AIP Advances

System-level co-simulation for embedded systems

Mossaad Ben Ayed (iD), Ayman Massaoudi (iD), Shaya A. Alshaya (iD), and Mohamed Abid

View Online   Export Citation   CrossMark

AVS Quantum Science

Co-Published by

AIP Publishing   AVS

RECEIVE THE LATEST UPDATES

System-level co-simulation for embedded systems

View Online   Export Citation   CrossMark

Mossaad Ben Ayed,[1,2,a)] (ID) Ayman Massaoudi,[3,4,b)] (ID) Shaya A. Alshaya,[1,c)] (ID) and Mohamed Abid[5,d)]

AFFILIATIONS

[1] Department of Computer Science, College of Sciences and Humanities at alGhat, Majmaah University,
 Majmaah 11952, Saudi Arabia
[2] Department of Computer Science, Computer and Embedded System Laboratory, Sfax University, Sfax 3029, Tunisia
[3] Department of Computer Science, Jouf University, Al Jouf 74331, Saudi Arabia
[4] Department of Computer Science, MEDIATRON Lab., Sup'Com, Carthage University, Tunis 1054, Tunisia
[5] Department of Computer Science, Sfax University, Sfax 3029, Tunisia

[a)] Author to whom correspondence should be addressed: mossaad_benayed@yahoo.fr and mm.ayed@mu.edu.sa.
Tel.: +216-50606360
[b)] Electronic addresses: ahmassaaoudi@ju.edu.sa and aymen.massaoudi@supcom.tn
[c)] shaya@mu.edu.sa
[d)] mohamed.abid@enis.rnu.tn

ABSTRACT

The technological revolution affects the growth of systems in terms of functionality and complexity. Industries of embedded systems become increasingly an area of interest for researchers to develop Computer-Aided Design (CAD) environments to support at the same time the complexity in terms of different components and functionalities in terms of application programming interface and libraries. Mainly, CAD tools based on multi-level co-simulation are challenged by the time-to-market constraint. As known, the behavior description at a higher level provides a speedy simulation, but it suffers from bad accuracy. Therefore, describing a customized model for a system behavior with sufficient functional details at an earlier stage of modeling is a great challenge to researchers. As an attempt to overcome the last challenge, this paper presents a co-simulation model based on a synchronization methodology to ensure the verification between the conceptual level and the functional level. The proposed system-level co-simulation model is implemented to interfaces to provide the switch context in the case of the Arena and the Simulink/Matlab environments. The evaluation was performed by using two case studies with different domains to prove the effectiveness of the proposed system-level co-simulation interfaces.

I. INTRODUCTION

Simulation modeling is widely used by developers in several fields such as embedded systems,[1] healthcare,[2] security systems,[3] and traffic systems.[4] The main goal of the simulation is to provide an abstraction of the real world with the most flexibility needed in the modeling phase.[5,6]

Indeed, there is an increasing need for customized systems. Customization has many advantages, e.g., software has given products improved capabilities. For example, controlling a conveyor belt can be programmed in diverse ways, such as starting, stopping at different intervals, and advancing a certain distance. In addition, the software can be used to gain design flexibility. The software can be designed with several parameters, which can be assigned different values and thereby be used for customization.

A queuing system[7] can be defined as a system designed at the conceptual level and it is based on processes and data flow. The queuing theory considered essentially the arrival process time, the service time, and the average waiting time or queue size.[8] Its design plays a crucial role in the challenges, especially in supporting

the complexity management and the integration of engineering domains to attain the desired results. A model of an embedded system can be represented as a queuing system when designed at the conceptual level. In this case, the functionality was not the purpose, but the time advance of each process.

The design and simulation of the queuing system became a real challenge for two reasons. The first, due to the hard competition between constructors, the reduction in the time-to-market became more important. The second factor was the choice of the adequate language/tool. This choice was tied to (1) models of computation used to model abstractions of systems (discrete, event discrete, continuous, heterogeneous, etc.), (2) the application field (mechanical, embedded, electronic, chemical, etc.), and (3) the design of the abstraction level.

Queuing system simulation has several limits: (1) the simulation modeling did not support the continuous model,[9] (2) the description in higher level tolerated many details, and (3) the service time of a process was assumed as a probabilistic function, which was not the real case. Therefore, the simulation could not perform a precise result and the error could be big for long execution time.

At certain stages of modeling, it is essential to check the functional conformity of the system under design. The modeling-simulation approach,[10] called the V Model, represents the life cycle process. This modeling level approach is verified by simulation. Figure 1 describes the V model in terms of system design and system verification. The system design is moved from system requirements to a component implementation. The verification system is moved from the acceptance test to the component test. Each level corresponds to a level of verification. System design steps begin from high-level descriptions (functional system design). A verification step is applied to this step. If the simulation results are accepted, the designer should be moved down to describe the system at a lower-level design, also some rectifications must be done for system requirements. This process is repeated for every system design level. The V model approach decreases the time-to-market in comparison with traditional approach modeling based on verification only at the implementation stage.

Nevertheless, the embedded system required much time to reach the implementation stage. In many cases, errors related to lack or insufficiency of resources are detected in the component implementation step as discussed in Refs. 11–14. Predicting these kinds of problems at a high level of simulation represents a great challenge.

Most designers begin their design from the functional level. In this stage, many details need to be known to perform the whole model of the System Under Design (SUD). However, the model in itself could not support the simulation of the system at the conceptual level based on the queuing approach. Related works in Refs. 15, 13, and 11 prove that some bugs and hardware conflicts are detected when the designer moves from the functional level to the physical level due to insufficient memory size or lack of memory.

In the light of this brief review, this study proposes to replace the functional system design by the aggregation of conceptual and functional levels based on co-simulation interfaces (as mentioned in red color in Fig. 1).

The paper will be organized as follows:

Section II defines general concepts related to the abstraction levels. Section III describes the literature review. Section IV presents the synchronization model for the proposed co-simulation environment. Section V introduces the case study used to evaluate the
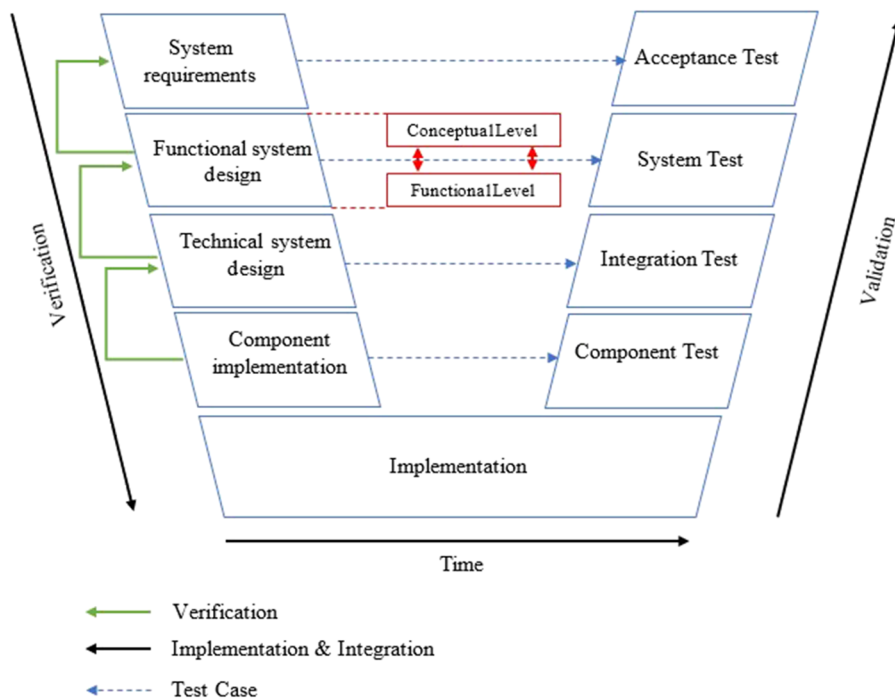

FIG. 1. V model approach.

attempted proposition. Section VI exhibits the experimental results and accuracy analysis. Finally, Sec. VII concludes the paper.

II. GENERAL CONCEPT: ABSTRACTION LEVELS

System's simulation is defined according to a model. This model represents the behavior of the system in terms of specification. Many classifications and criteria are discussed in the literature to provide levels of modeling.

A. Gajski's approach

In 1991, Gajski et al.[16] proposed a modeling approach based on the complexity of the system in terms of detail. This modeling approach describes the levels of abstraction that a circuit can be presented during its development in terms of representation. These levels represent the description of a circuit according to the level of detail moving from a high-level description with a minimum of details to the low-level description with more details. Gajski et al. defined the five levels of abstraction based on the system's specification. These levels are

- **System level**: this level presents the system at the specification level and gives the essential components of the system (processor, memory, I/O, etc.).
- **Algorithmic level**: This level describes the system in an algorithmic form based on Intellectual Properties (IPs). This description uses high-level languages.
- **Register Transfer Level (RTL)**: This level describes the system as a registry level description. This description uses the Hardware Description Language (HDL).
- **Logic level**: this level describes the system by logic gates. This description uses the hardware language at the component level.
- **Circuit level**: this level describes the system by differential equations. This representation is based on interconnections between transistors.

B. Cassandras approach

Cassandras et al.[17] propose three levels of abstraction for modeling and simulation: non-temporal language, temporal language, and stochastic temporal language.

- The non-temporal language model represents the set of all possible orders of events that could occur in the given system. The model thus ignores the temporal information that corresponds to the moments of the occurrences of events.
- The time-domain model is characterized by distribution functions associated with events. Stochastic time modeling contains information about events and the exact/estimated times of the event.
- The stochastic time-language model is the most detailed model. It provides aside from time-domain model, statistical information about successive occurrences of events.

C. Bombieri approach

Bombieri et al.[18] present another classification of the abstraction level. The main criteria defining the abstraction level are temporal granularity, state space granularity, interconnection model, and data aggregation. Temporal granularity represents an important factor in a heterogeneous environment. This can be a discrete-event or continuous model. The interconnection of the model describes communication and synchronization between components and transaction flowcharts. The state space granularity represents the system's behavior according to symbolic variables defined by the correspondent differential equations. Finally, data aggregation indicates whether a component is modeled by considering the minimum (black box) or the maximum (clear box) number of state space variables. Given these factors, it is possible to identify five main abstraction levels:

- Transactional level: the simulation is strictly event-driven and the inter-component communication is done via transactions. The state of the system is modeled with variables.
- Functional level: the simulation is event based, but the communication is based on an interconnection flow diagram.
- Structural level has two main approaches depending on the temporal granularity. The continuous time evolution is modeled with differential equations. The discrete time can adopt both an event based diagram or a synchronization flowchart, and the finite set variables are adopted.
- Component level: the simulation can be both continuous or discrete. In this level, all variables must be explicitly modeled, whereas at the structural level only the necessary variables are explicitly modeled.
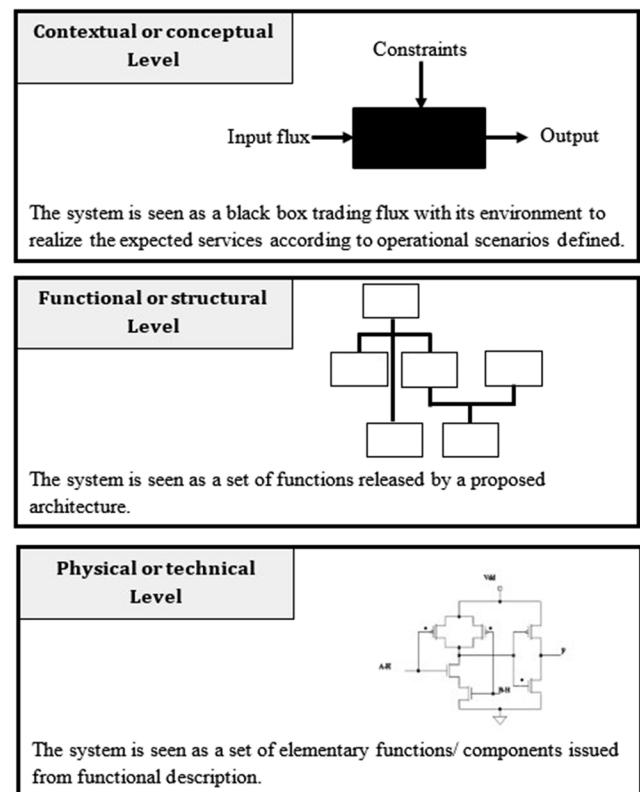

FIG. 2. Principle levels for system description.

- Physical level adopts the continuous synchronization with the preservation of the interconnect style. The state space is described with continuous time based on differential equations and all variables are modeled in a clear box approach.

D. The proposed classification of the abstraction level

The proposed classification is based on the depth of details related to the process: few details, medium details, deep details. As mentioned in Fig. 2, a system could be modeled at three principal levels.

The conceptual level, the process has the minimum details, only time details. In the functional level, the process defines the functionality using either the algorithm, structure, or register. Physical level merges logic and circuit levels defined by Gajski.

- The conceptual/contextual level is designed to describe the behavior of the system as a set of interconnected processes. Each one is defined essentially by a probabilistic function, which corresponds to the estimated service time. Therefore, the simulation could perform an unclear result and the error could be large for a long simulation time.
- The functional level involves more details about the behavior of the system. It aims to define the nature of the function used to perform the desired process.
- The physical level describes the system at the lowest level of abstraction using elementary components (logic gates or transistors).

III. LITERATURE REVIEW

As will be shown below, the choice of the description level presents a great challenge for designers. According to the proposed classification of the abstraction level, it is important to define the system level as the set of the conceptual and the function level. So far, most embedded systems were described at a unique level, at a particular functional level.

At an earlier stage of description, designers need an appropriate tool to describe the whole system in the conceptual layer. There are several description languages/tools in the literature as: Product Lifecycle Management (PLM),[19] SolidWorks,[20] Arena,[21] SysML,[22] PRISMSYS,[23] PLCOpenXML,[24] etc. Some of them ensure the description only on the conceptual level as Arena and SysML, but other tools try to describe briefly the controller part in the functional level as PLM and PRISMSYS.

The design at a higher level ensures the verification and the feasibility of the whole system at an earlier stage of development. It could estimate: (1) the transfer of data flow between processes, (2) queuing information related to each process, and (3) expected data.

Nevertheless, the description at the conceptual level based on probability estimations led to approximate results. The estimated service time constitutes the source of the problems met at this level. Doing on the real-time of the processes avoid misunderstanding of the simulation results.

This approximation on service time depends directly on the resource (puissance, frequency) and the spherical wave Software (SW) application used. In fact, the conceptual level suffers from the lack of simulation accuracy of the whole system.

Most languages/tools in the conceptual level do not support the functional level and others offer a limited functional block that is insufficient to rely on simulation results.[25,26]

To overcome these shortcomings, many co-simulation frameworks are proposed in the literature to design the system at the conceptual and functional levels simultaneously.

In Ref. 27, Liang *et al.*, described a queuing system using Arena tool for the discrete-event model and Matlab for the continuous model. In this case, the event discrete model described by Arena integrated the continuous model described by Matlab. The context switch is defined by a simple activation monitored through Arena by Visual Basic for Applications (VBA) and ActiveX technologies. The activation is ensured by an easier transaction between the two environments. The used co-simulation technique based on VBA requests a lot of time in the simulation due to the large overhead caused by VBA and Excel. Unfortunately, this work did not support the multilevel case of a discrete-event system that requires more complicated co-simulation interfaces in terms of exchanged data and time stamp. Our proposition attempts to support this feature.

In Ref. 28, Sanz *et al.* combined the Arena/Siman and Modelica libraries to model a heterogeneous system based on the continuous and the discrete-event model. Since Arena/Siman tool supports only the discrete-event model, the authors propose to integrate Modelica libraries to provide continuous packages as proposed by Liang *et al.*[27] but replacing Matlab by Modelica for the continuous model. The authors create an ExtrenalProcess module based on an ExternalAssign block to ensure the context switch between environments with service time feedback from the continuous model. Our attempt manages a multi-level event-discrete model, which requires many parameters to be exchanged between the levels. In contrast, the Sanz's model only needs the activation of the context switch between environments and returns the service time.

Süß *et al.*[29] studied the behavior simulation of a manufacturing system associated with the mechatronic components at the conceptual level. The PLCOpenXML framework describes the event-discrete model and the continuous model is designed using Modelica/Matlab. This work also did not support the multi-level of an event-discrete system in the case of an embedded system.

Barbieri *et al.*[30] combined the SysML environment based on the process and Matlab environment based on Hardware in the Loop (HIL) Simulation for the mechatronics system. Authors attempt to integrate Matlab into SysML diagrams at the conceptual level. The presented case study, robot system, is described by three main models: (1) discrete-event for the simulation of lines and plants using the FlexSim[31] tool; (2) virtual commissioning for the simulation of the physical status using SysML and IndustrialPhysics[32] tools, and (3) dynamics model for the simulation of the control unit using the hardware in the loop techniques (Matlab) performed by the PLC controller. The studied co-simulation ensures the verification of a heterogeneous system without taking consideration of the data exchanged status between processes. This problem has not appeared because the case study is composed only of three processes, and the continuous model, which requests much simulation time manages the system. This research work follows the same previous approaches[24–26] based on the activation of the context switch between models. In addition, the use of many tools results in a speed down of the simulation caused by the accumulated overhead delay between tools.

Chabibi et al.[33] attempted to generate the Simulink model from a conceptual level described by SysML based on model transformations and metamodeling. This method did not support the simulation of a multi-level model like in our case. Besides, a coffee machine with a little and easiest functionality, as a case study, is performed. This idea seems to be not feasible for complex systems such as an embedded system.

Dehghanimohammadabadi et al.[9] integrated Matlab in the SIMIO[34] environment. Matlab is used as a computational tool for the continuous model and the SIMIO describes the discrete-event model. The communication layer is performed according to a database to store the history of the context switch. However, this method causes a decrease in the simulation time. The authors attempt to add a new dimension to the queuing simulation by integrating optimization. It is ensured by the invoke of Matlab to perform an intelligent algorithm for the manufacturing system.

Lora et al.[35] described a meet-in-the-middle approach to represent a heterogeneous system at the system-level description. This approach benefits from the advantages related to the design flows of both top-down and bottom-up approaches. The authors attempt to integrate heterogeneous components into a homogeneous virtual platform. Through their findings, developers can model the system using components described at different levels of abstractions. Therefore, the authors perform mechanisms based on translation, abstraction, and integration of existing tools to ensure synchronization between heterogeneous components. Unfortunately, this great study is limited at the functional level (the description is done by a custom C/C++ language) and did not focus on the conceptual level.

Edward et al.[36] provided CyPhySim tool as an open-source environment for the verification of cyber-physical systems. It is based on the Ptolemy II simulator to ensure the heterogeneity of the model. The CyPhySim belongs to the commune approach in which the simulator supports the verification of both discrete and continuous models. It provides the modeling of a system composed of differential equations, discrete-event models, hybrid models, functional mockup interface models, algebraic loop solvers, and discrete-time models. This tool provides a flexible and accurate environment for developers, but it is limited also in modeling at the functional level.

Ben Ayed et al.[37] presented the CODIS+ environment as an extension of the CODIS tool. The CODIS+ provides the co-simulation of both discrete-event and continuous models. The synchronization between the SystemC for the discrete event, the Matlab/Simulink for the continuous model, and a hardware accelerator is described. Like the previous work, the CODIS+ supports the verification only at the functional level.

Törngren et al.[38] outlined challenges met in Computer-Aided Engineering (CAE) domains, especially in the modeling and verification of a complex system. The authors propose to integrate heterogeneous physical, cyber, cyber-physical system components, aspects, and systems. This integration is ensured by interfaces and interrelations. The main goal of this study is to provide an accurate CAE environment that permits to model according to different components (mechanical, electrical, chemical, informational, etc.). This attempt provides the modeling and simulation at the functional level based on the synchronization of different simulators.

Wehmeister et al.[39] attempted to reduce the cost related to the correction of errors during the design of embedded and real-time systems. The authors propose an automatic verification approach to expect errors at high-level specifications, particularly at Unified Modeling Language (UML) models. The proposed design by the authors starts from the algorithm level, which requests many essential details. In addition, the authors conclude that their proposed verification framework suffers from a low-speed. Therefore, this study does not support the modeling at the conceptual level.

All the previous attempts propose to simulate a heterogeneous system that needs only to move from an environment to another with the minimum parameters to exchange. Most of the studies did not provide an improvement for the V model by supporting the verification at the conceptual level in which the system is described only by their processes associated with their service time. However, in our case, we propose co-simulation interfaces for only a discrete-event model, which requires a well-studied synchronization scheme between simulators to support the data exchange and hardware interrupts with respect to the specificity of each simulator. The proposed co-simulation at the system level (conceptual/functional) represents a real need to predict problems of hardware conflict as soon as possible. As a result, the number of feedbacks from the physical level to the functional level will be reduced.

As a resume of the cited works, Table I classifies research works based on the following criteria: (1) kind of model, (2) environment used, (3) target application, and (4) the technique used for conceptual/functional design. In our study, the co-simulation technique is defined as the cooperation between fair simulators to produce simulation results. Otherwise, the integration technique is based on one simulator that directs the simulation and the second simulator becomes a sub-simulator that can be lunched or not according to the requests.

In light of this brief review, the necessity of a multi-level co-simulation environment is proved. As will be shown in the recent studies below, researchers had faced too many challenges:

- Most of the tools in the conceptual level are unable to simulate or define an intelligent entity.
- The resource of a process is assigned only to estimate time service. Therefore, the architecture of the resource is not possible.
- In queuing systems, designers still use an estimation time for the arrival time of entities and the service time. This estimation is based on probabilistic equations. Practitioners used the historical data collected from the real system to estimate the correspondent equation of the system behavior. In fact, the proposed model presents an estimate solution and the most shortcoming appears for the inexistent system.[40]

This paper is as an attempt to attain the following goals:

- Allow the multi-level design between the conceptual level and the functional level using the co-simulation technique in the case of embedded systems based on the discrete-event model.
- Obtain reliable simulation results of the system outputs by calculating the real-time service from the functional level.

For these reasons, the purpose of this study was to propose a co-simulation technique allowing the synchronization between the Arena simulator for the conceptual level and Matlab/Simulink

TABLE I. Comparison between different attempts to simulate conceptual/functional design.

	Kind of model	Environments	Target application	Technique
Dehghanimohammadabadi et al.,[9]	Discrete-event/continuous	SIMIO/Matlab	Manufacturing systems	Integration
Liang et al.,[27]	Discrete-event/continuous	Arena/Matlab	Manufacturing systems	Integration
Sanz et al.,[28]	Discrete-event/continuous	Arena/Siman and Modelica	Queuing systems	Integration
Süß et al.,[29]	Discrete-event/continuous	PLCOpenXML, Modelica, and Matlab	Manufacturing systems	Co-simulation
Barbieri et al.,[30]	Discrete-event/continuous	Matlab, FlexSim, SysML, IndustrialPhysics, and PLC	Mechatronics system	Co-simulation
Chabibi et al.,[33]	Discrete-event/continuous	Simulink/SysML	Queuing system	Integration
Lora et al.,[35]	Discrete-event	Custom C/C++	Embedded system	Integration
Edward et al.,[36]	Discrete-event/continuous	CyPhySim/Ptolemy II	Cyber-physical system	Commune
Ben Ayed et al.,[37]	Discrete-event/continuous	SystemC/Matlab/ HW accelerator	Mechatronics system	Co-simulation
Törngren et al.,[38]	Discrete-event/continuous	. . .	Cyber-physical system	Co-simulation

simulator for the functional level in the case of the discrete-event model.

IV. TECHNOLOGIES USED

At present, lot of powerful software, such as the MATLAB[41] and the Arena,[8] has been used in literature to model systems. In this section, a brief introduction of Arena and Matlab is emphasized. The Arena framework is used to describe the embedded system as a queuing system and Matlab/Simulink to describe the system at the functional level.

When describing the behavior of an embedded system, several languages could be used. Most of them are designed for the functional or the physical level. Nevertheless, the description of the embedded system in the conceptual level is requested because many systems are verified only for short time simulation. In this case, limits and errors in the size of buffer and frame accumulation in the whole system could be expensive and destructive. Therefore, the verification at an earlier stage at the conceptual level is important.

A. Arena environment for conceptual level

The Arena is a simulation software based on the discrete-event theory.[42] It has both great modeling power and friendly interfaces for users. Arena provides a visual integrated simulation environment.

Systems are described using Arena from the point of view of the entities that flow through them using the available resources. Arena models are structured in a hierarchical and modular way.

The simulation results are usually presented in the form of statistical indicators that are calculated during the simulation. The main purpose of the simulation results is the time advance during the system execution.

The Arena is based on the SIMAN simulation language. Arena modules are high-level constructs whose functionality is equivalent to sets of SIMAN blocks and elements. Arena predefined modules are internally built using SIMAN blocks and elements, which represent lower-level actions.

Arena uses the probabilistic time equation defined in the model to identify the time of the next event. The Arena simulator advances the time and then executes the event, as shown in Fig. 3. At first, the simulator calls time advance subroutine to find the imminent event and advance the internal clock to the imminent event time. Then, it calls an event subroutine to execute the event, updates system state, and puts in the top the next event in the future event list.

The Arena is challenged by several languages/tools,[43] especially the SimEvents library produced by Matlab/Simulink.[37] A brief comparison is presented in Table II. SimEvent library supports more statistics distribution than the Arena tool. However, Arena offers developers an easier graphics user interface than SimEvent. Arena generates a detailed simulation report, which is filled with several simulation results. Dias et al.[44] presented a comparison at the conceptual level between Arena and other tools as FlexSim, Simio, and Quest. The Arena is the best tool as a result of the last comparison.

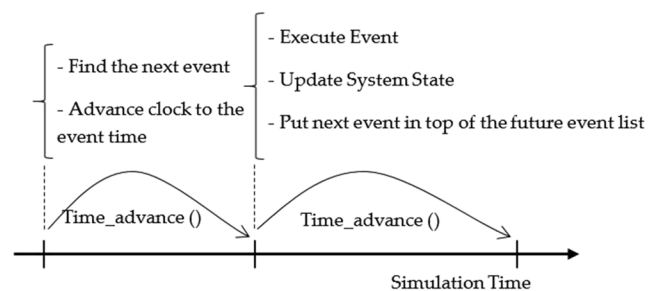As will be shown, the Arena environment seems to be more suitable for the conceptual level design.

FIG. 3. Arena simulator principle.

TABLE II. A theoretical comparison between Arena and SimEvent library.

	SimEvent library	Arena
Statistics distribution	• Does not support only Erlang and Johnson distributions	• Does not support the following distribution: Bernoulli, binomial, geometric, log-logistic, and arbitrary discrete.
Graphics	• Complicate	• Easy to define
Simulation report	• Should be added to the model to be performed	• There is a standard report that is performed. • They let the developer add a specific statistical accumulator.

B. Matlab/Simulink environment for the functional level

Simulink[37] simulator resolves system equations and updates states and outputs of blocks once per integration step, which can be fixed or variable. The order in which the blocks are updated is critical for results validity. The rule of data dependence is used, during the initialization phase, to determine statically the order of blocks activation. In fact, if the block outputs are a function of its inputs, the block must be updated after the blocks that drive its inputs.

Simulating a discrete system requires that the simulator take a simulation step at every sample time hit, that is, at integer multiples of the system's shortest sample time. Otherwise, the simulator might miss key transitions in the system's states. Simulink avoids this by choosing a simulation step size to ensure that steps coincide with sample time hits. The simulator performs the time advance using either a fixed-step or a variable-step discrete solver to solve a discrete system.[45]

V. THE PROPOSED CO-SIMULATION TECHNIQUE FOR CONCEPTUAL AND FUNCTIONAL LEVELS

This section presents an attempt to synchronize between the Arena simulator for the conceptual level and the Matlab simulator for the functional level in the case of an event discrete system. The proposed attempt is based on the simulator's strategies presented in Sec. IV. The co-simulation technique is ensured by communication and synchronization layers.

A. Communication layer

The communication layer presents an essential phase for synchronization. It provides a mechanism to ensure a safe and accurate method to exchange data between simulators. It ensures not only the data transfer between simulators, but also the context switch. There are several methods to ensure the data transfer as shared memory, file, etc. Communication-based on shared memory is used to keep track of the exchanged data between simulators and to ensure less time in the communication layer.

Using both ActiveX Automation, VBA technology, and excel link set up the communication between the Arena and the Matlab. ActiveX automation enables applications to control and manage each other and themselves through an application programming interface (API). The VBA allows the user to write code that automates other applications without installing a separate programming language. A Visual Basic programming environment is received in the Arena; it is accessible through the tools/show the Visual Basic Editor menu option.
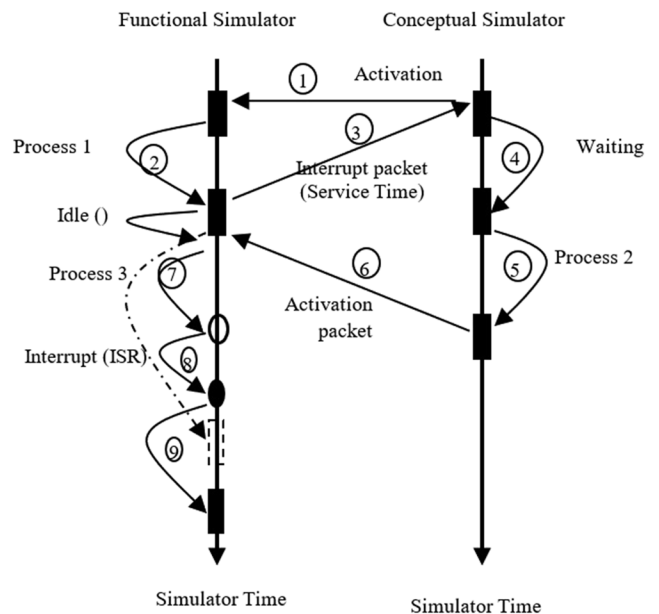

FIG. 4. Communication layer.

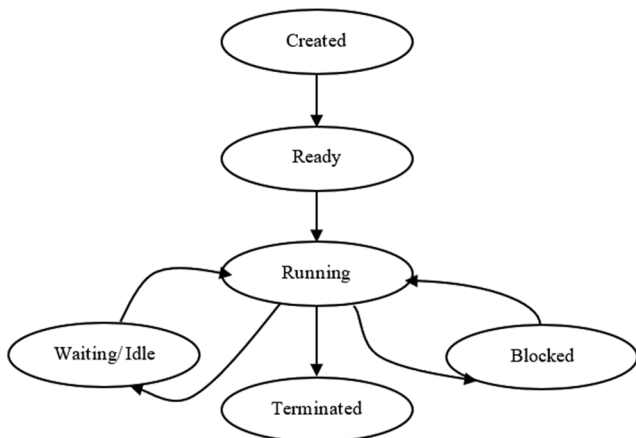FIG. 5. Synchronization scheme between conceptual and functional simulators.

FIG. 6. The system transition model.

This framework environment was adapted for the co-simulation platform and it was enhanced to ensure the interaction between both high-level simulators. The synchronization and the communication are assured by an application programming interface. Figure 4 shows the communication layer between the Arena and the Matlab.

In the Arena environment, the processing time is resolved according to the probability related to the service time. Nevertheless, in the proposed co-simulation environment, the processing time is determined through the time recorded by the model at the functional level. Therefore, the simulation of the model will be more accurate and closely correspond to the real system performance.

The communication layer needs to replace the logic model to the data in an excel file. Therefore, when an entity at the conceptual level enters the "ReadWrite" block, it reads the next value of the processing time returned by the functional level from the excel file and assigns this value to the "Call In" entity attribute. After the entity reads the value from the data file, it moves to the "Delay" module to delay according to the time data in the excel file.

B. Synchronization layer

The proposed synchronization scheme is based on a conceptual simulator and functional simulator mechanism independent of the used tool. The Conceptual simulator is defined as the master because it generates the dataflow. Figure 5 shows the synchronization scheme and the context switch possible between simulators.

The conceptual simulator begins the simulation. When a context switch is required, the simulator sends an activation packet to the functional simulator. The used packet contains the number of subroutines that should be started. Once the process execution is ended, the functional simulator uses an interrupt packet based on service time parameters to perform the context switch. At that moment, the functional simulator changes status to idle and executes an intermediate task. Then, the conceptual simulator moves from the waiting status to the ready status and advances the time according to the received service time. The exception is shown when an interruption has occurred as mentioned in process 3 (arrow 8), for example. In this case, the interruption mechanism is launched. Process 3 saves status parameters and changes its status to stop. This mechanism is ensured by calling the ISR function, which is responsible for context switch between process 3 and the interrupt function. When the interrupt is ended, process 3 is continued (arrow 9).

C. The system transition model

The system transition model describes the state of processes according to the proposed synchronization scheme. The process changes states as shown in Fig. 6.

The system moves into seven states

- **Created**: In the created state, the process waits to be selected by the simulator to start the execution.
- **Ready**: The process moves to the ready state if it is loaded and it is pending to a context switch to start the running step.
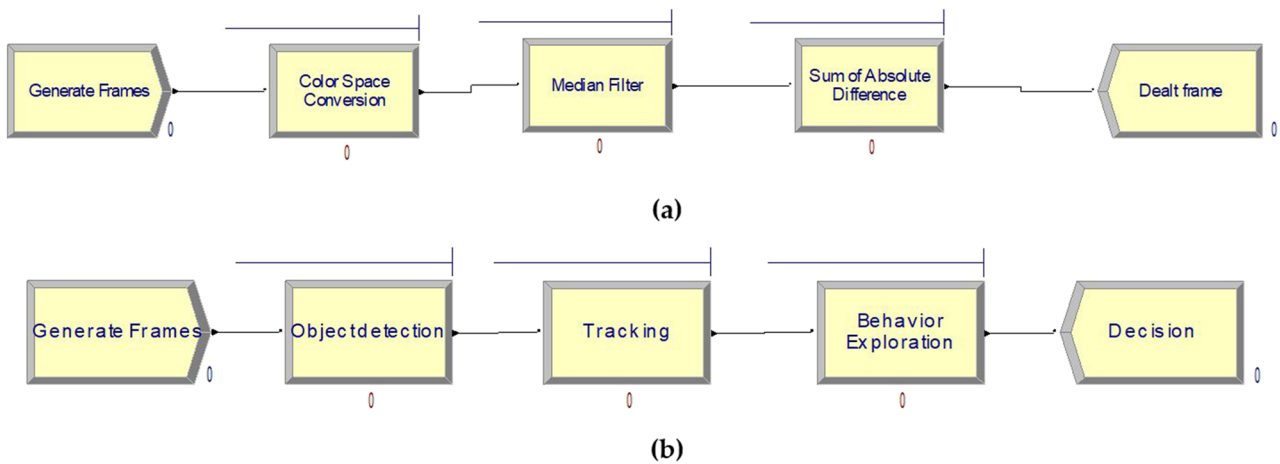

(a)

(b)

FIG. 7. Models using the Arena environment. (a) Motion detection model. (b) Surveillance system based on the suspicious behavior model.
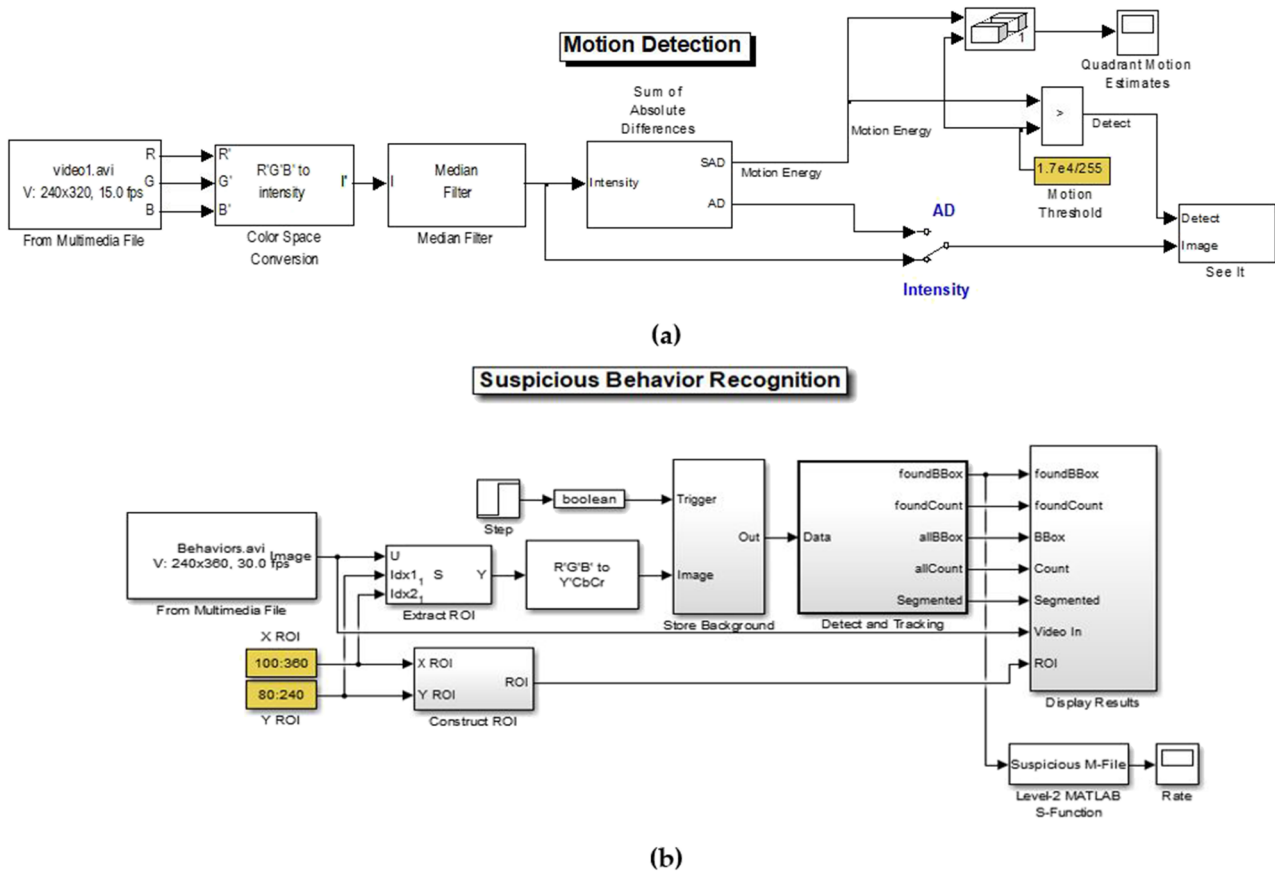
FIG. 8. System model using the Matlab/Simulink environment. (a) Motion detection model and (b) surveillance system based on the suspicious behavior model.

- **Running**: Once the process is selected by the simulator, the process begins with the running phase. The process, at this state, can move to blocked, waiting, or terminated states. Only one process can be selected at the running state at the same time.
- **Waiting/Idle**: When a context switch occurs, the running process moves to the waiting/idle state. In the waiting state, the simulator advances the time, but in the idle state, the time is fixed.
- **Blocked**: When a running process receives an interrupt from the functional simulator, it moves to the blocked state. It returns to the running state when the interrupt is ended.
- **Terminated**: When the process finishes its execution, the process moves from the running state to the terminated state.

VI. EXPERIMENTATIONS

This section presents the evaluation step of the proposed co-simulation environment. A motion detection system and a suspicious behavior recognition system are used as applications to perform the experiment results. This section is divided into five parts.

The first describes systems. The second presents models of systems at the conceptual level based on the Arena environment. The third describes systems at the functional level using the Matlab/Simulink environment. The fourth performs systems at the conceptual and functional level using the proposed co-simulation interface. In addition, the fifth discusses the benefits and limits of the system's model at the conceptual/functional level.

A. System description

Motion detection and suspicious behavior recognition are popular systems used in the surveillance field.
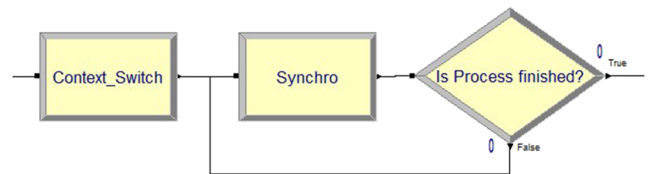

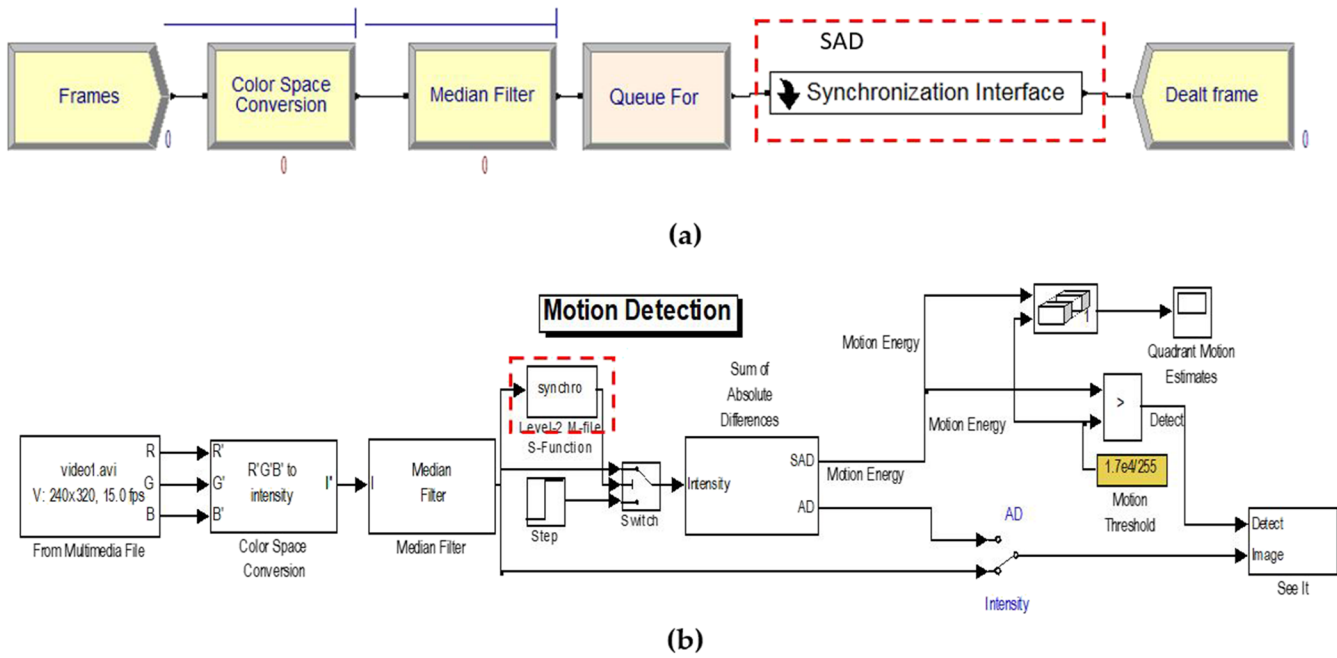**FIG. 9**. Synchronization interface in the side of Arena.

FIG. 10. Motion detection model based on system-level co-simulation. (a) Arena model side and (b) Matalb/Simulink model side.
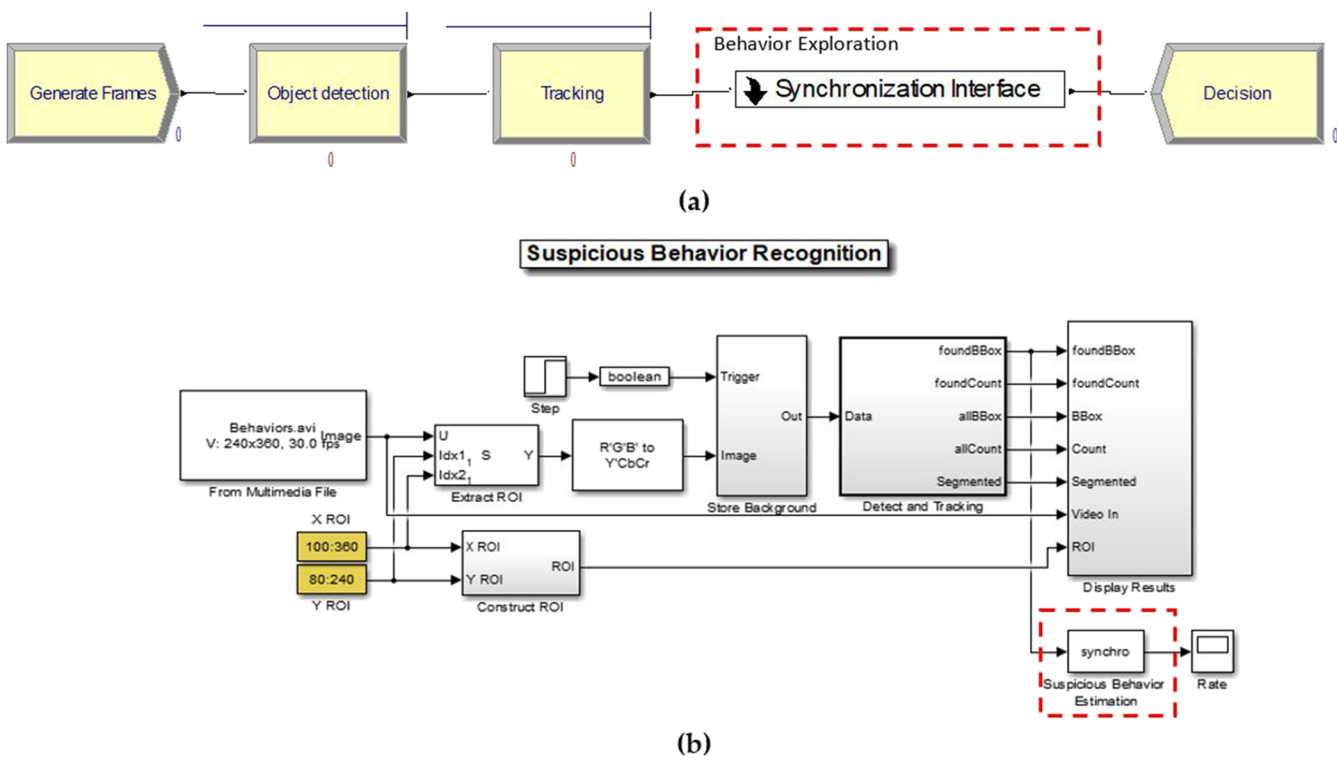

FIG. 11. Suspicious behavior recognition system based on system-level co-simulation. (a) Arena model side and (b) Matlab/Simulink model side.

Motion detection detects any movement in video streaming. The functionality of this system is implemented in three steps: color space conversion, median filter, and sum of absolute differences (SAD).[46]

The SAD method is a popular technique for motion detection in video processing.

The suspicious behavior system recognizes threatened actions. This system is composed of three steps: object detection, tracking, and behavior exploration.[11]

These systems are chosen especially for two reasons:

- The motion detection is implemented as an embedded system.
- The system manages a flow of data (frames).

B. Modeling and simulation at the conceptual level

In this section, Arena models are proposed. As mentioned in Fig. 7, models are composed of: one Create module, three Process modules, and one Dispose module. All processes use the same processor as a resource. The Arena's model for the two systems was standard because the model presents the time estimation of each module independent of the chosen algorithm. If the algorithm changed, only the service time parameter will be changed in each process module.

C. Modeling and simulation at the functional level

Matlab/Simulink provides a standard model of the motion detection system and the suspicious behavior recognition system, as shown in Fig. 8. These models represent the functional level of systems. Therefore, models could be modified by changing the used algorithm. For example, we could replace the median filter by the Gabor filter and the sum of absolute difference by the matrix comparison algorithm in the case of motion detection. This level provides more details about the system and the choice of algorithms is critical.

D. Modeling and simulation at the system level

The co-simulation is ensured by interfaces. The purpose of the interface is to perform the communication layer and the synchronization layer described in Sec. V.

Previous works, cited in the literature section, use the VBA package and Excel link provided by Arena to perform the communication. This communication technique opens the environment target and builds it. Therefore, the communication time increases rapidly and reached bottlenecks for a long time of execution.

We propose a communication layer based on Write/Read package without using the VBA package. A Write package is used to make the context switch from Arena to Matlab/Simulink. Then, a loop structure is performed to verify if a data (service time issued by Matlab/Simulink) was written in the input file, as mentioned in Fig. 9.

This interface uses two "ReadWrite" modules and a "Decide" module tied as a loop structure. It is performed as a synchronization interface in the Arena side, as shown in Figs. 10(a) and 11(a).

In the Matlab/Simulink side, Figs. 10(b) and 11(b), the synchronization is ensured by the "synchro" interface. This interface is based on an s-function module written in C language, as shown in Algorithm 1.

ALGORITHM 1. Synchro.

```
int *Synchro (int A[x])

{

int mem_id;

void* ptr_shared_memory

. . . . . .

while (ptr_shared_memory == true

        if (ptr_shared_memory != NULL)

        {

                Process_ID (time_stamp)

                If (Interrupt == true)

                {

                    t=time_stamp

                    Call(ISR)

                }

                time_stamp = t – tic

        }

    * ptr_shared_memory = time_stamp

    return ptr_shared_memory;

}
```

Figures 10 and 11 show systems modeled using the Arena environment at the conceptual level and the Matlab/Simulink environment at the functional level.

Figure 12 presents the category overview report. These reports describe results associated with the entities' waiting time and the waiting entities' number. For an embedded system, such information is very useful to understand deeply the interchange the process and to build intermediary memory. Based on results in Ref. 36, the simulation in the functional level verifies the functionality of each step of the system without highlighting constraints of the inter-exchange data between steps. In return, results shown by the Arena environment express some problems. For the motion detection system, the average waiting time for each entity (frame) related to processes is about 0.8 ms and the maximum number of waiting entities (frames in our case) in the queue is about 280 [see Fig. 12(a)].

For the suspicious behavior recognition system, the average waiting time of the behavior exploration process is about 1500 s and the maximum number of waiting entities (frames in our case) in the queue is about 3028 [see Fig. 12(b)]. The number of waiting frames related to the object detection process and the tracking process is limited and did not request a large memory. In contrast, the number of waiting frames in the behavior exploration queue was bigger and should be the cause of memory insufficiency.

FIG. 12. Category overview report. (a) Simulation results for the motion detection system and (b) simulation results for suspicious behavior recognition.

It is necessary to mention that in the real case, frames would be removed from the memory due to the lack of space. However, in the simulation case, especially at the conceptual level, hardware mechanisms are ignored.

The simulation results prove the necessity to add buffers with each process to avoid the loss of frames and request a hardware description to increase the execution speed. Likewise, results highlight the importance of simulation at the conceptual level and functional level.

E. Discussion and comparison

This section presents the results of the experiment stage for 5000 s simulation time. Simulations were conducted on a Windows 8 computer with a 2.2 GHz Intel Core i5-5200U CPU. Table III presents the recorded simulation results in Arena, Matlab/Simulink, and co-simulation environment. The time is monitored under the same condition during the replication (REP) execution.

The experiment results of simulation time prove that the proposed co-simulation Arena/Simulink is decreased in comparison with Arena about 17%. This rate is important especially when the simulation time is bigger. This result is well explained because the service time at the conceptual level is an estimation although the synchronization time is added to the whole simulation time.

It is necessary to mention that the simulation time of systems at the functional level is calculated for every replication, and the context switch is made also for all frames. We suppose that the motion detection system and suspicious behavior recognition system are based on a different camera. For these reasons, the frame size and quality are different and the context switch should be performed every replication. As dealt below, the main goal of the co-simulation conceptual/functional is to use the real service time calculated by the functional level and not the estimated service time.

TABLE III. Performance results using different simulation environments.

	Motion detection		Suspicious behavior	
	REP	Time (s)	REP	Time (s)
Arena	20	56.472	20	83.964
Matlab/Simulink	1	8.12	1	45.86
Proposed co-simulation Arena/Simulink	20	48.257	20	70.024

In fact, the service time was known for the first frame and used identically for the other frames. That is why the simulation time for the co-simulation environment is a little higher than the Arena simulation.

Based on the results shown in Fig. 12, the number of entities in the queue of the SAD process (motion detection system) and the behavior exploration process (Suspicious behavior recognition system) seems to be big and unstable. Simulations are run for a different duration to verify if the number will be stable. Figure 13 presents the evolution of entities in queues for a different duration. The shape of the curve demonstrates that the number of waiting entities is still increased over time. The curves have an increasing appearance. Consequently, queues related to the discussed processes are overloaded and this bottleneck will lead to conflicts on memory management. Therefore, the size of the intermediary memory is not fixed. Knowing that in the absence of an intermediary memory between processes, the loss of frames will occur when the average processing time of a frame exceeds the length of the frame. Even if the designer seizes an intermediary memory for storage to avoid the loss of the frame in the surpassed case, the problem occurs because the requested size of the memory is still increasing.

Functional level designers could detect this problem when computing the average processing time of a frame. However, they did not know if adding or seizing memory represents a solution to this problem. Then at the physical level, this problem occurs under the form of hardware conflicts.

Therefore, only in conceptual level simulation, designers could approve if adding or seizing a memory will solve this problem or not. The accuracy of the achieved simulation results at this level is conditioned by the accuracy of the service time of the process. For that, the proposed system-level co-simulation is not only the strongest method to predict the problem of lack or insufficiency of memory at the first stage of modeling, but also provides more details about memory usage.

As a conclusion, the motion detection system and suspicious behavior recognition system could not be implemented in standard architecture as PIC board, Arduino, or Raspberry. This system should be implemented using the co-design approach [Hardware (HW) and SW description], Graphics Processing Unit (GPU) accelerator, or multi-processor design.

Considering this discussion, Table IV resumes the benefits and the limits of each simulation environment.
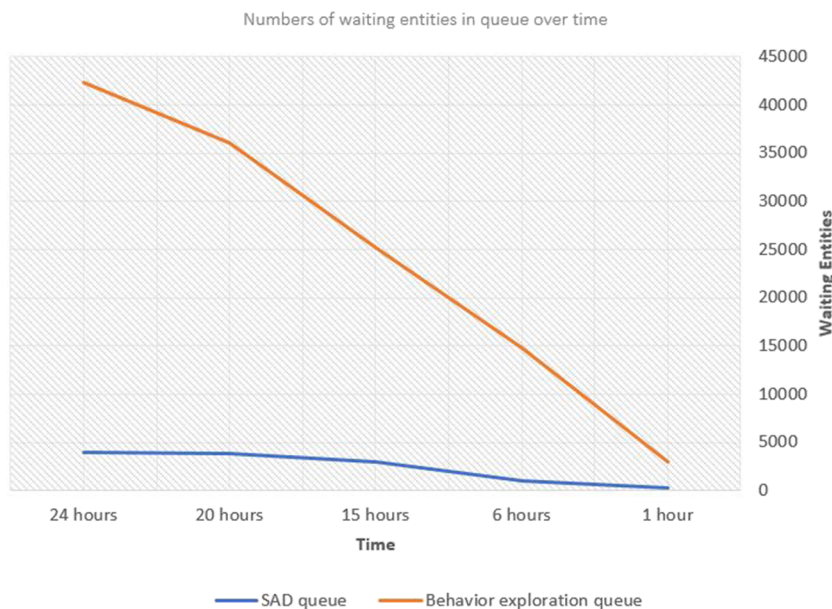

FIG. 13. Progress of entities in the SAD queue and behavior exploration queue.

TABLE IV. Benefit and limit review of the simulation tool.

	Benefits	Limits
Conceptual level (Arena environment)	• Easy to model. • Pursuit the entity during the simulation. • Support the queuing system theory. • The generated default report is rich with statistical values.	The service time is estimated using probabilistic functions. The model uses only the event's time independently with adequate function. In the case of an embedded system, there are some details that are required to obtain an acceptable simulation result. Customization is not supported.
Function level (Matlab/Simulink)	• The execution time (service time terminology in conceptual level) is calculated based on the used function. It is not a prediction. • The model is easily customized.	Do not considerate the queue size between different components. The advance of global simulation time during a long time is not the purpose of the design at the functional level. The model could be designed when the system detail is well known.
Proposed system-level co-simulation (Arena/Simulink)	• The service time presents the exact time based on functional level results. • Customization/optimization is supported. • The model is designed with favorite details (fewer details or more details). • Buffer size is assigned to queue status. • Simulation time is decreased by 17% compared with the Arena simulation time.	Do not support the continuous models.

VII. CONCLUSION

It is still difficult to simulate the system behavior with sufficient details at the conceptual level. This paper proposed a new co-simulation interface between the conceptual model implemented in the case of the Arena and the functional model implemented in the case of Matlab/Simulink. A system based on motion detection was modeled to prove the efficiency of the proposed contribution.

The main conclusions are that the co-simulation interface allows developers to benefit from the advantages of the conceptual level and the functional level at the same time, especially the reduction in the time-to-market and the support of the functional model. The improvement proposed to the V model at the system level reduces the feedback times requested to correct/improve the design. The simulation time is slightly larger than the Arena simulation time due to the added time of the context switch between simulators. We believe this advancement adds a new dimension to the simulation modeling approaches. This would allow developers to customize and optimize their models at the first steps of modeling.

In the light of this deep study, the proposed system-level co-simulation technique requests to be extended to support cooperation at the physical level. This future work proposition will let engineering to model/simulate a heterogeneous model.

ACKNOWLEDGMENTS

DATA AVAILABILITY

The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to their containing information that could compromise the privacy of research participants.

REFERENCES

[1] D. Niyonkuru and G. A. Wainer, "Discrete-event modeling and simulation for embedded systems," Comput. Sci. Eng. **17**(5), 52–63 (2015) (in English).

[2] T. Y. Zhu, L. Wang, J. Meng, and IEEE, "The exploitation and discussion of new mobile healthcare system model based on smart phone," in *2013 10th IEEE International Conference on Networking, Sensing and Control* (IEEE, New York, 2013), pp. 497–502.

[3] M. Ben Ayed and S. Elkosantini, "An accelerated architecture based on GPU and multi-processor design for fingerprint recognition," Int. J. Adv. Comput. Sci. Appl. **7**(3), 337–348 (2016) (in English).

[4] S. Elkosantini, "Toward a new generic behavior model for human centered system simulation," Simul. Modell. Pract. Theory **52**, 108–122 (2015) (in English).

[5] F. Fummi, M. Loghi, M. Poncino, and G. Pravadelli, "A cosimulation methodology for HW/SW validation and performance estimation," ACM Trans. Des. Autom. Electron. Syst. **14**(2), 1–32 (2009) (in English).

[6] T. Ma, S. Ali, and T. Yue, "Modeling foundations for executable model-based testing of self-healing cyber-physical systems," Software Syst. Model. **18**(5), 2843–2873 (2019) (in English).

[7] Y. Tang and N. W. Bergmann, "A hardware scheduler based on task queues for FPGA-based embedded real-time systems," IEEE Trans. Comput. **64**(5), 1254–1267 (2015) (in English).

[8] J. G. Shanthikumar, S. W. Ding, and M. T. Zhang, "Queueing theory for semiconductor manufacturing systems: A survey and open problems," IEEE Trans. Autom. Sci. Eng. **4**(4), 513–522 (2007) (in English).

[9] M. Dehghanimohammadabadi and T. K. Keyser, "Intelligent simulation: Integration of SIMIO and MATLAB to deploy decision support systems to simulation environment," Simul. Modell. Pract. Theory **71**, 45–60 (2017) (in English).

[10] M. S. Durmus, I. Ustoglu, R. Y. Tsarev, and J. Borcsok, "Enhanced V-model," Inf.-J. Comput. Inf. **42**(4), 577–585 (2018) (in English).

[11] M. Ben Ayed, S. Elkosantini, and M. Abid, "An automated surveillance system based on multi-processor and GPU architecture," Eng. Technol. Appl. Sci. Res. **7**(6), 2319–2323 (2017) (in English).

[12] M. B. Ayed, F. Bouchhima, L. Zouari, and M. Abid, "An accelerated hardware software in the loop technique for control units," Int. J. Mech. Mechatronics Eng. **14**(3), 10–21 (2014).

[13] L. Zouari, M. Ben Ayed, M. Abid, and IEEE, "Embedded control of robot arm driven by brushless DC motor on FPGA," in *2014 Second World Conference on Complex Systems* (IEEE, New York, 2014), pp. 722–727.

[14] L. Zouari, M. Ben Ayed, M. Abid, and IEEE, "Improved performance of a brushless DC motor using hardware in the loop control technique," in *2015 IEEE 12th International Multi-Conference on Systems, Signals & Devices* (IEEE, New York, 2015).

[15] T. Habib, "System design of mechatronic products models and methods to utilize mass customization," thesis report (2014).

[16] D. D. Gajski, F. Vahid, S. Narayan, and J. Gong, *Specification and Design of Embedded Systems* (Prentice-Hall, 1994).

[17] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems* (Springer Science & Business Media, 2009).

[18] N. Bombieri, M. Poncino, and G. Pravadelli, *Smart Systems Integration and Simulation* (Springer, 2016).

[19] Siemens Team, "Keiper speeds up product development with Tecnomatix, Siemens PLM Software's digital manufacturing technology," Assem. Autom. News Item **29**(2), 187 (2009) (in English).

[20] A. D. Smith, "SolidWorks 96 is a solid bet," Comput. Graph. World **19**(11), 91–101 (1996).

[21] J. A. Modi, *Simulation Modeling and Analysis with ARENA* (JSTOR, 2008).

[22] S. Friedenthal, A. Moore, and R. Steiner, *A Practical Guide to SysML: The Systems Modeling Language* (Morgan Kaufmann, 2014).

[23] A. Khecharem, C. Gomez, J. DeAntoni, F. Mallet, and R. de Simone, "Execution of heterogeneous models for thermal analysis with a multi-view approach," in *Proceedings of the 2014 Forum on Specification & Design Languages* (IEEE, New York, 2014).

[24] M. Bergert and J. Kiefer, "Mechatronic data models in production engineering," IFAC Proc. Vol. **43**(4), 60–65 (2010).

[25] C. Meng, S. S. Nageshwaraniyer, A. Maghsoudi, Y.-J. Son, and S. Dessureault, "Data-driven modeling and simulation framework for material handling systems in coal mines," Comput. Ind. Eng. **64**(3), 766–779 (2013).

[26] M. B. Ayed, Y. B. Salah, and M. Abid, "Conceptual/functional Co-simulation technique for embedded systems," in *2019 International Conference on Computer and Information Sciences (ICCIS)* (IEEE, 2019), pp. 1–5.

[27] S. A. Liang and X. F. Yao, "Multi-level modeling for hybrid manufacturing systems using arena and MATLAB," in *2008 International Workshop on Modelling, Simulation and Optimization, WMSO* (IEEE Computer Society, Los Alamitos, 2009), pp. 155–159.

[28] V. Sanz, A. Urquia, F. E. Cellier, and S. Dormido, "Hybrid system modeling using the SIMANLib and ARENALib modelica libraries," Simul. Modell. Pract. Theory **37**, 1–17 (2013) (in English).

[29] S. Suss, A. Strahilov, C. Diedrich, and IEEE, "Behaviour simulation for virtual commissioning using Co-simulation," in *Proceedings of 2015 IEEE International Conference on Emerging Technologies and Factory Automation-ETFA* (IEEE, New York, 2015).

[30] G. Barbieri, G. Goldoni, R. Borsari, and C. Fantuzzi, "Modelling and simulation for the integrated design of mechatronic systems," IFAC-PapersOnLine **48**(10), 75–80 (2015).

[31] W. B. Nordgren, "Flexsim simulation environment," in *Proceedings of the 2003 Winter Simulation Conference* (IEEE, New York, 2003), Vols. 1 and 2, pp. 197–200.

[32] K. Hitomi, *Manufacturing Systems Engineering: A Unified Approach to Manufacturing Technology, Production Management and Industrial Economics* (Routledge, 2017).

[33] B. Chabibi, A. Douche, A. Anwar, and M. Nassar, "Integrating SysML with simulation environments (Simulink) by model transformation approach," in *2016 IEEE 25th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)* (IEEE, 2016), pp. 148–150.

[34] C. D. Pegden and IEEE, "Simio: A new simulation system based on intelligent objects," in *Proceedings of the 2007 Winter Simulation Conference* (IEEE, New York, 2007), Vols. 1-5, pp. 2272–2279.

[35] M. Lora, S. Vinco, and F. Fummi, "Translation, abstraction and integration for effective smart system design," IEEE Trans. Comput. **68**(10), 1525–1538 (2019) (in English).

[36] E. A. Lee, M. Niknami, T. S. Nouidui, M. Wetter, and IEEE, "Modeling and simulating Cyber-physical systems using CyPhySim," in *2015 Proceedings of the International Conference on Embedded Software* (IEEE, New York, 2015), pp. 115–124.

[37] M. Ben Ayed, F. Bouchhima, and M. Abid, "CODIS plus: Co-simulation environment for heterogeneous systems," Control Eng. Appl. Inf. **20**(1), 98–107 (2018) (in English).

[38] M. Törngren and U. Sellgren, "Complexity challenges in development of cyberphysical systems," in *Principles of Modeling* (Springer, 2018), pp. 478–503.

[39] M. A. Wehrmeister, J. G. Packer, and L. M. Ceron, "Support for early verification of embedded real-time systems through UML models simulation," ACM SIGOPS Oper. Syst. Rev. **46**(1), 73–81 (2012).

[40] N. Sadeghi, A. R. Fayek, and N. G. Seresht, "A fuzzy discrete event simulation framework for construction applications: Improving the simulation time advancement," J. Constr. Eng. Manage. **142**(12), 04016071 (2016) (in English).

[41] E. A. Catchpole, "MATLAB: An environment for analyzing ring-recovery and recapture data," J. Appl. Stat. **22**(5-6), 801–816 (1995) (in English).

[42] T. Altiok and B. Melamed, *Simulation Modeling and Analysis with Arena* (Elsevier, 2010).

[43] L. S. Dias and S. Luís, "Automatic interactive modelling of simulation," Ph.D. thesis, written in Portuguese-Modelação Automática Interactiva de Simulação, 2005.

[44] L. Dias, G. Pereira, and G. Rodrigues, "A shortlist of the most popular discrete simulation tools," Simul. News Eur. **17**(1), 33–36 (2007).

[45] F. Bouchhima, G. Nicolescu, E. M. Aboulhamid, and M. Abid, "Generic discrete–continuous simulation model for accurate validation in heterogeneous systems design," Microelectron. J. **38**(6-7), 805–815 (2007).

[46] S. Chandana, "Real time video surveillance system using motion detection," in *2011 Annual IEEE India Conference* (IEEE, 2011), pp. 1–6.