Guideline for VHDL source code design

Feb 24, 1997 Tetsuji Oguchi

(1) Library to be referred

Use standardized library only as shown below. Eliminate using a kind of dialects.

Example:

```
library ieee, compass_lib;
  use ieee.std_logic_1164.all;
  use compass_lib.compass.all;
```

(2) Format

To make the source code reading easier, use the following format.

(a) Maximum number of characters per row must be smaller than 80.
(b) Position of ":" for "port", "signal", and so forth must be at column 40.
(c) The number of characters per tabulation must be 8.
(d) For output signals, use "buffer" instead of regular "out" except top-level module.
(e) As signal types use "std_logic" or "std_logic_vector".
(f) Eliminate type conversion descriptions such as "to_std_logic( )", if it is possible.
(g) Describe bus width from larger number to smaller number such as "15 downto 8".
(h) Use "space (blank)" and "tab" efficiently.

Example:

```
entity pru is
  port (ppd                          : in     std_logic_vector(15 downto 0);
        sela_r0, sela_r1, sela_r2    : in     std_logic;
        selb_r0, selb_r1, selb_r2    : in     std_logic;
        wr_r0, wr_r1, wr_r2, wr_r3   : in     std_logic;
        nreset, clk                  : in     std_logic;
        a_r0, b_r0                   : buffer std_logic_vector(15 downto 0));
end pru;
```

(3) Selectors

Example:

```
signal  sela_rr0                     : std_logic_vector( 7 downto 0);

begin
sela_rr0 <= (sela_r7, sela_r6, sela_r5, sela_r4, sela_r3, sela_r2, sela_r1, sela_r0);

with sela_rr0 select
a_r0 <= r0 when "00000001",
        r1 when "00000010",
        r2 when "00000100",
        r3 when "00001000",
        r4 when "00010000",
        r5 when "00100000",
        r6 when "01000000",
        r7 when "10000000",
        "0000000000000000" when others;
```

(4) Decoders

Example:

```
signal com_reg                          : std_logic_vector(7 downto 0);

com_decode: process (com_reg)
begin
case com_reg is
  when "00000000" =>   read_1_1 <= '1';
  when others     =>   read_1_1 <= '0';
end case;

case com_reg is
  when "00000001" =>   read_1_2 <= '1';
  when others     =>   read_1_2 <= '0';
end case;

case com_reg is
  when "01010000" =>   read_2   <= '1';
  when others     =>   read_2   <= '0';
end case;
end process com_decode;
```

(5) DFFs

Focus on each DFF and specify the clock and inputs.

Example:

```
reg: process (clk, nreset)
begin
if (nreset = '0')     then  r0 <= "0000000000000000";
elsif ((clk = '1') and clk'event) then
  if ((wr_r0) = '1') then  r0 <= ppd(15 downto 0);
                     else  r0 <= r0;
  end if;
end if;
if (nreset = '0')     then  r1 <= "0000000000000000";
elsif ((clk = '1') and clk'event) then
  if ((wr_r1) = '1') then  r1 <= ppd(15 downto 0);
                     else  r1 <= r1;
  end if;
end if;
end process reg;
```

(6) RAMs

Example:

```
type ramarray is array((2**10 - 1) downto 0) of std_logic_vector(7 downto 0);

signal  ram                             : ramarray;
signal  adr                             : integer range (2**10 - 1) downto 0;

begin
adr    <= to_integer(ain);
dout   <= ram(adr);
```

(7) ROMs

In order to make the ROM file, utility written in C is helpful.

Example:

```
entity ata_prom is
  port (din                              : in     std_logic_vector( 9 downto 0);
        dout                             : buffer std_logic_vector(23 downto 0));
end ata_prom;

architecture arc_1 of ata_prom is
begin
with din select

--nnnnn d          d
--wddcc a          d
--aiiss [          [
--ioo10210111111
--trw   ] 5432109876543210
--                ]
dout <=
 "111110000000000000000000" when "0000000000",
 "101011100000000000000000" when "0000000001",    -- read Alternate Status
 "101100000000000000000000" when "0000000010",    -- read Data
 "101100010000000000000000" when "0000000011",    -- read Error
 "101100100000000000000000" when "0000000100",    -- read Sector Count
 "101100110000000000000000" when "0000000101",    -- read Sector Number
 "101101000000000000000000" when "0000000110",    -- read Cylinder Low
 "101101010000000000000000" when "0000000111",    -- read Cylinder High
 "101101100000000000000000" when "0000001000",    -- read Device/Head
 "101101110000000000000000" when "0000001001",    -- read Status
 "110011101010010101011010" when "0000001010",    -- write Device Control
 "110100010101101010100101" when "0000001011",    -- write Features
-------------------------------------------------------------------------
 "101100000000000000000000" when "1000111101",    -- read  Data
 "101100000000000000000000" when "1000111110",    -- read  Data
 "101100000000000000000000" when "1000111111",    -- read  Data
 "111110000000000000000000" when others;
```