Structural Descriptions Using VHDL

Madhav Desai

January 19, 2018

A structural implementation of the 2-bit adder

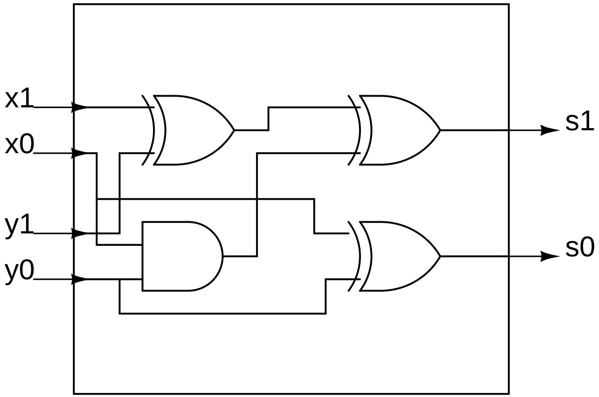

Figure: Structure

Instance Hierarchy

```
TwoBitAdder
     x1 (XOR2)
     x2 (XOR2)
     a1 (AND2)
     x3 (XOR2)
```

Structural description of the two-bit adder: the entity stays the same

```
entity TwoBitAdder is
   port(x1,x0,y1,y0: in bit;
        s1,s0: out bit);
end entity;
```

VHDL description of the structured architecture

```
architecture Struct of TwoBitAdder is
  signal w, z: bit;
  component XOR2 is
     port (a, b: in std_ulogic;
         c: out std_ulogic);
  and component XOR2;
  component AND2 is
     port (a, b: in std_ulogic;
         c: out std_ulogic);
  and component AND2;
begin
   x_1: XOR2 port map (a => x0, b => y0, c => s0);
   x_2: XOR2 port map (a => x1, b => y1, c => w);
   a_1: AND2 port map (a => x0, b => y0, c => z);
   x_3: XOR2 port map (a => w, b => z, c => s1);
end Struct;
```

What about XOR2, AND2?

- Need to describe entity and architecture for XOR2, AND2.
- Interface must be consistent with components for XOR2, AND2.
- Architecture can be
    - Structural: For example XOR2 can be implemented by using AND2 and NOT.
    - Behavioural: using expressions (formulas) as we did for the first implementation of the two-bit adder.

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
                         'O',  -- 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;
```

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..

A generic test-bench

- Prepare a trace-file which contains a list of input vectors and expected outputs.
- The generic testbench reads the trace-file, applies inputs to the device under test (DUT), and checks the outputs against what is expected.
- The generic testbench can be customized to your specific example.
- You need to produce the trace-file!

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.