SAMSUNG

3DIC 3DCODE

July 2024

Disclaimer

SAMSUNG ELECTRONICS RESERVES THE RIGHT TO CHANGE PRODUCTS, INFORMATION AND SPECIFICATIONS WITHOUT NOTICE.

Products and specifications discussed herein are for reference purposes only. All information discussed herein is provided on an “AS IS” basis, without warranties of any kind.

The document and all information discussed herein remain the sole and exclusive property of Samsung Electronics. No license of any patent, copyright, mask work, trademark or any other intellectual property right is granted by one party to the other party under this document, by implication, estoppel or otherwise.

Samsung products are not intended for use in life support, critical care, medical, safety equipment, or similar applications where product failure could result in loss of life or personal or physical harm, or any military or defense application, or any governmental procurement to which special terms or provisions may apply.

For updates or additional information about Samsung products, contact your nearest Samsung office.

All brand names, trademarks and registered trademarks belong to their respective owners.

Table of Contents

1. 3DCODE Overview

The exponential growth of the semiconductor industry has replaced the bygone era of Moore's law with the popular concept of 'More than Moore'—a phenomenon that helps add value to devices and incorporate functionality beyond Moore's law.

The various types of chiplet architectures that have been developed help to overcome the standard Moore's limit, thus increasing semiconductor integration and design complexity. Unlike traditional 2D designs that feature one die and one package, 3D designs such as 3DIC and Silicon in Package (SiP) feature multiple dies that include silicon interposers and packaging with ReDistribution Layer (RDL).

In addition, the increased complexity poses challenges in stacking the dies, placing each functional block, and identifying component variations, thus necessitating the need for a unified system language to address these challenges.

Samsung Foundry has developed 3DCODE, an exclusive 3DIC description language that helps manage the complexity of Multi-die Integration (MDI) designs. Compared to traditional GUI-based designs, 3DCODE offers:

- Simpler and more efficient script-based MDI designs.
- Consistency and ease of understanding across different levels of chip design and implementation. This means that packaging and OSAT engineers can use a common language across multiple 3DIC platforms.
- Script-based Design of Experiment (DoE), based on early analysis model, to handle 3DIC DB.

Note: Designers must create multiple DoEs to minimize design Turn-Around-Time (TAT) and overall cost.

1.1. Environment

1.1.1. Language

3DCODE uses the open source JavaScript Object Notation (JSON) format for seamless integration and multi-tool compatibility. To enable early-stage analysis, 3DCODE requires minimal design information such as the Verilog netlist, which includes the input and output ports. This provides the flexibility to assign 3DCODE variables related to physical information as optional properties.

Guide: JSON is a lightweight data interchange format based on a subset of the JavaScript Programming Language Standard, ECMA-262 Third Edition.

Note: For the latest updates on newly added tools and compatibility features, please contact the Samsung Foundry technical support team.

1.1.2. Tool Environment

3DCODE is compatible with the following 3DIC platform tools:

- 3DIC Compiler: U-2022.12-SP6-VAL-20230930 or later
- Integrity 3D-IC: 22.14_e056_1 or later

1.2. 3DCODE Structure

3DCODE provides a two-level hierarchy for operating the 3DIC platform. The code architecture has the following three key components:

1. 3dTop: Defines the netlist of the virtual top.
2. 3dBlock: Contains design information for each die and package.
3. 3dBuilder: Contains the die-to-die and die-to-package information.

The following sequence of events occur when building the 3DIC structure:

1. The 3dTop defines the virtual top netlist and calls the 3dBuilder and 3dBlock files.
2. The 3dBuilder calls the die and package instances based on the 3dBlock files.
3. The 3dBuilder connects each instance and builds the 3DIC structure.

Figure 1-1 shows the 3DCODE configuration.

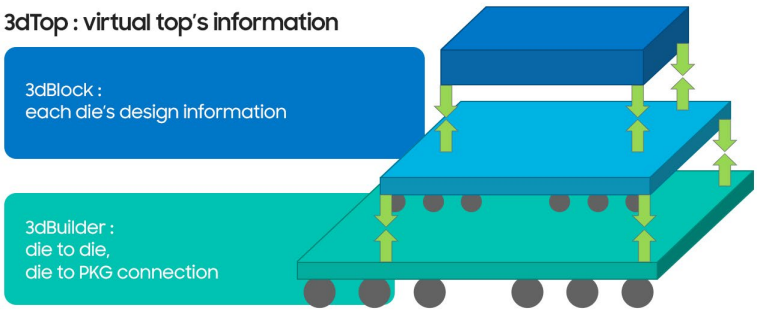

Figure 1-1. *3DCODE Configuration*

Figure 1-2 shows the 3DCODE script format. Chapter 3 describes the syntax and variable details of 3DCODE.

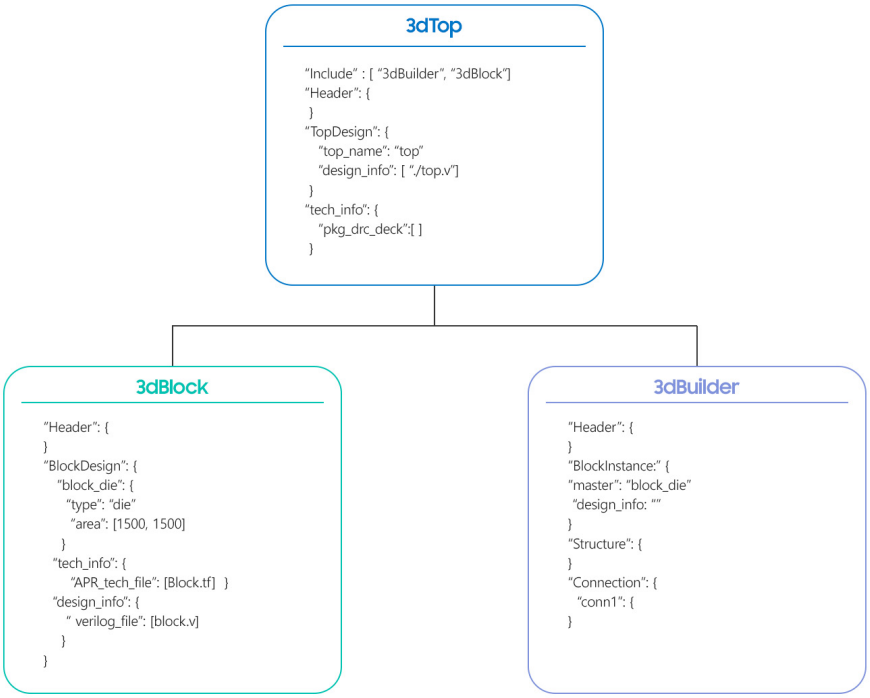

Figure 1-2. *3DCODE Structure with Pseudocode*

2. 3DCODE Operation

Figure 2-1 describes the 3DCODE workflow. 3DCODE covers all design processes from design planning to post-analysis. We have defined the design flow as design setup, implementation, and post-analysis. Each step is described in detail in each subchapter.

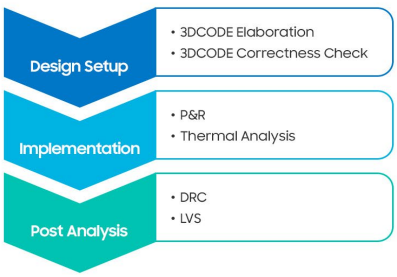

Figure 2-1. *3DCODE Workflow*

Figure 2-2 shows the process of the design setup stage with 3DCODE.

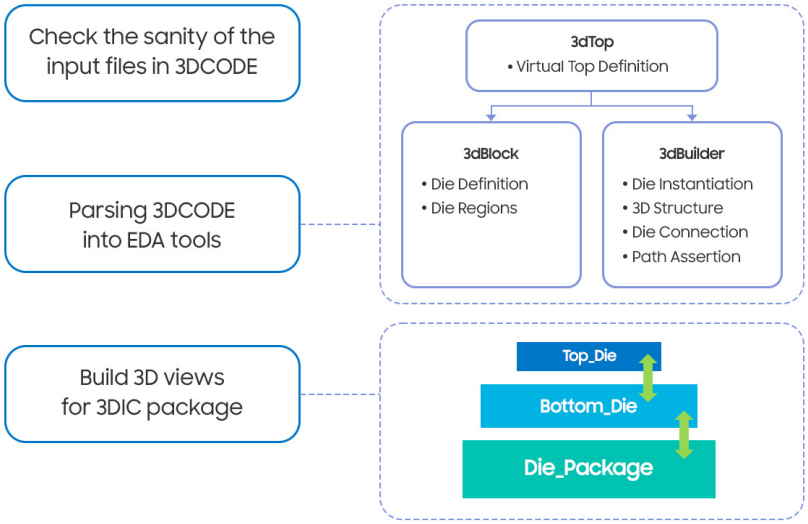

Figure 2-2. *Design Setup Stage Detail*

2.1. Design Setup Stage

In the design setup stage, the user collects information for implementation and checks for accuracy. The 3dBlock files contain each die information and I/O bump location information, however, they do not include GDS information and the ball map file. The 3dBuilder file defines the connection between the instance defined in the 3dBlock and the 3D structure of the design. These files can be used to create a simple 3D view of the design that represents the 3DIC package.

Early thermal analysis can also be performed at this stage. This helps designers create many DoEs for the 3DIC design in the early design phase. As a result, designers can minimize TAT and cost. Chapter 5 describes this in more detail.

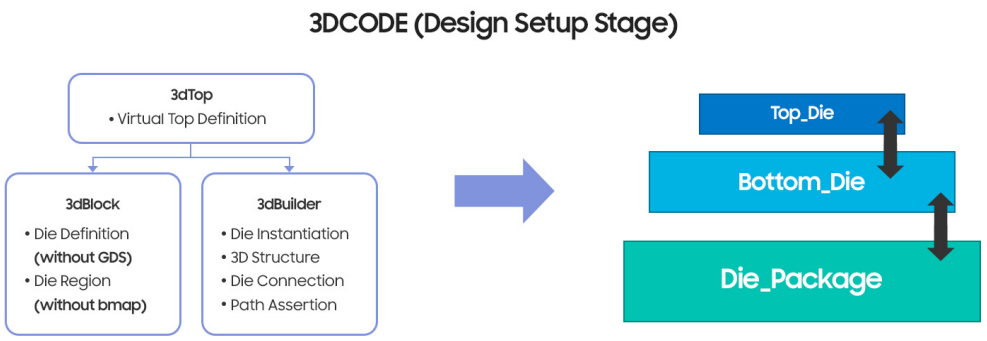

Figure 2-3. *Diagram of the Design Setup Stage*

2.2. Implementation Stage

Physical layout and timing information can be used in the implementation stage. Since the 3dBlock file contains a ball map file, a more accurate check can be performed based on the physical layout information.

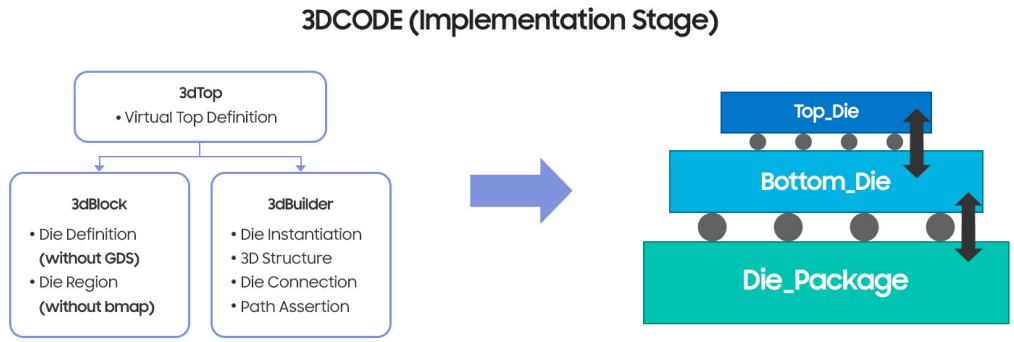

Figure 2-4. *Diagram of the Implementation Stage*

2.3. Post-Analysis Stage

In the post-analysis stage, DRC and LVS could be performed for the final sign-off. This step will be added in a later release. We plan to develop an efficient 3DIC DRC and LVS flow based on 3DCODE that describes the P&R information.

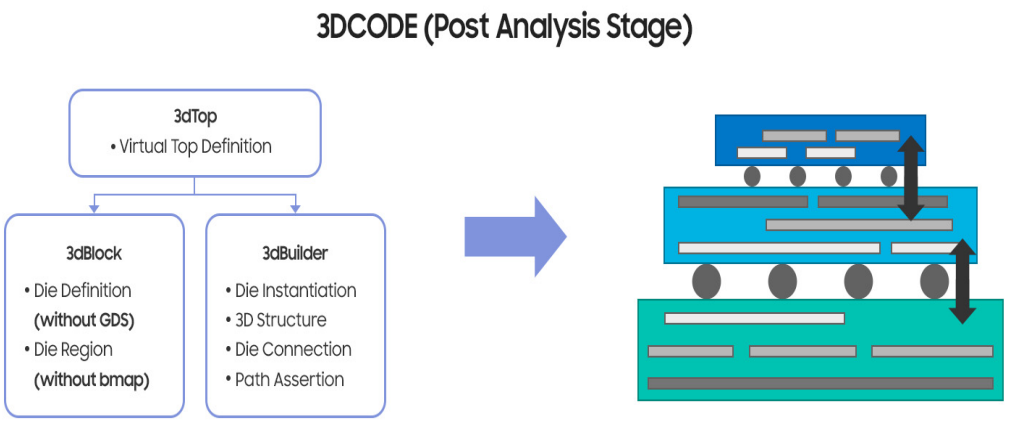

Figure 2-5. *Diagram of the Post-Analysis Stage*

3. 3DCODE Language Syntax and Variables

3.1. 3dTop

3.1.1. Syntax

3dTop is a virtual top that contains all 3dBlock files as well as a 3dBuilder file. Example 3-1 shows the syntax of 3dTop.

```
{
    "Include": ["*.3dBlock", "*.3dBuilder"],
    "Define": {"var1": "variable1", "var2": "variable2"},
    "Header": {
        "version": "version_number",
        "unit": "unit",
        "precision": "db_unit"
    },
    "TopDesign": {
        "top_name": "top_module_name",
        "design_info": {
            "verilog_file": ["/path/…/top/verilog"],
            "upf_file": ["/path/…/top/upf"]
        },
        "tech_info": {
            "pkg_lvs_deck": ["/path/…/pkg/lvs/lvs_deck"],
```

```
            "pkg_drc_deck": ["/path/…/pkg/drc/drc_deck"]
        }
    }
}
```

Example 3-1. *3dTop Syntax*

3.1.2. Variables

Table 3-1 shows the descriptions of the variables used in 3dTop.

	Variable Name	Optional / Mandatory	Stage	Description
	Header			"Header" defines the basic information used in 3DCODE.
1	version	Mandatory	DesignSetup	"version" is used to manage the version of 3DCODE.
2	unit	Mandatory	DesignSetup	"unit" means the standard of measurement and the default value is micro.
3	precision	Mandatory	DesignSetup	"precision" is a value used in the design database, such as a Design Exchange Format (DEF) file, and the default value is 2,000.
	TopDesign			"TopDesign" contains the information related to the top of the design.
1	top_name	Mandatory	DesignSetup	"top_name" requires the top module name of the design.
2	design_info			"design_info" in 3dTop means the design information related to the top module. An upf_file will be added in a later release.
	verilog_file	Mandatory	DesignSetup	"verilog_file" requires a Verilog file path of the top design.
	upf_file	Optional	DesignSetup	"upf_file" requires a UPF file path of the top design.
3	tech_info			"tech_info" in 3dTop means the process information associated with the top module. It has a DRC deck of the package design.
	pkg_lvs_deck	Optional	PostAnalysis	"pkg_lvs_deck" requires the package LVS rule deck file path.
	pkg_drc_deck	Optional	PostAnalysis	"pkg_drc_deck" requires the package design rule check file path.

Table 3-1. *Description of Variables in 3dTop*

3.2. 3dBlock

3.2.1. Syntax

3dBlock contains all information about each block. Example 3-2 shows the syntax of 3dBlock.

```
{
    "Header": {
        "version": "version_number",
```

```
        "unit": "unit",
        "precision": "db_unit"
    },
    "BlockDesign": {
        "block_module_name": {
            "type": "die" | "package",
            "area_info": [width, length],
            "guard_ring_width": [left_value, bot_value, right_value, top_value],
            "scribe_line_remaining_width": [left_value, bot_value, right_value, top_value],
            "thickness": thickness,
            "shrink": factor,
            "tsv": true | false,
            "regions": {
                    "region_name": {
                           "bmap": region's_bmap_file_path,
                           "side": "front" | "back",
                           "layer": "layer_name",
                           "gds_layer": "layer_name",
                           "coords": [[x, y], [x1, y1], ...]
                    }
            },
            "design_info": {
                    "verilog_file": ["/path/…/verilog_file"],
                    "sdc_file": ["/path/…/sdc_file"],
                    "def_file": ["/path/…/def_file"],
                    "gds_file": ["/path/…/gds_file"],
                    "package_design_mcm": ["/path/…/package/design/mcm_file"],
                    "package_design_odb": ["/path/…/package/design/odb_file"]
            },
            "package_info": {
                    "package_padstack_file": ["/path/…/package/padstack_file"],
                    "package_device_file": ["/path/…/package/device_file"],
                    "package_symbol_file": ["/path/…/package/symbol_file"],
                    "package_db_file": ["/path/…/package/db_file"],
                    "package_tech_file": ["/path/…/package/tech_file"]
            },
            "tech_info": {
                    "liberty_file": ["SF1*.lib", "SF2*.lib"],
                    "LEF_file": ["SF1*.lef", "SF2*.lef"],
                    "SPICE_model": ["/path/…/*.sp"],
                    "RC_tech_file": ["/path/…/RC_tech_file"],
                    "APR_tech_file": ["/path/…/apr_tech_file"],
                    "GDS_mapping_file": ["/path/…/gds_mapping_file"],
                    "DRC_deck": ["/path/…/drc/deck"],
                    "LVS_deck": ["/path/…/lvs/deck"]
            }
      }
```

```
    }
}
```

Example 3-2. *3dBlock Syntax*

3.2.2. Variables

Table 3-2 descrbes the variables used in 3dBlock.

	Variable Name	Optional / Mandatory	Step	Description
	BlockDesign			"BlockDesign" defines each block of the design.
	block_module_name			"block_module_name" requires the module name of the block design.
1	type	[Die] Mandatory [Pkg] Mandatory	DesignSetup	"type" has two options, die or package. Users can choose one depending on the block type.
2	area_info	[Die] Mandatory [Pkg] Mandatory	DesignSetup	"area_info" means the width and length of the block design area. It does not include physical area values such as guard rings and scribe lines.
3	guard_ring_width	[Die] Optional [Pkg] Optional	DesignSetup	"guard_ring_width" means the width of the guard ring extended from the DEF origin.
4	scribe_line_ remaining_width	[Die] Optional [Pkg] Optional	DesignSetup	"scribe_line_remaining_width" means the remaining width of the scribe line extended from the GDS origin.
5	thickness	[Die] Mandatory [Pkg] Mandatory	DesignSetup	"thickness" specifies the thickness of the block.
6	shrink	[Die] Optional [Pkg] Optional	DesignSetup	"shrink" is the optical shrink factor.
7	tsv	[Die] Mandatory [Pkg] Optional	DesignSetup	"tsv" indicates whether TSVs are present in the block.
8	**regions**			"regions" defines the I/O bump regions of the block.
	region_name			"region_name" requires the name of the I/O bump regions.
	bmap	[Die] Mandatory [Pkg] Mandatory	Implementation	"bmap" defines the bump map (.bmap) file path. The file has a specific six-column format as follows:Instance ref_name x y port_name net_name
	side	[Die] Mandatory [Pkg] Mandatory	DesignSetup	"side" defines the relative z-direction of the region to the block origin. If the regions of the block with face up is back, the z-coordinate of the region would be equal to the z-coordinate of the origin of the block. If the regions of the block with face up is front, the z-coordinate of the region would be equal to the z-coordinate of the origin plus the thickness of the block.
	layer	[Die] Mandatory [Pkg] Mandatory	DesignSetup	"layer" defines the contact layer responsible for any external connections.
	gds_layer	[Die] Optional [Pkg] Optional	PostAnalysis	"gds_layer" is optional. It is only required if there are additional layers in the GDS.
	coords	[Die] Mandatory [Pkg] Mandatory	DesignSetup	"coords" specifies the coordinates of the region.
9	**design_info**			"design_info" in 3dBlock is the design information associated with each block module. It has external collaterals required in 3DCODE.
	verilog_file	[Die] Mandatory [Pkg] Optional	DesignSetup	"verilog_file" requires the path to the .verilog files of the block design.
	sdc_file	[Die] Optional [Pkg] Optional	DesignSetup	"sdc_file" requires the path to the .sdc file of the block design.
	def_file	[Die] Optional [Pkg] Optional	Implementation	"def_file" requires the path to the .def file of the block design.
	gds_file	[Die] Optional [Pkg] Optional	PostAnalysis	"gds_file" requires the path to the block design's .gds file.
	package_ design_mcm	[Die] Optional [Pkg] Optional	Implementation	If users are using Cadence flow, they would define the "package_design_mcm". "package_design_mcm" requires the path of an .mcm file coming from the APD tool.
	package_ design_odb	[Die] Optional [Pkg] Optional	Implementation	If users are using other 3DIC platforms, they would define "package_design_odb". "package_design_odb" needs the path of an .odb++ file which is the open format to describe the package.

10	**package_info**			"package_info" is the package information.
	package_padstack_file	[Die] Optional [Pkg] Optional	DesignSetup	If type is the package, "package_padstack_file" needs the path to the pad stack file.
	package_device_file	[Die] Optional [Pkg] Optional	DesignSetup	If type is the package, "package_device_file" needs the path of the device file.
	package_symbol_file	[Die] Optional [Pkg] Optional	DesignSetup	If type is the package, "package_symbol_file" needs the path to the symbol file.
	package_db_file	[Die] Optional [Pkg] Optional	DesignSetup	If type is the package, "package_db_file" needs the path to the db file.
	package_tech_file	[Die] Optional [Pkg] Optional	DesignSetup	If type is the package, "package_tech_file" needs the path of the package tech file.
11	**tech_info**			"tech_info" in 3dBlock means the process information associated with each block module. It has external collateral required in 3DCODE.
	liberty_file	[Die] Optional [Pkg] Optional	DesignSetup	"liberty_file" requires the path to the Liberty files of the block design.
	LEF_file	[Die] Optional [Pkg] Optional	DesignSetup	"LEF_file" requires the path to the .lef files of the block design.
	SPICE_model	[Die] Optional [Pkg] Optional	Implementation	"SPICE_model" requires the path to the SPICE model file of the block design.
	RC_tech_file	[Die] Optional [Pkg] Optional	Implementation	"RC_tech_file" requires the path to the RC tech file of the block design.
	APR_tech_file	[Die] Optional [Pkg] Optional	Implementation	"APR_tech_file" requires the path to the APR tech file of the block design.
	GDS_mapping_file	[Die] Optional [Pkg] Optional	PostAnalysis	"GDS_mapping_file" requires the path to the block design's GDS mapping file.
	DRC_deck	[Die] Optional [Pkg] Optional	DesignSetup	"DRC_deck" requires the path to the DRC deck of the block design.
	LVS_deck	[Die] Optional [Pkg] Optional	DesignSetup	"LVS_deck" requires the path to the LVS deck of the block design.

Table 3-2. *Description of Variables in 3dBlock*

3.3. 3dBuilder

3.3.1. Syntax

3dBuilder describes the structure and connection between each block to create a 3DIC structure. Example 3-3 shows the syntax of a 3dBlock.

```
{
    "Header": {
        "version": "version_number",
        "unit": "unit",
        "precision": "db_unit"
    },
    "BlockInstance": {
        "instance_name": {
            "master": "block_module_name",
            "design_info": {
                 "sdc_file": ["/path/…/sdc/file"],
            }
```

```
        }
    },
    "Structure": {
        "instance_name": {
             "location": [x, y],
             "z": z,
             "rotation": "R0" | "R90" | "R180" | "R270" | "MZ" | "MZ_R90" | "MZ_R180" | "MZ_R270" |
"MZ_MX" | "MZ_MX_R90" | "MZ_MX_R180" | "MZ_MX_R270" | MZ_MY" | "MZ_MY_R90" | "MZ_MY_R180" |
"MZ_MY_R270" | "MX" | "MX_R90" | "MX_R180" | "MX_R270" | "MY" | "MY_R90" | "MY_R180" | "MY_R270",
             "root": "true" | "false"
        }
    },
    "Connection": {
        "connection_name": {
            "top": "top_instance.regions.region_name",
            "bot": "bottom_instance.regions.region_name",
            "thickness": "thickness",
            "LVS": "Interface_LVS_deck"
        }
    },
    "path": {
        "path_name": [
            "[NOT] instance_name.regions.region_name",
            ...
        ]
    }
}
```

Example 3-3. *3dBuilder Syntax*

3.3.2. Variables

Table 3-3 describes the variables used in 3dBuilder.

Variable Name	Optional / Mandatory	Step	Description
1 BlockInstance			"BlockInstance" specifies the block instance of the block design.
instance_name			"instance_name" requires the instance name of the block design.
master	Mandatory	DesignSetup	"master" is the referenced block design. The value of master is the block_module_name already defined in 3dBlock.
design_info			"design_info" in 3dBlock means the constraint files to be applied to the instance. The upf_file will be added in a later release.
sdc_file	[Die] Optional [Pkg] Optional	DesignSetup	"sdc_file" is the path to the .sdc file to be applied to the block instance.
2 Structure			"Structure" defines the structure of the package.
instance_name			"instance_name" requires the instance name of the block design.
location	Mandatory	DesignSetup	"location" specifies the physical location of the instance. The value of x, y means the coordinates of the lower-left corner of the block instance.
z	Mandatory	DesignSetup	"z" defines the physical height of the instance.
rotation	Mandatory	DesignSetup	"rotation" indicates the position of the instance with respect to a particular orientation. The value of this variable is one of the values shown in Figure 3-1. R0, R90, R180, and R270 represent counterclockwise rotation information. MX and MY refer to symmetry along the X-axis and Y-axis, respectively. MZ represents the reversing of the metal stack.
root	Mandatory	DesignSetup	"root" defines whether the instance has a role as the root instance of the structure.
3 Connection			"Connection" defines the connection information between adjacent blocks.
connection_name			"connection_name" takes the name of the connection.
top	Mandatory	DesignSetup	"top" defines the instance that is physically on top of the connection.
bot	Mandatory	DesignSetup	"bot" defines the instance that is physically below the connection.
thickness	Mandatory	DesignSetup	"thickness" defines the thickness of the connection.
LVS	Optional	Implementation	"LVS" defines the LVS deck for the connection.
4 Path			"path" is the path assertion.
path_name	Optional	DesignSetup	"upf_file" requires a UPF file path of the top design.

Table 3-3. *Description of Variables in 3dBuilder*

Figure 3-1 shows possible combinations of the rotation variable.

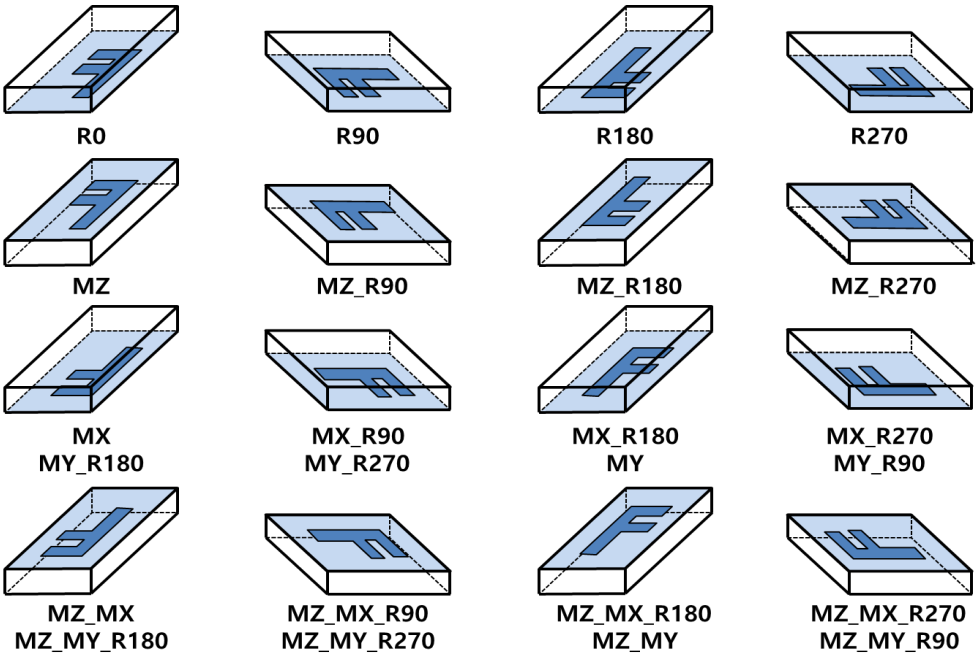

Figure 3-1. *Possible Combinations of the Rotation Variable*

4. Structure Description

We have prepared a 3DIC structure example using 3DCODE to enhance the understanding of 3DIC configuration. We assume that the virtual top name is "sfcore" and it is composed of 2 SoCs and 1 package die as described in Figure 4-1.

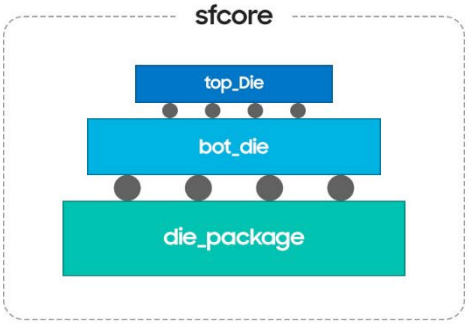

Figure 4-1. *Example 3DIC Structure*

The following shows 3DCODE examples of this structure. Example 4-1 describes 3dTop of sfcore, Example 4-2 is 3dBuilder, and Example 4-3 represents 3dBlock for die.

```
{
    "Include": [bottom_die.3dBlock, top_die.3dBlock, package.3dBlock, sfcore.3dBuilder]
    "Header": {
        "version": 1.0,
        "unit": "micron",
        "precision": 2000
    },
    "TopDesign": {
        "top_name": "top_die",
        "design_info": {
            "verilog_file": ["Collateral/riscv_top/design/riscv_top.v"],
            "upf_file": ["Collateral/riscv_top/design/riscv_top.upf"]
        },
        "tech_info": {
            "pkg_drc_deck": []
        },
    }
}
```

Example 4-1. *sfcore 3dTop*

```
{
      "Header": {
        "version": 1.0,
        "unit": "micron",
        "precision": 2000
    },
    "BlockInstance": {
        "ibottom": {
            "master": "bottom_die",
            "constraint": {
                "sdc_file": [],
                "upf_file": []
            }
        },
        "itop": {
            "master": "top_die",
            "constraint": {
                "sdc_file": [],
                "upf_file": []
            }
        },
        "ipackage": {
```

```
            "master": "die_package"
        }
    },
    "Structure": {
        "itop": {
            "location": [0.0, 0.0],
            "rotation": "MZ_MX",
            "z": 200.0,
            "root": false
        },
        "ibottom": {
            "location": [0.0, 0.0],
            "rotation": "R0",
            "z": 100.0,
            "root": false
        },
        "ipackage": {
            "location": [0.0, 0.0],
            "rotation": "R0",
            "z": 0.0,
            "root": true
        }
    },

    "Connection": {
        "Conn1": {
            "top": "itop.regions.r1",
            "bot": "ibottom.regions.r1",
            "LVS": null,
            "thickness": 0.0
        },
        "Conn2": {
            "top": "ibottom.regions.r2",
            "bot": "ipackage.regions.r1",
            "LVS": null,
            "thickness": 0.0
        }
    },

    "Path": {
        "Path1": [
            "itop.regions.r1",
            "ibottom.regions.r1",
            "ibottom.regions.r2"
            "ipackage.regions.r1"
        ]
    }
```

```
}
```

Example 4-2. *sfcore 3dBuilder*

```
{
    "Header": {
        "version": 1.0,
        "unit": "micron",
        "precision": 2000
    },
    "BlockDesign": {
        "top_die": {
            "type": "die",
            "area_info": [320.0, 320.0],
            "thickness": 100.0,
            "guard_ring_width": [0, 0, 0, 0],
            "scribe_line_remaining_width": [0, 0, 0, 0],
            "shrink": 1.0,
            "tsv": false,
            "regions": {
                "r1": {
                    "layer": "Metal9",
                    "side": "front",
                    "coords": [
                        [0.0, 0.0],
                        [320.0, 0.0],
                        [320.0, 320.0],
                        [0.0, 320.0]
                    ],
                    "bmap": "Collateral/top_die/design/top_die_r1.bmap"
                }
            },
            "tech_info": {
                "APR_tech_file": ["Collateral/top_die/tech/*.lef"],
                "LEF_file": ["Collateral/top_die/lib/*.lef"]
            },

            "design_info": {
                "verilog_file": ["Collateral/top_die/design/top_die.v"],
                "def_file": ["Collateral/top_die/design/top_die.def"],
                "upf_file": ["Collateral/top_die/design/top_die.upf"],
                "sdc_file": [],
                "gds_file": []
            }
        }
    }
```

```
}
```

Example 4-3. *sfcore 3dBlock for a Die*

Example 4-4 shows a 3dBlock file for the package die, which is a component of the 3DIC. 3dBuilder calls a package die as an instance of the 3DIC structure.

```
{
      "Header": {
        "version": 1.0,
        "unit": "micron",
        "precision": 2000
    },
    "BlockDesign": {
        "die_package": {
            "type": "package",
            "area_info": [320.0, 320.0],
            "thickness": 100.0,
            "regions": {
                "r1": {
                    "layer": "Metal9",
                    "side": "front",
                    "coords": [
                        [0.0, 0.0],
                        [320.0, 0.0],
                        [320.0, 320.0],
                        [0.0, 320.0]
                    ],
                    "bmap": "Collateral/package/design/package_r1.bmap"
                }
            },
            "tech_info": {
                "package_tech_file" : ["Collateral/package/tech/*.apd"]
            },
            "design_info": {
                "package_design_mcm" : ["Collateral/package/design/*.mcm ]
            }
        }
    }
}
```

Example 4-4. *sfcore 3dBlock for a package*

5. Early Analysis with 3DCODE

To reduce the TAT of MDI design, users need to run multiple test cases to find a better starting point. 3DCODE can help reduce the TAT of these DoEs through early analysis functions. .

The design TAT of MDI increases compared to 2D design due to the expanded search space for package type and additional considerations such as thermal and mechanical stress. This increased design TAT reduces the market responsiveness of the product and increases cost. One way to overcome this is to find a good starting point and start from there. In the current MDI design flow, the only way to find a good starting point is to perform DoEs to get the best values. 3DCODE can greatly assist in performing and automating early DoEs. Users can easily generate different physical MDI structures on each EDA MDI platform using 3DCODE. The generated structures are connected to analysis tools within the 3DCODE-based MDI platform to generate results. All of this work can be done on one platform using pre-described scripts and can help users reduce DoE TAT. This chapter describes how to generate the physical structure for early-level analysis on each EDA platform with 3DCODE and how to combine the power and thermal models of each block.

5.1. Early Thermal Analysis Flow with 3DCODE

3DCODE allows the user to perform early thermal analysis more efficiently. Figure 5-1 shows an example of early analysis with 3DCODE. Not only can the user create a 3DIC design test case with 3DCODE, but they can also analyze it with combined analysis tools. And this flow makes 3DIC experiments more efficient.

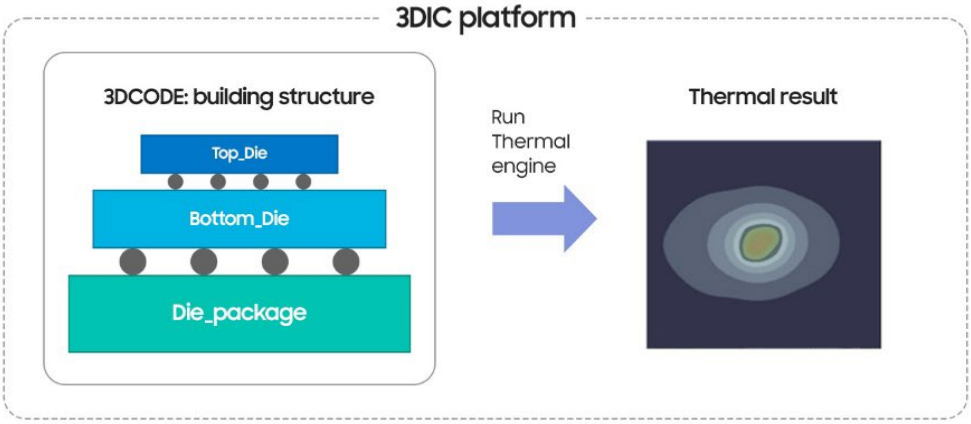

Figure 5-1. *Early Thermal Analysis with 3DCODE*

5.1.1. Early Thermal Analysis with 3DIC Compiler

Figure 5-2 describes the thermal analysis flow with the 3DIC Compiler. 3DCODE can connect this flow in the 3DIC Compiler. Currently, 3DCODE is used for test design. In a prototype design using 3DCODE, system modeling, including package design, can be performed in the 3DIC Compiler if required. The 3DIC Compiler supports two thermal engines: Ansys' RedHawk-SC-ET (RHSC-ET) and Synopsys's native thermal engine. The user must input the appropriate collaterals to the engine, and thermal analysis can be performed using a separate power scenario for both transient and static thermal analysis for the power map. More detailed information on thermal analysis commands can be found in the latest version of the Samsung_Foundry_Advanced_3DIC_PnR_Methodology_3DICCompiler_ApplicationNote. This document can be downloaded from Samsung Foundry's connect system.

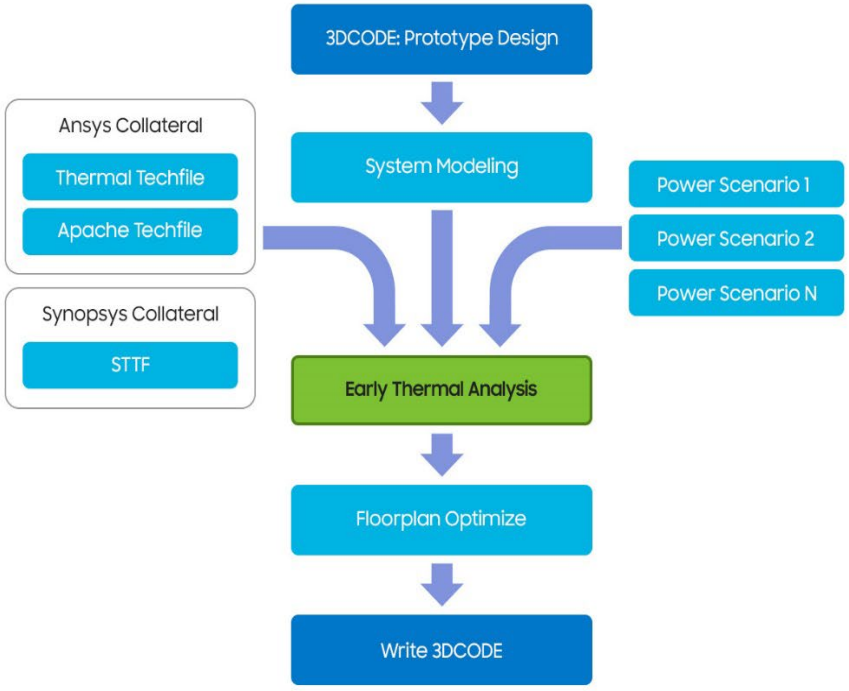

Figure 5-2. *Early Thermal Analysis with the 3DIC Compiler*

The power map file is a format for specifying the location and average power value for each power source as described in Example 5-1, and a power map should be prepared for each top and bottom die and used for thermal analysis.

```
BBOX  -100 -100 12100 12100
#name  llx   lly   urx   ury power(mW)
Block1 11319.96800  9489.96000  11751.58400  9893.88000 170
Block2 11319.96800  8893.80000  11751.58400  9297.72000 170
Block3 11319.96800  8297.64000  11751.58400  8701.56000 170
Block4 11319.96800  3431.16000  11751.58400  3835.08000 170
```

Example 5-1. *Multi-Heat Source Power Map*

5.1.1.1. Example (RHSC-ET)

```
read_3dcode -lib ./work/top ltvf_top.3dTop
```

Example 5-2. *Read 3DCODE*

```
# power map
set power_dir ./data/power/converted
#set power_dir ./data/power
set pow1 "u_top ${power_dir}/power.top_power_map.mhs"
set pow2 "u_bottom ${power_dir}/power.bot_power_map.mhs"
set powers ""
lappend powers $pow1 $pow2
set_app_options -name 3dic.thermal.power_maps -value $powers

set DPM "DPM"
set dpm1 "virtual_top_chip.ndm:virtual_top_chip 50um data/apache/virtual_top_chip.tech
data/metal/from_ECO/virtual_top_chip.csv"
set dpm2 "virtual_bottom_top.ndm:virtual_bottom_top 50um data/apache/virtual_bottom_top.tech
data/metal/from_ECO/virtual_bottom_top.csv"
set DPM_set {}
lappend DPM_set $dpm1 $dpm2
lappend DPM $DPM_set
# thermal tech info
set ThermalTechAttrs "ThermalTech"
set TTF "File data/thermal_tf/general.xml"
# die mapping between 3DIC and Thermal tech
set DieMap "Map"
set tf1 {virtual_top_chip.ndm:virtual_top_chip VIRTUAL_TOP_CHIP}
set tf2 {virtual_bottom_top.ndm:virtual_bottom_top VIRTUAL_BOTTOM_TOP}
set Maps {}
lappend Maps $tf1 $tf2
lappend DieMap $Maps
lappend ThermalTechAttrs [ list $TTF $DieMap ]
set DPM_Thermal {}
lappend DPM_Thermal $DPM
```

```
lappend DPM_Thermal $ThermalTechAttrs
set_app_options -name 3dic.thermal.density_power_model -value $DPM_Thermal

set_user_units -type time -value 1
set_thermal_transient_sequence -stage_durations {{stage_1 2 0.05} {stage_2 3 0.05} {stage_3 2 0.05}}

set POW_DIR ./data/power/converted
# stage_1
set pow1_st1 "u_top ${POW_DIR}/top_power_map_stage1.mhs"
#set pow2_st1 "u_bottom   ${POW_DIR}/power.bot_power_map.mhs"
set pow2_st1 "u_bottom   ${POW_DIR}/bot_power_map_stage1.mhs"
set powers ""
lappend powers $pow1_st1 $pow2_st1
set_thermal_transient_stage -name stage_1 -inst_power_profiles $powers

# stage_2

set pow1_st2 "u_top ${POW_DIR}/power.top_power_map.mhs"
set pow2_st2 "u_bottom   ${POW_DIR}/power.bot_power_map.mhs"
set powers ""
lappend powers $pow1_st2 $pow2_st2
set_thermal_transient_stage -name stage_2 -inst_power_profiles $powers

set pow1_st3 "u_top ${POW_DIR}/top_power_map_stage3.mhs"
set pow2_st3 "u_bottom   ${POW_DIR}/bot_power_map_stage3.mhs"
set powers ""
lappend powers $pow1_st3 $pow2_st3
set_thermal_transient_stage -name stage_3 -inst_power_profiles $powers
```

Example 5-3. *Collateral Setting for RHSC-ET Run*

```
set_app_options -name 3dic.thermal.engine -value fusion
set_host_options -max_cores $MAXCORE
set_app_options -name 3dic.common.redhawk_sc_et_path -value $REDHAWK_SC_ET_PATH
set_app_options -name 3dic.thermal.bump_via_data_model -value Solid
set_thermal_grid -inst_name u_top -top 0 \
  -user_grids { { {{14.04000 14.04000} {9985.96000 9985.95000}} 100} }
set_thermal_grid -inst_name u_bottom -top 0 \
  -user_grids { { {{-46.44000 -46.44000} {12046.44000 12046.4300}} 100} }
analyze_3d_thermal -type static -result $result
report_thermal -verbose -result $result
gui_zoom -window [gui_get_current_window -view] -full
gui_show_map -window [gui_get_current_window -types Layout -mru] -map {3dic_thermal} -show {true}
```

Example 5-4. *Script to Perform Early Thermal Analysis*

5.1.1.2. Example (Synopsys Native Engine)

In the Synopsys 3DIC Compiler, the Ansys RHSC-ET-based thermal analysis engine can be invoked and used on the in-design platform. In addition, the 3DIC Compiler can perform thermal analysis using Synopsys' native thermal analysis engine. Using the two engines requires the user to change only a few options. The user can use the native flow by setting the 3dic.thermal.engine app option to built-in.

```
set_app_options -name 3dic.thermal.engine -value built-in
set_app_options -name 3dic.thermal.mode -value fast
```

Example 5-5. *App Option Settings for Using the Native Thermal Analysis Engine*

```
set DPM “DPM”
set dpm1 "virtual_top_chip_hpdf_design_planning_final.nlib:virtual_top_chip 50um data/itf/virtual_top_chip.itf
data/metal/ndm/virtual_top_chip.profile"
set dpm2 "virtual_bottom_top.nlib:virtual_bottom_top 50um data/itf/virtual_bottom_top.itf
data/metal/ndm/virtual_bottom_top.profile"
set DPM_set {}
lappend DPM_set $dpm1 $dpm2
lappend DPM $DPM_set
set DPM_Thermal {}
lappend DPM_Thermal $DPM
set_app_options -name 3dic.thermal.density_power_model -value $DPM_Thermal
```

Example 5-6. *Script to Set Density Power Model Flow*

And for execution, there is a single command ‘analyze_3d_thermal’ as described in Example 5-7.

```
analyze_3d_thermal -type static -result $result
```

Example 5-7. *Early 3D Thermal Analysis Command*

5.1.1.3. Example Results

Figure 5-3 and Figure 5-4 show thermal analysis results using RHSC-ET and Synopsys Native thermal engine, respectively.

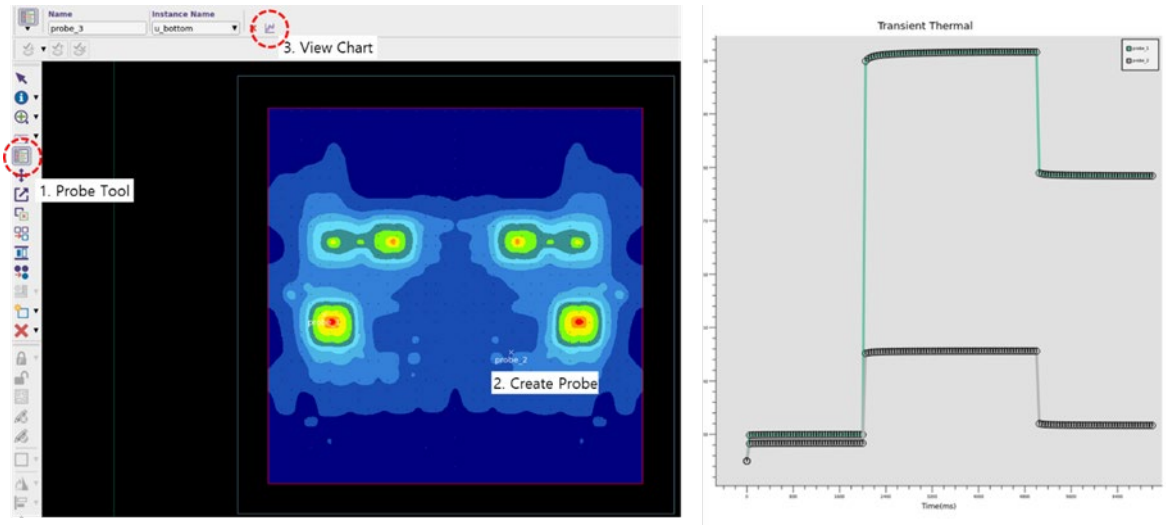

Figure 5-3. *Transient Early Thermal Analysis Results Using RHSC-ET*

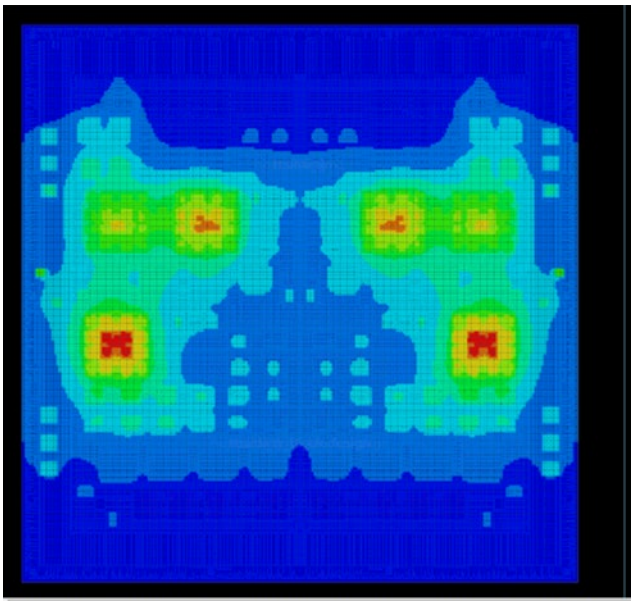

Figure 5-4. *Early Thermal Analysis Results Using Synopsys Native Thermal Engine*

5.1.2. Early Thermal Analysis with Integrity 3D-IC

When designing a 3DIC, thermal must be considered due to the structural limitations of the 3DIC. The early thermal analysis flow allows the user to freely place the necessary power blocks. Then, the user can check the estimated thermal effects based on the placed power areas. This helps the user to optimize the placement of power blocks in the early design stage. Figure 5-5 shows the early thermal analysis flow with 3DCODE. The user can set up a 3DIC design structure by reading 3DCODE files in Integrity 3D-IC. After the system planning is

complete, the user can set the early thermal analysis settings, and then perform the early static thermal analysis by invoking Celsius, which is the thermal analysis tool.

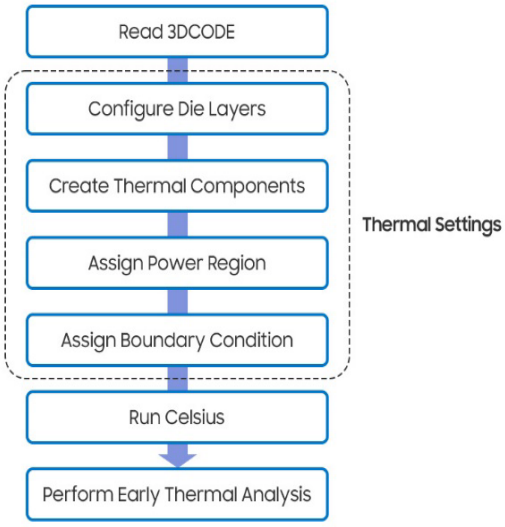

Figure 5-5. *Early Thermal Analysis with Integrity 3D-IC*

5.1.2.1. Example

NOTE: All commands are System Planner commands and should be executed with "planner::" or "eval_planner { }".

Before performing the early thermal analysis, 3DCODE files should be loaded into the Integrity 3D-IC platform. Example 5-8 shows how to load 3DCODE files into the Integrity 3D-IC.

```
planner::read_3dcode -files ./3dcode/virtual_top.3dTop
```

Example 5-8. *Reading 3DCODE into Integrity 3D-IC*

To set die thickness and metal density configurations, see Example 5-9.

```
## Bottom Die
planner::set_db BLK_BOT:LV .thickness 0.0 -category layer
```

```
planner::set_db BLK_BOT:DV .thickness 0.0 -category layer
planner::set_db BLK_BOT:BM1 .thickness 5.4 -category layer
planner::set_db BLK_BOT:TSV .thickness 42.5 -category layer

planner::thermal::set_layer_attribute BLK_BOT:LV -metal_percentage 0
planner::thermal::set_layer_attribute BLK_BOT:DV -metal_percentage 0
planner::thermal::set_layer_attribute BLK_BOT:BM1 -metal_percentage 60
planner::thermal::set_layer_attribute BLK_BOT:TSV -metal_percentage 25

## TOP Die
planner::set_db BLK_TOP:LV .thickness 0.0 -category layer
planner::set_db BLK_TOP:DV .thickness 0.0 -category layer
planner::set_db BLK_TOP:BM1 .thickness 5.4 -category layer
planner::set_db BLK_TOP:TSV .thickness 42.5 -category layer

planner::thermal::set_layer_attribute BLK_TOP:LV -metal_percentage 0
planner::thermal::set_layer_attribute BLK_TOP:DV -metal_percentage 0
planner::thermal::set_layer_attribute BLK_TOP:BM1 -metal_percentage 60
planner::thermal::set_layer_attribute BLK_TOP:TSV -metal_percentage 25
```

Example 5-9. *Configuring Die Layers*

To create thermal components for early thermal analysis, please refer to Example 5-10. The cross-section of the overall structure of the 3D stacked configuration in thermal flow is illustrated in Figure 5-6.

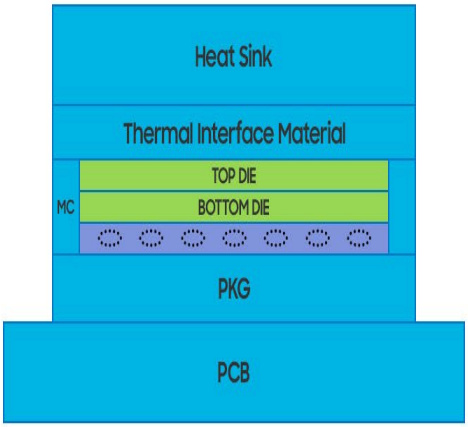

Figure 5-6. *Cross-section of 3D Stacked Configuration*

```
## create thermal components
## PKG
planner::thermal::create_component -name PKG -size "16.5mm 16.5mm 50um" -type package -loc "6495.3um 6476.9um
-64um" -origin center -layers "l1 l2 l3 l4 l5 l6 l7 l8 l9 l10" -layer_thickness "5um 5um 5um 5um 5um 5um 5um 5um 5um
5um" \
-layer_metal_density "80 20 80 20 80 20 80 20 80 20" -layer_metal_material "copper copper copper copper copper
copper copper copper copper copper" -layer_dielectric_material "FR4 FR4 FR4 FR4 FR4 FR4 FR4 FR4 FR4 FR4"

## PCB
planner::thermal::create_component -name PCB -size "50mm 50mm 1.6mm" -type PCB \
-loc "6495.3um 6476.9um -1755um" -origin center \
-layers "l1 l2" -layer_thickness "70um 1530um" \
-layer_metal_density "40 0" -layer_metal_material "copper copper" -layer_dielectric_material "FR4 FR4"
## MC
planner::thermal::create_component -name MC -size "16.5mm 16.5mm 250um" -type molding \
-loc "6495.3um 6476.9um -14um" -origin center \
-layers "l1" -layer_thickness "250um" \
-layer_dielectric_material MoldingCompound

## TIM
planner::thermal::create_component -name TIM -size "16.5mm 16.5mm 250um" -type tim \
-loc "6495.3um 6476.9um 236um" -origin center \
-layers "l1" -layer_thickness "0.25mm" -layer_metal_material copper \
-layer_dielectric_material HeatSinkAttach

## HS
planner::thermal::create_component -name heat_sink -size "16.5mm 16.5mm 1.0mm" -type heat_sink \
-loc "6495.3um 6476.9um 486um" -origin center \
-layers "l1" -layer_thickness "1.0mm" -layer_metal_density "100" -layer_metal_material copper
```

Example 5-10. *Creating Thermal Components*

Example 5-11 shows how to create the connection between cells after all cells have been defined.

```
##set contact layer between top die and bottom die
planner::set_thermal_contact_attribute -to_device BLK_BOT -to_layer LB -from_device BLK_TOP -from_layer BM1 -
pin_height 14 -pin_material COPPER -pin_underfill_material BT_EPOXY

## set contac layer between bottom die and pkg
planner::create_contact_layer -top_device BLK_BOT -top_layer LB -bottom_device PKG -bottom_layer l10 -cell {} -
cell_type pin -pin_height 14um -pin_material COPPER -pin_underfill_material BT_EPOXY

## set contact layer between pkg and pcb
planner::create_contact_layer -top_device PKG -top_layer l1 -bottom_device PCB -bottom_layer l2 -cell {} -cell_type pin -
pin_height 70um -pin_material COPPER -pin_underfill_material BT_EPOXY

## set contact layer between mc and pkg
```

```
planner::create_contact_layer -top_device PKG -top_layer l10 -bottom_device MC -bottom_layer l1 -cell {} -cell_type pin -pin_height 0um -pin_material COPPER -pin_underfill_material BT_EPOXY

## set contact layer between tim and mc
planner::create_contact_layer -top_device MC -top_layer l1 -bottom_device TIM -bottom_layer l1 -cell {} -cell_type pin -pin_height 0um -pin_material COPPER -pin_underfill_material BT_EPOXY

## set contact layer between tim and hs
planner::create_contact_layer -top_device TIM -top_layer l1 -bottom_device heat_sink -bottom_layer l1 -cell {} -cell_type pin -pin_height 0um -pin_material COPPER -pin_underfill_material BT_EPOXY
```

Example 5-11. *Creating Contact Layers*

In this early thermal analysis example, the power input should be defined for each block. See Example 5-12.

```
set bot_die_sub "BLK_BOT"
set bot_die_temp "BLK_BOT"
set bot_block_inst {BLK_STG_BOT_0 BLK_STG_BOT_1 BLK_STG_BOT_2 BLK_STG_BOT_3 BLK_STG_BOT_4 BLK_STG_BOT_5 BLK_STG_BOT_6 BLK_STG_BOT_7 BLK_STG_BOT_8 BLK_STG_BOT_9 BLK_STG_BOT_10 BLK_STG_BOT_11 BLK_STG_BOT_12 BLK_STG_BOT_13 BLK_STG_BOT_14 BLK_STG_BOT_15 BLK_MIF_1 BLK_MIF_0 BLK_LOGIC_BOT BLK_SRAM_BOT BLK_CPUCL_BOT BLK_D2D_BOT}
set bot_powerlist [list  0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 2.0 2.0 1.0 0.4 2.0 1.1]
set bot_power_layer "M1"
set bot_pg_net "VDD VSS"

set top_die_sub "BLK_TOP"
set top_die_temp "BLK_TOP"
set top_block_inst {BLK_STG0 BLK_STG1 BLK_STG2 BLK_STG3 BLK_STG4 BLK_STG5 BLK_STG6 BLK_STG7 BLK_STG8 BLK_STG9 BLK_STG10 BLK_STG11 BLK_STG12 BLK_STG13 BLK_STG14 BLK_STG15 BLK_SRAM BLK_LOGIC BLK_CPUCL BLK_D2D}
set top_powerlist [list 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.4 1.0 8.0 1.1]
set top_power_layer "M1"
set top_pg_net "VDD VSS"

planner::delete_current_region -all -cell $bot_die_sub
planner::delete_current_region -all -cell $top_die_sub

proc assignPower {die_sub die_temp block_inst powerlist power_layer pg_net} {
set region ""
foreach power $powerlist inst  $block_inst {
        set region [string map {\{ "" \} "" } [planner::get_db $die_sub:$die_temp:$inst -category cell_inst .bbox]]
        planner::create_current_region -cell $die_temp -layer $power_layer -name "region_$inst" -net $pg_net -region $region -power $power
}

planner::set_thermal_power $die_sub -type region_power
```

```
}

assignPower $bot_die_sub $bot_die_temp $bot_block_inst $bot_powerlist $bot_power_layer $bot_pg_net
assignPower $top_die_sub $top_die_temp $top_block_inst $top_powerlist $top_power_layer $top_pg_net

# specify power layer
planner::thermal::set_layer_attribute $bot_die_sub:M1 -is_power_layer
planner::thermal::set_layer_attribute $top_die_sub:M1 -is_power_layer
```

Example 5-12. *Assign Power Region*

Example 5-13 shows the final step in thermal modeling, assigning the thermal boundary condition.

```
planner::thermal::set_boundary_condition -device_path heat_sink -htc 1e8 -location top -ref_temp 25 -type htc
```

Example 5-13. *Assigning a Boundary Condition*

To perform the early thermal analysis and run Celsius, see Example 5-14.

```
planner::run_thermal -run_dir Run_Dir -tag "Thermal" -type run
```

Example 5-14. *Run Celsius*

5.1.2.2. Example Results

Figure 5-7 shows thermal analysis results using Integrity 3D-IC and Celsius. The figure on the right shows the temperature map for the bottom die.

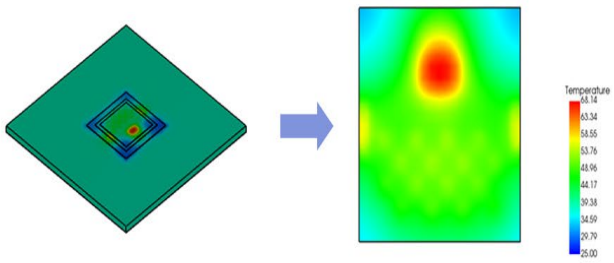

Figure 5-7. *Transient Early Thermal Analysis Results*

5.2. Early Static IR Drop Analysis Flow with 3DCODE

5.2.1. Early Static IR Drop Analysis with 3DIC Compiler

In the Synopsys 3DIC Compiler, after prototyping 3DIC design with 3DCODE, the user can perform a simple PDN and TSV placement, assign power to each block, and then proceed with early EM/IR analysis. Power analysis is performed by invoking the verified Ansys RedHawk-SC in-design.

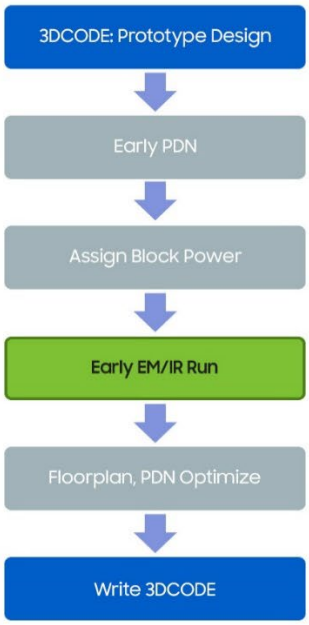

Figure 5-8. *Early Static IR Drop Analysis*

5.2.1.1. Example Run Script

Example 5-15 describes scripts for drawing early PDN and TSV placement. The user can refer to the EDA vendor manual page for more information and also to the Samsung 3DIC Design Methodology document, Samsung_Foundry_Advanced_3DIC_PnR_Methodology_3DICCompiler_ApplicationNote. This document can be downloaded from Samsung Foundry's connect system. Example 5-16 shows the setting to invoke Ansys' RedHawk-SC in the 3DIC Compiler. Please set the power information as shown in Example 5-17 and perform EM/IR analysis with a single command as shown in Example 5-18.

```
create_3d_template_objects -dies BotDie -types tsv -coordinates {{ 1007  1000} { 5992.96 10971.91 }}  -
coordinate_system local
set_3d_template  -pg_constraints {{ D10 0.04 18.8  v 6.8 } { D10 0.04 18.8  v 7.0 }  { D10 0.04 18.8  v 8.56 } { D10 0.04
18.8  v 8.76 } { D10 0.04 18.8  v 10.32 } { D10 0.04 18.8  v 10.52 } { D10 0.04 18.8  v 12.08 } { D10 0.04 18.8  v 12.28 }
{ D11 0.4 72 h 67.999 } { D11 0.4 72 h 73.319} { D11 0.4 72 h 77.299}   { D11 0.4 72 h 82.619}  { D11 0.4 72 h 87.939}
{ D11 0.4 72 h 93.259}  { D11 0.4 72 h 98.579}  { D11 0.4 72 h 102.999} { D11 0.4 72 h 108.319} { D11 0.4 72 h 113.299}
{ D11 0.4 72 h 118.619} { D11 0.4 72 h 123.939} { D11 0.4 72 h 129.259} { D11 0.4 72 h 134.579} } -net_pattern {  VDD }
create_3d_template_objects -dies BotDie -types { pg }

set_3d_template  -pg_constraints {{ D10 0.04 18.8  v 7.68 } { D10 0.04 18.8  v 7.88 } { D10 0.04 18.8  v 9.44 } { D10 0.04
18.8  v 9.64 } { D10 0.04 18.8  v 11.20 } { D10 0.04 18.8  v 11.40 } { D10 0.04 18.8  v  12.96 } { D10 0.04 18.8  v 13.16 }
{ D11 0.4 72 h 70.659 } { D11 0.4 72 h 75.979 } { D11 0.4 72 h 79.959 } { D11 0.4 72 h 85.279 } { D11 0.4 72 h 90.599 }
{ D11 0.4 72 h 95.919 } { D11 0.4 72 h 101.239 } { D11 0.4 72 h 105.659 } { D11 0.4 72 h 110.979 } { D11 0.4 72 h
115.959 } { D11 0.4 72 h 121.279 } { D11 0.4 72 h 126.599 } { D11 0.4 72 h 131.919 } { D11 0.4 72 h 137.239 }} -
net_pattern {  VSS }
create_3d_template_objects -dies BotDie -types { pg }  -pg_via_layers { D11 G1 }
```

Example 5-15. *TSV Placement and PDN Script*

```
set_host_options -submit_protocol lsf -submit_command {bsub -app batch  -n 2 -R "rusage[mem=50G]" }
set_app_options -name rail.enable_redhawk -value false
set_app_options -name rail.enable_redhawk_sc -value true
#set_app_options -name rail.database -value EMIR_exp1
set_app_options -name rail.dump_icc2_results_style -value 2.0
set_app_options -name rail.database -value EMIR_exp1
set_app_options -name rail.3dic_enable_single_phase_flow  -value True
set_app_options -name rail.redhawk_path -value
"/user/redhawk/update/RedHawk_SC/BUILDS/2024_R1.1/2024_R1.1/linux_x86_64_rhel7/bin"
set_app_options -name rail.enable_3dic -value True
set_app_options -name rail.tech_file -value {*
{ ./ln05lpe_13M_4Mx_7Dx_1Gx_1lz_LB_SigRCmaxDP_ErPlus_detailed_VERTICAL.tech }  }
set_app_options -name rail.temperature -value {25}
set_app_options -name rail.supply_net_voltage -value {VDD 1  VSS 0.0}

set_app_options -name rail.allow_redhawk_license_checkout -value 1
```

Example 5-16. *Script for Setting RHSC-ET Options*

```
set_block_power -block TopDie -net TopDie/VDD -current 30
set_block_power -block BotDie -net BotDie/VDD -current 40
create_taps -of_objects [get_pseudo_bumps   BotDie/bump_region_front_*] -honor_non_pg_pin
```

Example 5-17. *Script for Set Block Power*

```
analyze_3d_rail -nets  { VDD VSS } -voltage_drop static
```

Example 5-18. *Script to Perform Early Static IR Drop Analysis*

5.2.1.2. Example Results

Figure 5-9 shows the early static IR drop analysis results using RHSC in the 3DIC Compiler. The results show the hot spot map after power analysis.

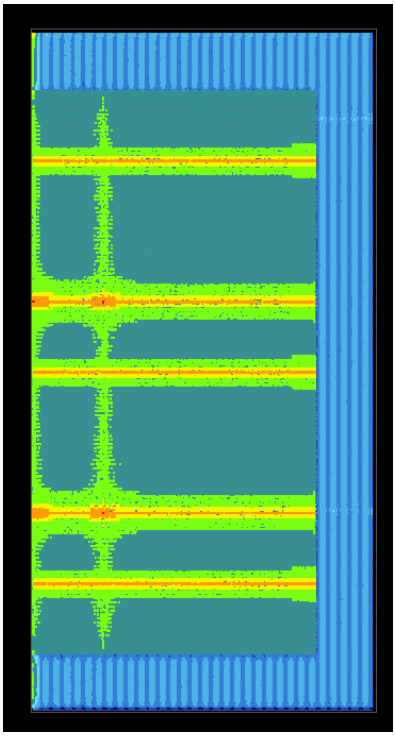

Figure 5-9. *Example of Early Static IR Drop Analysis Results*

5.2.2. Early Static IR Drop Analysis with Integrity 3D-IC

The user can set up a 3DIC design structure by reading the 3DCODE files. After system planning is complete, the user can set the early static IR drop analysis settings, create PG DEF using PG Structure Description Language (PSDL) files, and then assign the PG defs. to the early static IR drop analysis flow. Early static IR drop analysis is performed by invoking Voltus, which is the early static IR drop analysis tool.

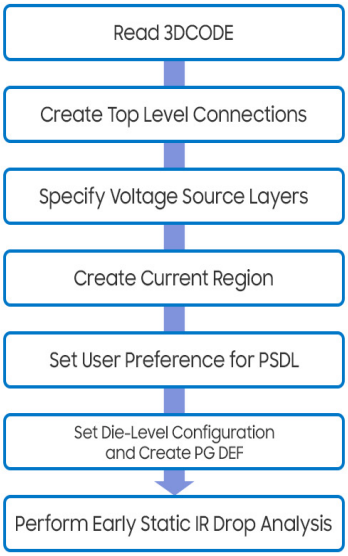

Figure 5-10. *Early Static IR Drop Analysis in Integrity 3D-IC*

The PSDL provides a comprehensive overview of the entire structure of the PG net. It is a set of statements consisting mainly of PATTERN and METAL statements. PATTERN statements describe the attributes and dimensions of metal patterns. METAL statements specify how these patterns are generated and placed within the specified area defined by REGION statements. The user can refer to the EDA vendor manual page on PSDL for more information. Example 5-19 shows a sample script for PSDL.

```
{ PATTERN M1_followpin { TYPE FOLLOWPIN } {WIDTH 0.038} }
{ PATTERN M2_stripe {TYPE STRIPE} {DIRECTION VERTICAL} {WIDTH 0.020} }
{ PATTERN M3_stripe {TYPE STRIPE} {DIRECTION HORIZONTAL} {WIDTH 0.020}}
{ REGION
        { COREAREA }

        { LAYER M1
                { METAL m1_vdd M1_followpin { NET VDD } }
                { METAL m1_vss M1_followpin { NET VSS } }
        }
        { LAYER M2 { SNAPTO GRID }
                { METAL m2_vdd M2_stripe {NET VDD} {OFFSET 0.026 *} {STEPDISTANCE 1.692 * } {FOLLOW *
m1_vdd} {ALIGNMENT * CENTER} }
                { METAL m2_vss M2_stripe {NET VSS } {OFFSET 0.206 * } {STEPDISTANCE 1.692 * } {FOLLOW *
m1_vss} {ALIGNMENT * CENTER} }
        }
```

Example 5-19. *Example PSDL Script*

Figure 5-11 shows parts of the top die PG structure created by PSDL.

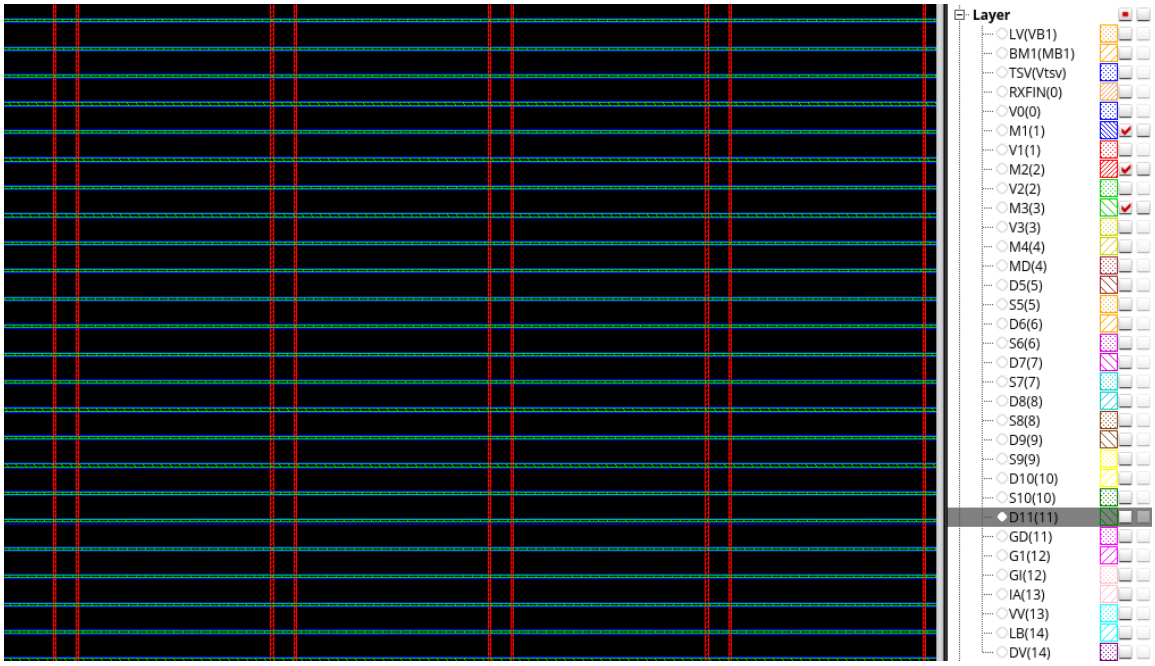

Figure 5-11. *Top Die PG Structure*

5.2.2.1. Example Run Script

NOTE: All commands are System Planner commands and should be executed with "planner::" or "eval_planner { }".

Before running the early static IR drop analysis, 3DCODE files should be read in Integrity 3D-IC. Example 5-8 shows how to read 3DCODE files in Integrity 3D-IC.

To create top level connections, see Example 5-20.

```
planner::edit_bulk_terms -mode create_to_parent -path BLK_D2D_BOT -select {(.*)} -new_name  {{1}}
planner::edit_bulk_terms -mode map_from_parent_net -path BLK_D2D_TOP -select {(.*)} -match {{1}}
```

Example 5-20. *Creating Top-Level Connections*

After reading the 3DCODE files, the user can set up the early static IR drop analysis. Since both dies are oriented face down, the topmost layer of the bottom die is used for the voltage source layer. To set the voltage source layer, see Example 5-21.

```
planner::set_vsrc_layers -layers LB -device_path BLK_D2D_BOT
```

Example 5-21. *Setting Voltage Source Layer*

To specify the current value, voltage value, and attached layers for each current region, please refer to Example 5-22.

```
planner::create_current_region -cell BLK_D2D_TOP -name BLK_D2D_TOP \
        -net {VDD VSS} -layer M1 \
        -current {5140mA} -region {0.0 0.0 3493.962 1002.672} \
        -voltage {0.8 0.0} -threshold {0.72 0.08}
planner::create_current_region -cell BLK_D2D_BOT -name BLK_D2D_BOT \
        -net {VDD VSS} -layer M1 \
        -current {2973mA} -region {0.0 0.0 3493.962 1002.672} \
        -voltage {0.8 0.0} -threshold {0.72 0.08}
```

Example 5-22. *Setting Current Region*

Example 5-23 sets the power structure configuration options and user preferences for the early static IR drop analysis flow.

```
planner::pnr::set_die_config \
        -device_path BLK_D2D_TOP \
        -power_structure_config top.psdl

planner::pnr::set_user_preference \
        pre_psdl_top.tcl \
        -device_path BLK_D2D_TOP \
        -flow psdl \
        -stage pre_run

planner::pnr::set_user_preference \
        post_psdl_top.tcl \
        -device_path BLK_D2D_TOP \
        -flow psdl \
        -stage run
```

```
planner::pnr::set_die_config \
        -device_path BLK_D2D_BOT \
        -power_structure_config bot.psdl

planner::pnr::set_user_preference \
        pre_psdl_bot.tcl \
        -device_path BLK_D2D_BOT \
        -flow psdl \
        -stage pre_run

planner::pnr::set_user_preference \
        post_psdl_bot.tcl \
        -device_path BLK_D2D_BOT \
        -flow psdl \
        -stage run
```

Example 5-23. *Setting User Preferences for PSDL*

Example 5-24 creates PG defs and then assigns them to the early static IR drop analysis flow. Bump region files are user-defined and correspond to the approximate number of bumps on each PG net. The TSV layer defines the top and bottom routing layers of the TSV cell.

```
planner::set_era_die_config -device_path BLK_D2D_TOP \
        -pgv BLK_D2D_TOP/PGV \
        -qrc BLK_D2D_TOP/qrcTechFile \
        -def [planner::pnr::run_psdl -run_dir PSDL -device_path BLK_D2D_TOP -tag BLK_D2D_TOP] \
        -bump_region BLK_D2D_TOP_bump_region.cfg \
        -vsrc_search_distance 150

planner::set_era_die_config -device_path BLK_D2D_BOT \
        -pgv BLK_D2D_BOT/PGV \
        -qrc BLK_D2D_BOT/qrcTechFile \
        -def [planner::pnr::run_psdl -run_dir PSDL -device_path BLK_D2D_BOT -tag BLK_D2D_BOT] \
        -bump_region BLK_D2D_BOT_bump_region.cfg \
        -tsv_layers "G1 BM1" \
        -tsv_subckt BLK_D2D_BOT/bottom_die.subckt \
        -vsrc_search_distance 300
```

Example 5-24. *Setting Die Configuration*

Refer to Example 5-25 to perform the early static IR drop analysis and run Voltus.

```
planner::run_era -flow pre_route -run_dir ERA_RUN \
        -tag "ERA.Pre_Route.run" -type run
```

Example 5-25. *Performing Early Static IR Drop Analysis*

5.2.2.2. Example Results

Figure 5-12 shows the 3DIC design structure after reading 3DCODE files.

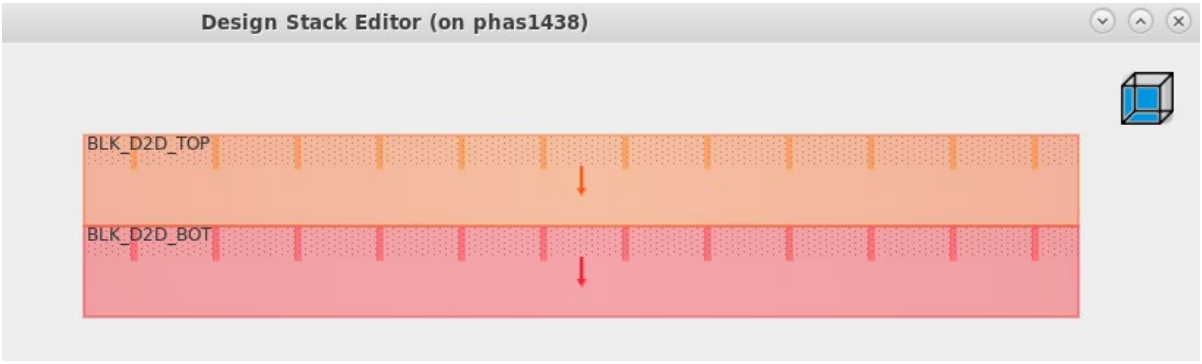

Figure 5-12. *3DIC Design Structure*

Figure 5-13 and Figure 5-14 show the early static IR drop analysis results of top die and bottom die using Voltus in Integrity 3D-IC. The results show the hot spot after power analysis.

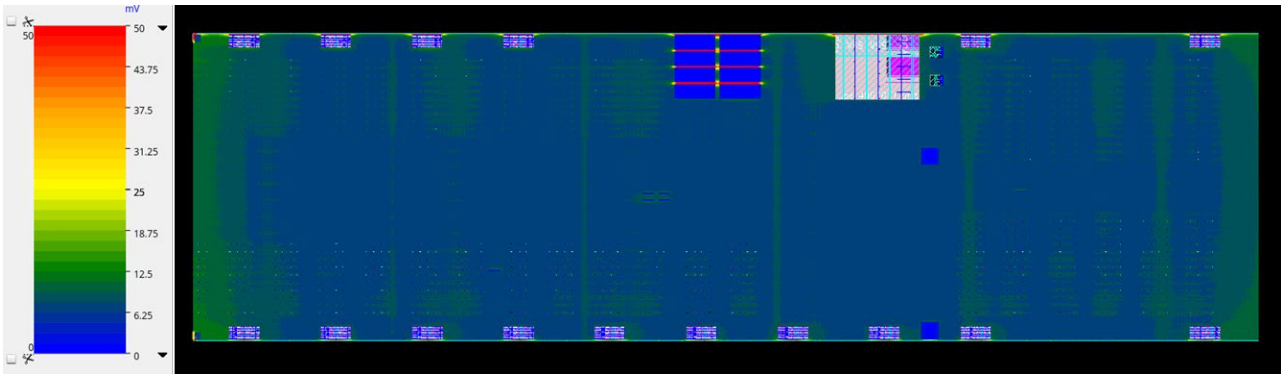

Figure 5-13. *Top Die Early Static IR Drop Analysis Results*

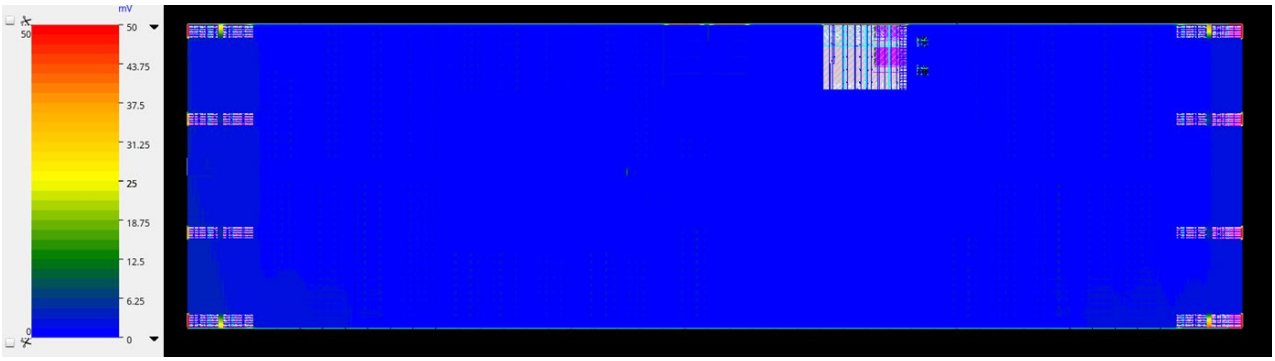

Figure 5-14. *Bottom Die Early Static IR Drop Analysis Results*

6. Appendix

List of Figures

List of Tables

List of Examples

7. Point of Contact

For any inquiries, please contact Samsung Foundry at E-mail: kiok22.kim@samsung.com