iSTART

Highly Cost-effective Memory Debugging Tool: EZ-Debug

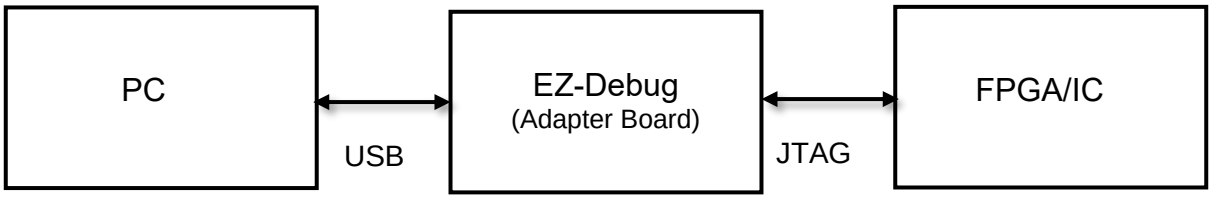
iSTART-TEK INC.

In the rapidly evolving IC industry, ensuring that ICs conform to design specifications and functionalities is indispensable and vital in the development and manufacturing processes. Thus, the IC industry extensively adopts automatic test equipment (ATE) to conduct testing tasks. However, in the past, dealing with small batches or non-mass-produced chip testing often required the use of ATE machines, which came with additional costs and lengthy testing times.

Recently, iSTART-TEK has developed a JTAG-to-USB debugging tool called EZ-Debug, based on the PC platform. This tool enables fast and efficient testing for non-mass-produced chips and chips under development. It not only reduces costs associated with ATE testing but also provides real-time insight into debugging results.

I. EZ-Debug Architecture

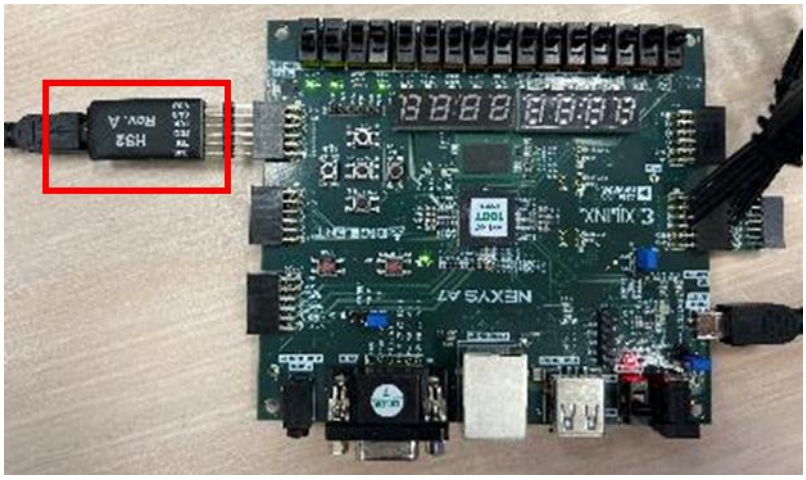
EZ-Debug enables fast and convenient debugging. The main communication between the PC and the FPGA/IC board is achieved through an adapter board. The specifications of the adapter board and the tool's overall architecture are illustrated in the diagram below. The PC is connected to the adapter board via the USB, and the adapter board, in conjunction with the tool, converts signals into the JTAG format before transmitting them to the FPGA/IC board for debugging.



Architecture Diagram

The figure below shows the actual usage. In the red box is the adapter board, and on the right side of the adapter board is the FPGA. The left side is connected to the

PC. This FPGA is utilized to simulate non-production chips and chips under development.



II. Practical Usage of EZ-Debug

After installing EZ-Debug and the driver of the adapter board, we can start using the debugging tool. EZ-Debug provides two types of testing modes: (1) auto test and (2) manual test.

```
Test mode Selection
(1)auto test
(2)manual test

Select an option:
```

1. EZ-Debug Testing Mode – Auto Test

```
Select an option: 1

set Jtag frequency(MHz): 10

set the file of test bench(.v): test.v

set the file of integ spec(.integ or .f): test.f
```

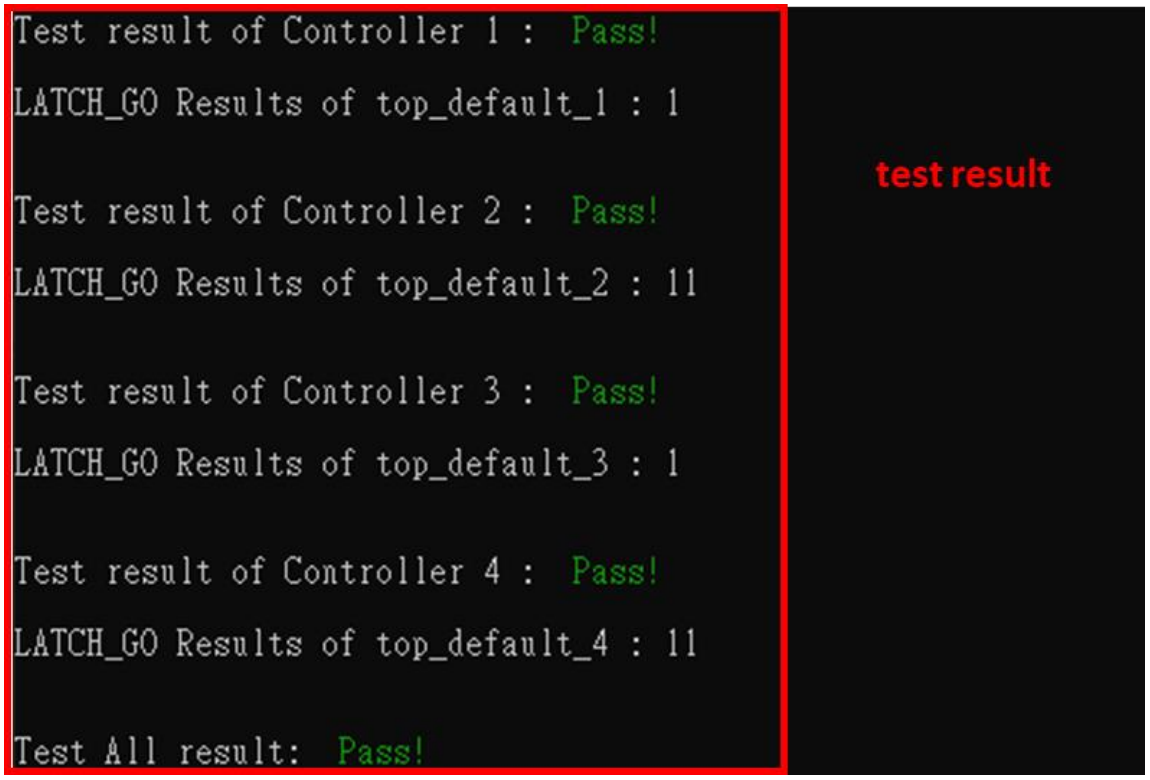
We need to set the JTAG frequency, along with the testbench file and integ spec file generated by START v3. After configuration, EZ-Debug will conduct BIST testing automatically.

- JTAG Frequency: This refers to the frequency at which JTAG operates on the FPGA/IC.
- Testbench File: This is the file generated by START™ v3 during BII for simulation (e.g., integ_tb.v).
- Integration Specification File: This is the file generated by START™ v3 during BFL for BII integration (e.g., [ctr_name]_spec.integ).

During auto test, EZ-Debug will conduct BIST testing for every controller in the design and display the testing results on the screen. If the inserted BIST circuits support the LATCH_GO function, EZ-Debug will also display the LATCH_GO results.

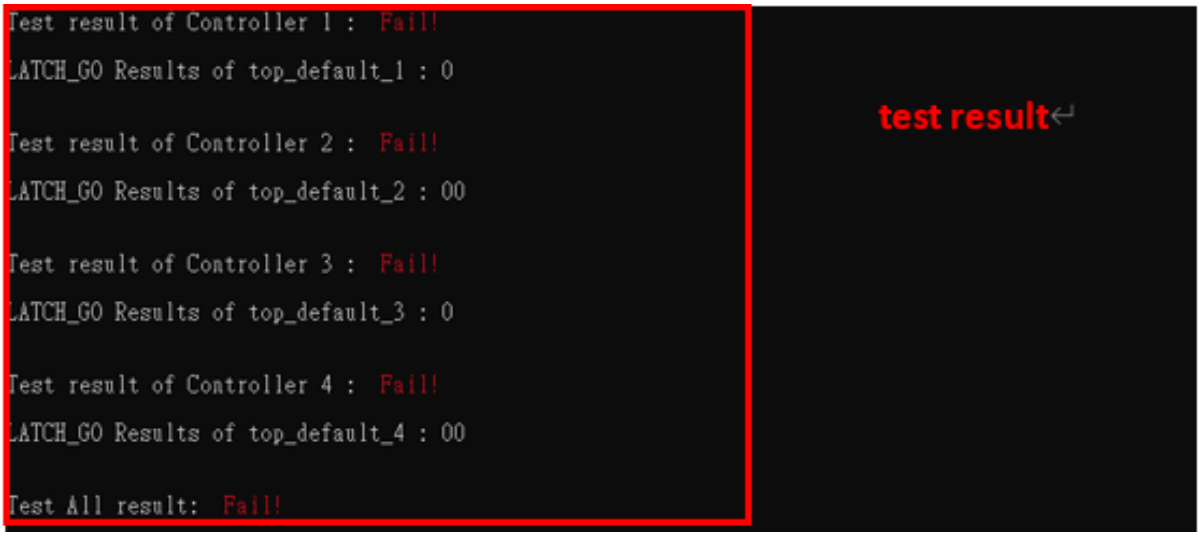
If the testing result is PASS, it will show "Test Pass!" and the LATCH_GO result is "1".

[Test Pass]



If the test result is FAIL, it will display "Test Fail!" and the LATCH_GO result will be "0". If the inserted BIST circuits support the Diagnostic function, EZ-Debug will perform diagnostic testing and show the information of completed diagnostic testing on the screen.

[Test Fail]



2. EZ-Debug Testing Mode – Manual Test

Firstly, refer to the "bist_testing" task in the INTEG testbench that has completed integration. In this task, we can find the CMD_DATA information as shown in the diagram below. Simply follow the information in the diagram to set the input binary value for sending commands using the JTAG's TDI, and then we can start testing.

```
top_default_CMD_DATA = {top_default_DIAG, top_default_ALG,
                        top_default_SEQ_ID, top_default_GRP_ID,
                        top_default_MEB_ID, top_default_MEN};
```

- [ctr_name]_DIAG: It determines whether to execute Diagnosis. Set to 1 to enable.
- [ctr_name]_ALG: When the algorithm_selection option is enabled in the BFL settings, the testbench generates the Controller_name_ALG command to control the desired testing algorithm.
- [ctr_name]_SEQ_ID, Controller_name_GRP_ID, Controller_name_MEB_ID: These are used to specify the ID of the memory being tested.
- [ctr_name]_MEN: It is the command to enable Controller BIST. Set to 1 to enable.

JTAG's TDO will generate capture_commad, which can be interpreted by referring to the test_result signal arrangement in the INTEG testbench.

```
{top_default_MGO, top_default_MRD,
top_default_SRD, top_default_LATCH_GO} = top_default_test_result;
```

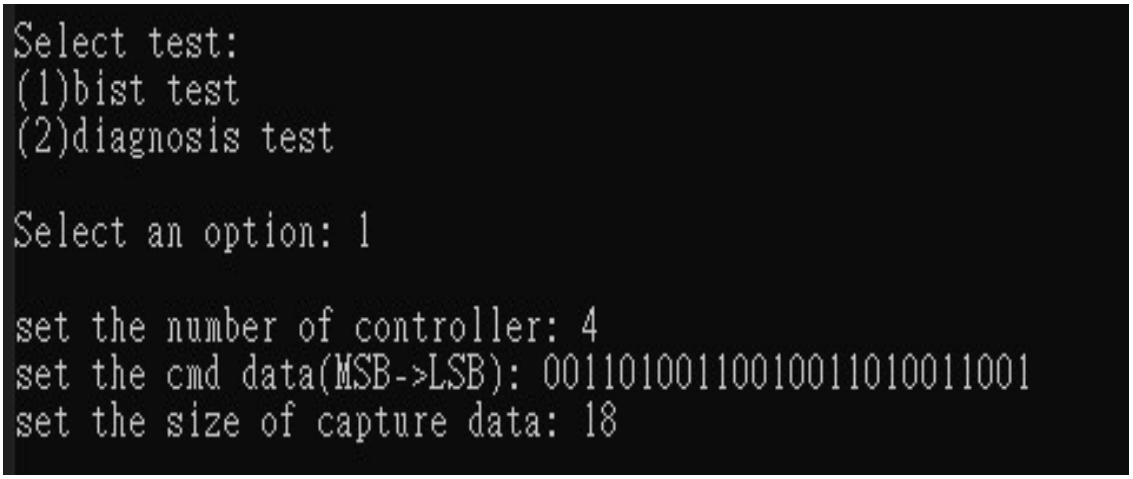
Signal Interpretation:

- [ctr_name]_MGO: It is the BIST testing result. When the testing fails, it is 0.
- [ctr_name]_MRD: It is 1 when the BIST testing is completed.

- [ctr_name]_SRD: When the Diagnosis Data is ready, it will be 1, indicating that capturing Diagnosis Data can proceed.
- [ctr_name]_LATCH_GO: The width of this signal is determined by the memory quantity in the meminfo file generated by START. When every signal of LATCH_GO turns from 1 to 0, it means that this memory testing fails.

When using the tool, input the number of controllers, the test commands, the length of capture result, and the bit number of MGO/MGD. EZ-Debug will then conduct the testing and display the row-data of the test results on the screen.

[Input the testing commands]



[Input the MGO/MRD bit number]

```
controller #1
set the bit number of MGO in test result: 17
set the bit number of MRD in test result: 16
controller #2
set the bit number of MGO in test result: 13
set the bit number of MRD in test result: 12
controller #3
set the bit number of MGO in test result: 8
set the bit number of MRD in test result: 7
controller #4
set the bit number of MGO in test result: 4
set the bit number of MRD in test result: 3
```

[Testing results]

```
Test result(MSB->LSB)
0 1 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0

Test result of Controller 1 :  Fail!
Test result of Controller 2 :  Fail!
Test result of Controller 3 :  Fail!
Test result of Controller 4 :  Fail!
```