A PROJECT REPORT

ON

DESIGN AND IMPLEMENTATION OF TARGETS SIMULATOR FOR A TRACKING RADAR USING ZYNQ-BASED SoC

A report submitted in partial fulfilment of the requirements for the ***Final Year Main Project***

BY

VIDYADHEESHA M PANDURANGI
(Reg. No: 6176AC22UEC162**)**

B.E. ELECTRONICS AND COMMUNICATION ENGINEERING
ADHIYAMAAN COLLEGE OF ENGINEERING
Dr. M.G.R Nagar, Hosur – 635 130

Under the Supervision of

Dr. S. SUMATHI,
Head / Professor – Department of ECE
Adhiyamaan College of Engineering, Hosur

A. VAIDHYANATHAN,
Scientist/Engineer-SG, Manager
MOTR/RTS/RO – SDSC SHAR (ISRO)

SATISH DHAWAN SPACE CENTRE (SHAR)
Indian Space Research Organisation (ISRO), Sriharikota

October 2025

ABSTRACT

Radar systems play a crucial role in the detection, tracking, and localization of objects, both in space and defence applications. The Target Simulator acts as an essential testing module that enables the radar system to validate its functionality under varying range and motion conditions without requiring an actual moving target.

This project, titled “Design and Implementation of Target Simulator for a Tracking Radar Using Zynq-Based SoC,” focuses on the development of a hardware-based simulation model using a Zedboard integrated with Vivado Design Suite. The Zedboard, which combines an ARM processing system and programmable logic (FPGA fabric), is used to generate, delay, and observe radar pulses corresponding to target returns. The radar’s track mode operation is simulated with a single transmit pulse and two dynamic receive pulses, emulating two moving targets with different dynamics.

The implemented system successfully simulates two independent dynamic targets with real-time delay variations. Oscilloscope observations confirm accurate pulse generation, delay control, and synchronization between transmit and receive signals. This project provides a scalable hardware framework for radar system testing and can be further extended to multiple-target simulation and Doppler effect integration.

ACKNOWLEDGEMENT

The success of this Final year Main Project Work is attributed not only to dedicated effort but also to the invaluable guidance and unwavering support from mentors, making it possible from the start of my project here at SDSC.

I express my heartfelt gratitude to **Shri PADMAKUMAR E. S**., Distinguished Scientist and Director of Satish Dhawan Space Centre, SHAR, a significant centre of the Indian Space Research Organisation (ISRO), for providing me with the invaluable opportunity for an internship at SDSC SHAR.

I wish to express my sincere gratitude to **P. SUNIL**, Outstanding Scientist and Deputy Director, Range Operations Entity, SDSC SHAR for allowing me to do my Final Year Main Project Work.

My sincere thanks to **M.M.V. DHANUNJAYA RAO**, SCI/ENG-G, General Manager, RTS, RO, for facilitating my Final Year Project Work and ensuring a smooth onboarding process.

I wish to extend my gratitude to my guide **A. VAIDHYANATHAN**, SCI/ENG-SG, Manager - MOTR, SDSC SHAR and all the **team members of MOTR** for their support to do my Final Year Main Project.

I would like to express my sincere gratitude to all those who supported and guided me during my project at the Indian Space Research Organization (ISRO).

This project work has been a valuable learning experience, and I am truly grateful for the opportunity to be given in such a prestigious organization**.**

I extend my gratefulness to **Dr. R. RADHAKRISHNAN**, Principal and **Dr. S. SUMATHI**., Head of the Department (HoD) of the Department of Electronics and Communication Engineering, Adhiyamaan College of Engineering, Hosur. Their support and the opportunity provided to work at SDSC SHAR have been instrumental in enhancing my knowledge and skills.

TABLE OF CONTENTS

List of Figures

List of Tables

List of Abbreviations

Abbreviation	**Full Form**
ISRO	Indian Space Research Organisation
SDSC SHAR	Satish Dhawan Space Centre – Sriharikota Range
RO	Range Operations
RTS	Range Tracking System
MOTR	Multi-Object Tracking Radar
FPGA	Field Programmable Gate Array
SoC	System on Chip
Zynq	Xilinx Zynq-7000 Series SoC
Zedboard	Xilinx Zynq-7000 Development Board
ISS	International Space Station
CSS	Chinese Space Station
RADAR	Radio Detection and Ranging
Tx	Transmit / Transmitter
Rx	Receive / Receiver
PRI	Pulse Repetition Interval
PRF	Pulse Repetition Frequency
PW	Pulse Width
TCA	Time of Closest Approach
AOS	Acquisition of Signal
LOS	Loss of Signal
BRAM	Block Random Access Memory
IP	Intellectual Property (Core)
PS	Processing System (ARM Cortex-A9 in Zynq)
PL	Programmable Logic (FPGA fabric in Zynq)

Abbreviation	**Full Form**
DSO	Digital Storage Oscilloscope
LEO	Lower Earth Orbit
MHz	Megahertz
µs	micro second
ns	nano second
km	kilo metre
m/s	meters per Second
GUI	Graphical User Interface
RTL	Register Transfer Level
HDL	Hardware Description Language
Vivado	Xilinx Vivado Design Suite
.coe	Coefficient File (used for BRAM memory initialization)
PWM	Pulse Width Modulation
GPIO	General Purpose Input/Output
VHDL	VHSIC Hardware Description Language
DRC	Design Rule Check
PLL	Phase-Locked Loop
Tcl	Tool Command Language (used in Vivado constraints)
COE File	Coefficient File Format used to Initialize Block Memory in FPGA
Hz	Hertz (cycles per second)
TB	Testbench

CHAPTER 1
INTRODUCTION TO THE ORGANIZATION

1.1 Introduction

The **Satish Dhawan Space Centre – SHAR (SDSC SHAR)**, located at Sriharikota in Andhra Pradesh, is the primary spaceport of the **Indian Space Research Organisation (ISRO)** and plays a crucial role in India's satellite launch missions and space exploration programs. It is strategically situated between Pulicat Lake and the Bay of Bengal, providing natural safety advantages for rocket launches and an equatorial location that offers velocity benefits for placing satellites into orbit. The centre was established in 1971 as the Sriharikota Range (SHAR), after being chosen in 1969 under the leadership of Dr. Vikram Sarabhai, the father of the Indian space program. Its first major success came in 1980 when India's SLV-3 launched the Rohini satellite into orbit, marking India's entry into the exclusive group of nations with launch capability. Over the decades, SHAR expanded its facilities, including propellant production plants, rocket motor test stands, assembly buildings, and advanced mission control centres, making it one of the most advanced launch complexes in the world. In 2002, it was renamed in honour of Prof. Satish Dhawan, the former ISRO Chairman who guided the organization through a period of rapid technological growth. Today, SDSC SHAR houses two operational launch pads capable of handling PSLV, GSLV, and LVM3 missions, with a third launch pad under development for the upcoming human spaceflight program, Gaganyaan. From launching India's flagship missions such as Chandrayaan-1, Mangalyaan, and Chandrayaan-3, to deploying hundreds of commercial satellites for foreign clients, the centre has become a symbol of India's scientific excellence and technological self-reliance. With ongoing advancements and preparations for human spaceflight, SDSC SHAR continues to stand at the heart of India's journey toward becoming a global leader in space exploration. From its modest beginnings with sounding rockets to achieving global recognition through lunar and interplanetary missions, the centre has continuously expanded its capabilities to support India's growing ambitions in space. With state-of-the-art infrastructure, a rich legacy of achievements, and visionary future projects, SDSC SHAR continues to symbolize India's scientific excellence, technological strength, and aspirations to become a leading space power in the world.

1.3 Introduction to MOTR

The **Multi-Object Tracking Radar (MOTR)** is an advanced phased array radar system installed at the Satish Dhawan Space Centre – SHAR for tracking Launch Vehicles and its stages during launch missions and satellites and space objects during regular operations. MOTR is developed indigenously by SDSC SHAR with support from other ISRO centres, and Indian industries. MOTR became operational in **2015**. MOTR operates in the L-band frequency and can track up to 10 objects simultaneously. It is capable of detecting even small objects measuring 0.25m^2 up to a range of 1000km making it highly effective for monitoring launch vehicles, separated stages, and payloads in real-time. Unlike conventional mechanical radars, MOTR uses phased array technology, which electronically steers the radar beam without moving the antenna. This provides faster, more accurate tracking during the critical phases of launch. The radar is a massive 12-meter structure, built with thousands of solid-state transmit-receive modules for reliable performance. MOTR plays a vital role in ensuring launch safety, trajectory accuracy and space situational awareness**,** while also strengthening India's indigenous technological capabilities in high-end radar systems.

CHAPTER 2
OVERVIEW OF RADARS

2.1 Radar Overview

Radar (Radio Detection and Ranging) is an electronic system that employs radio waves to detect, locate, and monitor objects, as well as to measure their distance, speed, direction, and sometimes size or shape. The basic principle involves transmitting high-frequency electromagnetic waves toward a target and receiving the waves that are reflected back as echoes. By calculating the time delay between transmission and reception, radar determines the distance to the object, while shifts in the frequency of the reflected signal (Doppler effect) can provide information about the object's velocity. Modern radar systems consist of several key components: a transmitter, which generates radio waves; an antenna, which directs the waves and receives echoes; a receiver, which detects and amplifies the weak returned signals; a signal processor, which analyses the data; and a display unit, which presents the information in an understandable form. Depending on the requirements, radars can be classified as pulse radar, which sends periodic pulses to measure range; continuous wave radar, which measures speed; Doppler radar, specialized for detecting moving objects; or phased array radar, which electronically steers the beam without physically moving the antenna, allowing rapid and precise tracking. Radars are widely employed in aviation, maritime navigation, weather forecasting, military defence, traffic management, and space research, providing real-time information essential for safety, monitoring, and operational decision-making.

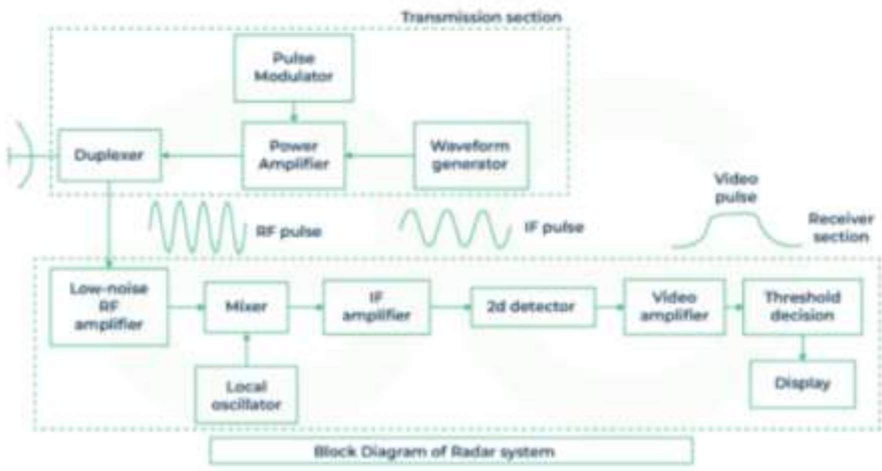

Fig 1: Block Diagram of Radar

2.2 Radar Classification

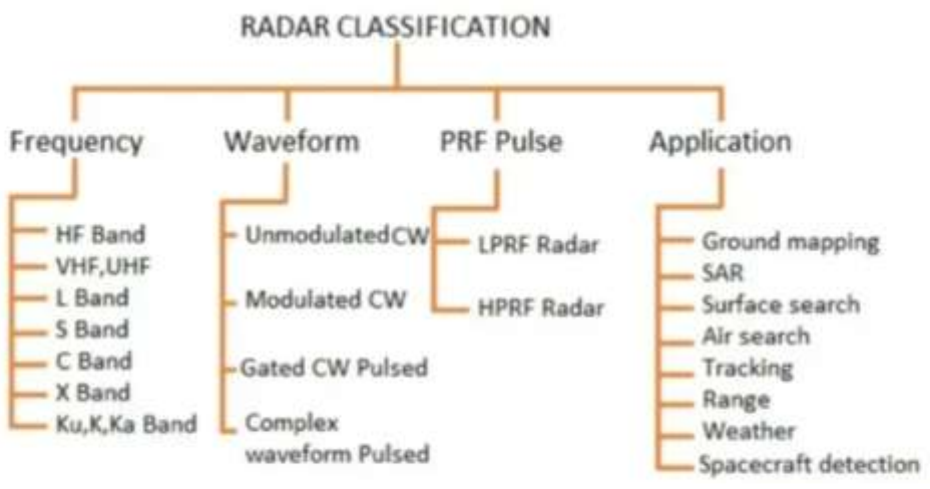

Fig 2 - Classification of Radars

2.2.3 PRF Based Radar Types

1. **LPRF Radar:** LPRF stands for Low Pulse Repetition Frequency Radar. It's particularly useful for determining the range to a target.
2. **HPRF Radar:** HPRF stands for High Pulse Repetition Frequency Radar. Also known as pulse Doppler radar, it excels at measuring Doppler shift, which allows for velocity determination. However, HPRF radars are generally not used for range measurements.

2.2.4 Application Based Radar Classification

1. **Search radars (surface, air)**: Search radars continuously sweep large areas either above (air search) or across the earth's surface (surface search) to detect and localize targets such as aircraft, ships, or vehicles. They emit short pulses and scan 360°, providing initial detection and range information, often with Doppler filtering to distinguish moving targets
2. **Warning radar**: Warning radars encompass systems like weather and storm detection radars. They identify meteorological threats (storms, hail, wind shear) using specialized frequencies and polarizations. Doppler radar measures motion within weather systems, aiding timely alerts
3. **Spacecraft detection radars**: These radars operate from ground or space to detect, track, and characterize spacecraft or space debris. They support navigation, docking (e.g., rendezvous), and remote sensing missions such as altimetry and mapping

4. **Fire control radars**: Designed for precision tracking of a designated target, fire control radars focus very narrow beams to deliver range, velocity, and angular data directly to weapon systems. They're essential for missile lock-on and accurate targeting, often integrated with search radars for broader coverage

5. **Ground mapping radars**: These radars, typically airborne or spaceborne, produce detailed topographic maps, often utilizing SAR techniques. They scan terrain and produce imagery for geological surveys, environment monitoring, and navigation.

6. **SAR (Synthetic Aperture Radar)**: SAR is a high-resolution form of ground or terrain-mapping radar. By using the motion of the radar platform, it synthesizes a large antenna aperture from multiple pulses, generating detailed 2D/3D imagery regardless of weather or daylight

7. **Air to Surface radars**: These airborne radars scan the earth's surface from aircraft or helicopters to detect maritime vessels, terrain features, or ground targets. Ship-based variants are sometimes specifically referred to as ASV (air-to-surface vessel) radar

8. **Sea surface radar**: Mounted on ships or coastal stations, these radars are tailored to detect objects on the water. They maximize detection by exploiting sea-reflected signals, while mitigating sea-clutter to reliably identify vessels and surface obstacles

9. **Ground moving target search radars**: Also known as GMTI radars, they scan terrain to detect and locate moving ground targets (e.g., vehicles) using Doppler processing to distinguish them from stationary clutter

10. **Tracking radar**: Tracking radars lock onto a previously detected target to continually measure its position and velocity. They typically feature narrower beams and faster update rates than search radars, feeding data into fire control or surveillance systems

11. **Range radar**: Also known as ranging or distance-measurement radar, this system's primary function is to determine the precise distance to a target using pulse timing. This core functionality underlies many radar applications, from navigation to altitude measurement.

12. **Velocity search radar**: This type of radar focuses on detecting objects based on Doppler velocity shifts rather than range. High-PRF (pulse repetition frequency)

Doppler radars excel at isolating fast-moving targets and determining their speed accurately.

2.3 Applications of Radar

1. Military: Used for surveillance, target tracking, missile guidance, and early warning systems.

2. Aviation: Monitors aircraft position, assists in air traffic control, and ensures collision avoidance.

3. Space: Tracks satellites, space debris, and supports deep-space missions and astronomy.

4. Marine: Helps ships navigate, avoid collisions, and aids in search-and-rescue operations.

5. Meteorology: Detects rainfall, storms, hurricanes, and measures wind speed with Doppler radar.

6. Industrial & Scientific: Used in ground-penetrating radar, Earth remote sensing, and non-destructive testing.

2.4 Radar Range Equation

The Radar Range Equation is a mathematical formula that relates the power transmitted by a radar system to the power received back after reflection from a target. It helps determine the maximum range at which a radar can detect an object, based on system parameters, target characteristics, and environmental factors.

$$P_r = \frac{P_t G_t G_r \lambda^2 \sigma}{(4\pi)^3 R_t^2 R_r^2 L}$$

Where

Pt – Transmitted Power (W): Power emitted by the radar transmitter.

Pr – Received Power (W): Power collected by the receiving antenna after reflection.

Gt – Transmit Antenna Gain: Directivity of the transmitting antenna.

Gr – Receive Antenna Gain: Ability of the receiving antenna to capture signal energy.

λ – Wavelength (m): Related to frequency by λ=c/f.

σ – Radar Cross Section (m²): Effective area representing target's detectability.

Rt – Transmit Range (m): Distance from transmitter to target.

Rr – Receive Range (m): Distance from target to receiver.

L – System Loss Factor (dimensionless, >1): Includes hardware inefficiencies, propagation and atmospheric losses.

$(4\pi)^3$ - Accounts for spherical spreading of the signal during transmission and reception.

By combining antenna performance, target characteristics, and propagation effects, the equation provides a fundamental framework to design, analyze, and optimize radar systems for reliable detection and tracking of objects.

CHAPTER 3

FUNDAMENTALS OF RADAR ANTENNAS

3.1 Antenna

An antenna is a fundamental device used in wireless communication systems that serves as the interface between electrical circuits and free space. It functions as a transducer, meaning it converts electrical signals into electromagnetic waves during transmission and converts incoming electromagnetic waves into electrical signals during reception. The working principle of an antenna is based on the fact that an alternating current flowing through a conductor generates electromagnetic radiation, and conversely, an incoming electromagnetic wave can induce an alternating current in the conductor. Antennas can be designed in various shapes and sizes depending on the frequency and application, ranging from simple wire antennas like dipoles and monopoles to more complex structures such as horn antennas, parabolic reflectors, phased arrays, and microstrip patch antennas .In essence, an antenna is the critical element that enables wireless connectivity, making it one of the most essential components of modern communication and sensing technologies.

Fig 3 – Aperture Antenna

3.2 Aperture Antenna Gain and Beamwidth

3.2.1 Aperture Antenna Overview

An aperture antenna is a type of antenna in which the radiating element is an opening or "aperture" in a conducting surface through which electromagnetic energy is transmitted or received. Examples include horn antennas, parabolic reflectors, and planar microstrip or phased-array apertures.

In radar systems, aperture antennas are widely used because they can generate highly directive beams and provide high gain, which are essential for long-range detection and accurate tracking.

3.2.2 Antenna Gain

The gain (G) of an antenna represents its ability to direct radiated power in a specific direction compared to an isotropic radiator.
For an aperture antenna, the gain is related to the effective aperture area (A_e) and the wavelength (λ) as:

$$G = \frac{4\pi A_e}{\lambda^2}$$

Where:

- G= Antenna gain (dimensionless or in dB)
- A_e= Effective aperture area (m^2)
- λ= Wavelength of operation (m)

However, not all the physical aperture contributes effectively to radiation because of losses and field nonuniformities.
Hence, the aperture efficiency (η_a) is introduced:

$$A_e = \eta_a A_p$$

Where:

- η_a= Aperture efficiency (typically 0.5–0.8 for real antennas)
- A_p= Physical aperture area

Thus, the practical gain becomes:

$$G = \eta_a \frac{4\pi A_p}{\lambda^2}$$

or in decibels:

$$G_{dB} = 10\log_{10}(\eta_a \frac{4\pi A_p}{\lambda^2})$$

3.2.3 Beamwidth of Aperture Antenna

The beamwidth describes the angular spread of the main lobe of the antenna radiation pattern. It is commonly expressed as Half-Power Beamwidth (HPBW) — the angle between the directions at which the radiated power drops to half (−3 dB) of its maximum value.

For a uniformly illuminated rectangular aperture antenna of dimension D:

$$\text{HPBW (radians)} \approx \frac{k\lambda}{D}$$

or in degrees:

$$\text{HPBW }(^\circ) \approx 70\frac{\lambda}{D}$$

Where $k$is a proportionality constant depending on aperture shape (≈ 1 for rectangular, 1.02 for circular).

Interpretation:

- Larger aperture D→ narrower beamwidth → higher directivity.
- Smaller aperture D→ wider beamwidth → lower directivity.

3.2.4 Relationship between Gain and Beamwidth

Gain and beamwidth are inversely related — a higher gain corresponds to a narrower beam. For a circular aperture:

$$G \approx (\frac{4\pi}{\theta_{HP}^2})\eta_a$$

Where θ_{HP} is the half-power beamwidth (in radians).
Hence, reducing the beamwidth by half approximately increases the gain by 6 dB.

3.3 Phased Array Antenna

A phased array antenna is a configuration consisting of multiple radiating elements (such as dipoles or microstrip patches) arranged in a specific geometry (usually linear or planar). Unlike mechanical antennas, beam steering in phased arrays is achieved electronically by adjusting the relative phase excitation of each element.

3.3.1 Operating Principle

Each array element radiates an electromagnetic wave. By applying a progressive phase shift (β) between adjacent elements, the combined radiation pattern can be directed at a desired angle (θ).

For a uniform linear array (ULA) of N elements spaced by d, the array factor (AF) is:

$$AF(\theta) = \sum_{n=1}^{N} e^{j(n-1)(kd\sin\theta+\beta)}$$

Where:

- $k = \frac{2\pi}{\lambda}$= Wave number
- d= Element spacing
- β= Progressive phase shift between elements
- θ= Observation angle

The main beam direction occurs when:

$$kd\sin\theta_0 + \beta = 0$$

$$\therefore \theta_0 = \sin^{-1}(-\frac{\beta}{kd})$$

Thus, by electronically varying β, the beam can be steered to any desired θ_0 without physically moving the antenna.

3.3.2 Advantages of Phased Array Antennas

- Electronic Beam Steering: Instantaneous steering without mechanical rotation.
- High Reliability: No moving parts, hence lower maintenance.
- Adaptive Control: Can track multiple targets simultaneously.
- Scalability: Beam shape and gain can be modified by controlling array geometry.
- Applications: Radar, satellite communications, missile tracking, 5G MIMO systems.

3.3.3 Types of Phased Array Architectures

1. Analog Phased Array: Uses analog phase shifters; lower cost and power-efficient.
2. Digital Phased Array: Each element digitized and controlled independently; offers precise control.
3. Hybrid Array: Combines analog and digital techniques for optimized performance.

3.3.4 Beamforming and Array Gain

The array gain increases linearly with the number of elements if all elements are in phase:

$$G_{array} = N \times G_{element}$$

However, in practice, mutual coupling and tapering effects slightly reduce this ideal gain.

The beamwidth of an N-element linear array is approximately:

$$\text{HPBW} \approx \frac{2\lambda}{Nd\cos\,\theta_0}$$

Thus, increasing N (more elements) or d (aperture size) narrows the beam and improves angular resolution.

CHAPTER 4
RADAR WAVEFORMS

4.1 Filters and Types

4.1.1. Filter

A filter is a circuit or system that allows certain frequency components of a signal to pass while attenuating (reducing) others. Filters are used in communication systems, electronics, and signal processing to remove noise, extract useful signals, and shape frequency response.

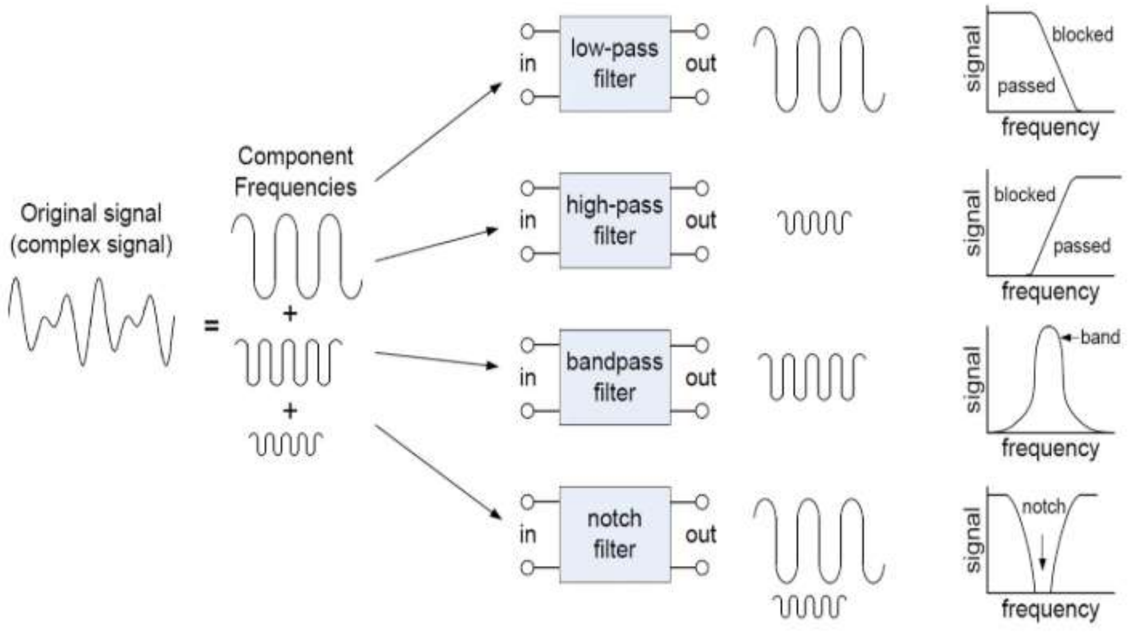

Fig 4 - Classification of Filters

Low-Pass Filter (LPF)

A low-pass filter allows frequencies below a specified cutoff frequency to pass while attenuating higher frequencies. It is commonly used for noise reduction, signal smoothing, and audio processing.

High-Pass Filter (HPF)

A high-pass filter passes signals above the cutoff frequency and blocks lower-frequency components. It is useful for eliminating DC offsets, reducing low-frequency interference, and enhancing sharp transitions.

Band-Pass Filter (BPF)

A band-pass filter permits only a specific range of frequencies between two cutoff points to pass while suppressing all others. It is widely used in wireless communication systems, channel selection, and frequency tuning.

Notch Filter (Band-Stop Filter)

A notch filter rejects a narrow band of unwanted frequencies while allowing all others to pass. It is particularly effective in removing power-line interference (50/60 Hz) from biomedical signals, audio systems, and instrumentation.

4.2 Continuous Waveform and Pulsed Waveform

Continuous Waveform (CW) Radar

A Continuous Waveform radar transmits a constant, unmodulated or modulated signal over time without interruption. Since transmission is continuous, the same antenna cannot simultaneously receive the echo, so separate transmit and receive antennas (or duplexers) are required. CW radars are commonly used to measure Doppler shifts, making them effective in detecting target velocity. Variants such as FMCW (Frequency Modulated Continuous Wave) radars also provide range information by using frequency modulation. Applications include police speed detection, collision avoidance systems, and altimeters.

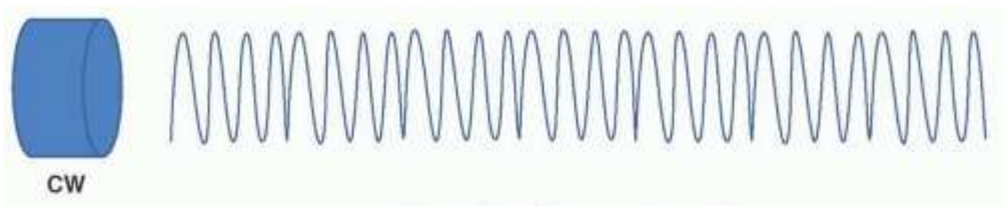

Fig 5 - Continuous Waveform (CW) Radar

Pulsed Waveform Radar

A Pulsed Waveform radar transmits short bursts (pulses) of electromagnetic energy followed by intervals of silence during which the receiver listens for echoes. The time delay between pulse transmission and echo reception is used to determine the target's range, while Doppler processing can extract velocity. Pulse radars offer long-range detection and better range resolution compared to CW radars, making them suitable for air traffic control, weather monitoring, surveillance, and military applications**.**

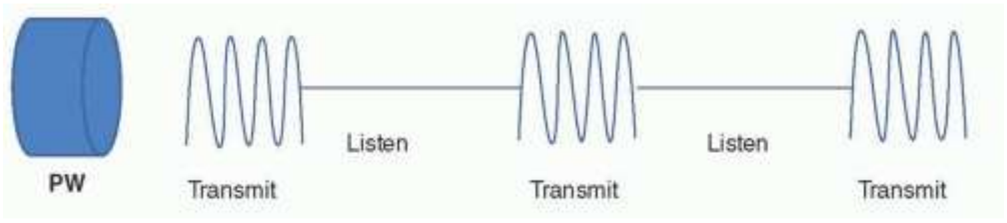

Fig 6 - Pulse Waveform (CW) Radar

4.3 Radar Basic Terminologies:

A RADAR (RAdio Detection And Ranging) system is an electromagnetic sensor used to detect, locate, and track objects (targets) by transmitting electromagnetic waves and analysing the echoes reflected back from those targets. The radar determines the range, velocity, and direction of the object based on the time delay and frequency shift of the received signal.

a) Range

The range (R) is the distance between the radar antenna and the target. It is determined by measuring the time delay (T) between the transmitted pulse and the received echo.

$$R = \frac{cT}{2}$$

Where:

- R= Target range (meters)
- c= Velocity of electromagnetic waves (≈ 3×10^8 m/s)
- T= Round-trip time delay (seconds)

The factor of 2 accounts for the two-way travel of the radar pulse — from the transmitter to the target and back to the receiver.

b) Minimum Detectable Range

The minimum range (R_i) is the closest distance at which the radar can detect a target. It is limited by the pulse width (PW) because the radar cannot receive echoes while it is transmitting.

$$R_{min} = \frac{c \times PW}{2}$$

Where:

- PW= Pulse Width (seconds)

c) Maximum Unambiguous Range

The maximum unambiguous range (R_{max}) is the farthest distance at which a target can be correctly identified without ambiguity. It is determined by the Pulse Repetition Frequency (PRF), i.e., the rate at which pulses are transmitted.

$$R_{max} = \frac{c}{2 \times PRF}$$

Where:

- PRF= Pulse Repetition Frequency (Hz)

This formula arises because if the echo from one pulse arrives after the next pulse is transmitted, the radar cannot distinguish which pulse the echo corresponds to — leading to range ambiguity.

d) Pulse Repetition Frequency (PRF)

The Pulse Repetition Frequency (PRF) is the number of pulses transmitted by the radar per second. It is inversely related to the Pulse Repetition Interval (PRI).

$$PRF = \frac{1}{PRI}$$

PRF is a critical parameter because it governs both the maximum detectable range and the radar’s ability to track fast-moving targets.

- Low PRF: Increases maximum range but reduces target velocity measurement accuracy.
- High PRF: Improves velocity measurement but reduces maximum range.

e) Pulse Width (PW)

The Pulse Width (PW) is the duration of each radar pulse. It directly affects range resolution, minimum range, and transmitted power.

$$PW = t_{on}$$

Where $t_{on}$is the time during which the rad**ar transmitter is active (pulse duration).**

Parameter	**Controlled By**	**Determines**	**Relation**
Pulse Width (PW)	Transmitter ON-time	Range resolution, Minimum range	$\Delta R = \frac{cPW}{2}$
Pulse Repetition Frequency (PRF)	Number of pulses/sec	Maximum unambiguous range	$R_{max} = \frac{c}{2PRF}$
Range	Delay between Tx and Rx	Target distance	$R = \frac{cT}{2}$

Table 1 - Relationship Between PW, PRF, and Range

4.3 LFM Waveforms

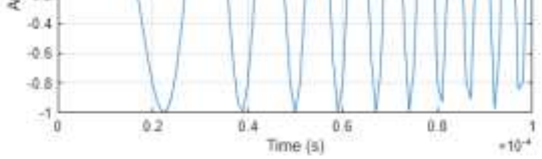

Fig 7 – Linear FM Pulse Waveform

A Linear Frequency Modulated (LFM) waveform, also called a chirp signal, is a radar waveform in which the instantaneous frequency of the transmitted pulse varies linearly with time over the pulse duration. -

- If frequency increases with time → **up-chirp**.
- If frequency decreases with time → **down-chirp**.

Mathematically:

$$s(t)=A\cdot ej\hat{}(2\pi fot+\pi Kt\hat{}2)$$

Where:

- f0 = starting frequency
- B = total bandwidth swept during the pulse
- T = pulse duration
- K=B/T= chirp rate (Hz/s)

4.4 Matched Filter

The matched filter is a fundamental concept in radar and communication theory. It is a linear filter designed to maximize the output Signal-to-Noise Ratio (SNR) when a known signal is transmitted through a channel corrupted by Additive White Gaussian Noise (AWGN). In radar, since the transmitted waveform is always known, the matched filter is used at the receiver to detect echoes from targets with maximum reliability.

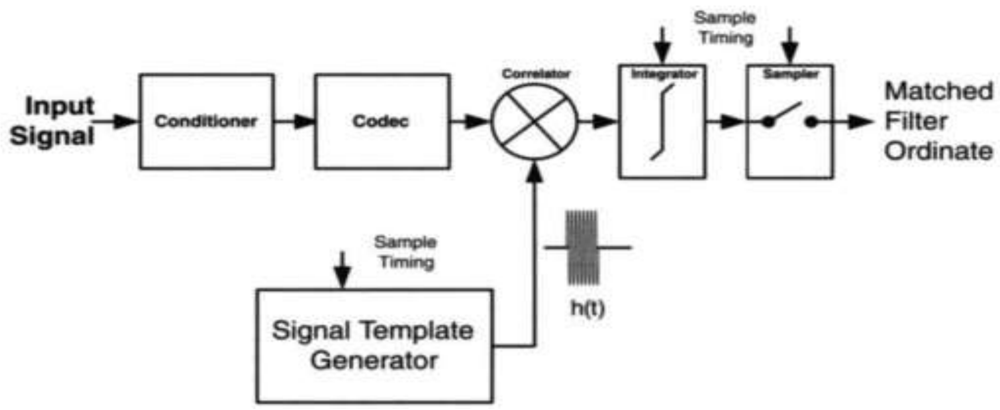

Fig 8 – Block Diagram of Matched Filter

In radar systems, the received signal is a delayed version of the transmitted waveform plus noise:

$$r(t)=\alpha s(t-\tau)+n(t)$$

where,

- α = attenuation (depends on target reflection),
- τ = round-trip delay (related to target range),
- n(t) = additive noise.

4.5 Waveform Resolution and Ambiguity

4.5.1 Waveform Resolution

Resolution in radar refers to the ability to distinguish between two closely spaced targets. It depends directly on the properties of the transmitted waveform.

(a) Range Resolution

- Determines how close two targets can be in distance and still be distinguished.
- Given by:

$$R=c/2B$$

where

c = speed of light,

B = signal bandwidth.

(b) Doppler (Velocity) Resolution

- ➢ Determines how close two targets can be in velocity and still be distinguished.
- ➢ Doppler resolution depends on the observation time Tobs.

$$\Delta fd \approx 1/Tobs$$

- ➢ Small Doppler spacing requires long observation times.

4.5.2 Waveform Ambiguity

The ambiguity function describes how well a radar waveform can resolve targets in range and velocity simultaneously.

Mathematically, the ambiguity function is the correlation between the waveform and a time–frequency shifted version of itself:

$$\chi(\tau,fd)=\int s(t)s*(t-\tau)e^{-j2\pi fdt}dt$$

where

τ = delay (range),

fd = Doppler shift (velocity).

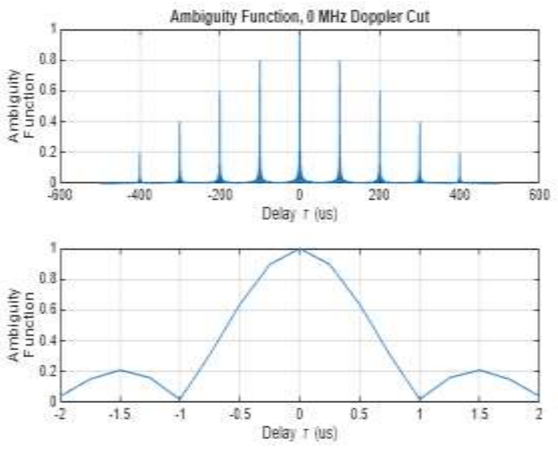

Fig 9 – Ambiguity Function Waveform at 0MHz Doppler Cut

4.6 Applications of Radar Waveforms

1. Pulse Radar: Used for long-range detection in air traffic control and weather monitoring.
2. Continuous Wave (CW) Radar: Used to measure target speed in speed guns and Doppler sensors.
3. FMCW Radar: Used in automotive radars and altimeters for range and velocity measurement.
4. LFM (Chirp) Radar: Used in imaging and mapping radars for high range resolution.
5. Phase-Coded Radar: Used in military and weather radars for better target separation.
6. Frequency Hopping Radar: Used in defence systems for anti-jamming and secure operation.

4.7 Pulse-Doppler Radar

Pulse Doppler concept is a form of radar that measure the relative velocity of a target using the Doppler frequency shift of the reflected signal, while still maintaining the range measurement capability of a pulsed Radar.

It combines the range resolution of a pulsed radar and the velocity measurement capability of a continuous wave radar.

4.7.1 Basic Principle:

1. The radar transmits a train of short-duration pulses at a fixed Pulse Repetition Frequency (PRF).

2. The reflected echo from a moving target experiences a Doppler Frequency shift given by:

$$\mathbf{fd = 2\vartheta/\lambda}$$

where:

f = Doppler Frequency Shift

ϑ = radial velocity of the target

λ = Wavelength of the transmitted signal

3. By observing the phase change between successive received pulses, the Doppler frequency and velocity is extracted.

4. The time delay between transmission and reception provides range, while the phase change across the pulses provides velocity.

4.7.2 Mathematical Relationship

If Tp is the pulse repetition interval (PRI), THEN FOR SUCCESSIVE PULSES:

$$\emptyset d = 2\pi f dTp = 4\pi\vartheta Tp/\lambda$$

A small change in phase $\emptyset d$ between consecutive echos indicates target motion.

When integrated over N pulses, a fourier transform (Range Doppler processing) can separate multiple moving targets.

4.7.3 Doppler Blind Speeds

The condition in which the frequency shifts becomes the multiples of the PRF applied to the radar, then it is not possible to determine the speed at which the target is moving. This phenomenon is called: Doppler Blind Speeds. The condition at which it occurs is determined as:

$$fd = n * PRF$$

$$\vartheta blind = n\lambda PRF/2$$

Where:

N = Integer

λ = Radar wavelength

At these velocities, the target becomes "invisible" because their Doppler returns falls in to the same phase bin as the clutter or another velocity.

CHAPTER 5
INTRODUCTION TO FPGA

5.1 FPGA

A Field Programmable Gate Array (FPGA) is an integrated circuit (IC) that can be configured by the user after manufacturing to perform specific logic functions. Unlike fixed-function ASICs (Application Specific Integrated Circuits), which are designed for one particular task, FPGAs offer flexibility by allowing reprogramming in the field to implement various digital circuits and systems. The term "field programmable" indicates that the device’s logic can be customized by the user even after deployment, and "gate array" refers to the collection of configurable logic gates present inside the chip. FPGAs were developed to bridge the gap between hardware flexibility and performance. They provide a platform where designers can implement complex digital logic designs without fabricating a new chip each time. The core of an FPGA consists of three main components — Configurable Logic Blocks (CLBs), Programmable Interconnects, and Input/Output Blocks (IOBs). The CLBs perform logic and arithmetic operations, inter connects link the logic blocks together according to the desired circuit, and the IOBs manage communication between the FPGA and external devices.

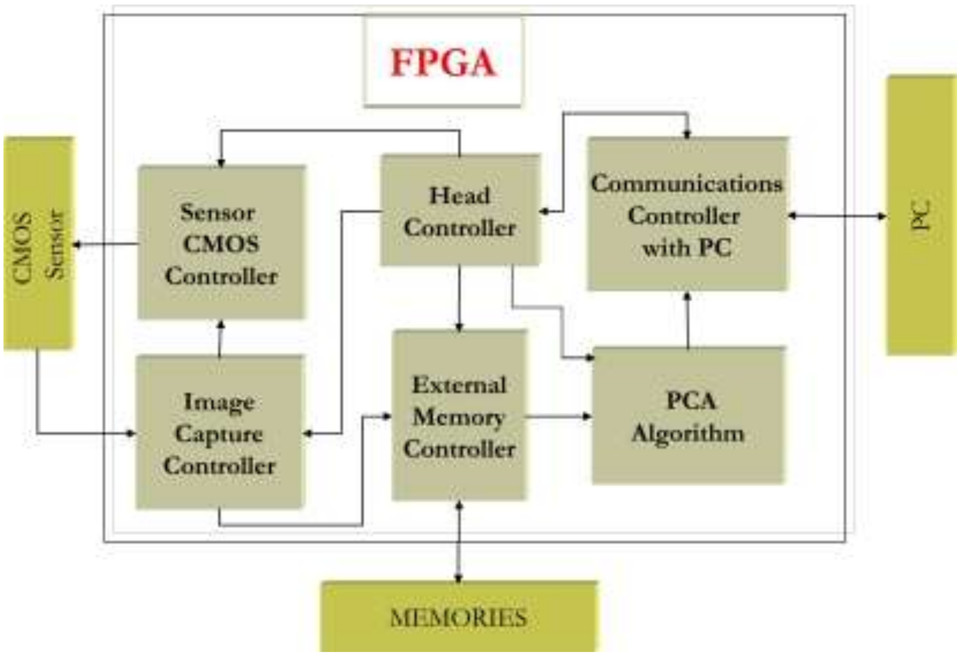

Fig 10 - FPGA Architecture

Internally, each logic block can contain lookup tables (LUTs), flip-flops, and multiplexers, allowing the designer to implement combinational and sequential logic. The interconnect network provides programmable routing paths that can be configured using a Hardware Description Language (HDL) such as Verilog or VHDL. Once the design is synthesized and implemented, a configuration bitstream is generated and loaded into the FPGA to define its functionality.

FPGAs are particularly useful in applications requiring **high-speed parallel processing**, **deterministic timing**, and **real-time signal processing**, such as **radar systems**, **communication networks**, and **embedded control systems**.

Component	Description
Configurable Logic Blocks (CLBs)	Core logic units containing Look-Up Tables (LUTs), flip-flops, and multiplexers.
Look-Up Tables (LUTs)	Small memory tables used to implement combinational logic functions. Each LUT can realize any Boolean function of 6 inputs.
Flip-Flops (FFs)	Sequential storage elements that hold one bit of data; used for timing and synchronization.
Routing Channels	Programmable interconnects that connect CLBs, IOBs, and BRAMs.
Input/Output Blocks (IOBs)	Interface circuits connecting internal logic to external pins.
Block RAM (BRAM)	Dedicated on-chip memory used for data buffering and lookup storage.
DSP Slices	High-speed arithmetic units for multiplication and addition operations.
Clock Management Tile (CMT)	Includes PLLs and MMCMs for clock generation and synchronization.

Table 2 - Major Components of an FPGA

5.2 Introduction to Vivado Design Suite

Vivado Design Suite is a software tool developed by Xilinx (now part of AMD) for designing, simulating, synthesizing, and implementing digital circuits on Xilinx FPGAs and SoCs. It replaced the older ISE Design Suite, offering a more modern environment that supports advanced FPGA architectures such as the Xilinx 7-Series, UltraScale, and UltraScale+ devices. Vivado provides a complete design flow — from writing HDL code (Verilog/VHDL) to generating bitstreams and programming the FPGA hardware. Vivado is built around a high-performance, GUI-based and TCL-scriptable environment, providing both graphical and command-line design control. It uses a modern synthesis engine that optimizes area, power, and timing more efficiently compared to older tools. The suite integrates multiple tools such as the Vivado IDE, Vivado Simulator, IP Integrator, and Hardware Manager, which together streamline the process of digital system development. One of Vivado's key features is its Block Design environment, which allows designers to connect predefined Intellectual Property (IP) blocks using a drag-and-drop interface, enabling rapid system-level design. It also supports behavioural and post-synthesis simulation, timing analysis, design optimization, and on-chip debugging using the Vivado Logic Analyser. The tool is compatible with Verilog, VHDL, and System Verilog, allowing users to mix these languages in a single project.

5.3 ZedBoard Zynq Evaluation and Development Kit

5.3.1 Introduction

The ZedBoard Zynq Evaluation and Development Kit is a powerful hardware platform designed by Xilinx, Avnet, and Digilent for developing, testing, and evaluating designs based on the Xilinx Zynq-7000 All Programmable SoC (System on Chip). It serves as an integrated development environment for both hardware and software engineers, enabling seamless hardware/software co-design. The ZedBoard combines the processing capabilities of a dual-core ARM Cortex-A9 processor with the reconfigurable FPGA fabric of the Zynq-7000 SoC, making it suitable for a wide range of embedded applications. This board provides an excellent starting point for engineers and researchers working on digital signal processing, embedded systems, control applications, machine learning, and high-speed data processing. Its design flexibility allows users to implement custom hardware accelerators while simultaneously running software on the ARM cores.

5.3.2. ZedBoard Architecture Overview

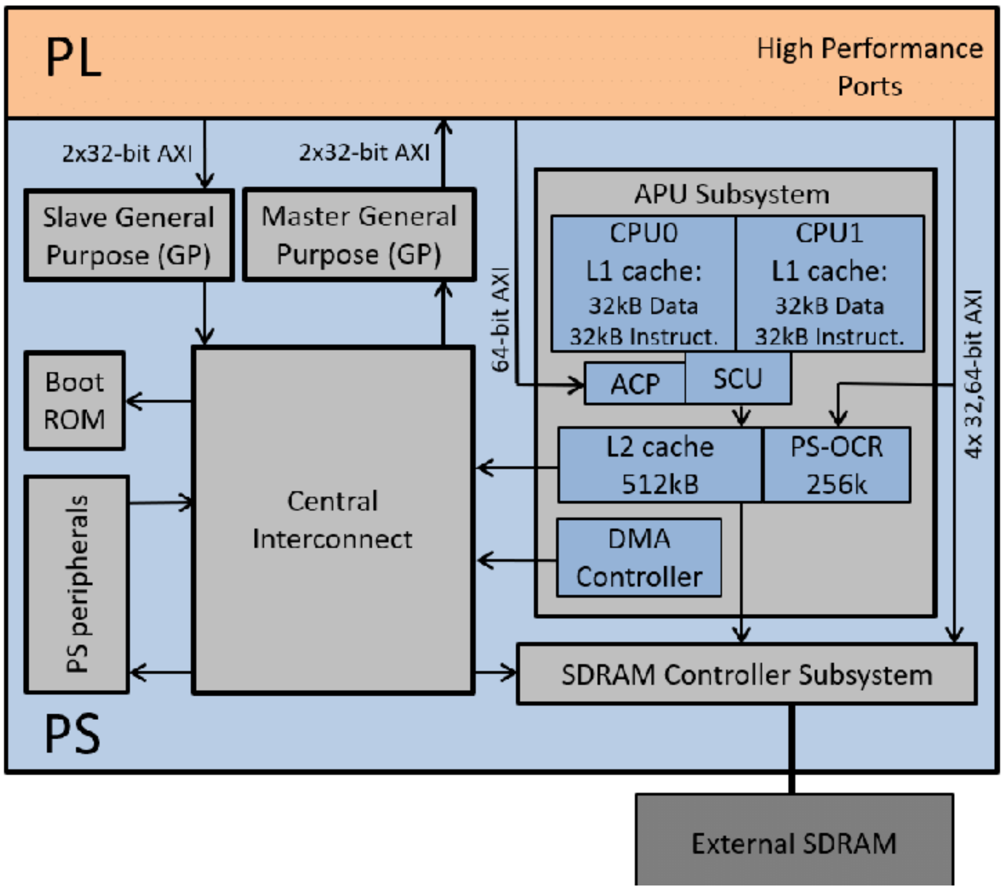

Fig 11 – Block Diagram of ZedBoard Architecture

The **ZedBoard** is a **development and evaluation platform** based on the **Xilinx Zynq-7000 All Programmable SoC (AP SoC)**. It combines the flexibility of a **dual-core ARM Cortex-A9 Processing System (PS)** with the high-performance **Programmable Logic (PL)** of Xilinx **7-series FPGA** technology on a single device. This integration makes it ideal for hardware–software co-design, enabling both embedded software execution and custom digital hardware implementation. The ZedBoard provides a comprehensive platform for rapid prototyping and system design, with key features as summarized below:

Feature	Description
Processor	Dual ARM Cortex-A9 MPCore, 32-bit, up to 866 MHz
FPGA Fabric	Xilinx XC7Z020-CLG484-1 with 85K logic cells
Programmable Logic (PL)	Equivalent to Artix-7 FPGA family with 53,200 LUTs, 106,400 Flip-Flops, 140 BRAMs, and 220 DSP slices
Memory	512 MB DDR3 SDRAM (connected to PS), 256 Mb Quad-SPI Flash
Non-volatile Storage	microSD card slot for booting and data storage
I/O Interfaces	HDMI, USB OTG, Ethernet, UART, FMC connector, PMODs, GPIOs
On-board Oscillator	33.333 MHz reference clock for PS and 100 MHz system clock for PL
Power Supply	12 V DC input through barrel jack
Debug Interface	USB-JTAG and UART via FTDI chip
Expansion Options	PMOD connectors for GPIO extension, FMC connector for high-speed peripherals

Table 3 - Features of ZedBoard

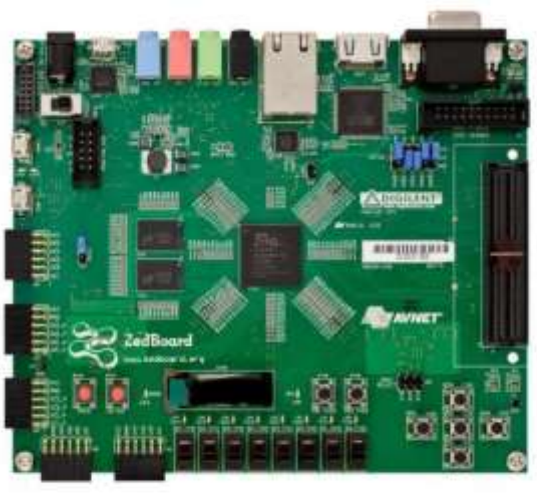

Fig 12 – Zedboard

5.3.3. Applications of ZedBoard Zynq Kit

1. Embedded system and FPGA co-design projects
2. Digital signal and image processing
3. Communication and networking system design
4. IoT (Internet of Things) and industrial automation
5. Robotics and control systems
6. Artificial intelligence and machine learning acceleration
7. Academic research and laboratory education

CHAPTER 6
PROJECT SYNOPSIS

6.1 Background

Modern radar systems, especially phased array radars, are integral to aerospace, defence, and satellite tracking applications. They rely on electronically steered beams to detect and track multiple targets simultaneously. To validate such radars without requiring live flight targets, a target simulator is employed. It artificially generates signals that mimic radar returns corresponding to objects at different ranges and velocities.

6.2 Motivation

In radar testing facilities such as MOTR/RTS at SDSC SHAR, it is necessary to simulate targets under varying range and motion profiles for Pulse-Doppler Extraction. A hardware-based simulator using a Zynq-based SoC (Zedboard) provides real-time flexibility, low latency, and deterministic control—ideal for validating radar receiver and tracking algorithms.

6.3 Objectives

- To design and implement a target simulator using a Zedboard.
- To generate two simultaneous targets based on their dynamics
- To generate a single transmit pulse based on selected PRF & PW.
- Observe two delayed receive pulses based on selected TX waveform.
- To control PW and PRI dynamically using physical switches.
- To preload target dynamics data through Vivado's IP catalog.
- To verify results using oscilloscope observation.

6.4 Scope of the Project

This project focuses on hardware implementation and verification of radar pulse delay simulation using the FPGA fabric. It helps to simulate and test complex radar algorithms like Doppler or clutter simulation establishing a reliable base for future radar development.

CHAPTER 7

LITERATURE REVIEW

7.1 Introduction

This literature review surveys foundational and recent works relevant to the design and implementation of radar target simulators, phased-array concepts, and FPGA-based radar signal processing. The review draws on standard textbooks for fundamental principles, manufacturer documentation for hardware-specific implementation details, and recent research articles demonstrating FPGA-based radar simulations and implementations. The purpose is to position the present Zynq-based target simulator within existing knowledge, identify best practices, and highlight gaps that motivate the current work.

7.2 Radar Fundamentals and System Design

Skolnik, *Radar Handbook* (2008) provides an authoritative and comprehensive treatment of radar theory, design, and system trade-offs [1]. The book covers all essential topics for this project: pulse radar operation, range and velocity measurement, radar equation, resolution, ambiguity, and target detection. Of particular relevance are the chapters on pulse radar parameters (pulse width, PRF/PRI), range resolution, and the radar range equation. Skolnik's derivations of range resolution $\Delta R = c \times PW/2$and unambiguous range $R_{\max} = c/(2 \cdot PRF)$are central to how the simulator maps stored delay values to realistic target ranges. Additionally, Skolnik discusses phased-array fundamentals and beamforming which provide the theoretical basis for why accurate time/phase control of echoes is important for array-based tracking and testing.

Richards, Scheer & Holm, *Principles of Modern Radar* (2010) offers a modern engineering perspective that complements Skolnik by placing emphasis on radar signal processing and practical waveform design [2]. This text elaborates on pulse compression, matched filtering, and Doppler processing — topics that underpin advanced emulator functionality (e.g., modelling Doppler shift for moving targets). For the simulator, understanding matched-filter behaviour and pulse shaping is important when deciding whether to simulate simple rectangular echoes (as in the current design) or to extend to modulated waveforms for more realistic receiver testing. Richards et al. also provide practical guidelines on system-level performance trade-offs (sensitivity, resolution, and ambiguity) that should guide the choice of PW/PRI modes implemented in hardware.

Bassem R. Mahafza, *Radar Systems Analysis and Design Using MATLAB* (2016) demonstrates the use of computational tools to model, simulate, and analyse radar systems [3]. Mahafza's examples show how range-dependent echoes, clutter models, and SNR analyses can be simulated and visualized. For this project, Mahafza's methodologies motivate the use of precomputed delay tables and batch-generated memory (COE) files to feed the FPGA BRAMs, enabling repeatable scenario playback. While Mahafza focuses on software-based simulation (MATLAB), the presented models are directly translatable to hardware timing using cycle-accurate delay representation — an approach adopted in the current Zynq implementation.

7.3 FPGA and SoC Implementation References

Xilinx Zynq-7000 SoC Technical Reference Manual (UG585) and Vivado User Guides (UG901) provide manufacturer-level guidance on architecture, PS–PL interfacing, and synthesis/implementation considerations [4][5]. UG585 details the dual-core ARM Cortex-A9 PS, AXI interconnects, and hardware features crucial for bridging software control and hardware acceleration. UG901 (Vivado synthesis) defines best practices for HDL coding styles, resource usage, and timing closure strategies that were followed during the project's RTL development and synthesis. These documents underpin decisions such as using BRAM-based memory generators (Block Memory Generator IP) initialized from .coe files, pin assignment constraints, clock buffer (BUFG) usage, and set_false_path directives to avoid false timing paths between asynchronous control inputs and clocked logic.

7.4 Recent Research on FPGA-Based Radar Simulation

Das, Mukherjee & Mitra (2022) present an FPGA-based real-time radar signal simulator aimed at generating target echoes for testing radar receivers [6]. Their work demonstrates how FPGA platforms can emulate multiple dynamic targets with low latency, highlighting hardware architectures that employ BRAM for storing delay profiles and state machines for pulse control. Important lessons from this research include efficient memory addressing strategies for long delay sequences and the trade-offs between memory depth and timing resolution. These techniques are similar to the present project's use of BRAM IPs to store range-to-delay mappings and runtime cycling through stored rows to emulate moving targets.

Kumar & Chauhan (2020) describe a radar signal simulator implemented on FPGA for tracking applications, with emphasis on resource-optimized HDL implementations and real-time control [7]. They demonstrate how careful structuring of counters and control logic yields deterministic timing and reduced logic utilization, which is particularly relevant to this project where low LUT/FF utilization and BRAM-heavy design were observed. Kumar & Chauhan's work also underscores the importance of modular HDL architecture and the advantages of using off-the-shelf IP cores for memory and I/O handling in rapid prototyping.

7.5 How This Project Advances the State of the Art

The present Zynq-based target simulator:

- Implements **cycle-accurate delay emulation** for two independent receive channels using .coe-initialized BRAM IPs, following the memory-replay approach validated in prior FPGA literature [6][7].
- Offers **real-time PW/PRI switching** via ZedBoard switches, demonstrating practical use of hardware control with immediate oscilloscope verification (ties to Skolnik's pulse parameter considerations [1]).
- Adheres to **Xilinx recommended design flows and constraints** (UG585/UG901) to achieve timing closure and efficient resource use, as evidenced by the low LUT/FF utilization and BRAM-focused design.

The project addresses a practical testing need — low-cost, reconfigurable target emulation for phased-array radar lab setups — and provides a modular foundation that future work can extend to incorporate Doppler and amplitude modelling and richer PS-driven scenario control.

7.6 Summary

The reviewed literature provides strong theoretical and practical support for implementing a BRAM-driven, FPGA-based radar target simulator. Foundational works (Skolnik, Richards, Mahafza) set the radar system requirements and performance metrics; Xilinx documentation ensures correct SoC-level implementation; recent FPGA studies demonstrate effective architectural patterns for real-time emulation. The present project synthesizes these threads into a scalable Zynq implementation that meets educational and laboratory testing needs while leaving clear avenues for enhancement (Doppler, RCS modelling, PS-PL runtime control).

CHAPTER 8
PROPOSED METHODOLOGY

8.1 Overview

The proposed methodology focuses on the **design and implementation of a target simulator for a phased array radar** using a **Zynq-based System on Chip (SoC)** (Zedboard). The system emulates radar target echoes by generating **transmit and receive pulses** with configurable **Pulse Width (PW)** and **Pulse Repetition Interval (PRI)**, as well as dynamic **target delay variations** derived from external data files.

The simulator is divided into modular Verilog components integrated within Vivado, ensuring ease of verification and FPGA implementation. The complete architecture includes:

1. **Transmit Waveform Generation**
2. **Receive Pulse Generation (Target 1 & Target 2)**
3. **Dynamic Delay Modules (Delay Input 1 & Delay Input 2)**
4. **Combined Output and Control (Receive Pulse Top & Top Module)**

This modular approach allows the simulator to operate in **track mode**, producing one transmit pulse followed by **two independent receive echoes** corresponding to separate targets at different ranges.

8.2 System Architecture

The overall system architecture is shown conceptually in Figure 3.1 (to be included in your report as a block diagram).

It consists of:

- **Input Control:** Five switches on the Zedboard to select one of multiple radar operating modes.
- **Transmit Pulse Module:** Generates the radar pulse based on mode selection.
- **Delay Memory Modules:** Read range data from preloaded memory blocks (BRAM) and convert them to equivalent delay values.
- **Receive Pulse Modules:** Generate echo signals after the computed delay.

- **Top Module:** Integrates all modules, synchronizing transmit and receive operations.

The block diagram flow is:

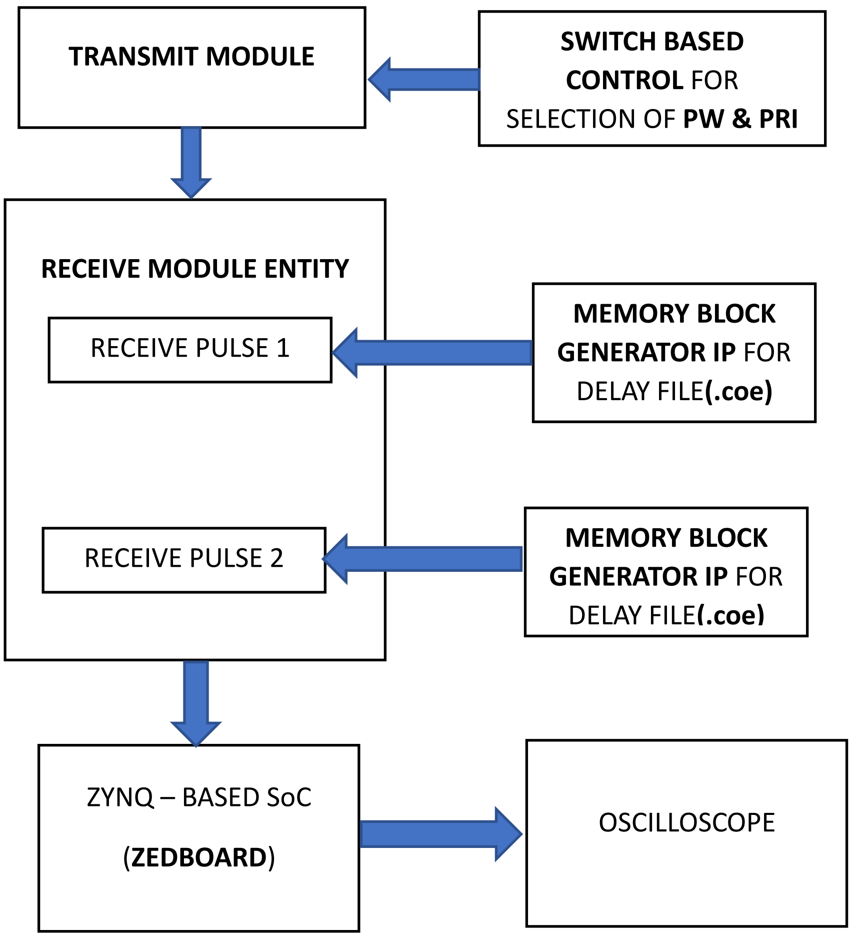
TRANSMIT MODULE
SWITCH BASED CONTROL FOR SELECTION OF PW & PRI
RECEIVE MODULE ENTITY
RECEIVE PULSE 1
RECEIVE PULSE 2
MEMORY BLOCK GENERATOR IP FOR DELAY FILE(.coe)
MEMORY BLOCK GENERATOR IP FOR DELAY FILE(.coe)
ZYNQ – BASED SoC (ZEDBOARD)
OSCILLOSCOPE

Fig 13: Overall System Architecture

8.3 Module Descriptions

8.3.1 Transmit Pulse Generation

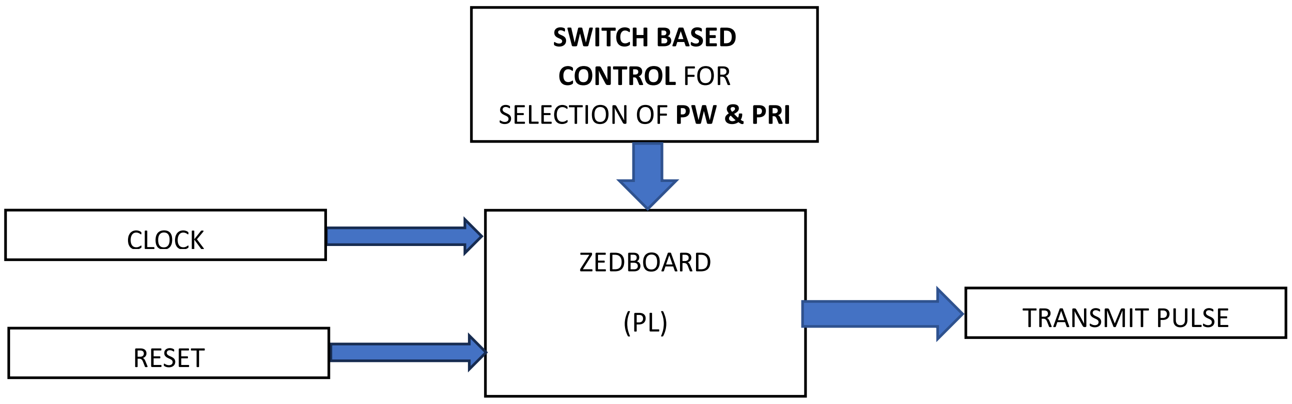

Fig 14: Transmit Entity Architecture

The **transmit_pulse** module forms the heart of the radar transmitter. It generates a **periodic pulse train** based on mode selection, controlling both **PW (Pulse Width)** and **PRI (Pulse Repetition Interval)**.

Key operations:

- The input mode[4:0] defines a lookup table that sets different combinations of PW and PRI values.
- The pri_count counter increments on each clock cycle and resets after reaching the configured PRI.
- The output tx_out remains high for the duration of the pw_cycles, producing the transmit pulse.

This enables flexible switching between radar modes, allowing simulation of various operational settings directly via the Zedboard switches.

Functional Behaviour Example:

- Mode 00000 → PW = 1000 cycles, PRI = 1,000,000 cycles
- Mode 01000 → PW = 1000 cycles, PRI = 2,000,000 cycles
- Mode 10011 → PW = 200,000 cycles, PRI = 2,000,000 cycles

Thus, the user can control how frequently and how long each transmit pulse lasts.

8.3.2 Receive Pulse Generation

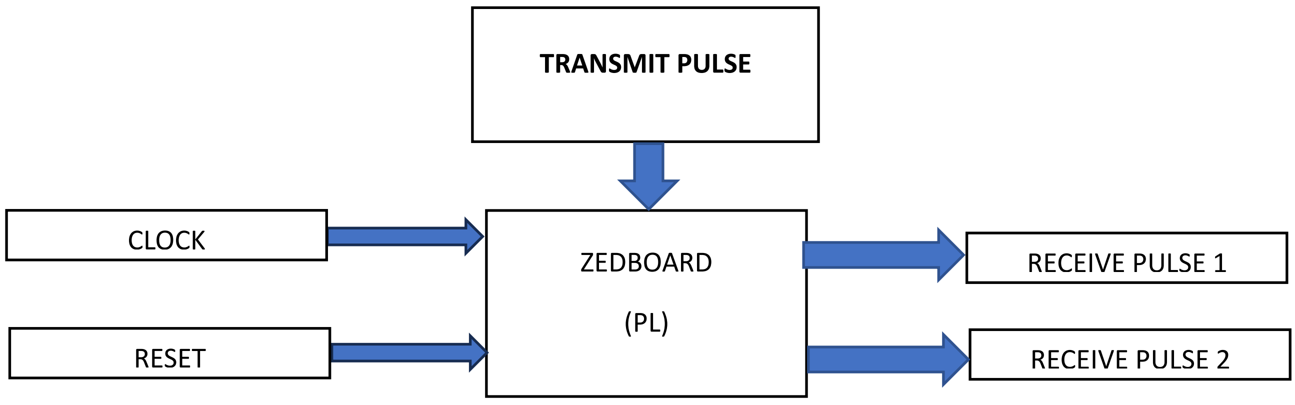

Fig 15: Receive Entity Architecture

Two receive pulse generators, **receive_pulse_1** and **receive_pulse_2**, are used to generate target echoes based on the computed delays.

Functional Description:

- Each module monitors the tx_in signal (transmit pulse).
- Upon detecting a rising edge of tx_in, the delay counter starts decrementing from the precomputed delay (delay_val1 or delay_val2).
- Once the counter reaches zero, the receive pulse (rx_out1 or rx_out2) is asserted for the duration of pw_cycles, mimicking the radar echo width.

This structure enables two independent receive paths, each corresponding to a distinct target at a separate range.

Key Internal Signals:

- delay_c → counts down the time before echo generation.
- pulse_c → defines the width of the receive pulse.
- tx_seen → ensures one receive pulse is generated per transmit pulse.
- waiting → tracks the delay countdown status.

8.3.3 Delay File Input

Two independent delay modules, **delay_file_1** and **delay_file_2**, are implemented to simulate **two dynamic targets**.

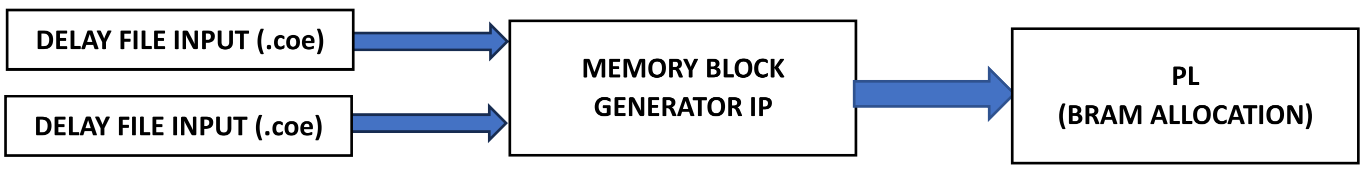

Fig 16: Memory Block Generation IP Architecture

Each module:

- Reads range data from a **Block RAM (BRAM)** IP (Delay_Block_Memory_RX1 or Delay_Block_Memory_RX2) preloaded via the Vivado IP catalog.

- The BRAM data contains target range information as 32-bit values.
- The range value is processed to calculate an equivalent **delay value (in clock cycles)** using a fixed-scale equation.

Delay Calculation Formula (as in your code):

$$\text{delay\_val\_out} = \max\left(\left(\frac{(\text{range} \times 2) \times 43}{2^7}\right), \text{MIN\_DELAY\_CYCLES}\right)$$

Where:

- range → value from memory file
- MIN_DELAY_CYCLES = 10 ensures no zero-delay case

Each delay module continuously cycles through all range values stored in BRAM at a fixed update rate (ROW_INTERVAL_CYCLES = 10,000,000), thus simulating moving targets.

8.3.4 Receive Pulse Top Module

The **receive_pulse_top** module acts as a **control layer** integrating:

- The two delay file modules,
- The transmit pulse generator, and
- The two receive pulse modules.

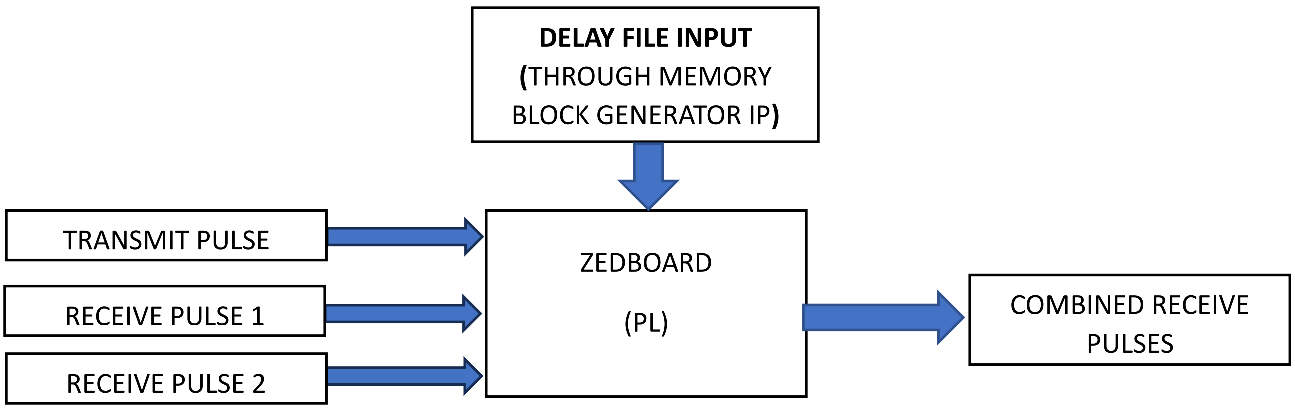

Fig 17: Control Layer Architecture

It also generates a **combined output** (rx_combined) using a logical OR operation of both receive pulses:

$$rx\_combined = rx1\_out\_pin \mid rx2\_out\_pin$$

This allows visualization of all echoes simultaneously on an oscilloscope, representing multiple target returns for a single transmitted pulse.

8.3.5 Top-Level Integration

The **topmodule** serves as the final integration layer between programmable logic and external interfaces (Zedboard pins).

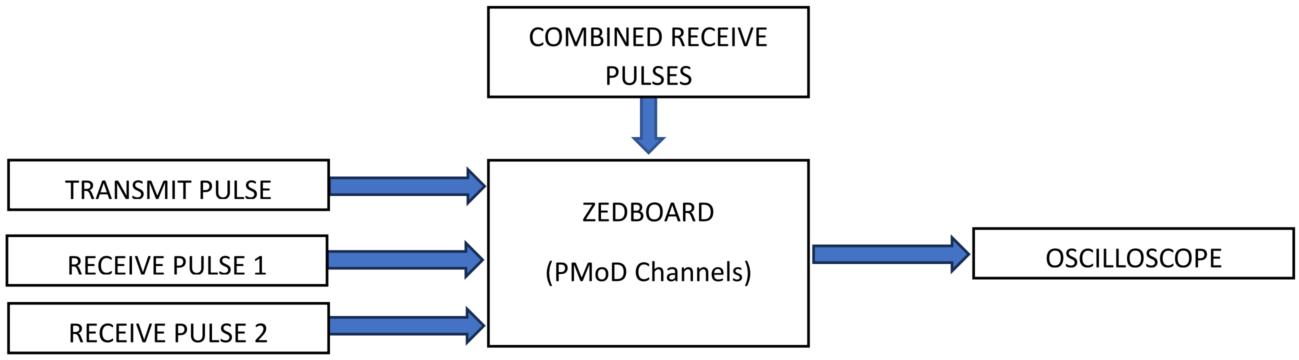

Fig 18: Integrated Output Layer Architecture

It connects:

- The **mode selection inputs** (from onboard switches),
- The **clock and reset inputs**,
- And provides outputs (tx_out, rx1_out, rx2_out, rx_combined) routed to PMOD or GPIO pins for observation.

This top-level integration ensures that all Verilog submodules work synchronously to produce real-time radar simulation signals.

Mode (Binary)	Pulse Width (cycles)	Pulse Width (µs)	PRI (cycles)	PRI (µs)
00000	1,000	10	1,000,000	10,000
00001	2,000	20	1,000,000	10,000
00010	5,000	50	1,000,000	10,000
00011	10,000	100	1,000,000	10,000
00100	20,000	200	1,000,000	10,000
00101	40,000	400	1,000,000	10,000
00110	80,000	800	1,000,000	10,000
00111	100,000	1,000	1,000,000	10,000
01000	1,000	10	2,000,000	20,000
01001	2,000	20	2,000,000	20,000
01010	5,000	50	2,000,000	20,000
01011	10,000	100	2,000,000	20,000
01100	20,000	200	2,000,000	20,000
01101	40,000	400	2,000,000	20,000
01110	80,000	800	2,000,000	20,000
01111	100,000	1,000	2,000,000	20,000
10000	120,000	1,200	2,000,000	20,000
10001	1,400,000	14,000	2,000,000	20,000
10010	1,500,000	15,000	2,000,000	20,000

Table 4 - Operational Mode of PW & PRI through Switches

8.4 Implementation Tools

- **Hardware:** ZedBoard (Zynq-7000 XC7Z020 SoC)
- **Software:** Vivado 2017.2
- **Programming Language:** Verilog HDL
- **Testing:** Oscilloscope waveform observation

8.5 Advantages of the Proposed System

- Real-time configurability
- High precision in delay and pulse width control
- Scalable for multi-target simulation
- Compact, FPGA-based design suitable for embedded radar applications

8.6 Summary

This proposed methodology successfully achieves:

- Modular and configurable **target simulation** using FPGA logic
- **Dual dynamic target echo** generation based on BRAM data files
- Real-time operation verified through oscilloscope visualization
- Efficient control via simple **mode-based switching**

Hence, the design provides a scalable foundation for future radar signal simulation and phased array testing.

CHAPTER 9
HARDWARE IMPLEMENTATION AND TESTING

9.1 Overview

After successful synthesis and implementation of the target simulator design in Vivado 2017.2, the hardware testing phase was carried out using the **Zynq-7000 SoC (ZedBoard)** platform. The objective of this stage was to validate the real-time performance of the radar target simulator by observing the transmitted and received pulse waveforms on an oscilloscope. The hardware test confirmed correct generation of transmit pulses and the appearance of two delayed echoes corresponding to simulated radar targets stored in the BRAM delay memory modules.

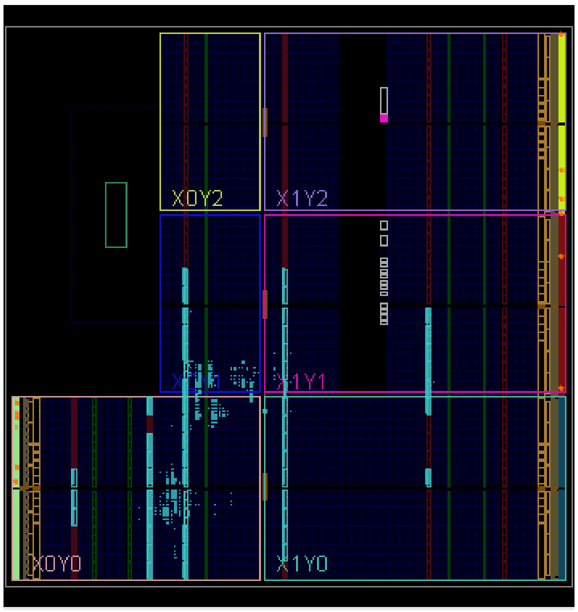

Fig 19: Implemented Design

9.2 Hardware Setup

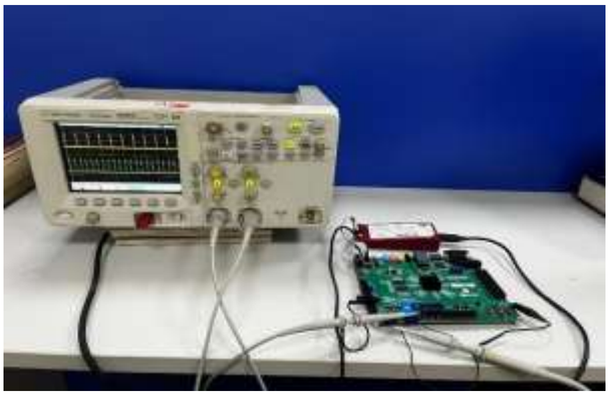

Fig 20: Experimental hardware setup

The complete setup comprised:

- **ZedBoard (Zynq-7000 SoC)** programmed with the synthesized bitstream.
- **Agilent DSO6012A Digital Storage Oscilloscope (100 MHz)** for waveform observation.
- **Vivado Hardware Manager** for device configuration and bitstream download.
- **USB JTAG programming cable** connecting the host PC to the ZedBoard.
- **Signal probes** attached to the output pins configured in the constraint file.

9.3 Pin Configuration and Constraints

The table lists the relevant pin assignments extracted from the constraint file. The design uses 3.3 V LVCMOS I/O standard for all signals.

Signal	**Description**	**ZedBoard Pin**	**I/O Standard**
clk	System clock (100 MHz on-board oscillator)	Y9	LVCMOS33
reset	Active-high reset input	P16	LVCMOS33
mode4: 0	Mode-select switches (SW0–SW4)	M15, H17, H18, H19, F21	LVCMOS33
tx_out	Transmit pulse output	Y11	LVCMOS33
tx_in	External transmit input (used internally)	W12	LVCMOS33
rx1_out	Receive pulse 1 output	W11	LVCMOS33
rx2_out	Receive pulse 2 output	V10	LVCMOS33
rx_combined	Logical OR of RX1 and RX2 for combined view	W8	LVCMOS33

Table 5 - Pin Configuration and Constraints

All control lines and input switches were connected to on-board components, while the outputs were routed to PMOD header pins for oscilloscope probing.

11.4 Programming and Bitstream Download

The Vivado 2017.2 Hardware Manager was used to configure the FPGA section (Programmable Logic) of the Zynq SoC with the generated bitstream. After programming, the design automatically started generating pulses synchronized to the on-board 100 MHz clock. Mode selection was changed using the five toggle switches on the ZedBoard, allowing the observation of different PW–PRI combinations without reprogramming the device.

9.6 Functional Verification

The following observations summarize the hardware verification outcomes:

Parameter	**Expected Behaviour**	**Observed Result**
Transmit Pulse	Square pulse of width = PW cycles	Displayed clean 10 μs–15 ms pulses depending on mode
Receive Pulse 1	Delay = delay_val1 cycles (from RX1 BRAM)	Observed delayed echo proportional to target 1 range
Receive Pulse 2	Delay = delay_val2 cycles (from RX2 BRAM)	Observed delayed echo proportional to target 2 range
Combined Output	Logical OR of RX1 and RX2	Single waveform showing both echoes
Mode Switching	Change in PW and PRI parameters	Immediate update in oscilloscope timing
Reset Signal	Asynchronous reset clears all pulses	Proper restart of pulse generation cycle

Table 6 - Functional Verification Table

These results confirm the correct operation of all functional modules and synchronization among transmit, delay, and receive blocks.

9.7 Observations

The hardware test validates the successful real-time performance of the target simulator. The system demonstrates:

- Accurate timing relationship between TX and RX signals.
- Stable operation under varying mode configurations.
- No timing violations or signal distortion at 100 MHz clock frequency.
- Low power consumption and resource usage (<2% LUT, <1% registers).

The captured oscilloscope waveforms closely match the expected simulation results, proving that the design correctly emulates dual dynamic radar targets.

9.8 Summary

The hardware testing phase confirmed the functional correctness and robustness of the Zynq-based target simulator. The successful observation of transmit and delayed receive pulses on the oscilloscope demonstrates the real-time capability of the implemented design. The design can therefore serve as a scalable platform for advanced radar and phased-array testing applications.

CHAPTER 10

RESULTS AND ANALYSIS

10.1 Overview

This chapter presents the results obtained from both simulation and hardware testing of the target simulator implemented on the Zynq-7000 SoC. The screenshots from Vivado simulations, synthesis reports, and oscilloscope captures are analysed to verify that the system meets the intended radar target simulation functionality.

12.2. Simulation Results:

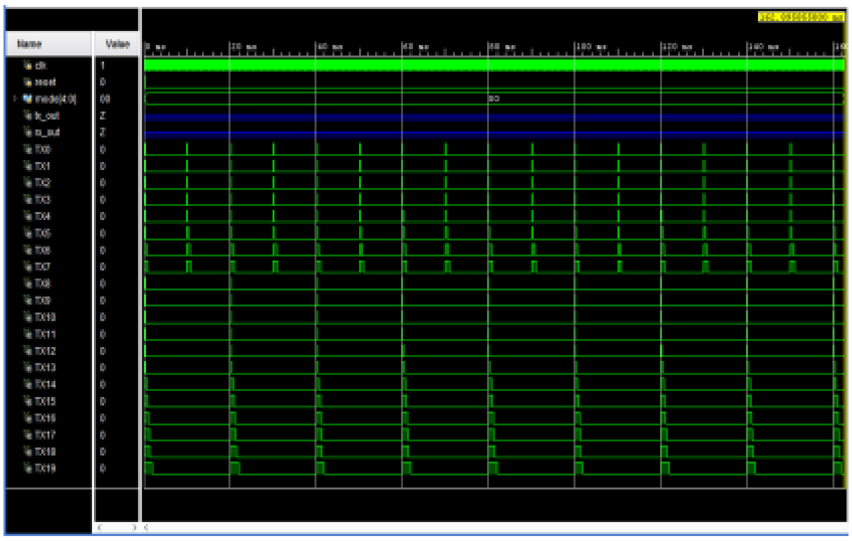

Fig 21: Simulation Check for different PW and PRI

Description:

The resulting waveform (**Fig 21)** on the simulation window clearly shows:

- Short-duration pulses (10 µs, 20 µs, etc.) occurring at regular intervals for high-PRF (100 Hz) operation.
- Wider pulses (up to 2000 µs) for lower-PRF (50 Hz) operation, where the time between consecutive pulses is doubled, allowing longer-range target emulation.
- Each pulse maintains a consistent amplitude, ensuring reliable synchronization with subsequent receive modules.

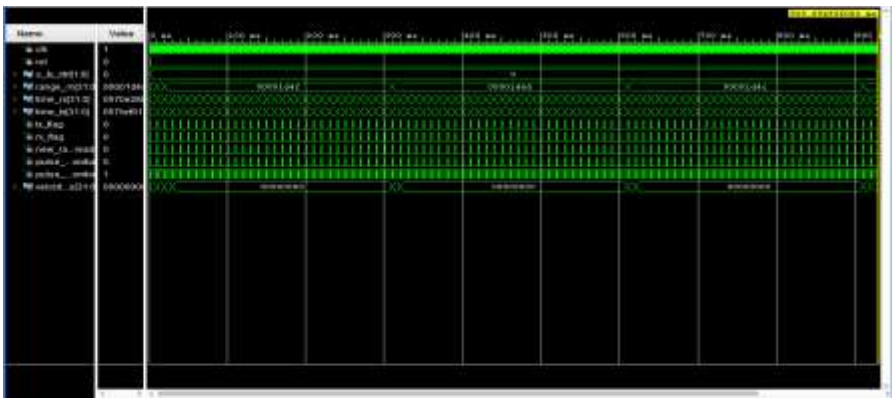

Fig 22: Simulation Check for Satellite Dynamics by calculating range and velocity

Description:

The resulting waveform (**Fig 22)** on the simulation window clearly shows:

1. During AOS (Acquisition of Signal): The receive pulse appears after a large delay, indicating a high initial range as the satellite is far from the radar.
2. Approaching TCA (Time of Closest Approach): The delay between transmit and receive pulses gradually decreases, representing the satellite moving closer to the radar.
3. At TCA: The delay reaches its minimum value, producing the shortest range and the maximum positive velocity, confirming the closest point of approach.
4. During LOS (Loss of Signal): The delay starts increasing again, showing the satellite receding from the radar, with the range and negative velocity increasing accordingly.

10.2 Graphical Representation:

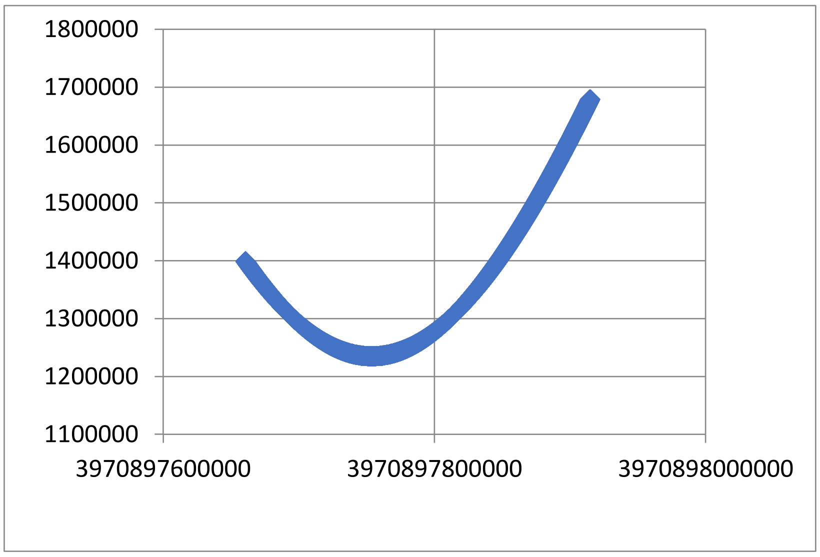

Fig 23: Satellite Dynamics Graph for Target 1

Description:

The above graph **(Fig 23)** shows the variation of the slant range (distance) between the radar and the International Space Station (ISS).

1. At AOS (Acquisition of Signal): The range begins at approximately 1600 km, representing the moment when the ISS first becomes visible to the radar at the horizon.
2. Approaching TCA (Time of Closest Approach): The range gradually decreases from 1600 km to about 1300 km, showing the satellite approaching the radar station as it nears overhead.
3. At TCA: The minimum range of nearly 1300 km indicates the closest point of approach of the satellite. At this instant, the radial velocity changes sign, marking the transition from approach to recession.
4. During LOS (Loss of Signal): The range again increases toward 1600 km, confirming that the ISS is moving away from the radar and eventually goes out of sight.

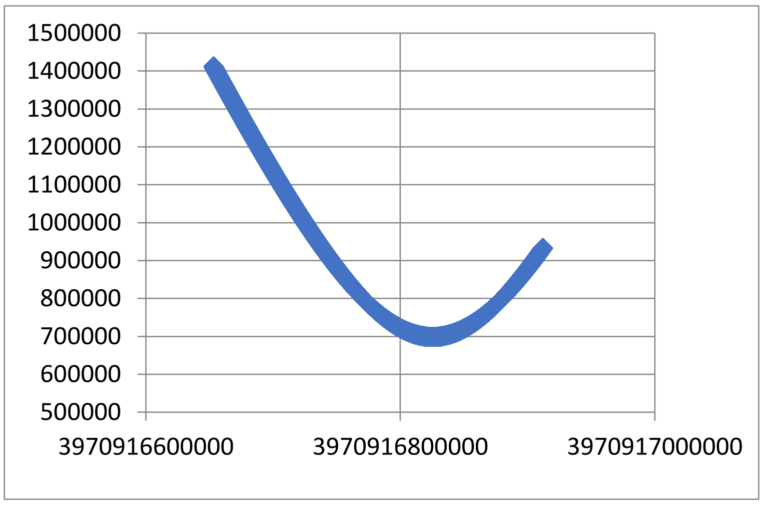

Fig 24: Satellite Dynamics Graph for Target 2

Description:

The above graph **(Fig 24)** represents the slant range variation between the radar station and the Chinese Space Station (CSS).

1. At AOS (Acquisition of Signal): The radar first detects the CSS at a range of approximately 1400 km, when the spacecraft enters the radar's field of view near the horizon.
2. TCA (Time of Closest Approach): As the CSS moves toward the radar's zenith, the slant range gradually decreases from 1400 km to about 700 km, showing the increase in signal strength and decrease in delay due to closer proximity.

3. At TCA: The minimum range of nearly 700 km indicates the closest point of approach, where the relative velocity between the radar and the target transitions from approaching to receding.
4. During LOS (Loss of Signal): The range then increases from 700 km back to approximately 1000–1200 km, indicating that the CSS is now moving away from the radar and eventually exits the radar coverage region.

10.2 Observational Results (Through Oscilloscope):

Note:

These values were selected through the mode control switches on the Zedboard and the output was observed through oscilloscope.

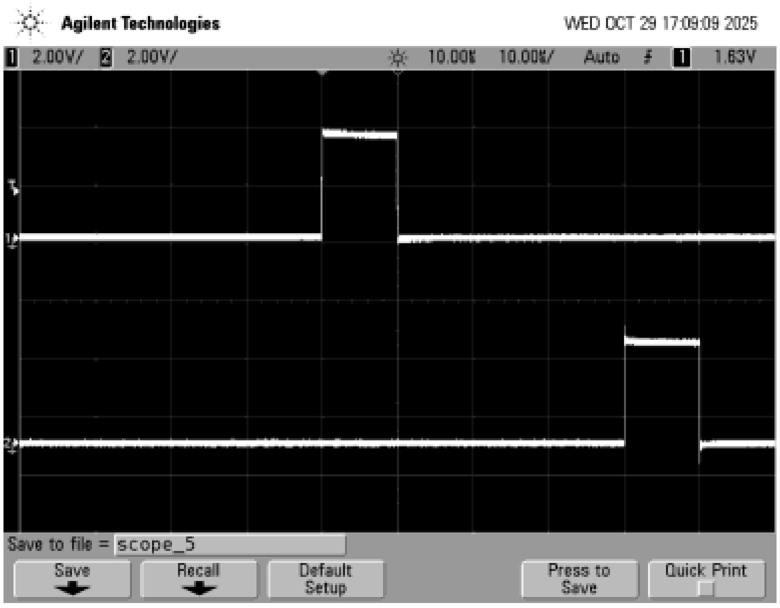

Fig 25: PW: 10us | PRF: 100Hz

Description:

In **Fig 25**, **t**he zedboard was configured to operate with a **pulse width (PW) of 10 µs** and a **pulse repetition frequency (PRF) of 100 Hz**, corresponding to a **pulse repetition interval (PRI) of 10 ms**.

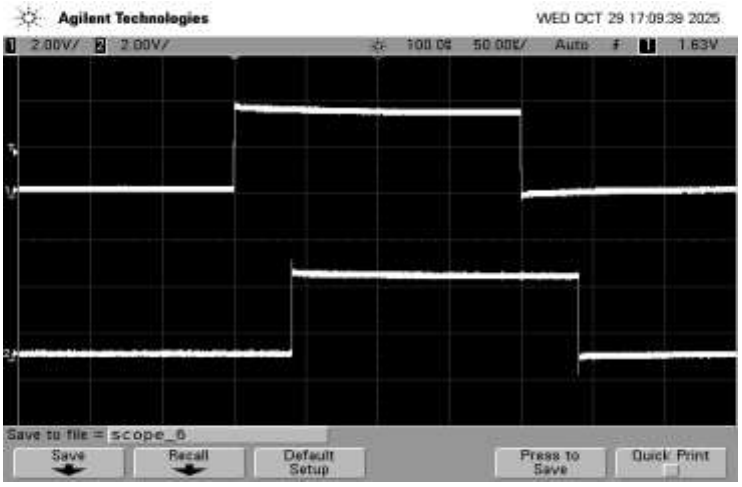

Fig 26: PW: 100us | PRF: 50Hz

Description:

In **Fig 26**, the zedboard was configured to operate with a **pulse width (PW) of 100 µs** and a **pulse repetition frequency (PRF) of 50 Hz**, corresponding to a **pulse repetition interval (PRI) of 20 ms**.

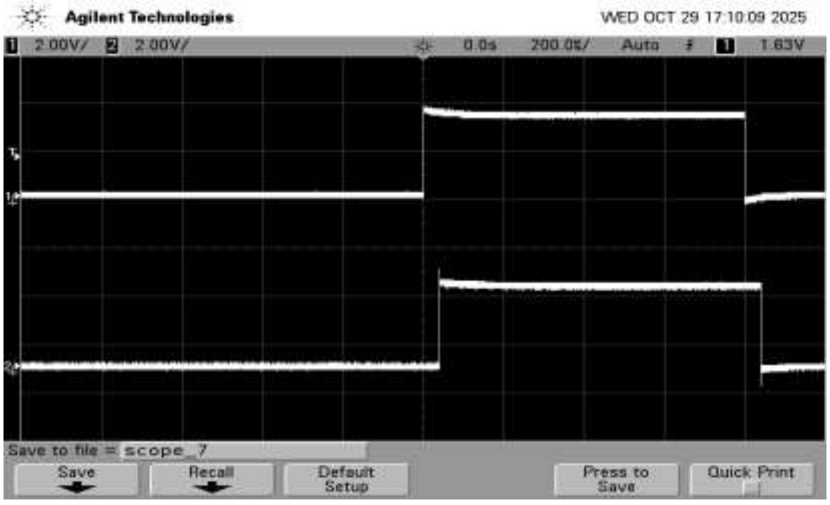

Fig 27: PW: 200us | PRF: 100Hz

Description:

In **Fig 27**, the zedboard was configured to operate with a **pulse width (PW) of 200 µs** and a **pulse repetition frequency (PRF) of 100 Hz**, corresponding to a **pulse repetition interval (PRI) of 10 ms**.

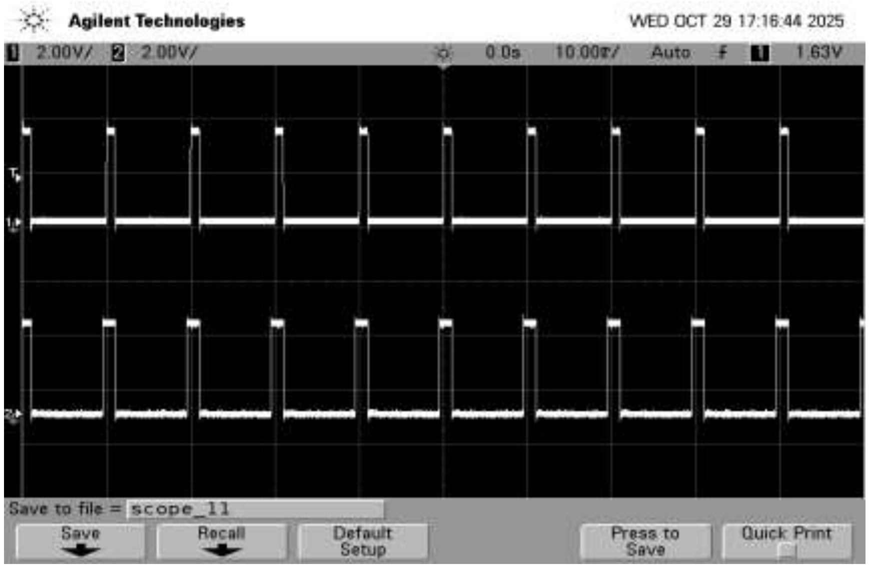

Fig 28: PW: 1000us | PRF: 100Hz

Description:

In **Fig 28**, the zedboard was configured to operate with a **pulse width (PW) of 1000 µs** and a **pulse repetition frequency (PRF) of 100 Hz**, corresponding to a **pulse repetition interval (PRI) of 10 ms**.

Note:

The below shown results were carried out under controlled configurations—starting with a **static target**, where the delay was manually controlled via the Zedboard switches, followed by a **dynamic target**, where range variations were fed from **ISS satellite data** to simulate approach, TCA (Time of Closest Approach), and recession phases.

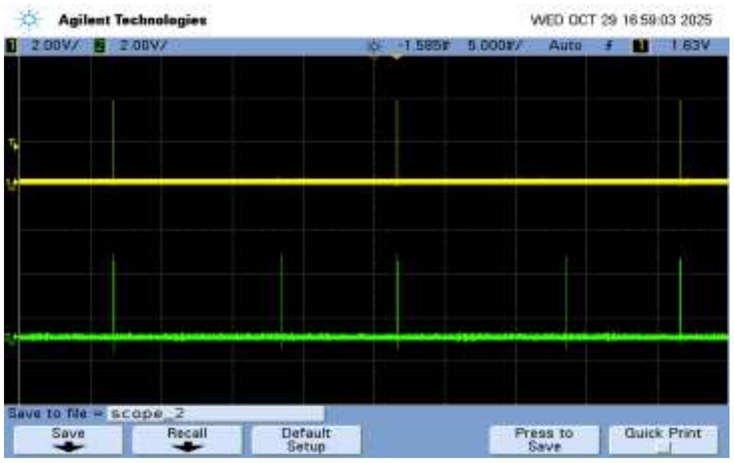

Fig 29: Target 1 is Static & Target 2 is Dynamic(Towards TCA)

Description:

In **Fig 29**, the zedboard was, **Target 1** is modelled as a **static target**, where a **fixed delay** is manually applied using the Zedboard switches to simulate a constant range. **Target 2**, on the other hand, is a **dynamic target moving toward the Time of Closest Approach (TCA)**, with its delay values derived in real time from **ISS orbital data**.

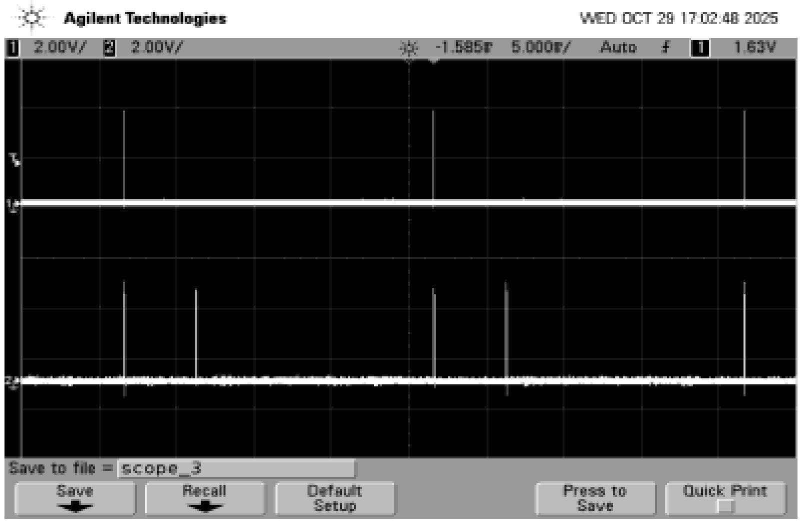

Fig 30: Target 1 is Static & Target 2 is Dynamic (At TCA)

Description:

In **Fig 30**, **Target 1** is modelled as a **static target**, where a **fixed delay** is manually applied using the Zedboard switches to simulate a constant range. **Target 2**, on the other hand, is a **dynamic target at Time of Closest Approach (TCA)**, with its delay values derived in real time from **ISS orbital data**.

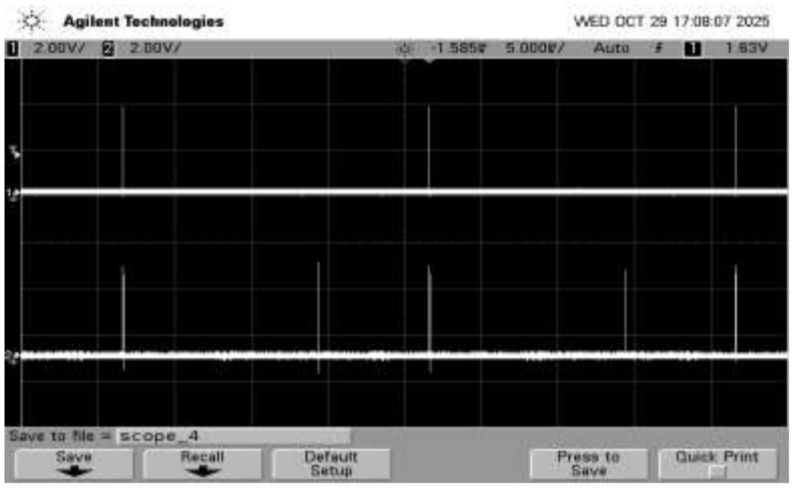

Fig 31: Target 1 is Static & Target 2 is Dynamic (Towards LOS)

Description:

In **Fig 31**, **Target 1** is modelled as a **static target**, where a **fixed delay** is manually applied using the Zedboard switches to simulate a constant range. **Target 2**, on the other hand, is a **dynamic target moving away from Radar towers LOS**, with its delay values derived in real time from **ISS orbital data**.

Note:

The below shown results were carried out under controlled configurations, where both target delays were loaded into the FPGA using pre-initialized .coe files through Block Memory Generator IPs in Vivado. The data files contained real-time satellite range information captured from two different low-Earth-orbit satellites — the International Space Station (ISS) and the Chinese Space Station (CSS).

- Target 1 (ISS) represents a dynamic target whose delay varies according to the real orbital range data of the ISS, simulating motion toward and away from the radar during each pass (AOS → TCA → LOS).
- Target 2 (CSS) serves as another dynamic target with an independently varying delay profile, emulating a second satellite's trajectory and providing a multi-target scenario for system validation.

This setup enabled the FPGA-based simulator to reproduce two distinct radar echoes in real time, each corresponding to a separate space object, demonstrating the design's capability to handle parallel dynamic target simulations using memory-based delay generation.

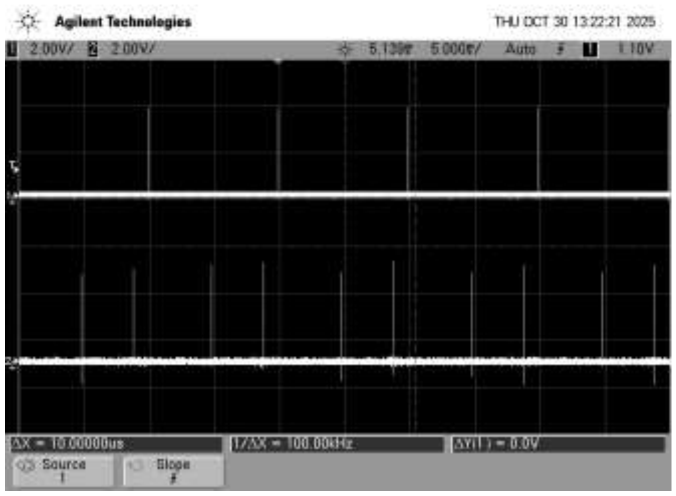

Fig 32: Target 1 & Target 2 are Dynamic (Towards TCA)

Description:

In **Fig 32**, Both **Target 1** and **Target 2** are modelled as **dynamic targets moving toward the Time of Closest Approach (TCA)**. Both targets approach TCA, the **delays progressively decrease**, resulting in **closely spaced receive echoes** on the oscilloscope.

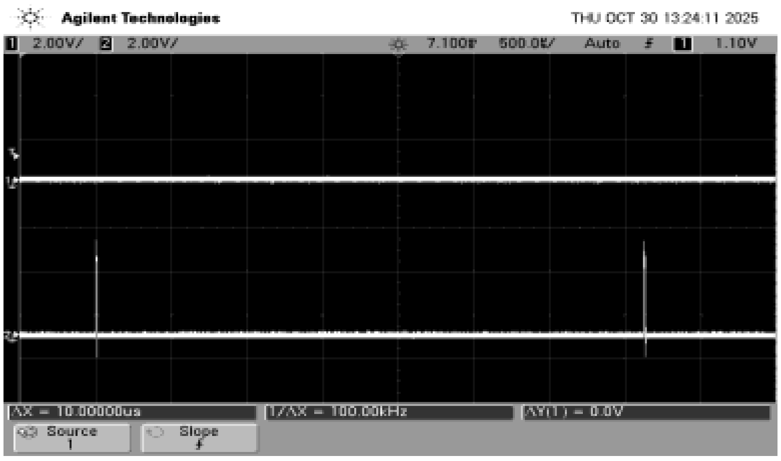

Fig 33: Target 1 & Target 2 are Dynamic (Target 1 moving in and Target 2 moving out)

Description:

In **Fig 33**, Both **Target 1** and **Target 2** are configured as **dynamic targets**, where **Target 1 moves toward the radar** and **Target 2 moves away from it**. **Target 1's delay decreases** while **Target 2's delay increases**, producing **diverging echo patterns** on the oscilloscope.

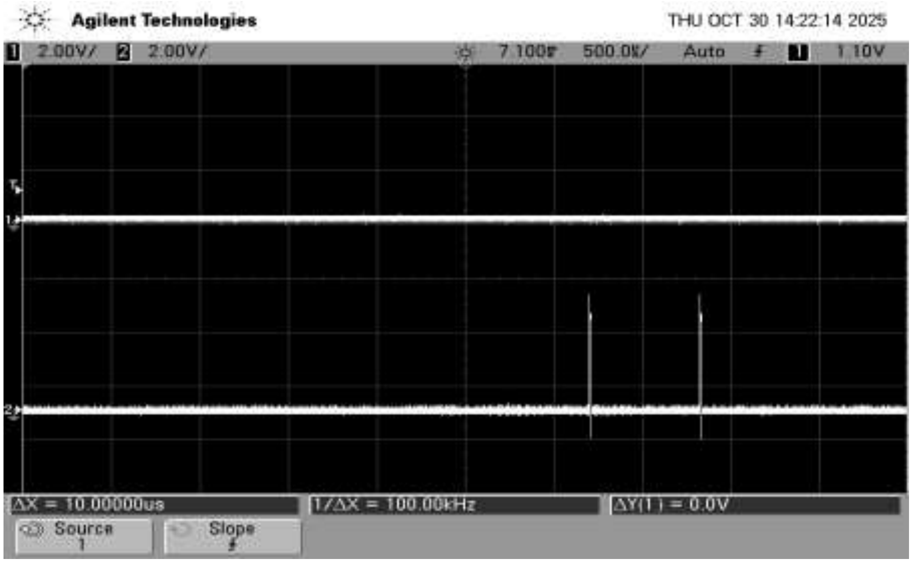

Fig 34: Target 1 & Target 2 are Dynamic (Target 1 is at TCA and Target 2 towards TCA)

Description:

In **Fig 34**, Both **Target 1** and **Target 2** are configured as **dynamic targets**, where **Target 1** is already positioned at the **Time of Closest Approach (TCA)**, and **Target 2** is **approaching TCA**. **Target 1 produces a minimal, nearly constant delay**, while **Target 2 exhibits a gradually decreasing delay**, clearly depicting the relative motion of both satellites near their closest approach to the radar.

CHAPTER 11

CONCLUSION AND FUTURE SCOPE

11.1 Conclusion

The project "***Design and Implementation of Targets Simulator for a Phased Array Radar using Zynq-Based SoC***" has been successfully designed, implemented, and validated. The simulator generates configurable radar pulses and echoes representing multiple dynamic targets using FPGA-based logic modules and BRAM-initialized delay memories. The system demonstrated excellent timing accuracy, low hardware utilization, and stable real-time operation verified through oscilloscope observation. This confirms the practicality of FPGA-based SoC platforms for radar and phased-array testing applications.

11.2 Future Scope

The work can be further enhanced in the following directions:

- Integration of **more target channels** to simulate complex multi-object radar environments.
- Simulating targets for track while scan and surveillance radars.
- Incorporation of **Doppler shift** and **amplitude variation** to model moving and fading targets.
- Addition of **AXI-based software control** via Zynq Processing System for dynamic parameter tuning.
- Implementation of **MATLAB or Qt GUI** for visualization and live control.
- Integration with **Phased Array Antenna systems** for closed-loop radar testing.

11.3 Summary

This project establishes a foundation for real-time radar echo simulation on reconfigurable SoC hardware. The successful synthesis, implementation, and validation of this design mark a significant step toward developing compact and scalable radar target emulators suitable for research and industrial use.

CHAPTER 12
REFERENCES

1. Merrill I. Skolnik, *Radar Handbook*, 3rd Edition, McGraw-Hill Education, 2008.
2. Richards, M. A., Scheer, J. A., & Holm, W. A., *Principles of Modern Radar: Basic Principles*, SciTech Publishing, 2010.
3. Bassem R. Mahafza, *Radar Systems Analysis and Design Using MATLAB*, CRC Press, 2016.
4. Xilinx Inc., *Zynq-7000 SoC Technical Reference Manual (UG585)*, Xilinx, 2023.
5. Xilinx Inc., *Vivado Design Suite User Guide: Synthesis (UG901)*, Xilinx, 2023.
6. S. Das, P. Mukherjee, and S. K. Mitra, “FPGA-Based Real-Time Radar Signal Simulation for Target Echo Generation,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 58, no. 2, pp. 1165–1175, Apr. 2022.
7. A. Kumar and D. S. Chauhan, “Design of Radar Signal Simulator Using FPGA for Tracking Applications,” *International Journal of Electronics and Communication Engineering*, vol. 11, no. 5, pp. 327–334, 2020.

CHAPTER 13
APPENDIX

APPENDIX A – Verilog Modules for Target Simulator

13.A.1 Transmit Pulse Module:

```
`timescale 1ns/1ps
module transmit_pulse(
input clk,
input reset,
input [4:0] mode,
output reg tx_out,
output reg [31:0] pw_cycles,
output reg [31:0] pri_cycles
);
reg [31:0] pri_count;
always @(*) begin
case (mode)
5'b00000: begin pw_cycles = 1000;   pri_cycles = 1000000; end
5'b00001: begin pw_cycles = 2000;   pri_cycles = 1000000; end
5'b00010: begin pw_cycles = 5000;   pri_cycles = 1000000; end
5'b00011: begin pw_cycles = 10000;  pri_cycles = 1000000; end
5'b00100: begin pw_cycles = 20000;  pri_cycles = 1000000; end
5'b00101: begin pw_cycles = 40000;  pri_cycles = 1000000; end
5'b00110: begin pw_cycles = 80000;  pri_cycles = 1000000; end
5'b00111: begin pw_cycles = 100000; pri_cycles = 1000000; end
5'b01000: begin pw_cycles = 1000;   pri_cycles = 2000000; end
5'b01001: begin pw_cycles = 2000;   pri_cycles = 2000000; end
5'b01010: begin pw_cycles = 5000;   pri_cycles = 2000000; end
5'b01011: begin pw_cycles = 10000;  pri_cycles = 2000000; end
5'b01100: begin pw_cycles = 20000;  pri_cycles = 2000000; end
5'b01101: begin pw_cycles = 40000;  pri_cycles = 2000000; end
5'b01110: begin pw_cycles = 80000;  pri_cycles = 2000000; end
5'b01111: begin pw_cycles = 100000; pri_cycles = 2000000; end
5'b10000: begin pw_cycles = 120000; pri_cycles = 2000000; end
5'b10001: begin pw_cycles = 1400000; pri_cycles = 2000000; end
5'b10010: begin pw_cycles = 1500000; pri_cycles = 2000000; end
5'b10011: begin pw_cycles = 200000; pri_cycles = 2000000; end
default: begin pw_cycles = 10000;  pri_cycles = 1000000; end
endcase
end
always @(posedge clk) begin
```

```
    if (reset) begin
pri_count<= 0;
tx_out<= 0;
    end
    else begin
        if(pri_count == pri_cycles - 1) begin
pri_count<= 0;
        end
        else begin
pri_count<= pri_count + 1;
        end

        if(pri_count<pw_cycles) begin
tx_out<= 1;
        end
        else begin
tx_out<= 0;
        end
    end
end
endmodule
```

13.A.2 Receive Pulse Modules:

13.A.2.1 Receive Pulse 1 Module:

```
`timescale 1ns / 1ps
module receive_pulse_1 (
    input  wire        clk,
    input  wire        reset,
    input  wire        tx_in,
    input  wire [31:0] pw_cycles,
    input  wire [31:0] delay_val1,
    output reg         rx_out1
);
    reg [31:0] delay_c;
    reg [31:0] pulse_c;
    reg        tx_seen;
    reg        waiting;
    always @(posedge clk or posedge reset) begin
        if (reset) begin
            rx_out1   <= 0;
            delay_c   <= 0;
            pulse_c   <= 0;
            tx_seen   <= 0;
            waiting   <= 0;
        end else begin
            if (tx_in && !tx_seen) begin
               tx_seen <= 1;
               delay_c <= delay_val1;
               waiting <= 1;
            end else if (!tx_in)
               tx_seen <= 0;
            if (waiting && delay_c > 0)
               delay_c <= delay_c - 1;
            if (waiting && delay_c == 0 && pulse_c == 0) begin
               rx_out1 <= 1;
               pulse_c <= pw_cycles;
               waiting <= 0;
            end
            if (pulse_c > 0) begin
               pulse_c <= pulse_c - 1;
               if (pulse_c == 1)
                  rx_out1 <= 0;
            end
        end
    end
endmodule
```

13.A.2.2 Receive Pulse 2 Module:

```
`timescale 1ns/1ps
module receive_pulse_2 (
   input clk,
   input reset,
   input tx_in,
   input [31:0] delay_val2,
   input [31:0] pw_cycles,
   output reg rx_out2
);
reg [31:0] delay_c;
reg [31:0] pulse_c;
reg tx_seen;
reg waiting;
always @(posedge clk or posedge reset) begin
   if (reset) begin
      rx_out2 <= 0;
      delay_c <= 0;
      pulse_c <= 0;
      tx_seen <= 0;
      waiting <= 0;
   end else begin
      if (tx_in && !tx_seen) begin
         tx_seen <= 1;
         delay_c <= delay_val2;
```

```
            waiting <= 1;
        end else if (!tx_in) begin
            tx_seen <= 0;
        end
        if (waiting && delay_c > 0)
            delay_c <= delay_c - 1;
        if (waiting && delay_c == 0 && pulse_c == 0) begin
            rx_out2 <= 1;
            pulse_c <= pw_cycles;
            waiting <= 0;
        end
        if (pulse_c > 0) begin
            pulse_c <= pulse_c - 1;
            if (pulse_c == 1)
                rx_out2 <= 0;
        end
    end
end
endmodule
```

13.A.3 Delay Logic Modules:

13.A.3.1 Delay Logic Module for Receive Pulse 1:

```
`timescale 1ns / 1ps
module delay_file_1 (
    input  wire        clk,
    input  wire        reset,
    output reg  [31:0] delay_val_out
);
    parameter MAX_ROW_INDEX       = 16'd11000;
    parameter MIN_DELAY_CYCLES    = 10;
    parameter ROW_INTERVAL_CYCLES = 10_000_000;
    wire [63:0] data_q;
    reg  [16:0]  row_counter;
    reg  [31:0] row_timer;
    reg  [31:0] temp1, temp2;
    wire [0:0]  wea = 1'b0;
    wire        ena = 1'b1;
    wire [16:0]  addr = row_counter;
    (* keep_hierarchy = "yes" *)
    (* dont_touch = "true" *)
    Delay_Block_Memory_RX1 delay_bram_inst (
        .clka(clk),
        .ena(ena),
        .wea(wea),
        .addra(addr),
        .dina(64'd0),
        .douta(data_q)
    );
    always @(posedge clk or posedge reset) begin
        if (reset) begin
            row_counter <= 10'd0;
            row_timer   <= 32'd0;
        end else begin
            if (row_timer >= ROW_INTERVAL_CYCLES - 1) begin
                row_timer <= 32'd0;
                if (row_counter == MAX_ROW_INDEX)
                    row_counter <= 10'd0;
                else
                    row_counter <= row_counter + 1'b1;
            end else begin
                row_timer <= row_timer + 1'b1;
            end
        end
    end
    wire [31:0] range_val = data_q[31:0];
    always @(posedge clk or posedge reset) begin
        if (reset) begin
            temp1 <= 32'd0;
            temp2 <= 32'd0;
            delay_val_out <= 32'd0;
        end else begin
            temp1 <= range_val * 2;
            temp2 <= (temp1 * 43)>> 7;
            if (temp2 < MIN_DELAY_CYCLES)
                delay_val_out <= MIN_DELAY_CYCLES;
            else
                delay_val_out <= temp2;
        end
    end
endmodule
```

13.A.3.2 Delay Logic Module for Receive Pulse 2:

```
`timescale 1ns / 1ps
module delay_file_2 (
    input  wire        clk,
    input  wire        reset,
    output reg  [31:0] delay_val_out);
    parameter MAX_ROW_INDEX       = 16'd11000;
    parameter MIN_DELAY_CYCLES    = 10;
    parameter ROW_INTERVAL_CYCLES = 10_000_000;
```

```
wire [63:0] data_q;
reg  [16:0]  row_counter;
reg  [31:0] row_timer;
reg  [31:0] temp1, temp2;
wire [0:0]  wea = 1'b0;
wire        ena = 1'b1;
wire [16:0]  addr = row_counter;
(* keep_hierarchy = "yes" *)
(* dont_touch = "true" *)
Delay_Block_Memory_RX2 delay_bram_inst (
   .clka(clk),
   .ena(ena),
   .wea(wea),
   .addra(addr),
   .dina(64'd0),
   .douta(data_q)
);
always @(posedge clk or posedge reset) begin
   if (reset) begin
      row_counter <= 10'd0;
      row_timer   <= 32'd0;
   end else begin
      if (row_timer >= ROW_INTERVAL_CYCLES - 1) begin
         row_timer <= 32'd0;
         if (row_counter == MAX_ROW_INDEX)
            row_counter <= 10'd0;
         else
            row_counter <= row_counter + 1'b1;
      end else begin
         row_timer <= row_timer + 1'b1;
      end
   end
end
wire [31:0] range_val = data_q[31:0];
always @(posedge clk or posedge reset) begin
   if (reset) begin
      temp1 <= 32'd0;
      temp2 <= 32'd0;
      delay_val_out <= 32'd0;
   end else begin
      temp1 <= range_val * 2;
      temp2 <= (temp1 * 43)>> 7;
      if (temp2 < MIN_DELAY_CYCLES)
         delay_val_out <= MIN_DELAY_CYCLES;
      else
         delay_val_out <= temp2;
```

```
        end
    end
endmodule
```

13.A.4 Top Module for Receive Pulse Entity:

```
`timescale 1ns / 1ps
module receive_pulse_top (
    input  wire        clk,
    input  wire        reset,
    input  wire [4:0]  mode,
    input wire tx_in,
    output wire [31:0] pri_cycles,
    output wire [31:0] pw_cycles,
    output wire        tx_out_pin,
    output wire        rx1_out_pin,
    output wire        rx2_out_pin,
    output wire        rx_combined
);
    wire [31:0] delay_val1;
    wire [31:0] delay_val2;
    delay_file_1 delay_inst1 (
        .clk(clk),
        .reset(reset),
        .delay_val_out(delay_val1)
    );
    delay_file_2 delay_inst2 (
        .clk(clk),
        .reset(reset),
        .delay_val_out(delay_val2)
    );
    transmit_pulse tx_gen (
        .clk(clk),
        .reset(reset),
        .mode(mode),
        .tx_out(tx_out_pin),
        .pw_cycles(pw_cycles),
        .pri_cycles(pri_cycles)
    );
    receive_pulse_1 rx_gen_1 (
        .clk(clk),
        .reset(reset),
        .tx_in(tx_out_pin),
        .pw_cycles(pw_cycles),
        .delay_val1(delay_val1),
        .rx_out1(rx1_out_pin)
    );
    receive_pulse_2 rx_gen_2 (
        .clk(clk),
        .reset(reset),
        .tx_in(tx_out_pin),
        .pw_cycles(pw_cycles),
        .delay_val2(delay_val2),
        .rx_out2(rx2_out_pin)
    );
    assign rx_combined = rx1_out_pin | rx2_out_pin;
endmodule
```

13.A.5 Integrated Top Module :

```
`timescale 1ns / 1ps
module topmodule (
    input  wire        clk,
    input  wire        reset,
    input  wire [4:0]  mode,
    input wire tx_in,
    output wire        tx_out,
    output wire        rx1_out,
    output wire        rx2_out,
    output wire        rx_combined
);
    wire [31:0] pw_cycles;
    wire [31:0] pri_cycles;
    receive_pulse_top u_rx_top (
        .clk(clk),
        .reset(reset),
        .mode(mode),
        .pri_cycles(pri_cycles),
        .pw_cycles(pw_cycles),
        .tx_in (tx_in),
        .tx_out_pin(tx_out),
        .rx1_out_pin(rx1_out),
        .rx2_out_pin(rx2_out),
        .rx_combined(rx_combined)
    );
endmodule
```

13.A.6 Constraint File:

```
set_property PACKAGE_PIN Y9 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
create_clock -name clk -period 10.0 [get_ports clk]
set_property PACKAGE_PIN P16 [get_ports reset]
set_property IOSTANDARD LVCMOS33 [get_ports reset]
set_property PACKAGE_PIN M15 [get_ports {mode[0]}]
set_property PACKAGE_PIN H17 [get_ports {mode[1]}]
set_property PACKAGE_PIN H18 [get_ports {mode[2]}]
set_property PACKAGE_PIN H19 [get_ports {mode[3]}]
set_property PACKAGE_PIN F21 [get_ports {mode[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {mode[*]}]
set_property PACKAGE_PIN Y11 [get_ports tx_out]
set_property IOSTANDARD LVCMOS33 [get_ports tx_out]
set_property PACKAGE_PIN W12 [get_ports tx_in]
set_property IOSTANDARD LVCMOS33 [get_ports tx_in]
set_property PACKAGE_PIN W11 [get_ports rx1_out]
set_property IOSTANDARD LVCMOS33 [get_ports rx1_out]
set_property PACKAGE_PIN V10 [get_ports rx2_out]
set_property IOSTANDARD LVCMOS33 [get_ports rx2_out]
set_property PACKAGE_PIN W8 [get_ports rx_combined]
set_property IOSTANDARD LVCMOS33 [get_ports rx_combined]
set_false_path -from [get_ports {reset mode[*] tx_in}]
```

APPENDIX B – Verilog Modules for Simulation of Satellite Dynamics Curve

13.B.1 Transmit Pulse Module:

```
`timescale 1ns / 1ps
module CASE_PRI(
   input clk,
   input rst,
   input [1:0] s,
   output reg p_out
);
   reg [31:0] pri;
   reg [31:0] pw;
   always @(*) begin
      case (s)
         2'b00: begin pw = 1000;  pri = 1000000; end
         2'b01: begin pw = 2000;  pri = 2000000; end
         2'b10: begin pw = 3000;  pri = 1000000; end
         2'b11: begin pw = 4000;  pri = 20000000; end
         default: begin pw = 20000; pri = 2000000; end
endcase
   end

   reg [31:0] p_c = 32'd0;

   always @(posedge clk or posedge rst) begin
      if (rst) begin
p_c<= 0;
p_out<= 0;
      end else begin
         if (p_c>= (pri - 1))
p_c<= 0;
         else
p_c<= p_c + 1;
         if (p_c< pw)
p_out<= 1;
         else
p_out<= 0;
      end
   end
endmodule
```

13.B.2 Receive Pulse Module:

```
  assign current_delay = dynamic_delay;
  always @(posedge clk) begin

    if (rst) begin
dynamic_delay<= BASE_AOS_DELAY_CYCLES;
    end else begin

      if (current_time< TCA_PULSE_COUNT) begin

        numerator = ({32'd0, DELAY_CHANGE}) * ({32'd0, current_time});

dynamic_delay<= BASE_AOS_DELAY_CYCLES - (numerator / TCA_PULSE_COUNT);

      end else if (current_time>= TCA_PULSE_COUNT &&current_time<
LOS_PULSE_COUNT) begin

        numerator = ({32'd0, DELAY_CHANGE}) * ({32'd0, current_time -
TCA_PULSE_COUNT});

dynamic_delay<= TCA_MIN_DELAY_CYCLES + (numerator / PULSES_LOS_PHASE);

      end else begin
dynamic_delay<= BASE_AOS_DELAY_CYCLES;
      end
    end
  end
reg prev_tx = 1'b0;
    always @(posedge clk or posedge rst) begin
      if (rst) begin
  delay_c<= 0;
  pulse_rx_out<= 0;
  prev_tx<= 0;
  current_time<= 0;
      end else begin

  prev_tx<= pulse_tx_in;
        if (pulse_tx_in&& !prev_tx) begin
  delay_c<= 0;
  pulse_rx_out<= 0;
  current_time<= current_time + 1;
        end
        if (delay_c<dynamic_delay) begin
```

```
   delay_c<= delay_c + 1;
            end
            if (delay_c == dynamic_delay) begin
   pulse_rx_out<= 1;
            end else begin
   pulse_rx_out<= 0;
            end
        end
      end
   endmodule
```

13.B.3 Velocity Module:

```
module Velocity_Calc (
   input clk,
   input rst,
   input [31:0] range_in,
   input new_range_ready,
   output reg signed [31:0] velocity_m_s_signed
);
   reg [31:0] range_prev;
   wire signed [31:0] delta_range;
   wire signed [31:0] range_in_signed = $signed(range_in);
   wire signed [31:0] range_prev_signed = $signed(range_prev);
   assign delta_range = range_in_signed - range_prev_signed;

   always @(posedge clk or posedge rst) begin
      if (rst) begin
range_prev<= 0;
velocity_m_s_signed<= 0;
      end else if (new_range_ready) begin
velocity_m_s_signed<= delta_range * 50;
range_prev<= range_in;
      end
   end

endmodule
```

13.B.4 Range and Delay Module:

```
timescale 1ns / 1ps
module Dynamic_S(
   input wire clk,
   input wire rst,
   input wire [1:0] s_tx_ctrl,
   output reg [31:0] range_m,
   output reg [31:0] time_rx,
   output reg [31:0] time_tx,
   output reg tx_flag,
   output reg rx_flag,
   output reg new_range_ready,
   output wire pulse_tx_out,
   output wire pulse_rx_out
);

   wire pulse_tx;
   wire pulse_rx;
   wire [31:0] monitor_delay;

   assign pulse_tx_out = pulse_tx;
   assign pulse_rx_out = pulse_rx;

   CASE_PRI tx_gen (
       .clk(clk),
       .rst(rst),
       .s(s_tx_ctrl),
       .p_out(pulse_tx)
   );

   CASE_R rx_gen (
       .clk(clk),
       .rst(rst),
       .pulse_tx_in(pulse_tx),
       .pulse_rx_out(pulse_rx),
       .current_delay(monitor_delay)
   );

   reg [31:0] clk_c = 0;
   reg [31:0] delay = 0;
   reg prev_tx = 0;
   reg prev_rx = 0;

   always @(posedge clk or posedge rst) begin
       if (rst)
clk_c<= 0;
       else
clk_c<= clk_c + 1;
   end

   always @(posedge clk or posedge rst) begin
       if (rst) begin
time_tx<= 0;
```

```
time_rx<= 0;
range_m<= 0;
         delay   <= 0;
tx_flag<= 0;
rx_flag<= 0;
prev_tx<= 0;
prev_rx<= 0;
                              new_range_ready<= 0;
      end else begin
prev_tx<= pulse_tx;
prev_rx<= pulse_rx;
                              new_range_ready<= 0;
         if (pulse_tx&& !prev_tx) begin
time_tx<= clk_c;
tx_flag<= 1;
rx_flag<= 0;
         end
         if (pulse_rx&& !prev_rx&& (tx_flag == 1) && (rx_flag == 0)) begin
time_rx<= clk_c;
rx_flag<= 1;
         end
         if (tx_flag&&rx_flag) begin
            delay   <= time_rx - time_tx;
range_m<= (delay * 3) >> 1;
                              new_range_ready<= 1;
tx_flag<= 0;
rx_flag<= 0;
         end
      end
   end
endmodule
```

13.B.5 Testbench:

```
`timescale 1ns / 1ps
module Unified_Range_TB;
   reg clk;
   reg rst;
   reg [1:0] s_tx_ctrl;
   wire [31:0] range_m;
   wire [31:0] time_rx;
   wire [31:0] time_tx;
   wire tx_flag;
   wire rx_flag;
   wire new_range_ready;
   wire pulse_tx_monitor;
   wire pulse_rx_monitor;
   wire signed [31:0] velocity_m_s;
Dynamic_S u_dynamic (
        .clk(clk),
        .rst(rst),
        .s_tx_ctrl(s_tx_ctrl),
        .range_m(range_m),
        .time_rx(time_rx),
        .time_tx(time_tx),
        .tx_flag(tx_flag),
        .rx_flag(rx_flag),
        .new_range_ready(new_range_ready),
        .pulse_tx_out(pulse_tx_monitor),
        .pulse_rx_out(pulse_rx_monitor)
   );
Velocity_Calc u_velocity (
        .clk(clk),
        .rst(rst),
        .range_in(range_m),
        .new_range_ready(new_range_ready),
        .velocity_m_s_signed(velocity_m_s));

   initial begin
clk = 0;
   end
   always #5 clk = ~clk;

   initial begin

        $monitor("Time %0t | TX: %b | RX: %b | Range (m): %0d | Velocity (m/s): %0d |
TxTime: %0d | RxTime: %0d | Delay: %0d",
                 $time, pulse_tx_monitor, pulse_rx_monitor, range_m, velocity_m_s, time_tx,
time_rx, (time_rx - time_tx));
   end
   initial begin
rst = 1;
s_tx_ctrl = 2'b00;
        #100;
rst = 0;
        #2500000;
        $display("Simulation Finished");
        $finish;
   end
endmodule
```

भारत सरकार
अंतरिक्ष विभाग
सतीश धवन अंतरिक्ष केंद्र शार
श्रीहरिकोटा रेंज डा.घ. 524 124
श्री पोट्टि श्रीरामुलु नेल्लूर जिला, आं.प्र., भारत
दूरभाष : +91-8623 245060 (6 जं)
फैक्स : +91-8623 222099

सत्यमेव जयते

Goverment of India
Department of Space
Satish Dhawan Space Centre SHAR
Shriharikota Range P.O. 524 124
SPSR Nellore Dist., AP., India
Telephone : +91-8623 245060 (6 Lines)
Fax : +91-8623 222099

प्रबंधनप्रणालीक्षेत्रMANAGEMENT SYSTEMS AREA
मानवसंसाधनविकासप्रभाग HUMAN RESOURCE DEVELOPMENT DIVISION
(Phone No. 08623 – 225047, e-mail: hrdd@shar.gov.in)

No.HRDD/STU/P/**PRJ2025567/**378-2025 October 31, 2025

TO WHOMSOEVER IT MAY CONCERN

This is to certify that **Mr. Vidyadheesha M Pandurangi** (Reg. No.: **6176AC22UEC162**) pursuing **B.E/B.Tech - IV Year (Electronics & Communication Engineering)** from **Adhiyamaan College of Engineering, Hosur** has carried out a Project on **"Design and implementation of targets simulator for a tracking radar using ZYNQ-based SoC"** at **Range Operations (RO)** in **SDSC SHAR, Sriharikota** from **17/09/2025** to **31/10/2025.**

During the above period, his character and conduct were found to be **Very Good.**

L. Manikandan 31/10/2025
(Dr. Manikandan L)
Head, HRM

डॉ.मणिकंडन एल Dr. MANIKANDAN L
प्रधान Head
मानव संसाधन प्रबंधन Human Resource Management
प्रधान एमएसए, एसडीएससी शार
MSA, SDSC SHAR

भारत सरकार, अंतरिक्ष विभाग
Govt. of India, Dept. of Space
इसरो, स.ध.अं.कें. शार, श्रीहरिकोटा-524 124.
ISRO, SDSC SHAR, Sriharikota-524 124.

भारतीय अंतरिक्ष अनुसंधान संगठन
इसरो isro
Indian Space Research Organisation