LEXICAL ELEMENTS

- Comments extend from -- to end of line
- Upper and lower case letters are equivalent except within string literals and character literals.
- Character literals: 'A' 'a' '*' ':' ' '
- String literals: "" """" "UPPER lower"
- Bit string literals: B"10_10" O"731" X"FF_DD"
- Physical literals: 5 kohm 20 msec
- Integer literals: 123_000 1E6 2#101# 16#F#E6
- Floating point literals: 1_000.00 1.2E6 2#10.1# 16#F.0#E-5

RESERVED WORDS

abs	else	nand	select
access	elsif	new	severity
after	end	next	signal
alias	entity	nor	subtype
all	exit	not	
and		null	then
architecture	file		to
array	for	of	transport
assert	function	on	type
attribute		open	
	generate	or	units
begin	generic	others	until
block	guarded	out	use
body			
buffer	if	package	variable
bus	in	port	
	inout	procedure	wait
case	is	process	when
component			while
configuration	label	range	with
constant	library	record	
	linkage	register	xor
disconnect	loop	rem	
downto		report	
	map	return	
	mod		

OBJECT DECLARATIONS

```
-- Constant declaration
constant DELAY : TIME := 10 NS;
-- Variable declaration
variable INSTRUCTION : BIT_VECTOR(15 downto 0)
  := X"0000";
-- Signal declaration
signal ENABLE : BIT;
-- Alias declaration
alias MODE : BIT is INSTRUCTION(15);
```

SCALAR TYPE DECLARATIONS

```
-- Integer type declaration
type ADDRESS is range 0 to 16#FFFF# ;
type BIT_NUMBER is range 7 downto 0 ;

-- Floating point type declaration
type FLOATING_POINT_REGISTER is range
  -16#0.7FFF_FF8#E+32 to 16#0.7FFF_FF8#E+32 ;

-- Enumeration type declaration
type LOGIC is ('X', '0', '1', 'Z') ;

-- Physical type declaration
type RESISTANCE is range 0 to 1E10
  units
    ohm ;
    kohm = 1000 ohm ;
  end units ;

-- Subtype with integer range constraint
subtype ROM_ADDRESS is ADDRESS range 16#A000# to
  16#FFFF# ;

-- Subtype with enumeration constraint
subtype THREE_LEVEL_LOGIC is LOGIC range '0' to 'Z' ;

-- Subtype with resolution function
subtype BUS_LOGIC is RESOLVE_DRIVERS
  THREE_LEVEL_LOGIC ;
```

OTHER TYPE DECLARATIONS

```
-- Constrained array type
type BYTE is array (7 downto 0) of BIT ;
type BYTE is array (BIT_NUMBER) of BIT ;

-- Unconstrained array type
type MEMORY is array (NATURAL range < >) of BYTE ;

-- Two-dimensional unconstrained array type
type PLOT is array (NATURAL range < >,
    NATURAL range < >) of BIT ;

-- Subtype with index constraint
subtype SMALL_PLOT is PLOT(0 to 20, 0 to 20) ;

-- Record type declaration
type MACHINE_STATE is record
  REGISTERS : REGISTER_ARRAY ;
  PROGRAM_STATUS : PSW ;
end record ;
```

```
-- Incomplete type declaration for use in access type declaration
type CELL ;
-- Access type declaration
type LINK is access CELL ;
type CELL is record
  VALUE : INTEGER ;
  NEXT_CELL : LINK ;
end record ;
```

NAMES

```
-- Selected names
CONTEXT.PSW     -- element of record object CONTEXT
GLOBAL_SIGNALS.GROUND   -- signal declared in a package
                        -- GLOBAL_SIGNALS
WORK.all   -- units in library WORK
CURRENT_CELL.all   -- object pointed to by access object
                   -- CURRENT_CELL

-- Indexed names
BYTE(0)    -- element of one-dimensional array
GP_REGISTERS(1)(15)  -- element of one-dimensional array
                     -- of one-dimensional arrays
SCREEN(10, 0) -- element of two-dimensional array

-- Slice names
ROM(0 to HIGH_ADDRESS)
GP_REGISTERS(1)(31 downto 31)

-- Attribute names
S'STABLE(2 NS)       -- predefined signal attribute
SCREEN'RANGE(1)      -- predefined array attribute
ADDER'LOCATION       -- user-defined attribute
```

OPERATORS

HIGHEST PRECEDENCE	
exponentiation, absolute value, complement	** abs not
multiplying	* / mod rem
sign (unary)	+ -
adding and concatenation	+ - &
relational	= /= < <= > >=
logical	and or nand nor xor
LOWEST PRECEDENCE	

EXPRESSIONS

```
-- Aggregates
('0', '1', '0', '1', '0', '1', '0', '1') -- positional association
(REGISTERS => (others => '0'),     -- named association
   (0 to 15 => (others => '0')),
   PROGRAM_STATUS => (others => '0') )

-- Function call
MAX(S1, S2)    -- function call with two parameters

-- Allocators
new CELL    -- new object not explicitly initialized
new CELL'(0, null)   -- new object initialized

-- Parenthesized expression
(S1 + S2)

-- Qualified expression
BYTE'("00000000")   -- specifies the type explicitly

-- Type conversion
LONG(INDEX)  -- converts value of INDEX to type LONG
```

ATTRIBUTE DECLARATIONS AND SPECIFICATIONS

```
-- Attribute declaration
attribute LEVEL : NATURAL ;

-- Attribute specification
attribute LEVEL of U0, U1 : label is 2 ;
attribute LEVEL of others : label is 1 ;
attribute TECHNOLOGY of all : signal is CMOS ;
attribute ADDRESS of MAX_TEMP : constant is 16#E020# ;
```

COMPONENT DECLARATIONS AND CONFIGURATION SPECIFICATIONS

```
-- Component declaration
component CNTR
  generic (T_RESET : TIME := 10 NS) ;
  port (CLOCK, RESET : in BIT ; INCR : buffer INTEGER) ;
end component ;

-- Configuration specification (binds component instantiations to
-- an entity/architecture pair and binds local generics/ports to
-- formal generics/ports)
for U1, U2 : CNTR use entity WORK.COUNTER(BEHAVIOR)
  generic map (RESET_DELAY = > T_RESET)
  port map (CLK = > CLOCK, RESET = > RESET,
            SUM = > INCR) ;
```

SEQUENTIAL STATEMENTS

```
-- If
if PRESET = '0' and CLEAR = '0' then
  assert FALSE report "Flip-flop error" ;
elsif PRESET = '0' and CLEAR = '1' then
  OUTPUT <= '1' ;  -- zero-delay signal assignment
elsif PRESET = '1' and CLEAR = '0' then
  OUTPUT <= '0' ;
else
  OUTPUT <= INPUT after DELAY ;
end if ;

-- Case
case OPCODE is
  when LOGICAL_OPS = >   -- a subtype of type of OPCODE
    X_LOGICAL_OPS(INSTRUCTION) ;  -- procedure call
  when ADD_OP | ADC_OP = >  -- union of choices
    X_ADD_OPS(INSTRUCTION) ;
  when EQ_OP to GT_OP = >  -- a bounded range
    X_REL_OPS(INSTRUCTION) ;
  when others = >
    assert FALSE report "Bad opcode" ;
end case ;

case SELECT_LINES is   -- case expression is one-dimensional
                       -- array of a character type
  when "00" = >
    OUTPUTS <= "0001" ;  -- zero-delay signal assignment
  when "01" = >
    OUTPUTS <= "0010" ;
  when "10" = >
    OUTPUTS <= "0100" ;
  when "11" = >
    OUTPUTS <= "1000" ;
end case ;

-- Loop, Exit, Next, Variable Assignment, Procedure Call
loop
  L1 : for I in S_HIGH downto S_LOW loop
    J := 1 ;   -- Variable assignment
    L2 : while X(I) > Y(J) loop
      if Y(J) = 0 then
        next L1 ;    -- goes directly to top of loop L1
      end if ;
      SUMMATION(X(I), Y(I), SUM) ;  -- Procedure call
      exit L2 when J = N ;  -- exits loop L2 when
                            -- condition is TRUE
      J := J + 1 ;  -- Variable assignment
    end loop L2 ;
  end loop L1 ;
  exit when SUM > LIMIT ;  -- exits main loop when
                           -- condition is TRUE
end loop ;
```

```
-- Assertion
assert not (PRESET = '0' and CLEAR = '0')
  report "PRESET and CLEAR both '0'" severity ERROR ;

-- Signal assignment with transport delay
S <= transport INPUT after 5 NS, '0' after 15 NS ;
-- Signal assignment with inertial delay and null transaction
S <= INPUT after 5 NS, null after 15 NS ;

-- Wait
wait on A, B until C = '0' for 100 NS ;

-- Return (see subprograms)
```

CONCURRENT STATEMENTS

```
-- Block
CONTROLLER:  -- label is obligatory
block
    port  -- a block may have formal generics and ports
    (S : out TRISTATE ; DATA_1, DATA_2 : TRISTATE ; C : BIT) ;
    port map  -- actuals are associated with these formals
    (S => DATA_BUS , DATA_1 => SENSOR_1 ,
     DATA_2 => SENSOR_2 , C => ENABLE) ;
  -- Type, object (but no variables), subprogram, attribute,
  -- component declarations, subprogram bodies, attribute,
  -- configuration and disconnection specifications, and use
  -- clauses may occur here
begin
  -- Concurrent statements, including nested blocks, may occur
  -- in a block
end block ;

-- Block with guard and disconnection specification
DEVICE_1:
block (C = '0')  -- guard expression
  signal S: TRISTATE register ;
  disconnect S : TRISTATE after 0 NS ;
begin
  -- Guarded signal assignment.  Since S was declared a
  --  register, when the guard goes false, this driver will
  --  disconnect after 0 NS
  S <= guarded DATA_1 after DELAY ;
end block ;
```

```
-- Process
U2:  -- label is optional
process
  -- Type, object (but no signals), subprogram, attribute
  --  declarations, subprogram bodies, attribute
  --  specifications, and use clauses may occur here
  variable V : BIT := '0' ;
begin
  -- Sequential statements may occur in a process
  OUT1 <= IN1 xor V after 5 NS ;  -- Sequential signal
                                  --  assignment
  V := IN1 and IN2 ;  -- Variable assignment
  wait on IN1, IN2 ;  -- Wait statement with sensitivity clause
end process ;

-- Generate
SHIFTER:  -- label is obligatory
for I in SIZE downto 1 generate
  -- Concurrent statements, including nested generates, may
  -- occur in a generate
  S(I) <= S(I - 1) ;
  EVEN_BITS:
  if I mod 2 = 0 generate
    EVEN(I/2) <= S(I) ;
  end generate ;
end generate ;

-- Component instantiation statement
HA1:  -- label is obligatory
HALF_ADDER generic map (DELAY = > 10 NS)
  port map (X, Y, SUM) ;

-- Conditional signal assignment, simplest form
SUM <= A + B after 10 NS ;
-- Conditional signal assignment, general form
FF:  -- label is optional
OUTPUT <=
  '1' after 10 NS when PRESET = '0' and CLEAR = '1' else
  '0' after 10 NS when PRESET = '1' and CLEAR = '0' else
  INPUT after 10 NS ;

-- Selected signal assignment
DECODE:   -- label is optional
with SELECT_LINES select
  OUTPUTS <= "0001" when "00" ,
             "0010" when "01" ,
             "0100" when "10" ,
             "1000" when "11" ;
```

```
-- Concurrent assertion
CHECK_PRESET_CLEAR:  -- Concurrent assertion has
                     -- optional label
assert not (PRESET = '0' and CLEAR = '0')
  report "PRESET and CLEAR both '0'" severity ERROR ;

-- Concurrent procedure call
DECODE_INSTR:    -- Concurrent procedure call has
                 -- optional label
DECODE(INSTRUCTION, OPCODE, OPERAND1, OPERAND2) ;
```

ENTITY DECLARATIONS AND ARCHITECTURE BODIES

```
entity COUNTER is
  generic (RESET_DELAY : TIME := 10 NS) ;
  port (CLK, RESET : in BIT ; SUM : buffer INTEGER) ;
  -- Type, object (no variables), subprogram, attribute
  -- declarations, subprogram bodies, attribute and
  -- disconnection specifications, and use clauses may occur
  -- here
begin
  -- Entity may contain concurrent statements that do
  -- not assign to signals
  assert CLK = '1' or RESET = '0'
    report "RESET timing error" ;
end COUNTER ;

architecture BEHAVIOR of COUNTER is
  -- Type, object (but no variables), subprogram, attribute,
  -- component declarations, subprogram bodies, attribute,
  -- configuration and disconnection specifications, and use
  -- clauses may occur here
  constant DELAY : TIME := 10 NS ;
begin
  -- Concurrent statements may occur in an architecture
  SUM <=
    SUM + 1 after DELAY
      when CLK = '1' and RESET = '0' else
    0 after RESET_DELAY
      when CLK = '1' and RESET = '1' else
    SUM after DELAY ;
end BEHAVIOR ;
```

PACKAGE AND SUBPROGRAM DECLARATIONS

```
package TRISTATE is
  -- Type, object (no variables), subprogram, component,
  -- attribute declarations, attribute and disconnection
  -- specifications, and use clauses may occur here
  type LOGIC is ('0', '1', 'Z') ;
  type WIRE_ARRAY is array (NATURAL range < >) of LOGIC ;
  function WIRED_OR (P: WIRE_ARRAY) return LOGIC ;
  subtype BUS_TYPE is WIRED_OR LOGIC ;
  procedure ADVANCE (variable STATE : inout CYCLE ;
    signal ENABLE : out LOGIC) ;
end TRISTATE ;
```

PACKAGE AND SUBPROGRAM BODIES

```
package body TRISTATE is
  -- Type, object (no signals or variables), subprogram
  -- declarations, subprogram bodies and use clauses may occur
  -- here

  -- Function body
  function WIRED_OR (P: WIRE_ARRAY) return LOGIC is
    -- Type, object (no signals), subprogram, attribute
    -- declarations, subprogram bodies, attribute specifications,
    -- and use clauses may occur here
  begin
    -- Sequential statements may occur in a function
    for I in P'RANGE loop
      if P(I) = '1' then
        return '1' ;
      end if ;
    end loop ;
    return '0' ;
  end WIRED_OR ;

  -- Procedure body
  procedure ADVANCE (variable STATE : inout CYCLE ;
    signal ENABLE : out LOGIC) is
    -- Type, object (no signals), subprogram, attribute
    -- declarations, subprogram bodies, attribute specifications,
    -- and use clauses may occur here
  begin
    -- Sequential statements may occur in a procedure
    STATE := CYCLE'SUCC(STATE) ;
    if STATE = WAIT_STATE then
      ENABLE <= '0' after 5 NS ;
    else
      ENABLE <= '1' after 5 NS ;
    end if ;
  end ADVANCE ;

end TRISTATE ;  -- end of package body
```

CONFIGURATION DECLARATION

```
configuration C1 of WORK.MEM_BOARD is  -- MEM_BOARD is
                                       -- an entity
  -- Attribute specifications and use clauses may appear at the
  -- top of a configuration declaration
  for MEM_BOARD_BODY  -- a block configuration;
    -- MEM_BOARD_BODY is an architecture of MEM_BOARD
    for M0 : MEMORY  -- a component configuration; configures
             -- a component instantiation in MEM_BOARD_BODY
      use entity PARTS.M_UNIT(TCA12_400) ;
      for TCA12_400
        for CONTROL : MCU use entity PARTS.MCU(TCA03_409) ;
        end for ;
        for U0 : RAM use entity PARTS.B_DRAM(TCA10_813) ;
        end for ;
      end for ;
    end for ;
    for DATA : DATA_BUS use entity PARTS.E_BUS(TCA22_665) ;
    end for ;
  end for ;
end C1 ;
```

VISIBILITY

- Library clause makes name of library visible

```
library PARTS ;
```

- Names of libraries WORK and STD are automatically visible, as if the library clause "library WORK, STD ;" preceded every library unit. WORK designates whatever library a unit is being analyzed into. STD contains the packages STANDARD and TEXTIO.
- Use clause makes names of packages visible or makes declarations inside a package visible

```
use PARTS.MCU ;     -- name of entity MCU in library PARTS
                    -- is visible
use PARTS.all ;  -- names of all entities, packages, and
                 -- configurations in library PARTS are visible
use TYPES.TRISTATE ;  -- name of package TRISTATE in
                      -- library TYPES is visible
use TRISTATE.LOGIC ;  -- type LOGIC in package TRISTATE
                      -- is visible
use TRISTATE.all ;  -- names of all declarations in package
                    -- TRISTATE are visible
use TYPES.TRISTATE.all ;  -- all declarations in package
                          -- TRISTATE in library TYPES
```

- The declarations in package STD.STANDARD are automatically visible, as if the use clause "use STD.STANDARD.all ;" preceded every library unit.

FILES

```
-- File type
type BIT_STREAM is file of BIT ;

-- File objects
file TEST_DATA : BIT_STREAM is in "/usr/cs/tst001.dat" ;
file RESULTS : BIT_STREAM is out "/usr/cs/tst001.res" ;

-- Subprograms predefined for each file type:
procedure READ (F : in BIT_STREAM ; VALUE : out BIT) ;
procedure WRITE (F : out BIT_STREAM ; VALUE : in BIT) ;
function ENDFILE (F : in BIT_STREAM) return BOOLEAN ;
```

PREDEFINED ATTRIBUTES

Attributes defined on type or subtype T.

- T'BASE – The base type of T.
- T'LEFT – The left bound of T.
- T'RIGHT – The right bound of T.
- T'HIGH – The upper bound of T.
- T'LOW – The lower bound of T.
- T'POS(X) – The position number of X in the base type of T.
- T'VAL(X) – The value in T of position X.
- T'SUCC(X) – T'VAL(T'POS(X) + 1)
- T'PRED(X) – T'VAL(T'POS(X)-1)
- T'LEFTOF(X) – T'PRED(X) if T is ascending; T'SUCC(X) if T is descending.
- T'RIGHTOF(X) – T'SUCC(X) if T is ascending; T'PRED(X) if T is descending.

Attributes defined on array object or constrained array subtype A.

- A'LEFT(N) – Left bound of the Nth index of A.
- A'RIGHT(N) – Right bound of the Nth index of A.
- A'HIGH(N) – High bound of the Nth index of A.
- A'LOW(N) – Low bound of the Nth index of A.
- A'RANGE(N) – Range of the Nth index of A.
- A'REVERSE_RANGE(N) – Reverse range of the Nth index of A.
- A'LENGTH(N) – Number of values in the Nth index of A.

Attributes which are defined on a signal object S; resulting attribute names are themselves signals.

- S'DELAYED(T) – Has the value of S at T units of time before NOW.
- S'STABLE(T) – Has the value TRUE if there has not been an event on S for T units of time.
- S'QUIET(T) – Has the value TRUE when S has been quiet for T units of time.
- S'TRANSACTION – A BIT signal whose value toggles each time S is active.

Attributes which are defined on a signal object S; resulting attribute names are not themselves signals.

- S'EVENT – Has the value TRUE if an event has just occurred on S.
- S'ACTIVE – Has the value TRUE if S is active.
- S'LAST_EVENT – The amount of time that has elapsed since the last event on S.
- S'LAST_ACTIVE – The amount of time that has elapsed since the last time S was active.
- S'LAST_VALUE – The value of S before the most recent event on S.

Attributes which are defined for architecture or block B.

- B'BEHAVIOR – Has the value FALSE if B contains a component instantiation statement.
- B'STRUCTURE – Has the value TRUE if B contains no signal assignments and contains no calls to procedures which contain signal assignments.

PACKAGES STANDARD AND TEXTIO

- STANDARD contains declarations for types BOOLEAN, BIT, CHARACTER, SEVERITY_LEVEL, TIME, INTEGER, REAL, NATURAL, POSITIVE, BIT_VECTOR, STRING, and a declaration for function NOW (returns current simulation time).
- TEXTIO contains declarations supporting formatted ASCII I/O. Procedures READ and WRITE (each overloaded for types BIT, BIT_VECTOR, BOOLEAN, CHARACTER, INTEGER, REAL, STRING, and TIME) fetch a value from a line and put a value to a line. Procedure READLINE reads a line from a file; procedure WRITELINE writes a line to a file. Functions ENDLINE and ENDFILE test for end of line or end of file.