Exclusively Customized Design Features for Automotive Electronics

As the global automotive industry trends towards intelligentization and electrification, the reliability test and safety features of chips for automotive electronics are becoming increasingly important to meet the AEC-Q100 specifications and enter the automotive market. iSTART-TEK's START™ v3 not only provides numerous memory testing functions and high-efficiency memory repairing solutions but also offers exclusive customized automotive electronics features, such as POT 2.0 (Power-On Test), Error-Correcting-Code (ECC) and User-Defined Algorithms (UDA). These features enable chip developers to accurately detect memory defects in automotive chips based on the applications and enhance driving safety.

I. POT 2.0 (Power_On Test)

1. Feature Introduction

   POT 2.0 is an essential function for electronic products, espacially for products that are related to automotive and safety. The feature ensures memory testing of the hardware circuits after power on process, and verifies the behavioral correctness. iSTART-TEK has developed POT 2.0 with memory testing and repairing capabilities, which have been integrated into the START^{TM} v3 tool. It allows users to incorporate memory POT functional circuits into their designs more easily, and offers the activation methods as follows.

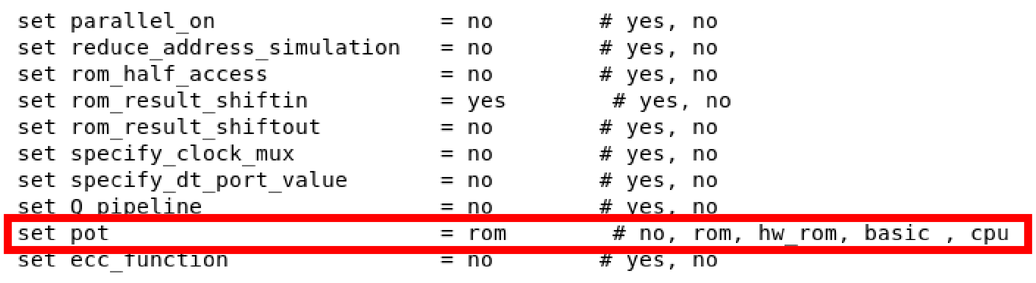
   - ROM: Storing test commands in ROM
   - RTL: Storing test commands in ROM described by RTL
   - Basic: Providing host_MEN signals for memory testing
   - CPU: Controlling the BIST circuit by giving test instructions through the CPU

   The LATCH_GO diagnostic feature can be added to the applications through marking the number of fault memory as one memory corresponding to one bit. The feature offers quick identification of memory error locations for the users. The Error Injection feature enables the insertion of error information into the Test Pattern Generator circuit to verify the correctness of the BIST circuit, significantly increasing its reliability. Additionally, when using POT 2.0, if new memory errors are detected, memory repairing can be performed.

2. Usage Methods

   START^{TM} v3 (BFL) Settings:

   Set the activation methods through the "set pot" options, as shown in Figure 1.



```
set parallel_on                 = no        # yes, no
set reduce_address_simulation   = no        # yes, no
set rom_half_access             = no        # yes, no
set rom_result_shiftin          = yes        # yes, no
set rom_result_shiftout         = no        # yes, no
set specify_clock_mux           = no        # yes, no
set specify_dt_port_value       = no        # yes, no
set Q_pipeline                  = no        # yes, no
set pot                         = rom        # no, rom, hw_rom, basic , cpu
set ecc_function                = no        # yes, no
```

Figure 1 POT Setting Options

i. set pot = rom or set pot = hw_rom

Set the pot option as "rom", and the testing commands will be stored in ROM. Set the pot option as "hw_rom", and the testing commands will be stored in the ROM described by RTL, also known as the hardwired ROM. Lastly, completing the BFL and BII workflow to generate the corresponding circuits and pins for usage.

POT consists of three main modules: ROM Memory/Hardwired ROM, ROM Controller, and MBIST/MBISR, as shown in Figure 2.

Copyright © iSTART-TEK INC.

Firstly, after receiving the commands of executing the POT function, the ROM Controller will read the testing commands stored in the ROM memory/ Hardwired ROM. Next, it will send the control signals to MBIST to start memory testing. If memory errors are detected, MBISR will automatically execute the repairing workflow.

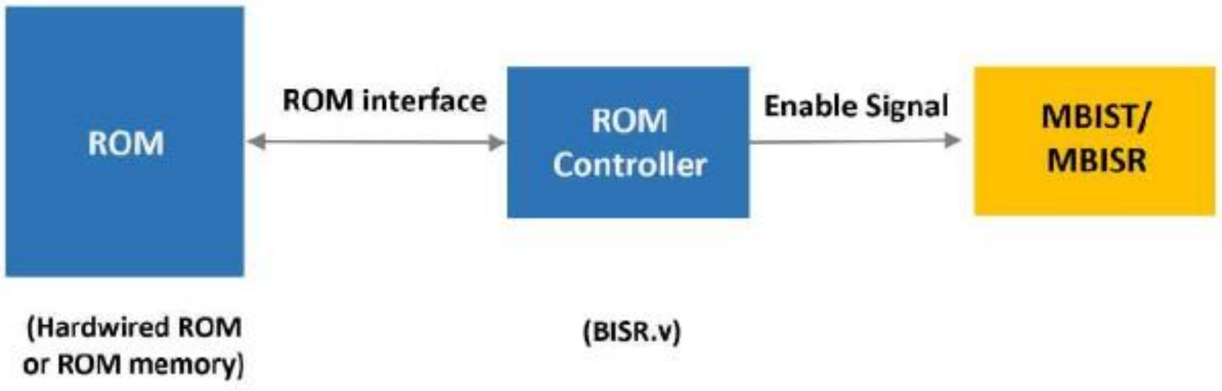


Figure 2 POT Modules

Figure 3 shows the waveforms of POT-related signals. SYS_POT is the enable signal for POT. Once the signal is activated, the ROM controller reads the test instructions from ROM memory/Hardwired ROM and initiates memory testing and repair. The test results can be obtained from the signals MGO, MRD, and RGO.

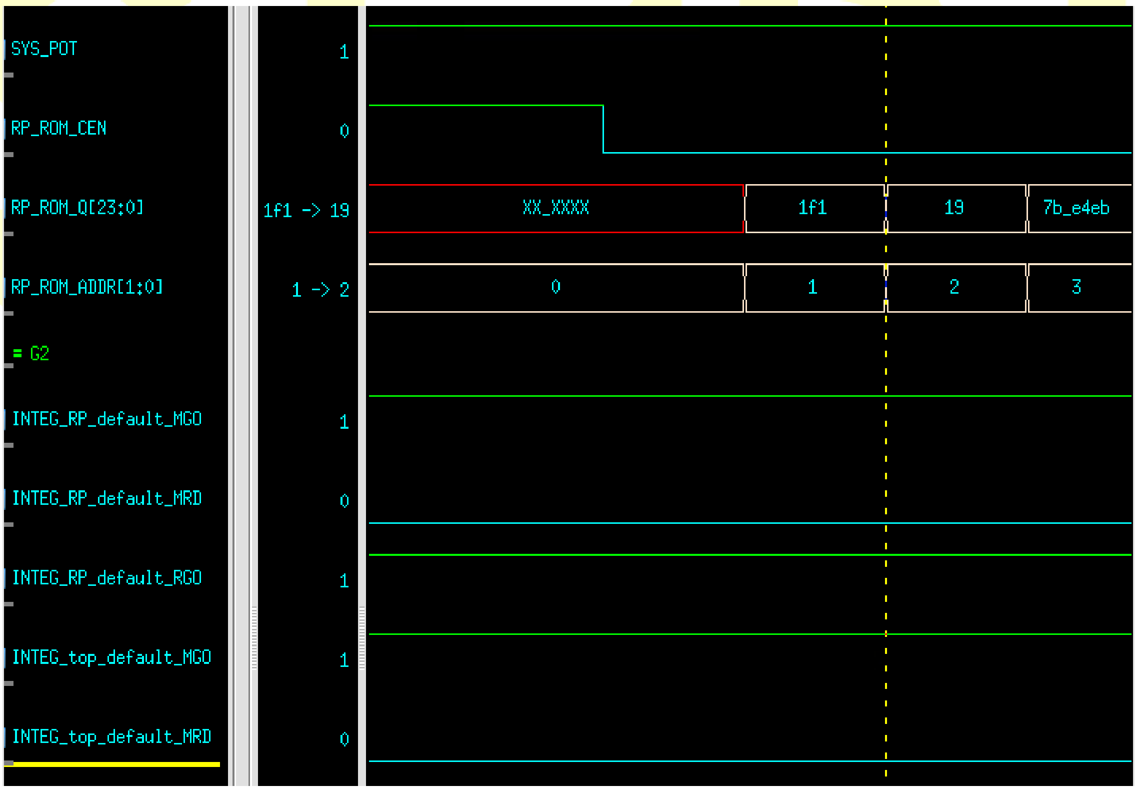


Figure 3 The Waveforms of POT-related Signals

The MBIST/MBISR signals of POT= "rom" or "hw_rom" are shown as Figure 4.

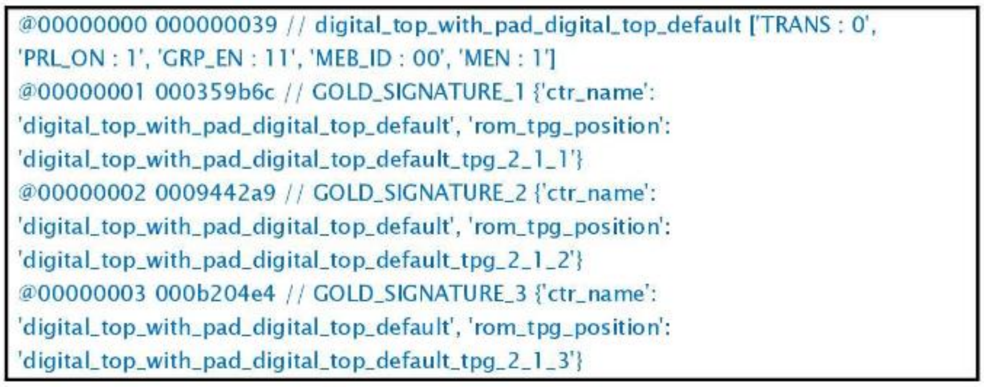
Name	Direction	Width	Description
SYS_READY	input	1	The system boot is ready to enable BISR logics (hard repair only)

Copyright © iSTART-TEK INC.

			"1'b1": Ready to load data from NVM storage)
SYS_POT	input	1	Enable Power on test (normal function test only)
BOOT_CFG_DONE	output	1	The shifting of configuration data is completed (hard repair only). "1'b1": the scan is completed "1'b0": the scan is progressing
RCK	input	1	The clock signal for storage device, BISR logics and configuration buffer
RRST	input	1	The reset signal for storage device, BISR logics and configuration buffer
MRD	output	1	Indicates if the test is ended or not. 0: The test is uncompleted. 1: The test is ended.
MGO	output	1	Indicates if the test is failed or not. 0: The test is failed. 1: The test is passed.
RGO	output	1	Indicates if the logic can be repaired or not. 0: The logic cannot be repaired. 1: The logic can be repaired. (MBISR CTR only.)

Figure 4 The MBIST/MBISR Signals of rom and hw_rom

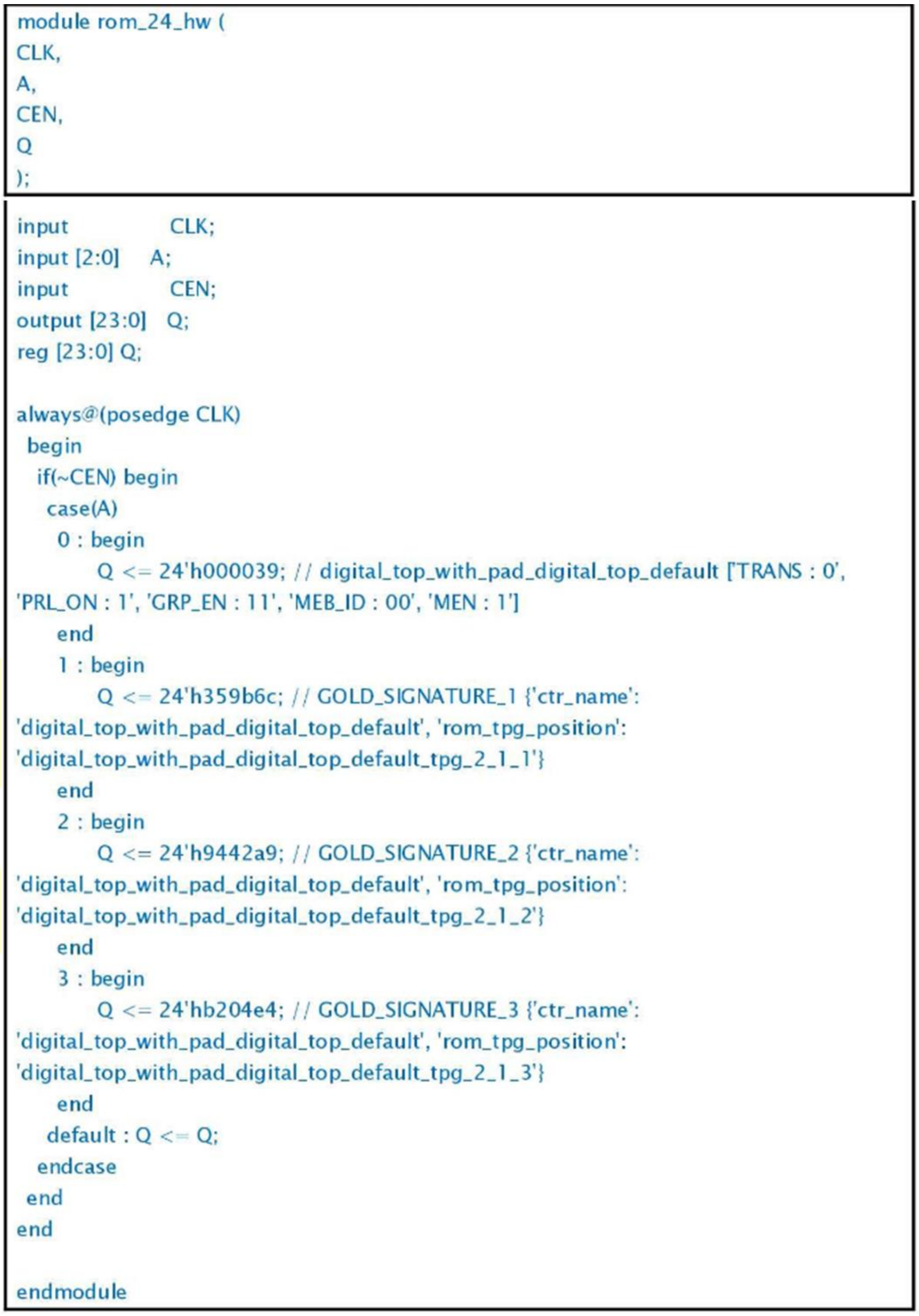
Finally, the Verilog file containing the test commands to be stored in ROM will be generated. It will generate the corresponding commands based on the BIST functions designed by users, as shown in Figure 5. The Verilog program example in Figure 6 shows the test commands stored in the ROM described by RTL.



```
@00000000 000000039 // digital_top_with_pad_digital_top_default ['TRANS : 0',
'PRL_ON : 1', 'GRP_EN : 11', 'MEB_ID : 00', 'MEN : 1']
@00000001 000359b6c // GOLD_SIGNATURE_1 {'ctr_name':
'digital_top_with_pad_digital_top_default', 'rom_tpg_position':
'digital_top_with_pad_digital_top_default_tpg_2_1_1'}
@00000002 0009442a9 // GOLD_SIGNATURE_2 {'ctr_name':
'digital_top_with_pad_digital_top_default', 'rom_tpg_position':
'digital_top_with_pad_digital_top_default_tpg_2_1_2'}
@00000003 000b204e4 // GOLD_SIGNATURE_3 {'ctr_name':
'digital_top_with_pad_digital_top_default', 'rom_tpg_position':
'digital_top_with_pad_digital_top_default_tpg_2_1_3'}
```

Figure 5 The Test Commands Stored in ROM

Copyright © iSTART-TEK INC.



```
module rom_24_hw (
CLK,
A,
CEN,
Q
);
```

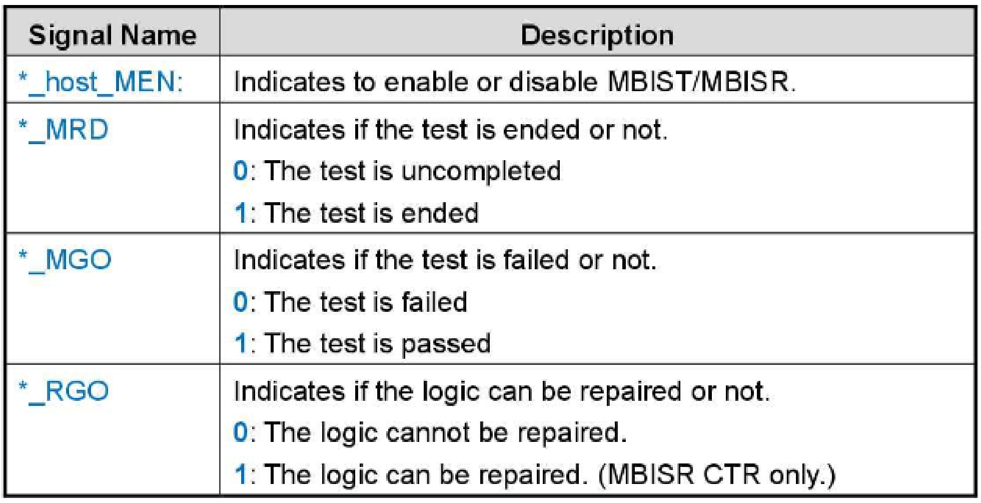
```
input           CLK;
input [2:0]     A;
input           CEN;
output [23:0]   Q;
reg [23:0] Q;

always@(posedge CLK)
 begin
  if(~CEN) begin
   case(A)
    0 : begin
        Q <= 24'h000039; // digital_top_with_pad_digital_top_default ['TRANS : 0',
'PRL_ON : 1', 'GRP_EN : 11', 'MEB_ID : 00', 'MEN : 1']
    end
    1 : begin
        Q <= 24'h359b6c; // GOLD_SIGNATURE_1 {'ctr_name':
'digital_top_with_pad_digital_top_default', 'rom_tpg_position':
'digital_top_with_pad_digital_top_default_tpg_2_1_1'}
    end
    2 : begin
        Q <= 24'h9442a9; // GOLD_SIGNATURE_2 {'ctr_name':
'digital_top_with_pad_digital_top_default', 'rom_tpg_position':
'digital_top_with_pad_digital_top_default_tpg_2_1_2'}
    end
    3 : begin
        Q <= 24'hb204e4; // GOLD_SIGNATURE_3 {'ctr_name':
'digital_top_with_pad_digital_top_default', 'rom_tpg_position':
'digital_top_with_pad_digital_top_default_tpg_2_1_3'}
    end
   default : Q <= Q;
  endcase
 end
end

endmodule
```

Figure 6 The Verilog Code of the Hardwired ROM

Copyright © iSTART-TEK INC.

ii. set pot = basic

When the POT option is “basic”, the host_MEN signal line is generated for users to activate memory testing. The results can be obtained through the MGO, MRD and RGO signals. The signal list generated by the "basic" option is shown in Figure 7.

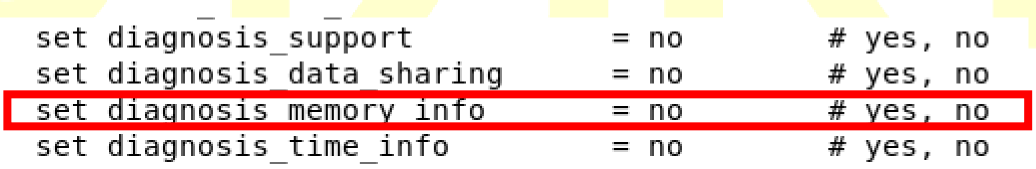


Signal Name	Description
*_host_MEN:	Indicates to enable or disable MBIST/MBISR.
*_MRD	Indicates if the test is ended or not. 0: The test is uncompleted 1: The test is ended
*_MGO	Indicates if the test is failed or not. 0: The test is failed 1: The test is passed
*_RGO	Indicates if the logic can be repaired or not. 0: The logic cannot be repaired. 1: The logic can be repaired. (MBISR CTR only.)

Figure 7 The MBIST/MBISR Signals of the “basic” Option

iii. set pot = cpu

By setting the POT option to "cpu", users can directly control the BIST circuit, and the settings can be adjusted through the .bfl file for additional features, as shown in Figure 8. For example, the LATCH_GO signals can be added to the diagnosis_memory_info function, allowing users to identify the memory number of occurring errors quickly. Figure 9 shows an example that an error occurs in memory number 6.



```
set diagnosis_support          = no       # yes, no
set diagnosis_data_sharing     = no       # yes, no
set diagnosis_memory_info      = no       # yes, no
set diagnosis_time_info        = no       # yes, no
```

Figure 8 The Latch_GO Setting on BFL

Copyright © iSTART-TEK INC.

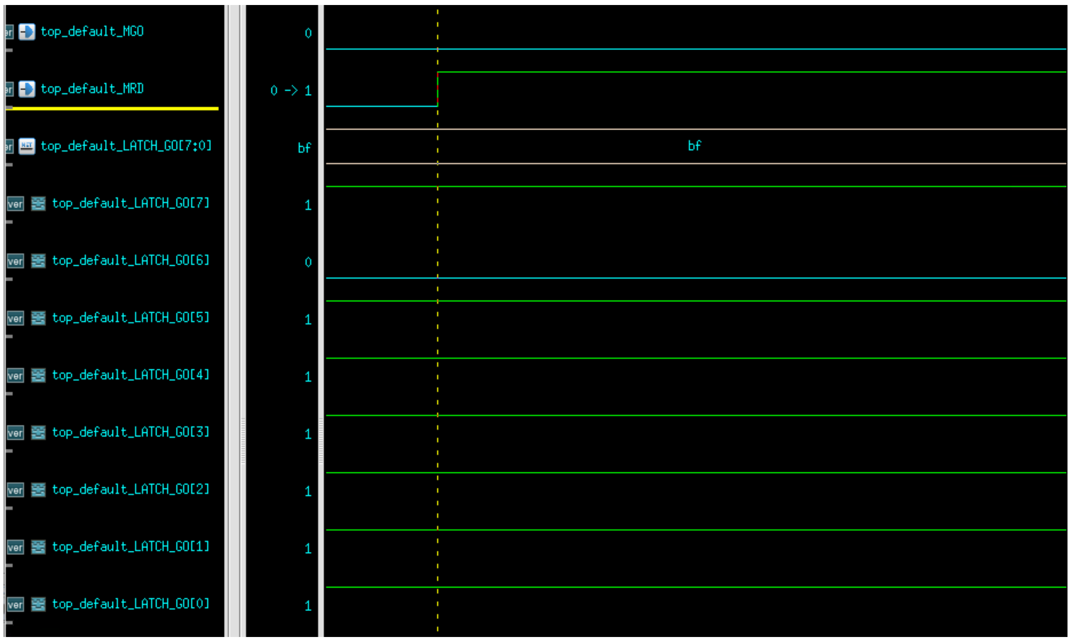


Figure 9 LATCH_GO Waveform

II. Error-Correcting-Code (ECC)

1. Feature Introduction

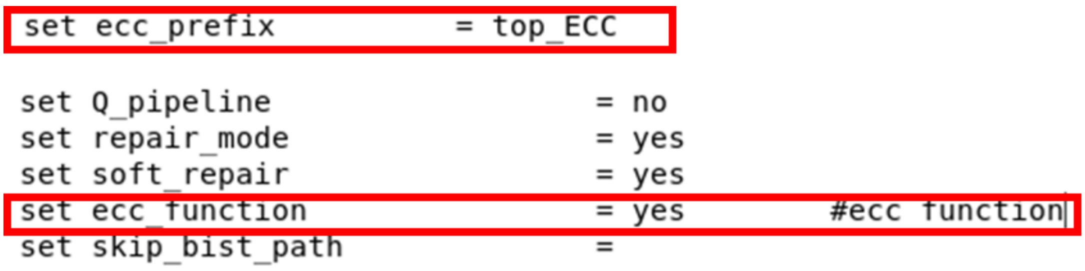
   Error-Correcting-Code (ECC) is an encoding method that allows the detection and correction of errors while transmitting. It can be applied to various data transmission and storage operations, where the receiving end uses the encoded data to detect and correct transmission errors. On applying to the memory, ECC can check the correctness of data stored in memory through its circuits.

   Memory failures cause serious consequences on various fields including automotive, industry, medical industry, and communication. The functionality of ECC improves the stability and reliability of chips during operation.

   iSTART-TEK provides ECC functionality that enables users to detect 2 bits errors and correct 1 bit error. When featuring ECC, the memory needs to allocate extra space for parity check to reconstruct the corrected data. The required space can be calculated as $2^{\text{Parity-1}}$ > Parity + Data bit. For example, if the memory data length is 22 bits, 6 bits of ECC space must be saved to check 16 bits of data.

2. Usage Methods

   START$^{\text{TM}}$ v3 (BFL) Setting:
   Adjust the options of set ecc_function to add ECC function and set the ECC name using the "set ecc_prefix" command, which are shown in Figure 10.



```
set ecc_prefix              = top_ECC

set Q_pipeline                    = no
set repair_mode                   = yes
set soft_repair                   = yes
set ecc_function                  = yes         #ecc function
set skip_bist_path                =
```

Figure 10 ECC Setting Options

After executing the BFL and BII workflow, it generates the encoder and decoder circuits of the ECC function, shown in Figure 11 & 12.

Copyright © iSTART-TEK INC.

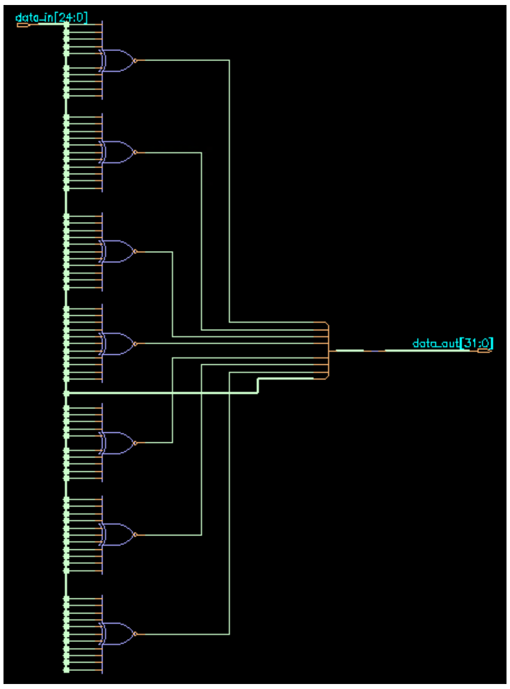


Figure 11 Encoder Circuits

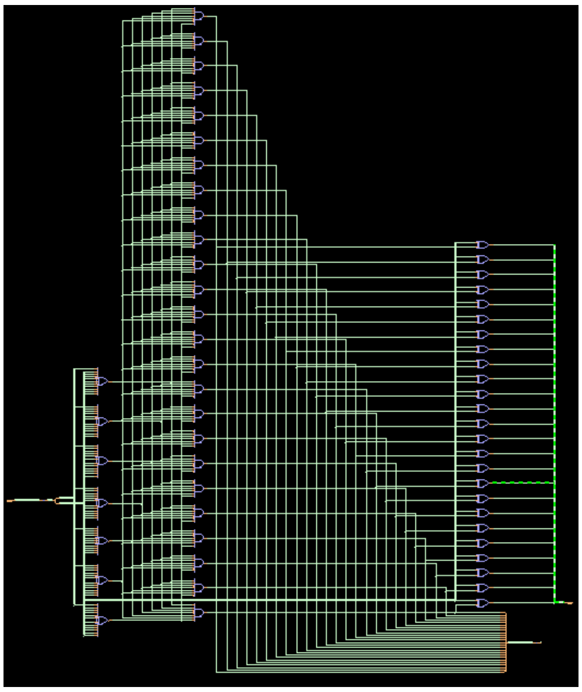


Figure 12 Decoder Circuits

Figure 13 shows the ECC waveforms, and the data_noise signal represents the data in the memory along with the encoded checking value. By decoding it, the correct data can be obtained, also allowing data correction.

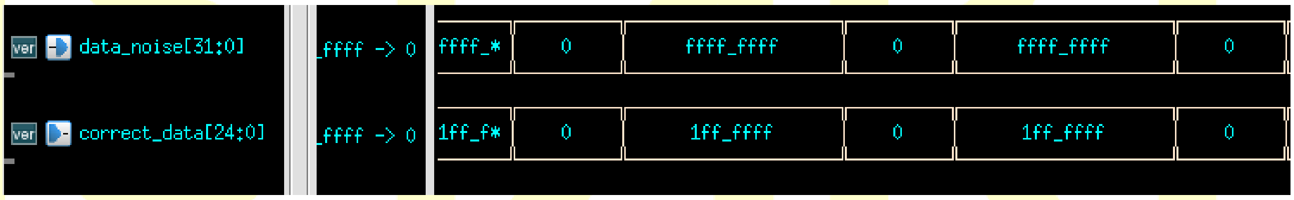


Figure 13 The ECC Waveform

II. UDA (User Defined Algorithm)

1. Feature Introduction

As technology evolves, newly developed advanced process memories combine with commonly used algorithms can result in longer testing time and behaviors of repeating testing patterns. For example, if users select both the March C^+ (14N) and March C^- (11N) algorithms, the testing time will be 25N.

March C+	>(wa) >(ra,wb,rb) >(rb,wa,ra) <(ra,wb,rb) <(rb,wa,ra) <(ra)
March C-	>(wa) >(ra,wb) >(rb,wa) >(ra) <(ra,wb) <(rb,wa) <(ra)

iSTART-TEK has developed the User Defined Algorithm (UDA) feature, which allows users to customize and edit algorithms. With this feature, repetitive elements can be removed, resulting in a shortened testing time of 23N.

```
>(wa) >(ra,wb,rb) >(rb,wa,ra) <(ra,wb,rb) <(rb,wa,ra)
         >(ra,wb) >(rb,wa) >(ra) <(ra,wb) <(rb,wa) <(ra)
```

Copyright © iSTART-TEK INC.

UDA is represented in the form of components, which can be regrouped and combined to generate new algorithms as shown in Figure 14.

Syntaxes	Functions
UP	Address counted up from 0
DN	Address counted down from the maximum value
ADD_INC	Address+1 or Address-1 decided by UP or DN
N	No read/ write operation
R(A)	A is marked as the processed pattern for the Read operation
W(A)	A is marked as the processed pattern for the Write operation
S	The memory testing has entered the sleeping state, and the sleeping time is defined by users.
,	Segment different operations
;	Complete the present elements

Figure 14 UDA is Represented in the Form of Components

UDA provides a friendly graphical user interface (GUI) that enables users to startup quickly, which is shown in Figure 15.

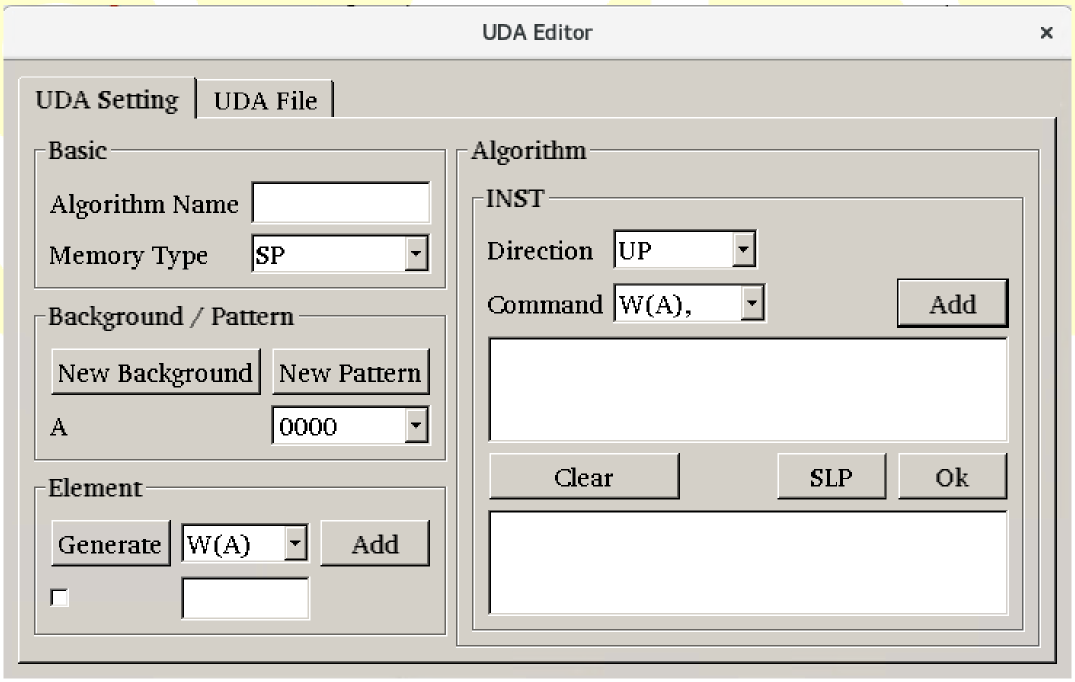


Figure 15 Graphical User Interface of UDA

2. Usage Methods

Through the GUI of UDA, users can quickly set the components. An overview of the blocks in the GUI is displayed in Figure 16, enabling easy setup of test patterns, read/write operations, and address increments/decrements. Once completed, the corresponding algorithm is generated.

Copyright © iSTART-TEK INC.

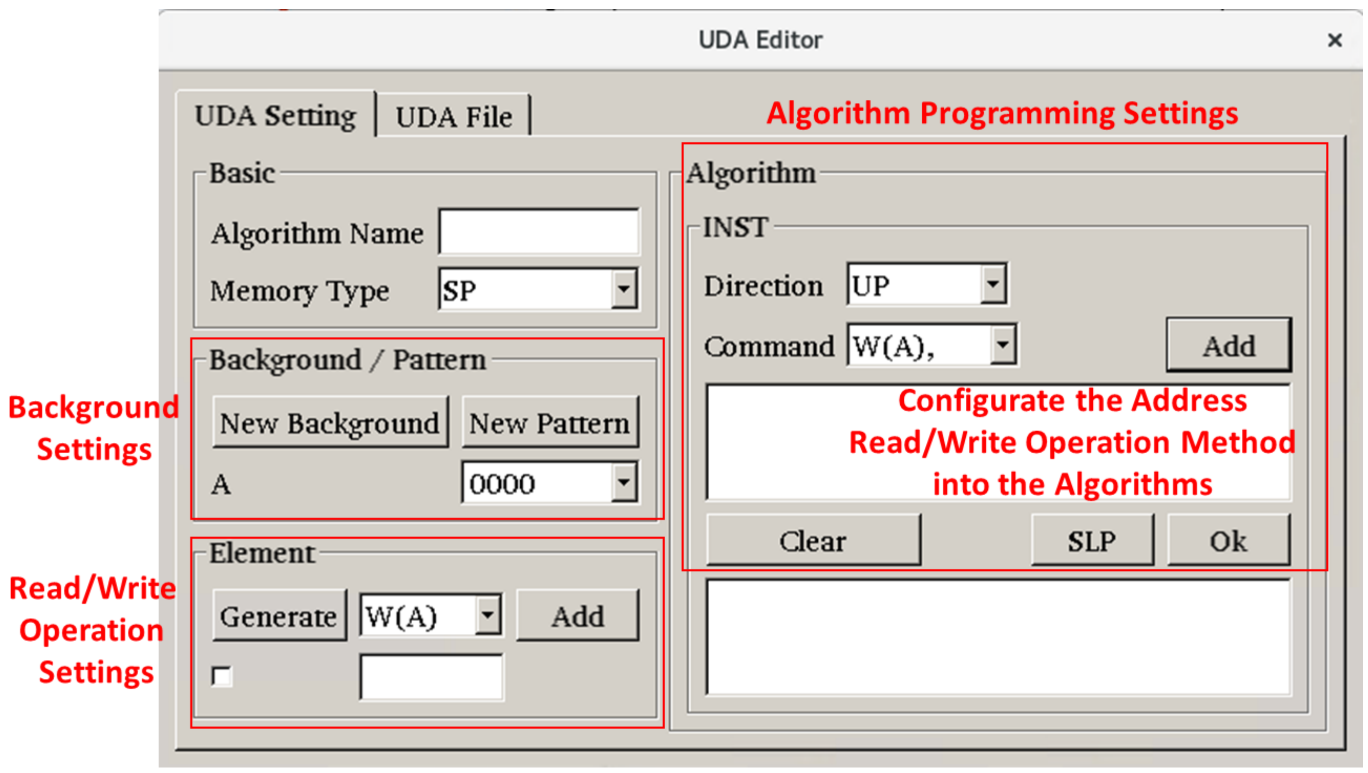
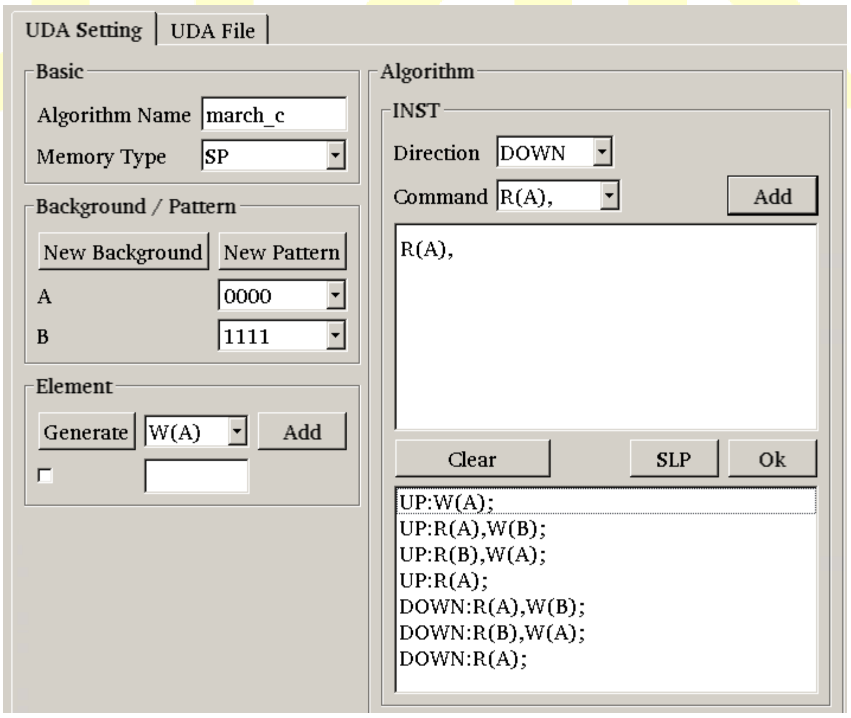


Figure 16 Overview of GUI

Taking a March C algorithm as an example: After completing the algorithm settings with the GUI, click the UDA file, and the algorithm setting results will be displayed. Then click "Export" to export the algorithm as a .txt file. Lastly, in the .bfl configuration file, specify the paths of the aforementioned .txt files. Once completed, the BIST circuits corresponding to the algorithm are generated as shown in Figures 17, 18, and 19.



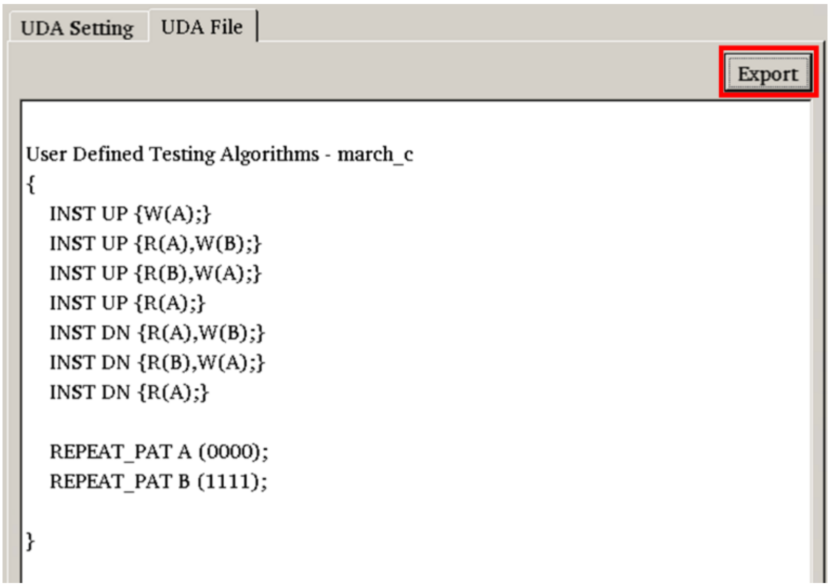
Algorithm Name: march_c
Memory Type: SP
A: 0000
B: 1111
Element: W(A)
Direction: DOWN
Command: R(A),

```
R(A),
```

```
UP:W(A);
UP:R(A),W(B);
UP:R(B),W(A);
UP:R(A);
DOWN:R(A),W(B);
DOWN:R(B),W(A);
DOWN:R(A);
```

Figure 17 Configuration of March C Algorithm

Copyright © iSTART-TEK INC.



```
User Defined Testing Algorithms - march_c
{
    INST UP {W(A);}
    INST UP {R(A),W(B);}
    INST UP {R(B),W(A);}
    INST UP {R(A);}
    INST DN {R(A),W(B);}
    INST DN {R(B),W(A);}
    INST DN {R(A);}

    REPEAT_PAT A (0000);
    REPEAT_PAT B (1111);

}
```

Figure 18 Export the Algorithm

```
define{user_define_algorithm}
  set SP_alg_path = ./UDA/uda_march_5w.txt
end_define{user_define_algorithm}
```

Figure 19 Configuration of a UDA File

Authored by iSTART-TEK

Copyright © iSTART-TEK INC.