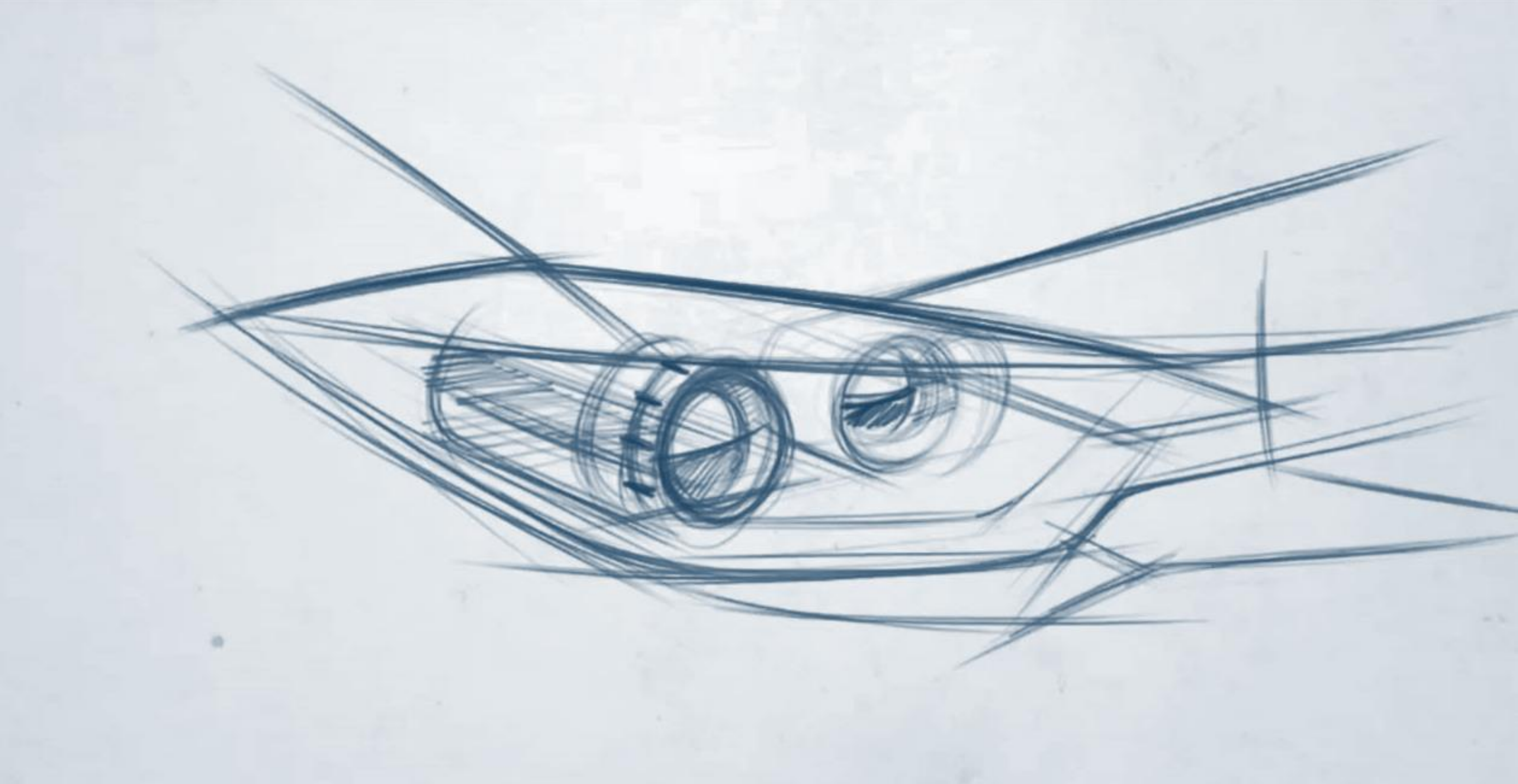


Development of Control Strategy for Adaptive Front-Lighting System using Simulink and Stateflow

ARAI®
Progress through Research

The Automotive Research Association of India

Agenda

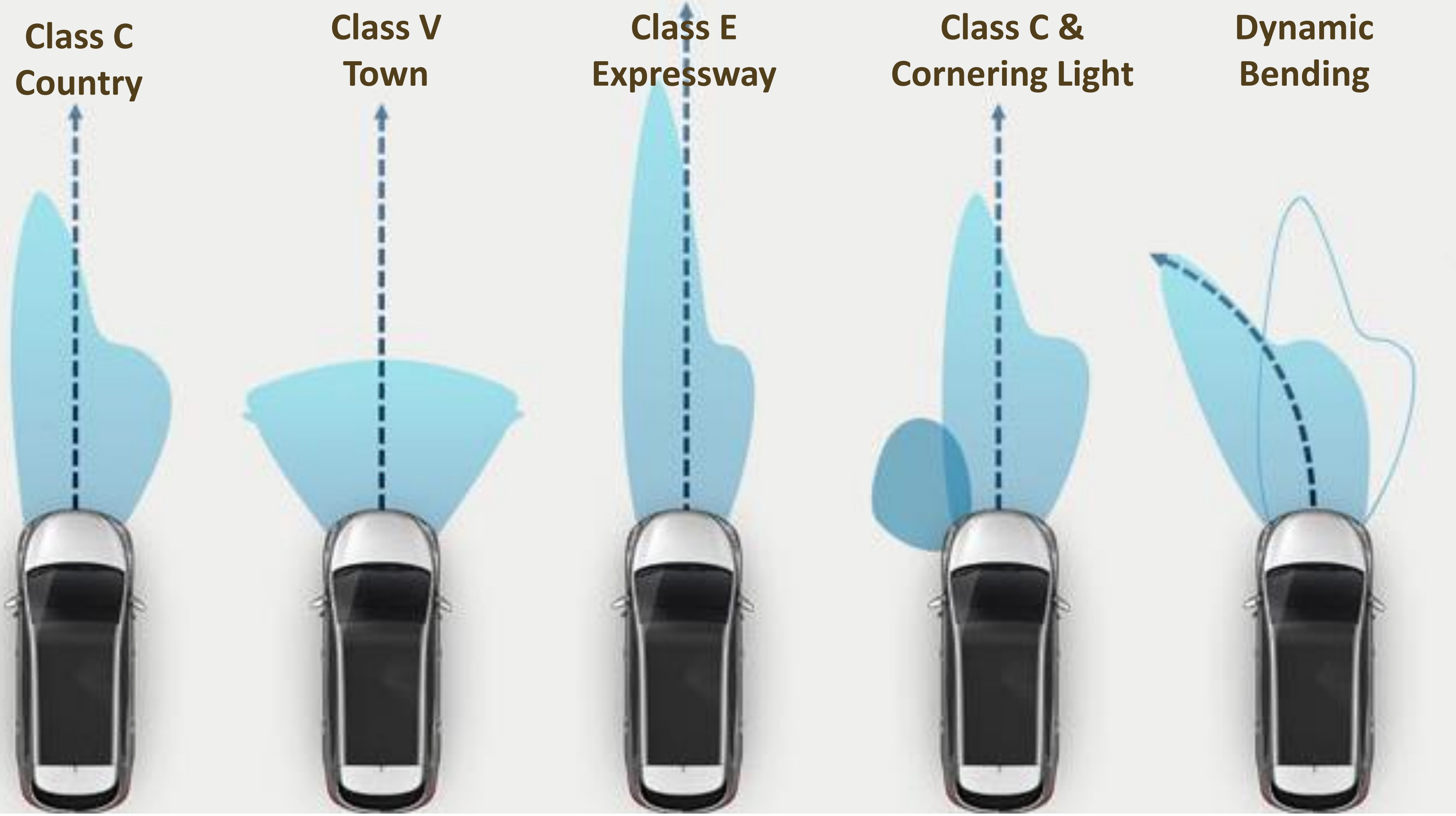
- What is Adaptive Front-Lighting (AFS) System?
- Architecture of AFS
- Objective of the Project
- Challenges faced
- Configuration & Workflow
- Structure of Algorithm
  - Master-Slave Model
- Control Algorithm for Actuators
  - Synchronization between Slaves
  - Position & Speed Control
- Application of Stateflow

Adaptive Front Lighting System

A lighting device, providing beams with differing characteristics for **automatic adaptation to varying conditions** of use of the dipped-beam (passing beam) and, if it applies, the main-beam (driving-beam) with a minimum functional content

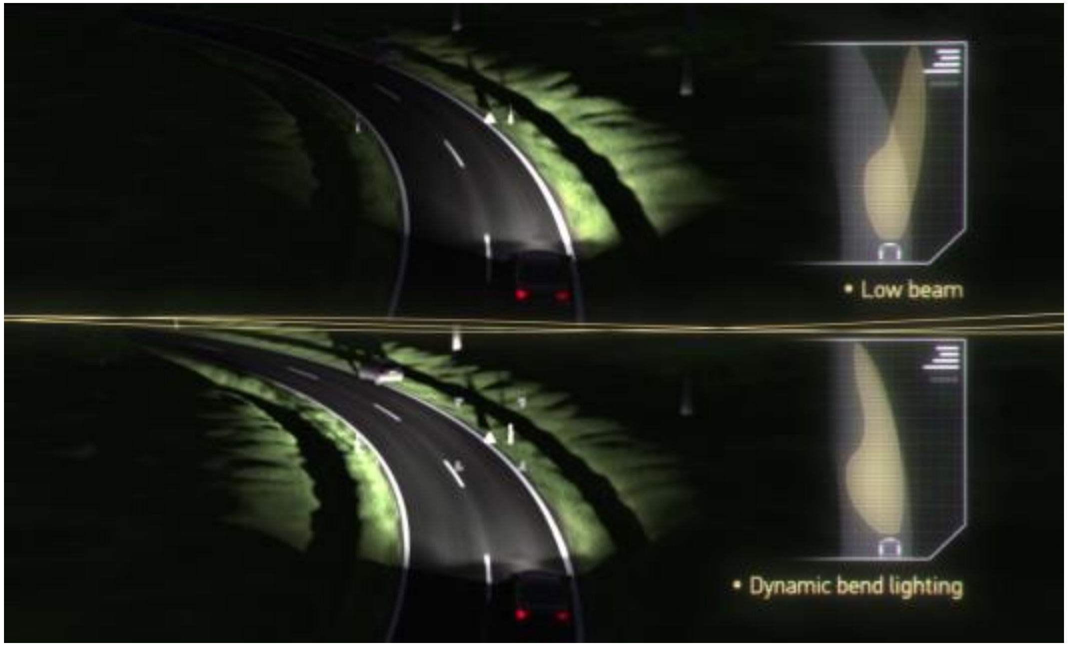
- Class C – Default Mode
- Class V – Vehicle Speed not exceeding 50km/h or roads with fixed illumination
- Class E – For Vehicle Speed greater than 80km/h
- Class W – Windshield Wiper switched on for 2 minutes / Wetness of road detected
- Class R – Normal Driving Beam or Adaptive Driving Beam
- Static Bend Lighting – Lamps projected at fixed angle for bend lighting
- Dynamic Bend Lighting – Light swivels according to Bend Radius

AFS-Classes of Passing Beam

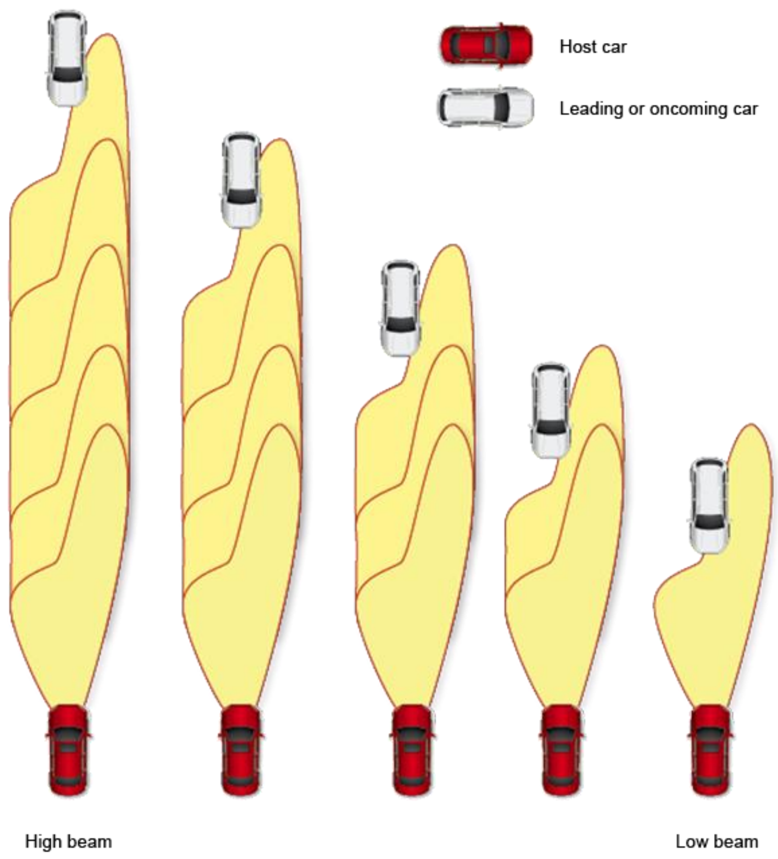


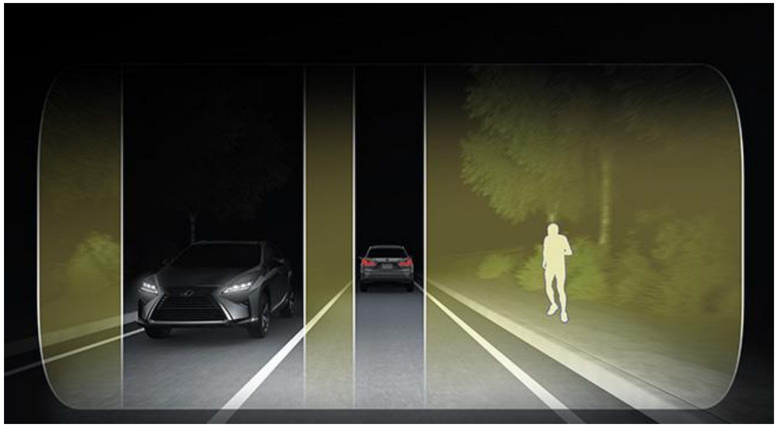
Dynamic Bend Lighting

- Inner Swivel Angle : 7° to 8°
- Outer Swivel Angle : 15°

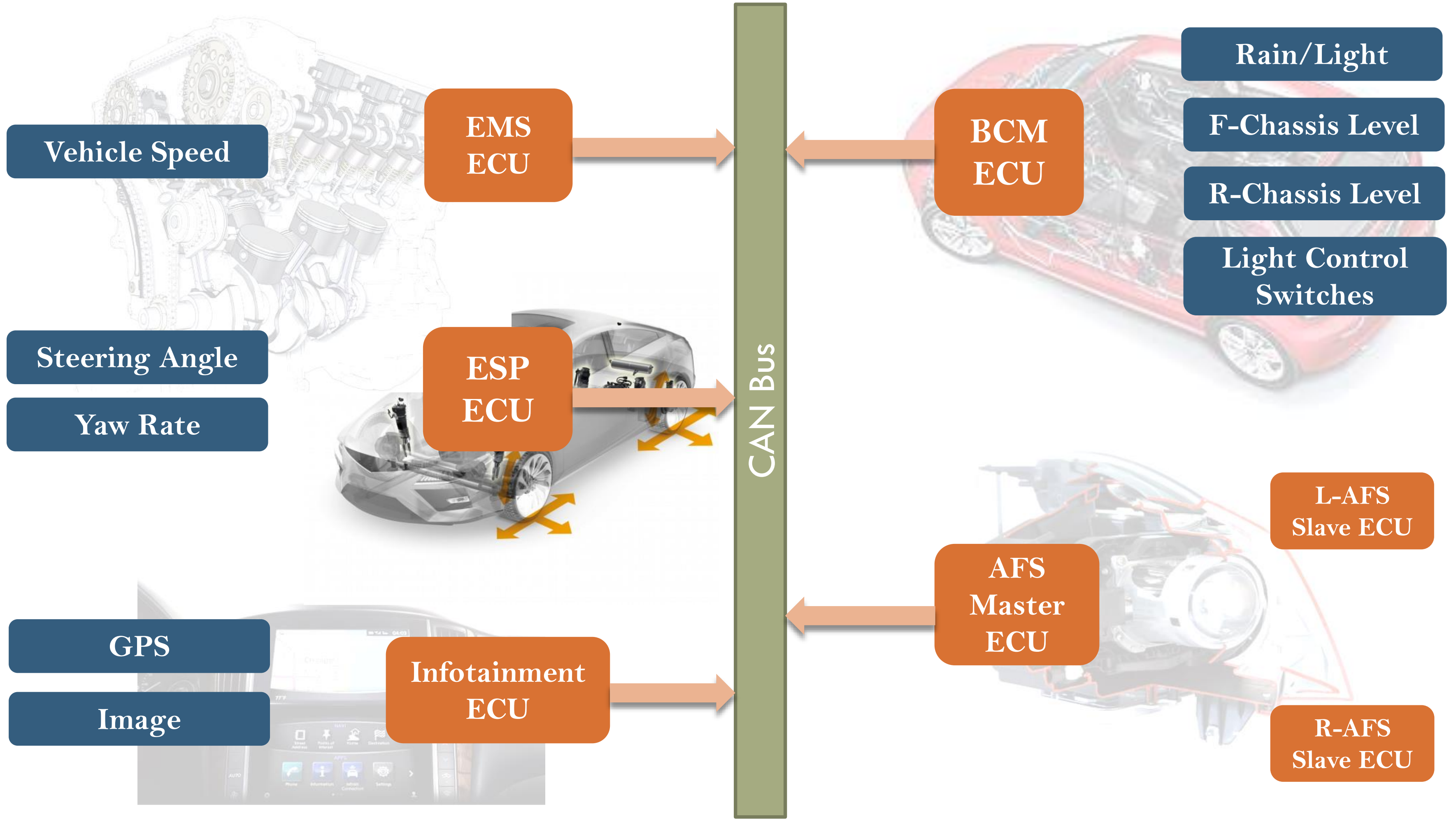


Adaptive Driving Beam



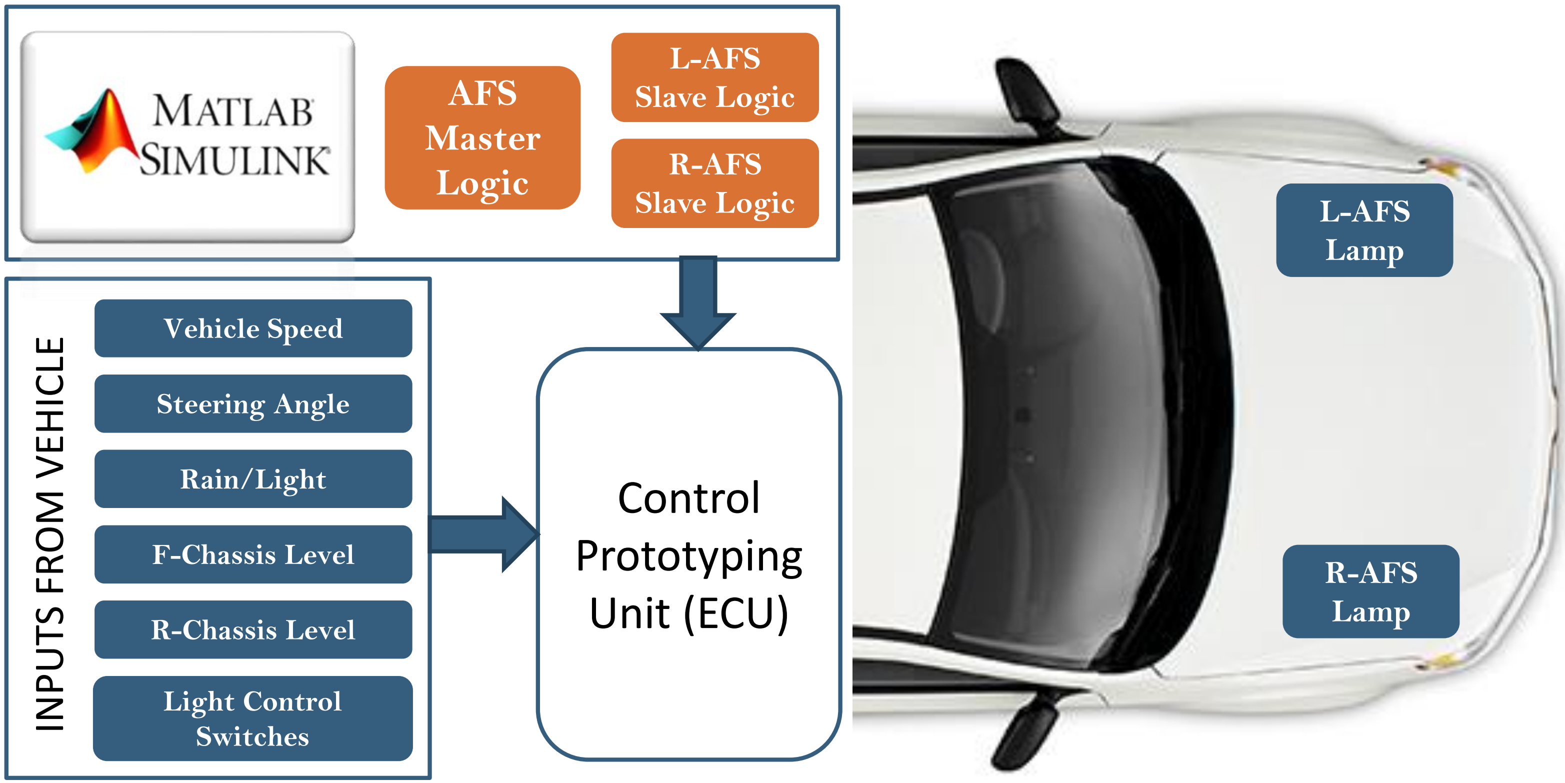


Vehicle Level Architecture



Objective

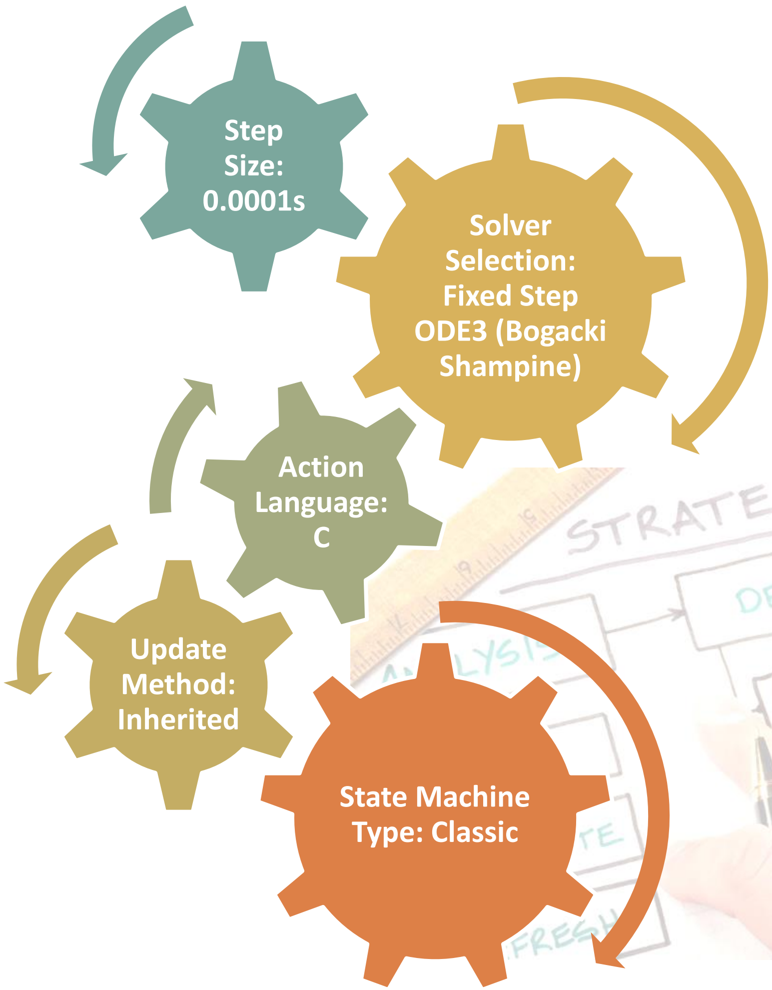
- Development of Control Strategy for AFS Passing Beam – Simulink® & Stateflow®
- Implementation of Control Strategy in Vehicle – Prototyping ECU



Challenges

- Modelling Master- [Slave + Slave] logic in a single model
- Synchronization between both the Slaves
- No use of Position Encoders
- Dynamic Swivelling Control Algorithm
- Smooth transition between Lighting Modes to avoid discomfort

Configuration & Workflow



States

- Identification of operating Modes
- Hierarchy in State Levels

State Actions

- Define Entry, During and Exit Actions of State
- Manage external variables
- Initialize state variables
- Set the behaviour of outputs in Simulink functions

State Transitions

- Default Transitions
- Guard Transitions
- Condition Actions and Transition Actions
- Inner Transitions to simplify chart

Interface

- Invoking the Stateflow Chart
- Events Input and Data Input
- Defining Chart Properties

Simulate & Debug

Structure of Algorithm

Inputs from Vehicle

- CAN Interface blocks from Prototyping ECU

Master ECU Control Logic

- Idle State
- Diagnosis State
- Limp Home State
- Normal State
  - Country
  - Town
  - Expressway
  - Driving Beam
- Reset Actuators
- Tell Tale
- Dynamic Bending algorithm

Right Slave ECU & Left Slave ECU Control Logic

- Self Test Logic
- Diagnosis Logic
- Reset Logic
- Swivelling Logic
- Levelling Logic

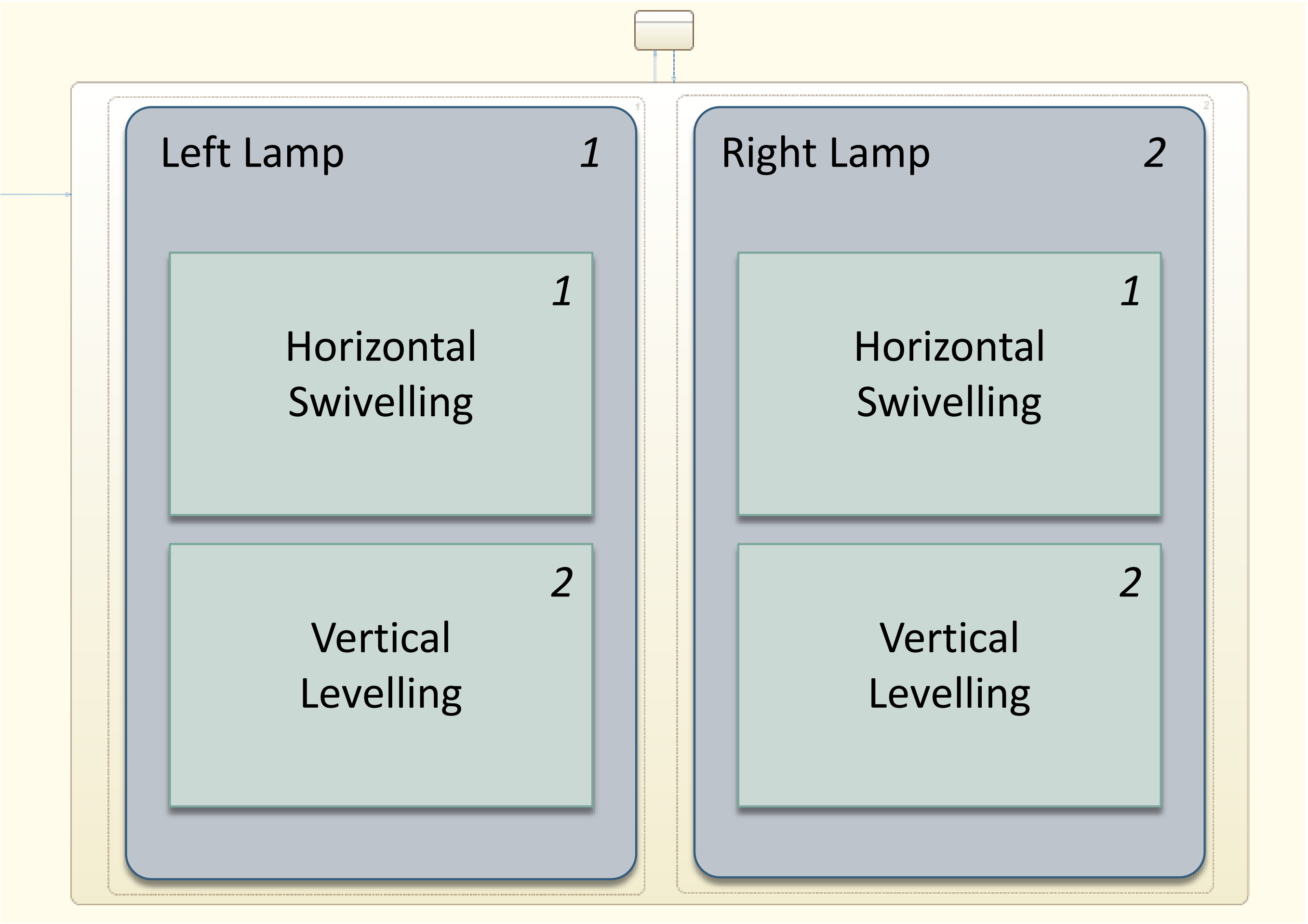
Master-Slave Model

- The model comprises of a Master – [Left Slave & Right Slave] logic
- Feedback to Master from Slave leads to Algebraic Loop Error
- Model looks cluttered if more number of feedback lines are used

Workarounds

- ✓ Use of Memory Blocks eliminating feedback lines
- ✓ Data Store Read block can be used anywhere in the model which eliminates data lines
- ✓ Data Store Memory block had to be placed outside all subsystems common to both Master and Slave

Control Algorithm for Actuators

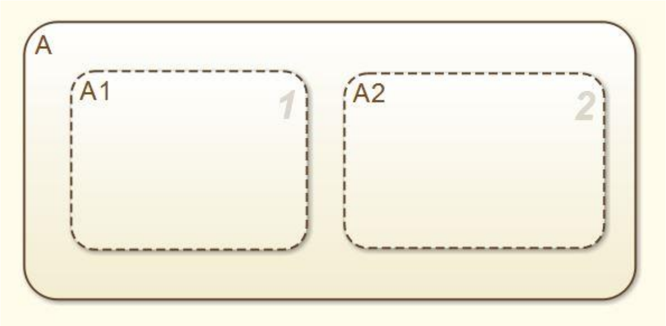


Synchronization between Slaves

- Stateflow charts run on a single thread
- Subsystem based slave logic leads to delay between both the slave actuators

Workarounds

- ✓ Both the slave models were built in single State
- ✓ Sub-States were used for constructing Right Slave and Left Slave entities
- ✓ Parallel AND Decomposition between the Slaves helps Synchronization
- ✓ Execution order of Parallel States can also be Explicit

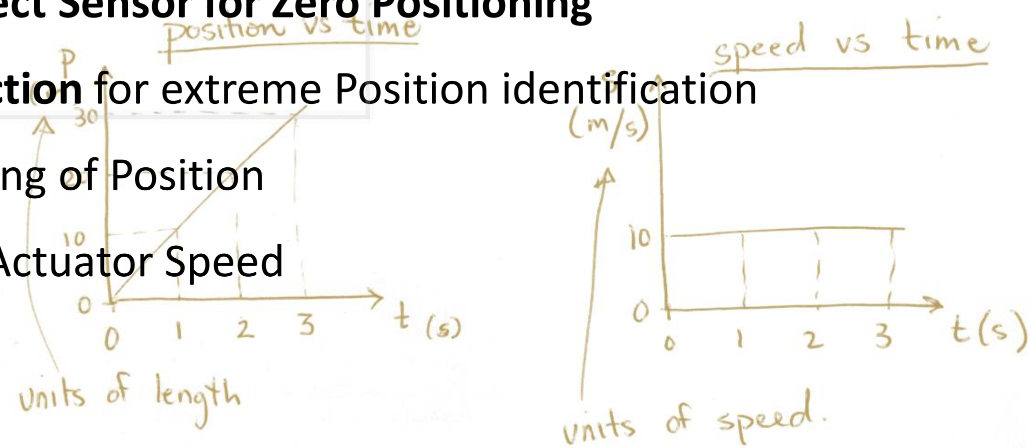


Position & Speed Control

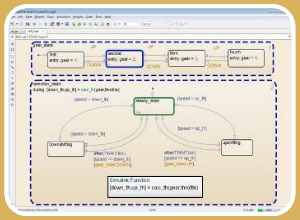
- Vehicle Speed and Steering Angle determine the Speed and Position for Actuator
- No Position encoders are used for continuous feedback
- Horizontal Swiveling Actuator: Hall Effect Sensor for Zero Position Sensing
- Vertical Leveling Actuator: No Sensor for position identification

Workarounds

- ✓ Initial Alignment of both the Actuators
  - Horizontal Actuator: **Hall Effect Sensor for Zero Positioning**
  - Vertical Actuator: **Stall Detection** for extreme Position identification
- ✓ Use of **Counters** for Normal tracking of Position
- ✓ **PWM Period Control** for varying Actuator Speed



Stateflow Application



GUI Support

- Graphical realization of Control Algorithm
- Implementation of Hierarchy in Algorithm
- Monitoring Control flow

Design Support

- Use of Matlab Function, Simulink Function, Truth Table and Graphical Functions
- Control over Simulink function variables
- Flexibility in controlling State Transitions
- Easy to manage variables
- Parallel Computation Algorithms using State Decomposition
- Temporal Operations

Debug Support

- Monitor Variable Changes and Data
- Breakpoints to halt simulation and monitor data
- Indication of State Inconsistencies and Conflicting Transitions
- Indication of Invalid usage of Data

Thank You…

Questions