VHDL Summary - CE1901/1911/1921 - Dr. Durant – revised 12 October 2015

Tools | Netlist Viewers | RTL Viewer

Types

- bit
- std_logic (IEEE 9-level equivalent): '0', '1', 'Z', 'x' (invalid), 'u' (unknown), …

```
library ieee;
use ieee.std_logic_1164.all;
```

- type NAME is (ITEM1, ITEM2, …);
- Besides these "enumeration" types, VHDL has the integer types integer, natural, and positive.
- Everything above is a VHDL "discrete" type.
- VHDL also has "array" types including bit_vector and std_logic_vector
  - downto
- VHDL has many other types that we don't need in CE1910/1911/1921.

Values

- Single bit: '1' or '0'
- String of bits for vectors: B"11010", X"7E", O"46", 8b"10" (leading 0s)
- Bitfield syntax
  - D <= (position => '1', position => '1', OTHERS => '0'); -- position can be to/downto
  - Y <= (x(3 downto 1), q(4 to 6), 6b"110"); -- 12-bit value

Entity declarations

```
entity ENAME is
      generic (GNAME: type := value); -- separate multiple with ;
      port (PNAME : in/out type; …);
end entity ENAME; -- or just end;
```

Architecture, before begin

signal NAME [, NAME2…] : type;

Attributes (work in Quartus 15, not 100% portable)

- attribute keep: boolean; -- allow showing internal node in simulation, Node Finder with "Post-synthesis" filter
  - attribute keep of keep_wire: signal is true;
- attribute chip_pin : string; -- alternative to graphical Pin Planner
  - attribute chip_pin of data : signal is "D1, D2, D3, D4";
- attribute enum_encoding : string; -- for CE1911
  - attribute enum_encoding of fruit : type is "gray"; -- or "johnson", "one-hot", …

Conditions

- INPUT_OR_SIGNAL = VALUE (can omit in simple cases, e.g., s = '1')
- INPUT_OR_SIGNAL /= VALUE
- ((COND1 or not COND2) and COND3)

Combinational syntax

- Direct assignments: not, and, or, nand, nor, xor, xnor

- `with S select D <= O1 when I1, [O2 when I2 | I3,] [O3 when others];`
  - must cover all
  - The type of S/In can be bit/std_logic, string, and user types (e.g., states)

- `D <= V1 when COND1 [else V2 when COND2]… [else VN];`
  - Caution: infers latches if not all cases clearly covered – check RTL viewer
  - When-else is often good for next-state logic, which is combinational, but whose conditions vary in structure (unconditional transitions only depend on previous state – others look at inputs)

Structural syntax

1.
```
component DFF port ( -- in architecture before begin (no “is”)
      CLK, CLRN, D, PRN: in std_logic;
      Q: out std_logic
);
end component DFF;
```
2. Port map (positional and named associations) (generic map follows same pattern)

```
U1: DFF port map(CLK, ‘1’, D(1), ‘1’, Q(1));
U2: DFF port map(D=>D(2), Q=>Q(2), PRN=>‘1’, CLRN=>‘1’, CLK=>CLK);
```
3.
```
LABEL: for CTR in VAL1 to|downto VAL2 generate
    CONCURRENT_STMTS
end generate LABEL;
```

Sequential syntax for CE1911

- process (sensitivity list)
- rising_edge
- if-else

```
if COND then
      SEQ_STMTS
[elsif COND then
      SEQ_STMTS]*
 [else SEQ_STMTS]
end if;
```

- case-when

```
case EXP is
      when CHOICE1 =>
            SEQ_STMTS […]
      [when OTHERS => SEQ_STMTS]
end case;
```