Understanding Boundary Scan

PCB Testing with IEEE 1149.1

Set of Laboratory Works

www.testonica.com

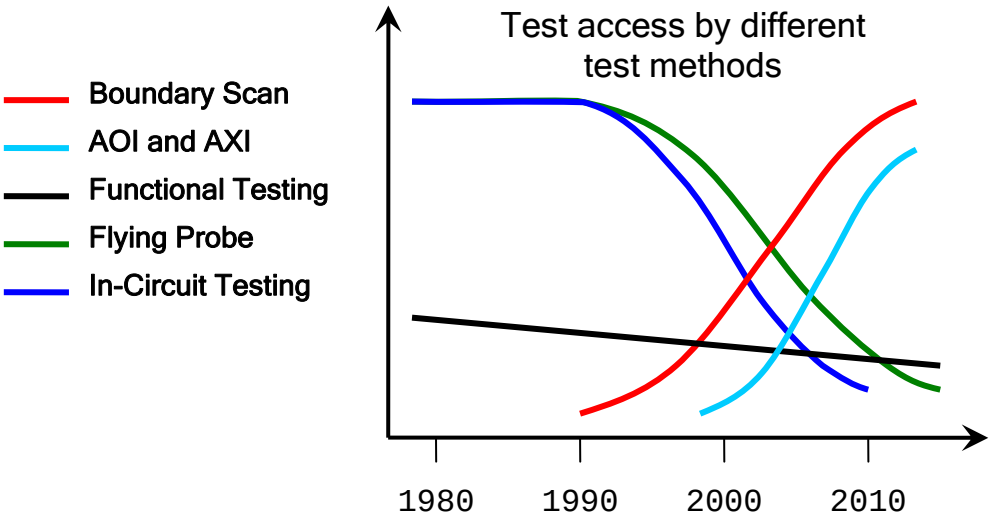
Glossary

AOI – Automatic Optical Inspection
AXI - Automatic X-Ray Inspection
BC – Boundary (Scan) Cell
BGA – Ball Grid Array
BIST - Built-In Self-Test
BS – Boundary Scan
BSDL – Boundary Scan Description Language
BST – Boundary Scan Testing
DFT – Design for Testability
DUT – Device under Test
ICT – In-Circuit Testing
IEEE - Institute of Electrical and Electronics Engineers
JTAG – Joint Test Action Group
PCB – Printed Circuit Board
PGA – Pin Grid Array
TAP – Test Access Port
TCK – Test Clock
TDI – Test Data In
TDO – Test Data Out
TMS – Test Mode Select
TRST – Test Reset
VHDL – VHSIC Hardware Description Language
VHSIC - Very-High-Speed Integrated Circuit

Introduction and Motivation

The printed circuit board (PCB) testing constitutes an essential step of the production cycle of microelectronic systems as it is an important instrument to ensure the product quality level. The state of the art of PCB testing is a mixture of Boundary Scan (BS), optical and x-ray inspection, functional and in-circuit testing.

Historically, most PCB testing was done using so-called *bed-of-nails* in-circuit test (ICT) equipment, which provides the physical contact to any desired point of the PCB surface. This method is very efficient when used with PCBs of a low complexity. However, an average modern PCB, normally, contains several inaccessible internal layers inside and complex components upon the surface. Modern fine pitch high count packaging types, like e.g. various grid arrays (PGA or BGA), can hold up to several hundreds of hidden pins. Since each such pin has to be tested, then complete in-circuit as well as functional testing becomes too expensive or even infeasible due to extremely reduced test access.



As the result, over the last decade, there was a considerable increase in the number of test applications that require Boundary Scan as well as optical/x-ray inspection (AOI/AXI) techniques. Although, no PCB testing method taken alone can guarantee full test coverage, the BS seems to be the most universal and the only realistic low-cost solution that supports structured approach and quantitative fault coverage measure.

Boundary scan allows complete controllability and observability of the boundary pins of a BS-compatible device via software control. In the field of board assembly testing, BS targets such faults as bad soldering, wrong component placement, component misplacement, broken interconnects, defective component legs, shorts between conductive lines, and others. At the component testing, BS facilitates isolation and testing of chips either via a test bus or by built-in self-test (BIST) hardware. Hence, boundary scan covers a large part of all structural faults in the system under test and substantially reduces both test equipment and DFT costs.

The widespread adoption of BS reflects an industry-wide need to simplify and structure complex tasks of testing PCBs and systems for manufacturing defects and performing other design debug, configuration, and maintenance tasks. Nowadays, BS is well supported by dedicated hardware instruments and software tools enabling a simple and standard means of automatically creating and applying tests at the device, board, and system levels.

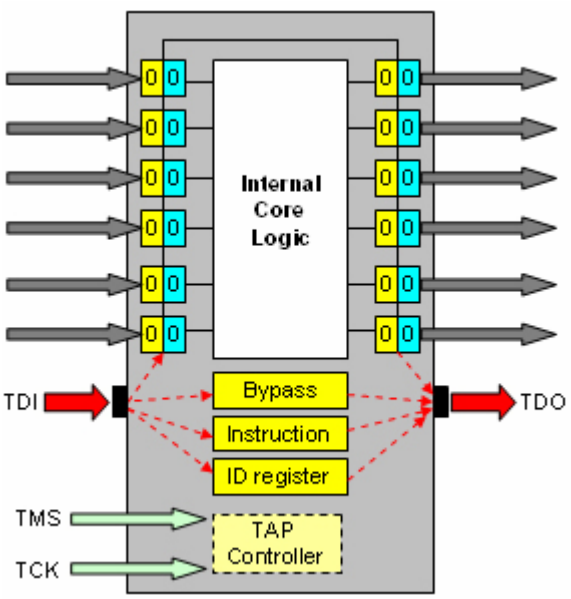
The importance and widespread usage of the Boundary Scan standard by the microelectronics industry at different design, debug, production, and maintenance phases, makes it a necessary technique to be studied by future engineers.

Boundary Scan Overview

The *Boundary Scan standard* (IEEE Std 1149.1 “Test Access Port and Boundary-Scan Architecture”) [1] also known as JTAG was officially approved in 1990. It was being developed by Joint Test Action Group (JTAG) since mid 1980s. Right from the beginning, the adoption of Boundary Scan standard by the industry was not very fast. However, it has much accelerated over time and facilitated appearance of other branches and derivatives of 1149 family.

Device Architecture

The Boundary Scan architecture implies the introduction of scan chains in such a way, that each pin of each chip receives an internal control point. Therefore, we do need to include extra circuits on an ASIC in order to test it. This is an example of increasing the cost and complexity (as well as potentially reducing the performance) of an ASIC to reduce the following testing costs.



The general boundary scan architecture is shown in figure on the right.

This configuration requires that the board and each IC that is part of the boundary scan include the following principal hardware components:

- A test access port (TAP) with at least four pins;
- A group of registers: a one instruction register (IR) and number of data registers (DRs);
- A TAP controller: a 16-state finite state machine.

The boundary scan architecture allows configuring the cells for at least following testing modes:

- External testing: interconnects between the chips on a board;
- Internal testing: testing of the logic within the chip.

Test Access Port

TAP controller can have a various configuration if signals and different implementations. But there should be the signals applied to the four mandatory pins

and the optional TRST, which is used asynchronous test logic reset. The four mandatory pins include two data pins: TDI and TDO, and two control pins: TMS and TCK.

Here is more detailed description of TAP controller signals functions:

- TDI. This signal allows the introduction of test data. The TDI of the board is connected to the TDO of the first chip in the scan chain. This signal is shifted in the registers at the positive edge of the TCK and, when not in use, is kept high.
- TDO. Test data output allows scanning out of the test data. The TDO of the board is connected to the TDO of the last chip in the scan chain. Data are shifted out at the negative edge of TCK.
- TCK. The test clock operates the testing function synchronously and independent of the system clock. It controls the shifting data and instruction's codes among the TAP registers.
- TMS. The input stream of this pin is interpreted by the TAP controller as a control signal and used to manage the various test operations.
- TRST. This signal is used to reset all testing logic asynchronously and independent of TCK. Implementation of TRST is recommended by the standard, but it's an optional one.

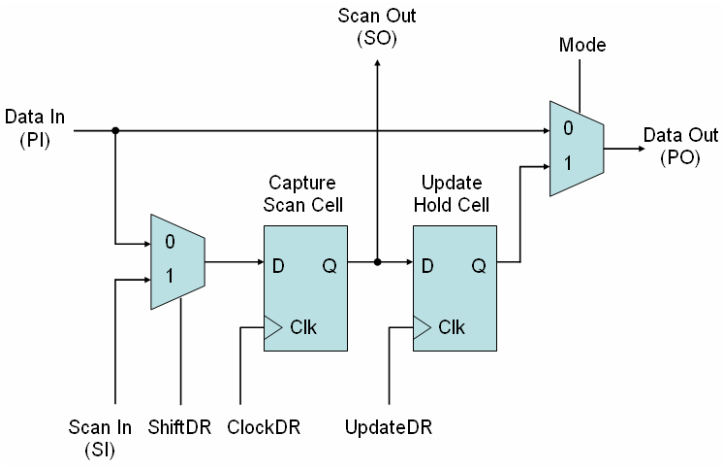
Registers

There are three mandatory registers defined by the standard: the instruction register, the bypass register, and the boundary scan register. The last one is actually a sequence of the boundary scan cells (Section 0). A brief description of these registers is given in further sections.

Boundary scan cell

Common boundary scan cell may be used inside input or output pins. During normal operation mode the input signal is applied to the data-in pin and passes to the internal logic through the right multiplexer. Thus the value of Test/Normal should be 0, while the Shift-Load mode may be either 0 or 1.

When the same cell is used as an output pin, the data coming from the internal logic of the chip pass, through the right multiplexer, to the output of the chip. An example of a boundary scan cell implementation is shown in the figure below.



For the test mode, the data are coming through TDI, thus the Shift/Load mode should be 1. The data are latched for internal testing or to be shifted to the next BSC when the clock is activated.

Bypass Register

The bypass register is set to logic 0 at the rising edge of TCK when the TAP controller is in the Capture-DR state. Use of the bypass register allows the signal at the TDI to pass directly to the TDO of the chip, thus bypassing all the other BSCs of the chip.

Boundary scan register

The boundary scan register consists of all the BSC cells on the periphery of the chip. Boundary scan register is part of the testing of the interconnects and of any logic between the boundary scan ICs on the board.

Instruction register

The instruction register is a serial-in, parallel-out register. In this register the appropriate instructions are shifted in serially and the individual instructions are captured in parallel. Due to this, an instruction register has a shift section that can be connected to TDI and TDO, and a hold section, holding the current instruction value. IR must be at least two-bits long (to allow coding of the three mandatory instructions - Bypass, Sample/Preload and Extest), but the maximum length of the IR is not defined. Standard defines that in capture mode, the two least significant bits must capture a 01 pattern. The values captured into higher-order bits are not defined. One possible use of these higher order bits is to capture an informal identification code if the 32-bit ID register is not implemented.

Device identification register

The device identification register is optional, but if included on the IC, it should comply with the standard. It must be a 32-bit-long parallel-in and serial-out. It is intended to contain the manufacturer's number and the version number. This information facilitates verifying that the correct IC is mounted in a correct position and it is the correct version of the chip. Unlike the other registers, the information is not passed to an input latch.

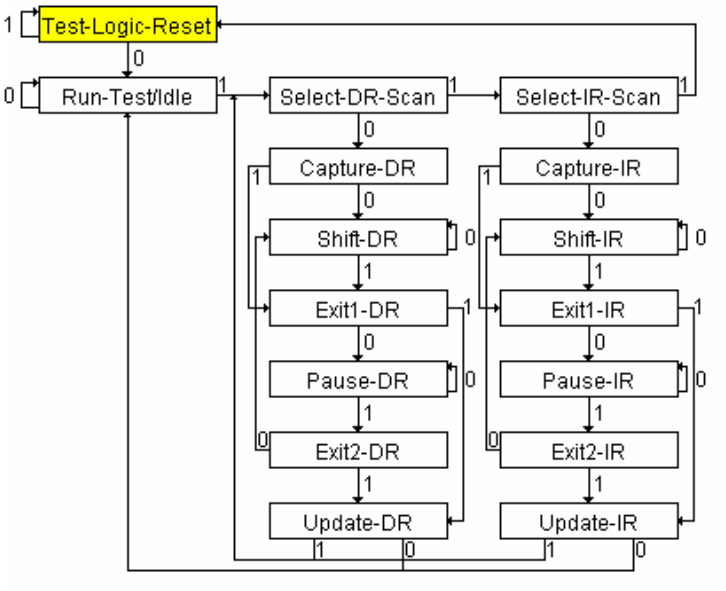
TAP Controller

The TAP controller has three main functions:

- Loading the instructions in the IR;
- Providing control signal to load and shift the test data into TDI and out of TDO;
- Performing test actions, such as capture, shift, and update test data.

The state diagram of the TAP controller is shown in **Error! Reference source not found.**. Some of the states correspond to actual operations on the data (DR) or the instructions (IR), while others allow some flexibility in the flow of operations.

TAP States



At power on, the controller is in the Test-Logic-Reset state. It remains as long as TMS is high and the circuit is in normal operation mode. As soon as TMS changes to logic 0, the controller is in the Run-Test-Idle state.

To start testing, the instruction needs to be loaded into IR. For this, TMS is held high and the TCK is clocked twice for the controller to reach the Select-IR-Scan state. Now TDI and TDO are connected to IR, and all IR registers on the board are serially connected. Next, the controller passes to the Capture-IR state with TMS = 0. Once the instruction is loaded in the IRs, then, with TMS still low, the controller stays in the Shift-IR state for as many clock cycles as needed by the test mode. In this state, the previously captured data are shifted via TDI and via TDO. If shifting is not needed, TMS = 1 and the controller bypasses Shift-IR and enters Exit1-IR state. The latter state as well as any of the other exit states is temporary. At the next positive edge of the clock, there is transition to another state. If TMS = 0, the next state is Pause-IR, and the control remains in this state until TMS = 1. The Pause-IR state is needed when the shift is done in a chain of different lengths. From this state, the control goes to Exit2-IR, then to Shift-IR if TMS = 0, or to Update-IR, if TMS = 1. The controller enters this state once the shifting process has been completed.

The new data are latched into their parallel outputs of the selected data registers at the falling edge of the TCK. Depending on the value of TMS, the next state is either Run-Test-Idle or Select-DR-Scan. When the controller is in the DR branch of the state diagram, it performs on the IR operations similar to those described above.

Instructions

There are ten instruction defined by the standard. Three of these are mandatory: BYPASS, EXTEST, and SAMPLE/PRELOAD. Six are optional: INTEST, IDCODE, USERCODE, RUNBIST, CLAMP, HIGHZ. Below is descriptions of three mandatory instructions as well as three commonly used optional instructions.

BYPASS

This is a mandatory instruction, which is used to allow quick passage through this device to another device connected in the scan chain. The bypass register is active during BYPASS instruction is loaded into IR.

SAMPLE/PRELOAD

This instruction is a mandatory one and used to preload known values in the boundary scan cells (boundary scan register is selected). Device is in functional mode, not test mode, so it is possible to scan BSR without interrupting the normal operation of the internal logic.

IDCODE

Although this optional instruction does not involve testing of the board, it helps identifying misplaced ICs. Often, it is difficult to distinguish between similar devices, and discovering the reason for malfunction of the board may take unnecessarily a long time. Identification register is selected in Test-Logic-Reset state, if available, else Bypass register selected.

INTEST

Boundary scan register selected during INTEST is active. This optional instruction is used to apply patterns to the device itself. Boundary scan cells have permission to write to their outputs (device in test mode) The test data are applied one at a time at the rate of TCK.

RUNBIST

Because of the growing importance of internal self-test structures, the behaviour of RUNBIST is defined in the standard. The self-test routine must be self-initializing (i.e., no external seed values are allowed), and the execution of RUNBIST essentially targets a self-test result register between TDI and TDO. At the end of the self-test cycle, the targeted data register holds the Pass/Fail result.

CLAMP

This optional instruction is used to control the output signal of a component to a constant level by means of a BSC. This is useful to hold values on some pins of the circuit, which are not involved in the test. These required signals are then loaded with other test patterns every time they are needed. This instruction, although useful, increases test application time.

HIGHZ

The HIGHZ instruction is optional and forces all outputs of a component to a high-impedance (High-Z) state. High-Z drives these values to the three-state controls causing them to go to their high-Z drive state but leaves bypass register as the selected register.

Exercise I

TAP controller study

1. Run the boundary scan applet. After that, it automatically enters the TAP mode and the default board will be loaded too.

2. Make transitions through the state diagram of the TAP controller. The state diagram is illustrated in the right upper corner of the applet window. As you see, the starting point is the Test-Logic-Reset state. On the diagram find following states: Pause-DR and Exit2-IR, and then using TMS and TCK buttons try to go there. Continue practicing with other chosen target states.

3. Be sure to find the answer to the following question: what is the maximum number of clock cycles needed in order to return to the Test-Logic-Reset state from a random unknown state if you keep TMS signal constantly 1?

Exercise II

The Instruction Register (IR) study

1. Find the IR branch on the TAP controller state diagram. Then make some necessary transitions through the state diagram in order to reach the Shift-IR state. Now, being in this state you have to load values that are defined in the step 2.2 to the IR of each chip that is currently illustrated on PCB (Printed Circuit Board) Panel. As you can see, the IR is a rectangle situated on each chip (yellow color of a rectangle shows currently loaded data, red colored rectangle means updated data).

2. If you have reached the Shift-IR state, load all 0's combination to the IR of the first chip using TMS and TCK signals (simply push TMS(0) and then click TCK; the number of clicks corresponds to the number of 0's you want to load to the IR). Then insert all 1's combination to the IR of the second chip (push TDI(1) and then again click TCK as much number of times as the number of 1's that you wish to be loaded to the IR) and so on.

3. In the similar way try to fill the IRs of all chips. Note that before filling you must define sizes of all IRs. For this, just count the number of bits of all IRs. Before the last bit insertion, push TMS(1) and then using TCK make transition to the Update-IR state of the state diagram. After the transition, you can see that every IR has the same data as you have loaded into them. Thus, you have updated the contests of all IRs. Finally, you have to have all 0's combination in the IR of the first chip, all 1's combination in the IR of the second chip and so on

Exercise III

The Data Register (DR) study

1. Choose Command mode under Mode menu. Afterwards, for the chip that has ID code register choose IDCODE instruction with an input 1, for other chips choose BYPASS and select any input combination you want.

2. When all selections are done, push the button Scan IR and then Scan DR. After that diagnostic information should appear. Now you can see the data that you have loaded to the DR and the data that has been read on its output. As you have already noticed, the ID code represents itself a number in a hexadecimal system (for example, 0000B143h). Now you must compare this ID code with CHIP.IDCODE output value, which you must convert from a binary format to a hexadecimal system.

3. Answer to the following question: what does it mean if the converted value is the same as the chip's ID code? And what if it differs?

Exercise IV

Boundary Scan instructions study

1. Try to understand difference between EXTEST, INTEST, BYPASS, SAMPLE, CLAMP and HIGHZ instructions. For this, review the theory and then on practice try to supplement your knowledge.

2. Choose EXTEST and INTEST commands for all chips. Select the test vectors. Note that you must take into account the control signals of all chips. Simulate chosen vectors: push Scan IR and then Scan DR. What information is observed on the outputs of all chips? Is there any difference between EXTEST and INTEST instructions? If yes, name it? Which of these instructions is optional, and which is mandatory? (not answered in an Example)

3. Perform similar tests with other instructions. Try to analyze the diagnostic information after their simulation. Find out what purposes these instructions are used for. How do you think why some of them are mandatory and some are optional instructions?

Exercise V

The interconnect diagnosis

1. Choose Command mode under Mode menu. Then, choose Mode → Random fault under Diagnostics menu. By this, you have inserted a random fault on one of the wires.

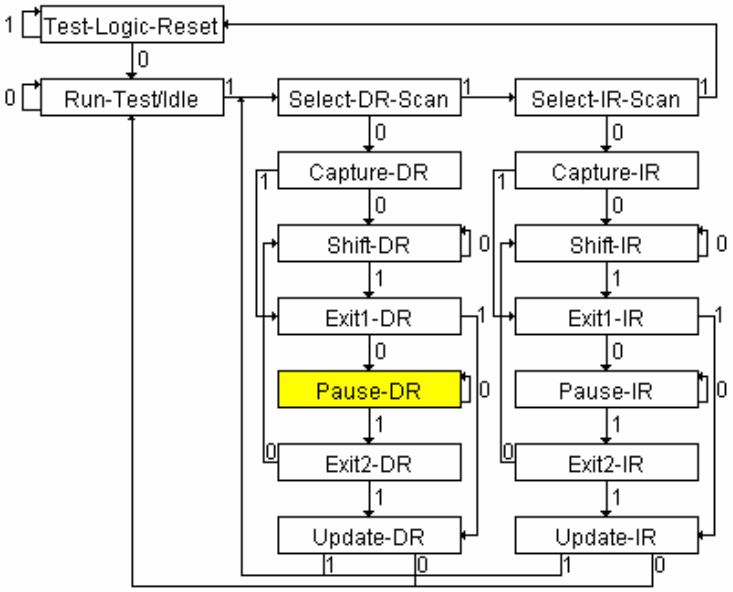
2. In order to perform the interconnect diagnosis we must set up all the chips in EXTEST mode. Now we need to select the proper test vectors taking into account the control signals of all chips. Selection of the right values of the control signals guarantees us that all outputs of the chips will not change to a high-impedance state.

3. Let's select the inputs for all chips. Let the first input pattern for all chips consist of all 1's. After the selection, push Scan IR and then Scan DR. Now the simulation process has started and after a few moments the diagnostic information should appear. But that is not all. Now select the second pattern for all chips. Let it consist of all 0's. After you have selected it, push Scan IR and then Scan DR like it was done in case of the first simulation. After the second simulation, repeat the second step with all 0's patterns again. In the end, we should get the proper diagnostic information for the 3 simulation processes.

4. Analyze the diagnostic information for the last 2 simulations. Try to answer to the following question: why the results of only the last 2 simulations should be analyzed? Find the random error you have inserted and then click on the proper connection wire (you should see the name of this wire). Afterwards, choose Give the answer under Diagnostics menu, and select the name of the wire with the same name. The applet tells you if your answer is correct or not.

Explanations and Examples

Here *TAP controller study*



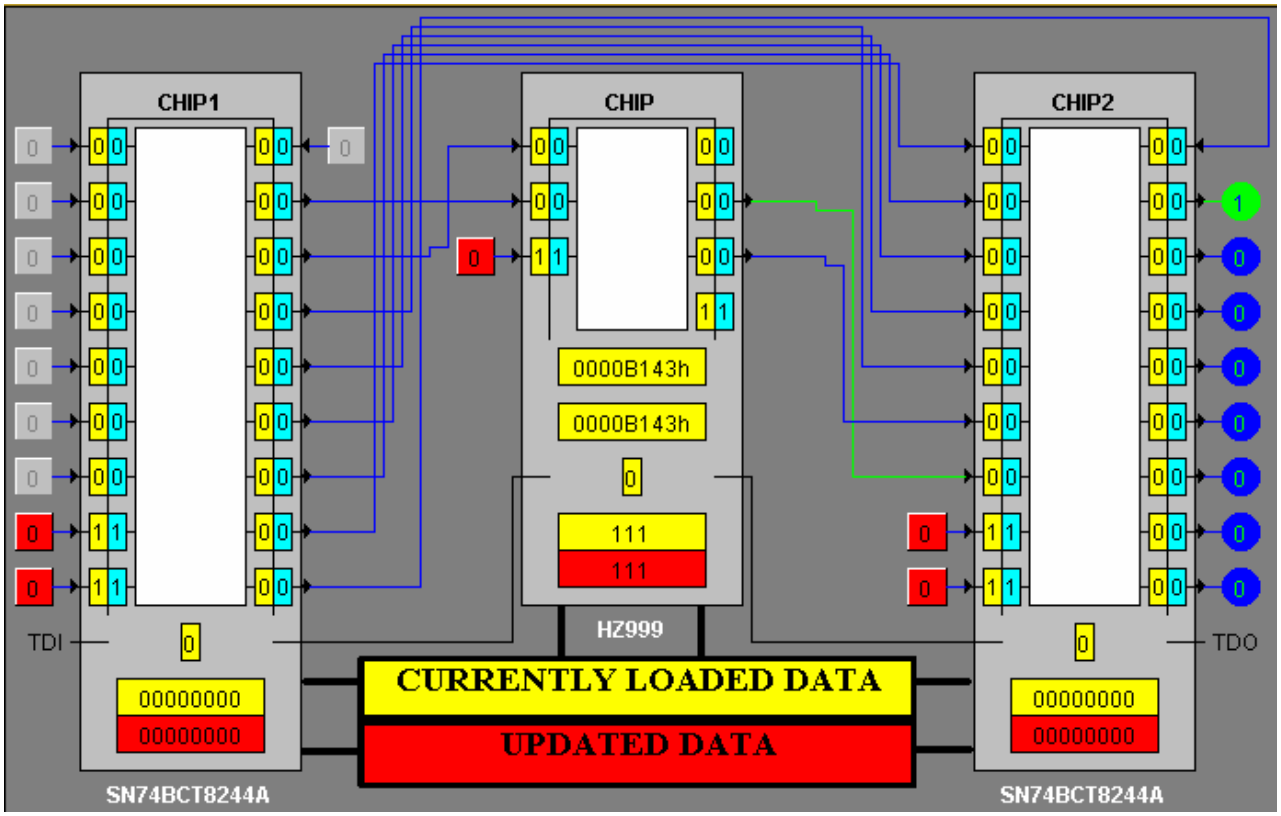
Accordingly to the required task, it is necessary to go to the following states on the diagram: Pause-DR and Exit2-IR. For this, we use TMS and TCK buttons. Suppose, we have chosen Pause-DR for our target state. At first, we must find the states on the diagram through which we could reach this state. Thus, these states are: Run-Test/Idle, Select-DR-Scan, Capture-DR, Shift-DR, and Exit1-DR. In order to be in Run-Test/Idle state we need TMS(0) to be chosen and then we push TCK button. After the transition the next state to reach is Select-DR-Scan. As you see on the diagram, we need to choose TMS(1). After that, we push TCK button in order to make a transition. Now we are in Select-DR-Scan state. Further try to make the rest of necessary transitions to reach Pause-DR state and then return back to Test-Logic-Reset state. Use the logic that is described above. See the figure below, which illustrates our target state's Pause-DR location on the diagram.

The Instruction Register (IR) study

Suppose, we are in Shift-IR state on the diagram. Now we need to load some values that are defined in the step 2.2 to the IR of the each chip on the board. Thus, we set to the next step.

So, let us define the number of bits for all IRs: for CHIP1 and CHIP2 the sizes are equal to 8 bits and CHIP has a 3-bit IR. Thus, we need to load to the registers of CHIP1 and CHIP2 eight 0's, and to CHIP register the test pattern of three 1's. Moreover, we have to remember that the direction of data flow through the chips is

from left to right. It means that the first IR to fill is a CHIP2 register, then CHIP IR, and the last is CHIP1 register. Suppose, we have entered the test patterns for all IRs. If you have done all the actions (the rules for a vector insertion are mentioned in a task step) correctly, you must get the same figure as you can see below.



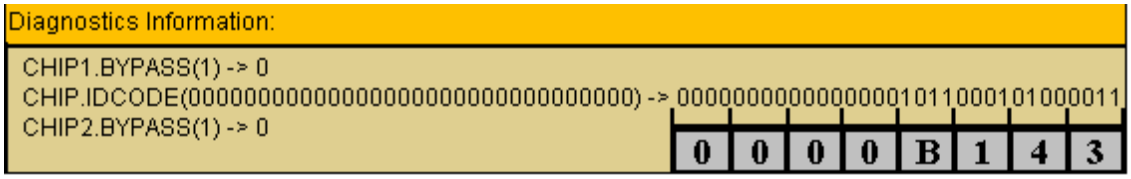
The Data Register (DR) study

As the default board has been loaded, we see that the ID code is defined only for CHIP (the ID code is illustrated in the center of it).

Now we need to select the inputs for all chips. For this, see the figure below.

Chip	Command	Input
CHIP1	BYPASS	1
CHIP	IDCODE	0000000000000000
CHIP2	BYPASS	1

Afterwards, we push the button Scan IR and then Scan DR. After the simulation, we have got the diagnostic information (see the figure below together with analyze information). After its analysis, we have convinced that the converted value of CHIP output is the same as the chip's ID code (0000B143h).



Boundary Scan instructions study

Suppose, we have reviewed the theory about following instructions: EXTEST, INTEST, BYPASS, SAMPLE, CLAMP and HIGHZ.

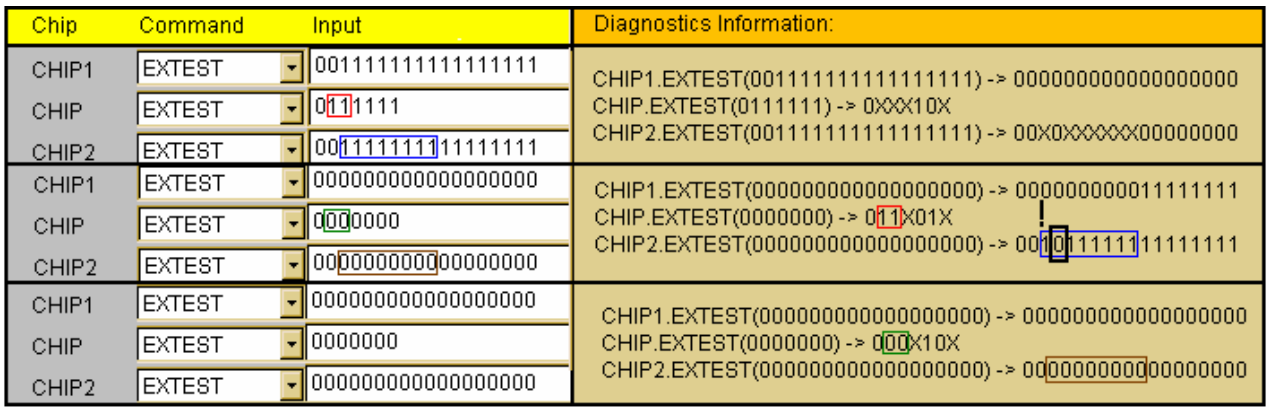
Let's choose HIGHZ and BYPASS instructions and study them. The theory says that HIGHZ instruction is used, for instance, when an in-circuit test is required for testing a non-BS compliant component; BYPASS instruction is also very useful, because it permits bypassing of the current circuit and places the one-bit bypass register between TDI and TDO of the chip when another circuit is being tested. Now, let's practice with these instructions. Suppose, we have done some necessary selections that are illustrated in the figure below. On PCB Panel (the second figure below), we can see that all CHIP wires are in a high-impedance state indeed (these wires are white colored). Thus, we have studied HIGHZ and BYPASS instructions on practice.

The interconnect diagnosis

Let us insert a random fault into the circuit.

Next we set up all the chips in EXTEST mode and select the proper test vectors. The control signals of all chips must be 0 valued.

As it is written in the task, we should get the proper diagnostic information for the 3 simulation processes. Suppose, we have done this and got some results. For this, see the figure below, which illustrates chosen inputs and the corresponding diagnostic information.



Let's analyze the diagnostic information for the last 2 simulations. We must compare the diagnostics information (namely the chips output values) of the second simulation with the input values of the previous step. The same manipulation is also applied for the third received diagnostics information, which we should compare with the input values of the second simulation. Okey, we start with CHIP.EXTEST output (2 bits that are framed by red rectangle). As you see, there is a 0 valued bit before these 2 bits. It is a control bit, so we do not pay our attention on it. Now take a look at the input pattern of the first simulation. It is clear that 2 bits in an input pattern of the first simulation are the same that were read on CHIP.EXTEST output. It means that there is no error occurred. Now we set to CHIP2.EXTEST output (8 bits that are framed by blue rectangle). Again, as it was done earlier, we do not pay attention on first 2 control bits. Instead, we look at the corresponding input pattern of the previous simulation. We have entered eight 1's, but when comparing it with CHIP2.EXTEST output, we realize that this chip's output is not the same as an input pattern. The difference is in the second bit: we have entered 1 value for this bit, but on output 0 valued bit has occurred. Thus, we have discovered the randomly inserted fault and it is stuck-at 0. Afterwards, by clicking on the corresponding wire, we find the name for it, which is S2B. The applet has confirmed our proposition: the fault has occurred on the connection wire S2B indeed. Now there is no need to analyze the rest of the diagnostics information.