Cruise Controller

Feneel Sanghavi and Utsav Trivedi (fsan110 and utri092)
Department of Computer, Electrical and Software Engineering
The University of Auckland

I. Abstract

This report gives an insight into the functionality, design overview and development process for a Cruise Controller. The concepts of data-flow, control-flow and decomposition are used to realise the complete system.The system utilises Esterel V6's features to implement hierarchical finite state machines working in synchronous concurrency. Automation testing for verifying functional correctness was explored.

II. Introduction

A Cruise Controller is a component for Automobiles. The following solution implements a Control System for a Cruise Controller using provided specifications. The system controls the Car Throttle to ensure that the vehicle maintains a constant speed, known as Cruising Speed. There are parameters which limit the speed to a certain range. The system enables the user to alter the Cruising Speed. It is to be noted that the Cruising Functionality can be turned off to manually control vehicle speed via the Accelerator Pedal. Cruise Mode is disabled or put on standby automatically when the Accelerator or Brake Pedals are pressed. Esterel is used to implement the solution in order to utilise Synchronous Concurrency, along with the surety of a Deterministic Program.

III. Development Process

A. Analyse Specifications

The provided specifications were analysed, with rough diagrams used for brainstorming. The assumptions made were listed, along with documentation of any clarifications required by the Client, in this case, the Teaching Assistants. The requirements were, therefore refined through consultation with the Clients during lab sessions.This ensured a resulting solution consistent with the desired system characteristics. A system context diagram was used and followed several steps of refinement to realise the program functionality.

B. Implementation

Program Functionality was split between three modules. These are Next State, Driver Control and Cruise Speed Manager. Port mapping of input/output signals in respect to the environment and between modules was drawn. The created diagrams were used to determine the implementation of the functionality within the modules.

The coding in Esterel was done using the provided lecture slides and the Esterel Primer for Version 6.0. The Esterel Manual was used to implement a simulator from the Linux command line for automated testing.

C. Testing

A simulation program created with Python was used to test the solution. A range of input/output Combinations consisting of provided test cases and new test cases were used. Outputs were used to validate the program behaviour was functionally correct and that it met the timing requirements. The Python scripts allowed the testing process to be automated, running all test cases every time changes were made to implementation.

IV. Specifications

The figure shown describes the overall interface of the System.

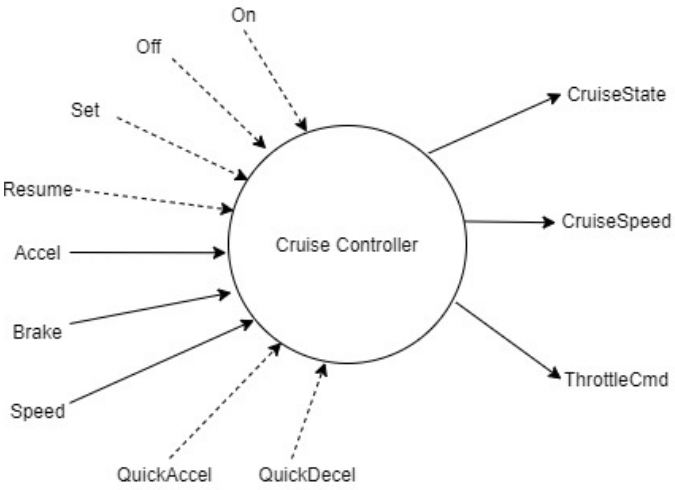

Fig. 1: System Context Diagram

A. Inputs

1) Pure Signals:

a) On: Turn the cruise control ON
b) Off: Turn the cruise control OFF
c) Set: Set Current Vehicle to Cruise Speed (When cruise control is enabled
d) Resume: Go from STANDBY to DISABLE or ON
e) QuickAccel: Increase Cruise Speed by 2.5km/hr
f) QuickDecel: Decrease Cruise Speed by 2.5km/hr

2) Valued Signals:

a) Speed: Current Vehicle Speed(km/hr)
b) Brake Pedal: Brake Pedal Positon in %
c) Accelerator Pedal: Accelerator Pedal Position in %

B. Outputs

1) Valued Signals:

a) Cruise State(Enumeration): Current State of the Cruise Controller
b) ThrottleCmd (float) : Throttle command (%)
c) Cruise Speed(float): Speed(km/hr) at which the Cruise Controller aims to keep the vehicle if enabled.

V. Design Discussion

A. Overall Design

The context diagram in Figure 1 is further decomposed into control unit and datapath, shown in Fig.2.

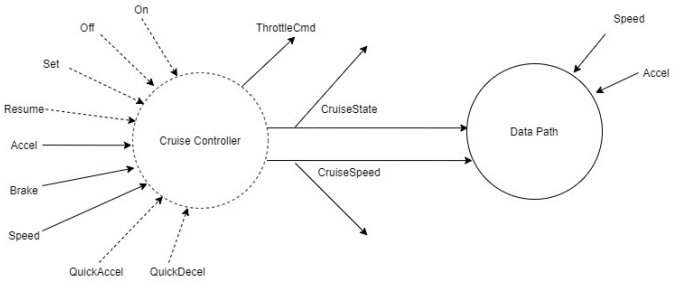

Fig. 2: First Level Refinement

Cruise contains three concurrent sub-modules Next State, Cruise Speed Manager and Driver Control. Fig.5(see appendix) shows the interface, the sub-modules in the system and dependencies between these inner modules. Cruise implements this design by mapping the input and output signals to the sub-modules, as well as between them. This mapping of signals to/from sub-modules is similar to Port Mapping in VHDL.

Next state uses system inputs to compute the transition that the Cruise Controller must take, which is then emitted to the User and fed to Cruise Speed Manager and Driver Control.

Cruise Speed Manager uses system inputs and Cruise State from the Next State Module to calculate the Cruise Speed of the vehicle. It emits the speed to the User and feeds it to Driver Control.

Driver Control uses system inputs, Cruise State and Cruise Speed from the other modules to calculate Throttle Cmd, which is emitted to the User. Refinement of this seen in Fig.3.

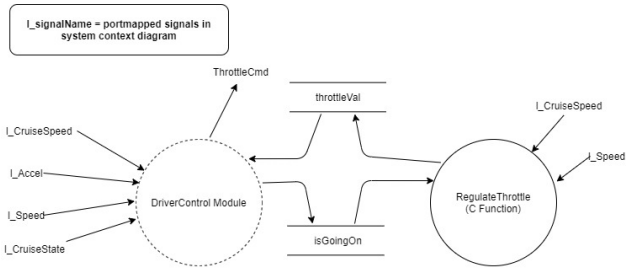

Fig. 3: Refinement of Driver Control Module

The sub-modules are further detailed in the following sections.

B. Module 1: Next State

This module represents the hierarchical finite state machine for the cruise controller. Initially, it sets the state of the controller to OFF(1) and transitions to the macro-state CRUISE_STATE_AUTOMATIC, when pure signal On is provided. While the cruise controller is enabled, it reacts to both pure and valued signals to transition between the different CRUISE_STATE_AUTOMATIC states or exit the macro-state. Esterel's strong abort feature was used to react to high priority input signals depicted in Fig.6(see appendix). It outputs the next CruiseState every tick. A short description of the states in Fig4(see appendix). is described below:-

- **OFF(1)**: This is the initial state the cruise controller is set to. Transitions to ON(2) immediately on receiving pure signal On.
- **CRUISE_STATE_AUTOMATIC**: This is a macro-state consisting of the inner three different states ON(2), DISABLE(3) and STANDBY(4). It represents the cruise controller when the cruise speed is automatically regulated. A visual representation is in Fig.6(appendix). It transitions immediately to OFF(1) on receiving pure signal Off. Instead of explicitly mentioning this state in Esterel, a hierarchy has been created using nested abort statements.
- **ON(2)**: This is the initial inner state inside the macro-state CRUISE_STATE_AUTOMATIC. It represents the cruise controller when the cruise speed is managed and vehicle speed is automatically regulated. A transition to DISABLE(3) occurs if the accelerator pedal is pressed and/or speed limit is crossed. This is denoted by isAccelPressed and isLegalSpeed respectively. A transition to STANDBY(4) occurs if the brake is detected. This is denoted by isBrakePressed. State actions will be skipped due to the strong abort.
- **DISABLE(3)**: This state represents the cruise controller when the cruise speed is managed and a speed limit is crossed or accelerator pedal is pressed or both. This is denoted by isSpeedLegal and isAccelPressed respectively. A transition to ON(2) occurs if the conditions mentioned are not violated. A transition to STANDBY(4) occurs if the brake is detected. This is denoted by isBrakePressed. State actions will be skipped due to the strong abort.
- **STANDBY(4)**: This state represents the cruise controller when the cruise speed is managed and a brake pedal is pressed. Transitions to the other inner states occur only if the pure signal Resume is present and the brake is not pressed. It goes to ON(2) if legal speed limits are obeyed and the accelerator is not pressed else it goes to DISABLE(3) for violation of the mentioned conditions.

C. Module 2: Cruise Speed Manager

This module manages the cruise speed of the vehicle if the state of the cruise controller is not off. It sets the cruise speed to the latest vehicle speed or speed limit boundaries whenever the macro state CRUISE_STATE_AUTOMATIC is entered in Fig.6. or pure signal Set is present. It changes

the cruise speed by a fixed amount when pure signal inputs QuickAccel/QuickDecel are present and outputs the latest CruiseSpeed on every update of the CruiseState.

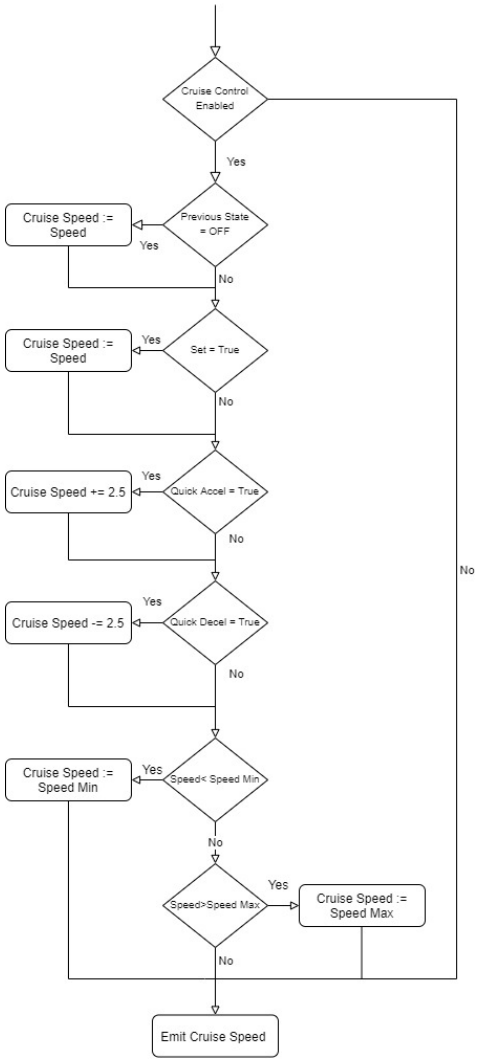

Fig. 4: Cruise Speed Manager Module control Flow

D. Module 3: Driver Control

This module is responsible for regulating the throttle of the vehicle. When the cruise controller is in OFF(1) state, the accelerator pedal controls the speed of the car else, it is automatically regulated in ON(2) state using algorithms in C functions emulating a PI controller that consists of proportional(Kp) and integral(Ki) factors. The regulation is protected against the overshoot of its integral action by being reset each time the macro-state CRUISE_STATE_AUTOMATIC is entered and frozen when the throttle is saturated for passenger comfort. It outputs the ThrottleCmd on every update of CruiseState. The throttleCmd is a non-negative output satured at an upper limit of ThrottleSatMax provided in the specifications.

VI. Testing and Validation

Test Cases are developed from the Functional Specifications. The vectors.in and vectors.out files are used for defining the test cases.

A Python script is developed to create a command-line simulator for Esterel (see Esterel Manual in installation_directory/doc/ for details) and run the test cases automatically. The success/failure of test cases is printed on the console. If a test case fails, the result is printed to console, and the testing program terminates at that point.

This Automated Testing Approach is useful as the program can be fully tested each time the functionality is changed.

Because Esterel is a deterministic language, for every single combination of inputs to the program, there is a single output combination.

VII. Conclusions

In this report, we present a cruise controller implementation where we have a top-level module running sub-modules in synchronous concurrency with dedicated responsibilities. The final design uses a hierarchical structure for the control unit and C functions in the data-path. Some key features of Esterel like aborts and local signals were used to ensure a causal and functionally correct program according to the specifications. This was tested using a Python script which invokes the command line simulator.

VIII. Acknowledgements

We thank Dr Avinash Malik for the lectures on Esterel, Dr Nathan Allen and Dr Hammond Pearce for their assistance during lab sessions.

IX. Time Taken

Time taken on planning and design development was **15 hours**, automation testing development was **10 hours** and report was **5 hours**.

X. Contributions

Both members had an equal part in understanding this assignment and Esterel. The pair-programming approach was applied where one person implements while the other supervises. The report work was shared equally.

APPENDIX

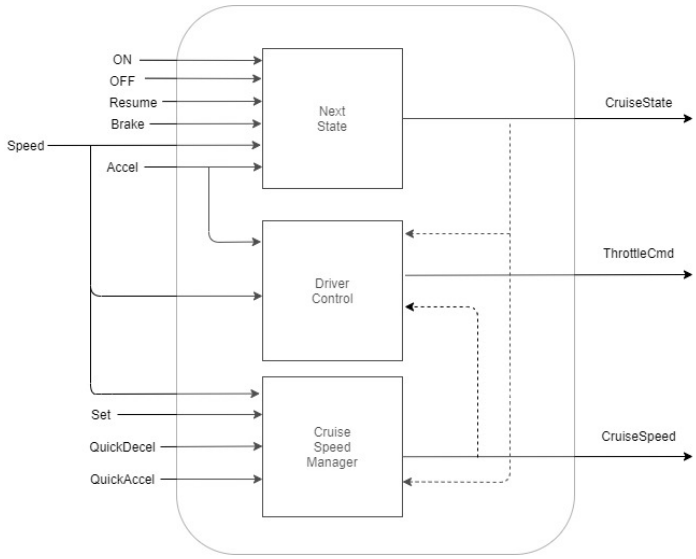

Fig. 5: Inner Modules

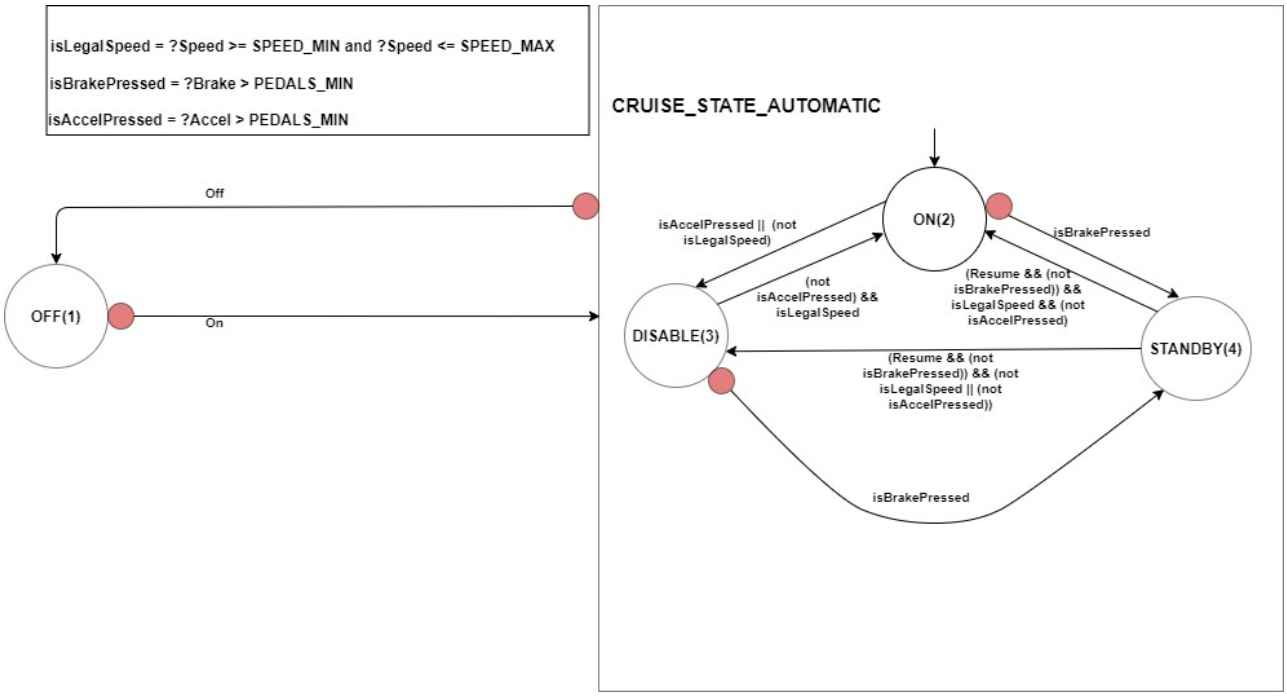

Fig. 6: State Chart

```
cruise> !trace all;
--- State: 0
--- Halts: 0
--- Awaited: On Off Resume Set QuickAccel QuickDecel Accel Brake Speed
cruise> Accel(0.0) Brake(0.0) Speed(0.0);
--- Output: CruiseSpeed(0.000000) ThrottleCmd(0.000000) CruiseState(1)
--- Local:
--- Trap:
--- Source Variables:
V15 = false  (source variable cruise.NextState.isLegalSpeed)
V16 = false  (source variable cruise.NextState.isAccelPressed)
V17 = 1  (source variable cruise.NextState.state)
V19 = false  (source variable cruise.DriverControl.enRegulate)
V20 = false  (source variable cruise.DriverControl.enThrottle)
V21 = 0  (source variable cruise.DriverControl.isGoingOn)
V22 = 0.000000  (source variable cruise.DriverControl.throttleVal)
V23 = false  (source variable cruise.CruiseSpeedManager.enManage)
V24 = 0.000000  (source variable cruise.CruiseSpeedManager.newCruiseSpeed)
--- Signal Variables:
V6 = 0.000000  (value of signal Accel)
V8 = 0.000000  (value of signal Brake)
V10 = 0.000000  (value of signal Speed)
V12 = 0.000000  (value of signal CruiseSpeed)
V13 = 0.000000  (value of signal ThrottleCmd)
V14 = 1  (value of signal CruiseState)
--- Counters:
--- State: 0
--- Halts: 3
halt 1: line: 148, column: 9 of file: "cruise.strl" (NextState#1)
halt 2: line: 87, column: 9 of file: "cruise.strl" (DriverControl#2)
halt 3: line: 236, column: 9 of file: "cruise.strl" (CruiseSpeedManager#3)
--- Awaited: On Off Resume Set QuickAccel QuickDecel Accel Brake Speed
```

Fig. 7: CLI Simulator

```
Press 1 for Interactive, 2 for All Test: 2
2
 Passed! Test #1
 Outputs: {'CruiseSpeed': '0.000000', 'ThrottleCmd': '0.000000', 'CruiseState': '1'}

 Passed! Test #2
 Outputs: {'CruiseSpeed': '0.000000', 'ThrottleCmd': '0.000000', 'CruiseState': '1'}

 Passed! Test #3
 Outputs: {'CruiseSpeed': '0.000000', 'ThrottleCmd': '51.234001', 'CruiseState': '1'}

 Passed! Test #4
 Outputs: {'CruiseSpeed': '0.000000', 'ThrottleCmd': '51.234001', 'CruiseState': '1'}

 Passed! Test #5
 Outputs: {'CruiseSpeed': '0.000000', 'ThrottleCmd': '51.234001', 'CruiseState': '1'}

 Passed! Test #6
 Outputs: {'CruiseSpeed': '0.000000', 'ThrottleCmd': '51.234001', 'CruiseState': '1'}

 Passed! Test #7
 Outputs: {'CruiseSpeed': '0.000000', 'ThrottleCmd': '51.234001', 'CruiseState': '1'}

 Passed! Test #8
 Outputs: {'CruiseSpeed': '0.000000', 'ThrottleCmd': '89.686996', 'CruiseState': '1'}

 Passed! Test #9
 Outputs: {'CruiseSpeed': '0.000000', 'ThrottleCmd': '89.686996', 'CruiseState': '1'}

 Passed! Test #10
 Outputs: {'CruiseSpeed': '0.000000', 'ThrottleCmd': '89.686996', 'CruiseState': '1'}

 Passed! Test #11
 Outputs: {'CruiseSpeed': '0.000000', 'ThrottleCmd': '89.686996', 'CruiseState': '1'}

 Passed! Test #12
 Outputs: {'CruiseSpeed': '0.000000', 'ThrottleCmd': '89.686996', 'CruiseState': '1'}

 Passed! Test #13
 Outputs: {'CruiseSpeed': '0.000000', 'ThrottleCmd': '89.686996', 'CruiseState': '1'}

 Passed! Test #14
 Outputs: {'CruiseSpeed': '0.000000', 'ThrottleCmd': '89.686996', 'CruiseState': '1'}

 Passed! Test #15
 Outputs: {'CruiseSpeed': '0.000000', 'ThrottleCmd': '89.686996', 'CruiseState': '1'}
```

Fig. 8: Test Suite