Problem Statement 2: AI-Driven Traffic Congestion Prediction and FPGA-Based Adaptive Signal Controller

Aastha Bhore, BTech Final year (EEE)
IIT Jodhpur
Github Repository

Abstract

This project presents a Machine Learning + FPGA traffic control system capable of dynamically adapting signal timings according to predicted traffic congestion. A Random Forest model trained using real-world data outputs and four congestion levels. These levels drive a LUT-based adaptive timing engine implemented in Verilog RTL on a PYNQ-Z2 FPGA. Simulation results from Vivado validate the correctness of the design, making it suitable for real-time smart-city deployment.

1 Introduction

Traffic congestion is a major urban challenge. Traditional fixed-time controllers fail to adapt to changing traffic demand, causing inefficiency and delays. This work integrates:

- Machine Learning for congestion prediction,
- FPGA-based real-time timing control,
- LUT-driven adaptive green timing,
- Automatic fail-safe protection.

2 Machine Learning Model: Random Forest

2.1 Dataset and Preprocessing

The model was developed using the Kaggle *Traffic Prediction Dataset* (Dataset Link). All machine learning steps were implemented in the Jupyter Notebook: `Traffic_Predictor.ipynb`.

1. Data cleaning, imputation, and encoding
2. Feature scaling using StandardScaler
3. Training Random Forest Regressor
4. Hyperparameter tuning using RandomizedSearchCV
5. Saving predictions and model metrics

2.2 Model Performance

```
Random Forest Model Metrics:
Mean Squared Error: 16.750267248545303
Root Mean Squared Error: 4.092709035412279
Mean Absolute Error: 2.4334663341645886
R-squared Score: 0.961129473121528
```

Figure 1: Model performance metrics: MSE, RMSE, MAE, and R^2.

2.3 Levels and Predictions

Continuous predictions were mapped into 4 levels:

$$0, 1, 2, 3$$

based on thresholds.

2.4 Exporting ML Output to FPGA

The ML model → FPGA connection uses a **file-based interface**:

1. ML generates predicted levels for every time slot.
2. These are saved into `levels.mem`.
3. Verilog testbench loads them using:

   ```
   $readmemh("levels.mem", level_mem);
   ```

4. Each value is applied to the RTL input $\texttt{congestion}_{level}$.

4. `LUT inside FPGA assigns matching green-time duration.`

This approach ensures:

- No on-chip ML hardware required,
- Fast development and retraining cycle,
- Clean separation between AI and hardware timing logic.

3 System Flowchart

Machine Learning Model (Random Forest)
Predicts traffic volume

↓

Quantized Levels (0–3)
Saved into `levels.mem`

↓

FPGA Traffic Controller (Verilog RTL)
Reads congestion_level from memory

↓

FSM + LUT + Safety Logic
Selects green/yellow/red timings

↓

Traffic Light Outputs (NS/EW)

Figure 2: System flowchart from ML model to FPGA-controlled traffic lights.

4 FPGA Controller Architecture

4.1 FSM States

1. NS Green
2. NS Yellow
3. All-Red 1
4. EW Green
5. EW Yellow
6. All-Red 2

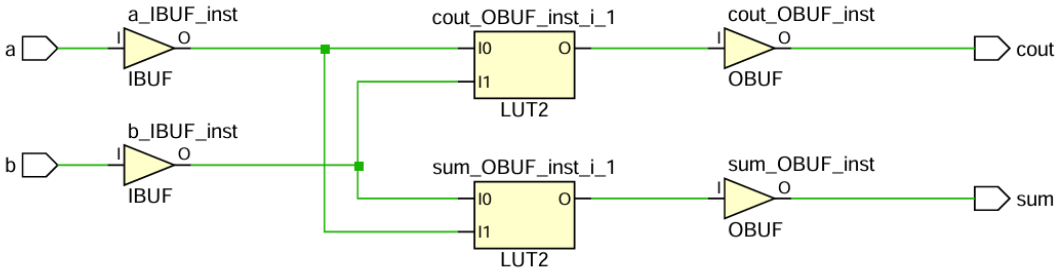

Figure 3: High-level schematic of FSM

4.2 LUT-Based Adaptive Timing

Green time = {4, 6, 10, 16} seconds based on ML congestion level.

Yellow and All-Red are fixed:

2s, 1s

4.3 Fail-Safe

Triggers when ML levels stop changing or manually enabled:

- Overrides green time to 8 seconds,
- Important for safety and determinism.

4.4 Why X Appears Initially

Verilog registers start in unknown (`X`) state until the first clock edge after reset.

5 Simulation Waveform and Results

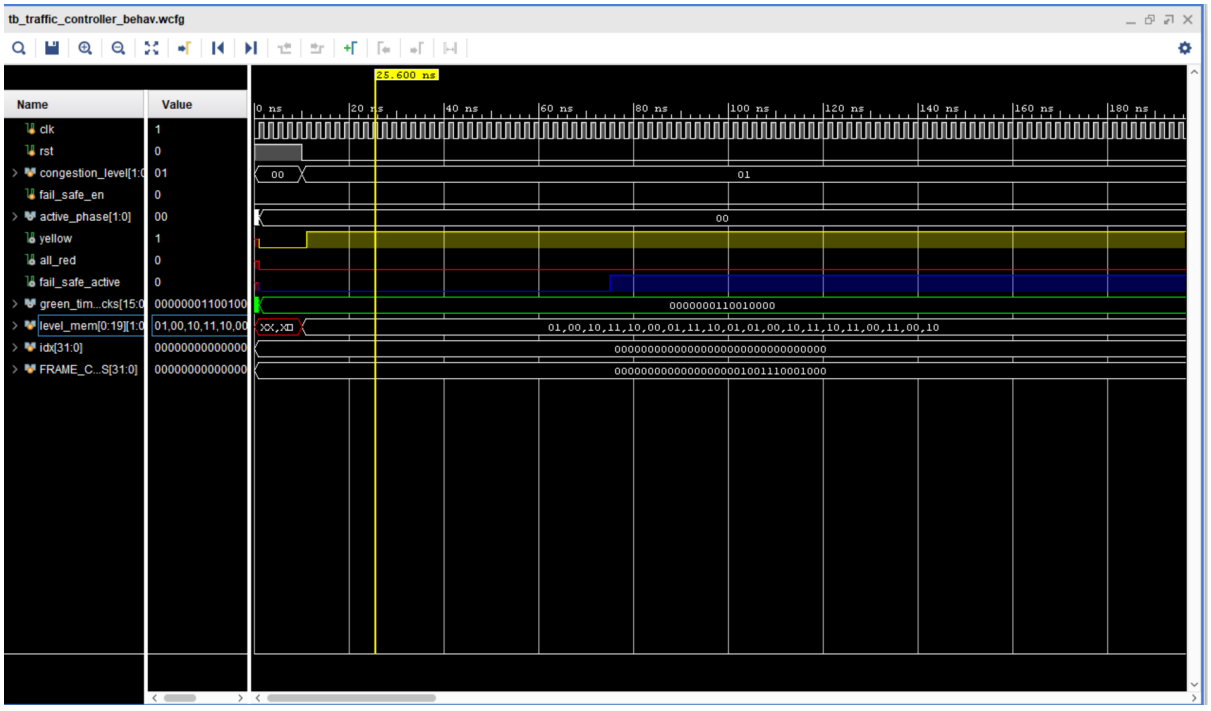

Figure 4: Simulation waveform: ML congestion input, LUT green-time output, FSM transitions, yellow/all-red intervals.

Key Observations

- As seen in this waveform which is shown for initial 20 LEVELS;Congestion levels transition correctly (`00 → 01`).

- The waveform clearly shows that when the ML-predicted congestion level transitions to `01`, the controller loads the corresponding LUT-mapped green time of `0000000110010000` (decimal 400), proving correct ML→LUT→FSM timing adaptation.
- Yellow and All-Red intervals remain fixed and consistent.
- FSM toggles between NS and EW directions as designed.

6 Conclusion

This project demonstrates a complete AI-assisted FPGA traffic controller:

- ML intelligently predicts congestion,
- FPGA provides deterministic, real-time timing control,
- LUT and FSM ensure safe, adaptive signal operation,
- Fail-safe protects against ML instability,
- Simulation confirms correctness.

The architecture is light-weight, scalable, and deployable in real-world smart-traffic systems.