1. A test instrument comprising:

a first processing system that is programmable to run one or more test programs to test a device interfaced to the test instrument, and that is programmed to control operation of the test instrument;

a second processing system that is dedicated to device testing, the second processing system being programmable to run one or more test programs to test the device; and

programmable logic configured to act as an interface between the test instrument and the device, the programmable logic being configurable to perform one or more tests on the device;

wherein the first processing system and the second processing system are programmable to access the device via the programmable logic.

2. The test instrument of claim 1, wherein the first processing system has a first testing latency, the second processing system has a second testing latency, and the programmable logic has a third testing latency, the first testing latency being greater than the second testing latency, and the second testing latency being greater than the third testing latency.

3. The test instrument of claim 2, wherein the first testing latency is on the order of milliseconds, the second testing latency is on the order of microseconds, and the third testing latency is on the order of nanoseconds.

4. The test instrument of claim 1, wherein the first processing system is programmed to run one or more test programs to test the device interfaced to the test instrument,

wherein the second processing system is not programmed to run one or more test programs to test the device; and

wherein the programmable logic configured is not configured to perform one or more tests on the device.

5. The test instrument of claim 1, wherein the first processing system is not programmed to run one or more test programs to test the device interfaced to the test instrument,

wherein the second processing system is programmed to run one or more test programs to test the device; and

wherein the programmable logic is not configured to perform one or more tests on the device.

6. The test instrument of claim 1, wherein the first processing system is not programmed to run one or more test programs to test the device interfaced to the test instrument,

wherein the second processing system is not programmed to run one or more test programs to test the device; and

wherein the programmable logic is configured to perform one or more tests on the device.

7. The test instrument of claim 1, wherein the first processing system comprises a processing device that executes a windowing operating system;

wherein the second processing system comprises one or more processing devices, each of the one or more processing devices corresponding to a different device to be tested by the test instrument; and

wherein the programmable logic comprises one or more field programmable gate arrays (FPGAs), each of the one or more FPGAs corresponding to a different device to be tested by the test instrument.

8. The test instrument of claim 1, wherein the programmable logic comprises field programmable gate arrays (FPGAs), at least one of the FPGAs being programmable logic being configurable to perform one or more tests on the device, and at least one of the FPGAs being pre-programmed to perform functions that do not involve exchange of data with the device to be tested.

9. The test instrument of claim 1, wherein at least one of the first processing system, the second processing system, and the programmable logic is reprogrammable via one or more interfaces.

10. The test instrument of claim 1, wherein controlling operation of the test instrument comprises one or more of the following: exchanging communications between the test instrument and one or more entities over a network, scanning the test instrument for malware, and performing memory management functions.

11. A test instrument comprising:

a first tier system for interacting with an environment external to the test instrument, the first tier system being programmable to perform testing operations on a device;

a second tier system that is programmable to perform testing operations on the device; and

a third tier system for interfacing to the device, the third tier system being configurable to perform testing operations on the device, the first tier system and the second tier system being programmed to access the device through the third tier system.

12. The test instrument of claim 11, wherein the first tier system has a first testing latency, the second tier system has a second testing latency, and the third tier system has a third testing latency, the first testing latency being greater than the second testing latency, and the second testing latency being greater than the third testing latency.

13. The test instrument of claim 12, wherein the first testing latency is on the order of milliseconds, the second testing latency is on the order of microseconds, and the third testing latency is on the order of nanoseconds.

14. The test instrument of claim 11, wherein the first tier system is programmed to run one or more test programs to test the device interfaced to the test instrument,

wherein the second tier system is not programmed to run one or more test programs to test the device; and

wherein third tier system configured is not configured to perform one or more tests on the device.

15. The test instrument of claim 11, wherein the first tier system is not programmed to run one or more test programs to test the device interfaced to the test instrument,

wherein the second tier system is programmed to run one or more test programs to test the device; and

wherein third tier system is not configured to perform one or more tests on the device.

16. The test instrument of claim 11, wherein the first tier system is not programmed to run one or more test programs to test the device interfaced to the test instrument,

wherein the second tier system is not programmed to run one or more test programs to test the device; and

wherein the third tier system is configured to perform one or more tests on the device.

17. The test instrument of claim 11, wherein the first tier system comprises a processing device that executes a windowing operating system;

wherein the second tier system comprises one or more processing devices, each of the one or more processing devices corresponding to a different device to be tested by the test instrument; and

wherein the third tier system comprises one or more field programmable gate arrays (FPGAs), each of the one or more FPGAs corresponding to a different device to be tested by the test instrument.

18. The test instrument of claim 11, wherein the third tier system comprises field programmable gate arrays (FPGAs), at least one of the FPGAs being programmable logic being configurable to perform one or more tests on the device, and at least one of the FPGAs being pre-programmed to perform functions that do not involve exchange of data with the device to be tested.

19. The test instrument of claim 11, wherein at least one of the first tier system, the second tier system, and the third tier system is reprogrammable via one or more interfaces.

20. The test instrument of claim 11, wherein controlling operation of the test instrument comprises one or more of the following: exchanging communications between the test instrument and one or more entities over a network, scanning the test instrument for malware, and performing memory management functions.

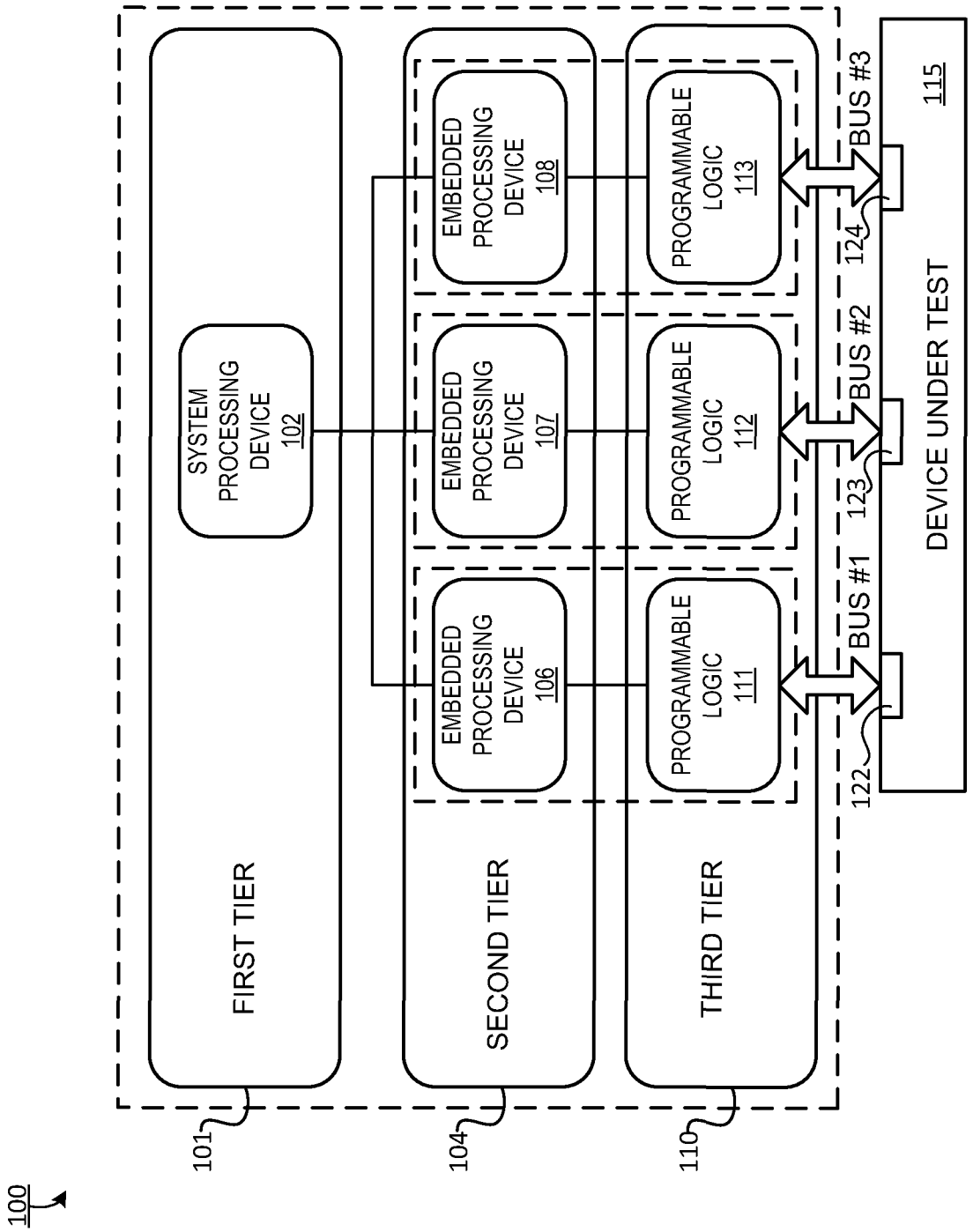
100
101
104
110
FIRST TIER
SECOND TIER
THIRD TIER
SYSTEM PROCESSING DEVICE 102
EMBEDDED PROCESSING DEVICE 106
EMBEDDED PROCESSING DEVICE 107
EMBEDDED PROCESSING DEVICE 108
PROGRAMMABLE LOGIC 111
PROGRAMMABLE LOGIC 112
PROGRAMMABLE LOGIC 113
BUS #1
BUS #2
BUS #3
122
123
124
DEVICE UNDER TEST
115

FIG. 1

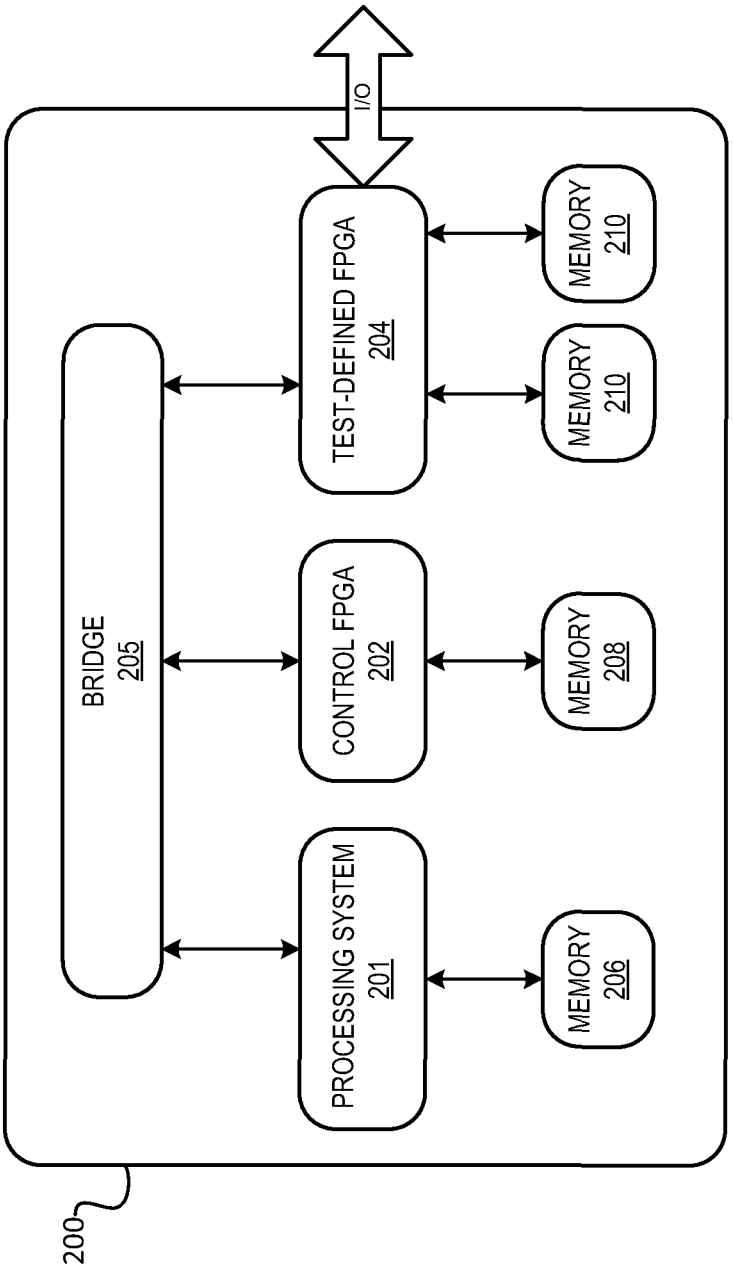
200
I/O
BRIDGE
205
PROCESSING SYSTEM
201
CONTROL FPGA
202
TEST-DEFINED FPGA
204
MEMORY
206
MEMORY
208
MEMORY
210
MEMORY
210

FIG. 2

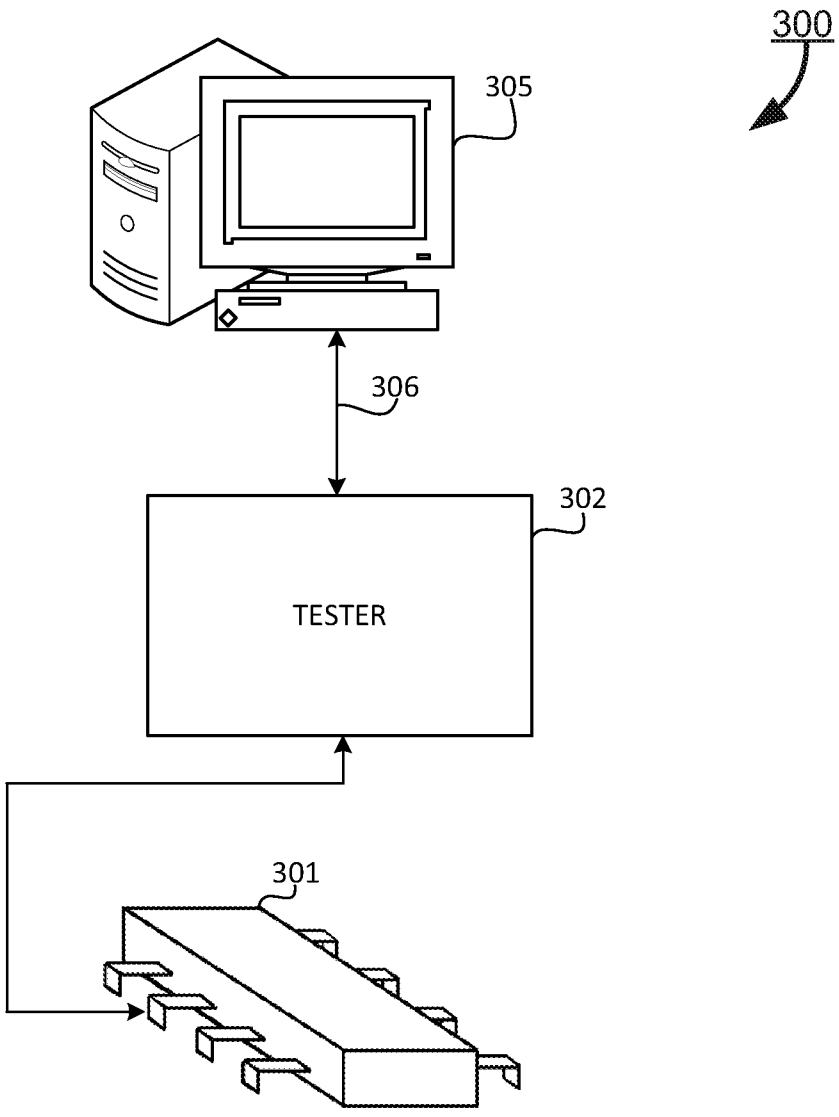

FIG. 3

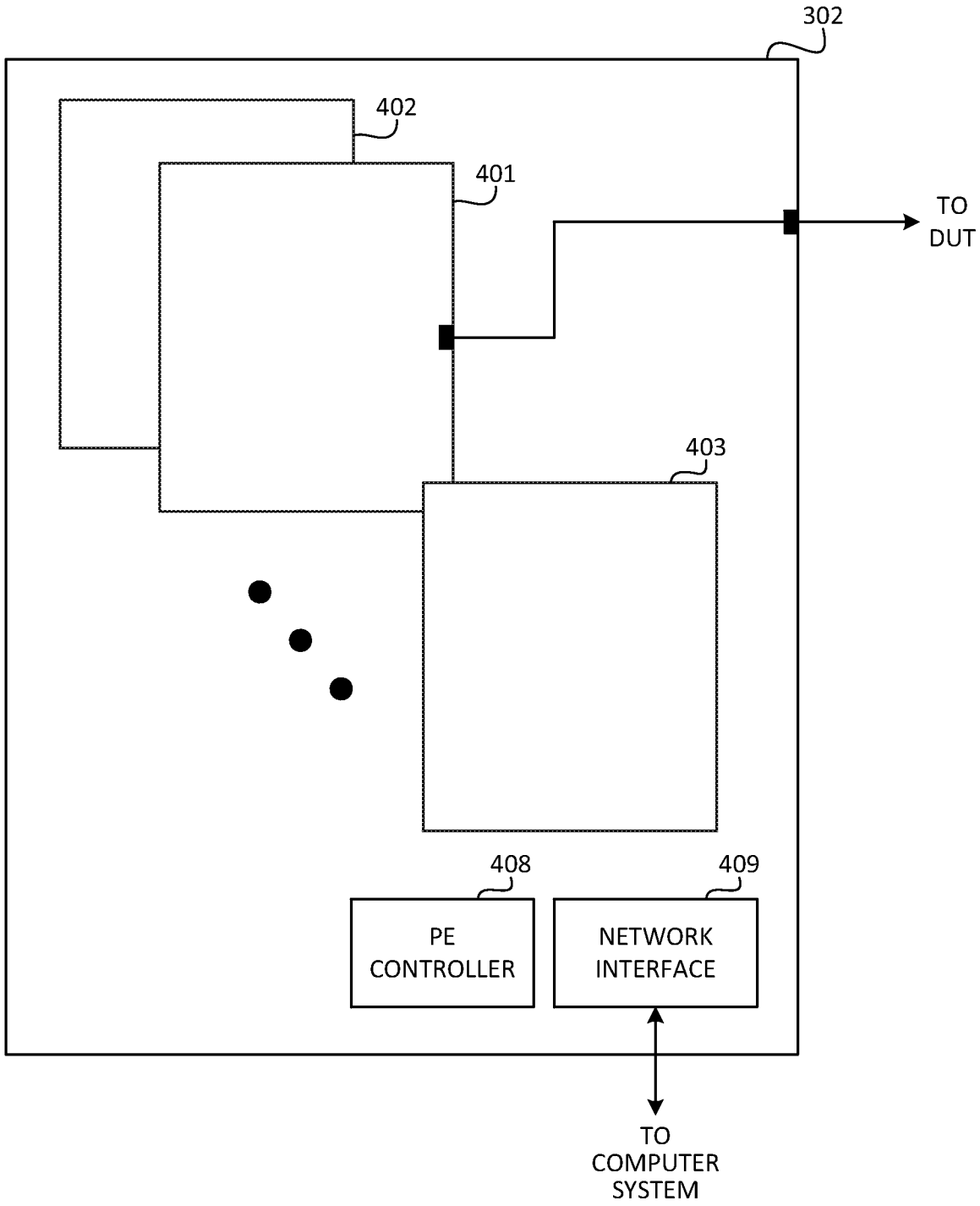
302
402
401
TO
DUT
403
408
409
PE
CONTROLLER
NETWORK
INTERFACE
TO
COMPUTER
SYSTEM

FIG. 4

PROGRAMMABLE TEST INSTRUMENT

TECHNICAL FIELD

This disclosure relates generally to a programmable test instrument

BACKGROUND

Automatic test equipment (ATE) plays a role in the manufacture of electronics, such as semiconductor devices and circuit board assemblies. Manufacturers generally use automatic test equipment, or "tester instruments", to verify the operation of devices during the manufacturing process. Such devices are referred to as a "device under test" (DUT) or a "unit under test" (UUT). Early detection of faults eliminates costs that would otherwise be incurred by processing defective devices, and thus reduces the overall costs of manufacturing. Manufacturers also use ATE to grade various specifications. Devices can be tested and binned according to different levels of performance in areas, such as speed. Devices can be labeled and sold according to their actual levels of performance.

SUMMARY

In general, in one aspect, a test instrument includes a first processing system that is programmable to run one or more test programs to test a device interfaced to the test instrument, and that is programmed to control operation of the test instrument, a second processing system that is dedicated to device testing, the

second processing system being programmable to run one or more test programs to test the device, and programmable logic configured to act as an interface between the test instrument and the device, the programmable logic being configurable to perform one or more tests on the device. The first processing system and the second processing system are programmable to access the device via the programmable logic.

In general, in another aspect, a test instrument includes a first tier system for interacting with an environment external to the test instrument, the first tier system being programmable to perform testing operations on a device, a second tier system that is programmable to perform testing operations on the device, and a third tier system for interfacing to the device, the third tier system being configurable to perform testing operations on the device. The first tier system and the second tier system are programmed to access the device through the third tier system.

Aspects may include one or more of the following features. The first processing system has a first testing latency, the second processing system has a second testing latency, and the programmable logic has a third testing latency, the first testing latency being greater than the second testing latency, and the second testing latency being greater than the third testing latency. The first testing latency is on the order of milliseconds, the second testing latency is on the order of microseconds, and the third testing latency is on the order of nanoseconds. The first processing system is programmed to run one or more test programs to test the

device interfaced to the test instrument, the second processing system is not programmed to run one or more test programs to test the device, and the programmable logic configured is not configured to perform one or more tests on the device.

The first processing system is not programmed to run one or more test programs to test the device interfaced to the test instrument, the second processing system is programmed to run one or more test programs to test the device, and the programmable logic is not configured to perform one or more tests on the device. The first processing system is not programmed to run one or more test programs to test the device interfaced to the test instrument, the second processing system is not programmed to run one or more test programs to test the device, and the programmable logic is configured to perform one or more tests on the device. The first processing system includes a processing device that executes a windowing operating system, the second processing system includes one or more processing devices, each of the one or more processing devices corresponding to a different device to be tested by the test instrument, and the programmable logic includes one or more field programmable gate arrays (FPGAs), each of the one or more FPGAs corresponding to a different device to be tested by the test instrument.

The programmable logic includes field programmable gate arrays (FPGAs), at least one of the FPGAs being programmable logic being configurable to perform one or more tests on the device, and at least one of the FPGAs being pre-programmed to perform functions that do not involve exchange of data with the device to be

tested. At least one of the first processing system, the second processing system, and the programmable logic is reprogrammable via one or more interfaces. Controlling operation of the test instrument includes one or more of the following: exchanging communications between the test instrument and one or more entities over a network, scanning the test instrument for malware, and performing memory management functions.

Two or more of the features described in this disclosure, including this summary section, may be combined to form embodiments not specifically described herein.

The systems and techniques described herein, or portions thereof, may be implemented as a computer program product that includes instructions that are stored on one or more non-transitory machine-readable storage media, and that are executable on one or more processing devices. The systems and techniques described herein, or portions thereof, may be implemented as an apparatus, method, or electronic system that may include one or more processing devices and memory to store executable instructions to implement the stated functions.

The details of one or more implementations are set forth in the accompanying drawings and the description below. Other features, objects, and advantages will be apparent from the description and drawings, and from the claims.

DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram of an example test instrument.

Fig. 2 is a block diagram of showing examples of features that may be incorporated into the example test instrument of Fig. 1

Fig. 3 is a block diagram of an example test system.

Fig. 4 is a block diagram of an example tester included in the test system.

DETAILED DESCRIPTION

Described herein is a test instrument having a multi-tiered architecture. For example, the architecture may include a first tier processing system that interacts with an environment external to the test instrument, and that is programmable to perform testing operations on a unit under test (UUT). The architecture may also include a second tier processing system that is programmable to perform testing operations on the UUT, and a third tier processing system that interfaces to the UUT and that is also configurable to perform testing operations on the DUT. The architecture may also be configured so that the first tier processing system and the second tier processing system access the device through the third tier system.

Fig. 1 is a block diagram of an example implementation of the foregoing test instrument 100. In Fig. 1, test instrument 100 includes a three-tiered processing system. However, in other example implementations, there may be more, or less, tiers. The different tiers of test instrument 100 reflect the relative relationship of the tiers to the DUT. In this example, first tier 101 includes a computer 102. Computer 102 controls various features of test instrument 100, such as communication with an external network. In addition, computer 102 is programmable to perform various

testing operations, as described below. Second tier 104 includes one or more processing devices 106 to 108 that are dedicated to testing. For example, processing devices 106 to 108 typically do not perform non-test functions like test instrument control and network communication; however, the processing devices 106 to 108 may perform tasks such as communication and flow of control, interrupts, and timing. Third tier 110 includes logic 111 to 113 that is programmable both to act as an interface to a DUT 115 and to perform one or more test operations on the DUT.

In this example first tier 101, computer 102 includes one or more processing devices, such as one or more microprocessors or a single multi-core microprocessor (not shown). Computer 102 also includes memory (not shown) that stores executable code to control test instrument communication with the external environment, and to perform various "housekeeping" functions to control operation of test instrument 100. For example, computer 102 may be responsible for exchanging communications between the test instrument and one or more external entities over a network interface 120, scanning the test instrument for malware, memory management, power control, and other functions that are not specifically related to testing the DUT.

Computer 102 is also programmable to perform test operations on a DUT (e.g., 115) interfaced to test instrument 100. The test operations may include, but are not limited to, testing bus speed, reaction time, or any other appropriate operational aspects of the DUT. In general, the testing that is performed is

dependent upon the type of device being tested, and the information sought during testing.

One or more test programs may be loaded into memory on computer 102, and executed by processing device(s) in computer 102 in order to perform the testing. While performing testing, computer 102 may continue to perform other functions, such as those described above, to keep test instrument 100 operational. Consequently, the test latency (e.g., the amount of time between the start of a test and receipt of test results) can be on the order of milliseconds. This is but an example of the test latency. In different systems, numerous factors may have an effect on test latency, such as the speed of the processing device(s) in computer 102, the amount of memory available in computer 102 to run the test programs, and so forth.

A possible advantage of performing testing via computer 102 relates to development costs of test programs. More specifically, computer 102 may run a an OS like Windows, or other relatively user-friendly, operating system. Tools available for development of test programs on such an operating system are typically widely available, and generally well known to test program developers. As a result, the cost of developing test programs on computer 102, to run on computer 102, can be less than the cost of developing test programs to run on the other tiers of the multi-tiered architecture. This generalization, however, may not apply in all cases.

In this example, second tier 104 includes multiple embedded processing devices 106 to 108. Here, three embedded processing devices are shown;

however, test instrument 100 may include any appropriate number of embedded processing devices, e.g., one, two, four, five or more. These processing devices are embedded in the sense that they are incorporated into test instrument 100; and are dedicated to performing test functions (e.g., to testing DUTs interfaced to test instrument 100). Embedded processing devices 106 to 108 typically are not responsible for test instrument operations like the "housekeeping" operations described above that are performed by computer 102. However, in some implementations, embedded processing devices 106 to 108 may be programmed to perform one or more such operations, or other operations not specifically directed to testing DUTs.

Each embedded processing device 106 to 108 may include, e.g., a microcontroller or a microprocessor having a single core or multiple cores. Each microprocessor is programmable, either directly or via computer 102. For example, a user of test instrument 100 may interact with the operating system of computer 102 to program an embedded processing device 106. Alternatively, there may a direct interface, e.g., hardware or software, through which each embedded processing device may be programmed. Programming, in this context, refers to storing one or more test programs onto a respective embedded processing device, which can be executed on that embedded processing device to test a DUT.

As shown in Fig. 1, each embedded processing device is interfaced to computer 102 and to respective programmable logic (in this example, a field programmable gate array (FPGA)). As explained below, each FPGA acts as an

interface to a separate DUT (not shown) or to a portion of a single DUT (e.g., a bus 122, 123, 124 on that DUT, as shown) for testing. Accordingly, in this example, each embedded processing device may be programmed with a test program designed specifically for the corresponding DUT, or portion thereof being tested. As noted, an appropriate test program may be loaded directly into the embedded processing device or it may be loaded via computer 102. Each embedded processing device may execute its own test program separately, and concurrently with, other embedded processing devices. In some implementations, there may be coordination among the embedded processing devices as to how their respective test programs are to be executed. Such coordination maybe implemented by the embedded processing device themselves or by computer 102. In some implementations, the coordination may involve devices at different tiers of the architecture. In some implementations, the different embedded processing devices 106 to 108 may implement different portions (e.g., modules) of the same test program, with or without appropriate coordination.

A possible advantage of performing testing via an embedded processing device relates to test latency. More specifically, because the embedded processing devices are dedicated to testing, their resources are not typically taxed by other tasks. As a result, testing latency can be less than that achieved by computer 102. For example, test latency for an embedded processing device can be on the order of microseconds. This, however, is but an example of embedded processing device test latency. In different systems, numerous factors may have an effect on test

latency, such as processing device speed, the amount of memory available to run the test programs, and so forth. Accordingly, the foregoing generalization may not apply in all cases.

Furthermore, tools are available for development of test programs on the embedded processing devices. As a result, the cost of developing test programs on an embedded processing device, to run on an embedded processing device, can be less than the cost of developing test programs to run on hardware, such as an FPGA.

Third tier 110 includes programmable logic, e.g., FPGAs 111 to 113, although other types of programmable logic may be used in lieu of FPGAs. Each FPGA is configured by loading a program image into the FPGA. This program image is referred to as an "FPGA load". In this example, each FPGA is configured to act as an interface between a DUT, or portion thereof (e.g., a DUT bus) and test instrument 100. For example, an FPGA may specify a port width, port speed (e.g., 10MHz to 400MHz), the number of input ports, the number of output ports, and so forth.

First tier 101 computing device(s) (e.g., computer 102) and second tier 104 computing device(s) (e.g., embedded processing devices 106 to 108) access DUT 115 through third tier 110. For example, as shown in Fig. 1, each embedded processing device may communicate with DUT 115 through a corresponding FPGA. Computer 102 may communicate with DUT 115 through one or more FPGAs, depending upon which DUT, or portion of a DUT, is being currently tested. In some implementations, each interface implemented by an FPGA is programmable. In

other implementations, the interface implemented by each FPGA is static (e.g., not programmable).

Each FPGA may also be configurable to perform one or more tests on a corresponding DUT, or portion thereof to which the FPGA is interfaced. For example, the FPGA load for each FPGA may include one or more test routines that are run by the FPGA to test various aspects of the DUT. As above, the routines that are implemented depend upon the device being tested, and the information sought during testing. Test routines run by each FPGA may be run independently of other test routines run by other FPGAs, or there may be coordination among the various FPGAs. Each FPGA may execute its own test routine separately, and concurrently with, other embedded processing devices. In some implementations, there may be coordination among the FPGAs as to how their respective test programs are to be executed. Such coordination maybe implemented by the FPGAs themselves, by their corresponding embedded processing devices, or by computer 102. In some implementations, the coordination may involve devices at different tiers of the architecture. For example, computer 102, in concert with embedded processing devices 106 to 108, may coordinate operation of respective FPGAs 111 to 113. In some implementations, the different FPGAs may implement different portions (e.g., modules) of the same test routine, with or without appropriate coordination.

A possible advantage of performing testing via an FPGA relates to test latency. More specifically, because the FPGAs are hardware devices, they are able to run at higher speeds than the test routines programmed into either the embedded

processing devices 106 to 108 or computer 102. As a result, testing latency can be less than that achieved by embedded processing devices 106 to 108 or computer 102. For example, test latency for an Programmable device can be on the order of nanoseconds. This, however, is but an example of FPGA test latency. In different systems, numerous factors may have an effect on test latency. Accordingly, the foregoing generalization may not apply in all cases.

In some implementations, testing may be performed exclusively by one tier or another of the architecture. For example, computer 102 may be programmed to run one or more test programs to test a DUT, while devices on other tiers of the architecture do not perform DUT tests. Embedded processing devices 106 to 108 may be programmed to run one or more test programs to test a DUT, while devices on other tiers of the architecture do not perform DUT tests. FPGAs 111 to 113 may be configured to run one or more tests on the device, while devices on other tiers of the architecture do not perform DUT tests. Devices that are not performing tests are not necessarily dormant during this time. For example, computer 102 may continue to perform the housekeeping operations described above; the FPGAs may continue to route data to/from the DUT (i.e., to act as interfaces to the DUT); and the embedded processing devices may continue be active in coordination or other communication (e.g., transmitting test results from the FPGAs to computer 102).

In other implementations, testing may be performed by different tiers of the architecture concurrently or in concert. For example, two or more of computer 102, embedded processing devices 106 to 108, and FPGAs 111 to 113 may act in

coordination, at the same time or within the same test sequence, to perform one or more test operations on a single DUT or on multiple DUTs. To effect such coordination, appropriate programming is loaded into computer 102 and/or embedded processing devices 106 to 108, and/or an appropriate image is loaded into the FPGAs. By way of example, a first test may be performed on a DUT by computer 102; a second test may be performed on the DUT by embedded processing device 106; and a third test may be performed on the DUT by FPGA 111. The first, second and third tests may be separate tests, or part of the same test sequence. Data from the first, second and third tests may be combined, e.g., in computer 102, and processed to obtain the appropriate test results. These test results may be sent to an external computer (not shown) for analysis and reporting. Any of tier of the architecture or another (e.g., third party) party computer (not shown) may perform the coordination.

In implementations where one or more tiers of the architecture have not been programmed, the unprogrammed tiers may be bypassed (at least as far as their test functionality is concerned). The unprogrammed tiers may be pre-programmed or pre-configured to perform various functions, such as those described above relating to programming and communication among the tiers and with an external network.

Devices at the various tiers may be programmed or configured in real-time. In this context, "real-time" includes programming at test time or shortly before test time. That is, the test instrument need not come pre-programmed with test programs that are to be run on a DUT. Those test programs may be incorporated

into the instrument at the appropriate time. Existing test programs on the test instrument may likewise be replaced with new test programs, as appropriate.

Fig. 2 shows another example implementation of a test instrument 200 having a multi-tiered architecture. In the example of Fig. 2, test instrument 200 includes a processing system 201, a control FPGA 202, and a test-defined FPGA 204.

Processing system 201 may be a computer, such as computer 102; an embedded processing device, such as embedded processing system 106 to 108; or a two-tiered processing system, such as tiers 101 and 1-4.

Control FPGA 202 may be a dedicated FPGA that is configured to perform various housekeeping functions that are not within the purview, e.g., of a computer, such as computer 102. For example, those functions may include reading memory, determining die temperatures, and regulating power in the test instrument. In this implementation, control FPGA 202 is not configurable; however, it may be configurable in other implementations.

Test-defined FPGA 204 may be a configurable FPGA, such as FPGAs 111 to 113 of Fig. 1. More specifically, test-defined FPGA 204 may be configurable to perform one or more tests on a corresponding DUT, or portion thereof to which the test-defined FPGA is interfaced. For example, the FPGA load for each test-defined FPGA may include one or more test routines that are run by the test-defined FPGA to test various aspects of the DUT. As above, the routines that are implemented depend upon the device being tested, and the information sought during testing. Test routines run by each test-defined FPGA may be run independently of other test

routines run by other test-defined FPGAs, or there may be coordination among test-defined FPGA 204 and other test-defined FPGAs (not shown) that are part of the test instrument. The types of coordination among test-defined FPGAs, embedded processing devices, and a computer are similar to those described above with respect to Fig. 1.

In the example of Fig. 2, control FGPA 202 and test-defined FPGA 204 are separate devices. In other implementations, their functionalities can be combined into a single, programmable FPGA.

Fig. 2 also shows a bridge 205. Bridge 205 may include one or more buses and other appropriate electronics for transmitting communications among the various devices included in test instrument 200.

As shown in Fig. 2, processing system 201 is associated with memory 206; control FPGA 202 is associated with memory 208; and test-defined FPGA 204 is associated with memory 210. Each such memory may be used for storing test data and/or test programs, as well as executing test programs. In this example implementation, each memory is dedicated to its corresponding device. However, control FPGA 202 may provide a path, through which test-defined FPGA 204 (or another system processing device) may access, and use, its corresponding memory.

Referring now to Fig. 3, that figure shows an example of a system on which the architecture may be implemented. Fig. 3 shows an example test system 300 for testing a device-under-test (DUT) 301. Test system 300 includes a tester 302, which may have the multi-tiered architecture of Figs. 1 or 2. To interact with tester

302, system 300 includes a computer system 305 that interfaces with tester 302 over a network connection 306. As noted below, computer system 305 may incorporate the functionality of computer 102 (Fig. 1) or it may be an external computer that interacts with computer 102 on the test instrument. Typically, computer system 305 sends commands to tester 302 to initiate execution of routines and programs for testing DUT 301. Such executing test programs may initiate the generation and transmission of test signals to the DUT 301 and collect responses from the DUT. Various types of DUTs may be tested by system 300. For example, DUTs may be avionics, radar, weaponry, semiconductor devices, and so forth.

To provide test signals and collect responses from the DUT, tester 302 is connected, via an appropriate FPGA interface, to one or more connector pins that provide an interface for the internal circuitry of DUT 301. For illustrative purposes, in this example, device tester 302 is connected to a connector pin of DUT 301 via a hardwire connection to deliver test signals (to the internal circuitry of DUT 301). Device tester 302 also senses signals at DUT 301 in response to the test signals provided by device tester 302. For example, a voltage signal or a current signal may be sensed at a DUT pin in response to a test signal. Such single port tests may also be performed on other pins included in DUT 301. For example, tester 302 may provide test signals to other pins and collect associated signals reflected back over conductors (that deliver the provided signals). By collecting the reflected signals, the input impedance of the pins may be characterized along with other single port testing quantities. In other test scenarios, a digital signal may be sent to DUT 301

for storage on DUT 301 . Once stored, DUT 301 may be accessed to retrieve and send the stored digital value to tester 302. The retrieved digital value may then be identified to determine if the proper value was stored on DUT 301 .

Along with performing one-port measurements, a two-port test may also be performed by device tester 302. For example, a test signal may be injected to a pin on DUT 301 and a response signal may be collected from one or more other pins of DUT 301 . This response signal is provided to device tester 302 to determine quantities, such as gain response, phase response, and other throughput measurement quantities.

Referring also to Fig. 4, to send and collect test signals from multiple connector pins of a DUT (or multiple DUTs), device tester 302 includes an interface card 401 that can communicate with numerous pins. For example, interface card 401 includes the one or more FPGAs described herein, which may be used to transmit test signals to the DUT and to collect corresponding responses. Each communication link to a pin on the DUT may constitute a channel and, by providing test signals to a large number of channels, testing time may be reduced since multiple tests may be performed simultaneously. Along with having many channels on an interface card, by including multiple interface cards in tester 302, the overall number of channels increases, thereby further reducing testing time. In this example, two additional interface cards 402 and 403 are shown to demonstrate that multiple interface cards may populate tester 302.

Each interface card may include dedicated integrated circuit circuitry, including, e.g., an FGPA and embedded processing device (as described, e.g., Fig. 1), for performing particular test functions. This circuitry may implement, e.g. a pin electronics (PE) stage for performing PE tests, and a parametric measurement unit (PMU) stage for performing tests. Typically PMU testing involves providing a (programmable) DC voltage or current signal to the DUT to determine such quantities as input and output impedance, current leakage, and other types of DC performance characterizations. PE testing involves sending DC or AC test signals, or waveforms, to a DUT (e.g., DUT 301) and collecting responses to further characterize the performance of the DUT. For example, the PE stage may transmit (to the DUT) AC test signals that represent a vector of binary values for storage on the DUT. Once these binary values have been stored, the DUT may be accessed by tester 302 to determine if the correct binary values have been stored.

In some arrangements, an interface device may be used to connect one or more conductors from tester 302 to the DUT. For example, the DUT may connect to an Interface Test Adapter (ITA) which interfaces with an Interface Connection Adapter (ICA) that connects with the tester. The DUT (e.g., DUT 301) may be mounted onto a device interface board (DIB) for providing access to each DUT pin. In such an arrangement, a DUT conductor may be connected to the DIB for placing test signals on the appropriate pin(s) of the DUT. Additionally, in some arrangements, tester 302 may connect to two or more DIBs for interfacing the channels provided by interface cards 401 to 403 to one or multiple DUTs.

To initiate and control the testing performed by interface cards 401 to 403, tester 302 includes a PE controller 408 (e.g., in a system processing device, in an embedded processing device, or in programmable logic) to provide test parameters (e.g., test signal voltage level, test signal current level, digital values, etc.) for producing test signals and analyzing DUT responses. Tester 302 also includes a network interface 409 that allows computer system 305 to control the operations executed by tester 302 and also allows data (e.g., test parameters, DUT responses, etc.) to pass between tester 302 and to computer system 305.

The computer system, or another processing device used on or associated with test system 300, may be configured to exchange communications with a test program running on tester 302 through active communication channels with the device tester. The computer system may be, or include, computer 102 of Fig. 1. Alternatively, computer 102 may be part of tester 302 and the computer system described with respect to Fig. 4 may communicate with computer 102.

The foregoing describes performing testing using a system processing device, embedded processing devices, or programmable logic. However, testing, as described herein, may be performed using a combination of system processing device, embedded processing devices, or programmable logic. For example, each of these different elements may run on or more test programs simultaneously to test the same device or portion thereof. Likewise, these different elements may coordinate testing so that, e.g., a system processing device (e.g., 102 of Fig. 1) performs a first part of a test sequence, an embedded processing device (e.g., 106

of Fig. 1) performs a second part of the same testing sequence, and programmable logic (e.g., FPGA 111 of Fig. 1) performs a third part of the same testing sequence. Any appropriate coordination may take place between the different programmable elements of the test instrument described herein.

Furthermore, in some implementations, a tier of processing may be circumvented. For example, testing may occur using a system processing device (e.g., 102) and programmable logic (e.g., FPGA 111), but not an embedded processing device. In such implementations, communications between the system processing device and the programmable logic may pass through an embedded processing device or bypass the embedded processing device tier altogether.

In some implementations, there may be more than three tiers of processing devices. For example, there may two tiers of embedded processing devices (resulting, e.g., in four tiers total). For example, a single embedded processing device may be used to coordinate testing of a single device, and different embedded processing devices (under the direction of that single embedded processing device) may be used to test different aspects or features of that single device.

In some implementations, one or more tiers of processing devices may be eliminated from the system of Fig. 1. For example, some implementations may not include a tier of embedded processing devices. In such example systems, there may be only a system processing device (e.g., 102 of Fig. 1) and programmable logic (e.g., FPGAs 111 to 113. In this regard, any appropriate combination of tiers may be employed in the test instrument described herein.

In some implementations, the system processing device (e.g., 102 of Fig. 1) may be external to the test instrument. For example, an external computer may be employed to control operations of the test instrument, and may interact with embedded processing device(s) and programmable logic on the test instrument in the manner described herein. In other implementations, the system processing device may be part of the test instrument or remote from the test instrument (e.g., connected to the test instrument over a network).

In some implementations, the programmable logic may be replaced with non-programmable logic. For example, rather than using an FPGA, one or more application-specific integrated circuits (ASICs) may be incorporated into the test instrument in place of, or in addition to, the programmable logic described herein.

The functionality described herein, or portions thereof, and its various modifications (hereinafter "the functions"), are not limited to the hardware described herein. All or part of the functions can be implemented, at least in part, via a computer program product, e.g., a computer program tangibly embodied in an information carrier, such as one or more non-transitory machine-readable media, for execution by, or to control the operation of, one or more data processing apparatus, e.g., a programmable processor, a computer, multiple computers, and/or programmable logic components.

A computer program can be written in any form of programming language, including compiled or interpreted languages, and it can be deployed in any form, including as a stand-alone program or as a module, component, subroutine, or other

unit suitable for use in a computing environment. A computer program can be deployed to be executed on one computer or on multiple computers at one site or distributed across multiple sites and interconnected by a network.

Actions associated with implementing all or part of the functions can be performed by one or more programmable processors executing one or more computer programs to perform the functions of the calibration process. All or part of the functions can be implemented as, special purpose logic circuitry, e.g., an FPGA and/or an ASIC (application-specific integrated circuit).

Processors suitable for the execution of a computer program include, by way of example, both general and special purpose microprocessors, and any one or more processors of any kind of digital computer. Generally, a processor will receive instructions and data from a read-only memory or a random access memory or both. Components of a computer include a processor for executing instructions and one or more memory devices for storing instructions and data.

Components of different embodiments described herein may be combined to form other embodiments not specifically set forth above. Components may be left out of the circuitry shown in Figs. 1 to 4 without adversely affecting its operation. Furthermore, various separate components may be combined into one or more individual components to perform the functions described herein.

Other embodiments not specifically described herein are also within the scope of the following claims.

What is claimed is: