Describing some common structures using VHDL

Madhav Desai

February 13, 2017

VHDL code structure: libraries

Use libraries: std, ieee (many others).

Useful types: std_ulogic, std_logic

Defined in library ieee, package std_logic_1164.

```
library ieee;
package std_logic_1164 is
  -- lots of stuff..

   TYPE std_ulogic IS ( 'U',  -- Uninitialized
                        'X',  -- Forcing  Unknown
                        '0',  -- Forcing  0
                        '1',  -- Forcing  1
                        'Z',  -- High Impedance
                        'W',  -- Weak     Unknown
                        'L',  -- Weak     0
                        'H',  -- Weak     1
                        '-'   -- Don't care
                      );
  SUBTYPE std_logic IS resolved std_ulogic;
end package;
```

Building blocks

We will describe some building blocks and how simple building blocks can be assembled to make more complex blocks.

Inverter

```
library ieee;
use ieee.std_logic_1164.all;
entity inverter is
  port (a: in std_ulogic;
        b: out std_ulogic);
end entity inverter;
architecture Behave of inverter is
begin
  b <= not a;
end Behave;
```

Component package

```
library ieee;
use ieee.std_logic_1164.all;
package EE224_Components is
  component inverter is
     port (a: in std_ulogic;
           b: out std_ulogic);
  and component inverter;
end package;
```

AND2 gate

```
library ieee;
use ieee.std_logic_1164.all;
entity and2 is
  port (a, b: in std_ulogic;
        c: out std_ulogic);
end entity and2;
architecture Behave of and2 is
begin
  c <= a and b;
end Behave;
```

Add AND2 to component package

```
library ieee;
use ieee.std_logic_1164.all;
package EE224_Components is
  component inverter is
     port (a: in std_ulogic;
             b: out std_ulogic);
  and component inverter;
  component and2 is
     port (a, b: in std_ulogic;
         c: out std_ulogic);
  and component and2;
end package;
```

NAND2 gate constructed out of INVERTER, AND2

```
library ieee;
use ieee.std_logic_1164.all;
library work;
use work.EE224_Components.all;
entity nand2 is
  port (a, b: in std_ulogic;
        c: out std_ulogic);
end entity nand2;
architecture Struct of nand2 is
  signal TMP: std_ulogic;
begin
  agate: AND2 port map (a => a,
                        b => b,
                        c => TMP);
  inv: INVERTER port map (a => TMP,
    b => c);
end Behave;
```

Add NAND2 to Components

```
library ieee;
use ieee.std_logic_1164.all;
package EE224_Components is
  component inverter is
     port (a: in std_ulogic;
             b: out std_ulogic);
  and component inverter;
  component and2 is
     port (a, b: in std_ulogic;
         c: out std_ulogic);
  and component and2;
  component nand2 is
     port (a, b: in std_ulogic;
         c: out std_ulogic);
  and component nand2;
end package;
```

XOR-gate using NAND2's

```
library ieee;
use ieee.std_logic_1164.all;
library work;
use work.EE224_Components.all;
entity xor2 is
  port (a, b: in std_ulogic;
        c: out std_ulogic);
end entity xor2;
architecture Struct of nand2 is
  signal U,V,W: std_ulogic;
begin
  n1: NAND2 port map (a => a, b => b, c => U);
  n2: NAND2 port map (a => a, b => U, c => V);
  n3: NAND2 port map (a => b, b => U, c => W);
  n4: NAND2 port map (a => V, b => W, c => c);
end Behave;
```

What will be the instance hierarchy?

```
xor2
   n1
     agate (1 driver)
     inv  (1 driver)
   n2
     agate (1 driver)
     inv  (1 driver)
   n3
     agate (1 driver)
     inv  (1 driver)
   n4
     agate (1 driver)
     inv  (1 driver)
```

Multiplexor

```
library ieee;
use ieee.std_logic_1164.all;
use work.EE224_Components.all;
entity mux2 is
  port (a, b, sel: in std_ulogic;
        c: out std_ulogic);
end entity mux2;
architecture Behave of mux2 is
begin
  c <= a when (sel = '1') else b;
end Behave;
```

2-4 Decoder

```
library ieee;
use ieee.std_logic_1164.all;
use work.EE224_Components.all;
entity decoder2x4 is
  port (s1, s0: in std_ulogic;
        d3,d2,d1,d0: out std_ulogic);
end entity decoder2x4;
architecture Behave of decoder2X4 is
begin
  d3 <= s1 and s0
  d2 <= s1 and (not s0);
  d1 <= (not s1) and s0;
  d0 <= (not s1) and (not s0);
end Behave;
```

Positive Level Triggered Latch

```
library ieee;
use ieee.std_logic_1164.all;
use work.EE224_Components.all;
entity positive_d_latch is
  port (d, clk: in std_ulogic; q: out std_ulogic);
end entity positive_d_latch;
architecture Equations of positive_d_latch is
   signal qsig: std_logic;
begin
   qsig    <= (d and clk) or (qsig and (not clk));
   q <= qsig;
end Equations;
```

Negative Level Triggered Latch

```
library ieee;
use ieee.std_logic_1164.all;
use work.EE224_Components.all;
entity negative_d_latch is
  port (d, clk: in std_ulogic; q: out std_ulogic);
end entity negative_d_latch;
architecture Equations of negative_d_latch is
   signal qsig: std_logic;
begin
   qsig    <= (d and (not clk)) or (qsig and clk);
   q <= qsig;
end Equations;
```

Positive Edge-triggered Flip-flop

```
library ieee;
use ieee.std_logic_1164.all;
use work.EE224_Components.all;
entity DFF is
  port (d, clk: in std_ulogic; q: out std_ulogic);
end entity DFF;
architecture Struct of DFF is
   signal U: std_logic;
begin
   master: negative_d_latch
            port map (d => d, clk => clk, q => U);
   slave: positve_d_latch
            port map (d => U, clk => clk, q => q);
end Struct;
```

Reality check

Can we routinely describe any combinational circuit in VHDL?

- Either by transcribing Boolean formulas using concurrent assignments.
- Or by instantiating gate level network.
- Or a combination of the two.
- Or..

Useful reading material

- D. Perry, "VHDL Programming by Example", Tata McGraw-Hill.
- P. Ashenden, "A VHDL Tutorial", hep.uchicago.edu/ tangjian/SVT_sub/FTK_ATLAS/.../vhdl-tutorial.pdf.