

### 3.1 VHDL Example

Monday, February 12, 2001

```
-- Massachusetts (Obsolete) Stoplight Example
library ieee;
use ieee.std_logic_1164.all;
entity check is port(
  r, y, g: in std_logic;
  ok: out std_logic);
end check;
architecture logical of check is
  signal t1, t2, t3: std_logic;
begin
  t1 <= r and (not g);
  t2 <= y and (not g);
  t3 <= (not r) and (not y) and g;
  ok <= t1 or t2 or t3;
end logical;
```



### 3.2 VHDL Designs

- 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.

### 3.3 PORTS

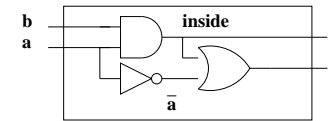
Entities always use a special signal, a PORT.

- 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.

### 3.4 Not Needing Mode BUFFER

```
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;
```



### 3.5 Extract of Report File

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 |

### 3.6 TYPES

- We will always use types
  - o STD\_LOGIC
  - o STD\_LOGIC\_VECTOR
- They are industry standard and are used when tri-state logic is required. These types require the statements

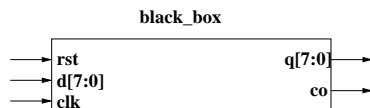
```
LIBRARY ieee;
use ieee.std_logic_1164.all;
```

- STD\_LOGIC types include
  - o U Uninitialized
  - o X Unknown
  - o 0 Zero
  - o 1 One
  - o Z Tristate (must be UPPERCASE!)
  - o W Weak unknown
  - o L Weak 0
  - o H Weak 1
  - o - Don't care

### 3.7 ENTITY Example

```
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!



### 3.8 Designs

- 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.
- Architectures can have two types of statements. We defer this to the next VHDL lecture.
  - o CONCURRENT
  - o SEQUENTIAL

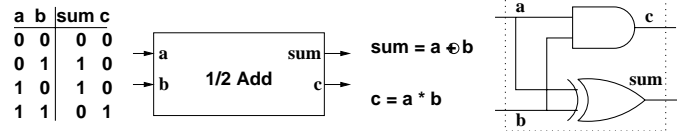
-- Comments start with a double hyphen.

### 3.9 Example: Half Adder

```
-- 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;

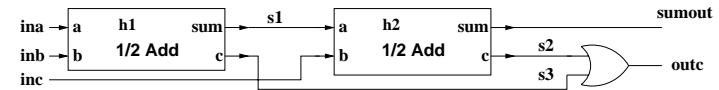
architecture comp of halfadd is
begin
  -- a concurrent statement implementing the and gate
  c <= a and b;
  -- a concurrent statement implementing the xor gate
  sum <= a xor b;
end comp;
```

Arithmetic Operation: one bit addition



### 3.10 Example: Full Adder

```
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;
```



### 3.11 DEMO of Galaxy

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



### 3.12 Attributes

- 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.

### 3.13 Easy Way to Assign Pins

- Don't assign pins first.
- Let galaxy pick them and wire to those pins.
- It 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.

### 3.14 Example Using Pin\_avoid Attribute

```
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".

### 3.15 Example xx.ctl File

```
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.

### 3.16 VHDL Statements

- 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

### 3.17 Optimization

```
-- 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;
```



### 3.18 Using DeMorgan's Theorem

```
-- 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

$$/z = /a * /b * /c + a * c$$