

## **VhdlCohen Publishing** & Consulting

http:www.vhdlcohen.com/ VhdlCohen@aol.com

## PSL/SUGAR QUICK REFERENCE CARD **VHDL**

From book: Using PSL/SUGAR with Verilog and VHDL, Guide to Property Specification Language for Assertion-Based Verification, Ben Cohen, 2003, isbn 0-9705394-4-4

| Rev A 5/30/03  |  |  |  |
|--|--|--|--|
| Property Format  | Example with embedded PSL  |  |  |
| <pre>property <name> = [operator] [enabling_condition(s)]</name></pre>                       | •  |  |  |
|  | psl property REQ_ACK_IN_4_CYCLES is  |  |  |
|  | <b>always</b> ({req and not ack}  =>{[*0 to 3]; ack} <b>abort</b>                            |  |  |
| [@(clock_expression)];   | reset_n='0') @(clk'event and clk='1');   |  |  |
|  | <b>never</b> (push and fifo_full and not pop);   |  |  |
|  | Implication_operators  |  |  |
| Operator: always   never   | -> RHS started when LHS is true. RHS is Boolean.   |  |  |
| Enabling condition: Boolean expression   | -> next RHS evaluated in next cycle after LHS true. RHS is Boolean.                          |  |  |
| Implication_operator: Relationship between   | -> RHS started in last cycle of the LHS true. LHS is a sequence {}                           |  |  |
| expressions  | => RHS started following cycle LHS condition true. LHS is a                                  |  |  |
| Fulfilling condition Tested behavior. Checked every  | sequence {}. Shorthand for  -> {true;  |  |  |
| verification cycle. Boolean expression or sequence. Last                                     | eventually! RHS is true in some future cycle, and must be true before                        |  |  |
| expression prior to any discharging condition.   | the end of simulation. RHS is Boolean or a sequence  |  |  |
|  | Discharging conditions   |  |  |
| clock expression When to sample the assertion.   | until(_) Fulfilling condition must hold until the expression is true.                        |  |  |
| Generally a clock edge, but can be any Boolean   | abort Cancels checking of an assertion.  |  |  |
| ·  | Default clock  |  |  |
|  | The clock default clock is (clk'event and clk='1');  |  |  |
|  | <b>property</b> NeverRdWrBothActive = <b>never</b> (read and write);                         |  |  |
| SERE Operators   | Examples   |  |  |
|  | always ({go; req}  => {not req and ack; data_transfer});                                     |  |  |
| $\{a; b; c\}$ a at cycle t, b at t+1, c at t+2 c is true.                                    | always ({go, req}  -> {not req and ack, data_transfer}),                                     |  |  |
| [* ] Consecutive repetition  |  |  |  |
| -  | <b>always</b> ({a}  => {b[*2]; c}); if $a$ , then $b$ ; $b$ ; $c$ sequence                   |  |  |
| [*n] Repeats for n cycles  | <b>always</b> ({a}  => {[*2]; c}); if a, then -; -; c sequence                               |  |  |
| [*] Repeats for zero or any number of cycles   | <b>always</b> ( $\{a\} \mid => \{b[*1 \text{ to } 3]; c\}$ ); if a then either of sequences: |  |  |
| [+] Repeats for one or more cycles   | b; c   b; b; c   b; b; b; c  |  |  |
| [*n:m] Repeats for min of n to max of m cycles   | <b>always</b> ( $\{a\} \mid => \{b[*0 \text{ to } 3]; c\}$ ); if a then either of sequences: |  |  |
| [*n:inf] Repeats for a minimum of n cycles [*0: m] Either skipped or repeats max of m cycles | c / b; c   b; b; c   b; b; b; c  |  |  |
| [10. III] Either skipped of repeats max of in cycles   | <b>always</b> ( $\{a\} \mid => \{b[+]; c\}$ ); if $a$ then either of sequence                |  |  |
|  | b; c   b; b; c   b; b;; b; c   |  |  |
|  | <b>always</b> ( $\{a\} \mid => \{[*]; b; c\}$ ); Cannot fail. For functional coverage        |  |  |
| I – 1 Non consecutive repetition   | if $a$ then assertion completes when $b$ ; $c$ sequence occurs.                              |  |  |
| [= ] Non-consecutive repetition  | <b>always</b> ( $\{a\} \mid => \{b[=2]; c\}$ ); If a then two occurrence of b before c;      |  |  |
| [->] GOTO repetition   | <b>always</b> ( $\{a\} = \{b[->2]; c\}$ ); If a then two occurrence of b any cycle           |  |  |
|  | before c, but last b must occur in cycle before c;   |  |  |
|  | Grant must always occur sometime after req.  |  |  |
|  | always ((req) -> eventually! (grant);  |  |  |
|  | always ((req) -> eventually! {grant; ok});   |  |  |
| Until  |  |  |  |
|  | always (( $\{a; b\}   => \{c\}$ ) until (rst));  |  |  |
|  | always ((state = S1) -> next ((state = S2) until (state=S3)));                               |  |  |
|  | always( req -> eventually! ack until data_xfr;   |  |  |
|  | always ((state = S1) -> next (state = S2) -> eventually! (state=S3)                          |  |  |
| -> eventually! Boolean until Boolean   | until done);   |  |  |
| · ·  | always( req -> eventually! ack -> eventually! data_xfr until done);                          |  |  |
| -> eventually! Boolean until Boolean   |  |  |  |
| ({ SERE}  => { SERE}) until Boolean  | <b>always</b> (({go; req}  => {!req && ack; data_xfr}) <b>until</b> done);                   |  |  |

| sequence fusion. Two sequences overlap by one cycle   sequence disjunction. One of two alternative sequences   hold at the current cycle. Remained to the current cycle. Repartment of the current cycle, repartless of whether they complete in same cycle or in different cycles. Reference onjunction. Two sequences both hold at the current cycle, and both complete in the same cycle.    Named sequences   Define common sequences by name.   Creates more readable and reusable code   Page sequence Col.   Length-Matching (a; [*]; b) & & (c[*1 to 5]; d) ↑  | Sequence composition operators          |   |  |  |  |  |
|--|---|---|--|--|--|--|
| sequence disjunction   |   |   |  |  |  |  |
| ## non-length-matching sequence conjunction. Two sequences both hold at the current cycle, regardless of whether they complete in same cycle or in different cycles.  ### Residual in the same cycle.    Named sequences Define common sequences by name.  |   |   |  |  |  |  |
| sequences both hold at the current cycle, regardless of whether they complete in same cycle or in different cycles. & length-matching sequence conjunction. Two scquences both hold at the current cycle, and both complete in the same cycle.  Named sequences  Define common sequences by name.  Creates more readable and reusable code    Page   |   |   |  |  |  |  |
| whether they complete in same eyele or in different cycles.  & Reight-matching sequence conjunction. Two sequences both hold at the current cycle, and both complete in the same cycle.    Named sequences   Define common sequences by name.  |   |   |  |  |  |  |
| ## Assertions in external files    Verification Unit   |   | <b>↑</b> Fusion {a; b} : {c; d}   |  |  |  |  |
| Named sequences by name. Creates more readable and reusable code   |   | ↑Sequence Disjuction {a: b}   {c: d}                                      |  |  |  |  |
| In the same cycle.   Length-Matching {a; {*}; b} & & {cf**1 to 5}; d} ↑   ↑  |   |   |  |  |  |  |
| Named sequences  |   | Non-Length-Matching $\{a; [*]; b\} & \{c[*] \text{ to } 5]; d\} \uparrow$ |  |  |  |  |
| Define common sequences by name.   Pals sequence AB is {a; b};   | in the same cycle.                      | Length-Matching {a; [*]; b} && {c[*1 <b>to</b> 5]; d} ♠                   |  |  |  |  |
| Define common sequences by name.   Pals sequence AB is {a; b};   |   |   |  |  |  |  |
| Creates more readable and reusable code  |   |   |  |  |  |  |
| - psl sequence AB_EV is {a;  *i;  *b}; - psl sequence CD_LIMITED is {c *e1 to 5; cl}; - psl sequence EMBEDDED is {c *e1 to 5; cl}; - psl sequence EMBEDDED is {c *e1 to 5; cl}; - psl property TEST I is - always ({go})  > {AB, CD, LIMITED}); - psl property FUSION TEST is - always ({go})  > {AB, Psl}; {CD *e2}}; - psl property FUSION TEST is - always ({go})  > {AB, Psl}; {CD *e2}}; - psl property SUSION TEST is - always ({go})  > {AB, Psl}; {CD *e2}}; - psl property SUSION TEST is - always ({go})  > {AB, Psl}; {CD *e2}}; - psl property SUSION TEST is - always ({go})  > {AB, Psl}; {CD *e2}}; - psl property SUSION TEST is - always ({go})  > {AB, Psl}; {CD *e2}}; - psl property TEST is - always ({go})  > {AB, Psl}; {CD *e2}}; - psl property Interest is - always ({go})  > {AB, Psl}; {CD *e2}}; - assertions for abe if vunit entity - architecture name: test - vunit abe if vun (abe if vunit(test)) { - assertions for abe if vunit entity - default clock is (elkevent and elk = '0'); - property SEBC_TEST is - always (a > (a) varys {i!* to 10}; b)); - property SEBC_TEST is - always (a > (a) varys {i!* to 10}; b)); - property ABCD_IF is - always (a > b > c > d); - interest is always (a > b > c > d); - inte  |   |   |  |  |  |  |
| - psl sequence CD_LIMITED is {d*1 to 5}; d); - psl sequence EMBEDDED is {AB; CD}; - psl property TEST1 is - always {go}  > {AB; CD_LIMITED}); - psl property TEST0 is - always {go}  >> {AB; CD_LIMITED}); - psl property DISUICTION_TEST is - always {go}  >> {AB; CD_LIMITED}); - psl property DISUICTION_TEST is - always {go}  >> {AB; CD_LIMITED}); - psl property DISUICTION_TEST is - always {go}  >> {AB; CD_LIMITED}); - psl property SEQUENCE_NON_MATCH_TEST is - always {go}  >> {AB; EV} & {CD_LIMITED}); - psl property SEQUENCE_NON_MATCH_TEST is - always {go}  >> {AB; EV} & {CD_LIMITED}); - psl property SEQUENCE_NON_MATCH_TEST is - always {go}  >> {AB; EV} & {CD_LIMITED}); - psl property SEQUENCE_NON_MATCH_TEST is - always {go}  >> {AB; EV} & {CD_LIMITED}); - architecture name: test - vunit abc_if_v unit entity - default clock is {clk event and clk = "0"}; - property SERE_TEST1 is - always {a>- kase io_nvalue} {always {a} {always | Creates more readable and reusable code |   |  |  |  |  |
| - psl sequence EMBEDDED is {AB; CD}; - psl property TEST1 is - always (go)  > {AB; CD_LIMITED ); - psl property PUSION_TEST is - always (go)  > {AB   CD}; - psl property DISIOUTION_TEST is - always (go)  > {AB   CD}; - psl property SEQUENCE_NON_MATCH_TEST is - always (go)  > {AB   CD}; - psl property SEQUENCE_NON_MATCH_TEST is - always (go)  > {AB   CD}; - psl property SEQUENCE_NON_MATCH_TEST is - always (go)  > {AB   CD}; - psl property SEQUENCE_NON_MATCH_TEST is - always (go)  > {AB   CD}; - psl property SEQUENCE_NON_MATCH_TEST is - always (go)  > {AB   CD}; - psl property SEQUENCE_NON_MATCH_TEST is - always (go)  > {AB   CD}; - psl property SEQUENCE_NON_MATCH_TEST is - always (go)  > {AB   CD}; - psl property SEQUENCE_NON_MATCH_TEST is - always (go)  > {AB   CD}; - psl property SEQUENCE_NON_MATCH_TEST is - always (go)  > {AB   CD}; - psl property SEQUENCE_NON_MATCH_TEST is - always (go)  >> {AB   CD}; - psl property SEQUENCE_NON_MATCH_TEST is - always (go)  >> {AB   CD}; - psl property SEQUENCE_NON_MATCH_TEST is - always (go)  >> {AB   CD}; - psl property SEQUENCE_NON_MATCH_TEST is - always (go)  >> {AB   CD}; - psl property SEQUENCE_NON_MATCH_TEST is - always (go)  >> {AB   CD}; - psl property SEQUENCE_NON_MATCH_TEST is - always (go)  >> {AB   CD}; - psl property SEQUENCE_NON_MATCH_TEST is - always (go)  >> {AB   CD}; - psl property SEQUENCE_NON_MATCH_TEST is - always (go)  >> {AB   CD}; - psl property SEQUENCE_NON_MATCH_TEST is - always (go)  >> {AB   CD}; - psl property Sequence in slow of the spoint slow of the slow of the spoint slow of the slow of the spoint slow of the spoint slow of the spoint slow of the slow of the spoint slow of the spoint slow of the slow of t  |   |   |  |  |  |  |
| $ \begin{array}{c ccccccccccccccccccccccccccccccccccc$   |   |   |  |  |  |  |
| - always ([go]  -> {AB; CD_LIMITED}); psl property FUSION_TEST is always ([go]  -> {AB ^2}] \}; (CD[^22]); psl property SEQUENCE_NON_MATCH_TEST is always ([go]  -> {AB ^2}] \}; (CD[^22]); psl property SEQUENCE_NON_MATCH_TEST is always ([go]  -> {AB ^2}] \}; (CD[^22]); psl property SEQUENCE_NON_MATCH_TEST is always ([go]  -> {AB_EV}] & {CD_LIMITED});  entity name: abe_if_vunit (rest) entity name: abe_if_vunit (rest) \} entity name: name: test entity name: abe_if_vunit (rest) \} entity name: abe_if_vunit (rest) \} entity name:   |   |   |  |  |  |  |
| - psl property FUSION_TEST is - always ({go}  => {AB[*2]} : {CD[*2]}); - psl property DISIUCITION_TEST is - always ({go}  => {AB}   {CD}; - psl property DISIUCITION_TEST is - always ({go}  => {AB}   {CD}; - psl property DISIUCITION_TEST is - always ({go}  => {AB}   {CD}; - psl property DISIUCITION_TEST is - always ({go}  => {AB}   {CD}; - psl property DISIUCITION_TEST is - always ({go}  => {AB}   {CD}; - psl property DISIUCITION_TEST is - always ({go}  => {AB}   {CD}   {CD}   {LIMITED});  - entity name: abc_if_yunit - architecture name: test vunit abc_if_yu (abc_if_yunit entity) default clock is (clk'event and clk = '0'); property SERE_TEST1 is always ((a) -> (always {*1 to 10}; b))); property SERE_TEST1 is always ((a) -> (always {*1 to 10}; b))); property ABCD_IEST is always ((a) -> next b -> next c -> next d); property ABCD_IEST is always ((a) -> beter or next d); property ABCD_IEST is always ((a) -> beter or next d); property ABCD_IEST is always ((a) -> beter or next d); property ABCD_IEST is always ((a) -> late or next d); property ABCD_IEST is always ((a) -> next b -> next c -> next d); property ABCD_IEST is always ((a) -> late or next d); property ABCD_IEST is always ((a) -> late or next d); property ABCD_IEST is always ((a) -> late or next d); property and assumed property both hold. Initialize design to get to a specific state before checking assertions. Constrain design so sequence holds and verify restrict sequence holds. Cover Sequence ; Check if a certain path was covered by the verification space fairness Boolean ; Clarifies properties of specifications. Great for requirements review. A path is fair if every fairness constraint holds along the path.  PSL in Design Process  • Requirements • Synthesizable HDL • Testbench • Integration into application • Verification • Verification • Documentation • Properties and simulation with assertion metrics enhance documentation. Properties and simulation with assertion metrics enhance documentation.   |   |   |  |  |  |  |
| - always ([go]  => [AB]*2]  : (CD]*2]  ); - psl property DISJUCTION_TEST is - always ([go]  => [AB]*2]  : (CD]*2]  ); - psl property SEQUENCE.NON_MATCH_TEST is - always ([go]  => [AB]*2]   (CD); - psl property SEQUENCE.NON_MATCH_TEST is - always ([go]  => [AB]*2]   (CD_LIMITED));  - rentity name: abc_if_vunit - architecture name: test - vunit abc_if_vunit (test) { - assertions for abc_if_vunit entity - architecture name: test - assertions for abc_if_vunit entity - architecture name: test - assertions for abc_if_vunit entity - architecture name: test - assertions for abc_if_vunit entity - assertions for abc_if_vunit(est)} - assertions distance and is always (a) - assertion entity - assertions for abc_if_vunit(est)} - assertions for abc_if_vunit(est)} - assertions for abc_if_vuni |   |   |  |  |  |  |
| Psi property DISJUCTION_TEST is  |   |   |  |  |  |  |
| Psi property SEQUENCE_NON_MATCH_TEST is   always ((go)  => {AB_EV} & {CD_LIMITED});   Verification Unit   entity name: abc if vunit   architecture name: abc if vunit entity   default clock is (clk'event and clk = '0');   property SERE_TEST1 is   always ((a) >- (always) {**1 to 10}; b}));   property ABCD_NEXT is   always (a >- bext c >- next d);   property ABCD_IF is   always (a >- b >- c >- d);  |   |   |  |  |  |  |
| Verification Unit       always ([go]  => {AB_EV} & {CD_LIMITED};         Assertions in external files       entity name: abc_if_vunit         architecture name: test       assertions for abc_if_vunit(test)) {   |   | always ({go}  => {AB}   {CD});  |  |  |  |  |
| Verification Unit  |   |   |  |  |  |  |
| Assertions in external files  - architecture name: test  vunit abc_if_vu (abc_if_vunit (test)) { - assertions for abc_if_vunit entity  default clock is (clk'event and clk = 0'); property SERE_TEST1 is always ((a) > (always {{*1 to 10}; b})); property ABCD_NEXT is always (a > hext b > next c > next d); property ABCD_IF is always (a > b > c > d); }  Verification directives assert Property; assume Property; Constrain verification (e.g., input behavior) so that a property holds. restrict Sequence; Property and assumed property both hold. Initialize design to get to a specific state before checking assertions. Constrain design so sequence holds and verify restrict sequence holds. cover Sequence; Constrain design so sequence holds and verify restrict sequence holds. Constrain path was covered by the verification space fairness Boolean; Guide to verify the property only over fair paths.  strong fairness Boolean; A path is fair if every fairness constraint holds along the path.  PSL in Design Process  PSL in Design Process Synthesizable HDL Testbench Synthesizable HDL Testbench Testchematics Testchematics Testchematics Testchematics Testchematics Testchematics Testchematics Testchematics Tes     |   |   |  |  |  |  |
| vunit abc_if_vu (abc_if_vunit(test)) {   |   |   |  |  |  |  |
| - assertions for abc_if_vunit entity default clock is (clk'event and clk = '0'); property SERE_TEST1 is always ((a) -> (always { *1 to 10 ; b})); property ABCD_NEXT is always (a -> next b -> next c -> next d); property ABCD_IF is always (a -> b -> c -> d); }  Verification directives assert Property; assume Property; Constrain verification (e.g., input behavior) so that a property holds. Property and assumed property both hold. Initialize design to get to a specific state before checking assertions. Constrain design so sequence holds and verify restrict sequence holds. Cover Sequence; Constrain design so sequence holds and verify restrict sequence holds. Cover Sequence; Constrain design so sequence holds and verify restrict sequence holds. Check if a certain path was covered by the verification space Guide to verify the property only over fair paths. strong fairness Boolean; A path is fair if every fairness constraint holds along the path.  PSL in Design Process  Requirements Synthesizable HDL Testbench Documents design implementation properties. Great for code reviews Facilitates TB designs. Eases uncertainties in microcycle timing of DUT. Detects errors during design integration. Detects errors, white-box verification. Provides functional verification. Properties and simulation with assertion metrics enhance documentation.  | Assertions in external files            |   |  |  |  |  |
| default clock is (clk'event and clk = '0');   property SERE_TEST1 is     always ((a) -> (always {[*1 to 10]; b}));   property ABCD_NEXT is     always (a -> next b -> next c -> next d);   property ABCD_IF is     always (a -> b -> c -> d);   makes always (a -> b -> c -> d);   makes always (a -> b -> c -> d);   makes always (a -> b -> c -> d);   makes always (a -> b -> c -> d);   makes always (a -> b -> c -> d);   makes always (a -> b -> c -> d);   makes always (a -> b -> c -> d);   makes always (a -> b -> c -> d);   makes always (a -> b -> c -> d);   makes always (a -> b -> c -> d);   makes always (a -> next b -> next c -> next d);   property ABCD_IF is     always (a -> next b -> next c -> next d);   property ABCD_IF is     always (a -> next b -> next c -> next d);   property alc D_IF is     always (a -> next b -> next c -> next d);   property alc D_IF is     always (a -> next b -> next c -> next d);   property alc D_IF is     always (a -> next b -> next c -> next d);   property alc D_IF is     always (a -> next b -> next c -> next d);   property alc D_IF is     always (a -> next b -> next c -> next d);   property alc D_IF is     always (a -> next b -> next c -> next d);   property alc D_IF is     always (a -> next b -> next c -> next d);   property alc D_IF is     always (a -> next b -> next c -> next d);   property alc D_IF is     always (a -> next c -> next d);   property alc D_IF is     always (a -> next c -> next d);   property alc D_IF is     always (a -> next c -> next d);   property alc D_IF is     always (a -> next c -> next d);   property alc D_IF is     always (a -> next c -> next d);   property alc D_IF is     always (a -> next c -> next d);   property alc D_IF is     always (a -> next c -> next d);   property alc  |   |   |  |  |  |  |
| property SERE_TEST1 is always ((a) -> (always { *1 to 10 ; b})); property ABCD_NEXT is always (a -> hext b -> hext c -> hext d); property ABCD_IF is always (a -> b -> c -> d); }    Verification directives   |   |   |  |  |  |  |
| Always ((a) -> (always [{*1 to 10]; b}));   property ABCD_NEXT is     always (a -> next b -> next c -> next d);   property ABCD_IF is     always (a -> b -> c -> d);   |   |   |  |  |  |  |
| property ABCD_NEXT is always (a -> next b -> next c -> next d); property ABCD_IF is always (a -> b -> c -> d); }  Verification directives assert Property; Assume Property; Constrain verification (e.g., input behavior) so that a property holds. Property and assumed property both hold. Initialize design to get to a specific state before checking assertions. Constrain design so sequence holds and verify restrict sequence holds. Cover Sequence; Constrain design so sequence holds and verify restrict sequence holds. Check if a certain path was covered by the verification space Guide to verify the property only over fair paths. A path is fair if every fairness constraint holds along the path.  PSL in Design Process  Requirements Synthesizable HDL Synthesizable HDL Testbench Integration into application Verification Verification Documentation Properties and simulation with assertion metrics enhance documentation.   |   |   |  |  |  |  |
| always (a > next b -> next c -> next d); property ABCD_IF is always (a -> b -> c -> d); }  Verification directives assert Property; Assume Property and assumed property both hold. Assume property and assumed property both hold.  Restrict Sequence; Assume Sequence; Assume Property and assumed property both hold.  Constrain design so sequence holds and verify restrict sequence holds.  Check if a certain path was covered by the verification space  Guide to verify the property only over fair paths.  A path is fair if every fairness constraint holds along the path.  PSL in Design Process  Requirements A path is fair if every fairness constraint holds along the path.  PSL in Design Process  Requirements A path is fair if every fairness constraint properties. Great for code reviews A path is fair if every fairness constraint in microcycle timing of DUT.  Documents design implementation properties. Great for code reviews  Facilitates TB designs. Eases uncertainties in microcycle timing of DUT.  Detects errors during design integration.  Detects errors, white-box verification. Provides functional verification.  Properties and simulation with assertion metrics enhance documentation.   |   |   |  |  |  |  |
| property ABCD_IF is always (a -> b -> c -> d); }  Verification directives assert Property;  assume Property;  constrain verification (e.g., input behavior) so that a property holds. Property and assumed property both hold. Initialize design to get to a specific state before checking assertions.  cover Sequence;  constrain design so sequence holds and verify restrict sequence holds.  cover Sequence;  check if a certain path was covered by the verification space  fairness Boolean;  strong fairness Boolean, Boolean;  PSL in Design Process  ■ Requirements ■ Synthesizable HDL ■ Testbench ■ Integration into application ■ Verification ■ Documentation  ■ Documentation  Properties and simulation with assertion metrics enhance documentation.  Properties and simulation with assertion metrics enhance documentation.   |   | always (a -> next b -> next c -> next d);                                 |  |  |  |  |
| Verification directives assert Property; Assume Property; Assume_guarantee Property only only only only only only only onl   |   |   |  |  |  |  |
| assume Property;   |   | always $(a \rightarrow b \rightarrow c \rightarrow d)$ ;                  |  |  |  |  |
| assume Property;   |   |   |  |  |  |  |
| assume Property;   |   | }   |  |  |  |  |
| assume Property; assume_guarantee Property; Property and assumed property both hold. Initialize design to get to a specific state before checking assertions.  restrict_guarantee Sequence; Constrain design so sequence holds and verify restrict sequence holds.  cover Sequence; Check if a certain path was covered by the verification space Guide to verify the property only over fair paths.  A path is fair if every fairness constraint holds along the path.  PSL in Design Process  Requirements Synthesizable HDL Synthesizable HDL Documents design implementation properties. Great for code reviews Facilitates TB designs. Eases uncertainties in microcycle timing of DUT. Detects errors during design integration. Verification Verification Documentation Properties and simulation with assertion metrics enhance documentation.   |   | W. 'C   |  |  |  |  |
| assume_guarantee Property;  restrict Sequence;  restrict_guarantee Sequence;  cover Sequence;  fairness Boolean;  strong fairness Boolean, Boolean;  PSL in Design Process  Requirements  Synthesizable HDL  Testbench  Integration into application  Verification  Documentation  Property and assumed property both hold.  Initialize design to get to a specific state before checking assertions.  Constrain design so sequence holds and verify restrict sequence holds.  Check if a certain path was covered by the verification space  Guide to verify the property only over fair paths.  A path is fair if every fairness constraint holds along the path.  Clarifies properties of specifications. Great for requirements review.  Documents design implementation properties. Great for code reviews  Facilitates TB designs. Eases uncertainties in microcycle timing of DUT.  Detects errors during design integration.  Detects errors, white-box verification. Provides functional verification.  Properties and simulation with assertion metrics enhance documentation.   |   |   |  |  |  |  |
| restrict Sequence; Initialize design to get to a specific state before checking assertions.  Constrain design so sequence holds and verify restrict sequence holds.  Cover Sequence; Check if a certain path was covered by the verification space Guide to verify the property only over fair paths.  A path is fair if every fairness constraint holds along the path.  PSL in Design Process  Requirements Synthesizable HDL Synthesizable HDL Testbench Integration into application Verification Verification Documentation Properties and simulation with assertion metrics enhance documentation. Properties and simulation with assertion metrics enhance documentation.   |   |   |  |  |  |  |
| restrict_guarantee Sequence;   |   |   |  |  |  |  |
| cover Sequence; Check if a certain path was covered by the verification space  fairness Boolean; Guide to verify the property only over fair paths.  A path is fair if every fairness constraint holds along the path.  PSL in Design Process  Requirements Synthesizable HDL Synthesizable HDL Testbench Integration into application Verification Verification Documentation Provides functional verification. Properties and simulation with assertion metrics enhance documentation. Properties and simulation with assertion metrics enhance documentation.   |   |   |  |  |  |  |
| fairness Boolean;Guide to verify the property only over fair paths.strong fairness Boolean, Boolean;A path is fair if every fairness constraint holds along the path.PSL in Design Process• Requirements. Clarifies properties of specifications. Great for requirements review.• Synthesizable HDL. Documents design implementation properties. Great for code reviews• Testbench. Facilitates TB designs. Eases uncertainties in microcycle timing of DUT.• Integration into application. Detects errors during design integration.• Verification. Detects errors, white-box verification. Provides functional verification.• Documentation. Properties and simulation with assertion metrics enhance documentation.   |   |   |  |  |  |  |
| strong fairness Boolean , Boolean ;A path is fair if every fairness constraint holds along the path.PSL in Design Process.• Requirements. Clarifies properties of specifications. Great for requirements review.• Synthesizable HDL. Documents design implementation properties. Great for code reviews• Testbench. Facilitates TB designs. Eases uncertainties in microcycle timing of DUT.• Integration into application. Detects errors during design integration.• Verification. Detects errors, white-box verification. Provides functional verification.• Documentation. Properties and simulation with assertion metrics enhance documentation.   |   |   |  |  |  |  |
| <ul> <li>Requirements</li> <li>Synthesizable HDL</li> <li>Testbench</li> <li>Integration into application</li> <li>Verification</li> <li>Documents design implementation properties. Great for code reviews</li> <li>Facilitates TB designs. Eases uncertainties in microcycle timing of DUT.</li> <li>Detects errors during design integration.</li> <li>Detects errors, white-box verification. Provides functional verification.</li> <li>Properties and simulation with assertion metrics enhance documentation.</li> </ul>  | strong fairness Boolean , Boolean ;     |   |  |  |  |  |
| <ul> <li>Synthesizable HDL</li> <li>Testbench</li> <li>Integration into application</li> <li>Verification</li> <li>Documentation</li> <li>Documentation implementation properties. Great for code reviews</li> <li>Facilitates TB designs. Eases uncertainties in microcycle timing of DUT.</li> <li>Detects errors during design integration.</li> <li>Detects errors, white-box verification. Provides functional verification.</li> <li>Properties and simulation with assertion metrics enhance documentation.</li> </ul>   |   |   |  |  |  |  |
| <ul> <li>Testbench</li> <li>Integration into application</li> <li>Verification</li> <li>Documentation</li> <li>Testbench</li> <li>Integration into application</li> <li>Detects errors during design integration.</li> <li>Detects errors, white-box verification. Provides functional verification.</li> <li>Properties and simulation with assertion metrics enhance documentation.</li> </ul>   | •                                       |   |  |  |  |  |
| <ul> <li>Integration into application</li> <li>Verification</li> <li>Detects errors during design integration.</li> <li>Detects errors, white-box verification. Provides functional verification.</li> <li>Properties and simulation with assertion metrics enhance documentation.</li> </ul>  |   |   |  |  |  |  |
| <ul> <li>Verification</li> <li>Documentation</li> <li>Detects errors, white-box verification. Provides functional verification.</li> <li>Properties and simulation with assertion metrics enhance documentation.</li> </ul>  |   |   |  |  |  |  |
| <ul> <li>Documentation</li> <li>Properties and simulation with assertion metrics enhance documentation.</li> </ul>   |   |   |  |  |  |  |
| Documentation  |   |   |  |  |  |  |
| Guide also available at http://www.vhdleahan.com/ Models and Papers  |   |   |  |  |  |  |
| Guide also available at http://www.vindiconen.com/ iviodels and 1 apels  | Guide also available at                 | http://www.vhdlcohen.com/ Models and Papers                               |  |  |  |  |