FPGA-Accelerated Quadcopter Flight Controller with Sensor Fusion on Xilinx Alveo U280

Prayag Sridhar
Northeastern University
Email: sridhar.pray@northeastern.edu

***Abstract*—This project implements a real-time quadcopter flight controller on the Xilinx Alveo U280 FPGA platform using Vitis HLS 2023.2. The controller integrates sensor fusion via complementary filtering, cascaded PID controllers for attitude and altitude regulation, motor mixing for X-configuration quadcopters, and safety monitoring with emergency stop capabilities. The design achieves timing closure at 175 MHz with WNS of +0.016 ns, control loop rate of 549 kHz (549× the industry-standard 1 kHz), and resource utilization of 8.26% LUT and 5.22% FF. Comparative analysis against CPU (Intel Xeon Gold 6132) and GPU (NVIDIA Tesla V100) demonstrates that the FPGA's deterministic execution with zero timing jitter makes it optimal for safety-critical flight control applications.**

I. Project Description

This work focuses on an FPGA/HLS implementation of a quadcopter flight controller using the Xilinx Alveo U280. The design performs real-time sensor fusion, cascaded PID control, and motor mixing using a fully pipelined dataflow architecture.

A. Key Components

- **Kernel:** `quadcopter_top.cpp` and supporting modules define the HLS dataflow kernel with parallel processing elements for sensor fusion, attitude control, altitude control, and motor mixing.
- **Host Runtime:** `host.cpp` uses XRT to load sensor data, launch the flight controller kernel, time execution, and validate FPGA results against CPU baseline.
- **Control Modules:** Eight functional modules including complementary filter, PID controller, attitude controller, altitude controller, motor mixer, and safety monitor.
- **Build/Configuration:** Vitis HLS synthesis at 175 MHz clock targeting the U280 platform with AXI memory interfaces.

II. Code Description

- **quadcopter_control.h** – Defines compile-time constants, fixed-point types (`fp32_t = ap_fixed<32,16>`, `sensor_data_t = ap_fixed<16,8>`), and data structures (`IMUData`, `StateVector`, `MotorSpeeds`).
- **complementary_filter.cpp** – Implements sensor fusion with $\alpha = 0.98$, fuses accelerometer and gyroscope data, computes roll/pitch angles with drift correction.
- **pid_controller.cpp** – Reusable PID with configurable K_p, K_i, K_d gains and anti-windup with integral clamping.
- **motor_mixer.cpp** – X-configuration mixing equations for four motors with output clamping to valid PWM range (0-100%).
- **safety_monitor.cpp** – Angle limit monitoring (± 45 threshold) and emergency stop with immediate motor shutdown.

III. Theoretical Background

A. Quadcopter Dynamics

A quadcopter achieves controlled flight through the differential rotation of four propellers in X-configuration. Fig. 1 shows the motor arrangement. The mixing equations are:

$$\begin{aligned} M_1 &= T + R - P + Y \quad \text{(Front-left, CCW)} \\ M_2 &= T - R - P - Y \quad \text{(Front-right, CW)} \\ M_3 &= T - R + P + Y \quad \text{(Rear-right, CCW)} \\ M_4 &= T + R + P - Y \quad \text{(Rear-left, CW)} \end{aligned}$$

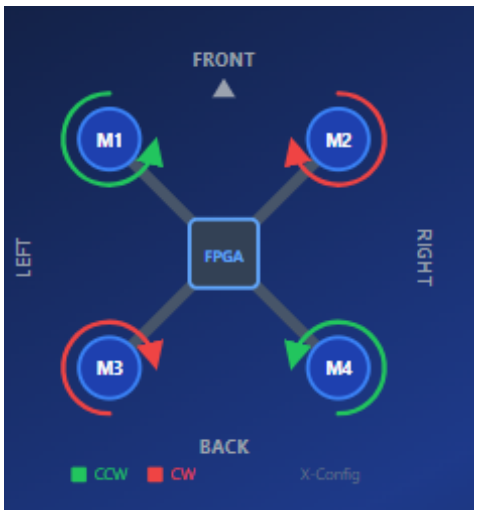

Fig. 1. X-configuration motor arrangement with CCW/CW rotation directions.

B. Control Axes

Fig. 2 illustrates the four primary control inputs: throttle, pitch, roll, and yaw, showing how motor speeds vary for each movement.

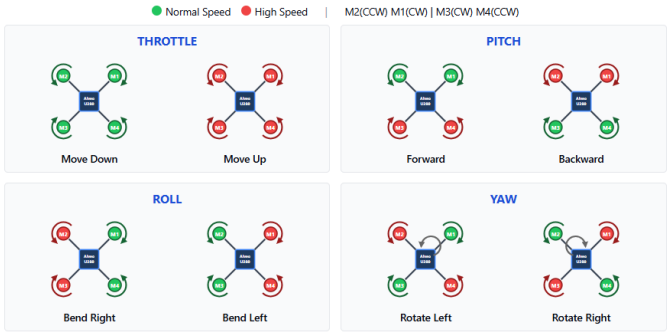

Fig. 2. Quadcopter control axes showing motor speed variations for each movement type.

C. Sensor Fusion

The complementary filter fuses accelerometer and gyroscope data:

$$\theta = \alpha \cdot (\theta + \omega \cdot dt) + (1 - \alpha) \cdot \theta_{accel} \tag{1}$$

where $\alpha = 0.98$ balances gyroscope responsiveness with accelerometer stability. Fig. 3 shows the filter architecture.

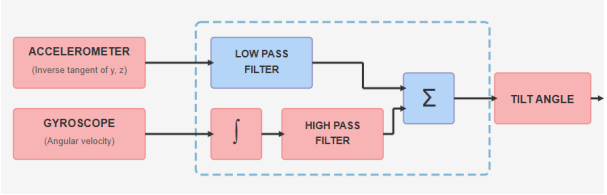

Fig. 3. Complementary filter combining accelerometer (low-pass) and gyroscope (high-pass) data.

D. PID Control

The PID controller computes corrections based on error:

$$u = K_p \cdot e + K_i \cdot \int e\,dt + K_d \cdot \frac{de}{dt} \tag{2}$$

Fig. 4 illustrates the PID control loop structure.

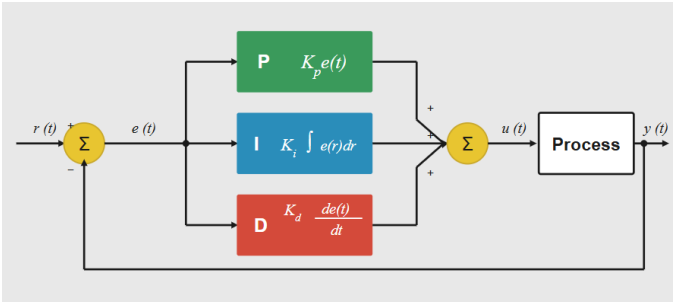

Fig. 4. PID controller block diagram showing proportional, integral, and derivative paths.

IV. System Architecture

A. Alveo U280 Platform Structure

Fig. 5 shows the complete system architecture on the Alveo U280, comprising the Static Shell (DMA, PCIe), User Logic Partition (flight controller kernel), and Memory subsystem.

B. Vivado Block Diagram

Fig. 6 presents the Vivado-generated block diagram showing hardware interconnections between the kernel and memory interfaces.

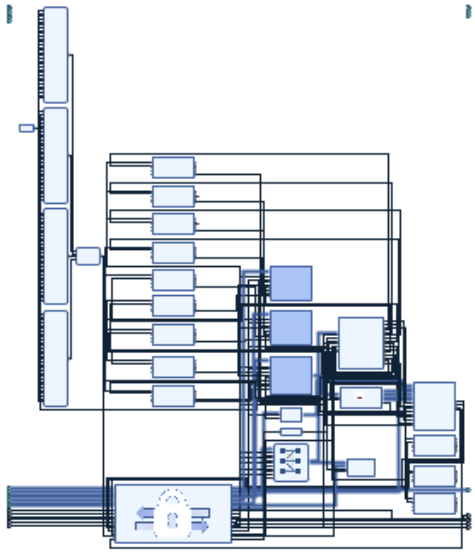

Fig. 6. Vivado block diagram showing kernel interfaces and memory connections.

C. Data Flow Pipeline

Fig. 7 illustrates the pipelined data flow through the flight controller modules. Table I shows the achieved frequencies.

TABLE I
Per-Module Timing Results

Module	Target (MHz)	Achieved (MHz)
process_keyboard_input	175	191
complementary_filter	175	214
pid_controller	175	264
attitude_controller	175	253
altitude_controller	175	252
motor_mixer	175	243
safety_monitor	175	196
flight_controller	175	196

V. Optimization Strategies

The flight controller employs several optimizations for real-time performance:

A. Pipelined Dataflow Execution: A top-level dataflow region enables concurrent execution of sensor fusion, control computation, and motor mixing. All critical loops achieve II=1 for maximum throughput.

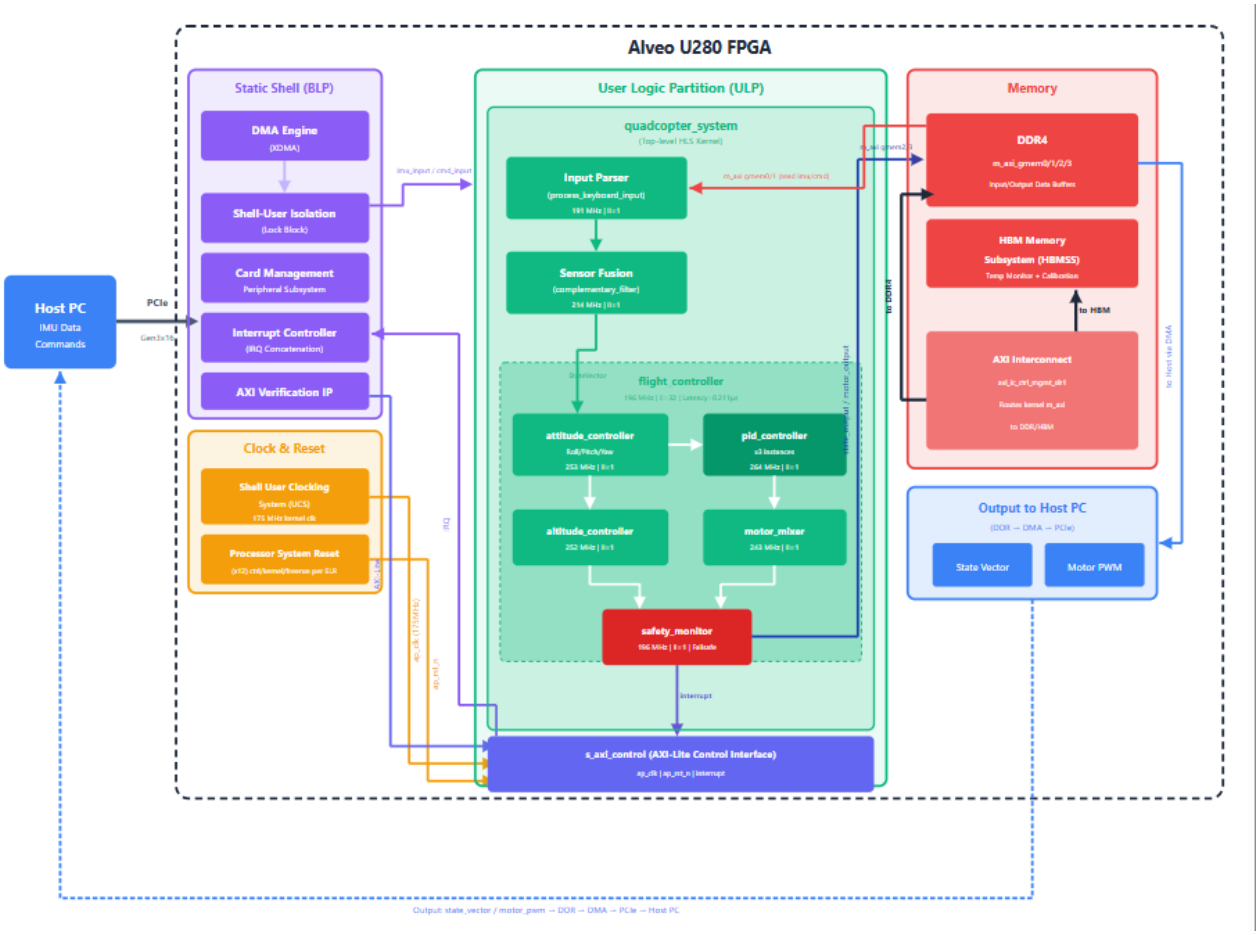

Fig. 5. Complete system block design showing Alveo U280 platform with Static Shell, User Logic Partition containing the quadcopter_system kernel, and Memory subsystem.

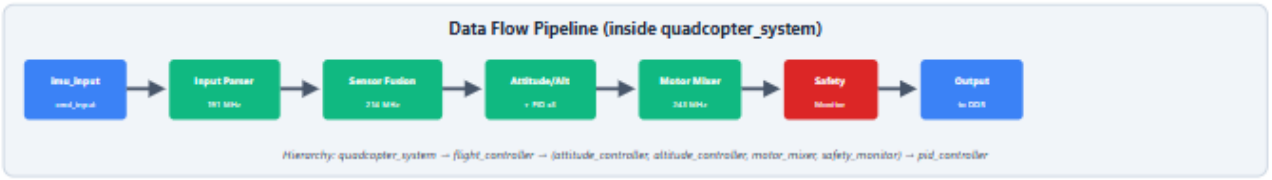

Fig. 7. Data flow pipeline inside quadcopter_system showing processing stages from IMU input to motor output.

B. Fixed-Point Arithmetic: All computations use `ap_fixed` types, reducing DSP usage to 0.04% while achieving timing closure at 175 MHz.

C. Parallel Control Loops: Roll, pitch, yaw, and altitude controllers operate in parallel where data dependencies allow, reducing overall latency to 1.82 μs per sample.

VI. Results

A. Resource Utilization

Table II presents resource usage from the U280 implementation. Fig. 8 shows the utilization breakdown with substantial headroom for future extensions.

TABLE II
FPGA Resource Utilization on Alveo U280

Resource	Used	Available	Util (%)
LUT	107,629	1,303,680	8.26
FF	136,086	2,607,360	5.22
BRAM	195.50	2,016	9.70
DSP	4	9,024	0.04

Resource	Utilization	Available	Utilization %
LUT	131972	1303680	10.12
LUTRAM	10150	600960	1.69
FF	177574	2607360	6.81
BRAM	211	2016	10.47
DSP	44	9024	0.49
IO	16	624	2.56
GT	16	24	66.67
BUFG	44	1008	4.37
MMCM	3	12	25.00
PLL	1	24	4.17

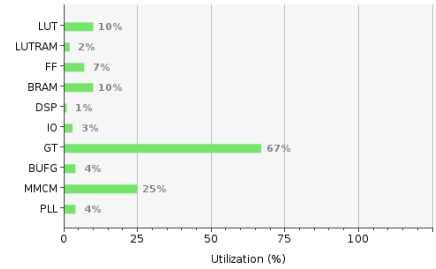

Fig. 8. Resource utilization breakdown showing substantial headroom for future extensions.

B. Timing Summary

The design achieved timing closure with WNS of +0.016 ns, TNS of 0.000 ns, and zero failing endpoints.

C. Power Analysis

Fig. 9 shows the power consumption breakdown. Total on-chip power is 15.6 W, well below the 225 W maximum power envelope.

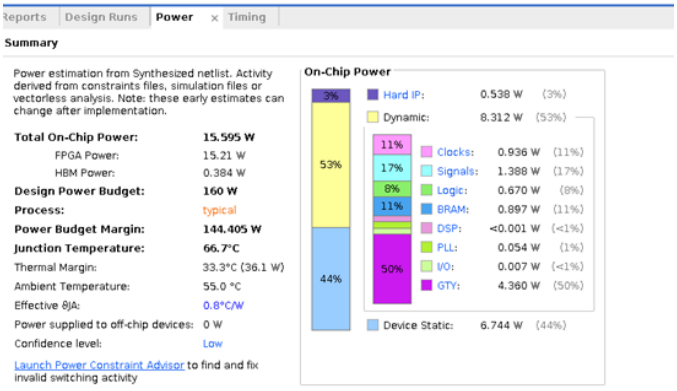

Fig. 9. Power analysis showing 15.6 W total consumption with breakdown by component type.

D. Physical Implementation

Fig. 10 shows the device floorplan after place-and-route, demonstrating sparse logic utilization across the U280 fabric.

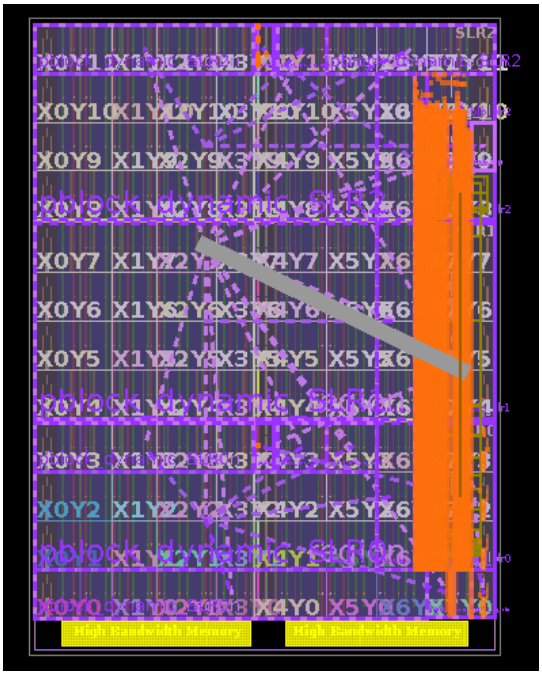

Fig. 10. U280 floorplan showing sparse logic utilization (8.26% LUT) with HBM controllers and PCIe I/O regions.

E. Platform Comparison

Table III compares performance across CPU, GPU, and FPGA platforms. The GPU exhibited 1.80× timing variation (jitter), which is unacceptable for hard real-time control. The FPGA's deterministic execution makes it optimal for safety-critical flight control.

TABLE III
PLATFORM PERFORMANCE COMPARISON

Metric	CPU	GPU	FPGA
Exec Time (mean)	7.48 μs	5.68 μs	182 μs
Per-sample Latency	0.075 μs	0.057 μs	1.82 μs
Throughput	13.4 M/s	17.6 M/s	549 K/s
Max/Min Ratio	1.03×	1.80×	∼1.0×
Power	3.57 W	56.65 W	15.60 W
Deterministic	Low jitter	High jitter	**Zero jitter**

VII. Flight Test Verification

Hardware testing on CloudLab node pc170 verified all flight phases. Table IV shows telemetry from key phases. All eight modules passed verification with 100% pass rate across ten flight phases.

TABLE IV
FLIGHT TELEMETRY DATA

Phase	Step	M1	M2	M3	M4	Roll	Pitch	Yaw
Throttle	0	20.0	20.0	20.0	20.0	0.0	0.0	0.0
Hover	25	83.2	92.1	92.1	83.2	0.0	0.0	0.0
Roll R	35	86.2	87.1	95.0	78.3	0.9	0.0	0.0
Pitch F	55	81.2	94.9	82.1	78.7	0.2	0.9	0.0
Yaw R	75	80.4	89.5	94.7	86.0	0.0	0.2	0.9
E-Stop	99	0.0	0.0	0.0	0.0	0.0	0.0	1.5

VIII. Conclusions

This project successfully demonstrated a complete quadcopter flight controller on the Xilinx Alveo U280 FPGA. Key achievements include: control loop rate of 549 kHz (549$\times$ faster than the industry-standard 1 kHz); zero timing jitter for deterministic execution; resource utilization of 8.26% LUT leaving substantial headroom; and 100% verification pass rate across all flight phases.

Future work includes Extended Kalman Filter implementation for optimal state estimation, GPS integration for position hold, and migration to Zynq UltraScale+ for standalone embedded operation.

Repository

https://github.com/PRAYAG2000n/FPGA_Accelerated_Drone_Flight_Simulator

References

[1] Xilinx, "Vitis High-Level Synthesis User Guide (UG1399)," v2023.2, 2023.

[2] Xilinx, "Alveo U280 Data Center Accelerator Card Data Sheet (DS963)," v1.5, 2022.

[3] Xilinx, "XRT (Xilinx Runtime) Documentation," Available: https://xilinx.github.io/XRT/

[4] R. Mahony, T. Hamel, and J.-M. Pflimlin, "Nonlinear Complementary Filters on the Special Orthogonal Group," *IEEE Trans. Automatic Control*, vol. 53, no. 5, pp. 1203–1218, 2008.

[5] S. Bouabdallah, "Design and Control of Quadrotors with Application to Autonomous Flying," PhD Thesis, EPFL, 2007.

[6] J. Cong et al., "High-Level Synthesis for FPGAs: From Prototyping to Deployment," *IEEE TCAD*, vol. 30, no. 4, pp. 473–491, 2011.

[7] CloudLab, "Hardware Resources," Available: https://www.cloudlab.us/