2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE MUNICH, GERMANY OCTOBER 15-16, 2024

Synthesis Strategy for Standard Cell Library Validation

H. Shin[†], S. Do, J. Lee

Foundry Division, Samsung Electronics,

Hwaseong-si, Republic of Korea (hy22.shin@samsung.com[†])

***Abstract*— In semiconductor design, synthesis can be considered a blessing. It not only reduces design time but also greatly helps in improving circuit performance and productivity. Most EDA tool company have conducted extensive research to develop more efficient synthesis algorithm, and most of these are focused at improving semiconductor performance, which is commonly referred to as Power, Performance and Area (PPA). However, from the perspective of a foundry engineer who design libraries, this algorithm can be disadvantageous in terms of coverage when they want to see if the developed library actually has no problems in synthesis flow. An efficient synthesis algorithm utilizes the optimal standard cells for the best solution, resulting in many unused standard cells in the library. Through this study, we introduce a synthesis strategy that focuses on the utilization of standard cells. Our strategy is to enforce the synthesis tool so that as many types of standard cells as possible can be used in the synthesis flow, and we have confirmed that the usage rate of standard cells can be increased through various designs. Furthermore, we introduce that the standard cell usage information obtained from the synthesis results can be used for library optimization.**

Keywords—standard cell; synthesis; library validation

I. INTRODUCTION

Semiconductor standard cells are the basic building blocks of semiconductor chips, referring to pre-designed functional blocks. They have a predetermined size and functionality that can be reused by designers when designing semiconductor chips. Standard cells come in various types, including logic gates, memory cells, and arithmetic operation cells. These standard cells are provided in the form of libraries, and designers can select and use the necessary standard cells. Using standard cells to design semiconductor chips can improve design efficiency and chip performance. In the synthesis process, the design is composed of the optimal combination of standard cells. However, not all standard cells in the library are used because they use synthesis algorithms for PPA, and there are many standard cells that are not used in actual designs. It is difficult for library designers to predict in advance whether there will be problems when using standard cells in semiconductor chips. Therefore, through this study, we have successfully increased the utilization rate by finding a way to force unused standard cells to be used in the synthesis process. The following basic and essential standard cells were targeted for research: Combinational cells, Sequential cells, Multi-bit cells, Clock-Gating cells and Synchronizer cells.

- Combinational cells: AND, OR, NOR, NAND
- Sequential cells: Latch, Scan flip-flop(FF)
- Multi-bit cells: Multi-bit flip-flop (MBFF)
- Clock-Gating cells: ICG
- Synchronizer cells: SDFFY

Table I shows the number of test cells targeted by our synthesis strategy and the "d*on't care list",* which is a type of cell excluded from the target of our synthesis strategy.

Table I. Synthesis Strategy Test Cells and don't care cells

Total number of cell	***Don't care list***			***# The number of Test cell***			
	Finishing	***Delay cell***	***DFF***	***SDFFY***	***ICG***	***MBFF***	***Combinational & Sequential***
154	24	6	9	9	3	14	103

II. Related Work

The integrity of standard cell libraries is already guaranteed from the perspective of chip designers, so it is not an active research area. However, there are some papers that deal with the verification of standard cell libraries, as follows:

The first paper [1] introduces a verification method for power management standard cells that are not covered in synthesis. The importance of low power design increases, so standard cells required for these designs are also used. Therefore, verifying low power libraries has become more important and different methods were required than before. They use power-intent strategy described in UPF and multi-voltage design platform including all cells to verify. The UPF commands specify the functional model and a list of implementation targets for cells. If the cell described through the command was used differently from what we intended, we could check it through the design verification tools and finally find that there was a problem with the library. It was also possible to check the consistency between each view through Verilog simulation using the same scenario.

The second paper [2] introduces a verification method for the physical view of standard cells. It presents a comprehensive standard cell abutment verification tool to generate test cases for quality assessment, ensuring that standard cells can be placed without causing design rule check (DRC) violations during placement, Vt swap, and engineering change orders (ECO). If a DRC violation is due to routing, it can be resolved by re-routing, but if it is due to cell abutment, fixes may involve inserting space or redesigning cells. The tool efficiently checks for safe abutment of standard cells while eliminating redundant cases, resulting in a significant reduction in the test area to cover all boundaries and maintaining 100% coverage of all standard cell abutment topologies.

These papers focused on verifying specific standard cells or the physical view, so research on different views or basic standard cells is necessary. We will cover the content that is not covered in the two papers introduced above.

III. Synthesis Strategy

To implement a chip, considering power, timing, and area constraints, the Register Transfer Level (RTL) needs to be converted into a gate-level netlist. This process is called synthesis. Customers utilize our standard cell library during synthesis. We validate the standard cell library to ensure that customers encounter no issues when using it. By using our synthesis strategy of pre-conducting synthesis using the OR1200 design for developed standard cells, we can ensure that our standard cell library is flawless. The figure below, Figure 1, represents our synthesis strategy in a flow chart. By analyzing the synthesized design, we generate a "don't use" list for the next run. This process will be explained in detail in *A. Don't Use Policy*. The overall flow is repeated until we achieve coverage. This process will be explained in detail in *B. Multiple Run*. This flow is used to cover all the cells we target as described in the Introduction. The following subsections contain descriptions of the individual steps in this flow.

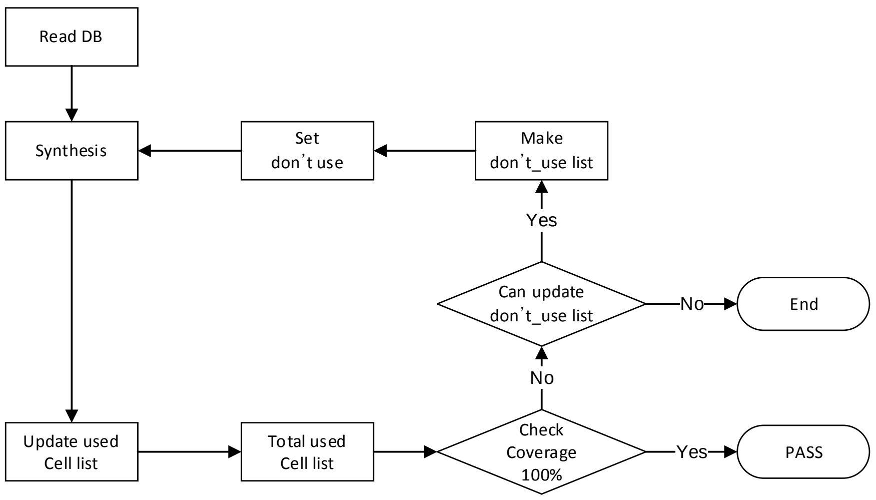


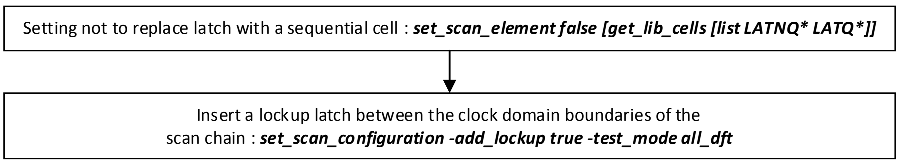
Figure 1. Synthesis strategy flow

A. Don't use Policy

The *dont_use* attribute is used to exclude objects from the target library during optimization when set to true. In other words, the standard cells with the *dont_use* attribute set to true are not used in the gate level netlist. Using the attributes of the *dont_use* attribute, below, we introduce the cell types that are targeted for verification in our synthesis strategy.

i. Latch Cell

In Design For Testability (DFT), latch cell is used as a lockup latch to match the clock skew between registers and prevent hold time violation. In our synthesis strategy, we perform multiple synthesis runs because not all cells are used in a single synthesis for latch or flip-flop. We will introduce this in detail in Section *B. Multiple Run*. In Figure 2, we explain the flow of using a latch cell as a lockup latch in the synthesis strategy and the corresponding TCL commands.



Setting not to replace latch with a sequential cell : ***set_scan_element false [get_lib_cells [list LATNQ* LATQ*]]***

Insert a lockup latch between the clock domain boundaries of the scan chain : ***set_scan_configuration -add_lockup true -test_mode all_dft***

Figure 2. Lockup latch setup

ii. Flip-Flop Cell

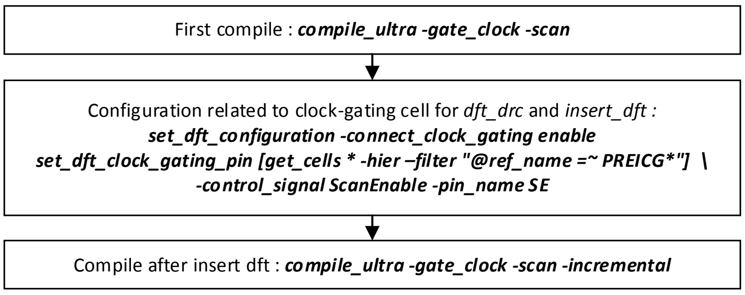
In the DFT of the synthesis, a Scan D Flip-Flop (FF) is used to configure the scan chain. The Scan DFF consists of a DFF and a Mux.

iii. Integrated clock gating Cell

Clock gating cells are used for low power consumption in clock gating methodologies, and the synthesis tool can determine where the cell can be used to provide a significant power savings for clock gating. In DFT, the clock-gating cell and its pin are designated and used with it. In Figure 3, we describe the flow for setting up a clock-gating cell related to *dft_drc* and *insert_dft* in the synthesis strategy and compiling it using the *gate_clock* option.

iv. Synchronizer Flip-Flop Cell

A synchronizer cell is used to transmit data from one clock domain to another asynchronous clock domain. We use two flows that can verify whether the synchronizer flip-flop cell is used well for the scan path: Using RTL with a synchronous reset pin (CASE 1), using RTL with an asynchronous reset pin (CASE 2).



First compile : ***compile_ultra -gate_clock -scan***

Configuration related to clock-gating cell for *dft_drc* and *insert_dft :*
set_dft_configuration -connect_clock_gating enable
***set_dft_clock_gating_pin [get_cells * -hier –filter "@ref_name =~ PREICG*"] ***
-control_signal ScanEnable -pin_name SE

Compile after insert dft : ***compile_ultra -gate_clock -scan -incremental***

Figure 3. Clock-gating cell setup in DFT flow

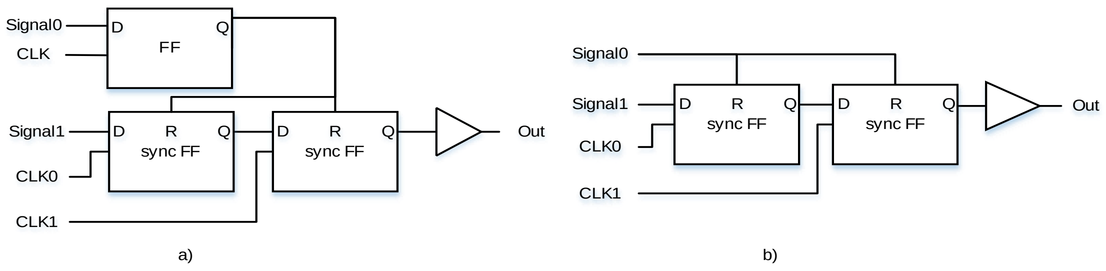
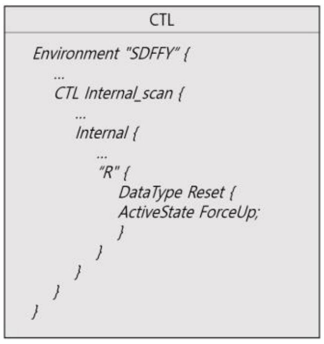


Figure 4. a) Synchronous reset design (CASE 1), b) asynchronous reset design (CASE 2)

Figure 4-a) is the synchronizer design using a synchronous reset with a DFF. If the reset is described in the Core Test Language (CTL) as an asynchronous pin, the scan chain is not tied due to mismatch between the RTL (synchronized reset pin) and the CTL (asynchronous reset pin). In conclusion, we can verify whether the reset is well described in the CTL. Figure 4-b) is the synchronizer design using an asynchronous reset. If the CTL recognizes the reset as an asynchronous pin, the scan is properly tied only if the CTL's *Active State* is properly described. In conclusion, we can verify whether the synchronizer cell is used well in the scan chain of the synthesis and whether the CTL's *ActiveState* is well described at the same time. Through both cases, we can verify both the CTL and the Liberty. Figure 5 shows an example of declaring the *ActiveState* of a reset type pin in CTL.



```
CTL
Environment "SDFFY" {
    ...
    CTL Internal_scan {
        ...
        Internal {
            ...
            "R" {
                DataType Reset {
                ActiveState ForceUp;
                }
            }
        }
    }
}
```

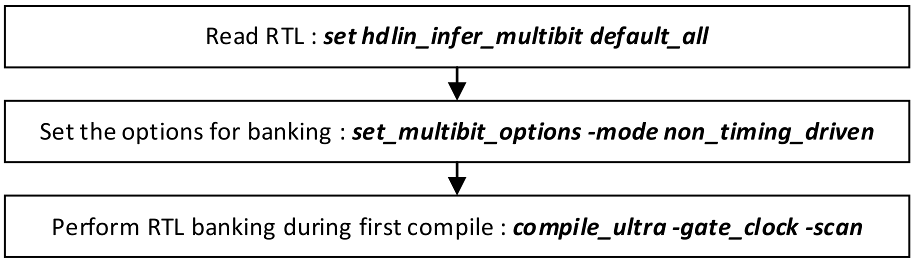
Figure 5. CTL description of SDFFY cell with reset type pin

v. Multi-bit Flip-Flop

In the synthesis process, single-bit FF is replaced with multi-bit FF (MBFF) for advantageous in terms of PPA. To perform synthesis using a MBFF cell, we use two methods of synthesis strategy: RTL inference Flow [3], Register mapping Flow [4].

a) RTL inference Flow

In RTL inference flow, the option for banking has been set to *non_timing_driven*. As a result, the tool uses MBFF cells whenever it can. Figure 6 illustrates the explanation and command for the "RTL inference flow of MBFF".



Read RTL : ***set hdlin_infer_multibit default_all***

Set the options for banking : ***set_multibit_options -mode non_timing_driven***

Perform RTL banking during first compile : ***compile_ultra -gate_clock -scan***

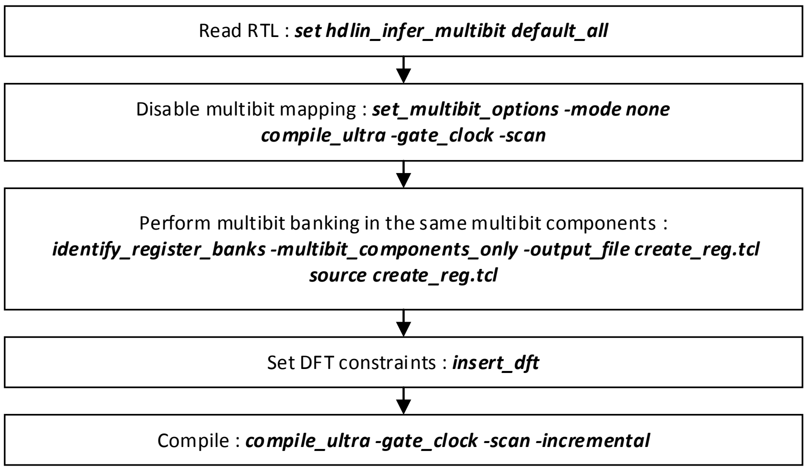
Figure 6. RTL inference flow of MBFF cells

b) Register mapping flow

In register mapping flow, to replace single-bit registers with multi-bit registers in a multi-bit component, we use the *identify_register_banks* command with the *multibit_components_only* option. Figure 7 illustrates the explanation and command for the "Register mapping flow of MBFF".

In the synthesis process, we can check if the MBFF properly replaces the single-bit FF in a report file from *report_multibit_banking*. Additionally, if there is a mismatch in pin consistency between single-bit FF and MBFF, and the value of the *nextstate_type* attribute of the scan-related pin of the MBFF is incorrectly described, the design compiler will not use the MBFF and the following error will occur, which can also be verified from the perspective of library consistency.

Error: The bank's pin is not consistent with candidate register cells' pin. (PSYN-1204)



Read RTL : ***set hdlin_infer_multibit default_all***

Disable multibit mapping : ***set_multibit_options -mode none***
compile_ultra -gate_clock -scan

Perform multibit banking in the same multibit components :
identify_register_banks -multibit_components_only -output_file create_reg.tcl
source create_reg.tcl

Set DFT constraints : ***insert_dft***

Compile : ***compile_ultra -gate_clock -scan -incremental***

Figure 7. Register mapping flow of MBFF cells

B. *Multiple Run*

We perform synthesis multiple runs to increase the utilization of standard cells in synthesis. In particular, we check the number of latch or flip-flops used in one synthesis execution and determine the number of synthesis executions accordingly.

- *The number of latch or flip-flop types in a library / The number of latch or flip-flops used in one synthesis run = The number of synthesis runs*

Since the cells used in previous synthesis have already been verified, the *dont_use* attribute is set to true and in the next synthesis, only cells that are not used in the previous synthesis are verified.

C. *Cells must include from the synthesis strategy scope*

To synthesize a design correctly, all logic must be implemented using standard cells from the library. However, if there are many cells set to *dont_use*, synthesis may become impossible. To prevent this, a universal gate has been excluded from the *dont_use* condition. The universal gate is a gate that can create all logic gates with just one type of gate, and there are NAND and NOR. Among them, we have taken measures to use NOR2 as a universal gate and always use it without being included in the *dont_use* list.

D. *Cells excluded from the synthesis strategy scope*

1) Physical Cell

Physical cells such as ANTENNA, FILLCAP Cell are used for P&R, not synthesis, so they are excluded from the target of our strategy.

2) Power management cell

To verify that power management cells are used in synthesis, a verification strategy using UPF is required. They will be verified in the verification method of *II. RELATED WORK.*

IV. Experimental Result

Through the synthesis strategy proposed in this study, we were able to increase effectively the utilization rate of standard cells to 100% using a small OR1200 design. The utilization rate was compiled based on the available standard cells among the standard cells that can be used in synthesis, and the standard cells used in the backend design stage or DFM stage were excluded from the parameter. Figure 8 represents the directory structure for the synthetic strategy. The environment for the experiment is divided into DB_DIR and RUN_DIR. The DB_DIR contains databases for synthesis, and for the OR1200 design, a design with SRAM removed is used to view only the Standard Cell. In the case of RUN_DIR, it is a directory where the actual synthesis is performed, and there are four run paths to view combinational & sequential cells, multi-bit FF, and synchronizer cells.

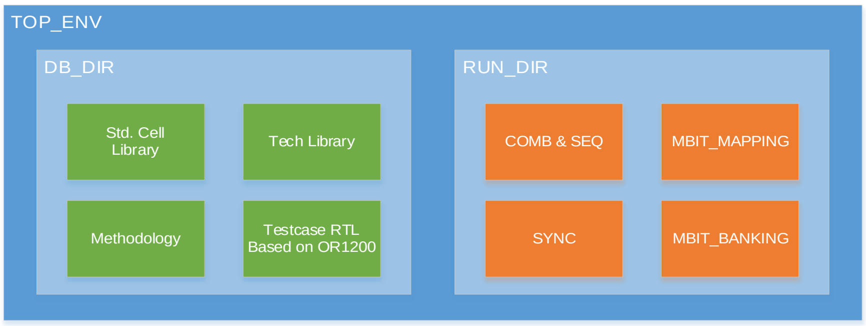


Figure 8. Experimental environment (directory structure)

A. *Combinational cells, Sequential cells and Multi-bit cells*

The results of our synthesis strategy case are as follows. In order to ensure that all cells are used in the synthesis, we perform multiple synthesis runs. The number of synthesis runs is determined by dividing the number of latches by the number of latches used in a single synthesis. In this synthesis strategy, a single synthesis uses one latch or two latches. The number of latches used in the test library is 14, and since it uses 1.5 latches per synthesis, the number of synthesis iterations is set to nine.

Each time a synthesis is performed, the cells that have already been used in the synthesis are set to *dont_use* attribute as *true* and excluded from the synthesis target in the next synthesis. This forces as many different types of standard cells to be used in the synthesis as possible. Table II shows the number of cells used, the coverage of the synthesis strategy, and the usage status of latch in each synthesis run. Through nine iterations of synthesis, we achieved a 100% utilization rate of Samsung Foundry's standard cells in the OR1200 design.

Table II. Utilization information of standard cells through multiple synthesis runs

	Initial	***Run1***	***Run2***	***Run3***	***Run4***	***Run5***	***Run6***	***Run7***	***Run8***
Used cell	70	18	5	2	1	2	2	2	1
Utilization rate of cells	67.9%	85.4%	90.2%	92.2%	93.2%	95.1%	97.0%	99.0%	100%
Total Used cell	70	88	93	95	96	98	100	102	103
Don't use(LAT)	+Q/QN	+SPQ	+RPQN/ RQ	+SQN	+SPRQ	+NQ/ NQN	+NRPQN /NRQ	+NSPQ/ NSQN	+NSPRQ

Figure 9 represents a chart of standard cell utilization and the number of cells used with multiple synthesis. After performing the fourth synthesis, more than 90% of the total cells are used, and latches not used in previous synthesis are used in subsequent synthesis.

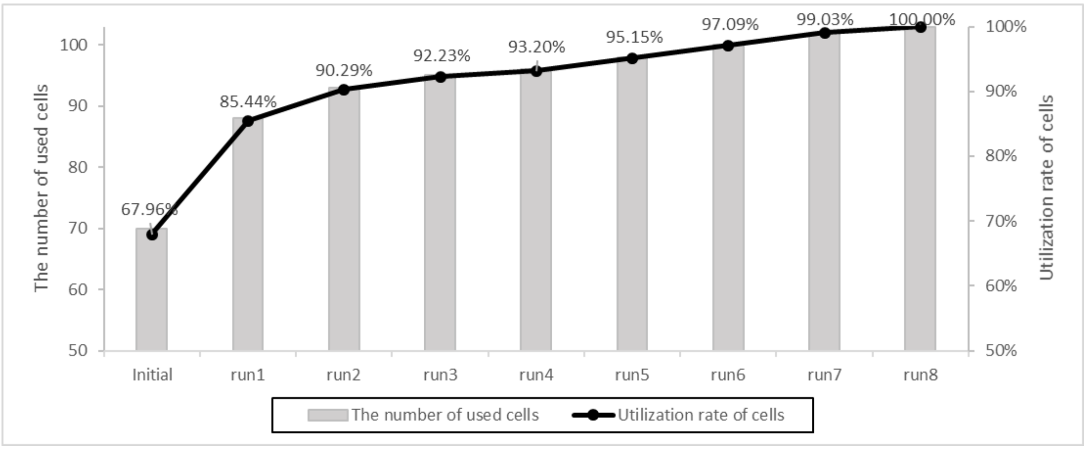


Figure 9. Chart of combination & Sequential cell utilization and the number of cell used

In Table III, the synthesis results for the MBFF register mapping flow are shown, including the number of all of multibit test cells and used cells, as well as the corresponding synthesis strategy coverage. After achieving a 100% utilization rate of standard cells with three synthesis runs, the results of subsequent runs were omitted.

Table III. Utilization information of MBFF cells through multiple synthesis runs

	Initial	Run1	Run2
MBFF used cell	9	11	14
Utilization rate of cells	64.20%	78.50%	100%
Total used cell	9	2	3

B. *Clock-Gating Cells*

In our synthesis strategy, we check if a clock-gating cell is properly used according to its function using the following command in Figure 10. When the clock-gating cell is used in clock gating logic, we can check the information: *ICG cells are used only as clock-gating*.

```
check_clock_gating_cell.tcl

set cg_cells [get_cells -hier -filter "clock_gating_logic==true && ref_name=~*ICG*"]
set cg_cell_list [get_attribute [get_cell $cg_cells] ref_name]

set all_cells [get_cells -hier -filter "is_hierarchical == false && ref_name=~*ICG*"]
set all_cell_list [get_attribute [get_cell $all_cells] ref_name]

if {$cg_cell_list == $all_cell_list} {
            puts "[INFO]: ICG cells are used only as clock-gating"
}
```

Figure 10. clock-gating cell checking command

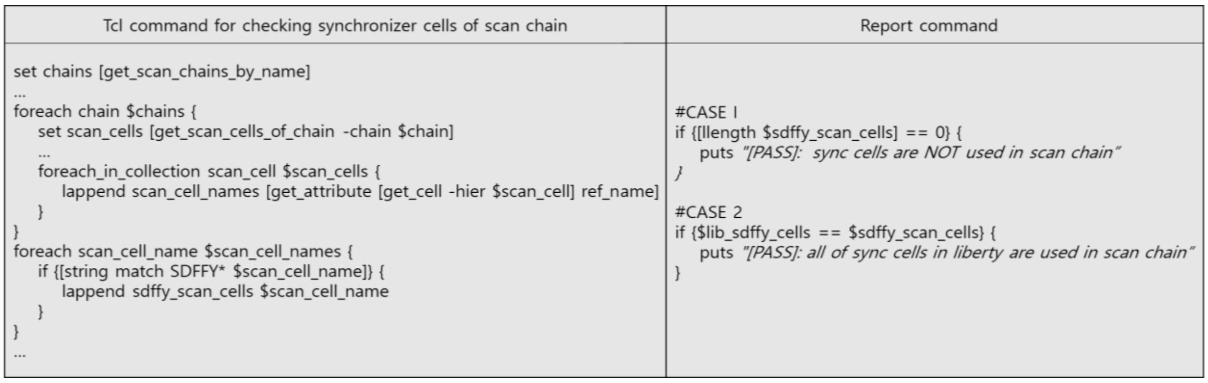
C. *Synchronizer Cells*

We use all nine types of synchronizer cells in our synthesis strategy by changing the values of the *cell_value* and *sync_depth* parameters in the *SEC_AP_GENERIC_SYNCHRONIZER* module. Figure 11 describes the synthesis strategy for synchronizer cells, including its RTL and corresponding parameters.

RTL for synthesis strategy of synchronizer cells	Description of parameters
SEC_AP_GENERIC_SYNCHRONIZER #(.RESET_VAL($cell_value),.DEPTH_SYNC($sync_depth)) CDC_TEST_CASE1 (.CK(CLK), .RN(Q0), .D(B), .Q(Q1)); SEC_AP_GENERIC_SYNCHRONIZER #(.RESET_VAL($cell_value),.DEPTH_SYNC($sync_depth)) CDC_TEST_CASE2 (.CK(CLK), .RN(A), .D(B), .Q(Q1));	<cell_value> If *cell_value* set to 0, synchronizer cell with reset pin is used. If *cell_value* set to 1, synchronizer cell with set pin is used. If *cell_value* set to 2, synchronizer cell without reset or set pin is used. <sync_depth> The *sync_depth* can be set to 2, 3, or 4.

Figure 11. RTL for synchronizer cells and description

In the design compiler, we can check the SDFFY cells in the scan chain. In the case 1, we confirmed that there are no corresponding SDFFY cells by checking the message: *"[PASS]: sync cells are NOT used in scan chain"*. In addition, in the case 2, we confirmed that all SDFFY cells are used in the scan chain by checking message: *"[PASS]: all of sync cells in liberty are used in scan chain"*. The detailed explanation of case 1 and case 2 can be found in *III. SYNTHESIS STRATEGY*. In Figure 12, we describe the TCL command to verify the SDFFY cell used in the scan chain of the design compiler and check the results for both cases.



Tcl command for checking synchronizer cells of scan chain	Report command
see below	see below

Tcl command for checking synchronizer cells of scan chain

```
set chains [get_scan_chains_by_name]
...
foreach chain $chains {
    set scan_cells [get_scan_cells_of_chain -chain $chain]
    ...
    foreach_in_collection scan_cell $scan_cells {
        lappend scan_cell_names [get_attribute [get_cell -hier $scan_cell] ref_name]
    }
}
foreach scan_cell_name $scan_cell_names {
    if {[string match SDFFY* $scan_cell_name]} {
        lappend sdffy_scan_cells $scan_cell_name
    }
}
...
```

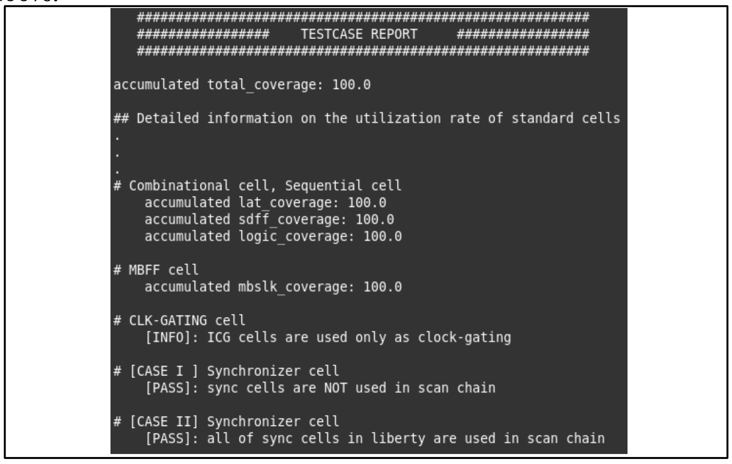
Report command

```
#CASE I
if {[llength $sdffy_scan_cells] == 0} {
    puts "[PASS]:  sync cells are NOT used in scan chain"
}

#CASE 2
if {$lib_sdffy_cells == $sdffy_scan_cells} {
    puts "[PASS]: all of sync cells in liberty are used in scan chain"
}
```

Figure 12. Command for checking synchronizer cells of scan chain and reporting

In Figure 13, we show a report on the results of performing synthesis using our synthesis strategy. By checking the coverage and log of this report, we can confirm that the utilization rate of standard cells in our synthesis strategy has reached 100%.



```
   ##########################################################
   #################    TESTCASE REPORT     #################
   ##########################################################

accumulated total_coverage: 100.0

## Detailed information on the utilization rate of standard cells
.
.
.
# Combinational cell, Sequential cell
    accumulated lat_coverage: 100.0
    accumulated sdff_coverage: 100.0
    accumulated logic_coverage: 100.0

# MBFF cell
    accumulated mbslk_coverage: 100.0

# CLK-GATING cell
    [INFO]: ICG cells are used only as clock-gating

# [CASE I ] Synchronizer cell
    [PASS]: sync cells are NOT used in scan chain

# [CASE II] Synchronizer cell
    [PASS]: all of sync cells in liberty are used in scan chain
```

Figure 13. Standard cell coverage report using our validation process

V. CONCLUSION

The ultimate goal of developing high-quality libraries is to support customers in successfully designing high-performance chips by effectively utilizing the library in the synthesis process from a development methodology perspective. Therefore, we have developed a synthesis strategy focused on standard cell utilization that can be checked by synthesis tools for each cell type. Samsung Foundry verifies that each cell is used correctly in the synthesis flow within the design before distributing the library to customers through the synthesis strategy presented in this paper. This allows us to verify the integrity of all types of standard cells, including combinatorial and sequential cells. Ultimately, we can provide customers with high-quality libraries.

REFERENCES

[1] J. Lee, H. Bak, S. Do, T. Yoo, Gowrishankar Srinivasan, Vishw Mitra Singh Bhadouria, “Low-Power Validation Framework for Standard Cell Library Including Front-End and Back-End Implementation,” DVCON Europe 2023 Oct. 2023

[2] J. -Y. Chueh and C. Tung, "Efficient standard cell abutment checker," 2013 IEEE 20th International Conference on Electronics, Circuits, and Systems (ICECS), Abu Dhabi, United Arab Emirates, 2013, pp. 847-850, doi: 10.1109/ICECS.2013.6815547.

[3] Solvnet, RTL Bus Inference Flow (synopsys.com)

[4] Solvnet, Placement-Aware Multibit Register Banking in Design Compiler Graphical (synopsys.com)