

SCAAP & IOWRAP GUIDE



Test & Implementation

Created by
Huong Dam

Released Date
31st_May_2009



Revision History

Rev.	Modified contents	Approval	Reviewed by	Created by
1.0	New creation		ToanNguyen	Huong Dam
1.1	Update purpose of each content and some incorrect points as Mido-san comments		ToanNguyen	HuongDam



Reference Manuals

No.	Title name	Description	Path
1	SCAAP test design guide (Ver 1.0)		
2	SCAAP User's manual SCL library creation procedure (Ver1.0)		
3	SCAAP User's manual Test logic addition procedure (v05r0300)		
4	IOWrap pattern generation tool (IOWRAPGEN – V2.04)		
5	PLASMA tool manual (Ver 1.3)		
6	BSD_compiler.pdf (Tnl's document)		

Table of Contents

I. SCAAP overview

(Scan circuit automation addition program).....	7
1. Some basic knowledge about boundary scan method.....	7
1.1 Introduction about boundary scan method.....	7
1.2 Introduction about boundary scan circuit.....	7
1.3 Introduction about boundary scan cell.....	9
1.3.1 Structure of boundary scan cell – general model.....	9
1.3.2 Operation of BS register general model.....	10
1.3.3 Types of BS register for each pin.....	11
1.3.4 The position of BS register after being inserted in netlist.....	13
1.3.5 Using BS register as MUX scan register.....	15
1.4 Introduction about types of TAP module.....	17
1.5 IO control logic corresponding JTAG.....	18
2. Introduction about SCAAP tool	20
2.1 SCAAP function 1: Adding boundary scan circuit.....	20
2.1.1 SCAAP execution only boundary scan method.....	20
2.1.2 SCAAP execution with SINGEN method.....	21
2.1.3 SCAAP execution with embedded TAP controller corresponding to EXMBIST method.....	23
2.2 SCAAP function 2: Adding module test logic.....	25
2.2.1 Introduction about module test logic.....	25
2.2.2 SCAAP execution with module test method.....	25
2.3 SCAAP function 3: Adding shared pin logics.....	26
2.3.1 Introduction about shared pin logic.....	26
2.3.2 SCAAP execution with shared pin logics.....	28
2.4 SCAAP function 4: Adding IO state control logic for various test.....	30
2.4.1 Introduction about IO state control logic.....	30
2.4.2 SCAAP execution with IO state control logics.....	31
2.5 SCAAP function 5: Adding various-purpose test logics.....	32
II. SCAAP EXECUTION.....	34
1. INPUTs.....	34
1.1 Verilog library file.....	34
1.2 SCL libraries.....	35
1.3 PM file (Pin multi table).....	36
1.3.1 Valid data table area definition (key info.).....	37
1.3.2 Pin information.....	37
1.3.3 Boundary scan testing information.....	41

1.4 Control files (CTLF).....	46
1.4.1 Product information.....	46
1.4.2 Boundary scan test information.....	47
1.4.3 Execution test mode of SCAAP.....	48
1.4.4 Information about additional logics.....	48
1.4.5 Information about function signals will be added by SCAAP.....	51
2. Execution environment.....	56
2.1 Structure of SCAAP environment.....	56
2.2 SCAAP working flow.....	57
2.3 SCAAP run file.....	57
2.3.1 Option : GMODE - MUST.....	58
2.3.2 Option: GSET - OPTIONAL.....	58
2.3.3 Option: TARGET – MUST.....	59
2.3.4 Option: TCL - OPTIONAL.....	59
2.3.5 Options related to user information.....	59
2.3.6 Option: SCOPE - MUST.....	59
2.3.7 Options for defining prefix	59
2.3.8 Options for VCC/GND definition.....	60
2.3.9 Options for input files.....	60
2.3.10 Options for output files.....	60
3. OUTPUTs.....	61
3.1 Log files.....	61
3.1.1 SMRY file.....	61
3.1.2 ERROR file.....	65
3.1.3 INFO file.....	67
3.1.4 LOG file.....	68
3.2 BSDL file.....	69
3.3 ILOC file.....	72
3.4 PIF file.....	73
3.5 BSRO file (BS order file).....	74
III.IOWRAPGEN.....	75
1. IOWRAPGEN keywords.....	75
2. Types of Boundary scan test patterns.....	77
2.1 BS Function confirmation test	77
2.2 CLAMP instruction test	79
2.3 HighZ confirmation test.....	80
2.4 DCP test.....	81
2.5 IOWRAP test	83
2.6 BS chain verification test	85

3. IOWRAPGEN Inputs/Outputs.....	87
3.1 Timeplate file (.timeplate).....	88
3.2 Initial pattern generation file (.initial).....	90
3.3 Modifying BSDL file	91
3.3.1 Updating information about differential motion pins.....	91
3.3.2 Modifying information about TAP pins.....	93
3.3.3 Modifying to generate patterns where added BS cell is disregarded.....	94
4. IOWRAPGEN execution.....	95
4.1 Environment.....	95
4.2 Execution flow	96
4.3 IOWrap run file.....	97
IV. Simulation flow.....	99
1. Verification of DIAGNOSTIC pattern.....	100
2. Verification of IOWrap pattern.....	100

I. SCAAP OVERVIEW

(SCAN CIRCUIT AUTOMATION ADDITION PROGRAM)

This overview part describes some basic knowledge about boundary scan method and introduction about main functions of SCAAP tool. It helps to have the background to understand the technique of inserting boundary scan circuit by SCAAP.

1. Some Basic Knowledge About Boundary Scan Method

1.1 Introduction about boundary scan method

Boundary-scan is an integrated method for testing interconnects on printed circuit boards that is implemented at the IC level without using physical test probes.

Specification for boundary-scan testing is developed in the 1980s by the Joint Test Action Group (JTAG) and is standardized as IEEE 1149.1. So in most designs in Renesas, boundary scan circuit has been named JTAG module.

1.2 Introduction about boundary scan circuit

Boundary scan circuit includes 2 parts:

- (1) boundary scan chain: is chain of boundary scan cells, start from TDI (boundary scan in port) through shift-in pin and shift-out pin of boundary scan cells ... end at TDO (boundary scan out port)
- (2) TAP (test access port) controller: is the module used to control the operation of boundary scan chains. There are 5 main ports in this module:
 TDI: Test Data In , TDO: Test Data Out
 TMS: Test Mode Select , TCK: Test Clock , TRST: Test Reset
 IDCODE register is 32bit register contains the publication code of the chip
 BYPASS register: is 1bit register used to run boundary scan test in bypass mode
 Instruction register is 8bit register it is used to specify the operation mode of boundary scan test. (the operation mode will be explained more clearly in IOWRAPGEN part)
 Decoder is module used to decode the data in and out of boundary scan chain.

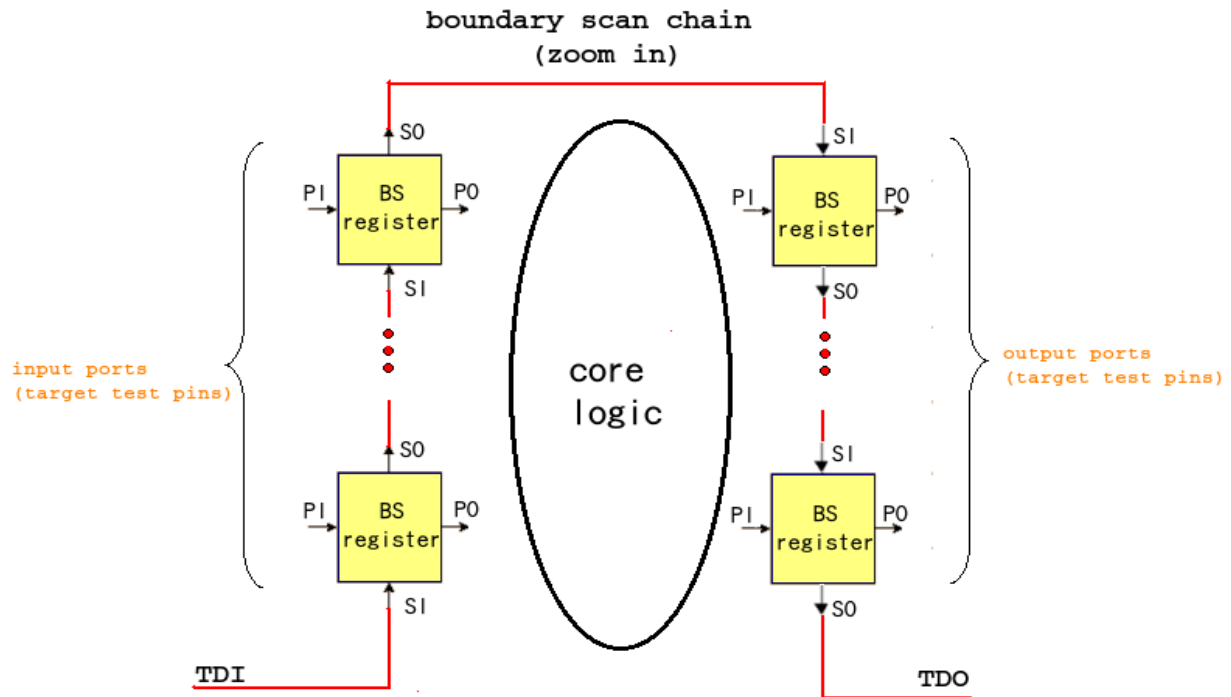


Figure 1: boundary scan register in boundary scan chain

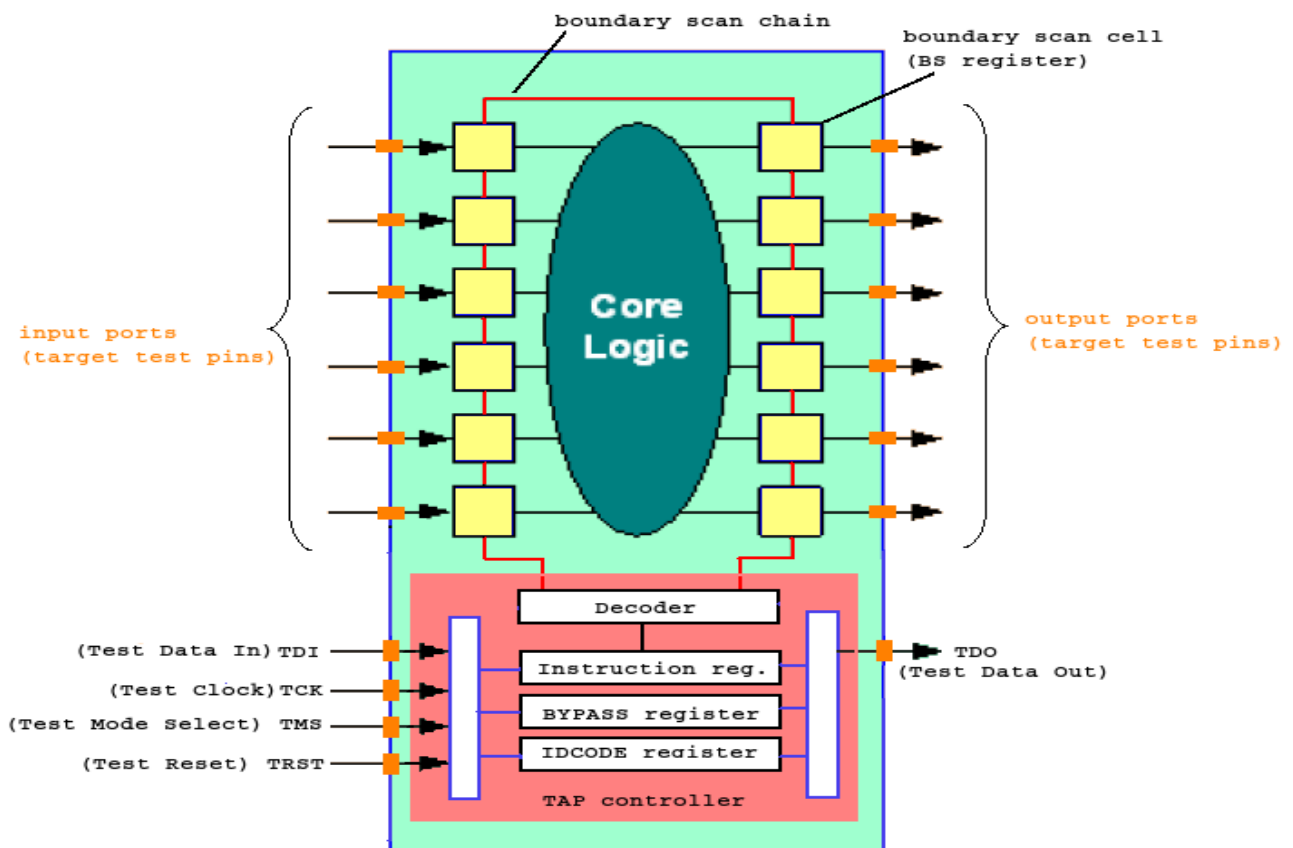


Figure 2: Overview of boundary scan circuit

Example:

The information about IDCODE register, Instruction register and Boundary scan chain is as follow:

// Information about Instruction register

attribute **INSTRUCTION_LENGTH** of TOP:entity is 8; ← the number of bit

attribute **INSTRUCTION_OPCODE** of TOP:entity is

```
"BYPASS (11111111),"&      ← code of bypass test mode
"EXTTEST (00000000),"&      ← code of external test mode
"SAMPLE (00000001),"&      ← code of sample test mode
"IDCODE (00000011),"&      ← code of ICODE test mode
"EXMBIST (00000010),"&      ← code for testing EXMBIST
"    11100000,"&
"    11100001,"&
"    11100010,"&
"    11000000,"&
"    11000001,"&
"    11010000,"&
"    11010001,"&
"    11010010);
```

// Information about IDCODE register

attribute **IDCODE_REGISTER** of TOP:entity is

```
"0000"&      ← 4bit : version code (the version of ID code)
"1000000010011111"&      ← 16bit : part code (code inheritance to product)
"01000100011"&      ← 11bit: manufacturer code (show an LSI vendor)
"1";          ← 1bit: This is a fixed code of IEEE 1149.1
```

// Information about boundary scan chain

attribute **BOUNDARY_LENGTH** of TOP:entity is **529**; ← the number of BS register in chain

attribute **BOUNDARY_REGISTER** of TOP:entity is

...

1.3 Introduction about boundary scan cell

1.3.1 Structure of boundary scan cell – general model

Inputs:

- Data In (pi = parallel in): data from input ports or user's logic

- UpdateDR (upd = update): control signal in update mode from TAP controller
- Scan In (si = serial in): test data from other BS cell or from TDI port
- ShiftDR (sdr = shift direct): control signal in shift mode from TAP controller
- ClockDR (ci = clock): a derivative of TCK clock
- Mode (md = mode): boundary scan mode

Inside:

There are 2 memory elements (1GFF and 1LATCH) with front-end and back-end MUX multiplexing data.

- GFF is capture scan cell
- LATCH is update/hold cell

Outputs:

- Data Out (po): data to user's logic or output ports
- Scan Out (so): test data to other BS cell or TDO port

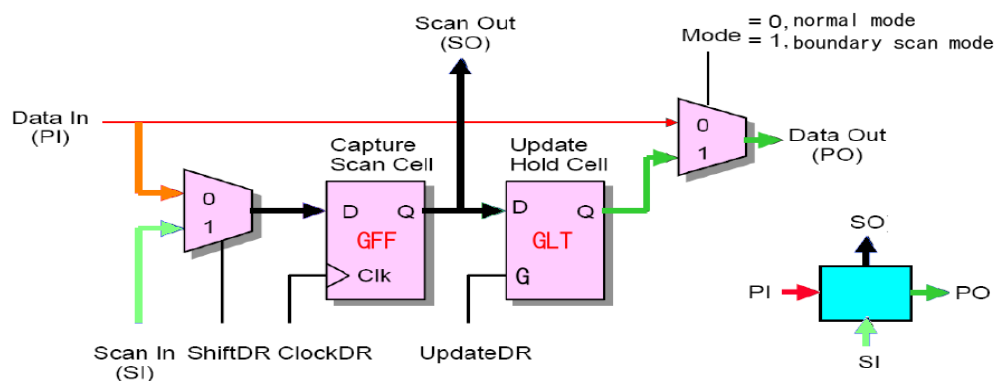


Figure 3: Structure of a boundary scan register – general model

1.3.2 Operation of BS register general model

Boundary scan cell works in 3 modes:

- In normal mode: data is passed straight through from data input PI to data output PO

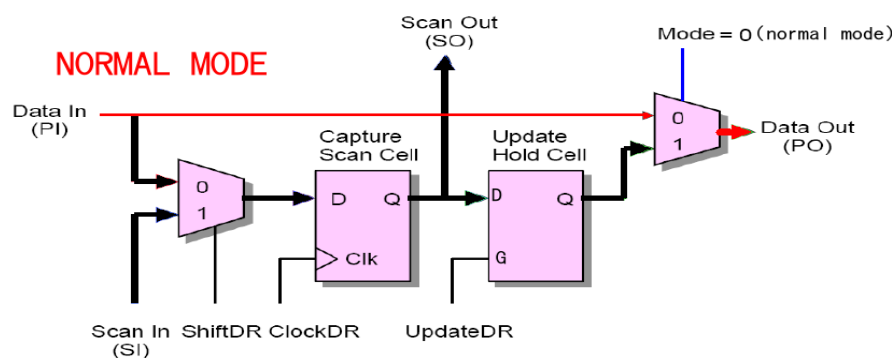


Figure 4: Boundary scan register in normal mode

- In parallel mode, there are 2 operations: capture operation and update operation.

During the capture operation, signal values on device input pins (PI) are loaded into input boundary scan cells, and signal values from the core logic are loaded into output boundary scan cells

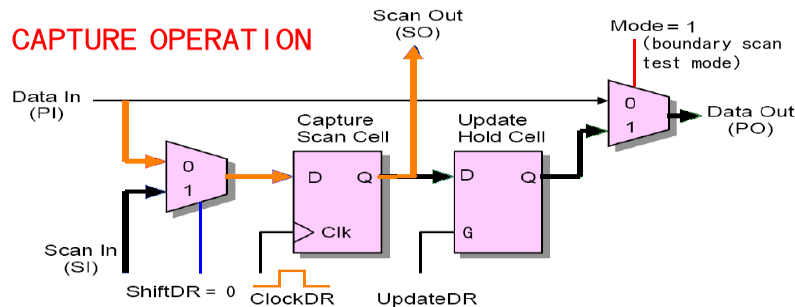


Figure 5: Boundary scan register in capture operation of parallel mode

During the update operation, data stored in Updated Hold cell is passed through Data Out (PO) pin of boundary scan cell. This operation used in case: test data from boundary scan chain is passed into combination logic (in LBIST case), or test data from boundary scan chain is exported to output port (in DCP -VOH/L test)

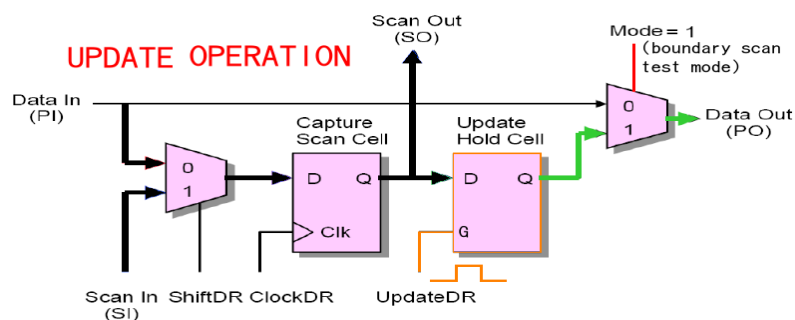


Figure 6: Boundary scan register in update operation of parallel mode

- In serial(shift) mode, data is shifted around the boundary scan chain, start from TDI (Test Data Input) through SI and SO pins of BS cells to TDO (Test Data Output)

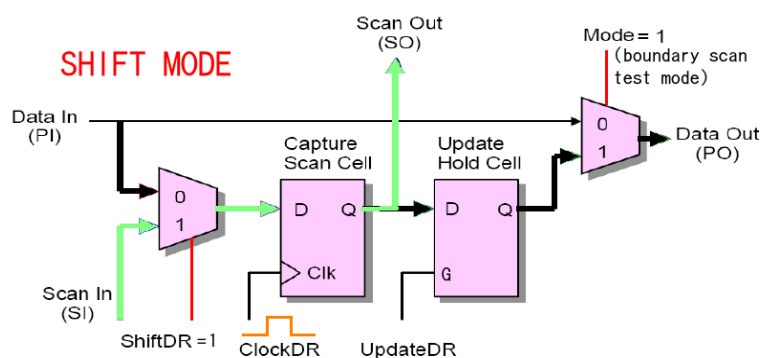


Figure 7: Boundary scan register in serial mode

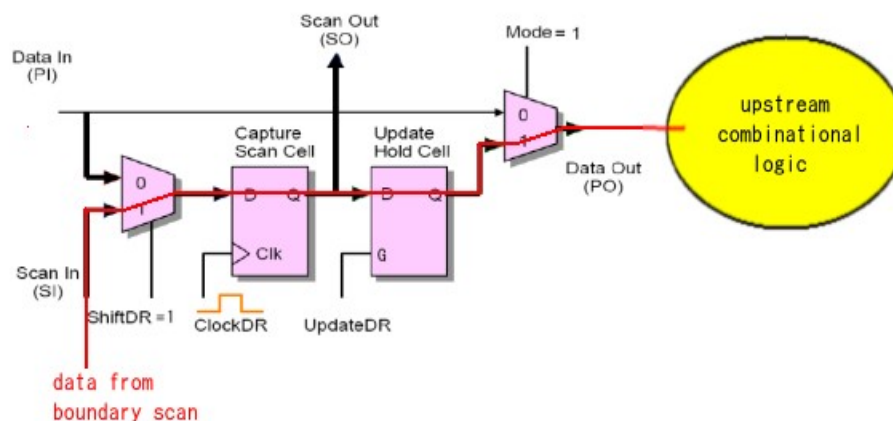
1.3.3 Types of BS register for each pin

Base on usage, BS cell is divided into 2 types:

- Insertion type (control and observe type):

This type can be used to control input data of upstream logic and observe data from downstream logic

(a) control data by BS cell



(b) observe data by BS cell

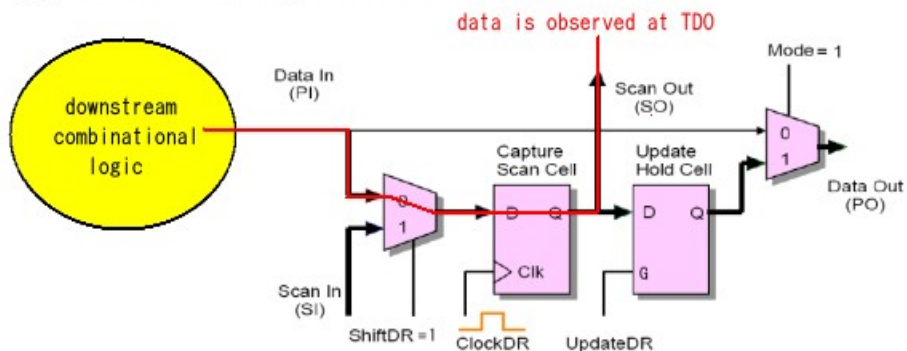


Figure 8: Boundary scan register - insertion type

BS cell Insertion type has same structure as general model. It has 2 MUXs (front-end, back-end), capture scan cell and update hold cell.

To control input data of up-stream logic, control data is input in TDI, shifted through boundary scan chain to target BS cell, then target BS cell is changed to update operation, control data is output to expected logics. (figure 8a)

To observe output data of down-stream logic, observed data is output from expected logic, input to PI pin of BS cell, target BS cell is changed to capture operation, observe data is captured into boundary scan chain and shifted to TDO port. (figure 8b)

- Observe-only type:

This type can be used to observe data from the logic before it ONLY.

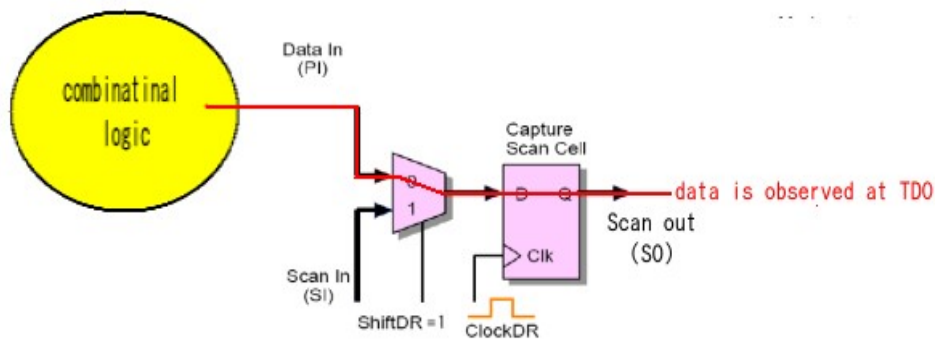


Figure 9: Boundary scan registers - observe-only type

BS cell observe-only type has **degenerated structure** as general model because of observe-only function. It has 1 front-end MUXs and capture scan cell ONLY.

To observe output data of down-stream logic, observed data is output from expected logic, input to PI pin of BS cell, target BS cell is changed to capture operation, observe data is captured into boundary scan chain and shifted to TDO port. (figure 9)

1.3.4 The position of BS register after being inserted in netlist

BS registers are inserted at IO cell terminals of test targeted pins. Based on attribute of each test targeted pins the number of BS registers inserted is different.

IO attribute: bi-directional
User function: bi-directional
BS IO attribute: B (bi-directional)

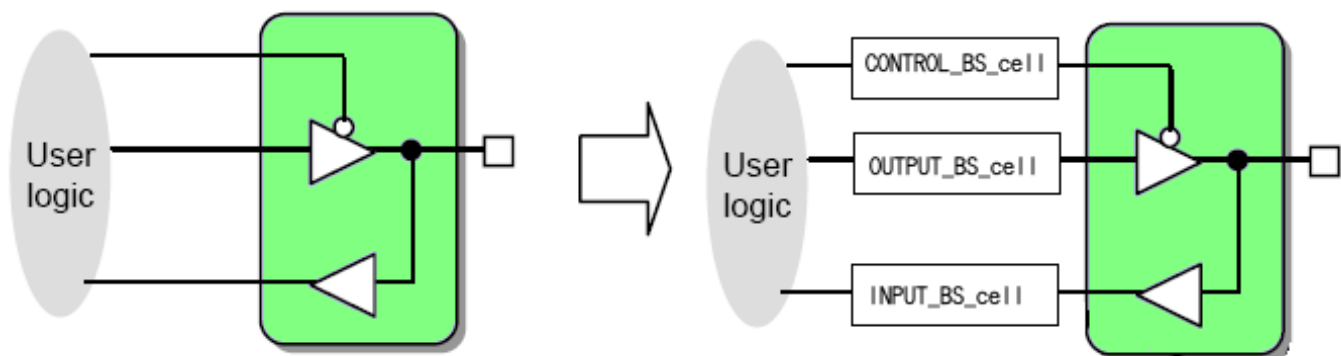
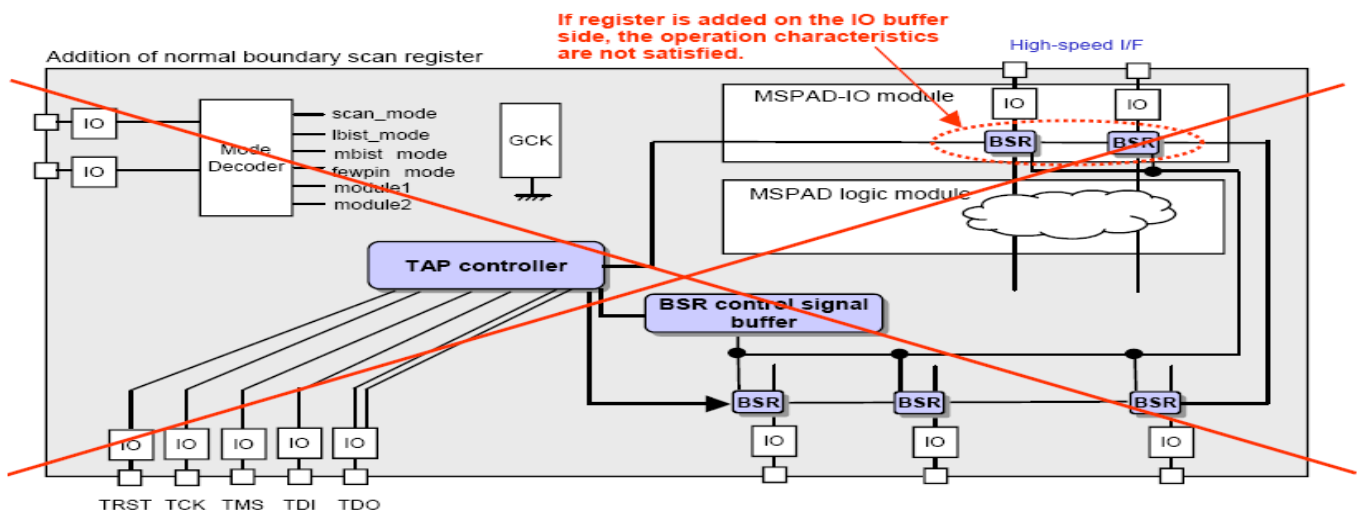


Figure 10: IO cell of bi-directional pin after inserting boundary scan

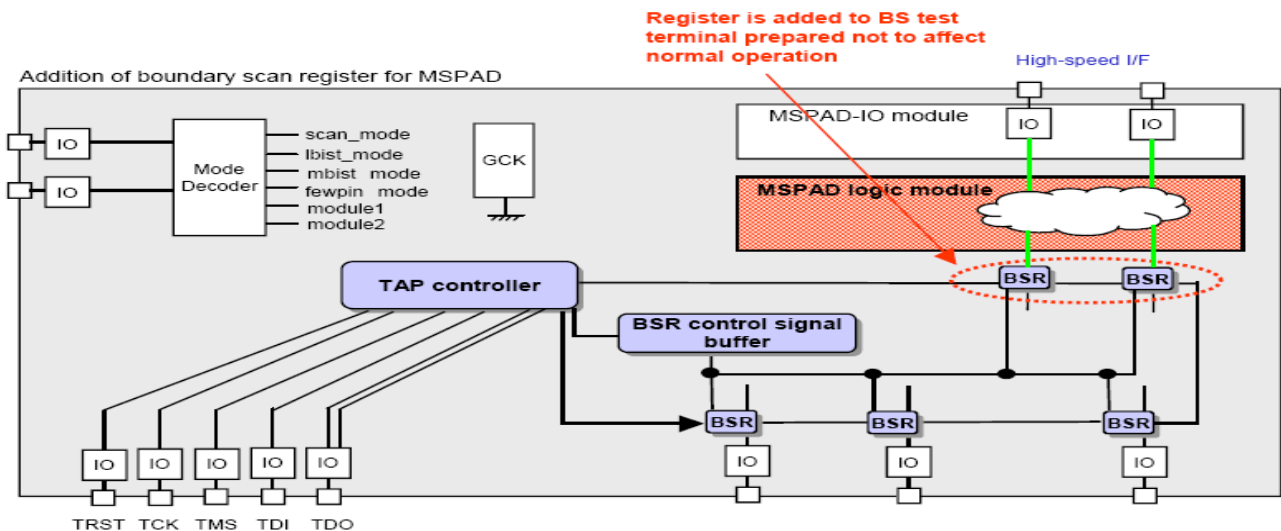
Figure 10 shows an example about IO cell terminals of a bi-directional pin before and after inserting BS register. Because attribute of this pin is bi-directional, so there are 3 BS registers inserted in input terminal (is named INPUT_BS_cell), output terminal (is named OUTPUT_BS_cell) and output enable terminal (is named CONTROL_BS_cell)

In special case, for example inserting BS register corresponding to MSPAD module, the position of BS registers are different. (MSPAD is the IP module which widely supports the interface to the SDRAM memory of DDR/SDR system, it is a high-speed interface module)

In MSPAD module, the operation characteristics including the IO buffer are optimized. If the BS register is inserted to the IO buffer terminal as other normal pins, the objective characters of the operation are not satisfied. To avoid this, the logic to be switched between the normal operation path and the BS register operation path has been embedded. By SCAAP, the BS register is added to the BS test dedicated terminal provided for the MSPAD module.



(a) adding BS register corresponding to MSPAD
(wrong execution)



(b) adding BS register corresponding to MSPAD
(correct execution)

Figure 11: An example about adding boundary scan register with MSPAD module

Figure 11a shows wrong execution when adding BS registers corresponding to MSPAD module. BS registers can not be inserted at terminals of MSPAD IO cells because the objective operation of MSPAD will not satisfied.

Figure 11b shows correct execution when adding BS registers corresponding to MSPAD module. BS registers are inserted to the BS test dedicated terminal provided for the MSPAD module. So they do not affect the operation of MSPAD and can work in boundary scan mode correctly.

NOTE:

- Pins to which BS register cannot be added
Ex: TAP, Compliance_enable_input, power supply/ground, analogue
- Some pins which shall be judged whether the BS register can be added by chip logic design Ex: High-speed I/F, clock input, clock output, reset, oscillation

1.3.5 Using BS register as MUX scan register

When non-probe pins are fixed at the reduced pin test, or when unused pins are fixed at scanning, the logics related to those pins are fixed also and they become untestable. So it makes the detection rate by scan test lower. To improve the detection rate, BS register shall be used as MUX scan register. By this method, testable data can be applied to the logics after non-probe pins or fix pins, those logics is testable and the detection rate of SCAN is increased.

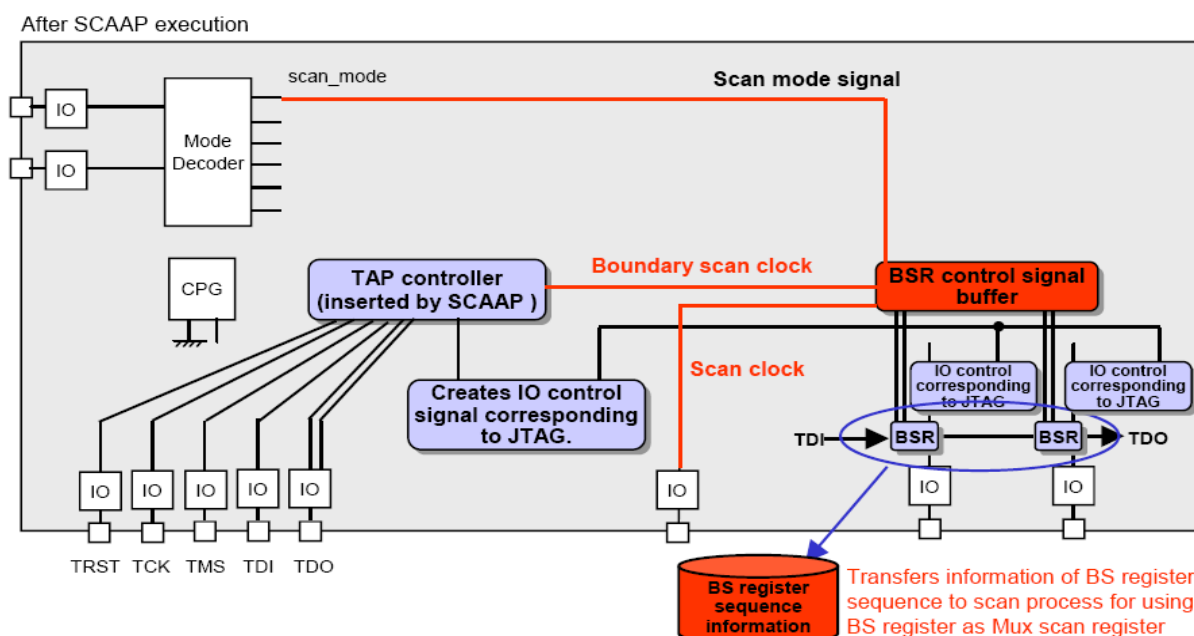


Figure 12: An example about using boundary scan circuit
when using BS cell as MUX scan register

Following figure 12 shows the circuit configuration when BS register is used as MUX scan

register. SCAAP BS register information (BS register instance name, BS register chain information, etc) will be passed to DFT SCAN step by using .bsro file (BS scan order file).

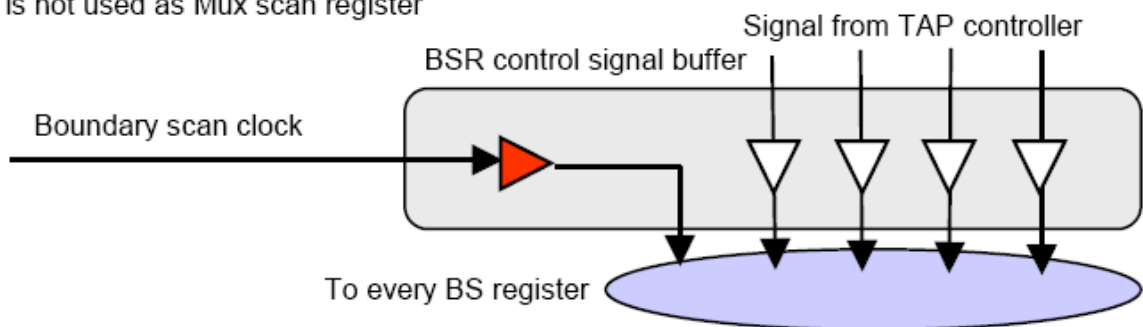
Because BS register is used as scan cell in scan test, MUST add a clock switching in clock source of BS register to switch boundary scan clock in boundary scan mode and scan clock in scan mode. This switching logic is added by SCAAP.

Figure 13 shows the BSR clock switching logic.

With method “BSR is not used as MUX scan register”, there is ONLY boundary scan clock applied to every BSR.

With method “BSR is used as MUX scan register”, there are 2 clocks applied to BSR. SCAAP inserted MUX gate to choose boundary scan clock in boundary scan mode and scan clock in scan mode. The control signal of that MUX is scan mode. The reason choose that control signal is: BSR can be used in boundary scan mode, scan test mode and user test mode (if BSR is shared for user test mode). Boundary scan mode and user test mode use the same clock source from TAP controller. So the important point is switching clock between boundary scan clock and scan clock. The scan mode is the best choice.

BS register is not used as Mux scan register



BS register is used as Mux scan register

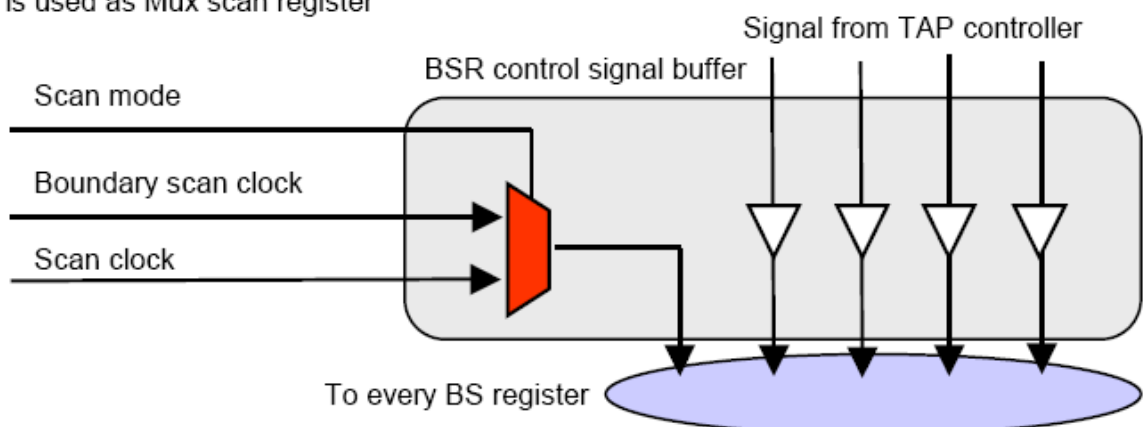


Figure 13: BSR clock switching logic

Figure 14 shows an example about structure of boundary scan cell used as MUX scan register

after SDC SCAN flow. If BS cell is used as MUX scan register, after inserting scan chains in netlist, GFF (capture scan cell) inside BR cell is replaced to an scan FF.

BS cell has 3 more pins:

SMC = scan enable : connected to scan enable port

SIN = scan test data in: connected to SO pin of previous BS cell or scan in port

SO = scan test data out: connected to SI pin of next BS cell or scan out port

In boundary scan mode, SMC is fixed to 0, capture scan cell operates as normal GFF. In scan test mode, SMC signal is controlled by outside data and boundary scan mode is fixed to 0, capture scan cell operates like other scan FF.

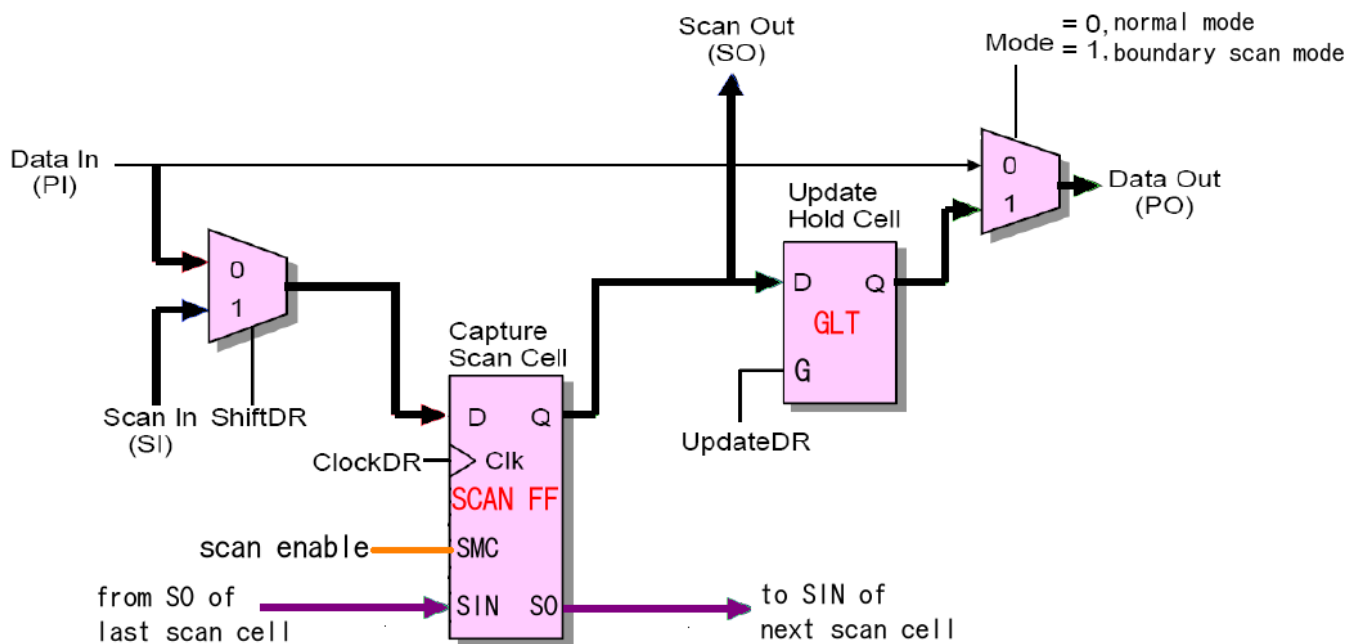


Figure 14: structure of boundary scan cell used as MUX scan register after SDC SCAN flow

1.4 Introduction about types of TAP module

Based on SCAAP tool, there are 4 types of TAP:

(1) TAP controller dedicated for boundary scan:

This is the TAP module which is used for boundary scan circuit ONLY. It means that netlist before running SCAAP has laid out IO buffer ONLY. SCAAP tool will be inserted TAP module and boundary scan registers into that netlist.

(2) Embedded TAP controller

TAP module is already inserted in netlist before running SCAAP. It means that SCAAP tool just inserts boundary scan registers into that netlist.

(3) TAP controller shared with user terminal

TAP module is shared and switched between boundary scan pins and user pins. SCAAP tool inserts the switching logic for user pins automatically.

NOTE: to avoid nonconformity to JTAG standard, the boundary scan function is not opened to the user. This can be shared only when TAP is required only for chip test in SINGEN or IOWRAP.

(4) TAP shared with debugger

TAP module is shared and switched between boundary scan pins and debugger (such as HUDI). SCAAP inserts the switching logic automatically towards the pin which is connected to TAP for debuggers

Base on the DFT test methods, there are 3 types of TAP:

- (1) Boundary scan dedicated TAP
- (2) SINGEN TAP : It means that TAP controller used in SINGEN. So TAP is inserted in SINGEN process (method of addition of boundary scan test shall be "SCAAP execution with SINGEN method")
- (3) Embedded TAP: this type of TAP corresponding to EXMBIST which is previously inserted to the netlist before running SCAAP. (method of addition of boundary scan test shall be "SCAAP execution with embedded TAP controller corresponding to EXMBIST method")

1.5 IO control logic corresponding JTAG

IO control logic is combinational logic used to conform IO buffer operation to TAP controller.

Base on JTAG standard, BS registers are inserted into input/output/output enable terminals of IO cells. So with following types of IO cells: simple input type, simple output type, 3-state output type and bidirectional type (output enable ONLY), there is no need additional control logic to conform those IO cell's operation to TAP.

But with other IO cell types (ex: bidirectional with both input and output enable), there are some IO terminals affects the logical operation of IO cells, they must be controlled to provide the operation complying with TAP controller. So there is an additional control logic is added to the uncontrolled IO terminal, it is IO control logic.

For example, figure 4 below shows the control circuit enable.

In figure15a, input enable terminal of IO cell is not controlled by TAP. So in boundary scan mode, data from input port can be passed through IO cell without any corresponding to TAP, just only based on value of input enable signal. Mismatches may occur at that IO cell during boundary scan testing.

In figure15b, IO control logic (OR gate) is added to control input enable terminal by control signal from TAP. So TAP can control the operation of that IO cell properly.

NOTE:

There are 2 types of control signals from TAP used for IO control logic:

	Normal mode	Boundary scan mode
iomask_mode_p	0	1
iomask_mode_n	1	0

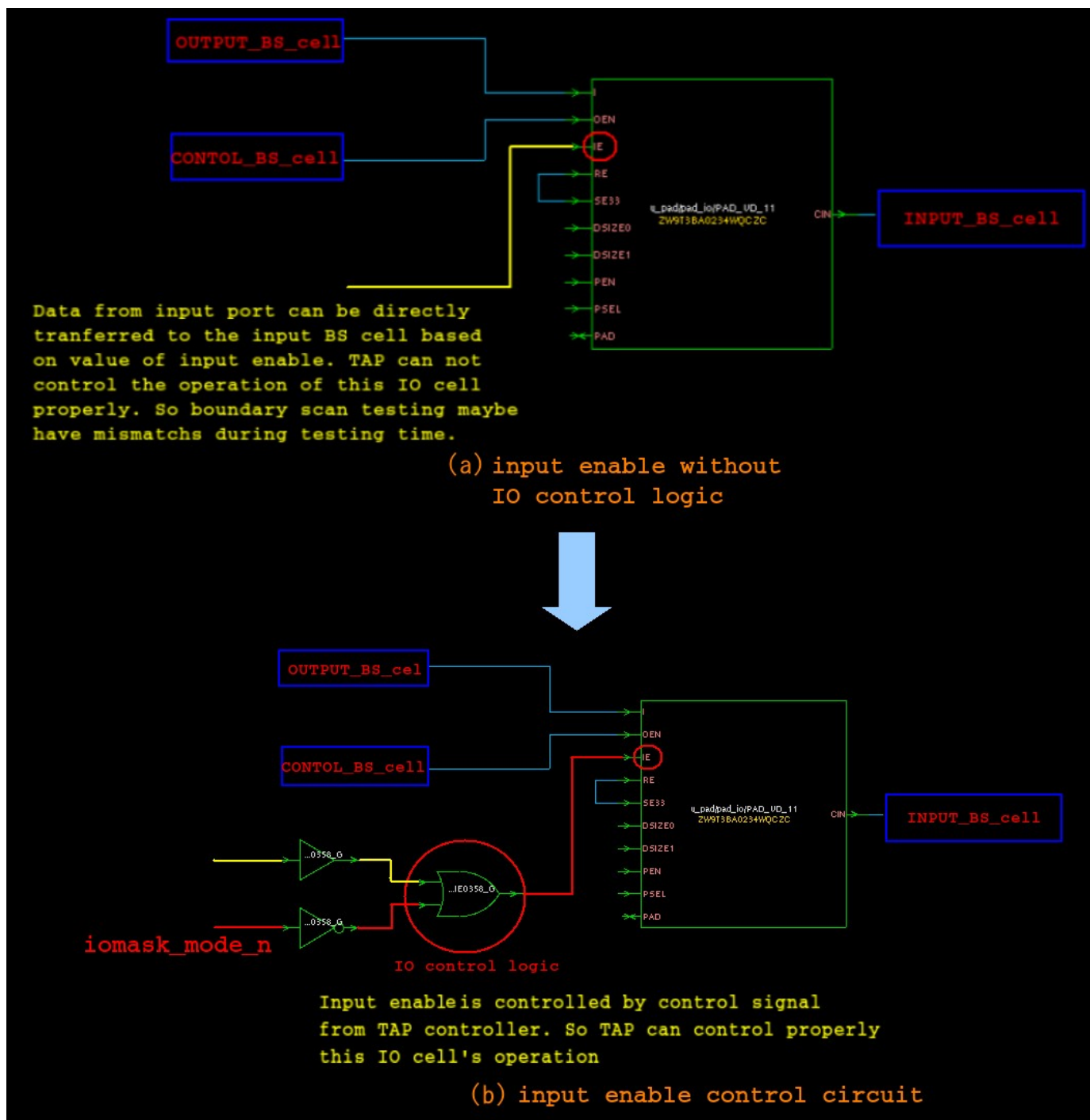


Figure 15: An example of IO control logic corresponding JTAG

2. Introduction About SCAAP Tool

SCAAP is an in-house tool.

SCAAP has 5 functions as follow:

- adding boundary scan circuit
- adding module test circuit
- adding circuit shared with DFT test pins
- adding IO state control circuits for various test
- adding other test circuits and connecting wires for others tests.

Even SCAAP has many functions but the main function is adding boundary scan circuit, other functions are just making next phases after SCAAP doing smoothly.

2.1 SCAAP function 1: Adding boundary scan circuit

2.1.1 SCAAP execution only boundary scan method

In this case, TAP controller is used only for boundary scan circuit. So SCAAP will insert boundary scan dedicated TAP controller, IO control logics, BS registers, and stitch BS registers into chain.

For more detail, refer below figure 16

In netlist before running SCAAP, there is no TAP controller or BS registers.

After running SCAAP, a dedicated TAP is inserted. Beside IO control logics, BS registers are also inserted into netlist.

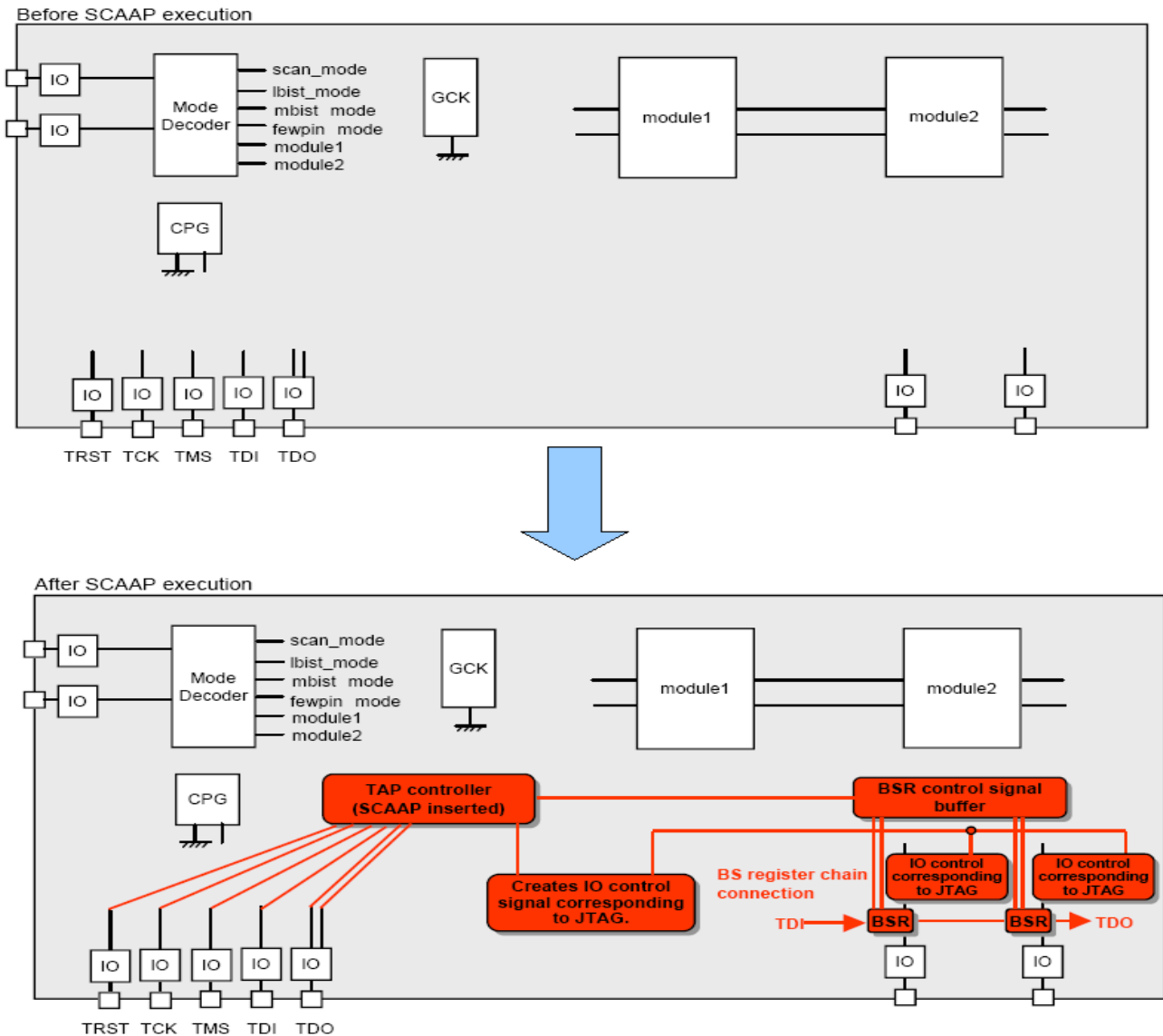


Figure 16: Configuration of boundary scan circuit by using SCAAP execution with boundary scan method only

2.1.2 SCAAP execution with SINGEN method

In this case, SINGEN TAP is used to control boundary scan circuit and LBIST circuit. In SCAAP execution, ONLY IO control logic and BS register are inserted. SINGEN TAP and boundary scan chain including the switching circuit for LBIST shall be inserted in LBIST phase.

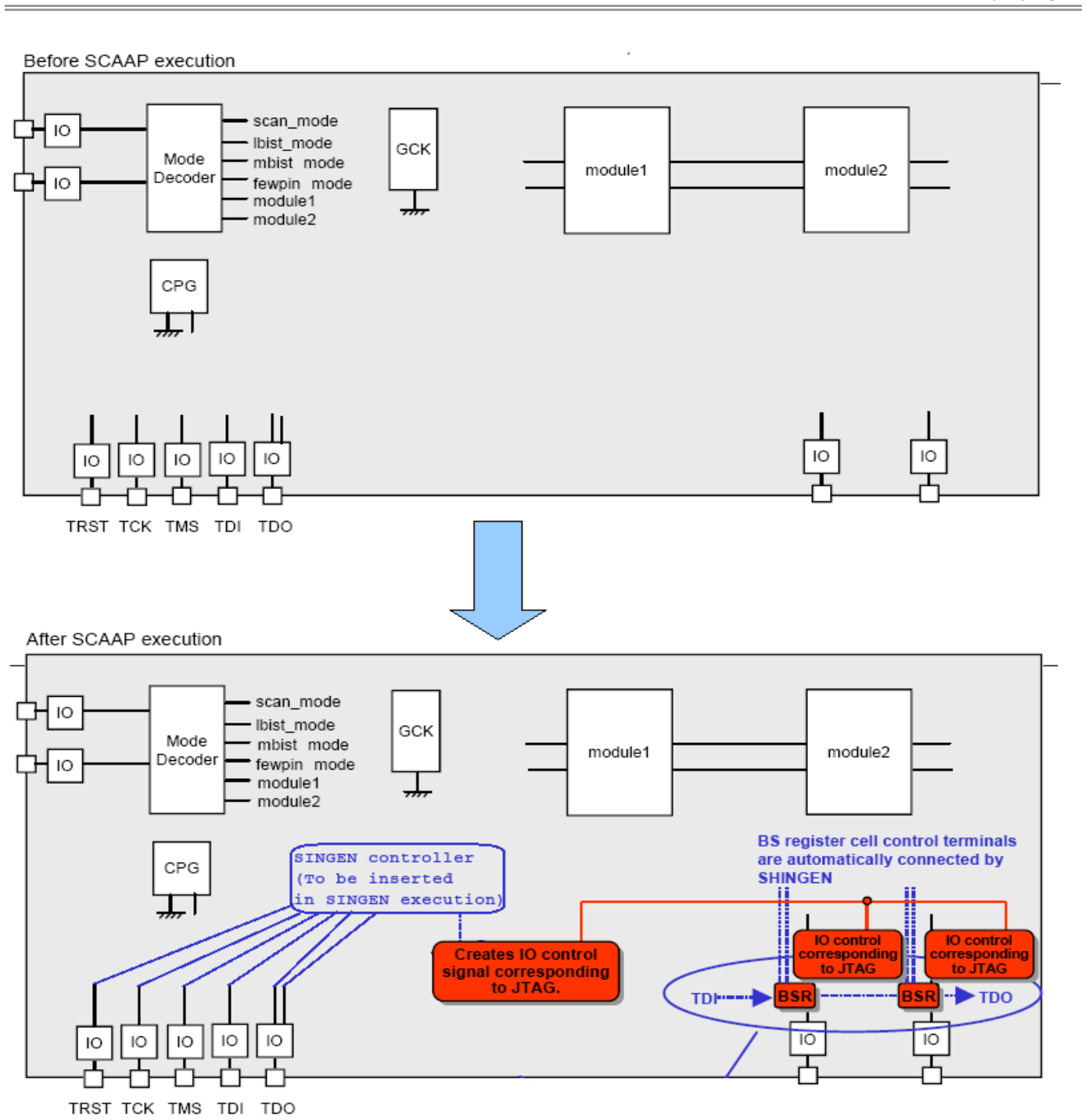


Figure 17: Configuration of boundary scan circuit by using SCAAP execution with SINGEN method

Figure 17 shows the configuration of the boundary scan circuit of the SCAAP execution with SINGEN method. By SCAAP, IO control logic corresponding to TAP controller and BS registers are inserted (red circuit). Then those information and control terminal connection information and BS register shift other information are passed to SINGEN tool by .dfp file. Based on that file, SINGEN tool will insert SINGEN TAP and perform boundary scan chain connection. Beside BS

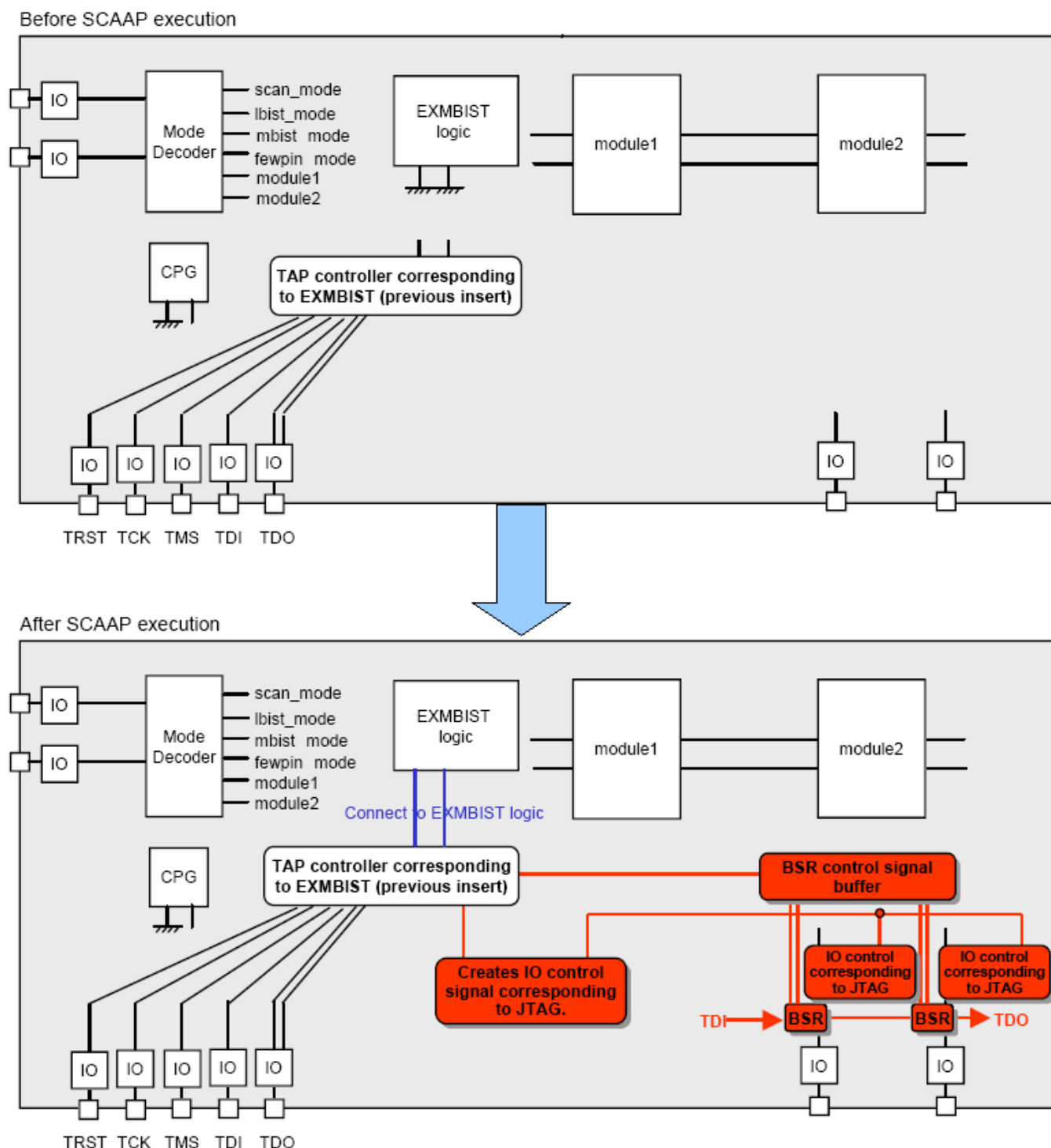


Figure 19: Configuration of boundary scan circuit by using SCAAP execution with embedded TAP controller

Figure 19 shows the configuration of boundary scan circuit by using SCAAP execution with embedded TAP controller corresponding to EXMBIST method. By SCAAP, the IO control logic and BS registers are inserted, control terminals and boundary scan chain are connected (red circuit).

2.2 SCAAP function 2: Adding module test logic

2.2.1 Introduction about module test logic

Because of the enlargement of LSI, whole chip test is getting difficult, so the solution is executing test by function module (instead of testing whole chip once time, testing each module of that chip one by one).

Figure 8 shows the configuration of the module test circuit.

- With input pins of targeted module, extraction path switching circuit is inserted
- With external ports used to input test data and get responded data from targeted module, the circuit shared with extraction is inserted at terminals of each IO cell to control the data flow for module testing. They are switched by the extraction test mode signal.

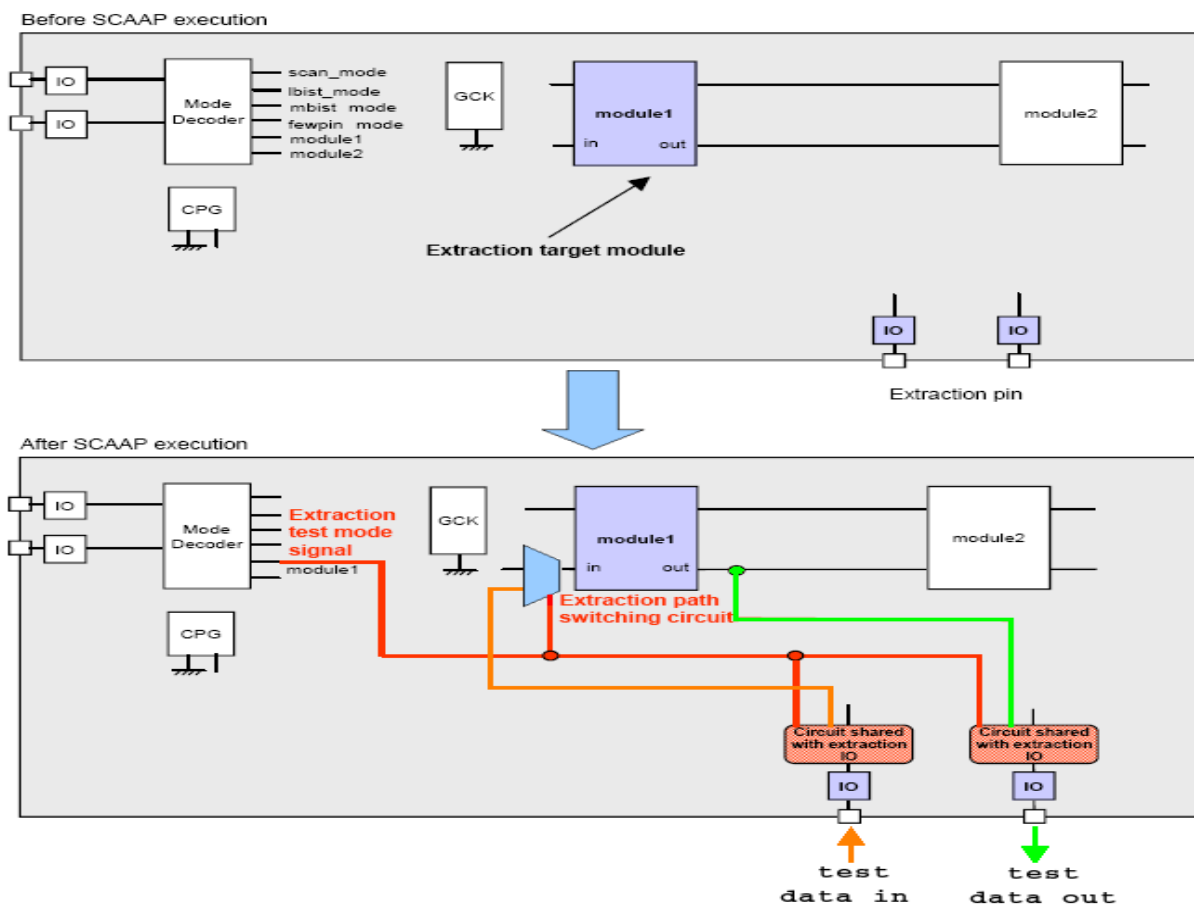


Figure 20: Configuration of module test circuit added by SCAAP

2.2.2 SCAAP execution with module test method

In this test, SCAAP adds circuit to extract the module terminal to external ports and user logic parts, ... Following figures show examples about the way SCAAP inserts logic for module test with bi-directional IO cells (input, output and output enable terminals).

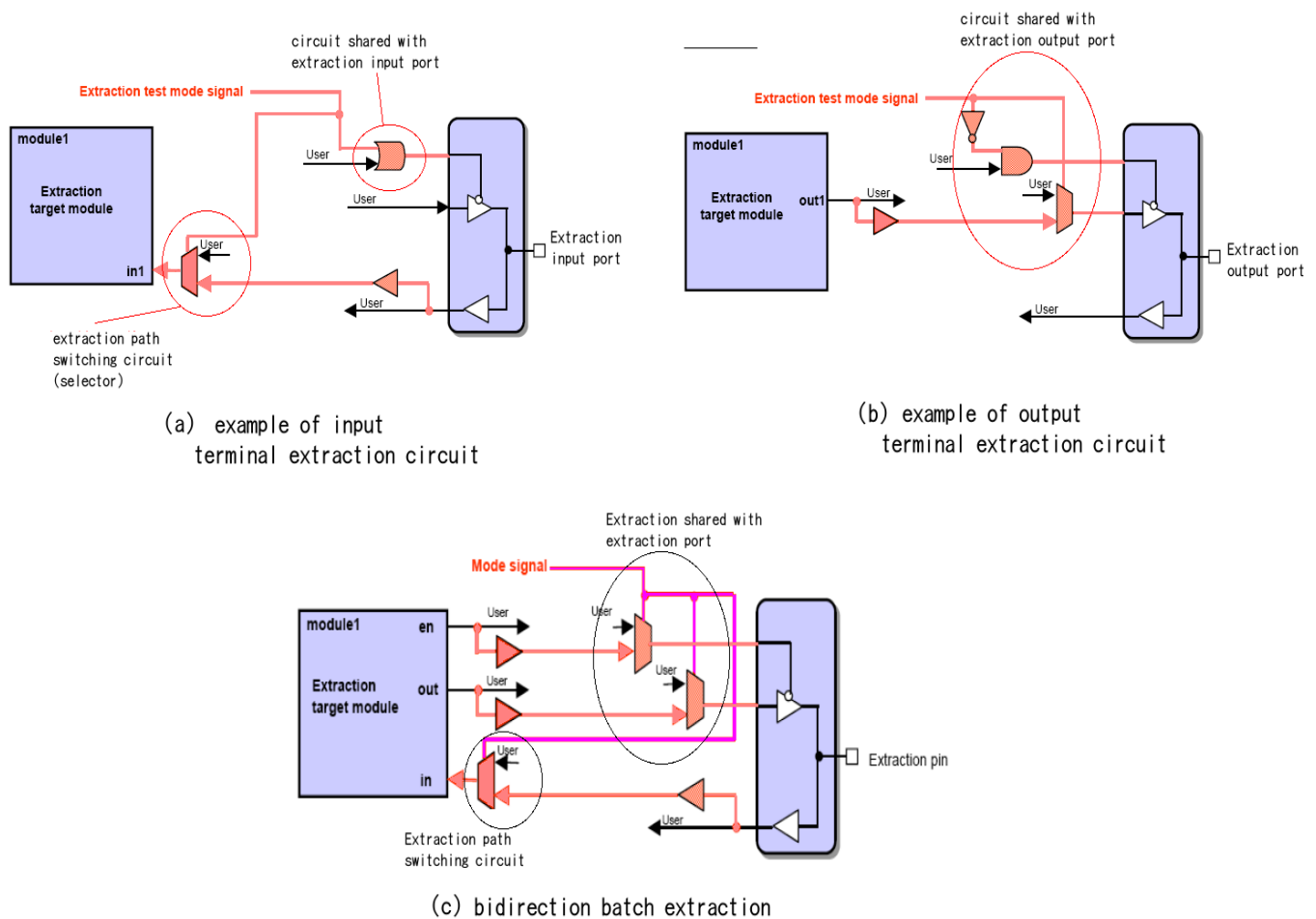


Figure 21: Examples of additional circuits by SCAAP execution with module test method

In figure 21a, bi-directional port is used as extraction input only. So output direction of its IO cell must be disabled and input direction is enabled. SCAAP inserted an OR gate at output enable to set output enable = 1 in module test mode. At input direction, a MUX is inserted to drive test data from extraction input port to input terminal of targeted module.

In figure 21b, bi-directional port is used as extraction output only. So output direction of its IO cell must be enabled and input direction is disabled. SCAAP inserted an AND gate at output enable to set output enable = 0 in module test mode. At output direction, an MUX gate is inserted to drive responded data from targeted module to extraction output port.

In figure 21c, input/output/output_enable terminal of bi-directional ports are extracted. In each terminal, SCAAP inserted a MUX gate to choose test signals in module test mode.

2.3 SCAAP function 3: Adding shared pin logics

2.3.1 Introduction about shared pin logic

There are many DFT methods are implemented into top LSI such as boundary scan, scan test, LBIST, MBIST, module test, ... But the number of external ports is limited. So DFT pins must

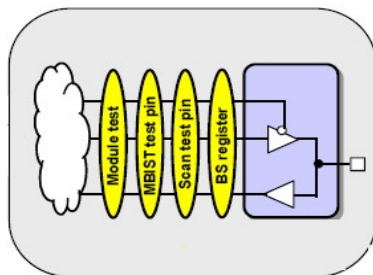
be shared with normal mode pins.

Shared pin logic will be added at IO terminals of each shared port. The structure of shared pin logic is based on how many DFT modes share that port and the priority of those DFT modes.

The priority of the control shall be considered among DFT logics according to each test specifications (combination of DFT methods).

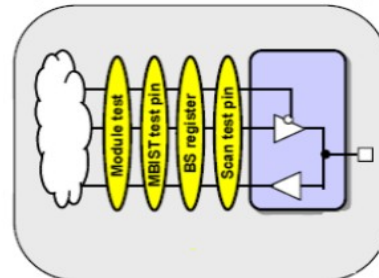
For example, the boundary scan logic is normally prioritized to take over the normal chips, but when the BS register is a target to scan, the test pins for scan test shall be operated regardless of the BS register state. The control in higher priority shall be applied to the IO buffer than the BS register. (When the BS register is not scanned, it is not necessary to be considered because it is passed during the scan operation.)

case 1: If MBIST and scan do not share one test circuit and BS register is not used as MUX register, priority: module test < MBIST < SCAN < Boundary scan



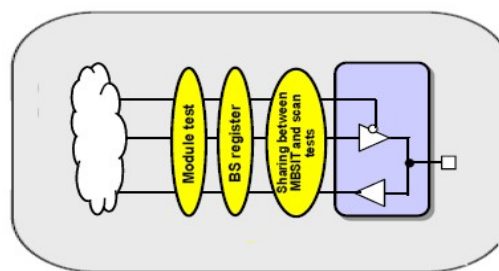
(a) priority of shared pin logic (case1)

case 2: If MBIST and scan do not share one test circuit and BS register is used as MUX register, priority: module test < MBIST < Boundary scan < SCAN



(b) priority of shared pin logic (case2)

case 3: If MBIST and scan share one test circuit and BS register is used as MUX register, priority: module test < Boundary scan < scan & MBIST share test circuit



(c) priority of shared pin logic (case3)

Figure 22: Priority of shared pin logics

Following figures show examples about the priority of DFT modes commonly meet.

Figure 22a shows case1, in this case, MBIST and scan test DO NOT share one test circuit and BS register is NOT used as scan cell. The highest priority is inserted shared logic for BS registers, then scan test pin, MBIST test pin, the lowest is circuit shared with extraction IO of module test.

Figure 22b shows case2, in this case, MBIST and scan test DO NOT share one test circuit and BS register is used as scan cell. The highest priority is inserted shared logic for scan test pin, then BS registers, MBIST test pin, the lowest is circuit shared with extraction IO of module test.

Figure 22c shows case3, in this case, MBIST and scan test share one test circuit and BS register is used as scan cell. The highest priority is inserted shared logic for scan and MBIST test pin, then scan test pin, the lowest is circuit shared with extraction IO of module test.

2.3.2 SCAAP execution with shared pin logics

By SCAAP, the circuits shared with test pin for various DFT can be collectively inserted in consideration of the priority of their control. (Control circuit placed closer to the IO side: Higher priority). From the inserted circuit shared with test pin, the connection data (DPF file, MC file, etc.) is passed to each DFT addition process, so that the test signals can be connected in the DFT addition process.

DPF file is the connection indication file to SHINGEN.

MC file is the connection indication file to MINORI.

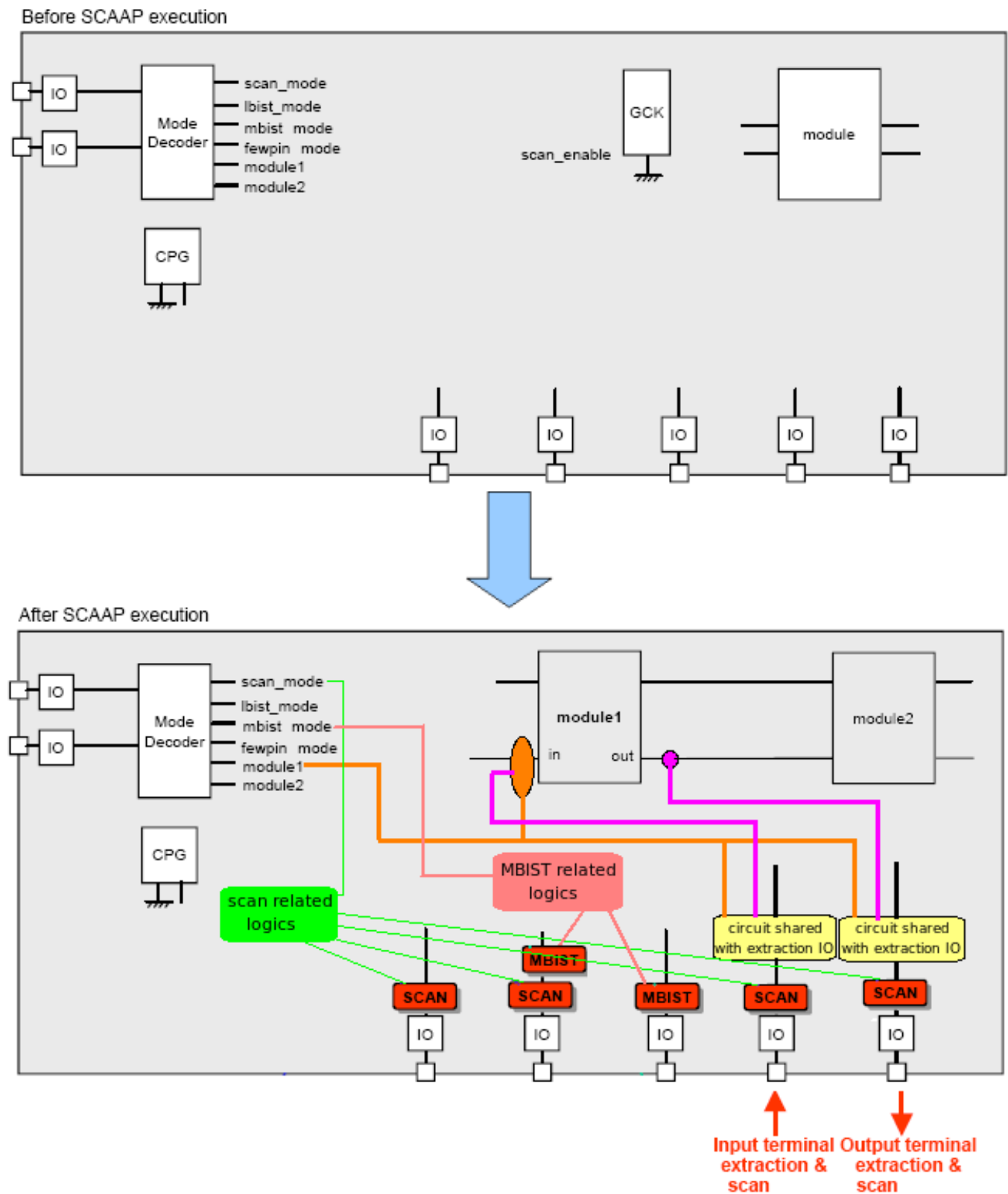


Figure 23: Addition of circuit shared with test pin

Figure 23 shows an example SCAAP inserted shared pin logics for 3 modes (scan , MBIST, module test). The priority is SCAN > MBIST > module test.

NOTE:

When there are destinations of test signal connection before SCAAP execution, addition of circuit shared with test pin and connection of test signal to inside logic can be simultaneously executed.

There is a big difference between extraction by module test and test signal extraction by function shared with test pin.

- In module test, the circuit extracts 2 or more signals to one pin can be added.
- In the sharing one pin for 2 or more kinds of test, an independent shared circuit is added to each kind of test.

==> two or more control circuits (selectors ...) are inserted in series. The module test circuit shall be normally lower-priority control than other DFT test pins.

2.4 SCAAP function 4: Adding IO state control logic for various test

2.4.1 Introduction about IO state control logic

Because there are many DFT modes in LSI. But in each mode, there are a few ports are used as control signals or test data pins. If unused pins are not controlled, there maybe power consumption or unexpected value propagated to test logic cause mismatch points during testing.

So the unused pins must be controlled in each specific test mode. To do this, IO state control logic is added IO cell terminals of unused pins to control them in specific test mode.

The types of IO state control logic are following:

- fixing the direction of non-probe pin at reduced pin test
- fixing the direction of unused user pin at scan test
- fixing the direction of unused pin at MBIST
- forcibly turning on/off of PULL up/down enable

Following figures show example about fixing IO state of a bi-directional port with output enable.

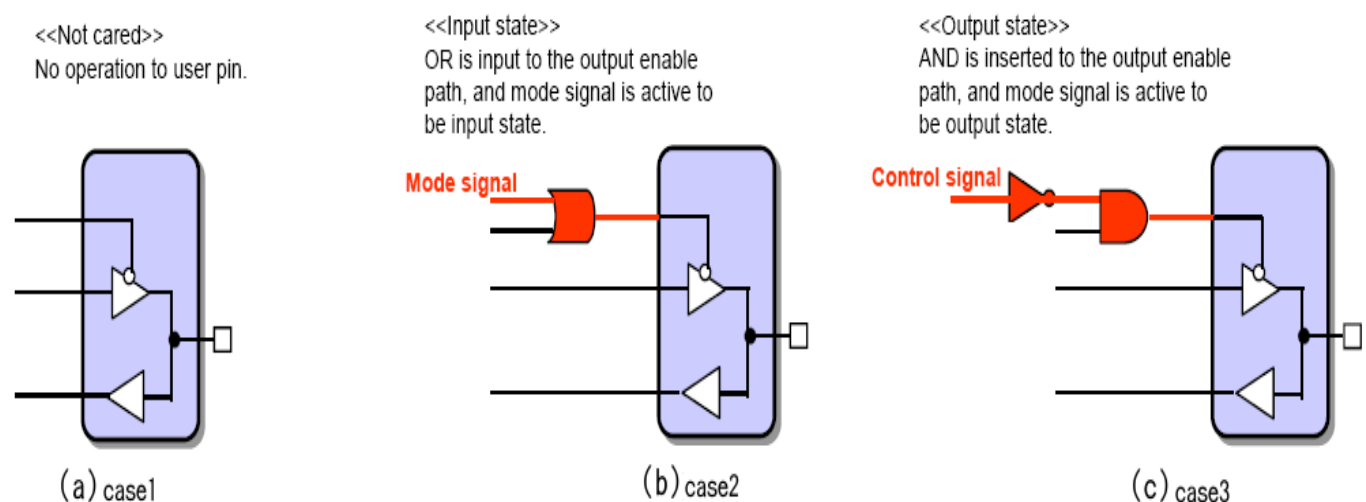


Figure 24: Structure of IO state control logic

Figure 24a shows case1, if user pins are not related to any operation of specific DFT mode, NO NEED to insert IO state logic at their IO cells. They can be ignored.

Figure 24b shows case2, if user pin is used as input test pin in specific DFT mode, an OR gate is inserted to active to be input state that DFT mode for reducing power consumption.

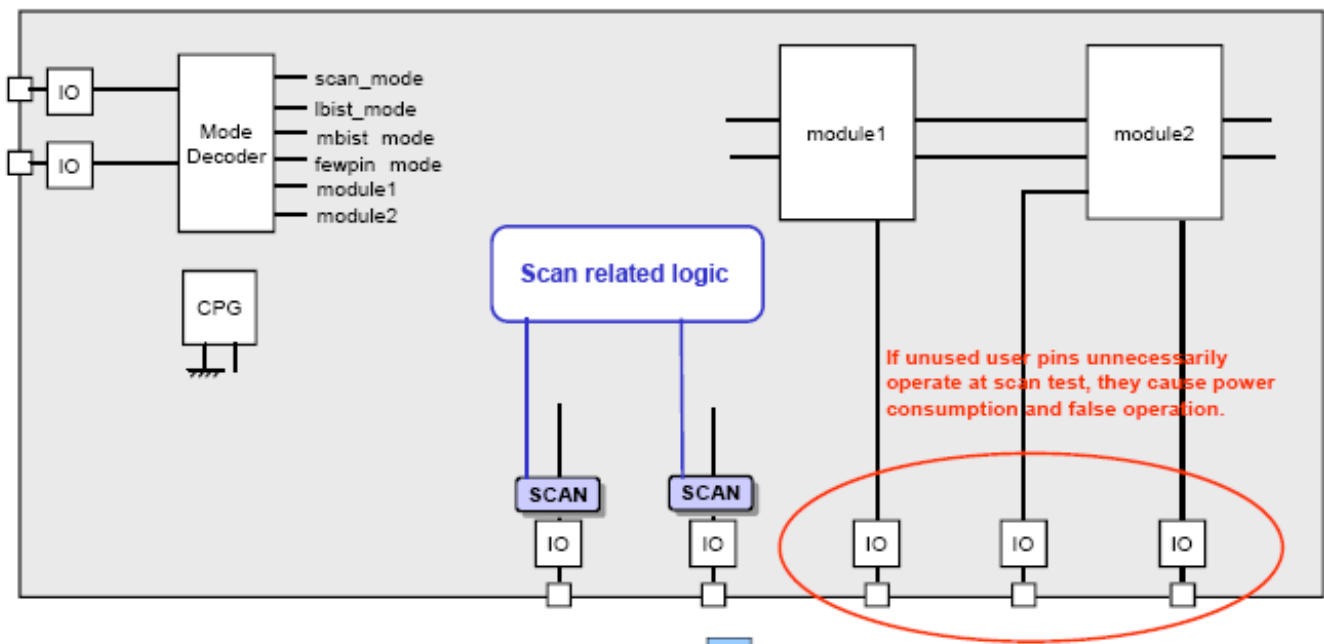
Figure 24c shows case3, if user pin is used as output test pin in specific DFT mode, an AND gate is inserted to active to be output state for preventing unexpected data from outside propagated into test logic during testing.

2.4.2 SCAAP execution with IO state control logics

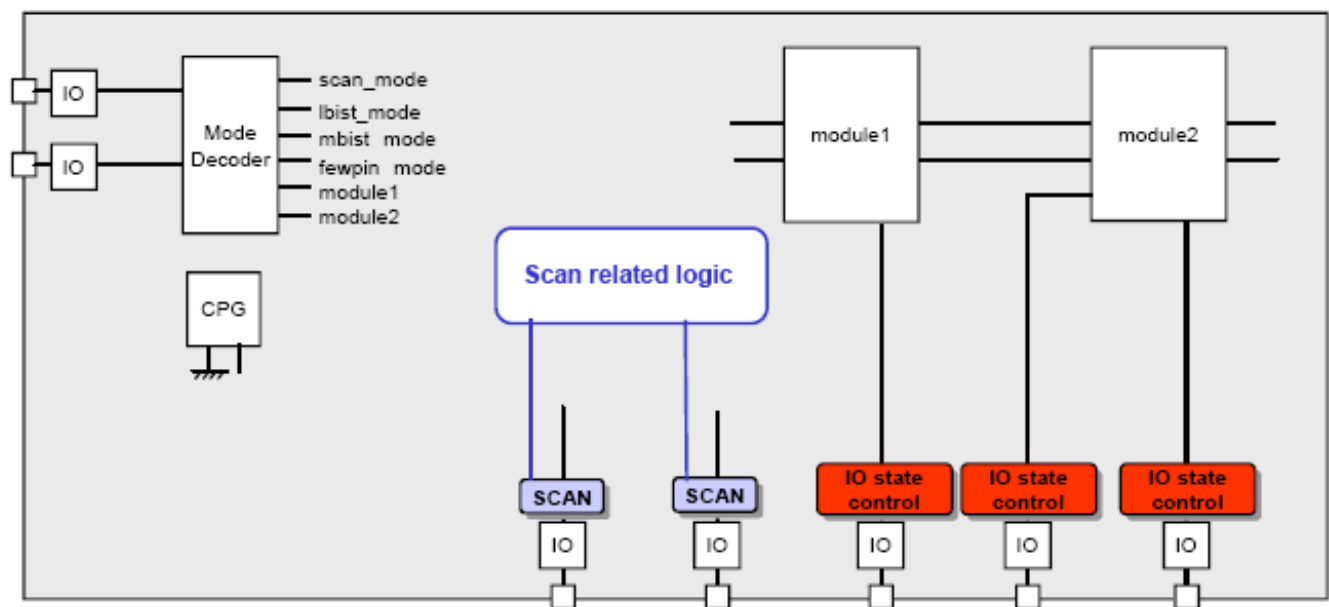
When the state of specific pins must be controlled in a specific test mode, the circuit to control the IO buffer state can be added in SCAAP.

The following figure shows an example of IO state control in scan mode. IO state circuit is added so that user pins which are not used as scan pin DO NOT operate unnecessarily.

After test circuit is added



IO state control



Add the logic to fix the IO direction for unused pins at the scan mode.

Figure 25: Adding IO state control logics by SCAAP

2.5 SCAAP function 5: Adding various-purpose test logics

In some cases, various circuits need to be added to perform testing as well as test logic normally added by all kinds of DFT tools, such as adding connection between test logics or fixing

module input terminal to a specific value.

SCAAP can:

- adding net to instance port, connection of net between hierarchies
- adding gate to a specified module
- inserting gate to a module terminal.

NOTE:

Circuits (gates) to be added are limited to the stylized circuits registered to the dedicated library (SCL). In default setting, buffer, inverter, basic gate (AND, OR), input terminal L/H fixed circuit, and signal selection (selector) circuit are registered. Circuits which can be added depends on the library version to use.

Following figure shows an example of various-purpose test logic addition. SCAAP inserted fixed circuit to module1.pinZ , connected module2.pinA to module3.pinB, inserted buffer to an IO cell terminal.

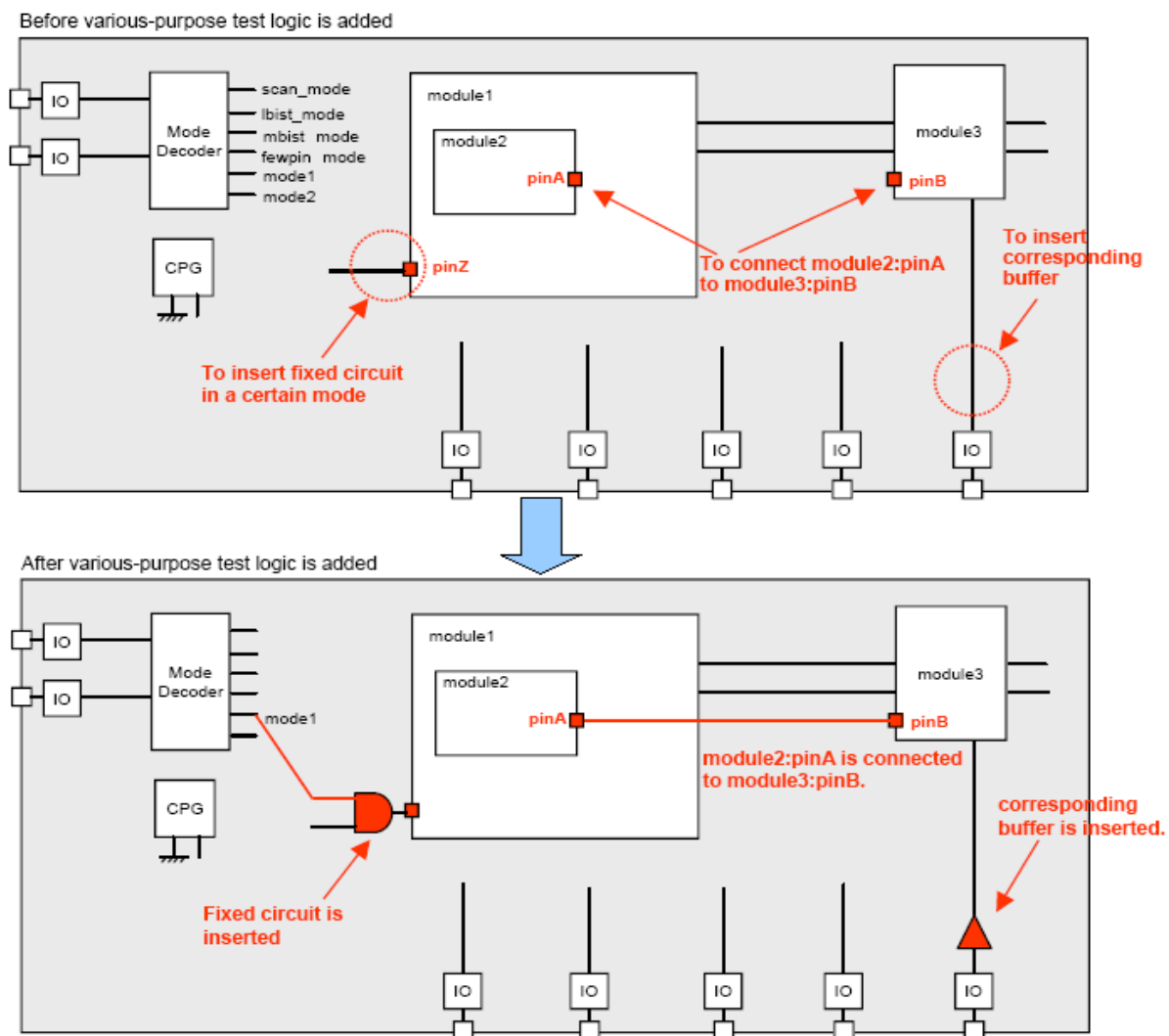


Figure 26: Example about various purpose test logic addition by SCAAP

II. SCAAP EXECUTION

Part “SCAAP execution” describes the total flow to insert boundary scan circuit by using SCAAP:

- Understanding and preparing the input files for running SCAAP
- Making run file and building environment for SCAAP
- Understanding and analysing the output files

From the top view, in/out flow of SCAAP is as below:

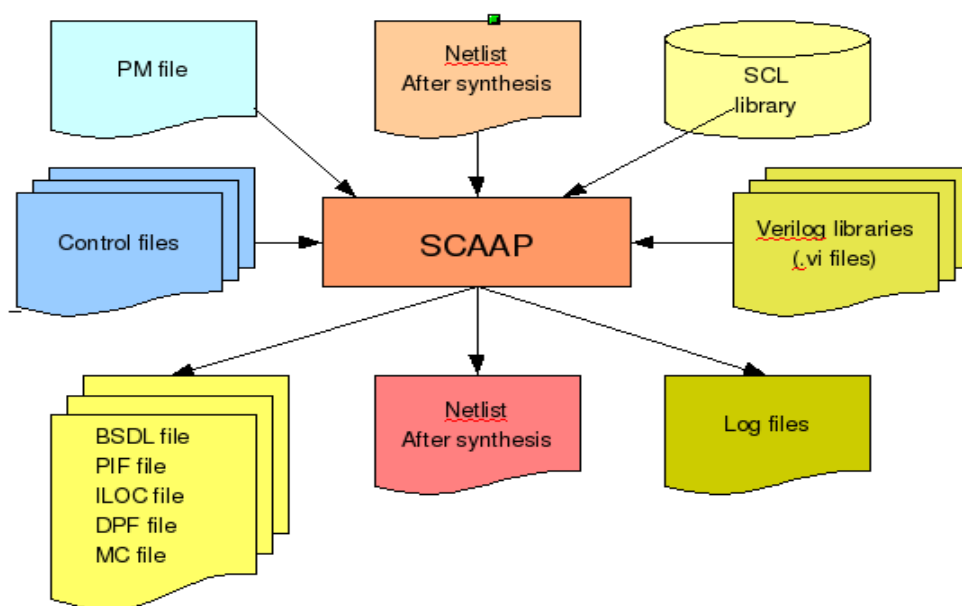


Figure 27: Inputs/Outputs of SCAAP execution

To have detail information contained in each files, please refer following parts.

1. INPUTS

1.1 Verilog library file

This file contains verilog library paths. It is used to read netlist in SCAAP before running.

The syntax is:

-f <file_name> : directs SCAAP to execute with the command options written in <file_name>.

That file contain other verilog commands.

-v <file_name> : to specify a library file

-y <directory_name> : to specify a library directory

+libext+<extension1>+... : to specify extensions of library file names which will be read in library directory. This option is used with option -y

For example:

```
-y ../../Lib/Verilog/RC03ATB7
```

```
-v ../Lib/Verilog/Product_Special/HM.vi
+libext+.v
+libext+.vm
```

NOTE:

Reading .vi files when only terminal definition for cell is required in SCAAP. This action will reduce the run time much especial with big netlist.

1.2 SCL libraries

SCL library is provided from System Setting and Development section. This kind of library has information to support SCAAP tool running. (refer document SCAAP User's manual SCL library creation procedure to have detail format of each file in SCL library)

Following figure shows the basic structure of SCL library

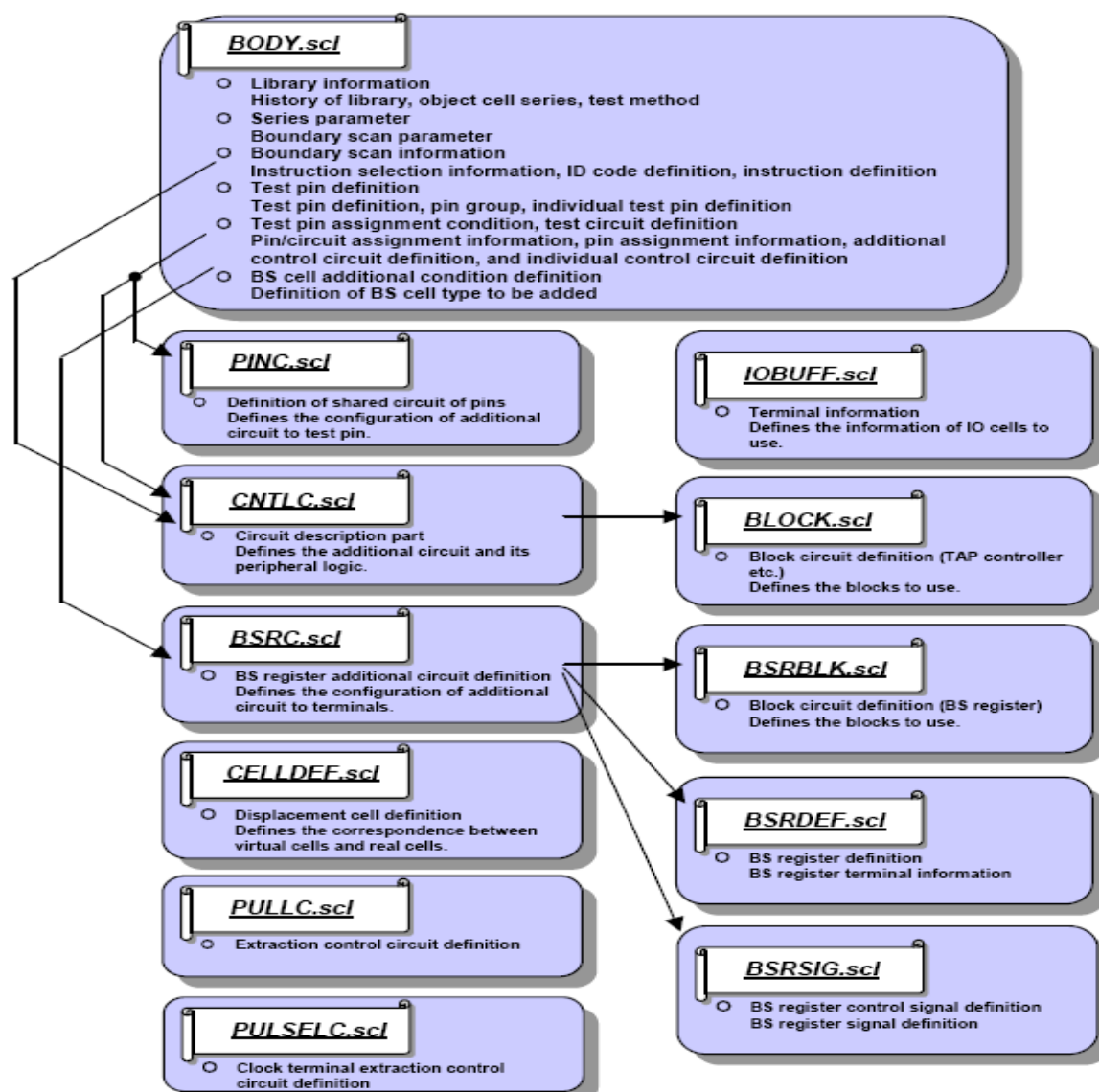


Figure 28: Structure of an SCL library

The number of library files and file name maybe different based on library type. But there MUST be at least 10 files:

- BODY.scl
- IOBUFF.scl
- PINC.scl
- CNTLC.scl
- BLOCK.scl
- BSRC.scl
- BSRBLK.scl
- BSRDEF.scl
- BSRSIG.scl
- CELLDEFF.scl

1.3 PM file (Pin multi table)

PM file is CSV format file which includes:

- Pin number, pin name and pin attribute of each LSI pin
- Boundary scan addition information, reduced test pin probe information
- Extracting information of shared pin for test and user logic
- Module terminal extracting information
- IO status control information as a circuit addition instruction

NOTE:

- Columns where the heading cell does not contain any defined word (explained in the following pages) are treated as comment columns.

#KEY	NOTES	#XXX	#NO

The whole columns are treated as comments.

- Cell definition format

The contents of cells can be separated by the following delimiters.

- 1st delimiter “/” : separates the settings for each of the pin functions to be multiplexed
- 2nd delimiter “+” : specifies multiple entries in one cell
- 3rd delimiter “:” : specifies a pair or a set of information
- Using “-” : when no data is to be specified

1.3.1 Valid data table area definition (key info.)

#KEY column : defines the valid data area of the pin multiplexing table

Example:

#KEY	#NO	#PKG	#IL	#LOC	#PIN	#CELL	#PINSPEC	#ATTR	#BSCAN	#PROBE	#BSATTR
@											
#BEGIN											
#DATA1											
#DATA2											
	1	1	259	I1	CXXIN	ZW9C3X12ZC		I	N	Y	
	2	2	258	I1	CXXOUT	ZW9C3X12ZC		O	N	Y	
	3	3	151	I1	CXMTCLKOUT	ZW9T3BA0234WQCPC	3S	B	Y	Y	B
	4	4	266	I1	CXRST_N	ZW9T3D21YWC		I	N	Y	I
	5	5	265	I1	CXPOWREN	ZW9T3D21YWC		I	N	Y	I
	6	6	101	I1	CXRSTOUT_N	ZW9T3BA0234WQCPC	3S	B	Y	Y	B
#END											

#BEGIN : declares the beginning of valid data

#END : declares the end of valid data

@ : makes a row as a comment

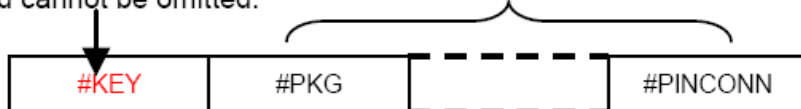
→ the row after #BEGIN and before #END will be defined as valid data.

NOTE:

In a PM file, column with heading #KEY must be in fixed position and can not be omitted, other columns can be changed the position or omitted.

Column position is fixed,
and cannot be omitted.

Column position can be changed, and the column can be omitted.



Heading positions determine column positions.

1.3.2 Pin information

#NO column : external pin control number

It is a serial number for reference. (Abbreviation is allowed)

Example:

#KEY	#NO	#PKG	#IL	#LOC	#PIN	#CELL	#PINSPEC	#ATTR	#BSCAN	#PROBE	#BSATTR
@											
#BEGIN											
#DATA1											
#DATA2											
	1	1	259	I1	CXXIN	ZW9C3X12ZC		I	N	Y	
	2	2	258	I1	CXXOUT	ZW9C3X12ZC		O	N	Y	
	3	3	151	I1	CXMTCLKOUT	ZW9T3BA0234WQCPC	3S	B	Y	Y	B
	4	4	266	I1	CXRST_N	ZW9T3D21YWC		I	N	Y	I
	5	5	265	I1	CXPOWREN	ZW9T3D21YWC		I	N	Y	I
	6	6	101	I1	CXRSTOUT_N	ZW9T3BA0234WQCPC	3S	B	Y	Y	B
#END											

#PKG column : package pin/ball number

It is output to BSDL file only. It is a not real number and does not affect circuit addition. However, this number is necessary information for BSDL for customer. To create the BSDL file for customer, all package pins need to be written, including power/NC pins. Pins which does not need to be written if it is for a package.

It is an arbitrary string but required, blank is not allowed.

Example:

#KEY	#NO	#PKG	#IL	#LOC	#PIN	#CELL	#PINSPEC	#ATTR	#BSCAN	#PROBE	#BSATTR
@											
#BEGIN											
#DATA1											
#DATA2											
	1	1	259	I1	CXXIN	ZW9C3X12ZC		I	N	Y	
	2	2	258	I1	CXXOUT	ZW9C3X12ZC		O	N	Y	
	3	3	151	I1	CXMTCLKOUT	ZW9T3BA0234WQCPC	3S	B	Y	Y	B
	4	4	266	I1	CXRST_N	ZW9T3D21YWC		I	N	Y	I
	5	5	265	I1	CXPOWREN	ZW9T3D21YWC		I	N	Y	I
	6	6	101	I1	CXRSTOUT_N	ZW9T3BA0234WQCPC	3S	B	Y	Y	B
#END											

#IL column : inner lead (pad) number

It is used as a base for BS register shift order.

As normal, pin has the position closed to a TDI pin position becomes boundary scan chain starting point and #IL is used as a relevant order for reference.

So when #IL is changed, order of boundary scan chain will be changed. This is very important point because that order will affect layout design and BS test patterns.

#IL is required, blank is not allowed. If a pin without PAD connection (NC/ring power), enter a DUMMY number (ex: a serial number after the maximum PAD number)

Example:

#KEY	#NO	#PKG	#IL	#LOC	#PIN	#CELL	#PINSPEC	#ATTR	#BSCAN	#PROBE	#BSATTR
@											
#BEGIN											
#DATA1											
#DATA2											
	1	1	259	I1	CXXIN	ZW9C3X12ZC		I	N	Y	
	2	2	258	I1	CXXOUT	ZW9C3X12ZC		O	N	Y	
	3	3	151	I1	CXMTCLKOUT	ZW9T3BA0234WQCPC	3S	B	Y	Y	B
	4	4	266	I1	CXRST_N	ZW9T3D21YWC		I	N	Y	I
	5	5	265	I1	CXPOWREN	ZW9T3D21YWC		I	N	Y	I
	6	6	101	I1	CXRSTOUT_N	ZW9T3BA0234WQCPC	3S	B	Y	Y	B
#END											

#LOC column : location information

It contains information on location in the chip, for reference in the conversion to a hierarchical netlist. This information is not used in SCAAP single execution, it is only necessary for PLASMA execution in PAD design task, because in SCAAP single execution, IO cell and other IO peripheral circuit of BS registers are inserted into the same hierarchical level regardless of #LOC.

A location name = an initial character + (any character after the initial letter).

Example:

#KEY	#NO	#PKG	#IL	#LOC	#PIN	#CELL	#PINSPEC	#ATTR	#BSCAN	#PROBE	#BSATTR
@											
#BEGIN											
#DATA1											
#DATA2											
	1	1	259	I1	CXXIN	ZW9C3X12ZC		I	N	Y	
	2	2	258	I1	CXXOUT	ZW9C3X12ZC		O	N	Y	
	3	3	151	I1	CXMTCLKOUT	ZW9T3BA0234WQCPC	3S	B	Y	Y	B
	4	4	266	I1	CXRST_N	ZW9T3D21YWC		I	N	Y	I
	5	5	265	I1	CXPOWREN	ZW9T3D21YWC		I	N	Y	I
	6	6	101	I1	CXRSTOUT_N	ZW9T3BA0234WQCPC	3S	B	Y	Y	B
#END											

#PIN column : name of external pin.

This is required information. The naming rule is as follow:

- name is an arbitrary string
- signal pin: its name should match the top module port name
- power pin: it does not need to be in a net. Its name can be overlapped or blank.
(However, #IL is required and #ATTR is PorG)
- NC pin: black is allowed. (but #IL is required)

Example:

#KEY	#NO	#PKG	#IL	#LOC	#PIN	#CELL	#PINSPEC	#ATTR	#BSCAN	#PROBE	#BSATTR
@											
#BEGIN											
#DATA1											
#DATA2											
	1	1	259	I1	CXXIN	ZW9C3X12ZC		I	N	Y	
	2	2	258	I1	CXXOUT	ZW9C3X12ZC		O	N	Y	
	3	3	151	I1	CXMTCLKOUT	ZW9T3BA0234WQCPC	3S	B	Y	Y	B
	4	4	266	I1	CXRST_N	ZW9T3D21YWC		I	N	Y	I
	5	5	265	I1	CXPOWREN	ZW9T3D21YWC		I	N	Y	I
	6	6	101	I1	CXRSTOUT_N	ZW9T3BA0234WQCPC	3S	B	Y	Y	B
#END											

#CELL column : name of IO buffer library cell.

This information does not affect any added circuit.

If pin is power pin, cell name is not required.

Example:

#KEY	#NO	#PKG	#IL	#LOC	#PIN	#CELL	#PINSPEC	#ATTR	#BSCAN	#PROBE	#BSATTR
@											
#BEGIN											
#DATA1											
#DATA2											
	1	1	259	I1	CXXIN	ZW9C3X12ZC		I	N	Y	
	2	2	258	I1	CXXOUT	ZW9C3X12ZC		O	N	Y	
	3	3	151	I1	CXMTCLKOUT	ZW9T3BA0234WQCPC	3S	B	Y	Y	B
	4	4	266	I1	CXRST_N	ZW9T3D21YWC		I	N	Y	I
	5	5	265	I1	CXPOWREN	ZW9T3D21YWC		I	N	Y	I
	6	6	101	I1	CXRSTOUT_N	ZW9T3BA0234WQCPC	3S	B	Y	Y	B
#END											

#PINSPEC column : pin specification

This information is required when specifying 3-state output pin.

Make sure that specify both #ATTR: O and #PINSPEC: 3S. If not specified #PINSPEC, that pin is considered as 2-state output.

Example:

#KEY	#NO	#PKG	#IL	#LOC	#PIN	#CELL	#PINSPEC	#ATTR	#BSCAN	#PROBE	#BSATTR
@											
#BEGIN											
#DATA1											
#DATA2											
	1	1	259	I1	CXXIN	ZW9C3X12ZC		I	N	Y	
	2	2	258	I1	CXXOUT	ZW9C3X12ZC		O	N	Y	
	3	3	151	I1	CXMTCLKOUT	ZW9T3BA0234WQCPC	3S	B	Y	Y	B
	4	4	266	I1	CXRST_N	ZW9T3D21YWC		I	N	Y	I
	5	5	265	I1	CXPOWREN	ZW9T3D21YWC		I	N	Y	I
	6	6	101	I1	CXRSTOUT_N	ZW9T3BA0234WQCPC	3S	B	Y	Y	B
#END											

#ATTR column : IO attribute

It contains definition of the functional IO attributes of external pins.

This is an important information, blank is not allowed.

There are 5 types of attributes:

- I : input
- O : output
- B : inout
- P : power
- G : gound

Example

#KEY	#NO	#PKG	#IL	#LOC	#PIN	#CELL	#PINSPEC	#ATTR	#BSCAN	#PROBE	#BSATTR
@											
#BEGIN											
#DATA1											
#DATA2											
	1	1	259	I1	CXXIN	ZW9C3X12ZC		I	N	Y	
	2	2	258	I1	CXXOUT	ZW9C3X12ZC		O	N	Y	
	3	3	151	I1	CXMTCLKOUT	ZW9T3BA0234WQCPC	3S	B	Y	Y	B
	4	4	266	I1	CXRST_N	ZW9T3D21YWC		I	N	Y	I
	5	5	265	I1	CXPOWREN	ZW9T3D21YWC		I	N	Y	I
	6	6	101	I1	CXRSTOUT_N	ZW9T3BA0234WQCPC	3S	B	Y	Y	B
#END											

1.3.3 Boundary scan testing information

#BSCAN column : BS register addition flag

It defines a BS register addition instruction. This information is required but

There are 3 instructions:

- “Y” : add control and observe cells
- “N” : DO NOT additional
- “C” : add observe-only cell

NOTE:

- If a pin needs to pass input values of external pins inside upon normal system operation or BS test, set it to a non-target pin for BS register addition.
Ex: test mode pin, reset pin, ...
- Set “C” for a pin which needs to pass external pin values inside upon scan (only making the MUXSCAN chain from BS registers) / SINGEN operation. However, minimize the settings since the detection rate may fall.

Example

#KEY	#NO	#PKG	#IL	#LOC	#PIN	#CELL	#PINSPEC	#ATTR	#BSCAN	#PROBE	#BSATTR
@											
#BEGIN											
#DATA1											
#DATA2											
	1	1	259	I1	CXXIN	ZW9C3X12ZC		I	N	Y	
	2	2	258	I1	CXXOUT	ZW9C3X12ZC		O	N	Y	
	3	3	151	I1	CXMTCLKOUT	ZW9T3BA0234WQCPC	3S	B	Y	Y	B
	4	4	266	I1	CXRST_N	ZW9T3D21YWC		I	N	Y	I
	5	5	265	I1	CXPOWREN	ZW9T3D21YWC		I	N	Y	I
	6	6	101	I1	CXRSTOUT_N	ZW9T3BA0234WQCPC	3S	B	Y	Y	B
#END											

#PROBE column : small pin test flag

- “Y” : this pin is probed. It is not tested in few pin testing (IOWRAP test)
- “N” : this pin is not probed. It is tested in few pin testing (IOWRAP test)

Example

#KEY	#NO	#PKG	#IL	#LOC	#PIN	#CELL	#PINSPEC	#ATTR	#BSCAN	#PROBE	#BSATTR
@											
#BEGIN											
#DATA1											
#DATA2											
	1	1	259	I1	CXXIN	ZW9C3X12ZC		I	N	Y	
	2	2	258	I1	CXXOUT	ZW9C3X12ZC		O	N	Y	
	3	3	151	I1	CXMTCLKOUT	ZW9T3BA0234WQCPC	3S	B	Y	Y	B
	4	4	266	I1	CXRST_N	ZW9T3D21YWC		I	N	Y	I
	5	5	265	I1	CXPOWREN	ZW9T3D21YWC		I	N	Y	I
	6	6	101	I1	CXRSTOUT_N	ZW9T3BA0234WQCPC	3S	B	Y	Y	B
#END											

NOTE:

This information can be omitted, default value is “Y”.

Specify “Y” for all pins if the chip does not apply a small pin test.

#BSATTR column : BS addition IO attribute

It is one of reserved words for the BS addition IO attribute.

Character	Attribute	BS addition contents
I	Input	Add for input only
O2	2 state output	Add for output only
O3	3 state output	Add for output data/enable
B	Bi-directional	Add for input data/output data/enable
P	Power supply	No addition
G	Ground	No addition

Basically, adding a BS register is according to an attribute used by a user function. However, for a bi-directional cell, only if IN/OUT enables are directly fixed, I or O2 attribute can be specified so that an IO cell will face a specific direction.

Similarly, as for a 3 state output cell, only if OUT enables are directly fixed, O2 attribute can be specified so that an IO cell will face the output direction.

For both of IO cells, if their directions are controlled by OUT enable signal from internal logic, specify based on the IO cell attribute regardless of user function.

Example

#KEY	#NO	#PKG	#IL	#LOC	#PIN	#CELL	#PINSPEC	#ATTR	#BSCAN	#PROBE	#BSATTR
@											
#BEGIN											
#DATA1											
#DATA2											
	1	1	259	I1	CXXIN	ZW9C3X12ZC		I	N	Y	
	2	2	258	I1	CXXOUT	ZW9C3X12ZC		O	N	Y	
	3	3	151	I1	CXMTCLKOUT	ZW9T3BA0234WQCPC	3S	B	Y	Y	B
	4	4	266	I1	CXRST_N	ZW9T3D21YWC		I	N	Y	I
	5	5	265	I1	CXPOWREN	ZW9T3D21YWC		I	N	Y	I
	6	6	101	I1	CXRSTOUT_N	ZW9T3BA0234WQCPC	3S	B	Y	Y	B
#END											

#BS_COMPLIANCE column :

It is a fixed value for external pin (L/H or 0/1) required for running TAP controller.

The setting affects the contents of BSDL file but not logic.

BS register can be added to this pin.

Example

#KEY	#NO	#PKG	#IL	#LOC	#PIN	#CELL	#PINSPEC	#ATTR	#BSCAN	#PROBE	#BSATTR	#BS_COMPLIANCE	#TESTPIN
@												lowrap	
#BEGIN													
#DATA1													BS
#DATA2													
	1	1	259	I1	CXXIN	ZW9C3X12ZC		I	N	Y		1	
	2	2	258	I1	CXXOUT	ZW9C3X12ZC		O	N	Y			
	3	3	151	I1	CXMTCLKOUT	ZW9T3BA0234WQCPC	3S	B	Y	Y	B		
	4	4	266	I1	CXRST_N	ZW9T3D21YWC		I	N	Y	I	0	
	5	5	265	I1	CXPOWREN	ZW9T3D21YWC		I	N	Y	I		
	6	6	101	I1	CXRSTOUT_N	ZW9T3BA0234WQCPC	3S	B	Y	Y	B		
#END													

#TESTPIN column : test pin information

It instructs test pin allocations (**shared circuit addition**) for DFT and others

Format:

	#TESTPIN	
#DATA1	①	⇒ An addition group name
#DATA2	②	⇒ A test mode signal name (hierarchy name)
	③	⇒ An instruction of shared pin for test and user logic

- Addition group name :

An addition group name is arbitrary.

For this name, link addition group names in a control file and an execution shell.

- Test mode signal name

<hier1>.<hier2>...<hiern>.<sig_name>[busbit] : 'L' or 'H'

A delimiter for hierarchy level is ".", and a delimiter for active value is ":".

- Instruction of shared pin for test and user logic

<Pin name> [: addition_attribute] [: fixed value when unused (L/H)] [+ instance for connection destination.terminal]...

- Pin name: Specify an arbitrary pin name or a pin name prepared in the SCL library
- Addition_attribute: I/O/B -> Addition of arbitrary logic to IO cell.
- "fixed value when unused" can be specified only if the attribute is input I or bi-

directional B. Be sure to set the addition attribute when setting the “fixed value when unused”. Error if the “fixed value when unused” is already set and the addition attribute is neither I nor B.

- Can add as many [+instance for connection destination terminal] as needed

NOTE:

About specification of an arbitrary pin name or a pin name defined in SCL

- A standard test pin for each DFT method defined in SCL

Enter its test pin name only in #TESTPIN.

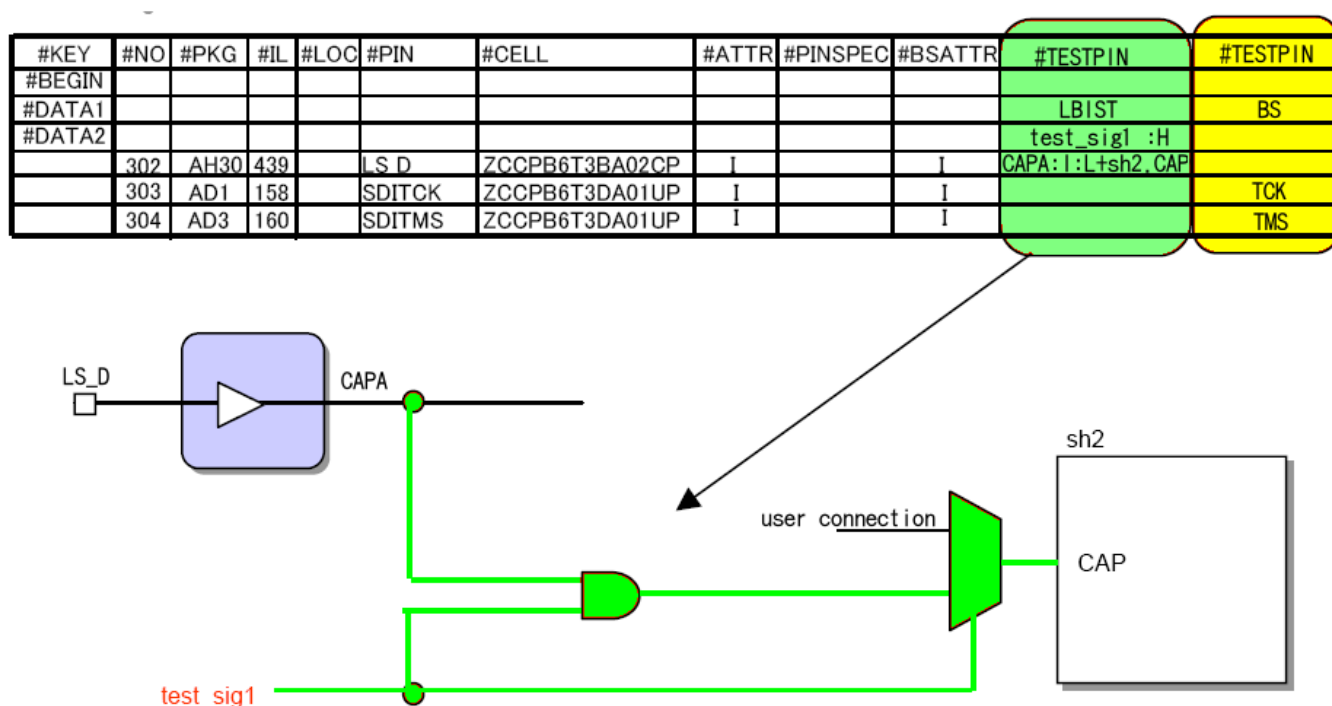
Ex: TCK, TMS, scan_enable, edt_clock, edt_update, ...

- A test pin which is not defined in SCL

Enter the followings in #TESTPIN.

<Test signal name>: <IO attribute (I/O, required)>:<Fixed value for signal when unused (L/H)>

Example



In above figure, green column is test pin definition of LBIST mode, test mode signal is test_sig1 with active value is H.

At row of pin LS_D information, an shared circuit is added in net CAPA to pin sh2.CAP with control signal is test_sig1. SCAAP adds an AND gate between CAPA signal and test_sig1 then connects its output to sh2.CAP, but that pin has been already driven by a user connection, so SCAAP add a MUX to switch the old connection and new one with select signal is test_sig1.

About yellow column, it is test pin definition of BS mode. This mode is controlled by TAP controller so no need specify the test mode in row #DATA2. About the instruction information, pin

name is prepared in SCL library so no need to specify <addition_attribute>, <fixed value when unused> and <instance for connection destination terminal>. AD1 pin is connected to TAP clock pin, AD3 is connected to TAP test mode select pin.

1.4 Control files (CTLF)

Control file is a text file. It contains 17 items belong to 6 main parts:

- Product information
- Boundary scan test information
- Execution mode of SCAAP
- Information about additional logics
- Information about function signals will be added by SCAAP

1.4.1 Product information

This part specifies cell series name, core cell library name, product name, package name and applied DFT method of the chip (special for LBIST flow).

Example

```
SERIES          RC03G ;
LIBRARY         RC03BTHH ;
PRODUCT        CHIPA ;
PACKAGE        BGA544 ;
DFT LBIST_METHOD SINGEN ;
```

- **Series name**

Syntax: **SERIES** <cell series name> ;

<cell series name> is registered in BODY.scl file

- **Library name**

Syntax: **LIBRARY** <core cell library name> ;

<core cell library name> is registered in BODY.scl file

- **Product information**

Syntax: **PRODUCT** <product name> ;

This product name will be reflected in every “entry” area within a BSDL file.

It does not affect circuit addition. It can be set as name “DUMMY” , and then modify BSDL manually.

- **Package name**

Syntax: **PACKAGE** <package name> ;

Its package name will be reflected in “generic” and “constant” within a BSDL file.

It does not affect circuit addition. It can be set as name “DUMMY” , and then modify BSDL manually.

- **DFT method**

Syntax: **DFT LBIST_METHOD** *<keyword for LBIST tool>* ;

This information is specified ONLY when the chip is applied in LBIST flow.

<keyword for LBIST tool> in Renesas is “SINGEN”

1.4.2 Boundary scan test information

NOTE:

This information is not required if boundary scan test is not executed.

Example

BOUNDARY_SCAN CONFORMANCE	STD_1149_1_1993 ;
BOUNDARY_SCAN BSDL	STD_1149_1_1994 ;
BOUNDARY_SCAN RESET	YES ;
BOUNDARY_SCAN MODE	EXTEST, SAMPLE, BYPASS, IDCODE, EXMBIST;
BOUNDARY_SCAN ID_VERSION	0000 ;
BOUNDARY_SCAN ID_PART	1000000010011111 ;

- **Boundary scan standard version**

Syntax: **BOUNDARY_SCAN CONFORMANCE** *<keyword for boundary scan compliance>* ;

Keyword for boundary scan compliance: STD_1149_1_1990
 STD_1149_1_1993
 STD_1149_1_2001

Currently, SCAAP supports STD_1149_1_1993 ONLY.

- **BSDL format standard version**

Syntax: **BOUNDARY_SCAN BSDL** *<keyword for NSDL format>* ;

Keyword for BSDL format: STD_1149_1_1990
 STD_1149_1_1994
 STD_1149_1_2001

Currently, SCAAP supports STD_1149_1_1994 ONLY.

- **Information about RESET terminal (TRST)**

Syntax: **BOUNDARY_SCAN RESET [YES/NO]** ;

Default: YES → reset terminal always exists.

- **Boundary scan operation modes**

Syntax: **BOUNDARY_SCAN MODE** [*mode1*], [*mode2*], ... ;

Keywords for test mode: BYPASS, EXTEST, SAMPLE, IDCODE, CLAMP, HIGHZ, INTEST, HUDI

Above keywords are from “B/S INSTRUCTION SELECT” information in BODY.scl file of SCL library.

- **Version code for ID code**

Syntax: **BOUNDARY_SCAN ID_VERSION** <*The version for ID code*> ;

The version for ID code contains 4 bits (0/1). This information is from part “B/S INSTRUCTION” in BODY.scl file of SCL library.

- **Part code for ID code**

Part code contains 16 bits (0/1). It is manufacturer number, automatically set by a tool.

Part code of RENESAS is 01000100011.

Syntax: **BOUNDARY_SCAN ID_PART** <*The product number for ID code*> ;

1.4.3 Execution test mode of SCAAP

This information is used to set a mode for test logic addition and a prefix name to be added in the mode. It is required.

Example

```
GENMODE BS ;
PREFIX SCAAP_BS_;
```

- **Specifying a test circuit creation mode**

Syntax: **GENMODE** <*keywords for test circuit creation mode*> ;

<keywords for test circuit creation mode>:

MTEST: Specify this mode in case of the addition of module test circuit.

BS: Specify this mode in case of the addition of boundary scan circuit.

LBIST: Specify this mode in case of the addition of LBIST test pin.

- **Specifying a prefix name for SCAAP added circuit and a net name**

Syntax: **PREFIX** <*prefix name*> ;

Any prefix name used in a net list cannot be set. If prefix names are specified in both a control file and a SCAAP execution shell, the name in the control file is valid.

1.4.4 Information about additional logics

This information is used when adding an arbitrary circuit.

- Adding a circuit at any position in module by SCAAP

Syntax:

ADDCIRCUIT [PIN/TERMINAL/MODULE/BLOCK/TOPBLOCK] <Pin name/hierarchy instance port name/module name/hierarchy module instance name>

<NAME The name of a circuit to add >

<[INTERNAL/EXTERNAL]>

<INSTNAME [Reference instance name] [Instance name after addition]>

<NETNAME [Reference net name] [Net name after addition]>

<NETFUNC [Reference function name] [Function name]>

<NETGLOBAL [Reference global name]> ;

- Setup a PIN: <pin name>

When setting pin name, option INTERNAL or EXTERNAL is invalid and it is ignored if setting.

In PINC.scl file for SCL library:

Using PIN to refer to a pin name

Using TERMINAL to refer to a top port

Using GATE to refer to an IO buffer instance

- Setup a TERMINAL: < hierarchy instance port name>

Setting option INTERNAL to insert a circuit to a port net inside a module.

Setting option EXTERNAL to insert a circuit to a port net outside a module.

In CNTLC.scl file of SCL library, using SIGNAL to refer to the connection net of the target port.

Example

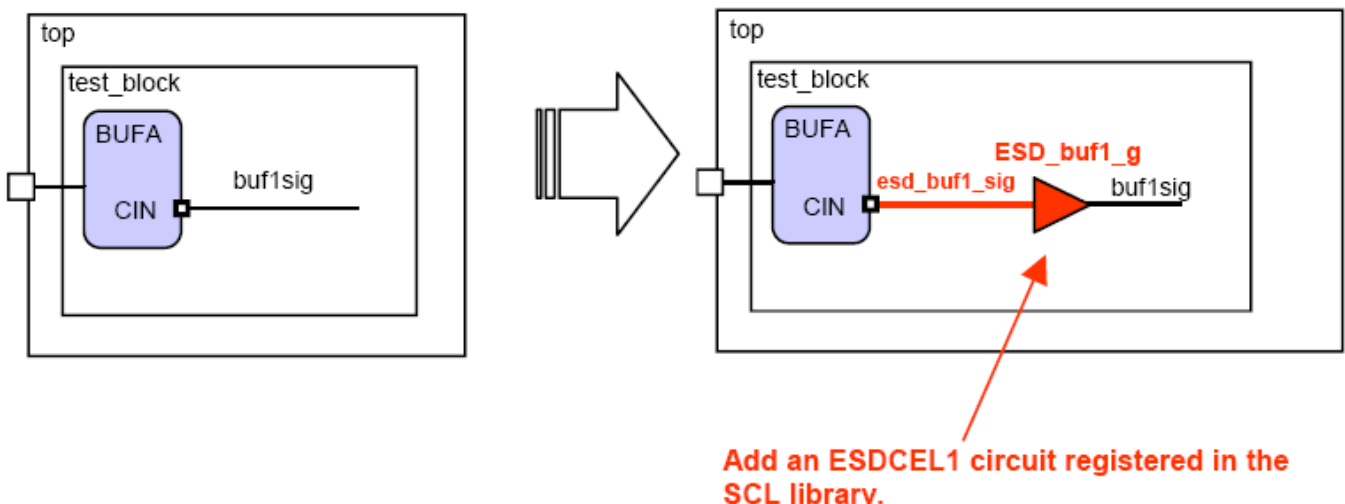
```
ADDCIRCUIT TERMINAL test_block.BUFA.CIN
```

```
NAME ESDCEL 1
```

```
EXTERNAL
```

```
INSTNAME #G1 ESD_buf 1 _g
```

```
NETNAME #l esd_buf1_sig ;
```



- Setup a MODULE: <module name>

Setting option INTERNAL to insert a circuit into a module.

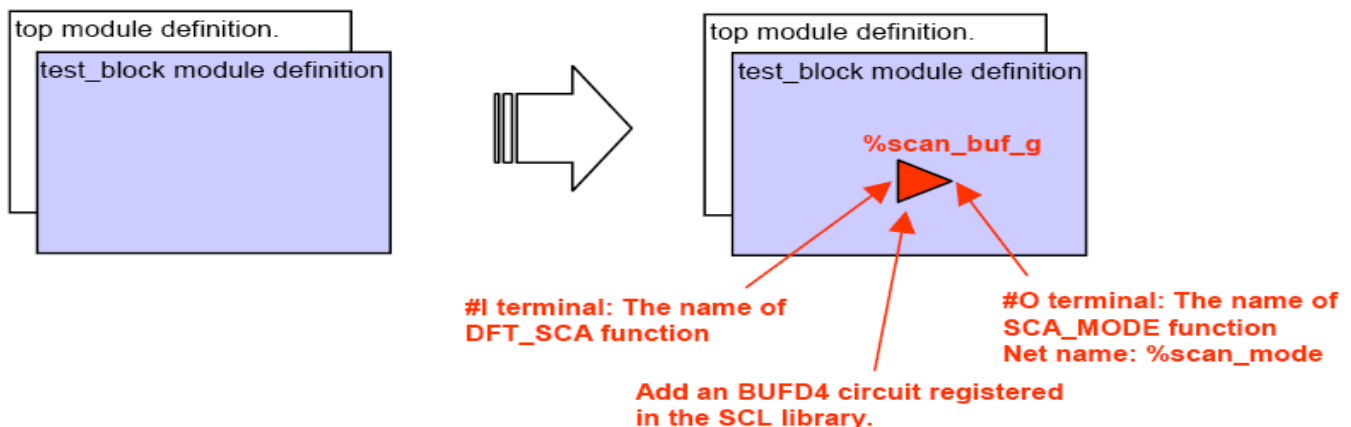
In CNTLC.scl file of SCL library, there is no target for reference. Place a circuit within a module. (change of module definition).

Setting option EXTERNAL to insert a circuit to a port net outside a module.

In CNTLC.scl file of SCL library, using TERSIG to refer to an instance port connection netlist of the target module.

Example

```
ADDCIRCUIT MODULE test_block
NAME BUFD4
INTERNAL
INSTNAME #G1 %scan_buf_g
NETFUNC #I DFT_SCAN
NETNAME #O %scan_mode
NETFUNC #O SCAN_MODE ;
```



- Setup a BLOCK: < hierarchy module instance name>

Setting option INTERNAL to insert a circuit into a module.

In CNTLC.scl file of SCL library, there is no target for reference. Place a circuit within a specified module instance ONLY. (NOT change of module definition).

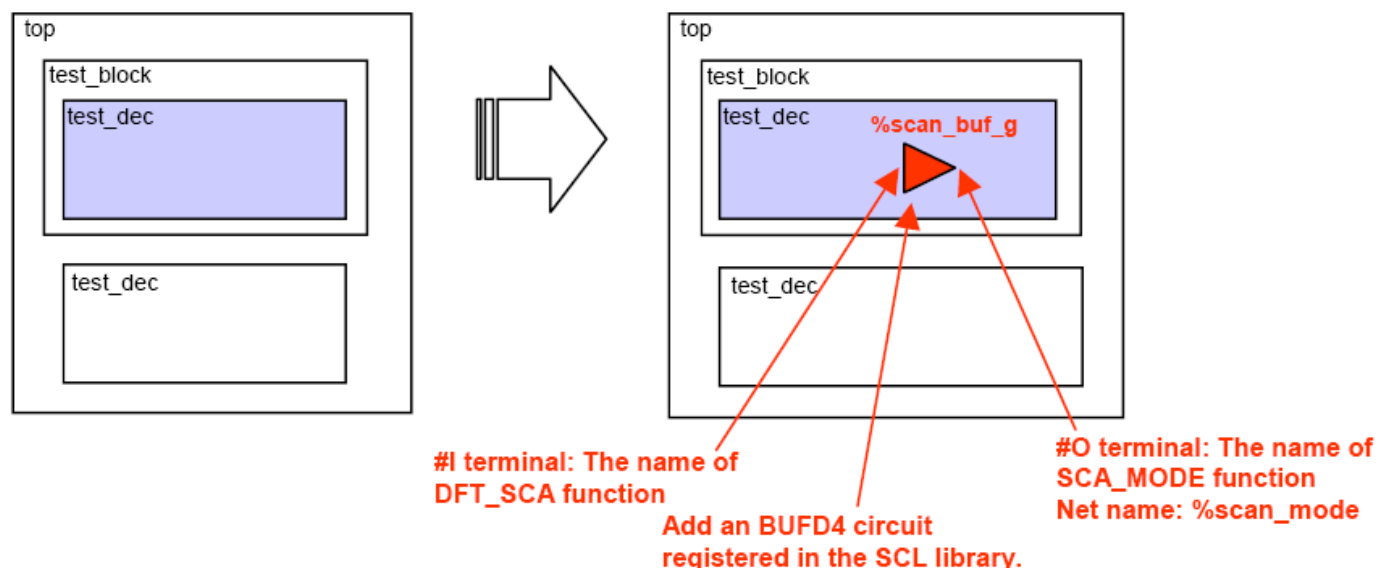
Setting option EXTERNAL to insert a circuit to a port net outside a module.

In CNTLC.scl file of SCL library, using TERSIG to refer to an instance port connection netlist of the target module instance.

Example

```
ADDCIRCUIT BLOCK test_block.test_dec
NAME BUFD4
```

```
INTERNAL
INSTNAME #G1 %scan_buf_g
NETFUNC #I DFT_SCAN
NETNAME #O %scan_mode
NETFUNC #O SCAN_MODE ;
```



- Setup a TOPBLOCK:

DO NOT need to set a specific insertion location for TOPBLOCK since TOPBLOCK is uniquely defined. Thus, omit to specify it. (If TOPBLOCK is specified, MUST NOT enter any specific insertion location. Error if entering.)

- **Adding a circuit when additional style defined in SCL library**

Syntax: **ASSIGN CIRCUIT** <keyword for a control circuit> ;

<keyword for a control circuit> is registered in part “PIN/CIRCUIT ASSIGN” in BODY.scl of SCL library.

Example:

```
ASSIGN CIRCUIT TestKompres ;
→ Be sure to specify it when a test pin for TestKompres is added.
ASSIGN CIRCUIT MINORI2 ;
→ Be sure to specify it when a test pin for MINORI is added.
```

1.4.5 Information about function signals will be added by SCAAP

- **Specifying the addition / connection of signals**

Syntax:

ADDSIGNAL [TERMINAL/BLOCK/TOPBLOCK] <Hierarchy instance port name/hierarchy module instance name>

<ADDNET *Addition net name* >
<DIRECTION [OUTPUT/INPUT]>
<CONNECT [GLOBAL/CONTROL]>
<FUNCTION *Function name* >
<FORCE>
<ACTIVE [0/1]>;

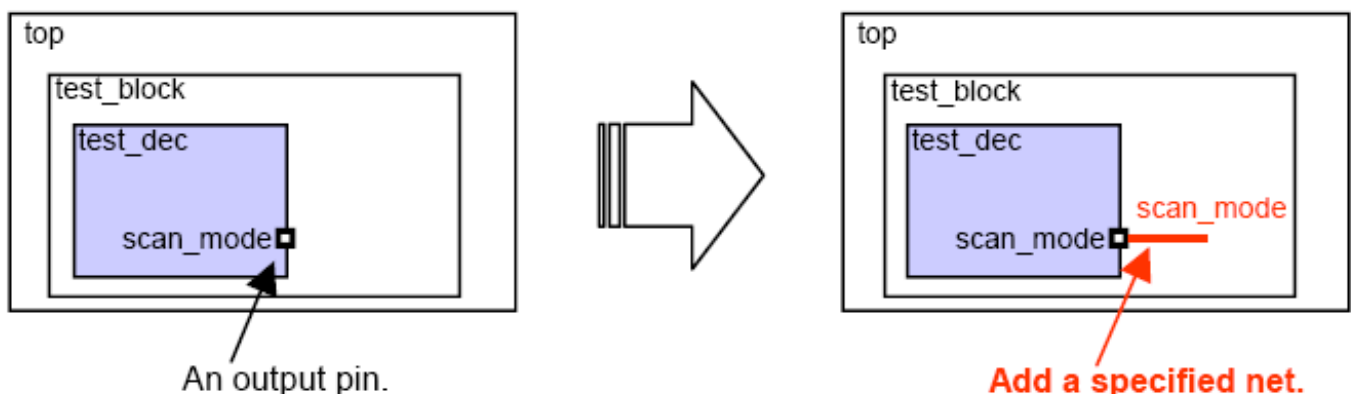
NOTE:

- **TERMINAL**-> A hierarchy instance port name is described. It does not include a top module name.
- **BLOCK**-> A hierarch module instance name is described. It does not include a top module name. If the instance name is omitted, consider as a top module.
- **TOPBLOCK**-> Hierarchy instance port name/hierarchy module instance name cannot be described.
- **ADDNET**, **DIRECTION**, **CONNECT**, **FUNCTION**, **ACTIVE** and **FORCE** can be described in any order. This record can be set more than once.

– Adding a net to a terminal

Example

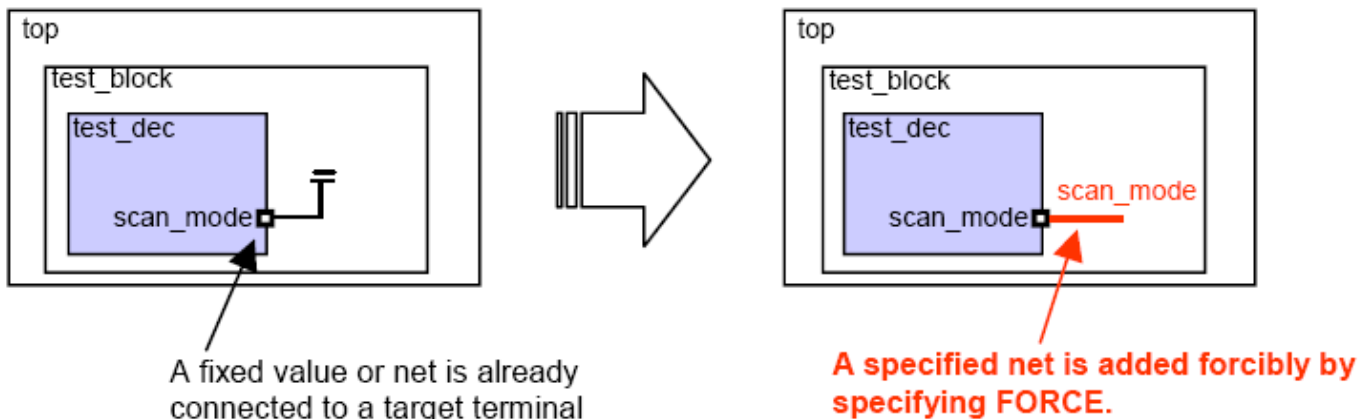
```
ADDSIGNAL TERMINAL test_block.test_dec.scan_mode
ADDNET scan_mode
DIRECTION OUTPUT
FUNCTION SCAN_MODE ;
```



- Adding a specified net forcibly to a terminal

Example:

```
ADDSIGNAL TERMINAL test_block.test_dec.scan_mode
ADDNET scan_mode
DIRECTION OUTPUT
FUNCTION SCAN_MODE
FORCE ;
```

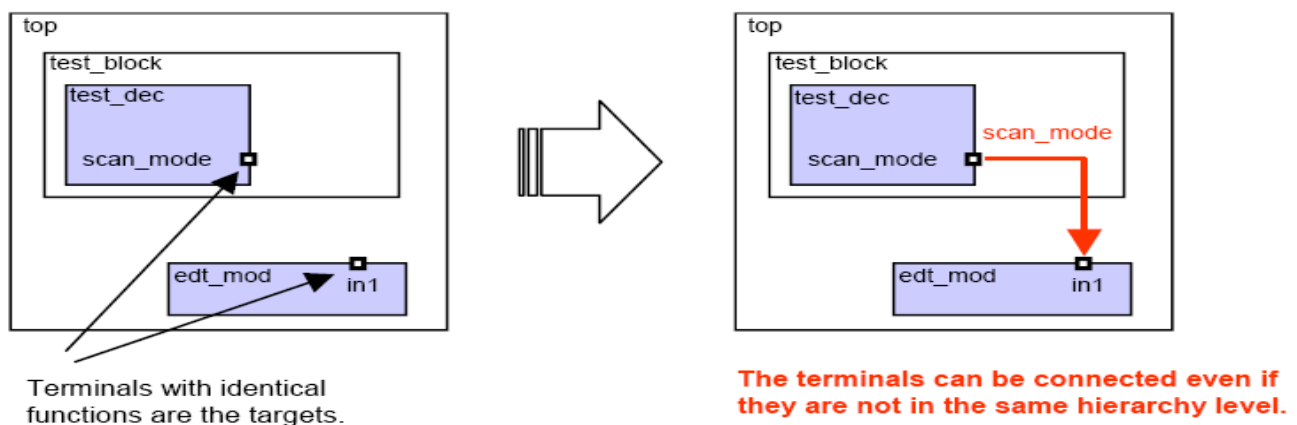


- Terminal connection

Example

```
ADDSIGNAL TERMINAL edt_mod.in1
ADDNET scan_mode
DIRECTION INPUT
FUNCTION SCAN_MODE ;
```

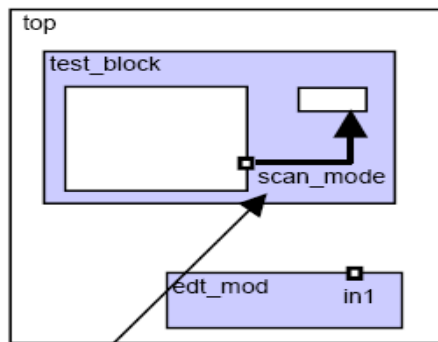
```
ADDSIGNAL TERMINAL test_block.test_dec.scan_mode
ADDNET scan_mode
DIRECTION OUTPUT
FUNCTION SCAN_MODE ;
```



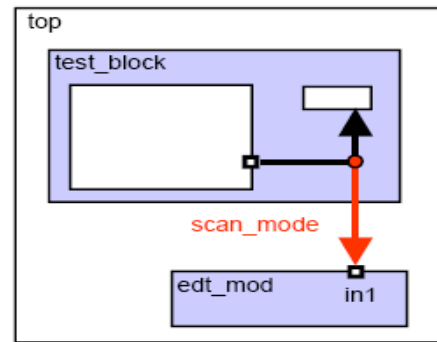
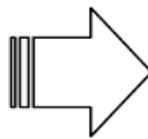
- Block connection

Example

```
ADDSIGNAL TERMINAL edt_mod.in1
ADDNET scan_mode
DIRECTION INPUT
FUNCTION SCAN_MODE ;
```



A net within a module is the target.



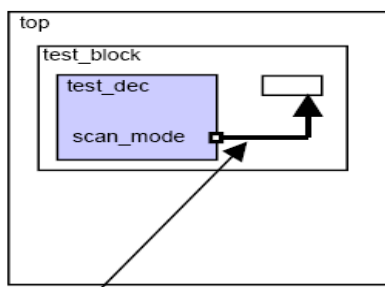
Take from the net, and connect to the same function terminal of the circuit.

- Library connection: connect a circuit terminal registered in the SCL library.

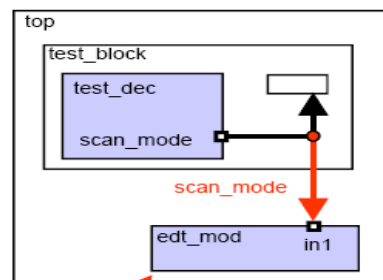
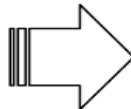
Example:

```
ADDSIGNAL TERMINAL test_block.scan_mode
ADDNET scan_mode
DIRECTION OUTPUT
FUNCTION SCAN_MODE ;
```

In SCL Library, setting SCAN_MODE as a function name of an in1 terminal for an edit_mode circuit.



Specify SCAN_MODE as a relevant net function.

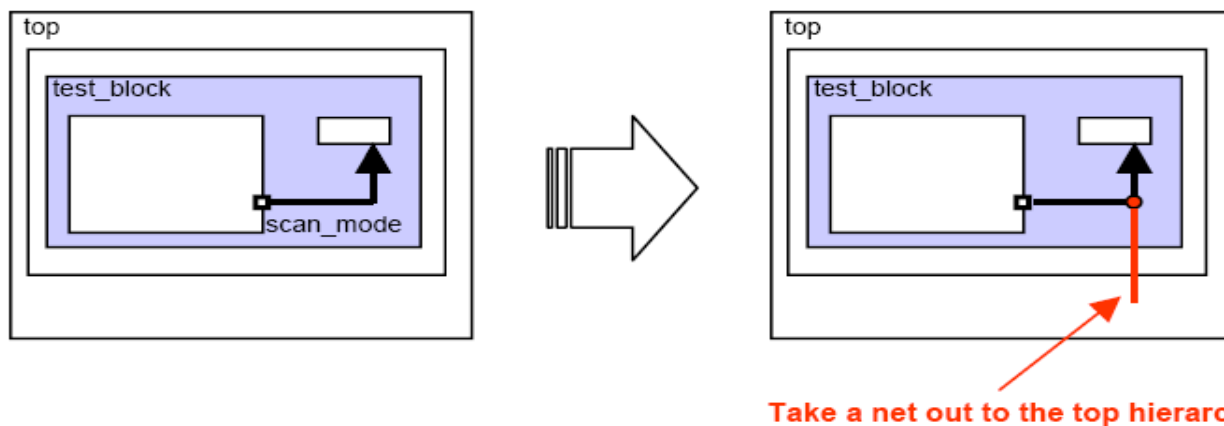


In SCAAP, add a circuit registered in the SCL library, and connect to the same function terminal of the added circuit.

- Net extraction

```
ADDSIGNAL BLOCK test_block
ADDNET scan_mode
```

DIRECTION OUTPUT FUNCTION SCAN_MODE ;

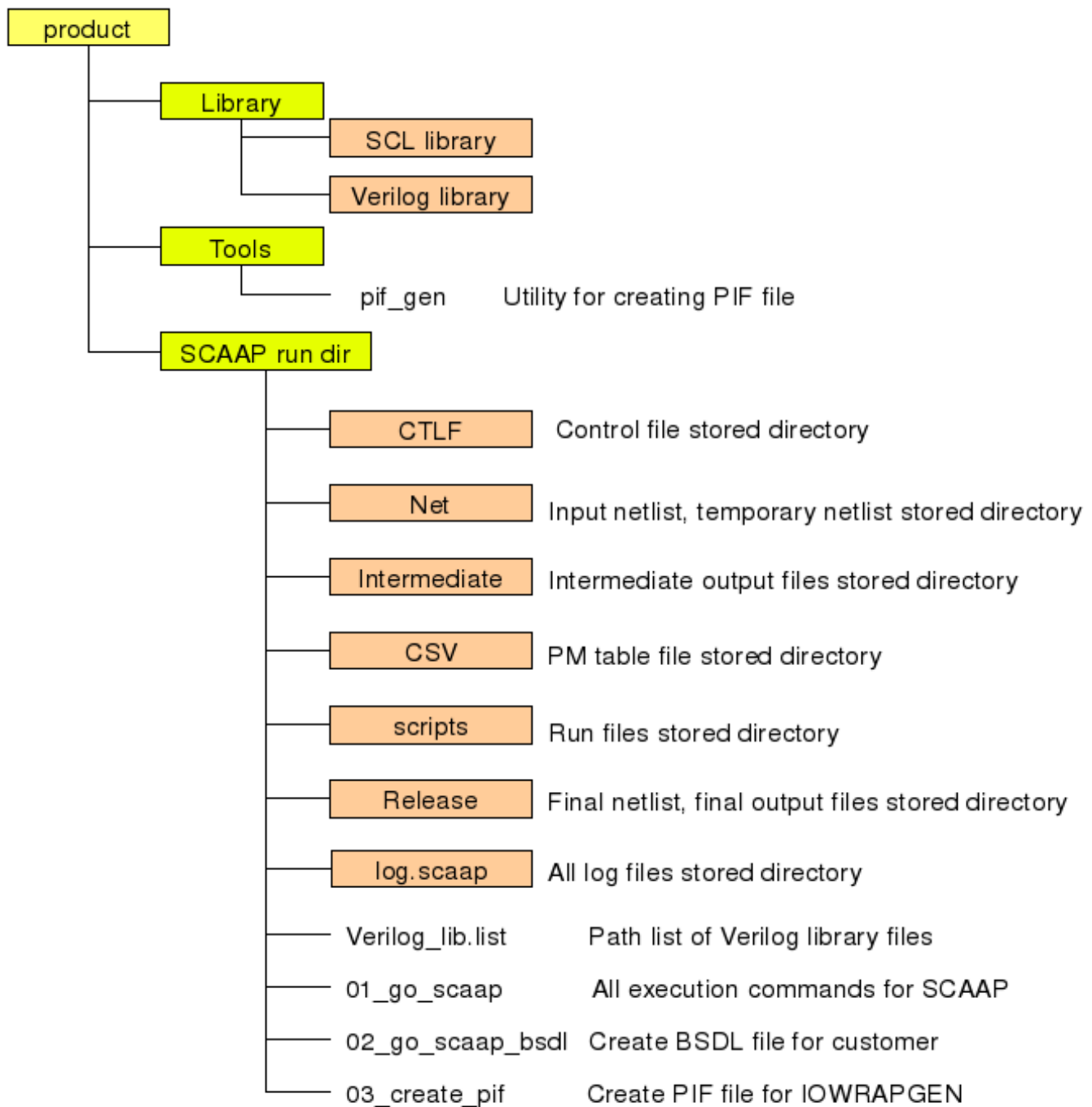


2. Execution Environment

This part introduces about SCAAP environment, SCAAP working flow to insert boundary scan circuit completely into top netlist and format of SCAAP main run file.

2.1 Structure of SCAAP environment

Below figure is an example of SCAAP environment.



2.2 SCAAP working flow

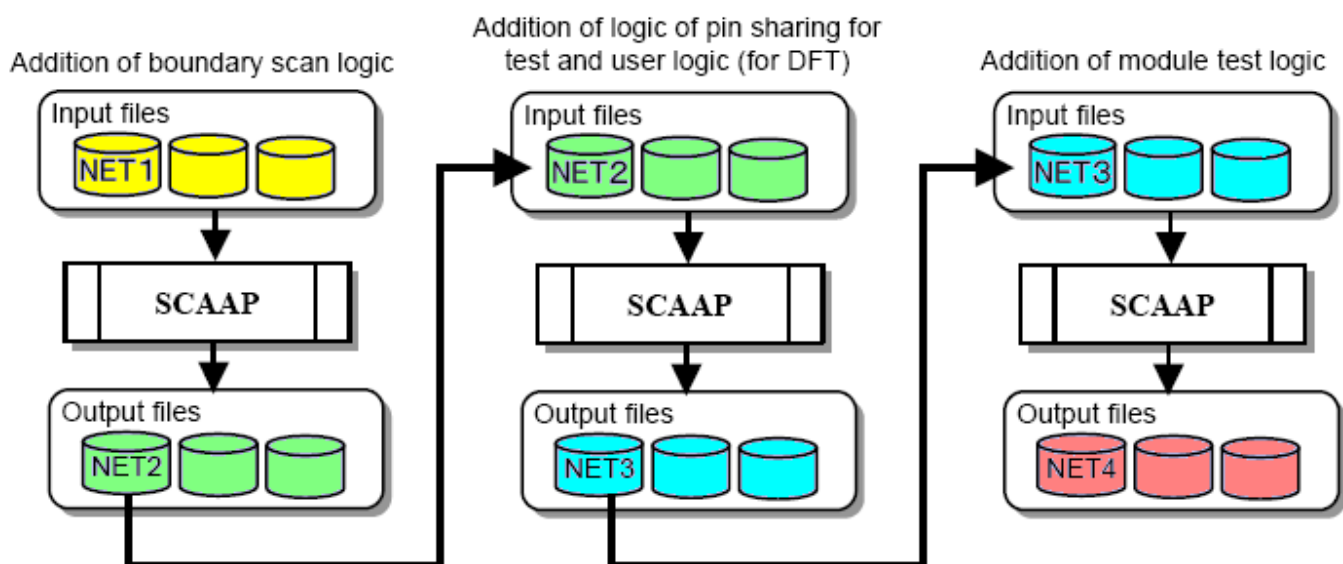
Basically, SCAAP can execute 1 method per execution time. So when there are multiple types of test logics added into netlist, separate them into many run files. Input netlist of one run is the output netlist in the last test logic addition.

For example, in below figure, there are 3 types of test logics need to be added into netlist.

They are : module test logic, DFT shared pin logic and boundary scan circuit.

Assume, BS cell is not used as MUX scan FF.

The priority is: adding boundary scan circuit, adding DFT shared logics and adding module test logic. The original netlist (NET1) is input netlist of adding boundary scan chain. So the output netlist of adding boundary scan chain (NET2) is input netlist of adding boundary scan circuit. Then the output netlist of adding boundary scan circuit (NET3) is input netlist of adding module test logic. The final netlist (NET4) has 3 types of test logics as expected.



2.3 SCAAP run file

Run file contains the commands for SCAAP to execute one method.

For example, below file is run file for adding boundary scan circuit

```
source /common/appl/Renesas/scaap/dotfiles/scaap_v05r0401.CSHRC \
scaap ${pre_net} \
-gset $gset \
-ctlf ${pre_ctlf} \
```

```
-pm    ${pre_pm} \
-scope ${root} \
-ov    ${post_net} \
-obsdl ${post_bsdl} \
-iloc  ${post_iloc} \
-mctl  ${post_mctl} \
-dftp  ${post_dftp} \
-pif    ${post_pif} \
-bsrseq ${post_bsro} \
-pathlist ${post_path} \
-f ${libf} \
-scldir ${scldir} \
    +sclex+.scl\
    +sclex+${sclex_bsrs}\
    +sclex+${sclex_prod}\
-info bpul ppul bsep \
-smryl  ${slist}.smry \
-erri   ${slist}.error \
-infol  ${slist}.info \
|& tee  ${slist}.log
```

2.3.1 Option : GMODE - MUST

Syntax: **-gmode [test circuit creation mode]**

Function: specify test circuit creation mode.

There are 10 modes:

- bs : select this mode if TAP controller and BSR are added
- bstap : select this mode if only TAP controller is added
- bsreg : select this mode if only BSRs are added
- sablk : select this mode for adding module test
- lbist : select this mode for adding for SINGEN test pin

NOTE: DO NOT set this option if already setting GENMODE parameter in control files.

2.3.2 Option: GSET - OPTIONAL

Syntax: **-gset [additional group name]**

Function: it sets name of additional group circuit which shall be added into netlist

For “additional group name”, refer to the name in row @DATA1 of column #TESTPIN in PM file.

2.3.3 Option: TARGET – MUST

Syntax: **-target [target type]**

Function: specify target which shall be applied by SCAAP

There are 2 types:

lsi: entire chip (default)

blk: specified block of chip

2.3.4 Option: TCL - OPTIONAL

Syntax: **-tcl [TCL file name]**

Function: specify input TCL files

2.3.5 Options related to user information

Syntax: **-pinspec "user/force"**

Function: selects user-specified attribute or IO buffer cell attribute

user: adopt user-specified attribute (default)

force: adopt IO buffer attribute

Syntax: **-bsrbind "yes/no"**

Function: specify availability of sharing a BS register for the enable side

no: (default)

2.3.6 Option: SCOPE - MUST

Syntax: **-scope [top module name]**

Function: set root block name

2.3.7 Options for defining prefix

Function : specify prefix name for SCAAP added circuit

Syntax:

-prefix	"common name"
-preblkdfname	"block def. Name"
-presigname	"signal name"
-preblkrfname	"block ref. Name"
-preblkedgname	"block edge name"
-prepinname	"LSI pin name"

2.3.8 Options for VCC/GND definition

Syntax:

-powersig "power signal name"
-groundsig "ground signal name"

2.3.9 Options for input files

Specifying SCL library files

Syntax:

-scl "scl file name" ...
-sclmdir "scl directory name" ...
+scl ext+"scl file extension"+...+
-scll "scl print file name"

Specifying control file

Syntax:

-ctlf "ctlf file name" ...
-ctlfdir "ctlf directory name" ...
+ctlf ext+"ctlf file extension"+...+

Specifying PM file

Syntax:

-pm "pm file name" ...
-pmdir "pm directory name" ...
+pm ext+"pm file extension"+...+

2.3.10 Options for output files

Syntax:

-obsdl "output bsdl file name"
-ospf "output spf directory name"
-fctl "output fandiv control file name"
-dftp "output DFT-Profile file name"
-mctl "output Minori Control file name"
-iloc "output Instance Location file name"
-pif "output Pin Information file name"
-bsrseq "output B/S seq ordering file name"
-pathlist "output STA path list file name"

NOTE:

- If there is no boundary scan test: the output of BSDL and PIF files is not required
- If there is no scan or LBIST test pin addition, the output of DPF file is not required
- If there is no MBIST test pin addition, the output of MC file is not required

3. OUTPUTS

This part explains the information of output files after running SCAAP. There are 5 types of SCAAP output files:

- logfiles contain information during SCAAP run
- BSDL file contain information about boundary scan circuit inserted into top netlist
- PIF file contains pin information of top netlist
- ILOC file contains mapping information between BS cells and external ports
- BS order file contain information about boundary scan chain ONLY

For more detail of each type, please refer following explanations

3.1 Log files

After each SCAAP run, there are 4 kinds of log files: error file, info file, smry file, log file.

Below

3.1.1 SMRY file

This file contains the summaries information of a SCAAP run: execution environment, execution result, warning and error.

```

Scan Circuit Automatic Addition Program / "SCAAP" / Version( 05-04-01 ) / Fri
Feb 27 11:13:59 AM ICT 2009

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%%%      FILE INPUT & CHECK      MESSAGES      %%%
%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
// Reading and checking SCL libraries
Scanning library directory "../scl_lib"
[1] SCL Library is opened. (../scl_lib/BODY.scl)
...
[17] SCL Library is opened. (../scl_lib/IOBUFF_KARI_RC04APAIW93.scl)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%%%      TARGET PRODUCT INFORMATION      %%%
%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
// Information about chip top
TARGET BLOCK NAME           : ""
GENERATION TARGET TYPE      : LSI
OUTPUT PACKAGE PIN ATTRIBUTE : USER
PACKAGE PIN SPEC MODIFY     : USER
BusPort Creation            : Create

```

```
PGT BUS MODE           : YES
PGT USE TPH NO        : 0
GENERATION SET         : BS
GENERATION MODE        : BOUNDARY SCAN(TAP)
                      : BOUNDARY SCAN(BSR)
BS REGISTER ADD ATTRIBUTE : USER
CONTROL BS REGISTER BIND : NOBIND
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%%%      AVAILABLE PARAMETERS LISTING      %%%
%%%                                         %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

// List of execution log files

```
LISTING FILE NAME
SUMMARY      "./log.scaap/BS1.smry"
INFORMATION  "./log.scaap/BS1.info"
ERROR        "./log.scaap/BS1.error"
```

```
INFORMATION LIST
NO BLOCK STRUCTURE
TEST PIN ASSIGNMENT
```

// Information about netlist, control files and library files

```
VERILOG NETLIST AND LIBRARY
NETLIST      "./Net/TOP_net.v"
LIBRARY FILE  ...
LIBRARY EXT  ".v"
              ".vpx"
OUTPUT NETLIST  "./Net/TOP_net_BS.v"

SCL
SCL DIR       "../scl_lib"
SCL EXT       ".scl"
              ".scan_scl"
              ".TOP_scl"

CTLF
CTLF FILE     "CTLF/TOP_net_BS.ctlf"
PM
PM FILE       "./csv/PM_table_SCAAP_v4_nrm.csv"
TCF
NO TCF OUTPUT
PGT
NO PGT OUTPUT
BSDL
OUTPUT BSDL   "Intermediate/TOP_net_BS.bsd1"
BSDL VERSION  "STD_1149_1_1994"
SPF
NO SPF OUTPUT
FANDIV
NO CONTROL FILE
DFT-Profile
CONTROL FILE  "Intermediate/TOP_net_BS.dpf"
MINORI-Control
CONTROL FILE  "Intermediate/TOP_net_BS.mc"
Instance Location
CONTROL FILE  "Intermediate"
Pin Information
CONTROL FILE  "Intermediate/TOP_net_BS.pif"
STA Path Information
CONTROL FILE  "Intermediate/TOP_net_BS.path1"
PREFIX STRINGS
```

```
COMMON      NAME "SCAAP_BS_"
PIN          NAME "SCAAP_BS_"
BLOCK DEF   NAME "SCAAP_BS_"
BLOCK REF   NAME "SCAAP_BS_"
TERMINAL    NAME "SCAAP_BS_"
SIGNAL      NAME "SCAAP_BS_"
```

POW/GND SIGNAL NAME

NOTHING

ENABLE ADD BUFFER

NOTHING

```
%%%%%%%%%%
%%%
%%%      EXECUTION    MONITORING    MESSAGES    %%%
%%%
%%%%%%%%%%
```

// Information about execution flow and execution results

```
.... "NETLIST" WAS UPDATED SUCCESSFULLY.
.... "BSDL"    WAS GENERATED SUCCESSFULLY.
.... "PIF"     WAS GENERATED SUCCESSFULLY.
.... "Ins-Loc" WAS GENERATED SUCCESSFULLY.
.... "DFTPROF" WAS GENERATED SUCCESSFULLY.
.... "MNRCNTL" WAS GENERATED SUCCESSFULLY.
.... "BSRSEQ"  WAS GENERATED SUCCESSFULLY.
.... "STAPATH" WAS GENERATED SUCCESSFULLY.
```

```
%%%%%%%%%%
%%%
%%%      EXECUTION      SUMMARY      %%%
%%%
%%%%%%%%%%
```

// Summarized information

<<< ERROR COUNT SUMMARY >>>

			WARNING	ERROR	UNRECOVERABLE-ERROR
1) INPUT	PROCESS	:	0	0	0
2) EDIT	PROCESS	:	5	0	0
3) CHECK	PROCESS	:	0	0	0
4) PRE	PROCESS	:	0	0	0
5) GENERATE	PROCESS	:	0	0	0
6) OUTPUT	PROCESS	:	0	0	0
7) COMMON & OTHERWISE		:	0	0	0

TOTAL	COUNT	:	5	0	0

// Information about CPU time, memory used

<<< EXECUTION PERFORMANCE >>>

1) TOTAL	CPU	TIME	:	3.563	MIN
2) TOTAL	USE	TIME	:	3.608	MIN
3) TOTAL	MEMORY	SIZE	:	1.292	GB

// Final execution result

```
%%%%%%%%%%
%%%
%%%      EXECUTION COMPLETED.      // // // %%%
%%%      COMPLETE CODE = 1      | - - | %%%
%%%      ^u^      | ^u^ | %%%
%%%      THERE ARE WARNING MESSAGES.      | - | %%%
%%%      ... PLEASE CONFIRM THE MESSAGES.      --- %%%
%%%
```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

There are 4 codes describe the final execution result.

Code: 0

Test logic addition by SCAAP has been completed successfully.

```
> %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
> %%%
> %%% EXECUTION COMPLETED.      //// %%%
> %%%          COMPLETE CODE = 0  |- -| %%%
> %%%                                | ^u^| %%%
> %%% NO MESSAGE EXISTS.  ALL RIGHT.  | 0 | %%%
> %%%          YEAH!! V__  --- %%%
> %%%
> %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Code: 1

Test logic addition by SCAAP has been completed.
There are warnings and output files.
-> Please check an execution log to see if the test logic addition has generated any error.
If it has, correct the error and execute again.

```
> %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
> %%%
> %%% EXECUTION COMPLETED.      //// %%%
> %%%          COMPLETE CODE = 1  |- -| %%%
> %%%                                | ^u^| %%%
> %%% THERE ARE WARNING MESSAGES.  | - | %%%
> %%%          ... PLEASE CONFIRM THE MESSAGES.  --- %%%
> %%%
> %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Code: 2

Test logic addition by SCAAP has not been completed.
Error(s) are detected. (DATA ERROR)
-> Check the errors in an execution log.
Correct the errors, and execute again.

```
> %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
> %%%
> %%% EXECUTION COMPLETED.      //// %%%
> %%%          COMPLETE CODE = 2  |- -| %%%
> %%%                                | ^u^| %%%
> %%% ERROR OCCURRED.           | ^ | %%%
> %%%          ... PLEASE CORRECT THE ERRORS,  --- %%%
> %%%          THEN RETRY.      %%%
> %%%
> %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Code: 3

Test logic addition by SCAAP has not been completed.
Error(s) are detected. (CONTINUATION ERROR)
-> Check the errors in an execution log.
If the cause of the errors is not found, please contact out support center.

```
> %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
> %%%
> %%% EXECUTION INCOMPLETED.    //// %%%
> %%%          COMPLETE CODE = 3  |- -| %%%
> %%%                                | *U*| %%%
> %%% UNRECOVERABLE ERROR OCCURRED.  | ^ | %%%
> %%%          ... PLEASE CORRECT THE ERRORS,  --- %%%
> %%%          THEN RETRY.      %%%
> %%%
> %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```


3.1.2 ERROR file

This file contains all warnings and errors occurred during SCAAP execution period.

Below table error message list:

Error code	Error message	Descriptions	Measures
spre4103-w	Boundary Scan CLOCK function is specified at Package Pin(#number), but not INPUT pin.	A pin with a package number (#number) is not an input attribute, but a hanging style BS cell is specified for the pin.	Check the addition style of the BS cell.
sedt5103-w	Package Pin signal is not connected to I/O Buffer.	An external pin is not connected to an IO cell.	Check to see if the IO cell is registered in IOBUFF.scl in the SCL library.
argr0002-e	Command(#name) is invalid.	An argument (#name) is not valid.	Check the scaap execution shell.
argr0003-e	Parameter(#param) is invalid at command(#name).	A parameter (#param) for an argument (#name) is not valid.	Check the scaap execution shell.
scl0004-e	<SCL> is inapplicable to SERIES (#name).	A cell series (#name) is not valid.	Check cell series registered in the SCL library and the control file.
scl3010-e	Terminal Name(#name) specified in *TERMPAIRBEGIN of I/O Buffer(#cell) is invalid.	The terminal (#name) of an IO cell (#cell) is not valid.	Check any terminal of #cell registered in IOBUFF.scl in the SCL library, and check any terminal of #cell registered in the Verilog library.
sout1003-e	Module (#name) is not defined in verilog libraries.	A module (#name) is not in the verilog library.	Check to see if the module exists in the verilog library, or check whether its real cell specified in CELLDEF.scl is valid.
logi2003-u	Illegal record "#name" exists in the Control file.	An illegal record (#name) exists in the control file.	Check the control file.
logi2004-u	Illegal record "#name" exists in the BOUNDARY_SCAN record.	An illegal record (#name) exists in the BOUNDARY_SCAN keyword within the control file.	Check the record of the BOUNDARY_SCAN keyword in the control file.
spre4002-u	Boundary Scan Register add circuit for Function(#type) /Polarity(#pol) is missing in	Buffer terminal types (#type/#pol) for the addition of a boundary scan register are not valid.	Check any definition for BS register addition condition in BODY.scl, and any definition

	<SCL>.		for buffer terminal type in IOBUFF.scl.
sngen4201-u	GLOBAL_SIGNAL (#name) is connected with no OUTPUT TERMINAL.	An addition signal (#name) has an input terminal, but not an output terminal.	Check to see if an output terminal is registered in the SCL library or the control file.
sngen4202-u	GLOBAL_SIGNAL (#name) is connected with too many OUTPUT TERMINAL.	An addition signal (#name) has multiple output terminals.	Check to see if multiple output terminals are registered in the SCL library or the control file.
sout1010-u	Verilog parser function (@) failure.	The output of a net list fails.	Check any cell name or terminal name in CELLDEF.scl of the SCL library and the Verilog library.

NOTE:

If there are errors in reading netlist, messages will not be output into error/smry files. Instead, they are output detail into log file. If the following messages appears in error file, please check detailed error messages in the log file.

The screenshot shows the output of the SCAAP program. Annotations include:

- SCAAP program version**: Points to "Version(05-03-00)" in the header.
- Execution date**: Points to "Tue Dec 19 19:13:56 PM JST 2007" in the header.
- The contents of warnings and errors**: A bracket on the left side of the message block.
- Keywords**: Points to the error code "logi1006-u" in the message "logi1006-u (si4scl) SCL reader abnormal end. Return code is 2."

The output text is as follows:

```

Scan Circuit Automatic Addition Program / "SCAAP" / Version( 05-03-00 ) / Tue Dec 19 19:13:56 PM JST 2007

sedt5103-w (se51upn) Package Pin signal is not connected to I/O Buffer.
sedt5000-i (se51upn) Check the Package Pin (244 / audata2).
sedt5112-w (se511bf) Package Pin signal connected Cell is not I/O Buffer.
sedt5000-i (se511bf) Check the Package Pin (245 / audata3).
sedt5103-w (se51upn) Package Pin signal is not connected to I/O Buffer.
sedt5000-i (se51upn) Check the Package Pin (245 / audata3).
spre4103-w (sp41chk) Boundary Scan CLOCK function is specified at Package Pin(148), but not INPUT pin
:
:
:
:
:
logi5004-w (si5prod) Block Diagnosis information is missing.
logi5012-w (si5prod) Block Test Method name is not specified.
scl0004-e (si42chk) <SCL> is inapplicable to SERIES (HG77).
logi1006-u (si4scl) SCL reader abnormal end. Return code is 2.
:
:
:
:

```

Keywords

- w: The contents of a warning
- e: The contents of an error
- u: The contents of a SEVERE error
- i: Information related to the warning or error above.

3.1.3 INFO file

This file contains information about any BS test, test pin and module extraction.

Example:

```
Scan Circuit Automatic Addition Program / "SCAAP" / Version( 05-04-01 ) / Fri
Feb 27 10:11:33 AM ICT 2009
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                               %%%
%%%   EXECUTION   INFORMATION   %%%
%%%                               %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

// Information about chip top

```
<<<  TARGET PRODUCT INFORMATION  >>>
```

```

TARGET BLOCK NAME           : "TOP_net"
GENERATION TARGET TYPE      : LSI

```

```
<<<  DIAGNOSIS   INFORMATION   >>>
```

```

1) AUTO_DIAGNOSIS INFORMATION.
   ##### NOT AVAILABLE #####

```

```

2) BLOCK TEST   INFORMATION.
   ##### NOT AVAILABLE #####

```

```

3) BOUNDARY SCAN INFORMATION.
   3.1) APPLIED METHOD           : "STD_1149_1_1993" ← JTAG info
                                   : BS TAP
                                   : BS REGISTER

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                               %%%
%%%   DIAGNOSIS PIN ASSIGNMENT LIST   %%%
%%%                               %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```
<<<  ASSIGNMENT TO PACKAGE PIN  >>>
```

DIAG PIN NAME	DIAG PIN KIND	ASSIGN TYPE	PKG PIN#	I/O/B CHANGE	PRE I/O BUFF	POST I/O BUFF	PACKAGE PIN NAME	PIN BIT
TMS	BS	BAND	185	B->B	ZW9T3BA2 234WQCZC	ZW9T3BA2 234WQCZC	CXTMS	---
TRST	BS	BAND	186	B->B	ZW9T3BA2 234WQCZC	ZW9T3BA2 234WQCZC	CXTRST_N	---
TCK	BS	BAND	180	B->B	ZW9T3BA2 234WQCZC	ZW9T3BA2 234WQCZC	CXTCK	---
TDO	BS	BAND	182	B->B	ZW9T3BA0 234WQCZC	ZW9T3BA0 234WQCZC	CXTDO	---
TDI	BS	BAND	181	B->B	ZW9T3BA2 234WQCZC	ZW9T3BA2 234WQCZC	CXTDI	---

```
<<<  ASSIGNMENT TO BLOCK TERMINAL  >>>

#####  NOTHING  #####
```

Test pin allocation information

- DIAG PIN NAME : It is the name of a test pin. This name is defined at column #TESTPIN in PM file.
- DIAG PIN KIND: It is test type of test pin.
Ex: BS → pin used in boundary scan circuit
- ASSIGN TYPE: It is an attribute for an IO cell that the test pin is allocated. This attribute is defined in SCL library.
- PKG PIN#: It is package pin number. This number is defined at column #IL in PM file.
- I/O/B CHANGE: It is information of terminal attributes before and after addition.
NOTE: IO attributes are changed only if ab IO buffer is replaced
- PRE I/O BUF: name of IO cell before addition
- POST I/O BUFF: name of IO cell after addition
- PACKAGE PIN NAME: It is the name of external pins. This name is defined at column #PIN in PM file.
- PIN BIT: It is bus bit number for the external pin name

3.1.4 LOG file

This file contains messages which are recorded when SCAAP is executed.

Example:

```

Scan Circuit Automatic Addition Program / "SCAAP" / Version( 05-03-00 ) / Tue Dec 19 19:13:56 PM JST 2007

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%% EXECUTION MONITORING MESSAGES %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Reading source file "/CHIPA_mt_mb_sc.v"
Scanning library directory "/Net/Net_div"
Scanning library file "/ssv/xxx/dft/...../Lib/Verilog/65LPZK3PLL/TC65APAE.vi"
Scanning library file "/ssv/xxx/dft/...../Lib/Verilog/TC65APAE/TC65APAE.vi"
Scanning library file "/ssv/xxx/FAB81Z/...../libraries/verilog/651pt0610120a/651pt0610120a.vi"
Scanning library file "/ssv/xxx/FAB81Z/...../libraries/verilog/tcn651pspramor/tcn651pspramor.vi"
Scanning library file "/ssv/xxx/dft/...../Lib/Verilog/tobn651p/tobn651p.vi"
Scanning library file "/ssv/xxx/dft/...../Lib/Verilog/tobn651p_c060712/tobn651p_c060712.vi"
Scanning library file "/ssv/xxx/dft/...../Lib/Verilog/tef651p256x1s_i_100b.v"
Scanning library file "/ssv/xxx/dft/...../Lib/Verilog/tef651psed_100a.v"

=====> Pin Multi Table Reader of PLASMA (v01r0200/2006.09.05) is invoked.

... "INPUT" PROCESS WAS COMPLETED. COMPLETE CODE = 0
... "EDIT" PROCESS WAS COMPLETED. COMPLETE CODE = 0
... "PRE" PROCESS WAS COMPLETED. COMPLETE CODE = 0
... "GENERATE" PROCESS WAS COMPLETED. COMPLETE CODE = 0
... "OUTPUT" PROCESS WAS COMPLETED. COMPLETE CODE = 0

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%% EXECUTION COMPLETED. %%%%%%%%%%
%%%%%%%% COMPLETE CODE = 0 %%%%%%%%%%
%%%%%%%% NO MESSAGE EXISTS. ALL RIGHT. %%%%%%%%%%
%%%%%%%% YEAH!! V__ %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

Annotations in the image:

- Execution messages:** Points to the "EXECUTION MONITORING MESSAGES" section.
- Net list name (read):** Points to the "Reading source file" line.
- Verilog library information (read):** Points to the "Scanning library file" lines.
- Code information for each process:** Points to the process completion table.

3.2 BSDL file

BSDL file is automatically generated by SCAAP in text format describing the attributes of BS circuit. This file is used to create BS logic verification patterns by IOWRAPGEN tool. Also. It is used as a base for BSDL to submit to customers.

Below table summarizes syntax of BSDL file:

Syntax name	Syntax	remarks
Generic parameter	generic(PHYSICAL_PIN_MAP:string)	Skips reading
Logical port description	port(<pin spec>)	Only bit is enabled. Creates table. Skips the port "Linkage" where only pin number is defined in the port name. NOTE: DOES NOT support the bus notation
Standard use statement	Use STD_1149_1_??? .all	Only 1990, 10994 are enabled
Component conformance statement	Attribute COMPONENT_CONFORMANCE of ...	Skips reading. NOT required in 1990
Device package pin mappings	Attribute PIN_MAP of ...; constant <pin mapping name>: ...	Skip reading
Grouped port identification	Attribute PORT_GROUPING of ...	Imports differential motion input/output information and positive/negative port names of differential motion pairs.
Scan port identification	Attribute TAP_SCAN_??? of <port_ID> ...	Creates table. In case of TAP, only RESET is "L" active, others are "H" active.
Compliance enable description	Attribute COMPLIANCE_PATTERNS of ...	Defines the specified port in <pin type> = "in" and checks that BS_cell is not added. Skip reading of fixed information. When fixed input is necessary, the fixed value is described in attributes of pin information file (.pif)
Instruction register description	Attribute INSTRUCTION_LENGTH of ...	LENGTH value extraction
	Attribute INSTRUCTION_OPCODE of ...	Extracts the values of BYPASS, SAMPLE, EXTEST, IDCODE, CLAMP, HIGHZ, and INSTRUCTION_CAPTURE. When 2 or more OPCODE are defined, the 1 st value is taken. However, only BYPASS extracts the entire OPCODE. It checks whether length corresponds with LENGTH.

Optional register description	Attribute IDCODE_REGISTER of ...	IDCODE value extraction
Register access description	Attribute REGISTER_ACCESS of ...	Skip reading
Boundary-scan register description	Attribute BOUNDARY_CELLS of ...	Only 1990 is enabled. Skips reading contents.
	Attribute BOUNDARY_LENGTH of ...	LENGTH value extraction
	Attribute BOUNDARY_REGISTER of ... “ <bsorder_index> (<BS_cell_type>, <pin spec name>, <function>, <value>,...) “ &	Creates table. Checks whether the cell total corresponds with LENGTH.

Below table describes the type name of BS cell used in BSDL file

Pin type	BS_cell_type	Function
In	BC_1/BC_2/BC_4	INPUT CLOCK OBSERVE_ONLY
Out	BC_1/BC_2	OUTPUT2 OUTPUT3
Inout	BC_1/BC_2	CONTROL
	BC_1/BC_2/BC_4	INPUT OBSERVE_ONLY
	BC_1/BC_2	OUTPUT2 (2state output) OUTPUT3 (3state output)

Note:

OUTPUT2 is enabled only when CONTROL_BS_cell is available.

For the boundary register, BS_0 to BC_10 are available.

For the SCL released by System Setting and Development Section, only BC_1, BC_2, and BC_4 are provided. (BC_1, BC_2 are insertion type, BC_4 is observe-only type)

Example:

```
entity TOP is
# generic parameter
    generic(PHYSICAL_PIN_MAP:string:="DUMMY");
# logical port descriptions
    port(P002_CXMA_3:inout bit;
        P003_CXMA_2:inout bit;
        P006_CXMA_1:inout bit;
        ...
# BSDL standard information
    use STD_1149_1_1994.all;
# component conformance information
    attribute COMPONENT_CONFORMANCE of TOP:entity is "STD_1149_1_1993";
# device package port identifications
    attribute PIN_MAP of TOP:entity is PHYSICAL_PIN_MAP;
    constant DUMMY:PIN_MAP_STRING:=
        "P002_CXMA_3:67,"&
        "P003_CXMA_2:68,"&
        "P006_CXMA_1:69,"&
        "P007_CXMA_0:70,"&
        ...
# scan port identifications
    attribute TAP_SCAN_MODE of P185_CXTMS:sigal is TRUE;
    attribute TAP_SCAN_RESET of P186_CXTRST_N:sigal is TRUE;
    attribute TAP_SCAN_CLOCK of P180_CXTCK:sigal is (10.0e6,BOTH);
    attribute TAP_SCAN_OUT of P182_CXTDO:sigal is TRUE;
    attribute TAP_SCAN_IN of P181_CXTDI:sigal is TRUE;
# compliance enable description
    attribute COMPLIANCE_PATTERNS of TOP:entity is
        "(P157_CXMODE0, P158_CXMODE1, P159_CXMODE2, P163_CXXIN,
        P167_CXTEST, P178_CXASBRKAK_MX_N, P187_CXASEMD) (0111101)";
# instruction register description
    attribute INSTRUCTION_LENGTH of TOP:entity is 8;
    attribute INSTRUCTION_OPCODE of TOP:entity is
        "BYPASS (11111111),"&
        "EXTEST (00000000),"&
        "SAMPLE (00000001),"&
        "IDCODE (00000011),"&
        "EXMBIST (00000010),"&
```

```
"    11100000,"&
"    11100001,"&
"    11100010,"&
"    11000000,"&
"    11000001,"&
"    11010000,"&
"    11010001,"&
"    11010010);
```

attribute INSTRUCTION_CAPTURE of TOP:entity is "00000001";

attribute INSTRUCTION_PRIVATE of TOP:entity is "EXMBIST";

optional register description

attribute IDCODE_REGISTER of TOP:entity is

```
"0000"&
"1000000010011111"&
"01000100011"&
"1";
```

boundary scan register description

attribute BOUNDARY_LENGTH of TOP:entity is 529;

attribute BOUNDARY_REGISTER of TOP:entity is

```
"0 (BC_2,P174_CXTRST_MX_N,INPUT,X),"&
"1 (BC_1,*,CONTROL,0),"&
"2 (BC_1,P174_CXTRST_MX_N,OUTPUT3,X,1,0,Z),"&
"3 (BC_2,P173_CXTMS_MX,INPUT,X),"&
"4 (BC_1,*,CONTROL,0),"&
"5 (BC_1,P173_CXTMS_MX,OUTPUT3,X,4,0,Z),"&
"6 (BC_2,P172_CXTDO_MX,INPUT,X),"&
"7 (BC_1,*,CONTROL,0),"&
```

...

3.3 ILOC file

This is the mapping table for logic (full instance display) added by SCAAP and LSI pin.

Example:

<#PKG>	<#IL>	<#PIN>	<internal terminal >	<power area>	<full name of BS cell>
### Added circuits which correspond to LSI pins					
67	2	CXMA[3]	--	MAIN	SCAAP_BS_BSR0225_G
67	2	CXMA[3]	--	MAIN	SCAAP_BS_BSR0224_G
67	2	CXMA[3]	--	MAIN	SCAAP_BS_BSR0223_G
68	3	CXMA[2]	--	MAIN	SCAAP_BS_BSR0222_G

68	3	CXMA[2]	--	MAIN	SCAAP_BS_BSR0221_G
68	3	CXMA[2]	--	MAIN	SCAAP_BS_BSR0220_G
69	6	CXMA[1]	--	MAIN	SCAAP_BS_BSR0219_G
69	6	CXMA[1]	--	MAIN	SCAAP_BS_BSR0218_G
### Added circuits which do not correspond to any LSI pin					
--	--	--	--	MAIN	SCAAP_BS_iomask_mode_n_g
--	--	--	--	MAIN	SCAAP_BS_iomask_mode_p_g
--	--	--	--	MAIN	SCAAP_BS_BCB_G
...					

There are 6 columns in ILOC file:

- 1st column: It is a package pin number. Package number of a pm file: A package number set in #PKG.
- 2nd column: It is an inner lead (pad) number. Package number of a pm file: An inner lead number set in #IL.
- 3rd column: It is the name of an external pin. Package number of a pm file: The name of an external pin set in #PIN.
- 4th column: It is an internal terminal for the IO buffer added to a circuit.
- 5th column: It is information about power supply area.
- 6th column: It is a full instance display of added circuits. Display from low hierarchy level of the top module. A delimiter for the hierarchy level is a period “.”.

3.4 PIF file

This is a text file describing the row order of pins and the presence/absence of the probe contact and attributes of pins.

Syntax:

<num> <pin_type> <BSDL_pin_name> <WGL_pin_name> <contact> <attribute>

<num> : pin appearance number (package pin number)

This value describes numbers in ascending order, starting from 0, in the targeted device pin number order. Using “dummy” keyword when there is the pin number exist in BSDL file but the port is not connected to net.

<pin_type> :

It is one of values : in/out/inout/ nonbs_in/nonbs_out/nonbs_inout/power/ground.

“nonbs_” is defined by <logical port description> of BSDL file 'linkage'

<BSDL pin name> :

This value describes the port name defined in <logical port description> of BSDL file

<WGL_pin_name>:

This value describes the port name of Verilog top module that is output to WGL

<contact>: presence or absence of probe contact

This value describes 'yes' (contact present) and 'no' (contact absent).

NOTE: contact absent port is the pin targeted for IOWRAP

When there is no IO of contact absent port, iowrap.wgl is not generated.

<attribute>: pin attribute

This value describes fixed input "0/1" or PULL present "PULL_UP/PULL_DOWN".

Example:

#num	type	BSDL	WGL	cont	attribute
dummy					
0	inout	P002_CXMA_3	CXMA[3]	no	
1	inout	P003_CXMA_2	CXMA[2]	no	
dummy					
2	inout	P006_CXMA_1	CXMA[1]	no	
3	inout	P007_CXMA_0	CXMA[0]	no	
dummy					
4	inout	P009_CXMA_10	CXMA[10]	no	
5	inout	P010_CXBA_0	CXBA[0]	no	
dummy					
6	inout	P012_CXBA_1	CXBA[1]	no	
dummy					
7	inout	P014_CXDCS_N	CXDCS_N	no	
dummy					
8	inout	P016_CXRAS_N	CXRAS_N	no	

3.5 BSRO file (BS order file)

This contain BS cell list base on chain order.

This file will be released to Scan task if using BSFF as scan cell.

Syntax: <full instance name of BSFFs> <position in BS chain> <index of BS chain>

Ex:

<full instance name of BSFFs>	<position in BS chain>	<index of BS chain>
/u_pad/pad_io/SCAAP_BS_BSR0000_G/GFF	0	1
/u_pad/pad_io/SCAAP_BS_BSR0001_G/GFF	1	1
/u_pad/pad_io/SCAAP_BS_BSR0002_G/GFF	2	1
/u_pad/pad_io/SCAAP_BS_BSR0003_G/GFF	3	1
/u_pad/pad_io/SCAAP_BS_BSR0004_G/GFF	4	1
/u_pad/pad_io/SCAAP_BS_BSR0005_G/GFF	5	1
/u_pad/pad_io/SCAAP_BS_BSR0006_G/GFF	6	1
/u_pad/pad_io/SCAAP_BS_BSR0007_G/GFF	7	1

III.IOWRAPGEN

This part describes basic knowledge about IOWRAPGEN tool and how to use this tool for generating boundary scan test patterns.

Before go in detail explanation, there are some keywords need to be made sense. Please refer below part to have those keyword's definitions.

1. IOWRAPGEN Keywords

- **Contact/Non-contact pin**

Contact pins are target pins for BS register addition (it means that BS_cells will be added to their IO terminal). As normal, they are bi-directional signal pins for all inputs and outputs, which basically do not include power supply of device and ground.

Non-contact pins are non-target pins for BS register addition (it means that BS_cells will be NOT added to their IO terminal). As normal, they are external ports related to analog, XTAL, USB, ... modules or 5pins of JTAG (TDI, TDO, TCK, TMS, TRST).

==> To get information about contact and non-contact pin, refer to column #BSCAN (BS addition flag) in PM file.

Y: contact pin with control and observe cell added

C: contact pin with only observe cell added

N: non-contact pin

- **PROBE/NON PROBE pins**

Pins of chip are divided into 2 sets:

1 – PROBE PIN: they are pins which have define “Y” value at column #PROBE in PM file, usually contact pins related to normal modules and 5pins of JTAG module.

2- NON PROBE PIN: they are pins which have define “N” value at column #PROBE in PM file, usually non-contact pins (except VCC/GND) or contact pins related to RENESAS PRIVATE module (ex: EXMBIST). It also related to "TESTER Limitation".

- **Types of pins in boundary scan test mode**

There are 4 types of pins:

- PROBE CONTACT: Renesas and customer can test by boundary scan method, usually they are ports related non-Renesas-private modules.
- NON-PROBE CONTACT: just Renesas can test by boundary scan method, usually they are ports related to Renesas-private modules.
- PROBE NON-CONTACT: Renesas and customer can test by boundary scan method, usually they are test mode ports.
- NON-PROBE NON-CONTACT: those ports can be tested by boundary scan

method, they are power/GND, ports related to analog, XTAL, USB, ...

- **All pins/ Few pins test**

In “all pins” test, all PROBE pins are tested.

In “few pins” test, ONLY PROBE NON-CONTACT pins are tested.

- **Boundary scan operation test modes**

Below table explains detail about each boundary scan operation test mode. Totally there are 10 types of modes.

NO.	Mode	Standard	Instruction code	SCAAP support	Usage
1	BYPASS	Required	3bit: 111 4bit: 1111	support	The objective chip is bypassed on the boundary scan. When no IDCODE is found, the operation is moved to this state by resetting with TRST signal. It is the normal operation mode and the chip ports are used by User.
2	IDCODE	Option	3bit: 100 4bit: 0100	support	The value in IDCODE register is read. It is the normal operation mode and the chip ports are used by User.
3	SAMPLE/PRELOAD	Required	3bit: 001 4bit: 0001	support	In this mode, the logic value of the chip port can be observed (sample), and the initial value of EXTEST and CLAMP can be set (preloaded) with the boundary scan while operating the chip normally.
4	CLAMP	Option	4bit: 0110	support	The chip is bypassed on the boundary scan but the chip ports output the logic value loaded with last SAMPLE/PRELOAD
5	HIGHZ	Option	4bit: 0111	support	Set all output pins of the chip to the HiZ state.
6	RUNBIST	Option	Optioned by designed	Not corresponded	BIST circuit in the chip starts for test
7	INTEST	Option	3bit: 010	support	The boundary scan controls all pins of the chip. When a series of patterns is provided, inside the chip can be tested by using the boundary scan.
8	EXTEST	Required	3bit: 000 4bit: 0000	support	Mode for the connection test between chips on a test for printed circuit board.

9	USERCODE	Option	Optioned by designed	Special correspondence	When chip corresponds to the user program, the ID code programmed by the user is read.
10	User setting code	Option	Optioned by designed	Not corresponded	The instruction defined by the user is used.

NOTE:

- The instruction code has 3 or 4 bit because the logic configuration is switched depending on the selected mode counts.
- Instruction of the TAP controller dedicated to SCAAP.
- Instruction register has 8bit

2. Types Of Boundary Scan Test Patterns

There are 6 types of Boundary scan test patterns. This part explains introduces detail about each type of patterns. With those information

2.1 BS Function confirmation test

This kind of test checks boundary scan information such as IDCODE instruction, BYPASS instruction and performs some sample instruction test at input direction.

1– IDCODE read test (IDCODE instruction + BYPASS instruction)

This test reads the 32bit data in the IDCODE register of TAP controller and checks whether it conforms with the IDCODE described in BSDL or not.

The execution flow is:

- After the initialization by TEST_LOGIC_RESET state, defining the IDCODE instruction in the instruction register
- 32 bit data in IDCODE register is serially read out from TDO and compared with information in BSDL file.

NOTE:

When there is no IDCODE (instruction), BYPASS register test is executed after initialization by TEST_LOGIC_RESET state.

2 – BYPASS register test (BYPASS instruction)

This is an I/O test of 1bit register between TDI-TDO.

The execution flow is:

- Defining BYPASS instruction in instruction register
- 4bit data is serially input to the BYPASS register from TDI, check whether this data can be read serially from TDO or not.

NOTE:

This test is executed for all OPCODE of the BYPASS instruction described in BSDL.

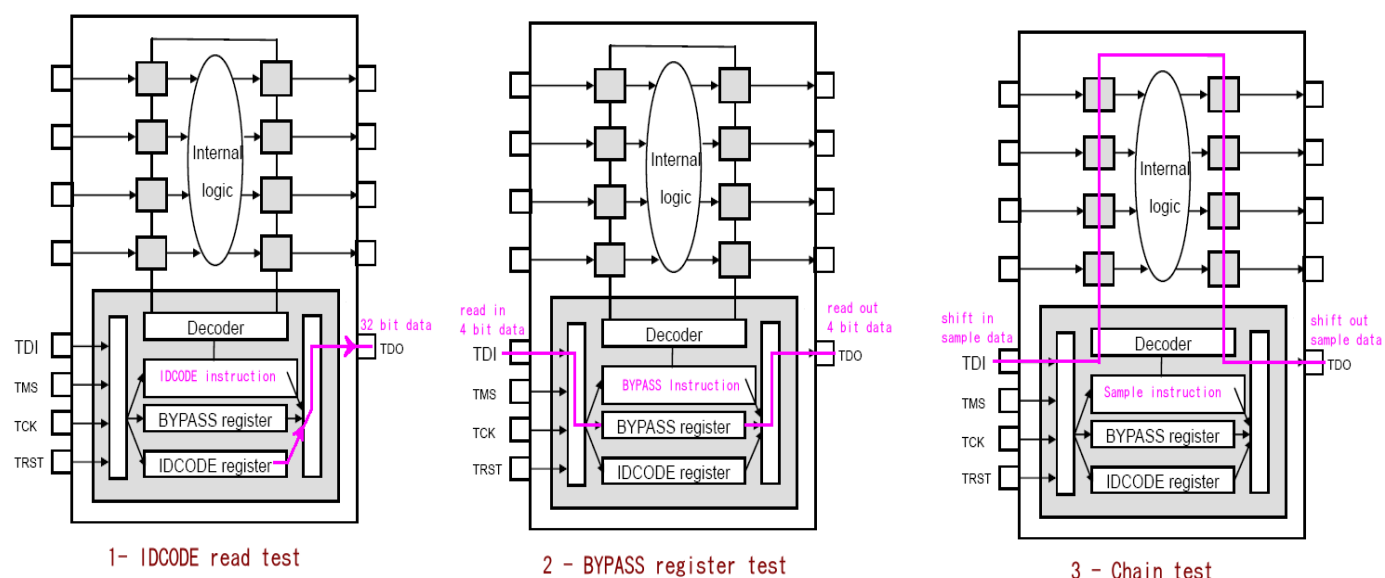
In IOWrapgen, I/O test of instruction register and INSTRUCTION_CAPTURE read test are also supported in the BYPASS register test.

3 – Chain test (SAMPLE instruction)

This is a connection confirmation test of boundary scan chain.

The execution flow is:

- Defining SAMPLE instruction in instruction register
- Data is serially input in all boundary scan registers from TDI, and check whether this data can be read serially from TDO or not.



4 – Sample instruction test (SAMPLE instruction)

This test is intended for input ports where the INPUT_BS_cell has been added.

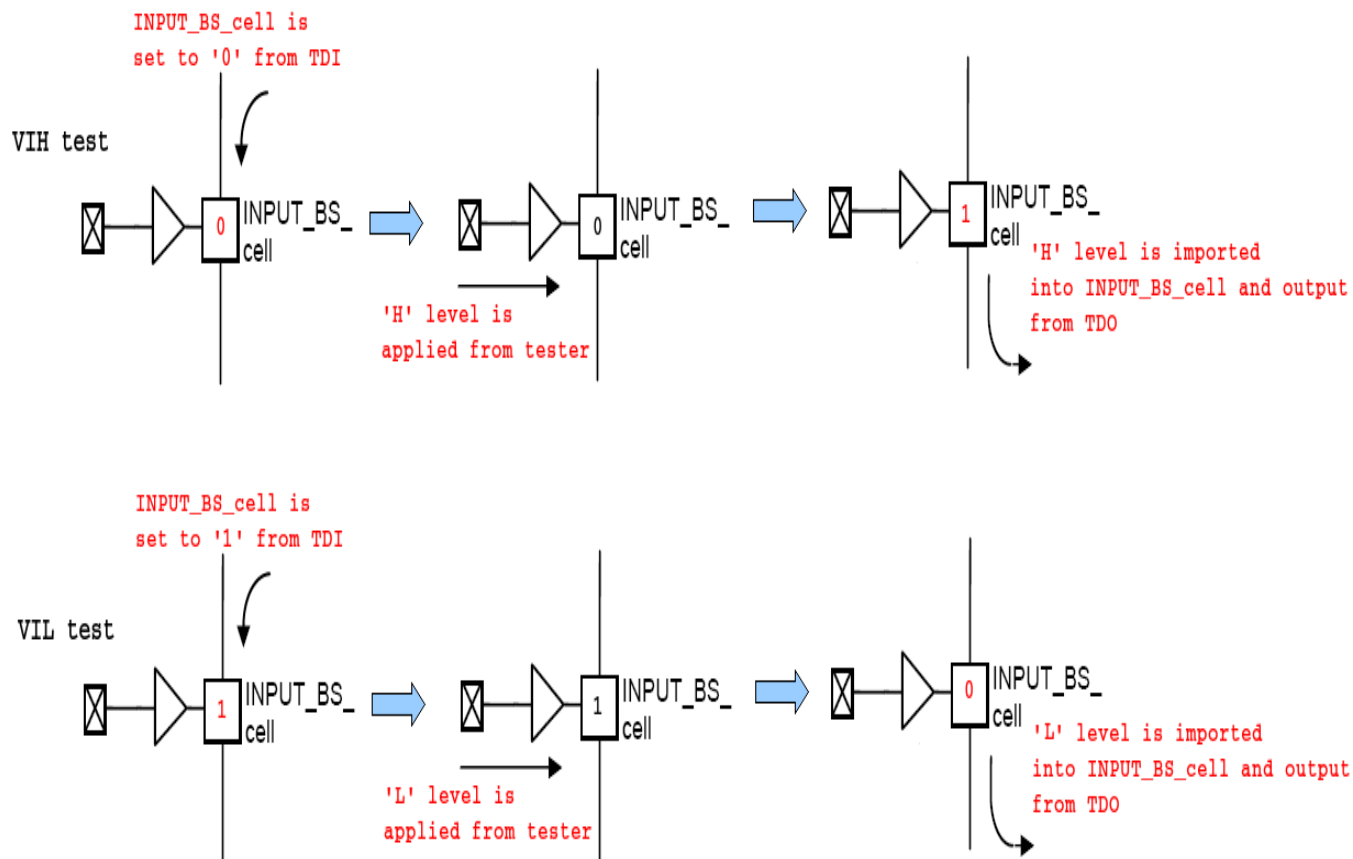
The execution flow is:

- Defining SAMPLE instruction test in the instruction register
- Data “H” or “L” is input from tester through those ports and imported into the INPUT_BS_cell
- Then data is shifted through boundary scan chain to TDO and check whether external input data is properly imported and serially output to TDO or not.

NOTE:

This test just forces to input port ONLY. If there is no input port which has BS_cell added to IO

terminal, this test is omitted.



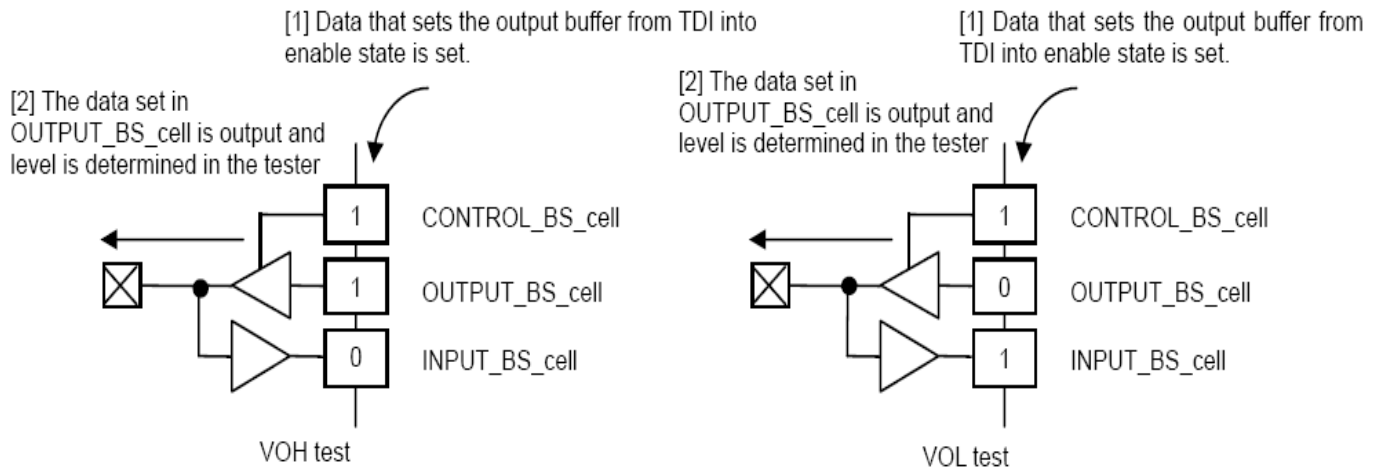
2.2 CLAMP instruction test

This kind of test performs some sample instruction test at output direction.

It is intended for the contact pins where `OUTPUT_BS_cell` has been added (including output and inout port).

The execution flow is:

- Defining CLAMP instruction in the instruction register
- All `CONTROL_BS_cells` are set to output enable state
- Then data '1' or '0' is set to `OUTPUT_BS_cell` and be kept. Tester determines whether that clamp data is output correctly or not.



2.3 HighZ confirmation test

This test checks whether the level of I/O is in the vicinity of $\frac{1}{2}$ VCC or not. In reality, there are 2 cases can be met in an output port:

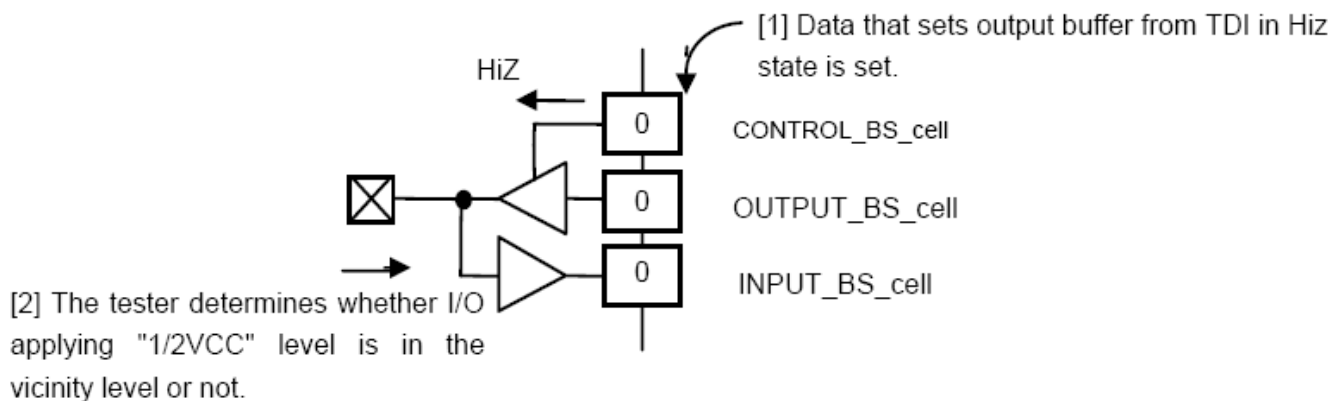
- Case 1: if there is output data, the value at that port is 0 or 1.
- Case 2: if that port is open, there are no value at that port.

To check the output control signal at output ports and bidirectional port in reality, there are 3 voltage levels are used:

- VCC: represents to 1 or PULLUP
- VDD: represents to 0 or PULLDOWN
- $\frac{1}{2}$ VCC: represents to Z value

There are 2 test items in this type of test:

- HIGHZ instruction confirmation test of TAP controller : checking the current status of output port and output directional of inout port. After defining HIGHZ instruction into the instruction register, $\frac{1}{2}$ VCC level is applied tester to output ports and inout ports, then checking at those ports the voltage levels are changed or not.
- HiZ confirmation test of ports : checking the status of output and inout port after applying data to output control signals. Next, define EXTEST instruction into the instruction register, data set to output buffer is input from TDI, then $\frac{1}{2}$ level is applied by output ports and input ports from tester. Tester checks the voltage value of those ports changed or not. If ports have PULL attributes, the PULLUP/DOWN confirmation test is executed.



2.4 DCP test

DCP test checks in/out direction of contact IO.

The execution flow is:

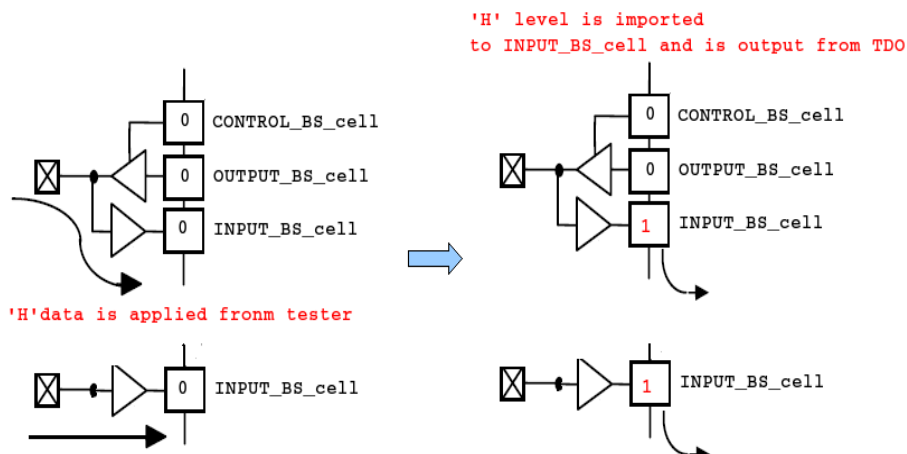
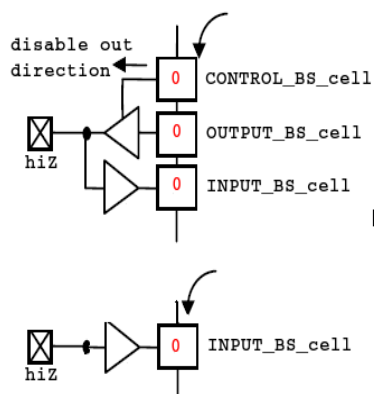
- VIH/L test:

This is in direction test of contact pin where INPUT_BS_cell has been added (including input ports and inout ports).

- Defining EXTEST instruction into the instruction register
- All CONTROL_BS_cells are set to enable input direction and other BS_cells are set to default value, 'H' or 'L' is input from tester in the input / inout contact ports
- Then that data is imported to INPUT_NS_cell, propagated through BS scan chain and tested in TDO.

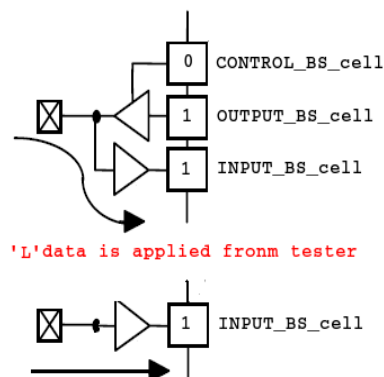
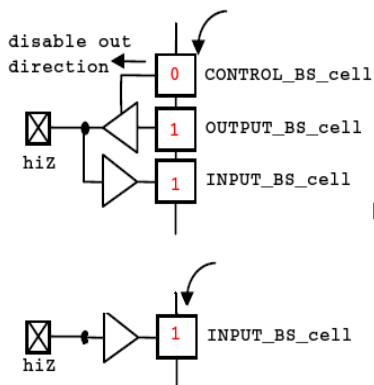
VIH test

All BS_cell is set to '0' from TDI

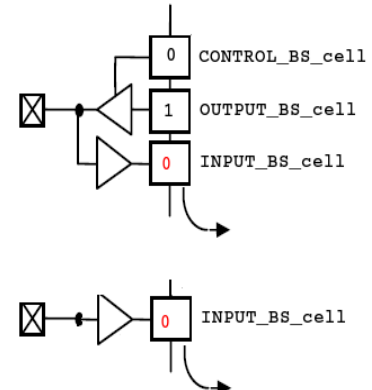


VIL test

All CONTROL_BS_cells are set to '0'
and other BS_cell are set to '1' from TDI



'L' level is imported
to INPUT_BS_cell and is output from TDO



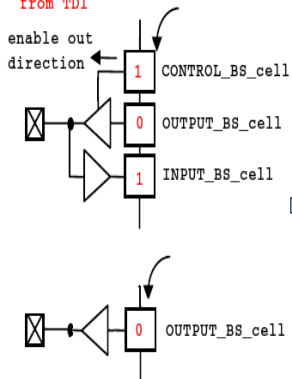
• VOH/L test:

This is out direction test of contact pin where OUTPUT_BS_cell has been added (including output ports and inout ports).

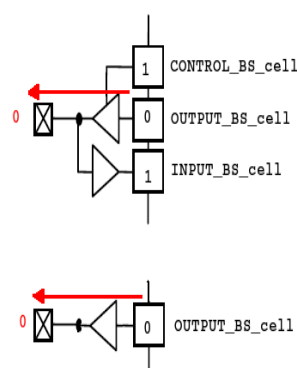
- Defining EXTEST instruction into the instruction register
- All CONTROL_BS_cells are set to enable out direction, '1' or '0' is set in OUTPUT_BS_cells and '0' or '1' is set in INPUT_BS_cells from TDI. Data from OUTPUT_BS_cell is externally output and tester determines whether the set data is correctly output or not.
- Then with inout port, output data is imported into INPUT_BS_cell by wraparound, propagated through BS scan chain and tested at TDO.

VOH test

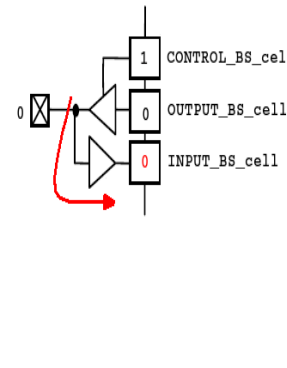
CONTROL_BS_cell is set to '1'
INPUT_BS_cell is set to '1'
OUTPUT_BS_cell is set to '0'
from TDI



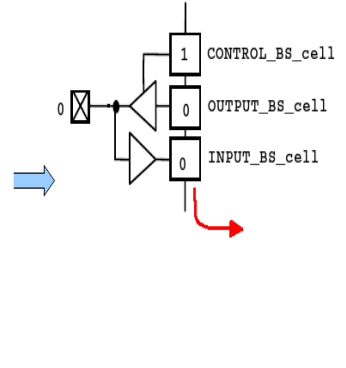
Data '0' from OUTPUT_BS_cell
is externally output
and tester determines whether
the set data is correctly output or not



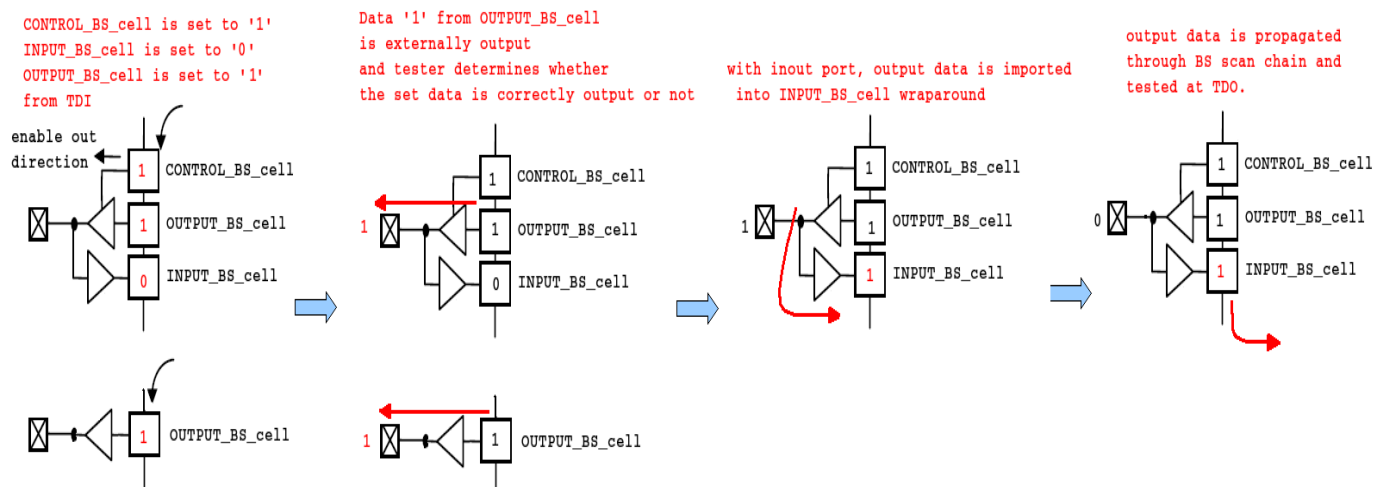
with inout port, output data is imported
into INPUT_BS_cell wraparound



output data is propagated
through BS scan chain and
tested at TDO.



VOH test



2.5 IOWRAP test

This type of test is intended to NON-PROBE CONTACT inout pins.

There are 3 test items in this type:

- Function test: checks stuck-at faults

After defining EXTEST instruction in the instruction register

(1) Data 0,1 are set inversely in OUTPUT_BS_cell, INPUT_BS_cell and control data is set into CONTROL_BS_cell of NON-PROBE CONTACT inout pins from TDI

(2) Import data set in OUTPUT_BS_cell into INPUT_BS_cell through output buffer → input buffer

(3) Determine whether the data of INPUT_BS_cell is correctly rewritten from the output value of TDO

- Leak test: checks leakage faults between the adjacent pins

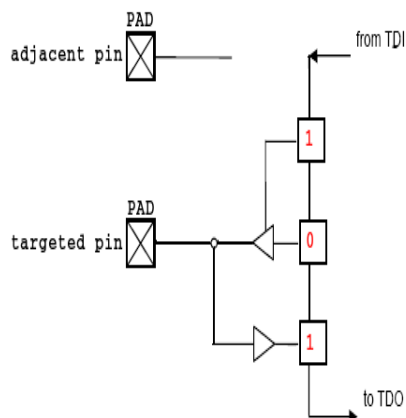
During testing, 0 or 1 are inversely set in OUTPUT_BS_cell of adjacent pins and pins targeted for leak test.

(4) Data 0,1 are set inversely in OUTPUT_BS_cell, INPUT_BS_cell and control data is set into CONTROL_BS_cell of NON-PROBE CONTACT inout pins from TDI

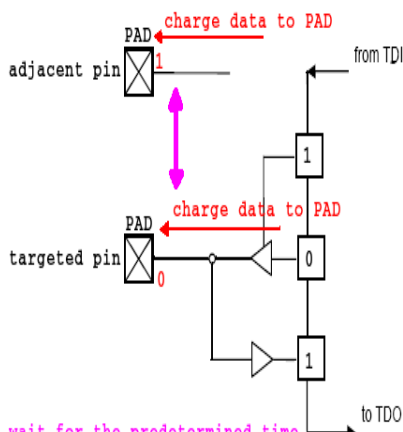
(5) After data in OUTPUT_BS_cell is charged to PAD, rewrite CONTROL_BS_cell data to disable out direction

(6) Wait for predetermined time to confirm the change in the charge, the Import data at PAD into INPUT_BS_cell, and determine it from the output value of TDO

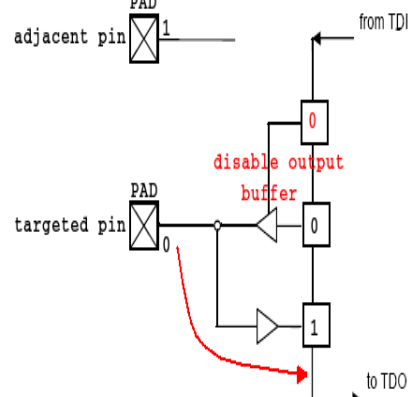
LEAK0 test



data 0, 1 are inversely set in OUTPUT_BS_cell of adjacent pins and targeted pins

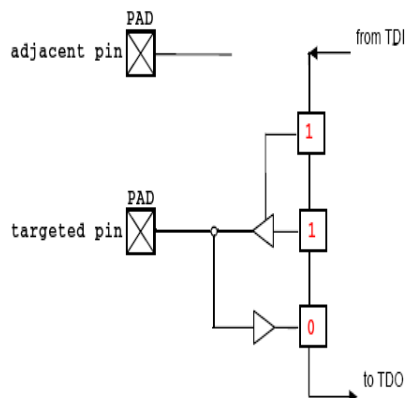


wait for the predetermined time to confirm the change in the charge. Maybe there is a leak between adjacent pins. It makes the charged data change value.

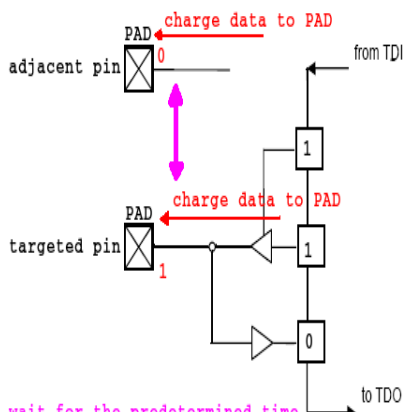


Import the charged data from PAD into INPUT_BS_cell, and determine it from the output value of TDO

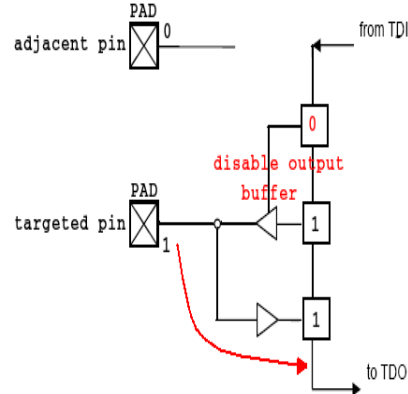
LEAK1 test



data 0, 1 are inversely set in OUTPUT_BS_cell of adjacent pins and targeted pins



wait for the predetermined time to confirm the change in the charge. Maybe there is a leak between adjacent pins. It makes the charged data change value.



Import the charged data from PAD into INPUT_BS_cell, and determine it from the output value of TDO

- PULLUP/PULLDOWN test: the execution flow is the same as leak test. But to specify this test, MUST define PULL information to target test pin in .pif file.

#	num	type	BSDL	WGL	cont	attribute
0		in	pin_in_0	in_0	yes	
1		in	pin_in_1	in_1	yes	0
2		in	pin_in_2	in_2	yes	1
dummy	←Use description of dummy when NC pin exists between num 2 and num 3 pins.					
3		out	pin_out_0	out_0	yes	
4		out	pin_out_1	out_1	yes	
5		out	pin_out_2	out_2	yes	
dummy						
6		inout	pin_io_0	io_0	yes	1
7		inout	pin_io_1	io_1	yes	
8		inout	pin_io_2	io_2	yes	
9		inout	pin_io_3	io_3	no	
10		inout	pin_io_4	io_4	no	
11		inout	pin_io_5	io_5	no	
dummy						
12		nonbs_in	pin_nbs_in_0	nbs_in_0	yes	
13		nonbs_in	pin_nbs_in_1	nbs_in_1	yes	0
14		nonbs_in	pin_nbs_in_2	nbs_in_2	yes	1
dummy						

PULL0
PULL1

define PULL information
for PULL testing in hiZ
and iowrap patterns

NOTE:

Leak test and PULLUP/DOWN confirmation test can not be executed simultaneously, separate pattern generation is required.

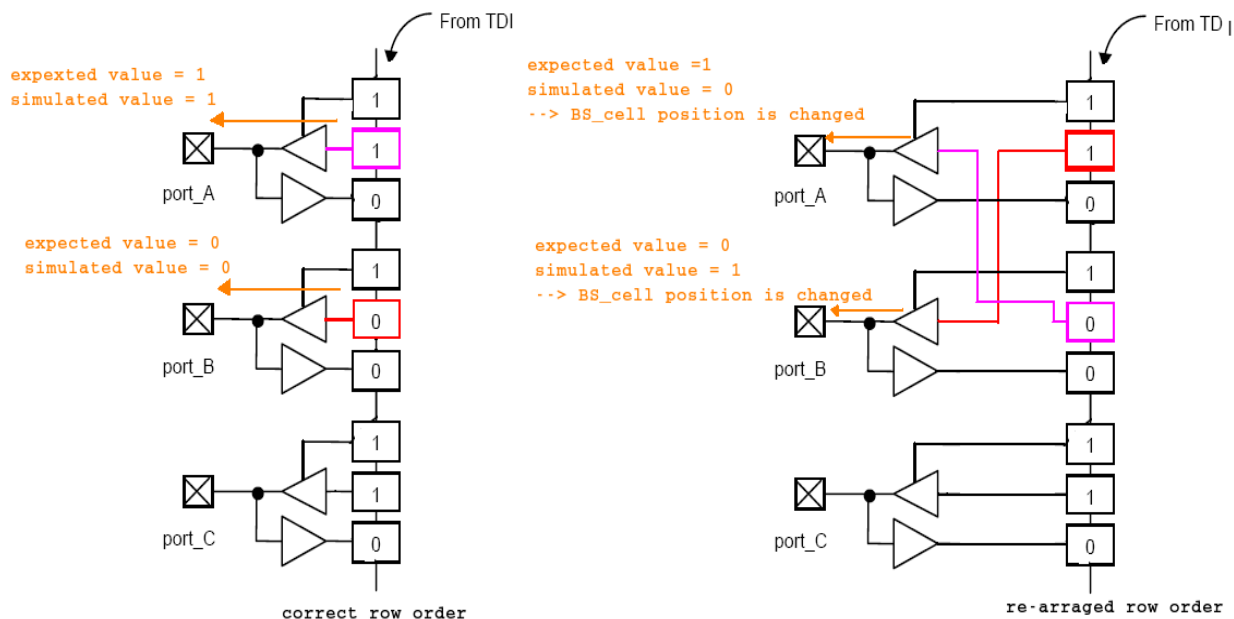
2.6 BS chain verification test

This test is executed only during verification, during mass production BS test and DCP test are applicable. So it has another name, that is DIAGNOSTIC test.

This test confirms there is no difference in the row order of the BS chain register of the targeted netlist and BSDL file.

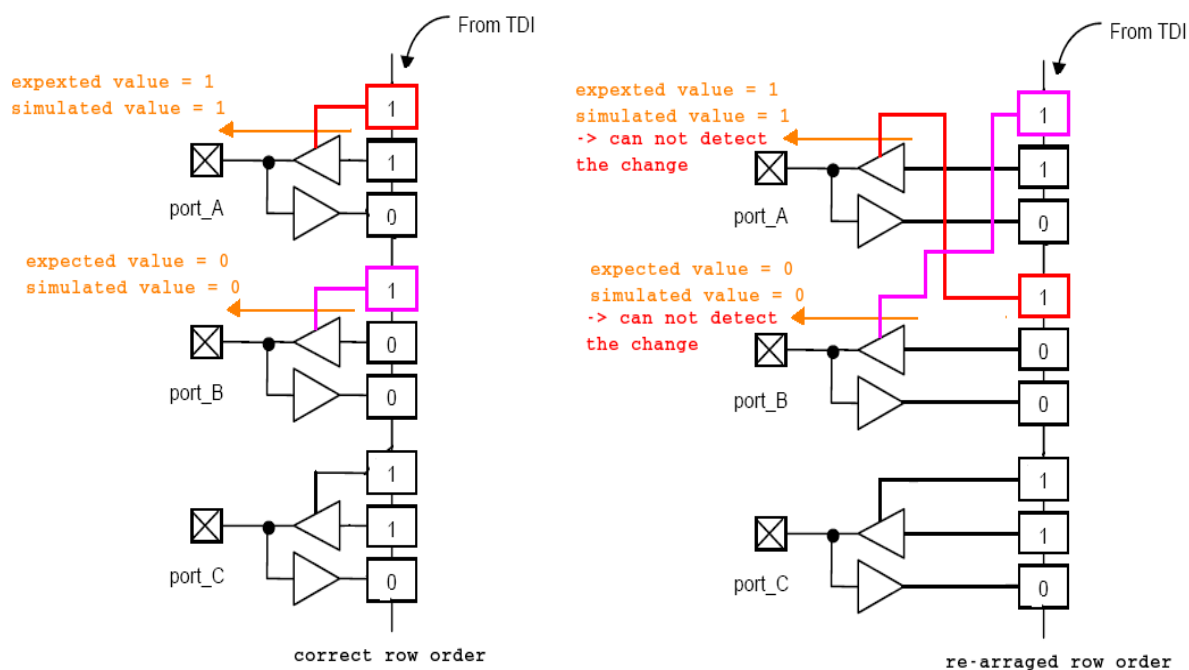
In some cases, the row order of BS register is re-arranged, DCP test can detect the change such as below example:

VOH/L test



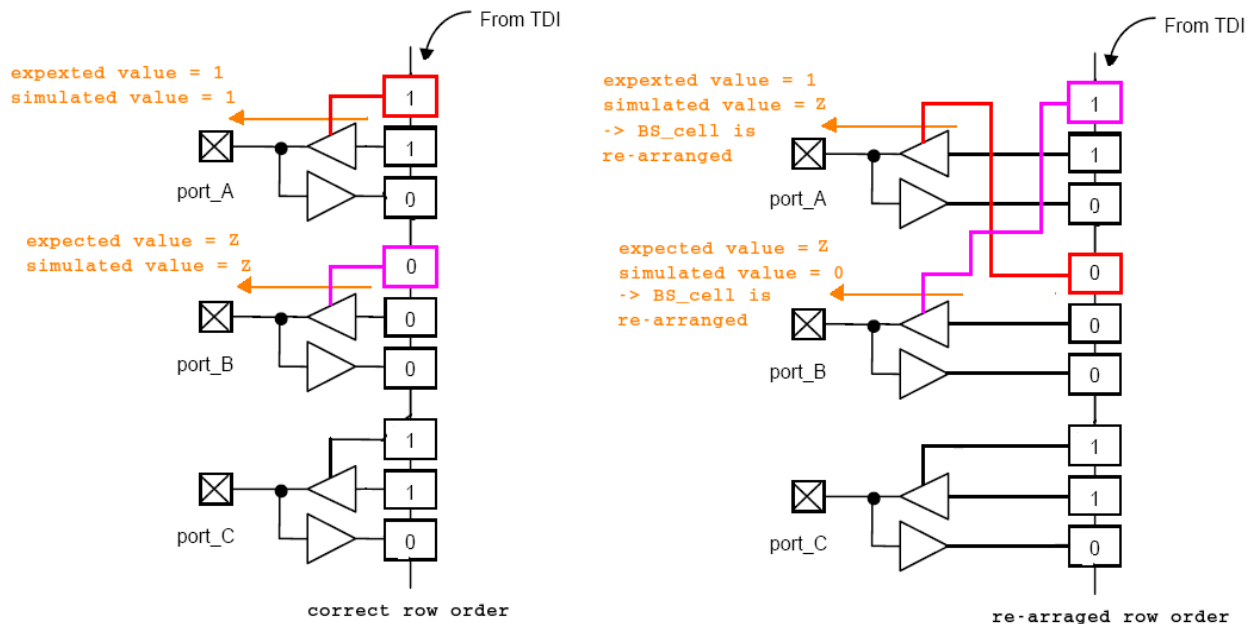
In some cases, even when the row order of BS register is re-arranged, DCP test can not detect the change such as below example:

VOH/L test



Because of the limit of DCP test, in DIAGNOSTIC test, it has not only VIH/L and VOH/L test but also has HiZ test to make sure that interchange of BS_cell can be test. For example, with above undetected case, DIAG test can detect:

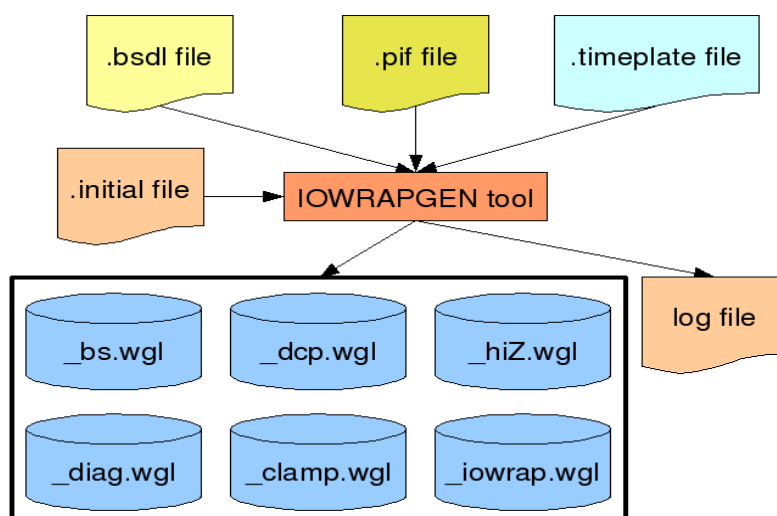
DIAGNOSTIC test



3. IOWRAPGEN Inputs/Outputs

Part "IOWRAPGEN" describes the total flow to generate boundary scan test patterns. This part helps to:

- Understand the format of input/output files
- How to prepare input files
- How to analyze output files



INPUTS:

- BSDL file: this file is an output of SCAAP. (refer section I.2.7 for basic format)
- Pin information file (.pif file): this file is an output of SCAAP. (refer section I.2.9)
- Initial pattern generation file (.initial file) contains timing information about patterns used to set up the chip to prepare for testing. This is optional input file.
- Timeplate file contains information about clock cycle and test timing.

OUTPUTS:

- Log file: shows each fault and the step that detects the fault. (refer section 8.2 of document I/O wrap pattern generation tool – IOWRAPGEN)
- `_bs.wgl` pattern:
if IDCODE instruction is not described in BSDL file, there is no IDCODE read test
if SAMPLE instruction test pattern is generated ONLY WHEN there is an input port where INPUT_BS_cell is added.
- `_dcp.wgl` pattern
- `_hiZ` pattern: is output only when 'yes' is specified in execution argument -hiz
If HIGHZ instruction is not described in BSDL file, HIGHZ confirmation test is not output.
- `_diag.wgl` pattern: is output ONLY when 'yes' is specified in the execution argument -diag
- `_clamp.wgl` pattern: is output ONLY when 'yes' is specified in the execution argument -clamp
- `_iowrap.wgl` pattern: is output ONLY when 'no' is specified in the contact of pin information file (.pif)

3.1 Timeplate file (.timeplate)

This file describes the clock cycle and the test timing. The format is as follow:

```

procedure timeplate {
  timescale ps | ns | us | ms | sec ;
  procedure runtest {      ← for TAP controller
    timeplate    <timeplate name > ;
    period       <clock cycle value > ;
    tck_up       < rising delay value of TCK pin > ;
    tck_down    < falling delay value of TCK pin > ;
    strobe_up    < rising delay value of TDO and output pin (including 2 way) > ;
    strobe_down < falling delay value of TDO and output pin (including 2 way) > ; ← not required
  }
}
for edge strobe
  
```



```

}
procedure clk ( <port_name> ) {   WGL_pin_name where waveform is to be defined → this
definition is valid ONLY in timeplate for TAP controller and the clock cycle exists in the timeplate
for TAP controller
    clk_up      <rising delay value of clock pin > ;
    clk_down    <falling delay value of clock pin > ;
}
procedure leak {                  ← for leak discharge wait
    timeplate   <timeplate name> ;
    period      <clock cycle> ;
    tck_up      < rising delay value of TCK pin > ;
    tck_down    < falling delay value of TCK pin > ;
    strobe_up   < rising delay value of TDO and output pin (including 2 way) > ;
    strobe_down < falling delay value of TDO and output pin (including 2 way) > ; ← not required
for edge strobe
}
}

```

By default, clock cycle of timeplate for leak discharge wait to 100times of timeplate for TAP controller, and adjust the appropriate values of each product in tester debugging.

Ex:

```

procedure timeplate {
    timescale ns ;
    procedure runtest {
        timeplate  RUNTEST ;
        period     100 ;
        tck_up     25 ;
        tck_down   75 ;
        strobe_up  10 ;
        strobe_down 15 ;
    }
    // procedure clk ( PortCLK ) {
    //   clk_up     25 ;
    //   clk_down   75 ;
    // }
    procedure leak {
        timeplate  LEAK ;
        period     100000 ;
    }
}

```

```
tck_up    25000 ;
tck_down  75000 ;
strobe_up 10000 ;
strobe_down 15000 ;
}
}
```

3.2 Initial pattern generation file (.initial)

This file is created and input by the user only when initial pattern generation is required. The format is as follow:

```
procedure step ( <repeat frequency> ) {
    <pin name> <logical value> ;
    .
    .
}
```

- repeat frequency: specifies the number of repetitions of the initial pattern
- pin name: port name of Verilog top module which has pin type is only “in” or “inout”
- logical value: specifies 0/1. ONLY the input value is specified.

NOTE:

Each “in” or “inout” pin may have a fixed input value by the attribute of pin information file. And that value is enabled even in the initial sequence. However, when a logical value is specified in the initial file, priority is given to logical value and this value is maintained in the initial sequence as long as there is no change in the logical value for the same port.

After the end of initial sequence, the fixed input value specified by the attribute of pin information file is enabled again.

Ex:

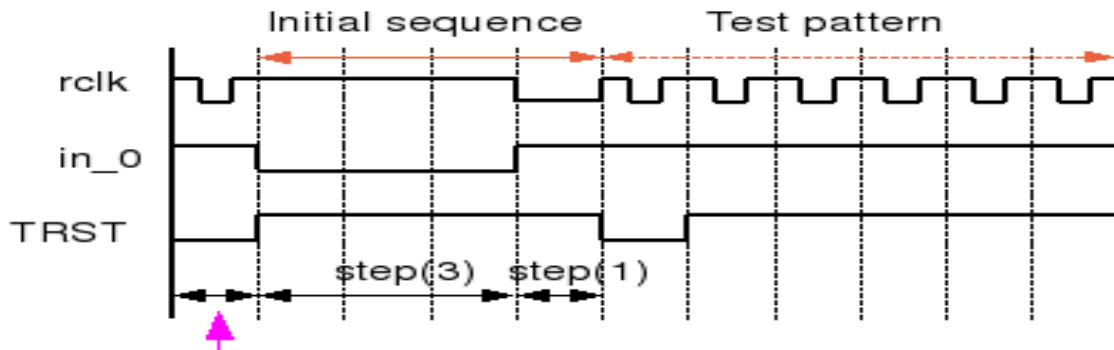
It is assumed that rclk has defined as clock in timeplate file procedure clk, and for in_0, fixed input value of “1” has been the specified attribute of the pin information file

The initial file is:

```
procedure step (3) {
    rclk 1;
    in_0 0 ;
}
procedure step (1) {
    rclk 0 :
    in_0 1;
```

}

The waveform is created is as follow:



Test logic reset cycle:

TRST is asserted immediately after pattern start

The initial sequence specified in initial file starts from the 2nd cycle, then test pattern starts.

3.3 Modifying BSDL file

BSDL file is output from SCAAP tool which may contain some appropriated points to run in IOWrap tool. So MUST modify and update correct information before generating patterns.

3.3.1 Updating information about differential motion pins

When using SCAAP tool, BSDL file does not have a definition for differential motion pins (external pins shared one IO cell instance). So can not generate patterns corresponding to the differential motion pins.

To create those patterns, MUST add more information in BSDL file.

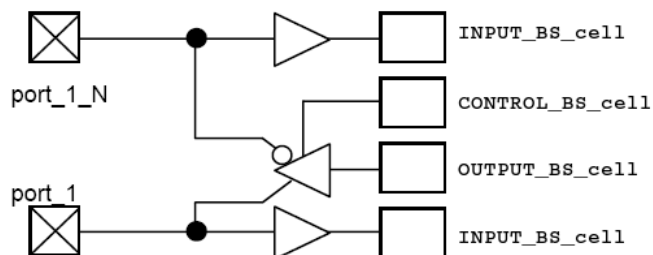
Because there are many type of differential motion pins, BSDL information corrected according to the type are stated below:

- **Case1:**

- both 2pins are inout contact pins or negative pin is no-contact pin
- each pin has its own INPUT_BS_cell
- both 2pins share CONTROL_BS_cell and OUTPUT_BS_cell

→ those ports are treated as differential motion output pins. It means that they have different output voltage.

```
port(
  :
  P006_port_1:inout bit;
  P007_port_1_N:inout bit;
  :
  attribute PORT_GROUPING of product name : entity is
  "Differential_Voltage ((P006_port_1,P007_port_1_N));";
  :
);
```

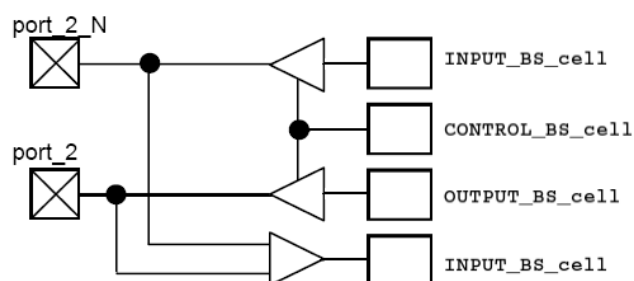


• **Case2:**

- both 2pins are inout contact pins or negative pin is non-contact pin.
- each pins has its own OUTPUT_BS_cell
- both 2pins share CONTROL_BS_cell and INPUT_BS_cell

→ those ports are treated as differential motion input pins. It means that they have different input current.

```
port(
  :
  P008_port_2:inout bit;
  P009_port_2_N:inout bit;
  :
  attribute PORT_GROUPING of product name : entity is
  "Differential_Current ((P008_port_2,P009_port_2_N));";
  :
);
```

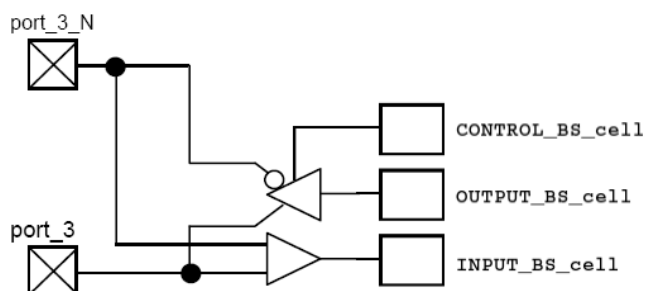


• **Case3:**

- both 2pins are inout contact pins
- both 2pins share CONTROL_BS_cell, OUTPUT_BS_cell and INPUT_BS_cell

→ Those ports are treated as differential inout motion pins. It means that they have different input current and different output voltage.

```
port(
  :
  P010_port_3:inout bit;
  P011_port_3_N:inout bit;
  :
  attribute PORT_GROUPING of product name : entity is
  "Differential_Voltage ((P010_port_3,P011_port_3_N));"&
  "Differential_Current ((P010_port_3,P011_port_3_N));";
  :
);
```



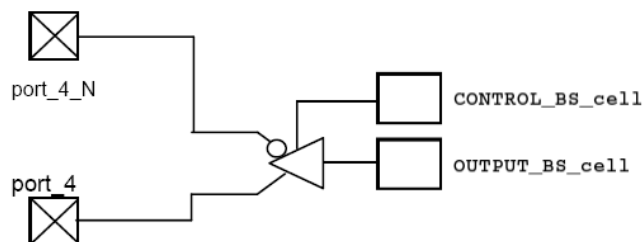
• **Case4:**

- both 2pins are output contact pins
- both 2pins share CONTROL_BS_cell and OUTPUT_BS_cell

→ those ports are treated as differential motion output pins. It means that they have different

output voltage.

```
port(
  :
  P012_port_4:out bit;
  P013_port_4_N:out bit;
  :
  attribute PORT_GROUPING of product name : entity is
  "Differential_Voltage ((P012_port_4,P013_port_4_N))";
  :
```

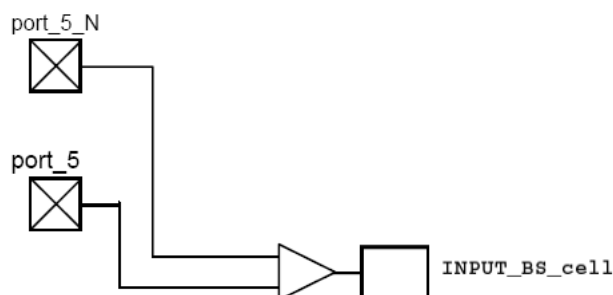


• **Case5:**

- both 2pins are input contact pins
- both 2pins share INPUT_BS_cell

→ those ports are treated as differential motion input pins. It means that they have different input current.

```
port(
  :
  P014_port_5:in bit;
  P015_port_5_N:in bit;
  :
  attribute PORT_GROUPING of product name : entity is
  "Differential_Current ((P014_port_5,P015_port_5_N))";
  :
```



3.3.2 Modifying information about TAP pins

Base on rule of IOWrapgen tool, pin attributes of TAP pins should be:

- TCK, TMS, TDI, TRST: input only (one-way input)
- TDO: output only (one-way output)

As normal, they are defined non-contact probe pins. But in case, those pins are used as 2-way user pins (contact probe pins), MUST modify the pin information file and pin attributes of BSDL file to pass input checking process of IOWrap and execute pattern generation.

Because TAP pins is test control pin, can not generate patterns for them. Although changing information of those pins, there is no influence on the test patterns and test quality.

With pin information file:


- pin allocated in TCK, TMS, TDI, TRST: inout → in
- pin allocated in TDO: inout → out

With BSDL file, in <boundary register description> statement:

- change type of BS cell: BC_1, BC_2 → BC_4
- change port_name → *
- change function: CONTROL, INPUT, OUTPUT → INTERNAL
- change value: 0 /1 → X

Example:

If P001_A is one of TAP pins (TCK, TMS, TDI, TDO, TRST), functions of BS_cells added in IO terminals of P001_A are changed to INTERNAL and pattern is generated by assuming that there is no connection with the external pin.

<pre>"0 (BC_2, *, CONTROL, 0)," & "1 (BC_2, P001_A, OUTPUT3, X, 0, 0, Z)," & "2 (BC_2, P001_A, INPUT, X)," &</pre>		<pre>"0 (BC_4, *, INTERNAL, X)," & "1 (BC_4, *, INTERNAL, X)," & "2 (BC_4, *, INTERNAL, X)," &</pre>
--------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------

3.3.3 Modifying to generate patterns where added BS_cell is disregarded

In IOWrapgen, although all the ports where BS_cell are added are treated as test targets.

But in some cases, some ports are excluded for test target. So MUST modify BSDL and pin information file to exclude those ports out of pattern generation process. Added BS_cell is treated as dummy_cell that is not connected to the external pin.

With pin information file:


- inout → nonbs_inout
- in → nonbs_in
- out → nonbs_out

With BSDL file:

- Change <pin type> in <logical port description>: input/output/inout → linkage
- In <boundary register description> statement:
 - Change type of BS cell: BC_1, BC_2 → BC_4
 - Change port_name → *
 - Change function: CONTROL, INPUT, OUTPUT → INTERNAL
 - Change value: 0 /1 → X

Example:

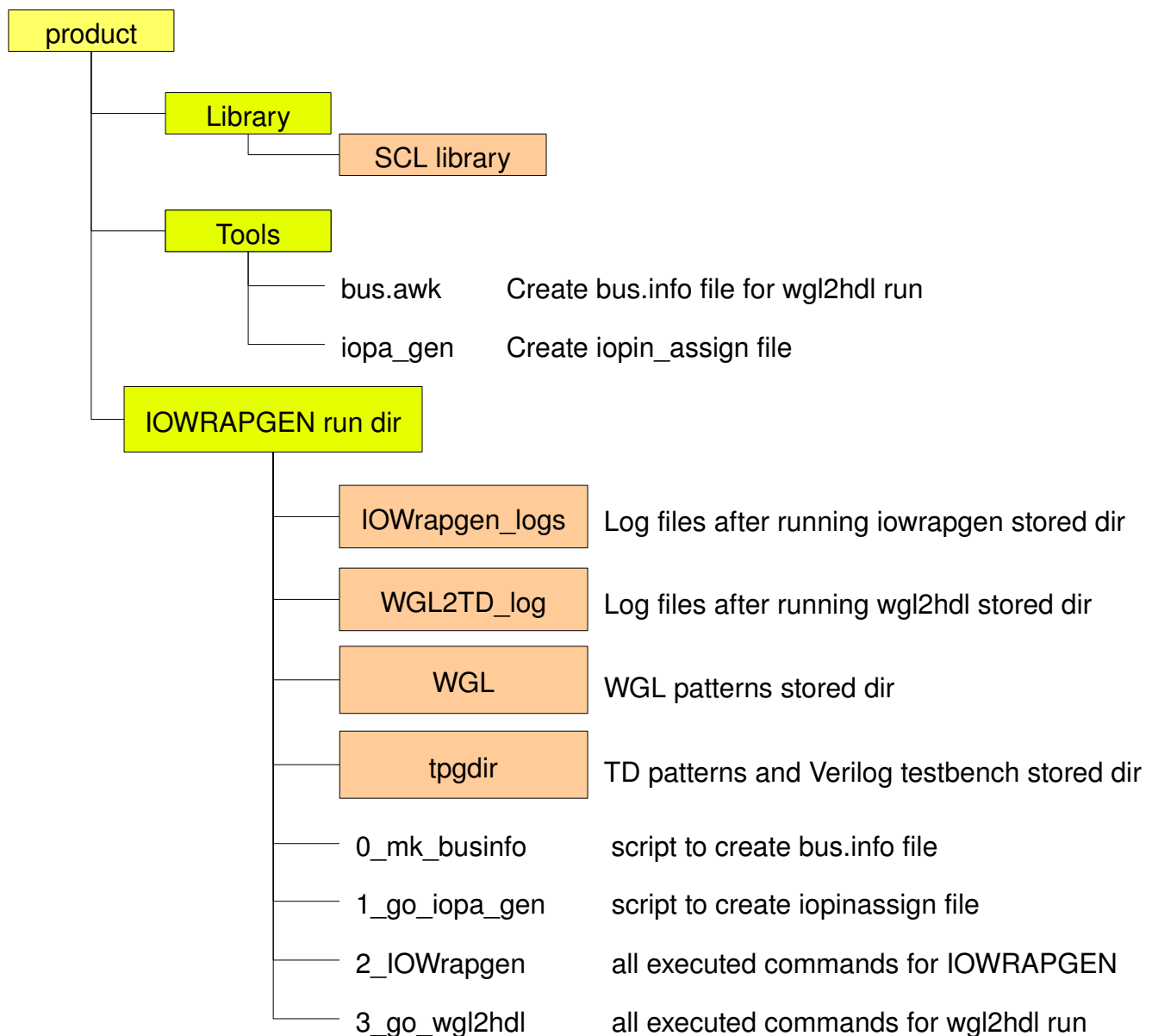
When patterns where P001_A is excluded from the test target is generated

<pre>Port(P001_A: inout bit; ...) ... "0 (BC_2, *, CONTROL, 0)," & "1 (BC_2, P001_A, OUTPUT3, X, 0, 0, Z)," & "2 (BC_2, P001_A, INPUT, X)," &</pre>		<pre>Port(P001_A: linkage; ...) ... "0 (BC_4, *, INTERNAL, X)," & "1 (BC_4, *, INTERNAL, X)," & "2 (BC_4, *, INTERNAL, X)," &</pre>
----------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------

4. IOWRAPGEN Execution

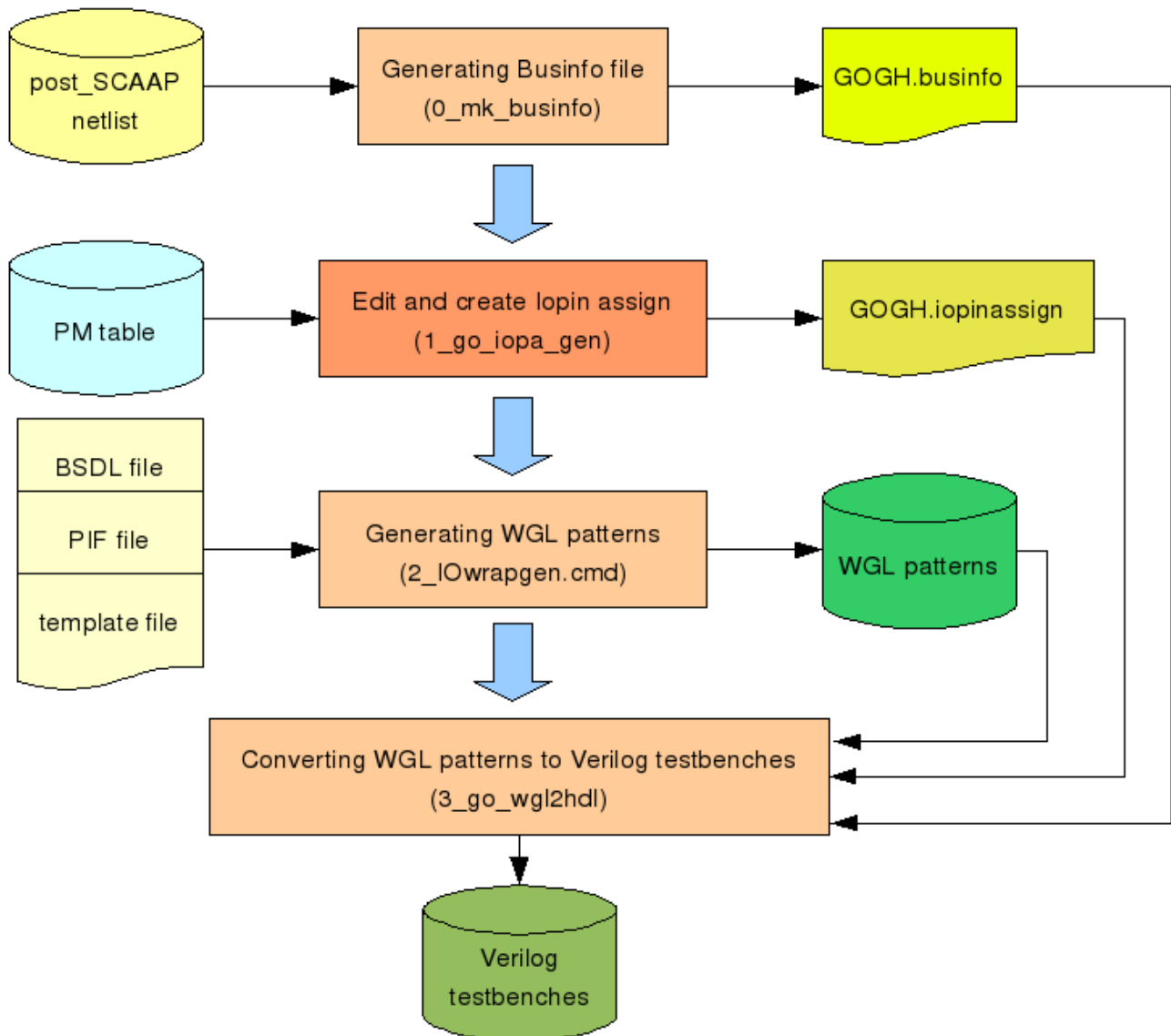
This part introduces about IOWRAPGEN environment, IOWRAPGEN working flow to generate the needed boundary scan test patterns.

4.1 Environment



4.2 Execution flow

To generate boundary scan patterns, follow below execution flow.



4.3 IOWrap run file

Run file contains the commands for IOWRAPGEN to generate 1 kind of pattern (BS and DCP patterns are generated together.)

For example, below file is run file for generating BS and DCP patterns – all pins type

```
set EMODE = WT_allpin
set SCAAP_dir = ../1_scaap
set BSDL = TOP_v4r2s1.bsd
set Prod = TOP_v4r2s1
bs -M 8000 -source /common/appl/Renesas/IOWrapgen/dotfiles/IOWrapgen_v2.04.CSHRC \
iowrapgen \
  -bsd ${{SCAAP_dir}}/Release/${{{BSDL}} } \
  -pinf ${{SCAAP_dir}}/${{{Prod}}}_${{{EMODE}}}.pif \
  -time ${{SCAAP_dir}}/${{{Prod}}}_${{{EMODE}}}.timeplate \
  -wait 1 \
  -pin target \
  -name ${{Prod}}_${{{EMODE}} } \
  -outdir ./WGL/
```

Syntax of command to invoke IOWRAPGEN tool is as follow:

```
iowrapgen \
  -pinf      <pin information file name> \
  -bsd      <BSDL file name> \
  -time      <timeplate file name> \
  -wait      <Leak discharge waiting cycle> \
  [-pin      <WGL output port selection>] \
  [-initial   <initial pattern generation file name>] \
  [-activate  <IO simultaneous change decrease setting (divN) >] \
  [-outdir    <output directory>] \
  -name      <output file name> \
  [-clamp     <CLAMP instruction pattern output flag>] \
  [-hiz       <HiZ confirmation pattern output flag>] \
  [-diag      <BS chain verification pattern output flag>]
```

-pinf: pin information file name . (must)

-bsd: BSDL file name. (must)

-time: timeplate file name. (must)

-wait: leak discharge wait cycle. (must)

[-pin]: WGL output port selection ('all/target/reduce'). (optional)

Default is 'target'.

'all': generates all pin contact (yes) patterns regardless of the description of <contact> of pin information file. (iowrap.wgl pattern is not generated. All pins are output to WGL)

'target': the pin where 'no' is specified in <contact> of the pin information file is also output to WGL. (logical value of pin where 'no' is specified in <contact> is always 'X')

'reduce': the pin where 'no' is specified in <contact> of pin information file is not output to WGL.

[-initial]: initial file name. (optional)

By default, there is no initial pattern generation.

[-activerate]: settings for reducing in number of simultaneous IO changes. (optional)

(DivN: even number $N \leq 32$)

By default, there is no reduction.

As per the 'DivN' settings, the number of simultaneous IO changes is reduced to (No.of default changes) * (1/N)

[-outdir]: name of directory where output file is stored (optional)

By default, it is current directory.

[-name]: first name of output file. (must)

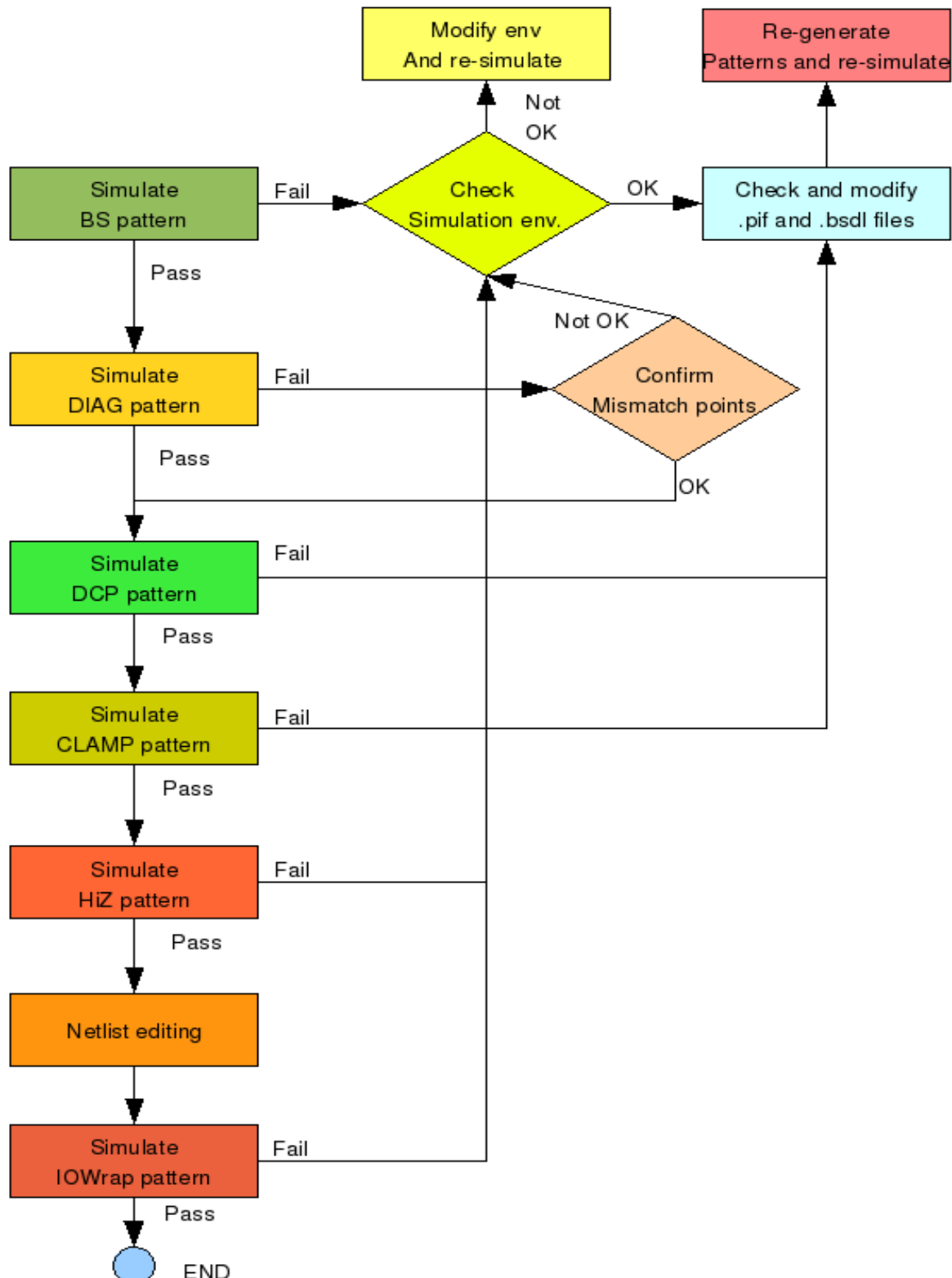
[-clamp]: CLAMP instruction file pattern generation flag('yes/no'). Default is 'no' (optional)

[-hiz]: HiZ confirmation test pattern generation flag ('yes/no'). Default is 'no' (optional)

[-diag]: DIAGNOSTIC pattern flag ('yes/no'). Default is 'no' (optional)

IV. SIMULATION FLOW

This part gives some suggestions when running boundary scan simulation. They are experiments after running simulation in past project. Please refer this carefully and find down the correct way to apply in current project (if any).



It is recommended to execute verification BS pattern and DIAGNOSTIC pattern on priority.

If BS pattern is FAIL, review simulation environment , .pif file and .bsdl file; then rerun again.

If DIAG pattern is FAIL, must confirm the mismatch points if they are dummy or not. This confirmation is described detail in section II.1

Verification is complete after confirmation. (excluding IOWrap pattern)

IOWrap pattern has special checking, so MUST modify netlist to check this pattern. The detail information is described in section II.2

1. Verification Of DIAGNOSTIC Pattern

DIAG test executes same as DCP test but in some pattern of DIAG test output_BS_cell is set to '0' in VOH/L test period. So at that time, external output of corresponding ports are set to 'Z' and expected value is 'HiZ'.

However, according to the simulation IO model, there may be cases: output value is 'X' or PULL0/PULL1. So there may be occurrence of expected value mismatch.

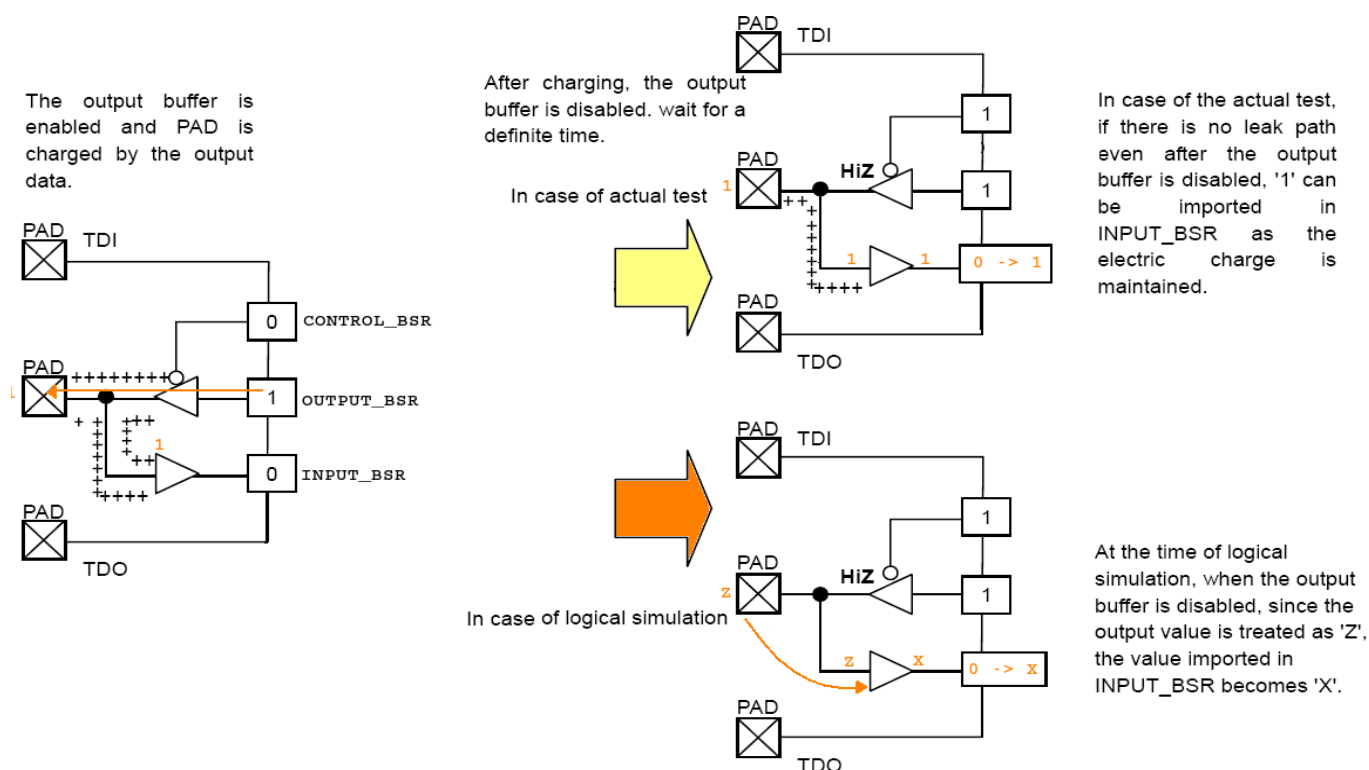
So make sure that when running simulation DIAG test, turn off PULL control signal in netlist and PULL definition in IO model. If mismatches are still exist. Checking below items:

- [1] all expected values of corresponding ports are 'Z'.
- [2] DCP test for all contact pins has no mismatch.
- [3] HiZ test has expected value mismatches occur at the same ports mentioned in [1]

→ If [1] ~ [3] are implemented, can be sure that there is no problem in DIAG test even if there are mismatches occur during simulation verification.

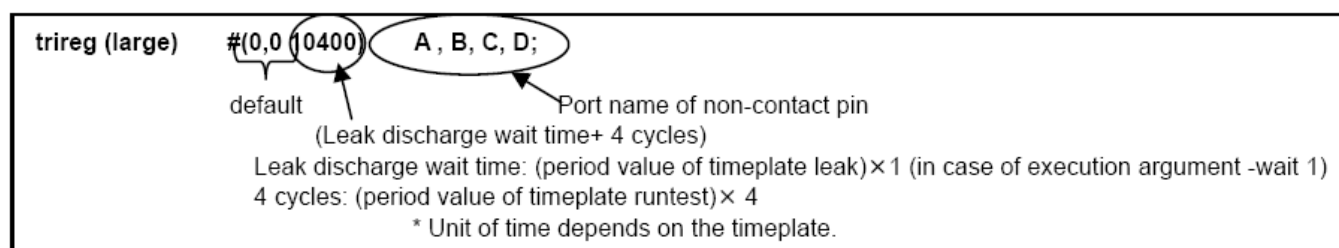
2. Verification Of IOWrap Pattern

Leak fault coverage in I/OWrap test pattern is a test that confirms that the electric charge charged to the PAD by output data disables the output buffer, and whether or not it is maintained even after being in standby for a definite time. However, in logical simulation, since there is no concept of maintaining the charged electric charge in PAD, when output buffer is disabled, output data is treated as 'Z'. When this 'Z' is imported in INPUT_BS_ cell, the value is 'X' and expected value mismatch occurs in TDO output judgment.



This expected value mismatch can be avoided by trireg declaration of non-contact pin in netlist for logical simulation. Non-contact pin is declared as trireg after defining inout pin of top module in netlist. Pin declared as trireg can maintain previous value even after the value is not active (When in HiZ condition) for the specified time.

Description format of trireg declaration is as given below.



As per the above-mentioned description, leak fault coverage of I/OWrap test pattern also can be confirmed by logical simulation. However, when pin declared as trireg is included in the Pull-up/down confirmation test, since the simulation result does not match with the value opposite to the expected value ('0' if expected value is '1' and '1' if expected value is '0').

Example:

- Period value of timeplate leak: 10000 [ns]
- Period value of timeplate runtest: 100 [ns]
- Data retention time as per trireg declaration is

$$10000 \text{ [ns]} \times 1[\text{cycle}] + 100 \text{ [ns]} \times 4[\text{cycle}] = 10400 \text{ [ns]}$$

Example:

```
inout CXVD5;
inout CXVD6;
inout CXVD7;
inout CXVD8;
inout CXVD9;
inout CXP027;

// trireg for IOWrap MSPAD (100400ns -> 100400000ps)
trireg (large) [1:0] #(0,0,100400000) CXBA ;
trireg (large) [13:0] #(0,0,100400000) CXMA ;
trireg (large) [31:0] #(0,0,100400000) CXDQ ;
trireg (large) [3:0] #(0,0,100400000) CXDQS ;
trireg (large) [3:0] #(0,0,100400000) CXDM ;
trireg (large)      #(0,0,100400000) CXDCS_N , CXCK , CXCK_N , CXRAS_N , CXCAS_N ,
CXDWE_N , CXDCKE ;
```