QA QoR Implementation

**Software Quality Team**

ABSTRACT

xxxxxxxx

# CONTENTS

[CONTENTS 2](#_Toc154060639)

[1. Overall introduction 4](#_Toc154060640)

[1.1 QoR implementation and methods 4](#_Toc154060641)

[1.2 QoR run location 4](#_Toc154060642)

[1.3 QoR report location 4](#_Toc154060643)

[2. Generic QoR 4](#_Toc154060644)

[2.1 Applicable areas 4](#_Toc154060645)

[2.2 Benchmark resource 4](#_Toc154060646)

[2.3 Default run and report 5](#_Toc154060647)

[2.3.1 Run command 5](#_Toc154060648)

[2.3.2 Report generation 5](#_Toc154060649)

[2.4 QoR implementation 5](#_Toc154060650)

[2.4.1 Generic QoR with pushbutton method 5](#_Toc154060651)

[2.4.2 Generic QoR with sweeping method 7](#_Toc154060652)

[2.4.3 Generic QoR with Iteration method 9](#_Toc154060653)

[2.4.4 Generic QoR with seed method 11](#_Toc154060654)

[3. Simulation QoR 13](#_Toc154060655)

[3.1 Applicable areas 13](#_Toc154060656)

[3.2 Benchmark resource 13](#_Toc154060657)

[3.3 Default run and report 13](#_Toc154060658)

[3.3.1 Run command 13](#_Toc154060659)

[3.3.2 Report generation 14](#_Toc154060660)

[3.4 QoR implementation 14](#_Toc154060661)

[3.4.1 Run method 14](#_Toc154060662)

[3.4.2 More run options 14](#_Toc154060663)

[3.4.3 More report options 15](#_Toc154060664)

[3.4.4 Demo run 15](#_Toc154060665)

[4. Power QoR 15](#_Toc154060666)

[4.1 Applicable areas 15](#_Toc154060667)

[4.2 Benchmark resource 16](#_Toc154060668)

[4.3 Default run and report 16](#_Toc154060669)

[4.3.1 Run command 16](#_Toc154060670)

[4.3.2 Report generation 16](#_Toc154060671)

[4.4 QoR implementation 16](#_Toc154060672)

[4.4.1 Run method 16](#_Toc154060673)

[4.4.2 More run options 17](#_Toc154060674)

[4.4.3 More report options 18](#_Toc154060675)

[4.4.4 Demo run 18](#_Toc154060676)

[5. Repeatability QoR 18](#_Toc154060677)

[5.1 Applicable areas 18](#_Toc154060678)

[5.2 Benchmark resource 18](#_Toc154060679)

[5.3 Default run and report 18](#_Toc154060680)

[5.3.1 Run command 18](#_Toc154060681)

[5.3.2 Report generation 19](#_Toc154060682)

[5.4 QoR implementation 19](#_Toc154060683)

[5.4.1 Run method 19](#_Toc154060684)

[5.4.2 More run options 20](#_Toc154060685)

[5.4.3 More report options 21](#_Toc154060686)

[5.4.4 Demo run 21](#_Toc154060687)

[6. Report generation details 21](#_Toc154060688)

[6.1 Applicable areas 21](#_Toc154060689)

[6.2 Script export 21](#_Toc154060690)

[6.3 Build environment 22](#_Toc154060691)

[6.3.1 Python and package installation 22](#_Toc154060692)

[6.3.2 local package installation 22](#_Toc154060693)

[6.4 Report generation Script methods 22](#_Toc154060694)

[6.4.1 Report for two data sources 22](#_Toc154060695)

[6.4.2 Report for same data source two sections 23](#_Toc154060696)

[6.5 Report generation simple commands 23](#_Toc154060697)

[6.5.1 Two data sources compare report 23](#_Toc154060698)

[6.5.2 One data source two sections compare report 24](#_Toc154060699)

[6.6 Report generation detail options 24](#_Toc154060700)

# Overall introduction

## QoR implementation and methods

Currently our QA can do following QoR benchmarks:

Generic QoR – General performance check for two comparable Software versions or devices

Simulation QoR -- General simulation performance check for comparable simulator or modules

Power QoR – General power consumption check for comparable devices

Repeatability QoR – Engine stability check

All these QoR run are supported by QA TMP platform with specific QoR methods, here are methods platform supported for QoR implementation:

Generic QoR – qor\_pushbutton, qor\_sweep, qor\_seed, qor\_iter

Simulatioin QoR – qor\_sim

Power QoR – qor\_power

Repeatability – qor\_step

For more information, please refer to detail run methods.

## QoR run location

All QoR implementation are based on TMP platform, with the platform help we can run QoR designs with multiple machines and threads.

It is recommended to run all QoR related test in TMP QoR project: [QoR\_Benchmark](http://lsh-tmp/testrail/index.php?/projects/overview/10)

And we have following two suites uploaded already for use:

[QoR\_Radiant\_standard](http://lsh-tmp/testrail/index.php?/suites/view/4139) – for Generic, power, repeatability QoR.

[radiant\_sim\_qor](http://lsh-tmp/testrail/index.php?/suites/view/4235) – for simulation QoR.

If you have more QoR design/suite to launch, please upload it to this project first.

## QoR report location

Currently QoR report tool is a standalone script needs to be run on your local machine. This script written with Python, so you have to install Python and its packages first. See [6.Report generation details](#_Report_generation_detail)

# Generic QoR

Generic QoR flow allows user to run suite with default or specific project setting and generate QoR benchmark report version by version, LSE by Synplify or user comparable runs. We have four general methods for Generic QoR: qor\_pushbutton, qor\_sweep, qor\_seed, qor\_iter and we use qor\_pushbutton for Generic QoR run by default.

## Applicable areas

* Radiant version by version QoR benchmark
* Radiant synthesis tools LSE and Synplify QoR benchmark
* Radiant engine wi/wo specific feature or environment QoR benchmark

## Benchmark resource

All test designs can be found here:

<http://lsh-tmp/viewvc/radiant/trunk/QoR_suite/01_performance>

In this folder we have Benchmark suite for every device:

Avant:

Avant/A26

Jedi:

Jedi/E30

Jedi/L26

Jedi/S21

Jedi-D1:

Jedi\_D1/M29

Jedi-D2:

Reuse Jedi with project device updated.

Jedi-D6:

Reuse Jedi with project device updated.

Also, we created a suite file for all above test designs: QoR\_Radiant\_standard.xlsx and we have uploaded test suite to TMP for future QoR launch already.

TMP suite address: [QoR\_Radiant\_standard](http://lsh-tmp/testrail/index.php?/suites/view/4139)

## Default run and report

### Run command

It is recommended to run following command on TMP platform for QoR benchmark suite: [QoR\_Radiant\_standard](http://lsh-tmp/testrail/index.php?/suites/view/4139).

***“python DEV/bin/run\_radiant.py qor\_pushbutton”***

With this command you will have following default settings for each design run:

* Default project settings (device, strategy, synthesis tool, constraints)
* Run design till ‘PAR and post-PAR timing’
* All design run results will be collected, including area, performance, runtime, memory, filesize, filehash, status

### Report generation

Report generation flow can only be launched on your local machine with ‘DEV’ script prepared. See [Report generation details](#_Report_generation_Details)

***“python DEV/tools/reportGen/reportgen.py report --src <run id> --dst <run id> -f DEV/tools/reportGen/demo/report/mixed\_inputs/reused.ini”***

With this command you will get an Excel file for these two runs. See [Report generation details](#_Report_generation_Details) if you want different titles and calculation methods.

## QoR implementation

### Generic QoR with pushbutton method

By default, Generic QoR will be launched with “qor\_pushbutton” method. And you can see simple launch and report command in [Default run and report](#_Default_run_and), here we list some more details for it.

#### 2.4.1.1 Run method

For every test design, platform will run design implementation flow and scan implementation results like area, performance, runtime, memory usage…

Then with report generation script on your local machine generate the data report for comparable runs.

1. A diagram of a project

   Description automatically generated

#### 2.4.1.2 Quick launch and report

See [Default run and report](#_Default_run_and)

#### 2.4.1.3 More run options

“qor\_pushbutton” allow you do some more specific settings for a run project like what we do in GUI mode. All you need to do is add options into run commands like following:

***“python DEV/bin/run\_radiant.py qor\_pushbutton <additional\_option1> <additional\_option2> …”***

Here are some more typical options:

|  |  |
| --- | --- |
| General Options | Option Details |
| -E SET\_ENV, --set-env | specify user's environment value (append action) example: --set-env=a=1 --set-env="b=2;9;x" |
| -T TIMEOUT, --timeout | specify pnmainc timeout value |

|  |  |
| --- | --- |
| Project Options | Option Details |
| --run-ipgen | run ipgen in models folder for Radiant flow |
| --devkit DEVKIT | Update project device name |
| --synthesis {synplify,lse} | specify synthesis name, else use default one |
| -S SET\_STRATEGY, --set-strategy SET\_STRATEGY | set strategy, example: --set-strategy="syn\_res\_sharing=False maptrce\_check\_unconstrained\_connections=True” |

|  |  |
| --- | --- |
| Implementation Options | Option Details |
| --run-synthesis | run synthesis flow |
| --run-map-trace | Run Map Trace flow |
| --run-par-trace | Run Par Trace flow |
| --run-export-bitstream | generate Export Bitstream File |

|  |  |
| --- | --- |
| Scan Options | Option Details |
| -R, --scan-rpt | scan reports after flow done (default: True) |
| --scan-pap-rpt | scan reports after flow done (default: False) |
| --fmax-sort {max,geomean} | specify fmax sort way (default: max) |
| --ignore-clock IGNORE\_CLOCK | specify ignore clocks when scanning timing data |
| --care-clock CARE\_CLOCK | specify care only clocks when scanning timing data |

For a full option list, please run following command:

***“python DEV/bin/run\_radiant.py qor\_pushbutton --help”***

#### 2.4.1.4 More report options

No matter which QoR method used all design implementation results will be scan and dump to database for future use. So please refer to [Report generation details](#_Report_generation_Details)

#### 2.4.1.5 Demo run

[QoR\_Radiant\_standard\_qor\_pushbutton](http://lsh-tmp/testrail/index.php?/runs/view/247762)

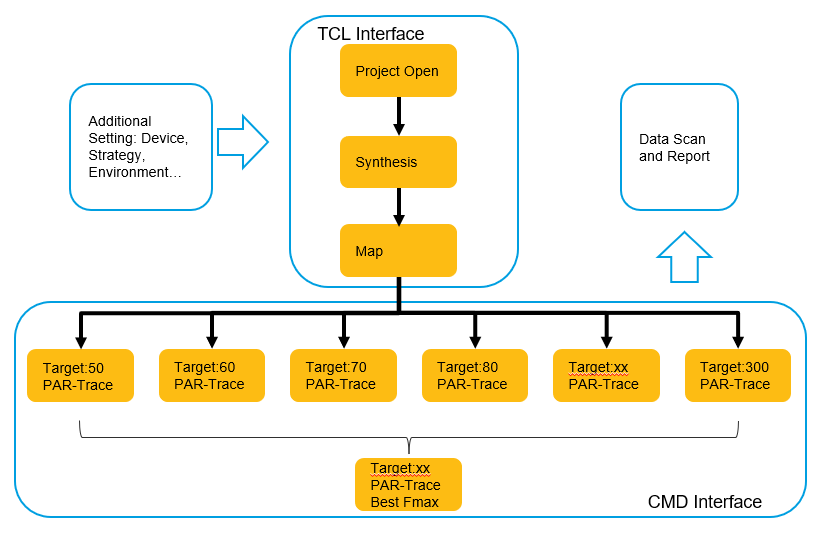
### Generic QoR with sweeping method

#### Run method

For every test design, platform will run design implementation flow to map, with this map result script will launch a bunch of sweeping points with different target fmax and run PAR-trace. Usually, the sweeping point can be defined as: from, to, interval, i.e., 50, 300, 10 which means target fmax set as: 50,60,70,80,90,100,110,120,130,140,150,160,170,180,190,200,210,220,230,240,250,260,270,280,290,300. Total 26 points (run 26 PAR-trace).

After implementation, scan the best fmax from all results and get performance, area data from that point. For other data like runtime, memory usage still based on the whole design.

Then with report generation script on your local machine generate the data report for above two comparable runs.

1. 

#### Quick launch and report

Launch:

It is recommended to run following command on TMP platform for QoR benchmark suite: [QoR\_Radiant\_standard](http://lsh-tmp/testrail/index.php?/suites/view/4139).

***“python DEV/bin/run\_radiant.py qor\_sweep”***

With this command you will have following default settings for each design run:

* Default project settings (device, strategy, synthesis tool, constraints)
* Run design till ‘MAP’ then fmax sweeping from 50 to 300 with a step of 10.
* No run result check (TMP run always pass)
* The best fmax sweeping point run results will be collected, including area, performance. Other data scan will be based on the whole design run: runtime, memory, filesize, filehash, status

Report:

Report generation flow can only be launched on your local machine with ‘DEV’ script prepared. See [Report generation details](#_Report_generation_Details) for details, here is a simple command for Generic QoR report generate.

***“python DEV/tools/reportGen/reportgen.py report --src <run id> --dst <run id> -f DEV/tools/reportGen/demo/report/mixed\_inputs/reused.ini”***

#### More run options

“qor\_sweep” allow you do some more specific settings for a run project like what we do in GUI mode. All you need to do is add options into run commands like following:

***“python DEV/bin/run\_radiant.py qor\_sweep <additional\_option1> <additional\_option2> …”***

Here are some more typical options:

|  |  |
| --- | --- |
| Sweeping Options | Option Details |
| --fmax-sweep <from,to,step> | specify fmax sweeping range (default: 50,300,10) |

|  |  |
| --- | --- |
| General Options | Option Details |
| -E SET\_ENV, --set-env | specify user's environment value (append action) example: --set-env=a=1 --set-env="b=2;9;x" |
| -T TIMEOUT, --timeout | specify pnmainc timeout value |

|  |  |
| --- | --- |
| Project Options | Option Details |
| --devkit DEVKIT | Update project device name |
| --synthesis {synplify,lse} | specify synthesis name, else use default one |

|  |  |
| --- | --- |
| Scan Options | Option Details |
| -R, --scan-rpt | scan reports after flow done (default: True) |
| --scan-pap-rpt | scan reports after flow done (default: False) |
| --fmax-sort {max,geomean} | specify fmax sort way (default: max) |
| --ignore-clock IGNORE\_CLOCK | specify ignore clocks when scanning timing data |
| --care-clock CARE\_CLOCK | specify care only clocks when scanning timing data |

For a full option list, please run following command:

***“python DEV/bin/run\_radiant.py qor\_sweep --help”***

#### 2.4.2.4 More report options

No matter which QoR method used all design implementation results will be scan and dump to database for future use. So please refer to [Report generation details](#_Report_generation_Details)

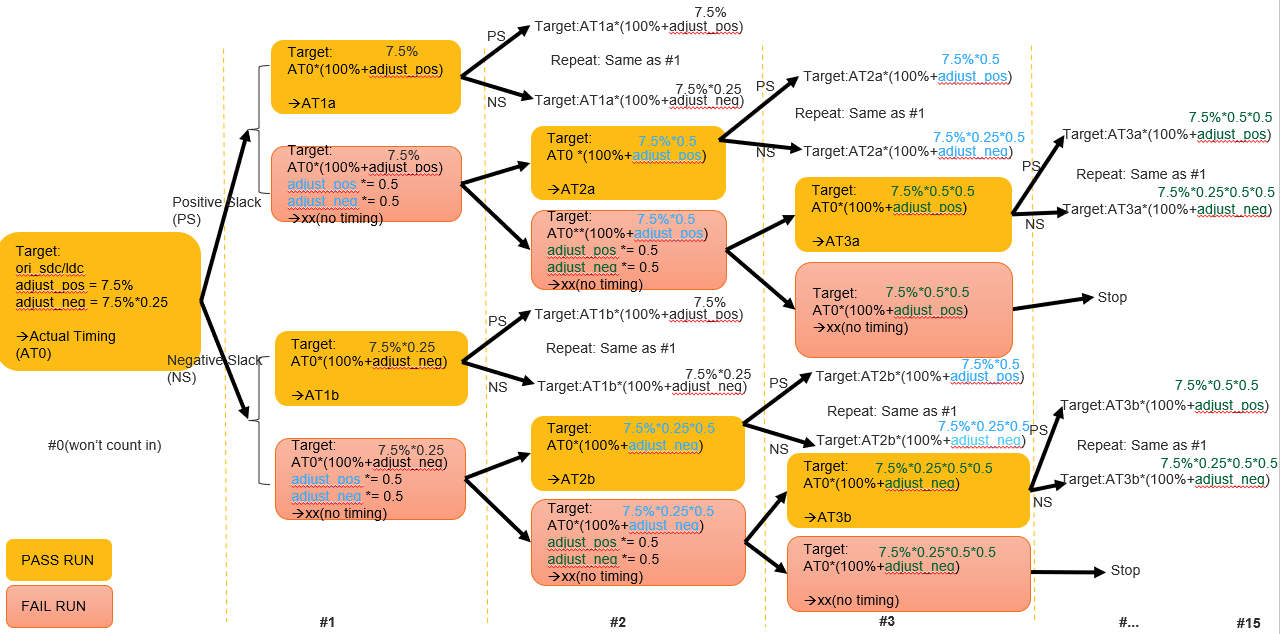
#### Demo run

[QoR\_Radiant\_standard\_qor\_sweep](http://lsh-tmp/testrail/index.php?/runs/view/247919)

### Generic QoR with Iteration method

#### Run method

“qor\_iter” will run an initial result based on design constraint file and run following 15 additional ‘PAR-trace’(iterations) based on previous actual timing result plus adjustments. Now “qor\_iter” is the default method for Avant device Generic QoR.

1. 

Key methods summary:

* Set adjust\_pos = 7.5%, adjust\_neg = 7.5% \* 0.25
* Run design flow till par-trace and get ‘actual frequency.’
* If current Run Pass:

Update target fmax based on timing report for next run.

If the slack is positive, Target fmax will be last\_actual\_frequency \* adjust\_pos

If the slack is negative, Target fmax will be last\_actual\_frequency \* adjust\_neg

* If current Run Fail: Update target fmax based on last success timing report for next run.

adjust\_pos = adjust\_pos \* 0.5

adjust\_neg = adjust\_neg \* 0.5

If last pass run slack is positive, Target fmax will be last\_actual\_frequency \* adjust\_pos

If last pass run slack is negative, Target fmax will be last\_actual\_frequency \* adjust\_neg

* Repeat 15 iterations (excluding the initial one) with the following exceptions:

If the fmax of current run is same as last run, will stop next iteration.

If current iteration flow fails 3 times continuously, stop next iteration.

* For each iteration, pick the best fmax for each clock,
* fmax calculation is based on geomean of all clocks for each iteration.
* For all 15 iterations, pick the best geofmax

#### Quick launch and report

Launch:

It is recommended to run following command on TMP platform for QoR benchmark suite: [QoR\_Radiant\_standard](http://lsh-tmp/testrail/index.php?/suites/view/4139).

***“python DEV/bin/run\_radiant.py qor\_iter”***

With this command you will have following default settings for each design run:

* Default project settings (device, strategy, synthesis tool, constraints)
* Run design till ‘PAR-trace’ then iteration based on previous actual fmax for 15 times.
* No run result check (TMP run always pass)
* The best fmax point run results will be collected, including area, performance. Other data scan will be based on the whole design run: runtime, memory, filesize, filehash, status.

Report:

Report generation flow can only be launched on your local machine with ‘DEV’ script prepared. See [Report generation details](#_Report_generation_Details) for details, here is a simple command for Generic QoR report generate.

***“python DEV/tools/reportGen/reportgen.py report --src <run id> --dst <run id> -f DEV/tools/reportGen/demo/report/mixed\_inputs/reused.ini”***

#### More run options

Here are some more typical options:

|  |  |
| --- | --- |
| Sweeping Options | Option Details |
| --fmax-iteration | specify fmax iteration setting: (adjust\_percentage iteration\_number (default: (7.5, 14)) |

|  |  |
| --- | --- |
| General Options | Option Details |
| -E SET\_ENV, --set-env | specify user's environment value (append action) example: --set-env=a=1 --set-env="b=2;9;x" |
| -T TIMEOUT, --timeout | specify pnmainc timeout value |

|  |  |
| --- | --- |
| Project Options | Option Details |
| --devkit DEVKIT | Update project device name |
| --synthesis {synplify,lse} | specify synthesis name, else use default one |

|  |  |
| --- | --- |
| Scan Options | Option Details |
| -R, --scan-rpt | scan reports after flow done (default: True) |
| --scan-pap-rpt | scan reports after flow done (default: False) |
| --fmax-sort {max,geomean} | specify fmax sort way (default: max) |
| --ignore-clock IGNORE\_CLOCK | specify ignore clocks when scanning timing data |
| --care-clock CARE\_CLOCK | specify care only clocks when scanning timing data |

For a full option list, please run following command:

***“python DEV/bin/run\_radiant.py qor\_iter --help”***

#### 2.4.3.4 More report options

No matter which QoR method used all design implementation results will be scan and dump to database for future use. So please refer to [Report generation details](#_Report_generation_Details)

#### 2.4.3.5 Demo run

[QoR\_Radiant\_standard\_qor\_iter](http://lsh-tmp/testrail/index.php?/runs/view/247920)

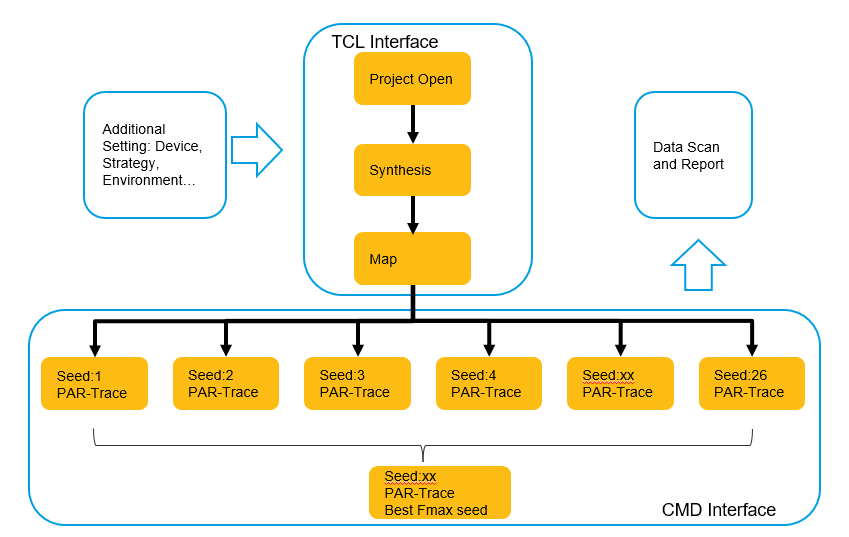
### Generic QoR with seed method

#### Run method

For every test design, platform will run design implementation flow to map, with this map result script will launch a bunch of seed points with different start point for PAR-trace. Usually, the seed point can be defined as: from, to, interval, i.e.: 1, 26, 1 which means PAR start point set as: 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26. Total 26 points (run 26 PAR-trace).

After implementation, scan the best fmax from all results and get performance, area data from that seed. For other data like runtime, memory usage still based on the whole design.

Then with report generation script on your local machine generate the data report for above two comparable runs.

1. 

#### Quick launch and report

Launch:

It is recommended to run following command on TMP platform for QoR benchmark suite: [QoR\_Radiant\_standard](http://lsh-tmp/testrail/index.php?/suites/view/4139).

***“python DEV/bin/run\_radiant.py qor\_seed”***

With this command you will have following default settings for each design run:

* Default project settings (device, strategy, synthesis tool, constraints)
* Run design till ‘MAP’ then seed sweeping from 1 to 26 with a step of 1.
* No run result check (TMP run always pass)
* The best seed sweeping point run results will be collected, including area, performance. Other data scan will be based on the whole design run: runtime, memory, filesize, filehash, status.

Report:

Report generation flow can only be launched on your local machine with ‘DEV’ script prepared. See [Report generation details](#_Report_generation_Details) for details, here is a simple command for Generic QoR report generate.

***“python DEV/tools/reportGen/reportgen.py report --src <run id> --dst <run id> -f DEV/tools/reportGen/demo/report/mixed\_inputs/reused.ini”***

#### More run options

“qor\_seed” allow you do some more specific settings for a run project like what we do in GUI mode. All you need to do is add options into run commands like following:

***“python DEV/bin/run\_radiant.py qor\_seed <additional\_option1> <additional\_option2> …”***

Here are some more typical options:

|  |  |
| --- | --- |
| Sweeping Options | Option Details |
| --seed-sweep <from,to,step> | specify seed sweeping range (default: 1,26,1) |

|  |  |
| --- | --- |
| General Options | Option Details |
| -E SET\_ENV, --set-env | specify user's environment value (append action) example: --set-env=a=1 --set-env="b=2;9;x" |
| -T TIMEOUT, --timeout | specify pnmainc timeout value |

|  |  |
| --- | --- |
| Project Options | Option Details |
| --devkit DEVKIT | Update project device name |
| --synthesis {synplify,lse} | specify synthesis name, else use default one |

|  |  |
| --- | --- |
| Scan Options | Option Details |
| -R, --scan-rpt | scan reports after flow done (default: True) |
| --scan-pap-rpt | scan reports after flow done (default: False) |
| --fmax-sort {max,geomean} | specify fmax sort way (default: max) |
| --ignore-clock IGNORE\_CLOCK | specify ignore clocks when scanning timing data |
| --care-clock CARE\_CLOCK | specify care only clocks when scanning timing data |

For a full option list, please run following command:

***“python DEV/bin/run\_radiant.py qor\_seed --help”***

#### 2.4.2.4 More report options

No matter which QoR method used all design implementation results will be scan and dump to database for future use. So please refer to [Report generation details](#_Report_generation_Details)

#### Demo run

[QoR\_Radiant\_standard\_qor\_seed](http://lsh-tmp/testrail/index.php?/runs/view/247921)

# Simulation QoR

## Applicable areas

* Simulator version by version QoR benchmark
* Simulation model version by version QoR benchmark
* Different device simulation models QoR benchmark
* Different simulators QoR benchmark

## Benchmark resource

All test designs can be found here:

<http://lsh-tmp/viewvc/radiant/trunk/QoR_suite/02_sim_model/>

In this folder we have Benchmark suite for following device and models:

Avant:

Avant/DDR

Avant/DSP

Avant/EBR

Avant/PLL

Avant/SLICE

Avant/Mix\_Design

Jedi:

Jedi/DDR

Jedi/DSP

Jedi/EBR

Jedi/PLL

Jedi/SLICE

Jedi/Mix\_Design

Also, we created a suite file for all above test designs: radiant\_sim\_qor.xlsx and we upload test suite to TMP for future QoR launch already.

TMP suite address: [radiant\_sim\_qor](http://lsh-tmp/testrail/index.php?/suites/view/4235)

## Default run and report

### Run command

It is recommended to run following command on TMP platform for QoR benchmark suite: [radiant\_sim\_qor](http://lsh-tmp/testrail/index.php?/suites/view/4235).

***“python DEV/bin/run\_radiant.py qor\_sim”***

With this command you will have following default settings for each design run:

* Default project settings (device, strategy, synthesis tool, constraints)
* Run design RTL simulation, Post-syn simulation, Post-par simulation
* No simulation waveform lst file dump
* All design run results will be collected, including simulation runtime, memory used.

### Report generation

Report generation flow can only be launched on your local machine with ‘DEV’ script prepared. See [Report generation details](#_Report_generation_Details)

***“python DEV/tools/reportGen/reportgen.py report --src <run id> --dst <run id> -f DEV/tools/reportGen/demo/report/mixed\_inputs/qor\_sim.ini”***

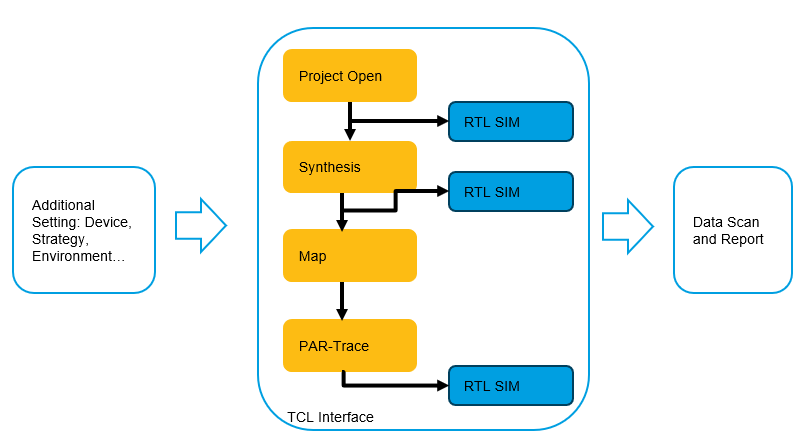
With this command you will get an Excel file for these two runs. See [Report generation details](#_Report_generation_Details) if you want different titles and calculation methods.

## QoR implementation

### Run method

For every test design, platform will run design simulation flow for RTL, Post-syn, Post-par and scan simulation results like runtime, memory used…

Then with report generation script on your local machine generate the data report for comparable runs.

1. 

### More run options

“qor\_sim” allow you do some more specific settings for a run project like what we do in GUI mode. All you need to do is add options into run commands like following:

***“python DEV/bin/run\_radiant.py qor\_sim <additional\_option1> <additional\_option2> …”***

Here are some more typical options:

|  |  |
| --- | --- |
| Simulation Options | Option Details |
| --sim-modelsim | run simulation with Modelsim (default: False) |
| --sim-questasim | run simulation with Questasim (default: False) |
| --library-path LIBRARY\_PATH | If no value for this option, will try to compile/search the library with given simulation tool (default: None) |
| --library-precompiled {yes,no,auto} | Valid when 'library-path' is empty.  1. yes: use precompiled simulation library, block flow if not found the library(questasim/riviera)  2. no: compile library with simulator  3. auto: default modelsim: use pre-compiled library;others: compile library (default: None) |
| --sim-all | run all simulation flow (default: False) |
| --sim-with-sdf {min,typ,max} | specify the type of running post-map/par simulation flow with sdf file, valid choices are min,typ,max (default: None) |

|  |  |
| --- | --- |
| General Options | Option Details |
| -E SET\_ENV, --set-env | specify user's environment value (append action) example: --set-env=a=1 --set-env="b=2;9;x" |
| -T TIMEOUT, --timeout | specify pnmainc timeout value |

|  |  |
| --- | --- |
| Project Options | Option Details |
| --devkit DEVKIT | Update project device name |
| --synthesis {synplify,lse} | specify synthesis name, else use default one |

|  |  |
| --- | --- |
| Scan Options | Option Details |
| -R, --scan-rpt | scan reports after flow done (default: True) |
| --scan-pap-rpt | scan reports after flow done (default: False) |
| --fmax-sort {max,geomean} | specify fmax sort way (default: max) |
| --ignore-clock IGNORE\_CLOCK | specify ignore clocks when scanning timing data |
| --care-clock CARE\_CLOCK | specify care only clocks when scanning timing data |

For a full option list please run following command:

***“python DEV/bin/run\_radiant.py qor\_sim --help”***

### More report options

No matter which QoR method used all design simulation results will be scan and dump to database for future use. So please refer to [Report generation details](#_Report_generation_Details)

### Demo run

[QoR\_Radiant\_standard\_qor\_sim](http://lsh-tmp/testrail/index.php?/runs/view/247973)

# Power QoR

## Applicable areas

* Power version by version QoR benchmark
* Different tools and devices power QoR benchmark

## Benchmark resource

Same as Generic QoR, see [2.2 Benchmark resource](#_Benchmark_source)

## Default run and report

### Run command

It is recommended to run following command on TMP platform for QoR benchmark suite: [QoR\_Radiant\_standard](http://lsh-tmp/testrail/index.php?/suites/view/4139).

***“python DEV/bin/run\_radiant.py qor\_power”***

With this command you will have following default settings for each design run:

* Default project settings (device, strategy, synthesis tool, constraints)
* Set Power Calculator Process Type: Typical
* Set Power Ambient Temperature: 25℃
* Set Power Effective Theta-JA: 4.25
* Set Power Activity Factor (Keep Clock AF Settings): 12.5
* Set Power Frequency: 0
* All design run results will be collected, including StaticPower, DynamicPower, TotalPower

### Report generation

Report generation flow can only be launched on your local machine with ‘DEV’ script prepared. See [Report generation details](#_Report_generation_Details)

***“python DEV/tools/reportGen/reportgen.py report --src <run id> --dst <run id> -f DEV/tools/reportGen/demo/report/mixed\_inputs/qor\_power.ini”***

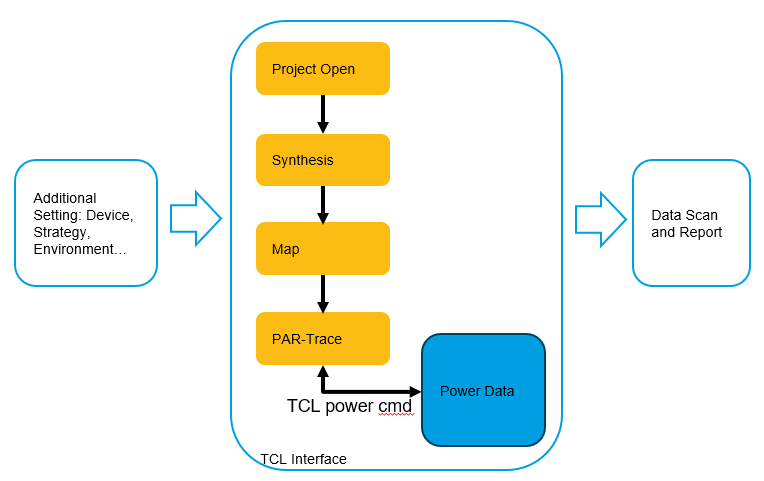
With this command you will get an Excel file for these two runs. See [Report generation details](#_Report_generation_Details) if you want different titles and calculation methods.

## QoR implementation

### Run method

For every test design, platform will run design general implementation flow and insert power TCL commands then scan power results like static power, dynamic power, and total power.

Then with report generation script on your local machine generate the data report for comparable runs.

1. 

As you see from above picture after PAR done script will run following TCL commands to get power data:

***pwc\_new\_project auto\_power.pcf -udb Impl/PrjName\_Impl.udb***

***pwc\_set\_processtype "Typical"***

***pwc\_set\_ambienttemp 25***

***pwc\_set\_thetaja 4.25***

***pwc\_set\_af 12.5***

***pwc\_set\_freq -freq 0 -usefreqtwr 1 -freqtwropt pref***

***pwc\_calculate***

***pwc\_gen\_report auto\_power\_result.pwr***

***pwc\_gen\_htmlreport auto\_power\_result.html***

***pwc\_save\_project auto\_power.pcf***

### More run options

“qor\_power” allow you do some more specific settings for a run project like what we do in GUI mode. All you need to do is add options into run commands like following:

***“python DEV/bin/run\_radiant.py qor\_power <additional\_option1> <additional\_option2> …”***

Here are some more typical options:

|  |  |
| --- | --- |
| PowerCalculator Options | Option Details |
| --pwr-type {Typical,Worst} | specify Power Calculator Process Type, default is Typical |
| --pwr-at PWR\_AT | specify Power Ambient Temperature, default is 25 |
| --pwr-etga PWR\_ETGA | specify Power Effective Theta-JA, default is 4.25 |
| --pwr-af PWR\_AF | specify Power Activity Factor (Keep Clock AF Settings), default is 12.5 |
| --pwr-fre PWR\_FRE | specify Power Frequency Setting, default is 0 |
| --pwr-fre-twr {min,pref,trace} | specify Power Use Frequency TWR mode |

|  |  |
| --- | --- |
| General Options | Option Details |
| -E SET\_ENV, --set-env | specify user's environment value (append action) example: --set-env=a=1 --set-env="b=2;9;x" |
| -T TIMEOUT, --timeout | specify pnmainc timeout value |

|  |  |
| --- | --- |
| Project Options | Option Details |
| --devkit DEVKIT | Update project device name |
| --synthesis {synplify,lse} | specify synthesis name, else use default one |

|  |  |
| --- | --- |
| Scan Options | Option Details |
| -R, --scan-rpt | scan reports after flow done (default: True) |
| --scan-pap-rpt | scan reports after flow done (default: False) |
| --fmax-sort {max,geomean} | specify fmax sort way (default: max) |
| --ignore-clock IGNORE\_CLOCK | specify ignore clocks when scanning timing data |
| --care-clock CARE\_CLOCK | specify care only clocks when scanning timing data |

For a full option list please run following command:

***“python DEV/bin/run\_radiant.py qor\_power --help”***

### More report options

No matter which QoR method used all design implementation results will be scan and dump to database for future use. So please refer to [Report generation details](#_Report_generation_Details)

### Demo run

[QoR\_Radiant\_standard\_qor\_power](http://lsh-tmp/testrail/index.php?/runs/view/247974)

# Repeatability QoR

## Applicable areas

* Engine stability check

## Benchmark resource

Same as Generic QoR, see [2.2 Benchmark resource](#_Benchmark_source)

## Default run and report

### Run command

It is recommended to run following command on TMP platform for QoR benchmark suite: [QoR\_Radiant\_standard](http://lsh-tmp/testrail/index.php?/suites/view/4139).

***“python DEV/bin/run\_radiant.py qor\_step”***

With this command you will have following default settings for each design run:

* Default project settings (device, strategy, synthesis tool, constraints)
* With previous run results rerun Map 3 times
* With previous run results rerun Place 3 times
* With previous run results rerun Route 3 times
* No run result check (TMP run always pass)
* All design run results will be collected, including runtime, memory, filesize, filehash.

### Report generation

Report generation flow can only be launched on your local machine with ‘DEV’ script prepared. See [Report generation details](#_Report_generation_Details)

***“python DEV/tools/reportGen/reportgen.py report --src <run id> --dst <run id> -f DEV/tools/reportGen/demo/report/mixed\_inputs/qor\_step.ini”***

With this command you will get an Excel file for these two runs. See [Report generation details](#_Report_generation_Details) if you want different titles and calculation methods.

## QoR implementation

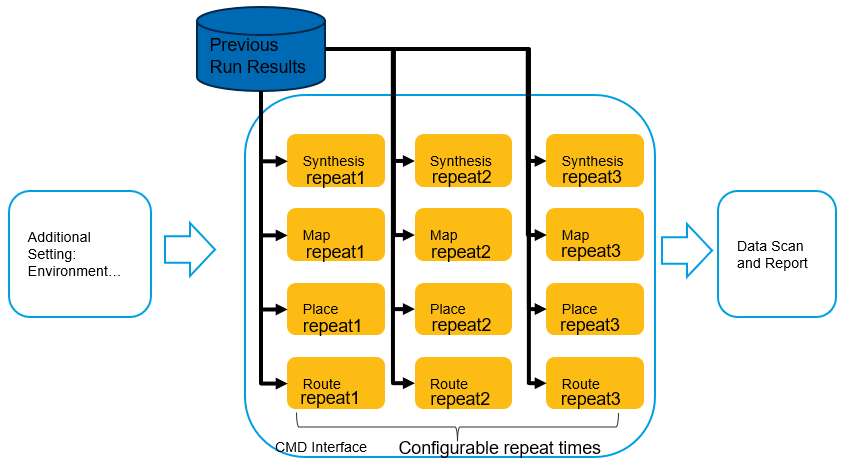
### Run method

This method can run one engine many times and dump the SHA1 code for engine's output file for future checking. In order to make sure the engine start from the same inputs, we should prepare the results by running command like:

* Prepare files like: PrjName\_Impl\_syn.udb, PrjName\_Impl\_map.udb and PrjName\_Impl.udb(placer output) by executing ***python DEV/bin/run\_radiant.py –setstrategy "{par\_cmdline\_args=-exp parMessageVerbose=1}" --run-par --set-strategy "par\_run\_place\_only=True"***
* Prepare files like: PrjName\_Impl\_syn.udb, PrjName\_Impl\_map.udb and PrjName\_Impl.udb(par output) by executing ***python DEV/bin/run\_radiant.py –setstrategy "{par\_cmdline\_args=-exp parMessageVerbose=1}" --pushbutton***

Since udb files are in bit format, we should transfer udb files to Text file (by udb2sv) and ignore timestamp message when calculating SHA1 code. At the meantime, checksum can be parsed from \*.mrp and \*.par. The SHA1 code for twr file will try to ignore line orders between "1.4 Unconstrained Report" and "1.5 Combinational Loop". Sometimes the list for "Start/End Points Without Timing Constraints" uses random order.

With previous design prepared, we can start “qor\_step” flow like following:

1. 

### More run options

“qor\_step” allow you do some more specific settings for a run project. All you need to do is add options into run commands like following:

***“python DEV/bin/run\_radiant.py qor\_step <additional\_option1> <additional\_option2> …”***

Here are some more basic commands:

LSE repeatability test:

***python DEV/bin/run\_radiant.py qor\_step --synthesis lse --run-step-synthesis --step-times 3***

Synplify repeatability test:

***python DEV/bin/run\_radiant.py qor\_step –synthesis synplify --run-step-synthesis --step-times 3***

Map, placer, router, par repeatability test:

***python DEV/bin/run\_radiant.py qor\_step --run-step map placer router --step-times 3 --par-threads 4***

Timing repeatability test:

***python DEV/bin/run\_radiant.py qor\_step--run-step timing --step-times 3***

Router-threads repeatability test:

***python DEV/bin/run\_radiant.py qor\_step --run-step router --par-threads 1 2 4 8***

Par-threads repeatability test:

***python DEV/bin/run\_radiant.py qor\_step --run-step par --par-threads 1 2 4 8***

Here are some more typical options:

|  |  |
| --- | --- |
| Step repeatability Options | Option Details |
| --run-step {map,placer,router,par,timing} | run single step flow |
| --par-threads {-1,0,1,2,4,8} | specify step par threads |
| --run-step-synthesis | run step synthesis flow |
| --step-times STEP\_TIMES | times number for running a step (default: 3) |

|  |  |
| --- | --- |
| General Options | Option Details |
| -E SET\_ENV, --set-env | specify user's environment value (append action) example: --set-env=a=1 --set-env="b=2;9;x" |
| -T TIMEOUT, --timeout | specify pnmainc timeout value |

|  |  |
| --- | --- |
| Scan Options | Option Details |
| -R, --scan-rpt | scan reports after flow done (default: True) |
| --scan-pap-rpt | scan reports after flow done (default: False) |
| --fmax-sort {max,geomean} | specify fmax sort way (default: max) |
| --ignore-clock IGNORE\_CLOCK | specify ignore clocks when scanning timing data |
| --care-clock CARE\_CLOCK | specify care only clocks when scanning timing data |

For a full option list, please run following command:

***“python DEV/bin/run\_radiant.py qor\_step --help”***

### More report options

No matter which QoR method used all design implementation results will be scan and dump to database for future use. So please refer to [Report generation details](#_Report_generation_Details)

### Demo run

[Repeatability\_ng2023\_2p.38.1](http://lsh-tmp/testrail/index.php?/plans/view/246658)

# Report generation details

Report generation script is a standalone script run on local machine to generate a compare report. So, it’s designed to support all previous QoR implementation.

## Applicable areas

Report generation script is designed with high flexibility. It supports:

* [Specific]Generic QoR report generation
* [Specific]Simulation QoR report generation
* [Specific]Repeatability QoR report generation
* [Specific]Power QoR report generation
* [General]Two TMP runs with same/specific section and designs.
* [General]One TMP run with Two comparable sections.

## Script export

Please use following command to export scripts:

***“svn export*** [***http://lsh-tmp/platform/trunk/tmp\_scripts/DEV***](http://lsh-tmp/platform/trunk/tmp_scripts/DEV)***”***

Then you will find report generation script here: DEV/tools/reportGen/reportgen.py.

## Build environment

### Python and package installation

Source:

[\\lsh-smb02\sw\qa\qa\_store\TMP\_tools\depend\_software\python\_and\_packages](file:///\\lsh-smb02\sw\qa\qa_store\TMP_tools\depend_software\python_and_packages)

Install Python:

1. Double click: python-3.8.7-amd64.exe (Windows) or unzip Python-3.8.7.tar.xz(Linux)

2. Put Python install path into environment ‘PATH’ variable

Install Python packages:

1. Copy ‘offline’ folder to your local machine

2. Go into ‘offline’ folder

3. Run following commands in console:

a. python -m pip install --no-index --find-links . psutil==5.8.0

b. python -m pip install --no-index --find-links . JIRA==2.0.0

c. python -m pip install --no-index --find-links . openpyxl==3.0.7

d. python -m pip install --no-index --find-links . pymysql==1.0.2

Check:

Run following commands in console:

1. python -- Python should be run with a correct version

2. import psutil -- Python package imported, no crash

3. import jira -- Python package imported, no crash

4. import openpyxl -- Python package imported, no crash

5. import pymysql -- Python package imported, no crash

### local package installation

Additional package for report generation: capuchin.

Install:

a. Export: svn export http://lsh-tmp/platform/trunk/python3/capuchin-master/

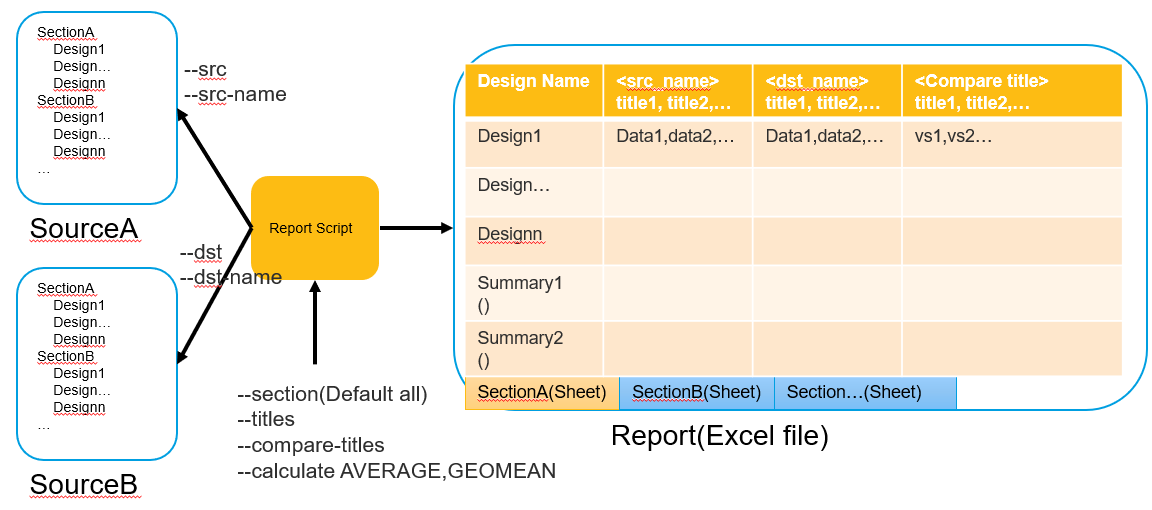
b. Install: python setup.py install

## Report generation Script methods

Report generation script is designed with high flexibility. It supports general data compare for two runs and for QoR report with specific format required.

### Report for two data sources

Report script can get data from two data sources and apply user specified titles for comparison as follows:

1. 

The ‘Sourcex’ can be:

* TMP run id
* An Excel data file with ‘Sectionx’ name as sheet name and ‘Designx’ data in details

### Report for same data source two sections

Report script can also get data from same data sources and apply user specified titles for comparison as follows:

1. A diagram of a data flow

   Description automatically generated with medium confidence

The ‘Sourcex’ can be:

* TMP run id
* An Excel data file with ‘Sectionx’ name as sheet name and ‘Designx’ data in details

## Report generation simple commands

### Two data sources compare report

***“python DEV/tools/reportGen/reportgen.py report --src <run id1> --dst <run id2> --titles version,device,Register,LUT,Slice,fmax,par\_cpu\_time --compare-titles fmax,Register,LUT,Slice,par\_cpu\_time --add-test-id”***

With this command we will have:

* Design data from two source
* All matched sections from two source will be compared
* With user specified/cared title
* With user specified compare title
* Add test id in table for will result search
* Use default statistical method ‘AVERAGE’

### One data source two sections compare report

***“python DEV/tools/reportGen/reportgen.py report --src <run id1> --dst <run id1> --src-section <section1> --dst-section <section2> --titles version,device,Register,LUT,Slice,fmax,par\_cpu\_time --compare-titles fmax,Register,LUT,Slice,par\_cpu\_time -c GEOMEAN”***

With this command we will have:

* Design data from one source, two different sections: section1, section2
* With user specified/cared title
* With user specified compare title
* Statistical method ‘GEOMEAN’ used

## Report generation detail options

Here is a list of typical options used during report generation:

|  |  |
| --- | --- |
| Scan Options | Option Details |
| -a SRC, --src SRC | specify source report file or source run id |
| -b DST, --dst DST | specify destination report file or destination run id |
| --src-name SRC\_NAME | specify source name, will be shown in Excel |
| --dst-name DST\_NAME | specify destination name, will be shown in Excel |
| --src-section SRC\_SECTION | specify source data section |
| --dst-section DST\_SECTION | specify destination data section |
| --section SECTION | specify section, will be used as src\_section or dst\_section ONLY when they are not specified |
| --src-titles SRC\_TITLES | specify source data titles |
| --dst-titles DST\_TITLES | specify destination data titles |
| --titles TITLES | specify titles, will be used as src\_titles or dst\_titles ONLY when they are not specified. |
| --compare-titles COMPARE\_TITLES | specify compare titles |
| --calculate {AVERAGE,STDEVPA,GEOMEAN} | specify statistical method(s) |
| --add-test-id | add column xTestID in final report |

For a full option list, please run following command:

***“python DEV/tools/reportGen/reportgen.py report --help”***