Esterel Based Cruise Control System: Design, Testing, and Implementation

I. Introduction

The Esterel based Cruise Control system is an advanced real-time reactive system, developed as part of the coursework in the Master's Degree program. The aim of this system is to control a car's speed and maintain it at a constant level specified by the user. Esterel, a synchronous programming language widely used in reactive system development, forms the backbone of this system.

II. System Description

The system can be envisioned as multiple collaborating modules, each responsible for a specific operation. The whole system is encapsulated within the `CruiseControl` module. This module receives inputs from various external entities such as the car's accelerator, brake, current speed, and several command inputs from the user.

Module Breakdown

1. SpeedLimitCheck: This module checks if the current speed is within the defined speed limits. If it is within `SpeedMin` and `SpeedMax`, it emits `WithinSpeedLimit` signal.

2. BrakeCheck: Checks whether the brake is pressed. If the brake pedal's pressure exceeds `PedalMin`, the `BrakePressed` signal is emitted.

3. AccelCheck: Similar to `BrakeCheck`, this module checks if the accelerator is pressed, emitting `AccelPressed` when the accelerator pressure is above `PedalMin`.

4. CruiseStateController: It is the central controller that adjusts the state of the cruise control based on the inputs. The controller has four states: OFF (1), ON (2), STANDBY (3), and DISABLE (4). The transitions between states are governed by the conditions stated in the Esterel `present` clauses.

5. CruiseSpeedController: This module is responsible for maintaining and adjusting the cruise speed. It receives the current state from `CruiseStateController` and performs various operations like setting speed, accelerating, and decelerating. It ensures the cruise speed stays within the minimum and maximum speed limits.

6. ThrottleController: It regulates the throttle output based on the `CruiseState`, `CruiseSpeed`, and the current speed. If the Cruise Control is ON, it calls the `regulateThrottle` function to adjust the throttle command. If Cruise Control is not ON, it directly outputs the accelerator input as the throttle command.

These modules work synchronously in every clock cycle. They interact through internal signals, interpreting inputs and generating outputs to control the car's speed efficiently.

The Cruise Control system transitions between states based on a variety of conditions such as the speed being within the defined limits, whether the accelerator or brake is pressed, and the inputs from the driver like turning the system ON/OFF or setting/resuming speed.

For instance, the system turns ON (State 2) from OFF (State 1) when the ON signal is present, the OFF signal is not present, and the speed is within limits. Conversely, the system moves to the DISABLE state (State 4) from the ON state if the accelerator is pressed or the speed goes out of limits, provided the system isn't being turned OFF or the brake isn't pressed.

The speed controller and throttle controller then take action based on the current state of the system. If the system is ON, the speed controller adjusts the cruise speed based on whether the Set, QuickAccel, or QuickDecel signals are present. Concurrently, the throttle controller calls the `regulateThrottle` function to adjust the throttle command to maintain the set cruise speed.

In essence, the Cruise Control system manifests a robust example of a state-driven real-time reactive system, effectively utilizing the Esterel language's synchronous programming paradigm to achieve desired functionality. It underscores the critical role of state management, inter-module communication, and precise control flow in building complex reactive systems.

Note that Esterel programs can be formally verified due to their synchronous nature and well-defined semantics. This property would be highly beneficial for safety-critical applications like cruise control systems, where system correctness is of paramount importance.

III. Testing

Testing plays a crucial role in ensuring the correctness of real-time reactive systems. This Cruise Control system undergoes a rigorous testing process using a test suite developed in the C programming language.

The C test program `cruiseControl\_test.c` generates different scenarios, simulating the actions of a driver and the car's responses. These scenarios include engaging the cruise control, setting a speed, accelerating, braking, and disengaging the system. Each test case checks whether the outputs generated by the Cruise Control system match the expected outputs for the given inputs.

The reason for utilizing C for testing is two-fold:

1. Compatibility: Esterel gets compiled into C, enabling a seamless integration between the system and the test suite.

2. Flexibility: C provides extensive control over the system level operations, allowing to simulate a wide range of scenarios.

IV. System Compilation and Execution

The system is compiled and executed using the Esterel v5\_92 compiler. The Makefile script, an essential part of the build system, outlines the compilation and linking process. It includes instructions to compile the Esterel code into C, compile the C test suite, and link them together to create an executable.

The Makefile leverages GCC (GNU Compiler Collection) to compile the C code. Additionally, the `-lrt` and `-lpthread` flags enable real-time scheduling and multi-threading capabilities, enhancing the system's performance.

V. Conclusion

The Cruise Control system is a significant example of a real-time reactive system, showcasing the capabilities of Esterel for creating such systems. The system design, combined with rigorous testing, ensures its reliability and correctness.

The use of C for testing illustrates the interoperability between synchronous programming languages like Esterel and traditional imperative languages. Furthermore, the Makefile script underlines the importance of an effective build system in managing complex software projects.

In future work, extending the system to include more advanced features, such as adaptive cruise control and collision avoidance, can be explored. Additionally, implementing more rigorous testing methods like formal verification can be considered to ensure the utmost reliability of the system.