

Verification Continuum™

**VC Formal**

**Quick Reference Guide**

---

Version U-2023.03, March 2023



# Copyright Notice and Proprietary Information

© 2023 Synopsys, Inc. All rights reserved. This software and documentation contain confidential and proprietary information that is the property of Synopsys, Inc. The software and documentation are furnished under a license agreement and may be used or copied only in accordance with the terms of the license agreement. No part of the software and documentation may be reproduced, transmitted, or translated, in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without prior written permission of Synopsys, Inc., or as expressly provided by the license agreement.

## Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

## Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

## Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at <http://www.synopsys.com/company/legal/trademarks-brands.html>.

All other product or company names may be trademarks of their respective owners.

## Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

## Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

[www.synopsys.com](http://www.synopsys.com)

---

## **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.

---

# Contents

VC Formal Introduction

Formal Property Verification Application

Automatically Extracted Property Application

Connectivity Check Application

Formal Coverage Analyzer Application

Formal X-Propagation Verification Application

Sequential Equivalence Checking Application

Formal Register Verification Application

Formal Testbench Analyzer Application

Functional Safety Verification Application

Formal Security Verification Application

Data Path Validation Application

## Quick Reference Guide - VC Formal











### INTRODUCTION

This Quick Reference Guide is to help you get started with VC Formal by showing you the most common flows, commands, and variables. The command options shown are only a sample of the options that are available.

For detailed information, see the user guides and reference material, available through Documentation on SolvNet:

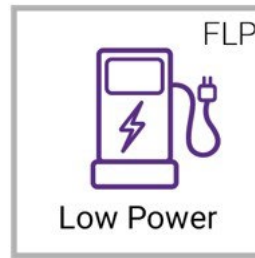
[https://spdocs.synopsys.com/dow\\_retrieve/latest/home\\_public/vc\\_formal.html](https://spdocs.synopsys.com/dow_retrieve/latest/home_public/vc_formal.html)

Use the FPV Quick Reference Guide as reference for commands and variables applicable to most VC Formal Apps.

 <p>FPV Property Verification</p>	<p>The Formal Property Verification (FPV) App verifies properties in the design including user-defined and Assertion IP (AIP) properties.</p>	 <p>FRV Register Verification</p>	<p>The Formal Register Verification (FRV) App automatically creates and formally verifies checks based on IP-XACT or RALF format for registers in the design.</p>
 <p>FCA Coverage Analyzer</p>	<p>The Formal Coverage Analyzer (FCA) App assists simulation-based verification coverage signoff and qualifies formal property verification with coverage signoff.</p>	 <p>FSV Security Verification</p>	<p>The Formal Security Verification (FSV) App ensures that unexpected data propagation does not happen between secure and non-secure areas.</p>
 <p>CC Connectivity Checking</p>	<p>The Formal Connectivity Checking (CC) App checks if there is a structural and functional connection between the source and the destination.</p>	 <p>FXP X-Propagation Verification</p>	<p>The X-Propagation Verification (FXP) App checks whether unknown signal values (X's) can propagate to dangerous points within the design.</p>
 <p>AEP Auto Checks</p>	<p>The Automatically Extracted Properties (AEP) App automatically extracts properties from the design and verifies them.</p>	 <p>DPV Datapath Validation</p>	<p>The Data Path Validation (DPV) App uses HECTOR technology to verify data transformation blocks between untimed C/C++ and RTL models.</p>
 <p>SEQ Sequential Equivalence</p>	<p>The Sequential Equivalence Checking (SEQ) App compares two RTL designs and verifies that they are functionally equivalent.</p>	 <p>FuSa Functional Safety</p>	<p>The Functional Safety (FuSa) App analyzes the controllability and observability of Z01X faults to calculate FMEDA (Failure Mode Effects and Diagnostic Analysis) metrics.</p>



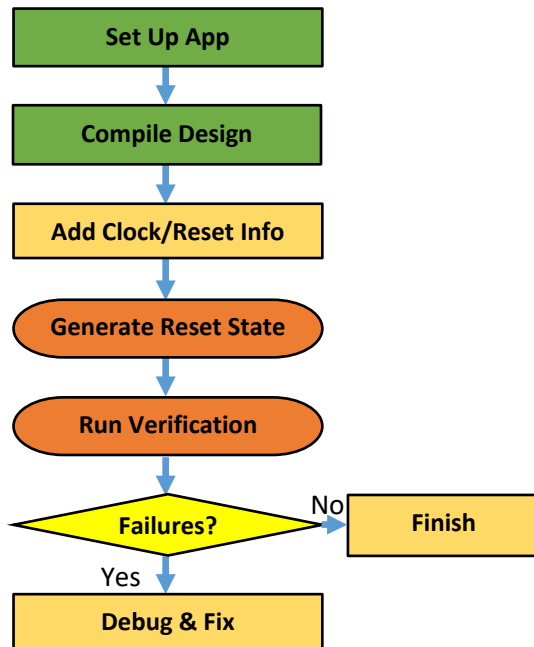
The Formal Testbench Analyzer (FTA) App checks if injected faults can be detected by the formal testbench thus measuring the quality of the formal testbench.



The Formal Low Power (FLP) App checks if the design with power intent works as per the functional specification of the design. You can use FLP along with the CC and SEQ Apps for verification.

## Quick Reference Guide - VC Formal - Formal Property Verification App

### FLOW



### DESIGN SETUP & INITIALIZATION

#### ##### SET VC FORMAL APP MODE

Set App mode    set\_fml\_appmode **FPV**

#### ##### ADD BLACKBOXES

Modules        set\_blackbox -designs *module*

Instances      set\_blackbox -cells *hier\_instance*

List blackboxes    report\_blackbox

#### ##### LOAD ASSERTION IPS

Load AIPs        aip\_load -protocol "*AHB AXIS*"

#### ##### COMPILE DESIGN

Read files        read\_file -top *top* -sva -format verilog \

-vcs {-f *filelist*}

Analyze & elaborate

analyze -format verilog -vcs {-f *filelist*}

elaborate *top* -sva

#### ##### CLOCKS, RESETS, AND CONSTANTS

Create clocks     create\_clock *clk* -period *100*

Create resets     create\_reset *rst* -sense high

Add constants    set\_constant *testmode* -value *1'b0*

Set input transition    set\_change\_at -clock *clk* -default

#### ##### INFER FORMAL SETUP

Infer clocks       infer\_formal\_setup -type clock

Configure        formal\_setup\_config

Report results    report\_formal\_setup

#### ##### INITIALIZE DESIGN BY SIMULATION

To stable state    sim\_run -stable

Or force value    sim\_force *signal* -apply *3'b000*

sim\_run *3* -clk *clk*

Override state    sim\_set\_state -uninitialized -user\_only \

-apply *0*

Save initial state    sim\_save\_reset

View waveform    view\_trace -reset

Get initial state    sim\_get -signals {*control\_reg\**}

#### ##### DESIGN COMPLEXITY QUERY

Report design complexity    report\_fv\_complexity

#### ##### CHECK DESIGN SETUP

Check setup        check\_fv\_setup

Configure check    fv\_setup\_config

Report results    report\_fv\_setup -list

### INPUT FILES FORMAT

RTL (Verilog, SystemVerilog, VHDL) file(s)

SystemVerilog Assertion (SVA) file(s)

Tool Command Language (Tcl) file(s)

### TOOL COMMAND HELP

VF Formal launch help    %vcf -help

General help groups      vcf> help Formal

List matching commands    vcf> help *report\**

Help for command vcf> report\_fv -help

Man page for command    vcf> man report\_fv

vcf> view\_help report\_fv

List matching fml vars    vcf> report\_fml\_var *fml\_max\_\**

List matching app vars    vcf> report\_app\_var *\*fml\**

### RUNNING VC FORMAL

Interactive with Verdi    %vcf -f *run.tcl* -verdi

Interactive without Verdi    %vcf -f *run.tcl*

Batch/regression mode    %vcf -f *run.tcl* -batch

Using Elite license configuration

%vcf -f *run.tcl* -verdi -fml\_elite

## VERIFICATION & DEBUG

### ##### MANAGE PROPERTIES

Create property   fvassume <name> -expr {<expression>}  
                  fvassert <name> -expr {<expression>}  
                  fvcover <name> -expr {<expression>}

Disable property               fvdisable *property*

Enable property                fvenable *property*

Change property to assertion   fvassert *property*

Change property to constraint   fvassume *property*

Change property to cover       fvcover *property*

### ##### CONFIGURE COMPUTE RESOURCES

Set max run time   set\_fml\_var fml\_max\_time *24H*

Set max memory   set\_fml\_var fml\_max\_memory *32GB*

Set engine progress timeout  
                  set\_fml\_var fml\_progress\_time\_limit *2H*

Set grid settings   set\_grid\_usage -type RSH=*12*  
                      set\_grid\_usage -file *hostfile*

### ##### ENABLE/DISABLE VACUITY CHECK

Check vacuity   set\_fml\_var fml\_vacuity\_on true

### ##### ENABLE/DISABLE WITNESS TRACE GENERATION

Find witness   set\_fml\_var fml\_witness\_on false

### ##### ENABLE REGRESSION MODE ACCELERATOR (RMA)

Enable RMA   fvlearn\_config -local\_dir *rma\_db*

### ##### RUN VERIFICATION

Run           check\_fv

Stop run       check\_fv -stop

Blocking mode   check\_fv -block

Callback task   check\_fv -run\_finish {*report\_fv -list*}

### ##### REPORT VERIFICATION RESULTS

Report results   report\_fv [-list] [-verbose]

### ##### SAVE/RESTORE SESSION

Save session   save\_session -session *session*

Restore session   restore\_session -session *session*

Start from stored session   %vcf -restore -session *session*

### ##### MANAGE TRACES FOR DEBUGGING

Open trace   view\_trace -property *property*

Save trace   fvtrace -property *property*

### ##### GENERATE VERIFICATION SUMMARY

Compute summary   compute\_verification\_summary

Configure tags   set\_verification\_summary

Waive/unwaive tags   waive\_verification\_summary

Report results summary   report\_verification\_summary

## PERFORMANCE & CONVERGENCE

### ##### SET RUN EFFORT LEVEL

Standard proof recipe   set\_fml\_var fml\_effort default

Heavy proof recipe   set\_fml\_var fml\_effort high

Falsification/covers   set\_fml\_var fml\_effort bug\_hunting

Target hard property   set\_fml\_var fml\_effort discovery

### ##### MONITOR PROGRESS QUERY

Engine status   report\_fml\_engines

Grid job status   report\_fml\_jobs

Grid hosts status   report\_fml\_hosts

### ##### ADD ABSTRACTIONS

Create cutpoints   snip\_driver *net*

Abstract design constructs  
                  set\_abstractions -construct {mult=*16* \  
  count=*4*}

Report abstracted constructs  
                  report\_abstractions

### ##### IDENTIFY REDUCED CONSTRAINTS FOR PROOFS

Compute reduced constraints  
                  compute\_reduced\_constraints \  
                                  -property *property*

Report results   report\_reduced\_constraints

Get results (Tcl)   get\_reduced\_constraints

### ##### IDENTIFY FORMAL CORE FOR PROOFS

Compute Formal Core  
                  compute\_formal\_core -property *property*

Report results   report\_formal\_core

Get results (Tcl)   get\_formal\_core

### ##### GET GUIDANCE FOR PROOF CONVERGENCE

Compute Proof Assist information  
                  compute\_proof\_assist -property *property*

Report results   report\_proof\_assist

### ##### MANAGE TASKS

Create task   fvtask -create *task*

Copy task   fvtask *new\_task* -copy *task*

Copy elements between tasks  
                  fvtask *new\_task* -copy *task* \  
                          [-assumes <list>] [-asserts <list>] \  
                          [-covers <list>] [-constants <list>] \  
                          [-snips <list>] [-changeats<list>] \  
                          [-keep\_status] [-keep\_task]

Set active task   fvtask *task*

Get active task   get\_fvtask

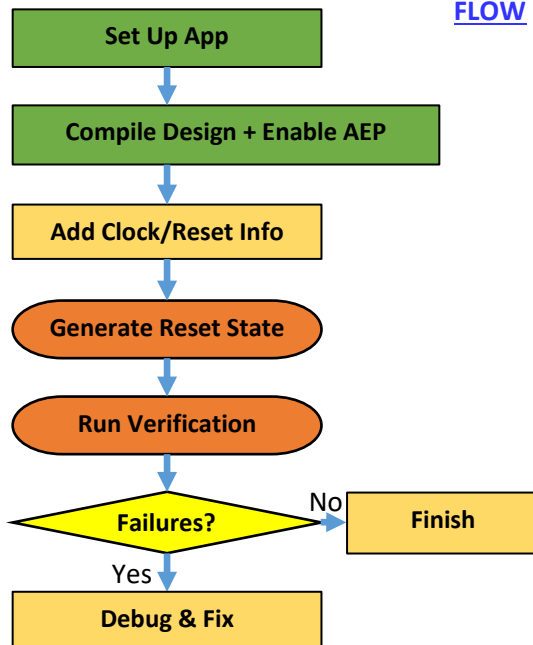
### ##### EXIT VC FORMAL

Exit VC Formal   quit



## Quick Reference Guide - VC Formal – Automatically Extracted Property App

### FLOW



### DESIGN SETUP & INITIALIZATION

#### ##### VC FORMAL APP MODE

**Set App mode** set\_fml\_appmode AEP

#### ##### Enable AEP for VHDL designs

**Set App var** set\_app\_var fml\_enable\_vhdl\_aep true

#### ##### Enable AEP for VHDL Bus checks

**Set App var** set\_app\_var fml\_enable\_vhdl\_bus\_check true

#### ##### Enable AEP for VHDL FSM check

**Set App var** set\_app\_var fml\_enable\_vhdl\_cov true

#### ##### Enable AEP for Unique Name setting

**Set Fml var** set\_fml\_var fml\_aep\_unique\_name true

#### ##### COMPILE DESIGN

**Read files <Normal>** read\_file -top *my\_top* -aep *AEP\_type* \  
-format *RTL\_format* -vcs {-f *my\_filelist*}

**Read files <Special type, like fsm\_sss>**

read\_file -top *my\_top* -aep fsm\_sss -cov fsm\_state+fsm\_trans \  
-format *RTL\_format* -vcs {-f *my\_filelist*}

**Read files <Special type, like fsm\_deadlock, fsm\_livelock>**

read\_file -top *my\_top* -aep fsm\_deadlock+fsm\_livelock \  
-format *RTL\_format* -vcs {-f *my\_filelist*}

#### Analyze & elaborate

analyze -format *RTL\_format* -vcs {-f *my\_filelist*}

elaborate *my\_top* -aep *AEP\_type*

#### ##### CLOCKS, RESETS AND CONSTANTS <Same as FPV >

**Create clocks** create\_clock *my\_clk* -period 100

**Create resets** create\_reset *my\_rst* -sense high

**Add constants** set\_constant *testmode* -value 1'b0

#### ##### INITIALIZE DESIGN BY SIMULATION <Same as FPV>

**To stable state** sim\_run -stable

**Or force value** sim\_force *sig\_name* -apply 3'b000  
sim\_run 3 -clk *my\_clk*

**Override state** sim\_set\_state -uninitialized -user\_only -apply 0

**Save initial state** sim\_save\_reset

**View waveform** view\_trace -reset

**Get initial state** sim\_get -signals {*control\_reg\**}

#### ##### DESIGN SETUP QUERY <Same as FPV>

**Check setup** check\_fv\_setup

**Configure check** fv\_check\_config

### INPUT FILES FORMAT

RTL Format (Verilog, SystemVerilog, VHDL) file(s)  
RTL\_Format keyword < verilog | sverilog | vhdl >  
Tool Command Language (Tcl) file(s)

### TOOL COMMAND HELP

**VF Formal launch help** %vcf -help  
**General help groups** vcf> help vcf> help Formal\_AEP  
**List matching commands** vcf> help *report\**  
**Help for command** vcf> report\_fv -help  
**Man page for command** vcf> man report\_fv  
vcf> view\_help report\_fv  
**List matching fml vars** vcf> report\_fml\_var *\*max\**  
**List matching app vars** vcf> report\_app\_var *\*fml\**

### RUNNING VC FORMAL

**Interactive with Verdi** %vcf -f *my\_run.tcl* -verdi  
**Interactive without Verdi** %vcf -f *my\_run.tcl*  
**Batch/regression mode** %vcf -f *my\_run.tcl* -batch  
**Using Elite license configuration**  
%vcf -f *run.tcl* -verdi -fml\_elite

#### ##### Setup FSM fairness constraint

**Add fairness** aep\_generate -type fsm\_fairness  
**Dump to a file** aep\_generate -type fsm\_fairness -tcl fairness.tcl

#### ##### Disable fsm\_deadlock, fsm\_livelock or fsm\_unionlock property generation for some state

**Disable state** aep\_config -dont\_generate -type fsm\_livelock {state\_reg::state\_name}

### VERIFICATION & DEBUG

#### ##### MANAGE PROPERTIES

**Disable property** fvdisable *prop\_name*  
**Enable property** fvenable *prop\_name*  
**Waive property** fvwaive *prop\_name*  
**Unwaive property** fvunwaive *prop\_name*

#### ##### CONFIGURE COMPUTE RESOURCES <Same as FPV>

**Set max run time** set\_fml\_var fml\_max\_time *24H*  
**Set max memory** set\_fml\_var fml\_max\_memory *32GB*  
**Set engine progress timeout** set\_fml\_var fml\_progress\_time\_limit *2H*  
**Set grid settings** set\_grid\_usage -type RSH=*10*

#### ##### RUN VERIFICATION <Same as FPV>

**Run** check\_fv  
**Stop run** check\_fv -stop  
**Blocking mode** check\_fv -block  
**Callback task** check\_fv -run\_finish {*report\_fv -list*}

#### ##### REPORT VERIFICATION RESULTS <Same as FPV>

**Report results** report\_fv  
**Report as list** report\_fv -list  
**Report details** report\_fv -verbose  
**Save Waiver Results** save\_waiver\_file -file aep.el  
**Reuse Waiver Results** read\_waiver\_file -elfiles aep.el

#### ##### SAVE/RESTORE SESSION <Same as FPV>

**Save session** save\_session -session *my\_session*  
**Restore session** restore\_session -session *my\_session*  
**Start from stored session** %vcf -restore -session *my\_session*

#### ##### VIEW TRACES FOR DEBUGGING

**Open falsified trace** view\_trace -property *prop\_name*

### PERFORMANCE & CONVERGENCE

#### ##### MONITOR PROGRESS QUERY <Same as FPV>

**Engine status** report\_fml\_engines  
**Grid job status** report\_fml\_jobs  
**Grid hosts status** report\_fml\_hosts

#### ##### MANAGE TASKS <Same as FPV>

**Create task** fvtask -create *my\_task*  
**Copy task** fvtask *my\_new\_task* -copy *my\_task*  
**Copy elements between tasks**  
fvtask *my\_new\_task* -copy *my\_task* \  
[-assumes <*list*>] [-asserts <*list*>] \  
[-covers <*list*>] [-constants <*list*>] \  
[-snips <*list*>] [-changeats<*list*>]

**Set active task** fvtask *my\_task*

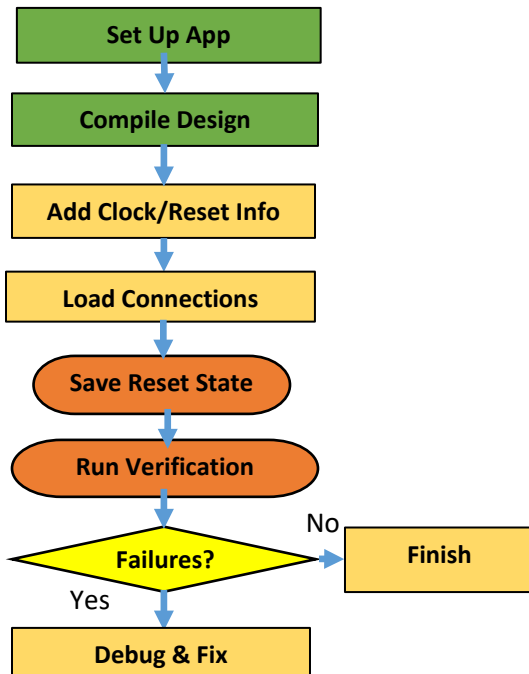
**Get active task** get\_fvtask

#### ##### EXIT VC FORMAL <Same as FPV>

**Exit VC Formal** quit

## Quick Reference Guide - VC Formal – Connectivity Check App

### FLOW



### DESIGN SETUP & INITIALIZATION

#### ##### VC FORMAL APP MODE

**Set App mode** set\_fml\_apemode CC

#### ##### BLACKBOX SETUP

**Modules** set\_blackbox -designs *module*

**Instances** set\_blackbox -cells *hier\_instance*

**List blackboxes** report\_blackbox

**Disable auto-blackboxing**

set\_fml\_var fml\_cc\_autobbox false

#### ##### COMPILE DESIGN

**Read files** read\_file -top *top* -format verilog \  
-vcs {-f *filelist*}

**Analyze & elaborate**

analyze -format verilog -vcs {-f *filelist*}

elaborate *top*

#### ##### CLOCKS, RESETS AND CONSTANTS <Same as FPV>

**Create clocks** create\_clock *clk* -period *100*

**Create resets** create\_reset *rst* -sense high

**Add constants** set\_constant *testmode* -value *1'b0*

#### ##### SPECIFY CONNECTIONS

**TCL Format** add\_cc -src *signal* -dest *signal*

**CSV Format** load\_cc\_set\_param *param\_name* "%<value>%"

load\_cc -format csv *csv file*

#### ##### SAVE INITIAL STATE

**Save initial state** sim\_save\_reset

#### ##### DESIGN SETUP QUERY <Same as FPV>

**Check setup** check\_fv\_setup

**Configure check** fv\_setup\_config

**Report results** report\_fv\_setup -list

#### ##### DESIGN COMPLEXITY QUERY <Same as FPV>

**Get design data** report\_fv\_complexity

#### ##### CONNECTIVITY EXTRACTION

**Extraction of connections**

generate\_cc -src *signal* -dest *signal* -run

### INPUT FILES FORMAT

RTL Format (Verilog, SystemVerilog, VHDL) file(s)

RTL\_Format keyword < verilog | sverilog | vhd1 >

Tool Command Language (Tcl) file(s)

Connections definition TCL or CSV

### TOOL COMMAND HELP

**VF Formal launch help** %vcf -help

**General help groups** vcf> help vcf> help Formal\_CC

**List matching commands** vcf> help *report\**

**Help for command** vcf> report\_fv -help

**Man page for command** vcf> man report\_fv

vcf> view\_help report\_fv

**List matching fml vars** vcf> report\_fml\_var *\*max\**

**List matching app vars** vcf> report\_app\_var *\*fml\**

### RUNNING VC FORMAL

**Interactive with Verdi** %vcf -f *my\_run.tcl* -verdi

**Interactive without Verdi** %vcf -f *my\_run.tcl*

**Batch/regression mode** %vcf -f *my\_run.tcl* -batch

**Using Elite license configuration**

%vcf -f *run.tcl* -verdi -fml\_elite

## VERIFICATION & DEBUG

### ##### MANAGE PROPERTIES

Disable property                      fvdisable *prop\_name*

Enable property                        fvenable *prop\_name*

### ##### CONFIGURE COMPUTE RESOURCES <Same as FPV>

Set max run time set\_fml\_var fml\_max\_time *24H*

Set max memory set\_fml\_var fml\_max\_memory *32GB*

Set engine progress timeout

set\_fml\_var fml\_progress\_time\_limit *2H*

Set grid settings set\_grid\_usage -type RSH=*10*

set\_grid\_usage -file *hostfile*

### ##### ENABLE VACUITY CHECK <Same as FPV>

Check vacuity set\_fml\_var fml\_vacuity\_on true

### ##### ENABLE REGRESSION MODE ACCELERATOR (RMA) <Same as FPV>

Enable RMA fvlern\_config -local\_dir *rma\_db*

### ##### ENABLE REVERSE CONNECTION

Reverse connection

set\_fml\_var fml\_allow\_reverse\_cc\_path true

### ##### RUN VERIFICATION <Same as FPV>

Run check\_fv

Stop run check\_fv -stop

Blocking mode check\_fv -block

Callback task check\_fv -run\_finish {*report\_fv-list*}

### ##### REPORT VERIFICATION RESULTS

Report load\_cc connection report\_load\_cc

Report results report\_fv

Report as list report\_fv -list

Report details report\_fv -verbose

Report in CC format report\_fv -formatCC csv|tcl|path

### ##### SAVE/RESTORE SESSION <Same as FPV>

Save session save\_session -session *session*

Restore session restore\_session -session *session*

Start from stored session %vcf -restore -session *session*

### ##### VIEW TRACES FOR DEBUGGING

Open falsified trace view\_trace -property *prop\_name*

### ##### VIEW SCHEMATIC FOR DEBUGGING

Open schematic view\_schematic -prop *prop\_name*

### ##### DEBUG SPECIFIC PATH

List specific path list\_path -src *signal* -dest *signal*

List property path list\_path -cc\_prop *prop\_name*

### ##### CONNECTIVITY CHECK COVERAGE

Enable CC coverage

set\_app\_var fml\_cc\_coverage\_analyzer true

Enable toggle coverage monitoring on input port

set\_app\_var fml\_cov\_tgl\_input\_port true

Enable toggle coverage at read\_file/analyze

read\_file -top *top* -cov *tgl*

Run CC coverage

compute\_cc\_cov

Save CC coverage

save\_cc\_cov\_results

## PERFORMANCE & CONVERGENCE

### ##### MONITOR PROGRESS QUERY <Same as FPV>

Engine status report\_fml\_engines

Grid job status report\_fml\_jobs

Grid hosts status report\_fml\_hosts

### ##### MANAGE TASKS <Same as FPV>

Create task fvtask -create *task*

Copy task fvtask *new\_task* -copy *task*

Copy elements between tasks

fvtask *new\_task* -copy *task* \  
[-assumes <*list*>] [-asserts <*list*>] \  
[-covers <*list*>] [-constants <*list*>] \  
[-snips <*list*>] [-changeats<*list*>] \  
[-keep\_status] [-keep\_task]

Set active task fvtask *task*

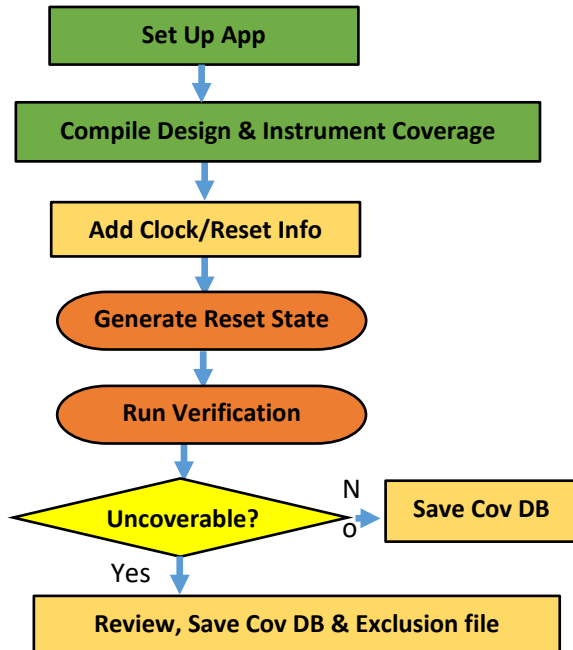
Get active task get\_fvtask

### ##### EXIT VC FORMAL <Same as FPV>

Exit VC Formal quit

## Quick Reference Guide - VC Formal - Formal Coverage Analyzer App

### FLOW



### DESIGN SETUP & INITIALIZATION

#### ##### VC FORMAL APP MODE

**Set App mode** set\_fml\_appmode **FCA**

#### ##### CONFIGURE COVERAGE

**Enable coverage for VHDL designs**

set\_app\_var fml\_enable\_vhdl\_cov true

**Enable the reset reachable check**

set\_fml\_var fml\_reset\_property\_check true

**Enable branch coverage, if required**

set\_fml\_var fml\_cov\_enable\_branch\_cov true

**Enable toggle coverage for input ports, if required**

set\_app\_var fml\_cov\_tgl\_input\_port true

**Extract of coverage options used in simulation**

set\_app\_var fml\_cov\_override\_db\_opt false

#### ##### MANAGE COVERAGE DB

**Read simulation coverage DB**

read\_covdb -cov\_input *simv.vdb* -cov\_dut  
*Testbench.Dut\_instance* [-elfile *elfile*]  
[-check\_el]

**Save coverage DB**

save\_covdb -status covered -name *cov.db*

**View coverage DB**

view\_coverage -cov\_input *cov.db* -elfiles *elfiles*

#### ##### BLACKBOX SETUP <Same as FPV>

**Modules** set\_blackbox -designs *module*

**Instances** set\_blackbox -cells *hier\_instance*

**List blackboxes** report\_blackbox

#### ##### COMPILE DESIGN

**Read files** read\_file -top *my\_top* [-sva] -cov  
*Coverage\_Metric* -format *RTL\_format* -vcs {-f *my\_filelist*}

**Analyze & elaborate**

analyze -format *RTL\_format* -vcs {-f *my\_filelist*}  
elaborate *my\_top* [-sva] -cov *Coverage\_Metric*

**with Coverage\_Metric a combination of**  
line+cond+tgl+fsm\_state+fsm\_transition+branch+cg

#### ##### CLOCKS, RESETS AND CONSTANTS <Same as FPV>

**Create clocks** create\_clock *clk* -period *100*

**Create resets** create\_reset *rst* -sense high

**Add constants** set\_constant *testmode* -value *1'b0*

### INPUT FILES FORMAT <Same as FPV>

RTL (Verilog, SystemVerilog, VHDL) file(s)  
SystemVerilog Assertion (SVA) file(s)  
Tool Command Language (Tcl) file(s)

### TOOL COMMAND HELP

**VF Formal launch help** %vcf -help  
**General help groups** vcf> help *Formal\_FCA*  
**List matching commands** vcf> help *report\**  
**Help for command** vcf> report\_fv -help  
**Man page for command** vcf> man report\_fv  
vcf> view\_help report\_fv  
**List matching fml vars** vcf> report\_fml\_var *fml\_max\_\**  
**List matching app vars** vcf> report\_app\_var *\*fml\**

### RUNNING VC FORMAL <Same as FPV>

**Interactive with Verdi** %vcf -f *run.tcl* -verdi  
**Interactive without Verdi** %vcf -f *run.tcl*  
**Batch/regression mode** %vcf -f *run.tcl* -batch  
**Using Elite license configuration**  
%vcf -f *run.tcl* -verdi -fml\_elite

## ##### INITIALIZE DESIGN BY SIMULATION

To stable state `sim_run -stable`

Save initial state `sim_save_reset`

## ##### DESIGN SETUP QUERY <Same as FPV>

Configure check `fv_check_config`

Check setup `check_fv_setup`

Report results `report_fv_setup -list`

## VERIFICATION & DEBUG

## ##### MANAGE PROPERTIES

Disable property `fvdisable prop_name`

Disable property by type `fvdisable -type toggle`

Enable property `fvenable prop_name`

## ##### CONFIGURE COMPUTE RESOURCES

Set max run time `set_fml_var fml_max_time 24H`

Set max memory `set_fml_var fml_max_memory 16GB`

Set engine progress timeout  
`set_fml_var fml_progress_time_limit 2H`

Set grid settings `set_grid_usage -type LSF=100 -control {bsub -q ...}`

## ##### ENABLE REGRESSION MODE ACCELERATOR (RMA) <Same as FPV>

Enable RMA `fvlearn_config -local_dir rma_db`

## ##### REPORT VERIFICATION RESULTS

Report results `report_fv`

Report as list `report_fv -list > list_report.txt`

Report details `report_fv -verbose > verbose_report.txt`

## ##### SAVE EXCLUSIONS IN A FILE

Save exclusion file `save_cov_exclusion -file exclusion.el`

## ##### SAVE/RESTORE SESSION <Same as FPV>

Save session `save_session -session session`

Restore session `restore_session -session session`

Start from stored session `%vcf -restore -session session`

## ##### Enable Trace for Covered goals

Set App var `set_fml_var fml_cov_gen_trace on`

## ##### VIEW TRACES FOR DEBUGGING

Open covered trace `view_trace -property prop_name`

## PERFORMANCE & CONVERGENCE

## ##### SET RUN EFFORT LEVEL <Same as FPV>

Standard proof recipe `set_fml_var fml_effort default`

Heavy proof recipe `set_fml_var fml_effort high`

To get covered quickly `set_fml_var fml_effort bug_hunting`

Target hard property `set_fml_var fml_effort discovery`

## ##### MONITOR PROGRESS QUERY

Engine status `report_fml_engines -engineStats`

Grid job status `report_fml_jobs -list > jobs_list.txt`

Grid hosts status `report_fml_hosts > workers_list.txt`

## ##### MANAGE TASKS <Same as FPV>

Create task `fvtask -create task`

Copy task `fvtask new_task -copy task`

Copy elements between tasks

`fvtask new_task -copy task \`  
`[-assumes <list>] [-asserts <list>] \`  
`[-covers <list>] [-constants <list>] \`  
`[-snips <list>] [-changeats<list>] \`  
`[-keep_status] [-keep_task]`

Set active task `fvtask task`

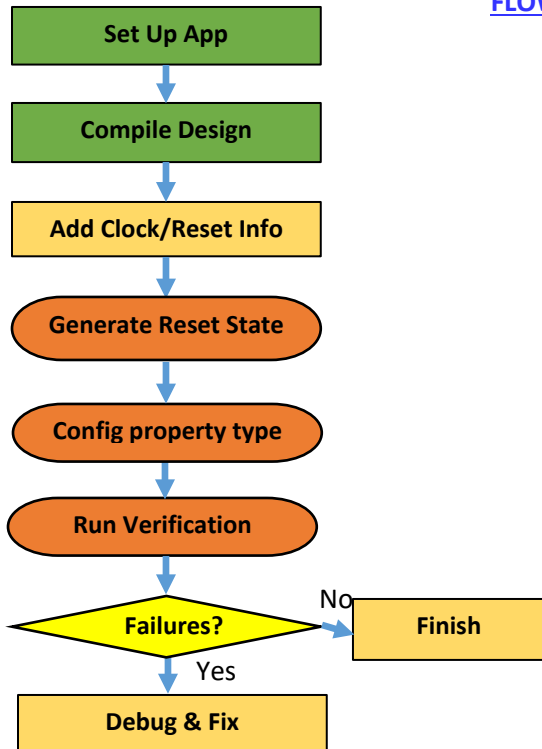
Get active task `get_fvtask`

## ##### EXIT VC FORMAL <Same as FPV>

Exit VC Formal `quit`

## Quick Reference Guide - VC Formal - Formal X-Propagation Verification App

### FLOW



### INPUT FILES FORMAT

RTL Format (Verilog, SystemVerilog, VHDL) file(s)  
 RTL\_Format keyword < verilog | sverilog | vhd | >  
 Tool Command Language (Tcl) file(s)

### TOOL COMMAND HELP

**VF Formal launch help**      %vcf -help  
**General help groups**      vcf> help    vcf> help Formal\_FXP  
**List matching commands**    vcf> help *report\**  
**Help for command** vcf> report\_fv -help  
**Man page for command**    vcf> man report\_fv  
                                  vcf> view\_help report\_fv  
**List matching fml vars**    vcf> report\_fml\_var *\*max\**  
**List matching app vars**    vcf> report\_app\_var *\*fml\**

### RUNNING VC FORMAL

**Interactive with Verdi**      %vcf -f *my\_run.tcl* -verdi  
**Interactive without Verdi**    %vcf -f *my\_run.tcl*  
**Batch/regression mode**      %vcf -f *my\_run.tcl* -batch  
**Using Elite license configuration**  
                                  %vcf -f *run.tcl* -verdi -fml\_elite

### DESIGN SETUP & INITIALIZATION

#### ##### VC FORMAL APP MODE

**Set App mode**      set\_fml\_appmode FXP

#### ##### BLACKBOX SETUP <Same as FPV >

**Modules**            set\_blackbox -design *my\_module\_name*  
**Instances**          set\_blackbox -cells *my\_hier\_instance\_name*  
**List blackboxes**    report\_blackbox

#### ##### COMPILE DESIGN <Same as FPV >

##### Read files

read\_file -top *my\_top* -sva -format sverilog -vcs {-f *my\_filelist*}

##### Analyze & elaborate

analyze -format sverilog -vcs {-f *my\_filelist*}

elaborate -sva *my\_top*

#### ##### CLOCKS, RESETS AND CONSTANTS <Same as FPV >

**Create clocks**      create\_clock *my\_clk* -period 100

**Create resets**      create\_reset *my\_rst* -sense high

**Add constants**    set\_constant *testmode* -value 1'b0

#### ##### INITIALIZE DESIGN BY SIMULATION <Same as FPV >

**To stable state**    sim\_run -stable

**Or force value**    sim\_force *sig\_name* -apply 3'b000  
                          sim\_run 3 -clk *my\_clk*

**Override state**    sim\_set\_state -uninitialized -user\_only -apply 0

**Save initial state** sim\_save\_reset

**View waveform**    view\_trace -reset

**Get initial state**   sim\_get -signals {*control\_reg\**}

#### ##### DESIGN SETUP QUERY <Same as FPV >

**Check setup**        check\_fv\_setup

**Configure check**   fv\_check\_config

**Report results**    report\_fv\_setup -list

#### ##### CONFIGURE FXP

**Create observe points** fxp\_generate <*name*> <*type*>

**Custom X injection**    fxp\_assume -injectx <*options\**>

**Custom X remove**      fxp\_assume -nox <*options\**>

**Custom X observe**      fxp\_assert <*options\**>



## VERIFICATION & DEBUG

### ##### MANAGE PROPERTIES

**Disable property**      fvdisable *prop\_name*

**Enable property**      fvenable *prop\_name*

### ##### CONFIGURE COMPUTE RESOURCES <Same as FPV >

**Set max run time** set\_fml\_var fml\_max\_time *24H*

**Set max memory** set\_fml\_var fml\_max\_memory *32GB*

**Set engine progress timeout**  
set\_fml\_var fml\_progress\_time\_limit *2H*

**Set grid settings** set\_grid\_usage -type RSH=*10*

### ##### RUN VERIFICATION <Same as FPV >

**Run**      check\_fv

**Stop run**      check\_fv -stop

**Blocking mode** check\_fv -block

**Callback task** check\_fv -run\_finish {*report\_fv -list*}

### ##### REPORT VERIFICATION RESULTS <Same as FPV >

**Report results**      report\_fv

**Report as list**      report\_fv -list

**Report details**      report\_fv -verbose

### ##### DEBUG METHODOLOGY

**Compute X Root Cause** fxp\_compute\_rootcause *prop\_name*

**Report X Root Cause**      fxp\_report\_rootcause

### ##### SAVE/RESTORE SESSION <Same as FPV >

**Save session**      save\_session -session *my\_session*

**Restore session** restore\_session -session *my\_session*

**Start from stored session** %vcf -restore -session *my\_session*

### ##### VIEW TRACES FOR DEBUGGING

**Open falsified trace**      view\_trace -property *prop\_name*

## PERFORMANCE & CONVERGENCE

### ##### MONITOR PROGRESS QUERY <Same as FPV >

**Engine status**      report\_fml\_engines

**Grid job status**      report\_fml\_jobs

**Grid hosts status** report\_fml\_hosts

### ##### MANAGE TASKS < Same as FPV >

**Create task**      fvtask -create *my\_task*

**Copy task**      fvtask *my\_new\_task* -copy *my\_task*

**Copy elements between tasks**

fvtask *my\_new\_task* -copy *my\_task* \  
[-assumes <*list*>] [-asserts <*list*>] \  
[-covers <*list*>] [-constants <*list*>] \  
[-snips <*list*>] [-changeats<*list*>]

**Set active task**      fvtask *my\_task*

**Get active task**      get\_fvtask

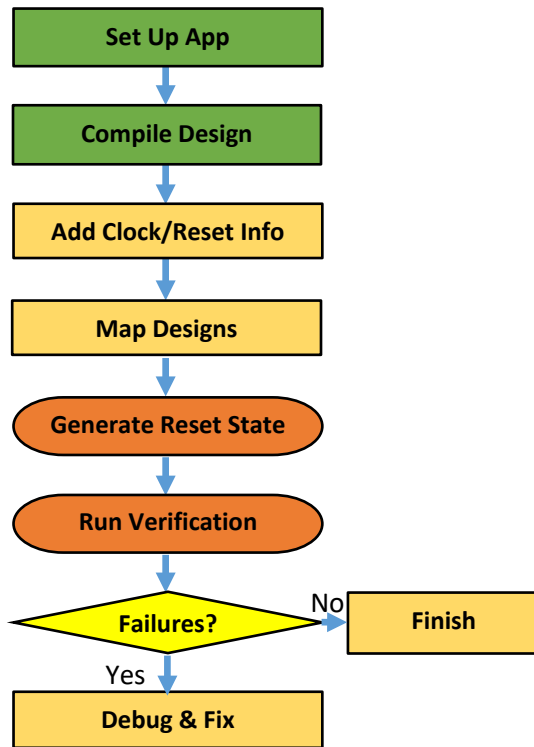
### ##### EXIT VC FORMAL < Same as FPV >

**Exit VC Formal**      quit



## Quick Reference Guide - VC Formal - Sequential Equivalence Checking App

### FLOW



### DESIGN SETUP & INITIALIZATION

#### ##### VC FORMAL APP MODE

**Set App mode** set\_fml\_appmode **SEQ**

#### ##### CONFIGURE SEQ APP

**Proof recipe** seq\_config -recipe orch\_generic\_medium

**Mapping** seq\_config -map\_uninit -map\_x zero

#### ##### BLACKBOX SETUP <Same as FPV>

**Modules** set\_blackbox -designs *module*

**Instances** set\_blackbox -cells *hier\_instance*

**List blackboxes** report\_blackbox

#### ##### COMPILE DESIGNS

##### Analyze & elaborate

analyze -format verilog -library *spec* -vcs {-f *sfilelist*}

analyze -format verilog -library *impl* -vcs {-f *ifilelist*}

elaborate\_seq -spectop *spec* -impltop *impl*

##### Or analyze & elaborate same design

analyze -format verilog -vcs {-f *filelist*}

elaborate\_seq -spectop *top* -same\_design

#### ##### CLOCKS, RESETS AND CONSTANTS <Same as FPV>

**Create clocks** create\_clock *clk* -period *100*

**Create resets** create\_reset *rst* -sense high

**Add constants** set\_constant *testmode* -value *1'b0*

#### ##### INITIALIZE DESIGN BY SIMULATION <Same as FPV>

**To stable state** sim\_run -stable

**Or force value** sim\_force *signal* -apply *3'b000*  
sim\_run *3* -clk *clk*

**Override state** sim\_set\_state -uninitialized -user\_only \  
-apply *0*

**Save initial state** sim\_save\_reset

**View waveform** view\_trace -reset

**Get initial state** sim\_get -signals {*control\_reg\**}

#### ##### MAP DESIGNS

**Map ports** map\_by\_name

**Map registers** seqmap

**Unmap registers** sequnmap

**Map by file** seq\_read\_mapping\_file *mappings.tcl*

**Report mappings** report\_seq\_mappings

### INPUT FILES FORMAT <Same as FPV>

RTL (Verilog, SystemVerilog, VHDL) file(s)

SystemVerilog Assertion (SVA) file(s)

Tool Command Language (Tcl) file(s)

### TOOL COMMAND HELP <Same as FPV>

**VF Formal launch help** %vcf -help

**General help groups** vcf> help

**List matching commands** vcf> help *report\**

**Help for command** vcf> report\_fv -help

**Man page for command** vcf> man report\_fv

**Using Elite license configuration**

%vcf -f *run.tcl* -verdi -fml\_elite

**List matching fml vars** vcf> report\_fml\_var *fml\_max\_\**

**List matching app vars** vcf> report\_app\_var *\*fml\**

### RUNNING VC FORMAL <Same as FPV>

**Interactive with Verdi** %vcf -f *run.tcl* -verdi

**Interactive without Verdi** %vcf -f *run.tcl*

**Batch/regression mode** %vcf -f *run.tcl* -batch

**Using Elite license configuration**

%vcf -f *run.tcl* -verdi -fml\_elite



## PERFORMANCE & CONVERGENCE (Continued)

### ##### IDENTIFY FORMAL CORE FOR PROOFS <Same as FPV>

#### Compute Formal Core

compute\_formal\_core -property *property*

Report results report\_formal\_core

Get results (Tcl) get\_formal\_core

### ##### MANAGE TASKS <Same as FPV>

Create task fvtask -create *task*

Copy task fvtask *new\_task* -copy *task*

#### Copy elements between tasks

fvtask *new\_task* -copy *task* \  
[-assumes <*list*>] [-asserts <*list*>] \  
[-covers <*list*>] [-constants <*list*>] \  
[-snips <*list*>] [-changeats<*list*>]

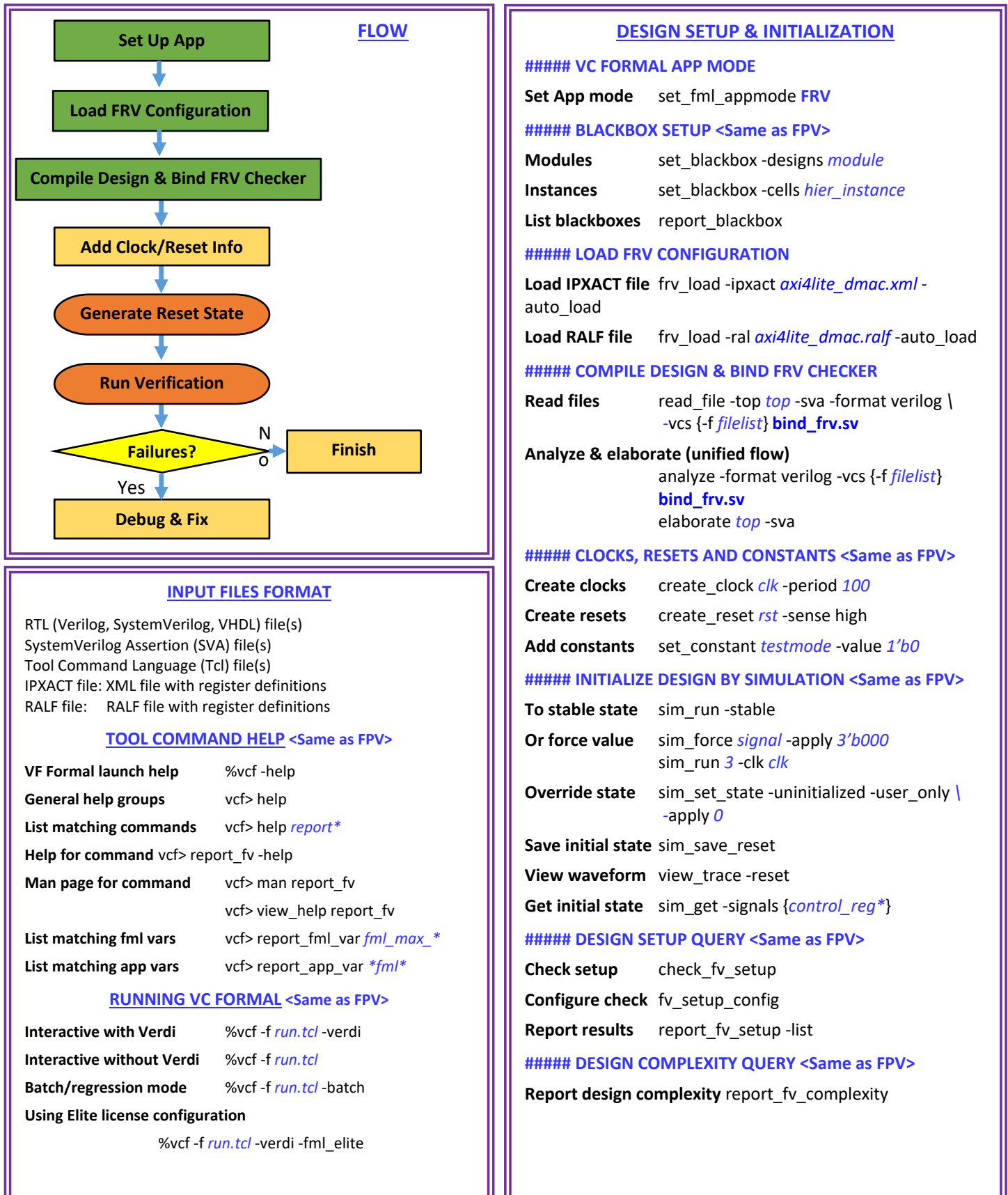
Set active task fvtask *task*

Get active task get\_fvtask

### ##### EXIT VC FORMAL <Same as FPV>

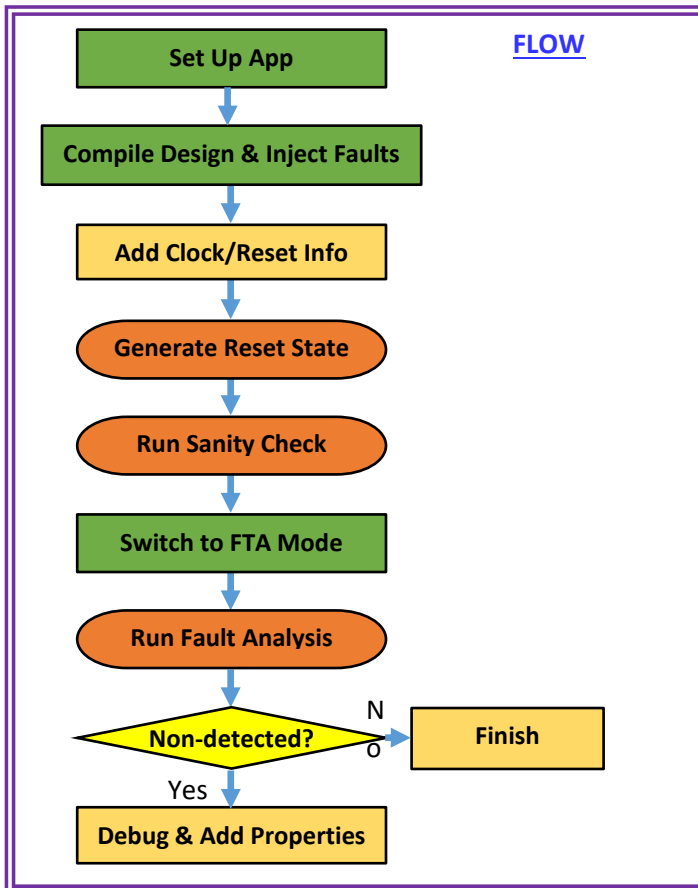
Exit VC Formal quit

## Quick Reference Guide - VC Formal - Formal Register Verification App





## Quick Reference Guide - VC Formal - Formal Testbench Analyzer App



**INPUT FILES FORMAT <Same as FPV>**

RTL (Verilog, SystemVerilog, VHDL) file(s)  
 SystemVerilog Assertion (SVA) file(s)  
 Tool Command Language (Tcl) file(s)

**TOOL COMMAND HELP <Same as FPV>**

**VF Formal launch help**    %vcf -help  
**General help groups**    vcf> help Formal\_FTA  
**List matching commands** vcf> help *report\**  
**Help for command**    vcf> report\_fv -help  
**Man page for command** vcf> man report\_fv  
**List matching fml vars**    vcf> report\_fml\_var *fml\_max\_\**  
**List matching app vars**    vcf> report\_app\_var *\*fml\**

**RUNNING VC FORMAL <Same as FPV>**

**Interactive with Verdi**    %vcf -f *run.tcl* -verdi  
**Interactive without Verdi** %vcf -f *run.tcl*  
**Batch/regression mode**    %vcf -f *run.tcl* -batch  
**Using Ultra or Elite license configuration**  
                                  %vcf -f *run.tcl* -verdi -fml\_elite

**DESIGN SETUP & INITIALIZATION**

**##### SET VC FORMAL APP MODE**

**Set App mode**    set\_fml\_appmode *FTA*

**##### ADD BLACKBOXES <Same as FPV>**

**Modules**    set\_blackbox -designs *module*  
**Instances**    set\_blackbox -cells *hier\_instance*  
**List blackboxes**    report\_blackbox

**##### CONFIGURE FAULT INJECTION & ANALYSIS**

**Specify targets**    fta\_init  
**Configure faults**    configure\_fta\_props

**##### COMPILE DESIGN & INJECT FAULTS**

**Read files**    read\_file -top *top* -sva -format verilog \  
                                  -vcs {-f *filelist*} -inject\_fault all

**Analyze & elaborate (unified flow)**  
                                  set\_app\_var fml\_multi\_step\_fta\_flow true  
                                  analyze -format verilog -vcs {-f *filelist*}  
                                  elaborate *top* -sva -inject\_fault all

**##### MANAGE CERTITUDE FAULT DATABASE**

**Read database**    read\_faultdb -name *certitude\_db*  
**Save database**    save\_faultdb -name *certitude\_db*

**##### CLOCKS, RESETS, AND CONSTANTS <Same as FPV>**

**Create clocks**    create\_clock *clk* -period *100*  
**Create resets**    create\_reset *rst* -sense high  
**Add constants**    set\_constant *testmode* -value *1'b0*  
**Set input transition**    set\_change\_at -clock *clk* -default

**##### INITIALIZE DESIGN BY SIMULATION <Same as FPV>**

**To stable state**    sim\_run -stable  
**Or force value**    sim\_force *signal* -apply *3'b000*  
                                  sim\_run *3* -clk *clk*  
**Override state**    sim\_set\_state -uninitialized -user\_only \  
                                  -apply *0*  
**Save initial state**    sim\_save\_reset  
**View waveform**    view\_trace -reset  
**Get initial state**    sim\_get -signals {*control\_reg\**}

**##### DESIGN COMPLEXITY QUERY <Same as FPV>**

**Report design complexity**    report\_fv\_complexity

**##### CHECK DESIGN SETUP <Same as FPV>**

**Check setup**    check\_fv\_setup  
**Configure check**    fv\_setup\_config  
**Report results**    report\_fv\_setup -list



## VERIFICATION & DEBUG

### ##### MANAGE PROPERTIES <Same as FPV>

Create property   fvassume <name> -expr {<expression>}  
                  fvassert <name> -expr {<expression>}  
                  fvcover <name> -expr {<expression>}

Disable property               fvdisable *property*

Enable property                fvenable *property*

Change property to assertion   fvassert *property*

Change property to constraint   fvassume *property*

Change property to cover       fvcover *property*

### ##### CONFIGURE COMPUTE RESOURCES <Same as FPV>

Set max run time   set\_fml\_var fml\_max\_time *24H*

Set max memory   set\_fml\_var fml\_max\_memory *32GB*

Set engine progress timeout  
                  set\_fml\_var fml\_progress\_time\_limit *2H*

Set grid settings   set\_grid\_usage -type RSH=*12*  
                  set\_grid\_usage -file *hostfile*

### ##### ENABLE/DISABLE VACUITY CHECK <Same as FPV>

Check vacuity   set\_fml\_var fml\_vacuity\_on true

### ##### ENABLE/DISABLE WITNESS TRACE GENERATION <Same as FPV>

Find witness   set\_fml\_var fml\_witness\_on false

### ##### RUN VERIFICATION

Run           compute\_fta -par\_task FPV  
              check\_fv

Stop run       check\_fv -stop

Blocking mode   compute\_fta -par\_task FPV -block  
                  check\_fv -block

Callback task   compute\_fta -par\_task FPV \  
                  -run\_finish {*report\_fv -list*}  
                  check\_fv -run\_finish {*report\_fv -list*}

### ##### REPORT VERIFICATION RESULTS <Same as FPV>

Report results   report\_fv [-list] -verbose]

### ##### QUERY FAULT INFORMATION

Report faults   fta\_report -instance *instance*  
                  get\_fta\_faults *instance*

Report activated properties  
                  get\_activated\_props *fault\_id*

### ##### MANAGE FTA WAIVER FILES

Read waiver files   read\_fta\_waiver -elfiles *filelist*

Remove waiver files   remove\_fta\_waiver -elfiles *filelist*

## VERIFICATION & DEBUG (Continued)

### ##### SAVE/RESTORE SESSION <Same as FPV>

Save session       save\_session -session *session*

Restore session   restore\_session -session *session*

Start from stored session %vcf -restore -session *session*

### ##### VIEW TRACES FOR DEBUGGING <Same as FPV>

Open trace       view\_trace -property *property*

Save trace       fvtrace -property *property*

### ##### GENERATE VERIFICATION SUMMARY <Same as FPV>

Compute summary       compute\_verification\_summary

Configure tags       set\_verification\_summary

Waive/unwaive tags   waive\_verification\_summary

Report results summary   report\_verification\_summary

### ##### CLUSTER FAULTS FOR ANALYSIS

Cluster faults       cluster\_fta\_faults

Report clusters   report\_fta\_fault\_clusters

### ##### STEPS TO ADVANCED DEBUGGING

Export detected faults (FPV/SEQ)   export\_fault

Enable non-detected fault debugging in SEQ App

                  set\_fml\_var fml\_fta\_seq\_debug true

Debug non-detected faults (SEQ)   debug\_fta

Dump user setup                   debug\_fta -dump\_setup

Run *modified* user setup           debug\_fta -run\_user\_setup

## PERFORMANCE & CONVERGENCE

### ##### SET RUN EFFORT LEVEL

Standard proof recipe   set\_fml\_var fml\_effort default

Heavy proof recipe for fault analysis

                  set\_fml\_var fml\_fta\_high\_effort true

### ##### SET RUN SCOPE

Check faults in Formal Core only

                  set\_fml\_var fml\_qual\_fault\_in\_fcore true

Include cover properties for fault analysis

                  set\_fml\_var fml\_fta\_enable\_cover true

### ##### MONITOR PROGRESS QUERY <Same as FPV>

Engine status       report\_fml\_engines

Grid job status   report\_fml\_jobs

Grid hosts status   report\_fml\_hosts

## **PERFORMANCE & CONVERGENCE (Continued)**

### **##### ADD ABSTRACTIONS <Same as FPV>**

**Create cutpoints** snip\_driver *net*

**Abstract design constructs**

set\_abstractions -construct {mult=*16* \  
count=*4*}

**Report abstracted constructs**

report\_abstractions

### **##### IDENTIFY REDUCED CONSTRAINTS FOR PROOFS <Same as FPV>**

**Compute reduced constraints**

compute\_reduced\_constraints \  
-property *property*

**Report results** report\_reduced\_constraints

**Get results (Tcl)** get\_reduced\_constraints

### **##### IDENTIFY FORMAL CORE FOR PROOFS <Same as FPV>**

**Compute Formal Core**

compute\_formal\_core -property *property*

**Report results** report\_formal\_core

**Get results (Tcl)** get\_formal\_core

### **##### MANAGE TASKS <Same as FPV>**

**Create task** fvtask -create *task*

**Copy task** fvtask *new\_task* -copy *task*

**Copy elements between tasks**

fvtask *new\_task* -copy *task* \  
[-assumes <*list*>] [-asserts <*list*>] \  
[-covers <*list*>] [-constants <*list*>] \  
[-snips <*list*>] [-changeats<*list*>] \  
[-keep\_status] [-keep\_task]

**Set active task** fvtask *task*

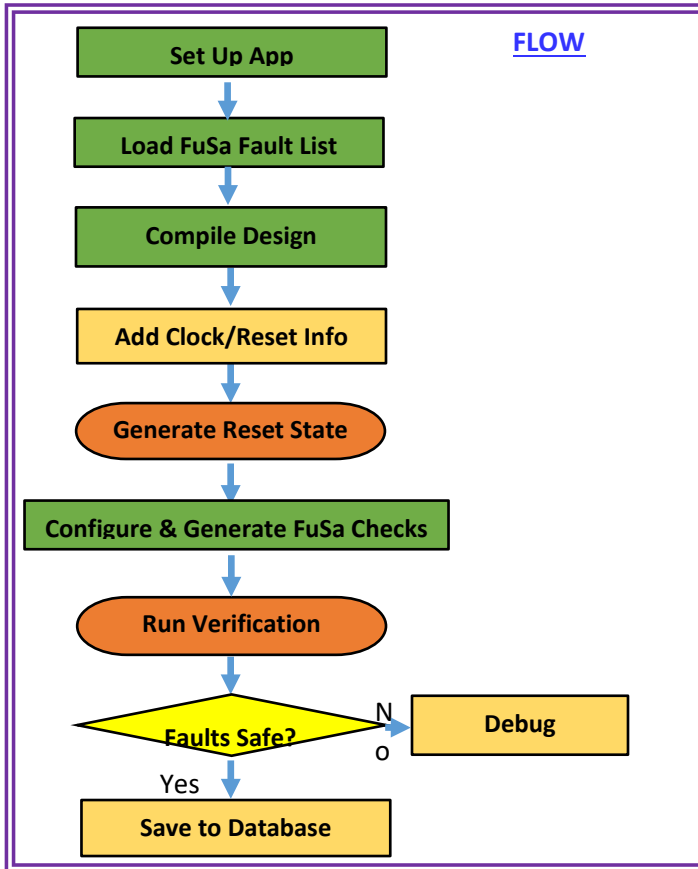
**Get active task** get\_fvtask

### **##### EXIT VC FORMAL <Same as FPV>**

**Exit VC Formal** quit



## Quick Reference Guide - VC Formal – Functional Safety Verification App



### DESIGN SETUP & INITIALIZATION

#### ##### VC FORMAL APP MODE

**Set App mode** set\_fml\_appmode *FUSA*

#### ##### BLACKBOX SETUP <Same as FPV>

**Modules** set\_blackbox -designs *module*

**Instances** set\_blackbox -cells *hier\_instance*

**List blackboxes** report\_blackbox

#### ##### LOAD FuSa FAULT LIST

**Load FDB** fusa\_config -fdb\_campaign *fdbname*

**Load SFF file** fusa\_config -sff *input.sff*

#### ##### COMPILE DESIGN <Same as FPV>

**Read files** read\_file -top *top* -sva -format verilog \ -vcs {-f *filelist*}

#### Analyze & elaborate (unified flow)

analyze -format verilog -vcs {-f *filelist*}  
elaborate *top* -sva

#### ##### CLOCKS, RESETS AND CONSTANTS <Same as FPV>

**Create clocks** create\_clock *clk* -period *100*

**Create resets** create\_reset *rst* -sense high

**Add constants** set\_constant *testmode* -value *1'b0*

#### ##### INITIALIZE DESIGN BY SIMULATION <Same as FPV>

**To stable state** sim\_run -stable

**Or force value** sim\_force *signal* -apply *3'b000*  
sim\_run *3* -clk *clk*

**Override state** sim\_set\_state -uninitialized -user\_only \ -apply *0*

**Save initial state** sim\_save\_reset

**View waveform** view\_trace -reset

**Get initial state** sim\_get -signals {*control\_reg\**}

#### ##### DESIGN SETUP QUERY <Same as FPV>

**Check setup** check\_fv\_setup

**Configure check** fv\_setup\_config

**Report results** report\_fv\_setup -list

#### ##### DESIGN COMPLEXITY QUERY <Same as FPV>

**Get design data** report\_fv\_complexity

#### ##### CONFIGURE & GENERATE FuSa CHECKS

**Add Observation Point** fusa\_observation -add *obs\_point*

**Add Detection Point** fusa\_detection -add *det\_point*

### INPUT FILES FORMAT

RTL (Verilog, SystemVerilog, VHDL) file(s)  
SystemVerilog Assertion (SVA) file(s)  
Tool Command Language (Tcl) file(s)  
Standard Fault File (SFF), Fault Database (FDB)

#### TOOL COMMAND HELP <Same as FPV>

**VF Formal launch help** %vcf -help  
**General help groups** vcf> help Formal\_FUSA  
**List matching commands** vcf> help *report\**  
**Help for command** vcf> report\_fv -help  
**Man page for command** vcf> man report\_fv  
vcf> view\_help report\_fv  
**List matching fml vars** vcf> report\_fml\_var *fml\_max\_\**  
**List matching app vars** vcf> report\_app\_var *\*fml\**

#### RUNNING VC FORMAL <Same as FPV>

**Interactive with Verdi** %vcf -f *run.tcl* -verdi  
**Interactive without Verdi** %vcf -f *run.tcl*  
**Batch/regression mode** %vcf -f *run.tcl* -batch  
**Using Elite license configuration**  
%vcf -f *run.tcl* -verdi -fml\_elite

**Define FuSa blackboxes** fusa\_blackbox -all \  
 -all\_auto\_path -seq\_path  
**Generate FuSa Checks** fusa\_generate

### VERIFICATION & DEBUG

#### ##### MANAGE PROPERTIES

**Create property** fvassume <name> -expr {<expression>}  
 fvassert <name> -expr {<expression>}  
 fvcover <name> -expr {<expression>}

**Disable property** fvdisable *property*

**Enable property** fvenable *property*

#### ##### CONFIGURE COMPUTE RESOURCES <Same as FPV>

**Set max run time** set\_fml\_var fml\_max\_time *24H*  
**Set max memory** set\_fml\_var fml\_max\_memory *32GB*  
**Set engine progress timeout**  
 set\_fml\_var fml\_progress\_time\_limit *2H*  
**Set grid settings** set\_grid\_usage -type RSH=*10*  
 set\_grid\_usage -file *hostfile*

#### ##### RUN VERIFICATION

##### Run structural analysis

set\_fml\_var fusa\_run\_mode structural  
 check\_fv

##### Run controllability analysis

set\_fml\_var fusa\_run\_mode control  
 check\_fv

##### Observability analysis

set\_fml\_var fusa\_run\_mode observe  
 check\_fv

**Stop run** check\_fv -stop

**Blocking mode** check\_fv -block

#### ##### REPORT VERIFICATION RESULTS

**Report results** report\_fv

**Report as list** report\_fv -list

**Report details** report\_fv -verbose

##### FuSa VCF input fault report

fusa\_report\_total\_faults

**FuSa report** fusa\_report

**FuSa report as list** fusa\_report -list

**FuSa report FDB Version** fusa\_report -fdb\_version

##### FuSa report all failure modes

fusa\_report -all\_failure\_modes

#### ##### SAVE FuSa RESULTS

**Save results in FDB** fusa\_save

**Save results as SFF** fusa\_save -sff *sff*

#### ##### SAVE/RESTORE SESSION <Same as FPV>

**Save session** save\_session -session *session*

**Restore session** restore\_session -session *session*

**Start from stored session** %vcf -restore -session *session*

### PERFORMANCE & CONVERGENCE

#### ##### SET RUN EFFORT LEVEL

**Standard proof recipe** set\_fml\_var fml\_effort default  
**Easy proof recipe** set\_fml\_var fml\_effort easy  
**Heavy proof recipe** set\_fml\_var fml\_effort high  
**Bounded proof** set\_fml\_var fml\_effort bounded  
**Falsification/covers** set\_fml\_var fml\_effort bug\_hunting

##### Large number of properties

set\_fml\_var fml\_effort No\_decompose\_expensive

#### ##### MONITOR PROGRESS QUERY <Same as FPV>

**Engine status** report\_fml\_engines

**Grid job status** report\_fml\_jobs

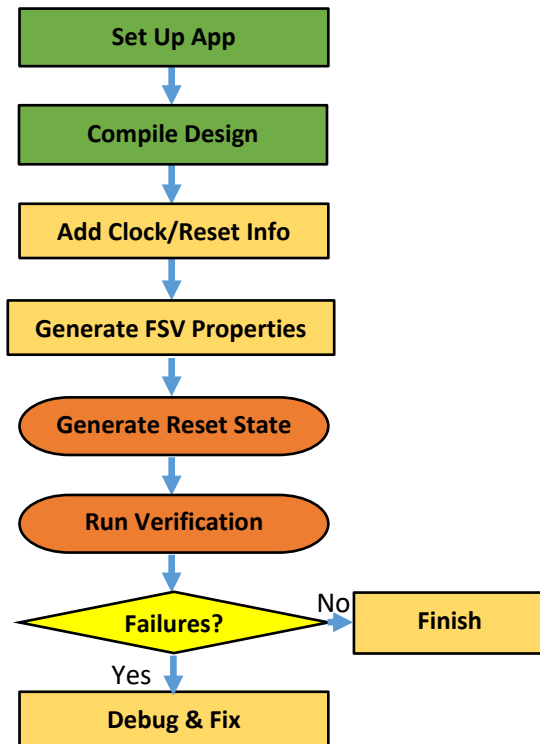
**Grid hosts status** report\_fml\_hosts

#### ##### EXIT VC FORMAL <Same as FPV>

**Exit VC Formal** quit

## Quick Reference Guide - VC Formal - Formal Security Verification App

### FLOW



### DESIGN SETUP & INITIALIZATION

#### ##### VC FORMAL APP MODE

**Set App mode**    set\_fml\_appmode **FSV**

#### ##### BLACKBOX SETUP

**Modules**        set\_blackbox -designs *module*

**Instances**      set\_blackbox -cells *hier\_instance*

**FSV blackbox**    fsv\_blackbox *instance*

**List blackboxes** report\_blackbox

#### ##### COMPILE DESIGN <Same as FPV>

**Read files**      read\_file -top *top* -sva -format verilog \

-vcs {-f *filelist*}

#### **Analyze & elaborate**

analyze -format verilog -vcs {-f *filelist*}

elaborate *top* -sva

#### ##### CLOCKS, RESETS AND CONSTANTS <Same as FPV>

**Create clocks**    create\_clock *clk* -period *100*

**Create resets**    create\_reset *rst* -sense high

**Add constants**   set\_constant *testmode* -value *1'b0*

#### ##### INITIALIZE DESIGN BY SIMULATION <Same as FPV>

**To stable state**   sim\_run -stable

**Or force value**    sim\_force *signal* -apply *3'b000*

sim\_run *3* -clk *clk*

**Override state**    sim\_set\_state -uninitialized -user\_only \

-apply *0*

**Save initial state** sim\_save\_reset

**View waveform**    view\_trace -reset

**Get initial state**   sim\_get -signals {*control\_reg\**}

#### ##### DESIGN SETUP QUERY <Same as FPV>

**Check setup**        check\_fv\_setup

**Configure check**   fv\_setup\_config

**Report results**     report\_fv\_setup -list

#### ##### DESIGN COMPLEXITY QUERY <Same as FPV>

**Get design data**    report\_fv\_complexity

### INPUT FILES FORMAT <Same as FPV>

RTL (Verilog, SystemVerilog, VHDL) file(s)

SystemVerilog Assertion (SVA) file(s)

Tool Command Language (Tcl) file(s)

### TOOL COMMAND HELP <Same as FPV>

**VF Formal launch help**    %vcf -help

**General help groups**      vcf> help Formal\_FSV

**List matching commands**   vcf> help *report\**

**Help for command**        vcf> report\_fv -help

**Man page for command**    vcf> man report\_fv

**List matching fml vars**    vcf> report\_fml\_var *fml\_max\_\**

**List matching app vars**    vcf> report\_app\_var *\*fml\**

### RUNNING VC FORMAL <Same as FPV>

**Interactive with Verdi**    %vcf -f *run.tcl* -verdi

**Interactive without Verdi** %vcf -f *run.tcl*

**Batch/regression mode**    %vcf -f *run.tcl* -batch

**Using Elite license configuration**

%vcf -f *run.tcl* -verdi -fml\_ultra

## VERIFICATION & DEBUG

### ##### MANAGE PROPERTIES <Same as FPV>

#### Generate FSV properties

fsv\_generate -src *src\_sig* -dest *dest\_sig*

Create property fvassume <*name*> -expr {<*expression*>}

fvassert <*name*> -expr {<*expression*>}

fvcover <*name*> -expr {<*expression*>}

Disable property fvdisable *property*

Enable property fvenable *property*

Change property to assertion fvassert *property*

Change property to constraint fvassume *property*

Change property to cover fvcover *property*

### ##### CONFIGURE COMPUTE RESOURCES

Set max run time set\_fml\_var fml\_max\_time *24H*

Set max memory set\_fml\_var fml\_max\_memory *32GB*

#### Set engine progress timeout

set\_fml\_var fml\_progress\_time\_limit *2H*

Set grid settings set\_grid\_usage -type RSH=*10*

set\_grid\_usage -file *hostfile*

#### Set backup grid settings

set\_backup\_grid\_usage -file *hostfile*

### ##### ENABLE/DISABLE VACUITY CHECK <Same as FPV>

Check vacuity set\_fml\_var fml\_vacuity\_on true

### ##### ENABLE/DISABLE WITNESS TRACE GENERATION <Same as FPV>

Find witness set\_fml\_var fml\_witness\_on false

### ##### RUN VERIFICATION <Same as FPV>

Run check\_fv

Stop run check\_fv -stop

Blocking mode check\_fv -block

Callback task check\_fv -run\_finish {*report\_fv -list*}

### ##### REPORT VERIFICATION RESULTS

#### Report FSV properties

fsv\_report

Report results report\_fv

Report as list report\_fv -list

Report details report\_fv -verbose

### ##### SAVE/RESTORE SESSION <Same as FPV>

Save session save\_session -session *session*

Restore session restore\_session -session *session*

Start from stored session %vcf -restore -session *session*

## VERIFICATION & DEBUG (Continued)

### ##### VIEW TRACES FOR DEBUGGING <Same as FPV>

Open trace view\_trace -property *property*

Open vacuity view\_trace -vacuity *property*

Open witness view\_trace -witness *property*

## PERFORMANCE & CONVERGENCE

### ##### SET RUN EFFORT LEVEL

Standard proof recipe set\_fml\_var fml\_effort default

Heavy proof recipe set\_fml\_var fml\_effort high

Bounded proof set\_fml\_var fml\_effort bounded

Falsification/covers set\_fml\_var fml\_effort bug\_hunting

Easy proof recipe set\_fml\_var fml\_effort easy

#### No expensive decomposition proof recipe

set\_fml\_var fml\_effort no\_decompose\_expensive

### ##### ENABLE RESUME FOR FSV

Enable resume set\_fml\_var fml\_enable\_resume true

### ##### MONITOR PROGRESS QUERY <Same as FPV>

Engine status report\_fml\_engines

Grid job status report\_fml\_jobs

Grid hosts status report\_fml\_hosts

### ##### ADD ABSTRACTIONS <Same as FPV>

Create cutpoints snip\_driver *net*

#### Abstract design constructs

set\_abstractions -construct {mult=*16* \  
count=*4*}

#### Report abstracted constructs

report\_abstractions

### ##### IDENTIFY REDUCED CONSTRAINTS FOR PROOFS <Same as FPV>

#### Compute reduced constraints

compute\_reduced\_constraints \  
-property *property*

Report results report\_reduced\_constraints

Get results (Tcl) get\_reduced\_constraints

### ##### IDENTIFY FORMAL CORE FOR PROOFS <Same as FPV>

#### Compute Formal Core

compute\_formal\_core -property *property*

Report results report\_formal\_core

Get results (Tcl) get\_formal\_core

## PERFORMANCE & CONVERGENCE (Continued)

### ##### MANAGE TASKS <Same as FPV>

**Create task**      fvtask -create *task*

**Copy task**        fvtask *new\_task* -copy *task*

**Copy elements between tasks**

fvtask *new\_task* -copy *task* \  
[-assumes <*list*>] [-asserts <*list*>] \  
[-covers <*list*>] [-constants <*list*>] \  
[-snips <*list*>] [-changeats<*list*>] \  
[-keep\_status] [-keep\_task]

**Set active task**   fvtask *task*

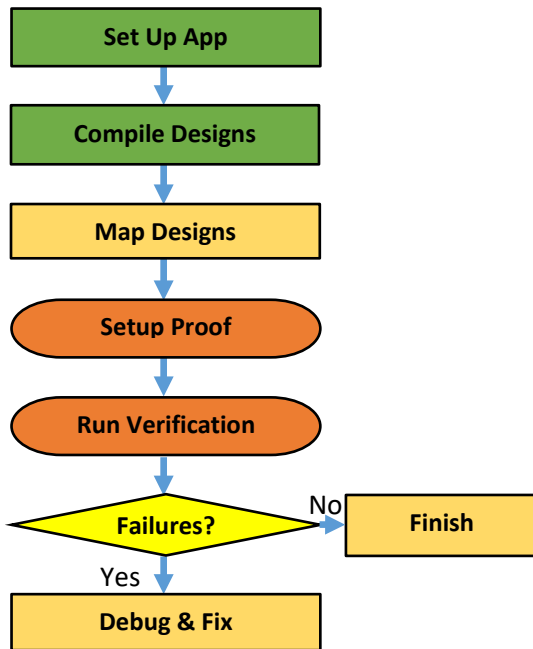
**Get active task**   get\_fvtask

### ##### EXIT VC FORMAL <Same as FPV>

**Exit VC Formal**   quit

## Quick Reference Guide - VC Formal - Data Path Validation App

### FLOW



### INPUT FILES FORMAT

C/C++/SystemC files(s)  
 RTL (Verilog, SystemVerilog, VHDL) file(s)  
 SystemVerilog Assertion (SVA) file(s)  
 Tool Command Language (Tcl) file(s)

### TOOL COMMAND HELP

**VF Formal launch help**    %vcf -help  
**General help groups**    vcf> help Formal\_DPV  
**List matching commands** vcf> help *report\**  
**Help for command**        vcf> listlemmas -help

### RUNNING VC FORMAL

**Interactive with Verdi**  
                               %vcf -f *run.tcl* -verdi -fmode DPV  
**Interactive without Verdi**  
                               %vcf -f *run.tcl* -fmode DPV  
**Batch/regression mode**  
                               %vcf -f *run.tcl* -batch -fmode DPV

### DESIGN SETUP & INITIALIZATION

#### ##### SET DPV APP MODE

**Set App mode**    set\_fml\_appmode **DPV**  
**Get App mode**    get\_fml\_appmode

#### ##### NEW COMPILATION FLOW SETTING

**C compilation**    set\_hector\_comp\_use\_new\_flow true

#### ##### CONFIGURE AEP CHECKS

**Set AEP checks**    set\_aep\_selection default  
**List AEP checks**    list\_aep\_selection

#### ##### BLACKBOX SETUP

**Modules**            set\_blackbox *module*  
**Instances**          set\_blackbox *hier\_instance*  
**Functions**          ignore\_functions "*list\_of\_functions*"  
**Blackbox Report**    get\_blackbox

#### ##### COMPILE DESIGN

**Create design**    create\_design -name *spec/impl* \  
   -top *spectop/impltop* \  
   -clock *clk* -reset *resetN* -negReset \  
   -options "*compilation\_options*" \  
   -lang "*c/c++/verilog/sverilog*"

**C/C++ compilation**    cppan -D<*defines*> \  
   -I <*include\_directories*> \  
   <*C/C++ filenames*>

**SystemC compilation**    scdtan -D<*defines*> \  
   -I <*include\_directories*> \  
   <*C/C++ filenames*>

**RTL compilation**        vcs -sverilog -pvalue+<> -f *filelist*  
   vlogan <*options*> <*filelist*>  
   vhdlan <*options*> <*filelist*>

**Compile design**        compile\_design *spec/impl*

#### ##### CONSTANTS

**Add constants**        fvassume -always \  
   -expr (*impl.testmode==1'b0*)

#### ##### MAPPING

##### Map by name inputs/outputs

                          map\_by\_name -specphase <*phase*> \  
   -implphase <*phase*> -inputs|-outputs

**Get Map info**            report\_dpv\_mappings

#### ##### DESIGN COMPLEXITY QUERY

**Get design data**    report\_fv\_complexity

## VERIFICATION & DEBUG

### ##### PROOF PROCEDURE

#### Set main proof procedure

set\_user\_assumes\_lemmas\_procedure *"ual"*

#### Set case split procedure

set\_hector\_case\_splitting\_procedure *"case\_split\_strategy"*

### ##### ASSUMPTIONS

Add assumptions fvassume -expr *"spec.opcode(1)==3'd3"*

### ##### LEMMAS

#### Add lemmas

fvassert -expr *"impl.vld(7) |-> (spec.out(1)==impl.res(7))"*

### ##### COVER PROPERTIES

Add covers fvcover -expr *"impl.res(3)==4'd6"*

### ##### DELETE PROPERTIES

Delete lemma/assume/cover fvdelete *<propName>*

### ##### CASE ASSUMPTIONS

Add case assume caseAssume (*spec.mult(1)==2'd0*)

### ##### HDPS SETUP

Enable HDPS set\_hector\_rew\_use\_dps\_engine *true*

Select HDPS solve script set\_hector\_rew\_dps\_solve\_script  
*\_\_hector\_orch\_custom\_dps2*

#### Set HDPS resource limit

set\_hector\_rew\_dps\_resource\_limit *1200*

#### Run All HDPS options

run\_all\_hdps\_options

### ##### ENABLE/DISABLE VACUITY or Witness CHECK

Check vacuity set\_app\_var fml\_vacuity\_on *true*

Check witness set\_app\_var fml\_witness\_on *true*

### ##### CONFIGURE COMPUTE RESOURCES

Set resource limit set\_resource\_limit *200*

Set task timeout set\_hector\_task\_timeout *1000*

Set grid settings set\_host\_file *hostfile*

### ##### RUN VERIFICATION

#### Run

check\_fv

solveNB *proofName*

solveNB\_init *proofName*

#### Stop run

check\_fv -stop

#### Blocking mode

check\_fv -block

solve\_init *proofName*

Wait for a proof proofwait

## VERIFICATION & DEBUG (Continued)

### ##### REPORT VERIFICATION RESULTS

Report proofs listproofs

Report proof listproof

Report tasks listtasks

Report task listtask *taskId*

Report assumes listassumes

Report lemmas listlemmas

Report covers listcovers

### ##### SAVE/RESTORE SESSION

Save session save\_session

Restore session restore\_session

Start from stored session %vcf -restore -session *session*

### ##### VIEW TRACES FOR DEBUGGING

Open trace view\_trace -property *propertyname*

Generate trace simcex *propertyname* -print -joint\_compile

GDB trace simcex *propertyname* -gdb -print

Open vacuity view\_trace -vacuity *property*

Check witness view\_trace -witness *property*

## PERFORMANCE & CONVERGENCE

### ##### SET RUN EFFORT LEVEL

#### Custom solve script

set\_custom\_solve\_script *"orch\_abo\_sat"*

#### Enable all solve scripts

set\_hector\_multiple\_solve\_scripts *true*

#### Enable selective solve scripts

set\_hector\_multiple\_solve\_scripts *true*

set\_hector\_multiple\_solve\_scripts\_list *[list scripts]*

Show all solve scripts show\_all\_solve\_scripts

Get all solve scripts get\_all\_solve\_scripts

### ##### MONITOR PROGRESS QUERY

Engine status report\_fml\_engines

Grid job status report\_fml\_jobs

### ##### ADD ABSTRACTIONS

Create cutpoint set\_cutpoint *impltop.partial*

Activate cutpoint cutpoint cutname = *impl.partial(3)*

### ##### IDENTIFY CONFLICTING CONSTRAINTS FOR PROOFS

Compute conflicting constraints conflictcore

## PERFORMANCE & CONVERGENCE (Continued)

### ##### MANAGE TASKS

**Create task**      fvtask -create *task*

**Delete task**      fvtask -delete *task*

**Set active task**    fvtask *task*

### ##### EXIT VC FORMAL

**Exit VC Formal**    quit