ECE 375: Introduction to Computer Architecture

# FPGA Spatial Mapping and Temperature-Control Project
# A Mixed-Signal Physics Control Laboratory on the Nexys A7

**By**

**David Richardson**

Instructor: Furat Al-Obaidy

November 28, 2025

Fall 2025

Demonstrating Program Operation

**Abstract**

This project implements a small but fully functional “physics control laboratory” on a Digilent Nexys A7 FPGA board. The system performs distance measurements using a Pmod ToF time-of-flight sensor, monitors on-board temperature via the Xilinx XADC, and drives a fan under the combined influence of a PIR motion sensor and temperature thresholds. A seven-segment display provides a human-readable time/status interface, while a UART link streams structured packets to a host PC for logging and visualization.

Beyond the concrete functionality, the primary goal of the project is educational: to gain first-hand experience designing a non-trivial FPGA system from scratch, integrating digital logic, communication protocols, mixed-signal measurement, and basic control behavior. The work demonstrates how project-based learning can connect theory from digital design, embedded systems, and physics to observable behavior on the bench.

## Contents

**1 Introduction** **4**

**2 Design Objectives and Requirements** **4**
2.1 Functional Objectives . . . . . 4
2.2 Architectural Objectives . . . . . 5

**3 System-Level Architecture** **5**
3.1 Top-Level Block Diagram . . . . . 5
3.2 Major RTL Modules . . . . . 6

**4 Clocking and Reset Strategy** **7**
4.1 System Clock and Pixel Clock . . . . . 7
4.2 Reset Synchronization . . . . . 7
4.3 Clock-Domain Crossings . . . . . 8

**5 Fixed-Point Representation and Scaling** **8**
5.1 Q1.15 Format . . . . . 8
5.2 Temperature . . . . . 8
5.3 Angle . . . . . 9
5.4 Duty Cycle . . . . . 9

**6 Sensing Subsystems** **9**
6.1 XADC Temperature Front-End . . . . . 9
6.2 PIR Motion Conditioning . . . . . 9
6.3 Rotary Encoder and Survey Control . . . . . 10
6.4 Time-of-Flight Sensor Interface . . . . . 10

**7 Control Subsystems** **11**
7.1 Temperature/PIR/Manual Fan Controller . . . . . 11
7.2 PWM DAC . . . . . 11
7.3 Surveyor FSM and Angle Indexer . . . . . 11

**8 VGA Rendering Pipeline** **12**
8.1 Pipeline Overview . . . . . 12
8.2 VGA Timing Core . . . . . 12
8.3 ToF Bit-Plane Framebuffer and Plot . . . . . 13
8.4 Dual-Buffered Logo Engine . . . . . 13
8.5 HUD Overlay . . . . . 13
8.6 Rotary Bar Overlay . . . . . 14

**9 Telemetry and Logo Streaming** **14**
9.1 Mapper Packetizer and UART TX . . . . . . 14
9.2 UART RX and Logo Loader . . . . . . 15
9.3 Host MATLAB/Python Workflow . . . . . . 15

**10 Memory Architecture** **15**
10.1 ToF Framebuffer . . . . . . 15
10.2 Dual-Buffered Logo Memory . . . . . . 16
10.3 Other Storage Elements . . . . . . 16

**11 Ready/Valid Pipelines and Dataflow** **16**
11.1 Packetizer → UART TX . . . . . . 17
11.2 UART RX → Logo Loader . . . . . . 17
11.3 ToF Sensor → Packetizer . . . . . . 17

**12 Verification and Bring-Up** **17**
12.1 Unit Testing of Subsystems . . . . . . 17
12.2 Integrated Spatial Mapping Experiments . . . . . . 18
12.3 Limitations and Observed Behavior . . . . . . 18

**13 Conclusions and Future Work** **18**

## 1 Introduction

This report presents a complete and unified description of the **FPGA_Signal_Control_System**, an integrated sensing, control, telemetry, and visualization platform built on a Digilent Nexys A7-100T (XC7A100T-1CSG324C) FPGA board. The design has evolved from a simple temperature/-fan controller into a mixed-signal *physics control laboratory* that combines:

- Time-of-Flight (ToF) distance sensing using a Pmod ISL29501.
- Temperature sensing via the on-chip XADC and an optional digital temperature sensor.
- Passive Infrared (PIR) motion detection with temporal decay logic.
- Rotary encoder input used for manual surveying and HUD interaction.
- Fan actuation through a PWM driver with temperature, PIR, and manual contributions.
- A full VGA 640×480 pipeline that renders a ToF range plot, heads-up display (HUD), and a dual-buffered image/logo viewport.
- Structured UART telemetry at 2 Mbit/s to a host PC for logging and MATLAB-based visualization.

All digital logic runs in a single 100 MHz fabric clock domain. Slow behaviors (seconds, milliseconds, microseconds) are implemented using tick-enable pulses derived from the master clock. A small number of clock-domain crossings are handled explicitly: the VGA pixel domain is derived from the 100 MHz clock via a Xilinx clocking wizard, and dual-clock RAMs plus synchronizers bridge between 25 MHz pixel logic and the 100 MHz system domain.

This document supersedes earlier reports and cheat sheets: it provides a single, self-consistent source of truth for the current design, including module-level theory of operation, dataflow, memory architecture, and verification strategy.

## 2 Design Objectives and Requirements

The project goals can be broadly divided into *functional objectives* and *architectural objectives.*

### 2.1 Functional Objectives

F1. **Real-time spatial mapping.** Use a Pmod ToF sensor and a survey mechanism (rotary encoder/servo) to build a 2D map of distance vs. angle, visualized on VGA and streamed to a PC.

F2. **Temperature-based fan control.** Measure temperature via the XADC (and optionally TMP-style digital sensor), and control a 5 V fan with hysteresis and user-adjustable behavior.

F3. **Occupancy-aware actuation.** Incorporate PIR motion sensing to bias fan behavior toward occupied states, and visualize motion with a temporal decay indicator.

F4. **High-speed telemetry.** Stream timestamped ToF distance, angle, temperature, and fan duty via UART at 2 Mbit/s using a framed binary protocol.

F5. **Rich on-board visualization.** Provide a 640×480 VGA output with:

- A ToF range plot in the left half of the screen.
- A right-side HUD panel showing temperature, fan state, motion, UART activity, and rotary position.
- A dual-buffered 320×240 logo/image viewport, with frames loaded via UART or ROM .hex initialization.

### 2.2 Architectural Objectives

A1. **Single system clock domain.** All "logic-domain" modules (sensor controllers, control FSMs, packetizers) run at 100 MHz. Slow effects are derived via tick-enable pulses; no arbitrary divided clocks are created.

A2. **Clean clock-domain crossings.** The only additional domain is the VGA pixel clock (e.g., 25 MHz), created by a clocking wizard and isolated using dual-clock RAMs and explicit synchronizers.

A3. **Ready/valid streaming interfaces.** Streaming pipelines (packetizer $\rightarrow$ UART TX, UART RX $\rightarrow$ frame loader, etc.) use ready/valid handshakes to avoid overruns and to make backpressure explicit.

A4. **Fixed-point numerics.** Use Q-format fixed-point (Q1.15) consistently for temperature, angles, and duty cycles, so that all math is integer-based and easily interpretable on the host.

A5. **Reusability and modularity.** Decompose the system into modules that can be reused in other projects: generic UART cores, an image dual-buffer, an I$^2$C master, etc.

## 3 System-Level Architecture

### 3.1 Top-Level Block Diagram

Figure 1 shows the high-level architecture of `spatial_mapping_temperature_control_top`. Sensors and actuators interface on the left and right; the FPGA core sits in the center; the host PC for telemetry and image streaming is at the bottom.

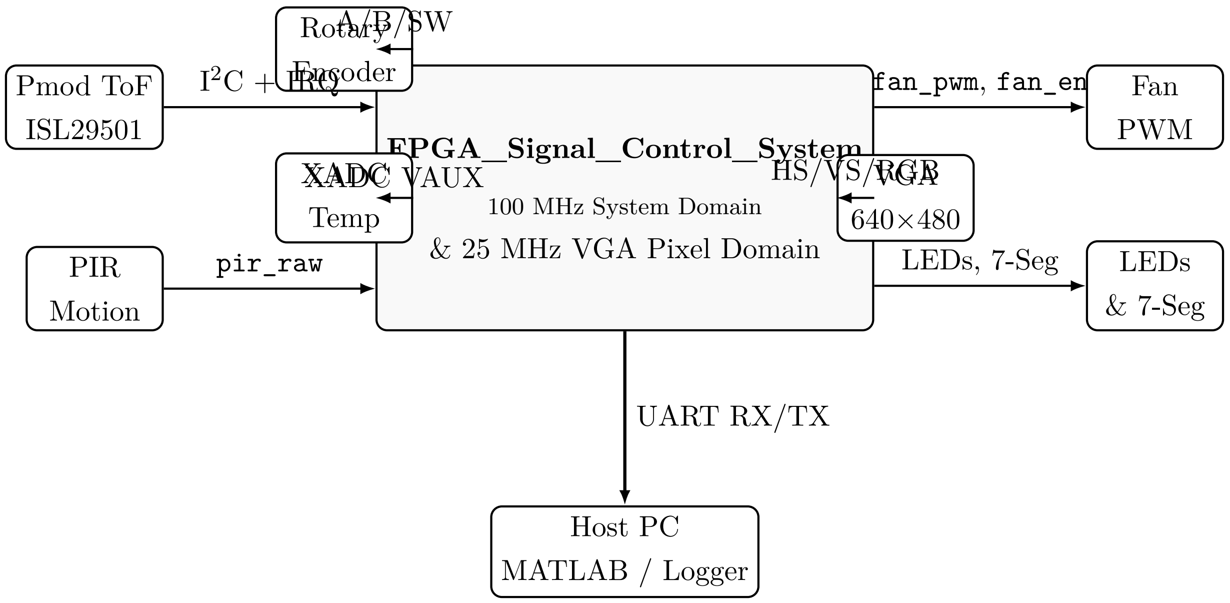

Figure 1: Top-level architecture: sensors and actuators interfacing with the FPGA core, plus host PC for telemetry and image streaming.

### 3.2 Major RTL Modules

The top-level module instantiates the following key blocks (non-exhaustive):

- `sevenseg_clock_buttons`: HH:MM:SS clock and 7-seg display.
- `xadc_sampler`: XADC configuration and Q1.15 temperature export.
- `temp_fan_ctrl`: Fan control FSM with temperature, PIR, and manual inputs.
- `pwm_dac`: General-purpose Q1.15 $\rightarrow$ PWM converter.
- `pir_conditioner`: Synchronization and hold logic for PIR.
- `rotary_encoder_quadrature`: Decode A/B/SW into position and events.
- `tof_sensor`: ISL29501 driver over `i2c_master`.
- `i2c_master`: Byte-granular I$^2$C engine (start/stop/ack).
- `surveyor_fsm`: Survey cadence and sweep-wrap generation.
- `angle_indexer`: Q1.15 angle and multi-turn index from step pulses.
- `mapper_packetizer`: Frame ToF + angle + status into UART packets.
- `uart_stream_tx`: 8-N-1 UART transmitter at 2 Mbit/s.

- `uart_rx`: UART receiver used for image frame loading.
- `logo_uart_frame_loader`: Decode image frames from RX stream.
- `image_dualbuf_320x240_rgb12`: Dual-buffered logo memory.
- `vga_range_plot_top`: VGA timing, ToF bit-plane plot, logo viewport mux.
- `vga_status_overlay`: HUD widgets on the right panel.
- `rotary_bar_overlay`: Final overlay of rotary activity bar.

These modules interact through well-defined buses, discussed in the following sections.

## 4 Clocking and Reset Strategy

### 4.1 System Clock and Pixel Clock

The Nexys A7 provides a 100 MHz oscillator, exposed as `clk_100MHz`. All system-domain logic (sensor interfaces, control FSMs, packetizers, UART cores) uses this as its only clock. Slow time scales are derived via tick-enable generators of the form

$$\texttt{tick\_en} = \begin{cases} 1 & \text{if counter reaches terminal value} \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

The VGA timing engine requires a pixel clock around 25 MHz for 640×480@60 Hz. This is obtained using a Xilinx clocking wizard instantiated inside `vga_range_plot_top`. The wizard input is `clk_100MHz`, and its output `pix_clk` becomes the sole clock for the VGA domain (`hcount`, `vcount`, `active_video`).

### 4.2 Reset Synchronization

The center pushbutton `btnc` is an asynchronous input. A two-stage synchronizer and a small filter convert it into a clean, synchronous reset `rst`:

- First, two flip-flops clocked at 100 MHz remove metastability and generate `btnc_sync`.
- Second, a counter stretches `btnc_sync` into a multi-cycle `rst` active-high pulse.

In the VGA domain, a synchronized reset `rst_pix` is derived from `rst` with another small synchronizer to avoid asynchronous release into the pixel clock domain.

### 4.3 Clock-Domain Crossings

There are three primary crossing patterns:

(a) **System → Pixel (status snapshot).** Telemetry signals such as temperature Q1.15, fan duty, UART counters, PIR flags, and rotary position originate in the 100 MHz domain but are used by `vga_status_overlay` in the 25 MHz pixel domain. These signals are sampled and latched at a safe rate (e.g. once per frame at `frame_tick`), using small multi-flop synchronizers and frame-based sampling to avoid tearing.

(b) **Pixel → System (logo active buffer monitoring).** The pixel domain exposes `display_buf_pix` (which logo bank is currently visible) back to the system domain via a 2-flop synchronizer, producing `active_buf_sys`.

(c) **Dual-clock RAM for logos.** `image_dualbuf_320x240_rgb12` is implemented as true dual-port RAM: one port is clocked by `clk_100MHz` and handles writes; the other is clocked by `pix_clk` and serves read requests.

This disciplined separation keeps each domain conceptually simple and bounds CDC complexity to a few well-understood interfaces.

## 5 Fixed-Point Representation and Scaling

### 5.1 Q1.15 Format

The project uses signed Q1.15fixed-point format where convenient. In this representation:

- A value `x_q15` is a 16-bit signed two's complement integer.
- Its real value is $x_{\text{real}} = \frac{\texttt{x\_q15}}{2^{15}}$.
- The representable range is approximately $[-1.0, 1.0)$ with resolution $2^{-15} \approx 3.05 \times 10^{-5}$.

Any addition or subtraction in Q1.15 is just integer addition/subtraction; no special hardware is needed. Multiplication of two Q1.15 values produces a Q2.30 value, which is then re-normalized by discarding the lower 15 bits or using rounding.

### 5.2 Temperature

`xadc_sampler` outputs `sample_q15` as a normalized temperature code. Internally it maps XADC raw codes into a signed Q1.15number:

$$\texttt{sample\_q15} \approx \alpha \cdot (C_{\text{raw}} - C_0),$$

where $C_{\text{raw}}$ is the XADC code and $C_0$ is a reference code at a known temperature. On the host, a linear calibration maps `sample_q15` to degrees Celsius or Fahrenheit.

### 5.3 Angle

`angle_indexer` represents the sweep angle as `theta_q15`. A full revolution $[0, 2\pi)$ is mapped to Q1.15 $[0, 1.0)$:

$$\theta_{\mathrm{rad}} = 2\pi \cdot \frac{\texttt{theta\_q15}}{2^{15}}, \qquad \theta_{\mathrm{deg}} = 360^\circ \cdot \frac{\texttt{theta\_q15}}{2^{15}}.$$

If `N_STEPS` survey rays are used per sweep, then `theta_q15` increments by:

$$\texttt{Q15\_PER\_STEP} = \left\lfloor \frac{2^{15}}{\texttt{N\_STEPS}} \right\rfloor$$

for each `step_pulse`. This establishes a simple mapping between step index and angle.

### 5.4 Duty Cycle

The fan controller outputs `duty_q15`, a Q1.15 duty cycle command between 0 and 1 (internally it may saturate above 100% only for debug). The `pwm_dac` module converts this into an $N$-bit carrier compare value:

$$\texttt{compare} = \left\lfloor \frac{\texttt{duty\_q15}}{2^{15}} \cdot 2^N \right\rfloor.$$

The PWM output is high while `ctr` $<$ `compare`, where `ctr` is an $N$-bit free-running counter at 100 MHz.

## 6 Sensing Subsystems

### 6.1 XADC Temperature Front-End

The `xadc_sampler` module configures the Xilinx XADC to operate in single-channel mode on a chosen VAUX input (e.g. VAUX1 for on-board temperature or an external sensor). Its responsibilities:

- Generate periodic conversion start pulses at a programmable rate (tens of Hz).
- Read XADC conversion results and apply offset/gain calibration.
- Export the result as `sample_q15` plus a one-cycle strobe `samp_valid`.

The canonical temperature bus `samp_q15`, `samp_valid` is then consumed by the fan controller, the HUD overlay, and the UART packetizer.

### 6.2 PIR Motion Conditioning

The PIR sensor provides a single digital output `pir_raw`, which is asynchronous and typically noisy at startup. The `pir_conditioner` module:

- Synchronizes `pir_raw` to `clk_100MHz`.
- Masks out the first few seconds after power-up (warm-up).
- Detects rising edges and generates `pir_rise`.
- Holds `pir_active` high for a configured time after each detection.

`pir_active` is used by `temp_fan_ctrl` as a proxy for occupancy, and is visualized in the HUD as a motion streak and indicator tile.

## 6.3 Rotary Encoder and Survey Control

A detented quadrature rotary encoder is connected to the JC header. The `rotary_encoder_quadrature` module:

- Synchronizes channels A and B.
- Implements a decode FSM that converts gray-coded transitions into:
  - `rot_step_pulse`: 1-clock pulse per detent.
  - `rot_dir`: direction bit (1 = CW).
  - `rot_pos`: signed 16-bit position counter.
- Debounces and detects the push switch (SW) to generate a clean `rot_btn_pulse`.

In manual surveying mode, the encoder acts as a *spinner*: each step advances the survey angle index, and the HUD displays a compass/spinner bar reflecting `rot_pos` and recent activity.

## 6.4 Time-of-Flight Sensor Interface

The Pmod ToF (ISL29501) is managed by the `tof_sensor` module, which sits atop a generic `i2c_master`. The high-level behavior:

1. On reset, execute an initialization table of registers over $I^2C$ to configure modulation frequency, timing, and mode.
2. For each measurement:
   (a) Assert `ss` (sample-start) pulse.
   (b) Wait for `irq_n` to assert or timeout.
   (c) Initiate an $I^2C$ read of the distance and status registers.
3. Present `dist_mm`, `status`, and `dist_vld` to the rest of the system.

The `i2c_master` interface uses a ready/valid byte protocol: the `tof_sensor` FSM pushes command bytes, reads response bytes, and generates START/STOP conditions by toggling SCL and SDA open-drain enables.

## 7 Control Subsystems

### 7.1 Temperature/PIR/Manual Fan Controller

`temp_fan_ctrl` implements a multi-input fan policy. Key signals:

- Inputs: `samp_q15`, `samp_valid`, `pir_active`, `btn_manual_pulse`, mode switches.
- Outputs: `fan_pwm`, `fan_en_ja4`, `fan_temp_on`, `fan_pir_on`, `fan_manual_on`, `duty_q15`.

The FSM maintains internal state for hysteresis and mode selection. A typical policy:

- If temperature-only mode is enabled, turn fan on when `samp_q15` exceeds a "hot" threshold and off when it falls below a "cool" threshold.
- If PIR mode is enabled, ensure the fan remains at least at LOW duty whenever `pir_active` = 1.
- If manual mode is enabled, the user can toggle between OFF and a manual duty level via `btn_manual_pulse`.

The final `duty_q15` is visualized in the HUD and streamed via UART.

### 7.2 PWM DAC

The `pwm_dac` module is a generic Q1.15-to-PWM converter. Given:

- A Q1.15 value `in_q15`.
- A carrier counter of width `CTR_W`.

It computes a compare value and emits a PWM waveform at frequency

$$f_{\mathrm{PWM}} = \frac{f_{\mathrm{clk}}}{2^{\texttt{CTR\_W}}}.$$

At 100 MHz and 12-bit carrier, this is about 24.4 kHz.

### 7.3 Surveyor FSM and Angle Indexer

The `surveyor_fsm` coordinates the acquisition of spatial data:

- Generates `step_pulse`, `step_dir` at a controlled cadence.
- Issues measurement triggers to `tof_sensor`.
- Asserts `sweep_wrap` at the completion of a sweep.

`angle_indexer` consumes `step_pulse` and `step_dir` and updates:

- `theta_q15`: wrapped Q1.15 angle in [0, 1).
- `theta_turn_q15`: multi-turn Q1.15 representation.
- Optional integer index for debug.

Together, these form the angular backbone of the spatial mapping pipeline.

## 8 VGA Rendering Pipeline

### 8.1 Pipeline Overview

The VGA pipeline has three major layers:

(a) **Base layer:** ToF range plot + logo viewport, generated in `vga_range_plot_top`.

(b) **HUD overlay:** Telemetry widgets added by `vga_status_overlay`.

(c) **Rotary bar overlay:** Final overlay by `rotary_bar_overlay`.

Figure 2 sketches the dataflow.

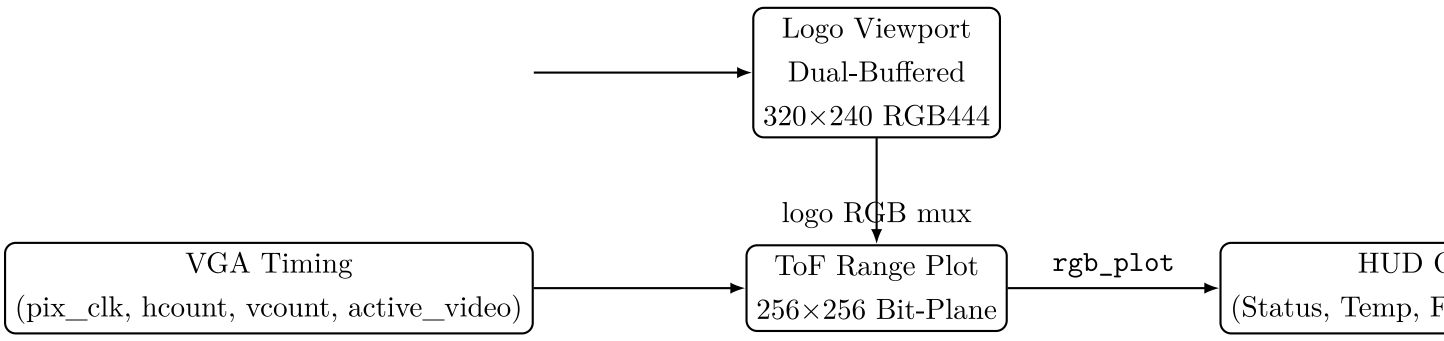

Figure 2: Conceptual VGA rendering pipeline: timing core, range plot, logo viewport mux, HUD overlay, and rotary bar overlay.

### 8.2 VGA Timing Core

Inside `vga_range_plot_top`, the timing core:

- Generates `hcount` and `vcount` counters that implement the 640×480@60 Hz timing.
- Asserts `active_video` only within the visible 640×480 region.
- Generates `vga_hsync_n` and `vga_vsync_n` with the appropriate pulse widths and polarities.
- Emits `frame_tick` as a single-cycle strobe at the top-left of the active region.

## 8.3 ToF Bit-Plane Framebuffer and Plot

A dual-port block RAM implements a 256×256 bit-plane framebuffer for ToF data. Port A (system domain) writes single bits for hits; Port B (pixel domain) reads during active video.

- System side: `tof_plot_point_writer` converts polar (`dist_mm`, `theta_q15`) into Cartesian pixel coordinates and sets bits accordingly.
- Pixel side: a colorizer maps bit-plane values and range rings into RGB:
  - Hit bit = 1 ⇒ bright green.
  - Hit bit = 0 ⇒ gray background.
  - Ring radii ⇒ cyan lines overlayed.
  - Axes ⇒ white horizontal/vertical lines.

## 8.4 Dual-Buffered Logo Engine

The logo engine consists of:

- `uart_rx`: receives image bytes at 2 Mbit/s.
- `logo_uart_frame_loader`: parses a simple frame protocol:
  - Header (sync + dimensions).
  - 320×240 pixels, each in RGB444 format.

  The loader writes incoming pixels into the *inactive* logo bank at `clk_100MHz`.
- `image_dualbuf_320x240_rgb12`: holds two independent 320×240 memories. Port A (system) performs writes; Port B (pixel) performs reads using scaled-down viewport coordinates `logo_x`, `logo_y`.
- A swap mechanism: `logo_swap_req_sys` requests a bank swap. The dual-buffer block performs the swap at the next `frame_tick_pix` to avoid tearing, updating `display_buf_pix` (pixel domain) and `active_buf_sys` (system domain).

A slide switch can optionally force selection of bank 0 or bank 1 for demonstration.

## 8.5 HUD Overlay

`vga_status_overlay` takes the background RGB stream from `vga_range_plot_top`, plus a set of synchronized telemetry signals (latched temperature, duty, PIR, etc.), and draws:

- A vertical temperature bar with color-coded ranges (cool, neutral, hot).
- Numeric temperature readout (simple 7-seg-like glyphs or coarse numeric tiles).

- Fan tiles indicating which contribution (TEMP, MANUAL, PIR) is active.
- A duty bar for `duty_q15`.
- UART traffic counters (bytes sent/received).
- Angle and heading indicators derived from `theta_q15`.
- A simple spinner based on rotary encoder activity.

All HUD drawing is implemented as a pure combinational function of (`hcount`, `vcount`, `active_video`) and the latest snapshotted telemetry values.

### 8.6 Rotary Bar Overlay

`rotary_bar_overlay` is the final stage. It receives `rgb_hud` and draws a vertical bar near the right edge:

- Bar height is proportional to |`enc_pos`|.
- Color encodes direction: green-ish for CW, red-ish for CCW.
- A decaying activity register gives a fade-out effect after motion.

This overlay provides a visually striking representation of manual interaction with the system.

## 9 Telemetry and Logo Streaming

### 9.1 Mapper Packetizer and UART TX

`mapper_packetizer` constructs binary frames with the structure:

| Bytes | Field |
|---|---|
| 0–1 | Sync: 0x55, 0xAA |
| 2–5 | `dbg_t_us` (32-bit timestamp) |
| 6–7 | `theta_q15` (Q1.15 angle) |
| 8–9 | `dist_mm` (16-bit distance) |
| 10–11 | `temp_q15` (canonical temperature, Q1.15) |
| 12–13 | `duty_q15` (fan duty, Q1.15) |
| 14 | `status` (ToF and system bits) |
| 15–16 | CRC16 (little-endian) |

For each valid ToF sample (`meas_vld`), a new frame is constructed and output byte-by-byte as (`pkt_byte`, `pkt_vld`). The module honors `pkt_rdy` from `uart_stream_tx` so that if the UART is busy, new bytes are withheld.

`uart_stream_tx` implements an 8-N-1 UART transmitter with a baud rate derived from 100 MHz. For 2 Mbit/s, the divisor is approximately 50 clock cycles per bit.

### 9.2 UART RX and Logo Loader

`uart_rx` listens on `uart_rxi` and reconstructs 8-bit bytes, signaling them on `rx_byte` with `rx_vld`. The `logo_uart_frame_loader` module consumes these bytes and implements a simple state machine:

1. Wait for header (magic sync bytes + dimensions).
2. For each pixel index $i \in [0, 320 \times 240 - 1]$, read two bytes and reconstruct RGB444 pixel:

$$\text{pixel}[11:0] = \{R[3:0], G[3:0], B[3:0]\}.$$

3. Emit `logo_wr_en_sys`, `logo_wr_addr_sys`, `logo_wr_data_sys` to write into the inactive bank.
4. Assert `logo_swap_req_sys` to request a bank swap at the next frame.

This path is fully decoupled from the telemetry TX chain.

### 9.3 Host MATLAB/Python Workflow

A typical host workflow:

1. Open the UART port at 2 Mbit/s (e.g. COMx).
2. In one thread, read ToF telemetry frames:
    - Search for 0x55, 0xAA.
    - Decode the following 15 bytes into the fields above.
    - Validate CRC16.
    - Log to CSV and plot distance vs. angle in real time.
3. In another thread or script, read an image, quantize to 320×240 RGB444, pack into bytes, and send to the logo loader protocol.

This decoupled architecture allows the same PHY (the FTDI UART link) to support both telemetry and image streaming as needed.

## 10 Memory Architecture

### 10.1 ToF Framebuffer

The ToF framebuffer is a dual-port block RAM:

- Resolution: 256×256 bits.

- Addressing: `[15:0]` (e.g. `y[7:0]` concatenated with `x[7:0]`).
- Port A: 100 MHz write port (plot writer + clear FSM).
- Port B: 25 MHz read port (colorizer).

A dedicated clear FSM triggers at `sweep_wrap` and steps through all addresses to set bits to zero, ensuring the next sweep has a fresh canvas.

### 10.2 Dual-Buffered Logo Memory

`image_dualbuf_320x240_rgb12` internally implements:

- Two independent 320×240×12-bit memories.
- System-domain write interface: `wr_en_sys`, `wr_addr_sys`, `wr_data_sys` always target `write_buf_sys`.
- Pixel-domain read interface: addresses formed from `logo_x`, `logo_y` and served from `display_buf_pix`.

Bank swap logic:

- `logo_swap_req_sys` asserts when a full frame has been loaded into the inactive bank.
- The dual-buffer module waits for `frame_tick_pix` and then toggles the bank select stored in a pixel-domain register.
- `display_buf_pix` is exported to the system domain via a 2-flop synchronizer as `active_buf_sys`.

### 10.3 Other Storage Elements

Additional small memories include:

- FIFO/buffer inside `mapper_packetizer` (if present).
- Optional circular buffers for averaging ToF samples.
- Frame counters and bargraph counters for UART activity.

All are implemented as simple BRAMs or LUTRAMs, depending on size.

## 11 Ready/Valid Pipelines and Dataflow

Streaming interfaces follow a simple invariant:

A transfer occurs in a cycle where `vld` = 1 and `rdy` = 1.

## 11.1 Packetizer → UART TX

- `pkt_byte`: data byte.
- `pkt_vld`: asserted when a packet byte is available.
- `pkt_rdy`: asserted by `uart_stream_tx` when idle.

The packetizer only increments its internal byte index when both signals are true. If `pkt_vld` = 1 but `pkt_rdy` = 0, the current byte is held and re-presented in the next cycle.

## 11.2 UART RX → Logo Loader

- `rx_byte`: valid byte.
- `rx_vld`: pulse when a new byte is available.

The logo loader consumes bytes opportunistically; the baud rate is low enough relative to 100 MHz that backpressure is not required.

## 11.3 ToF Sensor → Packetizer

- `meas_vld`: 1-cycle strobe from `tof_sensor` when a new measurement is ready.
- `meas_dist_mm`, `meas_status`: associated fields.

`mapper_packetizer` snapshots the measurement and starts a frame whenever `meas_vld` = 1. No additional backpressure is required because the ToF sample rate is intentionally lower than the UART throughput.

# 12 Verification and Bring-Up

## 12.1 Unit Testing of Subsystems

Each subsystem was brought up independently:

- Seven-seg clock verified stand-alone (HH:MM:SS and increment buttons).
- XADC path verified by monitoring `sample_q15` vs. known temperature changes.
- PIR conditioner tested by moving in front of the sensor and observing `pir_active` and HUD indicators.
- Fan controller tested by forcing different modes and watching the PWM duty and fan behavior.
- ToF subsystem checked via $I^2C$ waveforms and distance for known targets.

- UART path checked with loopback, then with `mapper_packetizer` frames and PC decoding.
- VGA timing confirmed with a color bar pattern prior to enabling the ToF plot, HUD, and logo layers.

### 12.2 Integrated Spatial Mapping Experiments

With all subsystems active:

1. The ToF sensor is placed in a room with walls and simple targets.
2. The surveyor sweeps through a configured set of angles.
3. The VGA left panel shows a polar range plot; the right HUD shows temperature, fan state, and motion.
4. The host PC logs frames and plots distance vs. angle.

Observed plots match the expected geometry (e.g. walls at consistent range, objects producing arcs at the appropriate angles).

### 12.3 Limitations and Observed Behavior

- Angular resolution is limited by the discrete number of steps per sweep; increasing resolution increases sweep time.
- ToF measurement noise introduces jitter in radial position; simple averaging can reduce this.
- Mechanical alignment of the sensor and any servo/rotary stage affects absolute map fidelity.
- Thermal calibration is approximate; a more rigorous two-point calibration would improve absolute accuracy.

Despite these limitations, the system robustly demonstrates coherent spatial mapping and meaningful fan/occupancy behavior.

## 13 Conclusions and Future Work

The `FPGA_Signal_Control_System` realizes a substantial mixed-signal experiment on an FPGA:

- It bridges ToF sensing, temperature measurement, motion detection, rotary input, PWM fan actuation, UART telemetry, and VGA graphics in a coherent architecture.
- It demonstrates disciplined use of a single system clock, explicit tick enables, fixed-point numerics, and ready/valid streaming.

- It provides both on-board visualization (VGA HUD) and host-side analysis (MATLAB).

Natural extensions include:

- Integrating a servo for true 2D or 3D scanning.
- Adding more advanced filtering and clustering to the spatial map.
- Introducing a soft-core CPU to orchestrate experiments while retaining RTL for hard real-time paths.
- Expanding the HUD for additional overlays (e.g. gridlines, labels, numerical readouts).
- Logging long-term environmental data for statistical analysis.

Overall, the system serves as a flexible physics/control sandbox and a demonstration of modern FPGA-based system integration.

## References

- **Associated Works**

  1. FPGA_Spatial_Mapping_and_Temperature_Control_Presentation_Report.
  2. FPGA_Spatial_Mapping_and_Temperature_Control_CheatSheet.

- **Board and Instrumentation Reference Manuals**

  1. Analog Discovery 2 Reference Manual - Digilent Reference.
  2. Nexys A7 Reference Manual - Digilent Reference.
  3. Pmod ToF Reference Manual - Digilent Reference.
  4. Pmod ToF Hierarchical Block Library - Digilent Reference.
  5. Pmod Interface Specification 1.3.1.

- **FPGA and XADC Documentation**

  1. UG480: 7 Series FPGAs XADC Dual 12-Bit 1 MSPS A2D Converter User Guide.
  2. Vivado Design Suite User Guide.
  3. Vivado Design Suite Tcl Command Reference Guide.
  4. UG888.
  5. UG899.
  6. UG937.

- **Data Sheets**

  1. HC-SR501 PIR MOTION DETECTOR Data Sheet.
  2. EDL3007S05 Fan Info.
  3. atmel-8896e-seeprom-at24c04d-datasheet.
  4. REN_isl29501_DST_20220317.