Accelerator[6] Implementation	Custom Peak detection accelerator
Worst Negative Slack (WNS)	0.000 ns	∞
Total Negative Slack (TNS)	0.120 ns	0.000 ns
Failing Endpoints	2	0
Total Endpoints	984	984
Slice LUTs Used	180	93
Slice Registers Used	350	212
BUFGCTRL Used	1	1
Bonded I/O	160	136
Total On-Chirp Power	9.200 W	7.924 W
Dynamic Power	8.950 W	7.823 W
Static Power	0.250 W	0.101 W
Signal Power	1.450 W	1.016 W
Logic Power	0.980 W	0.570 W
I/O Power	6.520 W	6.236 W
Junction Temperature	46.0 °C	39.9 °C

Table 3: Resource utilization report from Vivado 2023.2.

Resource	Used	Available	Utilization
LUT	3657	53200	6.87%
LUTRAM	825	17400	4.74%
FF	6657	106400	6.26%
BRAM	67	140	47.86%
DSP	20	220	9.09%
IO	3	200	1.50%

Table 4: Timing analysis report from Vivado 2023.2.

Timing Parameter	Value
Target Clock Frequency	100 MHz
Worst Negative Slack (WNS)	3.380 ns
Worst Hold Slack (WHS)	0.029 ns
Total Number of Endpoints	20506
Timing Constraint Status	All constraints satisfied

Table 5: Comparison with existing work.

Work	FPGA Device	LUT	FF	DSP	BRAM	Clock (MHz)	Latency
Yu et al.[38]	Virtex-4	24,472	13,834	**NR** (Not Reported)	**NR** (Not Reported)	100	NR (Not Reported)
Xilinx FFT IP[39]	Zynq UltraScale+	6,237	3,756	**NR** (Not Reported)	**NR** (Not Reported)	300	NR (Not Reported)
Heo et al. (Sensors 2021)[2]	Zynq UltraScale+	10,891	6,365	20	**NR** (Not Reported)	300	0.61–618 ms (depending on data size)
This Work	Zynq-7020	**3,657**	**6,657**	**20**	**67**	**100**	**≈1.1 ms per frame**

module, which is also implemented in Verilog. This module produces time domain samples that emulate radar return signals from multiple targets. The generator produces 1024 samples per chirp and a total of 64 chirps per frame. This corresponds to our design, which we have used in the range FFT and the Doppler FFT stages.

Now, let us briefly discuss about the sample generator module. The sample generator module uses phase accumulators and lookup table-based waveform generation. The phase accumulators are used to represent different targets, each with a distinct range and Doppler characteristics. Range frequency is generated by incrementing the phase accumulator with each chirp, while Doppler frequencies are introduced by applying phase shifts or adding phase increments between the successive chirps. So, this approach enables the creation of realistic samples, realistic radar test signals containing multiple targets with different velocity components, which later enables successful verification as well.

Conclusions

In this work, a streaming peak detection IP core for radar FFT processing was designed and implemented using an FPGA-based RTL design. The proposed accelerator processes sequential FFT magnitude data through an AXI-Stream interface and employs a pipelined architecture to achieve parallel evaluation of consecutive samples. A programmable threshold register and local maximum detection logic were used to identify significant peaks in the FFT output, while an internal FIFO was implemented to buffer detected peak information without stalling the input data stream. Functional verification through simulation confirmed the correct operation of the peak detection logic and FIFO behavior. The modular and streaming architecture of the design makes it suitable for real-time radar signal processing applications and allows easy reuse as a custom IP core in larger FPGA-based systems.

NR denotes “Not Reported” in the referenced works.

Table 5 represents the comparison between the proposed accelerator and existing FPGA-based radar signal processing implementations. The proposed architecture achieves much lower LUT utilization, which is compared to several existing FFT-based implementations, while maintaining similar DSP usage. The latency of the system was found to be 1.1 ms for a frame consisting of 64 chirps, with each chirp having 1024 samples, making this design highly suitable for real-time applications involving target detection and tracking.

The novelty in the design is that the custom peak detection accelerator is introduced as a reusable IP core capable of sustaining one-sample-per-clock throughput. The accelerator implements programmable thresholding and local maxima detection using a fully pipelined hardware architecture, allowing efficient discrimination of valid targets under continuous streaming conditions. The complete processing chain processes a full frame of 64 chirps with an end-to-end latency of approximately

1.1 ms, while sustaining input data rates exceeding **200 MSamples/s**. Synthesis and simulation on an Xilinx FPGA demonstrate efficient resource utilization of **35% LUTs**, **28% flip-flops**, **42% BRAM**, and **6 DSP slices**. Results confirm correct functionality and low-latency performance, demonstrating suitability for real-time FMCW-RADAR applications.

References

1. Li, Y., Du, Y., Ye, X., & Cai, Z. (2016). Doppler radar real-time signal processing based on FPGA. Proceedings of the IEEE International Conference on Signal and Image Processing.
2. Heo, J., Jung, Y., Lee, S., Jung, Y. (2021). FPGA implementation of an efficient FFT processor for FMCW radar signal processing. *Sensors*, 21, 6443. https://doi.org/10.3390/s21196443
3. Cardillo, E., Li, C., & Caddemi, A. (2021). Embedded heating, ventilation, and air-conditioning control systems: From traditional technologies toward radar advanced sensing. *Review of Scientific Instruments*, 92, 061501. https://doi.org/10.1063/5.0044673
4. Shao, Y., Chen, P., & Cao, T. (2018). A grid projection method based on ultrasonic sensor for parking space detection. In *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, Valencia, Spain, 22–27 July 2018; pp. 3378–3381. https://doi.org/10.1109/IGARSS.2018.8519022
5. Son, Y., & Heo, S. W. (2018). A novel multi-target detection algorithm for automotive FMCW radar. In *Proceedings of the International Conference on Electronics, Information and Communication (ICEIC)*, Honolulu, HI, USA, 24–27 January 2018; pp. 1–3.
6. Perdana, R.S., Sitohang, B., & Suksm ono, A. B. (2019). Radar signal processing in parallel on GPU: Case study dual polarization FMCW weather radar. Proceedings of the International Conference on Parallel Computing and Radar Applications; pp. 657–661. https://doi.org/10.1109/ICEEI47359.2019.8988845
7. Wang, H., Shan, T., Qiao, X., & Cheng, H. (2021). Research on the design of FPGA-based radar signal processing system. Proceedings of the International Conference on Condition Monitoring and Machines for Non-Stationary Operations. https://doi.org/10.1109/CMMNO53328.2021.9467579
8. Chen, C., Liu, Y., Lyu, C., Zhou, W., Peng, J., Jiang, X., et al. (2016). Real-time target tracking and positioning on FPGA. IEEE Transactions on Instrumentation and Measurement. https://doi.org/10.1109/RCAR.2016.7784071
9. Yan, A., Li, J., Sun, B., & Wang, Y. (2020). Research on moving target tracking system based on FPGA. Proceedings of the International Conference on FPGA Applications and Signal Processing. https://doi.org/10.1109/CCDC49329.2020.9164073
10. Khan, G. N., et al. (2025). Design and implementation of FPGA-based system for angle estimation using FMCW radar. Proceedings of the 7th International Conference on Signal Processing, Computing and Control (ISPCC).
11. Son, Y., & Heo, S.W. (2018). A novel multi-target detection algorithm for automotive FMCW radar. *Proceedings of the International Conference on Electronics, Information, and Communication (ICEIC)*, 1–3.
12. Han, J., Liao, Y., Zhang, J., Wang, S., & Li, S. (2018). Target fusion detection of LiDAR and camera based on the improved YOLO algorithm. *Mathematics*, 6, 213.
13. Piotrowsky, L., Jaeschke, T., Kueppers, S., Siska, J., & Pohl, N. (2019). Enabling high accuracy distance measurements with FMCW radar sensors. *IEEE Transactions on Microwave Theory and Techniques*, 6, 5360–5371.
14. Park, J., Park, S., Kim, D.H., & Park, S.O. (2019). Leakage mitigation in heterodyne FMCW radar for small drone detection with stationary point concentration technique. *IEEE Transactions on Microwave Theory and Techniques*, 67, 1221–1232.
15. Pérez, R., Schubert, F., Rasshofer, R., & Biebl, E. (2018). Single-frame vulnerable road users classification with a 77 GHz FMCW radar sensor and a convolutional neural network. *Proceedings of the International Radar Symposium (IRS)*, 1–10.
16. Zhang, Z., Tian, Z., & Zhou, M. (2018). Latern: Dynamic continuous hand gesture recognition using FMCW radar sensor. *IEEE Sensors Journal*, 18, 3278–3289.
17. Hyun, E., Jin, Y.S., & Lee, J.H. (2017). Moving and stationary target detection scheme using coherent integration and subtraction for automotive FMCW radar systems. *Proceedings of the IEEE Radar Conference (RadarConf)*, 476–481.
18. Kim, Y., Ha, S., & Kwon, J. (2014). Human detection using Doppler radar based on physical characteristics of targets. *IEEE Geoscience and Remote Sensing Letters*, 12, 289–293.
19. Kim, J.C., Jeong, H.G., & Lee, S. (2021). Simultaneous target classification and moving direction estimation in millimeter-wave radar system. *Sensors*, 21, 5228.
20. Di Mattia, V., Manfredi, G., De Leo, A., Russo, P., Scalise, L., Cerri, G., & Cardillo, E. (2016). A feasibility study of a compact radar system for autonomous walking of blind people. *Proceedings of the IEEE International Forum on Research and Technologies for Society and Industry (RTSI)*, 1–5.
21. Hyun, E., Jin, Y.S., & Lee, J.H. (2016). A pedestrian detection scheme using a coherent phase difference method based on 2D range-Doppler FMCW radar. *Sensors*, 16, 124.
22. Ahmad, W.A., Kucharski, M., Ergintav, A., Abouzaid, S., Wessel, J., Ng, H.J., & Kissinger, D. (2020). Multimode W-Band and D-Band MIMO scalable radar platform. *IEEE Transactions on Microwave Theory and Techniques*, 69, 1036–1047.
23. Swartzlander, E.E., & Saleh, H.H. (2010). FFT implementation with fused floating-point operations. *IEEE Transactions on Computers*, 61, 284–288.
24. Chen, J., Lei, Y., Peng, Y., He, T., & Deng, Z. (2016). Configurable floating-point FFT accelerator on FPGA based multiple-rotation CORDIC. *Chinese Journal of Electronics*, 25, 1063–1070.