CS723 - ASSIGNMENT 2 REPORT

Harrison Jae Warahi & Cassandra D'Souza

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

Abstract

This report covers the main elements of our design solution for assignment two of Computer Systems 723, including formal design specification diagrams. This assignment aims to create a functional specification for a cruise control system. We implemented this using the synchronous programming language Esterel. We produced an executable reactive program that met the given requirements for the cruise control system.

1. Introduction

This assignment is based on a cruise control system for a vehicle. Cruise control is a feature of modern-day vehicles. It provides drivers with an alternative way of controlling the speed of a car, other than using their foot to press the accelerator pedal. It uses sensors and an actuator that controls the throttle to regulate the speed of the vehicle.

To create this system, we used Esterel, a synchronous, system-level language for embedded systems. A benefit of using Esterel is that it is based on mathematical foundations, allowing for proper verification of programs. Due to its synchronous approach, there is no delay between the input and the corresponding output as the system executes infinitely fast. The program is reactive and deterministic, thus making testing and verification of our code possible.

2. Formal Specification Diagrams

Before we designed the system in Esterel, we created a context diagram and Finite State Machine (FSM) to assist with visualizing the inputs and outputs and the logic of the system.

2.1. Context Diagram

The context diagram in Figure 1 helped us understand the relation between inputs, outputs, and which area(s) of the cruise control system they are applied to. A cruise control system has three key features: a State Controller, Throttle Controller, and Speed Controller, with specific behavioural requirements.

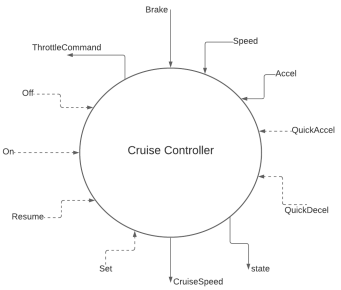


Figure 1: Context Diagram of Cruise Controller

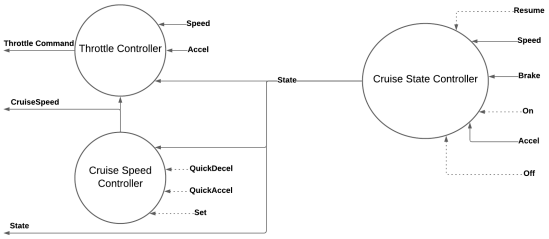


Figure 2: First Level Refinement.

The Cruise state Controller is an FSM that uses the external PURE inputs of the pedal sensors, ON, OFF, and RESUME, to determine the correct state of the CruiseController, which can be OFF, ON, DISABLE, STDBY. The state of the cruise control is used by both the Throttle and Cruise Speed Controller.

The Cruise speed controller uses the state information alongside the QuickAccel, QuickDecel, and SET inputs to determine the Cruise Speed of the system. Finally, the Throttle controller uses both the FSM state, pedal sensors, and the Cruise Speed to determine the throttle output of the system. Each of the separate modules has an output used by the entire

system, and each module relies on a result from the previous one (FSM → Cruise Speed → Throttle).

2.2. FSM Diagram

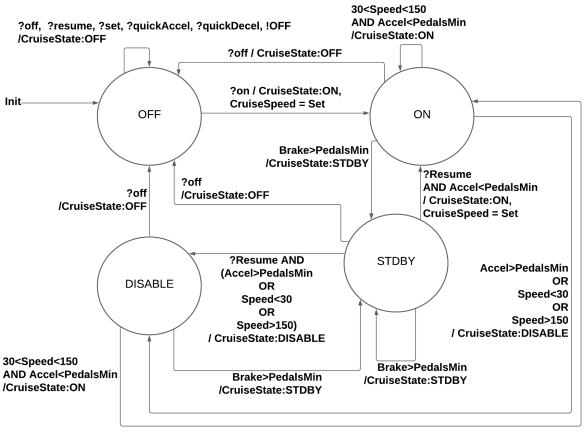


Figure 3: FSM of Cruise Control

The FSM helped us understand the transitions between the four states of the Cruise Controller; disabled (DISABLED), standby (STDBY), On (ON), and Off (OFF).

The FSM has specific requirements for reaching each state. The cruise control will start in an OFF state and remain in an off state until the ON button is pressed. The cruise control will return to the OFF state when the OFF input is received. The cruise control will enter into the ON state when the ON button is pressed. It will remain in the ON state until the speed of the car exceeds its limits, the accelerator or brake pedal is depressed beyond its threshold, or the OFF button is pressed. The cruise control will enter the DISABLE state when the accelerator pedal is depressed, the speed limit is exceeded, and the STDBY state when the brake pedal is depressed. The cruise control can return to the ON state from DISABLE when the pedals are not pressed and the speed is within its limits. The cruise control can return to the ON state from STDBY when the brake is not pressed and the RESUME button is pressed.

2.3. Refinement Levels

3. Design Decisions

The Cruise Controller is split into three submodules, a state controller, a cruise speed controller, and a throttle controller. These are executed parallel in one parent module.

We used a variety of signals such as pure signals, valued signals, and tick signals.

We decided to split the modules in their final layout based on the outputs required from the entire system. Dividing the system into components that each output one signal meant that when implementing the design, timing and signal flow only needed to be considered for one signal at a time.

A critical decision during the design process was the implementation of the state signal. To output the state signal across all the submodules and the external output, we implemented a loop that runs in parallel with each module, where the state output is sent from the FSM to an internal signal, which is emitted as an external signal. The internal signal from the FSM is forwarded to the other submodules, allowing the state to be read internally.

A challenge we encountered during the design of the project was implementing the submodules into separate .STRL files. To make maintenance of the code more manageable, we aimed to split the code, however compiling the project proved too challenging to achieve within the project's timeframe, so we opted to implement submodules within a single file instead.

4. Testing

To run and test our code with the Graphical User Interface (GUI), we used the following commands in eth terminal from the appropriate project directory:

- make CruiseControl.xes
- ./CruiseControl.xes

We manually went through the conditions supplied in the .in and compared our outputs to those in the .out files. We recorded the values of the results in the following tables and screenshots located in Appendix A. More rigorous testing is covered below in Appendix B, including all transitions illustrated in the FSM.

Appendix A

The results of the test case for the .in and .out files, referred to in section 4 of the report, are located here:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	On	Off	Resume	Set	QuickAccel	QuickDecel	Accel	Brake	Speed		CruiseSpeed	ThrottleCmd	CruiseState
2	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0	0	0		0.000000	0.000000	1
3	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0	0	0		0.000000	0.000000	1
4	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	51.234	0	0		0.000000	51.234001	1
5	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	51.234	0	2.0937		0.000000	51.234001	1
6	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	51.234	0	4.1862		0.000000	51.234001	1
7	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	51.234	0	6.2772		0.000000	51.234001	1
8	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	51.234	0	8.3661		0.000000	51.234001	1
9	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	89.687	0	10.453		0.000000	89.686996	1
10	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	89.687	0	13.051		0.000000	89.686996	1
11	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	89.687	0	15.644		0.000000	89.686996	1
12	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	89.687	0	18.233		0.000000	89.686996	1
13	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	89.687	0	20.816		0.000000	89.686996	1
14	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	89.687	0	23.393		0.000000	89.686996	1
15	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	89.687	0	25.962		0.000000	89.686996	1
16	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	89.687	0	28.524		0.000000	89.686996	1
17	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	89.687	0	31.078		0.000000	89.686996	1
18	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	89.687	0	33.623		0.000000	89.686996	1
19	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0	0	36.158		0.000000	0.000000	1
20	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	0	0	36.049		36.049000	0.000000	2
21	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0	0	35.94		36.049000	0.938827	2

Before Tick

After Tick

1.

2.

3.

4.

5.

6.

7.

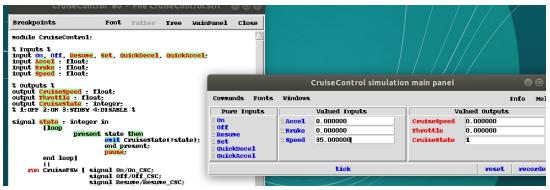
8.

Appendix B

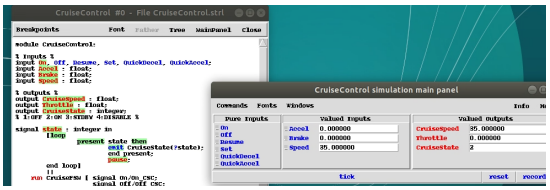
The results of additional testing referred to in section 4 of the report is located here:

Transitions going from OFF:

1. to OFF:

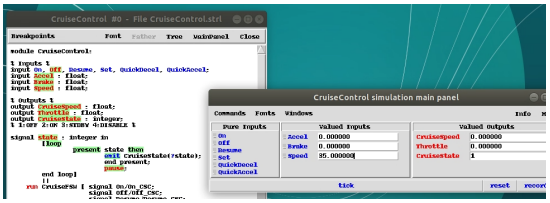


2. to ON:

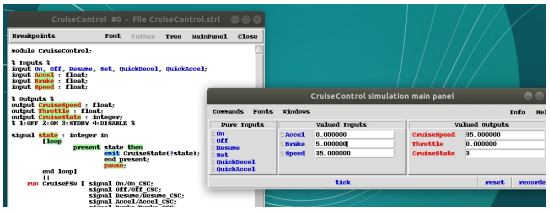


Transitions going from ON:

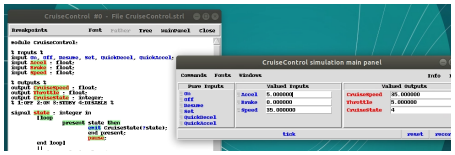
1. to OFF:



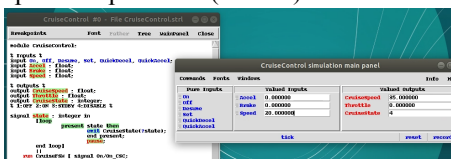
2. to STBY:



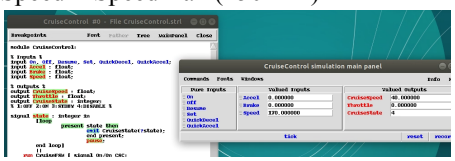
3. to DISABLE:
   a) Accel > PedalsMin



   b) Speed < SpeedMin (30kmh)

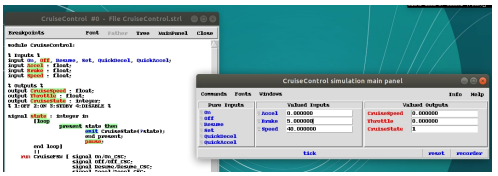


   c) Speed < SpeedMax (150kmh)

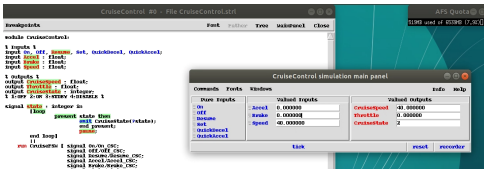


Transitions going from STDBY:

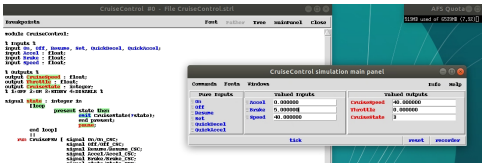
1. to OFF:



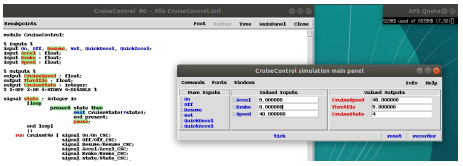
2. to ON:



3. to STBY:

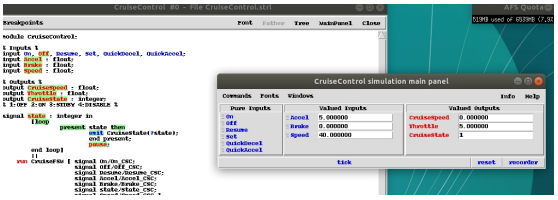


4. to DISABLE:

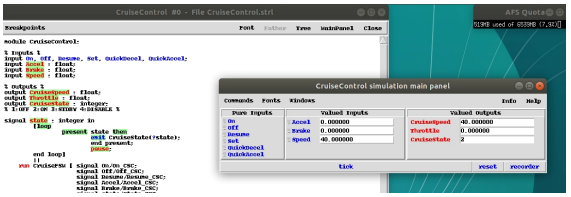


Transitions going from DISABLE:

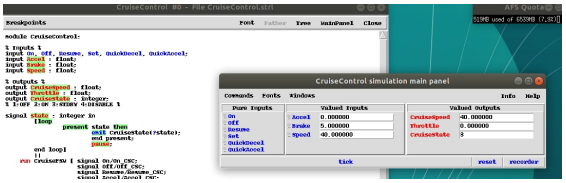
1. to OFF:



2. to ON:



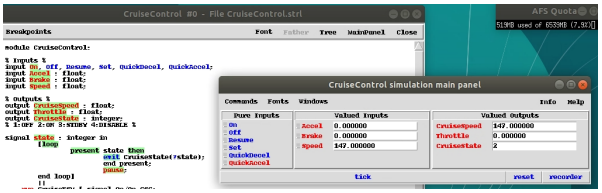
3. to STBY:



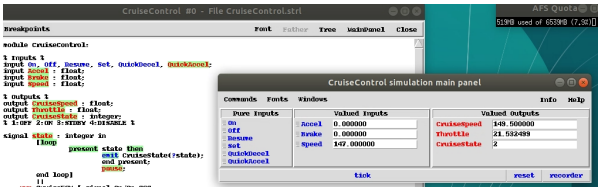
Testing QuickAccel:

Ensuring it increases

Before tick:

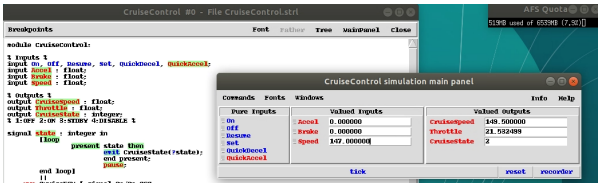


After tick:

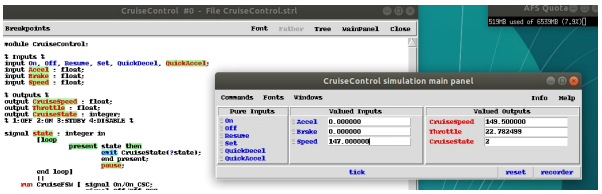


Ensuring it does not increase past 150kmph

Before tick:



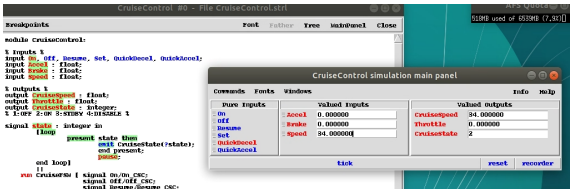
After tick:



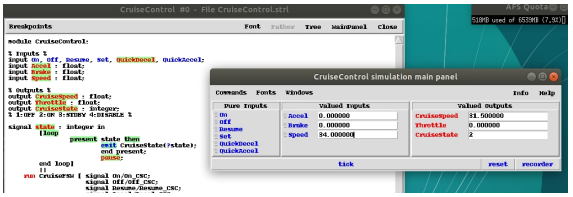
Testing QuickDecel:

Ensuring it Decreases

Before tick:

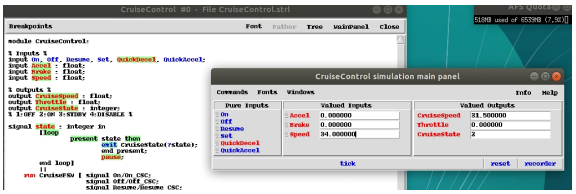


After tick:



Ensuring it does not Decrease past 30kmph

Before tick:



After tick: