Verification Continuum^{TM}

VC Formal
Quick Reference Guide

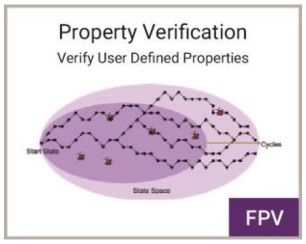
Version T-2022.06-SP1, September 2022

Synopsys Statement on Inclusivity and Diversity

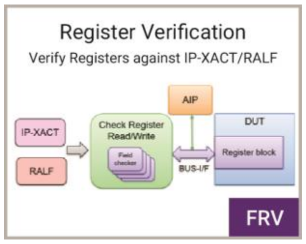
Quick Reference Guide - VC Formal

INTRODUCTION

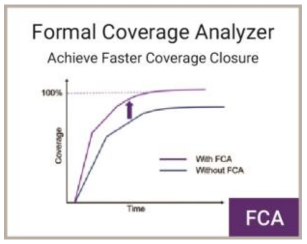
This Quick Reference Guide is to help you get started with VC Formal by showing you the most common flows, commands, and variables. The command options shown are only a sample of the many options that are available.
For detailed information, see the user guides and reference material, available through Documentation on SolvNet:
https://spdocs.synopsys.com/dow_retrieve/latest/home_public/vc_formal.html
Use the FPV Quick Reference Guide as reference for commands and variables applicable to most VC Formal Apps.



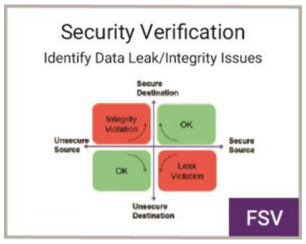
The Formal Property Verification (FPV) App verifies properties in the design including user-defined and Assertion IP (AIP) properties.



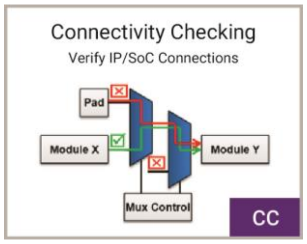
The Formal Register Verification (FRV) App automatically creates and formally verifies checks based on IP-XACT or RALF format for registers in the design



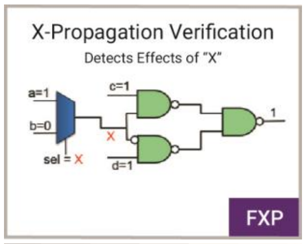
The Formal Coverage Analyzer (FCA) App assists simulation-based verification coverage signoff and qualifies formal property verification with coverage signoff.



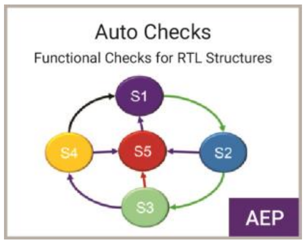
The Formal Security Verification (FSV) App ensures that unexpected data propagation does not happen between secure and non-secure areas.



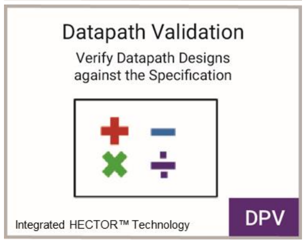
The Formal Connectivity Checking (CC) App checks if there is a structural and functional connection between the source and the destination.



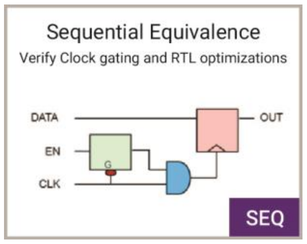
The X-Propagation Verification (FXP) App checks whether unknown signal values (X's) can propagate to dangerous points within the design.



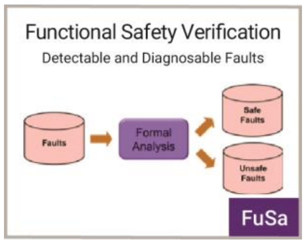
The Automatically Extracted Properties (AEP) App automatically extracts properties from the design and verify them.


Datapath Validation
Verify Datapath Designs against the Specification
Integrated HECTOR™ Technology
DPV

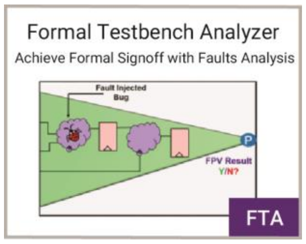
The Data Path Validation (DPV) App uses HECTOR technology to verify data transformation blocks between untimed C/C++ and RTL models.



The Sequential Equivalence Checking (SEQ) App compares two RTL designs and verifies that they are functionally equivalent.


Functional Safety Verification
Detectable and Diagnosable Faults
Faults
Formal Analysis
Safe Faults
Unsafe Faults
FuSa

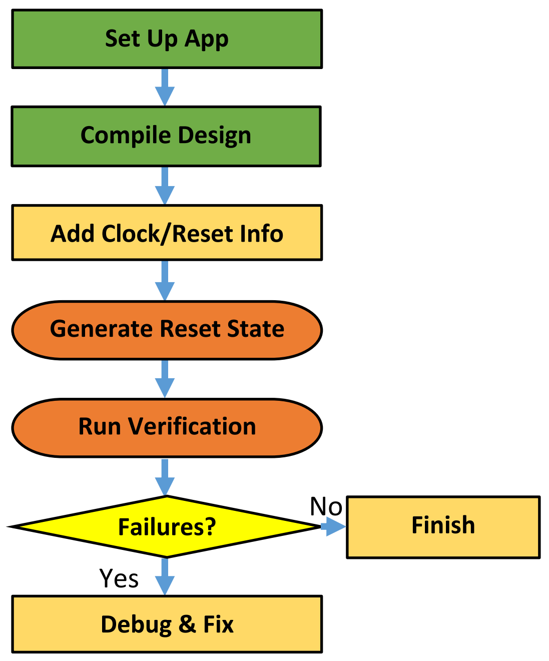
The Functional Safety (FuSa) App analyzes the controllability and observability of Z01X faults to calculate FMEDA (Failure Mode Effects and Diagnostic Analysis) metrics.


Formal Testbench Analyzer
Achieve Formal Signoff with Faults Analysis
Fault Injected Bug
P
FPV Result Y/N?
FTA

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

Quick Reference Guide - VC Formal - Formal Property Verification App

FLOW



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
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_ultra
%vcf -f run.tcl -verdi -fml_elite
```

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 AXI5*"

##### 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

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
Quick proof recipe	set_fml_var fml_effort low
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
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

##### MANAGE TASKS

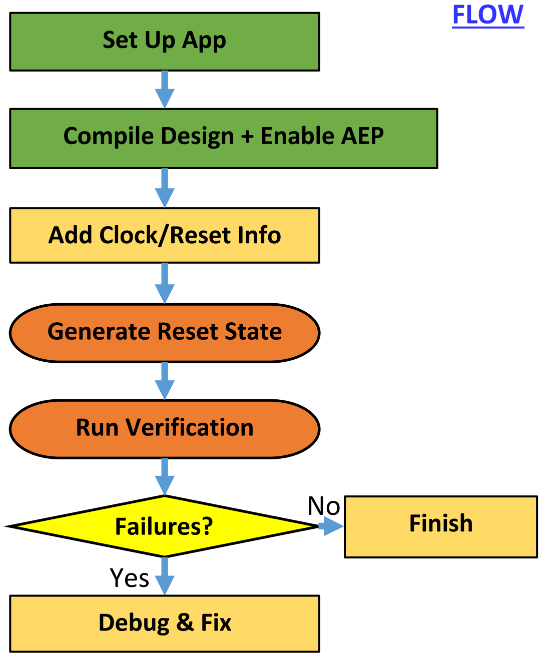
Create task	fvtask -create *task*
Copy task	fvtask *new_task* -copy *task*
Copy elements between tasks	fvtask *new_task* -copy *task* \\ [-assumes <*list*>] [-asserts <*list*>] \\ [-covers <*list*>] [-constants <*list*>] \\ [-snips <*list*>] [-changeats<*list*>] \\ [-keep_status] [-keep_task]
Set active task	fvtask *task*
Get active task	get_fvtask

##### EXIT VC FORMAL

Exit VC Formal	quit

Quick Guide - VC Formal – Automatically Extracted Property App

FLOW



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
List matching fml vars	vcf> report_fml_var **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_ultra
%vcf -f *run.tcl* -verdi -fml_elite

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*}

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 -unitialized -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 etup	check_fv_setup
Configure check	fv_check_config
Report results	report_fv_setup -list

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 *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]
```

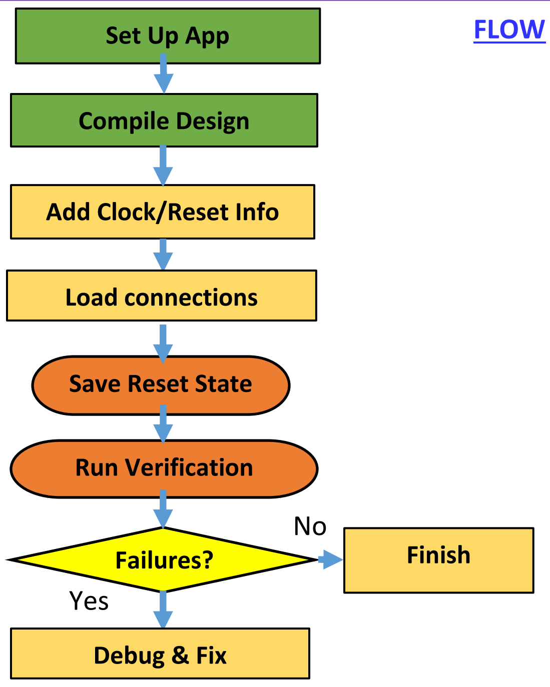
Set active task	fvtask *task*
Get active task	get_fvtask

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

Exit VC Formal	quit

Quick Guide - VC Formal – Connectivity Check App

FLOW



INPUT FILES FORMAT

RTL Format (Verilog, SystemVerilog, VHDL) file(s)
RTL_Format keyword < verilog | sverilog | vhdl >
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
List matching fml vars	vcf> report_fml_var **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_ultra
%vcf -f *run.tcl* -verdi -fml_elite

DESIGN SETUP & INITIALIZATION

##### VC FORMAL APP MODE

Set App mode set_fml_appmode 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

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 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*>]

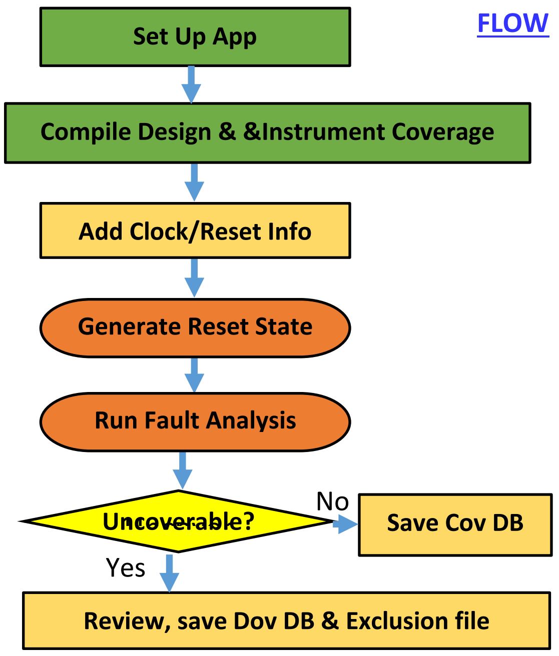
Set active task	fvtask *task*
Get active task	get_fvtask

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

Exit VC Formal	quit

Quick Reference Guide - VC Formal - Formal Coverage Analyzer App

FLOW



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
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_ultra
%vcf -f *run.tcl* -verdi -fml_elite

DESIGN SETUP & INITIALIZATION

##### VC FORMAL APP MODE

Set App mode set_fml_appmode **COV**

##### Enable Coverage for VHDL designs

Set App var set_app_var fml_enable_vhdl_cov true

##### Enable the reset reachable check

Set App var set_fml_var fml_reset_property_check true

##### Enable Branch Coverage, if required

Set App var set_fml_var fml_cov_enable_branch_cov true

##### Enable Toggle Coverage for input ports, if required

Set App var set_app_var fml_cov_tgl_input_port true

##### Extraction of Coverage Options used in Simulation

Set App var set_app_var fml_cov_override_db_opt false

##### Read Simulation Coverage Database, if available

read_covdb -cov_input simv.vdb -cov_dut Testbench.Dut_instance [-elfile elfilename] [-check_el]

##### 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*

##Where Coverage_Metric:
line+cond+tgl+fsm_state+fsm_transition+branch+cg

##### 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*

##### 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 …}

##### 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 Exclusion file save_cov_exclusion -file exclusion.el

Save Coverage db save_covdb -status covered -name cov.db

View Coverage view_coverage -cov_input cov.db -elfiles 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
Quick proof recipe	set_fml_var fml_effort low
Heavy proof recipe	set_fml_var fml_effort high
Bounded proof	set_fml_var fml_effort bounded
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>]
```

Set active task fvtask *task*

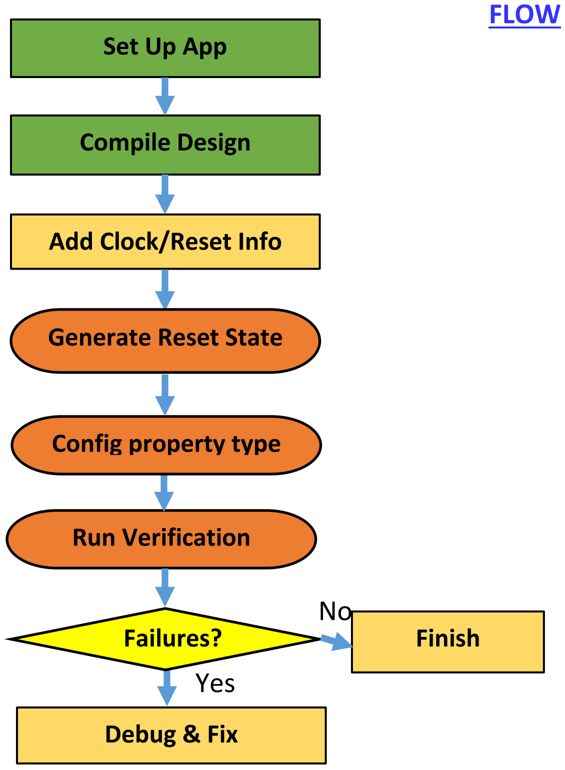
Get active task get_fvtask

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

Exit VC Formal quit

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

FLOW



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_FXP
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 **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_ultra
%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 -unitialized -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 -injecx <*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 *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]
```

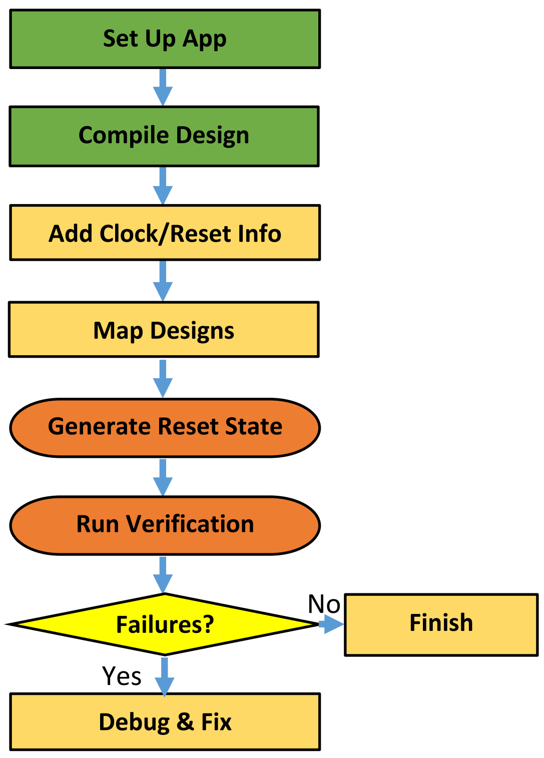
Set active task	fvtask *task*
Get active task	get_fvtask

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

Exit VC Formal	quit

Quick Reference Guide - VC Formal - Sequential Equivalence Checking App

FLOW



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
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_ultra
%vcf -f *run.tcl* -verdi -fml_elite

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*

##### 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

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

Check setup	check_fv_setup
Configure check	fv_setup_config
Report results	report_fv_setup -list

DESIGN SETUP & INITIALIZATION (Continued)

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

Get design data	report_fv_complexity

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

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

Run	check_fv
Stop run	check_fv -stop
Blocking mode	check_fv -block
Callback task	check_fv -run_finish {*report_fv -list*}
Resume analysis	resume_seq

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

Report results	report_fv
Report as list	report_fv -list
Report details	report_fv -verbose

##### QUERY SEQ PROOFS STATUS

Report proofs	report_proofs

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*
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
Quick proof recipe	set_fml_var fml_effort low
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
Hard bit-level proof recipe	set_fml_var fml_effort bitlevel_hard
No expensive decomposition proof recipe	set_fml_var fml_effort no_decompose_expensive

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

Enable RMA	fvlearn_config -local_dir *rma_db*

##### 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

Create cutpoints	snip_driver *net*
Abstract design constructs	set_seq_abstractions -construct {mult=*16* \ add=*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

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*

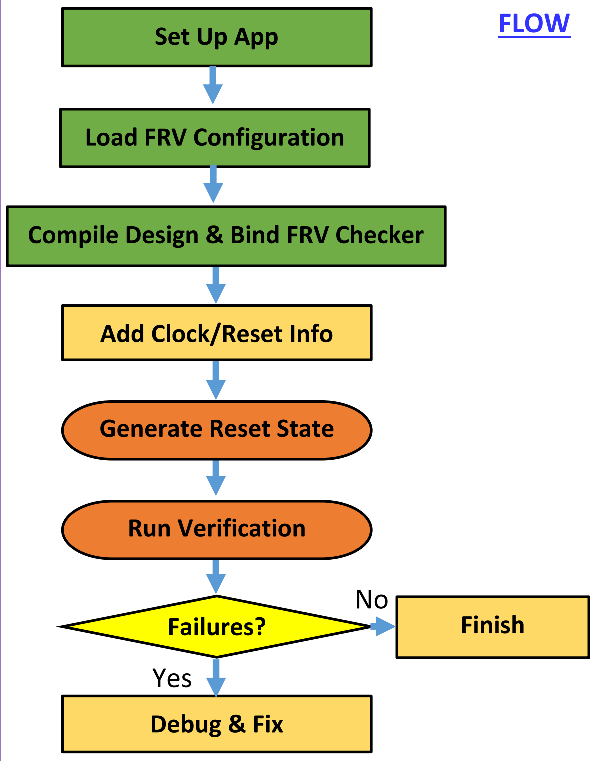
Get active task get_fvtask

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

Exit VC Formal quit

Quick Reference Guide - VC Formal - Formal Register Verification App

FLOW



INPUT FILES FORMAT

RTL (Verilog, SystemVerilog, VHDL) file(s)
SystemVerilog Assertion (SVA) file(s)
Tool Command Language (Tcl) file(s)
IPXACT file: XML file with register definitions
RALF file: RALF file with register definitions

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
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_ultra
%vcf -f *run.tcl* -verdi -fml_elite

DESIGN SETUP & INITIALIZATION

##### VC FORMAL APP MODE

Set App mode	set_fml_appmode **FRV**

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

Modules	set_blackbox -designs *module*
Instances	set_blackbox -cells *hier_instance*
List blackboxes	report_blackbox

##### LOAD FRV CONFIGURATION

Load IPXACT file	frv_load -ipxact *axi4lite_dmac.xml* -auto_load
Load RALF file	frv_load -ral *axi4lite_dmac.ralf* -auto_load

##### COMPILE DESIGN & BIND FRV CHECKER

Read files	read_file -top *top* -sva -format verilog \ -vcs {-f *filelist*} **bind_frv.sv**

Analyze & elaborate (unified flow)

analyze -format verilog -vcs {-f *filelist*} **bind_frv.sv**
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

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=*10* 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 <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 results	report_fv -class register
Report as list	report_fv -class register -list
Report details	report_fv -class register -verbose
Report register data	frv_report

##### 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*
Open vacuity	view_trace -vacuity *property*
Open witness	view_trace -witness *property*

PERFORMANCE & CONVERGENCE

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

Standard proof recipe	set_fml_var fml_effort default
Quick proof recipe	set_fml_var fml_effort low
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
Target hard property	set_fml_var fml_effort discovery

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

Enable RMA	fvlearn_config -local_dir *rma_db*

##### 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

##### 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

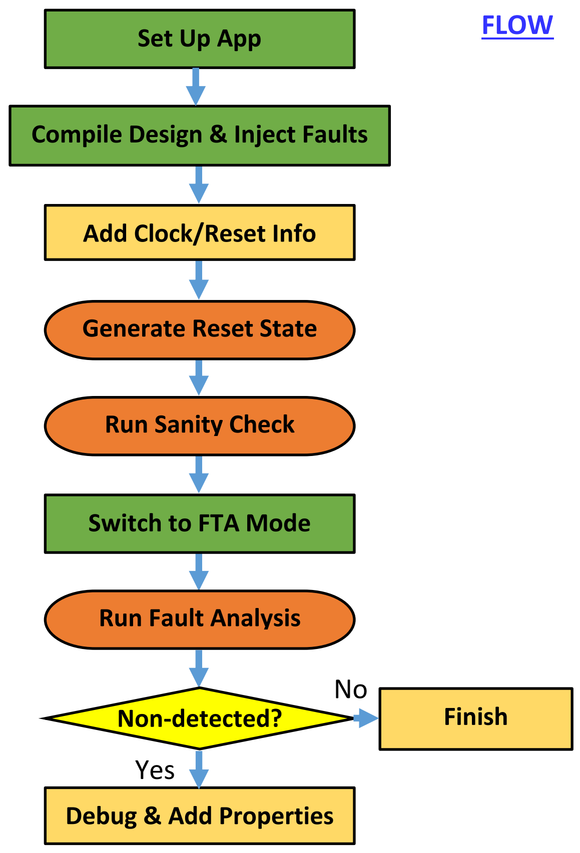
PERFORMANCE & CONVERGENCE (Continued)

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

Exit VC Formal	quit

Quick Reference Guide - VC Formal - Formal Testbench Analyzer App

FLOW



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
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_ultra
%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

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*

##### 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*
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

##### SWITCH TO ADVANCED DEBUGGING

Export detected faults to parent task	export_fault
Debug non-detected faults in SEQ App	debug_fta

PERFORMANCE & CONVERGENCE

##### SET RUN EFFORT LEVEL

Standard proof recipe	set_fml_var fml_effort default
Heavy proof recipe for parent task sanity check	set_fml_var fml_fta_par_app_high_effort true
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

##### 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

PERFORMANCE & CONVERGENCE (Continued)

##### 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]
```

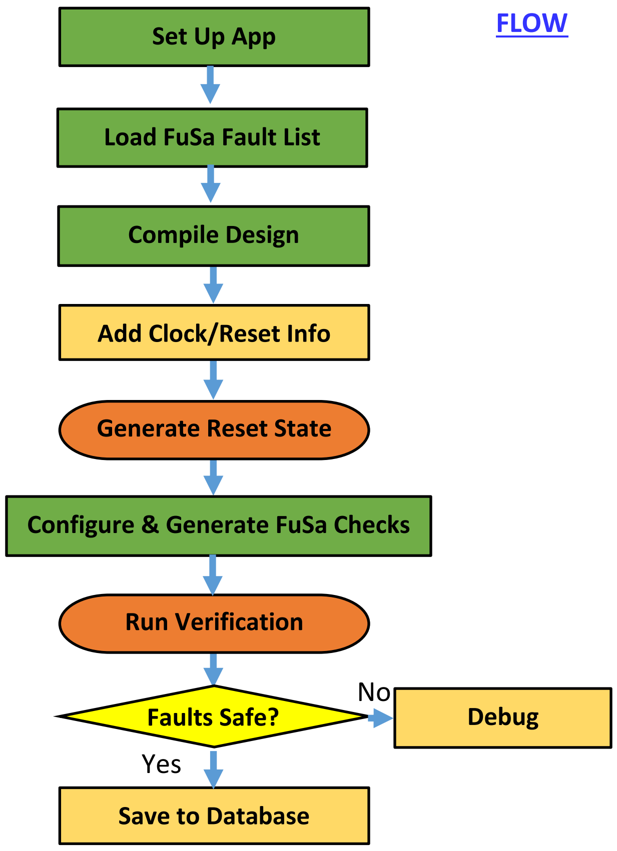
Set active task	fvtask *task*
Get active task	get_fvtask

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

Exit VC Formal	quit

Quick Reference Guide - VC Formal – Functional Safety Verification App

FLOW



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
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_ultra
%vcf -f *run.tcl* -verdi -fml_elite

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*
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 report	fusa_report
FuSa report as list	fusa_report -list

##### 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*

##### 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
Easy proof recipe	set_fml_var fml_effort easy
Quick proof recipe	set_fml_var fml_effort low
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_decopmpose_expensive

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

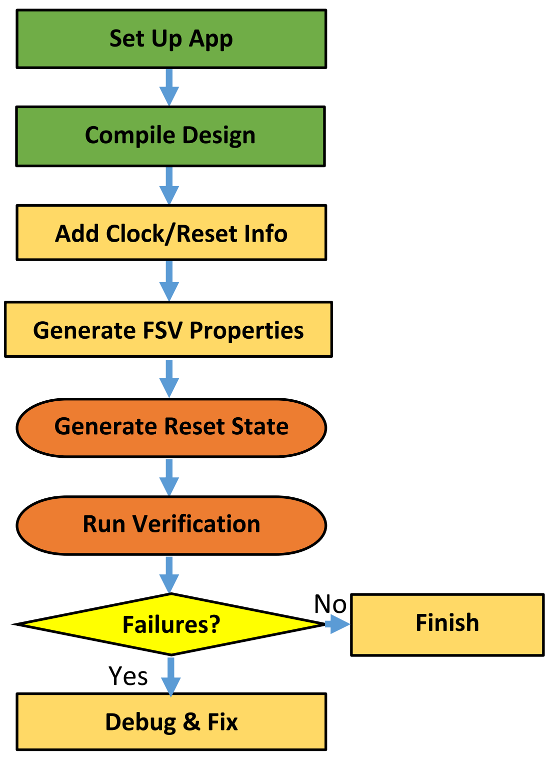
Engine status	report_fml_engines
Grid job status	report_fml_jobs
Grid hosts status	report_fml_hosts

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

Exit VC Formal	quit

Quick Reference Guide - VC Formal - Formal Security Verification App

FLOW



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
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_ultra
%vcf -f *run.tcl* -verdi -fml_elite

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

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
Quick proof recipe	set_fml_var fml_effort low
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

##### 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>]
```

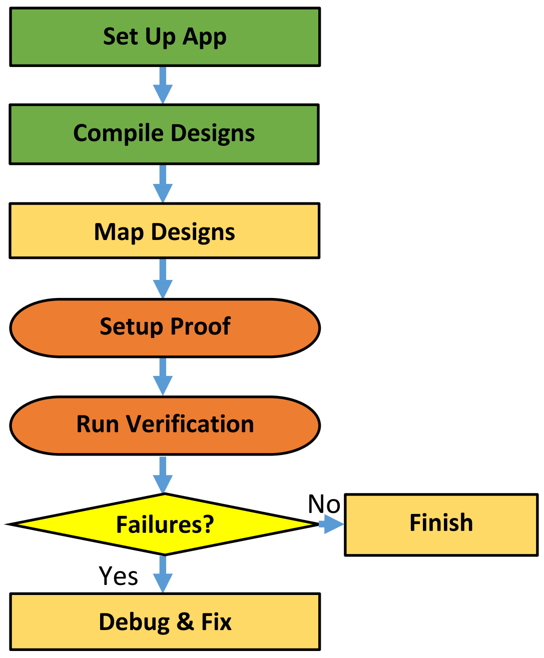
Set active task	fvtask *task*
Get active task	get_fvtask

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

Exit VC Formal	quit

Quick Reference Guide - VC Formal - 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 <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_ultra
%vcf -f *run.tcl* -verdi -fml_elite

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
RTL compilation	set _hector_enable_nldm_compile 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 assume -always *(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 assume (*spec.opcode(1)*==*3'd3*)

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

##### LEMMAS

Add lemmas
lemma *(impl.vld(7) |-> (spec.out(1)==impl.res(7)))*

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

##### COVER PROPERTIES

Add covers cover (*impl.res(3)*==*4'd6*)

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 -encoding *[list radix4]* *hdps_proofname*

##### ENABLE/DISABLE VACUITY CHECK

Check vacuity set_app_var fml_vacuity_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 *proofName*
	solve_init *proofName*

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
	save_proofs
Restore session	restore_session
	restore_proofs

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

##### VIEW TRACES FOR DEBUGGING

Open trace	view_trace -property *propertyname*
	simcex *propertyname* -print -joint_compile
GDB trace	simcex *propertyname* -gdb -print
Open vacuity	view_trace -vacuity *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 CONFLICTNG CONSTRAINTS FOR PROOFS

Compute conflicting constraints conflictcore

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

Exit VC Formal	quit