User Platform Core script tools support usage

**Cherry Xu**

Collect all tools used by LSH QA Team, includes location, description and usage sections for each tool.

REVISION HISTORY

|  |  |  |
| --- | --- | --- |
| REVISION | RELEASE DATE | COMMENTS |
| V1.0 | 12/5/2016 | Initial draft |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Contents

[1. Check 3](#_Toc468883164)

[1.1 Location 3](#_Toc468883165)

[1.2 Description 3](#_Toc468883166)

[1.3 Usage 3](#_Toc468883167)

[2. scan\_report 8](#_Toc468883168)

[2.1 Location 8](#_Toc468883169)

[2.2 Description 8](#_Toc468883170)

[2.3 Usage 8](#_Toc468883171)

**1 Check**

* 1. Location

http://Platform/trunk/bqs\_scripts/trunk/tools/check

* 1. Description

The tool usually performs checking report after running flow or other operation.

* 1. Usage

**python check.py --help**

You can use this demo to get detailed help for check script.

**Python** **check.py** **--design=07\_i2c\_port\_enable** **--top-dir=test**

Generally we should specify the --top-dir which is your work dir, --design which specify the design name.

The check is based on the config file. If there is any fail in the check-report, you should check the config file and debug with the test object.

**python suite\_support.py --top-dir=d:\test-dir\bchen\D3\_2 --suite-file=case\_list.ini --suite-name=suite --check --options=" --report-path=rpt --report=check\_result.csv"**

This case is applied to check the report by suite, and it specifies the report-path and report name.

**1.3.1 Options**

--h, --help show this help message and exit

--top-dir=TOP\_DIR specify top working directory

--design=DESIGN specify design name

--conf-file=CONF\_FILE specify configure file if you know

--report-path=REPORT\_PATH Specify where you want to store the report

--tag=TAG replace the tag in the conf file

--report=REPORT specify report name, default is check.csv

--rerun-path=RERUN\_PATH Specify the directory for the rerun.bat to be stored

--lse-check Just Check lse(create conf file)

--synp-check Just Check synplify(create conf file)

--map-check Just Check map(create conf file)

--partrce-check Just Check partrce(create conf file)

--case-command=CASE\_COMMAND

specify case command, create conf file according to the command

**1.3.2 Configuration**

For the check tool, the key point is conf file:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[configuration information]

area = STA

type = Arc

[method]

check\_lines\_1 = 1

check\_lines\_2 = 1

check\_block = 1

check\_data = 1

check\_flow = 1

[check\_lines\_1]

title = check\_test1

file = par\ecp3\<id>\NoPortName\_one.twr

check\_1 = Preference: FREQUENCY PORT "clka" 10.000000 MHz ;

check\_2 = 2 items scored

[check\_lines\_2]

title = check\_test2

file = par\ecp3\<id>\NoPortName\_one.twr

check\_1 = Preference: FREQUENCY 20.000000 MHz ;

check\_2 = 4 items scored

[check\_block]

title = check\_blocks

compare\_file = par\ecp3\<id>\NoPortName\_one.jed

golden\_file = Gold\_NoPortName\_one.jed

[check\_data]

file=par\ecp3\<id>\NoPortName\_one.twr

start\_line = Preference: INPUT\_SETUP GROUP "Data" 5.000000 ns CLKPORT "Clock" ;

result=6,8

line1=20,1,-

line2=19,1,+

line3=21,1,-

line4=22,1,-

line5=23,1

[check\_flow]

file=par\ecp3\<id>\NoPortName\_one.par

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

NOTE:(1) for auto check, one (or more than one) method failure will lead to the case's failure in report.

(2) user can use <id> to replace the folder named "id\_scratch" in <path>

(3) if the item value is set to 1, then script will enable this method

(4) if one method is used more than once, it can be named as check\_lines\_1, check\_lines\_2, ...

**1.3.3 Method**

1.check\_lines

(1)format

[check\_lines]

title = check\_test1

file = <path>\<file>

check\_1 = <string\_1>

times = <number> --optional

check\_<num> = <string\_2>

(2)description

this method will try to find the <string\_1> assigned by check\_1 in <file> and treat it as a start point (line 1),

(if times option is used, sripts will find <string\_1> <number> times and then treat the last one as start point),

then try to check whether line <num> (a shift value) has <string\_2>. if yes, the result of this method is true.

2.check\_data

(1)format

[check\_data]

file=<path>\<file>

start\_line = <string\_1>

times = <number> --optional

result = <num> / <line>,<shift>

line<num1> = <num>,<operation\_symbol> / <line>,<shift>,<operation\_symbol>

line<num2> = <num>,<operation\_symbol> / <line>,<shift>,<operation\_symbol>

...

line<numn> = <num> / <line>,<shift>

(2)description

this method will try to find the <string\_1> assigned by start\_line in <file> and treat it as a start point (line 1),

(if times option is used, sripts will find <string\_1> <number> times and then treat the last one as start point),

then try to calculate the the result.

user can either use the absolute number or to use the line + shift to indicate the number.

3.check\_block

(1)format

[check\_block]

title = check\_blocks

compare\_file = <path>\<file>

golden\_file = <Gold\_file>

(2)description

this method will try to compare the <Gold\_file> and <file>, if <Gold\_file> is included in <file>, return true.

4.check\_flow

(1)format

[check\_flow]

file=<path>\<file>

(2)description

this method will try to find the string "All signals are completely routed." in the par report.

<file> need to be a par report.

5.check\_multiline

(1)format

[check\_multiline]

file = <path>\<file>

check\_line = <strings\_in\_multi\_lines>

(2)description

Sometimes, the string you want to check will be divided and printed in multi-lines due to different reasons (like the length of path may be changed and affect the message you want).

This method will check a continuous string regardless the "space" and "line break".

6.check\_no

(1)format

[check\_no]

file = <path>\<file>

check\_line =<string>

(2)description

this method will try to find the <string>. If the <string> is not found, the method return pass.

7.check\_grep

(1)format

[check\_grep]

file = <path>\<file>

grep = <grep>

modifier = <modifier> --optional

(2)description

This method will try to search the file with the given "regular expression" list by check\_grep.

modifier will be used to list the search behavior, such as: re.I or re.IGNORECASE : ignore case sensitive.

For example:

if you want to check the format of the following string:

Number of registers: 4 out of 877 (0%)

you can use:

grep = number of registers:\s+(\d+?)\sout\sof\s+(\d+?)\s\(

modifier = re.IGNORECASE

8. check\_compare\_par

This method used to check the seed order.

1. format:

[method]

check\_compare\_par = 1

[check\_compare\_par]

mode = <MODE> at here, mode can be valued as ws or ts. The default is ws

par\_dir = <dir>, the par file directory.

9. check\_sdf

This method used to check the sdf file

1. format:

[method]

check\_sdf = 1

[check\_sdf]

sdf\_dir = <dir>, the directory of sdf file directory.

|  |  |  |
| --- | --- | --- |
| **Flow** | **File** | **Check string** |
| synp | \_scratch/impl/top\_impl.srr | Mapper successful! |
| lse | \_scratch/impl/synthesis.log | Elapsed CPU time for LSE flow |
| map | \_scratch/impl/top\_impl.mrp | Number of errors:    0 |
| par | \_scratch/impl/top\_impl.par | All signals are completely routed. |
| ibis | \_scratch/run\_pb.log | Generating: C:\Users\jye\Desktop\sdfsd\case\02\_synp\\_scratch\impl\IBIS\top\_impl.ibs |
| bitstream | \_scratch/run\_pb.log | Saving bit stream in "top\_impl.bit" or Saving bit stream in " top\_impl.rbt" |
| jedec | \_scratch/run\_pb.log | Saving bit stream in "top\_impl.jed" |
| prom | \_scratch/run\_pb.log | The file top\_impl.mcs was generated successfully |
| download | \_scratch/run\_pb.log | Saving bit stream in "top\_impl.bit" or Saving bit stream in " top\_impl.rbt" or Saving bit stream in "top\_impl.jed" |

**Attentions:**

Before you run this file, you have to make sure the pushbutton is done

1. if --force is used, the ldf file will transfer to info file
2. if not use --force and info file in the case directory, the info file will be used.
3. if synthesis tool is lse, a flag file will be used for pushbutton
4. self.case\_list+self.suite file be used to run parallel, and the \*\_temp file will be produced at that time so, before you run it, please del the self.case\_list\_self.suite\_lines and \*\_temp and also, please run the update diamond xml file before.
5. default method check\_flow if the conf file is not find
6. add tag options in the scripts, the script will replace '\*tag\*' with tag vaue in the conf file
7. add check\_grep in the scripts, In the method, we will use regular expression to check

## **2. scan\_report**

2.1 Location

http://Platform/[trunk](http://linux12v/viewvc/Platform/trunk/)/[bqs\_scripts](http://linux12v/viewvc/Platform/trunk/bqs_scripts/)/[trunk](http://linux12v/viewvc/Platform/trunk/bqs_scripts/trunk/)/[tools](http://linux12v/viewvc/Platform/trunk/bqs_scripts/trunk/tools/)/scan\_report

2.2 Description

The tool is for scan all flow result. They are in the same class while with different function. Till now, the class supports the following flow:

* QoR
* General tcl
* Mutli-seed

2.3 Usage

|  |  |
| --- | --- |
| --h, --help | show this help message and exit |
| --job-dir=JOB\_DIR | specify job working path |
| --design=DESIGN | specify design name |
| --conf-file=CONF\_FILE | specify the configure file |
| --pap | dump Lattice Performance Achievement Percentage data |
| --report-name=REPORT\_NAME | specify report\_name you want to store |
| --report-path=REPORT\_PATH | specify the report path |
| --force-na | force the data become NAN if cannot get data from par file |
| --special-structure=SPECIAL\_STRUCTURE | specify the special structure for the case |
| --flow=FLOW | specify the case flow |
| --qor-auto | Write the pass/fail log in QoR automation flow |
| --list-order | sort the report as spcial order listed in the case\_list file located in conf directory |

Specification:

--job-dir: specify the case location

--design: specify which case you want to scan

--special-structure: specify the special directory structure. The default value will be “\_scratch”

Using upon three parameters, we can scan any special case result.

For QoR, if you want to get the sorted report, you have to make sure the job dir is the same to the report path.

For example:

Suppose to scan cases located in //d27407/cases, the signal case result located in case\_name/\_scratch/impl.

For scan all the case results, we can set the parameters as:

--job-dir=//d27409/cases --special-structure=\_scratch

For scan one case named case1 results, we can set the parameters as:

--job-dir=//d27409/cases --special-structure=\_scratch –design=case1

--conf-file: the file you want to specify special titles for the source you want get.

In the conf-file, you should write each title in a signal line.

For example:

IO

LUT

Fmax

…

If you used this parameters, you will have a result named source\_list\_\*.csv. In which all the sources you wanted will be listed.

--force-na: if you used this parameter, the script will translate all the data to ‘\_’ if cannot get useful data for “Par\_Done” and “all signals are complete”.

--flow: specify which flow you want to scan as different flow has different result structure.

**Demo:**

**Scan a case for QoR:**

Python run\_scan\_lattice\_step\_general\_case.py --job-dir=Sapphire\_92 --design=21\_orca\_tut –flow=qor

**Scan all case for QoR:**

Python run\_scan\_lattice\_step\_general\_case.py --job-dir=Sapphire\_92 –flow=qor

**Scan all case for QoR and write pass/fail log files and also write rerun case bat file:**

Python run\_scan\_lattice\_step\_general\_case.py --job-dir=Sapphire\_92 –flow=qor –qor-auto

**Scan all case for run mutli-seed:**

Python run\_scan\_lattice\_step\_general\_case.py --job-dir=Sapphire\_92 –flow=seed

**Scan all case for tcl result:**

Python run\_scan\_lattice\_step\_general\_case.py --job-dir=Sapphire\_92