

L3.1 VHDL Designs Mon. February 11, 2002

VHDL design descriptions consist of two parts.

The ENTITY declaration describes the I/O.
 The ARCHITECTURE body describes the content.

The ENTITY describes the periphery of the design, i.e., the SIGNAL I/O.
 Both have names (identifiers).

ENTITY names must be unique within the design.
 ARCHITECTURES provide content for the ENTITY.

ARCHITECTURE names need not be unique. They are used to delineate the
 ARCHITECTURE declaration. They likely provide YOU with information.

VHDL is MoStLy case independent.

L3.2 PORTS Mon. February 11, 2002

Entities always use a special signal, a PORT.

PORTs have MODEs and TYPEs. MODEs are:

- IN
- OUT
- INOUT – A tri-state signal.
- BUFFER – An output which is also used internally and has a limited fan-out.

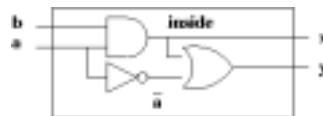
We will not use mode BUFFER. This will make it easier to use entities in
 hierarchical designs as VHDL is a strongly typed language.

We will declare a signal in the architecture to use and assign the output signal
 to this internal signal.

L3.3 Not Needing Mode BUFFER Mon. February 11, 2002

```
Library ieee;
use ieee.std_logic_1164.all;
entity foo is
    port(a, b: in std_logic;
         x, y: out std_logic);
end foo;
```

```
architecture no_buffer_mode of foo is
    signal inside: std_logic;
begin
    inside <= a AND b;
    x <= inside;
    y <= inside OR (not a);
    -- really wanted y <= x OR (not a);
end no_buffer_mode;
```



L3.4 Extract of a Report File (Cypress) Mon. February 11, 2002

Compiling: nobuffer.vhd

DESIGN EQUATIONS (11:07:14)

$$x = a * b$$

$$/y = a * /b$$

C16V8A

b = 1	20 * not used
a = 2	19 = y
not used * 3	18 * not used
not used * 4	17 * not used
not used * 5	16 * not used
not used * 6	15 * not used
not used * 7	14 * not used
not used * 8	13 * not used
not used * 9	12 = x
not used * 10	11 * not used

L3.5 Types Mon. February 11, 2002

We will always use types

```
STD_LOGIC
STD_LOGIC_VECTOR
```

They are industry standard and are used when tri-state logic is required.
 These types require the statements

```
LIBRARY ieee;
use ieee.std_logic_1164.all;
```

STD_LOGIC types include

```
U Uninitialized
X Unknown
0 Zero
1 One
Z Tristate (must be UPPERCASE!)
W Weak unknown
L Weak 0
H Weak 1
- Don't care
```

L3.6 Entity Example Mon. February 11, 2002

```
ENTITY black_box IS PORT
(clk, rst : IN std_logic;
 d       : IN std_logic_vector(7 DOWNTO 0);
 q       : OUT std_logic_vector(7 DOWNTO 0);
 co      : OUT std_logic);
END black_box;
```

Note that the entity has MORE information than the drawing, namely, the type of the signals!



L3.7 Designs Mon. February 11, 2002

Designs consist of an entity/architecture pair.

They are usually in the same file. This is a good idea!

A file may have multiple entity/architecture pairs;

however, one should declare entity/architecture pairs before they are used in another architecture.

Architectures can have two types of statements.

```
CONCURRENT
SEQUENTIAL
```

-- Comments start with a double hyphen.

L3.8 Example: Half Adder Mon. February 11, 2002

-- This comment is before the library and use clauses.

```
library ieee;
use ieee.std_logic_1164.all;
-- here is the entity
entity halfadd is
port (a, b : in std_logic;
      sum, c : out std_logic);
end halfadd;
```

architecture comp of halfadd is

begin

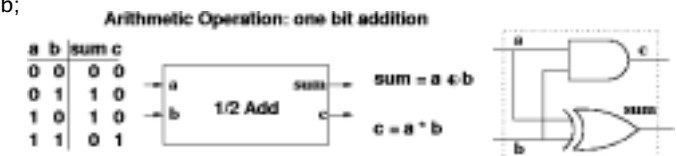
-- a concurrent statement implementing the and gate

c <= a and b;

-- a concurrent statement implementing the xor gate

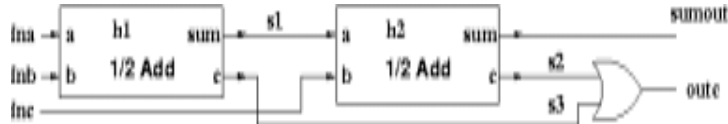
sum <= a xor b;

end comp;



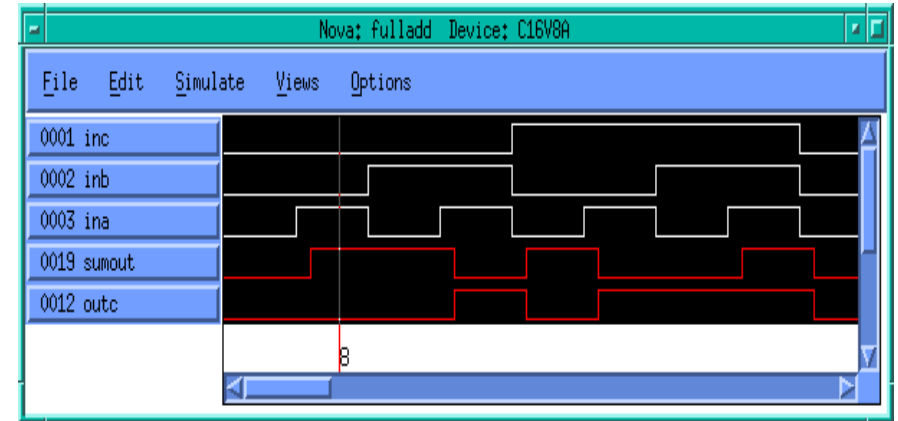
L3.9 Example: Full Adder Mon. February 11, 2002

```
library ieee;
use ieee.std_logic_1164.all;
entity fulladd is
  port (ina, inb, inc : in std_logic;
        sumout, outc : out std_logic);
end fulladd;
architecture top of fulladd is
  component halfadd
    port (a, b : in std_logic;
          sum, c : out std_logic);
  end component;
  signal s1, s2, s3 : std_logic;
begin
  -- a structural instantiation of two half adders
  h1: halfadd port map( a => ina, b => inb,
                      sum => s1, c => s3);
  h2: halfadd port map( a => s1, b => inc,
                      sum => sumout, c => s2);
  outc <= s2 or s3;
end top;
```



L3.10 Demo of Galaxy Mon. February 11, 2002

Do a demo of galaxy.
One should get a simulation window like this.



L3.11 Attributes (Cypress) Mon. February 11, 2002

Attributes provide information about VHDL constructs such as

- Entities
- Architectures
- Types
- Signals

Pin_numbers map external signals to specific pins.

Pin_avoid means to not use specific pins.

See the xxx.vhd files in /mit/6.111/cpld/sources/ for guidance in choosing pins and/or avoiding pins.

L3.12 Easy way to Assign Pins (Cypress) Mon. February 11, 2002

Don't assign pins first.

Let galaxy pick them and wire to those pins.

Your design doesn't work the first time!

Find out the pins and put them in to avoid rewiring.

Or click on Files->Annotate

After a pop up, this produces an xxx.ctl file which then is used along with xxx.vhd.

Be careful not to put a pin number in here which conflicts with a pin_avoid attribute in your xxx.vhd file.

L3.13 Example – Pin_avoid (Cypress) Mon. February 11, 2002

```
library ieee;
use ieee.std_logic_1164.all;
entity fulladd is
  port (ina, inb, inc : in std_logic;
        sumout, outc : out std_logic);
  ATTRIBUTE pin_avoid of fulladd :ENTITY is
    " 19 " &
    " 12 ";
end fulladd;
```

Note that a particular attribute can be used only once for a VHDL object.

The & means concatenation.
The pin number list is "space separated".

L3.14 Example xx.ctl File (Cypress) Mon. February 11, 2002

```
Attribute PIN_NUMBERS of Reserved2 is "19" ;
Attribute PIN_NUMBERS of outc is "14" ;
Attribute PIN_NUMBERS of sumout is "13" ;
Attribute PIN_NUMBERS of Reserved1 is "12" ;
Attribute PIN_NUMBERS of ina is "3" ;
Attribute PIN_NUMBERS of inb is "2" ;
Attribute PIN_NUMBERS of inc is "1" ;
```

Note that, unlike xxx.vhd files, multiple use of a particular attribute is allowed.

L3.15 VHDL Statements Mon. February 11, 2002

Concurrent

Signal assignment
Instantiation
when/else (more later)
with/select (more later)
process (as a wrapper for sequential statements)

Sequential – ONLY within a process

Signal assignment
if/then/elsif/else
case/when

L3.16 Optimization Mon. February 11, 2002

```
-- z = ((/a + c) * (a + /b) * (a + /c))
library ieee;
use ieee.std_logic_1164.all;
entity comb is
  port (a, b, c : in std_logic;
        z : out std_logic);
end comb;

architecture sf of comb is
  signal t1, t2, t3 : std_logic;
begin
  z <= not (t1 and t2 and t3);
  t1 <= (not a) or c;
  t2 <= a or (not b);
  t3 <= a or (not c);
end sf;
```



L3.17 Using DeMorgan's Theorem Mon. February 11, 2002

```
-- z = /((/a + c) * (a + /b) * (a + /c))  
-- by DeMorgan's Theorem  
-- z = (a * /c) + (/a * b) + (/a * c)  
-- z = (a * /c) + (/a * (b + c))
```

```
library ieee;  
use ieee.std_logic_1164.all;  
entity comb is  
  port (a, b, c : in std_logic;  
        z : out std_logic);  
end comb;
```

```
architecture dm of comb is  
  signal t1, t2, t3 : std_logic;  
begin  
  z <= (a and (not c)) or ((not a) and (b or c));  
end dm;
```

```
Compiling: sf.vhd  
DESIGN EQUATIONS  
  /z = /a * /b * /c + a * c  
Compiling: dm.vhd  
DESIGN EQUATIONS  
  /z = /a * /b * /c + a * c
```