

We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900

Open access books available

186,000

International authors and editors

200M

Downloads

Our authors are among the

154

Countries delivered to

TOP 1%

most cited scientists

12.2%

Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



Formal Verification of Hybrid Automotive Systems

Jairam Sukumar¹, Subir K Roy¹, Kusum Lata² and Navakanta Bhat²

¹*Texas Instruments*

²*Indian Institute of Science
India*

1. Introduction

State of the art automotive systems have become extremely complex in terms of functionality, system architecture and implementation. These systems consist of significant portions of embedded software as well, running in excess of millions of lines of C code. Given such enormous complexities it is never an easy task to verify their functional correctness. Traditional simulation based validation schemes cannot hope to cover the entire specification domain of such a system, as creating test cases to cover all the different corner cases of functional behaviour is extremely difficult. The number of such test cases can be extremely large; it is typically a manual process, at best. Moreover, the time taken to run all the test cases to verify the complete system under all possible scenarios, environmental constraints and input conditions will be inordinately large. This can strongly impact the system design and implementation turn-around-time, thereby reducing the window of opportunity in the market for the product. The amount of complex components being integrated into main stream automotive systems is continuously on the rise, latest being MEMS based sensor components. These MEMS components are becoming a part and parcel of modern automotive electronics systems, ranging from dynamic vehicle performance systems and safety systems to, infotainment and other accessories. To illustrate an example, the heart of any modern automotive engine cruise control (ACC) system includes MEMS based sensors, such as, accelerometers and gyroscopes. The verification of such MEMS components is, in general, hard and difficult, and their integration into a hybrid system exacerbates an already complex hybrid system verification challenge.

Formal analysis of hybrid systems integrating MEMS components, such as sensors, requires their models to be comprehended and integrated into the model of the hybrid system itself. While the cruise control behaviour is primarily discrete, the MEMS sensor behaviour is continuous and highly non-linear. Thus, a system designer will need to validate, both, the discrete time and the continuous time behaviour in any such hybrid system. The single largest challenge in the verification of such hybrid systems lies in appropriately modelling individual components and integrating them within a single verification framework. Another strong motivation in employing formal verification for the validation of hybrid systems, also, is to circumvent the well known drawbacks inherent in a simulation based validation approach. In the domain of discrete time formal verification, a wide range of commercial tools exist which have matured with an increasingly large user base. These

Source: Motion Control, Book edited by: Federico Casolo,
ISBN 978-953-7619-55-8, pp. 580, January 2010, INTECH, Croatia, downloaded from SCIYO.COM

tools, however, are incapable of validating traditional hybrid systems, let alone, hybrid systems integrating MEMS based sensor components. In this chapter, we first present the complexity inherent in hybrid systems and then show how formal verification can be employed to verify them.

2. Advances in automotive systems

Advances in automotive electronics have triggered large scale integration of myriad features for all categories of vehicles. The two fundamental vectors of feature development include safety and passenger comfort. These features can be further classified based on their means of integration, viz., passive and active. While the passive features amount to pre-installed and pre-verified components, each active feature needs a continuous sampling of ambient conditions and based on the condition, feedback actions are taken by the vehicular controller. All these features hence also require processing capabilities much higher than requirements needed in the past. Some of the active mode examples include integrated park and lane assistant, cruise control, climate control, etc.

Advent of these new features also brings to the horizon, the challenge of modelling and analyzing new types of sensors to support these features. We focus on one such type: the MEMS based sensors. The computational complexity of modelling MEMS components, increases manifolds as one needs to solve or model non-linear partial differential equations (PDE) to accurately capture the sensor characteristics.

This chapter attempts to address the challenges posed by integration of these components into mainstream VLSI systems and their verification. The chapter helps the reader to appreciate the nuances of verification methodologies and propose recipes to formally verify computationally complex hybrid automotive systems. A methodology is described, based on transformation techniques that can be deployed to solve these problems. We introduce Simulink-Stateflow based modelling and verification platform [1], as shown in Figure 1. The figure explains how the continuous and discrete time systems interact together in a Matlab based environment. It is shown how this can be used to integrate some of the complex MEMS based components into mainstream VLSI systems. We also introduce the reader to CheckMate, a formal analysis software tool available in public domain. Simulation traces from Simulink-Stateflow framework are used in the formal analysis engine present in CheckMate, instead of being derived implicitly by numerical integration, or explicitly by transformation based approaches.

To illustrate this approach, we choose one representative real life control system, fairly popular, from the automotive hybrid system space. This hybrid system directly interfaces with a MEMS based gyroscope used as a speed sensor. We then deploy our proposed approach to perform a formal analysis of the MEMS integrated ACC system (MIACCS). We model the adaptive cruise control system in a traditional simulation framework, and then proceed to build a formal analysis framework for the same system. The chapter is divided into the following sections. Section 2 introduces the reader to the state of the art in hybrid formal verification methodologies. Section 3 explains about the transformation based techniques for formal analysis. We discuss formal analysis platforms in Section 4 and introduce the reader to one of them in Section 5 viz. CheckMate, which works on Simulink-Stateflow (SSF) based system. SSF based methods are the most widely used platforms across the industry for hybrid and real time system. Finally we illustrate the approach with the case of a MEMS based adaptive cruise control system in Section 6 and through 8.

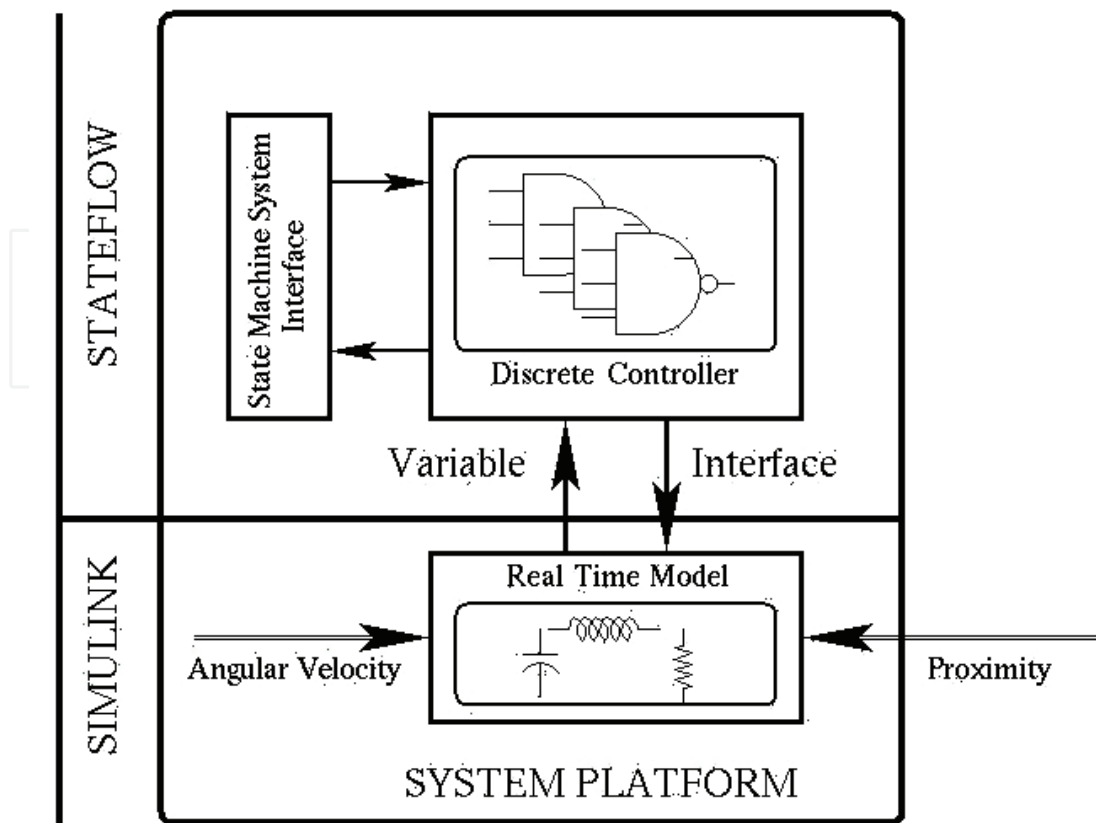


Fig. 1. Simulink Stateflow Platform

3. Hybrid verification methodologies: past, present & future

3.1 Introduction to hybrid formal analysis

Specifically the very low Defective Parts Per Million (DPPM) requirements to meet the stringent automotive fail safety norms and standards, poses new hurdles and therefore newer challenges to overcome them. Thus, it has become a topic of active research in the recent past [2] [3] [4]. Present generation, state of art automobiles, use MEMS based sensors for measurement of different vehicular parameters [5]. Given the fact that, verification of such mainstream automotive control systems need to be based on formal approaches, validation of system behaviour integrating such MEMs components, results in additional complexities. A hybrid system, typically, includes both discrete and continuous time components. To analyze hybrid systems, consisting of, both, discrete behaviour and continuous behaviour components, it becomes necessary to partition these distinct functional behaviours and use specialized analysis engines to target each behaviour domain. A typical hybrid system formulation is shown in Figure 2. First of all, there needs to exist a grammar to formally describe the hybrid system. A given hybrid system is then partitioned into its representative discrete and continuous time systems. Both these partitions are individually solved through their respective solvers. Continuous time solvers include symbolic analysis based solvers, while discrete time solutions can be obtained through traditional graph traversal based methods. Finally both these solutions are integrated by interleaving of variable, at each time step of computation. Decision to change a state is taken based on corresponding conditions being satisfied to trigger these changes.

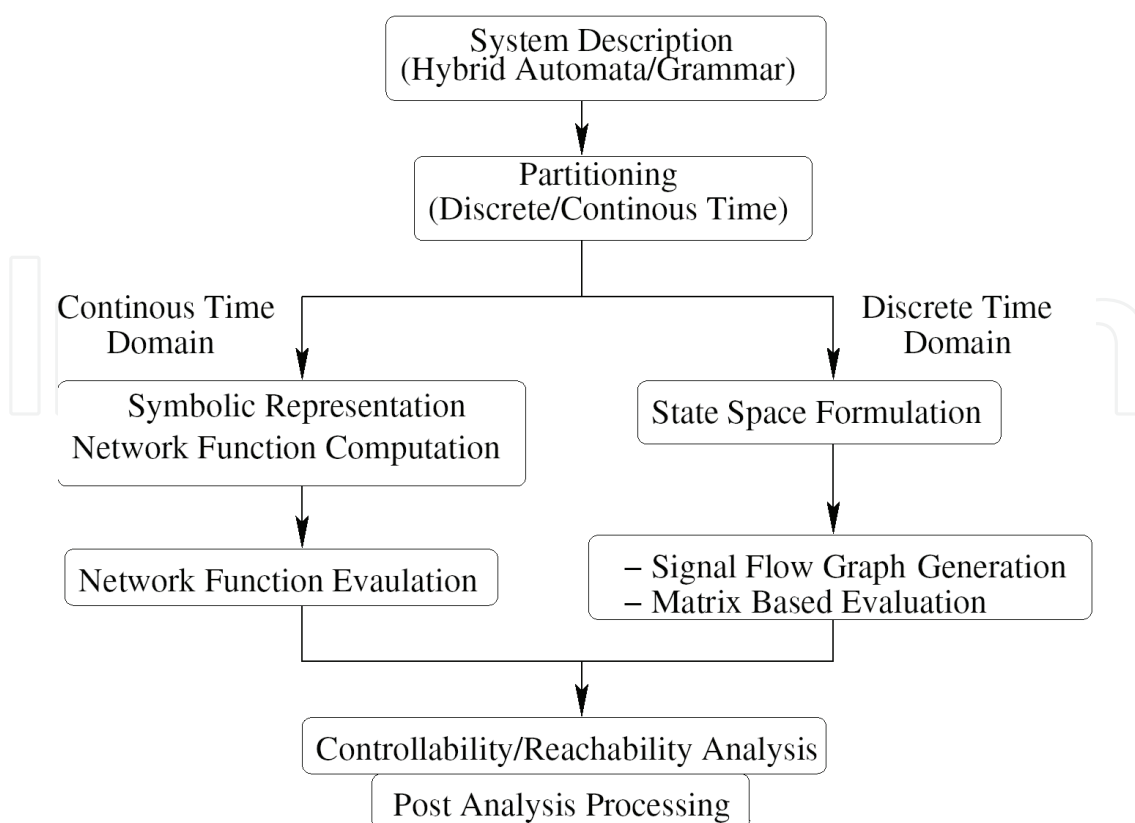


Fig. 2. Formal Verification Flow for Hybrid Systems

For a detailed introduction to the timed and hybrid automata interested readers are referred to [6] and [7]. A discrete time variable needs to be identified and then used to sample the system behaviour in temporal space. After each time step, the system state is determined and the state equations are solved. Changes in the state of the system are then evaluated based on state values and guards corresponding to the transition edges of the automata. The continuous time system formulation is thus, analyzed through time- discretisation and then solved through numerical routines based on Runge-Kutta or Newton-Raphson methods for non-linear dynamic functions [8]. A time discretisation step is followed by linearization of non-integer functions. For each time step, the continuous time domain state variables are concurrently analyzed with the discrete time variables to determine the reachability of the state space of the hybrid system needed to establish truth validity of the safety properties. These terms are explained in later in Section 3.2. Thus, the overall model checking methodology is based on progression in the time domain. Figure 3 illustrates the above methodology.

2.2 Formal analysis of timed automata

The concept of timed automata is illustrated to the reader by a simple example of a temperature controller. A diagram of a temperature controller is shown in Figure 4a. The controller has two states *viz.* ON and OFF. The controller enters into the initial ON state with the temperature variable x being initialized to the room temperature T_{room} . The controller switches the heater which increases the temperature with a linear rate until the temperature attains an upper bound. The heater is cut-off by the controller and the system cools down with a rate given by the differential equation shown in the OFF state (Figure 4b).

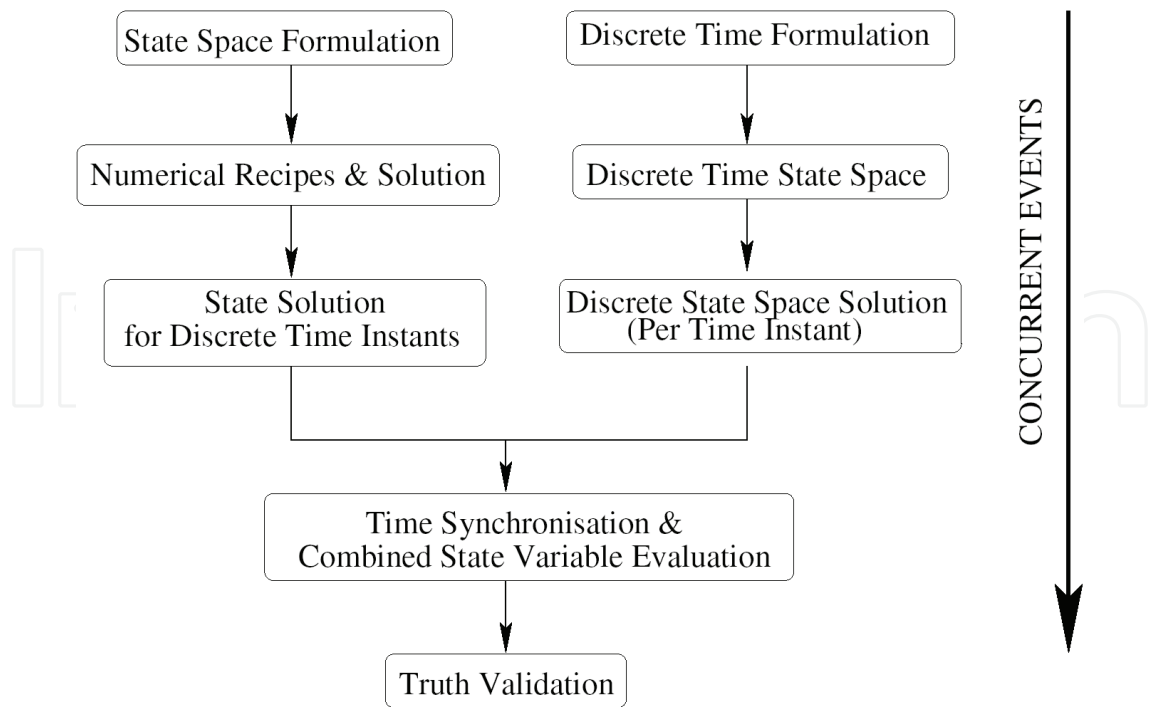


Fig. 3. Formal Verification Flow for Timed Automaton

State equations can be solved using Laplace transforms as follows:
ON STATE:

$$\frac{dx}{dt} = \phi(t) \tag{1}$$

$$X(s) = \frac{1}{s} \phi(s) \tag{2}$$

$$x(t) = \phi(t) + C \tag{3}$$

OFF STATE:

$$\frac{dx}{dt} + \alpha x = f(t) \tag{4}$$

$$sX(s) + \alpha X(s) = F(s) \tag{5}$$

$$X(s) = \frac{F(s)}{s + \alpha} \tag{6}$$

$$x(t) = e^{-\alpha t} f(t) + C \tag{7}$$

It can also be seen that for a class of rational polynomials, there exists a simple partial fraction decomposition which simplifies the Inverse Laplace Transform computation. Equations (3) and (7) are exact functions with respect to the time variable. Thus, exact values of these functions corresponding to each dynamic state variable (for example, temperature)

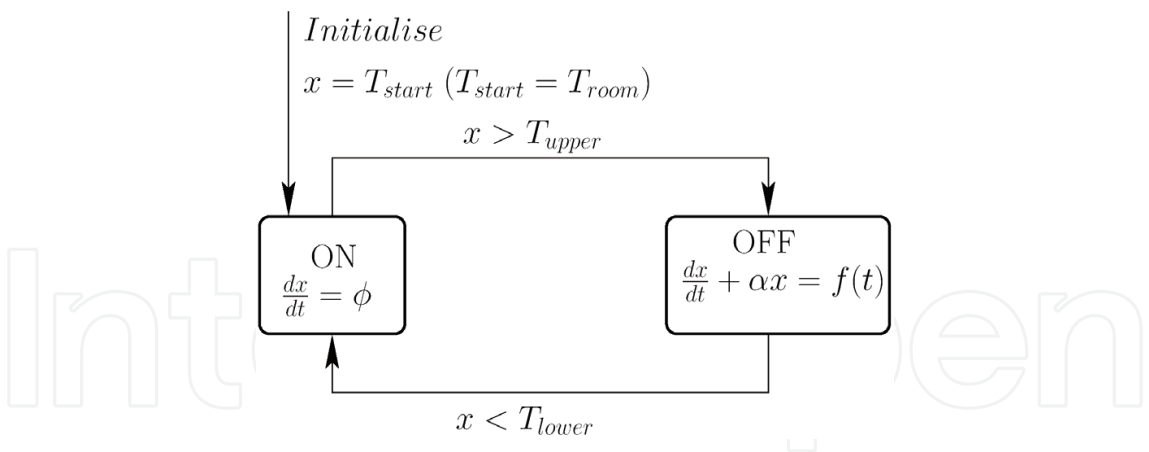


Fig. 4.a State Transition Graph of Temperature Controller

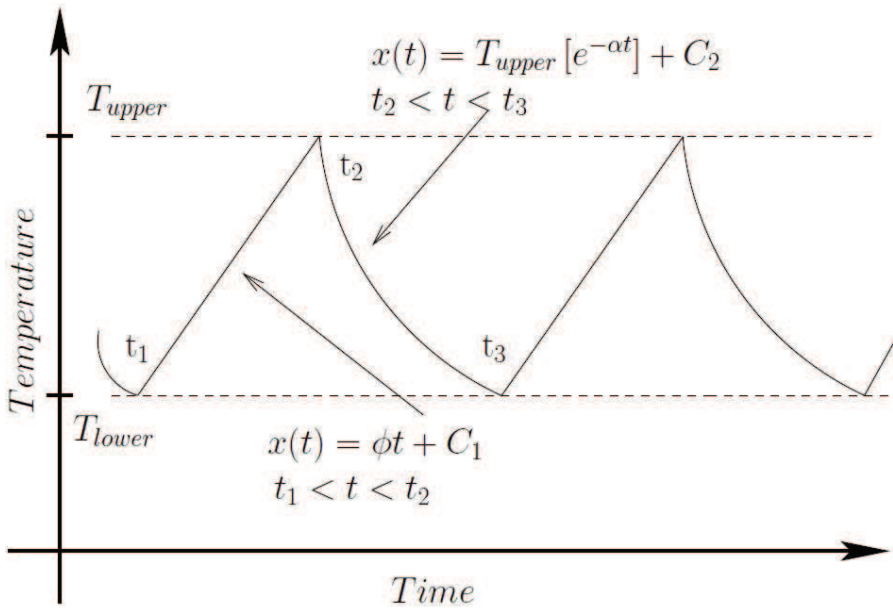


Fig. 4.b System Response

at different time points can be easily and exactly computed to determine the set of reachable states. The steady state solution space, which is also the reachable state space, is shown in Figure 4b. The reachable state space varies with the parameters Φ and α present in the differential equations. The reader should note that, for this parameterized reachable state space corresponding to the temperature variable, constraint solvers need to be deployed to prove the specification correctness.

3. Transformation based formal analysis

State of the art methods of formal analysis are based on timed or hybrid automata. There also exists a symbolic analytical method, for the solution of system description. If an analytical model can be obtained based on exact modeling, it can provide potential computational benefits. Hence let us explore solving linear differential equations symbolically through Laplace transforms instead of simulation. We can transform the state equations in the time domain and then solve the set of algebraic equations on them through the well known method of pole residue decomposition of transfer functions. The solutions

in the original time domain obtained through inverse Laplace transforms are then bound, based on the constraints imposed on the corresponding differential equations. Constraint solvers can be used to solve for these bounds and establish the truth of the properties.

3.1 Partial decomposition based solution

Let us now look at a simple n^{th} order linear differential equation and assume it to be a part of the continuous time system. Let us also assume all state equations to be defined by linear differential equations. The equations will be of the standard form:

$$A_n \frac{d^n x}{dt^n} + A_{n-1} \frac{d^{n-1} x}{dt^{n-1}} + \dots + A_1 \frac{dx}{dt} + A_0 x = f(t) \quad (8)$$

The time domain differential equation is re-written as the following:

$$(A_n D^n + A_{n-1} D^{n-1} + \dots + A_1 D + A_0)x = f(t) \quad (9)$$

Here \mathbf{D} is the differential operator. Transforming the time domain equation in (8)-(9) into the frequency domain through Laplace transform, we arrive at the following equation:

$$(A_n s^n + A_{n-1} s^{n-1} + \dots + A_1 s)X(s) = F(s) \quad (10)$$

$$X(s) = \frac{F(s)}{A_n s^n + A_{n-1} s^{n-1} + \dots + A_1 s} \quad (11)$$

Assuming $X(s)$ can be transformed into a rational polynomial of the type $A(s)/B(s)$, regular polynomial packages like Matlab™ or Octave can be used to convert this rational polynomial into the pole/residue form. The characteristic equation can be re-written as:

$$X(s) = \sum \frac{z_i}{s_i + p_i} + R \quad (12)$$

The time domain solution to (12) can be written as:

$$x(t) = \sum_i z_i e^{p_i t} \quad (13)$$

Simple Matlab/Octave routines can be used to solve a variety of rational polynomials. The reader can try many of such combinations. This approach can be rendered into a simple automated flow which invokes math solvers to solve the standard pole-zero decomposition problems. Using these computed poles and residues it is easy to derive the time domain response of the dynamic components in a hybrid system.

3.2 Constraint formulation and truth validation

Let us now look at the constraint formulation mechanism used in the context of the above example.

We need to formally state the property to be verified. It can be formally described as a constraint. However we need a formal language to describe this constraint. One can imagine a domain space corresponding to the span of a specified property. We call this the constraint

space. The constraint space needs to be intersected with the reachable state space (or the solution space) of the state variables obtained by approaches based on Laplace transforms or by numerical integration of differential equations. If the constraint property space subsumes the reachable state space, the correctness of the controller behaviour is established on its formal model. The solution space of the state variable and the property to be checked can be represented as:

$$X(t) = \begin{cases} X_1(t); t \in D_1 \\ X_2(t); t \in D_2 \\ X_3(t); t \in D_3 \end{cases} \tag{14}$$

The set $D1 \cup D2 \cup D3$ provides the complete time domain region of operation of the system. Let, for all specification S_i the constraint space for its corresponding property be P_i . The objective is to obtain the region of intersection of P_i and $X(t)$. For the temperature controller discussed earlier, we explain the constraint analysis methodology. Let us define three properties; that describe three different safety requirements of the controller in example which we mentioned previously:

- Temperature of the system remains between $[T_{upper} + T_{lower}] / 2$ and T_{lower} .
- The Temperature of the system is always below T_{upper} .
- The Temperature of the system is always between 300C and T_{room} . (Where $T_{upper}>300C$ and $T_{room}<T_{lower}$)

In other words the above three properties can be specified as a part of functional specification of the hybrid system. The functionality of the implemented controller should ensure that these specifications are met. The specifications can be transformed into mathematical inequalities or constraints as shown in the

S/N	Functional Spec	Controller Specifications
1	$T_{lower} < X(t) < [T_{upper} + T_{lower}] / 2$	$T_{lower} < x(t) < T_{upper}$
2	$X(t) < T_{upper}$	$T_{lower} < x(t) < T_{upper}$
3	$T_{room} < x(t) < 300; T_{upper} > 300 \& \& T_{room} < T_{lower}$	$T_{lower} < x(t) < T_{upper}$

TABLE I

A simple method as mentioned above is to formulate the solution surface of the constraints and evaluate the intersection. Solutions that can be incorporated in 2 or 3-dimensional surfaces can be very easily visualized. The solution surface for the properties in Table I can be illustrated as shown in Figure 5. It can be easily noted that the functional specification 1 is not satisfied as the controller specification does not bound the corresponding property. This results in a failure of the property to meet the desired specification.

4. Hybrid analysis platforms

Let us progress to the next level, where we now discuss the complexities involved with ordinary and partial differential equations, and see how we can tackle them. We assume

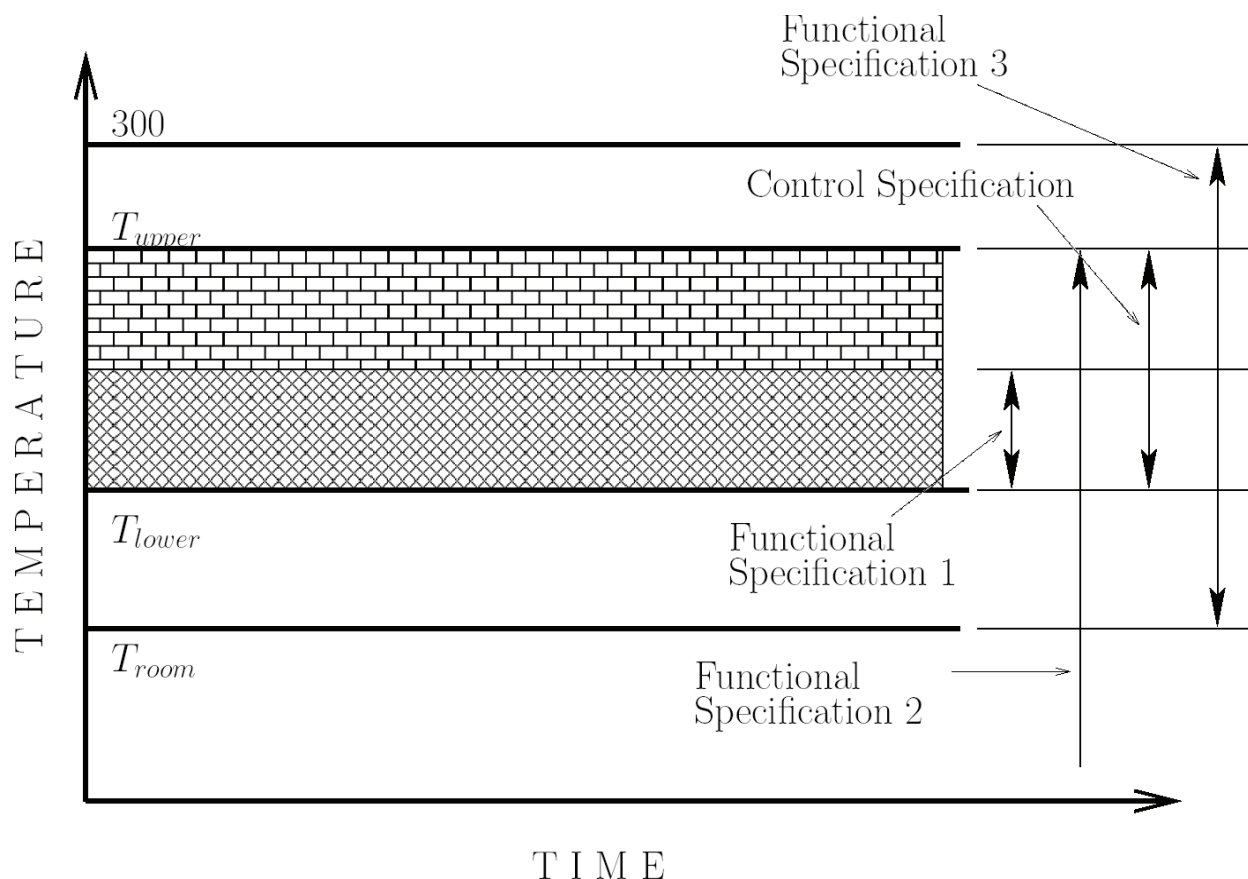


Fig. 5. Steady State System Temperature Response Domains

that the reader is fairly well aware about the basics of differential equations. Analysis of the continuous time domain behaviour modelled by Ordinary/Partial Differential Equations (O/PDE) can be solved by using a Linear Program Solver such as, LP Solver [9] for obtaining the state space solutions and discrete event solvers like SAT solvers [10] for discrete state transitions. In addition, to render the analysis formal, a reasoning engine to interpret the correctness of the behaviour of system implementation against the specified set of formal properties is needed. A fully automated approach would be the golden ideal; however, even for the simplest hybrid system with linear behaviour, it has been theoretically proven that establishing the truth value of properties captured in a first order logic on real numbers is un-decidable and NP-hard [11-13]. Therefore, several automated approaches which are inexact or approximate [14,15,16,17] and other non-automated approaches, based on theorem proving, requiring user inputs to drive the formal analysis [18, 19]; have arisen to tackle the intrinsic complexity inherent in hybrid systems. The presence of components like MEMS based sensors renders the formal analysis even harder, with their dynamic behaviour being modelled and described only by higher order non-linear partial differential equations (NL-PDEs). It is well known that behavioural models based on higher order non-linear partial differential equations are not amenable to pure formal analysis [2].

At the same time, a complete system level analysis suite based purely on simulation approaches for hybrid systems with such components leads to unacceptably high run times, in order to cover every possible combination of corner case behaviours due to internal interactions within system components and external interactions with the system

environment. While, detailed analysis of MEMs structures can be performed by finite element based approaches, carried out within respective energy domains of corresponding sub-systems [20], integrating such analysis framework for system level formal analysis results in huge computational bottlenecks [2]. Symbolic simulation methods are an alternative and can replace traditional simulation [21]. These are being increasingly used in many domains for system validation, where, an analytical solution of the system is achieved through symbolic analysis using solvers of various types [21]. Truth decidability in these cases becomes a computationally difficult task due to the infinite cardinality of the state spaces of the continuous dynamical systems. The authors of [21] give the family of linear differential equations with a decidable reachability problem, by symbolically computing the reachable state sets by posing it as a quantifier elimination problem in the decidable theory of reals. Public domain quantifier elimination tools such as REDLOG ([22]) and QEPCAD ([23]) implement these approaches and have been used in symbolic verification of hybrid systems. As discussed earlier, real life hybrid systems, however, require complex linear and non-linear differential equations to model their dynamic behaviour, thereby rendering symbolic approaches computationally expensive. Reachable state sets for these systems are computed approximately, using numerical methods based on time step integration of differential equations to contain the complexity in computing the exact reachable state sets. This approximate computation is either based on polyhedra, or level sets, or ellipsoids [24], [25], [26] and [27]. In [3], the authors explore new algorithms for accurate event detection for simulation based reachability analysis and abstraction methods. The advantages of transformation based approaches discussed earlier in Section 3, to alleviate problems arising out of misses in the detection of important system behaviour events can be easily seen. This helps in decreased computational overheads arising from numerical integration, accurate event detection and, therefore, increased robustness in formal analysis of hybrid system behaviour. It is easy to see that the analytical form of the proposed approach also, makes it amenable towards computation of approximate reachable state sets, needed for automated formal analysis. While the technique based on transformation approaches is appealing for analyzing hybrid system components described behaviourally by linear differential equations, it is inappropriate for MEMS based components described by non-linear PDEs, which may not even have an analytical solution describing their behavior. However, it is possible to obtain approximate analytical models for some of the MEMS based components, such as, gyroscopes, which involve a single convolution operation in either the time or frequency domains. Even this is computationally expensive for formal analysis. Some of these simulation complexity aspects of MEMS components have been described in [32].

5. Introduction to CheckMate

In this section we give a very brief introduction to CheckMate [27], a public domain tool from CMU. CheckMate can be employed to verify hybrid systems modeled as hybrid automata, either formally using model checking, or through a simulation mechanism. This solution is built on the very popular Simulink/Stateflow Framework (SSF) from Mathworks, widely accepted both in the academia and the industry. CheckMate supports three important custom SSF blocks, viz., Switched Continuous System Block (SCSB), Polyhedral Threshold Block (PTHB), and Finite State Machine Block (FSMB). A hybrid system is modeled primarily using these three blocks. CheckMate, however, supports a few other blocks present in SSF.

A SCSB is used to define the system continuous dynamics in terms of first order differential equations. A PTHB generates events whenever the system crosses a specified threshold described in terms of a linear constraint. This generated event is used as an input to the FSMB to trigger transitions from one state to another. Based on the sink state of a transition edge that is reached, the SCSB block on reaching that state generates the continuous state trajectory using the dynamics corresponding to that state.

Verification is performed using three distinct phases. In the first phase, it allows verification of the hybrid system using traditional simulation based on numerical integration, as supported in SSF. Thus, CheckMate models can be simulated in a manner similar to any other general SSF model. In the next phase, Explore, beginning with the initial location of the hybrid automata, it checks whether each simulation trajectory, starting with different initial conditions corresponding to the set of corner vertices in the convex polytope initial continuous set, satisfies a given formal property specified as an ACTL formula. It informs the user in case of any violation. In the third and final phase, Verify, CheckMate performs formal verification.

6. Adaptive Cruise Control system: a case study

The concept of adaptive cruise control (ACC) system has been developed to aid vehicular traffic on highways. It is an automatic closed loop system, through which a driver during a long drive, can volitionally transfer control to an intelligent vehicular controller system. Figure 6 illustrates a speed control mechanism used in automobiles. A MEMS sensor viz. a gyroscope is attached at the wheel base to provide details on the vehicular speed. Speed and proximity sensors are also installed at the front and rear end of the vehicle to continuously monitor the speed and distance of the vehicle ahead and behind. All these details are then sampled by engine controller to take a suitable course of action for the vehicle.

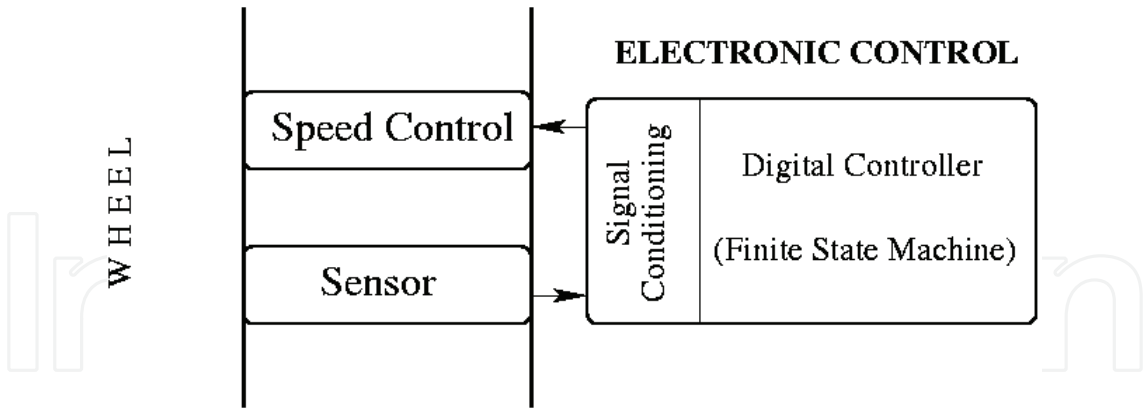


Fig. 6. Closed Loop Vehicular Speed Control

The vehicular control is marked by a closed loop control chain, which is executed in a continuous polling mode. This system is primarily a state machine, which is designed to translate the nature of the traffic conditions into a vehicular and generate an action for each state. The action is again governed by the parameters of highway control viz. speed limits, minimum proximity etc. Figure 7 describes the state transition graph for the ACC system. The system behaviour consists of four states, viz, 'HALT', 'ACCELERATE', 'CRUISE' and 'RETARD'. The variables Xp (for proximity to the front vehicle) and V (for speed) govern the assignments to different states and the transitions between these states. The engine

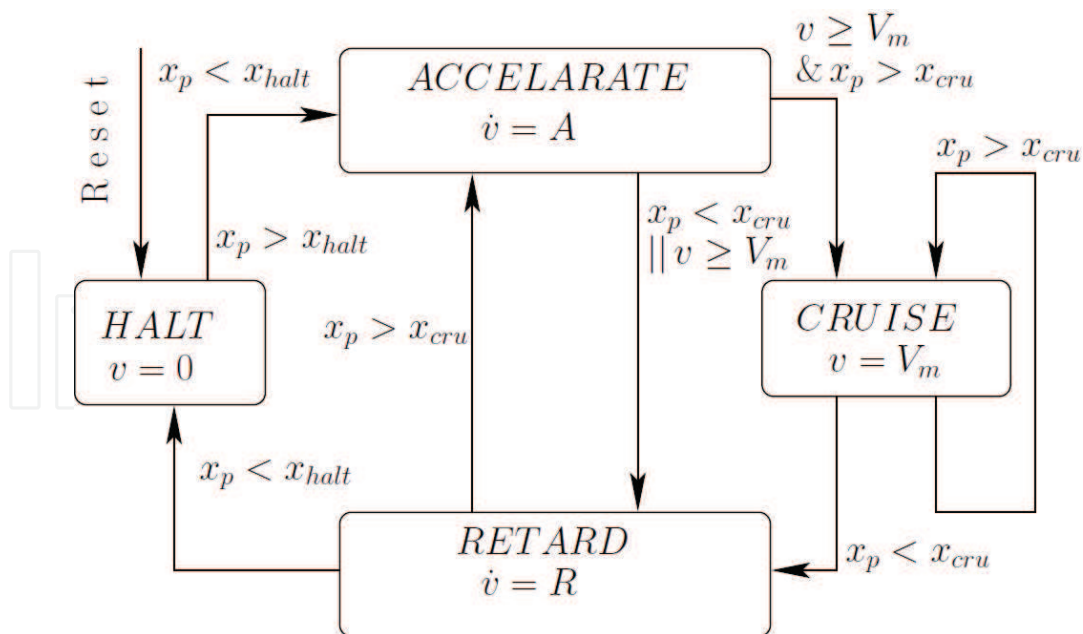


Fig. 7. ACC State Transition Graph

controller samples these states and accordingly sends control signals to various automotive peripheral subsystems to act. These functional tasks include acceleration, retardation or maintaining constant speed of the vehicle. The sensors on the vehicle, keep polling for front and rear vehicular proximity for a continuous update of system state.

7. Introduction to gyroscope model

This section provides an introduction to analysis and modeling of a gyroscope. Gyroscope is a device that is used to measure the angular velocity of a system. It primarily uses the concept of Coriolis force, to translate angular motion to linear motion detection, which is captured in the form of capacitance variation in measurements. The reader is encouraged to look into basic text books [28] [29] applied mechanics and MEMS [30] to understand the concepts of mechanical motion and gyroscope basics. [20] provides a very good summary paper for design and analysis of micromechanical gyroscope. For the clarity of the reader and completeness of the subject, we present a very basic analytical overview of the gyroscope. Figure 8 shows a mathematical model of a gyroscope and its Simulink representation.

A gyroscope can be abstracted as a simple set of coupled differential equations whose output can be equated to a capacitance equivalent to the angular velocity. The reader should note that, simple as it seems, there exists a fundamental difference between this model and most other models based on differential equations. The conventional models would utilize Laplace transform so that the convolution operations in time domain translate to multiplication operation in the frequency domain, thus, resulting in simpler computations. However in the case of gyroscopes, there exist both multiplication and convolution operation in both time and frequency domains. Hence a simplified computation is not possible. The reader is encouraged to explore options to simplify this computationally challenging problem. In this case we resort to a simplified time domain based simulation approach through Simulink.

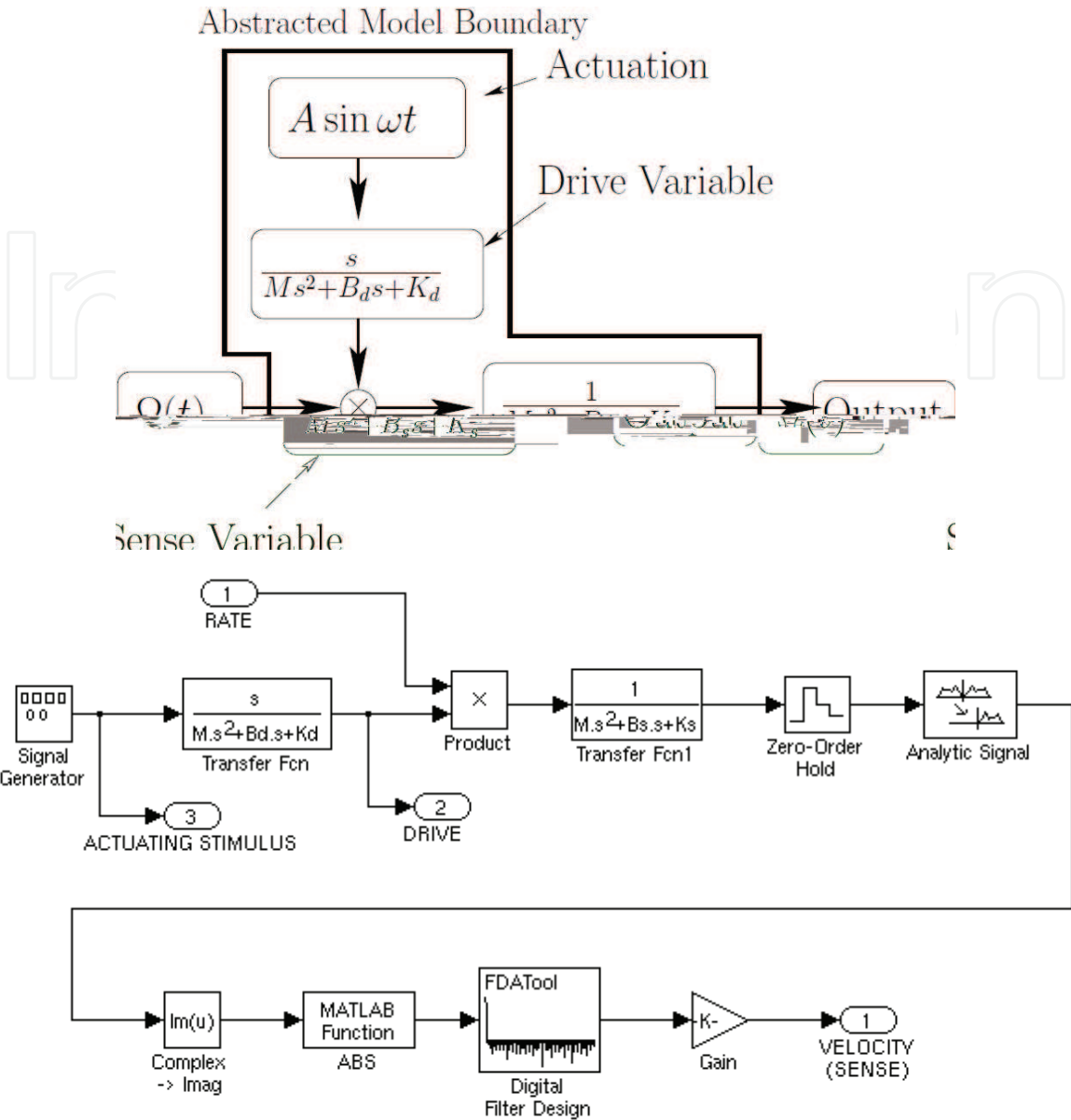


Fig. 8. Gyroscope Model [20] & Simulink Representation

8. Formal verification of ACC system

8.1 Planning the models

Let us first explore how the given system can be casted into the Check Mate framework. It is assumed that the reader is well aware of the Simulink and Stateflow tools. It is a fairly simple task to translate the ACC system requirements into an SSF diagram. These tasks today are routine for practicing system design engineers who program in Simulink/Stateflow in Matlab. The state transition graph of the ACC state machine (FSM) can be added as is, into the Stateflow system of Matlab. For each state in the hybrid automata, there is a need to associate, a real time Simulink model, which continuously modifies the state variables of the system. At each time step during the temporal evaluation of the hybrid system, based on state specific behaviour, conditions associated with each

transition edge are evaluated. An illustration of the same is shown in Figure 9. Simulink representation of a real time system involves simple hook up of maths based models, which are either user created or linked Simulink library instantiations. Similarly Stateflow diagrams can also be easily created through templates available in the Matlab framework. Reader can go through standard Matlab help manuals and tutorials to get a better grasp of these basics. The differential equation pertaining to each state can be easily modelled by Simulink modules. Figure 10 illustrates the integrated ACC system, with assumptions upon availability of the sensor inputs through direct means. Now let us look at the ACC system in more detail. Figure 7 represents its state transition graph.

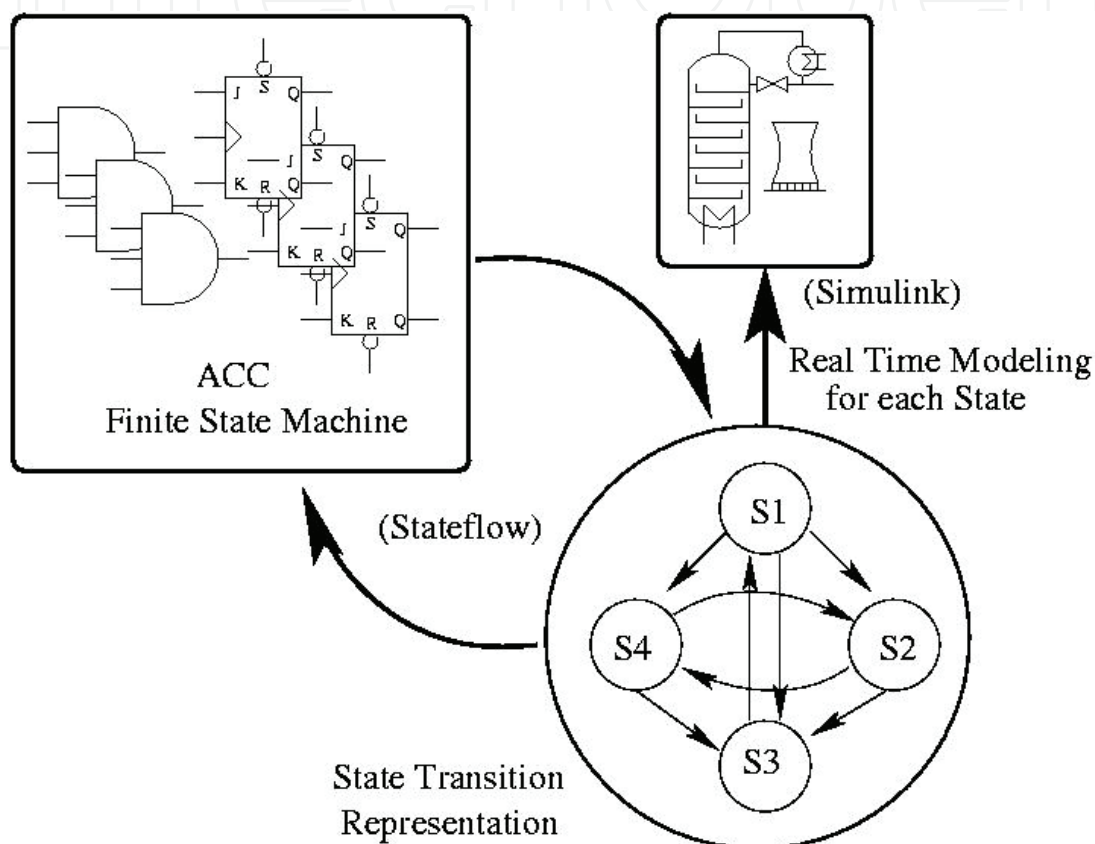


Fig. 9. Representation of a Simulink State flow system

This same representation has been cast into the system shown in Figure 10. Each state shown in the figure, models a real time differential equation defining that state. A switch multiplexes the activation of the state equation in time domain depending on the state decision. The Stateflow sub-box shown in the figure generates the control signal for the switch. This sub-box models the ACC transition graph. This implementation forms one of the simplest representations of an integrated Simulink-Stateflow system. The reader should again keep in mind that this system still does not include the sensor model into the overall model of the ACC system. Notably there are two points to mention. Firstly, there exists a sensor (not shown in the state transition graph), which senses the vehicular velocity, later processed by the controller. The need for the integration of the sensor into the main ACC system arises because due to the fact that, there exists error in measurements taken at the sensor output of the sensor, which may not be accounted into the error-analysis of the formal analysis model. The integration of the sensor helps solve this problem. Secondly, the

sensor processing and the ACC system processing time steps are the same in a Matlab based setup. Hence within the framework of a single discrete event solver, one cannot freeze the simulation time step against the other. This problem is illustrated in Figure 11. As the case in our example, we cannot freeze the ACC system time step progression and wait for the sensor to provide an event signalling completion. However the current versions of the Matlab solvers do not address this requirement. Working within the Matlab setup, can pose challenges, which need not even be addressed. A work around to this problem involves translating one of the time variables to static. In other words, if we can characterize the sensor model, not only can we account the model into the framework, but also perform a formal analysis with respect to the correctness of the sensor functionality.

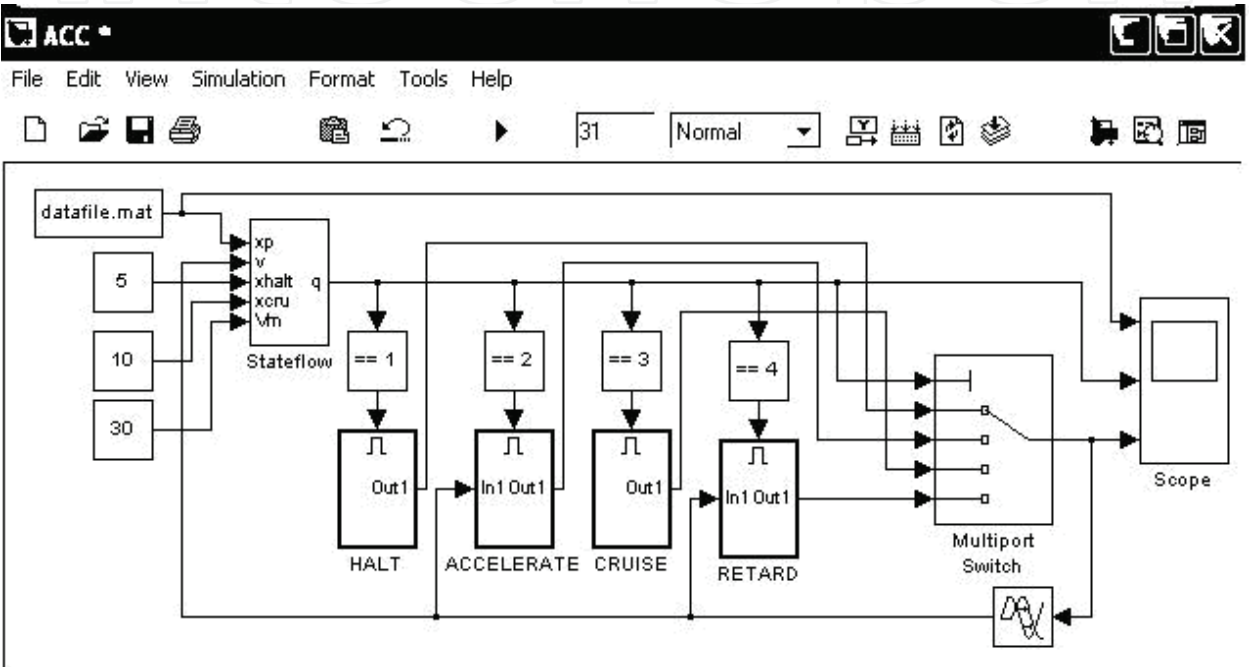


Fig. 10. Representation of an ACC System.

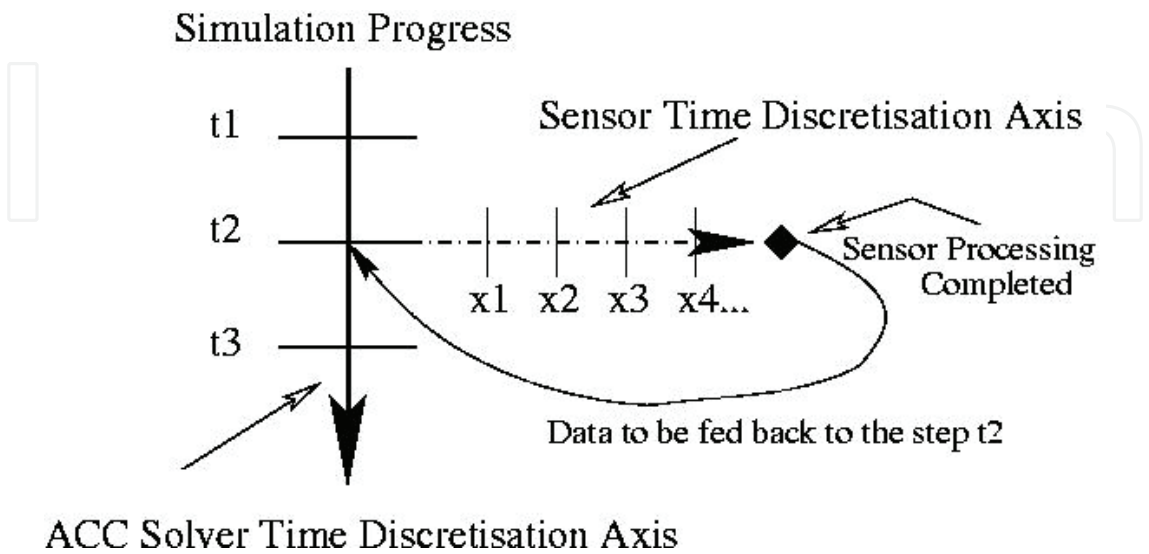


Fig. 11. Multiple Time Discretisation Stack Requirements

components allowed by CheckMate. This along with the multiple time step requirements leads one to explore an alternative method to integrate a MEMS sensor into the hybrid system.

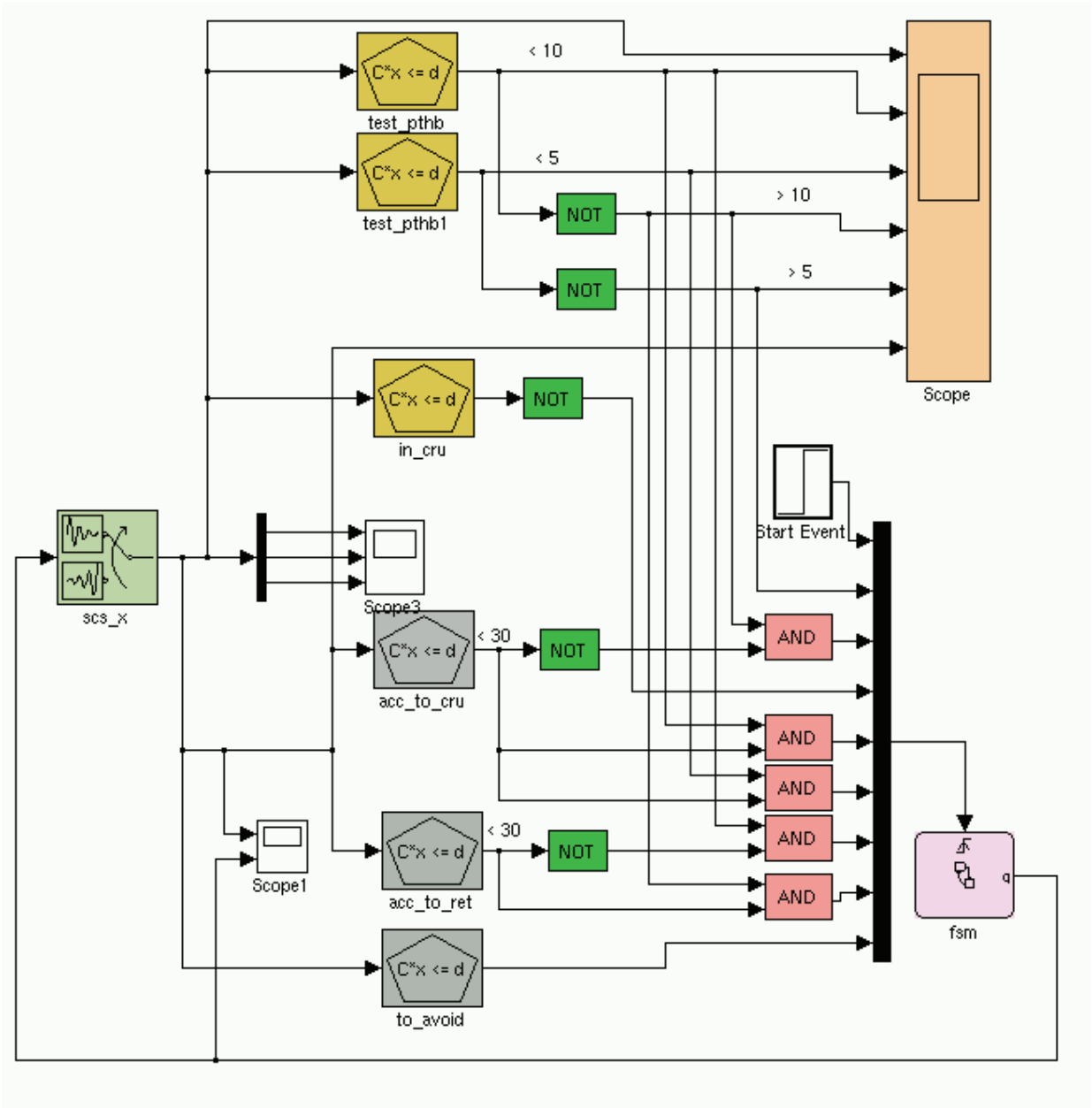


Fig. 13. Check Mate Representation of ACC System

8.3 Integrating sensors

Let us now devote ourselves to the task of integrating a component like a MEMS sensor. We have already discussed in Section 8.1 our reasons for searching for an alternative method to integrate real time sensors. In our case this is the MEMS gyroscope model. The standard SSF model of a gyroscope uses several continuous time domain dynamic components which do not belong to the set allowed by CheckMate. There are two methods possible to approach this problem. First method shown in Figure 14 is to build an independent interrupt driven discrete time solver solution.

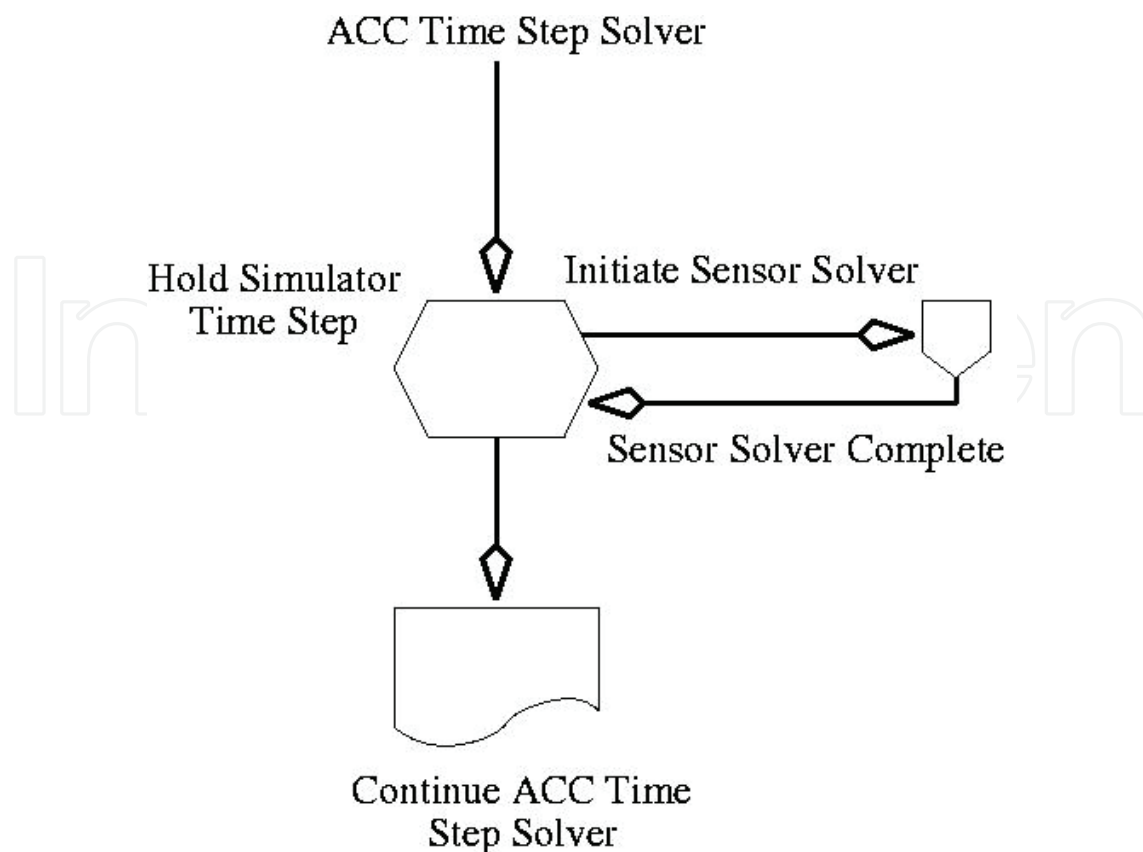


Fig. 14. An Independent Interrupt driven Solver Solution Approach

The second method consists of characterising the time domain behaviour of the sensor and capturing this behaviour as an empirical model. The empirical model can be implemented as a simple look-up-table (LUT) which can be easily interpolated (or extrapolated) as per the input conditions in the field. The data-points for the LUT can be obtained for a range of velocity values by carrying out dynamic simulation on an exact macro-model of the MEMS gyroscope in the SSF framework (Figure 7). To integrate this empirical LUT model of the MEMS gyroscope, it is necessary to make changes in its implementation code. We can access the LUT model through function calls in CheckMate to get the desired outputs and thereby obtain the continuous time trajectories needed for formal analysis. For non-sensor based systems, where there are no multiple time step requirements, we can also use a simulation approach and integrate it with the main system solver.

Readers should note that an LUT based approach will have a limited reach as a solution approach. It is clear that this approach cannot be used for general hybrid systems having dynamic components described with a system of strongly non-linear differential-algebraic equations, as in analog mixed signal design blocks. We can also look at using the exact simulation traces available from the general model in SSF. However they cannot be applied to real time sensor models due to reasons discussed earlier. For other real time components, the exact simulation traces enable the *Explore* and the *Verify* phase to construct accurate flow-pipes, and generate better approximations to these flow-pipes. This is illustrated in Figure 15. The method provides an alternative path to choose between a Simulink or a formal Checkmate model used in the computation. However for simplicity sake, let us restrict ourselves to the LUT based approach.

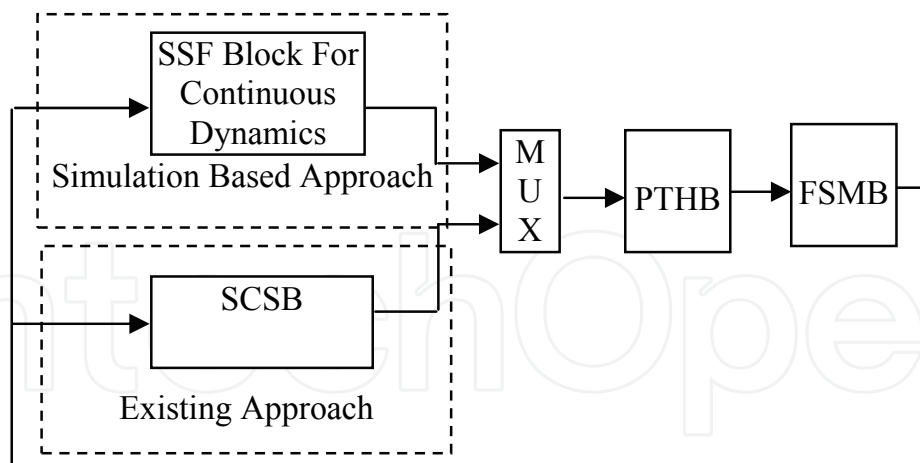


Fig. 15. Hybrid Component Integration Approaches

Figure 15 also includes the LUT macro-model between the switched dynamic block and the PTHB modules. A restriction imposed by CheckMate is that for formal analysis it assumes a hybrid system to be closed. In other words it needs to be a closed loop system. The ACC system model can easily be seen to be open with respect to the velocity of the leading vehicle V_L . This is because the control actions in the hybrid automata of the ACC system, depends on the behaviour of the leading vehicle resulting from changes in its velocity V_L . Regular SSF framework allows modelling of an open hybrid systems. However, it needs some effort to model this in CheckMate. We can model such a scenario by addition of a redundant equation in terms of V_L , V_T and proximity (X_p) in which we render V_L as a parameter (V_T and X_p as the closed system state variables). The equation used is,

$$R = \int (V_L - V_T) dt = X_p.$$

8.4 Verifying the specification properties

Let us now look at some of the controller specifications to be formally analysed:

- P1:** The tracking vehicle should never retard above X_{cru} .
- P2:** The tracking vehicle should never accelerate when $X_p < X_{halt}$.
- P3:** The tracking vehicle should not cruise when $X_p < X_{cru}$.
- P4:** The value of proximity X_p in all states will be always greater than 0.
- P5:** For $X_p < X_{halt}$ tracking vehicle is always in the HALT state.
- P6:** When $X_p > X_{cru}$ tracking vehicle never goes to the HALT state.
- P7:** When $V > V_m$ and $X_p > X_{cru}$ and tracking vehicle is always in the CRUISE state.

The ACC model without the MEMS gyroscope block can be easily verified using CheckMate. Adding the sensor block causes CheckMate to report non-compliance. In the STG (State Transition Graph) of the hybrid automata, five states have been used. The output values of the state-space variables from a previous state become the initial values for the next state continuous dynamics. To get the velocity value of the tracking vehicle to be zero before entering the halt state, we can add an intermediate state where the velocity is brought down to zero. The properties can be represented through ACTL. The same properties can also be verified by simulation. These results are shown in Figure 16.

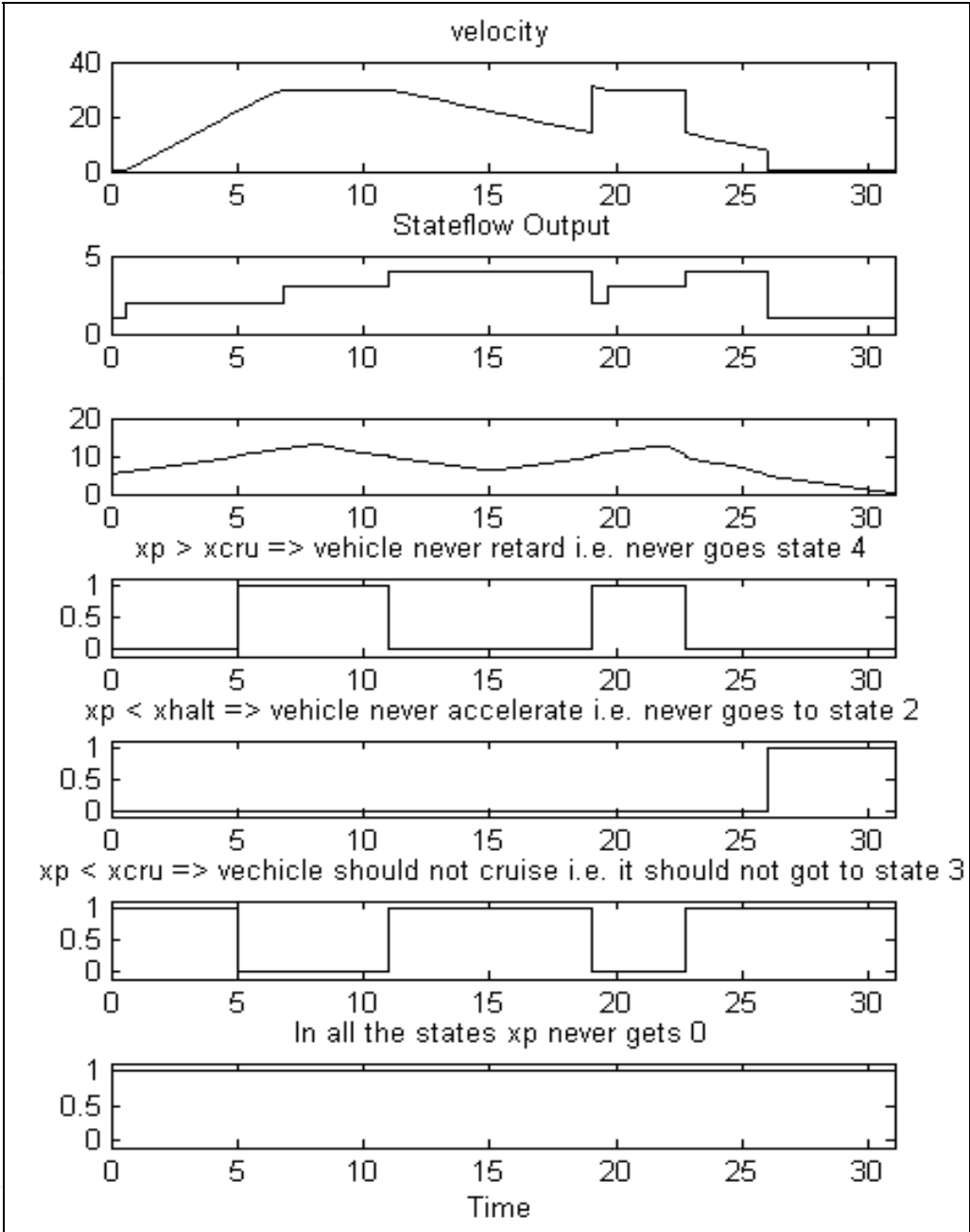


Fig. 16. SSF output of ACC model for different properties

9. Conclusions

In this chapter we give a brief introduction the reader on the basics of formal verification of hybrid systems, which are primarily targeted for automotive systems. CheckMate based approach has been introduced to explore other available hybrid system design and verification frameworks for analysing hybrid system implementation. The reader is then walked through a case study of an adaptive cruise control system, which contains a MEMS based velocity sensor. After reading this chapter, the reader should be able to plan a hybrid system formal analysis platform and apply various methods to integrate complex sensors and other complex real time components.

10. References

- [1] Simulink State-Flow, <http://www.mathworks.com/products/stateflow/>
- [2] C. J. Tomlin, I. Mitchell, A. M. Bayen, M. Oishi, Computational techniques for the verification of hybrid systems, *Proc. IEEE* 91 (7) (2003), 986{1001.
- [3] R. Alur. et. al., Hierarchical modeling and analysis of embedded systems, *Proc. IEEE* 91 (1) (2003) 11{28.}
- [4] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, no. 2, pp. 183-235, 1994.
- [5] D R Sparks, Application of MEMS technology in automotive sensors and actuators; *Micromechatronics and Human Science*, 1998. MHS '98. Proceedings of the 1998 International Symposium on 25-28 Nov. 1998 Page(s):9 - 15
- [6] T. A. Henzinger, "The theory of hybrid automata," *Logic in Computer Science*, vol. 96, no. 1, pp. 278-292, 1996.
- [7] P. Kopke, T. Henzinger, A. Puri and P. Varaiya, What's decidable about Hybrid Automata?, *ACM Symposium on Theory Of Computing*, 1995, 372-382.
- [8] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C*, 2nd ed. Cambridge, UK: Cambridge University Press, 1992.
- [9] E. M. Clarke, A. Biere, R. Raimi, and Y. Zhu. Bounded Model Checking Using Satisfiability Solving, *Formal Methods in System Design*, 19(1):7-34, 2001.
- [10] Bernd Becker, et. al, Bounded Model Checking and Inductive Verification of Hybrid Discrete-continuous Systems, *GI/ITG/GMMWorkshop*, 2002.
- [11] André Platzer and Jan-David Quesel. KeYmaera: A Hybrid Theorem Prover for Hybrid Systems, *Automated Reasoning, Third International Joint Conference, IJCAR 2008, Sydney, Australia, Proceedings*, volume 5195 of LNCS, pages 171-178. Springer-Verlag, 2008.
- [12] T. A. Henzinger, The theory of hybrid automata, *LICS*, *IEEE Computer* (1996) 278-292.
- [13] E. Abraham-Mumm, M. Steffen and U. Hannemann: Verification of hybrid systems: Formalization and proof rules in PVS, *ICECCS*, *IEEE Computer Society* (2001) 48-57.
- [14] "PHAVer: Polyhedral hybrid automaton verifier", <http://www.cs.ru.nl/goranf/>, 2006.
- [15] Eugene A, Thao D, and Maler, O., "The d/dt Tool for Verification of Hybrid Systems", *CAV'02 - Computer Aided Verification*, 2002, Copenhagen, Denmark, July 2002, 365-370, LNCS 2404. <http://www-verimag.imag.fr/tdang/ddt.html>
- [16] F. D. Torrisi and A. Bemporad, HYSDEL --- A tool for generating computational hybrid models for analysis and synthesis problems, *IEEE Transactions on Control Systems Technology*, 2004, Volume 12, 235-249.
- [17] CMU CheckMate website, <http://www.ece.cmu.edu/~webk/checkmate/>
- [18] André Platzer and Jan-David Quesel. KeYmaera: A Hybrid Theorem Prover for Hybrid Systems, *Automated Reasoning, Third International Joint Conference, IJCAR 2008, Sydney, Australia, Proceedings*, volume 5195 of LNCS, pages 171-178. Springer-Verlag, 2008.
- [19] E. Abraham-Mumm, M. Steffen and U. Hannemann: Verification of hybrid systems: Formalization and proof rules in PVS, *ICECCS*, *IEEE Computer Society* (2001) 48-57.
- [20] H. Xie, G. K. Fedder, Integrated microelectromechanical gyroscopes, *Journal of Aerospace Engg.* 6 (2) (2002) 65{75}

- [21] G. Lafferriere, G. J. Pappas, and S. Yovine, "Symbolic reachability computation for families of linear vector fields," *Journal of Symbolic Computation*, vol. 32, no. 3, pp. 231–253, 2001.
- [22] A. Dolzmann, T. Sturm, Redlog : Computer algebra meets computer logic, *ACM SIGSAM Bulletin* 31 (2) (1997) 2{9}
- [23] G. Collins, H. Hong, Partial cylindrical algebraic decomposition for quantifier elimination, *Journal of Symbolic Computation* 12 (1991) 299{328}
- [24] A. Kurzhanski, P. Varaiya, Ellipsoidal techniques for Reachability analysis, in: B. Krogh, N. Lynch (Eds.), *Hybrid Systems: Computation and Control*, Vol. 1790 of LNCS, Springer, 2000, pp. 203{213}.
- [25] A. Chutinan, B. Krogh, and Verification of polyhedral-invariant hybrid automata using polygonal Flow pipe approximations, in: F. Vaandrager, J. H. Van Schuppen (Eds.), *Hybrid Systems: Computation and Control*, Vol. 1569 of LNCS, Springer, 1999.
- [26] T. Dang, O. Maler, Reachability analysis via face lifting, in: T. Henzinger, S. Sastry (Eds.), *Hybrid Systems: Computation and Control*, Vol. 1386 of LNCS, Springer, 1998, pp. 96{109}.
- [27] Chutinan, B. Krogh, Computational techniques for hybrid system verification, *IEEE Trans. Automatic Control* 48 (1) (2003) 64{75.}
- [28] Irving Shames, *Engineering Mechanics: Dynamics*, 4th Edition, Prentice Hall 1997
- [29] Jerry Ginsberg , *Engineering Dynamics* (Hardcover), Cambridge University Press; 3rd edition, 2007
- [30] Stephen D. Senturia , *Microsystem Design* (Hardcover)
- [31] André Platzer and Jan-David Quesel. KeYmaera: A Hybrid Theorem Prover for Hybrid Systems, *Automated Reasoning, Third International Joint Conference, IJCAR 2008, Sydney, Australia, Proceedings*, volume 5195 of LNCS, pages 171-178. Springer-Verlag, 2008.
- [32] R. M. Kirby, G. E. Karniadakis, O. Mikulchenko and K. Mayaram. An Integrated Simulator for Coupled Domain Problems in MEMS, *IEEE Journal of Micromechanical Systems*, Vol. 10, No. 3, September 2001, 379-391
- [33] Jairam S., Kusum Lata, Subir K. Roy and Navakanta Bhat, "Formal Verification of a MEMS Based Adaptive Cruise Control System", *Proceedings of Modeling and Simulation of Micro Systems*, June 2008, Boston, USA.



Motion Control

Edited by Federico Casolo

ISBN 978-953-7619-55-8

Hard cover, 590 pages

Publisher InTech

Published online 01, January, 2010

Published in print edition January, 2010

The book reveals many different aspects of motion control and a wide multiplicity of approaches to the problem as well. Despite the number of examples, however, this volume is not meant to be exhaustive: it intends to offer some original insights for all researchers who will hopefully make their experience available for a forthcoming publication on the subject.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Jairam Sukumar, Subir K Roy, Kusum Lata and Navakanta Bhat (2010). Formal Verification of Hybrid Automotive Systems, Motion Control, Federico Casolo (Ed.), ISBN: 978-953-7619-55-8, InTech, Available from: <http://www.intechopen.com/books/motion-control/formal-verification-of-hybrid-automotive-systems>

INTECH
open science | open minds

InTech Europe

University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China

Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2010 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the [Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License](https://creativecommons.org/licenses/by-nc-sa/3.0/), which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.

IntechOpen

IntechOpen