Chapter 1

Appendix A: Metrics Dictionary

1.1 Introduction

This appendix lists the current metrics that need to be captured from each design process. The purpose of this metrics list is to define metric names such that:

- each metric is uniquely identified (i.e., no metric has more than one name);
- no two metrics have the same name.

Thus we will use the following naming convention for each metric: Each name is composed of literals separated by underscores. The allowed literals are: namespace literals, quantifier literals, object literals, attribute literals and reference literals. The metric name must follow the naming rule which is:

\[ \langle \text{namespace} \rangle \_\_ \langle \text{object} \rangle \_\_ \langle \text{attribute} \rangle \_\_ \langle \text{quantifier} \rangle \_\_ \langle \text{reference} \rangle \_\_. \]

The metric name is case insensitive so that “NAME” is equivalent to “name”. However the metric value (i.e., the value entry for the corresponding metric name) is case sensitive. Each of those literals will be explained in the following subsections.

1.1.1 Namespace Literals

Namespace literals indicate the grouping (or the categorization) of the metrics. The literal is used to differentiate one metric from another metrics by the source or the purpose
of that metric. These literals may be composed of two or more sub-components separated by an underscore. The sub-components (or sub-spaces) are used to indicate additional grouping of the metrics. The first component usually refers to the process/tool that the corresponding metric is specified (e.g., SYNTHESIS, PLACEMENT, ROUTING, etc). If the metric is a generic one, i.e., it is independent of tool and process, then the value for the first sub-component is GENERAL. The next sub-components classify the metrics further. For example, TIMING_SLACK_HOLD, TIMING_SLACK_SETUP, TIMING_SLACK_PERIOD, etc are the metrics that can be grouped into TIMING_SLACK category which indicates the group of metrics that are used to capture the timing slack information. The literals that belong to namespace literals are listed as follows:

- CLOCKGENERALCONSTRAINT
- CLOCKGENERALNUMBER
- CLOCKGENERALTIMING
- CLOCKGENERALWAVEFORM
- CLOCKMESHNUMBER
- CLOCKMESHSOURCE
- CLOCKMESH
- CLOCKETREENUMBER
- CLOCKETREEROOTTREE
- DESIGNASYNCRONOUS
- DESIGNBLOCK
- DESIGNFSMNUMBER
- DESIGNNUMBER
- DESIGNPOWER
- DESIGNRTLNUMBER
- DESIGNRTLTOPMODULE
- DESIGNRTL
- DESIGN
- GENERALCPUNUMBER
• \texttt{GENERAL\_CPU}
• \texttt{GENERAL\_Disk}
• \texttt{GENERAL\_Host}
• \texttt{GENERAL\_LicenseMGR}
• \texttt{GENERAL\_Network}
• \texttt{GENERAL\_OS}
• \texttt{GENERAL\_Project}
• \texttt{GENERAL\_Shell}
• \texttt{GENERAL\_WindowMGR}
• \texttt{LAYOUT\_Area}
• \texttt{LAYOUT\_Check}
• \texttt{LAYOUT\_Number}
• \texttt{LAYOUT}
• \texttt{LIBRARY\_Foundry}
• \texttt{LIBRARY\_Number}
• \texttt{LIBRARY\_Rule}
• \texttt{LIBRARY}
• \texttt{PLACEMENT\_Number}
• \texttt{PLACEMENT\_Stat}
• \texttt{PLACEMENT}
• \texttt{PROJECT\_Number}
• \texttt{PROJECT}
• \texttt{ROUTING\_Number}
• \texttt{ROUTING}
• \texttt{SIMULATION\_Number}
• \texttt{SYNTHESIS\_Area}
• \texttt{SYNTHESIS\_Clock}
• \texttt{SYNTHESIS\_Constraint}
1.1.2 Object Literals

Object literals are the main literals that indicate the metrics that are captured (e.g., \textsc{NAME}). To increase readability, all object literals in the list will be bold faced.

1.1.3 Attribute Literals

Attribute literals are additional categories that represent the further specific collection of metrics separated by the corresponding attributes. The union of all metrics that differ just by their attributes are equivalent to the same metric without the attribute literals (if the respected metric exists). For example, \texttt{ROUTING\_WIRELENGTH\_HORIZ\_TOTAL} and \texttt{ROUTING\_WIRELENGTH\_VERT\_TOTAL} are sub-categories of \texttt{ROUTING\_WIRELENGTH\_TOTAL} where the sum of \texttt{ROUTING\_WIRELENGTH\_HORIZ\_TOTAL} value and \texttt{ROUTING\_WIRELENGTH\_VERT\_TOTAL} value are equal to the value of \texttt{ROUTING\_WIRELENGTH\_TOTAL}. The literals that belong to attribute literals are listed as follows:
1.1.4 Quantifier Literals

Quantifier literals indicate the quantity of the metrics, i.e., how the metrics are calculated. The literals that belong to quantifier literals are listed as follows:

- **TOTAL** (this is the sum of all values indicated by the metric)
- **AVE** (this is the average of all values indicated by the metric)
- **MAX** (this is the maximum value of all values indicated by the metric)
- **MIN** (this is the minimum value of all values indicated by the metric)
- **PCT** (this is the percentage of the value over the maximum possible value indicated by the metric)

1.1.5 Reference Literals

Reference literals indicate the point of references on which the metrics are calculated. For example, if a metric gives the number of cells for the entire chip, the literal for that metric’s name will be “CHIP” and the metric name will be TOTAL_NUM.Cells.Chip. This means that the total number of cells includes I/O pins. On the other hand, TOTAL_NUM.Cells.Core metric gives the total number of core cells that does not include I/O pins. The literals that belong to this list are listed as follows:

- **CHIP** (the entire chip)
- **CORE** (core blocks only, exclude I/O pins)
- **CORE_NO_MEM** (core blocks without memory)
- **BLOCK** (a specified block only)
- **MEMORY** (memory blocks only)
- **CTRL** (control unit only)
CHAPTER 1. APPENDIX A: METRICS DICTIONARY

1. DPATH (data path only)
2. DIGITAL (digital blocks only)
3. ANALOG (analog blocks only)

1.2 Metrics List Format

The metrics names in this section are sorted alphabetically in order to ease the search for description. Each metric will be formatted as follows:

**METRICS NAME** <type> (<unit>)
<description>

where:

- **METRICS NAME** is the name of the metrics. Since the name of the metrics are case insensitive, we use large capitalization for all letters.

- <type> is the data type that represents the value of the corresponding metric. The available data types are as follows:
  - string (a string literal)
  - integer (a round number)
  - double (a floating point number)
  - boolean (true or false)
  - date (a date and/or time)

- <unit> is the unit of the metrics, e.g., seconds, microns, watts, etc. This field is optional because some metrics do not have any unit.

1.3 Metrics List

**CLOCK**.**GENERAL**.**CONSTRAINT**.**InsertionDelay**.**Max** Double (ps)

This is the required maximum insertion delay in the clock structure.
CHAPTER 1. APPENDIX A: METRICS DICTIONARY

**CLOCK**

**GENERAL**

**CONSTRAINT**

**InsertionDelay Min** Double (ps)

This is the required minimum insertion delay in the clock structure.

**Skew Max** Double (ps)

This is the required maximum clock skew in the clock structure.

**TargetClockFreq** Double (MHz)

This metric indicates the targeted clock frequency of the chip.

**Transition Max** Double (ps)

This is the required maximum transition time in the clock structure.

**NUMBER**

**AddedCells Total** Integer

This is the total number of new cells that are added when adding the clock structure in the design.

**Buffers Total** Integer

This is the total number of buffers in the clock structure.

**ClockComponents Total** Integer

The total number of clock components in the design. Clock components include buffers and combinational gates for clock gating.

**DeletedCells Total** Integer

This is the total number of cells that are removed when adding the clock structure in the design.

**Domains Total** Integer

The total number of clock domains in the chip.

**Inverters Total** Integer

This is the total number of inverters in the clock structure.

**Sites Total** Integer

This is the total number of cell sites for clock structure. These cell sites are reserved for clock components (buffers, inverters, etc.).
CHAPTER 1. APPENDIX A: METRICS DICTIONARY

**CLOCK**

**GENERAL**

**TIMING**

**InsertionDelay Max** Double (ps)
This is the maximum insertion delay for the current clock structure.

**InsertionDelay Min** Double (ps)
This is the minimum insertion delay for the current clock structure.

**Skew Max** Double (ps)
This is the maximum clock skew for the current clock structure.

**Transition Max** Double (ps)
This is the maximum transition time obtained from the current clock structure.

**Waveform Fall** Double (ps)
This is the fall time (80% to 20%) of the clock waveform.

**Waveform High** Double (ps)
This is the high time duration (50% to 50%) of the clock waveform.

**Waveform Low** Double (ps)
This is the low time duration (50% to 50%) of the clock waveform.

**Waveform Rise** Double (ps)
This is the rise time (20% to 80%) of the clock waveform.

**Mesh GridSize HORIZ** Integer
This is the horizontal size of the grid if the clock structure is a mesh.

**Mesh GridSize VERT** Integer
This is the vertical size of the grid if the clock structure is a mesh.

**Mesh NUMBER Sinks TOTAL** Integer
This is the total number of clock sinks for the clock mesh.

**Mesh Source Name** String
This is the name of the source for the clock mesh.
This is the total number of new nets that are added when adding the clock structure in the design.

This is the total number of nets that are modified when adding the clock structure in the design.

This is the total number of nets that are removed when adding the clock structure in the design.

This is the total number of excluded pins in the clock tree.

This is the total number of leaf pins in the clock tree.

This is the total number of subtree roots in the clock tree.

This is the name of the clock root if the structure is a tree.

This is the percentage number of asynchronous path in the design.

Last update time for the block. This value is usually the same as the TOOL_RUN_TIMEEnd metric.

The name of the block that is being processed by the tool. If the current block is the entire design (i.e., a flattened chip), then this metric may be the same as DESIGN_Name.
DESIGN.CreatorName  String
    The name of the initial creator of the design. This metric is usually the same as
    the first TOOL.Run.USER.Name for the specified block/design.

DESIGN.FSM.Number.States.Total  Integer
    This is the total number of states in the Finite State Machine (FSM).

DESIGN.FSM.Number.TerminatingStates.Total  Integer
    This is the total number of terminating states in the Finite State Machine (FSM).

DESIGN.LastUpdate  Date
    Last update time for the design. This value is usually the same as the TOOL.Run.TimeEnd
    metric.

DESIGN.Name  String
    The name of the design that is being processed by the tool. This metric may be
    the same as GENERAL.PROJECT.Name.

DESIGN.Number.BidirectionalPads.Total  Integer
    The total number of bidirectional pads in the chip.

DESIGN.Number.Cells.Total.Analog  Integer
    The total number of cells in the analog blocks of the chip.

DESIGN.Number.Cells.Total.Core  Integer
    The total number of cells in the core blocks. This value excludes the I/O pins for
    the chip, i.e., internal cells only.

DESIGN.Number.Cells.Total.Digital  Integer
    The total number of cells in the digital blocks of the chip.

DESIGN.Number.InputPads.Total  Integer
    The total number of input pads in the chip.
DESIGN\_NUMBER\_Layers\_TOTAL Integer
The total number of layers which includes poly and diffusion layers.

DESIGN\_NUMBER\_Nets\_TOTAL Integer
The total number of nets in the whole chip. This includes the memory blocks, I/O nets, etc.

DESIGN\_NUMBER\_Nets\_TOTAL\_ANALOG Integer
The total number of nets in the analog blocks of the chip.

DESIGN\_NUMBER\_Nets\_TOTAL\_CORE Integer
The total number of nets in the core blocks. This value excludes the I/O pins for the chip, i.e., internal nets only.

DESIGN\_NUMBER\_Nets\_TOTAL\_DIGITAL Integer
The total number of nets in the digital blocks of the chip.

DESIGN\_NUMBER\_OutputPads\_TOTAL Integer
The total number of output pads in the chip.

DESIGN\_NUMBER\_Pads\_TOTAL Integer
The total number of I/O pads in the chip.

DESIGN\_NUMBER\_PipelineStages\_TOTAL Integer
This is the total number of pipeline stages in the design. The number of stages is also known as the depth of the pipeline.

DESIGN\_NUMBER\_ReuseIPVendor\_TOTAL Integer
This is the number of IP vendors for reuse.

DESIGN\_NUMBER\_RoutingLayers\_TOTAL Integer
The total number of metal layers that are used specifically for routing.

DESIGN\_NUMBER\_SpacingViolations\_TOTAL Integer
This is the total number of spacing rule violations that occur after routing.
**DESIGN\_NUMBER\_Vias\_TOTAL** Integer

This is the total number of vias after routing the nets. A via is a metal connection that connects a wire segment at one metal layer to another wire segment at another metal layer. Typically, wire segments on one metal layer are routed in the same direction. This means that a connection from that layer to the next layer is necessary in order to turn the net to the different direction. At the end of each nets, vias are also needed to connect the wires with the pins below.

**DESIGN\_POWER\_Consumption** Double \((mWatt)\)

This is the power consumption of the design.

**DESIGN\_POWER\_Dissipation** Double \((mWatt)\)

This is the amount of power dissipation of the design.

**DESIGN\_Reuse\_Pct** Double

This is the percentage of IP reuse in the design.

**DESIGN\_RTL\_Language** String

This is the language(s) that is (are) used to describe the RTL, e.g., Verilog, VHDL, C, etc. If there are more than one language that are used in the design, those languages will be listed in this metric. Each language is separated by a comma.

**DESIGN\_RTL\_NUMBER\_CodeLines\_TOTAL** Integer

This is the total number of lines in the RTL code.

**DESIGN\_RTL\_NUMBER\_Files\_TOTAL** Integer

This is the total number of files that are used to describe the RTL code.

**DESIGN\_RTL\_NUMBER\_HierarchyLayers\_TOTAL** Integer

This is the total number of layers of hierarchy in the design.

**DESIGN\_RTL\_TOP\_MODULE\_Name** String

The name of the top module in the RTL code, the corresponding logic synthesis run.
DESIGN._WhiteSpace_PCT_CHIP Double
This metric indicates the percentage of white space (ranges from 0 to 100). The white space indicates the space that are not occupied by cells which can be computed by subtracting the LAYOUT_AREA_Design_TOTAL_CHIP with the LAYOUT_AREA_Cell_TOTAL_CHIP.

GENERAL_CPU_LoadEnd Double
The ending load value of the CPU.

GENERAL_CPU_LoadStart Double
The starting load value of the CPU.

GENERAL_CPU_Number_Processors Integer
This metric indicates the number of processors that are installed on the machine on which the tool is run.

GENERAL_CPU_Speed Double (MHz)
This metric indicates the speed of the CPUs/processors that are installed on the machine on which the tool is run (e.g., 500MHz, etc.).

GENERAL_CPU_Type String
This metric indicates the type of the CPUs/processors that are installed on the machine on which the tool is run. For example, if we use Sun Ultra workstations, then the value for this metric will be Ultra-SparcII.

GENERAL_CPU_Utilization_MAX Double
The maximum CPU utilization for the specified tool run. The value for this metric ranges from 0 to 100.

GENERAL_Disk_Name String
This metric indicates the physical name of the disk on which the tool is run. If the disk is mounted remotely from another machine, then the machine name that owns the disk will be stored as well.

GENERAL_Disk_Usage_MAX Double (MB)es
The maximum disk usage for the specified tool run. This measures the peak use
of the disk storage which includes temporary and/or intermediate files that might get deleted once the tools completed.

**GENERAL_HOST_Id** String
The ID of the machine on which the tool is run. In Unix environment, this metric can be obtained by running the `hostid` command.

**GENERAL_HOST_IPAddress** String
The IP address of the machine on which the tool is run. This is the dot format, e.g., `137.64.12.177`.

**GENERAL_HOST_Name** String
The name of the machine on which the tool is run. In Unix environment, this metric can be obtained by running the `hostname` command.

**GENERAL_HOST_PhysicalMemory** Double (MB ytes)
This metric indicates the size of the physical memory for the machine that run the tools. This is the actual memory of that machine which excludes the virtual memory.

**GENERAL_HOST_Type** String
This metric indicates the architecture type of the machine on which the tool is run. In the UNIX environment, this info can be obtained by running the `uname` command.

**GENERAL_HOST_VirtualMemory** Double (MB ytes)
This metric indicates the size of the virtual memory for the machine that run the tools.

**GENERAL_LICENSEMGR_Name** String
The name of the license manager (if any) that is used when running the tool. This will be the name of the license manager that is used to manage the licensing issue of tools usage. One example that is used by many EDA vendors is the Flex License Manager (FlexLM).
**GENERAL.LICENSEMGR.Version** String

This metric indicates the version of the license manager (if any) that is used when running the tool. This data is typically available in the log file of the license manager.

**GENERAL.NETWORK.Load_MAX** Double

The maximum network load when the tool run. This measures the network communication during the execution of the tools.

**GENERAL.OS.Name** String

This is the name of the operating system that is used when the tool is launched. For example: Solaris, WinNT, etc.

**GENERAL.OS-Version** String

The version of the operating system that is used when the tool is launched.

**GENERAL.PROJECT.Name** String

The name of the project on which the current design/block (that is being processed by the tool) belongs. This metric may be the same as **DESIGN.Name**.

**GENERAL.SHELL.Name** String

This metric indicate the name of the shell (in the Unix environment) that is used when running the tool. If the tool is run under MS Windows environment, this metric will be empty.

**GENERAL.WINDOWMGR.Name** String

The name of the windows manager that is used when running the tools. For example: CommonDesktop, OpenWindows, Win98, etc.

**GENERAL.WINDOWMGR-Version** String

The version of the windows manager that is used when running the tools.

**LAYOUT AREA_Cell_TOTAL_ANALOG** Double \((nm^2)\)

The total area occupied by the cells of the analog block in the chip.
### CHAPTER 1. APPENDIX A: METRICS DICTIONARY

<table>
<thead>
<tr>
<th>Metric Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAYOUT_AREA_Cell_TOTAL_CHIP Double (nm(^2))</td>
<td>The total area occupied by the cells of the whole chip. This includes the memory blocks, logic blocks, I/O pins, etc.</td>
</tr>
<tr>
<td>LAYOUT_AREA_Cell_TOTAL_CORE Double (nm(^2))</td>
<td>The total area occupied by the cells of the core cells which excludes the I/O pins.</td>
</tr>
<tr>
<td>LAYOUT_AREA_Cell_TOTAL_DIGITAL Double (nm(^2))</td>
<td>The total area occupied by the cells of the digital block in the chip.</td>
</tr>
<tr>
<td>LAYOUT_AREA_Cell_TOTAL_DPATH Double (nm(^2))</td>
<td>The total area occupied by the cells of the data path block in the chip.</td>
</tr>
<tr>
<td>LAYOUT_AREA_Cell_TOTAL_MEMORY Double (nm(^2))</td>
<td>The total area occupied by the cells of the memory block in the specified chip.</td>
</tr>
<tr>
<td>LAYOUT_AREA_Design_TOTAL_ANALOG Double (nm(^2))</td>
<td>The total area of the analog block in the chip. LAYOUT_AREA_Design_TOTAL_ANALOG is the sum of the LAYOUT_AREA_Cell_TOTAL_ANALOG and the LAYOUT_AREA_Net_TOTAL_ANALOG.</td>
</tr>
<tr>
<td>LAYOUT_AREA_Design_TOTAL_CHIP Double (nm(^2))</td>
<td>The total area of the whole chip. This includes the memory blocks, logic blocks, I/O pins, etc. LAYOUT_AREA_Design_TOTAL_CHIP is the sum of the LAYOUT_AREA_Cell_TOTAL_CHIP and the LAYOUT_AREA_Net_TOTAL_CHIP.</td>
</tr>
<tr>
<td>LAYOUT_AREA_Design_TOTAL_CORE Double (nm(^2))</td>
<td>The total area of the core cells which excludes the I/O pins. LAYOUT_AREA_Design_TOTAL_CORE is the sum of the LAYOUT_AREA_Cell_TOTAL_CORE and the LAYOUT_AREA_Net_TOTAL_CORE.</td>
</tr>
<tr>
<td>LAYOUT_AREA_Design_TOTAL_DIGITAL Double (nm(^2))</td>
<td>The total area of the digital block in the chip. LAYOUT_AREA_Design_TOTAL_DIGITAL is the sum of the LAYOUT_AREA_Cell_TOTAL_DIGITAL and the LAYOUT_AREA_Net_TOTAL_DIGITAL.</td>
</tr>
<tr>
<td>LAYOUT_AREA_Design_TOTAL_DPATH Double (nm(^2))</td>
<td>The total area of the data path block in the chip. LAYOUT_AREA_Design_TOTAL_DPATH is the sum of the LAYOUT_AREA_Cell_TOTAL_DPATH and the LAYOUT_AREA_Net_TOTAL_DPATH.</td>
</tr>
</tbody>
</table>
### Layout Area

**Design Total Memory** Double \((nm^2)\)

The total area of the memory block in the specified chip. **Design Total Memory** is the sum of the **Cell Total Memory** and the **Net Total Memory**.

**Net Total Analog** Double \((nm^2)\)

The total area occupied by the nets of the analog block in the chip.

**Net Total Chip** Double \((nm^2)\)

The total area occupied by the nets of the whole chip. This includes the memory blocks, logic blocks, I/O pins, etc.

**Net Total Core** Double \((nm^2)\)

The total area occupied by the nets of the core cells which excludes the I/O pins.

**Net Total Digital** Double \((nm^2)\)

The total area occupied by the nets of the digital block in the chip.

**Net Total Data Path** Double \((nm^2)\)

The total area occupied by the nets of the data path block in the chip.

**Net Total Memory** Double \((nm^2)\)

The total area occupied by the nets of the memory block in the specified chip.

**Check DRC** Boolean

This metric indicates whether the layout passes the Design Rule Check (DRC).

**Check LVS** Boolean

This metric indicates whether the layout passes the Layout versus Schematic (LVS) Check.

**Height** Double \((nm)\)

This is the height of the layout.

**Number FlipFlops Total** Integer

This is the total number of flip-flops in the designs. This metric counts the actual number of flip-flops that are instantiated in the design. Use **Number FlipFlops Total** to represent the number of flip-flop libraries that are available to use.
<table>
<thead>
<tr>
<th>Metric</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layout.Number.Latches&gt;Total</td>
<td>Integer</td>
<td>This is the total number of latches in the designs. This metric counts the actual number of latches that are instantiated in the design. Use Library.Number.Latches&gt;Total to represent the number of latch libraries that are available to use.</td>
</tr>
<tr>
<td>Layout.Number.Rows&gt;Total</td>
<td>Integer</td>
<td>Total number of cell rows for a row based standard cell design. In a row based design, standard cells are placed in rows. The location of each cell is snapped to the closest cell site in the corresponding row.</td>
</tr>
<tr>
<td>Layout.Number.SiteOverlaps&gt;Total</td>
<td>Integer</td>
<td>Number of overlapping sites in the design. This metric is nonzero if the design allows overlapping sites. However, overlapping sites do not correspond to overlapping cells.</td>
</tr>
<tr>
<td>Layout.Number.Sites&gt;Total</td>
<td>Integer</td>
<td>This is the total number of cell sites in the row based standard cell design. A cell site is a rectangular region on which the cell origin is aligned. A cell can occupies more than one site. The height of the site is the same as the height of the cell (with some exception of double height cells), which is also the height of the row.</td>
</tr>
<tr>
<td>Layout.RowHeight</td>
<td>Double</td>
<td>The height of the cell rows for a row based standard cell design.</td>
</tr>
<tr>
<td>Layout.Width</td>
<td>Double (nm)</td>
<td>This is the width of the layout.</td>
</tr>
<tr>
<td>Library.Foundry.Name</td>
<td>String</td>
<td>This is the name of the foundry that provides the library.</td>
</tr>
<tr>
<td>Library.Name</td>
<td>String</td>
<td>The name of the design library used for the specified design.</td>
</tr>
<tr>
<td>Library.Number.DoubleHeightMasterCell&gt;Total</td>
<td>Integer</td>
<td>This is the total number of double height master cells in the library.</td>
</tr>
</tbody>
</table>
CHAPTER 1. APPENDIX A: METRICS DICTIONARY

**Library.Number_FlipFlops_Total** Integer
This is the total number of flip-flop libraries that are available to be used in the designs. This metric represents the variation of flip-flops available. **Layout.Number_FlipFlops_Total** is the metric that capture the actual number of flip-flops in the design.

**Library.Number_Latches_Total** Integer
This is the total number of latch libraries that are available to be used in the designs. This metric represents the variation of latches available. **Layout_Number_Latches_Total** is the metric that capture the actual number of latches in the design.

**Library.Number_MasterCells_Total** Integer
This is the total number of master cells in the library.

**Library.Number_MasterCellVariations_Max** Integer
This is the maximum number of variation for master cells. E.g., for a given cell type, there may be different variations of master cells available. These variations indicate the difference in driver strengths and gate sizes.

**Library.Number_MasterCellVariations_Min** Integer
This is the minimum number of variation for master cells. The lowest value for this number is one since the library should contains at least one master cell for each cell type. This metric indicates how rich the library that is used in the design.

**Library.Number_MasterVias_Total** Integer
This is the total number of master vias in the library.

**Library.Number_UniqueMasterCells_Total** Integer
This is the total number of unique master cells in the library. All duplicates (with different size and driver strength) will only be counted once.

**Library.ProcessGeometry** Double (nm)
This is the process geometry for the design, i.e., 130nm, 100nm, etc.
**LIBRARY**. **RULE**. **MinSpacing**  Double (nm)

This is the minimum spacing, for metal on the layer, allowed between two regular geometries on different nets in the design. Spacing is the edge-to-edge separation, both orthogonal and diagonal. This spacing rule applies to line-to-line, line-to-via, and via-to-via distances.

**LIBRARY**. **Version**  String

The version of the design library used for the specified design.

**PLACEMENT**. **NUMBER**. **CellOverlaps**. **TOTAL**  Integer

This is the total number of overlapping cells. For some designs, overlapping cells are allowed because those cells share common pin (i.e., power/ground). Thus, if the cells are arranged so that every other row, the cell orientation is flipped, the power pin from one cell can be connected (shared) with the power pin from another cell below it (similarly for the ground pin).

**PLACEMENT**. **NUMBER**. **NetCut**. **HORIZ**. **MAX**  Integer

This is the maximum number of horizontal netcut. A horizontal netcut is the number of net that cross a horizontal line that cut the layout.

**PLACEMENT**. **NUMBER**. **NetCut**. **VERT**. **MAX**  Integer

This is the maximum number of vertical netcut. A vertical netcut is the number of net that cross a vertical line that cut the layout.

**PLACEMENT**. **STAT**. **CellNeighbor**. **AVE**  Double

This is the average number of neighboring cells connected to each cell in the design.

**PLACEMENT**. **STAT**. **CellNeighbor**. **MAX**  Double

This is the maximum number of neighboring cells connected to each cell in the design.

**PLACEMENT**. **STAT**. **CellNeighbor**. **MIN**  Double

This is the minimum number of neighboring cells connected to each cell in the design.
CHAPTER 1. APPENDIX A: METRICS DICTIONARY

**Placement Wirelength Horiz_Total** Double

This is the total horizontal wirelength after placement. This metric counts only the horizontal segment of the wirelength.

**Placement Wirelength Total** Double

This is the total estimated wirelength after placement. This metric is the sum of **Placement Wirelength Horiz_Total** and **Placement Wirelength Vert_Total**. The total estimated wirelength is calculated using half-perimeter bounding box of the nets.

**Placement Wirelength Vert_Total** Double

This is the total vertical wirelength after placement. This metric counts only the vertical segment of the wirelength.

**Project Experience Ave** Double (Years)

This is the average year of experience of the engineers that work on the project.

**Project Number Designers** Integer

This is the total number of designers that collaborate on the specified project. The designers are the persons that designing and writing the RTL code as well as the FSM.

**Project Number Engineers Total** Integer

This is the total number of engineers that work on the project, i.e., the size of the team. This metric includes the manager that leads the team.

**Project Number FlowEngineers** Integer

This is the total number of engineers that work specifically on the design flows. These engineers may or may not be the same as **Project Number Designers**.

**Project Number Sites** Integer

This is the total number of design sites for all the engineers that work on the specified project. If all engineers are local within a single company, which is a typical scenario for small companies, then the number of site is one.
CHAPTER 1. APPENDIX A: METRICS DICTIONARY

PROJECT\_TimeEnd Date
This is the ending time of the project.

PROJECT\_TimeStart Date
This is the starting time of the project.

ROUTING\_NUMBER\_Tracks\_HORIZ\_TOTAL Integer
This is the number of horizontal routing tracks. In a track-based routing, nets are routed on top of routing tracks.

ROUTING\_NUMBER\_Tracks\_VERT\_TOTAL Integer
This is the number of vertical routing tracks.

ROUTING\_NUMBER\_Vias\_TOTAL Integer
This is the number of vias that are used after routing.

ROUTING\_NUMBER\_Violations\_TOTAL Integer
This is the total number of routing violation after routing.

ROUTING\_NUMBER\_WrongWayWirelength\_TOTAL Double
This is the total number of wire segments that are routed in the non-preferred direction. *Wrong Way* routing is explained in ROUTING\_WrongWayWirelength\_TOTAL.

ROUTING\_Wirelength\_HORIZ\_TOTAL Double
This is the total horizontal wirelength after routing. The total horizontal wirelength is the sum of all wire segments length that are in horizontal direction.

ROUTING\_Wirelength\_TOTAL Double
This is the total net wirelength after routing for the entire chip. The length of a net is obtained by the total length of all wire segments that belong to the specified net. Typically, a wire segment is either a horizontal segment or a vertical segment. Total wirelength can usually be separated into two parts, the horizontal part and vertical part. Thus, the sum of ROUTING\_Wirelength\_HORIZ\_TOTAL and ROUTING\_Wirelength\_VERT\_TOTAL is equal to ROUTING\_Wirelength\_TOTAL.
CHAPTER 1. APPENDIX A: METRICS DICTIONARY

ROUTING_Wirelength_VERT_TOTAL Double
This is the total vertical wirelength after routing. The total vertical wirelength is the sum of all wire segments length that are in vertical direction.

ROUTING_WrongWayWirelength_AVE Double
This is the average length of wire segments that are routed in the non-preferred direction. Wrong Way routing is explained in ROUTING_WrongWayWirelength_TOTAL.

ROUTING_WrongWayWirelength_MAX Double
This is the maximum length of wire segments that are routed in the non-preferred direction. Wrong Way routing is explained in ROUTING_WrongWayWirelength_TOTAL.

ROUTING_WrongWayWirelength_MIN Double
This is the minimum non-zero length of wire segments that are routed in the non-preferred direction. Wrong Way routing is explained in ROUTING_WrongWayWirelength_TOTAL.

ROUTING_WrongWayWirelength_TOTAL Double
This is the total length of wire segments that are routed in the non-preferred direction. Each routing layer typically has a preferred routing direction which is either horizontal or vertical. However, due to some blockages, a detour may be required where the corresponding wire will be turned to the other direction. The wire segment that are routed in this non-preferred direction is called Wrong Way routing. This metric is the sum of all wrong-way routing segments.

SIMULATION_NUMBER_Failures_TOTAL Integer
This is the total number of failures encountered while simulating the design.

SIMULATION_NUMBER_Vectors_TOTAL Integer
This is the total number of stimulus vectors for simulation.

SYNTHESIS_AREA_Cells_TOTAL Double (\text{\mu m}^2)
This is the total area for cells after logic synthesis. This metric is the sum of SYNTHESIS_AREA_Cells_TOTAL_COMBINATIONAL and SYNTHESIS_AREA_Cells_TOTAL_NONCOMBINATIONAL.
SYNTHESIS_Area.Cells.Total.COMBINATIONAL Double (nm²)
This is the total area occupied by combinational cells. Combinational cells are instances / gates that are not controlled by clock signal.

SYNTHESIS_Area.Cells.Total.NONCOMBINATIONAL Double (nm²)
This is the total area occupied by non-combinational cells. Non-combinational cells are sequential instances or gates that are controlled by clock signal. These includes latches and flip-flops.

SYNTHESIS_Area.Design.Total Double (nm²)
This is the total area estimated after logic synthesis. This metric is the sum of SYNTHESIS_Area.Nets.Total and SYNTHESIS_Area.Cells.Total.

SYNTHESIS_Area.Nets.Total Double (nm²)
This is the total area occupied by nets after logic synthesis.

SYNTHESIS_Clock.Freq Double (MHz)
This is the clock frequency estimated in logic synthesis.

SYNTHESIS_Clock.Name String
This indicates the name of the clock.

SYNTHESIS_CONSTRAINT.Fanout.Max Double (ps)
This metric indicates the required maximum fanout time.

SYNTHESIS_CONSTRAINT.Transition.Max Double (ps)
This metric indicates the required maximum transition time.

SYNTHESIS_NUMBER.Instances.Total Integer
This is the total number of instances that are obtained after the logic synthesis run. The total number of instances is the sum of SYNTHESIS_NUMBER.Instances.Total.COMBINATIONAL, SYNTHESIS_NUMBER.Instances.Total.NONCOMBINATIONAL, SYNTHESIS_NUMBER.Instances.Total.HI and SYNTHESIS_NUMBER.Instances.Total.BLACKBOX.

SYNTHESIS_NUMBER.Instances.Total.BLACKBOX Integer
This is the total number of black boxes (blocks) on which the more detail information within each block is not available. These black boxes are typically
reused IPs. Only the interface to these boxes are available, i.e., the input/output connection to the boxes.

**SYNTHESIS_NUMBER_Instances_TOTAL_COMBINATIONAL** Integer
This is the total number of combinational instances. Combinational instances are combinational gates that are not controlled by clock signal.

**SYNTHESIS_NUMBER_Instances_TOTAL_HIERARCHICAL** Integer
This is the total number of hierarchical blocks on which the more detail information within each block is available. When necessary, these blocks can be expanded (or flattened).

**SYNTHESIS_NUMBER_Instances_TOTAL_NONCOMBINATIONAL** Integer
This is the total number of non-combinational instances. Non-combinational instances are sequential gates which includes latches and flip-flops.

**SYNTHESIS_NUMBER_Nets_TOTAL** Integer
This metric indicates the number of nets after logic synthesis.

**SYNTHESIS_NUMBER_Ports** Integer
This is the number of external ports generated after the logic synthesis. Ports are external connections that connect the internal circuit within the current block with the external circuits.

**SYNTHESIS_TIMING_Fanout_MAX** Double (ps)
This is the maximum fanout time estimated after the logic synthesis.

**SYNTHESIS_TIMING_Library** String
This is the name of the timing library that is used in logic synthesis. A timing library is a library that contains the delay for each cell.

**SYNTHESIS_TIMING_OperatingCondition** String
This indicates the operating condition that is used in logic synthesis. The options for the operating condition are best case, average case and worst case.
CHAPTER 1. APPENDIX A: METRICS DICTIONARY

SYNTHESIS_TIMING_Process Double
This is the process value that is used in logic synthesis.

SYNTHESIS_TIMING_PVTMode String
This indicates the Process, Voltage and Temperature (PVT) variation mode that is used in logic synthesis. The options are best case, average case and worst case.

SYNTHESIS_TIMING_SlewPropagation String
This indicates the type of timing slew propagation that is used in logic synthesis run. This type will be either the early propagation or the late propagation.

SYNTHESIS_TIMING_Temperature Double (F)
This is the temperature that is used in logic synthesis.

SYNTHESIS_TIMING_Transition_MAX Double (ps)
This is the maximum transition time estimated after the logic synthesis.

SYNTHESIS_TIMING_TreeType String
This indicates the type of clock tree used in logic synthesis. The options are best case, average case and worst case.

SYNTHESIS_TIMING_Voltage Double (mV)
This is the voltage that is used in logic synthesis.

SYNTHESIS_TIMING_Wireload String
This is the name of the wireload model that is used in logic synthesis. A wireload model is a table lookup formulae where it gives the capacitance value and the resistance value for a certain cell based on its pin. The capacitance and the resistance are used to calculate the delay for the specified cell.

TEST_FAULT_Coverage_PCT Double
This is the percentage of fault test coverage.

TEST_FAULT_Model String
This is the model used for fault test (e.g., stuck-at fault model, etc.).
**CHAPTER 1. APPENDIX A: METRICS DICTIONARY**

**Test_FullScan**  Boolean
This is an indication whether a full scan test is implemented or not.

**Test_IDDQ_Coverage_Pct**  Double
This is the percentage of test IDDQ coverage.

**Test_Number_Benches_Total**  Integer
This is the total number of test benches that are used for testing the design.

**Test_Number_Patterns_Total**  Integer
This is the total number of test patterns.

**Test_RTL_Code_Coverage_Pct**  Double
This is the percentage of RTL code coverage.

**Timing_Constraint_InputDelay_Ave (ps)**
This is the average input delay constraint for the input pins.

**Timing_Constraint_InputDelay_Max (ps)**
This is the maximum input delay constraint for the input pins.

**Timing_Constraint_InputDelay_Min (ps)**
This is the minimum input delay constraint for the input pins.

**Timing_Constraint_InputSlew_Ave (ps)**
This is the average input slew constraint for the input pins.

**Timing_Constraint_InputSlew_Max (ps)**
This is the maximum input slew constraint for the input pins.

**Timing_Constraint_InputSlew_Min (ps)**
This is the minimum input slew constraint for the input pins.

**Timing_Constraint_OutputDelay_Ave (ps)**
This is the average output delay constraint for the output pins.

**Timing_Constraint_OutputDelay_Max (ps)**
This is the maximum output delay constraint for the output pins.
CHAPTER 1. APPENDIX A: METRICS DICTIONARY

TIMING_CONSTRAINT_OutputDelay_MIN Double (ps)
This is the minimum output delay constraint for the output pins.

TIMING_CONSTRAINT_OutputSlew_AVE Double (ps)
This is the average output slew constraint for the output pins.

TIMING_CONSTRAINT_OutputSlew_MAX Double (ps)
This is the maximum output slew constraint for the output pins.

TIMING_CONSTRAINT_OutputSlew_MIN Double (ps)
This is the minimum output slew constraint for the output pins.

TIMING_NUMBER_FalsePaths_TOTAL Integer
This is the total number of false timing paths in the design.

TIMING_NUMBER_MultiCyclePaths_TOTAL Integer
This is the total number of multi-cycle paths in the design.

TIMING_RESULT_ClockFreq_MAX_CHIP Double (MHz)
This metric indicates the maximum achieved clock frequency of the chip. If the chip has only one clock, then TIMING_RESULT_ClockFreq_MAX_CHIP is the same as TIMING_RESULT_ClockFreq_MIN_CHIP.

TIMING_RESULT_ClockFreq_MIN_CHIP Double (MHz)
This metric indicates the minimum achieved clock frequency of the chip. If the chip has only one clock, then TIMING_RESULT_ClockFreq_MIN_CHIP is the same as TIMING_RESULT_ClockFreq_MAX_CHIP.

TIMING_SLACK_GatedClk_AVE Double (ps)
This is the average of gated clock slack.

TIMING_SLACK_GatedClk_MAX Double (ps)
This is the maximum gated clock slack.

TIMING_SLACK_GatedClk_MIN Double (ps)
This is the minimum gated clock slack. If there exists gated clock violation, then
this value equals to the negative value of TIMING\_VIOLATION\_GatedClk\_MAX metric.

**TIMING\_SLACK\_Hold\_Ave Double (ps)**

This is the average hold timing slack. A hold timing slack for a path is the difference between the time it takes to change the input signal to the latch (when the latch becomes enabled) and the time specified in the corresponding hold constraint. A hold constraint is a timing period that requires an input signal to remain the same for the specified time after the sink latch becomes active (enabled). This metric is the average of all hold timing slacks of all paths in the design. A negative hold slack indicates that there is a hold timing violation for the specified path.

**TIMING\_SLACK\_Hold\_Max Double (ps)**

This is the maximum hold timing slack (see TIMING\_SLACK\_Hold\_Ave for the definition of hold timing slack).

**TIMING\_SLACK\_Hold\_Min Double (ps)**

This is the minimum hold timing slack. If there exists hold violations, then this metric equals to the negative value of the TIMING\_VIOLATION\_Hold\_Max metric.

**TIMING\_SLACK\_Period\_Ave Double (ps)**

This is the average period slack.

**TIMING\_SLACK\_Period\_Max Double (ps)**

This is the maximum period slack.

**TIMING\_SLACK\_Period\_Min Double (ps)**

This is the minimum period slack. If there exists period violation, then this value equals to the negative value of TIMING\_VIOLATION\_Period\_Max metric.

**TIMING\_SLACK\_Setup\_Ave Double (ps)**

This is the average setup timing slack. A setup timing slack for a path is the difference between the clock period and the time it takes to propagate signals
from one latch to the next latch. A setup constraint is a timing period on which
the signal must arrived before the corresponding latch becomes active (enabled).
This metric is the average of all setup timing slacks of all paths in the design.
A negative setup slack indicates that there is a setup timing violation for the
specified path.

**TIMING\_SLACK\_Setup\_MAX** Double (ps)
This metric indicates the maximum setup timing slack for a given chip.

**TIMING\_SLACK\_Setup\_MIN** Double (ps)
This metric indicates the minimum setup timing slack for a given chip. If there
exists setup violations, then this metric equals to the negative value of TIM-
ING\_VIOLATION\_Setup\_MAX.

**TIMING\_SLACK\_Width\_AVE** Double (ps)
This is the average width slack.

**TIMING\_SLACK\_Width\_MAX** Double (ps)
This is the maximum width slack.

**TIMING\_SLACK\_Width\_MIN** Double (ps)
This is the minimum width slack. If there exists width violation, then this value
equals to the negative value of TIMING\_VIOLATION\_Width\_MAX metric.

**TIMING\_VIOLATION\_GatedClk\_AVE** Double (ps)
This is the average of gated clock violation.

**TIMING\_VIOLATION\_GatedClk\_MAX** Double (ps)
This is the maximum gated clock violation. If there is no gated clock violation,
then this metric is reset to zero.

**TIMING\_VIOLATION\_GatedClk\_MIN** Double (ps)
This is the minimum gated clock violation. If there is no gated clock violation,
then this metric is reset to zero.
**TIMING VIOLATION**

**GatedClk_Total** Integer

This is the total number of gated clock violations.

**TIMING VIOLATION**

**Hold_Ave** Double (ps)

The average hold timing violation. A hold timing violation for a path exists if the time it takes to change the input signal to the latch, when the latch becomes enabled, is smaller than the hold constraint (see **TIMING SLACK Hold_Ave** for the definition of a hold constraint). The hold violation value for a specific path is the same as the negation of the corresponding hold slack value; however, all positive hold slack paths will not be counted as hold violations. Thus, this metric only counts the hold violations that exist (i.e., the value is zero, 0, if there is no hold violation).

**TIMING VIOLATION**

**Hold_Max** Double (ps)

The maximum hold timing violation. If there is no hold timing violations, then this value is reset to zero (see **TIMING VIOLATION Hold_Ave** for the definition of hold timing violation).

**TIMING VIOLATION**

**Hold_Min** Double (ps)

The minimum hold timing violation. If there is no hold timing violations, then this value is reset to zero. Otherwise, this metric contains the smallest non-zero hold timing violation.

**TIMING VIOLATION**

**Hold_Total** Integer

The total number of hold timing violations (see **TIMING VIOLATION Hold_Ave** for the definition of hold timing violation).

**TIMING VIOLATION**

**Limit_Total** Integer

This is the total number of limit violations.

**TIMING VIOLATION**

**Period_Ave** Double (ps)

This is the average period violation.

**TIMING VIOLATION**

**Period_Max** Double (ps)

This is the maximum period violation. If there is no period violation, then this
metric is reset to zero.

**TIMING\_VIOLATION\_Period\_Min** Double (ps)

This is the minimum period violation. If there is no period violation, then this metric is reset to zero.

**TIMING\_VIOLATION\_Period\_Total** Integer

This is the total number of period timing violations.

**TIMING\_VIOLATION\_Setup\_AVE** Double (ps)

This is the average setup timing violation. A setup timing violation for a path exists if the time it takes to propagate signals to the latch is longer than the setup constraint (see **TIMING\_SLACK\_Setup\_AVE** for the definition of a setup constraint). The setup violation value for a specific path is the same as the negation of the corresponding setup slack value; however, all positive setup slack paths will not be counted as setup violations. Thus, this metric only counts the setup violations that exist (i.e., the value is zero, 0, if there is no setup violation).

**TIMING\_VIOLATION\_Setup\_MAX** Double (ps)

This indicates the maximum setup timing violation. This metric is zero if there is no setup timing violation.

**TIMING\_VIOLATION\_Setup\_MIN** Double (ps)

This indicates the minimum setup timing violation which is the smallest violation that is greater than zero. This metric is zero if there is no setup timing violation.

**TIMING\_VIOLATION\_Setup\_TOTAL** Integer

This is the total number of setup violations (see **TIMING\_VIOLATION\_Setup\_AVE** for the definition of hold timing violation).

**TIMING\_VIOLATION\_Width\_AVE** Double (ps)

This is the average width violation.

**TIMING\_VIOLATION\_Width\_MAX** Double (ps)

This is the maximum width violation. If there is no width violation, then this metric is reset to zero.
TIMING\_VIOLATION\_Width\_MIN Double (ps)

This is the minimum width violation. If there is no width violation, then this metric is reset to zero.

TIMING\_VIOLATION\_Width\_TOTAL Integer

This is the total number of width timing violations.

TOOL\_GENERAL\_CompiledDate Date

The time and date on which the corresponding tool is compiled (or build). Some tools print out their compilation date to the log files so this metric can be captured simply by scanning the log files. If the tools do not print their compilation time, the time stamp on the executable files of the tool can be used as the value for this metric.

TOOL\_GENERAL\_Dir String

The absolute path on which the tool is located (e.g., /usr/local/UCLA/bin).

TOOL\_GENERAL\_Memory\_MAX Double (MBytes)

Maximum memory used for the corresponding tool run.

TOOL\_GENERAL\_Name String

The name of the tool that is executed. This is the name of the program that is invoked by the user or by any other tools.

TOOL\_GENERAL\_Vendor String

This metric indicates the company/vendor that build/maintain the tool. For example, the vendor list for EDA tools will be: “Avant!”, “Cadence Design Systems”, “Magma Design Automation”, “Monterey Design Systems”, “Synopsys”, etc. For internally build tools, the value for this metric is the company name or the group name that produce the corresponding tools.

TOOL\_GENERAL\_Version String

The version of the tool that is executed. Usually, tools print their version number in the log files. For some tools, their version number can be obtained by executing the tool with a specific flag, such as -V or -version.
**CHAPTER 1. APPENDIX A: METRICS DICTIONARY**

**TOOL_RUN_CPUTime.Total** Double (seconds)

Total CPU time for the entire tool run.

**TOOL_RUN.ExitStatus** String

Tool’s exit status which is either SUCCESS or FAIL. In addition, failed run will have additional information about the reason of failure appended to the end of the string. E.g., FAIL_SIGINT indicates that a failure that is caused by interrupt signal (Ctrl-C or KILL)

**TOOL_RUN.ProcessID** String

This metric indicates the process ID for the corresponding tool run.

**TOOL_RUN_RealTime.Total** Double (seconds)

Total real time for the entire tool run.

**TOOL_RUN.RunDir** String

The directory on which the user execute the tool. This is usually the current working directory which is not the same as TOOL_GENERAL.Dir.

**TOOL_RUN.TimeEnd** Date

The ending date and time of the specified tool run. The time stamp is captured once the tool completes its execution.

**TOOL_RUN.TimeStart** Date

The starting date and time of the specified tool run. The time stamp is captured right before the tool started.

**TOOL_RUN.USER_ID** Integer

The Unix user id that represent the user that executes the tool. This metric represents the numerical id for the USER_NAME metric which can also be obtained directly from the OS.

**TOOL_RUN.USER_Name** String

The name of the user (Unix user name) that executes the tool. This metric can be obtained directly from the OS (in Unix environment) every time a tool is executed.