Tables from *The Student's Guide to VHDL, 2nd Edition* by Peter J. Ashenden

Morgan Kaufmann Publishers (an imprint of Elsevier)

TABLE 2.2 *VHDL operators in order of precedence, from most binding to least binding*

Operator	*Operation*	*Left operand type*	*Right operand type*	*Result type*
**	exponentiation	integer or floating-point	integer	same as left operand
abs	absolute value		numeric	same as operand
not	logical negation		boolean, bit, std_ulogic, 1-D array of boolean or bit, std_ulogic_vector	same as operand
and	logical and reduction		1-D array of boolean or bit, std_ulogic_vector	element type of operand
or	logical or reduction			
nand	negated logical and reduction			
nor	negated logical or reduction			
xor	exclusive or reduction			
xnor	negated exclusive or reduction			
*	multiplication	integer or floating-point	same as left operand	same as operands
		physical	integer or real	same as left operand
		integer or real	physical	same as right operand
/	division	integer or floating-point	same as left operand	same as operands
		physical	integer or real	same as left operand
		physical	same as left operand	universal integer
mod	modulo	integer or physical	same as left operand	same as operands
rem	remainder			
+	identity		numeric	same as operand
–	negation			
+	addition	numeric	same as left operand	same as operands
–	subtraction			
&	concatenation	1-D array	same as left operand	same as operands
		1-D array	element type of left operand	same as left operand
		element type of right operand	1-D array	same as right operand
		element type of result	element type of result	1-D array

Operator	*Operation*	*Left operand type*	*Right operand type*	*Result type*
sll **srl** **rol** **ror**	shift-left logical shift-right logical rotate left rotate right	1-D array of boolean or bit, std_ulogic_vector	integer	same as left operand
sla **sra**	shift-left arithmetic shift-right arithmetic	1-D array of boolean or bit	integer	same as left operand
= /=	equality inequality	any except file or protected type	same as left operand	boolean
< <= > >=	less than less than or equal to greater than greater than or equal to	scalar or 1-D array of any discrete type	same as left operand	boolean
?= ?/=	matching equality matching inequality	bit, std_ulogic or 1-D array of bit or std_ulogic	same as left operand	bit or std_ulogic
?< ?<= ?> ?>=	matching less than matching less than or equal to matching greater than matching greater than or equal to	bit or std_ulogic	same as left operand	bit or std_ulogic
and **or** **nand** **nor** **xor** **xnor**	logical and logical or negated logical and negated logical or exclusive or negated exclusive or	boolean, bit, std_ulogic, 1-D array of boolean or bit, std_ulogic_vector	same as left operand	same as operands

TABLE 9.1 *Constants defined in the package* math_real

Constant	*Value*	*Constant*	*Value*
math_e	e	math_log_of_2	$\ln 2$
math_1_over_e	$1/e$	math_log_of_10	$\ln 10$
math_pi	π	math_log2_of_e	$\log_2 e$
math_2_pi	2π	math_log10_of_e	$\log_{10} e$
math_1_over_pi	$1/\pi$	math_sqrt_2	$\sqrt{2}$
math_pi_over_2	$\pi/2$	math_1_over_sqrt_2	$1/\sqrt{2}$
math_pi_over_3	$\pi/3$	math_sqrt_pi	$\sqrt{\pi}$
math_pi_over_4	$\pi/4$	math_deg_to_rad	$2\pi/360$
math_3_pi_over_2	$3\pi/2$	math_rad_to_deg	$360/2\pi$

TABLE 9.2 *Functions defined in the package* math_real

Function	*Meaning*	*Function*	*Meaning*
ceil(x)	Ceiling of x (least integer $\geq x$)	sign(x)	Sign of x (–1.0, 0.0 or +1.0)
floor(x)	Floor of x (greatest integer $\leq x$)	"mod"(x, y)	Floating-point modulus of x / y
round(x)	x rounded to nearest integer value (ties rounded away from 0.0)	realmax(x, y)	Greater of x and y
trunc(x)	x truncated toward 0.0	realmin(x, y)	Lesser of x and y
sqrt(x)	$\sqrt{x}$	log(x)	$\ln x$
cbrt(x)	$\sqrt[3]{x}$	log2(x)	$\log_2 x$
"**"(n, y)	n^y	log10(x)	$\log_{10} x$
"**"(x, y)	x^y	log(x, y)	$\log_y x$
exp(x)	e^x		
sin(x)	sin x (x in radians)	arcsin(x)	arcsin x
cos(x)	cos x (x in radians)	arccos(x)	arccos x
tan(x)	tan x (x in radians)	arctan(x)	arctan x
		arctan(y, x)	arctan of point (x, y)
sinh(x)	sinh x	arcsinh(x)	arcsinh x
cosh(x)	cosh x	arccosh(x)	arccosh x
tanh(x)	tanh x	arctanh(x)	arctanh x

TABLE 9.3 *Overloaded operators defined in* math_complex

Operator	*Operation*		*Left operand*	*Right operand*	*Result*
=	equality		complex_polar	complex_polar	boolean
/=	inequality		complex_polar	complex_polar	boolean
abs	magnitude			complex	positive_real
				complex_polar	positive_real
–	negation			complex	complex
				complex_polar	complex_polar
+	addition	{	complex	complex	complex
–	subtraction		real	complex	complex
*	multiplication		complex	real	complex
/	division		complex_polar	complex_polar	complex_polar
			real	complex_polar	complex_polar
			complex_polar	real	complex_polar

TABLE 9.4 *Functions defined in the package* math_complex

Function	*Result type*	*Meaning*
cmplx(x, y)	complex	$x + jy$
get_principal_value(x)	principal_value	$x + 2\pi k$ for some k, such that $-\pi < \text{result} \leq \pi$
complex_to_polar(c)	complex_polar	c in polar form
polar_to_complex(p)	complex	p in Cartesian form
arg(z)	principal_value	$\arg(z)$ in radians
conj(z)	same as z	complex conjugate of z
sqrt(z)	same as z	$\sqrt{z}$
exp(z)	same as z	e^z
log(z)	same as z	$\ln z$
log2(z)	same as z	$\log_2 z$
log10(z)	same as z	$\log_{10} z$
log(z, y)	same as z	$\log_y z$
sin(z)	same as z	$\sin z$
cos(z)	same as z	$\cos z$
sinh(z)	same as z	$\sinh z$
cosh(z)	same as z	$\cosh z$

TABLE 9.5 *Operators defined in the IEEE standard synthesis packages*

Operator	*Operation*	*Left operand*	*Right operand*	*Result*
abs	absolute value		signed	signed
–	negation			
+	addition	unsigned	unsigned	unsigned
–	subtraction	unsigned	natural	unsigned
*	multiplication	natural	unsigned	unsigned
/	division	signed	signed	signed
rem	remainder	signed	integer	signed
mod	modulo	integer	signed	signed
+	addition	unsigned	*element type*	unsigned
–	subtraction	*element type*	unsigned	unsigned
		signed	*element type*	signed
		element type	signed	signed
=	equality	unsigned	unsigned	boolean
/=	inequality	unsigned	natural	boolean
<	less than	natural	unsigned	boolean
<=	less than or equal to	signed	signed	boolean
>	greater than	signed	integer	boolean
>=	greater than or equal to	integer	signed	boolean
?=	matching equality	unsigned	unsigned	*element type*
?/=	matching inequality	unsigned	natural	*element type*
?<	matching less than	natural	unsigned	*element type*
?<=	matching less than or equal to	signed	signed	*element type*
?>	matching greater than	signed	integer	*element type*
?>=	matching greater than or equal to	integer	signed	*element type*
sll	shift-left logical	unsigned	integer	unsigned
srl	shift-right logical	signed	integer	signed
sla	shift-left arithmetic			
sra	shift-right arithmetic			
rol	rotate left			
ror	rotate right			

Operator	*Operation*		*Left operand*	*Right operand*	*Result*
not	negation			unsigned	unsigned
				signed	signed
and	logical and		unsigned	unsigned	unsigned
or	logical or		unsigned	*element type*	unsigned
nand	negated logical and		*element type*	unsigned	unsigned
nor	negated logical or		signed	signed	signed
xor	exclusive or		signed	*element type*	signed
xnor	negated exclusive or		*element type*	signed	signed
and	logical and reduction			unsigned	*element type*
or	logical or reduction			signed	*element type*
nand	negated logical and reduction				
nor	negated logical or reduction				
xor	exclusive or reduction				
xnor	negated exclusive or reduction				

TABLE 9.6 *Functions defined in the IEEE standard synthesis packages*

Function		*First parameter*	*Second parameter*	*Result*
minimum maximum		unsigned unsigned natural signed signed integer	unsigned natural unsigned signed integer signed	unsigned unsigned unsigned signed signed signed
shift_left shift_right rotate_left rotate_right		unsigned signed	natural natural	unsigned signed
find_leftmost find_rightmost		unsigned signed	*element type* *element type*	integer integer
resize		unsigned unsigned signed signed	natural unsigned natural signed	unsigned unsigned signed signed
to_integer		unsigned signed		natural integer
to_unsigned		natural natural	natural unsigned	unsigned unsigned
to_signed		integer integer	natural signed	signed signed
rising_edge[a] falling_edge[a]		**signal** bit		boolean
std_match[b]		unsigned signed std_ulogic std_ulogic_vector	unsigned signed std_ulogic std_ulogic_vector	boolean boolean boolean boolean

Function		*First parameter*	*Second parameter*	*Result*
to_01[b] to_X01[b] to_X01Z[b] to_UX01[b]	{	unsigned signed	⟦ std_ulogic ⟧ ⟦ std_ulogic ⟧	unsigned signed
to_string to_bstring to_binary_string to_ostring to_octal_string to_hstring to_hex_string	{	unsigned signed		string string

a. Provided in **numeric_bit** only.
b. Provided in **numeric_std** only.

TABLE 9.7 *Operand and result types*

Operators	*Left*	*Right*	*Result*
Binary **and**, **or**, **nand**, **nor**, **xor**, **xnor**	std_ulogic	std_ulogic	std_ulogic
	LogicArrayType	*LogicArrayType*	*LogicArrayType*
	LogicArrayType	std_ulogic	*LogicArrayType*
	std_ulogic	*LogicArrayType*	*LogicArrayType*
not		std_ulogic	std_ulogic
		LogicArrayType	*LogicArrayType*
Unary reduction **and**, **or**, **nand**, **nor**, **xor**, **xnor**		*LogicArrayType*	std_ulogic
=, /=, <, <=, >, >=	*NumericArrayType*	*NumericArrayType*	boolean
	NumericArrayType	integer	boolean
	integer	*NumericArrayType*	boolean
?=, ?/=, ?<, ?<=, ?>, ?>=	*NumericArrayType*	*NumericArrayType*	*ArrayElementType*
	NumericArrayType	integer	*ArrayElementType*
	integer	*NumericArrayType*	*ArrayElementType*
rol, **ror**, **sll**, **srl**	*LogicArrayType*	integer	*LogicArrayType*
sla, **sra**	*NumericArrayType*	integer	*NumericArrayType*
Binary +, –, *, /, **mod**, **rem**	*NumericArrayType*	*NumericArrayType*	*NumericArrayType*
	NumericArrayType	integer	*NumericArrayType*
	integer	*NumericArrayType*	*NumericArrayType*
Binary +, –	*NumericArrayType*	std_ulogic	*NumericArrayType*
	std_ulogic	*NumericArrayType*	*NumericArrayType*
Unary –, **abs**		signed, sfixed, float	signed, sfixed, float
maximum, minimum	*NumericArrayType*	*NumericArrayType*	*NumericArrayType*
	NumericArrayType	integer	*NumericArrayType*
	integer	*NumericArrayType*	*NumericArrayType*

TABLE 9.8 *Result sizes and index ranges*

Operator	*Result type*	*Result size and/or range*
Array/array **and**, **or**, **nand**, **nor**, **xor**, **xnor**	*ArrayOfBits*	*Result*'length = *L*'length = *R*'length
Array/scalar **and**, **or**, **nand**, **nor**, **xor**, **xnor**	*ArrayOfBits*	*Result*'length = *A*'length
not	*ArrayOfBits*	*Result*'length = *R*'length
rol, **ror**, **sll**, **srl**, **sla**, **sra**	*ArrayOfBits*	*Result*'length = *A*'length
Binary +, –	unsigned, signed	maximum(*L*'length, *R*'length) – 1 down to 0
*	unsigned, signed	*L*'length + *R*'length – 1 down to 0
/	unsigned, signed	*L*'length – 1 down to 0
rem	unsigned, signed	*R*'length – 1 down to 0
mod	unsigned, signed	*R*'length – 1 down to 0
Unary –, **abs**	signed	*R*'length – 1 down to 0
minimum, maximum	*DiscreteArrayType*	*Result*'length = *A*'length
	unsigned, signed	maximum(*L*'length, *R*'length) – 1 down to 0

TABLE 9.9 *Conversions between bit and standard-logic types*

Function	*Result type*	*Parameter 1 type*	*Parameter 2*	*Package*
to_bit	bit	std_ulogic	xmap	1164
to_std_ulogic	std_ulogic	bit		1164
to_bv	bit_vector	sulv	xmap	1164
		natural	size	nbu
		natural	size_res	nbu
to_sulv	sulv	bv		1164
		slv		1164
		natural	size	nsu
		natural	size_res	nsu
to_slv	slv	bv		1164
		sulv		1164
		natural	natural	nsu
		natural	size_res	nsu

TABLE 9.10 *Conversion functions yielding* unsigned *and* signed *values*

Function	*Result type*	*Param 1 type*	*Param 2*	*Param 3*	*Param 4*	*Package*
to_unsigned	unsigned	natural	size			ns/b
			size_res			
to_signed	signed	integer	size			ns/b
			size_res			

TABLE 9.11 *Conversion functions yielding* integer *values*

Function	*Result type*	*Param 1 type*	*Package*
to_integer	natural	bv	nbu
	natural	sulv	nsu
	natural	unsigned	ns/b
	integer	signed	

TABLE 9.12 *Resizing functions*

Function	*Result type*	*Param 1 type*	*Param 2*	*Package*
resize	bv	bv	new_size	nbu
			size_res	
	sulv	sulv	new_size	nsu
			size_res	
	unsigned	unsigned	new_size	ns/b
			size_res	
	signed	signed	new_size	
			size_res	

TABLE 9.13 *Strength reduction mappings*

Function	*'U'*	*'X'*	*'0'*	*'1'*	*'Z'*	*'W'*	*'L'*	*'H'*	*'–'*
to_01	xmap	xmap	'0'	'1'	xmap	xmap	'0'	'1'	xmap
to_X01	'X'	'X'	'0'	'1'	'X'	'X'	'0'	'1'	'X'
to_X01Z	'X'	'X'	'0'	'1'	'Z'	'X'	'0'	'1'	'X'
to_UX01	'U'	'X'	'0'	'1'	'X'	'X'	'0'	'1'	'X'

TABLE 15.1 *The Gumnut instruction set*

Arithmetic and logical instructions	
add *rd*, *rs*, *op2*	Add *rs* and *op2*, result in *rd*
addc *rd*, *rs*, *op2*	Add *rs* and *op2* with carry, result in *rd*
sub *rd*, *rs*, *op2*	Subtract *op2* from *rs*, result in *rd*
subc *rd*, *rs*, *op2*	Subtract *op2* from *rs* with carry, result in *rd*
and *rd*, *rs*, *op2*	Logical AND of *rs* and *op2*, result in *rd*
or *rd*, *rs*, *op2*	Logical OR of *rs* and *op2*, result in *rd*
xor *rd*, *rs*, *op2*	Logical XOR of *rs* and *op2*, result in *rd*
mask *rd*, *rs*, *op2*	Logical AND of *rs* and NOT *op2*, result in *rd*
Shift instructions	
shl *rd*, *rs*, *count*	Shift *rs* value left *count* places, result in *rd*
shr *rd*, *rs*, *count*	Shift *rs* value right *count* places, result in *rd*
rol *rd*, *rs*, *count*	Rotate *rs* value left *count* places, result in *rd*
ror *rd*, *rs*, *count*	Rotate *rs* value right *count* places, result in *rd*
Memory and I/O instructions	
ldm *rd*, (*rs*)±*offset*	Load to *rd* from memory
stm *rd*, (*rs*)±*offset*	Store to memory from *rd*
inp *rd*, (*rs*)±*offset*	Input to *rd* from input controller register
out *rd*, (*rs*)±*offset*	Output to output controller register from *rd*
Branch instructions	
bz ±*disp*	Branch if Z is set
bnz ±*disp*	Branch is Z is not set
bc ±*disp*	Branch if C is set
bnc ±*disp*	Branch if C is not set
Jump instructions	
jmp *addr*	Jump to *addr*
jsb *addr*	Jump to subroutine at *addr*
Miscellaneous instructions	

ret	Return from subroutine
reti	Return from interrupt
enai	Enable interrupts
disi	Disable interrupts
wait	Wait for interrupts
stby	Enter low-power standby mode

TABLE 15.2 *Function code values*

add	000	addc	001	sub	010	subc	011
and	100	or	101	xor	110	mask	111
shl	00	shr	01	rol	10	ror	11
ldm	00	stm	01	inp	10	out	11
bz	00	bnz	01	bc	10	bnc	11
jmp	0	jsb	1				
ret	000	reti	001	enai	010	disi	011
wait	100	stby	101				