(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
2 April 2015 (02.04.2015)

WIPO | PCT

(10) International Publication Number
WO 2015/048366 A1

(51) International Patent Classification:
G06F 11/263 (2006.01)

(21) International Application Number:
PCT/US2014/057579

(22) International Filing Date:
26 September 2014 (26.09.2014)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
14/038,743 26 September 2013 (26.09.2013) US

(63) Related by continuation (CON) or continuation-in-part (CIP) to earlier application:
US 14/038,743 (CON)
Filed on 26 September 2013 (26.09.2013)

(71) Applicant: **TEXAS INSTRUMENTS INCORPORATED** [US/US]; P.O. Box 655474, Mail Station 3999, Dallas, TX 75265-5474 (US).

(71) Applicant *(for JP only)*: **TEXAS INSTRUMENTS JAPAN LIMITED** [JP/JP]; 24-1, Nishi-Shinjuku 6-chome, Shinjuku-ku Tokyo 160-8366 (JP).

(72) Inventors: **GAHOI, Anshul**; #102, 97/1, 7th Cross, Opposite Indian Oil Petrol, Domlur, Bangalore 560071 (IN). **SANTHANAGOPAL, Raghavendra**; 17, #02, Veerappa Reddy Layout, 6A Cross, Kaggadaspura, C.V. Raman Nagar Post, Bangalore 560093 (IN). **BABU, Pradeep, Kumar**; G6 Maruthi Enclave, 6thA Cross, Kaggadaspura Main Rd., C.V. Raman, Bangalore 560093 (IN).

(74) Agents: **DAVIS, Jr., Michael, A.** et al.; International Patent Manager, Texas Instruments Incorporated, P.O. Box 655474, Mail Station 3999, Dallas, TX 75265-5474 (US).

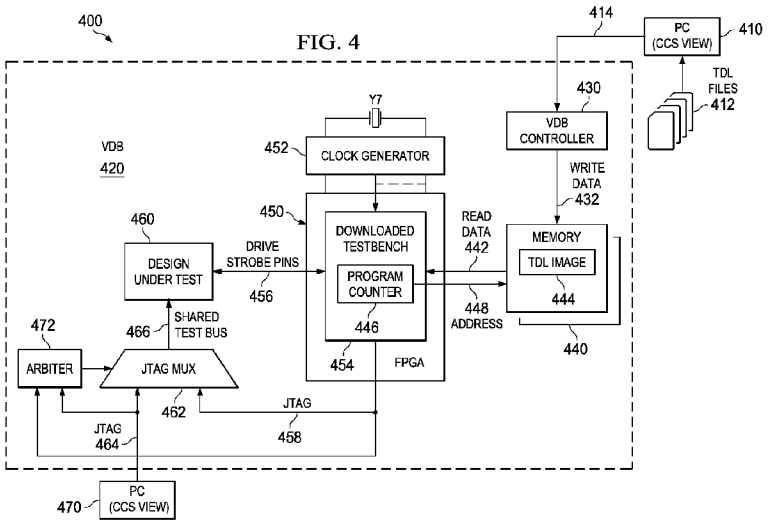
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK,

[Continued on next page]

(54) Title: PROGRAMMABLE INTERFACE-BASED VALIDATION AND DEBUG

FIG. 4



(57) Abstract: In described examples, a test connector is arranged to communicatively couple a design under test (460) to a test fixture. A programmable logic interface (450) is communicatively coupled to the test connector and is arranged to receive a downloadable test bench (454). The downloadable test bench (454) is arranged to apply test vectors from a first set of test vectors (444) to a first test control bus (458). A multiplexer (462) is arranged to selectively couple one of the first test control bus (458) and a second test control bus (464) to a shared test bus (466) that is coupled to the test connector. The second test control bus (464) is arranged to apply test vectors from a second set of test vectors.

WO 2015/048366 A1

SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

— *with international search report (Art. 21(3))*

PROGRAMMABLE INTERFACE-BASED VALIDATION AND DEBUG

[0001] This relates in general to automatic test equipment, and in particular to programmable interface-based validation and debug.

BACKGROUND

[0002] Automatic test equipment (ATE) is useful for validation and debugging of many electronic designs, and especially the electronic designs implemented using integrated circuits. Because of the relative complexity of the integrated circuits, debugging and device validation is typically time consuming and involves much iteration of test vectors using the ATE. Because of the complexity of the ATE, the cost for using the ATE for debugging of the designs-under-test is commensurately expensive. The difficulty of debugging is further compounded by input/output limitations of the interface to exercise the design-under-test.

SUMMARY

[0003] In described examples, a test connector is arranged to communicatively couple a design under test to a test fixture. A programmable logic interface is communicatively coupled to the test connector and is arranged to receive a downloadable test bench. The downloadable test bench is arranged to apply test vectors from a first set of test vectors to a first test control bus. A multiplexer is arranged to selectively couple one of the first test control bus and a second test control bus to a shared test bus that is coupled to the test connector. The second test control bus is arranged to apply test vectors from a second set of test vectors.

BRIEF DESCRIPTION OF THE DRAWINGS

[0004] FIG. 1 shows an illustrative electronic device in accordance with example embodiments of the disclosure.

[0005] FIG. 2 is a flow diagram of a test flow for performing debugging using a validation design and debug board in accordance with example embodiments of the disclosure.

[0006] FIG. 3 is a block diagram of components of a validation design and debug board in accordance with example embodiments of the disclosure.

[0007] FIG. 4 is an architectural diagram of components of a validation design and debug board system in accordance with example embodiments of the disclosure.

[0008] FIG. 5 is an architectural diagram of a validation design and debug board bus interface for a design under test in accordance with example embodiments of the disclosure.

DETAILED DESCRIPTION OF EXAMPLE EMBODIMENTS

[0009] FIG. 1 shows an illustrative computing device 100 in accordance with preferred embodiments of the disclosure. For example, the computing device 100 is, or is incorporated into, an electronic system 129, such as a laptop, a mobile device, a personal digital assistant, a personal computer, electronics control "box" or display, electronic test bench, or any other type of electronic system. The computing device 100 can be powered from line current, solar cells, and/or battery (or capacitive) storage. In one example, the computing device 100 includes a programmable interface-based validation and debug system that provides a cost-effective solution for enhancing ad-hoc network communications.

[0010] In some embodiments, the computing device 100 includes a megacell or a system-on-chip (SoC) which includes control logic such as a CPU 112 (central processing unit), a storage 114 (such as random access memory (RAM)) and a VDB debugger 110 (such as executable software at least partially embodied in hardware) program that it is arranged to access the VDB (validation design and debug board) 131. The CPU 112 can be, for example, a CISC-type (Complex Instruction Set Computer) CPU, RISC-type CPU (Reduced Instruction Set Computer), MCU-type (Microcontroller Unit), or a digital signal processor (DSP). The storage 114 (which can be memory such as on-processor cache, off-processor cache, RAM, flash memory, or disk storage) stores one or more software applications 130 (such as embedded applications) that, when executed by the CPU 112, perform any suitable function associated with the computing device 100.

[0011] The CPU 112 includes memory and logic that store information frequently accessed from memory provided by the storage 114. The computing device 100 is often controlled by a user using a UI (user interface) 116, which provides output to and receives input from the user during the execution the software application 130. The output is provided using the display 118, indicator lights, a speaker, vibrations, and other ways. The input is received using audio and/or video inputs (using, for example, voice or image recognition), and mechanical devices such as keypads, switches, and proximity detectors. The CPU 112 and VDB debugger 110 is coupled to I/O (Input-Output) port 128, which provides an interface such as a JTAG (joint test action group) interface that is configured to receive input from (and/or provide output to) the VDB 131. The

I/O port 128 can be a bus such as a USB (universal serial bus), RS-232 (Radio Sector of the Electronic Industries Alliance), and I2C (inter-integrated circuit). The computing device 100 can also be coupled to peripherals and/or computing devices, including tangible, non-transitory media (such as flash memory) and/or cabled or wireless media. These and other input and output devices are selectively coupled to the computing device 100 by external devices using wireless or cabled connections. The storage 114 can be accessed by, for example, external devices via I/O port 128.

[0012] The VDB debugger 110 is a monitoring and control system and includes logic and functionality (embodied at least partially in hardware and at least partially controlled by software application 130, which is at least partially stored in storage 114) that supports monitoring, controlling, testing, and debugging of a design-under-test (DUT). For example, the VDB debugger 110 is useful to download and/or control programmable logic (embodied at least partially in hardware) interface such as VDB FPGA (field-programmable gate array) 142 included in a test fixture such as validation design and debug board (VDB) 142. Accordingly, in one example, the debugger 110 is a self-contained computer system (such as a laptop personal computer) that is different from the test fixture such as the VDB 142.

[0013] The components of the VDB debugger 110 and VDB 142 allow validation of a physical DUT and/or an emulator of the DUT via connector J1. For example, by plugging an emulator's socket into the connector J1, the emulator can replicate the DUT's performance of operations in an intended final embodiment (such as a packaged integrated circuit) under various circumstances (such as how the components of the DUT would interact with the software application 130). In this way, the DUT can be debugged in a single environment that resembles pre-production or post-production operation.

[0014] Conventional test equipment typically provides a predominantly hardware-oriented environment that interacts with designs-under-test in a relatively fixed manner. For example, test vectors (such as information for setting or validating the setting of test pins, even when the test pins appear in a single line of code) are loaded into a DUT memory. In one example, after program code is loaded into the device memory through the test vectors, no external control is imposed on execution of the test vectors. After execution of the test vectors, test results are often limited to a signature that provides an indication of whether the test passed or failed.

[0015] In contrast, The VDB debugger 110 provides control over and visibility into DUT internal circuits. For example, the VDB debugger 110 includes debugger software such as code composer studio (CCS) 111 that is used, for example, for debugging a processor (such as a microprocessor, DSP, and microcontroller) integrated within the DUT. The CCS 111 provides debugging commands. In one example, the debugging commands are high-level commands, which are converted into JTAG commands for debugging the DUT. When used as a test master (as discussed further below), the CCS 111 provides visibility (such as of the status of DUT internal registers and memory) and debugging functionality (such as step, run, halt, and breakpoint) that surpasses, for example, validation low-level commands asserted using a JTAG (joint test action group) interface. Thus, a processor in a DUT can be halted and single stepped through the code while examining the state of internal registers as an aid to determine the causes and locations of failures that occur using the validation low-level commands.

[0016] Accordingly, the VDB 142 is useful for initial device testing, for device confidence building (such as by executing design validation and application test after the design is reduced to silicon), as a customer demonstration board, for field failure testing, and for other similar purposes. Additionally, the VDB FPGA 142 can be reprogrammed to adjust to (and/or provide) new interface requirements for revised or new designs-under-test, which promotes reusability and shortens design times for test equipment and fixtures.

[0017] FIG. 2 is a flow diagram of a test flow for performing debugging using a validation design and debug board in accordance with example embodiments of the disclosure. Generally described, test flow 200 includes an example process for loading test vectors onto a design under test via a programmable interface such as VDB FPGA 142. Although the test flow is described as being related to a DUT that is embodied in ("reduced to") silicon as a monolithic integrated circuit, the disclosure is applicable to emulators of the DUT that can be coupled to the VDB FPGA 142 (such as via connector J1).

[0018] More particularly, the test flow begins at node 202 and proceeds to operation 210, where the validation design and debug board 131 is initialized for testing. The validation design and debug board is initialized, for example, by using a netlist to program the VDB FPGA 142 to couple test vectors (such as received from VDB debugger 110) to specific inputs of the DUT. The specific inputs are assigned to specific pins (such as an ordered "pin-out") in accordance

with pins in a package (such as the pins in package of an integrated circuit) and/or connector (such as connector J1).

[0019] After programming of the VDB FPGA 142, signals on the VDB 131 are placed in correct operating states. For example, the reset signal can be de-asserted, clocks of selected operating frequencies can be coupled to respective inputs of the DUT, and the status of electronic fuses in the DUT can be ascertained. After initialization, test flow continues in operation 220.

[0020] In operation 220, a design for test (DFT) mode is initiated. In contrast to conventional art (where, for example, JTAG-based tests are initiated and control and supervision of the process is suspended until completion of the tests), the validation design and debug board is arranged to enter a DFT mode while allowing a supervisory process (such as controlled and viewed by CCS 111) to monitor and change values of individual registers and memory locations of the DUT, run specific ranges of test vectors (such as single-stepping and/or selected blocks of test vectors), and halt test execution (such as at predetermined locations). After the design for test mode is entered, test flow 200 continues in operation 230.

[0021] In operation 230, program code for testing a DUT is loaded into the validation design and debugs board 131. More particularly, specific program code for testing a particular portion of the DUT is loaded and the various portions of the DUT can be tested loading different program code that is specific to any selected portion of the DUT that is to be tested. Thus, for example, program code for configuring a test scenario for the DUT (as well as other such program code) can be loaded onto the validation design and debug board 131 so that the program code portions is resident on the validation design and debug board 131. The program code can be loaded into one of the memories as described below with respect to FIG. 3 and following.

[0022] Each of the operations in test flow 200 can be interrupted by a supervisory process in test flow 200 such that a different operation (such as operation 230 or operation 240) can be interrupted and the status of various components can be examined by the supervisory process. Thus, for example, results of a test being executed can be examined before the test being performed is halted or otherwise terminated. Test flow 200 continues in operation 240.

[0023] In operation 240, the program code is executed. For example, a command (such as a debugging command) is received from the supervisor process (such as CCS 111), which directs the validation design and debug board 131 to apply (in response to the received command) selected test vectors to respective pins of the DUT. In an example where the test vectors are

TDL- (test-design-language-) based, mastering of a shared JTAG bus is awarded to the supervisor process and a test sequence from the supervisor process is applied to JTAG-specific pins of the DUT in accordance with the received command.

[0024] For example, a test can be written in TDL where the test can be represented as a matrix having rows and columns. Each row of the TDL is associated with a particular time period (such as time slice or test clock) and where the stimulus of each row is sequentially applied to (and the response measured from) design pins of the DUT at a particular time of the test. In one example, each column is associated with a design pin, and each row of TDL has values that are applied as stimulus and/or values, which are compared against responses received on a design pin. Accordingly, values can be applied and/or measured at a point in the test flow that is associated with the particular row of TDL.

[0025] In various embodiments, the operations of the test flow 200 are not necessarily performed sequentially. For example, a first portion of the DUT can be tested by executing the code for testing the first portion, while a second a second portion of the DUT can be tested by executing the code for testing the second portion, where the code for testing the first portion is executed during a first time period that at least partially overlaps a second time period in which code for testing the second portion is executed. Accordingly responses to asynchronous events can be tested by varying the relative time at the code for testing the first and second portions are initiated (such as in response to separate commands received from CCS 111).

[0026] In operation 250, the test results are accessed ("dumped") for analysis. The test results can be accessed by monitoring I/O traces, comparing test signatures, reading on-DUT JTAG registers, reading memory and/or register contents, reading results of built-in-test (BIT) devices stored in memory, and by other similar operations.

[0027] FIG. 3 is a block diagram of components of a validation design and debug board in accordance with example embodiments of the disclosure. Validation design and debug board 300 is a validation design and debug board such as VDB 131 that includes one or more interfaces for communicating with a VDB controller such as VDB debugger 110. For example, a VDB debugger 110 is arranged to communicate with VDB 300 via proprietary interface (I/F) (such as VDB I/F 302) or via a standards-based interface (such as JTAG interface 304).

[0028] Validation design and debug board 300 includes one or more power supplies for powering the VDB 300 and a DUT coupled to the VDB 300 via connector J1. For example,

VDB 300 includes power supplies 310 such as programmable 312 power supplies PPS1-PPS10 and fixed 314 power supplies FPS1-FPS7. Programmable power supplies PPS1-PPS10 can be programmed by the VDB debugger 110 to provide voltages for programming VDB FGPA 336 and powering selected portions of a DUT. Fixed power supplies FPS1-FPS7 are arranged to provide voltages suitable for semiconductor designs, such as 0.9V, 1V, 1.2V, 1.8V, 2.5V, 3.3V, and 5V, respectively. Power supply output (as well as analog signal output from the DUT) can be measured using measurement unit 316. Measurement unit 316 includes an electronic switch matrix S1 that is arranged to selectively couple (such as in response to a command from the VDB debugger 111) signals from the (fixed or programmable) power supplies or DUT pins (such as via connector J1) to an input of the analog-to-digital converter (ADC1).

[0029] Validation design and debug board 300 also includes one or more clock generators 320 for powering the VDB 300 and a DUT coupled to the VDB 300 via connector J1. For example, VDB 300 includes clock generators 320 such as programmable 322 clock generators CLK1-CLK15 and fixed 324 clock generators Y1-Y7. Programmable clock generators CLK1-CLK15 can be programmed by the VDB debugger 110 to provide selected frequencies (and/or phases) for clocking selected portions of a DUT.

[0030] Validation design and debug board 300 also includes one or more daughter cards (DC) such as DUT DC 330, memory (MEM) DC 346, peripheral (PERIPH) DC 360, and a traceback DC 370. The VDB 300 is arranged to receive the daughter cards using connectors each of which is arranged to provide power and signals to a daughter card and receive signals from the daughter card. The functionality of each daughter card can be partially or completely implemented directly on the VDB 300. However, the use of daughter cards facilitates upgrading of the VDB 300 capabilities because, for example, an entire daughter card can be replaced as new generations of components (such as memory) for the daughter card being replaced become available.

[0031] The design-under-test daughter card (DUT DC) 330 includes functionality for hosting (and/or hosting an emulator socket for) the DUT. For example, DUT DC 330 includes a DUT connector J1 that is arranged to receive an integrated circuit containing a DUT or an emulation socket (and/or plug). The connector J1 is also arranged to couple (such as electrically couple) signals from the DUT to circuitry of the DUT DC 330. Thus, the VDB debugger 110 is arranged to communicate with the DUT via the VDB 131 and DUT DC 330. The DUT DC 330 also

includes direct communication interfaces such as JTAG interface 334, USB version 2.0 interface 338, and Ethernet (physical layer) interface 340. The direct communication interfaces allow direct communications with a variety of external peripheral devices for testing and evaluation of the DUT.

[0032] The DUT DC also includes the VDB FPGA 336. The VDB debugger 110 is arranged to program a synthesized "hardware test bench" that is arranged to apply selected test vectors to be used for testing the DUT. The synthesized hardware test bench program is ported to (such as used for programming) the VDB FPGA 336. The VDB FPGA 336 is programmed in accordance with the "pin-out" (including the functionality and ordering of the input/output signals) of the DUT. Thus, the VDB FPGA 336 is arranged to apply (such as sequentially) the selected test vectors onto corresponding pins of the DUT ("design pins") when testing the DUT (such as a manufactured monolithic semiconductor, an emulation device that is emulating a design, and an electronic system having an interface suitable for coupling to the connector J1).

[0033] Once the selected test vectors are loaded into the DUT memory (such as using a drive/strobe-pins bus 456 and/or JTAG bus 458) via the VDB FPGA 336, the control of the executing the selected test vectors is directed to a supervisory process such as the VDB debugger 110 coupled to the VDB 300. The VDB debugger 110 has, for example, a complete processor view of various registers or memory contents in the device under test. The VDB debugger 110 is capable of halting the execution, single stepping through the program code, and adding breakpoints, which helps to shorten and quicken debug cycles. Accordingly the downloading of the synthesizable hardware test bench to the VDB FPGA 336 provides expanded visibility and controllability over conventional ATE solutions.

[0034] Validation design and debug board 300 provides memory for use by the components of the DUT DC 330 for testing the DUT. For example, DUT DC 330 includes memory 332 that is closely coupled to the DUT connector J1 and can be accessed quickly with respect to reading from or writing to the DUT. Ancillary memory such as 32-bit wide static RAM (SRAM 342) and double-data rate RAM version 2 (DDR2) 344 provide large banks of memory that are selectively coupled to the DUT using wide data busses. The ancillary memory is useful to store, supply, and capture test stimulus and/or test results such as video or audio streams. One or more clock generators Y7 are provided for generating clock signals to control operation of components

of the DUT DC 330, such as to provide selected frequencies (and/or phases) for testing selected portions of a DUT.

[0035] Validation design and debug board 300 also includes one or more expanded memory cards (MEM DC) 360 for providing and/or coupling additional memory to the VDB 300. For example, VDB 300 includes asynchronous memory 348 and synchronous memory 352. Asynchronous memory includes asynchronous memory components 350, which can include memory such as flash memory and/or static RAM (SRAM). Synchronous memory includes synchronous memory components 352, which can include memory such as SRAM 354, DDR 356, and mobile DDR (MDDR) 358. Dual in-line package (DIP) sockets J2 are useful to expand or replace existing memory components such as asynchronous memory components 350 and synchronous memory components 352.

[0036] Validation design and debug board 300 also includes one or more peripheral daughter cards (PERIPH DC) 360 for providing and/or coupling additional functionality to the VDB 300. For example, VDB 300 includes a test interface such as JTAG interface (JTAG) 362 that provides a JTAG interface for coupling test vectors from external JTAG-enabled devices. The hardware test bench downloaded into the VDB FPGA 336 can be arranged to execute JTAG-related programming in test flows initiated by external JTAG-enabled devices. The VDB debugger 110 (such as hosted on PC 410) is arranged to control test flow programming initiated by external JTAG-enabled devices. Peripheral daughter cards (PERIPH DC) 360 provides other test interfaces for in-circuit emulation external peripherals such as Quickturn and Palladium test equipment using connectors such as J3-J9. An I2C (inter-integrated circuit) interface 366 is useful to couple (such as surface mounted) integrated circuits to the VDB FPGA 336, connector J1, VDB debugger 110, and other testers.

[0037] Validation design and debug board 300 also includes one or more traceback daughter cards (traceback DC) 370 for traceback functionality and/or interfacing to the VDB 300. For example, traceback DC 700 includes traceback memory 372, which is arranged to capture information for recording the environment that exists before a trigger and/or capture event occurs. The traceback memory can be arranged as a circular queue and to continuously record relatively large amounts of information where older information is overwritten such that the size of the circular queue determines how much traceback information is captured. When an event occurs (such as a command received in response to a JTAG controller (such as VDB debugger

110), the recording of the data (with the exception of a selected portion of after-event recording) is stopped so that the data surrounding the event is not overwritten. The captured data that surrounds the event is useful to debug the environment (such as device state) that surrounds the event.

[0038] The FPGA 384 can be programmed to implement control of the traceback functionality. In one example, oscillator Y7 is arranged to provide clocks for driving the traceback control circuitry, and to provide signals for writing to and reading from the traceback memory 372. An interface to the DUT (I/F to DUT) 370 is arranged to operate in response to the FPGA 384 and, for example, to couple information from the DUT to the traceback memory 372.

[0039] The traceback DC 370 includes various interfaces to provide traceback functionality The DUT DC 370 includes communication interfaces (such as serial communication interfaces), such as JTAG interface 374, USB version 2.0 interface 382, and Ethernet (physical layer) interface 380. The DUT DC 370 includes communication interfaces (such as serial and/or parallel communication interfaces), such as Hewlett-Packard interface (HPI), personal computer interface (PCI), and VLYNQ (Texas Instruments ® proprietary interface) interface 378. The direct communication interfaces extend internal busses, for example, to busses of external peripheral devices used for testing and evaluation of the DUT. The video interface 386 and "other" interface 388 are useful to receive stimulus and/or test results such as video or audio streams.

[0040] FIG. 4 is an architectural diagram of components of a validation design and debug board system in accordance with example embodiments of the disclosure. Validation design and debug board system 400 includes a personal computer (PC) 410 and TDL files 412 that are arranged for testing a DUT 460. The PC 410 is a computing device (such as a personal computing device), such as computing device 100 that is arranged to create a downloadable test bench for applying selected contents of the TDL files 412 to design pins of the DUT 460. The PC 410 is arranged to download the test bench and associated test vectors to the validation design and debug board 420 (which is a validation design and debug board such as VDB 300) using an interface such as JTAG bus 414.

[0041] VDB controller 430 is arranged to receive (via JTAG bus 414) an image of the downloadable test bench and associated test vectors. The received image of the downloadable test bench and associated test vectors stored in the selected TDL files are stored (via write bus

432) as contents of memory 440 (such as TDL image 444). The contents of memory 440 are arranged to be read by the FPGA 450 via read bus 442. The FPGA 450 (under the control of VDB controller 430, for example) is arranged to read and instantiate the received image of the downloadable test bench as downloaded test bench 454. When the downloadable test bench is instantiated (such as programmed) as downloaded test bench 454, oscillator Y7 and clock generator 452 are arranged to provide one or more clocks for driving and clocking the downloaded test bench 454.

[0042] The downloaded test bench 454 instantiated in FPGA 450 is arranged to communicate with the design under test (DUT) 460 via drive/strobe-pins bus 456 and JTAG bus 458. In one example, drive/strobe-pins bus 456 is not shared with another bus master. JTAG bus 458 is coupled to a first input of JTAG multiplexer (MUX) 462. When the downloaded test bench 454 is in a test master mode (such as with respect to the DUT 460), the JTAG multiplexer 462 is arranged to couple the JTAG bus 458 to the JTAG-bus design pins of the DUT 460 via shared test bus 466. Shared test bus 466 is a test controller bus (in contrast to the drive/strobe-pins bus 456) and is useful, for example, to initiate testing or reading from a JTAG scan chain on the DUT. In one example, the shared test bus is coupled to a different set of pins of the DUT than the set of pins on the DUT to which the drive/strobe-pins bus 456 is coupled.

[0043] The downloaded test bench 454 is arranged to access test vectors from the TDL image 444 and, upon entering a design for test mode for example, sequentially apply time slices of the TDL image to the design pins during a test. For example, each slice (such as each "row" or "line") of the test vectors can be sequentially read by incrementing program counter 446 in accordance with a clock received from clock generator 452. The value of the program counter is useful to form an address that is asserted on address bus 448 for the purpose of addressing memory contents of the TDL image 444.

[0044] The PC 470 is arranged to communicate with the design under test (DUT) 460 via JTAG bus 464. JTAG bus 458 is coupled to a second input of JTAG multiplexer (MUX) 462. When the PC 470 is in a test master mode (such as with respect to the DUT 460), the JTAG multiplexer 462 is arranged to couple the JTAG bus 464 to the JTAG-bus design pins of the DUT 460 via shared test bus 466. For example, the "mastership" of the DUT 460 (and shared test bus 466) by CCS 111 executing on PC 470 allows the CCS 111 to view and control the test flow that is arranged to execute TDL from TDL image 444

[0045] In contrast with conventional solutions, full control of the execution flow of the code of the TDL image 444 is permitted, for example, using two test masters in an alternative fashion. In various embodiments the test masters can be (or be derived from) the same or different devices such that, for example, the PC 410 and the PC 470 can be the same or different physical (or virtual) devices. In the example embodiment, control of the execution flow of the code of the TDL image 444 is accomplished by establishing CCS 111 of PC 470 connectivity to the design on VDB for providing full control over the execution flow of the TDL image 444 that is loaded into the memory 440.

[0046] Full control of the execution flow of the code of the TDL image 444 can be accomplished by transferring test master control from the downloaded test bench 454 to the CCS 111 of PC 470. For example, the downloaded test bench 454 is arranged to act as a test master (such as by controlling JTAG pins of the DUT 460 via shared test bus 466) over loading and executing the code of the TDL image 444 in the DUT 460. After the execution has reached a certain point (such as a predetermined breakpoint set in advance by a CCS 111 of PC 410) in the TDL image 444, the downloaded test bench 454 is arranged to transfer control of the test to the CCS 111 of PC 470 such that the CCS 111 of PC 470 becomes the new test master. Arbiter (such as a mechanism embodied as logic gates in a silicon substrate) 470 is implemented to facilitate transference of the mastership of the test bus between the downloaded test bench 454 and the CCS 111 of PC 470. When the CCS 111 no longer has need of the mastership of the test bus, the downloaded test bench can resume testing (including restarting or executing a newly loaded test sequence) by acting as the current master of the test bus.

[0047] The arbiter 470 can select which of the downloaded test bench 454 and the CCS 111 of PC 470 is to be the master of shared bus 466. The arbiter 470 can select which is to be the master of shared bus 466 in response to a reset state, a command received from a controller (such as the VDB controller 430 or the VDB debugger 110) or evaluating contents of the JTAG bus 464, a DUT toggled output, and/or the contents of the JTAG bus 458.

[0048] FIG. 5 is an architectural diagram of a validation design and debug board bus interface for a design under test in accordance with example embodiments of the disclosure. Validation design and debug board FPGA 336 (and/or FPGA 450) includes a vector memory 510 and an output enable memory 520 for driving input control logic (such as embodied at least partially in hardware) 530 via busses 512 and 522, respectively. Although the vector memory 510 and OEN

memory 520 are shown as being resident on the VDB FPGA 336, all or some portions of the vector memory 510 and OEN memory 520 can be resident off of the VDB FPGA 336 (while, for example, still being resident on the VDB 300).

[0049] Vector memory 440 is arranged to store each of the values that are to be driven onto each of the DUT 550 design pins. The vector memory 440 also is arranged to store information for controlling execution of various TDL constructs such as SETR (such as set values of a pin at an expected rate), RUNP (such as run "packed" by repeating a previously set value), and END (such as halt execution). As mentioned above, program counter 446 is useful to generate addresses for accessing test vectors. In an embodiment, the program counter includes a state machine that is arranged to fetch and pack the test vectors from the associated memories and apply the test vectors onto the design pins at the expected rate (such as the SETR construct). When a RUNP construct is encountered by the state machine, the previously packed value is driven onto the design pins until a RUNP count expires. When an END construct is encountered, the previously packed value is driven until, for example, an event that occurs externally to the FPGA 336 or 450 that resets (or changes) the state machine.

[0050] The OEN memory 522 is arranged to store the direction information (such as output and/or input) of a state of a DUT design pin (or group of pins) for a test vector (or group of test vectors). The memory storage can be optimized, for example, by associating direction information using one word (such as 32 bits) of direction information per vector or by using one word per line (such as per row) of TDL-based test vectors.

[0051] In operation, the downloaded test bench 454 is arranged, for example, to generate the direction information in response to the TDL image 444. When the test vectors in the vector memory 510 are being executed, addresses for selecting a particular line of TDL are also used for retrieving the associated direction information for the selected line of TDL. The retrieved associated direction information for the selected line of TDL is used for controlling the states of one or more tri-state buffers 540 (not all buffers are shown for the purpose of clarity). For example, the direction information is applied via bus 532, which determines whether selected contents (such as bits or words) of bus 534 are output by selected, enabled tri-state buffers 540 onto bus 542 (which is coupled to the DUT 550 via connector J1.

[0052] Modifications are possible in the described embodiments, and other embodiments are possible, within the scope of the claims.

CLAIMS

What is claimed is:

1. A test fixture, comprising:

a test connector that is arranged to communicatively couple a design under test to the test fixture;

a programmable logic interface that is communicatively coupled to the test connector and is arranged to receive a downloadable test bench, wherein the downloadable test bench is arranged to apply test vectors from a first set of test vectors to a first test control bus; and

a multiplexer that is arranged to selectively couple one of the first test control bus and a second test control bus to a shared test bus that is coupled to a first set of selected pins of the test connector, wherein the second test control bus is arranged to apply test vectors from a second set of test vectors.

2. The test fixture of claim 1, wherein the second set of test vectors is supplied by a debugger that is operable by a human to control execution of the first set of test vectors.

3. The test fixture of claim 2, comprising an arbiter that is arranged to transfer mastership of the shared test bus between the debugger and downloadable test bench.

4. The test fixture of claim 2, wherein the first set of test vectors is arranged to validate a performance of the design under test coupled to the test connector.

5. The test fixture of claim 1, comprising an addressable memory in which the first set of test vectors is stored.

6. The test fixture of claim 5, wherein the programmable logic interface includes a field-programmable gate array (FPGA) that is programmed to include a program counter that is arranged to generate an address by which test vectors from the first set of test vectors are retrieved.

7. The test fixture of claim 6, wherein the value of the program counter is changed in response to a value of retrieved test vectors from the first set of test vectors.

8. The test fixture of claim 7, wherein a first portion of the first set of retrieved test vectors from the first set of test vectors is applied to the first set of selected pins of the test connector, and wherein a second portion of the first set of retrieved test vectors for the first set of test vectors is applied to a third test bus that is coupled to a second set of selected pins of the test

connector, wherein the second set of selected pins of the test connector includes selected pins that are different from the selected pins in the first set of selected pins.

9. A test system, comprising:

a programmable logic interface that is arranged to receive a downloadable test bench, wherein the downloadable test bench is arranged to apply test vectors from a first set of test vectors to a shared test control bus that is coupled to a test connector that is arranged to communicatively couple a design under test to a shared control bus; and

a debugger having a user interface that is arranged to provide control over the operation of the first set of test vectors executing upon the design under test, wherein the debugger is arranged to apply test vectors from a second set of test vectors to the shared test control bus.

10. The test system of claim 9, comprising a multiplexer that is arranged to selectively couple one of the first test control bus and a second test control bus to the shared test bus, wherein the shared test bus is coupled to a first set of selected pins of the test connector.

11. The test system of claim 10, comprising an arbiter that is arranged to transfer mastership of the shared test bus between the debugger and downloadable test bench.

12. The test system of claim 11, wherein the programmable logic interface is arranged in a test fixture that includes the test connector and an interface for coupling the second control bus to the debugger, wherein the debugger is included in a computer system that is different from the test fixture.

13. The test system of claim 12, wherein the test fixture includes an addressable memory that is arranged to apply test vectors from the first set of test vectors to a second set of selected pins of the test connector, wherein the second set of selected pins of the test connector has pins that are different from pins included in the first set of selected pins of the test connector.

14. The test system of claim 13, wherein the programmable logic interface includes a field-programmable gate array (FPGA) that is programmed to include a program counter that is arranged to generate an address by which test vectors from the first set of test vectors is retrieved from the addressable memory.

15. The test system of claim 12, wherein the debugger is arranged to apply test vectors from the second set of test vectors that are arranged to perform high-level processor commands of a processor of the design under test.

16. The test system of claim 7, wherein the high-level processor commands include commands for transferring contents of the processor of the design under test to the debugger.

17. A method for validating and debugging a design under test including:

configuring a programmable logic device, wherein the programmable logic device is arranged in a test fixture having a test connector that is communicatively coupled to the design under test, and wherein the programmable logic device is configured as a downloaded test bench by using a netlist of a downloadable test bench;

coupling test vectors from the downloaded test bench to the design under test via a shared control bus; and

examining a result of an execution of the test vectors from the downloaded test bench by coupling a debugger to the shared test control bus and by executing test vectors coupled from the debugger upon the design under test.

18. The method of claim 17, wherein the test vectors coupled from the debugger include commands for transferring contents of the processor of the design under test to the debugger.

19 The method of claim 18, comprising coupling the debugger to a test fixture that includes the test connector, wherein the debugger is included in a computer system that is different from the test fixture.

20. The method of claim 20, comprising transferring control of the shared test bus from the programmable logic device to the debugger.

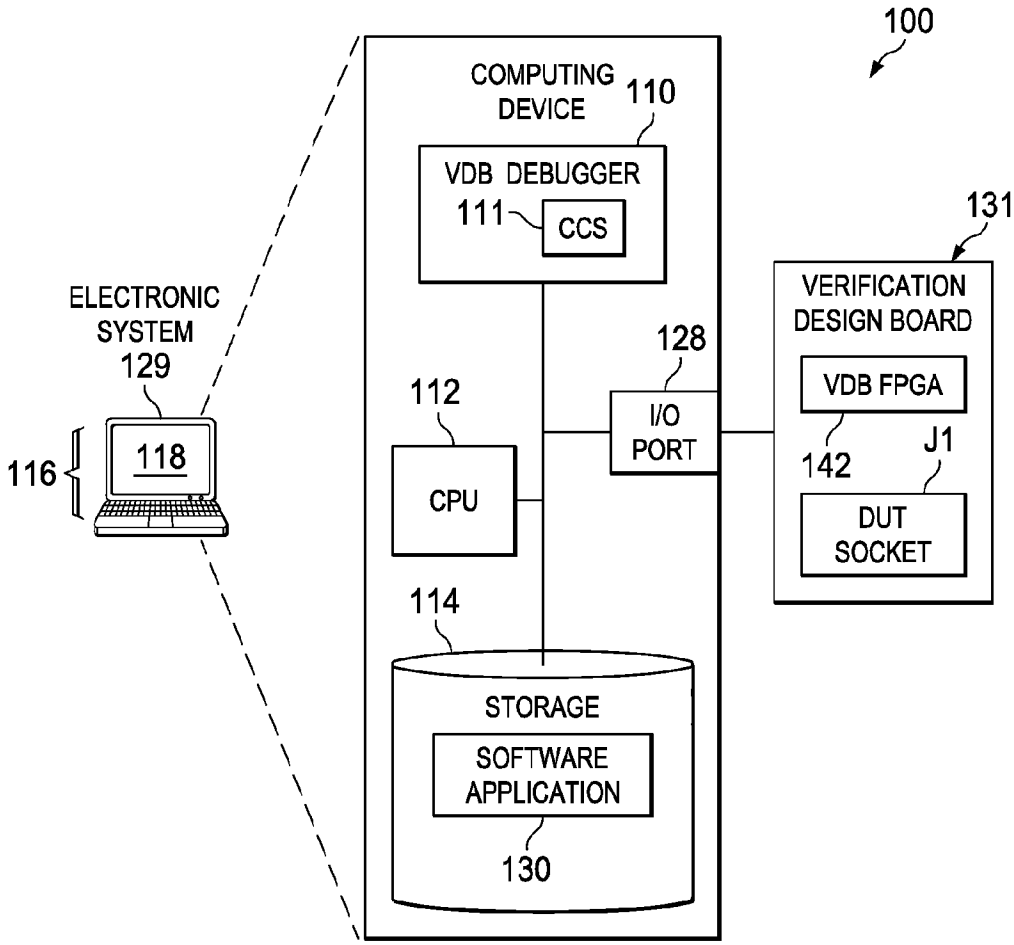

100
COMPUTING DEVICE
110
VDB DEBUGGER
111
CCS
131
VERIFICATION DESIGN BOARD
ELECTRONIC SYSTEM
129
128
112
I/O PORT
VDB FPGA
142
J1
116
118
CPU
DUT SOCKET
114
STORAGE
SOFTWARE APPLICATION
130

FIG. 1

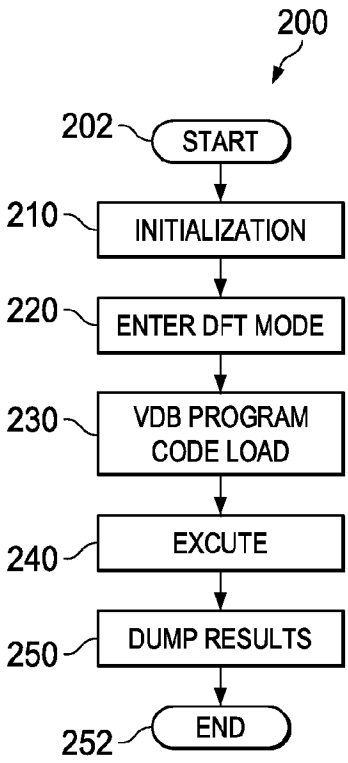

200
202
START
210
INITIALIZATION
220
ENTER DFT MODE
230
VDB PROGRAM CODE LOAD
240
EXCUTE
250
DUMP RESULTS
252
END

FIG. 2

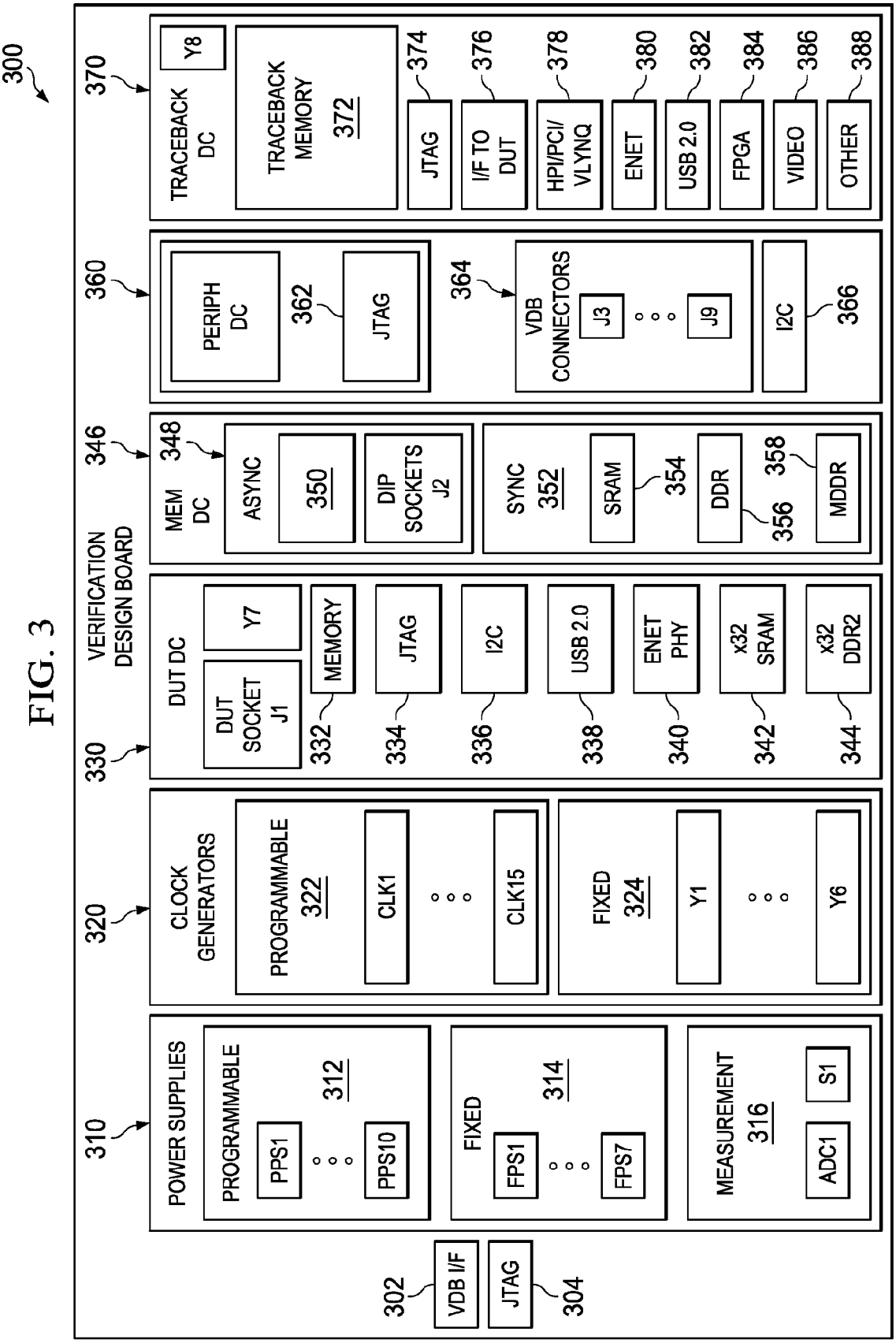

300
VERIFICATION DESIGN BOARD
310
POWER SUPPLIES
PROGRAMMABLE
PPS1
PPS10
312
FIXED
FPS1
FPS7
314
MEASUREMENT
316
ADC1
S1
320
CLOCK GENERATORS
PROGRAMMABLE
322
CLK1
CLK15
FIXED
324
Y1
Y6
330
DUT DC
DUT SOCKET J1
Y7
MEMORY
332
JTAG
334
I2C
336
USB 2.0
338
ENET PHY
340
x32 SRAM
342
x32 DDR2
344
346
MEM DC
348
ASYNC
350
DIP SOCKETS J2
SYNC
352
SRAM
354
DDR
356
MDDR
358
360
PERIPH DC
362
JTAG
364
VDB CONNECTORS
J3
J9
I2C
366
370
TRACEBACK DC
Y8
TRACEBACK MEMORY
372
JTAG
374
I/F TO DUT
376
HPI/PCI/ VLYNQ
378
ENET
380
USB 2.0
382
FPGA
384
VIDEO
386
OTHER
388
302
VDB I/F
JTAG
304

FIG. 3

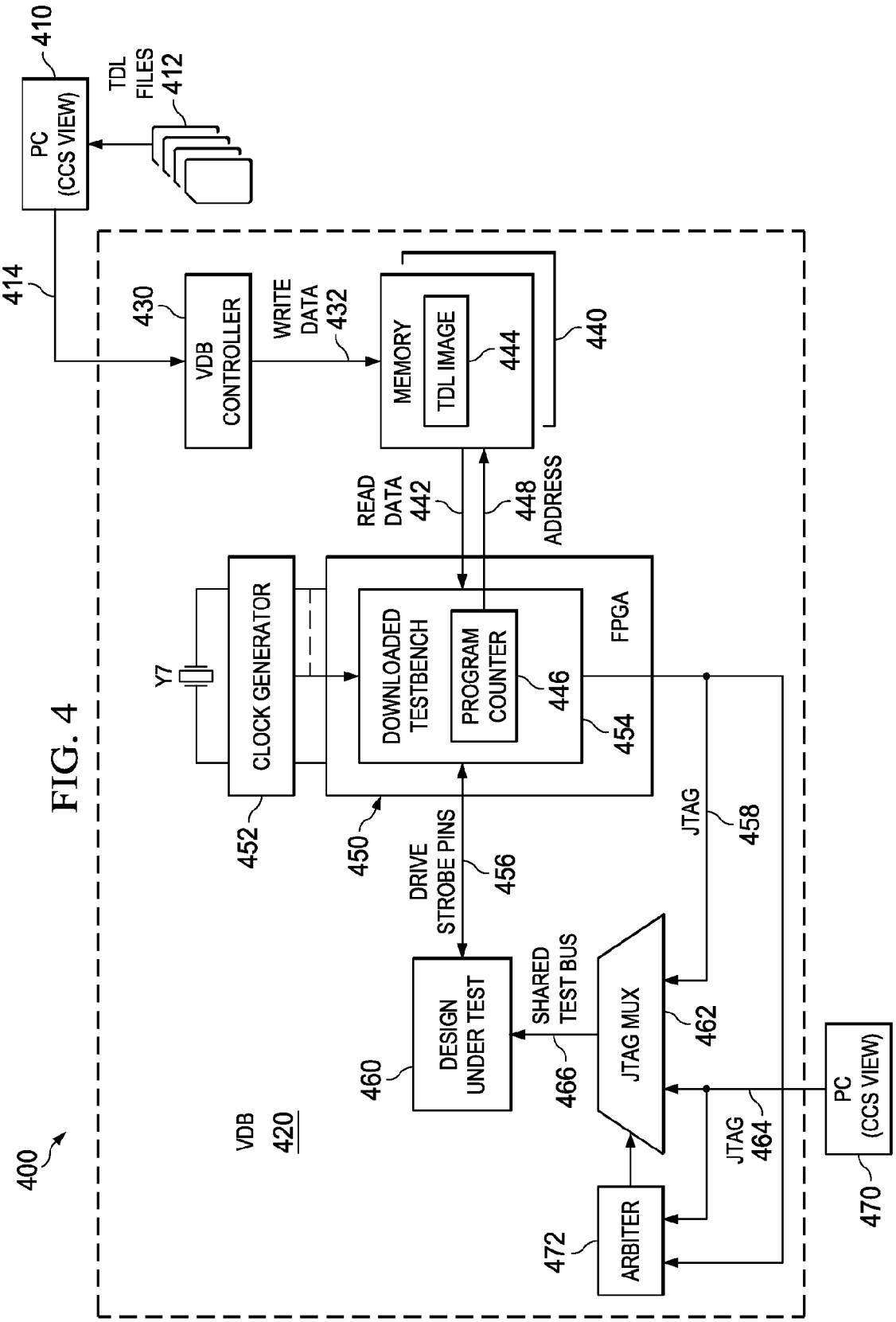
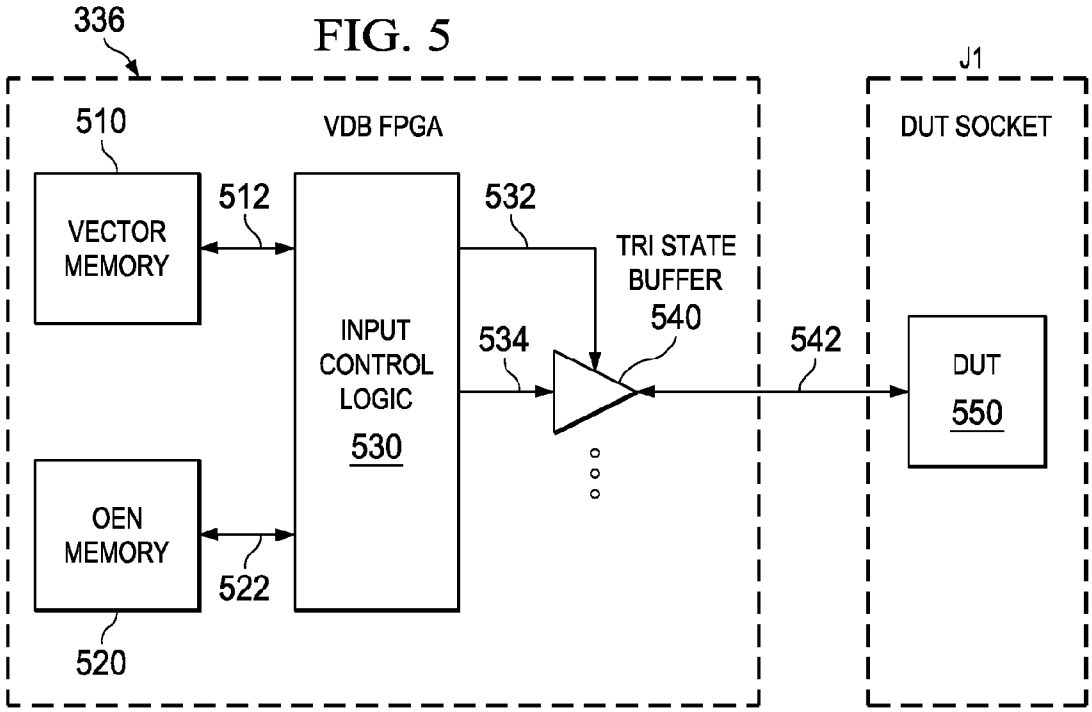

400
VDB
420
PC
(CCS VIEW)
410
TDL
FILES
412
414
VDB
CONTROLLER
430
WRITE
DATA
432
MEMORY
TDL IMAGE
444
440
READ
DATA
442
448
ADDRESS
Y7
CLOCK GENERATOR
452
DOWNLOADED
TESTBENCH
PROGRAM
COUNTER
446
FPGA
454
450
DRIVE
STROBE PINS
456
JTAG
458
DESIGN
UNDER TEST
460
SHARED
TEST BUS
466
JTAG MUX
462
JTAG
464
PC
(CCS VIEW)
470
ARBITER
472

FIG. 4

FIG. 5


336
J1
510
VDB FPGA
DUT SOCKET
VECTOR MEMORY
512
532
TRI STATE BUFFER
540
INPUT CONTROL LOGIC
530
534
542
DUT
550
OEN MEMORY
522
520

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US 2014/057579

A. CLASSIFICATION OF SUBJECT MATTER

G06F 11/263 *(2006.01)*

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F 3/00-3/18, 7/00-7/78, 9/00-9/54, 11/00-11/36, 13/00-13/42, 15/00-15/82, 17/00-17/50, G05B 19/00-19/46, G01R 31/00-31/44

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

PatSearch (RUPTO internal), USPTO, PAJ, K-PION, Esp@cenet, Information Retrieval System of FIPS

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5425036 A (QUICKTURN DESIGN SYSTEMS, INC.) 13.06.1995, abstract, col. 2, line 67 – col.3, line 24, lines 58-68, col. 4, lines 1-22, 50-61, col. 5, lines 3-35, line 56 –col. 6, lines 8-31, line 65 –col. 7, line 19, lines 55-68, col. 8, lines 1-13, col. 9, lines 1-16, claims 1, 11	1-20
A	US 5475624 A (SCHLUMBERGER TECHNOLOGIES, INC.) 12.12.1995	1-20
A	US 7188063 B1 (CYPRESS SEMICONDUCTOR CORPORATION) 06.03.2007	1-20

☐ Further documents are listed in the continuation of Box C.

☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
23 December 2014 (23.12.2014)	15 January 2015 (15.01.2015)
Name and mailing address of the ISA/RU: Federal Institute of Industrial Property, Berezhkovskaya nab., 30-1, Moscow, G-59, GSP-3, Russia, 125993 Facsimile No: (8-495) 531-63-18, (8-499) 243-33-37	Authorized officer V. Zhakovich Telephone No. (499) 240-25-91

Form PCT/ISA/210 (second sheet) (July 2009)