V3 User Manual

Last Update: 2014/11/20

Cheng-Yin Wu
gro070916@yahoo.com.tw, d99943034@ntu.edu.tw

Chung-Yang (Ric) Huang
ric@cc.ee.ntu.edu.tw

Design Verification Lab (DVLab), NTUGIEE
http://dvlab.ee.ntu.edu.tw

National Taiwan University (NTU), Taiwan
http://www.ntu.edu.tw

This document lists the set of commands developed in $\mathcal{V}3$, which is a new framework towards the research of hardware verification and debugging. Please also refer to the $\mathcal{V}3$ webpage `http://dvlab.ee.ntu.edu.tw/~publication/V3` as well as $\mathcal{V}3$ user guide for detailed usages of $\mathcal{V}3$.

Command-line Lexicographic Notations. Here we give the lexicographic notations of $\mathcal{V}3$ commands:

- **CAPITAL LETTERS or leading '-'**: The leading '-' and capital letters in the command name or parameters are mandatory entries and will be compared *case-insensitively*. The following letters can be partially skipped. However, when entered, they should match the specification *case-insensitively*.
- **Round bracket** `()`: Meaning it should be replaced by a proper argument as suggested by the `(type` *`variable`*`)` description in the round brackets.
- **Angle brackets** `<>`: Mandatory parameters; they should appear in the same relative order as specified in the command usage.
- **Square brackets** `[]`: Optional parameters; they can appear anywhere in the command parameters.
- **Or '|'**: Or condition; expecting either one of the argument.

List of Commands Basically $\mathcal{V}3$ offers a set of commands that are roughly categorized into the following functions, as listed here:

Functionality	Examples
General	`HELp` prints command information `Quit` terminates the program
Design I/O	`REAd Aig` reads in an AIGER [1, 3] design `WRIte Rtl` reports design into Verilog [10]
Design Reporting	`PRInt NTk` reports network information `PLOt NTk` depicts network topology
Design Synthesis	`REWrite NTk` performs rule-based rewriting techniques `FLAtten NTk` flattens a hierarchical network
Design Manipulation	`@CD` change current network into another
Intent Extraction	`EXTract FSM` extracts high-level design information from networks
Design Simulation	`SIM NTk` simulates networks either from given patterns or randomly
Design Verification	`SET PRoperty` creates a property `VERify PDR` runs PDR [7] algorithm for property checking
$\mathcal{V}3$ Model Checking	`READ PROperty` reads in a set of properties `RUN` performs multiple property checking under given resource bounds
Design Debugging	`SIMplify TRace` performs counterexample simplification

The following sections give the descriptions to commands and corresponding parameters. Please note that there is no complete regressions to the current release of $\mathcal{V}3$, so good luck! For bug reports or questions to us, please send an e-mail to `author.v3@gmail.com` or report an issue to `https://chengyinwu@bitbucket.org/chengyinwu/v3.git`.

1 General Commands

Command:	`HELp [<(string cmd) [-Verbose]>]`
Synopsis:	Print this help message.
Parameters:	`(string cmd):` The (partial) name of the command.
	`-Verbose:` Print usage in more detail.

Command:	`DOfile <(string fileName)>`
Synopsis:	Execute the commands in the dofile.
Parameters:	`(string fileName):` The file name of the script.

Command:	`HIStory [(int nPrint)]`
Synopsis:	Print command history.
Parameters:	`(int nPrint):` The number of the latest commands to be printed. (default = ∞)

Command:	`SET LOgfile [|-All | -Cmd | -Error | -Warning | -Info | -Debug] <(string fileName)> [|-File-only | -Both] [-APPend]`
Synopsis:	Redirect messages to files.
Parameters:	`(string fileName):` The file name for message redirection.
	`-APPend:` Append messages to the output file.
	`-All:` Enable any types of messages.
	`-Cmd:` Enable only executed commands.
	`-Error:` Enable only ERROR messages.
	`-Warning:` Enable only WARNING messages.
	`-Info:` Enable only messages for runtime information.
	`-Debug:` Enable only debug messages.
	`-File-only:` Disable message output to standard output.
	`-Both:` Remain message output to standard output.

Command:	`USAGE [-Time-only | -Memory-only]`
Synopsis:	Report resource usage.
Parameters:	`-Time-only:` Disable memory usage reporting.
	`-Memory-only:` Disable time usage reporting.

Command:	`Quit [-Force]`
Synopsis:	Quit the execution.
Parameters:	`-Force:` Quit the program forcedly.

2 Design I/O Commands

Command:	`REAd Aig <(string fileName)> [-Symbol]`
Synopsis:	Read AIGER [1, 3] Designs.
Parameters:	`(string fileName):` The file name of the input AIGER design.
	`-Symbol:` Enable reading of symbolic tables.

Command:	`REAd BTOR <(string fileName)> [-Symbol]`
Synopsis:	Read BTOR [5] Designs.
Parameters:	`(string fileName):` The file name of the input BTOR design.
	`-Symbol:` Enable the reading of symbols (i.e. name of vars).

Command:	`REAd Rtl <(string` *fileName*`)> [-Filelist]`	
	`[-FLAtten] [| -QuteRTL | -Primitive]`	
Synopsis:	Read RTL (Verilog) [10] Designs.	
Parameters:	`(string` *fileName*`):`	The file name of the input Verilog design or a list design files.
	`-Filelist:`	Indicate fileName is a list of design files.
	`-QuteRTL:`	Use QuteRTL [12] RTL front-end for design parsing and synthesis.
	`-Primitive:`	Use V3 Primitive RTL front-end for design parsing and synthesis.
	`-FLAtten:`	Flatten the design after parsing. (only available to -QuteRTL)

Command:	`WRIte Aig <(string` *fileName*`)> [-Symbol]`	
Synopsis:	Write AIGER [1, 3] Designs.	
Parameters:	`(string` *fileName*`):`	The file name for the AIGER output.
	`-Symbol:`	Enable using signal names specified in the input design.

Command:	`WRIte Btor <(string` *fileName*`)> [-Symbol]`	
Synopsis:	Write BTOR [5] Designs.	
Parameters:	`(string` *fileName*`):`	The file name for the BTOR output.
	`-Symbol:`	Enable using signal names specified in the input design.

Command:	`WRIte Rtl <(string` *fileName*`)> [-Symbol] [-Initial]`	
Synopsis:	Write RTL (Verilog) [10] Designs.	
Parameters:	`(string` *fileName*`):`	The file name for the RTL output.
	`-Symbol:`	Enable using signal names specified in the input design.
	`-Initial:`	Enable specifying initial states of a network.

3 Design Reporting Commands

Command:	`PRInt NTk [| -Summary | -Primary | -Verbose | -Netlist |`	
	`-CombLoops | -Floating | -Unreachable]`	
Synopsis:	Print Network Information.	
Parameters:	`-Summary:`	Print network summary.
	`-Primary:`	Print primary ports.
	`-Verbose:`	Print statistics of gates.
	`-Netlist:`	Print network topology.
	`-CombLoops:`	Print combinational loops.
	`-Floating:`	Print floating nets.
	`-Unreachable:`	Print unreachable nets.

Command:	`PRInt NEt <(unsigned` *netId*`)>`	
Synopsis:	Print Net Information.	
Parameters:	`(unsigned` *netId*`):`	The index of a net to be reported.

Command:	`PLOt NTk [| -DOT | -PNG | -PS] <(string fileName)> <(-Level | -Depth) (unsigned size)> [-Monochrome]`
Synopsis:	Plot Network Topology.
Parameters:	`(string fileName):` The file name for output.
	`(unsigned size):` The number of levels or depths to be reported.
	`-Monochrome:` Plot with only black and white colors.
	`-Level:` Enable plotting networks under a specified number of levels.
	`-Depth:` Enable plotting networks under a specified number of time-frames.
	`-DOT:` Output into *.dot files.
	`-PNG:` Output into *.png files (an executable dot required).
	`-PS:` Output into *.ps files (an executable dot required).

4 Design Synthesis Commands

Command:	`REDuce NTk`
Synopsis:	Perform COI Reduction on Current Network.

Command:	`STRash NTk`
Synopsis:	Perform Structural Hashing on Current Network.

Command:	`REWrite NTk`
Synopsis:	Perform Rule-based Rewriting on Current Network.

Command:	`SET NTKVerbosity [-All] [-REDuce] [-Strash] [-REWrite] [-Fwd-map] [-Bwd-map] [-ON |-OFF |-RESET]`
Synopsis:	Set Verbosities for Network Duplication.
Parameters:	`-RESET:` Reset everything to default.
	`-ON:` Turn specified attributes on.
	`-OFF:` Turn specified attributes off.
	`-All:` Toggle all the following attributes.
	`-REDuce:` Toggle COI Reduction. (default = on)
	`-Strash:` Toggle Structural Hashing. (default = off)
	`-REWrite:` Toggle Rule-based Rewriting. (default = off)
	`-Fwd-map:` Toggle Preservation of Forward (to Sucessor) ID Maps. (default = on)
	`-Bwd-map:` Toggle Preservation of Backward (to Ancestor) ID Maps. (default = on)

Command:	`PRInt NTKVerbosity`
Synopsis:	Print Verbosities for Network Duplication.

Command:	`DUPlicate NTk`
Synopsis:	Duplicate Current Ntk from Verbosity Settings.

Command:	`BLAst NTk [-Primary]`
Synopsis:	Bit-blast Word-level Networks into Boolean-level Networks.
Parameters:	`-Primary:` Bit-blast primary inputs, inouts, and latches only.

Command:	`EXPand NTk <(unsigned cycle)> [-Initial]`
Synopsis:	Perform Time-frame Expansion for Networks.
Parameters:	`(unsigned cycle):` The number of time-frames to be considered.
	`-Initial:` Set initial state values at the first timeframe.

Command:	`FLAtten NTk [(unsigned level)]`
Synopsis:	Flatten Hierarchical Networks.
Parameters:	`(unsigned level):` The number of hierarchical levels to be flattened. (default = ∞)

Command:	`MITer NTk <(unsigned ntkId1)> <(unsigned ntkId2)> [-Merge] [-Name <(string miterNtkName)>] [| -SEC	-CEC]`
Synopsis:	Miter Two Networks.
Parameters:	`(unsigned ntkId1):` The index of the first network.
	`((unsigned ntkId2):` The index of the second network.
	`(string miterNtkName):` The name of the resulting miter.
	`-Name:` Indicate the following token is the name of the miter.
	`-SEC:` Construct miter for sequential equivalence checking.
	`-CEC:` Construct miter for combinational equivalence checking. Please notice that latches are mapped by their names.
	`-Merge:` Merge miter outputs into a representative.

5 Design Manipulation Commands

Command:	`@CD [/	-	.	..	(Path Format)]`
Synopsis:	Change Design for Current Network.
Parameters:	Path Format: `[/(unsigned ntkID)][/(unsigned subModuleIndex)]*`

Command:	`@LN <(unsigned ntkID)> <(unsigned subModuleIndex)>`
Synopsis:	Link a Network with an instance of Current Network.
Parameters:	`(unsigned ntkID):` The index of a network.
	`(unsigned subModuleIndex):` The index of an instance of current network.

Command:	`@LS [(unsigned level)]`
Synopsis:	List Network Instances of Current Network.
Parameters:	`(unsigned level):` The number of levels of instances to be printed. (default = 1)

6 Intent Extraction Commands

Command:	`EXTract FSM [-Name <(string` *fsmName*`)>]`	
	`[-Output <(string` *outputIndex*`)>]`	
	`[-Time <(unsigned` *maxTime*`)>]`	
	`[-SCC | -NONE] [-CONFIRM]`	
Synopsis:	Extract Finite State Machines from Current Network.	
Parameters:	`(string` *fsmName*`):`	The name of FSMs to be extracted.
	`(string` *outputIndex*`):`	The index of output that represents the bad signal of a safety property.
	`(unsigned` *maxTime*`):`	The runtime limit for extraction.
	`-Time:`	Indicate that the following token is the maximum runtime limit.
	`-Output:`	Indicate that the following token is the index of an output.
	`-Name:`	Indicate that the following token is the name for FSMs.
	`-SCC:`	Cluster variables by strongly connected components.
	`-NONE:`	Disable clustering of variables.
	`-CONFIRM:`	Self checking if the extraction is successful.

Command:	`ELAborate FSM [-Input (string` *inputName*`)]`	
	`[-Name <(string` *fsmName*`)>]`	
	`[-Output <(string` *outputIndex*`)>] [-CONFIRM]`	
Synopsis:	Elaborate Network and Construct FSM from Input Specification.	
Parameters:	`(string` *inputName*`):`	The name of FSM specification input file.
	`(string` *fsmName*`):`	The name of FSMs to be extracted.
	`(string` *outputIndex*`):`	The index of output that represents the bad signal of a safety property.
	`-Input:`	Indicate that the following token is the name for the input file.
	`-Output:`	Indicate that the following token is the index of an output.
	`-Name:`	Indicate that the following token is the name for FSMs.
	`-CONFIRM:`	Self checking if the extraction is successful.

Command:	`PLOT FSM <(string` *fsmName*`)> <(string` *fsmDirName*`)>`	
Synopsis:	Plot Finite State Machines into *.png files.	
Parameters:	`(string` *fsmName*`):`	The name of FSMs to be plotted.
	`(string` *fsmDirName*`):`	The name of a directory for FSM outputs.

Command:	`WRIte FSM <(string` *fsmName*`)> <(string` *outputFile*`)>`	
Synopsis:	Output Finite State Machines Specifications.	
Parameters:	`(string` *fsmName*`):`	The name of FSMs to be plotted.
	`(string` *outputFile*`):`	The name of a file for FSM outputs.

7 Design Simulation Commands

Command:	`SIM NTk <(-Input <(string` *fileName*`)>) |`	
	`(-Random <(unsigned` *patterns*`)>)>`	
	`[-Output <(string outFileName)>] [-Event]`	
Synopsis:	Plot simulation or counterexample traces.	
Parameters:	`(string` *fileName*`):`	The file name of the input pattern file.
	`(unsigned` *patterns*`):`	The number of patterns for random simulation.
	`(string` *outFileName*`):`	The file name for simulation result output.
	`-Event:`	Enable event-driven simulation.
	`-Input:`	Enable simulation from input patterns.
	`-Random:`	Enable random simulation.
	`-Output:`	Enable dumping simulation results into a file.

Command:	`PLOt TRace <(string` *inputPatternFileName*`)>`	
	`<(string` *outputFileName*`)>`	
Synopsis:	Plot simulation or counterexample traces.	
Parameters:	`(string` *inputPatternFileName*`):`	The file name of the input pattern file.
	`(string` *outputFileName*`):`	The file name for simulation result output.

8 Design Verification Commands

Command:	`SET SAFEty [-Name <(string` *propertyName*`)>]`	
	`[(unsigned` *outputIndex*`)]`	
	`[-INVAriant <(string` *invName*`)*>]`	
	`[-INVConstraint <(string` *constrName*`)*>]`	
Synopsis:	Set Safety Properties on Current Network.	
Parameters:	`(string` *propertyName*`):`	The name of a property to be set.
	`(unsigned` *outputIndex*`):`	The index of a primary output serving as a bad signal.
	`(string` *invName*`)*:`	List of names of invariants.
	`(string` *constrName*`)*:`	List of names of (either invariant or fairness) constraints.
	`-Name:`	Indicate the following token is the name of a property.
	`-INVAriant:`	Indicate the starting of a list of invariants.
	`-INVConstraint:`	Indicate the starting of a list of invariant constraints.

Command:	`SET LIVEness [-Name <(string propertyName)>]` `[-INVAriant <(string invName)*>]` `[-INVConstraint <(string constrName)*>]` `[-FAIRnessConstraint <(string constrName)*>]`
Synopsis:	Set Liveness Properties on Current Network.
Parameters:	`(string propertyName):` The name of a property to be set.
	`(string invName)*:` List of names of invariants.
	`(string constrName)*:` List of names of (either invariant or fairness) constraints.
	`-Name:` Indicate the following token is the name of a property.
	`-INVAriant:` Indicate the starting of a list of invariants.
	`-INVConstraint:` Indicate the starting of a list of invariant constraints.
	`-FAIRnessConstraint:` Indicate the starting of a list of fairness constraints.

Command:	`ELAborate PRoperty [(string propertyName)]*`
Synopsis:	Elaborate Properties on a Duplicated Network.
Parameters:	`(string propertyName):` The name of a property.

Command:	`SET REport [-All] [-RESUlt] [-Endline] [-Solver] [-Usage]` `[-ON |-OFF |-RESET]`
Synopsis:	Set Verbosities for Verification Report.
Parameters:	`-RESET:` Reset everything to default.
	`-ON` Turn specified attributes on.
	`-OFF` Turn specified attributes off.
	`-All` Toggle all the following attributes.
	`-RESUlt` Toggle interactive verification status. (default = on)
	`-Endline` Toggle endline or carriage return. (default = off)
	`-Solver` Toggle solver information. (default = off)
	`-Usage` Toggle verification time usage. (default = on)
	`-Profile` Toggle checker specific profiling. (default = off)

Command:	`PRInt REport`
Synopsis:	Print Verbosities for Verification Report.

Command:	`SET SOlver [|-Default | -Minisat | -Boolector]`
Synopsis:	Set Active Solver for Verification.
Parameters:	`-Default:` Enable default solver. (default = minisat)
	`-MSat` Enable MiniSat as the active solver.
	`-Boolector` Enable Boolector as the active solver.

Command:	`PRInt SOlver`
Synopsis:	Print Active Solver for Verification.

Command:	`VERify SIM [(string` *propertyName*`)]`	
	`[<-Time (unsigned` *MaxTime*`)>]`	
	`[<-Cycle (unsigned` *MaxCycle*`)>]`	
Synopsis:	Perform (Constrained) Random Simulation.	
Parameters:	`(string` *propertyName*`):`	The name of a property to be verified.
	`(unsigned` *MaxTime*`):`	The upper bound of simulation runtime.
	`(unsigned` *MaxCycle*`):`	The upper bound of simulation cycle.
	`-Time:`	Enable setting of runtime limit.
	`-Cycle:`	Enable setting of cycle limit.

Command:	`VERify BMC [(string` *propertyName*`)]`	
	`[-Max-depth (unsigned` *MaxDepth*`)]`	
	`[-Pre-depth (unsigned` *PreDepth*`)]`	
	`[-Inc-depth (unsigned` *IncDepth*`)]`	
Synopsis:	Perform Bounded Model Checking [2].	
Parameters:	`(string` *propertyName*`):`	The name of a property to be verified.
	`(unsigned` *MaxDepth*`):`	The upper bound of time-frames to be reached. (default = 100)
	`(unsigned` *PreDepth*`):`	The number of frames at initial. (default = 0)
	`(unsigned` *IncDepth*`):`	The number of frames to be increased in each iteration. (default = 1)
	`-Max-depth:`	Indicate the following token is the time-frame limit.
	`-Pre-depth:`	Indicate the following token is the number of frames at initial.
	`-Inc-depth:`	Indicate the following token is the number of frames to be increased.

Command:	`VERify UMC [(string` *propertyName*`)]`	
	`[-Max-depth (unsigned` *MaxDepth*`)]`	
	`[-Pre-depth (unsigned` *PreDepth*`)]`	
	`[-Inc-depth (unsigned` *IncDepth*`)]`	
	`[-NOProve | -NOFire] [-Uniqueness]`	
Synopsis:	Perform Unbounded Model Checking [8].	
Parameters:	`(string` *propertyName*`):`	The name of a property to be verified.
	`(unsigned` *MaxDepth*`):`	The upper bound of time-frames to be reached. (default = 100)
	`(unsigned` *PreDepth*`):`	The number of frames at initial. (default = 0)
	`(unsigned` *IncDepth*`):`	The number of frames to be increased in each iteration. (default = 1)
	`-Max-depth:`	Indicate the following token is the time-frame limit.
	`-Pre-depth:`	Indicate the following token is the number of frames at initial.
	`-Inc-depth:`	Indicate the following token is the number of frames to be increased.
	`-NOProve:`	Disable running k-induction in UMC.
	`-NOFire:`	Disable performing bounded model checking (BMC) in UMC.
	`-Uniqueness:`	Enable adding uniqueness constraints.

Command:	`VERify ITP [(string` *propertyName*`)]`	
	`[-Max-depth (unsigned` *MaxDepth*`)]`	
	`[-Reverse] [-Increment] [-Force] [-RECycle]`	
	`[-Block (unsigned` *badCount*`)]`	
Synopsis:	Perform Interpolation-based Model Checking Algorithm *NewITP* [11].	
Parameters:	`(string` *propertyName*`):`	The name of a property to be verified.
	`(unsigned` *MaxDepth*`):`	The upper bound of time-frames to be reached. (default = 100)
	`(unsigned` *badCount*`):`	The maximum number of spurious cex for refinement. (default = 1)
	`-Max-depth:`	Indicate the following token is the limit of time-frames.
	`-Block:`	Indicate the following token is the limit to cex analysis.
	`-Reverse:`	Enables the reversed implementation of *NewITP*.
	`-Increment:`	Enables incrementing BMC depth dynamically.
	`-Force:`	Enables considering 1 k frames (instead of the k-th frame) in the BMC part.
	`-RECycle:`	Enables cube recycling for interpolant reuse.

Command:	`VERify PDR [(string` *propertyName*`)]`	
	`[-Max-depth (unsigned` *MaxDepth*`)]`	
	`[-Recycle (unsigned MaxCount)] [-Incremental]`	
Synopsis:	Perform Property Directed Reachability [4, 7].	
Parameters:	`(string` *propertyName*`):`	The name of a property to be verified.
	`(unsigned` *MaxDepth*`):`	The upper bound of time-frames to be reached. (default = 100)
	`(unsigned` *MaxCount*`):`	The upper bound of temporary assumption literals in solvers. (default = ∞)
	`-Max-depth:`	Indicate the following token is the limit of time-frames.
	`-Recycle:`	Enable setting the limit of assumption literals for recycle.
	`-Incremental:`	Implement with multiple solvers. (c.f. Monolithic)

Command:	`VERify SEC [(string` *propertyName*`)]`	
	`[-Max-depth (unsigned` *MaxDepth*`)]`	
	`[-UMC | -IPDR | -MPDR] [-CEC] [-SEC]`	
Synopsis:	Perform Sequential Equivalence Checking [9].	
Parameters:	`(string` *propertyName*`):`	The name of a property to be verified.
	`(unsigned` *MaxDepth*`):`	The upper bound of time-frames to be reached. (default = 100)
	`-Max-depth:`	Indicate the following token is the time-frame limit.
	`-UMC:`	Enable UMC as a safety checker.
	`-MPDR:`	Enable Monolithic PDR as a safety checker.
	`-IPDR:`	Enable Incremental PDR as a safety checker.
	`-CEC:`	Assume that the Network could be a CEC Miter.
	`-SEC:`	Assume that the Network could be a SEC Miter.

Command:	`VERify KLIVE [(string` *propertyName*`)]`	
	`[-Max-depth (unsigned` *MaxDepth*`)]`	
	`[-Inc-depth (unsigned` *IncDepth*`)]`	
	`[|-UMC | -IPDR | -MPDR]`	
Synopsis:	Perform K-Liveness [6] for Liveness Checking.	
Parameters:	`(string` *propertyName*`):`	The name of a property to be verified.
	`(unsigned` *MaxDepth*`):`	The upper bound of time-frames to be reached. (default = 100)
	`(unsigned` *IncDepth*`):`	The number of frames to be increased in each iteration. (default = 1)
	`-Max-depth:`	Indicate the following token is the time-frame limit.
	`-Inc-depth:`	Indicate the following token is the number of frames to be increased.
	`-UMC:`	Enable UMC as a safety checker.
	`-MPDR:`	Enable Monolithic PDR as a safety checker.
	`-IPDR:`	Enable Incremental PDR as a safety checker.

Command:	`CHEck REsult <(string` *propertyName*`)> [-Simulation | -Formal]`	
	`[[-Trace | -Invariant] <(string` *resultFileName*`)>]`	
Synopsis:	Verify Verification Result.	
	Note: Confirmation of Inductive Invariants is Not Available Yet !!	
Parameters:	`(string` *propertyName*`):`	The name of a verified property.
	`(string` *resultFileName*`):`	The file name of a verification result.
	`-Simulation:`	Enable simulation in verifying the result.
	`-Formal:`	Enable formal in verifying the result.
	`-Trace:`	Indicate resultFileName is a file of counterexample.
	`-Invariant:`	Indicate resultFileName is a file of inductive invariant.

Command:	`PLOt REsult <(string` *propertyName*`)> <(string` *resultFileName*`)>`	
Synopsis:	Elaborate Properties on a Duplicated Network.	
Parameters:	`(string` *propertyName*`):`	The name of a verified property.
	`(string` *resultFileName*`):`	The file name for the output of verification results.

Command:	`WRIte REsult <(string` *propertyName*`)> <(string` *resultFileName*`)>`	
Synopsis:	Write Verification Results.	
Parameters:	`(string` *propertyName*`):`	The name of a verified property.
	`(string` *resultFileName*`):`	The file name for the output of verification results.

9 V3 Model Checking Commands

Command:	`READ PROperty <(string` *fileName*`)> <-Aiger | -Prop>`	
Synopsis:	Read property specification from external file.	
Parameters:	`(string` *fileName*`):`	The file name of the property input.
	`-Aiger:`	Indicate fileName is an AIGER input.
	`-Prop:`	Indicate fileName is a PROP input (build on top of current ntk).

Command:	`WRITE PROperty <(string` *fileName*`)> <-Aiger | -Prop>`	
Synopsis:	Write property specification into file.	
Parameters:	`(string` *fileName*`):`	The file name of the property output.
	`-Aiger:`	Output network and properties into AIGER format.
	`-Prop:`	Output properties into PROP format (in terms of current network).

Command:	`RUN <-TIMEout (unsigned` *maxTime*`)>`	
	`<-MEMoryout (unsigned` *maxMemory*`)>`	
	`<-THReadout (unsigned` *maxThread*`)>`	
Synopsis:	Run $\mathcal{V}_3$ Model Checker. (see also my PhD thesis for detailed descriptions)	
Parameters:	`(unsigned` *maxTime*`):`	Wall Timeout limit in seconds.
	`(unsigned` *maxMemory*`):`	Memoryout limit in Mega Bytes.
	`(unsigned` *maxThread*`):`	The number of available CPU cores.
	`-TIMEout:`	Indicate the next number is the timeout limit.
	`-MEMoryout:`	Indicate the next number is the memoryout limit.
	`-THReadout:`	Indicate the next number is the CPU core limit.

10 Design Debugging Commands

Command:	`OPTimize TRace <(string propertyName)>`	
	`[-NOReduce | -NOGeneralize]`	
Synopsis:	Optimize a Counterexample Trace.	
Parameters:	`(string` *propertyName*`):`	The name of a failing property.
	`-NOReduce:`	Disable counterexample reduction.
	`-NOGeneralize:`	Disable counterexample generalization.

Command:	`SIMplify TRace <(string propertyName)>`	
	`[(unsigned maxNoFrames)]`	
	`[| -Care | -Transition]`	
Synopsis:	Simplify Counterexample Traces.	
Parameters:	`(string` *`propertyName`*`):`	The name of a fired property.
	`(unsigned maxNoFrames):`	Upper bound frame numbers in a subproblem. (default = ∞)
	`(string` *`constrName`*`)*:`	List of names of (either invariant or fairness) constraints.
	`-Care:`	Start the configuration of minimizing care signals.
	`-Transition:`	Start the configuration of minimizing transitions.

References

[1] A. Biere. The AIGER And-Inverter Graph (AIG) Format. *Available at fmv. jku. at/aiger*, 2007.

[2] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic Model Checking without BDDs. *Tools and Algorithms for the Construction and Analysis of Systems*, pages 193–207, 1999.

[3] A. Biere, K. Heljanko, and S. Wieringa. AIGER 1.9 and Beyond. *Available at fmv.jku.at/hwmcc11/beyond1.pdf*, 2011.

[4] A. Bradley. SAT-based Model Checking without Unrolling. In *Verification, Model Checking, and Abstract Interpretation*, pages 70–87. Springer, 2011.

[5] R. Brummayer, A. Biere, and F. Lonsing. BTOR: Bit-precise Modelling of Word-level Problems for Model Checking. In *Proceedings of the Joint Workshops of the 6th International Workshop on Satisfiability Modulo Theories and 1st International Workshop on Bit-Precise Reasoning*, pages 33–38. ACM, 2008.

[6] K. Claessen and N. Sörensson. A Liveness Checking Algorithm that Counts. In *FMCAD*, pages 52–59, 2012.

[7] N. Een, A. Mishchenko, and R. Brayton. Efficient Implementation of Property Directed Reachability. In *Formal Methods in Computer-Aided Design (FMCAD), 2011*, pages 125–134. IEEE, 2011.

[8] N. Eén and N. Sörensson. Temporal Induction by Incremental SAT Solving. *Electronic Notes in Theoretical Computer Science*, 89(4):543–560, 2003.

[9] A. Mishchenko, M. Case, R. Brayton, and S. Jang. Scalable and Scalably-verifiable Sequential Synthesis. In *Computer-Aided Design, 2008. ICCAD 2008. IEEE/ACM International Conference on*, pages 234–241. IEEE, 2008.

[10] D. Thomas and P. Moorby. *The Verilog® Hardware Description Language*, volume 2. Springer, 2002.

[11] C.-Y. Wu, C.-A. Wu, C.-Y. Lai, and C.-Y. Huang. A Counterexample-guided Interpolant Generation Algorithm for SAT-based Model Checking. In *Design Automation Conference (DAC), 2013 50th ACM/EDAC/IEEE*, pages 1–6. IEEE, 2013.

[12] H.-H. Yeh, C.-Y. Wu, and C.-Y. Huang. QuteRTL: Towards an Open Source Framework for RTL Design Synthesis and Verification. *Tools and Algorithms for the Construction and Analysis of Systems*, pages 377–391, 2012.