Robust hardware-software Co-simulation framework for design and validation of Hybrid Systems

1st Surinder Sood
Dept. of Electrical and Computer eng.
University of Auckland
Auckland, NewZealand
surinder.sood@gmail.com

2nd Avinash Malik
Dept. of Electrical and Computer eng.
University of Auckland
Auckland, Newzealand
avinash.malik@auckland.ac.nz

3rd Partha Roop
Dept. of Electrical and Computer eng.
University of Auckland
Auckland, Newzealand
p.roop@auckland.ac.nz

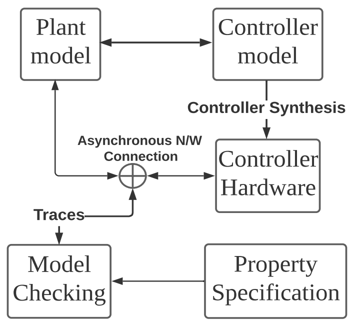
***Abstract*—Model based design of embedded controllers is prevalent across different industries. The final step in model based design is synthesis of hardware (or software) controller and then testing the synthesized controller in closed-loop with the plant model — this is termed as co-simulation. Standard co-simulation approaches use asynchronous communication fabric. However, they are known to suffer from race conditions, jitter, etc, making real-time property validation difficult. Current approaches to co-simulation problems either require complex middle-ware or require synthesis of the controller and plant for synchronous execution. However, these approaches are unsuited for hybrid system control design and validation, as they require the plant model to execute at an arbitrarily small simulation step, while the synthesized controller executes at its own rate if any. The small simulation step slows down the simulation and such a setup does not guarantee level crossing detection. In this paper, we propose a novel Metric Interval Temporal Logic (MITL) based validation and Hardware in Loop (HIL) co-simulation framework, which synchronizes and integrates the controller synthesized in hardware and the plant executing in software. A discrete controller handles a level crossing generated by the plant, which evolves on variable step size. The traces generated from the closed-loop operation of the overall system are used to validate MITL properties. Finally, the controller hardware and the plant model are adjoined via a communication architecture, whose sample time is dependent upon the robustness estimates of the MITL properties, which is necessary to guarantee validation correctness.**

***Index Terms*—Co-simulation, MITL, Robustness, validation, Hybrid Automata (HA)**

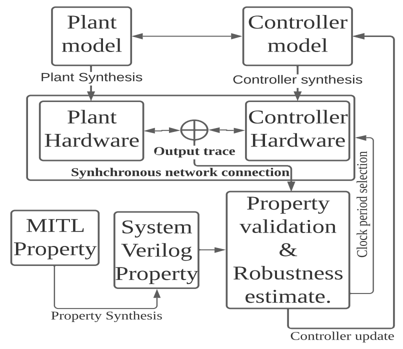
I. Introduction

Model based design of embedded controllers for hybrid systems integrates design, validation and implementation into a single seamless flow [1]. Co-simulation is one of the final steps in the model based design approach; where the synthesized controller is adjoined with the plant *model* for testing and validation. The standard approach to co-simulation, in the model based design flow, is shown in Figure 1a. The synthesized controller and the plant model are adjoined via an asynchronous communication fabric, e.g., the networking stack. However, asynchronous communication may introduce timing jitter and all types of race conditions [2]. A number of solutions have been proposed [3]–[5]. These are either based on the Functional Mockup Unit (FMU)-Functional Mockup Interface (FMI) standard or use Simulink®. But these solutions don't provide a proper platform for hardware-software co-simulation based MITL validation. Design and validation challenges arise when plant and controller are integrated and co-simulated are: ① common lock-step selection on which hardware and software components synchronize, ② level crossing detection to correctly control the physical process, ③faster controller execution than the plant so as that it responds to every plant state, which means that the reaction of controller on its inputs from plant should take no observable time — this is termed as the *synchrony hypothesis* [6] and finally, ④ timing validation of the controller functionality. The current approaches do not address the real-time concerns of controller design and don't guarantee *synchrony hypothesis* either. In order to address these concerns, a recent approach to co-simulation is described in [7] and presented in Figure 1b. In this co-simulation technique, the plant and controller models, *both*, are synthesized in hardware, and are synchronized on a common clock. The real-time MITL properties are validated on line, with the plant and the controller executing synchronously at the same clock frequency. The synchronous execution of the plant and the controller overcome most of the concerns, except one, i.e., correctly capturing all level/zero-crossings. The synchronous execution of the plant model in hardware (c.f. Figure 1b) may lead to missed level-crossings. In case of stiff systems [8], for instance, an arbitrarily small simulation step might be needed to correctly capture all level-crossings, when simulating hybrid plant models [9]. This is achievable in modeling software such as Simulink®with a variable step size solver, but is difficult to achieve, efficiently, when the plant is synthesized in hardware as proposed in [7] (c.f. Figure 1b). Consequently, the existing methodologies do not guarantee correct real-time controller functionality.

In this paper we remedy these issues with the following key **contributions**: ① we provide a novel Hardware in Loop (HIL) co-simulation framework based on HA [10]. The HA plant model is described in Simulink/Stateflow®. The discrete controller is synthesized in hardware and simulated using Modelsim®. ② We propose an efficient communication architecture adjoining the Simulink/Stateflow® plant and the controller hardware. ③ The framework is designed to satisfy the *synchrony hypothesis*, thereby guaranteeing (near) instantaneous response to plant inputs. ④ Robustness criteria



(a) Functional validation technique for non-safety controllers.



(b) Technique for real-time property validation of controllers.

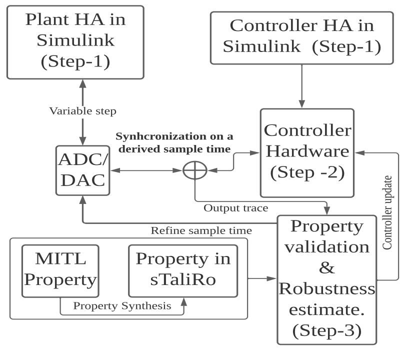

Plant HA in Simulink (Step-1)
Controller HA in Simulink (Step-1)
Variable step
Synhcronization on a derived sample time
ADC/ DAC
Controller Hardware (Step -2)
Output trace
Controller update
Refine sample time
MITL Property
Property in sTaliRo
Property Synthesis
Property validation & Robustness estimate. (Step-3)
(c) Proposed Methodology for validation of Safety Critical Controllers.

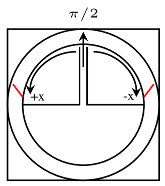
Fig. (1) Figure 1a and Figure 1b shows the current technique for validation of controllers. Figure 1c shows the proposed technique for validation of controllers.

[11] based sample time selection, which is used for data transfer across discrete and continuous domains. ⑤ Validation of MITL properties on the output traces on a range of real-world hybrid system benchmarks which are implemented in the proposed co-simulation framework.

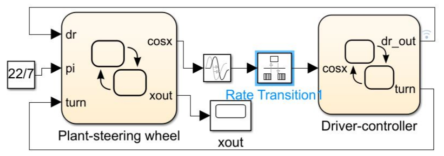
The overall proposed co-simulation methodology is shown in Figure 1c and is described in brief as follows:

- *Step-1*: At the very first step, the plant and controller HA [3] are modeled in Simulink®.
- *Step-2:* The controller is synthesized in hardware, while the plant is modeled in software and we choose Simulink/Stateflow®for that. The output data from the Simulink® model is sampled by Analog to Digital converter (ADC) before sending it to the synthesized digital controller. The most important information such as controller response to level crossings, are exchanged between the plant and the controller at a chosen sample rate based on robustness criteria [11], [12]. The plant uses a variable step Ordinary Differential Equation (ODE) solver to evolve its continuous variables. The controller acts on the plant outputs and controls its behavior based on certain events like level crossings.
- *Step-3 :* This step constitutes MITL property validation. The MITL properties validate real time controller functionality while it executes adjoined with the plant model. The properties are specified in MATLAB using the sTaLiRo [13] validation tool-box. If the property gives a negative robustness value,it is falsified, then either the sample time is decreased or the controller logic is fixed to achieve correct functionality.

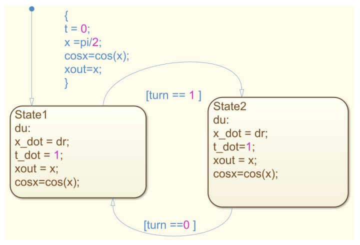
The rest of the paper is arranged as follows: We begin with description of the running example and problem statement in Section II. Then we describe syntax and semantics of HA in Section III, which is followed by controller hardware synthesis technique and co-simulation framework in Section IV.This is followed by benchmarks in Section V where we applied our methodology. We present related work in Section VI and conclusion in Section VII.



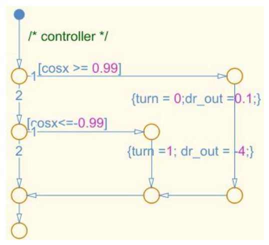
(a) Steering Wheel



(b) Stateflow Impementation of Steering wheel controller.



(c) Steering Wheel Plant



(d) Steering Wheel Controller

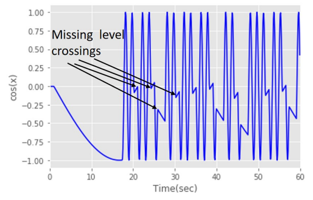
Fig. (2) Steering Wheel of a car and its implementation in Stateflow®

II. Running example and the problem description

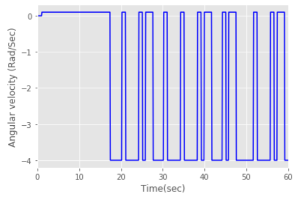
A. Running example

We take steering wheel controller from [14] as our running example. The steering wheel (Figure 2a) of an autonomous vehicle needs to be maintained within the upper half plane shown with red markers. The controller of the steering wheel is an example of sliding mode control without chattering. In Figure 2a, x represents the angular position of the steering wheel. The angle of the steering wheel is initially at $\frac{\pi}{2}$ radians, which is shown as an up arrow in Figure 2a. Following standard convention, clockwise movement of the steering wheel is considered negative and anticlockwise as positive.

The plant model and the adjoined controller are shown in Figure 2b. The Simulink/Stateflow® implementation of the steering wheel plant is shown in Figure 2c, which is controlled by a discrete controller shown in Figure 2d. The plant model has two modes: State1 and State2. It has three inputs and two outputs. The plant inputs are: `dr`, which is the controlled angular velocity, the other one is a constant value representing `pi` while the third one being the `turn` value which is used to move steering wheel in a specific direction. The rate of change of the angle in both the locations is represented as $\dot{x}$ is `dr`. As the steering wheel rotation reaches the left limit which is indicated by the level-crossing condition: `cos(x)` $\leq$ `-0.99`, the output signal `turn` is set to 1, the controller is triggered to produce a clockwise angular velocity of -4 rad/sec. Likewise, the right turn limit is detected when the level crossing condition `cos(x)` $\geq$ `0.99` is reached, in which case the angular velocity selected is 0.1 rad/sec. The two outputs of the plant are `xout` and `cosx`, which are the angular position of steering wheel and the horizontal projection of rotation, respectively. These outputs are used for visualization and debugging purpose. The controller is a collection of few control statements, typically being `if-else` structures. As shown in the Figure 2d, the controller outputs the `turn` and `dr_out` control signals. The output values are based on level crossing conditions which are provided to it as inputs from the plant. It should be noted here that there are no ODEs evolving in the controller. Based on the angular velocity passed by the controller, angular position (`xout`) of the steering is computed. If all the level crossings are detected, we'll get a plot as presented in Figure 5c.

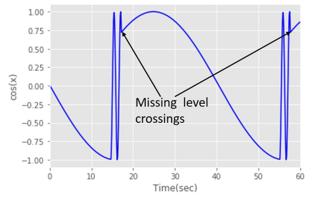


(a) Trace `cos(x)`.

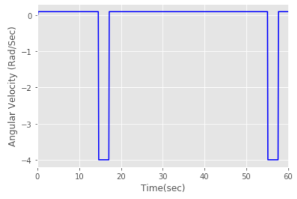


(b) Trace of angular velocity (`dr_out`).

Fig. (3) Traces of Simulink setup at a variable step-size with delay of 1 second between plant model and the synthesized controller; missing level crossings are highlighted.



(a) Trace `cos(x)`.



(b) Trace of angular velocity (`dr_out`).

Fig. (4) Traces generated at a clock period of 0.07 seconds when plant and controller are both synthesized in hardware with synchronous communication, but missing level crossings.

B. Problem Description

The standard technique of adjoining the synthesized controller with the plant model for co-simulation is presented in Figure 1a. The major issue with this approach is the unpredictable timing delays between the plant and the controller due to the asynchronous communication mechanism. The timing problems are highlighted by the traces generated from the co-simulation approach implemented in Simulink/Stateflow® for the steering-wheel running example in Figure 3. The values for `cos(x)` is shown in Figure 3a. The level crossings for this trace are `cos(x)` $\leq$ `-0.99` or `cos(x)` $\geq$ `0.99`. The trajectory of the trace changes abruptly and misses to detect these level crossings multiple times during co-simulation. The output trace for `dr_out` generated by the controller, is shown in Figure 3b; and is incorrect.

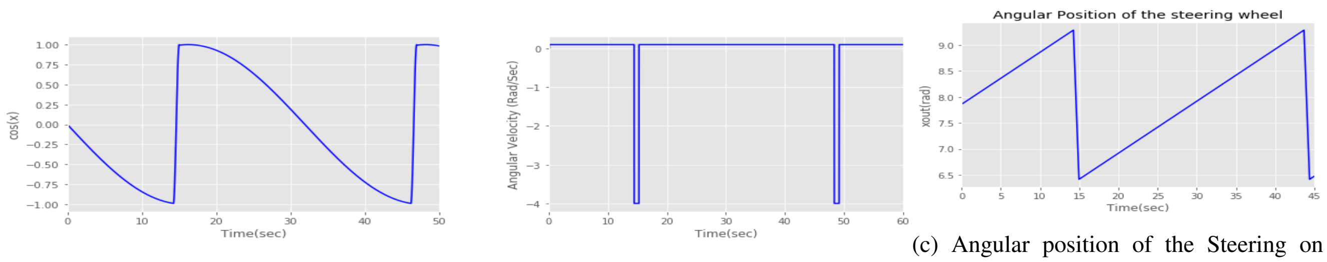
Recently proposed methodology in [7] and presented in Figure 1b overcomes the problem of communication delays between the plant model and the controller by synthesizing both the plant and controller in hardware. The synthesized closed-loop system is then executed synchronously at a fixed clock. The traces, for the running steering-wheel example, generated when the plant and controller are synthesized in hardware are shown in Figure 4. As shown in Figure 4a there are still missing level crossings, as the `cos(x)` trace abruptly changes its trajectory, at few points, and hence change in the value of angular velocity occurs at improper times (Figure 4b). Although delays are handled but level crossing issues impact correct trace generation. The incorrect level crossing detection can be overcome by using the technique described in Figure 1b and reducing the clock period significantly. However, this makes the system inefficient in both execution efficiency and energy consumption. Moreover, in the general case, the step-size might need to be arbitrarily small [15] (e.g., stiff systems), which is impossible to implement in hardware. To remedy this problem, it is required to execute plant on a variable step-size in software and have all the level crossings generated correctly. Figure 5a shows the correct output trace, for the running steering-wheel example, generated using the proposed methodology in Figure 1c. As we can see, none of the level crossings are missed. Co-simulation based validation is carried out on generated traces using Signal Temporal Logic (STL)/MITL properties [16]. For the running example, the level crossing conditions are: `cos(x)` $\leq$ `-0.99` or `cos(x)` $\geq$ `0.99`. The MITL property outlined in Equation 1 depends on the level crossings as it states that whenever within 0 to 20 seconds, value of `cos(x)` is greater than or equal to 0.99 the angular velocity should change to 0.1 *rad/sec* within 1 second. If the level crossings are missed, the property in Equation 1 will be validated on an incorrect trace.

$$\phi : \square_{[0,20]}((cos(x) \geq 0.99) \implies \lozenge_{[0,1]}(dr == 0.1)) \quad (1)$$

III. Design Specifications based on HA

HA is a vehicle to represent dynamical hybrid systems [17]. The overall system process external variables and is divided into a number of operational *modes* or *locations* that describe piece-wise continuous processes. Each location captures certain operational dynamics, which are specified by a set of ODEs that indicate the rate of change of continuous variables. In a given location, the continuous variables evolve with time as long as the location invariant holds.

HA can make transitions based on the outgoing edge *guard* conditions. Once a given outgoing edge *guard* is satisfied, a transition can be performed. Furthermore, *reset* relations



(a) All level crossings detected. (b) Expected angular velocity trace. (c) Angular position of the Steering on detection of all level crossings.

Fig. (5) Level crossing issues fixed with the variable step size and controller Worst case Response Time (WCRT) of 1.29ns as sample time. Figure 5a shows all level crossings detected, Figure 5b shows the correct angular velocity trace, while Figure 5c correct angular position of steering on detection of all level crossings.

update the variables on the outgoing edge guards. The formal definition of HA is provided in Definition 1 following [18].

Definition 1: A Hybrid Automata (HA), H $= \langle Q, X, W, Init, f, Inv, E, G, R\rangle$, where:

- $Q = \{q_0, q_1, \dots, q_n\}$ is the set of locations, with q_0 being the singleton initial location. $\mathbf{Q} = \|Q\|$.
- X is a finite collection of internal variables, with its domain represented as $\mathbf{X} = \mathbb{R}^{\|X\|}$.
- W is the finite set of external variables having domain $\mathbf{W}$, such that $W = W_I \cup W_O$ where W_I represent input variables while W_O represents output variables.
- $Init \subseteq Q \times \mathbf{X} \times \mathbf{W}$ is a set of initial states such that there is exactly one $q_0 \in \mathbf{Q}$, is the singleton initial location.
- $f : \mathbf{Q} \times \mathbf{X} \times \mathbf{W} \to \mathbb{R}$ is a vector field for specifying ODEs.
- $Inv : \mathbf{Q} \to 2^{\mathbf{X}\times\mathbf{W}}$ assigns to each $q \in \mathbf{Q}$ an invariant set.
- $E \subseteq \mathbf{Q} \times \mathbf{Q}$ is a collection of discrete edges.
- $G : E \to 2^{\mathbf{X}\times\mathbf{W}}$ assigns to each $e = (q, q') \in E$ a guard.
- $R : E \times \mathbf{X} \times \mathbf{W} \to 2^{\mathbf{X}\times\mathbf{W}}$ assigns to each $e = (q, q') \in E$, $x \in \mathbf{X}$ and $w \in \mathbf{W}$ is a reset relation.

A. HA of the Steering wheel plant model

The steering wheel plant model HA is based on [14] and presented as a Simulink/Stateflow® implementation in Figure 2. It has two locations in which ODEs evolve. Mapping the steering wheel HA, H_p to Definition 1 we have:

- $Q_p = \{State1, State2\}$
- X_p= $\{x\}$
- $W_p = \{pi, dr, turn, cosx, xout\}$
- $Init_p = (State1,\ \pi/2, \cos(\pi/2), 0)$
- $E_p = \{(State1, State2), (State2, State1)\}$
- G_p: $G_p(State1, State2)$=$\{turn$==$1\}$,$G_p(State2, State1) = \{turn == 0\}$
- The reset relation for the edges are defined as:
  $R_p(State1, State2) = [\phi]$,
  $R_p(State2, State1) = [\phi]$

The invariants of all locations are $True$. The vector field f in the Equation 2 defines a basic ODE.

$$f(q,x) \overset{def}{=} \begin{bmatrix}\dot{x}\end{bmatrix} = \begin{bmatrix}dr\end{bmatrix}, \forall q \in Q \tag{2}$$

B. HA of Steering wheel controller

The steering wheel controller is a single location HA and it has no ODEs. Mapping the controller HA to Definition 1 we have H_{dc}:

- $Q_{dc} = \{C_1\}$
- $X_{dc} = E_{dc} = G_{dc} = R_{dc} = \phi$
- $W_{dc} = \{cosx, dr\_out, turn\}$
- $Inv_{dc} : Inv_{dc}(C_1) = \{(cosx \leq 0.99), (cosx \geq -0.99)\}$

The HA of the controller is realized in Sinulink® as a Stateflow chart described in Figure 2d.

C. Composition of a network of HAs

A network of HAs execute synchronously, exchanging data at every step. Furthermore two HAs H_1 and H_2 can be composed, iff, they have disjoint set of continuous variables and $X_1 \cap V_2 = X_2 \cap V_1 = \phi$, where $X_1 \in H_1$, $X_2 \in H_2$, $V_1 = X_1 \cup W_1$ and $V_2 = X_2 \cup W_2$. The formal definition of composition of two HAs is given in Definition 2

Definition 2:

Let H$_1$=$\langle Q_1, X_1, W_1, Init_1, Inv_1, E_1, G_1, R_1\rangle$ and H$_2$ =$\langle Q_2, X_2, W_2, Init_2, Inv_2, E_2, G_2, R_2\rangle$ be the two HAs such that $X_1 \cap X_2 = \phi$. A synchronous parallel composition of H_1 and H_2 written as $H = H_1 \parallel H_2$, where H = $\langle Q, X, W, Init, f, Inv, E, G, R\rangle$ such that:

- $Q = Q_1 \times Q_2$
- $X = X_1 \cup X_2$ is a set of internal variables.
- $W = \begin{cases} W_{1O} \cup W_{2O}, & \text{if } W_{2O} = W_{1I} \text{ and } W_{1O} = W_{2I}, \\ \text{Where,} & W_{1\text{O}} \in W_1, W_{2\text{O}} \in W_2, \text{are output} \\ & \text{variables, } W_{1\text{I}} \in W_1, W_{2\text{I}} \in W_2 \\ & \text{are input variables.} \\ W_1 \cup W_2, & \text{if } W_{2O} \neq W_{1I} \text{ and } W_{1O} \neq W_{2I}. \end{cases}$
- $Init = Init(q_{01}) \wedge Init(q_{02})$ where $(q_{01}, q_{02}) \in Q$
- $Inv = Inv(q_1) \wedge Inv(q_2), where (q_1, q_2) \in Q$.
- E: For any composite HA it is defined as follows:
  - If $\{(q_1, q_2), (q_1^{'}, q_2)\} \in E$,then $\{(q_1, q_1^{'})\} \in E_1$ with $q_1, q_1{'} \in Q_1$.
  - If $\{(q_1, q_2), (q_1, q_2^{'})\} \in E$, then $\{(q_2, q_2^{'})\} \in E_2$, with $q_1, q_1{'} \in Q_1$ and $q_2, q_2{'} \in Q_2$.
  - If $\{(q_1, q_2), (q_1^{'}, q_2^{'})\} \in E$, then $\{(q_2, q_2^{'})\} \in E_2$ with $q_2, q_2{'} \in Q_2$.
- G : For any transition in the composite HA guards are defined as:

  - $G\{(q_1,q_2),(q_1^{'},q_2)\} = G(q_1,q_1^{'})$,with $q_1, q_1{'} \in Q_1$.
  - $G\{(q_1,q_2),(q_1,q_2^{'})\} = G(q_2,q_2^{'})$,with$(q_2,q_2^{'}) \in Q_2$.
  - $G\{(q_1,q_2),(q_1^{'},q_2^{'})\} = G(q_1,q_1^{'}) \wedge G(q_2,q_2^{'})$,with $q_1, q_1{'} \in Q_1$ and $q_2, q_2{'} \in Q_2$.
- $R : E \times \mathbf{X} \times \mathbf{W} \rightarrow 2^{\mathbf{X} \times \mathbf{W}}$, assigns to each $e = (q, q') \in E$, $x \in \mathbf{X}, w \in \mathbf{W}$ is a reset relation.

Remark 1:
When we have a common interface between the composing HAs such that output variable of one HA is input to the other and vice-versa, in that case, all external variables become internal shared variables, such that, $X = X_1 \cup X_2 \cup W_{1O} \cup W_{2O}$, and $W = \phi$. Such a composed HA is called Well Formed HA. On the contrary when there is no common interface, then the external variables of composed HA is simply the union of external variables of composing HA excluding any shared variables.

D. Composition of HAs of steering wheel and controller

We apply Definition 2 to compose the HAs of our running example. The HAs of steering wheel plant and controller are already presented in Section III-A and III-B. The composition of these HA is defined as follows:

- $Q = Q_p \times Q_{dc}$,
- $X = [\{x\}]$
- $W = [pi, xout, dr\_out, turn, cosx]$
- $Init = [\{state1, C_1\}]$,
- $Inv : Inv(state1, C_1) = \{(cosx \leq 0.99), (cosx \geq -0.99)\}$. $Inv(state2, C_1) = \{(cosx \leq 0.99), (cosx \geq -0.99)\}$.
- $E = [\{(State1, C_1), (State2, C_1)\}$, $\{(State2, C_1)$,$(State1, C_1)\}]$
- $G : G[\{(State1, C_1), (State2, C_1)\}] = (turn == 1)$, $G[\{(State2, C_1), (State1, C_1)\}] = (turn == 0)$
- The reset relation for the edges is as follows: $R\{(State1, C_1), (State2, C_1)\}$=$[dr\_out = 0.1; turn = 0;]$, $R\{(State2, C_1), (State1, C_1)\}$=$[dr\_out$=$-4; turn = 1;]$

Theorem 1: Let H$_1$=$\langle Q_1, X_1, W_1, Init_1, Inv_1, E_1, G_1, R_1\rangle$ and H$_2$ =$\langle Q_2, X_2, W_2, Init_2, Inv_2, E_2, G_2, R_2\rangle$ be the two HAs such that the composition $H_1 \parallel H_2$, is fully closed system represented by a well formed HA, H,where H = $\langle Q, X, W, Init, f, Inv, E, G, R\rangle$. Then, the fully closed system represented by this HA is deterministic for all reachable states such that $(q, x) \in H$, if it satisfies the following:

1 It contain a unique `Guard` for every edge connecting two states in a reachable set.
2 If $(q, q^{'}) \in E$, and $(q, q^{"}) \in E, with, q^{'} \neq q^{"}$,then, $x \notin G(q, q^{'}) \cap G(q, q")$.
3 If $(q, q^{'}) \in E$, and, $x \in (q, q^{'})$, then R contains at most one element.

Proof 1: We refer to [17], [18] for it's proof.
We obtain a fully closed system while we compose HAs of our running example. The outputs of controller are hidden under composition and are shared between plant and the controller.

IV. Co-simulation framework

A. Modeling the plant HAs in Simulink/Stateflow®

We synthesize steering wheel plant model in Simulink® and Stateflow® from its respective HA. The HA of steering wheel plant is described in Section III-A while its Simulink/Stateflow®model is shown in Figure 2. The operational details of the plant are already described in Section II.

B. Hardware synthesis of the controller

Controllers integrated with their respective plant models need to be fast and efficient in execution. To achieve this, each controller HA is mapped into a behavior preserving discrete control logic which commonly use control flow statements. These control structures generate the control decision whenever any of the level crossing conditions are detected during evolution of the plant. The benefit of having discrete controllers is that ① the responses generated by them are faster within their WCRT which is based on the controller critical path when synthesized in hardware. The WCRT of each controller for all our benchmarks tested is specified in Table I. In order to guarantee `synchrony hypothesis`, the plant execution is bound by the minimum step whose value is greater then the WCRT of the respective controller. ② these discrete controllers generate outputs based on present input values. It should be noted that data exchange with continuous domain happens at a specifically chosen sample time and is described in Section IV-D.

The inputs to controller describe the sensitivity list of the control statement, and, based on its value a control decision is made by the controller. These control structures get synthesized as multiplexers. All the logic in controller uses 32 bit fixed point numbers.

C. Design and implementation of the Hardware-Software Co-Simulation Framework

The co-simulation framework consists of a plant model implemented in Simulink® and a controller model which is synthesized in hardware as a discrete logic unit. The outputs of a discrete controller only relies on inputs present at a given time instant. Furthermore, the plant operates in a continuous domain, so any outputs of the plant need to be sampled before they are passed to the controller. We use Analog to Digital converter (ADC) which samples continuous data on a sample time and then pass it to the controller, while the reverse path is handled by Digital to analog converter (DAC). Apart from this there is a Schmitt trigger to handle signal loss or delays and is described in Section IV-C1. The setup is shown in Figure 6.

1) Handling Asynchronous communication: When both plant and controller run at their own pace without any coordination, this gives rise to many issues like race conditions etc., eventually, communication between the plant and controller goes asynchronous. Race conditions [2], [19] add to such random delays. A race condition is triggered when an output depends on timing of several events. For example,

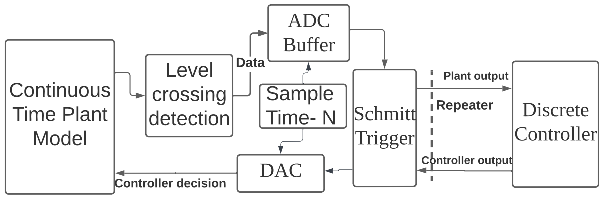


Fig. (6) Integrating Continuous Time (CT) and Discrete Time (DT) models for Hardware software co-simulation.

whenever one variable is updated by different drivers at the same time gives rise to a race condition which leads to intermediate signal values. To handle these situations, we propose the use of Schmitt trigger [20]. A Schmitt trigger is a comparator with hysteresis and can be plugged at the plant boundary. For a given signal to be transferred, two voltage levels are defined: upper threshold voltage and a lower threshold voltage. Whenever a signal value remains below the lower threshold voltage then a logic zero is generated at the output, on the contrary, whenever the signal value remain above the high threshold voltage then a high voltage value is generated. The threshold voltage value (high and low) can be any arbitrarily value as required. This logic helps in eliminating race conditions and provides a clear discrete output over the wire. In-order to eliminate transport delays over the wire, repeaters can be used on specific points in the wired connection between plant and controller. As shown in Fig. 6, a Schmitt trigger is integrated at the interface of discrete domain. This helps correct the signal value coming out from the analog domain before it is passed to the other domain and a repeater is installed on the wire connecting two domains to help prevent signal loss.

D. Sample time selection for hardware software co-simulation

The use of ADC helps sample and transmit data from continuous to discrete domain. As shown in Figure 6, plant HA modeled in Simulink®generates the data which is processed by a level crossing detection block. Upon detection of level crossings the data generated is sent into the ADC single entry buffer which acts as a continuous amplitude and discrete time sampler. On the selected sample time this data is read from the buffer and then transmitted to the controller. To transmit data across domains, a common sample time is chosen. The plant behavior and the controller response to the level crossings is sampled at this sample time. This means that the sample time value must be chosen adequately.

The criteria for sample time selection is based on ① level crossing generation and detection, and the ② justification of MITL properties with positive robustness values. We use WCRT of the respective benchmark controllers as the designated sample time. The WCRT values give positive robustness as specified in Table I. Robustness is described in Section IV-E.

E. Robustness estimate of MITL properties

Hybrid systems are continuous time systems, the validation of such systems is done on discrete traces. Consequently, we must guarantee that any MITL property that holds on discrete trace also holds on continuous trace. In this context, whatever sample time we select, must be chosen such that the MITL property under consideration must be satisfied on both discrete and continuous traces. In this section we describe the robustness estimate [12] technique used to select such a sample time/step size for the co-simulation framework. Robustness [11], [12] is defined as positive radius of neighborhood that can be fitted around a signal without changing its truth value and the maximum value of this radius is called the robustness degree. Any function that maps an MITL property ψ and a discrete output trace $x[t]$ as a real valued number defines the robustness estimate. A very large positive real number means that any property ψ which holds on the discrete trace $x[t]$ also holds easily on the continuous trace $x(t)$, while a small positive value means that the property ψ is close to being violated on the continuous trace. A non-positive number indicates that the property ψ is violated on continuous trace. We use the robustness estimation technique from [11], [12]. According to this technique, the robustness estimate depends on three basic conditions: ①All ODEs in HA are Lipshictz continuous, ② the sample time selected should be less than one-third of the difference of supermum (maximum) and infimum(minimum) of the non-singleton timing intervals specified for any given MITL property ψ. This constraint bounds the maximum sample time for any given property,and ③the trace length of the co-simulation should be greater than the longest timing interval of any MITL property.

In our co-simulation framework we write MITL properties using S-TaLiRo [13] in Matlab® which are then justified in the co-simulation environment. If any property is falsified, then reason for this can be some design issue or because of wrong sample time selection. After fixing any design issues, if the property is still falsified, then we fix the sample time. We select the optimal sample time which gives the maximum robustness. On top of this if different properties give optimal robustness on different sample times then the smallest of those sample times will give all the properties justified with optimal robustness. This also applies to the methodology in Figure 1b, but it does not guarantee detection of all level crossings even with optimal robustness.

The co-simulation framework for our running example is shown in Figure 7. The plant model is implemented in Simulink® while the controller model is implemented in System Verilog and is simulated by ModelSim®. The data generated by Simulink® is received by modelsim® after converting it into a compatible data type. For example, the `turn` signal although being a double type in simulink® only outputs either a one or a zero value is accordingly converted into System Verilog (SV) *Boolean* data type. On the other hand the SV outputs `dr_out` value in 32 bit signed fixed notation is converted into `double` data type. The block labeled *ADC* handles all the analog to digital conversion at a sample time selected based on the criteria specified above.

TABLE (I) MITL Properties for different benchmarks

S.No.	Informal description of MITL Property	MITL Specification	Robustness	Sample time (ns)	ODE Type
		Steering Wheel Controller properties			Linear ODEs.
1	Whenever cosx $\geq$ 0.99 then dr == 0.1 rad/sec	$\psi = \Box_{[0,20]}$ (($\neg$ p1) $\vee$ $\Diamond_{(0,1]}$ (p2)). Here, p1: cosx $\geq$ 0.99 and p2: dr = 0.1	1.9808	1.29	
2	Whenever cosx $\leq$ -0.99 then dr == -4 rad/sec	$\psi = \Box_{[0,20]}$ (($\neg$ p1) $\vee$ $\Diamond_{(0,1]}$ (p2)). Here, p1: cosx $\leq$ -0.99 and p2: dr == -4	4.1	1.29	
		Switch Tank Controller			Linear homogeneous ODEs.
1	Whenever CT_0 is zero within 4 timeunits then CT_1 is selected within 3 time units	$\Box_{[0,3]}$ (($\neg$ p1) $\vee$ $\Diamond_{(0,2]}$ (p2)). Here, p1: CT_0 ==0 and p2: CT_1 == 1	1.65	1.285	
2	Whenever $x_2 \leq 0.25$ within 3 timeunits then CT_1 ==1 within 3 time units	$\Box_{[0,2]}$ (($\neg$ p1) $\vee$ $\Diamond_{(0,3]}$ (p2)). Here, p1: $x_0 \leq 0.25$ and p2: CT_1 == 1	1.1	1.285	
		Temperature Controller in a Nuclear Reactor			Linear ODEs.
1	whenever rod_1 is selected then temperature in the reactor drops below 6 deg within 8 time units.	$\psi = \Box_{[0,6]}$ (($\neg$ p1) $\vee$ $\Diamond_{(0,8]}$ (p2)). Here, p1: Rod_1 is selected. p2: $\theta \leq 6$	6	1.4	
2	whenever rod_2 is selected then temperature in the reactor drops below 6 deg within 8 time units.	$\psi = \Box_{[0,7]}$ (($\neg$ p1) $\vee$ $\Diamond_{(0,8]}$ (p2)). Here, p1: Rod_1 is selected. p2: $\theta \leq 6$	6	1.4	
		Train Gate Controller			Non-Homogeneous linear ODEs.
1	Globally whenever Up is 1 every 16 time units then within every 15 to 30 Time units gate should be opened.	$\psi = \Box_{[0,16]}$ (($\neg$ p1) $\vee$ $\Diamond_{(15,30]}$ (p2)). P1: Up==1 P2: gate==1.	4.328	1.41	
2	Globally whenever Down is 1 and up=0 every 6 time units then within every 1 to 15 time units gate is down.	$\psi = \Box_{[0,6]}$ (($\neg$ p1) $\vee$ $\Diamond_{(1,15]}$ (p2)). P1: Down==1 P2: gate==0.	0.9343	1.41	
		Water Tank Heating System			Stiff linear ODEs.
1	Globally whenever OFF is 1 within every o to 2 time units then within every 0 to 2 Time units ON Should be De-asserted.	$\psi = \Box_{[0,2]}$ (($\neg$ p1) $\vee$ $\Diamond_{(0,2]}$ (p2)). P1: OFF==1 P2:ON==0.	1	1.27	
2	Globally whenever OFF is 1 withing 2 time units then Temperature drops below 99 within 2 time units.	$\psi = \Box_{[0,2]}$ (($\neg$ p1) $\vee$ $\Diamond_{(0,2]}$ (p2)). P1: OFF==1 P2: Temperature <0.	5.86	1.27	

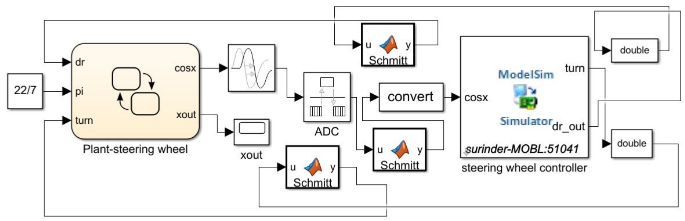


Fig. (7) Co-simulation framework of Steering wheel Controller

V. Benchmarks Tested

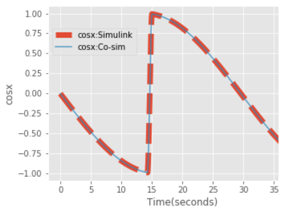
We implemented our methodology on five benchmarks which are presented below. The WCRT of controllers of these benchmarks are computed on arbitrarily selected device family (xczu7ev-ffvc1156-2-e) of Zynq-Ultrascale+ZCU106 evaluation platform in Vivado [21] and is their critical path. We show in every benchmark, our framework detects all level crossings and both plant and controller execute such that all race conditions and any eventual asynchronous communication are eliminated. We also show correlation of traces generated by the Simulink® model and it's implementation in the hardware software co-simulation framework. We used interpolation to fill in the missing trace values because of the different time steps chosen by the variable solver while we establish correlation between Simulink® model trace and the implementation trace generated by hardware software co-simulation framework. For example, in case of steering wheel controller, trace from Simulink® model and co-simulation framework have the trace values on different times as shown in Table II. As can be seen that after a time step of 3.16E-9, the next step chosen by co-simulation framework is 6.31E-09, while the Simulink® model chose 9.47E-9. The missing trace value for either of the traces are computed using interpolation techniques. In this case we compute missing trace for the Simulink® model for time 6.31E-9. Similarly, we computed missing trace at time 9.47E-9 in co-simulation framework. All the benchmark controllers have single state HA models which are implemented in hardware as pure combinational circuit.

TABLE (II) Example snippet from Steering wheel controller explaining different trace values between Simulink® and co-simulation framework.

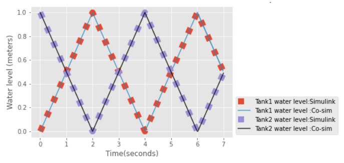
Time1	Cosx trace from Simulink	Time2	Cosx trace from co-simulation framework
0	-0.0032	0	-0.00323
3.16E-9	-0.00316121790318423	3.16E-9	-0.00316121790318423
9.47E-9	-0.00416096702073669	6.31E-9	-0.00316121790318423
11.5E-9	-0.00417096702073669	11.5E-9	-0.00416096703073669

A. Steering Wheel Controller

The MITL properties written for this benchmark are shown in Table I. We choose controller WCRT(1.29ns) as sample time to get the optimal robustness and have all level crossings detected. When of any level crossings are detected, the controller responds instantaneously (within it's WCRT) with the corresponding decision and action, thus guaranteeing the synchrony hypothesis. The plant samples controller response at this robust sample time. Consequently, it takes inter-location transition and continue evolving its ODEs. The minimum step chosen by the plant is bound by a value greater than the WCRT of the controller (which is 1.29 ns in this case), in order to guarantee the fact that controllers respond faster than the plant execution. The HA of steering wheel plant is already presented in Section III. We also compared the co-simulation traces with traces of Simulink® only model. We found exactly similar traces especially at points where level crossings occur. Moreover the correlation coefficient is found to be nearly unity in all the cases, thus concluding that Simulink® and hardware software co-simulation framework behave in similar way. This confirms the accuracy of our approach. Table III shows the correlation co-efficient between the traces generated for `cosx` by Simulink® and hardware software co-simulation framework. The traces are generated by selecting different variable step solvers available in Simulink®. One such trace for `cosx` evolution obtained from the Simulink and hardware software co-simulation framework using `ode45`



(a) Steering Wheel Controller: Trace comparison between Simulink®and Co-sim Framework using `ode45` solver .



(b) Switch Tank Controller: Trace comparison between Simulink®and Co-sim Framework using `ode45` solver.

Fig. (8) Trace comparison between Simulink®and Co-sim framework for Steering wheel and Switch tank controller.

solver is shown in Figure 8a. The figure shows similar traces obtained in both the cases.

TABLE (III) Correlation of `cosx` Evolution between Simulink® and Co-Sim models under different solvers.

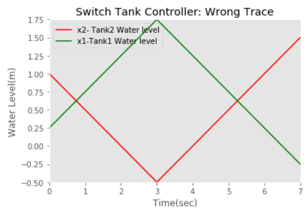
Co-Simulation and Simulink Solver	ode23tb	ode45	ode23	ode113	ode15s	ode23s	ode23t	daessc	odeN
Correlation Co-efficient	0.99	0.99	0.99	0.99	0.99	0.99	0.996	0.996	1

B. Switch Tank Controller

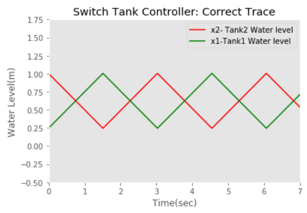
A switch tank system [8] consist of two water tanks. Water leaks from both the tanks at a constant rate. Water level in these tanks is represented by continuous variables x_1 and x_2 respectively. The job of the controller is to keep water level above a threshold level which we have taken as 0.25m for both the tanks. Whenever a level crossing condition $x_1 \leq 0.25$ is met for first tank then the water inflow is switched towards it instantaneously, at this time the water should be in the second tank above a threshold level which is taken as 1m in this example. Similar approach is applied for second tank whose level crossing condition is $x_2 \leq 0.25$. The ODEs for change of water level in one of tanks, is described in Equation 3.

$$f(x) \overset{def}{=} \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} w \text{ - } v_1 \\ -v_2 \end{bmatrix} \tag{3}$$

Here, w is constant flow of water into the tanks and is taken as one, while, v_1 is rate of water outflow from the first tank and v_2 is rate of water outflow from the other tank. We have taken $v_1 = 0.6$ while $v_2 = 0.5$ The controller does not evolve any ODEs and accepts water level in tanks as input and produces discrete values (which are used to control the water flow to one of the tanks) as outputs sent to the plant. The asynchronous communication and the missed level crossing leading to wrong trace is shown in Figure 9a. This is fixed (shown in Figure 9b) by choosing a robust sample time of 1.285 ns (plus techniques specified in our co-simulation framework) which is the controller WCRT as well, and on this sample time value the properties mentioned in Table I are robustly justified with maximum positive robustness. The synchrony hypothesis is guaranteed by choosing the minimum step size of the plant to a value greater than the WCRT (1.285ns) of the controller. The correlation co-efficient for water level trace x_1 obtained from hardware software co-simulation framework and Simulink® model is shown in Table IV. The value is nearly



(a) Improper water level evolution because of asynchronous communication.



(b) Correct trace after fixing asynchronous communication.

Fig. (9) Figures showing wrong trace because of missed level crossings and all level crossing detected while doing synchronous communication.

unity, which confirms similar behavior between the Simulink model and the co-simulation framework. Trace comparison between Simulink® model and implementation traces for water level in both tanks is shown in Figure 8 using `ode45` solver which also show similar traces obtained in both the cases.

TABLE (IV) Correlation of x_1 (water level in Tank_1) evolution between Simulink® and Co-Sim models under different solvers.

Co-Simulation and Simulink Solver	ode23tb	ode45	ode23	ode113	ode15s	ode23s	ode23t	daessc	odeN
Correlation Co-efficient	0.99	0.99	0.99	0.99	0.99	0.996	0.996	0.99	0.9996

C. Temperature Controller in a Nuclear Reactor

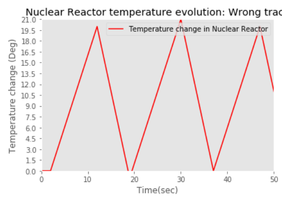
This benchmark is presented in [22]. There are two control rods in a nuclear reactor which act as coolants. The coolant temperature must be kept within the range [θ_m, θ_M]. When the temperature reaches a maximum value of θ_M, the tank is cooled with one of the rods and only that rod should be selected for which the time elapsed since it was last used is $\geq$ T time units. The rate change of temperature in the nuclear reactor with rod_1 is v_2, while with rod_2 is v_3. x_1 and x_2 represent the respective time since a specific rod was used previously. The temperature in the nuclear reactor rise at the rate v_1 while θ represents rate at which temperature change.

The ODE for the plant is described in Equation 4.

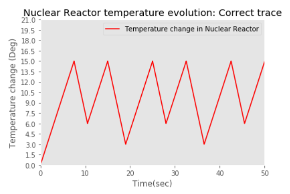
$$f(\theta, x_1, x_2) \overset{def}{=} \begin{bmatrix} \dot{\theta} \\ \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} v_i \\ 1 \\ 1 \end{bmatrix} \tag{4}$$

The controller model does not evolve any ODE and controls the plant by selecting the relevant control rod to be used to cool the reactor. The asynchronous communication between plant and the controller will lead to missed level crossings leading to an incorrect trace. We show this in Figure 10a.

The robust sample time chosen for the nuclear reactor is the WCRT of the controller(1.4 ns) on which MITL properties proven at this sample time are shown in Table I. The synchrony hypothesis is guaranteed by choosing the minimum step size of the plant to a value greater than the WCRT (1.4ns) of the controller. The correlation co-efficient for nuclear reactor temperature (θ) trace obtained from hardware software co-simulation framework and Simulink® model is shown in Table V. The near unity value confirms similar behavior between the Simulink® and hardware software co-simulation framework



(a) Wrong Trace of temperature evolution.



(b) Correct trace after fixing asynchronous communication.

Fig. (10) Figures showing Correct and incorrect traces in Nuclear temperature controller.

for this benchmark. We show these traces obtained for this benchmark using `ode45` solver in both Simulink® model and in hardware software co-simulation framework in Figure 11a. The figure shows similar traces for temperature (θ) obtained in both the cases.

TABLE (V) Correlation of θ (Nuclear reactor temperature) evolution between Simulink® and Co-Sim models under different solvers.

Co-Simulation and Simulink Solver	ode23tb	ode45	ode23	ode113	ode15s	ode23s	ode23t	daessc	odeN
Correlation Co-efficient	1	1	0.999	0.99	1	1	1	1	1

D. Train Gate Controller

A train gate controller is a hybrid system based on [23] that consists of a train running on a circular track. The traffic movement across the track is controlled by a Gate. Gate is opened or closed by a controller based on the train's movement.

The ODE for train is defined as :

$$f(y) \overset{def}{=} \begin{bmatrix}\dot{y}\end{bmatrix} = \begin{bmatrix} v_f \\ v_s \end{bmatrix} \tag{5}$$

Here the continuous variable `y` represents the position of the train which change at the rate of v_s or v_f depending on it's current position. When level crossings are detected then this information is passed to the controller so as to decide on the gate movement. When the position of train equals 5 units then the `down` signal is asserted indicating the gate to move down. Similarly, when train position equals 15 units then the `up` signal is asserted, when train position equals 25 units then both `up` and `down` are reset.

The ODE for the gate is is described in Equation 6.

$$f(x) \overset{def}{=} \begin{bmatrix}\dot{x}\end{bmatrix} = \begin{bmatrix} x \\ \frac{1-x}{2} \\ 5 - \frac{x}{2} \end{bmatrix} \tag{6}$$

Here the gate position changes at the rate of `x` which has different values based on it's position.

If there is asynchronous communication between the plant and controller, then the gate movement is impacted. One such trajectory for train and gate is shown in Figure 12a. Here the gate does not open or close at the right time, because level crossings are not generated at right time, eventually controller is unable to respond on level crossings as expected. When the plant and controller are synchronized on a sample time, which we select based on robustness criteria described previously, all level crossings are generated and detected, eventually controller responds at correct timestamps. Consequently, gate opens and closes at proper time. One such correct trace is shown in Figure 12b. The MITL properties listed in table I are justified with a sample time which is the WCRT of the controller (1.41ns). The synchrony hypothesis is guaranteed upon setting the lower bound on the step size of the plant to a value greater than the WCRT (1.41ns) of the controller. The correlation co-efficient (which is near unity) for train position (`y`) trace obtained from hardware software co-simulation framework and Simulink® models is shown in Table VI, confirms similar behavior between the Simulink® and hardware software co-simulation framework. We also show (in Figure 11b) similar traces of train position obtained from Simulink® and hardware software co-simulation framework using `ode45` solver.

TABLE (VI) Correlation of `y` (train position) evolution between Simulink® and Co-Sim models under different solvers.

Co-Simulation and Simulink Solver	ode23tb	ode45	ode23	ode113	ode15s	ode23s	ode23t	daessc	odeN
Correlation co-efficient	1	0.997	1	1	0.99	1	1	0.997	1

E. Water Tank Heating System

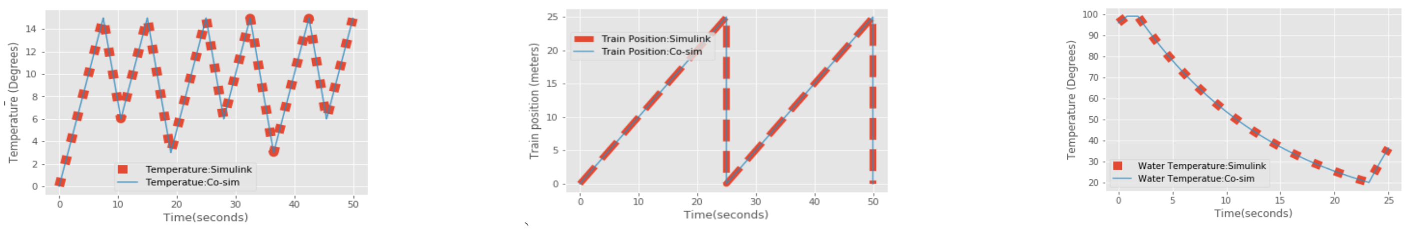
A water heating system [24] consists of a water tank with a thermometer attached to it which monitors temperature of the water. There is a gas burner heating water in the tank which can be turned ON when the temperature of water falls below a certain value T_1 (T_1 = 20 degrees). The burner is turned OFF when the temperature crosses a certain value T_2 (T_2 = 100 degrees). The water tank is modeled as a plant HA which is controlled by a temperature controller, which controls the ON/OFF switch used to ignite the gas burner. The evolution of water temperature is not purely continuous and depends on the mode of the system, i.e., the burner, which is either ON or OFF based on the temperature that can be below or above the upper temperature limit T_2. When the gas burner is OFF the water temperature decreases based on Equation 7, in which, I is the initial temperature of the water, K is a constant that depends on nature of the tank and t is the time.

$$x(t) = Ie^{-Kt} \tag{7}$$

When the burner is OFF and the water temperature is T_1 degrees, then it stays constant for sometime. On the other hand when the gas burner is ON then the temperature increases based on Equation 8.

$$x(t) = Ie^{Kt} + h(1 - e^{Kt}) \tag{8}$$

The controller acts on the level crossings detected for the water temperature and updates the switches ON,OFF while the plant samples the controller response. The MITL properties presented in Table I are justified on sample time of 1.27 ns which is the controller's WCRT. The trajectory of temperature evolution of water on these settings is shown in Figure 13b. On the contrary level crossings are missed when plant and controller communicate asynchronously. One such trace of a missing level crossing when temperature crosses 100 degrees is shown in Figure 13a. The synchrony hypothesis

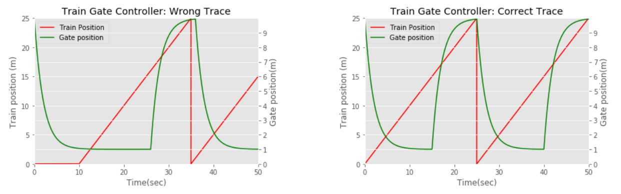


(a) Nuclear temperature Controller: temperature evolution.

(b) Train gate controller: train position evolution.

(c) Water Tank heating System:temperature evolution.

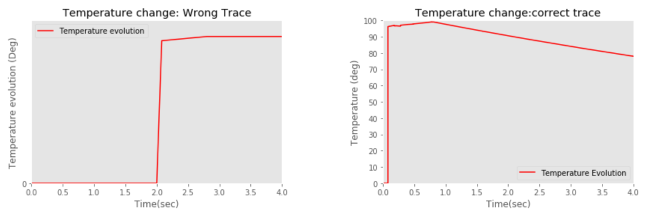
Fig. (11) Trace comparison between Simulink®and Co-sim Framework for different benchmarks using `ode45` solver.



(a) Wrong Trace of train position.

(b) Correct trace of train position.

Fig. (12) Figures showing correct and incorrect traces (because of asynch. communication) in Train Gate Controller.



(a) Wrong trace for temperature evolution of Water.

(b) Correct trace after fixing asynchronous communication.

Fig. (13) Figures showing correct and incorrect traces in water tank heating system.

is guaranteed upon setting the lower bound on the step size of the plant to a value greater than the WCRT (1.27ns) of the controller. The correlation co-efficient for water temperature trace obtained from co-sim and Simulink® models is shown in Table VII. Since the value is near unity, it confirms similar behavior between the Simulink® and hardware software co-simulation framework for this benchmark. We also plot temperature evolution trace obtained from Simulink and hardware software co-simulation model using `ode45` solver and is shown in Figure 11c. The trace is same in both the cases.

TABLE (VII) Correlation of `x` (water temperature) evolution between Simulink® and Co-Sim models under different solvers.

Co-Simulation and Simulink Solver	ode23tb	ode45	ode23	ode113	ode15s	ode23s	ode23t	daessc	odeN
Correlation co-efficient	0.999	0.9976	1	1	0.99	1	1	1	0.9989

VI. Related Work

Recent works in efficient level crossing detection is based on quantized state hybrid automata [14].This is a simulation algorithm for correctly and efficiently simulating a network of Quantized state HAs. But it has nothing to do with co-simulation based validation technique which we propose in this paper. Moreover, there are many approaches existing which perform co-simulation of Cyber-physical Systems (CPS).The most common co-simulation standard is based on the FMU [3]–[5]. Work done in [25] uses FMUs. They demonstrate that the FMI - FMU models exported from other tools which can be integrated into the setting of UPPAAL SMC [26]. They used Timed Automaton to compose clocks to achieve synchronization among different components. But this work does not do hardware model integration, and the idea of robust MITL validation is missing. Hence there is no notion of hardware and software synchronization with proper level crossing detection. Work done in [27] uses IEEE 1516-2010 High Level Architecture (HLA) standard for co-simulating distributed processes. They provide time synchronization between different simulators with different time semantics and a collection of such simulators (also called federates) which constitute joint simulation is called federation. They propose an architecture for a gateway federate similar to FMI standard to perform co-simulation. Here also the notion of robustness, hardware software synchronization and level crossings is missing. Another work [28] which deals in signal transformation from continuous to discrete domain fails to address co-simulation issues. Finally, work in [29] miss to apply efficient co-sim techniques although they use Simulink® to model a continuous system and System-C to model a discrete system.

VII. Conclusion

We have presented a novel approach of design and MITL based validation of controllers using HIL co-simulation. This work enables us to design fast, efficient and functionally correct controllers, as our methodology help validate the synchrony hypothesis, establish proper synchronization with the plant and perform MITL based validation of the controllers to ensure its proper functioning. There is no work done so far which remedy the issue of missed level crossings while doing hardware software synchronization, faster controller execution and MITL validation. We are scaling up this methodology so as to make it viable for composite system design and validation.

References

[1] S. Engell, G. Frehse, and E. Schnieder, *Modelling, analysis and design of hybrid systems*. Springer, 2003, vol. 279.

[2] H. Carlsson, B. Svensson, F. Danielsson, and B. Lennartson, "Methods for reliable simulation-based plc code verification," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 2, pp. 267–278, 2012.

[3] S. E. Saidi, A. Charif, T. Sassolas, P.-G. Le Guay, H. V. Souza, and N. Ventroux, "Fast virtual prototyping of cyber-physical systems using systemc and fmi: Adas use case," in *Proceedings of the 30th International Workshop on Rapid System Prototyping (RSP'19)*, 2019, pp. 43–49.

[4] H. Zhan, Q. Lin, S. Wang, J.-P. Talpin, X. Xu, and N. Zhan, "Unified graphical co-modelling of cyber-physical systems using aadl and simulink/stateflow," in *International Symposium on Unifying Theories of Programming*. Springer, 2019, pp. 109–129.

[5] A. Suzuki, K. Masutomi, I. Ono, H. Ishii, and T. Onoda, "Cps-sim: Co-simulation for cyber-physical systems with accurate time synchronization," *IFAC-PapersOnLine*, vol. 51, no. 23, pp. 70–75, 2018.

[6] F. Boussinot and R. de Simone, "The esterel language," *Proceedings of the IEEE*, vol. 79, no. 9, pp. 1293–1304, 1991.

[7] S. Sood, A. Malik, and P. Roop, "Robust design and validation of cyber-physical systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 18, no. 6, pp. 1–21, 2019.

[8] J. Lygeros, G. Pappas, and S. Sastry, "An introduction to hybrid system modeling, analysis, and control," *Preprints of the First Nonlinear Control Network Pedagogical School*, pp. 307–329, 1999.

[9] A. Malik and P. Roop, "A dynamic quantized state system execution framework for hybrid automata," *Nonlinear Analysis: Hybrid Systems*, vol. 36, p. 100870, 2020.

[10] N. Lynch, R. Segala, F. Vaandrager, and H. B. Weinberg, "Hybrid I/O Automata," in *International Hybrid Systems Workshop*. Springer, 1995, pp. 496–510.

[11] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications," in *Formal Approaches to Software Testing and Runtime Verification*. Springer, 2006, pp. 178–192.

[12] G. F. J. Pappas, "Robust sampling for mitl specifications," in *Formal Modeling and Analysis of Timed Systems: 5th International Conference, FORMATS 2007, Salzburg, Austria, October 3-5, 2007, Proceedings*, vol. 4763. Springer Science & Business Media, 2007, p. 147.

[13] Y. Annapureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan, "S-taliro: A tool for temporal logic falsification for hybrid systems," 2011.

[14] J. W. Ro, A. Malik, and P. Roop, "A compositional semantics of simulink/stateflow based on quantized state hybrid automata," in *Proceedings of the 17th ACM-IEEE International Conference on Formal Methods and Models for System Design*, 2019, pp. 1–11.

[15] R. Farkas, G. Bergmann, and Á. Horváth, "Adaptive step size control for hybrid ct simulation without rollback," in *Proceedings of the 13th International Modelica Conference, Regensburg, Germany, March 4–6, 2019*, no. 157. Linköping University Electronic Press, 2019.

[16] A. Dokhanchi, "From formal requirement analysis to testing and monitoring of cyber-physical systems," Ph.D. dissertation, Arizona State University, 2017.

[17] J. Lygeros, K. H. Johansson, S. Sastry, and M. Egerstedt, "On the existence of executions of hybrid automata," in *Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No. 99CH36304)*, vol. 3. IEEE, 1999, pp. 2249–2254.

[18] J. Lygeros, K. H. Johansson, S. N. Simic, J. Zhang, and S. S. Sastry, "Dynamical properties of hybrid automata," *IEEE Transactions on automatic control*, vol. 48, no. 1, pp. 2–17, 2003.

[19] B. Lincoln and A. Cervin, "Jitterbug: a tool for analysis of real-time control performance," in *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, vol. 2, 2002, pp. 1319–1324 vol.2.

[20] I. M. Filanovsky and H. Baltes, "Cmos schmitt trigger design," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 41, no. 1, pp. 46–49, 1994.

[21] I. Xilinx, "Vivado design suite user guide, using the vivado ide," 2014.

[22] R. Alur, C. Courcoubetis, T. Henzinger, P. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, "The algorithmic analysis of hybrid systems," in *11th International Conference on Analysis and Optimization of Systems Discrete Event Systems*. Springer, 1994, pp. 329–351.

[23] A. Malik, P. S. Roop, N. Allen, and T. Steger, "Emulation of cyber-physical systems using iec-61499," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 1, pp. 380–389, 2018.

[24] J.-F. Raskin, "An introduction to hybrid automata," in *Handbook of networked and embedded control systems*. Springer, 2005, pp. 491–517.

[25] P. G. Jensen, K. G. Larsen, A. Legay, and U. Nyman, "Integrating tools: Co-simulation in uppaal using fmi-fmu," in *2017 22nd International Conference on Engineering of Complex Computer Systems (ICECCS)*, 2017, pp. 11–19.

[26] A. David, K. G. Larsen, A. Legay, M. Mikučionis, and D. B. Poulsen, "Uppaal smc tutorial," *International Journal on Software Tools for Technology Transfer*, vol. 17, no. 4, pp. 397–415, 2015.

[27] T. Roth and M. Burns, "A gateway to easily integrate simulation platforms for co-simulation of cyber-physical systems," in *2018 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*, 2018, pp. 1–6.

[28] K. C. Rovers, J. Kuper, and G. J. Smit, "The problem with time in mixed continuous/discrete time modelling," *ACM SIGBED Review*, vol. 8, no. 2, pp. 27–30, 2011.

[29] F. Bouchhima, M. Briere, G. Nicolescu, M. Abid, and E. Aboulhamid, "A systemc/simulink co-simulation framework for continuous/discrete-events simulation," in *2006 IEEE International Behavioral Modeling and Simulation Workshop*. IEEE, 2006, pp. 1–6.