START™
Quick Start Guide

v2.2

CONTENTS

LIST OF FIGURES

1. START™ Tool Environment Setup

1.1. The Contents in the START™ Package

The START™ tool package includes the following items:

- **Demo case:** NDAcase.tar.gz
  The demo case folder contains an example case. Users can use this demo case to get familiar with the START™ tool.

- **DOC**
  The DOC folder contains all START™-related documents for designers' reference.

- **License:**
  It contains the START™ tool license system. The file name is like LM-CentOS-x.x-x86_64-xxxx.tar.gz. Before manipulating the START™ tool, users have to set up this license to their license servers. After setting up ISTART_LICENSE_FILE environment variables and invoking the license file, users can launch the START™ tool successfully.

- **START™ tool:**
  In the START™ package, the tool file name is like START-CentOS-x.x-x86_64-develop-xxxx.tar.gz. Users can extract this tarball to the working server and setup alias. Then, users can utilize the START™ tool to generate BIST/BISR circuits and integrate them with users' designs.

1.2. Untar Tarballs in the START™ Package

There are two tarballs in the START™ package. Users can follow the instructions below to extract these tarballs and untar them in the Linux system.

- **START™ License Manager:**
  The name of the START ™ license manager is similar with LM-CentOS-x.x-x86_64-xxxxx.tar.gz. Users can create a folder in the license server to store this license tarball. Then, use the following command to extract.

  $ tar xvf LM-CentOS-x.x-x86_64-xxxxx.tar.gz

  After decompressing correctly, users can find the following files:
  (1) iSTART_lic_2023xxxxx.lic
  (2) lmgrd
  (3) lmutil
  (4) istart

- **START™ tool:**
  The name of the START ™ tool is similar with START-CentOS-x.x-x86_64-develop-xxxx.tar.gz. Users can create a folder in the workstation to store this tool tarball. Then, use the following command to extract.

  $ tar -xvzf START-CentOS-x.x-x86_64-develop-xxxx.tar.gz

1.3. Set Up the START™ License

- Please put istart together with lmgrd in the same folder.

- Execute the following command under the folder LM-CentOS-x.x-x86_64
  $ ./lmgrd -c iSTART_lic_2023xxxxx.lic

- The way to confirm START™ license launched:
  $ ./lmutil lmstat -a

1.3.1 Kill Previous Registered License

iSTART tools have already adopted a new Flexnet license system. If users have the previous license system in their own license server, please refer to the following steps to terminate the existing license and iSTART_LIC_FILE.

1. $ ps -ef | grep ‘lmInvoke’
2. $ kill #license thread
3. $ unset iSTART_LIC_FILE

1.4. Set Up the Environment

Set up the environment with ISTART_LICENSE_FILE for invoking the license server.

- **Bash Shell:**
  $ export ISTART_LICENSE_FILE=4141@hostname
  Or
  $ export ISTART_LICENSE_FILE=4141@IP

- **C Shell (Tcsh):**
  $ setenv ISTART_LICENSE_FILE 4141@hostname
  Or
  $ setenv ISTART_LICENSE_FILE 4141@IP

1.5. Set Up the Alias in START™

Set the START™ tool alias names to easily invoke START™ at any working folder. The following shows the START™ alias settings in Bash shell & C shell.

- **Bash Shell:**
  $ alias start=/usr/home/tools/START-CentOS-6.5-x86_64-xxxxx/bin/start

- **C Shell (Tcsh):**
  $ alias start /usr/home/tools/START-CentOS-6.5-x86_64-xxxxx/bin/start

1.6. iSTART License Update

Once the license cannot be invoked successfully, please use lmutil lmdown to turn off the license server. After the iSTART license is off, please execute lmgrd with the new license again.

```
$ ./lmutil lmdown
$ ./lmgrd -c iSTART_lic_2023xxxxx.lic
```

2. BFL Flow

If it is the first time executing START™, here is an example case for users to understand the BFL (BIST Feature List) flow. Please note that this evaluation package, NDAcase, is designed only for the design of a single clock domain.

2.1. Untar the Example Case

```
$ tar xvzf NDAcase.tgz
$ cd NDAcase
```

2.2. Check If the START™ Tool Workable

Use the following command under the execution folder.

```
$ start --help
```

```
-- (c) Copyright 2009 - 2019 by iSTART-Technologies, Inc.
-- All rights reserved

    START - SRAM Built-in Testing and Repairing Technology : ver. 2.0.13 build 2020.11
    Build 201202


This computer program constitutes or contains trade secrets and confidential
information of iSTART-Technologies Inc. or its licensors. This computer program is
protected by copyright law and international treaties.


usage: start [-h] [-bii INTEGRATE_FILE] [-bfl BFL_FILE]
             [-f RUN_FILE [RUN_FILE ...]] [-v VERILOG_FILE [VERILOG_FILE ...]]
             [-W DIR] [-top MODULE] [-I] [--genmeminfo]
             [-integ FILE [FILE ...]] [-u FILE [FILE ...]] [-pm Verilog type]
             [--integrator] [--faultfree] [--ug UDM_FILE config_FILE]
             [--rcfg Addr_length Data_width output_FILE] [--tempgen]
             [--memchecker] [--memlib2udm MEMLIB_FILE]
             [--bflconfig [BFL_FILE]] [--biiconfig [BII_FILE]]
             [--pathconv work_path] [--STILloopformat work_path]
             [--latchgo_hier latchgo_data meminfo] [--udmgui [UDMGUI]]
             [--meminfogui [MEMINFO]]

optional arguments:
  -h, --help            show this help message and exit
  -bii INTEGRATE_FILE   input BII file
  -bfl BFL_FILE         input BFL file
  -f RUN_FILE [RUN_FILE ...]
                        input run file(s)
  -v VERILOG_FILE [VERILOG_FILE ...]
                        input verilog file(s)
  -W DIR                specify working path
  -top MODULE, -T MODULE
                        specify top module
  -I, --insert          insert BIST to design
  --genmeminfo          Generate meminfo file
  -integ FILE [FILE ...]
                        input integ file(s)
  -u FILE [FILE ...], -udm FILE [FILE ...]
```

Figure 2-1 START™ Command Option

2.3. Create a FileList File (*.f)

The easiest way to execute START™ is to provide a complete design and FileList file (*.f). The format of a FileList file is the same as an NC-Verilog file which includes the following items.

- Design.v (RTL or Netlist)
- Memory.v
- Standard_cell.v (when the user's design is Netlist)
- Parameter, e.g., +define+, +incdir+PATH/DIR

Figure 2-2 shows an example of a *.f file. To run *.f for this test case, NDAcase, users should add the -v option in front of each memory Verilog file.

```
-v ./memory/rf_2p_24x28.v
-v ./memory/sram_sp_4096x64.v
-v ./memory/rom_6144_64.v
-v ./memory/rf_sp_128x22.v
-v ./memory/sram_dp_1024x64.v
-v ./memory/rf_2p_24x56.v
-v ./memory/sram_sp_2048x64.v
-v ./memory/sram_sp_640x32.v
-v ./memory/rf_2p_64x64.v
-v ./memory/rf_2p_72x14.v
-v ./memory/sram_sp_1024x32.v

./top.v
```

Figure 2-2 Example of a *.f File

2.4. Memory Checking by START™ (Optional)

START™ assists to identify users' memory macros by executing the **memchecker** command. This command can check if users' memory models can be recognized by START™. For details, please refer to Appendix in the end of the document.

If memory models cannot be recognized by START™, users can edit **UDM** (User Defined Memory) and then add these UDM files into the BFL file. START™ also provides a *.udm file template, and users can modify it according to the memory models. For the details, please refer to Chapter 2 in START™ User Manual.

2.5. Generate and Set a BFL File

$ cd NDAcase
$ start --tempgen

Please choose item 1 as Figure 2-3 shows. The MBIST/BISR Feature List (BFL) and the start_template.bfl file will be generated to the working folder.

```
-- (c) Copyright 2009 - 2023 by iSTART-Technologies, Inc.
-- All rights reserved

    START - SRAM Built-in Testing and Repairing Technology : ver. 3.0.61 build 2023.01
    Build 230209


This computer program constitutes or contains trade secrets and confidential
information of iSTART-Technologies Inc. or its licensors. This computer program is
protected by copyright law and international treaties.


[18:14:32]  START : ver. 3.0.61 build 2023.01 : (c) Copyright 2009- iSTART-Technologies. All rights reserved.

[START][TEMPLATE] START template generator :
            1. BIST Feature List (BFL)
            2. BIST Integration Information (BII)
            3. User defined memory
            4. Pattern Gen File (PGF)
            5. QUIT
[START][TEMPLATE] Select an option(Enter ':q' to quit):
```

Figure 2-3 Generate BFL File

A BFL file includes the related requirements of MBIST circuit specifications. Users can modify it based on their project requirements. Figure 2-4 and Figure 2-5 show the examples of BFL settings for this test case. Users can also refer to the ref folder in the test case package to find a BFL file example.

```
define{OPTION}
    set verilog_path        = ./run.f           # /absolute path/design.f
    set user_define_memory  =                   # /absolute path/memory.udm
    set top_module_name     = top               # design top
    set top_hierarchy       = top               # BIST top
    set clock_trace         = no                # yes, no (User group instances will all be un-group when setting yes)
    set auto_group          = yes               # yes, no
    set insertion           = yes               # yes, no
    set integrator_mode     = no                # yes, no
    set work_path           = ./mbist           # ./work
    set fault_free          = no                # yes, no
    set parsing_mode        = RTL_only          # RTL_only, Netlist_only
    set repair_prefix       = RP                # prefix for repair module

    define{CLOCK}
        set sdc_file                =           # /absolute path/design.sdc
        define{100MHz}
            set clock_cycle         = 10        # integer
            set clock_source_list   = top CLK1      # top design1 CLK
        end_define{100MHz}
        define{10MHz}
            set clock_cycle         = 100       #integer
            set clock_source_list   = top CLK2      # top design2 CLK
        end_define{10MHz}
    end_define{CLOCK}

    define{GROUP}
        set sequencer_limit = 60                # integer
        set group_limit     = 30                # integer smaller than sequencer limit
        set memory_list     = #./test.meminfo   # /absolute path/design.meminfo
        set time_hierarchy  = 1                 # 0(time) < value <1(hierarchy)
        set lib_path        =                   # /absolute path/lib (Accept file dictionary)
        set power_limit     = 1.0               # mW (float bigger than 0)
        set hierarchy_limit = 0                 # integer (default: 0)

        define{PHYSICAL}
            set enable_physical         = no    # yes, no
            set physical_location_file  =       # /absolute path/design.def
            set distance_limit          = 1
            set phycical_logical        = 0.5
        end_define{PHYSICAL}
    end_define{GROUP}
end_define{OPTION}
```

Figure 2-4 BFL File Example (1)

```
define{BIST}
    set STIL_test_bench             = no        # yes, no
    set asynchronous_reset          = yes       # yes, no
    set bist_interface              = ieee1500  # minibist, basic, ieee1149.1, ieee1500
    set address_fast_y              = no        # yes, no
    set program_algorithm           = no        # yes, no
    set algorithm_selection         = no        # no, outside, scan
    set background_style            = SOLID     # SOLID, WORD, 5A
    set background_bit_inverse      = no        # yes, no
    set background_col_inverse      = no        # yes, no
    set bypass_support              = no        # no, wire, reg
    set bypass_clock                = no        # yes, no
    set bypass_include_bist_pin     = no        # yes, no
    set bypass_reg_sharing          = 1         # 1 ~ 1024
    set clock_function_hookup       = no        # yes, no
    set clock_switch_of_memory      = yes        # yes, no
    set clock_source_gated          = no        # yes, no
    set clock_source_switch         = no        # yes, no
    set clock_within_pll            = no        # yes, no
    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
    set diagnosis_faulty_items      = algorithm, operation, element, seq_id, grp_id, address, ram_data,
    set parallel_on                 = no        # yes, no
    set reduce_address_simulation   = no        # yes, no
    set rom_half_access             = no        # 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 repair_mode                 = yes       # yes, no. Repair mode with redundancy memory model.
    set soft_repair                 = no        # yes, no. yes = soft repair, no = hard repair
```

Figure 2-5 BFL File Example (2)

2.6. Execute START™ with the BFL File

The command to execute START™ with the BFL file is:

$ start -bfl start_template.bfl

Please note that if the location of files defined in the BFL file is a relative path instead of an absolute path, the relative path is based on the location of the BFL file.

After executing the commands above, users can see messages like Figure 2 6 and Figure 2 7 show. All the generated files will be output to the mbist folder. Users can find the start_memory_spec.meminfo file in the mbist folder, which represents the grouping architecture.

```
[16:36:33]  [CHECK][GROUPING] top_default: seq 1, grp 1, 8 members
[16:36:33]  [CHECK][GROUPING] top_default: seq 2, grp 1, 1 members
[16:36:33]  [CHECK][GROUPING] top_default: seq 3, grp 1, 2 members
[16:36:33]  [CHECK][GROUPING] top_default: seq 4, grp 1, 1 members
```

Figure 2-6 Grouping Information

```
[16:36:34]  [INSERT]
[16:36:34]  [INSERT] #======================  BIST Insert Path  ======================#
[16:36:34]  [INSERT] #                                                                #
[16:36:34]  [INSERT] #                     ----- Controller -----                     #
[16:36:34]  [INSERT] #                                                                #
[16:36:34]  [INSERT] #  CTR(top_default) : top                                        #
[16:36:34]  [INSERT] #                                                                #
[16:36:34]  [INSERT] #                     ----- Sequencer -----                      #
[16:36:34]  [INSERT] #                                                                #
[16:36:34]  [INSERT] #  SEQ 1 : top.u_t1                                              #
[16:36:34]  [INSERT] #  SEQ 2 : top.u_t1                                              #
[16:36:34]  [INSERT] #  SEQ 3 : top.u_t1                                              #
[16:36:34]  [INSERT] #  SEQ 4 : top.u_t1                                              #
[16:36:34]  [INSERT] #                                                                #
[16:36:34]  [INSERT] #                        ----- TPG -----                         #
[16:36:34]  [INSERT] #                                                                #
[16:36:34]  [INSERT] #  TPG top_default_t_1_1_1 : top.u_t1   [sram_sp_1024x32] (ram_1)  #
[16:36:34]  [INSERT] #  TPG top_default_t_1_1_2 : top.u_t1   [sram_sp_1024x32] (ram_2)  #
[16:36:34]  [INSERT] #  TPG top_default_t_1_1_3 : top.u_t1   [sram_sp_1024x32] (ram_3)  #
[16:36:34]  [INSERT] #  TPG top_default_t_1_1_4 : top.u_t1   [sram_sp_1024x32] (ram_4)  #
[16:36:34]  [INSERT] #  TPG top_default_t_1_1_5 : top.u_t1   [sram_sp_1024x32] (ram_e)  #
[16:36:34]  [INSERT] #  TPG top_default_t_1_1_6 : top.u_t1   [sram_sp_1024x32] (ram_w)  #
[16:36:34]  [INSERT] #  TPG top_default_t_1_1_7 : top.u_t1   [sram_sp_1024x32] (ram_x)  #
[16:36:34]  [INSERT] #  TPG top_default_t_1_1_8 : top.u_t1   [sram_sp_1024x32] (ram_y)  #
[16:36:34]  [INSERT] #  TPG top_default_t_2_1_1 : top.u_t1   [rf_2p_24x28] (u_2p)       #
[16:36:34]  [INSERT] #  TPG top_default_t_3_1_1 : top.u_t1   [sram_dp_1024x64] (u_dp)   #
[16:36:34]  [INSERT] #  TPG top_default_t_3_1_2 : top.u_t1   [sram_dp_1024x64] (u_dp2)  #
[16:36:34]  [INSERT] #  TPG top_default_t_4_1_1 : top.u_t1   [rom_6144_64] (u_rom)      #
[16:36:34]  [INSERT] #                                                                #
[16:36:34]  [INSERT] #                        ----- END -----                         #
[16:36:34]  [INSERT] #                                                                #
[16:36:34]  [INSERT] #================================================================#
[16:36:34]  [INSERT]
[16:36:34]  [INSERT] Perform auto insertion ... done (0.11 sec)
```

Figure 2-7 Auto-Insertion Information

2.7. Setting a Memory Info File (Optional)

After executing START™, the memory info file will be output to the mbist folder. A memory info file represents the grouping architecture. If users want to adjust memory grouping according to their design requirements, modify the memory info file directly.

A memory info file includes the following items. For the detailed information, please refer to Chapter 7 in Application Notes.

Clock Domain:	Memory clock domain and testing the clock cycle
Memory Module:	Memory module name and memory hierarchy
Bypass/Diagnosis:	Setting the values of the bypass function and diagnosis function
q_pipeline:	Setting the value of the q_pipeline option
Group Architecture:	Grouping architecture information (including the controller and sequencer)

Figure 2-8 shows the content of START_memory_spec.meminfo.

```
[DOMAIN=top_default, cycle=100.0ns]
     [CTR] # Hier: top
         [SEQ] # No.= 1,InstanceNo= 8,SEQ_max_addr_size= 1024,Hier: top u_t1
             [GROUP] # No.=1_1
                   [SP=1_1_1, byp=no, diag=no, q_pipe=no]sram_sp_1024x32                    top u_t1 ram_1
                   [SP=1_1_2, byp=no, diag=no, q_pipe=no]sram_sp_1024x32                    top u_t1 ram_2
                   [SP=1_1_3, byp=no, diag=no, q_pipe=no]sram_sp_1024x32                    top u_t1 ram_3
                   [SP=1_1_4, byp=no, diag=no, q_pipe=no]sram_sp_1024x32                    top u_t1 ram_4
                   [SP=1_1_5, byp=no, diag=no, q_pipe=no]sram_sp_1024x32                    top u_t1 ram_e
                   [SP=1_1_6, byp=no, diag=no, q_pipe=no]sram_sp_1024x32                    top u_t1 ram_w
                   [SP=1_1_7, byp=no, diag=no, q_pipe=no]sram_sp_1024x32                    top u_t1 ram_x
                   [SP=1_1_8, byp=no, diag=no, q_pipe=no]sram_sp_1024x32                    top u_t1 ram_y
         [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]rf_2p_24x28                        top u_t1 u_2p
         [SEQ] # No.= 3,InstanceNo= 2,SEQ_max_addr_size= 1024,Hier: top u_t1
             [GROUP] # No.=3_1
                   [DP=3_1_1, byp=no, diag=no, q_pipe=no]sram_dp_1024x64                    top u_t1 u_dp
                   [DP=3_1_2, byp=no, diag=no, q_pipe=no]sram_dp_1024x64                    top u_t1 u_dp2
         [SEQ] # No.= 4,InstanceNo= 1,SEQ_max_addr_size= 6144,Hier: top u_t1
             [GROUP] # No.=4_1
                   [ROM=4_1_1, byp=no, diag=no, q_pipe=no]rom_6144_64                        top u_t1 u_rom
```

Figure 2-8 Memory Info Setting Information

2.8. Using a Memory Info File as Default Memory Grouping

If users use a memory info file, START_memory_spec.meminfo, as memory grouping setting, they should turn off auto_group option and specify memory_list option to the path of START_memory_spec.meminfo in BFL configuration file as Figure 2-9 shows.

After executing the START™ BFL flow with the memory info file, START™ can automatically modify the naming and operating frequency of the BIST and BISR controllers. It also assists users to do grouping-related settings according to their requirements. There is a memory info file example in the ref folder of NDAcase. Execute START™ with the modified BFL file which includes the modified memory info file and commands as Figure 2-9. The prompts will appear as Figure 2-10 and Figure 2-11.

In this example case, there is one extra group for sequencer 1 and the name of the BIST controller is changed to testcase.

```
$ start -bfl start_template.bfl
```

```
define{GROUP}
    set sequencer_limit = 60                # integer
    set group_limit     = 30                # integer smaller than sequencer limit
    set memory_list     = ./test.meminfo             # /absolute path/design.meminfo
    set time_hierarchy  = 1              # 0(time) < value <1(hierarchy)
    set lib_path        =                   # /absolute path/lib (Accept file dictionary)
    set power_limit     = 1.0               # mW (float bigger than 0)
    set hierarchy_limit = 0                 # integer (default: 0)
```

Figure 2-9 memory_list Option

```
[17:39:24]  [CHECK][GROUPING] testcase: seq 1, grp 1, 5 members
[17:39:24]  [CHECK][GROUPING] testcase: seq 1, grp 2, 3 members
[17:39:24]  [CHECK][GROUPING] testcase: seq 2, grp 1, 1 members
[17:39:24]  [CHECK][GROUPING] testcase: seq 3, grp 1, 2 members
[17:39:24]  [CHECK][GROUPING] testcase: seq 4, grp 1, 1 members
```

Figure 2-10 Grouping Information with Memory Info File

```
[17:39:25]  [INSERT]
[17:39:25]  [INSERT] #=======================  BIST Insert Path  ========================#
[17:39:25]  [INSERT] #                                                                  #
[17:39:25]  [INSERT] #                     ----- Controller -----                       #
[17:39:25]  [INSERT] #                                                                  #
[17:39:25]  [INSERT] #  CTR(testcase) : top                                             #
[17:39:25]  [INSERT] #                                                                  #
[17:39:25]  [INSERT] #                     ----- Sequencer -----                        #
[17:39:25]  [INSERT] #                                                                  #
[17:39:25]  [INSERT] #  SEQ 1 : top.u_t1                                                #
[17:39:25]  [INSERT] #  SEQ 2 : top.u_t1                                                #
[17:39:25]  [INSERT] #  SEQ 3 : top.u_t1                                                #
[17:39:25]  [INSERT] #  SEQ 4 : top.u_t1                                                #
[17:39:25]  [INSERT] #                                                                  #
[17:39:25]  [INSERT] #                        ----- TPG -----                           #
[17:39:25]  [INSERT] #                                                                  #
[17:39:25]  [INSERT] #  TPG testcase_t_1_1_1 : top.u_t1   [sram_sp_1024x32] (ram_1)     #
[17:39:25]  [INSERT] #  TPG testcase_t_1_1_2 : top.u_t1   [sram_sp_1024x32] (ram_2)     #
[17:39:25]  [INSERT] #  TPG testcase_t_1_1_3 : top.u_t1   [sram_sp_1024x32] (ram_3)     #
[17:39:25]  [INSERT] #  TPG testcase_t_1_1_4 : top.u_t1   [sram_sp_1024x32] (ram_4)     #
[17:39:25]  [INSERT] #  TPG testcase_t_1_1_5 : top.u_t1   [sram_sp_1024x32] (ram_e)     #
[17:39:25]  [INSERT] #  TPG testcase_t_1_2_1 : top.u_t1   [sram_sp_1024x32] (ram_w)     #
[17:39:25]  [INSERT] #  TPG testcase_t_1_2_2 : top.u_t1   [sram_sp_1024x32] (ram_x)     #
[17:39:25]  [INSERT] #  TPG testcase_t_1_2_3 : top.u_t1   [sram_sp_1024x32] (ram_y)     #
[17:39:25]  [INSERT] #  TPG testcase_t_2_1_1 : top.u_t1   [rf_2p_24x28] (u_2p)          #
[17:39:25]  [INSERT] #  TPG testcase_t_3_1_1 : top.u_t1   [sram_dp_1024x64] (u_dp)      #
[17:39:25]  [INSERT] #  TPG testcase_t_3_1_2 : top.u_t1   [sram_dp_1024x64] (u_dp2)     #
[17:39:25]  [INSERT] #  TPG testcase_t_4_1_1 : top.u_t1   [rom_6144_64] (u_rom)         #
[17:39:25]  [INSERT] #                                                                  #
[17:39:25]  [INSERT] #                        ----- END -----                           #
[17:39:25]  [INSERT] #                                                                  #
[17:39:25]  [INSERT] #==================================================================#
[17:39:25]  [INSERT]
[17:39:25]  [INSERT] Perform auto insertion ... done (0.08 sec)
```

Figure 2-11 Auto-Insertion Information with Memory Info File

3. Simulation

3.1. Self-Simulation

Figure 3-1 shows the architecture of the testbench for self-simulation. This self-simulation is used to verify the function correctness of BIST and BISR circuits only. This system design is not included in self-simulation. The simulation environment is built by the make language. Users can refer to the Makefile.top_default file. This file defines commands and parameters for executing simulation.

If users want to debug with the waveform file, turn on the related dump parameters in the top_default.f file.

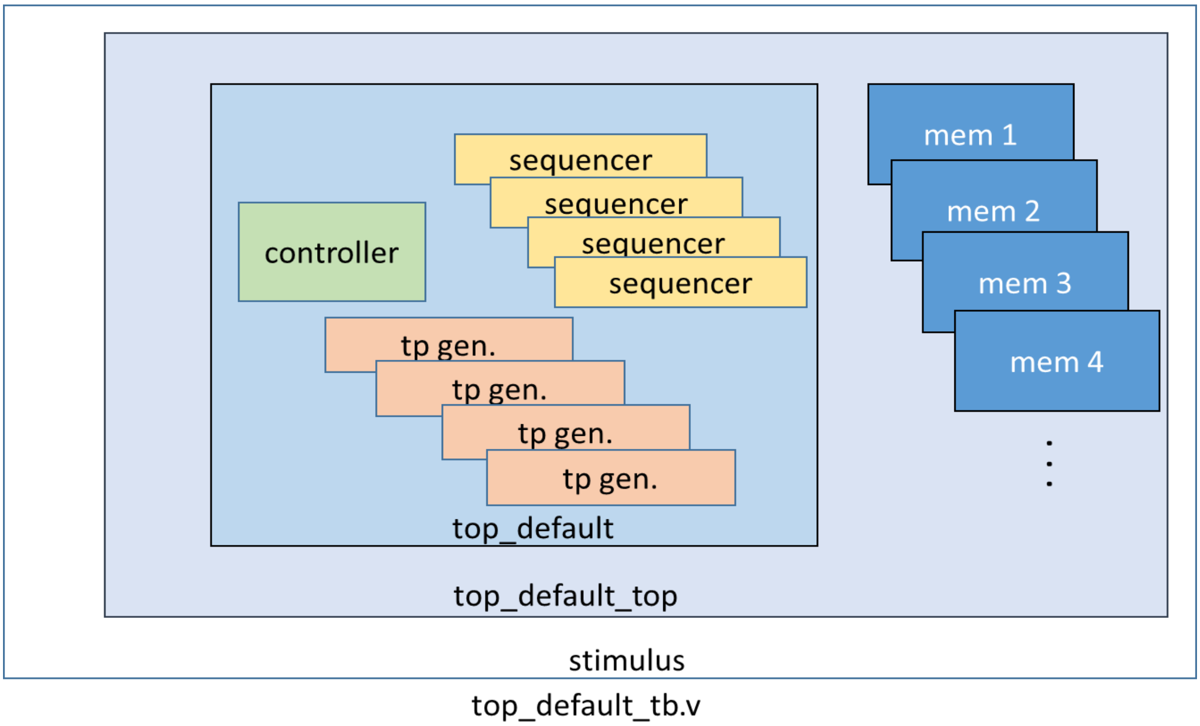

Figure 3-1 Testbench Architecture of Self-Simulation

If users adjust the clock domain, check the difference of the output file in the mbist folder. In a test case, the controller's name of the default clock domain is top_default.

The command for self-simulation is:

$ make top_default FUNC=tb

If the timeout message, "Simulation time-out!" appears during self-simulation, users can modify the delay parameter of the block "`ifndef FAULT" in the top_default_tb.v file as Figure 3-2 shows. This delay parameter is generated by START™ and is designed to prevent an infinite loop. Figure 3-3 shows the simulation results of self-simulation.

Note: If the user's design includes an ROM memory inside, please check the path setting of the ROM code file before executing simulation.

```
`ifndef FAULT
  initial begin
    #(cyc*414611);

    $display("\nSimulation time-out!\n");
    $finish;
  end
`endif
```

Figure 3-2 Delay Parameter

```
Loading snapshot worklib.stimulus:v .................... Done
*Verdi3* Loading libsscore_ius142.so
*Verdi3* : Enable Parallel Dumping.
ncsim> source /usr/cad/cadence/IUS/cur/tools/inca/files/ncsimrc
ncsim> run

 Test Result: Pass!

Simulation complete via $finish(1) at time 39383600 NS + 0
./top_default_tb.v:154   $finish;
ncsim> exit
make[1]: Leaving directory `/home/jeremy/TestCase/NDAcase/mbist'
```

Figure 3-3 Self-Simulation Result

3.2. Inserted Simulation

Figure 3-4 shows the architecture of the testbench for the inserted simulation. The inserted simulation is to verify the function correctness of the inserted design which combines BIST circuits and users' system design. The simulation environment is built by the make language. Users can refer to the Makefile.top_default file. This file defines commands and parameters for executing simulation.

If users want to debug with a waveform file, please turn on the related dump parameters in the top_default_INS_FAULT.f file, which is the same as self-simulation. If there are several clock domains, each clock domain should be passed when doing the inserted simulation.

The command of the inserted simulation is:

$ make top_default FUNC=tb_INS

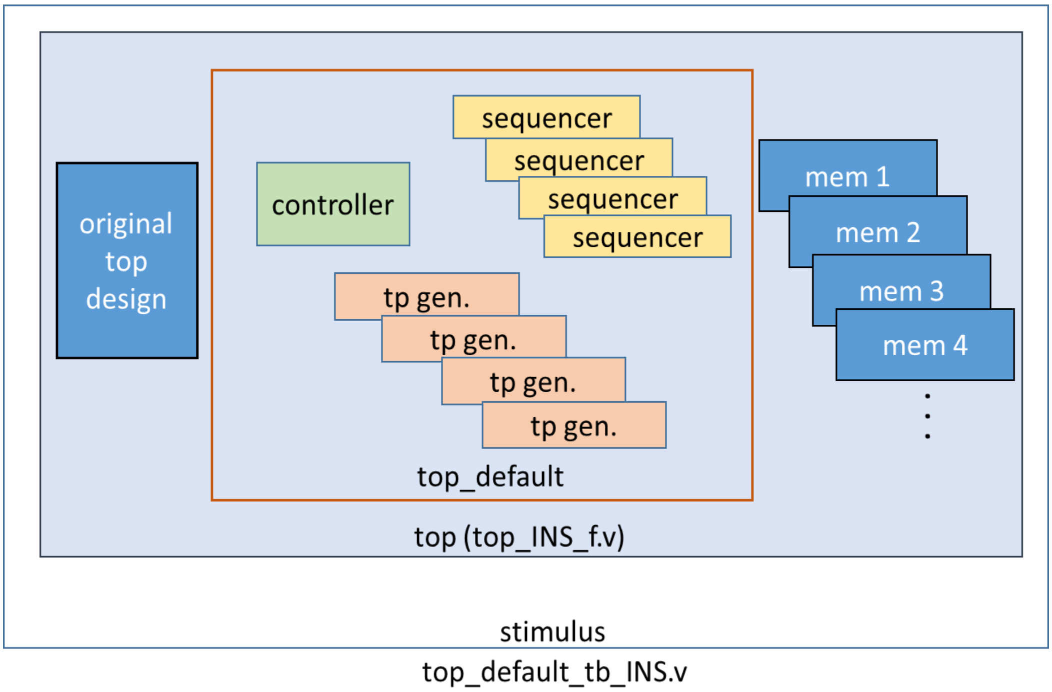

Figure 3-4 Testbench Architecture of Inserted Simulation

If the timeout message, "Simulation time-out!" appears during the period of executing simulation, users can modify the delay parameter of the block "`ifndef FAULT" in top_default_tb_INS.v. Figure 3-5 shows the prompt of the inserted simulation.

```
Loading snapshot worklib.stimulus:v .................... Done
*Verdi3* Loading libsscore_ius142.so
*Verdi3* : Enable Parallel Dumping.
ncsim> source /usr/cad/cadence/IUS/cur/tools/inca/files/ncsimrc
ncsim> run

 Test Result: Pass!

Simulation complete via $finish(1) at time 39383600 NS + 0
./top_default_tb_INS.v:188   $finish;
ncsim> exit
make[1]: Leaving directory `/home/jeremy/TestCase/NDAcase/mbist'
```

Figure 3-5 Inserted Simulation Result

3.3. Simulation with the Repair Function

Users can do the inserted simulation with the repair function when the repair mode is enabled. The simulation environment is built by the make language. Please refer to Makefile.RP_default file which defines commands and parameters for executing simulation.

Debugging with the waveform file, turn on the related dump parameters in the RP_default_INS_FAULT.f file, which is the same as the general inserted simulation. If there are several clock domains, each clock domain should be passed when doing the inserted simulation.

The command of inserted simulation with repair function:

$ make RP_default FUNC=tb_INS_RP

3.4. Simulation with Fault Memory Models

START™ can automatically generate fault memory models to verify the functional correctness of BIST circuits. These models can be found in the FAULT_MEMORY directory. Use the commands below to execute simulation with these models.

These operations will use fault_memory.f in the FAULT_MEMORY folder.

For self-simulation:

$ make top_default FUNC=tb_f

For inserted simulation:

$ make top_default FUNC=tb_INS_f

When executing this type of simulation, it will show a **failed prompt**. This is caused by pre-defined error bits in fault memory models. The simulation waveform can be viewed for users to understand the behavior of START™ designs and fault memory models. Figure 3-6 shows an example of running simulation with fault memory models. In this case, users can find the access sequence of the memories in **group 1** (1_1_8, sram_sp_640x32 memory model).

(1) Write access with data 32’hffff_ffff to address 10’h350
(2) Read access from address 10’h350
(3) Read data 32’hfffd_ffff

The data of reading does not equal to the data of writing in **“A”** in Figure 3-6 and this wrong behavior has caused the simulation to fail.

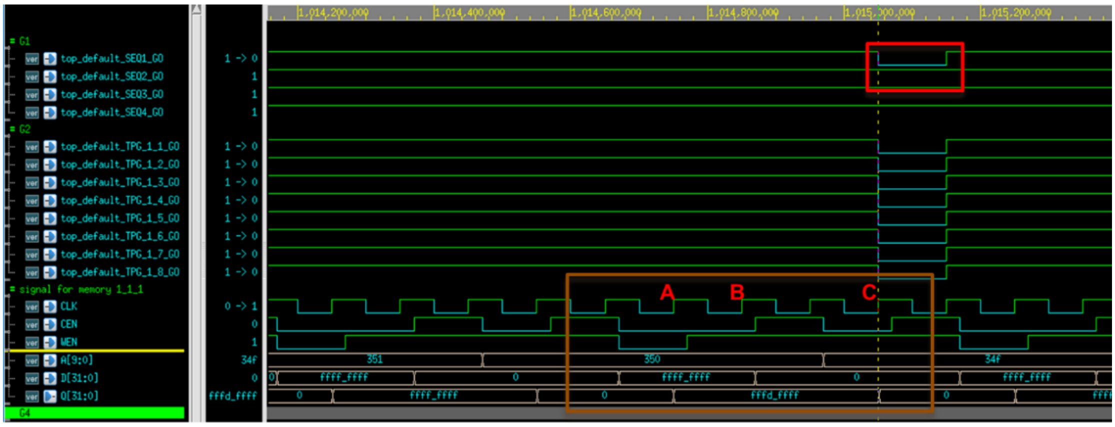

Figure 3-6Simulation Waveform of Fault Memory Models

Users can find pre-defined error bits in fault memory models. Figure 3-7 is an example of a sram_sp_1024x32 memory model in the FAULT_MEMORY directory.

```
module sram_sp_1024x32_f(
  Q,
  CLK,
  CEN,
  WEN,
  A,
  D,
  EMA,
  RETN
);
  integer   _addr;
  parameter _BITS = 32;
  parameter _sa_fault = 1'b0; // sa0
  parameter _faulty_bit = 17;
  parameter _faulty_addr = 10'h350;
```

Figure 3-7 Example of Error Bit Definitions

4. Synthesis

START™ also provides a synthesis script for BIST circuits. Users can find it in the output directory, named [design_name].tcl. Before executing synthesis, the related settings including the library path, standard cell type and path of the memory library file should be completed. If there are different clock domains, each clock domain should undergo synthesis.

START™ provides a referenced synthesis script in the mbist folder. The command of synthesis is:

$ make top_default FUNC=dc

Figure 4-1 shows the prompt during the execution of the synthesis command. After synthesis is completed, users can find the synthesis results including area and timing reports in the REPORT folder.

```
write_sdc ${WORK_PATH}/${TOP}_netlist.sdc
1
#/*************** worst case timing report *******************/
redirect ${WORK_PATH}/REPORT/${TOP}_maxtiming.rpt { report_timing -nets -delay max -max_paths 5 -transition_time -nosplit }
redirect ${WORK_PATH}/REPORT/${TOP}_mintiming.rpt { report_timing -nets -delay min -max_paths 5 -transition_time -nosplit }
redirect ${WORK_PATH}/REPORT/${TOP}_looptim.rpt { report_timing -loops -max_paths 5 }
#/******************** area report *******************/
redirect ${WORK_PATH}/REPORT/${TOP}_area.rpt { report_area -hier }
redirect -append ${WORK_PATH}/REPORT/${TOP}_area.rpt { report_reference }
redirect ${WORK_PATH}/REPORT/${TOP}_power.rpt {report_power}
redirect ${WORK_PATH}/REPORT/${TOP}_qor.rpt {report_qor}
#/******************** write_script *******************/
write_script -output ${WORK_PATH}/REPORT/${TOP}_scrip.rpt
1
#/******************** all violation *****************/
redirect ${WORK_PATH}/REPORT/${TOP}_constraint.rpt { report_constraint -all_violators -verbose -nosplit }
#/******************** check_design *****************/
#redirect ${WORK_PATH}/REPORT/${TOP}_check_design.rpt { check_design }
#/******************** check_test *****************/
#redirect ${WORK_PATH}/REPORT/${TOP}_check_test.rpt { check_design }
#/******************** check_timing *****************/
#redirect ${WORK_PATH}/REPORT/${TOP}_check_timing.rpt { check_timing }
exit

Memory usage for main task 226 Mbytes.
Memory usage for this session 226 Mbytes.
CPU usage for this session 13 seconds ( 0.00 hours ).

Thank you...
make[1]: Leaving directory `/mnt/raid/home/jeremy/LAB/NDAcase/work'
```

Figure 4-1 Synthesis Output of top_default Controller

5. Appendix: Memchecker Usage

The appendix introduces how to do memory checking with START™ memchecker option. This can make sure if the customer's memory models can be recognized by the START™ tool.

START™ assists to identify memory macros in customer's design by executing the memchecker command. Here is an example to identify memories and output results in the memck folder.

```
$ cd NDAcase
$ start --memchecker -N -W memck -v run.f
```

Users can also identify the memories in the memory folder directly.

```
$ cd NDAcase/memory
$ start --memchecker -N -W memck -v *.v
```

Figure 5-1 shows the output message of the memchecker command.

```
Input file(s):
  [1] /home/jeremy/LAB_e/NDAcase/memory/rom_6144_64.v
  [2] /home/jeremy/LAB_e/NDAcase/memory/sram_sp_4096x64.v
  [3] /home/jeremy/LAB_e/NDAcase/memory/sram_sp_640x32.v
  [4] /home/jeremy/LAB_e/NDAcase/memory/rf_sp_128x22.v
  [5] /home/jeremy/LAB_e/NDAcase/memory/rf_2p_72x14.v
  [6] /home/jeremy/LAB_e/NDAcase/top.v
  [7] /home/jeremy/LAB_e/NDAcase/memory/sram_sp_2048x64.v
  [8] /home/jeremy/LAB_e/NDAcase/memory/rf_2p_64x64.v
  [9] /home/jeremy/LAB_e/NDAcase/memory/sram_dp_1024x64.v
  [10] /home/jeremy/LAB_e/NDAcase/memory/rf_2p_24x28.v
  [11] /home/jeremy/LAB_e/NDAcase/memory/rf_2p_24x56.v
  [12] /home/jeremy/LAB_e/NDAcase/memory/sram_sp_1024x32.v

Valid file(s):
  [1] /home/jeremy/LAB_e/NDAcase/memory/rom_6144_64.v
  [2] /home/jeremy/LAB_e/NDAcase/memory/sram_sp_4096x64.v
  [3] /home/jeremy/LAB_e/NDAcase/memory/sram_sp_640x32.v
  [4] /home/jeremy/LAB_e/NDAcase/memory/rf_sp_128x22.v
  [5] /home/jeremy/LAB_e/NDAcase/memory/rf_2p_72x14.v
  [6] /home/jeremy/LAB_e/NDAcase/memory/sram_sp_2048x64.v
  [7] /home/jeremy/LAB_e/NDAcase/memory/rf_2p_64x64.v
  [8] /home/jeremy/LAB_e/NDAcase/memory/sram_dp_1024x64.v
  [9] /home/jeremy/LAB_e/NDAcase/memory/rf_2p_24x28.v
  [10] /home/jeremy/LAB_e/NDAcase/memory/rf_2p_24x56.v
  [11] /home/jeremy/LAB_e/NDAcase/memory/sram_sp_1024x32.v

Unrecognized file(s):
  [1] /home/jeremy/LAB_e/NDAcase/top.v
```

Figure 5-1 Memcheker Information

Contact Information

If there are any questions or comments, please contact iSTART-TEK at support@istart-tek.com. The following information might be included in the mail.

- ★ Document title
- ★ Document version
- ★ Page number
- ★ Simple and clear descriptions of the problem

Any suggestions for improvements are welcome.