

DE0 NANO v/f

Generated by Doxygen 1.8.9.1

Wed Apr 1 2015 09:49:04

Contents

1 Namespace Index	1
1.1 Packages	1
2 Design Unit Index	3
2.1 Design Unit Hierarchy	3
3 Design Unit Index	5
3.1 Design Unit List	5
4 File Index	7
4.1 File List	7
5 Namespace Documentation	9
5.1 fixed_float_types Namespace Reference	9
5.2 fixed_pkg Namespace Reference	9
5.3 my_types_pkg Namespace Reference	9
6 Class Documentation	11
6.1 fixed_pkg Package Body Reference	11
6.1.1 Detailed Description	27
6.1.2 Member Function Documentation	27
6.1.2.1 ""*""	27
6.1.2.2 ""*""	27
6.1.2.3 ""*""	27
6.1.2.4 ""*""	28
6.1.2.5 ""*""	28
6.1.2.6 ""*""	28
6.1.2.7 ""*""	28
6.1.2.8 ""*""	28
6.1.2.9 ""*""	28
6.1.2.10 ""*""	28
6.1.2.11 ""+""	28
6.1.2.12 ""+""	28

6.1.2.13	"+"	28
6.1.2.14	"+"	28
6.1.2.15	"+"	28
6.1.2.16	"+"	29
6.1.2.17	"+"	29
6.1.2.18	"+"	29
6.1.2.19	"+"	29
6.1.2.20	"+"	29
6.1.2.21	"-"	29
6.1.2.22	"-"	29
6.1.2.23	"-"	29
6.1.2.24	"-"	29
6.1.2.25	"-"	29
6.1.2.26	"-"	29
6.1.2.27	"-"	29
6.1.2.28	"-"	30
6.1.2.29	"-"	30
6.1.2.30	"-"	30
6.1.2.31	"-"	30
6.1.2.32	"/"	30
6.1.2.33	"/"	30
6.1.2.34	"/"	30
6.1.2.35	"/"	30
6.1.2.36	"/"	30
6.1.2.37	"/"	30
6.1.2.38	"/"	30
6.1.2.39	"/"	30
6.1.2.40	"/"	31
6.1.2.41	"/"	31
6.1.2.42	"/=	31
6.1.2.43	"/=	31
6.1.2.44	"/=	31
6.1.2.45	"/=	31
6.1.2.46	"/=	31
6.1.2.47	"/=	31
6.1.2.48	"/=	31
6.1.2.49	"/=	31
6.1.2.50	"/=	31
6.1.2.51	"/=	31
6.1.2.52	"/<	32

6.1.2.53 ""<""	32
6.1.2.54 ""<""	32
6.1.2.55 ""<""	32
6.1.2.56 ""<""	32
6.1.2.57 ""<""	32
6.1.2.58 ""<""	32
6.1.2.59 ""<""	32
6.1.2.60 ""<""	32
6.1.2.61 ""<""	32
6.1.2.62 ""<=""	32
6.1.2.63 ""<=""	32
6.1.2.64 ""<=""	33
6.1.2.65 ""<=""	33
6.1.2.66 ""<=""	33
6.1.2.67 ""<=""	33
6.1.2.68 ""<=""	33
6.1.2.69 ""<=""	33
6.1.2.70 ""<=""	33
6.1.2.71 ""<=""	33
6.1.2.72 ""=""	33
6.1.2.73 ""=""	33
6.1.2.74 ""=""	33
6.1.2.75 ""=""	33
6.1.2.76 ""=""	34
6.1.2.77 ""=""	34
6.1.2.78 ""=""	34
6.1.2.79 ""=""	34
6.1.2.80 ""=""	34
6.1.2.81 ""=""	34
6.1.2.82 "">""	34
6.1.2.83 "">""	34
6.1.2.84 "">""	34
6.1.2.85 "">""	34
6.1.2.86 "">""	34
6.1.2.87 "">""	34
6.1.2.88 "">""	35
6.1.2.89 "">""	35
6.1.2.90 "">""	35
6.1.2.91 "">""	35
6.1.2.92 "">=""	35

6.1.2.93 "">=""	35
6.1.2.94 "">=""	35
6.1.2.95 "">=""	35
6.1.2.96 "">=""	35
6.1.2.97 "">=""	35
6.1.2.98 "">=""	35
6.1.2.99 "">=""	35
6.1.2.100 "">=""	36
6.1.2.101 "">=""	36
6.1.2.102 ""abs""	36
6.1.2.103 ""and""	36
6.1.2.104 ""and""	36
6.1.2.105 ""and""	36
6.1.2.106 ""and""	36
6.1.2.107 ""and""	36
6.1.2.108 ""and""	36
6.1.2.109 ""mod""	36
6.1.2.110 ""mod""	36
6.1.2.111 ""mod""	37
6.1.2.112 ""mod""	37
6.1.2.113 ""mod""	37
6.1.2.114 ""mod""	37
6.1.2.115 ""mod""	37
6.1.2.116 ""mod""	37
6.1.2.117 ""mod""	37
6.1.2.118 ""mod""	37
6.1.2.119 ""nand""	37
6.1.2.120 ""nand""	37
6.1.2.121 ""nand""	37
6.1.2.122 ""nand""	37
6.1.2.123 ""nand""	38
6.1.2.124 ""nand""	38
6.1.2.125 ""nor""	38
6.1.2.126 ""nor""	38
6.1.2.127 ""nor""	38
6.1.2.128 ""nor""	38
6.1.2.129 ""nor""	38
6.1.2.130 ""nor""	38
6.1.2.131 ""not""	38
6.1.2.132 ""not""	38

6.1.2.133 ""or""	38
6.1.2.134 ""or""	39
6.1.2.135 ""or""	39
6.1.2.136 ""or""	39
6.1.2.137 ""or""	39
6.1.2.138 ""or""	39
6.1.2.139 ""rem""	39
6.1.2.140 ""rem""	39
6.1.2.141 ""rem""	39
6.1.2.142 ""rem""	39
6.1.2.143 ""rem""	39
6.1.2.144 ""rem""	39
6.1.2.145 ""rem""	39
6.1.2.146 ""rem""	40
6.1.2.147 ""rem""	40
6.1.2.148 ""rem""	40
6.1.2.149 ""rol""	40
6.1.2.150 ""rol""	40
6.1.2.151 ""ror""	40
6.1.2.152 ""ror""	40
6.1.2.153 ""sla""	40
6.1.2.154 ""sla""	40
6.1.2.155 ""sll""	40
6.1.2.156 ""sll""	40
6.1.2.157 ""sra""	40
6.1.2.158 ""sra""	41
6.1.2.159 ""sra""	41
6.1.2.160 ""srl""	41
6.1.2.161 ""srl""	41
6.1.2.162 ""xnor""	41
6.1.2.163 ""xnor""	41
6.1.2.164 ""xnor""	41
6.1.2.165 ""xnor""	41
6.1.2.166 ""xnor""	41
6.1.2.167 ""xnor""	41
6.1.2.168 ""xor""	41
6.1.2.169 ""xor""	42
6.1.2.170 ""xor""	42
6.1.2.171 ""xor""	42
6.1.2.172 ""xor""	42

6.1.2.173 ""xor""	42
6.1.2.174 \?/=	42
6.1.2.175 \?/=\\	42
6.1.2.176 \?/=\\	42
6.1.2.177 \?/=\\	42
6.1.2.178 \?/=\\	42
6.1.2.179 \?/=\\	42
6.1.2.180 \?/=\\	42
6.1.2.181 \?/=\\	43
6.1.2.182 \?/=\\	43
6.1.2.183 \?/=\\	43
6.1.2.184 \?/=\\	43
6.1.2.185 \?/=\\	43
6.1.2.186 \?/=\\	43
6.1.2.187 \?<=\\	43
6.1.2.188 \?<=\\	43
6.1.2.189 \?<=\\	43
6.1.2.190 \?<=\\	43
6.1.2.191 \?<=\\	43
6.1.2.192 \?<=\\	43
6.1.2.193 \?<=\\	44
6.1.2.194 \?<=\\	44
6.1.2.195 \?<=\\	44
6.1.2.196 \?<=\\	44
6.1.2.197 \?<=\\	44
6.1.2.198 \?<=\\	44
6.1.2.199 \?<\\	44
6.1.2.200 \?<\\	44
6.1.2.201 \?<\\	44
6.1.2.202 \?<\\	44
6.1.2.203 \?<\\	44
6.1.2.204 \?<\\	44
6.1.2.205 \?<\\	45
6.1.2.206 \?<\\	45
6.1.2.207 \?<\\	45
6.1.2.208 \?<\\	45
6.1.2.209 \?<\\	45
6.1.2.210 \?<\\	45
6.1.2.211 \?=<\\	45
6.1.2.212 \?=<\\	45

6.1.2.213 \?=\\	45
6.1.2.214 \?=&\\	45
6.1.2.215 \?=&\\	45
6.1.2.216 \?=&\\	45
6.1.2.217 \?=&\\	46
6.1.2.218 \?=&\\	46
6.1.2.219 \?=&\\	46
6.1.2.220 \?=&\\	46
6.1.2.221 \?=&\\	46
6.1.2.222 \?=&\\	46
6.1.2.223 \?=&\\	46
6.1.2.224 \?>=&\\	46
6.1.2.225 \?>=&\\	46
6.1.2.226 \?>=&\\	46
6.1.2.227 \?>=&\\	46
6.1.2.228 \?>=&\\	46
6.1.2.229 \?>=&\\	47
6.1.2.230 \?>=&\\	47
6.1.2.231 \?>=&\\	47
6.1.2.232 \?>=&\\	47
6.1.2.233 \?>=&\\	47
6.1.2.234 \?>=&\\	47
6.1.2.235 \?>=&\\	47
6.1.2.236 \?>\\	47
6.1.2.237 \?>\\	47
6.1.2.238 \?>\\	47
6.1.2.239 \?>\\	47
6.1.2.240 \?>\\	47
6.1.2.241 \?>\\	48
6.1.2.242 \?>\\	48
6.1.2.243 \?>\\	48
6.1.2.244 \?>\\	48
6.1.2.245 \?>\\	48
6.1.2.246 \?>\\	48
6.1.2.247 \?>\\	48
6.1.2.248 add_carry	48
6.1.2.249 add_carry	48
6.1.2.250 and_reduce	48
6.1.2.251 and_reduce	48
6.1.2.252 and_reduce	48

6.1.2.253 calculate_string_boundary	49
6.1.2.254 Char2QuadBits	49
6.1.2.255 Char2TriBits	49
6.1.2.256 cleanvec	49
6.1.2.257 cleanvec	49
6.1.2.258 divide	49
6.1.2.259 divide	49
6.1.2.260 errmes	49
6.1.2.261 errmes	49
6.1.2.262 find_leftmost	49
6.1.2.263 find_leftmost	49
6.1.2.264 find_rightmost	50
6.1.2.265 find_rightmost	50
6.1.2.266 from_hstring	50
6.1.2.267 from_hstring	50
6.1.2.268 from_hstring	50
6.1.2.269 from_hstring	50
6.1.2.270 from_hstring	50
6.1.2.271 from_hstring	50
6.1.2.272 from_ostring	50
6.1.2.273 from_ostring	50
6.1.2.274 from_ostring	50
6.1.2.275 from_ostring	51
6.1.2.276 from_ostring	51
6.1.2.277 from_ostring	51
6.1.2.278 from_string	51
6.1.2.279 from_string	51
6.1.2.280 from_string	51
6.1.2.281 from_string	51
6.1.2.282 from_string	51
6.1.2.283 from_string	51
6.1.2.284 HREAD	51
6.1.2.285 HREAD	51
6.1.2.286 HREAD	52
6.1.2.287 HREAD	52
6.1.2.288 HREAD_common	52
6.1.2.289 hwrite	52
6.1.2.290 hwrite	52
6.1.2.291 ls_Negative	52
6.1.2.292 ls_X	52

6.1.2.293 ls_X	52
6.1.2.294 ls_X	52
6.1.2.295 ls_X	52
6.1.2.296 maximum	52
6.1.2.297 maximum	53
6.1.2.298 maximum	53
6.1.2.299 maximum	53
6.1.2.300 maximum	53
6.1.2.301 maximum	53
6.1.2.302 maximum	53
6.1.2.303 maximum	53
6.1.2.304 maximum	53
6.1.2.305 maximum	53
6.1.2.306 maximum	53
6.1.2.307 mine	53
6.1.2.308 minimum	53
6.1.2.309 minimum	54
6.1.2.310 minimum	54
6.1.2.311 minimum	54
6.1.2.312 minimum	54
6.1.2.313 minimum	54
6.1.2.314 minimum	54
6.1.2.315 minimum	54
6.1.2.316 minimum	54
6.1.2.317 minimum	54
6.1.2.318 minimum	54
6.1.2.319 mins	54
6.1.2.320 modulo	55
6.1.2.321 modulo	55
6.1.2.322 nand_reduce	55
6.1.2.323 nand_reduce	55
6.1.2.324 nand_reduce	55
6.1.2.325 nor_reduce	55
6.1.2.326 nor_reduce	55
6.1.2.327 nor_reduce	55
6.1.2.328 or_reduce	55
6.1.2.329 or_reduce	55
6.1.2.330 or_reduce	55
6.1.2.331 OREAD	56
6.1.2.332 OREAD	56

6.1.2.333 OREAD	56
6.1.2.334 OREAD	56
6.1.2.335 OREAD_common	56
6.1.2.336 owrite	56
6.1.2.337 owrite	56
6.1.2.338 READ	56
6.1.2.339 READ	56
6.1.2.340 READ	56
6.1.2.341 READ	56
6.1.2.342 reciprocal	57
6.1.2.343 reciprocal	57
6.1.2.344 remainder	57
6.1.2.345 remainder	57
6.1.2.346 resize	57
6.1.2.347 resize	57
6.1.2.348 resize	57
6.1.2.349 resize	57
6.1.2.350 round_fixed	58
6.1.2.351 round_fixed	58
6.1.2.352 round_up	58
6.1.2.353 round_up	58
6.1.2.354 saturate	58
6.1.2.355 saturate	58
6.1.2.356 saturate	58
6.1.2.357 saturate	58
6.1.2.358 scalb	58
6.1.2.359 scalb	58
6.1.2.360 scalb	58
6.1.2.361 scalb	59
6.1.2.362 sfixed_high	59
6.1.2.363 sfixed_low	59
6.1.2.364 sfixed_high	59
6.1.2.365 sfixed_high	59
6.1.2.366 sfixed_low	59
6.1.2.367 sfixed_low	59
6.1.2.368 SHIFT_LEFT	59
6.1.2.369 SHIFT_LEFT	59
6.1.2.370 SHIFT_RIGHT	59
6.1.2.371 SHIFT_RIGHT	60
6.1.2.372 skip_whitespace	60

6.1.2.373 std_match	60
6.1.2.374 std_match	60
6.1.2.375 to_01	60
6.1.2.376 to_01	60
6.1.2.377 to_fixed	60
6.1.2.378 to_fixed	60
6.1.2.379 to_hstring	60
6.1.2.380 to_hstring	60
6.1.2.381 to_hstring	60
6.1.2.382 to_integer	61
6.1.2.383 to_integer	61
6.1.2.384 to_ostring	61
6.1.2.385 to_ostring	61
6.1.2.386 to_ostring	61
6.1.2.387 to_real	61
6.1.2.388 to_real	61
6.1.2.389 to_s	61
6.1.2.390 to_SFix	61
6.1.2.391 to_SFix	61
6.1.2.392 to_sfixed	61
6.1.2.393 to_sfixed	62
6.1.2.394 to_sfixed	62
6.1.2.395 to_sfixed	62
6.1.2.396 to_sfixed	62
6.1.2.397 to_sfixed	62
6.1.2.398 to_sfixed	62
6.1.2.399 to_sfixed	62
6.1.2.400 to_sfixed	62
6.1.2.401 to_sfixed	62
6.1.2.402 to_sfixed	63
6.1.2.403 to_sfixed	63
6.1.2.404 to_signed	63
6.1.2.405 to_signed	63
6.1.2.406 to_slv	63
6.1.2.407 to_slv	63
6.1.2.408 to_string	63
6.1.2.409 to_string	63
6.1.2.410 to_suv	63
6.1.2.411 to_suv	63
6.1.2.412 to_UFix	63

6.1.2.413 to_UFix	64
6.1.2.414 to_ufixed	64
6.1.2.415 to_ufixed	64
6.1.2.416 to_ufixed	64
6.1.2.417 to_ufixed	64
6.1.2.418 to_ufixed	64
6.1.2.419 to_ufixed	64
6.1.2.420 to_ufixed	64
6.1.2.421 to_ufixed	64
6.1.2.422 to_ufixed	65
6.1.2.423 to_ufixed	65
6.1.2.424 to_ufixed	65
6.1.2.425 to_ufixed	65
6.1.2.426 to_uns	65
6.1.2.427 to_unsigned	65
6.1.2.428 to_unsigned	65
6.1.2.429 To_UX01	65
6.1.2.430 to_UX01	65
6.1.2.431 To_X01	65
6.1.2.432 to_X01	66
6.1.2.433 To_X01Z	66
6.1.2.434 to_X01Z	66
6.1.2.435 ufix_high	66
6.1.2.436 ufix_low	66
6.1.2.437 ufixed_high	66
6.1.2.438 ufixed_high	66
6.1.2.439 ufixed_low	66
6.1.2.440 ufixed_low	66
6.1.2.441 write	66
6.1.2.442 write	66
6.1.2.443 xnor_reduce	67
6.1.2.444 xnor_reduce	67
6.1.2.445 xnor_reduce	67
6.1.2.446 xor_reduce	67
6.1.2.447 xor_reduce	67
6.1.2.448 xor_reduce	67
6.1.3 Member Data Documentation	67
6.1.3.1 c	67
6.1.3.2 char_indexed_by_MVL9	67
6.1.3.3 char_to_MVL9	67

6.1.3.4	char_to_MVL9plus	67
6.1.3.5	fixedsynth_or_real	67
6.1.3.6	founddot	68
6.1.3.7	i	68
6.1.3.8	IEEE	68
6.1.3.9	lastu	68
6.1.3.10	match_logic_table	68
6.1.3.11	MATH_REAL	68
6.1.3.12	MVL9_indexed_by_char	68
6.1.3.13	MVL9_to_char	68
6.1.3.14	MVL9plus	68
6.1.3.15	MVL9plus_indexed_by_char	68
6.1.3.16	NASF	68
6.1.3.17	NAUF	68
6.1.3.18	NBSP	69
6.1.3.19	NSLV	69
6.1.3.20	NUS	69
6.1.3.21	nybble	69
6.1.3.22	nybble	69
6.1.3.23	stdlogic_table	69
6.1.3.24	xgood	69
6.2	Behavioral Architecture Reference	69
6.2.1	Detailed Description	70
6.2.2	Member Function Documentation	70
6.2.2.1	PROCESS_13	70
6.2.3	Member Data Documentation	70
6.2.3.1	i	70
6.2.3.2	memory_type	70
6.2.3.3	sine	70
6.3	clk_div Entity Reference	70
6.3.1	Detailed Description	71
6.3.2	Member Data Documentation	71
6.3.2.1	clk_in	71
6.3.2.2	clk_out	71
6.3.2.3	div	71
6.3.2.4	en	71
6.3.2.5	IEEE	71
6.3.2.6	STD_LOGIC_1164	71
6.3.2.7	STD_LOGIC_UNSIGNED	71
6.4	comparador Architecture Reference	71

6.4.1	Detailed Description	72
6.4.2	Member Function Documentation	72
6.4.2.1	PROCESS_1	72
6.4.2.2	PROCESS_2	72
6.4.3	Member Data Documentation	72
6.4.3.1	comp_int	72
6.5	comparador Entity Reference	72
6.5.1	Detailed Description	73
6.5.2	Member Data Documentation	73
6.5.2.1	amost	73
6.5.2.2	c	73
6.5.2.3	clk	73
6.5.2.4	comp	73
6.5.2.5	comp_out	73
6.5.2.6	en	73
6.5.2.7	fixed_pkg	73
6.5.2.8	IEEE	74
6.5.2.9	n_bits_c	74
6.5.2.10	numeric_std	74
6.5.2.11	STD_LOGIC_1164	74
6.5.2.12	STD_LOGIC_UNSIGNED	74
6.6	comparadores Entity Reference	74
6.6.1	Detailed Description	75
6.6.2	Member Data Documentation	75
6.6.2.1	amost	75
6.6.2.2	c	75
6.6.2.3	clk	75
6.6.2.4	comp	75
6.6.2.5	comp_out	75
6.6.2.6	en	75
6.6.2.7	fixed_pkg	75
6.6.2.8	IEEE	76
6.6.2.9	MAX	76
6.6.2.10	my_types_pkg	76
6.6.2.11	n_bits_c	76
6.6.2.12	numeric_std	76
6.6.2.13	STD_LOGIC_1164	76
6.6.2.14	STD_LOGIC_UNSIGNED	76
6.7	comparadores Architecture Reference	76
6.7.1	Detailed Description	76

6.7.2	Member Data Documentation	77
6.7.2.1	comparador	77
6.7.2.2	u	77
6.8	contador Architecture Reference	77
6.8.1	Detailed Description	77
6.8.2	Member Function Documentation	77
6.8.2.1	PROCESS_3	77
6.8.3	Member Data Documentation	77
6.8.3.1	count_int	77
6.9	contador Entity Reference	77
6.9.1	Detailed Description	78
6.9.2	Member Data Documentation	78
6.9.2.1	clk	78
6.9.2.2	comp	78
6.9.2.3	count	78
6.9.2.4	count_comp	78
6.9.2.5	count_ini	78
6.9.2.6	count_max	79
6.9.2.7	en	79
6.9.2.8	IEEE	79
6.9.2.9	n_bits	79
6.9.2.10	numeric_std	79
6.9.2.11	reset	79
6.9.2.12	sinc	79
6.9.2.13	STD_LOGIC_1164	79
6.9.2.14	STD_LOGIC_UNSIGNED	79
6.10	DE0_NANO_VF Entity Reference	79
6.10.1	Detailed Description	81
6.10.2	Member Data Documentation	81
6.10.2.1	CLOCK_50	81
6.10.2.2	F	81
6.10.2.3	fixed_pkg	81
6.10.2.4	GPIO_0	81
6.10.2.5	I	81
6.10.2.6	ID_MEM_DAC	82
6.10.2.7	ID_MEM_SW1	82
6.10.2.8	ieee	82
6.10.2.9	INT0_FA	82
6.10.2.10	INT0_FB	82
6.10.2.11	INT0_FC	82

6.10.2.12 KEY	82
6.10.2.13 LED	82
6.10.2.14 my_types_pkg	82
6.10.2.15 N	82
6.10.2.16 n_bits_c	82
6.10.2.17 n_bits_phase	82
6.10.2.18 NBITS_MEM_ADDRESS	83
6.10.2.19 Nin	83
6.10.2.20 Nout	83
6.10.2.21 NUMERIC_STD	83
6.10.2.22 PWM1H_FA	83
6.10.2.23 PWM1H_FB	83
6.10.2.24 PWM1H_FC	83
6.10.2.25 PWM1L_FA	83
6.10.2.26 PWM1L_FB	83
6.10.2.27 PWM1L_FC	83
6.10.2.28 PWM2H_FA	83
6.10.2.29 PWM2H_FB	83
6.10.2.30 PWM2H_FC	84
6.10.2.31 PWM2L_FA	84
6.10.2.32 PWM2L_FB	84
6.10.2.33 PWM2L_FC	84
6.10.2.34 RESET_FA	84
6.10.2.35 RESET_FB	84
6.10.2.36 RESET_FC	84
6.10.2.37 std_logic_1164	84
6.10.2.38 std_logic_arith	84
6.10.2.39 std_logic_unsigned	84
6.10.2.40 SW	84
6.10.2.41 TAM_MEM	84
6.11 deadtime Entity Reference	85
6.11.1 Detailed Description	85
6.11.2 Member Data Documentation	85
6.11.2.1 CLK	85
6.11.2.2 ieee	85
6.11.2.3 p_Pwm1_Out	85
6.11.2.4 p_Pwm2_Out	85
6.11.2.5 p_Pwm_In	85
6.11.2.6 std_logic_1164	86
6.11.2.7 std_logic_arith	86

6.11.2.8 std_logic_unsigned	86
6.12 deadtime_archetecture Architecture Reference	86
6.12.1 Detailed Description	86
6.12.2 Member Function Documentation	86
6.12.2.1 PROCESS_6	86
6.12.3 Member Data Documentation	86
6.12.3.1 sig_Not_Pwm_In	86
6.13 div Architecture Reference	86
6.13.1 Detailed Description	87
6.13.2 Member Function Documentation	87
6.13.2.1 PROCESS_0	87
6.13.3 Member Data Documentation	87
6.13.3.1 clk_out.bi	87
6.13.3.2 count	87
6.14 fbpspwm dt Entity Reference	87
6.14.1 Detailed Description	88
6.14.2 Member Data Documentation	88
6.14.2.1 almost	88
6.14.2.2 c	88
6.14.2.3 c_Dead_t	88
6.14.2.4 clk	88
6.14.2.5 comp	88
6.14.2.6 en	88
6.14.2.7 fixed_pkg	89
6.14.2.8 IEEE	89
6.14.2.9 n_bits_c	89
6.14.2.10 numeric_std	89
6.14.2.11 port_PWM01	89
6.14.2.12 port_PWM02	89
6.14.2.13 STD_LOGIC_1164	89
6.14.2.14 STD_LOGIC_UNSIGNED	89
6.15 fbpspwm dt_arch Architecture Reference	89
6.15.1 Detailed Description	90
6.15.2 Member Function Documentation	90
6.15.2.1 PROCESS_7	90
6.15.2.2 PROCESS_8	90
6.15.2.3 PROCESS_9	90
6.15.3 Member Data Documentation	90
6.15.3.1 comp_int	90
6.15.3.2 comp_out	90

6.15.3.3 p_Pwm_In	90
6.15.3.4 sig_Not_Pwm_In	90
6.16 fixed_float_types Package Reference	90
6.16.1 Detailed Description	90
6.16.2 Member Data Documentation	91
6.16.2.1 fixed_overflow_style_type	91
6.16.2.2 fixed_round_style_type	91
6.16.2.3 round_type	91
6.17 fixed_pkg Package Reference	91
6.17.1 Detailed Description	106
6.17.2 Member Function Documentation	106
6.17.2.1 ""*""	106
6.17.2.2 ""*""	107
6.17.2.3 ""*""	107
6.17.2.4 ""*""	107
6.17.2.5 ""*""	107
6.17.2.6 ""*""	107
6.17.2.7 ""*""	107
6.17.2.8 ""*""	107
6.17.2.9 ""*""	107
6.17.2.10 ""*""	107
6.17.2.11 ""+""	107
6.17.2.12 ""+""	107
6.17.2.13 ""+""	107
6.17.2.14 ""+""	108
6.17.2.15 ""+""	108
6.17.2.16 ""+""	108
6.17.2.17 ""+""	108
6.17.2.18 ""+""	108
6.17.2.19 ""+""	108
6.17.2.20 ""+""	108
6.17.2.21 ""-""	108
6.17.2.22 ""-""	108
6.17.2.23 ""-""	108
6.17.2.24 ""-""	108
6.17.2.25 ""-""	108
6.17.2.26 ""-""	109
6.17.2.27 ""-""	109
6.17.2.28 ""-""	109
6.17.2.29 ""-""	109

6.17.2.30 ""-""	109
6.17.2.31 ""-""	109
6.17.2.32 ""/""	109
6.17.2.33 ""/""	109
6.17.2.34 ""/""	109
6.17.2.35 ""/""	109
6.17.2.36 ""/""	109
6.17.2.37 ""/""	109
6.17.2.38 ""/""	110
6.17.2.39 ""/""	110
6.17.2.40 ""/""	110
6.17.2.41 ""/""	110
6.17.2.42 ""/=""	110
6.17.2.43 ""/=""	110
6.17.2.44 ""/=""	110
6.17.2.45 ""/=""	110
6.17.2.46 ""/=""	110
6.17.2.47 ""/=""	110
6.17.2.48 ""/=""	110
6.17.2.49 ""/=""	110
6.17.2.50 ""/=""	111
6.17.2.51 ""/=""	111
6.17.2.52 ""<""	111
6.17.2.53 ""<""	111
6.17.2.54 ""<""	111
6.17.2.55 ""<""	111
6.17.2.56 ""<""	111
6.17.2.57 ""<""	111
6.17.2.58 ""<""	111
6.17.2.59 ""<""	111
6.17.2.60 ""<""	111
6.17.2.61 ""<""	111
6.17.2.62 ""<=""	112
6.17.2.63 ""<=""	112
6.17.2.64 ""<=""	112
6.17.2.65 ""<=""	112
6.17.2.66 ""<=""	112
6.17.2.67 ""<=""	112
6.17.2.68 ""<=""	112
6.17.2.69 ""<=""	112

6.17.2.70 ""<=""	112
6.17.2.71 ""<=""	112
6.17.2.72 ""==""	112
6.17.2.73 ""==""	112
6.17.2.74 ""==""	113
6.17.2.75 ""==""	113
6.17.2.76 ""==""	113
6.17.2.77 ""==""	113
6.17.2.78 ""==""	113
6.17.2.79 ""==""	113
6.17.2.80 ""==""	113
6.17.2.81 ""==""	113
6.17.2.82 "">""	113
6.17.2.83 "">""	113
6.17.2.84 "">""	113
6.17.2.85 "">""	113
6.17.2.86 "">""	114
6.17.2.87 "">""	114
6.17.2.88 "">""	114
6.17.2.89 "">""	114
6.17.2.90 "">""	114
6.17.2.91 "">""	114
6.17.2.92 "">=""	114
6.17.2.93 "">=""	114
6.17.2.94 "">=""	114
6.17.2.95 "">=""	114
6.17.2.96 "">=""	114
6.17.2.97 "">=""	114
6.17.2.98 "">=""	115
6.17.2.99 "">=""	115
6.17.2.100 "">=""	115
6.17.2.101 "">=""	115
6.17.2.102 "abs"	115
6.17.2.103 "and"	115
6.17.2.104 "and"	115
6.17.2.105 "and"	115
6.17.2.106 "and"	115
6.17.2.107 "and"	115
6.17.2.108 "and"	115
6.17.2.109 "mod"	115

6.17.2.110"mod""	116
6.17.2.111"mod""	116
6.17.2.112"mod""	116
6.17.2.113"mod""	116
6.17.2.114"mod""	116
6.17.2.115"mod""	116
6.17.2.116"mod""	116
6.17.2.117"mod""	116
6.17.2.118"mod""	116
6.17.2.119"nand""	116
6.17.2.120"nand""	116
6.17.2.121"nand""	116
6.17.2.122"nand""	117
6.17.2.123"nand""	117
6.17.2.124"nand""	117
6.17.2.125"nor""	117
6.17.2.126"nor""	117
6.17.2.127"nor""	117
6.17.2.128"nor""	117
6.17.2.129"nor""	117
6.17.2.130"nor""	117
6.17.2.131"not""	117
6.17.2.132"not""	117
6.17.2.133"or""	117
6.17.2.134"or""	118
6.17.2.135"or""	118
6.17.2.136"or""	118
6.17.2.137"or""	118
6.17.2.138"or""	118
6.17.2.139"rem""	118
6.17.2.140"rem""	118
6.17.2.141"rem""	118
6.17.2.142"rem""	118
6.17.2.143"rem""	118
6.17.2.144"rem""	118
6.17.2.145"rem""	118
6.17.2.146"rem""	119
6.17.2.147"rem""	119
6.17.2.148"rem""	119
6.17.2.149"rol""	119

6.17.2.190?<=\	122
6.17.2.191?<=\	122
6.17.2.192?<=\	122
6.17.2.193?<\	123
6.17.2.194?<\	123
6.17.2.195?<\	123
6.17.2.196?<\	123
6.17.2.197?<\	123
6.17.2.198?<\	123
6.17.2.199?<\	123
6.17.2.200?<\	123
6.17.2.201?<\	123
6.17.2.202?<\	123
6.17.2.203?=\	123
6.17.2.204?=\	123
6.17.2.205?=\	124
6.17.2.206?=\	124
6.17.2.207?=\	124
6.17.2.208?=\	124
6.17.2.209?=\	124
6.17.2.210?=\	124
6.17.2.211?=\	124
6.17.2.212?=\	124
6.17.2.213?>=\	124
6.17.2.214?>=\	124
6.17.2.215?>=\	124
6.17.2.216?>=\	124
6.17.2.217?>=\	125
6.17.2.218?>=\	125
6.17.2.219?>=\	125
6.17.2.220?>=\	125
6.17.2.221?>=\	125
6.17.2.222?>=\	125
6.17.2.223?>\	125
6.17.2.224?>\	125
6.17.2.225?>\	125
6.17.2.226?>\	125
6.17.2.227?>\	125
6.17.2.228?>\	125
6.17.2.229?>\	126

6.17.2.230?>\	126
6.17.2.231?>\	126
6.17.2.232?>\	126
6.17.2.233add_carry	126
6.17.2.234add_carry	126
6.17.2.235and_reduce	126
6.17.2.236and_reduce	126
6.17.2.237divide	126
6.17.2.238divide	126
6.17.2.239find_leftmost	126
6.17.2.240find_leftmost	127
6.17.2.241find_rightmost	127
6.17.2.242find_rightmost	127
6.17.2.243from_hstring	127
6.17.2.244from_hstring	127
6.17.2.245from_hstring	127
6.17.2.246from_hstring	127
6.17.2.247from_hstring	127
6.17.2.248from_hstring	127
6.17.2.249from_ostring	127
6.17.2.250from_ostring	127
6.17.2.251from_ostring	128
6.17.2.252from_ostring	128
6.17.2.253from_ostring	128
6.17.2.254from_ostring	128
6.17.2.255from_string	128
6.17.2.256from_string	128
6.17.2.257from_string	128
6.17.2.258from_string	128
6.17.2.259from_string	128
6.17.2.260from_string	128
6.17.2.261HREAD	128
6.17.2.262HREAD	129
6.17.2.263HREAD	129
6.17.2.264HREAD	129
6.17.2.265HWRITE	129
6.17.2.266HWRITE	129
6.17.2.267ls_Negative	129
6.17.2.268ls_X	129
6.17.2.269ls_X	129

6.17.2.270maximum	129
6.17.2.271maximum	129
6.17.2.272maximum	129
6.17.2.273maximum	130
6.17.2.274maximum	130
6.17.2.275maximum	130
6.17.2.276maximum	130
6.17.2.277maximum	130
6.17.2.278maximum	130
6.17.2.279maximum	130
6.17.2.280minimum	130
6.17.2.281minimum	130
6.17.2.282minimum	130
6.17.2.283minimum	130
6.17.2.284minimum	130
6.17.2.285minimum	131
6.17.2.286minimum	131
6.17.2.287minimum	131
6.17.2.288minimum	131
6.17.2.289minimum	131
6.17.2.290modulo	131
6.17.2.291modulo	131
6.17.2.292hand_reduce	131
6.17.2.293hand_reduce	131
6.17.2.294hor_reduce	131
6.17.2.295hor_reduce	131
6.17.2.296or_reduce	132
6.17.2.297or_reduce	132
6.17.2.298OREAD	132
6.17.2.299OREAD	132
6.17.2.300OREAD	132
6.17.2.301OREAD	132
6.17.2.302OWRITE	132
6.17.2.303OWRITE	132
6.17.2.304READ	132
6.17.2.305READ	132
6.17.2.306READ	132
6.17.2.307READ	133
6.17.2.308reciprocal	133
6.17.2.309reciprocal	133

6.17.2.310remainder	133
6.17.2.311remainder	133
6.17.2.312resize	133
6.17.2.313resize	133
6.17.2.314resize	133
6.17.2.315resize	133
6.17.2.316saturate	134
6.17.2.317saturate	134
6.17.2.318saturate	134
6.17.2.319saturate	134
6.17.2.320scalb	134
6.17.2.321scalb	134
6.17.2.322scalb	134
6.17.2.323scalb	134
6.17.2.324SFix_high	134
6.17.2.325SFix_low	134
6.17.2.326sfixed_high	134
6.17.2.327sfixed_high	135
6.17.2.328sfixed_low	135
6.17.2.329sfixed_low	135
6.17.2.330SHIFT_LEFT	135
6.17.2.331SHIFT_LEFT	135
6.17.2.332SHIFT_RIGHT	135
6.17.2.333SHIFT_RIGHT	135
6.17.2.334std_match	135
6.17.2.335std_match	135
6.17.2.336to_01	135
6.17.2.337to_01	136
6.17.2.338to_hstring	136
6.17.2.339to_hstring	136
6.17.2.340to_integer	136
6.17.2.341to_integer	136
6.17.2.342to_ostring	136
6.17.2.343to_ostring	136
6.17.2.344to_real	136
6.17.2.345to_real	136
6.17.2.346to_SFix	136
6.17.2.347to_SFix	136
6.17.2.348to_sfixed	137
6.17.2.349to_sfixed	137

6.17.2.350to_sfixed	137
6.17.2.351to_sfixed	137
6.17.2.352to_sfixed	137
6.17.2.353to_sfixed	137
6.17.2.354to_sfixed	137
6.17.2.355to_sfixed	137
6.17.2.356to_sfixed	137
6.17.2.357to_sfixed	138
6.17.2.358to_sfixed	138
6.17.2.359to_sfixed	138
6.17.2.360to_signed	138
6.17.2.361to_signed	138
6.17.2.362to_slv	138
6.17.2.363to_slv	138
6.17.2.364to_string	138
6.17.2.365to_string	138
6.17.2.366to_sulv	138
6.17.2.367to_sulv	138
6.17.2.368to_UFix	139
6.17.2.369to_UFix	139
6.17.2.370to_ufixed	139
6.17.2.371to_ufixed	139
6.17.2.372to_ufixed	139
6.17.2.373to_ufixed	139
6.17.2.374to_ufixed	139
6.17.2.375to_ufixed	139
6.17.2.376to_ufixed	139
6.17.2.377to_ufixed	140
6.17.2.378to_ufixed	140
6.17.2.379to_ufixed	140
6.17.2.380to_ufixed	140
6.17.2.381to_unsigned	140
6.17.2.382to_unsigned	140
6.17.2.383to_UX01	140
6.17.2.384to_UX01	140
6.17.2.385to_X01	140
6.17.2.386to_X01	140
6.17.2.387to_X01Z	140
6.17.2.388to_X01Z	141
6.17.2.389UFix_high	141

6.17.2.390UFix_low	141
6.17.2.391ufixed_high	141
6.17.2.392ufixed_high	141
6.17.2.393ufixed_low	141
6.17.2.394ufixed_low	141
6.17.2.395WRITE	141
6.17.2.396WRITE	141
6.17.2.397xnor_reduce	141
6.17.2.398xnor_reduce	141
6.17.2.399xor_reduce	142
6.17.2.400xor_reduce	142
6.17.3 Member Data Documentation	142
6.17.3.1 BINARY_READ	142
6.17.3.2 BINARY_READ	142
6.17.3.3 BINARY_READ	142
6.17.3.4 BINARY_READ	142
6.17.3.5 BINARY_WRITE	142
6.17.3.6 BINARY_WRITE	142
6.17.3.7 bread	142
6.17.3.8 bread	142
6.17.3.9 bread	142
6.17.3.10 bread	142
6.17.3.11 bwrite	143
6.17.3.12 bwrite	143
6.17.3.13 fixed_float_types	143
6.17.3.14 fixed_guard_bits	143
6.17.3.15 fixed_overflow_style	143
6.17.3.16 fixed_round_style	143
6.17.3.17 from_binary_string	143
6.17.3.18 from_binary_string	143
6.17.3.19 from_binary_string	143
6.17.3.20 from_binary_string	143
6.17.3.21 from_binary_string	143
6.17.3.22 from_binary_string	144
6.17.3.23 from_bstring	144
6.17.3.24 from_bstring	144
6.17.3.25 from_bstring	144
6.17.3.26 from_bstring	144
6.17.3.27 from_bstring	144
6.17.3.28 from_bstring	144

6.17.3.29 from_hex_string	144
6.17.3.30 from_hex_string	144
6.17.3.31 from_hex_string	144
6.17.3.32 from_hex_string	144
6.17.3.33 from_hex_string	145
6.17.3.34 from_hex_string	145
6.17.3.35 from_octal_string	145
6.17.3.36 from_octal_string	145
6.17.3.37 from_octal_string	145
6.17.3.38 from_octal_string	145
6.17.3.39 from_octal_string	145
6.17.3.40 from_octal_string	145
6.17.3.41 HEX_READ	145
6.17.3.42 HEX_READ	145
6.17.3.43 HEX_READ	145
6.17.3.44 HEX_READ	146
6.17.3.45 HEX_WRITE	146
6.17.3.46 HEX_WRITE	146
6.17.3.47 IEEE	146
6.17.3.48 IEEE_PROPOSED	146
6.17.3.49 no_warning	146
6.17.3.50 NUMERIC_STD	146
6.17.3.51 OCTAL_READ	146
6.17.3.52 OCTAL_READ	146
6.17.3.53 OCTAL_READ	146
6.17.3.54 OCTAL_READ	146
6.17.3.55 OCTAL_WRITE	146
6.17.3.56 OCTAL_WRITE	147
6.17.3.57 sfixed	147
6.17.3.58 STD_LOGIC_1164	147
6.17.3.59 TEXTIO	147
6.17.3.60 TO_BINARY_STRING	147
6.17.3.61 TO_BINARY_STRING	147
6.17.3.62 to_bstring	147
6.17.3.63 to_bstring	147
6.17.3.64 TO_HEX_STRING	147
6.17.3.65 TO_HEX_STRING	147
6.17.3.66 TO_OCTAL_STRING	147
6.17.3.67 TO_OCTAL_STRING	147
6.17.3.68 to_Std_Logic_Vector	148

6.17.3.69 to_Std_Logic_Vector	148
6.17.3.70 to_Std_ULogic_Vector	148
6.17.3.71 to_Std_ULogic_Vector	148
6.17.3.72 to_StdLogicVector	148
6.17.3.73 to_StdLogicVector	148
6.17.3.74 to_StdULogicVector	148
6.17.3.75 to_StdULogicVector	148
6.17.3.76 U_sfixed	148
6.17.3.77 U_ufixed	148
6.17.3.78 ufixed	148
6.17.3.79 UNRESOLVED_sfixed	148
6.17.3.80 UNRESOLVED_ufixed	149
6.18 integrador Entity Reference	149
6.18.1 Detailed Description	149
6.18.2 Member Data Documentation	150
6.18.2.1 clk	150
6.18.2.2 en	150
6.18.2.3 fixed_pkg	150
6.18.2.4 IEEE	150
6.18.2.5 int_data	150
6.18.2.6 MAX	150
6.18.2.7 Nin	150
6.18.2.8 Nout	150
6.18.2.9 out_data	150
6.18.2.10 reset	150
6.18.2.11 sinc	150
6.18.2.12 STD_LOGIC_1164	150
6.18.2.13 STD_LOGIC_UNSIGNED	151
6.19 integrador Architecture Reference	151
6.19.1 Detailed Description	151
6.19.2 Member Function Documentation	151
6.19.2.1 PROCESS_10	151
6.19.3 Member Data Documentation	151
6.19.3.1 out_int	151
6.19.3.2 sinc_int	151
6.20 LEDs Entity Reference	151
6.20.1 Detailed Description	152
6.20.2 Member Data Documentation	152
6.20.2.1 CLOCK_50	152
6.20.2.2 ieee	152

6.20.2.3	KEY	152
6.20.2.4	LED	152
6.20.2.5	std_logic_1164	152
6.20.2.6	std_logic_arith	152
6.20.2.7	std_logic_unsigned	152
6.20.2.8	SW	153
6.21	LEDs Architecture Reference	153
6.21.1	Detailed Description	153
6.21.2	Member Function Documentation	153
6.21.2.1	PROCESS_11	153
6.21.3	Member Data Documentation	153
6.21.3.1	blink	153
6.21.3.2	BLINK_FREQ	153
6.21.3.3	CLK_FREQ	153
6.21.3.4	cnt	154
6.21.3.5	CNT_MAX	154
6.22	MAIN Architecture Reference	154
6.22.1	Detailed Description	156
6.22.2	Member Function Documentation	156
6.22.2.1	PROCESS_4	156
6.22.2.2	PROCESS_5	156
6.22.3	Member Data Documentation	156
6.22.3.1	amostragem_moduladoras	156
6.22.3.2	bidir	156
6.22.3.3	clk_div	157
6.22.3.4	clk_int	157
6.22.3.5	clk_led	157
6.22.3.6	clk_pll	157
6.22.3.7	clk_vf	157
6.22.3.8	clk_wt	157
6.22.3.9	contador	157
6.22.3.10	cPWM1	157
6.22.3.11	cPWM2	157
6.22.3.12	cPWM3	157
6.22.3.13	cTRI1	157
6.22.3.14	cTRI2	157
6.22.3.15	cTRI3	158
6.22.3.16	cTRI4	158
6.22.3.17	cTRI5	158
6.22.3.18	cTRI6	158

6.22.3.19 dirPWM1	158
6.22.3.20 dirPWM2	158
6.22.3.21 dirPWM3	158
6.22.3.22 dirTRI1	158
6.22.3.23 dirTRI2	158
6.22.3.24 dirTRI3	158
6.22.3.25 dirTRI4	158
6.22.3.26 dirTRI5	158
6.22.3.27 dirTRI6	159
6.22.3.28 en_PWM	159
6.22.3.29 en_PWMA	159
6.22.3.30 en_PWMB	159
6.22.3.31 en_PWMC	159
6.22.3.32 err_FA	159
6.22.3.33 err_FABC	159
6.22.3.34 err_FB	159
6.22.3.35 err_FC	159
6.22.3.36 fbpspwmdt	159
6.22.3.37 incVF	159
6.22.3.38 integrador	159
6.22.3.39 key0_ant	160
6.22.3.40 key1_ant	160
6.22.3.41 LEDs	160
6.22.3.42 ma	160
6.22.3.43 mb	160
6.22.3.44 mc	160
6.22.3.45 moduladoras	160
6.22.3.46 mVF	160
6.22.3.47 omega_pll	160
6.22.3.48 OSC_BUS1	160
6.22.3.49 pll	160
6.22.3.50 pll_lock	160
6.22.3.51 portadora_tringular	161
6.22.3.52 portadoras	161
6.22.3.53 pulso_key0	161
6.22.3.54 pulso_key1	161
6.22.3.55 pwm1_fa01	161
6.22.3.56 pwm1_fa02	161
6.22.3.57 pwm1_fa03	161
6.22.3.58 pwm1_fb01	161

6.22.3.59 pwm1_fb02	161
6.22.3.60 pwm1_fb03	161
6.22.3.61 pwm1_fc01	161
6.22.3.62 pwm1_fc02	161
6.22.3.63 pwm1_fc03	162
6.22.3.64 pwm2_fa01	162
6.22.3.65 pwm2_fa02	162
6.22.3.66 pwm2_fa03	162
6.22.3.67 pwm2_fb01	162
6.22.3.68 pwm2_fb02	162
6.22.3.69 pwm2_fb03	162
6.22.3.70 pwm2_fc01	162
6.22.3.71 pwm2_fc02	162
6.22.3.72 pwm2_fc03	162
6.22.3.73 reset	162
6.22.3.74 rst	162
6.22.3.75 sigPWM01	163
6.22.3.76 sigPWM02	163
6.22.3.77 sin_a	163
6.22.3.78 sin_b	163
6.22.3.79 sin_c	163
6.22.3.80 sinc_int	163
6.22.3.81 sinc_wt	163
6.22.3.82 tabela_sin	163
6.22.3.83 th_a	163
6.22.3.84 th_b	163
6.22.3.85 th_c	163
6.22.3.86 theta_abc	163
6.22.3.87 theta_pll	164
6.22.3.88 theta_wt	164
6.22.3.89 toggle_key0	164
6.22.3.90 toggle_key1	164
6.22.3.91 u1	164
6.22.3.92 u5	164
6.22.3.93 u6	164
6.22.3.94 uclkvf	164
6.22.3.95 ucr1	164
6.22.3.96 ucr2	164
6.22.3.97 ucr3	164
6.22.3.98 ucr4	164

6.22.3.99 ucr5	165
6.22.3.100ucr6	165
6.22.3.101upll	165
6.22.3.102usin_a	165
6.22.3.103usin_b	165
6.22.3.104usin_c	165
6.22.3.105uvf	165
6.22.3.106fcontrol	165
6.23 my_types_pkg Package Reference	165
6.23.1 Detailed Description	166
6.23.2 Member Data Documentation	166
6.23.2.1 ARRAY_10_24	166
6.23.2.2 ARRAY_10_6	166
6.23.2.3 ARRAY_12_24	166
6.23.2.4 ARRAY_12_6	166
6.23.2.5 ARRAY_16_16	167
6.23.2.6 ARRAY_16_24	167
6.23.2.7 ARRAY_16_6	167
6.23.2.8 ARRAY_16x6_12	167
6.23.2.9 ARRAY_17_24	167
6.23.2.10 ARRAY_18_24	167
6.23.2.11 ARRAY_20_24	167
6.23.2.12 ARRAY_21_24	167
6.23.2.13 ARRAY_24_24	167
6.23.2.14 ARRAY_2_24	167
6.23.2.15 ARRAY_3_16	167
6.23.2.16 ARRAY_3_6	167
6.23.2.17 ARRAY_4_6	168
6.23.2.18 ARRAY_8_24	168
6.23.2.19 ARRAY_8_5_0	168
6.23.2.20 ARRAY_8_6	168
6.23.2.21 ARRAY_8_8	168
6.23.2.22 ARRAY_9_24	168
6.23.2.23 ARRAY_9_6	168
6.23.2.24 COMP_ARRAY	168
6.23.2.25 IEEE	168
6.23.2.26 OPP_GAMMA	168
6.23.2.27 OPP_PULSO	168
6.23.2.28 STD_LOGIC_1164	168
6.23.2.29 STD_LOGIC_UNSIGNED	169

6.23.2.30 WORD_ARRAY	169
6.24 pll Entity Reference	169
6.24.1 Detailed Description	169
6.24.2 Member Data Documentation	170
6.24.2.1 all	170
6.24.2.2 altera_mf	170
6.24.2.3 areset	170
6.24.2.4 c0	170
6.24.2.5 ieee	170
6.24.2.6 inclk0	170
6.24.2.7 locked	170
6.24.2.8 std_logic_1164	170
6.25 portadora_tringular Entity Reference	170
6.25.1 Detailed Description	171
6.25.2 Member Data Documentation	171
6.25.2.1 c	171
6.25.2.2 clk	171
6.25.2.3 count_ini	171
6.25.2.4 dir	171
6.25.2.5 dir_ini	171
6.25.2.6 en	172
6.25.2.7 fixed_pkg	172
6.25.2.8 IEEE	172
6.25.2.9 MAX	172
6.25.2.10 N	172
6.25.2.11 reset	172
6.25.2.12 STD_LOGIC_1164	172
6.25.2.13 STD_LOGIC_UNSIGNED	172
6.26 portadora_tringular Architecture Reference	172
6.26.1 Detailed Description	172
6.26.2 Member Function Documentation	173
6.26.2.1 PROCESS_12	173
6.26.3 Member Data Documentation	173
6.26.3.1 c_int	173
6.26.3.2 dir_int	173
6.27 sinewave Entity Reference	173
6.27.1 Detailed Description	173
6.27.2 Member Data Documentation	173
6.27.2.1 clk	173
6.27.2.2 dataout	174

6.27.2.3 IEEE	174
6.27.2.4 NUMERIC_STD	174
6.27.2.5 STD_LOGIC_1164	174
6.28 SYN Architecture Reference	174
6.28.1 Detailed Description	174
6.28.2 Member Data Documentation	174
6.28.2.1 altpll	174
6.28.2.2 altpll_component	175
6.28.2.3 sub_wire0	175
6.28.2.4 sub_wire1	175
6.28.2.5 sub_wire2	175
6.28.2.6 sub_wire2_bv	175
6.28.2.7 sub_wire3	175
6.28.2.8 sub_wire4	175
6.28.2.9 sub_wire5	175
6.29 tabela_sin Architecture Reference	175
6.29.1 Detailed Description	176
6.29.2 Member Function Documentation	176
6.29.2.1 PROCESS_14	176
6.29.3 Member Data Documentation	176
6.29.3.1 id	176
6.29.3.2 sin	176
6.29.3.3 va_Q14	176
6.30 tabela_sin Entity Reference	176
6.30.1 Detailed Description	177
6.30.2 Member Data Documentation	177
6.30.2.1 clk	177
6.30.2.2 F	177
6.30.2.3 fixed_pkg	177
6.30.2.4 I	177
6.30.2.5 ieee	177
6.30.2.6 ma	177
6.30.2.7 MAX	178
6.30.2.8 n_bits_c	178
6.30.2.9 n_bits_phase	178
6.30.2.10 numeric_std	178
6.30.2.11 std_logic_1164	178
6.30.2.12 std_logic_unsigned	178
6.30.2.13 theta	178
6.30.2.14 va	178

6.31 theta_abc Architecture Reference	178
6.31.1 Detailed Description	179
6.31.2 Member Function Documentation	179
6.31.2.1 PROCESS_15	179
6.31.3 Member Data Documentation	179
6.31.3.1 th_a	179
6.31.3.2 th_ai	179
6.31.3.3 th_b	179
6.31.3.4 th_bi	179
6.31.3.5 th_c	179
6.31.3.6 th_ci	179
6.32 theta_abc Entity Reference	179
6.32.1 Detailed Description	180
6.32.2 Member Data Documentation	180
6.32.2.1 clk	180
6.32.2.2 en	180
6.32.2.3 fixed_pkg	180
6.32.2.4 IEEE	180
6.32.2.5 Nin	181
6.32.2.6 Nout	181
6.32.2.7 numeric_std	181
6.32.2.8 reset	181
6.32.2.9 STD_LOGIC_1164	181
6.32.2.10 STD_LOGIC_UNSIGNED	181
6.32.2.11 theta_a	181
6.32.2.12 theta_b	181
6.32.2.13 theta_c	181
6.32.2.14 theta_in	181
6.33 vfcontrol Entity Reference	181
6.33.1 Detailed Description	182
6.33.2 Member Data Documentation	182
6.33.2.1 clk	182
6.33.2.2 en	183
6.33.2.3 F	183
6.33.2.4 fixed_pkg	183
6.33.2.5 I	183
6.33.2.6 IEEE	183
6.33.2.7 inc_data	183
6.33.2.8 incMAX	183
6.33.2.9 incMIN	183

6.33.2.10 m_vf	183
6.33.2.11 mMAX	183
6.33.2.12 mMIN	183
6.33.2.13 n_bits_c	183
6.33.2.14 numeric_std	184
6.33.2.15 STD_LOGIC_1164	184
6.33.2.16 STD_LOGIC_ARITH	184
6.33.2.17 STD_LOGIC_UNSIGNED	184
6.34 vfcontrol_arch Architecture Reference	184
6.34.1 Detailed Description	184
6.34.2 Member Function Documentation	184
6.34.2.1 PROCESS_16	184
6.34.3 Member Data Documentation	184
6.34.3.1 incsignal	184
6.34.3.2 incstep	185
6.34.3.3 msignal	185
6.34.3.4 mstep	185
6.35 wt Entity Reference	185
6.35.1 Detailed Description	185
6.35.2 Member Data Documentation	186
6.35.2.1 clk	186
6.35.2.2 en	186
6.35.2.3 fixed_pkg	186
6.35.2.4 IEEE	186
6.35.2.5 int_data	186
6.35.2.6 MAX	186
6.35.2.7 Nbits	186
6.35.2.8 out_data	186
6.35.2.9 reset	186
6.35.2.10 sinc	186
6.35.2.11 STD_LOGIC_1164	186
6.35.2.12 STD_LOGIC_UNSIGNED	187
6.36 wt_arch Architecture Reference	187
6.36.1 Detailed Description	187
6.36.2 Member Function Documentation	187
6.36.2.1 PROCESS_17	187
6.36.3 Member Data Documentation	187
6.36.3.1 out_int	187
6.36.3.2 sinc_int	187

7 File Documentation	189
7.1 clk_div.vhd File Reference	189
7.2 clk_div.vhd	189
7.3 comparador.vhd File Reference	189
7.4 comparador.vhd	190
7.5 comparadores.vhd File Reference	190
7.6 comparadores.vhd	190
7.7 contador.vhd File Reference	191
7.8 contador.vhd	191
7.9 DE0_NANO_VF.qsf File Reference	192
7.9.1 Variable Documentation	194
7.9.1.1 CDF_FILE	194
7.9.1.2 CDF_FILE	194
7.9.1.3 CDF_FILE	195
7.9.1.4 CDF_FILE	195
7.9.1.5 CDF_FILE	195
7.9.1.6 CDF_FILE	195
7.9.1.7 CDF_FILE	195
7.9.1.8 DEVICE	195
7.9.1.9 EDA_OUTPUT_DATA_FORMAT	195
7.9.1.10 EDA_SIMULATION_TOOL	195
7.9.1.11 ERROR_CHECK_FREQUENCY_DIVISOR	195
7.9.1.12 FAMILY	195
7.9.1.13 LAST_QUARTUS_VERSION	195
7.9.1.14 MAX_CORE_JUNCTION_TEMP	195
7.9.1.15 MIN_CORE_JUNCTION_TEMP	196
7.9.1.16 NOMINAL_CORE_SUPPLY_VOLTAGE	196
7.9.1.17 ORIGINAL_QUARTUS_VERSION	196
7.9.1.18 PARTITION_COLOR	196
7.9.1.19 PARTITION_FITTER_PRESERVATION_LEVEL	196
7.9.1.20 PARTITION_NETLIST_TYPE	196
7.9.1.21 PIN_A11	196
7.9.1.22 PIN_A12	196
7.9.1.23 PIN_A13	196
7.9.1.24 PIN_A15	196
7.9.1.25 PIN_A2	196
7.9.1.26 PIN_A3	196
7.9.1.27 PIN_A4	197
7.9.1.28 PIN_A5	197
7.9.1.29 PIN_A6	197

7.9.1.30 PIN_A7	197
7.9.1.31 PIN_B11	197
7.9.1.32 PIN_B12	197
7.9.1.33 PIN_B13	197
7.9.1.34 PIN_B1	197
7.9.1.35 PIN_B3	197
7.9.1.36 PIN_B4	197
7.9.1.37 PIN_B5	197
7.9.1.38 PIN_B6	197
7.9.1.39 PIN_B7	198
7.9.1.40 PIN_B9	198
7.9.1.41 PIN_C11	198
7.9.1.42 PIN_C3	198
7.9.1.43 PIN_C6	198
7.9.1.44 PIN_C8	198
7.9.1.45 PIN_C9	198
7.9.1.46 PIN_D11	198
7.9.1.47 PIN_D12	198
7.9.1.48 PIN_D1	198
7.9.1.49 PIN_D3	198
7.9.1.50 PIN_D5	198
7.9.1.51 PIN_D6	199
7.9.1.52 PIN_D8	199
7.9.1.53 PIN_D9	199
7.9.1.54 PIN_E10	199
7.9.1.55 PIN_E11	199
7.9.1.56 PIN_E1	199
7.9.1.57 PIN_E6	199
7.9.1.58 PIN_E7	199
7.9.1.59 PIN_E8	199
7.9.1.60 PIN_E9	199
7.9.1.61 PIN_F3	199
7.9.1.62 PIN_F8	199
7.9.1.63 PIN_F9	200
7.9.1.64 PIN_J15	200
7.9.1.65 PIN_K16	200
7.9.1.66 PIN_L13	200
7.9.1.67 PIN_L14	200
7.9.1.68 PIN_L15	200
7.9.1.69 PIN_L16	200

7.9.1.70 PIN_L3	200
7.9.1.71 PIN_M10	200
7.9.1.72 PIN_M15	200
7.9.1.73 PIN_M1	200
7.9.1.74 PIN_N11	200
7.9.1.75 PIN_N12	201
7.9.1.76 PIN_N14	201
7.9.1.77 PIN_N15	201
7.9.1.78 PIN_N9	201
7.9.1.79 PIN_P11	201
7.9.1.80 PIN_P14	201
7.9.1.81 PIN_P15	201
7.9.1.82 PIN_P16	201
7.9.1.83 PIN_P9	201
7.9.1.84 PIN_R10	201
7.9.1.85 PIN_R11	201
7.9.1.86 PIN_R12	201
7.9.1.87 PIN_R13	202
7.9.1.88 PIN_R16	202
7.9.1.89 PIN_R8	202
7.9.1.90 PIN_T10	202
7.9.1.91 PIN_T11	202
7.9.1.92 PIN_T12	202
7.9.1.93 PIN_T8	202
7.9.1.94 POWER_BOARD_THERMAL_MODEL	202
7.9.1.95 POWER_PRESET_COOLING_SOLUTION	202
7.9.1.96 PROJECT_CREATION_TIME_DATE	202
7.9.1.97 PROJECT_OUTPUT_DIRECTORY	202
7.9.1.98 QIP_FILE	202
7.9.1.99 STRATIX_DEVICE_IO_STANDARD	203
7.9.1.100 TOP_LEVEL_ENTITY	203
7.9.1.101 VHDL_FILE	203
7.9.1.102 VHDL_FILE	203
7.9.1.103 VHDL_FILE	203
7.9.1.104 VHDL_FILE	203
7.9.1.105 VHDL_FILE	203
7.9.1.106 VHDL_FILE	203
7.9.1.107 VHDL_FILE	203
7.9.1.108 VHDL_FILE	203
7.9.1.109 VHDL_FILE	203

7.9.1.110 VHDL_FILE	203
7.9.1.111 VHDL_FILE	204
7.9.1.112 VHDL_FILE	204
7.9.1.113 VHDL_FILE	204
7.9.1.114 VHDL_FILE	204
7.10 DE0_NANO_VF.qsf	204
7.11 DE0_NANO_VF.vhd File Reference	206
7.12 DE0_NANO_VF.vhd	206
7.13 deadtime.vhd File Reference	214
7.14 deadtime.vhd	214
7.15 fbpsspwmdt.vhd File Reference	215
7.16 fbpsspwmdt.vhd	215
7.17 fixed_float_types_c.vhdl File Reference	217
7.18 fixed_float_types_c.vhdl	217
7.19 fixed_pkg_c.vhdl File Reference	217
7.20 fixed_pkg_c.vhdl	217
7.21 integrador.vhd File Reference	316
7.22 integrador.vhd	317
7.23 LEDs.vhd File Reference	317
7.24 LEDs.vhd	318
7.25 my_types_pkg.vhd File Reference	318
7.26 my_types_pkg.vhd	318
7.27 pll.vhd File Reference	319
7.28 pll.vhd	319
7.29 portadora_tringular.vhd File Reference	323
7.30 portadora_tringular.vhd	324
7.31 sinewave.vhd File Reference	324
7.32 sinewave.vhd	324
7.33 tabela_sin.m File Reference	325
7.33.1 Function Documentation	325
7.33.1.1 fclose	325
7.33.1.2 fprintf	325
7.33.1.3 fprintf	325
7.33.1.4 fprintf	325
7.33.1.5 fprintf	325
7.33.1.6 fprintf	325
7.33.1.7 fprintf	325
7.33.1.8 fprintf	325
7.33.1.9 log2	325
7.33.1.10 log2	325

7.33.1.11 log2	326
7.33.1.12 plot	326
7.33.1.13 plotar	326
7.33.2 Variable Documentation	326
7.33.2.1 a	326
7.33.2.2 amp	326
7.33.2.3 cmax	326
7.33.2.4 comp	326
7.33.2.5 duty	326
7.33.2.6 duty60Hz	326
7.33.2.7 fid	326
7.33.2.8 fimax	326
7.33.2.9 fin	326
7.33.2.10 fint	326
7.33.2.11 fixo	327
7.33.2.12 fout	327
7.33.2.13 foutMin	327
7.33.2.14 id	327
7.33.2.15 idfi	327
7.33.2.16 INC60Hz	327
7.33.2.17 INCdata	327
7.33.2.18 INCmax	327
7.33.2.19 INCmin	327
7.33.2.20 int_data	327
7.33.2.21 k	327
7.33.2.22 MAX	328
7.33.2.23 max_phase	328
7.33.2.24 MAXmax	328
7.33.2.25 med	328
7.33.2.26 mMIN	328
7.33.2.27 N16	328
7.33.2.28 Nbits	328
7.33.2.29 Nin	328
7.33.2.30 Nout	329
7.33.2.31 out	329
7.33.2.32 senofi	329
7.33.2.33 slope	329
7.33.2.34 th17bits	329
7.33.2.35 th_ai	329
7.33.2.36 th_bi	330

7.33.2.37 th_ci	330
7.33.2.38 thasum	330
7.33.2.39 thbsum	330
7.33.2.40 thcsum	330
7.33.2.41 theta_a	330
7.33.2.42 theta_b	330
7.33.2.43 theta_c	330
7.33.2.44 wt	330
7.33.2.45 y	330
7.33.2.46 yQ15	330
7.34 tabela_sin.m	330
7.35 tabela_sin.vhd File Reference	333
7.36 tabela_sin.vhd	333
7.37 theta_abc.vhd File Reference	357
7.38 theta_abc.vhd	358
7.39 trash.vhd File Reference	358
7.40 trash.vhd	358
7.41 vfcontrol.vhd File Reference	360
7.42 vfcontrol.vhd	360
7.43 wt.vhd File Reference	361
7.44 wt.vhd	361

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

fixed_float_types	9
fixed_pkg	9
my_types_pkg	9

Chapter 2

Design Unit Index

2.1 Design Unit Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

comparadores	74
comparador	72
contador	77
DE0_NANO_VF	79
pll	169
clk_div	70
vfcontrol	181
integrador	149
portadora_tringular	170
theta_abc	179
tabela_sin	176
fbppspwm dt	87
deadtime	85
LEDs	151
sinewave	173
wt	185

Chapter 3

Design Unit Index

3.1 Design Unit List

Here is a list of all design unit members with links to the Entities they belong to:

architecture Behavioral	69
entity clk_div	70
architecture comparador	71
entity comparador	72
entity comparadores	74
architecture comparadores	76
architecture contador	77
entity contador	77
entity DE0_NANO_VF	79
entity deadtime	85
architecture deadtime_archetecture	86
architecture div	86
entity fbpspwmdt	87
architecture fbpspwmdt_arch	89
entity integrador	149
architecture integrador	151
entity LEDs	151
architecture LEDs	153
architecture MAIN	154
entity pll	169
entity portadora_tringular	170
architecture portadora_tringular	172
entity sinewave	173
architecture SYN	174
architecture tabela_sin	175
entity tabela_sin	176
architecture theta_abc	178
entity theta_abc	179
entity vfcontrol	181
architecture vfcontrol_arch	184
entity wt	185
architecture wt_arch	187

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

clk_div.vhd	189
comparador.vhd	189
comparadores.vhd	190
contador.vhd	191
DE0_NANO_VF.qsf	192
DE0_NANO_VF.vhd	206
deadtime.vhd	214
fbpspwmtd.vhd	215
fixed_float_types_c.vhdl	217
fixed_pkg_c.vhdl	217
integrador.vhd	316
LEDs.vhd	317
my_types_pkg.vhd	318
pll.vhd	319
portadora_tringular.vhd	323
sinewave.vhd	324
tabela_sin.m	325
tabela_sin.vhd	333
theta_abc.vhd	357
trash.vhd	358
vfcontrol.vhd	360
wt.vhd	361

Chapter 5

Namespace Documentation

5.1 fixed_float_types Namespace Reference

5.2 fixed_pkg Namespace Reference

5.3 my_types_pkg Namespace Reference

Chapter 6

Class Documentation

6.1 fixed_pkg Package Body Reference

Package >> [fixed_pkg](#)

Functions

- `integer maximum(I: in integer, r: in integer)`
- `integer minimum(I: in integer, r: in integer)`
- `SIGNED "sra"(arg: in SIGNED, count: in INTEGER)`
- `STD_LOGIC or_reduce(arg: in STD_ULOGIC_VECTOR)`
- `STD_LOGIC and_reduce(arg: in STD_ULOGIC_VECTOR)`
- `STD_ULOGIC xor_reduce(arg: in STD_ULOGIC_VECTOR)`
- `STD_ULOGIC nand_reduce(arg: in std_ulogic_vector)`
- `STD_ULOGIC nor_reduce(arg: in std_ulogic_vector)`
- `STD_ULOGIC xnor_reduce(arg: in std_ulogic_vector)`
- `STD_ULOGIC \?=\\(I: in STD_ULOGIC, r: in STD_ULOGIC)`
- `STD_ULOGIC \?/=\\(I: in STD_ULOGIC, r: in STD_ULOGIC)`
- `STD_ULOGIC \?=\\(L: in UNSIGNED, R: in UNSIGNED)`
- `std_ulogic \?=\\(L: in SIGNED, R: in SIGNED)`
- `std_ulogic \?/=\\(L: in UNSIGNED, R: in UNSIGNED)`
- `std_ulogic \?/=\\(L: in SIGNED, R: in SIGNED)`
- `BOOLEAN Is_X(s: in UNSIGNED)`
- `BOOLEAN Is_X(s: in SIGNED)`
- `STD_ULOGIC \?>\\(L: in UNSIGNED, R: in UNSIGNED)`
- `STD_ULOGIC \?>\\(L: in SIGNED, R: in SIGNED)`
- `STD_ULOGIC \?>=\\(L: in UNSIGNED, R: in UNSIGNED)`
- `STD_ULOGIC \?>=\\(L: in SIGNED, R: in SIGNED)`
- `STD_ULOGIC \?<\\(L: in UNSIGNED, R: in UNSIGNED)`
- `STD_ULOGIC \?<\\(L: in SIGNED, R: in SIGNED)`
- `STD_ULOGIC \?<=\\(L: in UNSIGNED, R: in UNSIGNED)`
- `STD_ULOGIC \?<=\\(L: in SIGNED, R: in SIGNED)`
- `INTEGER mins(I: in INTEGER, r: in INTEGER)`
- `INTEGER mine(I: in INTEGER, r: in INTEGER)`
- `UNRESOLVED_sfixed cleanvec(arg: in UNRESOLVED_sfixed)`
- `UNRESOLVED_ufixed cleanvec(arg: in UNRESOLVED_ufixed)`

- **UNRESOLVED_ufixed** `to_fixed(`
`arg: in UNSIGNED`
`constantleft_index: in INTEGER`
`constantright_index: in INTEGER`
`)`
- **UNRESOLVED_sfixed** `to_fixed(`
`arg: in SIGNED`
`constantleft_index: in INTEGER`
`constantright_index: in INTEGER`
`)`
- **UNSIGNED** `to_uns(arg: in UNRESOLVED_ufixed)`
- **SIGNED** `to_s(arg: in UNRESOLVED_sfixed)`
- **UNRESOLVED_ufixed** `round_fixed(`
`arg: in UNRESOLVED_ufixed`
`remainder: in UNRESOLVED_ufixed`
`overflow_style: in fixed_overflow_style_type fixed_overflow_style`
`)`
- **UNRESOLVED_sfixed** `round_fixed(`
`arg: in UNRESOLVED_sfixed`
`remainder: in UNRESOLVED_sfixed`
`overflow_style: in fixed_overflow_style_type fixed_overflow_style`
`)`
- **UNRESOLVED_ufixed** `to_ufixed(arg: in UNRESOLVED_sfixed)`
- **STD_ULOGIC_VECTOR** `to_suv(arg: in UNRESOLVED_ufixed)`
- **STD_ULOGIC_VECTOR** `to_suv(arg: in UNRESOLVED_sfixed)`
- **STD_LOGIC_VECTOR** `to_slv(arg: in UNRESOLVED_ufixed)`
- **STD_LOGIC_VECTOR** `to_slv(arg: in UNRESOLVED_sfixed)`
- **unresolved_ufixed** `to_ufixed(`
`arg: in STD_ULOGIC_VECTOR`
`constantleft_index: in INTEGER`
`constantright_index: in INTEGER`
`)`
- **unresolved_sfixed** `to_sfixed(`
`arg: in STD_ULOGIC_VECTOR`
`constantleft_index: in INTEGER`
`constantright_index: in INTEGER`
`)`
- **UNRESOLVED_sfixed** `"abs"(arg: in UNRESOLVED_sfixed)`
- **UNRESOLVED_sfixed** `"-"(arg: in UNRESOLVED_sfixed)`
- **UNRESOLVED_ufixed** `"+"(l: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)`
- **UNRESOLVED_sfixed** `"+"(l: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)`
- **UNRESOLVED_ufixed** `"-"(l: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)`
- **UNRESOLVED_sfixed** `"-"(l: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)`
- **UNRESOLVED_ufixed** `"*(l: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)`
- **UNRESOLVED_sfixed** `"*(l: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)`
- **UNRESOLVED_ufixed** `"/"(l: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)`
- **UNRESOLVED_sfixed** `"/"(l: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)`
- **UNRESOLVED_ufixed** `divide(`
`l: in UNRESOLVED_ufixed`
`r: in UNRESOLVED_ufixed`
`constantround_style: in fixed_round_style_type fixed_round_style`
`constantguard_bits: in NATURAL fixed_guard_bits`
`)`

- **UNRESOLVED_sfixed divide(**
 I: in UNRESOLVED_sfixed
 r: in UNRESOLVED_sfixed
constantround_style: in fixed_round_style_type fixed_round_style
constantguard_bits: in NATURAL fixed_guard_bits
)
- **UNRESOLVED_ufixed reciprocal(**
 arg: in UNRESOLVED_ufixed
constantround_style: in fixed_round_style_type fixed_round_style
constantguard_bits: in NATURAL fixed_guard_bits
)
- **UNRESOLVED_sfixed reciprocal(**
 arg: in UNRESOLVED_sfixed
constantround_style: in fixed_round_style_type fixed_round_style
constantguard_bits: in NATURAL fixed_guard_bits
)
- **UNRESOLVED_ufixed "rem"(I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)**
- **UNRESOLVED_sfixed "rem"(I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)**
- **UNRESOLVED_ufixed remainder(**
 I: in UNRESOLVED_ufixed
 r: in UNRESOLVED_ufixed
constantround_style: in fixed_round_style_type fixed_round_style
constantguard_bits: in NATURAL fixed_guard_bits
)
- **UNRESOLVED_sfixed remainder(**
 I: in UNRESOLVED_sfixed
 r: in UNRESOLVED_sfixed
constantround_style: in fixed_round_style_type fixed_round_style
constantguard_bits: in NATURAL fixed_guard_bits
)
- **UNRESOLVED_ufixed "mod"(I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)**
- **UNRESOLVED_sfixed "mod"(I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)**
- **UNRESOLVED_ufixed modulo(**
 I: in UNRESOLVED_ufixed
 r: in UNRESOLVED_ufixed
constantround_style: in fixed_round_style_type fixed_round_style
constantguard_bits: in NATURAL fixed_guard_bits
)
- **UNRESOLVED_sfixed modulo(**
 I: in UNRESOLVED_sfixed
 r: in UNRESOLVED_sfixed
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
constantguard_bits: in NATURAL fixed_guard_bits
)
- **UNRESOLVED_ufixed scalb(y: in UNRESOLVED_ufixed, N: in INTEGER)**
- **UNRESOLVED_ufixed scalb(y: in UNRESOLVED_ufixed, N: in SIGNED)**
- **UNRESOLVED_sfixed scalb(y: in UNRESOLVED_sfixed, N: in INTEGER)**
- **UNRESOLVED_sfixed scalb(y: in UNRESOLVED_sfixed, N: in SIGNED)**
- **BOOLEAN Is_Negative(arg: in UNRESOLVED_sfixed)**
- **INTEGER find_rightmost(arg: in UNRESOLVED_ufixed, y: in STD_ULOGIC)**
- **INTEGER find_leftmost(arg: in UNRESOLVED_ufixed, y: in STD_ULOGIC)**
- **INTEGER find_rightmost(arg: in UNRESOLVED_sfixed, y: in STD_ULOGIC)**
- **INTEGER find_leftmost(arg: in UNRESOLVED_sfixed, y: in STD_ULOGIC)**
- **UNRESOLVED_ufixed "sll"(ARG: in UNRESOLVED_ufixed, COUNT: in INTEGER)**
- **UNRESOLVED_ufixed "srll"(ARG: in UNRESOLVED_ufixed, COUNT: in INTEGER)**
- **UNRESOLVED_ufixed "rol"(ARG: in UNRESOLVED_ufixed, COUNT: in INTEGER)**

- UNRESOLVED_ufixed "ror"(ARG: in UNRESOLVED_ufixed, COUNT: in INTEGER)
- UNRESOLVED_ufixed "sla"(ARG: in UNRESOLVED_ufixed, COUNT: in INTEGER)
- UNRESOLVED_ufixed "sra"(ARG: in UNRESOLVED_ufixed, COUNT: in INTEGER)
- UNRESOLVED_sfixed "sll"(ARG: in UNRESOLVED_sfixed, COUNT: in INTEGER)
- UNRESOLVED_sfixed "srl"(ARG: in UNRESOLVED_sfixed, COUNT: in INTEGER)
- UNRESOLVED_sfixed "rol"(ARG: in UNRESOLVED_sfixed, COUNT: in INTEGER)
- UNRESOLVED_sfixed "ror"(ARG: in UNRESOLVED_sfixed, COUNT: in INTEGER)
- UNRESOLVED_sfixed "sla"(ARG: in UNRESOLVED_sfixed, COUNT: in INTEGER)
- UNRESOLVED_sfixed "sra"(ARG: in UNRESOLVED_sfixed, COUNT: in INTEGER)
- UNRESOLVED_ufixed SHIFT_LEFT(ARG: in UNRESOLVED_ufixed, COUNT: in NATURAL)
- UNRESOLVED_ufixed SHIFT_RIGHT(ARG: in UNRESOLVED_ufixed, COUNT: in NATURAL)
- UNRESOLVED_sfixed SHIFT_LEFT(ARG: in UNRESOLVED_sfixed, COUNT: in NATURAL)
- UNRESOLVED_sfixed SHIFT_RIGHT(ARG: in UNRESOLVED_sfixed, COUNT: in NATURAL)
- UNRESOLVED_ufixed "not"(L: in UNRESOLVED_ufixed)
- UNRESOLVED_ufixed "and"(L: in UNRESOLVED_ufixed, R: in UNRESOLVED_ufixed)
- UNRESOLVED_ufixed "or"(L: in UNRESOLVED_ufixed, R: in UNRESOLVED_ufixed)
- UNRESOLVED_ufixed "hand"(L: in UNRESOLVED_ufixed, R: in UNRESOLVED_ufixed)
- UNRESOLVED_ufixed "nor"(L: in UNRESOLVED_ufixed, R: in UNRESOLVED_ufixed)
- UNRESOLVED_ufixed "xor"(L: in UNRESOLVED_ufixed, R: in UNRESOLVED_ufixed)
- UNRESOLVED_ufixed "xnor"(L: in UNRESOLVED_ufixed, R: in UNRESOLVED_ufixed)
- UNRESOLVED_sfixed "not"(L: in UNRESOLVED_sfixed)
- UNRESOLVED_sfixed "and"(L: in UNRESOLVED_sfixed, R: in UNRESOLVED_sfixed)
- UNRESOLVED_sfixed "or"(L: in UNRESOLVED_sfixed, R: in UNRESOLVED_sfixed)
- UNRESOLVED_sfixed "hand"(L: in UNRESOLVED_sfixed, R: in UNRESOLVED_sfixed)
- UNRESOLVED_sfixed "nor"(L: in UNRESOLVED_sfixed, R: in UNRESOLVED_sfixed)
- UNRESOLVED_sfixed "xor"(L: in UNRESOLVED_sfixed, R: in UNRESOLVED_sfixed)
- UNRESOLVED_sfixed "xnor"(L: in UNRESOLVED_sfixed, R: in UNRESOLVED_sfixed)
- UNRESOLVED_ufixed "and"(L: in STD_ULOGIC, R: in UNRESOLVED_ufixed)
- UNRESOLVED_ufixed "and"(L: in UNRESOLVED_ufixed, R: in STD_ULOGIC)
- UNRESOLVED_ufixed "or"(L: in STD_ULOGIC, R: in UNRESOLVED_ufixed)
- UNRESOLVED_ufixed "or"(L: in UNRESOLVED_ufixed, R: in STD_ULOGIC)
- UNRESOLVED_ufixed "hand"(L: in STD_ULOGIC, R: in UNRESOLVED_ufixed)
- UNRESOLVED_ufixed "hand"(L: in UNRESOLVED_ufixed, R: in STD_ULOGIC)
- UNRESOLVED_ufixed "nor"(L: in STD_ULOGIC, R: in UNRESOLVED_ufixed)
- UNRESOLVED_ufixed "nor"(L: in UNRESOLVED_ufixed, R: in STD_ULOGIC)
- UNRESOLVED_ufixed "xor"(L: in STD_ULOGIC, R: in UNRESOLVED_ufixed)
- UNRESOLVED_ufixed "xor"(L: in UNRESOLVED_ufixed, R: in STD_ULOGIC)
- UNRESOLVED_ufixed "xnor"(L: in STD_ULOGIC, R: in UNRESOLVED_ufixed)
- UNRESOLVED_ufixed "xnor"(L: in UNRESOLVED_ufixed, R: in STD_ULOGIC)
- UNRESOLVED_sfixed "and"(L: in STD_ULOGIC, R: in UNRESOLVED_sfixed)
- UNRESOLVED_sfixed "and"(L: in UNRESOLVED_sfixed, R: in STD_ULOGIC)
- UNRESOLVED_sfixed "or"(L: in STD_ULOGIC, R: in UNRESOLVED_sfixed)
- UNRESOLVED_sfixed "or"(L: in UNRESOLVED_sfixed, R: in STD_ULOGIC)
- UNRESOLVED_sfixed "hand"(L: in STD_ULOGIC, R: in UNRESOLVED_sfixed)
- UNRESOLVED_sfixed "hand"(L: in UNRESOLVED_sfixed, R: in STD_ULOGIC)
- UNRESOLVED_sfixed "nor"(L: in STD_ULOGIC, R: in UNRESOLVED_sfixed)
- UNRESOLVED_sfixed "nor"(L: in UNRESOLVED_sfixed, R: in STD_ULOGIC)
- UNRESOLVED_sfixed "xor"(L: in STD_ULOGIC, R: in UNRESOLVED_sfixed)
- UNRESOLVED_sfixed "xor"(L: in UNRESOLVED_sfixed, R: in STD_ULOGIC)
- UNRESOLVED_sfixed "xnor"(L: in STD_ULOGIC, R: in UNRESOLVED_sfixed)
- UNRESOLVED_sfixed "xnor"(L: in UNRESOLVED_sfixed, R: in STD_ULOGIC)
- STD_ULOGIC and_reduce(I: in UNRESOLVED_ufixed)
- STD_ULOGIC hand_reduce(I: in UNRESOLVED_ufixed)
- STD_ULOGIC or_reduce(I: in UNRESOLVED_ufixed)
- STD_ULOGIC nor_reduce(I: in UNRESOLVED_ufixed)

- **STD_ULOGIC xor_reduce(I: in UNRESOLVED_ufixed)**
- **STD_ULOGIC xnor_reduce(I: in UNRESOLVED_ufixed)**
- **STD_ULOGIC and_reduce(I: in UNRESOLVED_sfixed)**
- **STD_ULOGIC nand_reduce(I: in UNRESOLVED_sfixed)**
- **STD_ULOGIC or_reduce(I: in UNRESOLVED_sfixed)**
- **STD_ULOGIC nor_reduce(I: in UNRESOLVED_sfixed)**
- **STD_ULOGIC xor_reduce(I: in UNRESOLVED_sfixed)**
- **STD_ULOGIC xnor_reduce(I: in UNRESOLVED_sfixed)**
- **STD_ULOGIC \?=\\(L: in UNRESOLVED_ufixed, R: in UNRESOLVED_ufixed)**
- **STD_ULOGIC \?/=\\(L: in UNRESOLVED_ufixed, R: in UNRESOLVED_ufixed)**
- **STD_ULOGIC \?>\\(L: in UNRESOLVED_ufixed, R: in UNRESOLVED_ufixed)**
- **STD_ULOGIC \?>=\\(L: in UNRESOLVED_ufixed, R: in UNRESOLVED_ufixed)**
- **STD_ULOGIC \?<\\(L: in UNRESOLVED_ufixed, R: in UNRESOLVED_ufixed)**
- **STD_ULOGIC \?<=\\(L: in UNRESOLVED_ufixed, R: in UNRESOLVED_ufixed)**
- **STD_ULOGIC \?=\\(L: in UNRESOLVED_sfixed, R: in UNRESOLVED_sfixed)**
- **STD_ULOGIC \?/=\\(L: in UNRESOLVED_sfixed, R: in UNRESOLVED_sfixed)**
- **STD_ULOGIC \?>\\(L: in UNRESOLVED_sfixed, R: in UNRESOLVED_sfixed)**
- **STD_ULOGIC \?>=\\(L: in UNRESOLVED_sfixed, R: in UNRESOLVED_sfixed)**
- **STD_ULOGIC \?<\\(L: in UNRESOLVED_sfixed, R: in UNRESOLVED_sfixed)**
- **STD_ULOGIC \?<=\\(L: in UNRESOLVED_sfixed, R: in UNRESOLVED_sfixed)**
- **BOOLEAN std_match(L: in UNRESOLVED_ufixed, R: in UNRESOLVED_ufixed)**
- **BOOLEAN std_match(L: in UNRESOLVED_sfixed, R: in UNRESOLVED_sfixed)**
- **BOOLEAN "=(I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)**
- **BOOLEAN "=(I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)**
- **BOOLEAN "/=(I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)**
- **BOOLEAN "/=(I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)**
- **BOOLEAN ">\\(I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)**
- **BOOLEAN ">\\(I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)**
- **BOOLEAN "<\\(I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)**
- **BOOLEAN "<\\(I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)**
- **BOOLEAN ">=\\(I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)**
- **BOOLEAN ">=\\(I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)**
- **BOOLEAN "<=\\(I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)**
- **BOOLEAN "<=\\(I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)**
- **UNRESOLVED_ufixed maximum(I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)**
- **UNRESOLVED_sfixed maximum(I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)**
- **UNRESOLVED_ufixed minimum(I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)**
- **UNRESOLVED_sfixed minimum(I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)**
- **UNRESOLVED_ufixed to_ufixed(**
arg: in **NATURAL**
constantleft_index: in INTEGER
constantright_index: in INTEGER 0
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
)
- **UNRESOLVED_sfixed to_sfixed(**
arg: in **INTEGER**
constantleft_index: in INTEGER
constantright_index: in INTEGER 0
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
)

- UNRESOLVED_ufixed `to_ufixed(`
`arg: in REAL`
`constantleft_index: in INTEGER`
`constantright_index: in INTEGER`
`constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style`
`constantround_style: in fixed_round_style_type fixed_round_style`
`constantguard_bits: in NATURAL fixed_guard_bits`
`)`
- UNRESOLVED_sfixed `to_sfixed(`
`arg: in REAL`
`constantleft_index: in INTEGER`
`constantright_index: in INTEGER`
`constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style`
`constantround_style: in fixed_round_style_type fixed_round_style`
`constantguard_bits: in NATURAL fixed_guard_bits`
`)`
- UNRESOLVED_ufixed `to_ufixed(`
`arg: in UNSIGNED`
`constantleft_index: in INTEGER`
`constantright_index: in INTEGER 0`
`constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style`
`constantround_style: in fixed_round_style_type fixed_round_style`
`)`
- UNRESOLVED_ufixed `to_ufixed(arg: in UNSIGNED)`
- UNRESOLVED_sfixed `to_sfixed(`
`arg: in SIGNED`
`constantleft_index: in INTEGER`
`constantright_index: in INTEGER 0`
`constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style`
`constantround_style: in fixed_round_style_type fixed_round_style`
`)`
- UNRESOLVED_sfixed `to_sfixed(arg: in SIGNED)`
- UNRESOLVED_sfixed `to_sfixed(arg: in UNRESOLVED_ufixed)`
- **INTEGER** `ufixed_high(`
`left_index: in INTEGER`
`right_index: in INTEGER`
`operation: in CHARACTER 'X'`
`left_index2: in INTEGER 0`
`right_index2: in INTEGER 0`
`)`
- **INTEGER** `ufixed_low(`
`left_index: in INTEGER`
`right_index: in INTEGER`
`operation: in CHARACTER 'X'`
`left_index2: in INTEGER 0`
`right_index2: in INTEGER 0`
`)`
- **INTEGER** `sfixed_high(`
`left_index: in INTEGER`
`right_index: in INTEGER`
`operation: in CHARACTER 'X'`
`left_index2: in INTEGER 0`
`right_index2: in INTEGER 0`
`)`

- **INTEGER** sfixed_low(
left_index: in **INTEGER**
right_index: in **INTEGER**
operation: in CHARACTER 'X'
left_index2: in **INTEGER** 0
right_index2: in **INTEGER** 0
)
- **INTEGER** ufixed_high(
size_res: in UNRESOLVED_ufixed
operation: in CHARACTER 'X'
size_res2: in UNRESOLVED_ufixed
)
- **INTEGER** ufixed_low(
size_res: in UNRESOLVED_ufixed
operation: in CHARACTER 'X'
size_res2: in UNRESOLVED_ufixed
)
- **INTEGER** sfixed_high(
size_res: in UNRESOLVED_sfixed
operation: in CHARACTER 'X'
size_res2: in UNRESOLVED_sfixed
)
- **INTEGER** sfixed_low(
size_res: in UNRESOLVED_sfixed
operation: in CHARACTER 'X'
size_res2: in UNRESOLVED_sfixed
)
- UNRESOLVED_ufixed saturate(constantleft_index: in **INTEGER**, constantright_index: in **INTEGER**)
- UNRESOLVED_sfixed saturate(constantleft_index: in **INTEGER**, constantright_index: in **INTEGER**)
- UNRESOLVED_ufixed saturate(size_res: in UNRESOLVED_ufixed)
- UNRESOLVED_sfixed saturate(size_res: in UNRESOLVED_sfixed)
- UNRESOLVED_ufixed to_UFix(
arg: in **STD_ULOGIC_VECTOR**
width: in **NATURAL**
fraction: in **NATURAL**
)
- UNRESOLVED_sfixed to_SFix(
arg: in **STD_ULOGIC_VECTOR**
width: in **NATURAL**
fraction: in **NATURAL**
)
- **INTEGER** ufix_high(
width: in **NATURAL**
fraction: in **NATURAL**
operation: in CHARACTER 'X'
width2: in **NATURAL** 0
fraction2: in **NATURAL** 0
)
- **INTEGER** ufix_low(
width: in **NATURAL**
fraction: in **NATURAL**
operation: in CHARACTER 'X'
width2: in **NATURAL** 0
fraction2: in **NATURAL** 0
)

- **INTEGER** `sfix_high`(
width: in **NATURAL**
fraction: in **NATURAL**
operation: in CHARACTER 'X'
width2: in NATURAL 0
fraction2: in NATURAL 0
)
- **INTEGER** `sfix_low`(
width: in **NATURAL**
fraction: in **NATURAL**
operation: in CHARACTER 'X'
width2: in NATURAL 0
fraction2: in NATURAL 0
)
- **UNSIGNED** `to_unsigned`(
arg: in UNRESOLVED_ufixed
constantsize: in **NATURAL**
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
)
- **UNSIGNED** `to_unsigned`(
arg: in UNRESOLVED_ufixed
size_res: in **UNSIGNED**
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
)
- **SIGNED** `to_signed`(
arg: in UNRESOLVED_sfixed
constantsize: in **NATURAL**
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
)
- **SIGNED** `to_signed`(
arg: in UNRESOLVED_sfixed
size_res: in **SIGNED**
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
)
- **REAL** `to_real`(arg: in UNRESOLVED_ufixed)
- **REAL** `to_real`(arg: in UNRESOLVED_sfixed)
- **NATURAL** `to_integer`(
arg: in UNRESOLVED_ufixed
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
)
- **INTEGER** `to_integer`(
arg: in UNRESOLVED_sfixed
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
)
- **UNRESOLVED_ufixed** `to_01`(s: in UNRESOLVED_ufixed, **constantXMAP**: in STD_ULOGIC '0')
- **UNRESOLVED_sfixed** `to_01`(s: in UNRESOLVED_sfixed, **constantXMAP**: in STD_ULOGIC '0')
- **BOOLEAN** `Is_X`(arg: in UNRESOLVED_ufixed)
- **BOOLEAN** `Is_X`(arg: in UNRESOLVED_sfixed)
- **UNRESOLVED_ufixed** `To_X01`(arg: in UNRESOLVED_ufixed)
- **UNRESOLVED_sfixed** `To_X01`(arg: in UNRESOLVED_sfixed)
- **UNRESOLVED_ufixed** `To_X01Z`(arg: in UNRESOLVED_ufixed)

- UNRESOLVED_sfixed `to_X01Z(arg: in UNRESOLVED_sfixed)`
- UNRESOLVED_ufixed `To_UX01(arg: in UNRESOLVED_ufixed)`
- UNRESOLVED_sfixed `to_UX01(arg: in UNRESOLVED_sfixed)`
- UNRESOLVED_ufixed `resize(
arg: in UNRESOLVED_ufixed
constantleft_index: in INTEGER
constantright_index: in INTEGER
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
)`
- UNRESOLVED_sfixed `resize(
arg: in UNRESOLVED_sfixed
constantleft_index: in INTEGER
constantright_index: in INTEGER
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
)`
- UNRESOLVED_ufixed `to_ufixed(arg: in STD_ULOGIC_VECTOR, size_res: in UNRESOLVED_ufixed)`
- UNRESOLVED_sfixed `to_sfixed(arg: in STD_ULOGIC_VECTOR, size_res: in UNRESOLVED_sfixed)`
- UNRESOLVED_ufixed `to_ufixed(
arg: in NATURAL
size_res: in UNRESOLVED_ufixed
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
)`
- UNRESOLVED_sfixed `to_sfixed(
arg: in INTEGER
size_res: in UNRESOLVED_sfixed
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
)`
- UNRESOLVED_ufixed `to_ufixed(
arg: in REAL
size_res: in UNRESOLVED_ufixed
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
constantguard_bits: in NATURAL fixed_guard_bits
)`
- UNRESOLVED_sfixed `to_sfixed(
arg: in REAL
size_res: in UNRESOLVED_sfixed
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
constantguard_bits: in NATURAL fixed_guard_bits
)`
- UNRESOLVED_ufixed `to_ufixed(
arg: in UNSIGNED
size_res: in UNRESOLVED_ufixed
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
)`
- UNRESOLVED_sfixed `to_sfixed(
arg: in SIGNED
size_res: in UNRESOLVED_sfixed
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
)`

- **UNRESOLVED_ufixed** `resize(`
`arg: in UNRESOLVED_ufixed`
`size_res: in UNRESOLVED_ufixed`
`constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style`
`constantround_style: in fixed_round_style_type fixed_round_style`
`)`
- **UNRESOLVED_sfixed** `resize(`
`arg: in UNRESOLVED_sfixed`
`size_res: in UNRESOLVED_sfixed`
`constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style`
`constantround_style: in fixed_round_style_type fixed_round_style`
`)`
- **UNRESOLVED_ufixed** `"+"(I: in UNRESOLVED_ufixed, r: in REAL)`
- **UNRESOLVED_ufixed** `"+"(I: in REAL, r: in UNRESOLVED_ufixed)`
- **UNRESOLVED_sfixed** `"+"(I: in UNRESOLVED_sfixed, r: in REAL)`
- **UNRESOLVED_sfixed** `"+"(I: in REAL, r: in UNRESOLVED_sfixed)`
- **UNRESOLVED_ufixed** `"-(I: in UNRESOLVED_ufixed, r: in REAL)`
- **UNRESOLVED_ufixed** `"-(I: in REAL, r: in UNRESOLVED_ufixed)`
- **UNRESOLVED_sfixed** `"-(I: in UNRESOLVED_sfixed, r: in REAL)`
- **UNRESOLVED_sfixed** `"-(I: in REAL, r: in UNRESOLVED_sfixed)`
- **UNRESOLVED_ufixed** `"*(I: in UNRESOLVED_ufixed, r: in REAL)`
- **UNRESOLVED_ufixed** `"*(I: in REAL, r: in UNRESOLVED_ufixed)`
- **UNRESOLVED_sfixed** `"*(I: in UNRESOLVED_sfixed, r: in REAL)`
- **UNRESOLVED_sfixed** `"*(I: in REAL, r: in UNRESOLVED_sfixed)`
- **UNRESOLVED_ufixed** `"/"(I: in UNRESOLVED_ufixed, r: in REAL)`
- **UNRESOLVED_ufixed** `"/"(I: in REAL, r: in UNRESOLVED_ufixed)`
- **UNRESOLVED_sfixed** `"/"(I: in UNRESOLVED_sfixed, r: in REAL)`
- **UNRESOLVED_sfixed** `"/"(I: in REAL, r: in UNRESOLVED_sfixed)`
- **UNRESOLVED_ufixed** `"rem"(I: in UNRESOLVED_ufixed, r: in REAL)`
- **UNRESOLVED_ufixed** `"rem"(I: in REAL, r: in UNRESOLVED_ufixed)`
- **UNRESOLVED_sfixed** `"rem"(I: in UNRESOLVED_sfixed, r: in REAL)`
- **UNRESOLVED_sfixed** `"rem"(I: in REAL, r: in UNRESOLVED_sfixed)`
- **UNRESOLVED_ufixed** `"mod"(I: in UNRESOLVED_ufixed, r: in REAL)`
- **UNRESOLVED_ufixed** `"mod"(I: in REAL, r: in UNRESOLVED_ufixed)`
- **UNRESOLVED_sfixed** `"mod"(I: in UNRESOLVED_sfixed, r: in REAL)`
- **UNRESOLVED_sfixed** `"mod"(I: in REAL, r: in UNRESOLVED_sfixed)`
- **UNRESOLVED_ufixed** `"+"(I: in UNRESOLVED_ufixed, r: in NATURAL)`
- **UNRESOLVED_ufixed** `"+"(I: in NATURAL, r: in UNRESOLVED_ufixed)`
- **UNRESOLVED_sfixed** `"+"(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- **UNRESOLVED_sfixed** `"+"(I: in INTEGER, r: in UNRESOLVED_sfixed)`
- **UNRESOLVED_ufixed** `"-(I: in UNRESOLVED_ufixed, r: in NATURAL)`
- **UNRESOLVED_ufixed** `"-(I: in NATURAL, r: in UNRESOLVED_ufixed)`
- **UNRESOLVED_sfixed** `"-(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- **UNRESOLVED_sfixed** `"-(I: in INTEGER, r: in UNRESOLVED_sfixed)`
- **UNRESOLVED_ufixed** `"*(I: in UNRESOLVED_ufixed, r: in NATURAL)`
- **UNRESOLVED_ufixed** `"*(I: in NATURAL, r: in UNRESOLVED_ufixed)`
- **UNRESOLVED_sfixed** `"*(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- **UNRESOLVED_sfixed** `"*(I: in INTEGER, r: in UNRESOLVED_sfixed)`
- **UNRESOLVED_ufixed** `"/"(I: in UNRESOLVED_ufixed, r: in NATURAL)`
- **UNRESOLVED_ufixed** `"/"(I: in NATURAL, r: in UNRESOLVED_ufixed)`
- **UNRESOLVED_sfixed** `"/"(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- **UNRESOLVED_sfixed** `"/"(I: in INTEGER, r: in UNRESOLVED_sfixed)`
- **UNRESOLVED_ufixed** `"rem"(I: in UNRESOLVED_ufixed, r: in NATURAL)`
- **UNRESOLVED_ufixed** `"rem"(I: in NATURAL, r: in UNRESOLVED_ufixed)`
- **UNRESOLVED_sfixed** `"rem"(I: in UNRESOLVED_sfixed, r: in INTEGER)`

- UNRESOLVED_sfixed "rem"(I: in INTEGER, r: in UNRESOLVED_sfixed)
- UNRESOLVED_ufixed "mod"(I: in UNRESOLVED_ufixed, r: in NATURAL)
- UNRESOLVED_ufixed "mod"(I: in NATURAL, r: in UNRESOLVED_ufixed)
- UNRESOLVED_sfixed "mod"(I: in UNRESOLVED_sfixed, r: in INTEGER)
- UNRESOLVED_sfixed "mod"(I: in INTEGER, r: in UNRESOLVED_sfixed)
- BOOLEAN "="(I: in UNRESOLVED_ufixed, r: in NATURAL)
- BOOLEAN "/="(I: in UNRESOLVED_ufixed, r: in NATURAL)
- BOOLEAN ">="(I: in UNRESOLVED_ufixed, r: in NATURAL)
- BOOLEAN "<="(I: in UNRESOLVED_ufixed, r: in NATURAL)
- BOOLEAN ">"(I: in UNRESOLVED_ufixed, r: in NATURAL)
- BOOLEAN "<"(I: in UNRESOLVED_ufixed, r: in NATURAL)
- STD_ULOGIC ?=(I: in UNRESOLVED_ufixed, r: in NATURAL)
- STD_ULOGIC ?/=(I: in UNRESOLVED_ufixed, r: in NATURAL)
- STD_ULOGIC ?>=(I: in UNRESOLVED_ufixed, r: in NATURAL)
- STD_ULOGIC ?<=(I: in UNRESOLVED_ufixed, r: in NATURAL)
- STD_ULOGIC ?> (I: in UNRESOLVED_ufixed, r: in NATURAL)
- STD_ULOGIC ?< (I: in UNRESOLVED_ufixed, r: in NATURAL)
- UNRESOLVED_ufixed maximum(I: in UNRESOLVED_ufixed, r: in NATURAL)
- UNRESOLVED_ufixed minimum(I: in UNRESOLVED_ufixed, r: in NATURAL)
- BOOLEAN "="(I: in NATURAL, r: in UNRESOLVED_ufixed)
- BOOLEAN "/="(I: in NATURAL, r: in UNRESOLVED_ufixed)
- BOOLEAN ">="(I: in NATURAL, r: in UNRESOLVED_ufixed)
- BOOLEAN "<="(I: in NATURAL, r: in UNRESOLVED_ufixed)
- BOOLEAN ">"(I: in NATURAL, r: in UNRESOLVED_ufixed)
- BOOLEAN "<"(I: in NATURAL, r: in UNRESOLVED_ufixed)
- STD_ULOGIC ?=(I: in NATURAL, r: in UNRESOLVED_ufixed)
- STD_ULOGIC ?/=(I: in NATURAL, r: in UNRESOLVED_ufixed)
- STD_ULOGIC ?>=(I: in NATURAL, r: in UNRESOLVED_ufixed)
- STD_ULOGIC ?<=(I: in NATURAL, r: in UNRESOLVED_ufixed)
- STD_ULOGIC ?> (I: in NATURAL, r: in UNRESOLVED_ufixed)
- STD_ULOGIC ?< (I: in NATURAL, r: in UNRESOLVED_ufixed)
- UNRESOLVED_ufixed maximum(I: in NATURAL, r: in UNRESOLVED_ufixed)
- UNRESOLVED_ufixed minimum(I: in NATURAL, r: in UNRESOLVED_ufixed)
- BOOLEAN "="(I: in UNRESOLVED_ufixed, r: in REAL)
- BOOLEAN "/="(I: in UNRESOLVED_ufixed, r: in REAL)
- BOOLEAN ">="(I: in UNRESOLVED_ufixed, r: in REAL)
- BOOLEAN "<="(I: in UNRESOLVED_ufixed, r: in REAL)
- BOOLEAN ">"(I: in UNRESOLVED_ufixed, r: in REAL)
- BOOLEAN "<"(I: in UNRESOLVED_ufixed, r: in REAL)
- STD_ULOGIC ?=(I: in UNRESOLVED_ufixed, r: in REAL)
- STD_ULOGIC ?/=(I: in UNRESOLVED_ufixed, r: in REAL)
- STD_ULOGIC ?>=(I: in UNRESOLVED_ufixed, r: in REAL)
- STD_ULOGIC ?<=(I: in UNRESOLVED_ufixed, r: in REAL)
- STD_ULOGIC ?> (I: in UNRESOLVED_ufixed, r: in REAL)
- STD_ULOGIC ?< (I: in UNRESOLVED_ufixed, r: in REAL)
- UNRESOLVED_ufixed maximum(I: in UNRESOLVED_ufixed, r: in REAL)
- UNRESOLVED_ufixed minimum(I: in UNRESOLVED_ufixed, r: in REAL)
- BOOLEAN "="(I: in REAL, r: in UNRESOLVED_ufixed)
- BOOLEAN "/="(I: in REAL, r: in UNRESOLVED_ufixed)
- BOOLEAN ">="(I: in REAL, r: in UNRESOLVED_ufixed)
- BOOLEAN "<="(I: in REAL, r: in UNRESOLVED_ufixed)
- BOOLEAN ">"(I: in REAL, r: in UNRESOLVED_ufixed)
- BOOLEAN "<"(I: in REAL, r: in UNRESOLVED_ufixed)
- STD_ULOGIC ?=(I: in REAL, r: in UNRESOLVED_ufixed)
- STD_ULOGIC ?/=(I: in REAL, r: in UNRESOLVED_ufixed)

- `STD_ULOGIC \?>=(I: in REAL, r: in UNRESOLVED_ufixed)`
- `STD_ULOGIC \?<=(I: in REAL, r: in UNRESOLVED_ufixed)`
- `STD_ULOGIC \?>(I: in REAL, r: in UNRESOLVED_ufixed)`
- `STD_ULOGIC \?<(I: in REAL, r: in UNRESOLVED_ufixed)`
- `UNRESOLVED_ufixed maximum(I: in REAL, r: in UNRESOLVED_ufixed)`
- `UNRESOLVED_ufixed minimum(I: in REAL, r: in UNRESOLVED_ufixed)`
- `BOOLEAN "="(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- `BOOLEAN "/="(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- `BOOLEAN ">="(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- `BOOLEAN "<="(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- `BOOLEAN ">"(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- `BOOLEAN "<"(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- `STD_ULOGIC \?<=(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- `STD_ULOGIC \?/=(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- `STD_ULOGIC \?>=(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- `STD_ULOGIC \?<=(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- `STD_ULOGIC \?>(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- `STD_ULOGIC \?<(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- `UNRESOLVED_sfixed maximum(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- `UNRESOLVED_sfixed minimum(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- `BOOLEAN "="(I: in INTEGER, r: in UNRESOLVED_sfixed)`
- `BOOLEAN "/="(I: in INTEGER, r: in UNRESOLVED_sfixed)`
- `BOOLEAN ">="(I: in INTEGER, r: in UNRESOLVED_sfixed)`
- `BOOLEAN "<="(I: in INTEGER, r: in UNRESOLVED_sfixed)`
- `BOOLEAN ">"(I: in INTEGER, r: in UNRESOLVED_sfixed)`
- `BOOLEAN "<"(I: in INTEGER, r: in UNRESOLVED_sfixed)`
- `STD_ULOGIC \?<=(I: in INTEGER, r: in UNRESOLVED_sfixed)`
- `STD_ULOGIC \?/=(I: in INTEGER, r: in UNRESOLVED_sfixed)`
- `STD_ULOGIC \?>=(I: in INTEGER, r: in UNRESOLVED_sfixed)`
- `STD_ULOGIC \?<=(I: in INTEGER, r: in UNRESOLVED_sfixed)`
- `STD_ULOGIC \?>(I: in INTEGER, r: in UNRESOLVED_sfixed)`
- `STD_ULOGIC \?<(I: in INTEGER, r: in UNRESOLVED_sfixed)`
- `UNRESOLVED_sfixed maximum(I: in INTEGER, r: in UNRESOLVED_sfixed)`
- `UNRESOLVED_sfixed minimum(I: in INTEGER, r: in UNRESOLVED_sfixed)`
- `BOOLEAN "="(I: in UNRESOLVED_sfixed, r: in REAL)`
- `BOOLEAN "/="(I: in UNRESOLVED_sfixed, r: in REAL)`
- `BOOLEAN ">="(I: in UNRESOLVED_sfixed, r: in REAL)`
- `BOOLEAN "<="(I: in UNRESOLVED_sfixed, r: in REAL)`
- `BOOLEAN ">"(I: in UNRESOLVED_sfixed, r: in REAL)`
- `BOOLEAN "<"(I: in UNRESOLVED_sfixed, r: in REAL)`
- `STD_ULOGIC \?<=(I: in UNRESOLVED_sfixed, r: in REAL)`
- `STD_ULOGIC \?/=(I: in UNRESOLVED_sfixed, r: in REAL)`
- `STD_ULOGIC \?>=(I: in UNRESOLVED_sfixed, r: in REAL)`
- `STD_ULOGIC \?<=(I: in UNRESOLVED_sfixed, r: in REAL)`
- `STD_ULOGIC \?>(I: in UNRESOLVED_sfixed, r: in REAL)`
- `STD_ULOGIC \?<(I: in UNRESOLVED_sfixed, r: in REAL)`
- `UNRESOLVED_sfixed maximum(I: in UNRESOLVED_sfixed, r: in REAL)`
- `UNRESOLVED_sfixed minimum(I: in UNRESOLVED_sfixed, r: in REAL)`
- `BOOLEAN "="(I: in REAL, r: in UNRESOLVED_sfixed)`
- `BOOLEAN "/="(I: in REAL, r: in UNRESOLVED_sfixed)`
- `BOOLEAN ">="(I: in REAL, r: in UNRESOLVED_sfixed)`
- `BOOLEAN "<="(I: in REAL, r: in UNRESOLVED_sfixed)`
- `BOOLEAN ">"(I: in REAL, r: in UNRESOLVED_sfixed)`
- `BOOLEAN "<"(I: in REAL, r: in UNRESOLVED_sfixed)`
- `STD_ULOGIC \?<=(I: in REAL, r: in UNRESOLVED_sfixed)`

- **STD_ULOGIC** `\?=/`(*I*: in **REAL**, *r*: in **UNRESOLVED_sfixed**)
- **STD_ULOGIC** `\?>=`(*I*: in **REAL**, *r*: in **UNRESOLVED_sfixed**)
- **STD_ULOGIC** `\?<=`(*I*: in **REAL**, *r*: in **UNRESOLVED_sfixed**)
- **STD_ULOGIC** `\?>`(*I*: in **REAL**, *r*: in **UNRESOLVED_sfixed**)
- **STD_ULOGIC** `\?<`(*I*: in **REAL**, *r*: in **UNRESOLVED_sfixed**)
- **UNRESOLVED_sfixed** **maximum**(*I*: in **REAL**, *r*: in **UNRESOLVED_sfixed**)
- **UNRESOLVED_sfixed** **minimum**(*I*: in **REAL**, *r*: in **UNRESOLVED_sfixed**)
- **STRING** **to_ostring**(*value*: in **STD_ULOGIC_VECTOR**)
- **STRING** **to_hstring**(*value*: in **STD_ULOGIC_VECTOR**)
- **STRING** **to_string**(*value*: in **UNRESOLVED_ufixed**)
- **STRING** **to_string**(*value*: in **UNRESOLVED_sfixed**)
- **STRING** **to_ostring**(*value*: in **UNRESOLVED_ufixed**)
- **STRING** **to_hstring**(*value*: in **UNRESOLVED_ufixed**)
- **STRING** **to_ostring**(*value*: in **UNRESOLVED_sfixed**)
- **STRING** **to_hstring**(*value*: in **UNRESOLVED_sfixed**)
- **UNRESOLVED_ufixed** **from_string**(
 bstring: in **STRING**
 constantleft_index: in **INTEGER**
 constantright_index: in **INTEGER**
)
- **UNRESOLVED_ufixed** **from_ostring**(
 ostring: in **STRING**
 constantleft_index: in **INTEGER**
 constantright_index: in **INTEGER**
)
- **UNRESOLVED_ufixed** **from_hstring**(
 hstring: in **STRING**
 constantleft_index: in **INTEGER**
 constantright_index: in **INTEGER**
)
- **UNRESOLVED_sfixed** **from_string**(
 bstring: in **STRING**
 constantleft_index: in **INTEGER**
 constantright_index: in **INTEGER**
)
- **UNRESOLVED_sfixed** **from_ostring**(
 ostring: in **STRING**
 constantleft_index: in **INTEGER**
 constantright_index: in **INTEGER**
)
- **UNRESOLVED_sfixed** **from_hstring**(
 hstring: in **STRING**
 constantleft_index: in **INTEGER**
 constantright_index: in **INTEGER**
)
- **UNRESOLVED_ufixed** **from_string**(*bstring*: in **STRING**, *size_res*: in **UNRESOLVED_ufixed**)
- **UNRESOLVED_ufixed** **from_ostring**(*ostring*: in **STRING**, *size_res*: in **UNRESOLVED_ufixed**)
- **UNRESOLVED_ufixed** **from_hstring**(*hstring*: in **STRING**, *size_res*: in **UNRESOLVED_ufixed**)
- **UNRESOLVED_sfixed** **from_string**(*bstring*: in **STRING**, *size_res*: in **UNRESOLVED_sfixed**)
- **UNRESOLVED_sfixed** **from_ostring**(*ostring*: in **STRING**, *size_res*: in **UNRESOLVED_sfixed**)
- **UNRESOLVED_sfixed** **from_hstring**(*hstring*: in **STRING**, *size_res*: in **UNRESOLVED_sfixed**)
- **UNRESOLVED_ufixed** **from_string**(*bstring*: in **STRING**)
- **UNRESOLVED_ufixed** **from_ostring**(*ostring*: in **STRING**)
- **UNRESOLVED_ufixed** **from_hstring**(*hstring*: in **STRING**)
- **UNRESOLVED_sfixed** **from_string**(*bstring*: in **STRING**)
- **UNRESOLVED_sfixed** **from_ostring**(*ostring*: in **STRING**)

- UNRESOLVED_sfixed from_hstring(hstring: in STRING)
- UNRESOLVED_ufixed to_ufixed(

arg: in STD_LOGIC_VECTOR

constantleft_index: in INTEGER

constantright_index: in INTEGER

)
- UNRESOLVED_ufixed to_ufixed(arg: in STD_LOGIC_VECTOR, size_res: in UNRESOLVED_ufixed)
- UNRESOLVED_sfixed to_sfixed(

arg: in STD_LOGIC_VECTOR

constantleft_index: in INTEGER

constantright_index: in INTEGER

)
- UNRESOLVED_sfixed to_sfixed(arg: in STD_LOGIC_VECTOR, size_res: in UNRESOLVED_sfixed)
- UNRESOLVED_ufixed to_UFix(

arg: in STD_LOGIC_VECTOR

width: in NATURAL

fraction: in NATURAL

)
- UNRESOLVED_sfixed to_SFix(

arg: in STD_LOGIC_VECTOR

width: in NATURAL

fraction: in NATURAL

)

Procedures

- round_up(

arg: inUNRESOLVED_ufixed

result: outUNRESOLVED_ufixed

overflowx: outBOOLEAN

)
- round_up(

arg: inUNRESOLVED_sfixed

result: outUNRESOLVED_sfixed

overflowx: outBOOLEAN

)
- add_carry(

L: inUNRESOLVED_ufixed

R: inUNRESOLVED_ufixed

c_in: inSTD_ULOGIC

result: outUNRESOLVED_ufixed

c_out: outSTD_ULOGIC

)
- add_carry(

L: inUNRESOLVED_sfixed

R: inUNRESOLVED_sfixed

c_in: inSTD_ULOGIC

result: outUNRESOLVED_sfixed

c_out: outSTD_ULOGIC

)
- Char2TriBits(

C: CHARACTER

RESULT: outSTD_ULOGIC_VECTOR(2 downto 0)

GOOD: outBOOLEAN

ISSUE_ERROR: inBOOLEAN

)

- Char2QuadBits(
C: CHARACTER
RESULT: outSTD_ULOGIC_VECTOR(3 downto 0)
GOOD: outBOOLEAN
ISSUE_ERROR: inBOOLEAN
)
- skip_whitespace(L: inoutLINE)
- write(
L: inoutLINE
VALUE: inUNRESOLVED_ufixed
JUSTIFIED: inSIDEright
FIELD: inWIDTH 0
)
- write(
L: inoutLINE
VALUE: inUNRESOLVED_sfixed
JUSTIFIED: inSIDEright
FIELD: inWIDTH 0
)
- READ(L: inoutLINE, VALUE: outUNRESOLVED_ufixed)
- READ(
L: inoutLINE
VALUE: outUNRESOLVED_ufixed
GOOD: outBOOLEAN
)
- READ(L: inoutLINE, VALUE: outUNRESOLVED_sfixed)
- READ(
L: inoutLINE
VALUE: outUNRESOLVED_sfixed
GOOD: outBOOLEAN
)
- owrite(
L: inoutLINE
VALUE: inUNRESOLVED_ufixed
JUSTIFIED: inSIDEright
FIELD: inWIDTH 0
)
- owrite(
L: inoutLINE
VALUE: inUNRESOLVED_sfixed
JUSTIFIED: inSIDEright
FIELD: inWIDTH 0
)
- OREAD_common(
L: inoutLINE
slv: outSTD_ULOGIC_VECTOR
igood: outBOOLEAN
index: outINTEGER
constant bpoint: inINTEGER
constant message: inBOOLEAN
constant smath: inBOOLEAN
)
- errmes(constant mess: inSTRING)
- OREAD(L: inoutLINE, VALUE: outUNRESOLVED_ufixed)
- OREAD(
L: inoutLINE
VALUE: outUNRESOLVED_ufixed
GOOD: outBOOLEAN

-)
- OREAD(L: **inoutLINE**, VALUE: **outUNRESOLVED_sfixed**)
- OREAD(
 - L: **inoutLINE**
 - VALUE: **outUNRESOLVED_sfixed**
 - GOOD: **outBOOLEAN**
 -)
)
- hwrite(
 - L: **inoutLINE**
 - VALUE: **inUNRESOLVED_ufixed**
 - JUSTIFIED: **inSIDEright**
 - FIELD: **inWIDTH 0**
 -)
)
- hwrite(
 - L: **inoutLINE**
 - VALUE: **inUNRESOLVED_sfixed**
 - JUSTIFIED: **inSIDEright**
 - FIELD: **inWIDTH 0**
 -)
)
- HREAD_common(
 - L: **inoutLINE**
 - slv: **outSTD_ULOGIC_VECTOR**
 - igood: **outBOOLEAN**
 - index: **outINTEGER**
 - constant bpoint: **inINTEGER**
 - constant message: **inBOOLEAN**
 - constant smath: **inBOOLEAN**
 -)
)
- errmes(constant mess: **inSTRING**)
- HREAD(L: **inoutLINE**, VALUE: **outUNRESOLVED_ufixed**)
- HREAD(
 - L: **inoutLINE**
 - VALUE: **outUNRESOLVED_ufixed**
 - GOOD: **outBOOLEAN**
 -)
)
- HREAD(L: **inoutLINE**, VALUE: **outUNRESOLVED_sfixed**)
- HREAD(
 - L: **inoutLINE**
 - VALUE: **outUNRESOLVED_sfixed**
 - GOOD: **outBOOLEAN**
 -)
)
- calculate_string_boundry(
 - arg: **inSTRING**
 - left_index: **outINTEGER**
 - right_index: **outINTEGER**
 -)
)

Libraries

- IEEE

Use Clauses

- MATH_REAL

Constants

- NAUF UNRESOLVED_ufixed(0 downto 1):=(others=>' 0 ')
- NASF UNRESOLVED_sfixed(0 downto 1):=(others=>' 0 ')
- NSLV STD_ULOGIC_VECTOR(0 downto 1):=(others=>' 0 ')
- fixedsynth_or_real BOOLEAN:=true
- match_logic_table stdlogic_table :=((‘U’,‘U’,‘U’,‘U’,‘U’,‘U’,‘U’,‘U’,‘ 1 ’), (‘U’,‘X’,‘X’,‘X’,‘X’,‘X’,‘X’,‘X’,‘ 1 ’), (‘U’,‘X’,‘ 1 ’,‘ 0 ’,‘X’,‘X’,‘ 1 ’,‘ 0 ’,‘ 1 ’), (‘U’,‘X’,‘ 0 ’,‘ 1 ’,‘X’,‘X’,‘ 0 ’,‘ 1 ’,‘ 1 ’), (‘U’,‘X’,‘X’,‘X’,‘X’,‘X’,‘X’,‘X’,‘ 1 ’), (‘U’,‘X’,‘X’,‘X’,‘X’,‘X’,‘X’,‘X’,‘ 1 ’), (‘U’,‘X’,‘ 1 ’,‘ 0 ’,‘X’,‘X’,‘ 1 ’,‘ 0 ’,‘ 1 ’), (‘U’,‘X’,‘ 0 ’,‘ 1 ’,‘X’,‘X’,‘ 0 ’,‘ 1 ’,‘ 1 ’), (‘ 1 ’,‘ 1 ’,‘ 1 ’,‘ 1 ’,‘ 1 ’,‘ 1 ’,‘ 1 ’,‘ 1 ’))
- MVL9_to_char char_indexed_by_MVL9 :=" UX01ZWLH- "
- char_to_MVL9 MVL9_indexed_by_char :=('U'=>'U','X'=>'X',' 0 '=>' 0 ',' 1 '=>' 1 ','Z'=>'Z','W'=>'W','L'=>'L','H'=>'H','-'=>'-',others=>'U')
- char_to_MVL9plus MVL9plus_indexed_by_char :=('U'=>'U','X'=>'X',' 0 '=>' 0 ',' 1 '=>' 1 ','Z'=>'Z','W'=>'W','L'=>'L','H'=>'H','-'=>'-',others=>error)
- NBSP CHARACTER:=CHARACTER'val(160)
- NUS STRING(2 to 1):=(others=>"")

Types

- stdlogic_table(STD_ULOGIC,STD_ULOGIC)STD_ULOGIC
- MVL9plus('U','X',' 0 ',' 1 ','Z','W','L','H','-',error)
- char_indexed_by_MVL9(STD_ULOGIC)CHARACTER
- MVL9_indexed_by_char(CHARACTER)STD_ULOGIC
- MVL9plus_indexed_by_char(CHARACTER)MVL9plus

Shared Variables

- xgood BOOLEAN
- nybble STD_ULOGIC_VECTOR(2 downto 0)
- c CHARACTER
- i INTEGER
- lastu BOOLEAN:=::false
- founddot BOOLEAN:=::false
- nybble STD_ULOGIC_VECTOR(3 downto 0)

6.1.1 Detailed Description

Definition at line 1466 of file [fixed_pkg_c.vhdl](#).

6.1.2 Member Function Documentation

6.1.2.1 UNRESOLVED_ufixed "*"(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 2470 of file [fixed_pkg_c.vhdl](#).

6.1.2.2 UNRESOLVED_sfixed "*"(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 2490 of file [fixed_pkg_c.vhdl](#).

6.1.2.3 UNRESOLVED_ufixed "*"(*lin* UNRESOLVED_ufixed , *rin* REAL) [Function]

Definition at line 5630 of file [fixed_pkg_c.vhdl](#).

6.1.2.4 **UNRESOLVED_ufixed** "*"(*lin* **REAL** , *rin* **UNRESOLVED_ufixed**) [Function]

Definition at line 5638 of file [fixed_pkg_c.vhdl](#).

6.1.2.5 **UNRESOLVED_sfixed** "*"(*lin* **UNRESOLVED_sfixed** , *rin* **REAL**) [Function]

Definition at line 5646 of file [fixed_pkg_c.vhdl](#).

6.1.2.6 **UNRESOLVED_sfixed** "*"(*lin* **REAL** , *rin* **UNRESOLVED_sfixed**) [Function]

Definition at line 5654 of file [fixed_pkg_c.vhdl](#).

6.1.2.7 **UNRESOLVED_ufixed** "*"(*lin* **UNRESOLVED_ufixed** , *rin* **NATURAL**) [Function]

Definition at line 5825 of file [fixed_pkg_c.vhdl](#).

6.1.2.8 **UNRESOLVED_ufixed** "*"(*lin* **NATURAL** , *rin* **UNRESOLVED_ufixed**) [Function]

Definition at line 5833 of file [fixed_pkg_c.vhdl](#).

6.1.2.9 **UNRESOLVED_sfixed** "*"(*lin* **UNRESOLVED_sfixed** , *rin* **INTEGER**) [Function]

Definition at line 5841 of file [fixed_pkg_c.vhdl](#).

6.1.2.10 **UNRESOLVED_sfixed** "*"(*lin* **INTEGER** , *rin* **UNRESOLVED_sfixed**) [Function]

Definition at line 5849 of file [fixed_pkg_c.vhdl](#).

6.1.2.11 **UNRESOLVED_ufixed** "+"(*lin* **UNRESOLVED_ufixed** , *rin* **UNRESOLVED_ufixed**) [Function]

Definition at line 2377 of file [fixed_pkg_c.vhdl](#).

6.1.2.12 **UNRESOLVED_sfixed** "+"(*lin* **UNRESOLVED_sfixed** , *rin* **UNRESOLVED_sfixed**) [Function]

Definition at line 2401 of file [fixed_pkg_c.vhdl](#).

6.1.2.13 **UNRESOLVED_ufixed** "+"(*lin* **UNRESOLVED_ufixed** , *rin* **REAL**) [Function]

Definition at line 5566 of file [fixed_pkg_c.vhdl](#).

6.1.2.14 **UNRESOLVED_ufixed** "+"(*lin* **REAL** , *rin* **UNRESOLVED_ufixed**) [Function]

Definition at line 5574 of file [fixed_pkg_c.vhdl](#).

6.1.2.15 **UNRESOLVED_sfixed** "+"(*lin* **UNRESOLVED_sfixed** , *rin* **REAL**) [Function]

Definition at line 5582 of file [fixed_pkg_c.vhdl](#).

6.1.2.16 **UNRESOLVED_sfixed "+"(*lin* REAL , *rin* UNRESOLVED_sfixed)** [Function]

Definition at line 5590 of file [fixed_pkg_c.vhdl](#).

6.1.2.17 **UNRESOLVED_ufixed "+"(*lin* UNRESOLVED_ufixed , *rin* NATURAL)** [Function]

Definition at line 5759 of file [fixed_pkg_c.vhdl](#).

6.1.2.18 **UNRESOLVED_ufixed "+"(*lin* NATURAL , *rin* UNRESOLVED_ufixed)** [Function]

Definition at line 5767 of file [fixed_pkg_c.vhdl](#).

6.1.2.19 **UNRESOLVED_sfixed "+"(*lin* UNRESOLVED_sfixed , *rin* INTEGER)** [Function]

Definition at line 5775 of file [fixed_pkg_c.vhdl](#).

6.1.2.20 **UNRESOLVED_sfixed "+"(*lin* INTEGER , *rin* UNRESOLVED_sfixed)** [Function]

Definition at line 5783 of file [fixed_pkg_c.vhdl](#).

6.1.2.21 **UNRESOLVED_sfixed "-"(*argin* UNRESOLVED_sfixed)** [Function]

Definition at line 2359 of file [fixed_pkg_c.vhdl](#).

6.1.2.22 **UNRESOLVED_ufixed "-"(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed)** [Function]

Definition at line 2424 of file [fixed_pkg_c.vhdl](#).

6.1.2.23 **UNRESOLVED_sfixed "-"(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed)** [Function]

Definition at line 2448 of file [fixed_pkg_c.vhdl](#).

6.1.2.24 **UNRESOLVED_ufixed "-"(*lin* UNRESOLVED_ufixed , *rin* REAL)** [Function]

Definition at line 5598 of file [fixed_pkg_c.vhdl](#).

6.1.2.25 **UNRESOLVED_ufixed "-"(*lin* REAL , *rin* UNRESOLVED_ufixed)** [Function]

Definition at line 5606 of file [fixed_pkg_c.vhdl](#).

6.1.2.26 **UNRESOLVED_sfixed "-"(*lin* UNRESOLVED_sfixed , *rin* REAL)** [Function]

Definition at line 5614 of file [fixed_pkg_c.vhdl](#).

6.1.2.27 **UNRESOLVED_sfixed "-"(*lin* REAL , *rin* UNRESOLVED_sfixed)** [Function]

Definition at line 5622 of file [fixed_pkg_c.vhdl](#).

6.1.2.28 **UNRESOLVED_ufixed** "-"(*lin* UNRESOLVED_ufixed , *rin* NATURAL) [Function]

Definition at line 5792 of file [fixed_pkg_c.vhdl](#).

6.1.2.29 **UNRESOLVED_ufixed** "-"(*lin* NATURAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 5800 of file [fixed_pkg_c.vhdl](#).

6.1.2.30 **UNRESOLVED_sfixed** "-"(*lin* UNRESOLVED_sfixed , *rin* INTEGER) [Function]

Definition at line 5808 of file [fixed_pkg_c.vhdl](#).

6.1.2.31 **UNRESOLVED_sfixed** "-"(*lin* INTEGER , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 5816 of file [fixed_pkg_c.vhdl](#).

6.1.2.32 **UNRESOLVED_ufixed** "/"(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 2510 of file [fixed_pkg_c.vhdl](#).

6.1.2.33 **UNRESOLVED_sfixed** "/"(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 2517 of file [fixed_pkg_c.vhdl](#).

6.1.2.34 **UNRESOLVED_ufixed** "/"(*lin* UNRESOLVED_ufixed , *rin* REAL) [Function]

Definition at line 5662 of file [fixed_pkg_c.vhdl](#).

6.1.2.35 **UNRESOLVED_ufixed** "/"(*lin* REAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 5670 of file [fixed_pkg_c.vhdl](#).

6.1.2.36 **UNRESOLVED_sfixed** "/"(*lin* UNRESOLVED_sfixed , *rin* REAL) [Function]

Definition at line 5678 of file [fixed_pkg_c.vhdl](#).

6.1.2.37 **UNRESOLVED_sfixed** "/"(*lin* REAL , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 5686 of file [fixed_pkg_c.vhdl](#).

6.1.2.38 **UNRESOLVED_ufixed** "/"(*lin* UNRESOLVED_ufixed , *rin* NATURAL) [Function]

Definition at line 5858 of file [fixed_pkg_c.vhdl](#).

6.1.2.39 **UNRESOLVED_ufixed** "/"(*lin* NATURAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 5866 of file [fixed_pkg_c.vhdl](#).

6.1.2.40 **UNRESOLVED_sfixed** "/"(*lin* UNRESOLVED_sfixed , *rin* INTEGER) [Function]

Definition at line 5874 of file [fixed_pkg_c.vhdl](#).

6.1.2.41 **UNRESOLVED_sfixed** "/"(*lin* INTEGER , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 5882 of file [fixed_pkg_c.vhdl](#).

6.1.2.42 **BOOLEAN** "/="(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 4013 of file [fixed_pkg_c.vhdl](#).

6.1.2.43 **BOOLEAN** "/="(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 4041 of file [fixed_pkg_c.vhdl](#).

6.1.2.44 **BOOLEAN** "/="(*lin* UNRESOLVED_ufixed , *rin* NATURAL) [Function]

Definition at line 5963 of file [fixed_pkg_c.vhdl](#).

6.1.2.45 **BOOLEAN** "/="(*lin* NATURAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 6076 of file [fixed_pkg_c.vhdl](#).

6.1.2.46 **BOOLEAN** "/="(*lin* UNRESOLVED_ufixed , *rin* REAL) [Function]

Definition at line 6189 of file [fixed_pkg_c.vhdl](#).

6.1.2.47 **BOOLEAN** "/="(*lin* REAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 6302 of file [fixed_pkg_c.vhdl](#).

6.1.2.48 **BOOLEAN** "/="(*lin* UNRESOLVED_sfixed , *rin* INTEGER) [Function]

Definition at line 6415 of file [fixed_pkg_c.vhdl](#).

6.1.2.49 **BOOLEAN** "/="(*lin* INTEGER , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 6528 of file [fixed_pkg_c.vhdl](#).

6.1.2.50 **BOOLEAN** "/="(*lin* UNRESOLVED_sfixed , *rin* REAL) [Function]

Definition at line 6641 of file [fixed_pkg_c.vhdl](#).

6.1.2.51 **BOOLEAN** "/="(*lin* REAL , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 6754 of file [fixed_pkg_c.vhdl](#).

6.1.2.52 **BOOLEAN** "*<*"(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 4125 of file [fixed_pkg_c.vhdl](#).

6.1.2.53 **BOOLEAN** "*<*"(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 4153 of file [fixed_pkg_c.vhdl](#).

6.1.2.54 **BOOLEAN** "*<*"(*lin* UNRESOLVED_ufixed , *rin* NATURAL) [Function]

Definition at line 5995 of file [fixed_pkg_c.vhdl](#).

6.1.2.55 **BOOLEAN** "*<*"(*lin* NATURAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 6108 of file [fixed_pkg_c.vhdl](#).

6.1.2.56 **BOOLEAN** "*<*"(*lin* UNRESOLVED_ufixed , *rin* REAL) [Function]

Definition at line 6221 of file [fixed_pkg_c.vhdl](#).

6.1.2.57 **BOOLEAN** "*<*"(*lin* REAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 6334 of file [fixed_pkg_c.vhdl](#).

6.1.2.58 **BOOLEAN** "*<*"(*lin* UNRESOLVED_sfixed , *rin* INTEGER) [Function]

Definition at line 6447 of file [fixed_pkg_c.vhdl](#).

6.1.2.59 **BOOLEAN** "*<*"(*lin* INTEGER , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 6560 of file [fixed_pkg_c.vhdl](#).

6.1.2.60 **BOOLEAN** "*<*"(*lin* UNRESOLVED_sfixed , *rin* REAL) [Function]

Definition at line 6673 of file [fixed_pkg_c.vhdl](#).

6.1.2.61 **BOOLEAN** "*<*"(*lin* REAL , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 6786 of file [fixed_pkg_c.vhdl](#).

6.1.2.62 **BOOLEAN** "*<=*"(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 4237 of file [fixed_pkg_c.vhdl](#).

6.1.2.63 **BOOLEAN** "*<=*"(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 4265 of file [fixed_pkg_c.vhdl](#).

6.1.2.64 **BOOLEAN** " \leq "(*lin UNRESOLVED_ufixed*, *rin NATURAL*) [Function]

Definition at line 5979 of file [fixed_pkg_c.vhdl](#).

6.1.2.65 **BOOLEAN** " \leq "(*lin NATURAL*, *rin UNRESOLVED_ufixed*) [Function]

Definition at line 6092 of file [fixed_pkg_c.vhdl](#).

6.1.2.66 **BOOLEAN** " \leq "(*lin UNRESOLVED_ufixed*, *rin REAL*) [Function]

Definition at line 6205 of file [fixed_pkg_c.vhdl](#).

6.1.2.67 **BOOLEAN** " \leq "(*lin REAL*, *rin UNRESOLVED_ufixed*) [Function]

Definition at line 6318 of file [fixed_pkg_c.vhdl](#).

6.1.2.68 **BOOLEAN** " \leq "(*lin UNRESOLVED_sfixed*, *rin INTEGER*) [Function]

Definition at line 6431 of file [fixed_pkg_c.vhdl](#).

6.1.2.69 **BOOLEAN** " \leq "(*lin INTEGER*, *rin UNRESOLVED_sfixed*) [Function]

Definition at line 6544 of file [fixed_pkg_c.vhdl](#).

6.1.2.70 **BOOLEAN** " \leq "(*lin UNRESOLVED_sfixed*, *rin REAL*) [Function]

Definition at line 6657 of file [fixed_pkg_c.vhdl](#).

6.1.2.71 **BOOLEAN** " \leq "(*lin REAL*, *rin UNRESOLVED_sfixed*) [Function]

Definition at line 6770 of file [fixed_pkg_c.vhdl](#).

6.1.2.72 **BOOLEAN** " $=$ "(*lin UNRESOLVED_ufixed*, *rin UNRESOLVED_ufixed*) [Function]

Definition at line 3957 of file [fixed_pkg_c.vhdl](#).

6.1.2.73 **BOOLEAN** " $=$ "(*lin UNRESOLVED_sfixed*, *rin UNRESOLVED_sfixed*) [Function]

Definition at line 3985 of file [fixed_pkg_c.vhdl](#).

6.1.2.74 **BOOLEAN** " $=$ "(*lin UNRESOLVED_ufixed*, *rin NATURAL*) [Function]

Definition at line 5955 of file [fixed_pkg_c.vhdl](#).

6.1.2.75 **BOOLEAN** " $=$ "(*lin NATURAL*, *rin UNRESOLVED_ufixed*) [Function]

Definition at line 6068 of file [fixed_pkg_c.vhdl](#).

6.1.2.76 **BOOLEAN** " $=$ "(*lin UNRESOLVED_ufixed*, *rin REAL*) [Function]

Definition at line 6181 of file [fixed_pkg_c.vhdl](#).

6.1.2.77 **BOOLEAN** " $=$ "(*lin REAL*, *rin UNRESOLVED_ufixed*) [Function]

Definition at line 6294 of file [fixed_pkg_c.vhdl](#).

6.1.2.78 **BOOLEAN** " $=$ "(*lin UNRESOLVED_sfixed*, *rin INTEGER*) [Function]

Definition at line 6407 of file [fixed_pkg_c.vhdl](#).

6.1.2.79 **BOOLEAN** " $=$ "(*lin INTEGER*, *rin UNRESOLVED_sfixed*) [Function]

Definition at line 6520 of file [fixed_pkg_c.vhdl](#).

6.1.2.80 **BOOLEAN** " $=$ "(*lin UNRESOLVED_sfixed*, *rin REAL*) [Function]

Definition at line 6633 of file [fixed_pkg_c.vhdl](#).

6.1.2.81 **BOOLEAN** " $=$ "(*lin REAL*, *rin UNRESOLVED_sfixed*) [Function]

Definition at line 6746 of file [fixed_pkg_c.vhdl](#).

6.1.2.82 **BOOLEAN** " $>$ "(*lin UNRESOLVED_ufixed*, *rin UNRESOLVED_ufixed*) [Function]

Definition at line 4069 of file [fixed_pkg_c.vhdl](#).

6.1.2.83 **BOOLEAN** " $>$ "(*lin UNRESOLVED_sfixed*, *rin UNRESOLVED_sfixed*) [Function]

Definition at line 4097 of file [fixed_pkg_c.vhdl](#).

6.1.2.84 **BOOLEAN** " $>$ "(*lin UNRESOLVED_ufixed*, *rin NATURAL*) [Function]

Definition at line 5987 of file [fixed_pkg_c.vhdl](#).

6.1.2.85 **BOOLEAN** " $>$ "(*lin NATURAL*, *rin UNRESOLVED_ufixed*) [Function]

Definition at line 6100 of file [fixed_pkg_c.vhdl](#).

6.1.2.86 **BOOLEAN** " $>$ "(*lin UNRESOLVED_ufixed*, *rin REAL*) [Function]

Definition at line 6213 of file [fixed_pkg_c.vhdl](#).

6.1.2.87 **BOOLEAN** " $>$ "(*lin REAL*, *rin UNRESOLVED_ufixed*) [Function]

Definition at line 6326 of file [fixed_pkg_c.vhdl](#).

6.1.2.88 **BOOLEAN** ">"(*lin* UNRESOLVED_sfixed , *rin* INTEGER) [Function]

Definition at line 6439 of file [fixed_pkg_c.vhdl](#).

6.1.2.89 **BOOLEAN** ">"(*lin* INTEGER , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 6552 of file [fixed_pkg_c.vhdl](#).

6.1.2.90 **BOOLEAN** ">"(*lin* UNRESOLVED_sfixed , *rin* REAL) [Function]

Definition at line 6665 of file [fixed_pkg_c.vhdl](#).

6.1.2.91 **BOOLEAN** ">"(*lin* REAL , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 6778 of file [fixed_pkg_c.vhdl](#).

6.1.2.92 **BOOLEAN** ">="(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 4181 of file [fixed_pkg_c.vhdl](#).

6.1.2.93 **BOOLEAN** ">="(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 4209 of file [fixed_pkg_c.vhdl](#).

6.1.2.94 **BOOLEAN** ">="(*lin* UNRESOLVED_ufixed , *rin* NATURAL) [Function]

Definition at line 5971 of file [fixed_pkg_c.vhdl](#).

6.1.2.95 **BOOLEAN** ">="(*lin* NATURAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 6084 of file [fixed_pkg_c.vhdl](#).

6.1.2.96 **BOOLEAN** ">="(*lin* UNRESOLVED_ufixed , *rin* REAL) [Function]

Definition at line 6197 of file [fixed_pkg_c.vhdl](#).

6.1.2.97 **BOOLEAN** ">="(*lin* REAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 6310 of file [fixed_pkg_c.vhdl](#).

6.1.2.98 **BOOLEAN** ">="(*lin* UNRESOLVED_sfixed , *rin* INTEGER) [Function]

Definition at line 6423 of file [fixed_pkg_c.vhdl](#).

6.1.2.99 **BOOLEAN** ">="(*lin* INTEGER , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 6536 of file [fixed_pkg_c.vhdl](#).

6.1.2.100 **BOOLEAN** ">="(*lin* UNRESOLVED_sfixed , *rin* REAL) [Function]

Definition at line 6649 of file [fixed_pkg_c.vhdl](#).

6.1.2.101 **BOOLEAN** ">="(*lin* REAL , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 6762 of file [fixed_pkg_c.vhdl](#).

6.1.2.102 **UNRESOLVED_sfixed** "abs"(*argin* UNRESOLVED_sfixed) [Function]

Definition at line 2341 of file [fixed_pkg_c.vhdl](#).

6.1.2.103 **UNRESOLVED_ufixed** "and"(*Lin* UNRESOLVED_ufixed , *Rin* UNRESOLVED_ufixed) [Function]

Definition at line 3187 of file [fixed_pkg_c.vhdl](#).

6.1.2.104 **UNRESOLVED_sfixed** "and"(*Lin* UNRESOLVED_sfixed , *Rin* UNRESOLVED_sfixed) [Function]

Definition at line 3284 of file [fixed_pkg_c.vhdl](#).

6.1.2.105 **UNRESOLVED_ufixed** "and"(*Lin* STD_ULOGIC , *Rin* UNRESOLVED_ufixed) [Function]

Definition at line 3375 of file [fixed_pkg_c.vhdl](#).

6.1.2.106 **UNRESOLVED_ufixed** "and"(*Lin* UNRESOLVED_ufixed , *Rin* STD_ULOGIC) [Function]

Definition at line 3385 of file [fixed_pkg_c.vhdl](#).

6.1.2.107 **UNRESOLVED_sfixed** "and"(*Lin* STD_ULOGIC , *Rin* UNRESOLVED_sfixed) [Function]

Definition at line 3495 of file [fixed_pkg_c.vhdl](#).

6.1.2.108 **UNRESOLVED_sfixed** "and"(*Lin* UNRESOLVED_sfixed , *Rin* STD_ULOGIC) [Function]

Definition at line 3505 of file [fixed_pkg_c.vhdl](#).

6.1.2.109 **UNRESOLVED_ufixed** "mod"(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 2750 of file [fixed_pkg_c.vhdl](#).

6.1.2.110 **UNRESOLVED_sfixed** "mod"(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 2759 of file [fixed_pkg_c.vhdl](#).

6.1.2.111 **UNRESOLVED_ufixed "mod"**(*lin* UNRESOLVED_ufixed , *rin* REAL) [Function]

Definition at line 5726 of file [fixed_pkg_c.vhdl](#).

6.1.2.112 **UNRESOLVED_ufixed "mod"**(*lin* REAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 5734 of file [fixed_pkg_c.vhdl](#).

6.1.2.113 **UNRESOLVED_sfixed "mod"**(*lin* UNRESOLVED_sfixed , *rin* REAL) [Function]

Definition at line 5742 of file [fixed_pkg_c.vhdl](#).

6.1.2.114 **UNRESOLVED_sfixed "mod"**(*lin* REAL , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 5750 of file [fixed_pkg_c.vhdl](#).

6.1.2.115 **UNRESOLVED_ufixed "mod"**(*lin* UNRESOLVED_ufixed , *rin* NATURAL) [Function]

Definition at line 5922 of file [fixed_pkg_c.vhdl](#).

6.1.2.116 **UNRESOLVED_ufixed "mod"**(*lin* NATURAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 5930 of file [fixed_pkg_c.vhdl](#).

6.1.2.117 **UNRESOLVED_sfixed "mod"**(*lin* UNRESOLVED_sfixed , *rin* INTEGER) [Function]

Definition at line 5938 of file [fixed_pkg_c.vhdl](#).

6.1.2.118 **UNRESOLVED_sfixed "mod"**(*lin* INTEGER , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 5946 of file [fixed_pkg_c.vhdl](#).

6.1.2.119 **UNRESOLVED_ufixed "nand"**(*Lin* UNRESOLVED_ufixed , *Rin* UNRESOLVED_ufixed) [Function]

Definition at line 3217 of file [fixed_pkg_c.vhdl](#).

6.1.2.120 **UNRESOLVED_sfixed "nand"**(*Lin* UNRESOLVED_sfixed , *Rin* UNRESOLVED_sfixed) [Function]

Definition at line 3314 of file [fixed_pkg_c.vhdl](#).

6.1.2.121 **UNRESOLVED_ufixed "nand"**(*Lin* STD_ULOGIC , *Rin* UNRESOLVED_ufixed) [Function]

Definition at line 3415 of file [fixed_pkg_c.vhdl](#).

6.1.2.122 **UNRESOLVED_ufixed "nand"**(*Lin* UNRESOLVED_ufixed , *Rin* STD_ULOGIC) [Function]

Definition at line 3425 of file [fixed_pkg_c.vhdl](#).

6.1.2.123 **UNRESOLVED_sfixed "nand"**(*Lin* **STD_ULOGIC** , *Rin* **UNRESOLVED_sfixed**) [Function]

Definition at line 3535 of file [fixed_pkg_c.vhdl](#).

6.1.2.124 **UNRESOLVED_sfixed "nand"**(*Lin* **UNRESOLVED_sfixed** , *Rin* **STD_ULOGIC**) [Function]

Definition at line 3545 of file [fixed_pkg_c.vhdl](#).

6.1.2.125 **UNRESOLVED_ufixed "nor"**(*Lin* **UNRESOLVED_ufixed** , *Rin* **UNRESOLVED_ufixed**) [Function]

Definition at line 3232 of file [fixed_pkg_c.vhdl](#).

6.1.2.126 **UNRESOLVED_sfixed "nor"**(*Lin* **UNRESOLVED_sfixed** , *Rin* **UNRESOLVED_sfixed**) [Function]

Definition at line 3329 of file [fixed_pkg_c.vhdl](#).

6.1.2.127 **UNRESOLVED_ufixed "nor"**(*Lin* **STD_ULOGIC** , *Rin* **UNRESOLVED_ufixed**) [Function]

Definition at line 3435 of file [fixed_pkg_c.vhdl](#).

6.1.2.128 **UNRESOLVED_ufixed "nor"**(*Lin* **UNRESOLVED_ufixed** , *Rin* **STD_ULOGIC**) [Function]

Definition at line 3445 of file [fixed_pkg_c.vhdl](#).

6.1.2.129 **UNRESOLVED_sfixed "nor"**(*Lin* **STD_ULOGIC** , *Rin* **UNRESOLVED_sfixed**) [Function]

Definition at line 3555 of file [fixed_pkg_c.vhdl](#).

6.1.2.130 **UNRESOLVED_sfixed "nor"**(*Lin* **UNRESOLVED_sfixed** , *Rin* **STD_ULOGIC**) [Function]

Definition at line 3565 of file [fixed_pkg_c.vhdl](#).

6.1.2.131 **UNRESOLVED_ufixed "not"**(*Lin* **UNRESOLVED_ufixed**) [Function]

Definition at line 3180 of file [fixed_pkg_c.vhdl](#).

6.1.2.132 **UNRESOLVED_sfixed "not"**(*Lin* **UNRESOLVED_sfixed**) [Function]

Definition at line 3277 of file [fixed_pkg_c.vhdl](#).

6.1.2.133 **UNRESOLVED_ufixed "or"**(*Lin* **UNRESOLVED_ufixed** , *Rin* **UNRESOLVED_ufixed**) [Function]

Definition at line 3202 of file [fixed_pkg_c.vhdl](#).

6.1.2.134 **UNRESOLVED_sfixed "or"**(*Lin* UNRESOLVED_sfixed , *Rin* UNRESOLVED_sfixed)
[Function]

Definition at line 3299 of file [fixed_pkg_c.vhdl](#).

6.1.2.135 **UNRESOLVED_ufixed "or"**(*Lin* STD_ULOGIC , *Rin* UNRESOLVED_ufixed) [Function]

Definition at line 3395 of file [fixed_pkg_c.vhdl](#).

6.1.2.136 **UNRESOLVED_ufixed "or"**(*Lin* UNRESOLVED_ufixed , *Rin* STD_ULOGIC) [Function]

Definition at line 3405 of file [fixed_pkg_c.vhdl](#).

6.1.2.137 **UNRESOLVED_sfixed "or"**(*Lin* STD_ULOGIC , *Rin* UNRESOLVED_sfixed) [Function]

Definition at line 3515 of file [fixed_pkg_c.vhdl](#).

6.1.2.138 **UNRESOLVED_sfixed "or"**(*Lin* UNRESOLVED_sfixed , *Rin* STD_ULOGIC) [Function]

Definition at line 3525 of file [fixed_pkg_c.vhdl](#).

6.1.2.139 **UNRESOLVED_ufixed "rem"**(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed)
[Function]

Definition at line 2643 of file [fixed_pkg_c.vhdl](#).

6.1.2.140 **UNRESOLVED_sfixed "rem"**(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed)
[Function]

Definition at line 2653 of file [fixed_pkg_c.vhdl](#).

6.1.2.141 **UNRESOLVED_ufixed "rem"**(*lin* UNRESOLVED_ufixed , *rin* REAL) [Function]

Definition at line 5694 of file [fixed_pkg_c.vhdl](#).

6.1.2.142 **UNRESOLVED_ufixed "rem"**(*lin* REAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 5702 of file [fixed_pkg_c.vhdl](#).

6.1.2.143 **UNRESOLVED_sfixed "rem"**(*lin* UNRESOLVED_sfixed , *rin* REAL) [Function]

Definition at line 5710 of file [fixed_pkg_c.vhdl](#).

6.1.2.144 **UNRESOLVED_sfixed "rem"**(*lin* REAL , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 5718 of file [fixed_pkg_c.vhdl](#).

6.1.2.145 **UNRESOLVED_ufixed "rem"**(*lin* UNRESOLVED_ufixed , *rin* NATURAL) [Function]

Definition at line 5890 of file [fixed_pkg_c.vhdl](#).

6.1.2.146 **UNRESOLVED_ufixed "rem"**(*l*in*NATURAL , r*in*UNRESOLVED_ufixed*) [Function]

Definition at line 5898 of file [fixed_pkg_c.vhdl](#).

6.1.2.147 **UNRESOLVED_sfixed "rem"**(*l*in*UNRESOLVED_sfixed , r*in*INTEGER*) [Function]

Definition at line 5906 of file [fixed_pkg_c.vhdl](#).

6.1.2.148 **UNRESOLVED_sfixed "rem"**(*l*in*INTEGER , r*in*UNRESOLVED_sfixed*) [Function]

Definition at line 5914 of file [fixed_pkg_c.vhdl](#).

6.1.2.149 **UNRESOLVED_ufixed "rol"**(*ARG*in*UNRESOLVED_ufixed , COUNT*in*INTEGER*) [Function]

Definition at line 3018 of file [fixed_pkg_c.vhdl](#).

6.1.2.150 **UNRESOLVED_sfixed "rol"**(*ARG*in*UNRESOLVED_sfixed , COUNT*in*INTEGER*) [Function]

Definition at line 3086 of file [fixed_pkg_c.vhdl](#).

6.1.2.151 **UNRESOLVED_ufixed "ror"**(*ARG*in*UNRESOLVED_ufixed , COUNT*in*INTEGER*) [Function]

Definition at line 3029 of file [fixed_pkg_c.vhdl](#).

6.1.2.152 **UNRESOLVED_sfixed "ror"**(*ARG*in*UNRESOLVED_sfixed , COUNT*in*INTEGER*) [Function]

Definition at line 3097 of file [fixed_pkg_c.vhdl](#).

6.1.2.153 **UNRESOLVED_ufixed "sla"**(*ARG*in*UNRESOLVED_ufixed , COUNT*in*INTEGER*) [Function]

Definition at line 3040 of file [fixed_pkg_c.vhdl](#).

6.1.2.154 **UNRESOLVED_sfixed "sla"**(*ARG*in*UNRESOLVED_sfixed , COUNT*in*INTEGER*) [Function]

Definition at line 3108 of file [fixed_pkg_c.vhdl](#).

6.1.2.155 **UNRESOLVED_ufixed "sll"**(*ARG*in*UNRESOLVED_ufixed , COUNT*in*INTEGER*) [Function]

Definition at line 2996 of file [fixed_pkg_c.vhdl](#).

6.1.2.156 **UNRESOLVED_sfixed "sll"**(*ARG*in*UNRESOLVED_sfixed , COUNT*in*INTEGER*) [Function]

Definition at line 3064 of file [fixed_pkg_c.vhdl](#).

6.1.2.157 **SIGNED "sra"**(*argin SIGNED , count*in*INTEGER*) [Function]

Definition at line 1497 of file [fixed_pkg_c.vhdl](#).

6.1.2.158 **UNRESOLVED_ufixed "sra"**(*ARGin* UNRESOLVED_ufixed , *COUNTin* INTEGER) [Function]

Definition at line 3052 of file [fixed_pkg_c.vhdl](#).

6.1.2.159 **UNRESOLVED_sfixed "sra"**(*ARGin* UNRESOLVED_sfixed , *COUNTin* INTEGER) [Function]

Definition at line 3124 of file [fixed_pkg_c.vhdl](#).

6.1.2.160 **UNRESOLVED_ufixed "srl"**(*ARGin* UNRESOLVED_ufixed , *COUNTin* INTEGER) [Function]

Definition at line 3007 of file [fixed_pkg_c.vhdl](#).

6.1.2.161 **UNRESOLVED_sfixed "srl"**(*ARGin* UNRESOLVED_sfixed , *COUNTin* INTEGER) [Function]

Definition at line 3075 of file [fixed_pkg_c.vhdl](#).

6.1.2.162 **UNRESOLVED_ufixed "xnor"**(*Lin* UNRESOLVED_ufixed , *Rin* UNRESOLVED_ufixed)
[Function]

Definition at line 3262 of file [fixed_pkg_c.vhdl](#).

6.1.2.163 **UNRESOLVED_sfixed "xnor"**(*Lin* UNRESOLVED_sfixed , *Rin* UNRESOLVED_sfixed)
[Function]

Definition at line 3359 of file [fixed_pkg_c.vhdl](#).

6.1.2.164 **UNRESOLVED_ufixed "xnor"**(*Lin* STD_ULOGIC , *Rin* UNRESOLVED_ufixed) [Function]

Definition at line 3475 of file [fixed_pkg_c.vhdl](#).

6.1.2.165 **UNRESOLVED_ufixed "xnor"**(*Lin* UNRESOLVED_ufixed , *Rin* STD_ULOGIC) [Function]

Definition at line 3485 of file [fixed_pkg_c.vhdl](#).

6.1.2.166 **UNRESOLVED_sfixed "xnor"**(*Lin* STD_ULOGIC , *Rin* UNRESOLVED_sfixed) [Function]

Definition at line 3595 of file [fixed_pkg_c.vhdl](#).

6.1.2.167 **UNRESOLVED_sfixed "xnor"**(*Lin* UNRESOLVED_sfixed , *Rin* STD_ULOGIC) [Function]

Definition at line 3605 of file [fixed_pkg_c.vhdl](#).

6.1.2.168 **UNRESOLVED_ufixed "xor"**(*Lin* UNRESOLVED_ufixed , *Rin* UNRESOLVED_ufixed)
[Function]

Definition at line 3247 of file [fixed_pkg_c.vhdl](#).

6.1.2.169 **UNRESOLVED_sfixed "xor"**(*Lin UNRESOLVED_sfixed , Rin UNRESOLVED_sfixed*) [Function]

Definition at line 3344 of file [fixed_pkg_c.vhdl](#).

6.1.2.170 **UNRESOLVED_ufixed "xor"**(*Lin STD_ULOGIC , Rin UNRESOLVED_ufixed*) [Function]

Definition at line 3455 of file [fixed_pkg_c.vhdl](#).

6.1.2.171 **UNRESOLVED_ufixed "xor"**(*Lin UNRESOLVED_ufixed , Rin STD_ULOGIC*) [Function]

Definition at line 3465 of file [fixed_pkg_c.vhdl](#).

6.1.2.172 **UNRESOLVED_sfixed "xor"**(*Lin STD_ULOGIC , Rin UNRESOLVED_sfixed*) [Function]

Definition at line 3575 of file [fixed_pkg_c.vhdl](#).

6.1.2.173 **UNRESOLVED_sfixed "xor"**(*Lin UNRESOLVED_sfixed , Rin STD_ULOGIC*) [Function]

Definition at line 3585 of file [fixed_pkg_c.vhdl](#).

6.1.2.174 **STD_ULOGIC \?/=**(*lin STD_ULOGIC , rin STD_ULOGIC*) [Function]

Definition at line 1617 of file [fixed_pkg_c.vhdl](#).

6.1.2.175 **std_ulogic \?/=**(*Lin UNSIGNED , Rin UNSIGNED*) [Function]

Definition at line 1691 of file [fixed_pkg_c.vhdl](#).

6.1.2.176 **std_ulogic \?/=**(*Lin SIGNED , Rin SIGNED*) [Function]

Definition at line 1724 of file [fixed_pkg_c.vhdl](#).

6.1.2.177 **STD_ULOGIC \?/=**(*Lin UNRESOLVED_ufixed , Rin UNRESOLVED_ufixed*) [Function]

Definition at line 3698 of file [fixed_pkg_c.vhdl](#).

6.1.2.178 **STD_ULOGIC \?/=**(*Lin UNRESOLVED_sfixed , Rin UNRESOLVED_sfixed*) [Function]

Definition at line 3824 of file [fixed_pkg_c.vhdl](#).

6.1.2.179 **STD_ULOGIC \?/=**(*lin UNRESOLVED_ufixed , rin NATURAL*) [Function]

Definition at line 6011 of file [fixed_pkg_c.vhdl](#).

6.1.2.180 **STD_ULOGIC \?/=**(*lin NATURAL , rin UNRESOLVED_ufixed*) [Function]

Definition at line 6124 of file [fixed_pkg_c.vhdl](#).

6.1.2.181 **STD_ULOGIC** \?/= (*lin* UNRESOLVED_ufixed , *rin* REAL) [Function]

Definition at line 6237 of file [fixed_pkg_c.vhdl](#).

6.1.2.182 **STD_ULOGIC** \?/= (*lin* REAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 6350 of file [fixed_pkg_c.vhdl](#).

6.1.2.183 **STD_ULOGIC** \?/= (*lin* UNRESOLVED_sfixed , *rin* INTEGER) [Function]

Definition at line 6463 of file [fixed_pkg_c.vhdl](#).

6.1.2.184 **STD_ULOGIC** \?/= (*lin* INTEGER , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 6576 of file [fixed_pkg_c.vhdl](#).

6.1.2.185 **STD_ULOGIC** \?/= (*lin* UNRESOLVED_sfixed , *rin* REAL) [Function]

Definition at line 6689 of file [fixed_pkg_c.vhdl](#).

6.1.2.186 **STD_ULOGIC** \?/= (*lin* REAL , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 6802 of file [fixed_pkg_c.vhdl](#).

6.1.2.187 **STD_ULOGIC** \?<= (*Lin* UNSIGNED , *Rin* UNSIGNED) [Function]

Definition at line 1957 of file [fixed_pkg_c.vhdl](#).

6.1.2.188 **STD_ULOGIC** \?<= (*Lin* SIGNED , *Rin* SIGNED) [Function]

Definition at line 1990 of file [fixed_pkg_c.vhdl](#).

6.1.2.189 **STD_ULOGIC** \?<= (*Lin* UNRESOLVED_ufixed , *Rin* UNRESOLVED_ufixed) [Function]

Definition at line 3782 of file [fixed_pkg_c.vhdl](#).

6.1.2.190 **STD_ULOGIC** \?<= (*Lin* UNRESOLVED_sfixed , *Rin* UNRESOLVED_sfixed) [Function]

Definition at line 3908 of file [fixed_pkg_c.vhdl](#).

6.1.2.191 **STD_ULOGIC** \?<= (*lin* UNRESOLVED_ufixed , *rin* NATURAL) [Function]

Definition at line 6027 of file [fixed_pkg_c.vhdl](#).

6.1.2.192 **STD_ULOGIC** \?<= (*lin* NATURAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 6140 of file [fixed_pkg_c.vhdl](#).

6.1.2.193 **STD_ULOGIC**!<=(*lin* UNRESOLVED_ufixed , *rin* REAL) [Function]

Definition at line 6253 of file [fixed_pkg_c.vhdl](#).

6.1.2.194 **STD_ULOGIC**!<=(*lin* REAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 6366 of file [fixed_pkg_c.vhdl](#).

6.1.2.195 **STD_ULOGIC**!<=(*lin* UNRESOLVED_sfixed , *rin* INTEGER) [Function]

Definition at line 6479 of file [fixed_pkg_c.vhdl](#).

6.1.2.196 **STD_ULOGIC**!<=(*lin* INTEGER , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 6592 of file [fixed_pkg_c.vhdl](#).

6.1.2.197 **STD_ULOGIC**!<=(*lin* UNRESOLVED_sfixed , *rin* REAL) [Function]

Definition at line 6705 of file [fixed_pkg_c.vhdl](#).

6.1.2.198 **STD_ULOGIC**!<=(*lin* REAL , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 6818 of file [fixed_pkg_c.vhdl](#).

6.1.2.199 **STD_ULOGIC**!<!(*Lin* UNSIGNED , *Rin* UNSIGNED) [Function]

Definition at line 1893 of file [fixed_pkg_c.vhdl](#).

6.1.2.200 **STD_ULOGIC**!<!(*Lin* SIGNED , *Rin* SIGNED) [Function]

Definition at line 1926 of file [fixed_pkg_c.vhdl](#).

6.1.2.201 **STD_ULOGIC**!<!(*Lin* UNRESOLVED_ufixed , *Rin* UNRESOLVED_ufixed) [Function]

Definition at line 3761 of file [fixed_pkg_c.vhdl](#).

6.1.2.202 **STD_ULOGIC**!<!(*Lin* UNRESOLVED_sfixed , *Rin* UNRESOLVED_sfixed) [Function]

Definition at line 3887 of file [fixed_pkg_c.vhdl](#).

6.1.2.203 **STD_ULOGIC**!<!(*lin* UNRESOLVED_ufixed , *rin* NATURAL) [Function]

Definition at line 6043 of file [fixed_pkg_c.vhdl](#).

6.1.2.204 **STD_ULOGIC**!<!(*lin* NATURAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 6156 of file [fixed_pkg_c.vhdl](#).

6.1.2.205 **STD_ULONGIC** \?<\(*lin* UNRESOLVED_ufixed , *rin* REAL) [Function]

Definition at line 6269 of file [fixed_pkg_c.vhdl](#).

6.1.2.206 **STD_ULONGIC** \?<\(*lin* REAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 6382 of file [fixed_pkg_c.vhdl](#).

6.1.2.207 **STD_ULONGIC** \?<\(*lin* UNRESOLVED_sfixed , *rin* INTEGER) [Function]

Definition at line 6495 of file [fixed_pkg_c.vhdl](#).

6.1.2.208 **STD_ULONGIC** \?<\(*lin* INTEGER , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 6608 of file [fixed_pkg_c.vhdl](#).

6.1.2.209 **STD_ULONGIC** \?<\(*lin* UNRESOLVED_sfixed , *rin* REAL) [Function]

Definition at line 6721 of file [fixed_pkg_c.vhdl](#).

6.1.2.210 **STD_ULONGIC** \?<\(*lin* REAL , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 6834 of file [fixed_pkg_c.vhdl](#).

6.1.2.211 **STD_ULONGIC** \?=\<(*lin* STD_ULONGIC , *rin* STD_ULONGIC) [Function]

Definition at line 1613 of file [fixed_pkg_c.vhdl](#).

6.1.2.212 **STD_ULONGIC** \?=\<(*Lin* UNSIGNED , *Rin* UNSIGNED) [Function]

Definition at line 1623 of file [fixed_pkg_c.vhdl](#).

6.1.2.213 **std_ulogic** \?=\<(*Lin* SIGNED , *Rin* SIGNED) [Function]

Definition at line 1658 of file [fixed_pkg_c.vhdl](#).

6.1.2.214 **STD_ULONGIC** \?=\<(*Lin* UNRESOLVED_ufixed , *Rin* UNRESOLVED_ufixed) [Function]

Definition at line 3677 of file [fixed_pkg_c.vhdl](#).

6.1.2.215 **STD_ULONGIC** \?=\<(*Lin* UNRESOLVED_sfixed , *Rin* UNRESOLVED_sfixed) [Function]

Definition at line 3803 of file [fixed_pkg_c.vhdl](#).

6.1.2.216 **STD_ULONGIC** \?=\<(*lin* UNRESOLVED_ufixed , *rin* NATURAL) [Function]

Definition at line 6003 of file [fixed_pkg_c.vhdl](#).

6.1.2.217 **STD_ULOGIC** \?:\(*lin* NATURAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 6116 of file [fixed_pkg_c.vhdl](#).

6.1.2.218 **STD_ULOGIC** \?:\(*lin* UNRESOLVED_ufixed , *rin* REAL) [Function]

Definition at line 6229 of file [fixed_pkg_c.vhdl](#).

6.1.2.219 **STD_ULOGIC** \?:\(*lin* REAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 6342 of file [fixed_pkg_c.vhdl](#).

6.1.2.220 **STD_ULOGIC** \?:\(*lin* UNRESOLVED_sfixed , *rin* INTEGER) [Function]

Definition at line 6455 of file [fixed_pkg_c.vhdl](#).

6.1.2.221 **STD_ULOGIC** \?:\(*lin* INTEGER , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 6568 of file [fixed_pkg_c.vhdl](#).

6.1.2.222 **STD_ULOGIC** \?:\(*lin* UNRESOLVED_sfixed , *rin* REAL) [Function]

Definition at line 6681 of file [fixed_pkg_c.vhdl](#).

6.1.2.223 **STD_ULOGIC** \?:\(*lin* REAL , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 6794 of file [fixed_pkg_c.vhdl](#).

6.1.2.224 **STD_ULOGIC** \?>:\(*Lin* UNSIGNED , *Rin* UNSIGNED) [Function]

Definition at line 1829 of file [fixed_pkg_c.vhdl](#).

6.1.2.225 **STD_ULOGIC** \?>:\(*Lin* SIGNED , *Rin* SIGNED) [Function]

Definition at line 1862 of file [fixed_pkg_c.vhdl](#).

6.1.2.226 **STD_ULOGIC** \?>:\(*Lin* UNRESOLVED_ufixed , *Rin* UNRESOLVED_ufixed) [Function]

Definition at line 3740 of file [fixed_pkg_c.vhdl](#).

6.1.2.227 **STD_ULOGIC** \?>:\(*Lin* UNRESOLVED_sfixed , *Rin* UNRESOLVED_sfixed) [Function]

Definition at line 3866 of file [fixed_pkg_c.vhdl](#).

6.1.2.228 **STD_ULOGIC** \?>:\(*lin* UNRESOLVED_ufixed , *rin* NATURAL) [Function]

Definition at line 6019 of file [fixed_pkg_c.vhdl](#).

6.1.2.229 **STD_ULOGIC**!>=(*lin* NATURAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 6132 of file [fixed_pkg_c.vhdl](#).

6.1.2.230 **STD_ULOGIC**!>=(*lin* UNRESOLVED_ufixed , *rin* REAL) [Function]

Definition at line 6245 of file [fixed_pkg_c.vhdl](#).

6.1.2.231 **STD_ULOGIC**!>=(*lin* REAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 6358 of file [fixed_pkg_c.vhdl](#).

6.1.2.232 **STD_ULOGIC**!>=(*lin* UNRESOLVED_sfixed , *rin* INTEGER) [Function]

Definition at line 6471 of file [fixed_pkg_c.vhdl](#).

6.1.2.233 **STD_ULOGIC**!>=(*lin* INTEGER , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 6584 of file [fixed_pkg_c.vhdl](#).

6.1.2.234 **STD_ULOGIC**!>=(*lin* UNRESOLVED_sfixed , *rin* REAL) [Function]

Definition at line 6697 of file [fixed_pkg_c.vhdl](#).

6.1.2.235 **STD_ULOGIC**!>=(*lin* REAL , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 6810 of file [fixed_pkg_c.vhdl](#).

6.1.2.236 **STD_ULOGIC**!>!(*Lin* UNSIGNED , *Rin* UNSIGNED) [Function]

Definition at line 1765 of file [fixed_pkg_c.vhdl](#).

6.1.2.237 **STD_ULOGIC**!>!(*Lin* SIGNED , *Rin* SIGNED) [Function]

Definition at line 1798 of file [fixed_pkg_c.vhdl](#).

6.1.2.238 **STD_ULOGIC**!>!(*Lin* UNRESOLVED_ufixed , *Rin* UNRESOLVED_ufixed) [Function]

Definition at line 3719 of file [fixed_pkg_c.vhdl](#).

6.1.2.239 **STD_ULOGIC**!>!(*Lin* UNRESOLVED_sfixed , *Rin* UNRESOLVED_sfixed) [Function]

Definition at line 3845 of file [fixed_pkg_c.vhdl](#).

6.1.2.240 **STD_ULOGIC**!>!(*lin* UNRESOLVED_ufixed , *rin* NATURAL) [Function]

Definition at line 6035 of file [fixed_pkg_c.vhdl](#).

6.1.2.241 **STD_ULOGIC**!>!(*lin* NATURAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 6148 of file [fixed_pkg_c.vhdl](#).

6.1.2.242 **STD_ULOGIC**!>!(*lin* UNRESOLVED_ufixed , *rin* REAL) [Function]

Definition at line 6261 of file [fixed_pkg_c.vhdl](#).

6.1.2.243 **STD_ULOGIC**!>!(*lin* REAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 6374 of file [fixed_pkg_c.vhdl](#).

6.1.2.244 **STD_ULOGIC**!>!(*lin* UNRESOLVED_sfixed , *rin* INTEGER) [Function]

Definition at line 6487 of file [fixed_pkg_c.vhdl](#).

6.1.2.245 **STD_ULOGIC**!>!(*lin* INTEGER , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 6600 of file [fixed_pkg_c.vhdl](#).

6.1.2.246 **STD_ULOGIC**!>!(*lin* UNRESOLVED_sfixed , *rin* REAL) [Function]

Definition at line 6713 of file [fixed_pkg_c.vhdl](#).

6.1.2.247 **STD_ULOGIC**!>!(*lin* REAL , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 6826 of file [fixed_pkg_c.vhdl](#).

6.1.2.248 **add_carry(*L* inUNRESOLVED_ufixed , *R* inUNRESOLVED_ufixed , *c_in* inSTD_ULOGIC , *result* outUNRESOLVED_ufixed , *c_out* outSTD_ULOGIC)** [Procedure]

Definition at line 2842 of file [fixed_pkg_c.vhdl](#).

6.1.2.249 **add_carry(*L* inUNRESOLVED_sfixed , *R* inUNRESOLVED_sfixed , *c_in* inSTD_ULOGIC , *result* outUNRESOLVED_sfixed , *c_out* outSTD_ULOGIC)** [Procedure]

Definition at line 2872 of file [fixed_pkg_c.vhdl](#).

6.1.2.250 **STD_LOGIC and_reduce(*argin* STD_ULOGIC_VECTOR)** [Function]

Definition at line 1534 of file [fixed_pkg_c.vhdl](#).

6.1.2.251 **STD_ULOGIC and_reduce(*lin* UNRESOLVED_ufixed)** [Function]

Definition at line 3616 of file [fixed_pkg_c.vhdl](#).

6.1.2.252 **STD_ULOGIC and_reduce(*lin* UNRESOLVED_sfixed)** [Function]

Definition at line 3646 of file [fixed_pkg_c.vhdl](#).

6.1.2.253 calculate_string_boundry(*arg* inSTRING , *left_index* outINTEGER , *right_index* outINTEGER)
[Procedure]

Definition at line 8182 of file [fixed_pkg_c.vhdl](#).

6.1.2.254 Char2QuadBits(*C CHARACTER* , *RESULT* outSTD_ULOGIC_VECTOR(3 downto 0) , *GOOD*
outBOOLEAN , *ISSUE_ERROR* inBOOLEAN) [Procedure]

Definition at line 6905 of file [fixed_pkg_c.vhdl](#).

6.1.2.255 Char2TriBits(*C CHARACTER* , *RESULT* outSTD_ULOGIC_VECTOR(2 downto 0) , *GOOD*
outBOOLEAN , *ISSUE_ERROR* inBOOLEAN) [Procedure]

Definition at line 6876 of file [fixed_pkg_c.vhdl](#).

6.1.2.256 UNRESOLVED_sfixed cleanvec(*argin* UNRESOLVED_sfixed) [Function]

Definition at line 2049 of file [fixed_pkg_c.vhdl](#).

6.1.2.257 UNRESOLVED_ufixed cleanvec(*argin* UNRESOLVED_ufixed) [Function]

Definition at line 2064 of file [fixed_pkg_c.vhdl](#).

6.1.2.258 UNRESOLVED_ufixed divide(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed , *round_style* in
fixed_round_style_type *fixed_round_style* , *guard_bits* in NATURAL *fixed_guard_bits*)
[Function]

Definition at line 2526 of file [fixed_pkg_c.vhdl](#).

6.1.2.259 UNRESOLVED_sfixed divide(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed , *round_style* in
fixed_round_style_type *fixed_round_style* , *guard_bits* in NATURAL *fixed_guard_bits*)
[Function]

Definition at line 2567 of file [fixed_pkg_c.vhdl](#).

6.1.2.260 errmes(constant *mess* inSTRING) [Procedure]

Definition at line 7323 of file [fixed_pkg_c.vhdl](#).

6.1.2.261 errmes(constant *mess* inSTRING) [Procedure]

Definition at line 7580 of file [fixed_pkg_c.vhdl](#).

6.1.2.262 INTEGER find_leftmost(*argin* UNRESOLVED_ufixed , *yin* STD_ULOGIC) [Function]

Definition at line 2963 of file [fixed_pkg_c.vhdl](#).

6.1.2.263 INTEGER find_leftmost(*argin* UNRESOLVED_sfixed , *yin* STD_ULOGIC) [Function]

Definition at line 2985 of file [fixed_pkg_c.vhdl](#).

6.1.2.264 **INTEGER** `find_rightmost(argin UNRESOLVED_ufixed , yin STD_ULOGIC)` [Function]

Definition at line 2952 of file [fixed_pkg_c.vhdl](#).

6.1.2.265 **INTEGER** `find_rightmost(argin UNRESOLVED_sfixed , yin STD_ULOGIC)` [Function]

Definition at line 2974 of file [fixed_pkg_c.vhdl](#).

6.1.2.266 **UNRESOLVED_ufixed** `from_hstring(hstringin STRING , left_indexin INTEGER , right_indexin INTEGER)` [Function]

Definition at line 8060 of file [fixed_pkg_c.vhdl](#).

6.1.2.267 **UNRESOLVED_sfixed** `from_hstring(hstringin STRING , left_indexin INTEGER , right_indexin INTEGER)` [Function]

Definition at line 8114 of file [fixed_pkg_c.vhdl](#).

6.1.2.268 **UNRESOLVED_ufixed** `from_hstring(hstringin STRING , size_resin UNRESOLVED_ufixed)` [Function]

Definition at line 8149 of file [fixed_pkg_c.vhdl](#).

6.1.2.269 **UNRESOLVED_sfixed** `from_hstring(hstringin STRING , size_resin UNRESOLVED_sfixed)` [Function]

Definition at line 8173 of file [fixed_pkg_c.vhdl](#).

6.1.2.270 **UNRESOLVED_ufixed** `from_hstring(hstringin STRING)` [Function]

Definition at line 8254 of file [fixed_pkg_c.vhdl](#).

6.1.2.271 **UNRESOLVED_sfixed** `from_hstring(hstringin STRING)` [Function]

Definition at line 8281 of file [fixed_pkg_c.vhdl](#).

6.1.2.272 **UNRESOLVED_ufixed** `from_ostring(ostringin STRING , left_indexin INTEGER , right_indexin INTEGER)` [Function]

Definition at line 8042 of file [fixed_pkg_c.vhdl](#).

6.1.2.273 **UNRESOLVED_sfixed** `from_ostring(ostringin STRING , left_indexin INTEGER , right_indexin INTEGER)` [Function]

Definition at line 8096 of file [fixed_pkg_c.vhdl](#).

6.1.2.274 **UNRESOLVED_ufixed** `from_ostring(ostringin STRING , size_resin UNRESOLVED_ufixed)` [Function]

Definition at line 8141 of file [fixed_pkg_c.vhdl](#).

6.1.2.275 **UNRESOLVED_sfixed from_ostring(*ostring*in STRING , *size_res*in UNRESOLVED_sfixed)** [Function]

Definition at line 8165 of file [fixed_pkg_c.vhdl](#).

6.1.2.276 **UNRESOLVED_ufixed from_ostring(*ostring*in STRING)** [Function]

Definition at line 8245 of file [fixed_pkg_c.vhdl](#).

6.1.2.277 **UNRESOLVED_sfixed from_ostring(*ostring*in STRING)** [Function]

Definition at line 8272 of file [fixed_pkg_c.vhdl](#).

6.1.2.278 **UNRESOLVED_ufixed from_string(*bstring*in STRING , *left_index*in INTEGER , *right_index*in INTEGER)** [Function]

Definition at line 8021 of file [fixed_pkg_c.vhdl](#).

6.1.2.279 **UNRESOLVED_sfixed from_string(*bstring*in STRING , *left_index*in INTEGER , *right_index*in INTEGER)** [Function]

Definition at line 8078 of file [fixed_pkg_c.vhdl](#).

6.1.2.280 **UNRESOLVED_ufixed from_string(*bstring*in STRING , *size_res*in UNRESOLVED_ufixed)** [Function]

Definition at line 8133 of file [fixed_pkg_c.vhdl](#).

6.1.2.281 **UNRESOLVED_sfixed from_string(*bstring*in STRING , *size_res*in UNRESOLVED_sfixed)** [Function]

Definition at line 8157 of file [fixed_pkg_c.vhdl](#).

6.1.2.282 **UNRESOLVED_ufixed from_string(*bstring*in STRING)** [Function]

Definition at line 8232 of file [fixed_pkg_c.vhdl](#).

6.1.2.283 **UNRESOLVED_sfixed from_string(*bstring*in STRING)** [Function]

Definition at line 8263 of file [fixed_pkg_c.vhdl](#).

6.1.2.284 **HREAD(*L* inoutLINE , *VALUE* outUNRESOLVED_ufixed)** [Procedure]

Definition at line 7661 of file [fixed_pkg_c.vhdl](#).

6.1.2.285 **HREAD(*L* inoutLINE , *VALUE* outUNRESOLVED_ufixed , *GOOD* outBOOLEAN)** [Procedure]

Definition at line 7697 of file [fixed_pkg_c.vhdl](#).

6.1.2.286 HREAD(*L inoutLINE*, *VALUE outUNRESOLVED_sfixed*) [Procedure]

Definition at line 7726 of file [fixed_pkg_c.vhdl](#).

6.1.2.287 HREAD(*L inoutLINE*, *VALUE outUNRESOLVED_sfixed*, *GOOD outBOOLEAN*)
[Procedure]

Definition at line 7765 of file [fixed_pkg_c.vhdl](#).

6.1.2.288 HREAD_common(*L inoutLINE*, *slv outSTD_ULOGIC_VECTOR*, *igood outBOOLEAN*, *index outINTEGER*, constant *bpoint inINTEGER*, constant *message inBOOLEAN*, constant *smath inBOOLEAN*) [Procedure]

Definition at line 7570 of file [fixed_pkg_c.vhdl](#).

6.1.2.289 hwrite(*L inoutLINE*, *VALUE inUNRESOLVED_ufixed*, *JUSTIFIED inSIDERight*, *FIELD inWIDTH 0*) [Procedure]

Definition at line 7544 of file [fixed_pkg_c.vhdl](#).

6.1.2.290 hwrite(*L inoutLINE*, *VALUE inUNRESOLVED_sfixed*, *JUSTIFIED inSIDERight*, *FIELD inWIDTH 0*) [Procedure]

Definition at line 7557 of file [fixed_pkg_c.vhdl](#).

6.1.2.291 **BOOLEAN** Is_Negative(*argin UNRESOLVED_sfixed*) [Function]

Definition at line 2943 of file [fixed_pkg_c.vhdl](#).

6.1.2.292 **BOOLEAN** Is_X(*sin UNSIGNED*) [Function]

Definition at line 1756 of file [fixed_pkg_c.vhdl](#).

6.1.2.293 **BOOLEAN** Is_X(*sin SIGNED*) [Function]

Definition at line 1761 of file [fixed_pkg_c.vhdl](#).

6.1.2.294 **BOOLEAN** Is_X(*argin UNRESOLVED_ufixed*) [Function]

Definition at line 5135 of file [fixed_pkg_c.vhdl](#).

6.1.2.295 **BOOLEAN** Is_X(*argin UNRESOLVED_sfixed*) [Function]

Definition at line 5144 of file [fixed_pkg_c.vhdl](#).

6.1.2.296 **integer** maximum(*lin integer*, *rin integer*) [Function]

Definition at line 1479 of file [fixed_pkg_c.vhdl](#).

6.1.2.297 **UNRESOLVED_ufixed maximum(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed)** [Function]

Definition at line 4294 of file [fixed_pkg_c.vhdl](#).

6.1.2.298 **UNRESOLVED_sfixed maximum(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed)** [Function]

Definition at line 4309 of file [fixed_pkg_c.vhdl](#).

6.1.2.299 **UNRESOLVED_ufixed maximum(*lin* UNRESOLVED_ufixed , *rin* NATURAL)** [Function]

Definition at line 6051 of file [fixed_pkg_c.vhdl](#).

6.1.2.300 **UNRESOLVED_ufixed maximum(*lin* NATURAL , *rin* UNRESOLVED_ufixed)** [Function]

Definition at line 6164 of file [fixed_pkg_c.vhdl](#).

6.1.2.301 **UNRESOLVED_ufixed maximum(*lin* UNRESOLVED_ufixed , *rin* REAL)** [Function]

Definition at line 6277 of file [fixed_pkg_c.vhdl](#).

6.1.2.302 **UNRESOLVED_ufixed maximum(*lin* REAL , *rin* UNRESOLVED_ufixed)** [Function]

Definition at line 6390 of file [fixed_pkg_c.vhdl](#).

6.1.2.303 **UNRESOLVED_sfixed maximum(*lin* UNRESOLVED_sfixed , *rin* INTEGER)** [Function]

Definition at line 6503 of file [fixed_pkg_c.vhdl](#).

6.1.2.304 **UNRESOLVED_sfixed maximum(*lin* INTEGER , *rin* UNRESOLVED_sfixed)** [Function]

Definition at line 6616 of file [fixed_pkg_c.vhdl](#).

6.1.2.305 **UNRESOLVED_sfixed maximum(*lin* UNRESOLVED_sfixed , *rin* REAL)** [Function]

Definition at line 6729 of file [fixed_pkg_c.vhdl](#).

6.1.2.306 **UNRESOLVED_sfixed maximum(*lin* REAL , *rin* UNRESOLVED_sfixed)** [Function]

Definition at line 6842 of file [fixed_pkg_c.vhdl](#).

6.1.2.307 **INTEGER mine(*lin* INTEGER , *rin* INTEGER)** [Function]

Definition at line 2034 of file [fixed_pkg_c.vhdl](#).

6.1.2.308 **integer minimum(*lin* integer , *rin* integer)** [Function]

Definition at line 1488 of file [fixed_pkg_c.vhdl](#).

6.1.2.309 **UNRESOLVED_ufixed** `minimum(lin UNRESOLVED_ufixed , rin UNRESOLVED_ufixed)` [Function]

Definition at line [4324](#) of file [fixed_pkg_c.vhdl](#).

6.1.2.310 **UNRESOLVED_sfixed** `minimum(lin UNRESOLVED_sfixed , rin UNRESOLVED_sfixed)` [Function]

Definition at line [4339](#) of file [fixed_pkg_c.vhdl](#).

6.1.2.311 **UNRESOLVED_ufixed** `minimum(lin UNRESOLVED_ufixed , rin NATURAL)` [Function]

Definition at line [6059](#) of file [fixed_pkg_c.vhdl](#).

6.1.2.312 **UNRESOLVED_ufixed** `minimum(lin NATURAL , rin UNRESOLVED_ufixed)` [Function]

Definition at line [6172](#) of file [fixed_pkg_c.vhdl](#).

6.1.2.313 **UNRESOLVED_ufixed** `minimum(lin UNRESOLVED_ufixed , rin REAL)` [Function]

Definition at line [6285](#) of file [fixed_pkg_c.vhdl](#).

6.1.2.314 **UNRESOLVED_ufixed** `minimum(lin REAL , rin UNRESOLVED_ufixed)` [Function]

Definition at line [6398](#) of file [fixed_pkg_c.vhdl](#).

6.1.2.315 **UNRESOLVED_sfixed** `minimum(lin UNRESOLVED_sfixed , rin INTEGER)` [Function]

Definition at line [6511](#) of file [fixed_pkg_c.vhdl](#).

6.1.2.316 **UNRESOLVED_sfixed** `minimum(lin INTEGER , rin UNRESOLVED_sfixed)` [Function]

Definition at line [6624](#) of file [fixed_pkg_c.vhdl](#).

6.1.2.317 **UNRESOLVED_sfixed** `minimum(lin UNRESOLVED_sfixed , rin REAL)` [Function]

Definition at line [6737](#) of file [fixed_pkg_c.vhdl](#).

6.1.2.318 **UNRESOLVED_sfixed** `minimum(lin REAL , rin UNRESOLVED_sfixed)` [Function]

Definition at line [6850](#) of file [fixed_pkg_c.vhdl](#).

6.1.2.319 **INTEGER** `mins(lin INTEGER , rin INTEGER)` [Function]

Definition at line [2024](#) of file [fixed_pkg_c.vhdl](#).

6.1.2.320 **UNRESOLVED_ufixed modulo(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed , *round_style*in fixed_round_style_typefixed_round_style , *guard_bits*in NATURALfixed_guard_bits)** [Function]

Definition at line 2769 of file [fixed_pkg_c.vhdl](#).

6.1.2.321 **UNRESOLVED_sfixed modulo(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed , *overflow_style*in fixed_overflow_style_typefixed_overflow_style , *round_style*in fixed_round_style_typefixed_round_style , *guard_bits*in NATURALfixed_guard_bits)** [Function]

Definition at line 2783 of file [fixed_pkg_c.vhdl](#).

6.1.2.322 **STD_ULOGIC nand_reduce(*argin* std_ulogic_vector)** [Function]

Definition at line 1581 of file [fixed_pkg_c.vhdl](#).

6.1.2.323 **STD_ULOGIC nand_reduce(*lin* UNRESOLVED_ufixed)** [Function]

Definition at line 3621 of file [fixed_pkg_c.vhdl](#).

6.1.2.324 **STD_ULOGIC nand_reduce(*lin* UNRESOLVED_sfixed)** [Function]

Definition at line 3651 of file [fixed_pkg_c.vhdl](#).

6.1.2.325 **STD_ULOGIC nor_reduce(*argin* std_ulogic_vector)** [Function]

Definition at line 1585 of file [fixed_pkg_c.vhdl](#).

6.1.2.326 **STD_ULOGIC nor_reduce(*lin* UNRESOLVED_ufixed)** [Function]

Definition at line 3631 of file [fixed_pkg_c.vhdl](#).

6.1.2.327 **STD_ULOGIC nor_reduce(*lin* UNRESOLVED_sfixed)** [Function]

Definition at line 3661 of file [fixed_pkg_c.vhdl](#).

6.1.2.328 **STD_LOGIC or_reduce(*argin* STD_ULOGIC_VECTOR)** [Function]

Definition at line 1507 of file [fixed_pkg_c.vhdl](#).

6.1.2.329 **STD_ULOGIC or_reduce(*lin* UNRESOLVED_ufixed)** [Function]

Definition at line 3626 of file [fixed_pkg_c.vhdl](#).

6.1.2.330 **STD_ULOGIC or_reduce(*lin* UNRESOLVED_sfixed)** [Function]

Definition at line 3656 of file [fixed_pkg_c.vhdl](#).

6.1.2.331 OREAD(*L inoutLINE* , *VALUE outUNRESOLVED_ufixed*) [Procedure]

Definition at line 7407 of file [fixed_pkg_c.vhdl](#).

6.1.2.332 OREAD(*L inoutLINE* , *VALUE outUNRESOLVED_ufixed* , *GOOD outBOOLEAN*)
[Procedure]

Definition at line 7443 of file [fixed_pkg_c.vhdl](#).

6.1.2.333 OREAD(*L inoutLINE* , *VALUE outUNRESOLVED_sfixed*) [Procedure]

Definition at line 7472 of file [fixed_pkg_c.vhdl](#).

6.1.2.334 OREAD(*L inoutLINE* , *VALUE outUNRESOLVED_sfixed* , *GOOD outBOOLEAN*)
[Procedure]

Definition at line 7511 of file [fixed_pkg_c.vhdl](#).

6.1.2.335 OREAD_common(*L inoutLINE* , *slv outSTD_ULOGIC_VECTOR* , *igood outBOOLEAN* , *index outINTEGER* , constant *bpoint inINTEGER* , constant *message inBOOLEAN* , constant *smax inBOOLEAN*) [Procedure]

Definition at line 7313 of file [fixed_pkg_c.vhdl](#).

6.1.2.336 owrite(*L inoutLINE* , *VALUE inUNRESOLVED_ufixed* , *JUSTIFIED inSIDEright* , *FIELD inWIDTH 0*) [Procedure]

Definition at line 7288 of file [fixed_pkg_c.vhdl](#).

6.1.2.337 owrite(*L inoutLINE* , *VALUE inUNRESOLVED_sfixed* , *JUSTIFIED inSIDEright* , *FIELD inWIDTH 0*) [Procedure]

Definition at line 7300 of file [fixed_pkg_c.vhdl](#).

6.1.2.338 READ(*L inoutLINE* , *VALUE outUNRESOLVED_ufixed*) [Procedure]

Definition at line 7078 of file [fixed_pkg_c.vhdl](#).

6.1.2.339 READ(*L inoutLINE* , *VALUE outUNRESOLVED_ufixed* , *GOOD outBOOLEAN*)
[Procedure]

Definition at line 7151 of file [fixed_pkg_c.vhdl](#).

6.1.2.340 READ(*L inoutLINE* , *VALUE outUNRESOLVED_sfixed*) [Procedure]

Definition at line 7207 of file [fixed_pkg_c.vhdl](#).

6.1.2.341 READ(*L inoutLINE* , *VALUE outUNRESOLVED_sfixed* , *GOOD outBOOLEAN*)
[Procedure]

Definition at line 7278 of file [fixed_pkg_c.vhdl](#).

6.1.2.342 **UNRESOLVED_ufixed reciprocal(*argin* UNRESOLVED_ufixed , *round_style*in
fixed_round_style_typefixed_round_style , *guard_bits*in NATURALfixed_guard_bits)**
[Function]

Definition at line 2608 of file [fixed_pkg_c.vhdl](#).

6.1.2.343 **UNRESOLVED_sfixed reciprocal(*argin* UNRESOLVED_sfixed , *round_style*in
fixed_round_style_typefixed_round_style , *guard_bits*in NATURALfixed_guard_bits)**
[Function]

Definition at line 2622 of file [fixed_pkg_c.vhdl](#).

6.1.2.344 **UNRESOLVED_ufixed remainder(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed ,
*round_style*in fixed_round_style_typefixed_round_style , *guard_bits*in NATURALfixed_guard_bits)**
[Function]

Definition at line 2662 of file [fixed_pkg_c.vhdl](#).

6.1.2.345 **UNRESOLVED_sfixed remainder(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed ,
*round_style*in fixed_round_style_typefixed_round_style , *guard_bits*in NATURALfixed_guard_bits)**
[Function]

Definition at line 2718 of file [fixed_pkg_c.vhdl](#).

6.1.2.346 **UNRESOLVED_ufixed resize(*argin* UNRESOLVED_ufixed , *left_index*in INTEGER , *right_index*in
INTEGER , *overflow_style*in fixed_overflow_style_typefixed_overflow_style , *round_style*in
fixed_round_style_typefixed_round_style)** [Function]

Definition at line 5195 of file [fixed_pkg_c.vhdl](#).

6.1.2.347 **UNRESOLVED_sfixed resize(*argin* UNRESOLVED_sfixed , *left_index*in INTEGER , *right_index*in
INTEGER , *overflow_style*in fixed_overflow_style_typefixed_overflow_style , *round_style*in
fixed_round_style_typefixed_round_style)** [Function]

Definition at line 5259 of file [fixed_pkg_c.vhdl](#).

6.1.2.348 **UNRESOLVED_ufixed resize(*argin* UNRESOLVED_ufixed , *size_resin* UNRESOLVED_ufixed
, *overflow_style*in fixed_overflow_style_typefixed_overflow_style , *round_style*in
fixed_round_style_typefixed_round_style)** [Function]

Definition at line 5523 of file [fixed_pkg_c.vhdl](#).

6.1.2.349 **UNRESOLVED_sfixed resize(*argin* UNRESOLVED_sfixed , *size_resin* UNRESOLVED_sfixed
, *overflow_style*in fixed_overflow_style_typefixed_overflow_style , *round_style*in
fixed_round_style_typefixed_round_style)** [Function]

Definition at line 5544 of file [fixed_pkg_c.vhdl](#).

6.1.2.350 **UNRESOLVED_ufixed round_fixed(*argin* UNRESOLVED_ufixed , *remainderin* UNRESOLVED_ufixed , *overflow_style*in fixed_overflow_style_typefixed_overflow_style)** [Function]

Definition at line 2156 of file [fixed_pkg_c.vhdl](#).

6.1.2.351 **UNRESOLVED_sfixed round_fixed(*argin* UNRESOLVED_sfixed , *remainderin* UNRESOLVED_sfixed , *overflow_style*in fixed_overflow_style_typefixed_overflow_style)** [Function]

Definition at line 2188 of file [fixed_pkg_c.vhdl](#).

6.1.2.352 **round_up(*arg* inUNRESOLVED_ufixed , *result* outUNRESOLVED_ufixed , *overflowx* outBOOLEAN)** [Procedure]

Definition at line 2125 of file [fixed_pkg_c.vhdl](#).

6.1.2.353 **round_up(*arg* inUNRESOLVED_sfixed , *result* outUNRESOLVED_sfixed , *overflowx* outBOOLEAN)** [Procedure]

Definition at line 2139 of file [fixed_pkg_c.vhdl](#).

6.1.2.354 **UNRESOLVED_ufixed saturate(*left_index*in INTEGER , *right_index*in INTEGER)** [Function]

Definition at line 4784 of file [fixed_pkg_c.vhdl](#).

6.1.2.355 **UNRESOLVED_sfixed saturate(*left_index*in INTEGER , *right_index*in INTEGER)** [Function]

Definition at line 4795 of file [fixed_pkg_c.vhdl](#).

6.1.2.356 **UNRESOLVED_ufixed saturate(*size_res*in UNRESOLVED_ufixed)** [Function]

Definition at line 4807 of file [fixed_pkg_c.vhdl](#).

6.1.2.357 **UNRESOLVED_sfixed saturate(*size_res*in UNRESOLVED_sfixed)** [Function]

Definition at line 4814 of file [fixed_pkg_c.vhdl](#).

6.1.2.358 **UNRESOLVED_ufixed scalb(*yin* UNRESOLVED_ufixed , *Nin*INTEGER)** [Function]

Definition at line 2905 of file [fixed_pkg_c.vhdl](#).

6.1.2.359 **UNRESOLVED_ufixed scalb(*yin* UNRESOLVED_ufixed , *Nin*SIGNED)** [Function]

Definition at line 2917 of file [fixed_pkg_c.vhdl](#).

6.1.2.360 **UNRESOLVED_sfixed scalb(*yin* UNRESOLVED_sfixed , *Nin*INTEGER)** [Function]

Definition at line 2924 of file [fixed_pkg_c.vhdl](#).

6.1.2.361 **UNRESOLVED_sfixed scalb(*yin* UNRESOLVED_sfixed , *Nin* SIGNED)** [Function]

Definition at line 2936 of file [fixed_pkg_c.vhdl](#).

6.1.2.362 **INTEGER sfix_high(*widthin* NATURAL , *fractionin* NATURAL , *operationin* CHARACTER'X' ,
width2in NATURAL 0 , *fraction2in* NATURAL 0)** [Function]

Definition at line 4905 of file [fixed_pkg_c.vhdl](#).

6.1.2.363 **INTEGER sfix_low(*widthin* NATURAL , *fractionin* NATURAL , *operationin* CHARACTER'X' ,
width2in NATURAL 0 , *fraction2in* NATURAL 0)** [Function]

Definition at line 4918 of file [fixed_pkg_c.vhdl](#).

6.1.2.364 **INTEGER sfixed_high(*left_indexin* INTEGER , *right_indexin* INTEGER , *operationin* CHARACTER'X' ,
left_index2in INTEGER 0 , *right_index2in* INTEGER 0)** [Function]

Definition at line 4697 of file [fixed_pkg_c.vhdl](#).

6.1.2.365 **INTEGER sfixed_high(*size_resin* UNRESOLVED_sfixed , *operationin* CHARACTER'X' , *size_res2in*
UNRESOLVED_sfixed)** [Function]

Definition at line 4759 of file [fixed_pkg_c.vhdl](#).

6.1.2.366 **INTEGER sfixed_low(*left_indexin* INTEGER , *right_indexin* INTEGER , *operationin* CHARACTER'X' ,
left_index2in INTEGER 0 , *right_index2in* INTEGER 0)** [Function]

Definition at line 4715 of file [fixed_pkg_c.vhdl](#).

6.1.2.367 **INTEGER sfixed_low(*size_resin* UNRESOLVED_sfixed , *operationin* CHARACTER'X' , *size_res2in*
UNRESOLVED_sfixed)** [Function]

Definition at line 4771 of file [fixed_pkg_c.vhdl](#).

6.1.2.368 **UNRESOLVED_ufixed SHIFT_LEFT(*ARGin* UNRESOLVED_ufixed , *COUNTin* NATURAL)**
[Function]

Definition at line 3141 of file [fixed_pkg_c.vhdl](#).

6.1.2.369 **UNRESOLVED_sfixed SHIFT_LEFT(*ARGin* UNRESOLVED_sfixed , *COUNTin* NATURAL)**
[Function]

Definition at line 3159 of file [fixed_pkg_c.vhdl](#).

6.1.2.370 **UNRESOLVED_ufixed SHIFT_RIGHT(*ARGin* UNRESOLVED_ufixed , *COUNTin* NATURAL)**
[Function]

Definition at line 3150 of file [fixed_pkg_c.vhdl](#).

6.1.2.371 **UNRESOLVED_sfixed SHIFT_RIGHT(*ARGin UNRESOLVED_sfixed , COUNTin NATURAL*)**
 [Function]

Definition at line 3168 of file [fixed_pkg_c.vhdl](#).

6.1.2.372 **skip_whitespace(*L inoutLINE*)** [Procedure]

Definition at line 6941 of file [fixed_pkg_c.vhdl](#).

6.1.2.373 **BOOLEAN std_match(*Lin UNRESOLVED_ufixed , Rin UNRESOLVED_ufixed*)** [Function]

Definition at line 3930 of file [fixed_pkg_c.vhdl](#).

6.1.2.374 **BOOLEAN std_match(*Lin UNRESOLVED_sfixed , Rin UNRESOLVED_sfixed*)** [Function]

Definition at line 3943 of file [fixed_pkg_c.vhdl](#).

6.1.2.375 **UNRESOLVED_ufixed to_01(*sin UNRESOLVED_ufixed , XMAPin STD_ULOGIC' 0 '*)**
 [Function]

Definition at line 5103 of file [fixed_pkg_c.vhdl](#).

6.1.2.376 **UNRESOLVED_sfixed to_01(*sin UNRESOLVED_sfixed , XMAPin STD_ULOGIC' 0 '*)**
 [Function]

Definition at line 5119 of file [fixed_pkg_c.vhdl](#).

6.1.2.377 **UNRESOLVED_ufixed to_fixed(*argin UNSIGNED , left_indexin INTEGER , right_indexin INTEGER*)** [Function]

Definition at line 2079 of file [fixed_pkg_c.vhdl](#).

6.1.2.378 **UNRESOLVED_sfixed to_fixed(*argin SIGNED , left_indexin INTEGER , right_indexin INTEGER*)**
 [Function]

Definition at line 2091 of file [fixed_pkg_c.vhdl](#).

6.1.2.379 **STRING to_hstring(*valuein STD_ULOGIC_VECTOR*)** [Function]

Definition at line 6990 of file [fixed_pkg_c.vhdl](#).

6.1.2.380 **STRING to_hstring(*valuein UNRESOLVED_ufixed*)** [Function]

Definition at line 7904 of file [fixed_pkg_c.vhdl](#).

6.1.2.381 **STRING to_hstring(*valuein UNRESOLVED_sfixed*)** [Function]

Definition at line 7977 of file [fixed_pkg_c.vhdl](#).

6.1.2.382 **NATURAL** to_integer(*argin UNRESOLVED_ufixed*, *overflow_style*in **fixed_overflow←
style_typefixed_overflow_style**, *round_style*in **fixed_round_style_typefixed_round_style**)
[Function]

Definition at line 5043 of file [fixed_pkg_c.vhdl](#).

6.1.2.383 **INTEGER** to_integer(*argin UNRESOLVED_sfixed*, *overflow_style*in **fixed_overflow_style←
typefixed_overflow_style**, *round_style*in **fixed_round_style_typefixed_round_style**)
[Function]

Definition at line 5073 of file [fixed_pkg_c.vhdl](#).

6.1.2.384 **STRING** to_ostring(*value*in **STD_ULOGIC_VECTOR**) [Function]

Definition at line 6955 of file [fixed_pkg_c.vhdl](#).

6.1.2.385 **STRING** to_ostring(*value*in **UNRESOLVED_ufixed**) [Function]

Definition at line 7867 of file [fixed_pkg_c.vhdl](#).

6.1.2.386 **STRING** to_ostring(*value*in **UNRESOLVED_sfixed**) [Function]

Definition at line 7941 of file [fixed_pkg_c.vhdl](#).

6.1.2.387 **REAL** to_real(*argin UNRESOLVED_ufixed*) [Function]

Definition at line 4985 of file [fixed_pkg_c.vhdl](#).

6.1.2.388 **REAL** to_real(*argin UNRESOLVED_sfixed*) [Function]

Definition at line 5013 of file [fixed_pkg_c.vhdl](#).

6.1.2.389 **SIGNED** to_s(*argin UNRESOLVED_sfixed*) [Function]

Definition at line 2114 of file [fixed_pkg_c.vhdl](#).

6.1.2.390 **UNRESOLVED_sfixed** to_SFix(*argin STD_ULOGIC_VECTOR*, *width*in **NATURAL**, *fraction*in
NATURAL) [Function]

Definition at line 4850 of file [fixed_pkg_c.vhdl](#).

6.1.2.391 **UNRESOLVED_sfixed** to_SFix(*argin STD_LOGIC_VECTOR*, *width*in **NATURAL**, *fraction*in
NATURAL) [Function]

Definition at line 8351 of file [fixed_pkg_c.vhdl](#).

6.1.2.392 **unresolved_sfixed** to_sfixed(*argin STD_ULOGIC_VECTOR*, *left_index*in **INTEGER**, *right_index*in
INTEGER) [Function]

Definition at line 2314 of file [fixed_pkg_c.vhdl](#).

6.1.2.393 **UNRESOLVED_sfixed to_sfixed(*argin* INTEGER , *left_index*in INTEGER , *right_index*in INTEGER 0 , *overflow_style*in fixed_overflow_style_typefixed_overflow_style , *round_style*in fixed_round_style_typefixed_round_style)** [Function]

Definition at line 4400 of file [fixed_pkg_c.vhdl](#).

6.1.2.394 **UNRESOLVED_sfixed to_sfixed(*argin* REAL , *left_index*in INTEGER , *right_index*in INTEGER , *overflow_style*in fixed_overflow_style_typefixed_overflow_style , *round_style*in fixed_round_style_typefixed_round_style , *guard_bits*in NATURALfixed_guard_bits)** [Function]

Definition at line 4516 of file [fixed_pkg_c.vhdl](#).

6.1.2.395 **UNRESOLVED_sfixed to_sfixed(*argin* SIGNED , *left_index*in INTEGER , *right_index*in INTEGER 0 , *overflow_style*in fixed_overflow_style_typefixed_overflow_style , *round_style*in fixed_round_style_typefixed_round_style)** [Function]

Definition at line 4609 of file [fixed_pkg_c.vhdl](#).

6.1.2.396 **UNRESOLVED_sfixed to_sfixed(*argin* SIGNED)** [Function]

Definition at line 4632 of file [fixed_pkg_c.vhdl](#).

6.1.2.397 **UNRESOLVED_sfixed to_sfixed(*argin* UNRESOLVED_ufixed)** [Function]

Definition at line 4644 of file [fixed_pkg_c.vhdl](#).

6.1.2.398 **UNRESOLVED_sfixed to_sfixed(*argin* STD_ULONG_VECTOR , *size_resin* UNRESOLVED_sfixed)** [Function]

Definition at line 5376 of file [fixed_pkg_c.vhdl](#).

6.1.2.399 **UNRESOLVED_sfixed to_sfixed(*argin* INTEGER , *size_resin* UNRESOLVED_sfixed , *overflow_style*in fixed_overflow_style_typefixed_overflow_style , *round_style*in fixed_round_style_typefixed_round_style)** [Function]

Definition at line 5414 of file [fixed_pkg_c.vhdl](#).

6.1.2.400 **UNRESOLVED_sfixed to_sfixed(*argin* REAL , *size_resin* UNRESOLVED_sfixed , *overflow_style*in fixed_overflow_style_typefixed_overflow_style , *round_style*in fixed_round_style_typefixed_round_style , *guard_bits*in NATURALfixed_guard_bits)** [Function]

Definition at line 5458 of file [fixed_pkg_c.vhdl](#).

6.1.2.401 **UNRESOLVED_sfixed to_sfixed(*argin* SIGNED , *size_resin* UNRESOLVED_sfixed , *overflow_style*in fixed_overflow_style_typefixed_overflow_style , *round_style*in fixed_round_style_typefixed_round_style)** [Function]

Definition at line 5502 of file [fixed_pkg_c.vhdl](#).

6.1.2.402 **UNRESOLVED_sfixed** `to_sfixed(argin STD_LOGIC_VECTOR , left_indexin INTEGER , right_indexin INTEGER)` [Function]

Definition at line 8315 of file [fixed_pkg_c.vhdl](#).

6.1.2.403 **UNRESOLVED_sfixed** `to_sfixed(argin STD_LOGIC_VECTOR , size_resin UNRESOLVED_sfixed)` [Function]

Definition at line 8327 of file [fixed_pkg_c.vhdl](#).

6.1.2.404 **SIGNED** `to_signed(argin UNRESOLVED_sfixed , sizein NATURAL , overflow_stylein fixed_overflow_style_typefixed_overflow_style , round_stylein fixed_round_style_typefixed_round_style)` [Function]

Definition at line 4958 of file [fixed_pkg_c.vhdl](#).

6.1.2.405 **SIGNED** `to_signed(argin UNRESOLVED_sfixed , size_resin SIGNED , overflow_stylein fixed_overflow_style_typefixed_overflow_style , round_stylein fixed_round_style_typefixed_round_style)` [Function]

Definition at line 4972 of file [fixed_pkg_c.vhdl](#).

6.1.2.406 **STD_LOGIC_VECTOR** `to_slv(argin UNRESOLVED_ufixed)` [Function]

Definition at line 2275 of file [fixed_pkg_c.vhdl](#).

6.1.2.407 **STD_LOGIC_VECTOR** `to_slv(argin UNRESOLVED_sfixed)` [Function]

Definition at line 2282 of file [fixed_pkg_c.vhdl](#).

6.1.2.408 **STRING** `to_string(valuein UNRESOLVED_ufixed)` [Function]

Definition at line 7797 of file [fixed_pkg_c.vhdl](#).

6.1.2.409 **STRING** `to_string(valuein UNRESOLVED_sfixed)` [Function]

Definition at line 7834 of file [fixed_pkg_c.vhdl](#).

6.1.2.410 **STD_ULOGIC_VECTOR** `to_sluv(argin UNRESOLVED_ufixed)` [Function]

Definition at line 2251 of file [fixed_pkg_c.vhdl](#).

6.1.2.411 **STD_ULOGIC_VECTOR** `to_sluv(argin UNRESOLVED_sfixed)` [Function]

Definition at line 2263 of file [fixed_pkg_c.vhdl](#).

6.1.2.412 **UNRESOLVED_ufixed** `to_UFix(argin STD_ULOGIC_VECTOR , widthin NATURAL , fractionin NATURAL)` [Function]

Definition at line 4828 of file [fixed_pkg_c.vhdl](#).

6.1.2.413 **UNRESOLVED_ufixed to_UFix(*argin STD_LOGIC_VECTOR*, *widthin NATURAL*, *fractionin NATURAL*)** [Function]

Definition at line 8338 of file [fixed_pkg_c.vhdl](#).

6.1.2.414 **UNRESOLVED_ufixed to_ufixed(*argin UNRESOLVED_sfixed*)** [Function]

Definition at line 2228 of file [fixed_pkg_c.vhdl](#).

6.1.2.415 **unresolved_ufixed to_ufixed(*argin STD_ULOGIC_VECTOR*, *left_indexin INTEGER*, *right_indexin INTEGER*)** [Function]

Definition at line 2289 of file [fixed_pkg_c.vhdl](#).

6.1.2.416 **UNRESOLVED_ufixed to_ufixed(*argin NATURAL*, *left_indexin INTEGER*, *right_indexin INTEGER* *0*, *overflow_stylein fixed_overflow_style_typefixed_overflow_style*, *round_stylein fixed_round_style_typefixed_round_style*)** [Function]

Definition at line 4354 of file [fixed_pkg_c.vhdl](#).

6.1.2.417 **UNRESOLVED_ufixed to_ufixed(*argin REAL*, *left_indexin INTEGER*, *right_indexin INTEGER* *0*, *overflow_stylein fixed_overflow_style_typefixed_overflow_style*, *round_stylein fixed_round_style_typefixed_round_style*, *guard_bitsin NATURALfixed_guard_bits*)** [Function]

Definition at line 4458 of file [fixed_pkg_c.vhdl](#).

6.1.2.418 **UNRESOLVED_ufixed to_ufixed(*argin UNSIGNED*, *left_indexin INTEGER*, *right_indexin INTEGER* *0*, *overflow_stylein fixed_overflow_style_typefixed_overflow_style*, *round_stylein fixed_round_style_typefixed_round_style*)** [Function]

Definition at line 4574 of file [fixed_pkg_c.vhdl](#).

6.1.2.419 **UNRESOLVED_ufixed to_ufixed(*argin UNSIGNED*)** [Function]

Definition at line 4597 of file [fixed_pkg_c.vhdl](#).

6.1.2.420 **UNRESOLVED_ufixed to_ufixed(*argin STD_ULOGIC_VECTOR*, *size_resin UNRESOLVED_ufixed*)** [Function]

Definition at line 5359 of file [fixed_pkg_c.vhdl](#).

6.1.2.421 **UNRESOLVED_ufixed to_ufixed(*argin NATURAL*, *size_resin UNRESOLVED_ufixed*, *overflow_stylein fixed_overflow_style_typefixed_overflow_style*, *round_stylein fixed_round_style_typefixed_round_style*)** [Function]

Definition at line 5393 of file [fixed_pkg_c.vhdl](#).

6.1.2.422 **UNRESOLVED_ufixed to_ufixed(*argin REAL*, *size_resin UNRESOLVED_ufixed*, *overflow_stylein fixed_overflow_style_typefixed_overflow_style*, *round_stylein fixed_round_style_typefixed_round_style*, *guard_bitsin NATURALfixed_guard_bits*)** [Function]

Definition at line 5435 of file [fixed_pkg_c.vhdl](#).

6.1.2.423 **UNRESOLVED_ufixed to_ufixed(*argin UNSIGNED*, *size_resin UNRESOLVED_ufixed*, *overflow_stylein fixed_overflow_style_typefixed_overflow_style*, *round_stylein fixed_round_style_typefixed_round_style*)** [Function]

Definition at line 5481 of file [fixed_pkg_c.vhdl](#).

6.1.2.424 **UNRESOLVED_ufixed to_ufixed(*argin STD_LOGIC_VECTOR*, *left_indexin INTEGER*, *right_indexin INTEGER*)** [Function]

Definition at line 8293 of file [fixed_pkg_c.vhdl](#).

6.1.2.425 **UNRESOLVED_ufixed to_ufixed(*argin STD_LOGIC_VECTOR*, *size_resin UNRESOLVED_ufixed*)** [Function]

Definition at line 8305 of file [fixed_pkg_c.vhdl](#).

6.1.2.426 **UNSIGNED to_uns(*argin UNRESOLVED_ufixed*)** [Function]

Definition at line 2103 of file [fixed_pkg_c.vhdl](#).

6.1.2.427 **UNSIGNED to_unsigned(*argin UNRESOLVED_ufixed*, *sizein NATURAL*, *overflow_stylein fixed_overflow_style_typefixed_overflow_style*, *round_stylein fixed_round_style_typefixed_round_style*)** [Function]

Definition at line 4931 of file [fixed_pkg_c.vhdl](#).

6.1.2.428 **UNSIGNED to_unsigned(*argin UNRESOLVED_ufixed*, *size_resin UNSIGNED*, *overflow_stylein fixed_overflow_style_typefixed_overflow_style*, *round_stylein fixed_round_style_typefixed_round_style*)** [Function]

Definition at line 4945 of file [fixed_pkg_c.vhdl](#).

6.1.2.429 **UNRESOLVED_ufixed To_UX01(*argin UNRESOLVED_ufixed*)** [Function]

Definition at line 5181 of file [fixed_pkg_c.vhdl](#).

6.1.2.430 **UNRESOLVED_sfixed to_UX01(*argin UNRESOLVED_sfixed*)** [Function]

Definition at line 5188 of file [fixed_pkg_c.vhdl](#).

6.1.2.431 **UNRESOLVED_ufixed To_X01(*argin UNRESOLVED_ufixed*)** [Function]

Definition at line 5153 of file [fixed_pkg_c.vhdl](#).

6.1.2.432 **UNRESOLVED_sfixed to_X01(*argin* UNRESOLVED_sfixed)** [Function]

Definition at line 5160 of file [fixed_pkg_c.vhdl](#).

6.1.2.433 **UNRESOLVED_ufixed To_X01Z(*argin* UNRESOLVED_ufixed)** [Function]

Definition at line 5167 of file [fixed_pkg_c.vhdl](#).

6.1.2.434 **UNRESOLVED_sfixed to_X01Z(*argin* UNRESOLVED_sfixed)** [Function]

Definition at line 5174 of file [fixed_pkg_c.vhdl](#).

6.1.2.435 **INTEGER ufix_high(*width*in NATURAL , *fraction*in NATURAL , *operation*in CHARACTER'X' , *width2*in NATURAL 0 , *fraction2*in NATURAL 0)** [Function]

Definition at line 4879 of file [fixed_pkg_c.vhdl](#).

6.1.2.436 **INTEGER ufix_low(*width*in NATURAL , *fraction*in NATURAL , *operation*in CHARACTER'X' , *width2*in NATURAL 0 , *fraction2*in NATURAL 0)** [Function]

Definition at line 4892 of file [fixed_pkg_c.vhdl](#).

6.1.2.437 **INTEGER ufixed_high(*left_index*in INTEGER , *right_index*in INTEGER , *operation*in CHARACTER'X' , *left_index2*in INTEGER 0 , *right_index2*in INTEGER 0)** [Function]

Definition at line 4665 of file [fixed_pkg_c.vhdl](#).

6.1.2.438 **INTEGER ufixed_high(*size_res*in UNRESOLVED_ufixed , *operation*in CHARACTER'X' , *size_res2*in UNRESOLVED_ufixed)** [Function]

Definition at line 4735 of file [fixed_pkg_c.vhdl](#).

6.1.2.439 **INTEGER ufixed_low(*left_index*in INTEGER , *right_index*in INTEGER , *operation*in CHARACTER'X' , *left_index2*in INTEGER 0 , *right_index2*in INTEGER 0)** [Function]

Definition at line 4681 of file [fixed_pkg_c.vhdl](#).

6.1.2.440 **INTEGER ufixed_low(*size_res*in UNRESOLVED_ufixed , *operation*in CHARACTER'X' , *size_res2*in UNRESOLVED_ufixed)** [Function]

Definition at line 4747 of file [fixed_pkg_c.vhdl](#).

6.1.2.441 **write(*L*inoutLINE , *VALUE*inUNRESOLVED_ufixed , *JUSTIFIED*inSIDERight , *FIELD*inWIDTH 0)** [Procedure]

Definition at line 7037 of file [fixed_pkg_c.vhdl](#).

6.1.2.442 **write(*L*inoutLINE , *VALUE*inUNRESOLVED_sfixed , *JUSTIFIED*inSIDERight , *FIELD*inWIDTH 0)** [Procedure]

Definition at line 7058 of file [fixed_pkg_c.vhdl](#).

6.1.2.443 **STD_ULOGIC** xnor_reduce(*argin std_ulogic_vector*) [Function]

Definition at line 1589 of file [fixed_pkg_c.vhdl](#).

6.1.2.444 **STD_ULOGIC** xnor_reduce(*lin UNRESOLVED_ufixed*) [Function]

Definition at line 3641 of file [fixed_pkg_c.vhdl](#).

6.1.2.445 **STD_ULOGIC** xnor_reduce(*lin UNRESOLVED_sfixed*) [Function]

Definition at line 3671 of file [fixed_pkg_c.vhdl](#).

6.1.2.446 **STD_ULOGIC** xor_reduce(*argin STD_ULOGIC_VECTOR*) [Function]

Definition at line 1559 of file [fixed_pkg_c.vhdl](#).

6.1.2.447 **STD_ULOGIC** xor_reduce(*lin UNRESOLVED_ufixed*) [Function]

Definition at line 3636 of file [fixed_pkg_c.vhdl](#).

6.1.2.448 **STD_ULOGIC** xor_reduce(*lin UNRESOLVED_sfixed*) [Function]

Definition at line 3666 of file [fixed_pkg_c.vhdl](#).

6.1.3 Member Data Documentation

6.1.3.1 **c** **CHARACTER** [Shared Variable]

Definition at line 7342 of file [fixed_pkg_c.vhdl](#).

6.1.3.2 **char_indexed_by_MVL9 (STD_ULOGIC)CHARACTER** [Type]

Definition at line 6861 of file [fixed_pkg_c.vhdl](#).

6.1.3.3 **char_to_MVL9 MVL9_indexed_by_char :=('U'=>'U','X'=>'X','0'=>'0','1'=>'1','Z'=>'Z','W'=>'W','L'=>'L','H'=>'H','-'=>'-',others=>'U')** [Constant]

Definition at line 6866 of file [fixed_pkg_c.vhdl](#).

6.1.3.4 **char_to_MVL9plus MVL9plus_indexed_by_char :=('U'=>'U','X'=>'X','0'=>'0','1'=>'1','Z'=>'Z','W'=>'W','L'=>'L','H'=>'H','-'=>'-',others=>error)** [Constant]

Definition at line 6869 of file [fixed_pkg_c.vhdl](#).

6.1.3.5 **fixedsynth_or_real BOOLEAN:=true** [Constant]

Definition at line 1476 of file [fixed_pkg_c.vhdl](#).

6.1.3.6 **founddot** **BOOLEAN****::=****false** [Shared Variable]

Definition at line 7345 of file [fixed_pkg_c.vhdl](#).

6.1.3.7 **i** **INTEGER** [Shared Variable]

Definition at line 7343 of file [fixed_pkg_c.vhdl](#).

6.1.3.8 **IEEE** [Library]

Definition at line 1463 of file [fixed_pkg_c.vhdl](#).

6.1.3.9 **lastu** **BOOLEAN****::=****false** [Shared Variable]

Definition at line 7344 of file [fixed_pkg_c.vhdl](#).

6.1.3.10 **match_logic_table** **stdlogic_table** **::=**(('U','U','U','U','U','U','U','U','1'), ('U','X','X','X','X','X','X','X','1'), ('U','X','1','0','X','X','1','0','1'), ('U','X','0','1','X','X','0','1','1'), ('U','X','X','X','X','X','X','X','1'), ('U','X','X','X','X','X','X','X','1'), ('U','X','1','0','X','X','1','0','1'), ('U','X','0','1','X','X','0','1','1')) [Constant]

Definition at line 1595 of file [fixed_pkg_c.vhdl](#).

6.1.3.11 **MATH_REAL** [Package]

Definition at line 1464 of file [fixed_pkg_c.vhdl](#).

6.1.3.12 **MVL9_indexed_by_char**(**CHARACTER**)**STD_ULOGIC** [Type]

Definition at line 6862 of file [fixed_pkg_c.vhdl](#).

6.1.3.13 **MVL9_to_char** **char_indexed_by_MVL9** **::=**"UX01ZWLH-" [Constant]

Definition at line 6865 of file [fixed_pkg_c.vhdl](#).

6.1.3.14 **MVL9plus** ('U','X','0','1','Z','W','L','H','-','error') [Type]

Definition at line 6860 of file [fixed_pkg_c.vhdl](#).

6.1.3.15 **MVL9plus_indexed_by_char**(**CHARACTER**)**MVL9plus** [Type]

Definition at line 6863 of file [fixed_pkg_c.vhdl](#).

6.1.3.16 **NASF UNRESOLVED_sfixed**(**0** **downto** **1**)**::=(others=>'0')** [Constant]

Definition at line 1471 of file [fixed_pkg_c.vhdl](#).

6.1.3.17 **NAUF UNRESOLVED_ufixed**(**0** **downto** **1**)**::=(others=>'0')** [Constant]

Definition at line 1470 of file [fixed_pkg_c.vhdl](#).

6.1.3.18 **NBSP CHARACTER:=CHARACTER'val(160)** [Constant]

Definition at line 6872 of file [fixed_pkg_c.vhdl](#).

6.1.3.19 **NSLV STD_ULONGIC_VECTOR(0 downto 1):=("0")** [Constant]

Definition at line 1472 of file [fixed_pkg_c.vhdl](#).

6.1.3.20 **NUS STRING(2 to 1):=("0")** [Constant]

Definition at line 6873 of file [fixed_pkg_c.vhdl](#).

6.1.3.21 **nybble STD_ULONGIC_VECTOR(2 downto 0)** [Shared Variable]

Definition at line 7341 of file [fixed_pkg_c.vhdl](#).

6.1.3.22 **nybble STD_ULONGIC_VECTOR(3 downto 0)** [Shared Variable]

Definition at line 7598 of file [fixed_pkg_c.vhdl](#).

6.1.3.23 **stdlogic_table (STD_ULONGIC,STD_ULONGIC)STD_ULONGIC** [Type]

Definition at line 1594 of file [fixed_pkg_c.vhdl](#).

6.1.3.24 **xgood BOOLEAN** [Shared Variable]

Definition at line 7340 of file [fixed_pkg_c.vhdl](#).

The documentation for this class was generated from the following file:

- [fixed_pkg_c.vhdl](#)

6.2 Behavioral Architecture Reference

Processes

- [PROCESS_13\(clk \)](#)

Types

- [memory_type\(0 to 29 \)integerrange- 128 to 127](#)

Signals

- [i integerrange 0 to 30 := 0](#)
- [sine memory_type :=\(0 , 16 , 31 , 45 , 58 , 67 , 74 , 77 , 77 , 74 , 67 , 58 , 45 , 31 , 16 , 0 , -16 , -31 , -45 , -58 , -67 , -74 , -77 , -77 , -74 , -67 , -58 , -45 , -31 , -16 \)](#)

6.2.1 Detailed Description

Definition at line 12 of file [sinewave.vhd](#).

6.2.2 Member Function Documentation

6.2.2.1 PROCESS_13(clk) [Process]

Definition at line 21 of file [sinewave.vhd](#).

6.2.3 Member Data Documentation

6.2.3.1 i integerrange 0 to 30 := 0 [Signal]

Definition at line 13 of file [sinewave.vhd](#).

6.2.3.2 memory_type (0 to 29)integerrange- 128 to 127 [Type]

Definition at line 14 of file [sinewave.vhd](#).

6.2.3.3 sine memory_type :=(0 , 16 , 31 , 45 , 58 , 67 , 74 , 77 , 77 , 74 , 67 , 58 , 45 , 31 , 16 , 0 , - 16 , - 31 , - 45 , - 58 , - 67 , - 74 , - 77 , - 74 , - 67 , - 58 , - 45 , - 31 , - 16) [Signal]

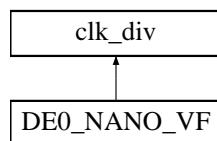
Definition at line 17 of file [sinewave.vhd](#).

The documentation for this class was generated from the following file:

- [sinewave.vhd](#)

6.3 clk_div Entity Reference

Inheritance diagram for clk_div:



Entities

- [div](#) architecture

Libraries

- [IEEE](#)

Use Clauses

- [STD_LOGIC_UNSIGNED](#)
- [STD_LOGIC_1164](#)

Ports

- `clk_in` in `STD_LOGIC`
- `en` in `STD_LOGIC`
- `div` in `STD_LOGIC_VECTOR(15 DOWNTO 0)`
- `clk_out` out `STD_LOGIC`

6.3.1 Detailed Description

Definition at line 5 of file `clk_div.vhd`.

6.3.2 Member Data Documentation

6.3.2.1 `clk_in` in `STD_LOGIC` [Port]

Definition at line 7 of file `clk_div.vhd`.

6.3.2.2 `clk_out` out `STD_LOGIC` [Port]

Definition at line 10 of file `clk_div.vhd`.

6.3.2.3 `div` in `STD_LOGIC_VECTOR(15 DOWNTO 0)` [Port]

Definition at line 8 of file `clk_div.vhd`.

6.3.2.4 `en` in `STD_LOGIC` [Port]

Definition at line 7 of file `clk_div.vhd`.

6.3.2.5 `IEEE` [Library]

Definition at line 1 of file `clk_div.vhd`.

6.3.2.6 `STD_LOGIC_1164` [Package]

Definition at line 3 of file `clk_div.vhd`.

6.3.2.7 `STD_LOGIC_UNSIGNED` [Package]

Definition at line 2 of file `clk_div.vhd`.

The documentation for this class was generated from the following file:

- `clk_div.vhd`

6.4 comparador Architecture Reference

Processes

- `PROCESS_1(clk)`
- `PROCESS_2(almost)`

Signals

- `comp_int std_logic_vector(n_bits_c - 1 downto 0)`

6.4.1 Detailed Description

Definition at line 24 of file [comparador.vhd](#).

6.4.2 Member Function Documentation

6.4.2.1 PROCESS_1(clk) [Process]

Definition at line 32 of file [comparador.vhd](#).

6.4.2.2 PROCESS_2(amost) [Process]

Definition at line 45 of file [comparador.vhd](#).

6.4.3 Member Data Documentation

6.4.3.1 comp_int std_logic_vector(n_bits_c - 1 downto 0) [Signal]

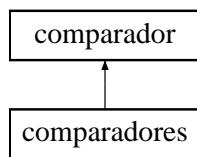
Definition at line 27 of file [comparador.vhd](#).

The documentation for this class was generated from the following file:

- [comparador.vhd](#)

6.5 comparador Entity Reference

Inheritance diagram for comparador:



Entities

- [comparador](#) architecture

Libraries

- [IEEE](#)

Use Clauses

- [STD_LOGIC_UNSIGNED](#)
- [STD_LOGIC_1164](#)

- `fixed_pkg`
- `numeric_std`

Generics

- `n_bits_c integer:= 16`

Ports

- `clk in std_logic`
- `en in std_logic`
- `comp in std_logic_vector(n_bits_c - 1 downto 0)`
- `c in std_logic_vector(n_bits_c - 1 downto 0)`
- `amost in std_logic`
- `comp_out out std_logic`

6.5.1 Detailed Description

Definition at line 8 of file `comparador.vhd`.

6.5.2 Member Data Documentation

6.5.2.1 `amost in std_logic [Port]`

Definition at line 17 of file `comparador.vhd`.

6.5.2.2 `c in std_logic_vector(n_bits_c - 1 downto 0) [Port]`

Definition at line 16 of file `comparador.vhd`.

6.5.2.3 `clk in std_logic [Port]`

Definition at line 13 of file `comparador.vhd`.

6.5.2.4 `comp in std_logic_vector(n_bits_c - 1 downto 0) [Port]`

Definition at line 15 of file `comparador.vhd`.

6.5.2.5 `comp_out out std_logic [Port]`

Definition at line 19 of file `comparador.vhd`.

6.5.2.6 `en in std_logic [Port]`

Definition at line 14 of file `comparador.vhd`.

6.5.2.7 `fixed_pkg [Package]`

Definition at line 5 of file `comparador.vhd`.

6.5.2.8 IEEE [Library]

Definition at line 2 of file [comparador.vhd](#).

6.5.2.9 n_bits_c integer:=16 [Generic]

Definition at line 11 of file [comparador.vhd](#).

6.5.2.10 numeric_std [Package]

Definition at line 6 of file [comparador.vhd](#).

6.5.2.11 STD_LOGIC_1164 [Package]

Definition at line 4 of file [comparador.vhd](#).

6.5.2.12 STD_LOGIC_UNSIGNED [Package]

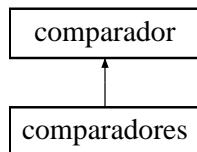
Definition at line 3 of file [comparador.vhd](#).

The documentation for this class was generated from the following file:

- [comparador.vhd](#)

6.6 comparadores Entity Reference

Inheritance diagram for comparadores:



Entities

- [comparadores](#) architecture

Libraries

- [IEEE](#)

Use Clauses

- [STD_LOGIC_UNSIGNED](#)
- [STD_LOGIC_1164](#)
- [fixed_pkg](#)
- [numeric_std](#)
- [my_types_pkg](#)

Generics

- MAX integer:= 3526000
- n_bits_c integer:= 22

Ports

- clk in std_logic
- en in std_logic
- comp in COMP_ARRAY
- c in COMP_ARRAY
- amost in std_logic_vector(24 downto 1)
- comp_out out std_logic_vector(24 downto 1)

6.6.1 Detailed Description

Definition at line 19 of file [comparadores.vhd](#).

6.6.2 Member Data Documentation

6.6.2.1 amost in std_logic_vector(24 downto 1) [Port]

Definition at line 28 of file [comparadores.vhd](#).

6.6.2.2 c in COMP_ARRAY [Port]

Definition at line 27 of file [comparadores.vhd](#).

6.6.2.3 clk in std_logic [Port]

Definition at line 24 of file [comparadores.vhd](#).

6.6.2.4 comp in COMP_ARRAY [Port]

Definition at line 26 of file [comparadores.vhd](#).

6.6.2.5 comp_out out std_logic_vector(24 downto 1) [Port]

Definition at line 30 of file [comparadores.vhd](#).

6.6.2.6 en in std_logic [Port]

Definition at line 25 of file [comparadores.vhd](#).

6.6.2.7 fixed_pkg [Package]

Definition at line 4 of file [comparadores.vhd](#).

6.6.2.8 IEEE [Library]

Definition at line 1 of file [comparadores.vhd](#).

6.6.2.9 MAX integer:= 3526000 [Generic]

Definition at line 20 of file [comparadores.vhd](#).

6.6.2.10 my_types_pkg [Package]

Definition at line 6 of file [comparadores.vhd](#).

6.6.2.11 n_bits_c integer:= 22 [Generic]

Definition at line 22 of file [comparadores.vhd](#).

6.6.2.12 numeric_std [Package]

Definition at line 5 of file [comparadores.vhd](#).

6.6.2.13 STD_LOGIC_1164 [Package]

Definition at line 3 of file [comparadores.vhd](#).

6.6.2.14 STD_LOGIC_UNSIGNED [Package]

Definition at line 2 of file [comparadores.vhd](#).

The documentation for this class was generated from the following file:

- [comparadores.vhd](#)

6.7 comparadores Architecture Reference

Components

- [comparador](#)

Instantiations

- [u_comparador](#)

6.7.1 Detailed Description

Definition at line 34 of file [comparadores.vhd](#).

6.7.2 Member Data Documentation

6.7.2.1 comparador [Component]

Definition at line 36 of file [comparadores.vhd](#).

6.7.2.2 u comparador [Instantiation]

Definition at line 61 of file [comparadores.vhd](#).

The documentation for this class was generated from the following file:

- [comparadores.vhd](#)

6.8 contador Architecture Reference

Processes

- [PROCESS_3\(clk \)](#)

Signals

- [count_int std_logic_vector\(n_bits - 1 downto 0\)](#)

6.8.1 Detailed Description

Definition at line 25 of file [contador.vhd](#).

6.8.2 Member Function Documentation

6.8.2.1 PROCESS_3(clk) [Process]

Definition at line 31 of file [contador.vhd](#).

6.8.3 Member Data Documentation

6.8.3.1 count_int std_logic_vector(n_bits - 1 downto 0) [Signal]

Definition at line 27 of file [contador.vhd](#).

The documentation for this class was generated from the following file:

- [contador.vhd](#)

6.9 contador Entity Reference

Entities

- [contador](#) architecture

Libraries

- IEEE

Use Clauses

- STD_LOGIC_UNSIGNED
- STD_LOGIC_1164
- numeric_std

Generics

- n_bits integer:= 16

Ports

- clk **in std_logic**
- en **in std_logic**
- reset **in std_logic**
- sinc **out std_logic**
- count_max **in std_logic_vector(n_bits - 1 downto 0)**
- count_ini **in std_logic_vector(n_bits - 1 downto 0)**
- count_comp **in std_logic_vector(n_bits - 1 downto 0)**
- count **out std_logic_vector(n_bits - 1 downto 0)**
- comp **out std_logic**

6.9.1 Detailed Description

Definition at line 6 of file [contador.vhd](#).

6.9.2 Member Data Documentation

6.9.2.1 clk **in std_logic** [Port]

Definition at line 11 of file [contador.vhd](#).

6.9.2.2 comp **out std_logic** [Port]

Definition at line 20 of file [contador.vhd](#).

6.9.2.3 count **out std_logic_vector(n_bits - 1 downto 0)** [Port]

Definition at line 18 of file [contador.vhd](#).

6.9.2.4 count_comp **in std_logic_vector(n_bits - 1 downto 0)** [Port]

Definition at line 17 of file [contador.vhd](#).

6.9.2.5 count_ini **in std_logic_vector(n_bits - 1 downto 0)** [Port]

Definition at line 16 of file [contador.vhd](#).

6.9.2.6 **count_max** **in std_logic_vector(n_bits - 1 downto 0)** [Port]

Definition at line 15 of file [contador.vhd](#).

6.9.2.7 **en** **in std_logic** [Port]

Definition at line 12 of file [contador.vhd](#).

6.9.2.8 **IEEE** [Library]

Definition at line 1 of file [contador.vhd](#).

6.9.2.9 **n_bits** **integer:= 16** [Generic]

Definition at line 9 of file [contador.vhd](#).

6.9.2.10 **numeric_std** [Package]

Definition at line 4 of file [contador.vhd](#).

6.9.2.11 **reset** **in std_logic** [Port]

Definition at line 13 of file [contador.vhd](#).

6.9.2.12 **sinc** **out std_logic** [Port]

Definition at line 14 of file [contador.vhd](#).

6.9.2.13 **STD_LOGIC_1164** [Package]

Definition at line 3 of file [contador.vhd](#).

6.9.2.14 **STD_LOGIC_UNSIGNED** [Package]

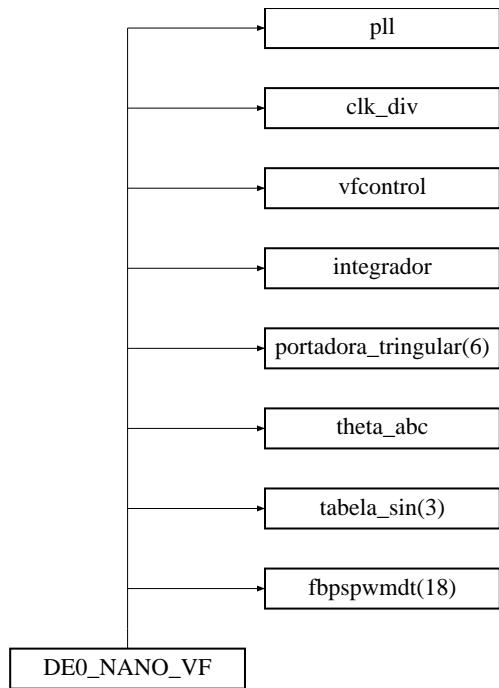
Definition at line 2 of file [contador.vhd](#).

The documentation for this class was generated from the following file:

- [contador.vhd](#)

6.10 DE0_NANO_VF Entity Reference

Inheritance diagram for DE0_NANO_VF:



Entities

- **MAIN** architecture

Libraries

- ieee

Use Clauses

- std_logic_1164
- std_logic_arith
- std_logic_unsigned
- NUMERIC_STD
- fixed_pkg
- my_types_pkg

Generics

- N integer:= 3
- Nin integer:= 13
- Nout integer:= 30
- n_bits_phase integer:= 30
- n_bits_c integer:= 16
- TAM_MEM integer:= 32
- NBITS_MEM_ADDRESS integer:= 6
- ID_MEM_DAC integer:= 28
- ID_MEM_SW1 integer:= 30
- I integer:= 1
- F integer:= 14

Ports

- **CLOCK_50** **in std_logic**
- **LED** **out std_logic_vector(7 DOWNTO 0)**
- **SW** **in std_logic_vector(3 DOWNTO 0)**
- **KEY** **in std_logic_vector(1 DOWNTO 0)**
- **RESET_FA** **out std_logic_vector(N - 1 DOWNTO 0)**
- **PWM1L_FA** **out std_logic_vector(N - 1 DOWNTO 0)**
- **PWM1H_FA** **out std_logic_vector(N - 1 DOWNTO 0)**
- **PWM2L_FA** **out std_logic_vector(N - 1 DOWNTO 0)**
- **PWM2H_FA** **out std_logic_vector(N - 1 DOWNTO 0)**
- **INT0_FA** **in std_logic_vector(N - 1 DOWNTO 0)**
- **RESET_FB** **out std_logic_vector(N - 1 DOWNTO 0)**
- **PWM1L_FB** **out std_logic_vector(N - 1 DOWNTO 0)**
- **PWM1H_FB** **out std_logic_vector(N - 1 DOWNTO 0)**
- **PWM2L_FB** **out std_logic_vector(N - 1 DOWNTO 0)**
- **PWM2H_FB** **out std_logic_vector(N - 1 DOWNTO 0)**
- **INT0_FB** **in std_logic_vector(N - 1 DOWNTO 0)**
- **RESET_FC** **out std_logic_vector(N - 1 DOWNTO 0)**
- **PWM1L_FC** **out std_logic_vector(N - 1 DOWNTO 0)**
- **PWM1H_FC** **out std_logic_vector(N - 1 DOWNTO 0)**
- **PWM2L_FC** **out std_logic_vector(N - 1 DOWNTO 0)**
- **PWM2H_FC** **out std_logic_vector(N - 1 DOWNTO 0)**
- **INT0_FC** **in std_logic_vector(N - 1 DOWNTO 0)**
- **GPIO_0** **out std_logic_vector(3 DOWNTO 0)**

6.10.1 Detailed Description

Definition at line 15 of file [DE0_NANO_VF.vhd](#).

6.10.2 Member Data Documentation

6.10.2.1 CLOCK_50 **in std_logic** [Port]

Definition at line 30 of file [DE0_NANO_VF.vhd](#).

6.10.2.2 F **integer:= 14** [Generic]

Definition at line 28 of file [DE0_NANO_VF.vhd](#).

6.10.2.3 fixed_pkg [Package]

Definition at line 10 of file [DE0_NANO_VF.vhd](#).

6.10.2.4 GPIO_0 **out std_logic_vector(3 DOWNTO 0)** [Port]

Definition at line 70 of file [DE0_NANO_VF.vhd](#).

6.10.2.5 I **integer:= 1** [Generic]

Definition at line 26 of file [DE0_NANO_VF.vhd](#).

6.10.2.6 **ID_MEM_DAC** **integer:= 28** [Generic]

Definition at line 24 of file [DE0_NANO_VF.vhd](#).

6.10.2.7 **ID_MEM_SW1** **integer:= 30** [Generic]

Definition at line 25 of file [DE0_NANO_VF.vhd](#).

6.10.2.8 **ieee** [Library]

Definition at line 3 of file [DE0_NANO_VF.vhd](#).

6.10.2.9 **INT0_FA** **in std_logic_vector(N - 1 DOWNTO 0)** [Port]

Definition at line 44 of file [DE0_NANO_VF.vhd](#).

6.10.2.10 **INT0_FB** **in std_logic_vector(N - 1 DOWNTO 0)** [Port]

Definition at line 52 of file [DE0_NANO_VF.vhd](#).

6.10.2.11 **INT0_FC** **in std_logic_vector(N - 1 DOWNTO 0)** [Port]

Definition at line 61 of file [DE0_NANO_VF.vhd](#).

6.10.2.12 **KEY** **in std_logic_vector(1 DOWNTO 0)** [Port]

Definition at line 33 of file [DE0_NANO_VF.vhd](#).

6.10.2.13 **LED** **out std_logic_vector(7 DOWNTO 0)** [Port]

Definition at line 31 of file [DE0_NANO_VF.vhd](#).

6.10.2.14 **my_types_pkg** [Package]

Definition at line 11 of file [DE0_NANO_VF.vhd](#).

6.10.2.15 **N** **integer:= 3** [Generic]

Definition at line 17 of file [DE0_NANO_VF.vhd](#).

6.10.2.16 **n_bits_c** **integer:= 16** [Generic]

Definition at line 21 of file [DE0_NANO_VF.vhd](#).

6.10.2.17 **n_bits_phase** **integer:= 30** [Generic]

Definition at line 20 of file [DE0_NANO_VF.vhd](#).

6.10.2.18 NBITS_MEM_ADDRESS **integer:= 6** [Generic]

Definition at line 23 of file [DE0_NANO_VF.vhd](#).

6.10.2.19 Nin **integer:= 13** [Generic]

Definition at line 18 of file [DE0_NANO_VF.vhd](#).

6.10.2.20 Nout **integer:= 30** [Generic]

Definition at line 19 of file [DE0_NANO_VF.vhd](#).

6.10.2.21 NUMERIC_STD [Package]

Definition at line 7 of file [DE0_NANO_VF.vhd](#).

6.10.2.22 PWM1H_FA **out std_logic_vector(N - 1 DOWNTO 0)** [Port]

Definition at line 41 of file [DE0_NANO_VF.vhd](#).

6.10.2.23 PWM1H_FB **out std_logic_vector(N - 1 DOWNTO 0)** [Port]

Definition at line 49 of file [DE0_NANO_VF.vhd](#).

6.10.2.24 PWM1H_FC **out std_logic_vector(N - 1 DOWNTO 0)** [Port]

Definition at line 58 of file [DE0_NANO_VF.vhd](#).

6.10.2.25 PWM1L_FA **out std_logic_vector(N - 1 DOWNTO 0)** [Port]

Definition at line 40 of file [DE0_NANO_VF.vhd](#).

6.10.2.26 PWM1L_FB **out std_logic_vector(N - 1 DOWNTO 0)** [Port]

Definition at line 48 of file [DE0_NANO_VF.vhd](#).

6.10.2.27 PWM1L_FC **out std_logic_vector(N - 1 DOWNTO 0)** [Port]

Definition at line 57 of file [DE0_NANO_VF.vhd](#).

6.10.2.28 PWM2H_FA **out std_logic_vector(N - 1 DOWNTO 0)** [Port]

Definition at line 43 of file [DE0_NANO_VF.vhd](#).

6.10.2.29 PWM2H_FB **out std_logic_vector(N - 1 DOWNTO 0)** [Port]

Definition at line 51 of file [DE0_NANO_VF.vhd](#).

6.10.2.30 **PWM2H_FC** **out std_logic_vector(N - 1 DOWNTO 0)** [Port]

Definition at line 60 of file [DE0_NANO_VF.vhd](#).

6.10.2.31 **PWM2L_FA** **out std_logic_vector(N - 1 DOWNTO 0)** [Port]

Definition at line 42 of file [DE0_NANO_VF.vhd](#).

6.10.2.32 **PWM2L_FB** **out std_logic_vector(N - 1 DOWNTO 0)** [Port]

Definition at line 50 of file [DE0_NANO_VF.vhd](#).

6.10.2.33 **PWM2L_FC** **out std_logic_vector(N - 1 DOWNTO 0)** [Port]

Definition at line 59 of file [DE0_NANO_VF.vhd](#).

6.10.2.34 **RESET_FA** **out std_logic_vector(N - 1 DOWNTO 0)** [Port]

Definition at line 39 of file [DE0_NANO_VF.vhd](#).

6.10.2.35 **RESET_FB** **out std_logic_vector(N - 1 DOWNTO 0)** [Port]

Definition at line 47 of file [DE0_NANO_VF.vhd](#).

6.10.2.36 **RESET_FC** **out std_logic_vector(N - 1 DOWNTO 0)** [Port]

Definition at line 56 of file [DE0_NANO_VF.vhd](#).

6.10.2.37 **std_logic_1164** [Package]

Definition at line 4 of file [DE0_NANO_VF.vhd](#).

6.10.2.38 **std_logic_arith** [Package]

Definition at line 5 of file [DE0_NANO_VF.vhd](#).

6.10.2.39 **std_logic_unsigned** [Package]

Definition at line 6 of file [DE0_NANO_VF.vhd](#).

6.10.2.40 **SW** **in std_logic_vector(3 DOWNTO 0)** [Port]

Definition at line 32 of file [DE0_NANO_VF.vhd](#).

6.10.2.41 **TAM_MEM** **integer:= 32** [Generic]

Definition at line 22 of file [DE0_NANO_VF.vhd](#).

The documentation for this class was generated from the following file:

- [DE0_NANO_VF.vhd](#)

6.11 deadtime Entity Reference

Entities

- `deadtime_archetecture` architecture

Libraries

- `ieee`

Use Clauses

- `std_logic_1164`
- `std_logic_arith`
- `std_logic_unsigned`

Ports

- `p_Pwm_In in std_logic`
- `CLK in std_logic`
- `p_Pwm1_Out out std_logic`
- `p_Pwm2_Out out std_logic`

6.11.1 Detailed Description

Definition at line 10 of file `deadtime.vhd`.

6.11.2 Member Data Documentation

6.11.2.1 CLK in std_logic [Port]

Definition at line 14 of file `deadtime.vhd`.

6.11.2.2 ieee [Library]

Definition at line 3 of file `deadtime.vhd`.

6.11.2.3 p_Pwm1_Out out std_logic [Port]

Definition at line 15 of file `deadtime.vhd`.

6.11.2.4 p_Pwm2_Out out std_logic [Port]

Definition at line 16 of file `deadtime.vhd`.

6.11.2.5 p_Pwm_In in std_logic [Port]

Definition at line 13 of file `deadtime.vhd`.

6.11.2.6 std_logic_1164 [Package]

Definition at line 4 of file [deadtime.vhd](#).

6.11.2.7 std_logic_arith [Package]

Definition at line 5 of file [deadtime.vhd](#).

6.11.2.8 std_logic_unsigned [Package]

Definition at line 6 of file [deadtime.vhd](#).

The documentation for this class was generated from the following file:

- [deadtime.vhd](#)

6.12 deadtime_archetecture Architecture Reference

Processes

- [PROCESS_6\(CLK \)](#)

Signals

- [sig_Not_Pwm_In std_logic](#)

6.12.1 Detailed Description

Definition at line 19 of file [deadtime.vhd](#).

6.12.2 Member Function Documentation

6.12.2.1 PROCESS_6(CLK) [Process]

Definition at line 24 of file [deadtime.vhd](#).

6.12.3 Member Data Documentation

6.12.3.1 sig_Not_Pwm_In std_logic [Signal]

Definition at line 20 of file [deadtime.vhd](#).

The documentation for this class was generated from the following file:

- [deadtime.vhd](#)

6.13 div Architecture Reference

Processes

- [PROCESS_0\(clk_in , en , div \)](#)

Signals

- `count STD_LOGIC_VECTOR(15 DOWNTO 0)`
- `clk_out_bi STD_LOGIC`

6.13.1 Detailed Description

Definition at line 13 of file [clk_div.vhd](#).

6.13.2 Member Function Documentation

6.13.2.1 PROCESS_0(clk_in , en , div) [Process]

Definition at line 18 of file [clk_div.vhd](#).

6.13.3 Member Data Documentation

6.13.3.1 clk_out_bi STD_LOGIC [Signal]

Definition at line 15 of file [clk_div.vhd](#).

6.13.3.2 count STD_LOGIC_VECTOR(15 DOWNTO 0) [Signal]

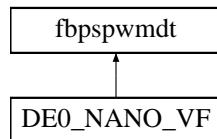
Definition at line 14 of file [clk_div.vhd](#).

The documentation for this class was generated from the following file:

- [clk_div.vhd](#)

6.14 fbpspwm dt Entity Reference

Inheritance diagram for fbpspwm dt:



Entities

- [fbpspwm dt_arch](#) architecture

Libraries

- IEEE

Use Clauses

- STD_LOGIC_UNSIGNED
- STD_LOGIC_1164
- fixed_pkg
- numeric_std

Generics

- n_bits_c **integer:= 16**
- c_Dead_t **integer:= 45**

Ports

- clk **in std_logic**
- en **in std_logic**
- comp **std_logic_vector(n_bits_c - 1 downto 0)**
- c **in std_logic_vector(n_bits_c - 1 downto 0)**
- amost **in std_logic**
- port_PWM01 **out std_logic**
- port_PWM02 **out std_logic**

6.14.1 Detailed Description

Definition at line 11 of file [fbpspwm.vhd](#).

6.14.2 Member Data Documentation

6.14.2.1 amost **in std_logic** [Port]

Definition at line 21 of file [fbpspwm.vhd](#).

6.14.2.2 c **in std_logic_vector(n_bits_c - 1 downto 0)** [Port]

Definition at line 20 of file [fbpspwm.vhd](#).

6.14.2.3 c_Dead_t **integer:= 45** [Generic]

Definition at line 15 of file [fbpspwm.vhd](#).

6.14.2.4 clk **in std_logic** [Port]

Definition at line 17 of file [fbpspwm.vhd](#).

6.14.2.5 comp **std_logic_vector(n_bits_c - 1 downto 0)** [Port]

Definition at line 19 of file [fbpspwm.vhd](#).

6.14.2.6 en **in std_logic** [Port]

Definition at line 18 of file [fbpspwm.vhd](#).

6.14.2.7 fixed_pkg [Package]

Definition at line 7 of file fbpspwmdt.vhd.

6.14.2.8 IEEE [Library]

Definition at line 4 of file fbpspwmdt.vhd.

6.14.2.9 n_bits_c integer:= 16 [Generic]

Definition at line 13 of file fbpspwmdt.vhd.

6.14.2.10 numeric_std [Package]

Definition at line 8 of file fbpspwmdt.vhd.

6.14.2.11 port_PWM01 out std_logic [Port]

Definition at line 22 of file fbpspwmdt.vhd.

6.14.2.12 port_PWM02 out std_logic [Port]

Definition at line 24 of file fbpspwmdt.vhd.

6.14.2.13 STD_LOGIC_1164 [Package]

Definition at line 6 of file fbpspwmdt.vhd.

6.14.2.14 STD_LOGIC_UNSIGNED [Package]

Definition at line 5 of file fbpspwmdt.vhd.

The documentation for this class was generated from the following file:

- fbpspwmdt.vhd

6.15 fbpspwmdt_arch Architecture Reference

Processes

- PROCESS_7(clk)
- PROCESS_8(almost)
- PROCESS_9(clk)

Signals

- comp_int std_logic_vector(n_bits_c - 1 downto 0)
- sig_Not_Pwm_In std_logic
- comp_out std_logic
- p_Pwm_In std_logic

6.15.1 Detailed Description

Definition at line 31 of file [fbpspwm.vhd](#).

6.15.2 Member Function Documentation

6.15.2.1 PROCESS_7(clk) [Process]

Definition at line 42 of file [fbpspwm.vhd](#).

6.15.2.2 PROCESS_8(amost) [Process]

Definition at line 55 of file [fbpspwm.vhd](#).

6.15.2.3 PROCESS_9(clk) [Process]

Definition at line 67 of file [fbpspwm.vhd](#).

6.15.3 Member Data Documentation

6.15.3.1 comp_int std_logic_vector(n_bits_c - 1 downto 0) [Signal]

Definition at line 34 of file [fbpspwm.vhd](#).

6.15.3.2 comp_out std_logic [Signal]

Definition at line 36 of file [fbpspwm.vhd](#).

6.15.3.3 p_Pwm_In std_logic [Signal]

Definition at line 37 of file [fbpspwm.vhd](#).

6.15.3.4 sig_Not_Pwm_In std_logic [Signal]

Definition at line 35 of file [fbpspwm.vhd](#).

The documentation for this class was generated from the following file:

- [fbpspwm.vhd](#)

6.16 fixed_float_types Package Reference

Types

- [fixed_round_style_type](#)(fixed_round,fixed_truncate)
- [fixed_overflow_style_type](#)(fixed_saturate,fixed_wrap)
- [round_type](#)(round_nearest,round_inf,round_neginf,round_zero)

6.16.1 Detailed Description

Definition at line 16 of file [fixed_float_types_c.vhdl](#).

6.16.2 Member Data Documentation

6.16.2.1 **fixed_overflow_style_type (fixed_saturate,fixed_wrap)** [Type]

Definition at line 22 of file [fixed_float_types_c.vhdl](#).

6.16.2.2 **fixed_round_style_type (fixed_round,fixed_truncate)** [Type]

Definition at line 20 of file [fixed_float_types_c.vhdl](#).

6.16.2.3 **round_type (round_nearest,round_inf,round_neginf,round_zero)** [Type]

Definition at line 29 of file [fixed_float_types_c.vhdl](#).

The documentation for this class was generated from the following file:

- [fixed_float_types_c.vhdl](#)

6.17 fixed_pkg Package Reference

Package Body >> [fixed_pkg](#)

Functions

- **UNRESOLVED_sfixed "abs"(arg: in UNRESOLVED_sfixed)**
- **UNRESOLVED_sfixed "-"(arg: in UNRESOLVED_sfixed)**
- **UNRESOLVED_ufixed "+"(I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)**
- **UNRESOLVED_sfixed "+"(I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)**
- **UNRESOLVED_ufixed "-"(I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)**
- **UNRESOLVED_sfixed "-"(I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)**
- **UNRESOLVED_ufixed "*"(I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)**
- **UNRESOLVED_sfixed "*"(I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)**
- **UNRESOLVED_ufixed "/"(I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)**
- **UNRESOLVED_sfixed "/"(I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)**
- **UNRESOLVED_ufixed "rem"(I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)**
- **UNRESOLVED_sfixed "rem"(I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)**
- **UNRESOLVED_ufixed "mod"(I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)**
- **UNRESOLVED_sfixed "mod"(I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)**
- **UNRESOLVED_ufixed "+"(I: in UNRESOLVED_ufixed, r: in REAL)**
- **UNRESOLVED_ufixed "+"(I: in REAL, r: in UNRESOLVED_ufixed)**
- **UNRESOLVED_ufixed "+"(I: in UNRESOLVED_ufixed, r: in NATURAL)**
- **UNRESOLVED_ufixed "+"(I: in NATURAL, r: in UNRESOLVED_ufixed)**
- **UNRESOLVED_ufixed "-"(I: in UNRESOLVED_ufixed, r: in REAL)**
- **UNRESOLVED_ufixed "-"(I: in REAL, r: in UNRESOLVED_ufixed)**
- **UNRESOLVED_ufixed "-"(I: in UNRESOLVED_ufixed, r: in NATURAL)**
- **UNRESOLVED_ufixed "-"(I: in NATURAL, r: in UNRESOLVED_ufixed)**
- **UNRESOLVED_ufixed "*"(I: in UNRESOLVED_ufixed, r: in REAL)**
- **UNRESOLVED_ufixed "*"(I: in REAL, r: in UNRESOLVED_ufixed)**
- **UNRESOLVED_ufixed "*"(I: in UNRESOLVED_ufixed, r: in NATURAL)**
- **UNRESOLVED_ufixed "*"(I: in NATURAL, r: in UNRESOLVED_ufixed)**
- **UNRESOLVED_ufixed "/"(I: in UNRESOLVED_ufixed, r: in REAL)**
- **UNRESOLVED_ufixed "/"(I: in REAL, r: in UNRESOLVED_ufixed)**

- **UNRESOLVED_ufixed** `"/"(I: in UNRESOLVED_ufixed, r: in NATURAL)`
- **UNRESOLVED_ufixed** `"/"(I: in NATURAL, r: in UNRESOLVED_ufixed)`
- **UNRESOLVED_ufixed** `"rem"(I: in UNRESOLVED_ufixed, r: in REAL)`
- **UNRESOLVED_ufixed** `"rem"(I: in REAL, r: in UNRESOLVED_ufixed)`
- **UNRESOLVED_ufixed** `"rem"(I: in UNRESOLVED_ufixed, r: in NATURAL)`
- **UNRESOLVED_ufixed** `"rem"(I: in NATURAL, r: in UNRESOLVED_ufixed)`
- **UNRESOLVED_ufixed** `"mod"(I: in UNRESOLVED_ufixed, r: in REAL)`
- **UNRESOLVED_ufixed** `"mod"(I: in REAL, r: in UNRESOLVED_ufixed)`
- **UNRESOLVED_ufixed** `"mod"(I: in UNRESOLVED_ufixed, r: in NATURAL)`
- **UNRESOLVED_ufixed** `"mod"(I: in NATURAL, r: in UNRESOLVED_ufixed)`
- **UNRESOLVED_sfixed** `"+"(I: in UNRESOLVED_sfixed, r: in REAL)`
- **UNRESOLVED_sfixed** `"+"(I: in REAL, r: in UNRESOLVED_sfixed)`
- **UNRESOLVED_sfixed** `"+"(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- **UNRESOLVED_sfixed** `"+"(I: in INTEGER, r: in UNRESOLVED_sfixed)`
- **UNRESOLVED_sfixed** `"-(I: in UNRESOLVED_sfixed, r: in REAL)`
- **UNRESOLVED_sfixed** `"-(I: in REAL, r: in UNRESOLVED_sfixed)`
- **UNRESOLVED_sfixed** `"-(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- **UNRESOLVED_sfixed** `"-(I: in INTEGER, r: in UNRESOLVED_sfixed)`
- **UNRESOLVED_sfixed** `"*(I: in UNRESOLVED_sfixed, r: in REAL)`
- **UNRESOLVED_sfixed** `"*(I: in REAL, r: in UNRESOLVED_sfixed)`
- **UNRESOLVED_sfixed** `"*(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- **UNRESOLVED_sfixed** `"*(I: in INTEGER, r: in UNRESOLVED_sfixed)`
- **UNRESOLVED_sfixed** `"/"(I: in UNRESOLVED_sfixed, r: in REAL)`
- **UNRESOLVED_sfixed** `"/"(I: in REAL, r: in UNRESOLVED_sfixed)`
- **UNRESOLVED_sfixed** `"/"(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- **UNRESOLVED_sfixed** `"/"(I: in INTEGER, r: in UNRESOLVED_sfixed)`
- **UNRESOLVED_sfixed** `"rem"(I: in UNRESOLVED_sfixed, r: in REAL)`
- **UNRESOLVED_sfixed** `"rem"(I: in REAL, r: in UNRESOLVED_sfixed)`
- **UNRESOLVED_sfixed** `"mod"(I: in UNRESOLVED_sfixed, r: in REAL)`
- **UNRESOLVED_sfixed** `"mod"(I: in REAL, r: in UNRESOLVED_sfixed)`
- **UNRESOLVED_sfixed** `"mod"(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- **UNRESOLVED_sfixed** `"mod"(I: in INTEGER, r: in UNRESOLVED_sfixed)`
- **UNRESOLVED_ufixed** `divide(`
 `I: in UNRESOLVED_ufixed`
 `r: in UNRESOLVED_ufixed`
 `constantround_style: in fixed_round_style_type fixed_round_style`
 `constantguard_bits: in NATURAL fixed_guard_bits`
 `)`
- **UNRESOLVED_sfixed** `divide(`
 `I: in UNRESOLVED_sfixed`
 `r: in UNRESOLVED_sfixed`
 `constantround_style: in fixed_round_style_type fixed_round_style`
 `constantguard_bits: in NATURAL fixed_guard_bits`
 `)`
- **UNRESOLVED_ufixed** `reciprocal(`
 `arg: in UNRESOLVED_ufixed`
 `constantround_style: in fixed_round_style_type fixed_round_style`
 `constantguard_bits: in NATURAL fixed_guard_bits`
 `)`
- **UNRESOLVED_sfixed** `reciprocal(`
 `arg: in UNRESOLVED_sfixed`
 `constantround_style: in fixed_round_style_type fixed_round_style`
 `constantguard_bits: in NATURAL fixed_guard_bits`
 `)`

- **UNRESOLVED_ufixed remainder(**
I: in UNRESOLVED_ufixed
r: in UNRESOLVED_ufixed
constantround_style: in fixed_round_style_type fixed_round_style
constantguard_bits: in NATURAL fixed_guard_bits
)
- **UNRESOLVED_sfixed remainder(**
I: in UNRESOLVED_sfixed
r: in UNRESOLVED_sfixed
constantround_style: in fixed_round_style_type fixed_round_style
constantguard_bits: in NATURAL fixed_guard_bits
)
- **UNRESOLVED_ufixed modulo(**
I: in UNRESOLVED_ufixed
r: in UNRESOLVED_ufixed
constantround_style: in fixed_round_style_type fixed_round_style
constantguard_bits: in NATURAL fixed_guard_bits
)
- **UNRESOLVED_sfixed modulo(**
I: in UNRESOLVED_sfixed
r: in UNRESOLVED_sfixed
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
constantguard_bits: in NATURAL fixed_guard_bits
)
- **UNRESOLVED_ufixed scalb(** y: in UNRESOLVED_ufixed, N: in INTEGER)
• **UNRESOLVED_ufixed scalb(** y: in UNRESOLVED_ufixed, N: in SIGNED)
• **UNRESOLVED_sfixed scalb(** y: in UNRESOLVED_sfixed, N: in INTEGER)
• **UNRESOLVED_sfixed scalb(** y: in UNRESOLVED_sfixed, N: in SIGNED)
• **BOOLEAN Is_Negative(** arg: in UNRESOLVED_sfixed)
• **BOOLEAN ">"(** I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)
• **BOOLEAN ">"(** I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)
• **BOOLEAN "<"(** I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)
• **BOOLEAN "<"(** I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)
• **BOOLEAN "<="(** I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)
• **BOOLEAN "<="(** I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)
• **BOOLEAN ">="(** I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)
• **BOOLEAN ">="(** I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)
• **BOOLEAN ">=(** I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)
• **BOOLEAN ">=(** I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)
• **BOOLEAN "="(** I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)
• **BOOLEAN "="(** I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)
• **BOOLEAN "/="(** I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)
• **BOOLEAN "/="(** I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)
• **STD_ULOGIC \?=**(I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)
• **STD_ULOGIC \?/=(** I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)
• **STD_ULOGIC \?>**(I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)
• **STD_ULOGIC \?>=**(I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)
• **STD_ULOGIC \?<**(I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)
• **STD_ULOGIC \?<=**(I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)
• **STD_ULOGIC \?=**(I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)
• **STD_ULOGIC \?/=(** I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)
• **STD_ULOGIC \?>**(I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)
• **STD_ULOGIC \?>=**(I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)
• **STD_ULOGIC \?<**(I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)
• **STD_ULOGIC \?<=**(I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)
• **BOOLEAN std_match(** I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)
• **BOOLEAN std_match(** I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)

- **UNRESOLVED_ufixed** `maximum(I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)`
- **UNRESOLVED_ufixed** `minimum(I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)`
- **UNRESOLVED_sfixed** `maximum(I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)`
- **UNRESOLVED_sfixed** `minimum(I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)`
- **BOOLEAN** `"="(I: in UNRESOLVED_ufixed, r: in NATURAL)`
- **BOOLEAN** `"/="(I: in UNRESOLVED_ufixed, r: in NATURAL)`
- **BOOLEAN** `">="(I: in UNRESOLVED_ufixed, r: in NATURAL)`
- **BOOLEAN** `"<="(I: in UNRESOLVED_ufixed, r: in NATURAL)`
- **BOOLEAN** `">"(I: in UNRESOLVED_ufixed, r: in NATURAL)`
- **BOOLEAN** `"<"(I: in UNRESOLVED_ufixed, r: in NATURAL)`
- **BOOLEAN** `"="(I: in NATURAL, r: in UNRESOLVED_ufixed)`
- **BOOLEAN** `"/="(I: in NATURAL, r: in UNRESOLVED_ufixed)`
- **BOOLEAN** `">="(I: in NATURAL, r: in UNRESOLVED_ufixed)`
- **BOOLEAN** `"<="(I: in NATURAL, r: in UNRESOLVED_ufixed)`
- **BOOLEAN** `">"(I: in NATURAL, r: in UNRESOLVED_ufixed)`
- **BOOLEAN** `"<"(I: in NATURAL, r: in UNRESOLVED_ufixed)`
- **STD_ULOGIC** `\?=(I: in UNRESOLVED_ufixed, r: in NATURAL)`
- **STD_ULOGIC** `\?/=(I: in UNRESOLVED_ufixed, r: in NATURAL)`
- **STD_ULOGIC** `\?>=(I: in UNRESOLVED_ufixed, r: in NATURAL)`
- **STD_ULOGIC** `\?<=(I: in UNRESOLVED_ufixed, r: in NATURAL)`
- **STD_ULOGIC** `\?>=(I: in UNRESOLVED_ufixed, r: in NATURAL)`
- **STD_ULOGIC** `\?<=(I: in NATURAL, r: in UNRESOLVED_ufixed)`
- **STD_ULOGIC** `\?/=(I: in NATURAL, r: in UNRESOLVED_ufixed)`
- **STD_ULOGIC** `\?>=(I: in NATURAL, r: in UNRESOLVED_ufixed)`
- **STD_ULOGIC** `\?<=(I: in NATURAL, r: in UNRESOLVED_ufixed)`
- **STD_ULOGIC** `\?>=(I: in NATURAL, r: in UNRESOLVED_ufixed)`
- **STD_ULOGIC** `\?<=(I: in NATURAL, r: in UNRESOLVED_ufixed)`
- **UNRESOLVED_ufixed** `maximum(I: in UNRESOLVED_ufixed, r: in NATURAL)`
- **UNRESOLVED_ufixed** `minimum(I: in UNRESOLVED_ufixed, r: in NATURAL)`
- **UNRESOLVED_ufixed** `maximum(I: in NATURAL, r: in UNRESOLVED_ufixed)`
- **UNRESOLVED_ufixed** `minimum(I: in NATURAL, r: in UNRESOLVED_ufixed)`
- **BOOLEAN** `"="(I: in UNRESOLVED_ufixed, r: in REAL)`
- **BOOLEAN** `"/="(I: in UNRESOLVED_ufixed, r: in REAL)`
- **BOOLEAN** `">="(I: in UNRESOLVED_ufixed, r: in REAL)`
- **BOOLEAN** `"<="(I: in UNRESOLVED_ufixed, r: in REAL)`
- **BOOLEAN** `">"(I: in UNRESOLVED_ufixed, r: in REAL)`
- **BOOLEAN** `"<"(I: in UNRESOLVED_ufixed, r: in REAL)`
- **BOOLEAN** `"="(I: in REAL, r: in UNRESOLVED_ufixed)`
- **BOOLEAN** `"/="(I: in REAL, r: in UNRESOLVED_ufixed)`
- **BOOLEAN** `">="(I: in REAL, r: in UNRESOLVED_ufixed)`
- **BOOLEAN** `"<="(I: in REAL, r: in UNRESOLVED_ufixed)`
- **BOOLEAN** `">"(I: in REAL, r: in UNRESOLVED_ufixed)`
- **BOOLEAN** `"<"(I: in REAL, r: in UNRESOLVED_ufixed)`
- **STD_ULOGIC** `\?=(I: in UNRESOLVED_ufixed, r: in REAL)`
- **STD_ULOGIC** `\?/=(I: in UNRESOLVED_ufixed, r: in REAL)`
- **STD_ULOGIC** `\?>=(I: in UNRESOLVED_ufixed, r: in REAL)`
- **STD_ULOGIC** `\?<=(I: in UNRESOLVED_ufixed, r: in REAL)`
- **STD_ULOGIC** `\?>=(I: in UNRESOLVED_ufixed, r: in REAL)`
- **STD_ULOGIC** `\?<=(I: in UNRESOLVED_ufixed, r: in REAL)`
- **STD_ULOGIC** `\?/=(I: in REAL, r: in UNRESOLVED_ufixed)`
- **STD_ULOGIC** `\?>=(I: in REAL, r: in UNRESOLVED_ufixed)`
- **STD_ULOGIC** `\?<=(I: in REAL, r: in UNRESOLVED_ufixed)`
- **STD_ULOGIC** `\?>=(I: in REAL, r: in UNRESOLVED_ufixed)`

- **STD_ULOGIC** `\?<\(I: in REAL, r: in UNRESOLVED_ufixed)`
- **UNRESOLVED_ufixed** `maximum(I: in UNRESOLVED_ufixed, r: in REAL)`
- **UNRESOLVED_ufixed** `maximum(I: in REAL, r: in UNRESOLVED_ufixed)`
- **UNRESOLVED_ufixed** `minimum(I: in UNRESOLVED_ufixed, r: in REAL)`
- **UNRESOLVED_ufixed** `minimum(I: in REAL, r: in UNRESOLVED_ufixed)`
- **BOOLEAN** `"="(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- **BOOLEAN** `"/="(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- **BOOLEAN** `">="(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- **BOOLEAN** `"<="(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- **BOOLEAN** `">=(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- **BOOLEAN** `"<=(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- **BOOLEAN** `"="(I: in INTEGER, r: in UNRESOLVED_sfixed)`
- **BOOLEAN** `"/="(I: in INTEGER, r: in UNRESOLVED_sfixed)`
- **BOOLEAN** `">="(I: in INTEGER, r: in UNRESOLVED_sfixed)`
- **BOOLEAN** `"<="(I: in INTEGER, r: in UNRESOLVED_sfixed)`
- **BOOLEAN** `"<=(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- **BOOLEAN** `\?=/\=(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- **STD_ULOGIC** `\?>=\(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- **STD_ULOGIC** `\?<=\(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- **STD_ULOGIC** `\?>=\(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- **STD_ULOGIC** `\?<=\(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- **STD_ULOGIC** `\?>=\(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- **STD_ULOGIC** `\?<=\(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- **STD_ULOGIC** `\?>=\(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- **STD_ULOGIC** `\?<=\(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- **STD_ULOGIC** `\?>=\(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- **STD_ULOGIC** `\?<=\(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- **STD_ULOGIC** `\?>=\(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- **STD_ULOGIC** `\?<=\(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- **UNRESOLVED_sfixed** `maximum(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- **UNRESOLVED_sfixed** `maximum(I: in INTEGER, r: in UNRESOLVED_sfixed)`
- **UNRESOLVED_sfixed** `minimum(I: in UNRESOLVED_sfixed, r: in INTEGER)`
- **UNRESOLVED_sfixed** `minimum(I: in INTEGER, r: in UNRESOLVED_sfixed)`
- **BOOLEAN** `"="(I: in UNRESOLVED_sfixed, r: in REAL)`
- **BOOLEAN** `"/="(I: in UNRESOLVED_sfixed, r: in REAL)`
- **BOOLEAN** `">="(I: in UNRESOLVED_sfixed, r: in REAL)`
- **BOOLEAN** `"<="(I: in UNRESOLVED_sfixed, r: in REAL)`
- **BOOLEAN** `"="(I: in REAL, r: in UNRESOLVED_sfixed)`
- **BOOLEAN** `"/="(I: in REAL, r: in UNRESOLVED_sfixed)`
- **BOOLEAN** `">="(I: in REAL, r: in UNRESOLVED_sfixed)`
- **BOOLEAN** `"<="(I: in REAL, r: in UNRESOLVED_sfixed)`
- **BOOLEAN** `"<=(I: in REAL, r: in UNRESOLVED_sfixed)`
- **BOOLEAN** `">=(I: in REAL, r: in UNRESOLVED_sfixed)`
- **STD_ULOGIC** `\?=/\=(I: in UNRESOLVED_sfixed, r: in REAL)`
- **STD_ULOGIC** `\?>=\(I: in UNRESOLVED_sfixed, r: in REAL)`
- **STD_ULOGIC** `\?<=\(I: in UNRESOLVED_sfixed, r: in REAL)`
- **STD_ULOGIC** `\?>=\(I: in UNRESOLVED_sfixed, r: in REAL)`
- **STD_ULOGIC** `\?<=\(I: in UNRESOLVED_sfixed, r: in REAL)`
- **STD_ULOGIC** `\?>=\(I: in UNRESOLVED_sfixed, r: in REAL)`
- **STD_ULOGIC** `\?<=\(I: in REAL, r: in UNRESOLVED_sfixed)`
- **STD_ULOGIC** `\?=/\=(I: in REAL, r: in UNRESOLVED_sfixed)`
- **STD_ULOGIC** `\?>=\(I: in REAL, r: in UNRESOLVED_sfixed)`
- **STD_ULOGIC** `\?<=\(I: in REAL, r: in UNRESOLVED_sfixed)`

- `STD_ULOGIC !?>(I: in REAL, r: in UNRESOLVED_sfixed)`
- `STD_ULOGIC !?<(I: in REAL, r: in UNRESOLVED_sfixed)`
- `UNRESOLVED_sfixed maximum(I: in UNRESOLVED_sfixed, r: in REAL)`
- `UNRESOLVED_sfixed maximum(I: in REAL, r: in UNRESOLVED_sfixed)`
- `UNRESOLVED_sfixed minimum(I: in UNRESOLVED_sfixed, r: in REAL)`
- `UNRESOLVED_sfixed minimum(I: in REAL, r: in UNRESOLVED_sfixed)`
- `UNRESOLVED_ufixed "sll"(ARG: in UNRESOLVED_ufixed, COUNT: in INTEGER)`
- `UNRESOLVED_ufixed "srl"(ARG: in UNRESOLVED_ufixed, COUNT: in INTEGER)`
- `UNRESOLVED_ufixed "rol"(ARG: in UNRESOLVED_ufixed, COUNT: in INTEGER)`
- `UNRESOLVED_ufixed "ror"(ARG: in UNRESOLVED_ufixed, COUNT: in INTEGER)`
- `UNRESOLVED_ufixed "sla"(ARG: in UNRESOLVED_ufixed, COUNT: in INTEGER)`
- `UNRESOLVED_ufixed "sra"(ARG: in UNRESOLVED_ufixed, COUNT: in INTEGER)`
- `UNRESOLVED_sfixed "sll"(ARG: in UNRESOLVED_sfixed, COUNT: in INTEGER)`
- `UNRESOLVED_sfixed "srl"(ARG: in UNRESOLVED_sfixed, COUNT: in INTEGER)`
- `UNRESOLVED_sfixed "rol"(ARG: in UNRESOLVED_sfixed, COUNT: in INTEGER)`
- `UNRESOLVED_sfixed "ror"(ARG: in UNRESOLVED_sfixed, COUNT: in INTEGER)`
- `UNRESOLVED_sfixed "sla"(ARG: in UNRESOLVED_sfixed, COUNT: in INTEGER)`
- `UNRESOLVED_sfixed "sra"(ARG: in UNRESOLVED_sfixed, COUNT: in INTEGER)`
- `UNRESOLVED_ufixed SHIFT_LEFT(ARG: in UNRESOLVED_ufixed, COUNT: in NATURAL)`
- `UNRESOLVED_ufixed SHIFT_RIGHT(ARG: in UNRESOLVED_ufixed, COUNT: in NATURAL)`
- `UNRESOLVED_sfixed SHIFT_LEFT(ARG: in UNRESOLVED_sfixed, COUNT: in NATURAL)`
- `UNRESOLVED_sfixed SHIFT_RIGHT(ARG: in UNRESOLVED_sfixed, COUNT: in NATURAL)`
- `UNRESOLVED_ufixed "not"(I: in UNRESOLVED_ufixed)`
- `UNRESOLVED_ufixed "and"(I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)`
- `UNRESOLVED_ufixed "or"(I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)`
- `UNRESOLVED_ufixed "hand"(I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)`
- `UNRESOLVED_ufixed "nor"(I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)`
- `UNRESOLVED_ufixed "xor"(I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)`
- `UNRESOLVED_ufixed "xnor"(I: in UNRESOLVED_ufixed, r: in UNRESOLVED_ufixed)`
- `UNRESOLVED_sfixed "not"(I: in UNRESOLVED_sfixed)`
- `UNRESOLVED_sfixed "and"(I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)`
- `UNRESOLVED_sfixed "or"(I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)`
- `UNRESOLVED_sfixed "hand"(I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)`
- `UNRESOLVED_sfixed "nor"(I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)`
- `UNRESOLVED_sfixed "xor"(I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)`
- `UNRESOLVED_sfixed "xnor"(I: in UNRESOLVED_sfixed, r: in UNRESOLVED_sfixed)`
- `UNRESOLVED_ufixed "and"(I: in STD_ULOGIC, r: in UNRESOLVED_ufixed)`
- `UNRESOLVED_ufixed "and"(I: in UNRESOLVED_ufixed, r: in STD_ULOGIC)`
- `UNRESOLVED_ufixed "or"(I: in STD_ULOGIC, r: in UNRESOLVED_ufixed)`
- `UNRESOLVED_ufixed "or"(I: in UNRESOLVED_ufixed, r: in STD_ULOGIC)`
- `UNRESOLVED_ufixed "hand"(I: in STD_ULOGIC, r: in UNRESOLVED_ufixed)`
- `UNRESOLVED_ufixed "hand"(I: in UNRESOLVED_ufixed, r: in STD_ULOGIC)`
- `UNRESOLVED_ufixed "nor"(I: in STD_ULOGIC, r: in UNRESOLVED_ufixed)`
- `UNRESOLVED_ufixed "nor"(I: in UNRESOLVED_ufixed, r: in STD_ULOGIC)`
- `UNRESOLVED_ufixed "xor"(I: in STD_ULOGIC, r: in UNRESOLVED_ufixed)`
- `UNRESOLVED_ufixed "xor"(I: in UNRESOLVED_ufixed, r: in STD_ULOGIC)`
- `UNRESOLVED_ufixed "xnor"(I: in STD_ULOGIC, r: in UNRESOLVED_ufixed)`
- `UNRESOLVED_ufixed "xnor"(I: in UNRESOLVED_ufixed, r: in STD_ULOGIC)`
- `UNRESOLVED_sfixed "and"(I: in STD_ULOGIC, r: in UNRESOLVED_sfixed)`
- `UNRESOLVED_sfixed "and"(I: in UNRESOLVED_sfixed, r: in STD_ULOGIC)`
- `UNRESOLVED_sfixed "or"(I: in STD_ULOGIC, r: in UNRESOLVED_sfixed)`
- `UNRESOLVED_sfixed "or"(I: in UNRESOLVED_sfixed, r: in STD_ULOGIC)`
- `UNRESOLVED_sfixed "hand"(I: in STD_ULOGIC, r: in UNRESOLVED_sfixed)`
- `UNRESOLVED_sfixed "hand"(I: in UNRESOLVED_sfixed, r: in STD_ULOGIC)`
- `UNRESOLVED_sfixed "nor"(I: in STD_ULOGIC, r: in UNRESOLVED_sfixed)`

- **UNRESOLVED_sfixed** "nor"(I: in UNRESOLVED_sfixed, r: in STD_ULOGIC)
- **UNRESOLVED_sfixed** "xor"(I: in STD_ULOGIC, r: in UNRESOLVED_sfixed)
- **UNRESOLVED_sfixed** "xor"(I: in UNRESOLVED_sfixed, r: in STD_ULOGIC)
- **UNRESOLVED_sfixed** "xnor"(I: in STD_ULOGIC, r: in UNRESOLVED_sfixed)
- **UNRESOLVED_sfixed** "xnor"(I: in UNRESOLVED_sfixed, r: in STD_ULOGIC)
- **STD_ULOGIC** and_reduce(I: in UNRESOLVED_ufixed)
- **STD_ULOGIC** nand_reduce(I: in UNRESOLVED_ufixed)
- **STD_ULOGIC** or_reduce(I: in UNRESOLVED_ufixed)
- **STD_ULOGIC** nor_reduce(I: in UNRESOLVED_ufixed)
- **STD_ULOGIC** xor_reduce(I: in UNRESOLVED_ufixed)
- **STD_ULOGIC** xnor_reduce(I: in UNRESOLVED_ufixed)
- **STD_ULOGIC** and_reduce(I: in UNRESOLVED_sfixed)
- **STD_ULOGIC** nand_reduce(I: in UNRESOLVED_sfixed)
- **STD_ULOGIC** or_reduce(I: in UNRESOLVED_sfixed)
- **STD_ULOGIC** nor_reduce(I: in UNRESOLVED_sfixed)
- **STD_ULOGIC** xor_reduce(I: in UNRESOLVED_sfixed)
- **STD_ULOGIC** xnor_reduce(I: in UNRESOLVED_sfixed)
- **INTEGER** find_leftmost(arg: in UNRESOLVED_ufixed, y: in STD_ULOGIC)
- **INTEGER** find_leftmost(arg: in UNRESOLVED_sfixed, y: in STD_ULOGIC)
- **INTEGER** find_rightmost(arg: in UNRESOLVED_ufixed, y: in STD_ULOGIC)
- **INTEGER** find_rightmost(arg: in UNRESOLVED_sfixed, y: in STD_ULOGIC)
- **UNRESOLVED_ufixed** resize(
arg: in UNRESOLVED_ufixed
constantleft_index: in INTEGER
constantright_index: in INTEGER
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
)
- **UNRESOLVED_ufixed** resize(
arg: in UNRESOLVED_ufixed
size_res: in UNRESOLVED_ufixed
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
)
- **UNRESOLVED_sfixed** resize(
arg: in UNRESOLVED_sfixed
constantleft_index: in INTEGER
constantright_index: in INTEGER
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
)
- **UNRESOLVED_sfixed** resize(
arg: in UNRESOLVED_sfixed
size_res: in UNRESOLVED_sfixed
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
)
- **UNRESOLVED_ufixed** to_ufixed(
arg: in NATURAL
constantleft_index: in INTEGER
constantright_index: in INTEGER 0
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
)

- **UNRESOLVED_ufixed to_ufixed(**
 arg: in NATURAL
 size_res: in UNRESOLVED_ufixed
 constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
 constantround_style: in fixed_round_style_type fixed_round_style
)
- **UNRESOLVED_ufixed to_ufixed(**
 arg: in REAL
 constantleft_index: in INTEGER
 constantright_index: in INTEGER
 constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
 constantround_style: in fixed_round_style_type fixed_round_style
 constantguard_bits: in NATURAL fixed_guard_bits
)
- **UNRESOLVED_ufixed to_ufixed(**
 arg: in REAL
 size_res: in UNRESOLVED_ufixed
 constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
 constantround_style: in fixed_round_style_type fixed_round_style
 constantguard_bits: in NATURAL fixed_guard_bits
)
- **UNRESOLVED_ufixed to_ufixed(**
 arg: in UNSIGNED
 constantleft_index: in INTEGER
 constantright_index: in INTEGER 0
 constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
 constantround_style: in fixed_round_style_type fixed_round_style
)
- **UNRESOLVED_ufixed to_ufixed(**
 arg: in UNSIGNED
 size_res: in UNRESOLVED_ufixed
 constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
 constantround_style: in fixed_round_style_type fixed_round_style
)
- **UNRESOLVED_ufixed to_ufixed(arg: in UNSIGNED)**
- **UNSIGNED to_unsigned(**
 arg: in UNRESOLVED_ufixed
 constantsize: in NATURAL
 constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
 constantround_style: in fixed_round_style_type fixed_round_style
)
- **UNSIGNED to_unsigned(**
 arg: in UNRESOLVED_ufixed
 size_res: in UNSIGNED
 constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
 constantround_style: in fixed_round_style_type fixed_round_style
)
- **REAL to_real(arg: in UNRESOLVED_ufixed)**
- **NATURAL to_integer(**
 arg: in UNRESOLVED_ufixed
 constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
 constantround_style: in fixed_round_style_type fixed_round_style
)
- **UNRESOLVED_sfixed to_sfixed(**

```
arg: in INTEGER
constantleft_index: in INTEGER
constantright_index: in INTEGER 0
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
)
• UNRESOLVED_sfixed to_sfixed(
arg: in INTEGER
size_res: in UNRESOLVED_sfixed
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
)
• UNRESOLVED_sfixed to_sfixed(
arg: in REAL
constantleft_index: in INTEGER
constantright_index: in INTEGER
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
constantguard_bits: in NATURAL fixed_guard_bits
)
• UNRESOLVED_sfixed to_sfixed(
arg: in REAL
size_res: in UNRESOLVED_sfixed
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
constantguard_bits: in NATURAL fixed_guard_bits
)
• UNRESOLVED_sfixed to_sfixed(
arg: in SIGNED
constantleft_index: in INTEGER
constantright_index: in INTEGER 0
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
)
• UNRESOLVED_sfixed to_sfixed(
arg: in SIGNED
size_res: in UNRESOLVED_sfixed
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
)
• UNRESOLVED_sfixed to_sfixed( arg: in SIGNED)
• UNRESOLVED_sfixed to_sfixed( arg: in UNRESOLVED_ufixed)
• SIGNED to_signed(
arg: in UNRESOLVED_sfixed
constantsize: in NATURAL
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
)
• SIGNED to_signed(
arg: in UNRESOLVED_sfixed
size_res: in SIGNED
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
)
• REAL to_real( arg: in UNRESOLVED_sfixed)
• INTEGER to_integer(
```

```

arg: in UNRESOLVED_sfixed
constantoverflow_style: in fixed_overflow_style_type fixed_overflow_style
constantround_style: in fixed_round_style_type fixed_round_style
)
• INTEGER ufixed_high(
left_index: in INTEGER
right_index: in INTEGER
operation: in CHARACTER 'X'
left_index2: in INTEGER 0
right_index2: in INTEGER 0
)
• INTEGER ufixed_low(
left_index: in INTEGER
right_index: in INTEGER
operation: in CHARACTER 'X'
left_index2: in INTEGER 0
right_index2: in INTEGER 0
)
• INTEGER sfixed_high(
left_index: in INTEGER
right_index: in INTEGER
operation: in CHARACTER 'X'
left_index2: in INTEGER 0
right_index2: in INTEGER 0
)
• INTEGER sfixed_low(
left_index: in INTEGER
right_index: in INTEGER
operation: in CHARACTER 'X'
left_index2: in INTEGER 0
right_index2: in INTEGER 0
)
• INTEGER ufixed_high(
size_res: in UNRESOLVED_ufixed
operation: in CHARACTER 'X'
size_res2: in UNRESOLVED_ufixed
)
• INTEGER ufixed_low(
size_res: in UNRESOLVED_ufixed
operation: in CHARACTER 'X'
size_res2: in UNRESOLVED_ufixed
)
• INTEGER sfixed_high(
size_res: in UNRESOLVED_sfixed
operation: in CHARACTER 'X'
size_res2: in UNRESOLVED_sfixed
)
• INTEGER sfixed_low(
size_res: in UNRESOLVED_sfixed
operation: in CHARACTER 'X'
size_res2: in UNRESOLVED_sfixed
)
• UNRESOLVED_ufixed saturate( constantleft_index: in INTEGER, constantright_index: in INTEGER)
• UNRESOLVED_sfixed saturate( constantleft_index: in INTEGER, constantright_index: in INTEGER)
• UNRESOLVED_ufixed saturate( size_res: in UNRESOLVED_ufixed)
• UNRESOLVED_sfixed saturate( size_res: in UNRESOLVED_sfixed)
• UNRESOLVED_ufixed to_01( s: in UNRESOLVED_ufixed, constantXMAP: in STD_ULOGIC '0')

```

- **UNRESOLVED_sfixed to_01(s: in UNRESOLVED_sfixed, constantXMAP: in STD_ULOGIC '0')**
- **BOOLEAN Is_X(arg: in UNRESOLVED_ufixed)**
- **BOOLEAN Is_X(arg: in UNRESOLVED_sfixed)**
- **UNRESOLVED_ufixed to_X01(arg: in UNRESOLVED_ufixed)**
- **UNRESOLVED_sfixed to_X01(arg: in UNRESOLVED_sfixed)**
- **UNRESOLVED_ufixed to_X01Z(arg: in UNRESOLVED_ufixed)**
- **UNRESOLVED_sfixed to_X01Z(arg: in UNRESOLVED_sfixed)**
- **UNRESOLVED_ufixed to_UX01(arg: in UNRESOLVED_ufixed)**
- **UNRESOLVED_sfixed to_UX01(arg: in UNRESOLVED_sfixed)**
- **STD_LOGIC_VECTOR to_slv(arg: in UNRESOLVED_ufixed)**
- **STD_LOGIC_VECTOR to_slv(arg: in UNRESOLVED_sfixed)**
- **STD_ULOGIC_VECTOR to_sulv(arg: in UNRESOLVED_ufixed)**
- **STD_ULOGIC_VECTOR to_sulv(arg: in UNRESOLVED_sfixed)**
- **UNRESOLVED_ufixed to_ufixed(arg: in STD_ULOGIC_VECTOR
constantleft_index: in INTEGER
constantright_index: in INTEGER
)**
- **UNRESOLVED_ufixed to_ufixed(arg: in STD_ULOGIC_VECTOR, size_res: in UNRESOLVED_ufixed)**
- **UNRESOLVED_sfixed to_sfixed(arg: in STD_ULOGIC_VECTOR
constantleft_index: in INTEGER
constantright_index: in INTEGER
)**
- **UNRESOLVED_sfixed to_sfixed(arg: in STD_ULOGIC_VECTOR, size_res: in UNRESOLVED_sfixed)**
- **UNRESOLVED_ufixed to_UFix(arg: in STD_ULOGIC_VECTOR
width: in NATURAL
fraction: in NATURAL
)**
- **UNRESOLVED_sfixed to_SFix(arg: in STD_ULOGIC_VECTOR
width: in NATURAL
fraction: in NATURAL
)**
- **INTEGER UFix_high(width: in NATURAL
fraction: in NATURAL
operation: in CHARACTER 'X'
width2: in NATURAL 0
fraction2: in NATURAL 0
)**
- **INTEGER UFix_low(width: in NATURAL
fraction: in NATURAL
operation: in CHARACTER 'X'
width2: in NATURAL 0
fraction2: in NATURAL 0
)**
- **INTEGER SFix_high(width: in NATURAL
fraction: in NATURAL
operation: in CHARACTER 'X'
width2: in NATURAL 0
fraction2: in NATURAL 0
)**

- **INTEGER** SFix_low(
width: in **NATURAL**
fraction: in **NATURAL**
operation: in CHARACTER 'X'
width2: in **NATURAL** 0
fraction2: in **NATURAL** 0
)
- **STRING** to_string(value: in UNRESOLVED_ufixed)
- **STRING** to_ostring(value: in UNRESOLVED_ufixed)
- **STRING** to_hstring(value: in UNRESOLVED_ufixed)
- **STRING** to_string(value: in UNRESOLVED_sfixed)
- **STRING** to_ostring(value: in UNRESOLVED_sfixed)
- **STRING** to_hstring(value: in UNRESOLVED_sfixed)
- **UNRESOLVED_ufixed** from_string(
bstring: in **STRING**
constantleft_index: in **INTEGER**
constantright_index: in **INTEGER**
)
- **UNRESOLVED_ufixed** from_ostring(
ostring: in **STRING**
constantleft_index: in **INTEGER**
constantright_index: in **INTEGER**
)
- **UNRESOLVED_ufixed** from_hstring(
hstring: in **STRING**
constantleft_index: in **INTEGER**
constantright_index: in **INTEGER**
)
- **UNRESOLVED_sfixed** from_string(
bstring: in **STRING**
constantleft_index: in **INTEGER**
constantright_index: in **INTEGER**
)
- **UNRESOLVED_sfixed** from_ostring(
ostring: in **STRING**
constantleft_index: in **INTEGER**
constantright_index: in **INTEGER**
)
- **UNRESOLVED_sfixed** from_hstring(
hstring: in **STRING**
constantleft_index: in **INTEGER**
constantright_index: in **INTEGER**
)
- **UNRESOLVED_ufixed** from_string(bstring: in **STRING**, size_res: in UNRESOLVED_ufixed)
- **UNRESOLVED_ufixed** from_ostring(ostring: in **STRING**, size_res: in UNRESOLVED_ufixed)
- **UNRESOLVED_ufixed** from_hstring(hstring: in **STRING**, size_res: in UNRESOLVED_ufixed)
- **UNRESOLVED_sfixed** from_string(bstring: in **STRING**, size_res: in UNRESOLVED_sfixed)
- **UNRESOLVED_sfixed** from_ostring(ostring: in **STRING**, size_res: in UNRESOLVED_sfixed)
- **UNRESOLVED_sfixed** from_hstring(hstring: in **STRING**, size_res: in UNRESOLVED_sfixed)
- **UNRESOLVED_ufixed** from_string(bstring: in **STRING**)
- **UNRESOLVED_ufixed** from_ostring(ostring: in **STRING**)
- **UNRESOLVED_ufixed** from_hstring(hstring: in **STRING**)
- **UNRESOLVED_sfixed** from_string(bstring: in **STRING**)
- **UNRESOLVED_sfixed** from_ostring(ostring: in **STRING**)
- **UNRESOLVED_sfixed** from_hstring(hstring: in **STRING**)

- **UNRESOLVED_ufixed to_ufixed(**
arg: **in STD_LOGIC_VECTOR**
constantleft_index: in INTEGER
constantright_index: in INTEGER
)
- **UNRESOLVED_ufixed to_ufixed(arg: in STD_LOGIC_VECTOR, size_res: in UNRESOLVED_ufixed)**
- **UNRESOLVED_sfixed to_sfixed(**
arg: **in STD_LOGIC_VECTOR**
constantleft_index: in INTEGER
constantright_index: in INTEGER
)
- **UNRESOLVED_sfixed to_sfixed(arg: in STD_LOGIC_VECTOR, size_res: in UNRESOLVED_sfixed)**
- **UNRESOLVED_ufixed to_UFix(**
arg: **in STD_LOGIC_VECTOR**
width: **in NATURAL**
fraction: **in NATURAL**
)
- **UNRESOLVED_sfixed to_SFix(**
arg: **in STD_LOGIC_VECTOR**
width: **in NATURAL**
fraction: **in NATURAL**
)

Procedures

- **add_carry(**
L: **inUNRESOLVED_ufixed**
R: **inUNRESOLVED_ufixed**
c_in: **inSTD_ULOGIC**
result: **outUNRESOLVED_ufixed**
c_out: **outSTD_ULOGIC**
)
- **add_carry(**
L: **inUNRESOLVED_sfixed**
R: **inUNRESOLVED_sfixed**
c_in: **inSTD_ULOGIC**
result: **outUNRESOLVED_sfixed**
c_out: **outSTD_ULOGIC**
)
- **WRITE(**
L: **inoutLINE**
VALUE: **inUNRESOLVED_ufixed**
JUSTIFIED: **inSIDERight**
FIELD: **inWIDTH 0**
)
- **WRITE(**
L: **inoutLINE**
VALUE: **inUNRESOLVED_sfixed**
JUSTIFIED: **inSIDERight**
FIELD: **inWIDTH 0**
)
- **READ(L: inoutLINE, VALUE: outUNRESOLVED_ufixed)**
- **READ(**
L: **inoutLINE**
VALUE: **outUNRESOLVED_ufixed**
GOOD: **outBOOLEAN**
)

- READ(L: **inoutLINE**,**VALUE**: **outUNRESOLVED_sfixed**)
- READ(

L: **inoutLINE**

VALUE: **outUNRESOLVED_sfixed**

GOOD: **outBOOLEAN**

)
- OWRITE(

L: **inoutLINE**

VALUE: **inUNRESOLVED_ufixed**

JUSTIFIED: **inSIDErigh**

FIELD: **inWIDTH 0**

)
- OWRITE(

L: **inoutLINE**

VALUE: **inUNRESOLVED_sfixed**

JUSTIFIED: **inSIDErigh**

FIELD: **inWIDTH 0**

)
- OREAD(L: **inoutLINE**,**VALUE**: **outUNRESOLVED_ufixed**)
- OREAD(

L: **inoutLINE**

VALUE: **outUNRESOLVED_ufixed**

GOOD: **outBOOLEAN**

)
- OREAD(L: **inoutLINE**,**VALUE**: **outUNRESOLVED_sfixed**)
- OREAD(

L: **inoutLINE**

VALUE: **outUNRESOLVED_sfixed**

GOOD: **outBOOLEAN**

)
- HWRITE(

L: **inoutLINE**

VALUE: **inUNRESOLVED_ufixed**

JUSTIFIED: **inSIDErigh**

FIELD: **inWIDTH 0**

)
- HWRITE(

L: **inoutLINE**

VALUE: **inUNRESOLVED_sfixed**

JUSTIFIED: **inSIDErigh**

FIELD: **inWIDTH 0**

)
- HREAD(L: **inoutLINE**,**VALUE**: **outUNRESOLVED_ufixed**)
- HREAD(

L: **inoutLINE**

VALUE: **outUNRESOLVED_ufixed**

GOOD: **outBOOLEAN**

)
- HREAD(L: **inoutLINE**,**VALUE**: **outUNRESOLVED_sfixed**)
- HREAD(

L: **inoutLINE**

VALUE: **outUNRESOLVED_sfixed**

GOOD: **outBOOLEAN**

)

Libraries

- IEEE

- IEEE_PROPOSED

Use Clauses

- TEXTIO
- STD_LOGIC_1164
- NUMERIC_STD
- fixed_float_types

Constants

- fixed_round_style **fixed_round_style_type** :=fixed_round
- fixed_overflow_style **fixed_overflow_style_type** :=fixed_saturate
- fixed_guard_bits **NATURAL**:= 3
- no_warning **BOOLEAN**:=(false)

Types

- **UNRESOLVED_ufixedarray**(**INTEGERrange**<>)ofSTD_ULOGIC
- **UNRESOLVED_sfixedarray**(**INTEGERrange**<>)ofSTD_ULOGIC

Subtypes

- U_ufixed **UNRESOLVED_ufixed**
- U_sfixed **UNRESOLVED_sfixed**
- ufixed **UNRESOLVED_ufixed**
- sfixed **UNRESOLVED_sfixed**

Aliases

- to_StdLogicVector **isto_slv[UNRESOLVED_ufixedreturnSTD_LOGIC_VECTOR]**
- to_Std_Logic_Vector **isto_slv[UNRESOLVED_ufixedreturnSTD_LOGIC_VECTOR]**
- to_StdLogicVector **isto_slv[UNRESOLVED_sfixedreturnSTD_LOGIC_VECTOR]**
- to_Std_Logic_Vector **isto_slv[UNRESOLVED_sfixedreturnSTD_LOGIC_VECTOR]**
- to_StdULogicVector **isto_suv[UNRESOLVED_ufixedreturnSTD_ULOGIC_VECTOR]**
- to_Std_ULogic_Vector **isto_suv[UNRESOLVED_ufixedreturnSTD_ULOGIC_VECTOR]**
- to_StdULogicVector **isto_suv[UNRESOLVED_sfixedreturnSTD_ULOGIC_VECTOR]**
- to_Std_ULogic_Vector **isto_suv[UNRESOLVED_sfixedreturnSTD_ULOGIC_VECTOR]**
- bwrite **isWRITE[LINE,UNRESOLVED_ufixed ,SIDE,width]**
- bwrite **isWRITE[LINE,UNRESOLVED_sfixed ,SIDE,width]**
- bread **isREAD[LINE,UNRESOLVED_ufixed]**
- bread **isREAD[LINE,UNRESOLVED_ufixed ,BOOLEAN]**
- bread **isREAD[LINE,UNRESOLVED_sfixed]**
- bread **isREAD[LINE,UNRESOLVED_sfixed ,BOOLEAN]**
- BINARY_WRITE **isWRITE[LINE,UNRESOLVED_ufixed ,SIDE,width]**
- BINARY_WRITE **isWRITE[LINE,UNRESOLVED_sfixed ,SIDE,width]**
- BINARY_READ **isREAD[LINE,UNRESOLVED_ufixed ,BOOLEAN]**
- BINARY_READ **isREAD[LINE,UNRESOLVED_ufixed]**
- BINARY_READ **isREAD[LINE,UNRESOLVED_sfixed ,BOOLEAN]**
- BINARY_READ **isREAD[LINE,UNRESOLVED_sfixed]**
- OCTAL_READ **isOREAD[LINE,UNRESOLVED_ufixed ,BOOLEAN]**
- OCTAL_READ **isOREAD[LINE,UNRESOLVED_ufixed]**

- OCTAL_READ `isOREAD[LINE,UNRESOLVED_sfixed ,BOOLEAN]`
- OCTAL_READ `isOREAD[LINE,UNRESOLVED_sfixed]`
- OCTAL_WRITE `isOWRITE[LINE,UNRESOLVED_ufixed ,SIDE,WIDTH]`
- OCTAL_WRITE `isOWRITE[LINE,UNRESOLVED_sfixed ,SIDE,WIDTH]`
- HEX_READ `isHREAD[LINE,UNRESOLVED_ufixed ,BOOLEAN]`
- HEX_READ `isHREAD[LINE,UNRESOLVED_sfixed ,BOOLEAN]`
- HEX_READ `isHREAD[LINE,UNRESOLVED_ufixed]`
- HEX_READ `isHREAD[LINE,UNRESOLVED_sfixed]`
- HEX_WRITE `isHWRITE[LINE,UNRESOLVED_ufixed ,SIDE,WIDTH]`
- HEX_WRITE `isHWRITE[LINE,UNRESOLVED_sfixed ,SIDE,WIDTH]`
- `to_bstring isto_string[UNRESOLVED_ufixedreturnSTRING]`
- `TO_BINARY_STRING isTO_STRING[UNRESOLVED_ufixedreturnSTRING]`
- `TO_OCTAL_STRING isTO_OSTRING[UNRESOLVED_ufixedreturnSTRING]`
- `TO_HEX_STRING isTO_HSTRING[UNRESOLVED_ufixedreturnSTRING]`
- `to_bstring isto_string[UNRESOLVED_sfixedreturnSTRING]`
- `TO_BINARY_STRING isTO_STRING[UNRESOLVED_sfixedreturnSTRING]`
- `TO_OCTAL_STRING isTO_OSTRING[UNRESOLVED_sfixedreturnSTRING]`
- `TO_HEX_STRING isTO_HSTRING[UNRESOLVED_sfixedreturnSTRING]`
- `from_bstring isfrom_string[STRING,INTEGER,INTEGERreturnUNRESOLVED_ufixed]`
- `from_binary_string isfrom_string[STRING,INTEGER,INTEGERreturnUNRESOLVED_ufixed]`
- `from_octal_string isfrom_ostring[STRING,INTEGER,INTEGERreturnUNRESOLVED_ufixed]`
- `from_hex_string isfrom_hstring[STRING,INTEGER,INTEGERreturnUNRESOLVED_ufixed]`
- `from_bstring isfrom_string[STRING,INTEGER,INTEGERreturnUNRESOLVED_sfixed]`
- `from_binary_string isfrom_string[STRING,INTEGER,INTEGERreturnUNRESOLVED_sfixed]`
- `from_octal_string isfrom_ostring[STRING,INTEGER,INTEGERreturnUNRESOLVED_sfixed]`
- `from_hex_string isfrom_hstring[STRING,INTEGER,INTEGERreturnUNRESOLVED_sfixed]`
- `from_bstring isfrom_string[STRING,UNRESOLVED_ufixedreturnUNRESOLVED_ufixed]`
- `from_binary_string isfrom_string[STRING,UNRESOLVED_ufixedreturnUNRESOLVED_ufixed]`
- `from_octal_string isfrom_ostring[STRING,UNRESOLVED_ufixedreturnUNRESOLVED_ufixed]`
- `from_hex_string isfrom_hstring[STRING,UNRESOLVED_ufixedreturnUNRESOLVED_ufixed]`
- `from_bstring isfrom_string[STRING,UNRESOLVED_sfixedreturnUNRESOLVED_sfixed]`
- `from_binary_string isfrom_string[STRING,UNRESOLVED_sfixedreturnUNRESOLVED_sfixed]`
- `from_octal_string isfrom_ostring[STRING,UNRESOLVED_sfixedreturnUNRESOLVED_sfixed]`
- `from_hex_string isfrom_hstring[STRING,UNRESOLVED_sfixedreturnUNRESOLVED_sfixed]`
- `from_bstring isfrom_string[STRINGreturnUNRESOLVED_ufixed]`
- `from_binary_string isfrom_string[STRINGreturnUNRESOLVED_ufixed]`
- `from_octal_string isfrom_ostring[STRINGreturnUNRESOLVED_ufixed]`
- `from_hex_string isfrom_hstring[STRINGreturnUNRESOLVED_ufixed]`
- `from_bstring isfrom_string[STRINGreturnUNRESOLVED_sfixed]`
- `from_binary_string isfrom_string[STRINGreturnUNRESOLVED_sfixed]`
- `from_octal_string isfrom_ostring[STRINGreturnUNRESOLVED_sfixed]`
- `from_hex_string isfrom_hstring[STRINGreturnUNRESOLVED_sfixed]`

6.17.1 Detailed Description

Definition at line 25 of file `fixed_pkg_c.vhdl`.

6.17.2 Member Function Documentation

6.17.2.1 UNRESOLVED_ufixed "*"(`lin UNRESOLVED_ufixed , rin UNRESOLVED_ufixed`) [Function]

Definition at line 82 of file `fixed_pkg_c.vhdl`.

6.17.2.2 **UNRESOLVED_sfixed** "*"(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 85 of file [fixed_pkg_c.vhdl](#).

6.17.2.3 **UNRESOLVED_ufixed** "*"(*lin* UNRESOLVED_ufixed , *rin* REAL) [Function]

Definition at line 146 of file [fixed_pkg_c.vhdl](#).

6.17.2.4 **UNRESOLVED_ufixed** "*"(*lin* REAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 149 of file [fixed_pkg_c.vhdl](#).

6.17.2.5 **UNRESOLVED_ufixed** "*"(*lin* UNRESOLVED_ufixed , *rin* NATURAL) [Function]

Definition at line 152 of file [fixed_pkg_c.vhdl](#).

6.17.2.6 **UNRESOLVED_ufixed** "*"(*lin* NATURAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 155 of file [fixed_pkg_c.vhdl](#).

6.17.2.7 **UNRESOLVED_sfixed** "*"(*lin* UNRESOLVED_sfixed , *rin* REAL) [Function]

Definition at line 218 of file [fixed_pkg_c.vhdl](#).

6.17.2.8 **UNRESOLVED_sfixed** "*"(*lin* REAL , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 221 of file [fixed_pkg_c.vhdl](#).

6.17.2.9 **UNRESOLVED_sfixed** "*"(*lin* UNRESOLVED_sfixed , *rin* INTEGER) [Function]

Definition at line 224 of file [fixed_pkg_c.vhdl](#).

6.17.2.10 **UNRESOLVED_sfixed** "+"(*lin* INTEGER , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 227 of file [fixed_pkg_c.vhdl](#).

6.17.2.11 **UNRESOLVED_ufixed** "+"(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 65 of file [fixed_pkg_c.vhdl](#).

6.17.2.12 **UNRESOLVED_sfixed** "+"(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 69 of file [fixed_pkg_c.vhdl](#).

6.17.2.13 **UNRESOLVED_ufixed** "+"(*lin* UNRESOLVED_ufixed , *rin* REAL) [Function]

Definition at line 122 of file [fixed_pkg_c.vhdl](#).

6.17.2.14 **UNRESOLVED_ufixed "+"(*lin* REAL , *rin* UNRESOLVED_ufixed)** [Function]

Definition at line 125 of file [fixed_pkg_c.vhdl](#).

6.17.2.15 **UNRESOLVED_ufixed "+"(*lin* UNRESOLVED_ufixed , *rin* NATURAL)** [Function]

Definition at line 128 of file [fixed_pkg_c.vhdl](#).

6.17.2.16 **UNRESOLVED_ufixed "+"(*lin* NATURAL , *rin* UNRESOLVED_ufixed)** [Function]

Definition at line 131 of file [fixed_pkg_c.vhdl](#).

6.17.2.17 **UNRESOLVED_sfixed "+"(*lin* UNRESOLVED_sfixed , *rin* REAL)** [Function]

Definition at line 194 of file [fixed_pkg_c.vhdl](#).

6.17.2.18 **UNRESOLVED_sfixed "+"(*lin* REAL , *rin* UNRESOLVED_sfixed)** [Function]

Definition at line 197 of file [fixed_pkg_c.vhdl](#).

6.17.2.19 **UNRESOLVED_sfixed "+"(*lin* UNRESOLVED_sfixed , *rin* INTEGER)** [Function]

Definition at line 200 of file [fixed_pkg_c.vhdl](#).

6.17.2.20 **UNRESOLVED_sfixed "+"(*lin* INTEGER , *rin* UNRESOLVED_sfixed)** [Function]

Definition at line 203 of file [fixed_pkg_c.vhdl](#).

6.17.2.21 **UNRESOLVED_sfixed "-"(*argin* UNRESOLVED_sfixed)** [Function]

Definition at line 60 of file [fixed_pkg_c.vhdl](#).

6.17.2.22 **UNRESOLVED_ufixed "-"(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed)** [Function]

Definition at line 74 of file [fixed_pkg_c.vhdl](#).

6.17.2.23 **UNRESOLVED_sfixed "-"(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed)** [Function]

Definition at line 78 of file [fixed_pkg_c.vhdl](#).

6.17.2.24 **UNRESOLVED_ufixed "-"(*lin* UNRESOLVED_ufixed , *rin* REAL)** [Function]

Definition at line 134 of file [fixed_pkg_c.vhdl](#).

6.17.2.25 **UNRESOLVED_ufixed "-"(*lin* REAL , *rin* UNRESOLVED_ufixed)** [Function]

Definition at line 137 of file [fixed_pkg_c.vhdl](#).

6.17.2.26 **UNRESOLVED_ufixed** "-"(*lin* UNRESOLVED_ufixed , *rin* NATURAL) [Function]

Definition at line 140 of file [fixed_pkg_c.vhdl](#).

6.17.2.27 **UNRESOLVED_ufixed** "-"(*lin* NATURAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 143 of file [fixed_pkg_c.vhdl](#).

6.17.2.28 **UNRESOLVED_sfixed** "-"(*lin* UNRESOLVED_sfixed , *rin* REAL) [Function]

Definition at line 206 of file [fixed_pkg_c.vhdl](#).

6.17.2.29 **UNRESOLVED_sfixed** "-"(*lin* REAL , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 209 of file [fixed_pkg_c.vhdl](#).

6.17.2.30 **UNRESOLVED_sfixed** "-"(*lin* UNRESOLVED_sfixed , *rin* INTEGER) [Function]

Definition at line 212 of file [fixed_pkg_c.vhdl](#).

6.17.2.31 **UNRESOLVED_sfixed** "-"(*lin* INTEGER , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 215 of file [fixed_pkg_c.vhdl](#).

6.17.2.32 **UNRESOLVED_ufixed** "/"(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 89 of file [fixed_pkg_c.vhdl](#).

6.17.2.33 **UNRESOLVED_sfixed** "/"(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 92 of file [fixed_pkg_c.vhdl](#).

6.17.2.34 **UNRESOLVED_ufixed** "/"(*lin* UNRESOLVED_ufixed , *rin* REAL) [Function]

Definition at line 158 of file [fixed_pkg_c.vhdl](#).

6.17.2.35 **UNRESOLVED_ufixed** "/"(*lin* REAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 161 of file [fixed_pkg_c.vhdl](#).

6.17.2.36 **UNRESOLVED_ufixed** "/"(*lin* UNRESOLVED_ufixed , *rin* NATURAL) [Function]

Definition at line 164 of file [fixed_pkg_c.vhdl](#).

6.17.2.37 **UNRESOLVED_ufixed** "/"(*lin* NATURAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 167 of file [fixed_pkg_c.vhdl](#).

6.17.2.38 **UNRESOLVED_sfixed** "*l*"(*lin* UNRESOLVED_sfixed , *rin* REAL) [Function]

Definition at line 230 of file [fixed_pkg_c.vhdl](#).

6.17.2.39 **UNRESOLVED_sfixed** "*l*"(*lin* REAL , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 233 of file [fixed_pkg_c.vhdl](#).

6.17.2.40 **UNRESOLVED_sfixed** "*l*"(*lin* UNRESOLVED_sfixed , *rin* INTEGER) [Function]

Definition at line 236 of file [fixed_pkg_c.vhdl](#).

6.17.2.41 **UNRESOLVED_sfixed** "*l*"(*lin* INTEGER , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 239 of file [fixed_pkg_c.vhdl](#).

6.17.2.42 **BOOLEAN** "*l*"(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 371 of file [fixed_pkg_c.vhdl](#).

6.17.2.43 **BOOLEAN** "*l*"(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 372 of file [fixed_pkg_c.vhdl](#).

6.17.2.44 **BOOLEAN** "*l*"(*lin* UNRESOLVED_ufixed , *rin* NATURAL) [Function]

Definition at line 403 of file [fixed_pkg_c.vhdl](#).

6.17.2.45 **BOOLEAN** "*l*"(*lin* NATURAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 410 of file [fixed_pkg_c.vhdl](#).

6.17.2.46 **BOOLEAN** "*l*"(*lin* UNRESOLVED_ufixed , *rin* REAL) [Function]

Definition at line 444 of file [fixed_pkg_c.vhdl](#).

6.17.2.47 **BOOLEAN** "*l*"(*lin* REAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 451 of file [fixed_pkg_c.vhdl](#).

6.17.2.48 **BOOLEAN** "*l*"(*lin* UNRESOLVED_sfixed , *rin* INTEGER) [Function]

Definition at line 481 of file [fixed_pkg_c.vhdl](#).

6.17.2.49 **BOOLEAN** "*l*"(*lin* INTEGER , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 488 of file [fixed_pkg_c.vhdl](#).

6.17.2.50 **BOOLEAN** " $=$ "(*lin* UNRESOLVED_sfixed , *rin* REAL) [Function]

Definition at line 522 of file [fixed_pkg_c.vhdl](#).

6.17.2.51 **BOOLEAN** " $=$ "(*lin* REAL , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 529 of file [fixed_pkg_c.vhdl](#).

6.17.2.52 **BOOLEAN** " $<$ "(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 363 of file [fixed_pkg_c.vhdl](#).

6.17.2.53 **BOOLEAN** " $<$ "(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 364 of file [fixed_pkg_c.vhdl](#).

6.17.2.54 **BOOLEAN** " $<$ "(*lin* UNRESOLVED_ufixed , *rin* NATURAL) [Function]

Definition at line 407 of file [fixed_pkg_c.vhdl](#).

6.17.2.55 **BOOLEAN** " $<$ "(*lin* NATURAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 414 of file [fixed_pkg_c.vhdl](#).

6.17.2.56 **BOOLEAN** " $<$ "(*lin* UNRESOLVED_ufixed , *rin* REAL) [Function]

Definition at line 448 of file [fixed_pkg_c.vhdl](#).

6.17.2.57 **BOOLEAN** " $<$ "(*lin* REAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 455 of file [fixed_pkg_c.vhdl](#).

6.17.2.58 **BOOLEAN** " $<$ "(*lin* UNRESOLVED_sfixed , *rin* INTEGER) [Function]

Definition at line 485 of file [fixed_pkg_c.vhdl](#).

6.17.2.59 **BOOLEAN** " $<$ "(*lin* INTEGER , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 492 of file [fixed_pkg_c.vhdl](#).

6.17.2.60 **BOOLEAN** " $<$ "(*lin* UNRESOLVED_sfixed , *rin* REAL) [Function]

Definition at line 526 of file [fixed_pkg_c.vhdl](#).

6.17.2.61 **BOOLEAN** " $<$ "(*lin* REAL , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 533 of file [fixed_pkg_c.vhdl](#).

6.17.2.62 **BOOLEAN** " \leq "(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 365 of file [fixed_pkg_c.vhdl](#).

6.17.2.63 **BOOLEAN** " \leq "(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 366 of file [fixed_pkg_c.vhdl](#).

6.17.2.64 **BOOLEAN** " \leq "(*lin* UNRESOLVED_ufixed , *rin* NATURAL) [Function]

Definition at line 405 of file [fixed_pkg_c.vhdl](#).

6.17.2.65 **BOOLEAN** " \leq "(*lin* NATURAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 412 of file [fixed_pkg_c.vhdl](#).

6.17.2.66 **BOOLEAN** " \leq "(*lin* UNRESOLVED_ufixed , *rin* REAL) [Function]

Definition at line 446 of file [fixed_pkg_c.vhdl](#).

6.17.2.67 **BOOLEAN** " \leq "(*lin* REAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 453 of file [fixed_pkg_c.vhdl](#).

6.17.2.68 **BOOLEAN** " \leq "(*lin* UNRESOLVED_sfixed , *rin* INTEGER) [Function]

Definition at line 483 of file [fixed_pkg_c.vhdl](#).

6.17.2.69 **BOOLEAN** " \leq "(*lin* INTEGER , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 490 of file [fixed_pkg_c.vhdl](#).

6.17.2.70 **BOOLEAN** " \leq "(*lin* UNRESOLVED_sfixed , *rin* REAL) [Function]

Definition at line 524 of file [fixed_pkg_c.vhdl](#).

6.17.2.71 **BOOLEAN** " \leq "(*lin* REAL , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 531 of file [fixed_pkg_c.vhdl](#).

6.17.2.72 **BOOLEAN** " $=$ "(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 369 of file [fixed_pkg_c.vhdl](#).

6.17.2.73 **BOOLEAN** " $=$ "(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 370 of file [fixed_pkg_c.vhdl](#).

6.17.2.74 **BOOLEAN** "*=*"(*lin* UNRESOLVED_ufixed , *rin* NATURAL) [Function]

Definition at line 402 of file [fixed_pkg_c.vhdl](#).

6.17.2.75 **BOOLEAN** "*=*"(*lin* NATURAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 409 of file [fixed_pkg_c.vhdl](#).

6.17.2.76 **BOOLEAN** "*=*"(*lin* UNRESOLVED_ufixed , *rin* REAL) [Function]

Definition at line 443 of file [fixed_pkg_c.vhdl](#).

6.17.2.77 **BOOLEAN** "*=*"(*lin* REAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 450 of file [fixed_pkg_c.vhdl](#).

6.17.2.78 **BOOLEAN** "*=*"(*lin* UNRESOLVED_sfixed , *rin* INTEGER) [Function]

Definition at line 480 of file [fixed_pkg_c.vhdl](#).

6.17.2.79 **BOOLEAN** "*=*"(*lin* INTEGER , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 487 of file [fixed_pkg_c.vhdl](#).

6.17.2.80 **BOOLEAN** "*=*"(*lin* UNRESOLVED_sfixed , *rin* REAL) [Function]

Definition at line 521 of file [fixed_pkg_c.vhdl](#).

6.17.2.81 **BOOLEAN** "*=*"(*lin* REAL , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 528 of file [fixed_pkg_c.vhdl](#).

6.17.2.82 **BOOLEAN** ">"(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 361 of file [fixed_pkg_c.vhdl](#).

6.17.2.83 **BOOLEAN** ">"(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 362 of file [fixed_pkg_c.vhdl](#).

6.17.2.84 **BOOLEAN** ">"(*lin* UNRESOLVED_ufixed , *rin* NATURAL) [Function]

Definition at line 406 of file [fixed_pkg_c.vhdl](#).

6.17.2.85 **BOOLEAN** ">"(*lin* NATURAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 413 of file [fixed_pkg_c.vhdl](#).

6.17.2.86 **BOOLEAN** ">"(*lin* UNRESOLVED_ufixed , *rin* REAL) [Function]

Definition at line 447 of file [fixed_pkg_c.vhdl](#).

6.17.2.87 **BOOLEAN** ">"(*lin* REAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 454 of file [fixed_pkg_c.vhdl](#).

6.17.2.88 **BOOLEAN** ">"(*lin* UNRESOLVED_sfixed , *rin* INTEGER) [Function]

Definition at line 484 of file [fixed_pkg_c.vhdl](#).

6.17.2.89 **BOOLEAN** ">"(*lin* INTEGER , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 491 of file [fixed_pkg_c.vhdl](#).

6.17.2.90 **BOOLEAN** ">"(*lin* UNRESOLVED_sfixed , *rin* REAL) [Function]

Definition at line 525 of file [fixed_pkg_c.vhdl](#).

6.17.2.91 **BOOLEAN** ">"(*lin* REAL , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 532 of file [fixed_pkg_c.vhdl](#).

6.17.2.92 **BOOLEAN** ">="(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 367 of file [fixed_pkg_c.vhdl](#).

6.17.2.93 **BOOLEAN** ">="(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 368 of file [fixed_pkg_c.vhdl](#).

6.17.2.94 **BOOLEAN** ">="(*lin* UNRESOLVED_ufixed , *rin* NATURAL) [Function]

Definition at line 404 of file [fixed_pkg_c.vhdl](#).

6.17.2.95 **BOOLEAN** ">="(*lin* NATURAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 411 of file [fixed_pkg_c.vhdl](#).

6.17.2.96 **BOOLEAN** ">="(*lin* UNRESOLVED_ufixed , *rin* REAL) [Function]

Definition at line 445 of file [fixed_pkg_c.vhdl](#).

6.17.2.97 **BOOLEAN** ">="(*lin* REAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 452 of file [fixed_pkg_c.vhdl](#).

6.17.2.98 **BOOLEAN** ">="(*lin* UNRESOLVED_sfixed , *rin* INTEGER) [Function]

Definition at line 482 of file [fixed_pkg_c.vhdl](#).

6.17.2.99 **BOOLEAN** ">="(*lin* INTEGER , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 489 of file [fixed_pkg_c.vhdl](#).

6.17.2.100 **BOOLEAN** ">="(*lin* UNRESOLVED_sfixed , *rin* REAL) [Function]

Definition at line 523 of file [fixed_pkg_c.vhdl](#).

6.17.2.101 **BOOLEAN** ">="(*lin* REAL , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 530 of file [fixed_pkg_c.vhdl](#).

6.17.2.102 **UNRESOLVED_sfixed** "abs"(*argin* UNRESOLVED_sfixed) [Function]

Definition at line 56 of file [fixed_pkg_c.vhdl](#).

6.17.2.103 **UNRESOLVED_ufixed** "and"(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 596 of file [fixed_pkg_c.vhdl](#).

6.17.2.104 **UNRESOLVED_sfixed** "and"(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 603 of file [fixed_pkg_c.vhdl](#).

6.17.2.105 **UNRESOLVED_ufixed** "and"(*lin* STD_ULOGIC , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 611 of file [fixed_pkg_c.vhdl](#).

6.17.2.106 **UNRESOLVED_ufixed** "and"(*lin* UNRESOLVED_ufixed , *rin* STD_ULOGIC) [Function]

Definition at line 613 of file [fixed_pkg_c.vhdl](#).

6.17.2.107 **UNRESOLVED_sfixed** "and"(*lin* STD_ULOGIC , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 635 of file [fixed_pkg_c.vhdl](#).

6.17.2.108 **UNRESOLVED_sfixed** "and"(*lin* UNRESOLVED_sfixed , *rin* STD_ULOGIC) [Function]

Definition at line 637 of file [fixed_pkg_c.vhdl](#).

6.17.2.109 **UNRESOLVED_ufixed** "mod"(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 106 of file [fixed_pkg_c.vhdl](#).

6.17.2.110 **UNRESOLVED_sfixed "mod"**(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 110 of file [fixed_pkg_c.vhdl](#).

6.17.2.111 **UNRESOLVED_ufixed "mod"**(*lin* UNRESOLVED_ufixed , *rin* REAL) [Function]

Definition at line 182 of file [fixed_pkg_c.vhdl](#).

6.17.2.112 **UNRESOLVED_ufixed "mod"**(*lin* REAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 185 of file [fixed_pkg_c.vhdl](#).

6.17.2.113 **UNRESOLVED_ufixed "mod"**(*lin* UNRESOLVED_ufixed , *rin* NATURAL) [Function]

Definition at line 188 of file [fixed_pkg_c.vhdl](#).

6.17.2.114 **UNRESOLVED_ufixed "mod"**(*lin* NATURAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 191 of file [fixed_pkg_c.vhdl](#).

6.17.2.115 **UNRESOLVED_sfixed "mod"**(*lin* UNRESOLVED_sfixed , *rin* REAL) [Function]

Definition at line 254 of file [fixed_pkg_c.vhdl](#).

6.17.2.116 **UNRESOLVED_sfixed "mod"**(*lin* REAL , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 257 of file [fixed_pkg_c.vhdl](#).

6.17.2.117 **UNRESOLVED_sfixed "mod"**(*lin* UNRESOLVED_sfixed , *rin* INTEGER) [Function]

Definition at line 260 of file [fixed_pkg_c.vhdl](#).

6.17.2.118 **UNRESOLVED_sfixed "mod"**(*lin* INTEGER , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 263 of file [fixed_pkg_c.vhdl](#).

6.17.2.119 **UNRESOLVED_ufixed "nand"**(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 598 of file [fixed_pkg_c.vhdl](#).

6.17.2.120 **UNRESOLVED_sfixed "nand"**(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 605 of file [fixed_pkg_c.vhdl](#).

6.17.2.121 **UNRESOLVED_ufixed "nand"**(*lin* STD_ULOGIC , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 619 of file [fixed_pkg_c.vhdl](#).

6.17.2.122 **UNRESOLVED_ufixed "nand"**(*lin* UNRESOLVED_ufixed , *rin* STD_ULOGIC) [Function]

Definition at line 621 of file [fixed_pkg_c.vhdl](#).

6.17.2.123 **UNRESOLVED_sfixed "nand"**(*lin* STD_ULOGIC , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 643 of file [fixed_pkg_c.vhdl](#).

6.17.2.124 **UNRESOLVED_sfixed "nand"**(*lin* UNRESOLVED_sfixed , *rin* STD_ULOGIC) [Function]

Definition at line 645 of file [fixed_pkg_c.vhdl](#).

6.17.2.125 **UNRESOLVED_ufixed "nor"**(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 599 of file [fixed_pkg_c.vhdl](#).

6.17.2.126 **UNRESOLVED_sfixed "nor"**(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 606 of file [fixed_pkg_c.vhdl](#).

6.17.2.127 **UNRESOLVED_ufixed "nor"**(*lin* STD_ULOGIC , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 623 of file [fixed_pkg_c.vhdl](#).

6.17.2.128 **UNRESOLVED_ufixed "nor"**(*lin* UNRESOLVED_ufixed , *rin* STD_ULOGIC) [Function]

Definition at line 625 of file [fixed_pkg_c.vhdl](#).

6.17.2.129 **UNRESOLVED_sfixed "nor"**(*lin* STD_ULOGIC , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 647 of file [fixed_pkg_c.vhdl](#).

6.17.2.130 **UNRESOLVED_sfixed "nor"**(*lin* UNRESOLVED_sfixed , *rin* STD_ULOGIC) [Function]

Definition at line 649 of file [fixed_pkg_c.vhdl](#).

6.17.2.131 **UNRESOLVED_ufixed "not"**(*lin* UNRESOLVED_ufixed) [Function]

Definition at line 595 of file [fixed_pkg_c.vhdl](#).

6.17.2.132 **UNRESOLVED_sfixed "not"**(*lin* UNRESOLVED_sfixed) [Function]

Definition at line 602 of file [fixed_pkg_c.vhdl](#).

6.17.2.133 **UNRESOLVED_ufixed "or"**(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 597 of file [fixed_pkg_c.vhdl](#).

6.17.2.134 **UNRESOLVED_sfixed "or"**(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed)
[Function]

Definition at line 604 of file [fixed_pkg_c.vhdl](#).

6.17.2.135 **UNRESOLVED_ufixed "or"**(*lin* STD_ULOGIC , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 615 of file [fixed_pkg_c.vhdl](#).

6.17.2.136 **UNRESOLVED_ufixed "or"**(*lin* UNRESOLVED_ufixed , *rin* STD_ULOGIC) [Function]

Definition at line 617 of file [fixed_pkg_c.vhdl](#).

6.17.2.137 **UNRESOLVED_sfixed "or"**(*lin* STD_ULOGIC , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 639 of file [fixed_pkg_c.vhdl](#).

6.17.2.138 **UNRESOLVED_sfixed "or"**(*lin* UNRESOLVED_sfixed , *rin* STD_ULOGIC) [Function]

Definition at line 641 of file [fixed_pkg_c.vhdl](#).

6.17.2.139 **UNRESOLVED_ufixed "rem"**(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed)
[Function]

Definition at line 97 of file [fixed_pkg_c.vhdl](#).

6.17.2.140 **UNRESOLVED_sfixed "rem"**(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed)
[Function]

Definition at line 101 of file [fixed_pkg_c.vhdl](#).

6.17.2.141 **UNRESOLVED_ufixed "rem"**(*lin* UNRESOLVED_ufixed , *rin* REAL) [Function]

Definition at line 170 of file [fixed_pkg_c.vhdl](#).

6.17.2.142 **UNRESOLVED_ufixed "rem"**(*lin* REAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 173 of file [fixed_pkg_c.vhdl](#).

6.17.2.143 **UNRESOLVED_ufixed "rem"**(*lin* UNRESOLVED_ufixed , *rin* NATURAL) [Function]

Definition at line 176 of file [fixed_pkg_c.vhdl](#).

6.17.2.144 **UNRESOLVED_ufixed "rem"**(*lin* NATURAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 179 of file [fixed_pkg_c.vhdl](#).

6.17.2.145 **UNRESOLVED_sfixed "rem"**(*lin* UNRESOLVED_sfixed , *rin* REAL) [Function]

Definition at line 242 of file [fixed_pkg_c.vhdl](#).

6.17.2.146 **UNRESOLVED_sfixed "rem"**(*lin* **REAL** , *rin* **UNRESOLVED_sfixed**) [Function]

Definition at line 245 of file [fixed_pkg_c.vhdl](#).

6.17.2.147 **UNRESOLVED_sfixed "rem"**(*lin* **UNRESOLVED_sfixed** , *rin* **INTEGER**) [Function]

Definition at line 248 of file [fixed_pkg_c.vhdl](#).

6.17.2.148 **UNRESOLVED_sfixed "rem"**(*lin* **INTEGER** , *rin* **UNRESOLVED_sfixed**) [Function]

Definition at line 251 of file [fixed_pkg_c.vhdl](#).

6.17.2.149 **UNRESOLVED_ufixed "rol"**(*ARGin* **UNRESOLVED_ufixed** , *COUNTin* **INTEGER**) [Function]

Definition at line 562 of file [fixed_pkg_c.vhdl](#).

6.17.2.150 **UNRESOLVED_sfixed "rol"**(*ARGin* **UNRESOLVED_sfixed** , *COUNTin* **INTEGER**) [Function]

Definition at line 574 of file [fixed_pkg_c.vhdl](#).

6.17.2.151 **UNRESOLVED_ufixed "ror"**(*ARGin* **UNRESOLVED_ufixed** , *COUNTin* **INTEGER**) [Function]

Definition at line 564 of file [fixed_pkg_c.vhdl](#).

6.17.2.152 **UNRESOLVED_sfixed "ror"**(*ARGin* **UNRESOLVED_sfixed** , *COUNTin* **INTEGER**) [Function]

Definition at line 576 of file [fixed_pkg_c.vhdl](#).

6.17.2.153 **UNRESOLVED_ufixed "sla"**(*ARGin* **UNRESOLVED_ufixed** , *COUNTin* **INTEGER**) [Function]

Definition at line 566 of file [fixed_pkg_c.vhdl](#).

6.17.2.154 **UNRESOLVED_sfixed "sla"**(*ARGin* **UNRESOLVED_sfixed** , *COUNTin* **INTEGER**) [Function]

Definition at line 578 of file [fixed_pkg_c.vhdl](#).

6.17.2.155 **UNRESOLVED_ufixed "sll"**(*ARGin* **UNRESOLVED_ufixed** , *COUNTin* **INTEGER**) [Function]

Definition at line 558 of file [fixed_pkg_c.vhdl](#).

6.17.2.156 **UNRESOLVED_sfixed "sll"**(*ARGin* **UNRESOLVED_sfixed** , *COUNTin* **INTEGER**) [Function]

Definition at line 570 of file [fixed_pkg_c.vhdl](#).

6.17.2.157 **UNRESOLVED_ufixed "sra"**(*ARGin* **UNRESOLVED_ufixed** , *COUNTin* **INTEGER**) [Function]

Definition at line 568 of file [fixed_pkg_c.vhdl](#).

6.17.2.158 **UNRESOLVED_sfixed "sra"**(*ARGin* UNRESOLVED_sfixed , *COUNTin* INTEGER)
[Function]

Definition at line 580 of file [fixed_pkg_c.vhdl](#).

6.17.2.159 **UNRESOLVED_ufixed "srl"**(*ARGin* UNRESOLVED_ufixed , *COUNTin* INTEGER) [Function]

Definition at line 560 of file [fixed_pkg_c.vhdl](#).

6.17.2.160 **UNRESOLVED_sfixed "srl"**(*ARGin* UNRESOLVED_sfixed , *COUNTin* INTEGER) [Function]

Definition at line 572 of file [fixed_pkg_c.vhdl](#).

6.17.2.161 **UNRESOLVED_ufixed "xnor"**(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed)
[Function]

Definition at line 601 of file [fixed_pkg_c.vhdl](#).

6.17.2.162 **UNRESOLVED_sfixed "xnor"**(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed)
[Function]

Definition at line 608 of file [fixed_pkg_c.vhdl](#).

6.17.2.163 **UNRESOLVED_ufixed "xnor"**(*lin* STD_ULOGIC , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 631 of file [fixed_pkg_c.vhdl](#).

6.17.2.164 **UNRESOLVED_ufixed "xnor"**(*lin* UNRESOLVED_ufixed , *rin* STD_ULOGIC) [Function]

Definition at line 633 of file [fixed_pkg_c.vhdl](#).

6.17.2.165 **UNRESOLVED_sfixed "xnor"**(*lin* STD_ULOGIC , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 655 of file [fixed_pkg_c.vhdl](#).

6.17.2.166 **UNRESOLVED_sfixed "xnor"**(*lin* UNRESOLVED_sfixed , *rin* STD_ULOGIC) [Function]

Definition at line 657 of file [fixed_pkg_c.vhdl](#).

6.17.2.167 **UNRESOLVED_ufixed "xor"**(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed)
[Function]

Definition at line 600 of file [fixed_pkg_c.vhdl](#).

6.17.2.168 **UNRESOLVED_sfixed "xor"**(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed)
[Function]

Definition at line 607 of file [fixed_pkg_c.vhdl](#).

6.17.2.169 **UNRESOLVED_ufixed "xor"(*lin* STD_ULOGIC , *rin* UNRESOLVED_ufixed)** [Function]

Definition at line 627 of file [fixed_pkg_c.vhdl](#).

6.17.2.170 **UNRESOLVED_ufixed "xor"(*lin* UNRESOLVED_ufixed , *rin* STD_ULOGIC)** [Function]

Definition at line 629 of file [fixed_pkg_c.vhdl](#).

6.17.2.171 **UNRESOLVED_sfixed "xor"(*lin* STD_ULOGIC , *rin* UNRESOLVED_sfixed)** [Function]

Definition at line 651 of file [fixed_pkg_c.vhdl](#).

6.17.2.172 **UNRESOLVED_sfixed "xor"(*lin* UNRESOLVED_sfixed , *rin* STD_ULOGIC)** [Function]

Definition at line 653 of file [fixed_pkg_c.vhdl](#).

6.17.2.173 **STD_ULOGIC \?/=\\(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed)** [Function]

Definition at line 375 of file [fixed_pkg_c.vhdl](#).

6.17.2.174 **STD_ULOGIC \?/=\\(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed)** [Function]

Definition at line 381 of file [fixed_pkg_c.vhdl](#).

6.17.2.175 **STD_ULOGIC \?/=\\(*lin* UNRESOLVED_ufixed , *rin* NATURAL)** [Function]

Definition at line 417 of file [fixed_pkg_c.vhdl](#).

6.17.2.176 **STD_ULOGIC \?/=\\(*lin* NATURAL , *rin* UNRESOLVED_ufixed)** [Function]

Definition at line 424 of file [fixed_pkg_c.vhdl](#).

6.17.2.177 **STD_ULOGIC \?/=\\(*lin* UNRESOLVED_ufixed , *rin* REAL)** [Function]

Definition at line 458 of file [fixed_pkg_c.vhdl](#).

6.17.2.178 **STD_ULOGIC \?/=\\(*lin* REAL , *rin* UNRESOLVED_ufixed)** [Function]

Definition at line 465 of file [fixed_pkg_c.vhdl](#).

6.17.2.179 **STD_ULOGIC \?/=\\(*lin* UNRESOLVED_sfixed , *rin* INTEGER)** [Function]

Definition at line 495 of file [fixed_pkg_c.vhdl](#).

6.17.2.180 **STD_ULOGIC \?/=\\(*lin* INTEGER , *rin* UNRESOLVED_sfixed)** [Function]

Definition at line 502 of file [fixed_pkg_c.vhdl](#).

6.17.2.181 **STD_ULOGIC** $\text{?}/=(\text{lin UNRESOLVED_sfixed}, \text{rin REAL})$ [Function]

Definition at line 536 of file [fixed_pkg_c.vhdl](#).

6.17.2.182 **STD_ULOGIC** $\text{?}/=(\text{lin REAL}, \text{rin UNRESOLVED_sfixed})$ [Function]

Definition at line 543 of file [fixed_pkg_c.vhdl](#).

6.17.2.183 **STD_ULOGIC** $\text{?}<=(\text{lin UNRESOLVED_ufixed}, \text{rin UNRESOLVED_ufixed})$ [Function]

Definition at line 379 of file [fixed_pkg_c.vhdl](#).

6.17.2.184 **STD_ULOGIC** $\text{?}<=(\text{lin UNRESOLVED_sfixed}, \text{rin UNRESOLVED_sfixed})$ [Function]

Definition at line 385 of file [fixed_pkg_c.vhdl](#).

6.17.2.185 **STD_ULOGIC** $\text{?}<=(\text{lin UNRESOLVED_ufixed}, \text{rin NATURAL})$ [Function]

Definition at line 419 of file [fixed_pkg_c.vhdl](#).

6.17.2.186 **STD_ULOGIC** $\text{?}<=(\text{lin NATURAL}, \text{rin UNRESOLVED_ufixed})$ [Function]

Definition at line 426 of file [fixed_pkg_c.vhdl](#).

6.17.2.187 **STD_ULOGIC** $\text{?}<=(\text{lin UNRESOLVED_ufixed}, \text{rin REAL})$ [Function]

Definition at line 460 of file [fixed_pkg_c.vhdl](#).

6.17.2.188 **STD_ULOGIC** $\text{?}<=(\text{lin REAL}, \text{rin UNRESOLVED_ufixed})$ [Function]

Definition at line 467 of file [fixed_pkg_c.vhdl](#).

6.17.2.189 **STD_ULOGIC** $\text{?}<=(\text{lin UNRESOLVED_sfixed}, \text{rin INTEGER})$ [Function]

Definition at line 497 of file [fixed_pkg_c.vhdl](#).

6.17.2.190 **STD_ULOGIC** $\text{?}<=(\text{lin INTEGER}, \text{rin UNRESOLVED_sfixed})$ [Function]

Definition at line 504 of file [fixed_pkg_c.vhdl](#).

6.17.2.191 **STD_ULOGIC** $\text{?}<=(\text{lin UNRESOLVED_sfixed}, \text{rin REAL})$ [Function]

Definition at line 538 of file [fixed_pkg_c.vhdl](#).

6.17.2.192 **STD_ULOGIC** $\text{?}<=(\text{lin REAL}, \text{rin UNRESOLVED_sfixed})$ [Function]

Definition at line 545 of file [fixed_pkg_c.vhdl](#).

6.17.2.193 **STD_ULOGIC**!<\(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 378 of file [fixed_pkg_c.vhdl](#).

6.17.2.194 **STD_ULOGIC**!<\(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 384 of file [fixed_pkg_c.vhdl](#).

6.17.2.195 **STD_ULOGIC**!<\(*lin* UNRESOLVED_ufixed , *rin* NATURAL) [Function]

Definition at line 421 of file [fixed_pkg_c.vhdl](#).

6.17.2.196 **STD_ULOGIC**!<\(*lin* NATURAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 428 of file [fixed_pkg_c.vhdl](#).

6.17.2.197 **STD_ULOGIC**!<\(*lin* UNRESOLVED_ufixed , *rin* REAL) [Function]

Definition at line 462 of file [fixed_pkg_c.vhdl](#).

6.17.2.198 **STD_ULOGIC**!<\(*lin* REAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 469 of file [fixed_pkg_c.vhdl](#).

6.17.2.199 **STD_ULOGIC**!<\(*lin* UNRESOLVED_sfixed , *rin* INTEGER) [Function]

Definition at line 499 of file [fixed_pkg_c.vhdl](#).

6.17.2.200 **STD_ULOGIC**!<\(*lin* INTEGER , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 506 of file [fixed_pkg_c.vhdl](#).

6.17.2.201 **STD_ULOGIC**!<\(*lin* UNRESOLVED_sfixed , *rin* REAL) [Function]

Definition at line 540 of file [fixed_pkg_c.vhdl](#).

6.17.2.202 **STD_ULOGIC**!<\(*lin* REAL , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 547 of file [fixed_pkg_c.vhdl](#).

6.17.2.203 **STD_ULOGIC**!?=\<(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 374 of file [fixed_pkg_c.vhdl](#).

6.17.2.204 **STD_ULOGIC**!?=\<(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 380 of file [fixed_pkg_c.vhdl](#).

6.17.2.205 **STD_ULOGIC** $\text{?}=(\text{lin UNRESOLVED_ufixed}, \text{rin NATURAL})$ [Function]

Definition at line 416 of file [fixed_pkg_c.vhdl](#).

6.17.2.206 **STD_ULOGIC** $\text{?}=(\text{lin NATURAL}, \text{rin UNRESOLVED_ufixed})$ [Function]

Definition at line 423 of file [fixed_pkg_c.vhdl](#).

6.17.2.207 **STD_ULOGIC** $\text{?}=(\text{lin UNRESOLVED_ufixed}, \text{rin REAL})$ [Function]

Definition at line 457 of file [fixed_pkg_c.vhdl](#).

6.17.2.208 **STD_ULOGIC** $\text{?}=(\text{lin REAL}, \text{rin UNRESOLVED_ufixed})$ [Function]

Definition at line 464 of file [fixed_pkg_c.vhdl](#).

6.17.2.209 **STD_ULOGIC** $\text{?}=(\text{lin UNRESOLVED_sfixed}, \text{rin INTEGER})$ [Function]

Definition at line 494 of file [fixed_pkg_c.vhdl](#).

6.17.2.210 **STD_ULOGIC** $\text{?}=(\text{lin INTEGER}, \text{rin UNRESOLVED_sfixed})$ [Function]

Definition at line 501 of file [fixed_pkg_c.vhdl](#).

6.17.2.211 **STD_ULOGIC** $\text{?}=(\text{lin UNRESOLVED_sfixed}, \text{rin REAL})$ [Function]

Definition at line 535 of file [fixed_pkg_c.vhdl](#).

6.17.2.212 **STD_ULOGIC** $\text{?}=(\text{lin REAL}, \text{rin UNRESOLVED_sfixed})$ [Function]

Definition at line 542 of file [fixed_pkg_c.vhdl](#).

6.17.2.213 **STD_ULOGIC** $\text{?}>=(\text{lin UNRESOLVED_ufixed}, \text{rin UNRESOLVED_ufixed})$ [Function]

Definition at line 377 of file [fixed_pkg_c.vhdl](#).

6.17.2.214 **STD_ULOGIC** $\text{?}>=(\text{lin UNRESOLVED_sfixed}, \text{rin UNRESOLVED_sfixed})$ [Function]

Definition at line 383 of file [fixed_pkg_c.vhdl](#).

6.17.2.215 **STD_ULOGIC** $\text{?}>=(\text{lin UNRESOLVED_ufixed}, \text{rin NATURAL})$ [Function]

Definition at line 418 of file [fixed_pkg_c.vhdl](#).

6.17.2.216 **STD_ULOGIC** $\text{?}>=(\text{lin NATURAL}, \text{rin UNRESOLVED_ufixed})$ [Function]

Definition at line 425 of file [fixed_pkg_c.vhdl](#).

6.17.2.217 **STD_ULOGIC**!>=(*lin* UNRESOLVED_ufixed , *rin* REAL) [Function]

Definition at line 459 of file [fixed_pkg_c.vhdl](#).

6.17.2.218 **STD_ULOGIC**!>=(*lin* REAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 466 of file [fixed_pkg_c.vhdl](#).

6.17.2.219 **STD_ULOGIC**!>=(*lin* UNRESOLVED_sfixed , *rin* INTEGER) [Function]

Definition at line 496 of file [fixed_pkg_c.vhdl](#).

6.17.2.220 **STD_ULOGIC**!>=(*lin* INTEGER , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 503 of file [fixed_pkg_c.vhdl](#).

6.17.2.221 **STD_ULOGIC**!>=(*lin* UNRESOLVED_sfixed , *rin* REAL) [Function]

Definition at line 537 of file [fixed_pkg_c.vhdl](#).

6.17.2.222 **STD_ULOGIC**!>=(*lin* REAL , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 544 of file [fixed_pkg_c.vhdl](#).

6.17.2.223 **STD_ULOGIC**!>=(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 376 of file [fixed_pkg_c.vhdl](#).

6.17.2.224 **STD_ULOGIC**!>=(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 382 of file [fixed_pkg_c.vhdl](#).

6.17.2.225 **STD_ULOGIC**!>=(*lin* UNRESOLVED_ufixed , *rin* NATURAL) [Function]

Definition at line 420 of file [fixed_pkg_c.vhdl](#).

6.17.2.226 **STD_ULOGIC**!>=(*lin* NATURAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 427 of file [fixed_pkg_c.vhdl](#).

6.17.2.227 **STD_ULOGIC**!>=(*lin* UNRESOLVED_ufixed , *rin* REAL) [Function]

Definition at line 461 of file [fixed_pkg_c.vhdl](#).

6.17.2.228 **STD_ULOGIC**!>=(*lin* REAL , *rin* UNRESOLVED_ufixed) [Function]

Definition at line 468 of file [fixed_pkg_c.vhdl](#).

6.17.2.229 **STD_ULOGIC**!>\(*lin* UNRESOLVED_sfixed , *rin* INTEGER) [Function]

Definition at line 498 of file [fixed_pkg_c.vhdl](#).

6.17.2.230 **STD_ULOGIC**!>\(*lin* INTEGER , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 505 of file [fixed_pkg_c.vhdl](#).

6.17.2.231 **STD_ULOGIC**!>\(*lin* UNRESOLVED_sfixed , *rin* REAL) [Function]

Definition at line 539 of file [fixed_pkg_c.vhdl](#).

6.17.2.232 **STD_ULOGIC**!>\(*lin* REAL , *rin* UNRESOLVED_sfixed) [Function]

Definition at line 546 of file [fixed_pkg_c.vhdl](#).

6.17.2.233 add_carry(*L* inUNRESOLVED_ufixed , *R* inUNRESOLVED_ufixed , *c_in* in**STD_ULOGIC** ,
result outUNRESOLVED_ufixed , *c_out* out**STD_ULOGIC**) [Procedure]

Definition at line 334 of file [fixed_pkg_c.vhdl](#).

6.17.2.234 add_carry(*L* inUNRESOLVED_sfixed , *R* inUNRESOLVED_sfixed , *c_in* in**STD_ULOGIC** ,
result outUNRESOLVED_sfixed , *c_out* out**STD_ULOGIC**) [Procedure]

Definition at line 342 of file [fixed_pkg_c.vhdl](#).

6.17.2.235 **STD_ULOGIC** and_reduce(*lin* UNRESOLVED_ufixed) [Function]

Definition at line 661 of file [fixed_pkg_c.vhdl](#).

6.17.2.236 **STD_ULOGIC** and_reduce(*lin* UNRESOLVED_sfixed) [Function]

Definition at line 667 of file [fixed_pkg_c.vhdl](#).

6.17.2.237 UNRESOLVED_ufixed divide(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed
, *round_style* in fixed_round_style_type fixed_round_style , *guard_bits* in
NATURALfixed_guard_bits) [Function]

Definition at line 267 of file [fixed_pkg_c.vhdl](#).

6.17.2.238 UNRESOLVED_sfixed divide(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed
, *round_style* in fixed_round_style_type fixed_round_style , *guard_bits* in
NATURALfixed_guard_bits) [Function]

Definition at line 275 of file [fixed_pkg_c.vhdl](#).

6.17.2.239 **INTEGER** find_leftmost(*argin* UNRESOLVED_ufixed , *yin* **STD_ULOGIC**) [Function]

Definition at line 675 of file [fixed_pkg_c.vhdl](#).

6.17.2.240 **INTEGER** `find_leftmost(argin UNRESOLVED_sfixed , yin STD_ULOGIC)` [Function]

Definition at line 677 of file [fixed_pkg_c.vhdl](#).

6.17.2.241 **INTEGER** `find_rightmost(argin UNRESOLVED_ufixed , yin STD_ULOGIC)` [Function]

Definition at line 681 of file [fixed_pkg_c.vhdl](#).

6.17.2.242 **INTEGER** `find_rightmost(argin UNRESOLVED_sfixed , yin STD_ULOGIC)` [Function]

Definition at line 683 of file [fixed_pkg_c.vhdl](#).

6.17.2.243 **UNRESOLVED_ufixed** `from_hstring(hstringin STRING , left_indexin INTEGER , right_indexin INTEGER)` [Function]

Definition at line 1289 of file [fixed_pkg_c.vhdl](#).

6.17.2.244 **UNRESOLVED_sfixed** `from_hstring(hstringin STRING , left_indexin INTEGER , right_indexin INTEGER)` [Function]

Definition at line 1315 of file [fixed_pkg_c.vhdl](#).

6.17.2.245 **UNRESOLVED_ufixed** `from_hstring(hstringin STRING , size_resin UNRESOLVED_ufixed)` [Function]

Definition at line 1340 of file [fixed_pkg_c.vhdl](#).

6.17.2.246 **UNRESOLVED_sfixed** `from_hstring(hstringin STRING , size_resin UNRESOLVED_sfixed)` [Function]

Definition at line 1363 of file [fixed_pkg_c.vhdl](#).

6.17.2.247 **UNRESOLVED_ufixed** `from_hstring(hstringin STRING)` [Function]

Definition at line 1392 of file [fixed_pkg_c.vhdl](#).

6.17.2.248 **UNRESOLVED_sfixed** `from_hstring(hstringin STRING)` [Function]

Definition at line 1408 of file [fixed_pkg_c.vhdl](#).

6.17.2.249 **UNRESOLVED_ufixed** `from_ostring(ostringin STRING , left_indexin INTEGER , right_indexin INTEGER)` [Function]

Definition at line 1281 of file [fixed_pkg_c.vhdl](#).

6.17.2.250 **UNRESOLVED_sfixed** `from_ostring(ostringin STRING , left_indexin INTEGER , right_indexin INTEGER)` [Function]

Definition at line 1307 of file [fixed_pkg_c.vhdl](#).

6.17.2.251 **UNRESOLVED_ufixed from_ostring(*ostringin STRING*, *size_resin UNRESOLVED_ufixed*)**
[Function]

Definition at line 1333 of file [fixed_pkg_c.vhdl](#).

6.17.2.252 **UNRESOLVED_sfixed from_ostring(*ostringin STRING*, *size_resin UNRESOLVED_sfixed*)**
[Function]

Definition at line 1356 of file [fixed_pkg_c.vhdl](#).

6.17.2.253 **UNRESOLVED_ufixed from_ostring(*ostringin STRING*)** [Function]

Definition at line 1387 of file [fixed_pkg_c.vhdl](#).

6.17.2.254 **UNRESOLVED_sfixed from_ostring(*ostringin STRING*)** [Function]

Definition at line 1403 of file [fixed_pkg_c.vhdl](#).

6.17.2.255 **UNRESOLVED_ufixed from_string(*bstringin STRING*, *left_indexin INTEGER*, *right_indexin INTEGER*)** [Function]

Definition at line 1267 of file [fixed_pkg_c.vhdl](#).

6.17.2.256 **UNRESOLVED_sfixed from_string(*bstringin STRING*, *left_indexin INTEGER*, *right_indexin INTEGER*)** [Function]

Definition at line 1297 of file [fixed_pkg_c.vhdl](#).

6.17.2.257 **UNRESOLVED_ufixed from_string(*bstringin STRING*, *size_resin UNRESOLVED_ufixed*)**
[Function]

Definition at line 1324 of file [fixed_pkg_c.vhdl](#).

6.17.2.258 **UNRESOLVED_sfixed from_string(*bstringin STRING*, *size_resin UNRESOLVED_sfixed*)**
[Function]

Definition at line 1347 of file [fixed_pkg_c.vhdl](#).

6.17.2.259 **UNRESOLVED_ufixed from_string(*bstringin STRING*)** [Function]

Definition at line 1376 of file [fixed_pkg_c.vhdl](#).

6.17.2.260 **UNRESOLVED_sfixed from_string(*bstringin STRING*)** [Function]

Definition at line 1397 of file [fixed_pkg_c.vhdl](#).

6.17.2.261 **HREAD(*L inoutLINE*, *VALUE outUNRESOLVED_ufixed*)** [Procedure]

Definition at line 1217 of file [fixed_pkg_c.vhdl](#).

6.17.2.262 HREAD(*L inoutLINE*, *VALUE outUNRESOLVED_ufixed* , *GOOD outBOOLEAN*)
[Procedure]

Definition at line 1220 of file [fixed_pkg_c.vhdl](#).

6.17.2.263 HREAD(*L inoutLINE*, *VALUE outUNRESOLVED_sfixed*) [Procedure]

Definition at line 1224 of file [fixed_pkg_c.vhdl](#).

6.17.2.264 HREAD(*L inoutLINE*, *VALUE outUNRESOLVED_sfixed* , *GOOD outBOOLEAN*)
[Procedure]

Definition at line 1227 of file [fixed_pkg_c.vhdl](#).

6.17.2.265 HWRITE(*L inoutLINE*, *VALUE inUNRESOLVED_ufixed* , *JUSTIFIED inSIDEright* , *FIELD inWIDTH 0*) [Procedure]

Definition at line 1204 of file [fixed_pkg_c.vhdl](#).

6.17.2.266 HWRITE(*L inoutLINE*, *VALUE inUNRESOLVED_sfixed* , *JUSTIFIED inSIDEright* , *FIELD inWIDTH 0*) [Procedure]

Definition at line 1211 of file [fixed_pkg_c.vhdl](#).

6.17.2.267 **BOOLEAN** Is_Negative(*argin UNRESOLVED_sfixed*) [Function]

Definition at line 355 of file [fixed_pkg_c.vhdl](#).

6.17.2.268 **BOOLEAN** Is_X(*argin UNRESOLVED_ufixed*) [Function]

Definition at line 1002 of file [fixed_pkg_c.vhdl](#).

6.17.2.269 **BOOLEAN** Is_X(*argin UNRESOLVED_sfixed*) [Function]

Definition at line 1003 of file [fixed_pkg_c.vhdl](#).

6.17.2.270 **UNRESOLVED_ufixed** maximum(*lin UNRESOLVED_ufixed* , *rin UNRESOLVED_ufixed*)
[Function]

Definition at line 392 of file [fixed_pkg_c.vhdl](#).

6.17.2.271 **UNRESOLVED_sfixed** maximum(*lin UNRESOLVED_sfixed* , *rin UNRESOLVED_sfixed*)
[Function]

Definition at line 394 of file [fixed_pkg_c.vhdl](#).

6.17.2.272 **UNRESOLVED_ufixed** maximum(*lin UNRESOLVED_ufixed* , *rin NATURAL*) [Function]

Definition at line 430 of file [fixed_pkg_c.vhdl](#).

6.17.2.273 **UNRESOLVED_ufixed maximum(*lin* NATURAL , *rin* UNRESOLVED_ufixed)** [Function]

Definition at line 434 of file [fixed_pkg_c.vhdl](#).

6.17.2.274 **UNRESOLVED_ufixed maximum(*lin* UNRESOLVED_ufixed , *rin* REAL)** [Function]

Definition at line 471 of file [fixed_pkg_c.vhdl](#).

6.17.2.275 **UNRESOLVED_ufixed maximum(*lin* REAL , *rin* UNRESOLVED_ufixed)** [Function]

Definition at line 472 of file [fixed_pkg_c.vhdl](#).

6.17.2.276 **UNRESOLVED_sfixed maximum(*lin* UNRESOLVED_sfixed , *rin* INTEGER)** [Function]

Definition at line 508 of file [fixed_pkg_c.vhdl](#).

6.17.2.277 **UNRESOLVED_sfixed maximum(*lin* INTEGER , *rin* UNRESOLVED_sfixed)** [Function]

Definition at line 510 of file [fixed_pkg_c.vhdl](#).

6.17.2.278 **UNRESOLVED_sfixed maximum(*lin* UNRESOLVED_sfixed , *rin* REAL)** [Function]

Definition at line 549 of file [fixed_pkg_c.vhdl](#).

6.17.2.279 **UNRESOLVED_sfixed maximum(*lin* REAL , *rin* UNRESOLVED_sfixed)** [Function]

Definition at line 550 of file [fixed_pkg_c.vhdl](#).

6.17.2.280 **UNRESOLVED_ufixed minimum(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed)** [Function]

Definition at line 393 of file [fixed_pkg_c.vhdl](#).

6.17.2.281 **UNRESOLVED_sfixed minimum(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed)** [Function]

Definition at line 395 of file [fixed_pkg_c.vhdl](#).

6.17.2.282 **UNRESOLVED_ufixed minimum(*lin* UNRESOLVED_ufixed , *rin* NATURAL)** [Function]

Definition at line 432 of file [fixed_pkg_c.vhdl](#).

6.17.2.283 **UNRESOLVED_ufixed minimum(*lin* NATURAL , *rin* UNRESOLVED_ufixed)** [Function]

Definition at line 436 of file [fixed_pkg_c.vhdl](#).

6.17.2.284 **UNRESOLVED_ufixed minimum(*lin* UNRESOLVED_ufixed , *rin* REAL)** [Function]

Definition at line 473 of file [fixed_pkg_c.vhdl](#).

6.17.2.285 **UNRESOLVED_ufixed minimum(*lin* REAL , *rin* UNRESOLVED_ufixed)** [Function]

Definition at line 474 of file [fixed_pkg_c.vhdl](#).

6.17.2.286 **UNRESOLVED_sfixed minimum(*lin* UNRESOLVED_sfixed , *rin* INTEGER)** [Function]

Definition at line 512 of file [fixed_pkg_c.vhdl](#).

6.17.2.287 **UNRESOLVED_sfixed minimum(*lin* INTEGER , *rin* UNRESOLVED_sfixed)** [Function]

Definition at line 514 of file [fixed_pkg_c.vhdl](#).

6.17.2.288 **UNRESOLVED_sfixed minimum(*lin* UNRESOLVED_sfixed , *rin* REAL)** [Function]

Definition at line 551 of file [fixed_pkg_c.vhdl](#).

6.17.2.289 **UNRESOLVED_sfixed minimum(*lin* REAL , *rin* UNRESOLVED_sfixed)** [Function]

Definition at line 552 of file [fixed_pkg_c.vhdl](#).

6.17.2.290 **UNRESOLVED_ufixed modulo(*lin* UNRESOLVED_ufixed , *rin* UNRESOLVED_ufixed , *round_style*in fixed_round_style_type fixed_round_style , *guard_bits*in NATURALfixed_guard_bits)** [Function]

Definition at line 316 of file [fixed_pkg_c.vhdl](#).

6.17.2.291 **UNRESOLVED_sfixed modulo(*lin* UNRESOLVED_sfixed , *rin* UNRESOLVED_sfixed , *overflow_style*in fixed_overflow_style_type fixed_overflow_style , *round_style*in fixed_round_style_type fixed_round_style , *guard_bits*in NATURALfixed_guard_bits)** [Function]

Definition at line 324 of file [fixed_pkg_c.vhdl](#).

6.17.2.292 **STD_ULOGIC nand_reduce(*lin* UNRESOLVED_ufixed)** [Function]

Definition at line 662 of file [fixed_pkg_c.vhdl](#).

6.17.2.293 **STD_ULOGIC nand_reduce(*lin* UNRESOLVED_sfixed)** [Function]

Definition at line 668 of file [fixed_pkg_c.vhdl](#).

6.17.2.294 **STD_ULOGIC nor_reduce(*lin* UNRESOLVED_ufixed)** [Function]

Definition at line 664 of file [fixed_pkg_c.vhdl](#).

6.17.2.295 **STD_ULOGIC nor_reduce(*lin* UNRESOLVED_sfixed)** [Function]

Definition at line 670 of file [fixed_pkg_c.vhdl](#).

6.17.2.296 **STD_ULOGIC** **or_reduce(** *lin* **UNRESOLVED_ufixed** **)** [Function]

Definition at line 663 of file [fixed_pkg_c.vhdl](#).

6.17.2.297 **STD_ULOGIC** **or_reduce(** *lin* **UNRESOLVED_sfixed** **)** [Function]

Definition at line 669 of file [fixed_pkg_c.vhdl](#).

6.17.2.298 **OREAD(** *L inoutLINE*, **VALUE** *outUNRESOLVED_ufixed* **)** [Procedure]

Definition at line 1183 of file [fixed_pkg_c.vhdl](#).

6.17.2.299 **OREAD(** *L inoutLINE*, **VALUE** *outUNRESOLVED_ufixed*, **GOOD** *outBOOLEAN* **)**
[Procedure]

Definition at line 1186 of file [fixed_pkg_c.vhdl](#).

6.17.2.300 **OREAD(** *L inoutLINE*, **VALUE** *outUNRESOLVED_sfixed* **)** [Procedure]

Definition at line 1190 of file [fixed_pkg_c.vhdl](#).

6.17.2.301 **OREAD(** *L inoutLINE*, **VALUE** *outUNRESOLVED_sfixed*, **GOOD** *outBOOLEAN* **)**
[Procedure]

Definition at line 1193 of file [fixed_pkg_c.vhdl](#).

6.17.2.302 **OWRITE(** *L inoutLINE*, **VALUE** *inUNRESOLVED_ufixed*, **JUSTIFIED** *inSIDERight*, **FIELD**
inWIDTH *0* **)** [Procedure]

Definition at line 1171 of file [fixed_pkg_c.vhdl](#).

6.17.2.303 **OWRITE(** *L inoutLINE*, **VALUE** *inUNRESOLVED_sfixed*, **JUSTIFIED** *inSIDERight*, **FIELD**
inWIDTH *0* **)** [Procedure]

Definition at line 1177 of file [fixed_pkg_c.vhdl](#).

6.17.2.304 **READ(** *L inoutLINE*, **VALUE** *outUNRESOLVED_ufixed* **)** [Procedure]

Definition at line 1143 of file [fixed_pkg_c.vhdl](#).

6.17.2.305 **READ(** *L inoutLINE*, **VALUE** *outUNRESOLVED_ufixed*, **GOOD** *outBOOLEAN* **)**
[Procedure]

Definition at line 1146 of file [fixed_pkg_c.vhdl](#).

6.17.2.306 **READ(** *L inoutLINE*, **VALUE** *outUNRESOLVED_sfixed* **)** [Procedure]

Definition at line 1150 of file [fixed_pkg_c.vhdl](#).

6.17.2.307 **READ(*L inoutLINE* , *VALUE outUNRESOLVED_sfixed* , *GOOD outBOOLEAN*)**
[Procedure]

Definition at line 1153 of file [fixed_pkg_c.vhdl](#).

6.17.2.308 **UNRESOLVED_ufixed reciprocal(*argin UNRESOLVED_ufixed* , *round_stylein fixed_round_style_type fixed_round_style* , *guard_bitsin NATURALfixed_guard_bits*)**
[Function]

Definition at line 283 of file [fixed_pkg_c.vhdl](#).

6.17.2.309 **UNRESOLVED_sfixed reciprocal(*argin UNRESOLVED_sfixed* , *round_stylein fixed_round_style_type fixed_round_style* , *guard_bitsin NATURALfixed_guard_bits*)**
[Function]

Definition at line 290 of file [fixed_pkg_c.vhdl](#).

6.17.2.310 **UNRESOLVED_ufixed remainder(*lin UNRESOLVED_ufixed* , *rin UNRESOLVED_ufixed* , *round_stylein fixed_round_style_type fixed_round_style* , *guard_bitsin NATURALfixed_guard_bits*)**
[Function]

Definition at line 299 of file [fixed_pkg_c.vhdl](#).

6.17.2.311 **UNRESOLVED_sfixed remainder(*lin UNRESOLVED_sfixed* , *rin UNRESOLVED_sfixed* , *round_stylein fixed_round_style_type fixed_round_style* , *guard_bitsin NATURALfixed_guard_bits*)**
[Function]

Definition at line 307 of file [fixed_pkg_c.vhdl](#).

6.17.2.312 **UNRESOLVED_ufixed resize(*argin UNRESOLVED_ufixed* , *left_indexin INTEGER* , *right_indexin INTEGER* , *overflow_stylein fixed_overflow_style_type fixed_overflow_style* , *round_stylein fixed_round_style_type fixed_round_style*)**
[Function]

Definition at line 700 of file [fixed_pkg_c.vhdl](#).

6.17.2.313 **UNRESOLVED_ufixed resize(*argin UNRESOLVED_ufixed* , *size_resin UNRESOLVED_ufixed* , *overflow_stylein fixed_overflow_style_type fixed_overflow_style* , *round_stylein fixed_round_style_type fixed_round_style*)**
[Function]

Definition at line 710 of file [fixed_pkg_c.vhdl](#).

6.17.2.314 **UNRESOLVED_sfixed resize(*argin UNRESOLVED_sfixed* , *left_indexin INTEGER* , *right_indexin INTEGER* , *overflow_stylein fixed_overflow_style_type fixed_overflow_style* , *round_stylein fixed_round_style_type fixed_round_style*)**
[Function]

Definition at line 719 of file [fixed_pkg_c.vhdl](#).

6.17.2.315 **UNRESOLVED_sfixed resize(*argin UNRESOLVED_sfixed* , *size_resin UNRESOLVED_sfixed* , *overflow_stylein fixed_overflow_style_type fixed_overflow_style* , *round_stylein fixed_round_style_type fixed_round_style*)**
[Function]

Definition at line 727 of file [fixed_pkg_c.vhdl](#).

6.17.2.316 **UNRESOLVED_ufixed** **saturate(** *left_index***in** INTEGER , *right_index***in** INTEGER) [Function]

Definition at line 967 of file [fixed_pkg_c.vhdl](#).

6.17.2.317 **UNRESOLVED_sfixed** **saturate(** *left_index***in** INTEGER , *right_index***in** INTEGER) [Function]

Definition at line 973 of file [fixed_pkg_c.vhdl](#).

6.17.2.318 **UNRESOLVED_ufixed** **saturate(** *size_res***in** UNRESOLVED_ufixed) [Function]

Definition at line 978 of file [fixed_pkg_c.vhdl](#).

6.17.2.319 **UNRESOLVED_sfixed** **saturate(** *size_res***in** UNRESOLVED_sfixed) [Function]

Definition at line 982 of file [fixed_pkg_c.vhdl](#).

6.17.2.320 **UNRESOLVED_ufixed** **scalb(** *y***in** UNRESOLVED_ufixed , *N***in** INTEGER) [Function]

Definition at line 350 of file [fixed_pkg_c.vhdl](#).

6.17.2.321 **UNRESOLVED_ufixed** **scalb(** *y***in** UNRESOLVED_ufixed , *N***in** SIGNED) [Function]

Definition at line 351 of file [fixed_pkg_c.vhdl](#).

6.17.2.322 **UNRESOLVED_sfixed** **scalb(** *y***in** UNRESOLVED_sfixed , *N***in** INTEGER) [Function]

Definition at line 352 of file [fixed_pkg_c.vhdl](#).

6.17.2.323 **UNRESOLVED_sfixed** **scalb(** *y***in** UNRESOLVED_sfixed , *N***in** SIGNED) [Function]

Definition at line 353 of file [fixed_pkg_c.vhdl](#).

6.17.2.324 **INTEGER** **SFix_high(** *width***in** NATURAL , *fraction***in** NATURAL , *operation***in** CHARACTER'X' , *width2***in** NATURAL 0 , *fraction2***in** NATURAL 0) [Function]

Definition at line 1114 of file [fixed_pkg_c.vhdl](#).

6.17.2.325 **INTEGER** **SFix_low(** *width***in** NATURAL , *fraction***in** NATURAL , *operation***in** CHARACTER'X' , *width2***in** NATURAL 0 , *fraction2***in** NATURAL 0) [Function]

Definition at line 1119 of file [fixed_pkg_c.vhdl](#).

6.17.2.326 **INTEGER** **sfixed_high(** *left_index***in** INTEGER , *right_index***in** INTEGER , *operation***in** CHARACTER'X' , *left_index2***in** INTEGER 0 , *right_index2***in** INTEGER 0) [Function]

Definition at line 931 of file [fixed_pkg_c.vhdl](#).

6.17.2.327 **INTEGER** sfixed_high(*size_res1* UNRESOLVED_sfixed , *operation*in **CHARACTER**'X' ,
*size_res2*in UNRESOLVED_sfixed) [Function]

Definition at line 956 of file [fixed_pkg_c.vhdl](#).

6.17.2.328 **INTEGER** sfixed_low(*left_index*in **INTEGER** , *right_index*in **INTEGER** , *operation*in
CHARACTER'X' , *left_index2*in **INTEGER** 0 , *right_index2*in **INTEGER** 0) [Function]

Definition at line 936 of file [fixed_pkg_c.vhdl](#).

6.17.2.329 **INTEGER** sfixed_low(*size_res1* UNRESOLVED_sfixed , *operation*in **CHARACTER**'X' ,
*size_res2*in UNRESOLVED_sfixed) [Function]

Definition at line 961 of file [fixed_pkg_c.vhdl](#).

6.17.2.330 **UNRESOLVED_ufixed** SHIFT_LEFT(*ARG*in **UNRESOLVED_ufixed** , *COUNT*in **NATURAL**)
[Function]

Definition at line 582 of file [fixed_pkg_c.vhdl](#).

6.17.2.331 **UNRESOLVED_sfixed** SHIFT_LEFT(*ARG*in **UNRESOLVED_sfixed** , *COUNT*in **NATURAL**)
[Function]

Definition at line 586 of file [fixed_pkg_c.vhdl](#).

6.17.2.332 **UNRESOLVED_ufixed** SHIFT_RIGHT(*ARG*in **UNRESOLVED_ufixed** , *COUNT*in **NATURAL**)
[Function]

Definition at line 584 of file [fixed_pkg_c.vhdl](#).

6.17.2.333 **UNRESOLVED_sfixed** SHIFT_RIGHT(*ARG*in **UNRESOLVED_sfixed** , *COUNT*in **NATURAL**)
[Function]

Definition at line 588 of file [fixed_pkg_c.vhdl](#).

6.17.2.334 **BOOLEAN** std_match(*l*in **UNRESOLVED_ufixed** , *r*in **UNRESOLVED_ufixed**) [Function]

Definition at line 387 of file [fixed_pkg_c.vhdl](#).

6.17.2.335 **BOOLEAN** std_match(*l*in **UNRESOLVED_sfixed** , *r*in **UNRESOLVED_sfixed**) [Function]

Definition at line 388 of file [fixed_pkg_c.vhdl](#).

6.17.2.336 **UNRESOLVED_ufixed** to_01(*s*in **UNRESOLVED_ufixed** , *XMAP*in **STD_ULOGIC**'0')
[Function]

Definition at line 991 of file [fixed_pkg_c.vhdl](#).

6.17.2.337 **UNRESOLVED_sfixed to_01(*sin* UNRESOLVED_sfixed , *XMAPin* STD_ULOGIC' 0 ')**
 [Function]

Definition at line 997 of file [fixed_pkg_c.vhdl](#).

6.17.2.338 **STRING to_hstring(*valuein* UNRESOLVED_ufixed)** [Function]

Definition at line 1246 of file [fixed_pkg_c.vhdl](#).

6.17.2.339 **STRING to_hstring(*valuein* UNRESOLVED_sfixed)** [Function]

Definition at line 1256 of file [fixed_pkg_c.vhdl](#).

6.17.2.340 **NATURAL to_integer(*argin* UNRESOLVED_ufixed , *overflow_stylein* fixed_overflow_style_type
 fixed_overflow_style , *round_stylein* fixed_round_style_type fixed_round_style)**
 [Function]

Definition at line 817 of file [fixed_pkg_c.vhdl](#).

6.17.2.341 **INTEGER to_integer(*argin* UNRESOLVED_sfixed , *overflow_stylein* fixed_overflow_style_type
 fixed_overflow_style , *round_stylein* fixed_round_style_type fixed_round_style)**
 [Function]

Definition at line 905 of file [fixed_pkg_c.vhdl](#).

6.17.2.342 **STRING to_ostring(*valuein* UNRESOLVED_ufixed)** [Function]

Definition at line 1243 of file [fixed_pkg_c.vhdl](#).

6.17.2.343 **STRING to_ostring(*valuein* UNRESOLVED_sfixed)** [Function]

Definition at line 1253 of file [fixed_pkg_c.vhdl](#).

6.17.2.344 **REAL to_real(*argin* UNRESOLVED_ufixed)** [Function]

Definition at line 812 of file [fixed_pkg_c.vhdl](#).

6.17.2.345 **REAL to_real(*argin* UNRESOLVED_sfixed)** [Function]

Definition at line 900 of file [fixed_pkg_c.vhdl](#).

6.17.2.346 **UNRESOLVED_sfixed to_SFix(*argin* STD_ULOGIC_VECTOR , *widthin* NATURAL , *fractionin*
 NATURAL)** [Function]

Definition at line 1085 of file [fixed_pkg_c.vhdl](#).

6.17.2.347 **UNRESOLVED_sfixed to_SFix(*argin* STD_LOGIC_VECTOR , *widthin* NATURAL , *fractionin*
 NATURAL)** [Function]

Definition at line 1447 of file [fixed_pkg_c.vhdl](#).

6.17.2.348 **UNRESOLVED_sfixed to_sfixed(*argin* INTEGER , *left_indexin* INTEGER , *right_indexin* INTEGER 0 , *overflow_stylein* fixed_overflow_style_type fixed_overflow_style , *round_stylein* fixed_round_style_type fixed_round_style)** [Function]

Definition at line 824 of file [fixed_pkg_c.vhdl](#).

6.17.2.349 **UNRESOLVED_sfixed to_sfixed(*argin* INTEGER , *size_resin* UNRESOLVED_sfixed , *overflow_stylein* fixed_overflow_style_type fixed_overflow_style , *round_stylein* fixed_round_style_type fixed_round_style)** [Function]

Definition at line 832 of file [fixed_pkg_c.vhdl](#).

6.17.2.350 **UNRESOLVED_sfixed to_sfixed(*argin* REAL , *left_indexin* INTEGER , *right_indexin* INTEGER , *overflow_stylein* fixed_overflow_style_type fixed_overflow_style , *round_stylein* fixed_round_style_type fixed_round_style , *guard_bitsin* NATURALfixed_guard_bits)** [Function]

Definition at line 840 of file [fixed_pkg_c.vhdl](#).

6.17.2.351 **UNRESOLVED_sfixed to_sfixed(*argin* REAL , *size_resin* UNRESOLVED_sfixed , *overflow_stylein* fixed_overflow_style_type fixed_overflow_style , *round_stylein* fixed_round_style_type fixed_round_style , *guard_bitsin* NATURALfixed_guard_bits)** [Function]

Definition at line 849 of file [fixed_pkg_c.vhdl](#).

6.17.2.352 **UNRESOLVED_sfixed to_sfixed(*argin* SIGNED , *left_indexin* INTEGER , *right_indexin* INTEGER 0 , *overflow_stylein* fixed_overflow_style_type fixed_overflow_style , *round_stylein* fixed_round_style_type fixed_round_style)** [Function]

Definition at line 858 of file [fixed_pkg_c.vhdl](#).

6.17.2.353 **UNRESOLVED_sfixed to_sfixed(*argin* SIGNED , *size_resin* UNRESOLVED_sfixed , *overflow_stylein* fixed_overflow_style_type fixed_overflow_style , *round_stylein* fixed_round_style_type fixed_round_style)** [Function]

Definition at line 866 of file [fixed_pkg_c.vhdl](#).

6.17.2.354 **UNRESOLVED_sfixed to_sfixed(*argin* SIGNED)** [Function]

Definition at line 874 of file [fixed_pkg_c.vhdl](#).

6.17.2.355 **UNRESOLVED_sfixed to_sfixed(*argin* UNRESOLVED_ufixed)** [Function]

Definition at line 879 of file [fixed_pkg_c.vhdl](#).

6.17.2.356 **UNRESOLVED_sfixed to_sfixed(*argin* STD_ULOGIC_VECTOR , *left_indexin* INTEGER , *right_indexin* INTEGER)** [Function]

Definition at line 1059 of file [fixed_pkg_c.vhdl](#).

6.17.2.357 **UNRESOLVED_sfixed to_sfixed(*argin* STD_ULOGIC_VECTOR , *size_resin* UNRESOLVED_sfixed)** [Function]

Definition at line 1065 of file [fixed_pkg_c.vhdl](#).

6.17.2.358 **UNRESOLVED_sfixed to_sfixed(*argin* STD_LOGIC_VECTOR , *left_indexin* INTEGER , *right_indexin* INTEGER)** [Function]

Definition at line 1428 of file [fixed_pkg_c.vhdl](#).

6.17.2.359 **UNRESOLVED_sfixed to_sfixed(*argin* STD_LOGIC_VECTOR , *size_resin* UNRESOLVED_sfixed)** [Function]

Definition at line 1434 of file [fixed_pkg_c.vhdl](#).

6.17.2.360 **SIGNED to_signed(*argin* UNRESOLVED_sfixed , *sizein* NATURAL , *overflow_stylein* fixed_overflow_style_type fixed_overflow_style , *round_stylein* fixed_round_style_type fixed_round_style)** [Function]

Definition at line 884 of file [fixed_pkg_c.vhdl](#).

6.17.2.361 **SIGNED to_signed(*argin* UNRESOLVED_sfixed , *size_resin* SIGNED , *overflow_stylein* fixed_overflow_style_type fixed_overflow_style , *round_stylein* fixed_round_style_type fixed_round_style)** [Function]

Definition at line 892 of file [fixed_pkg_c.vhdl](#).

6.17.2.362 **STD_LOGIC_VECTOR to_slv(*argin* UNRESOLVED_ufixed)** [Function]

Definition at line 1016 of file [fixed_pkg_c.vhdl](#).

6.17.2.363 **STD_LOGIC_VECTOR to_slv(*argin* UNRESOLVED_sfixed)** [Function]

Definition at line 1024 of file [fixed_pkg_c.vhdl](#).

6.17.2.364 **STRING to_string(*valuein* UNRESOLVED_ufixed)** [Function]

Definition at line 1239 of file [fixed_pkg_c.vhdl](#).

6.17.2.365 **STRING to_string(*valuein* UNRESOLVED_sfixed)** [Function]

Definition at line 1249 of file [fixed_pkg_c.vhdl](#).

6.17.2.366 **STD_ULOGIC_VECTOR to_suv(*argin* UNRESOLVED_ufixed)** [Function]

Definition at line 1032 of file [fixed_pkg_c.vhdl](#).

6.17.2.367 **STD_ULOGIC_VECTOR to_suv(*argin* UNRESOLVED_sfixed)** [Function]

Definition at line 1040 of file [fixed_pkg_c.vhdl](#).

6.17.2.368 **UNRESOLVED_ufixed to_UFix(*argin STD_ULOGIC_VECTOR*, *widthin NATURAL*, *fractionin NATURAL*)** [Function]

Definition at line 1078 of file [fixed_pkg_c.vhdl](#).

6.17.2.369 **UNRESOLVED_ufixed to_UFix(*argin STD_LOGIC_VECTOR*, *widthin NATURAL*, *fractionin NATURAL*)** [Function]

Definition at line 1440 of file [fixed_pkg_c.vhdl](#).

6.17.2.370 **UNRESOLVED_ufixed to_ufixed(*argin NATURAL*, *left_indexin INTEGER*, *right_indexin INTEGER 0*, *overflow_stylein fixed_overflow_style_type fixed_overflow_style*, *round_stylein fixed_round_style_type fixed_round_style*)** [Function]

Definition at line 741 of file [fixed_pkg_c.vhdl](#).

6.17.2.371 **UNRESOLVED_ufixed to_ufixed(*argin NATURAL*, *size_resin UNRESOLVED_ufixed*, *overflow_stylein fixed_overflow_style_type fixed_overflow_style*, *round_stylein fixed_round_style_type fixed_round_style*)** [Function]

Definition at line 749 of file [fixed_pkg_c.vhdl](#).

6.17.2.372 **UNRESOLVED_ufixed to_ufixed(*argin REAL*, *left_indexin INTEGER*, *right_indexin INTEGER*, *overflow_stylein fixed_overflow_style_type fixed_overflow_style*, *round_stylein fixed_round_style_type fixed_round_style*, *guard_bitsin NATURALfixed_guard_bits*)** [Function]

Definition at line 757 of file [fixed_pkg_c.vhdl](#).

6.17.2.373 **UNRESOLVED_ufixed to_ufixed(*argin REAL*, *size_resin UNRESOLVED_ufixed*, *overflow_stylein fixed_overflow_style_type fixed_overflow_style*, *round_stylein fixed_round_style_type fixed_round_style*, *guard_bitsin NATURALfixed_guard_bits*)** [Function]

Definition at line 766 of file [fixed_pkg_c.vhdl](#).

6.17.2.374 **UNRESOLVED_ufixed to_ufixed(*argin UNSIGNED*, *left_indexin INTEGER*, *right_indexin INTEGER 0*, *overflow_stylein fixed_overflow_style_type fixed_overflow_style*, *round_stylein fixed_round_style_type fixed_round_style*)** [Function]

Definition at line 775 of file [fixed_pkg_c.vhdl](#).

6.17.2.375 **UNRESOLVED_ufixed to_ufixed(*argin UNSIGNED*, *size_resin UNRESOLVED_ufixed*, *overflow_stylein fixed_overflow_style_type fixed_overflow_style*, *round_stylein fixed_round_style_type fixed_round_style*)** [Function]

Definition at line 783 of file [fixed_pkg_c.vhdl](#).

6.17.2.376 **UNRESOLVED_ufixed to_ufixed(*argin UNSIGNED*)** [Function]

Definition at line 791 of file [fixed_pkg_c.vhdl](#).

6.17.2.377 **UNRESOLVED_ufixed to_ufixed(*argin* STD_ULOGIC_VECTOR , *left_index*in INTEGER , *right_index*in INTEGER)** [Function]

Definition at line 1048 of file [fixed_pkg_c.vhdl](#).

6.17.2.378 **UNRESOLVED_ufixed to_ufixed(*argin* STD_ULOGIC_VECTOR , *size_res*in UNRESOLVED_ufixed)** [Function]

Definition at line 1054 of file [fixed_pkg_c.vhdl](#).

6.17.2.379 **UNRESOLVED_ufixed to_ufixed(*argin* STD_LOGIC_VECTOR , *left_index*in INTEGER , *right_index*in INTEGER)** [Function]

Definition at line 1417 of file [fixed_pkg_c.vhdl](#).

6.17.2.380 **UNRESOLVED_ufixed to_ufixed(*argin* STD_LOGIC_VECTOR , *size_res*in UNRESOLVED_ufixed)** [Function]

Definition at line 1423 of file [fixed_pkg_c.vhdl](#).

6.17.2.381 **UNSIGNED to_unsigned(*argin* UNRESOLVED_ufixed , *size*in NATURAL , *overflow_style*in fixed_overflow_style_type fixed_overflow_style , *round_style*in fixed_round_style_type fixed_round_style)** [Function]

Definition at line 796 of file [fixed_pkg_c.vhdl](#).

6.17.2.382 **UNSIGNED to_unsigned(*argin* UNRESOLVED_ufixed , *size_res*in UNSIGNED , *overflow_style*in fixed_overflow_style_type fixed_overflow_style , *round_style*in fixed_round_style_type fixed_round_style)** [Function]

Definition at line 804 of file [fixed_pkg_c.vhdl](#).

6.17.2.383 **UNRESOLVED_ufixed to_UX01(*argin* UNRESOLVED_ufixed)** [Function]

Definition at line 1008 of file [fixed_pkg_c.vhdl](#).

6.17.2.384 **UNRESOLVED_sfixed to_UX01(*argin* UNRESOLVED_sfixed)** [Function]

Definition at line 1009 of file [fixed_pkg_c.vhdl](#).

6.17.2.385 **UNRESOLVED_ufixed to_X01(*argin* UNRESOLVED_ufixed)** [Function]

Definition at line 1004 of file [fixed_pkg_c.vhdl](#).

6.17.2.386 **UNRESOLVED_sfixed to_X01(*argin* UNRESOLVED_sfixed)** [Function]

Definition at line 1005 of file [fixed_pkg_c.vhdl](#).

6.17.2.387 **UNRESOLVED_ufixed to_X01Z(*argin* UNRESOLVED_ufixed)** [Function]

Definition at line 1006 of file [fixed_pkg_c.vhdl](#).

6.17.2.388 **UNRESOLVED_sfixed to_X01Z(*argin* UNRESOLVED_sfixed)** [Function]

Definition at line 1007 of file [fixed_pkg_c.vhdl](#).

6.17.2.389 **INTEGER UFix_high(*width*in NATURAL , *fraction*in NATURAL , *operation*in CHARACTER'X' , *width2*in NATURAL 0 , *fraction2*in NATURAL 0)** [Function]

Definition at line 1100 of file [fixed_pkg_c.vhdl](#).

6.17.2.390 **INTEGER UFix_low(*width*in NATURAL , *fraction*in NATURAL , *operation*in CHARACTER'X' , *width2*in NATURAL 0 , *fraction2*in NATURAL 0)** [Function]

Definition at line 1105 of file [fixed_pkg_c.vhdl](#).

6.17.2.391 **INTEGER ufixed_high(*left_index*in INTEGER , *right_index*in INTEGER , *operation*in CHARACTER'X' , *left_index2*in INTEGER 0 , *right_index2*in INTEGER 0)** [Function]

Definition at line 921 of file [fixed_pkg_c.vhdl](#).

6.17.2.392 **INTEGER ufixed_high(*size_res*in UNRESOLVED_ufixed , *operation*in CHARACTER'X' , *size_res2*in UNRESOLVED_ufixed)** [Function]

Definition at line 946 of file [fixed_pkg_c.vhdl](#).

6.17.2.393 **INTEGER ufixed_low(*left_index*in INTEGER , *right_index*in INTEGER , *operation*in CHARACTER'X' , *left_index2*in INTEGER 0 , *right_index2*in INTEGER 0)** [Function]

Definition at line 926 of file [fixed_pkg_c.vhdl](#).

6.17.2.394 **INTEGER ufixed_low(*size_res*in UNRESOLVED_ufixed , *operation*in CHARACTER'X' , *size_res2*in UNRESOLVED_ufixed)** [Function]

Definition at line 951 of file [fixed_pkg_c.vhdl](#).

6.17.2.395 **WRITE(*L* inoutLINE , *VALUE*inUNRESOLVED_ufixed , *JUSTIFIED*inSIDE*right* , *FIELD* inWIDTH 0)** [Procedure]

Definition at line 1130 of file [fixed_pkg_c.vhdl](#).

6.17.2.396 **WRITE(*L* inoutLINE , *VALUE*inUNRESOLVED_sfixed , *JUSTIFIED*inSIDE*right* , *FIELD* inWIDTH 0)** [Procedure]

Definition at line 1137 of file [fixed_pkg_c.vhdl](#).

6.17.2.397 **STD_ULOGIC xnor_reduce(*lin* UNRESOLVED_ufixed)** [Function]

Definition at line 666 of file [fixed_pkg_c.vhdl](#).

6.17.2.398 **STD_ULOGIC xnor_reduce(*lin* UNRESOLVED_sfixed)** [Function]

Definition at line 672 of file [fixed_pkg_c.vhdl](#).

6.17.2.399 **STD_ULOGIC xor_reduce(/in UNRESOLVED_ufixed)** [Function]

Definition at line 665 of file [fixed_pkg_c.vhdl](#).

6.17.2.400 **STD_ULOGIC xor_reduce(/in UNRESOLVED_sfixed)** [Function]

Definition at line 671 of file [fixed_pkg_c.vhdl](#).

6.17.3 Member Data Documentation

6.17.3.1 **BINARY_READ isREAD[LINE,UNRESOLVED_ufixed ,BOOLEAN]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.2 **BINARY_READ isREAD[LINE,UNRESOLVED_ufixed]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.3 **BINARY_READ isREAD[LINE,UNRESOLVED_sfixed ,BOOLEAN]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.4 **BINARY_READ isREAD[LINE,UNRESOLVED_sfixed]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.5 **BINARY_WRITE isWRITE[LINE,UNRESOLVED_ufixed ,SIDE,width]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.6 **BINARY_WRITE isWRITE[LINE,UNRESOLVED_sfixed ,SIDE,width]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.7 **bread isREAD[LINE,UNRESOLVED_ufixed]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.8 **bread isREAD[LINE,UNRESOLVED_ufixed ,BOOLEAN]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.9 **bread isREAD[LINE,UNRESOLVED_sfixed]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.10 **bread isREAD[LINE,UNRESOLVED_sfixed ,BOOLEAN]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.11 **bwrite** **isWRITE[LINE,UNRESOLVED_sfixed ,SIDE,width]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.12 **bwrite** **isWRITE[LINE,UNRESOLVED_ufixed ,SIDE,width]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.13 **fixed_float_types** [Package]

Definition at line 23 of file [fixed_pkg_c.vhdl](#).

6.17.3.14 **fixed_guard_bits** **NATURAL:=3** [Constant]

Definition at line 32 of file [fixed_pkg_c.vhdl](#).

6.17.3.15 **fixed_overflow_style** **fixed_overflow_style_type :=fixed_saturate** [Constant]

Definition at line 30 of file [fixed_pkg_c.vhdl](#).

6.17.3.16 **fixed_round_style** **fixed_round_style_type :=fixed_round** [Constant]

Definition at line 28 of file [fixed_pkg_c.vhdl](#).

6.17.3.17 **from_binary_string** **isfrom_string[STRING,UNRESOLVED_ufixedreturnUNRESOLVED_ufixed]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.18 **from_binary_string** **isfrom_string[STRING,UNRESOLVED_sfixedreturnUNRESOLVED_sfixed]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.19 **from_binary_string** **isfrom_string[STRINGreturnUNRESOLVED_ufixed]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.20 **from_binary_string** **isfrom_string[STRINGreturnUNRESOLVED_sfixed]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.21 **from_binary_string** **isfrom_string[STRING,INTEGER,INTEGERreturnUNRESOLVED_ufixed]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.22 **from_binary_string** **isfrom_string[STRING,INTEGER,INTEGERreturnUNRESOLVED_sfixed]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.23 **from_bstring** **isfrom_string[STRING,UNRESOLVED_ufixedreturnUNRESOLVED_ufixed]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.24 **from_bstring** **isfrom_string[STRING,UNRESOLVED_sfixedreturnUNRESOLVED_sfixed]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.25 **from_bstring** **isfrom_string[STRINGreturnUNRESOLVED_ufixed]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.26 **from_bstring** **isfrom_string[STRINGreturnUNRESOLVED_sfixed]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.27 **from_bstring** **isfrom_string[STRING,INTEGER,INTEGERreturnUNRESOLVED_ufixed]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.28 **from_bstring** **isfrom_string[STRING,INTEGER,INTEGERreturnUNRESOLVED_sfixed]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.29 **from_hex_string** **isfrom_hstring[STRING,INTEGER,INTEGERreturnUNRESOLVED_sfixed]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.30 **from_hex_string** **isfrom_hstring[STRING,UNRESOLVED_sfixedreturnUNRESOLVED_sfixed]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.31 **from_hex_string** **isfrom_hstring[STRINGreturnUNRESOLVED_ufixed]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.32 **from_hex_string** **isfrom_hstring[STRINGreturnUNRESOLVED_sfixed]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.33 **from_hex_string** **is****from_hstring**[**STRING**,**UNRESOLVED_ufixed****return****UNRESOLVED_ufixed**]
[Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.34 **from_hex_string** **is****from_hstring**[**STRING**,**INTEGER**,**INTEGER****return****UNRESOLVED_ufixed**]
[Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.35 **from_octal_string** **is****from_ostring**[**STRING**,**UNRESOLVED_ufixed****return****UNRESOLVED_ufixed**]
[Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.36 **from_octal_string** **is****from_ostring**[**STRING****return****UNRESOLVED_sfixed**] [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.37 **from_octal_string** **is****from_ostring**[**STRING****return****UNRESOLVED_ufixed**] [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.38 **from_octal_string** **is****from_ostring**[**STRING**,**UNRESOLVED_sfixed****return****UNRESOLVED_sfixed**]
[Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.39 **from_octal_string** **is****from_ostring**[**STRING**,**INTEGER**,**INTEGER****return****UNRESOLVED_ufixed**]
[Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.40 **from_octal_string** **is****from_ostring**[**STRING**,**INTEGER**,**INTEGER****return****UNRESOLVED_sfixed**]
[Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.41 **HEX_READ** **is****HREAD**[**LINE**,**UNRESOLVED_ufixed** ,**BOOLEAN**] [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.42 **HEX_READ** **is****HREAD**[**LINE**,**UNRESOLVED_sfixed** ,**BOOLEAN**] [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.43 **HEX_READ** **is****HREAD**[**LINE**,**UNRESOLVED_ufixed**] [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.44 **HEX_READ** **isHREAD[LINE,UNRESOLVED_sfixed]** [Alias]

Definition at line [1734308965](#) of file [fixed_pkg_c.vhdl](#).

6.17.3.45 **HEX_WRITE** **isHWRITE[LINE,UNRESOLVED_ufixed ,SIDE,WIDTH]** [Alias]

Definition at line [1734308965](#) of file [fixed_pkg_c.vhdl](#).

6.17.3.46 **HEX_WRITE** **isHWRITE[LINE,UNRESOLVED_sfixed ,SIDE,WIDTH]** [Alias]

Definition at line [1734308965](#) of file [fixed_pkg_c.vhdl](#).

6.17.3.47 **IEEE** [Library]

Definition at line [19](#) of file [fixed_pkg_c.vhdl](#).

6.17.3.48 **IEEE_PROPOSED** [Library]

Definition at line [22](#) of file [fixed_pkg_c.vhdl](#).

6.17.3.49 **no_warning BOOLEAN:=(false)** [Constant]

Definition at line [34](#) of file [fixed_pkg_c.vhdl](#).

6.17.3.50 **NUMERIC_STD** [Package]

Definition at line [21](#) of file [fixed_pkg_c.vhdl](#).

6.17.3.51 **OCTAL_READ** **isOREAD[LINE,UNRESOLVED_ufixed]** [Alias]

Definition at line [1734308965](#) of file [fixed_pkg_c.vhdl](#).

6.17.3.52 **OCTAL_READ** **isOREAD[LINE,UNRESOLVED_ufixed ,BOOLEAN]** [Alias]

Definition at line [1734308965](#) of file [fixed_pkg_c.vhdl](#).

6.17.3.53 **OCTAL_READ** **isOREAD[LINE,UNRESOLVED_sfixed]** [Alias]

Definition at line [1734308965](#) of file [fixed_pkg_c.vhdl](#).

6.17.3.54 **OCTAL_READ** **isOREAD[LINE,UNRESOLVED_sfixed ,BOOLEAN]** [Alias]

Definition at line [1734308965](#) of file [fixed_pkg_c.vhdl](#).

6.17.3.55 **OCTAL_WRITE** **isOWRITE[LINE,UNRESOLVED_ufixed ,SIDE,WIDTH]** [Alias]

Definition at line [1734308965](#) of file [fixed_pkg_c.vhdl](#).

6.17.3.56 OCTAL_WRITE **isOWRITE[LINE,UNRESOLVED_sfixed ,SIDE,WIDTH]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.57 sfixed UNRESOLVED_sfixed [Subtype]

Definition at line 48 of file [fixed_pkg_c.vhdl](#).

6.17.3.58 STD_LOGIC_1164 [Package]

Definition at line 20 of file [fixed_pkg_c.vhdl](#).

6.17.3.59 TEXTIO [Package]

Definition at line 18 of file [fixed_pkg_c.vhdl](#).

6.17.3.60 TO_BINARY_STRING **isTO_STRING[UNRESOLVED_ufixedreturnSTRING]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.61 TO_BINARY_STRING **isTO_STRING[UNRESOLVED_sfixedreturnSTRING]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.62 to_bstring **isto_string[UNRESOLVED_ufixedreturnSTRING]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.63 to_bstring **isto_string[UNRESOLVED_sfixedreturnSTRING]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.64 TO_HEX_STRING **isTO_HSTRING[UNRESOLVED_sfixedreturnSTRING]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.65 TO_HEX_STRING **isTO_HSTRING[UNRESOLVED_ufixedreturnSTRING]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.66 TO_OCTAL_STRING **isTO_OSTRING[UNRESOLVED_ufixedreturnSTRING]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.67 TO_OCTAL_STRING **isTO_OSTRING[UNRESOLVED_sfixedreturnSTRING]** [Alias]

Definition at line 1734308965 of file [fixed_pkg_c.vhdl](#).

6.17.3.68 **to_Std_Logic_Vector** **isto_slv[UNRESOLVED_sfixedreturnSTD_LOGIC_VECTOR]** [Alias]

Definition at line [1734308965](#) of file [fixed_pkg_c.vhdl](#).

6.17.3.69 **to_Std_Logic_Vector** **isto_slv[UNRESOLVED_ufixedreturnSTD_LOGIC_VECTOR]** [Alias]

Definition at line [1734308965](#) of file [fixed_pkg_c.vhdl](#).

6.17.3.70 **to_Std_ULogic_Vector** **isto_suv[UNRESOLVED_sfixedreturnSTD_ULOGIC_VECTOR]** [Alias]

Definition at line [1734308965](#) of file [fixed_pkg_c.vhdl](#).

6.17.3.71 **to_Std_ULogic_Vector** **isto_suv[UNRESOLVED_ufixedreturnSTD_ULOGIC_VECTOR]**
[Alias]

Definition at line [1734308965](#) of file [fixed_pkg_c.vhdl](#).

6.17.3.72 **to_StdLogicVector** **isto_slv[UNRESOLVED_ufixedreturnSTD_LOGIC_VECTOR]** [Alias]

Definition at line [1734308965](#) of file [fixed_pkg_c.vhdl](#).

6.17.3.73 **to_StdLogicVector** **isto_slv[UNRESOLVED_sfixedreturnSTD_LOGIC_VECTOR]** [Alias]

Definition at line [1734308965](#) of file [fixed_pkg_c.vhdl](#).

6.17.3.74 **to_StdULogicVector** **isto_suv[UNRESOLVED_sfixedreturnSTD_ULOGIC_VECTOR]** [Alias]

Definition at line [1734308965](#) of file [fixed_pkg_c.vhdl](#).

6.17.3.75 **to_StdULogicVector** **isto_suv[UNRESOLVED_ufixedreturnSTD_ULOGIC_VECTOR]** [Alias]

Definition at line [1734308965](#) of file [fixed_pkg_c.vhdl](#).

6.17.3.76 **U_sfixed UNRESOLVED_sfixed** [Subtype]

Definition at line [45](#) of file [fixed_pkg_c.vhdl](#).

6.17.3.77 **U_ufixed UNRESOLVED_ufixed** [Subtype]

Definition at line [44](#) of file [fixed_pkg_c.vhdl](#).

6.17.3.78 **ufixed UNRESOLVED_ufixed** [Subtype]

Definition at line [47](#) of file [fixed_pkg_c.vhdl](#).

6.17.3.79 **UNRESOLVED_sfixed array(INTEGERrange<>)ofSTD_ULOGIC** [Type]

Definition at line [42](#) of file [fixed_pkg_c.vhdl](#).

6.17.3.80 UNRESOLVED_ufixed array(INTEGERrange<>)ofSTD_ULOGIC [Type]

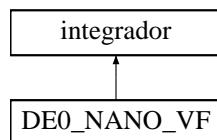
Definition at line 40 of file [fixed_pkg_c.vhdl](#).

The documentation for this class was generated from the following file:

- [fixed_pkg_c.vhdl](#)

6.18 integrador Entity Reference

Inheritance diagram for integrador:



Entities

- [integrador](#) architecture

Libraries

- [IEEE](#)

Use Clauses

- [STD_LOGIC_UNSIGNED](#)
- [STD_LOGIC_1164](#)
- [fixed_pkg](#)

Generics

- [Nin integer:= 13](#)
- [Nout integer:= 30](#)

Ports

- [clk in std_logic](#)
- [en in std_logic](#)
- [reset in std_logic](#)
- [sinc out std_logic](#)
- [MAX in std_logic_vector\(Nout - 1 downto 0 \)](#)
- [out_data out std_logic_vector\(Nout - 1 downto 0 \)](#)
- [int_data in std_logic_vector\(Nin - 1 downto 0 \)](#)

6.18.1 Detailed Description

Definition at line 6 of file [integrador.vhd](#).

6.18.2 Member Data Documentation

6.18.2.1 **clk** **in std_logic** [Port]

Definition at line 11 of file [integrador.vhd](#).

6.18.2.2 **en** **in std_logic** [Port]

Definition at line 12 of file [integrador.vhd](#).

6.18.2.3 **fixed_pkg** [Package]

Definition at line 4 of file [integrador.vhd](#).

6.18.2.4 **IEEE** [Library]

Definition at line 1 of file [integrador.vhd](#).

6.18.2.5 **int_data** **in std_logic_vector(Nin - 1 downto 0)** [Port]

Definition at line 18 of file [integrador.vhd](#).

6.18.2.6 **MAX** **in std_logic_vector(Nout - 1 downto 0)** [Port]

Definition at line 15 of file [integrador.vhd](#).

6.18.2.7 **Nin** **integer:= 13** [Generic]

Definition at line 7 of file [integrador.vhd](#).

6.18.2.8 **Nout** **integer:= 30** [Generic]

Definition at line 9 of file [integrador.vhd](#).

6.18.2.9 **out_data** **out std_logic_vector(Nout - 1 downto 0)** [Port]

Definition at line 16 of file [integrador.vhd](#).

6.18.2.10 **reset** **in std_logic** [Port]

Definition at line 13 of file [integrador.vhd](#).

6.18.2.11 **sinc** **out std_logic** [Port]

Definition at line 14 of file [integrador.vhd](#).

6.18.2.12 **STD_LOGIC_1164** [Package]

Definition at line 3 of file [integrador.vhd](#).

6.18.2.13 STD_LOGIC_UNSIGNED [Package]

Definition at line 2 of file [integrador.vhd](#).

The documentation for this class was generated from the following file:

- [integrador.vhd](#)

6.19 integrador Architecture Reference

Processes

- [PROCESS_10\(clk \)](#)

Signals

- [out_int std_logic_vector\(Nout - 1 downto 0\)](#)
- [sinc_int std_logic](#)

6.19.1 Detailed Description

Definition at line 23 of file [integrador.vhd](#).

6.19.2 Member Function Documentation

6.19.2.1 PROCESS_10(clk) [Process]

Definition at line 31 of file [integrador.vhd](#).

6.19.3 Member Data Documentation

6.19.3.1 out_int std_logic_vector(Nout - 1 downto 0) [Signal]

Definition at line 25 of file [integrador.vhd](#).

6.19.3.2 sinc_int std_logic [Signal]

Definition at line 26 of file [integrador.vhd](#).

The documentation for this class was generated from the following file:

- [integrador.vhd](#)

6.20 LEDs Entity Reference

Entities

- [LEDs architecture](#)

Libraries

- ieee

Use Clauses

- std_logic_1164
- std_logic_arith
- std_logic_unsigned

Ports

- CLOCK_50 in std_logic
- LED out std_logic_vector(7 DOWNTO 0)
- SW in std_logic_vector(3 DOWNTO 0)
- KEY in std_logic_vector(1 DOWNTO 0)

6.20.1 Detailed Description

Definition at line 9 of file [LEDs.vhd](#).

6.20.2 Member Data Documentation

6.20.2.1 CLOCK_50 in std_logic [Port]

Definition at line 11 of file [LEDs.vhd](#).

6.20.2.2 ieee [Library]

Definition at line 4 of file [LEDs.vhd](#).

6.20.2.3 KEY in std_logic_vector(1 DOWNTO 0) [Port]

Definition at line 15 of file [LEDs.vhd](#).

6.20.2.4 LED out std_logic_vector(7 DOWNTO 0) [Port]

Definition at line 12 of file [LEDs.vhd](#).

6.20.2.5 std_logic_1164 [Package]

Definition at line 5 of file [LEDs.vhd](#).

6.20.2.6 std_logic_arith [Package]

Definition at line 6 of file [LEDs.vhd](#).

6.20.2.7 std_logic_unsigned [Package]

Definition at line 7 of file [LEDs.vhd](#).

6.20.2.8 SW **in std_logic_vector(3 DOWNTO 0)** [Port]

Definition at line 13 of file [LEDs.vhd](#).

The documentation for this class was generated from the following file:

- [LEDs.vhd](#)

6.21 LEDs Architecture Reference

Processes

- [PROCESS_11\(CLOCK_50 \)](#)

Constants

- [CLK_FREQ integer:= 50000000](#)
- [BLINK_FREQ integer:= 1](#)
- [CNT_MAX integer:=CLK_FREQ /BLINK_FREQ / 2 - 1](#)

Signals

- [cnt unsigned\(24 downto 0 \)](#)
- [blink std_logic](#)

6.21.1 Detailed Description

Definition at line 22 of file [LEDs.vhd](#).

6.21.2 Member Function Documentation

6.21.2.1 PROCESS_11(CLOCK_50) [Process]

Definition at line 29 of file [LEDs.vhd](#).

6.21.3 Member Data Documentation

6.21.3.1 blink std_logic [Signal]

Definition at line 27 of file [LEDs.vhd](#).

6.21.3.2 BLINK_FREQ integer:= 1 [Constant]

Definition at line 24 of file [LEDs.vhd](#).

6.21.3.3 CLK_FREQ integer:= 50000000 [Constant]

Definition at line 23 of file [LEDs.vhd](#).

6.21.3.4 cnt **unsigned(24 downto 0)** [Signal]

Definition at line 26 of file [LEDs.vhd](#).

6.21.3.5 CNT_MAX **integer:=CLK_FREQ /BLINK_FREQ / 2 - 1** [Constant]

Definition at line 25 of file [LEDs.vhd](#).

The documentation for this class was generated from the following file:

- [LEDs.vhd](#)

6.22 MAIN Architecture Reference

Processes

- [PROCESS_4\(err_FABC \)](#)
- [PROCESS_5\(clk_pll \)](#)

Components

- [LEDs](#)
- [contador](#)
- [tabela_sin](#)
- [pll](#)
- [integrador](#)
- [theta_abc](#)
- [clk_div](#)
- [portadora_tringular](#)
- [fbpspwm](#)
- [vfcontrol](#)

Signals

- [clk_pll std_logic](#)
- [pll_lock std_logic](#)
- [clk_wt std_logic](#)
- [clk_int std_logic](#)
- [clk_vf std_logic](#)
- [clk_led std_logic](#)
- [reset std_logic](#)
- [th_a std_logic_vector\(15 downto 0 \)](#)
- [th_b std_logic_vector\(15 downto 0 \)](#)
- [th_c std_logic_vector\(15 downto 0 \)](#)
- [sigPWM01 std_logic](#)
- [sigPWM02 std_logic](#)
- [OSC_BUS1 std_logic_vector\(9 downto 0 \)](#)
- [sinc_int std_logic](#)
- [sinc_wt std_logic](#)
- [pulso_key0 std_logic](#)
- [key0_ant std_logic](#)
- [pulso_key1 std_logic](#)

- key1_ant std_logic
- toggle_key1 std_logic:=‘1’
- toggle_key0 std_logic:=‘1’
- rst std_logic:=‘1’
- moduladoras COMP_ARRAY
- portadoras COMP_ARRAY
- amostragem_moduladoras std_logic_vector(24 downto 1)
- en_PWM std_logic:=‘1’
- en_PWMA std_logic:=‘1’
- en_PWMB std_logic:=‘1’
- en_PWMC std_logic:=‘1’
- err_FA std_logic
- err_FB std_logic
- err_FC std_logic
- err_FABC std_logic
- sin_a std_logic_vector(n_bits_c - 1 downto 0)
- sin_b std_logic_vector(n_bits_c - 1 downto 0)
- sin_c std_logic_vector(n_bits_c - 1 downto 0)
- ma std_logic_vector(n_bits_c - 1 downto 0)
- mb std_logic_vector(n_bits_c - 1 downto 0)
- mc std_logic_vector(n_bits_c - 1 downto 0)
- mVF sfixed(I downto F)
- incVF std_logic_vector(12 downto 0)
- bidir std_logic
- dirPWM1 std_logic
- dirPWM2 std_logic
- dirPWM3 std_logic
- cPWM1 std_logic_vector(15 downto 0)
- cPWM2 std_logic_vector(15 downto 0)
- cPWM3 std_logic_vector(15 downto 0)
- dirTRI1 std_logic
- dirTRI2 std_logic
- dirTRI3 std_logic
- dirTRI4 std_logic
- dirTRI5 std_logic
- dirTRI6 std_logic
- cTRI1 std_logic_vector(15 downto 0)
- cTRI2 std_logic_vector(15 downto 0)
- cTRI3 std_logic_vector(15 downto 0)
- cTRI4 std_logic_vector(15 downto 0)
- cTRI5 std_logic_vector(15 downto 0)
- cTRI6 std_logic_vector(15 downto 0)
- omega_pll std_logic_vector(Nin - 1 downto 0)
- theta_pll std_logic_vector(Nout - 1 downto 0)
- theta_wt std_logic_vector(15 downto 0)

Instantiations

- upll pll
- u1_clk_div
- uclkvf clk_div
- uvf vfcontrol
- u5_integrador
- ucr1_portadora_tringular

- [ucr2 portadora_tringular](#)
- [ucr3 portadora_tringular](#)
- [ucr4 portadora_tringular](#)
- [ucr5 portadora_tringular](#)
- [ucr6 portadora_tringular](#)
- [u6 theta_abc](#)
- [usin_a tabela_sin](#)
- [usin_b tabela_sin](#)
- [usin_c tabela_sin](#)
- [pwm1_fa01 fbp pwm dt](#)
- [pwm2_fa01 fbp pwm dt](#)
- [pwm1_fa02 fbp pwm dt](#)
- [pwm2_fa02 fbp pwm dt](#)
- [pwm1_fa03 fbp pwm dt](#)
- [pwm2_fa03 fbp pwm dt](#)
- [pwm1_fb01 fbp pwm dt](#)
- [pwm2_fb01 fbp pwm dt](#)
- [pwm1_fb02 fbp pwm dt](#)
- [pwm2_fb02 fbp pwm dt](#)
- [pwm1_fb03 fbp pwm dt](#)
- [pwm2_fb03 fbp pwm dt](#)
- [pwm1_fc01 fbp pwm dt](#)
- [pwm2_fc01 fbp pwm dt](#)
- [pwm1_fc02 fbp pwm dt](#)
- [pwm2_fc02 fbp pwm dt](#)
- [pwm1_fc03 fbp pwm dt](#)
- [pwm2_fc03 fbp pwm dt](#)

6.22.1 Detailed Description

Definition at line [77](#) of file [DE0_NANO_VF.vhd](#).

6.22.2 Member Function Documentation

6.22.2.1 PROCESS_4(*err_FABC*)

Definition at line [414](#) of file [DE0_NANO_VF.vhd](#).

6.22.2.2 PROCESS_5(*clk_pll*)

Definition at line [657](#) of file [DE0_NANO_VF.vhd](#).

6.22.3 Member Data Documentation

6.22.3.1 amostragem_moduladoras std_logic_vector(**24** **downto** **1**) [Signal]

Definition at line [220](#) of file [DE0_NANO_VF.vhd](#).

6.22.3.2 bidir std_logic [Signal]

Definition at line [231](#) of file [DE0_NANO_VF.vhd](#).

6.22.3.3 **clk_div** [Component]

Definition at line 153 of file [DE0_NANO_VF.vhd](#).

6.22.3.4 **clk_int std_logic** [Signal]

Definition at line 201 of file [DE0_NANO_VF.vhd](#).

6.22.3.5 **clk_led std_logic** [Signal]

Definition at line 201 of file [DE0_NANO_VF.vhd](#).

6.22.3.6 **clk_pll std_logic** [Signal]

Definition at line 200 of file [DE0_NANO_VF.vhd](#).

6.22.3.7 **clk_vf std_logic** [Signal]

Definition at line 201 of file [DE0_NANO_VF.vhd](#).

6.22.3.8 **clk_wt std_logic** [Signal]

Definition at line 200 of file [DE0_NANO_VF.vhd](#).

6.22.3.9 **contador** [Component]

Definition at line 90 of file [DE0_NANO_VF.vhd](#).

6.22.3.10 **cPWM1 std_logic_vector(15 downto 0)** [Signal]

Definition at line 235 of file [DE0_NANO_VF.vhd](#).

6.22.3.11 **cPWM2 std_logic_vector(15 downto 0)** [Signal]

Definition at line 235 of file [DE0_NANO_VF.vhd](#).

6.22.3.12 **cPWM3 std_logic_vector(15 downto 0)** [Signal]

Definition at line 235 of file [DE0_NANO_VF.vhd](#).

6.22.3.13 **cTRI1 std_logic_vector(15 downto 0)** [Signal]

Definition at line 239 of file [DE0_NANO_VF.vhd](#).

6.22.3.14 **cTRI2 std_logic_vector(15 downto 0)** [Signal]

Definition at line 239 of file [DE0_NANO_VF.vhd](#).

6.22.3.15 **cTRI3 std_logic_vector(15 downto 0)** [Signal]

Definition at line 239 of file [DE0_NANO_VF.vhd](#).

6.22.3.16 **cTRI4 std_logic_vector(15 downto 0)** [Signal]

Definition at line 239 of file [DE0_NANO_VF.vhd](#).

6.22.3.17 **cTRI5 std_logic_vector(15 downto 0)** [Signal]

Definition at line 239 of file [DE0_NANO_VF.vhd](#).

6.22.3.18 **cTRI6 std_logic_vector(15 downto 0)** [Signal]

Definition at line 239 of file [DE0_NANO_VF.vhd](#).

6.22.3.19 **dirPWM1 std_logic** [Signal]

Definition at line 234 of file [DE0_NANO_VF.vhd](#).

6.22.3.20 **dirPWM2 std_logic** [Signal]

Definition at line 234 of file [DE0_NANO_VF.vhd](#).

6.22.3.21 **dirPWM3 std_logic** [Signal]

Definition at line 234 of file [DE0_NANO_VF.vhd](#).

6.22.3.22 **dirTRI1 std_logic** [Signal]

Definition at line 238 of file [DE0_NANO_VF.vhd](#).

6.22.3.23 **dirTRI2 std_logic** [Signal]

Definition at line 238 of file [DE0_NANO_VF.vhd](#).

6.22.3.24 **dirTRI3 std_logic** [Signal]

Definition at line 238 of file [DE0_NANO_VF.vhd](#).

6.22.3.25 **dirTRI4 std_logic** [Signal]

Definition at line 238 of file [DE0_NANO_VF.vhd](#).

6.22.3.26 **dirTRI5 std_logic** [Signal]

Definition at line 238 of file [DE0_NANO_VF.vhd](#).

6.22.3.27 **dirTRI6 std_logic** [Signal]

Definition at line 238 of file [DE0_NANO_VF.vhd](#).

6.22.3.28 **en_PWM std_logic:= '1'** [Signal]

Definition at line 221 of file [DE0_NANO_VF.vhd](#).

6.22.3.29 **en_PWMA std_logic:= '1'** [Signal]

Definition at line 221 of file [DE0_NANO_VF.vhd](#).

6.22.3.30 **en_PWMB std_logic:= '1'** [Signal]

Definition at line 221 of file [DE0_NANO_VF.vhd](#).

6.22.3.31 **en_PWMC std_logic:= '1'** [Signal]

Definition at line 221 of file [DE0_NANO_VF.vhd](#).

6.22.3.32 **err_FA std_logic** [Signal]

Definition at line 222 of file [DE0_NANO_VF.vhd](#).

6.22.3.33 **err_FABC std_logic** [Signal]

Definition at line 222 of file [DE0_NANO_VF.vhd](#).

6.22.3.34 **err_FB std_logic** [Signal]

Definition at line 222 of file [DE0_NANO_VF.vhd](#).

6.22.3.35 **err_FC std_logic** [Signal]

Definition at line 222 of file [DE0_NANO_VF.vhd](#).

6.22.3.36 **fbspwmtd** [Component]

Definition at line 176 of file [DE0_NANO_VF.vhd](#).

6.22.3.37 **incVF std_logic_vector(12 downto 0)** [Signal]

Definition at line 229 of file [DE0_NANO_VF.vhd](#).

6.22.3.38 **integrador** [Component]

Definition at line 127 of file [DE0_NANO_VF.vhd](#).

6.22.3.39 **key0_ant std_logic** [Signal]

Definition at line 211 of file [DE0_NANO_VF.vhd](#).

6.22.3.40 **key1_ant std_logic** [Signal]

Definition at line 212 of file [DE0_NANO_VF.vhd](#).

6.22.3.41 **LEDs** [Component]

Definition at line 80 of file [DE0_NANO_VF.vhd](#).

6.22.3.42 **ma std_logic_vector(n_bits_c - 1 downto 0)** [Signal]

Definition at line 227 of file [DE0_NANO_VF.vhd](#).

6.22.3.43 **mb std_logic_vector(n_bits_c - 1 downto 0)** [Signal]

Definition at line 227 of file [DE0_NANO_VF.vhd](#).

6.22.3.44 **mc std_logic_vector(n_bits_c - 1 downto 0)** [Signal]

Definition at line 227 of file [DE0_NANO_VF.vhd](#).

6.22.3.45 **moduladoras COMP_ARRAY** [Signal]

Definition at line 218 of file [DE0_NANO_VF.vhd](#).

6.22.3.46 **mVF sfixed(I downto F)** [Signal]

Definition at line 228 of file [DE0_NANO_VF.vhd](#).

6.22.3.47 **omega_pll std_logic_vector(Nin - 1 downto 0)** [Signal]

Definition at line 242 of file [DE0_NANO_VF.vhd](#).

6.22.3.48 **OSC_BUS1 std_logic_vector(9 downto 0)** [Signal]

Definition at line 209 of file [DE0_NANO_VF.vhd](#).

6.22.3.49 **pll** [Component]

Definition at line 116 of file [DE0_NANO_VF.vhd](#).

6.22.3.50 **pll_lock std_logic** [Signal]

Definition at line 200 of file [DE0_NANO_VF.vhd](#).

6.22.3.51 **portadora_tringular** [Component]

Definition at line 162 of file [DE0_NANO_VF.vhd](#).

6.22.3.52 **portadoras COMP_ARRAY** [Signal]

Definition at line 219 of file [DE0_NANO_VF.vhd](#).

6.22.3.53 **pulso_key0 std_logic** [Signal]

Definition at line 211 of file [DE0_NANO_VF.vhd](#).

6.22.3.54 **pulso_key1 std_logic** [Signal]

Definition at line 212 of file [DE0_NANO_VF.vhd](#).

6.22.3.55 **pwm1_fa01 fbpspwmdt** [Instantiation]

Definition at line 443 of file [DE0_NANO_VF.vhd](#).

6.22.3.56 **pwm1_fa02 fbpspwmdt** [Instantiation]

Definition at line 466 of file [DE0_NANO_VF.vhd](#).

6.22.3.57 **pwm1_fa03 fbpspwmdt** [Instantiation]

Definition at line 490 of file [DE0_NANO_VF.vhd](#).

6.22.3.58 **pwm1_fb01 fbpspwmdt** [Instantiation]

Definition at line 518 of file [DE0_NANO_VF.vhd](#).

6.22.3.59 **pwm1_fb02 fbpspwmdt** [Instantiation]

Definition at line 541 of file [DE0_NANO_VF.vhd](#).

6.22.3.60 **pwm1_fb03 fbpspwmdt** [Instantiation]

Definition at line 565 of file [DE0_NANO_VF.vhd](#).

6.22.3.61 **pwm1_fc01 fbpspwmdt** [Instantiation]

Definition at line 595 of file [DE0_NANO_VF.vhd](#).

6.22.3.62 **pwm1_fc02 fbpspwmdt** [Instantiation]

Definition at line 618 of file [DE0_NANO_VF.vhd](#).

6.22.3.63 **pwm1_fc03 fbpspwmdt** [Instantiation]

Definition at line 642 of file [DE0_NANO_VF.vhd](#).

6.22.3.64 **pwm2_fa01 fbpspwmdt** [Instantiation]

Definition at line 454 of file [DE0_NANO_VF.vhd](#).

6.22.3.65 **pwm2_fa02 fbpspwmdt** [Instantiation]

Definition at line 477 of file [DE0_NANO_VF.vhd](#).

6.22.3.66 **pwm2_fa03 fbpspwmdt** [Instantiation]

Definition at line 501 of file [DE0_NANO_VF.vhd](#).

6.22.3.67 **pwm2_fb01 fbpspwmdt** [Instantiation]

Definition at line 529 of file [DE0_NANO_VF.vhd](#).

6.22.3.68 **pwm2_fb02 fbpspwmdt** [Instantiation]

Definition at line 552 of file [DE0_NANO_VF.vhd](#).

6.22.3.69 **pwm2_fb03 fbpspwmdt** [Instantiation]

Definition at line 576 of file [DE0_NANO_VF.vhd](#).

6.22.3.70 **pwm2_fc01 fbpspwmdt** [Instantiation]

Definition at line 606 of file [DE0_NANO_VF.vhd](#).

6.22.3.71 **pwm2_fc02 fbpspwmdt** [Instantiation]

Definition at line 629 of file [DE0_NANO_VF.vhd](#).

6.22.3.72 **pwm2_fc03 fbpspwmdt** [Instantiation]

Definition at line 653 of file [DE0_NANO_VF.vhd](#).

6.22.3.73 **reset std_logic** [Signal]

Definition at line 202 of file [DE0_NANO_VF.vhd](#).

6.22.3.74 **rst std_logic:='1'** [Signal]

Definition at line 215 of file [DE0_NANO_VF.vhd](#).

6.22.3.75 **sigPWM01 std_logic** [Signal]

Definition at line 207 of file [DE0_NANO_VF.vhd](#).

6.22.3.76 **sigPWM02 std_logic** [Signal]

Definition at line 207 of file [DE0_NANO_VF.vhd](#).

6.22.3.77 **sin_a std_logic_vector(n_bits_c - 1 downto 0)** [Signal]

Definition at line 226 of file [DE0_NANO_VF.vhd](#).

6.22.3.78 **sin_b std_logic_vector(n_bits_c - 1 downto 0)** [Signal]

Definition at line 226 of file [DE0_NANO_VF.vhd](#).

6.22.3.79 **sin_c std_logic_vector(n_bits_c - 1 downto 0)** [Signal]

Definition at line 226 of file [DE0_NANO_VF.vhd](#).

6.22.3.80 **sinc_int std_logic** [Signal]

Definition at line 210 of file [DE0_NANO_VF.vhd](#).

6.22.3.81 **sinc_wt std_logic** [Signal]

Definition at line 210 of file [DE0_NANO_VF.vhd](#).

6.22.3.82 **tabela_sin** [Component]

Definition at line 105 of file [DE0_NANO_VF.vhd](#).

6.22.3.83 **th_a std_logic_vector(15 downto 0)** [Signal]

Definition at line 205 of file [DE0_NANO_VF.vhd](#).

6.22.3.84 **th_b std_logic_vector(15 downto 0)** [Signal]

Definition at line 205 of file [DE0_NANO_VF.vhd](#).

6.22.3.85 **th_c std_logic_vector(15 downto 0)** [Signal]

Definition at line 205 of file [DE0_NANO_VF.vhd](#).

6.22.3.86 **theta_abc** [Component]

Definition at line 140 of file [DE0_NANO_VF.vhd](#).

6.22.3.87 **theta_pll** **std_logic_vector**(Nout - 1 **downto** 0) [Signal]

Definition at line 243 of file [DE0_NANO_VF.vhd](#).

6.22.3.88 **theta_wt** **std_logic_vector**(15 **downto** 0) [Signal]

Definition at line 244 of file [DE0_NANO_VF.vhd](#).

6.22.3.89 **toggle_key0** **std_logic**:= '1' [Signal]

Definition at line 213 of file [DE0_NANO_VF.vhd](#).

6.22.3.90 **toggle_key1** **std_logic**:= '1' [Signal]

Definition at line 213 of file [DE0_NANO_VF.vhd](#).

6.22.3.91 **u1 clk_div** [Instantiation]

Definition at line 262 of file [DE0_NANO_VF.vhd](#).

6.22.3.92 **u5 integrador** [Instantiation]

Definition at line 295 of file [DE0_NANO_VF.vhd](#).

6.22.3.93 **u6 theta_abc** [Instantiation]

Definition at line 376 of file [DE0_NANO_VF.vhd](#).

6.22.3.94 **uclkvf clk_div** [Instantiation]

Definition at line 269 of file [DE0_NANO_VF.vhd](#).

6.22.3.95 **ucr1 portadora_tringular** [Instantiation]

Definition at line 308 of file [DE0_NANO_VF.vhd](#).

6.22.3.96 **ucr2 portadora_tringular** [Instantiation]

Definition at line 319 of file [DE0_NANO_VF.vhd](#).

6.22.3.97 **ucr3 portadora_tringular** [Instantiation]

Definition at line 330 of file [DE0_NANO_VF.vhd](#).

6.22.3.98 **ucr4 portadora_tringular** [Instantiation]

Definition at line 341 of file [DE0_NANO_VF.vhd](#).

6.22.3.99 **ucr5 portadora_tringular** [Instantiation]

Definition at line 352 of file [DE0_NANO_VF.vhd](#).

6.22.3.100 **ucr6 portadora_tringular** [Instantiation]

Definition at line 363 of file [DE0_NANO_VF.vhd](#).

6.22.3.101 **upll pll** [Instantiation]

Definition at line 254 of file [DE0_NANO_VF.vhd](#).

6.22.3.102 **usin_a tabela_sin** [Instantiation]

Definition at line 384 of file [DE0_NANO_VF.vhd](#).

6.22.3.103 **usin_b tabela_sin** [Instantiation]

Definition at line 391 of file [DE0_NANO_VF.vhd](#).

6.22.3.104 **usin_c tabela_sin** [Instantiation]

Definition at line 399 of file [DE0_NANO_VF.vhd](#).

6.22.3.105 **uvf vfcontrol** [Instantiation]

Definition at line 280 of file [DE0_NANO_VF.vhd](#).

6.22.3.106 **vfcontrol** [Component]

Definition at line 189 of file [DE0_NANO_VF.vhd](#).

The documentation for this class was generated from the following file:

- [DE0_NANO_VF.vhd](#)

6.23 my_types_pkg Package Reference

Libraries

- [IEEE](#)

Use Clauses

- [STD_LOGIC_UNSIGNED](#)
- [STD_LOGIC_1164](#)

Types

- WORD_ARRAY(31 downto 0)std_logic_vector(15 downto 0)
- ARRAY_8_24(24 downto 1)std_logic_vector(7 downto 0)
- ARRAY_9_24(24 downto 1)std_logic_vector(8 downto 0)
- ARRAY_10_24(24 downto 1)std_logic_vector(9 downto 0)
- ARRAY_12_24(24 downto 1)std_logic_vector(11 downto 0)
- ARRAY_24_24(24 downto 1)std_logic_vector(23 downto 0)
- ARRAY_2_24(24 downto 1)std_logic_vector(1 downto 0)
- ARRAY_21_24(24 downto 1)std_logic_vector(20 downto 0)
- ARRAY_16_24(24 downto 1)std_logic_vector(15 downto 0)
- ARRAY_17_24(24 downto 1)std_logic_vector(16 downto 0)
- ARRAY_18_24(24 downto 1)std_logic_vector(17 downto 0)
- ARRAY_20_24(24 downto 1)std_logic_vector(19 downto 0)
- COMP_ARRAY(24 downto 1)std_logic_vector(21 downto 0)
- ARRAY_8_8(7 downto 0)std_logic_vector(7 downto 0)
- ARRAY_10_6(6 downto 1)std_logic_vector(9 downto 0)
- ARRAY_12_6(6 downto 1)std_logic_vector(11 downto 0)
- ARRAY_9_6(6 downto 1)std_logic_vector(8 downto 0)
- ARRAY_3_6(6 downto 1)std_logic_vector(2 downto 0)
- ARRAY_4_6(6 downto 1)std_logic_vector(3 downto 0)
- ARRAY_8_6(6 downto 1)std_logic_vector(7 downto 0)
- ARRAY_8_5_0(5 downto 0)std_logic_vector(7 downto 0)
- ARRAY_16_16(16 downto 1)std_logic_vector(15 downto 0)
- ARRAY_3_16(16 downto 1)std_logic_vector(2 downto 0)
- ARRAY_16_6(6 downto 1)std_logic_vector(15 downto 0)
- ARRAY_16x6_12(12 downto 1)ARRAY_16_6
- OPP_GAMMA(27 downto 1)std_logic_vector(28 downto 0)
- OPP_PULSO(27 downto 1)std_logic_vector(2 downto 0)

6.23.1 Detailed Description

Definition at line 7 of file [my_types_pkg.vhd](#).

6.23.2 Member Data Documentation

6.23.2.1 ARRAY_10_24 (24 downto 1)std_logic_vector(9 downto 0) [Type]

Definition at line 12 of file [my_types_pkg.vhd](#).

6.23.2.2 ARRAY_10_6 (6 downto 1)std_logic_vector(9 downto 0) [Type]

Definition at line 24 of file [my_types_pkg.vhd](#).

6.23.2.3 ARRAY_12_24 (24 downto 1)std_logic_vector(11 downto 0) [Type]

Definition at line 13 of file [my_types_pkg.vhd](#).

6.23.2.4 ARRAY_12_6 (6 downto 1)std_logic_vector(11 downto 0) [Type]

Definition at line 25 of file [my_types_pkg.vhd](#).

6.23.2.5 **ARRAY_16_16(16 downto 1)std_logic_vector(15 downto 0)** [Type]

Definition at line 33 of file [my_types_pkg.vhd](#).

6.23.2.6 **ARRAY_16_24(24 downto 1)std_logic_vector(15 downto 0)** [Type]

Definition at line 17 of file [my_types_pkg.vhd](#).

6.23.2.7 **ARRAY_16_6(6 downto 1)std_logic_vector(15 downto 0)** [Type]

Definition at line 36 of file [my_types_pkg.vhd](#).

6.23.2.8 **ARRAY_16x6_12(12 downto 1)ARRAY_16_6** [Type]

Definition at line 37 of file [my_types_pkg.vhd](#).

6.23.2.9 **ARRAY_17_24(24 downto 1)std_logic_vector(16 downto 0)** [Type]

Definition at line 18 of file [my_types_pkg.vhd](#).

6.23.2.10 **ARRAY_18_24(24 downto 1)std_logic_vector(17 downto 0)** [Type]

Definition at line 19 of file [my_types_pkg.vhd](#).

6.23.2.11 **ARRAY_20_24(24 downto 1)std_logic_vector(19 downto 0)** [Type]

Definition at line 20 of file [my_types_pkg.vhd](#).

6.23.2.12 **ARRAY_21_24(24 downto 1)std_logic_vector(20 downto 0)** [Type]

Definition at line 16 of file [my_types_pkg.vhd](#).

6.23.2.13 **ARRAY_24_24(24 downto 1)std_logic_vector(23 downto 0)** [Type]

Definition at line 14 of file [my_types_pkg.vhd](#).

6.23.2.14 **ARRAY_2_24(24 downto 1)std_logic_vector(1 downto 0)** [Type]

Definition at line 15 of file [my_types_pkg.vhd](#).

6.23.2.15 **ARRAY_3_16(16 downto 1)std_logic_vector(2 downto 0)** [Type]

Definition at line 34 of file [my_types_pkg.vhd](#).

6.23.2.16 **ARRAY_3_6(6 downto 1)std_logic_vector(2 downto 0)** [Type]

Definition at line 27 of file [my_types_pkg.vhd](#).

6.23.2.17 **ARRAY_4_6** (**6** **downto** **1**)**std_logic_vector**(**3** **downto** **0**) [Type]

Definition at line [28](#) of file [my_types_pkg.vhd](#).

6.23.2.18 **ARRAY_8_24** (**24** **downto** **1**)**std_logic_vector**(**7** **downto** **0**) [Type]

Definition at line [10](#) of file [my_types_pkg.vhd](#).

6.23.2.19 **ARRAY_8_5_0** (**5** **downto** **0**)**std_logic_vector**(**7** **downto** **0**) [Type]

Definition at line [30](#) of file [my_types_pkg.vhd](#).

6.23.2.20 **ARRAY_8_6** (**6** **downto** **1**)**std_logic_vector**(**7** **downto** **0**) [Type]

Definition at line [29](#) of file [my_types_pkg.vhd](#).

6.23.2.21 **ARRAY_8_8** (**7** **downto** **0**)**std_logic_vector**(**7** **downto** **0**) [Type]

Definition at line [23](#) of file [my_types_pkg.vhd](#).

6.23.2.22 **ARRAY_9_24** (**24** **downto** **1**)**std_logic_vector**(**8** **downto** **0**) [Type]

Definition at line [11](#) of file [my_types_pkg.vhd](#).

6.23.2.23 **ARRAY_9_6** (**6** **downto** **1**)**std_logic_vector**(**8** **downto** **0**) [Type]

Definition at line [26](#) of file [my_types_pkg.vhd](#).

6.23.2.24 **COMP_ARRAY** (**24** **downto** **1**)**std_logic_vector**(**21** **downto** **0**) [Type]

Definition at line [22](#) of file [my_types_pkg.vhd](#).

6.23.2.25 **IEEE** [Library]

Definition at line [1](#) of file [my_types_pkg.vhd](#).

6.23.2.26 **OPP_GAMMA** (**27** **downto** **1**)**std_logic_vector**(**28** **downto** **0**) [Type]

Definition at line [41](#) of file [my_types_pkg.vhd](#).

6.23.2.27 **OPP_PULSO** (**27** **downto** **1**)**std_logic_vector**(**2** **downto** **0**) [Type]

Definition at line [42](#) of file [my_types_pkg.vhd](#).

6.23.2.28 **STD_LOGIC_1164** [Package]

Definition at line [3](#) of file [my_types_pkg.vhd](#).

6.23.2.29 STD_LOGIC_UNSIGNED [Package]

Definition at line 2 of file [my_types_pkg.vhd](#).

6.23.2.30 WORD_ARRAY(31 downto 0)std_logic_vector(15 downto 0) [Type]

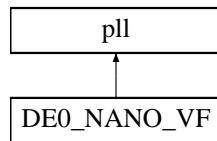
Definition at line 8 of file [my_types_pkg.vhd](#).

The documentation for this class was generated from the following file:

- [my_types_pkg.vhd](#)

6.24 pll Entity Reference

Inheritance diagram for pll:



Entities

- [SYN](#) architecture

Libraries

- [ieee](#)
- [altera_mf](#)

Use Clauses

- [std_logic_1164](#)
- [all](#)

Ports

- [areset in STD_LOGIC:='0'](#)
- [inclk0 in STD_LOGIC:='0'](#)
- [c0 out STD_LOGIC](#)
- [locked out STD_LOGIC](#)

6.24.1 Detailed Description

Definition at line 43 of file [pll.vhd](#).

6.24.2 Member Data Documentation

6.24.2.1 **all** [Package]

Definition at line 41 of file [pll.vhd](#).

6.24.2.2 **altera_mf** [Library]

Definition at line 40 of file [pll.vhd](#).

6.24.2.3 **areset** in STD_LOGIC:=‘0’ [Port]

Definition at line 46 of file [pll.vhd](#).

6.24.2.4 **c0** out STD_LOGIC [Port]

Definition at line 48 of file [pll.vhd](#).

6.24.2.5 **ieee** [Library]

Definition at line 37 of file [pll.vhd](#).

6.24.2.6 **inclk0** in STD_LOGIC:=‘0’ [Port]

Definition at line 47 of file [pll.vhd](#).

6.24.2.7 **locked** out STD_LOGIC [Port]

Definition at line 50 of file [pll.vhd](#).

6.24.2.8 **std_logic_1164** [Package]

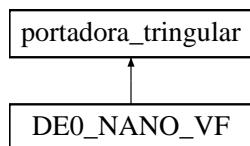
Definition at line 38 of file [pll.vhd](#).

The documentation for this class was generated from the following file:

- [pll.vhd](#)

6.25 portadora_tringular Entity Reference

Inheritance diagram for portadora_tringular:



Entities

- [portadora_tringular](#) architecture

Libraries

- IEEE

Use Clauses

- STD_LOGIC_UNSIGNED
- STD_LOGIC_1164
- fixed_pkg

Generics

- N integer:= 16

Ports

- clk in std_logic
- en in std_logic
- reset in std_logic
- count_ini in std_logic_vector(N - 1 downto 0)
- dir_ini in std_logic
- MAX in std_logic_vector(N - 1 downto 0)
- dir out std_logic
- c out std_logic_vector(N - 1 downto 0)

6.25.1 Detailed Description

Definition at line 6 of file portadora_tringular.vhd.

6.25.2 Member Data Documentation

6.25.2.1 c out std_logic_vector(N - 1 downto 0) [Port]

Definition at line 18 of file portadora_tringular.vhd.

6.25.2.2 clk in std_logic [Port]

Definition at line 10 of file portadora_tringular.vhd.

6.25.2.3 count_ini in std_logic_vector(N - 1 downto 0) [Port]

Definition at line 13 of file portadora_tringular.vhd.

6.25.2.4 dir out std_logic [Port]

Definition at line 16 of file portadora_tringular.vhd.

6.25.2.5 dir_ini in std_logic [Port]

Definition at line 14 of file portadora_tringular.vhd.

6.25.2.6 en **in std_logic** [Port]

Definition at line 11 of file [portadora_tringular.vhd](#).

6.25.2.7 fixed_pkg [Package]

Definition at line 4 of file [portadora_tringular.vhd](#).

6.25.2.8 IEEE [Library]

Definition at line 1 of file [portadora_tringular.vhd](#).

6.25.2.9 MAX **in std_logic_vector(N - 1 downto 0)** [Port]

Definition at line 15 of file [portadora_tringular.vhd](#).

6.25.2.10 N **integer:= 16** [Generic]

Definition at line 8 of file [portadora_tringular.vhd](#).

6.25.2.11 reset **in std_logic** [Port]

Definition at line 12 of file [portadora_tringular.vhd](#).

6.25.2.12 STD_LOGIC_1164 [Package]

Definition at line 3 of file [portadora_tringular.vhd](#).

6.25.2.13 STD_LOGIC_UNSIGNED [Package]

Definition at line 2 of file [portadora_tringular.vhd](#).

The documentation for this class was generated from the following file:

- [portadora_tringular.vhd](#)

6.26 portadora_tringular Architecture Reference

Processes

- [PROCESS_12\(clk \)](#)

Signals

- [c_int std_logic_vector\(N - 1 downto 0\)](#)
- [dir_int std_logic](#)

6.26.1 Detailed Description

Definition at line 23 of file [portadora_tringular.vhd](#).

6.26.2 Member Function Documentation

6.26.2.1 `PROCESS_12(clk)` [Process]

Definition at line 31 of file [portadora_tringular.vhd](#).

6.26.3 Member Data Documentation

6.26.3.1 `c_int std_logic_vector(N - 1 downto 0)` [Signal]

Definition at line 25 of file [portadora_tringular.vhd](#).

6.26.3.2 `dir_int std_logic` [Signal]

Definition at line 26 of file [portadora_tringular.vhd](#).

The documentation for this class was generated from the following file:

- [portadora_tringular.vhd](#)

6.27 sinewave Entity Reference

Entities

- [Behavioral](#) architecture

Libraries

- [IEEE](#)

Use Clauses

- [STD_LOGIC_1164](#)
- [NUMERIC_STD](#)

Ports

- `clk in std_logic`
- `dataout out integer range -128 to 127`

6.27.1 Detailed Description

Definition at line 6 of file [sinewave.vhd](#).

6.27.2 Member Data Documentation

6.27.2.1 `clk in std_logic` [Port]

Definition at line 7 of file [sinewave.vhd](#).

6.27.2.2 **dataout** **out** **integer** **range** **-128** **to** **127** [Port]

Definition at line 9 of file [sinewave.vhd](#).

6.27.2.3 **IEEE** [Library]

Definition at line 2 of file [sinewave.vhd](#).

6.27.2.4 **NUMERIC_STD** [Package]

Definition at line 4 of file [sinewave.vhd](#).

6.27.2.5 **STD_LOGIC_1164** [Package]

Definition at line 3 of file [sinewave.vhd](#).

The documentation for this class was generated from the following file:

- [sinewave.vhd](#)

6.28 SYN Architecture Reference

Components

- [altpll](#)

Signals

- [sub_wire0 STD_LOGIC](#)
- [sub_wire1 STD_LOGIC_VECTOR\(1 DOWNTO 0 \)](#)
- [sub_wire2_bv BIT_VECTOR\(0 DOWNTO 0 \)](#)
- [sub_wire2 STD_LOGIC_VECTOR\(0 DOWNTO 0 \)](#)
- [sub_wire3 STD_LOGIC_VECTOR\(4 DOWNTO 0 \)](#)
- [sub_wire4 STD_LOGIC](#)
- [sub_wire5 STD_LOGIC](#)

Instantiations

- [altpll_component altpll](#)

6.28.1 Detailed Description

Definition at line 54 of file [pll.vhd](#).

6.28.2 Member Data Documentation

6.28.2.1 **altpll** [Component]

Definition at line 66 of file [pll.vhd](#).

6.28.2.2 altpll_component altpll [Instantiation]

Definition at line 204 of file [pll.vhd](#).

6.28.2.3 sub_wire0 STD_LOGIC [Signal]

Definition at line 56 of file [pll.vhd](#).

6.28.2.4 sub_wire1 STD_LOGIC_VECTOR(1 DOWNTO 0) [Signal]

Definition at line 57 of file [pll.vhd](#).

6.28.2.5 sub_wire2 STD_LOGIC_VECTOR(0 DOWNTO 0) [Signal]

Definition at line 59 of file [pll.vhd](#).

6.28.2.6 sub_wire2_bv BIT_VECTOR(0 DOWNTO 0) [Signal]

Definition at line 58 of file [pll.vhd](#).

6.28.2.7 sub_wire3 STD_LOGIC_VECTOR(4 DOWNTO 0) [Signal]

Definition at line 60 of file [pll.vhd](#).

6.28.2.8 sub_wire4 STD_LOGIC [Signal]

Definition at line 61 of file [pll.vhd](#).

6.28.2.9 sub_wire5 STD_LOGIC [Signal]

Definition at line 62 of file [pll.vhd](#).

The documentation for this class was generated from the following file:

- [pll.vhd](#)

6.29 tabela_sin Architecture Reference

Processes

- [PROCESS_14\(clk \)](#)

Signals

- [id std_logic_vector\(10 downto 0 \)](#)
- [sin sfixed\(1 downto- F \)](#)
- [va_Q14 sfixed\(1 downto- 14 \)](#)

6.29.1 Detailed Description

Definition at line 23 of file [tabela_sin.vhd](#).

6.29.2 Member Function Documentation

6.29.2.1 PROCESS_14(clk) [Process]

Definition at line 33 of file [tabela_sin.vhd](#).

6.29.3 Member Data Documentation

6.29.3.1 id std_logic_vector(10 downto 0) [Signal]

Definition at line 24 of file [tabela_sin.vhd](#).

6.29.3.2 sin sfixed(I downto-F) [Signal]

Definition at line 25 of file [tabela_sin.vhd](#).

6.29.3.3 va_Q14 sfixed(1 downto- 14) [Signal]

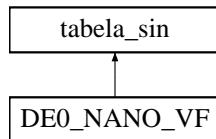
Definition at line 26 of file [tabela_sin.vhd](#).

The documentation for this class was generated from the following file:

- [tabela_sin.vhd](#)

6.30 tabela_sin Entity Reference

Inheritance diagram for tabela_sin:



Entities

- [tabela_sin](#) architecture

Libraries

- [ieee](#)

Use Clauses

- [std_logic_1164](#)
- [std_logic_unsigned](#)

- numeric_std
- fixed_pkg

Generics

- n_bits_phase integer:= 16
- n_bits_c integer:= 16
- l integer:= 1
- F integer:= 14

Ports

- clk in std_logic
- theta in std_logic_vector(15 downto 0)
- MAX in sfixed(15 downto 0)
- ma in sfixed(l downto-F)
- va out std_logic_vector(15 downto 0)

6.30.1 Detailed Description

Definition at line 7 of file [tabela_sin.vhd](#).

6.30.2 Member Data Documentation

6.30.2.1 clk in std_logic [Port]

Definition at line 15 of file [tabela_sin.vhd](#).

6.30.2.2 F integer:= 14 [Generic]

Definition at line 13 of file [tabela_sin.vhd](#).

6.30.2.3 fixed_pkg [Package]

Definition at line 5 of file [tabela_sin.vhd](#).

6.30.2.4 l integer:= 1 [Generic]

Definition at line 11 of file [tabela_sin.vhd](#).

6.30.2.5 ieee [Library]

Definition at line 1 of file [tabela_sin.vhd](#).

6.30.2.6 ma in sfixed(l downto-F) [Port]

Definition at line 18 of file [tabela_sin.vhd](#).

6.30.2.7 **MAX** in **sfixed(15 downto 0)** [Port]

Definition at line 17 of file [tabela_sin.vhd](#).

6.30.2.8 **n_bits_c** **integer:= 16** [Generic]

Definition at line 10 of file [tabela_sin.vhd](#).

6.30.2.9 **n_bits_phase** **integer:= 16** [Generic]

Definition at line 9 of file [tabela_sin.vhd](#).

6.30.2.10 **numeric_std** [Package]

Definition at line 4 of file [tabela_sin.vhd](#).

6.30.2.11 **std_logic_1164** [Package]

Definition at line 2 of file [tabela_sin.vhd](#).

6.30.2.12 **std_logic_unsigned** [Package]

Definition at line 3 of file [tabela_sin.vhd](#).

6.30.2.13 **theta** in **std_logic_vector(15 downto 0)** [Port]

Definition at line 16 of file [tabela_sin.vhd](#).

6.30.2.14 **va** out **std_logic_vector(15 downto 0)** [Port]

Definition at line 20 of file [tabela_sin.vhd](#).

The documentation for this class was generated from the following file:

- [tabela_sin.vhd](#)

6.31 theta_abc Architecture Reference

Processes

- [PROCESS_15\(clk \)](#)

Signals

- [th_a std_logic_vector\(Nout - 1 downto 0\)](#)
- [th_b std_logic_vector\(Nout - 1 downto 0\)](#)
- [th_c std_logic_vector\(Nout - 1 downto 0\)](#)
- [th_ai std_logic_vector\(Nout downto 0\)](#)
- [th_bi std_logic_vector\(Nout downto 0\)](#)
- [th_ci std_logic_vector\(Nout downto 0\)](#)

6.31.1 Detailed Description

Definition at line 25 of file [theta_abc.vhd](#).

6.31.2 Member Function Documentation

6.31.2.1 PROCESS_15(clk) [Process]

Definition at line 33 of file [theta_abc.vhd](#).

6.31.3 Member Data Documentation

6.31.3.1 th_a std_logic_vector(Nout - 1 downto 0) [Signal]

Definition at line 27 of file [theta_abc.vhd](#).

6.31.3.2 th_ai std_logic_vector(Nout downto 0) [Signal]

Definition at line 28 of file [theta_abc.vhd](#).

6.31.3.3 th_b std_logic_vector(Nout - 1 downto 0) [Signal]

Definition at line 27 of file [theta_abc.vhd](#).

6.31.3.4 th_bi std_logic_vector(Nout downto 0) [Signal]

Definition at line 28 of file [theta_abc.vhd](#).

6.31.3.5 th_c std_logic_vector(Nout - 1 downto 0) [Signal]

Definition at line 27 of file [theta_abc.vhd](#).

6.31.3.6 th_ci std_logic_vector(Nout downto 0) [Signal]

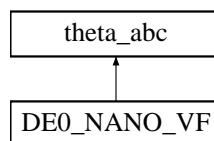
Definition at line 28 of file [theta_abc.vhd](#).

The documentation for this class was generated from the following file:

- [theta_abc.vhd](#)

6.32 theta_abc Entity Reference

Inheritance diagram for theta_abc:



Entities

- `theta_abc` architecture

Libraries

- IEEE

Use Clauses

- STD_LOGIC_UNSIGNED
- STD_LOGIC_1164
- fixed_pkg
- numeric_std

Generics

- `Nin integer:= 30`
- `Nout integer:= 16`

Ports

- `clk in std_logic`
- `en in std_logic`
- `reset in std_logic`
- `theta_a out std_logic_vector(Nout - 1 downto 0)`
- `theta_b out std_logic_vector(Nout - 1 downto 0)`
- `theta_c out std_logic_vector(Nout - 1 downto 0)`
- `theta_in in std_logic_vector(Nin - 1 downto 0)`

6.32.1 Detailed Description

Definition at line 8 of file `theta_abc.vhd`.

6.32.2 Member Data Documentation

6.32.2.1 `clk in std_logic` [Port]

Definition at line 13 of file `theta_abc.vhd`.

6.32.2.2 `en in std_logic` [Port]

Definition at line 14 of file `theta_abc.vhd`.

6.32.2.3 `fixed_pkg` [Package]

Definition at line 4 of file `theta_abc.vhd`.

6.32.2.4 `IEEE` [Library]

Definition at line 1 of file `theta_abc.vhd`.

6.32.2.5 **Nin** **integer:= 30** [Generic]

Definition at line 9 of file [theta_abc.vhd](#).

6.32.2.6 **Nout** **integer:= 16** [Generic]

Definition at line 11 of file [theta_abc.vhd](#).

6.32.2.7 **numeric_std** [Package]

Definition at line 5 of file [theta_abc.vhd](#).

6.32.2.8 **reset** **in std_logic** [Port]

Definition at line 15 of file [theta_abc.vhd](#).

6.32.2.9 **STD_LOGIC_1164** [Package]

Definition at line 3 of file [theta_abc.vhd](#).

6.32.2.10 **STD_LOGIC_UNSIGNED** [Package]

Definition at line 2 of file [theta_abc.vhd](#).

6.32.2.11 **theta_a** **out std_logic_vector(Nout - 1 downto 0)** [Port]

Definition at line 16 of file [theta_abc.vhd](#).

6.32.2.12 **theta_b** **out std_logic_vector(Nout - 1 downto 0)** [Port]

Definition at line 17 of file [theta_abc.vhd](#).

6.32.2.13 **theta_c** **out std_logic_vector(Nout - 1 downto 0)** [Port]

Definition at line 18 of file [theta_abc.vhd](#).

6.32.2.14 **theta_in** **in std_logic_vector(Nin - 1 downto 0)** [Port]

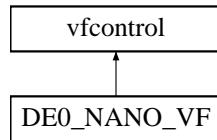
Definition at line 20 of file [theta_abc.vhd](#).

The documentation for this class was generated from the following file:

- [theta_abc.vhd](#)

6.33 vfcontrol Entity Reference

Inheritance diagram for vfcontrol:



Entities

- [vfcontrol_arch](#) architecture

Libraries

- [IEEE](#)

Use Clauses

- [STD_LOGIC_UNSIGNED](#)
- [STD_LOGIC_1164](#)
- [STD_LOGIC_ARITH](#)
- [fixed_pkg](#)
- [numeric_std](#)

Generics

- `n_bits_c integer:= 16`
- `incMAX std_logic_vector(12 downto 0):=std_logic_vector(to_unsigned(2416 , 13))`
- `incMIN std_logic_vector(12 downto 0):=std_logic_vector(to_unsigned(483 , 13))`
- `I integer:= 1`
- `F integer:= 14`
- `mMAX sfixed(1 downto - 27):=to_sfixed(0 . 4069 , 1 ,- 27)`
- `mMIN sfixed(1 downto - 27):=to_sfixed(0 . 08137 , 1 ,- 27)`

Ports

- `clk in std_logic`
- `en in std_logic`
- `inc_data out std_logic_vector(12 downto 0)`
- `m_vf out sfixed(I downto-F)`

6.33.1 Detailed Description

Definition at line 10 of file [vfcontrol.vhd](#).

6.33.2 Member Data Documentation

6.33.2.1 `clk in std_logic [Port]`

Definition at line 24 of file [vfcontrol.vhd](#).

6.33.2.2 **en** **in std_logic** [Port]

Definition at line 25 of file vfcontrol.vhd.

6.33.2.3 **F integer:= 14** [Generic]

Definition at line 17 of file vfcontrol.vhd.

6.33.2.4 **fixed_pkg** [Package]

Definition at line 6 of file vfcontrol.vhd.

6.33.2.5 **I integer:= 1** [Generic]

Definition at line 16 of file vfcontrol.vhd.

6.33.2.6 **IEEE** [Library]

Definition at line 2 of file vfcontrol.vhd.

6.33.2.7 **inc_data out std_logic_vector(12 downto 0)** [Port]

Definition at line 26 of file vfcontrol.vhd.

6.33.2.8 **incMAX std_logic_vector(12 downto 0):=std_logic_vector(to_unsigned(2416 , 13))**
[Generic]

Definition at line 14 of file vfcontrol.vhd.

6.33.2.9 **incMIN std_logic_vector(12 downto 0):=std_logic_vector(to_unsigned(483 , 13))** [Generic]

Definition at line 15 of file vfcontrol.vhd.

6.33.2.10 **m_vf out sfixed(I downto-F)** [Port]

Definition at line 28 of file vfcontrol.vhd.

6.33.2.11 **mMAX sfixed(1 downto- 27):=to_sfixed(0 . 4069 ,1,- 27)** [Generic]

Definition at line 19 of file vfcontrol.vhd.

6.33.2.12 **mMIN sfixed(1 downto- 27):=to_sfixed(0 . 08137 ,1,- 27)** [Generic]

Definition at line 21 of file vfcontrol.vhd.

6.33.2.13 **n_bits_c integer:= 16** [Generic]

Definition at line 12 of file vfcontrol.vhd.

6.33.2.14 numeric_std [Package]

Definition at line 7 of file vfcontrol.vhd.

6.33.2.15 STD_LOGIC_1164 [Package]

Definition at line 4 of file vfcontrol.vhd.

6.33.2.16 STD_LOGIC_ARITH [Package]

Definition at line 5 of file vfcontrol.vhd.

6.33.2.17 STD_LOGIC_UNSIGNED [Package]

Definition at line 3 of file vfcontrol.vhd.

The documentation for this class was generated from the following file:

- [vfcontrol.vhd](#)

6.34 vfcontrol_arch Architecture Reference

Processes

- [PROCESS_16\(clk \)](#)

Signals

- [incsignal std_logic_vector\(12 downto 0 \):=std_logic_vector\(to_unsigned\(0 , 13 \)\)](#)
- [msignal sfixed\(1 downto- 27 \):=to_sfixed\(0 . 08137 , 1 ,- 27 \)](#)
- [incstep std_logic_vector\(12 downto 0 \):=std_logic_vector\(to_unsigned\(1 , 13 \)\)](#)
- [mstep sfixed\(1 downto- 27 \):=to_sfixed\(1 . 6839e - 04 , 1 ,- 27 \)](#)

6.34.1 Detailed Description

Definition at line 35 of file vfcontrol.vhd.

6.34.2 Member Function Documentation

6.34.2.1 PROCESS_16(clk) [Process]

Definition at line 44 of file vfcontrol.vhd.

6.34.3 Member Data Documentation

6.34.3.1 incsignal std_logic_vector(12 downto 0):=std_logic_vector(to_unsigned(0 , 13)) [Signal]

Definition at line 37 of file vfcontrol.vhd.

6.34.3.2 incstep **std_logic_vector(12 downto 0):=std_logic_vector(to_unsigned(1 , 13))** [Signal]

Definition at line 39 of file [vfcontrol.vhd](#).

6.34.3.3 msignal **sfixed(1 downto- 27):=to_sfixed(0 . 08137 , 1 ,- 27)** [Signal]

Definition at line 38 of file [vfcontrol.vhd](#).

6.34.3.4 mstep **sfixed(1 downto- 27):=to_sfixed(1 . 6839e - 04 , 1 ,- 27)** [Signal]

Definition at line 40 of file [vfcontrol.vhd](#).

The documentation for this class was generated from the following file:

- [vfcontrol.vhd](#)

6.35 wt Entity Reference

Entities

- [wt_arch](#) architecture

Libraries

- [IEEE](#)

Use Clauses

- [STD_LOGIC_UNSIGNED](#)
- [STD_LOGIC_1164](#)
- [fixed_pkg](#)

Generics

- [Nbits integer:= 16](#)

Ports

- [clk in std_logic](#)
- [en in std_logic](#)
- [reset in std_logic](#)
- [sinc out std_logic](#)
- [MAX in std_logic_vector\(Nbits - 1 downto 0 \)](#)
- [out_data out std_logic_vector\(Nbits - 1 downto 0 \)](#)
- [int_data in std_logic_vector\(Nbits - 1 downto 0 \)](#)

6.35.1 Detailed Description

Definition at line 8 of file [wt.vhd](#).

6.35.2 Member Data Documentation

6.35.2.1 **clk** **in std_logic** [Port]

Definition at line 13 of file [wt.vhd](#).

6.35.2.2 **en** **in std_logic** [Port]

Definition at line 14 of file [wt.vhd](#).

6.35.2.3 **fixed_pkg** [Package]

Definition at line 4 of file [wt.vhd](#).

6.35.2.4 **IEEE** [Library]

Definition at line 1 of file [wt.vhd](#).

6.35.2.5 **int_data** **in std_logic_vector(Nbits - 1 downto 0)** [Port]

Definition at line 20 of file [wt.vhd](#).

6.35.2.6 **MAX** **in std_logic_vector(Nbits - 1 downto 0)** [Port]

Definition at line 17 of file [wt.vhd](#).

6.35.2.7 **Nbits integer:= 16** [Generic]

Definition at line 11 of file [wt.vhd](#).

6.35.2.8 **out_data** **out std_logic_vector(Nbits - 1 downto 0)** [Port]

Definition at line 18 of file [wt.vhd](#).

6.35.2.9 **reset** **in std_logic** [Port]

Definition at line 15 of file [wt.vhd](#).

6.35.2.10 **sinc** **out std_logic** [Port]

Definition at line 16 of file [wt.vhd](#).

6.35.2.11 **STD_LOGIC_1164** [Package]

Definition at line 3 of file [wt.vhd](#).

6.35.2.12 STD_LOGIC_UNSIGNED [Package]

Definition at line 2 of file [wt.vhd](#).

The documentation for this class was generated from the following file:

- [wt.vhd](#)

6.36 wt_arch Architecture Reference

Processes

- [PROCESS_17\(clk \)](#)

Signals

- [out_int std_logic_vector\(Nbits - 1 downto 0\)](#)
- [sinc_int std_logic](#)

6.36.1 Detailed Description

Definition at line 25 of file [wt.vhd](#).

6.36.2 Member Function Documentation

6.36.2.1 PROCESS_17(clk) [Process]

Definition at line 33 of file [wt.vhd](#).

6.36.3 Member Data Documentation

6.36.3.1 out_int std_logic_vector(Nbits - 1 downto 0) [Signal]

Definition at line 27 of file [wt.vhd](#).

6.36.3.2 sinc_int std_logic [Signal]

Definition at line 28 of file [wt.vhd](#).

The documentation for this class was generated from the following file:

- [wt.vhd](#)

Chapter 7

File Documentation

7.1 clk_div.vhd File Reference

Entities

- `clk_div` entity
- `div` architecture

7.2 clk_div.vhd

```
00001 library IEEE;
00002 use IEEE.STD_LOGIC_UNSIGNED.all;
00003 use IEEE.STD_LOGIC_1164.all;
00004
00005 entity clk_div is
00006     port(
00007         clk_in, en : IN STD_LOGIC;
00008         div : IN STD_LOGIC_VECTOR(15 DOWNTO 0); --f_out = f_in/(2*div)
00009         clk_out: OUT STD_LOGIC
00010     );
00011 END ENTITY clk_div;
00012
00013 ARCHITECTURE div OF clk_div IS
00014     SIGNAL count : STD_LOGIC_VECTOR(15 DOWNTO 0);
00015     SIGNAL clk_out_bi : STD_LOGIC;
00016 BEGIN
00017
00018     PROCESS(clk_in,en,div)
00019         BEGIN
00020             IF en = '0' THEN
00021                 --count <= (OTHERS => '0');
00022                 count <= (div-1);
00023                 clk_out_bi <= '0';
00024             ELSIF rising_edge(clk_in) THEN
00025                 IF count < (div - 1) THEN
00026                     count <= count + 1;
00027                 ELSE
00028                     count <= (OTHERS => '0');
00029                     clk_out_bi <= NOT clk_out_bi;
00030                 END IF;
00031             END IF;
00032
00033         END PROCESS;
00034         clk_out <= clk_out_bi;
00035
00036 END ARCHITECTURE div;
```

7.3 comparador.vhd File Reference

Entities

- **comparador** entity
- **comparador** architecture

7.4 comparador.vhd

```

00001
00002 library IEEE;
00003 use IEEE.STD_LOGIC_UNSIGNED.all;
00004 use IEEE.STD_LOGIC_1164.all;
00005 use ieee.fixed_pkg.all;           --arquivo deve ser adicionado ao projeto
00006 use ieee.numeric_std.all;
00007
00008 entity comparador is
00009 generic (
00010     constant n_bits_c: integer := 16 --numero de bits da portadora
00011 );
00012 port(
00013     clk : in std_logic; -- clock
00014     en : in std_logic; -- habilita modulo
00015     comp : in std_logic_vector(n_bits_c-1 downto 0); -- moduladora
00016     c : in std_logic_vector(n_bits_c-1 downto 0); -- portadora
00017     amost : in std_logic; -- amostra moduladora na borda de amost
00018     comp_out : out std_logic
00019 );
00020
00021 end entity comparador;
00022
00023
00024 architecture comparador of comparador is
00025
00026
00027     signal comp_int : std_logic_vector(n_bits_c-1 downto 0); -- data out
00028
00029
00030
00031 begin
00032     process(clk)
00033     begin
00034         if en = '0' then
00035             comp_out <= '0';
00036         elsif falling_edge(clk) then
00037             if (comp_int > c) then
00038                 comp_out <= '0';
00039             else
00040                 comp_out <= '1';
00041             end if;
00042         end if;
00043     end process;
00044
00045     process(amost)
00046     begin
00047         if rising_edge(amost) then
00048             comp_int <= comp;
00049         end if;
00050     end process;
00051
00052 end architecture comparador;

```

7.5 comparadores.vhd File Reference

Entities

- **comparadores** entity
- **comparadores** architecture

7.6 comparadores.vhd

```

00001 library IEEE;
00002 use IEEE.STD_LOGIC_UNSIGNED.all;

```

```

00003 use IEEE.STD_LOGIC_1164.all;           --arquivo deve ser adicionado ao projeto
00004 use ieee.fixed_pkg.all;
00005 use ieee.numeric_std.all;
00006 use work.my_types_pkg.all;
00007
00008
00009 --comunicacao com submodulos
00010 --
00011 -- conexao hardware:
00012 --   out -> 24 sinais de TX UART (8 bits)
00013 --   in  -> 24 sinais de RX UART (9 bits)
00014 --
00015 -- mode:
00016 --   RUN  -> transmite sinais PWM nos sinais TX e envia dados de medicao de tensao dos modulos para a memoria compartilhado com DSP
00017 --   STOP ->
00018
00019 entity comparadores is
00020 generic  (constant MAX : integer := 3526000;
00021             constant n_bits_c: integer := 22  --numero de bits da portadora
00022         );
00023     port(
00024         clk : in std_logic; -- clock
00025         en : in std_logic; -- habilita modulo
00026         comp : in COMP_ARRAY; -- moduladora
00027         c : in COMP_ARRAY; -- portadora
00028         amost : in std_logic_vector(24 downto 1); -- amostra moduladora na borda se subida de c_sinc
00029         comp_out : out std_logic_vector(24 downto 1)
00030     );
00031 end entity comparadores;
00032
00033
00034 architecture comparadores of comparadores is
00035
00036     component comparador
00037         port(
00038             clk : in std_logic; -- clock
00039             en : in std_logic; -- habilita modulo
00040             comp : in std_logic_vector(n_bits_c-1 downto 0); -- moduladora
00041             c : in std_logic_vector(n_bits_c-1 downto 0); -- portadora
00042             amost : in std_logic; -- amostra moduladora na borda se subida de c_sinc
00043             comp_out : out std_logic
00044         );
00045     end component;
00046
00047
00048
00049
00050     begin
00051
00052     t: for i in 1 to 24 generate
00053         u: comparador port map(
00054             clk => clk,
00055             en => en,
00056             comp => comp(i),
00057             amost => amost(i),
00058             c => c(i),
00059             comp_out => comp_out(i)
00060         );
00061     end generate t;
00062
00063
00064 end architecture comparadores;

```

7.7 contador.vhd File Reference

Entities

- **contador** entity
- **contador** architecture

7.8 contador.vhd

```

00001 library IEEE;
00002 use IEEE.STD_LOGIC_UNSIGNED.all;
00003 use IEEE.STD_LOGIC_1164.all;

```

```

00004 use ieee.numeric_std.all;
00005
00006 entity contador is
00007 generic (
00008     constant n_bits : integer :=16 --numero de bits do contador
00009 );
00010 port(
00011     clk : in std_logic; -- clock
00012     en : in std_logic; -- habilita modulo
00013     reset : in std_logic;
00014     sinc : out std_logic;
00015     count_max : in std_logic_vector(n_bits-1 downto 0); -- estouro contagem
00016     count_ini : in std_logic_vector(n_bits-1 downto 0); -- valor carregado ao
00017     contador no reset
00018     count_comp : in std_logic_vector(n_bits-1 downto 0); -- valor comparacao
00019     count : out std_logic_vector(n_bits-1 downto 0); -- valor contagem
00020     comp : out std_logic -- resultado comparacao
00021 );
00022 end entity contador;
00023
00024
00025 architecture contador of contador is
00026
00027     signal count_int : std_logic_vector(n_bits-1 downto 0);
00028
00029 begin
00030
00031     process(clk)
00032     begin
00033         if en = '1' then
00034             if reset = '1' then
00035                 count_int <= count_ini;
00036             elsif rising_edge(clk) then
00037                 if count_int < count_max then
00038                     count_int <= count_int+1;
00039                     sinc <= '0';
00040                 else
00041                     sinc <= '1';
00042                     count_int <= (others=> '0');
00043                 end if;
00044                 if count_int > count_comp then
00045                     comp <= '1';
00046                 else
00047                     comp <= '0';
00048                 end if;
00049             end if;
00050         end if;
00051     end process;
00052
00053 count <= count_int;
00054
00055 end architecture contador;

```

7.9 DE0_NANO_VF.qsf File Reference

Constraints

- FAMILY "CycloneIV-E"
- DEVICE EP4CE22F17C6
- TOP_LEVEL_ENTITY DE0_NANO_VF
- ORIGINAL_QUARTUS_VERSION 14.1.0
- PROJECT_CREATION_TIME_DATE "16:03:05 JANUARY 19, 2015"
- LAST_QUARTUS_VERSION 14.1.0
- PROJECT_OUTPUT_DIRECTORY output_files
- MIN_CORE_JUNCTION_TEMP 0
- MAX_CORE_JUNCTION_TEMP 85
- ERROR_CHECK_FREQUENCY_DIVISOR 1
- NOMINAL_CORE_SUPPLY_VOLTAGE 1.2V
- EDA_SIMULATION_TOOL "ModelSim-Altera(VHDL)"
- EDA_OUTPUT_DATA_FORMAT VHDL-section_ideda_simulation
- STRATIX_DEVICE_IO_STANDARD "2.5 V"
- PIN_R8-toCLOCK_50

- PIN_A15 -toLED[0]
- PIN_A13 -toLED[1]
- PIN_B13 -toLED[2]
- PIN_A11 -toLED[3]
- PIN_D1 -toLED[4]
- PIN_F3 -toLED[5]
- PIN_B1 -toLED[6]
- PIN_L3 -toLED[7]
- PIN_J15 -toKEY[0]
- PIN_E1 -toKEY[1]
- PIN_M1 -toSW[0]
- PIN_T8 -toSW[1]
- PIN_B9 -toSW[2]
- PIN_M15 -toSW[3]
- PIN_D3 -toGPIO_0[0]
- PIN_C3 -toGPIO_0[1]
- PIN_A2 -toGPIO_0[2]
- PIN_A3 -toGPIO_0[3]
- PIN_B3 -toPWM1H_FA[1]
- PIN_B4 -toPWM1L_FA[1]
- PIN_A4 -toPWM2H_FA[1]
- PIN_B5 -toPWM2L_FA[1]
- PIN_A5 -toINT0_FA[1]
- PIN_D5 -toRESET_FA[1]
- PIN_B6 -toPWM1H_FA[2]
- PIN_A6 -toPWM1L_FA[2]
- PIN_B7 -toPWM2H_FA[2]
- PIN_D6 -toPWM2L_FA[2]
- PIN_A7 -toINT0_FA[2]
- PIN_C6 -toRESET_FA[2]
- PIN_C8 -toPWM1H_FB[0]
- PIN_E6 -toPWM1L_FB[0]
- PIN_E7 -toPWM2H_FB[0]
- PIN_D8 -toPWM2L_FB[0]
- PIN_E8 -toINT0_FB[0]
- PIN_F8 -toRESET_FB[0]
- PIN_F9 -toPWM1H_FB[1]
- PIN_E9 -toPWM1L_FB[1]
- PIN_C9 -toPWM2H_FB[1]
- PIN_D9 -toPWM2L_FB[1]
- PIN_E11 -toINT0_FB[1]
- PIN_E10 -toRESET_FB[1]
- PIN_C11 -toPWM1H_FB[2]
- PIN_B11 -toPWM1L_FB[2]
- PIN_A12 -toPWM2H_FB[2]
- PIN_D11 -toPWM2L_FB[2]
- PIN_D12 -toINT0_FB[2]
- PIN_B12 -toRESET_FB[2]
- PIN_R13 -toPWM1H_FC[1]
- PIN_T12 -toPWM1L_FC[1]
- PIN_R12 -toPWM2H_FC[1]
- PIN_T11 -toPWM2L_FC[1]
- PIN_T10 -toINT0_FC[1]
- PIN_R11 -toRESET_FC[1]
- PIN_P11 -toPWM1H_FC[2]

- `PIN_R10 -toPWM1L_FC[2]`
- `PIN_N12 -toPWM2H_FC[2]`
- `PIN_P9 -toPWM2L_FC[2]`
- `PIN_N9 -toINT0_FC[2]`
- `PIN_N11 -toRESET_FC[2]`
- `PIN_L16 -toPWM1H_FC[0]`
- `PIN_K16 -toPWM1L_FC[0]`
- `PIN_R16 -toPWM2H_FC[0]`
- `PIN_L15 -toPWM2L_FC[0]`
- `PIN_P15 -toINT0_FC[0]`
- `PIN_P16 -toRESET_FC[0]`
- `PIN_N15 -toPWM1H_FA[0]`
- `PIN_P14 -toPWM1L_FA[0]`
- `PIN_L14 -toPWM2H_FA[0]`
- `PIN_N14 -toPWM2L_FA[0]`
- `PIN_M10 -toINT0_FA[0]`
- `PIN_L13 -toRESET_FA[0]`
- `POWER_PRESET_COOLING_SOLUTION " 23 MMHEATSINKWITH 200 LFPMAIRFLOW"`
- `POWER_BOARD_THERMAL_MODEL "NONE(CONSERVATIVE)"`
- `PARTITION_NETLIST_TYPE SOURCE-section_idTop`
- `PARTITION_FITTER_PRESERVATION_LEVEL PLACEMENT_AND_ROUTING-section_idTop`
- `PARTITION_COLOR 16764057 -section_idTop`
- `QIP_FILE pll.qip`
- `VHDL_FILE my_types_pkg.vhd`
- `VHDL_FILE fixed_pkg_c.vhdl`
- `VHDL_FILE fixed_float_types_c.vhdl`
- `CDF_FILE output_files/Chain1.cdf`
- `VHDL_FILE contador.vhd`
- `VHDL_FILE tabela_sin.vhd`
- `VHDL_FILE theta_abc.vhd`
- `VHDL_FILE portadora_tringular.vhd`
- `VHDL_FILE clk_div.vhd`
- `VHDL_FILE LEDs.vhd`
- `VHDL_FILE integrador.vhd`
- `CDF_FILE output_files/Chain3.cdf`
- `CDF_FILE output_files/Chain2.cdf`
- `CDF_FILE output_files/Chain4.cdf`
- `CDF_FILE Chain1.cdf`
- `VHDL_FILE fbpspwmdt.vhd`
- `VHDL_FILE wt.vhd`
- `CDF_FILE output_files/Chain5.cdf`
- `CDF_FILE output_files/Chain6.cdf`
- `VHDL_FILE DE0_NANO_VF.vhd`
- `VHDL_FILE vfcontrol.vhd`

7.9.1 Variable Documentation

7.9.1.1 CDF_FILE output_files/Chain3.cdf [Constraints]

Definition at line 180 of file `DE0_NANO_VF.qsf`.

7.9.1.2 CDF_FILE output_files/Chain2.cdf [Constraints]

Definition at line 181 of file `DE0_NANO_VF.qsf`.

7.9.1.3 CDF_FILE output_files/Chain4.cdf [Constraints]

Definition at line 182 of file [DE0_NANO_VF.qsf](#).

7.9.1.4 CDF_FILE Chain1.cdf [Constraints]

Definition at line 183 of file [DE0_NANO_VF.qsf](#).

7.9.1.5 CDF_FILE output_files/Chain5.cdf [Constraints]

Definition at line 186 of file [DE0_NANO_VF.qsf](#).

7.9.1.6 CDF_FILE output_files/Chain6.cdf [Constraints]

Definition at line 187 of file [DE0_NANO_VF.qsf](#).

7.9.1.7 CDF_FILE output_files/Chain1.cdf [Constraints]

Definition at line 172 of file [DE0_NANO_VF.qsf](#).

7.9.1.8 DEVICE EP4CE22F17C6 [Constraints]

Definition at line 41 of file [DE0_NANO_VF.qsf](#).

7.9.1.9 EDA_OUTPUT_DATA_FORMAT VHDL-section_ideda_simulation [Constraints]

Definition at line 52 of file [DE0_NANO_VF.qsf](#).

7.9.1.10 EDA_SIMULATION_TOOL "ModelSim-Altera(VHDL)" [Constraints]

Definition at line 51 of file [DE0_NANO_VF.qsf](#).

7.9.1.11 ERROR_CHECK_FREQUENCY_DIVISOR 1 [Constraints]

Definition at line 49 of file [DE0_NANO_VF.qsf](#).

7.9.1.12 FAMILY "CycloneIVE" [Constraints]

Definition at line 40 of file [DE0_NANO_VF.qsf](#).

7.9.1.13 LAST_QUARTUS_VERSION 14.1.0 [Constraints]

Definition at line 45 of file [DE0_NANO_VF.qsf](#).

7.9.1.14 MAX_CORE_JUNCTION_TEMP 85 [Constraints]

Definition at line 48 of file [DE0_NANO_VF.qsf](#).

7.9.1.15 **MIN_CORE_JUNCTION_TEMP 0** [Constraints]

Definition at line 47 of file [DE0_NANO_VF.qsf](#).

7.9.1.16 **NOMINAL_CORE_SUPPLY_VOLTAGE 1.2V** [Constraints]

Definition at line 50 of file [DE0_NANO_VF.qsf](#).

7.9.1.17 **ORIGINAL_QUARTUS_VERSION 14.1.0** [Constraints]

Definition at line 43 of file [DE0_NANO_VF.qsf](#).

7.9.1.18 **PARTITION_COLOR 16764057 -section_idTop** [Constraints]

Definition at line 167 of file [DE0_NANO_VF.qsf](#).

7.9.1.19 **PARTITION_FITTER_PRESERVATION_LEVEL PLACEMENT_AND_ROUTING-section_idTop**
[Constraints]

Definition at line 166 of file [DE0_NANO_VF.qsf](#).

7.9.1.20 **PARTITION_NETLIST_TYPE SOURCE-section_idTop** [Constraints]

Definition at line 165 of file [DE0_NANO_VF.qsf](#).

7.9.1.21 **PIN_A11 -toLED[3]** [Constraints]

Definition at line 58 of file [DE0_NANO_VF.qsf](#).

7.9.1.22 **PIN_A12 -toPWM2H_FB[2]** [Constraints]

Definition at line 125 of file [DE0_NANO_VF.qsf](#).

7.9.1.23 **PIN_A13 -toLED[1]** [Constraints]

Definition at line 56 of file [DE0_NANO_VF.qsf](#).

7.9.1.24 **PIN_A15 -toLED[0]** [Constraints]

Definition at line 55 of file [DE0_NANO_VF.qsf](#).

7.9.1.25 **PIN_A2 -toGPIO_0[2]** [Constraints]

Definition at line 74 of file [DE0_NANO_VF.qsf](#).

7.9.1.26 **PIN_A3 -toGPIO_0[3]** [Constraints]

Definition at line 75 of file [DE0_NANO_VF.qsf](#).

7.9.1.27 PIN_A4 -toPWM2H_FA[1] [Constraints]

Definition at line 93 of file [DE0_NANO_VF.qsf](#).

7.9.1.28 PIN_A5 -toINT0_FA[1] [Constraints]

Definition at line 95 of file [DE0_NANO_VF.qsf](#).

7.9.1.29 PIN_A6 -toPWM1L_FA[2] [Constraints]

Definition at line 100 of file [DE0_NANO_VF.qsf](#).

7.9.1.30 PIN_A7 -toINT0_FA[2] [Constraints]

Definition at line 103 of file [DE0_NANO_VF.qsf](#).

7.9.1.31 PIN_B11 -toPWM1L_FB[2] [Constraints]

Definition at line 124 of file [DE0_NANO_VF.qsf](#).

7.9.1.32 PIN_B12 -toRESET_FB[2] [Constraints]

Definition at line 128 of file [DE0_NANO_VF.qsf](#).

7.9.1.33 PIN_B13 -toLED[2] [Constraints]

Definition at line 57 of file [DE0_NANO_VF.qsf](#).

7.9.1.34 PIN_B1 -toLED[6] [Constraints]

Definition at line 61 of file [DE0_NANO_VF.qsf](#).

7.9.1.35 PIN_B3 -toPWM1H_FA[1] [Constraints]

Definition at line 91 of file [DE0_NANO_VF.qsf](#).

7.9.1.36 PIN_B4 -toPWM1L_FA[1] [Constraints]

Definition at line 92 of file [DE0_NANO_VF.qsf](#).

7.9.1.37 PIN_B5 -toPWM2L_FA[1] [Constraints]

Definition at line 94 of file [DE0_NANO_VF.qsf](#).

7.9.1.38 PIN_B6 -toPWM1H_FA[2] [Constraints]

Definition at line 99 of file [DE0_NANO_VF.qsf](#).

7.9.1.39 PIN_B7 -toPWM2H_FA[2] [Constraints]

Definition at line 101 of file [DE0_NANO_VF.qsf](#).

7.9.1.40 PIN_B9 -toSW[2] [Constraints]

Definition at line 67 of file [DE0_NANO_VF.qsf](#).

7.9.1.41 PIN_C11 -toPWM1H_FB[2] [Constraints]

Definition at line 123 of file [DE0_NANO_VF.qsf](#).

7.9.1.42 PIN_C3 -toGPIO_0[1] [Constraints]

Definition at line 73 of file [DE0_NANO_VF.qsf](#).

7.9.1.43 PIN_C6 -toRESET_FA[2] [Constraints]

Definition at line 104 of file [DE0_NANO_VF.qsf](#).

7.9.1.44 PIN_C8 -toPWM1H_FB[0] [Constraints]

Definition at line 107 of file [DE0_NANO_VF.qsf](#).

7.9.1.45 PIN_C9 -toPWM2H_FB[1] [Constraints]

Definition at line 117 of file [DE0_NANO_VF.qsf](#).

7.9.1.46 PIN_D11 -toPWM2L_FB[2] [Constraints]

Definition at line 126 of file [DE0_NANO_VF.qsf](#).

7.9.1.47 PIN_D12 -toINT0_FB[2] [Constraints]

Definition at line 127 of file [DE0_NANO_VF.qsf](#).

7.9.1.48 PIN_D1 -toLED[4] [Constraints]

Definition at line 59 of file [DE0_NANO_VF.qsf](#).

7.9.1.49 PIN_D3 -toGPIO_0[0] [Constraints]

Definition at line 72 of file [DE0_NANO_VF.qsf](#).

7.9.1.50 PIN_D5 -toRESET_FA[1] [Constraints]

Definition at line 96 of file [DE0_NANO_VF.qsf](#).

7.9.1.51 PIN_D6 -toPWM2L_FA[2] [Constraints]

Definition at line 102 of file [DE0_NANO_VF.qsf](#).

7.9.1.52 PIN_D8 -toPWM2L_FB[0] [Constraints]

Definition at line 110 of file [DE0_NANO_VF.qsf](#).

7.9.1.53 PIN_D9 -toPWM2L_FB[1] [Constraints]

Definition at line 118 of file [DE0_NANO_VF.qsf](#).

7.9.1.54 PIN_E10 -toRESET_FB[1] [Constraints]

Definition at line 120 of file [DE0_NANO_VF.qsf](#).

7.9.1.55 PIN_E11 -toINT0_FB[1] [Constraints]

Definition at line 119 of file [DE0_NANO_VF.qsf](#).

7.9.1.56 PIN_E1 -toKEY[1] [Constraints]

Definition at line 64 of file [DE0_NANO_VF.qsf](#).

7.9.1.57 PIN_E6 -toPWM1L_FB[0] [Constraints]

Definition at line 108 of file [DE0_NANO_VF.qsf](#).

7.9.1.58 PIN_E7 -toPWM2H_FB[0] [Constraints]

Definition at line 109 of file [DE0_NANO_VF.qsf](#).

7.9.1.59 PIN_E8 -toINT0_FB[0] [Constraints]

Definition at line 111 of file [DE0_NANO_VF.qsf](#).

7.9.1.60 PIN_E9 -toPWM1L_FB[1] [Constraints]

Definition at line 116 of file [DE0_NANO_VF.qsf](#).

7.9.1.61 PIN_F3 -toLED[5] [Constraints]

Definition at line 60 of file [DE0_NANO_VF.qsf](#).

7.9.1.62 PIN_F8 -toRESET_FB[0] [Constraints]

Definition at line 112 of file [DE0_NANO_VF.qsf](#).

7.9.1.63 PIN_F9 -toPWM1H_FC[1] [Constraints]

Definition at line 115 of file [DE0_NANO_VF.qsf](#).

7.9.1.64 PIN_J15 -toKEY[0] [Constraints]

Definition at line 63 of file [DE0_NANO_VF.qsf](#).

7.9.1.65 PIN_K16 -toPWM1L_FC[0] [Constraints]

Definition at line 148 of file [DE0_NANO_VF.qsf](#).

7.9.1.66 PIN_L13 -toRESET_FA[0] [Constraints]

Definition at line 160 of file [DE0_NANO_VF.qsf](#).

7.9.1.67 PIN_L14 -toPWM2H_FA[0] [Constraints]

Definition at line 157 of file [DE0_NANO_VF.qsf](#).

7.9.1.68 PIN_L15 -toPWM2L_FC[0] [Constraints]

Definition at line 150 of file [DE0_NANO_VF.qsf](#).

7.9.1.69 PIN_L16 -toPWM1H_FC[0] [Constraints]

Definition at line 147 of file [DE0_NANO_VF.qsf](#).

7.9.1.70 PIN_L3 -toLED[7] [Constraints]

Definition at line 62 of file [DE0_NANO_VF.qsf](#).

7.9.1.71 PIN_M10 -toINT0_FA[0] [Constraints]

Definition at line 159 of file [DE0_NANO_VF.qsf](#).

7.9.1.72 PIN_M15 -toSW[3] [Constraints]

Definition at line 68 of file [DE0_NANO_VF.qsf](#).

7.9.1.73 PIN_M1 -toSW[0] [Constraints]

Definition at line 65 of file [DE0_NANO_VF.qsf](#).

7.9.1.74 PIN_N11 -toRESET_FC[2] [Constraints]

Definition at line 144 of file [DE0_NANO_VF.qsf](#).

7.9.1.75 PIN_N12 -toPWM2H_FC[2] [Constraints]

Definition at line 141 of file [DE0_NANO_VF.qsf](#).

7.9.1.76 PIN_N14 -toPWM2L_FA[0] [Constraints]

Definition at line 158 of file [DE0_NANO_VF.qsf](#).

7.9.1.77 PIN_N15 -toPWM1H_FA[0] [Constraints]

Definition at line 155 of file [DE0_NANO_VF.qsf](#).

7.9.1.78 PIN_N9 -toINT0_FC[2] [Constraints]

Definition at line 143 of file [DE0_NANO_VF.qsf](#).

7.9.1.79 PIN_P11 -toPWM1H_FC[2] [Constraints]

Definition at line 139 of file [DE0_NANO_VF.qsf](#).

7.9.1.80 PIN_P14 -toPWM1L_FA[0] [Constraints]

Definition at line 156 of file [DE0_NANO_VF.qsf](#).

7.9.1.81 PIN_P15 -toINT0_FC[0] [Constraints]

Definition at line 151 of file [DE0_NANO_VF.qsf](#).

7.9.1.82 PIN_P16 -toRESET_FC[0] [Constraints]

Definition at line 152 of file [DE0_NANO_VF.qsf](#).

7.9.1.83 PIN_P9 -toPWM2L_FC[2] [Constraints]

Definition at line 142 of file [DE0_NANO_VF.qsf](#).

7.9.1.84 PIN_R10 -toPWM1L_FC[2] [Constraints]

Definition at line 140 of file [DE0_NANO_VF.qsf](#).

7.9.1.85 PIN_R11 -toRESET_FC[1] [Constraints]

Definition at line 136 of file [DE0_NANO_VF.qsf](#).

7.9.1.86 PIN_R12 -toPWM2H_FC[1] [Constraints]

Definition at line 133 of file [DE0_NANO_VF.qsf](#).

7.9.1.87 PIN_R13 -toPWM1H_FC[1] [Constraints]

Definition at line 131 of file [DE0_NANO_VF.qsf](#).

7.9.1.88 PIN_R16 -toPWM2H_FC[0] [Constraints]

Definition at line 149 of file [DE0_NANO_VF.qsf](#).

7.9.1.89 PIN_R8 -toCLOCK_50 [Constraints]

Definition at line 54 of file [DE0_NANO_VF.qsf](#).

7.9.1.90 PIN_T10 -toINT0_FC[1] [Constraints]

Definition at line 135 of file [DE0_NANO_VF.qsf](#).

7.9.1.91 PIN_T11 -toPWM2L_FC[1] [Constraints]

Definition at line 134 of file [DE0_NANO_VF.qsf](#).

7.9.1.92 PIN_T12 -toPWM1L_FC[1] [Constraints]

Definition at line 132 of file [DE0_NANO_VF.qsf](#).

7.9.1.93 PIN_T8 -toSW[1] [Constraints]

Definition at line 66 of file [DE0_NANO_VF.qsf](#).

7.9.1.94 POWER_BOARD_THERMAL_MODEL "NONE(CONSERVATIVE)" [Constraints]

Definition at line 164 of file [DE0_NANO_VF.qsf](#).

7.9.1.95 POWER_PRESET_COOLING_SOLUTION "23 MMHEATSINKWITH 200 LFPMAIRFLOW" [Constraints]

Definition at line 163 of file [DE0_NANO_VF.qsf](#).

7.9.1.96 PROJECT_CREATION_TIME_DATE "16 : 03 : 05 JANUARY 19 , 2015" [Constraints]

Definition at line 44 of file [DE0_NANO_VF.qsf](#).

7.9.1.97 PROJECT_OUTPUT_DIRECTORY output_files [Constraints]

Definition at line 46 of file [DE0_NANO_VF.qsf](#).

7.9.1.98 QIP_FILE pll.qip [Constraints]

Definition at line 168 of file [DE0_NANO_VF.qsf](#).

7.9.1.99 **STRATIX_DEVICE_IO_STANDARD "2.5 V"** [Constraints]

Definition at line 53 of file [DE0_NANO_VF.qsf](#).

7.9.1.100 **TOP_LEVEL_ENTITY DE0_NANO_VF** [Constraints]

Definition at line 42 of file [DE0_NANO_VF.qsf](#).

7.9.1.101 **VHDL_FILE portadora_tringular.vhd** [Constraints]

Definition at line 176 of file [DE0_NANO_VF.qsf](#).

7.9.1.102 **VHDL_FILE clk_div.vhd** [Constraints]

Definition at line 177 of file [DE0_NANO_VF.qsf](#).

7.9.1.103 **VHDL_FILE LEDs.vhd** [Constraints]

Definition at line 178 of file [DE0_NANO_VF.qsf](#).

7.9.1.104 **VHDL_FILE integrador.vhd** [Constraints]

Definition at line 179 of file [DE0_NANO_VF.qsf](#).

7.9.1.105 **VHDL_FILE fbpspwm.vhd** [Constraints]

Definition at line 184 of file [DE0_NANO_VF.qsf](#).

7.9.1.106 **VHDL_FILE wt.vhd** [Constraints]

Definition at line 185 of file [DE0_NANO_VF.qsf](#).

7.9.1.107 **VHDL_FILE DE0_NANO_VF.vhd** [Constraints]

Definition at line 188 of file [DE0_NANO_VF.qsf](#).

7.9.1.108 **VHDL_FILE vfcontrol.vhd** [Constraints]

Definition at line 189 of file [DE0_NANO_VF.qsf](#).

7.9.1.109 **VHDLFILE my_types_pkg.vhd** [Constraints]

Definition at line 169 of file [DE0_NANO_VF.qsf](#).

7.9.1.110 **VHDLFILE fixed_pkg_c.vhdl** [Constraints]

Definition at line 170 of file [DE0_NANO_VF.qsf](#).

7.9.1.111 VHDL_FILE fixed_float_types_c.vhd [Constraints]

Definition at line 171 of file [DE0_NANO_VF.qsf](#).

7.9.1.112 VHDL_FILE contador.vhd [Constraints]

Definition at line 173 of file [DE0_NANO_VF.qsf](#).

7.9.1.113 VHDL_FILE tabela_sin.vhd [Constraints]

Definition at line 174 of file [DE0_NANO_VF.qsf](#).

7.9.1.114 VHDLFILE theta_abc.vhd [Constraints]

Definition at line 175 of file [DE0_NANO_VF.qsf](#).

7.10 DE0_NANO_VF.qsf

```

00001 # -----
00002 #
00003 # Copyright (C) 1991-2014 Altera Corporation. All rights reserved.
00004 # Your use of Altera Corporation's design tools, logic functions
00005 # and other software and tools, and its AMPP partner logic
00006 # functions, and any output files from any of the foregoing
00007 # (including device programming or simulation files), and any
00008 # associated documentation or information are expressly subject
00009 # to the terms and conditions of the Altera Program License
00010 # Subscription Agreement, the Altera Quartus II License Agreement,
00011 # the Altera MegaCore Function License Agreement, or other
00012 # applicable license agreement, including, without limitation,
00013 # that your use is for the sole purpose of programming logic
00014 # devices manufactured by Altera and sold by Altera or its
00015 # authorized distributors. Please refer to the applicable
00016 # agreement for further details.
00017 #
00018 # -----
00019 #
00020 # Quartus II 64-Bit
00021 # Version 14.1.0 Build 186 12/03/2014 SJ Web Edition
00022 # Date created = 16:03:05 January 19, 2015
00023 #
00024 # -----
00025 #
00026 # Notes:
00027 #
00028 # 1) The default values for assignments are stored in the file:
00029 #     DE0_NANO_base_assignment_defaults.qdf
00030 #     If this file doesn't exist, see file:
00031 #         assignment_defaults.qdf
00032 #
00033 # 2) Altera recommends that you do not modify this file. This
00034 #     file is updated automatically by the Quartus II software
00035 #     and any changes you make may be lost or overwritten.
00036 #
00037 # -----
00038
00039
00040 set_global_assignment -name FAMILY "Cyclone IV E"
00041 set_global_assignment -name DEVICE EP4CE22F17C6
00042 set_global_assignment -name TOP_LEVEL_ENTITY DE0_NANO_VF
00043 set_global_assignment -name ORIGINAL_QUARTUS_VERSION 14.1.0
00044 set_global_assignment -name PROJECT_CREATION_TIME_DATE "16:03:05 JANUARY 19, 2015"
00045 set_global_assignment -name LAST_QUARTUS_VERSION 14.1.0
00046 set_global_assignment -name PROJECT_OUTPUT_DIRECTORY output_files
00047 set_global_assignment -name MIN_CORE_JUNCTION_TEMP 0
00048 set_global_assignment -name MAX_CORE_JUNCTION_TEMP 85
00049 set_global_assignment -name ERROR_CHECK_FREQUENCY_DIVISOR 1
00050 set_global_assignment -name NOMINAL_CORE_SUPPLY_VOLTAGE 1.2V
00051 set_global_assignment -name EDA_SIMULATION_TOOL "ModelSim-Altera (VHDL)"
00052 set_global_assignment -name EDA_OUTPUT_DATA_FORMAT VHDL -section_id eda_simulation
00053 set_global_assignment -name STRATIX_DEVICE_IO_STANDARD "2.5 V"
00054 set_location_assignment PIN_R8 -to CLOCK_50

```

```

00055 set_location_assignment PIN_A15 -to LED[0]
00056 set_location_assignment PIN_A13 -to LED[1]
00057 set_location_assignment PIN_B13 -to LED[2]
00058 set_location_assignment PIN_A11 -to LED[3]
00059 set_location_assignment PIN_D1 -to LED[4]
00060 set_location_assignment PIN_F3 -to LED[5]
00061 set_location_assignment PIN_B1 -to LED[6]
00062 set_location_assignment PIN_L3 -to LED[7]
00063 set_location_assignment PIN_J15 -to KEY[0]
00064 set_location_assignment PIN_E1 -to KEY[1]
00065 set_location_assignment PIN_M1 -to SW[0]
00066 set_location_assignment PIN_T8 -to SW[1]
00067 set_location_assignment PIN_B9 -to SW[2]
00068 set_location_assignment PIN_M15 -to SW[3]
00069
00070 # set_location_assignment PIN_A8 -to GPIO_0_IN[0]
00071 # set_location_assignment PIN_B8 -to GPIO_0_IN[1]
00072 set_location_assignment PIN_D3 -to GPIO_0[0]
00073 set_location_assignment PIN_C3 -to GPIO_0[1]
00074 set_location_assignment PIN_A2 -to GPIO_0[2]
00075 set_location_assignment PIN_A3 -to GPIO_0[3]
00076
00077 # set_location_assignment PIN_T9 -to GPIO_1_IN[0]
00078 # set_location_assignment PIN_R9 -to GPIO_1_IN[1]
00079 # set_location_assignment PIN_F13 -to GPIO_1[0]
00080 # set_location_assignment PIN_T15 -to GPIO_1[1]
00081 # set_location_assignment PIN_T14 -to GPIO_1[2]
00082 # set_location_assignment PIN_T13 -to GPIO_1[3]
00083 # set_location_assignment PIN_R14 -to GPIO_1[4]
00084 # set_location_assignment PIN_N16 -to GPIO_1[5]
00085 # set_location_assignment PIN_J16 -to GPIO_1[6]
00086 # set_location_assignment PIN_K15 -to GPIO_1[7]
00087 # set_location_assignment PIN_J13 -to GPIO_1[8]
00088 # set_location_assignment PIN_J14 -to GPIO_1[9]
00089
00090 # PHASE A UNIT 2
00091 set_location_assignment PIN_B3 -to PWM1H_FA[1]
00092 set_location_assignment PIN_B4 -to PWM1L_FA[1]
00093 set_location_assignment PIN_A4 -to PWM2H_FA[1]
00094 set_location_assignment PIN_B5 -to PWM2L_FA[1]
00095 set_location_assignment PIN_A5 -to INT0_FA[1]
00096 set_location_assignment PIN_D5 -to RESET_FA[1]
00097
00098 # PHASE A UNIT 3
00099 set_location_assignment PIN_B6 -to PWM1H_FA[2]
00100 set_location_assignment PIN_A6 -to PWM1L_FA[2]
00101 set_location_assignment PIN_B7 -to PWM2H_FA[2]
00102 set_location_assignment PIN_D6 -to PWM2L_FA[2]
00103 set_location_assignment PIN_A7 -to INT0_FA[2]
00104 set_location_assignment PIN_C6 -to RESET_FA[2]
00105
00106 # PHASE B UNIT 1
00107 set_location_assignment PIN_C8 -to PWM1H_FB[0]
00108 set_location_assignment PIN_E6 -to PWM1L_FB[0]
00109 set_location_assignment PIN_E7 -to PWM2H_FB[0]
00110 set_location_assignment PIN_D8 -to PWM2L_FB[0]
00111 set_location_assignment PIN_E8 -to INT0_FB[0]
00112 set_location_assignment PIN_F8 -to RESET_FB[0]
00113
00114 # PHASE B UNIT 2
00115 set_location_assignment PIN_F9 -to PWM1H_FB[1]
00116 set_location_assignment PIN_E9 -to PWM1L_FB[1]
00117 set_location_assignment PIN_C9 -to PWM2H_FB[1]
00118 set_location_assignment PIN_D9 -to PWM2L_FB[1]
00119 set_location_assignment PIN_E11 -to INT0_FB[1]
00120 set_location_assignment PIN_E10 -to RESET_FB[1]
00121
00122 # PHASE B UNIT 3
00123 set_location_assignment PIN_C11 -to PWM1H_FB[2]
00124 set_location_assignment PIN_B11 -to PWM1L_FB[2]
00125 set_location_assignment PIN_A12 -to PWM2H_FB[2]
00126 set_location_assignment PIN_D11 -to PWM2L_FB[2]
00127 set_location_assignment PIN_D12 -to INT0_FB[2]
00128 set_location_assignment PIN_B12 -to RESET_FB[2]
00129
00130 # PHASE C UNIT 2
00131 set_location_assignment PIN_R13 -to PWM1H_FC[1]
00132 set_location_assignment PIN_T12 -to PWM1L_FC[1]
00133 set_location_assignment PIN_R12 -to PWM2H_FC[1]
00134 set_location_assignment PIN_T11 -to PWM2L_FC[1]
00135 set_location_assignment PIN_T10 -to INT0_FC[1]
00136 set_location_assignment PIN_R11 -to RESET_FC[1]
00137
00138 # PHASE C UNIT 3
00139 set_location_assignment PIN_P11 -to PWM1H_FC[2]
00140 set_location_assignment PIN_R10 -to PWM1L_FC[2]
00141 set_location_assignment PIN_N12 -to PWM2H_FC[2]

```

```

00142 set_location_assignment PIN_P9 -to PWM2L_FC[2]
00143 set_location_assignment PIN_N9 -to INT0_FC[2]
00144 set_location_assignment PIN_N11 -to RESET_FC[2]
00145
00146 # PHASE C UNIT 1
00147 set_location_assignment PIN_L16 -to PWM1H_FC[0]
00148 set_location_assignment PIN_K16 -to PWM1L_FC[0]
00149 set_location_assignment PIN_R16 -to PWM2H_FC[0]
00150 set_location_assignment PIN_L15 -to PWM2L_FC[0]
00151 set_location_assignment PIN_P15 -to INT0_FC[0]
00152 set_location_assignment PIN_P16 -to RESET_FC[0]
00153
00154 # PHASE A UNIT 1
00155 set_location_assignment PIN_N15 -to PWM1H_FA[0]
00156 set_location_assignment PIN_P14 -to PWM1L_FA[0]
00157 set_location_assignment PIN_L14 -to PWM2H_FA[0]
00158 set_location_assignment PIN_N14 -to PWM2L_FA[0]
00159 set_location_assignment PIN_M10 -to INT0_FA[0]
00160 set_location_assignment PIN_L13 -to RESET_FA[0]
00161
00162
00163 set_global_assignment -name POWER_PRESET_COOLING_SOLUTION "23 MM HEAT SINK WITH 200 LFPM AIRFLOW"
00164 set_global_assignment -name POWER_BOARD_THERMAL_MODEL "NONE (CONSERVATIVE)"
00165 set_global_assignment -name PARTITION_NETLIST_TYPE SOURCE -section_id Top
00166 set_global_assignment -name PARTITION_FITTER_PRESERVATION_LEVEL PLACEMENT_AND_ROUTING -section_id Top
00167 set_global_assignment -name PARTITION_COLOR 16764057 -section_id Top
00168 set_global_assignment -name QIP_FILE pll.qip
00169 set_global_assignment -name VHDL_FILE my_types_pkg.vhd
00170 set_global_assignment -name VHDL_FILE fixed_pkg_c.vhdl
00171 set_global_assignment -name VHDL_FILE fixed_float_types_c.vhdl
00172 set_global_assignment -name CDF_FILE output_files/Chain1.cdf
00173 set_global_assignment -name VHDL_FILE contador.vhd
00174 set_global_assignment -name VHDL_FILE tabela_sin.vhd
00175 set_global_assignment -name VHDL_FILE theta_abc.vhd
00176 set_global_assignment -name VHDL_FILE portadora_tringular.vhd
00177 set_global_assignment -name VHDL_FILE clk_div.vhd
00178 set_global_assignment -name VHDL_FILE LEDs.vhd
00179 set_global_assignment -name VHDL_FILE integrador.vhd
00180 set_global_assignment -name CDF_FILE output_files/Chain3.cdf
00181 set_global_assignment -name CDF_FILE output_files/Chain2.cdf
00182 set_global_assignment -name CDF_FILE output_files/Chain4.cdf
00183 set_global_assignment -name CDF_FILE Chain1.cdf
00184 set_global_assignment -name VHDL_FILE fbp pwm dt.vhd
00185 set_global_assignment -name VHDL_FILE wt.vhd
00186 set_global_assignment -name CDF_FILE output_files/Chain5.cdf
00187 set_global_assignment -name CDF_FILE output_files/Chain6.cdf
00188 set_global_assignment -name VHDL_FILE DE0_NANO_VF.vhd
00189 set_global_assignment -name VHDL_FILE vfcontrol.vhd
00190 set_instance_assignment -name PARTITION_HIERARCHY root_partition -to | -section_id Top

```

7.11 DE0_NANO_VF.vhd File Reference

Entities

- **DE0_NANO_VF** entity
- **MAIN** architecture

7.12 DE0_NANO_VF.vhd

```

00001 -- Arquivo Base
00002
00003 LIBRARY ieee;
00004 USE ieee.std_logic_1164.all;
00005 USE ieee.std_logic_arith.all;
00006 USE ieee.std_logic_unsigned.all;
00007 use IEEE.NUMERIC_STD.all;
00008
00009 --arquivo deve ser adicionado ao projeto
00010 USE ieee.fixed_pkg.all;
00011 USE work.my_types_pkg.all;
00012
00013
00014
00015 ENTITY DE0_NANO_VF IS -- Base entity
00016   GENERIC(
00017     constant N : integer := 3; -- Number of inverters in the same phase
00018     constant Nin : integer := 13; -- numero de bits da parte inteira excluindo sinal de entrada

```

```

00019      constant Nout : integer := 30; --numero de bits da parte inteira excluindo sinal de
00020      sada
00021      constant n_bits_phase : integer := 30; --numero de bits que representa a fase
00022      da rede
00023      constant n_bits_c: integer := 16; --numero de bits da portadora
00024      constant TAM_MEM : integer := 32; -- tamanho da memoria (numero de palavras de 16
00025      bits)
00026      constant NBITS_MEM_ADDRESS : integer := 6; --numero de bits de endereço
00027      do banco de memoria (read do dsp)
00028      constant ID_MEM_DAC : integer := 28; --inicio do endereço de memoria destinado ao
00029      DAC conectado ao FPGA
00030      constant ID_MEM_SW1 : integer := 30; --inicio do endereço de memoria destinado ao
00031      DAC conectado ao FPGA
00032      constant I : integer := 1; --número de bits da parte inteira excluindo sinal
00033      constant F : integer := 14 --número de bits da parte fracionária
00034      );
00035  PORT(
00036    CLOCK_50 : in std_logic;
00037    LED : out std_logic_vector(7 DOWNTO 0);
00038    SW : in std_logic_vector(3 DOWNTO 0);
00039    KEY : in std_logic_vector(1 DOWNTO 0);
00040    -- GPIO_0
00041    -- GPIO_0_IN : in std_logic_vector(1 DOWNTO 0);
00042    -- Phase A
00043    RESET_FA : out std_logic_vector(N-1 DOWNTO 0);
00044    PWM1L_FA : out std_logic_vector(N-1 DOWNTO 0);
00045    PWM1H_FA : out std_logic_vector(N-1 DOWNTO 0);
00046    PWM2L_FA : out std_logic_vector(N-1 DOWNTO 0);
00047    PWM2H_FA : out std_logic_vector(N-1 DOWNTO 0);
00048    INT0_FA : in std_logic_vector(N-1 DOWNTO 0);
00049    -- Phase B
00050    RESET_FB : out std_logic_vector(N-1 DOWNTO 0);
00051    PWM1L_FB : out std_logic_vector(N-1 DOWNTO 0);
00052    PWM1H_FB : out std_logic_vector(N-1 DOWNTO 0);
00053    PWM2L_FB : out std_logic_vector(N-1 DOWNTO 0);
00054    PWM2H_FB : out std_logic_vector(N-1 DOWNTO 0);
00055    INT0_FB : in std_logic_vector(N-1 DOWNTO 0);
00056    -- Phase C
00057    RESET_FC : out std_logic_vector(N-1 DOWNTO 0);
00058    PWM1L_FC : out std_logic_vector(N-1 DOWNTO 0);
00059    PWM1H_FC : out std_logic_vector(N-1 DOWNTO 0);
00060    PWM2L_FC : out std_logic_vector(N-1 DOWNTO 0);
00061    PWM2H_FC : out std_logic_vector(N-1 DOWNTO 0);
00062    INT0_FC : in std_logic_vector(N-1 DOWNTO 0);
00063    -- GPIO_0 : out std_logic_vector(3 DOWNTO 0)
00064    -- GPIO_1
00065    -- GPIO_1_IN : in std_logic_vector(1 DOWNTO 0);
00066    -- GPIO_1 : out std_logic_vector(9 DOWNTO 0)
00067  );
00068 END DE0_NANO_VF;
00069
00070 );
00071
00072
00073
00074
00075
00076 -- Simples programa para piscar LED
00077 architecture MAIN of DE0_NANO_VF is
00078
00079
00080 COMPONENT LEDs  -- LEDs
00081   PORT(
00082     CLOCK_50 : in std_logic;
00083     LED : out std_logic_vector(7 DOWNTO 0)
00084   );
00085 END COMPONENT;
00086
00087
00088
00089 --
00090 component contador
00091   port(
00092     clk : in std_logic; -- clock
00093     en : in std_logic; -- habilita módulo
00094     reset : in std_logic;
00095     sinc : out std_logic;
00096     count_max : in std_logic_vector(15 downto 0); -- estouro contagem
00097     count_ini : in std_logic_vector(15 downto 0); -- valor carregado ao contador no reset
00098     count_comp : in std_logic_vector(15 downto 0); -- valor comparação
00099     count : out std_logic_vector(15 downto 0); -- valor contagem

```

```

00100      comp : out std_logic -- resultado comparacao
00101  );
00102 end component;
00103
00104 --
00105 component tabela_sin
00106   port(
00107     clk : in std_logic;
00108     theta: in std_logic_vector(15 downto 0);
00109     MAX : in sfixed(15 downto 0); -- valor de contagem maximo
00110     ma : in sfixed(I downto -F); -- Indice de modulação em Q15
00111     va : out std_logic_vector(15 downto 0)
00112   );
00113 end component;
00114
00115 --
00116 component pll
00117   port(
00118     areset      : IN STD_LOGIC  := '0';
00119     inclk0     : IN STD_LOGIC  := '0';
00120     c0         : OUT STD_LOGIC ;
00121     locked     : OUT STD_LOGIC
00122   );
00123 end component;
00124
00125
00126
00127 component integrador
00128   port(
00129     clk : in std_logic; -- clock
00130     en : in std_logic; -- habilita modulo tx
00131     reset : in std_logic; -- inicia transmissao (busy deve estar em '0')
00132     sinc : out std_logic;
00133     MAX : in std_logic_vector(Nout-1 downto 0);
00134     out_data : out std_logic_vector(Nout-1 downto 0); -- data out
00135     int_data : in std_logic_vector(Nin-1 downto 0) -- data in
00136   );
00137 end component;
00138
00139 --
00140 component theta_abc
00141   port(
00142     clk : in std_logic; -- clock
00143     en : in std_logic; -- habilita modulo
00144     reset : in std_logic; --
00145     theta_a : out std_logic_vector(15 downto 0); -- data out
00146     theta_b : out std_logic_vector(15 downto 0); -- data out
00147     theta_c : out std_logic_vector(15 downto 0); -- data out
00148     theta_in : in std_logic_vector(Nout-1 downto 0) -- data in
00149   );
00150 end component;
00151
00152 --
00153 component clk_div
00154   port(
00155     clk_in, en : in std_logic;
00156     div : in std_logic_vector(15 downto 0); -- f_out = f_in/(2*div)
00157     clk_out: out std_logic
00158   );
00159 end component;
00160
00161 --
00162 component portadora_tringular
00163   port(
00164     clk : in std_logic; -- clock
00165     en : in std_logic; -- habilita modulo
00166     reset : in std_logic; --
00167     count_ini : in std_logic_vector(15 downto 0); -- valor inicial da contagem
00168     dir_ini : in std_logic; -- direcao inicial da contagem 0: crescente ou 1: decrescente
00169     MAX : in std_logic_vector(15 downto 0); -- valor de contagem maximo
00170     dir : out std_logic; -- direcao atual 0: crescente ou 1: decrescente
00171     c : out std_logic_vector(15 downto 0) -- data out
00172   );
00173 end component;
00174
00175 --
00176 component fbpspwmdt
00177   port(
00178     clk : in std_logic; -- clock
00179     en : in std_logic; -- habilita modulo
00180     comp : in std_logic_vector(n_bits_c-1 downto 0); -- moduladora
00181     c : in std_logic_vector(n_bits_c-1 downto 0); -- portadora
00182     amost : in std_logic; -- amostra moduladora na borda de amost ??????
00183     port_PWM01 : out std_logic;
00184     port_PWM02 : out std_logic
00185   );
00186 end component;

```

```

00187
00188 --
00189 component vfcontrol
00190     port(
00191         clk : in std_logic; -- clock
00192         en : in std_logic; -- enable
00193         inc_data : out std_logic_vector(12 downto 0); -- incremento do integrador
00194         m_vf : out sfixed(I downto -F) -- Indice de modulação em Q15
00195     );
00196 end component;
00197
00198
00199 -- SIGNALS ---
00200 signal clk_pll, pll_lock, clk_wt : std_logic;
00201 signal clk_int, clk_vf, clk_led : std_logic;
00202 signal reset : std_logic;
00203
00204 signal th_a, th_b, th_c : std_logic_vector(15 downto 0); --
00206
00207 signal sigPWM01,sigPWM02 : std_logic;
00208
00209 signal OSC_BUS1 : std_logic_vector(9 downto 0);
00210 signal sinc_int, sinc_wt : std_logic;
00211 signal pulso_key0, key0_ant : std_logic;
00212 signal pulso_key1, key1_ant : std_logic;
00213 signal toggle_key1, toggle_key0 : std_logic := '1';
00214
00215 signal rst : std_logic := '1';
00216
00217 -- comparadores
00218 signal moduladoras : COMP_ARRAY;
00219 signal portadoras : COMP_ARRAY;
00220 signal amostragem_moduladoras : std_logic_vector(24 downto 1);
00221 signal en_PWM,en_PWMA,en_PWMB,en_PWMC : std_logic := '1'; -- Enable all PWM by
00222 default
00222 signal err_FA,err_FB,err_FC,err_FABC :std_logic;
00223
00224
00225
00226 signal sin_a, sin_b, sin_c : std_logic_vector(n_bits_c-1 downto 0);
00227 signal ma, mb, mc : std_logic_vector(n_bits_c-1 downto 0);
00228 signal mVF : sfixed(I downto -F);
00229 signal incVF : std_logic_vector(12 downto 0);
00230
00231 signal bidir : std_logic;
00232
00233 --PWM para indice de modulacao baixo
00234 signal dirPWM1,dirPWM2,dirPWM3 : std_logic;
00235 signal cPWM1,cPWM2,cPWM3 : std_logic_vector(15 downto 0);
00236
00237
00238 signal dirTRI1,dirTRI2,dirTRI3,dirTRI4,dirTRI5,
00239     dirTRI6 : std_logic;
00240 signal cTRI1,cTRI2,cTRI3,cTRI4,cTRI5,cTRI6 : std_logic_vector(15 downto 0);
00241
00242 signal omega_pll : std_logic_vector(Nin-1 downto 0);
00243 signal theta_pll : std_logic_vector(Nout-1 downto 0);
00244 signal theta_wt : std_logic_vector(15 downto 0);
00245
00246 begin
00247
00248
00249     -- PLL -> pll_lock = 53.333_ MHz
00250     upll: pll port map (areset => '0',
00251                           inclk0 => CLOCK_50,
00252                           c0 => clk_pll,
00253                           locked => pll_lock
00254 );
00255
00256
00257     -- clk_int = 6.666_ MHz
00258     u1: clk_div port map (clk_in => clk_pll,
00259                             en => '1',
00260                             div => std_logic_vector(to_unsigned(4, 16)),
00261                             clk_out => clk_int
00262 );
00263
00264     -- clk_vf = 217.45 Hz -> 20 s
00265     uclkVF: clk_div port map (clk_in => clk_int, -- 4349
00266                               en => '1',
00267                               div => std_logic_vector(to_unsigned(15329, 16)), -- Divide clk por
00268                               clk_out => clk_vf
00269 );
00270

```

```

00271
00272
00273     -- int_data = 4832 => 60 Hz
00274     -- 483 => 6 Hz
00275
00276 uVF: vfcontrol port map( clk => clk_vf, -- clock
00277     en => SW(3) and en_PWM, -- enable VF
00278     inc_data => incVF, -- incremento do integrador
00279     m_vf => mVF -- Indice de modulação em Q15
00280 );
00281
00282 LED(3) <= SW(3); --
00283
00284
00285
00286 u5: integrador port map(
00287     clk => clk_int, -- clk_int = 6.666_ MHz
00288     en => '1',
00289     reset => rst,
00290     MAX => std_logic_vector(to_unsigned(536870911, 30)),
00291     sinc => sinc_int,
00292     out_data => theta_pll,
00293     int_data => incVF -- Incremento do contador.
00294     --int_data => omega_pll
00295 );
00296
00297
00298 ----- Portadoras Triangulares -----
00299 ucr1: portadora_tringular port map(
00300     clk => clk_pll, -- clock
00301     en => '1', -- habilita modulo
00302     reset => rst, --
00303     count_ini => std_logic_vector(to_unsigned( 0, 16)), -- valor inicial da contagem
00304     dir_ini => '0', -- direcao inicial da contagem 0: crescente ou 1: decrescente
00305     MAX => std_logic_vector(to_unsigned( 1335, 16)), -- valor de contagem maximo
00306     -- dir => dirTRI1, -- direcao atual 0: crescente ou 1: decrescente
00307     c => cTRI1 -- data out
00308 );
00309
00310 ucr2: portadora_tringular port map(
00311     clk => clk_pll, -- clock
00312     en => '1', -- habilita modulo
00313     reset => rst, --
00314     count_ini => std_logic_vector(to_unsigned( 1335, 16)), -- valor inicial da contagem
00315     dir_ini => '0', -- direcao inicial da contagem 0: crescente ou 1: decrescente
00316     MAX => std_logic_vector(to_unsigned( 1335, 16)), -- valor de contagem maximo
00317     -- dir => dirTRI2, -- direcao atual 0: crescente ou 1: decrescente
00318     c => cTRI2 -- data out
00319 );
00320
00321 ucr3: portadora_tringular port map(
00322     clk => clk_pll, -- clock
00323     en => '1', -- habilita modulo
00324     reset => rst, --
00325     count_ini => std_logic_vector(to_unsigned( 445, 16)), -- valor inicial da contagem
00326     dir_ini => '1', -- direcao inicial da contagem 0: crescente ou 1: decrescente
00327     MAX => std_logic_vector(to_unsigned( 1335, 16)), -- valor de contagem maximo
00328     -- dir => dirTRI3, -- direcao atual 0: crescente ou 1: decrescente
00329     c => cTRI3 -- data out
00330 );
00331
00332 ucr4: portadora_tringular port map(
00333     clk => clk_pll, -- clock
00334     en => '1', -- habilita modulo
00335     reset => rst, --
00336     count_ini => std_logic_vector(to_unsigned( 890, 16)), -- valor inicial da contagem
00337     dir_ini => '0', -- direcao inicial da contagem 0: crescente ou 1: decrescente
00338     MAX => std_logic_vector(to_unsigned( 1335, 16)), -- valor de contagem maximo
00339     -- dir => dirTRI4, -- direcao atual 0: crescente ou 1: decrescente
00340     c => cTRI4 -- data out
00341 );
00342
00343 ucr5: portadora_tringular port map(
00344     clk => clk_pll, -- clock
00345     en => '1', -- habilita modulo
00346     reset => rst, --
00347     count_ini => std_logic_vector(to_unsigned( 890, 16)), -- valor inicial da contagem
00348     dir_ini => '1', -- direcao inicial da contagem 0: crescente ou 1: decrescente
00349     MAX => std_logic_vector(to_unsigned( 1335, 16)), -- valor de contagem maximo
00350     -- dir => dirTRI5, -- direcao atual 0: crescente ou 1: decrescente
00351     c => cTRI5 -- data out
00352 );
00353
00354 ucr6: portadora_tringular port map(
00355     clk => clk_pll, -- clock
00356     en => '1', -- habilita modulo
00357     reset => rst, --

```

```

00358      count_ini => std_logic_vector(to_unsigned( 445, 16)), -- valor inicial da contagem
00359      dir_ini => '0', -- direcao inicial da contagem 0: crescente ou 1: decrescente
00360      MAX => std_logic_vector(to_unsigned( 1335, 16)), -- valor de contagem maximo Q0
00361      -- dir => dirTRI6, -- direcao atual 0: crescente ou 1: decrescente
00362      c => cTRI6 -- data out
00363      );
00364
00365
00366
00367 -- defasa theta para sistema trifasico
00368 u6: theta_abc port map(
00369
00370
00371
00372
00373
00374
00375
00376
00377
00378      clk => clk_int,
00379      en => '1',
00380      reset => rst,
00381      theta_a => th_a, -- signed
00382      theta_b => th_b,
00383      theta_c => th_c, -- 16 bits
00384      theta_in => theta_pll -- 30 bits
00385      );
00386
00387
00388
00389
00390
00391
00392
00393
00394
00395
00396
00397
00398
00399
00400
00401
00402 ----- PWM ENABLE -----
00403
00404 err_FA <= INT0_FA(0) and INT0_FA(1) and INT0_FA(2); -- Ativo Baixo
00405 err_FB <= INT0_FB(0) and INT0_FB(1) and INT0_FB(2); -- Ativo Baixo
00406 err_FC <= INT0_FC(0) and INT0_FC(1) and INT0_FC(2); -- Ativo Baixo
00407 err_FABC <= err_FA and err_FB and err_FC;
00408
00409
00410 LED(6) <= not err_FABC;
00411 LED(7) <= en_PWM;
00412
00413
00414 process(err_FABC)
00415 begin
00416     if falling_edge(clk_pll) then
00417         if err_FABC = '0' then
00418             en_PWM <= '0';
00419         elsif KEY(0) = '0' and err_FABC = '1' then
00420             en_PWM <= '1';
00421         end if;
00422     end if;
00423 end process;
00424
00425
00426
00427
00428 ----- PHASE A -----
00429
00430     en_PWMA<=SW(0);
00431     LED(0)<=en_PWMA;
00432
00433
00434 PWM1_FA01 : fbpswmdt -- One leg of the Full Bridge
00435     port map(
00436         clk => clk_pll, -- clock
00437         en => en_PWMA and en_PWM , -- habilita modulo
00438         comp => ma, -- moduladora
00439         c => cTRI1, -- portadora
00440         amost => clk_pll, -- amostra moduladora na borda de amost
00441         port_PWM01 => PWM1_FA(0), -- PWM1_LOW
00442         port_PWM02 => PWM1_FA(0) --PWM1_HIGH

```

```

00443         );
00444
00445 PWM2_FA01 : fbpspwm dt -- One leg of the Full Bridge
00446     port map(
00447         clk => clk_pll, -- clock
00448         en => en_PWM_A and en_PWM , -- habilita modulo
00449         comp => ma, -- moduladora
00450         c => cTRI2, -- portadora
00451         amost => clk_pll, -- amostra moduladora na borda de amost
00452         port_PWM01 => PWM2H_FA(0) , -- PWM2_HIGH
00453         port_PWM02 => PWM2L_FA(0) -- PWM2_LOW
00454     );
00455
00456
00457 PWM1_FA02 : fbpspwm dt -- One leg of the Full Bridge
00458     port map(
00459         clk => clk_pll, -- clock
00460         en => en_PWM_A and en_PWM , -- habilita modulo
00461         comp => ma, -- moduladora
00462         c => cTRI3, -- portadora
00463         amost => clk_pll, -- amostra moduladora na borda de amost
00464         port_PWM01 => PWM1L_FA(1) , -- PWM1_LOW
00465         port_PWM02 => PWM1H_FA(1) -- PWM1_HIGH
00466     );
00467
00468 PWM2_FA02 : fbpspwm dt -- One leg of the Full Bridge
00469     port map(
00470         clk => clk_pll, -- clock
00471         en => en_PWM_A and en_PWM , -- habilita modulo
00472         comp => ma, -- moduladora
00473         c => cTRI4, -- portadora
00474         amost => clk_pll, -- amostra moduladora na borda de amost
00475         port_PWM01 => PWM2H_FA(1) , -- PWM2_HIGH
00476         port_PWM02 => PWM2L_FA(1) -- PWM2_LOW
00477     );
00478
00479
00480
00481 PWM1_FA03 : fbpspwm dt -- One leg of the Full Bridge
00482     port map(
00483         clk => clk_pll, -- clock
00484         en => en_PWM_A and en_PWM , -- habilita modulo
00485         comp => ma, -- moduladora
00486         c => cTRI5, -- portadora
00487         amost => clk_pll, -- amostra moduladora na borda de amost
00488         port_PWM01 => PWM1L_FA(2) , -- PWM1_LOW
00489         port_PWM02 => PWM1H_FA(2) -- PWM1_HIGH
00490     );
00491
00492 PWM2_FA03 : fbpspwm dt -- One leg of the Full Bridge
00493     port map(
00494         clk => clk_pll, -- clock
00495         en => en_PWM_A and en_PWM , -- habilita modulo
00496         comp => ma, -- moduladora
00497         c => cTRI6, -- portadora
00498         amost => clk_pll, -- amostra moduladora na borda de amost
00499         port_PWM01 => PWM2H_FA(2) , -- PWM2_HIGH
00500         port_PWM02 => PWM2L_FA(2) -- PWM2_LOW
00501     );
00502
00503 ----- PHASE B -----
00504     en_PWM_B<=SW(1);
00505     LED(1)<=en_PWM_B;
00506
00507
00508
00509 PWM1_FB01 : fbpspwm dt -- One leg of the Full Bridge
00510     port map(
00511         clk => clk_pll, -- clock
00512         en => en_PWM_B and en_PWM , -- habilita modulo
00513         comp => mb, -- moduladora
00514         c => cTRI1, -- portadora
00515         amost => clk_pll, -- amostra moduladora na borda de amost
00516         port_PWM01 => PWM1L_FB(0) , -- PWM1_LOW
00517         port_PWM02 => PWM1H_FB(0) -- PWM1_HIGH
00518     );
00519
00520 PWM2_FB01 : fbpspwm dt -- One leg of the Full Bridge
00521     port map(
00522         clk => clk_pll, -- clock
00523         en => en_PWM_B and en_PWM , -- habilita modulo
00524         comp => mb, -- moduladora
00525         c => cTRI2, -- portadora
00526         amost => clk_pll, -- amostra moduladora na borda de amost
00527         port_PWM01 => PWM2H_FB(0) , -- PWM2_HIGH
00528         port_PWM02 => PWM2L_FB(0) -- PWM2_LOW
00529     );

```

```

00530
00531
00532 PWM1_FB02 : fbp pwm dt -- One leg of the Full Bridge
00533     port map(
00534         clk => clk_pll, -- clock
00535         en => en_PWM_B and en_PWM , -- habilita modulo
00536         comp => mb, -- moduladora
00537         c => cTRI3, -- portadora
00538         amost => clk_pll, -- amostra moduladora na borda de amost
00539         port_PWM01 => PWM1L_FB(1) , -- PWM1_LOW
00540         port_PWM02 => PWM1H_FB(1) --PWM1_HIGH
00541     );
00542
00543 PWM2_FB02 : fbp pwm dt -- One leg of the Full Bridge
00544     port map(
00545         clk => clk_pll, -- clock
00546         en => en_PWM_B and en_PWM , -- habilita modulo
00547         comp => mb, -- moduladora
00548         c => cTRI4, -- portadora
00549         amost => clk_pll, -- amostra moduladora na borda de amost
00550         port_PWM01 => PWM2H_FB(1), -- PWM2_HIGH
00551         port_PWM02 => PWM2L_FB(1) -- PWM2_LOW
00552     );
00553
00554
00555
00556 PWM1_FB03 : fbp pwm dt -- One leg of the Full Bridge
00557     port map(
00558         clk => clk_pll, -- clock
00559         en => en_PWM_B and en_PWM , -- habilita modulo
00560         comp => mb, -- moduladora
00561         c => cTRI5, -- portadora
00562         amost => clk_pll, -- amostra moduladora na borda de amost
00563         port_PWM01 => PWM1L_FB(2) , -- PWM1_LOW
00564         port_PWM02 => PWM1H_FB(2) --PWM1_HIGH
00565     );
00566
00567 PWM2_FB03 : fbp pwm dt -- One leg of the Full Bridge
00568     port map(
00569         clk => clk_pll, -- clock
00570         en => en_PWM_B and en_PWM , -- habilita modulo
00571         comp => mb, -- moduladora
00572         c => cTRI6, -- portadora
00573         amost => clk_pll, -- amostra moduladora na borda de amost
00574         port_PWM01 => PWM2H_FB(2), -- PWM2_HIGH
00575         port_PWM02 => PWM2L_FB(2) -- PWM2_LOW
00576     );
00577
00578
00579
00580
00581 ----- PHASE C -----
00582     en_PWM_C<=SW(2);
00583     LED(2)<=en_PWM_C;
00584
00585
00586 PWM1_FC01 : fbp pwm dt -- One leg of the Full Bridge
00587     port map(
00588         clk => clk_pll, -- clock
00589         en => en_PWM_C and en_PWM , -- habilita modulo
00590         comp => mc, -- moduladora
00591         c => cTRI1, -- portadora
00592         amost => clk_pll, -- amostra moduladora na borda de amost
00593         port_PWM01 => PWM1L_FC(0) , -- PWM1_LOW
00594         port_PWM02 => PWM1H_FC(0) --PWM1_HIGH
00595     );
00596
00597 PWM2_FC01 : fbp pwm dt -- One leg of the Full Bridge
00598     port map(
00599         clk => clk_pll, -- clock
00600         en => en_PWM_C and en_PWM , -- habilita modulo
00601         comp => mc, -- moduladora
00602         c => cTRI2, -- portadora
00603         amost => clk_pll, -- amostra moduladora na borda de amost
00604         port_PWM01 => PWM2H_FC(0), -- PWM2_HIGH
00605         port_PWM02 => PWM2L_FC(0) -- PWM2_LOW
00606     );
00607
00608
00609 PWM1_FC02 : fbp pwm dt -- One leg of the Full Bridge
00610     port map(
00611         clk => clk_pll, -- clock
00612         en => en_PWM_C and en_PWM , -- habilita modulo
00613         comp => mc, -- moduladora
00614         c => cTRI3, -- portadora
00615         amost => clk_pll, -- amostra moduladora na borda de amost
00616         port_PWM01 => PWM1L_FC(1) , -- PWM1_LOW

```

```

00617      port_PWM02 => PWM1H_FC(1)    --PWM1_HIGH
00618      );
00619
00620 PWM2_FC02 : fbpspwm dt -- One leg of the Full Bridge
00621   port map(
00622     clk => clk_pll, -- clock
00623     en => en_PWMC and en_PWM , -- habilita modulo
00624     comp => mc, -- moduladora
00625     c => cTRI4, -- portadora
00626     amost => clk_pll, -- amostra moduladora na borda de amost
00627     port_PWM01 => PWM2H_FC(1), -- PWM2_HIGH
00628     port_PWM02 => PWM2L_FC(1)  -- PWM2_LOW
00629   );
00630
00631
00632
00633 PWM1_FC03 : fbpspwm dt -- One leg of the Full Bridge
00634   port map(
00635     clk => clk_pll, -- clock
00636     en => en_PWMC and en_PWM , -- habilita modulo
00637     comp => mc, -- moduladora
00638     c => cTRI5, -- portadora
00639     amost => clk_pll, -- amostra moduladora na borda de amost
00640     port_PWM01 => PWM1L_FC(2) , -- PWM1_LOW
00641     port_PWM02 => PWM1H_FC(2)  --PWM1_HIGH
00642   );
00643
00644 PWM2_FC03 : fbpspwm dt -- One leg of the Full Bridge
00645   port map(
00646     clk => clk_pll, -- clock
00647     en => en_PWMC and en_PWM , -- habilita modulo
00648     comp => mc, -- moduladora
00649     c => cTRI6, -- portadora
00650     amost => clk_pll, -- amostra moduladora na borda de amost
00651     port_PWM01 => PWM2H_FC(2), -- PWM2_HIGH
00652     port_PWM02 => PWM2L_FC(2)  -- PWM2_LOW
00653   );
00654
00655
00656
00657 process(clk_pll)
00658   begin
00659     if falling_edge(clk_pll) then
00660       rst<='0';
00661     end if;
00662   end process;
00663
00664
00665
00666 end MAIN;

```

7.13 deadtime.vhd File Reference

Entities

- **deadtime** entity
- **deadtime_archetecture** architecture

7.14 deadtime.vhd

```

00001
00002 -- Código obtido de https://elviseno.wordpress.com/2011/02/18/vhdl-code-generate-bridge-pwm-with-dead-band/
00003 library ieee;
00004 use ieee.std_logic_1164.all;
00005 use ieee.std_logic_arith.all;
00006 USE ieee.std_logic_unsigned.all;
00007
00008
00009
00010 entity deadtime is
00011
00012 port(
00013   p_Pwm_In: in std_logic ;
00014   CLK: in std_logic ;
00015   p_Pwm1_Out: out std_logic ;
00016   p_Pwm2_Out: out std_logic);
00017 end;

```

```

00018
00019 architecture deadtime_archetecture of deadtime is
00020 signal sig_Not_Pwm_In: std_logic;
00021 begin
00022     sig_Not_Pwm_In <= not p_Pwm_In;
00023
00024 process (CLK)
00025     variable var_Dead_Count1: integer range 0 to 127 := 0;
00026     variable var_Dead_Count2: integer range 0 to 127 := 0;
00027     constant c_Dead_t: integer := 15;
00028 begin
00029     if (CLK 'event and CLK = '1') then
00030         if (p_Pwm_In = '1') then
00031             if (var_Dead_Count1 < c_Dead_t) then
00032                 var_Dead_Count1 := var_Dead_Count1 + 1;
00033             else null;
00034             end if;
00035         else
00036             var_Dead_Count1 := 0;
00037             p_Pwm1_Out <= p_Pwm_In;
00038         end if;
00039
00040         if (var_Dead_Count1 = c_Dead_t) then
00041             p_Pwm1_Out <= p_Pwm_In;
00042         else null;
00043         end if;
00044 -----
00045         if (sig_Not_Pwm_In = '1') then
00046             if (var_Dead_Count2 < c_Dead_t) then
00047                 var_Dead_Count2 := var_Dead_Count2 + 1;
00048             else null;
00049             end if;
00050         else
00051             var_Dead_Count2 := 0;
00052             p_Pwm2_Out <= sig_Not_Pwm_In;
00053         end if;
00054
00055         if (var_Dead_Count2 = c_Dead_t) then
00056             p_Pwm2_Out <= sig_Not_Pwm_In;
00057         else null;
00058         end if;
00059     end if;
00060 end process;
00061
00062
00063
00064 end deadtime_archetecture;

```

7.15 fbpspwm.vhd File Reference

Entities

- [fbpspwm.vhd entity](#)
- [fbpspwm.vhd architecture](#)

7.16 fbpspwm.vhd

```

00001
00002 -- FBSPWM Full Bridge Phase Shift PWM with Dead Time
00003
00004 library IEEE;
00005 use IEEE.STD_LOGIC_UNSIGNED.all;
00006 use IEEE.STD_LOGIC_1164.all;
00007 use ieee.fixed_pkg.all;           --arquivo deve ser adicionado ao projeto
00008 use ieee.numeric_std.all;
00009
00010
00011 entity fbpspwm is
00012     generic (
00013         constant n_bits_c: integer := 16; --numero de bits da portadora
00014         constant c_Dead_t : integer := 45
00015     );
00016     port(
00017         clk : in std_logic; -- clock
00018         en : in std_logic; -- habilita modulo
00019         comp : std_logic_vector(n_bits_c-1 downto 0);-- Razão cíclica em Q0
00020         c : in std_logic_vector(n_bits_c-1 downto 0); -- portadora

```

```

00021      amost : in std_logic; -- amostra moduladora na borda de amost
00022      port_PWM01 : out std_logic;
00023      port_PWM02 : out std_logic
00024      );
00025 end entity fbp pwm dt;
00026
00027
00028 -- slv7 <= to_slv (uf7_3);
00029
00030
00031 architecture fbp pwm dt_arch of fbp pwm dt is
00032
00033
00034     signal comp_int : std_logic_vector(n_bits_c-1 downto 0); -- data out
00035     signal sig_Not_Pwm_In: std_logic;
00036     signal comp_out : std_logic;
00037     signal p_Pwm_In: std_logic ;
00038
00039
00040
00041 begin
00042     process(clk)
00043     begin
00044         if en = '0' then
00045             comp_out <= '0';
00046         elsif falling_edge(clk) then
00047             if (comp_int > c) then
00048                 comp_out <= '0';
00049             else
00050                 comp_out <= '1';
00051             end if;
00052         end if;
00053     end process;
00054
00055     process(amost)
00056     begin
00057         if rising_edge(amost) then
00058             comp_int <= comp;
00059         end if;
00060     end process;
00061
00062
00063     p_Pwm_In <= comp_out;
00064     sig_Not_Pwm_In <= not p_Pwm_In;
00065
00066
00067     process (clk)
00068     variable var_Dead_Count1: integer range 0 to 127 := 0;
00069     variable var_Dead_Count2: integer range 0 to 127 := 0;
00070
00071     begin
00072         if en = '0' then
00073             port_PWM01 <= '0';
00074             port_PWM02 <= '0';
00075             elsif (clk 'event and clk = '1') then
00076                 if (p_Pwm_In = '1') then
00077                     if (var_Dead_Count1 < c_Dead_t) then
00078                         var_Dead_Count1 := var_Dead_Count1 + 1;
00079                     else null;
00080                     end if;
00081                 else
00082                     var_Dead_Count1 := 0;
00083                     port_PWM01 <= p_Pwm_In;
00084                 end if;
00085
00086                 if (var_Dead_Count1 = c_Dead_t) then
00087                     port_PWM01 <= p_Pwm_In;
00088                 else null;
00089                 end if;
00090
00091             if (sig_Not_Pwm_In = '1') then
00092                 if (var_Dead_Count2 < c_Dead_t) then
00093                     var_Dead_Count2 := var_Dead_Count2 + 1;
00094                 else null;
00095                 end if;
00096             else
00097                 var_Dead_Count2 := 0;
00098                 port_PWM02 <= sig_Not_Pwm_In;
00099             end if;
00100
00101             if (var_Dead_Count2 = c_Dead_t) then
00102                 port_PWM02 <= sig_Not_Pwm_In;
00103             else null;
00104             end if;
00105         end if;
00106     end process;
00107

```

```
00108
00109 end architecture fbspwmtd_arch;
```

7.17 fixed_float_types_c.vhdl File Reference

Entities

- [fixed_float_types](#) package

7.18 fixed_float_types_c.vhdl

```
00001 --
00002 -- "fixed_float_types" package contains types used in the fixed and floating
00003 -- point packages..
00004 -- Please see the documentation for the floating point package.
00005 -- This package should be compiled into "ieee_proposed" and used as follows:
00006 --
00007 -- This verison is designed to work with the VHDL-93 compilers. Please
00008 -- note the "%%%" comments. These are where we diverge from the
00009 -- VHDL-200X LRM.
00010 --
00011 --
00012 -- Version      : $Revision: 1.1 $
00013 -- Date        : $Date: 2010/09/22 18:44:20 $
00014 --
00015
00016 package fixed_float_types is
00017
00018   -- Types used for generics of fixed_generic_pkg
00019
00020   type fixed_round_style_type is (fixed_round, fixed_truncate);
00021
00022   type fixed_overflow_style_type is (fixed_saturate, fixed_wrap);
00023
00024   -- Type used for generics of float_generic_pkg
00025
00026   -- These are the same as the C FE_TONEAREST, FE_UPWARD, FE_DOWNWARD,
00027   -- and FE_TOWARDZERO floating point rounding macros.
00028
00029   type round_type is (round_nearest,      -- Default, nearest LSB '0'
00030                         round_inf,          -- Round toward positive infinity
00031                         round_neginf,       -- Round toward negative infinity
00032                         round_zero);        -- Round toward zero (truncate)
00033
00034 end package fixed_float_types;
```

7.19 fixed_pkg_c.vhdl File Reference

Entities

- [fixed_pkg](#) package
- [fixed_pkg](#) package body

7.20 fixed_pkg_c.vhdl

```
00001 --
00002 -- "fixed_pkg_c.vhdl" package contains functions for fixed point math.
00003 -- Please see the documentation for the fixed point package.
00004 -- This package should be compiled into "ieee_proposed" and used as follows:
00005 -- use ieee.std_logic_1164.all;
00006 -- use ieee.numeric_std.all;
00007 -- use ieee_proposed.fixed_float_types.all;
00008 -- use ieee_proposed.fixed_pkg.all;
00009 --
00010 -- This verison is designed to work with the VHDL-93 compilers
00011 -- synthesis tools. Please note the "%%% comments. These are where we
00012 -- diverge from the VHDL-200X LRM.
```

```

00013 --
00014 -- Version      : $Revision: 1.22 $
00015 -- Date        : $Date: 2010/09/22 18:34:14 $
00016 --
00017
00018 use STD.TEXTIO.all;
00019 library IEEE;
00020 use IEEE.STD_LOGIC_1164.all;
00021 use IEEE.NUMERIC_STD.all;
00022 library IEEE_PROPOSED;
00023 use IEEE_PROPOSED.fixed_float_types.all;
00024
00025 package fixed_pkg is
00026   -- generic (
00027     -- Rounding routine to use in fixed point, fixed_round or fixed_truncate
00028     constant fixed_round_style    : fixed_round_style_type      :=
00029       fixed_round;
00029   -- Overflow routine to use in fixed point, fixed_saturate or fixed_wrap
00030   constant fixed_overflow_style : fixed_overflow_style_type := 
00031     fixed_saturate;
00031   -- Extra bits used in divide routines
00032   constant fixed_guard_bits    : NATURAL                      := 3;
00033   -- If TRUE, then turn off warnings on "X" propagation
00034   constant no_warning         : BOOLEAN                     := (false
00035                                         );
00036
00037   -- Author David Bishop (dbishop@vhdl.org)
00038
00039   -- base Unsigned fixed point type, downto direction assumed
00040   type UNRESOLVED_ufixed is array (INTEGER range <>) of STD_ULOGIC;
00041   -- base Signed fixed point type, downto direction assumed
00042   type UNRESOLVED_sfixed is array (INTEGER range <>) of STD_ULOGIC;
00043
00044   subtype U_ufixed is UNRESOLVED_ufixed;
00045   subtype U_sfixed is UNRESOLVED_sfixed;
00046
00047   subtype ufixed is UNRESOLVED_ufixed;
00048   subtype sfixed is UNRESOLVED_sfixed;
00049
00050 =====
00051   -- Arithmetic Operators:
00052 =====
00053
00054   -- Absolute value, 2's complement
00055   -- abs sfixed(a downto b) = sfixed(a+1 downto b)
00056   function "abs" (arg : UNRESOLVED_sfixed) return
00057     UNRESOLVED_sfixed;
00058
00059   -- Negation, 2's complement
00060   -- - sfixed(a downto b) = sfixed(a+1 downto b)
00061   function "-" (arg : UNRESOLVED_sfixed) return
00062     UNRESOLVED_sfixed;
00063
00064   -- Addition
00065   -- ufixed(a downto b) + ufixed(c downto d)
00066   --   = ufixed(maximum(a,c)+1 downto minimum(b,d))
00067   function "+" (l, r : UNRESOLVED_ufixed) return
00068     UNRESOLVED_ufixed;
00069
00070   -- sfixed(a downto b) + sfixed(c downto d)
00071   --   = sfixed(maximum(a,c)+1 downto minimum(b,d))
00072   function "+" (l, r : UNRESOLVED_sfixed) return
00073     UNRESOLVED_sfixed;
00074
00075   -- Subtraction
00076   -- ufixed(a downto b) - ufixed(c downto d)
00077   --   = ufixed(maximum(a,c)+1 downto minimum(b,d))
00078   function "-" (l, r : UNRESOLVED_ufixed) return
00079     UNRESOLVED_ufixed;
00080
00081   -- sfixed(a downto b) - sfixed(c downto d)
00082   --   = sfixed(maximum(a,c)+1 downto minimum(b,d))
00083   function "-" (l, r : UNRESOLVED_sfixed) return
00084     UNRESOLVED_sfixed;
00085
00086   -- Multiplication
00087   -- ufixed(a downto b) * ufixed(c downto d) = ufixed(a+c+1 downto b+d)
00088   function "*" (l, r : UNRESOLVED_ufixed) return
00089     UNRESOLVED_ufixed;
00090
00091   -- sfixed(a downto b) * sfixed(c downto d) = sfixed(a+c+1 downto b+d)
00092   function "*" (l, r : UNRESOLVED_sfixed) return
00093     UNRESOLVED_sfixed;
00094
00095   -- Division
00096   -- ufixed(a downto b) / ufixed(c downto d) = ufixed(a-d downto b-c-1)
00097   function "/" (l, r : UNRESOLVED_ufixed) return
00098     UNRESOLVED_ufixed;
00099

```

```

UNRESOLVED_ufixed;
00090
00091  -- sfixed(a downto b) / sfixed(c downto d) = sfixed(a-d+1 downto b-c)
00092  function "/" (l, r : UNRESOLVED_sfixed) return
UNRESOLVED_sfixed;
00093
00094  -- Remainder
00095  -- ufixed (a downto b) rem ufixed (c downto d)
00096  -- = ufixed (minimum(a,c) downto minimum(b,d))
00097  function "rem" (l, r : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed;
00098
00099  -- sfixed (a downto b) rem sfixed (c downto d)
00100  -- = sfixed (minimum(a,c) downto minimum(b,d))
00101  function "rem" (l, r : UNRESOLVED_sfixed) return
UNRESOLVED_sfixed;
00102
00103  -- Modulo
00104  -- ufixed (a downto b) mod ufixed (c downto d)
00105  -- = ufixed (minimum(a,c) downto minimum(b, d))
00106  function "mod" (l, r : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed;
00107
00108  -- sfixed (a downto b) mod sfixed (c downto d)
00109  -- = sfixed (c downto minimum(b, d))
00110  function "mod" (l, r : UNRESOLVED_sfixed) return
UNRESOLVED_sfixed;
00111
00112  -----
00113  -- In these routines the "real" or "natural" (integer)
00114  -- are converted into a fixed point number and then the operation is
00115  -- performed. It is assumed that the array will be large enough.
00116  -- If the input is "real" then the real number is converted into a fixed of
00117  -- the same size as the fixed point input. If the number is an "integer"
00118  -- then it is converted into fixed with the range (l'high downto 0).
00119  -----
00120
00121  -- ufixed(a downto b) + ufixed(a downto b) = ufixed(a+1 downto b)
00122  function "+" (l : UNRESOLVED_ufixed; r : REAL) return
UNRESOLVED_ufixed;
00123
00124  -- ufixed(c downto d) + ufixed(c downto d) = ufixed(c+1 downto d)
00125  function "+" (l : REAL; r : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed;
00126
00127  -- ufixed(a downto b) + ufixed(a downto 0) = ufixed(a+1 downto minimum(0,b))
00128  function "+" (l : UNRESOLVED_ufixed; r : NATURAL) return
UNRESOLVED_ufixed;
00129
00130  -- ufixed(a downto 0) + ufixed(c downto d) = ufixed(c+1 downto minimum(0,d))
00131  function "+" (l : NATURAL; r : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed;
00132
00133  -- ufixed(a downto b) - ufixed(a downto b) = ufixed(a+1 downto b)
00134  function "-" (l : UNRESOLVED_ufixed; r : REAL) return
UNRESOLVED_ufixed;
00135
00136  -- ufixed(c downto d) - ufixed(c downto d) = ufixed(c+1 downto d)
00137  function "-" (l : REAL; r : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed;
00138
00139  -- ufixed(a downto b) - ufixed(a downto 0) = ufixed(a+1 downto minimum(0,b))
00140  function "-" (l : UNRESOLVED_ufixed; r : NATURAL) return
UNRESOLVED_ufixed;
00141
00142  -- ufixed(a downto 0) + ufixed(c downto d) = ufixed(c+1 downto minimum(0,d))
00143  function "-" (l : NATURAL; r : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed;
00144
00145  -- ufixed(a downto b) * ufixed(a downto b) = ufixed(2a+1 downto 2b)
00146  function "*" (l : UNRESOLVED_ufixed; r : REAL) return
UNRESOLVED_ufixed;
00147
00148  -- ufixed(c downto d) * ufixed(c downto d) = ufixed(2c+1 downto 2d)
00149  function "*" (l : REAL; r : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed;
00150
00151  -- ufixed (a downto b) * ufixed (a downto 0) = ufixed (2a+1 downto b)
00152  function "*" (l : UNRESOLVED_ufixed; r : NATURAL) return
UNRESOLVED_ufixed;
00153
00154  -- ufixed (a downto b) * ufixed (a downto 0) = ufixed (2a+1 downto b)
00155  function "*" (l : NATURAL; r : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed;
00156
00157  -- ufixed(a downto b) / ufixed(a downto b) = ufixed(a-b downto b-a-1)
00158  function "/" (l : UNRESOLVED_ufixed; r : REAL) return

```

```

UNRESOLVED_ufixed;
00159
00160  -- ufixed(a downto b) / ufixed(a downto b) = ufixed(a-b downto b-a-1)
00161  function "/" (l : REAL; r : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed;
00162
00163  -- ufixed(a downto b) / ufixed(a downto 0) = ufixed(a downto b-a-1)
00164  function "/" (l : UNRESOLVED_ufixed; r : NATURAL) return
UNRESOLVED_ufixed;
00165
00166  -- ufixed(c downto 0) / ufixed(c downto d) = ufixed(c-d downto -c-1)
00167  function "/" (l : NATURAL; r : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed;
00168
00169  -- ufixed (a downto b) rem ufixed (a downto b) = ufixed (a downto b)
00170  function "rem" (l : UNRESOLVED_ufixed; r : REAL) return
UNRESOLVED_ufixed;
00171
00172  -- ufixed (c downto d) rem ufixed (c downto d) = ufixed (c downto d)
00173  function "rem" (l : REAL; r : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed;
00174
00175  -- ufixed (a downto b) rem ufixed (a downto 0) = ufixed (a downto minimum(b,0))
00176  function "rem" (l : UNRESOLVED_ufixed; r : NATURAL) return
UNRESOLVED_ufixed;
00177
00178  -- ufixed (c downto 0) rem ufixed (c downto d) = ufixed (c downto minimum(d,0))
00179  function "rem" (l : NATURAL; r : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed;
00180
00181  -- ufixed (a downto b) mod ufixed (a downto b) = ufixed (a downto b)
00182  function "mod" (l : UNRESOLVED_ufixed; r : REAL) return
UNRESOLVED_ufixed;
00183
00184  -- ufixed (c downto d) mod ufixed (c downto d) = ufixed (c downto d)
00185  function "mod" (l : REAL; r : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed;
00186
00187  -- ufixed (a downto b) mod ufixed (a downto 0) = ufixed (a downto minimum(b,0))
00188  function "mod" (l : UNRESOLVED_ufixed; r : NATURAL) return
UNRESOLVED_ufixed;
00189
00190  -- ufixed (c downto 0) mod ufixed (c downto d) = ufixed (c downto minimum(d,0))
00191  function "mod" (l : NATURAL; r : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed;
00192
00193  -- sfixed(a downto b) + sfixed(a downto b) = sfixed(a+1 downto b)
00194  function "+" (l : UNRESOLVED_sfixed; r : REAL) return
UNRESOLVED_sfixed;
00195
00196  -- sfixed(c downto d) + sfixed(c downto d) = sfixed(c+1 downto d)
00197  function "+" (l : REAL; r : UNRESOLVED_sfixed) return
UNRESOLVED_sfixed;
00198
00199  -- sfixed(a downto b) + sfixed(a downto 0) = sfixed(a+1 downto minimum(0,b))
00200  function "+" (l : UNRESOLVED_sfixed; r : INTEGER) return
UNRESOLVED_sfixed;
00201
00202  -- sfixed(c downto 0) + sfixed(c downto d) = sfixed(c+1 downto minimum(0,d))
00203  function "+" (l : INTEGER; r : UNRESOLVED_sfixed) return
UNRESOLVED_sfixed;
00204
00205  -- sfixed(a downto b) - sfixed(a downto b) = sfixed(a+1 downto b)
00206  function "-" (l : UNRESOLVED_sfixed; r : REAL) return
UNRESOLVED_sfixed;
00207
00208  -- sfixed(c downto d) - sfixed(c downto d) = sfixed(c+1 downto d)
00209  function "-" (l : REAL; r : UNRESOLVED_sfixed) return
UNRESOLVED_sfixed;
00210
00211  -- sfixed(a downto b) - sfixed(a downto 0) = sfixed(a+1 downto minimum(0,b))
00212  function "-" (l : UNRESOLVED_sfixed; r : INTEGER) return
UNRESOLVED_sfixed;
00213
00214  -- sfixed(c downto 0) - sfixed(c downto d) = sfixed(c+1 downto minimum(0,d))
00215  function "-" (l : INTEGER; r : UNRESOLVED_sfixed) return
UNRESOLVED_sfixed;
00216
00217  -- sfixed(a downto b) * sfixed(a downto b) = sfixed(2a+1 downto 2b)
00218  function "*" (l : UNRESOLVED_sfixed; r : REAL) return
UNRESOLVED_sfixed;
00219
00220  -- sfixed(c downto d) * sfixed(c downto d) = sfixed(2c+1 downto 2d)
00221  function "*" (l : REAL; r : UNRESOLVED_sfixed) return
UNRESOLVED_sfixed;
00222
00223  -- sfixed(a downto b) * sfixed(a downto 0) = sfixed(2a+1 downto b)

```

```

00224      function "*" (l : UNRESOLVED_sfixed; r : INTEGER) return
00225          UNRESOLVED_sfixed;
00226      -- sfixed(c downto 0) * sfixed(c downto d) = sfixed(2c+1 downto d)
00227      function "*" (l : INTEGER; r : UNRESOLVED_sfixed) return
00228          UNRESOLVED_sfixed;
00229      -- sfixed(a downto b) / sfixed(a downto b) = sfixed(a-b+1 downto b-a)
00230      function "/" (l : UNRESOLVED_sfixed; r : REAL) return
00231          UNRESOLVED_sfixed;
00232      -- sfixed(c downto d) / sfixed(c downto d) = sfixed(c-d+1 downto d-c)
00233      function "/" (l : REAL; r : UNRESOLVED_sfixed) return
00234          UNRESOLVED_sfixed;
00235      -- sfixed(a downto b) / sfixed(a downto 0) = sfixed(a+1 downto b-a)
00236      function "/" (l : UNRESOLVED_sfixed; r : INTEGER) return
00237          UNRESOLVED_sfixed;
00238      -- sfixed(c downto 0) / sfixed(c downto d) = sfixed(c-d+1 downto -c)
00239      function "/" (l : INTEGER; r : UNRESOLVED_sfixed) return
00240          UNRESOLVED_sfixed;
00241      -- sfixed (a downto b) rem sfixed (a downto b) = sfixed (a downto b)
00242      function "rem" (l : UNRESOLVED_sfixed; r : REAL) return
00243          UNRESOLVED_sfixed;
00244      -- sfixed (c downto d) rem sfixed (c downto d) = sfixed (c downto d)
00245      function "rem" (l : REAL; r : UNRESOLVED_sfixed) return
00246          UNRESOLVED_sfixed;
00247      -- sfixed (a downto b) rem sfixed (a downto 0) = sfixed (a downto minimum(b,0))
00248      function "rem" (l : UNRESOLVED_sfixed; r : INTEGER) return
00249          UNRESOLVED_sfixed;
00250      -- sfixed (c downto 0) rem sfixed (c downto d) = sfixed (c downto minimum(d,0))
00251      function "rem" (l : INTEGER; r : UNRESOLVED_sfixed) return
00252          UNRESOLVED_sfixed;
00253      -- sfixed (a downto b) mod sfixed (a downto b) = sfixed (a downto b)
00254      function "mod" (l : UNRESOLVED_sfixed; r : REAL) return
00255          UNRESOLVED_sfixed;
00256      -- sfixed (c downto d) mod sfixed (c downto d) = sfixed (c downto d)
00257      function "mod" (l : REAL; r : UNRESOLVED_sfixed) return
00258          UNRESOLVED_sfixed;
00259      -- sfixed (a downto b) mod sfixed (a downto 0) = sfixed (a downto minimum(b,0))
00260      function "mod" (l : UNRESOLVED_sfixed; r : INTEGER) return
00261          UNRESOLVED_sfixed;
00262      -- sfixed (c downto 0) mod sfixed (c downto d) = sfixed (c downto minimum(d,0))
00263      function "mod" (l : INTEGER; r : UNRESOLVED_sfixed) return
00264          UNRESOLVED_sfixed;
00265      -- This version of divide gives the user more control
00266      -- ufixed(a downto b) / ufixed(c downto d) = ufixed(a-d downto b-c-1)
00267      function divide (
00268          l, r           : UNRESOLVED_ufixed;
00269          constant round_style : fixed_round_style_type := 
00270              fixed_round_style;
00271          constant guard_bits : NATURAL                  := fixed_guard_bits)
00272      return UNRESOLVED_ufixed;
00273      -- This version of divide gives the user more control
00274      -- sfixed(a downto b) / sfixed(c downto d) = sfixed(a-d+1 downto b-c)
00275      function divide (
00276          l, r           : UNRESOLVED_sfixed;
00277          constant round_style : fixed_round_style_type := 
00278              fixed_round_style;
00279          constant guard_bits : NATURAL                  := fixed_guard_bits)
00280      return UNRESOLVED_sfixed;
00281      -- These functions return 1/X
00282      -- 1 / ufixed(a downto b) = ufixed(-b downto -a-1)
00283      function reciprocal (
00284          arg            : UNRESOLVED_ufixed; -- fixed point input
00285          constant round_style : fixed_round_style_type := 
00286              fixed_round_style;
00287          constant guard_bits : NATURAL                  := fixed_guard_bits)
00288      return UNRESOLVED_ufixed;
00289      -- 1 / sfixed(a downto b) = sfixed(-b+1 downto -a)
00290      function reciprocal (
00291          arg            : UNRESOLVED_sfixed; -- fixed point input
00292          constant round_style : fixed_round_style_type := 
00293              fixed_round_style;

```

```

00293     constant guard_bits : NATURAL           := fixed_guard_bits)
00294     return UNRESOLVED_sfixed;
00295
00296 -- REM function
00297 -- ufixed (a downto b) rem ufixed (c downto d)
00298 --   = ufixed (minimum(a,c) downto minimum(b,d))
00299 function remainder (
00300     l, r           : UNRESOLVED_ufixed;
00301     constant round_style : fixed_round_style_type := 
00302         fixed_round_style;
00303     constant guard_bits : NATURAL           := fixed_guard_bits)
00304     return UNRESOLVED_ufixed;
00305
00306 -- sfixed (a downto b) rem sfixed (c downto d)
00307 --   = sfixed (minimum(a,c) downto minimum(b,d))
00308 function remainder (
00309     l, r           : UNRESOLVED_sfixed;
00310     constant round_style : fixed_round_style_type := 
00311         fixed_round_style;
00312     constant guard_bits : NATURAL           := fixed_guard_bits)
00313     return UNRESOLVED_sfixed;
00314
00315 -- mod function
00316 -- ufixed (a downto b) mod ufixed (c downto d)
00317 --   = ufixed (minimum(a,c) downto minimum(b,d))
00318 function modulo (
00319     l, r           : UNRESOLVED_ufixed;
00320     constant round_style : fixed_round_style_type := 
00321         fixed_round_style;
00322     constant guard_bits : NATURAL           := fixed_guard_bits)
00323     return UNRESOLVED_ufixed;
00324
00325 -- sfixed (a downto b) mod sfixed (c downto d)
00326 --   = sfixed (c downto minimum(b, d))
00327 function modulo (
00328     l, r           : UNRESOLVED_sfixed;
00329     constant overflow_style : fixed_overflow_style_type := 
00330         fixed_overflow_style;
00331     constant round_style : fixed_round_style_type := 
00332         fixed_round_style;
00333     constant guard_bits : NATURAL           := fixed_guard_bits)
00334     return UNRESOLVED_sfixed;
00335
00336 -- Procedure for those who need an "accumulator" function.
00337 -- add_carry (ufixed(a downto b), ufixed (c downto d)
00338 --   = ufixed (maximum(a,c) downto minimum(b,d))
00339 procedure add_carry (
00340     L, R : in UNRESOLVED_ufixed;
00341     c_in : in STD_ULONGIC;
00342     result : out UNRESOLVED_ufixed;
00343     c_out : out STD_ULONGIC);
00344
00345 -- add_carry (sfixed(a downto b), sfixed (c downto d))
00346 --   = sfixed (maximum(a,c) downto minimum(b,d))
00347 procedure add_carry (
00348     L, R : in UNRESOLVED_sfixed;
00349     c_in : in STD_ULONGIC;
00350     result : out UNRESOLVED_sfixed;
00351     c_out : out STD_ULONGIC);
00352
00353 -- Scales the result by a power of 2. Width of input = width of output with
00354 -- the binary point moved.
00355 function scalb (y : UNRESOLVED_ufixed; N : INTEGER) return
00356     UNRESOLVED_ufixed;
00357 function scalb (y : UNRESOLVED_sfixed; N : SIGNED) return
00358     UNRESOLVED_ufixed;
00359 function scalb (y : UNRESOLVED_sfixed; N : INTEGER) return
00360     UNRESOLVED_sfixed;
00361 function scalb (y : UNRESOLVED_sfixed; N : SIGNED) return
00362     UNRESOLVED_sfixed;
00363
00364 function Is_Negative (arg : UNRESOLVED_sfixed) return BOOLEAN;
00365
00366 =====
00367 -- Comparison Operators
00368 =====
00369
00370 function ">"  (l, r : UNRESOLVED_ufixed) return BOOLEAN;
00371 function ">"  (l, r : UNRESOLVED_sfixed) return BOOLEAN;
00372 function "<"  (l, r : UNRESOLVED_ufixed) return BOOLEAN;
00373 function "<"  (l, r : UNRESOLVED_sfixed) return BOOLEAN;
00374 function "<=" (l, r : UNRESOLVED_ufixed) return BOOLEAN;
00375 function "<=" (l, r : UNRESOLVED_sfixed) return BOOLEAN;
00376 function ">=" (l, r : UNRESOLVED_ufixed) return BOOLEAN;
00377 function ">=" (l, r : UNRESOLVED_sfixed) return BOOLEAN;
00378 function "="   (l, r : UNRESOLVED_ufixed) return BOOLEAN;
00379 function "="   (l, r : UNRESOLVED_sfixed) return BOOLEAN;

```

```

00371  function "/=" (l, r : UNRESOLVED_ufixed) return BOOLEAN;
00372  function "/=" (l, r : UNRESOLVED_sfixed) return BOOLEAN;
00373
00374  function "\?=" (l, r : UNRESOLVED_ufixed) return STD_ULOGIC;
00375  function "\?/=" (l, r : UNRESOLVED_ufixed) return STD_ULOGIC;
00376  function "\?>" (l, r : UNRESOLVED_ufixed) return STD_ULOGIC;
00377  function "\?>=" (l, r : UNRESOLVED_ufixed) return STD_ULOGIC;
00378  function "\?<" (l, r : UNRESOLVED_ufixed) return STD_ULOGIC;
00379  function "\?<=" (l, r : UNRESOLVED_ufixed) return STD_ULOGIC;
00380  function "\?=" (l, r : UNRESOLVED_sfixed) return STD_ULOGIC;
00381  function "\?/=" (l, r : UNRESOLVED_sfixed) return STD_ULOGIC;
00382  function "\?>" (l, r : UNRESOLVED_sfixed) return STD_ULOGIC;
00383  function "\?>=" (l, r : UNRESOLVED_sfixed) return STD_ULOGIC;
00384  function "\?<" (l, r : UNRESOLVED_sfixed) return STD_ULOGIC;
00385  function "\?<=" (l, r : UNRESOLVED_sfixed) return STD_ULOGIC;
00386
00387  function std_match (l, r : UNRESOLVED_ufixed) return BOOLEAN;
00388  function std_match (l, r : UNRESOLVED_sfixed) return BOOLEAN;
00389
00390  -- Overloads the default "maximum" and "minimum" function
00391
00392  function maximum (l, r : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed;
00393  function minimum (l, r : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed;
00394  function maximum (l, r : UNRESOLVED_sfixed) return
UNRESOLVED_sfixed;
00395  function minimum (l, r : UNRESOLVED_sfixed) return
UNRESOLVED_sfixed;
00396
00397  -----
00398  -- In these compare functions a natural is converted into a
00399  -- fixed point number of the bounds "maximum(l'high,0) downto 0"
00400
00401
00402  function "==" (l : UNRESOLVED_ufixed; r : NATURAL) return BOOLEAN;
00403  function "/=" (l : UNRESOLVED_ufixed; r : NATURAL) return BOOLEAN;
00404  function ">=" (l : UNRESOLVED_ufixed; r : NATURAL) return BOOLEAN;
00405  function "<=" (l : UNRESOLVED_ufixed; r : NATURAL) return BOOLEAN;
00406  function ">" (l : UNRESOLVED_ufixed; r : NATURAL) return BOOLEAN;
00407  function "<" (l : UNRESOLVED_ufixed; r : NATURAL) return BOOLEAN;
00408
00409  function "==" (l : NATURAL; r : UNRESOLVED_ufixed) return BOOLEAN;
00410  function "/=" (l : NATURAL; r : UNRESOLVED_ufixed) return BOOLEAN;
00411  function ">=" (l : NATURAL; r : UNRESOLVED_ufixed) return BOOLEAN;
00412  function "<=" (l : NATURAL; r : UNRESOLVED_ufixed) return BOOLEAN;
00413  function ">" (l : NATURAL; r : UNRESOLVED_ufixed) return BOOLEAN;
00414  function "<" (l : NATURAL; r : UNRESOLVED_ufixed) return BOOLEAN;
00415
00416  function "\?=" (l : UNRESOLVED_ufixed; r : NATURAL) return STD_ULOGIC;
00417  function "\?/=" (l : UNRESOLVED_ufixed; r : NATURAL) return STD_ULOGIC;
00418  function "\?>=" (l : UNRESOLVED_ufixed; r : NATURAL) return STD_ULOGIC;
00419  function "\?<=" (l : UNRESOLVED_ufixed; r : NATURAL) return STD_ULOGIC;
00420  function "\?>" (l : UNRESOLVED_ufixed; r : NATURAL) return STD_ULOGIC;
00421  function "\?<" (l : UNRESOLVED_ufixed; r : NATURAL) return STD_ULOGIC;
00422
00423  function "\?=" (l : NATURAL; r : UNRESOLVED_ufixed) return STD_ULOGIC;
00424  function "\?/=" (l : NATURAL; r : UNRESOLVED_ufixed) return STD_ULOGIC;
00425  function "\?>=" (l : NATURAL; r : UNRESOLVED_ufixed) return STD_ULOGIC;
00426  function "\?<=" (l : NATURAL; r : UNRESOLVED_ufixed) return STD_ULOGIC;
00427  function "\?>" (l : NATURAL; r : UNRESOLVED_ufixed) return STD_ULOGIC;
00428  function "\?<" (l : NATURAL; r : UNRESOLVED_ufixed) return STD_ULOGIC;
00429
00430  function maximum (l : UNRESOLVED_ufixed; r : NATURAL)
00431    return UNRESOLVED_ufixed;
00432  function minimum (l : UNRESOLVED_ufixed; r : NATURAL)
00433    return UNRESOLVED_ufixed;
00434  function maximum (l : NATURAL; r : UNRESOLVED_ufixed)
00435    return UNRESOLVED_ufixed;
00436  function minimum (l : NATURAL; r : UNRESOLVED_ufixed)
00437    return UNRESOLVED_ufixed;
00438  -----
00439  -- In these compare functions a real is converted into a
00440  -- fixed point number of the bounds "l'high+1 downto l'low"
00441
00442
00443  function "==" (l : UNRESOLVED_ufixed; r : REAL) return BOOLEAN;
00444  function "/=" (l : UNRESOLVED_ufixed; r : REAL) return BOOLEAN;
00445  function ">=" (l : UNRESOLVED_ufixed; r : REAL) return BOOLEAN;
00446  function "<=" (l : UNRESOLVED_ufixed; r : REAL) return BOOLEAN;
00447  function ">" (l : UNRESOLVED_ufixed; r : REAL) return BOOLEAN;
00448  function "<" (l : UNRESOLVED_ufixed; r : REAL) return BOOLEAN;
00449
00450  function "==" (l : REAL; r : UNRESOLVED_ufixed) return BOOLEAN;
00451  function "/=" (l : REAL; r : UNRESOLVED_ufixed) return BOOLEAN;
00452  function ">=" (l : REAL; r : UNRESOLVED_ufixed) return BOOLEAN;
00453  function "<=" (l : REAL; r : UNRESOLVED_ufixed) return BOOLEAN;

```

```

00454     function ">"  (l : REAL; r : UNRESOLVED_ufixed) return BOOLEAN;
00455     function "<"  (l : REAL; r : UNRESOLVED_ufixed) return BOOLEAN;
00456
00457     function \?=\ (l : UNRESOLVED_ufixed; r : REAL) return STD_ULOGIC;
00458     function \?/= (l : UNRESOLVED_ufixed; r : REAL) return STD_ULOGIC;
00459     function \?>= (l : UNRESOLVED_ufixed; r : REAL) return STD_ULOGIC;
00460     function \?<= (l : UNRESOLVED_ufixed; r : REAL) return STD_ULOGIC;
00461     function \?> (l : UNRESOLVED_ufixed; r : REAL) return STD_ULOGIC;
00462     function \?< (l : UNRESOLVED_ufixed; r : REAL) return STD_ULOGIC;
00463
00464     function \?=\ (l : REAL; r : UNRESOLVED_ufixed) return STD_ULOGIC;
00465     function \?/= (l : REAL; r : UNRESOLVED_ufixed) return STD_ULOGIC;
00466     function \?>= (l : REAL; r : UNRESOLVED_ufixed) return STD_ULOGIC;
00467     function \?<= (l : REAL; r : UNRESOLVED_ufixed) return STD_ULOGIC;
00468     function \?> (l : REAL; r : UNRESOLVED_ufixed) return STD_ULOGIC;
00469     function \?< (l : REAL; r : UNRESOLVED_ufixed) return STD_ULOGIC;
00470
00471     function maximum (l : UNRESOLVED_ufixed; r : REAL) return
UNRESOLVED_ufixed;
00472     function maximum (l : REAL; r : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed;
00473     function minimum (l : UNRESOLVED_ufixed; r : REAL) return
UNRESOLVED_ufixed;
00474     function minimum (l : REAL; r : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed;
00475
-----  

00476 -- In these compare functions an integer is converted into a
00477 -- fixed point number of the bounds "maximum(l'high,1) downto 0"
00478 -----  

00479
00480     function "="  (l : UNRESOLVED_sfixed; r : INTEGER) return BOOLEAN;
00481     function "/=" (l : UNRESOLVED_sfixed; r : INTEGER) return BOOLEAN;
00482     function ">=" (l : UNRESOLVED_sfixed; r : INTEGER) return BOOLEAN;
00483     function "<=" (l : UNRESOLVED_sfixed; r : INTEGER) return BOOLEAN;
00484     function ">"  (l : UNRESOLVED_sfixed; r : INTEGER) return BOOLEAN;
00485     function "<"  (l : UNRESOLVED_sfixed; r : INTEGER) return BOOLEAN;
00486
00487     function "="  (l : INTEGER; r : UNRESOLVED_sfixed) return BOOLEAN;
00488     function "/=" (l : INTEGER; r : UNRESOLVED_sfixed) return BOOLEAN;
00489     function ">=" (l : INTEGER; r : UNRESOLVED_sfixed) return BOOLEAN;
00490     function "<=" (l : INTEGER; r : UNRESOLVED_sfixed) return BOOLEAN;
00491     function ">"  (l : INTEGER; r : UNRESOLVED_sfixed) return BOOLEAN;
00492     function "<"  (l : INTEGER; r : UNRESOLVED_sfixed) return BOOLEAN;
00493
00494     function \?=\ (l : UNRESOLVED_sfixed; r : INTEGER) return STD_ULOGIC;
00495     function \?/= (l : UNRESOLVED_sfixed; r : INTEGER) return STD_ULOGIC;
00496     function \?>= (l : UNRESOLVED_sfixed; r : INTEGER) return STD_ULOGIC;
00497     function \?<= (l : UNRESOLVED_sfixed; r : INTEGER) return STD_ULOGIC;
00498     function \?> (l : UNRESOLVED_sfixed; r : INTEGER) return STD_ULOGIC;
00499     function \?< (l : UNRESOLVED_sfixed; r : INTEGER) return STD_ULOGIC;
00500
00501     function \?=\ (l : INTEGER; r : UNRESOLVED_sfixed) return STD_ULOGIC;
00502     function \?/= (l : INTEGER; r : UNRESOLVED_sfixed) return STD_ULOGIC;
00503     function \?>= (l : INTEGER; r : UNRESOLVED_sfixed) return STD_ULOGIC;
00504     function \?<= (l : INTEGER; r : UNRESOLVED_sfixed) return STD_ULOGIC;
00505     function \?> (l : INTEGER; r : UNRESOLVED_sfixed) return STD_ULOGIC;
00506     function \?< (l : INTEGER; r : UNRESOLVED_sfixed) return STD_ULOGIC;
00507
00508     function maximum (l : UNRESOLVED_sfixed; r : INTEGER)
00509         return UNRESOLVED_sfixed;
00510     function maximum (l : INTEGER; r : UNRESOLVED_sfixed)
00511         return UNRESOLVED_sfixed;
00512     function minimum (l : UNRESOLVED_sfixed; r : INTEGER)
00513         return UNRESOLVED_sfixed;
00514     function minimum (l : INTEGER; r : UNRESOLVED_sfixed)
00515         return UNRESOLVED_sfixed;
00516
-----  

00517 -- In these compare functions a real is converted into a
00518 -- fixed point number of the bounds "l'high+1 downto l'low"
00519 -----  

00520
00521     function "="  (l : UNRESOLVED_sfixed; r : REAL) return BOOLEAN;
00522     function "/=" (l : UNRESOLVED_sfixed; r : REAL) return BOOLEAN;
00523     function ">=" (l : UNRESOLVED_sfixed; r : REAL) return BOOLEAN;
00524     function "<=" (l : UNRESOLVED_sfixed; r : REAL) return BOOLEAN;
00525     function ">"  (l : UNRESOLVED_sfixed; r : REAL) return BOOLEAN;
00526     function "<"  (l : UNRESOLVED_sfixed; r : REAL) return BOOLEAN;
00527
00528     function "="  (l : REAL; r : UNRESOLVED_sfixed) return BOOLEAN;
00529     function "/=" (l : REAL; r : UNRESOLVED_sfixed) return BOOLEAN;
00530     function ">=" (l : REAL; r : UNRESOLVED_sfixed) return BOOLEAN;
00531     function "<=" (l : REAL; r : UNRESOLVED_sfixed) return BOOLEAN;
00532     function ">"  (l : REAL; r : UNRESOLVED_sfixed) return BOOLEAN;
00533     function "<"  (l : REAL; r : UNRESOLVED_sfixed) return BOOLEAN;
00534
00535     function \?=\ (l : UNRESOLVED_sfixed; r : REAL) return STD_ULOGIC;
00536     function \?/= (l : UNRESOLVED_sfixed; r : REAL) return STD_ULOGIC;

```

```

00537  function \?>=\ (l : UNRESOLVED_sfixed; r : REAL) return STD_ULOGIC;
00538  function \?<=\ (l : UNRESOLVED_sfixed; r : REAL) return STD_ULOGIC;
00539  function \?>\ (l : UNRESOLVED_sfixed; r : REAL) return STD_ULOGIC;
00540  function \?<\ (l : UNRESOLVED_sfixed; r : REAL) return STD_ULOGIC;
00541
00542  function \?=\ (l : REAL; r : UNRESOLVED_sfixed) return STD_ULOGIC;
00543  function \?/-\ (l : REAL; r : UNRESOLVED_sfixed) return STD_ULOGIC;
00544  function \?>=\ (l : REAL; r : UNRESOLVED_sfixed) return STD_ULOGIC;
00545  function \?<=\ (l : REAL; r : UNRESOLVED_sfixed) return STD_ULOGIC;
00546  function \?>\ (l : REAL; r : UNRESOLVED_sfixed) return STD_ULOGIC;
00547  function \?<\ (l : REAL; r : UNRESOLVED_sfixed) return STD_ULOGIC;
00548
00549  function maximum (l : UNRESOLVED_sfixed; r : REAL) return
UNRESOLVED_sfixed;
00550  function maximum (l : REAL; r : UNRESOLVED_sfixed) return
UNRESOLVED_sfixed;
00551  function minimum (l : UNRESOLVED_sfixed; r : REAL) return
UNRESOLVED_sfixed;
00552  function minimum (l : REAL; r : UNRESOLVED_sfixed) return
UNRESOLVED_sfixed;
00553  =====
00554  -- Shift and Rotate Functions.
00555  -- Note that sra and sla are not the same as the BIT_VECTOR version
00556  =====
00557
00558  function "sll" (ARG : UNRESOLVED_ufixed; COUNT : INTEGER)
00559    return UNRESOLVED_ufixed;
00560  function "srl" (ARG : UNRESOLVED_ufixed; COUNT : INTEGER)
00561    return UNRESOLVED_ufixed;
00562  function "rol" (ARG : UNRESOLVED_ufixed; COUNT : INTEGER)
00563    return UNRESOLVED_ufixed;
00564  function "ror" (ARG : UNRESOLVED_ufixed; COUNT : INTEGER)
00565    return UNRESOLVED_ufixed;
00566  function "sla" (ARG : UNRESOLVED_ufixed; COUNT : INTEGER)
00567    return UNRESOLVED_ufixed;
00568  function "sra" (ARG : UNRESOLVED_ufixed; COUNT : INTEGER)
00569    return UNRESOLVED_ufixed;
00570  function "sll" (ARG : UNRESOLVED_sfixed; COUNT : INTEGER)
00571    return UNRESOLVED_sfixed;
00572  function "srl" (ARG : UNRESOLVED_sfixed; COUNT : INTEGER)
00573    return UNRESOLVED_sfixed;
00574  function "rol" (ARG : UNRESOLVED_sfixed; COUNT : INTEGER)
00575    return UNRESOLVED_sfixed;
00576  function "ror" (ARG : UNRESOLVED_sfixed; COUNT : INTEGER)
00577    return UNRESOLVED_sfixed;
00578  function "sla" (ARG : UNRESOLVED_sfixed; COUNT : INTEGER)
00579    return UNRESOLVED_sfixed;
00580  function "sra" (ARG : UNRESOLVED_sfixed; COUNT : INTEGER)
00581    return UNRESOLVED_sfixed;
00582  function SHIFT_LEFT (ARG : UNRESOLVED_ufixed; COUNT : NATURAL)
00583    return UNRESOLVED_ufixed;
00584  function SHIFT_RIGHT (ARG : UNRESOLVED_ufixed; COUNT : NATURAL)
00585    return UNRESOLVED_ufixed;
00586  function SHIFT_LEFT (ARG : UNRESOLVED_sfixed; COUNT : NATURAL)
00587    return UNRESOLVED_sfixed;
00588  function SHIFT_RIGHT (ARG : UNRESOLVED_sfixed; COUNT : NATURAL)
00589    return UNRESOLVED_sfixed;
00590
00591  -----
00592  -- logical functions
00593  -----
00594
00595  function "not" (l : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed;
00596  function "and" (l, r : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed;
00597  function "or" (l, r : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed;
00598  function "nand" (l, r : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed;
00599  function "nor" (l, r : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed;
00600  function "xor" (l, r : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed;
00601  function "xnor" (l, r : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed;
00602  function "not" (l : UNRESOLVED_sfixed) return
UNRESOLVED_sfixed;
00603  function "and" (l, r : UNRESOLVED_sfixed) return
UNRESOLVED_sfixed;
00604  function "or" (l, r : UNRESOLVED_sfixed) return
UNRESOLVED_sfixed;
00605  function "nand" (l, r : UNRESOLVED_sfixed) return
UNRESOLVED_sfixed;
00606  function "nor" (l, r : UNRESOLVED_sfixed) return
UNRESOLVED_sfixed;
00607  function "xor" (l, r : UNRESOLVED_sfixed) return
UNRESOLVED_sfixed;

```

```

UNRESOLVED_sfixed;
00608 function "xnor" (l, r : UNRESOLVED_sfixed) return
UNRESOLVED_sfixed;
00609
00610 -- Vector and std_ulogic functions, same as functions in numeric_std
00611 function "and" (l : STD_ULOGIC; r : UNRESOLVED_ufixed)
00612   return UNRESOLVED_ufixed;
00613 function "and" (l : UNRESOLVED_ufixed; r : STD_ULOGIC)
00614   return UNRESOLVED_ufixed;
00615 function "or" (l : STD_ULOGIC; r : UNRESOLVED_ufixed)
00616   return UNRESOLVED_ufixed;
00617 function "or" (l : UNRESOLVED_ufixed; r : STD_ULOGIC)
00618   return UNRESOLVED_ufixed;
00619 function "nand" (l : STD_ULOGIC; r : UNRESOLVED_ufixed)
00620   return UNRESOLVED_ufixed;
00621 function "nand" (l : UNRESOLVED_ufixed; r : STD_ULOGIC)
00622   return UNRESOLVED_ufixed;
00623 function "nor" (l : STD_ULOGIC; r : UNRESOLVED_ufixed)
00624   return UNRESOLVED_ufixed;
00625 function "nor" (l : UNRESOLVED_ufixed; r : STD_ULOGIC)
00626   return UNRESOLVED_ufixed;
00627 function "xor" (l : STD_ULOGIC; r : UNRESOLVED_ufixed)
00628   return UNRESOLVED_ufixed;
00629 function "xor" (l : UNRESOLVED_ufixed; r : STD_ULOGIC)
00630   return UNRESOLVED_ufixed;
00631 function "xnor" (l : STD_ULOGIC; r : UNRESOLVED_ufixed)
00632   return UNRESOLVED_ufixed;
00633 function "xnor" (l : UNRESOLVED_ufixed; r : STD_ULOGIC)
00634   return UNRESOLVED_ufixed;
00635 function "and" (l : STD_ULOGIC; r : UNRESOLVED_sfixed)
00636   return UNRESOLVED_sfixed;
00637 function "and" (l : UNRESOLVED_sfixed; r : STD_ULOGIC)
00638   return UNRESOLVED_sfixed;
00639 function "or" (l : STD_ULOGIC; r : UNRESOLVED_sfixed)
00640   return UNRESOLVED_sfixed;
00641 function "or" (l : UNRESOLVED_sfixed; r : STD_ULOGIC)
00642   return UNRESOLVED_sfixed;
00643 function "nand" (l : STD_ULOGIC; r : UNRESOLVED_sfixed)
00644   return UNRESOLVED_sfixed;
00645 function "nor" (l : STD_ULOGIC; r : UNRESOLVED_sfixed)
00646   return UNRESOLVED_sfixed;
00647 function "xor" (l : STD_ULOGIC; r : UNRESOLVED_sfixed)
00648   return UNRESOLVED_sfixed;
00649 function "nor" (l : UNRESOLVED_sfixed; r : STD_ULOGIC)
00650   return UNRESOLVED_sfixed;
00651 function "xor" (l : STD_ULOGIC; r : UNRESOLVED_sfixed)
00652   return UNRESOLVED_sfixed;
00653 function "xor" (l : UNRESOLVED_sfixed; r : STD_ULOGIC)
00654   return UNRESOLVED_sfixed;
00655 function "xnor" (l : STD_ULOGIC; r : UNRESOLVED_sfixed)
00656   return UNRESOLVED_sfixed;
00657 function "xnor" (l : UNRESOLVED_sfixed; r : STD_ULOGIC)
00658   return UNRESOLVED_sfixed;
00659
00660 -- Reduction operators, same as numeric_std functions
00661 function and_reduce (l : UNRESOLVED_ufixed) return STD_ULOGIC;
00662 function hand_reduce (l : UNRESOLVED_ufixed) return STD_ULOGIC;
00663 function or_reduce (l : UNRESOLVED_ufixed) return STD_ULOGIC;
00664 function nor_reduce (l : UNRESOLVED_ufixed) return STD_ULOGIC;
00665 function xor_reduce (l : UNRESOLVED_ufixed) return STD_ULOGIC;
00666 function xnor_reduce (l : UNRESOLVED_ufixed) return STD_ULOGIC;
00667 function and_reduce (l : UNRESOLVED_sfixed) return STD_ULOGIC;
00668 function hand_reduce (l : UNRESOLVED_sfixed) return STD_ULOGIC;
00669 function or_reduce (l : UNRESOLVED_sfixed) return STD_ULOGIC;
00670 function nor_reduce (l : UNRESOLVED_sfixed) return STD_ULOGIC;
00671 function xor_reduce (l : UNRESOLVED_sfixed) return STD_ULOGIC;
00672 function xnor_reduce (l : UNRESOLVED_sfixed) return STD_ULOGIC;
00673
00674 -- returns arg'low-1 if not found
00675 function find_leftmost (arg : UNRESOLVED_ufixed; y : STD_ULOGIC)
00676   return INTEGER;
00677 function find_leftmost (arg : UNRESOLVED_sfixed; y : STD_ULOGIC)
00678   return INTEGER;
00679
00680 -- returns arg'high+1 if not found
00681 function find_rightmost (arg : UNRESOLVED_ufixed; y : STD_ULOGIC)
00682   return INTEGER;
00683 function find_rightmost (arg : UNRESOLVED_sfixed; y : STD_ULOGIC)
00684   return INTEGER;
00685
00686 =====
00687 -- RESIZE Functions
00688 =====
00689 -- resizes the number (larger or smaller)
00690 -- The returned result will be ufixed (left_index downto right_index)
00691 -- If "round_style" is fixed_round, then the result will be rounded.
00692 -- If the MSB of the remainder is a "1" AND the LSB of the unrounded result

```

```

00693 -- is a '1' or the lower bits of the remainder include a '1' then the result
00694 -- will be increased by the smallest representable number for that type.
00695 -- "overflow_style" can be fixed_saturate or fixed_wrap.
00696 -- In saturate mode, if the number overflows then the largest possible
00697 -- representable number is returned. If wrap mode, then the upper bits
00698 -- of the number are truncated.
00699
00700 function resize (
00701     arg                  : UNRESOLVED_ufixed; -- input
00702     constant left_index   : INTEGER;    -- integer portion
00703     constant right_index  : INTEGER;    -- size of fraction
00704     constant overflow_style : fixed_overflow_style_type := 
00705         fixed_overflow_style;
00706     constant round_style   : fixed_round_style_type   := 
00707         fixed_round_style);
00708     return UNRESOLVED_ufixed;
00709
00710 -- "size_res" functions create the size of the output from the indices
00711 -- of the "size_res" input. The actual value of "size_res" is not used.
00712 function resize (
00713     arg                  : UNRESOLVED_ufixed; -- input
00714     size_res              : UNRESOLVED_ufixed; -- for size only
00715     constant overflow_style : fixed_overflow_style_type := 
00716         fixed_overflow_style;
00717     constant round_style   : fixed_round_style_type   := 
00718         fixed_round_style);
00719     return UNRESOLVED_ufixed;
00720
00721 -- Note that in "wrap" mode the sign bit is not replicated. Thus the
00722 -- resize of a negative number can have a positive result in wrap mode.
00723 function resize (
00724     arg                  : UNRESOLVED_sfixed; -- input
00725     constant left_index   : INTEGER;    -- integer portion
00726     constant right_index  : INTEGER;    -- size of fraction
00727     constant overflow_style : fixed_overflow_style_type := 
00728         fixed_overflow_style;
00729     constant round_style   : fixed_round_style_type   := 
00730         fixed_round_style);
00731     return UNRESOLVED_sfixed;
00732
00733 function resize (
00734     arg                  : UNRESOLVED_sfixed; -- input
00735     size_res              : UNRESOLVED_sfixed; -- for size only
00736     constant overflow_style : fixed_overflow_style_type := 
00737         fixed_overflow_style;
00738     constant round_style   : fixed_round_style_type   := 
00739         fixed_round_style);
00740     return UNRESOLVED_sfixed;
00741
00742 =====
00743 -- Conversion Functions
00744 =====
00745
00746 -- integer (natural) to unsigned fixed point.
00747 -- arguments are the upper and lower bounds of the number, thus
00748 -- ufixed (7 downto -3) <= to_ufixed (int, 7, -3);
00749 function to_ufixed (
00750     arg                  : NATURAL; -- integer
00751     constant left_index   : INTEGER; -- left index (high index)
00752     constant right_index  : INTEGER; -- right index
00753     constant overflow_style : fixed_overflow_style_type := 
00754         fixed_overflow_style;
00755     constant round_style   : fixed_round_style_type   := 
00756         fixed_round_style);
00757     return UNRESOLVED_ufixed;
00758
00759 -- real to unsigned fixed point
00760 function to_ufixed (
00761     arg                  : REAL;      -- real
00762     constant left_index   : INTEGER; -- left index (high index)
00763     constant right_index  : INTEGER; -- right index
00764     constant overflow_style : fixed_overflow_style_type := 
00765         fixed_overflow_style;
00766     constant round_style   : fixed_round_style_type   := 
00767         fixed_round_style;
00768     constant guard_bits    : NATURAL;      -- := fixed_guard_bits)
00769     return UNRESOLVED_ufixed;
00770

```

```

00766   function to_ufixed (
00767     arg                  : REAL;      -- real
00768     size_res             : UNRESOLVED_ufixed; -- for size only
00769     constant overflow_style : fixed_overflow_style_type := 
00770       fixed_overflow_style;
00771     constant round_style  : fixed_round_style_type    := 
00772       fixed_round_style;
00773     constant guard_bits   : NATURAL                 := fixed_guard_bits)
00774   return UNRESOLVED_ufixed;
00775
00776   -- unsigned to unsigned fixed point
00777   function to_ufixed (
00778     arg                  : UNSIGNED;           -- unsigned
00779     constant left_index   : INTEGER;            -- left index (high index)
00780     constant right_index  : INTEGER             := 0; -- right index
00781     constant overflow_style : fixed_overflow_style_type := 
00782       fixed_overflow_style;
00783     constant round_style  : fixed_round_style_type    := 
00784       fixed_round_style;
00785   return UNRESOLVED_ufixed;
00786
00787   -- Performs a conversion. ufixed (arg'range) is returned
00788   function to_ufixed (
00789     arg : UNSIGNED)           -- unsigned
00790   return UNRESOLVED_ufixed;
00791
00792   -- unsigned fixed point to unsigned
00793   function to_unsigned (
00794     arg                  : UNRESOLVED_ufixed; -- fixed point input
00795     constant size         : NATURAL;            -- length of output
00796     constant overflow_style : fixed_overflow_style_type := 
00797       fixed_overflow_style;
00798     constant round_style  : fixed_round_style_type    := 
00799       fixed_round_style;
00800   return UNSIGNED;
00801
00802   -- unsigned fixed point to unsigned
00803   function to_unsigned (
00804     arg                  : UNRESOLVED_ufixed; -- fixed point input
00805     size_res             : UNSIGNED;           -- used for length of output
00806     constant overflow_style : fixed_overflow_style_type := 
00807       fixed_overflow_style;
00808     constant round_style  : fixed_round_style_type    := 
00809       fixed_round_style;
00810   return UNSIGNED;
00811
00812   -- unsigned fixed point to real
00813   function to_real (
00814     arg : UNRESOLVED_ufixed)           -- fixed point input
00815   return REAL;
00816
00817   -- unsigned fixed point to integer
00818   function to_integer (
00819     arg                  : UNRESOLVED_ufixed; -- fixed point input
00820     constant overflow_style : fixed_overflow_style_type := 
00821       fixed_overflow_style;
00822     constant round_style  : fixed_round_style_type    := 
00823       fixed_round_style;
00824   return NATURAL;
00825
00826   -- Integer to UNRESOLVED_sfixed
00827   function to_sfixed (
00828     arg                  : INTEGER;           -- integer
00829     constant left_index   : INTEGER;            -- left index (high index)
00830     constant right_index  : INTEGER             := 0; -- right index
00831     constant overflow_style : fixed_overflow_style_type := 
00832       fixed_overflow_style;
00833     constant round_style  : fixed_round_style_type    := 
00834       fixed_round_style;
00835   return UNRESOLVED_sfixed;
00836

```

```

00837      return UNRESOLVED_sfixed;
00838
00839  -- Real to sfixed
00840  function to_sfixed (
00841    arg          : REAL;      -- real
00842    constant left_index   : INTEGER;  -- left index (high index)
00843    constant right_index  : INTEGER;  -- right index
00844    constant overflow_style : fixed_overflow_style_type := 
00845      fixed_overflow_style;
00846    constant round_style   : fixed_round_style_type    :=
00847      fixed_round_style;
00848    constant guard_bits    : NATURAL                  := fixed_guard_bits)
00849  return UNRESOLVED_sfixed;
00850
00851  -- signed to sfixed
00852  function to_sfixed (
00853    arg          : REAL;      -- real
00854    size_res     : UNRESOLVED_sfixed; -- for size only
00855    constant overflow_style : fixed_overflow_style_type := 
00856      fixed_overflow_style;
00857    constant round_style   : fixed_round_style_type    :=
00858      fixed_round_style;
00859    constant guard_bits    : NATURAL                  := fixed_guard_bits)
00860  return UNRESOLVED_sfixed;
00861
00862  -- signed to sfixed
00863  function to_sfixed (
00864    arg          : SIGNED;     -- signed
00865    constant left_index   : INTEGER;  -- left index (high index)
00866    constant right_index  : INTEGER   := 0; -- right index
00867    constant overflow_style : fixed_overflow_style_type := 
00868      fixed_overflow_style;
00869    constant round_style   : fixed_round_style_type    :=
00870      fixed_round_style);
00871  return UNRESOLVED_sfixed;
00872
00873  -- signed to sfixed (output assumed to be size of signed input)
00874  function to_sfixed (
00875    arg : SIGNED)           -- signed
00876  return UNRESOLVED_sfixed;
00877
00878  -- Conversion from ufixed to sfixed
00879  function to_sfixed (
00880    arg : UNRESOLVED_ufixed)
00881  return UNRESOLVED_sfixed;
00882
00883  -- signed fixed point to signed
00884  function to_signed (
00885    arg          : UNRESOLVED_sfixed; -- fixed point input
00886    constant size        : NATURAL;    -- length of output
00887    constant overflow_style : fixed_overflow_style_type := 
00888      fixed_overflow_style;
00889    constant round_style   : fixed_round_style_type    :=
00890      fixed_round_style);
00891  return SIGNED;
00892
00893  -- signed fixed point to signed
00894  function to_signed (
00895    arg          : UNRESOLVED_sfixed; -- fixed point input
00896    size_res     : SIGNED;       -- used for length of output
00897    constant overflow_style : fixed_overflow_style_type := 
00898      fixed_overflow_style;
00899    constant round_style   : fixed_round_style_type    :=
00900      fixed_round_style);
00901  return SIGNED;
00902
00903  -- signed fixed point to real
00904  function to_real (
00905    arg : UNRESOLVED_sfixed)      -- fixed point input
00906    return REAL;
00907
00908  -- signed fixed point to integer
00909  function to_integer (
00910    arg          : UNRESOLVED_sfixed; -- fixed point input
00911    constant overflow_style : fixed_overflow_style_type := 
00912      fixed_overflow_style;
00913    constant round_style   : fixed_round_style_type    :=
00914      fixed_round_style);
00915  return INTEGER;

```

```

00910
00911  -- Because of the fairly complicated sizing rules in the fixed point
00912  -- packages these functions are provided to compute the result ranges
00913  -- Example:
00914  -- signal ufl1 : ufixed (3 downto -3);
00915  -- signal uf2 : ufixed (4 downto -2);
00916  -- signal uflmultuf2 : ufixed (ufixed_high (3, -3, '*', 4, -2) downto
00917  --                                ufixed_low (3, -3, '*', 4, -2));
00918  -- uflmultuf2 <= ufl1 * uf2;
00919  -- Valid characters: '+', '-', '*', '/', 'r' or 'R' (rem), 'm' or 'M' (mod),
00920  -- '1' (reciprocal), 'a' or 'A' (abs), 'n' or 'N' (unary -)
00921  function ufixed_high (left_index, right_index : INTEGER;
00922          operation : CHARACTER := 'X';
00923          left_index2, right_index2 : INTEGER := 0)
00924    return INTEGER;
00925
00926  function ufixed_low (left_index, right_index : INTEGER;
00927          operation : CHARACTER := 'X';
00928          left_index2, right_index2 : INTEGER := 0)
00929    return INTEGER;
00930
00931  function sfixed_high (left_index, right_index : INTEGER;
00932          operation : CHARACTER := 'X';
00933          left_index2, right_index2 : INTEGER := 0)
00934    return INTEGER;
00935
00936  function sfixed_low (left_index, right_index : INTEGER;
00937          operation : CHARACTER := 'X';
00938          left_index2, right_index2 : INTEGER := 0)
00939    return INTEGER;
00940
00941  -- Same as above, but using the "size_res" input only for their ranges:
00942  -- signal uflmultuf2 : ufixed (ufixed_high (ufl1, '*', uf2) downto
00943  --                                ufixed_low (ufl1, '*', uf2));
00944  -- uflmultuf2 <= ufl1 * uf2;
00945  --
00946  function ufixed_high (size_res : UNRESOLVED_ufixed;
00947          operation : CHARACTER := 'X';
00948          size_res2 : UNRESOLVED_ufixed)
00949    return INTEGER;
00950
00951  function ufixed_low (size_res : UNRESOLVED_ufixed;
00952          operation : CHARACTER := 'X';
00953          size_res2 : UNRESOLVED_ufixed)
00954    return INTEGER;
00955
00956  function sfixed_high (size_res : UNRESOLVED_sfixed;
00957          operation : CHARACTER := 'X';
00958          size_res2 : UNRESOLVED_sfixed)
00959    return INTEGER;
00960
00961  function sfixed_low (size_res : UNRESOLVED_sfixed;
00962          operation : CHARACTER := 'X';
00963          size_res2 : UNRESOLVED_sfixed)
00964    return INTEGER;
00965
00966  -- purpose: returns a saturated number
00967  function saturate (
00968      constant left_index : INTEGER;
00969      constant right_index : INTEGER)
00970    return UNRESOLVED_ufixed;
00971
00972  -- purpose: returns a saturated number
00973  function saturate (
00974      constant left_index : INTEGER;
00975      constant right_index : INTEGER)
00976    return UNRESOLVED_sfixed;
00977
00978  function saturate (
00979      size_res : UNRESOLVED_ufixed)           -- only the size of this is used
00980    return UNRESOLVED_ufixed;
00981
00982  function saturate (
00983      size_res : UNRESOLVED_sfixed)           -- only the size of this is used
00984    return UNRESOLVED_sfixed;
00985
00986 =====
00987  -- Translation Functions
00988 =====
00989
00990  -- maps meta-logical values
00991  function to_01 (
00992      s : UNRESOLVED_ufixed; -- fixed point input
00993      constant XMAP : STD_ULOGIC := '0') -- Map x to
00994    return UNRESOLVED_ufixed;
00995
00996  -- maps meta-logical values

```

```

0097  function to_01 (
0098    s           : UNRESOLVED_sfixed; -- fixed point input
0099    constant XMAP : STD_ULOGIC := '0') -- Map x to
0100    return UNRESOLVED_sfixed;
0101
0102  function Is_X   (arg : UNRESOLVED_ufixed) return BOOLEAN;
0103  function Is_X   (arg : UNRESOLVED_sfixed) return BOOLEAN;
0104  function to_X01 (arg : UNRESOLVED_ufixed) return
0105    UNRESOLVED_ufixed;
0106  function to_X01 (arg : UNRESOLVED_sfixed) return
0107    UNRESOLVED_ufixed;
0108  function to_X01Z (arg : UNRESOLVED_ufixed) return
0109    UNRESOLVED_sfixed;
0110  function to_X01Z (arg : UNRESOLVED_sfixed) return
0111    UNRESOLVED_ufixed;
0112  function to_UX01 (arg : UNRESOLVED_ufixed) return
0113    UNRESOLVED_ufixed;
0114  function to_UX01 (arg : UNRESOLVED_sfixed) return
0115    UNRESOLVED_sfixed;
0116
0117  -- straight vector conversion routines, needed for synthesis.
0118  -- These functions are here so that a std_logic_vector can be
0119  -- converted to and from sfixed and ufixed. Note that you can
0120  -- not convert these vectors because of their negative index.
0121
0122  function to_slv (
0123    arg : UNRESOLVED_ufixed)          -- fixed point vector
0124    return STD_LOGIC_VECTOR;
0125  alias to_StdLogicVector is to_slv [UNRESOLVED_ufixed
0126                                         return STD_LOGIC_VECTOR];
0127  alias to_Std_Logic_Vector is to_slv [UNRESOLVED_ufixed
0128                                         return STD_LOGIC_VECTOR];
0129  alias to_StdLogicVector is to_slv [UNRESOLVED_sfixed
0130                                         return STD_LOGIC_VECTOR];
0131
0132  function to_slv (
0133    arg : UNRESOLVED_sfixed)          -- fixed point vector
0134    return STD_LOGIC_VECTOR;
0135  alias to_StdULogicVector is to_slv [UNRESOLVED_ufixed
0136                                         return STD_ULOGIC_VECTOR];
0137  alias to_Std_ULogic_Vector is to_slv [UNRESOLVED_ufixed
0138                                         return STD_ULOGIC_VECTOR];
0139
0140  function to_sluv (
0141    arg : UNRESOLVED_sfixed)          -- fixed point vector
0142    return STD_ULOGIC_VECTOR;
0143  alias to_StdULogicVector is to_sluv [UNRESOLVED_sfixed
0144                                         return STD_ULOGIC_VECTOR];
0145  alias to_Std_ULogic_Vector is to_sluv [UNRESOLVED_sfixed
0146                                         return STD_ULOGIC_VECTOR];
0147
0148  function to_ufixed (
0149    arg           : STD_ULOGIC_VECTOR; -- shifted vector
0150    constant left_index : INTEGER;
0151    constant right_index : INTEGER)
0152    return UNRESOLVED_ufixed;
0153
0154  function to_ufixed (
0155    arg           : STD_ULOGIC_VECTOR; -- shifted vector
0156    size_res : UNRESOLVED_ufixed)      -- for size only
0157    return UNRESOLVED_ufixed;
0158
0159  function to_sfixed (
0160    arg           : STD_ULOGIC_VECTOR; -- shifted vector
0161    constant left_index : INTEGER;
0162    constant right_index : INTEGER)
0163    return UNRESOLVED_sfixed;
0164
0165  function to_sfixed (
0166    arg           : STD_ULOGIC_VECTOR; -- shifted vector
0167    size_res : UNRESOLVED_sfixed)      -- for size only
0168    return UNRESOLVED_sfixed;
0169
0170  -- As a concession to those who use a graphical DSP environment,
0171  -- these functions take parameters in those tools format and create
0172  -- fixed point numbers. These functions are designed to convert from
0173  -- a std_logic_vector to the VHDL fixed point format using the conventions
0174  -- of these packages. In a pure VHDL environment you should use the
0175  -- "to_ufixed" and "to_sfixed" routines.
0176
0177  -- unsigned fixed point

```

```

01078  function to_UFix (
01079      arg      : STD_ULONGIC_VECTOR;
01080      width    : NATURAL;                      -- width of vector
01081      fraction : NATURAL)                   -- width of fraction
01082      return UNRESOLVED_ufixed;
01083
01084  -- signed fixed point
01085  function to_SFix (
01086      arg      : STD_ULONGIC_VECTOR;
01087      width    : NATURAL;                      -- width of vector
01088      fraction : NATURAL)                   -- width of fraction
01089      return UNRESOLVED_sfixed;
01090
01091  -- finding the bounds of a number. These functions can be used like this:
01092  -- signal xxx : ufixed (7 downto -3);
01093  -- -- Which is the same as "ufixed (UFix_high (11,3) downto UFix_low(11,3))"
01094  -- signalyyy : ufixed (UFix_high (11, 3, "+", 11, 3)
01095  --           downto UFix_low(11, 3, "+", 11, 3));
01096  -- Where "11" is the width of xxx (xxx'length),
01097  -- and 3 is the lower bound (abs (xxx'low))
01098  -- In a pure VHDL environment use "ufixed_high" and "ufixed_low"
01099
01100 function UFix_high (width, fraction  : NATURAL;
01101          operation     : CHARACTER := 'X';
01102          width2, fraction2 : NATURAL := 0)
01103      return INTEGER;
01104
01105 function UFix_low (width, fraction   : NATURAL;
01106          operation     : CHARACTER := 'X';
01107          width2, fraction2 : NATURAL := 0)
01108      return INTEGER;
01109
01110  -- Same as above but for signed fixed point. Note that the width
01111  -- of a signed fixed point number ignores the sign bit, thus
01112  -- width = sxxx'length-1
01113
01114 function SFix_high (width, fraction  : NATURAL;
01115          operation     : CHARACTER := 'X';
01116          width2, fraction2 : NATURAL := 0)
01117      return INTEGER;
01118
01119 function SFix_low (width, fraction   : NATURAL;
01120          operation     : CHARACTER := 'X';
01121          width2, fraction2 : NATURAL := 0)
01122      return INTEGER;
01123 -- rtl_synthesis off
01124 -- pragma synthesis_off
01125 =====
01126 -- string and textio Functions
01127 =====
01128
01129  -- purpose: writes fixed point into a line
01130  procedure WRITE (
01131      L      : inout LINE;                  -- input line
01132      VALUE : in  UNRESOLVED_ufixed;    -- fixed point input
01133      JUSTIFIED : in  SIDE := right;
01134      FIELD  : in  WIDTH := 0);
01135
01136  -- purpose: writes fixed point into a line
01137  procedure WRITE (
01138      L      : inout LINE;                  -- input line
01139      VALUE : in  UNRESOLVED_sfixed;    -- fixed point input
01140      JUSTIFIED : in  SIDE := right;
01141      FIELD  : in  WIDTH := 0);
01142
01143  procedure READ(L : inout LINE;
01144                  VALUE : out UNRESOLVED_ufixed);
01145
01146  procedure READ(L : inout LINE;
01147                  VALUE : out UNRESOLVED_ufixed;
01148                  GOOD  : out BOOLEAN);
01149
01150  procedure READ(L : inout LINE;
01151                  VALUE : out UNRESOLVED_sfixed);
01152
01153  procedure READ(L : inout LINE;
01154                  VALUE : out UNRESOLVED_sfixed;
01155                  GOOD  : out BOOLEAN);
01156
01157  alias bwrite is WRITE [LINE, UNRESOLVED_ufixed, SIDE, width];
01158  alias bwrite is WRITE [LINE, UNRESOLVED_sfixed, SIDE, width];
01159  alias bread is READ [LINE, UNRESOLVED_ufixed];
01160  alias bread is READ [LINE, UNRESOLVED_ufixed, BOOLEAN];
01161  alias bread is READ [LINE, UNRESOLVED_sfixed];
01162  alias bread is READ [LINE, UNRESOLVED_sfixed, BOOLEAN];
01163  alias BINARY_WRITE is WRITE [LINE, UNRESOLVED_ufixed, SIDE, width];
01164  alias BINARY_WRITE is WRITE [LINE, UNRESOLVED_sfixed, SIDE, width];

```

```

01165 alias BINARY_READ is READ [LINE, UNRESOLVED_ufixed, BOOLEAN];
01166 alias BINARY_READ is READ [LINE, UNRESOLVED_ufixed];
01167 alias BINARY_READ is READ [LINE, UNRESOLVED_sfixed, BOOLEAN];
01168 alias BINARY_READ is READ [LINE, UNRESOLVED_sfixed];
01169
01170 -- octal read and write
01171 procedure OWRITE (
01172     L      : inout LINE;           -- input line
01173     VALUE   : in  UNRESOLVED_ufixed; -- fixed point input
01174     JUSTIFIED : in  SIDE := right;
01175     FIELD    : in  WIDTH := 0);
01176
01177 procedure OWRITE (
01178     L      : inout LINE;           -- input line
01179     VALUE   : in  UNRESOLVED_sfixed; -- fixed point input
01180     JUSTIFIED : in  SIDE := right;
01181     FIELD    : in  WIDTH := 0);
01182
01183 procedure OREAD(L   : inout LINE;
01184                   VALUE : out UNRESOLVED_ufixed);
01185
01186 procedure OREAD(L   : inout LINE;
01187                   VALUE : out UNRESOLVED_ufixed;
01188                   GOOD  : out BOOLEAN);
01189
01190 procedure OREAD(L   : inout LINE;
01191                   VALUE : out UNRESOLVED_sfixed);
01192
01193 procedure OREAD(L   : inout LINE;
01194                   VALUE : out UNRESOLVED_sfixed;
01195                   GOOD  : out BOOLEAN);
01196 alias OCTAL_READ is OREAD [LINE, UNRESOLVED_ufixed, BOOLEAN];
01197 alias OCTAL_READ is OREAD [LINE, UNRESOLVED_ufixed];
01198 alias OCTAL_READ is OREAD [LINE, UNRESOLVED_sfixed, BOOLEAN];
01199 alias OCTAL_READ is OREAD [LINE, UNRESOLVED_sfixed];
01200 alias OCTAL_WRITE is OWRITE [LINE, UNRESOLVED_ufixed, SIDE, WIDTH];
01201 alias OCTAL_WRITE is OWRITE [LINE, UNRESOLVED_sfixed, SIDE, WIDTH];
01202
01203 -- hex read and write
01204 procedure HWRITE (
01205     L      : inout LINE;           -- input line
01206     VALUE   : in  UNRESOLVED_ufixed; -- fixed point input
01207     JUSTIFIED : in  SIDE := right;
01208     FIELD    : in  WIDTH := 0);
01209
01210 -- purpose: writes fixed point into a line
01211 procedure HWRITE (
01212     L      : inout LINE;           -- input line
01213     VALUE   : in  UNRESOLVED_sfixed; -- fixed point input
01214     JUSTIFIED : in  SIDE := right;
01215     FIELD    : in  WIDTH := 0);
01216
01217 procedure HREAD(L   : inout LINE;
01218                   VALUE : out UNRESOLVED_ufixed);
01219
01220 procedure HREAD(L   : inout LINE;
01221                   VALUE : out UNRESOLVED_ufixed;
01222                   GOOD  : out BOOLEAN);
01223
01224 procedure HREAD(L   : inout LINE;
01225                   VALUE : out UNRESOLVED_sfixed);
01226
01227 procedure HREAD(L   : inout LINE;
01228                   VALUE : out UNRESOLVED_sfixed;
01229                   GOOD  : out BOOLEAN);
01230 alias HEX_READ is HREAD [LINE, UNRESOLVED_ufixed, BOOLEAN];
01231 alias HEX_READ is HREAD [LINE, UNRESOLVED_sfixed, BOOLEAN];
01232 alias HEX_READ is HREAD [LINE, UNRESOLVED_ufixed];
01233 alias HEX_READ is HREAD [LINE, UNRESOLVED_sfixed];
01234 alias HEX_WRITE is HWRITE [LINE, UNRESOLVED_ufixed, SIDE, WIDTH];
01235 alias HEX_WRITE is HWRITE [LINE, UNRESOLVED_sfixed, SIDE, WIDTH];
01236
01237 -- returns a string, useful for:
01238 -- assert (x = y) report "error found " & to_string(x) severity error;
01239 function to_string (value : UNRESOLVED_ufixed) return STRING;
01240 alias to_bstring is to_string [UNRESOLVED_ufixed return STRING];
01241 alias TO_BINARY_STRING is TO_STRING [UNRESOLVED_ufixed return STRING];
01242
01243 function to_ostring (value : UNRESOLVED_ufixed) return STRING;
01244 alias TO_OCTAL_STRING is TO_OSTRING [UNRESOLVED_ufixed return STRING];
01245
01246 function to_hstring (value : UNRESOLVED_ufixed) return STRING;
01247 alias TO_HEX_STRING is TO_HSTRING [UNRESOLVED_ufixed return STRING];
01248
01249 function to_string (value : UNRESOLVED_sfixed) return STRING;
01250 alias to_bstring is to_string [UNRESOLVED_sfixed return STRING];
01251 alias TO_BINARY_STRING is TO_STRING [UNRESOLVED_sfixed return STRING];

```

```

01252
01253     function to_ostring (value : UNRESOLVED_sfixed) return STRING;
01254     alias TO_OCTAL_STRING is TO_OSTRING [UNRESOLVED_sfixed return STRING];
01255
01256     function to_hstring (value : UNRESOLVED_sfixed) return STRING;
01257     alias TO_HEX_STRING is TO_HSTRING [UNRESOLVED_sfixed return STRING];
01258
01259     -- From string functions allow you to convert a string into a fixed
01260     -- point number. Example:
01261     -- signal ufl : ufixed (3 downto -3);
01262     -- ufl <= from_string ("0110.100", ufl'high, ufl'low); -- 6.5
01263     -- The "." is optional in this syntax, however it exist and is
01264     -- in the wrong location an error is produced. Overflow will
01265     -- result in saturation.
01266
01267     function from_string (
01268         bstring          : STRING;      -- binary string
01269         constant left_index : INTEGER;
01270         constant right_index : INTEGER)
01271     return UNRESOLVED_ufixed;
01272     alias from_bstring is from_string [STRING, INTEGER, INTEGER
01273                                         return UNRESOLVED_ufixed];
01274     alias from_binary_string is from_string [STRING, INTEGER, INTEGER
01275                                         return UNRESOLVED_ufixed];
01276
01277     -- Octal and hex conversions work as follows:
01278     -- ufl <= from_hstring ("6.8", 3, -3); -- 6.5 (bottom zeros dropped)
01279     -- ufl <= from_ostring ("06.4", 3, -3); -- 6.5 (top zeros dropped)
01280
01281     function from_ostring (
01282         ostring          : STRING;      -- Octal string
01283         constant left_index : INTEGER;
01284         constant right_index : INTEGER)
01285     return UNRESOLVED_ufixed;
01286     alias from_octal_string is from_ostring [STRING, INTEGER, INTEGER
01287                                         return UNRESOLVED_ufixed];
01288
01289     function from_hstring (
01290         hstring          : STRING;      -- hex string
01291         constant left_index : INTEGER;
01292         constant right_index : INTEGER)
01293     return UNRESOLVED_ufixed;
01294     alias from_hex_string is from_hstring [STRING, INTEGER, INTEGER
01295                                         return UNRESOLVED_ufixed];
01296
01297     function from_string (
01298         bstring          : STRING;      -- binary string
01299         constant left_index : INTEGER;
01300         constant right_index : INTEGER)
01301     return UNRESOLVED_sfixed;
01302     alias from_bstring is from_string [STRING, INTEGER, INTEGER
01303                                         return UNRESOLVED_sfixed];
01304     alias from_binary_string is from_string [STRING, INTEGER, INTEGER
01305                                         return UNRESOLVED_sfixed];
01306
01307     function from_ostring (
01308         ostring          : STRING;      -- Octal string
01309         constant left_index : INTEGER;
01310         constant right_index : INTEGER)
01311     return UNRESOLVED_sfixed;
01312     alias from_octal_string is from_ostring [STRING, INTEGER, INTEGER
01313                                         return UNRESOLVED_sfixed];
01314
01315     function from_hstring (
01316         hstring          : STRING;      -- hex string
01317         constant left_index : INTEGER;
01318         constant right_index : INTEGER)
01319     return UNRESOLVED_sfixed;
01320     alias from_hex_string is from_hstring [STRING, INTEGER, INTEGER
01321                                         return UNRESOLVED_sfixed];
01322
01323     -- Same as above, "size_res" is used for it's range only.
01324     function from_string (
01325         bstring          : STRING;      -- binary string
01326         size_res : UNRESOLVED_ufixed)
01327     return UNRESOLVED_ufixed;
01328     alias from_bstring is from_string [STRING, UNRESOLVED_ufixed
01329                                         return UNRESOLVED_ufixed];
01330     alias from_binary_string is from_string [STRING,
01331                                         UNRESOLVED_ufixed
01332                                         return UNRESOLVED_ufixed];
01333
01334     function from_ostring (
01335         ostring          : STRING;      -- Octal string
01336         size_res : UNRESOLVED_ufixed)
01337     return UNRESOLVED_ufixed;
01338     alias from_octal_string is from_ostring [STRING,

```

```

UNRESOLVED_ufixed
01338                                     return UNRESOLVED_ufixed];
01339
01340     function from_hstring (
01341         hstring : STRING;                      -- hex string
01342         size_res : UNRESOLVED_ufixed)
01343         return UNRESOLVED_ufixed;
01344     alias from_hex_string is from_hstring [STRING,
UNRESOLVED_ufixed
01345                                     return UNRESOLVED_ufixed];
01346
01347     function from_string (
01348         bstring : STRING;                      -- binary string
01349         size_res : UNRESOLVED_sfixed)
01350         return UNRESOLVED_sfixed;
01351     alias from_bstring is from_string [STRING, UNRESOLVED_sfixed
01352                                     return UNRESOLVED_sfixed];
01353     alias from_binary_string is from_string [STRING,
UNRESOLVED_sfixed
01354                                     return UNRESOLVED_sfixed];
01355
01356     function from_ostring (
01357         ostring : STRING;                      -- Octal string
01358         size_res : UNRESOLVED_sfixed)
01359         return UNRESOLVED_sfixed;
01360     alias from_octal_string is from_ostring [STRING,
UNRESOLVED_sfixed
01361                                     return UNRESOLVED_sfixed];
01362
01363     function from_hstring (
01364         hstring : STRING;                      -- hex string
01365         size_res : UNRESOLVED_sfixed)
01366         return UNRESOLVED_sfixed;
01367     alias from_hex_string is from_hstring [STRING,
UNRESOLVED_sfixed
01368                                     return UNRESOLVED_sfixed];
01369
01370 -- Direct conversion functions. Example:
01371 -- signal ufl : ufixed (3 downto -3);
01372 -- ufl <= from_string ("0110.100"); -- 6.5
01373 -- In this case the "." is not optional, and the size of
01374 -- the output must match exactly.
01375
01376     function from_string (
01377         bstring : STRING)                      -- binary string
01378         return UNRESOLVED_ufixed;
01379     alias from_bstring is from_string [STRING return UNRESOLVED_ufixed];
01380     alias from_binary_string is from_string [STRING return
UNRESOLVED_ufixed];
01381
01382 -- Direct octal and hex conversion functions. In this case
01383 -- the string lengths must match. Example:
01384 -- signal sfl := sfixed (5 downto -3);
01385 -- sfl <= from_ostring ("71.4") -- -6.5
01386
01387     function from_ostring (
01388         ostring : STRING)                      -- Octal string
01389         return UNRESOLVED_ufixed;
01390     alias from_octal_string is from_ostring [STRING return
UNRESOLVED_ufixed];
01391
01392     function from_hstring (
01393         hstring : STRING)                      -- hex string
01394         return UNRESOLVED_ufixed;
01395     alias from_hex_string is from_hstring [STRING return
UNRESOLVED_ufixed];
01396
01397     function from_string (
01398         bstring : STRING)                      -- binary string
01399         return UNRESOLVED_sfixed;
01400     alias from_bstring is from_string [STRING return UNRESOLVED_sfixed];
01401     alias from_binary_string is from_string [STRING return
UNRESOLVED_sfixed];
01402
01403     function from_ostring (
01404         ostring : STRING)                      -- Octal string
01405         return UNRESOLVED_sfixed;
01406     alias from_octal_string is from_ostring [STRING return
UNRESOLVED_sfixed];
01407
01408     function from_hstring (
01409         hstring : STRING)                      -- hex string
01410         return UNRESOLVED_sfixed;
01411     alias from_hex_string is from_hstring [STRING return
UNRESOLVED_sfixed];
01412 -- rtl_synthesis on
01413 -- pragma synthesis_on

```

```

01414
01415  -- IN VHDL-2006 std_logic_vector is a subtype of std_ulegic_vector, so these
01416  -- extra functions are needed for compatibility.
01417  function to_ufixed (
01418      arg          : STD_LOGIC_VECTOR;  -- shifted vector
01419      constant left_index  : INTEGER;
01420      constant right_index : INTEGER)
01421  return UNRESOLVED_ufixed;
01422
01423  function to_ufixed (
01424      arg          : STD_LOGIC_VECTOR;  -- shifted vector
01425      size_res : UNRESOLVED_ufixed)    -- for size only
01426  return UNRESOLVED_ufixed;
01427
01428  function to_sfixed (
01429      arg          : STD_LOGIC_VECTOR;  -- shifted vector
01430      constant left_index  : INTEGER;
01431      constant right_index : INTEGER)
01432  return UNRESOLVED_sfixed;
01433
01434  function to_sfixed (
01435      arg          : STD_LOGIC_VECTOR;  -- shifted vector
01436      size_res : UNRESOLVED_sfixed)    -- for size only
01437  return UNRESOLVED_sfixed;
01438
01439  -- unsigned fixed point
01440  function to_UFix (
01441      arg          : STD_LOGIC_VECTOR;
01442      width       : NATURAL;           -- width of vector
01443      fraction   : NATURAL)          -- width of fraction
01444  return UNRESOLVED_ufixed;
01445
01446  -- signed fixed point
01447  function to_SFfix (
01448      arg          : STD_LOGIC_VECTOR;
01449      width       : NATURAL;           -- width of vector
01450      fraction   : NATURAL)          -- width of fraction
01451  return UNRESOLVED_sfixed;
01452
01453 end package fixed_pkg;
01454 -----
01455 -- Proposed package body for the VHDL-200x-FT fixed_pkg package
01456 -- (Fixed point math package)
01457 -- This package body supplies a recommended implementation of these functions
01458 -- Version     : $Revision: 1.22 $
01459 -- Date        : $Date: 2010/09/22 18:34:14 $
01460 --
01461 -- Created for VHDL-200X-ft, David Bishop (dbishop@vhdl.org)
01462 -----
01463 library IEEE;
01464 use IEEE.MATH_REAL.all;
01465
01466 package body fixed_pkg is
01467  -- Author David Bishop (dbishop@vhdl.org)
01468  -- Other contributers: Jim Lewis, Yannick Grugni, Ryan W. Hilton
01469  -- null array constants
01470  constant NAUF : UNRESOLVED_ufixed (0 downto 1) := (others => '0');
01471  constant NASF : UNRESOLVED_sfixed (0 downto 1) := (others => '0');
01472  constant NSLV : STD_ULOGIC_VECTOR (0 downto 1) := (others => '0');
01473
01474  -- This differed constant will tell you if the package body is synthesizable
01475  -- or implemented as real numbers, set to "true" if synthesizable.
01476  constant fixedsynth_or_real : BOOLEAN := true;
01477
01478  -- %% Replicated functions
01479  function maximum (
01480      l, r : integer)                      -- inputs
01481  return integer is
01482  begin  -- function max
01483      if l > r then return l;
01484      else return r;
01485      end if;
01486  end function maximum;
01487
01488  function minimum (
01489      l, r : integer)                      -- inputs
01490  return integer is
01491  begin  -- function min
01492      if l > r then return r;
01493      else return l;
01494      end if;
01495  end function minimum;
01496
01497  function "sra" (arg : SIGNED; count : INTEGER)
01498  return SIGNED is
01499  begin
01500      if (COUNT >= 0) then

```

```

01501      return SHIFT_RIGHT(arg, count);
01502      else
01503          return SHIFT_LEFT(arg, -count);
01504      end if;
01505  end function "sra";
01506
01507  function or_reduce (arg : STD_ULOGIC_VECTOR)
01508      return STD_LOGIC is
01509      variable Upper, Lower : STD_ULOGIC;
01510      variable Half        : INTEGER;
01511      variable BUS_int     : STD_ULOGIC_VECTOR (arg'length - 1 downto 0);
01512      variable Result       : STD_ULOGIC;
01513  begin
01514      if (arg'length < 1) then           -- In the case of a NULL range
01515          Result := '0';
01516      else
01517          BUS_int := to_ux01 (arg);
01518          if (BUS_int'length = 1) then
01519              Result := BUS_int (BUS_int'left);
01520          elsif (BUS_int'length = 2) then
01521              Result := BUS_int (BUS_int'right) or BUS_int (BUS_int'left);
01522          else
01523              Half  := (BUS_int'length + 1) / 2 + BUS_int'right;
01524              Upper := or_reduce (BUS_int (BUS_int'left downto Half));
01525              Lower := or_reduce (BUS_int (Half - 1 downto BUS_int'right));
01526              Result := Upper or Lower;
01527          end if;
01528      end if;
01529      return Result;
01530  end function or_reduce;
01531
01532  -- purpose: AND all of the bits in a vector together
01533  -- This is a copy of the proposed "and_reduce" from 1076.3
01534  function and_reduce (arg : STD_ULOGIC_VECTOR)
01535      return STD_LOGIC is
01536      variable Upper, Lower : STD_ULOGIC;
01537      variable Half        : INTEGER;
01538      variable BUS_int     : STD_ULOGIC_VECTOR (arg'length - 1 downto 0);
01539      variable Result       : STD_ULOGIC;
01540  begin
01541      if (arg'length < 1) then           -- In the case of a NULL range
01542          Result := '1';
01543      else
01544          BUS_int := to_ux01 (arg);
01545          if (BUS_int'length = 1) then
01546              Result := BUS_int (BUS_int'left);
01547          elsif (BUS_int'length = 2) then
01548              Result := BUS_int (BUS_int'right) and BUS_int (BUS_int'left);
01549          else
01550              Half  := (BUS_int'length + 1) / 2 + BUS_int'right;
01551              Upper := and_reduce (BUS_int (BUS_int'left downto Half));
01552              Lower := and_reduce (BUS_int (Half - 1 downto BUS_int'right));
01553              Result := Upper and Lower;
01554          end if;
01555      end if;
01556      return Result;
01557  end function and_reduce;
01558
01559  function xor_reduce (arg : STD_ULOGIC_VECTOR) return STD_ULOGIC is
01560      variable Upper, Lower : STD_ULOGIC;
01561      variable Half        : INTEGER;
01562      variable BUS_int     : STD_ULOGIC_VECTOR (arg'length - 1 downto 0);
01563      variable Result       : STD_ULOGIC := '0'; -- In the case of a NULL range
01564  begin
01565      if (arg'length >= 1) then
01566          BUS_int := to_ux01 (arg);
01567          if (BUS_int'length = 1) then
01568              Result := BUS_int (BUS_int'left);
01569          elsif (BUS_int'length = 2) then
01570              Result := BUS_int(BUS_int'right) xor BUS_int(BUS_int'left);
01571          else
01572              Half  := (BUS_int'length + 1) / 2 + BUS_int'right;
01573              Upper := xor_reduce (BUS_int (BUS_int'left downto Half));
01574              Lower := xor_reduce (BUS_int (Half - 1 downto BUS_int'right));
01575              Result := Upper xor Lower;
01576          end if;
01577      end if;
01578      return Result;
01579  end function xor_reduce;
01580
01581  function nand_reduce(arg : std_ulogic_vector) return STD_ULOGIC is
01582  begin
01583      return not and_reduce (arg);
01584  end function nand_reduce;
01585  function nor_reduce(arg : std_ulogic_vector) return STD_ULOGIC is
01586  begin
01587      return not or_reduce (arg);

```

```

01588 end function nor_reduce;
01589 function xnor_reduce(arg : std_ulogic_vector) return STD_ULOGIC is
01590 begin
01591     return not xor_reduce (arg);
01592 end function xnor_reduce;
01593 -- Match table, copied form new std_logic_1164
01594 type stdlogic_table is array(STD_ULOGIC, STD_ULOGIC) of STD_ULOGIC;
01595 constant match_logic_table : stdlogic_table := (
01596     -----
01597     -- U   X   0   1   Z   W   L   H   -   |   |
01598     -----
01599     ('U', 'U', 'U', 'U', 'U', 'U', 'U', 'U', '1'),  -- | U |
01600     ('U', 'X', 'X', 'X', 'X', 'X', 'X', 'X', '1'),  -- | X |
01601     ('U', 'X', '1', '0', 'X', 'X', '1', '0', '1'),  -- | 0 |
01602     ('U', 'X', '0', '1', 'X', 'X', '0', '1', '1'),  -- | 1 |
01603     ('U', 'X', 'X', 'X', 'X', 'X', 'X', 'X', '1'),  -- | Z |
01604     ('U', 'X', 'X', 'X', 'X', 'X', 'X', 'X', '1'),  -- | W |
01605     ('U', 'X', '1', '0', 'X', 'X', '1', '0', '1'),  -- | L |
01606     ('U', 'X', '0', '1', 'X', 'X', '0', '1', '1'),  -- | H |
01607     ('1', '1', '1', '1', '1', '1', '1', '1', '1')  -- | - |
01608 );
01609 -----
01610 -----?
01611 -- ?= functions, Similar to "std_match", but returns "std_ulogic".
01612 -----
01613 function \?=\\(l, r : STD_ULOGIC) return STD_ULOGIC is
01614 begin
01615     return match_logic_table (l, r);
01616 end function \?=?;
01617 function \?=/\\(l, r : STD_ULOGIC) return STD_ULOGIC is
01618 begin
01619     return not match_logic_table (l, r);
01620 end function \?=/?;
01621 -- "?=" operator is similar to "std_match", but returns a std_ulogic..
01622 -- Id: M.2B
01623 function \?=\\(L, R: UNSIGNED) return STD_ULOGIC is
01624     constant L_LEFT : INTEGER := L'LENGTH-1;
01625     constant R_LEFT : INTEGER := R'LENGTH-1;
01626     alias XL      : UNSIGNED(L_LEFT downto 0) is L;
01627     alias XR      : UNSIGNED(R_LEFT downto 0) is R;
01628     constant SIZE   : NATURAL := MAXIMUM(L'LENGTH, R'LENGTH);
01629     variable LX      : UNSIGNED(SIZE-1 downto 0);
01630     variable RX      : UNSIGNED(SIZE-1 downto 0);
01631     variable result, result1 : STD_ULOGIC;           -- result
01632 begin
01633     -- Logically identical to an "=" operator.
01634     if ((L'LENGTH < 1) or (R'LENGTH < 1)) then
01635         assert NO_WARNING
01636             report "NUMERIC_STD.""?=""": null detected, returning X"
01637             severity warning;
01638         return 'X';
01639     else
01640         LX := RESIZE(XL, SIZE);
01641         RX := RESIZE(XR, SIZE);
01642         result := '1';
01643         for i in LX'low to LX'high loop
01644             result1 := \?=\\(LX(i), RX(i));
01645             if result1 = 'U' then
01646                 return 'U';
01647             elsif result1 = 'X' or result = 'X' then
01648                 result := 'X';
01649             else
01650                 result := result and result1;
01651             end if;
01652         end loop;
01653         return result;
01654     end if;
01655 end function \?=\\;
01656 --
01657 -- Id: M.3B
01658 function \?=\\(L, R: SIGNED) return std_ulogic is
01659     constant L_LEFT : INTEGER := L'LENGTH-1;
01660     constant R_LEFT : INTEGER := R'LENGTH-1;
01661     alias XL      : SIGNED(L_LEFT downto 0) is L;
01662     alias XR      : SIGNED(R_LEFT downto 0) is R;
01663     constant SIZE   : NATURAL := MAXIMUM(L'LENGTH, R'LENGTH);
01664     variable LX      : SIGNED(SIZE-1 downto 0);
01665     variable RX      : SIGNED(SIZE-1 downto 0);
01666     variable result, result1 : STD_ULOGIC;           -- result
01667 begin
01668     if ((L'LENGTH < 1) or (R'LENGTH < 1)) then
01669         assert NO_WARNING
01670             report "NUMERIC_STD.""?=""": null detected, returning X"
01671             severity warning;
01672         return 'X';
01673     else
01674         LX := RESIZE(XL, SIZE);

```

```

01675      RX := RESIZE(XR, SIZE);
01676      result := '1';
01677      for i in LX'low to LX'high loop
01678          resultl := \?=\ (LX(i), RX(i));
01679          if resultl = 'U' then
01680              return 'U';
01681          elsif resultl = 'X' or result = 'X' then
01682              result := 'X';
01683          else
01684              result := result and resultl;
01685          end if;
01686      end loop;
01687      return result;
01688  end if;
01689 end function \?=\;

01690 function \?=/\ (L, R : UNSIGNED) return std_ulogic is
01691  constant L_LEFT : INTEGER := L'LENGTH-1;
01692  constant R_LEFT : INTEGER := R'LENGTH-1;
01693  alias XL : UNSIGNED(L_LEFT downto 0) is L;
01694  alias XR : UNSIGNED(R_LEFT downto 0) is R;
01695  constant SIZE : NATURAL := MAXIMUM(L'LENGTH, R'LENGTH);
01696  variable LX : UNSIGNED(SIZE-1 downto 0);
01697  variable RX : UNSIGNED(SIZE-1 downto 0);
01698  variable result, resultl : STD_ULOGIC; -- result
01699 begin
01700      -- ?=
01701      if ((L'LENGTH < 1) or (R'LENGTH < 1)) then
01702          assert NO_WARNING
01703              report "NUMERIC_STD.\"?=/\": null detected, returning X"
01704              severity warning;
01705          return 'X';
01706      else
01707          LX := RESIZE(XL, SIZE);
01708          RX := RESIZE(XR, SIZE);
01709          result := '0';
01710          for i in LX'low to LX'high loop
01711              resultl := \?=/\ (LX(i), RX(i));
01712              if resultl = 'U' then
01713                  result := 'U';
01714              elsif resultl = 'X' or result = 'X' then
01715                  result := 'X';
01716              else
01717                  result := result or resultl;
01718              end if;
01719          end loop;
01720          return result;
01721      end if;
01722 end function \?=/\;

01723 function \?=/\ (L, R : SIGNED) return std_ulogic is
01724  constant L_LEFT : INTEGER := L'LENGTH-1;
01725  constant R_LEFT : INTEGER := R'LENGTH-1;
01726  alias XL : SIGNED(L_LEFT downto 0) is L;
01727  alias XR : SIGNED(R_LEFT downto 0) is R;
01728  constant SIZE : NATURAL := MAXIMUM(L'LENGTH, R'LENGTH);
01729  variable LX : SIGNED(SIZE-1 downto 0);
01730  variable RX : SIGNED(SIZE-1 downto 0);
01731  variable result, resultl : STD_ULOGIC; -- result
01732 begin
01733      -- ?=
01734      if ((L'LENGTH < 1) or (R'LENGTH < 1)) then
01735          assert NO_WARNING
01736              report "NUMERIC_STD.\"?=/\": null detected, returning X"
01737              severity warning;
01738          return 'X';
01739      else
01740          LX := RESIZE(XL, SIZE);
01741          RX := RESIZE(XR, SIZE);
01742          result := '0';
01743          for i in LX'low to LX'high loop
01744              resultl := \?=/\ (LX(i), RX(i));
01745              if resultl = 'U' then
01746                  return 'U';
01747              elsif resultl = 'X' or result = 'X' then
01748                  result := 'X';
01749              else
01750                  result := result or resultl;
01751              end if;
01752          end loop;
01753          return result;
01754      end if;
01755 end function \?=/\;

01756 function Is_X (s : UNSIGNED) return BOOLEAN is
01757 begin
01758     return Is_X (STD_LOGIC_VECTOR (s));
01759 end function Is_X;

01760 function Is_X (s : SIGNED) return BOOLEAN is

```

```

01762 begin
01763     return Is_X (STD_LOGIC_VECTOR (s));
01764 end function Is_X;
01765 function \?>\ (L, R : UNSIGNED) return STD_ULOGIC is
01766 begin
01767     if ((l'length < 1) or (r'length < 1)) then
01768         assert NO_WARNING
01769             report "NUMERIC_STD.\"?>\": null detected, returning X"
01770             severity warning;
01771         return 'X';
01772     else
01773         for i in L'range loop
01774             if L(i) = '-' then
01775                 report "NUMERIC_STD.\"?>\": '-' found in compare string"
01776                     severity error;
01777                 return 'X';
01778             end if;
01779         end loop;
01780         for i in R'range loop
01781             if R(i) = '-' then
01782                 report "NUMERIC_STD.\"?>\": '-' found in compare string"
01783                     severity error;
01784                 return 'X';
01785             end if;
01786         end loop;
01787         if is_x(l) or is_x(r) then
01788             return 'X';
01789         elsif l > r then
01790             return '1';
01791         else
01792             return '0';
01793         end if;
01794     end if;
01795 end function \?>\;
01796 -- %% function "?>" (L, R : UNSIGNED) return std_ulogic is
01797 -- %% end function "?>"\;
01798 function \?>\ (L, R : SIGNED) return STD_ULOGIC is
01799 begin
01800     if ((l'length < 1) or (r'length < 1)) then
01801         assert NO_WARNING
01802             report "NUMERIC_STD.\"?>\": null detected, returning X"
01803             severity warning;
01804         return 'X';
01805     else
01806         for i in L'range loop
01807             if L(i) = '-' then
01808                 report "NUMERIC_STD.\"?>\": '-' found in compare string"
01809                     severity error;
01810                 return 'X';
01811             end if;
01812         end loop;
01813         for i in R'range loop
01814             if R(i) = '-' then
01815                 report "NUMERIC_STD.\"?>\": '-' found in compare string"
01816                     severity error;
01817                 return 'X';
01818             end if;
01819         end loop;
01820         if is_x(l) or is_x(r) then
01821             return 'X';
01822         elsif l > r then
01823             return '1';
01824         else
01825             return '0';
01826         end if;
01827     end if;
01828 end function \?>\;
01829 function \?>=\ (L, R : UNSIGNED) return STD_ULOGIC is
01830 begin
01831     if ((l'length < 1) or (r'length < 1)) then
01832         assert NO_WARNING
01833             report "NUMERIC_STD.\"?>=\": null detected, returning X"
01834             severity warning;
01835         return 'X';
01836     else
01837         for i in L'range loop
01838             if L(i) = '-' then
01839                 report "NUMERIC_STD.\"?>=\": '-' found in compare string"
01840                     severity error;
01841                 return 'X';
01842             end if;
01843         end loop;
01844         for i in R'range loop
01845             if R(i) = '-' then
01846                 report "NUMERIC_STD.\"?>=\": '-' found in compare string"
01847                     severity error;
01848                 return 'X';

```

```

01849      end if;
01850   end loop;
01851   if is_x(l) or is_x(r) then
01852     return 'X';
01853   elsif l >= r then
01854     return '1';
01855   else
01856     return '0';
01857   end if;
01858   end if;
01859 end function \?>=;
01860 -- %% function "?>=" (L, R : UNSIGNED) return std_ulogic is
01861 -- %% end function "?>=";
01862 function \?>=\ (L, R : SIGNED) return STD_ULOGIC is
01863 begin
01864   if ((l'length < 1) or (r'length < 1)) then
01865     assert NO_WARNING
01866     report "NUMERIC_STD.""?>=": null detected, returning X"
01867     severity warning;
01868     return 'X';
01869   else
01870     for i in L'range loop
01871       if L(i) = '-' then
01872         report "NUMERIC_STD.""?>=": '-' found in compare string"
01873         severity error;
01874         return 'X';
01875       end if;
01876     end loop;
01877     for i in R'range loop
01878       if R(i) = '-' then
01879         report "NUMERIC_STD.""?>=": '-' found in compare string"
01880         severity error;
01881         return 'X';
01882       end if;
01883     end loop;
01884     if is_x(l) or is_x(r) then
01885       return 'X';
01886     elsif l >= r then
01887       return '1';
01888     else
01889       return '0';
01890     end if;
01891   end if;
01892 end function \?>=;
01893 function \?<\ (L, R : UNSIGNED) return STD_ULOGIC is
01894 begin
01895   if ((l'length < 1) or (r'length < 1)) then
01896     assert NO_WARNING
01897     report "NUMERIC_STD.""?<": null detected, returning X"
01898     severity warning;
01899     return 'X';
01900   else
01901     for i in L'range loop
01902       if L(i) = '-' then
01903         report "NUMERIC_STD.""?<": '-' found in compare string"
01904         severity error;
01905         return 'X';
01906       end if;
01907     end loop;
01908     for i in R'range loop
01909       if R(i) = '-' then
01910         report "NUMERIC_STD.""?<": '-' found in compare string"
01911         severity error;
01912         return 'X';
01913       end if;
01914     end loop;
01915     if is_x(l) or is_x(r) then
01916       return 'X';
01917     elsif l < r then
01918       return '1';
01919     else
01920       return '0';
01921     end if;
01922   end if;
01923 end function \?<\;
01924 -- %% function "?<" (L, R : UNSIGNED) return std_ulogic is
01925 -- %% end function "?<";
01926 function \?<\ (L, R : SIGNED) return STD_ULOGIC is
01927 begin
01928   if ((l'length < 1) or (r'length < 1)) then
01929     assert NO_WARNING
01930     report "NUMERIC_STD.""?<": null detected, returning X"
01931     severity warning;
01932     return 'X';
01933   else
01934     for i in L'range loop
01935       if L(i) = '-' then

```

```

01936      report "NUMERIC_STD.""?<"": '-' found in compare string"
01937          severity error;
01938          return 'X';
01939      end if;
01940  end loop;
01941  for i in R'range loop
01942      if R(i) = '-' then
01943          report "NUMERIC_STD.""?<"": '-' found in compare string"
01944          severity error;
01945          return 'X';
01946      end if;
01947  end loop;
01948  if is_x(l) or is_x(r) then
01949      return 'X';
01950  elsif l < r then
01951      return '1';
01952  else
01953      return '0';
01954  end if;
01955 end if;
01956 end function \?<\;
01957 function \?<=\ (L, R : UNSIGNED) return STD_ULONGIC is
01958 begin
01959     if ((l'length < 1) or (r'length < 1)) then
01960         assert NO_WARNING
01961         report "NUMERIC_STD.""?<=""": null detected, returning X"
01962         severity warning;
01963         return 'X';
01964     else
01965         for i in L'range loop
01966             if L(i) = '-' then
01967                 report "NUMERIC_STD.""?<=""": '-' found in compare string"
01968                 severity error;
01969                 return 'X';
01970             end if;
01971         end loop;
01972         for i in R'range loop
01973             if R(i) = '-' then
01974                 report "NUMERIC_STD.""?<=""": '-' found in compare string"
01975                 severity error;
01976                 return 'X';
01977             end if;
01978         end loop;
01979         if is_x(l) or is_x(r) then
01980             return 'X';
01981         elsif l <= r then
01982             return '1';
01983         else
01984             return '0';
01985         end if;
01986     end if;
01987 end function \?<=\;
01988 -- %% function "?<=" (L, R : UNSIGNED) return std_ulogic is
01989 -- %% end function "?<=";
01990 function \?<=\ (L, R : SIGNED) return STD_ULONGIC is
01991 begin
01992     if ((l'length < 1) or (r'length < 1)) then
01993         assert NO_WARNING
01994         report "NUMERIC_STD.""?<=""": null detected, returning X"
01995         severity warning;
01996         return 'X';
01997     else
01998         for i in L'range loop
01999             if L(i) = '-' then
02000                 report "NUMERIC_STD.""?<=""": '-' found in compare string"
02001                 severity error;
02002                 return 'X';
02003             end if;
02004         end loop;
02005         for i in R'range loop
02006             if R(i) = '-' then
02007                 report "NUMERIC_STD.""?<=""": '-' found in compare string"
02008                 severity error;
02009                 return 'X';
02010             end if;
02011         end loop;
02012         if is_x(l) or is_x(r) then
02013             return 'X';
02014         elsif l <= r then
02015             return '1';
02016         else
02017             return '0';
02018         end if;
02019     end if;
02020 end function \?<=\
02021 -- %% END replicated functions

```

```

02023 -- Special version of "minimum" to do some boundary checking without errors
02024 function mins (l, r : INTEGER)
02025   return INTEGER is
02026 begin -- function mins
02027   if (L = INTEGER'low or R = INTEGER'low) then
02028     return 0;                                -- error condition, silent
02029   end if;
02030   return minimum (L, R);
02031 end function mins;
02032
02033 -- Special version of "minimum" to do some boundary checking with errors
02034 function mine (l, r : INTEGER)
02035   return INTEGER is
02036 begin -- function mine
02037   if (L = INTEGER'low or R = INTEGER'low) then
02038     report fixed_pkg'instance_name
02039       & " Unbounded number passed, was a literal used?"
02040       severity error;
02041     return 0;
02042   end if;
02043   return minimum (L, R);
02044 end function mine;
02045
02046 -- The following functions are used only internally. Every function
02047 -- calls "cleanvec" either directly or indirectly.
02048 -- purpose: Fixes "downto" problem and resolves meta states
02049 function cleanvec (
02050   arg : UNRESOLVED_sfixed)           -- input
02051   return UNRESOLVED_sfixed is
02052   constant left_index : INTEGER := maximum(arg'left, arg'right);
02053   constant right_index : INTEGER := mins(arg'left, arg'right);
02054   variable result    : UNRESOLVED_sfixed (arg'range);
02055 begin -- function cleanvec
02056   assert not (arg'ascending and (arg'low /= INTEGER'low))
02057   report fixed_pkg'instance_name
02058     & " Vector passed using a ""to"" range, expected is ""downto"""
02059     severity error;
02060   return arg;
02061 end function cleanvec;
02062
02063 -- purpose: Fixes "downto" problem and resolves meta states
02064 function cleanvec (
02065   arg : UNRESOLVED_ufixed)           -- input
02066   return UNRESOLVED_ufixed is
02067   constant left_index : INTEGER := maximum(arg'left, arg'right);
02068   constant right_index : INTEGER := mins(arg'left, arg'right);
02069   variable result    : UNRESOLVED_ufixed (arg'range);
02070 begin -- function cleanvec
02071   assert not (arg'ascending and (arg'low /= INTEGER'low))
02072   report fixed_pkg'instance_name
02073     & " Vector passed using a ""to"" range, expected is ""downto"""
02074     severity error;
02075   return arg;
02076 end function cleanvec;
02077
02078 -- Type convert a "unsigned" into a "ufixed", used internally
02079 function to_fixed (
02080   arg          : UNSIGNED; -- shifted vector
02081   constant left_index : INTEGER;
02082   constant right_index : INTEGER)
02083   return UNRESOLVED_ufixed is
02084   variable result : UNRESOLVED_ufixed (left_index downto right_index);
02085 begin -- function to_fixed
02086   result := UNRESOLVED_ufixed(arg);
02087   return result;
02088 end function to_fixed;
02089
02090 -- Type convert a "signed" into an "sfixed", used internally
02091 function to_fixed (
02092   arg          : SIGNED; -- shifted vector
02093   constant left_index : INTEGER;
02094   constant right_index : INTEGER)
02095   return UNRESOLVED_sfixed is
02096   variable result : UNRESOLVED_sfixed (left_index downto right_index);
02097 begin -- function to_fixed
02098   result := UNRESOLVED_sfixed(arg);
02099   return result;
02100 end function to_fixed;
02101
02102 -- Type convert a "ufixed" into an "unsigned", used internally
02103 function to_uns (
02104   arg : UNRESOLVED_ufixed)           -- fp vector
02105   return UNSIGNED is
02106   subtype t is UNSIGNED(arg'high - arg'low downto 0);
02107   variable slv : t;
02108 begin -- function to_uns
02109   slv := t(arg);

```

```

02110      return slv;
02111  end function to_uns;
02112
02113  -- Type convert an "sfixed" into a "signed", used internally
02114  function to_s (
02115    arg : UNRESOLVED_sfixed)           -- fp vector
02116  return SIGNED is
02117    subtype t is SIGNED(arg'high - arg'low downto 0);
02118    variable slv : t;
02119  begin -- function to_s
02120    slv := t(arg);
02121    return slv;
02122  end function to_s;
02123
02124  -- adds 1 to the LSB of the number
02125  procedure round_up (arg      : in UNRESOLVED_ufixed;
02126                      result   : out UNRESOLVED_ufixed;
02127                      overflowx : out BOOLEAN) is
02128    variable arguns, resuns : UNSIGNED (arg'high-arg'low+1 downto 0)
02129    := (others => '0');
02130  begin -- round_up
02131    arguns (arguns'high-1 downto 0) := to_uns (arg);
02132    resuns            := arguns + 1;
02133    result := to_fixed(resuns(arg'high-arg'low
02134                                downto 0), arg'high, arg'low);
02135    overflowx := (resuns(resuns'high) = '1');
02136  end procedure round_up;
02137
02138  -- adds 1 to the LSB of the number
02139  procedure round_up (arg      : in UNRESOLVED_sfixed;
02140                      result   : out UNRESOLVED_sfixed;
02141                      overflowx : out BOOLEAN) is
02142    variable args, ress : SIGNED (arg'high-arg'low+1 downto 0);
02143  begin -- round_up
02144    args (args'high-1 downto 0) := to_s (arg);
02145    args(args'high)          := arg(arg'high); -- sign extend
02146    ress            := args + 1;
02147    result := to_fixed(ress (ress'high-1
02148                                downto 0), arg'high, arg'low);
02149    overflowx := ((arg(arg'high) /= ress(ress'high-1))
02150                  and (or_reduce (STD_ULOGIC_VECTOR(ress)) /= '0'));
02151  end procedure round_up;
02152
02153  -- Rounding - Performs a "round_nearest" (IEEE 754) which rounds up
02154  -- when the remainder is > 0.5. If the remainder IS 0.5 then if the
02155  -- bottom bit is a "1" it is rounded, otherwise it remains the same.
02156  function round_fixed (arg      : UNRESOLVED_ufixed;
02157                      remainder : UNRESOLVED_ufixed;
02158                      overflow_style : fixed_overflow_style_type :=
02159    fixed_overflow_style)
02160  return UNRESOLVED_ufixed is
02161  variable rounds      : BOOLEAN;
02162  variable round_overflow : BOOLEAN;
02163  variable result       : UNRESOLVED_ufixed (arg'range);
02164  begin
02165    rounds := false;
02166    if (remainder'length > 1) then
02167      if (remainder(remainder'high) = '1') then
02168        rounds := (arg(arg'low) = '1')
02169        or (or_reduce (to_suv(remainder'high-1 downto
02170                           remainder'low))) = '1';
02171      end if;
02172    else
02173      rounds := (arg(arg'low) = '1') and (remainder (remainder'high) = '1');
02174    end if;
02175    if rounds then
02176      round_up(arg      => arg,
02177                 result   => result,
02178                 overflowx => round_overflow);
02179    else
02180      result := arg;
02181    end if;
02182    if (overflow_style = fixed_saturate) and round_overflow then
02183      result := saturate(result'high, result'low);
02184    end if;
02185    return result;
02186  end function round_fixed;
02187
02188  -- Rounding case statement
02189  function round_fixed (arg      : UNRESOLVED_sfixed;
02190                      remainder : UNRESOLVED_sfixed;
02191                      overflow_style : fixed_overflow_style_type :=
02192    fixed_overflow_style)
02193  return UNRESOLVED_sfixed is
02194  variable rounds      : BOOLEAN;
02195  variable round_overflow : BOOLEAN;
02196  variable result       : UNRESOLVED_sfixed (arg'range);

```

```

02195 begin
02196     rounds := false;
02197     if (remainder'length > 1) then
02198         if (remainder(remainder'high) = '1') then
02199             rounds := (arg(arg'low) = '1')
02200                 or (or_reduce (to_sulv(remainder'high-1 downto
02201                                         remainder'low))) = '1');
02202         end if;
02203     else
02204         rounds := (arg(arg'low) = '1') and (remainder(remainder'high) = '1');
02205     end if;
02206     if rounds then
02207         round_up(arg      => arg,
02208                 result   => result,
02209                 overflowx => round_overflow);
02210     else
02211         result := arg;
02212     end if;
02213     if round_overflow then
02214         if (overflow_style = fixed_saturate) then
02215             if arg(arg'high) = '0' then
02216                 result := saturate(result'high, result'low);
02217             else
02218                 result := not saturate(result'high, result'low);
02219             end if;
02220             -- Sign bit not fixed when wrapping
02221         end if;
02222     end if;
02223     return result;
02224 end function round_fixed;
02225
02226 -- converts an sfixed into a ufixed. The output is the same length as the
02227 -- input, because abs("1000") = "1000" = 8.
02228 function to_ufixed (
02229     arg : UNRESOLVED_sfixed)
02230     return UNRESOLVED_ufixed
02231 is
02232     constant left_index : INTEGER := arg'high;
02233     constant right_index : INTEGER := min(arg'low, arg'high);
02234     variable xarg      : UNRESOLVED_sfixed(left_index+1 downto right_index);
02235     variable result     : UNRESOLVED_ufixed(left_index downto right_index);
02236 begin
02237     if arg'length < 1 then
02238         return NAUF;
02239     end if;
02240     xarg := abs(arg);
02241     result := UNRESOLVED_ufixed(xarg(left_index downto right_index));
02242     return result;
02243 end function to_ufixed;
02244
02245 -----
02246 -- Visible functions
02247 -----
02248
02249 -- Conversion functions. These are needed for synthesis where typically
02250 -- the only input and output type is a std_logic_vector.
02251 function to_sulv (
02252     arg : UNRESOLVED_ufixed)           -- fixed point vector
02253     return STD_ULOGIC_VECTOR is
02254     variable result : STD_ULOGIC_VECTOR (arg'length-1 downto 0);
02255 begin
02256     if arg'length < 1 then
02257         return NSLV;
02258     end if;
02259     result := STD_ULOGIC_VECTOR(arg);
02260     return result;
02261 end function to_sulv;
02262
02263 function to_sulv (
02264     arg : UNRESOLVED_sfixed)           -- fixed point vector
02265     return STD_ULOGIC_VECTOR is
02266     variable result : STD_ULOGIC_VECTOR (arg'length-1 downto 0);
02267 begin
02268     if arg'length < 1 then
02269         return NSLV;
02270     end if;
02271     result := STD_ULOGIC_VECTOR(arg);
02272     return result;
02273 end function to_sulv;
02274
02275 function to_slv (
02276     arg : UNRESOLVED_ufixed)           -- fixed point vector
02277     return STD_LOGIC_VECTOR is
02278 begin
02279     return to_stdlogicvector(to_sulv(arg));
02280 end function to_slv;
02281

```

```

02282   function to_slv (
02283     arg : UNRESOLVED_sfixed)                      -- fixed point vector
02284   return STD_LOGIC_VECTOR is
02285 begin
02286   return to_stdlogicvector(to_sulv(arg));
02287 end function to_slv;
02288
02289   function to_ufixed (
02290     arg          : STD_ULOGIC_VECTOR;    -- shifted vector
02291     constant left_index  : INTEGER;
02292     constant right_index : INTEGER)
02293   return unresolved_ufixed is
02294   variable result : UNRESOLVED_ufixed (left_index downto right_index);
02295 begin
02296   if (arg'length < 1 or right_index > left_index) then
02297     return NAUF;
02298   end if;
02299   if (arg'length /= result'length) then
02300     report fixed_pkg'instance_name & "TO_UFIXED(SLV) "
02301       & "Vector lengths do not match. Input length is "
02302       & INTEGER'image(arg'length) & " and output will be "
02303       & INTEGER'image(result'length) & " wide."
02304     severity error;
02305   return NAUF;
02306 else
02307   result := to_fixed (arg      => UNSIGNED(arg),
02308                      left_index => left_index,
02309                      right_index => right_index);
02310   return result;
02311 end if;
02312 end function to_ufixed;
02313
02314   function to_sfixed (
02315     arg          : STD_ULOGIC_VECTOR;    -- shifted vector
02316     constant left_index  : INTEGER;
02317     constant right_index : INTEGER)
02318   return unresolved_sfixed is
02319   variable result : UNRESOLVED_sfixed (left_index downto right_index);
02320 begin
02321   if (arg'length < 1 or right_index > left_index) then
02322     return NASF;
02323   end if;
02324   if (arg'length /= result'length) then
02325     report fixed_pkg'instance_name & "TO_SFIXED(SLV) "
02326       & "Vector lengths do not match. Input length is "
02327       & INTEGER'image(arg'length) & " and output will be "
02328       & INTEGER'image(result'length) & " wide."
02329     severity error;
02330   return NASF;
02331 else
02332   result := to_fixed (arg      => SIGNED(arg),
02333                      left_index => left_index,
02334                      right_index => right_index);
02335   return result;
02336 end if;
02337 end function to_sfixed;
02338
02339 -- Two's complement number, Grows the vector by 1 bit.
02340 -- because "abs (1000.000) = 01000.000" or abs(-16) = 16.
02341 function "abs" (
02342   arg : UNRESOLVED_sfixed)                      -- fixed point input
02343   return UNRESOLVED_sfixed is
02344   constant left_index  : INTEGER := arg'high;
02345   constant right_index : INTEGER := mine(arg'low, arg'low);
02346   variable ressns   : SIGNED (arg'length downto 0);
02347   variable result    : UNRESOLVED_sfixed (left_index+1 downto right_index);
02348 begin
02349   if (arg'length < 1 or result'length < 1) then
02350     return NASF;
02351   end if;
02352   ressns (arg'length-1 downto 0) := to_s (cleanvec (arg));
02353   ressns (arg'length)           := ressns (arg'length-1);  -- expand sign bit
02354   result                  := to_fixed (abs(ressns), left_index+1, right_index);
02355   return result;
02356 end function "abs";
02357
02358 -- also grows the vector by 1 bit.
02359 function "-" (
02360   arg : UNRESOLVED_sfixed)                      -- fixed point input
02361   return UNRESOLVED_sfixed is
02362   constant left_index  : INTEGER := arg'high+1;
02363   constant right_index : INTEGER := mine(arg'low, arg'low);
02364   variable ressns   : SIGNED (arg'length downto 0);
02365   variable result    : UNRESOLVED_sfixed (left_index downto right_index);
02366 begin
02367   if (arg'length < 1 or result'length < 1) then
02368     return NASF;

```

```

02369      end if;
02370      ressns (arg'length-1 downto 0) := to_s (cleanvec(arg));
02371      ressns (arg'length)           := ressns (arg'length-1); -- expand sign bit
02372      result                      := to_fixed (-ressns, left_index, right_index);
02373      return result;
02374  end function "-";
02375
02376  -- Addition
02377  function "+" (
02378      l, r : UNRESOLVED_ufixed)  -- ufixed(a downto b) + ufixed(c downto d) =
02379      return UNRESOLVED_ufixed is -- ufixed(max(a,c)+1 downto min(b,d))
02380      constant left_index       : INTEGER := maximum(l'high, r'high)+1;
02381      constant right_index      : INTEGER := mine(l'low, r'low);
02382      variable lresize, rresize : UNRESOLVED_ufixed (left_index downto right_index);
02383      variable result          : UNRESOLVED_ufixed (left_index downto right_index);
02384      variable lslv, rslv      : UNSIGNED (left_index-right_index
02385                                         downto 0);
02386      variable result_slv      : UNSIGNED (left_index-right_index
02387                                         downto 0);
02388  begin
02389      if (l'length < 1 or r'length < 1 or result'length < 1) then
02390          return NAUF;
02391      end if;
02392      lresize := resize (l, left_index, right_index);
02393      rresize := resize (r, left_index, right_index);
02394      lslv    := to_uns (lresize);
02395      rslv    := to_uns (rresize);
02396      result_slv := lslv + rslv;
02397      result   := to_fixed(result_slv, left_index, right_index);
02398      return result;
02399  end function "+";
02400
02401  function "+" (
02402      l, r : UNRESOLVED_sfixed)  -- sfixed(a downto b) + sfixed(c downto d) =
02403      return UNRESOLVED_sfixed is -- sfixed(max(a,c)+1 downto min(b,d))
02404      constant left_index       : INTEGER := maximum(l'high, r'high)+1;
02405      constant right_index      : INTEGER := mine(l'low, r'low);
02406      variable lresize, rresize : UNRESOLVED_sfixed (left_index downto right_index);
02407      variable result          : UNRESOLVED_sfixed (left_index downto right_index);
02408      variable lslv, rslv      : SIGNED (left_index-right_index downto 0);
02409      variable result_slv      : SIGNED (left_index-right_index downto 0);
02410  begin
02411      if (l'length < 1 or r'length < 1 or result'length < 1) then
02412          return NASF;
02413      end if;
02414      lresize := resize (l, left_index, right_index);
02415      rresize := resize (r, left_index, right_index);
02416      lslv    := to_s (lresize);
02417      rslv    := to_s (rresize);
02418      result_slv := lslv + rslv;
02419      result   := to_fixed(result_slv, left_index, right_index);
02420      return result;
02421  end function "+";
02422
02423  -- Subtraction
02424  function "-" (
02425      l, r : UNRESOLVED_ufixed)  -- ufixed(a downto b) - ufixed(c downto d) =
02426      return UNRESOLVED_ufixed is -- ufixed(max(a,c)+1 downto min(b,d))
02427      constant left_index       : INTEGER := maximum(l'high, r'high)+1;
02428      constant right_index      : INTEGER := mine(l'low, r'low);
02429      variable lresize, rresize : UNRESOLVED_ufixed (left_index downto right_index);
02430      variable result          : UNRESOLVED_ufixed (left_index downto right_index);
02431      variable lslv, rslv      : UNSIGNED (left_index-right_index
02432                                         downto 0);
02433      variable result_slv      : UNSIGNED (left_index-right_index
02434                                         downto 0);
02435  begin
02436      if (l'length < 1 or r'length < 1 or result'length < 1) then
02437          return NAUF;
02438      end if;
02439      lresize := resize (l, left_index, right_index);
02440      rresize := resize (r, left_index, right_index);
02441      lslv    := to_uns (lresize);
02442      rslv    := to_uns (rresize);
02443      result_slv := lslv - rslv;
02444      result   := to_fixed(result_slv, left_index, right_index);
02445      return result;
02446  end function "-";
02447
02448  function "-" (
02449      l, r : UNRESOLVED_sfixed)  -- sfixed(a downto b) - sfixed(c downto d) =
02450      return UNRESOLVED_sfixed is -- sfixed(max(a,c)+1 downto min(b,d))
02451      constant left_index       : INTEGER := maximum(l'high, r'high)+1;
02452      constant right_index      : INTEGER := mine(l'low, r'low);
02453      variable lresize, rresize : UNRESOLVED_sfixed (left_index downto right_index);
02454      variable result          : UNRESOLVED_sfixed (left_index downto right_index);
02455      variable lslv, rslv      : SIGNED (left_index-right_index downto 0);

```

```

02456     variable result_slv      : SIGNED (left_index-right_index downto 0);
02457 begin
02458   if (l'length < 1 or r'length < 1 or result'length < 1) then
02459     return NASF;
02460   end if;
02461   lresize    := resize (l, left_index, right_index);
02462   rresize    := resize (r, left_index, right_index);
02463   lslv       := to_s (lresize);
02464   rslv       := to_s (rresize);
02465   result_slv := lslv - rslv;
02466   result     := to_fixed(result_slv, left_index, right_index);
02467   return result;
02468 end function "-";
02469
02470 function "*" (
02471   l, r : UNRESOLVED_ufixed)  -- ufixed(a downto b) * ufixed(c downto d) =
02472   return UNRESOLVED_ufixed is           -- ufixed(a+c+1 downto b+d)
02473   variable lslv      : UNSIGNED (l'length-1 downto 0);
02474   variable rslv      : UNSIGNED (r'length-1 downto 0);
02475   variable result_slv : UNSIGNED (r'length+l'length-1 downto 0);
02476   variable result     : UNRESOLVED_ufixed (l'high + r'high+1 downto
02477                                         mine(l'low, l'low) + mine(r'low, r'low));
02478 begin
02479   if (l'length < 1 or r'length < 1 or
02480       result'length /= result_slv'length) then
02481     return NAUF;
02482   end if;
02483   lslv      := to_uns (cleanvec(l));
02484   rslv      := to_uns (cleanvec(r));
02485   result_slv := lslv * rslv;
02486   result     := to_fixed (result_slv, result'high, result'low);
02487   return result;
02488 end function "*";
02489
02490 function "*" (
02491   l, r : UNRESOLVED_sfixed)  -- sfixed(a downto b) * sfixed(c downto d) =
02492   return UNRESOLVED_sfixed is           -- sfixed(a+c+1 downto b+d)
02493   variable lslv      : SIGNED (l'length-1 downto 0);
02494   variable rslv      : SIGNED (r'length-1 downto 0);
02495   variable result_slv : SIGNED (r'length+l'length-1 downto 0);
02496   variable result     : UNRESOLVED_sfixed (l'high + r'high+1 downto
02497                                         mine(l'low, l'low) + mine(r'low, r'low));
02498 begin
02499   if (l'length < 1 or r'length < 1 or
02500       result'length /= result_slv'length) then
02501     return NASF;
02502   end if;
02503   lslv      := to_s (cleanvec(l));
02504   rslv      := to_s (cleanvec(r));
02505   result_slv := lslv * rslv;
02506   result     := to_fixed (result_slv, result'high, result'low);
02507   return result;
02508 end function "*";
02509
02510 function "/" (
02511   l, r : UNRESOLVED_ufixed)  -- ufixed(a downto b) / ufixed(c downto d) =
02512   return UNRESOLVED_ufixed is           -- ufixed(a-d downto b-c-1)
02513 begin
02514   return divide (l, r);
02515 end function "/";
02516
02517 function "/" (
02518   l, r : UNRESOLVED_sfixed)  -- sfixed(a downto b) / sfixed(c downto d) =
02519   return UNRESOLVED_sfixed is           -- sfixed(a-d+1 downto b-c)
02520 begin
02521   return divide (l, r);
02522 end function "/";
02523
02524 -- This version of divide gives the user more control
02525 -- ufixed(a downto b) / ufixed(c downto d) = ufixed(a-d downto b-c-1)
02526 function divide (
02527   l, r           : UNRESOLVED_ufixed;
02528   constant round_style : fixed_round_style_type :=
02529   fixed_round_style;
02530   constant guard_bits : NATURAL          := fixed_guard_bits)
02531   variable result : UNRESOLVED_ufixed (l'high - mine(r'low, r'low) downto
02532                                         mine (l'low, l'low) - r'high-1);
02533   variable dreslt  : UNRESOLVED_ufixed (result'high downto result'low -guard_bits);
02534   variable lresize  : UNRESOLVED_ufixed (l'high downto l'high - dreslt'length+1);
02535   variable lslv     : UNSIGNED (lresize'length-1 downto 0);
02536   variable rslv     : UNSIGNED (r'length-1 downto 0);
02537   variable result_slv : UNSIGNED (lresize'length-1 downto 0);
02538 begin
02539   if (l'length < 1 or r'length < 1 or
02540       mins(r'low, r'low) /= r'low or mins(l'low, l'low) /= l'low) then
02541     return NAUF;

```

```

02542     end if;
02543     lresize := resize (arg          => l,
02544                  left_index    => lresize'high,
02545                  right_index   => lresize'low,
02546                  overflow_style => fixed_wrap, -- vector only grows
02547                  round_style    => fixed_truncate);
02548     lslv := to_uns (cleanvec (lresize));
02549     rslv := to_uns (cleanvec (r));
02550     if (rslv = 0) then
02551       report fixed_pkg'instance_name
02552         & "DIVIDE(ufixed) Division by zero" severity error;
02553     result := saturate (result'high, result'low); -- saturate
02554   else
02555     result_slv := lslv / rslv;
02556     dreslt    := to_fixed (result_slv, dreslt'high, dreslt'low);
02557     result := resize (arg          => dreslt,
02558                  left_index    => result'high,
02559                  right_index   => result'low,
02560                  overflow_style => fixed_wrap, -- overflow impossible
02561                  round_style    => round_style);
02562   end if;
02563   return result;
02564 end function divide;
02565
02566 -- sfixed(a downto b) / sfixed(c downto d) = sfixed(a-d+1 downto b-c)
02567 function divide (
02568   l, r           : UNRESOLVED_sfixed;
02569   constant round_style : fixed_round_style_type :=
02570     fixed_round_style;
02571   constant guard_bits  : NATURAL           := fixed_guard_bits)
02572   return UNRESOLVED_sfixed is
02573     variable result      : UNRESOLVED_sfixed (l'high - mine(r'low, r'low) + 1 downto
02574                                               mine (l'low, l'low) - r'high);
02574     variable dreslt      : UNRESOLVED_sfixed (result'high downto result'low-guard_bits);
02575     variable lresize     : UNRESOLVED_sfixed (l'high+1 downto l'high+1 -dreslt'length+1);
02576     variable lslv        : SIGNED (lresize'length-1 downto 0);
02577     variable rslv        : SIGNED (r'length-1 downto 0);
02578     variable result_slv : SIGNED (lresize'length-1 downto 0);
02579 begin
02580   if (l'length < 1 or r'length < 1 or
02581     mins(r'low, r'low) /= r'low or mins(l'low, l'low) /= l'low) then
02582     return NASF;
02583   end if;
02584   lresize := resize (arg          => l,
02585                      left_index    => lresize'high,
02586                      right_index   => lresize'low,
02587                      overflow_style => fixed_wrap, -- vector only grows
02588                      round_style    => fixed_truncate);
02589   lslv := to_s (cleanvec (lresize));
02590   rslv := to_s (cleanvec (r));
02591   if (rslv = 0) then
02592     report fixed_pkg'instance_name
02593       & "DIVIDE(sfixed) Division by zero" severity error;
02594     result := saturate (result'high, result'low);
02595   else
02596     result_slv := lslv / rslv;
02597     dreslt    := to_fixed (result_slv, dreslt'high, dreslt'low);
02598     result := resize (arg          => dreslt,
02599                      left_index    => result'high,
02600                      right_index   => result'low,
02601                      overflow_style => fixed_wrap, -- overflow impossible
02602                      round_style    => round_style);
02603   end if;
02604   return result;
02605 end function divide;
02606
02607 -- 1 / ufixed(a downto b) = ufixed(-b downto -a-1)
02608 function reciprocal (
02609   arg          : UNRESOLVED_ufixed; -- fixed point input
02610   constant round_style : fixed_round_style_type :=
02611     fixed_round_style;
02612   constant guard_bits  : NATURAL           := fixed_guard_bits)
02613   constant one : UNRESOLVED_ufixed (0 downto 0) := "1";
02614 begin
02615   return divide (1          => one,
02616                  r          => arg,
02617                  round_style => round_style,
02618                  guard_bits  => guard_bits);
02619 end function reciprocal;
02620
02621 -- 1 / sfixed(a downto b) = sfixed(-b+1 downto -a)
02622 function reciprocal (
02623   arg          : UNRESOLVED_sfixed; -- fixed point input
02624   constant round_style : fixed_round_style_type :=
02625     fixed_round_style;
02626   constant guard_bits  : NATURAL           := fixed_guard_bits)

```

```

02626     return UNRESOLVED_sfixed is
02627     constant one      : UNRESOLVED_sfixed (1 downto 0) := "01"; -- extra bit.
02628     variable resultx : UNRESOLVED_sfixed (-mine(arg'low, arg'low)+2 downto -arg'high);
02629 begin
02630     if (arg'length < 1 or resultx'length < 1) then
02631         return NASF;
02632     else
02633         resultx := divide (l              => one,
02634                           r              => arg,
02635                           round_style => round_style,
02636                           guard_bits   => guard_bits);
02637         return resultx (resultx'high-1 downto resultx'low); -- remove extra bit
02638     end if;
02639 end function reciprocal;
02640
02641 -- ufixed (a downto b) rem ufixed (c downto d)
02642 --      = ufixed (min(a,c) downto min(b,d))
02643 function "rem" (
02644     l, r : UNRESOLVED_ufixed)           -- fixed point input
02645     return UNRESOLVED_ufixed is
02646 begin
02647     return remainder (l, r);
02648 end function "rem";
02649
02650 -- remainder
02651 -- sfixed (a downto b) rem sfixed (c downto d)
02652 --      = sfixed (min(a,c) downto min(b,d))
02653 function "rem" (
02654     l, r : UNRESOLVED_sfixed)           -- fixed point input
02655     return UNRESOLVED_sfixed is
02656 begin
02657     return remainder (l, r);
02658 end function "rem";
02659
02660 -- ufixed (a downto b) rem ufixed (c downto d)
02661 --      = ufixed (min(a,c) downto min(b,d))
02662 function remainder (
02663     l, r              : UNRESOLVED_ufixed;           -- fixed point input
02664     constant round_style : fixed_round_style_type := fixed_round_style;
02665     constant guard_bits  : NATURAL                  := fixed_guard_bits)
02666     return UNRESOLVED_ufixed is
02667     variable result      : UNRESOLVED_ufixed (minimum(l'high, r'high) downto
02668                               mine(l'low, r'low));
02669     variable lresize     : UNRESOLVED_ufixed (maximum(l'high, r'low) downto
02670                               mins(r'low, r'low)-guard_bits);
02671     variable rresize     : UNRESOLVED_ufixed (r'high downto r'low-guard_bits);
02672     variable dresult    : UNRESOLVED_ufixed (rresize'range);
02673     variable lslv       : UNSIGNED (lresize'length-1 downto 0);
02674     variable rslv       : UNSIGNED (rresize'length-1 downto 0);
02675     variable result_slv : UNSIGNED (rslv'r'range);
02676 begin
02677     if (l'length < 1 or r'length < 1 or
02678         mins(r'low, r'low) /= r'low or mins(l'low, l'low) /= l'low) then
02679         return NAUF;
02680     end if;
02681     lresize := resize (arg          => l,
02682                           left_index    => lresize'high,
02683                           right_index   => lresize'low,
02684                           overflow_style => fixed_wrap, -- vector only grows
02685                           round_style   => fixed_truncate);
02686     lslv := to_uns (lresize);
02687     rresize := resize (arg          => r,
02688                           left_index    => rresize'high,
02689                           right_index   => rresize'low,
02690                           overflow_style => fixed_wrap, -- vector only grows
02691                           round_style   => fixed_truncate);
02692     rslv := to_uns (rresize);
02693     if (rslv = 0) then
02694         report fixed_pkg'instance_name
02695             & "remainder(ufixed) Division by zero" severity error;
02696         result := saturate (result'high, result'low); -- saturate
02697     else
02698         if (r'low <= l'high) then
02699             result_slv := lslv rem rslv;
02700             dresult := to_fixed (result_slv, dresult'high, dresult'low);
02701             result := resize (arg          => dresult,
02702                               left_index    => result'high,
02703                               right_index   => result'low,
02704                               overflow_style => fixed_wrap, -- can't overflow
02705                               round_style   => round_style);
02706         end if;
02707         if l'low < r'low then
02708             result(mins(r'low-1, l'high) downto l'low) :=
02709                 cleanvec(l(mins(r'low-1, l'high) downto l'low));
02710         end if;
02711     end if;

```

```

02712      return result;
02713  end function remainder;
02714
02715  -- remainder
02716  -- sfixed (a downto b) rem sfixed (c downto d)
02717  --      = sfixed (min(a,c) downto min(b,d))
02718  function remainder (
02719    l, r           : UNRESOLVED_sfixed; -- fixed point input
02720    constant round_style : fixed_round_style_type := 
02721      fixed_round_style;
02722    constant guard_bits : NATURAL             := fixed_guard_bits)
02723  return UNRESOLVED_sfixed is
02724  variable l_abs   : UNRESOLVED_ufixed (l'range);
02725  variable r_abs   : UNRESOLVED_ufixed (r'range);
02726  variable result  : UNRESOLVED_sfixed (minimum(r'high, l'high) downto
02727                                mine(r'low, l'low));
02728  variable neg_result : UNRESOLVED_sfixed (minimum(r'high, l'high)+1 downto
02729                                mins(r'low, l'low));
02729 begin
02730  if (l'length < 1 or r'length < 1 or
02731      mins(r'low, r'low) /= r'low or mins(l'low, l'low) /= l'low) then
02732    return NASF;
02733  end if;
02734  l_abs := to_ufixed (l);
02735  r_abs := to_ufixed (r);
02736  result := UNRESOLVED_sfixed (remainder (
02737    1              => l_abs,
02738    r              => r_abs,
02739    round_style => round_style));
02740  neg_result := -result;
02741  if l(l'high) = '1' then
02742    result := neg_result(result'range);
02743  end if;
02744  return result;
02745 end function remainder;
02746
02747  -- modulo
02748  -- ufixed (a downto b) mod ufixed (c downto d)
02749  --      = ufixed (min(a,c) downto min(b, d))
02750  function "mod"
02751    l, r : UNRESOLVED_ufixed; -- fixed point input
02752  return UNRESOLVED_ufixed is
02753  begin
02754    return modulo (l, r);
02755  end function "mod";
02756
02757  -- sfixed (a downto b) mod sfixed (c downto d)
02758  --      = sfixed (c downto min(b, d))
02759  function "mod"
02760    l, r : UNRESOLVED_sfixed; -- fixed point input
02761  return UNRESOLVED_sfixed is
02762  begin
02763    return modulo(l, r);
02764  end function "mod";
02765
02766  -- modulo
02767  -- ufixed (a downto b) mod ufixed (c downto d)
02768  --      = ufixed (min(a,c) downto min(b, d))
02769  function modulo
02770    l, r           : UNRESOLVED_ufixed; -- fixed point input
02771    constant round_style : fixed_round_style_type := 
02772      fixed_round_style;
02773    constant guard_bits : NATURAL             := fixed_guard_bits)
02774  begin
02775    return remainder(l           => l,
02776                      r           => r,
02777                      round_style => round_style,
02778                      guard_bits  => guard_bits);
02779  end function modulo;
02780
02781  -- sfixed (a downto b) mod sfixed (c downto d)
02782  --      = sfixed (c downto min(b, d))
02783  function modulo
02784    l, r           : UNRESOLVED_sfixed; -- fixed point input
02785    constant overflow_style : fixed_overflow_style_type := 
02786      fixed_overflow_style;
02787    constant round_style   : fixed_round_style_type := 
02788      fixed_round_style;
02789    constant guard_bits   : NATURAL             := fixed_guard_bits)
02790  return UNRESOLVED_sfixed is
02791  variable l_abs   : UNRESOLVED_ufixed (l'range);
02792  variable r_abs   : UNRESOLVED_ufixed (r'range);
02793  variable result  : UNRESOLVED_sfixed (r'high downto
02794                                mine(r'low, l'low));
02793  variable dresult : UNRESOLVED_sfixed (minimum(r'high, l'high)+1 downto
02794                                mins(r'low, l'low));

```

```

02795     variable dresult_not_zero : BOOLEAN;
02796 begin
02797     if (l'length < 1 or r'length < 1 or
02798         mins(r'low, r'low) /= r'low or mins(l'low, l'low) /= l'low) then
02799         return NASF;
02800     end if;
02801     l_abs := to_ufixed (l);
02802     r_abs := to_ufixed (r);
02803     dresult := "0" & UNRESOLVED_sfixed(remainder (l           => l_abs,
02804                                         r           => r_abs,
02805                                         round_style => round_style));
02806     if (to_s(dresult) = 0) then
02807         dresult_not_zero := false;
02808     else
02809         dresult_not_zero := true;
02810     end if;
02811     if to_x01(l(l'high)) = '1' and to_x01(r(r'high)) = '0'
02812         and dresult_not_zero then
02813         result := resize (arg      => r - dresult,
02814                           left_index => result'high,
02815                           right_index => result'low,
02816                           overflow_style => overflow_style,
02817                           round_style   => round_style);
02818     elsif to_x01(l(l'high)) = '1' and to_x01(r(r'high)) = '1' then
02819         result := resize (arg      => -dresult,
02820                           left_index => result'high,
02821                           right_index => result'low,
02822                           overflow_style => overflow_style,
02823                           round_style   => round_style);
02824     elsif to_x01(l(l'high)) = '0' and to_x01(r(r'high)) = '1'
02825         and dresult_not_zero then
02826         result := resize (arg      => dresult + r,
02827                           left_index => result'high,
02828                           right_index => result'low,
02829                           overflow_style => overflow_style,
02830                           round_style   => round_style);
02831     else
02832         result := resize (arg      => dresult,
02833                           left_index => result'high,
02834                           right_index => result'low,
02835                           overflow_style => overflow_style,
02836                           round_style   => round_style);
02837     end if;
02838     return result;
02839 end function modulo;
02840
02841 -- Procedure for those who need an "accumulator" function
02842 procedure add_carry (
02843     L, R : in UNRESOLVED_ufixed;
02844     c_in : in STD_ULONGIC;
02845     result : out UNRESOLVED_ufixed;
02846     c_out : out STD_ULONGIC) is
02847     constant left_index : INTEGER := maximum(l'high, r'high)+1;
02848     constant right_index : INTEGER := mins(l'low, r'low);
02849     variable lresize, rresize : UNRESOLVED_ufixed (left_index downto right_index);
02850     variable lslv, rslv : UNSIGNED (left_index-right_index
02851                                     downto 0);
02852     variable result_slv : UNSIGNED (left_index-right_index
02853                                     downto 0);
02854     variable cx : UNSIGNED (0 downto 0); -- Carry in
02855 begin
02856     if (l'length < 1 or r'length < 1) then
02857         result := NAUU;
02858         c_out := '0';
02859     else
02860         cx (0) := c_in;
02861         lresize := resize (l, left_index, right_index);
02862         rresize := resize (r, left_index, right_index);
02863         lslv := to_uns (lresize);
02864         rslv := to_uns (rresize);
02865         result_slv := lslv + rslv + cx;
02866         c_out := result_slv(left_index);
02867         result := to_fixed(result_slv (left_index-right_index-1 downto 0),
02868                            left_index-1, right_index);
02869     end if;
02870 end procedure add_carry;
02871
02872 procedure add_carry (
02873     L, R : in UNRESOLVED_sfixed;
02874     c_in : in STD_ULONGIC;
02875     result : out UNRESOLVED_sfixed;
02876     c_out : out STD_ULONGIC) is
02877     constant left_index : INTEGER := maximum(l'high, r'high)+1;
02878     constant right_index : INTEGER := mins(l'low, r'low);
02879     variable lresize, rresize : UNRESOLVED_sfixed (left_index downto right_index);
02880     variable lslv, rslv : SIGNED (left_index-right_index
02881                                 downto 0);

```

```

02882     variable result_slv : SIGNED (left_index-right_index
02883                               downto 0);
02884     variable cx : SIGNED (1 downto 0); -- Carry in
02885 begin
02886     if (l'length < 1 or r'length < 1) then
02887         result := NASF;
02888         c_out := '0';
02889     else
02890         cx (1) := '0';
02891         cx (0) := c_in;
02892         lresize := resize (l, left_index, right_index);
02893         rresize := resize (r, left_index, right_index);
02894         lslv := to_s (lresize);
02895         rslv := to_s (rresize);
02896         result_slv := lslv + rslv + cx;
02897         c_out := result_slv(left_index);
02898         result := to_fixed(result_slv (left_index-right_index-1 downto 0),
02899                           left_index-1, right_index);
02900     end if;
02901 end procedure add_carry;
02902
02903 -- Scales the result by a power of 2. Width of input = width of output with
02904 -- the decimal point moved.
02905 function scalb (y : UNRESOLVED_ufixed; N : INTEGER)
02906     return UNRESOLVED_ufixed is
02907     variable result : UNRESOLVED_ufixed (y'high+N downto y'low+N);
02908 begin
02909     if y'length < 1 then
02910         return NAUF;
02911     else
02912         result := y;
02913         return result;
02914     end if;
02915 end function scalb;
02916
02917 function scalb (y : UNRESOLVED_ufixed; N : SIGNED)
02918     return UNRESOLVED_ufixed is
02919 begin
02920     return scalb (y => y,
02921                   N => to_integer(N));
02922 end function scalb;
02923
02924 function scalb (y : UNRESOLVED_sfixed; N : INTEGER)
02925     return UNRESOLVED_sfixed is
02926     variable result : UNRESOLVED_sfixed (y'high+N downto y'low+N);
02927 begin
02928     if y'length < 1 then
02929         return NASF;
02930     else
02931         result := y;
02932         return result;
02933     end if;
02934 end function scalb;
02935
02936 function scalb (y : UNRESOLVED_sfixed; N : SIGNED)
02937     return UNRESOLVED_sfixed is
02938 begin
02939     return scalb (y => y,
02940                   N => to_integer(N));
02941 end function scalb;
02942
02943 function Is_Negative (arg : UNRESOLVED_sfixed) return BOOLEAN is
02944 begin
02945     if to_X01(arg(arg'high)) = '1' then
02946         return true;
02947     else
02948         return false;
02949     end if;
02950 end function Is_Negative;
02951
02952 function find_rightmost (arg : UNRESOLVED_ufixed; y : STD_ULOGIC)
02953     return INTEGER is
02954 begin
02955     for_loop : for i in arg'reverse_range loop
02956         if \?=\ (arg(i), y) = '1' then
02957             return i;
02958         end if;
02959     end loop;
02960     return arg'high+1;           -- return out of bounds 'high
02961 end function find_rightmost;
02962
02963 function find_leftmost (arg : UNRESOLVED_ufixed; y : STD_ULOGIC)
02964     return INTEGER is
02965 begin
02966     for_loop : for i in arg'range loop
02967         if \?=\ (arg(i), y) = '1' then
02968             return i;

```

```

02969      end if;
02970      end loop;
02971      return arg'low-1;                                -- return out of bounds 'low
02972  end function find_leftmost;
02973
02974  function find_rightmost (arg : UNRESOLVED_sfixed; y : STD_ULONGIC)
02975    return INTEGER is
02976  begin
02977    for_loop : for i in arg'reverse_range loop
02978      if \?=\ (arg(i), y) = '1' then
02979        return i;
02980      end if;
02981    end loop;
02982    return arg'high+1;                                -- return out of bounds 'high
02983  end function find_rightmost;
02984
02985  function find_leftmost (arg : UNRESOLVED_sfixed; y : STD_ULONGIC)
02986    return INTEGER is
02987  begin
02988    for_loop : for i in arg'range loop
02989      if \?=\ (arg(i), y) = '1' then
02990        return i;
02991      end if;
02992    end loop;
02993    return arg'low-1;                                -- return out of bounds 'low
02994  end function find_leftmost;
02995
02996  function "sll" (ARG : UNRESOLVED_ufixed; COUNT : INTEGER)
02997    return UNRESOLVED_ufixed is
02998    variable argslv : UNSIGNED (arg'length-1 downto 0);
02999    variable result : UNRESOLVED_ufixed (arg'range);
03000  begin
03001    argslv := to_uns (arg);
03002    argslv := argslv sll COUNT;
03003    result := to_fixed (argslv, result'high, result'low);
03004    return result;
03005  end function "sll";
03006
03007  function "srl" (ARG : UNRESOLVED_ufixed; COUNT : INTEGER)
03008    return UNRESOLVED_ufixed is
03009    variable argslv : UNSIGNED (arg'length-1 downto 0);
03010    variable result : UNRESOLVED_ufixed (arg'range);
03011  begin
03012    argslv := to_uns (arg);
03013    argslv := argslv srl COUNT;
03014    result := to_fixed (argslv, result'high, result'low);
03015    return result;
03016  end function "srl";
03017
03018  function "rol" (ARG : UNRESOLVED_ufixed; COUNT : INTEGER)
03019    return UNRESOLVED_ufixed is
03020    variable argslv : UNSIGNED (arg'length-1 downto 0);
03021    variable result : UNRESOLVED_ufixed (arg'range);
03022  begin
03023    argslv := to_uns (arg);
03024    argslv := argslv rol COUNT;
03025    result := to_fixed (argslv, result'high, result'low);
03026    return result;
03027  end function "rol";
03028
03029  function "ror" (ARG : UNRESOLVED_ufixed; COUNT : INTEGER)
03030    return UNRESOLVED_ufixed is
03031    variable argslv : UNSIGNED (arg'length-1 downto 0);
03032    variable result : UNRESOLVED_ufixed (arg'range);
03033  begin
03034    argslv := to_uns (arg);
03035    argslv := argslv ror COUNT;
03036    result := to_fixed (argslv, result'high, result'low);
03037    return result;
03038  end function "ror";
03039
03040  function "sla" (ARG : UNRESOLVED_ufixed; COUNT : INTEGER)
03041    return UNRESOLVED_ufixed is
03042    variable argslv : UNSIGNED (arg'length-1 downto 0);
03043    variable result : UNRESOLVED_ufixed (arg'range);
03044  begin
03045    argslv := to_uns (arg);
03046    -- Arithmetic shift on an unsigned is a logical shift
03047    argslv := argslv sll COUNT;
03048    result := to_fixed (argslv, result'high, result'low);
03049    return result;
03050  end function "sla";
03051
03052  function "sra" (ARG : UNRESOLVED_ufixed; COUNT : INTEGER)
03053    return UNRESOLVED_ufixed is
03054    variable argslv : UNSIGNED (arg'length-1 downto 0);
03055    variable result : UNRESOLVED_ufixed (arg'range);

```

```

03056    begin
03057        argslv := to_uns (arg);
03058        -- Arithmetic shift on an unsigned is a logical shift
03059        argslv := argslv srl COUNT;
03060        result := to_fixed (argslv, result'high, result'low);
03061        return result;
03062    end function "sra";
03063
03064    function "sll" (ARG : UNRESOLVED_sfixed; COUNT : INTEGER)
03065        return UNRESOLVED_sfixed is
03066        variable argslv : SIGNED (arg'length-1 downto 0);
03067        variable result : UNRESOLVED_sfixed (arg'range);
03068    begin
03069        argslv := to_s (arg);
03070        argslv := argslv sll COUNT;
03071        result := to_fixed (argslv, result'high, result'low);
03072        return result;
03073    end function "sll";
03074
03075    function "srl" (ARG : UNRESOLVED_sfixed; COUNT : INTEGER)
03076        return UNRESOLVED_sfixed is
03077        variable argslv : SIGNED (arg'length-1 downto 0);
03078        variable result : UNRESOLVED_sfixed (arg'range);
03079    begin
03080        argslv := to_s (arg);
03081        argslv := argslv srl COUNT;
03082        result := to_fixed (argslv, result'high, result'low);
03083        return result;
03084    end function "srl";
03085
03086    function "rol" (ARG : UNRESOLVED_sfixed; COUNT : INTEGER)
03087        return UNRESOLVED_sfixed is
03088        variable argslv : SIGNED (arg'length-1 downto 0);
03089        variable result : UNRESOLVED_sfixed (arg'range);
03090    begin
03091        argslv := to_s (arg);
03092        argslv := argslv rol COUNT;
03093        result := to_fixed (argslv, result'high, result'low);
03094        return result;
03095    end function "rol";
03096
03097    function "ror" (ARG : UNRESOLVED_sfixed; COUNT : INTEGER)
03098        return UNRESOLVED_sfixed is
03099        variable argslv : SIGNED (arg'length-1 downto 0);
03100        variable result : UNRESOLVED_sfixed (arg'range);
03101    begin
03102        argslv := to_s (arg);
03103        argslv := argslv ror COUNT;
03104        result := to_fixed (argslv, result'high, result'low);
03105        return result;
03106    end function "ror";
03107
03108    function "sla" (ARG : UNRESOLVED_sfixed; COUNT : INTEGER)
03109        return UNRESOLVED_sfixed is
03110        variable argslv : SIGNED (arg'length-1 downto 0);
03111        variable result : UNRESOLVED_sfixed (arg'range);
03112    begin
03113        argslv := to_s (arg);
03114        if COUNT > 0 then
03115            -- Arithmetic shift left on a 2's complement number is a logic shift
03116            argslv := argslv sll COUNT;
03117        else
03118            argslv := argslv sra -COUNT;
03119        end if;
03120        result := to_fixed (argslv, result'high, result'low);
03121        return result;
03122    end function "sla";
03123
03124    function "sra" (ARG : UNRESOLVED_sfixed; COUNT : INTEGER)
03125        return UNRESOLVED_sfixed is
03126        variable argslv : SIGNED (arg'length-1 downto 0);
03127        variable result : UNRESOLVED_sfixed (arg'range);
03128    begin
03129        argslv := to_s (arg);
03130        if COUNT > 0 then
03131            argslv := argslv sra COUNT;
03132        else
03133            -- Arithmetic shift left on a 2's complement number is a logic shift
03134            argslv := argslv sll -COUNT;
03135        end if;
03136        result := to_fixed (argslv, result'high, result'low);
03137        return result;
03138    end function "sra";
03139
03140    -- Because some people want the older functions.
03141    function SHIFT_LEFT (ARG : UNRESOLVED_ufixed; COUNT : NATURAL)
03142        return UNRESOLVED_ufixed is

```

```

03143 begin
03144   if (ARG'length < 1) then
03145     return NAUF;
03146   end if;
03147   return ARG sla COUNT;
03148 end function SHIFT_LEFT;
03149
03150 function SHIFT_RIGHT (ARG : UNRESOLVED_ufixed; COUNT : NATURAL)
03151   return UNRESOLVED_ufixed is
03152 begin
03153   if (ARG'length < 1) then
03154     return NAUF;
03155   end if;
03156   return ARG sra COUNT;
03157 end function SHIFT_RIGHT;
03158
03159 function SHIFT_LEFT (ARG : UNRESOLVED_sfixed; COUNT : NATURAL)
03160   return UNRESOLVED_sfixed is
03161 begin
03162   if (ARG'length < 1) then
03163     return NASF;
03164   end if;
03165   return ARG sla COUNT;
03166 end function SHIFT_LEFT;
03167
03168 function SHIFT_RIGHT (ARG : UNRESOLVED_sfixed; COUNT : NATURAL)
03169   return UNRESOLVED_sfixed is
03170 begin
03171   if (ARG'length < 1) then
03172     return NASF;
03173   end if;
03174   return ARG sra COUNT;
03175 end function SHIFT_RIGHT;
03176
03177 -----
03178 -- logical functions
03179 -----
03180 function "not" (L : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed is
03181   variable RESULT : STD_ULOGIC_VECTOR(L'length-1 downto 0); -- force downto
03182 begin
03183   RESULT := not to_sulv(L);
03184   return to_ufixed(RESULT, L'high, L'low);
03185 end function "not";
03186
03187 function "and" (L, R : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed is
03188   variable RESULT : STD_ULOGIC_VECTOR(L'length-1 downto 0); -- force downto
03189 begin
03190   if (L'high = R'high and L'low = R'low) then
03191     RESULT := to_sulv(L) and to_sulv(R);
03192   else
03193     assert NO_WARNING
03194       report fixed_pkg'instance_name
03195         & """and"": Range error L'RANGE /= R'RANGE"
03196         severity warning;
03197     RESULT := (others => 'X');
03198   end if;
03199   return to_ufixed(RESULT, L'high, L'low);
03200 end function "and";
03201
03202 function "or" (L, R : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed is
03203   variable RESULT : STD_ULOGIC_VECTOR(L'length-1 downto 0); -- force downto
03204 begin
03205   if (L'high = R'high and L'low = R'low) then
03206     RESULT := to_sulv(L) or to_sulv(R);
03207   else
03208     assert NO_WARNING
03209       report fixed_pkg'instance_name
03210         & """or"": Range error L'RANGE /= R'RANGE"
03211         severity warning;
03212     RESULT := (others => 'X');
03213   end if;
03214   return to_ufixed(RESULT, L'high, L'low);
03215 end function "or";
03216
03217 function "nand" (L, R : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed is
03218   variable RESULT : STD_ULOGIC_VECTOR(L'length-1 downto 0); -- force downto
03219 begin
03220   if (L'high = R'high and L'low = R'low) then
03221     RESULT := to_sulv(L) nand to_sulv(R);
03222   else
03223     assert NO_WARNING
03224       report fixed_pkg'instance_name
03225         & """nand"": Range error L'RANGE /= R'RANGE"

```

```

03226      severity warning;
03227      RESULT := (others => 'X');
03228  end if;
03229  return to_ufixed(RESULT, L'high, L'low);
03230 end function "nand";
03231
03232 function "nor" (L, R : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed is
03233  variable RESULT : STD_ULONGIC_VECTOR(L'length-1 downto 0); -- force downto
03234 begin
03235  if (L'high = R'high and L'low = R'low) then
03236    RESULT := to_sulv(L) nor to_sulv(R);
03237  else
03238    assert NO_WARNING
03239      report fixed_pkg'instance_name
03240      & """nor"": Range error L'RANGE /= R'RANGE"
03241      severity warning;
03242    RESULT := (others => 'X');
03243  end if;
03244  return to_ufixed(RESULT, L'high, L'low);
03245 end function "nor";
03246
03247 function "xor" (L, R : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed is
03248  variable RESULT : STD_ULONGIC_VECTOR(L'length-1 downto 0); -- force downto
03249 begin
03250  if (L'high = R'high and L'low = R'low) then
03251    RESULT := to_sulv(L) xor to_sulv(R);
03252  else
03253    assert NO_WARNING
03254      report fixed_pkg'instance_name
03255      & """xor"": Range error L'RANGE /= R'RANGE"
03256      severity warning;
03257    RESULT := (others => 'X');
03258  end if;
03259  return to_ufixed(RESULT, L'high, L'low);
03260 end function "xor";
03261
03262 function "xnor" (L, R : UNRESOLVED_ufixed) return
UNRESOLVED_ufixed is
03263  variable RESULT : STD_ULONGIC_VECTOR(L'length-1 downto 0); -- force downto
03264 begin
03265  if (L'high = R'high and L'low = R'low) then
03266    RESULT := to_sulv(L) xnor to_sulv(R);
03267  else
03268    assert NO_WARNING
03269      report fixed_pkg'instance_name
03270      & """xnor"": Range error L'RANGE /= R'RANGE"
03271      severity warning;
03272    RESULT := (others => 'X');
03273  end if;
03274  return to_ufixed(RESULT, L'high, L'low);
03275 end function "xnor";
03276
03277 function "not" (L : UNRESOLVED_sfixed) return
UNRESOLVED_sfixed is
03278  variable RESULT : STD_ULONGIC_VECTOR(L'length-1 downto 0); -- force downto
03279 begin
03280  RESULT := not to_sulv(L);
03281  return to_sfixed(RESULT, L'high, L'low);
03282 end function "not";
03283
03284 function "and" (L, R : UNRESOLVED_sfixed) return
UNRESOLVED_sfixed is
03285  variable RESULT : STD_ULONGIC_VECTOR(L'length-1 downto 0); -- force downto
03286 begin
03287  if (L'high = R'high and L'low = R'low) then
03288    RESULT := to_sulv(L) and to_sulv(R);
03289  else
03290    assert NO_WARNING
03291      report fixed_pkg'instance_name
03292      & """and"": Range error L'RANGE /= R'RANGE"
03293      severity warning;
03294    RESULT := (others => 'X');
03295  end if;
03296  return to_sfixed(RESULT, L'high, L'low);
03297 end function "and";
03298
03299 function "or" (L, R : UNRESOLVED_sfixed) return
UNRESOLVED_sfixed is
03300  variable RESULT : STD_ULONGIC_VECTOR(L'length-1 downto 0); -- force downto
03301 begin
03302  if (L'high = R'high and L'low = R'low) then
03303    RESULT := to_sulv(L) or to_sulv(R);
03304  else
03305    assert NO_WARNING
03306      report fixed_pkg'instance_name

```

```

03307      & """or"": Range error L'RANGE /= R'RANGE"
03308      severity warning;
03309      RESULT := (others => 'X');
03310      end if;
03311      return to_sfixed(RESULT, L'high, L'low);
03312  end function "or";
03313
03314  function "nand" (L, R : UNRESOLVED_sfixed) return
UNRESOLVED_sfixed is
03315      variable RESULT : STD_ULOGIC_VECTOR(L'length-1 downto 0); -- force downto
03316  begin
03317      if (L'high = R'high and L'low = R'low) then
03318          RESULT := to_sulv(L) nand to_sulv(R);
03319      else
03320          assert NO_WARNING
03321              report fixed_pkg'instance_name
03322              & """nand"": Range error L'RANGE /= R'RANGE"
03323              severity warning;
03324          RESULT := (others => 'X');
03325      end if;
03326      return to_sfixed(RESULT, L'high, L'low);
03327  end function "nand";
03328
03329  function "nor" (L, R : UNRESOLVED_sfixed) return
UNRESOLVED_sfixed is
03330      variable RESULT : STD_ULOGIC_VECTOR(L'length-1 downto 0); -- force downto
03331  begin
03332      if (L'high = R'high and L'low = R'low) then
03333          RESULT := to_sulv(L) nor to_sulv(R);
03334      else
03335          assert NO_WARNING
03336              report fixed_pkg'instance_name
03337              & """nor"": Range error L'RANGE /= R'RANGE"
03338              severity warning;
03339          RESULT := (others => 'X');
03340      end if;
03341      return to_sfixed(RESULT, L'high, L'low);
03342  end function "nor";
03343
03344  function "xor" (L, R : UNRESOLVED_sfixed) return
UNRESOLVED_sfixed is
03345      variable RESULT : STD_ULOGIC_VECTOR(L'length-1 downto 0); -- force downto
03346  begin
03347      if (L'high = R'high and L'low = R'low) then
03348          RESULT := to_sulv(L) xor to_sulv(R);
03349      else
03350          assert NO_WARNING
03351              report fixed_pkg'instance_name
03352              & """xor"": Range error L'RANGE /= R'RANGE"
03353              severity warning;
03354          RESULT := (others => 'X');
03355      end if;
03356      return to_sfixed(RESULT, L'high, L'low);
03357  end function "xor";
03358
03359  function "xnor" (L, R : UNRESOLVED_sfixed) return
UNRESOLVED_sfixed is
03360      variable RESULT : STD_ULOGIC_VECTOR(L'length-1 downto 0); -- force downto
03361  begin
03362      if (L'high = R'high and L'low = R'low) then
03363          RESULT := to_sulv(L) xnor to_sulv(R);
03364      else
03365          assert NO_WARNING
03366              report fixed_pkg'instance_name
03367              & """xnor"": Range error L'RANGE /= R'RANGE"
03368              severity warning;
03369          RESULT := (others => 'X');
03370      end if;
03371      return to_sfixed(RESULT, L'high, L'low);
03372  end function "xnor";
03373
03374  -- Vector and std_ulogic functions, same as functions in numeric_std
03375  function "and" (L : STD_ULOGIC; R : UNRESOLVED_ufixed)
03376      return UNRESOLVED_ufixed is
03377      variable result : UNRESOLVED_ufixed (R'range);
03378  begin
03379      for i in result'range loop
03380          result(i) := L and R(i);
03381      end loop;
03382      return result;
03383  end function "and";
03384
03385  function "and" (L : UNRESOLVED_ufixed; R : STD_ULOGIC)
03386      return UNRESOLVED_ufixed is
03387      variable result : UNRESOLVED_ufixed (L'range);
03388  begin
03389      for i in result'range loop

```

```
03390     result(i) := L(i) and R;
03391   end loop;
03392   return result;
03393 end function "and";
03394
03395 function "or" (L : STD_ULONGIC; R : UNRESOLVED_ufixed)
03396   return UNRESOLVED_ufixed is
03397   variable result : UNRESOLVED_ufixed (R'range);
03398 begin
03399   for i in result'range loop
03400     result(i) := L or R(i);
03401   end loop;
03402   return result;
03403 end function "or";
03404
03405 function "or" (L : UNRESOLVED_ufixed; R : STD_ULONGIC)
03406   return UNRESOLVED_ufixed is
03407   variable result : UNRESOLVED_ufixed (L'range);
03408 begin
03409   for i in result'range loop
03410     result(i) := L(i) or R;
03411   end loop;
03412   return result;
03413 end function "or";
03414
03415 function "nand" (L : STD_ULONGIC; R : UNRESOLVED_ufixed)
03416   return UNRESOLVED_ufixed is
03417   variable result : UNRESOLVED_ufixed (R'range);
03418 begin
03419   for i in result'range loop
03420     result(i) := L nand R(i);
03421   end loop;
03422   return result;
03423 end function "nand";
03424
03425 function "nand" (L : UNRESOLVED_ufixed; R : STD_ULONGIC)
03426   return UNRESOLVED_ufixed is
03427   variable result : UNRESOLVED_ufixed (L'range);
03428 begin
03429   for i in result'range loop
03430     result(i) := L(i) nand R;
03431   end loop;
03432   return result;
03433 end function "nand";
03434
03435 function "nor" (L : STD_ULONGIC; R : UNRESOLVED_ufixed)
03436   return UNRESOLVED_ufixed is
03437   variable result : UNRESOLVED_ufixed (R'range);
03438 begin
03439   for i in result'range loop
03440     result(i) := L nor R(i);
03441   end loop;
03442   return result;
03443 end function "nor";
03444
03445 function "nor" (L : UNRESOLVED_ufixed; R : STD_ULONGIC)
03446   return UNRESOLVED_ufixed is
03447   variable result : UNRESOLVED_ufixed (L'range);
03448 begin
03449   for i in result'range loop
03450     result(i) := L(i) nor R;
03451   end loop;
03452   return result;
03453 end function "nor";
03454
03455 function "xor" (L : STD_ULONGIC; R : UNRESOLVED_ufixed)
03456   return UNRESOLVED_ufixed is
03457   variable result : UNRESOLVED_ufixed (R'range);
03458 begin
03459   for i in result'range loop
03460     result(i) := L xor R(i);
03461   end loop;
03462   return result;
03463 end function "xor";
03464
03465 function "xor" (L : UNRESOLVED_ufixed; R : STD_ULONGIC)
03466   return UNRESOLVED_ufixed is
03467   variable result : UNRESOLVED_ufixed (L'range);
03468 begin
03469   for i in result'range loop
03470     result(i) := L(i) xor R;
03471   end loop;
03472   return result;
03473 end function "xor";
03474
03475 function "xnor" (L : STD_ULONGIC; R : UNRESOLVED_ufixed)
03476   return UNRESOLVED_ufixed is
```

```

03477     variable result : UNRESOLVED_ufixed (R'range);
03478 begin
03479   for i in result'range loop
03480     result(i) := L xnor R(i);
03481   end loop;
03482   return result;
03483 end function "xnor";
03484
03485 function "xnor" (L : UNRESOLVED_ufixed; R : STD_ULONGIC)
03486   return UNRESOLVED_ufixed is
03487   variable result : UNRESOLVED_ufixed (L'range);
03488 begin
03489   for i in result'range loop
03490     result(i) := L(i) xnor R;
03491   end loop;
03492   return result;
03493 end function "xnor";
03494
03495 function "and" (L : STD_ULONGIC; R : UNRESOLVED_sfixed)
03496   return UNRESOLVED_sfixed is
03497   variable result : UNRESOLVED_sfixed (R'range);
03498 begin
03499   for i in result'range loop
03500     result(i) := L and R(i);
03501   end loop;
03502   return result;
03503 end function "and";
03504
03505 function "and" (L : UNRESOLVED_sfixed; R : STD_ULONGIC)
03506   return UNRESOLVED_sfixed is
03507   variable result : UNRESOLVED_sfixed (L'range);
03508 begin
03509   for i in result'range loop
03510     result(i) := L(i) and R;
03511   end loop;
03512   return result;
03513 end function "and";
03514
03515 function "or" (L : STD_ULONGIC; R : UNRESOLVED_sfixed)
03516   return UNRESOLVED_sfixed is
03517   variable result : UNRESOLVED_sfixed (R'range);
03518 begin
03519   for i in result'range loop
03520     result(i) := L or R(i);
03521   end loop;
03522   return result;
03523 end function "or";
03524
03525 function "or" (L : UNRESOLVED_sfixed; R : STD_ULONGIC)
03526   return UNRESOLVED_sfixed is
03527   variable result : UNRESOLVED_sfixed (L'range);
03528 begin
03529   for i in result'range loop
03530     result(i) := L(i) or R;
03531   end loop;
03532   return result;
03533 end function "or";
03534
03535 function "nand" (L : STD_ULONGIC; R : UNRESOLVED_sfixed)
03536   return UNRESOLVED_sfixed is
03537   variable result : UNRESOLVED_sfixed (R'range);
03538 begin
03539   for i in result'range loop
03540     result(i) := L nand R(i);
03541   end loop;
03542   return result;
03543 end function "nand";
03544
03545 function "nand" (L : UNRESOLVED_sfixed; R : STD_ULONGIC)
03546   return UNRESOLVED_sfixed is
03547   variable result : UNRESOLVED_sfixed (L'range);
03548 begin
03549   for i in result'range loop
03550     result(i) := L(i) nand R;
03551   end loop;
03552   return result;
03553 end function "nand";
03554
03555 function "nor" (L : STD_ULONGIC; R : UNRESOLVED_sfixed)
03556   return UNRESOLVED_sfixed is
03557   variable result : UNRESOLVED_sfixed (R'range);
03558 begin
03559   for i in result'range loop
03560     result(i) := L nor R(i);
03561   end loop;
03562   return result;
03563 end function "nor";

```

```

03564
03565   function "nor" (L : UNRESOLVED_sfixed; R : STD_ULOGIC)
03566     return UNRESOLVED_sfixed is
03567       variable result : UNRESOLVED_sfixed (L'range);
03568   begin
03569     for i in result'range loop
03570       result(i) := L(i) nor R;
03571     end loop;
03572     return result;
03573   end function "nor";
03574
03575   function "xor" (L : STD_ULOGIC; R : UNRESOLVED_sfixed)
03576     return UNRESOLVED_sfixed is
03577       variable result : UNRESOLVED_sfixed (R'range);
03578   begin
03579     for i in result'range loop
03580       result(i) := L xor R(i);
03581     end loop;
03582     return result;
03583   end function "xor";
03584
03585   function "xor" (L : UNRESOLVED_sfixed; R : STD_ULOGIC)
03586     return UNRESOLVED_sfixed is
03587       variable result : UNRESOLVED_sfixed (L'range);
03588   begin
03589     for i in result'range loop
03590       result(i) := L(i) xor R;
03591     end loop;
03592     return result;
03593   end function "xor";
03594
03595   function "xnor" (L : STD_ULOGIC; R : UNRESOLVED_sfixed)
03596     return UNRESOLVED_sfixed is
03597       variable result : UNRESOLVED_sfixed (R'range);
03598   begin
03599     for i in result'range loop
03600       result(i) := L xnor R(i);
03601     end loop;
03602     return result;
03603   end function "xnor";
03604
03605   function "xnor" (L : UNRESOLVED_sfixed; R : STD_ULOGIC)
03606     return UNRESOLVED_sfixed is
03607       variable result : UNRESOLVED_sfixed (L'range);
03608   begin
03609     for i in result'range loop
03610       result(i) := L(i) xnor R;
03611     end loop;
03612     return result;
03613   end function "xnor";
03614
03615   -- Reduction operator_reduces
03616   function and_reduce (l : UNRESOLVED_ufixed) return STD_ULOGIC is
03617   begin
03618     return and_reduce (to_sulv(l));
03619   end function and_reduce;
03620
03621   function nand_reduce (l : UNRESOLVED_ufixed) return STD_ULOGIC is
03622   begin
03623     return nand_reduce (to_sulv(l));
03624   end function nand_reduce;
03625
03626   function or_reduce (l : UNRESOLVED_ufixed) return STD_ULOGIC is
03627   begin
03628     return or_reduce (to_sulv(l));
03629   end function or_reduce;
03630
03631   function nor_reduce (l : UNRESOLVED_ufixed) return STD_ULOGIC is
03632   begin
03633     return nor_reduce (to_sulv(l));
03634   end function nor_reduce;
03635
03636   function xor_reduce (l : UNRESOLVED_ufixed) return STD_ULOGIC is
03637   begin
03638     return xor_reduce (to_sulv(l));
03639   end function xor_reduce;
03640
03641   function xnor_reduce (l : UNRESOLVED_ufixed) return STD_ULOGIC is
03642   begin
03643     return xnor_reduce (to_sulv(l));
03644   end function xnor_reduce;
03645
03646   function and_reduce (l : UNRESOLVED_sfixed) return STD_ULOGIC is
03647   begin
03648     return and_reduce (to_sulv(l));
03649   end function and_reduce;
03650

```

```

03651  function nand_reduce (l : UNRESOLVED_sfixed) return STD_ULONGIC is
03652  begin
03653      return nand_reduce (to_sulv(l));
03654  end function nand_reduce;
03655
03656  function or_reduce (l : UNRESOLVED_sfixed) return STD_ULONGIC is
03657  begin
03658      return or_reduce (to_sulv(l));
03659  end function or_reduce;
03660
03661  function nor_reduce (l : UNRESOLVED_sfixed) return STD_ULONGIC is
03662  begin
03663      return nor_reduce (to_sulv(l));
03664  end function nor_reduce;
03665
03666  function xor_reduce (l : UNRESOLVED_sfixed) return STD_ULONGIC is
03667  begin
03668      return xor_reduce (to_sulv(l));
03669  end function xor_reduce;
03670
03671  function xnor_reduce (l : UNRESOLVED_sfixed) return STD_ULONGIC is
03672  begin
03673      return xnor_reduce (to_sulv(l));
03674  end function xnor_reduce;
03675  -- End reduction operator_reduces
03676
03677  function \?=\ (L, R : UNRESOLVED_ufixed) return STD_ULONGIC is
03678  constant left_index      : INTEGER := maximum(l'high, r'high);
03679  constant right_index     : INTEGER := mins(l'low, r'low);
03680  variable lresize, rresize : UNRESOLVED_ufixed (left_index downto right_index);
03681  variable lslv, rslv      : UNSIGNED (lresize'length-1 downto 0);
03682  begin -- ?=
03683      if ((L'length < 1) or (R'length < 1)) then
03684          assert NO_WARNING
03685              report fixed_pkg'instance_name
03686              & """?=: null detected, returning X"
03687              severity warning;
03688          return 'X';
03689      else
03690          lresize := resize (l, left_index, right_index);
03691          rresize := resize (r, left_index, right_index);
03692          lslv   := to_uns (lresize);
03693          rslv   := to_uns (rresize);
03694          return \?=\ (lslv, rslv);
03695      end if;
03696  end function \?=\;
03697
03698  function \?/= (L, R : UNRESOLVED_ufixed) return STD_ULONGIC is
03699  constant left_index      : INTEGER := maximum(l'high, r'high);
03700  constant right_index     : INTEGER := mins(l'low, r'low);
03701  variable lresize, rresize : UNRESOLVED_ufixed (left_index downto right_index);
03702  variable lslv, rslv      : UNSIGNED (lresize'length-1 downto 0);
03703  begin -- ?/
03704      if ((L'length < 1) or (R'length < 1)) then
03705          assert NO_WARNING
03706              report fixed_pkg'instance_name
03707              & """?/=": null detected, returning X"
03708              severity warning;
03709          return 'X';
03710      else
03711          lresize := resize (l, left_index, right_index);
03712          rresize := resize (r, left_index, right_index);
03713          lslv   := to_uns (lresize);
03714          rslv   := to_uns (rresize);
03715          return \?/= (lslv, rslv);
03716      end if;
03717  end function \?/=;
03718
03719  function \?>\ (L, R : UNRESOLVED_ufixed) return STD_ULONGIC is
03720  constant left_index      : INTEGER := maximum(l'high, r'high);
03721  constant right_index     : INTEGER := mins(l'low, r'low);
03722  variable lresize, rresize : UNRESOLVED_ufixed (left_index downto right_index);
03723  variable lslv, rslv      : UNSIGNED (lresize'length-1 downto 0);
03724  begin -- ?>
03725      if ((l'length < 1) or (r'length < 1)) then
03726          assert NO_WARNING
03727              report fixed_pkg'instance_name
03728              & """?>": null detected, returning X"
03729              severity warning;
03730          return 'X';
03731      else
03732          lresize := resize (l, left_index, right_index);
03733          rresize := resize (r, left_index, right_index);
03734          lslv   := to_uns (lresize);
03735          rslv   := to_uns (rresize);
03736          return \?>\ (lslv, rslv);
03737      end if;

```

```

03738   end function \?>\;
03739
03740   function \?>=\ (L, R : UNRESOLVED_ufixed) return STD_ULOGIC is
03741     constant left_index      : INTEGER := maximum(l'high, r'high);
03742     constant right_index     : INTEGER := mins(l'low, r'low);
03743     variable lresize, rresize : UNRESOLVED_ufixed (left_index downto right_index);
03744     variable lslv, rslv      : UNSIGNED (lresize'length-1 downto 0);
03745 begin -- ?>=
03746   if ((l'length < 1) or (r'length < 1)) then
03747     assert NO_WARNING
03748       report fixed_pkg'instance_name
03749       & """?>=": null detected, returning X"
03750       severity warning;
03751     return 'X';
03752   else
03753     lresize := resize (l, left_index, right_index);
03754     rresize := resize (r, left_index, right_index);
03755     lslv    := to_uns (lresize);
03756     rslv    := to_uns (rresize);
03757     return \?>=\ (lslv, rslv);
03758   end if;
03759 end function \?>=\
03760
03761   function \?<\ (L, R : UNRESOLVED_ufixed) return STD_ULOGIC is
03762     constant left_index      : INTEGER := maximum(l'high, r'high);
03763     constant right_index     : INTEGER := mins(l'low, r'low);
03764     variable lresize, rresize : UNRESOLVED_ufixed (left_index downto right_index);
03765     variable lslv, rslv      : UNSIGNED (lresize'length-1 downto 0);
03766 begin -- ?<=
03767   if ((l'length < 1) or (r'length < 1)) then
03768     assert NO_WARNING
03769       report fixed_pkg'instance_name
03770       & """?<"": null detected, returning X"
03771       severity warning;
03772     return 'X';
03773   else
03774     lresize := resize (l, left_index, right_index);
03775     rresize := resize (r, left_index, right_index);
03776     lslv    := to_uns (lresize);
03777     rslv    := to_uns (rresize);
03778     return \?<\ (lslv, rslv);
03779   end if;
03780 end function \?<\
03781
03782   function \?<=\ (L, R : UNRESOLVED_ufixed) return STD_ULOGIC is
03783     constant left_index      : INTEGER := maximum(l'high, r'high);
03784     constant right_index     : INTEGER := mins(l'low, r'low);
03785     variable lresize, rresize : UNRESOLVED_ufixed (left_index downto right_index);
03786     variable lslv, rslv      : UNSIGNED (lresize'length-1 downto 0);
03787 begin -- ?<=
03788   if ((l'length < 1) or (r'length < 1)) then
03789     assert NO_WARNING
03790       report fixed_pkg'instance_name
03791       & """?<=": null detected, returning X"
03792       severity warning;
03793     return 'X';
03794   else
03795     lresize := resize (l, left_index, right_index);
03796     rresize := resize (r, left_index, right_index);
03797     lslv    := to_uns (lresize);
03798     rslv    := to_uns (rresize);
03799     return \?<=\ (lslv, rslv);
03800   end if;
03801 end function \?<=\
03802
03803   function \?=\< (L, R : UNRESOLVED_sfixed) return STD_ULOGIC is
03804     constant left_index      : INTEGER := maximum(l'high, r'high);
03805     constant right_index     : INTEGER := mins(l'low, r'low);
03806     variable lresize, rresize : UNRESOLVED_sfixed (left_index downto right_index);
03807     variable lslv, rslv      : SIGNED (lresize'length-1 downto 0);
03808 begin -- ?=
03809   if ((L'length < 1) or (R'length < 1)) then
03810     assert NO_WARNING
03811       report fixed_pkg'instance_name
03812       & """?=\<": null detected, returning X"
03813       severity warning;
03814     return 'X';
03815   else
03816     lresize := resize (l, left_index, right_index);
03817     rresize := resize (r, left_index, right_index);
03818     lslv    := to_s (lresize);
03819     rslv    := to_s (rresize);
03820     return \?=\< (lslv, rslv);
03821   end if;
03822 end function \?=\
03823
03824   function \?/=\ (L, R : UNRESOLVED_sfixed) return STD_ULOGIC is

```

```

03825  constant left_index      : INTEGER := maximum(l'high, r'high);
03826  constant right_index     : INTEGER := mins(l'low, r'low);
03827  variable lresize, rresize : UNRESOLVED_sfixed (left_index downto right_index);
03828  variable lslv, rslv      : SIGNED (lresize'length-1 downto 0);
03829 begin -- ?/
03830  if ((l'length < 1) or (r'length < 1)) then
03831    assert NO_WARNING
03832      report fixed_pkg'instance_name
03833      & """?/: null detected, returning X"
03834      severity warning;
03835    return 'X';
03836  else
03837    lresize := resize (l, left_index, right_index);
03838    rresize := resize (r, left_index, right_index);
03839    lslv   := to_s (lresize);
03840    rslv   := to_s (rresize);
03841    return \?=/\ (lslv, rslv);
03842  end if;
03843 end function \?=/\;
03844
03845 function \?>\ (L, R : UNRESOLVED_sfixed) return STD_ULONGIC is
03846  constant left_index      : INTEGER := maximum(l'high, r'high);
03847  constant right_index     : INTEGER := mins(l'low, r'low);
03848  variable lresize, rresize : UNRESOLVED_sfixed (left_index downto right_index);
03849  variable lslv, rslv      : SIGNED (lresize'length-1 downto 0);
03850 begin -- ?>
03851  if ((l'length < 1) or (r'length < 1)) then
03852    assert NO_WARNING
03853      report fixed_pkg'instance_name
03854      & """?>": null detected, returning X"
03855      severity warning;
03856    return 'X';
03857  else
03858    lresize := resize (l, left_index, right_index);
03859    rresize := resize (r, left_index, right_index);
03860    lslv   := to_s (lresize);
03861    rslv   := to_s (rresize);
03862    return \?>\ (lslv, rslv);
03863  end if;
03864 end function \?>|;
03865
03866 function \?>=\ (L, R : UNRESOLVED_sfixed) return STD_ULONGIC is
03867  constant left_index      : INTEGER := maximum(l'high, r'high);
03868  constant right_index     : INTEGER := mins(l'low, r'low);
03869  variable lresize, rresize : UNRESOLVED_sfixed (left_index downto right_index);
03870  variable lslv, rslv      : SIGNED (lresize'length-1 downto 0);
03871 begin -- ?>=
03872  if ((l'length < 1) or (r'length < 1)) then
03873    assert NO_WARNING
03874      report fixed_pkg'instance_name
03875      & """?>=: null detected, returning X"
03876      severity warning;
03877    return 'X';
03878  else
03879    lresize := resize (l, left_index, right_index);
03880    rresize := resize (r, left_index, right_index);
03881    lslv   := to_s (lresize);
03882    rslv   := to_s (rresize);
03883    return \?>=\ (lslv, rslv);
03884  end if;
03885 end function \?>=|;
03886
03887 function \?<\ (L, R : UNRESOLVED_sfixed) return STD_ULONGIC is
03888  constant left_index      : INTEGER := maximum(l'high, r'high);
03889  constant right_index     : INTEGER := mins(l'low, r'low);
03890  variable lresize, rresize : UNRESOLVED_sfixed (left_index downto right_index);
03891  variable lslv, rslv      : SIGNED (lresize'length-1 downto 0);
03892 begin -- ?<
03893  if ((l'length < 1) or (r'length < 1)) then
03894    assert NO_WARNING
03895      report fixed_pkg'instance_name
03896      & """?<": null detected, returning X"
03897      severity warning;
03898    return 'X';
03899  else
03900    lresize := resize (l, left_index, right_index);
03901    rresize := resize (r, left_index, right_index);
03902    lslv   := to_s (lresize);
03903    rslv   := to_s (rresize);
03904    return \?<\ (lslv, rslv);
03905  end if;
03906 end function \?<|;
03907
03908 function \?<=\ (L, R : UNRESOLVED_sfixed) return STD_ULONGIC is
03909  constant left_index      : INTEGER := maximum(l'high, r'high);
03910  constant right_index     : INTEGER := mins(l'low, r'low);
03911  variable lresize, rresize : UNRESOLVED_sfixed (left_index downto right_index);

```

```

03912     variable lslv, rslv      : SIGNED (lresize'length-1 downto 0);
03913 begin -- ?<=
03914   if ((l'length < 1) or (r'length < 1)) then
03915     assert NO_WARNING
03916       report fixed_pkg'instance_name
03917       & """?<=": null detected, returning X"
03918       severity warning;
03919     return 'X';
03920   else
03921     lresize := resize (l, left_index, right_index);
03922     rresize := resize (r, left_index, right_index);
03923     lslv    := to_s (lresize);
03924     rslv    := to_s (rresize);
03925     return \?<=\ (lslv, rslv);
03926   end if;
03927 end function \?<=\;

03928 -- Match function, similar to "std_match" from numeric_std
03929 function std_match (L, R : UNRESOLVED_ufixed) return BOOLEAN is
03930 begin
03931   if (L'high = R'high and L'low = R'low) then
03932     return std_match(to_sulv(L), to_sulv(R));
03933   else
03934     assert NO_WARNING
03935       report fixed_pkg'instance_name
03936       & "STD_MATCH: L'RANGE /= R'RANGE, returning FALSE"
03937       severity warning;
03938     return false;
03939   end if;
03940 end function std_match;

03941 function std_match (L, R : UNRESOLVED_sfixed) return BOOLEAN is
03942 begin
03943   if (L'high = R'high and L'low = R'low) then
03944     return std_match(to_sulv(L), to_sulv(R));
03945   else
03946     assert NO_WARNING
03947       report fixed_pkg'instance_name
03948       & "STD_MATCH: L'RANGE /= R'RANGE, returning FALSE"
03949       severity warning;
03950     return false;
03951   end if;
03952 end function std_match;

03953 -- compare functions
03954 function "=" (
03955   l, r : UNRESOLVED_ufixed)           -- fixed point input
03956 begin
03957   constant left_index      : INTEGER := maximum(l'high, r'high);
03958   constant right_index     : INTEGER := mins(l'low, r'low);
03959   variable lresize, rresize : UNRESOLVED_ufixed (left_index downto right_index);
03960   variable lslv, rslv      : UNSIGNED (lresize'length-1 downto 0);
03961   if (l'length < 1 or r'length < 1) then
03962     assert NO_WARNING
03963       report fixed_pkg'instance_name
03964       & """": null argument detected, returning FALSE"
03965       severity warning;
03966     return false;
03967   elsif (Is_X(l) or Is_X(r)) then
03968     assert NO_WARNING
03969       report fixed_pkg'instance_name
03970       & """=: metavalue detected, returning FALSE"
03971       severity warning;
03972     return false;
03973   end if;
03974   lresize := resize (l, left_index, right_index);
03975   rresize := resize (r, left_index, right_index);
03976   lslv    := to_uns (lresize);
03977   rslv    := to_uns (rresize);
03978   return lslv = rslv;
03979 end function "=";

03980 function "=" (
03981   l, r : UNRESOLVED_sfixed)           -- fixed point input
03982 begin
03983   constant left_index      : INTEGER := maximum(l'high, r'high);
03984   constant right_index     : INTEGER := mins(l'low, r'low);
03985   variable lresize, rresize : UNRESOLVED_sfixed (left_index downto right_index);
03986   variable lslv, rslv      : SIGNED (lresize'length-1 downto 0);
03987   if (l'length < 1 or r'length < 1) then
03988     assert NO_WARNING
03989       report fixed_pkg'instance_name
03990       & """=: null argument detected, returning FALSE"
03991       severity warning;
03992     return false;

```

```

03999  elsif (Is_X(l) or Is_X(r)) then
04000    assert NO_WARNING
04001      report fixed_pkg'instance_name
04002      & """": metavalue detected, returning FALSE"
04003      severity warning;
04004      return false;
04005  end if;
04006  lresize := resize (l, left_index, right_index);
04007  rresize := resize (r, left_index, right_index);
04008  lslv   := to_s (lresize);
04009  rslv   := to_s (rresize);
04010  return lslv = rslv;
04011 end function "=";
04012
04013 function "/=" (
04014   l, r : UNRESOLVED_ufixed)           -- fixed point input
04015   return BOOLEAN is
04016     constant left_index      : INTEGER := maximum(l'high, r'high);
04017     constant right_index     : INTEGER := mins(l'low, r'low);
04018     variable lresize, rresize : UNRESOLVED_ufixed (left_index downto right_index);
04019     variable lslv, rslv      : UNSIGNED (lresize'length-1 downto 0);
04020 begin
04021  if (l'length < 1 or r'length < 1) then
04022    assert NO_WARNING
04023      report fixed_pkg'instance_name
04024      & """/=": null argument detected, returning TRUE"
04025      severity warning;
04026      return true;
04027  elsif (Is_X(l) or Is_X(r)) then
04028    assert NO_WARNING
04029      report fixed_pkg'instance_name
04030      & """/=": metavalue detected, returning TRUE"
04031      severity warning;
04032      return true;
04033  end if;
04034  lresize := resize (l, left_index, right_index);
04035  rresize := resize (r, left_index, right_index);
04036  lslv   := to_uns (lresize);
04037  rslv   := to_uns (rresize);
04038  return lslv /= rslv;
04039 end function "/=";
04040
04041 function "/=" (
04042   l, r : UNRESOLVED_sfixed)           -- fixed point input
04043   return BOOLEAN is
04044     constant left_index      : INTEGER := maximum(l'high, r'high);
04045     constant right_index     : INTEGER := mins(l'low, r'low);
04046     variable lresize, rresize : UNRESOLVED_sfixed (left_index downto right_index);
04047     variable lslv, rslv      : SIGNED (lresize'length-1 downto 0);
04048 begin
04049  if (l'length < 1 or r'length < 1) then
04050    assert NO_WARNING
04051      report fixed_pkg'instance_name
04052      & """/=": null argument detected, returning TRUE"
04053      severity warning;
04054      return true;
04055  elsif (Is_X(l) or Is_X(r)) then
04056    assert NO_WARNING
04057      report fixed_pkg'instance_name
04058      & """/=": metavalue detected, returning TRUE"
04059      severity warning;
04060      return true;
04061  end if;
04062  lresize := resize (l, left_index, right_index);
04063  rresize := resize (r, left_index, right_index);
04064  lslv   := to_s (lresize);
04065  rslv   := to_s (rresize);
04066  return lslv /= rslv;
04067 end function "/=";
04068
04069 function ">" (
04070   l, r : UNRESOLVED_ufixed)           -- fixed point input
04071   return BOOLEAN is
04072     constant left_index      : INTEGER := maximum(l'high, r'high);
04073     constant right_index     : INTEGER := mins(l'low, r'low);
04074     variable lresize, rresize : UNRESOLVED_ufixed (left_index downto right_index);
04075     variable lslv, rslv      : UNSIGNED (lresize'length-1 downto 0);
04076 begin
04077  if (l'length < 1 or r'length < 1) then
04078    assert NO_WARNING
04079      report fixed_pkg'instance_name
04080      & """>": null argument detected, returning FALSE"
04081      severity warning;
04082      return false;
04083  elsif (Is_X(l) or Is_X(r)) then
04084    assert NO_WARNING
04085      report fixed_pkg'instance_name

```

```

04086      & """>": metavalue detected, returning FALSE"
04087      severity warning;
04088      return false;
04089  end if;
04090  lresize := resize (l, left_index, right_index);
04091  rresize := resize (r, left_index, right_index);
04092  lslv    := to_uns (lresize);
04093  rslv    := to_uns (rresize);
04094  return lslv > rslv;
04095 end function ">";
04096
04097 function ">" (
04098  l, r : UNRESOLVED_sfixed)           -- fixed point input
04099  return BOOLEAN is
04100  constant left_index      : INTEGER := maximum(l'high, r'high);
04101  constant right_index     : INTEGER := mins(l'low, r'low);
04102  variable lresize, rresize : UNRESOLVED_sfixed (left_index downto right_index);
04103  variable lslv, rslv      : SIGNED (lresize'length-1 downto 0);
04104 begin
04105  if (l'length < 1 or r'length < 1) then
04106    assert NO_WARNING
04107    report fixed_pkg'instance_name
04108    & """>": null argument detected, returning FALSE"
04109    severity warning;
04110    return false;
04111  elsif (Is_X(l) or Is_X(r)) then
04112    assert NO_WARNING
04113    report fixed_pkg'instance_name
04114    & """>": metavalue detected, returning FALSE"
04115    severity warning;
04116    return false;
04117  end if;
04118  lresize := resize (l, left_index, right_index);
04119  rresize := resize (r, left_index, right_index);
04120  lslv   := to_s (lresize);
04121  rslv   := to_s (rresize);
04122  return lslv > rslv;
04123 end function ">";
04124
04125 function "<" (
04126  l, r : UNRESOLVED_ufixed)           -- fixed point input
04127  return BOOLEAN is
04128  constant left_index      : INTEGER := maximum(l'high, r'high);
04129  constant right_index     : INTEGER := mins(l'low, r'low);
04130  variable lresize, rresize : UNRESOLVED_ufixed (left_index downto right_index);
04131  variable lslv, rslv      : UNSIGNED (lresize'length-1 downto 0);
04132 begin
04133  if (l'length < 1 or r'length < 1) then
04134    assert NO_WARNING
04135    report fixed_pkg'instance_name
04136    & """<": null argument detected, returning FALSE"
04137    severity warning;
04138    return false;
04139  elsif (Is_X(l) or Is_X(r)) then
04140    assert NO_WARNING
04141    report fixed_pkg'instance_name
04142    & """<": metavalue detected, returning FALSE"
04143    severity warning;
04144    return false;
04145  end if;
04146  lresize := resize (l, left_index, right_index);
04147  rresize := resize (r, left_index, right_index);
04148  lslv   := to_uns (lresize);
04149  rslv   := to_uns (rresize);
04150  return lslv < rslv;
04151 end function "<";
04152
04153 function "<" (
04154  l, r : UNRESOLVED_sfixed)           -- fixed point input
04155  return BOOLEAN is
04156  constant left_index      : INTEGER := maximum(l'high, r'high);
04157  constant right_index     : INTEGER := mins(l'low, r'low);
04158  variable lresize, rresize : UNRESOLVED_sfixed (left_index downto right_index);
04159  variable lslv, rslv      : SIGNED (lresize'length-1 downto 0);
04160 begin
04161  if (l'length < 1 or r'length < 1) then
04162    assert NO_WARNING
04163    report fixed_pkg'instance_name
04164    & """<": null argument detected, returning FALSE"
04165    severity warning;
04166    return false;
04167  elsif (Is_X(l) or Is_X(r)) then
04168    assert NO_WARNING
04169    report fixed_pkg'instance_name
04170    & """<": metavalue detected, returning FALSE"
04171    severity warning;
04172    return false;

```

```

04173     end if;
04174     lresize := resize (l, left_index, right_index);
04175     rresize := resize (r, left_index, right_index);
04176     lslv    := to_s (lresize);
04177     rslv    := to_s (rresize);
04178     return lslv < rslv;
04179 end function "<";

04180
04181 function ">=" (
04182     l, r : UNRESOLVED_ufixed)           -- fixed point input
04183     return BOOLEAN is
04184     constant left_index      : INTEGER := maximum(l'high, r'high);
04185     constant right_index     : INTEGER := mins(l'low, r'low);
04186     variable lresize, rresize : UNRESOLVED_ufixed (left_index downto right_index);
04187     variable lslv, rslv      : UNSIGNED (lresize'length-1 downto 0);
04188 begin
04189     if (l'length < 1 or r'length < 1) then
04190         assert NO_WARNING
04191             report fixed_pkg'instance_name
04192             & """>=": null argument detected, returning FALSE"
04193             severity warning;
04194         return false;
04195     elsif (Is_X(l) or Is_X(r)) then
04196         assert NO_WARNING
04197             report fixed_pkg'instance_name
04198             & """>=": metavalue detected, returning FALSE"
04199             severity warning;
04200         return false;
04201     end if;
04202     lresize := resize (l, left_index, right_index);
04203     rresize := resize (r, left_index, right_index);
04204     lslv    := to_uns (lresize);
04205     rslv    := to_uns (rresize);
04206     return lslv >= rslv;
04207 end function ">=";

04208
04209 function ">=" (
04210     l, r : UNRESOLVED_sfixed)           -- fixed point input
04211     return BOOLEAN is
04212     constant left_index      : INTEGER := maximum(l'high, r'high);
04213     constant right_index     : INTEGER := mins(l'low, r'low);
04214     variable lresize, rresize : UNRESOLVED_sfixed (left_index downto right_index);
04215     variable lslv, rslv      : SIGNED (lresize'length-1 downto 0);
04216 begin
04217     if (l'length < 1 or r'length < 1) then
04218         assert NO_WARNING
04219             report fixed_pkg'instance_name
04220             & """>=": null argument detected, returning FALSE"
04221             severity warning;
04222         return false;
04223     elsif (Is_X(l) or Is_X(r)) then
04224         assert NO_WARNING
04225             report fixed_pkg'instance_name
04226             & """>=": metavalue detected, returning FALSE"
04227             severity warning;
04228         return false;
04229     end if;
04230     lresize := resize (l, left_index, right_index);
04231     rresize := resize (r, left_index, right_index);
04232     lslv    := to_s (lresize);
04233     rslv    := to_s (rresize);
04234     return lslv >= rslv;
04235 end function ">=";

04236
04237 function "<=" (
04238     l, r : UNRESOLVED_ufixed)           -- fixed point input
04239     return BOOLEAN is
04240     constant left_index      : INTEGER := maximum(l'high, r'high);
04241     constant right_index     : INTEGER := mins(l'low, r'low);
04242     variable lresize, rresize : UNRESOLVED_ufixed (left_index downto right_index);
04243     variable lslv, rslv      : UNSIGNED (lresize'length-1 downto 0);
04244 begin
04245     if (l'length < 1 or r'length < 1) then
04246         assert NO_WARNING
04247             report fixed_pkg'instance_name
04248             & """<=": null argument detected, returning FALSE"
04249             severity warning;
04250         return false;
04251     elsif (Is_X(l) or Is_X(r)) then
04252         assert NO_WARNING
04253             report fixed_pkg'instance_name
04254             & """<=": metavalue detected, returning FALSE"
04255             severity warning;
04256         return false;
04257     end if;
04258     lresize    := resize (l, left_index, right_index);
04259     rresize    := resize (r, left_index, right_index);

```

```

04260     lslv      := to_uns (lresize);
04261     rslv      := to_uns (rresize);
04262     return lslv <= rslv;
04263 end function "<=";
04264
04265 function "<=" (
04266     l, r : UNRESOLVED_sfixed)           -- fixed point input
04267 begin
04268     constant left_index      : INTEGER := maximum(l'high, r'high);
04269     constant right_index     : INTEGER := mins(l'low, r'low);
04270     variable lresize, rresize : UNRESOLVED_sfixed (left_index downto right_index);
04271     variable lslv, rslv      : SIGNED (lresize'length-1 downto 0);
04272
04273     if (l'length < 1 or r'length < 1) then
04274         assert NO_WARNING
04275             report fixed_pkg'instance_name
04276             & """<=": null argument detected, returning FALSE"
04277             severity warning;
04278         return false;
04279     elsif (Is_X(l) or Is_X(r)) then
04280         assert NO_WARNING
04281             report fixed_pkg'instance_name
04282             & """<=": metavalue detected, returning FALSE"
04283             severity warning;
04284         return false;
04285     end if;
04286     lresize    := resize (l, left_index, right_index);
04287     rresize    := resize (r, left_index, right_index);
04288     lslv       := to_s (lresize);
04289     rslv       := to_s (rresize);
04290     return lslv <= rslv;
04291 end function "<=";
04292
04293 -- overloads of the default maximum and minimum functions
04294 function maximum (l, r : UNRESOLVED_ufixed) return
04295     UNRESOLVED_ufixed is
04296     constant left_index      : INTEGER := maximum(l'high, r'high);
04297     constant right_index     : INTEGER := mins(l'low, r'low);
04298     variable lresize, rresize : UNRESOLVED_ufixed (left_index downto right_index);
04299 begin
04300     if (l'length < 1 or r'length < 1) then
04301         return NAUF;
04302     end if;
04303     lresize := resize (l, left_index, right_index);
04304     rresize := resize (r, left_index, right_index);
04305     if lresize > rresize then return lresize;
04306     else return rresize;
04307     end if;
04308 end function maximum;
04309
04310 function maximum (l, r : UNRESOLVED_sfixed) return
04311     UNRESOLVED_sfixed is
04312     constant left_index      : INTEGER := maximum(l'high, r'high);
04313     constant right_index     : INTEGER := mins(l'low, r'low);
04314     variable lresize, rresize : UNRESOLVED_sfixed (left_index downto right_index);
04315 begin
04316     if (l'length < 1 or r'length < 1) then
04317         return NASF;
04318     end if;
04319     lresize := resize (l, left_index, right_index);
04320     rresize := resize (r, left_index, right_index);
04321     if lresize > rresize then return lresize;
04322     else return rresize;
04323     end if;
04324 end function maximum;
04325
04326 function minimum (l, r : UNRESOLVED_ufixed) return
04327     UNRESOLVED_ufixed is
04328     constant left_index      : INTEGER := maximum(l'high, r'high);
04329     constant right_index     : INTEGER := mins(l'low, r'low);
04330     variable lresize, rresize : UNRESOLVED_ufixed (left_index downto right_index);
04331 begin
04332     if (l'length < 1 or r'length < 1) then
04333         return NAUF;
04334     end if;
04335     lresize := resize (l, left_index, right_index);
04336     rresize := resize (r, left_index, right_index);
04337     if lresize > rresize then return rresize;
04338     else return lresize;
04339     end if;
04340 end function minimum;
04341
04342 function minimum (l, r : UNRESOLVED_sfixed) return
04343     UNRESOLVED_sfixed is
04344     constant left_index      : INTEGER := maximum(l'high, r'high);
04345     constant right_index     : INTEGER := mins(l'low, r'low);
04346     variable lresize, rresize : UNRESOLVED_sfixed (left_index downto right_index);

```

```

04343 begin
04344   if (l'length < 1 or r'length < 1) then
04345     return NASF;
04346   end if;
04347   lresize := resize (l, left_index, right_index);
04348   rresize := resize (r, left_index, right_index);
04349   if lresize > rresize then return rresize;
04350   else return lresize;
04351   end if;
04352 end function minimum;
04353
04354 function to_ufixed (
04355   arg          : NATURAL; -- integer
04356   constant left_index    : INTEGER; -- left index (high index)
04357   constant right_index   : INTEGER           := 0; -- right index
04358   constant overflow_style : fixed_overflow_style_type :=
04359     fixed_overflow_style;
04360   constant round_style    : fixed_round_style_type := fixed_round_style)
04361   return UNRESOLVED_ufixed is
04362   constant fw      : INTEGER := mins (right_index, right_index); -- catch literals
04363   variable result  : UNRESOLVED_ufixed (left_index downto fw);
04364   variable sresult : UNRESOLVED_ufixed (left_index downto 0) :=
04365     (others => '0'); -- integer portion
04366   variable argx   : NATURAL; -- internal version of arg
04367 begin
04368   if (result'length < 1) then
04369     return NAUF;
04370   end if;
04371   if arg /= 0 then
04372     argx := arg;
04373     for I in 0 to sresult'left loop
04374       if (argx mod 2) = 0 then
04375         sresult(I) := '0';
04376       else
04377         sresult(I) := '1';
04378       end if;
04379       argx := argx/2;
04380     end loop;
04381     if argx /= 0 then
04382       assert NO_WARNING
04383         report fixed_pkg'instance_name
04384           & "TO_UFIXED(NATURAL): vector truncated"
04385           severity warning;
04386       if overflow_style = fixed_saturate then
04387         return saturate (left_index, right_index);
04388       end if;
04389     end if;
04390     result := resize (arg
04391                   left_index  => left_index,
04392                   right_index => right_index,
04393                   round_style  => round_style,
04394                   overflow_style => overflow_style);
04395   else
04396     result := (others => '0');
04397   end if;
04398   return result;
04399 end function to_ufixed;
04400
04401 function to_sfixed (
04402   arg          : INTEGER; -- integer
04403   constant left_index    : INTEGER; -- left index (high index)
04404   constant right_index   : INTEGER           := 0; -- right index
04405   constant overflow_style : fixed_overflow_style_type :=
04406     fixed_overflow_style;
04407   constant round_style    : fixed_round_style_type := fixed_round_style)
04408   return UNRESOLVED_sfixed is
04409   constant fw      : INTEGER := mins (right_index, right_index); -- catch literals
04410   variable result  : UNRESOLVED_sfixed (left_index downto fw);
04411   variable sresult : UNRESOLVED_sfixed (left_index downto 0) :=
04412     (others => '0'); -- integer portion
04413   variable argx   : INTEGER; -- internal version of arg
04414   variable sign    : STD_ULOGIC; -- sign of input
04415 begin
04416   if (result'length < 1) then -- null range
04417     return NASF;
04418   end if;
04419   if arg /= 0 then
04420     if (arg < 0) then
04421       sign := '1';
04422       argx := -(arg + 1);
04423     else
04424       sign := '0';
04425       argx := arg;
04426     end if;
04427     for I in 0 to sresult'left loop

```

```

04426      if (argx mod 2) = 0 then
04427          sresult(I) := sign;
04428      else
04429          sresult(I) := not sign;
04430      end if;
04431      argx := argx/2;
04432  end loop;
04433  if argx /= 0 or left_index < 0 or sign /= sresult(sresult'left) then
04434      assert NO_WARNING
04435          report fixed_pkg'instance_name
04436          & "TO_SFIXED(INTEGER): vector truncated"
04437          severity warning;
04438  if overflow_style = fixed_saturate then           -- saturate
04439      if arg < 0 then
04440          result := not saturate(result'high, result'low); -- underflow
04441      else
04442          result := saturate(result'high, result'low);   -- overflow
04443      end if;
04444      return result;
04445  end if;
04446 end if;
04447 result := resize(arg
04448                 left_index    => left_index,
04449                 right_index   => right_index,
04450                 round_style   => round_style,
04451                 overflow_style=> overflow_style);
04452 else
04453     result := (others => '0');
04454 end if;
04455 return result;
04456 end function to_sfixed;
04457
04458 function to_ufixed (
04459     arg
04460         : REAL;        -- real
04461     constant left_index
04462         : INTEGER;    -- left index (high index)
04463     constant right_index
04464         : INTEGER;    -- right index
04465     constant overflow_style
04466         : fixed_overflow_style_type := fixed_overflow_style;
04467     constant round_style
04468         : fixed_round_style_type := fixed_round_style;
04469     constant guard_bits
04470         : NATURAL        := fixed_guard_bits) -- # of guard
04471 bits
04472 return UNRESOLVED_ufixed;
04473 constant fw
04474     : INTEGER := mins(right_index, right_index); -- catch literals
04475 variable result
04476     : UNRESOLVED_ufixed (left_index downto fw) :=
04477     (others => '0');
04478 variable Xresult
04479     : UNRESOLVED_ufixed (left_index downto
04480                           fw-guard_bits) :=
04481     (others => '0');
04482 variable presult
04483     : REAL;
04484 --  variable overflow_needed : BOOLEAN;
04485 begin
04486 -- If negative or null range, return.
04487  if (left_index < fw) then
04488      return NAUF;
04489  end if;
04490  if (arg < 0.0) then
04491      report fixed_pkg'instance_name
04492          & "TO_UFIXED: Negative argument passed "
04493          & REAL'image(arg) severity error;
04494  end if;
04495  presult := arg;
04496  if presult >= (2.0**left_index+1) then
04497      assert NO_WARNING report fixed_pkg'instance_name
04498          & "TO_UFIXED(REAL): vector truncated"
04499          severity warning;
04500  if overflow_style = fixed_wrap then
04501      presult := presult mod (2.0**left_index+1)); -- wrap
04502  else
04503      return saturate(result'high, result'low);
04504  end if;
04505 end if;
04506 for i in Xresult'range loop
04507  if presult >= 2.0**i then
04508      Xresult(i) := '1';
04509      presult    := presult - 2.0**i;
04510  else
04511      Xresult(i) := '0';
04512  end if;
04513 end loop;
04514 if guard_bits > 0 and round_style = fixed_round then
04515     result := round_fixed (arg => Xresult (left_index
04516                               downto right_index),
04517                               remainder => Xresult (right_index-1 downto
04518                               right_index-guard_bits),
04519                               overflow_style => overflow_style);

```

```

04510     else
04511         result := Xresult (result'range);
04512     end if;
04513     return result;
04514 end function to_ufixed;
04515
04516 function to_sfixed (
04517     arg                  : REAL;      -- real
04518     constant left_index   : INTEGER;  -- left index (high index)
04519     constant right_index  : INTEGER;  -- right index
04520     constant overflow_style : fixed_overflow_style_type :=
04521         fixed_overflow_style;
04522     constant round_style   : fixed_round_style_type      :=
04523         fixed_round_style;
04524     constant guard_bits    : NATURAL                := fixed_guard_bits) -- # of guard
04525     bits
04526     return UNRESOLVED_sfixed is
04527     constant fw      : INTEGER := mins (right_index, right_index); -- catch literals
04528     variable result : UNRESOLVED_sfixed (left_index downto fw) :=
04529         (others => '0');
04530     variable Xresult : UNRESOLVED_sfixed (left_index+1 downto fw-guard_bits) :=
04531         (others => '0');
04532     variable presresult : REAL;
04533 begin
04534     if (left_index < fw) then          -- null range
04535         return NASF;
04536     end if;
04537     if (arg >= (2.0**left_index) or arg < -(2.0**left_index)) then
04538         assert NO_WARNING report fixed_pkg'instance_name
04539             & "TO_SFIXED(REAL): vector truncated"
04540             severity warning;
04541         if overflow_style = fixed_saturate then
04542             if arg < 0.0 then           -- saturate
04543                 result := not saturate (result'high, result'low);      -- underflow
04544             else
04545                 result := saturate (result'high, result'low);        -- overflow
04546             end if;
04547             return result;
04548         else
04549             presresult := abs(arg) mod (2.0**((left_index+1)));
04550             end if;
04551         else
04552             presresult := abs(arg);
04553             for i in Xresult'range loop
04554                 if presresult >= 2.0**i then
04555                     Xresult(i) := '1';
04556                     presresult := presresult - 2.0**i;
04557                 else
04558                     Xresult(i) := '0';
04559                 end if;
04560             end loop;
04561             if arg < 0.0 then
04562                 Xresult := to_fixed(-to_s(Xresult), Xresult'high, Xresult'low);
04563             end if;
04564             if guard_bits > 0 and round_style = fixed_round then
04565                 result := round_fixed (arg => Xresult (left_index
04566                                         downto right_index),
04567                                         remainder => Xresult (right_index-1 downto
04568                                         right_index-guard_bits),
04569                                         overflow_style => overflow_style);
04570             else
04571                 result := Xresult (result'range);
04572             end if;
04573         end function to_sfixed;
04574
04575 function to_ufixed (
04576     arg                  : UNSIGNED;      -- unsigned
04577     constant left_index   : INTEGER;  -- left index (high index)
04578     constant right_index  : INTEGER;  -- right index
04579     constant overflow_style : fixed_overflow_style_type :=
04580         fixed_overflow_style;
04581     constant round_style   : fixed_round_style_type      :=
04582         fixed_round_style;
04583     return UNRESOLVED_ufixed is
04584     constant ARG_LEFT : INTEGER := ARG'length-1;
04585     alias XARG      : UNSIGNED(ARG_LEFT downto 0) is ARG;
04586     variable result   : UNRESOLVED_ufixed (left_index downto right_index);
04587 begin
04588     if arg'length < 1 or (left_index < right_index) then
04589         return NAUF;
04590     end if;
04591     result := resize (arg
04592                         => UNRESOLVED_ufixed (XARG),
04593                         left_index      => left_index,
04594                         right_index     => right_index,
04595                         round_style     => round_style,

```

```

04592           overflow_style => overflow_style);
04593     return result;
04594   end function to_ufixed;
04595
04596 -- converted version
04597 function to_ufixed (
04598   arg : UNSIGNED) -- unsigned
04599   return UNRESOLVED_ufixed is
04600   constant ARG_LEFT : INTEGER := ARG'length-1;
04601   alias XARG : UNSIGNED(ARG_LEFT downto 0) is ARG;
04602 begin
04603   if arg'length < 1 then
04604     return NAUF;
04605   end if;
04606   return UNRESOLVED_ufixed(xarg);
04607 end function to_ufixed;
04608
04609 function to_sfixed (
04610   arg : SIGNED; -- signed
04611   constant left_index : INTEGER; -- left index (high index)
04612   constant right_index : INTEGER := 0; -- right index
04613   constant overflow_style : fixed_overflow_style_type := fixed_overflow_style;
04614   constant round_style : fixed_round_style_type := fixed_round_style;
04615   return UNRESOLVED_sfixed is
04616   constant ARG_LEFT : INTEGER := ARG'length-1;
04617   alias XARG : SIGNED(ARG_LEFT downto 0) is ARG;
04618   variable result : UNRESOLVED_sfixed (left_index downto right_index);
04619 begin
04620   if arg'length < 1 or (left_index < right_index) then
04621     return NASF;
04622   end if;
04623   result := resize (arg --> UNRESOLVED_sfixed (XARG),
04624                     left_index --> left_index,
04625                     right_index --> right_index,
04626                     round_style --> round_style,
04627                     overflow_style => overflow_style);
04628   return result;
04629 end function to_sfixed;
04630
04631 -- converted version
04632 function to_sfixed (
04633   arg : SIGNED) -- signed
04634   return UNRESOLVED_sfixed is
04635   constant ARG_LEFT : INTEGER := ARG'length-1;
04636   alias XARG : SIGNED(ARG_LEFT downto 0) is ARG;
04637 begin
04638   if arg'length < 1 then
04639     return NASF;
04640   end if;
04641   return UNRESOLVED_sfixed(xarg);
04642 end function to_sfixed;
04643
04644 function to_sfixed (arg : UNRESOLVED_ufixed) return UNRESOLVED_sfixed is
04645   variable result : UNRESOLVED_sfixed (arg'high+1 downto arg'low);
04646 begin
04647   if arg'length < 1 then
04648     return NASF;
04649   end if;
04650   result (arg'high downto arg'low) := UNRESOLVED_sfixed(cleanvec(arg));
04651   result (arg'high+1) := '0';
04652   return result;
04653 end function to_sfixed;
04654
04655 -- Because of the fairly complicated sizing rules in the fixed point
04656 -- packages these functions are provided to compute the result ranges
04657 -- Example:
04658 -- signal ufl : ufixed (3 downto -3);
04659 -- signal uf2 : ufixed (4 downto -2);
04660 -- signal uflmultuf2 : ufixed (ufixed_high (3, -3, '*', 4, -2) downto
04661 --                                ufixed_low (3, -3, '*', 4, -2));
04662 -- uflmultuf2 <= ufl * uf2;
04663 -- Valid characters: '+', '-', '*', '/', 'r' or 'R' (rem), 'm' or 'M' (mod),
04664 -- '1' (reciprocal), 'A', 'a' (abs), 'N', 'n' (-sfixed)
04665 function ufixed_high (left_index, right_index : INTEGER;
04666   operation : CHARACTER := 'X';
04667   left_index2, right_index2 : INTEGER := 0)
04668   return INTEGER is
04669 begin
04670   case operation is
04671     when '+' | '-' => return maximum (left_index, left_index2) + 1;
04672     when '*' => return left_index + left_index2 + 1;
04673     when '/' => return left_index - right_index2;
04674     when '1' => return -right_index; -- reciprocal
04675     when 'R' | 'r' => return mins (left_index, left_index2); -- "rem"

```

```

04676      when 'M' | 'm'  => return mins (left_index, left_index2); -- "mod"
04677      when others    => return left_index; -- For abs and default
04678    end case;
04679  end function ufixed_high;
04680
04681  function ufixed_low (left_index, right_index : INTEGER;
04682                      operation       : CHARACTER := 'X';
04683                      left_index2, right_index2 : INTEGER := 0)
04684    return INTEGER is
04685  begin
04686    case operation is
04687      when '+' | '-' => return mins (right_index, right_index2);
04688      when '*'        => return right_index + right_index2;
04689      when '/'        => return right_index - left_index2 - 1;
04690      when '1'         => return -left_index - 1; -- reciprocal
04691      when 'R' | 'r'   => return mins (right_index, right_index2); -- "rem"
04692      when 'M' | 'm'   => return mins (right_index, right_index2); -- "mod"
04693      when others     => return right_index; -- for abs and default
04694    end case;
04695  end function ufixed_low;
04696
04697  function sfixed_high (left_index, right_index : INTEGER;
04698                      operation       : CHARACTER := 'X';
04699                      left_index2, right_index2 : INTEGER := 0)
04700    return INTEGER is
04701  begin
04702    case operation is
04703      when '+' | '-' => return maximum (left_index, left_index2) + 1;
04704      when '*'        => return left_index + left_index2 + 1;
04705      when '/'        => return left_index - right_index2 + 1;
04706      when '1'         => return -right_index + 1; -- reciprocal
04707      when 'R' | 'r'   => return mins (left_index, left_index2); -- "rem"
04708      when 'M' | 'm'   => return left_index2; -- "mod"
04709      when 'A' | 'a'   => return left_index + 1; -- "abs"
04710      when 'N' | 'n'   => return left_index + 1; -- -sfixed
04711      when others     => return left_index;
04712    end case;
04713  end function sfixed_high;
04714
04715  function sfixed_low (left_index, right_index : INTEGER;
04716                      operation       : CHARACTER := 'X';
04717                      left_index2, right_index2 : INTEGER := 0)
04718    return INTEGER is
04719  begin
04720    case operation is
04721      when '+' | '-' => return mins (right_index, right_index2);
04722      when '*'        => return right_index + right_index2;
04723      when '/'        => return right_index - left_index2;
04724      when '1'         => return -left_index; -- reciprocal
04725      when 'R' | 'r'   => return mins (right_index, right_index2); -- "rem"
04726      when 'M' | 'm'   => return mins (right_index, right_index2); -- "mod"
04727      when others     => return right_index; -- default for abs, neg and default
04728    end case;
04729  end function sfixed_low;
04730
04731  -- Same as above, but using the "size_res" input only for their ranges:
04732  -- signal uflmultuf2 : ufixed (uf1, '*' , uf2) downto
04733  --                      ufixed_low (uf1, '*' , uf2));
04734  -- uflmultuf2 <= uf1 * uf2;
04735  function ufixed_high (size_res : UNRESOLVED_ufixed;
04736                         operation : CHARACTER := 'X';
04737                         size_res2 : UNRESOLVED_ufixed)
04738    return INTEGER is
04739  begin
04740    return ufixed_high (left_index  => size_res'high,
04741                        right_index => size_res'low,
04742                        operation   => operation,
04743                        left_index2 => size_res2'high,
04744                        right_index2 => size_res2'low);
04745  end function ufixed_high;
04746
04747  function ufixed_low (size_res : UNRESOLVED_ufixed;
04748                         operation : CHARACTER := 'X';
04749                         size_res2 : UNRESOLVED_ufixed)
04750    return INTEGER is
04751  begin
04752    return ufixed_low (left_index  => size_res'high,
04753                        right_index => size_res'low,
04754                        operation   => operation,
04755                        left_index2 => size_res2'high,
04756                        right_index2 => size_res2'low);
04757  end function ufixed_low;
04758
04759  function sfixed_high (size_res : UNRESOLVED_sfixed;
04760                         operation : CHARACTER := 'X';
04761                         size_res2 : UNRESOLVED_sfixed)
04762    return INTEGER is

```

```

04763 begin
04764     return sfixed_high (left_index  => size_res'high,
04765                 right_index => size_res'low,
04766                 operation   => operation,
04767                 left_index2 => size_res2'high,
04768                 right_index2 => size_res2'low);
04769 end function sfixed_high;
04770
04771 function sfixed_low (size_res : UNRESOLVED_sfixed;
04772                      operation : CHARACTER := 'X';
04773                      size_res2 : UNRESOLVED_sfixed)
04774     return INTEGER is
04775 begin
04776     return sfixed_low (left_index  => size_res'high,
04777                         right_index => size_res'low,
04778                         operation   => operation,
04779                         left_index2 => size_res2'high,
04780                         right_index2 => size_res2'low);
04781 end function sfixed_low;
04782
04783 -- purpose: returns a saturated number
04784 function saturate (
04785     constant left_index : INTEGER;
04786     constant right_index : INTEGER)
04787     return UNRESOLVED_ufixed is
04788     constant sat : UNRESOLVED_ufixed (left_index downto right_index) :=
04789         (others => '1');
04790 begin
04791     return sat;
04792 end function saturate;
04793
04794 -- purpose: returns a saturated number
04795 function saturate (
04796     constant left_index : INTEGER;
04797     constant right_index : INTEGER)
04798     return UNRESOLVED_sfixed is
04799     variable sat : UNRESOLVED_sfixed (left_index downto right_index) :=
04800         (others => '1');
04801 begin
04802     -- saturate positive, to saturate negative, just do "not saturate()"
04803     sat (left_index) := '0';
04804     return sat;
04805 end function saturate;
04806
04807 function saturate (
04808     size_res : UNRESOLVED_ufixed)      -- only the size of this is used
04809     return UNRESOLVED_ufixed is
04810 begin
04811     return saturate (size_res'high, size_res'low);
04812 end function saturate;
04813
04814 function saturate (
04815     size_res : UNRESOLVED_sfixed)      -- only the size of this is used
04816     return UNRESOLVED_sfixed is
04817 begin
04818     return saturate (size_res'high, size_res'low);
04819 end function saturate;
04820
04821 -- As a concession to those who use a graphical DSP environment,
04822 -- these functions take parameters in those tools format and create
04823 -- fixed point numbers. These functions are designed to convert from
04824 -- a std_logic_vector to the VHDL fixed point format using the conventions
04825 -- of these packages. In a pure VHDL environment you should use the
04826 -- "to_ufixed" and "to_sfixed" routines.
04827 -- Unsigned fixed point
04828 function to_UFix (
04829     arg      : STD_ULONGIC_VECTOR;
04830     width   : NATURAL;                  -- width of vector
04831     fraction : NATURAL)                -- width of fraction
04832     return UNRESOLVED_ufixed is
04833     variable result : UNRESOLVED_ufixed (width-fraction-1 downto -fraction);
04834 begin
04835     if (arg'length /= result'length) then
04836         report fixed_pkg'instance_name
04837             & "TO_UFIX (STD_ULONGIC_VECTOR) "
04838             & "Vector lengths do not match. Input length is "
04839             & INTEGER'image(arg'length) & " and output will be "
04840             & INTEGER'image(result'length) & " wide."
04841         severity error;
04842         return NAUF;
04843     else
04844         result := to_ufixed (arg, result'high, result'low);
04845         return result;
04846     end if;
04847 end function to_UFix;
04848
04849 -- signed fixed point

```

```

04850  function to_SFix (
04851    arg      : STD_ULONGIC_VECTOR;
04852    width   : NATURAL;                      -- width of vector
04853    fraction : NATURAL)                   -- width of fraction
04854    return UNRESOLVED_sfixed is
04855    variable result : UNRESOLVED_sfixed (width-fraction-1 downto -fraction);
04856 begin
04857  if (arg'length /= result'length) then
04858    report fixed_pkg'instance_name
04859    & "TO_SFIX (STD_ULONGIC_VECTOR) "
04860    & "Vector lengths do not match. Input length is "
04861    & INTEGER'image(arg'length) & " and output will be "
04862    & INTEGER'image(result'length) & " wide."
04863    severity error;
04864  return NASF;
04865 else
04866  result := to_sfixed (arg, result'high, result'low);
04867  return result;
04868 end if;
04869 end function to_SFix;
04870
-- finding the bounds of a number. These functions can be used like this:
04871 -- signal xxx : ufixed (7 downto -3);
04872 -- -- Which is the same as "ufixed (UFix_high (11,3) downto UFix_low(11,3))"
04873 -- signal yyy : ufixed (UFix_high (11, 3, "+", 11, 3)
04874 --                      downto UFix_low(11, 3, "+", 11, 3));
04875 -- Where "11" is the width of xxx (xxx'length),
04876 -- and 3 is the lower bound (abs (xxx'low))
04877 -- In a pure VHDL environment use "ufixed_high" and "ufixed_low"
04878
04879 function UFix_high (
04880  width, fraction : NATURAL;
04881  operation       : CHARACTER := 'X';
04882  width2, fraction2 : NATURAL := 0)
04883  return INTEGER is
04884 begin
04885  return ufixed_high (left_index  => width - 1 - fraction,
04886                      right_index => -fraction,
04887                      operation   => operation,
04888                      left_index2 => width2 - 1 - fraction2,
04889                      right_index2 => -fraction2);
04890 end function ufix_high;
04891
04892 function UFix_low (
04893  width, fraction : NATURAL;
04894  operation       : CHARACTER := 'X';
04895  width2, fraction2 : NATURAL := 0)
04896  return INTEGER is
04897 begin
04898  return ufixed_low (left_index  => width - 1 - fraction,
04899                      right_index => -fraction,
04900                      operation   => operation,
04901                      left_index2 => width2 - 1 - fraction2,
04902                      right_index2 => -fraction2);
04903 end function ufix_low;
04904
04905 function SFix_high (
04906  width, fraction : NATURAL;
04907  operation       : CHARACTER := 'X';
04908  width2, fraction2 : NATURAL := 0)
04909  return INTEGER is
04910 begin
04911  return sfixed_high (left_index  => width - fraction,
04912                      right_index => -fraction,
04913                      operation   => operation,
04914                      left_index2 => width2 - fraction2,
04915                      right_index2 => -fraction2);
04916 end function sfix_high;
04917
04918 function SFix_low (
04919  width, fraction : NATURAL;
04920  operation       : CHARACTER := 'X';
04921  width2, fraction2 : NATURAL := 0)
04922  return INTEGER is
04923 begin
04924  return sfixed_low (left_index  => width - fraction,
04925                      right_index => -fraction,
04926                      operation   => operation,
04927                      left_index2 => width2 - fraction2,
04928                      right_index2 => -fraction2);
04929 end function sfix_low;
04930
04931 function to_unsigned (
04932  arg                  : UNRESOLVED_ufixed; -- ufixed point input
04933  constant size        : NATURAL;           -- length of output
04934  constant overflow_style : fixed_overflow_style_type :=
04935    fixed_overflow_style;
04936  constant round_style  : fixed_round_style_type :=
```

```

    fixed_round_style)
04936     return UNSIGNED is
04937 begin
04938     return to_uns(resize (arg
04939         left_index      => size-1,
04940         right_index     => 0,
04941         round_style     => round_style,
04942         overflow_style  => overflow_style));
04943 end function to_unsigned;
04944
04945 function to_unsigned (
04946     arg           : UNRESOLVED_ufixed;      -- ufixed point input
04947     size_res       : UNSIGNED;             -- length of output
04948     constant overflow_style : fixed_overflow_style_type := 
04949         fixed_overflow_style;
04950     constant round_style   : fixed_round_style_type   := 
04951         fixed_round_style);
04952     return UNSIGNED is
04953 begin
04954     return to_unsigned (arg
04955         size          => size_res'length,
04956         round_style   => round_style,
04957         overflow_style => overflow_style);
04958 end function to_unsigned;
04959
04960 function to_signed (
04961     arg           : UNRESOLVED_sfixed;      -- sfixed point input
04962     constant size        : NATURAL;          -- length of output
04963     constant overflow_style : fixed_overflow_style_type := 
04964         fixed_overflow_style;
04965     constant round_style   : fixed_round_style_type   := 
04966         fixed_round_style);
04967     return SIGNED is
04968 begin
04969     return to_s(resize (arg
04970         left_index      => size-1,
04971         right_index     => 0,
04972         round_style     => round_style,
04973         overflow_style  => overflow_style));
04974 end function to_signed;
04975
04976 function to_signed (
04977     arg           : UNRESOLVED_sfixed;      -- sfixed point input
04978     size_res       : SIGNED;               -- used for length of output
04979     constant overflow_style : fixed_overflow_style_type := 
04980         fixed_overflow_style;
04981     constant round_style   : fixed_round_style_type   := 
04982         fixed_round_style);
04983     return SIGNED is
04984 begin
04985     return to_signed (arg
04986         size          => size_res'length,
04987         round_style   => round_style,
04988         overflow_style => overflow_style);
04989 end function to_signed;
04990
04991 function to_real (
04992     arg : UNRESOLVED_ufixed)           -- ufixed point input
04993 begin
04994     if (arg'length < 1) then
04995         return 0.0;
04996     end if;
04997     arg_int := to_x01(cleanvec(arg));
04998     if (Is_X(arg_int)) then
04999         assert NO_WARNING
05000             report fixed_pkg'instance_name
05001                 & "TO_REAL (ufixed): metavalue detected, returning 0.0"
05002             severity warning;
05003         return 0.0;
05004     end if;
05005     result := 0.0;
05006     for i in arg_int'range loop
05007         if (arg_int(i) = '1') then
05008             result := result + (2.0**i);
05009         end if;
05010     end loop;
05011     return result;
05012 end function to_real;
05013
05014 function to_real (
05015     arg : UNRESOLVED_sfixed)           -- ufixed point input
05016     return REAL is

```

```

05016  constant left_index : INTEGER := arg'high;
05017  constant right_index : INTEGER := arg'low;
05018  variable result      : REAL;           -- result
05019  variable arg_int     : UNRESOLVED_sfixed (left_index downto right_index);
05020  -- unsigned version of argument
05021  variable arg_uns     : UNRESOLVED_ufixed (left_index downto right_index);
05022  -- absolute of argument
05023 begin
05024  if (arg'length < 1) then
05025    return 0.0;
05026  end if;
05027  arg_int := to_x01(cleanvec(arg));
05028  if (Is_X(arg_int)) then
05029    assert NO_WARNING
05030      report fixed_pkg'instance_name
05031      & "TO_REAL (sfixed): metavalue detected, returning 0.0"
05032      severity warning;
05033    return 0.0;
05034  end if;
05035  arg_uns := to_ufixed (arg_int);
05036  result  := to_real (arg_uns);
05037  if (arg_int(arg_int'high) = '1') then
05038    result := -result;
05039  end if;
05040  return result;
05041 end function to_real;
05042
05043 function to_integer (
05044  arg          : UNRESOLVED_ufixed; -- fixed point input
05045  constant overflow_style : fixed_overflow_style_type :=
05046    fixed_overflow_style;
05047  constant round_style   : fixed_round_style_type   :=
05048    fixed_round_style)
05049 begin
05050  constant left_index : INTEGER := arg'high;
05051  variable arg_uns    : UNSIGNED (left_index+1 downto 0)
05052  := (others => '0');
05053 begin
05054  if (arg'length < 1) then
05055    return 0;
05056  end if;
05057  if (Is_X (arg)) then
05058    assert NO_WARNING
05059      report fixed_pkg'instance_name
05060      & "TO_INTEGER (ufixed): metavalue detected, returning 0"
05061      severity warning;
05062    return 0;
05063  end if;
05064  if (left_index < -1) then
05065    return 0;
05066  end if;
05067  arg_uns := to_uns(resize (arg
05068                left_index  => arg_uns'high,
05069                right_index => 0,
05070                round_style  => round_style,
05071                overflow_style => overflow_style));
05072 end function to_integer;
05073
05074 function to_integer (
05075  arg          : UNRESOLVED_sfixed; -- fixed point input
05076  constant overflow_style : fixed_overflow_style_type :=
05077    fixed_overflow_style;
05078  constant round_style   : fixed_round_style_type   :=
05079    fixed_round_style)
05080 begin
05081  constant left_index : INTEGER := arg'high;
05082  constant right_index : INTEGER := arg'low;
05083  variable arg_s       : SIGNED (left_index+1 downto 0);
05084 begin
05085  if (arg'length < 1) then
05086    return 0;
05087  end if;
05088  if (Is_X (arg)) then
05089    assert NO_WARNING
05090      report fixed_pkg'instance_name
05091      & "TO_INTEGER (sfixed): metavalue detected, returning 0"
05092      severity warning;
05093    return 0;
05094  end if;
05095  arg_s := to_s(resize (arg
05096                left_index  => arg_s'high,
05097                right_index => 0,
05098                round_style  => round_style,

```

```

05099         overflow_style => overflow_style));
05100     return to_integer (arg_s);
05101 end function to_integer;
05102
05103 function to_01 (
05104     s           : UNRESOLVED_ufixed;          -- ufixed point input
05105     constant XMAP : STD_ULOGIC := '0'        -- Map x to
05106     return UNRESOLVED_ufixed is
05107     variable result : UNRESOLVED_ufixed (s'range); -- result
05108 begin
05109     if (s'length < 1) then
05110         assert NO_WARNING
05111         report fixed_pkg'instance_name
05112             & "TO_01(ufixed): null detected, returning NULL"
05113             severity warning;
05114         return NAUF;
05115     end if;
05116     return to_fixed (to_01(to_uns(s), XMAP), s'high, s'low);
05117 end function to_01;
05118
05119 function to_01 (
05120     s           : UNRESOLVED_sfixed;    -- sfixed point input
05121     constant XMAP : STD_ULOGIC := '0'  -- Map x to
05122     return UNRESOLVED_sfixed is
05123     variable result : UNRESOLVED_sfixed (s'range);
05124 begin
05125     if (s'length < 1) then
05126         assert NO_WARNING
05127         report fixed_pkg'instance_name
05128             & "TO_01(sfixed): null detected, returning NULL"
05129             severity warning;
05130         return NASF;
05131     end if;
05132     return to_fixed (to_01(to_s(s), XMAP), s'high, s'low);
05133 end function to_01;
05134
05135 function Is_X (
05136     arg : UNRESOLVED_ufixed)
05137     return BOOLEAN is
05138     variable argslv : STD_ULOGIC_VECTOR (arg'length-1 downto 0); -- slv
05139 begin
05140     argslv := to_sulv(arg);
05141     return Is_X (argslv);
05142 end function Is_X;
05143
05144 function Is_X (
05145     arg : UNRESOLVED_sfixed)
05146     return BOOLEAN is
05147     variable argslv : STD_ULOGIC_VECTOR (arg'length-1 downto 0); -- slv
05148 begin
05149     argslv := to_sulv(arg);
05150     return Is_X (argslv);
05151 end function Is_X;
05152
05153 function to_X01 (
05154     arg : UNRESOLVED_ufixed)
05155     return UNRESOLVED_ufixed is
05156 begin
05157     return to_ufixed (To_X01(to_sulv(arg)), arg'high, arg'low);
05158 end function To_X01;
05159
05160 function to_X01 (
05161     arg : UNRESOLVED_sfixed)
05162     return UNRESOLVED_sfixed is
05163 begin
05164     return to_sfixed (To_X01(to_sulv(arg)), arg'high, arg'low);
05165 end function To_X01;
05166
05167 function to_X01Z (
05168     arg : UNRESOLVED_ufixed)
05169     return UNRESOLVED_ufixed is
05170 begin
05171     return to_ufixed (To_X01Z(to_sulv(arg)), arg'high, arg'low);
05172 end function To_X01Z;
05173
05174 function to_X01Z (
05175     arg : UNRESOLVED_sfixed)
05176     return UNRESOLVED_sfixed is
05177 begin
05178     return to_sfixed (To_X01Z(to_sulv(arg)), arg'high, arg'low);
05179 end function To_X01Z;
05180
05181 function to_UX01 (
05182     arg : UNRESOLVED_ufixed)
05183     return UNRESOLVED_ufixed is
05184 begin
05185     return to_ufixed (To_UX01(to_sulv(arg)), arg'high, arg'low);

```

```

05186    end function To_UX01;
05187
05188    function to_UX01 (
05189        arg : UNRESOLVED_sfixed)
05190        return UNRESOLVED_sfixed is
05191    begin
05192        return to_sfixed (To_UX01(to_sulv(arg)), arg'high, arg'low);
05193    end function To_UX01;
05194
05195    function resize (
05196        arg                      : UNRESOLVED_ufixed;           -- input
05197        constant left_index       : INTEGER;   -- integer portion
05198        constant right_index      : INTEGER;   -- size of fraction
05199        constant overflow_style   : fixed_overflow_style_type :=  

05200            fixed_overflow_style;
05201        constant round_style      : fixed_round_style_type     :=  

05202            fixed_round_style);
05203        return UNRESOLVED_ufixed is
05204            constant arghigh : INTEGER := maximum (arg'high, arg'low);
05205            constant arglow  : INTEGER := mine (arg'high, arg'low);
05206            variable invec   : UNRESOLVED_ufixed (arghigh downto arglow);
05207            variable result   : UNRESOLVED_ufixed(left_index downto right_index) :=  

05208                (others => '0');
05209            variable needs_rounding : BOOLEAN := false;
05210        begin -- resize
05211            if (arg'length < 1) or (result'length < 1) then
05212                return NAUF;
05213            elsif (invec'length < 1) then
05214                return result;           -- string literal value
05215            else
05216                invec := cleanvec(arg);
05217                if (right_index > arghigh) then -- return top zeros
05218                    needs_rounding := (round_style = fixed_round) and  

05219                        (right_index = arghigh+1);
05220                elsif (left_index < arglow) then -- return overflow
05221                    if (overflow_style = fixed_saturate) and  

05222                        (or_reduce(to_sulv(invec)) = '1') then
05223                        result := saturate (result'high, result'low); -- saturate
05224                    end if;
05225                elsif (arghigh > left_index) then
05226                    -- wrap or saturate?
05227                    if (overflow_style = fixed_saturate and  

05228                        or_reduce (to_sulv(invec(arghigh downto left_index+1))) = '1')
05229                    then
05230                        result := saturate (result'high, result'low); -- saturate
05231                    else
05232                        if (arglow >= right_index) then
05233                            result (left_index downto arglow) :=  

05234                                invec(left_index downto arglow);
05235                        else
05236                            result (left_index downto right_index) :=  

05237                                invec (left_index downto right_index);
05238                            needs_rounding := (round_style = fixed_round); -- round
05239                        end if;
05240                    end if;
05241                else
05242                    -- arghigh <= integer width
05243                    if (arglow >= right_index) then
05244                        result (arghigh downto arglow) := invec;
05245                    else
05246                        result (arghigh downto right_index) :=  

05247                            invec (arghigh downto right_index);
05248                        needs_rounding := (round_style = fixed_round); -- round
05249                    end if;
05250                end if;
05251            -- Round result
05252            if needs_rounding then
05253                result := round_fixed (arg          => result,
05254                                remainder     => invec (right_index-1  

05255                                         downto arglow),
05256                                         overflow_style => overflow_style);
05257            end if;
05258        end function resize;
05259
05260    function resize (
05261        arg                      : UNRESOLVED_sfixed;           -- input
05262        constant left_index       : INTEGER;   -- integer portion
05263        constant right_index      : INTEGER;   -- size of fraction
05264        constant overflow_style   : fixed_overflow_style_type :=  

05265            fixed_overflow_style;
05266        constant round_style      : fixed_round_style_type     :=  

05267            fixed_round_style);
05268        return UNRESOLVED_sfixed is
05269            constant arghigh : INTEGER := maximum (arg'high, arg'low);
05270            constant arglow  : INTEGER := mine (arg'high, arg'low);
05271            variable invec   : UNRESOLVED_sfixed (arghigh downto arglow);

```

```

05269     variable result  : UNRESOLVED_sfixed(left_index downto right_index) :=
05270         (others => '0');
05271     variable reduced      : STD_ULONGIC;
05272     variable needs_rounding : BOOLEAN := false;                      -- rounding
05273 begin -- resize
05274     if (arg'length < 1) or (result'length < 1) then
05275         return NASF;
05276     elsif (invec'length < 1) then
05277         return result;                                -- string literal value
05278     else
05279         invec := cleanvec(arg);
05280         if (right_index > arghigh) then    -- return top zeros
05281             if (arg'low /= INTEGER'low) then   -- check for a literal
05282                 result := (others => arg(arhigh));           -- sign extend
05283             end if;
05284             needs_rounding := (round_style = fixed_round) and
05285                 (right_index = arghigh+1);
05286         elsif (left_index < arglow) then    -- return overflow
05287             if (overflow_style = fixed_saturate) then
05288                 reduced := or_reduce (to_sulv(invec));
05289                 if (reduced = '1') then
05290                     if (invec(arhigh) = '0') then
05291                         -- saturate POSITIVE
05292                         result := saturate (result'high, result'low);
05293                     else
05294                         -- saturate negative
05295                         result := not saturate (result'high, result'low);
05296                     end if;
05297                     -- else return 0 (input was 0)
05298                 end if;
05299                 -- else return 0 (wrap)
05300             end if;
05301         elsif (arhigh > left_index) then
05302             if (invec(arhigh) = '0') then
05303                 reduced := or_reduce (to_sulv(invec(arhigh-1 downto
05304                                         left_index)));
05305                 if overflow_style = fixed_saturate and reduced = '1' then
05306                     -- saturate positive
05307                     result := saturate (result'high, result'low);
05308                 else
05309                     if (right_index > arglow) then
05310                         result := invec (left_index downto right_index);
05311                         needs_rounding := (round_style = fixed_round);
05312                     else
05313                         result (left_index downto arglow) :=
05314                             invec (left_index downto arglow);
05315                         end if;
05316                     end if;
05317                 else
05318                     reduced := and_reduce (to_sulv(invec(arhigh-1 downto
05319                                         left_index)));
05320                     if overflow_style = fixed_saturate and reduced = '0' then
05321                         result := not saturate (result'high, result'low);
05322                     else
05323                         if (right_index > arglow) then
05324                             result := invec (left_index downto right_index);
05325                             needs_rounding := (round_style = fixed_round);
05326                         else
05327                             result (left_index downto arglow) :=
05328                                 invec (left_index downto arglow);
05329                             end if;
05330                         end if;
05331                     end if;
05332                 end if;                                -- arghigh <= integer width
05333                 if (arglow >= right_index) then
05334                     result (arhigh downto arglow) := invec;
05335                 else
05336                     result (arhigh downto right_index) :=
05337                         invec (arhigh downto right_index);
05338                     needs_rounding := (round_style = fixed_round); -- round
05339                 end if;
05340                 if (left_index > arhigh) then -- sign extend
05341                     result(left_index downto arhigh+1) := (others => invec(arhigh));
05342                 end if;
05343             end if;
05344             -- Round result
05345             if (needs_rounding) then
05346                 result := round_fixed (arg          => result,
05347                                         remainder      => invec (right_index-1
05348                                         downto arglow),
05349                                         overflow_style => overflow_style);
05350             end if;
05351             return result;
05352         end if;
05353     end function resize;
05354
05355 -- size_res functions

```

```

05356  -- These functions compute the size from a passed variable named "size_res"
05357  -- The only part of this variable used it it's size, it is never passed
05358  -- to a lower level routine.
05359  function to_ufixed (
05360      arg      : STD_ULOGIC_VECTOR;          -- shifted vector
05361      size_res : UNRESOLVED_ufixed)        -- for size only
05362  return UNRESOLVED_ufixed is
05363  constant fw    : INTEGER := mine (size_res'low, size_res'low); -- catch literals
05364  variable result : UNRESOLVED_ufixed (size_res'left downto fw);
05365 begin
05366  if (result'length < 1 or arg'length < 1) then
05367  return NAUF;
05368  else
05369  result := to_ufixed (arg           => arg,
05370                  left_index  => size_res'high,
05371                  right_index => size_res'low);
05372  return result;
05373 end if;
05374 end function to_ufixed;
05375
05376 function to_sfixed (
05377      arg      : STD_ULOGIC_VECTOR;          -- shifted vector
05378      size_res : UNRESOLVED_sfixed)        -- for size only
05379  return UNRESOLVED_sfixed is
05380  constant fw    : INTEGER := mine (size_res'low, size_res'low); -- catch literals
05381  variable result : UNRESOLVED_sfixed (size_res'left downto fw);
05382 begin
05383  if (result'length < 1 or arg'length < 1) then
05384  return NASF;
05385  else
05386  result := to_sfixed (arg           => arg,
05387                  left_index  => size_res'high,
05388                  right_index => size_res'low);
05389  return result;
05390 end if;
05391 end function to_sfixed;
05392
05393 function to_ufixed (
05394      arg      : NATURAL;                  -- integer
05395      size_res : UNRESOLVED_ufixed;        -- for size only
05396  constant overflow_style : fixed_overflow_style_type :=
05397      fixed_overflow_style;
05398  constant round_style   : fixed_round_style_type   :=
05399      fixed_round_style;
05400  return UNRESOLVED_ufixed is
05401  constant fw    : INTEGER := mine (size_res'low, size_res'low); -- catch literals
05402  variable result : UNRESOLVED_ufixed (size_res'left downto fw);
05403 begin
05404  if (result'length < 1) then
05405  return NAUF;
05406  else
05407  result := to_ufixed (arg           => arg,
05408                  left_index  => size_res'high,
05409                  right_index => size_res'low,
05410                  round_style => round_style,
05411                  overflow_style => overflow_style);
05412  return result;
05413 end if;
05414 end function to_ufixed;
05415
05416 function to_sfixed (
05417      arg      : INTEGER;                  -- integer
05418      size_res : UNRESOLVED_sfixed;        -- for size only
05419  constant overflow_style : fixed_overflow_style_type :=
05420      fixed_overflow_style;
05421  constant round_style   : fixed_round_style_type   :=
05422      fixed_round_style;
05423  return UNRESOLVED_sfixed is
05424  constant fw    : INTEGER := mine (size_res'low, size_res'low); -- catch literals
05425  variable result : UNRESOLVED_sfixed (size_res'left downto fw);
05426 begin
05427  if (result'length < 1) then
05428  return NASF;
05429  else
05430  result := to_sfixed (arg           => arg,
05431                  left_index  => size_res'high,
05432                  right_index => size_res'low,
05433                  round_style => round_style,
05434                  overflow_style => overflow_style);
05435  return result;
05436 end if;
05437 end function to_sfixed;
05438
05439 function to_ufixed (
05440      arg      : REAL;                   -- real
05441      size_res : UNRESOLVED_ufixed;        -- for size only
05442  constant overflow_style : fixed_overflow_style_type :=

```

```

      fixed_overflow_style;
05439   constant round_style    : fixed_round_style_type    := 
fixed_round_style;
05440   constant guard_bits     : NATURAL                  := fixed_guard_bits) -- # of guard
bits
05441   return UNRESOLVED_ufixed is
05442   constant fw      : INTEGER := mine (size_res'low, size_res'low); -- catch literals
05443   variable result : UNRESOLVED_ufixed (size_res'left downto fw);
05444 begin
05445   if (result'length < 1) then
05446     return NAUF;
05447   else
05448     result := to_ufixed (arg          => arg,
05449                           left_index  => size_res'high,
05450                           right_index => size_res'low,
05451                           guard_bits  => guard_bits,
05452                           round_style  => round_style,
05453                           overflow_style => overflow_style);
05454     return result;
05455   end if;
05456 end function to_ufixed;
05457
05458 function to_sfixed (
05459   arg          : REAL;        -- real
05460   size_res     : UNRESOLVED_sfixed; -- for size only
05461   constant overflow_style : fixed_overflow_style_type := 
fixed_overflow_style;
05462   constant round_style    : fixed_round_style_type    := 
fixed_round_style;
05463   constant guard_bits     : NATURAL                  := fixed_guard_bits) -- # of guard
bits
05464   return UNRESOLVED_sfixed is
05465   constant fw      : INTEGER := mine (size_res'low, size_res'low); -- catch literals
05466   variable result : UNRESOLVED_sfixed (size_res'left downto fw);
05467 begin
05468   if (result'length < 1) then
05469     return NASF;
05470   else
05471     result := to_sfixed (arg          => arg,
05472                           left_index  => size_res'high,
05473                           right_index => size_res'low,
05474                           guard_bits  => guard_bits,
05475                           round_style  => round_style,
05476                           overflow_style => overflow_style);
05477     return result;
05478   end if;
05479 end function to_sfixed;
05480
05481 function to_ufixed (
05482   arg          : UNSIGNED;   -- unsigned
05483   size_res     : UNRESOLVED_ufixed; -- for size only
05484   constant overflow_style : fixed_overflow_style_type := 
fixed_overflow_style;
05485   constant round_style    : fixed_round_style_type    := 
fixed_round_style)
05486   return UNRESOLVED_ufixed is
05487   constant fw      : INTEGER := mine (size_res'low, size_res'low); -- catch literals
05488   variable result : UNRESOLVED_ufixed (size_res'left downto fw);
05489 begin
05490   if (result'length < 1 or arg'length < 1) then
05491     return NAUF;
05492   else
05493     result := to_ufixed (arg          => arg,
05494                           left_index  => size_res'high,
05495                           right_index => size_res'low,
05496                           round_style  => round_style,
05497                           overflow_style => overflow_style);
05498     return result;
05499   end if;
05500 end function to_ufixed;
05501
05502 function to_sfixed (
05503   arg          : SIGNED;    -- signed
05504   size_res     : UNRESOLVED_sfixed; -- for size only
05505   constant overflow_style : fixed_overflow_style_type := 
fixed_overflow_style;
05506   constant round_style    : fixed_round_style_type    := 
fixed_round_style)
05507   return UNRESOLVED_sfixed is
05508   constant fw      : INTEGER := mine (size_res'low, size_res'low); -- catch literals
05509   variable result : UNRESOLVED_sfixed (size_res'left downto fw);
05510 begin
05511   if (result'length < 1 or arg'length < 1) then
05512     return NASF;
05513   else
05514     result := to_sfixed (arg          => arg,
05515                           left_index  => size_res'high,

```

```

05516                               right_index    => size_res'low,
05517                               round_style   => round_style,
05518                               overflow_style => overflow_style);
05519         return result;
05520     end if;
05521 end function to_sfixed;
05522
05523 function resize (
05524     arg                  : UNRESOLVED_ufixed; -- input
05525     size_res              : UNRESOLVED_ufixed; -- for size only
05526     constant overflow_style : fixed_overflow_style_type :=  

05527         fixed_overflow_style;
05528     constant round_style   : fixed_round_style_type      :=  

05529         fixed_round_style);
05530     return UNRESOLVED_ufixed is
05531     constant fw      : INTEGER := mine (size_res'low, size_res'low); -- catch literals
05532     variable result : UNRESOLVED_ufixed (size_res'high downto fw);
05533 begin
05534     if (result'length < 1 or arg'length < 1) then
05535         return NAUF;
05536     else
05537         result := resize (arg
05538             left_index      => size_res'high,
05539             right_index     => size_res'low,
05540             round_style     => round_style,
05541             overflow_style  => overflow_style);
05542     end if;
05543 end function resize;
05544
05545 function resize (
05546     arg                  : UNRESOLVED_sfixed; -- input
05547     size_res              : UNRESOLVED_sfixed; -- for size only
05548     constant overflow_style : fixed_overflow_style_type :=  

05549         fixed_overflow_style;
05550     constant round_style   : fixed_round_style_type      :=  

05551         fixed_round_style);
05552     return UNRESOLVED_sfixed is
05553     constant fw      : INTEGER := mine (size_res'low, size_res'low); -- catch literals
05554     variable result : UNRESOLVED_sfixed (size_res'high downto fw);
05555 begin
05556     if (result'length < 1 or arg'length < 1) then
05557         return NASF;
05558     else
05559         result := resize (arg
05560             left_index      => size_res'high,
05561             right_index     => size_res'low,
05562             round_style     => round_style,
05563             overflow_style  => overflow_style);
05564     end if;
05565 end function resize;
05566
05567 -- Overloaded math functions for real
05568 function "+" (
05569     l : UNRESOLVED_ufixed;           -- fixed point input
05570     r : REAL)
05571     return UNRESOLVED_ufixed is
05572 begin
05573     return (l + to_ufixed (r, l'high, l'low));
05574 end function "+";
05575
05576 function "+" (
05577     l : REAL;
05578     r : UNRESOLVED_ufixed)           -- fixed point input
05579     return UNRESOLVED_ufixed is
05580 begin
05581     return (to_ufixed (l, r'high, r'low) + r);
05582 end function "+";
05583
05584 function "+" (
05585     l : UNRESOLVED_sfixed;           -- fixed point input
05586     r : REAL)
05587     return UNRESOLVED_sfixed is
05588 begin
05589     return (l + to_sfixed (r, l'high, l'low));
05590 end function "+";
05591
05592 function "-" (
05593     l : REAL;
05594     r : UNRESOLVED_sfixed)           -- fixed point input
05595     return UNRESOLVED_sfixed is
05596 begin
05597     return (to_sfixed (l, r'high, r'low) + r);
05598 end function "-";
05599

```

```

05599      l : UNRESOLVED_ufixed;           -- fixed point input
05600      r : REAL);
05601      return UNRESOLVED_ufixed is
05602 begin
05603      return (l - to_ufixed (r, l'high, l'low));
05604 end function "-";
05605
05606 function "-" (
05607      l : REAL;
05608      r : UNRESOLVED_ufixed)           -- fixed point input
05609      return UNRESOLVED_ufixed is
05610 begin
05611      return (to_ufixed (l, r'high, r'low) - r);
05612 end function "-";
05613
05614 function "-" (
05615      l : UNRESOLVED_sfixed;           -- fixed point input
05616      r : REAL);
05617      return UNRESOLVED_sfixed is
05618 begin
05619      return (l - to_sfixed (r, l'high, l'low));
05620 end function "-";
05621
05622 function "-" (
05623      l : REAL;
05624      r : UNRESOLVED_sfixed)           -- fixed point input
05625      return UNRESOLVED_sfixed is
05626 begin
05627      return (to_sfixed (l, r'high, r'low) - r);
05628 end function "-";
05629
05630 function "*" (
05631      l : UNRESOLVED_ufixed;           -- fixed point input
05632      r : REAL);
05633      return UNRESOLVED_ufixed is
05634 begin
05635      return (l * to_ufixed (r, l'high, l'low));
05636 end function "*";
05637
05638 function "*" (
05639      l : REAL;
05640      r : UNRESOLVED_ufixed)           -- fixed point input
05641      return UNRESOLVED_ufixed is
05642 begin
05643      return (to_ufixed (l, r'high, r'low) * r);
05644 end function "*";
05645
05646 function "*" (
05647      l : UNRESOLVED_sfixed;           -- fixed point input
05648      r : REAL);
05649      return UNRESOLVED_sfixed is
05650 begin
05651      return (l * to_sfixed (r, l'high, l'low));
05652 end function "*";
05653
05654 function "*" (
05655      l : REAL;
05656      r : UNRESOLVED_sfixed)           -- fixed point input
05657      return UNRESOLVED_sfixed is
05658 begin
05659      return (to_sfixed (l, r'high, r'low) * r);
05660 end function "*";
05661
05662 function "/" (
05663      l : UNRESOLVED_ufixed;           -- fixed point input
05664      r : REAL);
05665      return UNRESOLVED_ufixed is
05666 begin
05667      return (l / to_ufixed (r, l'high, l'low));
05668 end function "/";
05669
05670 function "/" (
05671      l : REAL;
05672      r : UNRESOLVED_ufixed)           -- fixed point input
05673      return UNRESOLVED_ufixed is
05674 begin
05675      return (to_ufixed (l, r'high, r'low) / r);
05676 end function "/";
05677
05678 function "/" (
05679      l : UNRESOLVED_sfixed;           -- fixed point input
05680      r : REAL);
05681      return UNRESOLVED_sfixed is
05682 begin
05683      return (l / to_sfixed (r, l'high, l'low));
05684 end function "/";
05685

```

```

05686   function "/" (
05687     l : REAL;
05688     r : UNRESOLVED_sfixed)           -- fixed point input
05689   return UNRESOLVED_sfixed is
05690 begin
05691   return (to_sfixed (l, r'high, r'low) / r);
05692 end function "/";
05693
05694   function "rem" (
05695     l : UNRESOLVED_ufixed;           -- fixed point input
05696     r : REAL)
05697   return UNRESOLVED_ufixed is
05698 begin
05699   return (l rem to_ufixed (r, l'high, l'low));
05700 end function "rem";
05701
05702   function "rem" (
05703     l : REAL;
05704     r : UNRESOLVED_ufixed)           -- fixed point input
05705   return UNRESOLVED_ufixed is
05706 begin
05707   return (to_ufixed (l, r'high, r'low) rem r);
05708 end function "rem";
05709
05710   function "rem" (
05711     l : UNRESOLVED_sfixed;           -- fixed point input
05712     r : REAL)
05713   return UNRESOLVED_sfixed is
05714 begin
05715   return (l rem to_sfixed (r, l'high, l'low));
05716 end function "rem";
05717
05718   function "rem" (
05719     l : REAL;
05720     r : UNRESOLVED_sfixed)           -- fixed point input
05721   return UNRESOLVED_sfixed is
05722 begin
05723   return (to_sfixed (l, r'high, r'low) rem r);
05724 end function "rem";
05725
05726   function "mod" (
05727     l : UNRESOLVED_ufixed;           -- fixed point input
05728     r : REAL)
05729   return UNRESOLVED_ufixed is
05730 begin
05731   return (l mod to_ufixed (r, l'high, l'low));
05732 end function "mod";
05733
05734   function "mod" (
05735     l : REAL;
05736     r : UNRESOLVED_ufixed)           -- fixed point input
05737   return UNRESOLVED_ufixed is
05738 begin
05739   return (to_ufixed (l, r'high, r'low) mod r);
05740 end function "mod";
05741
05742   function "mod" (
05743     l : UNRESOLVED_sfixed;           -- fixed point input
05744     r : REAL)
05745   return UNRESOLVED_sfixed is
05746 begin
05747   return (l mod to_sfixed (r, l'high, l'low));
05748 end function "mod";
05749
05750   function "mod" (
05751     l : REAL;
05752     r : UNRESOLVED_sfixed)           -- fixed point input
05753   return UNRESOLVED_sfixed is
05754 begin
05755   return (to_sfixed (l, r'high, r'low) mod r);
05756 end function "mod";
05757
05758 -- Overloaded math functions for integers
05759   function "+" (
05760     l : UNRESOLVED_ufixed;           -- fixed point input
05761     r : NATURAL)
05762   return UNRESOLVED_ufixed is
05763 begin
05764   return (l + to_ufixed (r, l'high, 0));
05765 end function "+";
05766
05767   function "+" (
05768     l : NATURAL;
05769     r : UNRESOLVED_ufixed)           -- fixed point input
05770   return UNRESOLVED_ufixed is
05771 begin
05772   return (to_ufixed (l, r'high, 0) + r);

```

```

05773  end function "+";
05774
05775  function "+" (
05776    l : UNRESOLVED_sfixed;           -- fixed point input
05777    r : INTEGER)
05778    return UNRESOLVED_sfixed is
05779  begin
05780    return (l + to_sfixed (r, l'high, 0));
05781  end function "+";
05782
05783  function "+" (
05784    l : INTEGER;
05785    r : UNRESOLVED_sfixed)          -- fixed point input
05786    return UNRESOLVED_sfixed is
05787  begin
05788    return (to_sfixed (l, r'high, 0) + r);
05789  end function "+";
05790
05791  -- Overloaded functions
05792  function "-" (
05793    l : UNRESOLVED_ufixed;          -- fixed point input
05794    r : NATURAL)
05795    return UNRESOLVED_ufixed is
05796  begin
05797    return (l - to_ufixed (r, l'high, 0));
05798  end function "-";
05799
05800  function "-" (
05801    l : NATURAL;
05802    r : UNRESOLVED_ufixed)          -- fixed point input
05803    return UNRESOLVED_ufixed is
05804  begin
05805    return (to_ufixed (l, r'high, 0) - r);
05806  end function "-";
05807
05808  function "-" (
05809    l : UNRESOLVED_sfixed;          -- fixed point input
05810    r : INTEGER)
05811    return UNRESOLVED_sfixed is
05812  begin
05813    return (l - to_sfixed (r, l'high, 0));
05814  end function "-";
05815
05816  function "-" (
05817    l : INTEGER;
05818    r : UNRESOLVED_sfixed)          -- fixed point input
05819    return UNRESOLVED_sfixed is
05820  begin
05821    return (to_sfixed (l, r'high, 0) - r);
05822  end function "-";
05823
05824  -- Overloaded functions
05825  function "*" (
05826    l : UNRESOLVED_ufixed;          -- fixed point input
05827    r : NATURAL)
05828    return UNRESOLVED_ufixed is
05829  begin
05830    return (l * to_ufixed (r, l'high, 0));
05831  end function "*";
05832
05833  function "*" (
05834    l : NATURAL;
05835    r : UNRESOLVED_ufixed)          -- fixed point input
05836    return UNRESOLVED_ufixed is
05837  begin
05838    return (to_ufixed (l, r'high, 0) * r);
05839  end function "*";
05840
05841  function "*" (
05842    l : UNRESOLVED_sfixed;          -- fixed point input
05843    r : INTEGER)
05844    return UNRESOLVED_sfixed is
05845  begin
05846    return (l * to_sfixed (r, l'high, 0));
05847  end function "*";
05848
05849  function "*" (
05850    l : INTEGER;
05851    r : UNRESOLVED_sfixed)          -- fixed point input
05852    return UNRESOLVED_sfixed is
05853  begin
05854    return (to_sfixed (l, r'high, 0) * r);
05855  end function "*";
05856
05857  -- Overloaded functions
05858  function "/" (
05859    l : UNRESOLVED_ufixed)          -- fixed point input

```

```

05860      r : NATURAL)
05861      return UNRESOLVED_ufixed is
05862 begin
05863      return (l / to_ufixed (r, l'high, 0));
05864 end function "/";
05865
05866 function "/" (
05867   l : NATURAL;
05868   r : UNRESOLVED_ufixed)           -- fixed point input
05869   return UNRESOLVED_ufixed is
05870 begin
05871   return (to_ufixed (l, r'high, 0) / r);
05872 end function "/";
05873
05874 function "/" (
05875   l : UNRESOLVED_sfixed;          -- fixed point input
05876   r : INTEGER)
05877   return UNRESOLVED_sfixed is
05878 begin
05879   return (l / to_sfixed (r, l'high, 0));
05880 end function "/";
05881
05882 function "/" (
05883   l : INTEGER;
05884   r : UNRESOLVED_sfixed)          -- fixed point input
05885   return UNRESOLVED_sfixed is
05886 begin
05887   return (to_sfixed (l, r'high, 0) / r);
05888 end function "/";
05889
05890 function "rem" (
05891   l : UNRESOLVED_ufixed;          -- fixed point input
05892   r : NATURAL)
05893   return UNRESOLVED_ufixed is
05894 begin
05895   return (l rem to_ufixed (r, l'high, 0));
05896 end function "rem";
05897
05898 function "rem" (
05899   l : NATURAL;
05900   r : UNRESOLVED_ufixed)          -- fixed point input
05901   return UNRESOLVED_ufixed is
05902 begin
05903   return (to_ufixed (l, r'high, 0) rem r);
05904 end function "rem";
05905
05906 function "rem" (
05907   l : UNRESOLVED_sfixed;          -- fixed point input
05908   r : INTEGER)
05909   return UNRESOLVED_sfixed is
05910 begin
05911   return (l rem to_sfixed (r, l'high, 0));
05912 end function "rem";
05913
05914 function "rem" (
05915   l : INTEGER;
05916   r : UNRESOLVED_sfixed)          -- fixed point input
05917   return UNRESOLVED_sfixed is
05918 begin
05919   return (to_sfixed (l, r'high, 0) rem r);
05920 end function "rem";
05921
05922 function "mod" (
05923   l : UNRESOLVED_ufixed;          -- fixed point input
05924   r : NATURAL)
05925   return UNRESOLVED_ufixed is
05926 begin
05927   return (l mod to_ufixed (r, l'high, 0));
05928 end function "mod";
05929
05930 function "mod" (
05931   l : NATURAL;
05932   r : UNRESOLVED_ufixed)          -- fixed point input
05933   return UNRESOLVED_ufixed is
05934 begin
05935   return (to_ufixed (l, r'high, 0) mod r);
05936 end function "mod";
05937
05938 function "mod" (
05939   l : UNRESOLVED_sfixed;          -- fixed point input
05940   r : INTEGER)
05941   return UNRESOLVED_sfixed is
05942 begin
05943   return (l mod to_sfixed (r, l'high, 0));
05944 end function "mod";
05945
05946 function "mod" (

```

```

05947      l : INTEGER;
05948      r : UNRESOLVED_sfixed)                      -- fixed point input
05949      return UNRESOLVED_sfixed is
05950 begin
05951      return (to_sfixed (l, r'high, 0) mod r);
05952 end function "mod";
05953
05954 -- overloaded ufixed compare functions with integer
05955 function "==" (
05956      l : UNRESOLVED_ufixed;                         -- fixed point input
05957      r : NATURAL)
05958      return BOOLEAN is
05959 begin
05960      return (l = to_ufixed (r, l'high, l'low));
05961 end function "==";
05962
05963 function "/=" (
05964      l : UNRESOLVED_ufixed;                         -- fixed point input
05965      r : NATURAL)
05966      return BOOLEAN is
05967 begin
05968      return (l /= to_ufixed (r, l'high, l'low));
05969 end function "/=";
05970
05971 function ">=" (
05972      l : UNRESOLVED_ufixed;                         -- fixed point input
05973      r : NATURAL)
05974      return BOOLEAN is
05975 begin
05976      return (l >= to_ufixed (r, l'high, l'low));
05977 end function ">=";
05978
05979 function "<=" (
05980      l : UNRESOLVED_ufixed;                         -- fixed point input
05981      r : NATURAL)
05982      return BOOLEAN is
05983 begin
05984      return (l <= to_ufixed (r, l'high, l'low));
05985 end function "<=";
05986
05987 function ">" (
05988      l : UNRESOLVED_ufixed;                         -- fixed point input
05989      r : NATURAL)
05990      return BOOLEAN is
05991 begin
05992      return (l > to_ufixed (r, l'high, l'low));
05993 end function ">";
05994
05995 function "<" (
05996      l : UNRESOLVED_ufixed;                         -- fixed point input
05997      r : NATURAL)
05998      return BOOLEAN is
05999 begin
06000      return (l < to_ufixed (r, l'high, l'low));
06001 end function "<";
06002
06003 function "\?=" (
06004      l : UNRESOLVED_ufixed;                         -- fixed point input
06005      r : NATURAL)
06006      return STD_ULOGIC is
06007 begin
06008      return \?=(\ (l, to_ufixed (r, l'high, l'low));
06009 end function "\?=";
06010
06011 function "\?/=" (
06012      l : UNRESOLVED_ufixed;                         -- fixed point input
06013      r : NATURAL)
06014      return STD_ULOGIC is
06015 begin
06016      return \?/=(\ (l, to_ufixed (r, l'high, l'low));
06017 end function "\?/=";
06018
06019 function "\?>=" (
06020      l : UNRESOLVED_ufixed;                         -- fixed point input
06021      r : NATURAL)
06022      return STD_ULOGIC is
06023 begin
06024      return \?>=(\ (l, to_ufixed (r, l'high, l'low));
06025 end function "\?>=";
06026
06027 function "\?<=" (
06028      l : UNRESOLVED_ufixed;                         -- fixed point input
06029      r : NATURAL)
06030      return STD_ULOGIC is
06031 begin
06032      return \?<=(\ (l, to_ufixed (r, l'high, l'low));
06033 end function "\?<=";

```

```

06034
06035   function \?>\ (
06036     l : UNRESOLVED_ufixed;
06037     r : NATURAL)                      -- fixed point input
06038     return STD_ULOGIC is
06039   begin
06040     return \?>\(l, to_ufixed(r, l'high, l'low));
06041   end function \?>\;
06042
06043   function \?<\ (
06044     l : UNRESOLVED_ufixed;
06045     r : NATURAL)                      -- fixed point input
06046     return STD_ULOGIC is
06047   begin
06048     return \?<\(l, to_ufixed(r, l'high, l'low));
06049   end function \?<\;
06050
06051   function maximum (
06052     l : UNRESOLVED_ufixed;           -- fixed point input
06053     r : NATURAL)
06054     return UNRESOLVED_ufixed is
06055   begin
06056     return maximum(l, to_ufixed(r, l'high, l'low));
06057   end function maximum;
06058
06059   function minimum (
06060     l : UNRESOLVED_ufixed;           -- fixed point input
06061     r : NATURAL)
06062     return UNRESOLVED_ufixed is
06063   begin
06064     return minimum(l, to_ufixed(r, l'high, l'low));
06065   end function minimum;
06066
06067   -- NATURAL to ufixed
06068   function "=" (
06069     l : NATURAL;
06070     r : UNRESOLVED_ufixed)          -- fixed point input
06071     return BOOLEAN is
06072   begin
06073     return (to_ufixed(l, r'high, r'low) = r);
06074   end function "=";
06075
06076   function "/=" (
06077     l : NATURAL;
06078     r : UNRESOLVED_ufixed)          -- fixed point input
06079     return BOOLEAN is
06080   begin
06081     return (to_ufixed(l, r'high, r'low) /= r);
06082   end function "/=";
06083
06084   function ">=" (
06085     l : NATURAL;
06086     r : UNRESOLVED_ufixed)          -- fixed point input
06087     return BOOLEAN is
06088   begin
06089     return (to_ufixed(l, r'high, r'low) >= r);
06090   end function ">=";
06091
06092   function "<=" (
06093     l : NATURAL;
06094     r : UNRESOLVED_ufixed)          -- fixed point input
06095     return BOOLEAN is
06096   begin
06097     return (to_ufixed(l, r'high, r'low) <= r);
06098   end function "<=";
06099
06100   function ">" (
06101     l : NATURAL;
06102     r : UNRESOLVED_ufixed)          -- fixed point input
06103     return BOOLEAN is
06104   begin
06105     return (to_ufixed(l, r'high, r'low) > r);
06106   end function ">";
06107
06108   function "<" (
06109     l : NATURAL;
06110     r : UNRESOLVED_ufixed)          -- fixed point input
06111     return BOOLEAN is
06112   begin
06113     return (to_ufixed(l, r'high, r'low) < r);
06114   end function "<";
06115
06116   function \?=\
06117     l : NATURAL;
06118     r : UNRESOLVED_ufixed)          -- fixed point input
06119     return STD_ULOGIC is
06120   begin

```

```

06121      return \?=\ (to_ufixed (l, r'high, r'low), r);
06122  end function \?=\;
06123
06124  function \?/=\ (
06125    l : NATURAL;
06126    r : UNRESOLVED_ufixed)                      -- fixed point input
06127    return STD_ULONGIC is
06128  begin
06129    return \?/=\ (to_ufixed (l, r'high, r'low), r);
06130  end function \?/=\;
06131
06132  function \?>=\
06133    l : NATURAL;
06134    r : UNRESOLVED_ufixed)                      -- fixed point input
06135    return STD_ULONGIC is
06136  begin
06137    return \?>= (to_ufixed (l, r'high, r'low), r);
06138  end function \?>=;
06139
06140  function \?<=
06141    l : NATURAL;
06142    r : UNRESOLVED_ufixed)                      -- fixed point input
06143    return STD_ULONGIC is
06144  begin
06145    return \?<= (to_ufixed (l, r'high, r'low), r);
06146  end function \?<=;
06147
06148  function \?> \
06149    l : NATURAL;
06150    r : UNRESOLVED_ufixed)                      -- fixed point input
06151    return STD_ULONGIC is
06152  begin
06153    return \?>\ (to_ufixed (l, r'high, r'low), r);
06154  end function \?>\;
06155
06156  function \?<\ (
06157    l : NATURAL;
06158    r : UNRESOLVED_ufixed)                      -- fixed point input
06159    return STD_ULONGIC is
06160  begin
06161    return \?<\ (to_ufixed (l, r'high, r'low), r);
06162  end function \?<\;
06163
06164  function maximum (
06165    l : NATURAL;
06166    r : UNRESOLVED_ufixed)                      -- fixed point input
06167    return UNRESOLVED_ufixed is
06168  begin
06169    return maximum (to_ufixed (l, r'high, r'low), r);
06170  end function maximum;
06171
06172  function minimum (
06173    l : NATURAL;
06174    r : UNRESOLVED_ufixed)                      -- fixed point input
06175    return UNRESOLVED_ufixed is
06176  begin
06177    return minimum (to_ufixed (l, r'high, r'low), r);
06178  end function minimum;
06179
06180  -- overloaded ufixed compare functions with real
06181  function "==" (
06182    l : UNRESOLVED_ufixed;
06183    r : REAL)
06184    return BOOLEAN is
06185  begin
06186    return (l = to_ufixed (r, l'high, l'low));
06187  end function "==";
06188
06189  function "/=" (
06190    l : UNRESOLVED_ufixed;
06191    r : REAL)
06192    return BOOLEAN is
06193  begin
06194    return (l /= to_ufixed (r, l'high, l'low));
06195  end function "/=";
06196
06197  function ">=" (
06198    l : UNRESOLVED_ufixed;
06199    r : REAL)
06200    return BOOLEAN is
06201  begin
06202    return (l >= to_ufixed (r, l'high, l'low));
06203  end function ">=";
06204
06205  function "<=" (
06206    l : UNRESOLVED_ufixed;
06207    r : REAL)

```

```

06208      return BOOLEAN is
06209      begin
06210          return (l <= to_ufixed (r, l'high, l'low));
06211      end function "<=";
06212
06213      function ">" (
06214          l : UNRESOLVED_ufixed;
06215          r : REAL)
06216          return BOOLEAN is
06217          begin
06218              return (l > to_ufixed (r, l'high, l'low));
06219          end function ">";
06220
06221      function "<" (
06222          l : UNRESOLVED_ufixed;
06223          r : REAL)
06224          return BOOLEAN is
06225          begin
06226              return (l < to_ufixed (r, l'high, l'low));
06227          end function "<";
06228
06229      function \?=\ (
06230          l : UNRESOLVED_ufixed;
06231          r : REAL)
06232          return STD_ULOGIC is
06233          begin
06234              return \?=\ (l, to_ufixed (r, l'high, l'low));
06235          end function \?=\;
06236
06237      function \?/=\ (
06238          l : UNRESOLVED_ufixed;
06239          r : REAL)
06240          return STD_ULOGIC is
06241          begin
06242              return \?/=\ (l, to_ufixed (r, l'high, l'low));
06243          end function \?/=\;
06244
06245      function \?>=\
06246          l : UNRESOLVED_ufixed;
06247          r : REAL)
06248          return STD_ULOGIC is
06249          begin
06250              return \?>=\ (l, to_ufixed (r, l'high, l'low));
06251          end function \?>=;
06252
06253      function \?<=
06254          l : UNRESOLVED_ufixed;
06255          r : REAL)
06256          return STD_ULOGIC is
06257          begin
06258              return \?<=\ (l, to_ufixed (r, l'high, l'low));
06259          end function \?<=;
06260
06261      function \?> \
06262          l : UNRESOLVED_ufixed;
06263          r : REAL)
06264          return STD_ULOGIC is
06265          begin
06266              return \?>\ (l, to_ufixed (r, l'high, l'low));
06267          end function \?>\;
06268
06269      function \?< \
06270          l : UNRESOLVED_ufixed;
06271          r : REAL)
06272          return STD_ULOGIC is
06273          begin
06274              return \?<\ (l, to_ufixed (r, l'high, l'low));
06275          end function \?<\;
06276
06277      function maximum (
06278          l : UNRESOLVED_ufixed;
06279          r : REAL)
06280          return UNRESOLVED_ufixed is
06281          begin
06282              return maximum (l, to_ufixed (r, l'high, l'low));
06283          end function maximum;
06284
06285      function minimum (
06286          l : UNRESOLVED_ufixed;
06287          r : REAL)
06288          return UNRESOLVED_ufixed is
06289          begin
06290              return minimum (l, to_ufixed (r, l'high, l'low));
06291          end function minimum;
06292
06293      -- real and ufixed
06294      function "=" (

```

```

06295      l : REAL;
06296      r : UNRESOLVED_ufixed)          -- fixed point input
06297      return BOOLEAN is
06298      begin
06299          return (to_ufixed (l, r'high, r'low) = r);
06300      end function "=";
06301
06302      function "/=" (
06303          l : REAL;
06304          r : UNRESOLVED_ufixed)          -- fixed point input
06305          return BOOLEAN is
06306          begin
06307              return (to_ufixed (l, r'high, r'low) /= r);
06308          end function "/=";
06309
06310      function ">=" (
06311          l : REAL;
06312          r : UNRESOLVED_ufixed)          -- fixed point input
06313          return BOOLEAN is
06314          begin
06315              return (to_ufixed (l, r'high, r'low) >= r);
06316          end function ">=";
06317
06318      function "<=" (
06319          l : REAL;
06320          r : UNRESOLVED_ufixed)          -- fixed point input
06321          return BOOLEAN is
06322          begin
06323              return (to_ufixed (l, r'high, r'low) <= r);
06324          end function "<=";
06325
06326      function ">" (
06327          l : REAL;
06328          r : UNRESOLVED_ufixed)          -- fixed point input
06329          return BOOLEAN is
06330          begin
06331              return (to_ufixed (l, r'high, r'low) > r);
06332          end function ">";
06333
06334      function "<" (
06335          l : REAL;
06336          r : UNRESOLVED_ufixed)          -- fixed point input
06337          return BOOLEAN is
06338          begin
06339              return (to_ufixed (l, r'high, r'low) < r);
06340          end function "<";
06341
06342      function "\?=\\" (
06343          l : REAL;
06344          r : UNRESOLVED_ufixed)          -- fixed point input
06345          return STD_ULOGIC is
06346          begin
06347              return \?=\ (to_ufixed (l, r'high, r'low), r);
06348          end function "\?=\";
06349
06350      function "\?/=\\" (
06351          l : REAL;
06352          r : UNRESOLVED_ufixed)          -- fixed point input
06353          return STD_ULOGIC is
06354          begin
06355              return \?/=\ (to_ufixed (l, r'high, r'low), r);
06356          end function "\?/=\";
06357
06358      function "\?>=\\" (
06359          l : REAL;
06360          r : UNRESOLVED_ufixed)          -- fixed point input
06361          return STD_ULOGIC is
06362          begin
06363              return \?>=\ (to_ufixed (l, r'high, r'low), r);
06364          end function "\?>=";
06365
06366      function "\?<=\\" (
06367          l : REAL;
06368          r : UNRESOLVED_ufixed)          -- fixed point input
06369          return STD_ULOGIC is
06370          begin
06371              return \?<=\ (to_ufixed (l, r'high, r'low), r);
06372          end function "\?<=";
06373
06374      function "\?>\\" (
06375          l : REAL;
06376          r : UNRESOLVED_ufixed)          -- fixed point input
06377          return STD_ULOGIC is
06378          begin
06379              return \?>\ (to_ufixed (l, r'high, r'low), r);
06380          end function "\?>";
06381

```

```

06382   function \?<\ (
06383     l : REAL;
06384     r : UNRESOLVED_ufixed           -- fixed point input
06385     return STD_ULOGIC is
06386   begin
06387     return \?<\ (to_ufixed (l, r'high, r'low), r);
06388   end function \?<\;
06389
06390   function maximum (
06391     l : REAL;
06392     r : UNRESOLVED_ufixed           -- fixed point input
06393     return UNRESOLVED_ufixed is
06394   begin
06395     return maximum (to_ufixed (l, r'high, r'low), r);
06396   end function maximum;
06397
06398   function minimum (
06399     l : REAL;
06400     r : UNRESOLVED_ufixed           -- fixed point input
06401     return UNRESOLVED_ufixed is
06402   begin
06403     return minimum (to_ufixed (l, r'high, r'low), r);
06404   end function minimum;
06405
06406   -- overloaded sfixed compare functions with integer
06407   function "=" (
06408     l : UNRESOLVED_sfixed;
06409     r : INTEGER)
06410     return BOOLEAN is
06411   begin
06412     return (l = to_sfixed (r, l'high, l'low));
06413   end function "=";
06414
06415   function "/=" (
06416     l : UNRESOLVED_sfixed;
06417     r : INTEGER)
06418     return BOOLEAN is
06419   begin
06420     return (l /= to_sfixed (r, l'high, l'low));
06421   end function "/=";
06422
06423   function ">=" (
06424     l : UNRESOLVED_sfixed;
06425     r : INTEGER)
06426     return BOOLEAN is
06427   begin
06428     return (l >= to_sfixed (r, l'high, l'low));
06429   end function ">=";
06430
06431   function "<=" (
06432     l : UNRESOLVED_sfixed;
06433     r : INTEGER)
06434     return BOOLEAN is
06435   begin
06436     return (l <= to_sfixed (r, l'high, l'low));
06437   end function "<=";
06438
06439   function ">" (
06440     l : UNRESOLVED_sfixed;
06441     r : INTEGER)
06442     return BOOLEAN is
06443   begin
06444     return (l > to_sfixed (r, l'high, l'low));
06445   end function ">";
06446
06447   function "<" (
06448     l : UNRESOLVED_sfixed;
06449     r : INTEGER)
06450     return BOOLEAN is
06451   begin
06452     return (l < to_sfixed (r, l'high, l'low));
06453   end function "<";
06454
06455   function \?=\ (
06456     l : UNRESOLVED_sfixed;
06457     r : INTEGER)
06458     return STD_ULOGIC is
06459   begin
06460     return \?=\ (l, to_sfixed (r, l'high, l'low));
06461   end function \?=\;
06462
06463   function \?/=\ (
06464     l : UNRESOLVED_sfixed;
06465     r : INTEGER)
06466     return STD_ULOGIC is
06467   begin
06468     return \?/=\ (l, to_sfixed (r, l'high, l'low));

```

```

06469  end function \?/=\;
06470
06471  function \?>=\(
06472    l : UNRESOLVED_sfixed;
06473    r : INTEGER)
06474    return STD_ULONGIC is
06475 begin
06476    return \?>=\(l, to_sfixed (r, l'high, l'low));
06477 end function \?>=`;
06478
06479  function \?<=\(
06480    l : UNRESOLVED_sfixed;
06481    r : INTEGER)
06482    return STD_ULONGIC is
06483 begin
06484    return \?<=\(l, to_sfixed (r, l'high, l'low));
06485 end function \?<=`;
06486
06487  function \?>\(
06488    l : UNRESOLVED_sfixed;
06489    r : INTEGER)
06490    return STD_ULONGIC is
06491 begin
06492    return \?>\(l, to_sfixed (r, l'high, l'low));
06493 end function \?>\;
06494
06495  function \?<\(
06496    l : UNRESOLVED_sfixed;
06497    r : INTEGER)
06498    return STD_ULONGIC is
06499 begin
06500    return \?<\(l, to_sfixed (r, l'high, l'low));
06501 end function \?<\;
06502
06503  function maximum (
06504    l : UNRESOLVED_sfixed;
06505    r : INTEGER)
06506    return UNRESOLVED_sfixed is
06507 begin
06508    return maximum (l, to_sfixed (r, l'high, l'low));
06509 end function maximum;
06510
06511  function minimum (
06512    l : UNRESOLVED_sfixed;
06513    r : INTEGER)
06514    return UNRESOLVED_sfixed is
06515 begin
06516    return minimum (l, to_sfixed (r, l'high, l'low));
06517 end function minimum;
06518
06519 -- integer and sfixed
06520  function "=" (
06521    l : INTEGER;
06522    r : UNRESOLVED_sfixed) -- fixed point input
06523    return BOOLEAN is
06524 begin
06525    return (to_sfixed (l, r'high, r'low) = r);
06526 end function "=";
06527
06528  function "/=" (
06529    l : INTEGER;
06530    r : UNRESOLVED_sfixed) -- fixed point input
06531    return BOOLEAN is
06532 begin
06533    return (to_sfixed (l, r'high, r'low) /= r);
06534 end function "/=";
06535
06536  function ">=" (
06537    l : INTEGER;
06538    r : UNRESOLVED_sfixed) -- fixed point input
06539    return BOOLEAN is
06540 begin
06541    return (to_sfixed (l, r'high, r'low) >= r);
06542 end function ">=";
06543
06544  function "<=" (
06545    l : INTEGER;
06546    r : UNRESOLVED_sfixed) -- fixed point input
06547    return BOOLEAN is
06548 begin
06549    return (to_sfixed (l, r'high, r'low) <= r);
06550 end function "<=";
06551
06552  function ">" (
06553    l : INTEGER;
06554    r : UNRESOLVED_sfixed) -- fixed point input
06555    return BOOLEAN is

```

```

06556 begin
06557     return (to_sfixed (l, r'high, r'low) > r);
06558 end function ">";
06559
06560 function "<" (
06561     l : INTEGER;
06562     r : UNRESOLVED_sfixed) -- fixed point input
06563     return BOOLEAN is
06564 begin
06565     return (to_sfixed (l, r'high, r'low) < r);
06566 end function "<";
06567
06568 function \?=\ (
06569     l : INTEGER;
06570     r : UNRESOLVED_sfixed) -- fixed point input
06571     return STD_ULONGIC is
06572 begin
06573     return \?=\ (to_sfixed (l, r'high, r'low), r);
06574 end function \?=\;
06575
06576 function \?/=\ (
06577     l : INTEGER;
06578     r : UNRESOLVED_sfixed) -- fixed point input
06579     return STD_ULONGIC is
06580 begin
06581     return \?/=\ (to_sfixed (l, r'high, r'low), r);
06582 end function \?/=\;
06583
06584 function \?>=\
06585     l : INTEGER;
06586     r : UNRESOLVED_sfixed) -- fixed point input
06587     return STD_ULONGIC is
06588 begin
06589     return \?>= (to_sfixed (l, r'high, r'low), r);
06590 end function \?>=;
06591
06592 function \?<=\
06593     l : INTEGER;
06594     r : UNRESOLVED_sfixed) -- fixed point input
06595     return STD_ULONGIC is
06596 begin
06597     return \?<= (to_sfixed (l, r'high, r'low), r);
06598 end function \?<=;
06599
06600 function \?>\
06601     l : INTEGER;
06602     r : UNRESOLVED_sfixed) -- fixed point input
06603     return STD_ULONGIC is
06604 begin
06605     return \?> (to_sfixed (l, r'high, r'low), r);
06606 end function \?>;
06607
06608 function \?<\
06609     l : INTEGER;
06610     r : UNRESOLVED_sfixed) -- fixed point input
06611     return STD_ULONGIC is
06612 begin
06613     return \?< (to_sfixed (l, r'high, r'low), r);
06614 end function \?<;
06615
06616 function maximum (
06617     l : INTEGER;
06618     r : UNRESOLVED_sfixed)
06619     return UNRESOLVED_sfixed is
06620 begin
06621     return maximum (to_sfixed (l, r'high, r'low), r);
06622 end function maximum;
06623
06624 function minimum (
06625     l : INTEGER;
06626     r : UNRESOLVED_sfixed)
06627     return UNRESOLVED_sfixed is
06628 begin
06629     return minimum (to_sfixed (l, r'high, r'low), r);
06630 end function minimum;
06631
06632 -- overloaded sfixed compare functions with real
06633 function "==" (
06634     l : UNRESOLVED_sfixed;
06635     r : REAL)
06636     return BOOLEAN is
06637 begin
06638     return (l = to_sfixed (r, l'high, l'low));
06639 end function "==";
06640
06641 function "/==" (
06642     l : UNRESOLVED_sfixed;

```

```

06643      r : REAL)
06644      return BOOLEAN is
06645 begin
06646      return (l /= to_sfixed (r, l'high, l'low));
06647 end function "/=";
06648
06649 function ">=" (
06650      l : UNRESOLVED_sfixed;
06651      r : REAL)
06652      return BOOLEAN is
06653 begin
06654      return (l >= to_sfixed (r, l'high, l'low));
06655 end function ">=";
06656
06657 function "<=" (
06658      l : UNRESOLVED_sfixed;
06659      r : REAL)
06660      return BOOLEAN is
06661 begin
06662      return (l <= to_sfixed (r, l'high, l'low));
06663 end function "<=";
06664
06665 function ">" (
06666      l : UNRESOLVED_sfixed;
06667      r : REAL)
06668      return BOOLEAN is
06669 begin
06670      return (l > to_sfixed (r, l'high, l'low));
06671 end function ">";
06672
06673 function "<" (
06674      l : UNRESOLVED_sfixed;
06675      r : REAL)
06676      return BOOLEAN is
06677 begin
06678      return (l < to_sfixed (r, l'high, l'low));
06679 end function "<";
06680
06681 function \?=\ (
06682      l : UNRESOLVED_sfixed;
06683      r : REAL)
06684      return STD_ULONGIC is
06685 begin
06686      return \?=\ (l, to_sfixed (r, l'high, l'low));
06687 end function \?=\;
06688
06689 function \?/=\ (
06690      l : UNRESOLVED_sfixed;
06691      r : REAL)
06692      return STD_ULONGIC is
06693 begin
06694      return \?/=\ (l, to_sfixed (r, l'high, l'low));
06695 end function \?/=\;
06696
06697 function \?>=\
06698      l : UNRESOLVED_sfixed;
06699      r : REAL)
06700      return STD_ULONGIC is
06701 begin
06702      return \?>=\ (l, to_sfixed (r, l'high, l'low));
06703 end function \?>=\
06704
06705 function \?<=\
06706      l : UNRESOLVED_sfixed;
06707      r : REAL)
06708      return STD_ULONGIC is
06709 begin
06710      return \?<=\ (l, to_sfixed (r, l'high, l'low));
06711 end function \?<=\
06712
06713 function \?> \
06714      l : UNRESOLVED_sfixed;
06715      r : REAL)
06716      return STD_ULONGIC is
06717 begin
06718      return \?>\ (l, to_sfixed (r, l'high, l'low));
06719 end function \?>|;
06720
06721 function \?< \
06722      l : UNRESOLVED_sfixed;
06723      r : REAL)
06724      return STD_ULONGIC is
06725 begin
06726      return \?<\ (l, to_sfixed (r, l'high, l'low));
06727 end function \?<|;
06728
06729 function maximum (

```

```

06730      l : UNRESOLVED_sfixed;
06731      r : REAL)
06732      return UNRESOLVED_sfixed is
06733 begin
06734      return maximum (l, to_sfixed (r, l'high, l'low));
06735 end function maximum;
06736
06737 function minimum (
06738      l : UNRESOLVED_sfixed;
06739      r : REAL)
06740      return UNRESOLVED_sfixed is
06741 begin
06742      return minimum (l, to_sfixed (r, l'high, l'low));
06743 end function minimum;
06744
06745 -- REAL and sfixed
06746 function "=" (
06747      l : REAL;
06748      r : UNRESOLVED_sfixed)           -- fixed point input
06749      return BOOLEAN is
06750 begin
06751      return (to_sfixed (l, r'high, r'low) = r);
06752 end function "=";
06753
06754 function "/=" (
06755      l : REAL;
06756      r : UNRESOLVED_sfixed)           -- fixed point input
06757      return BOOLEAN is
06758 begin
06759      return (to_sfixed (l, r'high, r'low) /= r);
06760 end function "/=";
06761
06762 function ">=" (
06763      l : REAL;
06764      r : UNRESOLVED_sfixed)           -- fixed point input
06765      return BOOLEAN is
06766 begin
06767      return (to_sfixed (l, r'high, r'low) >= r);
06768 end function ">=";
06769
06770 function "<=" (
06771      l : REAL;
06772      r : UNRESOLVED_sfixed)           -- fixed point input
06773      return BOOLEAN is
06774 begin
06775      return (to_sfixed (l, r'high, r'low) <= r);
06776 end function "<=";
06777
06778 function ">" (
06779      l : REAL;
06780      r : UNRESOLVED_sfixed)           -- fixed point input
06781      return BOOLEAN is
06782 begin
06783      return (to_sfixed (l, r'high, r'low) > r);
06784 end function ">";
06785
06786 function "<" (
06787      l : REAL;
06788      r : UNRESOLVED_sfixed)           -- fixed point input
06789      return BOOLEAN is
06790 begin
06791      return (to_sfixed (l, r'high, r'low) < r);
06792 end function "<";
06793
06794 function \?=\ (
06795      l : REAL;
06796      r : UNRESOLVED_sfixed)           -- fixed point input
06797      return STD_ULOGIC is
06798 begin
06799      return \?=\ (to_sfixed (l, r'high, r'low), r);
06800 end function \?=\;
06801
06802 function \?/=\
06803      l : REAL;
06804      r : UNRESOLVED_sfixed)           -- fixed point input
06805      return STD_ULOGIC is
06806 begin
06807      return \?/=\\ (to_sfixed (l, r'high, r'low), r);
06808 end function \?/=\\;
06809
06810 function \?>=
06811      l : REAL;
06812      r : UNRESOLVED_sfixed)           -- fixed point input
06813      return STD_ULOGIC is
06814 begin
06815      return \?>=\\ (to_sfixed (l, r'high, r'low), r);
06816 end function \?>=\\;

```

```

06817
06818   function \?<=\
06819     l : REAL;
06820     r : UNRESOLVED_sfixed)           -- fixed point input
06821   begin
06822     return \?<=\(to_sfixed (l, r'high, r'low), r);
06823   end function \?<=;
06824
06825
06826   function \?>\
06827     l : REAL;
06828     r : UNRESOLVED_sfixed)           -- fixed point input
06829   begin
06830     return \?>\(to_sfixed (l, r'high, r'low), r);
06831   end function \?>;
06832
06833
06834   function \?<\
06835     l : REAL;
06836     r : UNRESOLVED_sfixed)           -- fixed point input
06837   begin
06838     return \?<\(to_sfixed (l, r'high, r'low), r);
06839   end function \?<;
06840
06841
06842   function maximum (
06843     l : REAL;
06844     r : UNRESOLVED_sfixed)
06845   begin
06846     return maximum (to_sfixed (l, r'high, r'low), r);
06847   end function maximum;
06848
06849
06850   function minimum (
06851     l : REAL;
06852     r : UNRESOLVED_sfixed)
06853   begin
06854     return minimum (to_sfixed (l, r'high, r'low), r);
06855   end function minimum;
06856
06857 -- rtl_synthesis off
06858 -- pragma synthesis_off
06859 -- copied from std_logic_textio
06860 type MVL9plus is ('U', 'X', '0', '1', 'Z', 'W', 'L', 'H', '-', error);
06861 type char_indexed_by_MVL9 is array (STD_ULOGIC) of CHARACTER;
06862 type MVL9_indexed_by_char is array (CHARACTER) of STD_ULOGIC;
06863 type MVL9plus_indexed_by_char is array (CHARACTER) of
06864   MVL9plus;
06865 constant MVL9_to_char : char_indexed_by_MVL9 := "UX01ZWLH-";
06866 constant char_to_MVL9 : MVL9_indexed_by_char :=
06867   ('U' => 'U', 'X' => 'X', '0' => '0', '1' => '1', 'Z' => 'Z',
06868   'W' => 'W', 'L' => 'L', 'H' => 'H', '-' => '-', others => 'U');
06869 constant char_to_MVL9plus : MVL9plus_indexed_by_char :=
06870   ('U' => 'U', 'X' => 'X', '0' => '0', '1' => '1', 'Z' => 'Z',
06871   'W' => 'W', 'L' => 'L', 'H' => 'H', '-' => '-', others => error);
06872 constant NBSP : CHARACTER := CHARACTER'val(160); -- space character
06873 constant NUS : STRING(2 to 1) := (others => ' ');
06874
06875 -- %% Replicated Textio functions
06876 procedure Char2TriBits (C          :      CHARACTER;
06877                           RESULT      : out STD_ULOGIC_VECTOR(2 downto 0);
06878                           GOOD        : out BOOLEAN;
06879                           ISSUE_ERROR : in  BOOLEAN) is
06880 begin
06881   case c is
06882     when '0' => result := o"0"; good := true;
06883     when '1' => result := o"1"; good := true;
06884     when '2' => result := o"2"; good := true;
06885     when '3' => result := o"3"; good := true;
06886     when '4' => result := o"4"; good := true;
06887     when '5' => result := o"5"; good := true;
06888     when '6' => result := o"6"; good := true;
06889     when '7' => result := o"7"; good := true;
06890     when 'Z' => result := "ZZZ"; good := true;
06891     when 'X' => result := "XXX"; good := true;
06892     when others =>
06893       assert not ISSUE_ERROR
06894         report fixed_pkg'instance_name
06895           & "OREAD Error: Read a '" & c &
06896           "'", expected an Octal character (0-7)."
06897           severity error;
06898       result := "UUU";
06899       good  := false;
06900     end case;
06901 end procedure Char2TriBits;
06902 -- Hex Read and Write procedures for STD_ULOGIC_VECTOR.

```

```

06903 -- Modified from the original to be more forgiving.
06904
06905 procedure Char2QuadBits (C           : CHARACTER;
06906                      RESULT      : out STD_ULOGIC_VECTOR(3 downto 0);
06907                      GOOD       : out BOOLEAN;
06908                      ISSUE_ERROR : in  BOOLEAN) is
06909 begin
06910     case c is
06911         when '0'    => result := x"0"; good := true;
06912         when '1'    => result := x"1"; good := true;
06913         when '2'    => result := x"2"; good := true;
06914         when '3'    => result := x"3"; good := true;
06915         when '4'    => result := x"4"; good := true;
06916         when '5'    => result := x"5"; good := true;
06917         when '6'    => result := x"6"; good := true;
06918         when '7'    => result := x"7"; good := true;
06919         when '8'    => result := x"8"; good := true;
06920         when '9'    => result := x"9"; good := true;
06921         when 'A' | 'a' => result := x"A"; good := true;
06922         when 'B' | 'b' => result := x"B"; good := true;
06923         when 'C' | 'c' => result := x"C"; good := true;
06924         when 'D' | 'd' => result := x"D"; good := true;
06925         when 'E' | 'e' => result := x"E"; good := true;
06926         when 'F' | 'f' => result := x"F"; good := true;
06927         when 'Z'    => result := "ZZZZ"; good := true;
06928         when 'X'    => result := "XXXX"; good := true;
06929         when others =>
06930             assert not ISSUE_ERROR
06931                 report fixed_pkg'instance_name
06932                 & "HREAD Error: Read a '" & c &
06933                 & "', expected a Hex character (0-F)."
06934                 severity error;
06935             result := "UUUU";
06936             good   := false;
06937         end case;
06938     end procedure Char2QuadBits;
06939
06940 -- purpose: Skips white space
06941 procedure skip_whitespace (
06942     L : inout LINE) is
06943     variable readOk : BOOLEAN;
06944     variable c : CHARACTER;
06945 begin
06946     while L /= null and L.all'length /= 0 loop
06947         if (L.all(1) = ' ' or L.all(1) = NBSP or L.all(1) = HT) then
06948             read (l, c, readOk);
06949         else
06950             exit;
06951         end if;
06952     end loop;
06953 end procedure skip_whitespace;
06954
06955 function to_ostring (value      : STD_ULOGIC_VECTOR) return STRING is
06956     constant ne      : INTEGER := (value'length+2)/3;
06957     variable pad     : STD_ULOGIC_VECTOR(0 to (ne*3 - value'length) - 1);
06958     variable ivalue : STD_ULOGIC_VECTOR(0 to ne*3 - 1);
06959     variable result : STRING(1 to ne);
06960     variable tri     : STD_ULOGIC_VECTOR(0 to 2);
06961 begin
06962     if value'length < 1 then
06963         return NUS;
06964     else
06965         if value (value'left) = 'Z' then
06966             pad := (others => 'Z');
06967         else
06968             pad := (others => '0');
06969         end if;
06970         ivalue := pad & value;
06971         for i in 0 to ne-1 loop
06972             tri := To_X01Z(ivalue(3*i to 3*i+2));
06973             case tri is
06974                 when o"0"  => result(i+1) := '0';
06975                 when o"1"  => result(i+1) := '1';
06976                 when o"2"  => result(i+1) := '2';
06977                 when o"3"  => result(i+1) := '3';
06978                 when o"4"  => result(i+1) := '4';
06979                 when o"5"  => result(i+1) := '5';
06980                 when o"6"  => result(i+1) := '6';
06981                 when o"7"  => result(i+1) := '7';
06982                 when "ZZZ" => result(i+1) := 'Z';
06983                 when others => result(i+1) := 'X';
06984             end case;
06985         end loop;
06986         return result;
06987     end if;
06988 end function to_ostring;
06989 -----
```

```

06990  function to_hstring (value      : STD_ULOGIC_VECTOR) return STRING is
06991  constant ne      : INTEGER := (value'length+3)/4;
06992  variable pad     : STD_ULOGIC_VECTOR(0 to (ne*4 - value'length) - 1);
06993  variable ivalue : STD_ULOGIC_VECTOR(0 to ne*4 - 1);
06994  variable result : STRING(1 to ne);
06995  variable quad   : STD_ULOGIC_VECTOR(0 to 3);
06996 begin
06997  if value'length < 1 then
06998    return NUS;
06999  else
07000    if value (value'left) = 'Z' then
07001      pad := (others => 'Z');
07002    else
07003      pad := (others => '0');
07004    end if;
07005    ivalue := pad & value;
07006    for i in 0 to ne-1 loop
07007      quad := To_X01Z(ivalue(4*i to 4*i+3));
07008    case quad is
07009      when x"0"  => result(i+1) := '0';
07010      when x"1"  => result(i+1) := '1';
07011      when x"2"  => result(i+1) := '2';
07012      when x"3"  => result(i+1) := '3';
07013      when x"4"  => result(i+1) := '4';
07014      when x"5"  => result(i+1) := '5';
07015      when x"6"  => result(i+1) := '6';
07016      when x"7"  => result(i+1) := '7';
07017      when x"8"  => result(i+1) := '8';
07018      when x"9"  => result(i+1) := '9';
07019      when x"A"  => result(i+1) := 'A';
07020      when x"B"  => result(i+1) := 'B';
07021      when x"C"  => result(i+1) := 'C';
07022      when x"D"  => result(i+1) := 'D';
07023      when x"E"  => result(i+1) := 'E';
07024      when x"F"  => result(i+1) := 'F';
07025      when "ZZZZ" => result(i+1) := 'Z';
07026      when others => result(i+1) := 'X';
07027    end case;
07028  end loop;
07029  return result;
07030 end if;
07031 end function to_hstring;
07032
07033
07034 -- %% END replicated textio functions
07035
07036 -- purpose: writes fixed point into a line
07037 procedure WRITE (
07038   L      : inout LINE;           -- input line
07039   VALUE  : in  UNRESOLVED_ufixed; -- fixed point input
07040   JUSTIFIED : in  SIDE := right;
07041   FIELD   : in  WIDTH := 0) is
07042   variable s   : STRING(1 to value'length +1) := (others => ' ');
07043   variable sindx : INTEGER;
07044 begin -- function write Example: 0011.1100
07045   sindx := 1;
07046   for i in value'high downto value'low loop
07047     if i = -1 then
07048       s(sindx) := '.';
07049       sindx := sindx + 1;
07050     end if;
07051     s(sindx) := MVL9_to_char(STD_ULOGIC(value(i)));
07052     sindx := sindx + 1;
07053   end loop;
07054   write(l, s, justified, field);
07055 end procedure write;
07056
07057 -- purpose: writes fixed point into a line
07058 procedure write (
07059   L      : inout LINE;           -- input line
07060   VALUE  : in  UNRESOLVED_sfixed; -- fixed point input
07061   JUSTIFIED : in  SIDE := right;
07062   FIELD   : in  WIDTH := 0) is
07063   variable s   : STRING(1 to value'length +1);
07064   variable sindx : INTEGER;
07065 begin -- function write Example: 0011.1100
07066   sindx := 1;
07067   for i in value'high downto value'low loop
07068     if i = -1 then
07069       s(sindx) := '.';
07070       sindx := sindx + 1;
07071     end if;
07072     s(sindx) := MVL9_to_char(STD_ULOGIC(value(i)));
07073     sindx := sindx + 1;
07074   end loop;
07075   write(l, s, justified, field);
07076 end procedure write;

```

```

07077
07078 procedure READ(L      : inout LINE;
07079           VALUE  : out  UNRESOLVED_ufixed) is
07080   -- Possible data: 00000.000000
07081   --          000000000000
07082   variable c      : CHARACTER;
07083   variable readOk : BOOLEAN;
07084   variable i       : INTEGER;           -- index variable
07085   variable mv    : ufixed (VALUE'range);
07086   variable lastu : BOOLEAN := false;    -- last character was an "_"
07087   variable founddot : BOOLEAN := false; -- found a "."
07088 begin -- READ
07089   VALUE := (VALUE'range => 'U');
07090   Skip_whitespace (L);
07091   if VALUE'length > 0 then           -- non Null input string
07092     read (l, c, readOk);
07093     i := value'high;
07094     while i >= VALUE'low loop
07095       if readOk = false then         -- Bail out if there was a bad read
07096         report fixed_pkg'instance_name & "READ(ufixed)"
07097         & "End of string encountered"
07098         severity error;
07099         return;
07100       elsif c = '_' then
07101         if i = value'high then
07102           report fixed_pkg'instance_name & "READ(ufixed) "
07103           & "String begins with an ""_"" severity error";
07104           return;
07105       elsif lastu then
07106         report fixed_pkg'instance_name & "READ(ufixed) "
07107         & "Two underscores detected in input string ""__"""
07108         severity error;
07109         return;
07110       else
07111         lastu := true;
07112       end if;
07113     elsif c = '.' then             -- binary point
07114       if founddot then
07115         report fixed_pkg'instance_name & "READ(ufixed) "
07116         & "Two binary points found in input string ""__"""
07117         severity error;
07118       return;
07119     elsif i /= -1 then           -- Separator in the wrong spot
07120       report fixed_pkg'instance_name & "READ(ufixed) "
07121         & "Decimal point does not match number format "
07122         severity error;
07123       return;
07124     end if;
07125     founddot := true;
07126     lastu := false;
07127   elsif c = ' ' or c = NBSP or c = HT then -- reading done.
07128     report fixed_pkg'instance_name & "READ(ufixed) "
07129       & "Short read, Space encountered in input string"
07130       severity error;
07131     return;
07132   elsif char_to_MVL9plus(c) = error then
07133     report fixed_pkg'instance_name & "READ(ufixed) "
07134       & "Character '" &
07135         c & "' read, expected STD_ULOGIC literal."
07136       severity error;
07137     return;
07138   else
07139     mv(i) := char_to_MVL9(c);
07140     i := i - 1;
07141     if i < mv'low then
07142       VALUE := mv;
07143       return;
07144     end if;
07145     lastu := false;
07146   end if;
07147   read(L, c, readOk);
07148 end loop;
07149 end if;
07150 end procedure READ;
07151
07152 procedure READ(L      : inout LINE;
07153           VALUE  : out  UNRESOLVED_ufixed;
07154           GOOD   : out  BOOLEAN) is
07155   -- Possible data: 00000.000000
07156   --          000000000000
07157   variable c      : CHARACTER;
07158   variable readOk : BOOLEAN;
07159   variable mv    : ufixed (VALUE'range);
07160   variable i       : INTEGER;           -- index variable
07161   variable lastu : BOOLEAN := false;    -- last character was an "_"
07162 begin -- READ
07163   VALUE := (VALUE'range => 'U');

```

```

07164      Skip_whitespace (L);
07165      if VALUE'length > 0 then
07166          read (l, c, readOk);
07167          i := value'high;
07168          GOOD := false;
07169          while i >= VALUE'low loop
07170              if not readOk then      -- Bail out if there was a bad read
07171                  return;
07172              elsif c = '_' then
07173                  if i = value'high then      -- Begins with an "_"
07174                      return;
07175                  elsif lastu then          -- "_" detected
07176                      return;
07177                  else
07178                      lastu := true;
07179                  end if;
07180              elsif c = '.' then        -- binary point
07181                  if founddot then
07182                      return;
07183                  elsif i /= -1 then       -- Separator in the wrong spot
07184                      return;
07185                  end if;
07186                  founddot := true;
07187                  lastu := false;
07188              elsif (char_to_MVL9plus(c) = error) then    -- Illegal character/short read
07189                  return;
07190              else
07191                  mv(i) := char_to_MVL9(c);
07192                  i := i - 1;
07193                  if i < mv'low then        -- reading done
07194                      GOOD := true;
07195                      VALUE := mv;
07196                      return;
07197                  end if;
07198                  lastu := false;
07199              end if;
07200              read(L, c, readOk);
07201          end loop;
07202      else
07203          GOOD := true;           -- read into a null array
07204      end if;
07205  end procedure READ;
07206
07207 procedure READ(L : inout LINE;
07208                      VALUE : out UNRESOLVED_sfixed) is
07209     variable c : CHARACTER;
07210     variable readOk : BOOLEAN;
07211     variable i : INTEGER;           -- index variable
07212     variable mv : sfixed (VALUE'range);
07213     variable lastu : BOOLEAN := false;   -- last character was an "_"
07214     variable founddot : BOOLEAN := false; -- found a "."
07215 begin -- READ
07216     VALUE := (VALUE'range => 'U');
07217     Skip_whitespace (L);
07218     if VALUE'length > 0 then        -- non Null input string
07219         read (l, c, readOk);
07220         i := value'high;
07221         while i >= VALUE'low loop
07222             if readOk = false then      -- Bail out if there was a bad read
07223                 report fixed_pkg'instance_name & "READ(sfixed) "
07224                     & "End of string encountered"
07225                     severity error;
07226                 return;
07227             elsif c = '_' then
07228                 if i = value'high then
07229                     report fixed_pkg'instance_name & "READ(sfixed) "
07230                         & "String begins with an ""_"" severity error;
07231                     return;
07232                 elsif lastu then
07233                     report fixed_pkg'instance_name & "READ(sfixed) "
07234                         & "Two underscores detected in input string ""__"""
07235                         severity error;
07236                     return;
07237                 else
07238                     lastu := true;
07239                 end if;
07240             elsif c = '.' then        -- binary point
07241                 if founddot then
07242                     report fixed_pkg'instance_name & "READ(sfixed) "
07243                         & "Two binary points found in input string" severity error;
07244                     return;
07245                 elsif i /= -1 then       -- Separator in the wrong spot
07246                     report fixed_pkg'instance_name & "READ(sfixed) "
07247                         & "Decimal point does not match number format "
07248                         severity error;
07249                     return;
07250                 end if;

```

```

07251      founddot := true;
07252      lastu := false;
07253      elsif c = ' ' or c = NBSP or c = HT then -- reading done.
07254          report fixed_pkg'instance_name & "READ(sfixed) "
07255              & "Short read, Space encountered in input string"
07256              severity error;
07257          return;
07258      elsif char_to_MVL9plus(c) = error then
07259          report fixed_pkg'instance_name & "READ(sfixed) "
07260              & "Character '" &
07261                  c & "' read, expected STD_ULOGIC literal."
07262              severity error;
07263          return;
07264      else
07265          mv(i) := char_to_MVL9(c);
07266          i := i - 1;
07267          if i < mv'low then
07268              VALUE := mv;
07269              return;
07270          end if;
07271          lastu := false;
07272      end if;
07273      read(L, c, readOk);
07274  end loop;
07275 end if;
07276 end procedure READ;
07277
07278 procedure READ(L : inout LINE;
07279                 VALUE : out UNRESOLVED_sfixed;
07280                 GOOD : out BOOLEAN) is
07281     variable value_ufixed : UNRESOLVED_ufixed (VALUE'range);
07282 begin -- READ
07283     READ (L => L, VALUE => value_ufixed, GOOD => GOOD);
07284     VALUE := UNRESOLVED_sfixed (value_ufixed);
07285 end procedure READ;
07286
07287 -- octal read and write
07288 procedure owrite (
07289     L       : inout LINE;                      -- input line
07290     VALUE   : in  UNRESOLVED_ufixed;           -- fixed point input
07291     JUSTIFIED : in  SIDE  := right;
07292     FIELD    : in  WIDTH := 0) is
07293 begin -- Example 03.30
07294     write (L        => L,
07295            VALUE    => to_ostring (VALUE),
07296            JUSTIFIED => JUSTIFIED,
07297            FIELD     => FIELD);
07298 end procedure owrite;
07299
07300 procedure owrite (
07301     L       : inout LINE;                      -- input line
07302     VALUE   : in  UNRESOLVED_sfixed;           -- fixed point input
07303     JUSTIFIED : in  SIDE  := right;
07304     FIELD    : in  WIDTH := 0) is
07305 begin -- Example 03.30
07306     write (L        => L,
07307            VALUE    => to_ostring (VALUE),
07308            JUSTIFIED => JUSTIFIED,
07309            FIELD     => FIELD);
07310 end procedure owrite;
07311
07312 -- purpose: Routines common to the OREAD routines
07313 procedure OREAD_common (
07314     L       : inout LINE;
07315     slv    : out STD_ULOGIC_VECTOR;
07316     igood  : out BOOLEAN;
07317     idex   : out INTEGER;
07318     constant bpoint : in INTEGER;             -- binary point
07319     constant message : in  BOOLEAN;
07320     constant smath  : in  BOOLEAN) is
07321
07322 -- purpose: error message routine
07323 procedure errmes (
07324     constant mess : in STRING) is      -- error message
07325 begin
07326     if message then
07327         if smath then
07328             report fixed_pkg'instance_name
07329                 & "OREAD(sfixed) "
07330                 & mess
07331                 severity error;
07332         else
07333             report fixed_pkg'instance_name
07334                 & "OREAD(ufixed) "
07335                 & mess
07336                 severity error;
07337         end if;

```

```

07338      end if;
07339  end procedure errmes;
07340  variable xgood : BOOLEAN;
07341  variable nybble : STD_ULONGIC_VECTOR (2 downto 0);           -- 3 bits
07342  variable c : CHARACTER;
07343  variable i : INTEGER;
07344  variable lastu : BOOLEAN := false;      -- last character was an "_"
07345  variable founddot : BOOLEAN := false;   -- found a dot.
07346 begin
07347  Skip_whitespace (L);
07348  if slv'length > 0 then
07349    i := slv'high;
07350    read (l, c, xgood);
07351    while i > 0 loop
07352      if xgood = false then
07353        errmes ("Error: end of string encountered");
07354        exit;
07355      elsif c = '_' then
07356        if i = slv'length then
07357          errmes ("Error: String begins with an ""_""");
07358          xgood := false;
07359          exit;
07360        elsif lastu then
07361          errmes ("Error: Two underscores detected in input string ""__""");
07362          xgood := false;
07363          exit;
07364        else
07365          lastu := true;
07366        end if;
07367      elsif (c = '.') then
07368        if (i + 1 /= bpoint) then
07369          errmes ("encountered ""."" at wrong index");
07370          xgood := false;
07371          exit;
07372        elsif i = slv'length then
07373          errmes ("encountered a ""."" at the beginning of the line");
07374          xgood := false;
07375          exit;
07376        elsif founddot then
07377          errmes ("Two ""."" encountered in input string");
07378          xgood := false;
07379          exit;
07380        end if;
07381        founddot := true;
07382        lastu := false;
07383      else
07384        Char2triBits(c, nybble, xgood, message);
07385        if not xgood then
07386          exit;
07387        end if;
07388        slv (i downto i-2) := nybble;
07389        i := i - 3;
07390        lastu := false;
07391      end if;
07392      if i > 0 then
07393        read (L, c, xgood);
07394      end if;
07395    end loop;
07396    index := i;
07397    igood := xgood;
07398  else
07399    igood := true;           -- read into a null array
07400    index := -1;
07401  end if;
07402 end procedure OREAD_common;
07403
07404 -- Note that for Octal and Hex read, you can not start with a ".",
07405 -- the read is for numbers formatted "A.BC". These routines go to
07406 -- the nearest bounds, so "F.E" will fit into an sfixed (2 downto -3).
07407 procedure OREAD (L : inout LINE;
07408                      VALUE : out UNRESOLVED_ufixed) is
07409  constant hbv : INTEGER := ((maximum(3, (VALUE'high+1))+2)/3)*3)-1;
07410  constant lbv : INTEGER := ((mine(0, VALUE'low)-2)/3)*3;
07411  variable slv : STD_ULONGIC_VECTOR (hbv-lbv downto 0); -- high bits
07412  variable valuex : UNRESOLVED_ufixed (hbv downto lbv);
07413  variable igood : BOOLEAN;
07414  variable i : INTEGER;
07415 begin
07416  VALUE := (VALUE'range => 'U');
07417  OREAD_common ( L => L,
07418                  slv => slv,
07419                  igood => igood,
07420                  index => i,
07421                  bpoint => -lbv,
07422                  message => true,
07423                  smath => false);
07424  if igood then           -- We did not get another error

```

```

07425     if not ((i = -1) and           -- We read everything, and high bits 0
07426         (or_reduce (slv(hbv-lbv downto VALUE'high+1-lbv)) = '0')) then
07427         report fixed_pkg'instance_name
07428             & "OREAD(ufixed): Vector truncated."
07429             severity error;
07430     else
07431         if (or_reduce (slv(VALUE'low-lbv-1 downto 0)) = '1') then
07432             assert NO_WARNING
07433             report fixed_pkg'instance_name
07434                 & "OREAD(ufixed): Vector truncated"
07435                 severity warning;
07436         end if;
07437         valuex := to_ufixed (slv, hbv, lbv);
07438         VALUE := valuex (VALUE'range);
07439     end if;
07440 end if;
07441 end procedure OREAD;
07442
07443 procedure OREAD(L      : inout LINE;
07444                  VALUE : out  UNRESOLVED_ufixed;
07445                  GOOD  : out  BOOLEAN) is
07446     constant hbv    : INTEGER := (((maximum(3, (VALUE'high+1))+2)/3)*3)-1;
07447     constant lbv    : INTEGER := ((mine(0, VALUE'low)-2)/3)*3;
07448     variable slv   : STD_ULOGIC_VECTOR (hbv-lbv downto 0);  -- high bits
07449     variable valuex : UNRESOLVED_ufixed (hbv downto lbv);
07450     variable igood  : BOOLEAN;
07451     variable i       : INTEGER;
07452 begin
07453     VALUE := (VALUE'range => 'U');
07454     OREAD_common ( L => L,
07455                     slv => slv,
07456                     igood => igood,
07457                     idex => i,
07458                     bpoint => -lbv,
07459                     message => false,
07460                     smath => false);
07461     if (igood and           -- We did not get another error
07462         (i = -1) and           -- We read everything, and high bits 0
07463         (or_reduce (slv(hbv-lbv downto VALUE'high+1-lbv)) = '0')) then
07464         valuex := to_ufixed (slv, hbv, lbv);
07465         VALUE := valuex (VALUE'range);
07466         good := true;
07467     else
07468         good := false;
07469     end if;
07470 end procedure OREAD;
07471
07472 procedure OREAD(L      : inout LINE;
07473                  VALUE : out  UNRESOLVED_sfixed) is
07474     constant hbv    : INTEGER := (((maximum(3, (VALUE'high+1))+2)/3)*3)-1;
07475     constant lbv    : INTEGER := ((mine(0, VALUE'low)-2)/3)*3;
07476     variable slv   : STD_ULOGIC_VECTOR (hbv-lbv downto 0);  -- high bits
07477     variable valuex : UNRESOLVED_sfixed (hbv downto lbv);
07478     variable igood  : BOOLEAN;
07479     variable i       : INTEGER;
07480 begin
07481     VALUE := (VALUE'range => 'U');
07482     OREAD_common ( L => L,
07483                     slv => slv,
07484                     igood => igood,
07485                     idex => i,
07486                     bpoint => -lbv,
07487                     message => true,
07488                     smath => true);
07489     if igood then           -- We did not get another error
07490         if not ((i = -1) and           -- We read everything
07491             ((slv(VALUE'high-lbv) = '0' and           -- sign bits = extra bits
07492               or_reduce (slv(hbv-lbv downto VALUE'high+1-lbv)) = '0') or
07493               (slv(VALUE'high-lbv) = '1' and
07494                 and_reduce (slv(hbv-lbv downto VALUE'high+1-lbv)) = '1'))) then
07495             report fixed_pkg'instance_name
07496                 & "OREAD(sfixed): Vector truncated."
07497                 severity error;
07498         else
07499             if (or_reduce (slv(VALUE'low-lbv-1 downto 0)) = '1') then
07500                 assert NO_WARNING
07501                 report fixed_pkg'instance_name
07502                     & "OREAD(sfixed): Vector truncated"
07503                     severity warning;
07504             end if;
07505             valuex := to_sfixed (slv, hbv, lbv);
07506             VALUE := valuex (VALUE'range);
07507         end if;
07508     end if;
07509 end procedure OREAD;
07510
07511 procedure OREAD(L      : inout LINE;

```

```

07512           VALUE : out  UNRESOLVED_sfixed;
07513           GOOD  : out  BOOLEAN) is
07514   constant hbv  : INTEGER := (((maximum(3, (VALUE'high+1))+2)/3)*3)-1;
07515   constant lbv  : INTEGER := ((mine(0, VALUE'low)-2)/3)*3;
07516   variable slv  : STD_ULOGIC_VECTOR (hbv-lbv downto 0); -- high bits
07517   variable valuem: UNRESOLVED_sfixed (hbv downto lbv);
07518   variable igood : BOOLEAN;
07519   variable i     : INTEGER;
07520 begin
07521   VALUE := (VALUE'range => 'U');
07522   OREAD_common ( L => L,
07523                 slv => slv,
07524                 igood => igood,
07525                 idex => i,
07526                 bpoint => -lbv,
07527                 message => false,
07528                 smath => true);
07529   if (igood                               -- We did not get another error
07530       and (i = -1)                      -- We read everything
07531       and ((slv(VALUE'high-lbv) = '0' and -- sign bits = extra bits
07532             or_reduce (slv(hbv-lbv downto VALUE'high+1-lbv)) = '0') or
07533             (slv(VALUE'high-lbv) = '1' and
07534               and_reduce (slv(hbv-lbv downto VALUE'high+1-lbv)) = '1'))) then
07535     valuem := to_sfixed (slv, hbv, lbv);
07536     VALUE := valuem (VALUE'range);
07537     good := true;
07538   else
07539     good := false;
07540   end if;
07541 end procedure OREAD;
07542
07543 -- hex read and write
07544 procedure hwrite (
07545   L      : inout LINE;                  -- input line
07546   VALUE : in  UNRESOLVED_ufixed; -- fixed point input
07547   JUSTIFIED : in SIDE := right;
07548   FIELD  : in  WIDTH := 0) is
07549 begin -- Example 03.30
07550   write (L      => L,
07551         VALUE  => to_hstring (VALUE),
07552         JUSTIFIED => JUSTIFIED,
07553         FIELD   => FIELD);
07554 end procedure hwrite;
07555
07556 -- purpose: writes fixed point into a line
07557 procedure hwrite (
07558   L      : inout LINE;                  -- input line
07559   VALUE : in  UNRESOLVED_sfixed; -- fixed point input
07560   JUSTIFIED : in SIDE := right;
07561   FIELD  : in  WIDTH := 0) is
07562 begin -- Example 03.30
07563   write (L      => L,
07564         VALUE  => to_hstring (VALUE),
07565         JUSTIFIED => JUSTIFIED,
07566         FIELD   => FIELD);
07567 end procedure hwrite;
07568
07569 -- purpose: Routines common to the OREAD routines
07570 procedure HREAD_common (
07571   L      : inout LINE;
07572   slv   : out  STD_ULOGIC_VECTOR;
07573   igood : out  BOOLEAN;
07574   idex  : out  INTEGER;
07575   constant bpoint : in INTEGER;        -- binary point
07576   constant message : in  BOOLEAN;
07577   constant smath  : in  BOOLEAN) is
07578
07579 -- purpose: error message routine
07580 procedure errmes (
07581   constant mess : in STRING) is      -- error message
07582 begin
07583   if message then
07584     if smath then
07585       report fixed_pkg'instance_name
07586         & "HREAD(sfixed) "
07587         & mess
07588         severity error;
07589     else
07590       report fixed_pkg'instance_name
07591         & "HREAD(ufixed) "
07592         & mess
07593         severity error;
07594     end if;
07595   end if;
07596 end procedure errmes;
07597 variable xgood : BOOLEAN;
07598 variable nybble : STD_ULOGIC_VECTOR (3 downto 0);      -- 4 bits

```

```

07599     variable c : CHARACTER;
07600     variable i : INTEGER;
07601     variable lastu : BOOLEAN := false;      -- last character was an "_"
07602     variable founddot : BOOLEAN := false;   -- found a dot.
07603 begin
07604     Skip_whitespace (L);
07605     if slv'length > 0 then
07606         i := slv'high;
07607         read (l, c, xgood);
07608         while i > 0 loop
07609             if xgood = false then
07610                 errmes ("Error: end of string encountered");
07611                 exit;
07612             elsif c = '_' then
07613                 if i = slv'length then
07614                     errmes ("Error: String begins with an \"_\"");
07615                     xgood := false;
07616                     exit;
07617                 elsif lastu then
07618                     errmes ("Error: Two underscores detected in input string \"__\"");
07619                     xgood := false;
07620                     exit;
07621                 else
07622                     lastu := true;
07623                 end if;
07624             elsif (c = '.') then
07625                 if (i + 1 /= bpoint) then
07626                     errmes ("encountered \".\" at wrong index");
07627                     xgood := false;
07628                     exit;
07629                 elsif i = slv'length then
07630                     errmes ("encountered a \".\" at the beginning of the line");
07631                     xgood := false;
07632                     exit;
07633                 elsif founddot then
07634                     errmes ("Two \".\" encountered in input string");
07635                     xgood := false;
07636                     exit;
07637                 end if;
07638                 founddot := true;
07639                 lastu := false;
07640             else
07641                 Char2QuadBits(c, nybble, xgood, message);
07642                 if not xgood then
07643                     exit;
07644                 end if;
07645                 slv (i downto i-3) := nybble;
07646                 i := i - 4;
07647                 lastu := false;
07648             end if;
07649             if i > 0 then
07650                 read (L, c, xgood);
07651             end if;
07652         end loop;
07653         index := i;
07654         igood := xgood;
07655     else
07656         index := -1;
07657         igood := true;           -- read null string
07658     end if;
07659 end procedure HREAD_common;
07660
07661 procedure HREAD(L      : inout LINE;
07662                      VALUE : out  UNRESOLVED_ufixed) is
07663     constant hbv    : INTEGER := (((maximum(4, (VALUE'high+1))+3)/4)*4)-1;
07664     constant lbv    : INTEGER := ((mine(0, VALUE'low)-3)/4)*4;
07665     variable slv    : STD_ULOGIC_VECTOR (hbv-lbv downto 0);  -- high bits
07666     variable valuem: UNRESOLVED_ufixed (hbv downto lbv);
07667     variable igood  : BOOLEAN;
07668     variable i       : INTEGER;
07669 begin
07670     VALUE := (VALUE'range => 'U');
07671     HREAD_common ( L => L,
07672                     slv => slv,
07673                     igood => igood,
07674                     index => i,
07675                     bpoint => -lbv,
07676                     message => false,
07677                     smath => false);
07678     if igood then
07679         if not ((i = -1) and          -- We read everything, and high bits 0
07680                  (or_reduce (slv(hbv-lbv downto VALUE'high+1-lbv)) = '0')) then
07681             report fixed_pkg'instance_name
07682               & "HREAD(ufixed): Vector truncated."
07683               severity error;
07684         else
07685             if (or_reduce (slv(VALUE'low-lbv-1 downto 0)) = '1') then

```

```

07686      assert NO_WARNING
07687          report fixed_pkg'instance_name
07688              & "HREAD(ufixed): Vector truncated"
07689              severity warning;
07690      end if;
07691      valuem := to_ufixed (slv, hbv, lbv);
07692      VALUE := valuem (VALUE'range);
07693  end if;
07694 end if;
07695 end procedure HREAD;
07696
07697 procedure HREAD(L    : inout LINE;
07698                 VALUE : out UNRESOLVED_ufixed;
07699                 GOOD  : out BOOLEAN) is
07700     constant hbv   : INTEGER := (((maximum(4, (VALUE'high+1))+3)/4)*4)-1;
07701     constant lbv   : INTEGER := ((mine(0, VALUE'low)-3)/4)*4;
07702     variable slv    : STD_ULOGIC_VECTOR (hbv-lbv downto 0); -- high bits
07703     variable valuem : UNRESOLVED_ufixed (hbv downto lbv);
07704     variable igood  : BOOLEAN;
07705     variable i       : INTEGER;
07706 begin
07707     VALUE := (VALUE'range => 'U');
07708     HREAD_common ( L => L,
07709                     slv => slv,
07710                     igood => igood,
07711                     idex => i,
07712                     bpoint => -lbv,
07713                     message => false,
07714                     smath => false);
07715     if (igood and           -- We did not get another error
07716         (i = -1) and        -- We read everything, and high bits 0
07717         (or_reduce (slv(hbv-lbv downto VALUE'high+1-lbv)) = '0')) then
07718         valuem := to_ufixed (slv, hbv, lbv);
07719         VALUE := valuem (VALUE'range);
07720         good := true;
07721     else
07722         good := false;
07723     end if;
07724 end procedure HREAD;
07725
07726 procedure HREAD(L    : inout LINE;
07727                 VALUE : out UNRESOLVED_sfixed) is
07728     constant hbv   : INTEGER := (((maximum(4, (VALUE'high+1))+3)/4)*4)-1;
07729     constant lbv   : INTEGER := ((mine(0, VALUE'low)-3)/4)*4;
07730     variable slv    : STD_ULOGIC_VECTOR (hbv-lbv downto 0); -- high bits
07731     variable valuem : UNRESOLVED_sfixed (hbv downto lbv);
07732     variable igood  : BOOLEAN;
07733     variable i       : INTEGER;
07734 begin
07735     VALUE := (VALUE'range => 'U');
07736     HREAD_common ( L => L,
07737                     slv => slv,
07738                     igood => igood,
07739                     idex => i,
07740                     bpoint => -lbv,
07741                     message => true,
07742                     smath => true);
07743     if igood then           -- We did not get another error
07744         if not ((i = -1)        -- We read everything
07745             and ((slv(VALUE'high-lbv) = '0' and -- sign bits = extra bits
07746                  or_reduce (slv(hbv-lbv downto VALUE'high+1-lbv)) = '0') or
07747                  (slv(VALUE'high-lbv) = '1' and
07748                  and_reduce (slv(hbv-lbv downto VALUE'high+1-lbv)) = '1'))) then
07749             report fixed_pkg'instance_name
07750                 & "HREAD(sfixed): Vector truncated."
07751                 severity error;
07752         else
07753             if (or_reduce (slv(VALUE'low-lbv-1 downto 0)) = '1') then
07754                 assert NO_WARNING
07755                     report fixed_pkg'instance_name
07756                         & "HREAD(sfixed): Vector truncated"
07757                         severity warning;
07758             end if;
07759             valuem := to_sfixed (slv, hbv, lbv);
07760             VALUE := valuem (VALUE'range);
07761         end if;
07762     end if;
07763 end procedure HREAD;
07764
07765 procedure HREAD(L    : inout LINE;
07766                 VALUE : out UNRESOLVED_sfixed;
07767                 GOOD  : out BOOLEAN) is
07768     constant hbv   : INTEGER := (((maximum(4, (VALUE'high+1))+3)/4)*4)-1;
07769     constant lbv   : INTEGER := ((mine(0, VALUE'low)-3)/4)*4;
07770     variable slv    : STD_ULOGIC_VECTOR (hbv-lbv downto 0); -- high bits
07771     variable valuem : UNRESOLVED_sfixed (hbv downto lbv);
07772     variable igood  : BOOLEAN;

```

```

07773     variable i      : INTEGER;
07774 begin
07775     VALUE := (VALUE'range => 'U');
07776     HREAD_common ( L => L,
07777                 slv => slv,
07778                 igood => igood,
07779                 index => i,
07780                 bpoint => -lbv,
07781                 message => false,
07782                 smath => true);
07783     if (igood and           -- We did not get another error
07784         (i = -1) and        -- We read everything
07785         ((slv(VALUE'high-lbv) = '0' and -- sign bits = extra bits
07786           or_reduce (slv(hbv-lbv downto VALUE'high+1-lbv)) = '0') or
07787           (slv(VALUE'high-lbv) = '1' and
07788             and_reduce (slv(hbv-lbv downto VALUE'high+1-lbv)) = '1'))) then
07789     valuex := to_sfixed (slv, hbv, lbv);
07790     VALUE := valuex (VALUE'range);
07791     good := true;
07792   else
07793     good := false;
07794   end if;
07795 end procedure HREAD;
07796
07797 function to_string (value : UNRESOLVED_ufixed) return STRING is
07798   variable s    : STRING(1 to value'length + 1) := (others => ' ');
07799   variable subval : UNRESOLVED_ufixed (value'high downto -1);
07800   variable sindx : INTEGER;
07801 begin
07802   if value'length < 1 then
07803     return NUS;
07804   else
07805     if value'high < 0 then
07806       if value(value'high) = 'Z' then
07807         return to_string (resize (sfixed(value), 0, value'low));
07808       else
07809         return to_string (resize (value, 0, value'low));
07810       end if;
07811     elsif value'low >= 0 then
07812       if Is_X (value(value'low)) then
07813         subval := (others => value(value'low));
07814         subval (value'range) := value;
07815         return to_string(subval);
07816       else
07817         return to_string (resize (value, value'high, -1));
07818       end if;
07819     else
07820       sindx := 1;
07821       for i in value'high downto value'low loop
07822         if i = -1 then
07823           s(sindx) := '.';
07824           sindx := sindx + 1;
07825         end if;
07826         s(sindx) := MVL9_to_char(STD_ULONGIC(value(i)));
07827         sindx := sindx + 1;
07828       end loop;
07829       return s;
07830     end if;
07831   end if;
07832 end function to_string;
07833
07834 function to_string (value : UNRESOLVED_sfixed) return STRING is
07835   variable s    : STRING(1 to value'length + 1) := (others => ' ');
07836   variable subval : UNRESOLVED_sfixed (value'high downto -1);
07837   variable sindx : INTEGER;
07838 begin
07839   if value'length < 1 then
07840     return NUS;
07841   else
07842     if value'high < 0 then
07843       return to_string (resize (value, 0, value'low));
07844     elsif value'low >= 0 then
07845       if Is_X (value(value'low)) then
07846         subval := (others => value(value'low));
07847         subval (value'range) := value;
07848         return to_string(subval);
07849       else
07850         return to_string (resize (value, value'high, -1));
07851       end if;
07852     else
07853       sindx := 1;
07854       for i in value'high downto value'low loop
07855         if i = -1 then
07856           s(sindx) := '.';
07857           sindx := sindx + 1;
07858         end if;
07859         s(sindx) := MVL9_to_char(STD_ULONGIC(value(i)));

```

```

07860      sindx    := sindx + 1;
07861      end loop;
07862      return s;
07863   end if;
07864 end if;
07865 end function to_string;
07866
07867 function to_ostring (value : UNRESOLVED_ufixed) return STRING is
07868   constant lne : INTEGER := (-VALUE'low+2)/3;
07869   variable subval : UNRESOLVED_ufixed (value'high downto -3);
07870   variable lpad : STD_ULOGIC_VECTOR (0 to (lne*3 + VALUE'low) -1);
07871   variable slv : STD_ULOGIC_VECTOR (value'length-1 downto 0);
07872 begin
07873   if value'length < 1 then
07874     return NUS;
07875   else
07876     if value'high < 0 then
07877       if value(value'high) = 'Z' then
07878         return to_ostring (resize (sfixed(value), 2, value'low));
07879       else
07880         return to_ostring (resize (value, 2, value'low));
07881       end if;
07882     elsif value'low >= 0 then
07883       if Is_X (value(value'low)) then
07884         subval := (others => value(value'low));
07885         subval (value'range) := value;
07886         return to_ostring(subval);
07887       else
07888         return to_ostring (resize (value, value'high, -3));
07889       end if;
07890     else
07891       slv := to_sulv (value);
07892       if Is_X (value (value'low)) then
07893         lpad := (others => value (value'low));
07894       else
07895         lpad := (others => '0');
07896       end if;
07897       return to_ostring(slv(slv'high downto slv'high-VALUE'high)
07898         & "."
07899         & to_ostring(slv(slv'high-VALUE'high-1 downto 0) & lpad);
07900       end if;
07901     end if;
07902   end function to_ostring;
07903
07904 function to_hstring (value : UNRESOLVED_ufixed) return STRING is
07905   constant lne : INTEGER := (-VALUE'low+3)/4;
07906   variable subval : UNRESOLVED_ufixed (value'high downto -4);
07907   variable lpad : STD_ULOGIC_VECTOR (0 to (lne*4 + VALUE'low) -1);
07908   variable slv : STD_ULOGIC_VECTOR (value'length-1 downto 0);
07909 begin
07910   if value'length < 1 then
07911     return NUS;
07912   else
07913     if value'high < 0 then
07914       if value(value'high) = 'Z' then
07915         return to_hstring (resize (sfixed(value), 3, value'low));
07916       else
07917         return to_hstring (resize (value, 3, value'low));
07918       end if;
07919     elsif value'low >= 0 then
07920       if Is_X (value(value'low)) then
07921         subval := (others => value(value'low));
07922         subval (value'range) := value;
07923         return to_hstring(subval);
07924       else
07925         return to_hstring (resize (value, value'high, -4));
07926       end if;
07927     else
07928       slv := to_sulv (value);
07929       if Is_X (value (value'low)) then
07930         lpad := (others => value (value'low));
07931       else
07932         lpad := (others => '0');
07933       end if;
07934       return to_hstring(slv(slv'high downto slv'high-VALUE'high)
07935         & "."
07936         & to_hstring(slv(slv'high-VALUE'high-1 downto 0)&lpad);
07937       end if;
07938     end if;
07939   end function to_hstring;
07940
07941 function to_ostring (value : UNRESOLVED_sfixed) return STRING is
07942   constant ne : INTEGER := ((value'high+1)+2)/3;
07943   variable pad : STD_ULOGIC_VECTOR(0 to (ne*3 - (value'high+1)) - 1);
07944   constant lne : INTEGER := (-VALUE'low+2)/3;
07945   variable subval : UNRESOLVED_sfixed (value'high downto -3);
07946   variable lpad : STD_ULOGIC_VECTOR (0 to (lne*3 + VALUE'low) -1);

```

```

07947     variable slv : STD_ULOGIC_VECTOR (VALUE'high - VALUE'low downto 0);
07948 begin
07949     if value'length < 1 then
07950         return NUS;
07951     else
07952         if value'high < 0 then
07953             return to_ostring (resize (value, 2, value'low));
07954         elsif value'low >= 0 then
07955             if Is_X (value(value'low)) then
07956                 subval := (others => value(value'low));
07957                 subval (value'range) := value;
07958                 return to_ostring(subval);
07959             else
07960                 return to_ostring (resize (value, value'high, -3));
07961             end if;
07962         else
07963             pad := (others => value(value'high));
07964             slv := to_solv (value);
07965             if Is_X (value (value'low)) then
07966                 lpad := (others => value(value'low));
07967             else
07968                 lpad := (others => '0');
07969             end if;
07970             return to_ostring(pad & slv(slv'high downto slv'high-VALUE'high))
07971             & ".";
07972             & to_ostring(slv(slv'high-VALUE'high-1 downto 0) & lpad);
07973         end if;
07974     end if;
07975 end function to_ostring;
07976
07977 function to_hstring (value : UNRESOLVED_sfixed) return STRING is
07978     constant ne : INTEGER := ((value'high+1)+3)/4;
07979     variable pad : STD_ULOGIC_VECTOR(0 to (ne*4 - (value'high+1)) - 1);
07980     constant lne : INTEGER := (-VALUE'low+3)/4;
07981     variable subval : UNRESOLVED_sfixed (value'high downto -4);
07982     variable lpad : STD_ULOGIC_VECTOR (0 to (lne*4 + VALUE'low) -1);
07983     variable slv : STD_ULOGIC_VECTOR (value'length-1 downto 0);
07984 begin
07985     if value'length < 1 then
07986         return NUS;
07987     else
07988         if value'high < 0 then
07989             return to_hstring (resize (value, 3, value'low));
07990         elsif value'low >= 0 then
07991             if Is_X (value(value'low)) then
07992                 subval := (others => value(value'low));
07993                 subval (value'range) := value;
07994                 return to_hstring(subval);
07995             else
07996                 return to_hstring (resize (value, value'high, -4));
07997             end if;
07998         else
07999             slv := to_solv (value);
08000             pad := (others => value(value'high));
08001             if Is_X (value (value'low)) then
08002                 lpad := (others => value(value'low));
08003             else
08004                 lpad := (others => '0');
08005             end if;
08006             return to_hstring(pad & slv(slv'high downto slv'high-VALUE'high))
08007             & ".";
08008             & to_hstring(slv(slv'high-VALUE'high-1 downto 0) & lpad);
08009         end if;
08010     end if;
08011 end function to_hstring;
08012
08013 -- From string functions allow you to convert a string into a fixed
08014 -- point number. Example:
08015 -- signal ufl : ufixed (3 downto -3);
08016 -- ufl <= from_string ("0110.100", ufl'high, ufl'low); -- 6.5
08017 -- The "." is optional in this syntax, however it exist and is
08018 -- in the wrong location an error is produced. Overflow will
08019 -- result in saturation.
08020
08021 function from_string (
08022     bstring           : STRING;      -- binary string
08023     constant left_index : INTEGER;
08024     constant right_index : INTEGER)
08025 return UNRESOLVED_ufixed is
08026     variable result : UNRESOLVED_ufixed (left_index downto right_index);
08027     variable L       : LINE;
08028     variable good    : BOOLEAN;
08029 begin
08030     L := new STRING'(bstring);
08031     read (L, result, good);
08032     deallocate (L);
08033     assert (good)

```

```

08034      report fixed_pkg'instance_name
08035          & "from_string: Bad string "& bstring severity error;
08036      return result;
08037  end function from_string;
08038
08039  -- Octal and hex conversions work as follows:
08040  -- ufl <= from_hstring ("6.8", 3, -3); -- 6.5 (bottom zeros dropped)
08041  -- ufl <= from_ostring ("06.4", 3, -3); -- 6.5 (top zeros dropped)
08042  function from_ostring (
08043      ostring           : STRING;          -- Octal string
08044      constant left_index : INTEGER;
08045      constant right_index : INTEGER)
08046  return UNRESOLVED_ufixed is
08047      variable result : UNRESOLVED_ufixed (left_index downto right_index);
08048      variable L       : LINE;
08049      variable good    : BOOLEAN;
08050  begin
08051      L := new STRING'(ostring);
08052      oread (L, result, good);
08053      deallocate (L);
08054      assert (good)
08055          report fixed_pkg'instance_name
08056              & "from_ostring: Bad string "& ostring severity error;
08057      return result;
08058  end function from_ostring;
08059
08060  function from_hstring (
08061      hstring           : STRING;          -- hex string
08062      constant left_index : INTEGER;
08063      constant right_index : INTEGER)
08064  return UNRESOLVED_ufixed is
08065      variable result : UNRESOLVED_ufixed (left_index downto right_index);
08066      variable L       : LINE;
08067      variable good    : BOOLEAN;
08068  begin
08069      L := new STRING'(hstring);
08070      hread (L, result, good);
08071      deallocate (L);
08072      assert (good)
08073          report fixed_pkg'instance_name
08074              & "from_hstring: Bad string "& hstring severity error;
08075      return result;
08076  end function from_hstring;
08077
08078  function from_string (
08079      bstring           : STRING;          -- binary string
08080      constant left_index : INTEGER;
08081      constant right_index : INTEGER)
08082  return UNRESOLVED_sfixed is
08083      variable result : UNRESOLVED_sfixed (left_index downto right_index);
08084      variable L       : LINE;
08085      variable good    : BOOLEAN;
08086  begin
08087      L := new STRING'(bstring);
08088      read (L, result, good);
08089      deallocate (L);
08090      assert (good)
08091          report fixed_pkg'instance_name
08092              & "from_string: Bad string "& bstring severity error;
08093      return result;
08094  end function from_string;
08095
08096  function from_ostring (
08097      ostring           : STRING;          -- Octal string
08098      constant left_index : INTEGER;
08099      constant right_index : INTEGER)
08100  return UNRESOLVED_sfixed is
08101      variable result : UNRESOLVED_sfixed (left_index downto right_index);
08102      variable L       : LINE;
08103      variable good    : BOOLEAN;
08104  begin
08105      L := new STRING'(ostring);
08106      oread (L, result, good);
08107      deallocate (L);
08108      assert (good)
08109          report fixed_pkg'instance_name
08110              & "from_ostring: Bad string "& ostring severity error;
08111      return result;
08112  end function from_ostring;
08113
08114  function from_hstring (
08115      hstring           : STRING;          -- hex string
08116      constant left_index : INTEGER;
08117      constant right_index : INTEGER)
08118  return UNRESOLVED_sfixed is
08119      variable result : UNRESOLVED_sfixed (left_index downto right_index);
08120      variable L       : LINE;

```

```

08121     variable good    : BOOLEAN;
08122 begin
08123     L := new STRING'(hstring);
08124     hread (L, result, good);
08125     deallocate (L);
08126     assert (good)
08127       report fixed_pkg'instance_name
08128         & "from_hstring: Bad string "& hstring severity error;
08129     return result;
08130 end function from_hstring;
08131
08132 -- Same as above, "size_res" is used for it's range only.
08133 function from_string (
08134   bstring : STRING;           -- binary string
08135   size_res : UNRESOLVED_ufixed)
08136   return UNRESOLVED_ufixed is
08137 begin
08138   return from_string (bstring, size_res'high, size_res'low);
08139 end function from_string;
08140
08141 function from_ostring (
08142   ostring : STRING;          -- Octal string
08143   size_res : UNRESOLVED_ufixed)
08144   return UNRESOLVED_ufixed is
08145 begin
08146   return from_ostring (ostring, size_res'high, size_res'low);
08147 end function from_ostring;
08148
08149 function from_hstring (
08150   hstring : STRING;          -- hex string
08151   size_res : UNRESOLVED_ufixed)
08152   return UNRESOLVED_ufixed is
08153 begin
08154   return from_hstring(hstring, size_res'high, size_res'low);
08155 end function from_hstring;
08156
08157 function from_string (
08158   bstring : STRING;          -- binary string
08159   size_res : UNRESOLVED_sfixed)
08160   return UNRESOLVED_sfixed is
08161 begin
08162   return from_string (bstring, size_res'high, size_res'low);
08163 end function from_string;
08164
08165 function from_ostring (
08166   ostring : STRING;          -- Octal string
08167   size_res : UNRESOLVED_sfixed)
08168   return UNRESOLVED_sfixed is
08169 begin
08170   return from_ostring (ostring, size_res'high, size_res'low);
08171 end function from_ostring;
08172
08173 function from_hstring (
08174   hstring : STRING;          -- hex string
08175   size_res : UNRESOLVED_sfixed)
08176   return UNRESOLVED_sfixed is
08177 begin
08178   return from_hstring (hstring, size_res'high, size_res'low);
08179 end function from_hstring;
08180
08181 -- purpose: Calculate the string boundaries
08182 procedure calculate_string_boundary (
08183   arg      : in STRING;        -- input string
08184   left_index : out INTEGER;    -- left
08185   right_index : out INTEGER) is -- right
08186   -- examples "10001.111" would return +4, -3
08187   -- "07X.44" would return +2, -2 (then the octal routine would multiply)
08188   -- "A_B..C" would return +1, -1 (then the hex routine would multiply)
08189   alias xarg : STRING (arg'length downto 1) is arg; -- make it downto range
08190   variable l, r : INTEGER;      -- internal indexes
08191   variable founddot : BOOLEAN := false;
08192 begin
08193   if arg'length > 0 then
08194     l := xarg'high - 1;
08195     r := 0;
08196     for i in xarg'range loop
08197       if xarg(i) = '_' then
08198         if r = 0 then
08199           l := l - 1;
08200         else
08201           r := r + 1;
08202         end if;
08203       elsif xarg(i) = ' ' or xarg(i) = NBSP or xarg(i) = HT then
08204         report fixed_pkg'instance_name
08205           & "Found a space in the input STRING " & xarg
08206           severity error;
08207       elsif xarg(i) = '.' then

```

```

08208      if founddot then
08209          report fixed_pkg'instance_name
08210              & "Found two binary points in input string " & xarg
08211              severity error;
08212      else
08213          l := 1 - i;
08214          r := -i + 1;
08215          founddot := true;
08216      end if;
08217  end if;
08218 end loop;
08219 left_index := l;
08220 right_index := r;
08221 else
08222     left_index := 0;
08223     right_index := 0;
08224 end if;
08225 end procedure calculate_string_boundary;
08226
08227 -- Direct conversion functions. Example:
08228 -- signal ufl : ufixed (3 downto -3);
08229 -- ufl <= from_string ("0110.100"); -- 6.5
08230 -- In this case the "." is not optional, and the size of
08231 -- the output must match exactly.
08232 function from_string (
08233     bstring : STRING)                                     -- binary string
08234     return UNRESOLVED_ufixed is
08235     variable left_index, right_index : INTEGER;
08236 begin
08237     calculate_string_boundary (bstring, left_index, right_index);
08238     return from_string (bstring, left_index, right_index);
08239 end function from_string;
08240
08241 -- Direct octal and hex conversion functions. In this case
08242 -- the string lengths must match. Example:
08243 -- signal sfl := sfixed (5 downto -3);
08244 -- sfl <= from_ostring ("71.4") -- -6.5
08245 function from_ostring (
08246     ostring : STRING)                                    -- Octal string
08247     return UNRESOLVED_ufixed is
08248     variable left_index, right_index : INTEGER;
08249 begin
08250     calculate_string_boundary (ostring, left_index, right_index);
08251     return from_ostring (ostring, ((left_index+1)*3)-1, right_index*3);
08252 end function from_ostring;
08253
08254 function from_hstring (
08255     hstring : STRING)                                   -- hex string
08256     return UNRESOLVED_ufixed is
08257     variable left_index, right_index : INTEGER;
08258 begin
08259     calculate_string_boundary (hstring, left_index, right_index);
08260     return from_hstring (hstring, ((left_index+1)*4)-1, right_index*4);
08261 end function from_hstring;
08262
08263 function from_string (
08264     bstring : STRING)                                     -- binary string
08265     return UNRESOLVED_sfixed is
08266     variable left_index, right_index : INTEGER;
08267 begin
08268     calculate_string_boundary (bstring, left_index, right_index);
08269     return from_string (bstring, left_index, right_index);
08270 end function from_string;
08271
08272 function from_ostring (
08273     ostring : STRING)                                    -- Octal string
08274     return UNRESOLVED_sfixed is
08275     variable left_index, right_index : INTEGER;
08276 begin
08277     calculate_string_boundary (ostring, left_index, right_index);
08278     return from_ostring (ostring, ((left_index+1)*3)-1, right_index*3);
08279 end function from_ostring;
08280
08281 function from_hstring (
08282     hstring : STRING)                                   -- hex string
08283     return UNRESOLVED_sfixed is
08284     variable left_index, right_index : INTEGER;
08285 begin
08286     calculate_string_boundary (hstring, left_index, right_index);
08287     return from_hstring (hstring, ((left_index+1)*4)-1, right_index*4);
08288 end function from_hstring;
08289 -- pragma synthesis_on
08290 -- rtl_synthesis on
08291 -- IN VHDL-2006 std_logic_vector is a subtype of std_ulegic_vector, so these
08292 -- extra functions are needed for compatibility.
08293 function to_ufixed (
08294     arg           : STD_LOGIC_VECTOR; -- shifted vector

```

```

08295      constant left_index : INTEGER;
08296      constant right_index : INTEGER)
08297      return UNRESOLVED_ufixed is
08298 begin
08299      return to_ufixed (
08300          arg => to_stdulogicvector (arg),
08301          left_index => left_index,
08302          right_index => right_index);
08303 end function to_ufixed;
08304
08305 function to_ufixed (
08306    arg      : STD_LOGIC_VECTOR;           -- shifted vector
08307    size_res : UNRESOLVED_ufixed)        -- for size only
08308    return UNRESOLVED_ufixed is
08309 begin
08310    return to_ufixed (
08311        arg => to_stdulogicvector (arg),
08312        size_res => size_res);
08313 end function to_ufixed;
08314
08315 function to_sfixed (
08316    arg      : STD_LOGIC_VECTOR;           -- shifted vector
08317    constant left_index : INTEGER;
08318    constant right_index : INTEGER)
08319    return UNRESOLVED_sfixed is
08320 begin
08321    return to_sfixed (
08322        arg => to_stdulogicvector (arg),
08323        left_index => left_index,
08324        right_index => right_index);
08325 end function to_sfixed;
08326
08327 function to_sfixed (
08328    arg      : STD_LOGIC_VECTOR;           -- shifted vector
08329    size_res : UNRESOLVED_sfixed)        -- for size only
08330    return UNRESOLVED_sfixed is
08331 begin
08332    return to_sfixed (
08333        arg => to_stdulogicvector (arg),
08334        size_res => size_res);
08335 end function to_sfixed;
08336
08337 -- unsigned fixed point
08338 function to_UFix (
08339    arg      : STD_LOGIC_VECTOR;
08340    width   : NATURAL;                  -- width of vector
08341    fraction : NATURAL)               -- width of fraction
08342    return UNRESOLVED_ufixed is
08343 begin
08344    return to_UFix (
08345        arg => to_stdulogicvector (arg),
08346        width => width,
08347        fraction => fraction);
08348 end function to_UFix;
08349
08350 -- signed fixed point
08351 function to_SFfix (
08352    arg      : STD_LOGIC_VECTOR;
08353    width   : NATURAL;                  -- width of vector
08354    fraction : NATURAL)               -- width of fraction
08355    return UNRESOLVED_sfixed is
08356 begin
08357    return to_SFfix (
08358        arg => to_stdulogicvector (arg),
08359        width => width,
08360        fraction => fraction);
08361 end function to_SFfix;
08362
08363 end package body fixed_pkg;

```

7.21 integrador.vhd File Reference

Entities

- [integrador](#) entity
- [integrador](#) architecture

7.22 integrador.vhd

```

00001 library IEEE;
00002 use IEEE.STD_LOGIC_UNSIGNED.all;
00003 use IEEE.STD_LOGIC_1164.all;
00004 use ieee.fixed_pkg.all;           --arquivo deve ser adicionado ao projeto
00005
00006 entity integrador is
00007 generic  (constant Nin : integer := 13;  --numero de bits da parte inteira excluindo sinal de entrada
00008           constant Nout : integer := 30  --numero de bits da parte inteira excluindo sinal de saida
00009         );
00010 port(
00011     clk : in std_logic; -- clock
00012     en : in std_logic; -- habilita modulo
00013     reset : in std_logic; --
00014     sinc : out std_logic;
00015     MAX : in std_logic_vector(Nout-1 downto 0);
00016     out_data : out std_logic_vector(Nout-1 downto 0); -- data out 30 bits
00017     int_data : in std_logic_vector(Nin-1 downto 0) -- data in
00018   );
00019
00020 end entity integrador;
00021
00022
00023 architecture integrador of integrador is
00024
00025   signal out_int : std_logic_vector(Nout-1 downto 0);
00026   signal sinc_int : std_logic;
00027
00028 begin
00029
00030   process(clk)
00031   begin
00032     if en = '1' then
00033       if reset = '1' then
00034         out_int <= (others => '0');
00035         sinc_int <= '0';
00036       elsif rising_edge(clk) then
00037         if out_int < MAX then
00038           out_int <= out_int+int_data;
00039           sinc_int <= '0';
00040         if out_int(Nout-2 downto 0) < MAX(Nout-1 downto 1) then
00041           sinc_int <= '0';
00042         else
00043           sinc_int <= '1';
00044         end if;
00045       else
00046         out_int <= (others => '0');
00047         sinc_int <= '1';
00048       end if;
00049     end if;
00050
00051   end process;
00052
00053
00054   else
00055     out_int <= (others => '0');
00056     sinc_int <= (others => '0');
00057   end if;
00058 end process;
00059
00060   out_data <= out_int;
00061   sinc <= sinc_int;
00062 end architecture integrador;
00063
00064

```

7.23 LEDs.vhd File Reference

Entities

- [LEDs entity](#)
- [LEDs architecture](#)

7.24 LEDs.vhd

```

00001
00002
00003
00004 LIBRARY ieee;
00005 USE ieee.std_logic_1164.all;
00006 USE ieee.std_logic_arith.all;
00007 USE ieee.std_logic_unsigned.all;
00008
00009 ENTITY LEDs IS -- Base entity
00010 PORT(
00011     CLOCK_50 : in std_logic;
00012     LED :    out std_logic_vector(7 DOWNTO 0);
00013     SW :    in std_logic_vector(3 DOWNTO 0);
00014     KEY :    in std_logic_vector(1 DOWNTO 0)
00015 );
00016 END LEDs;
00017
00018
00019
00020
00021 -- Simples programa para piscar LED
00022 architecture LEDs of LEDs is
00023     constant CLK_FREQ : integer := 50000000; -- 50 MHz
00024     constant BLINK_FREQ : integer := 1;
00025     constant CNT_MAX : integer := CLK_FREQ/BLINK_FREQ/2-1;
00026     signal cnt : unsigned(24 downto 0);
00027     signal blink : std_logic;
00028 begin
00029     process(CLOCK_50)
00030     begin
00031         if rising_edge(CLOCK_50) then
00032             if cnt=CNT_MAX then
00033                 cnt <= (others => '0');
00034                 blink <= not blink;
00035             else
00036                 cnt <= cnt + 1;
00037             end if;
00038         end if;
00039     end process;
00040
00041     LED(0) <= blink;
00042     LED(1) <= not blink;
00043
00044     LED(2) <= SW(0);
00045     LED(3) <= SW(1);
00046     LED(4) <= SW(2);
00047     LED(5) <= SW(3);
00048
00049     LED(6) <= not KEY(0);
00050     LED(7) <= not KEY(1);
00051
00052 end LEDs;
00053
00054
00055
00056
00057
00058

```

7.25 my_types_pkg.vhd File Reference

Entities

- [my_types_pkg package](#)

7.26 my_types_pkg.vhd

```

00001 library IEEE;
00002 use IEEE.STD_LOGIC_UNSIGNED.all;
00003 use IEEE.STD_LOGIC_1164.all;
00004
00005
00006
00007 package my_types_pkg is

```

```

00008  type WORD_ARRAY is array(31 downto 0) of std_logic_vector(15 downto 0);
00009
00010 type ARRAY_8_24 is array(24 downto 1) of std_logic_vector(7 downto 0);
00011  type ARRAY_9_24 is array(24 downto 1) of std_logic_vector(8 downto 0);
00012 type ARRAY_10_24 is array(24 downto 1) of std_logic_vector(9 downto 0);
00013 type ARRAY_12_24 is array(24 downto 1) of std_logic_vector(11 downto 0);
00014 type ARRAY_24_24 is array(24 downto 1) of std_logic_vector(23 downto 0);
00015 type ARRAY_2_24 is array(24 downto 1) of std_logic_vector(1 downto 0);
00016 type ARRAY_21_24 is array(24 downto 1) of std_logic_vector(20 downto 0);
00017 type ARRAY_16_24 is array(24 downto 1) of std_logic_vector(15 downto 0);
00018  type ARRAY_17_24 is array(24 downto 1) of std_logic_vector(16 downto 0);
00019 type ARRAY_18_24 is array(24 downto 1) of std_logic_vector(17 downto 0);
00020 type ARRAY_20_24 is array(24 downto 1) of std_logic_vector(19 downto 0);
00021
00022 type COMP_ARRAY is array(24 downto 1) of std_logic_vector(21 downto 0);
00023 type ARRAY_8_8 is array(7 downto 0) of std_logic_vector(7 downto 0);
00024 type ARRAY_10_6 is array(6 downto 1) of std_logic_vector(9 downto 0);
00025 type ARRAY_12_6 is array(6 downto 1) of std_logic_vector(11 downto 0);
00026 type ARRAY_9_6 is array(6 downto 1) of std_logic_vector(8 downto 0);
00027 type ARRAY_3_6 is array(6 downto 1) of std_logic_vector(2 downto 0);
00028 type ARRAY_4_6 is array(6 downto 1) of std_logic_vector(3 downto 0);
00029 type ARRAY_8_6 is array(6 downto 1) of std_logic_vector(7 downto 0);
00030 type ARRAY_8_5_0 is array(5 downto 0) of std_logic_vector(7 downto 0);
00031
00032
00033 type ARRAY_16_16 is array(16 downto 1) of std_logic_vector(15 downto 0);
00034 type ARRAY_3_16 is array(16 downto 1) of std_logic_vector(2 downto 0);
00035
00036 type ARRAY_16_6 is array(6 downto 1) of std_logic_vector(15 downto 0);
00037 type ARRAY_16x6_12 is array (12 downto 1) of      ARRAY_16_6;
00038
00039
00040
00041 type OPP_GAMMA is array (27 downto 1) of std_logic_vector(28 downto 0);
00042 type OPP_PULSO is array (27 downto 1) of std_logic_vector(2 downto 0);
00043
00044
00045 end package;

```

7.27 pll.vhd File Reference

Entities

- **pll** entity
- **SYN** architecture

7.28 pll.vhd

```

00001 -- megafunction wizard: %ALTPPLL%
00002 -- GENERATION: STANDARD
00003 -- VERSION: WM1.0
00004 -- MODULE: altpll
00005
00006 -- ****
00007 -- File Name: pll.vhd
00008 -- Megafunction Name(s):
00009 --      altpll
00010 --
00011 -- Simulation Library Files(s):
00012 --      altera_mf
00013 -- ****
00014 -- THIS IS A WIZARD-GENERATED FILE. DO NOT EDIT THIS FILE!
00015
00016 --
00017 -- 14.1.0 Build 186 12/03/2014 SJ Web Edition
00018 -- ****
00019
00020
00021 --Copyright (C) 1991-2014 Altera Corporation. All rights reserved.
00022 --Your use of Altera Corporation's design tools, logic functions
00023 --and other software and tools, and its AMPP partner logic
00024 --functions, and any output files from any of the foregoing
00025 --(including device programming or simulation files), and any
00026 --associated documentation or information are expressly subject
00027 --to the terms and conditions of the Altera Program License
00028 --Subscription Agreement, the Altera Quartus II License Agreement,
00029 --the Altera MegaCore Function License Agreement, or other

```

```

00030 --applicable license agreement, including, without limitation,
00031 --that your use is for the sole purpose of programming logic
00032 --devices manufactured by Altera and sold by Altera or its
00033 --authorized distributors. Please refer to the applicable
00034 --agreement for further details.
00035
00036
00037 LIBRARY ieee;
00038 USE ieee.std_logic_1164.all;
00039
00040 LIBRARY altera_mf;
00041 USE altera_mf.all;
00042
00043 ENTITY pll IS
00044   PORT
00045   (
00046     areset      : IN STD_LOGIC  := '0';
00047     inclk0      : IN STD_LOGIC  := '0';
00048     c0          : OUT STD_LOGIC ;
00049     locked      : OUT STD_LOGIC
00050   );
00051 END pll;
00052
00053
00054 ARCHITECTURE SYN OF pll IS
00055
00056   SIGNAL sub_wire0  : STD_LOGIC ;
00057   SIGNAL sub_wire1  : STD_LOGIC_VECTOR (1 DOWNTO 0);
00058   SIGNAL sub_wire2_bv : BIT_VECTOR (0 DOWNTO 0);
00059   SIGNAL sub_wire2  : STD_LOGIC_VECTOR (0 DOWNTO 0);
00060   SIGNAL sub_wire3  : STD_LOGIC_VECTOR (4 DOWNTO 0);
00061   SIGNAL sub_wire4  : STD_LOGIC ;
00062   SIGNAL sub_wire5  : STD_LOGIC ;
00063
00064
00065
00066   COMPONENT altpll
00067   GENERIC (
00068     bandwidth_type      : STRING;
00069     clk0_divide_by      : NATURAL;
00070     clk0_duty_cycle     : NATURAL;
00071     clk0_multiply_by    : NATURAL;
00072     clk0_phase_shift    : STRING;
00073     compensate_clock    : STRING;
00074     inclk0_input_frequency : NATURAL;
00075     intended_device_family : STRING;
00076     lpm_hint            : STRING;
00077     lpm_type             : STRING;
00078     operation_mode       : STRING;
00079     pll_type             : STRING;
00080     port_activeclock     : STRING;
00081     port_areset           : STRING;
00082     port_clkbad0          : STRING;
00083     port_clkbad1          : STRING;
00084     port_clkloss           : STRING;
00085     port_clkswitch         : STRING;
00086     port_configupdate      : STRING;
00087     port_fbin              : STRING;
00088     port_inclk0             : STRING;
00089     port_inclk1             : STRING;
00090     port_locked              : STRING;
00091     port_pfdena             : STRING;
00092     port_phasecounterselect : STRING;
00093     port_phasedone            : STRING;
00094     port_phasestep             : STRING;
00095     port_phaseupdown          : STRING;
00096     port_pllena              : STRING;
00097     port_scanaclr             : STRING;
00098     port_scanclk              : STRING;
00099     port_scanclkena           : STRING;
00100     port_scandata             : STRING;
00101     port_scandataout          : STRING;
00102     port_scandone              : STRING;
00103     port_scanread              : STRING;
00104     port_scanwrite              : STRING;
00105     port_clk0                 : STRING;
00106     port_clk1                 : STRING;
00107     port_clk2                 : STRING;
00108     port_clk3                 : STRING;
00109     port_clk4                 : STRING;
00110     port_clk5                 : STRING;
00111     port_clkena0               : STRING;
00112     port_clkena1               : STRING;
00113     port_clkena2               : STRING;
00114     port_clkena3               : STRING;
00115     port_clkena4               : STRING;
00116     port_clkena5               : STRING;

```

```

00117      port_extclk0      : STRING;
00118      port_extclk1      : STRING;
00119      port_extclk2      : STRING;
00120      port_extclk3      : STRING;
00121      self_reset_on_loss_lock : STRING;
00122      width_clock       : NATURAL
00123  );
00124  PORT (
00125      areset    : IN STD_LOGIC ;
00126      inclk     : IN STD_LOGIC_VECTOR (1 DOWNTO 0);
00127      clk       : OUT STD_LOGIC_VECTOR (4 DOWNTO 0);
00128      locked    : OUT STD_LOGIC
00129  );
00130  END COMPONENT;
00131
00132 BEGIN
00133     sub_wire2_bv(0 DOWNTO 0) <= "0";
00134     sub_wire2    <= To_stdlogicvector(sub_wire2_bv);
00135     sub_wire0    <= inclk0;
00136     sub_wire1    <= sub_wire2(0 DOWNTO 0) & sub_wire0;
00137     sub_wire4    <= sub_wire3(0);
00138     c0          <= sub_wire4;
00139     locked      <= sub_wire5;
00140
00141  altpll_component : altpll
00142  GENERIC MAP (
00143      bandwidth_type => "AUTO",
00144      clk0_divide_by => 15,
00145      clk0_duty_cycle => 50,
00146      clk0_multiply_by => 16,
00147      clk0_phase_shift => "0",
00148      compensate_clock => "CLK0",
00149      inclk0_input_frequency => 20000,
00150      intended_device_family => "Cyclone IV E" ,
00151      lpm_hint => "CBX_MODULE_PREFIX=pll",
00152      lpm_type => "altpll",
00153      operation_mode => "NORMAL",
00154      pll_type => "AUTO",
00155      port_activeclock => "PORT_UNUSED",
00156      port_areset => "PORT_USED",
00157      port_clkb0d0 => "PORT_UNUSED",
00158      port_clkb0d1 => "PORT_UNUSED",
00159      port_clkloss => "PORT_UNUSED",
00160      port_clkswitch => "PORT_UNUSED",
00161      port_configupdate => "PORT_UNUSED",
00162      port_fbin => "PORT_UNUSED",
00163      port_inclk0 => "PORT_USED",
00164      port_inclk1 => "PORT_UNUSED",
00165      port_locked => "PORT_USED",
00166      port_pfdena => "PORT_UNUSED",
00167      port_phasecounterselect => "PORT_UNUSED",
00168      port_phasedone => "PORT_UNUSED",
00169      port_phasestep => "PORT_UNUSED",
00170      port_phaseupdown => "PORT_UNUSED",
00171      port_pllena => "PORT_UNUSED",
00172      port_scanaclr => "PORT_UNUSED",
00173      port_scanclk => "PORT_UNUSED",
00174      port_scanclnena => "PORT_UNUSED",
00175      port_scandata => "PORT_UNUSED",
00176      port_scandataout => "PORT_UNUSED",
00177      port_scandone => "PORT_UNUSED",
00178      port_scanread => "PORT_UNUSED",
00179      port_scanwrite => "PORT_UNUSED",
00180      port_clk0 => "PORT_USED",
00181      port_clk1 => "PORT_UNUSED",
00182      port_clk2 => "PORT_UNUSED",
00183      port_clk3 => "PORT_UNUSED",
00184      port_clk4 => "PORT_UNUSED",
00185      port_clk5 => "PORT_UNUSED",
00186      port_clkena0 => "PORT_UNUSED",
00187      port_clkena1 => "PORT_UNUSED",
00188      port_clkena2 => "PORT_UNUSED",
00189      port_clkena3 => "PORT_UNUSED",
00190      port_clkena4 => "PORT_UNUSED",
00191      port_clkena5 => "PORT_UNUSED",
00192      port_extclk0 => "PORT_UNUSED",
00193      port_extclk1 => "PORT_UNUSED",
00194      port_extclk2 => "PORT_UNUSED",
00195      port_extclk3 => "PORT_UNUSED",
00196      self_reset_on_loss_lock => "OFF",
00197      width_clock => 5
00198  )
00199  PORT MAP (
00200      areset => areset,
00201      inclk => sub_wire1,
00202      clk => sub_wire3,
00203      locked => sub_wire5

```

```

00204      );
00205
00206
00207
00208 END SYN;
00209
00210 -- =====
00211 -- CNX file retrieval info
00212 -- =====
00213 -- Retrieval info: PRIVATE: ACTIVECLK_CHECK STRING "0"
00214 -- Retrieval info: PRIVATE: BANDWIDTH STRING "1.000"
00215 -- Retrieval info: PRIVATE: BANDWIDTH_FEATURE_ENABLED STRING "1"
00216 -- Retrieval info: PRIVATE: BANDWIDTH_FREQ_UNIT STRING "MHz"
00217 -- Retrieval info: PRIVATE: BANDWIDTH_PRESET STRING "Low"
00218 -- Retrieval info: PRIVATE: BANDWIDTH_USE_AUTO STRING "1"
00219 -- Retrieval info: PRIVATE: BANDWIDTH_USE_PRESET STRING "0"
00220 -- Retrieval info: PRIVATE: CLKBAD_SWITCHOVER_CHECK STRING "0"
00221 -- Retrieval info: PRIVATE: CLKLOSS_CHECK STRING "0"
00222 -- Retrieval info: PRIVATE: CLKSWITCH_CHECK STRING "0"
00223 -- Retrieval info: PRIVATE: CNX_NO_COMPENSATE_RADIO STRING "0"
00224 -- Retrieval info: PRIVATE: CREATE_CLKBAD_CHECK STRING "0"
00225 -- Retrieval info: PRIVATE: CREATE_INCLK1_CHECK STRING "0"
00226 -- Retrieval info: PRIVATE: CUR_DEDICATED_CLK STRING "c0"
00227 -- Retrieval info: PRIVATE: CUR_FBIN_CLK STRING "c0"
00228 -- Retrieval info: PRIVATE: DEVICE_SPEED_GRADE STRING "6"
00229 -- Retrieval info: PRIVATE: DIV_FACTOR0 NUMERIC "15"
00230 -- Retrieval info: PRIVATE: DUTY_CYCLE0 STRING "50.0000000"
00231 -- Retrieval info: PRIVATE: EFF_OUTPUT_FREQ_VALUE0 STRING "53.333332"
00232 -- Retrieval info: PRIVATE: EXPLICIT_SWITCHOVER_COUNTER STRING "0"
00233 -- Retrieval info: PRIVATE: EXT_FEEDBACK_RADIO STRING "0"
00234 -- Retrieval info: PRIVATE: GLOCKED_COUNTER_EDIT_CHANGED STRING "1"
00235 -- Retrieval info: PRIVATE: GLOCKED_FEATURE_ENABLED STRING "0"
00236 -- Retrieval info: PRIVATE: GLOCKED_MODE_CHECK STRING "0"
00237 -- Retrieval info: PRIVATE: GLOCK_COUNTER_EDIT NUMERIC "1048575"
00238 -- Retrieval info: PRIVATE: HAS_MANUAL_SWITCHOVER STRING "1"
00239 -- Retrieval info: PRIVATE: INCLK0_FREQ_EDIT STRING "50.000"
00240 -- Retrieval info: PRIVATE: INCLK0_FREQ_UNIT_COMBO STRING "MHz"
00241 -- Retrieval info: PRIVATE: INCLK1_FREQ_EDIT STRING "100.000"
00242 -- Retrieval info: PRIVATE: INCLK1_FREQ_EDIT_CHANGED STRING "1"
00243 -- Retrieval info: PRIVATE: INCLK1_FREQ_UNIT_CHANGED STRING "1"
00244 -- Retrieval info: PRIVATE: INCLK1_FREQ_UNIT_COMBO STRING "MHz"
00245 -- Retrieval info: PRIVATE: INTENDED_DEVICE_FAMILY STRING "Cyclone IV E"
00246 -- Retrieval info: PRIVATE: INT_FEEDBACK_MODE_RADIO STRING "1"
00247 -- Retrieval info: PRIVATE: LOCKED_OUTPUT_CHECK STRING "1"
00248 -- Retrieval info: PRIVATE: LONG_SCAN_RADIO STRING "1"
00249 -- Retrieval info: PRIVATE: LVDS_MODE_DATA_RATE STRING "Not Available"
00250 -- Retrieval info: PRIVATE: LVDS_MODE_DATA_RATE_DIRTY NUMERIC "0"
00251 -- Retrieval info: PRIVATE: LVDS_PHASE_SHIFT_UNITO STRING "deg"
00252 -- Retrieval info: PRIVATE: MIG_DEVICE_SPEED_GRADE STRING "Any"
00253 -- Retrieval info: PRIVATE: MIRROR_CLK0 STRING "0"
00254 -- Retrieval info: PRIVATE: MULT_FACTOR0 NUMERIC "16"
00255 -- Retrieval info: PRIVATE: NORMAL_MODE_RADIO STRING "1"
00256 -- Retrieval info: PRIVATE: OUTPUT_FREQ0 STRING "53.3333300"
00257 -- Retrieval info: PRIVATE: OUTPUT_FREQ_MODE0 STRING "0"
00258 -- Retrieval info: PRIVATE: OUTPUT_FREQ_UNITO STRING "MHz"
00259 -- Retrieval info: PRIVATE: PHASE_RECONFIG_FEATURE_ENABLED STRING "1"
00260 -- Retrieval info: PRIVATE: PHASE_RECONFIG_INPUTS_CHECK STRING "0"
00261 -- Retrieval info: PRIVATE: PHASE_SHIFT0 STRING "0.0000000"
00262 -- Retrieval info: PRIVATE: PHASE_SHIFT_STEP_ENABLED_CHECK STRING "0"
00263 -- Retrieval info: PRIVATE: PHASE_SHIFT_UNITO STRING "deg"
00264 -- Retrieval info: PRIVATE: PLL_ADVANCED_PARAM_CHECK STRING "0"
00265 -- Retrieval info: PRIVATE: PLL_ARESET_CHECK STRING "1"
00266 -- Retrieval info: PRIVATE: PLL_AUTOPLL_CHECK NUMERIC "1"
00267 -- Retrieval info: PRIVATE: PLL_ENHPLL_CHECK NUMERIC "0"
00268 -- Retrieval info: PRIVATE: PLL_FASTPLL_CHECK NUMERIC "0"
00269 -- Retrieval info: PRIVATE: PLL_FBIMIMIC_CHECK STRING "0"
00270 -- Retrieval info: PRIVATE: PLL_LVDS_PLL_CHECK NUMERIC "0"
00271 -- Retrieval info: PRIVATE: PLL_PFDENA_CHECK STRING "0"
00272 -- Retrieval info: PRIVATE: PLL_TARGET_HARCOPY_CHECK NUMERIC "0"
00273 -- Retrieval info: PRIVATE: PRIMARY_CLK_COMBO STRING "inclk0"
00274 -- Retrieval info: PRIVATE: RECONFIG_FILE STRING "pll.mif"
00275 -- Retrieval info: PRIVATE: SACN_INPUTS_CHECK STRING "0"
00276 -- Retrieval info: PRIVATE: SCAN_FEATURE_ENABLED STRING "1"
00277 -- Retrieval info: PRIVATE: SELF_RESET_LOCK_LOSS STRING "0"
00278 -- Retrieval info: PRIVATE: SHORT_SCAN_RADIO STRING "0"
00279 -- Retrieval info: PRIVATE: SPREAD_FEATURE_ENABLED STRING "0"
00280 -- Retrieval info: PRIVATE: SPREAD_FREQ STRING "50.000"
00281 -- Retrieval info: PRIVATE: SPREAD_FREQ_UNIT STRING "KHz"
00282 -- Retrieval info: PRIVATE: SPREAD_PERCENT STRING "0.500"
00283 -- Retrieval info: PRIVATE: SPREAD_USE STRING "0"
00284 -- Retrieval info: PRIVATE: SRC_SYNCH_COMP_RADIO STRING "0"
00285 -- Retrieval info: PRIVATE: STICKY_CLK0 STRING "1"
00286 -- Retrieval info: PRIVATE: SWITCHOVER_COUNT_EDIT NUMERIC "1"
00287 -- Retrieval info: PRIVATE: SWITCHOVER_FEATURE_ENABLED STRING "1"
00288 -- Retrieval info: PRIVATE: SYNTH_WRAPPER_GEN_POSTFIX STRING "0"
00289 -- Retrieval info: PRIVATE: USE_CLK0 STRING "1"
00290 -- Retrieval info: PRIVATE: USE_CLKENAO STRING "0"

```

```

00291 -- Retrieval info: PRIVATE: USE_MIL_SPEED_GRADE NUMERIC "0"
00292 -- Retrieval info: PRIVATE: ZERO_DELAY_RADIO STRING "0"
00293 -- Retrieval info: LIBRARY: altera_mf altera_mf_components.all
00294 -- Retrieval info: CONSTANT: BANDWIDTH_TYPE STRING "AUTO"
00295 -- Retrieval info: CONSTANT: CLK0_DIVIDE_BY NUMERIC "15"
00296 -- Retrieval info: CONSTANT: CLK0_DUTY_CYCLE NUMERIC "50"
00297 -- Retrieval info: CONSTANT: CLK0_MULTIPLY_BY NUMERIC "16"
00298 -- Retrieval info: CONSTANT: CLK0_PHASE_SHIFT STRING "0"
00299 -- Retrieval info: CONSTANT: COMPENSATE_CLOCK STRING "CLK0"
00300 -- Retrieval info: CONSTANT: INCLK0_INPUT_FREQUENCY NUMERIC "20000"
00301 -- Retrieval info: CONSTANT: INTENDED_DEVICE_FAMILY STRING "Cyclone IV E"
00302 -- Retrieval info: CONSTANT: LPM_TYPE STRING "altpll"
00303 -- Retrieval info: CONSTANT: OPERATION_MODE STRING "NORMAL"
00304 -- Retrieval info: CONSTANT: PLL_TYPE STRING "AUTO"
00305 -- Retrieval info: CONSTANT: PORT_ACTIVECLOCK STRING "PORT_UNUSED"
00306 -- Retrieval info: CONSTANT: PORT_ARESET STRING "PORT_USED"
00307 -- Retrieval info: CONSTANT: PORT_CLKBADO STRING "PORT_UNUSED"
00308 -- Retrieval info: CONSTANT: PORT_CLKBAD1 STRING "PORT_UNUSED"
00309 -- Retrieval info: CONSTANT: PORT_CLKLOSS STRING "PORT_UNUSED"
00310 -- Retrieval info: CONSTANT: PORT_CLKSWITCH STRING "PORT_UNUSED"
00311 -- Retrieval info: CONSTANT: PORT_CONFIGUPDATE STRING "PORT_UNUSED"
00312 -- Retrieval info: CONSTANT: PORT_FBIN STRING "PORT_UNUSED"
00313 -- Retrieval info: CONSTANT: PORT_INCLK0 STRING "PORT_USED"
00314 -- Retrieval info: CONSTANT: PORT_INCLK1 STRING "PORT_UNUSED"
00315 -- Retrieval info: CONSTANT: PORT_LOCKED STRING "PORT_USED"
00316 -- Retrieval info: CONSTANT: PORT_PFDENA STRING "PORT_UNUSED"
00317 -- Retrieval info: CONSTANT: PORT_PHASECOUNTERSELECT STRING "PORT_UNUSED"
00318 -- Retrieval info: CONSTANT: PORT_PASEDONE STRING "PORT_UNUSED"
00319 -- Retrieval info: CONSTANT: PORT_PASESTEP STRING "PORT_UNUSED"
00320 -- Retrieval info: CONSTANT: PORT_PHASEUPDOWN STRING "PORT_UNUSED"
00321 -- Retrieval info: CONSTANT: PORT_PLLENA STRING "PORT_UNUSED"
00322 -- Retrieval info: CONSTANT: PORT_SCANACLR STRING "PORT_UNUSED"
00323 -- Retrieval info: CONSTANT: PORT_SCANCLK STRING "PORT_UNUSED"
00324 -- Retrieval info: CONSTANT: PORT_SCANCLKENA STRING "PORT_UNUSED"
00325 -- Retrieval info: CONSTANT: PORT_SCANDATA STRING "PORT_UNUSED"
00326 -- Retrieval info: CONSTANT: PORT_SCANDATAOUT STRING "PORT_UNUSED"
00327 -- Retrieval info: CONSTANT: PORT_SCANDONE STRING "PORT_UNUSED"
00328 -- Retrieval info: CONSTANT: PORT_SCANREAD STRING "PORT_UNUSED"
00329 -- Retrieval info: CONSTANT: PORT_SCANWRITE STRING "PORT_UNUSED"
00330 -- Retrieval info: CONSTANT: PORT_clk0 STRING "PORT_USED"
00331 -- Retrieval info: CONSTANT: PORT_clk1 STRING "PORT_UNUSED"
00332 -- Retrieval info: CONSTANT: PORT_clk2 STRING "PORT_UNUSED"
00333 -- Retrieval info: CONSTANT: PORT_clk3 STRING "PORT_UNUSED"
00334 -- Retrieval info: CONSTANT: PORT_clk4 STRING "PORT_UNUSED"
00335 -- Retrieval info: CONSTANT: PORT_clk5 STRING "PORT_UNUSED"
00336 -- Retrieval info: CONSTANT: PORT_ckena0 STRING "PORT_UNUSED"
00337 -- Retrieval info: CONSTANT: PORT_ckenal STRING "PORT_UNUSED"
00338 -- Retrieval info: CONSTANT: PORT_ckena2 STRING "PORT_UNUSED"
00339 -- Retrieval info: CONSTANT: PORT_ckena3 STRING "PORT_UNUSED"
00340 -- Retrieval info: CONSTANT: PORT_ckena4 STRING "PORT_UNUSED"
00341 -- Retrieval info: CONSTANT: PORT_ckena5 STRING "PORT_UNUSED"
00342 -- Retrieval info: CONSTANT: PORT_extclk0 STRING "PORT_UNUSED"
00343 -- Retrieval info: CONSTANT: PORT_extclk1 STRING "PORT_UNUSED"
00344 -- Retrieval info: CONSTANT: PORT_extclk2 STRING "PORT_UNUSED"
00345 -- Retrieval info: CONSTANT: PORT_extclk3 STRING "PORT_UNUSED"
00346 -- Retrieval info: CONSTANT: SELF_RESET_ON_LOSS_LOCK STRING "OFF"
00347 -- Retrieval info: CONSTANT: WIDTH_CLOCK NUMERIC "5"
00348 -- Retrieval info: USED_PORT: @clk 0 0 5 0 OUTPUT_CLK_EXT VCC "@clk[4..0]"
00349 -- Retrieval info: USED_PORT: @inclk 0 0 2 0 INPUT_CLK_EXT VCC "@inclk[1..0]"
00350 -- Retrieval info: USED_PORT: areset 0 0 0 0 INPUT GND "areset"
00351 -- Retrieval info: USED_PORT: c0 0 0 0 0 OUTPUT_CLK_EXT VCC "c0"
00352 -- Retrieval info: USED_PORT: inclk0 0 0 0 0 INPUT_CLK_EXT GND "inclk0"
00353 -- Retrieval info: USED_PORT: locked 0 0 0 0 OUTPUT GND "locked"
00354 -- Retrieval info: CONNECT: @areset 0 0 0 0 areset 0 0 0 0
00355 -- Retrieval info: CONNECT: @inclk 0 0 1 1 GND 0 0 0 0
00356 -- Retrieval info: CONNECT: @inclk 0 0 1 0 inclk0 0 0 0 0
00357 -- Retrieval info: CONNECT: c0 0 0 0 0 @clk 0 0 1 0
00358 -- Retrieval info: CONNECT: locked 0 0 0 0 @locked 0 0 0 0
00359 -- Retrieval info: GEN_FILE: TYPE_NORMAL pll.vhd TRUE
00360 -- Retrieval info: GEN_FILE: TYPE_NORMAL pll.ppf TRUE
00361 -- Retrieval info: GEN_FILE: TYPE_NORMAL pll.inc FALSE
00362 -- Retrieval info: GEN_FILE: TYPE_NORMAL pll.cmp TRUE
00363 -- Retrieval info: GEN_FILE: TYPE_NORMAL pll.bsf FALSE
00364 -- Retrieval info: GEN_FILE: TYPE_NORMAL pll_inst.vhd FALSE
00365 -- Retrieval info: LIB_FILE: altera_mf
00366 -- Retrieval info: CBX_MODULE_PREFIX: ON

```

7.29 portadora_tringular.vhd File Reference

Entities

- [portadora_tringular](#) entity

- [portadora_tringular](#) architecture

7.30 portadora_tringular.vhd

```

00001 library IEEE;
00002 use IEEE.STD_LOGIC_UNSIGNED.all;
00003 use IEEE.STD_LOGIC_1164.all;           -- arquivo deve ser adicionado ao projeto
00005
00006 entity portadora_tringular is
00007 generic  (constant N : integer := 16  -- numero de bitsa
00008           );
00009   port(
00010     clk : in std_logic; -- clock
00011     en : in std_logic; -- habilita modulo
00012     reset : in std_logic; --
00013     count_ini : in std_logic_vector(N-1 downto 0); -- valor inicial da contagem
00014     dir_ini : in std_logic; -- direcao inicial da contagem 0: crescente ou 1: decrescente
00015     MAX :  in std_logic_vector(N-1 downto 0); -- valor de contagem maximo
00016     dir : out std_logic; -- direcao atual 0: crescente ou 1: decrescente
00017     c : out std_logic_vector(N-1 downto 0) -- data out
00018   );
00019
00020 end entity portadora_tringular;
00021
00022
00023 architecture portadora_tringular of portadora_tringular is
00024
00025   signal c_int : std_logic_vector(N-1 downto 0);
00026   signal dir_int : std_logic;
00027
00028
00029 begin
00030
00031   process(clk)
00032     begin
00033       if en = '1' then
00034         if reset = '1' then
00035           c_int <= count_ini;
00036           dir_int <= dir_ini;
00037         elsif rising_edge(clk) then
00038           if c_int > (MAX-1) then
00039             dir_int <= '1';
00040             c_int <= c_int - 1;
00041           elsif c_int < 1 then
00042             dir_int <= '0';
00043             c_int <= c_int + 1;
00044           elsif dir_int = '0' then
00045             c_int <= c_int + 1;
00046           else
00047             c_int <= c_int - 1;
00048           end if;
00049         end if;
00050       else
00051         c_int <= count_ini;
00052         dir_int <= dir_ini;
00053       end if;
00054     end process;
00055
00056   c <= c_int;
00057   dir <= dir_int;
00058 end architecture portadora_tringular;

```

7.31 sinewave.vhd File Reference

Entities

- [sinewave](#) entity
- [Behavioral](#) architecture

7.32 sinewave.vhd

00001

```

00002 library IEEE;
00003 use IEEE.STD_LOGIC_1164.ALL;
00004 use IEEE.NUMERIC_STD.ALL; --try to use this library as much as possible.
00005
00006 entity sinewave is
00007 port (clk :in std_logic;
00008         dataout : out integer range -128 to 127
00009       );
00010 end sinewave;
00011
00012 architecture Behavioral of sinewave is
00013 signal i : integer range 0 to 30:=0;
00014 type memory_type is array (0 to 29) of integer range -128 to 127;
00015 --ROM for storing the sine values generated by MATLAB.
00016 signal sine : memory_type :=(0,16,31,45,58,67,74,77,77,74,67,58,45,31,16,0,
00017 -16,-31,-45,-58,-67,-74,-77,-74,-67,-58,-45,-31,-16);
00018
00019 begin
00020
00021 process(clk)
00022 begin
00023   --to check the rising edge of the clock signal
00024 if(rising_edge(clk)) then
00025   dataout <= sine(i);
00026   i <= i+1;
00027 if(i = 29) then
00028   i <= 0;
00029 end if;
00030 end if;
00031 end process;
00032
00033 end Behavioral;
00034

```

7.33 tabela_sin.m File Reference

7.33.1 Function Documentation

7.33.1.1 `fclose(fid)`

7.33.1.2 `fprintf(fid , 'with id select\n')`

7.33.1.3 `fprintf(fid , 'sin <=')`

7.33.1.4 `fprintf(fid , 'std_logic_vector(to_unsigned(%u, n_bits_c)) when "%s" , \n' , roundamp*sin((id-1)/(cmax)*pi*2)+med, id. bin)`

7.33.1.5 `end fprintf(fid , 'std_logic_vector(to_unsigned(0, n_bits_c)) when others;')`

7.33.1.6 `fprintf(fid , 'std_logic_vector(to_unsigned(%u, n_bits_c)) when "%s" , \n' , senok, id. bin)`

7.33.1.7 `fprintf(fid , 'to_sfixed(%1.6f, l, -F) when "%s" , \n' , yk, id. bin)`

7.33.1.8 `end fprintf(fid , 'to_sfixed(0, l, -F) when others;')`

7.33.1.9 `log2(735801)`

7.33.1.10 `log2(1335)`

Initial value:

= 10.3826

id=0:1:2^Nbits-1

7.33.1.11 log2(cmax)

7.33.1.12 figure plot(seno)

7.33.1.13 plotar(id)

7.33.2 Variable Documentation

7.33.2.1 a = fi(4832, 0,13, 0)

Definition at line 54 of file [tabela_sin.m](#).

7.33.2.2 amp = med*duty

Definition at line 111 of file [tabela_sin.m](#).

7.33.2.3 cmax = 1335

Definition at line 109 of file [tabela_sin.m](#).

7.33.2.4 comp =fi((yQ15*fimax/2+fimax/2),0,16,0)

Definition at line 198 of file [tabela_sin.m](#).

7.33.2.5 Duty cycle duty =0.858794

Definition at line 108 of file [tabela_sin.m](#).

7.33.2.6 duty60Hz =0.858794

Definition at line 68 of file [tabela_sin.m](#).

7.33.2.7 Limites para operation Tabela sin para FPGA fid = fopen('tabela_sin.txt', 'w')

Definition at line 102 of file [tabela_sin.m](#).

7.33.2.8 fimax =fi(1335,0,16,0)

Definition at line 195 of file [tabela_sin.m](#).

7.33.2.9 numero de bits da parte inteira excluindo sinal de saida fin =fint

Definition at line 43 of file [tabela_sin.m](#).

7.33.2.10 fint =fpll/8

Definition at line 37 of file [tabela_sin.m](#).

7.33.2.11 tabela em ponto fixo

Definition at line 186 of file [tabela_sin.m](#).

7.33.2.12 fout = (int_data * fin) / MAX

Definition at line 63 of file [tabela_sin.m](#).

7.33.2.13 foutMin = (INCmin * fin) / MAX

Definition at line 72 of file [tabela_sin.m](#).

7.33.2.14 for id

Initial value:

```
=1:1:2^Nbits  
idd = fi(id-1,0,Nbits,0)
```

Definition at line 118 of file [tabela_sin.m](#).

7.33.2.15 gera todos os indices idfi = fi(id,0,Nbits,0)

Definition at line 150 of file [tabela_sin.m](#).

7.33.2.16 INC60Hz = 4832

Definition at line 61 of file [tabela_sin.m](#).

7.33.2.17 INCdata = fout * MAX / fin

Definition at line 66 of file [tabela_sin.m](#).

7.33.2.18 INCmax = 2^Nin-1

Definition at line 46 of file [tabela_sin.m](#).

7.33.2.19 Maximun values for inc INCmin = 1

Definition at line 47 of file [tabela_sin.m](#).

7.33.2.20 int_data = 4832

Definition at line 60 of file [tabela_sin.m](#).

7.33.2.21 for k

Initial value:

```
=1:2^Nbits  
id = idfi(k)
```

Definition at line 174 of file [tabela_sin.m](#).

7.33.2.22 MAX

Initial value:

```
=> std_logic_vector(to_unsigned( 1335, 16)), -- valor de contagem maxima
% 1/2^15
% to_sfixed(0.999300704788399, I, -F) when "000000010",
fid = fopen('tabelaSFIXEDsin.txt', 'w')
```

Definition at line 200 of file [tabela_sin.m](#).

7.33.2.23 Gera objeto fi com todos os indices max_phase = 2^16-1

Definition at line 190 of file [tabela_sin.m](#).

7.33.2.24 Minimun values for inc MAXmax =2^Nout-1

Definition at line 48 of file [tabela_sin.m](#).

7.33.2.25 med = cmax/2

Definition at line 110 of file [tabela_sin.m](#).

7.33.2.26 mMIN

Initial value:

```
= to_sfixed(0.08137, 1, -14)
mMAX = to_sfixed(0.8137, 1, -14)
(mMAX-mMIN) / (4832-483)
3/2^14
istep=0.73233/(4832-483)
log2(istep)
TotalBits=16
```

Definition at line 2 of file [tabela_sin.m](#).

7.33.2.27 N16 = 16

Definition at line 52 of file [tabela_sin.m](#).

7.33.2.28 Nbits =11

Definition at line 105 of file [tabela_sin.m](#).

7.33.2.29 Nin = 13

Definition at line 40 of file [tabela_sin.m](#).

7.33.2.30 numero de bits da parte inteira excluindo sinal de entrada Nout = 30

Definition at line 41 of file [tabela_sin.m](#).

7.33.2.31 out

Initial value:

```
= fixptbestprec(istep,16,1)
out = fixptbestexp(istep,TotalBits,1)

to_sfixed(istep, 1, -27)

%% Ganho estatico

Vcc*n*m

Vout=sqrt(2)*220

Vout/400

%% Integrador

fpll=53.33333e6
```

Definition at line 14 of file [tabela_sin.m](#).

7.33.2.32 senofi

Initial value:

```
=fi(y,0,16,0)
seno=round(amp*y+med)
```

Definition at line 157 of file [tabela_sin.m](#).

7.33.2.33 Duty cycle slope

Initial value:

```
=duty60Hz/INC60Hz
foutMax=(INCmax*fin)/MAX
```

Definition at line 69 of file [tabela_sin.m](#).

7.33.2.34 th17bits =fi(bitsrl(theta_pll,13),1,17,0)

Definition at line 83 of file [tabela_sin.m](#).

7.33.2.35 th_ai =th17bits+thasum

Definition at line 85 of file [tabela_sin.m](#).

7.33.2.36 th.bi =th17bits+thbsum

Definition at line 86 of file [tabela_sin.m](#).

7.33.2.37 th.ci =th17bits+thcsum

Definition at line 87 of file [tabela_sin.m](#).

7.33.2.38 thasum =fi(-32767,1,17,0)

Definition at line 78 of file [tabela_sin.m](#).

7.33.2.39 thbsum =fi(-10922,1,17,0)

Definition at line 79 of file [tabela_sin.m](#).

7.33.2.40 thcsum =fi(10922,1,17,0)

Definition at line 80 of file [tabela_sin.m](#).

7.33.2.41 theta_a = fi(th.ai,1,16,0)

Definition at line 91 of file [tabela_sin.m](#).

7.33.2.42 theta_b = fi(th.bi,1,16,0)

Definition at line 92 of file [tabela_sin.m](#).

7.33.2.43 theta_c = fi(th.ci,1,16,0)

Definition at line 93 of file [tabela_sin.m](#).

7.33.2.44 wt=id*2*pi/(2^Nbits)

Definition at line 152 of file [tabela_sin.m](#).

7.33.2.45 y=sin(wt)

Definition at line 153 of file [tabela_sin.m](#).

7.33.2.46 yQ15 =fi(y,1,16,15)

Definition at line 196 of file [tabela_sin.m](#).

7.34 tabela_sin.m

```
00001
00002 mMIN = to_sfixed(0.08137, 1, -14)
00003
00004 mMAX = to_sfixed(0.8137, 1, -14)
```

```

00005
00006 (mMAX-mMIN) / (4832-483)
00007
00008 3/2^14
00009
00010 istep=0.73233/(4832-483)
00011 log2(istep)
00012
00013 TotalBits=16;
00014 out = fixptbestprec(istep,16,1)
00015
00016 out = fixptbestexp(istep,TotalBits,1)
00017
00018
00019 to_sfixed(istep, 1, -27)
00020
00021
00022 %% Ganho estatico
00023
00024
00025 Vcc*n*m
00026
00027 Vout=sqrt(2)*220
00028
00029
00030 Vout/400
00031
00032
00033 %% Integrador
00034
00035
00036 fpll=53.33333e6;
00037 fint=fpll/8;
00038
00039
00040 Nin = 13; % numero de bits da parte inteira excluindo sinal de entrada
00041 Nout = 30; % numero de bits da parte inteira excluindo sinal de saida
00042
00043 fin=fint;
00044
00045
00046 INCmax=2^Nin-1; % Maximun values for inc
00047 INCmin=1; % Minimum values for inc
00048 MAXmax=2^Nout-1;
00049
00050
00051
00052 N16= 16;
00053
00054 a = fi(4832, 0,13, 0);
00055
00056 log2(735801)
00057
00058
00059 MAX=536870911;
00060 int_data=4832;
00061 INC60Hz=4832;
00062
00063 fout=(int_data*fin)/MAX;
00064
00065 fout = 6;
00066 INCdata=fout*MAX/fin;
00067
00068 duty60Hz=0.858794; % Duty cycle
00069 slope=duty60Hz/INC60Hz
00070
00071 foutMax=(INCmax*fin)/MAX;
00072 foutMin=(INCmin*fin)/MAX;
00073
00074 % int_data => std_logic_vector(to_unsigned(4832, 13)) -- daonde vem esse numero?
00075
00076 theta_pll = fi(0:int_data:MAX,0,Nout,0);
00077
00078 thasum=fi(-32767,1,17,0);
00079 thbsum=fi(-10922,1,17,0);
00080 thcsum=fi(10922,1,17,0);
00081
00082
00083 th17bits=fi(bitsrl(theta_pll,13),1,17,0);
00084
00085 th_ai=th17bits+thasum;
00086 th_bi=th17bits+thbsum;
00087 th_ci=th17bits+thcsum;
00088
00089
00090
00091 theta_a = fi(th_ai,1,16,0);

```

```

00092 theta_b = fi(th_bi,1,16,0);
00093 theta_c = fi(th_ci,1,16,0);
00094
00095
00096 %% Limites para operacao
00097
00098
00099
00100
00101 %% Tabela sin para FPGA
00102 fid = fopen('tabela_sin.txt', 'w');
00103
00104
00105 Nbts=11; % log2(1335) = 10.3826
00106 max_phase = 2^16-1;
00107
00108 duty=0.858794; % Duty cycle
00109 cmax = 1335;
00110 med = cmax/2;
00111 amp = med*duty;
00112
00113
00114
00115 fprintf(fid, 'with id select\n');
00116 fprintf(fid, ' sin <=');
00117
00118 for id=1:1:2^Nbts
00119     idd = fi(id-1,0,Nbts,0);
00120     plotar(id)=round(amp*sin((id-1)/(cmax)*pi*2)+med);
00121     fprintf(fid, '      std_logic_vector(to_unsigned(%u, n_bits_c)) when "%s", \n', round(
00122         amp*sin((id-1)/(cmax)*pi*2)+med), idd.bin);
00123 end
00124
00125 fprintf(fid, '      std_logic_vector(to_unsigned(0, n_bits_c)) when others;');
00126
00127 fclose(fid);
00128
00129 %%
00130
00131
00132 duty=0.8137; % Duty cycle
00133
00134 duty=1;
00135 cmax = 1335;
00136
00137 log2(cmax)
00138
00139
00140 fimax=fi(1335,0,16,0);
00141
00142 med = cmax/2;
00143 % amp = med;
00144 amp = med*duty;
00145
00146
00147 Nbts=11; % log2(1335) = 10.3826
00148
00149 id=0:1:2^Nbts-1;
00150 idfi = fi(id,0,Nbts,0);
00151
00152 wt=id*2*pi/(2^Nbts);
00153 y=sin(wt);
00154
00155
00156
00157 % senofi=fi(y,0,16,0)
00158
00159 seno=round(amp*y+med);
00160
00161 figure
00162 plot(seno)
00163 hold all
00164 plot(amp*y+med)
00165
00166
00167 %% Escreve no arquivo .txt
00168 fid = fopen('tabelaDuty0_8137_sin.txt', 'w');
00169
00170 fprintf(fid, 'with id select\n');
00171 fprintf(fid, ' sin <=');
00172
00173
00174 for k=1:2^Nbts
00175     id = idfi(k);
00176     fprintf(fid, '      std_logic_vector(to_unsigned(%u, n_bits_c)) when "%s", \n', seno(k), id
00177         .bin);

```

```

00177 end
00178
00179
00180 fprintf(fid, '           std_logic_vector(to_unsigned(0, n_bits_c)) when others;');
00181
00182 fclose(fid);
00183
00184 %% tabela em ponto fixo:
00185
00186 Nbites=11; % log2(1335) = 10.3826
00187
00188 id=0:1:2^Nbites-1; % gera todos os indices
00189 idfi = fi(id,0,Nbites,0); % Gera objeto fi com todos os indices
00190 max_phase = 2^16-1;
00191
00192 wt=id*2*pi/(2^Nbites);
00193 y=sin(wt);
00194
00195 fimax=fi(1335,0,16,0);
00196 yQ15=fi(y,1,16,15);
00197
00198 comp=fi((yQ15*fimax/2+fimax/2),0,16,0);
00199
00200 % MAX => std_logic_vector(to_unsigned( 1335, 16)), -- valor de contagem maximo
00201
00202 % 1/2^15
00203 % to_sfixed(0.999300704788399, I, -F) when "000000010",
00204
00205 fid = fopen('tabelaSFIXEDsin.txt', 'w');
00206
00207 fprintf(fid, 'with id select\n');
00208 fprintf(fid, '  sin <=' );
00209
00210
00211 for k=1:2^Nbites
00212   id = idfi(k);
00213   fprintf(fid, '          to_sfixed(%1.6f, I, -F) when "%s", \n', y(k), id.bin);
00214 end
00215
00216
00217 fprintf(fid, '          to_sfixed(0, I, -F) when others;');
00218
00219 fclose(fid);
00220
00221
00222
00223
00224
00225

```

7.35 tabela_sin.vhd File Reference

Entities

- **tabela_sin** entity
- **tabela_sin** architecture

7.36 tabela_sin.vhd

```

00001 library ieee;
00002 use ieee.std_logic_1164.all;
00003 use ieee.std_logic_unsigned.all;
00004 use ieee.numeric_std.all;
00005 use ieee.fixed_pkg.all;      --arquivo deve ser adicionado ao projeto
00006
00007 entity tabela_sin is
00008   generic (constant THETA_MAX : integer := 380808000; --eqquivalente a 2*pi
00009             constant n_bits_phase : integer :=16; --numero de bits que representa a fase
00010             constant n_bits_c: integer := 16; --numero de bits da portadora
00011             constant I : integer := 1; --número de bits da parte inteira excluindo sinal
00012             constant F : integer := 14 --número de bits da parte fracionária
00013           );
00014   port(
00015     clk : in std_logic;
00016     theta: in std_logic_vector(15 downto 0);
00017     MAX :  in sfixed(15 downto 0); -- valor de contagem maximo 16 bits em sQ15.0

```

```

00018      ma : in sfixed(I downto -F); -- Indice de modulação em Q15
00019      va : out std_logic_vector(15 downto 0) -- Razão ciclica em Q0
00020  );
00021 end tabela_sin;
00022
00023 architecture tabela_sin of tabela_sin is
00024   signal id : std_logic_vector(10 downto 0);
00025   signal sin : sfixed(I downto -F);
00026   signal va_Q14 : sfixed(1 downto -14); -- signed Q13
00027 begin
00028
00029   id <= theta(n_bits_phase-1 downto n_bits_phase-11); -- 15 downto 5 -
00030   -- resolução do pwm?
00031   -- id <= theta;
00032   -- MAX == to_sfixed(1335,16,0);
00033
00034   process(clk)
00035   begin
00036     if rising_edge(clk) then
00037       va_Q14 <= resize((resize(sin*ma, 1, -14)+ to_sfixed(1, 1, -14)),1,-14); -- Valor
00038       entre +=1
00039       va <= to_slv(resize(va_Q14*scalb(MAX, -1),15,0)); --
00040     end if;
00041   end process;
00042   with id select
00043     sin <= to_sfixed(0.000000, I, -F) when "000000000000",
00044     to_sfixed(0.003068, I, -F) when "000000000001",
00045     to_sfixed(0.006136, I, -F) when "000000000010",
00046     to_sfixed(0.009204, I, -F) when "000000000011",
00047     to_sfixed(0.012272, I, -F) when "000000000100",
00048     to_sfixed(0.015339, I, -F) when "000000000101",
00049     to_sfixed(0.018407, I, -F) when "000000000110",
00050     to_sfixed(0.021474, I, -F) when "000000000111",
00051     to_sfixed(0.024541, I, -F) when "000000001000",
00052     to_sfixed(0.027608, I, -F) when "000000001001",
00053     to_sfixed(0.030675, I, -F) when "000000001010",
00054     to_sfixed(0.033741, I, -F) when "000000001011",
00055     to_sfixed(0.036807, I, -F) when "000000001100",
00056     to_sfixed(0.039873, I, -F) when "000000001101",
00057     to_sfixed(0.042938, I, -F) when "000000001110",
00058     to_sfixed(0.046003, I, -F) when "000000001111",
00059     to_sfixed(0.049068, I, -F) when "000000010000",
00060     to_sfixed(0.052132, I, -F) when "000000010001",
00061     to_sfixed(0.055195, I, -F) when "000000010010",
00062     to_sfixed(0.058258, I, -F) when "000000010011",
00063     to_sfixed(0.061321, I, -F) when "000000010100",
00064     to_sfixed(0.064383, I, -F) when "000000010101",
00065     to_sfixed(0.067444, I, -F) when "000000010110",
00066     to_sfixed(0.070505, I, -F) when "000000010111",
00067     to_sfixed(0.073565, I, -F) when "000000011000",
00068     to_sfixed(0.076624, I, -F) when "000000011001",
00069     to_sfixed(0.079682, I, -F) when "000000011010",
00070     to_sfixed(0.082740, I, -F) when "000000011011",
00071     to_sfixed(0.085797, I, -F) when "000000011100",
00072     to_sfixed(0.088854, I, -F) when "000000011101",
00073     to_sfixed(0.091909, I, -F) when "000000011110",
00074     to_sfixed(0.094963, I, -F) when "000000011111",
00075     to_sfixed(0.098017, I, -F) when "00000100000",
00076     to_sfixed(0.101070, I, -F) when "00000100001",
00077     to_sfixed(0.104122, I, -F) when "00000100010",
00078     to_sfixed(0.107172, I, -F) when "00000100011",
00079     to_sfixed(0.110222, I, -F) when "00000100100",
00080     to_sfixed(0.113271, I, -F) when "00000100101",
00081     to_sfixed(0.116319, I, -F) when "00000100110",
00082     to_sfixed(0.119365, I, -F) when "00000100111",
00083     to_sfixed(0.122411, I, -F) when "00000101000",
00084     to_sfixed(0.125455, I, -F) when "00000101001",
00085     to_sfixed(0.128498, I, -F) when "00000101010",
00086     to_sfixed(0.131540, I, -F) when "00000101011",
00087     to_sfixed(0.134581, I, -F) when "00000101100",
00088     to_sfixed(0.137620, I, -F) when "00000101101",
00089     to_sfixed(0.140658, I, -F) when "00000101110",
00090     to_sfixed(0.143695, I, -F) when "00000101111",
00091     to_sfixed(0.146730, I, -F) when "00000110000",
00092     to_sfixed(0.149765, I, -F) when "00000110001",
00093     to_sfixed(0.152797, I, -F) when "00000110010",
00094     to_sfixed(0.155828, I, -F) when "00000110011",
00095     to_sfixed(0.158858, I, -F) when "00000110100",
00096     to_sfixed(0.161886, I, -F) when "00000110101",
00097     to_sfixed(0.164913, I, -F) when "00000110110",
00098     to_sfixed(0.167938, I, -F) when "00000110111",
00099     to_sfixed(0.170962, I, -F) when "00000111000",
00100     to_sfixed(0.173984, I, -F) when "00000111001",
00101     to_sfixed(0.177004, I, -F) when "00000111010",
00102     to_sfixed(0.180023, I, -F) when "00000111011",
00103     to_sfixed(0.183040, I, -F) when "00000111100",
00104     to_sfixed(0.186055, I, -F) when "00000111101",

```

```
00103      to_sfixed(0.189069, I, -F) when "00000111110",
00104      to_sfixed(0.192080, I, -F) when "00000111111",
00105      to_sfixed(0.195090, I, -F) when "00001000000",
00106      to_sfixed(0.198098, I, -F) when "00001000001",
00107      to_sfixed(0.201105, I, -F) when "00001000010",
00108      to_sfixed(0.204109, I, -F) when "00001000011",
00109      to_sfixed(0.207111, I, -F) when "00001000100",
00110      to_sfixed(0.210112, I, -F) when "00001000101",
00111      to_sfixed(0.213110, I, -F) when "00001000110",
00112      to_sfixed(0.216107, I, -F) when "00001000111",
00113      to_sfixed(0.219101, I, -F) when "00001001000",
00114      to_sfixed(0.222094, I, -F) when "00001001001",
00115      to_sfixed(0.225084, I, -F) when "00001001010",
00116      to_sfixed(0.228072, I, -F) when "00001001011",
00117      to_sfixed(0.231058, I, -F) when "00001001100",
00118      to_sfixed(0.234042, I, -F) when "00001001101",
00119      to_sfixed(0.237024, I, -F) when "00001001110",
00120      to_sfixed(0.240003, I, -F) when "00001001111",
00121      to_sfixed(0.242980, I, -F) when "00001010000",
00122      to_sfixed(0.245955, I, -F) when "00001010001",
00123      to_sfixed(0.248928, I, -F) when "00001010010",
00124      to_sfixed(0.251898, I, -F) when "00001010011",
00125      to_sfixed(0.254866, I, -F) when "00001010100",
00126      to_sfixed(0.257831, I, -F) when "00001010101",
00127      to_sfixed(0.260794, I, -F) when "00001010110",
00128      to_sfixed(0.263755, I, -F) when "00001010111",
00129      to_sfixed(0.266713, I, -F) when "00001011000",
00130      to_sfixed(0.269668, I, -F) when "00001011001",
00131      to_sfixed(0.272621, I, -F) when "00001011010",
00132      to_sfixed(0.275572, I, -F) when "00001011011",
00133      to_sfixed(0.278520, I, -F) when "00001011100",
00134      to_sfixed(0.281465, I, -F) when "00001011101",
00135      to_sfixed(0.284408, I, -F) when "00001011110",
00136      to_sfixed(0.287347, I, -F) when "00001011111",
00137      to_sfixed(0.290285, I, -F) when "00001100000",
00138      to_sfixed(0.293219, I, -F) when "00001100001",
00139      to_sfixed(0.296151, I, -F) when "00001100010",
00140      to_sfixed(0.299080, I, -F) when "00001100011",
00141      to_sfixed(0.302006, I, -F) when "00001100100",
00142      to_sfixed(0.304929, I, -F) when "00001100101",
00143      to_sfixed(0.307850, I, -F) when "00001100110",
00144      to_sfixed(0.310767, I, -F) when "00001100111",
00145      to_sfixed(0.313682, I, -F) when "00001101000",
00146      to_sfixed(0.316593, I, -F) when "00001101001",
00147      to_sfixed(0.319502, I, -F) when "00001101010",
00148      to_sfixed(0.322408, I, -F) when "00001101011",
00149      to_sfixed(0.325310, I, -F) when "00001101100",
00150      to_sfixed(0.328210, I, -F) when "00001101101",
00151      to_sfixed(0.331106, I, -F) when "00001101110",
00152      to_sfixed(0.334000, I, -F) when "00001101111",
00153      to_sfixed(0.336890, I, -F) when "00001110000",
00154      to_sfixed(0.339777, I, -F) when "00001110001",
00155      to_sfixed(0.342661, I, -F) when "00001110010",
00156      to_sfixed(0.345541, I, -F) when "00001110011",
00157      to_sfixed(0.348419, I, -F) when "00001110100",
00158      to_sfixed(0.351293, I, -F) when "00001110101",
00159      to_sfixed(0.354164, I, -F) when "00001110110",
00160      to_sfixed(0.357031, I, -F) when "00001110111",
00161      to_sfixed(0.359895, I, -F) when "00001111000",
00162      to_sfixed(0.362756, I, -F) when "00001111001",
00163      to_sfixed(0.365613, I, -F) when "00001111010",
00164      to_sfixed(0.368467, I, -F) when "00001111011",
00165      to_sfixed(0.371317, I, -F) when "00001111100",
00166      to_sfixed(0.374164, I, -F) when "00001111101",
00167      to_sfixed(0.377007, I, -F) when "00001111110",
00168      to_sfixed(0.379847, I, -F) when "00001111111",
00169      to_sfixed(0.382683, I, -F) when "00010000000",
00170      to_sfixed(0.385516, I, -F) when "00010000001",
00171      to_sfixed(0.388345, I, -F) when "00010000010",
00172      to_sfixed(0.391170, I, -F) when "00010000011",
00173      to_sfixed(0.393992, I, -F) when "00010000100",
00174      to_sfixed(0.396810, I, -F) when "00010000101",
00175      to_sfixed(0.399624, I, -F) when "00010000110",
00176      to_sfixed(0.402435, I, -F) when "00010000111",
00177      to_sfixed(0.405241, I, -F) when "00010001000",
00178      to_sfixed(0.408044, I, -F) when "00010001001",
00179      to_sfixed(0.410843, I, -F) when "00010001010",
00180      to_sfixed(0.413638, I, -F) when "00010001011",
00181      to_sfixed(0.416430, I, -F) when "00010001100",
00182      to_sfixed(0.419217, I, -F) when "00010001101",
00183      to_sfixed(0.422000, I, -F) when "00010001110",
00184      to_sfixed(0.424780, I, -F) when "00010001111",
00185      to_sfixed(0.427555, I, -F) when "00010010000",
00186      to_sfixed(0.430326, I, -F) when "00010010001",
00187      to_sfixed(0.433094, I, -F) when "00010010010",
00188      to_sfixed(0.435857, I, -F) when "00010010011",
00189      to_sfixed(0.438616, I, -F) when "00010010100",
```

```

00190      to_sfixed(0.441371, I, -F) when "00010010101",
00191      to_sfixed(0.444122, I, -F) when "00010010110",
00192      to_sfixed(0.446869, I, -F) when "00010010111",
00193      to_sfixed(0.449611, I, -F) when "00010011000",
00194      to_sfixed(0.452350, I, -F) when "00010011001",
00195      to_sfixed(0.455084, I, -F) when "00010011010",
00196      to_sfixed(0.457813, I, -F) when "00010011011",
00197      to_sfixed(0.460539, I, -F) when "00010011100",
00198      to_sfixed(0.463260, I, -F) when "00010011101",
00199      to_sfixed(0.465976, I, -F) when "00010011110",
00200      to_sfixed(0.468689, I, -F) when "00010011111",
00201      to_sfixed(0.471397, I, -F) when "00010100000",
00202      to_sfixed(0.474100, I, -F) when "00010100001",
00203      to_sfixed(0.476799, I, -F) when "00010100010",
00204      to_sfixed(0.479494, I, -F) when "00010100011",
00205      to_sfixed(0.482184, I, -F) when "00010100100",
00206      to_sfixed(0.484869, I, -F) when "00010100101",
00207      to_sfixed(0.487550, I, -F) when "00010100110",
00208      to_sfixed(0.490226, I, -F) when "00010100111",
00209      to_sfixed(0.492898, I, -F) when "00010101000",
00210      to_sfixed(0.495565, I, -F) when "00010101001",
00211      to_sfixed(0.498228, I, -F) when "00010101010",
00212      to_sfixed(0.500885, I, -F) when "00010101011",
00213      to_sfixed(0.503538, I, -F) when "00010101100",
00214      to_sfixed(0.506187, I, -F) when "00010101101",
00215      to_sfixed(0.508830, I, -F) when "00010101110",
00216      to_sfixed(0.511469, I, -F) when "00010101111",
00217      to_sfixed(0.514103, I, -F) when "00010110000",
00218      to_sfixed(0.516732, I, -F) when "00010110001",
00219      to_sfixed(0.519356, I, -F) when "00010110010",
00220      to_sfixed(0.521975, I, -F) when "00010110011",
00221      to_sfixed(0.524590, I, -F) when "00010110100",
00222      to_sfixed(0.527199, I, -F) when "00010110101",
00223      to_sfixed(0.529804, I, -F) when "00010110110",
00224      to_sfixed(0.532403, I, -F) when "00010110111",
00225      to_sfixed(0.534998, I, -F) when "00010111000",
00226      to_sfixed(0.537587, I, -F) when "00010111001",
00227      to_sfixed(0.540171, I, -F) when "00010111010",
00228      to_sfixed(0.542751, I, -F) when "00010111011",
00229      to_sfixed(0.545325, I, -F) when "00010111100",
00230      to_sfixed(0.547894, I, -F) when "00010111101",
00231      to_sfixed(0.550458, I, -F) when "00010111110",
00232      to_sfixed(0.553017, I, -F) when "00010111111",
00233      to_sfixed(0.555570, I, -F) when "00011000000",
00234      to_sfixed(0.558119, I, -F) when "00011000001",
00235      to_sfixed(0.560662, I, -F) when "00011000010",
00236      to_sfixed(0.563199, I, -F) when "00011000011",
00237      to_sfixed(0.565732, I, -F) when "00011000100",
00238      to_sfixed(0.568259, I, -F) when "00011000101",
00239      to_sfixed(0.570781, I, -F) when "00011000110",
00240      to_sfixed(0.573297, I, -F) when "00011000111",
00241      to_sfixed(0.575808, I, -F) when "00011001000",
00242      to_sfixed(0.578314, I, -F) when "00011001001",
00243      to_sfixed(0.580814, I, -F) when "00011001010",
00244      to_sfixed(0.583309, I, -F) when "00011001011",
00245      to_sfixed(0.585798, I, -F) when "00011001100",
00246      to_sfixed(0.588282, I, -F) when "00011001101",
00247      to_sfixed(0.590760, I, -F) when "00011001110",
00248      to_sfixed(0.593232, I, -F) when "00011001111",
00249      to_sfixed(0.595699, I, -F) when "00011010000",
00250      to_sfixed(0.598161, I, -F) when "00011010001",
00251      to_sfixed(0.600616, I, -F) when "00011010010",
00252      to_sfixed(0.603067, I, -F) when "00011010011",
00253      to_sfixed(0.605511, I, -F) when "00011010100",
00254      to_sfixed(0.607950, I, -F) when "00011010101",
00255      to_sfixed(0.610383, I, -F) when "00011010110",
00256      to_sfixed(0.612810, I, -F) when "00011010111",
00257      to_sfixed(0.615232, I, -F) when "00011011000",
00258      to_sfixed(0.617647, I, -F) when "00011011001",
00259      to_sfixed(0.620057, I, -F) when "00011011010",
00260      to_sfixed(0.622461, I, -F) when "00011011011",
00261      to_sfixed(0.624859, I, -F) when "00011011100",
00262      to_sfixed(0.627252, I, -F) when "00011011101",
00263      to_sfixed(0.629638, I, -F) when "00011011110",
00264      to_sfixed(0.632019, I, -F) when "00011011111",
00265      to_sfixed(0.634393, I, -F) when "00011100000",
00266      to_sfixed(0.636762, I, -F) when "00011100001",
00267      to_sfixed(0.639124, I, -F) when "00011100010",
00268      to_sfixed(0.641481, I, -F) when "00011100011",
00269      to_sfixed(0.643832, I, -F) when "00011100100",
00270      to_sfixed(0.646176, I, -F) when "00011100101",
00271      to_sfixed(0.648514, I, -F) when "00011100110",
00272      to_sfixed(0.650847, I, -F) when "00011100111",
00273      to_sfixed(0.653173, I, -F) when "00011101000",
00274      to_sfixed(0.655493, I, -F) when "00011101001",
00275      to_sfixed(0.657807, I, -F) when "00011101010",
00276      to_sfixed(0.660114, I, -F) when "00011101011",

```

```

00277      to_sfixed(0.662416, I, -F) when "00011101100",
00278      to_sfixed(0.664711, I, -F) when "00011101101",
00279      to_sfixed(0.667000, I, -F) when "00011101110",
00280      to_sfixed(0.669283, I, -F) when "00011101111",
00281      to_sfixed(0.671559, I, -F) when "00011110000",
00282      to_sfixed(0.673829, I, -F) when "00011110001",
00283      to_sfixed(0.676093, I, -F) when "00011110010",
00284      to_sfixed(0.678350, I, -F) when "00011110011",
00285      to_sfixed(0.680601, I, -F) when "00011110100",
00286      to_sfixed(0.682846, I, -F) when "00011110101",
00287      to_sfixed(0.685084, I, -F) when "00011110110",
00288      to_sfixed(0.687315, I, -F) when "00011110111",
00289      to_sfixed(0.689541, I, -F) when "00011111000",
00290      to_sfixed(0.691759, I, -F) when "00011111001",
00291      to_sfixed(0.693971, I, -F) when "00011111010",
00292      to_sfixed(0.696177, I, -F) when "00011111011",
00293      to_sfixed(0.698376, I, -F) when "00011111100",
00294      to_sfixed(0.700569, I, -F) when "00011111101",
00295      to_sfixed(0.702755, I, -F) when "00011111110",
00296      to_sfixed(0.704934, I, -F) when "00011111111",
00297      to_sfixed(0.707107, I, -F) when "00100000000",
00298      to_sfixed(0.709273, I, -F) when "00100000001",
00299      to_sfixed(0.711432, I, -F) when "00100000010",
00300      to_sfixed(0.713585, I, -F) when "00100000011",
00301      to_sfixed(0.715731, I, -F) when "00100000100",
00302      to_sfixed(0.717870, I, -F) when "00100000101",
00303      to_sfixed(0.720003, I, -F) when "00100000110",
00304      to_sfixed(0.722128, I, -F) when "00100000111",
00305      to_sfixed(0.724247, I, -F) when "00100001000",
00306      to_sfixed(0.726359, I, -F) when "00100001001",
00307      to_sfixed(0.728464, I, -F) when "00100001010",
00308      to_sfixed(0.730563, I, -F) when "00100001011",
00309      to_sfixed(0.732654, I, -F) when "00100001100",
00310      to_sfixed(0.734739, I, -F) when "00100001101",
00311      to_sfixed(0.736817, I, -F) when "00100001110",
00312      to_sfixed(0.738887, I, -F) when "00100001111",
00313      to_sfixed(0.740951, I, -F) when "00100010000",
00314      to_sfixed(0.743008, I, -F) when "00100010001",
00315      to_sfixed(0.745058, I, -F) when "00100010010",
00316      to_sfixed(0.747101, I, -F) when "00100010011",
00317      to_sfixed(0.749136, I, -F) when "00100010100",
00318      to_sfixed(0.751165, I, -F) when "00100010101",
00319      to_sfixed(0.753187, I, -F) when "00100010110",
00320      to_sfixed(0.755201, I, -F) when "00100010111",
00321      to_sfixed(0.757209, I, -F) when "00100011000",
00322      to_sfixed(0.759209, I, -F) when "00100011001",
00323      to_sfixed(0.761202, I, -F) when "00100011010",
00324      to_sfixed(0.763188, I, -F) when "00100011011",
00325      to_sfixed(0.765167, I, -F) when "00100011100",
00326      to_sfixed(0.767139, I, -F) when "00100011101",
00327      to_sfixed(0.769103, I, -F) when "00100011110",
00328      to_sfixed(0.771061, I, -F) when "00100011111",
00329      to_sfixed(0.773010, I, -F) when "00100100000",
00330      to_sfixed(0.774953, I, -F) when "00100100001",
00331      to_sfixed(0.776888, I, -F) when "00100100010",
00332      to_sfixed(0.778817, I, -F) when "00100100011",
00333      to_sfixed(0.780737, I, -F) when "00100100100",
00334      to_sfixed(0.782651, I, -F) when "00100100101",
00335      to_sfixed(0.784557, I, -F) when "00100100110",
00336      to_sfixed(0.786455, I, -F) when "00100100111",
00337      to_sfixed(0.788346, I, -F) when "00100101000",
00338      to_sfixed(0.790230, I, -F) when "00100101001",
00339      to_sfixed(0.792107, I, -F) when "00100101010",
00340      to_sfixed(0.793975, I, -F) when "00100101011",
00341      to_sfixed(0.795837, I, -F) when "00100101100",
00342      to_sfixed(0.797691, I, -F) when "00100101101",
00343      to_sfixed(0.799537, I, -F) when "00100101110",
00344      to_sfixed(0.801376, I, -F) when "00100101111",
00345      to_sfixed(0.803208, I, -F) when "00100110000",
00346      to_sfixed(0.805031, I, -F) when "00100110001",
00347      to_sfixed(0.806848, I, -F) when "00100110010",
00348      to_sfixed(0.808656, I, -F) when "00100110011",
00349      to_sfixed(0.810457, I, -F) when "00100110100",
00350      to_sfixed(0.812251, I, -F) when "00100110101",
00351      to_sfixed(0.814036, I, -F) when "00100110110",
00352      to_sfixed(0.815814, I, -F) when "00100110111",
00353      to_sfixed(0.817585, I, -F) when "00100111000",
00354      to_sfixed(0.819348, I, -F) when "00100111001",
00355      to_sfixed(0.821103, I, -F) when "00100111010",
00356      to_sfixed(0.822850, I, -F) when "00100111011",
00357      to_sfixed(0.824589, I, -F) when "00100111100",
00358      to_sfixed(0.826321, I, -F) when "00100111101",
00359      to_sfixed(0.828045, I, -F) when "00100111110",
00360      to_sfixed(0.829761, I, -F) when "00100111111",
00361      to_sfixed(0.831470, I, -F) when "00101000000",
00362      to_sfixed(0.833170, I, -F) when "00101000001",
00363      to_sfixed(0.834863, I, -F) when "00101000010",

```

```

00364      to_sfixed(0.836548, I, -F) when "00101000011",
00365      to_sfixed(0.838225, I, -F) when "00101000100",
00366      to_sfixed(0.839894, I, -F) when "00101000101",
00367      to_sfixed(0.841555, I, -F) when "00101000110",
00368      to_sfixed(0.843208, I, -F) when "00101000111",
00369      to_sfixed(0.844854, I, -F) when "00101001000",
00370      to_sfixed(0.846491, I, -F) when "00101001001",
00371      to_sfixed(0.848120, I, -F) when "00101001010",
00372      to_sfixed(0.849742, I, -F) when "00101001011",
00373      to_sfixed(0.851355, I, -F) when "00101001100",
00374      to_sfixed(0.852961, I, -F) when "00101001101",
00375      to_sfixed(0.854558, I, -F) when "00101001110",
00376      to_sfixed(0.856147, I, -F) when "00101001111",
00377      to_sfixed(0.857729, I, -F) when "00101010000",
00378      to_sfixed(0.859302, I, -F) when "00101010001",
00379      to_sfixed(0.860867, I, -F) when "00101010010",
00380      to_sfixed(0.862424, I, -F) when "00101010011",
00381      to_sfixed(0.863973, I, -F) when "00101010100",
00382      to_sfixed(0.865514, I, -F) when "00101010101",
00383      to_sfixed(0.867046, I, -F) when "00101010110",
00384      to_sfixed(0.868571, I, -F) when "00101010111",
00385      to_sfixed(0.870087, I, -F) when "00101011000",
00386      to_sfixed(0.871595, I, -F) when "00101011001",
00387      to_sfixed(0.873095, I, -F) when "00101011010",
00388      to_sfixed(0.874587, I, -F) when "00101011011",
00389      to_sfixed(0.876070, I, -F) when "00101011100",
00390      to_sfixed(0.877545, I, -F) when "00101011101",
00391      to_sfixed(0.879012, I, -F) when "00101011110",
00392      to_sfixed(0.880471, I, -F) when "00101011111",
00393      to_sfixed(0.881921, I, -F) when "00101100000",
00394      to_sfixed(0.883363, I, -F) when "00101100001",
00395      to_sfixed(0.884797, I, -F) when "00101100010",
00396      to_sfixed(0.886223, I, -F) when "00101100011",
00397      to_sfixed(0.887640, I, -F) when "00101100100",
00398      to_sfixed(0.889048, I, -F) when "00101100101",
00399      to_sfixed(0.890449, I, -F) when "00101100110",
00400      to_sfixed(0.891841, I, -F) when "00101100111",
00401      to_sfixed(0.893224, I, -F) when "00101101000",
00402      to_sfixed(0.894599, I, -F) when "00101101001",
00403      to_sfixed(0.895966, I, -F) when "00101101010",
00404      to_sfixed(0.897325, I, -F) when "00101101011",
00405      to_sfixed(0.898674, I, -F) when "00101101100",
00406      to_sfixed(0.900016, I, -F) when "00101101101",
00407      to_sfixed(0.901349, I, -F) when "00101101110",
00408      to_sfixed(0.902673, I, -F) when "00101101111",
00409      to_sfixed(0.903989, I, -F) when "00101110000",
00410      to_sfixed(0.905297, I, -F) when "00101110001",
00411      to_sfixed(0.906596, I, -F) when "00101110010",
00412      to_sfixed(0.907886, I, -F) when "00101110011",
00413      to_sfixed(0.909168, I, -F) when "00101110100",
00414      to_sfixed(0.910441, I, -F) when "00101110101",
00415      to_sfixed(0.911706, I, -F) when "00101110110",
00416      to_sfixed(0.912962, I, -F) when "00101110111",
00417      to_sfixed(0.914210, I, -F) when "00101111000",
00418      to_sfixed(0.915449, I, -F) when "00101111001",
00419      to_sfixed(0.916679, I, -F) when "00101111010",
00420      to_sfixed(0.917901, I, -F) when "00101111011",
00421      to_sfixed(0.919114, I, -F) when "00101111100",
00422      to_sfixed(0.920318, I, -F) when "00101111101",
00423      to_sfixed(0.921514, I, -F) when "00101111110",
00424      to_sfixed(0.922701, I, -F) when "00101111111",
00425      to_sfixed(0.923880, I, -F) when "00110000000",
00426      to_sfixed(0.925049, I, -F) when "00110000001",
00427      to_sfixed(0.926210, I, -F) when "00110000010",
00428      to_sfixed(0.927363, I, -F) when "00110000011",
00429      to_sfixed(0.928506, I, -F) when "00110000100",
00430      to_sfixed(0.929641, I, -F) when "00110000101",
00431      to_sfixed(0.930767, I, -F) when "00110000110",
00432      to_sfixed(0.931884, I, -F) when "00110000111",
00433      to_sfixed(0.932993, I, -F) when "00110001000",
00434      to_sfixed(0.934093, I, -F) when "00110001001",
00435      to_sfixed(0.935184, I, -F) when "00110001010",
00436      to_sfixed(0.936266, I, -F) when "00110001011",
00437      to_sfixed(0.937339, I, -F) when "00110001100",
00438      to_sfixed(0.938404, I, -F) when "00110001101",
00439      to_sfixed(0.939459, I, -F) when "00110001110",
00440      to_sfixed(0.940506, I, -F) when "00110001111",
00441      to_sfixed(0.941544, I, -F) when "00110010000",
00442      to_sfixed(0.942573, I, -F) when "00110010001",
00443      to_sfixed(0.943593, I, -F) when "00110010010",
00444      to_sfixed(0.944605, I, -F) when "00110010011",
00445      to_sfixed(0.945607, I, -F) when "00110010100",
00446      to_sfixed(0.946601, I, -F) when "00110010101",
00447      to_sfixed(0.947586, I, -F) when "00110010110",
00448      to_sfixed(0.948561, I, -F) when "00110010111",
00449      to_sfixed(0.949528, I, -F) when "00110011000",
00450      to_sfixed(0.950486, I, -F) when "00110011001",

```

```
00451      to_sfixed(0.951435, I, -F) when "00110011010",
00452      to_sfixed(0.952375, I, -F) when "00110011011",
00453      to_sfixed(0.953306, I, -F) when "00110011100",
00454      to_sfixed(0.954228, I, -F) when "00110011101",
00455      to_sfixed(0.955141, I, -F) when "00110011110",
00456      to_sfixed(0.956045, I, -F) when "00110011111",
00457      to_sfixed(0.956940, I, -F) when "00110100000",
00458      to_sfixed(0.957826, I, -F) when "00110100001",
00459      to_sfixed(0.958703, I, -F) when "00110100010",
00460      to_sfixed(0.959572, I, -F) when "00110100011",
00461      to_sfixed(0.960431, I, -F) when "00110100100",
00462      to_sfixed(0.961280, I, -F) when "00110100101",
00463      to_sfixed(0.962121, I, -F) when "00110100110",
00464      to_sfixed(0.962953, I, -F) when "00110100111",
00465      to_sfixed(0.963776, I, -F) when "00110101000",
00466      to_sfixed(0.964590, I, -F) when "00110101001",
00467      to_sfixed(0.965394, I, -F) when "00110101010",
00468      to_sfixed(0.966190, I, -F) when "00110101011",
00469      to_sfixed(0.966976, I, -F) when "00110101100",
00470      to_sfixed(0.967754, I, -F) when "00110101101",
00471      to_sfixed(0.968522, I, -F) when "00110101110",
00472      to_sfixed(0.969281, I, -F) when "00110101111",
00473      to_sfixed(0.970031, I, -F) when "00110110000",
00474      to_sfixed(0.970772, I, -F) when "00110110001",
00475      to_sfixed(0.971504, I, -F) when "00110110010",
00476      to_sfixed(0.972226, I, -F) when "00110110011",
00477      to_sfixed(0.972940, I, -F) when "00110110100",
00478      to_sfixed(0.973644, I, -F) when "00110110101",
00479      to_sfixed(0.974339, I, -F) when "00110110110",
00480      to_sfixed(0.975025, I, -F) when "00110110111",
00481      to_sfixed(0.975702, I, -F) when "00110111000",
00482      to_sfixed(0.976370, I, -F) when "00110111001",
00483      to_sfixed(0.977028, I, -F) when "00110111010",
00484      to_sfixed(0.977677, I, -F) when "00110111011",
00485      to_sfixed(0.978317, I, -F) when "00110111100",
00486      to_sfixed(0.978948, I, -F) when "00110111101",
00487      to_sfixed(0.979570, I, -F) when "00110111110",
00488      to_sfixed(0.980182, I, -F) when "00110111111",
00489      to_sfixed(0.980785, I, -F) when "00111000000",
00490      to_sfixed(0.981379, I, -F) when "00111000001",
00491      to_sfixed(0.981964, I, -F) when "00111000010",
00492      to_sfixed(0.982539, I, -F) when "00111000011",
00493      to_sfixed(0.983105, I, -F) when "00111000100",
00494      to_sfixed(0.983662, I, -F) when "00111000101",
00495      to_sfixed(0.984210, I, -F) when "00111000110",
00496      to_sfixed(0.984749, I, -F) when "00111000111",
00497      to_sfixed(0.985278, I, -F) when "00111001000",
00498      to_sfixed(0.985798, I, -F) when "00111001001",
00499      to_sfixed(0.986308, I, -F) when "00111001010",
00500      to_sfixed(0.986809, I, -F) when "00111001011",
00501      to_sfixed(0.987301, I, -F) when "00111001100",
00502      to_sfixed(0.987784, I, -F) when "00111001101",
00503      to_sfixed(0.988258, I, -F) when "00111001110",
00504      to_sfixed(0.988722, I, -F) when "00111001111",
00505      to_sfixed(0.989177, I, -F) when "00111010000",
00506      to_sfixed(0.989622, I, -F) when "00111010001",
00507      to_sfixed(0.990058, I, -F) when "00111010010",
00508      to_sfixed(0.990485, I, -F) when "00111010011",
00509      to_sfixed(0.990903, I, -F) when "00111010100",
00510      to_sfixed(0.991311, I, -F) when "00111010101",
00511      to_sfixed(0.991710, I, -F) when "00111010110",
00512      to_sfixed(0.992099, I, -F) when "00111010111",
00513      to_sfixed(0.992480, I, -F) when "00111011000",
00514      to_sfixed(0.992850, I, -F) when "00111011001",
00515      to_sfixed(0.993212, I, -F) when "00111011010",
00516      to_sfixed(0.993564, I, -F) when "00111011011",
00517      to_sfixed(0.993907, I, -F) when "00111011100",
00518      to_sfixed(0.994240, I, -F) when "00111011101",
00519      to_sfixed(0.994565, I, -F) when "00111011110",
00520      to_sfixed(0.994879, I, -F) when "00111011111",
00521      to_sfixed(0.995185, I, -F) when "00111100000",
00522      to_sfixed(0.995481, I, -F) when "00111100001",
00523      to_sfixed(0.995767, I, -F) when "00111100010",
00524      to_sfixed(0.996045, I, -F) when "00111100011",
00525      to_sfixed(0.996313, I, -F) when "00111100100",
00526      to_sfixed(0.996571, I, -F) when "00111100101",
00527      to_sfixed(0.996820, I, -F) when "00111100110",
00528      to_sfixed(0.997060, I, -F) when "00111100111",
00529      to_sfixed(0.997290, I, -F) when "00111101000",
00530      to_sfixed(0.997511, I, -F) when "00111101001",
00531      to_sfixed(0.997723, I, -F) when "00111101010",
00532      to_sfixed(0.997925, I, -F) when "00111101011",
00533      to_sfixed(0.998118, I, -F) when "00111101100",
00534      to_sfixed(0.998302, I, -F) when "00111101101",
00535      to_sfixed(0.998476, I, -F) when "00111101110",
00536      to_sfixed(0.998640, I, -F) when "00111101111",
00537      to_sfixed(0.998795, I, -F) when "00111100000",
```

```

00538      to_sfixed(0.998941, I, -F) when "00111110001",
00539      to_sfixed(0.999078, I, -F) when "00111110010",
00540      to_sfixed(0.999205, I, -F) when "00111110011",
00541      to_sfixed(0.999322, I, -F) when "00111110100",
00542      to_sfixed(0.999431, I, -F) when "00111110101",
00543      to_sfixed(0.999529, I, -F) when "00111110110",
00544      to_sfixed(0.999619, I, -F) when "00111110111",
00545      to_sfixed(0.999699, I, -F) when "0011111000",
00546      to_sfixed(0.999769, I, -F) when "0011111001",
00547      to_sfixed(0.999831, I, -F) when "0011111010",
00548      to_sfixed(0.999882, I, -F) when "0011111011",
00549      to_sfixed(0.999925, I, -F) when "0011111100",
00550      to_sfixed(0.999958, I, -F) when "00111111101",
00551      to_sfixed(0.999981, I, -F) when "00111111110",
00552      to_sfixed(0.999995, I, -F) when "00111111111",
00553      to_sfixed(1.000000, I, -F) when "01000000000",
00554      to_sfixed(0.999995, I, -F) when "01000000001",
00555      to_sfixed(0.999981, I, -F) when "01000000010",
00556      to_sfixed(0.999958, I, -F) when "01000000011",
00557      to_sfixed(0.999925, I, -F) when "010000000100",
00558      to_sfixed(0.999882, I, -F) when "010000000101",
00559      to_sfixed(0.999831, I, -F) when "010000000110",
00560      to_sfixed(0.999769, I, -F) when "010000000111",
00561      to_sfixed(0.999699, I, -F) when "010000001000",
00562      to_sfixed(0.999619, I, -F) when "010000001001",
00563      to_sfixed(0.999529, I, -F) when "010000001010",
00564      to_sfixed(0.999431, I, -F) when "010000001011",
00565      to_sfixed(0.999322, I, -F) when "010000001100",
00566      to_sfixed(0.999205, I, -F) when "010000001101",
00567      to_sfixed(0.999078, I, -F) when "010000001110",
00568      to_sfixed(0.998941, I, -F) when "010000001111",
00569      to_sfixed(0.998795, I, -F) when "01000010000",
00570      to_sfixed(0.998640, I, -F) when "01000010001",
00571      to_sfixed(0.998476, I, -F) when "01000010010",
00572      to_sfixed(0.998302, I, -F) when "01000010011",
00573      to_sfixed(0.998118, I, -F) when "01000010100",
00574      to_sfixed(0.997925, I, -F) when "01000010101",
00575      to_sfixed(0.997723, I, -F) when "01000010110",
00576      to_sfixed(0.997511, I, -F) when "01000010111",
00577      to_sfixed(0.997290, I, -F) when "01000011000",
00578      to_sfixed(0.997060, I, -F) when "01000011001",
00579      to_sfixed(0.996820, I, -F) when "01000011010",
00580      to_sfixed(0.996571, I, -F) when "01000011011",
00581      to_sfixed(0.996313, I, -F) when "01000011100",
00582      to_sfixed(0.996045, I, -F) when "01000011101",
00583      to_sfixed(0.995767, I, -F) when "01000011110",
00584      to_sfixed(0.995481, I, -F) when "01000011111",
00585      to_sfixed(0.995185, I, -F) when "01000100000",
00586      to_sfixed(0.994879, I, -F) when "01000100001",
00587      to_sfixed(0.994565, I, -F) when "01000100010",
00588      to_sfixed(0.994240, I, -F) when "01000100011",
00589      to_sfixed(0.993907, I, -F) when "01000100100",
00590      to_sfixed(0.993564, I, -F) when "01000100101",
00591      to_sfixed(0.993212, I, -F) when "01000100110",
00592      to_sfixed(0.992850, I, -F) when "01000100111",
00593      to_sfixed(0.992480, I, -F) when "01000101000",
00594      to_sfixed(0.992099, I, -F) when "01000101001",
00595      to_sfixed(0.991710, I, -F) when "01000101010",
00596      to_sfixed(0.991311, I, -F) when "01000101011",
00597      to_sfixed(0.990903, I, -F) when "01000101100",
00598      to_sfixed(0.990485, I, -F) when "01000101101",
00599      to_sfixed(0.990058, I, -F) when "01000101110",
00600      to_sfixed(0.989622, I, -F) when "01000101111",
00601      to_sfixed(0.989177, I, -F) when "01000110000",
00602      to_sfixed(0.988722, I, -F) when "01000110001",
00603      to_sfixed(0.988258, I, -F) when "01000110010",
00604      to_sfixed(0.987784, I, -F) when "01000110011",
00605      to_sfixed(0.987301, I, -F) when "01000110100",
00606      to_sfixed(0.986809, I, -F) when "01000110101",
00607      to_sfixed(0.986308, I, -F) when "01000110110",
00608      to_sfixed(0.985798, I, -F) when "01000110111",
00609      to_sfixed(0.985278, I, -F) when "01000111000",
00610      to_sfixed(0.984749, I, -F) when "01000111001",
00611      to_sfixed(0.984210, I, -F) when "01000111010",
00612      to_sfixed(0.983662, I, -F) when "01000111011",
00613      to_sfixed(0.983105, I, -F) when "01000111100",
00614      to_sfixed(0.982539, I, -F) when "01000111101",
00615      to_sfixed(0.981964, I, -F) when "01000111110",
00616      to_sfixed(0.981379, I, -F) when "01000111111",
00617      to_sfixed(0.980785, I, -F) when "01001000000",
00618      to_sfixed(0.980182, I, -F) when "01001000001",
00619      to_sfixed(0.979570, I, -F) when "01001000010",
00620      to_sfixed(0.978948, I, -F) when "01001000011",
00621      to_sfixed(0.978317, I, -F) when "01001000100",
00622      to_sfixed(0.977677, I, -F) when "01001000101",
00623      to_sfixed(0.977028, I, -F) when "01001000110",
00624      to_sfixed(0.976370, I, -F) when "01001000111",

```

```
00625      to_sfixed(0.975702, I, -F) when "01001001000",
00626      to_sfixed(0.975025, I, -F) when "01001001001",
00627      to_sfixed(0.974339, I, -F) when "01001001010",
00628      to_sfixed(0.973644, I, -F) when "01001001011",
00629      to_sfixed(0.972940, I, -F) when "01001001100",
00630      to_sfixed(0.972226, I, -F) when "01001001101",
00631      to_sfixed(0.971504, I, -F) when "01001001110",
00632      to_sfixed(0.970772, I, -F) when "01001001111",
00633      to_sfixed(0.970031, I, -F) when "01001010000",
00634      to_sfixed(0.969281, I, -F) when "01001010001",
00635      to_sfixed(0.968522, I, -F) when "01001010010",
00636      to_sfixed(0.967754, I, -F) when "01001010011",
00637      to_sfixed(0.966976, I, -F) when "01001010100",
00638      to_sfixed(0.966190, I, -F) when "01001010101",
00639      to_sfixed(0.965394, I, -F) when "01001010110",
00640      to_sfixed(0.964590, I, -F) when "01001010111",
00641      to_sfixed(0.963776, I, -F) when "01001011000",
00642      to_sfixed(0.962953, I, -F) when "01001011001",
00643      to_sfixed(0.962121, I, -F) when "01001011010",
00644      to_sfixed(0.961280, I, -F) when "01001011011",
00645      to_sfixed(0.960431, I, -F) when "01001011100",
00646      to_sfixed(0.959572, I, -F) when "01001011101",
00647      to_sfixed(0.958703, I, -F) when "01001011110",
00648      to_sfixed(0.957826, I, -F) when "01001011111",
00649      to_sfixed(0.956940, I, -F) when "01001100000",
00650      to_sfixed(0.956045, I, -F) when "01001100001",
00651      to_sfixed(0.955141, I, -F) when "01001100010",
00652      to_sfixed(0.954228, I, -F) when "01001100011",
00653      to_sfixed(0.953306, I, -F) when "01001100100",
00654      to_sfixed(0.952375, I, -F) when "01001100101",
00655      to_sfixed(0.951435, I, -F) when "01001100110",
00656      to_sfixed(0.950486, I, -F) when "01001100111",
00657      to_sfixed(0.949528, I, -F) when "01001101000",
00658      to_sfixed(0.948561, I, -F) when "01001101001",
00659      to_sfixed(0.947586, I, -F) when "01001101010",
00660      to_sfixed(0.946601, I, -F) when "01001101011",
00661      to_sfixed(0.945607, I, -F) when "01001101100",
00662      to_sfixed(0.944605, I, -F) when "01001101101",
00663      to_sfixed(0.943593, I, -F) when "01001101110",
00664      to_sfixed(0.942573, I, -F) when "01001101111",
00665      to_sfixed(0.941544, I, -F) when "01001110000",
00666      to_sfixed(0.940506, I, -F) when "01001110001",
00667      to_sfixed(0.939459, I, -F) when "01001110010",
00668      to_sfixed(0.938404, I, -F) when "01001110011",
00669      to_sfixed(0.937339, I, -F) when "01001110100",
00670      to_sfixed(0.936266, I, -F) when "01001110101",
00671      to_sfixed(0.935184, I, -F) when "01001110110",
00672      to_sfixed(0.934093, I, -F) when "01001110111",
00673      to_sfixed(0.932993, I, -F) when "01001111000",
00674      to_sfixed(0.931884, I, -F) when "01001111001",
00675      to_sfixed(0.930767, I, -F) when "01001111010",
00676      to_sfixed(0.929641, I, -F) when "01001111011",
00677      to_sfixed(0.928506, I, -F) when "01001111100",
00678      to_sfixed(0.927363, I, -F) when "01001111101",
00679      to_sfixed(0.926210, I, -F) when "01001111110",
00680      to_sfixed(0.925049, I, -F) when "01001111111",
00681      to_sfixed(0.923880, I, -F) when "01010000000",
00682      to_sfixed(0.922701, I, -F) when "01010000001",
00683      to_sfixed(0.921514, I, -F) when "01010000010",
00684      to_sfixed(0.920318, I, -F) when "01010000011",
00685      to_sfixed(0.919114, I, -F) when "01010000100",
00686      to_sfixed(0.917901, I, -F) when "01010000101",
00687      to_sfixed(0.916679, I, -F) when "01010000110",
00688      to_sfixed(0.915449, I, -F) when "01010000111",
00689      to_sfixed(0.914210, I, -F) when "01010001000",
00690      to_sfixed(0.912962, I, -F) when "01010001001",
00691      to_sfixed(0.911706, I, -F) when "01010001010",
00692      to_sfixed(0.910441, I, -F) when "01010001011",
00693      to_sfixed(0.909168, I, -F) when "01010001100",
00694      to_sfixed(0.907886, I, -F) when "01010001101",
00695      to_sfixed(0.906596, I, -F) when "01010001110",
00696      to_sfixed(0.905297, I, -F) when "01010001111",
00697      to_sfixed(0.903989, I, -F) when "01010010000",
00698      to_sfixed(0.902673, I, -F) when "01010010001",
00699      to_sfixed(0.901349, I, -F) when "01010010010",
00700      to_sfixed(0.900016, I, -F) when "01010010011",
00701      to_sfixed(0.898674, I, -F) when "01010010100",
00702      to_sfixed(0.897325, I, -F) when "01010010101",
00703      to_sfixed(0.895966, I, -F) when "01010010110",
00704      to_sfixed(0.894599, I, -F) when "01010010111",
00705      to_sfixed(0.893224, I, -F) when "01010011000",
00706      to_sfixed(0.891841, I, -F) when "01010011001",
00707      to_sfixed(0.890449, I, -F) when "01010011010",
00708      to_sfixed(0.889048, I, -F) when "01010011011",
00709      to_sfixed(0.887640, I, -F) when "01010011100",
00710      to_sfixed(0.886223, I, -F) when "01010011101",
00711      to_sfixed(0.884797, I, -F) when "01010011110",
```

```

00712      to_sfixed(0.883363, I, -F) when "01010011111",
00713      to_sfixed(0.881921, I, -F) when "01010100000",
00714      to_sfixed(0.880471, I, -F) when "01010100001",
00715      to_sfixed(0.879012, I, -F) when "01010100010",
00716      to_sfixed(0.877545, I, -F) when "01010100011",
00717      to_sfixed(0.876070, I, -F) when "01010100100",
00718      to_sfixed(0.874587, I, -F) when "01010100101",
00719      to_sfixed(0.873095, I, -F) when "01010100110",
00720      to_sfixed(0.871595, I, -F) when "01010100111",
00721      to_sfixed(0.870087, I, -F) when "01010101000",
00722      to_sfixed(0.868571, I, -F) when "01010101001",
00723      to_sfixed(0.867046, I, -F) when "01010101010",
00724      to_sfixed(0.865514, I, -F) when "01010101011",
00725      to_sfixed(0.863973, I, -F) when "01010101100",
00726      to_sfixed(0.862424, I, -F) when "01010101101",
00727      to_sfixed(0.860867, I, -F) when "01010101110",
00728      to_sfixed(0.859302, I, -F) when "01010101111",
00729      to_sfixed(0.857729, I, -F) when "01010110000",
00730      to_sfixed(0.856147, I, -F) when "01010110001",
00731      to_sfixed(0.854558, I, -F) when "01010110010",
00732      to_sfixed(0.852961, I, -F) when "01010110011",
00733      to_sfixed(0.851355, I, -F) when "01010110100",
00734      to_sfixed(0.849742, I, -F) when "01010110101",
00735      to_sfixed(0.848120, I, -F) when "01010110110",
00736      to_sfixed(0.846491, I, -F) when "01010110111",
00737      to_sfixed(0.844854, I, -F) when "01010111000",
00738      to_sfixed(0.843208, I, -F) when "01010111001",
00739      to_sfixed(0.841555, I, -F) when "01010111010",
00740      to_sfixed(0.839894, I, -F) when "01010111011",
00741      to_sfixed(0.838225, I, -F) when "01010111100",
00742      to_sfixed(0.836548, I, -F) when "01010111101",
00743      to_sfixed(0.834863, I, -F) when "01010111110",
00744      to_sfixed(0.833170, I, -F) when "01010111111",
00745      to_sfixed(0.831470, I, -F) when "01011000000",
00746      to_sfixed(0.829761, I, -F) when "01011000001",
00747      to_sfixed(0.828045, I, -F) when "01011000010",
00748      to_sfixed(0.826321, I, -F) when "01011000011",
00749      to_sfixed(0.824589, I, -F) when "01011000100",
00750      to_sfixed(0.822850, I, -F) when "01011000101",
00751      to_sfixed(0.821103, I, -F) when "01011000110",
00752      to_sfixed(0.819348, I, -F) when "01011000111",
00753      to_sfixed(0.817585, I, -F) when "01011001000",
00754      to_sfixed(0.815814, I, -F) when "01011001001",
00755      to_sfixed(0.814036, I, -F) when "01011001010",
00756      to_sfixed(0.812251, I, -F) when "01011001011",
00757      to_sfixed(0.810457, I, -F) when "01011001100",
00758      to_sfixed(0.808656, I, -F) when "01011001101",
00759      to_sfixed(0.806848, I, -F) when "01011001110",
00760      to_sfixed(0.805031, I, -F) when "01011001111",
00761      to_sfixed(0.803208, I, -F) when "01011010000",
00762      to_sfixed(0.801376, I, -F) when "01011010001",
00763      to_sfixed(0.799537, I, -F) when "01011010010",
00764      to_sfixed(0.797691, I, -F) when "01011010011",
00765      to_sfixed(0.795837, I, -F) when "01011010100",
00766      to_sfixed(0.793975, I, -F) when "01011010101",
00767      to_sfixed(0.792107, I, -F) when "01011010110",
00768      to_sfixed(0.790230, I, -F) when "01011010111",
00769      to_sfixed(0.788346, I, -F) when "01011011000",
00770      to_sfixed(0.786455, I, -F) when "01011011001",
00771      to_sfixed(0.784557, I, -F) when "01011011010",
00772      to_sfixed(0.782651, I, -F) when "01011011011",
00773      to_sfixed(0.780737, I, -F) when "01011011100",
00774      to_sfixed(0.778817, I, -F) when "01011011101",
00775      to_sfixed(0.776888, I, -F) when "01011011110",
00776      to_sfixed(0.774953, I, -F) when "01011011111",
00777      to_sfixed(0.773010, I, -F) when "01011100000",
00778      to_sfixed(0.771061, I, -F) when "01011100001",
00779      to_sfixed(0.769103, I, -F) when "01011100010",
00780      to_sfixed(0.767139, I, -F) when "01011100011",
00781      to_sfixed(0.765167, I, -F) when "01011100100",
00782      to_sfixed(0.763188, I, -F) when "01011100101",
00783      to_sfixed(0.761202, I, -F) when "01011100110",
00784      to_sfixed(0.759209, I, -F) when "01011100111",
00785      to_sfixed(0.757209, I, -F) when "01011101000",
00786      to_sfixed(0.755201, I, -F) when "01011101001",
00787      to_sfixed(0.753187, I, -F) when "01011101010",
00788      to_sfixed(0.751165, I, -F) when "01011101011",
00789      to_sfixed(0.749136, I, -F) when "01011101100",
00790      to_sfixed(0.747101, I, -F) when "01011101101",
00791      to_sfixed(0.745058, I, -F) when "01011101110",
00792      to_sfixed(0.743008, I, -F) when "01011101111",
00793      to_sfixed(0.740951, I, -F) when "01011110000",
00794      to_sfixed(0.738887, I, -F) when "01011110001",
00795      to_sfixed(0.736817, I, -F) when "01011110010",
00796      to_sfixed(0.734739, I, -F) when "01011110011",
00797      to_sfixed(0.732654, I, -F) when "01011110100",
00798      to_sfixed(0.730563, I, -F) when "01011110101",

```

```

00799      to_sfixed(0.728464, I, -F) when "01011110110",
00800      to_sfixed(0.726359, I, -F) when "01011110111",
00801      to_sfixed(0.724247, I, -F) when "01011111000",
00802      to_sfixed(0.722128, I, -F) when "01011111001",
00803      to_sfixed(0.720003, I, -F) when "01011111010",
00804      to_sfixed(0.717870, I, -F) when "01011111011",
00805      to_sfixed(0.715731, I, -F) when "01011111100",
00806      to_sfixed(0.713585, I, -F) when "01011111101",
00807      to_sfixed(0.711432, I, -F) when "01011111110",
00808      to_sfixed(0.709273, I, -F) when "01011111111",
00809      to_sfixed(0.707107, I, -F) when "01100000000",
00810      to_sfixed(0.704934, I, -F) when "01100000001",
00811      to_sfixed(0.702755, I, -F) when "01100000010",
00812      to_sfixed(0.700569, I, -F) when "01100000011",
00813      to_sfixed(0.698376, I, -F) when "01100000100",
00814      to_sfixed(0.696177, I, -F) when "01100000101",
00815      to_sfixed(0.693971, I, -F) when "01100000110",
00816      to_sfixed(0.691759, I, -F) when "01100000111",
00817      to_sfixed(0.689541, I, -F) when "01100001000",
00818      to_sfixed(0.687315, I, -F) when "01100001001",
00819      to_sfixed(0.685084, I, -F) when "01100001010",
00820      to_sfixed(0.682846, I, -F) when "01100001011",
00821      to_sfixed(0.680601, I, -F) when "01100001100",
00822      to_sfixed(0.678350, I, -F) when "01100001101",
00823      to_sfixed(0.676093, I, -F) when "01100001110",
00824      to_sfixed(0.673829, I, -F) when "01100001111",
00825      to_sfixed(0.671559, I, -F) when "01100010000",
00826      to_sfixed(0.669283, I, -F) when "01100010001",
00827      to_sfixed(0.667000, I, -F) when "01100010010",
00828      to_sfixed(0.664711, I, -F) when "01100010011",
00829      to_sfixed(0.662416, I, -F) when "01100010100",
00830      to_sfixed(0.660114, I, -F) when "01100010101",
00831      to_sfixed(0.657807, I, -F) when "01100010110",
00832      to_sfixed(0.655493, I, -F) when "01100010111",
00833      to_sfixed(0.653173, I, -F) when "01100011000",
00834      to_sfixed(0.650847, I, -F) when "01100011001",
00835      to_sfixed(0.648514, I, -F) when "01100011010",
00836      to_sfixed(0.646176, I, -F) when "01100011011",
00837      to_sfixed(0.643832, I, -F) when "01100011100",
00838      to_sfixed(0.641481, I, -F) when "01100011101",
00839      to_sfixed(0.639124, I, -F) when "01100011110",
00840      to_sfixed(0.636762, I, -F) when "01100011111",
00841      to_sfixed(0.634393, I, -F) when "01100100000",
00842      to_sfixed(0.632019, I, -F) when "01100100001",
00843      to_sfixed(0.629638, I, -F) when "01100100010",
00844      to_sfixed(0.627252, I, -F) when "01100100011",
00845      to_sfixed(0.624859, I, -F) when "01100100100",
00846      to_sfixed(0.622461, I, -F) when "01100100101",
00847      to_sfixed(0.620057, I, -F) when "01100100110",
00848      to_sfixed(0.617647, I, -F) when "01100100111",
00849      to_sfixed(0.615232, I, -F) when "01100101000",
00850      to_sfixed(0.612810, I, -F) when "01100101001",
00851      to_sfixed(0.610383, I, -F) when "01100101010",
00852      to_sfixed(0.607950, I, -F) when "01100101011",
00853      to_sfixed(0.605511, I, -F) when "01100101100",
00854      to_sfixed(0.603067, I, -F) when "01100101101",
00855      to_sfixed(0.600616, I, -F) when "01100101110",
00856      to_sfixed(0.598161, I, -F) when "01100101111",
00857      to_sfixed(0.595699, I, -F) when "01100110000",
00858      to_sfixed(0.593232, I, -F) when "01100110001",
00859      to_sfixed(0.590760, I, -F) when "01100110010",
00860      to_sfixed(0.588282, I, -F) when "01100110011",
00861      to_sfixed(0.585798, I, -F) when "01100110100",
00862      to_sfixed(0.583309, I, -F) when "01100110101",
00863      to_sfixed(0.580814, I, -F) when "01100110110",
00864      to_sfixed(0.578314, I, -F) when "01100110111",
00865      to_sfixed(0.575808, I, -F) when "01100111000",
00866      to_sfixed(0.573297, I, -F) when "01100111001",
00867      to_sfixed(0.570781, I, -F) when "01100111010",
00868      to_sfixed(0.568259, I, -F) when "01100111011",
00869      to_sfixed(0.565732, I, -F) when "01100111100",
00870      to_sfixed(0.563199, I, -F) when "01100111101",
00871      to_sfixed(0.560662, I, -F) when "01100111110",
00872      to_sfixed(0.558119, I, -F) when "01100111111",
00873      to_sfixed(0.555570, I, -F) when "01101000000",
00874      to_sfixed(0.553017, I, -F) when "01101000001",
00875      to_sfixed(0.550458, I, -F) when "01101000010",
00876      to_sfixed(0.547894, I, -F) when "01101000011",
00877      to_sfixed(0.545325, I, -F) when "01101000100",
00878      to_sfixed(0.542751, I, -F) when "01101000101",
00879      to_sfixed(0.540171, I, -F) when "01101000110",
00880      to_sfixed(0.537587, I, -F) when "01101000111",
00881      to_sfixed(0.534998, I, -F) when "01101001000",
00882      to_sfixed(0.532403, I, -F) when "01101001001",
00883      to_sfixed(0.529804, I, -F) when "01101001010",
00884      to_sfixed(0.527199, I, -F) when "01101001011",
00885      to_sfixed(0.524590, I, -F) when "01101001100",

```

```

00886      to_sfixed(0.521975, I, -F) when "01101001101",
00887      to_sfixed(0.519356, I, -F) when "01101001110",
00888      to_sfixed(0.516732, I, -F) when "01101001111",
00889      to_sfixed(0.514103, I, -F) when "01101010000",
00890      to_sfixed(0.511469, I, -F) when "01101010001",
00891      to_sfixed(0.508830, I, -F) when "01101010010",
00892      to_sfixed(0.506187, I, -F) when "01101010011",
00893      to_sfixed(0.503538, I, -F) when "01101010100",
00894      to_sfixed(0.500885, I, -F) when "01101010101",
00895      to_sfixed(0.498228, I, -F) when "01101010110",
00896      to_sfixed(0.495565, I, -F) when "01101010111",
00897      to_sfixed(0.492898, I, -F) when "01101011000",
00898      to_sfixed(0.490226, I, -F) when "01101011001",
00899      to_sfixed(0.487550, I, -F) when "01101011010",
00900      to_sfixed(0.484869, I, -F) when "01101011011",
00901      to_sfixed(0.482184, I, -F) when "01101011100",
00902      to_sfixed(0.479494, I, -F) when "01101011101",
00903      to_sfixed(0.476799, I, -F) when "01101011110",
00904      to_sfixed(0.474100, I, -F) when "01101011111",
00905      to_sfixed(0.471397, I, -F) when "01101100000",
00906      to_sfixed(0.468689, I, -F) when "01101100001",
00907      to_sfixed(0.465976, I, -F) when "01101100010",
00908      to_sfixed(0.463260, I, -F) when "01101100011",
00909      to_sfixed(0.460539, I, -F) when "01101100100",
00910      to_sfixed(0.457813, I, -F) when "01101100101",
00911      to_sfixed(0.455084, I, -F) when "01101100110",
00912      to_sfixed(0.452350, I, -F) when "01101100111",
00913      to_sfixed(0.449611, I, -F) when "01101101000",
00914      to_sfixed(0.446869, I, -F) when "01101101001",
00915      to_sfixed(0.444122, I, -F) when "01101101010",
00916      to_sfixed(0.441371, I, -F) when "01101101011",
00917      to_sfixed(0.438616, I, -F) when "01101101100",
00918      to_sfixed(0.435857, I, -F) when "01101101101",
00919      to_sfixed(0.433094, I, -F) when "01101101110",
00920      to_sfixed(0.430326, I, -F) when "01101101111",
00921      to_sfixed(0.427555, I, -F) when "01101110000",
00922      to_sfixed(0.424780, I, -F) when "01101110001",
00923      to_sfixed(0.422000, I, -F) when "01101110010",
00924      to_sfixed(0.419217, I, -F) when "01101110011",
00925      to_sfixed(0.416430, I, -F) when "01101110100",
00926      to_sfixed(0.413638, I, -F) when "01101110101",
00927      to_sfixed(0.410843, I, -F) when "01101110110",
00928      to_sfixed(0.408044, I, -F) when "01101110111",
00929      to_sfixed(0.405241, I, -F) when "01101111000",
00930      to_sfixed(0.402435, I, -F) when "01101111001",
00931      to_sfixed(0.399624, I, -F) when "01101111010",
00932      to_sfixed(0.396810, I, -F) when "01101111011",
00933      to_sfixed(0.393992, I, -F) when "01101111100",
00934      to_sfixed(0.391170, I, -F) when "01101111101",
00935      to_sfixed(0.388345, I, -F) when "01101111110",
00936      to_sfixed(0.385516, I, -F) when "01101111111",
00937      to_sfixed(0.382683, I, -F) when "01110000000",
00938      to_sfixed(0.379847, I, -F) when "01110000001",
00939      to_sfixed(0.377007, I, -F) when "01110000010",
00940      to_sfixed(0.374164, I, -F) when "01110000011",
00941      to_sfixed(0.371317, I, -F) when "01110000100",
00942      to_sfixed(0.368467, I, -F) when "01110000101",
00943      to_sfixed(0.365613, I, -F) when "01110000110",
00944      to_sfixed(0.362756, I, -F) when "01110000111",
00945      to_sfixed(0.359895, I, -F) when "01110001000",
00946      to_sfixed(0.357031, I, -F) when "01110001001",
00947      to_sfixed(0.354164, I, -F) when "01110001010",
00948      to_sfixed(0.351293, I, -F) when "01110001011",
00949      to_sfixed(0.348419, I, -F) when "01110001100",
00950      to_sfixed(0.345541, I, -F) when "01110001101",
00951      to_sfixed(0.342661, I, -F) when "01110001110",
00952      to_sfixed(0.339777, I, -F) when "01110001111",
00953      to_sfixed(0.336890, I, -F) when "01110010000",
00954      to_sfixed(0.334000, I, -F) when "01110010001",
00955      to_sfixed(0.331106, I, -F) when "01110010010",
00956      to_sfixed(0.328210, I, -F) when "01110010011",
00957      to_sfixed(0.325310, I, -F) when "01110010100",
00958      to_sfixed(0.322408, I, -F) when "01110010101",
00959      to_sfixed(0.319502, I, -F) when "01110010110",
00960      to_sfixed(0.316593, I, -F) when "01110010111",
00961      to_sfixed(0.313682, I, -F) when "01110011000",
00962      to_sfixed(0.310767, I, -F) when "01110011001",
00963      to_sfixed(0.307850, I, -F) when "01110011010",
00964      to_sfixed(0.304929, I, -F) when "01110011011",
00965      to_sfixed(0.302006, I, -F) when "01110011100",
00966      to_sfixed(0.299080, I, -F) when "01110011101",
00967      to_sfixed(0.296151, I, -F) when "01110011110",
00968      to_sfixed(0.293219, I, -F) when "01110011111",
00969      to_sfixed(0.290285, I, -F) when "01110100000",
00970      to_sfixed(0.287347, I, -F) when "01110100001",
00971      to_sfixed(0.284408, I, -F) when "01110100010",
00972      to_sfixed(0.281465, I, -F) when "01110100011",

```

```
00973      to_sfixed(0.278520, I, -F) when "01110100100",
00974      to_sfixed(0.275572, I, -F) when "01110100101",
00975      to_sfixed(0.272621, I, -F) when "01110100110",
00976      to_sfixed(0.269668, I, -F) when "01110100111",
00977      to_sfixed(0.266713, I, -F) when "01110101000",
00978      to_sfixed(0.263755, I, -F) when "01110101001",
00979      to_sfixed(0.260794, I, -F) when "01110101010",
00980      to_sfixed(0.257831, I, -F) when "01110101011",
00981      to_sfixed(0.254866, I, -F) when "01110101100",
00982      to_sfixed(0.251898, I, -F) when "01110101101",
00983      to_sfixed(0.248928, I, -F) when "01110101110",
00984      to_sfixed(0.245955, I, -F) when "01110101111",
00985      to_sfixed(0.242980, I, -F) when "01110110000",
00986      to_sfixed(0.240003, I, -F) when "01110110001",
00987      to_sfixed(0.237024, I, -F) when "01110110010",
00988      to_sfixed(0.234042, I, -F) when "01110110011",
00989      to_sfixed(0.231058, I, -F) when "01110110100",
00990      to_sfixed(0.228072, I, -F) when "01110110101",
00991      to_sfixed(0.225084, I, -F) when "01110110110",
00992      to_sfixed(0.222094, I, -F) when "01110110111",
00993      to_sfixed(0.219101, I, -F) when "01110111000",
00994      to_sfixed(0.216107, I, -F) when "01110111001",
00995      to_sfixed(0.213110, I, -F) when "01110111010",
00996      to_sfixed(0.210112, I, -F) when "01110111011",
00997      to_sfixed(0.207111, I, -F) when "01110111100",
00998      to_sfixed(0.204109, I, -F) when "01110111101",
00999      to_sfixed(0.201105, I, -F) when "01110111110",
01000      to_sfixed(0.198098, I, -F) when "01110111111",
01001      to_sfixed(0.195090, I, -F) when "01111000000",
01002      to_sfixed(0.192080, I, -F) when "01111000001",
01003      to_sfixed(0.189069, I, -F) when "01111000010",
01004      to_sfixed(0.186055, I, -F) when "01111000011",
01005      to_sfixed(0.183040, I, -F) when "011110000100",
01006      to_sfixed(0.180023, I, -F) when "01111000101",
01007      to_sfixed(0.177004, I, -F) when "01111000110",
01008      to_sfixed(0.173984, I, -F) when "01111000111",
01009      to_sfixed(0.170962, I, -F) when "01111001000",
01010      to_sfixed(0.167938, I, -F) when "01111001001",
01011      to_sfixed(0.164913, I, -F) when "01111001010",
01012      to_sfixed(0.161886, I, -F) when "01111001011",
01013      to_sfixed(0.158858, I, -F) when "01111001100",
01014      to_sfixed(0.155828, I, -F) when "01111001101",
01015      to_sfixed(0.152797, I, -F) when "01111001110",
01016      to_sfixed(0.149765, I, -F) when "01111001111",
01017      to_sfixed(0.146730, I, -F) when "01111010000",
01018      to_sfixed(0.143695, I, -F) when "01111010001",
01019      to_sfixed(0.140658, I, -F) when "01111010010",
01020      to_sfixed(0.137620, I, -F) when "01111010011",
01021      to_sfixed(0.134581, I, -F) when "01111010100",
01022      to_sfixed(0.131540, I, -F) when "01111010101",
01023      to_sfixed(0.128498, I, -F) when "01111010110",
01024      to_sfixed(0.125455, I, -F) when "01111010111",
01025      to_sfixed(0.122411, I, -F) when "01111011000",
01026      to_sfixed(0.119365, I, -F) when "01111011001",
01027      to_sfixed(0.116319, I, -F) when "01111011010",
01028      to_sfixed(0.113271, I, -F) when "01111011011",
01029      to_sfixed(0.110222, I, -F) when "01111011100",
01030      to_sfixed(0.107172, I, -F) when "01111011101",
01031      to_sfixed(0.104122, I, -F) when "01111011110",
01032      to_sfixed(0.101070, I, -F) when "01111011111",
01033      to_sfixed(0.098017, I, -F) when "01111100000",
01034      to_sfixed(0.094963, I, -F) when "01111100001",
01035      to_sfixed(0.091909, I, -F) when "01111100010",
01036      to_sfixed(0.088854, I, -F) when "01111100011",
01037      to_sfixed(0.085797, I, -F) when "01111100100",
01038      to_sfixed(0.082740, I, -F) when "01111100101",
01039      to_sfixed(0.079682, I, -F) when "01111100110",
01040      to_sfixed(0.076624, I, -F) when "01111100111",
01041      to_sfixed(0.073565, I, -F) when "01111101000",
01042      to_sfixed(0.070505, I, -F) when "01111101001",
01043      to_sfixed(0.067444, I, -F) when "01111101010",
01044      to_sfixed(0.064383, I, -F) when "01111101011",
01045      to_sfixed(0.061321, I, -F) when "01111101100",
01046      to_sfixed(0.058258, I, -F) when "01111101101",
01047      to_sfixed(0.055195, I, -F) when "01111101110",
01048      to_sfixed(0.052132, I, -F) when "01111101111",
01049      to_sfixed(0.049068, I, -F) when "01111110000",
01050      to_sfixed(0.046003, I, -F) when "01111110001",
01051      to_sfixed(0.042938, I, -F) when "01111110010",
01052      to_sfixed(0.039873, I, -F) when "01111110011",
01053      to_sfixed(0.036807, I, -F) when "01111110100",
01054      to_sfixed(0.033741, I, -F) when "01111110101",
01055      to_sfixed(0.030675, I, -F) when "01111110110",
01056      to_sfixed(0.027608, I, -F) when "01111110111",
01057      to_sfixed(0.024541, I, -F) when "01111111000",
01058      to_sfixed(0.021474, I, -F) when "01111111001",
01059      to_sfixed(0.018407, I, -F) when "01111111010",
```

```

01060      to_sfixed(0.015339, I, -F) when "01111111011",
01061      to_sfixed(0.012272, I, -F) when "01111111100",
01062      to_sfixed(0.009204, I, -F) when "01111111101",
01063      to_sfixed(0.006136, I, -F) when "01111111110",
01064      to_sfixed(0.003068, I, -F) when "01111111111",
01065      to_sfixed(0.000000, I, -F) when "10000000000",
01066      to_sfixed(-0.003068, I, -F) when "10000000001",
01067      to_sfixed(-0.006136, I, -F) when "10000000010",
01068      to_sfixed(-0.009204, I, -F) when "10000000011",
01069      to_sfixed(-0.012272, I, -F) when "10000000100",
01070      to_sfixed(-0.015339, I, -F) when "10000000101",
01071      to_sfixed(-0.018407, I, -F) when "10000000110",
01072      to_sfixed(-0.021474, I, -F) when "10000000111",
01073      to_sfixed(-0.024541, I, -F) when "100000001000",
01074      to_sfixed(-0.027608, I, -F) when "100000001001",
01075      to_sfixed(-0.030675, I, -F) when "100000001010",
01076      to_sfixed(-0.033741, I, -F) when "100000001011",
01077      to_sfixed(-0.036807, I, -F) when "100000001100",
01078      to_sfixed(-0.039873, I, -F) when "100000001101",
01079      to_sfixed(-0.042938, I, -F) when "100000001110",
01080      to_sfixed(-0.046003, I, -F) when "100000001111",
01081      to_sfixed(-0.049068, I, -F) when "10000010000",
01082      to_sfixed(-0.052132, I, -F) when "10000010001",
01083      to_sfixed(-0.055195, I, -F) when "10000010010",
01084      to_sfixed(-0.058258, I, -F) when "10000010011",
01085      to_sfixed(-0.061321, I, -F) when "10000010100",
01086      to_sfixed(-0.064383, I, -F) when "10000010101",
01087      to_sfixed(-0.067444, I, -F) when "10000010110",
01088      to_sfixed(-0.070505, I, -F) when "10000010111",
01089      to_sfixed(-0.073565, I, -F) when "10000011000",
01090      to_sfixed(-0.076624, I, -F) when "10000011001",
01091      to_sfixed(-0.079682, I, -F) when "10000011010",
01092      to_sfixed(-0.082740, I, -F) when "10000011011",
01093      to_sfixed(-0.085797, I, -F) when "10000011100",
01094      to_sfixed(-0.088854, I, -F) when "10000011101",
01095      to_sfixed(-0.091909, I, -F) when "10000011110",
01096      to_sfixed(-0.094963, I, -F) when "10000011111",
01097      to_sfixed(-0.098017, I, -F) when "10000100000",
01098      to_sfixed(-0.101070, I, -F) when "10000100001",
01099      to_sfixed(-0.104122, I, -F) when "10000100010",
01100      to_sfixed(-0.107172, I, -F) when "10000100011",
01101      to_sfixed(-0.110222, I, -F) when "10000100100",
01102      to_sfixed(-0.113271, I, -F) when "10000100101",
01103      to_sfixed(-0.116319, I, -F) when "10000100110",
01104      to_sfixed(-0.119365, I, -F) when "10000100111",
01105      to_sfixed(-0.122411, I, -F) when "10000101000",
01106      to_sfixed(-0.125455, I, -F) when "10000101001",
01107      to_sfixed(-0.128498, I, -F) when "10000101010",
01108      to_sfixed(-0.131540, I, -F) when "10000101011",
01109      to_sfixed(-0.134581, I, -F) when "10000101100",
01110      to_sfixed(-0.137620, I, -F) when "10000101101",
01111      to_sfixed(-0.140658, I, -F) when "10000101110",
01112      to_sfixed(-0.143695, I, -F) when "10000101111",
01113      to_sfixed(-0.146730, I, -F) when "10000110000",
01114      to_sfixed(-0.149765, I, -F) when "10000110001",
01115      to_sfixed(-0.152797, I, -F) when "10000110010",
01116      to_sfixed(-0.155828, I, -F) when "10000110011",
01117      to_sfixed(-0.158858, I, -F) when "10000110100",
01118      to_sfixed(-0.161886, I, -F) when "10000110101",
01119      to_sfixed(-0.164913, I, -F) when "10000110110",
01120      to_sfixed(-0.167938, I, -F) when "10000110111",
01121      to_sfixed(-0.170962, I, -F) when "10000111000",
01122      to_sfixed(-0.173984, I, -F) when "10000111001",
01123      to_sfixed(-0.177004, I, -F) when "10000111010",
01124      to_sfixed(-0.180023, I, -F) when "10000111011",
01125      to_sfixed(-0.183040, I, -F) when "10000111100",
01126      to_sfixed(-0.186055, I, -F) when "10000111101",
01127      to_sfixed(-0.189069, I, -F) when "10000111110",
01128      to_sfixed(-0.192080, I, -F) when "10000111111",
01129      to_sfixed(-0.195090, I, -F) when "10001000000",
01130      to_sfixed(-0.198098, I, -F) when "10001000001",
01131      to_sfixed(-0.201105, I, -F) when "10001000010",
01132      to_sfixed(-0.204109, I, -F) when "10001000011",
01133      to_sfixed(-0.207111, I, -F) when "10001000100",
01134      to_sfixed(-0.210112, I, -F) when "10001000101",
01135      to_sfixed(-0.213110, I, -F) when "10001000110",
01136      to_sfixed(-0.216107, I, -F) when "10001000111",
01137      to_sfixed(-0.219101, I, -F) when "10001001000",
01138      to_sfixed(-0.222094, I, -F) when "10001001001",
01139      to_sfixed(-0.225084, I, -F) when "10001001010",
01140      to_sfixed(-0.228072, I, -F) when "10001001011",
01141      to_sfixed(-0.231058, I, -F) when "10001001100",
01142      to_sfixed(-0.234042, I, -F) when "10001001101",
01143      to_sfixed(-0.237024, I, -F) when "10001001110",
01144      to_sfixed(-0.240003, I, -F) when "10001001111",
01145      to_sfixed(-0.242980, I, -F) when "10001010000",
01146      to_sfixed(-0.245955, I, -F) when "10001010001",

```

```

01147      to_sfixed(-0.248928, I, -F) when "10001010010",
01148      to_sfixed(-0.251898, I, -F) when "10001010011",
01149      to_sfixed(-0.254866, I, -F) when "10001010100",
01150      to_sfixed(-0.257831, I, -F) when "10001010101",
01151      to_sfixed(-0.260794, I, -F) when "10001010110",
01152      to_sfixed(-0.263755, I, -F) when "10001010111",
01153      to_sfixed(-0.266713, I, -F) when "10001011000",
01154      to_sfixed(-0.269668, I, -F) when "10001011001",
01155      to_sfixed(-0.272621, I, -F) when "10001011010",
01156      to_sfixed(-0.275572, I, -F) when "10001011011",
01157      to_sfixed(-0.278520, I, -F) when "10001011100",
01158      to_sfixed(-0.281465, I, -F) when "10001011101",
01159      to_sfixed(-0.284408, I, -F) when "10001011110",
01160      to_sfixed(-0.287347, I, -F) when "10001011111",
01161      to_sfixed(-0.290285, I, -F) when "10001100000",
01162      to_sfixed(-0.293219, I, -F) when "10001100001",
01163      to_sfixed(-0.296151, I, -F) when "10001100010",
01164      to_sfixed(-0.299080, I, -F) when "10001100011",
01165      to_sfixed(-0.302006, I, -F) when "10001100100",
01166      to_sfixed(-0.304929, I, -F) when "10001100101",
01167      to_sfixed(-0.307850, I, -F) when "10001100110",
01168      to_sfixed(-0.310767, I, -F) when "10001100111",
01169      to_sfixed(-0.313682, I, -F) when "10001101000",
01170      to_sfixed(-0.316593, I, -F) when "10001101001",
01171      to_sfixed(-0.319502, I, -F) when "10001101010",
01172      to_sfixed(-0.322408, I, -F) when "10001101011",
01173      to_sfixed(-0.325310, I, -F) when "10001101100",
01174      to_sfixed(-0.328210, I, -F) when "10001101101",
01175      to_sfixed(-0.331106, I, -F) when "10001101110",
01176      to_sfixed(-0.334000, I, -F) when "10001101111",
01177      to_sfixed(-0.336890, I, -F) when "10001110000",
01178      to_sfixed(-0.339777, I, -F) when "10001110001",
01179      to_sfixed(-0.342661, I, -F) when "10001110010",
01180      to_sfixed(-0.345541, I, -F) when "10001110011",
01181      to_sfixed(-0.348419, I, -F) when "10001110100",
01182      to_sfixed(-0.351293, I, -F) when "10001110101",
01183      to_sfixed(-0.354164, I, -F) when "10001110110",
01184      to_sfixed(-0.357031, I, -F) when "10001110111",
01185      to_sfixed(-0.359895, I, -F) when "10001111000",
01186      to_sfixed(-0.362756, I, -F) when "10001111001",
01187      to_sfixed(-0.365613, I, -F) when "10001111010",
01188      to_sfixed(-0.368467, I, -F) when "10001111011",
01189      to_sfixed(-0.371317, I, -F) when "10001111100",
01190      to_sfixed(-0.374164, I, -F) when "10001111101",
01191      to_sfixed(-0.377007, I, -F) when "10001111110",
01192      to_sfixed(-0.379847, I, -F) when "10001111111",
01193      to_sfixed(-0.3822683, I, -F) when "10010000000",
01194      to_sfixed(-0.385516, I, -F) when "10010000001",
01195      to_sfixed(-0.388345, I, -F) when "10010000010",
01196      to_sfixed(-0.391170, I, -F) when "10010000011",
01197      to_sfixed(-0.393992, I, -F) when "10010000100",
01198      to_sfixed(-0.396810, I, -F) when "10010000101",
01199      to_sfixed(-0.399624, I, -F) when "10010000110",
01200      to_sfixed(-0.402435, I, -F) when "10010000111",
01201      to_sfixed(-0.405241, I, -F) when "10010001000",
01202      to_sfixed(-0.408044, I, -F) when "10010001001",
01203      to_sfixed(-0.410843, I, -F) when "10010001010",
01204      to_sfixed(-0.413638, I, -F) when "10010001011",
01205      to_sfixed(-0.416430, I, -F) when "10010001100",
01206      to_sfixed(-0.419217, I, -F) when "10010001101",
01207      to_sfixed(-0.422000, I, -F) when "10010001110",
01208      to_sfixed(-0.424780, I, -F) when "10010001111",
01209      to_sfixed(-0.427555, I, -F) when "10010010000",
01210      to_sfixed(-0.430326, I, -F) when "10010010001",
01211      to_sfixed(-0.433094, I, -F) when "10010010010",
01212      to_sfixed(-0.435857, I, -F) when "10010010011",
01213      to_sfixed(-0.438616, I, -F) when "10010010100",
01214      to_sfixed(-0.441371, I, -F) when "10010010101",
01215      to_sfixed(-0.444122, I, -F) when "10010010110",
01216      to_sfixed(-0.446869, I, -F) when "10010010111",
01217      to_sfixed(-0.449611, I, -F) when "10010011000",
01218      to_sfixed(-0.452350, I, -F) when "10010011001",
01219      to_sfixed(-0.455084, I, -F) when "10010011010",
01220      to_sfixed(-0.457813, I, -F) when "10010011011",
01221      to_sfixed(-0.460539, I, -F) when "10010011100",
01222      to_sfixed(-0.463260, I, -F) when "10010011101",
01223      to_sfixed(-0.465976, I, -F) when "10010011110",
01224      to_sfixed(-0.468689, I, -F) when "10010011111",
01225      to_sfixed(-0.471397, I, -F) when "10010100000",
01226      to_sfixed(-0.474100, I, -F) when "10010100001",
01227      to_sfixed(-0.476799, I, -F) when "10010100010",
01228      to_sfixed(-0.479494, I, -F) when "10010100011",
01229      to_sfixed(-0.482184, I, -F) when "10010100100",
01230      to_sfixed(-0.484869, I, -F) when "10010100101",
01231      to_sfixed(-0.487550, I, -F) when "10010100110",
01232      to_sfixed(-0.490226, I, -F) when "10010100111",
01233      to_sfixed(-0.492898, I, -F) when "10010101000",

```

```

01234      to_sfixed(-0.495565, I, -F) when "10010101001",
01235      to_sfixed(-0.498228, I, -F) when "10010101010",
01236      to_sfixed(-0.500885, I, -F) when "10010101011",
01237      to_sfixed(-0.503538, I, -F) when "10010101100",
01238      to_sfixed(-0.506187, I, -F) when "10010101101",
01239      to_sfixed(-0.508830, I, -F) when "10010101110",
01240      to_sfixed(-0.511469, I, -F) when "10010101111",
01241      to_sfixed(-0.514103, I, -F) when "10010110000",
01242      to_sfixed(-0.516732, I, -F) when "10010110001",
01243      to_sfixed(-0.519356, I, -F) when "10010110010",
01244      to_sfixed(-0.521975, I, -F) when "10010110011",
01245      to_sfixed(-0.524590, I, -F) when "10010110100",
01246      to_sfixed(-0.527199, I, -F) when "10010110101",
01247      to_sfixed(-0.529804, I, -F) when "10010110110",
01248      to_sfixed(-0.532403, I, -F) when "10010110111",
01249      to_sfixed(-0.534998, I, -F) when "10010111000",
01250      to_sfixed(-0.537587, I, -F) when "10010111001",
01251      to_sfixed(-0.540171, I, -F) when "10010111010",
01252      to_sfixed(-0.542751, I, -F) when "10010111011",
01253      to_sfixed(-0.545325, I, -F) when "10010111100",
01254      to_sfixed(-0.547894, I, -F) when "10010111101",
01255      to_sfixed(-0.550458, I, -F) when "10010111110",
01256      to_sfixed(-0.553017, I, -F) when "10010111111",
01257      to_sfixed(-0.555570, I, -F) when "10011000000",
01258      to_sfixed(-0.558119, I, -F) when "10011000001",
01259      to_sfixed(-0.560662, I, -F) when "10011000010",
01260      to_sfixed(-0.563199, I, -F) when "10011000011",
01261      to_sfixed(-0.565732, I, -F) when "10011000100",
01262      to_sfixed(-0.568259, I, -F) when "10011000101",
01263      to_sfixed(-0.570781, I, -F) when "10011000110",
01264      to_sfixed(-0.573297, I, -F) when "10011000111",
01265      to_sfixed(-0.575808, I, -F) when "10011001000",
01266      to_sfixed(-0.578314, I, -F) when "10011001001",
01267      to_sfixed(-0.580814, I, -F) when "10011001010",
01268      to_sfixed(-0.583309, I, -F) when "10011001011",
01269      to_sfixed(-0.585798, I, -F) when "10011001100",
01270      to_sfixed(-0.588282, I, -F) when "10011001101",
01271      to_sfixed(-0.590760, I, -F) when "10011001110",
01272      to_sfixed(-0.593232, I, -F) when "10011001111",
01273      to_sfixed(-0.595699, I, -F) when "10011010000",
01274      to_sfixed(-0.598161, I, -F) when "10011010001",
01275      to_sfixed(-0.600616, I, -F) when "10011010010",
01276      to_sfixed(-0.603067, I, -F) when "10011010011",
01277      to_sfixed(-0.605511, I, -F) when "10011010100",
01278      to_sfixed(-0.607950, I, -F) when "10011010101",
01279      to_sfixed(-0.610383, I, -F) when "10011010110",
01280      to_sfixed(-0.612810, I, -F) when "10011010111",
01281      to_sfixed(-0.615232, I, -F) when "10011011000",
01282      to_sfixed(-0.617647, I, -F) when "10011011001",
01283      to_sfixed(-0.620057, I, -F) when "10011011010",
01284      to_sfixed(-0.622461, I, -F) when "10011011011",
01285      to_sfixed(-0.624859, I, -F) when "10011011100",
01286      to_sfixed(-0.627252, I, -F) when "10011011101",
01287      to_sfixed(-0.629638, I, -F) when "10011011110",
01288      to_sfixed(-0.632019, I, -F) when "10011011111",
01289      to_sfixed(-0.634493, I, -F) when "10011100000",
01290      to_sfixed(-0.636762, I, -F) when "10011100001",
01291      to_sfixed(-0.639124, I, -F) when "10011100010",
01292      to_sfixed(-0.641481, I, -F) when "10011100011",
01293      to_sfixed(-0.643832, I, -F) when "10011100100",
01294      to_sfixed(-0.646176, I, -F) when "10011100101",
01295      to_sfixed(-0.648514, I, -F) when "10011100110",
01296      to_sfixed(-0.650847, I, -F) when "10011100111",
01297      to_sfixed(-0.653173, I, -F) when "10011101000",
01298      to_sfixed(-0.655493, I, -F) when "10011101001",
01299      to_sfixed(-0.657807, I, -F) when "10011101010",
01300      to_sfixed(-0.660114, I, -F) when "10011101011",
01301      to_sfixed(-0.662416, I, -F) when "10011101100",
01302      to_sfixed(-0.664711, I, -F) when "10011101101",
01303      to_sfixed(-0.667000, I, -F) when "10011101110",
01304      to_sfixed(-0.669283, I, -F) when "10011101111",
01305      to_sfixed(-0.671559, I, -F) when "10011110000",
01306      to_sfixed(-0.673829, I, -F) when "10011110001",
01307      to_sfixed(-0.676093, I, -F) when "10011110010",
01308      to_sfixed(-0.678350, I, -F) when "10011110011",
01309      to_sfixed(-0.680601, I, -F) when "10011110100",
01310      to_sfixed(-0.682846, I, -F) when "10011110101",
01311      to_sfixed(-0.685084, I, -F) when "10011110110",
01312      to_sfixed(-0.687315, I, -F) when "10011110111",
01313      to_sfixed(-0.689541, I, -F) when "10011111000",
01314      to_sfixed(-0.691759, I, -F) when "10011111001",
01315      to_sfixed(-0.693971, I, -F) when "10011111010",
01316      to_sfixed(-0.696177, I, -F) when "10011111011",
01317      to_sfixed(-0.698376, I, -F) when "10011111100",
01318      to_sfixed(-0.700569, I, -F) when "10011111101",
01319      to_sfixed(-0.702755, I, -F) when "10011111110",
01320      to_sfixed(-0.704934, I, -F) when "10011111111",

```

```

01321      to_sfixed(-0.707107, I, -F) when "10100000000",
01322      to_sfixed(-0.709273, I, -F) when "10100000001",
01323      to_sfixed(-0.711432, I, -F) when "10100000010",
01324      to_sfixed(-0.713585, I, -F) when "10100000011",
01325      to_sfixed(-0.715731, I, -F) when "10100000100",
01326      to_sfixed(-0.717870, I, -F) when "10100000101",
01327      to_sfixed(-0.720003, I, -F) when "10100000110",
01328      to_sfixed(-0.722128, I, -F) when "10100000111",
01329      to_sfixed(-0.724247, I, -F) when "10100001000",
01330      to_sfixed(-0.726359, I, -F) when "10100001001",
01331      to_sfixed(-0.728464, I, -F) when "10100001010",
01332      to_sfixed(-0.730563, I, -F) when "10100001011",
01333      to_sfixed(-0.732654, I, -F) when "10100001100",
01334      to_sfixed(-0.734739, I, -F) when "10100001101",
01335      to_sfixed(-0.736817, I, -F) when "10100001110",
01336      to_sfixed(-0.738887, I, -F) when "10100001111",
01337      to_sfixed(-0.740951, I, -F) when "10100010000",
01338      to_sfixed(-0.743008, I, -F) when "10100010001",
01339      to_sfixed(-0.745058, I, -F) when "10100010010",
01340      to_sfixed(-0.747101, I, -F) when "10100010011",
01341      to_sfixed(-0.749136, I, -F) when "10100010100",
01342      to_sfixed(-0.751165, I, -F) when "10100010101",
01343      to_sfixed(-0.753187, I, -F) when "10100010110",
01344      to_sfixed(-0.755201, I, -F) when "10100010111",
01345      to_sfixed(-0.757209, I, -F) when "10100011000",
01346      to_sfixed(-0.759209, I, -F) when "10100011001",
01347      to_sfixed(-0.761202, I, -F) when "10100011010",
01348      to_sfixed(-0.763188, I, -F) when "10100011011",
01349      to_sfixed(-0.765167, I, -F) when "10100011100",
01350      to_sfixed(-0.767139, I, -F) when "10100011101",
01351      to_sfixed(-0.769103, I, -F) when "10100011110",
01352      to_sfixed(-0.771061, I, -F) when "10100011111",
01353      to_sfixed(-0.773010, I, -F) when "10100100000",
01354      to_sfixed(-0.774953, I, -F) when "10100100001",
01355      to_sfixed(-0.776888, I, -F) when "10100100010",
01356      to_sfixed(-0.778817, I, -F) when "10100100011",
01357      to_sfixed(-0.780737, I, -F) when "10100100100",
01358      to_sfixed(-0.782651, I, -F) when "10100100101",
01359      to_sfixed(-0.784557, I, -F) when "10100100110",
01360      to_sfixed(-0.786455, I, -F) when "10100100111",
01361      to_sfixed(-0.788346, I, -F) when "10100101000",
01362      to_sfixed(-0.790230, I, -F) when "10100101001",
01363      to_sfixed(-0.792107, I, -F) when "10100101010",
01364      to_sfixed(-0.793975, I, -F) when "10100101011",
01365      to_sfixed(-0.795837, I, -F) when "10100101100",
01366      to_sfixed(-0.797691, I, -F) when "10100101101",
01367      to_sfixed(-0.799537, I, -F) when "10100101110",
01368      to_sfixed(-0.801376, I, -F) when "10100101111",
01369      to_sfixed(-0.803208, I, -F) when "10100110000",
01370      to_sfixed(-0.805031, I, -F) when "10100110001",
01371      to_sfixed(-0.806848, I, -F) when "10100110010",
01372      to_sfixed(-0.808656, I, -F) when "10100110011",
01373      to_sfixed(-0.810457, I, -F) when "10100110100",
01374      to_sfixed(-0.812251, I, -F) when "10100110101",
01375      to_sfixed(-0.814036, I, -F) when "10100110110",
01376      to_sfixed(-0.815814, I, -F) when "10100110111",
01377      to_sfixed(-0.817585, I, -F) when "10100111000",
01378      to_sfixed(-0.819348, I, -F) when "10100111001",
01379      to_sfixed(-0.821103, I, -F) when "10100111010",
01380      to_sfixed(-0.822850, I, -F) when "10100111011",
01381      to_sfixed(-0.824589, I, -F) when "10100111100",
01382      to_sfixed(-0.826321, I, -F) when "10100111101",
01383      to_sfixed(-0.828045, I, -F) when "10100111110",
01384      to_sfixed(-0.829761, I, -F) when "10100111111",
01385      to_sfixed(-0.831470, I, -F) when "10101000000",
01386      to_sfixed(-0.833170, I, -F) when "10101000001",
01387      to_sfixed(-0.834863, I, -F) when "10101000010",
01388      to_sfixed(-0.836548, I, -F) when "10101000011",
01389      to_sfixed(-0.838225, I, -F) when "10101000100",
01390      to_sfixed(-0.839894, I, -F) when "10101000101",
01391      to_sfixed(-0.841555, I, -F) when "10101000110",
01392      to_sfixed(-0.843208, I, -F) when "10101000111",
01393      to_sfixed(-0.844854, I, -F) when "10101000100",
01394      to_sfixed(-0.846491, I, -F) when "10101000101",
01395      to_sfixed(-0.848120, I, -F) when "10101000102",
01396      to_sfixed(-0.849742, I, -F) when "10101001011",
01397      to_sfixed(-0.851355, I, -F) when "10101001100",
01398      to_sfixed(-0.852961, I, -F) when "10101001101",
01399      to_sfixed(-0.854558, I, -F) when "10101001110",
01400      to_sfixed(-0.856147, I, -F) when "10101001111",
01401      to_sfixed(-0.857729, I, -F) when "10101010000",
01402      to_sfixed(-0.859302, I, -F) when "10101010001",
01403      to_sfixed(-0.860867, I, -F) when "10101010010",
01404      to_sfixed(-0.862424, I, -F) when "10101010011",
01405      to_sfixed(-0.863973, I, -F) when "10101010100",
01406      to_sfixed(-0.865514, I, -F) when "10101010101",
01407      to_sfixed(-0.867046, I, -F) when "10101010110",

```

```

01408      to_sfixed(-0.868571, I, -F) when "10101010111",
01409      to_sfixed(-0.870087, I, -F) when "10101011000",
01410      to_sfixed(-0.871595, I, -F) when "10101011001",
01411      to_sfixed(-0.873095, I, -F) when "10101011010",
01412      to_sfixed(-0.874587, I, -F) when "10101011011",
01413      to_sfixed(-0.876070, I, -F) when "10101011100",
01414      to_sfixed(-0.877545, I, -F) when "10101011101",
01415      to_sfixed(-0.879012, I, -F) when "10101011110",
01416      to_sfixed(-0.880471, I, -F) when "10101011111",
01417      to_sfixed(-0.881921, I, -F) when "10101100000",
01418      to_sfixed(-0.883363, I, -F) when "10101100001",
01419      to_sfixed(-0.884797, I, -F) when "10101100010",
01420      to_sfixed(-0.886223, I, -F) when "10101100011",
01421      to_sfixed(-0.887640, I, -F) when "10101100100",
01422      to_sfixed(-0.889048, I, -F) when "10101100101",
01423      to_sfixed(-0.890449, I, -F) when "10101100110",
01424      to_sfixed(-0.891841, I, -F) when "10101100111",
01425      to_sfixed(-0.893224, I, -F) when "10101101000",
01426      to_sfixed(-0.894599, I, -F) when "10101101001",
01427      to_sfixed(-0.895966, I, -F) when "10101101010",
01428      to_sfixed(-0.897325, I, -F) when "10101101011",
01429      to_sfixed(-0.898674, I, -F) when "10101101100",
01430      to_sfixed(-0.900016, I, -F) when "10101101101",
01431      to_sfixed(-0.901349, I, -F) when "10101101110",
01432      to_sfixed(-0.902673, I, -F) when "10101101111",
01433      to_sfixed(-0.903989, I, -F) when "10101110000",
01434      to_sfixed(-0.905297, I, -F) when "10101110001",
01435      to_sfixed(-0.906596, I, -F) when "10101110010",
01436      to_sfixed(-0.907886, I, -F) when "10101110011",
01437      to_sfixed(-0.909168, I, -F) when "10101110100",
01438      to_sfixed(-0.910441, I, -F) when "10101110101",
01439      to_sfixed(-0.911706, I, -F) when "10101110110",
01440      to_sfixed(-0.912962, I, -F) when "10101110111",
01441      to_sfixed(-0.914210, I, -F) when "10101111000",
01442      to_sfixed(-0.915449, I, -F) when "10101111001",
01443      to_sfixed(-0.916679, I, -F) when "10101111010",
01444      to_sfixed(-0.917901, I, -F) when "10101111011",
01445      to_sfixed(-0.919114, I, -F) when "10101111100",
01446      to_sfixed(-0.920318, I, -F) when "10101111101",
01447      to_sfixed(-0.921514, I, -F) when "10101111110",
01448      to_sfixed(-0.922701, I, -F) when "10101111111",
01449      to_sfixed(-0.923880, I, -F) when "10110000000",
01450      to_sfixed(-0.925049, I, -F) when "10110000001",
01451      to_sfixed(-0.926210, I, -F) when "10110000010",
01452      to_sfixed(-0.927363, I, -F) when "10110000011",
01453      to_sfixed(-0.928506, I, -F) when "10110000100",
01454      to_sfixed(-0.929641, I, -F) when "10110000101",
01455      to_sfixed(-0.930767, I, -F) when "10110000110",
01456      to_sfixed(-0.931884, I, -F) when "10110000111",
01457      to_sfixed(-0.932993, I, -F) when "10110001000",
01458      to_sfixed(-0.934093, I, -F) when "10110001001",
01459      to_sfixed(-0.935184, I, -F) when "10110001010",
01460      to_sfixed(-0.936266, I, -F) when "10110001011",
01461      to_sfixed(-0.937339, I, -F) when "10110001100",
01462      to_sfixed(-0.938404, I, -F) when "10110001101",
01463      to_sfixed(-0.939459, I, -F) when "10110001110",
01464      to_sfixed(-0.940506, I, -F) when "10110001111",
01465      to_sfixed(-0.941544, I, -F) when "10110010000",
01466      to_sfixed(-0.942573, I, -F) when "10110010001",
01467      to_sfixed(-0.943593, I, -F) when "10110010010",
01468      to_sfixed(-0.944605, I, -F) when "10110010011",
01469      to_sfixed(-0.945607, I, -F) when "10110010100",
01470      to_sfixed(-0.946601, I, -F) when "10110010101",
01471      to_sfixed(-0.947586, I, -F) when "10110010110",
01472      to_sfixed(-0.948561, I, -F) when "10110010111",
01473      to_sfixed(-0.949528, I, -F) when "10110011000",
01474      to_sfixed(-0.950486, I, -F) when "10110011001",
01475      to_sfixed(-0.951435, I, -F) when "10110011010",
01476      to_sfixed(-0.952375, I, -F) when "10110011011",
01477      to_sfixed(-0.953306, I, -F) when "10110011100",
01478      to_sfixed(-0.954228, I, -F) when "10110011101",
01479      to_sfixed(-0.955141, I, -F) when "10110011110",
01480      to_sfixed(-0.956045, I, -F) when "10110011111",
01481      to_sfixed(-0.956940, I, -F) when "10110100000",
01482      to_sfixed(-0.957826, I, -F) when "10110100001",
01483      to_sfixed(-0.958703, I, -F) when "10110100010",
01484      to_sfixed(-0.959572, I, -F) when "10110100011",
01485      to_sfixed(-0.960431, I, -F) when "10110100100",
01486      to_sfixed(-0.961280, I, -F) when "10110100101",
01487      to_sfixed(-0.962121, I, -F) when "10110100110",
01488      to_sfixed(-0.962953, I, -F) when "10110100111",
01489      to_sfixed(-0.963776, I, -F) when "10110101000",
01490      to_sfixed(-0.964590, I, -F) when "10110101001",
01491      to_sfixed(-0.965394, I, -F) when "10110101010",
01492      to_sfixed(-0.966190, I, -F) when "10110101011",
01493      to_sfixed(-0.966976, I, -F) when "10110101100",
01494      to_sfixed(-0.967754, I, -F) when "10110101101",

```

```

01495      to_sfixed(-0.968522, I, -F) when "10110101110",
01496      to_sfixed(-0.969281, I, -F) when "10110101111",
01497      to_sfixed(-0.970031, I, -F) when "10110110000",
01498      to_sfixed(-0.970772, I, -F) when "10110110001",
01499      to_sfixed(-0.971504, I, -F) when "10110110010",
01500      to_sfixed(-0.972226, I, -F) when "10110110011",
01501      to_sfixed(-0.972940, I, -F) when "10110110100",
01502      to_sfixed(-0.973644, I, -F) when "10110110101",
01503      to_sfixed(-0.974339, I, -F) when "10110110110",
01504      to_sfixed(-0.975025, I, -F) when "10110110111",
01505      to_sfixed(-0.975702, I, -F) when "10110110000",
01506      to_sfixed(-0.976370, I, -F) when "10110111001",
01507      to_sfixed(-0.977028, I, -F) when "10110111101",
01508      to_sfixed(-0.977677, I, -F) when "101101111011",
01509      to_sfixed(-0.978317, I, -F) when "10110111100",
01510      to_sfixed(-0.978948, I, -F) when "10110111101",
01511      to_sfixed(-0.979570, I, -F) when "10110111110",
01512      to_sfixed(-0.980182, I, -F) when "10110111111",
01513      to_sfixed(-0.980785, I, -F) when "10111000000",
01514      to_sfixed(-0.981379, I, -F) when "10111000001",
01515      to_sfixed(-0.981964, I, -F) when "10111000010",
01516      to_sfixed(-0.982539, I, -F) when "10111000011",
01517      to_sfixed(-0.983105, I, -F) when "10111000100",
01518      to_sfixed(-0.983662, I, -F) when "10111000101",
01519      to_sfixed(-0.984210, I, -F) when "10111000110",
01520      to_sfixed(-0.984749, I, -F) when "10111000111",
01521      to_sfixed(-0.985278, I, -F) when "101110001000",
01522      to_sfixed(-0.985798, I, -F) when "10111001001",
01523      to_sfixed(-0.986308, I, -F) when "10111001010",
01524      to_sfixed(-0.986809, I, -F) when "10111001011",
01525      to_sfixed(-0.987301, I, -F) when "10111001100",
01526      to_sfixed(-0.987784, I, -F) when "10111001101",
01527      to_sfixed(-0.988258, I, -F) when "10111001110",
01528      to_sfixed(-0.988722, I, -F) when "10111001111",
01529      to_sfixed(-0.989177, I, -F) when "10111010000",
01530      to_sfixed(-0.989622, I, -F) when "10111010001",
01531      to_sfixed(-0.990058, I, -F) when "10111010010",
01532      to_sfixed(-0.990485, I, -F) when "10111010011",
01533      to_sfixed(-0.990903, I, -F) when "10111010100",
01534      to_sfixed(-0.991311, I, -F) when "10111010101",
01535      to_sfixed(-0.991710, I, -F) when "10111010110",
01536      to_sfixed(-0.992099, I, -F) when "10111010111",
01537      to_sfixed(-0.992480, I, -F) when "10111011000",
01538      to_sfixed(-0.992850, I, -F) when "10111011001",
01539      to_sfixed(-0.993212, I, -F) when "10111011010",
01540      to_sfixed(-0.993564, I, -F) when "10111011011",
01541      to_sfixed(-0.993907, I, -F) when "10111011100",
01542      to_sfixed(-0.994240, I, -F) when "10111011101",
01543      to_sfixed(-0.994565, I, -F) when "10111011110",
01544      to_sfixed(-0.994879, I, -F) when "10111011111",
01545      to_sfixed(-0.995185, I, -F) when "10111100000",
01546      to_sfixed(-0.995481, I, -F) when "10111100001",
01547      to_sfixed(-0.995767, I, -F) when "10111100010",
01548      to_sfixed(-0.996045, I, -F) when "10111100011",
01549      to_sfixed(-0.996313, I, -F) when "10111100100",
01550      to_sfixed(-0.996571, I, -F) when "10111100101",
01551      to_sfixed(-0.996820, I, -F) when "10111100110",
01552      to_sfixed(-0.997060, I, -F) when "10111100111",
01553      to_sfixed(-0.997290, I, -F) when "10111101000",
01554      to_sfixed(-0.997511, I, -F) when "10111101001",
01555      to_sfixed(-0.997723, I, -F) when "10111101010",
01556      to_sfixed(-0.997925, I, -F) when "10111101011",
01557      to_sfixed(-0.998118, I, -F) when "10111101100",
01558      to_sfixed(-0.998302, I, -F) when "10111101101",
01559      to_sfixed(-0.998476, I, -F) when "10111101110",
01560      to_sfixed(-0.998640, I, -F) when "10111101111",
01561      to_sfixed(-0.998795, I, -F) when "10111110000",
01562      to_sfixed(-0.998941, I, -F) when "10111110001",
01563      to_sfixed(-0.999078, I, -F) when "10111110010",
01564      to_sfixed(-0.999205, I, -F) when "10111110011",
01565      to_sfixed(-0.999322, I, -F) when "10111110100",
01566      to_sfixed(-0.999431, I, -F) when "10111110101",
01567      to_sfixed(-0.999529, I, -F) when "10111110110",
01568      to_sfixed(-0.999619, I, -F) when "10111110111",
01569      to_sfixed(-0.999699, I, -F) when "10111111000",
01570      to_sfixed(-0.999769, I, -F) when "10111111001",
01571      to_sfixed(-0.999831, I, -F) when "10111111101",
01572      to_sfixed(-0.999882, I, -F) when "101111111011",
01573      to_sfixed(-0.999925, I, -F) when "101111111100",
01574      to_sfixed(-0.999958, I, -F) when "101111111101",
01575      to_sfixed(-0.999981, I, -F) when "101111111110",
01576      to_sfixed(-0.999995, I, -F) when "101111111111",
01577      to_sfixed(-1.000000, I, -F) when "11000000000",
01578      to_sfixed(-0.999995, I, -F) when "110000000001",
01579      to_sfixed(-0.999981, I, -F) when "110000000010",
01580      to_sfixed(-0.999958, I, -F) when "110000000011",
01581      to_sfixed(-0.999925, I, -F) when "11000000100",

```

```

01582      to_sfixed(-0.999882, I, -F) when "11000000101",
01583      to_sfixed(-0.999831, I, -F) when "11000000110",
01584      to_sfixed(-0.999769, I, -F) when "11000000111",
01585      to_sfixed(-0.999699, I, -F) when "110000001000",
01586      to_sfixed(-0.999619, I, -F) when "110000001001",
01587      to_sfixed(-0.999529, I, -F) when "110000001010",
01588      to_sfixed(-0.999431, I, -F) when "110000001011",
01589      to_sfixed(-0.999322, I, -F) when "110000001100",
01590      to_sfixed(-0.999205, I, -F) when "110000001101",
01591      to_sfixed(-0.999078, I, -F) when "110000001110",
01592      to_sfixed(-0.998941, I, -F) when "110000001111",
01593      to_sfixed(-0.998795, I, -F) when "110000001000",
01594      to_sfixed(-0.998640, I, -F) when "110000001001",
01595      to_sfixed(-0.998476, I, -F) when "1100000010010",
01596      to_sfixed(-0.998302, I, -F) when "1100000010011",
01597      to_sfixed(-0.998118, I, -F) when "1100000010100",
01598      to_sfixed(-0.997925, I, -F) when "1100000010101",
01599      to_sfixed(-0.997723, I, -F) when "1100000010110",
01600      to_sfixed(-0.997511, I, -F) when "1100000010111",
01601      to_sfixed(-0.997290, I, -F) when "1100000011000",
01602      to_sfixed(-0.997060, I, -F) when "1100000011001",
01603      to_sfixed(-0.996820, I, -F) when "1100000011010",
01604      to_sfixed(-0.996571, I, -F) when "1100000011011",
01605      to_sfixed(-0.996313, I, -F) when "1100000011100",
01606      to_sfixed(-0.996045, I, -F) when "1100000011101",
01607      to_sfixed(-0.995767, I, -F) when "1100000011110",
01608      to_sfixed(-0.995481, I, -F) when "1100000011111",
01609      to_sfixed(-0.995185, I, -F) when "11000100000",
01610      to_sfixed(-0.994879, I, -F) when "11000100001",
01611      to_sfixed(-0.994565, I, -F) when "11000100010",
01612      to_sfixed(-0.994240, I, -F) when "11000100011",
01613      to_sfixed(-0.993907, I, -F) when "11000100100",
01614      to_sfixed(-0.993564, I, -F) when "11000100101",
01615      to_sfixed(-0.993212, I, -F) when "11000100110",
01616      to_sfixed(-0.992850, I, -F) when "11000100111",
01617      to_sfixed(-0.992480, I, -F) when "11000101000",
01618      to_sfixed(-0.992099, I, -F) when "11000101001",
01619      to_sfixed(-0.991710, I, -F) when "11000101010",
01620      to_sfixed(-0.991311, I, -F) when "11000101011",
01621      to_sfixed(-0.990903, I, -F) when "11000101100",
01622      to_sfixed(-0.990485, I, -F) when "11000101101",
01623      to_sfixed(-0.990058, I, -F) when "11000101110",
01624      to_sfixed(-0.989622, I, -F) when "11000101111",
01625      to_sfixed(-0.989177, I, -F) when "11000110000",
01626      to_sfixed(-0.988722, I, -F) when "11000110001",
01627      to_sfixed(-0.988258, I, -F) when "11000110010",
01628      to_sfixed(-0.987784, I, -F) when "11000110011",
01629      to_sfixed(-0.987301, I, -F) when "11000110100",
01630      to_sfixed(-0.986809, I, -F) when "11000110101",
01631      to_sfixed(-0.986308, I, -F) when "11000110110",
01632      to_sfixed(-0.985798, I, -F) when "11000110111",
01633      to_sfixed(-0.985278, I, -F) when "11000111000",
01634      to_sfixed(-0.984749, I, -F) when "11000111001",
01635      to_sfixed(-0.984210, I, -F) when "11000111010",
01636      to_sfixed(-0.983662, I, -F) when "11000111011",
01637      to_sfixed(-0.983105, I, -F) when "11000111100",
01638      to_sfixed(-0.982539, I, -F) when "11000111101",
01639      to_sfixed(-0.981964, I, -F) when "11000111110",
01640      to_sfixed(-0.981379, I, -F) when "11000111111",
01641      to_sfixed(-0.980785, I, -F) when "11001000000",
01642      to_sfixed(-0.980182, I, -F) when "11001000001",
01643      to_sfixed(-0.979570, I, -F) when "11001000010",
01644      to_sfixed(-0.978948, I, -F) when "11001000011",
01645      to_sfixed(-0.978317, I, -F) when "11001000100",
01646      to_sfixed(-0.977677, I, -F) when "11001000101",
01647      to_sfixed(-0.977028, I, -F) when "11001000110",
01648      to_sfixed(-0.976370, I, -F) when "11001000111",
01649      to_sfixed(-0.975702, I, -F) when "11001000100",
01650      to_sfixed(-0.975025, I, -F) when "11001001001",
01651      to_sfixed(-0.974339, I, -F) when "11001001010",
01652      to_sfixed(-0.973644, I, -F) when "11001001011",
01653      to_sfixed(-0.972940, I, -F) when "11001001100",
01654      to_sfixed(-0.972226, I, -F) when "11001001101",
01655      to_sfixed(-0.971504, I, -F) when "11001001110",
01656      to_sfixed(-0.970772, I, -F) when "11001001111",
01657      to_sfixed(-0.970031, I, -F) when "11001010000",
01658      to_sfixed(-0.969281, I, -F) when "11001010001",
01659      to_sfixed(-0.968522, I, -F) when "11001010010",
01660      to_sfixed(-0.967754, I, -F) when "11001010011",
01661      to_sfixed(-0.966976, I, -F) when "11001010100",
01662      to_sfixed(-0.966190, I, -F) when "11001010101",
01663      to_sfixed(-0.965394, I, -F) when "11001010110",
01664      to_sfixed(-0.964590, I, -F) when "11001010111",
01665      to_sfixed(-0.963776, I, -F) when "11001011000",
01666      to_sfixed(-0.962953, I, -F) when "11001011001",
01667      to_sfixed(-0.962121, I, -F) when "11001011010",
01668      to_sfixed(-0.961280, I, -F) when "11001011011",

```

```

01669      to_sfixed(-0.960431, I, -F) when "11001011100",
01670      to_sfixed(-0.959572, I, -F) when "11001011101",
01671      to_sfixed(-0.958703, I, -F) when "11001011110",
01672      to_sfixed(-0.957826, I, -F) when "11001011111",
01673      to_sfixed(-0.956940, I, -F) when "11001100000",
01674      to_sfixed(-0.956045, I, -F) when "11001100001",
01675      to_sfixed(-0.955141, I, -F) when "11001100010",
01676      to_sfixed(-0.954228, I, -F) when "11001100011",
01677      to_sfixed(-0.953306, I, -F) when "11001100100",
01678      to_sfixed(-0.952375, I, -F) when "11001100101",
01679      to_sfixed(-0.951435, I, -F) when "11001100110",
01680      to_sfixed(-0.950486, I, -F) when "11001100111",
01681      to_sfixed(-0.949528, I, -F) when "11001101000",
01682      to_sfixed(-0.948561, I, -F) when "11001101001",
01683      to_sfixed(-0.947586, I, -F) when "11001101010",
01684      to_sfixed(-0.946601, I, -F) when "11001101011",
01685      to_sfixed(-0.945607, I, -F) when "11001101100",
01686      to_sfixed(-0.944605, I, -F) when "11001101101",
01687      to_sfixed(-0.943593, I, -F) when "11001101110",
01688      to_sfixed(-0.942573, I, -F) when "11001101111",
01689      to_sfixed(-0.941544, I, -F) when "11001110000",
01690      to_sfixed(-0.940506, I, -F) when "11001110001",
01691      to_sfixed(-0.939459, I, -F) when "11001110010",
01692      to_sfixed(-0.938404, I, -F) when "11001110011",
01693      to_sfixed(-0.937339, I, -F) when "11001110100",
01694      to_sfixed(-0.936266, I, -F) when "11001110101",
01695      to_sfixed(-0.935184, I, -F) when "11001110110",
01696      to_sfixed(-0.934093, I, -F) when "11001110111",
01697      to_sfixed(-0.932993, I, -F) when "11001111000",
01698      to_sfixed(-0.931884, I, -F) when "11001111001",
01699      to_sfixed(-0.930767, I, -F) when "11001111010",
01700      to_sfixed(-0.929641, I, -F) when "11001111011",
01701      to_sfixed(-0.928506, I, -F) when "11001111100",
01702      to_sfixed(-0.927363, I, -F) when "11001111101",
01703      to_sfixed(-0.926210, I, -F) when "11001111110",
01704      to_sfixed(-0.925049, I, -F) when "11001111111",
01705      to_sfixed(-0.923880, I, -F) when "11010000000",
01706      to_sfixed(-0.922701, I, -F) when "11010000001",
01707      to_sfixed(-0.921514, I, -F) when "11010000010",
01708      to_sfixed(-0.920318, I, -F) when "11010000011",
01709      to_sfixed(-0.919114, I, -F) when "11010000100",
01710      to_sfixed(-0.917901, I, -F) when "11010000101",
01711      to_sfixed(-0.916679, I, -F) when "11010000110",
01712      to_sfixed(-0.915449, I, -F) when "11010000111",
01713      to_sfixed(-0.914210, I, -F) when "11010001000",
01714      to_sfixed(-0.912962, I, -F) when "11010001001",
01715      to_sfixed(-0.911706, I, -F) when "11010001010",
01716      to_sfixed(-0.910441, I, -F) when "11010001011",
01717      to_sfixed(-0.909168, I, -F) when "11010001100",
01718      to_sfixed(-0.907886, I, -F) when "11010001101",
01719      to_sfixed(-0.906596, I, -F) when "11010001110",
01720      to_sfixed(-0.905297, I, -F) when "11010001111",
01721      to_sfixed(-0.903989, I, -F) when "11010010000",
01722      to_sfixed(-0.902673, I, -F) when "11010010001",
01723      to_sfixed(-0.901349, I, -F) when "11010010010",
01724      to_sfixed(-0.900016, I, -F) when "11010010011",
01725      to_sfixed(-0.898674, I, -F) when "11010010100",
01726      to_sfixed(-0.897325, I, -F) when "11010010101",
01727      to_sfixed(-0.895966, I, -F) when "11010010110",
01728      to_sfixed(-0.894599, I, -F) when "11010010111",
01729      to_sfixed(-0.893224, I, -F) when "11010011000",
01730      to_sfixed(-0.891841, I, -F) when "11010011001",
01731      to_sfixed(-0.890449, I, -F) when "11010011010",
01732      to_sfixed(-0.889048, I, -F) when "11010011011",
01733      to_sfixed(-0.887640, I, -F) when "11010011100",
01734      to_sfixed(-0.886223, I, -F) when "11010011101",
01735      to_sfixed(-0.884797, I, -F) when "11010011110",
01736      to_sfixed(-0.883363, I, -F) when "11010011111",
01737      to_sfixed(-0.881921, I, -F) when "11010100000",
01738      to_sfixed(-0.880471, I, -F) when "11010100001",
01739      to_sfixed(-0.879012, I, -F) when "11010100010",
01740      to_sfixed(-0.877545, I, -F) when "11010100011",
01741      to_sfixed(-0.876070, I, -F) when "11010100100",
01742      to_sfixed(-0.874587, I, -F) when "11010100101",
01743      to_sfixed(-0.873095, I, -F) when "11010100110",
01744      to_sfixed(-0.871595, I, -F) when "11010100111",
01745      to_sfixed(-0.870087, I, -F) when "11010101000",
01746      to_sfixed(-0.868571, I, -F) when "11010101001",
01747      to_sfixed(-0.867046, I, -F) when "11010101010",
01748      to_sfixed(-0.865514, I, -F) when "11010101011",
01749      to_sfixed(-0.863973, I, -F) when "11010101100",
01750      to_sfixed(-0.862424, I, -F) when "11010101101",
01751      to_sfixed(-0.860867, I, -F) when "11010101110",
01752      to_sfixed(-0.859302, I, -F) when "11010101111",
01753      to_sfixed(-0.857729, I, -F) when "11010110000",
01754      to_sfixed(-0.856147, I, -F) when "11010110001",
01755      to_sfixed(-0.854558, I, -F) when "11010110010",

```

```

01756      to_sfixed(-0.852961, I, -F) when "11010110011",
01757      to_sfixed(-0.851355, I, -F) when "11010110100",
01758      to_sfixed(-0.849742, I, -F) when "11010110101",
01759      to_sfixed(-0.848120, I, -F) when "11010110110",
01760      to_sfixed(-0.846491, I, -F) when "11010110111",
01761      to_sfixed(-0.844854, I, -F) when "11010111000",
01762      to_sfixed(-0.843208, I, -F) when "11010111001",
01763      to_sfixed(-0.841555, I, -F) when "11010111010",
01764      to_sfixed(-0.839894, I, -F) when "11010111011",
01765      to_sfixed(-0.838225, I, -F) when "11010111100",
01766      to_sfixed(-0.836548, I, -F) when "11010111101",
01767      to_sfixed(-0.834863, I, -F) when "11010111110",
01768      to_sfixed(-0.833170, I, -F) when "11010111111",
01769      to_sfixed(-0.831470, I, -F) when "11011000000",
01770      to_sfixed(-0.829761, I, -F) when "11011000001",
01771      to_sfixed(-0.828045, I, -F) when "11011000010",
01772      to_sfixed(-0.826321, I, -F) when "11011000011",
01773      to_sfixed(-0.824589, I, -F) when "11011000100",
01774      to_sfixed(-0.822850, I, -F) when "11011000101",
01775      to_sfixed(-0.821103, I, -F) when "11011000110",
01776      to_sfixed(-0.819348, I, -F) when "11011000111",
01777      to_sfixed(-0.817585, I, -F) when "11011001000",
01778      to_sfixed(-0.815814, I, -F) when "11011001001",
01779      to_sfixed(-0.814036, I, -F) when "11011001010",
01780      to_sfixed(-0.812251, I, -F) when "11011001011",
01781      to_sfixed(-0.810457, I, -F) when "11011001100",
01782      to_sfixed(-0.808656, I, -F) when "11011001101",
01783      to_sfixed(-0.806848, I, -F) when "11011001110",
01784      to_sfixed(-0.805031, I, -F) when "11011001111",
01785      to_sfixed(-0.803208, I, -F) when "11011010000",
01786      to_sfixed(-0.801376, I, -F) when "11011010001",
01787      to_sfixed(-0.799537, I, -F) when "11011010010",
01788      to_sfixed(-0.797691, I, -F) when "11011010011",
01789      to_sfixed(-0.795837, I, -F) when "11011010100",
01790      to_sfixed(-0.793975, I, -F) when "11011010101",
01791      to_sfixed(-0.792107, I, -F) when "11011010110",
01792      to_sfixed(-0.790230, I, -F) when "11011010111",
01793      to_sfixed(-0.788346, I, -F) when "11011011000",
01794      to_sfixed(-0.786455, I, -F) when "11011011001",
01795      to_sfixed(-0.784557, I, -F) when "11011011010",
01796      to_sfixed(-0.782651, I, -F) when "11011011011",
01797      to_sfixed(-0.780737, I, -F) when "11011011100",
01798      to_sfixed(-0.778817, I, -F) when "11011011101",
01799      to_sfixed(-0.776888, I, -F) when "11011011110",
01800      to_sfixed(-0.774953, I, -F) when "11011011111",
01801      to_sfixed(-0.773010, I, -F) when "11011100000",
01802      to_sfixed(-0.771061, I, -F) when "11011100001",
01803      to_sfixed(-0.769103, I, -F) when "11011100010",
01804      to_sfixed(-0.767139, I, -F) when "11011100011",
01805      to_sfixed(-0.765167, I, -F) when "11011100100",
01806      to_sfixed(-0.763188, I, -F) when "11011100101",
01807      to_sfixed(-0.761202, I, -F) when "11011100110",
01808      to_sfixed(-0.759209, I, -F) when "11011100111",
01809      to_sfixed(-0.757209, I, -F) when "11011101000",
01810      to_sfixed(-0.755201, I, -F) when "11011101001",
01811      to_sfixed(-0.753187, I, -F) when "11011101010",
01812      to_sfixed(-0.751165, I, -F) when "11011101011",
01813      to_sfixed(-0.749136, I, -F) when "11011101100",
01814      to_sfixed(-0.747101, I, -F) when "11011101101",
01815      to_sfixed(-0.745058, I, -F) when "11011101110",
01816      to_sfixed(-0.743008, I, -F) when "11011101111",
01817      to_sfixed(-0.740951, I, -F) when "11011110000",
01818      to_sfixed(-0.738887, I, -F) when "11011110001",
01819      to_sfixed(-0.736817, I, -F) when "11011110010",
01820      to_sfixed(-0.734739, I, -F) when "11011110011",
01821      to_sfixed(-0.732654, I, -F) when "11011110100",
01822      to_sfixed(-0.730563, I, -F) when "11011110101",
01823      to_sfixed(-0.728464, I, -F) when "11011110110",
01824      to_sfixed(-0.726359, I, -F) when "11011110111",
01825      to_sfixed(-0.724247, I, -F) when "11011111000",
01826      to_sfixed(-0.722128, I, -F) when "11011111001",
01827      to_sfixed(-0.720003, I, -F) when "11011111010",
01828      to_sfixed(-0.717870, I, -F) when "11011111011",
01829      to_sfixed(-0.715731, I, -F) when "11011111100",
01830      to_sfixed(-0.713585, I, -F) when "11011111101",
01831      to_sfixed(-0.711432, I, -F) when "11011111110",
01832      to_sfixed(-0.709273, I, -F) when "11011111111",
01833      to_sfixed(-0.707107, I, -F) when "11100000000",
01834      to_sfixed(-0.704934, I, -F) when "11100000001",
01835      to_sfixed(-0.702755, I, -F) when "11100000010",
01836      to_sfixed(-0.700569, I, -F) when "11100000011",
01837      to_sfixed(-0.698376, I, -F) when "11100000100",
01838      to_sfixed(-0.696177, I, -F) when "11100000101",
01839      to_sfixed(-0.693971, I, -F) when "11100000110",
01840      to_sfixed(-0.691759, I, -F) when "11100000111",
01841      to_sfixed(-0.689541, I, -F) when "11100001000",
01842      to_sfixed(-0.687315, I, -F) when "11100001001",

```

```

01843      to_sfixed(-0.685084, I, -F) when "11100001010",
01844      to_sfixed(-0.682846, I, -F) when "11100001011",
01845      to_sfixed(-0.680601, I, -F) when "11100001100",
01846      to_sfixed(-0.678350, I, -F) when "11100001101",
01847      to_sfixed(-0.676093, I, -F) when "11100001110",
01848      to_sfixed(-0.673829, I, -F) when "11100001111",
01849      to_sfixed(-0.671559, I, -F) when "11100010000",
01850      to_sfixed(-0.669283, I, -F) when "11100010001",
01851      to_sfixed(-0.667000, I, -F) when "11100010010",
01852      to_sfixed(-0.664711, I, -F) when "11100010011",
01853      to_sfixed(-0.662416, I, -F) when "11100010100",
01854      to_sfixed(-0.660114, I, -F) when "11100010101",
01855      to_sfixed(-0.657807, I, -F) when "11100010110",
01856      to_sfixed(-0.655493, I, -F) when "11100010111",
01857      to_sfixed(-0.653173, I, -F) when "11100011000",
01858      to_sfixed(-0.650847, I, -F) when "11100011001",
01859      to_sfixed(-0.648514, I, -F) when "11100011010",
01860      to_sfixed(-0.646176, I, -F) when "11100011011",
01861      to_sfixed(-0.643832, I, -F) when "11100011100",
01862      to_sfixed(-0.641481, I, -F) when "11100011101",
01863      to_sfixed(-0.639124, I, -F) when "11100011110",
01864      to_sfixed(-0.636762, I, -F) when "11100011111",
01865      to_sfixed(-0.634393, I, -F) when "11100100000",
01866      to_sfixed(-0.632019, I, -F) when "11100100001",
01867      to_sfixed(-0.629638, I, -F) when "11100100010",
01868      to_sfixed(-0.627252, I, -F) when "11100100011",
01869      to_sfixed(-0.624859, I, -F) when "11100100100",
01870      to_sfixed(-0.622461, I, -F) when "11100100101",
01871      to_sfixed(-0.620057, I, -F) when "11100100110",
01872      to_sfixed(-0.617647, I, -F) when "11100100111",
01873      to_sfixed(-0.615232, I, -F) when "11100101000",
01874      to_sfixed(-0.612810, I, -F) when "11100101001",
01875      to_sfixed(-0.610383, I, -F) when "11100101010",
01876      to_sfixed(-0.607950, I, -F) when "11100101011",
01877      to_sfixed(-0.605511, I, -F) when "11100101100",
01878      to_sfixed(-0.603067, I, -F) when "11100101101",
01879      to_sfixed(-0.600616, I, -F) when "11100101110",
01880      to_sfixed(-0.598161, I, -F) when "11100101111",
01881      to_sfixed(-0.595699, I, -F) when "11100110000",
01882      to_sfixed(-0.593232, I, -F) when "11100110001",
01883      to_sfixed(-0.590760, I, -F) when "11100110010",
01884      to_sfixed(-0.588282, I, -F) when "11100110011",
01885      to_sfixed(-0.585798, I, -F) when "11100110100",
01886      to_sfixed(-0.583309, I, -F) when "11100110101",
01887      to_sfixed(-0.580814, I, -F) when "11100110110",
01888      to_sfixed(-0.578314, I, -F) when "11100110111",
01889      to_sfixed(-0.575808, I, -F) when "11100111000",
01890      to_sfixed(-0.573297, I, -F) when "11100111001",
01891      to_sfixed(-0.570781, I, -F) when "11100111010",
01892      to_sfixed(-0.568259, I, -F) when "11100111011",
01893      to_sfixed(-0.565732, I, -F) when "11100111100",
01894      to_sfixed(-0.563199, I, -F) when "11100111101",
01895      to_sfixed(-0.560662, I, -F) when "11100111110",
01896      to_sfixed(-0.558119, I, -F) when "11100111111",
01897      to_sfixed(-0.555570, I, -F) when "11101000000",
01898      to_sfixed(-0.553017, I, -F) when "11101000001",
01899      to_sfixed(-0.550458, I, -F) when "11101000010",
01900      to_sfixed(-0.547894, I, -F) when "11101000011",
01901      to_sfixed(-0.545325, I, -F) when "11101000100",
01902      to_sfixed(-0.542751, I, -F) when "11101000101",
01903      to_sfixed(-0.540171, I, -F) when "11101000110",
01904      to_sfixed(-0.537587, I, -F) when "11101000111",
01905      to_sfixed(-0.534998, I, -F) when "11101001000",
01906      to_sfixed(-0.532403, I, -F) when "11101001001",
01907      to_sfixed(-0.529804, I, -F) when "11101001010",
01908      to_sfixed(-0.527199, I, -F) when "11101001011",
01909      to_sfixed(-0.524590, I, -F) when "11101001100",
01910      to_sfixed(-0.521975, I, -F) when "11101001101",
01911      to_sfixed(-0.519356, I, -F) when "11101001110",
01912      to_sfixed(-0.516732, I, -F) when "11101001111",
01913      to_sfixed(-0.514103, I, -F) when "11101010000",
01914      to_sfixed(-0.511469, I, -F) when "11101010001",
01915      to_sfixed(-0.508830, I, -F) when "11101010010",
01916      to_sfixed(-0.506187, I, -F) when "11101010011",
01917      to_sfixed(-0.503538, I, -F) when "11101010100",
01918      to_sfixed(-0.500885, I, -F) when "11101010101",
01919      to_sfixed(-0.498228, I, -F) when "11101010110",
01920      to_sfixed(-0.495565, I, -F) when "11101010111",
01921      to_sfixed(-0.492898, I, -F) when "11101011000",
01922      to_sfixed(-0.490226, I, -F) when "11101011001",
01923      to_sfixed(-0.487550, I, -F) when "11101011010",
01924      to_sfixed(-0.484869, I, -F) when "11101011011",
01925      to_sfixed(-0.482184, I, -F) when "11101011100",
01926      to_sfixed(-0.479494, I, -F) when "11101011101",
01927      to_sfixed(-0.476799, I, -F) when "11101011110",
01928      to_sfixed(-0.474100, I, -F) when "11101011111",
01929      to_sfixed(-0.471397, I, -F) when "11101000000",

```

```

01930      to_sfixed(-0.468689, I, -F) when "11101100001",
01931      to_sfixed(-0.465976, I, -F) when "11101100010",
01932      to_sfixed(-0.463260, I, -F) when "11101100011",
01933      to_sfixed(-0.460539, I, -F) when "11101100100",
01934      to_sfixed(-0.457813, I, -F) when "11101100101",
01935      to_sfixed(-0.455084, I, -F) when "11101100110",
01936      to_sfixed(-0.452350, I, -F) when "11101100111",
01937      to_sfixed(-0.449611, I, -F) when "11101101000",
01938      to_sfixed(-0.446869, I, -F) when "11101101001",
01939      to_sfixed(-0.444122, I, -F) when "11101101010",
01940      to_sfixed(-0.441371, I, -F) when "11101101011",
01941      to_sfixed(-0.438616, I, -F) when "11101101100",
01942      to_sfixed(-0.435857, I, -F) when "11101101101",
01943      to_sfixed(-0.433094, I, -F) when "11101101110",
01944      to_sfixed(-0.430326, I, -F) when "11101101111",
01945      to_sfixed(-0.427555, I, -F) when "11101110000",
01946      to_sfixed(-0.424780, I, -F) when "11101110001",
01947      to_sfixed(-0.422000, I, -F) when "11101110010",
01948      to_sfixed(-0.419217, I, -F) when "11101110011",
01949      to_sfixed(-0.416430, I, -F) when "11101110100",
01950      to_sfixed(-0.413638, I, -F) when "11101110101",
01951      to_sfixed(-0.410843, I, -F) when "11101110110",
01952      to_sfixed(-0.408044, I, -F) when "11101110111",
01953      to_sfixed(-0.405241, I, -F) when "11101111000",
01954      to_sfixed(-0.402435, I, -F) when "11101111001",
01955      to_sfixed(-0.399624, I, -F) when "11101111010",
01956      to_sfixed(-0.396810, I, -F) when "11101111011",
01957      to_sfixed(-0.393992, I, -F) when "11101111100",
01958      to_sfixed(-0.391170, I, -F) when "11101111101",
01959      to_sfixed(-0.388345, I, -F) when "11101111110",
01960      to_sfixed(-0.385516, I, -F) when "11101111111",
01961      to_sfixed(-0.382683, I, -F) when "11110000000",
01962      to_sfixed(-0.379847, I, -F) when "11110000001",
01963      to_sfixed(-0.377007, I, -F) when "11110000010",
01964      to_sfixed(-0.374164, I, -F) when "11110000011",
01965      to_sfixed(-0.371317, I, -F) when "11110000100",
01966      to_sfixed(-0.368467, I, -F) when "11110000101",
01967      to_sfixed(-0.365613, I, -F) when "11110000110",
01968      to_sfixed(-0.362756, I, -F) when "11110000111",
01969      to_sfixed(-0.359895, I, -F) when "11110001000",
01970      to_sfixed(-0.357031, I, -F) when "11110001001",
01971      to_sfixed(-0.354164, I, -F) when "11110001010",
01972      to_sfixed(-0.351293, I, -F) when "11110001011",
01973      to_sfixed(-0.348419, I, -F) when "11110001100",
01974      to_sfixed(-0.345541, I, -F) when "11110001101",
01975      to_sfixed(-0.342661, I, -F) when "11110001110",
01976      to_sfixed(-0.339777, I, -F) when "11110001111",
01977      to_sfixed(-0.336890, I, -F) when "11110010000",
01978      to_sfixed(-0.334000, I, -F) when "11110010001",
01979      to_sfixed(-0.331106, I, -F) when "11110010010",
01980      to_sfixed(-0.328210, I, -F) when "11110010011",
01981      to_sfixed(-0.325310, I, -F) when "11110010100",
01982      to_sfixed(-0.322408, I, -F) when "11110010101",
01983      to_sfixed(-0.319502, I, -F) when "11110010110",
01984      to_sfixed(-0.316593, I, -F) when "11110010111",
01985      to_sfixed(-0.313682, I, -F) when "11110011000",
01986      to_sfixed(-0.310767, I, -F) when "11110011001",
01987      to_sfixed(-0.307850, I, -F) when "11110011010",
01988      to_sfixed(-0.304929, I, -F) when "11110011011",
01989      to_sfixed(-0.302006, I, -F) when "11110011100",
01990      to_sfixed(-0.299080, I, -F) when "11110011101",
01991      to_sfixed(-0.296151, I, -F) when "11110011110",
01992      to_sfixed(-0.293219, I, -F) when "11110011111",
01993      to_sfixed(-0.290285, I, -F) when "11110100000",
01994      to_sfixed(-0.287347, I, -F) when "11110100001",
01995      to_sfixed(-0.284408, I, -F) when "11110100010",
01996      to_sfixed(-0.281465, I, -F) when "11110100011",
01997      to_sfixed(-0.278520, I, -F) when "11110100100",
01998      to_sfixed(-0.275572, I, -F) when "11110100101",
01999      to_sfixed(-0.272621, I, -F) when "11110100110",
02000      to_sfixed(-0.269668, I, -F) when "11110100111",
02001      to_sfixed(-0.266713, I, -F) when "11110101000",
02002      to_sfixed(-0.263755, I, -F) when "11110101001",
02003      to_sfixed(-0.260794, I, -F) when "11110101010",
02004      to_sfixed(-0.257831, I, -F) when "11110101011",
02005      to_sfixed(-0.254866, I, -F) when "11110101100",
02006      to_sfixed(-0.251898, I, -F) when "11110101101",
02007      to_sfixed(-0.248928, I, -F) when "11110101110",
02008      to_sfixed(-0.245955, I, -F) when "11110101111",
02009      to_sfixed(-0.242980, I, -F) when "11110110000",
02010      to_sfixed(-0.240003, I, -F) when "11110110001",
02011      to_sfixed(-0.237024, I, -F) when "11110110010",
02012      to_sfixed(-0.234042, I, -F) when "11110110011",
02013      to_sfixed(-0.231058, I, -F) when "11110110100",
02014      to_sfixed(-0.228072, I, -F) when "11110110101",
02015      to_sfixed(-0.225084, I, -F) when "11110110110",
02016      to_sfixed(-0.222094, I, -F) when "11110110111",

```

```

02017      to_sfixed(-0.219101, I, -F) when "11110111000",
02018      to_sfixed(-0.216107, I, -F) when "11110111001",
02019      to_sfixed(-0.213110, I, -F) when "11110111010",
02020      to_sfixed(-0.210112, I, -F) when "11110111011",
02021      to_sfixed(-0.207111, I, -F) when "11110111100",
02022      to_sfixed(-0.204109, I, -F) when "11110111101",
02023      to_sfixed(-0.201105, I, -F) when "11110111110",
02024      to_sfixed(-0.198098, I, -F) when "11110111111",
02025      to_sfixed(-0.195090, I, -F) when "11111000000",
02026      to_sfixed(-0.192080, I, -F) when "11111000001",
02027      to_sfixed(-0.189069, I, -F) when "11111000010",
02028      to_sfixed(-0.186055, I, -F) when "11111000011",
02029      to_sfixed(-0.183040, I, -F) when "11111000100",
02030      to_sfixed(-0.180023, I, -F) when "11111000101",
02031      to_sfixed(-0.177004, I, -F) when "11111000110",
02032      to_sfixed(-0.173984, I, -F) when "11111000111",
02033      to_sfixed(-0.170962, I, -F) when "11111001000",
02034      to_sfixed(-0.167938, I, -F) when "11111001001",
02035      to_sfixed(-0.164913, I, -F) when "11111001010",
02036      to_sfixed(-0.161886, I, -F) when "11111001011",
02037      to_sfixed(-0.158858, I, -F) when "11111001100",
02038      to_sfixed(-0.155828, I, -F) when "11111001101",
02039      to_sfixed(-0.152797, I, -F) when "11111001110",
02040      to_sfixed(-0.149765, I, -F) when "11111001111",
02041      to_sfixed(-0.146730, I, -F) when "11111010000",
02042      to_sfixed(-0.143695, I, -F) when "11111010001",
02043      to_sfixed(-0.140658, I, -F) when "11111010010",
02044      to_sfixed(-0.137620, I, -F) when "11111010011",
02045      to_sfixed(-0.134581, I, -F) when "11111010100",
02046      to_sfixed(-0.131540, I, -F) when "11111010101",
02047      to_sfixed(-0.128498, I, -F) when "11111010110",
02048      to_sfixed(-0.125455, I, -F) when "11111010111",
02049      to_sfixed(-0.122411, I, -F) when "11111011000",
02050      to_sfixed(-0.119365, I, -F) when "11111011001",
02051      to_sfixed(-0.116319, I, -F) when "11111011100",
02052      to_sfixed(-0.113271, I, -F) when "11111011101",
02053      to_sfixed(-0.110222, I, -F) when "11111011100",
02054      to_sfixed(-0.107172, I, -F) when "11111011101",
02055      to_sfixed(-0.104122, I, -F) when "11111011110",
02056      to_sfixed(-0.101070, I, -F) when "11111011111",
02057      to_sfixed(-0.098017, I, -F) when "11111100000",
02058      to_sfixed(-0.094963, I, -F) when "11111100001",
02059      to_sfixed(-0.091909, I, -F) when "11111100010",
02060      to_sfixed(-0.088854, I, -F) when "11111100011",
02061      to_sfixed(-0.085797, I, -F) when "11111100100",
02062      to_sfixed(-0.082740, I, -F) when "11111100101",
02063      to_sfixed(-0.079682, I, -F) when "11111100110",
02064      to_sfixed(-0.076624, I, -F) when "11111100111",
02065      to_sfixed(-0.073565, I, -F) when "11111101000",
02066      to_sfixed(-0.070505, I, -F) when "11111101001",
02067      to_sfixed(-0.067444, I, -F) when "11111101010",
02068      to_sfixed(-0.0644383, I, -F) when "11111101011",
02069      to_sfixed(-0.061321, I, -F) when "11111101100",
02070      to_sfixed(-0.058258, I, -F) when "11111101101",
02071      to_sfixed(-0.055195, I, -F) when "11111101110",
02072      to_sfixed(-0.052132, I, -F) when "11111101111",
02073      to_sfixed(-0.049068, I, -F) when "11111110000",
02074      to_sfixed(-0.046003, I, -F) when "11111110001",
02075      to_sfixed(-0.042938, I, -F) when "11111110010",
02076      to_sfixed(-0.039873, I, -F) when "11111110011",
02077      to_sfixed(-0.036807, I, -F) when "11111110100",
02078      to_sfixed(-0.033741, I, -F) when "11111110101",
02079      to_sfixed(-0.030675, I, -F) when "11111110110",
02080      to_sfixed(-0.027608, I, -F) when "11111110111",
02081      to_sfixed(-0.024541, I, -F) when "11111111000",
02082      to_sfixed(-0.021474, I, -F) when "11111111001",
02083      to_sfixed(-0.018407, I, -F) when "11111111010",
02084      to_sfixed(-0.015339, I, -F) when "11111111011",
02085      to_sfixed(-0.012272, I, -F) when "11111111100",
02086      to_sfixed(-0.009204, I, -F) when "11111111101",
02087      to_sfixed(-0.006136, I, -F) when "11111111110",
02088      to_sfixed(-0.003068, I, -F) when "11111111111",
02089      to_sfixed(0, I, -F) when others;
02090
02091 end architecture tabela_sin;
02092

```

7.37 theta_abc.vhd File Reference

Entities

- [theta_abc entity](#)

- `theta_abc` architecture

7.38 theta_abc.vhd

```

00001 library IEEE;
00002 use IEEE.STD_LOGIC_UNSIGNED.all;
00003 use IEEE.STD_LOGIC_1164.all;
00004 use ieee.fixed_pkg.all;
00005 use ieee.numeric_std.all;
00006 use ieee.fixed_pkg.all;           --arquivo deve ser adicionado ao projeto
00007
00008 entity theta_abc is
00009 generic  (constant Nin : integer := 30;  --numero de bits da parte inteira excluindo sinal de entrada
00010          constant Nout : integer := 16   --numero de bits da parte inteira excluindo sinal de saida
00011          );
00012 port(
00013     clk : in std_logic; -- clock
00014     en : in std_logic; -- habilita modulo
00015     reset : in std_logic; --
00016     theta_a : out std_logic_vector(Nout-1 downto 0); -- data out
00017     theta_b : out std_logic_vector(Nout-1 downto 0); -- data out
00018     theta_c : out std_logic_vector(Nout-1 downto 0); -- data out
00019     theta_in : in std_logic_vector(Nin-1 downto 0) -- data in
00020 );
00021
00022 end entity theta_abc;
00023
00024
00025 architecture theta_abc of theta_abc is
00026
00027     signal th_a, th_b, th_c : std_logic_vector(Nout-1 downto 0);
00028     signal th_ai, th_bi, th_ci : std_logic_vector(Nout downto 0);
00029
00030
00031 begin
00032
00033     process(clk)
00034     begin
00035         if en = '1' then
00036             if reset = '1' then
00037                 th_a <= (others => '0');
00038                 th_b <= (others => '0');
00039                 th_c <= (others => '0');
00040             elsif rising_edge(clk) then
00041                 th_ai <= std_logic_vector(signed(signed(theta_in(
00042                     Nin-1 downto Nin-Nout-1)) +to_signed(-32767,17))); -- 29 downto 13
00043                 th_bi <= std_logic_vector(signed(signed(theta_in(
00044                     Nin-1 downto Nin-Nout-1)) +to_signed(-10922,17)));
00045                 th_ci <= std_logic_vector(signed(signed(theta_in(
00046                     Nin-1 downto Nin-Nout-1)) +to_signed(10922,17)));
00047
00048                 th_a <= th_ai(Nout-1 downto 0); -- Retorna para 16 bits
00049                 th_b <= th_bi(Nout-1 downto 0);
00050                 th_c <= th_ci(Nout-1 downto 0);
00051             end if;
00052         else
00053             th_a <= (others => '0');
00054             th_b <= (others => '0');
00055             th_c <= (others => '0');
00056         end if;
00057     end process;
00058
00059     theta_a <= th_a;
00060     theta_b <= th_b;
00061     theta_c <= th_c;
00062 end architecture theta_abc;

```

7.39 trash.vhd File Reference

7.40 trash.vhd

```

00001
00002
00003
00004
00005
00006 /*u2comp: comparador port map(

```

```

00007      clk => clk_pll, -- clock
00008      en => '1', -- habilita modulo
00009      comp => std_logic_vector(to_unsigned(267, 16)), -- moduladora
00010      c => cPWM2, -- portadora
00011      amost => clk_pll, -- amostra moduladora na borda de amost
00012      comp_out => GPIO_0(2)
00013      );
00014
00015  /*
00016
00017 */
00018
00019  -- led indica frequencia da fundamental
00020  uled: clk_div port map (clk_in => sinc_int,
00021                           en => '1',
00022                           div => std_logic_vector(to_unsigned(6, 16)),
00023                           clk_out => LED(0)
00024                           --clk_out => GPIO_0(2)
00025                           );
00026
00027
00028
00029
00030
00031
00032  ucomp: comparador port map(
00033      clk => clk_pll, -- clock
00034      en => '1', -- habilita modulo
00035      comp => std_logic_vector(to_unsigned(267, 16)), -- moduladora
00036      c => cTR1I, -- portadora
00037      amost => clk_pll, -- amostra moduladora na borda de amost
00038      comp_out => sigPWM01
00039      );
00040
00041
00042
00043  deadcomp : deadtime port map(
00044      p_Pwm_In => sigPWM01,
00045      CLK => clk_pll, -- clock
00046      p_Pwm1_Out => GPIO_0(0),
00047      p_Pwm2_Out => GPIO_0(1)
00048      );
00049
00050
00051
00052
00053  -- defasa theta para sistema trifasico
00054  u6: theta_abc port map(
00055      clk => clk_int,
00056      en => '1',
00057      reset => '0',
00058      theta_a => th_a,
00059      theta_b => th_b,
00060      theta_c => th_c,
00061      theta_in => theta_pll
00062      );
00063
00064
00065
00066  usin: tabela_sin port map (clk => clk_pll,
00067      theta => th_a,
00068      va => sin_a
00069      );
00070
00071
00072
00073  -- clk_int = 6.666_ MHz
00074  ul: clk_div port map (clk_in => clk_pll,
00075                           en => '1',
00076                           div => std_logic_vector(to_unsigned(4, 16)), -- function TO_UNSIGNED
00077                           (ARG, SIZE: NATURAL) return UNSIGNED;
00078                           clk_out => clk_int
00079                           );
00080
00081
00082
00083  -- int_data = 3428 => 60 Hz
00084  u5: integrador port map(
00085      clk => clk_int,
00086      en => '1',
00087      reset => '0',
00088      MAX => std_logic_vector(to_unsigned(536870911, 30)),
00089      sinc => sinc_int,
00090      out_data => theta_pll,
00091      int_data => std_logic_vector(to_unsigned(4832, 13))
00092      --int_data => omega_pll

```

```
00093
00094 );
```

7.41 vfcontrol.vhd File Reference

Entities

- `vfcontrol` entity
- `vfcontrol_arch` architecture

7.42 vfcontrol.vhd

```
00001
00002 library IEEE;
00003 use IEEE.STD_LOGIC_UNSIGNED.all;
00004 use IEEE.STD_LOGIC_1164.all;
00005 use IEEE.STD_LOGIC_ARITH.ALL;
00006 use ieee.fixed_pkg.all;           --arquivo deve ser adicionado ao projeto
00007 use ieee.numeric_std.all;
00008
00009
00010 entity vfcontrol is
00011     generic (
00012         constant n_bits_c: integer := 16; --numero de bits da portadora
00013         -- constant incMAX : std_logic_vector(12 downto 0) := std_logic_vector(to_unsigned(4832, 13)); -- Max
00014         constant incMAX : std_logic_vector(12 downto 0) := std_logic_vector(to_unsigned(2416, 13));
00015         -- Max INC for 30Hz
00016         constant incMIN : std_logic_vector(12 downto 0) := std_logic_vector(to_unsigned(483, 13));
00017         -- Min INC for 6Hz
00018         constant I : integer := 1;   --número de bits da parte inteira excluindo sinal
00019         constant F : integer := 14; --número de bits da parte fracionária
00020         -- constant mMAX : sfixed(1 downto -27) := to_sfixed(0.8137, 1, -27); -- Max modulation index
00021         constant mMAX : sfixed(1 downto -27) := to_sfixed(0.4069, 1, -27); -- Max modulation index
00022         constant mMIN : sfixed(1 downto -27) := to_sfixed(0.08137, 1, -27) -- Min modulation index
00023     );
00024
00025     port(
00026         clk : in std_logic; -- clock
00027         en : in std_logic; -- reset
00028         inc_data : out std_logic_vector(12 downto 0); -- incremento do integrador
00029         m_vf : out sfixed(I downto -F) --
00030     );
00031
00032 -- int_data = 4832 => 60 Hz      Delta = 4349 Dm= 0.73233
00033 -- 483 => 6 Hz
00034 end entity vfcontrol;
00035
00036
00037 architecture vfcontrol_arch of vfcontrol is
00038     signal incsignal : std_logic_vector(12 downto 0) := std_logic_vector(to_unsigned(0, 13)); -- 13
00039     bits signal
00040     signal msignal : sfixed(1 downto -27) := to_sfixed(0.08137, 1, -27); -- 16 bits signal
00041     signal incstep : std_logic_vector(12 downto 0) := std_logic_vector(to_unsigned(1, 13)); -- data
00042     inc step
00043     signal mstep : sfixed(1 downto -27) := to_sfixed(1.6839e-04, 1, -27); -- 16 bits signal
00044
00045 begin
00046     process(clk)
00047         begin
00048             if en = '0' then
00049                 incsignal <= incMIN; -- 483
00050                 msignal <= mMIN; -- 0.08137
00051
00052             elsif rising_edge(clk) then
00053                 if incsignal(12 downto 0) < incMAX(12 downto 0) then
00054                     incsignal <= incsignal+incstep; -- data inc step
00055                     msignal <= resize((msignal + mstep),1,-27);
00056
00057                 else
00058                     incsignal <= incMAX; -- 4832
00059                     msignal <= mMAX; -- 0.8137
00060                 end if;
00061             end if;
00062         end process;
00063
```

```

00061      inc_data <= incsignal; -- incremento do integrador
00062      m_vf <= resize(msignal,1,-14); -- Indice de Modulação
00063
00064
00065 end architecture vfcontrol_arch;

```

7.43 wt.vhd File Reference

Entities

- [wt entity](#)
- [wt_arch architecture](#)

7.44 wt.vhd

```

00001 library IEEE;
00002 use IEEE.STD_LOGIC_UNSIGNED.all;
00003 use IEEE.STD_LOGIC_1164.all;           -- arquivo deve ser adicionado ao projeto    16 bits 0 to 65535
00005
00006 -- 16 bits 0 to 65535
00007
00008 entity wt is
00009 generic (
00010     constant Nbits : integer := 16 -- numero de bits
00011 );
00012 port(
00013     clk : in std_logic; -- clock
00014     en : in std_logic; -- habilita modulo
00015     reset : in std_logic; --
00016     sinc : out std_logic;
00017     MAX : in std_logic_vector(Nbits-1 downto 0);
00018     out_data : out std_logic_vector(Nbits-1 downto 0); -- data out
00019     int_data : in std_logic_vector(Nbits-1 downto 0) -- data in
00020 );
00021
00022 end entity wt;
00023
00024
00025 architecture wt_arch of wt is
00026
00027     signal out_int : std_logic_vector(Nbits-1 downto 0);
00028     signal sinc_int : std_logic;
00029
00030
00031 begin
00032
00033     process(clk)
00034         begin
00035             if en = '1' then
00036
00037                 if reset = '1' then
00038                     out_int <= (others => '0');
00039                     sinc_int <= '0';
00040                 elsif rising_edge(clk) then
00041                     if out_int < MAX then
00042                         out_int <= out_int+int_data;
00043                         --sinc_int <= '0';
00044                         if out_int(Nbits-2 downto 0) < MAX(
00045                             Nbits-1 downto 1) then
00046                             sinc_int <= '0';
00047                         else
00048                             sinc_int <= '1';
00049                         end if;
00050                     else
00051                         out_int <= (others => '0');
00052                         --sinc_int <= '1';
00053                     end if;
00054                 else
00055                     out_int <= (others => '0');
00056                     --sinc_int <= (others => '0');
00057                 end if;
00058             end process;
00059
00060             out_data <= out_int;
00061             sinc <= sinc_int;

```

```
00062 end architecture wt_arch;
```