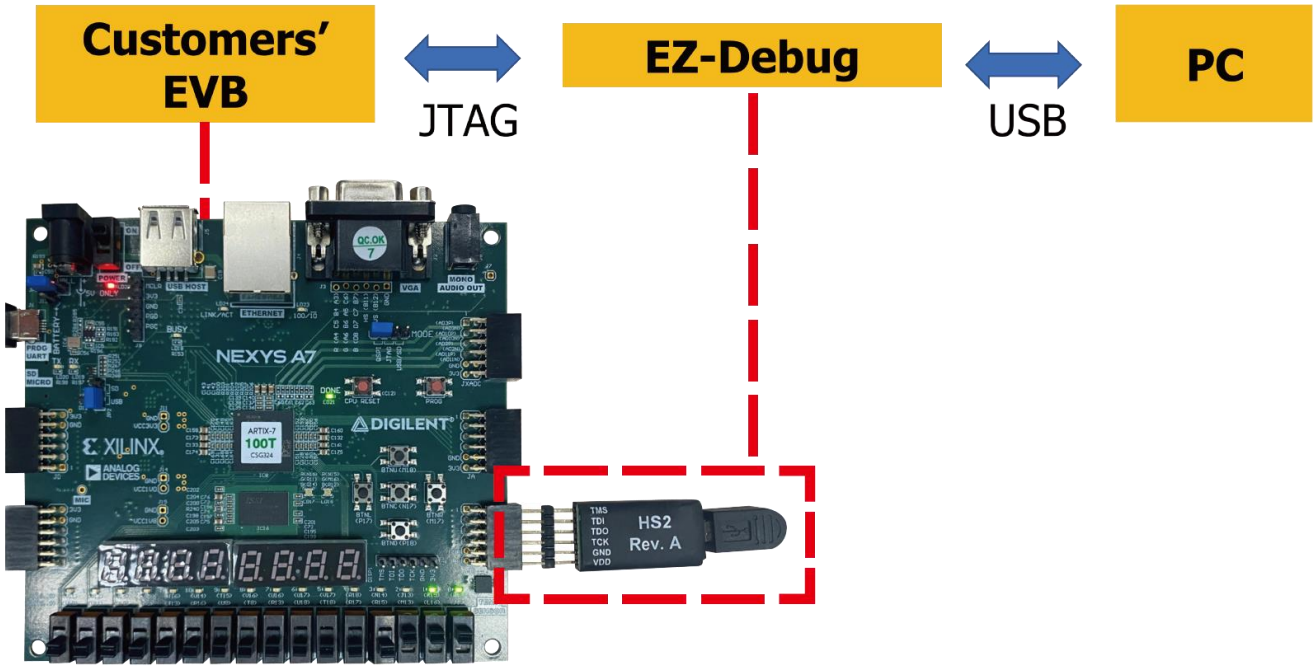
The Diagnosis Tool for Memory Testing and Development of Automotive SoCs

Automatic test equipment (ATE) is commonly used for SoC testing. The ATE then generates a log file of the testing results. When the ATE log file only consists of the signals 0 or 1, this log file is not easy to understand without using analysis tools. To simplify and expedite the analysis of ATE testing results for memories, the ATE diagnosis tool of START™ v3 is introduced. Along with the files generated by iSTART tools, it can easily analyze faulty memory information. iSTART has also developed a PC-based EZ-Debug, that can debug some chips after shuttle tape out or chips under development. This tool helps reduce ATE testing costs and obtain real-time diagnosis results.

I. EZ-Debug

1. The Block Diagram of the EZ-Debug

   The EZ-Debug debugging tool makes diagnosis convenient by allowing communication between the PC and FPGA through an adapter cable. This tool is primarily used for simulating some chips after shuttle tape out or chips under development. The adapter cable and tool specifications are shown in the figure below. The PC connects to the adapter cable via USB, and the adapter cable and tool convert the signals to JTAG (IEEE 1149.1) for FPGA/IC diagnosis.



The figure below shows actual usage. The red box indicates the adapter cable, with the FPGA on the right side and the PC on the left side.

2. Actual Applications:
   Start by referring to the "bist_testing" task in the complete INTEG testbench. In this task, we can find information about "CMD_DATA" as shown in the figure below. Then, fill in the input binary value based on the information in the figure and use TDI of JTAG (ieee1149.1) to do send_command to 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};
```

   Signal Interpretation:
   Controller_name_DIAG: It determines whether to execute the diagnosis or not. When set to 1, it will be activated.
   Controller_name_ALG: When the program_algorithm in the BFL option is enabled, the Controller_name_ALG command will be generated in the testbench to control the algorithm that is to be tested.
   Controller_name_SEQ_ID, Controller_name_GRP_ID and Controller_name_MEB_ID: These are used to specify the memory IDs that are to be tested.
   Controller_name_MEN: It is the command that enables the Controller BIST. When set to 1, it will be activated.

The TDO of JTAG will generate capture_commad. Users can interpret the content of capture_commad by referring to the “test_result” signal arrangement in the INTEG testbench shown in the figure below.

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

   Signal Interpretation:
   Controller_name_MGO: This is the BIST testing result. It will be 0 when the BIST testing fails.
   Controller_name_MRD: It will be 1 when the BIST testing is completed.
   Controller_name_SRD: It will be 1 when diagnosis data is ready and can be captured.
   Controller_name_LATCH_GO: The width of this signal depends on the memory amount in the meminfo file generated by START™. When each signal in LATCH_GO changes from 1 to 0, it means that the memory testing fails.

II. ATE Diagnosis Tool

START™ v3 (BFL & BII) Settings:

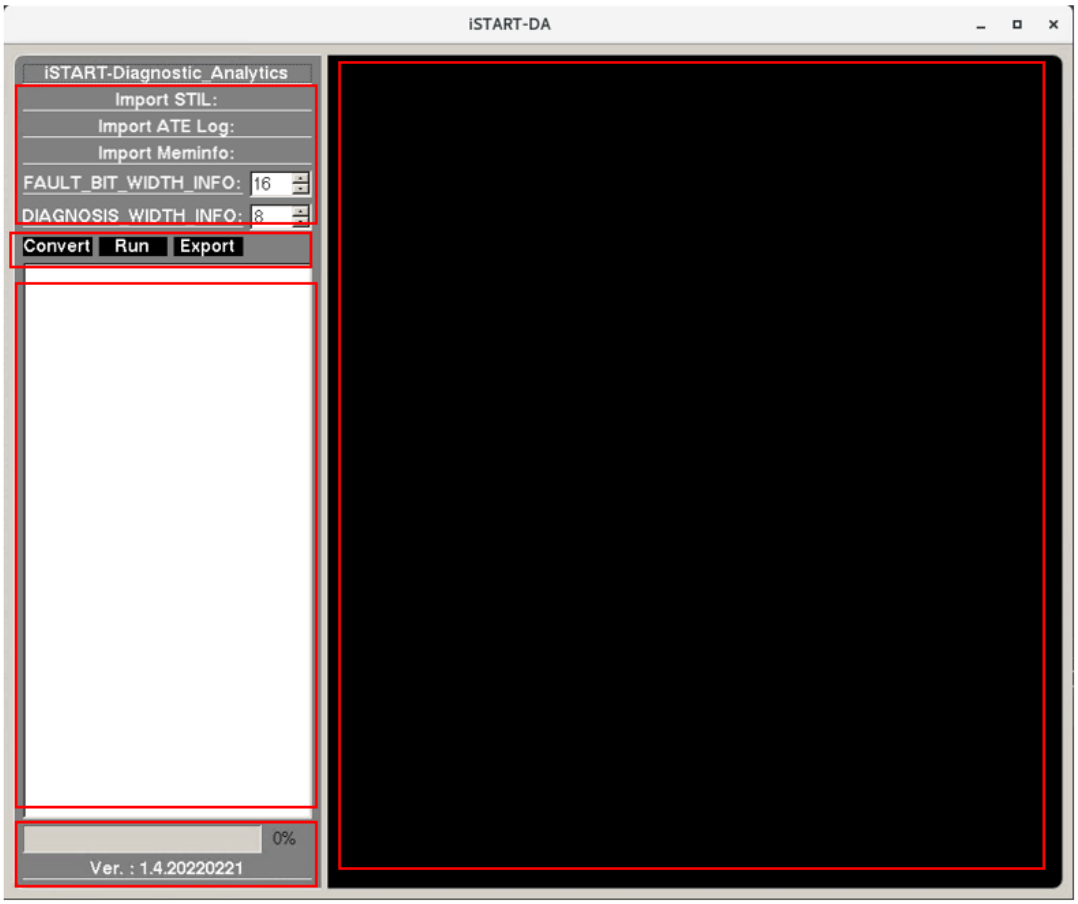
1. *.bfl Settings
   - A. diagnosis_support and diagnosis_width_info Settings
     - i. diagnosis_support: START™ v3 provides a diagnosis mode and reports diagnostic information.
     - ii. diagnosis_width_info: Align the length of test information to facilitate diagnosis and interpretation.
   - B. diagnosis_faulty_items: This provides users with the option to select diagnostic information for their own designs, including algorithms, algorithmic units, algorithmic elements, memory grouping locations, memory addresses, and memory data.

```
define{BIST}
      set diagnosis_support          = yes       # yes, no
      set diagnosis_width_info       = yes       # yes, no
      set diagnosis_faulty_items     = algorithm, operation, element, seq_id, grp_id, address, ram_data, rom_data
```

2. *.bii Settings :
   - A. Under define{Testbench}[INTEG_tb], choose the STIL as file_format to generate a memory test pattern STIL file after simulation.

```
define{Testbench}[INTEG_tb]
    set pll_wait_cycle      = 10
    set reset_cycle         = 10                # integer bigger than 0
    set file_format         = STIL             # verilog, STIL
```

Introduction of the Diagnosis Tool Interface:



1. Files and Settings:
   A. Import the STIL file for ATE: The STIL file will generate the corresponding memory test pattern based on the options set in START™ v3 (as the START™ v3 settings mentioned above).
   B. ATE log: It is the file of the ATE testing result.
   C. meminfo File: This is generated by START™ and it contains all the information of the memories.

```
# if the instance is the alais type , The ** is .
[DOMAIN=top_default, cycle=100.0ns]
     [CTR] # Hier: top
         [SEQ] # No.= 1,InstanceNo= 3,SEQ_max_addr_size= 1024,Hier: top u_t1
             [GROUP] # No.=1_1
                   [SP=1_1_1, byp=no, diag=no, q_pipe=no, repair=no] sram_sp_1024x32
                   [SP=1_1_2, byp=no, diag=no, q_pipe=no, repair=no] sram_sp_1024x32
                   [SP=1_1_3, byp=no, diag=no, q_pipe=no, repair=no] sram_sp_1024x32
         [SEQ] # No.= 2,InstanceNo= 1,SEQ_max_addr_size= 24,Hier: top u_t1
             [GROUP] # No.=2_1
                   [2P=2_1_1, byp=no, diag=no, q_pipe=no, repair=no] rf_2p_24x28
         [SEQ] # No.= 3,InstanceNo= 1,SEQ_max_addr_size= 1024,Hier: top u_t1
             [GROUP] # No.=3_1
                   [DP=3_1_1, byp=no, diag=no, q_pipe=no, repair=no] sram_dp_1024x64

###Total mbist memory instance = 5
###Total SRAM/REGFILE = 3
###Total SRAM_2P = 1
###Total SRAM_DP = 1
###Total ROM = 0
###top_default algorithm is= (March C- @2P,SOLID)(March C-,SOLID)(March C+,SOLID)(March C- @DP,SOLID)

##################Not Group Memory List#############################################
```

Hierarchy

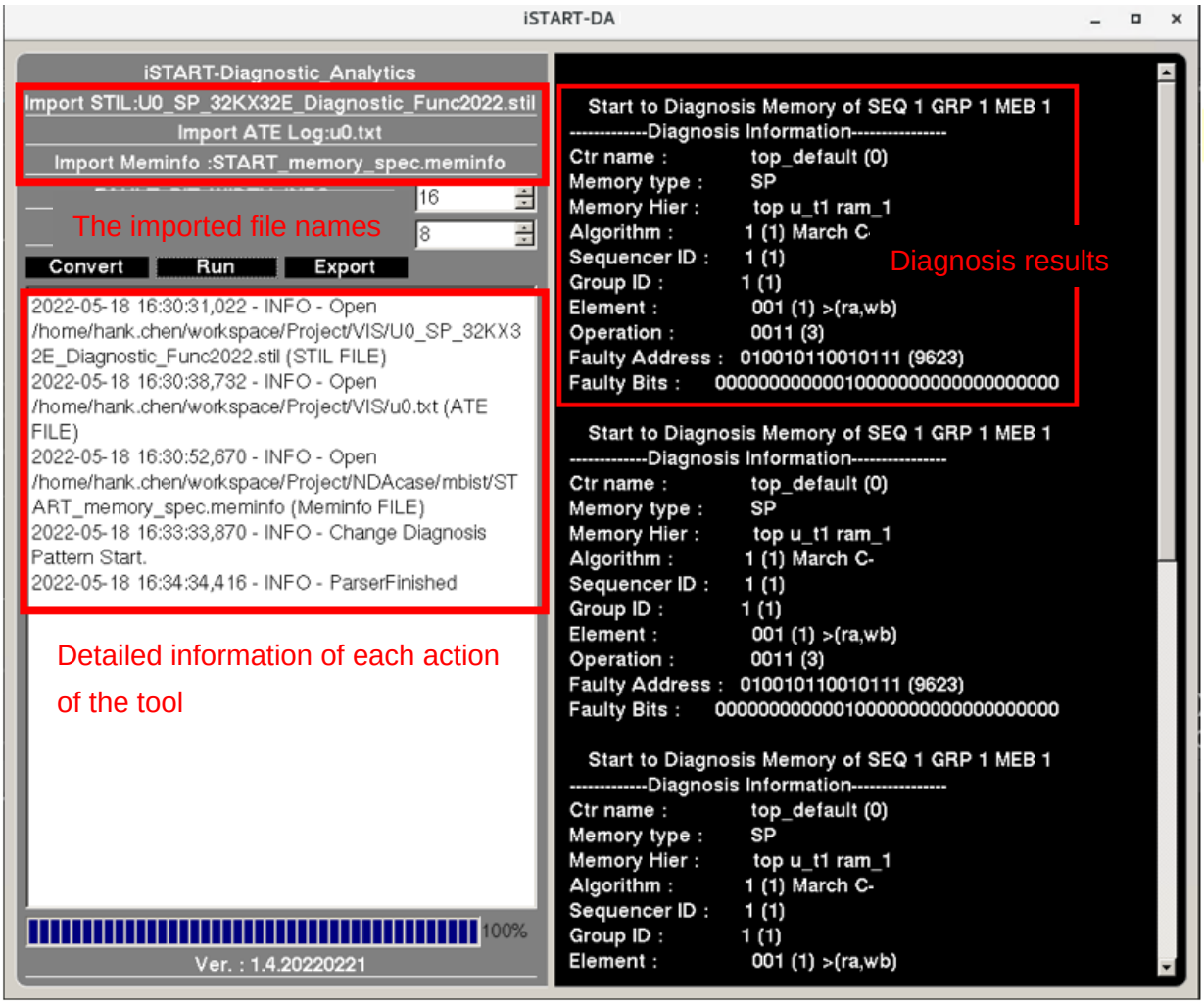
```
top u_t1 ram_1
top u_t1 ram_2
top u_t1 ram_3


top u_t1 u_2p


top u_t1 u_dp
```

Set FAULT_BIT_WIDTH and DAGNOSIS_WIDTH.

2. Convert: Perform conversion after importing all files.
   Run: Perform analysis after conversion.
   Export: Export the analysis result as a txt file.
3. Tool Execution Information: This section provides information and status updates for the tool execution
4. Progress Bar and Version Number
5. Tool Analysis Results: The tool's analysis result will be generated in this section.

Analysis Method: After importing all files and settings, the tool first analyzes the ATE result file and locate the faulty patterns. It then cross-references the STIL file to find and analyze the corresponding test commands. After analysis, it compares the memory order in the meminfo file and prints out the problematic memory information when exporting the result.

Analysis Result: The tool analysis is shown in the figure below. The right-hand block displays detailed information about the faulty memory, including the controller, memory type, memory hierarchy, and the information set in diagnosis_faulty_items of START™ v3.



Authored by Hank Chen, Senior Engineer at iSTART-TEK