REAL TIME SIMULATION AND HARDWARE-IN-LOOP TESTING OF A HYBRID ELECTRIC VEHICLE CONTROL SYSTEM

A Thesis

Presented to

The Graduate Faculty of The University of Akron

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

Praveen Medisetti

May, 2007

REAL TIME SIMULATION AND HARDWARE-IN-LOOP TESTING
OF A HYBRID ELECTRIC VEHICLE CONTROL SYSTEM

Praveen Medisetti

Thesis

Approved

Co-Advisor
Dr. Iqbal Husain

Co-Advisor
Dr. Robert J. Veillette

Committee Member
Dr. Alexis De Abreu Garcia

Accepted

Department Chair
Dr. Alexis De Abreu Garcia

Dean of the College
Dr. George K. Haritos

Dean of the Graduate School
Dr. George R. Newkome

Date

ABSTRACT

This thesis explains various stages of the vehicle controller development, especially for a Hybrid Electric Vehicle (HEV), and documents the development of a platform for vehicle controller testing. Two stages of testing a vehicle controller, namely Software-in-Loop (SIL) simulation and Hardware-in-Loop (HIL) simulation, are explained in a stepwise manner for the series-parallel 2x2 HEV. The idea of using a common tool from the design stage to the prototyping stage is demonstrated.

The series-parallel 2x2 HEV is modeled using the Powertrain Systems Analysis Toolkit (PSAT) in Matlab/Simulink. A rule based vehicle control strategy is added to the existing control libraries in PSAT. The SIL testing of the HEV model is done by exercising it over various drive cycles. A HIL platform is built from the ground up using commercially available off-the-shelf computers and Input/Output cards. The offline model of the HEV is simulated on the HIL platform to start the vehicle controller testing process. The preliminary HEV model was used to demonstrate the capabilities of the HIL setup. The HIL simulation setup is scalable and allows the incorporation of additional computational nodes for distributed simulation of complex systems without a major change to the original setup. The HEV model is run in real time on two computation nodes and the differences between offline and online simulations are discussed. The HIL simulation platform is successfully built and can be used for testing and tuning the vehicle controller.

DEDICATION

Dedicated to my family and teachers.

ACKNOWLEDGEMENTS

I would like to thank the committee members Dr. Iqbal Husain, Dr. Robert J. Veillette and Dr. Alexis De Abreu Garcia for their guidance and support throughout my Master's program. I would like to thank Dr. Husain for giving me an opportunity to work on such a prestigious project. I would like to thank Dr.Veillette for providing a solid foundation in Controls Engineering and helping me in shaping my research work, thoughts and ideas into a good manuscript. I would like to thank Dr. De Abreu Garcia who has been there with me in difficult times and motivated me for working on this project. I also would like to thank the Faculty of Department of Electrical and Computer Engineering especially Dr. Tom Hartley who always entertained my questions and motivated me for doing the HIL simulation. Thanks to Gay Boden, Greg Lewis, and Erik Rinaldo for their help right from the day I stepped into the Graduate School.

I would like to thank my colleague in Power Research Laboratory Ahmed Khalil who has helped me out in looking at the prototyping tools right from the day I stepped into the research lab. I am thankful to Nayeem and Rakib for their quick and sharp comments which always had a touch of experience behind them. I also would like to thank Samuel, Christophe and Ludovic who were always there when I needed them.

I also would like to express my gratitude to my fellow Challenge X team members. They have been cooperative throughout the project. Don Whitmore who was not only a friend

but also mentor right from the projects inception in 2003. Chuck Van Horn, Nathan Picot, Angelina Bellino who put in so many man hours and kept up with my working schedule and never complaining about it. They spent many long hours in the lab waiting for the simulations. I also would like to thank my fellow grad students Omar and Ian who were always there to share the ups and downs.

I would like to thank sponsors from the industry Liz Callanan and Gayathri Seenumani of The MathWorks who provided the xPC TargetBox and software which is the backbone of the project. The project would have not been possible without the excellent help and support of Martin Belanger and Pierre François Allaire of OPAL-RT Technologies who I knew were always a phone call away. They were the best support team I ever worked with.

I would like to thank my Mom, Dad and Sister Siri for their support and patience. I owe them all my success.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 Introduction

Conventional vehicles employing an internal combustion engine (ICE) have gone through a lot of improvements over the past few decades. The ICE now has electronically controlled firing and other advanced technologies that have made it highly sophisticated. Modern cars also incorporate more power electronics driving the electromechanical systems, such as electric power steering, integrated starter/alternator to improve the performance as well as fuel economy. Still the relatively large ICEs needed to supply the desired peak power for acceleration tend to operate inefficiently at cruise speeds and during frequent start-stop operation, which is typical for urban driving.

The advancements in conventional vehicles provide some improvement in vehicle performance and fuel economy, but long term benefits in terms of reducing the dependency on oil and the emissions from ICE vehicles cannot be achieved without a more drastic change. An interim technology and solution for this is a hybrid electric vehicle (HEV). A graph showing the automotive development over the past few decades and a trend for future technology is given in Figure 1.1.

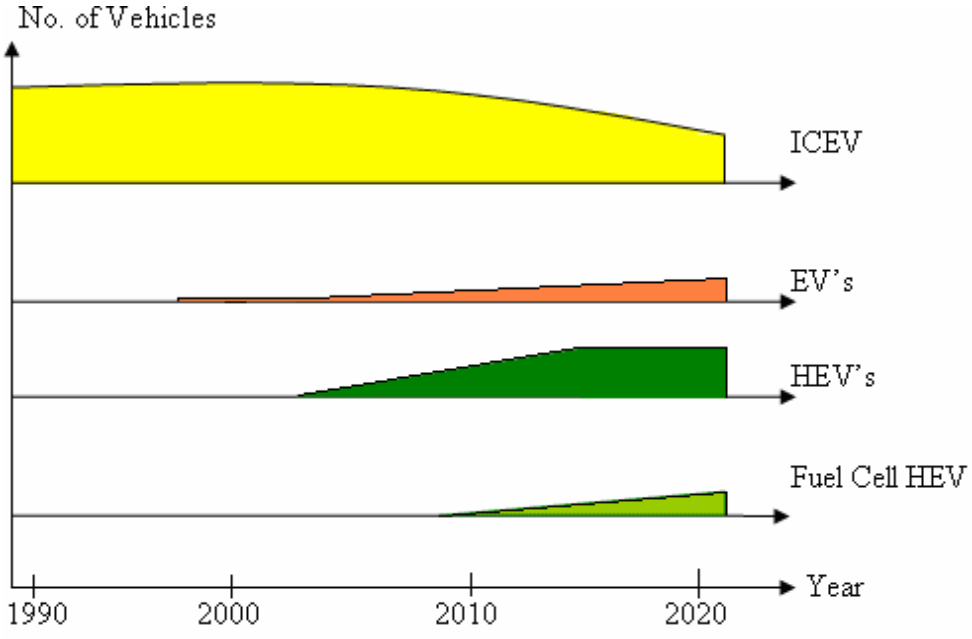


Figure 1.1: Automotive technologies, 1990-2020 [1].

The number of conventional vehicles produced is predicted to decline with the start of the 21st century. The HEVs are seen to be gaining popularity as the automotive community is realizing the importance of interim technology [1] [2]. With the current rate of advancement in technology, less oil-dependent future technologies like fuel cells and hydrogen powered vehicles are expected to be seen in the market by the early next decade.

1.2 Hybrid Electric Vehicle Basics

A HEV typically employs two propulsion systems: an ICE and an electric motor (EM) powered by an onboard energy storage device. The idea of a HEV dates back as early as 1905 but the technology did not flourish because of the unavailability of an efficient energy storage device. As small, reliable, cost effective energy storage becomes more

readily available, the idea of utilizing an EM as a propulsion device in a passenger vehicle is now becoming more practical.

A schematic representation of the power contribution from the ICE and the EM in various classes of vehicles (from conventional vehicle to EV) is given in Figure 1.2. The conventional vehicles employing an ICE and the EV employing stored electrical energy are on either side of the graph. The various classes of HEVs fall in the region between these two classes.

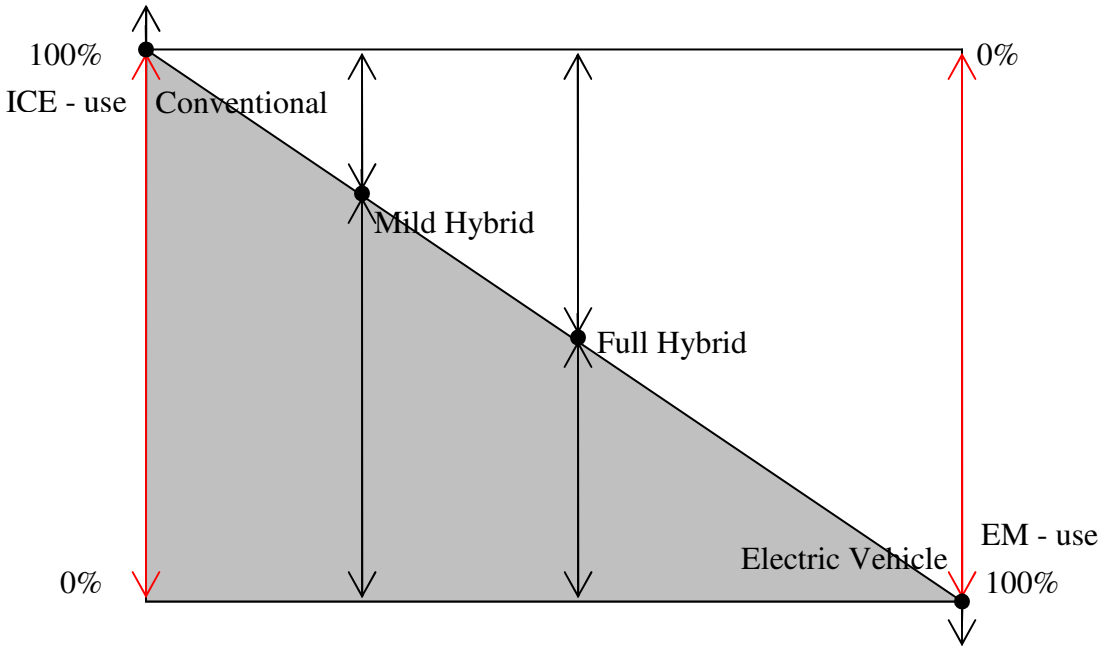


Figure 1.2: Sources of power in conventional vehicles and HEVs.

A HEV employs an ICE and an EM to power the wheels incorporating features from a conventional vehicle and an EV. The two power sources are brought into or out of operation based on the vehicle operating conditions and the driver requirements. The two power sources in a HEV additionally provide a degree of freedom in the operation of the vehicle. The ICE in a HEV is typically sized for operation at cruise speeds, and the EM to

meet the peak power demands. The EM is powered by an onboard Energy Storage System (ESS) such as a battery-pack or an ultracapacitor-pack or both. The EM is used independently or in conjunction with the ICE to meet the starting and acceleration requirements. This is possible because of the high starting torque capability of the EM. At cruise speeds, the ICE provides the required power while running at optimal or high efficiency operating points. This enables the use of a downsized ICE tuned for a narrow operating range which in turn improves the overall system efficiency. Additionally an EM can propel the vehicle at low speeds (urban driving) where the ICE is usually inefficient. The advantage of employing the EM is not only the additional power during acceleration, but also the capability of capturing the regenerative energy during vehicle braking when a percentage of the braking energy is recovered to recharge the onboard ESS. The ICE in a HEV can also be shut down instead of idling when the vehicle is stopped. Shutting down the ICE instead of idling reduces fuel consumption and emissions, especially in the urban stop and go driving situations. The gain in fuel economy due to vehicle hybridization can be attributed to the flexibility in choosing the ICE operating point and energy transmission path to gain energy efficiency, the reduction of ICE idling operation and the recovery of braking energy losses.

1.3 Research Motivation

The increased complexity of vehicle control and vital data communication in a HEV require a careful and thorough testing of the vehicle controller and its coordination with various subsystems. Testing of a vehicle controller in a HEV is essential before it is used to run a real vehicle based on both economic and safety grounds. A Hardware-in-Loop

(HIL) simulation platform enables an on-the-bench testing of a vehicle controller to be employed in a HEV. The testing can be also carried out on a production controller or on a prototype controller, which is typical in the early development stage. A HIL simulation setup provides the necessary bridge between offline simulations and real time implementation of a vehicle controller. Most of the research work done earlier demonstrates many offline studies done on a vehicle model [3]. The path between the offline simulations and real world implementation needs many factors to be considered. In a HIL simulation setup, offline simulations are modified to run in real time, and the hurdles in real time implementation of a HEV controller can be seen earlier in the design process. The importance and utility of this technology motivated the development a HIL setup to be used for HEV controller testing by the Challenge X team at The University of Akron. Challenge X is a HEV design competition with headline sponsorship from General Motors (GM), and the United States Department of Energy (USDOE). The primary objective was to reengineer a GM Chevrolet Equinox into a fuel efficient, environment friendly HEV while maintaining the performance of a stock vehicle.

1.4 Research Contribution

A HIL simulation platform was proposed and built as a part of the Challenge X competition in coordination with various sponsors. The thesis describes the real time simulation of a HEV on a HIL simulation platform. The software and hardware used for the HIL setup was sponsored by The MathWorks, OPAL-RT Technologies and Softing, in coordination with The Department of Electrical and Computer Engineering, The University of Akron.

In the HIL setup built, a model representing the vehicle runs on a Real Time Simulator (RTS) with a Control Area Network (CAN) [4] communication interface. CAN has evolved to become a standard mode of communication in modern vehicles, and the interface mimics the in-vehicle communications. A schematic representation of HIL simulation platform is given in Figure 1.3. The RTS may consist of a single node or many nodes based on the complexity of the vehicle model. The vehicle subsystems like ICE, EM and generator can run on different nodes communicating on a CAN network. The nodes are interfaced to an Electronic Control Unit (ECU). The ECU may be a prototype ECU or a production ECU.

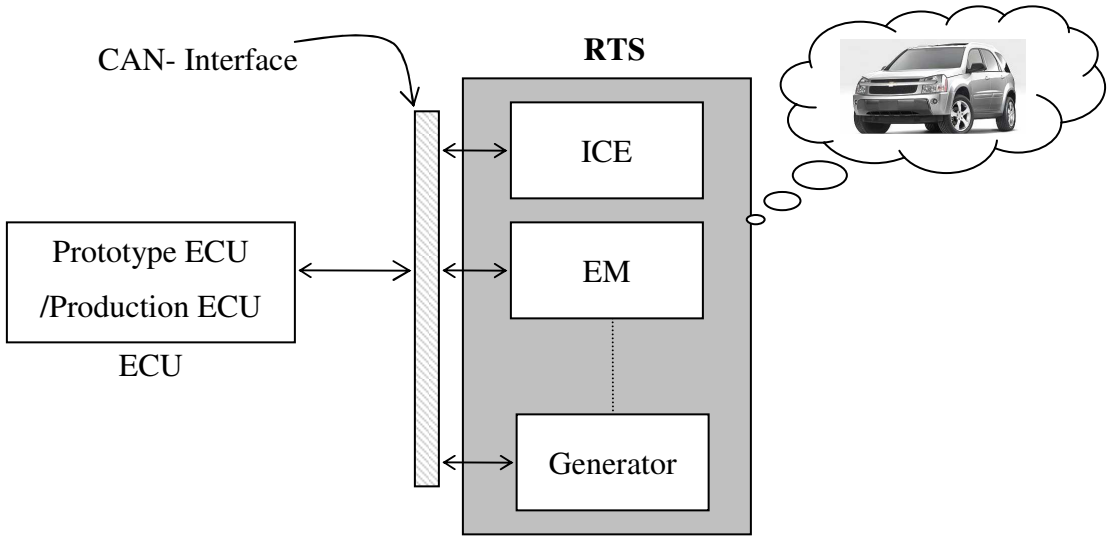


Figure 1.3: The concept of HIL simulation.

In the research work presented, the capability of using a single modeling and simulation tool (Matlab/Simulink) from the design stage to a prototyping stage is demonstrated. The Matlab/Simulink models of HEV obtained using Powertrain Systems Analysis Toolkit (PSAT) are presented. PSAT is a vehicle modeling and simulation software provided by the Argonne National Labs (ANL).

In a HEV, the operation of the propulsion system and other subsystems is controlled by a vehicle control strategy. A basic vehicle control strategy has been developed and added to the existing control libraries in PSAT for future development and testing. The model of a HEV controller is checked by running a Software-in-Loop (SIL) simulation to validate the initial sizing of the components, and also to ensure that the vehicle performance is satisfactory. The vehicle controller is further tested in real time on the HIL simulation setup and it is demonstrated that the HEV meets initial design requirements.

The HIL simulation setup built is scalable and can be easily upgraded depending on the requirements of the real time simulation. This is not like the earlier systems where it was customized for a particular application or a project. The real time code obtained from a vehicle model in Matlab/Simulink is run on a RedHawk Real Time Operating System (RTOS) provided by Concurrent Computer Corporation. The various tools for real time simulation of the vehicle model are presented and explained. Versatility of the HIL simulation setup is demonstrated by real time simulation of a vehicular subsystem (Electric Motor Drive). The intricacies and the changes in the offline model when implemented in real time are demonstrated. To demonstrate the scalability of the HIL setup, additional computation nodes were included and the setup was used to demonstrate distributed simulation of a vehicle and its subsystems. The HIL simulation setup can be used effectively in future years and tied up into the ongoing automotive research at The University of Akron as a turn key system.

1.5 Thesis Outline

The research work done is presented in the following four chapters. Chapter II provides the necessary background information about HEVs and briefly explains some of the HEV architectures. An overview of the development process for a HEV controller is presented. Additionally the integration of SIL simulation and HIL simulation into the development process of a vehicle controller is explained.

Chapter III explains the Software-in-Loop simulation of a HEV. The procedure for obtaining the Matlab/Simulink models of HEVs with several alternative architectures using Powertrain Systems Analysis Toolkit (PSAT) is presented.

Chapter IV describes the basics of real time simulation and delves into various hardware components used in the HIL setup. The real time simulation of the HEV model obtained in Chapter III is described. The various capabilities of the setup are demonstrated.

Chapter V provides the results and analysis based on the real time data logged from the RTS. The differences between the offline and online simulation are explained.

Chapter VI concludes the thesis and provides the suggested future work.

CHAPTER II

BACKGROUND

2.1 Introduction

This chapter briefly describes the various HEV architectures and introduces the series-parallel 2x2 HEV. The stages in the development process of a vehicle control system, Software-in-Loop simulation and Hardware-in-Loop simulation are explained. The advantages of using HIL simulation for vehicle controller testing are described.

2.2 HEV Architectures

There are several HEV architectures, each with its distinct design and operating characteristics. HEVs usually fall into one of the following categories based on the arrangement and operation of the propulsion systems in the vehicle.

1. Series HEV
2. Parallel HEV
3. Split HEV
4. Series-Parallel 2x2 HEV

2.2.1 Series Hybrid Architecture

Series HEV is the simplest of HEV architectures. Series hybrid architectures employ an ICE coupled to a generator, which supplies electric power to an onboard energy storage device such as a battery and to an EM that serves as the propulsion system.

The EM drives the wheels through a differential (DIFF). A schematic representation of a series HEV is given in Figure 2.1. For simplicity gear coupling associated between the ICE and generator (GEN) is not shown. By controlling the GEN operating point, the ICE is operated in its most efficient operating range. In a series HEV there is no mechanical transmission and the performance characteristics are governed by the EM and energy storage assembly. The generator is also used as a starter to start the ICE.

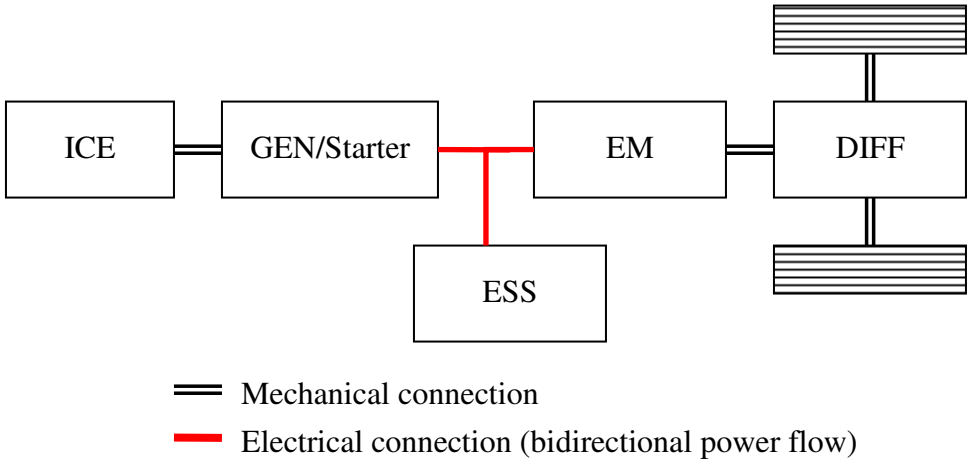


Figure 2.1: Series HEV architecture.

2.2.2 Parallel Hybrid Architecture

Parallel hybrid architectures use a single electric machine, which draws current from an energy storage device to provide mechanical assistance to the ICE. The parallel systems are best suited to low-power vehicles where the EM and the ICE are operated together to enhance the overall performance [1]. The simplest is the one in which the EM and ICE are on the same shaft. The Honda Insight and Honda Civic Hybrid are examples of production HEVs with similar parallel architectures.

There are three variations of a parallel HEV based on the position of the EM in the drive train. In parallel pre-transmission HEV the EM is coupled to the ICE shaft before the transmission and in a parallel post-transmission HEV, the EM is coupled to the ICE shaft after the transmission (TX). The layouts of the parallel pre-transmission HEV and post-transmission HEV are given in Figure 2.2 and Figure 2.3.

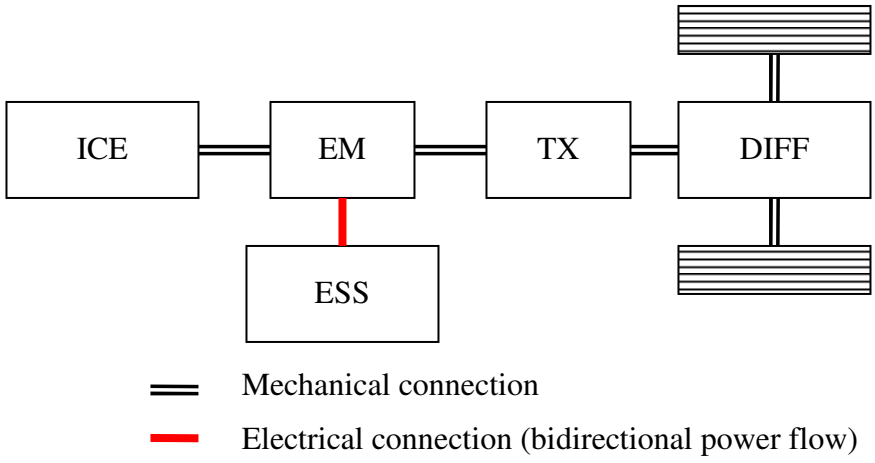


Figure 2.2: Parallel pre-transmission HEV architecture.

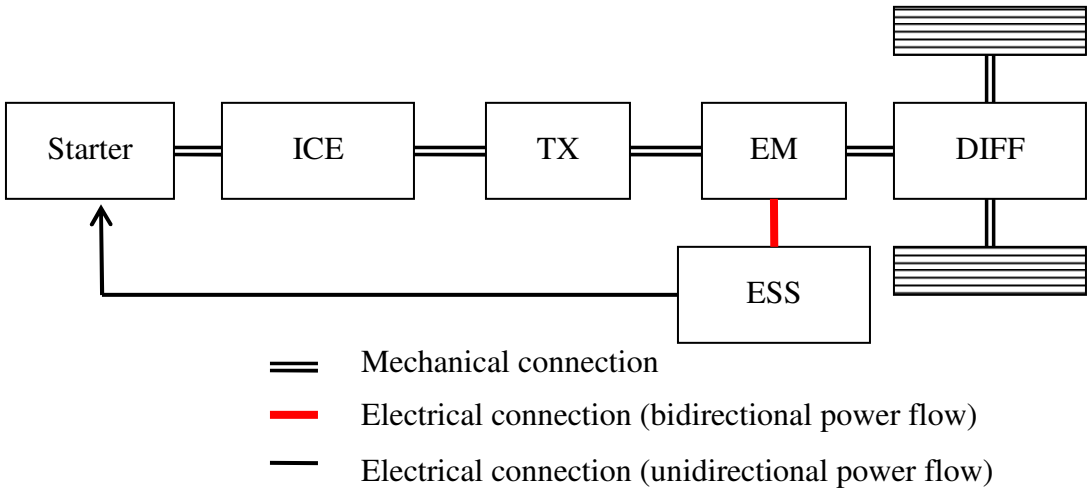


Figure 2.3: Parallel post-transmission HEV architecture.

A variant of the parallel HEV is the parallel 2x2 where the ICE shaft is separated from the EM. The ICE drives the front axle and the EM drives the rear axle. A small starter motor is mounted on the ICE shaft for starting purposes. A schematic is shown in Figure 2.4. The rear drive motor is capable of regenerating. This is also called parallel 'through the road' architecture.

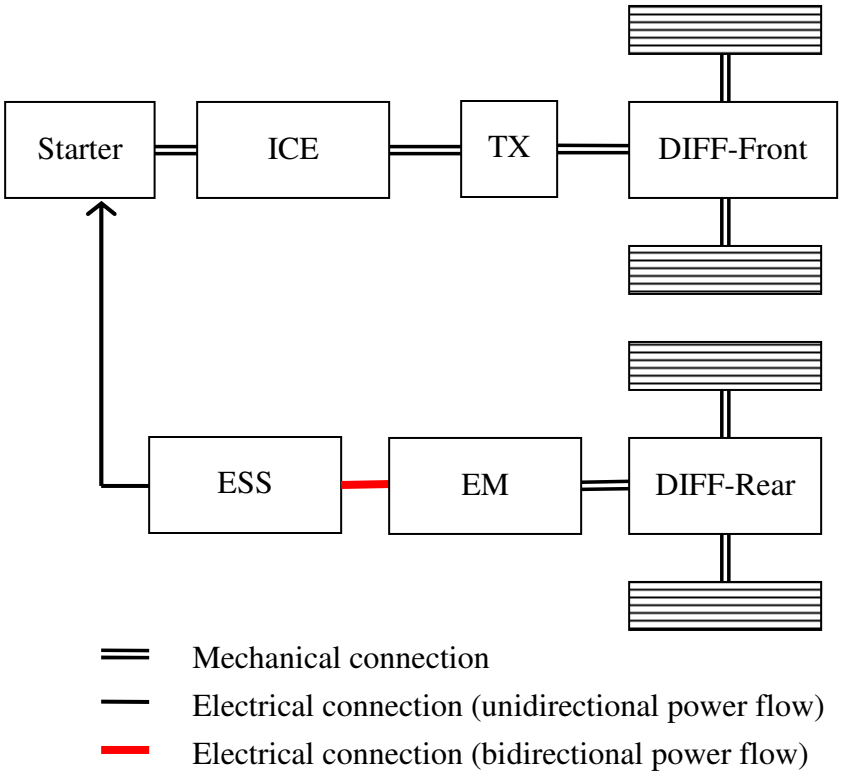


Figure 2.4: Parallel 2x2 HEV architecture.

2.2.3 Split Hybrid Architecture

The power split configuration is a blend of series and parallel designs. The split architecture uses a planetary gearset in which the ICE drives the planet carrier gear (C), the ring gear (R) is coupled to a Motor/Generator (MG) and to a differential, and the sun gear (S) is coupled to a Starter/Alternator (SA) [5]. The power split provides a

mechanical path from the ICE to the vehicle's wheels. In addition, the planetary gearset allows the effective gear ratio between the ICE and the output shaft to be continuously varied, similar to a Continuously Variable Transmission (CVT). This gives the flexibility of operating the ICE independent of vehicle speed. A schematic layout of a HEV with split architecture is given in Figure 2.5. The drawback of this architecture is the complex control involved.

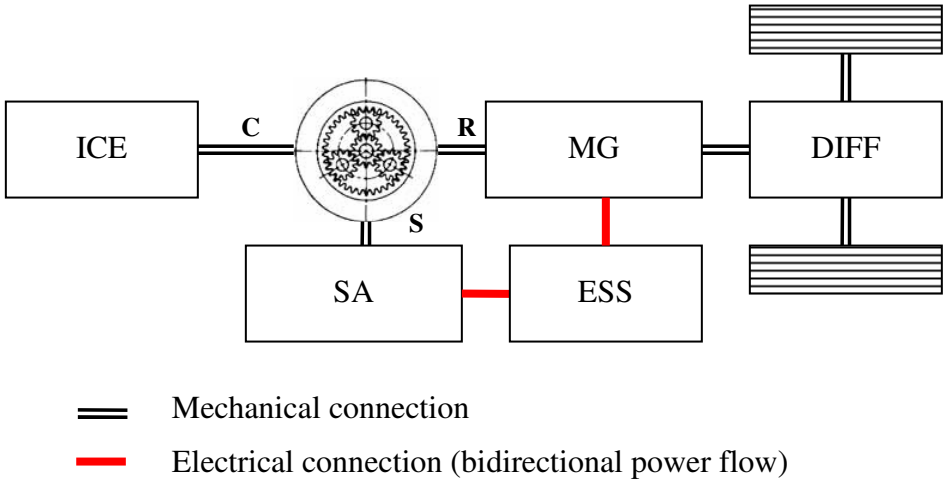


Figure 2.5: Split HEV architecture.

2.2.4 Series - Parallel 2x2 HEV Architecture

To blend the features of controlling the ICE operating point seen in a series architecture and excellent performance seen in a parallel architecture a unique architecture named series-parallel 2x2 was introduced [6]. The series-parallel 2x2 architecture gives an additional degree of freedom in operating the ICE. A correctly sized electric machine operating as a generator coupled to the ICE improves fuel economy ratings over the parallel 2x2 by operating the ICE in an optimal efficiency zone.

A schematic layout of the series-parallel 2x2 architecture used in the Akron Challenge X HEV is given in Figure 2.6. This architecture is relatively more complicated, involving an additional mechanical link when compared to a series hybrid HEV, and also an additional generator coupled to the ICE shaft compared to a parallel 2x2 HEV. More information on the various HEV architectures can be obtained in [1] [5] [6] [7].

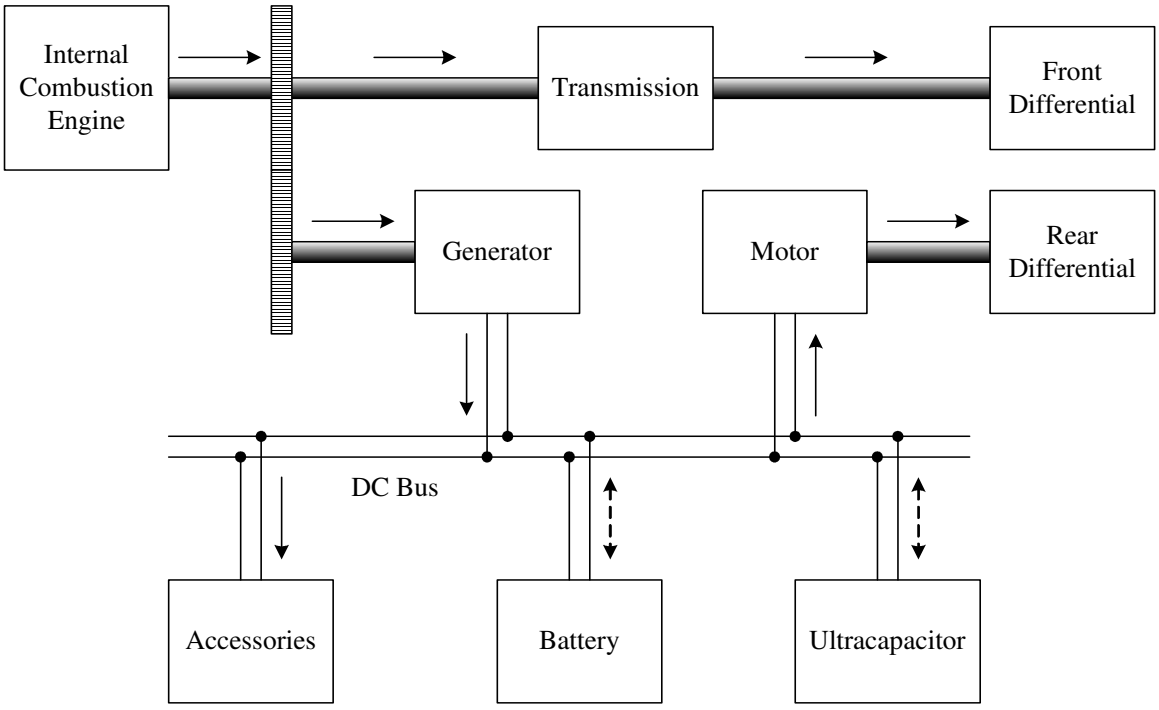


Figure 2.6: The University of Akron HEV architecture model.

The series-parallel 2x2 architecture will be used in the following chapters for Software-in-Loop and Hardware-in-Loop simulation studies.

2.3 Vehicle Controller: Development and Testing

Before it is installed in a vehicle, the vehicle controller is developed and tested using two classes of configurations namely SIL configurations and HIL configurations. This section

explains the various steps that can be carried out in each of these configurations. This also gives an overview of how the HIL setup built is tied into the development process [6] [8].

2.3.1 Software-in-Loop Configurations

In these configurations the controller and the plant are modeled and simulated using software at graphical level or code level, to evaluate the performance of a vehicle (a controller and chosen powertrain). The most straightforward type of Software-in-Loop simulation is depicted in Figure 2.7. In this setup, known as Model-in-Loop (MIL) Simulation, offline simulations of the HEV (plant and controller) are run on a single computational node. The simulations may be done on a non-real time platform like Windows XP (Microsoft) or GNU Linux (Free Software Foundation). Every subsystem in the vehicle model is programmed using the same software. The software coding may be accomplished using graphical tools like Simulink. Typically, a MIL simulation does not involve any real time code.

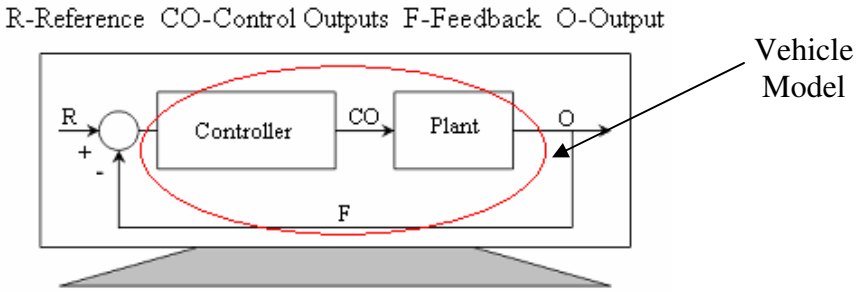


Figure 2.7: Model-in-Loop simulation.

An alternative type of SIL simulation that is a step closer to the controller implementation is shown in Figure 2.8. Here, the controller part of the model is replaced by controller

code generated from the model using auto code generation tools (like Real Time Workshop-RTW). The controller code generated can be put through tools available in the market to see the net memory requirement and any existing memory leaks. This gives the control engineer a good idea about the timing and memory issues the controller needs and prepares the stage for real time implementation. In this simulation the plant can still be at model level. The plant model and the controller code still run in an offline mode.

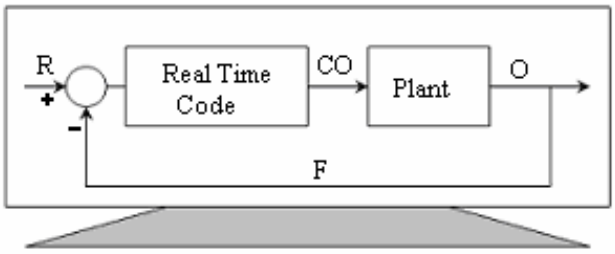


Figure 2.8: Software-in-Loop simulation with real time controller code.

There are various simulation tools available in the market for the SIL simulation of conventional vehicles and HEVs. Some of them are PSAT, Advisor (AVL-www.avl.com) and SimDriveline (The MathWorks). In the research work presented, PSAT was used for modeling and simulating a HEV because of its accuracy, availability and access to its developers. PSAT is a modeling and simulation tool that simulates vehicles, and predicts their fuel economy and performance in a realistic manner. It can simulate many predefined vehicle configurations (conventional, electric, fuel cell, series hybrid, parallel hybrid, and power split hybrid). PSAT is primarily a Graphical User Interface (GUI) built on top of Matlab/Simulink environment. The PSAT modeling environment is very modular enabling easy removal of default models of vehicle subsystems and replacing them with new models.

2.3.2 Hardware-in-Loop Configurations

In these configurations the real time code from the vehicle controller runs on a prototype or a production controller which is connected to a real time platform mimicking the plant. There are three types of simulation that can be done based on the number of real hardware components incorporated into the simulation. These are discussed in the rest of this section.

2.3.2 (a) Basic Hardware-in-Loop Simulation

In this simulation, the real time code runs on a prototype or production controller. The plant is run on a real time platform with a real time operating system (RTOS) capable of handling inputs and outputs (I/O) via appropriate I/O hardware. The plant simulated in real time mimics the vehicle. A schematic representation is given in Figure 2.9.

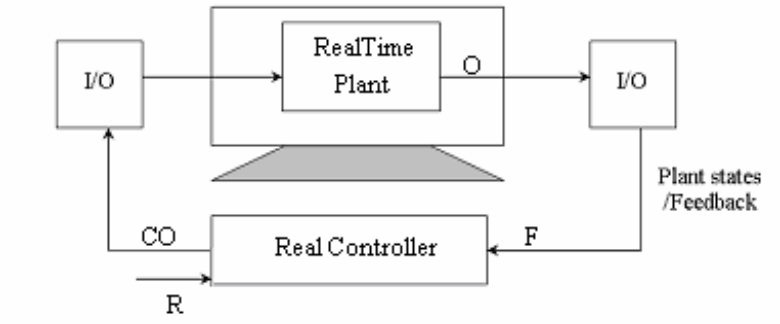


Figure 2.9: Hardware-in-Loop simulation.

The real I/O enables the resolving of time delay issues and hardware interface issues between the plant and the controller. For vehicle applications the I/O interface is typically accomplished using CAN. The controller receives the vehicle states as a feedback on the CAN, computes the necessary control commands, and sends them back to the plant through the CAN.

2.3.2 (b) Distributed Hardware-in-Loop Simulation

In this simulation, the subsystems constituting the plant are run on a distributed real time simulation platform. The use of multiple real time nodes may allow more detailed dynamic subsystem models to be implemented. It also may enable replacing the individual nodes with real subsystems when they are available for field testing. The layout of a distributed simulation is shown in Figure 2.10, which incorporates a dynamic model of a battery subsystem as an example. The distributed simulation can be accomplished using commercially available software like RT-LAB.

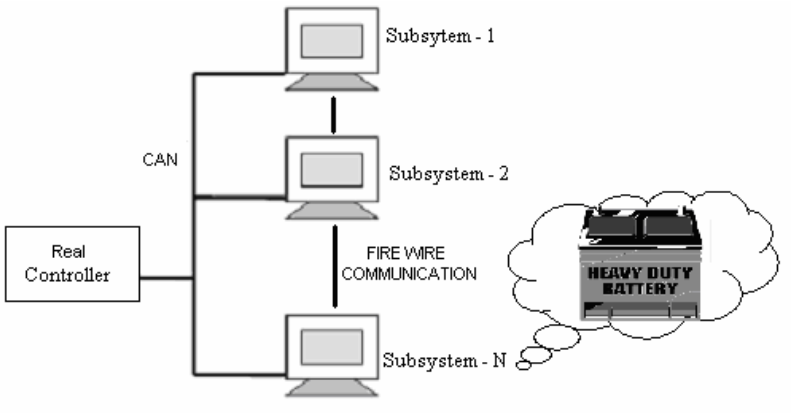


Figure 2.10: Distributed Hardware-in-Loop simulation.

Typically, the subsystems are synchronized by a high speed connection (FireWire – IEEE 1394) to make sure all the subsystems are at the same time step (synchronization) [9]. The dynamic model of a subsystem such as a battery is run on a dedicated real time node connected to the CAN.

2.3.2 (c) Distributed Hardware-in-Loop Simulation with Vehicle Subsystem Hardware

This simulation involves replacing a subsystem simulation with actual subsystem hardware. Figure 2.11 shows the distributed simulation including actual hardware for the

battery and its control module (BCM). This configuration allows benchmark testing of several battery packs and choosing the best one without ever mounting them into a real vehicle. This will also allow an easy performance evaluation and validation of the dynamic models of the battery subsystem. This procedure can be repeated for other subsystems namely EM, generator and ICE. In a stepwise procedure, the simulated subsystems can be replaced with actual components. The primary advantage is that testing of individual ECUs can start as the hardware implementation decisions are made and the components become available. This is very important because of the different lead times while ordering the subsystems for a HEV.

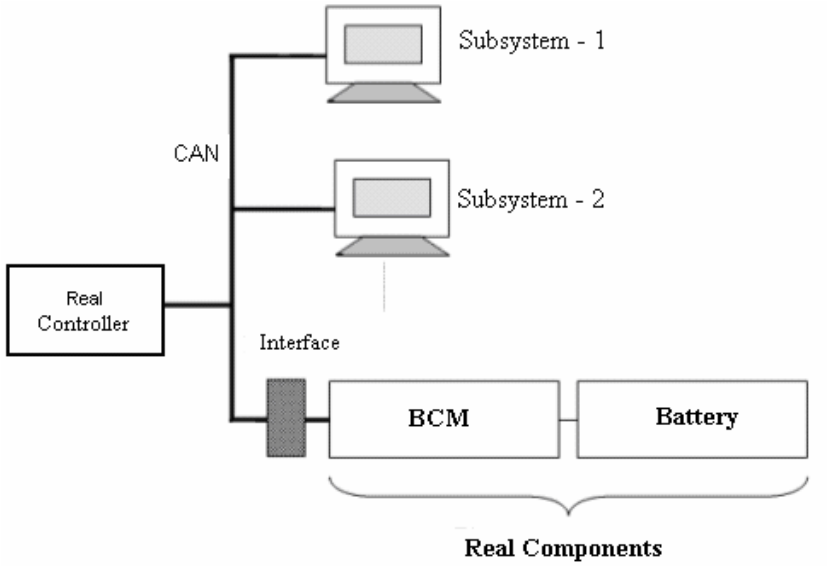


Figure 2.11: Distributed Hardware-in-Loop simulation including a real battery module.

2.4. Hardware-in-Loop Simulation: Description

Hardware-in-Loop simulation is an important step in the development process of a vehicle. A typical car today employs a large number of ECUs resulting in a complex vehicle control system. ECU testing plays a major role in the Vehicle Development

Process (VDP) [10] [11] to ensure fault tolerance and safe operation of the vehicle subsystems. This becomes more prominent in case of an HEV with two propulsion units (ICE and EM) and onboard energy storage devices. The development engineers have to consider all the possible failure modes and ensure a safe state of the vehicle in all known extreme conditions. In reality, it is not feasible to test all the failure modes on a real vehicle primarily because of the cost involved and also because of a possible lack of infrastructure or personnel. HIL simulation provides a reliable and affordable means of controller tuning and failure mode analysis from an early stage of the vehicle controller development and acts as a bridge between offline simulations and real time implementation. To demonstrate a few of the many capabilities, a HIL setup is built from the ground up and its various capabilities are demonstrated in the following sections.

2.4.1 HIL Simulation: History

HIL simulation has been used by the aerospace industry since the early 1980's. The systems used were expensive, because the hardware and software used were not generic and were customized for an application [12]. It was not until the early 1990's when the processing speed of a PC was increased at a rapid rate and the unit cost of a PC came down that a computer with sufficient computation capability became available on every office desk.

The early application of HIL simulation in automotive systems was seen for Antilock Braking System (ABS) control applications [12]. The ABS controller could be tested on a vehicle model running in real time under various test conditions that would otherwise have been expensive and difficult to be done on the testing track. The next application of

the HIL simulation seen in the automotive field was for vehicle dynamics control (VDC) [12] [13]. It was demonstrated that the HIL technique reduced many man-hours involved and improved the reliability of the system designed. The VDC was primarily used to actively superimpose a stabilizing yaw torque on the vehicle by applying separate brake pressures to each wheel. The HIL was subsequently used for engine control applications in testing the Engine Control Module (ECM). Testing the ECM using HIL simulation under various operating conditions and faults became an easy alternative to initial dynamometer testing [14].

Realizing the importance of the HIL platform, more automotive companies started utilizing the technique. A photograph of a HIL platform at Audi is given in Figure 2.12 [15]. A complete powertrain simulation including all the networked ECUs interfaced to a real time simulator was demonstrated. Another example of the HIL technique being employed into the development process is at the Bugatti design center [16]. The HIL system was successfully utilized in the design of the Bugatti Veyron powertrain.

Figure 2.12: Simulation platform at Audi (Courtesy: dSpace, USA)

Besides the automotive applications, another area where the HIL technique has gained popularity is space research. The Canadian space agency has successfully employed the HIL technique in tuning the controllers used in space stations. The HIL system simulated

a model of the zero gravity environments for tuning the controllers. The results were very close to the actual tests done [17]. The above mentioned examples are some of the important areas where HIL platforms are currently being used in industry. Various HIL simulation platforms have been reported in University research. A HIL platform to simulate an environment for testing flight control algorithms has been studied at the University of Maryland, College Park [18]. In another application, a small scale HIL setup was used to test an Antilock Braking System (ABS) controller at Sung Kyun Kwan University, Korea [19].

The main concern in a HIL implementation is the fidelity of the real time model in representing the plant. This immediately asks for a good dynamic model of the plant. With the vehicular systems becoming more and more complex, faster computational systems are needed to simulate the plant accurately, and distributed real time simulation is a useful tool for this purpose. Commercial software capable of performing the distributed real time simulations is available [20]. The objective is to use regular IBM compatible PCs running a RTOS, connected in a network and capable of simulating the vehicle. This idea can be applied to the entire vehicle or its individual subsystems. The following section describes the development process for the vehicle controller and how SIL and HIL are incorporated.

2.4.2 HIL Simulation of a Hybrid Electric Vehicle

A hierarchical layout of various ECUs in a series-parallel 2x2 HEV is shown in Figure 2.13. The vehicle controller, referred to as Supervisory Control Module (SCM), communicates with the subsystem controllers over the CAN bus.

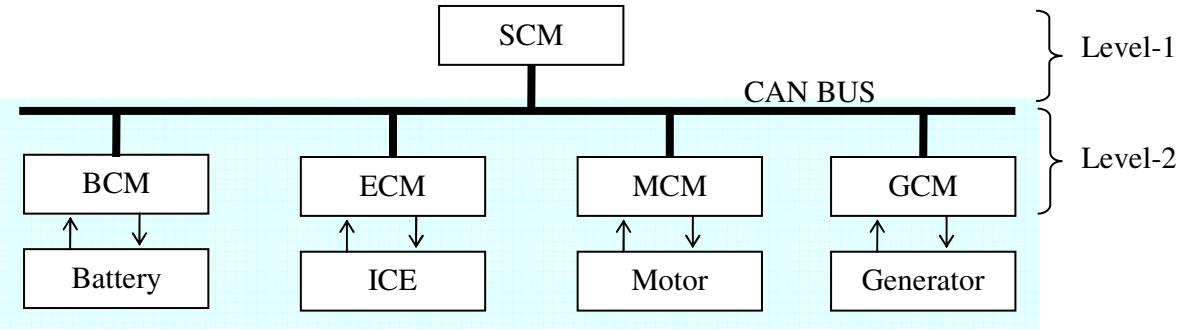


Figure 2.13: Various ECUs in a series-parallel 2x2 HEV.

A HIL simulation can be done at two levels.

1. Level – 1: A vehicle level simulation where a real controller (SCM) is interfaced to a simulated vehicle.
2. Level – 2: A subsystem level simulation where a real subsystem control module is interfaced to a simulated subsystem.

The various possibilities are:

1. A real SCM interfaced to a simulated HEV.
2. A real Battery Control Module (BCM) interfaced to a simulated Battery
3. A real Engine Control Module (ECM) interfaced to a simulated Engine
4. A real Motor Control Module (MCM) interfaced to a simulated Electric Motor
5. A real Generator Control Module (GCM) interfaced to a simulated Generator

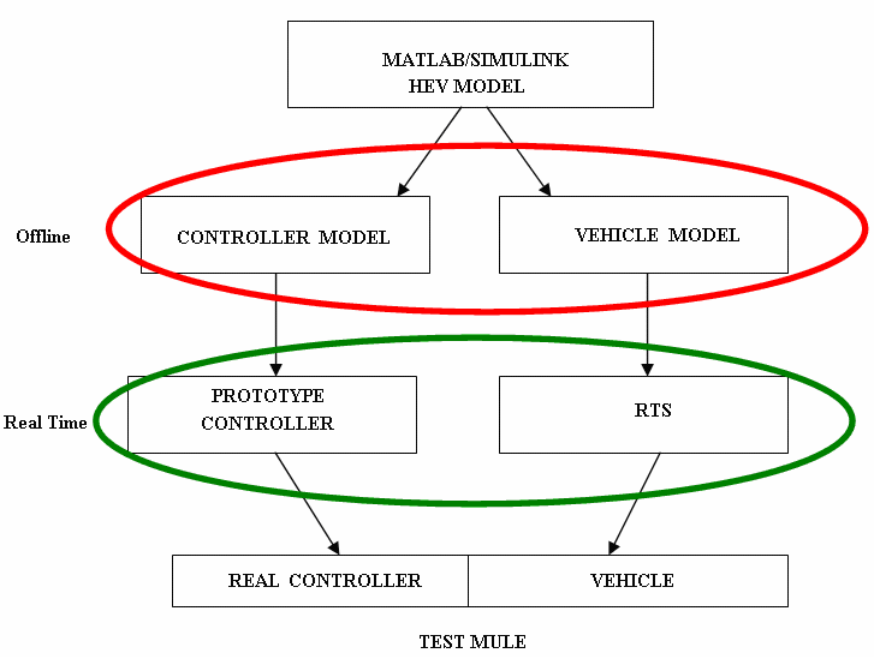


Figure 2.14: SCM development process.

Once an offline model of the HEV ensures that the vehicle meets the base requirements, the above mentioned testing is done in stages as the ECU modules become available. The development process for the vehicle controller development can be summarized as given in Figure 2.14.

Simulation in Matlab/Simulink of a HEV including the controller and the vehicle model is run offline and then run in real time on a HIL simulation platform, and then incorporated into the test vehicle. Once the vehicle controller is thoroughly tested the code can be embedded into a production controller.

The effective testing of various ECUs, the validation of vehicle and subsystem models of a HEV requires a scalable and reusable HIL simulation platform. The HIL platform

documented in the Chapters III and IV provides the required flexibility and ease of implementation.

2.5 Conclusion

This chapter introduced the series-parallel 2x2 HEV architecture to be used for SIL and HIL simulation in the following chapters. The various stages in the development process of a vehicle controller are described. The necessary background for SIL and HIL simulations is presented. The following chapters provide details of the SIL and HIL simulations of a series-parallel 2x2 HEV.

CHAPTER III

SOFTWARE-IN-LOOP SIMULATION AND CONTROL STRATEGY DEVELOPMENT

3.1 Introduction

This chapter describes the SIL simulation of a series-parallel 2x2 HEV architecture. The unique HEV configuration was incorporated into PSAT by ANL on a request to facilitate the simulation of The University of Akron HEV [8]. This chapter outlines a preliminary control strategy for a series-parallel 2x2 HEV and the procedure for adding a new control strategy into PSAT for SIL simulation of the HEV.

The series-parallel 2x2 HEV model with the new control strategy incorporated has been simulated in PSAT on various drive cycles that represent various driving scenarios like highway driving, urban driving and acceleration. As discussed in Section 2.3.2 the SIL simulation and MIL simulation are closely tied together and hence the HEV model was used for obtaining the offline results. This chapter explains the structure and the results of offline simulations conducted in a non real time environment to ensure that the sizing of subsystem components was satisfactory. The Matlab/Simulink model of the series-parallel 2x2 HEV is used to obtain real time code subsequently used in Chapter IV for HIL simulation.

3.2 HEV Control Strategies

Development of a control strategy for a HEV is an interesting and challenging task for a control engineer. A HEV has a complex combination of dynamic systems of various time constants and transients. The existence of high speed electrical transients in an EM drive and slower mechanical transients in a vehicle make it difficult to develop a comprehensive strategy.

The main objective of The University of Akron team in the Challenge X competition was to hybridize a stock Chevrolet Equinox into a series-parallel 2x2 HEV while maintaining or exceeding the Vehicle Technical Specifications (VTS) as given in Table 3.1 by the Challenge X competition organizers.

Table 3.1: Vehicle Technical Specifications

Description	**Base Vehicle Performance**	**Challenge X Competition VTS**
IVM * – 60 mph	8.9 sec	≤ 8.0 sec
50 – 70 mph	6.8 sec	≤ 6.8 sec
Vehicle Mass	3825 lb	≤ 4400 lb
MPG Combined EPA	23.3 mpgge	≥ 32 mpgge

* IVM – Initial Vehicle Movement.

The series-parallel 2x2 HEV mainly consists of the following powertrain subsystems:

1. ICE subsystem
2. Electric motor drive subsystem
3. Generator subsystem

4. Energy storage subsystem - battery pack

The SCM supervises vehicle subsystems and decides the set points of operation for them over a given drive cycle. Drive cycle is a profile of desired vehicle speed vs. time. A layout of the SCM in a vehicle control system is given in Figure 3.1. A hierarchical layout of the SCM interfaced to the ECUs of vehicle subsystems is given in Figure 3.2. The Battery Control Module (BCM), Motor Control Module (MCM), Generator Control Module (GCM), and the Engine Control Module (ECM) receive the commands from the SCM over a CAN bus and control the respective subsystem operation as required and within their operating constraints.

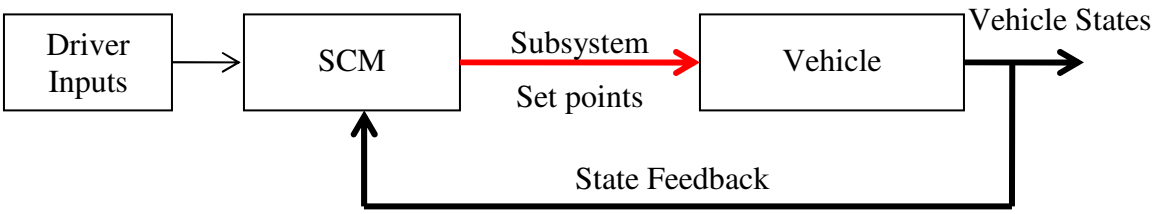


Figure 3.1: SCM interface with the vehicle.

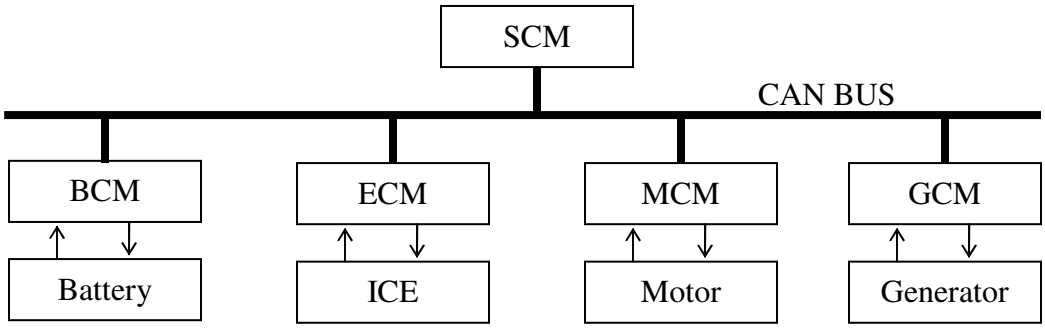


Figure 3.2: Hierarchical layout of various ECUs in a HEV.

The SCM needs to ensure that the primary requirement of meeting the driver request is met within the operating constraints of the subsystems. These requirements and constraints are described below.

1) Driver Demands: The primary requirement is to satisfy the driver demand. The SCM has to ensure that the vehicle speed meets the driver speed demand. For offline simulations the speed demand schedule is available in the form of a drive cycle. In simulation study the SCM along with a model of the driver monitors the vehicle speed to ensure that it is within 2% of the drive cycle. In real life this schedule is decided by the accelerator and brake pedals positions set by a driver in a vehicle.

2) Battery State of Charge (SOC): The SCM needs to ensure that the energy flow in and out of the battery is balanced over a given drive cycle, i.e. the vehicle operation must not require charging the onboard battery using an external power supply. This property, which is called charge sustainability, is necessary for a HEV to have the same functionality as a conventional vehicle [8].

3) Constrained operation: The SCM should make sure that the subsystems operate within their safe operating constraints and ensure an emergency shutdown in case of subsystem failure(s).

4) Optimization and Emissions: The SCM has to ensure that the vehicle systems operate at the best system efficiency, and perform the necessary trade off regarding the tail pipe emissions without any intervention from the driver. For an ICE, the zones of

highest efficiency and least emissions lie in different areas on the engine map and hence a trade off is needed [21].

In essence, the control strategy running in a SCM makes sure that the driver demand is met by coordinating the operation of ICE, EM and a generator, and also at the same time minimizes the fuel consumption (cost of operation) and emissions.

HEV control strategies fall mainly into two categories. The control strategies in the first category are based on optimizing vehicle fuel consumption using optimal control or dynamic programming techniques. These control strategies assume a prior knowledge of the drive cycle or vehicle operating conditions and are not practical for real-time application. The second category of control strategies effectively comprises rule based strategies. These control techniques are based on calculating the power requests to the subsystems based on the current requirements from the driver and feedback of current states of the vehicular subsystems like state of charge of battery, vehicle speed and EM speed.

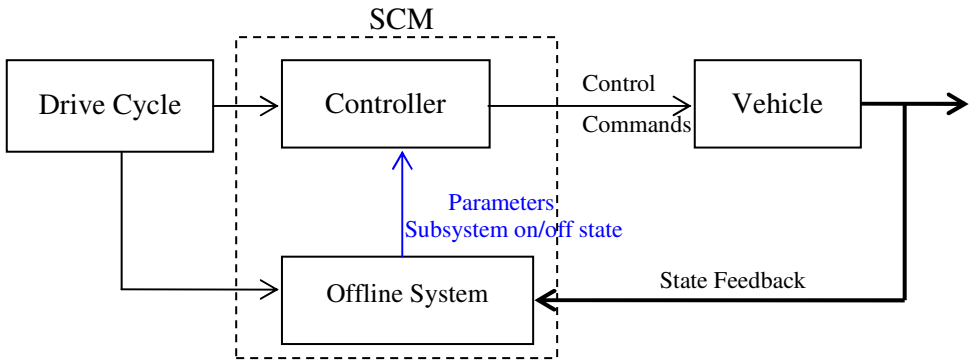


Figure 3.3: Layout of a HEV controller using optimal control techniques.

The layout of a HEV control system using optimal control techniques is given in Figure 3.3. The drive cycle data and the states of the vehicle subsystems are fed into an offline

system. In a simple case the offline system provides the subsystem on/off states by optimizing a cost function representing the fuel consumption. This optimization requires prior knowledge of the drive cycle and therefore must be done offline. Dynamic programming techniques provide a flexible and accurate means of optimization but with an overhead of computing time and memory. The dynamic programming technique applied to HEV control has been reported by Chau [22]. At run time, a controller calculates the required commands to the subsystems based on the subsystem on/off states provided by the offline system. Because the offline system needs the operating conditions ahead of time, it is impossible to realize the controllers in a real life situation where the requests from the driver are quite varied and arbitrary [3] [23]. However, the simulation results of a HEV following a given drive cycle are still necessary for benchmarking the performance of the strategies when compared to strategies that have already been developed.

The layout of the second category of HEV control system using rule based control techniques is given in Figure 3.1. There is no offline system and the instantaneous set points of the subsystems are decided based on preset thresholds and rules. The advantage of this category of the control techniques is that they are easily implemented and can be tested in real time. Hellegren [24] and Hyoeun [25] provided a detailed analysis of the HEV control which was primarily rule based and used fuzzy logic.

Owing to the ease and feasibility of writing a rule based strategy and running it in real time, a rule based strategy was written and implemented in PSAT for a series-parallel 2x2

HEV. The performance of the vehicle using this strategy is presented in the results section of this chapter.

3.3 Preliminary Control Strategy for a Series-Parallel 2x2 HEV

A preliminary rule based control strategy was written that ensures the main requirements mentioned in Section 3.2 are successfully met. The rules used for distributing the torque demand from the driver to the three subsystems, namely engine, EM and generator, are provided as follows. The overall logic for how the torque request is distributed among the subsystems is summarized in the following pseudo code:

Start of Pseudo Code

COMPUTE required demand torque from the driver

COMPUTE SOC of the energy storage system (SOC_{CUR})

IF power can be drawn from the energy storage based on SOC_{MIN} ($SOC_{CUR} > SOC_{MIN}$)

    IF EM alone can provide the demand

        Use the EM alone to provide the demand

    ELSEIF EM alone cannot provide the required demand

        # Engine needs to be turned on

        Employ the EM to provide its maximum output until the SOC_{CUR} falls below the minimum set point

        Rest of the required demand will be provided by the ICE in the efficient operating zone powering the front wheels and the generator if needed.

IF power cannot be drawn from the energy storage based on SOC_{MIN} ($SOC_{CUR} \leq SOC_{MIN}$)

    IF optimal ICE output $\geq$ Driver torque demand

        Use the ICE at its optimal operating point

The requested demand goes to wheels

Rest of the unused power from ICE is used to run the generator until SOC_{CUR} builds up in the storage unit

IF generator is running at full capacity

EM engages in regeneration

ELSEIF optimal ICE output < Driver torque demand ≤ maximum ICE output

Use ICE at maximum (wide open throttle)

Rest of the unused power from ICE is used to run the generator until SOC_{CUR} builds in storage unit

IF generator is running at full capacity

EM engages in regeneration

ELSEIF maximum ICE output < Driver torque demand

Use ICE at maximum torque (wide open Throttle)

Driver torque demand is not satisfied.

End of Pseudo Code#####

A schematic representation of the information flow from one decision block to other when modeled in Simulink is shown in Figure 3.4. Subsystem SS-1 calculates the drive motor torque that can be provided. Based on this torque availability from the drive motor the SS-2 subsystem calculates the torque to be provided by the ICE. The subsystem SS-3 calculates the torque to the generator that should be produced for charging the energy storage system.

This preliminary strategy was modeled in Simulink and saved as a control library in PSAT. The lookup tables and some of the basic building blocks were taken from other

control libraries. The above mentioned logic is realized using various Simulink blocks. The next stage included integrating the strategy into PSAT to build the HEV model. A screenshot of the control library is given in Figure A.1 in the Appendix A.

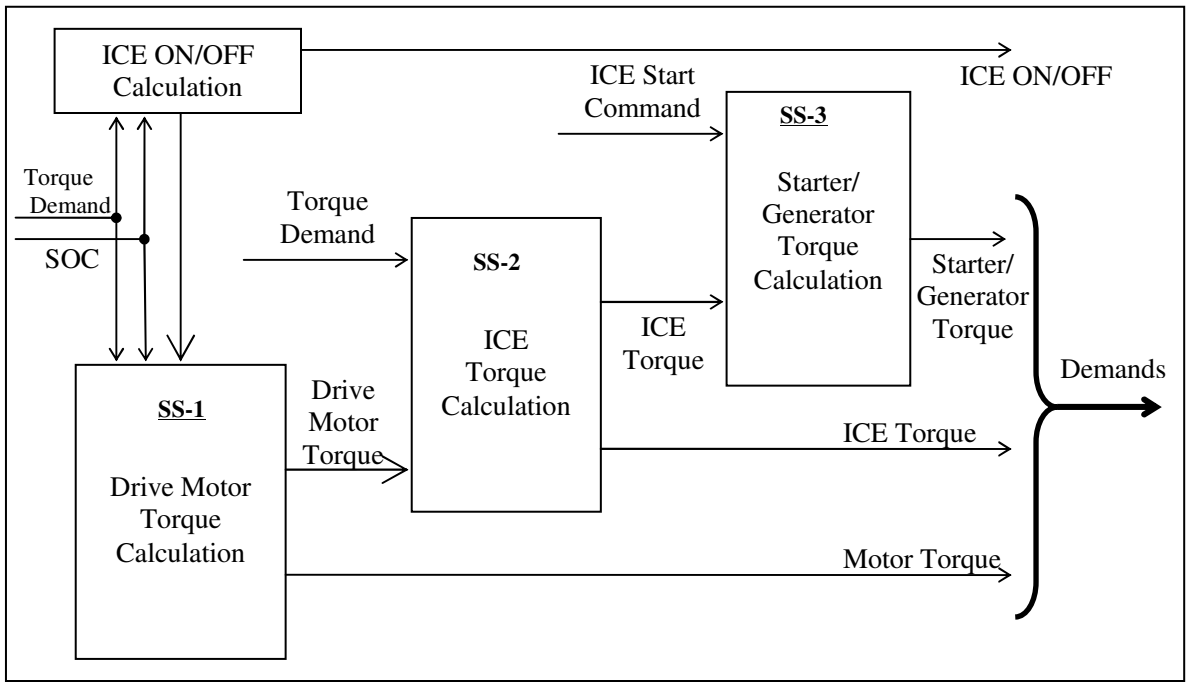


Figure 3.4: Schematic representation of the information flow in the series-parallel 2x2 HEV control strategy.

3.4 Modeling in PSAT - Simulation of a HEV

PSAT is a modeling and simulation tool that predicts vehicle fuel economy and performance using a steady state representation of the vehicle components. PSAT can simulate predefined vehicle configurations such as conventional, electric, fuel cell, series hybrid, parallel hybrid, and power split hybrid. PSAT is primarily a Graphical User Interface (GUI) built over the Matlab/Simulink environment with access to libraries of vehicle subsystems built in Simulink and corresponding powertrain initialization data obtained from field tests.

The working of PSAT can be understood from Figure 3.5. The various configurations and subsystem components are selected by the user via a GUI. In Stage-1, the powertrain layout and component selections are made. In Stage-2, the powertrain data and subsystem component data from field tests used by the subsystem models are loaded into the Matlab workspace. The Matlab workspace also contains the drive cycle data, simulation parameters like step size, solver type used for simulation.

In Stage-3, a Matlab/Simulink model of the vehicle is built and the simulations can be started by the user. Once the simulation is run the results are read using the GUI and displayed for visualization and further analysis. The simulations can be run manually or in an automated manner. To enable the user to keep track of simulations, a Matlab file (rerun0.m) is created to rerun the simulation any time later, if necessary. The GUI coordinates all the stages in a seamless manner.

A series-parallel 2x2 HEV model was constructed in PSAT. The control strategy for the series-parallel 2x2 HEV architecture needed was not available in the existing library and hence a new one was added. This was needed to predict the performance of the vehicle and to ensure that the design specifications were met with the chosen powertrain components.

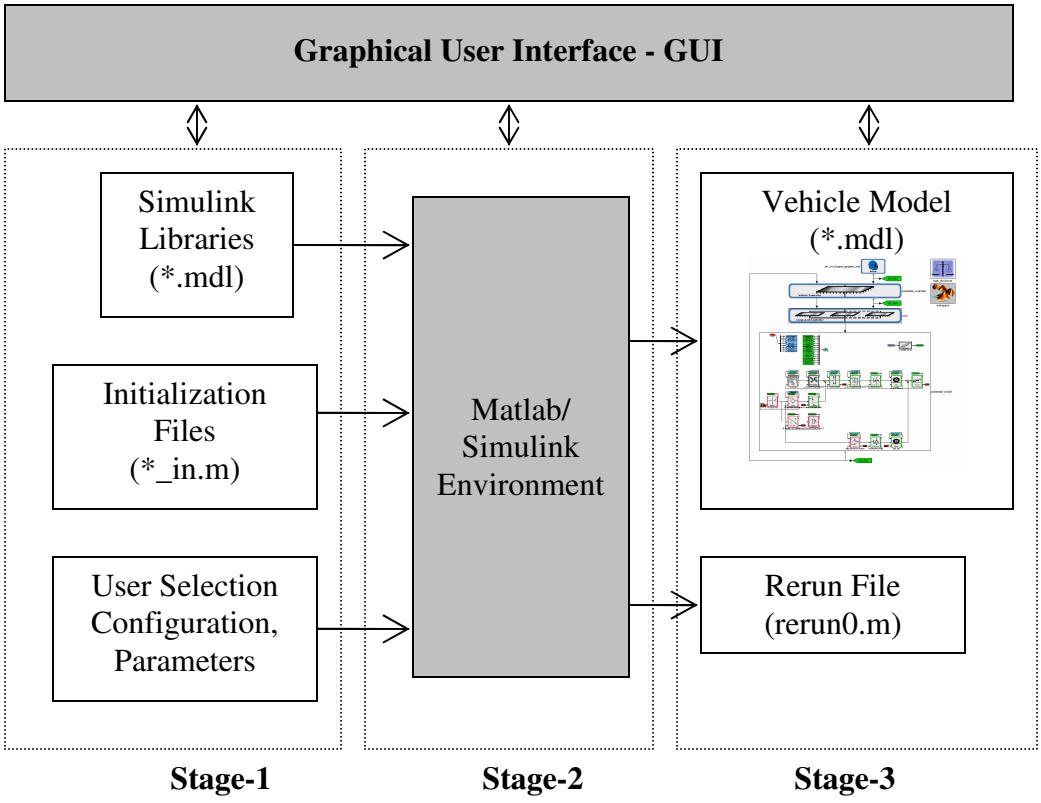


Figure 3.5: PSAT: Vehicle model building and associated files.

The preliminary control strategy saved in the PSAT control library described in Section 3.3 was added. The control strategy was structured so that its inputs and outputs match with those of other strategies. The control strategy is chosen in conjunction with the powertrain components in the PSAT control panel as discussed in Section 3.5. The following sections provide information on incorporation of the control strategy into PSAT and the results of simulations of the HEV on various drive cycles.

3.5 Model Description and Adding a Control Strategy

The flow diagram of a HEV model in Simulink built using PSAT is given in Figure 3.6.

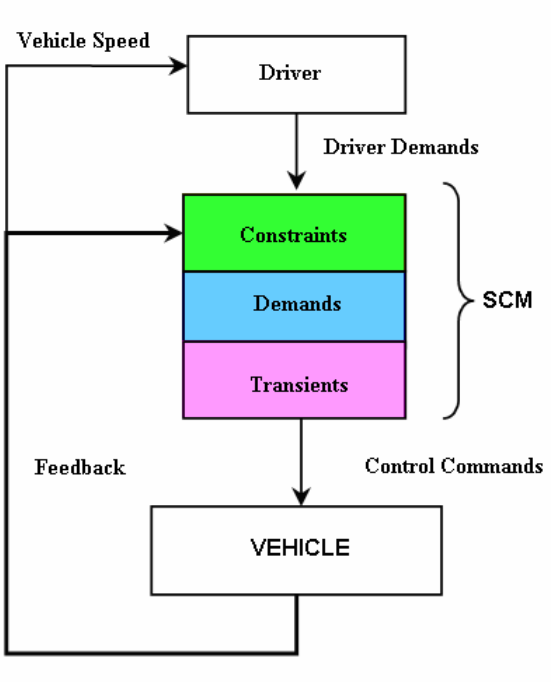


Figure 3.6: Flow diagram of a vehicle model in PSAT.

The speed profile (drive cycle) is accessed from the Matlab workspace in the driver subsystem. The desired speed can be customized by changing the appropriate data file that is loaded into the workspace. In the driver subsystem, the desired speed is compared with the feedback vehicle speed and a driver torque demand is given as an input to the SCM. The SCM does the necessary computation and sends the control commands i.e. instantaneous set points of operation for the subsystems. The SCM is subdivided into three Simulink subsystems namely constraints, demands, and transients. The constraints block has limiting and saturation blocks to ensure that realistic commands within the operating limits of the subsystems are requested. The demands block constitutes the main control algorithm, which computes the best combination of the subsystem operating points. For the series-parallel 2x2 HEV these are the engine torque, EM torque, generator torque, and engine on/off commands. The transients block incorporates necessary delays

corresponding to the additional time taken for the communication between the SCM and the subsystem controls making the controller model realistic. A schematic of the SCM with the three subsystems is shown in Figure 3.7. A Simulink control block having this same set of outputs was built ensuring that the data flow of the PSAT simulation is maintained.

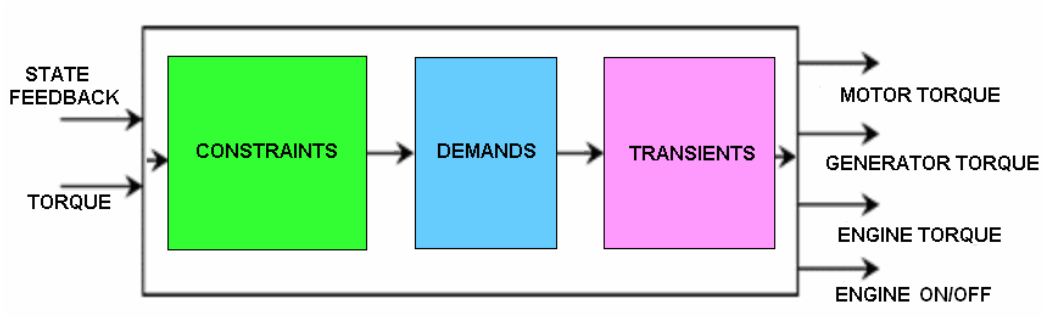


Figure 3.7**:** Layout of a SCM.

The control strategy discussed earlier was incorporated in Simulink and saved into the PSAT control library. This control library in Simulink is called during the model building stage of PSAT (Stage-2) and incorporated into the demands block of the SCM. The control strategy saved in PSAT is modular and can be improved in the future.

The Matlab/Simulink model of a series-parallel 2x2 HEV model was built using PSAT as explained in the next section, followed by the results of the SIL testing done to evaluate the HEV performance over various drive cycles.

3.6 Configuring SIL Simulation of a Series-Parallel 2x2 HEV in PSAT

The various steps to obtain a desired vehicle configuration in PSAT and simulating it over a drive cycle are given in this section.

Step 1: Vehicle Configuration

The first step is to select a vehicle configuration. The user can get started in PSAT by selecting one of the many pre-defined vehicle configurations available. A sample screenshot of the list of existing configurations in PSAT is given in Figure 3.8.

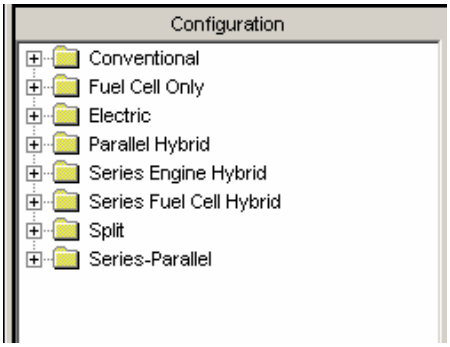


Figure 3.8: Screenshot of the list of existing architectures in PSAT.

Step 2: Vehicle Components

The second step is to select the subsystem components that constitute the powertrain of a vehicle. The GUI enables the user to add new vehicle components or update the current component models with ease. The GUI also allows changing parameters of the components as and when the components change at a later stage. The HEV model is configured in a stepwise manner adding the required subsystem components. The list of components used for the current series-parallel 2x2 HEV is given in Table 3.2.

Table 3.2: List of components used to model the series-parallel 2x2 HEV

Component	Type	Rating
Engine	Compression Ignition	VW 1.9L 110 kW
Electric Motor Drive	Induction Motor	35 kW (Continuous) 70 kW (Peak)
Starter/Generator	Induction Motor	21 kW (Continuous) 40 kW (Peak)
Vehicle	-	Chevrolet Equinox
Transmission	Manual	VW 5 speed
Final Drive (Front)	-	3.77 :1
Final Drive (Rear)	-	3.42 :1 10.1:1 (with coupling)
Wheels	-	Equinox wheels
Electrical Accessory	-	100 W
Mechanical Accessory	-	0 W
Energy Storage	NiMH battery pack	6.5 Ah/ 275 V

Step 3: Control Strategy

The next step is to select the appropriate control strategy. A screenshot of the list of strategies in PSAT that includes the new control strategy added for series-parallel 2x2 HEV is given in Figure 3.9.

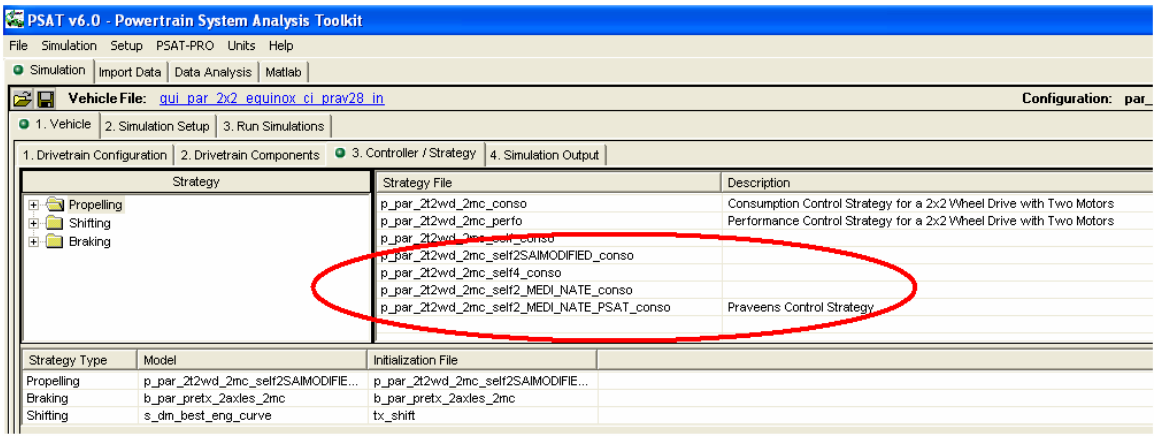


Figure 3.9: List of control strategy libraries: propelling.

Step 4: Simulation Setup

The vehicle is configured for simulation once the components and control strategy are selected. The drive cycle over which the vehicle needs to be simulated is also selected at this step. New drive cycles can be added as per the requirements. A screenshot of the list of drive cycles including a sample drive cycle named ‘CHALLENGEX’ is given in Figure 3.10. The solver for simulating the Simulink model and the time step for simulation are also selected.

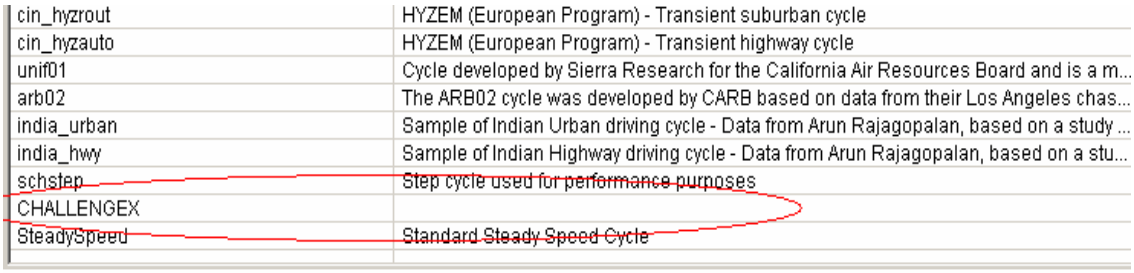


Figure 3.10: Sample list of drive cycles.

Step 5: Build Model and Run Simulations

The next step in the HIL process is running the simulations. The vehicle model is built based on the selections made in earlier steps and simulated over the list of drive cycles selected. Once the simulations are complete, the results are automatically displayed in the GUI. Also, a spreadsheet in Microsoft Excel is generated for reporting purposes. A top level screenshot of the series-parallel 2x2 HEV Simulink model generated by PSAT is given in Figure 3.11.

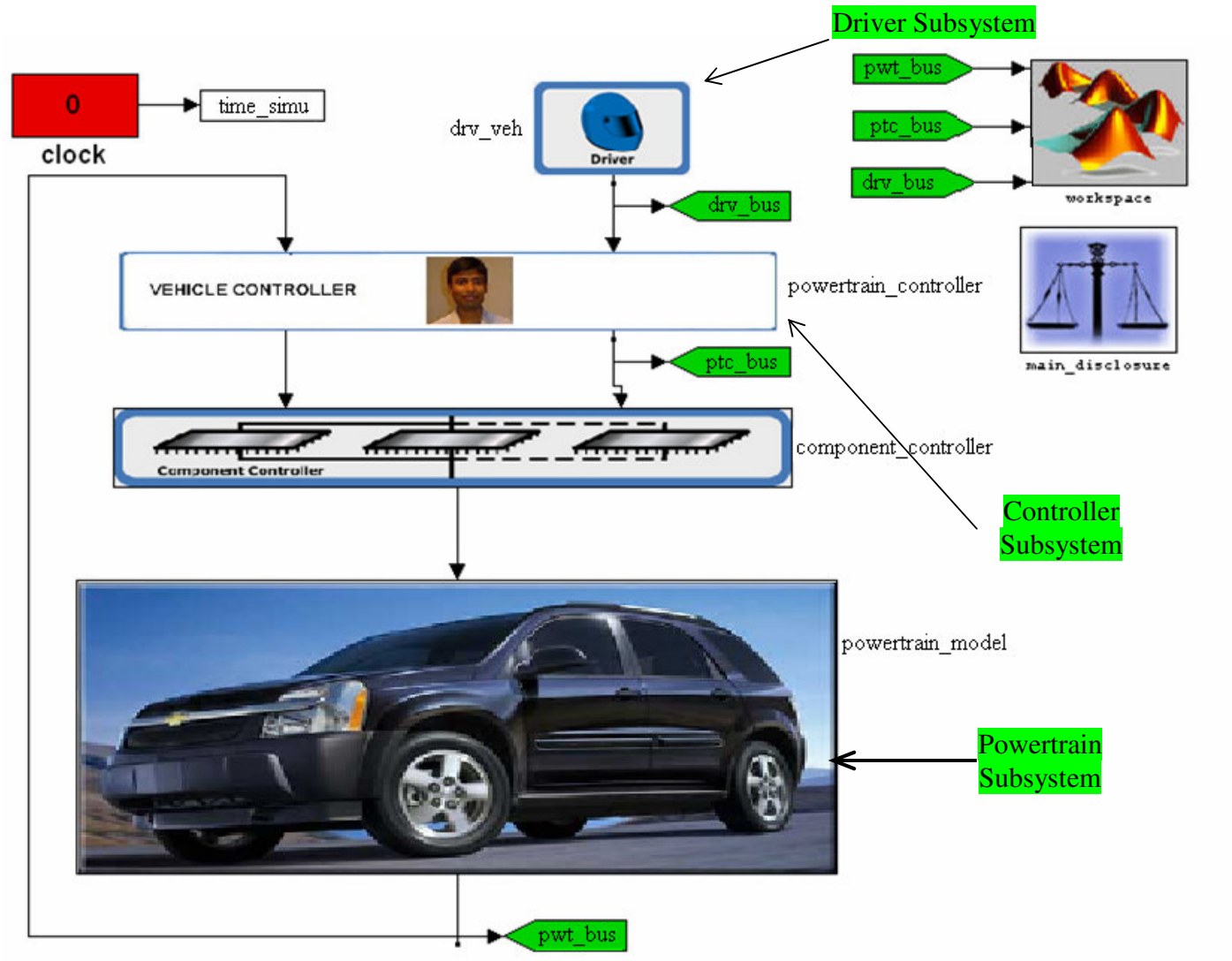


Figure 3.11: Screenshot of a series-parallel 2x2 HEV Simulink model.

3.7 SIL Simulation Results

To understand and demonstrate the various operating modes, the HEV model was simulated at a steady speed. The operation of the HEV in parallel mode is demonstrated in Figures 3.12 and 3.13. It can be seen that the battery SOC is controlled by bringing the generator into operation when $SOC \leq 0.6$ (Region 1) and bringing the drive motor into operation when $SOC \geq 0.65$ (Region 2). The SOC can be seen to vary smoothly between the limits. The propulsion torque, which is the sum of engine, generator, and drive motor torques, remains within 3.3% of its average value, except for some narrow spikes.

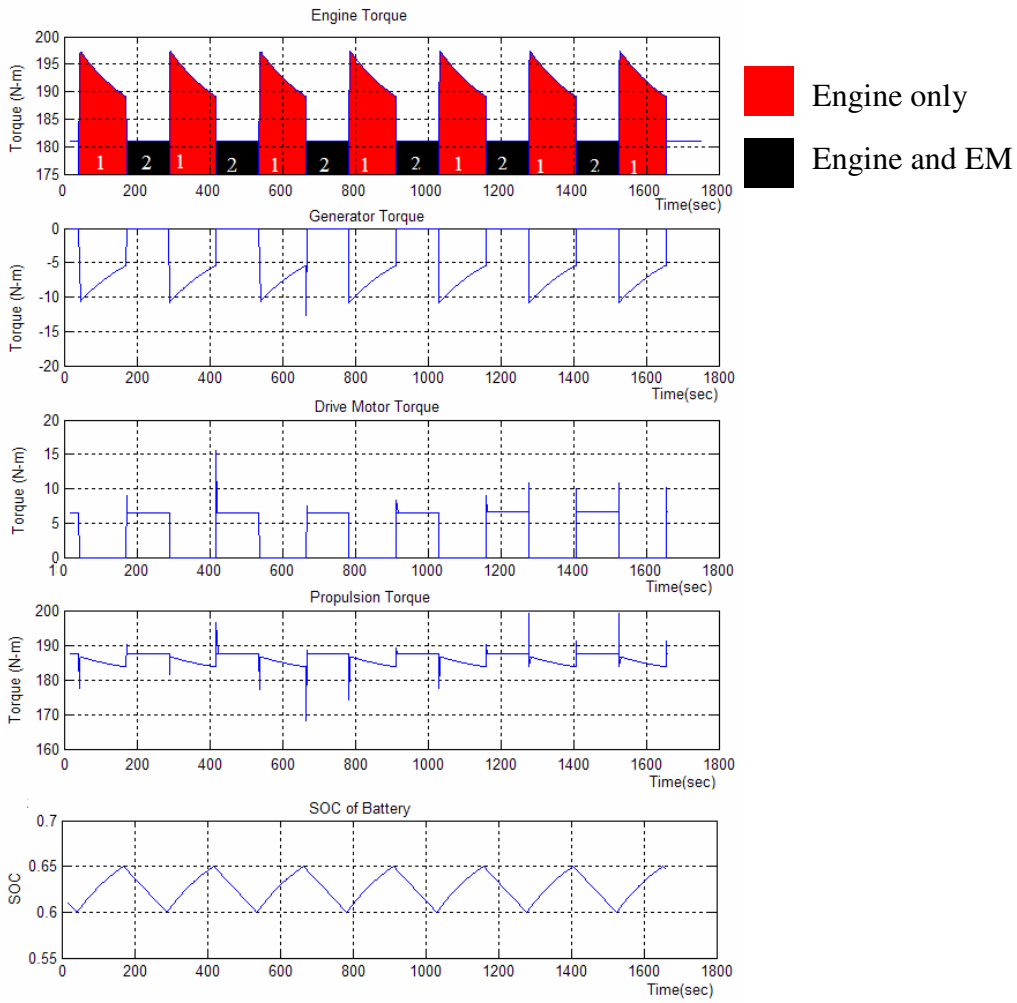


Figure 3.12: Engine, generator and EM operating torques and battery SOC.

It can be seen in Figure 3.13 that the vehicle speed error is small, demonstrating that the vehicle was following the speed profile as expected and discussed in Section 3.2.

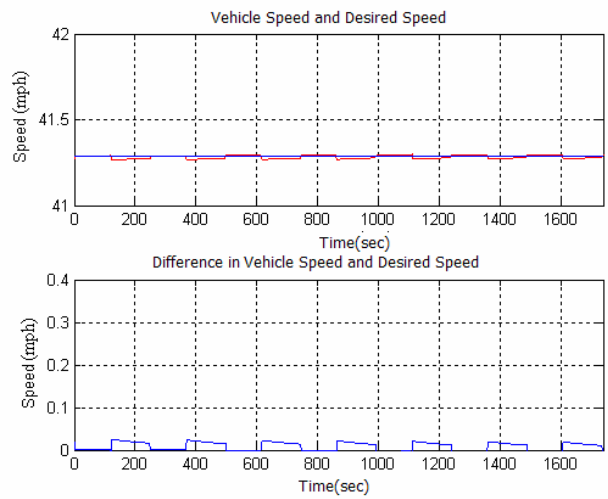


Figure 3.13: Vehicle speed, desired speed, and their difference.

Next the HEV was simulated for a low speed acceleration. The operation of the vehicle in electric only mode and the transition to engine only mode is shown in Figure 3.14. It can be seen that at low speeds the required torque is provided by the EM and when a higher torque is requested by the driver the engine is brought into operation to meet the requirements. This demonstrates the electric only mode and engine only mode.

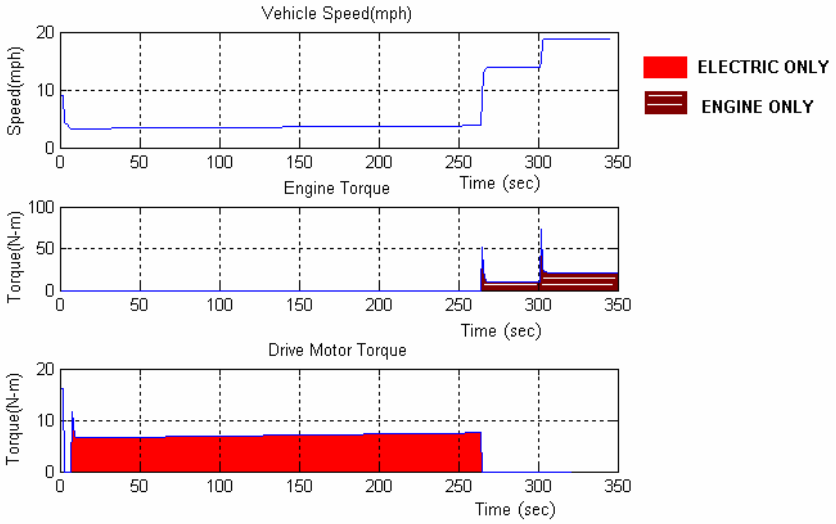


Figure 3.14: Vehicle speed (low speed), engine torque (N-m) and EM torque (N-m).

The Matlab/Simulink model of the series-parallel 2x2 HEV section is simulated over various drive cycles, to exercise the control strategy and to ensure that the HEV meets the initial requirements given in Table 3.1.

The simulation results were saved as a '.mat' file ('.mat' file is a format for storing data in Matlab) for further visualization. The operation of the HEV and the subsystem operating points in various driving scenarios, namely urban driving, highway driving, and acceleration test, are given in the following sections.

3.7.1 Urban Dynamometer Driving Schedule (UDDS): Urban Driving

The operating points of engine in series-parallel 2x2 HEV when run on a UDDS cycle are given in Figure 3.15. An engine operating point denoted as a " * " on the engine map is a plot of the value of engine torque and engine speed at a particular instant. This is done for each time step of the simulation. The operating points for the EM and generator are also obtained in the same manner. The engine operating points in a conventional vehicle over the same drive cycle are given in Figure 3.16. From the graphs it can be seen that in a HEV the ICE is used fewer times than the conventional vehicle because the ICE in a HEV is not running all the time as seen in a conventional vehicle. In a conventional vehicle, the ICE operating points are quite dense in the low torque, low speed zone corresponding to the low efficiency zone of the ICE. It can be seen that in a HEV the density of ICE operating points in the low torque, low speed zone corresponding to a low efficiency zone is less, compared to a conventional vehicle.

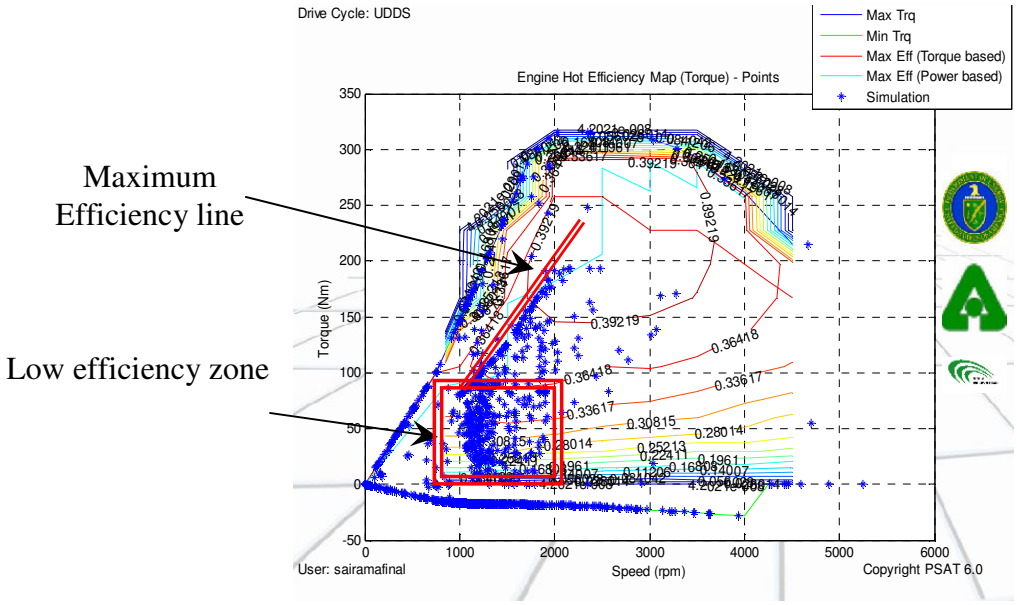


Figure 3.15: UDDS - engine operating points in a HEV.

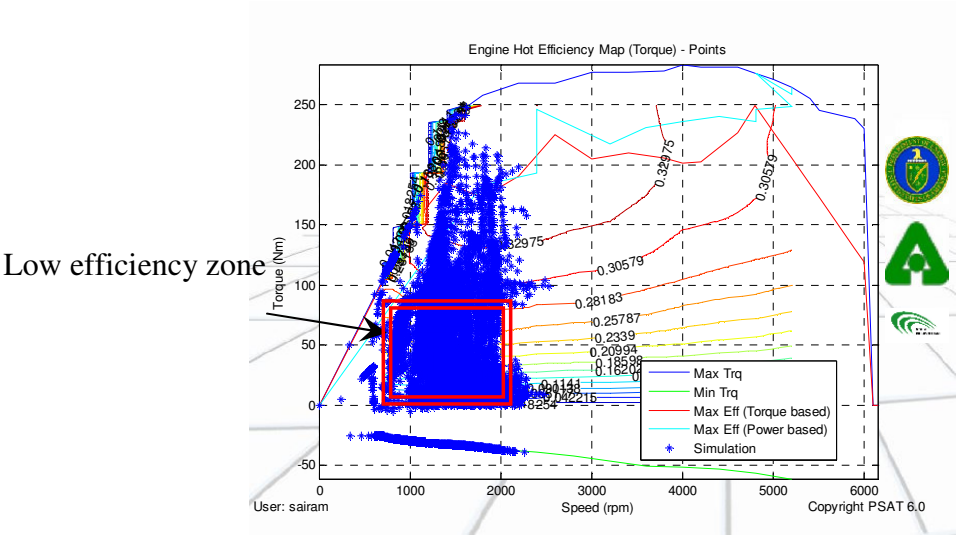


Figure 3.16: UDDS - engine operating points in a conventional vehicle.

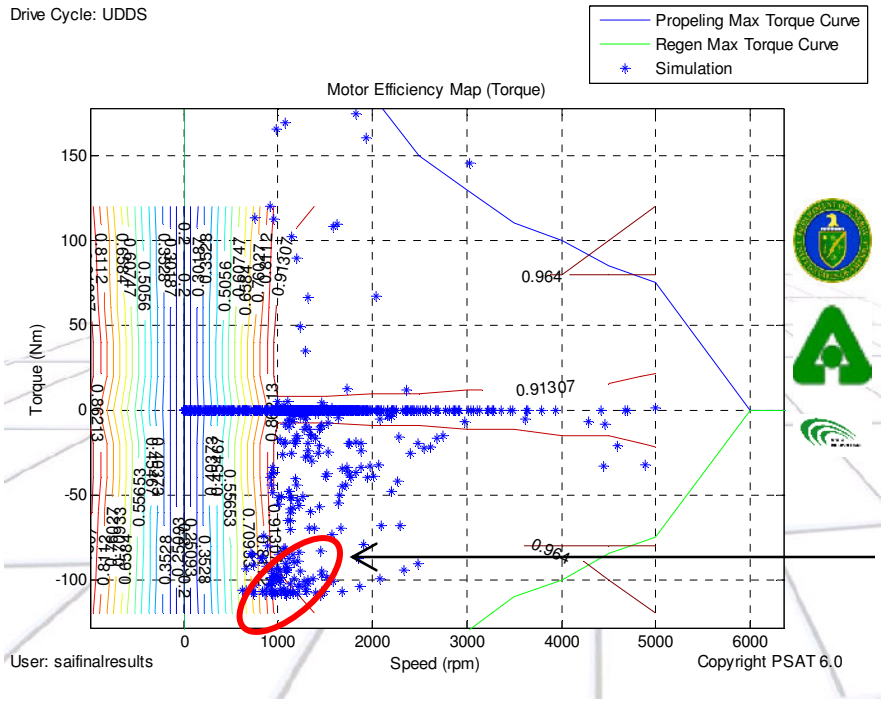
To perform a qualitative analysis, the ratio of average engine efficiency to the peak engine efficiency termed as Operating Efficiency Ratio (OER) was computed for the engines in a HEV and a conventional vehicle on a UDDS cycle. The efficiencies and OER for each of the engines are given in Table 3.3. The two engines are of different sizes

with different peak efficiencies, and hence OER is a good measure for their comparison rather than their average operating efficiency. It can be seen that the OER of the engine in a HEV is greater than the OER of the engine in a conventional vehicle indicating that the engine in a HEV is operating closer to the higher efficiency zone compared to the engine in a conventional vehicle.

Table 3.3: UDDS - engine efficiencies in a conventional vehicle and a HEV

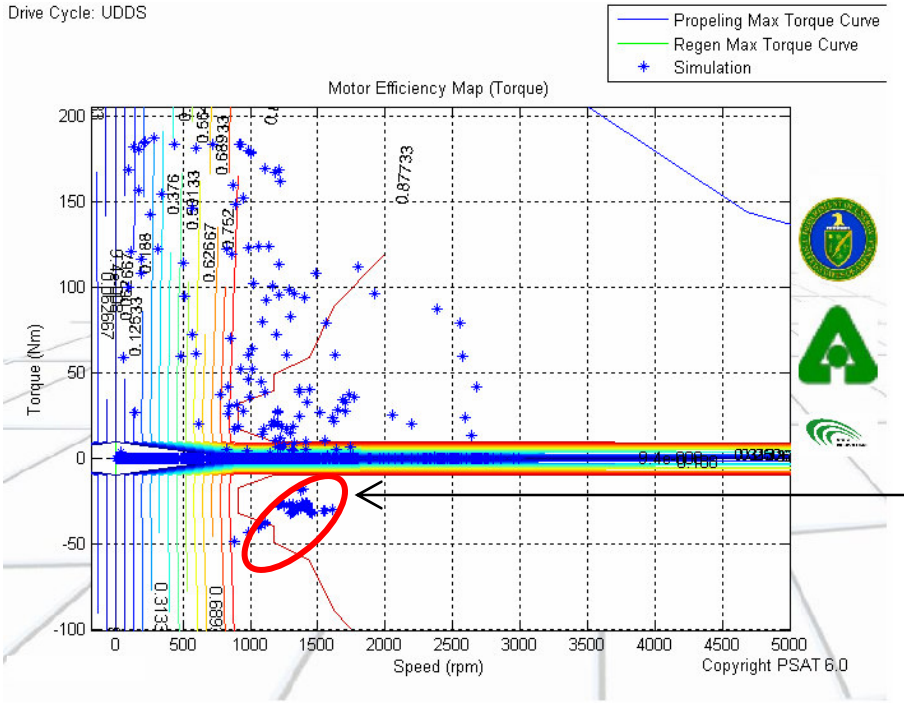
	Engine in a conventional vehicle	Engine in a HEV
Average Efficiency (%)	23.69	38.58
Peak Efficiency (%)	35.37	42.02
Operating Efficiency Ratio (OER) (Average Efficiency / Peak Efficiency)	66.97	91.8

The operating points of the starter/generator along with the contours of constant efficiency are given in Figure 3.17. It can be seen that the electric machine is used as a generator in the high efficiency zone. The operating points of the EM are given in Figure 3.18. It can be seen from Figure 3.18 that there are many operating points for the EM in the regenerative zone ($T_{EM} < 0$) demonstrating braking which is common in urban driving.



Operating points in high efficiency zone (Close to the contours of 90% and above)

Figure 3.17: UDDS - starter/generator operating points.



Regenerative Capability

Figure 3.18: UDDS - EM operating points.

3.7.2 Highway Fuel Economy Test (HWFET) Cycle: Highway Driving

The PSAT model of the HEV is simulated over a HWFET cycle. The operating points of ICE on a HWFET cycle are given in Figure 3.19.

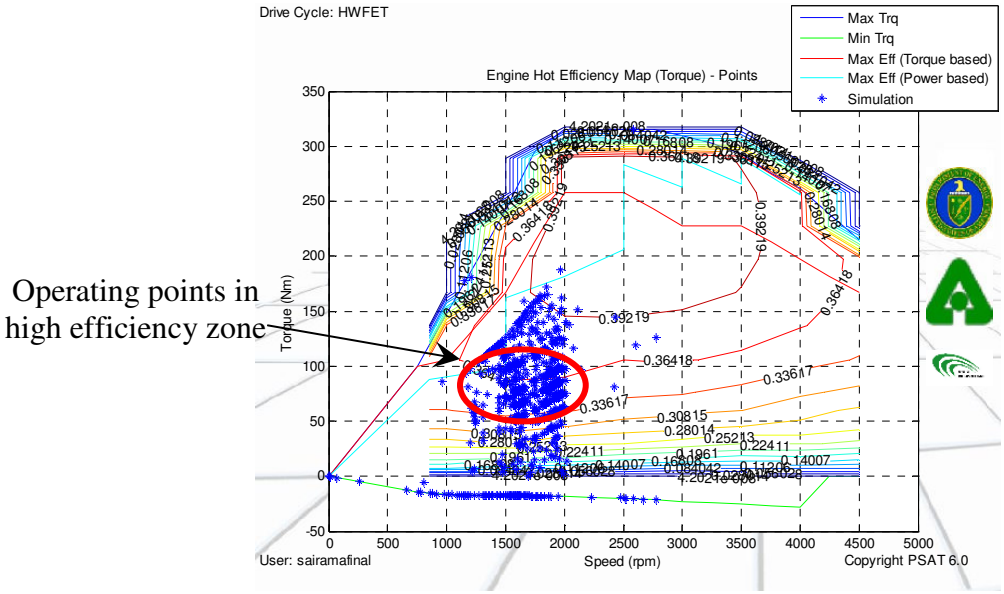


Figure 3.19: HWFET - engine operating points.

It can be seen that the operating points on a HWFET cycle lie near the higher efficiency zone when compared to those on a UDDS cycle as seen in Figure 3.16. To perform a qualitative analysis, average engine efficiency, peak engine efficiency and OER were computed for the engines in a HEV and a conventional vehicle on a HWFET cycle, and are given in Table 3.4. It can be seen that the OER of the engine in a HEV is greater than the OER of the engine in a conventional vehicle indicating that the engine in a HEV is operating closer to the higher efficiency zone compared to the engine in a conventional vehicle, similar to the results seen in Section 3.7.1 on a UDDS cycle.

Table 3.4: HWFET - engine efficiencies in a conventional vehicle and a HEV

	Engine in a conventional vehicle	Engine in a HEV
Average Efficiency (%)	29.57	36.97
Peak Efficiency (%)	35.37	42.02
Ratio (Current usage/Best possible)	83.59	87.97

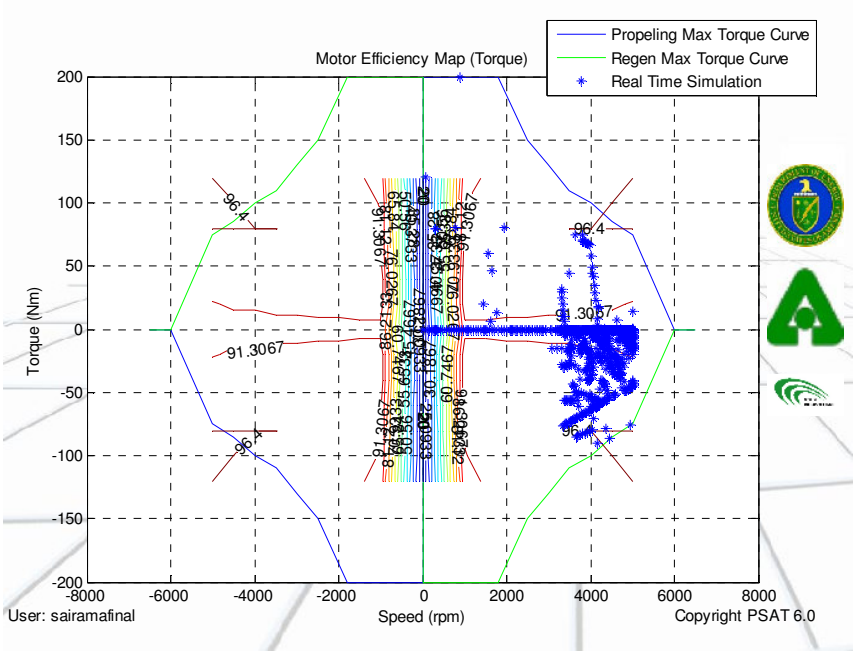


Figure 3.20: HWFET - starter/generator operating points.

The operating points for the starter/generator and EM are given in Figures 3.20 and 3.21. It can be seen that the utilization of the EM is less when compared to the UDDS cycle because of the cruise speeds in a HWFET.

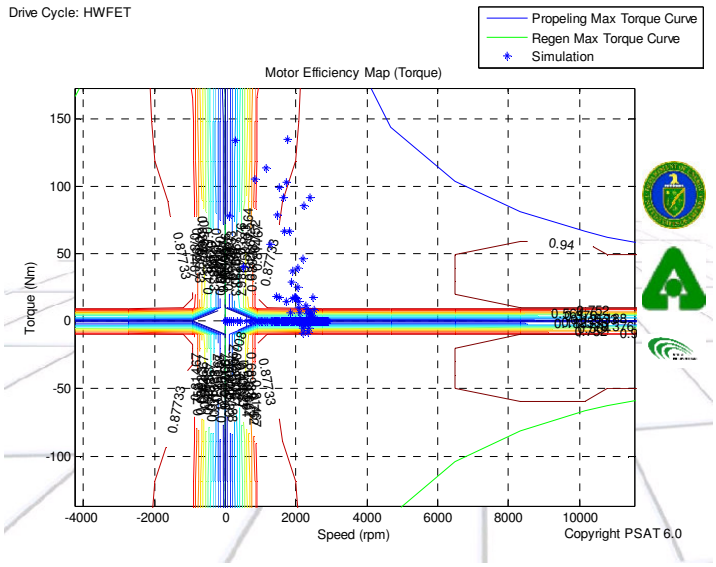


Figure 3.21: HWFET - EM operating points.

3.7.3 Acceleration Test (0-60 mph)

The engine, EM and generator operating points for the Acceleration test are given in Figures 3.22-3.24. It can be seen that maximum torques are requested from the engine and the EM.

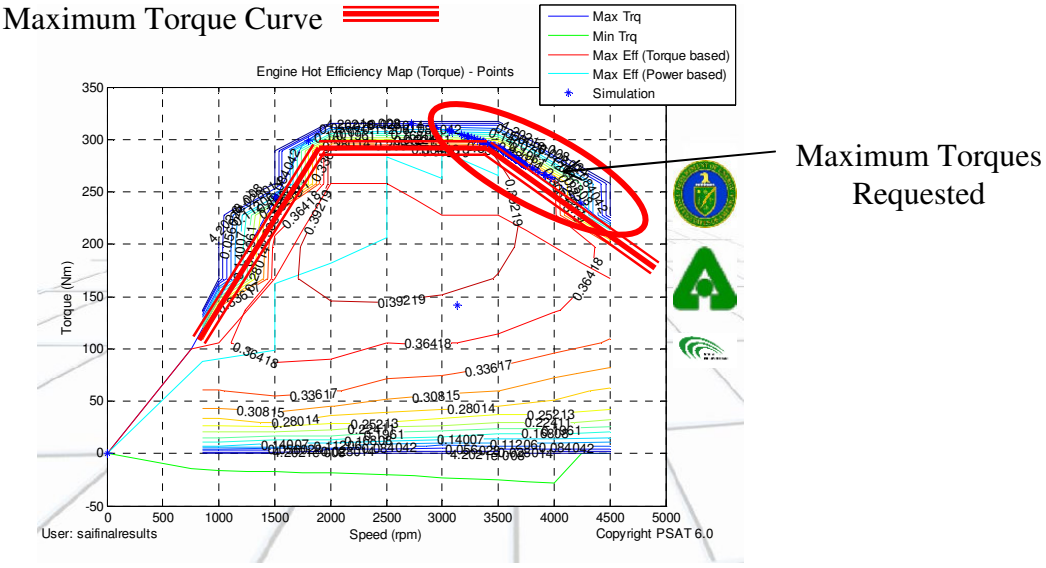


Figure 3.22: Acceleration test - engine operating points.

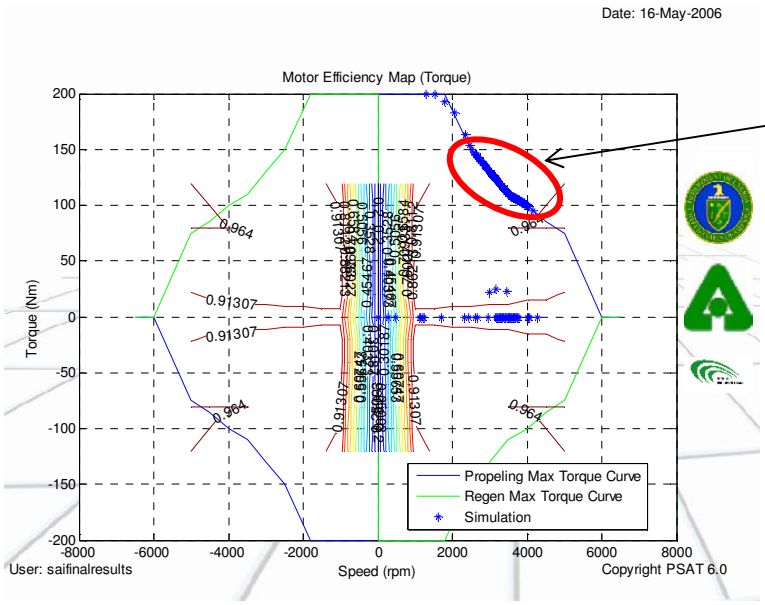


Figure 3.23: Acceleration test - EM operating points.

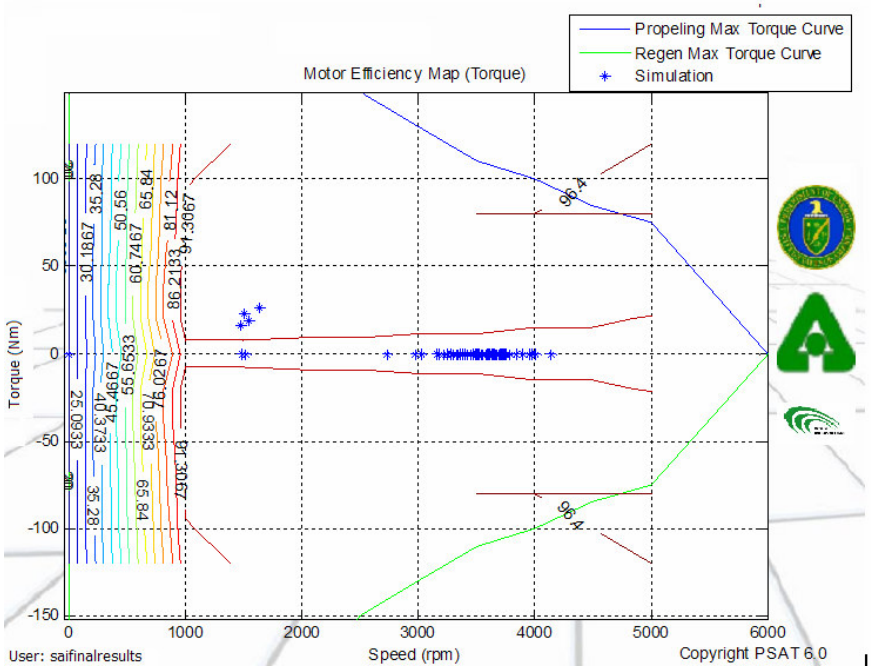


Figure 3.24: Acceleration test - starter/generator operating points.

The plot of vehicle speed versus time is given in Figure 3.25. It can be seen that 0-60 mph is around 8 sec and 50-70 mph is around 3.6 sec. Thus, the simulations indicate that the HEV will satisfy the VTS given in Table 3.1.

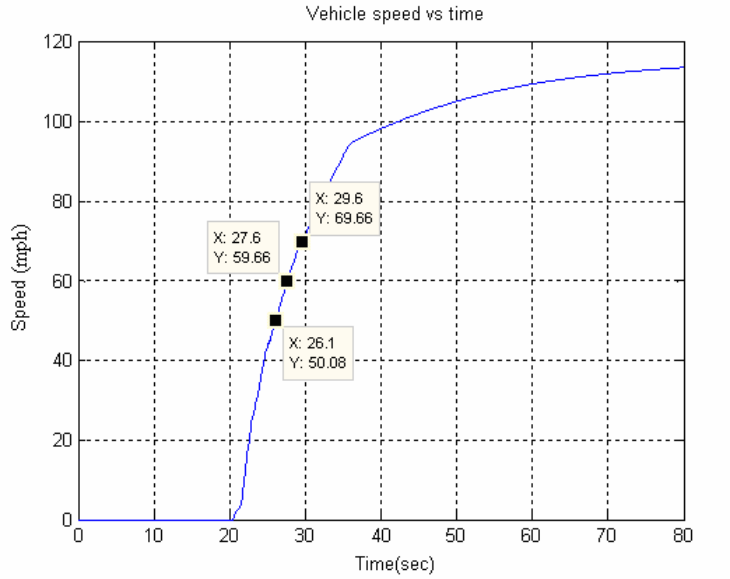


Figure 3.25: Acceleration test - vehicle speed.

The plots of engine torque, EM torque, generator torque and battery SOC are given in Figures 3.26-3.29. Because of a constraint on the lower limit of the battery SOC, the EM cannot provide torque after about t=36 sec. It can also be seen that the battery subsystem operates within the SOC limits 0.6 to 0.8 as set in the control strategy.

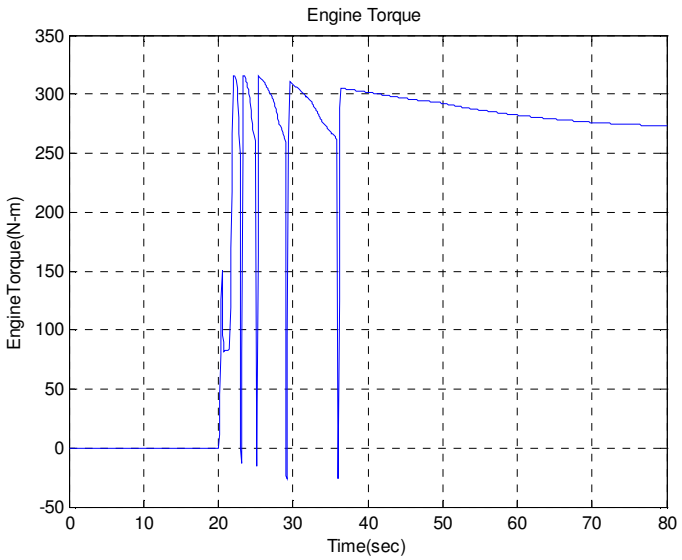


Figure 3.26: Acceleration test - engine operating torque (N-m).

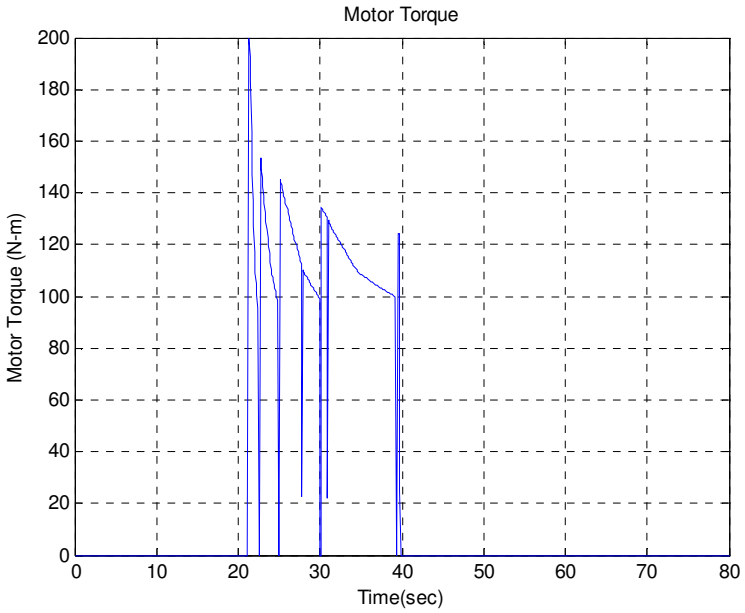


Figure 3.27: Acceleration test - EM operating torque (N-m).

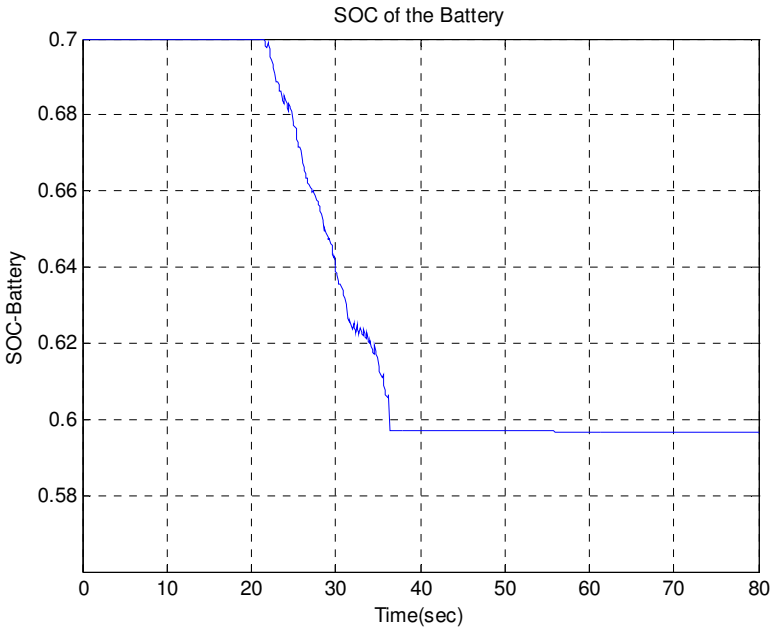


Figure 3.28: Acceleration test - battery SOC.

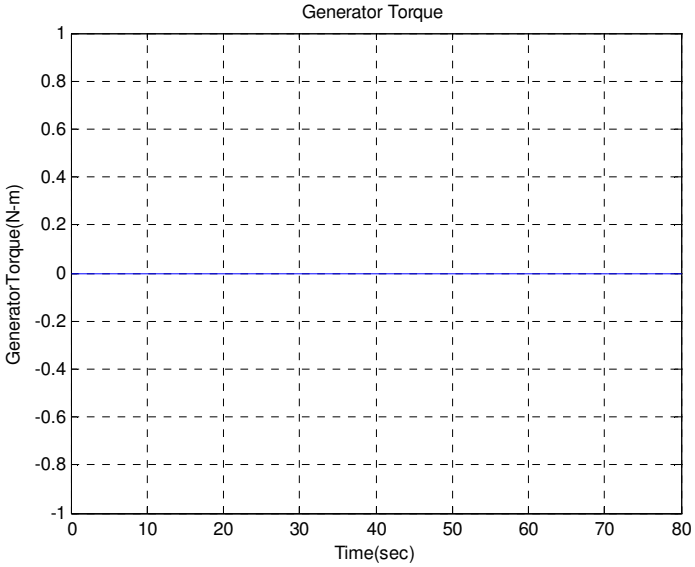


Figure 3.29: Acceleration test - generator operating torque (N-m).

It can be seen that the HEV design with the new control strategy and the selected vehicle subsystems satisfies the proposed specifications as given in Table 3.5. The SIL

simulation results predict that the series-parallel 2x2 HEV model outperforms the initial Vehicle Technical Specifications (VTS).

Table 3.5: PSAT simulation results

Description	**Initial VTS**	**PSAT series-parallel 2x2 Results**
IVM – 60 mph	≤ 8.0 sec	8.0 sec
50 – 70 mph	≤ 4.0 sec	3.6 sec
Vehicle Mass	≤ 4400 lb	4200 lb
MPG Combined EPA	≥ 35.6 mpgge	39.15 mpgge

The control strategy can be further improved for a better performance and fuel economy; however, this task is a project in itself and is left for future study. More control strategies can be added in a similar manner and the current strategy serves as a benchmark for comparison with new strategies. Further improvement is possible with careful deployment of ultracapacitors. This is suggested for a future study.

3.8 Conclusion: Offline Model

The development of a HEV model in PSAT is demonstrated in this chapter. The model along with the associated powertrain data is obtained for The University of Akron HEV designed as a part of the collegiate level student competition, Challenge X. The simulation model thus obtained using the control strategy described, has demonstrated to satisfy the initial VTS of the series-parallel 2x2 HEV.

The layout of the HEV model is described and the data flow in the model has been explained. The operation of the HEV in various driving conditions is demonstrated. The subsystems are seen to operate within limits. The new strategy is added in the list of PSAT control libraries for further improvement in the future. The procedure for modifying the offline HEV model to run it in real time on a HIL platform is given in Chapter IV.

CHAPTER IV

REAL TIME HARDWARE-IN-LOOP SIMULATION SETUP

4.1 Introduction

This chapter describes the HIL simulation setup built from the ground up for real time simulation of a HEV. The HIL setup was built using commercially available off-the-shelf hardware and software components. In the HIL setup, a Real Time Simulator is interfaced to a prototype controller. The stepwise procedure for rearranging the subsystems of the offline model developed in Chapter III to run in real time on a HIL simulation setup is explained. The data from a vehicle model running on the real time nodes is obtained over various drive cycles for further analysis in Chapter V. Also the versatility of the setup is demonstrated by running a HIL simulation of a vehicular subsystem namely the EM Drive.

4.2 Basics of Real Time Simulation

The Matlab/Simulink model of the series-parallel 2x2 HEV developed in Chapter III is used to obtain real time code. The tools used to obtain the real time code are Real Time Workshop (RTW) and RT-LAB. Rephrasing from Chapter II, a RTS includes the computer system(s) that monitor, respond to, or control an external environment.

In offline simulations, a system simulation can take one minute to simulate one hour of system behavior or one hour to simulate one minute of system behavior. In a real time simulation, the simulation time (t) must be synchronized with the actual time (T). Hence, the computations for each simulation time step have to be completed within the same corresponding interval of real-world time.

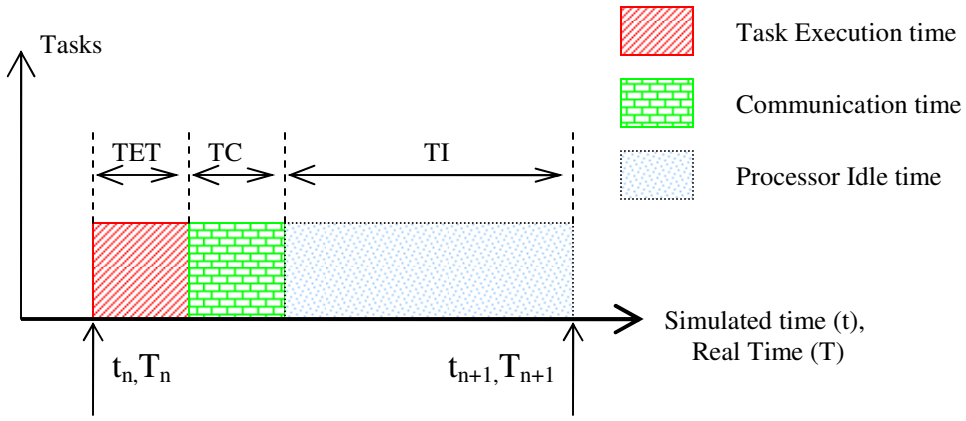


Figure 4.1: Timeline of a real time simulation.

A task running on a RTS is illustrated in Figure 4.1. A task is an instance of real time code running on a RTS. The time steps are given as t_n and t_{n+1}. In a perfect real time simulation one step of an event or task starts at t_n and ends at or before t_{n+1}. Task Execution Time (TET) is the computation time taken for the real time code of the model obtained from Matlab/Simulink to run on the RTS. Effectively it is the time taken to solve the system equations and obtain the system states at the next time step. TC is the time taken between the instant the states of the system are calculated and the instant they are reflected at other real time nodes or I/O interface of the RTS. This is typically due to the communication delays and A/D and D/A conversions. TI is the processor idle time until the next time step. TI is absent in centralized offline simulations as the simulation

time is not synchronized with a real clock. This is shown in Figure 4.2. In a centralized offline simulation the computation is done on a single node/processor and hence the task doesn't have to wait for synchronization with any external task.

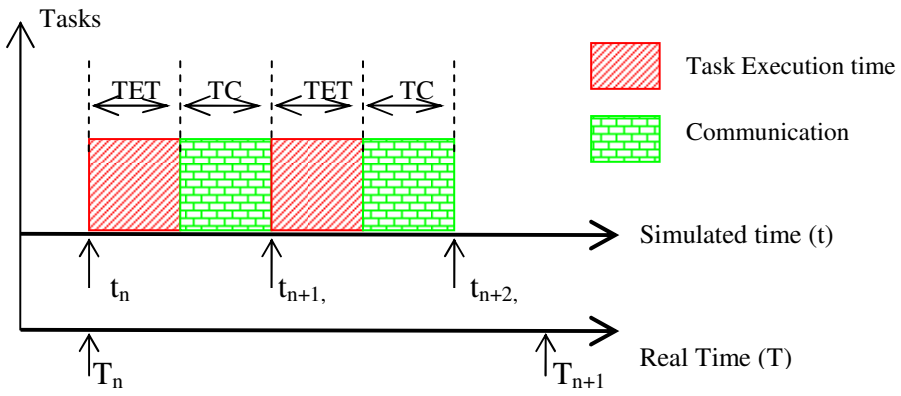


Figure 4.2: Timeline of an offline simulation.

An overrun is caused when the time taken for the real time code to execute on the RTS is more than the step size chosen for real time simulation. A limited number of overruns are tolerated and this simulation is classified as a soft real time simulation.

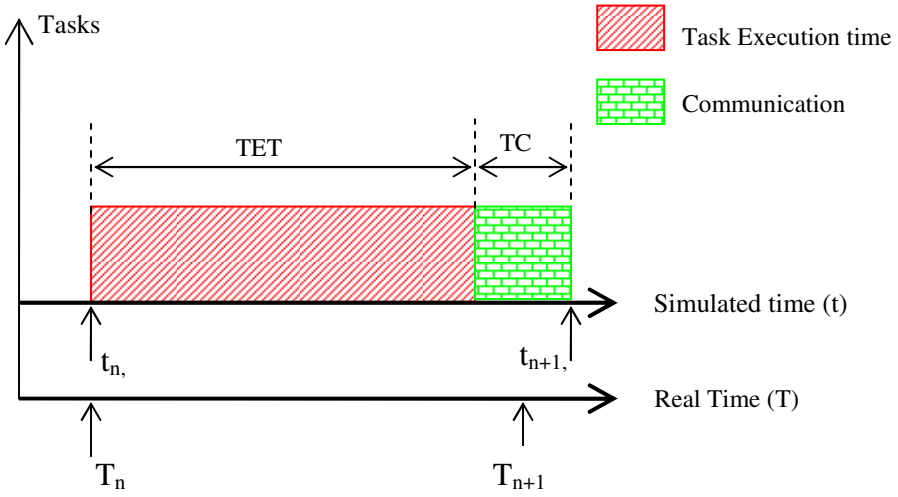


Figure 4.3: Overruns in a real time simulation.

Figure 4.3 shows an overrun in a real time simulation. Overruns can be eliminated by increasing the time step of the real time simulation, but at the expense of a less accurate representation of the plant (vehicle in this context). Hence, there is a tradeoff between the number of overruns and the accuracy of representation of the plant.

4.3 Hardware-in-Loop Simulation Setup

In HIL simulation, a model of the plant runs in real time on a RTS. The RTS may consist of a single node or multiple nodes as per the complexity of the plant model. The plant model in this case happens to be a HEV and its subsystems. A schematic representation of HIL simulation of a HEV and its subsystems with the CAN interface is given in Figure 4.4. It should be ensured that the model runs on the RTS without any overruns as discussed in the previous section.

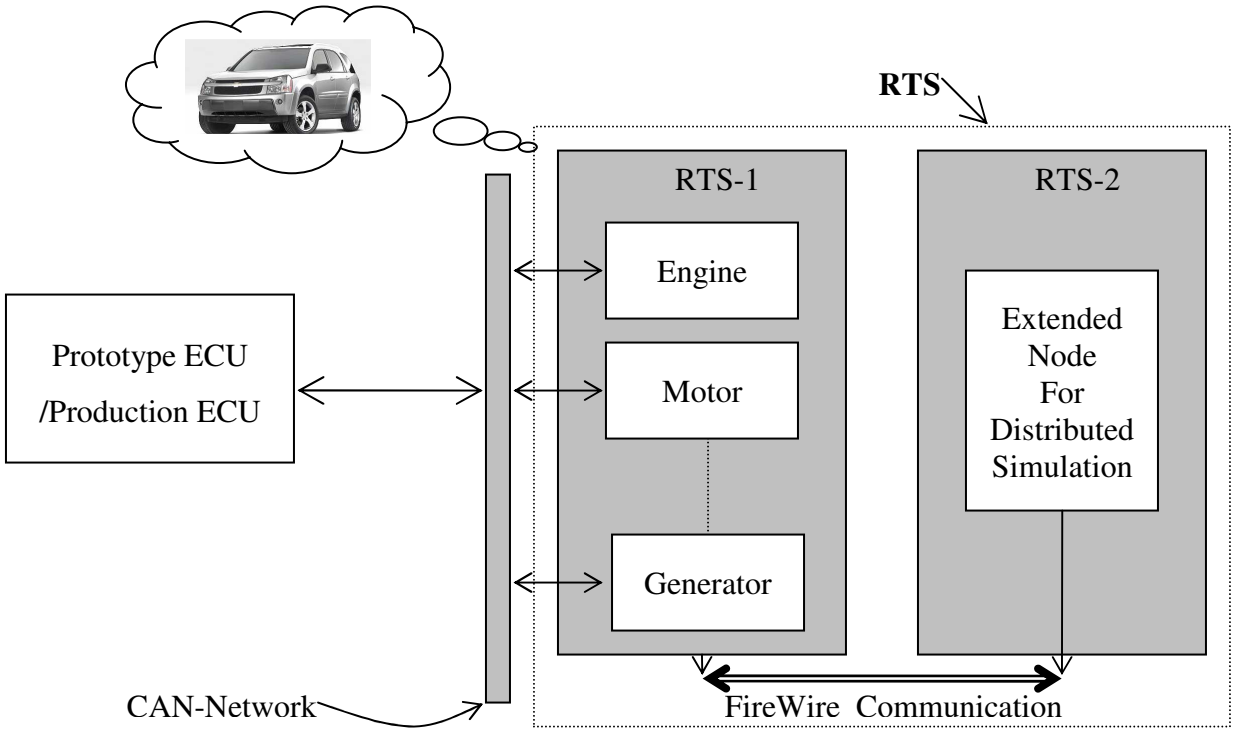


Figure 4.4: Representation of a HIL simulation.

For HIL simulation of a HEV, the SCM is interfaced to the RTS using CAN communication. The CAN interface mimics the in-vehicle communications.

A top level layout of the HIL setup for a vehicle is shown in Figure 4.5. The accelerator and brake pedals give user demands to the controller and the controller sends out the set points for the subsystems in the vehicle. The various signals while simulating a vehicle are captured for visualization and further analysis on the host PC. The various parts used in the HIL setup, their description, their utility and their images are given and discussed in the following sections.

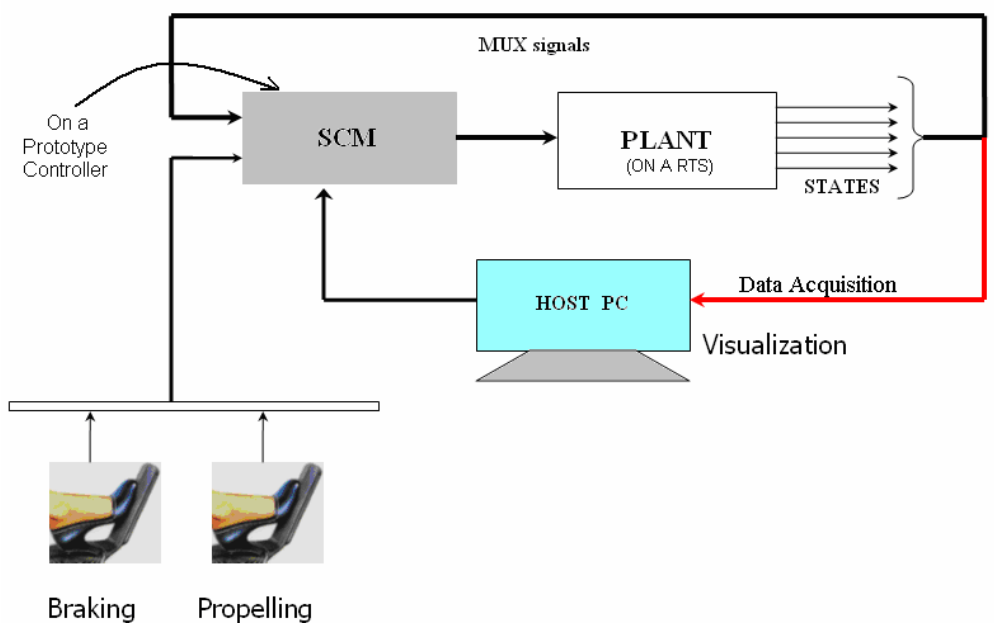


Figure 4.5: Schematic layout for the HIL setup.

4.3.1 Real Time Simulator

The RTS consists of real time computational nodes RTS-1 and RTS-2. In the real time simulation of a HEV, RTS-1 might mimic a controller and RTS-2 might mimic a plant. A

schematic representation showing the hardware layout of the RTS-1 is shown in Figure 4.6. Various I/O capabilities are obtained by using off-the-shelf PCI cards.

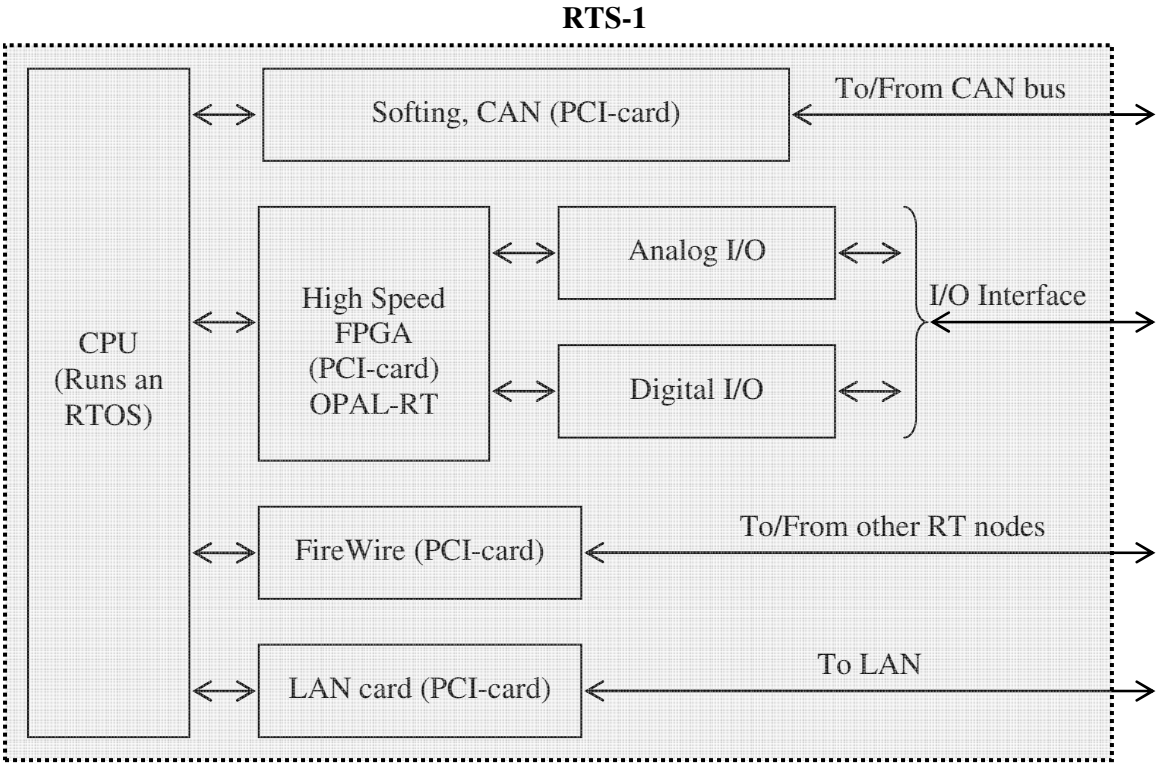


Figure 4.6: Hardware layout of RTS-1.

The code generated from the plant model is downloaded onto RTS-1 to run in real time. RedHawk RTOS is installed on RTS-1 to ensure the real time tasking. RTS-1 is interfaced to the prototyping/real controller either by a CAN bus or by analog and digital I/O. In case the plant model is complex and demands more computation power, the RTS-1 connects to another node, RTS-2, for distributed simulation. RTS-1 and RTS-2, and other real time nodes if necessary, are connected using FireWire communication cables. RTS-1 is connected to a Local Area Network (LAN) for downloading the real time code from a host PC. The Hardware for RTS-1 includes an IBM compatible PC, supplemented

with analog and digital I/O cards from Opal-RT Technologies. The I/O boards have a 10 ns resolution event-detection capability ensuring a high degree of accuracy.

A schematic representation showing the hardware layout of RTS-2 is shown in Figure 4.7. RTS-2 is the additional real time computational node for demonstrating a distributed HIL simulation.

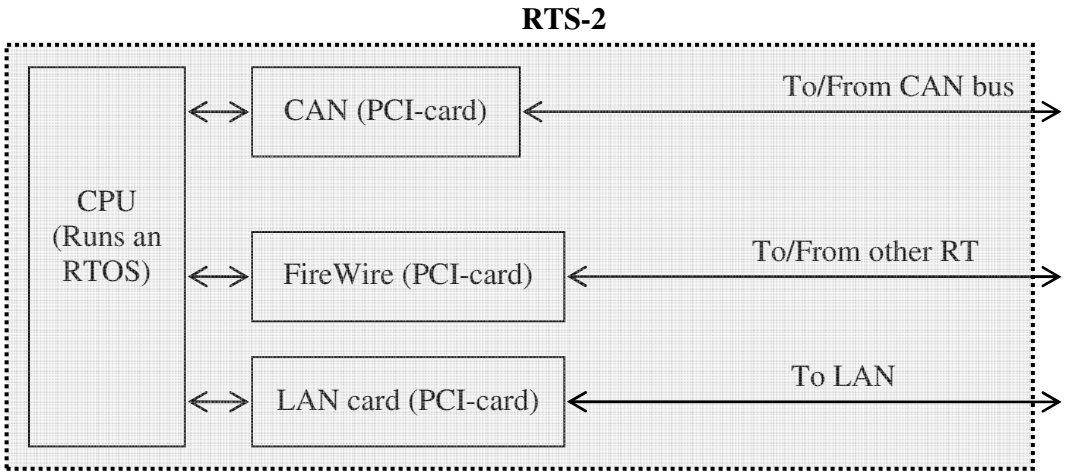


Figure 4.7: Hardware layout of RTS-2.

A part of the computationally intensive subsystem can run on the extended node taking the load off RTS-1. The node also connects to the prototyping or production controller by a CAN bus. RTS-2 is also connected to a LAN for downloading the real time code from a host PC. The images of the RTS built are given in Figures 4.8-4.12.

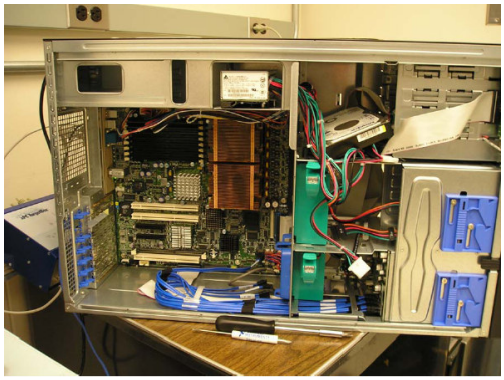


Figure 4.8: Target 1: Dual processor RTS-1.

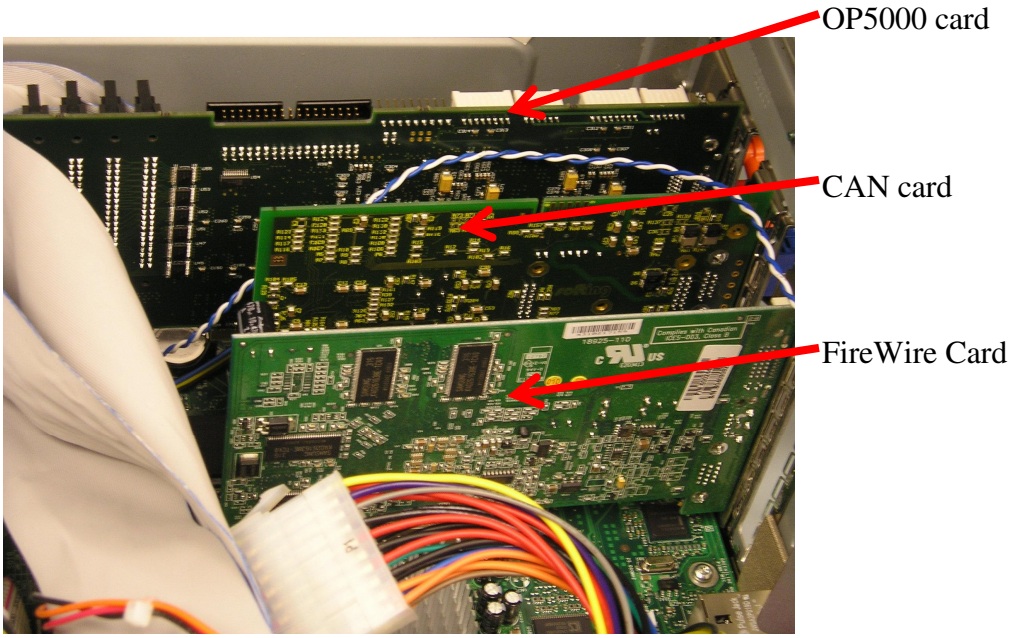


Figure 4.9: RTS-1 with high speed analog and digital I/O cards, CAN card (RTS) and a FireWire card.

Figure 4.10: FireWire (IEEE 1394) cards.

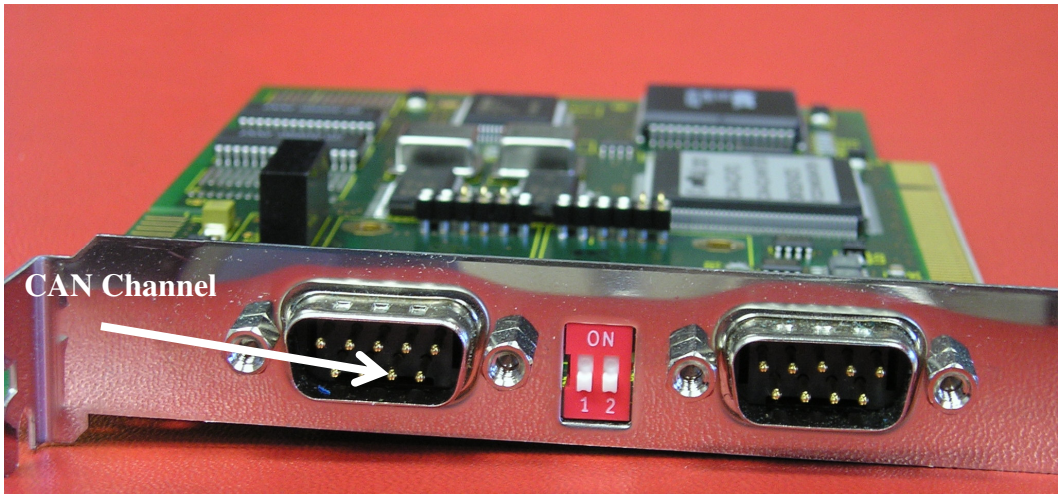


Figure 4.11: Dual channel CAN card from Softing.

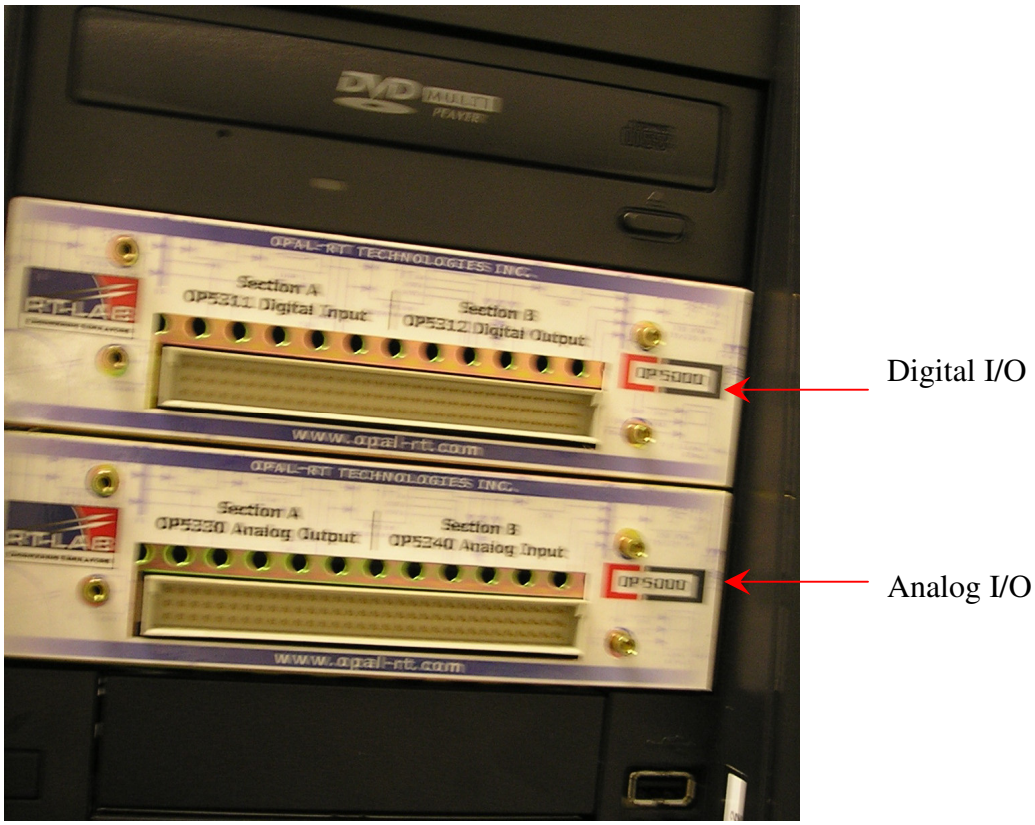


Figure 4.12: Analog and digital I/O interface for RTS-1.

4.3.2 Prototyping Controller

An xPC TargetBox provided by The MathWorks is used as a prototyping controller. A schematic representation showing the hardware layout of the xPC TargetBox is shown in Figure 4.13. The TargetBox consists of various built in I/O capabilities. The real time code for the controller is generated on the host PC and downloaded into an xPC TargetBox through the LAN. The xPC TargetBox has analog and digital I/O, CAN communication ports and a monitor port to view any critical signals and also for visualization. The xPC TargetBox is preferred for the prototyping controller because of its robustness and portability and also its availability for the project. Images of the xPC TargetBox are given in Figures 4.14-4.15. Figure 4.14 shows the xPC TargetBox and an analog I/O screw terminal board. Figure 4.15 shows the entire xPC TargetBox setup.

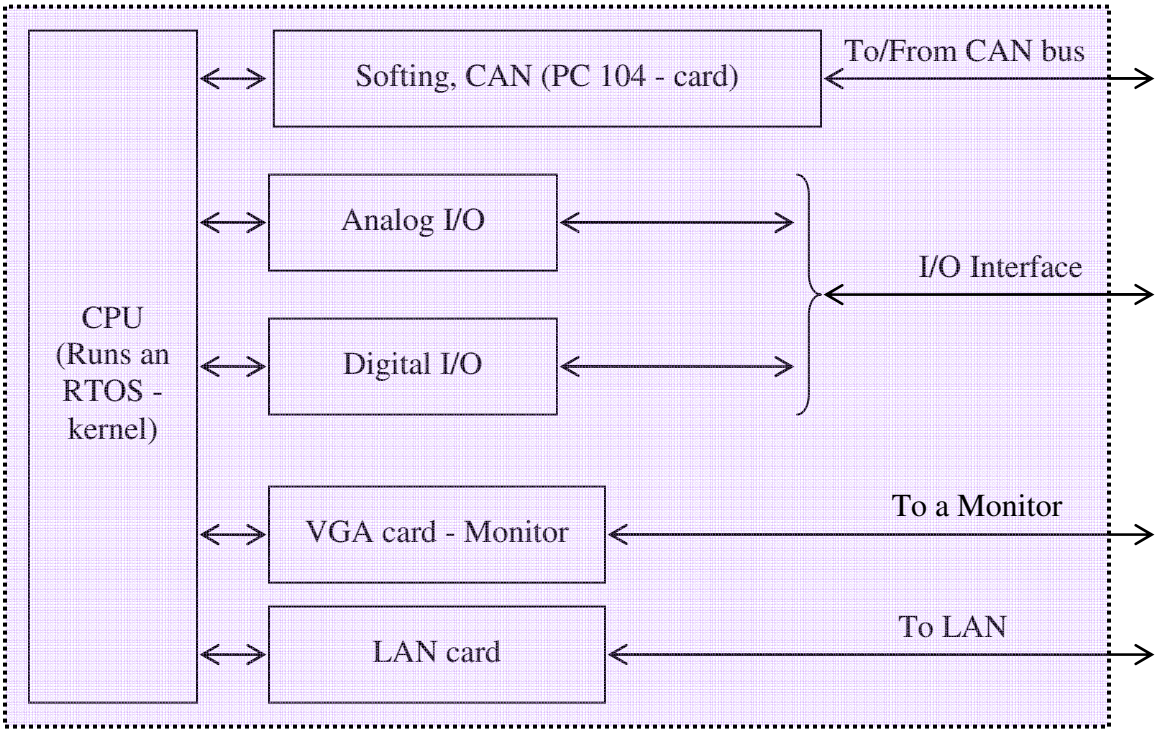


Figure 4.13 Hardware layout of the xPC TargetBox.

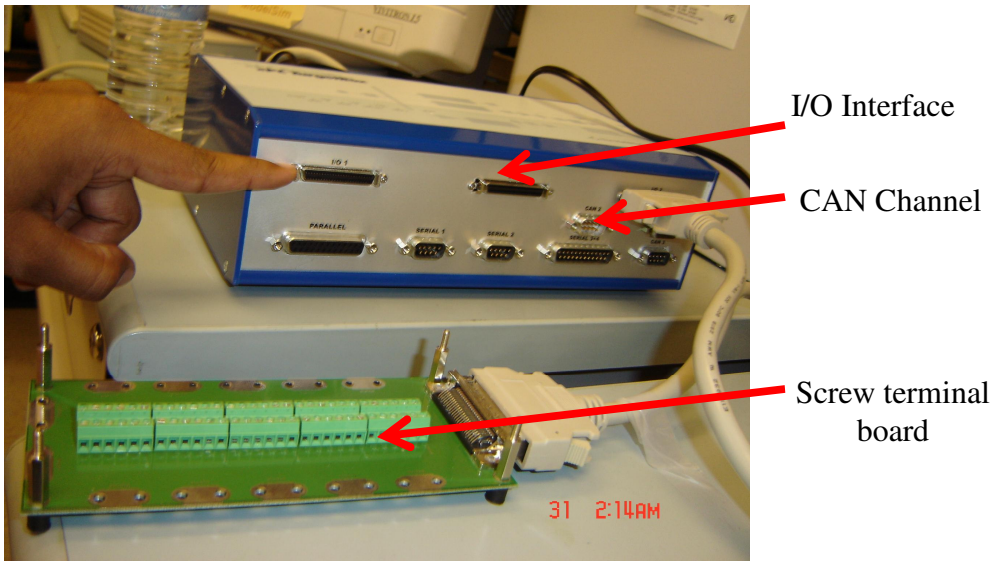


Figure 4.14: xPC TargetBox for controller prototyping.

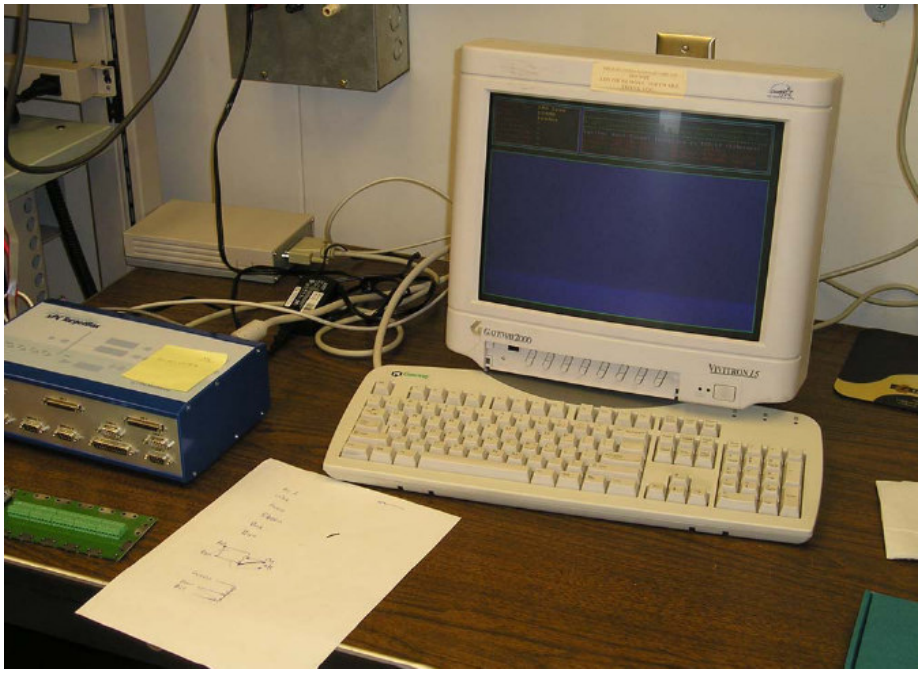


Figure 4.15: Target 2: xPC TargetBox setup including a monitor for displaying xPC scopes.

4.3.3 Host PC: User Console

The host PC is used for initial analysis and offline simulations using Matlab/Simulink and PSAT as discussed in Chapter III. Additionally, it is equipped with real time code generation tools, RTW for obtaining the code running on the xPC TargetBox and RT-LAB for obtaining the real time code running on the RTS. In addition to code generation, the host PC helps to visualize various signals and for data acquisition. It is connected to the LAN, similar to the other nodes in the system. It runs on a Windows XP operating system. The setup also includes two pedal inputs (accelerator and brake) interfaced to the controller to demonstrate a virtual vehicle platform. Figure 4.16 shows the accelerator and brake pedals used.

Figure 4.16: Accelerator and brake pedals.

The layout of the HIL setup on a LAN for a virtual vehicle simulation is shown in Figure 4.17. The image of the overall setup in the research lab is shown in Figure 4.18.

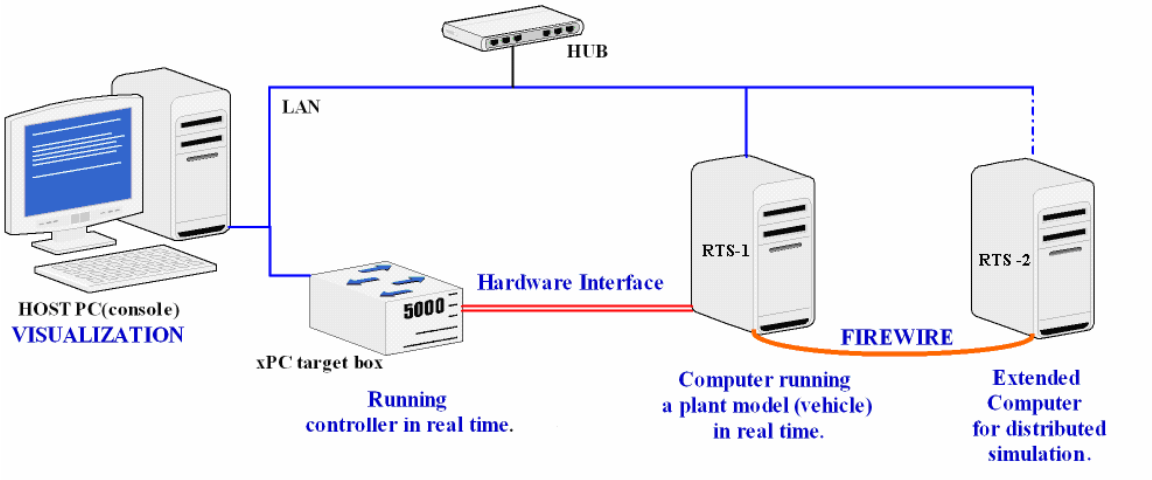


Figure 4.17: Layout of the HIL simulation platform with off-the-shelf computers and xPC TargetBox.7

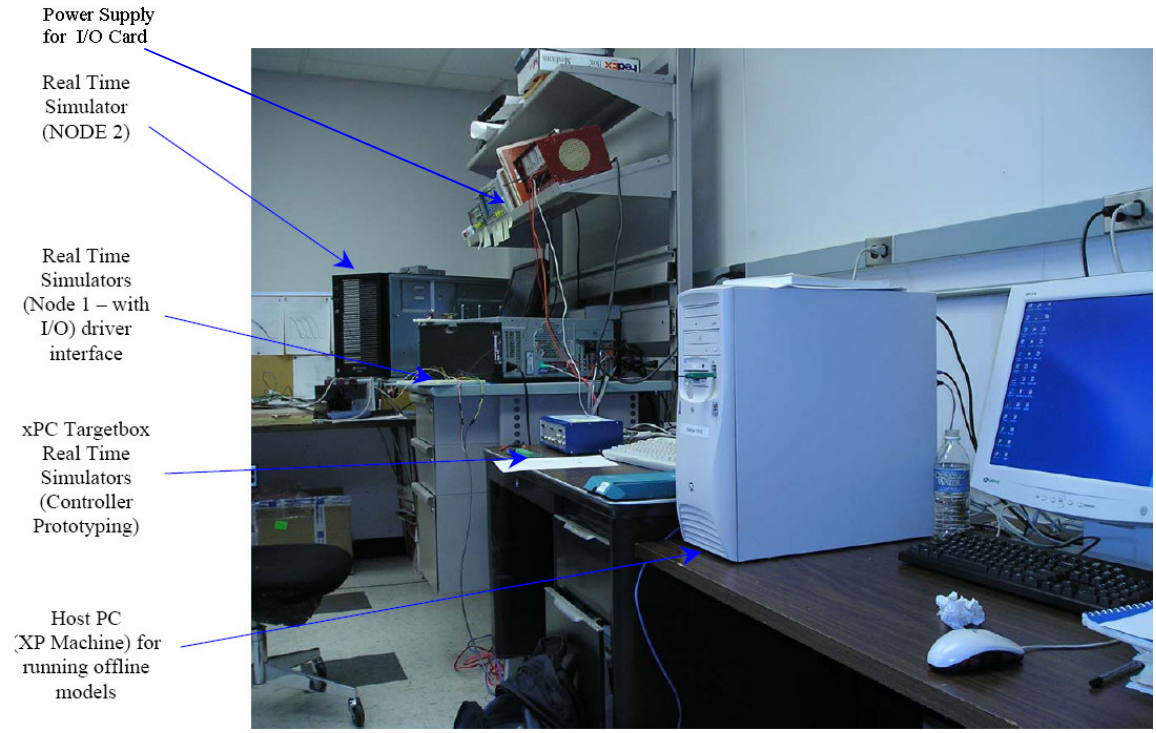


Figure 4.18: Photograph of the entire HIL setup.

The HIL setup was built after careful testing of all the I/O cards, and the components as specified by their manufacturer. Figure 4.18 shows the entire platform.

- List of Hardware used:
  - Computers
    - Host PC: Intel Pentium III, 2 GHz, 512 MB SDRAM.
    - Real Time Simulators
      - RTS-1: Gateway 9510 SATA, Dual Processor, Intel 2.8GHz Xeon with 800MHz FSB and 1MB L2 Cache, 1024 MB RAM.
      - RTS-2: Gateway, Dual Core, Intel 2.4 GHz, 512 MB RAM.
  - xPC TargetBox: The MathWorks.
    - Intel, Pentium III, 700 MHz, 128 MB RAM, 32 MB flash memory (For standalone mode).
    - A standard VGA monitor.
  - OP5000 signal conditioning card: OPAL-RT Technologies.
    - Xilinx Virtex II Pro, Embeddable RAM, 16KB (Expandable to 2 Mbytes), 144 software-configurable I/O lines.
  - 4 port Ethernet HUB (For TCP/IP Networking).
  - CAN cable with termination resistors.
- List of Software used
  - Matlab/Simulink/Real Time Workshop – The MathWorks.
  - RT-LAB 7.2.1 – OPAL-RT Technologies.
  - RedHawk Linux Real Time Operating System: Concurrent Technologies.

4.4 Real Time Simulation of a Series-Parallel 2x2 HEV

The Matlab/Simulink model of the series-parallel 2x2 HEV developed in Chapter III was updated for the HIL simulation. The updated model was run in real time on the HIL setup. The real time code for the plant model was generated using Real Time Workshop (RTW- MathWorks) and RT-LAB (OPAL-RT Technologies).

The vehicle model built using PSAT in Chapter III is updated by grouping the appropriate blocks into Simulink subsystems. A schematic representation of the HEV model with subsystems grouped to facilitate HIL testing is given in Figure 4.19. The plant subsystem includes the HEV and its propulsion systems and is renamed in Simulink as SM_powertrain. The controller subsystem (SCM) is renamed as SS_Controller. The drive cycle input and the buses carrying the feedback signals are taken and grouped as SC_dataacquisition.

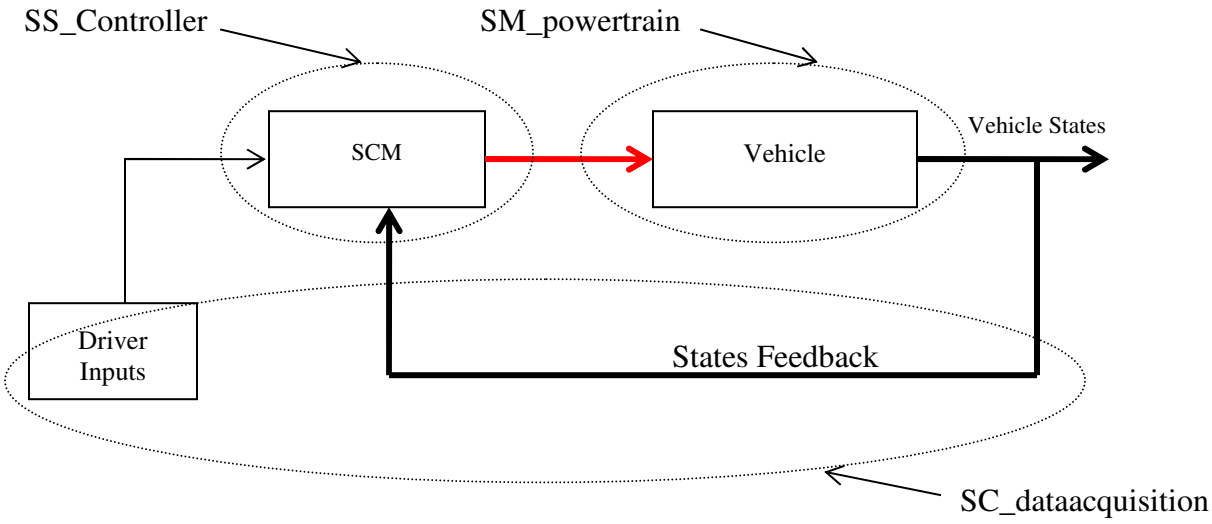


Figure 4.19: Grouping the Simulink subsystems of a HEV model.

RT-LAB requires the Simulink subsystems be grouped as master, slave and console subsystems as shown in Figure 4.20. The description and the reason for reorganizing the Simulink model follows.

SM_powertrain: SM stands for master subsystem. This includes the powertrain model in the context of a HEV. The inputs are the commands, corresponding to the subsystem operating points set by SCM. The outputs are the states of the various subsystems in the powertrain. The inputs and outputs are handled over the CAN bus. With a prefix SM added to the name of the subsystem, RT-LAB automatically recognizes that the subsystem is scheduled to run in real time.

SS_controller: SS stands for slave subsystem. This includes a model of SCM in the context of a HEV. The inputs are the driver commands from the console block and state feedback from the master subsystem. The outputs are the commands corresponding to subsystem operating points set by the SCM. The inputs and outputs are handled over the CAN bus. With a prefix SS added to the name of the subsystem, RT-LAB automatically recognizes that the subsystem is scheduled to run in real time.

SC_dataacquisition: SC stands for console subsystem. The console subsystem captures the various signals for data acquisition and visualization. With a prefix SC added to the name of the subsystem, RT-LAB automatically recognizes that the subsystem is scheduled to run on the host PC.

A schematic representation of the reorganized model, ready to be run on the HIL simulation platform is shown in Figure 4.20. Once the model is reorganized it can be used for HIL simulation and testing.

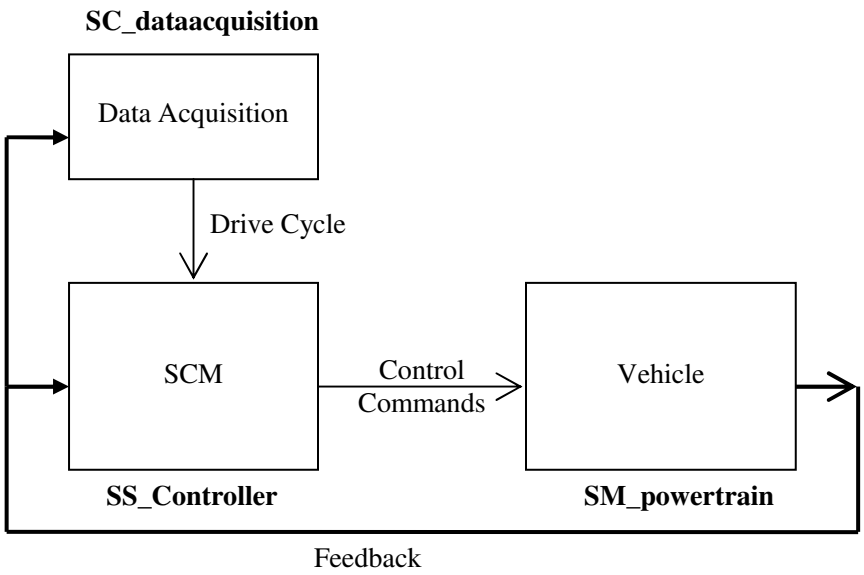


Figure 4.20: Schematic of reorganized Simulink model of a HEV to run in real time.

Three configurations were used in the development of the HIL setup to test different aspects of real time simulation and HIL simulation of a HEV and its subsystems. The layout of various nodes in each of the configurations is given in this section. The configurations were designed to accommodate the needs of the real time simulation of a HEV as explained in Section 2.4.2. The configurations also progress towards the hardware testing of a subsystem using real controller hardware.

Configuration - 1: Using a single node RTS - real time SIL simulation.

In this configuration, the real time code from the controller and plant model run on the same node (RTS-1). The real time code of the controller and plant use the shared memory concept for data exchange. This is made possible by RT-LAB and RedHawk RTOS. A schematic representation is shown in Figure 4.21. This is an intermediate stage between offline and HIL simulation and can be called real time SIL simulation. This configuration

allowed a preliminary check, to ensure that the host PC, RTS-1, and host-target communication would work as expected, before expanding the setup further.

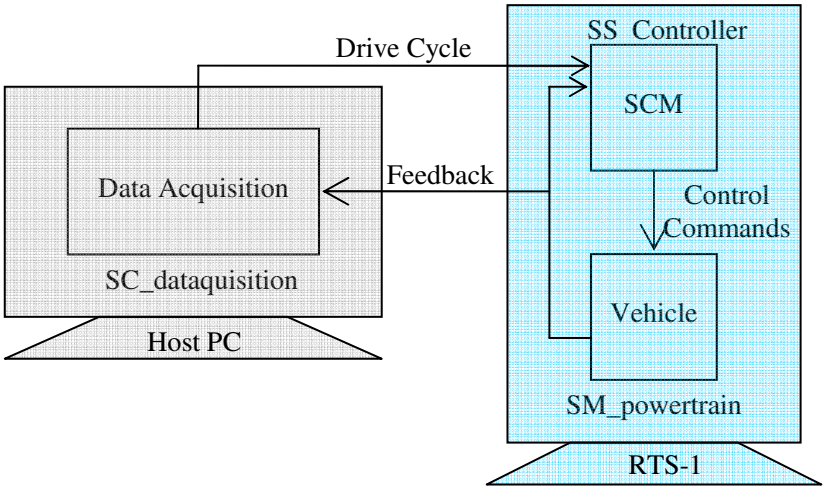


Figure 4.21: Controller and plant run on a single real time node (RTS-1).

Configuration - 2: Using RTS-1 and RTS-2 with a CAN interface

In this configuration, the real time code for the controller runs on RTS-1 and real time code for the plant runs on RTS-2. The data exchange and timer synchronization are managed via a CAN interface. A schematic representation is shown in Figure 4.22.

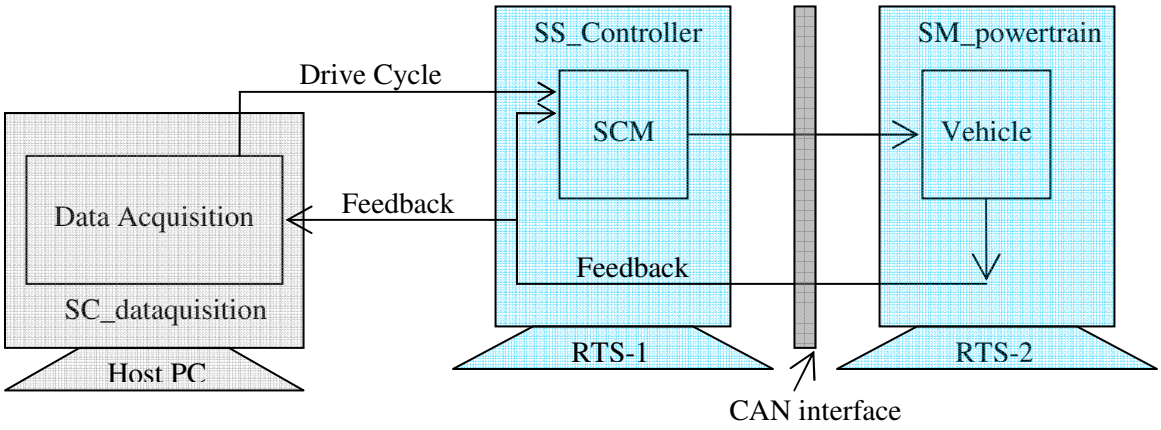


Figure 4.22: Controller runs on RTS-1 and plant runs on RTS-2.

Configuration - 3: Using RTS-1 and a prototype controller with a CAN interface

In this configuration, the real time code generated for the controller runs on an xPC TargetBox and the real time code of the plant runs on RTS-1. The controller and the plant communicate over the CAN bus or using the analog I/O interface. A schematic representation is shown in Figure 4.23.

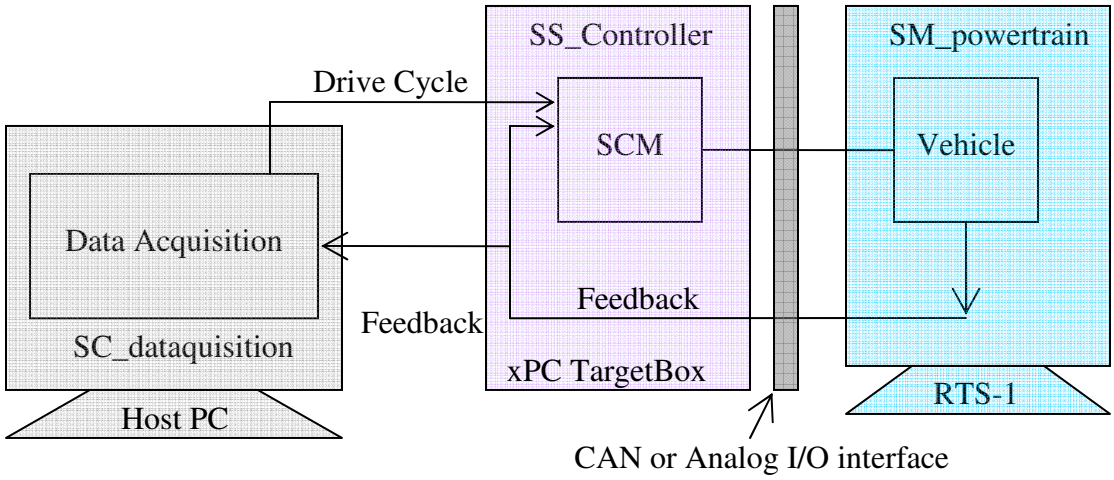


Figure 4.23: Controller runs on xPC TargetBox and plant runs on RTS-1.

For a complex model requiring more computation time, this configuration may be expanded to include an additional real time node (not shown in Figure 4.23). The controller is not concerned with the distributed simulation of the plant since the data comes through a single interface.

Data Acquisition for HIL Simulation of a HEV

The model of HEV updated for real time simulation as discussed earlier in Section 4.4 was run on the HIL setup in Configuration-1 to check the communication works between RTS-1 and the host PC. Once this was ascertained the HEV was run on the HIL setup in Configuration-2. The data thus obtained was saved on the host PC for further analysis

and discussion. The HEV was subjected to various drive cycles representing various driving scenarios, namely highway driving (HWFET), urban driving (UDDS) and acceleration test. The corresponding data from the real time nodes was taken in the form of '.mat' files and saved on the host PC for further analysis. The data also included the real time simulation parameters like overruns and computation time. The data thus obtained were reloaded back to PSAT for reporting purposes. The step wise procedure of downloading the generated code to the real time nodes using RT-LAB GUI and obtaining the data from the real time simulation is explained in Appendix B. The results and further discussion based on the data obtained are given in Chapter V.

4.5 HIL Simulation of an EM Drive

To demonstrate the versatility of using the setup in Configuration-3, a Simulink model of an induction motor drive was used. A schematic representation of the induction motor drive model reorganized in Simulink for HIL simulation is shown in Figure 4.24.

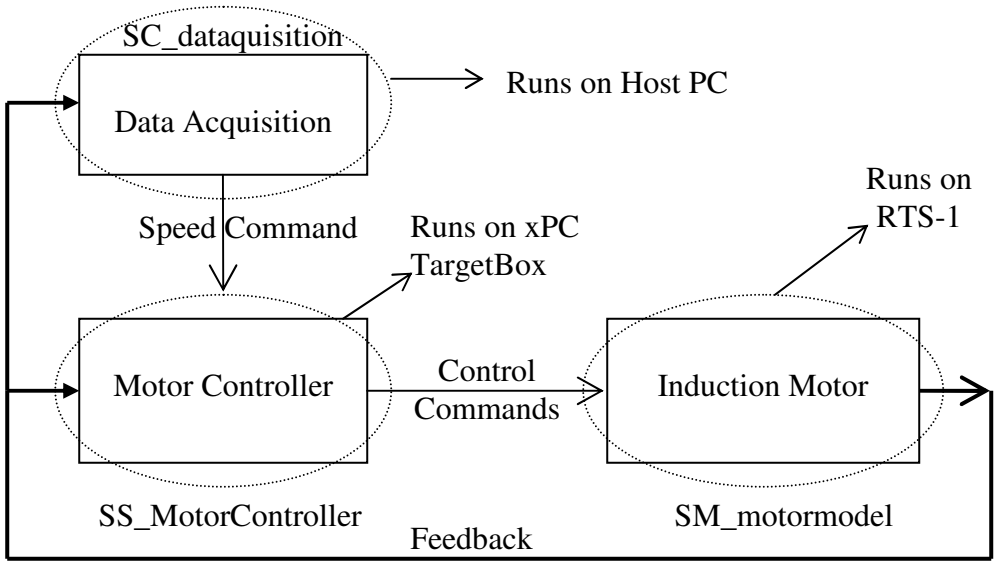


Figure 4.24: Schematic representation of an induction motor drive Simulink model.

The analog I/O interface was used between the xPC TargetBox and the RTS-1. The model of the induction motor is run on the RTS-1 and the motor controller ran in real time on the xPC TargetBox. The response of the motor to the speed command was captured by a cathode ray oscilloscope to see the response from the simulated motor drive. The real time simulation information was saved on the host PC for further analysis. The results and further discussion based on the data obtained are given in Chapter V.

4.6 Conclusion

The HIL simulation setup was successfully built from the ground up and a series-parallel 2x2 HEV model and controller was successfully run in real time on the HIL platform. This chapter demonstrated the procedure for building a configurable HIL setup to test various aspects of real time simulation of a HEV and its subsystems. The offline model of a HEV obtained in Chapter III and an EM are successfully run in real time on the HIL setup. The data from real time simulations is used for the analysis presented in Chapter V.

CHAPTER V

HARDWARE-IN-LOOP IMPLEMENTATION RESULTS

5.1 Results and Analysis

In this chapter the results and analysis of the data obtained from real time simulation of a series-parallel 2x2 HEV on a HIL simulation platform are presented. The comparison of results from HIL simulation of a HEV and offline simulations in Chapter III is presented. The powertrain states were captured and stored in '.mat' files for post processing and comparison. The data thus obtained gave an insight into the differences between the offline and online simulations of the HEV. This also ensured the fidelity of a controller in real time. To demonstrate the capability of the HIL setup a Matlab/Simulink model of an EM Drive was simulated and the results are also presented.

The details pertaining to real time simulation namely computation time and overruns were obtained using a RT-LAB blockset in Simulink. These details are important and give an insight into the processor time being used and also confirm that the RTS performs as expected and mimics the Simulink model of the plant in real time. The HEV model was run in real time on the HWFET, UDDS and Acceleration Test drive cycles.

5.2 HWFET Cycle: Highway Driving

The results of the HIL simulation of a series-parallel 2x2 HEV on HWFET is presented in this section. The HIL setup was used in Configuration-2 as explained in Section 4.4 to perform the real time simulation and the data from the real time nodes was obtained. The plots of vehicle speed (online) and speed demands are shown in Figure 5.1. The plots of engine torque, EM torque, generator torque, and battery SOC are given in Figures 5.2-5.5. It can be seen that the vehicle speed follows closely with the driver demand schedule. The highway cycle is mostly at cruise speeds and hence there is minimal involvement of the EM and generator. It can be seen from Figure 5.3 that the EM is used extensively in the early part of the drive cycle (0-20 sec) indicating the assist action of EM. The generator is barely used in this drive cycle as can be seen from Figure 5.4 corroborated by the fact that there is hardly any oscillation in the battery SOC as seen in Figure 5.5. It can be seen that the final SOC of the battery in Figure 5.5 is near to the initial SOC demonstrating the charge sustaining feature of the HEV.

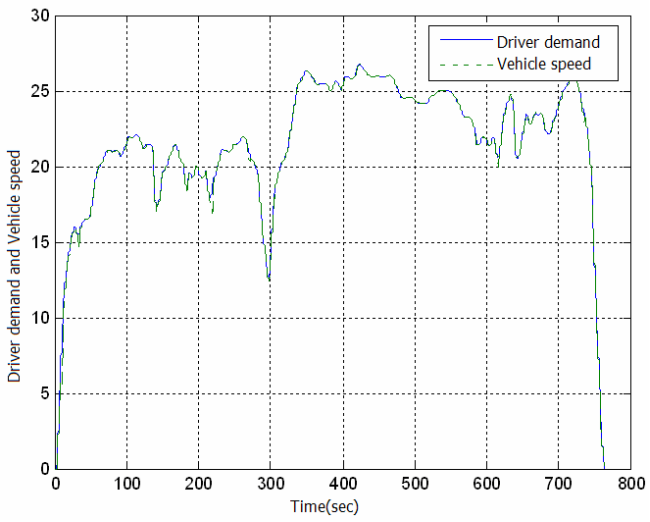


Figure 5.1: Driver demand and vehicle speed from real time simulation for a HWFET cycle.

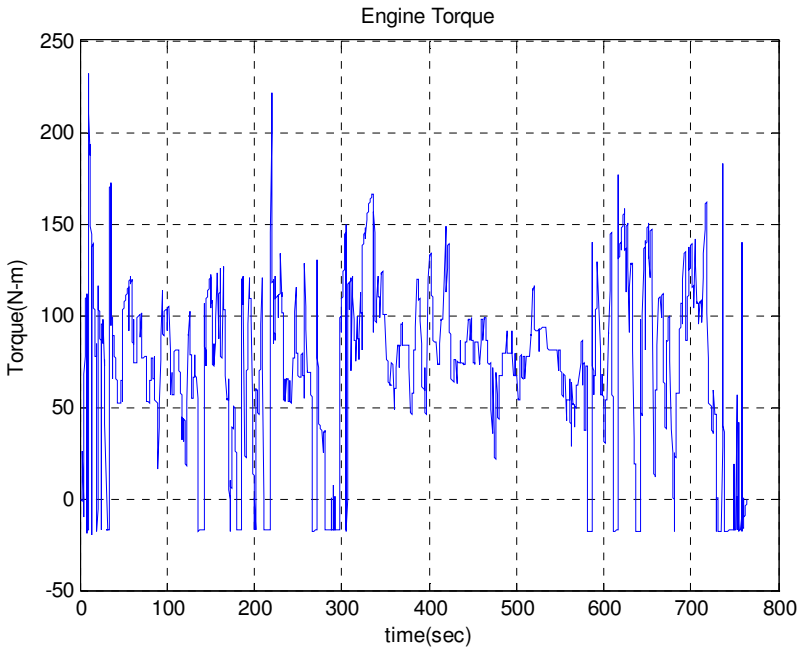


Figure 5.2: Engine torque (N-m) from real time simulation for a HWFET cycle.

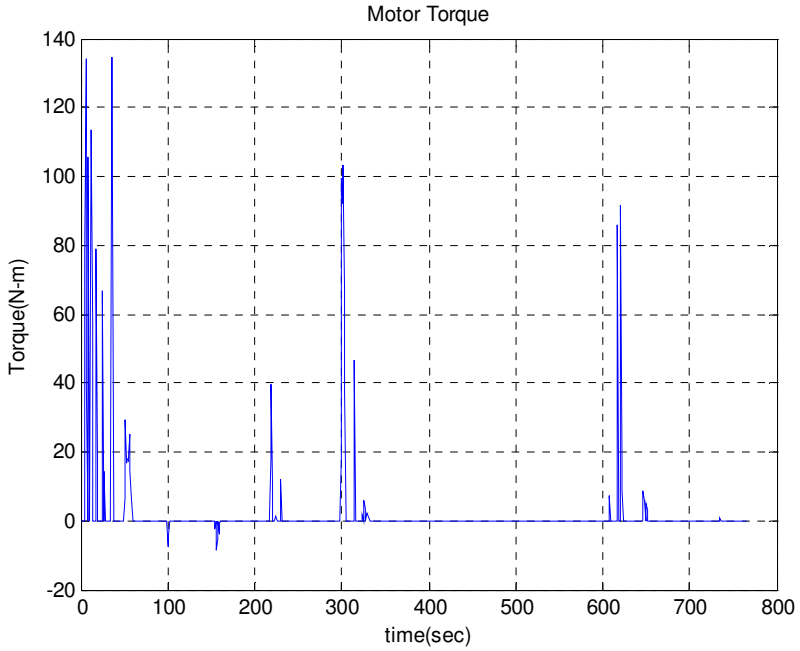


Figure 5.3: EM torque (N-m) from real time simulation for a HWFET cycle.

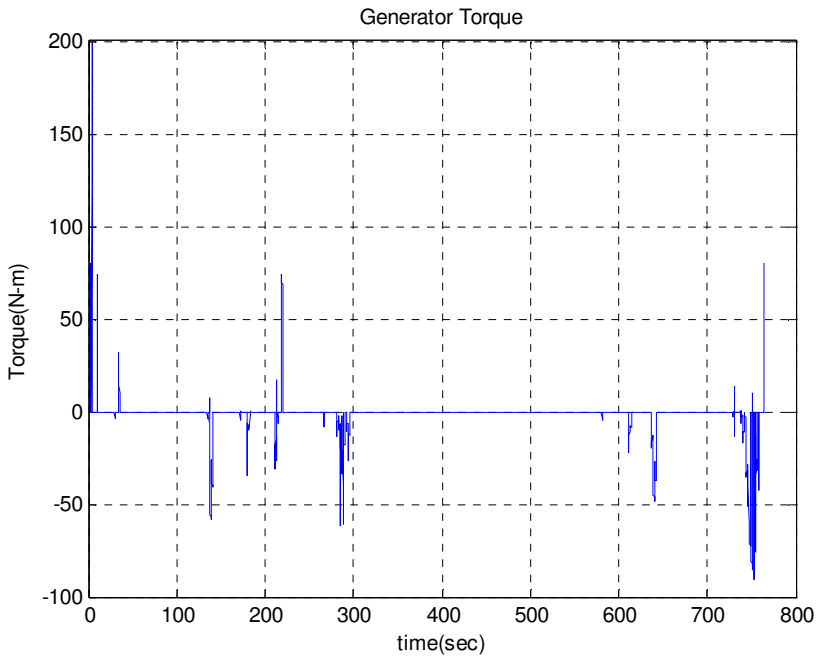


Figure 5.4: Generator torque (N-m) from real time simulation for a HWFET cycle.

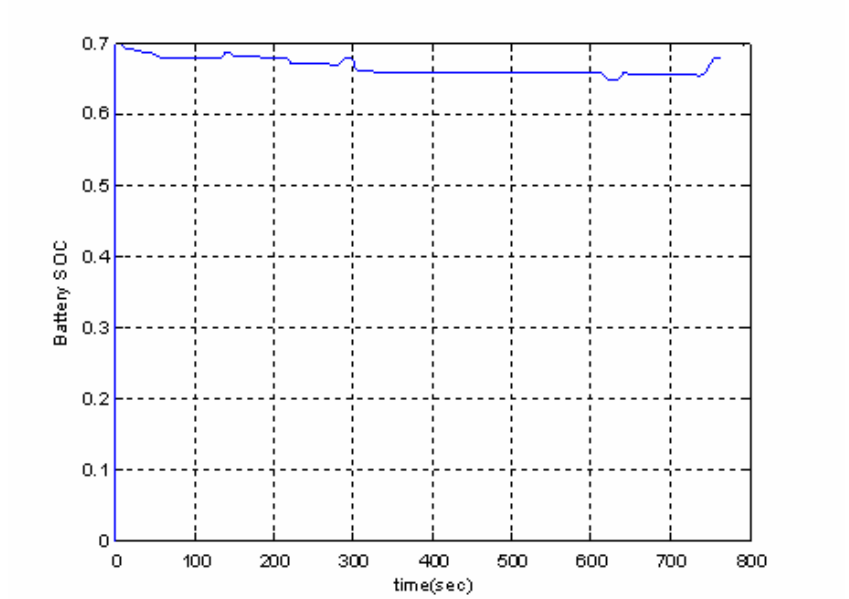


Figure 5.5: Battery SOC from real time simulation for a HWFET cycle.

5.2.1 Real Time Simulation Information

The plots of number of overruns, computation time and step size are given in Figures 5.6-5.7. It can be seen that the number of overruns is zero indicating that the code generated from the Simulink model of a HEV is running in real time; that is, the effective total time taken (TET+TC) is always less than the fixed step size. This is a requirement of real time simulation as discussed in Section 4.1.

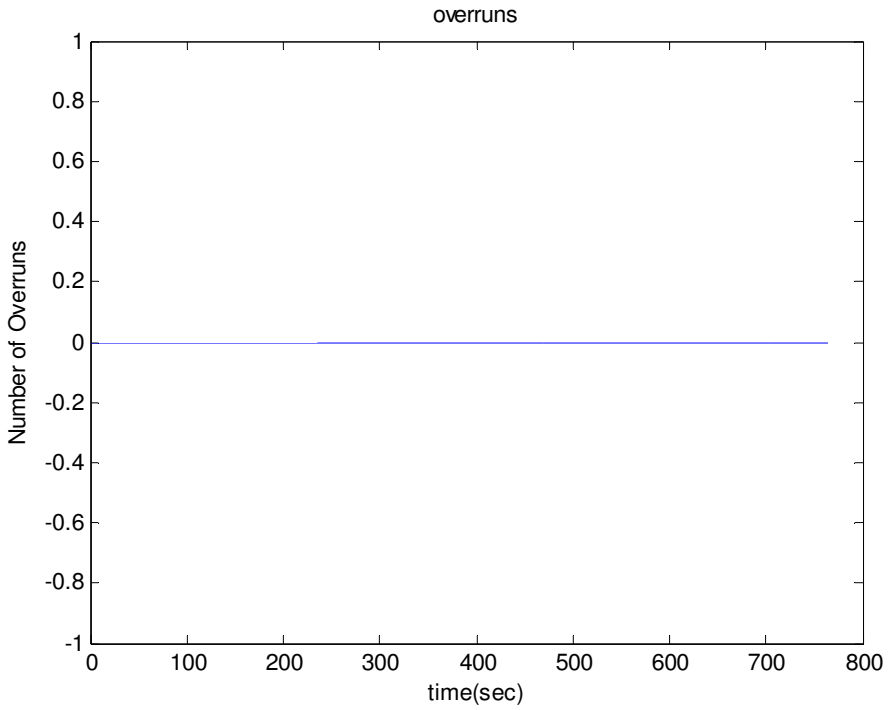


Figure 5.6: Number of overruns from real time simulation for a HWFET cycle.

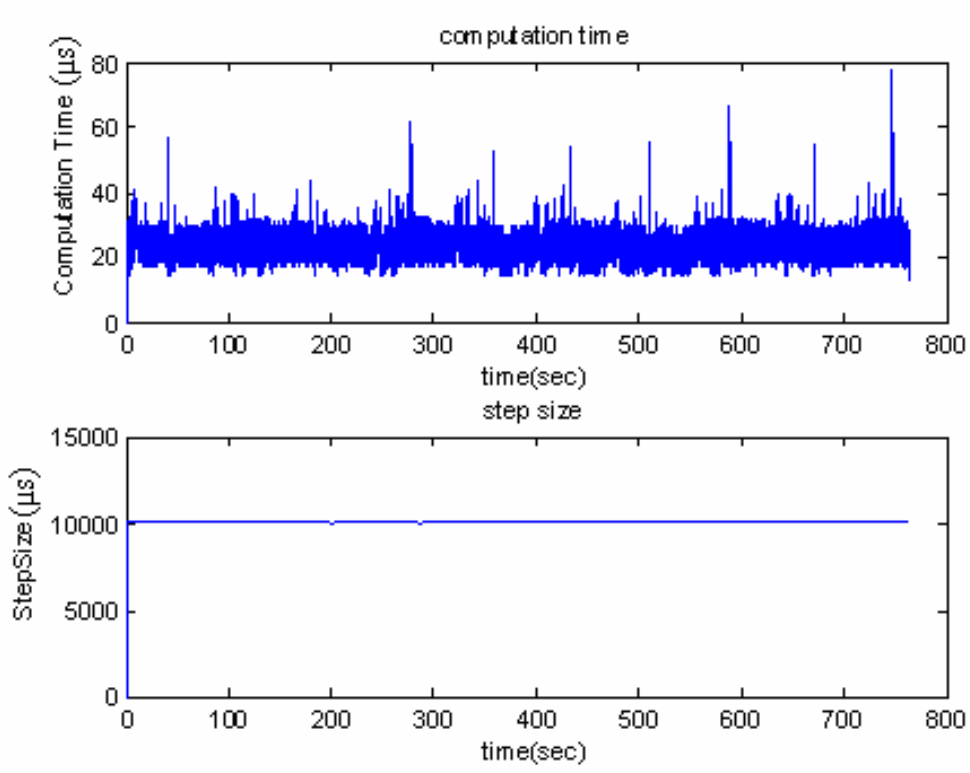


Figure 5.7: Computation time and step size from real time simulation for a HWFET cycle.

5.2.2 Comparison

The results of the offline and online simulations appear to be matching closely as can be seen in Figure 5.8. The real time fuel economy was close to the one from offline simulation. However careful observations show that there is a difference in the response of the vehicle speed for the same drive cycle input in the two simulations. Closer comparisons for several intervals in the simulations are shown in Figures 5.9-5.13. The variations are very minute and are acceptable as a limitation of the HIL setup.

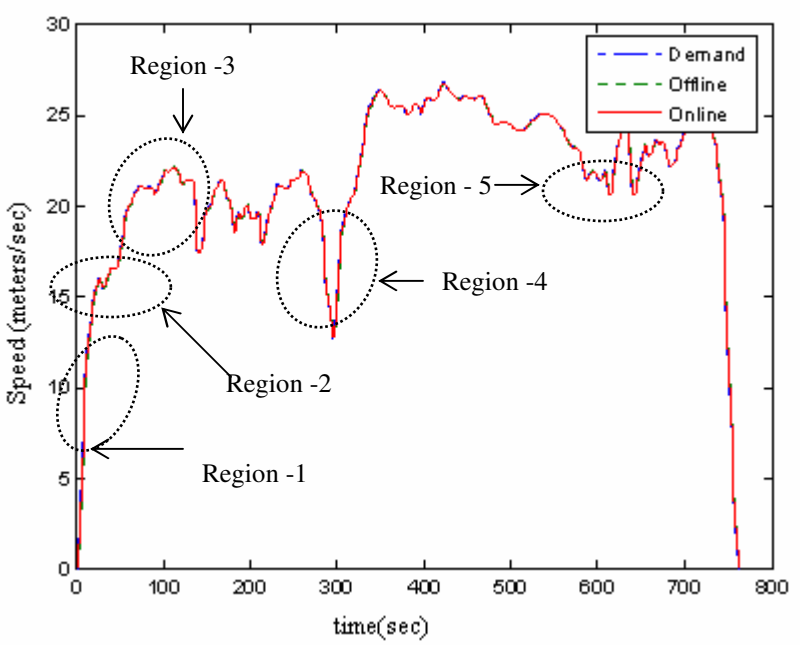


Figure 5.8: Comparison of vehicle speed from offline and real time simulations with HWFET speed profile.

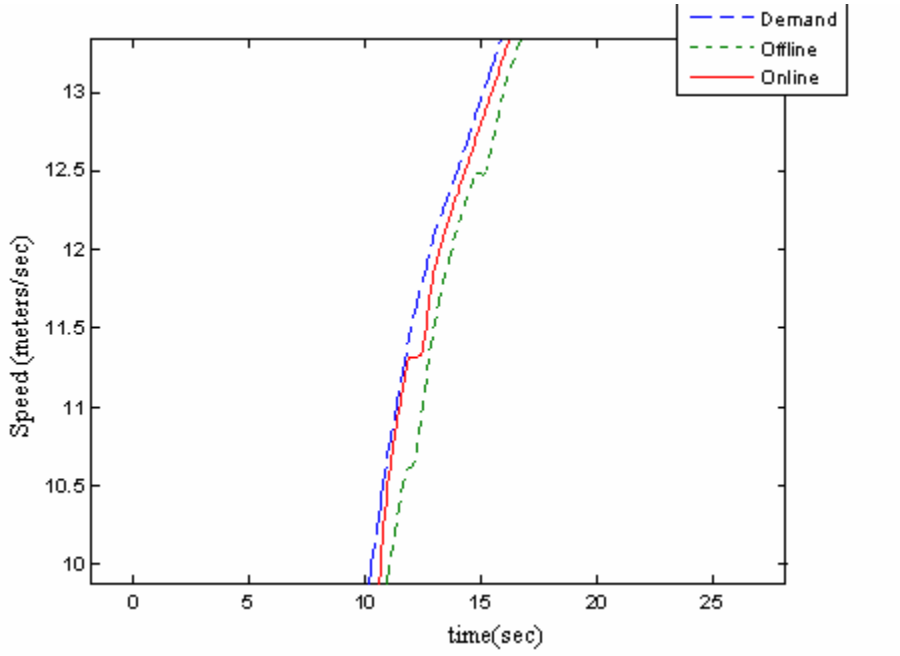


Figure 5.9: (Region 1) Comparison of vehicle speed from offline and real time simulations with HWFET speed profile.

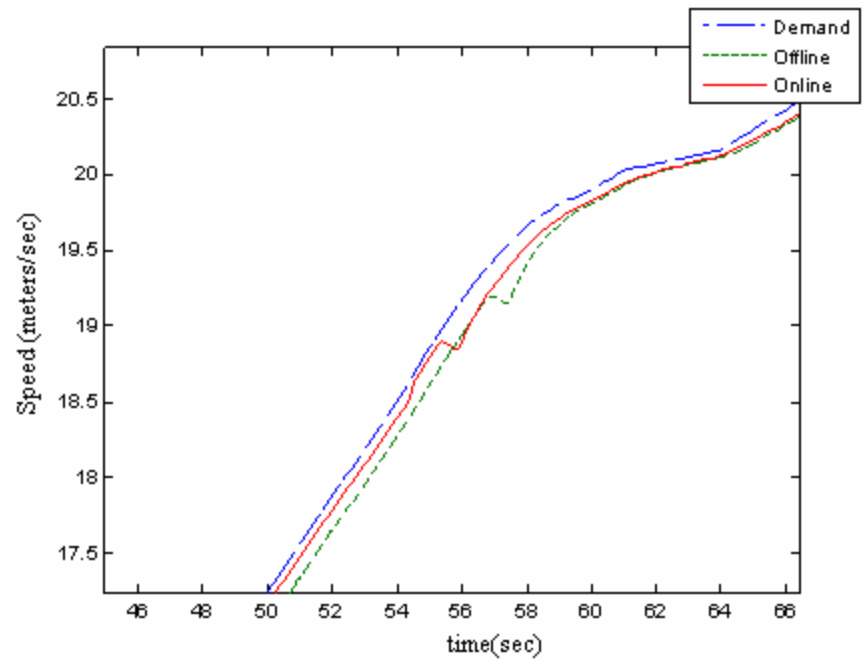


Figure 5.10: (Region 2) Comparison of vehicle speed from offline and real time simulations with HWFET speed profile.

These differences are attributed to changes in the simulated dynamics of the plant model that are caused due to

- A/D and D/A conversions: When the models are running offline on a single node there are no signal conversions. In a HIL setup there are signal conversions from analog to digital and vice versa and cause a change in the dynamics of the vehicle model simulation and hence minor differences between the offline and HIL simulations are seen. This can be clearly seen in Figures 5.9-5.13.
- I/O delays: I/O operations require an additional computation time that is very dependent on the loading of the processor on a RTS.
- CAN delays: There are inherent CAN delays for transmitting data over the bus. The differences are also due to the CAN loop incorporated, which is slow compared to the inter-task communication when running on a single node.

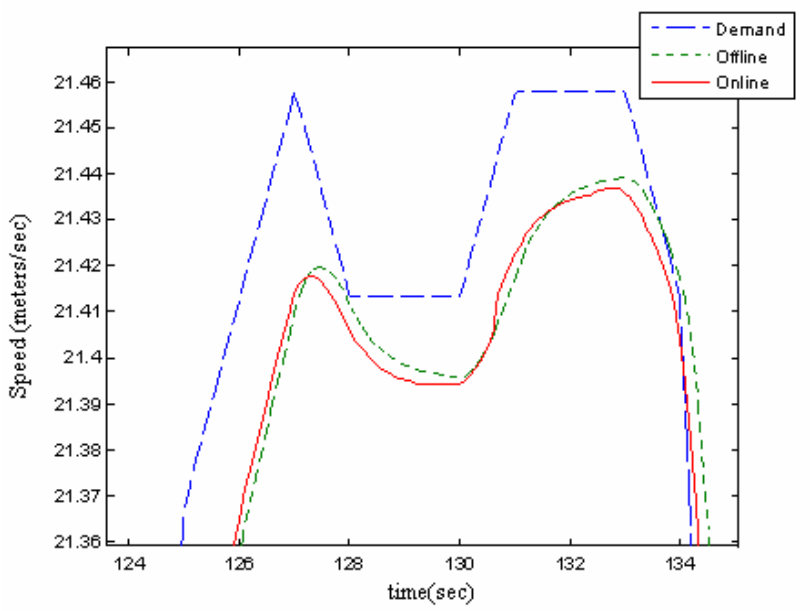


Figure 5.11: (Region 3) Comparison of both offline and real time simulations of vehicle speed with HWFET speed profile.

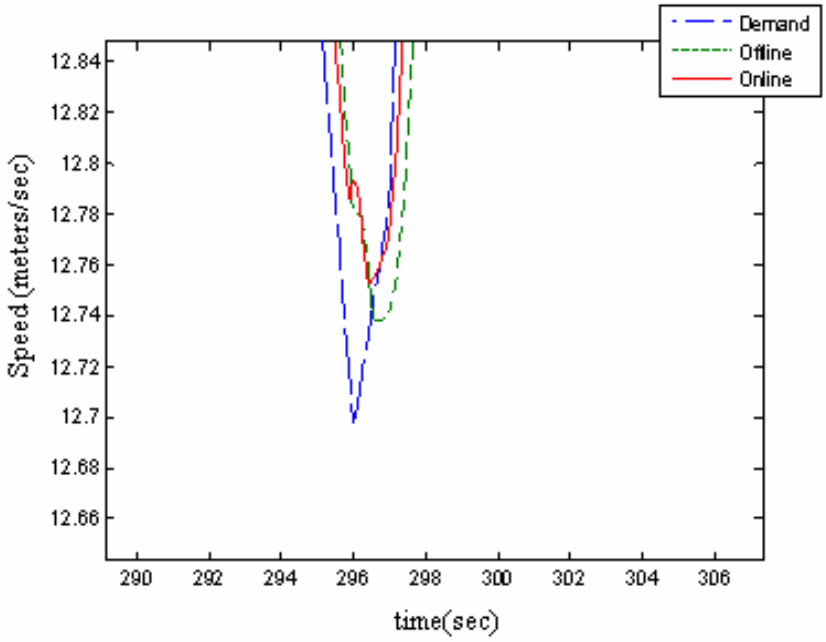


Figure 5.12: (Region 4) Comparison of both offline and real time simulations of vehicle speed with HWFET speed profile.

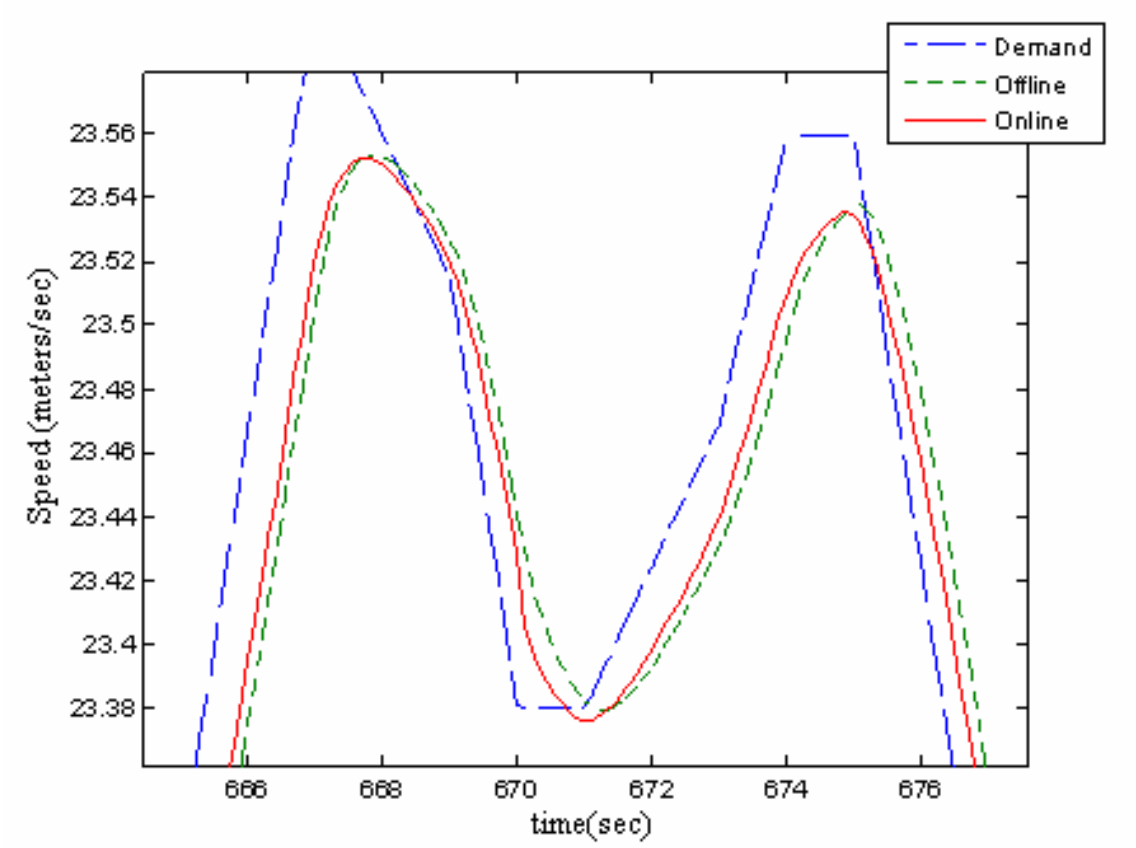


Figure 5.13: (Region 5) Comparison of both offline and real time simulations of vehicle speed with HWFET speed profile.

5.3 UDDS Cycle: Urban Driving

The results of the HIL simulation of a series-parallel 2x2 HEV on UDDS is presented in this section. The HIL setup was used in Configuration-2 to perform the real time simulation and the data from the real time nodes was obtained. The plots of speed demand and vehicle speed for offline and online simulations are shown in Figure 5.14(a) and 5.14(b). The plots of engine torque, EM torque, generator torque, and battery SOC are given in Figures 5.15-5.18.

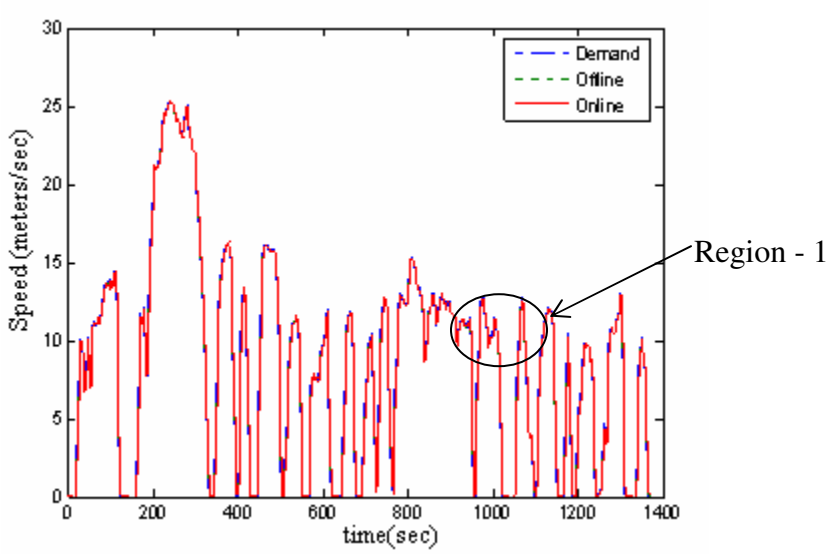


Figure 5.14(a): Comparison of both offline and real time simulations of vehicle speed with UDDS speed profile.

The results of the offline and online simulations can be seen to be matching closely in Figure 5.14(a). The real time fuel economy was close to the one from offline simulation. A closer view of the two simulations can be seen in Figure 5.14(b), which shows that there is a difference in response of the vehicle speed for the same drive cycle input. The difference is similar to the observations seen in the response on a HWFET cycle. Figure 5.16 shows that there is plenty of regeneration from the EM evident from the stop and go

nature of the UDDS drive cycle. It can be seen that the final battery SOC in Figure 5.18 is near to the initial SOC demonstrating the charge sustaining feature of a HEV.

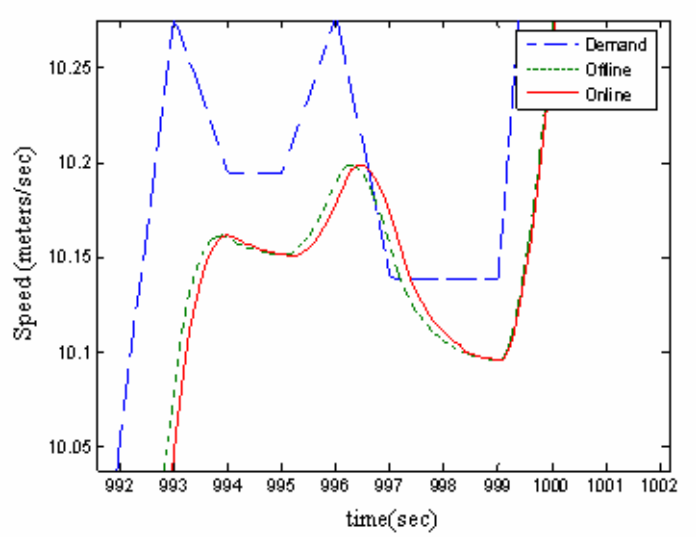


Figure 5.14(b): (Region 1) Comparison of both offline and real time simulations of vehicle speed with UDDS speed profile.

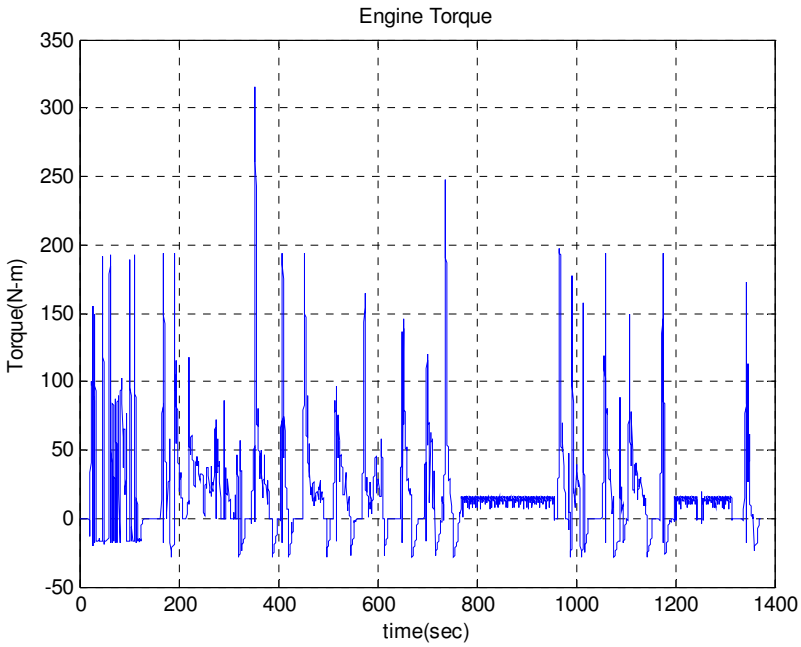


Figure 5.15: Engine torque (N-m) from real time simulation for a UDDS cycle.

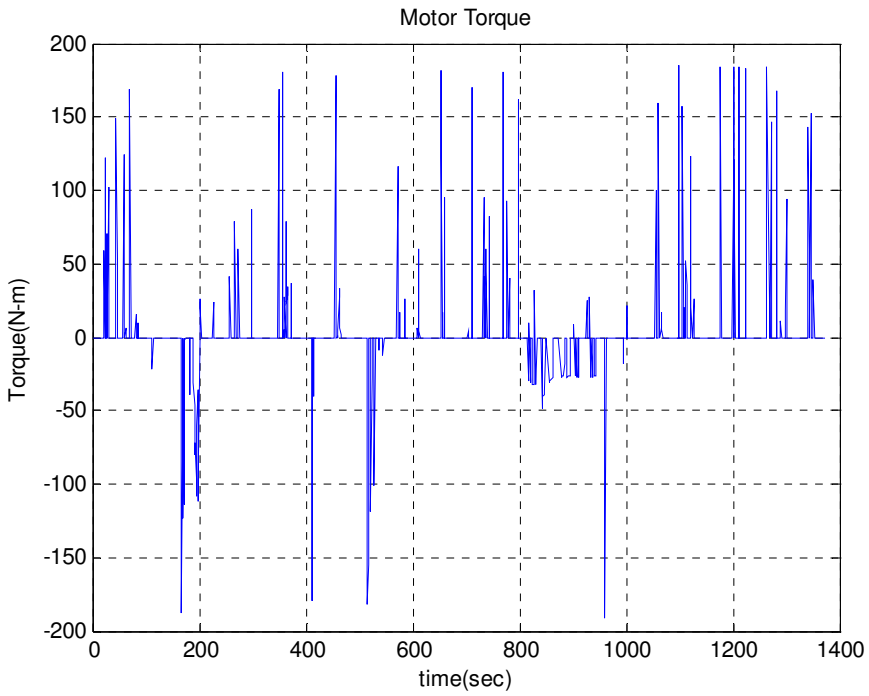


Figure 5.16: EM torque (N-m) from real time simulation for a UDDS cycle.

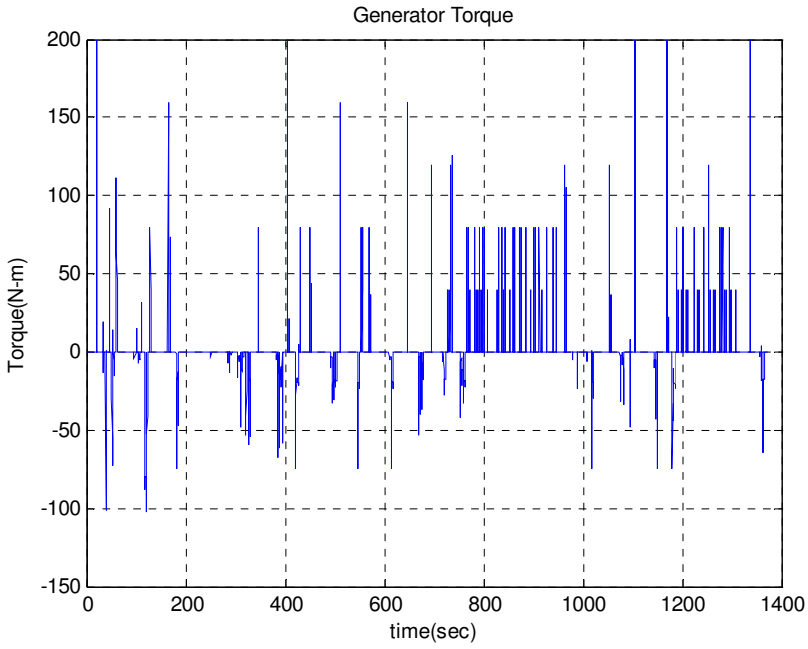


Figure 5.17: Generator torque (N-m) from real time simulation for a UDDS cycle.

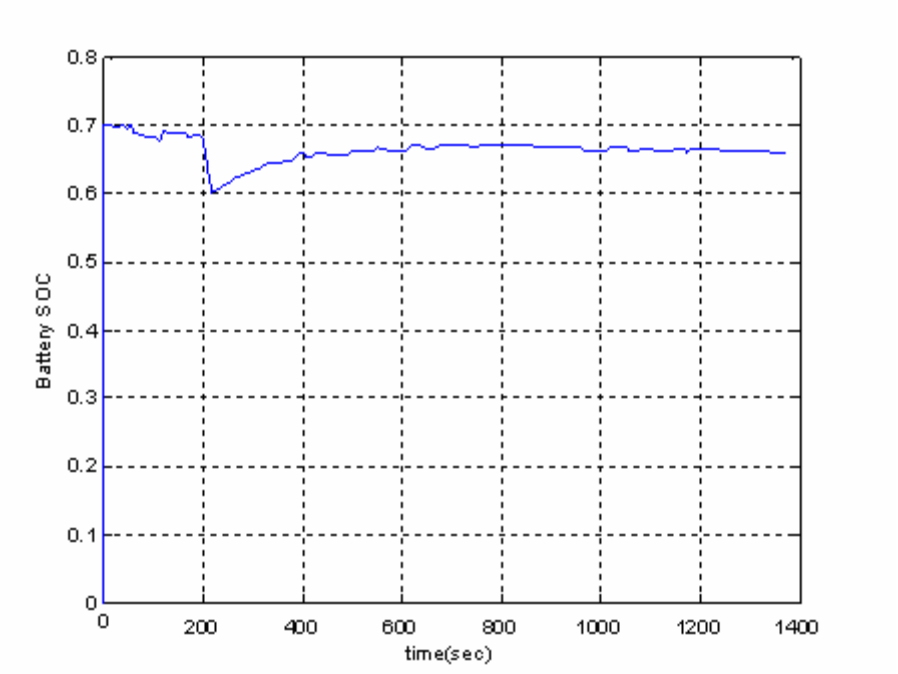


Figure 5.18: Battery SOC from real time simulation for a UDDS cycle.

Real Time Simulation Information (UDDS)

The plots of number of overruns, computation time and step size for a real time simulation of HEV on a UDDS cycle are given in Figures 5.19-5.20. It can be seen that the number of overruns is zero indicating that the code generated from simulink model of the HEV is running in real time; that is, the effective total time taken (TET+TC) is always less than the fixed step size. This is a requirement of real time simulation as discussed in Section 4.1.

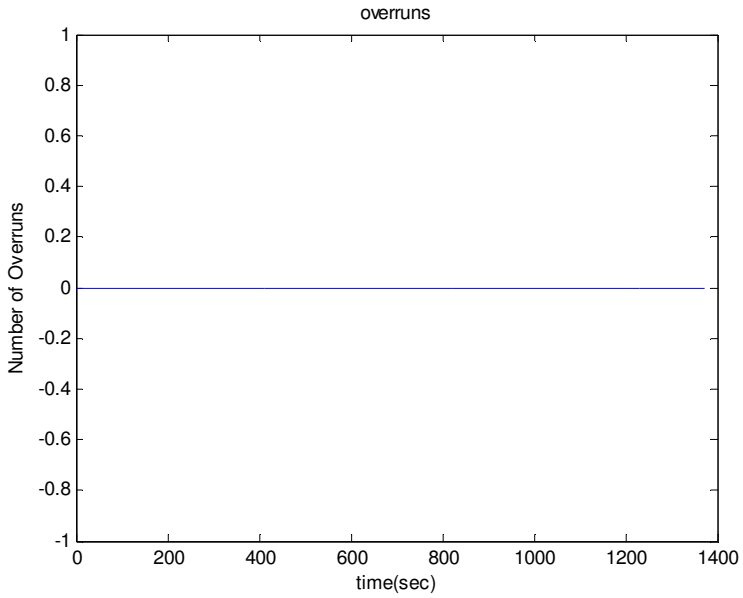


Figure 5.19: Number of overruns from real time simulation for a UDDS cycle.

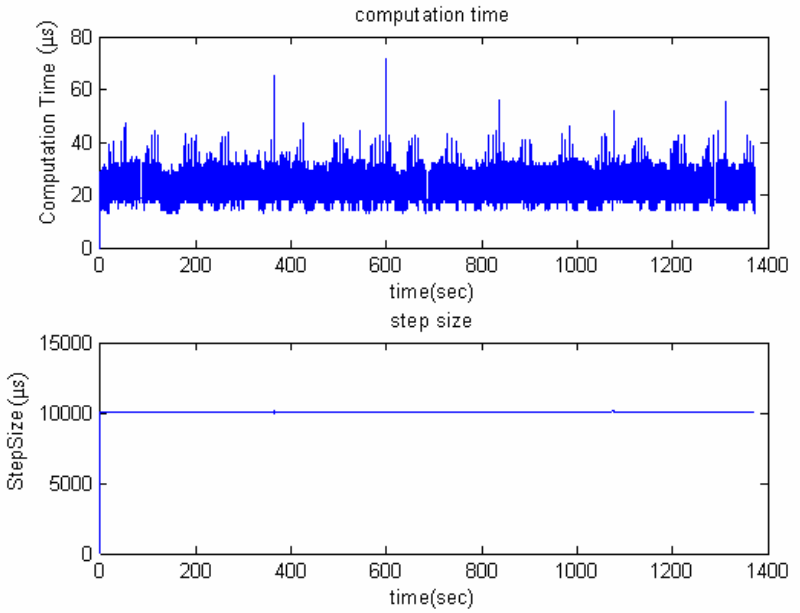


Figure 5.20: Computation time and step size from real time simulation for a UDDS cycle.

5.4 Acceleration Test (0-60 mph)

The results of the HIL simulation of a series-parallel 2x2 HEV on UDDS is presented in this section. The HIL setup was used in Configuration-2 to perform the real time simulation and the data from the real time nodes was obtained. The plots of speed demand, offline and real time vehicle speed are given in Figure 5.21. It can be seen that the vehicle speed in both offline and online simulations match closely. The 0-60 mph acceleration time is around 8.5 sec, which is not much greater than the specification of 8.0 sec given in Section 3.2.

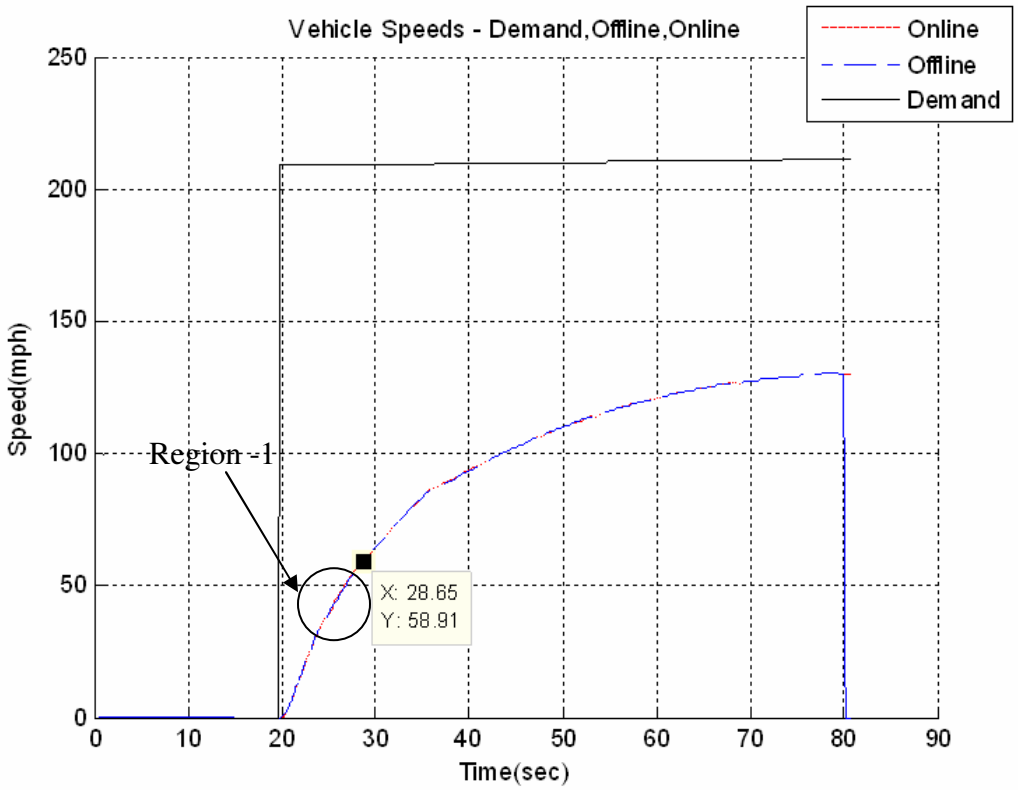


Figure 5.21: Comparison of both offline and real time simulations of vehicle speed with speed profile for Acceleration test.

A closer view of the plots, given in Figure 5.22, shows a slight difference between the offline and real time simulation results. This difference may be attributed to change in

simulated dynamics of the plant due to I/O delays and CAN delays. The difference is very negligible and confirms that the prototyping controller performs as expected. A production controller can be interfaced to the CAN bus and is reserved for a future study. The engine torque, the EM torque, the generator torque, and the battery SOC are given in Figures 5.23-5.26.

Figure 5.22: (Region 1) Comparison of both offline and real time simulations of vehicle speed with speed profile for an Acceleration test.

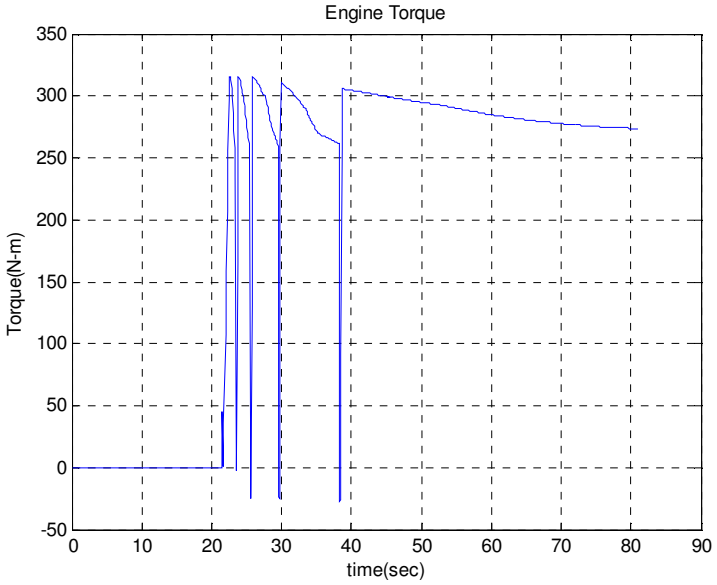


Figure 5.23: Engine torque (N-m) from real time simulation for an Acceleration test.

It can be seen from Figures 5.23 and 5.24 that the output torque of the EM is zero after nearly 30 sec run time. This is because the battery SOC reaches the lower threshold of 0.6 and hence only the engine provides the demanded torque.

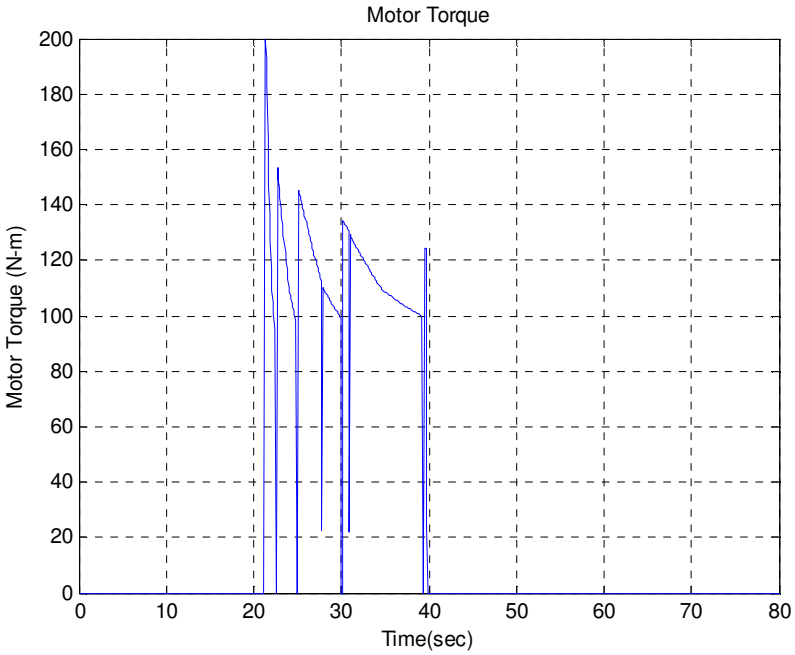


Figure 5.24: EM torque (N-m) from real time simulation for an Acceleration test.

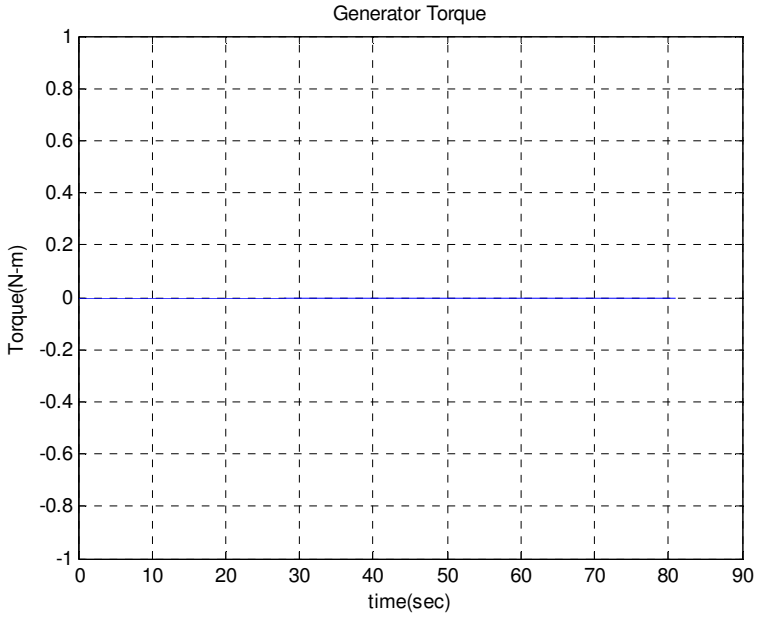


Figure 5.25: Generator torque (N-m) from real time simulation for an Acceleration test.

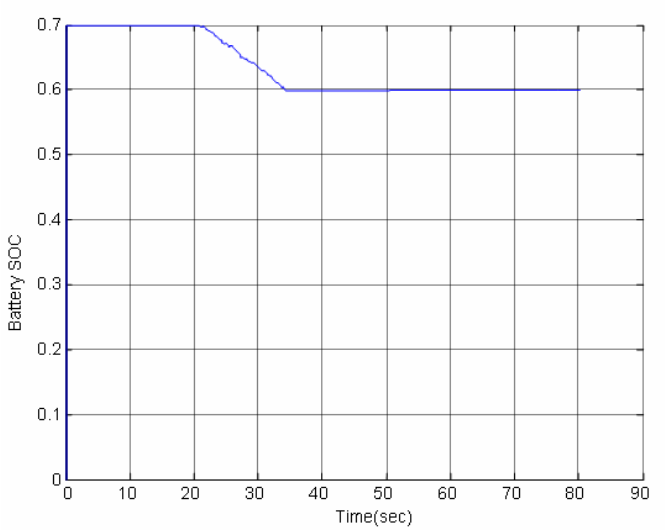


Figure 5.26: Battery SOC from real time simulation for an Acceleration test.

Real Time Simulation Information (Acceleration Test)

The plots of the number of overruns, computation time and step size are given in Figures 5.27-5.28. It can be seen that the number of overruns is zero indicating that the code generated from simulink model of the HEV is running in real time; that is, the effective total time taken (TET+TC) is always less than the fixed step size. This is a requirement of real time simulation as discussed in Section 4.1.

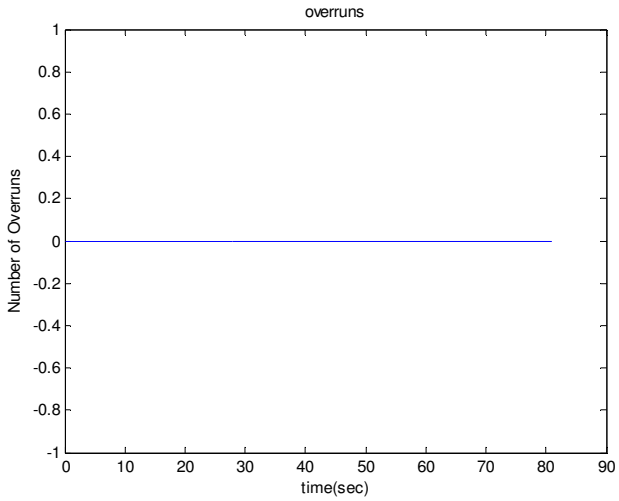


Figure 5.27: Number of overruns from real time simulation for an Acceleration test.

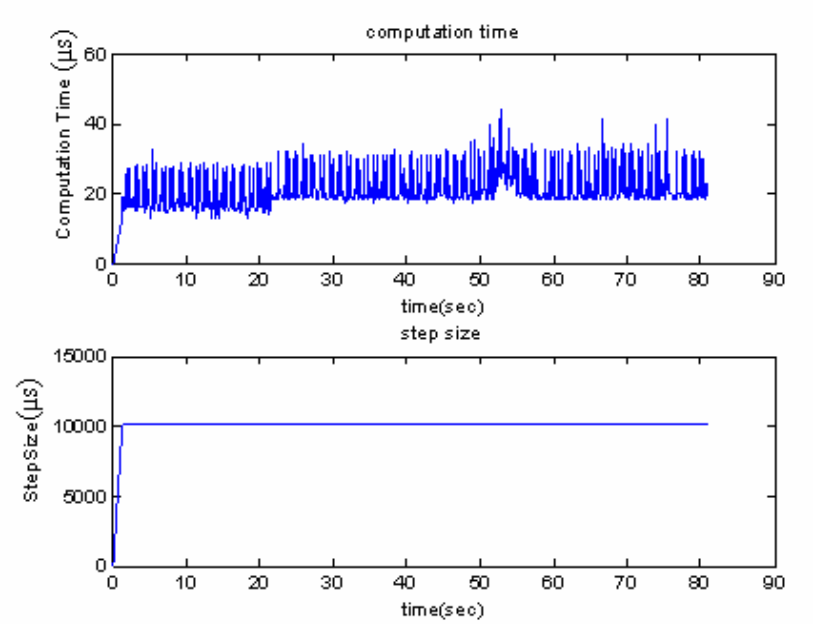


Figure 5.28: Computation time and step size from real time simulation for an Acceleration test.

From the Sections 5.2-5.4 it can be seen that the HIL simulation platform was able to mimic the HEV model. This is verified by making sure that the HEV performs as per the expectations and within its operating constraints in real time similar to the offline simulation. The comparison of VTS at various stages is given in Table 5.1.

Table 5.1: VTS comparison

Description	**Base**	**ChallengeX VTS**	**Initial VTS**	**Offline**	**Online**
IVM – 60 mph	8.9 sec	≤ 8.0 sec	≤ 8.0 sec	8.0 sec	8.5 sec
50 – 70 mph	6.8 sec	≤ 6.8 sec	≤ 4.0 sec	3.6 sec	3.92 sec
Vehicle Mass	3825 lb	≤ 4400 lb	≤ 4400 lb	4200 lb	4200lb
MPG Combined EPA(mpgge)	23.3	≥ 32	≥ 35.6	39.15	39.1

5.5 SCM Tuning

The next stage of the controller development is to fine tune the parameters and operating thresholds such as the engine operating line, the minimum battery SOC and the minimum engine on time in hybrid mode. The tuning process has been initiated for a series-parallel 2x2 HEV and the procedure for making a tradeoff between the fuel consumption and emissions is given in this section.

The series-parallel 2x2 HEV operates in a series mode at certain vehicle speeds and driver power requests. In a series mode, the engine operates at an approximately fixed rotating speed to provide the starter/generator and EM with sufficient power to drive the HEV. The engine operating point can be chosen to be in the best efficiency zone. Hence, the series mode gives an excellent opportunity to improve the fuel economy and reduce emissions.

The high fuel economy and low emission zones for a compression ignition (diesel) engine can be seen in Figure 5.29. It can be seen that the high miles per gallon (MPG) area does not coincide with low engine emissions [21]. The challenge is to balance engine efficiency with the priority of reducing emissions.

The HIL setup provides an opportunity to tune the SCM to achieve the best design objectives of a series-parallel 2x2 HEV. This is accomplished by determining a new set of engine operating points located between the optimal engine efficiency line and least emission line. The “NEW OPERATING LINE” shown in Figure 5.30, illustrates a line on which the engine can be operated instead of the one based on best efficiency. To

demonstrate the tuning capability, a new operating line was obtained by scaling down the set of points representing the best efficiency torque curve to 80% of the original torque values. The SCM would then maintain engine operation along this new curve.

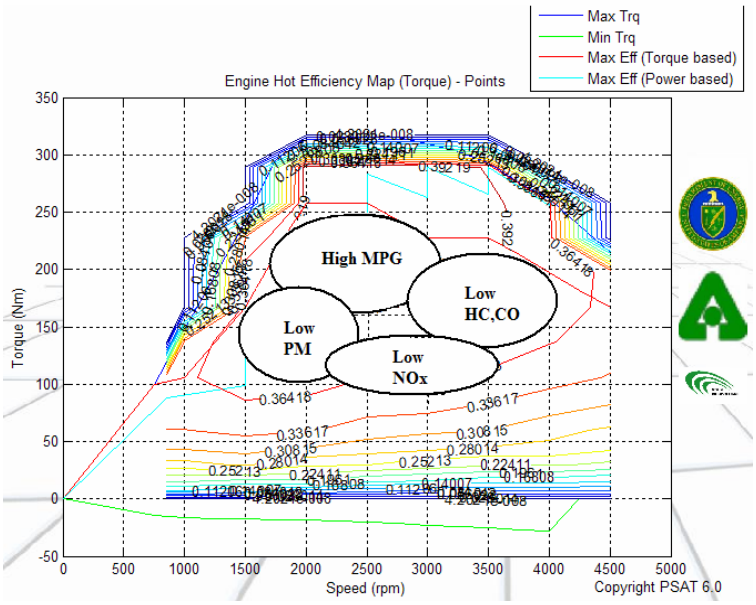


Figure 5.29: Fuel economy and emissions tradeoffs for a diesel engine.

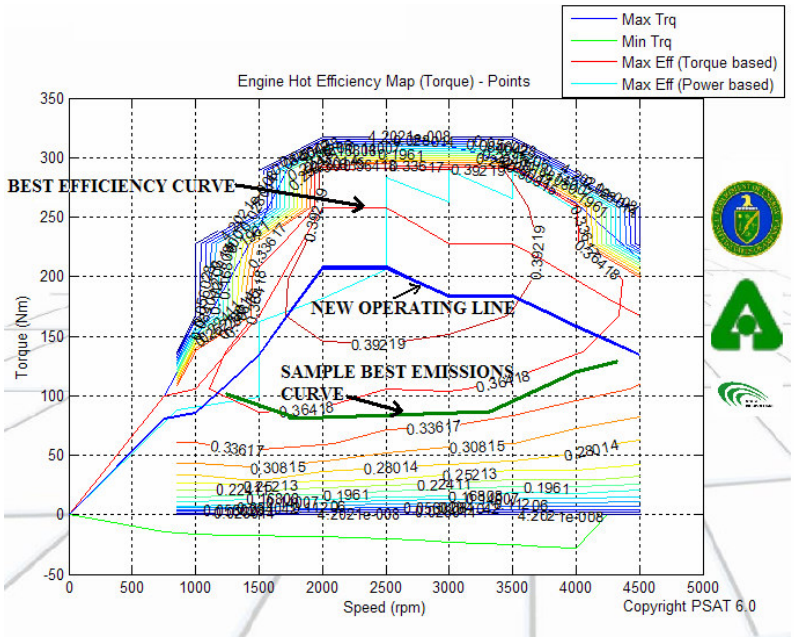


Figure 5.30: Engine map with best efficiency curve, sample low emission curve and a new engine operating line.

The engine map before with its operating points for a HWFET cycle is shown in Figure 5.31. It can be seen that the SCM is controlling engine operation along the locus (Locus-1) of the operating points for maximum efficiency (torque based). The locus can be seen to be in the high emission zone.

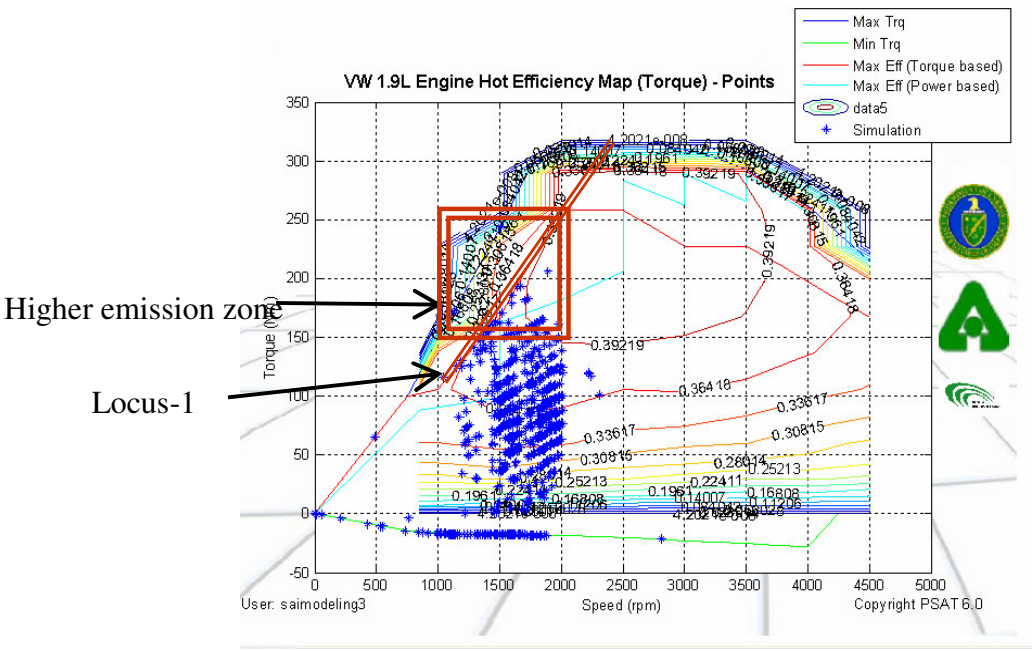


Figure 5.31: Operating points on an engine map (HWFET driving cycle).

To understand the effect of tuning, the SCM strategy was then tuned to use the new operating line and relocate engine operation towards the lower emissions. The series-parallel 2x2 HEV model was again run on the HWFET driving cycle. The engine operating points on the same map are given in Figure 5.32. It can be seen that the SCM is controlling engine operation along the Locus-2 which is away from the high emission zone compared to Locus-1. Also it can be seen that in Figure 5.32 there are hardly any engine operating points in the high emission zone shown compared to Figure 5.31.

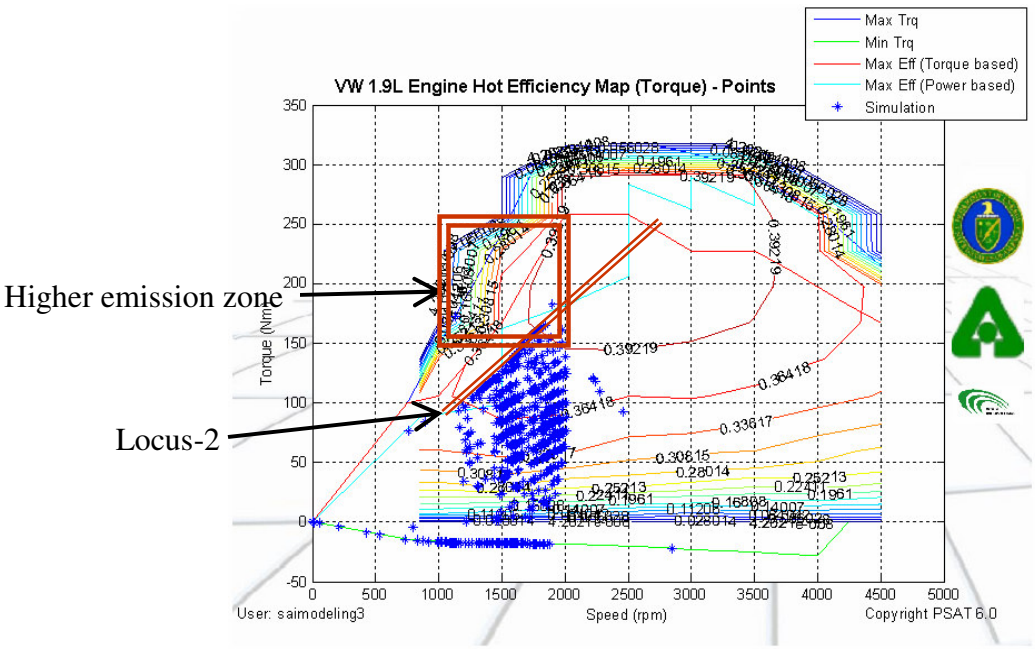


Figure 5.32: Operating points on a tuned engine map (HWFET driving cycle).

The results from the two simulations indicate that this tuning had a minor impact on fuel mileage (about a reduction of 3 mpg), but should give a significant reduction in engine emissions. The current modeling environment and infrastructure currently do not provide any emission analysis. Therefore, the accurate tuning has to be verified by field tests that are proposed for a future study. This process can be further refined when dynamic engine models are developed and integrated into HIL testing.

5.6 Real Time Simulation of an EM drive

A model of EM drive was used to demonstrate the versatility of the HIL simulation setup in running a subsystem in real time. The HIL setup is used in Configuration-3, as explained in Section 4.5 where the motor controller runs on the xPC TargetBox and the RTS mimics an EM. A model of EM drive was simulated in real time and the output speed matched the reference speed and was seen that the time scale matched with real clock (Figure 5.33). The real time simulation information was saved on the host PC to ensure that no overruns were seen. The EM model was run with a fixed time step of 10 µs.

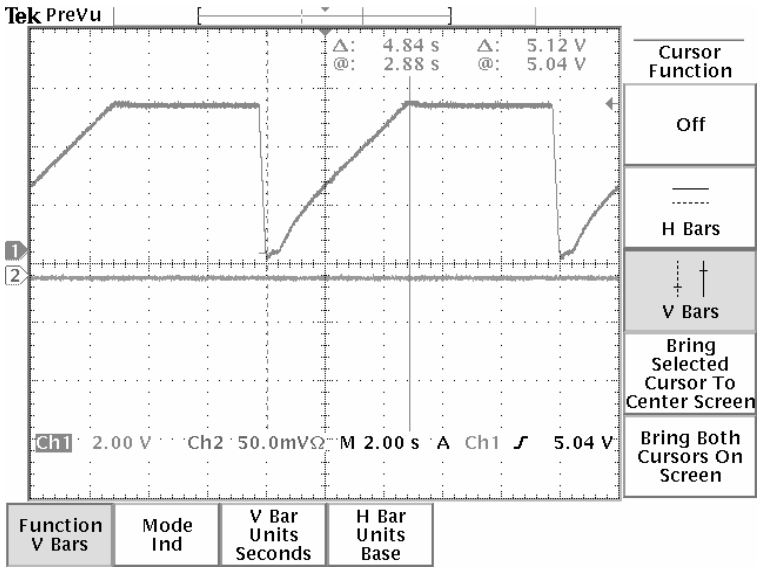


Figure 5.33: Speed profile (real data taken from oscilloscope).

Real Time Simulation Information (EM drive)

The plots of number of overruns, computation time and step size are given in Figures 5.34-5.36. It can be seen that the number of overruns is zero indicating that the HIL setup can be successfully used for real time simulation of HEV subsystems. Also the effective

total time taken (TET+TC) is always less than the fixed step size. This is a requirement of real time simulation as discussed in Section 4.1.

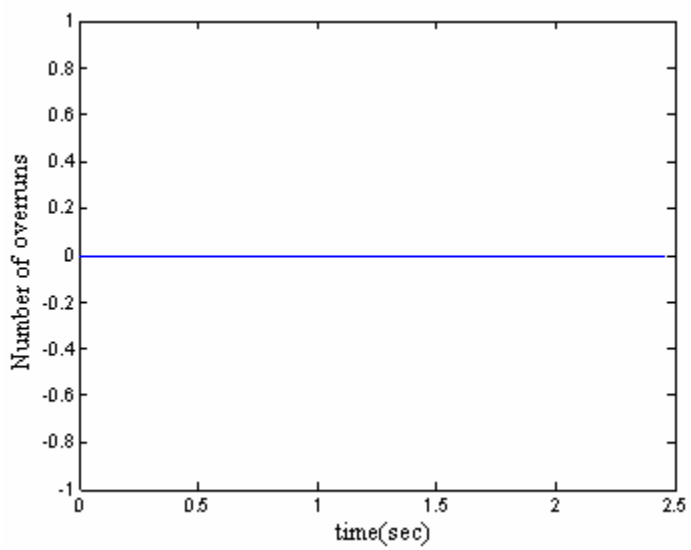


Figure 5.34: Number of overruns in real time simulation of an EM drive.

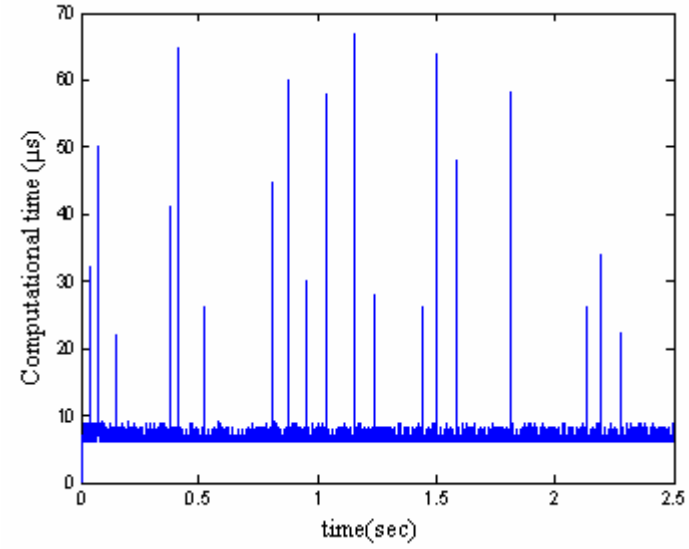


Figure 5.35: Computation time in real time simulation of an EM drive.

5.7 Conclusion

The series-parallel 2x2 HEV is simulated successfully on the developed HIL simulation platform. The real time HEV simulations can be used to determine whether the chosen subsystem components enable the HEV to meet the required VTS. The developed HIL setup was incorporated into the vehicle development process associated with the Challenge X Hybrid Electric Vehicle design competition. The pedals interfaced to the controller allowed a customized input in the form of accelerator and brake positions. The interface was demonstrated to work with the HIL setup making it a good virtual vehicle simulation platform.

The SCM has been tested to work and meets the expectations in the real time environment. The models are ascertained to be running in real time by closely monitoring the real time simulation parameters. The SCM running on a prototype module successfully interfaced to a RTS demonstrates the usefulness of a HIL setup.

A vehicular subsystem, namely an EM drive is successfully run on the HIL setup at a fixed time step of 10µs without any overruns. The configuration in which the HIL setup can be used to test various aspects of real time simulation of a HEV and its subsystem was demonstrated.

CHAPTER VI

CONCLUSION AND FUTURE WORK

6.1 Summary of Research

In this thesis, an overview of the operation of a HEV is given and the various stages in the development of a HEV control strategy from Model-in-Loop simulation to Hardware-in-Loop implementation are discussed. A HEV control strategy operative for a series-parallel 2x2 architecture was added to the existing control libraries in PSAT for further development. The HEV design was demonstrated to meet the initial specifications and was tested in real time on a HIL simulation setup.

6.2 Contributions of Research

A HIL simulation platform was built from the ground up in a step wise manner. The Matlab/Simulink model of a series-parallel 2x2 HEV was run on the HIL setup subjected to various driving cycles and custom inputs. The real time simulation demonstrated that the HEV architecture with the new controller added meets the preliminary requirements of performance, fuel efficiency and weight as mentioned in the requirements. The capabilities of the HIL simulation platform and its various operating configurations were demonstrated successfully. The tuning process of the SCM is initiated and a case of tuning the engine operating line is demonstrated. The HEV controller is tested on the HIL

setup incorporating a CAN interface mimicking the communication in a real vehicle to ensure that the series-parallel 2x2 HEV meets the design requirements. The minor differences observed are attributed to the I/O delays that include analog I/O and CAN I/O. A model of an EM drive was also run at a time step of 10µs to demonstrate the versatility of the HIL platform. An additional computational node was added to the HIL platform to demonstrate the distributed HIL simulation. The entire setup stands as a virtual vehicle simulation platform and can be customized as per the requirements.

6.3 Suggested Future Work

The entire work for the thesis was inspired and done as a part of Challenge X, a collegiate level HEV design competition. The discussion covers the topics from HEV simulation to its deployment onto a Real Time Simulator. Suggested future work is given in the following sections.

6.3.1 HEV Control Strategy

A preliminary control strategy is added to the libraries, and its operation is successfully demonstrated. The development of the control strategy is an ongoing process as the models of the subsystem are improved at every stage until the vehicle is actually tested on a testing track. The procedure for building a control strategy in a particular vehicle simulation platform is described and demonstrated. The control strategy performance can be greatly improved from knowledge of past data and future driving requirements. A block can be added which will monitor the driver habits, and based on heuristic approaches the control parameters can be changed based on the recorded driving habits.

A schematic representation is shown in Figure 6.1. The monitoring block does not have to be in the time critical loop. One of the variables that can be set is the SOC_{MIN} threshold, which influences the SOC correction, and hence, decides the enabling or disabling of the engine. This variable can be adjusted to reduce the number of engine on/offs to enhance the drivability of the vehicle.

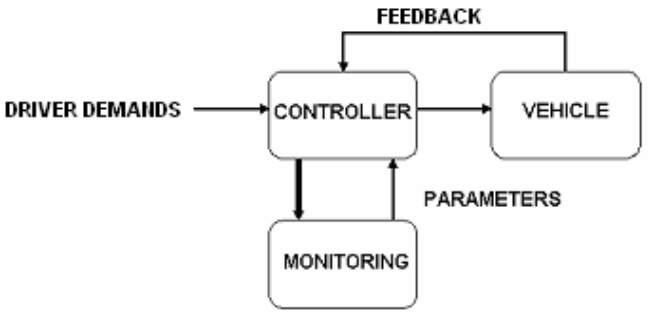


Figure 6.1: Parameter tuning.

6.3.2 Drivability Issues

The current model of the HEV is static and is based on lookup tables. The drivability over specific drive cycles can be predicted and compared to the conventional vehicle performance by incorporating dynamic models of the subsystems and the vehicle body. The HIL setup works as a turnkey system for this application as the platform is ready to simulate a complex model of a HEV in real time. If the computations are complex, additional nodes can be added easily.

6.3.3 Hardware Switching

The HIL platform can be used together with a hardware switching device to simulate a real failure such as a failed electrical connection or any dead data bus (Figure 6.2). This

will ensure a fault tolerant system. This is very inexpensive and can be easily done as a project to demonstrate the robustness of the control strategy.

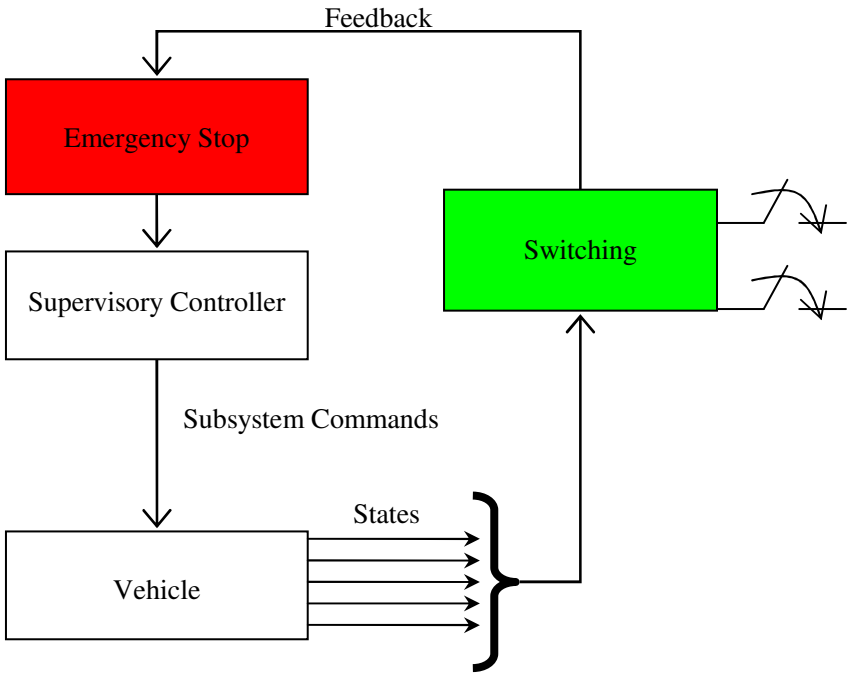


Figure 6.2: Simulating a failure.

REFERENCES

1. C.C. Chan and K.T. Chau, *Modern Electric Vehicle Technology*, Oxford University Press, Oxford, UK, 2001.

2. LMC Automotive Forecasting Services-J.D. Power and Associates, "The Hybrid-Electric Vehicle Outlook," February 2005.

3. S. Delprat, T.M. Guerra, J. Rimaux, "Optimal control of a parallel powertrain: from global optimization to real time control strategy," *55th IEEE Vehicular Technology Conference*, Spring 2002, pages 2082- 2088, Vol. 4.

4. "CAN Specification, Version 2.0," Robert Bosch, GmbH, Stuttgart 70442, Deutschland. www.can.bosch.com.

5. J.M. Miller, *Propulsion Systems for Hybrid Electric Vehicles,* Institution of Electrical Engineers, 2004.

6. The University of Akron, "Challenge X Report #2[1]: Vehicle Architecture Selection and Analysis," November 2004.

7. I. Husain, *Electric and hybrid vehicles: Design fundamentals,* CRC Press, 2003.

8. The University of Akron, "Challenge X Report #3[1]: Control System Hardware and Software Development," March 2005.

9. C.A. Rabbath, M. Abdoune, J. Belanger and K. Butts, "Simulating Hybrid Dynamic Systems," *IEEE Robotics and Automation Magazine*, Vol. 9, No. 2, June 2002, pp. 39-47.

10. D. Ramaswamy et al, "A Case Study in Hardware-in-the-Loop Testing: Development of an ECU for a Hybrid Electric Vehicle," SP-1857, *SAE World Congress*, March 2004.

11. The University of Akron, "Challenge X Report #5[1]: Series-Parallel 2x2 Hybrid Electric Vehicle Architecture Design," May, 2005.

[1] Challenge X reports can be obtained from The Department of Electrical Engineering, The University of Akron, Akron, OH 44325

12. H. Hanselman, "Hardware-in-the-Loop Simulation Testing and its Integration into a CACSD Toolset," *The IEEE International Symposium on Computer-Aided Control System Design*, Dearborn, Michigan, USA, September 1996, pp. 152-156.

13. G. R. Babbitt, J. J. Moskwa, "Implementation Details and Test Results for a Transient Engine Dynamometer and Hardware-in-the-Loop Vehicle Model," *Proceedings of the 1999 IEEE Intemational Symposium on Computer Aided Control System Design*, Kohata Coast, Island of Hawaii, August 22-27, 1999, pp. 569-574.

14. J.J Moskwa, "Automotive Engine Modeling for Real-time Control," Department of Mechanical Engineering, Ph.D. Thesis, 1988, Massachusetts Institute of Technology.

15. J, Adrian, "Audi: Complete Powertrain Simulation," dSpace News, May 2001, dSPACE GmbH. www.dspace.de.

16. W, Baeker, "Bugatti: Powerful Cars Need Powerful Tests," dSpace News, January 2004, dSPACE GmbH, www.dspace.de.

17. J.C. Piedboeuf, M. Doyon, R. L'Archeveque, E. Martin, "Simulation Environments for Space Robot Design and Verification," *Advanced Space Technologies for Robotics and Automation,* December 2000, Noordwijk, The Netherlands.

18. J.D. Graves, "Design and Control of a Vehicle for Neutral Buoyancy Simulation of Space Operations," Master of Science Thesis, 1997, University of Maryland at College Park.

19. M.W. Suh, J.H. Chung, C.S. Seok, Y.J. Kim, "Hardware-in-the-loop simulation for ABS based on PC," *International Journal of Vehicle Design*, 2000, Vol. 24, No. 2/3 pp. 157 – 170.

20. P. Baracos, G. Murere, C.A. Rabbath, W. Jin, "Enabling PC-based HIL simulation for automotive applications," *Electric Machines and Drives Conference, 2001, 1IEEE International,* Cambridge, MA, July 16-20, 2001pp 721-729

21. A. Rousseau, S. Saglini, M. Jakov, D. Gray, K. Hardy, "Trade-Offs Between Fuel Economy and NOx Emissions Using Fuzzy Logic Control with a Hybrid CVT Configuration," *19th International Battery, Hybrid and Fuel Cell Electric Vehicle Symposium and Exhibition, EVS19*, Busan, Korea, August 2002.

22. K.T. Chau, Y.S. Wong, "Overview of Power Management Strategies in Hybrid Electric Vehicles," *Energy Conversion and Management* Vol. 43, Issue 15, October 2002, pp. 1953-1968.

23. A. Sciaretta et al., "Optimal Control of a Parallel Hybrid Electric Vehicle," *IEEE Transactions on Control Systems Technology*, May 2004, Vol. 12, No. 3, pp. 352-363.

24. J. Hellegren, K. Jonasson, “Comparison of Two Algorithms for Energy Management of Hybrid Powertrains,” *Nordic Workshop on Power and Industrial Electronics*, Trondheim, Norway, Vol. 1, Paper 060.

25. L.D. Hyoeun et al., “Torque Control Strategy for a Parallel-Hybrid Vehicle using Fuzzy Logic,” *IEEE Industry Applications Conference*, October 1998, Vol. 3, pp. 1715-1720.

26. C.C. Lin, H. Peng, J.W. Grizzle, “Power Management Strategy for a Parallel Hybrid Electric Vehicle,” *IEEE Transactions on Control Systems Technology*, November 2003, Vol. 11, Issue 6, pp. 839-849.

APPENDICES

APPENDIX A

SERIES-PARALLEL 2x2 HEV CONTROL STRATEGY

The screenshots of the propelling block of the control strategy are given in the Figures A.1-A.6.

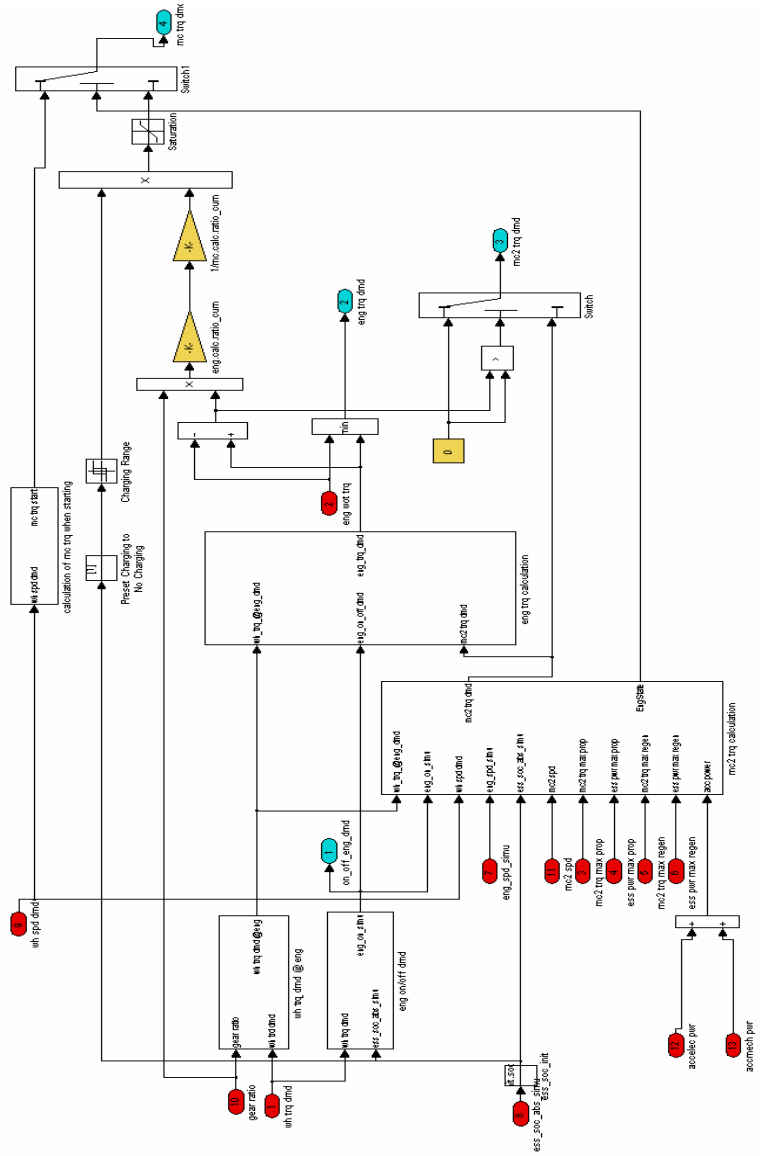


Figure A.1: Screenshot of the HEV control strategy: propelling

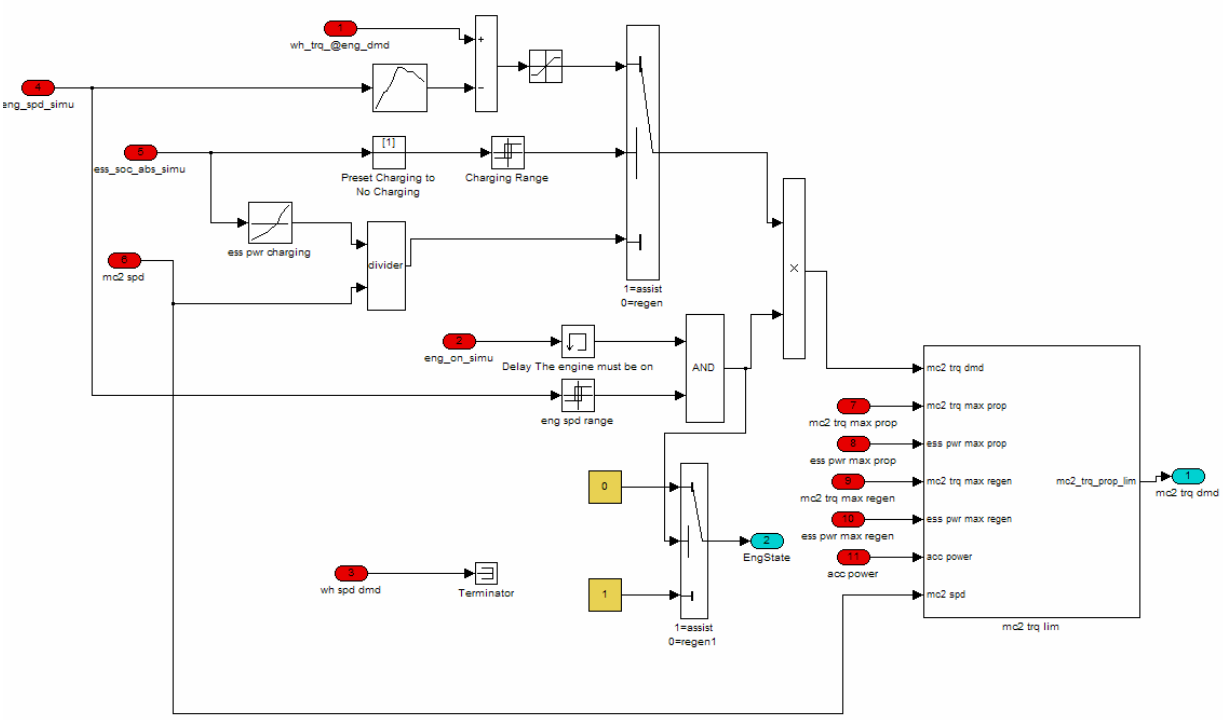


Figure A.2: Screenshot of the EM torque calculation subsystem (mc2 trq calculation).

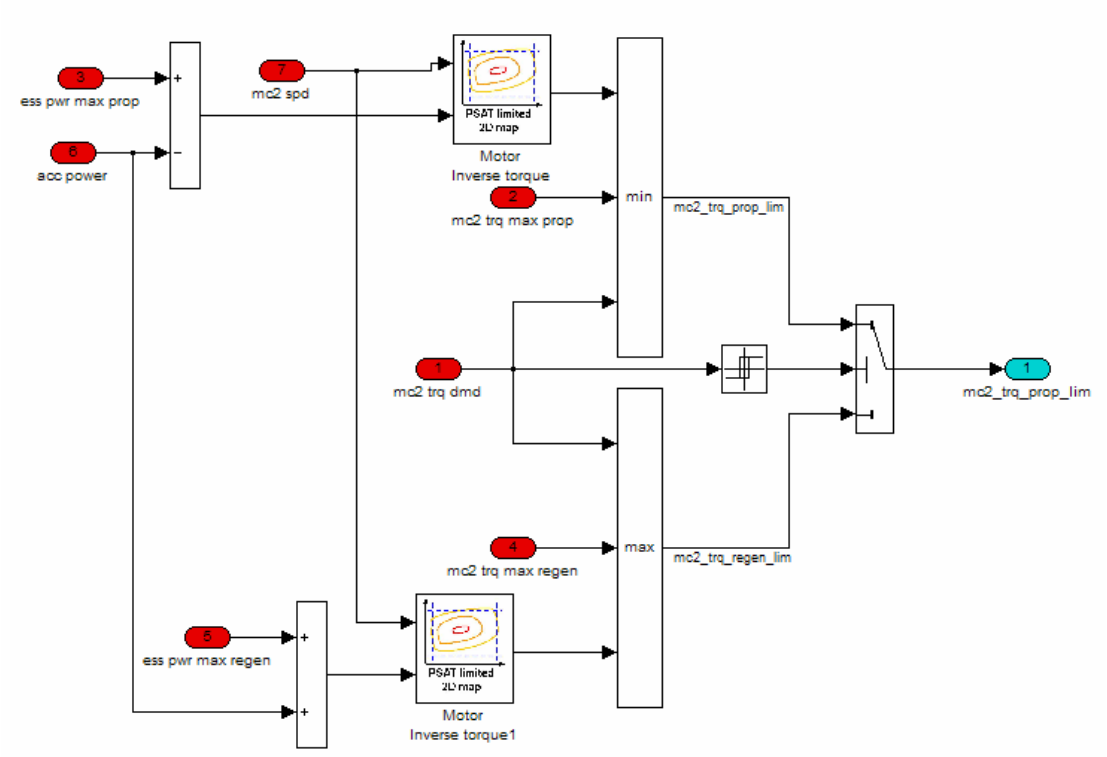


Figure A.3: Screenshot of the EM torque limit calculation subsystem (mc2 trq lim).

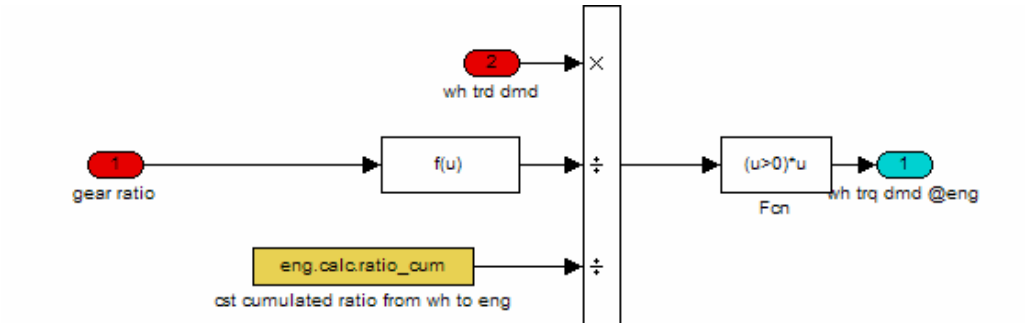


Figure A.4: Screenshot of the calculation of wheel torque demand at engine subsystem (wh trq dmd @ eng).

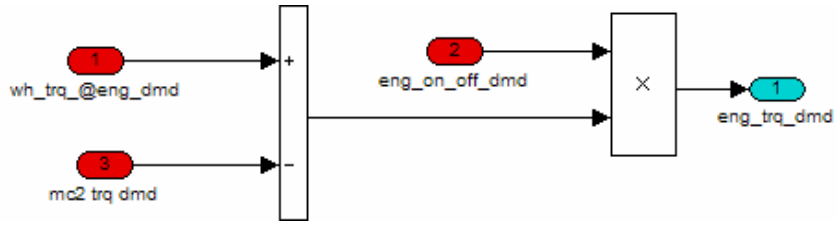


Figure A.5: Screenshot of the engine torque calculation subsystem (eng trq calculation).

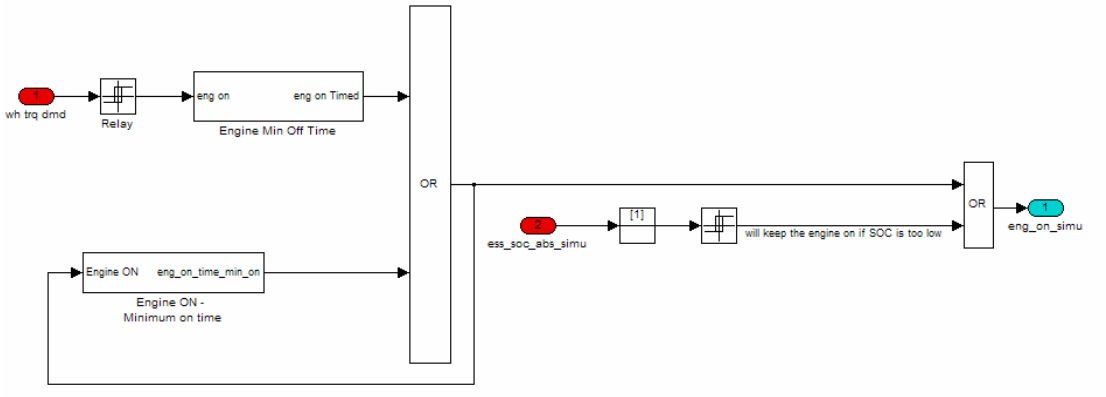


Figure A.6: Screenshot of the engine on/off subsystem (eng on/off demand).

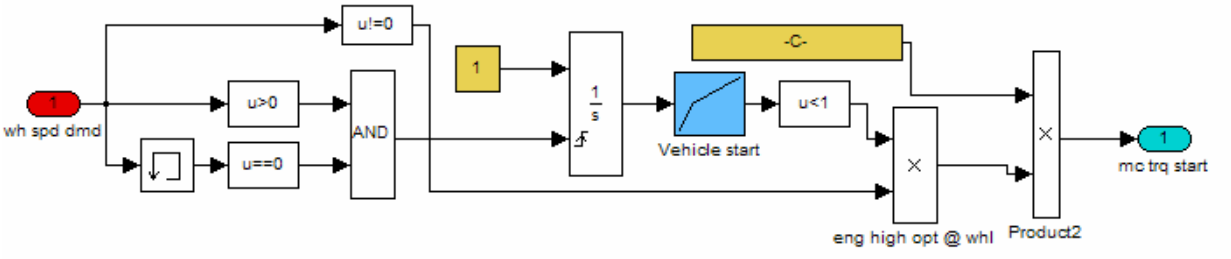


Figure A.7: Screenshot of the starter torque calculation subsystem (calculation of mc trq when starting).

APPENDIX B

SETTING UP REAL TIME SIMULATION OF THE VEHICLE MODEL IN RT-LAB

This Appendix explains the step wise procedure for changing the offline Simulink model into a model ready for real time code generation (for RT-LAB), downloading the generated code to the real time nodes and obtaining the data back from the nodes.

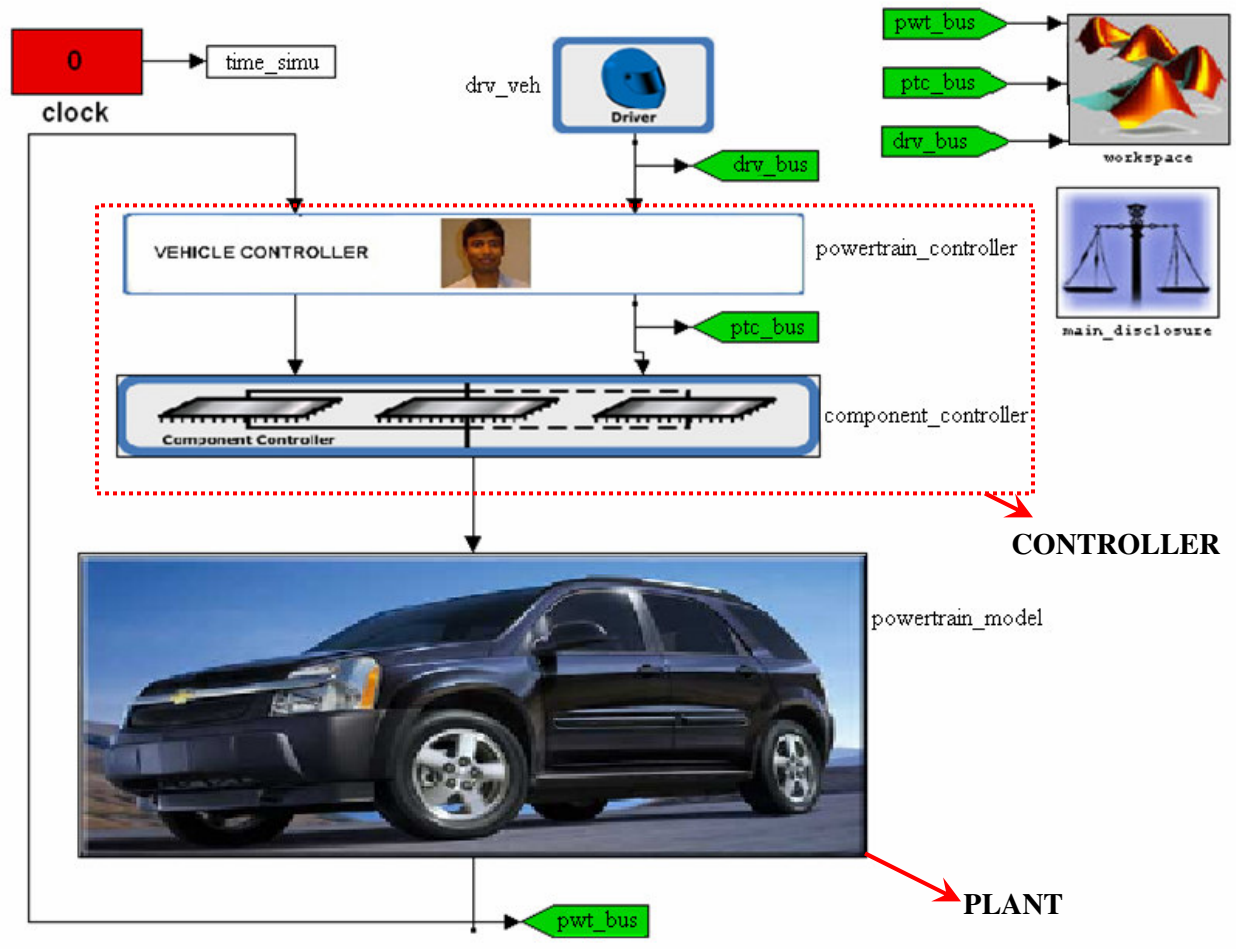


Figure B.1: Screenshot of Simulink model representing the HEV.

Step 1: Rearrange the original model

Obtain the Matlab/Simulink of the vehicle (conventional/HEV). A screenshot of the PSAT model is shown in Figure B.1. Rearrange the subsystems in the model into three Simulink subsystems viz. master, slave and console.

Follow the naming convention

1) Plant subsystem <name_of_subsystem> to sm_<name_of_subsystem>
2) Controller Subsystem <name_of_subsystem> to ss_<name_of_subsystem>
3) Driver subsystem and data acquisition subsystems into sc_<name_of_subsystem>

The starting two characters tell RT-LAB the subsystems for which real time code has to be generated. Save the new Simulink file as RT_VEH.mdl. Save the data in the workspace into a '.mat' file (such as vehdata.mat).

The model after being rearranged looks like in Figure B.2.

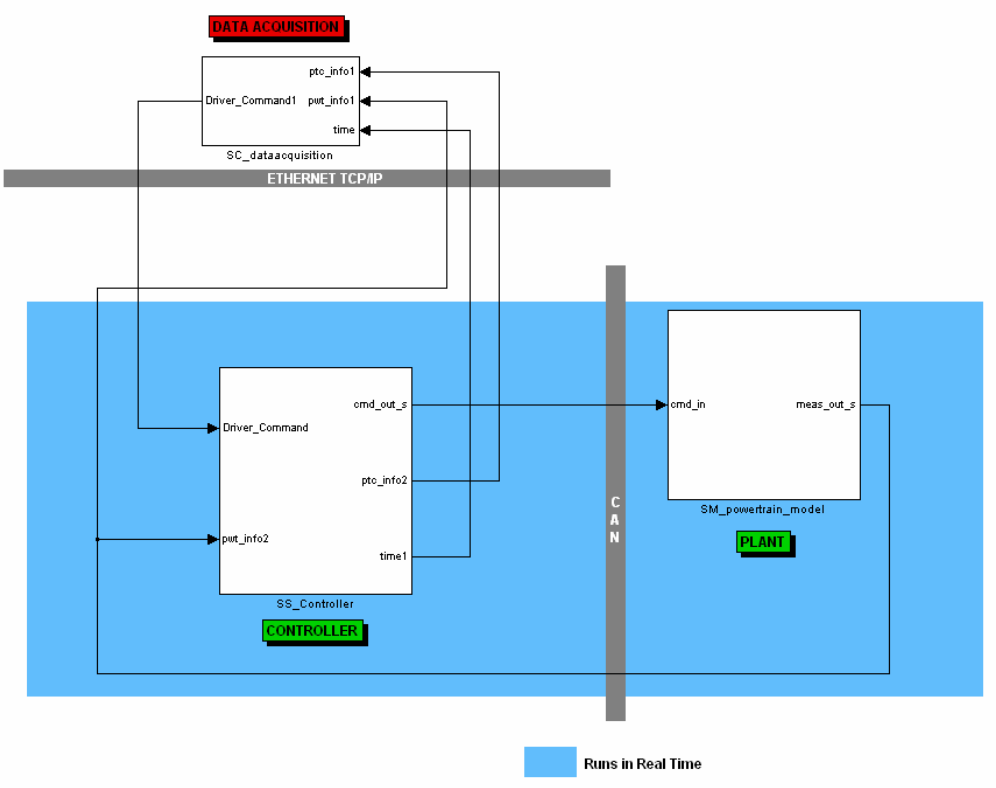


Figure B.2: Rearranged Matlab/Simulink model for HIL simulation.

Step 2: Start RT-LAB

Make sure the RT-LAB Meta Controller is running in the windows system tray as shown in Figure B.3. Open RT-LAB from the programs menu in windows. The window as shown in Figure B.4 comes up.

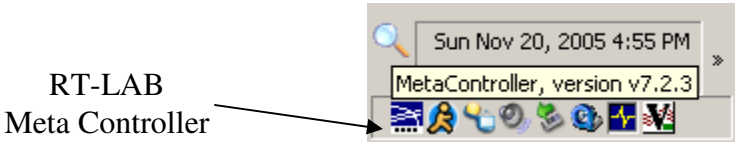


Figure B.3: Screenshot of system tray in Windows XP showing a Meta Controller running.

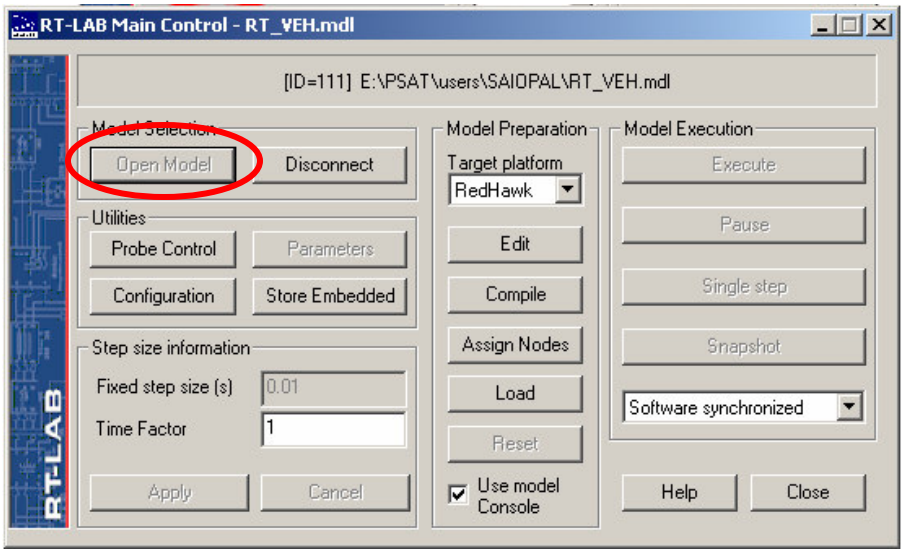


Figure B.4: RT-LAB Main Control: loading the vehicle file.

Left click on the 'Open-Model' and browse for the Simulink file RT_VEH.mdl saved earlier.

Step 3: Link the data file

Left click on 'Configuration' button on the RT-LAB Main Control as shown in Figure B.5.

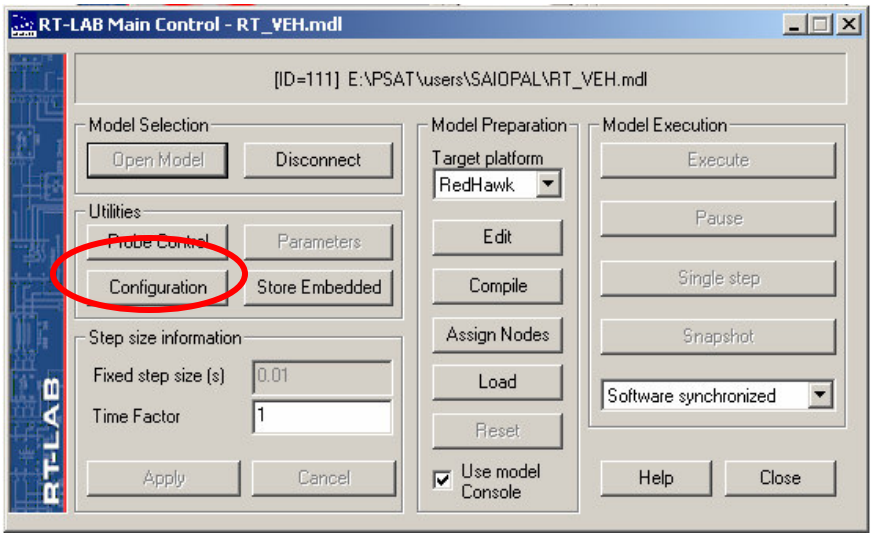


Figure B.5: Linking the vehicle data file-1.

Left click on the ‘Advanced’ button and navigate to Files & Commands tab on the top of the window as shown in Figure B.6.

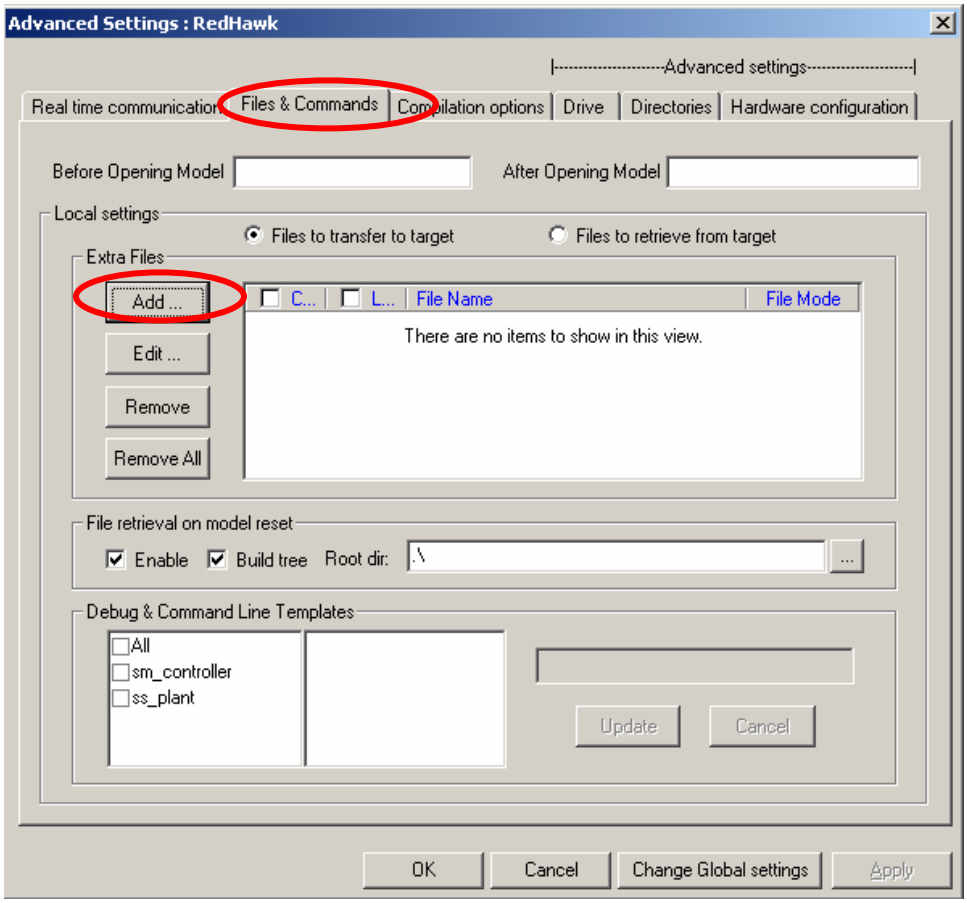


Figure B.6: Linking the vehicle data file-2.

Left click on ‘Add’ button as shown in B.6 and browse for the vehdata.mat file saved earlier. This will be loaded automatically onto the target nodes for Real Time Simulation. Left click on ‘OK’ to get back to the RT-LAB Main Control window.

Step 3: Adding the Real Time blocks from RT-LAB into the Simulink model

Click on ‘Edit’ and the RT_VEH.mdl opens up. Add the required Simulink blocks from the RT-LAB blockset for data acquisition and synchronization between the various nodes. Snapshot of the data acquisition block is given in Figure B.7.

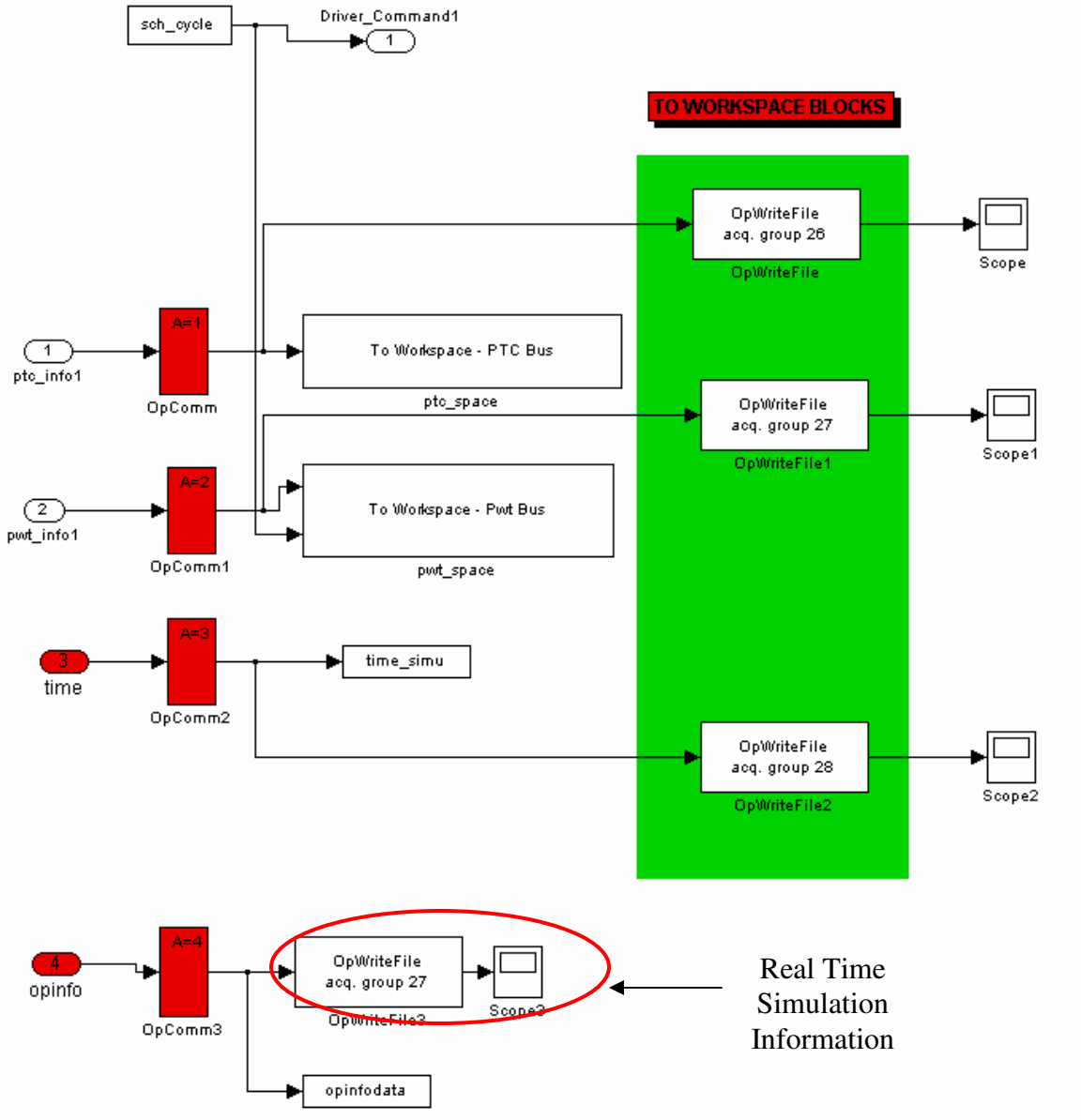


Figure B.7: RT-LAB blocks in the data acquisition subsystem.

Step 4: Compilation

Left click on the ‘Compile’ button shown in Figure B.8. The RT-LAB Display window comes up and the real time code is generated.

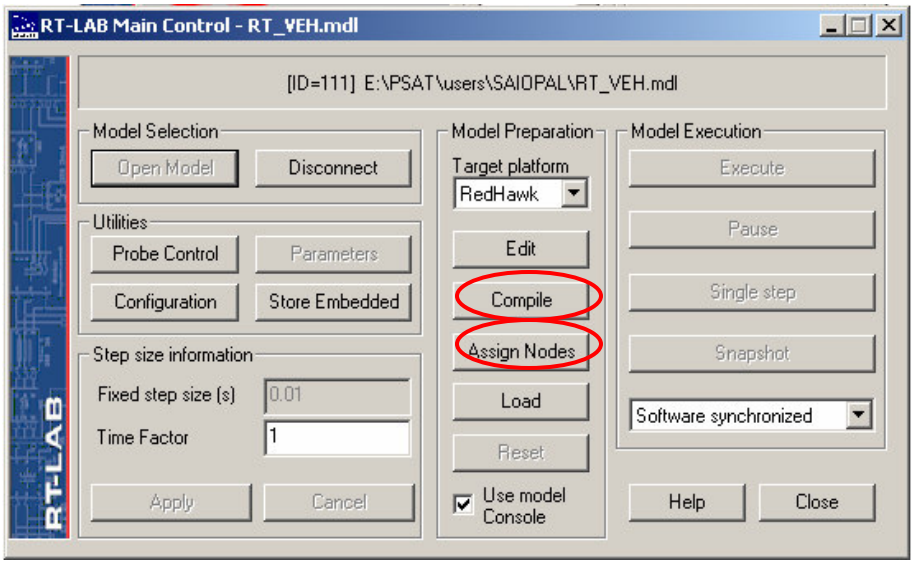


Figure B.8: Compiling using RT-LAB.

Step 5: Assigning real time nodes

Left click on ‘Assign Nodes’ as in Figure B.8. Assign each subsystem a real time node in the network as shown in Figure B.9. It can be seen it is easy to configure and the nodes can be any where in the LAN. Left click on ‘OK’ to get back to RT-LAB main control.

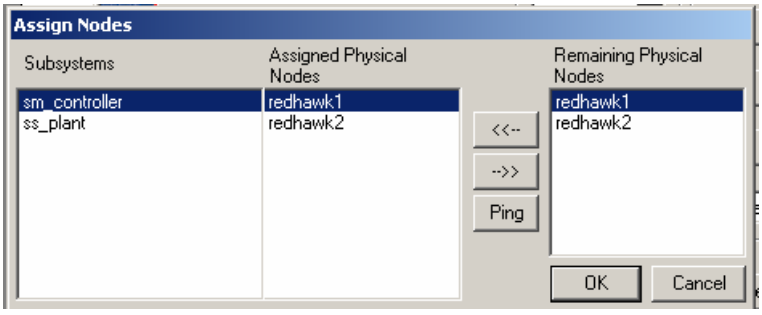


Figure B.9: Assigning the real time nodes.

Step 6: Load and Execute

Left click on ‘Load’ and ‘Execute’ to see the subsystems running on the nodes assigned in the previous step. This is shown in Figure B.10.

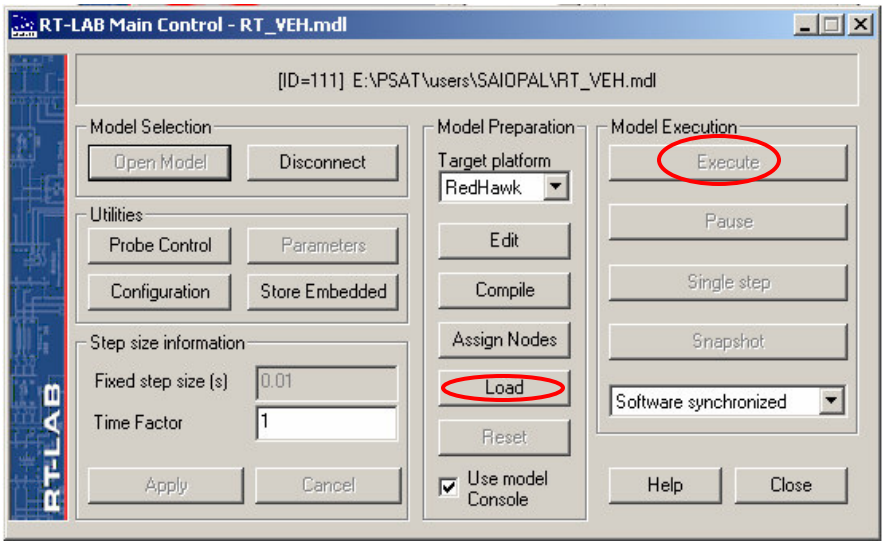


Figure B.10: Load and Execute.