HES Machine: Harmonic Equivalent State Machine Modeling for Cyber-Physical Systems

Maral Amir, Tony Givargis
Donald Bren School of Information and Computer Sciences, CS Department
University of California, Irvine, California, USA
{mamir, givargis}@uci.edu

***Abstract*—Cyber-Physical Systems (CPS) are composed of computation, networking, and physical processes. Model-based design is a powerful technique to apply mathematical modeling in CPS design. A model of a physical system is the description of variations in some aspects and properties of the system such as motion, velocity, and pressure. The variations of physical quantities such as motion, velocity, and pressure as a function of time or space may be captured as a set of Ordinary Differential Equations (ODE). As such, system engineers model physical problems using mathematical equations, and then solve these equations to study the behavior of the target system. Therefore, fast executable models of physical systems are required especially for Model-based Predictive Control (MPC) algorithms or real-time Hardware-In-the-Loop (HIL) simulations. A complex physical model may comprise thousands of ODEs which pose scalability, performance and power consumption challenges. One approach to address these model complexity challenges are model-to-model transformation, and frameworks and tools that automate their implementation and development. In this paper, we present a framework to generate a Harmonic Equivalent State (HES) Machine model of the physical systems. One of the merits of the proposed state machine-based model is that the state machines can eliminate execution of compute-intensive and iterative tasks for describing the behavior of the physical systems. The model accommodates reconfigurable parameters that allow the user to have tradeoff between accuracy and execution time in CPS design. For validation purposes, we compare our model performance with state-of-the-art models in terms of execution time and accuracy. The simulation results indicate that our generated HES model executes 38% faster than ODE-based equivalent model with same level of model accuracy.**

***Index Terms*—CPS, Modeling, Simulation, Model-Based Design, Model Generation, State Machine, FFT**

I. Introduction

Model-based design in Cyber-Physical Systems (CPS) provides abstraction and modeling techniques to integrate the dynamics of the physical processes with software and communication components. As opposed to desktop computing, CPS should be dynamically reconfigurable and adapt to changes in the environment. Application-specific disturbance models may be included to predict the effect of unknown physical disturbances that perturb system behavior and incorporate these effects on the input and state variables for control system design. Complex CPS applications such as in industrial machines, land vehicles, medical equipment, spacecraft, jet engines require new computer-aided methods for modeling, simulation and offline design. These methodologies are influenced by the need for lower time to market and higher quality, reliability and safety for the CPS design.

In the literature, some research has applied Hardware-in-the-Loop (HIL) real-time simulation as a technique to improve estimation accuracy and validate the developed strategies. In real-time simulation methods, the input and output signals show the same time-dependent values as the real dynamic system. The HIL technique aims to model the real world scenarios in an abstracted environment in which the "real physical system (plant)" is replaced with the "simulated physical system". The models of the physical systems may be employed to emulate their real behavior with regards to the laws of physics and enable execution of test scenarios that would be prohibitively dangerous in a real system. Moreover, the physical model should account for the impact of measurable and unmeasurable intruding components caused by the surrounding environment (e.g. wind, noise, etc.) in order to evaluate and verify the robustness of the system under test. Therefore, dynamic model reduction in terms of accuracy may benefit HIL in emulating real-life scenarios during testing and verification. Figure 1 illustrates the application of a real-time HIL simulation to test the performance of the *Controller Unit* in a closed-loop powertrain system model [1]. The *Powertrain* block includes a group of fully assembled components, e.g. engine, transmission, drive shafts, differentials, etc., that generate power and transfer it to the road surface. The power consumption in the loop is also dependent on the speed of the vehicle which may be modeled as a time-varying variable in a *Driving Route Model* block. For this purpose, the standard driving cycles such as NEDC, ECE and UDDS [2, 3] as a set of sampled data from the environment may be applied.

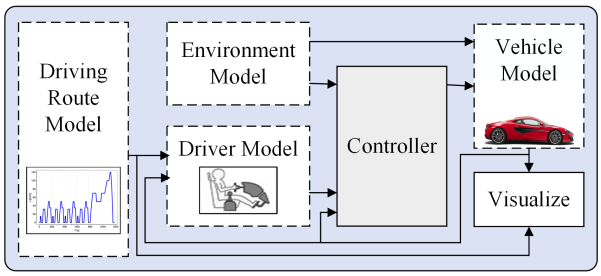


Fig. 1. Hardware-in-the-loop testing for power train system model.

Another use of model-based design in CPS application is Model Predictive Control (MPC). MPC systems are a class of

978-1-5090-3997-5/17/$31.00 ©2017 IEEE

control algorithms that estimate the behavior of the physical system under control through the use of computational models. The control inputs are optimized to drive the predicted outputs of the model towards the desired trajectory. Closed-loop performance of MPC algorithms is directly correlated with the accuracy of the physical model. Two general techniques to eliminate the steady-state offset error in the closed-loop systems are: 1) including the tracking error in the objective function of the controller, 2) augmenting the predictive model with a data-based disturbance model [4]. Most industrial MPC applications add a constant step disturbance to the output of the physical model to consider the impact of the disturbance in the closed-loop system. The work in [5] proposes a robust MPC algorithm in which a linearized model of a ship is integrated with a wind disturbance model to solve the problem of the ship's control actions in the presence of wind disturbance. This approach requires the user to design a disturbance model and integrate it in the loop with predictive model of the physical system [6]. The state-of-the-art modeling techniques for HIL simulations and MPC applications followed by our contributions are summarized in Section II.

II. Related Work

In control system study, the model of the physical system is developed to conform with dynamical system analysis and control system design requirements; that is, simplification and adaptation with respect to state of the system is required [7]. In model-based design applications such as MPC or real-time HIL simulations, the complexity of the model under control has direct influence on the global performance of the system. Specifically, different levels of complexity for the target physical system shall be provided by the user for a specific application. The work in [8] proposes an integrated library of electro-hydraulic models with different complexities. The purpose of the work is to provide the appropriate model with regards to the domain and timing requirements in design-time. However, run-time dynamic disturbance caused by the environment remains neglected.

Complex physical system models may be implemented as thousands of ODEs. The ODE description of a system requires approximations via solver methods such as Euler and Runge-Kutta, to be suitable for computations in computing devices [9]. The demand for more accurate and mathematically sound CPS solutions, cause an increase in resource utilization and energy consumption [10]. Research in model-based design techniques for CPS have introduced solutions to overcome some of the challenges induced by the complexity of ODE models. One approach is to implement the ODEs on Field-Programmable Gate Arrays (FPGA) using Lookup Tables (LUTs) to speed up simulation and enable parallel execution [11]. In general, even though the FPGA implementation of ODE models may improve the execution efficiency for real-time applications, it has implementation challenges regarding limited resources especially for complex ODE models. Hence, a better approach of modeling and solving of ODE may be required to reduce the complexity not only on FPGAs but also on general CPUs.

A state space representation of ordinary differential equations is described in [12] to obtain a discrete-time solution prior to FPGA implementation. Here, a state space meta model is introduced to model the ordinary differential equation and the respective discrete-time solution. Atlas Transformation Language (ATL) [13] is employed as the framework to implement the model transformation. Here in this paper, the experimental results are based upon simple first order differential equations and their technique may not be applicable to more complex ODE physical models. Moreover, the proposed meta models may still carry the complexity of thousands of ODEs, resulting in an implementation overhead that is prohibitive in most constrained system architectures.

The work in [14] proposes a state-based heart model generated from real specifications to be used in a closed-loop system. Implantable cardiac pacemakers monitor and repair the abnormalities in heart rhythms. HIL simulation of a pacemaker is essential to test and verify its functionality with respect to a heart model prior to real implantation. The heart model is implemented in Simulink environment and the HDL coder toolbox is used to generate Verilog code for hardware implementation. The proposed approach is application specific which requires user expertise to implement the model of the heart. Moreover, relying on the HDL coder toolbox for more complex models may require fundamental modifications in the generated Verilog code.

One solution to challenges that arise from the complexity of the ODE-based physical models is frameworks and associated tools that automate model generation and transformation for the target application [15]. Model transformations conduct automated and semi-automated mapping of one or couple of models into another alternative models in order to incorporate flexibility and compatibility in model-based design for CPS [16].

In this paper we present an automation framework to generate dynamic state machine model of a physical system augmented with a disturbance feature for Cyber-Physical Systems (CPS) applications. The model accommodates reconfigurable parameters that allow the user to have tradeoff between accuracy and execution time in CPS design. The accuracy of the physical signal may get adjusted during runtime to adopt to the system performance and robustness in the case of sudden changes that may impact the system dynamics. Our contributions in this work can be summarized as follows:

1) Designing a dynamic reconfigurable state machine model for targeted physical systems.

2) Providing tunable parameters to adjust the granularity of the generated model for adaptation to coarse-grained time critical situations or fine-grained safety critical scenarios.

3) Develop an automated framework to generate the model and its executable C code. The code may be implemented in a hardware-in-the-loop for final system testing and inte-

gration. Design objectives, model accuracy, and execution time, facilitate the evaluation and verification of the model for embedded systems implementation.

The rest of the paper is organized as follows; Section III describes the proposed automated framework that captures physical systems as a set of generated state machine equivalents. We demonstrate the performance of our framework using two benchmarks and present the results in Section IV. Finally, we state our conclusions in Section V.

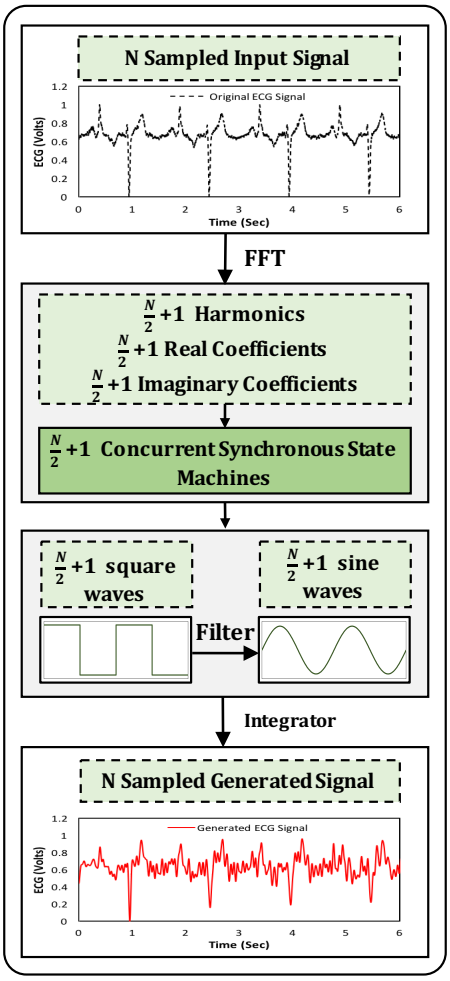


Fig. 2. High level architecture of the proposed framework illustrates the HES machine model generation process.

III. Methodology

In this work, we propose Harmonic Equivalent State (HES) Machine model generation framework that captures sampled data of a physical signal as the input in respective time windows. HES Machine generates reconfigurable state-machine model of the physical system with an intrinsic disturbance feature to adjust the overall model accuracy with respect to proposed tuning parameters for dynamic accuracy. Moreover, the execution time of the model may be adjusted in trade-off with accuracy in order to adapt to coarse-grained time critical situations or fine-grained safety critical scenarios. The main contribution of the proposed modeling framework is the inclusion of frequency domain properties in signal synthesis to adopt the reconfigurability feature to the model. Also, as opposed to ODE equivalents, the proposed framework do not perform compute-intensive and iterative tasks to solve the proposed physical model.

The high-level architecture of the proposed framework is depicted in Figure 2. The whole framework is based on the concept of signal decomposition and synthesis to generate a reconfigurable state machine model of a target physical system. The input to the framework is sampled signal of size N. Fast Fourier Transform (FFT) algorithm is employed to decompose the signal and derive the frequency information. A synthesis algorithm is presented to integrate the decomposed components of the physical signal in the form of a set of concurrent state machines. The synthesis algorithm employs $(N/2+1)$ inverse of the signal harmonic frequencies and respective FFT coefficient values, as the periods and output magnitudes of concurrent state machines respectively. Band-pass filter is used to translate the output square waves of the concurrent state machine models into sinusoidal signals. The sinusoidal output signals, one per state machine, represent the signal harmonic components. Finally, the harmonic components are integrated to generate a dynamic state machine model for the target physical system. The decomposition (analysis) and synthesis algorithms are described in details in the following sections.

A. *Decomposition*

The input to the proposed framework is N number of samples for a given physical signal in time windows of length T. The FFT algorithm is used to derive the frequency spectrum of the physical signal on each time window. Later on, this frequency domain information is employed to synthesize the signal into a state machine model representation.

1) Frequency Domain Information:

The FFT algorithm on a sampled signal of size N decomposes the signal into a series of $(N/2+1)$ sine and cosine wave components which are referred to as basis functions. The process of calculating the frequency domain information of the signal from time domain representation is called decomposition and the inverse process is signal synthesis [17]. The basis functions are a set of sine and cosine waves oscillating at signal harmonic frequencies. For a sample signal represented as array $x[]$ of size N in time domain, the FFT algorithm calculates the frequency domain signals $X[]$ as two arrays of size $(N/2+1)$. The arrays contain the coefficients (amplitudes) of the sine and cosine components as imaginary part $imX[]$ and real part $reX[]$ of $X[]$ respectively for harmonic frequencies $frX[]$. The output values of the FFT algorithm, $frX[]$, $reX[]$, and $imX[]$ are defined as input parameters for the subsequent synthesis process.

B. *Synthesis*

The synthesis function for sampled signal $x[i]$ of size N is represented in equation 1 [17]. The arrays $re\overline{X}[]$ and $im\overline{X}[]$ are the normalized coefficients of the sine and cosine waves with index k running from 0 to $N/2$ for the respective harmonic frequencies.

$$x[i] = \sum_{k=0}^{N/2} re\overline{X}[k]cos(2\pi ki/N) + \sum_{k=0}^{N/2} im\overline{X}[k]sin(2\pi ki/N) \tag{1}$$

1) HES Machine Synthesis Algorithm:

State machines can be used to break complex systems into manageable states and state transitions. Therefore, the state machine model of computation fits the synthesis function components as concurrent state machines with time-interval behavior. A global clock conducts the trigger to update the state variables and output actions. The components of the physical signal may all be generated by a five-state synchronous harmonic state machine (HES Machine).

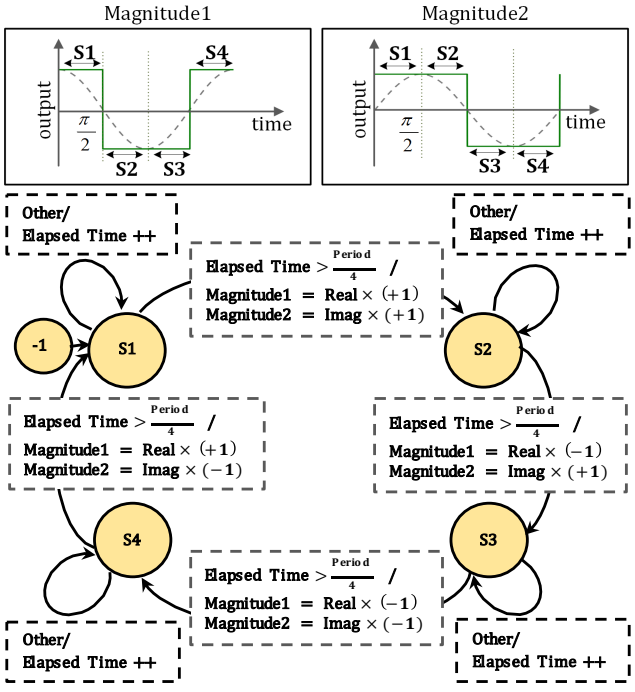


Fig. 3. 5-State synchronous state machine with the inverse of the signal harmonic frequency as the period.

The architecture of the five-state synchronous harmonic equivalent state machine is depicted in Figure 3. Each state machine is designed to represent a harmonic frequency component of the physical signal. Outputs of each state machine are two square wave signals approximating the sine and cosine components in equation 1. The square waves will later be integrated into the synthesis function. Inverse of the harmonic frequency is the period and FFT coefficient values are the magnitudes for the corresponding square waves. One period of the square wave signals is divided into four phases that are represented by the outputs of the states $S1$, $S2$, $S3$, and $S4$. The transitions between theses four states and phases are triggered at each *Period*$/4$ elapse of time. Finally, $(N/2{+}1)$ harmonic equivalent concurrent state machines are integrated to synthesize the physical signal. We have utilized this five-state synchronous state machine architecture for our model to highlight the strength of state machines in representation of physical systems. Other state machine designs may be applied for the purpose of this paper and are in queue for our future work.

Algorithm 1 illustrates the structure of the *Tick* function for the executable state machine model of computation. *HES[i]* represents a data structure that includes associated data values per harmonic state machine. Here, i is the index for harmonic state machine ranging from 0 to $(N/2{+}1)$. The parameter HES_{size} stores the number of concurrent harmonic state machines which are synthesized in a signal synthesis process and may be selected as framework parameters for design configuration. The output array values computed by the FFT algorithm, $reX[]$ and $imX[]$ and $frX[]$, are placed in the *HES* data structure to represent $HES[i].real$, $HES[i].imag$ and $HES[i].period$ respectively. The variable $HES[i].elapsedTime$ is tracked on each call of the *Tick* function. When $(HES[i].elapsedTime \geq HES[i].period/4)$ condition evaluates to true, a state transition occurs and an output action is determined with respect to the current state. N samples of signals are fed into the HES machine model generator in intervening time windows of T. Each execution of the *Tick* function updates the $HES[i].elapsedTime$ variable by adding T_{res} values. The values for the new time window are evaluated when the $HES[i].elapsedTime$ variable surpasses the value T and resets to zero.

The generated square waves are to be translated into sinusoidal equivalents to represent the sine components of the original physical signal. A band-pass filter is applied to attenuate the unwanted square wave frequencies. In future work, we plan to apply further measurements to compensate for filter error. $(N/2{+}1)$ sinusoidal signals are integrated to synthesize the decomposed signal according to Equation 1. The tool generates an executable C code in state machine representation for the physical signal to be implemented on a target platform.

2) HES Machine Tuning Parameters:

Two tuning parameters HES_{size} and T_{res} are proposed to adjust the HES Machine model in order to meet system requirements (e.g., accuracy and timing).

1) **Machine Size (HES_{size})** is the number of harmonic concurrent state machines to be integrated during the synthesis process ranging from 1 to $(N/2{+}1)$. The model accuracy may be adjusted with respect to this parameter by inclusion/elimination of certain harmonic frequencies.
2) **Time Resolution (T_{res})** parameter indicates the smallest time unit in the proposed framework by which the generated state machine will be executed. The proposed framework tracks the value of T_{res} as an actual wall-clock

Algorithm 1: Global Tick Function

Input: index of the state machine i

```
global variable HES
global variable magnitude1, magnitude2
const timeResolution
switch HES[i].state do
    case −1
        HES[i].state = S1
    case S1
        if HES[i].elapsedTime ≥ HES[i].period/4 then
            HES[i].state = S2
            HES[i].elapsedTime = 0
        else
            HES[i].state = S1
    case S2
        if HES[i].elapsedTime ≥ HES[i].period/4 then
            HES[i].state = S3
            HES[i].elapsedTime = 0
        else
            HES[i].state = S2
    case S3
        if HES[i].elapsedTime ≥ HES[i].period/4 then
            HES[i].state = S4
            HES[i].elapsedTime = 0
        else
            HES[i].state = S3
    case S4
        if HES[i].elapsedTime ≥ HES[i].period/4 then
            HES[i].state = S1
            HES[i].elapsedTime = 0
        else
            HES[i].state = S4
    otherwise
        HES[i].state = −1
switch HES[i].state do
    case S1
        magnitude1[i][HES[i].N1] = HES[i].real × 1.0
        magnitude2[i][HES[i].N1] = HES[i].imag × 1.0
    case S2
        magnitude1[i][HES[i].N1] = HES[i].real × −1.0
        magnitude2[i][HES[i].N1] = HES[i].imag × 1.0
    case S3
        magnitude1[i][HES[i].N1] = HES[i].real × −1.0
        magnitude2[i][HES[i].N1] = HES[i].imag × −1.0
    case S4
        magnitude1[i][HES[i].N1] = HES[i].real × 1.0
        magnitude2[i][HES[i].N1] = HES[i].imag × −1.0
    otherwise
        magnitude1[i][HES[i].N1] = HES[i].real × 1.0
        magnitude2[i][HES[i].N1] = HES[i].imag × 1.0
HES[i].elapsedTime+ = timeResolution
HES[i].N1 + +
return
```

(real) time. The parameter T_{res} specifies the timer values for the periodic programmable interval timers to trigger the interrupt service routine (ISR).

IV. Experimental Results

A. Implementation and Setup

Simulation experiments are conducted using data for real physical signals and the global clock of the state machine model is updated by interrupt handlers of the operating system. The framework is implemented using C/C++ programming language in order to enable it to be highly portable for compilation and execution. The process from data acquisition to model generation is automated and reconfigurable with respect to model parameters. Our specific experiments were performed on a PC with a quad-core Intel Core i5 and 8 GB of DDR3 RAM. The performance of the proposed model generation framework is evaluated using two examples of ECG signal and NEDC signal. It needs to be noted that one of the merits of our framework is its applicability to any example and application of physical signals.

- **ECG Signal:** The electrocardiogram (ECG) digitized signal is provided by PhysioBank [18] as the reference signal. The signal consists of 7200 double-sized sample values recorded with 720 Hz sampling frequency and 12-bit resolution in T=10-seconds window of time. The model is reconfigured at run-time to serve for HIL testing of implantable medical devices such as pacemaker, smart ECG monitor, etc. [19].
- **NEDC Signal:** The New European Driving Cycle (NEDC) is selected from driving cycle standards (ECE, UDDS, etc.) that are typically employed in model-based vehicle design applications [2]. The NEDC signal contains vehicle velocity data that is captured from a Simulink block [20]. The signal is 8192 double-sized values sampled at frequency of 10 Hz in T=819.2-seconds window of time. The generated dynamic HES Machine model may be reconfigured at run-time to emulate the system behavior in presence of environment disturbance or mis-prediction of trajectory.

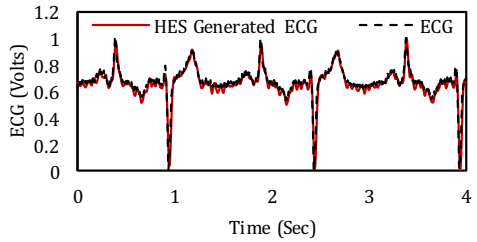
The generated signal models of the proposed framework for ECG and NEDC examples are compared in Figures 4(a) and 4(b) with their original signals. The results justify the validity of our proposed model generation framework for signal synthesis and state machine model generation with average of 0.1% error in model accuracy.

B. Analysis and Verification

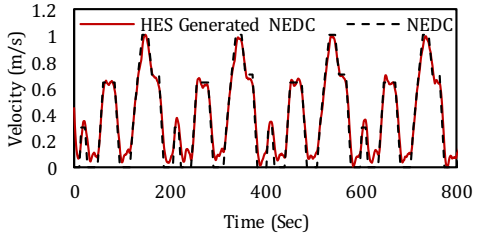
1) Performance Metrics:

Two performance metrics of execution time and precision are considered for comparing the performance of our HES Machine model with state-of-the-art models.

- **Execution Time:** is the time required by the computer to perform a given set of computations.
- **Root Mean Squared Error (RMSE):** is the quality factor to measure the error between the values evaluated by the



(a) Comparing the original ECG signal with the HES Machine generated output signal.

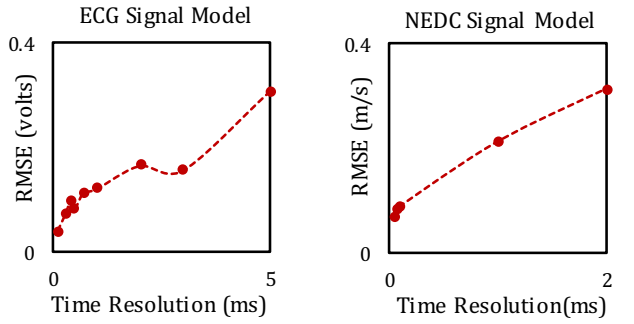


(b) Comparing the original NEDC signal with the HES Machine generated output signal.

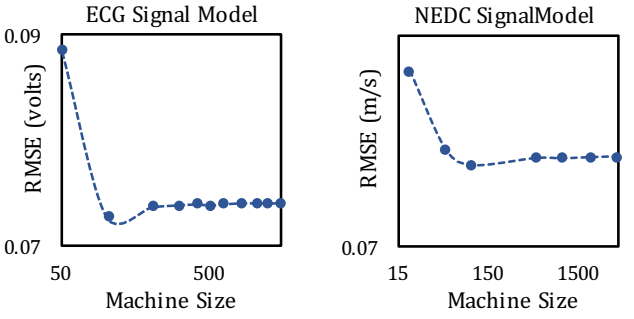
Fig. 4. HES Machine's generated signal of ECG and NEDC.

model and the corresponding expected values for N number of samples.

$$RMSE = \sqrt{\frac{(\sum(Expected - Evaluated)^2)}{N}} \quad (2)$$



(a) "Time Resolution" analysis considering the RMSE metric, for ECG and NEDC.



(b) "Machine Size" analysis considering the RMSE metric, for ECG and NEDC.

Fig. 5. Analyses for different parameter configurations.

*2) **Parameter Analysis**:*

The performance of the proposed state machine model generation framework is evaluated under variations of two parameters by which the model may be configured:

1) **Time Resolution (T_{res}):** Figure 5(a) illustrates the change in model accuracy for NEDC and ECG examples with respect to variations in T_{res} parameter values. The results shows improvement in model accuracy for smaller values of T_{res} in case of need for finer physical model.

2) **Machine Size (HES_{size}):** the variations in model accuracy with respect to different values of parameter HES_{size} is illustrated in Figure 5(b). Here, The accuracy of the model improves for larger values of HES_{size}. In our experiments, the harmonic frequencies and their corresponding state machines are sorted in ascending order to select HES_{size} number of state machines for integration. Other selection algorithms may be applied in accordance with target application which may further improve the precision of the generated model.

*3) **Comparison to State-of-the-Art**:*

We evaluated the performance of the proposed framework and generated model in comparison with an ODE-based ECG signal generator, ECGSYN [21]. The model in [21] emulates the quasi-periodic waveform of the ECG signal by tracing around a limit cycle in x-y plane. The ECG signal is generated by using a series of exponentials formulated to follow PQRST-waveform in the z-direction. (P, Q, R, S, T) represent the peaks in ECG signal for one complete heartbeat. The model of motion dynamics is defined as a set of following differential equations

$$\dot{x} = \alpha x - \omega y \quad (3)$$

$$\dot{y} = \alpha x + \omega y \quad (4)$$

$$\dot{z} = -\sum_{i \in P,Q,R,S,T} a_i \Delta\theta_i \exp(-\Delta{\theta_i}^2/2{b_i}^2) - (z - z_0) \quad (5)$$

where ω is the angular velocity, α equals to $(1 - \sqrt{x^2 + y^2})$ and $\Delta\theta$ equals to $(\theta - \theta_i)\%2\pi$. Also, $\theta = \arctan(y, x)$ and a_i and b_i are model coefficients. The fourth-order Runge-Kutta method [22] is applied to solve the ordinary differential equation model. The performance of HES and ECGSYN models in terms of execution time and accuracy is evaluated for time interval of T=16 seconds. The functions involved in execution of the HES model are FFT function, state machine, band-pass filter, and integration for signal synthesis. The experimental results in Figures 6 and 7 illustrate the behavior of the generated HES model in terms of execution time and accuracy based on variations in framework parameters: T_{res} and HES_{size}. The execution time overhead of the generated HES model is attributed to three main functions: state machine, filter and integrator. The figure shows variations in execution time for different model parameter configurations. Table I shows the execution time values for state-of-the-art

model ECGSYN with respect to corresponding accuracy of the model. The frequency f_s represents the step size for the ODE solver and is considered as the model parameter to adjust the accuracy accordingly.

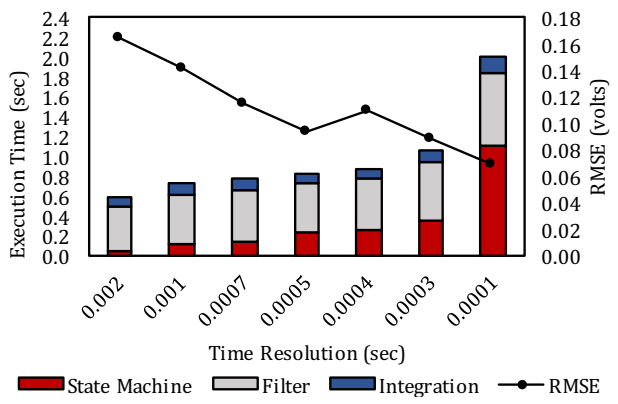


Fig. 6. Time complexity and error analysis of the proposed HES model with respect to "Time Resolution" parameter.

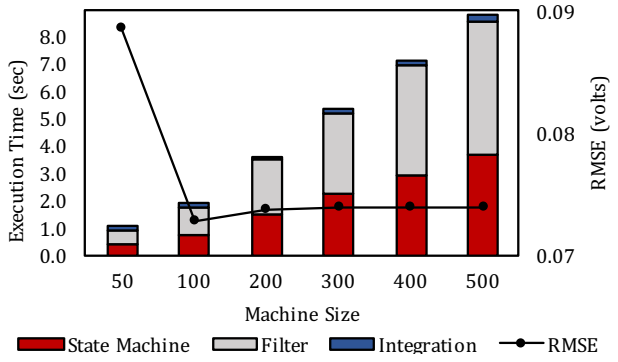


Fig. 7. Time complexity and error analysis of the proposed HES model with respect to "Machine Size" parameter.

The results in Figure 6 shows that the execution time increases as the model accuracy improves with smaller values for T_{res} parameter. Moreover, Figure 7 show that smaller values of HES_{size} reduce the execution time since less number of state machines are to be generated and integrated. As shown in the picture, for HES_{size} larger than a certain value, the change in model accuracy will be marginally negligible. We can use this property to improve the execution time for coarse-grained time critical situations. The generated ECG signal with f_s=720 is considered as the reference signal for HES and ECGSYN models to be evaluated; that is, this reference signal is given as the input to our proposed framework to generate the equivalent of reconfigurable HES model. For fairness of comparison, the execution time of the proposed HES model is compared with ECGSYN for the same range of accuracy as shown in Figure 8. The results for HES model is derived for HES_{size}=50 which includes 50 harmonics in synthesis of the generated signal. The experimental results show that for same level of model accuracy, the HES model may be executed 38% faster than ECGSYN model. In future work, we plan to take measurements to replace the filter for further improvement in model performance.

TABLE I
COMPLEXITY ANALYSIS OF THE ECGSYN STATE-OF-THE-ART MODEL WITH RESPECT TO "FREQUENCY" PARAMETER.

Model	f_s	Execution Time (sec)	RMSE (volts)
ECGSYN	720	0.988	0.000
	718	0.966	0.178
	715	0.960	0.176
	710	0.956	0.164
	705	0.949	0.151
	700	0.949	0.170
	690	0.949	0.179
	650	0.911	0.244
	600	0.919	0.213

The improvement in execution time is due to the novel approach to solve the HES model in contrast to ODE models (ECGSYN). The proposed state machine-based model do not execute compute-intensive and iterative tasks to describe the behavior of a physical system. Moreover, concurrent operation of the state machines are perfectly suitable for intrinsic parallel characteristics of physical systems. In other words, it allows multiple sub-state machines to react to a set of events at the same time. In general, the time complexity of a solver to solve N samples of ordinary differential equations may grow with respect to $c'N$, where c' is a constant factor defined by the type of the discretization algorithm, numerical ODE solver, number and order of the ordinary differential equations in the physical model. On the other hand, the time complexity of the proposed HES model grows with the term cN, where c is determined by the model parameters HES_{size} and T_{res}. Therefore, our HES Machine model is suitable for systems that are more tolerable against model error in tradeoff for reduction in execution time. Here, the ECGSYN includes three simple first-order ordinary differential equations which results in small value of c'. However, we expect that execution of the HES model equivalent to more complex ODE models with larger values of c' will present even smaller values for execution time.

V. CONCLUSION

In this paper, we presented an automated model generation framework for physical systems in CPSs. The proposed method utilizes frequency information properties to generate a dynamic state machine model of the physical system. Two tuning parameters are provided to adjust the granularity of the generated model for adaptation to coarse-grained time critical situations or fine-grained safety critical scenarios. Simulation is conducted to evaluate performance of the framework and model using two real physical signals of ECG and NEDC. Moreover, the generated state machine model is compared with ODE-based state-of-the-art equivalent model in terms of

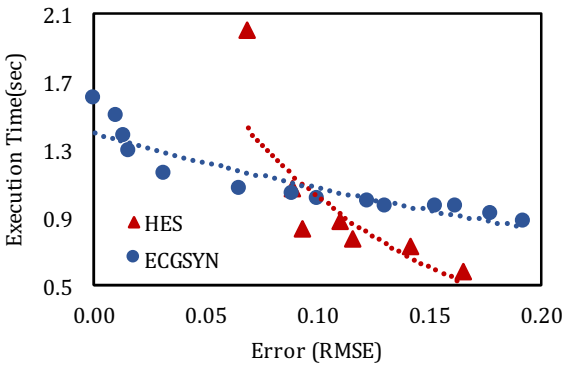


Fig. 8. Comparison of execution time versus error for the proposed model HES and state-of-the-art ECGSYN.

accuracy and execution time. The simulation results indicate that our generated model surpasses the state-of-the-art model by 38% in execution time for same level of model accuracy. The proposed dynamic state machine system may be an excellent replacement for complex ODE solvers when used for testing or embedding CPSs.

ACKNOWLEDGMENT

This work was supported in part by the National Science Foundation under NSF grant number 1563652.

REFERENCES

[1] Korosh Vatanparvar and Mohammad Abdullah Al Faruque. Eco-friendly automotive climate control and navigation system for electric vehicles. In *Cyber-Physical Systems (ICCPS), 2016 ACM/IEEE 7th International Conference on*, pages 1–10. IEEE, 2016.

[2] Korosh Vatanparvar, Al Faruque, and Mohammad Abdullah. Battery lifetime-aware automotive climate control for electric vehicles. In *Proceedings of the 52nd Annual Design Automation Conference*, page 37. ACM, 2015.

[3] Korosh Vatanparvar, Al Faruque, and Mohammad Abdullah. Otem: optimized thermal and energy management for hybrid electrical energy storage in electric vehicles. In *Proceedings of the 2016 Conference on Design, Automation & Test in Europe*, pages 19–24. EDA Consortium, 2016.

[4] Kenneth R Muske and Thomas A Badgwell. Disturbance modeling for offset-free linear model predictive control. *Journal of Process Control*, 12(5):617–632, 2002.

[5] Anna Miller. Model predictive control of the ships motion in presence of wind disturbances. *Zeszyty Naukowe/Akademia Morska w Szczecinie*, 2014.

[6] Fitri Yakub, Aminudin Abu, Shamsul Sarip, and Yasuchika Mori. Study of model predictive control for path-following autonomous ground vehicle control under crosswind effect. *Journal of Control Science and Engineering*, 2016, 2016.

[7] Alessandro Sassone, Donghwa Shin, Alberto Bocca, Alberto Macii, Enrico Macii, and Massimo Poncino. Modeling of the charging behavior of li-ion batteries based on manufacturer's data. In *Proceedings of the 24th edition of the great lakes symposium on VLSI*, pages 39–44. ACM, 2014.

[8] Jorge A Ferreira, João E De Oliveira, and Vítor A Costa. Modeling of hydraulic systems for hardware-in-the-loop simulation: A methodology proposal. In *Proc. Int. Mechanical Eng. Congress & Exposition, Nashville, USA*, 1999.

[9] Lawrence F Shampine, Ian Gladwell, and Skip Thompson. *Solving ODEs with matlab*. Cambridge University Press, 2003.

[10] Edward A Lee. Cyber physical systems: Design challenges. In *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, pages 363–369. IEEE, 2008.

[11] Chen Huang, Frank Vahid, and Tony Givargis. A custom fpga processor for physical model ordinary differential equation solving. *IEEE Embedded Systems Letters*, 3(4):113–116, 2011.

[12] Alonso Gutiérrez Galeano and Fernando Jiménez Vargas. Automatic code generation for hardware-in-the-loop simulation of differential equations using model-driven engineering.

[13] Frédéric Jouault, Freddy Allilaire, Jean Bézivin, and Ivan Kurtev. Atl: A model transformation tool. *Science of computer programming*, 72(1):31–39, 2008.

[14] Zhihao Jiang, Sriram Radhakrishnan, Varun Sampath, Shilpa Sarode, and Rahul Mangharam. Heart-on-a-chip: a closed-loop testing platform for implantable pacemakers. 2014.

[15] Claudius Ptolemaeus, editor. *System Design, Modeling, and Simulation using Ptolemy II*. Ptolemy.org, 2014.

[16] Maral Amir and Tony Givargis. Hybrid State Machine Model for Fast Model Predictive Control: Application to Path Tracking. *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2017.

[17] Steven W Smith et al. The scientist and engineer's guide to digital signal processing. 1997.

[18] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. Ch. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000 (June 13). Circulation Electronic Pages: http://circ.ahajournals.org/content/101/23/e215.full PMID:1085218; doi: 10.1161/01.CIR.101.23.e215.

[19] Benoît Barbot, Marta Kwiatkowska, Alexandru Mereacre, and Nicola Paoletti. Estimation and verification of heart models for personalised medical and wearable devices. In *International Conference on Computational Methods in Systems Biology*, pages 3–7. Springer, 2015.

[20] Daniel Auger. *Driving-Cycle*. https://www.mathworks.com/matlabcentral/fileexchange/46777-driving-cycle--simulink-block-.

[21] Patrick E McSharry, Gari D Clifford, Lionel Tarassenko, and Leonard A Smith. A dynamical model for generating synthetic electrocardiogram signals. *IEEE Transactions on Biomedical Engineering*, 50(3):289–294, 2003.

[22] Saul A Teukolsky, Brian P Flannery, WH Press, and WT Vetterling. Numerical recipes in c. *SMR*, 693:1, 1992.