(12)Indian Patent Application

---

(21) Application Number: 202241053031

(22) Filing Date: 16/09/2022 (43) Publication Date: 22/03/2024

(71) Applicant(s): L&T TECHNOLOGY SERVICES LIMITED

(72) Inventor(s): Komath, Vimal
Govindarajan, Stalin

(51) International Classifications: G06F 11/36 G06F 11/30 G06F 11/26 G06F 117/08 G06F 30/33

(54) Title: METHOD AND SYSTEM FOR VIRTUALLY TESTING A HARDWARE SYSTEM

(57) Abstract: A method and a system of virtually testing an embedded hardware system is disclosed. The hardware system is tested through a hardware testing system comprising a plurality of hardware testing sub-systems, each implementing a respective testing application. A common data channel is established for the plurality of hardware testing sub-systems to intercommunicate and to communicate with one or more simulation models. A hardware abstraction layer (HAL) of a respective testing application associated with each of the plurality of hardware testing sub-systems is configured by compiling a source code associated with the respective testing application with a virtualized driver package. The virtualized driver package may route application layer calls to the common data channel. Each of the plurality of hardware testing sub-systems are communicatively linked with the one or more simulation models via the common data channel and the configured HAL, for testing the hardware system.

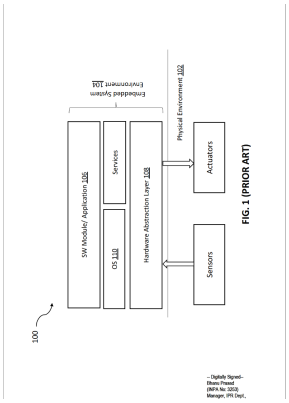

FIG. 1 (PRIOR ART)

FORM 2

THE PATENTS ACT 1970
(39 OF 1970)
&
The Patent Rules, 2003

Complete Specification
(See Section 10 and Rule 13)

1. TITLE OF THE INVENTION

METHOD AND SYSTEM FOR VIRTUALLY TESTING A HARDWARE SYSTEM

2. APPLICANT(S)

(a) NAME : **L&T TECHNOLOGY SERVICES LIMITED**
(b) NATIONALITY : **INDIAN**
(c) ADDRESS : DLF IT SEZ Park, 2nd Floor – Block 3
1/124, Mount Poonamallee Road,
Ramapuram, Chennai – 600 089,
INDIA.

3. PREAMBLE TO THE DESCRIPTION

COMPLETE

The following specification describes the invention and the manner in which it is to be performed

DESCRIPTION

Technical Field

[001] This disclosure relates generally to testing embedded systems, and more particularly to a system and method of virtually testing an embedded system software.

BACKGROUND

[002] Embedded systems may include a software component and hardware component. The software component may be implemented using different software modules which may be integrated logically to be tested as a group. In performing integration testing, the interface between software modules may be tested along with the hardware interface through a hardware abstraction layer. However, since, the software comprises of different software modules and each software module may be coded by different programmers and could use different processing environments. Further, the output of one software module may act as input to another software module. Thus, determining the correctness of the interface between the two software modules is important for correct functioning of the entire software. Further, in a scenario the software module may be required to be tested with respect to its interface with a hardware which may not be available. In such scenarios, a mock software module or a mock of the hardware or a proxy hardware may be used to perform integration testing. However, in doing so the integration testing of the embedded system is done with a mock interface and not the actual software. Thus, a true integration testing is only possible at a hardware level and cannot be performed at software based environment.

[003] However, it becomes difficult for the software developers to test the software when the physical hardware is not available, or the physical hardware is located at remote regions. Further, the modeling of hardware components using Hardware Description Language (HDL) in Verilog requires customized workstations and may not be suitable for testing of all software. This may lead to ineffective testing of the software in absence of the hardware and lead to delays and capital losses.

[004] Therefore, there is a need in the art to provide a system that enables a true integration testing on a multi-processor or controller embedded environment.

SUMMARY OF THE INVENTION

[005] In an embodiment, a method of virtually testing a hardware system is provided. The hardware system may be tested through a hardware testing system comprising a plurality of hardware testing sub-systems. It may be noted that each hardware testing sub-system may

implement a respective testing application. Further, the method may comprise establishing a common data channel for the plurality of hardware testing sub-systems. It may be noted that the common data channel may enable a plurality of hardware testing sub-systems which may intercommunicate and may further enable each of the plurality of hardware testing sub-systems to communicate with one or more simulation models. The method may further include configuring a hardware abstraction layer (HAL) of a respective testing application associated with each of the plurality of hardware testing sub-systems by compiling a source code associated with the respective testing application with a virtualized driver package. The virtualized driver package may be configured to route application layer calls to the common data channel. Further, the method may include communicatively linking each of the plurality of hardware testing sub-systems with the one or more simulation models via the common data channel and the configured hardware abstraction layer (HAL), for testing the hardware system.

[006] In an embodiment, a system for virtually testing a hardware system is disclosed. The system may comprise one or more processors in a hardware testing system and a memory storing a plurality of processor executable instructions which upon execution cause the one or more processors to establish by the hardware testing system a common data channel for the plurality of hardware testing sub-systems. It may be noted that the common data channel may enable the plurality of hardware testing sub-systems to intercommunicate and may further enable each of the plurality of hardware testing sub-systems to communicate with one or more simulation models. Further, the processor executable instructions, on execution, may further cause the processor to configure a hardware abstraction layer (HAL) of a respective testing application associated with each of the plurality of hardware testing sub-systems by compiling a source code associated with the respective testing application with a virtualized driver package. It may be noted that the virtualized driver package may be configured to route application layer calls to the common data channel. Further, the processor executable instructions, on execution, may further cause the processor to communicatively link each of the plurality of hardware testing sub-systems with the one or more simulation models, via the common data channel and the configured hardware abstraction layer (HAL) for testing the hardware system.

BRIEF DESCRIPTION OF THE DRAWINGS

[007] The accompanying drawings, which are incorporated in and constitute a part of this disclosure, illustrate exemplary embodiments and, together with the description, serve to explain the disclosed principles.

[008] FIG. 1 illustrates a generic architecture of an embedded system, in accordance with a prior art embodiment.

[009] FIG. 2 illustrates an architecture of an embedded system, in accordance with an embodiment of the present disclosure.

[0010] FIG. 3 illustrates an architecture of the embedded system in a testing system, in accordance with an embodiment of the present disclosure.

[0011] FIG. 4 illustrates an architecture of the virtual HAL of **FIG. 3**, in accordance with an embodiment of the present disclosure.

[0012] FIG. 5 illustrates a detail block diagram of the simulation stack, in accordance with an embodiment of the present disclosure.

[0013] FIG. 6 illustrates functional modules of the testing system enabling integration testing of an embedded system software, in accordance with an embodiment of the present disclosure.

[0014] FIG. 7 illustrates a method for virtually testing a hardware system, in accordance with an embodiment of present disclosure.

DETAILED DESCRIPTION OF THE DRAWINGS

[0015] Exemplary embodiments are described with reference to the accompanying drawings. Wherever convenient, the same reference numbers are used throughout the drawings to refer to the same or like parts. While examples and features of disclosed principles are described herein, modifications, adaptations, and other implementations are possible without departing from the spirit and scope of the disclosed embodiments. It is intended that the following detailed description be considered as exemplary only, with the true scope being indicated by the following claims. Additional illustrative embodiments are listed below.

[0016] In the figures, similar components and/or features may have the same reference label. Further, various components of the same type may be distinguished by following the reference label with a second label that distinguishes among the similar components. If only the first reference label is used in the specification, the description is applicable to any one of the similar components having the same first reference label irrespective of the second reference label.

[0017] In the present scenario, with the rise in number of startups, the number of innovative products entering the market have increased. In a common practice, the innovative products may include software applications, etc. which may be implemented using embedded systems. Designing such systems requires capital and resources which may not be readily available. The software components of such systems may however be created and validated

and tested. Also, for testing the software an integration testing of all the various software modules may be required. To perform the functional verification and to validate the integration of the software modules with simulated hardware.

[0018] The objective of the current invention is to overcome the hurdle of dependency over the physical environment by providing a framework that may run the software code of a software module on a virtual environment without any dependency over the physical environment. The invention aims to virtualize the corresponding physical sensors and actuators providing a real-world stimulus, by using a virtualized driver packages into the testing framework of a testing device.

[0019] FIG. 1 illustrates a generic architecture of an embedded system, in accordance with a prior art embodiment. **FIG. 1** depicts a generic architecture 100 which includes a physical environment 102 comprising computer-implemented hardware components such as, but not limited to, sensors and actuators and one or more communication channels. The generic architecture 100 may comprise of an embedded system environment 104 comprising an application layer 106 which may be responsible for implementing the logical and functional aspect of the embedded system and may exchange data with the physical environment 102 by making application calls to the hardware abstraction layer (HAL) 108. The HAL 108 may be a part of the operating system layer (OS) 110 and may be implemented as a software abstraction layer between the physical environment 102 and OS 110. In general, the HAL 108 may include bootloader, board support package (BSP), device drivers, and other components.

[0020] FIG. 2 illustrates an architecture of an embedded system 200, in accordance with an embodiment of the present disclosure. The embedded system 200 may be under development and is required to be tested through the testing system 300. In an embodiment, the embedded system 200 may involve integration of one or more sub-systems 202a-n. The one or more sub-systems 202a-n may include an existing sub-system 210 and a newly-designed subsystem 220. In an embodiment, the embedded system 200 is shown and described as having one existing subsystem 110 and one newly-designed subsystem 120. However, it will be appreciated that the embedded system 200 can include any number and/or configuration of sub-systems 202a-n comprising the existing subsystems 210 and/or newly-designed subsystems 220. The existing subsystem 210 and the newly-designed subsystem 220 are configured to provide at least one hardware function, which contributes to the overall system function of the embedded system 200. Each of the hardware functions can be provided by one or both of the existing subsystem 210 and the newly-designed subsystem 220. When development is complete, the embedded system 200 can be integrated into an electronic package, such as a

system-on-chip (SoC) module, and the existing subsystem 210 and the newly-designed subsystem 220 can be coupled, and configured to communicate, via one or more communication channels, such as a communication bus 204, as shown in FIG. 2. As shown in FIG. 2, the existing subsystems 210 may include any type of central processing units (CPUs) 212, memory systems 214, field programmable gate arrays (FPGAs), or a combination thereof, and are configured to communicate with each other via one or more internal communication channels, such as an internal communication bus 216.

[0021] FIG. 3 illustrates an architecture of the embedded system 200 in a testing system 300, in accordance with an embodiment of the present disclosure. During development, the existing subsystem 210 and the newly-designed subsystem 220 of the embedded system 200 can be provided on the testing system 300 for testing as illustrated in FIG. 3. The testing system 300 may provide a plurality of system testing elements, such as one or more hardware testing sub-system comprising a virtual hardware abstraction layer (HAL) 302 each. The testing system 300 may further provide simulated physical environment 305 which may include a simulation (SIM) stack 304 and visualization and test (VT) stack 306, which are coupled, and configured to communicate via a communication (COMM) stack 307. Thus, the testing system 300 are configured to execute a portion of the overall system function of the embedded system 200 and/or to provide testing and/or debugging services for evaluating the embedded system 200. Although the testing system 300 shown and described below as comprising physical elements such as CPU (not shown) and memory (not shown) for implementing the virtual system testing elements, it is understood that the testing system 300 can be extended to any number and/or configuration of physical elements. In an embodiment, the physical elements may be shared between the existing subsystem 210 and the newly-designed subsystem 220 and/or other sub-systems (220n).

[0022] The existing subsystem 210 and the newly-designed subsystem 220 may be provided in the testing system 300 to comprise an application layer 308 and a virtual HAL 302. The application layer 308 may include an application code 312 and an HAL 310. The virtual HAL 302 may virtualize the hardware dependency of the application layer 308 through HAL 310 which may comprise of special function registers, interrupts and peripheral interfaces in order for the application layer 308 to utilize the physical elements including the processing environment of the testing system 300. In an embodiment, the virtual HAL 302 may mask all the hardware calls/interactions from the HAL 310 which in general is the last point of interaction between the application layer 308 and physical hardware of an embedded system. The masking of the hardware calls/interactions from the HAL 310 by the virtual HAL 302 may

be done by providing the necessary data to the application layer 308, which may be provided by the physical hardware of an embedded system. In an embodiment, the newly-designed subsystem 220 may

[0023] In an embodiment, virtual HAL 302 may include virtualized driver package which may re-compile a source code of the application layer 308 in order to route the application layer 308 calls to a generic data channel (not shown) established by the testing system 300. The generic data channel may get linked to other components in the testing system 300 responsible to respond to stimuli from the application layer 308. Thus, the testing system 300 may create a closed loop as well as an open loop interaction by means of SIM stack 304 which may implement different one or more simulation models such as, but not limited to, (eg. Matlab and Simulink models and simulators) that emulate the physical environment. The virtual HAL 302 may push the application layer 308 calls to the physical hardware into the COMM Stack layer 307 for the one or more simulation models provided by the SIM stack 304 to consume. The one or more simulation models provided by the SIM stack 304 layer may be controlled using the VT stack 306 for different scenario creation. In an embodiment, the COMM stack layer 307 may provide a common channel of interaction for all the sub-systems 202 that interact with each other in the testing system 300.

[0024] In an embodiment, the SIM stack 304 may include but not limited to a memory simulator 314 (EEPROM, Data Flash etc), an ADC simulator 316 (Internal, Extrenal ADC connected over SPI), a stepper simulator 318 (Stepper motor simulation model with direction control and home sensing simulation), a plant model 320 (external physical system simulation), a protocol simulator 322, a scenario simulator 324 (simulator model to create negative and positive sequence) that can implement respective functionality as per the intended use case/physical interaction. In an embodiment, the memory simulator 314 may provide a simulation of a memory which may save data corresponding to a hardware's outputs as a test output to be provided to the virtual HAL 302 in response the calls from the application layer 308. The ADC simulator 316 may simulate an analog to digital convertor in order to convert analog data to digital data.

[0025] In an embodiment, the VT stack 306 may include an AR visualizer 326 and a test engine framework 328. The AR visualizer may provide augmented reality visualization elements like dashboard or an input/output interface that can feed data to the application layer 308. The AR visualization 326 may include augmented reality models for example Vuforia, Wikitude, ARKIT, ARCore, MaxST, Kudan, etc. wherein the application under test may be provided with a hardware like simulation. Further, with the use of AR visualization 326, the

system may be able to simulate results similar to the results which may be achieved by using actual hardware devices. The test engine framework may include test runners and test API (API to interface to the framework) that can invoke scripts to automate test scenarios and achieve functional testing of the system.

[0026] In an embodiment, the testing system 300 may be implemented in a testing device (not shown) which may be a computing device comprising one or more computer-implemented hardware components, software components and one or more communication channels in the COMM stack 307. In particular, the testing device (not shown) may have the capability to communicatively link a plurality of sub-systems 202 with the simulated physical environment 305 via a common data channel and a virtual HAL 302. Examples of the hardware testing device 102 may include, but are not limited to a desktop, a laptop, a notebook, a netbook, a tablet, a smartphone, or a mobile phone.

[0027] The COMM stack 307 may support wired and wireless communication protocols may include, but are not limited to, a Transmission Control Protocol and Internet Protocol (TCP/IP), User Datagram Protocol (UDP), Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP), Zig Bee, EDGE, IEEE 802.11, light fidelity (Li-Fi), 802.16, IEEE 802.11s, IEEE 802.11g, multi-hop communication, wireless access point (AP), device to device communication, cellular communication protocols, and Bluetooth (BT) communication protocols.

[0028] Thus, the testing system 300 may be configured to perform one or more functionalities which may include virtually testing an embedded system software. The testing system 300 may provide a plurality of hardware testing sub-systems 202 where each hardware testing sub-systems 202 may implement respective testing applications comprising its application code 312. The one or more hardware testing subsystems 202 may intercommunicate and further enables each of the plurality of hardware testing sub-systems to communicate with one or more simulation models provided by SIM stack 304 through a common data channel implemented by COMM stack 307. The testing system 300 may configure a hardware abstraction layer 310 of respective application under test, which may further be associated with the one or more hardware testing subsystems 202. The configuration by the testing system 300 may be achieved by compiling a source code associated with the respective application with a virtualized driver package provided in the virtual HAL 302. Finally, the one or more hardware testing subsystems 202 may be communicatively linked with the one or more simulation models provided by SIM stack 304 via the common data channel provided by COMM stack 307 and the virtual HAL 302.

[0029] In order to perform the above-discussed functionalities, the testing system 300 may include a processor (not shown), a memory (not shown), and an input/output device (not shown). The processor (not shown) may include suitable logic, circuitry, interfaces, and/or code that may be configured to link the one or more simulation models provided by SIM stack 304 to each of a plurality of hardware testing sub-systems 202 via a predefined interface and perform the testing of a embedded system software. The processor (not shown) may be implemented based on temporal and spatial processor technologies, which may be known to one ordinarily skilled in the art. Examples of implementations of the processor (not shown) may be a Graphics Processing Unit (GPU), a Reduced Instruction Set Computing (RISC) processor, an Application-Specific Integrated Circuit (ASIC) processor, a Complex Instruction Set Computing (CISC) processor, a microcontroller, Artificial Intelligence (AI) accelerator chips, a co-processor, a central processing unit (CPU), and/or a combination thereof. The memory (not shown) may include suitable logic, circuitry, and/or interfaces that may be configured to store instructions executable by the processor (not shown). The memory (not shown) may store instructions that, when executed by the processor (not shown), may cause the processor (not shown) to virtually test an embedded systems software. The memory (not shown) may be a non-volatile memory or a volatile memory. Examples of non-volatile memory may include, but are not limited to a flash memory, a Read-Only Memory (ROM), a Programmable ROM (PROM), Erasable PROM (EPROM), and Electrically EPROM (EEPROM) memory. Examples of volatile memory may include but are not limited to Dynamic Random-Access Memory (DRAM), and Static Random-Access memory (SRAM). The memory (not shown) may also store various data that may be captured, processed, and/or required by the system 300.

[0030] The testing system 300 may further include the input/output device (not shown). Examples may include, but are not limited to a display, keypad, microphone, etc. The I/O devices (not shown) may receive input from a user through a dashboard implementing the VT stack 306. The input/output device (not shown) may display an output of the simulation and the processor (not shown). In an embodiment, the simulation may provide an output of the testing being done and the parameters used to create the various simulations of the virtual HAL 302 and the simulation models by the SIM stack 304 through the VT stack 306.

[0031] FIG. 4 illustrates an architecture of the virtual HAL of **FIG. 3**, in accordance with an embodiment of the present disclosure. The virtual HAL 302 may include a set of functions which may be compiled with the original source code in order to routes the call from the application layer 308 to the COMM stack layer 307. The set of functions of the virtual HAL

302 may contain implementation that may eliminate all the platform specific codes and make the code module executable in the processing environment of the testing system 300, thus removing the requirement of implementing special hardware for the sub-systems 202. The virtual HAL 302 may also include code modules that pushes the application layer 308 calls to the physical hardware into a COMMS stack layer 307 to be sent to the one or more simulation models provided by the SIM stack 304.

[0032] In an embodiment, the COMM stack layer 307 may provide a failsafe mechanism to deliver/store a request to the sub-systems 202 or the one or more simulation models provided by the SIM stack 304. Accordingly, all the communications from the one or more simulation models provided by the SIM stack 304 are sent to the COMM stack layer 307 and then to the virtual HAL 302. Further, the calls/requests from the application layer 308 are sent to the one or more simulation models provided by the SIM stack 304 through the virtual HAL 302 and the COMM stack layer 307. The virtual HAL 302 may contain a peripheral interface layer 402, a message buffer 404, an Analog to Digital convertor (ADC) 406, a Digital Input/Output (DIO) 408, a protocol register 410, a GUID Partition Table (GPT) 412. In an embodiment, the message buffer 404 may save the messages received from the COMMS stack layer 307 and the messages related to the calls received from the application layer 308. The calls received from the application layer 308 may be received by the peripheral interface layer 402 and the ADC 406 may convert any analogue data received from the simulated models through the COMM stack layer 307. The protocols register 410 may store multiple protocols which may be utilized in the communication and data transfer in the virtual HAL 302. The GPT contains general purpose timer/counter implementation using generic timing functions used in the program. The DIO 408 may provide an interface to effectively relay digital signals.

[0033] FIG. 5 illustrates a detail block diagram of the simulation stack 304, in accordance with an embodiment of the present disclosure. The simulation stack 304 may include a simulation interface 502 that allows to connect various model instances 504a-n of simulation model to one or more peripheral communication channels. The simulation stack 304 enables to simulate entire physical environment 102 of an embedded system 100 and bind that to the application layer 308. The simulation stack 304 may enable a virtual integration testing of the sub-systems 202a-n with the actual application code of a newly designed sub-system 220. Thus, a simulated physical environment 305 may provide a virtual testing bench in order to test application code of a newly designed sub-system 220 and provide an error free code. Further, the simulation stack 304 may include model instances 504a-n which may include the one or more simulation models for example the MATLAB Simulink model, GNU Octave, NI

Multism, Scilab, Comsol Multiphysics, etc. Further, the model instances 504a-n, may help verify the codes corresponding to the application under test.

[0034] In an embodiment, the simulation stack 304 may provide a simulated physical environment which generate an input stimulus similar for equipment control for simulating the real time environment. The simulated physical environment may generate an input stimulus similar to that of a control signal from a complex DAQ system which may be done by utilizing specialized software/library. The simulation stack 304 may provide automation scripts that may configure and control the real time environment. The data source 510 may include data which may act as test output of a sensor or an actuator and may send data feed corresponding to the model instances 504a-n to the COMM stack 307 in order to address the application layer calls. In an embodiment, the simulation stack 304 may include a library of test cases, where the library of test cases may include a one or more simulation builds for different hardware and virtual protocols. The simulation stack 304 may further comprise of a memory simulator 314, a protocol simulator 322, a DIO simulator 508 and an ADC simulator 316. The memory simulator 314 may include the virtualized source code of a storage of a processing device, for example the RAM of a system, and may hence simulate the memory. Further, the DIO simulator 508 may include the virtualized source code from a plurality of peripherals which in accordance with the embodiment may be various sensors, actuators, etc. Similarly, the protocol simulator 322, may include virtualized codes of protocols which may provide with a predefined procedures based upon which the information sharing from the simulation stack 304 may be done. Some common examples of protocols may include TCP/IP (Transmission Control Protocol/Internet Protocol), HTTPS (Secure Hypertext Transmission Protocol), SMTP (Simple Mail Transfer Protocol), and DNS (Domain Name System), etc.

[0035] Thus, the simulated physical environment 305 solved a common problem encountered during the test execution phase that is the unavailability of the test system for test execution. The simulated physical environment 305 consists of only software components that can be easily replicated in a desktop environment or deployed in the cloud and puts no limitation on the number of test environments available for testing.

[0036] FIG. 6 illustrates functional modules of the testing system 300 enabling integration testing of an embedded system software, in accordance with an embodiment of the present disclosure. The testing system 300 may include a development module 602, a build generation module 604, a test execution module 606 and other module. In the development module 602 a software application may be in design phase where the developers prepare a final software code. In the build generation module 604 the final software code may be converted to

a binary/executable code for the target environment. In an embodiment, the target environment may be a physical hardware which may not be available or developed. Therefore, a virtual HAL 302 may be used to route the application layer calls to a simulation module that may be created from a library of virtualization tools in the VT stack 306. The simulation module is hardware independent and can run/execute on a normal PC. The drivers of the virtual HAL 302 may have functional implementation that may allow to route any hardware calls to a common communication interface the COMM stack 307. In the test execution module 606 various sub-systems 202 of a software may be integrated to generate a unified system. Accordingly, a system integration of virtual HAL 302 and simulated modules are used for testing the software components under development with no requirement of hardware components. The system integration by the test execution module 606 may further include a physical environment simulator, a virtual HAL, a test code which may be integrated with a mocked software code with its corresponding virtual HAL to perform the functional verification.

[0037] FIG. 7 illustrates a flow diagram 700 of a method for virtually testing a hardware system, in accordance with an embodiment of present disclosure. At step 702, a hardware testing system may be provided which may comprise a plurality of hardware testing sub-systems which may test the hardware system. In an embodiment, each hardware testing sub-system may implement a respective testing application.

[0038] At step 704, a common data channel may be established for the plurality of hardware testing sub-systems. In an embodiment, the common data channel may enable the plurality of hardware testing sub-systems to intercommunicate. In an embodiment, the common data channel may further enable each of the plurality of hardware testing sub-systems to communicate with one or more simulation models.

[0039] At step 706, a hardware abstraction layer (HAL) of a respective testing application associated with each of the plurality of hardware testing sub-systems may be configured by the hardware testing system. In an embodiment, the HAL may be configured by compiling a source code associated with the respective testing application with a virtualized driver package and where the virtualized driver package may be configured to route application layer calls to the common data channel.

[0040] At step 708, each of the plurality of hardware testing sub-systems may be communicatively linked with the one or more simulation models through the common data channel and the configured HAL by the hardware testing system.

[0041] It is intended that the disclosure and examples be considered as exemplary only, with a true scope of disclosed embodiments being indicated by the following claims.

WE CLAIM:

1. A method of virtually testing a hardware system, the method comprising:

providing a hardware testing system comprising a plurality of hardware testing sub-systems to test the hardware system, wherein each hardware testing sub-system implements a respective testing application;

establishing a common data channel for the plurality of hardware testing sub-systems, wherein the common data channel enables the plurality of hardware testing sub-systems to intercommunicate and further enables each of the plurality of hardware testing sub-systems to communicate with one or more simulation models;

configuring, by the hardware testing system, a hardware abstraction layer (HAL) of a respective testing application associated with each of the plurality of hardware testing sub-systems by compiling a source code associated with the respective testing application with a virtualized driver package, the virtualized driver package being configured to route application layer calls to the common data channel; and

communicatively linking, by the hardware testing system, each of the plurality of hardware testing sub-systems with the one or more simulation models, via the common data channel and the configured hardware abstraction layer (HAL), for testing the hardware system.

2. The method as claimed in claim 1, wherein each of one or more simulation models is communicatively linked to each of the plurality of hardware testing sub-systems via a respective predefined interface.

3. The method as claimed in claim 1, wherein the one or more simulation models comprises a MATLAB Simulink model.

4. The method as claimed in claim 1, further comprising:

generating a simulation build comprising a library of test cases, wherein the simulation build is hardware independent.

5. The method as claimed in claim 4, further comprising:

receiving a selection of one or more target test cases from the library of test cases;

executing the one or more target test cases, based on a test script, wherein executing comprises:

receiving virtual sensor readings from the one or more simulation models, via the common data channel; and

sending actuation signals to the one or more simulation models, in response to the virtual sensor readings, via the common data channel.

6. A system of virtually testing a hardware system, comprising:

one or more processors in a hardware testing system;

a memory communicably connected to the one or more processors storing a plurality of processor executable instructions which upon execution cause the one or more processors to:

establish, by the hardware testing system comprising a plurality of hardware testing sub-systems and each hardware testing sub-system implementing a respective testing application, a common data channel for the plurality of hardware testing sub-systems, wherein the common data channel enables the plurality of hardware testing sub-systems to intercommunicate and further enables each of the plurality of hardware testing sub-systems to communicate with one or more simulation models;

configure, by the hardware testing system, a hardware abstraction layer (HAL) of a respective testing application associated with each of the plurality of hardware testing sub-systems by compiling a source code associated with the respective testing application with a virtualized driver package, the virtualized driver package being configured to route application layer calls to the common data channel; and

communicatively link, by the hardware testing system, each of the plurality of hardware testing sub-systems with the one or more simulation models, via the common data channel and the configured hardware abstraction layer (HAL), for testing the hardware system.

7. The system as claimed in claim 6, wherein each of one or more simulation models is communicatively linked to each of the plurality of hardware testing sub-systems via a respective predefined interface.

8. The system as claimed in claim 6, wherein the one or more simulation models comprises a MATLAB Simulink model.

9. The system as claimed in claim 6, wherein the one or more processors are configured to generate a simulation build comprising a library of test cases, wherein the simulation build is hardware independent.

10. The system as claimed in claim 9, wherein the one or more processors are configured to:

receive a selection of one or more target test cases from the library of test cases;

execute the one or more target test cases, based on a test script, wherein to execute the one or more target cases, the one or more processors are further configured to:

receive virtual sensor readings from the one or more simulation models, via the common data channel; and

send actuation signals to the one or more simulation models, in response to the virtual sensor readings, via the common data channel.

Dated this 16th day of September 2022

-- Digitally Signed--
Bhanu Prasad
(INPA No: **3253**)
Manager, IPR Dept.,
L&T Technology Services Limited,
DLF 3rd Block, 2nd Floor,
Manapakkam, Chennai - 600089.

ABSTRACT

METHOD AND SYSTEM FOR VIRTUALLY TESTING A HARDWARE SYSTEM

A method and a system of virtually testing an embedded hardware system is disclosed. The hardware system is tested through a hardware testing system comprising a plurality of hardware testing sub-systems, each implementing a respective testing application. A common data channel is established for the plurality of hardware testing sub-systems to intercommunicate and to communicate with one or more simulation models. A hardware abstraction layer (HAL) of a respective testing application associated with each of the plurality of hardware testing sub-systems is configured by compiling a source code associated with the respective testing application with a virtualized driver package. The virtualized driver package may route application layer calls to the common data channel. Each of the plurality of hardware testing sub-systems are communicatively linked with the one or more simulation models via the common data channel and the configured HAL, for testing the hardware system.

[To be published with FIG. 2]

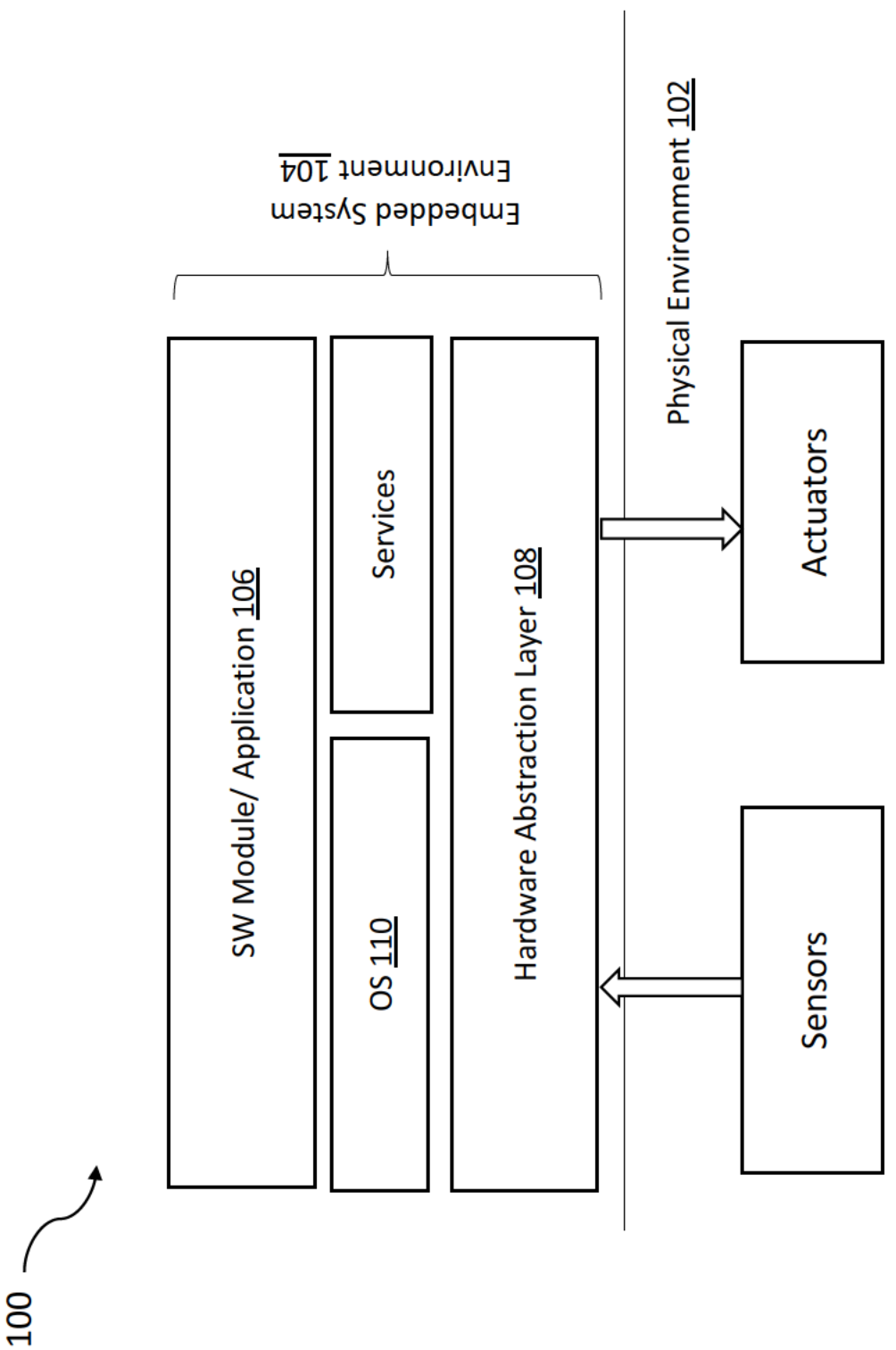

FIG. 1 (PRIOR ART)

-- Digitally Signed--
Bhanu Prasad
(INPA No: 3253)
Manager, IPR Dept.,
L&T Technology Services Limited,
DLF 3rd Block, 2nd Floor,
Manapakkam, Chennai - 600089.

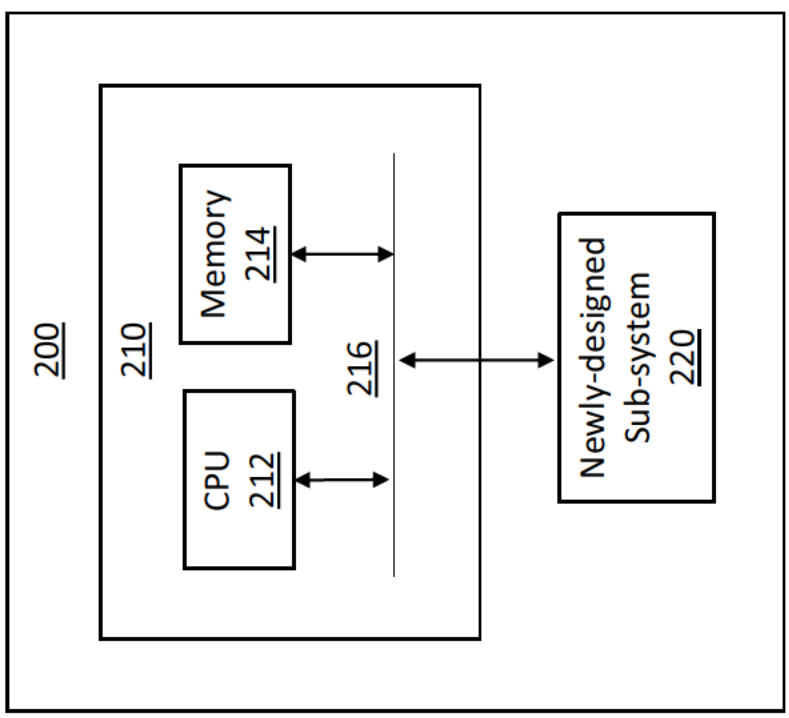

FIG. 2

-- Digitally Signed--
Bhanu Prasad
(INPA No: 3253)
Manager, IPR Dept.,
L&T Technology Services Limited,
DLF 3rd Block, 2nd Floor,
Manapakkam, Chennai - 600089.

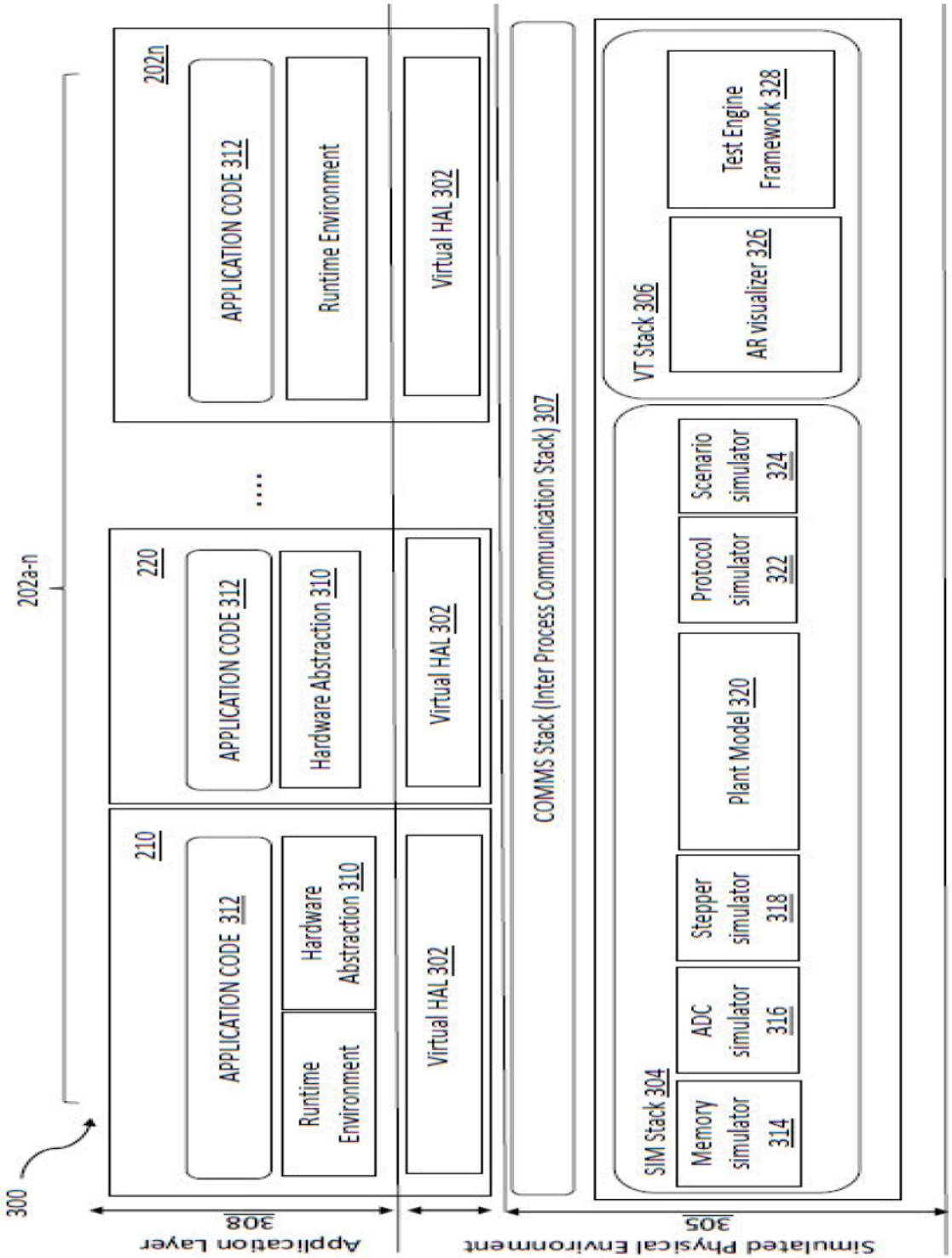

FIG 3

-- Digitally Signed--
Bhanu Prasad
(INPA No: 3253)
Manager, IPR Dept.,
L&T Technology Services Limited,
DLF 3rd Block, 2nd Floor,
Manapakkam, Chennai - 600089.

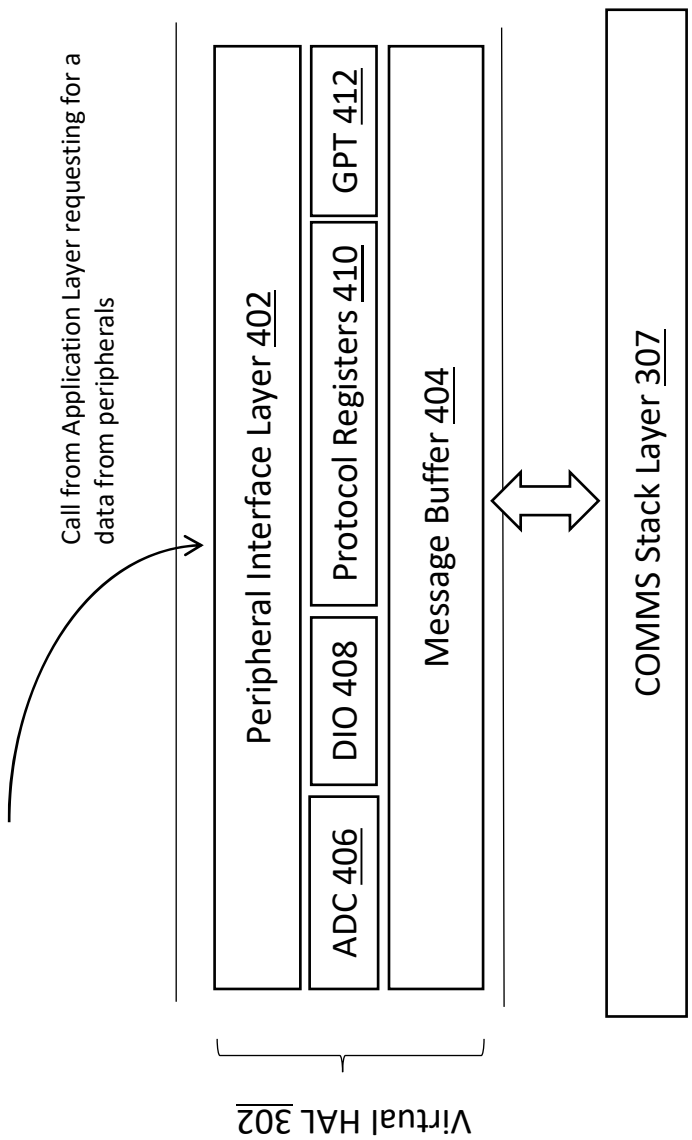

FIG. 4

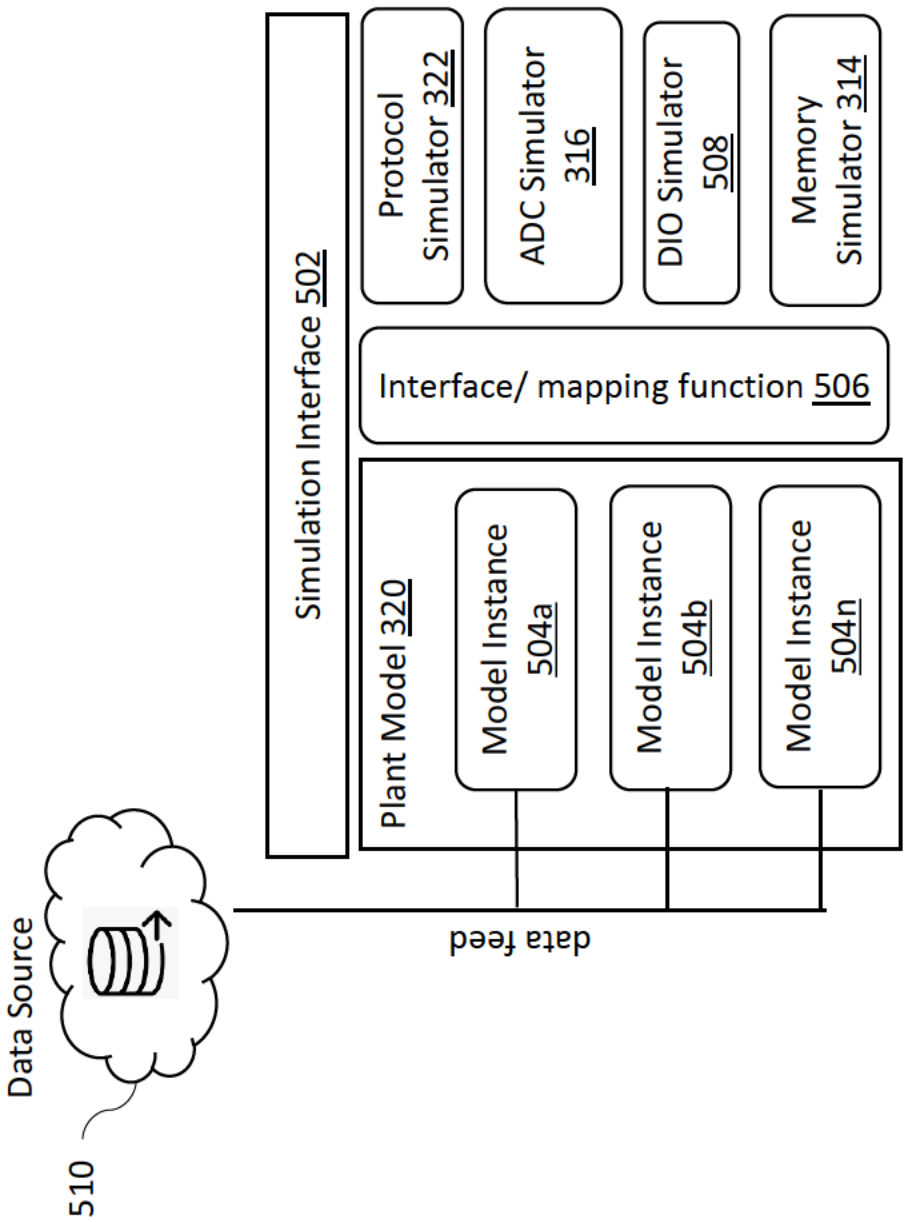

FIG. 5

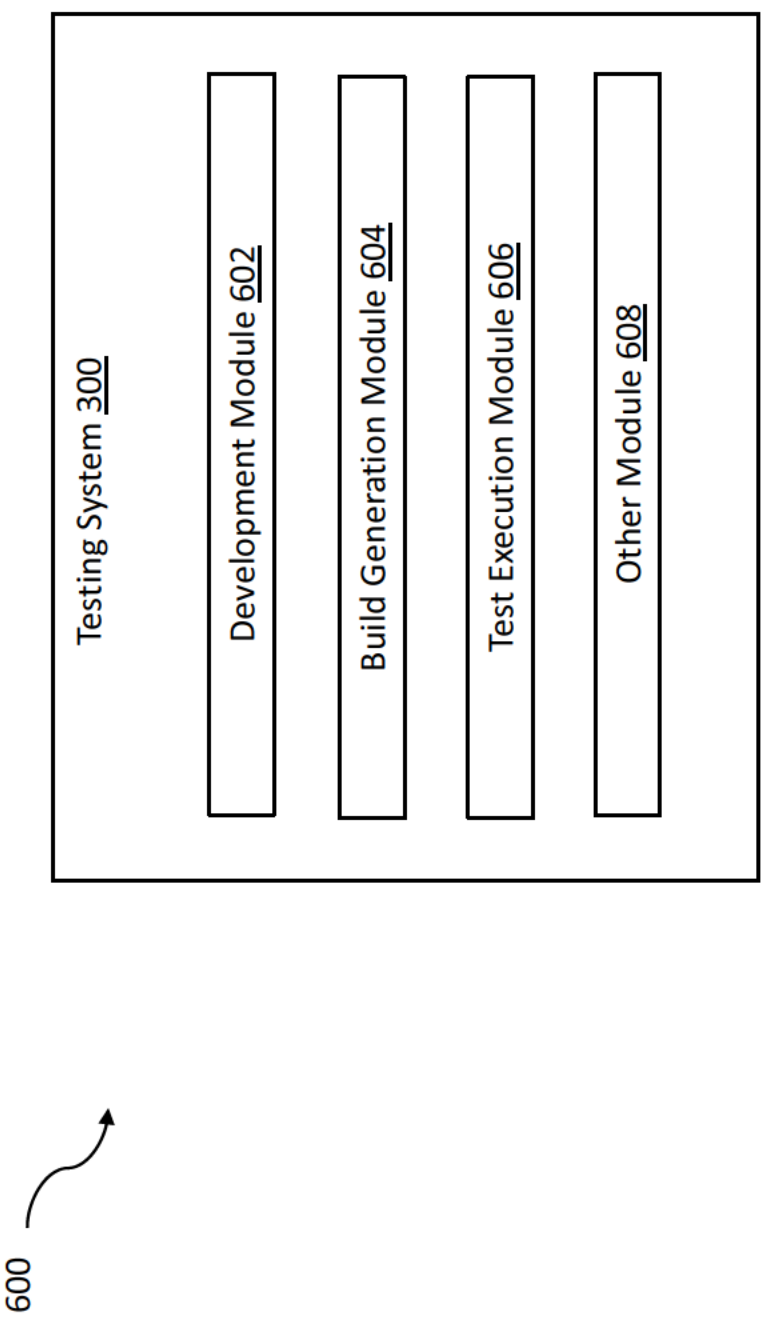

FIG. 6

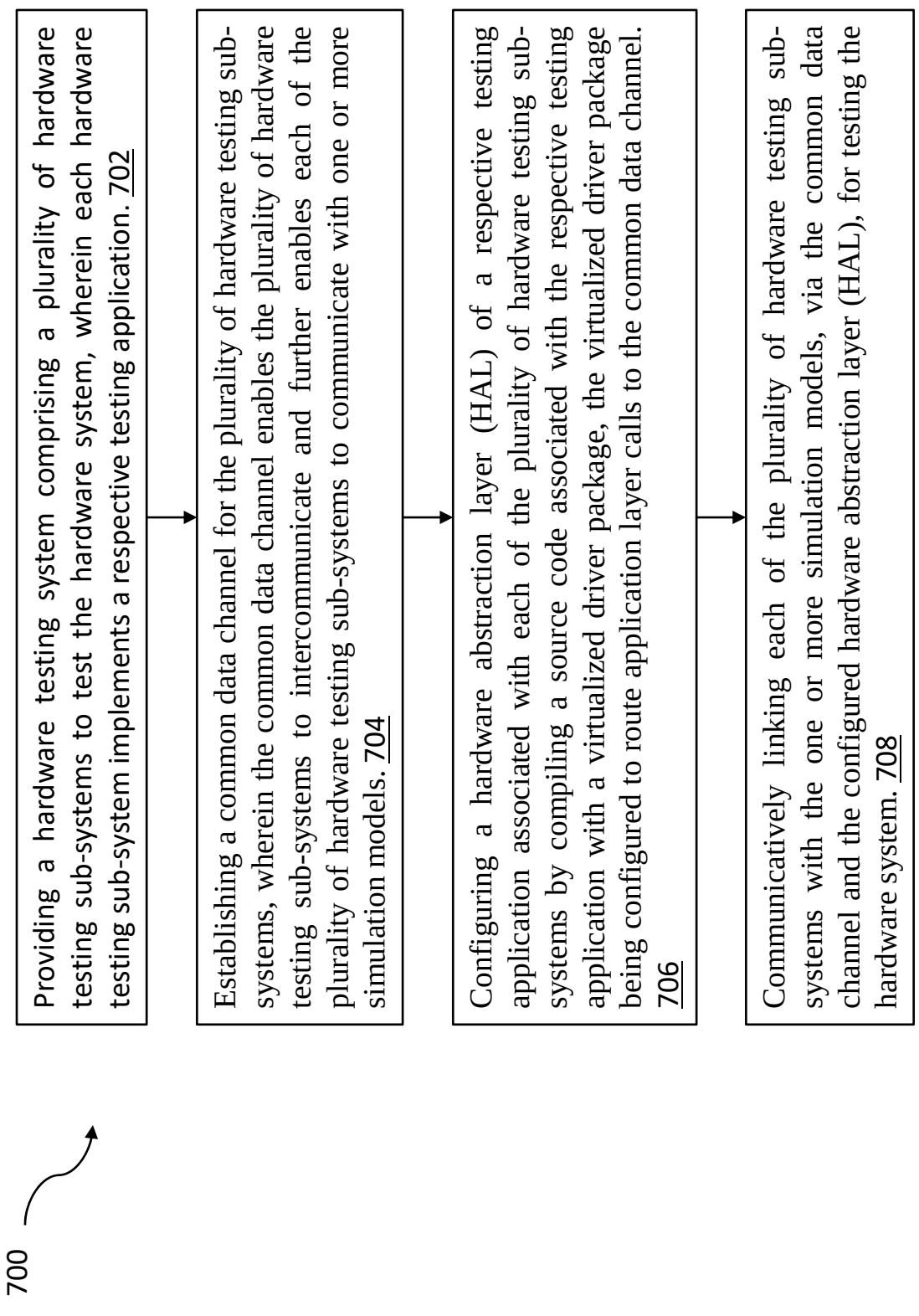

FIG. 7