***notation used below:

b, b1, b2, etc. are boolean expressions

r, r1, r2, etc. are regular expressions

p, p1, p2, etc. are properties (or formulas)

Feature Name	**Sugar (Verilog Flavor)**	**OVA**
Boolean Expressions		
boolean expression	any Verilog expression	any Verilog expression
Bit vector expressions	**(**bit_vector_expression**)**	**(**bit_vector_expression**)**
named boolean expression	**property** name = expr ;	**bool** name : expr ;
parameterized boolean expression	**property** name(params) = expr ;	**bool** name(params) : expr ;
count macro		**$countones**
Refer to past values in prior cycle	**prev**	**past (**bit_vector_expr**)**
Refer to past value N cycles back	**prev**(expr,n)	**past (**bit_vector_expr, n**)**
Refere to future value in next cycle	**next**	**future (**bit_vector_expr**)**
Clock Definition		
clock expressions		
rising edge of bit_vector_exprssion	(**posedge** clk), **rose**(clk)	**posedge (**boolean**)**
falling edge of bit_vector_expression	(**negedge** clk), **fell**(clk)	**negedge (**boolean**)**
rising or falling edge of bit_vector_expr		**edge (**boolean**)**
define sampling condition		

weak clock	b**@**<clk-expr>
	{r}**@**<clk-expr>
	(p)**@**<clk-expr>
		clock <clk-expr> {
		....
		}

strong clock (clock must occur)	b**@**<clk-expr> !
	{r}**@**<clk-expr> !
	(p)**@**<clk-expr> !
		sclock <clk-expr> {
		....
		}

handling of multiple clocks		
specification	distributed, on b/r/p	centralized, bracketing
combination	"inner" clock rules	"outer" clock rules

Regular Expressions		
regular expression	series of boolean expressions (or subordinate regular expressions)	series of boolean expressions (or subordinate regular expressions)
named regular expression (aka "sequence")	**sequence** name = {r1} ;	**event** name : r1 ;
parameterized sequence	**sequence** name(params) = {r1} ;	**event** name(params) : r1 ;
regular expression bracketing	{.....} --req'd in some cases	(.....) --optional
Regular Expression Forms		
concatenation	r1 **;** r2	r1 #1 r2
one-cycle overlap	r1 **:** r2	r1 #[0] r2
next regexp occurs in		
nth cycle	r1: [*n]; r2	r1 #n r2
nth to mth cycle	r1: [*n:m]; r2	r1 #[n..m] r2
nth or later cycle	r1: [*n:**inf**]; r2	r1 #[n..] r2
mth or earlier cycle	r1: [*0:m]; r2	r1 #[0..m] r2
any cycle	r1: [*0:inf]; r2	r1 #[0..] r2
delay between successive regexps of		
n cycles	r1; [*n]; r2	r1 #[n+1] r2
n to m cycles	r1; [*n:m]; r2	r1 #[n+1..m+1] r2
at least n cycles	r1; [*n:**inf**]; r2	r1 #[n+1..] r2
at most m cycles	r1; [*0:m]; r2	r1 #[1..m+1] r2
any number of cycles	r1; [*]; r2	r1 #[0..] r2
at least 1 cycle	r1; [+]; r2	r1 #[1..] r2
regexp repeated consecutively for		
n cycles	r1[*n]	r1 *[n]
n to m cycles	r1[*n:m]	r1 *[n..m]
at least n cycles	r1[*n:**inf**]	r1 *[n..]
at most m cycles	r1[*0:m]	r1 *[0..m]
any number of cycles	r1[*]	r1 *[0..]
at least 1 cycle	r1[+]	r1 *[1..]
boolean repeated non-consecutively for		
n times	b[=n]	#[0..] (b #[2..])*[n]
n to m times	b[=n:m]	#[0..] (b #[2..])*[n..m]
at least n times	b[=n:**inf**]	#[0..] (b #[2..])*[n..]
at most m times	b[=0:m]	#[0..] (b #[2..])*[0..m]
zero times	b[=0]	!b*[0..]
logical operations on regular expressions		

or	{r1} | {r2}	r1 **or** r2
and (non-length-matching)	{r1} & {r2}	r1 **and** r2
and (length-matching)	{r1} && {r2}	r1 **intersect** r2
"inversion"		**inv** r1
restrictions on regular expressions		
length restriction	{r1} && [*n] {r1} && [*n:m]	**length** [n] **in** r1 **length** [n..m] **in** r1
condition restriction	{r1} && b[*]	**istrue** b **in** r1
window restriction	{[*]; r1; [*]} && {r2}	
regular expression matching		
first occurrence only		**first_match** r1
all occurrences (overlapping)	r1	r1
convergent matching (end of r1 in r2)	**endpoint** e = {r1} ; ... {..; e; ..}	 **matched** r1 - different clock domain **ended** r1 - same clock domain
Regular Expression Based Properties (or Formulas)		
weak suffix implication	{r1} |-> {r2}	if (b) then r1 if (ended r1) then r2
weak next suffix implication	{r1} |=> {r2}	if (matched r1) then r2
suffix implication	{r} (p)	
negative clause of implication		if (b) then r1 [else r2]
suffix implication with single match		r1 **followed_by** r2
suffix implication with formula suffix	{r1} |-> {r2}	r1 **triggers** r2
always	**always** {r}	
eventually	**eventually** {r}	
never	**never** {r}	
Boolean-Based Properties (or Formulas)		
boolean implication	b -> b; b -> p;	
implication boolean	b <-> b	
or operator	p1 || p2	p1 || p2
and operator	p1 && p2	p1 **&&** p2
unary operator to negate result	!p	!p
implication on formulas	p1 -> p2	p1 **implies** p2
binary equivalence operator	p1 **iff** p2	p1 **iff** p2
always/globally	**always** p	**globally** p

eventually	**eventually** p	**eventually** p
never/globally not	**never** p	**globally** !(p)
next and extended next properties	**next** (p) **next** [n] (p) **next_a** [range] (p) **next_e** [range] (p) **next_event** b [n] (p) **next_event_a** b [range] (p) **next_event_e** b [range] (p)	
next (after something else)	p1 -> **next [**n**]** p2	p1 **next [**n**]** p2
(strong) until	p1 **until!** p2	p1 **until** p2
(strong) until - overlapping	p1 **until!_** p2	
weak until	p1 **until_** p2	p1 **wuntil** p2
weak until - overlapping	p1 **until** p2	
(strong) before	p1 **before!** p2	
(strong) before - overlapping	p1 **before!_** p2	
weak before	p1 **before** p2	
weak before - overlapping	p1 **before_** p2	
within window	**within** ({r1}, b) {r2}	
positive reset	p1 **abort** b	**accept** b
negative reset		**reject** b
Directives		
property (both assert and assume)	**property** ...	
assert	**assert** p;	**assert**
assumption with proof obligation	**assume_guarantee** p; **restrict_guarantee** {r};	**assume**
model behavior	Verilog statements in **vunit**s	**model**
restrict w/o proof obligation	**restrict** {r}; **assume** p;	**restrict**
cover toggle to control coverage	**cover** {r};	**cover**