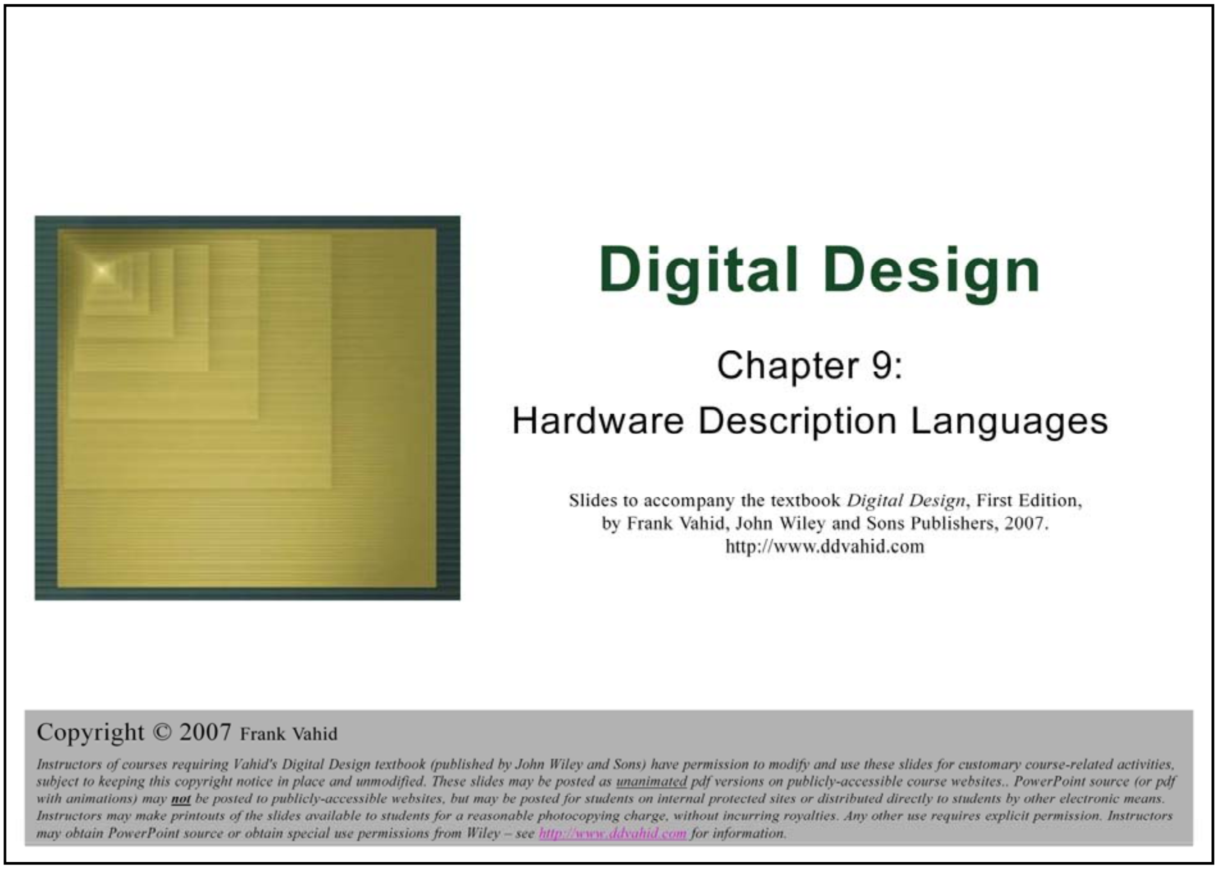
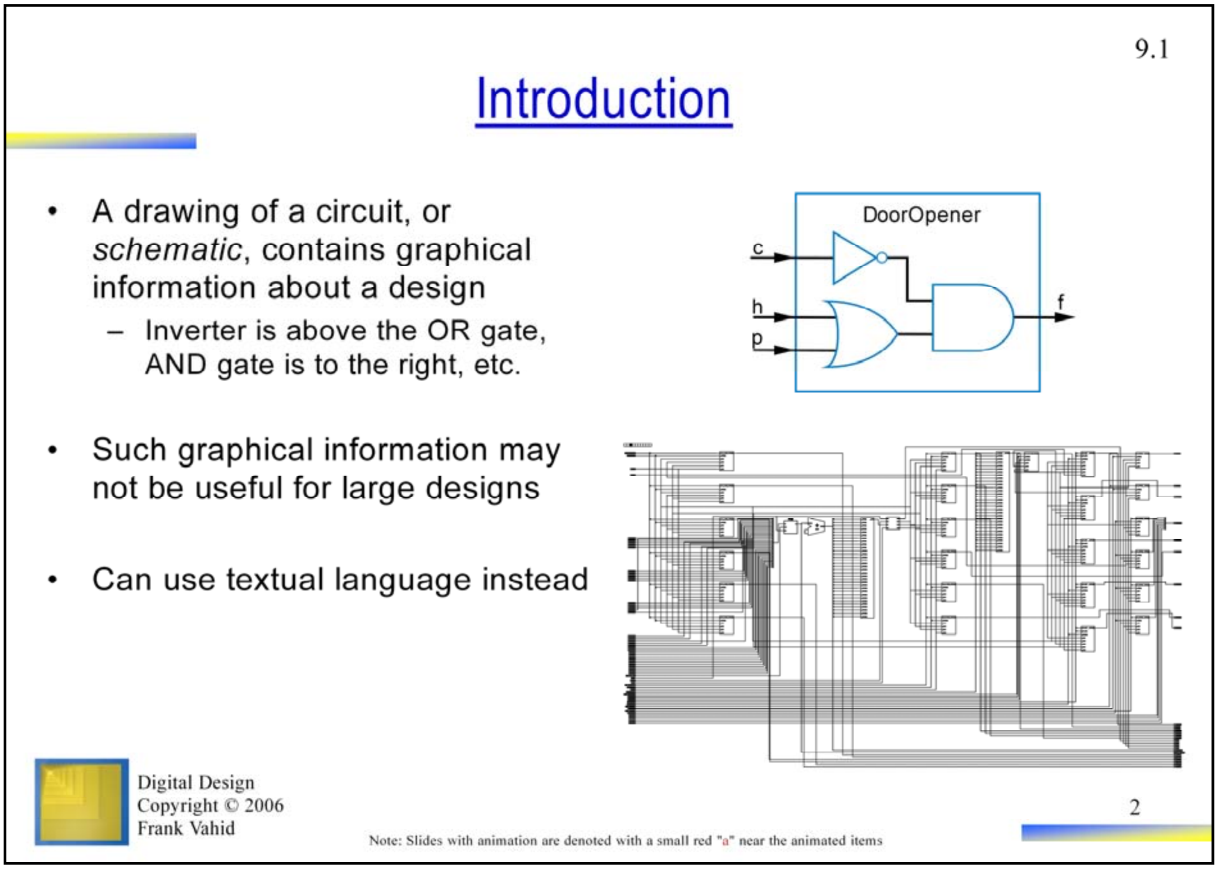


Digital Design

Chapter 9:
Hardware Description Languages

Slides to accompany the textbook *Digital Design*, First Edition,
by Frank Vahid, John Wiley and Sons Publishers, 2007.
http://www.ddvahid.com



Introduction

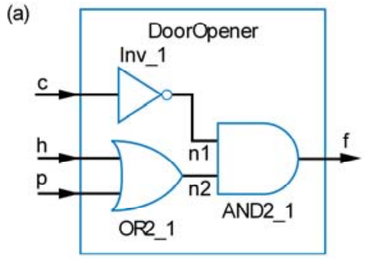
- A drawing of a circuit, or *schematic*, contains graphical information about a design
  - Inverter is above the OR gate, AND gate is to the right, etc.
- Such graphical information may not be useful for large designs
- Can use textual language instead

Note: Slides with animation are denoted with a small red "a" near the animated items

Textual Language – English

- Can describe circuit using English text rather than using a drawing
  - Of course, English isn't a good language for a computer to read
  - Need a more precise, computer-oriented language

(a)

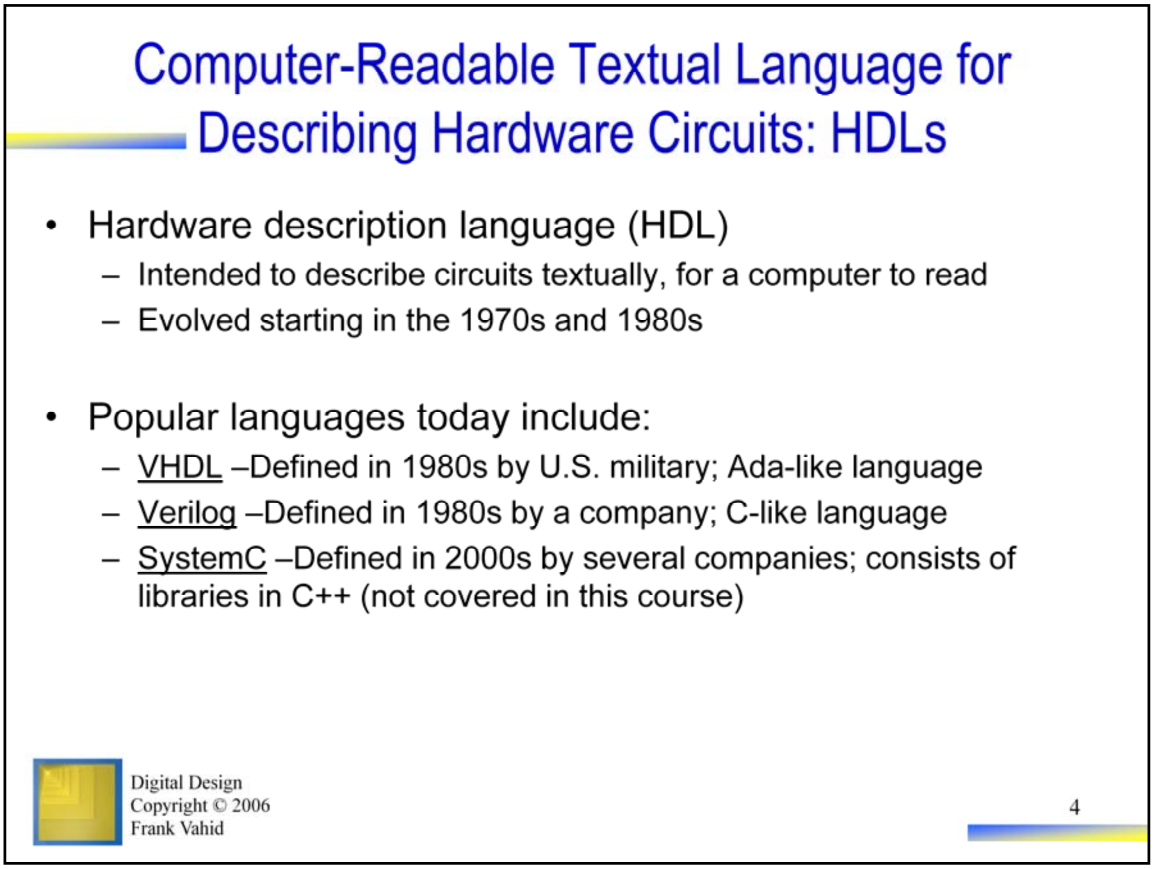


(b)

We'll now describe a circuit whose name is DoorOpener.
The external inputs are c, h and p, which are bits.
The external output is f, which is a bit.

We assume you know the behavior of these components:
An inverter, which has a bit input x, and bit output F.
A 2-input OR gate, which has inputs x and y, and bit output F.
A 2-input AND gate, which has bit inputs x and y, and bit output F.

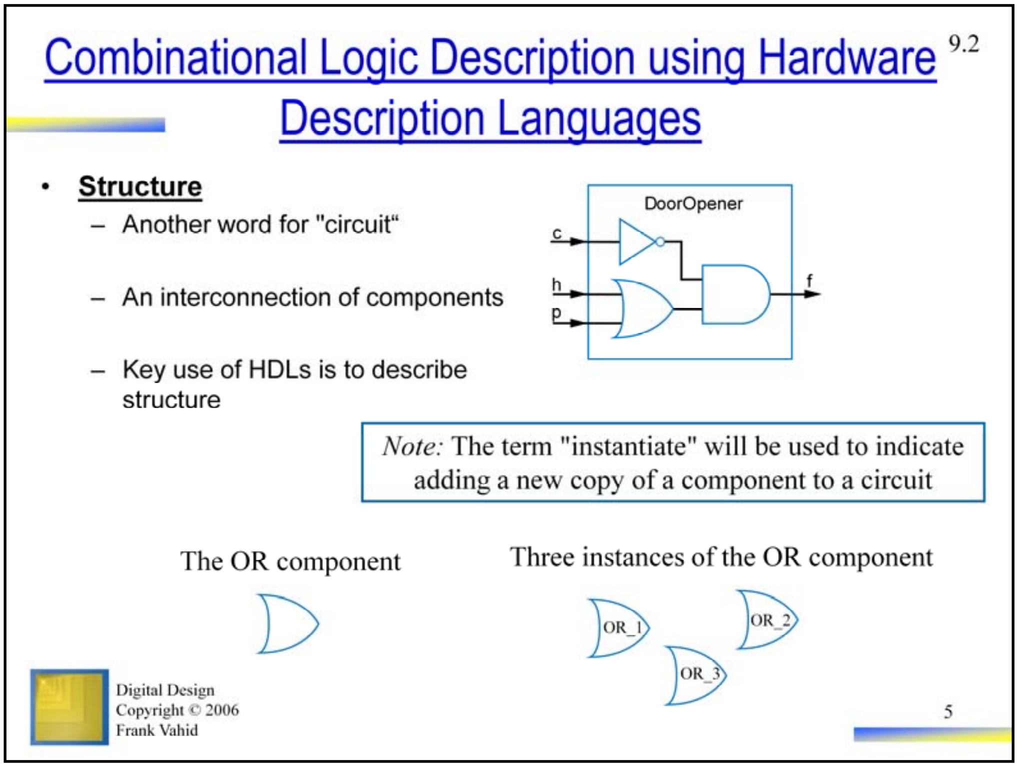
The circuit has internal wires n1 and n2, both bits.
The DoorOpener circuit internally consists of:
An inverter named Inv_1, whose input x connects to external input c, and whose output connects to n1.
A 2-input OR gate named OR2_1, whose inputs connect to external inputs h and p, and whose output connects to n2.
A 2-input AND gate named AND2_1, whose inputs connect to n1 and n2, and whose output connects to external output f.
That's all.



Computer-Readable Textual Language for Describing Hardware Circuits: HDLs

- Hardware description language (HDL)
  - Intended to describe circuits textually, for a computer to read
  - Evolved starting in the 1970s and 1980s

- Popular languages today include:
  - VHDL –Defined in 1980s by U.S. military; Ada-like language
  - Verilog –Defined in 1980s by a company; C-like language
  - SystemC –Defined in 2000s by several companies; consists of libraries in C++ (not covered in this course)



Combinational Logic Description using Hardware Description Languages

9.2

- **Structure**
  - Another word for "circuit"
  - An interconnection of components
  - Key use of HDLs is to describe structure

DoorOpener
c
h
p
f

Note: The term "instantiate" will be used to indicate adding a new copy of a component to a circuit

The OR component

Three instances of the OR component

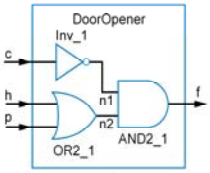
OR_1
OR_2
OR_3

Digital Design
Copyright © 2006
Frank Vahid

5

Describing Structure in VHDL

- Entity – Defines new item's name & ports (inputs/outputs)
  - std_logic means bit type, defined in ieee library
- Architecture – Describes internals, which we named "Circuit"
  - Declares 3 previously-defined components
  - Declares internal signals
    - Note "--" comment
  - Instantiates and connects those components



(a)

We'll now describe a circuit whose name is DoorOpener.
The external inputs are c, h and p, which are bits.
The external output is f, which is a bit.

We assume you know the behavior of these components:
An inverter, which has a bit input x, and bit output F.
A 2-input OR gate, which has inputs x and y, and bit output F.
A 2-input AND gate, which has bit inputs x and y, and bit output F.

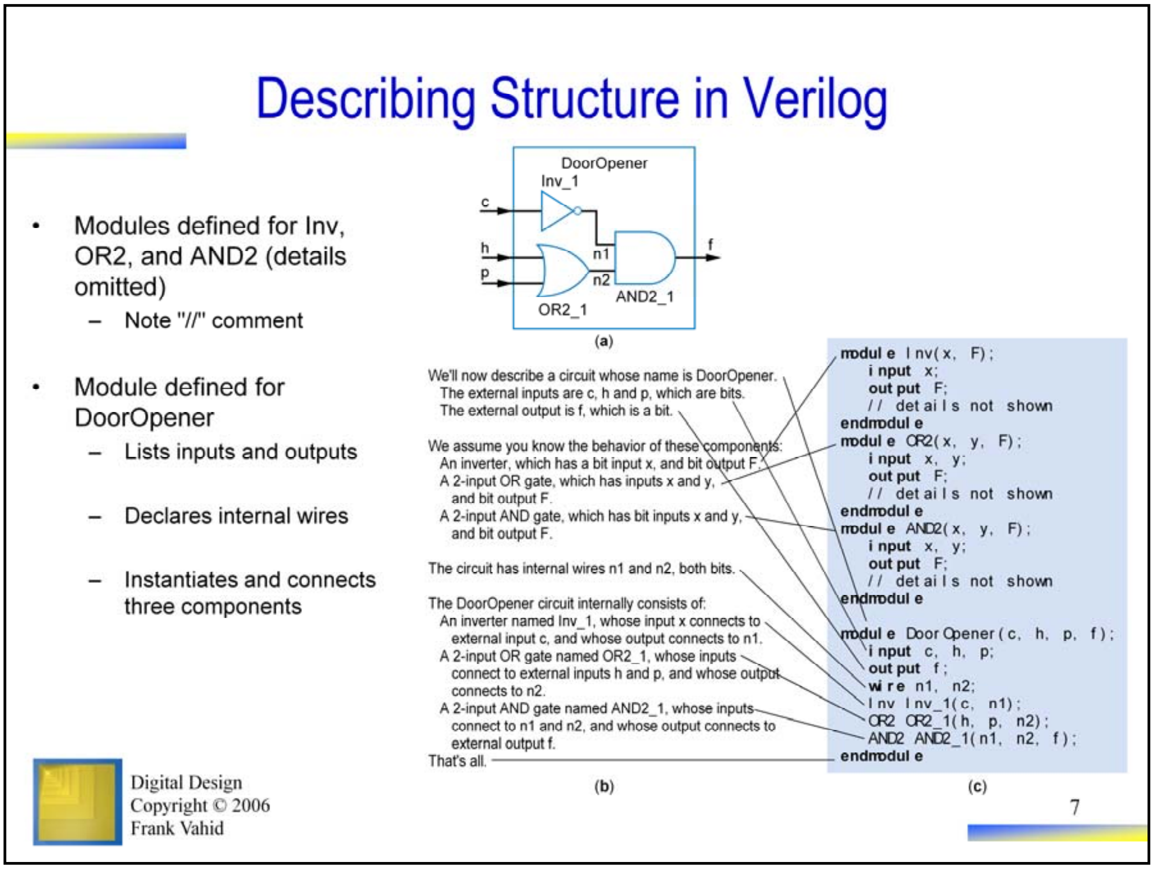
The circuit has internal wires n1 and n2, both bits.

The DoorOpener circuit internally consists of:
An inverter named Inv_1, whose input x connects to external input c, and whose output connects to n1.
A 2-input OR gate named OR2_1, whose inputs connect to external inputs h and p, and whose output connects to n2.
A 2-input AND gate named AND2_1, whose inputs connect to n1 and n2, and whose output connects to external output f.
That's all.

(b)

```
library ieee;
use ieee.std_logic_1164.all;
entity DoorOpener is
  port ( c, h, p: in std_logic;
         f: out std_logic
  );
end DoorOpener;

architecture Circuit of DoorOpener is
  component Inv
    port (x: in std_logic;
          F: out std_logic);
  end component;
  component OR2
    port (x, y: in std_logic;
          F: out std_logic);
  end component;
  component AND2
    port (x, y: in std_logic;
          F: out std_logic);
  end component;
  signal n1, n2: std_logic; --internal wires

begin
  Inv_1: Inv port map (x=>c, F=>n1);
  OR2_1: OR2 port map (x=>h, y=>p, F=>n2);
  AND2_1: AND2 port map (x=>n1, y=>n2, F=>f);
end Circuit;
```

(c)



Describing Structure in Verilog

- Modules defined for Inv, OR2, and AND2 (details omitted)
  - Note "//" comment
- Module defined for DoorOpener
  - Lists inputs and outputs
  - Declares internal wires
  - Instantiates and connects three components

(a)

We'll now describe a circuit whose name is DoorOpener.
The external inputs are c, h and p, which are bits.
The external output is f, which is a bit.

We assume you know the behavior of these components:
An inverter, which has a bit input x, and bit output F.
A 2-input OR gate, which has inputs x and y, and bit output F.
A 2-input AND gate, which has bit inputs x and y, and bit output F.

The circuit has internal wires n1 and n2, both bits.

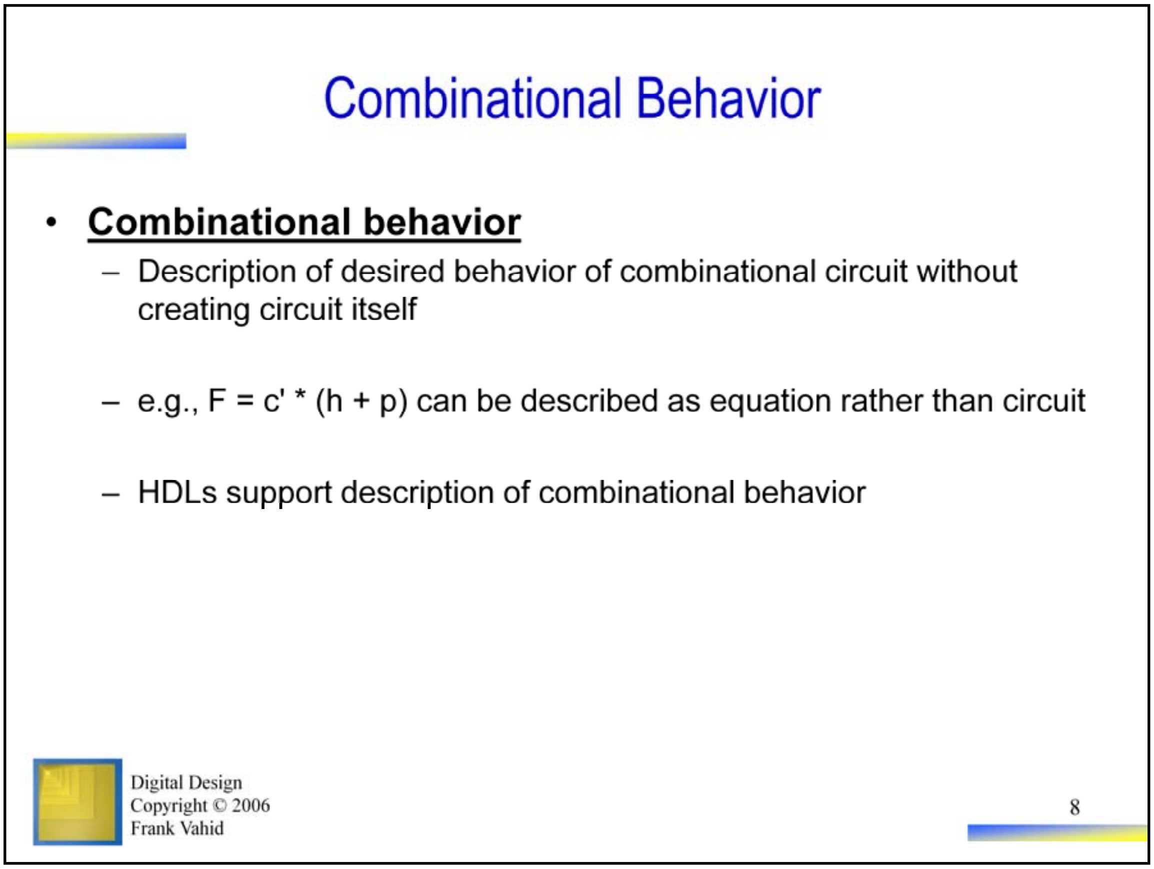
The DoorOpener circuit internally consists of:
An inverter named Inv_1, whose input x connects to external input c, and whose output connects to n1.
A 2-input OR gate named OR2_1, whose inputs connect to external inputs h and p, and whose output connects to n2.
A 2-input AND gate named AND2_1, whose inputs connect to n1 and n2, and whose output connects to external output f.
That's all.

(b)

```
module Inv(x, F);
    input x;
    output F;
    // details not shown
endmodule
module OR2(x, y, F);
    input x, y;
    output F;
    // details not shown
endmodule
module AND2(x, y, F);
    input x, y;
    output F;
    // details not shown
endmodule

module DoorOpener(c, h, p, f);
    input c, h, p;
    output f;
    wire n1, n2;
    Inv Inv_1(c, n1);
    OR2 OR2_1(h, p, n2);
    AND2 AND2_1(n1, n2, f);
endmodule
```

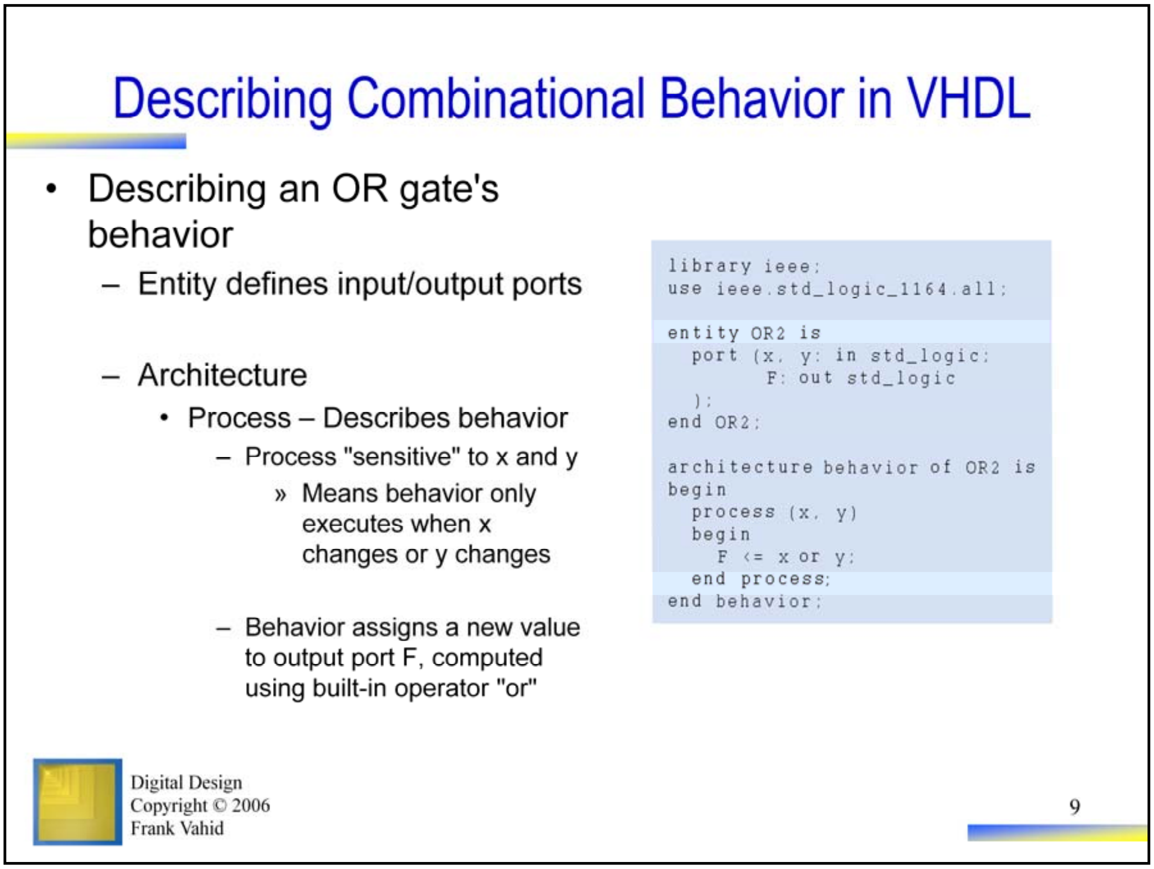
(c)

Digital Design
Copyright © 2006
Frank Vahid



Combinational Behavior

- **Combinational behavior**
  - Description of desired behavior of combinational circuit without creating circuit itself
  - e.g., F = c' * (h + p) can be described as equation rather than circuit
  - HDLs support description of combinational behavior



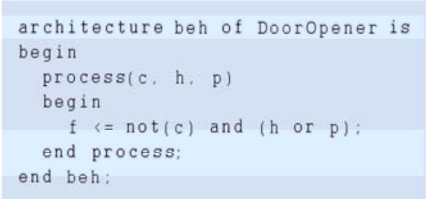
Describing Combinational Behavior in VHDL

- Describing an OR gate's behavior
  - Entity defines input/output ports
  - Architecture
    - Process – Describes behavior
      - Process "sensitive" to x and y
        - » Means behavior only executes when x changes or y changes
      - Behavior assigns a new value to output port F, computed using built-in operator "or"

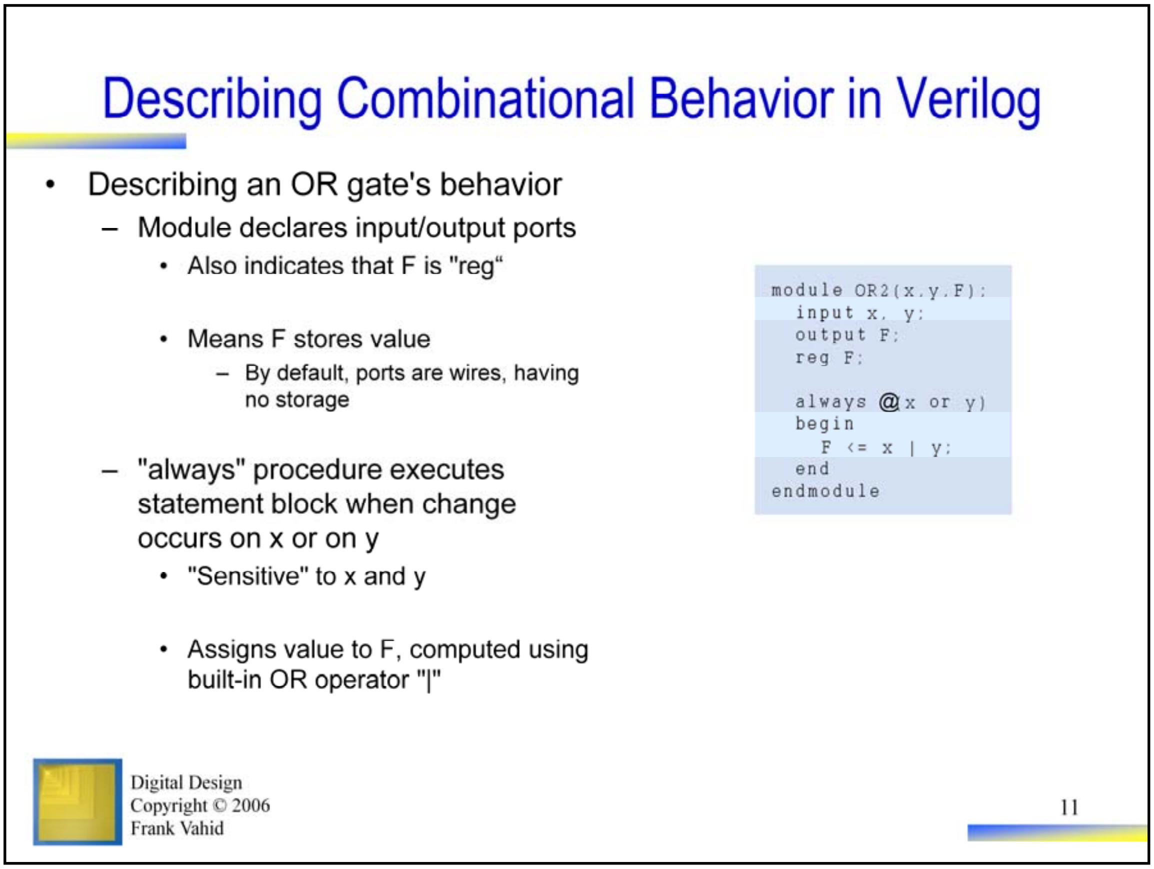
```
library ieee;
use ieee.std_logic_1164.all;

entity OR2 is
  port (x, y: in std_logic;
        F: out std_logic
  );
end OR2;

architecture behavior of OR2 is
begin
  process (x, y)
  begin
    F <= x or y;
  end process;
end behavior;
```

Describing Combinational Behavior in VHDL

- Describing a custom function's behavior
  - Desired function: f = c'*(h+p)
  - Entity defines input/output ports (not shown)
  - Architecture
    - Process
      - Sensitive to c, h, and p
      - Assigns a new value to output port f, computed using built-in operators "not", "and", and "or"



```
architecture beh of DoorOpener is
begin
  process(c, h, p)
  begin
    f <= not(c) and (h or p);
  end process;
end beh;
```



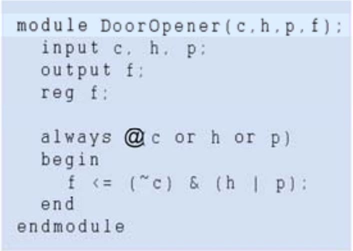
Describing Combinational Behavior in Verilog

- Describing an OR gate's behavior
  - Module declares input/output ports
    - Also indicates that F is "reg“
    - Means F stores value
      - By default, ports are wires, having no storage
  - "always" procedure executes statement block when change occurs on x or on y
    - "Sensitive" to x and y
    - Assigns value to F, computed using built-in OR operator "|"

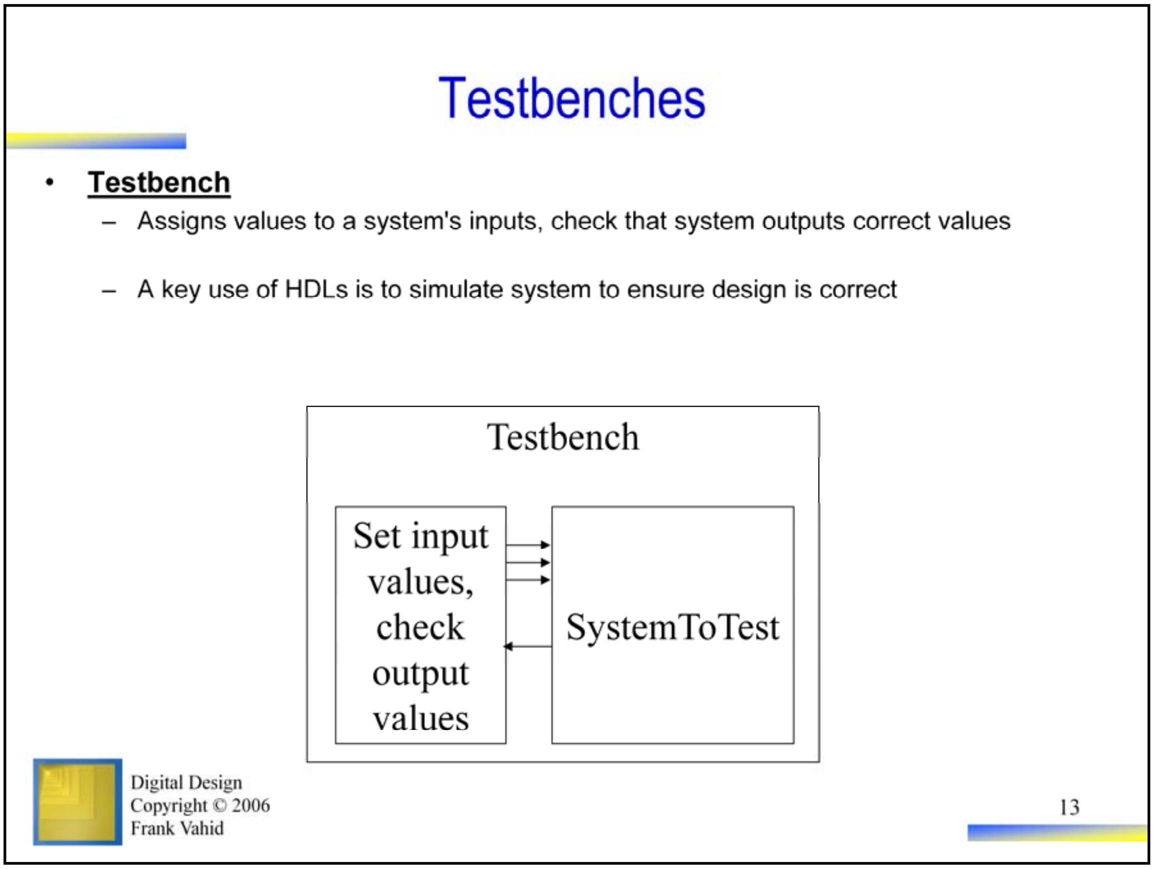
```
module OR2(x,y,F);
   input x, y;
   output F;
   reg F;

   always @(x or y)
   begin
     F <= x | y;
   end
endmodule
```

Describing Combinational Behavior in Verilog

- Describing a custom function's behavior
  - Desired function: f = c'*(h+p)
  - Module defines input/output ports
    - Output f defined as "reg“
  - "always" procedure sensitive to inputs
    - Assigns value to f, computed using built-in operators for NOT (~), AND (&), and OR (|)



```
module DoorOpener(c,h,p,f);
  input c, h, p;
  output f;
  reg f;

  always @(c or h or p)
  begin
    f <= (~c) & (h | p);
  end
endmodule
```



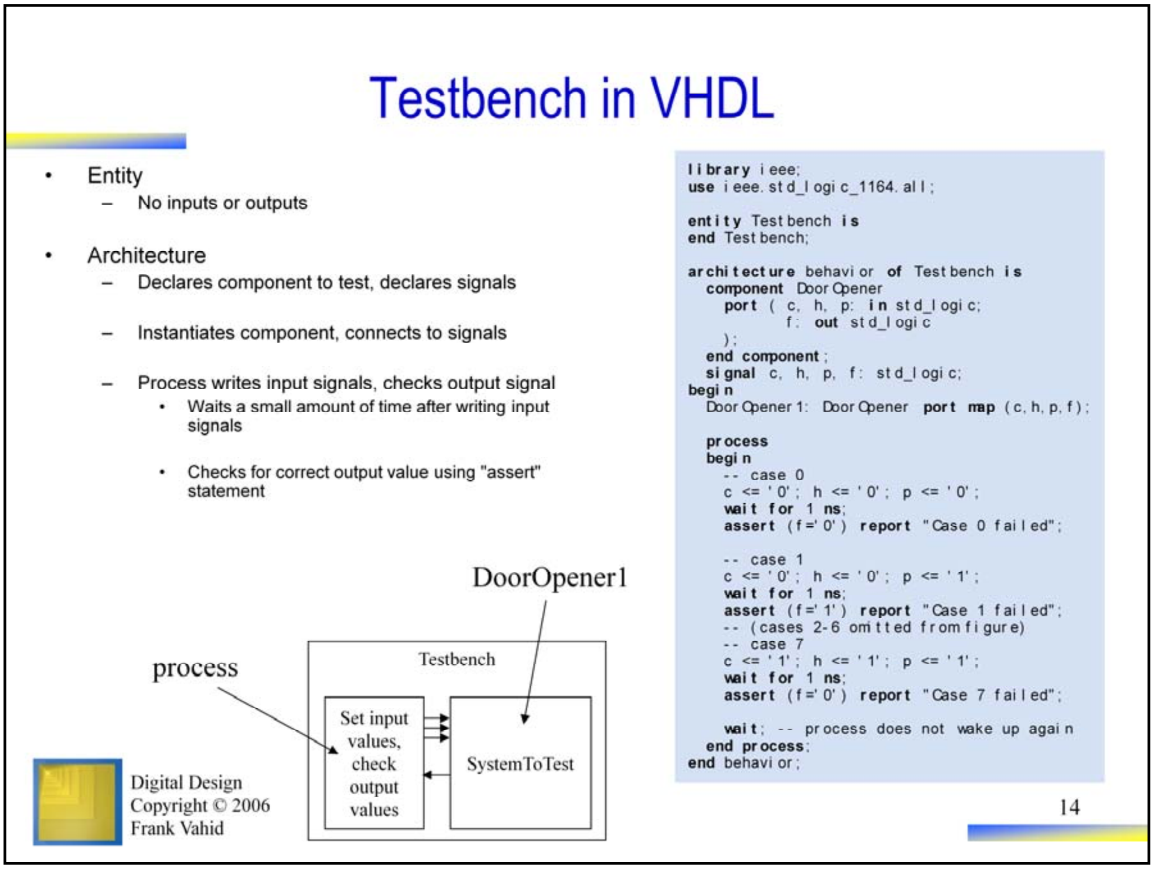
Testbenches

- **Testbench**
  - Assigns values to a system's inputs, check that system outputs correct values
  - A key use of HDLs is to simulate system to ensure design is correct

Testbench

Set input values, check output values

SystemToTest



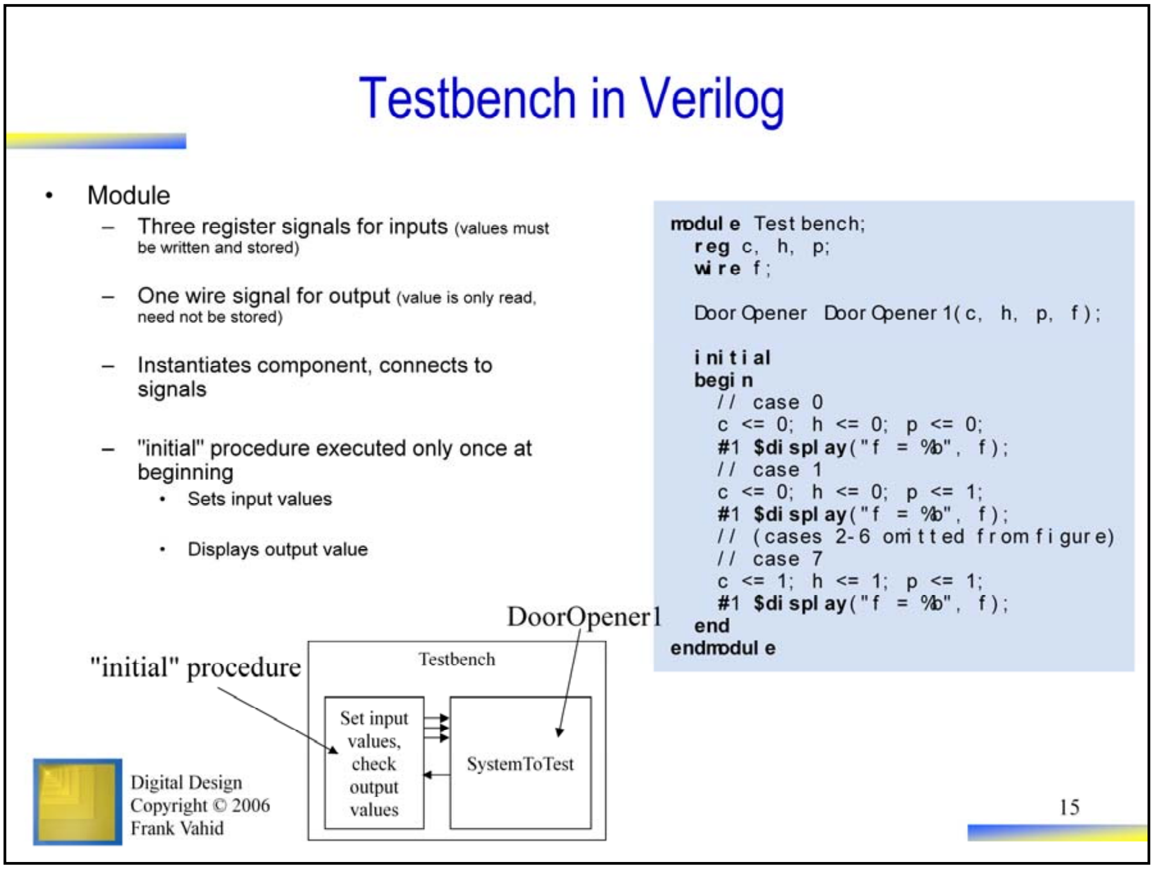
Testbench in VHDL

- Entity
  - No inputs or outputs
- Architecture
  - Declares component to test, declares signals
  - Instantiates component, connects to signals
  - Process writes input signals, checks output signal
    - Waits a small amount of time after writing input signals
    - Checks for correct output value using "assert" statement

DoorOpener1
process
Testbench
Set input values, check output values
SystemToTest

```
library ieee;
use ieee.std_logic_1164.all;

entity Testbench is
end Testbench;

architecture behavior of Testbench is
  component DoorOpener
    port ( c, h, p: in std_logic;
             f: out std_logic
    );
  end component;
  signal c, h, p, f: std_logic;
begin
  DoorOpener1: DoorOpener port map (c,h,p,f);

  process
  begin
    -- case 0
    c <= '0'; h <= '0'; p <= '0';
    wait for 1 ns;
    assert (f='0') report "Case 0 failed";

    -- case 1
    c <= '0'; h <= '0'; p <= '1';
    wait for 1 ns;
    assert (f='1') report "Case 1 failed";
    -- (cases 2-6 omitted from figure)
    -- case 7
    c <= '1'; h <= '1'; p <= '1';
    wait for 1 ns;
    assert (f='0') report "Case 7 failed";

    wait; -- process does not wake up again
  end process;
end behavior;
```

Digital Design
Copyright © 2006
Frank Vahid



Testbench in Verilog

- Module
  - Three register signals for inputs (values must be written and stored)
  - One wire signal for output (value is only read, need not be stored)
  - Instantiates component, connects to signals
  - "initial" procedure executed only once at beginning
    - Sets input values
    - Displays output value

```
module Testbench;
  reg c, h, p;
  wire f;

  DoorOpener DoorOpener1(c, h, p, f);

  initial
  begin
    // case 0
    c <= 0; h <= 0; p <= 0;
    #1 $display("f = %b", f);
    // case 1
    c <= 0; h <= 0; p <= 1;
    #1 $display("f = %b", f);
    // (cases 2-6 omitted from figure)
    // case 7
    c <= 1; h <= 1; p <= 1;
    #1 $display("f = %b", f);
  end
endmodule
```

Digital Design
Copyright © 2006
Frank Vahid