COMPSYS 723 ASSIGNMENT 2 DESIGN REPORT

Jinkai Zhang, Bing Yan

Department of Electrical, Computer and Software Engineering
The University of Auckland, Auckland, New Zealand

Abstract

We present the design of a cruise control system using a model-based approach. The specification of the cruise control system is demonstrated by using the system context diagram, data-flow decomposition diagram and a finite state machine diagram. This design will be implemented using the synchronous programming language Esterel [1]. The result will be an executable response program that meets the given requirements of the cruise control system.

1. Introduction

This design is based on the cruise control system introduced in Compsys-723 Assignment 2. The purpose of the cruise control system is to maintain the car at a constant speed without pressing the accelerator or brake pedals. The control system can work normally within the specified speed limit (**SpeedMin** to **SpeedMax**). In this report, we will implement the design of the cruise controller by first creating a specification and then mapping that specification in Esterel.

This design involved several features of the Esterel language. We illustrate the use of concurrency and synchronous synchronization using signals, the use of data handling functions in C and the use of multiple modules. The organization of this report is as follows. In Section 2, the specification of the cruise controller is presented using multiple diagrams. In Section 3, the detailed implementation in Esterel is introduced. Finally, in Section 4 we make some concluding remarks.

2. Specification

The overall input-output behavior of the cruise controller is described using the context diagram as shown in Figure 1. The cruise control system reads the driver's inputs entered through the cruise interface, pedals, and sensors of the car. The cruise interface includes the following input:

- On (pure): Enable the cruise control.
- Off (pure): Disable the cruise control.
- Resume (pure): Resume the cruise control after braking.
- Set (pure): Set the current speed as the new cruise speed.
- QuickDecel (pure): Decrease the cruise speed.
- QuickAccel (pure): Increase the cruise speed.

The pedals and sensors include:

- Accel (float): Accelerator pedal sensor.
- Brake (float): Brake pedal sensor.
- Speed (float): Car speed sensor.

The cruise control system has the following outputs:

- CruiseSpeed (float): Cruise speed value.
- ThrottleCmd (float): Throttle command.
- CruiseState (enumeration): State of the cruise control. It can be either OFF, ON, STDBY, or DISABLE.

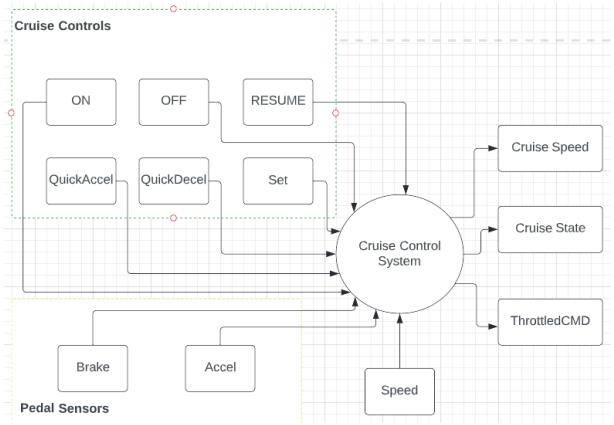

Figure 1: Context Diagram of the Cruise Control System

When the car starts for the first time, the cruise control system is in the OFF state, and the user can manually turn ON/OFF the cruise control or adjust the cruise speed through the buttons in the cruise control interface. The increment of adjustment is (**SpeedInc**: 2.5km /h). The cruise control works if the car's speed is within The speed limit (**SpeedMin** = 30km/h, **SpeedMax** = 150 km/h). The pedals and the speed of the car can both affect the state of the cruise control system.

The top-level context diagram can be further refined into the next level diagrams as shown in Figures 2 to 4. If the speed of the car is within the speed limit, the signal **SpeedBetweenLimit** will be sent. If the driver presses the Accel/Brake pedal over the threshold which is **PedalsMin** > 3.0 percent, the signals

AccelPressed/BrakePressed will be sent. These signals will be used to determine the state and speed of the cruise control system and the car's behaviours.

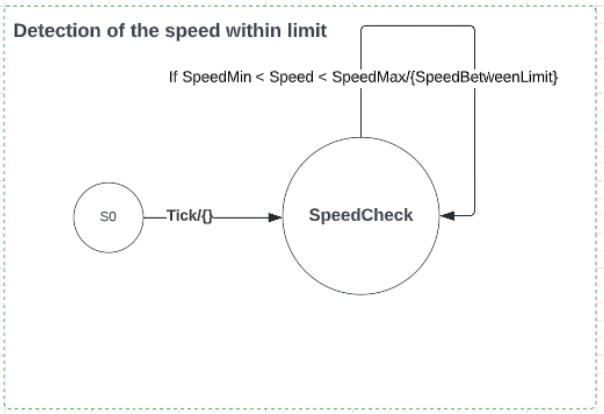

Figure 2: Detection of the Pressed Pedals

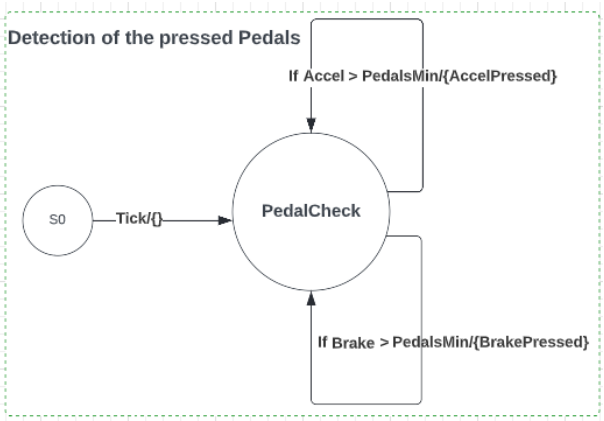

Figure 3: Detection of the Speed within Limit

The design has three main components which are Cruise State Management Module, Cruise Speed Management Module and Car Driving Control Module. The Cruise State Management Module determines the state of the cruise system and generates the output **CruiseState**. The Cruise Speed Management generates the output **CruiseSpeed** which is calculated from the input and displayed to the user to show the speed maintained by the cruise system. Both the **CruiseState** and **CruiseSpeed** will be used in the Car Driving Control Module to calculate the **ThrottleCmd**. It uses the **regulateThrottle** which is an external function with the parameters **Kp** = 8.113 and **Ki** = 0.5 to compute the **ThrottleCmd** via the integration algorithm. The value of **ThrottleCmd** saturates to a user-defined value. In this design, the **ThrottleSatMax** = 45 percent is used to ensure passenger comfort. Both the Cruise Speed Management Module and The Car Driving Control Module are designed according to the defined properties in the assignment handout.

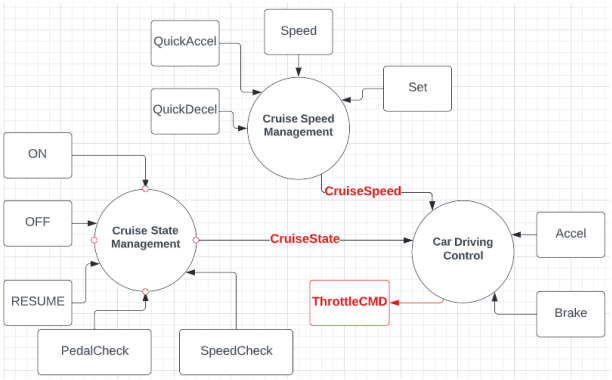

Figure 4: Overview of function decomposition

The Cruise State Management Module is a pure control flow function described as FSM in Figure 5. The module is further refined into four distinct states: OFF, ON, STDBY and DISABLE. According to the behavioral requirements in the assignment handout, the states of the cruise control system are switched as shown below.

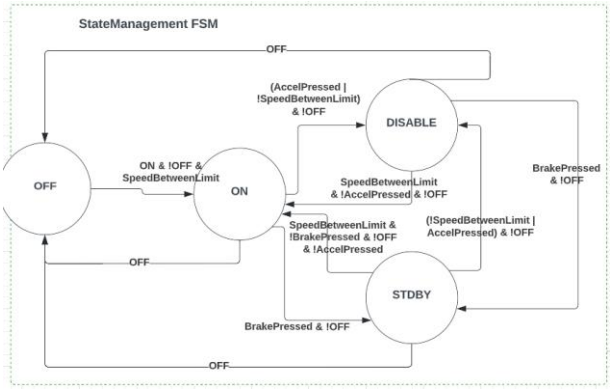

Figure 5: Cruise Control System State FSM

3. Design in Esterel

The Esterel design of the cruise control system directly follows the specification described in Section 2. Esterel program consists of one top module which is the Cruise Control System and five sub-modules within the top module and the five sub-modules are run in parallel. The detailed implementation is intended below.

CruiseControlSystem: The top-level module defines all the inputs & outputs in the interface and the internal signals which are **SpeedBetweenLimit, AccelPressed** and **BrakePressed**. The five sub-modules run in parallel inside the top module.

PedalCheckModule: This module checks the **Accel** & **Brake** are pressed or not. It compares the value of

Accel & **Brake** with the **PedalMin**, if the value is larger than the **PedalMin** then the signal **AccelPressed/BrakePressed** will be emitted.

CheckSpeedLimitModule: This module checks the **Speed** is between **SpeedMin** and **SpeedMax** or not. It compares the value of **Speed** with the **SpeedMin & SpeedMax**, if the value is between the **SpeedMin** and **SpeedMax** then the signal **SpeedBetweenLimit** will be emitted.

CuriseStatementManagementModule: This module determines the state of the cruise control system. The module takes the inputs of **ON, OFF, Resume, AccelPressed, BrakePressed** and **SpeedBetweenLimit.** According to the FSM diagram in Figure 5, the module generates an integer output **CruiseState.** The meaning of the value is 1 = **OFF**, 2 = **ON**, 3 = **STDBY** and 4 = **DISABLE**.

CarDrivingControlModule: This module regulates the throttle. The module takes the inputs of **CruiseState, CruiseSpeed, Accel** and **Speed**. This module uses the **regulateThrottle** function and the input boolean value is **True** when the **CruiseState** changes from **OFF** to **ON**, Otherwise the boolean value is **False** when the current state is **ON**. Given **Kp** and **Ki** values, the function generates the output **ThrottleCmd** using the integral algorithm.

CruiseSpeedManagementModule: This module manages the speed of the cruise control system. The module takes the inputs of **CuriseState**, **QuickAccel, QuickDecel**, **Set** and **Speed**. When the cruise system is **ON**, the module sets the output **CruiseSpeed** according to the **Cruisestate**. **CruiseSpeed** is increased or decreased by a predetermined value **SpeedInc.** If the input **Speed** is outside the range determined by the **SpeedMin** and **SpeedMax**, the speed is set to **SpeedMin** or **SpeedMax** respectively.

4. Conclusions

The simple cruise control system is designed according to assignment requirements. First, we analyze the requirements and transfer them into specifications. We produce the context diagram, functional diagram, and FSM. Finally, we implemented the design in Esterel V6. For further improvement, the About statement could be added to the Pedals detection module. This will automatically reset the brake and throttle to zero after each tick.

5. References

[1] G. Berry. Programming and verifying an elevator in esterel v7. Technical report, Esterel Technologies, 2004.