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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
27 January 2022 (27.01.2022)

WIPO | PCT

(10) International Publication Number
WO 2022/017589 A1

(51) International Patent Classification:
G06F 11/263 (2006.01) *G01R 31/3185* (2006.01)

(21) International Application Number:
PCT/EP2020/070599

(22) International Filing Date:
21 July 2020 (21.07.2020)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant *(for all designated States except US)*: **ADVANTEST CORPORATION** [JP/JP]; 1-6-2, Marunouchi, Chiyoda-ku, Tokyo, 100-0005 (JP).

(72) Inventor; and

(71) Applicant *(for US only)*: **RIVOIR, Jochen** [DE/DE]; Warmbronner Straße 13/3, 71106 Magstadt (DE).

(74) Agent: **BURGER, Markus** et al.; Schoppe, Zimmermann, Stöckeler, Zinkler, Schenk & Partner mbB, Radlkoferstraße 2, 81373 München (DE).

(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, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, IT, JO, JP, KE, KG, KH, KN, KP, KR, KW, 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, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, 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, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

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

WO 2022/017589 A1

(54) **Title:** AN AUTOMATED TEST EQUIPMENT, A PROCESS AND A COMPUTER PROGRAM FOR TESTING ONE OR MORE DEVICES UNDER TEST, WHEREIN DIFFERENT TEST ACTIVITIES MAKE USE OF SUBSETS OF THE DEVICE UNDER TEST RESOURCES

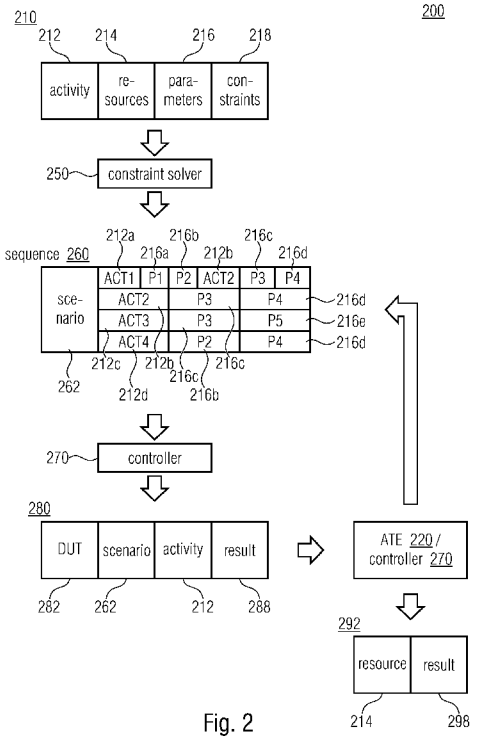

Fig. 2

(57) **Abstract:** An embodiment according to the present invention comprises an automated test equipment for testing one or more devices under test. The automated test equipment is configured to automatically, dynamically generate one or more, preferably a plurality of test scenarios comprising a plurality of test activities. Different test activities make use of subsets of the devices under test resources that can overlap. The automated test equipment is configured to generate a plurality of test scenarios, such that resources of the devices under test associated with a plurality of test activities of a test scenario do not conflict with each other.

An Automated Test Equipment, a Process and a Computer Program for Testing One or More Devices under Test, wherein Different Test Activities Make Use of Subsets of the Device under Test Resources

Technical Field

Embodiments according to the invention are related to automated test equipment. Further embodiments according to the invention are related to system level tests. Further embodiments according to the invention are related to on-chip system tests. Embodiments according to the invention are related to covering tests.

Background of the Invention

It is an empirical observation or truth that some devices fail system level tests (SLT) although they pass structural and/or parametric tests. This is usually attributed to unrealistic test conditions during structural tests.

For example, an activity pattern across the die area is very different during structural tests compared to system level tests. Structural tests spend most of the time with shift operations where the frequency is very low, but the pattern activity is very high. This leads to an unnatural loading of power supplies and therefore unrealistic voltage profiles versus location on die and time.

A further example is, that the power domain crossings and/or clock domain crossings are often not included in structural tests due to difficulties in automatic test pattern generation. Faults in this area will be neither sensitized nor detected.

Structural tests suffer from unrealistic test conditions, they are optimized for excellent fault detection capabilities. On the contrary, SLT consists exclusively of legitimate user scenarios. Without knowing any fault model, it is agreed or commonly believed that a failing SLT identifies a truly bad device.

There are certain disadvantages of system level tests, such as an SLT is by no means exhaustive, because a device under test (DUT) runs only a small subset of possible uses

and scenarios, in only one selected environment, such as supply voltages, frequencies, temperature.

Another disadvantage is that SLT fails are difficult to debug, because the time between fault sensitization and detection can be extremely long and/or the exact order of all activities is unknown and/or a simulation time for SLT would be way too long.

Further, SLTs have very long runtimes, ten minutes or more are no exception.

Moreover, SLT is difficult to deploy in test houses, because for each DUT a large number of system-boards must be maintained.

There is a need for an improvement of the system level test, which provides a better tradeoff between possible user scenarios, duration of runtimes, and practicality.

Summary of the Invention

An embodiment according to the present invention comprises an automated test equipment for testing one or more devices under test (DUT), e.g. system on a chip (SoC). The automated test equipment (ATE) is configured to automatically, dynamically generate one or more, preferably a plurality of test scenarios comprising a plurality of test activities. Different test activities, e.g. A1, A2, A3 make use of subsets of the DUT resources that can overlap, e.g. A1: CPU2, Mem3, A2: Mem3, A3: MPEG. The automated test equipment is configured to generate a plurality of test scenarios, such that resources of the DUT associated with a plurality of test activities of a test scenario do not conflict with each other.

An on-chip system test (OCST) can be expected to find more relevant problems than SLT does, because it exposes the DUT to a larger variety of legitimate test conditions than the SLT does. A problem can be considered relevant when it occurs during a combination of test activities, which could simulate or be part of a real use case.

OCST may vary external test conditions, such as supply voltages and frequencies to sensitize faults and/or internal test conditions, such as intensity of test activities, e.g. speed of data movements, in order to sensitize workload related faults. Once a fault is sensitized, OCST provides good observability to detect failing behavior.

An OCST may prove its effectiveness by showing which subset of SLT fails it is able to find. In other words, the OCST may be able to find the failing subset of the SLT.

Ideally, OCST tests can be reproduced in a simulation environment, for example, for debugging. OCST should also leverage existing on-chip resources such as design for test (DFT), design for debug (DFD), scan chains, logic built-in self-tests (LBIST) and/or memory built-in self-tests (MBIST). Also, multi-threading may be supported at interfaces between multi-core DUT and multiple ATE resources.

In a preferred embodiment, the plurality of test activities of a given test scenario is configured to be executed concurrently.

As the ATE is configured to generate test scenarios, in which the test activities use non-overlapping DUT resources, such that, test activities of a test scenarios do not conflict with each other. That is, testing can be sped up by executing the test activities of a given test scenario concurrently.

A general idea of the invention is to concurrently run a large number of simple, realistic, self-checking test activities on DUTs, using DUT resources, such as data block moves or built-in self-tests in a varying combination of test activity intensity and concurrency. Each test activity can check selected or involved IP blocks, but also contributes to the test conditions of other blocks through loading of supplies, clock trees and thermal coupling.

According to an embodiment, one or more of the test activities are associated with one or more test parameters, such as voltage, speed, size of data transfers, times between data transfers, block size into which the total data transfer is split, etc. Test parameters control the behavior of test activities. The test parameters are characterized by respective test parameter values, such as on/off, 5 V, 50 GB/s, etc. and/or are associated with one or more constraints or limits, e.g. for dynamic creation of test parameter values.

The same test activities with different parameter values may be executed in a test sequence comprising one or more test scenarios. Test activities may be associated with more than one test parameter and/or constraint. For example, a test parameter value, that is a value of a test parameter, such as voltage, can be varied from test activity to test activity. Test activities might also have some constraints or limits, for example, to protect the DUT or the

DUT resources, or to avoid scenarios that will not occur in real usage. Such constraint might be a limited writing speed, a temperature limit, a limited usage of memory, etc.

According to an embodiment, one or more of the test activities are characterized by one or more test parameter values outside predefined limitations or outside of chip specification, such as too low voltage values and/or too fast frequencies.

Results of testing DUTs outside or slightly outside of the predefined limitations, or chip specifications, might correlate with the performance of the DUTs within the predefined limitations or chip specifications. That is, DUTs with good test results outside of the chip specifications may hint at a good performance of the DUTs within the chip specifications. On the contrary, a fail of the test outside of the chip specification is considered not a hard hint or proof of a failing test within the chip specification, but indicates a weakness.

According to an embodiment, the plurality of test activities of a given test scenario are chosen randomly, under the constraints that resources of the device under test associated with the plurality of test activities of the given test scenario do not conflict with each other in order to avoid conflicting requirements on usage or test parameter values from different test activities.

A random mixture of test activities within a test scenario is needed for testing unknown interactions between activities or test activities, which occur in a real working environment. Randomization of parametrizable test activities covers many combinations of local workloads and thus mimics a large number of realistic workload patterns that are considered responsible for many system-level faults.

In a preferred embodiment the automated test equipment comprises a constraint solver for generating test scenarios in order to prevent occurrences of conflicts between the resources of the device under test.

Each test activity may have test parameters and optional constraints. For example, a data block write and read-back test activity can have the test parameters of initial time delay, block size, start address, subblock size, time delay before read back. Constraints can also be at the test scenario level, involving test parameters from multiple test activities.

A workstation or a controller, such as an on-chip OCST controller or an OCST card, may use a constraint solver to automatically generate test scenarios from test activities. The

constraint solver matches test activities that can coexist concurrently without violating resource constraints.

In a preferred embodiment, one or more of the test activities comprise activating a stress generator arranged on the device under test, e.g. an on-chip stress generator, such as a traffic generator to internal and/or external buses.

For example, existing or new design for test (DFT) structure can generate controlled stress and increase observability in order to increase the likelihood of detection (or for more likely fault detection). The stress generator may comprise a traffic generator to internal buses, and/or a traffic generator to external inputs in order to mimic traffic from missing external devices. The stress generator allows running built-in self-tests (BIST), such as memory or logic built-in self-tests (MBIST, LBIST) in selected wrapped IP blocks. Further, the stress generator may run input/output (I/O) lookback tests. Also, the stress generator may isolate IP blocks using a wrapper, such as IEEE1500 wrapper, and switch its scan chains into a linear-feedback shift register (LFSR).

According to an embodiment, the automated test equipment is configured to generate a test sequence, e.g. a sequence of test steps of test scenarios.

A test sequence of test scenarios is generated in order to test different mixtures of concurrent test activities in order to mimic possible realistic workload combinations.

According to an embodiment, the test sequence comprises two or more test scenarios with the same test activities, wherein the test scenarios differ by at least one test parameter value.

Test sequences may comprise test scenarios with the same test activities with different test parameter values in order to test the impact of the test parameters on the test result. Changing only one test parameter value of an IP block of the DUT may impact or influence another IP block of the DUT as well.

Further, conducting test scenarios with the same test activities but different test parameter values may further help to locate a faulty IP block in the test evaluation process.

In a preferred embodiment the plurality of test scenarios of the test sequences are chosen and/or ordered randomly in order to mimic realistic workload patterns, e.g. by the automated test equipment or by a controller.

A randomization of the test scenarios of the test sequences is further improving the test process by creating or generating realistic workload. Ideally, the test scenarios are independent of each other, that is a change in the order of the execution of the test scenarios is not expected to change the behavior of the DUT. Randomization of the order of the execution of that scenario may test, for example, whether rest heat or rest data from previous test scenarios can influence the current test scenario.

In a preferred embodiment the automated test equipment is configured to generate the test sequence, such that the test sequence configured to be executed by a controller in order to collect test data.

The generated test sequences are executed by the controller or OCST controller. The task of the controller is to trigger execution of multiple test activities and read out the test results, such as pass/fail results and/or measurement results. The controller knows which test scenario is executing with which test parameter values. It is necessary for debugging when an error occurs.

According to an embodiment, the controller is an on-chip processor, e.g. along with an operating system, or a controller card, e.g. as part of the automated test equipment configured to communicate with the automated test equipment and with the device under test.

The controller executes the test sequences on DUTs and provides measurement data or measurement results to the ATE. The on-chip processor with operating system or the controller card in the ATE communicates the measurement values or test results or collected data from the DUT to the ATE.

In a preferred embodiment the controller comprises one or more interfaces configured to read out DUT data such as memory data and/or device under test sensor data, e.g. on-chip sensor data. Accordingly, the ATE is configured to set up the controller to read-out DUT data and/or DUT sensor data.

Reading out DUT test data and/or DUT sensor data over an interface further improves the tests by allowing to compare the result data and/or result sensor data with the expected result data or expected sensor data. The controller, the OCST controller is preferably an on-chip processor along with an optional test operating system, which interfaces to on-chip IJTAG in order to read out on-chip sensor data.

According to an embodiment the controller interfaces to one or more sensors distributed over the device under test area, such that the controller is configured to read out sensor test data of the one or more sensors. Accordingly, the automated test equipment is configured to set up the controller to read out sensor test data of the one or more sensors distributed over the device under test area comprised by the controller.

Whether the controller is an on-chip processor or a OCST control card that communicates with the DUT or the ATE workstation, the controller may comprise additional sensors distributed over the die area of the DUT in order to detect local abnormalities. Applied sensors may comprise, for example, temperature, current and/or voltage sensors.

In a preferred embodiments, the controller is configured to react, for example, collect extra data, such as stored system data, program counter or memory register information, in case the collected test data fulfills a predetermined condition, such as exceeds one or more predefined thresholds. Accordingly, the automated test equipment is configured to set up the controller to react in case the collected test data fulfills predetermined conditions.

For example, a controller may interface to one or more sensors with optional thresholded alarm capabilities for a temperature value, such as current/maximum/minimum peak temperature, for a voltage value, such as current/maximum/minimum peak voltage, and/or timing violations, for example in razor circuits. The alarm capabilities can trigger, for example, storage of important system data, such as, for example, program counters, memory controller registers in order to simplify debugging.

According to an embodiment, the controller is configured to communicate the collected test data to the automated test equipment.

Whether it is test result data or measured data or any collected data triggered by measured data or by test result data, the collected test data is provided by the controller to the ATE.

According to an embodiment, the controller or the automated test equipment is configured to dynamically create test parameter values based on the test activity constraints and/or based on collected test data.

The collected test data may be analyzed by the ATE or by the controller during the test process in order to modify existing test parameter values or add one or more test scenarios to the test sequence with new test parameter values. The creation of new test parameter values may consider the constraints of the given test activity.

In a preferred embodiment, the controller or the automated test equipment is configured to analyze the collected test data in order to optimize test sequences, for example, to solving the minimum covering set problem, reduce redundancies and/or never failing test activities.

The collected test data is not only used to dynamically create test parameter values but also to optimize test sequences. A general idea of testing is to identify those test parameters that influence the occurrence of test fails. Redundant test activities may help to evaluate the effect of, for example, a single or a small group of test parameters. After identifying the most influencing test activities and/or test scenarios, the number of test scenarios or test activities in the test scenarios can be reduced to a minimum set of test activities in order to reduce the running time of the test sequence.

According to an embodiment the controller of the automated test equipment is configured to compare or calculate the correlations of results of system level tests within the predefined limitation with the collected test data, e.g. device under test data, device under test sensor data, sensor test data, within and/or outside the predefined limitation in order to optimize test time.

Tests are conducted outside the predefined limitations or specification of the DUT and are compared with the system level tests within the predefined limitations. Test activities with positive test results outside of the predefined limitations may indicate a passing system level test within the predefined limitation. Comparing the test results within the predefined limitations with test results outside of the predefined limitations may help to reduce the number of test scenarios and/or test activities within the test scenarios. It may also help to identify or decide to which extent test parameter values can be outside of the predefined limitation in order to provide good correlation between tests performed inside and outside of the predefined limitations.

In a preferred embodiment the automated test equipment comprises an artificial intelligence or a machine-learning unit, wherein the controller or the automated test equipment is configured to train the artificial intelligence or the machine-learning unit with results of system level tests and/or with the collected test data and/or with the comparison of the system level tests and the collected test data in order to optimize test sequences.

Further, in a preferred embodiment, the trained artificial intelligence or machine-learning unit is configured to predict the results of system level tests based on the collected test data.

The artificial intelligence or machine-learning unit may be trained with the result of system level tests combined with the collected test data and/or with collected sensor data to compare the system level test results with the collected test data in order to predict the results of the system level tests based on the collected test data or measurement data.

Using a prediction of the system level test results based on collected test data may further reduce the number of test scenarios and/or the number of test activities in a test scenario of the test sequence.

In a preferred embodiment, the controller or the automated test equipment is configured to analyze the collected test data in order to obtain a test result, e.g. to identify a faulty device under test and/or a faulty device under test resource, and/or to classify a device under test.

Analyzing the collected test data by, for example, using statistical methods, may have to define the faulty IP block of the DUT, or classify an IP block of the DUT.

In a preferred embodiment, the trained artificial intelligence or machine-learning unit is configured to analyze the collected test data in order to optimize test sequences and/or obtained test result.

The already trained machine-learning unit may further improve the test sequence and/or improve the statistical method to classify the IP blocks of the DUT and/or define faulty IP blocks of the DUT.

Further embodiments according to the invention create respective methods and respective computer programs.

However, it should be noted that the methods are based on the same considerations as the corresponding automated test equipments. Moreover, the methods can be supplemented by any of the features, functionalities and details which are described herein with respect to the automated test equipments, both individually and taken in combination.

Brief Description of the Figures

Embodiments according to the present application will subsequently be described taking reference to the enclosed figures, in which:

Fig. 1 shows a schematic representation of a test arrangement comprising an automated test equipment for testing one or more devices under test, according to an embodiment;

Fig. 2 shows a block diagram describing a testing process conducted by a test arrangement, according to an embodiment;

Fig. 3 shows an exemplary test activity table, input of a constraint solver, according to an embodiment;

Fig. 4 shows an exemplary resource conflict table used by a constraint solver, according to an embodiment;

Fig. 5 shows an exemplary test scenario table used by a constraint solver, according to an embodiment;

Fig. 6 shows an exemplary test step table used by a constraint solver, according to an embodiment;

Fig. 7 shows an empty test result table with an exemplary test sequence created by a constraint solver, according to an embodiment;

Fig. 8 shows an exemplary fail per test step table used for optimizing the number of test steps, according to an embodiment;

Fig. 9 shows a comparison table between the SLT testing and the OCST testing, according to an embodiment;

Fig. 10 shows a table combining the collected test data with SLT results used as a training data set for a machine learning module, according to an embodiment;

Fig. 11 shows a table combining the collected test data with OCST results used as a training data set for a machine learning module, according to an embodiment.

Detailed Description of the Embodiments

In the following, different inventive embodiments and aspects will be described. Also, further embodiments will be defined by the enclosed claims.

It should be noted that any embodiments as defined by the claims can be supplemented by any of the details, features and functionalities described herein. Also, the embodiments described herein can be used individually, and can also optionally be supplemented by any of the details, features and functionalities included in the claims. Also, it should be noted that individual aspects described herein can be used individually or in combination. Thus, details can be added to each of said individual aspects without adding details to another one of said aspects. It should also be noted that the present disclosure describes explicitly or implicitly features usable in an automated test equipment. Thus, any of the features described herein can be used in the context of an automated test equipment.

Moreover, features and functionalities disclosed herein relating to a method can also be used in an apparatus, configured to perform such functionality. Furthermore, any features and functionalities disclosed herein with respect to an apparatus can also be used in a corresponding method. In other words, the methods disclosed herein can be supplemented by any of the features and functionalities described with respect to the apparatuses.

The invention will be understood more fully from the detailed description given below and from the accompanying drawings of embodiments of the invention, which, however, should not be taken to limit the invention to the specific embodiments described, but are for explanation and understanding only.

Embodiment According to Fig. 1

Fig. 1 shows a schematic representation of a test arrangement 100 comprising an automated test equipment (ATE) 110 for testing one or more devices under test (DUT) 120. Fig. 1 further comprises an exemplary DUT 120 comprising a plurality of DUT resources 130a-e. The DUT resources 130a-e may comprise different IP blocks, such as CPU, memory, MPEG, etc.

The ATE 110 is configured to generate a plurality of test scenarios 140a-c, each comprising a plurality of test activities. For example, the test scenario 140c comprises the test activities 150a-c.

Each test activity is configured to make use of one or more DUT resources 130a-e. For example, the test activity 150a is configured to use the DUT resource 130e. Or for example, the test activity 150b is configured to use the DUT resources 130c and 130d. Another example may be the test activity 150c, which is configured to use the DUT resources 130a and 130b.

The automated test equipment 110 is configured to generate the plurality of test scenarios, such as the test scenarios 140a-c, such that the DUT resources, like the DUT resources 130a-e, associated with the plurality of test activities, like the test activities 150a-c, do not conflict with each other. For example the test activities 150a-c are grouped into the test scenario 140c, such that the DUT resources 130a-e of the test activities 150a-c are non-conflicting, allowing concurrent execution of the test activities 150a-c of the given test scenario 140c.

A general idea of on-chip-system tests (OCST) is to concurrently run large number of simple, realistic test activities on DUTs 120 using DUT resources 130a-e, wherein combinations and/or an intensities of the concurrent test activities varies. Each test activity can check involved IP blocks but also contributes to the test conditions of other IP blocks through loading of supplies, clock trees and thermal coupling. Examples of test activities may comprise moving data blocks or executing built in self tests, such as memory built in self tests (MBIST), or logic built in self tests (LBIST).

For example the OCST controller runs structural tests, such as LBIST, MBIST, etc., locally in some IP blocks. These test activities are obviously self-checking but can also serve as

stress generator to control the test conditions of other simultaneously running test activities. In test scenarios some IP cores run structural tests, while some other cores are involved e.g. in code-based test activities.

The involved IP blocks may apply design for test (DFT) structures or techniques, such as generating stress in order to sensitize faults and/or increasing observability so that sensitized faults are likely to be detected and/or providing access to existing structures for debug or for in-system tests.

A randomization of test activities or parameterizable test activities covers many combinations of local workloads and thus mimics large number of realistic work load patterns that are considered responsible for many system-level faults. Existing or new design for tests (DFT) structures can generate controlled stress and increase the observability in order to increase the likelihood of fault detection (or for more likely fault detection).

DFT structures may improve observability by, for example, the on-chip OCST controller interfaces to on-chip IJTAG in order to read out on-chip sensor data.

For example, added sensors with optional thresholded alarm capabilities for current / max peak / min peak temperature and/or for current / max peak / min peak voltage and/or timing violation, for example in razor circuits, may further improve the observability. The alarm can trigger storage or saving of important system status, such as program counters, memory controller registers in order to simplify debugging.

Observability may be further improved by extra sensors distributed over the die area to detect local anomalies. Or by connecting a processor, in order to trace memory comparing its content to expectations. Further, assertion checkers may be added, such as protocol checkers, or CRC, or a bus traffic logger to measure coverage and assist in debugging.

The test activities are orchestrated by, for example, an on-chip-system test (OCST) controller in a controlled way, so that most effective test conditions can be determined and fails can be attributed to specific test activities and their test parameter values.

Embodiments According to Fig. 2

Fig. 2 shows a block diagram 200 describing a testing process conducted by a test arrangement, similar to the test arrangement 100 of Fig. 1.

The block diagram 200 starts with a test activity table 210. The test activity table 210 comprises a column of test activities 212, wherein each test activity 212 may use one or more DUT resources 214. The corresponding DUT resources 214 is provided in a DUT resources 214 column. Further, the test activities 212 may have corresponding test parameters 216 and/or constraints 218 provided in separate columns. Constraints can be at the test scenario level and reference test parameters from multiple test activities.

The test activity table 210 is fed into the constraint solver 250. The constraint solver 250 may be comprised by an ATE 220, by a controller 270 or it may be a separate entity, as shown in Fig. 2. The constraint solver 250 of Fig. 2 has one input and one output. The input of the constraint solver 250 is fed by the test activity table 210. The output of the constraint solver is a test sequence table 260.

The test sequence table 260 comprises test scenarios 262, similar to the test scenarios 140a-c on Fig. 1. The test scenarios may comprise one or more test activities 212a-e, wherein each test activity 212a-e may comprise test parameter values 216a-e. The test sequence table 260 is provided to the ATE 220 or to the controller 270.

For example, the first scenario may comprise a first test activity 212a with the test parameters P1 216a, P2 216b and the test activity 212b with the test parameters P3 216c and P4 216d. The second scenario may comprise the second test activity 212b with the test parameters P3 216c and the test parameter P4 216d. A third scenario may comprise a third test activity 212c with the test parameters P3 216c and P5 216e. A fourth scenario may comprise a fourth test activity 212d with the test parameter P2 216b and the test parameter P4 266d.

The controller 270 takes the test sequence table 260 as input and outputs a test result table 280. The test result table 280 provided by the test block 270 comprises the test results 288 of the one or more test activities 212 of the test scenarios 262 executed on the DUTs 282. The test result table 280 is fed into the ATE 220 and/or or back into the controller 270.

The ATE 220 or the controller 270 has accepts the test result table 280 as input and provides an improved test sequence table 260 and/or a result table 292 as an output.

The improved test sequence table 260 may further be used as input of the controller 270 to provide a new test result table 280 which can be fed back into the ATE 220 or into the controller 270.

The result table 292 provided by the ATE 220 or controller 270 comprises pass/fail test results 298 of the DUT resources 296. Additionally and/or alternatively the result table 292 may comprise a classification of the DUT resources 296.

The test activity table 210 with constraints 218, test parameters 216 and resources 214 required by the test activities 212 are provided to the constraint solver 250. The test activity table 210 or a library of test activities may further comprise a pool of code for the controller 270 or the OCST controller 270 to activate or execute test activities 212. The library may also know which DUT or on-chip and ATE resources are used by a given test activity 212.

The constraint solver 250 is configured to create a test sequence table 260 from the test activity table 210. The test sequence table 260 comprises test scenarios 262, wherein test scenarios 262 comprise one or more test activities 212a-e, which can coexist or can be executed concurrently without violating resource constraints 218. The test scenarios 262 are generated automatically by the constraint solver 250 running on the work station and/or on the controller 270, such as an OCST card or an on-chip OCST controller. The constraints may be modelled in PSS.

The test sequence table 260 provided by the constraint solver 250 comprises the scenarios 262, wherein one or more test activities 212a-d associated with one or more test parameters 216a-e may be executed concurrently. The test parameter values characterizing the respective test parameters 216a-e are chosen randomly or so as to have an emphasis on extreme values. The order of the test scenarios 262 is, for example, generated randomly, in order to simulate real life workload.

The generated test sequence table 260 is provided to the controller 270, such as an on-chip OCST controller, which is configured to execute the test scenarios 262 of the test sequence table 260, in order to collect test data 280. The controller 270 may comprise interfaces to communicate with the ATE and/or interfaces to communicate with the DUT.

The controller 270 may read out DUT test data and/or DUT sensor data and/or the controller may comprise sensors over the DUT area or die area to detect local anomalies. The measured values or collected data 280 may trigger the controller 270 to collect further information such as memory information or status information of the DUT, if the measurement values and/or the sensor values fulfil certain predefined conditions.

The controller 270 is configured to communicate the collected test data 280 or test results 288 of test activities 212 of the scenarios 262 to the ATE 220. The ATE 220 or the controller 270 is configured to further improve the testing process and/or debug or diagnose the DUT.

The controller 270, such as an on-chip OCST controller or an OCST card, or an ATE 220 is configured to dynamically create or modify the set of test parameters of the test scenarios that may meet constraints and may allow to know current test parameters for debugging. Methods to create test parameter value sets for test activities 212 include randomization, which follows a desired distribution which optional emphasis on extreme values, using, for example, a constraint solver to maximize coverage, or using, e.g. nested loops for an exhaustive coverage of a few test parameters.

In order to further improve the testing environment a test learning environment may be required. Extensive characterization tests are the basis for the test learning, wherein many DUTs are exposed to many test steps or to many test scenarios. Preferably, not all DUTs 282 are exposed to the same test steps or test scenarios, in order to cover a large amount of combinations of test parameters 216. In order to avoid bias of test scenarios towards certain DUTs 282, test steps or test scenarios are executed in random permutation order.

The ATE 220 or the controller 270 may use machine learning or AI module as well, trained with the collected test result data 280 and the system level test results. The machine learning module may analyze the collected test result data 280 and the system level test result and may predict system level test results based on a new set of collected test data 280.

The AI module may further be trained by measured result data without specification limits, such as on-chip sensor data or by test parameters outside specification limits, such as too low voltages or too fast frequencies, in order to predict system level test fails from the test results. Failing test steps or test activities outside specification limits is not an evidence for bad DUTs.

However, machine learning modules or models can be constructed to predict system level test results from test step or test activity result, including measured results without or outside specification limits and including test steps or scenarios with out-of-specification test parameters and test steps or scenarios related properties, such as involved test activities and test resources.

Such model may likely find some of the previously escaped SLT fails, but may also fail some DUTs that pass SLT and pass all other legitimate OCST tests. These cases may be considered yield loss through test and may be traded carefully against additionally found SLT fails, preferably based on a cost model. Only those additional test steps or scenarios may be included in production steps that are needed by such models. Other additional test steps can be removed again.

The ATE 220 or the controller 270 may further configured to debug and/or diagnose the DUT. As the test result table 280 comprises results 288 of concurrently executed test activities 212 of test scenarios 262, there is a need for further analyses of the collected test data 280 in order to identify and/or classify faulty DUTs or DUT resources.

The general idea for debugging or diagnosing is to identify those test parameters 216 that influence the occurrence of OCST fails most. Test parameters 216, associated with test activities 212 that involve certain actions in specific IP blocks, are providing informative hints for debugging.

A machine learning module trained by a table combining test steps or scenarios with test activities, DUT resources, test parameters, test results and overall OCST result, optionally for multiple DUTs may be used for classifying or identifying faulty DUT resources. The machine learning module or machine learning feature selection algorithm may identify which test activities, test or DUT resources and test results are important to explain OCST results that contribute to the occurrence of OCST fails.

In other words, the controller 270 is controlling the test process. The controller, or OCST controller is preferably an on-chip processor along with an optional test operating system, but can also be an OCST card that communicates with the DUT or the ATE workstation.

A task of the OCST controller may be, for example, to trigger execution of multiple test activities and read-out their pass/fail results and/or measurement results. The test activities may include a combination of optional stress generation and/or optional fault detection.

Further examples of test activities are listed below:

- The ATE sets external test conditions of the DUT, e.g. DUT supply voltage or frequency, i.e. the OCST controller can control ATE resources.
- ATE performs measurements, e.g. supply current measurements.
- Moving data block between IP cores and checking the content before and/or afterwards.
- Running memory tests on on-chip CPUs.
- Running memory self-tests.
- Compressing and decompressing images and checking the differences between the original and the decompressed image. (stress & check)
- Running I/O loopback tests.
- Applying stress generation, using DFT techniques.
- Activating and reading-out of any of the observability structures, using DFT techniques.

The controller always knows which test scenario is executing with which test (activity) parameters, which is necessary for debugging when errors occur.
Further, the controller or the OCST controller could dynamically create or modify test activity parameters based on constraints, and/or could work or import them from precomputed lists. Also, the controller, or the processor running the OCST controller code could also generate test activities.

Exemplary Test Activity Table According to Fig. 3

Fig. 3 shows an exemplary test activity table 300 according to an embodiment. The test activity table 300 is similar to or an example of the test activity table 210 of Fig. 2.

The test activity table 300 comprises columns of test activities, resources, test parameters and possible results. The test activity table is configured to list all possible test to be executed on a DUT with the required resourced, adjustable parameters and possible outcomes or results.

The test activity table 300 may be used by a constraint solver, such as the constraint solver 250 of Fig. 2, as input in order to create test sequences. A test sequence may define which test activity with which test parameters are executed on which DUT in what order.

The test activity table 300 may comprise common test activities or DUT specific test activities. Some examples of test activities are contained by the exemplary test activity table 300.

For example, a first test activity A1 may comprises a processing unit 2 (CPU2) writing data to memory 3 (MEM3) and checking the content of MEM3. The activity A1 requires the resources R1: CPU2, R2: MEM3, R3: ATE DPS for core supply. The adjustable test parameters of test activity A1 is P1: bandwidth, P2: DPS voltage. Results may include two values r1 which is a fail/pass value and r2 which is an electrical current value.

For example, a test activity A2 is a memory built in self-test (MBIST) of the MEM3, which requires the resource R2, that is MEM3, without any adjustable parameters and with an outcome of a pass/fail value as a result.

For example, a test activity A3 is an MPEG self-test, requiring an MPEG resource having adjustable block sizes as test parameter P3, with an outcome of a pass/fail value.

Resource Conflict Table According to Fig. 4

Fig. 4 shows a resource conflict table 400 which might be used and/or created by the constraint solver 250 of Fig. 2. The resource conflict table 400 has a test activities column and a column for every resource of the DUT. Each test activity of the table 400 uses one or more DUT test resources, which is represented by an 'X' in the respective resource column.

A constraint solver, like the constraint solver 250 of Fig. 2 may take a test activity table, like table 300 of Fig. 3 as an input in order to create a test sequence. A step of creating a test sequence is to create a resource conflict table, such as a resource conflict table 400. The resource conflict table 400 shows which test activities are using the same DUT resources and therefore which test activities cannot run concurrently.

For example, the resource conflict table 400 show conflicting resources of the test activities of the test activity table 300. For example, test activity A1 uses the resources R1, R2, R3 and does not use R4. For example, the test activity A2 uses only R2 as a resource. For example, the test activity A3 uses the DUT resource R4.

As the resource conflict table 400 shows, the test activity A1 and the test activity A2 are conflicting test activities because both of them requires the test resource R2. Test activities A1 and A2 both require test resource R2 and can therefore not be run concurrently. That is, the test activity A1 and test activity A2 cannot run concurrently, e.g., cannot be put in the same test scenario. The test activities without resource conflicts can be combined to test scenarios.

Test Scenario Table According to Fig. 5

Fig. 5 shows a test scenario table 500 which might be used or created by a constraints solver 250 of Fig. 2. The test scenario table 500 comprises a test scenario column and a column for each test activity provided. The test scenario table 500 comprises all the possible test scenarios. The one or more test activities of the test scenarios are run concurrently.

The test scenario table 500 shows all the test scenarios which might be created from the test activities, like the test activities A1, A2 and A3 of the test activity table 300 of Fig 3. The test activities of a test scenario are required to be non-conflicting. Conflicting test activities are shown in an resource conflict table, similar to the resource conflict table 400 of Fig. 4. Excluding the test scenarios with conflicting test activities, taken from the resource conflict table, from all the possible test scenarios results in the test scenario table with non-conflicting test activities, such as the test scenario table 500.

For example, the test scenario table 500 comprises the test scenario column and a column for every test activity, such as A1, A2, A3. According to the resource conflict table 400 of Fig. 4, A1 and A2 use the same resource R2, therefore the test activity A1 and the test activity A2 cannot run concurrently and cannot be in the same test scenario.

For example, the test scenarios of the exemplary test activities of Fig. 3 are the test scenario S1 with the test activity A1, the test scenario S2 with the test activity A2, the test scenario S3 with the test activity A3, the test scenario S4 with the concurrent test activities A1, A3,

and the test scenario S5 with the concurrent test activities A2, A3. Due to resource constraints, no test scenario runs test activities A1 and A2 concurrently.

Whether test activities are really run concurrently can depend on test parameter settings and other unknown factors. A test sequence or a test suit consists of multiple test steps that execute a given test scenario with specified test parameter values for their test activities.

Test Step Table According to Fig. 6

Fig. 6 shows a test step table 600 which can be used or created by a constraint solver 250 of Fig. 2. The test step table comprise columns of test steps, test scenarios, test activities and columns for test parameters corresponding to test activities.

The test step column of the test step table 600 comprises a running number in order to identify the different test steps. The test scenario column may comprise all the possible test scenarios at least once. The test scenarios may be tested more than once with several different test parameters. Similar to the test scenario table 500 of Fig. 5, the test activities of the test scenarios are represented by an 'X' in the corresponding test activity column.

If the test scenario comprises test activities, the corresponding test parameter columns comprise test parameter values. Test parameter values are preferably generated randomly, optionally following a predefined distribution and/or optionally with a certain percentage concentrating on extreme values that tend to provoke more problems than random generated test parameter values.

A test sequence, such as the test sequence table 260 of Fig. 2 is generated from the test step table 600, by mapping test steps of the test step table 600 to DUTs randomly or following a predefined distribution in order to provoke more problems than a randomly mapped test sequence.

The test scenario column comprises all the possible test scenarios, such as the test scenarios S1 to S5 from the test scenario table 500, wherein the test scenarios may be tested with several different test parameters.

For example, a first test step might be the first test scenario S1 with a test activity A1, corresponding with the test parameters P1, bandwidth of 10 GB/S, and test parameters P2,

DPS voltage of 0.9 V. For example, a test step 2 comprises the same test scenario S1 with different test parameters P1, a bandwidth of 20 GB/S, and P2, a DPS voltage of 0.86 V.

For example, a test step 3 may comprise the test scenario S2 which comprise a test activity A2 without any adjustable test parameters.

For example, a test step 4 comprises the test scenario S3 with a test activity A3 wherein the test parameter P3 is a block size of 128 kB. For example, a test step 5 comprises the same scenario S3 with a test parameter P3 of a block size of 1 MB.

For example, a test step 6 comprises the test scenario S4 with the test activities A1 and A3, with the test parameters of P1, a bandwidth of 50 GB/S, P2, DPS voltage of 1.04 V and a test parameter P3, a block size of 6 MB. For example, a step 7 comprises the same scenario S4 with the test parameter of P3, a bandwidth of 3 GB/S, P2, a DPS voltage of 0.97 V and a P3 a block size of 500 KB. For example, a step 8 comprises the same scenario S4 with the test parameters P1 a bandwidth of 27 GB/S, P2 a DPS voltage of 0.88 V and P3 a block size of 21 MB.

For example, a test step 9 comprises a test scenario of S5 with a test activity A2 without test parameters and the test activity A3 with a test parameter of P3, a block size of 7 MB. For example, a test step 10 comprises the same test scenario as S5 with a test parameter of P3 a block size of 780 KB. For example, a step 11 comprises the same test scenario as S5 with a test parameter of P3 a block size of 13 MB.

Test Result Table According to Fig. 7

Fig. 7 shows an empty test result table 700 which might be filled out by the controller after conducting test activities of the test sequence table 260 of Fig. 2. The test result table comprise columns of DUTs, test steps, test results of test activities, and an overall test result column.

The test result table 700 comprise the test sequence, such as the test sequence table 260 of Fig. 2. The first two columns of the test result table 700, the DUT and the test step column, define which test step is performed on which DUT. The test sequence is generated from the test step table 600 of Fig 6, by mapping test steps of the test step table 600 to DUTs

randomly or following a predefined distribution and/or fulfilling predefined conditions in order to provoke more problems than the randomly mapped test sequence.

An exemplary test sequence may comprise, for example, a DUT1 tested by the test steps 6, 11, 7, 1 and 8, and a DUT2 tested by the test steps of 5, 2, 10, 4 and 9.

The controller, such as an on-chip controller or a controller card performs the tests according to the test sequence. The results of the test activities, represented by the columns r1 (#fails), r2 (current), r3 (#fails), r4 (p/f) are collected by the controller. The overall OCST result may be based on all of the results of the test activities or on a subset of the test activity results. Additionally or alternatively the OCST result may comprise a classification of the DUT based on the test activity results or some of the test activity results.

In this example, the overall OCST result is calculated from the results r1, r3 and r4, that is, the measured result r2 does not contribute to the over test result because it has no specification limits in this example.

Fail Per Test Step Table According to Fig. 8

Fig. 8 shows a fail per test step table 800 comprising a DUT column and columns for every test steps. The tested DUTs are listed in the DUT column. A row of a tested DUT comprises a letter 'P' in a given test step column, if the DUT has passed the given test step and a letter 'F' with a highlighted background if the DUT has failed the given test step.

The controller or the ATE is configured to use the fail per test step table 800 for optimize, preferably reduce, the number of test steps. For example, the number of never failing tests and/or redundant test steps may be reduced. The task of the controller or the ATE may comprise selecting a minimum set of test steps, which is known as the minimum covering set problem, which has a known solution.

For example, in the case of the fail per test step table 800 may comprise the test results of four DUTs, DUT 1-4. For example, DUT 1 may pass all the test steps except test step 6. For example, DUT 2 may pass all the test steps except the test steps 2 and 4. For example, DUT 3 may pass all the test steps except the test steps 1 and 6. For example, DUT 4 may pass all the test steps except the test step 8.

For example, as the controller or the ATE analyze the fail per test step table 800, the controller may come to the conclusion that the test steps 3, 5 and 7 never fail and can therefore be removed from the production tests. Further, test step 4 is redundant to test step 2 and may also be removed.

Comparing OCST with SLT According to Fig. 9

Fig. 9 shows an exemplary comparison table 900 between the OCST test results and the SLT test results. The exemplary comparison table 900 shows differences that may occur when testing the same set of DUTs with both OCST and SLT testing methods.

In this example, 9900 devices passed and 70 failed the both of the test methods, OCST and SLT. There were 25 devices or DUTs failing the OCST test, but passing the SLT test, while there were only 5 DUTs passing the OCST test and failing the SLT test.

In other words, in the above example, OCST misses 5 devices that fail SLT, but finds problems in 25 DUT that SLT does not find, which is a good balance assuming all test steps describe realistic scenarios with realistic test parameter values.

Training Table Combining Test Results with SLT Results According to Fig. 10

Fig. 10 shows a training table 1000 combining the collected test data with the SLT results. The training table 1000 is configured to be used as a training table or training data set for a machine learning or AI module. The training table 1000 may comprise all the test activities, test resources and test results of test steps conducted on DUTs with corresponding SLT results. The training table 1000 may comprise multiple test steps executed on multiple DUTs.

The training table 1000 may comprise test data collected outside, or slightly outside, of the specification of the DUTs combined with SLT results, wherein the SLT tests were conducted within the specification of the DUTs. For example, in this case, a failing OCST result is not considered a strong sign of a failing DUT, but a passed OCST test outside the specification limits may considered as a strong sign of a good DUT and may result in classifying the DUT as a high quality DUT.

The ATE or the controller may comprise a machine learning unit or AI module which may be trained by the training table 1000 and may be configured to predict the SLT result from a newly conducted test step on a DUT. The machine learning unit may be used for improving the test process as well.

Training Table with Test Results According to Fig. 11

Fig. 11 shows a training table 1100 comprising the collected test data and the overall OCST results. The training table 1100 is configured to be used as a training table or training data set for a machine learning or AI module. The training table 1100 may comprise test activities, test resources and test results of test steps conducted on DUTs with corresponding overall OCST results.

The table 1100 is used for debugging and diagnosis purposes. A general idea of debugging is to identify those test parameters that influence the occurrence of OCST fails most. The ATE or the controller may comprise the AI or machine learning unit in order to identify and/or classify frequently failing DUT resources or DUTs. For efficiency reasons the table might be restricted to failing devices. The analyzers can be done across many DUTs to identify frequently occurring fault mechanisms.

The machine learning unit or module may be trained by the test result table 1100 in order to predict the failing DUT, or DUT resources.

Implementation alternatives

Although some aspects have been described in the context of an apparatus, it is clear that these aspects also represent a description of the corresponding method, where a block or device corresponds to a method step or a feature of a method step. Analogously, aspects described in the context of a method step also represent a description of a corresponding block or item or feature of a corresponding apparatus.

Depending on certain implementation requirements, embodiments of the invention can be implemented in hardware or in software. The implementation can be performed using a digital storage medium, for example a floppy disk, a DVD, a CD, a ROM, a PROM, an EPROM, an EEPROM or a FLASH memory, having electronically readable control signals

stored thereon, which cooperate (or are capable of cooperating) with a programmable computer system such that the respective method is performed.

Some embodiments according to the invention comprise a data carrier having electronically readable control signals, which are capable of cooperating with a programmable computer system, such that one of the methods described herein is performed.

Generally, embodiments of the present invention can be implemented as a computer program product with a program code, the program code being operative for performing one of the methods when the computer program product runs on a computer. The program code can for example be stored on a machine readable carrier.

Other embodiments comprise the computer program for performing one of the methods described herein, stored on a machine readable carrier.

In other words, an embodiment of the inventive method is, therefore, a computer program having a program code for performing one of the methods described herein, when the computer program runs on a computer.

A further embodiment of the inventive methods is, therefore, a data carrier (or a digital storage medium, or a computer-readable medium) comprising, recorded thereon, the computer program for performing one of the methods described herein. The data carrier, the digital storage medium or the recorded medium are typically tangible and/or non–transitionary.

A further embodiment of the inventive method is, therefore, a data stream or a sequence of signals representing the computer program for performing one of the methods described herein. The data stream or the sequence of signals can for example be configured to be transferred via a data communication connection, for example via the internet.

A further embodiment comprises a processing means, for example a computer, or a programmable logic device, configured to or adapted to perform one of the methods described herein.

A further embodiment comprises a computer having installed thereon the computer program for performing one of the methods described herein.

A further embodiment according to the invention comprises an apparatus or a system configured to transfer (for example, electronically or optically) a computer program for performing one of the methods described herein to a receiver. The receiver can, for example, be a computer, a mobile device, a memory device or the like. The apparatus or system can, for example, comprise a file server for transferring the computer program to the receiver.

In some embodiments, a programmable logic device (for example a field programmable gate array) can be used to perform some or all of the functionalities of the methods described herein. In some embodiments, a field programmable gate array can cooperate with a microprocessor in order to perform one of the methods described herein. Generally, the methods are preferably performed by any hardware apparatus.

The apparatus described herein can be implemented using a hardware apparatus, or using a computer, or using a combination of a hardware apparatus and a computer.

The apparatus described herein, or any components of the apparatus described herein, can be implemented at least partially in hardware and/or in software.

The methods described herein can be performed using a hardware apparatus, or using a computer, or using a combination of a hardware apparatus and a computer.

Claims

1. An automated test equipment (110, 220) for testing one or more devices under test (120, 282),

wherein the automated test equipment is configured to generate one or more test scenarios (140a-c, 262) comprising a plurality of test activities (150a-c, 212, 212a-d);

wherein different test activities make use of subsets of the device under test resources(130a-e, 214);

wherein the automated test equipment is configured to generate the plurality of test scenarios

such that resources of the device under test associated with a plurality of test activities of a test scenario do not conflict with each other.

2. The automated test equipment according to claim 1, wherein the plurality of test activities of a given test scenario are configured to be executed concurrently.

3. The automated test equipment according to claim 1 or 2, wherein one or more of the test activities are associated with one or more test parameters (216, 216a-e) characterized by respective test parameter values and/or are associated with one or more constraints (218).

4. The automated test equipment according to one of the previous claims, wherein one or more of the test activities are characterized by one or more test parameter values outside predefined limitations.

5. The automated test equipment according to one of the previous claims, wherein the plurality of test activities of a test scenario are chosen randomly, under the constraints that resources of the device under test associated with the plurality of test activities of the test scenario do not conflict with each other.

6. The automated test equipment according to one of the previous claims, wherein the automated test equipment comprises a constraint solver (250) for generating test scenarios in order to prevent occurrences of conflicts between the resources of the device under test.

7. The automated test equipment according to one of the previous claims, wherein one or more of the test activities comprise activating a stress generator arranged on the device under test.

8. The automated test equipment according to one of the previous claims, wherein the automated test equipment is configured to generate a test sequence of test scenarios.

9. The automated test equipment according to claim 8, wherein the test sequence comprises two or more test scenarios with the same test activities, wherein the test scenarios differ by at least one test parameter value.

10. The automated test equipment according to claim 8 or 9, wherein the plurality of test scenarios of the test sequence are chosen and/or ordered randomly.

11. The automated test equipment according to one of the claims 7 to 10, wherein the automated test equipment is configured to generate the test sequence, such that the test sequence is configured to be executed by a controller (270) in order to collect test data.

12. The automated test equipment according to claim 11, wherein the controller is an on-chip processor or a controller card configured to communicate with the automated test equipment and with the device under test.

13. The automated test equipment according to claim 11 or 12, wherein the controller comprises one or more interfaces configured to read out device under test data and/or device under test sensor data.

14. The automated test equipment according to one of the claims 11 to 13, wherein the controller comprises one or more sensors distributed over the device under test area, such that the controller is configured to read out sensor test data of the one or more sensors.

15. The automated test equipment according to one of the claims 11 to 14, wherein the controller is configured to react in case of the collected test data (280) fulfills a predetermined condition.

16. The automated test equipment according to one of the claims 11 to 15, wherein the controller is configured to communicate the collected test data to the automated test equipment.

17. The automated test equipment according to one of the claims 11 to 16, wherein the controller or the automated test equipment is configured to dynamically create test parameter values based on the test activity constraints and/or based on collected test data.

18. The automated test equipment according to one of the claims 11 to 17, wherein the controller or the automated test equipment is configured to analyze the collected test data in order to optimize test sequences.

19. The automated test equipment according to one of the claims 11 to 18, wherein the controller or the automated test equipment is configured to compare results of system level tests within the predefined limitations with the collected test data within and/or outside the predefined limitations in order to optimize test sequences.

20. The automated test equipment according to one of the claims 11 to 19, wherein the automated test equipment comprises an artificial intelligence or a machine learning unit,

wherein the controller or the automated test equipment is configured to train the artificial intelligence or the machine learning unit with results of system level tests and/or with the collected test data and/or with the comparison of the system level tests and the collected test data in order to optimize test sequences.

21. The automated test equipment according to claim 20, wherein the trained artificial intelligence or machine learning unit is configured to predict the results of system level tests based on the collected test data.

22. The automated test equipment according to one of the claims 11 to 21, wherein the controller or the automated test equipment is configured to analyze the collected test data in order to obtain a test result.

23. The automated test equipment according to one of the claims 17 to 22, wherein the trained artificial intelligence or machine learning unit is configured to

analyze the collected test data in order to optimize test sequences and/or to obtain test results.

24. A process for testing one or more devices under test by an automated test equipment,

wherein the automated test equipment generates one or more test scenarios comprising a plurality of test activities;

wherein different test activities make use of subsets of the device under test resources;

wherein the automated test equipment generates the plurality of test scenarios such that resources of the device under test associated with a plurality of test activities of a test scenario do not conflict with each other.

25. A computer program for implementing the method of claim 24 when being executed on a computer or signal processor.

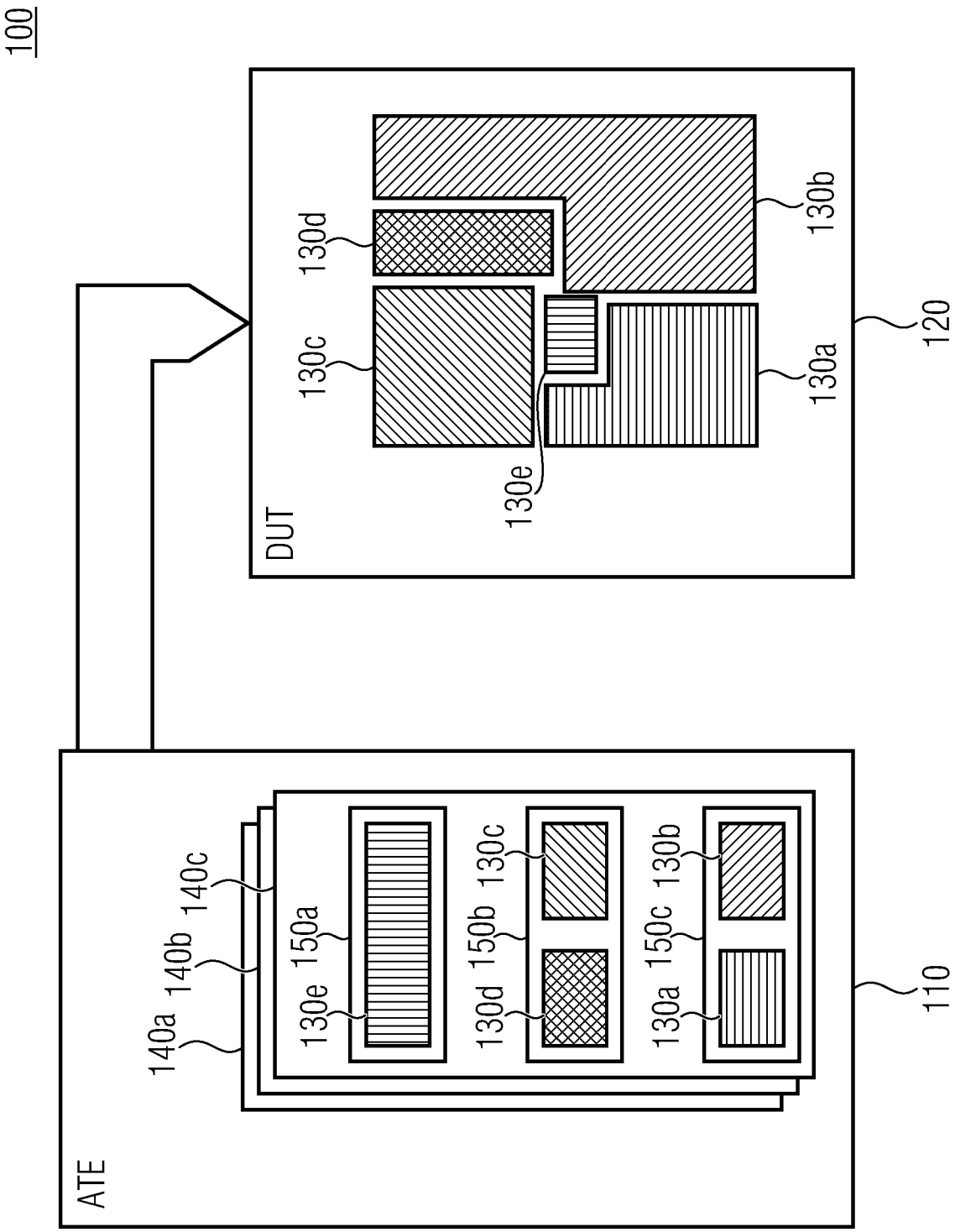
100
DUT
130d
130b
130c
130a
130e
120
ATE
140a
140b
140c
150a
150b
150c
130e
130d
130a
130c
130b
110

Fig. 1

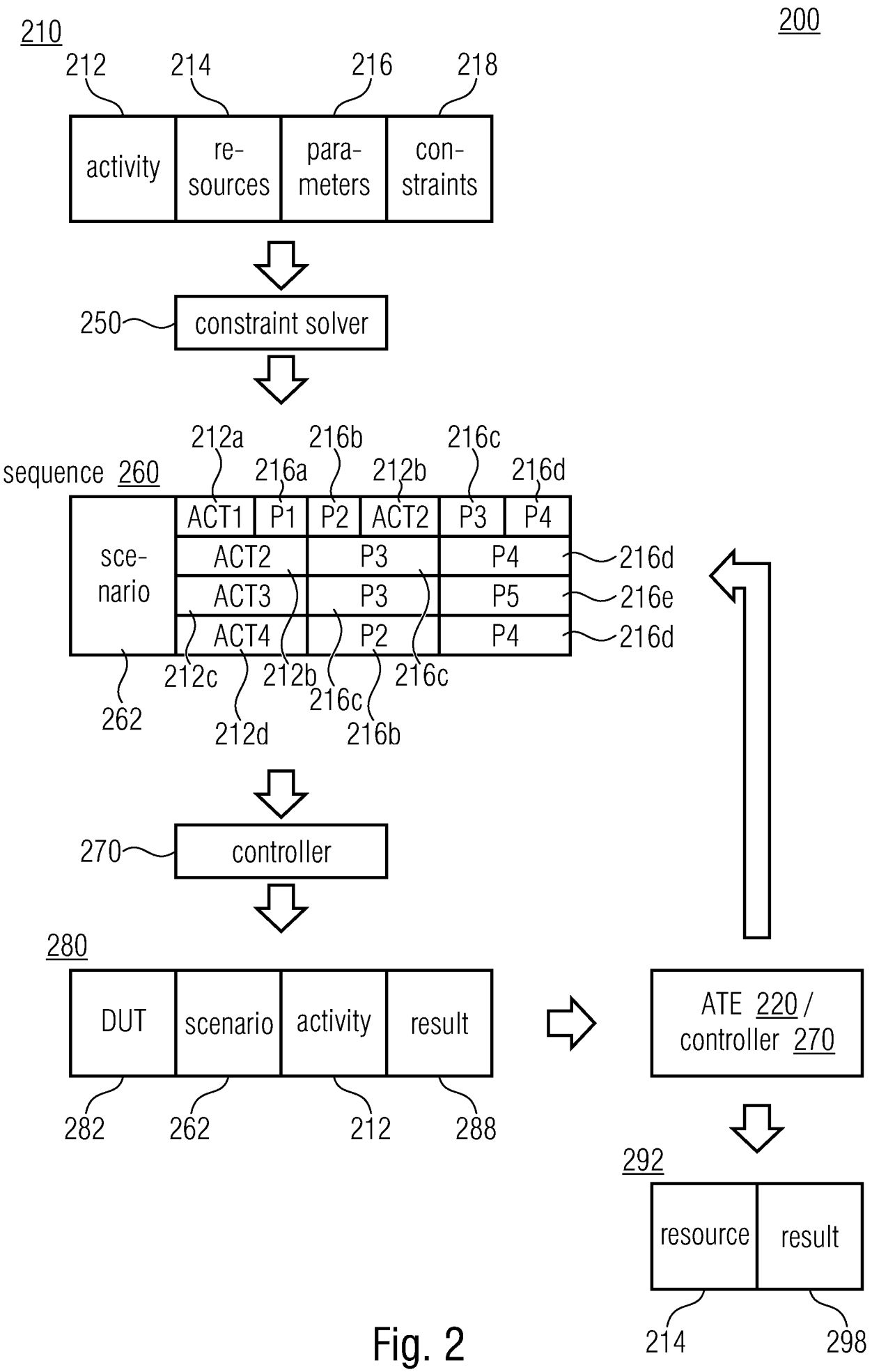
200
210
212
214
216
218
activity
re-sources
para-meters
con-straints
250
constraint solver
sequence 260
212a
216b
216c
216a
212b
216d
ACT1
P1
P2
ACT2
P3
P4
sce-nario
ACT2
P3
P4
216d
ACT3
P3
P5
216e
ACT4
P2
P4
216d
212c
212b
216c
216c
262
212d
216b
270
controller
280
DUT
scenario
activity
result
282
262
212
288
ATE 220 / controller 270
292
resource
result
214
298

Fig. 2

300

test activity	resources (IP cores, ATE)	parameters	results
A1: CPU2 writes data to Mem3 and checks content	R1: CPU2 R2: Mem3 R3: ATE DPS for core supply	P1: bandwidth P2: DPS voltage	r1: #fail r2: current
A2: MBIST of Mem3	R2: Mem3		r3: #fail
A3: MPEG self-test	R4: MPEG	P3: blocksize	r4: pass/fail
•••			

Fig. 3

400

test activity	test resources (ATE, on-chip)				
	R1	R2	R3	R4	•••
A1	x	x	x		
A2		x			
A3				x	
•••					

Fig. 4

500

test scenario	test activities		
	A1	A2	A3
S1	x		
S2		x	
S3			x
S4	x		x
S5		x	x

Fig. 5

600

test step	test scena-rio	test activities and parameters						
		A1	P1 band-width	P2 DPS volt.	A2	A3	P3 block-size	•••
1	S1	x	10 GB/s	0.9 V				
2	S1	x	20 GB/s	0.86 V				
3	S2				x			
4	S3					x	128 kb	
5	S3					x	1 MB	
6	S4	x	50 GB/s	1.04 V		x	6 MB	
7	S4	x	3 GB/s	0.97 V		x	500 kb	
8	S4	x	27 GB/s	0.88 V		x	21 MB	
9	S5				x	x	7 MB	
10	S5				x	x	780 kb	
11	S5				x	x	13 MB	
•••	•••							

Fig. 6

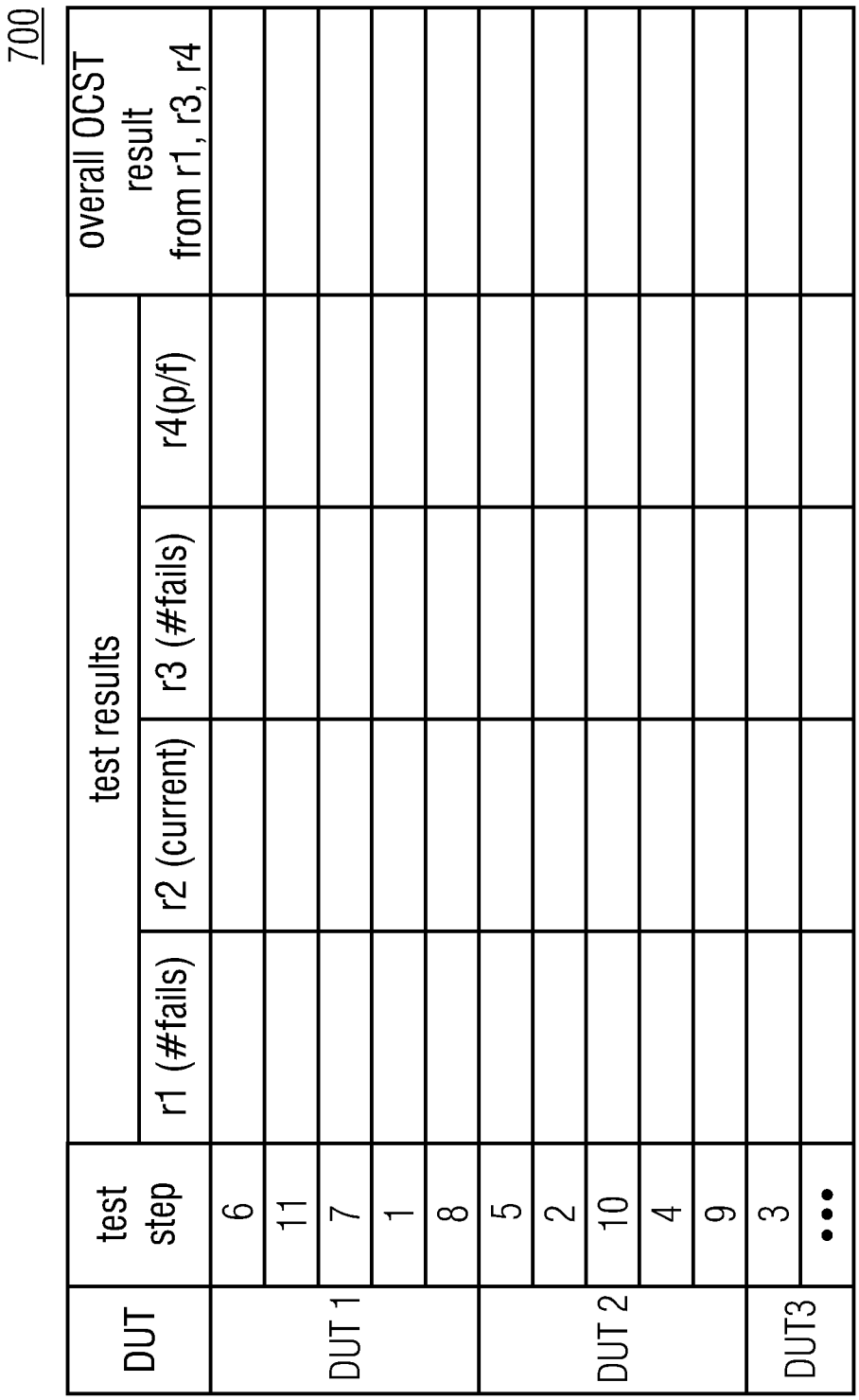
700
DUT
test step
test results
r1 (#fails)
r2 (current)
r3 (#fails)
r4(p/f)
overall OCST result from r1, r3, r4
DUT 1
6
11
7
1
8
DUT 2
5
2
10
4
9
DUT3
3
...

Fig. 7

800

DUT	fail per test step								
	1	2	3	4	5	6	7	8	•••
DUT1	P	P	P	P	P	F	P	P	
DUT2	P	F	P	F	P	P	P	P	
DUT3	F	P	P	P	P	F	P	P	
DUT4	P	P	P	P	P	P	P	F	

Fig. 8

900

		SLT	
		pass	fail
OCST suite	pass	9900	5
	fail	25	70

Fig. 9

1000

DUT	test step	test activities			test resources				test results				SLT result
		A1	A2	A3	R1	R2	R3	R4	r1	r2	r3	r4	
DUT 1	6	x		x	x	x	x	x	0	0.8V	0	0	fail
•••													

Fig. 10

1100

DUT	test step	test activities			test resources				test results				OCST result
		A1	A2	A3	R1	R2	R3	R4	r1	r2	r3	r4	
DUT 1	6	x		x	x	x	x	x	0	0.8V	0	0	fail
•••													

Fig. 11

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2020/070599

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F11/263 G01R31/3185
ADD.

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 G01R

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)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2005/235263 A1 (BUNDY LAURA M [US] ET AL) 20 October 2005 (2005-10-20) abstract; figure 6 paragraphs [0001] - [0005], [0008], [0019] - [0022], [0026] - [0030], [0032], [0037] - [0040], [0058] - [0062], [0067], [0068], [0074]	1-25
A	US 2015/253384 A1 (CHEN HARRY HAI [TW] ET AL) 10 September 2015 (2015-09-10) the whole document	1-25

☐ 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 application or patent 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
3 May 2021	14/05/2021

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Leuridan, Koen

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2020/070599

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2005235263 A1	20-10-2005	TW I364704 B	21-05-2012
		US 2005235263 A1	20-10-2005
US 2015253384 A1	10-09-2015	CN 104898042 A	09-09-2015
		US 2015253384 A1	10-09-2015
		US 2016377678 A1	29-12-2016