

**SQP Command Line Interface**

**User’s Guide**

**Neeraj Mallampet**

**Table of Contents**

1. **Introduction**
   1. Purpose
   2. Location
      1. Contents
   3. Requirements
      1. Software Requirements
      2. Other Requirements
2. **Usage**
   1. Command Line Arguments and Parameters
   2. Input File
      1. Input file format
3. **Output and Generated Files**
   1. Generated Hierarchy
   2. Console Output (output to stdout)
   3. Generated Files
      1. Log Files
      2. Stimulus package output file
      3. Results File
4. **Supported EDAs** 
   1. Diamond
      1. Generating Bitgen
      2. Programmer
         1. Supported Devices
5. **Introduction**
   1. **Purpose** Systems Qualification Platform’s (SQP)Command Line Interface (CLI) is an alternative to the SQP’s GUI for user usage. In addition, the CLI can also be added to the Test Management Platform (TMP) which is very convenient for TMP’s objective of being the central hub for all automation software.
   2. **Location** The most recent version (version 1.6) of SQP CLI can be located here:  
       [\\lscfs02\LDC-svsw27\Document-Lib\fphillip\SQP\SQP\_OP\release\](file:///\\lscfs02\LDC-svsw27\Document-Lib\fphillip\SQP\SQP_OP\release\)
      1. **Contents** Extracting the most the recent version of SQP CLI should yield the following:  
          - **sqp.jar** (the program itself)   
          - **lib** (directory containing java libraries required to run sqp.jar)  
          - **res** (resource directory containing scripts required to run sqp.jar)  
          - **input.txt** (a sample input.txt to run the program, user can modify)  
          **\*\*\*The following files are including for testing purposes\*\*\*  
          - case** (directory containing a daisy chain test case)  
          - **wb** (directory containing a mult8x8 test case)
   3. **Requirements** In order to use the CLI, the user must meet the following requirements. For convenience, most of these requirements align with running the TMP client.   
      1. **Software Requirements  
           
         1) Java 1.6+   
          Recommended: Java 1.8+ (or most recent version of Java)**

SQP CLI was developed using java 8, so java is mandatory for execution. Java version 1.8 is recommended because SQP CLI uses lambda expressions which may affect execution time with versions of java 1.7 or lower.

**2) Python 2.7** A set of scripts known as the ‘trunk script’ are required to run the eda tool of choice (diamond, iCEcube, etc.). These scripts were written in python and compiled with python 2.7

**3) EDA tools**

The computer must already have the eda software to be used pre installed before running the program.   
Example: User must already have Diamond installed in order for SQP CLI to use it.

* + 1. **Other Requirements   
          
        1) Input File** SQP CLI requires an input file which contains information on testcase and other file locations needed for the CLI to function properly.  
         
        More information on the input file will be covered in sections 2.1 and 2.2  
         
       **2) EDA tools and their environment variables** The EDA tool to be used must also have its bin path location stored as an environmental variable.   
         
       Format and example as follows:   
         
         
         
         
         
         
       **3) res and lib folder** The res and lib directories **must** be present when running sqp.jar. These folders contain libraries and scripts that are necessary for CLI to execute properly.

|  |  |
| --- | --- |
| **Environment Variable Name Format** | **Environment Variable Value** |
| EXTERNAL\_[EDA NAME]\_PATH | EDA’S BIN PATH lOCATION |
| **Example:** EXTERNAL\_DIAMOND\_PATH | **Example:**C:\lscc\diamond\3.8\_x64 |

1. **Usage**
   1. **Command Line Arguments and Parameters   
        
       Usage:** java –jar sqp.jar –infile <arg> -out <arg>   
        
       Both options are **required**:  
       **-infile <arg>** This option is to pass in the input file. The input file contains the   
       suitename, testcase locations, and each of the testcase’s eda tool of choice,  
       configuration and ldf file location.   
        
       More information of the input file and its format will be covered in   
       section 2.2.  
        
       **-out <arg>** The arg of this option indicates the output directory; the location where  
       bitgen is generated and results stored  
        
       More information on output directory hierarchy will be discussed in  
       section 3.1.  
        
       **Example command usage:** java –jar sqp.jar –infile input.txt –out tmp\out  
       This will take in input.txt and store all the results in tmp\out\.  
        
       **Better usage:**  
       This java program can be incorporated into a scripting language of your choice  
       (python) and executed from there.   
         
       An example usage would be a script that sets the eda tool env variable and runs  
       the sqp jar and records the exit code.
   2. **Input File** The input file contains suite names, their testcases (1 – n testcases) and each testcase’s ldf, configuration file locations and eda tool to use.  **Note:** If a suite contains more than one testcase, it’s automatically treated as a daisy chain case. 
      1. **Input File Format** Each suite name and its contents (testcase and its information) can be considered to be a block. The block’s format is as follows:  
           
           
         **<suitename>  
         \*testcasename  
         des=[design directory containing the location of testcase]  
         conf=[configuration file location]  
         eda=[name of eda to be used]  
         ldf=[ldf file location for testcase]  
         \*testcasename2  
         des  
         ..  
         ..  
         <suitename2>**  *(start of new block/suite case)* **\*testcasename etc.  
            
         Notes:**The locations in the input file are relative to the input file’s location. For instance if the design directory is located in the same directory as the input file, the input may contain just the directory’s name and not its absolute path.   
           
         If the Conf= value is blank, (Conf= must be present however), then the default location of testcase\sim\input\_files\input.txt will be used.   
           
         Ldf file path can be relative to the design directory and testcase name. For example, the user can just input the location of the ldf file from the testcase directory instead of giving its absolute path.  **Example:**  
         Consider the included testcases **wb** and **case** in the distributed zip folder.  **wb** and **case** would be considered the design directories containing testcases.   
           
         **wb** contains testcase **mult8x8** and **case** contains testcases **board1** and **board2** The resulting input file will look like this:  
          <suite1>  
          \*board1  
          des=[Location to **case** directory]  
          conf= [Location to configuration file]  
          eda = diamond  
          ldf = [Location to ldf file]  
          \*board2  
          des=[Location to **case** directory]  
          conf= [Location to configuration file]  
          eda = diamond  
          ldf = [Location to ldf file]  
          <suite2>  
          \*mult8x8  
          des=[location to **wb** directory]  
          conf= [Location to configuration file]  
          eda = diamond  
          ldf = [Location to ldf file]  
           
         In the output directory, directories suite1 and suite2 will be created. Each of the directories will contains the results from its respective testcases. More information on this will be covered in section 3.1.

**Suite block #1 (suite1)**

**Suite block #2 (suite2)**

1. **Output and Generated Files**  
   The section describes the output directory hierarchy, console output, and generated files.   
   1. **Generated hierarchy** Suppose the output directory is out and the input file contains two suite blocks suite1 and suite2. Suite1 contains testcases test1 and test2 while Suite2 contains testcase testA.   
        
       Running SQP CLI will result in the following output directory structure:   
         
        
        
        
        
        
        
       Each testcase’s (test1, test2, testA) results will be stored in their respective directory.
   2. **Console Output** SQP CLI has 5 steps:  
       1) Generate output directory’s structure  
       2) Generate Bitgen  
       3) Run Programmer   
       4) Send stimulus package to board(s)  
       5) Generate final results file   
        
      Steps 2-4 are printed to stdout:   
        
      [START] Generating Bitstream(s) (end of step 1 and start of step 2)  
      [DONE] Generating Bitstream(s) (end of step 2 and start of step 3)  
      [START] Running Programmer   
       Running Programmer for [suitename]/[testcasename]  
       ERROR: Programmer Cable not connected  
       Running Programmer for [suitenameN]/[testcasenameN]  
      [DONE]Running Programmer (end of step 3 and start of step 4)  
      [START] Sending Configuration File(s) to Board(s)  
      [DONE] Sending Configuration File(s) to Board(s) (end of step 4)  
      1 testcase failed (start of step 5)  
        
      The print messages above can be used to determine which step the SQP CLI is on.
   3. **Generating Files**   
       The following are generated from SQP CLI regardless of EDA tool choice. Files generated depending on EDA are discussed in section 4.   
      1. **Log Files** Each testcase directory has a log directory which contains the following files:  
          1) **trunkLog.txt** :- contains the output of running trunk script  
          2) **ProgrammerOutput.txt** :- contains the output of running programmer   
          3) **ftdiLog.txt** :- contains the output of sending stimulus to board
      2. **Stimulus package output file** Each testcase directory has a sim directory. The output directory in sim will contain the stimulus package output file as output.txt.  
          The output.txt file indicates if the data read back from the board is a match or a mismatch.
      3. **Results File** The CLI concludes by generating a results.txt file in the output directory. This file contains information on all the testcases on whether it passed or where it failed. All the testcases are organized by suites.  
           
         **Example:**  
         Suitename#1  
          testcase1: Failed (Failed in Programmer)  
          testcase2: Failed (Failed in generating bitgen)  
         Suitename#2  
          testcaseA: Passed
2. **Supported EDAs**The following EDAs are compatible with SQP CLI:  
   1. **Diamond** 
      1. **Generating Bitgen**   
          For each testcase using diamond as its EDA tool, a \_scratch directory is created. The ldf file specific to this testcase will be used to generate the bitgen file. All the associated files with the generated bitgen, including the bitgen file, will be stored in the \_scratch directory.
      2. **Programmer** SQP CLI generates a xcf file after generating bitstream. If the suite contains more than one testcase, then a xcf file appropriate for daisy chain is generating, otherwise the standard xcf file is generated.   
         1. **Supported Devices**  
            The following devices are supported in generating the xcf file:  
            1) ecp5  
            2) ecp3  
            3) ecp2  
            4) ecp  
            5) macgyver  
            6) xo3  
            7) xo2  
            8) machxo  
            9) snow