# ZeBu® Partition Viewer Application Note

Version V-2024.03-1, July 2024

# Contents

Feedback

# Preface

This chapter consists of the following topics:

## About This Book

The *ZeBu® Partition Viewer Application Note* describes Partition Viewer that is designed to help with partitioning in customer designs through some advanced graphical visualization components.

## Contents of This Book

The *ZeBu® Partition Viewer Application Note* has the following chapters.

| Chapter | Describes... |
|---|---|
| Introduction to Partition Viewer | Describes brief overview of partition viewer. |
| Partition Viewer HTML Page | Describes a typical partition viewer HTML page. |
| Partition Viewer Page Generation | Describes two different ways to generate partition viewer page. |
| zNetVision | Describes the zNetVision tool. |
| Limitations of Partition Viewer | Describes the partition viewer. |
| Appendix IO Graph | Describes the IO graph that represents connectivity between the partitioned instances. |

## Related Documentation

| Document Name | Description |
|---|---|
| *ZeBu User Guide* | Provides detailed information on using ZeBu. |
| *ZeBu Debug Guide* | Provides information on tools you can use for debugging. |
| *ZeBu Debug Methodology Guide* | Provides debug methodologies that you can use for debugging. |
| *ZeBu Unified Command-Line User Guide* | Provides the usage of Unified Command-Line Interface (UCLI) for debugging your design. |
| *ZeBu UTF Reference Guide* | Describes Unified Tcl Format (UTF) commands used with ZeBu. |
| *ZeBu Power Aware Verification User Guide* | Describes how to use Power Aware verification in ZeBu environment, from the source files to runtime. |
| *ZeBu Functional Coverage User Guide* | Describes collecting functional coverage in emulation. |
| *Simulation Acceleration User Guide* | Provides information on how to use Simulation Acceleration to enable cosimulating SystemVerilog testbenches with the DUT |
| *ZeBu Verdi Integration Guide* | Provides Verdi features that you can use with ZeBu. This document is available in the Verdi documentation set. |
| *ZeBu Runtime Performance Analysis With zTune User Guide* | Provides information about runtime emulation performance analysis with zTune. |
| *ZeBu Custom DPI Based Transactors User Guide* | Describes ZEMI-3 that enables writing transactors for functional testing of a design. |
| *ZeBu LCA Features Guide* | Provides a list of Limited Customer Availability (LCA) features available with ZeBu. |
| *ZeBu Synthesis Verification User Guide* | Provides a description of zFmCheck. |
| *ZeBu Transactors Compilation Application Note* | Provides detailed steps to instantiate and compile a ZeBu transactor. |
| *ZeBu zManualPartitioner Application Note* | Describes the zManualPartitioner feature for ZeBu. It is a graphical interface to manually partition a design. |
| *ZeBu Hybrid Emulation Application Note* | Provides an overview of the hybrid emulation solution and its components. |

Feedback

## Typographical Conventions

This document uses the following typographical conventions:

| To indicate | Convention Used |
|---|---|
| Program code | `OUT <= IN;` |
| Object names | `OUT` |
| Variables representing objects names | `<sig-name>` |
| Message | Active low signal name '<sig-name>' must end with _X. |
| Message location | `OUT` <= IN; |
| Reworked example with message removed | `OUT_X` <= IN; |
| Important Information | **NOTE:** This rule... |

The following table describes the syntax used in this document:

| Syntax | Description |
|---|---|
| [ ] (Square brackets) | An optional entry |
| { } (Curly braces) | An entry that can be specified once or multiple times |
| \| (Vertical bar) | A list of choices out of which you can choose one |
| ... (Horizontal ellipsis) | Other options that you can specify |

## Synopsys Statement on Inclusivity and Diversity

Synopsys is committed to creating an inclusive environment where every employee, customer, and partner feels welcomed. We are reviewing and removing exclusionary language from our products and supporting customer-facing collateral. Our effort also includes internal initiatives to remove biased language from our engineering and working environment, including terms that are embedded in our software and IPs. At the same time, we are working to ensure that our web content and software applications are usable to people of varying abilities. You may still find examples of non-inclusive language in our software or documentation as our IPs implement industry-standard specifications that are currently under review to remove exclusionary language.

# 1 Introduction to Partition Viewer

Partition Viewer is a tool allowing to display in a single HTML page, from a design compilation directory, several useful information such as the overall design hierarchy, its partitioning on the configuration, FPGA resource utilization, filling rates and IO cut.

Partition Viewer is designed to help with partitioning in customer designs through some advanced graphical visualization components.

Use the following command line to run the tool's generation:

```
?> source <zebu.release>
?> zPython --pythonpath ${ZEBU_ROOT}/etc/scripts/netlist_tools
 ${ZEBU_ROOT}/etc/scripts/netlist_tools/tools/partition_viewer/flow/pview
_database_gen.py <zebu.work> -o <output.dir>
```

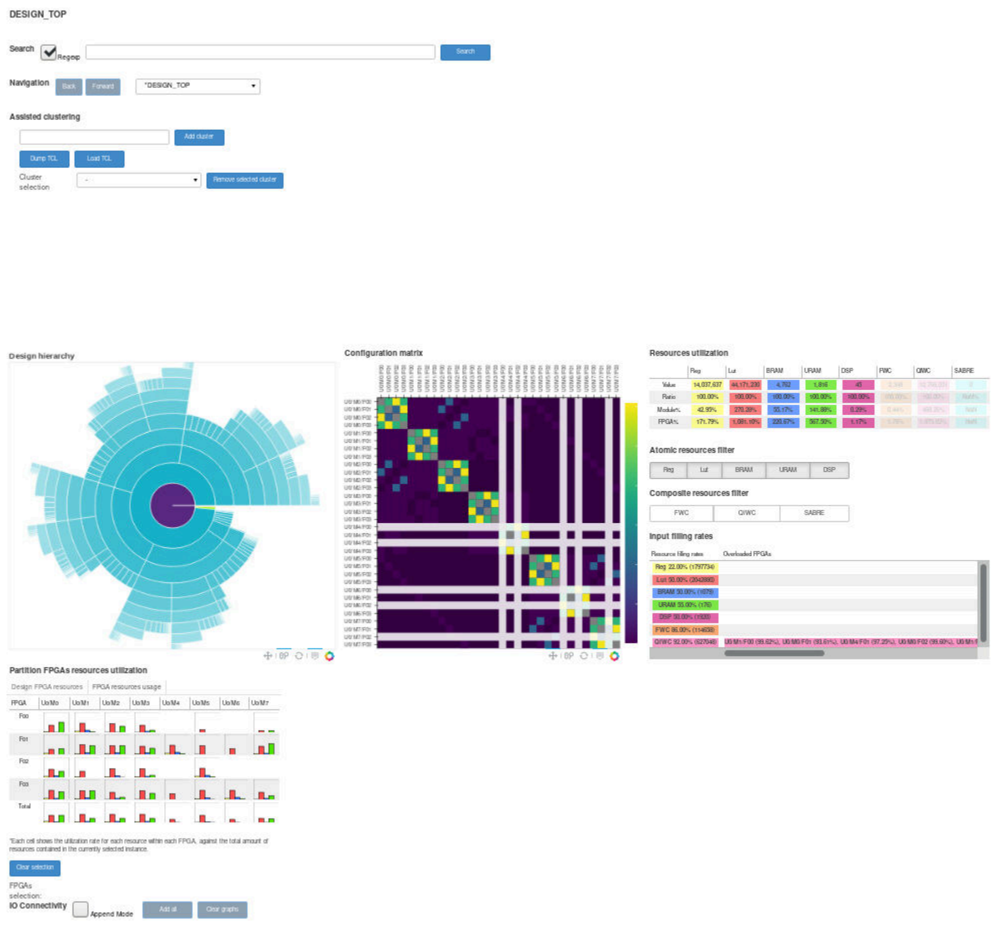
Where,

- `<zebu.work>` is a path to a `zebu.work` compilation folder
- `<output.dir>` is the output path in which the HTML page is generated

# 2
# Partition Viewer HTML Page

An example of a typical Partition Viewer HTML page is shown as follows:

*Figure 1 Partition viewer HTML page overview*



Resources utilization

| | Reg | Lut | BRAM | URAM | DSP | FWC | QIWC | SABRE |
|---|---|---|---|---|---|---|---|---|
| Value | 14,037,637 | 44,171,230 | 4,762 | 1,816 | 45 | [illegible] | [illegible] | 0 |
| Ratio | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | [illegible] | [illegible] | NaN% |
| Module% | 42.95% | 270.28% | 55.17% | 141.88% | 0.29% | [illegible] | [illegible] | NaN |
| FPGA% | 171.79% | 1,081.10% | 220.67% | 567.50% | 1.17% | [illegible] | [illegible] | NaN |

Input filling rates

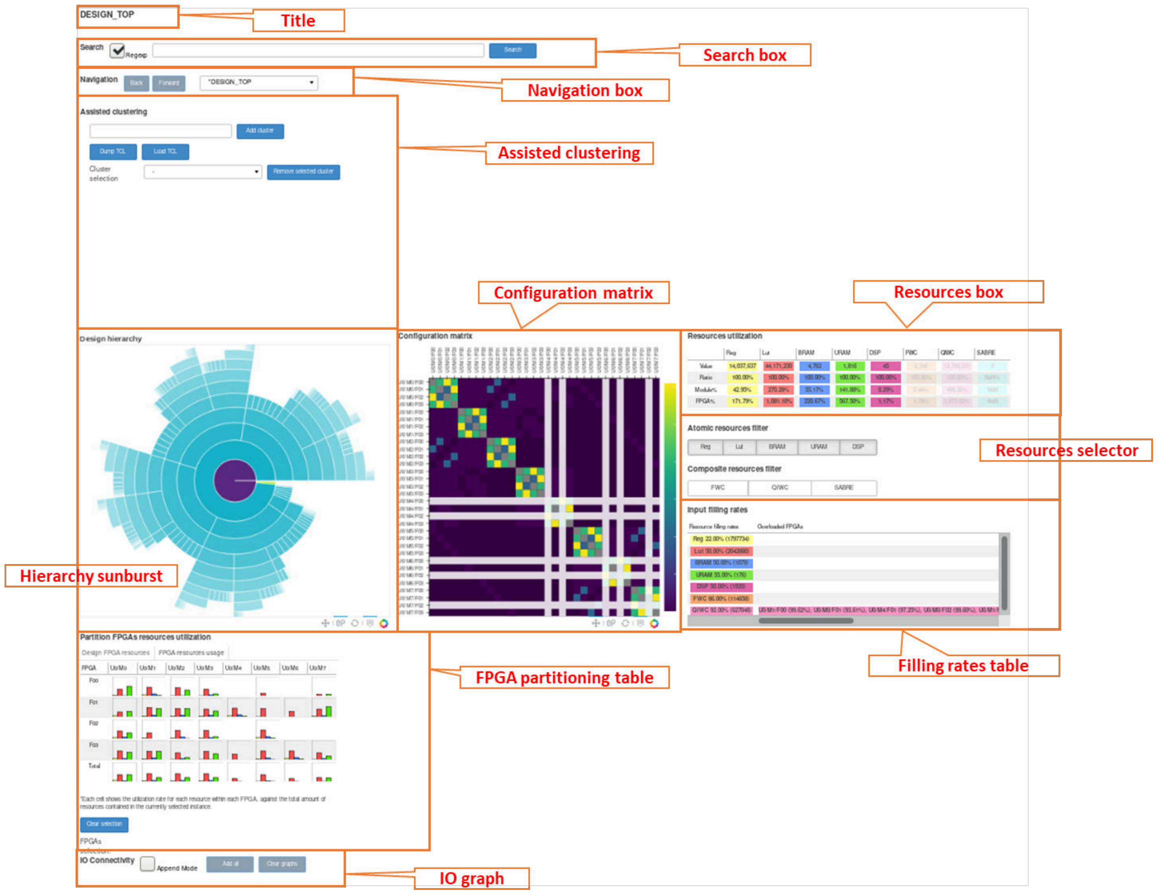
| Resource filling rates | Overloaded FPGAs |
|---|---|
| Reg 22.00% (1797734) | |
| Lut 50.00% (2042880) | |
| BRAM 50.00% (1079) | |
| URAM 55.00% (176) | |
| DSP 50.00% (1920) | |
| FWC 86.00% (114638) | |
| QIWC 92.00% (627048) | U0/M1/F00 (99.62%), U0/M0/F01 (93.61%), U0/M4/F01 (97.25%), U0/M0/F02 (99.60%), U0/M1/ |

| FPGA | U0/M0 | U0/M1 | U0/M2 | U0/M3 | U0/M4 | U0/M5 | U0/M6 | U0/M7 |
|---|---|---|---|---|---|---|---|---|
| F00 | | | | | | | | |
| F01 | | | | | | | | |
| F02 | | | | | | | | |
| F03 | | | | | | | | |
| Total | | | | | | | | |

*Each cell shows the utilization rate for each resource within each FPGA, against the total amount of resources contained in the currently selected instance.

The page is divided into 11 components as shown in the following figure:

*Figure 2 Partition viewer HTML page components*

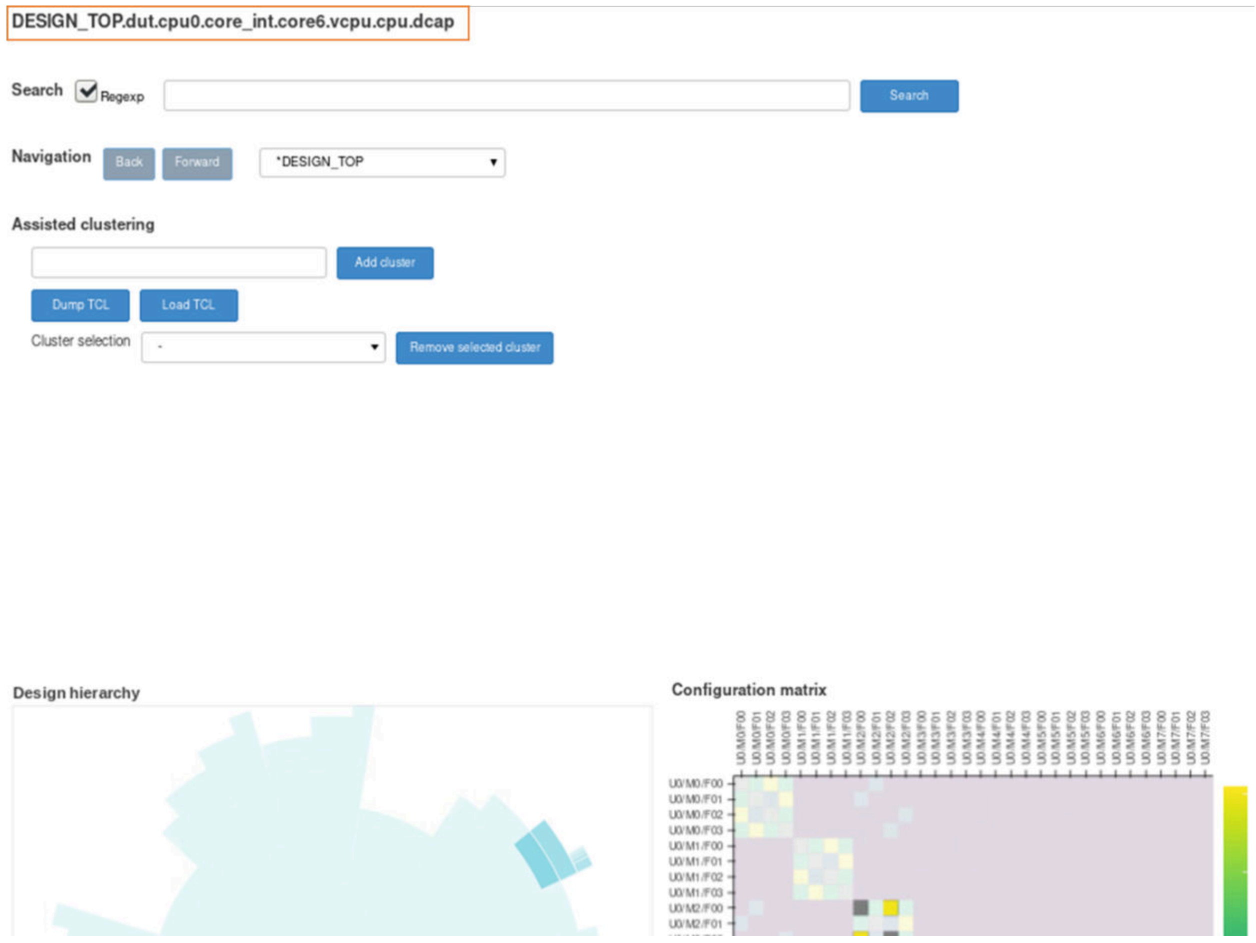


## Title

This component shows either the name of the top instance selected (the context instance) into the hierarchy sunburst (component 6), or, if the mouse cursor is hovering the sunburst, the path of the currently hovered instance.

It is possible to copy the text to paste it in another tool, email, and so on.

*Figure 3    Partition viewer HTML page title*



## Search Box

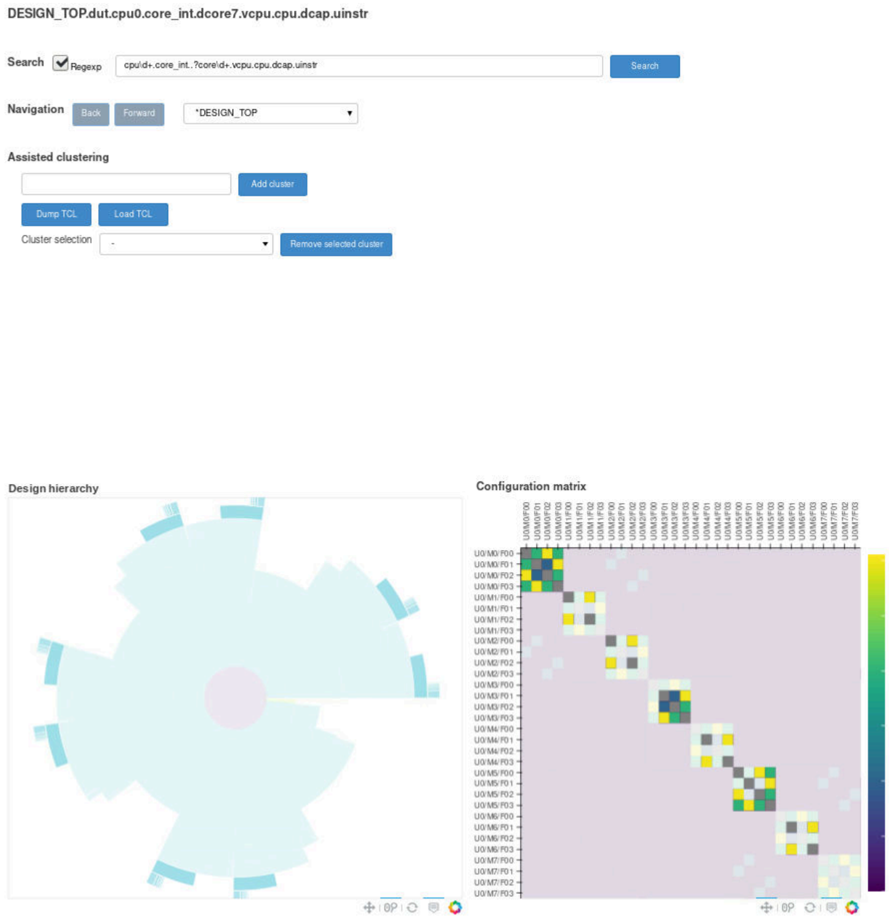
It allows the users to search for any instance in the hierarchy according to a pattern. The latter can be either a regular expression if the **Regexp** check box checked, or a **Fnmatch** pattern otherwise.

The search is fired as soon as the user presses the Enter key in the text box, or clicks on the **Search** button.

Search matches are all highlighted within the hierarchy sunburst (component 6), or, if there is no matches, all the instances are highlighted. The configuration matrix (component 7) and the FPGA table (component 9) also react to the search, highlighting only the FPGAs on which the matching instances are partitioned.

Example with regular expression pattern `cpu\d+.core_int..?core\d+.vcpu.cpu.dcap.uinstr` is shown as follows:

*Figure 4 Search box regexp pattern 1*



DESIGN_TOP.dut.cpu0.core_int.dcore7.vcpu.cpu.dcap.uinstr

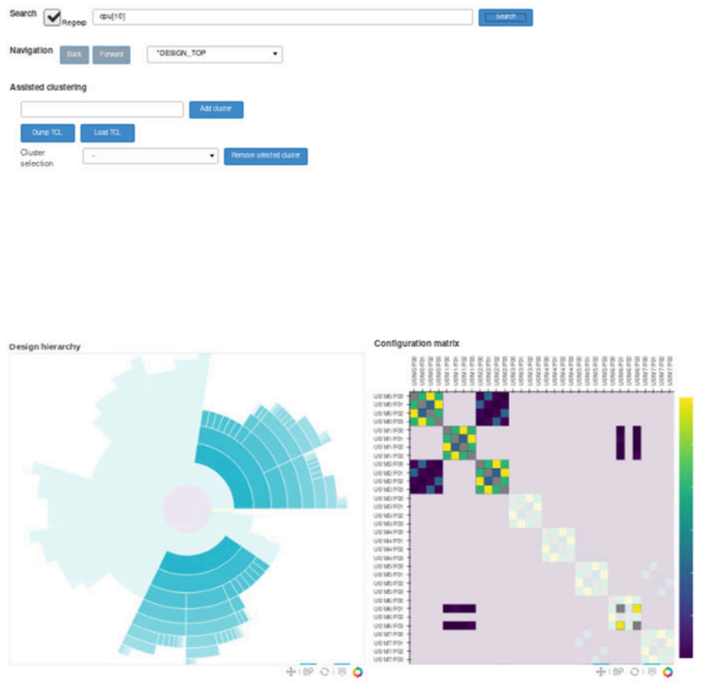
Search ✓ Regexp `cpu\d+.core_int..?core\d+.vcpu.cpu.dcap.uinstr`

Navigation: *DESIGN_TOP

Cluster selection: -

Another example with regular expression `cpu[10]` is shown as follows:

*Figure 5        Search box regexp pattern 2*

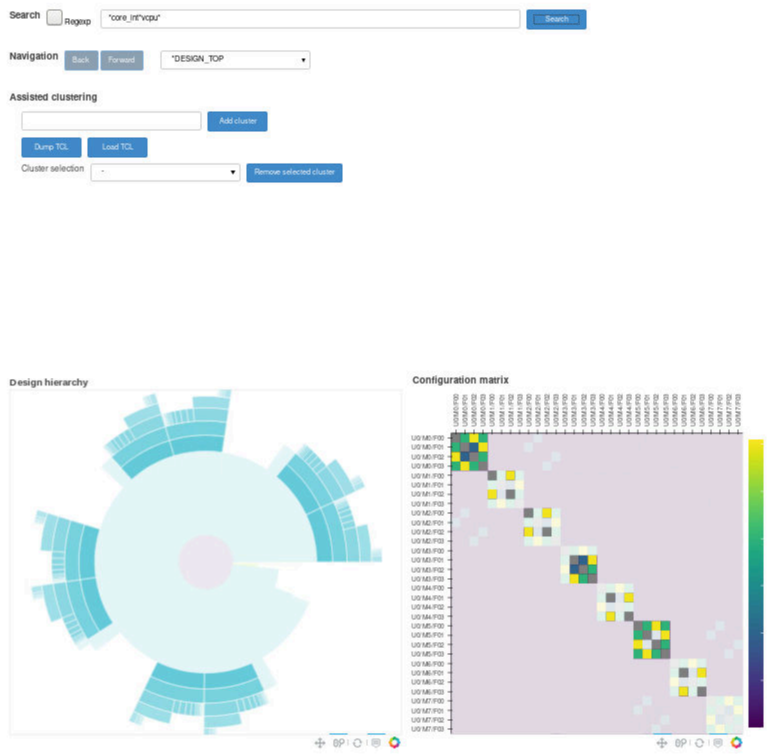


Search ✓ Regexp cpu[10]

Navigation *DESIGN_TOP

Example with **Fnmatch** pattern `*core_int*vcpu*` is shown as follows:

*Figure 6 Search box fnmatch pattern*



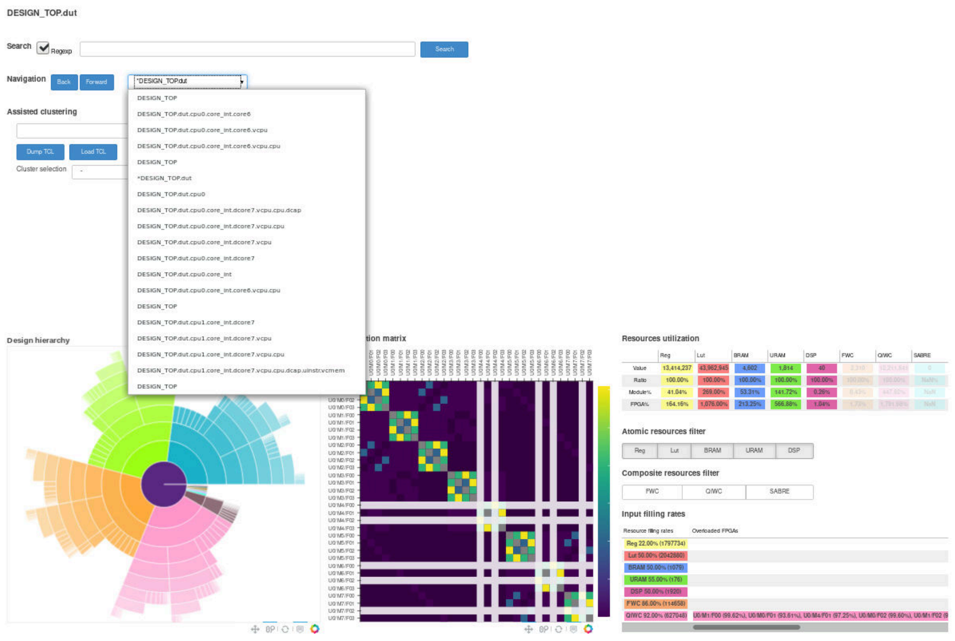
## Navigation Box

The aim of this component is to record the user's moves as user navigates through the hierarchy sunburst (component 6).

Three widgets compound this box:

- Back button: to go back one step in the history, if any. If it is possible to go back, the button is enabled, otherwise it is disabled.
- Forward button: to advance one step in the history, if any. If it ispossible to go forward, the button is enabled, otherwise it is disabled.
- Drop-down list: it contains the whole navigation history. An asterisk precedes the name of the currently selected instance (the context). When an instance is selected from this drop-down list, navigation jumps to that instance, and the newly selected instance is marked with an aster

*Figure 7     Navigation box drop-down list*


DESIGN_TOP.dut
Search
Regexp
Search
Navigation
Back
Forward
*DESIGN_TOP.dut
DESIGN_TOP
DESIGN_TOP.dut.cpu0.core_int.core6
DESIGN_TOP.dut.cpu0.core_int.core6.vcpu
DESIGN_TOP.dut.cpu0.core_int.core6.vcpu.cpu
DESIGN_TOP
*DESIGN_TOP.dut
DESIGN_TOP.dut.cpu0
DESIGN_TOP.dut.cpu0.core_int.dcore7.vcpu.cpu.dcap
DESIGN_TOP.dut.cpu0.core_int.dcore7.vcpu.cpu
DESIGN_TOP.dut.cpu0.core_int.dcore7.vcpu
DESIGN_TOP.dut.cpu0.core_int.dcore7
DESIGN_TOP.dut.cpu0.core_int
DESIGN_TOP.dut.cpu0.core_int.core6.vcpu.cpu
DESIGN_TOP
DESIGN_TOP.dut.cpu1.core_int.dcore7
DESIGN_TOP.dut.cpu1.core_int.dcore7.vcpu
DESIGN_TOP.dut.cpu1.core_int.dcore7.vcpu.cpu
DESIGN_TOP.dut.cpu1.core_int.dcore7.vcpu.cpu.dcap.uinstr.vcmem
DESIGN_TOP
Assisted clustering
Dump TCL
Load TCL
Cluster selection
Design hierarchy
Resources utilization
Atomic resources filter
Reg
Lut
BRAM
URAM
DSP
Composite resources filter
FWC
QIWC
SABRE
Input filling rates

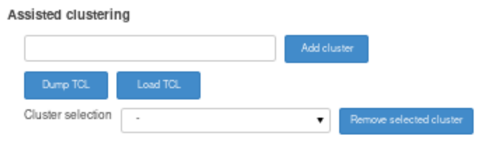
Each time the user navigates through this history (clicks on one of the buttons or selects an instance in the list), the overall context instance changes and all of the page's components are updated accordingly.

## Assisted Clustering

This tool allows the users to create several groups of instances, also called clusters, to populate them using the hierarchy sunburst, and to dump them all to a Tcl containing zTopBuild clustering commands.

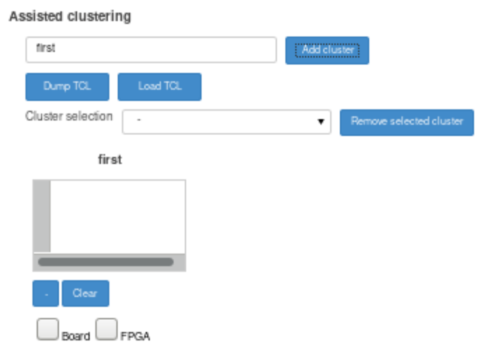
The tool's initial panel is shown as follows:

*Figure 8     Assisted clustering initial panel*



From that point, the user chooses a cluster name and click **Add cluster** button. The name must not be already used.

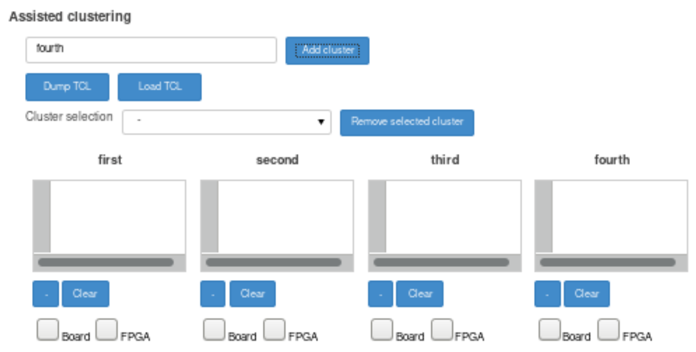
*Figure 9 Newly created cluster*



A cluster is divided into following four parts:

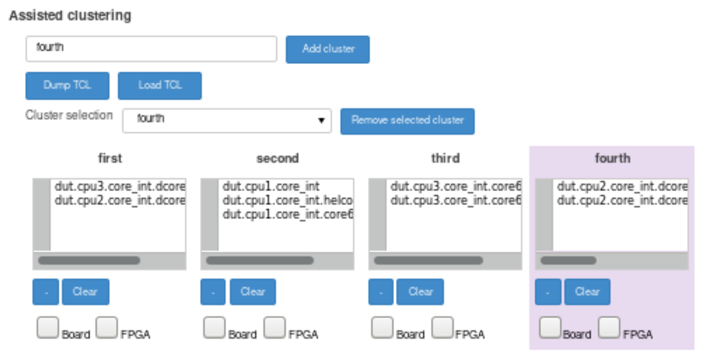
- The first contains a radio button and a title showing the cluster name. The radio button is used the cluster selection.
- The second is a select widget in which the instances will be added.
- The third contains three buttons.
  - **-** is for removing an instance from the cluster
  - **Clear** to remove all the instances in one click
  - **Remove cluster** to remove completely the current cluster
- The fourth contains two check boxes.
  - **Board** to notice the cluster will be applying to zTopBuild clustering
  - **FPGA** to zCoreBuild clustering.

*Figure 10 Four clusters added to the assisted clustering tool*



From there, the user can select the cluster to fill in, and then select the desired instance by clicking it directly in the sunburst.

*Figure 11 Instances added to the clusters*



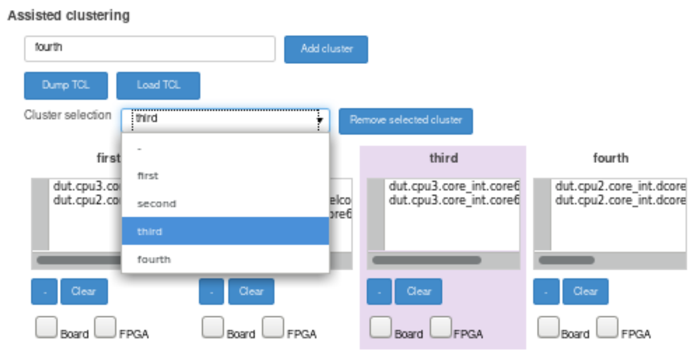
fourth

fourth

first: dut.cpu3.core_int.dcore, dut.cpu2.core_int.dcore
second: dut.cpu1.core_int, dut.cpu1.core_int.helco, dut.cpu1.core_int.core6
third: dut.cpu3.core_int.core6, dut.cpu3.core_int.core6
fourth: dut.cpu2.core_int.dcore, dut.cpu2.core_int.dcore

The fact that a cluster is selected (Cluster selection widget), the behavior changes slightly in the sunburst. Indeed, while a radio button is checked, it is not possible to lock any instance. Navigating through the hierarchy is still possible, but not locking instances. To re-enable the default behavior, all the radio buttons must be unchecked. This is performed by the **Unselect** button. Furthermore, when a cluster is selected, the resources box shows the amount of resources contained in this cluster. De-selecting it (selecting first item - in the selection widget) allows the resources box to return to its normal behavior.

*Figure 12 Cluster selection*



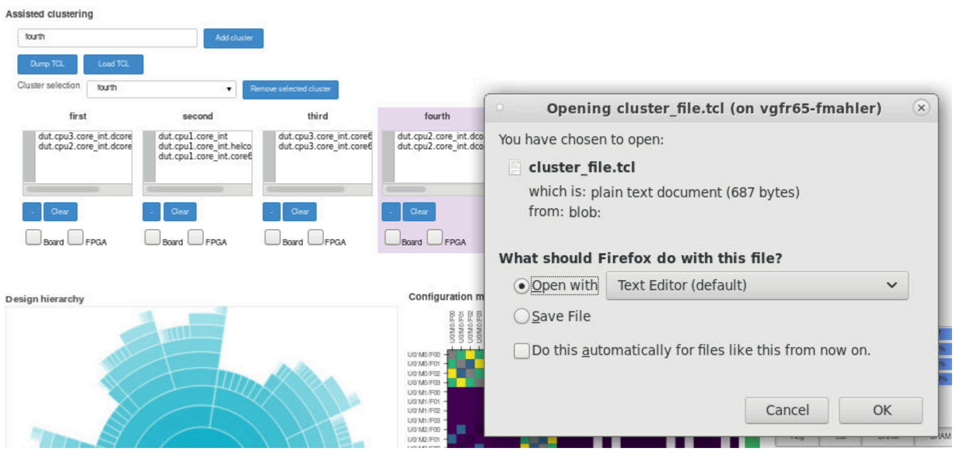
fourth

third

third: dut.cpu3.core_int.core6, dut.cpu3.core_int.core6
fourth: dut.cpu2.core_int.dcore, dut.cpu2.core_int.dcore

Once the user has filled in all the clusters as required, user can dump them into a zTopBuild-compatible Tcl script by clicking *Dump TCL* button. A popup is displayed to ask the user to choose a destination file.

*Figure 13    Dump TCL action file chooser*



Opening cluster_file.tcl (on vgfr65-fmahler)

You have chosen to open:

**cluster_file.tcl**

which is: plain text document (687 bytes)

from: blob:

**What should Firefox do with this file?**

(●) Open with [Text Editor (default)]

( ) Save File

[ ] Do this automatically for files like this from now on.

Cancel    OK

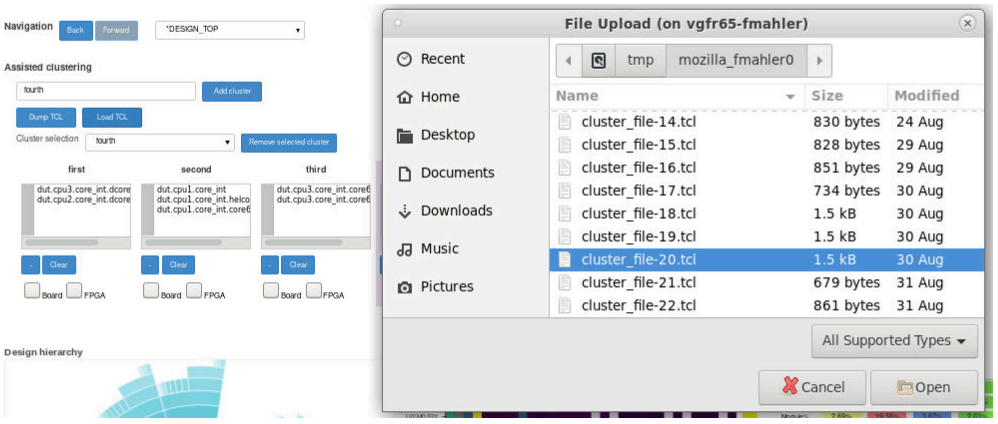
In this example, the file is shown as follows:

*Figure 14    Dumped TCL file example*

```
#
# Automatically generated on 9/8/2023, 12:06:03 PM
# It must be called within a ZTB script this way:
#    ztopbuild -advanced_command_file "cluster_file.tcl"
#

cluster_constraint add_group {first} -fnmatch -path_list {*.dut.cpu3.core_int.dcore7.vcpu *.dut.cpu2.core_int.dcore7}
cluster_constraint add_group {second} -fnmatch -path_list {*.dut.cpu1.core_int *.dut.cpu1.core_int.helcore5 *.dut.cpu1.core_int.core6.vcpu}
cluster_constraint add_group {third} -fnmatch -path_list {*.dut.cpu3.core_int.core6 *.dut.cpu3.core_int.core6.vcpu}
cluster_constraint add_group {fourth} -fnmatch -path_list {*.dut.cpu2.core_int.dcore7.vcpu.cpu *.dut.cpu2.core_int.dcore7.vcpu.cpu.dcap.uinstr.ucmem}
```

The tool also has the capability to load a previously stored TCL file. Click the *Load TCL* button. A popup will be displayed to choose the file, and the parsed clusters will take place in the GUI like any other cluster. Note that if there were already clusters, they will be removed and replaced.

*Figure 15    Load TCL action file chooser.*



File Upload (on vgfr65-fmahler)

| Name | Size | Modified |
|---|---|---|
| cluster_file-14.tcl | 830 bytes | 24 Aug |
| cluster_file-15.tcl | 828 bytes | 29 Aug |
| cluster_file-16.tcl | 851 bytes | 29 Aug |
| cluster_file-17.tcl | 734 bytes | 30 Aug |
| cluster_file-18.tcl | 1.5 kB | 30 Aug |
| cluster_file-19.tcl | 1.5 kB | 30 Aug |
| cluster_file-20.tcl | 1.5 kB | 30 Aug |
| cluster_file-21.tcl | 679 bytes | 31 Aug |
| cluster_file-22.tcl | 861 bytes | 31 Aug |

## Resources Box and Resources Selector

It is a table showing how many resources of each type the context instance contains.

The columns are the types of resources, and there are four rows:

- The first row, **Value**, is the amount of each kind of resources within the current instance.
- The second row, **Ratio**, is the percentage of each type of resource among the total counted in the context instance for this same type.
- The third row, **Module%**, represents the percentage of each kind of resource among the total amount of resources available within one module.
- Similarly, the fourth row, **FPGA%**, represents the percentage of each kind of resource among the total amount of resources available within one FPGA.

The last two rows allow the user to answer the questions:

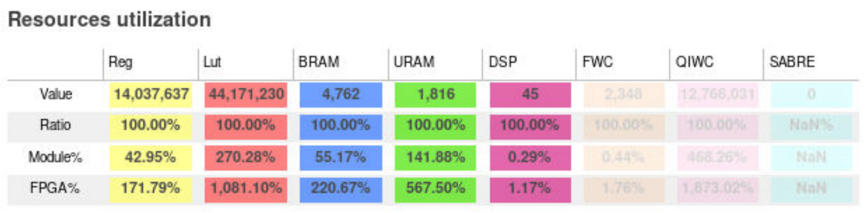
- Could this instance sit in one module?
- In one FPGA?, If not, which resources overload the module? The FPGA?

When the context instance changes, the first row changes accordingly and all of the cells of the second row show 100%.

When the mouse is hovering instances from the sunburst (component 6), the first row changes to show amount of resources from the hovered instance, and the second row also changes to show the percentage of resources among the context instance resources, as well as the last two rows.

**Example**: the first figure shows the resources for the context instance, and the second one shows while hovering sub-instances.

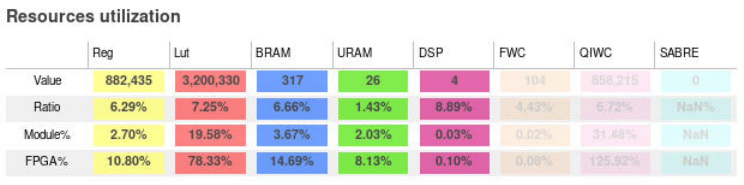
*Figure 16 Resources box at top level*



Resources utilization

| | Reg | Lut | BRAM | URAM | DSP | FWC | QIWC | SABRE |
|---|---|---|---|---|---|---|---|---|
| Value | 14,037,637 | 44,171,230 | 4,762 | 1,816 | 45 | 2,348 | 12,768,031 | 0 |
| Ratio | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | NaN% |
| Module% | 42.95% | 270.28% | 55.17% | 141.88% | 0.29% | 0.44% | 468.26% | NaN |
| FPGA% | 171.79% | 1,081.10% | 220.67% | 567.50% | 1.17% | 1.76% | 1,873.02% | NaN |

*Figure 17 Resources box at lower level.*



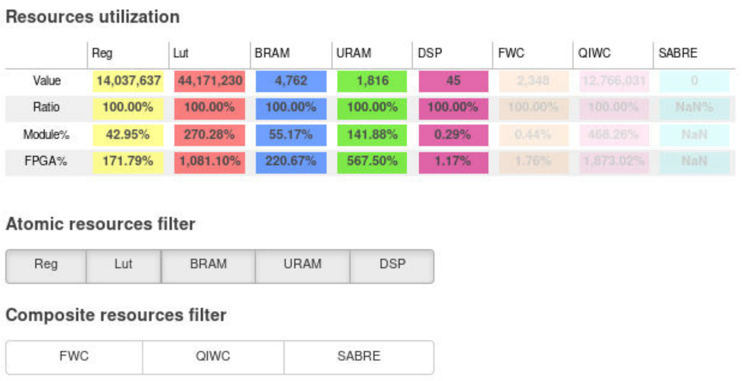
**Resources utilization**

| | Reg | Lut | BRAM | URAM | DSP | FWC | QIWC | SABRE |
|---|---|---|---|---|---|---|---|---|
| Value | 882,435 | 3,200,330 | 317 | 26 | 4 | 104 | 858,215 | 0 |
| Ratio | 6.29% | 7.25% | 6.66% | 1.43% | 8.89% | 4.43% | 6.72% | NaN% |
| Module% | 2.70% | 19.58% | 3.67% | 2.03% | 0.03% | 0.02% | 31.48% | NaN |
| FPGA% | 10.80% | 78.33% | 14.69% | 8.13% | 0.10% | 0.08% | 125.92% | NaN |

Similarly, when a cluster is selected from Assisted Clustering tool (section 2.4), the resource box is updated to show the amount of resources contained in that cluster. When the cluster is deselected, the resource box behavior returns to normal.

Below that table, there are two sets of buttons allowing to enable and disable resources from the above table. That's the resources selector.

*Figure 18 Resources box and resources selector*



**Resources utilization**

| | Reg | Lut | BRAM | URAM | DSP | FWC | QIWC | SABRE |
|---|---|---|---|---|---|---|---|---|
| Value | 14,037,637 | 44,171,230 | 4,762 | 1,816 | 45 | 2,348 | 12,766,031 | 0 |
| Ratio | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | NaN% |
| Module% | 42.95% | 270.28% | 55.17% | 141.88% | 0.29% | 0.44% | 468.26% | NaN |
| FPGA% | 171.79% | 1,081.10% | 220.67% | 567.50% | 1.17% | 1.76% | 1,873.02% | NaN |

The tool differentiates between atomic and composite resources. Composite ones are build from several atomic resources. So far, atomic resources are registers "Reg", LUTs, BRAMs, URAMs and DSPs. Composite are FWC, QIWC and SABRE. These lists may be extended in the future.

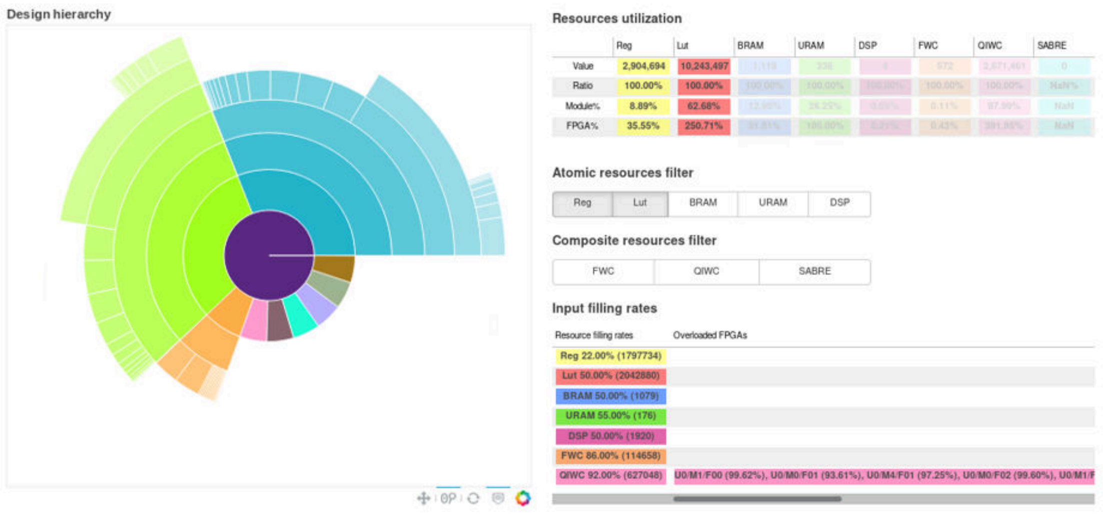
The user can select all the combinations of buttons he wants among Atomic. Equally, can select all the combinations of buttons he wants among Composite. But Atomic and Composite are mutually exclusive. That means no buttons from Atomic list can be checked at the same time as Composite list buttons, and vice versa.

Whenever a button is selected, the resources box is updated by highlighting only checked resources and hide the others. Similarly, FPGA partitioning table (component 9) is also updated by showing only checked resources and hiding the others. Since the surface of hierarchy sunburst glyphs (component 6) is computed according to the selected resources, the sunburst is also updated with the selection.

Feedback

Example of Reg and LUTs buttons selected is shown as follows:

*Figure 19 Resources selector enabled*



**Resources utilization**

| | Reg | Lut | BRAM | URAM | DSP | FWC | QIWC | SABRE |
|---|---|---|---|---|---|---|---|---|
| Value | 2,904,694 | 10,243,497 | 1,118 | 338 | 8 | 572 | 2,671,461 | 0 |
| Ratio | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | NaN% |
| Module% | 8.89% | 62.68% | 12.90% | 26.25% | 0.03% | 0.11% | 97.99% | NaN |
| FPGA% | 35.55% | 250.71% | 31.81% | 100.00% | 0.21% | 0.43% | 391.85% | NaN |

**Atomic resources filter**

Reg | Lut | BRAM | URAM | DSP

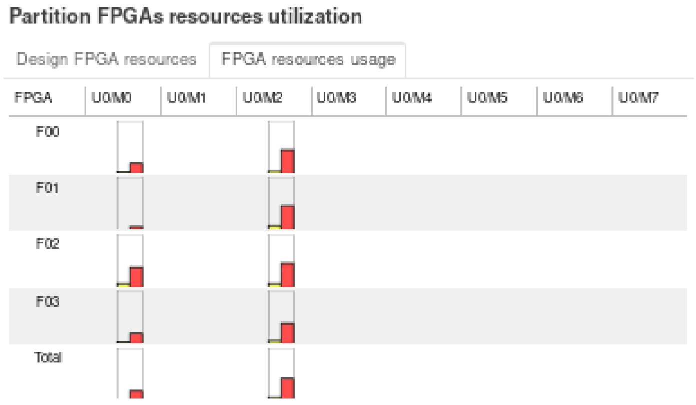
**Composite resources filter**

FWC | QIWC | SABRE

**Input filling rates**

| Resource filling rates | Overloaded FPGAs |
|---|---|
| Reg 22.00% (1797734) | |
| Lut 50.00% (2042880) | |
| BRAM 50.00% (1079) | |
| URAM 55.00% (176) | |
| DSP 50.00% (1920) | |
| FWC 86.00% (114658) | |
| QIWC 92.00% (627048) | U0/M1/F00 (99.62%), U0/M0/F01 (93.61%), U0/M4/F01 (97.25%), U0/M0/F02 (99.60%), U0/M1/F |

*Figure 20 Resources selector REG and LUT enabled, impact on FPGA table display.*



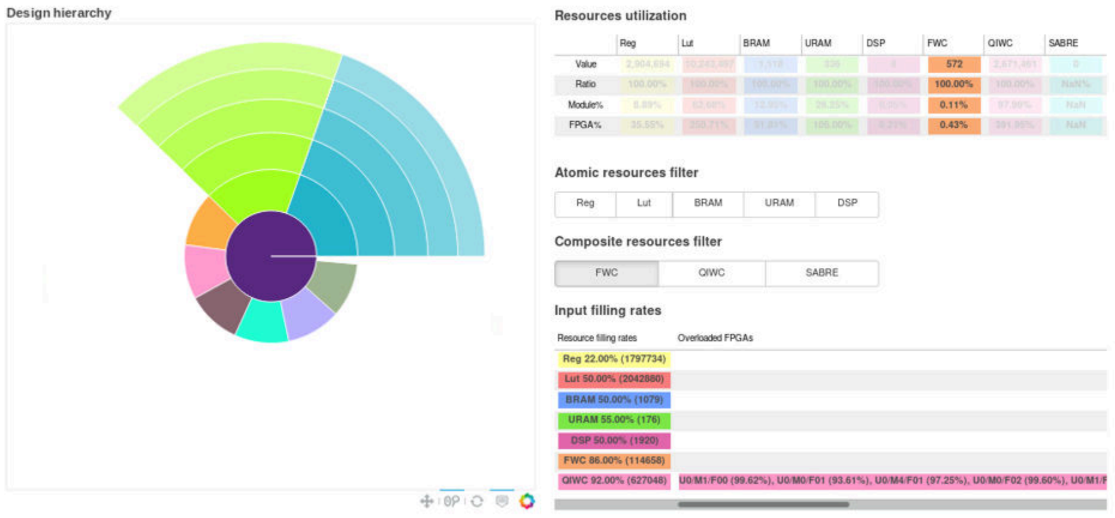
**Partition FPGAs resources utilization**

Design FPGA resources | FPGA resources usage

| FPGA | U0/M0 | U0/M1 | U0/M2 | U0/M3 | U0/M4 | U0/M5 | U0/M6 | U0/M7 |
|---|---|---|---|---|---|---|---|---|
| F00 | | | | | | | | |
| F01 | | | | | | | | |
| F02 | | | | | | | | |
| F03 | | | | | | | | |
| Total | | | | | | | | |

Example for same design, same context instance, FWC button selected is shown as follows:

*Figure 21 Resources selector FWC enabled, impact on sunburst.*



Resources utilization

| | Reg | Lut | BRAM | URAM | DSP | FWC | QIWC | SABRE |
|---|---|---|---|---|---|---|---|---|
| Value | 2,904,694 | 10,242,497 | 1,118 | 336 | 0 | 572 | 2,671,461 | 0 |
| Ratio | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | NaN% |
| Module% | 8.89% | 62.08% | 12.95% | 28.25% | 0.00% | 0.11% | 97.90% | NaN |
| FPGA% | 35.55% | 250.71% | 51.81% | 100.00% | 0.21% | 0.43% | 391.95% | NaN |

Atomic resources filter

Reg | Lut | BRAM | URAM | DSP

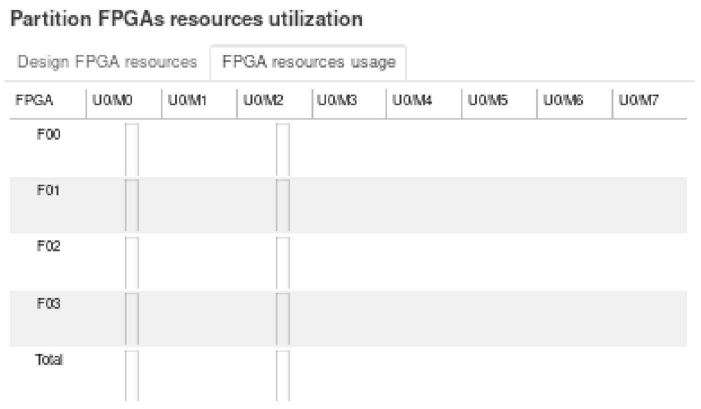
Composite resources filter

FWC | QIWC | SABRE

Input filling rates

| Resource filling rates | Overloaded FPGAs |
|---|---|
| Reg 22.00% (1797734) | |
| Lut 50.00% (2042880) | |
| BRAM 50.00% (1079) | |
| URAM 55.00% (176) | |
| DSP 50.00% (1920) | |
| FWC 86.00% (114658) | |
| QIWC 92.00% (627048) | U0/M1/F00 (99.62%), U0/M0/F01 (93.61%), U0/M4/F01 (97.25%), U0/M0/F02 (99.60%), U0/M1/F |

*Figure 22 Resources selector FWC enabled, impact on FPGA table display.*



Partition FPGAs resources utilization

Design FPGA resources | FPGA resources usage

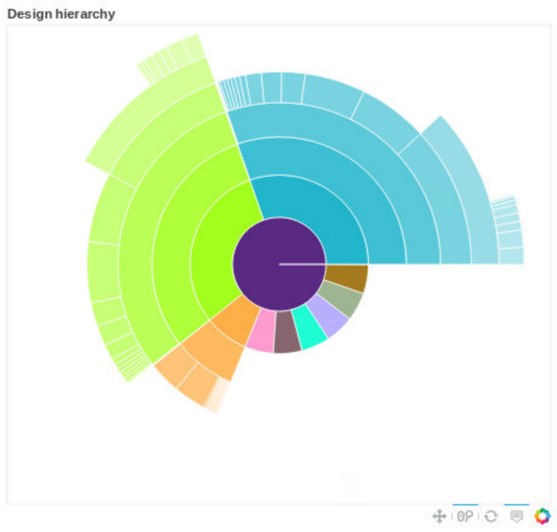
| FPGA | U0/M0 | U0/M1 | U0/M2 | U0/M3 | U0/M4 | U0/M5 | U0/M6 | U0/M7 |
|---|---|---|---|---|---|---|---|---|
| F00 | | | | | | | | |
| F01 | | | | | | | | |
| F02 | | | | | | | | |
| F03 | | | | | | | | |
| Total | | | | | | | | |

## Hierarchy Sunburst

It is one of the main pieces of the GUI. It represents the design's hierarchy as a sunburst view.

Example is shown as follows:

*Figure 23 Hierarchy sunburst component*



## Layout

The sunburst is a set of concentric arcs circling a central disk. It’s the only way to navigate through the design hierarchy.

The central disk is the context instance, and the concentric arcs are the sub-instances.

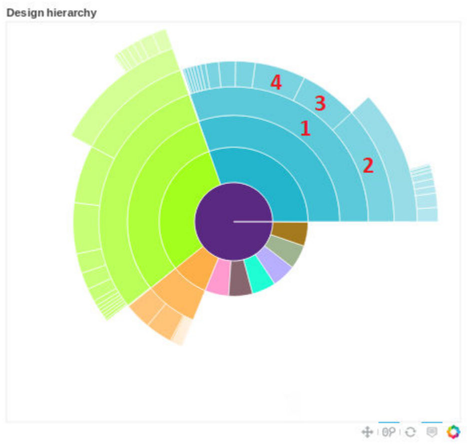
They are arranged in descending order of resource density, from the right boundary to the left boundary of the parent instance. The denser the resources, the wider the arc.

The first belt around the center contains the sub-instances of the context instance. Each sub-instance in this belt has a completely different color from the others.

Recursively, the second belt contains the sub-instances of the first belt's instances. The difference with the first belt is the color of the arcs. From the second belt onwards, arcs have the same color as their parent instance, but their transparency increases with the depth of the hierarchy.

For example, let consider the following annotated figure.

*Figure 24 Hierarchy sunburst component explanation*



The instance marked 1 is the parent instance of instances 2, 3 and 4. Instance 2 is the densest among all instance 1's sub-instances. That's why it's printed at the right boundary of instance 1. Instance 3 is the second densest, it's printed after 2. Then instance 4, and so on.

## Interactions

There are many ways to interact with the hierarchy sunburst component.

- Hovering over the arcs and central disk:

Each time an instance of the sunburst is hovered, the following components are updated:

- Title (component 1): the hovered instance's path is updated.
- Configuration matrix (component 7):
- FPGAs, on which the hovered instance is partitioned, are highlighted.
- Resources box (component 4): the selected resources values are updated as well as the ratios against the context instance.
- FPGA partitioning table (component 9): the FPGAs, on which the hovered instance is partitioned, are displayed. The others are hidden.
- IO graph (component 10): if the graph is displayed, the hovered instance, all the links connected to it, and the instances connected to those links, are highlighted.
- Clicking on one arc of on central disk:

This action locks the current selection. So that the hovering interactions are disabled. If an instance is already locked, clicking somewhere on the sunburst unlocks it, and re-enabled hovering interactions.

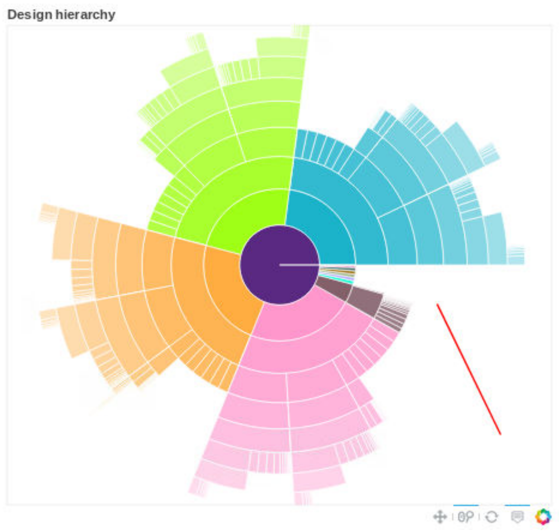
- Title (component 1): the locked instance’s path is updated.
- Configuration matrix (component 7): the FPGAs, on which the
- Locked instance is partitioned, are highlighted.
- Resources box (component 4): the
- Locked instance’s resources values sare updated as well as the ratios against the context instance.
- FPGA partitioning table (component 9): The FPGAs, on which the locked instance is partitioned, are displayed. The others are hidden.
- IO graph (component 10): if the graph is displayed, the locked instance, all the links connected to it, and the instances connected to those links, are highlighted.
- Double-clicking on one arc of on central disk:

This action allows to enter the selected instance. The latter becomes the new context instance, and the GUI is updated accordingly.

- Title (component 1): the
- New context instance’s path is updated.
- Configuration matrix (component 7): the FPGAs, on which the new context instance is partitioned, are highlighted.
- Resources box (component 4): the new context instance’s resources values are updated as well as the ratios which all print 100%.
- FPGA partitioning table (component 9): the FPGAs, on which the new context instance is partitioned, are displayed. The others are hidden.
- Navigation box (component 3): it records the move in the hierarchy and updates its Back/Forward buttons, and drop-down list accordingly.
- Double-clicking in the sunburst, but outside the arcs and central disk: This action allows to go back to the top of the design. GUI behaves the exact same way as if one arc is double-clicked, but here the new context instance is the top of the design.
- Drag and drop anywhere into the sunburst: This allows to navigate through the netlist by entering the first densest instance (drag & drop down) or the or the parent instance (drag & drop up) of the current context instance. The new context instance becomes respectively the first densest instance or the parent instance. All the interactions are the same as previously. In addition to that, a red line is drawn on the sunburst to indicate the user the action is registered and on-going. It is the current context instance is the top-level instance, drag & drop up does nothing.

It is the current context instance has no sub-instances, drag & drop down does nothing.

*Figure 25 Hierarchy sunburst drag and drop interaction*



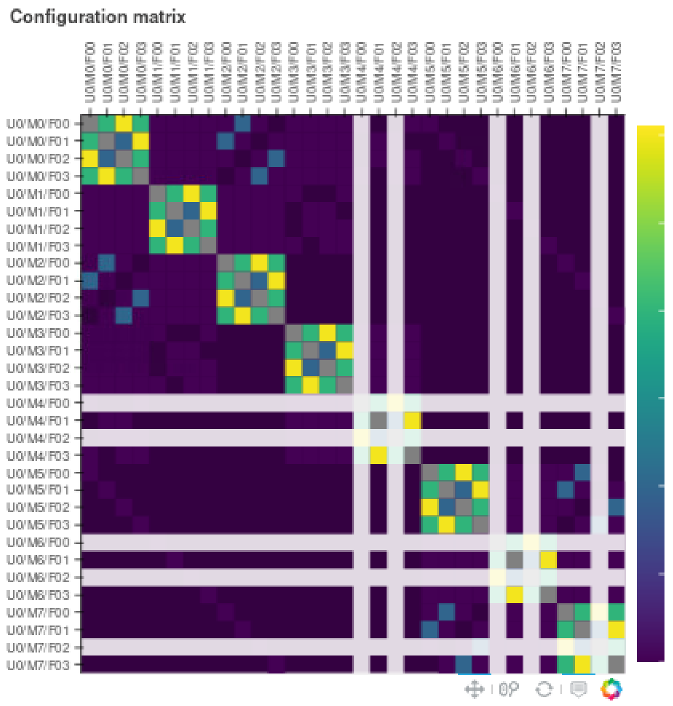
## Configuration Matrix

This square matrix shows the available interconnections between the FPGAs of the target configuration.

It’s a heat map. The warmer the color, the more interconnections there are between the two FPGAs. Thus, dark purple means no interconnections, purple means 20, blue 40, green 60, yellow 80. Gray means no applicable (there is no interconnections between an FPGA and itself, which is the case for the whole diagonal of the matrix). And we also can observe long white lines that are the hidden FPGAs, which are not involved in the partitioning of the current selected context instance.

Example of configuration matrix is shown as follows:

*Figure 26 Configuration matrix component*



In this example screenshot, it is shown that 5 FPGAs are not involved in the partitioning of the current context instance. U0/M4/F00, U0/M4/F02, U0/M6/F00, U0/M6/F02 and U0/M7/F03.

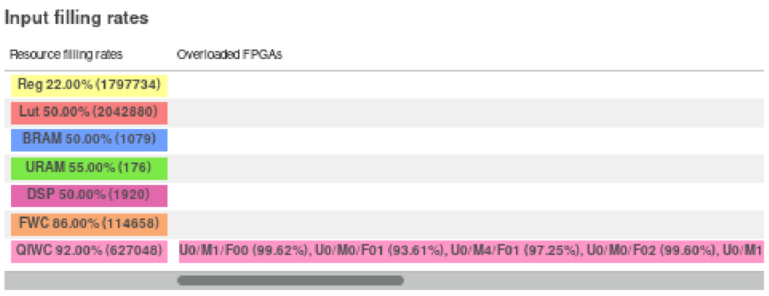
There is no notable interaction on this component.

## Filling Rates Table

This table shows, for each type of resources, a list of FPGAs overloading the maximum configured filling rate if any.

The first column contains the types of resources, the max filling rate and the corresponding absolute value between parentheses. The second column contains the list of FPGAs. Each row corresponds to a type a resource.

*Figure 27 Filling rate table*



Input filling rates

| Resource filling rates | Overloaded FPGAs |
|---|---|
| Reg 22.00% (1797734) | |
| Lut 50.00% (2042880) | |
| BRAM 50.00% (1079) | |
| URAM 55.00% (176) | |
| DSP 50.00% (1920) | |
| FWC 86.00% (114658) | |
| QIWC 92.00% (627048) | U0/M1/F00 (99.62%), U0/M0/F01 (93.61%), U0/M4/F01 (97.25%), U0/M0/F02 (99.60%), U0/M1 |

As we can see in this example, the maximum QIWC's filling rate is 92% per FPGA, and there are 9 FPGAs overloading this resource:

- U0/M1/F00 (99.62%): on this FPGA, QIWC resources utilization reaches 99.62% of all of the available QIWCs, which overloads the QIWC filling rate 92%.
- U0/M0/F01 (93.61%)
- U0/M4/F01 (97.25%)
- U0/M0/F02 (99.60%)
- U0/M1/F02 (95.97%)
- U0/M2/F02 (92.34%)
- U0/M5/F02 (98.48%)
- U0/M0/F03 (94.78%)
- U0/M6/F03 (98.47%)

There are no interactions in this component.

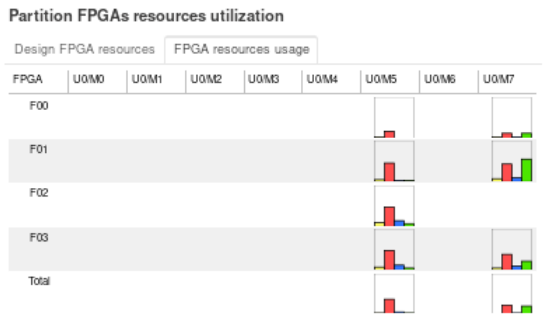
## FPGA Partitioning Table

This table is another main piece of the Partition Viewer.

It shows the set of FPGAs on which the currently selected instance (context instance, hovered instance or locked instance) is partitioned.

Example of partition of FPGA table is shown as follows:

*Figure 28 FPGA table*



In this example, we can see that the currently selected instance is partitioned on 7 FPGAs: all the FPGAs of U0/M5 module and ¾ FPGAs of U0/M7. The others are not involved in the partitioning.

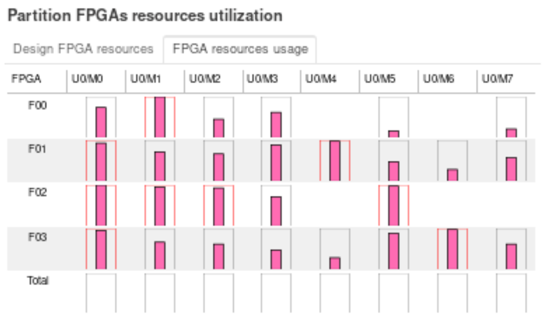
Thus, the table's columns are the modules of the configuration, and the rows the FPGAs.

The last row, **Total**, will be explained a in the following section.

As it is shown, at each FPGA spot is represented a transparent outline rectangle containing several colored smaller rectangles. The latter are histograms showing the proportion of each resource type within the FPGA. The colors are the same than those used in resources box (component 4) and filling rates table (component 8). Furthermore, only histograms corresponding to selected resources (component 5) are displayed.

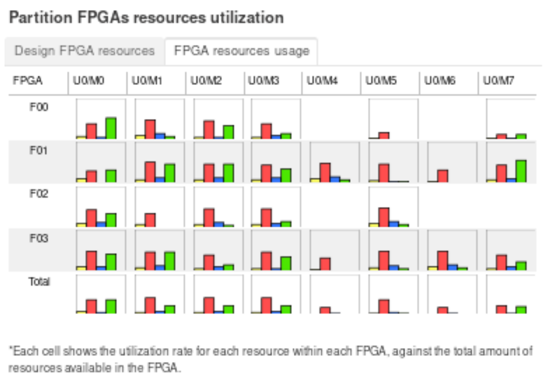
For example, here is the table displaying all the composite resources.

*Figure 29 FPGA table composite resources*



And the same displaying all the atomic resources.

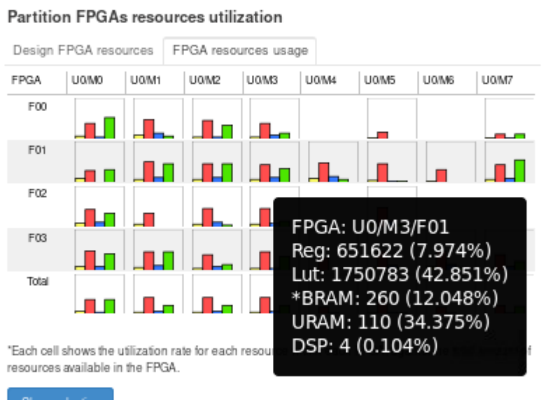
*Figure 30 FPGA table atomic resources*



*Each cell shows the utilization rate for each resource within each FPGA, against the total amount of resources available in the FPGA.

Also, there are tool tips on each FPGA rectangle providing the numerical values of the underlying histograms.

*Figure 31 FPGA table cell toolbox*



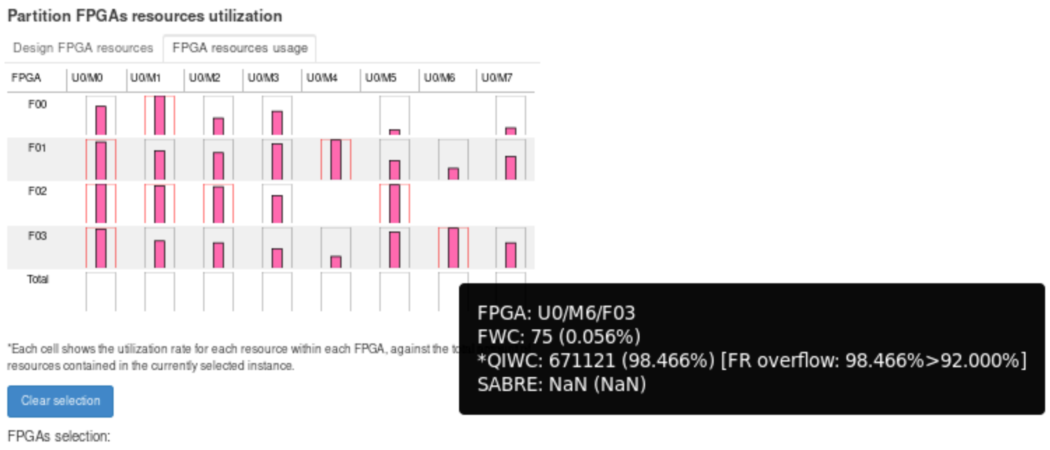
FPGA: U0/M3/F01
Reg: 651622 (7.974%)
Lut: 1750783 (42.851%)
*BRAM: 260 (12.048%)
URAM: 110 (34.375%)
DSP: 4 (0.104%)

*Each cell shows the utilization rate for each resou
resources available in the FPGA.

Sometimes, you can observe that outline rectangles are red instead of black. That means the corresponding FPGAs overload at least one resource. The tool FPGA table cell toolbox overloaded FPGAtip provide information about which ones are overloaded.

Example of FPGA table cell toolbox overloaded FPGA is shown as follows:

*Figure 32 FPGA table cell toolbox overloaded FPGA*



*Each cell shows the utilization rate for each resource within each FPGA, against the to
resources contained in the currently selected instance.

```
FPGA: U0/M6/F03
FWC: 75 (0.056%)
*QIWC: 671121 (98.466%) [FR overflow: 98.466%>92.000%]
SABRE: NaN (NaN)
```

As we can see here, 9 FPGAs are overloaded and tool tip of U0/M6/F03 shows that the culprit resource is QIWC, which filling rate is 98.466% whereas the max allowed is 92%.

In addition to that, this table has two tabs:

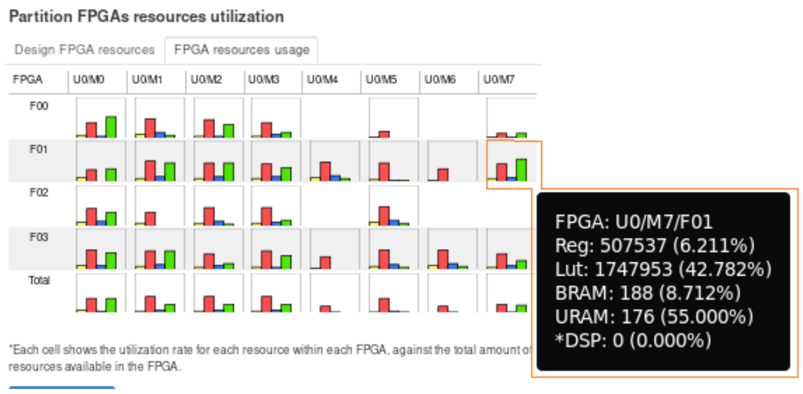
- FPGA resources usage
- Design FPGA resources

Switching between these two tabs changes the meaning of the histograms and the **Total**row.

FPGA resources usage

This tab is selected by default. Here, each histogram represents the filling rate for its resource against the FPGA available resource. For example, consider the cell of FPGA U0/M7/F01:

*Figure 33 FPGA table cell toolbox explanation*


Partition FPGAs resources utilization
Design FPGA resources
FPGA resources usage
FPGA
U0/M0
U0/M1
U0/M2
U0/M3
U0/M4
U0/M5
U0/M6
U0/M7
F00
F01
F02
F03
Total
*Each cell shows the utilization rate for each resource within each FPGA, against the total amount of resources available in the FPGA.

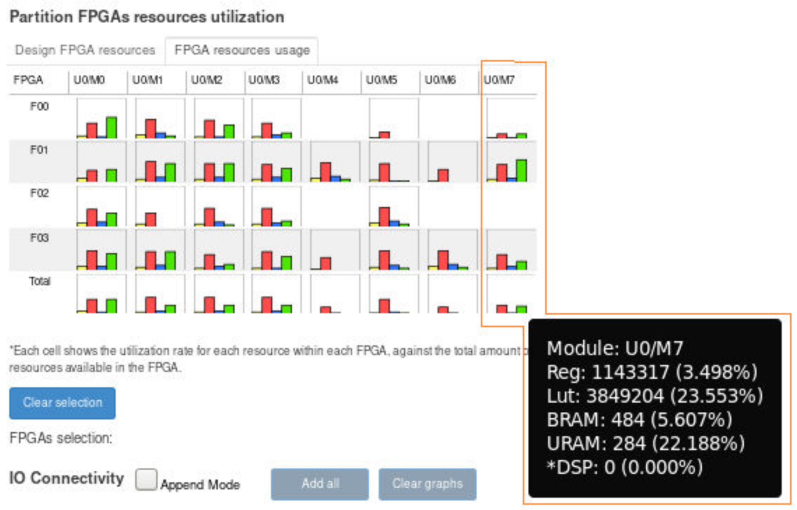
FPGA: U0/M7/F01
Reg: 507537 (6.211%)
Lut: 1747953 (42.782%)
BRAM: 188 (8.712%)
URAM: 176 (55.000%)
*DSP: 0 (0.000%)

There are 507537 registers, representing 6.211% of all of the available registers within the FPGA(a small yellow bar is drawn of the left), 1747953 LUTs, representing 42.782% of all of the available LUTs (a red bar is drawn which height is 42% the total height of the outline rectangle), 188 BRAMs and so on.

The **Total** row sums up the FPGA cells’ values and displays the filling rate against the whole module available resources.

In the same example, consider the U0/M7 module column:

*Figure 34 FPGA table module cell toolbox explanation*



Module: U0/M7
Reg: 1143317 (3.498%)
Lut: 3849204 (23.553%)
BRAM: 484 (5.607%)
URAM: 284 (22.188%)
*DSP: 0 (0.000%)

The partition of the selected instance that has been assigned to module U0/M7 contains 1143317 registers, representing 3.498% of all of the available registers within the module (a small yellow bar is drawn of the left), 3849204 LUTs, representing 23.553% of all of the available LUTs within the module (a red bar is drawn which height is almost 1/4 the total height of the outline rectangle), 484 BRAMs and so on.

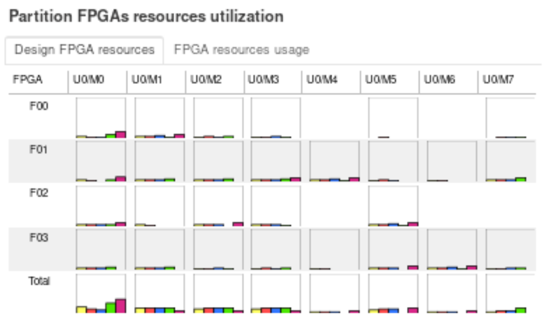
**Design FPGA resources**:

The meaning here is different. The histograms represent the resource distribution of the currently selected instance across the FPGAs involved in the partitioning. That way we can see easily whether the distribution is balanced.

The **Total** row sums up those values and prints the resource distribution of the currently selected across the modules.

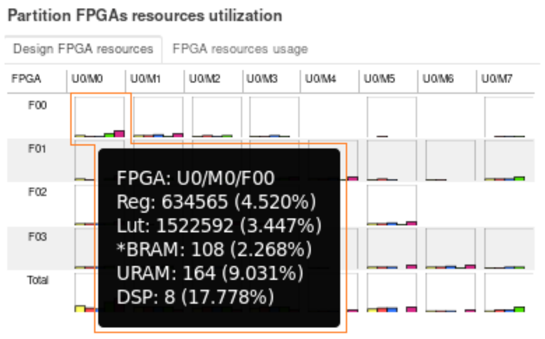
Example with the same instance than the previously is shown as follows:

*Figure 35 FPGA table, design FPGA resources explanation*



Since the selected instance is partitioned across almost all the FPGAs, the bars are all small. Let’s focus on FPGA U0/M0/F00 which looks like the most used FPGA.

*Figure 36 FPGA table design FPGA resources cell explanation*

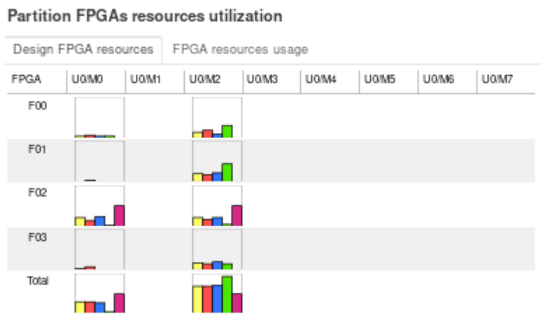


FPGA: U0/M0/F00
Reg: 634565 (4.520%)
Lut: 1522592 (3.447%)
*BRAM: 108 (2.268%)
URAM: 164 (9.031%)
DSP: 8 (17.778%)

We can read the values that way: amongst all of the registers of the instance, 634565 (= 4.520%) are assigned on FPGA U0/M0/F00, as well as 1522592 LUTs (=3.447%), 108 BRAMs, and so on.

The significant screenshot on another selected instance is shown as follows:

*Figure 37 FPGA table, design FPGA resources*



In the previous figure, it is shown that partitioning involves 8 FPGAs, 2 full modules U0/M0 and U0/M2. We can quickly deduce that resources are mostly assigned to module U0/M2.

It is possible to interact with this table by clicking on the FPGA cells. This allows the user to select one or more FPGAs in the table, and thus highlight only those sunburst instances (component 6) that are partitioned on the selected FPGAs. Other instances are hidden.

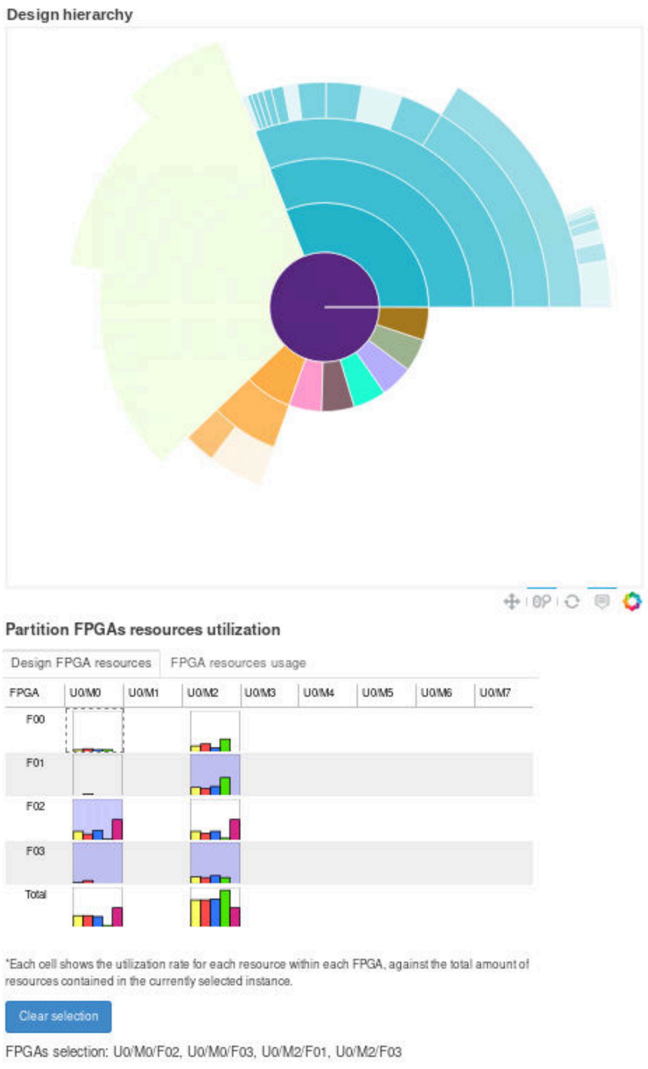
If the user clicks an already selected FPGA, that cancels the selection. Clicking a **Total** cell selects all the FPGAs of the module.

Pushing **Clear selection** button cancels all the selected FPGAs, if any.

The list of selected FPGAs is updated below the table and the resources box (component 4) contains the sum of the resources of the highlighted instances.

Example is shown as follows:

*Figure 38 FPGA table, impact of FPGA selection*



**Partition FPGAs resources utilization**

Design FPGA resources | FPGA resources usage

| FPGA | U0/M0 | U0/M1 | U0/M2 | U0/M3 | U0/M4 | U0/M5 | U0/M6 | U0/M7 |
|---|---|---|---|---|---|---|---|---|
| F00 | | | | | | | | |
| F01 | | | | | | | | |
| F02 | | | | | | | | |
| F03 | | | | | | | | |
| Total | | | | | | | | |

*Each cell shows the utilization rate for each resource within each FPGA, against the total amount of resources contained in the currently selected instance.

FPGAs selection: U0/M0/F02, U0/M0/F03, U0/M2/F01, U0/M2/F03

Here, 4 FPGAs are selected, and the corresponding instances are highlighted.

# 3

# Partition Viewer Page Generation

Partition Viewer tool can be generated in two different ways:

- Regular zCui compilation flow
- Standalone generation flow

## zCui Compilation Flow

The feature is not enabled by default in zCui. To enable it, use the following command in the UTF file before running zCui:

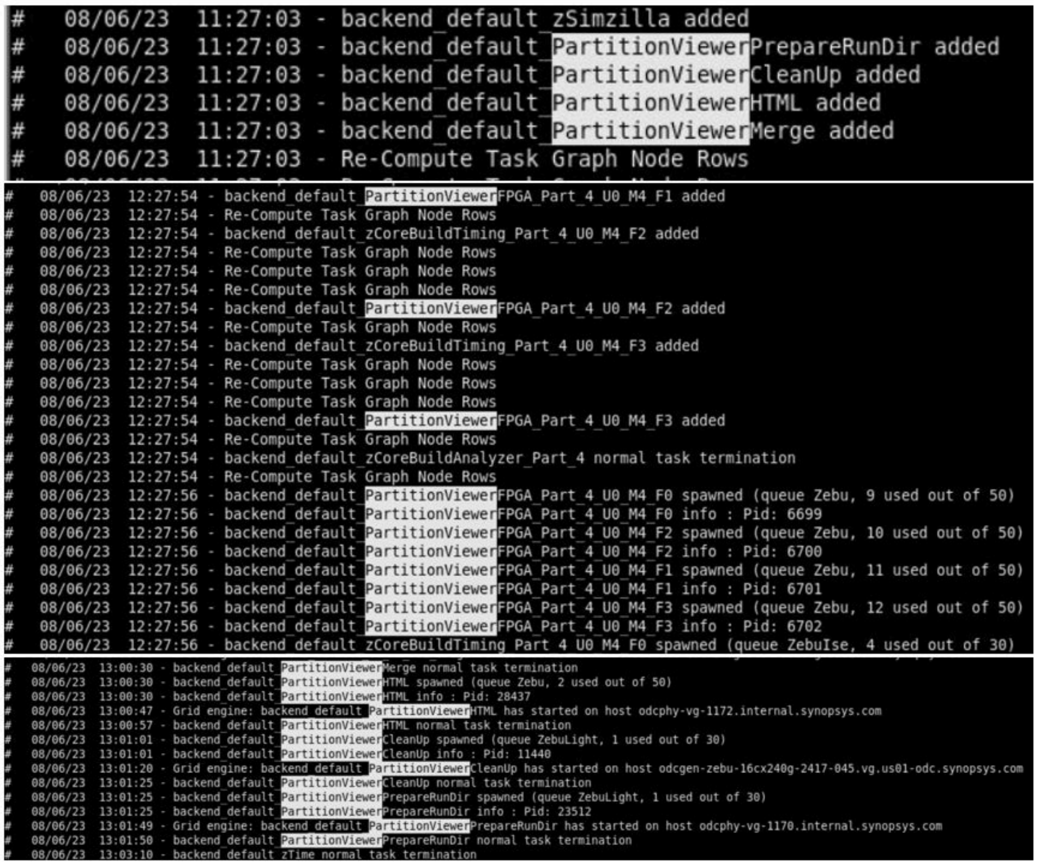
```
set_app_var internal::zcui_enable_partition_viewer true
```

zCui Partition Viewer generation flow consists in a sequence of several tasks.

First, a group of `PartitionViewerFPGA` are parallelized, one per compiled FPGA. Then, one sequential `PartitionViewerMerge` task gathering data coming from `PartitionViewerFPGA` tasks. Then `PartitionViewerHTML` is spawned to generate the actual HTML page. Finally, `PartitionViewerPrepareRunDir` is launched to prepare run directory using `nodeJS` and `PartitionViewerCleanUp` is run to remove intermediate temporary files.

Like all other zCui tasks, all Partition Viewer tasks produce a log file in `<zcui.work>/zCui/log` folder, and `zCui.log` contains references Partition Viewer tasks.

*Figure 39 zCui flow, outputs*



```
#   08/06/23  11:27:03 - backend_default_zSimzilla added
#   08/06/23  11:27:03 - backend_default_PartitionViewerPrepareRunDir added
#   08/06/23  11:27:03 - backend_default_PartitionViewerCleanUp added
#   08/06/23  11:27:03 - backend_default_PartitionViewerHTML added
#   08/06/23  11:27:03 - backend_default_PartitionViewerMerge added
#   08/06/23  11:27:03 - Re-Compute Task Graph Node Rows
```

```
#   08/06/23  12:27:54 - backend_default_PartitionViewerFPGA_Part_4_U0_M4_F1 added
#   08/06/23  12:27:54 - Re-Compute Task Graph Node Rows
#   08/06/23  12:27:54 - backend_default_zCoreBuildTiming_Part_4_U0_M4_F2 added
#   08/06/23  12:27:54 - Re-Compute Task Graph Node Rows
#   08/06/23  12:27:54 - Re-Compute Task Graph Node Rows
#   08/06/23  12:27:54 - Re-Compute Task Graph Node Rows
#   08/06/23  12:27:54 - backend_default_PartitionViewerFPGA_Part_4_U0_M4_F2 added
#   08/06/23  12:27:54 - Re-Compute Task Graph Node Rows
#   08/06/23  12:27:54 - backend_default_zCoreBuildTiming_Part_4_U0_M4_F3 added
#   08/06/23  12:27:54 - Re-Compute Task Graph Node Rows
#   08/06/23  12:27:54 - Re-Compute Task Graph Node Rows
#   08/06/23  12:27:54 - Re-Compute Task Graph Node Rows
#   08/06/23  12:27:54 - backend_default_PartitionViewerFPGA_Part_4_U0_M4_F3 added
#   08/06/23  12:27:54 - Re-Compute Task Graph Node Rows
#   08/06/23  12:27:54 - backend_default_zCoreBuildAnalyzer_Part_4 normal task termination
#   08/06/23  12:27:54 - Re-Compute Task Graph Node Rows
#   08/06/23  12:27:56 - backend_default_PartitionViewerFPGA_Part_4_U0_M4_F0 spawned (queue Zebu, 9 used out of 50)
#   08/06/23  12:27:56 - backend_default_PartitionViewerFPGA_Part_4_U0_M4_F0 info : Pid: 6699
#   08/06/23  12:27:56 - backend_default_PartitionViewerFPGA_Part_4_U0_M4_F2 spawned (queue Zebu, 10 used out of 50)
#   08/06/23  12:27:56 - backend_default_PartitionViewerFPGA_Part_4_U0_M4_F2 info : Pid: 6700
#   08/06/23  12:27:56 - backend_default_PartitionViewerFPGA_Part_4_U0_M4_F1 spawned (queue Zebu, 11 used out of 50)
#   08/06/23  12:27:56 - backend_default_PartitionViewerFPGA_Part_4_U0_M4_F1 info : Pid: 6701
#   08/06/23  12:27:56 - backend_default_PartitionViewerFPGA_Part_4_U0_M4_F3 spawned (queue Zebu, 12 used out of 50)
#   08/06/23  12:27:56 - backend_default_PartitionViewerFPGA_Part_4_U0_M4_F3 info : Pid: 6702
#   08/06/23  12:27:56 - backend_default_zCoreBuildTiming_Part_4_U0_M4_F0 spawned (queue ZebuIse, 4 used out of 30)
```

```
#   08/06/23  13:00:30 - backend_default_PartitionViewerMerge normal task termination
#   08/06/23  13:00:30 - backend_default_PartitionViewerHTML spawned (queue Zebu, 2 used out of 50)
#   08/06/23  13:00:30 - backend_default_PartitionViewerHTML info : Pid: 28437
#   08/06/23  13:00:47 - Grid engine: backend_default_PartitionViewerHTML has started on host odcphy-vg-1172.internal.synopsys.com
#   08/06/23  13:00:57 - backend_default_PartitionViewerHTML normal task termination
#   08/06/23  13:01:01 - backend_default_PartitionViewerCleanUp spawned (queue ZebuLight, 1 used out of 30)
#   08/06/23  13:01:01 - backend_default_PartitionViewerCleanUp info : Pid: 11440
#   08/06/23  13:01:20 - Grid engine: backend_default_PartitionViewerCleanUp has started on host odcgen-zebu-16cx240g-2417-045.vg.us01-odc.synopsys.com
#   08/06/23  13:01:25 - backend_default_PartitionViewerCleanUp normal task termination
#   08/06/23  13:01:25 - backend_default_PartitionViewerPrepareRunDir spawned (queue ZebuLight, 1 used out of 30)
#   08/06/23  13:01:25 - backend_default_PartitionViewerPrepareRunDir info : Pid: 23512
#   08/06/23  13:01:49 - Grid engine: backend_default_PartitionViewerPrepareRunDir has started on host odcphy-vg-1170.internal.synopsys.com
#   08/06/23  13:01:50 - backend_default_PartitionViewerPrepareRunDir normal task termination
#   08/06/23  13:03:10 - backend_default_zTime normal task termination
```

The output of the zCui Partition Viewer generation is located in `<zcui.work>/zebu.work/tools/partition_viewer`.

To open the Partition Viewer page, either run:

```
firefox<zcui.work>/zebu.work/tools/partition_viewer/run/tool.html
```

Or, if the environment has access to `nodeJS`:

```
cd <zcui.work>/zebu.work/tools/partition_viewer/run; ./run.sh
```

## Standalone Generation Flow

This flow is written in Python that uses the exact same code than zCui Partition Viewer tasks.

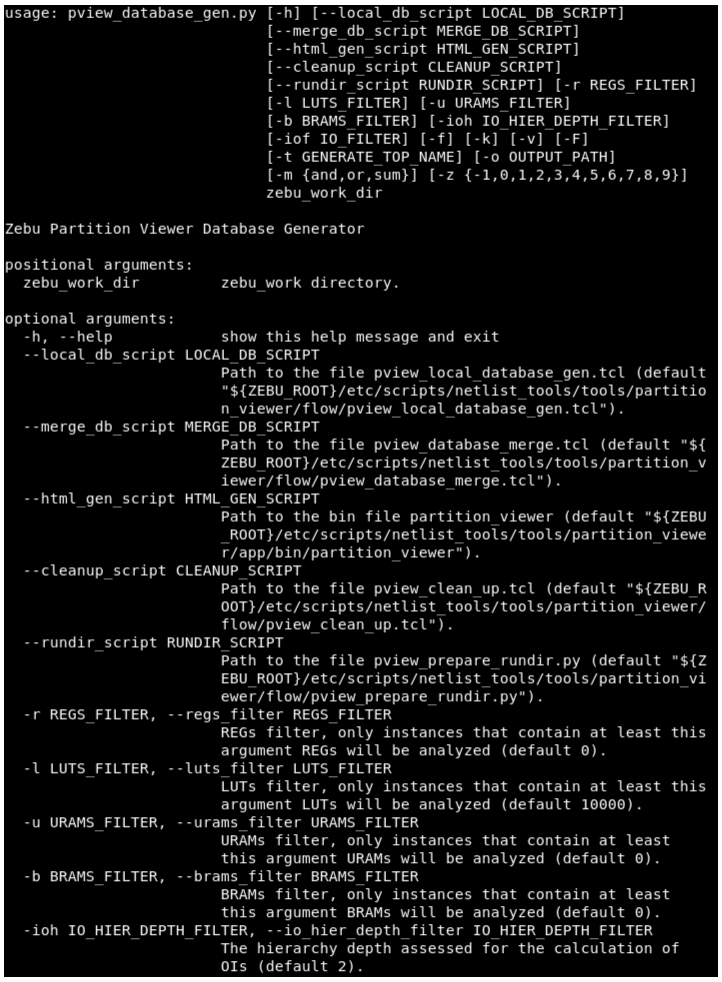
It can be executed using a completed compilation directory.

Here is a typical usage of the command:

```
?> source <zebu.release>
?> zPython --pythonpath ${ZEBU_ROOT}/etc/scripts/netlist_tools
 ${ZEBU_ROOT}/etc/scripts/netlist_tools/tools/partition_viewer/flow/pview
_database_gen.py <zebu.work> -o <output.dir>
```

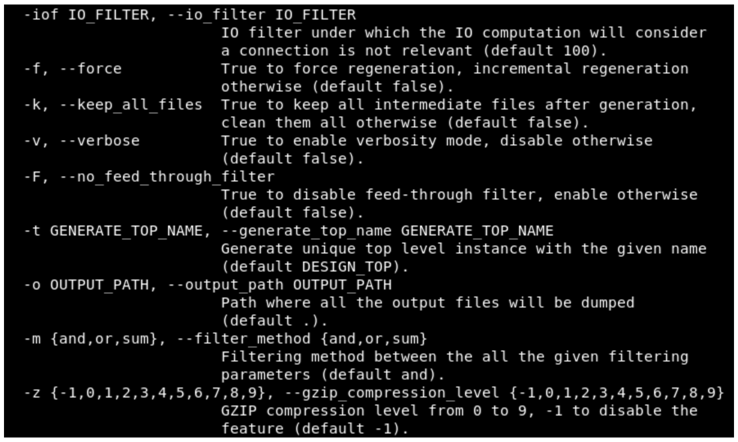
Although it is sufficient to generate a full HTML page, there are several arguments with default values. The following figure shows the help output of the script:

*Figure 40  Partition viewer standalone flow usage*



```
usage: pview_database_gen.py [-h] [--local_db_script LOCAL_DB_SCRIPT]
                             [--merge_db_script MERGE_DB_SCRIPT]
                             [--html_gen_script HTML_GEN_SCRIPT]
                             [--cleanup_script CLEANUP_SCRIPT]
                             [--rundir_script RUNDIR_SCRIPT] [-r REGS_FILTER]
                             [-l LUTS_FILTER] [-u URAMS_FILTER]
                             [-b BRAMS_FILTER] [-ioh IO_HIER_DEPTH_FILTER]
                             [-iof IO_FILTER] [-f] [-k] [-v] [-F]
                             [-t GENERATE_TOP_NAME] [-o OUTPUT_PATH]
                             [-m {and,or,sum}] [-z {-1,0,1,2,3,4,5,6,7,8,9}]
                             zebu_work_dir

Zebu Partition Viewer Database Generator

positional arguments:
  zebu_work_dir         zebu_work directory.

optional arguments:
  -h, --help            show this help message and exit
  --local_db_script LOCAL_DB_SCRIPT
                        Path to the file pview_local_database_gen.tcl (default
                        "${ZEBU_ROOT}/etc/scripts/netlist_tools/tools/partitio
                        n_viewer/flow/pview_local_database_gen.tcl").
  --merge_db_script MERGE_DB_SCRIPT
                        Path to the file pview_database_merge.tcl (default "${
                        ZEBU_ROOT}/etc/scripts/netlist_tools/tools/partition_v
                        iewer/flow/pview_database_merge.tcl").
  --html_gen_script HTML_GEN_SCRIPT
                        Path to the bin file partition_viewer (default "${ZEBU
                        _ROOT}/etc/scripts/netlist_tools/tools/partition_viewe
                        r/app/bin/partition_viewer").
  --cleanup_script CLEANUP_SCRIPT
                        Path to the file pview_clean_up.tcl (default "${ZEBU_R
                        OOT}/etc/scripts/netlist_tools/tools/partition_viewer/
                        flow/pview_clean_up.tcl").
  --rundir_script RUNDIR_SCRIPT
                        Path to the file pview_prepare_rundir.py (default "${Z
                        EBU_ROOT}/etc/scripts/netlist_tools/tools/partition_vi
                        ewer/flow/pview_prepare_rundir.py").
  -r REGS_FILTER, --regs_filter REGS_FILTER
                        REGs filter, only instances that contain at least this
                        argument REGs will be analyzed (default 0).
  -l LUTS_FILTER, --luts_filter LUTS_FILTER
                        LUTs filter, only instances that contain at least this
                        argument LUTs will be analyzed (default 10000).
  -u URAMS_FILTER, --urams_filter URAMS_FILTER
                        URAMs filter, only instances that contain at least
                        this argument URAMs will be analyzed (default 0).
  -b BRAMS_FILTER, --brams_filter BRAMS_FILTER
                        BRAMs filter, only instances that contain at least
                        this argument BRAMs will be analyzed (default 0).
  -ioh IO_HIER_DEPTH_FILTER, --io_hier_depth_filter IO_HIER_DEPTH_FILTER
                        The hierarchy depth assessed for the calculation of
                        OIs (default 2).
```

Feedback



```
-iof IO_FILTER, --io_filter IO_FILTER
                      IO filter under which the IO computation will consider
                      a connection is not relevant (default 100).
-f, --force           True to force regeneration, incremental regeneration
                      otherwise (default false).
-k, --keep_all_files  True to keep all intermediate files after generation,
                      clean them all otherwise (default false).
-v, --verbose         True to enable verbosity mode, disable otherwise
                      (default false).
-F, --no_feed_through_filter
                      True to disable feed-through filter, enable otherwise
                      (default false).
-t GENERATE_TOP_NAME, --generate_top_name GENERATE_TOP_NAME
                      Generate unique top level instance with the given name
                      (default DESIGN_TOP).
-o OUTPUT_PATH, --output_path OUTPUT_PATH
                      Path where all the output files will be dumped
                      (default .).
-m {and,or,sum}, --filter_method {and,or,sum}
                      Filtering method between the all the given filtering
                      parameters (default and).
-z {-1,0,1,2,3,4,5,6,7,8,9}, --gzip_compression_level {-1,0,1,2,3,4,5,6,7,8,9}
                      GZIP compression level from 0 to 9, -1 to disable the
                      feature (default -1).
```

In zCui flow, user can setup the tasks' grid slot assignment. In zCui has UTF commands for that, Partition Viewer flow relies on environment variables.

There are following five environment variables to setup:

- `REMOTECMD_FPGA`
- `REMOTECMD_MERGE`
- `REMOTECMD_HTML`
- `REMOTECMD_CLEANUP`
- `REMOTECMD_RUNDIR`

They all have the same default value: `$REMOTECMD`.

The recommended amount of memory is provided in the following table:

*Table 1    Partition Viewer standalone flow, remote commands recommendations*

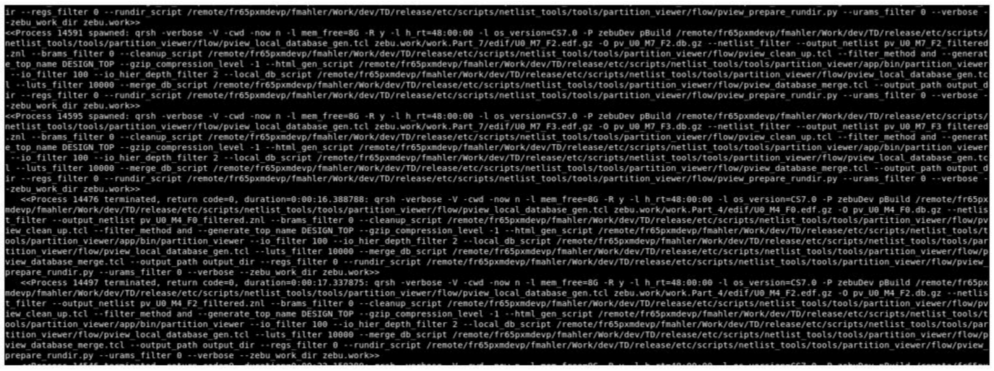
| **Remote Command** | **Low Memory (<=1GB)** | **Medium Memory (]1GB; 8GB])** | **High Memory (>8GB)** |
|---|---|---|---|
| REMOTECMD_FPGA | | X | |
| REMOTECMD_MERGE | | | X |
| REMOTECMD_HTML | X | | |
| REMOTECMD_CLEANUP | X | | |
| REMOTECMD_RUNDIR | X | | |

This flow generates log files. They are located in the output path (option `-o` or `--output_path`).

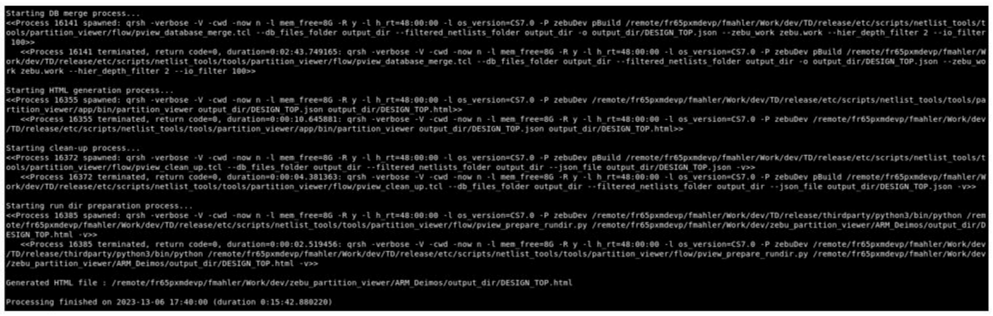
Furthermore, the verbose mode (`-v`) allows the flow to display several useful information, regarding the tasks currently on-going.

The examples of typical verbose output are shown as follows:

*Figure 41 Partition viewer standalone flow outputs*



```
ir --regs_filter 0 --rundir_script /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/flow/pview_prepare_rundir.py --urams_filter 0 --verbose -
-zebu_work_dir zebu.work>>
<<Process 14591 spawned: qrsh -verbose -V -cwd -now n -l mem_free=8G -R y -l h_rt=48:00:00 -l os_version=CS7.0 -P zebuDev pBuild /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/
netlist_tools/tools/partition_viewer/flow/pview_local_database_gen.tcl zebu.work/work.Part_7/edif/U0_M7_F2.edf.gz -O pv_U0_M7_F2.db.gz --netlist_filter --output_netlist pv_U0_M7_F2_filtered
.znl --brams_filter 0 --cleanup_script /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/flow/pview_clean_up.tcl --filter_method and --generat
e_top_name DESIGN_TOP --gzip_compression_level -1 --html_gen_script /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/app/bin/partition_viewer
 --io_filter 100 --io_hier_depth_filter 2 --local_db_script /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/flow/pview_local_database_gen.tc
l --luts_filter 10000 --merge_db_script /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/flow/pview_database_merge.tcl --output_path output_d
ir --regs_filter 0 --rundir_script /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/flow/pview_prepare_rundir.py --urams_filter 0 --verbose -
-zebu_work_dir zebu.work>>
<<Process 14595 spawned: qrsh -verbose -V -cwd -now n -l mem_free=8G -R y -l h_rt=48:00:00 -l os_version=CS7.0 -P zebuDev pBuild /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/
netlist_tools/tools/partition_viewer/flow/pview_local_database_gen.tcl zebu.work/work.Part_7/edif/U0_M7_F3.edf.gz -O pv_U0_M7_F3.db.gz --netlist_filter --output_netlist pv_U0_M7_F3_filtered
.znl --brams_filter 0 --cleanup_script /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/flow/pview_clean_up.tcl --filter_method and --generat
e_top_name DESIGN_TOP --gzip_compression_level -1 --html_gen_script /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/app/bin/partition_viewer
 --io_filter 100 --io_hier_depth_filter 2 --local_db_script /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/flow/pview_local_database_gen.tc
l --luts_filter 10000 --merge_db_script /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/flow/pview_database_merge.tcl --output_path output_d
ir --regs_filter 0 --rundir_script /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/flow/pview_prepare_rundir.py --urams_filter 0 --verbose -
-zebu_work_dir zebu.work>>
   <<Process 14476 terminated, return code=0, duration=0:00:16.388788: qrsh -verbose -V -cwd -now n -l mem_free=8G -R y -l h_rt=48:00:00 -l os_version=CS7.0 -P zebuDev pBuild /remote/fr65px
mdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/flow/pview_local_database_gen.tcl zebu.work/work.Part_4/edif/U0_M4_F0.edf.gz -O pv_U0_M4_F0.db.gz --netlis
t_filter --output_netlist pv_U0_M4_F0_filtered.znl --brams_filter 0 --cleanup_script /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/flow/pv
iew_clean_up.tcl --filter_method and --generate_top_name DESIGN_TOP --gzip_compression_level -1 --html_gen_script /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/t
ools/partition_viewer/app/bin/partition_viewer --io_filter 100 --io_hier_depth_filter 2 --local_db_script /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/par
tition_viewer/flow/pview_local_database_gen.tcl --luts_filter 10000 --merge_db_script /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/flow/p
view_database_merge.tcl --output_path output_dir --regs_filter 0 --rundir_script /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/flow/pview_
prepare_rundir.py --urams_filter 0 --verbose --zebu_work_dir zebu.work>>
   <<Process 14497 terminated, return code=0, duration=0:00:17.337875: qrsh -verbose -V -cwd -now n -l mem_free=8G -R y -l h_rt=48:00:00 -l os_version=CS7.0 -P zebuDev pBuild /remote/fr65px
mdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/flow/pview_local_database_gen.tcl zebu.work/work.Part_4/edif/U0_M4_F2.edf.gz -O pv_U0_M4_F2.db.gz --netlis
t_filter --output_netlist pv_U0_M4_F2_filtered.znl --brams_filter 0 --cleanup_script /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/flow/pv
iew_clean_up.tcl --filter_method and --generate_top_name DESIGN_TOP --gzip_compression_level -1 --html_gen_script /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/t
ools/partition_viewer/app/bin/partition_viewer --io_filter 100 --io_hier_depth_filter 2 --local_db_script /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/par
tition_viewer/flow/pview_local_database_gen.tcl --luts_filter 10000 --merge_db_script /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/flow/p
view_database_merge.tcl --output_path output_dir --regs_filter 0 --rundir_script /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/flow/pview_
prepare_rundir.py --urams_filter 0 --verbose --zebu_work_dir zebu.work>>
```



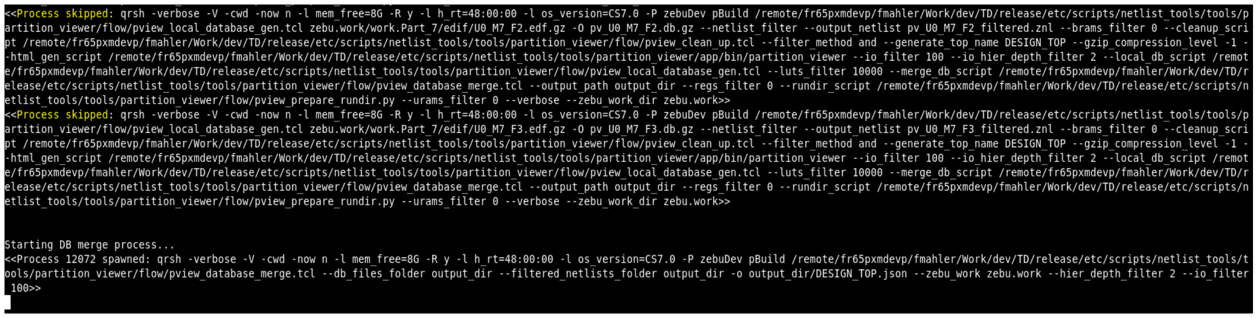
```
Starting DB merge process...
<<Process 16141 spawned: qrsh -verbose -V -cwd -now n -l mem_free=8G -R y -l h_rt=48:00:00 -l os_version=CS7.0 -P zebuDev pBuild /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/t
ools/partition_viewer/flow/pview_database_merge.tcl --db_files_folder output_dir --filtered_netlists_folder output_dir -o output_dir/DESIGN_TOP.json --zebu_work zebu.work --hier_depth_filter 2 --io_filter
 100>>
   <<Process 16141 terminated, return code=0, duration=0:02:43.749165: qrsh -verbose -V -cwd -now n -l mem_free=8G -R y -l h_rt=48:00:00 -l os_version=CS7.0 -P zebuDev pBuild /remote/fr65pxmdevp/fmahler/W
ork/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/flow/pview_database_merge.tcl --db_files_folder output_dir --filtered_netlists_folder output_dir -o output_dir/DESIGN_TOP.json --zebu_wo
rk zebu.work --hier_depth_filter 2 --io_filter 100>>

Starting HTML generation process...
<<Process 16355 spawned: qrsh -verbose -V -cwd -now n -l mem_free=8G -R y -l h_rt=48:00:00 -l os_version=CS7.0 -P zebuDev /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/pa
rtition_viewer/app/bin/partition_viewer output_dir/DESIGN_TOP.json output_dir/DESIGN_TOP.html>>
   <<Process 16355 terminated, return code=0, duration=0:00:10.645881: qrsh -verbose -V -cwd -now n -l mem_free=8G -R y -l h_rt=48:00:00 -l os_version=CS7.0 -P zebuDev /remote/fr65pxmdevp/fmahler/Work/dev
/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/app/bin/partition_viewer output_dir/DESIGN_TOP.json output_dir/DESIGN_TOP.html>>

Starting clean-up process...
<<Process 16372 spawned: qrsh -verbose -V -cwd -now n -l mem_free=8G -R y -l h_rt=48:00:00 -l os_version=CS7.0 -P zebuDev pBuild /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/t
ools/partition_viewer/flow/pview_clean_up.tcl --db_files_folder output_dir --filtered_netlists_folder output_dir --json_file output_dir/DESIGN_TOP.json -v>>
   <<Process 16372 terminated, return code=0, duration=0:00:04.381363: qrsh -verbose -V -cwd -now n -l mem_free=8G -R y -l h_rt=48:00:00 -l os_version=CS7.0 -P zebuDev pBuild /remote/fr65pxmdevp/fmahler/W
ork/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/flow/pview_clean_up.tcl --db_files_folder output_dir --filtered_netlists_folder output_dir --json_file output_dir/DESIGN_TOP.json -v>>

Starting run dir preparation process...
<<Process 16385 spawned: qrsh -verbose -V -cwd -now n -l mem_free=8G -R y -l h_rt=48:00:00 -l os_version=CS7.0 -P zebuDev /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/thirdparty/python3/bin/python /rem
ote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/flow/pview_prepare_rundir.py /remote/fr65pxmdevp/fmahler/Work/dev/zebu_partition_viewer/ARM_Deimos/output_dir/D
ESIGN_TOP.html -v>>
   <<Process 16385 terminated, return code=0, duration=0:00:02.519456: qrsh -verbose -V -cwd -now n -l mem_free=8G -R y -l h_rt=48:00:00 -l os_version=CS7.0 -P zebuDev /remote/fr65pxmdevp/fmahler/Work/dev
/TD/release/thirdparty/python3/bin/python /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/flow/pview_prepare_rundir.py /remote/fr65pxmdevp/fmahler/Work/dev
/zebu_partition_viewer/ARM_Deimos/output_dir/DESIGN_TOP.html -v>>

Generated HTML file : /remote/fr65pxmdevp/fmahler/Work/dev/zebu_partition_viewer/ARM_Deimos/output_dir/DESIGN_TOP.html

Processing finished on 2023-13-06 17:40:00 (duration 0:15:42.880220)
```

Finally, the standalone generation flow provides its own incrementality feature. When one of its tasks fails, the flow stops, indicating what task has failed, which log file to read and how to reproduce.

Then, if nothing changed, if the user refire the same command, all the successfully terminated tasks are skipped and the flow resume from the faulty one.

When all the tasks have successfully finished, the intermediate temporary files are removed and the incrementality is no longer active. It is possible to force the flow to keep the intermediate temporary files to force the incrementality. The option for that is `-k` or `--keep_all_files`.

*Figure 42 Partition viewer standalone flow incremental output*



```
<<Process skipped: qrsh -verbose -V -cwd -now n -l mem_free=8G -R y -l h_rt=48:00:00 -l os_version=CS7.0 -P zebuDev pBuild /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/p
artition_viewer/flow/pview_local_database_gen.tcl zebu.work/work.Part_7/edif/U0_M7_F2.edf.gz -O pv_U0_M7_F2.db.gz --netlist_filter --output_netlist pv_U0_M7_F2_filtered.znl --brams_filter 0 --cleanup_scri
pt /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/flow/pview_clean_up.tcl --filter_method and --generate_top_name DESIGN_TOP --gzip_compression_level -1 -
-html_gen_script /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/app/bin/partition_viewer --io_filter 100 --io_hier_depth_filter 2 --local_db_script /remot
e/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/flow/pview_local_database_gen.tcl --luts_filter 10000 --merge_db_script /remote/fr65pxmdevp/fmahler/Work/dev/TD/r
elease/etc/scripts/netlist_tools/tools/partition_viewer/flow/pview_database_merge.tcl --output_path output_dir --regs_filter 0 --rundir_script /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/n
etlist_tools/tools/partition_viewer/flow/pview_prepare_rundir.py --urams_filter 0 --verbose --zebu_work_dir zebu.work>>
<<Process skipped: qrsh -verbose -V -cwd -now n -l mem_free=8G -R y -l h_rt=48:00:00 -l os_version=CS7.0 -P zebuDev pBuild /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/p
artition_viewer/flow/pview_local_database_gen.tcl zebu.work/work.Part_7/edif/U0_M7_F3.edf.gz -O pv_U0_M7_F3.db.gz --netlist_filter --output_netlist pv_U0_M7_F3_filtered.znl --brams_filter 0 --cleanup_scri
pt /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/flow/pview_clean_up.tcl --filter_method and --generate_top_name DESIGN_TOP --gzip_compression_level -1 -
-html_gen_script /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/app/bin/partition_viewer --io_filter 100 --io_hier_depth_filter 2 --local_db_script /remot
e/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/flow/pview_local_database_gen.tcl --luts_filter 10000 --merge_db_script /remote/fr65pxmdevp/fmahler/Work/dev/TD/r
elease/etc/scripts/netlist_tools/tools/partition_viewer/flow/pview_database_merge.tcl --output_path output_dir --regs_filter 0 --rundir_script /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/n
etlist_tools/tools/partition_viewer/flow/pview_prepare_rundir.py --urams_filter 0 --verbose --zebu_work_dir zebu.work>>


Starting DB merge process...
<<Process 12072 spawned: qrsh -verbose -V -cwd -now n -l mem_free=8G -R y -l h_rt=48:00:00 -l os_version=CS7.0 -P zebuDev pBuild /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/t
ools/partition_viewer/flow/pview_database_merge.tcl --db_files_folder output_dir --filtered_netlists_folder output_dir -o output_dir/DESIGN_TOP.json --zebu_work zebu.work --hier_depth_filter 2 --io_filter
 100>>
```

The output of the standalone flow is located in the setup output_path folder (option `-o` or `--output_path`).

For zCui flow, to open the **Partition Viewer** page, either run:

```
firefox <output_path>/run/tool.html
```

Or, if the environment has access to `nodeJS`:

```
cd <output_path>/run; ./run.sh
```

# 4

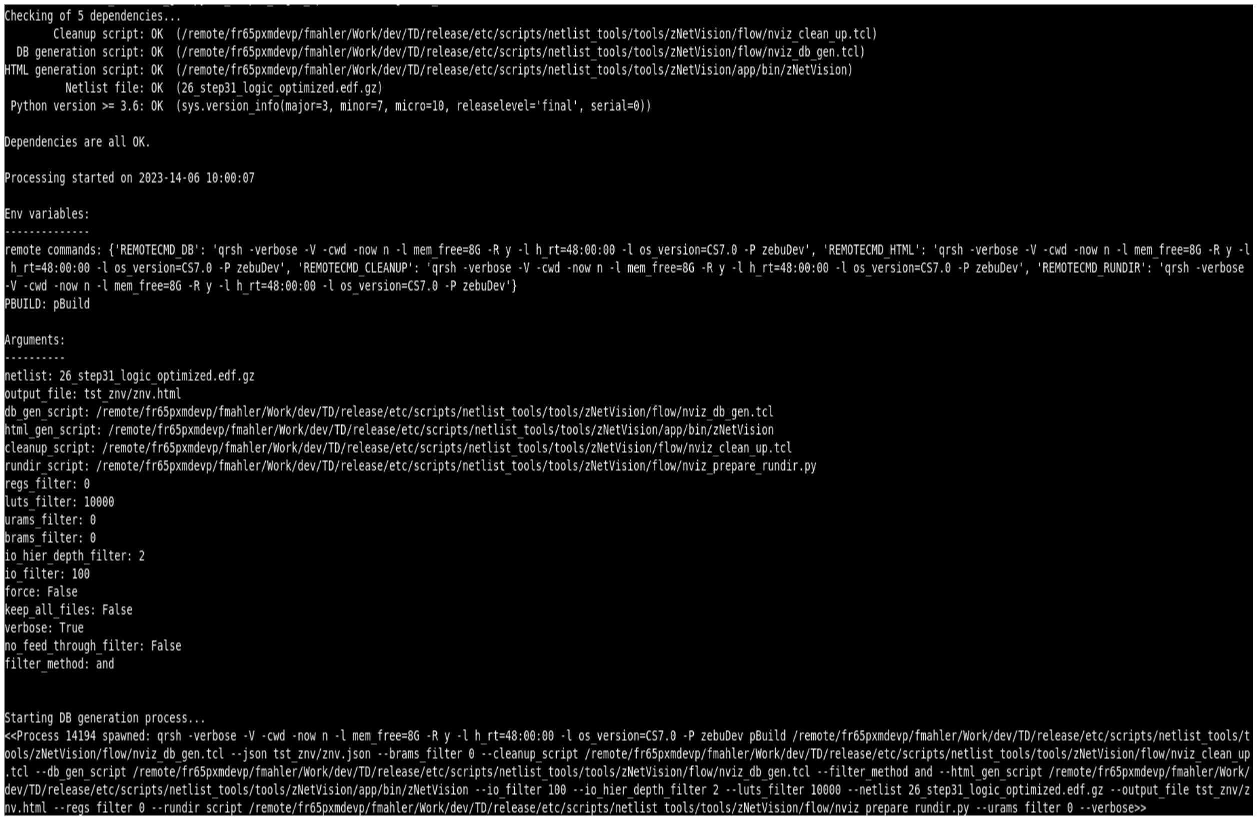
# zNetVision

Leveraging from the Partition Viewer code, zNetVision is a tool that allows the users to visualize netlists using the same GUI components as Partition Viewer.

It is a Partition Viewer light that only displays hierarchy component, resources selection, search feature and IO graph. Partitioning information is excluded.

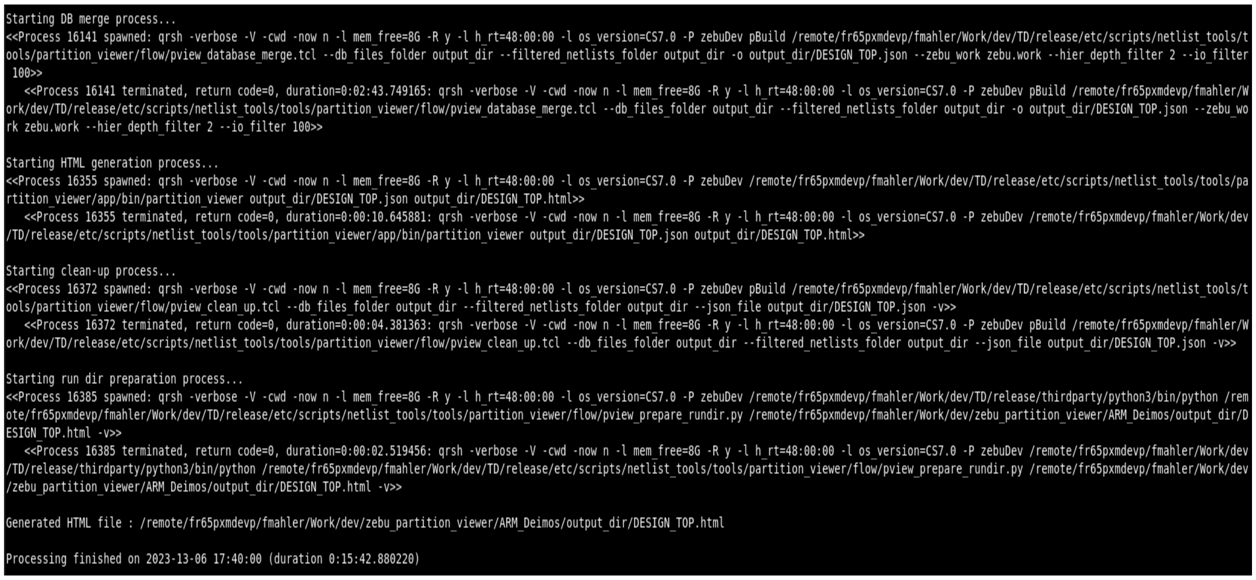
It takes a ZNL or EDIF netlist as its only input and produces an HTML page.

*Figure 43　zNetVision standalone flow outputs*



```
Checking of 5 dependencies...
        Cleanup script: OK  (/remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/zNetVision/flow/nviz_clean_up.tcl)
  DB generation script: OK  (/remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/zNetVision/flow/nviz_db_gen.tcl)
HTML generation script: OK  (/remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/zNetVision/app/bin/zNetVision)
          Netlist file: OK  (26_step31_logic_optimized.edf.gz)
 Python version >= 3.6: OK  (sys.version_info(major=3, minor=7, micro=10, releaselevel='final', serial=0))

Dependencies are all OK.

Processing started on 2023-14-06 10:00:07

Env variables:
--------------
remote commands: {'REMOTECMD_DB': 'qrsh -verbose -V -cwd -now n -l mem_free=8G -R y -l h_rt=48:00:00 -l os_version=CS7.0 -P zebuDev', 'REMOTECMD_HTML': 'qrsh -verbose -V -cwd -now n -l mem_free=8G -R y -l
 h_rt=48:00:00 -l os_version=CS7.0 -P zebuDev', 'REMOTECMD_CLEANUP': 'qrsh -verbose -V -cwd -now n -l mem_free=8G -R y -l h_rt=48:00:00 -l os_version=CS7.0 -P zebuDev', 'REMOTECMD_RUNDIR': 'qrsh -verbose
-V -cwd -now n -l mem_free=8G -R y -l h_rt=48:00:00 -l os_version=CS7.0 -P zebuDev'}
PBUILD: pBuild

Arguments:
----------
netlist: 26_step31_logic_optimized.edf.gz
output_file: tst_znv/znv.html
db_gen_script: /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/zNetVision/flow/nviz_db_gen.tcl
html_gen_script: /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/zNetVision/app/bin/zNetVision
cleanup_script: /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/zNetVision/flow/nviz_clean_up.tcl
rundir_script: /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/zNetVision/flow/nviz_prepare_rundir.py
regs_filter: 0
luts_filter: 10000
urams_filter: 0
brams_filter: 0
io_hier_depth_filter: 2
io_filter: 100
force: False
keep_all_files: False
verbose: True
no_feed_through_filter: False
filter_method: and


Starting DB generation process...
<<Process 14194 spawned: qrsh -verbose -V -cwd -now n -l mem_free=8G -R y -l h_rt=48:00:00 -l os_version=CS7.0 -P zebuDev pBuild /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/t
ools/zNetVision/flow/nviz_db_gen.tcl --json tst_znv/znv.json --brams_filter 0 --cleanup_script /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/zNetVision/flow/nviz_clean_up
.tcl --db_gen_script /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/zNetVision/flow/nviz_db_gen.tcl --filter_method and --html_gen_script /remote/fr65pxmdevp/fmahler/Work/
dev/TD/release/etc/scripts/netlist_tools/tools/zNetVision/app/bin/zNetVision --io_filter 100 --io_hier_depth_filter 2 --luts_filter 10000 --netlist 26_step31_logic_optimized.edf.gz --output_file tst_znv/z
nv.html --regs_filter 0 --rundir_script /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/zNetVision/flow/nviz_prepare_rundir.py --urams_filter 0 --verbose>>
```

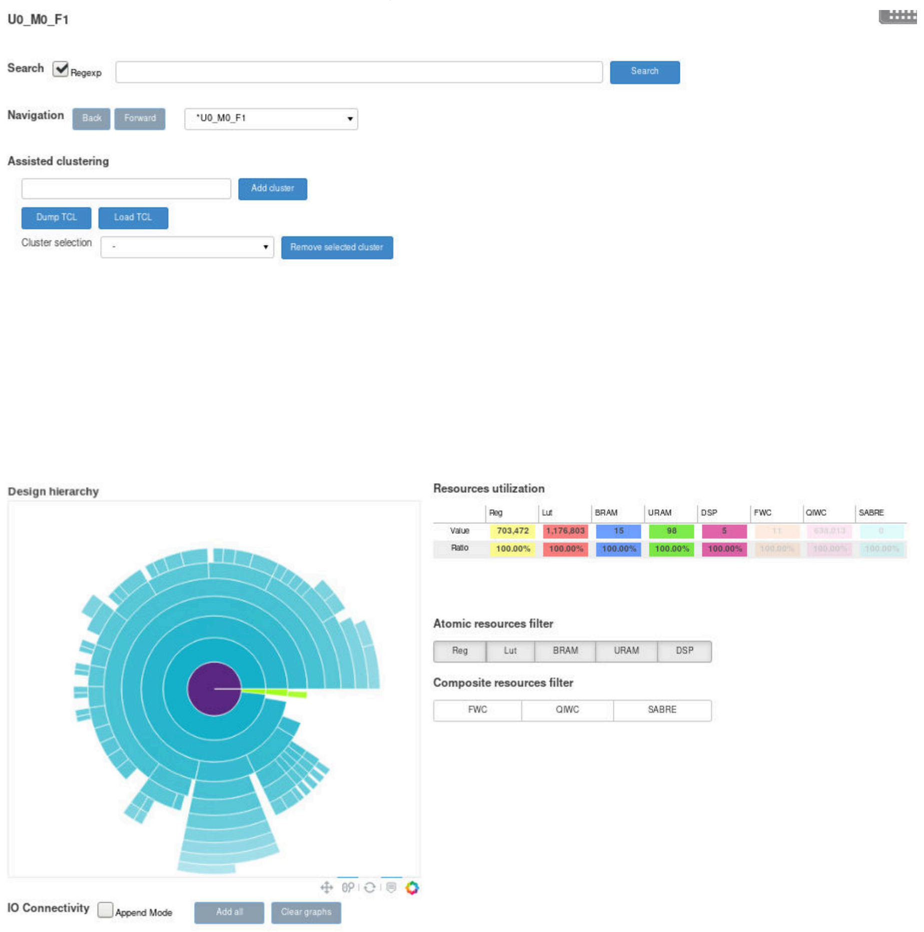
Feedback



```
Starting DB merge process...
<<Process 16141 spawned: qrsh -verbose -V -cwd -now n -l mem_free=8G -R y -l h_rt=48:00:00 -l os_version=CS7.0 -P zebuDev pBuild /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/t
ools/partition_viewer/flow/pview_database_merge.tcl --db_files_folder output_dir --filtered_netlists_folder output_dir -o output_dir/DESIGN_TOP.json --zebu_work zebu.work --hier_depth_filter 2 --io_filter
 100>>
   <<Process 16141 terminated, return code=0, duration=0:02:43.749165: qrsh -verbose -V -cwd -now n -l mem_free=8G -R y -l h_rt=48:00:00 -l os_version=CS7.0 -P zebuDev pBuild /remote/fr65pxmdevp/fmahler/W
ork/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/flow/pview_database_merge.tcl --db_files_folder output_dir --filtered_netlists_folder output_dir -o output_dir/DESIGN_TOP.json --zebu_wo
rk zebu.work --hier_depth_filter 2 --io_filter 100>>

Starting HTML generation process...
<<Process 16355 spawned: qrsh -verbose -V -cwd -now n -l mem_free=8G -R y -l h_rt=48:00:00 -l os_version=CS7.0 -P zebuDev /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/pa
rtition_viewer/app/bin/partition_viewer output_dir/DESIGN_TOP.json output_dir/DESIGN_TOP.html>>
   <<Process 16355 terminated, return code=0, duration=0:00:10.645881: qrsh -verbose -V -cwd -now n -l mem_free=8G -R y -l h_rt=48:00:00 -l os_version=CS7.0 -P zebuDev /remote/fr65pxmdevp/fmahler/Work/dev
/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/app/bin/partition_viewer output_dir/DESIGN_TOP.json output_dir/DESIGN_TOP.html>>

Starting clean-up process...
<<Process 16372 spawned: qrsh -verbose -V -cwd -now n -l mem_free=8G -R y -l h_rt=48:00:00 -l os_version=CS7.0 -P zebuDev pBuild /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/t
ools/partition_viewer/flow/pview_clean_up.tcl --db_files_folder output_dir --filtered_netlists_folder output_dir --json_file output_dir/DESIGN_TOP.json -v>>
   <<Process 16372 terminated, return code=0, duration=0:00:04.381363: qrsh -verbose -V -cwd -now n -l mem_free=8G -R y -l h_rt=48:00:00 -l os_version=CS7.0 -P zebuDev pBuild /remote/fr65pxmdevp/fmahler/W
ork/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/flow/pview_clean_up.tcl --db_files_folder output_dir --filtered_netlists_folder output_dir --json_file output_dir/DESIGN_TOP.json -v>>

Starting run dir preparation process...
<<Process 16385 spawned: qrsh -verbose -V -cwd -now n -l mem_free=8G -R y -l h_rt=48:00:00 -l os_version=CS7.0 -P zebuDev /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/thirdparty/python3/bin/python /rem
ote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/flow/pview_prepare_rundir.py /remote/fr65pxmdevp/fmahler/Work/dev/zebu_partition_viewer/ARM_Deimos/output_dir/D
ESIGN_TOP.html -v>>
   <<Process 16385 terminated, return code=0, duration=0:00:02.519456: qrsh -verbose -V -cwd -now n -l mem_free=8G -R y -l h_rt=48:00:00 -l os_version=CS7.0 -P zebuDev /remote/fr65pxmdevp/fmahler/Work/dev
/TD/release/thirdparty/python3/bin/python /remote/fr65pxmdevp/fmahler/Work/dev/TD/release/etc/scripts/netlist_tools/tools/partition_viewer/flow/pview_prepare_rundir.py /remote/fr65pxmdevp/fmahler/Work/dev
/zebu_partition_viewer/ARM_Deimos/output_dir/DESIGN_TOP.html -v>>

Generated HTML file : /remote/fr65pxmdevp/fmahler/Work/dev/zebu_partition_viewer/ARM_Deimos/output_dir/DESIGN_TOP.html

Processing finished on 2023-13-06 17:40:00 (duration 0:15:42.880220)
```

The following figure depicts the generated page:

*Figure 44 zNetVision HTML page overview*



Resources utilization

| | Reg | Lut | BRAM | URAM | DSP | FWC | QIWC | SABRE |
|---|---|---|---|---|---|---|---|---|
| Value | 703,472 | 1,176,803 | 15 | 98 | 5 | 11 | 638,013 | 0 |
| Ratio | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% |

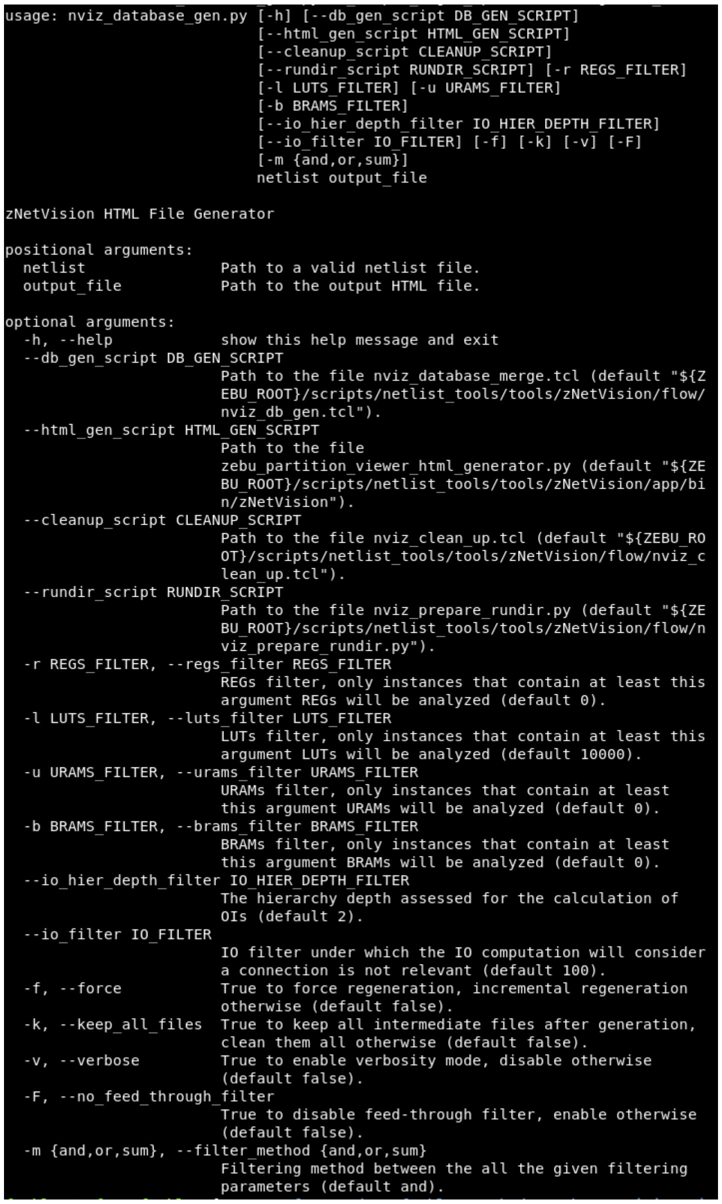
Although the features are the same as the Partition Viewer, not all features are present. The only way to generate the tool is by using a standalone flow, very similar to Partition Viewer.

Here is a typical usage of the command:

```
?> source <zebu.release>
?> zPython --pythonpath ${ZEBU_ROOT}/etc/scripts/netlist_tools
 ${ZEBU_ROOT}/etc/scripts/netlist_tools/tools/zNetVision/flow/nviz_databa
se_gen.py <netlist_path> <output_html_path>
```

Although it is sufficient to generate a full HTML page, there are several arguments with default values. Here is the help output of the script:

*Figure 45 zNetVision command line usage*



```
usage: nviz_database_gen.py [-h] [--db_gen_script DB_GEN_SCRIPT]
                            [--html_gen_script HTML_GEN_SCRIPT]
                            [--cleanup_script CLEANUP_SCRIPT]
                            [--rundir_script RUNDIR_SCRIPT] [-r REGS_FILTER]
                            [-l LUTS_FILTER] [-u URAMS_FILTER]
                            [-b BRAMS_FILTER]
                            [--io_hier_depth_filter IO_HIER_DEPTH_FILTER]
                            [--io_filter IO_FILTER] [-f] [-k] [-v] [-F]
                            [-m {and,or,sum}]
                            netlist output_file

zNetVision HTML File Generator

positional arguments:
  netlist               Path to a valid netlist file.
  output_file           Path to the output HTML file.

optional arguments:
  -h, --help            show this help message and exit
  --db_gen_script DB_GEN_SCRIPT
                        Path to the file nviz_database_merge.tcl (default "${Z
                        EBU_ROOT}/scripts/netlist_tools/tools/zNetVision/flow/
                        nviz_db_gen.tcl").
  --html_gen_script HTML_GEN_SCRIPT
                        Path to the file
                        zebu_partition_viewer_html_generator.py (default "${ZE
                        BU_ROOT}/scripts/netlist_tools/tools/zNetVision/app/bi
                        n/zNetVision").
  --cleanup_script CLEANUP_SCRIPT
                        Path to the file nviz_clean_up.tcl (default "${ZEBU_RO
                        OT}/scripts/netlist_tools/tools/zNetVision/flow/nviz_c
                        lean_up.tcl").
  --rundir_script RUNDIR_SCRIPT
                        Path to the file nviz_prepare_rundir.py (default "${ZE
                        BU_ROOT}/scripts/netlist_tools/tools/zNetVision/flow/n
                        viz_prepare_rundir.py").
  -r REGS_FILTER, --regs_filter REGS_FILTER
                        REGs filter, only instances that contain at least this
                        argument REGs will be analyzed (default 0).
  -l LUTS_FILTER, --luts_filter LUTS_FILTER
                        LUTs filter, only instances that contain at least this
                        argument LUTs will be analyzed (default 10000).
  -u URAMS_FILTER, --urams_filter URAMS_FILTER
                        URAMs filter, only instances that contain at least
                        this argument URAMs will be analyzed (default 0).
  -b BRAMS_FILTER, --brams_filter BRAMS_FILTER
                        BRAMs filter, only instances that contain at least
                        this argument BRAMs will be analyzed (default 0).
  --io_hier_depth_filter IO_HIER_DEPTH_FILTER
                        The hierarchy depth assessed for the calculation of
                        OIs (default 2).
  --io_filter IO_FILTER
                        IO filter under which the IO computation will consider
                        a connection is not relevant (default 100).
  -f, --force           True to force regeneration, incremental regeneration
                        otherwise (default false).
  -k, --keep_all_files  True to keep all intermediate files after generation,
                        clean them all otherwise (default false).
  -v, --verbose         True to enable verbosity mode, disable otherwise
                        (default false).
  -F, --no_feed_through_filter
                        True to disable feed-through filter, enable otherwise
                        (default false).
  -m {and,or,sum}, --filter_method {and,or,sum}
                        Filtering method between the all the given filtering
                        parameters (default and).
```

This tool is useful for getting an overview of a design without worrying about whether this or that file is actually present in the design compilation directory.

Feedback

5

# Limitations of Partition Viewer

The Partition Viewer has some limitations of varying severity.

- Some compilation folders do not contain all the required files (or the files do not contain all the required information) in which Partition Viewer relies on. Although the generation flow goes through, this can cause the HTML to be incomplete, and even empty in some extreme cases.

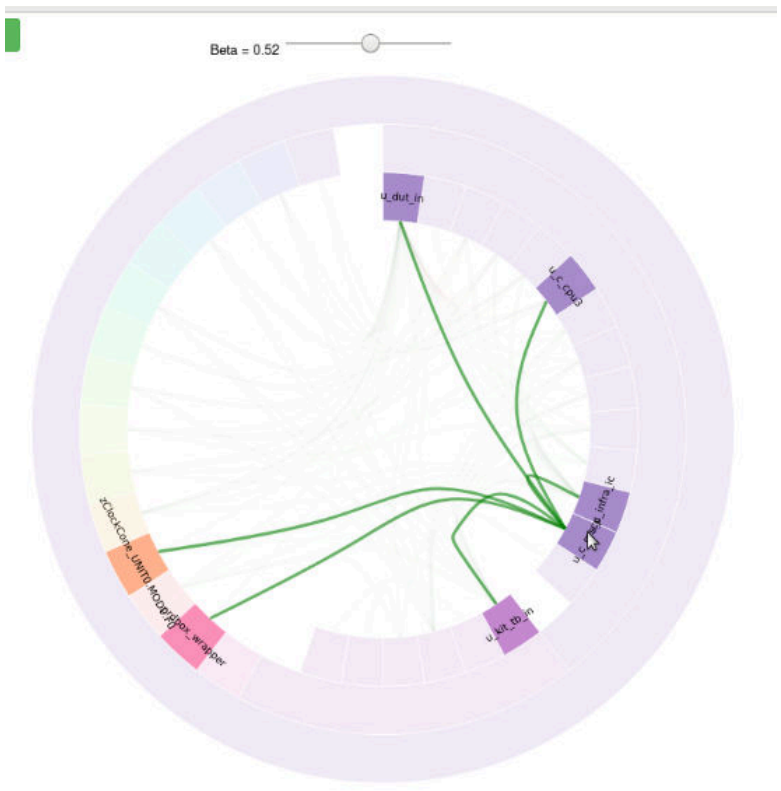
A new mode in the generation flow, called the Segmented Flow is planned. This will allow the user to generate partially the Partition Viewer page using available information, so that the user need not wait for compilation to be completed.

- Partition Viewer flow relies on files to gather all required information. When the format of those files changes, it can prevent files parsing.
- Enhancement of the zAudit tool is in progress, so that it can provide all the information required by the Partition Viewer.

Feedback

- For large designs, the IO Graph can be totally illegible and incomprehensible.
- Work on an advanced IO graph visualization is in progress. This will improve the current IO Graph implemented in the Partition Viewer. It will look like something like that:

*Figure 46 Advanced IO graph visualization*



- The number of instances of the design can impact the duration of the generations and the responsiveness of the interface.
- One of the major causes of slowdown of the generation is that the flow is written using TCL pBuild scripts. It can be highly improved by rewriting them in C++.
- The responsiveness can be improved by changing GUI framework, Bokeh, and by filtering the number of instances.
- The GUI cannot adapt to the size of the window, and users often must zoom out to see all the components. This Bokeh limitation can be improved by changing the framework. There is no regression test cases at any level of validation.

- Assisted clustering tool is on its preliminary version, and it needs some enhancements:
    - The dumped zTopBuild clustering commands should use more generic instance paths.
    - The name of the top currently used is arbitrary and may not be found by zTopBuild.

# 6

# Appendix IO Graph

The IO Graph component is available at the bottom of Partition Viewer page. It represents connectivity between the partitioned instances.

There are two IO graphs: the graph on the left takes into account FPGAs involved in chosen instances partitioning, and the graph on the right the FPGAs that are not involved in chosen instances partitioning.

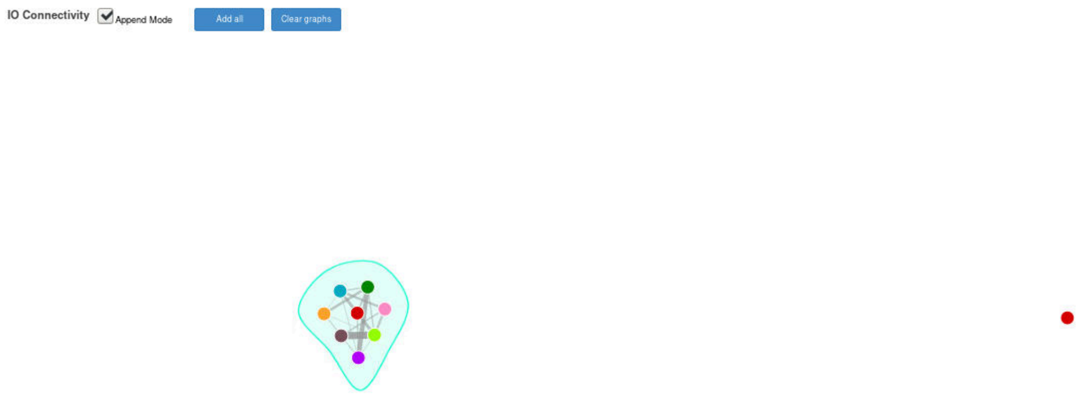
Initially, the graph appears as a check box and some buttons. The sunburst (component 6) behavior changes when the check box **Append Mode** is checked, to allow the user to choose instances from it.

When **Append Mode** is checked, the sunburst highlights only the instances that can be appended in the IO Graph. Also, **Add all** and **Clear graphs** buttons are enabled.

When an instance is chosen from the hierarchy sunburst, it is added to the set of instances shown in the IO graphs.

The following figure depicts the graph when one instance is selected:

*Figure 47 IO Graph one instance selected*

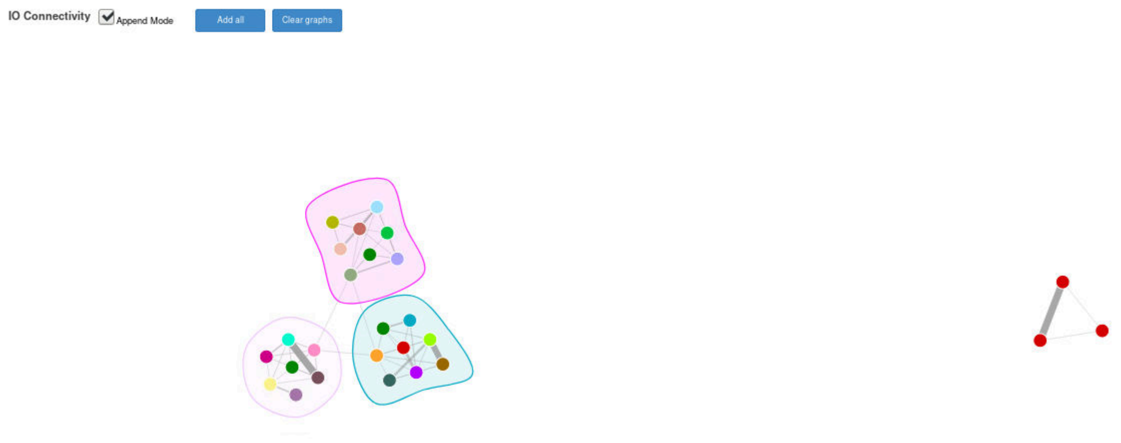


On the left graph, observe several colored nodes, eventually linked with lines of varying thickness. The nodes are the pieces of the partitioned instance onto the FPGAs. The selected instance involves 8 FPGAs in its partitioning. And these pieces can be connected to others with varying IO cut, which is represented by the thickness of the lines. The blue cluster is the instance itself.

On the right, just one node, the selected instance.

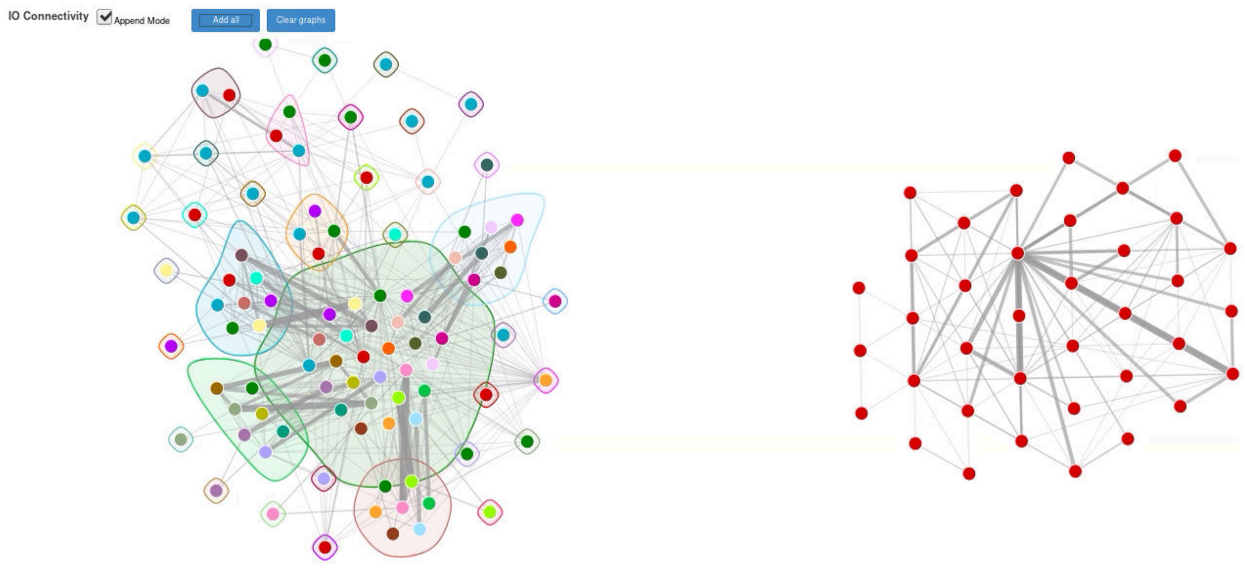
After selecting two more instances, observe that the graph appears as depicted in this figure:

*Figure 48        IO graph thee instances selected*



After clicking **Add all**, the graph appears as depicted in the following image:

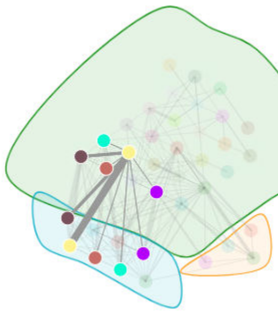
*Figure 49        IO graph all instances selected*



All the “addable” instances (highlighted in the sunburst) are added automatically to the graphs.

Another interaction is the hovering of the instances in the IO graphs. All links connected to the hovered instance, and the instances connected to those links, are highlighted.

*Figure 50 IO graph highlighted links*

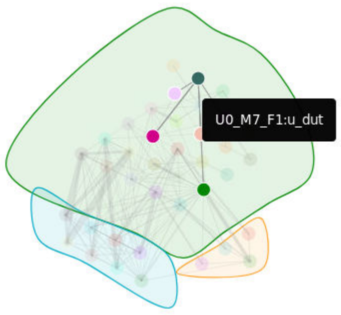


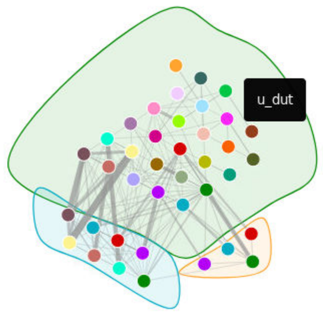
Clicking **Clear graphs** removes instances from the graph.

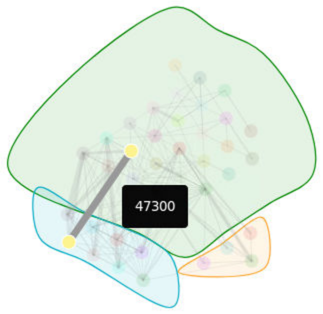
When the **Append mode** check box is unchecked, IO graphs remains but it's no longer possible to add instances to them.

Finally, there are tooltips on the nodes to display their FPGA & paths, on the large cluster to display the path of the instance, and on the links to display IO cut value.

*Figure 51 IO graph, tool tips*




u_dut


47300