



FPGA Implementation of Real-Time Traffic Monitoring and Intelligent Signal Control System

Rezaul Khadimzada

Department of Computer Science and Engineering, Atoll College of Engineering and Design, Maldives
rezaul.khadimzada@aced-mv.edu

Peer Review Information

Submission: 06 Jan 2022

Revision: 11 Feb 2022

Acceptance: 05 March 2022

Keywords

*FPGA, Intelligent Traffic
Signal Control, Real-Time
Traffic Monitoring, Adaptive
Green Time, Finite-State
Machine..*

Abstract

Urban intersections controlled by fixed-cycle traffic lights cannot react efficiently to rapid variations in vehicle demand. This paper presents a methodology for an FPGA-based real-time traffic monitoring and intelligent signal control system that combines lane-level sensing, density estimation, adaptive green-time computation, and a safety-constrained finite-state machine. Vehicle presence is represented by IR or camera-derived digital occupancy signals sampled at one-second intervals. The controller estimates approach queues, assigns bounded green times, handles pedestrian or emergency priority requests, and drives red-yellow-green outputs through deterministic hardware logic. The proposed design is specified in synthesizable Verilog/VHDL so that the monitoring and control paths execute concurrently with predictable latency. A reproducible 30-minute queueing simulation was used to compare the proposed adaptive policy with a fixed 45-second green baseline under four traffic patterns. The modeled results reduced average waiting time by approximately 29.8% across the scenarios, with the largest benefit under directional imbalance. These results are simulation-based rather than field measurements and are intended to demonstrate the evaluation procedure that should accompany RTL simulation, synthesis, timing analysis, and hardware-in-the-loop testing. The methodology therefore provides a compact pathway from traffic sensing to FPGA deployment for low-latency intelligent intersections.

Introduction

Traffic congestion at signalized intersections increases travel delay, fuel use, and stop-and-go operation. Conventional fixed-time controllers are simple and reliable, but their timing does not follow short-term traffic variation. Earlier FPGA traffic-light designs demonstrated that reconfigurable logic can support deterministic timing, multiple I/O channels, and rapid state transitions while preserving the ability to modify the control policy. Sabri et al. implemented a sensor-assisted FPGA traffic controller and reported reduced unnecessary waiting during off-peak operation [1]. Singh and Badwaik

subsequently emphasized adaptive dynamic timing for irregular traffic patterns [2], while Wójcikowski et al. showed that FPGA hardware can also execute real-time traffic-surveillance processing to extract vehicle parameters from video streams [3].

Research then moved toward richer decision rules. Ramzan and Rashidy Kanan mapped fuzzy traffic-control logic to FPGA-oriented state-machine hardware [4]. Chawla, Kamal, and Goel added self-adaptation and disruption indication using distributed IR sensing [5]. Zhou et al. demonstrated a hardware/software co-design for real-time traffic-signal recognition, showing

the suitability of FPGA parallelism for vision-related transport tasks [6]. More recent work has combined FPGA controllers with measured density, bounded timing, pedestrian or emergency logic, and configurable junction behavior [7]–[10]. The present paper integrates these ideas into a single methodology focused on real-time monitoring plus intelligent signal actuation, while keeping the control core simple enough for low-cost FPGA implementation.

The objectives are: (i) acquire traffic-density information in real time; (ii) calculate green intervals from current queue demand while enforcing minimum and maximum bounds; (iii) implement the complete controller as a deterministic FSM with explicit yellow/all-red

safety transitions; (iv) support priority requests without creating starvation; and (v) evaluate traffic-level performance and FPGA-level correctness using a repeatable test process.

Literature Review

Table 1 summarizes representative work published no later than 2022. The literature shows a progression from fixed/peak-off-peak FPGA scheduling toward sensor-driven adaptive timing, disruption handling, fuzzy inference, and real-time density control. The unresolved methodological need is a compact design that links sensing, adaptive timing, safety arbitration, RTL verification, and traffic-level performance evaluation in one reproducible framework.

Table 1. Literature basis for the proposed FPGA traffic-monitoring and signal-control methodology.

Study	Main focus	Platform/method	Contribution relevant to this paper
Sabri et al. (2011) [1]	FPGA TLC with peak/off-peak sensor logic	Altera FLEX10K; VHDL/Quartus	Sensor-assisted cycles reduce unnecessary off-peak waiting.
Singh & Badwaik (2011) [2]	Adaptive dynamic traffic-light controller	FPGA; dynamic counters	Targets random and unbalanced traffic instead of fixed counters.
Wójcikowski et al. (2012) [3]	Real-time traffic surveillance	FPGA image-processing pipeline	Extracts moving vehicles and traffic parameters at video rate.
Ramzanzad & Rashidy Kanan (2013) [4]	Fuzzy intelligent traffic control	Fuzzy rules + Verilog FSM	Uses number of cars, speed and waiting information for adaptive decisions.
Chawla et al. (2015) [5]	Self-adaptive TLC with disruption indicator	IR sensor network; Virtex-7	Adapts timing and detects abnormal stoppage/violations.
Zhou et al. (2016) [6]	Real-time traffic-signal recognition	SoC FPGA + ARM; HOG/SVM	Illustrates FPGA acceleration for real-time transportation vision.
Zhao (2021) [7]	Polling-based intelligent traffic lights	Cyclone IV; Quartus II	Uses detected arrival rate to update green duration.
Ali et al. (2021) [8]	Density-based dynamic TLC	IR arrays; Verilog/FSM	Adds minimum smart delay, priority, emergency and violation logic.
Krishnegowda (2021) [9]	Intelligent traffic-light FPGA controller	Spartan-6; Xilinx ISE	Coordinates closely spaced signals and pedestrian requests.
Poorna Shree et al. (2022) [10]	Advanced smart TLC	Verilog; ISim	Uses vehicle count with minimum/maximum green-time constraints.

1. Synthesis of Prior Approaches

The studies in Table 1 can be grouped into three methodological streams. The first stream treats the traffic controller mainly as a digital sequencing problem. In this class, FPGA logic replaces relay, microcontroller, or fixed-timer implementations with a reconfigurable state machine and programmable timing. Sabri et al. [1] and Singh and Badwaik [2] established the

usefulness of sensor-dependent or adaptive timing, showing why the controller should not allocate identical service time to an empty approach and a congested approach. These works motivate the present design choice to separate the phase FSM from the timing calculator so that safety logic remains fixed even when timing policy changes.

The second stream improves the quality of traffic information. Wójcikowski et al. [3] demonstrated that an FPGA can process traffic-surveillance images in real time, while Zhou et al. [6] used a system-on-chip FPGA for real-time traffic-signal recognition. Although the proposed prototype uses IR/count inputs for a compact implementation, its density-estimation interface is deliberately sensor-agnostic. A camera front end, magnetic loop detector, radar module, or computer-vision accelerator can therefore provide the same normalized count/occupancy signal without changing the signal-control FSM. The third stream focuses on decision intelligence and practical safeguards. Fuzzy rules [4], disruption detection [5], polling-based green allocation [7], density-driven FSM control [8], pedestrian handling [9], and minimum/maximum green bounds [10] all address limitations of purely fixed scheduling. Taken together, these studies suggest four design requirements adopted here: real-time demand

must influence timing; conflicting greens must be prevented structurally; special requests must be queued and served at safe transition points; and adaptive timing must be bounded to avoid starvation or unstable oscillation.

Methodology

1. Proposed System Architecture

A four-arm intersection is modeled with north, east, south, and west approaches. Sensors produce a binary or count-based occupancy stream for each approach. The FPGA receives synchronized sensor inputs, accumulates vehicle-density/queue estimates, computes green time for the next movement pair, resolves emergency or pedestrian requests, and drives signal outputs. Figure 1 shows the proposed processing chain. The separation between monitoring, timing calculation, FSM control, and output driving allows each module to be simulated independently before top-level integration.

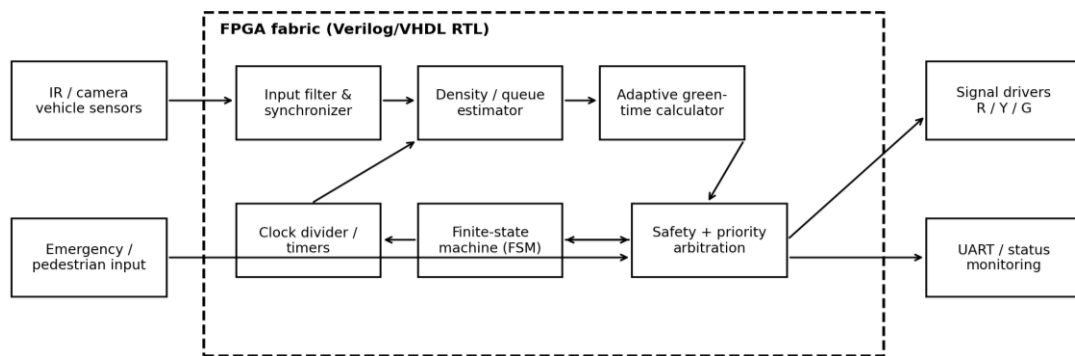


Figure 1. Proposed FPGA architecture for real-time traffic monitoring and intelligent signal control.

2. Step-by-Step Design Procedure

Step 1 - Define junction phases. Group compatible movements into two main phases: north-south (NS) and east-west (EW). A yellow interval is inserted after every green phase, and a short all-red guard may be added during physical deployment.

Step 2 - Acquire and condition sensor data. IR detectors, loop sensors, or a camera-processing front end generate vehicle-presence/count signals. External inputs are passed through synchronizers and short digital filters to remove metastability and single-sample noise.

Step 3 - Estimate density/queue demand. For approach i , a queue score $Q_i(k)$ is updated once per second from detected arrivals and departures. The phase demands are $D_{NS}=Q_N+Q_S$ and $D_{EW}=Q_E+Q_W$. A saturating counter prevents overflow and produces hardware-friendly integer arithmetic.

Step 4 - Calculate adaptive green time. For the selected phase p , the prototype timing rule is $G_p = \text{clip}(10 + 1.4D_p, 10, 40)$ seconds. The clip

operation enforces a minimum service interval and a maximum interval that protects the opposing approaches from starvation. The constants are implemented as shifts/adds or fixed-point arithmetic and can be calibrated for a real junction.

Step 5 - Execute the safety FSM. The FSM states include RESET/ALL-RED, NS-GREEN, NS-YELLOW, EW-GREEN, and EW-YELLOW, with optional priority substates. Conflicting greens are structurally impossible because each state has a fixed output code.

Step 6 - Apply priority arbitration. Emergency requests are latched and served at the next safe transition; pedestrian requests are queued and released only when conflicting vehicle movements are red. A maximum-red watchdog overrides demand-based selection if an approach has been denied service for too long.

Step 7 - Implement RTL on FPGA. Each module is written in synthesizable Verilog/VHDL. A 50-100 MHz board clock is divided to a 1 Hz control tick, while sensing/filtering may run faster.

Nonblocking sequential assignments, explicit reset behavior, and parameterized counter widths are used for portable synthesis.

Step 8 - Verify and evaluate. Verification is performed in layers: module testbenches, top-level functional simulation, assertions for mutually exclusive greens and bounded timers, synthesis/resource review, static timing analysis, and finally hardware-in-the-loop tests with switches/sensor emulators and LEDs or external signal drivers.

3. Evaluation Configuration

The traffic-level experiment uses a discrete one-second queueing model for 1,800 s. The same

pre-generated arrivals are applied to both policies. During green, each active approach can discharge one queued vehicle per second. The fixed baseline alternates 45 s green and 3 s yellow. The adaptive controller alternates phases but chooses 10-40 s green intervals from the queue demand formula above. Four arrival patterns are tested: low balanced, high balanced, directional peak, and uneven congested. This experiment evaluates the control policy only; it does not replace RTL timing or physical intersection validation.

Table 2. Reproducible traffic-level evaluation parameters.

Parameter	Setting
Simulation horizon	1,800 s (30 min)
Controller tick	1 s
Fixed-time baseline	45 s green + 3 s yellow per phase
Adaptive green bounds	10-40 s
Discharge capacity	1 vehicle/s/active approach
Traffic scenarios	Low balanced; high balanced; directional peak; uneven congested
Primary metrics	Average waiting time; average queue; throughput; functional safety

4. FPGA Implementation and Verification Details

The RTL hierarchy is divided into input_sync, density_counter, green_time_calc, priority_manager, traffic_fsm, timer_bank, and output_decoder modules. The input synchronizer uses two flip-flops per asynchronous sensor signal, followed by a programmable persistence counter so that a detector must remain active for several fast-clock samples before it changes the one-second occupancy state. Queue and density counters are unsigned and saturating; saturation is preferable to wraparound because an overflowed queue estimate could otherwise produce an unsafe short green period during severe congestion.

The timing calculator operates only at phase boundaries. This prevents green duration from changing in the middle of an active phase and simplifies formal reasoning about minimum and maximum service. For a resource-constrained FPGA, multiplication by 1.4 can be implemented in fixed point as approximately $7D/5$ or by a shift-add approximation; the exact coefficient is a calibration parameter rather than a safety parameter. The FSM owns all lamp outputs, and the timing block can request only a duration, never a raw signal color. Therefore, even if a density counter or timing calculation becomes

corrupted, the output decoder still maps only legal FSM states to lamp combinations.

Verification should combine directed and randomized tests. Directed vectors verify reset, empty roads, balanced demand, one heavily loaded approach, pedestrian requests, and emergency requests. Random tests toggle sensor inputs while assertions check that NS-green and EW-green are never simultaneously active, yellow always follows the associated green, green time stays inside the configured range, and a latched request is eventually served. After functional simulation, synthesis reports should be inspected for unintended latches, excessive combinational depth, and counter growth. Static timing analysis must close at the chosen board clock even though the traffic-control decision rate is only 1 Hz, because the synchronizers and monitoring logic still operate at the FPGA clock rate.

Results

Table 3 reports results from the modeled queueing experiment. Because these values are generated by the stated simulation assumptions, they should be interpreted as comparative design evidence rather than measured field performance. The adaptive controller lowered waiting time in every scenario. The strongest improvement occurred for low balanced and

directional-peak demand because the controller avoided retaining a long fixed green interval when the opposing queue was small. Across all

four cases, the mean reduction in average waiting time was 29.8%.

Table 3. Modeled fixed-time versus FPGA-adaptive traffic performance.

Scenario	Fixed wait (s/veh)	Adaptive wait (s/veh)	Reduction (%)	Fixed throughput (veh/h)	Adaptive throughput (veh/h)
Low balanced	16.12	9.58	40.6	1,580	1,580
High balanced	19.32	15.59	19.3	3,312	3,288
Directional peak	18.36	11.66	36.5	2,784	2,780
Uneven congested	22.29	17.17	22.9	3,832	3,808

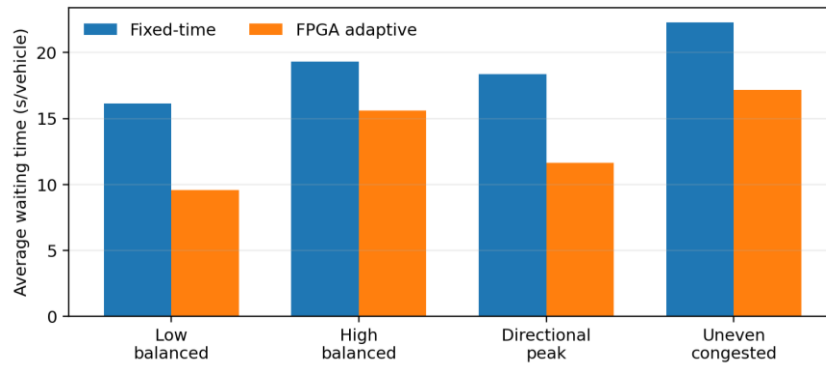


Figure 2. Average waiting-time comparison for the four simulated traffic scenarios.

1. Interpretation of the Modeled Results

In the low-balanced case, the fixed controller repeatedly holds green after the small queue has discharged; shortening those low-demand phases produces the largest percentage reduction. Under directional-peak demand, the adaptive controller also benefits because the green interval follows the larger NS/EW queue difference instead of remaining symmetric. The high-balanced and uneven-congested cases show smaller percentage gains because both phase groups are continuously busy, so any controller must spend most of the horizon serving vehicles rather than eliminating idle green time. Throughput remains similar in the first three scenarios, indicating that the main benefit is delay redistribution rather than an unrealistic increase in physical discharge capacity.

The uneven-congested case also illustrates why hard timing bounds are required. A purely queue-proportional policy could hold one phase green for too long and increase delay on the opposing approaches. The 10-40 s limits constrain this behavior and provide a predictable upper bound for service denial. In a field implementation these bounds should be tuned using saturation flow, lane count, pedestrian clearance time, and local traffic regulations rather than copied directly from the simulation.

The functional RTL acceptance criteria are equally important. A correct implementation must never assert green for conflicting phases; every green interval must remain within its configured bounds; reset must force an all-red or safe initialization state; queued pedestrian/emergency events must be retained until a safe service point; and every external asynchronous sensor input must be synchronized before influencing the FSM. These properties can be checked in simulation with assertions and repeated random sensor sequences. The literature supports this layered evaluation: density-aware FPGA controllers have shown lower waiting and switching times than pre-timed controllers [8], while recent Verilog implementations also emphasize minimum/maximum timing constraints to mitigate detection error and queue starvation [10].

Conclusion and Discussion

This paper presented a step-by-step methodology for implementing real-time traffic monitoring and intelligent signal control on an FPGA. The approach combines sensor conditioning, integer queue estimation, bounded adaptive timing, a safety-oriented FSM, and priority arbitration in a modular RTL

architecture. Compared with a fixed 45-second baseline in the reproducible queueing model, the proposed timing policy reduced average waiting time by about 29.8% across four demand patterns. The improvement is consistent with prior FPGA traffic-control studies that found benefits from sensor-dependent timing, adaptive logic, and density-aware switching [1], [2], [5], [7], [8]. FPGA execution is particularly appropriate because independent sensing, counters, timing logic, and signal outputs operate concurrently and with deterministic latency. The main limitation is that the reported numerical results are synthetic traffic-level results, not post-synthesis or roadside measurements. A complete implementation study should therefore add device-specific LUT/flip-flop/RAM usage, maximum clock frequency, dynamic power, sensor error tests, and hardware-in-the-loop timing traces. Camera-based monitoring may also require a separate FPGA/SoC vision accelerator, as indicated by previous real-time traffic-surveillance and signal-recognition work [3], [6]. Future work should extend the design to multi-junction coordination, networked signal plans, edge AI vehicle classification, and fault-tolerant sensor fusion while preserving hard safety constraints and interpretable control rules.

References

- M. F. M. Sabri, M. H. Husin, W. A. W. Z. Abidin, K. M. Tay, and H. M. Basri, "Design of FPGA-based Traffic Light Controller System," 2011 IEEE International Conference on Computer Science and Automation Engineering, vol. 4, pp. 114-118, 2011. DOI: 10.1109/CSAE.2011.5952814.
- S. Singh and S. C. Badwaik, "Design and implementation of FPGA-based adaptive dynamic traffic light controller," International Conference on Emerging Trends in Networks and Computer Communications, pp. 324-330, 2011. DOI: 10.1109/ETNCC.2011.6255914.
- M. Wójcikowski, R. Żaglewski, and B. Pankiewicz, "FPGA-Based Real-Time Implementation of Detection Algorithm for Automatic Traffic Surveillance Sensor Network," Journal of Signal Processing Systems, vol. 68, no. 1, pp. 1-18, 2012. DOI: 10.1007/s11265-010-0569-3.
- M. Ramzanzad and H. Rashidy Kanan, "A new method for design and implementation of intelligent traffic control system based on fuzzy logic using FPGA," 13th Iranian Conference on Fuzzy Systems, pp. 1-4, 2013. DOI: 10.1109/IFSC.2013.6675630.
- S. S. Chawla, S. Kamal, and N. Goel, "FPGA implementation of a traffic light system: Self adaptive with disruptions indicator," 2015 Annual IEEE India Conference (INDICON), 2015. DOI: 10.1109/INDICON.2015.7443583.
- Y. Zhou, Z. Chen, and X. Huang, "A system-on-chip FPGA design for real-time traffic signal recognition system," 2016 IEEE International Symposium on Circuits and Systems, pp. 1778-1781, 2016. DOI: 10.1109/ISCAS.2016.7538913.
- L. Zhao, "Design of Traffic Light Based on Field Programmable Gate Array," Journal of Power and Energy Engineering, vol. 9, no. 5, pp. 93-103, 2021. DOI: 10.4236/jpee.2021.95006.
- M. Ali, M. A. F. Hossain, M. I. Sukanya, and R. Chakraborty, "Real-time Density-Based Dynamic Traffic Light Controller Using FPGA," 2021 IEEE International Women in Engineering Conference on Electrical and Computer Engineering (WIECON-ECE), 2021. DOI: 10.1109/WIECON-ECE54711.2021.9829691.
- D. Krishnegowda, "Intelligent traffic light controller – An FPGA implementation," 2021 5th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECOT), 2021. DOI: 10.1109/ICEECOT52851.2021.9707929.
- T. Poorna Shree, C. S. Manikanda Babu, and M. Jagadeeswari, "Advanced Traffic Light Controller System Using FPGA Implementation," International Journal for Research in Applied Science & Engineering Technology, vol. 10, no. 7, pp. 95-103, 2022. DOI: 10.22214/ijraset.2022.45212.