Digital System Design

by
Dr. Lesley Shannon
Email: lshannon@ensc.sfu.ca
Course Website: http://www.ensc.sfu.ca/~lshannon/courses/ensc350

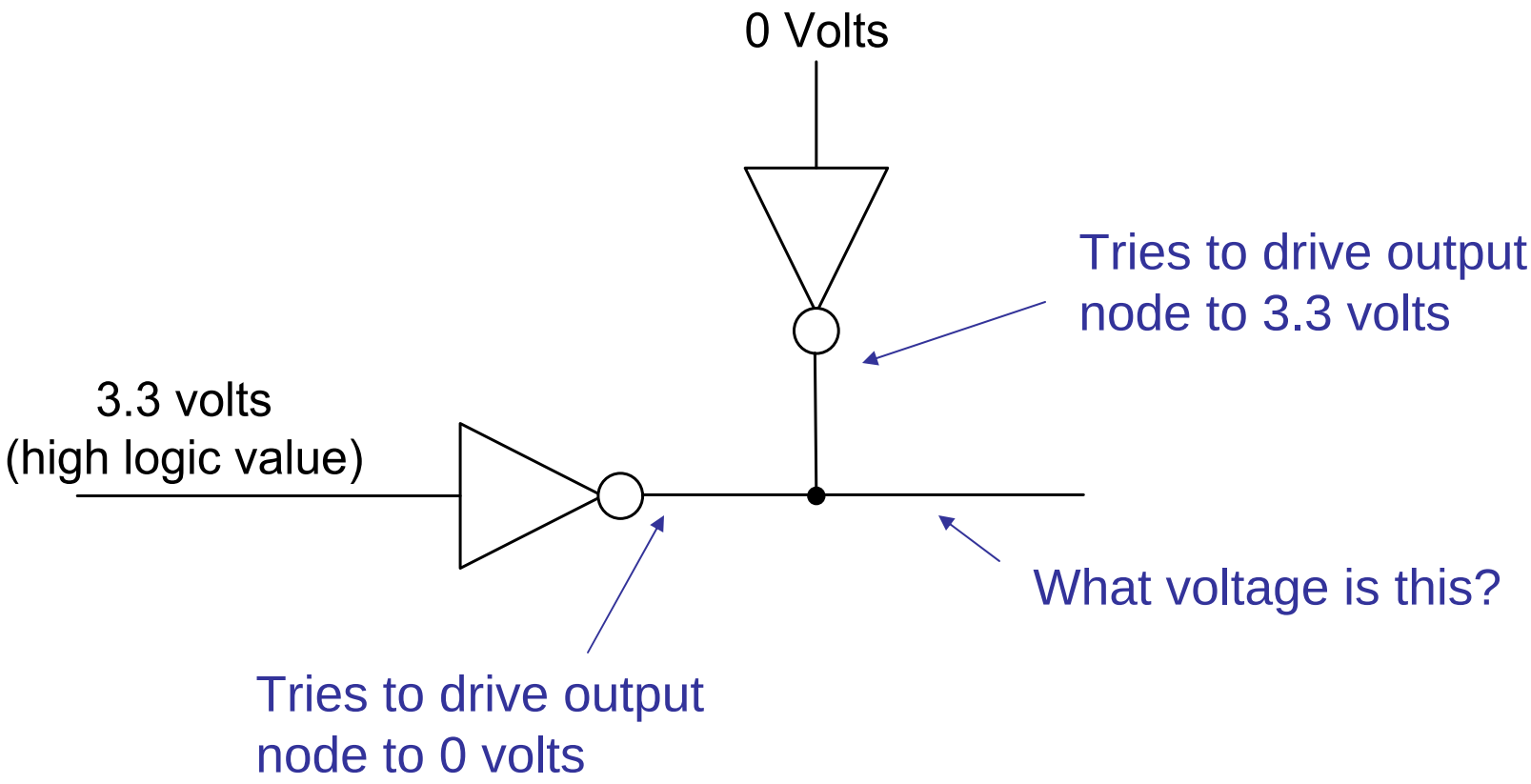
Simon Fraser University

Slide Set: 9
Date: March 2, 2009

Slide Set Overview

- The remainder of this semesters lectures will cover other useful odds and ends and some design examples (as time permits)
  - E.g. buses, floating point calculations, balancing a pipeline, asynchronous circuitry, datapaths, CPU design, and memories
- This lecture will cover
  - Tri-state drivers
  - Generate statements

Tri-State Drivers



Fighting

- This is an example of “fighting”
  - What happens depends on the the circuits are implemented
  - For CMOS (most chips are designed using CMOS), fighting can cause high currents and even damage the chip!

- Interesting question: what would happen if we measure the voltage of such a wire?

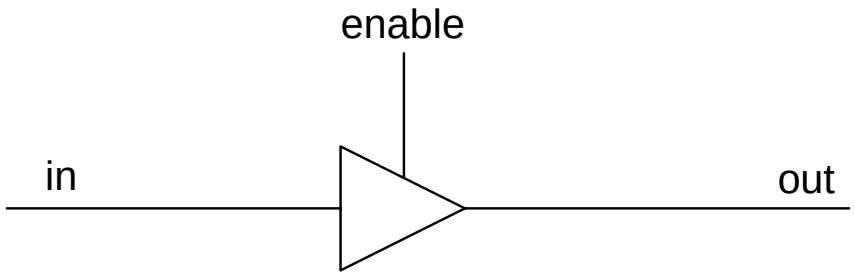
Simulation

- Now think of what happens if we simulate a VHDL specification of this circuit…

- What should the value of this output node be?
  - A signal of type “bit” can be 0 or 1
  - But neither of those is the right answer.
  - One of the possible values of a signal of type ‘std_logic’ is X
    - X stands for unknown

- When the VHDL simulator sees this fighting, it sets the value of the output to X. That tells you something is wrong.
- Remember, in the real circuit, the node will have some value… (can’t measure an X in the lab)

Tri-State Drivers



If “enable” is 1, then out is driven with the value on in.

If “enable” is 0, then out is *not driven*.

- What does that mean? If no one else is driving this signal, then the signal is “floating”
- What happens if you try to measure the voltage of a floating signal in the lab?

Simulation

- Now think of what happens if we simulate a VHDL specification of this circuit…
- If enable is zero, what should the simulation say the output is?
  - One of the possible values of a signal of type ‘std_logic’ is Z
    - Z stands for high impedance

in	enable	out
0	0	z
1	0	z
0	1	0
1	1	1

Remember: in the real world, a signal will have a voltage, even if it is not being driven

- Can’t measure Z in the lab

```
entity tristatedriver is
    port(   A, enable : in std_logic;
            Q : out std_logic);
end tristatedriver;

architecture behavioural of tristatedriver is
begin
    process (A, enable)
    begin
        if (enable = '1') then
          Q <= A;
        end if;
     end process;
end behavioural;
```

WRONG!

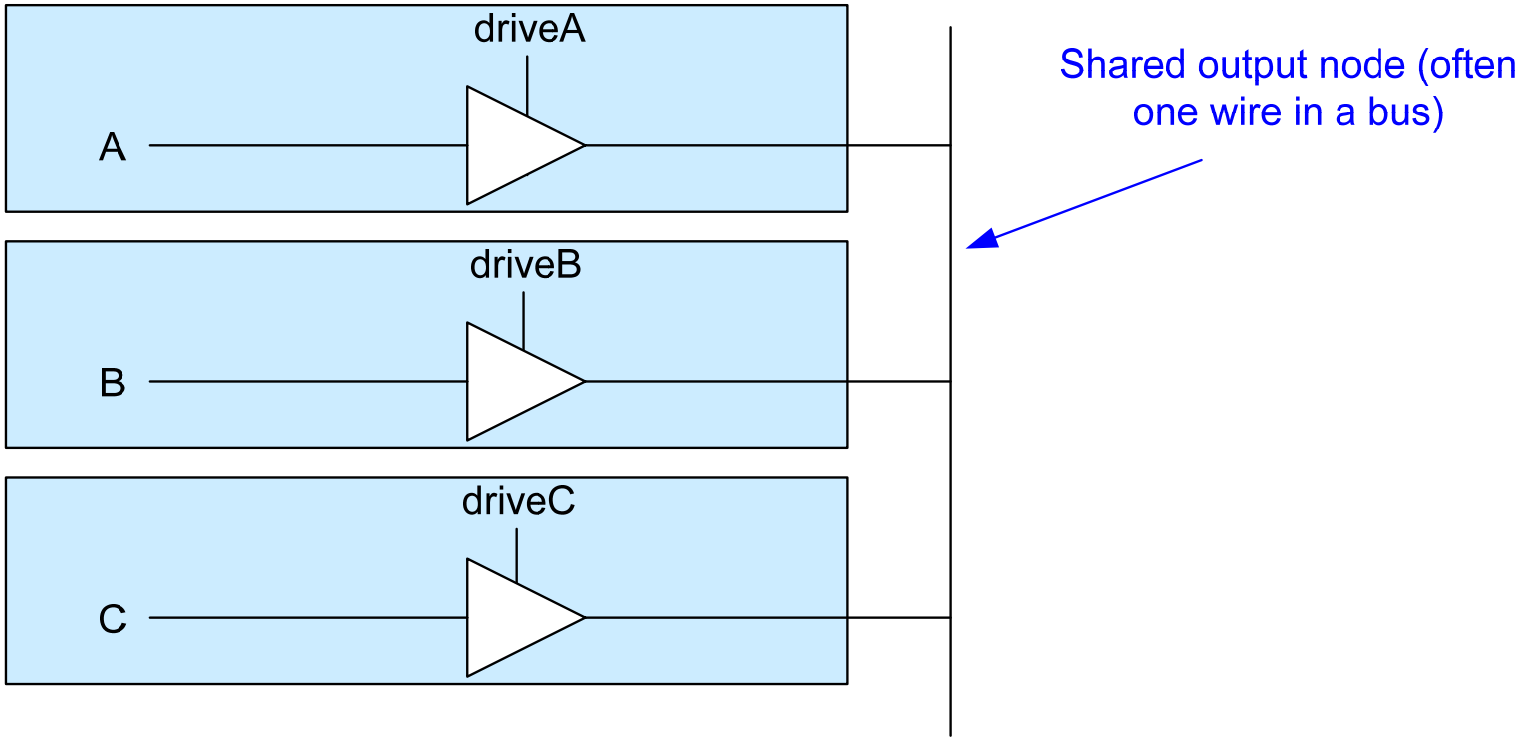
The problem with the previous specification is that if enable is zero, the output is not driven. VHDL syntax says that in this case, the output maintains its old value.

- Not the behaviour we want to specify

```
entity tristatedriver is
    port(   A, enable : in std_logic;
            Q : out std_logic);
end tristatedriver;

architecture behavioural of tristatedriver is
begin
    process (A, enable)
    begin
       if (enable = '1') then
         Q <= A;
       else
         Q <= 'Z';
       end if;
     end process;
end behavioural;
```

How a tri-state driver is usually used



Some Technical Details

Each “type” in VHDL has an associated *resolution function*

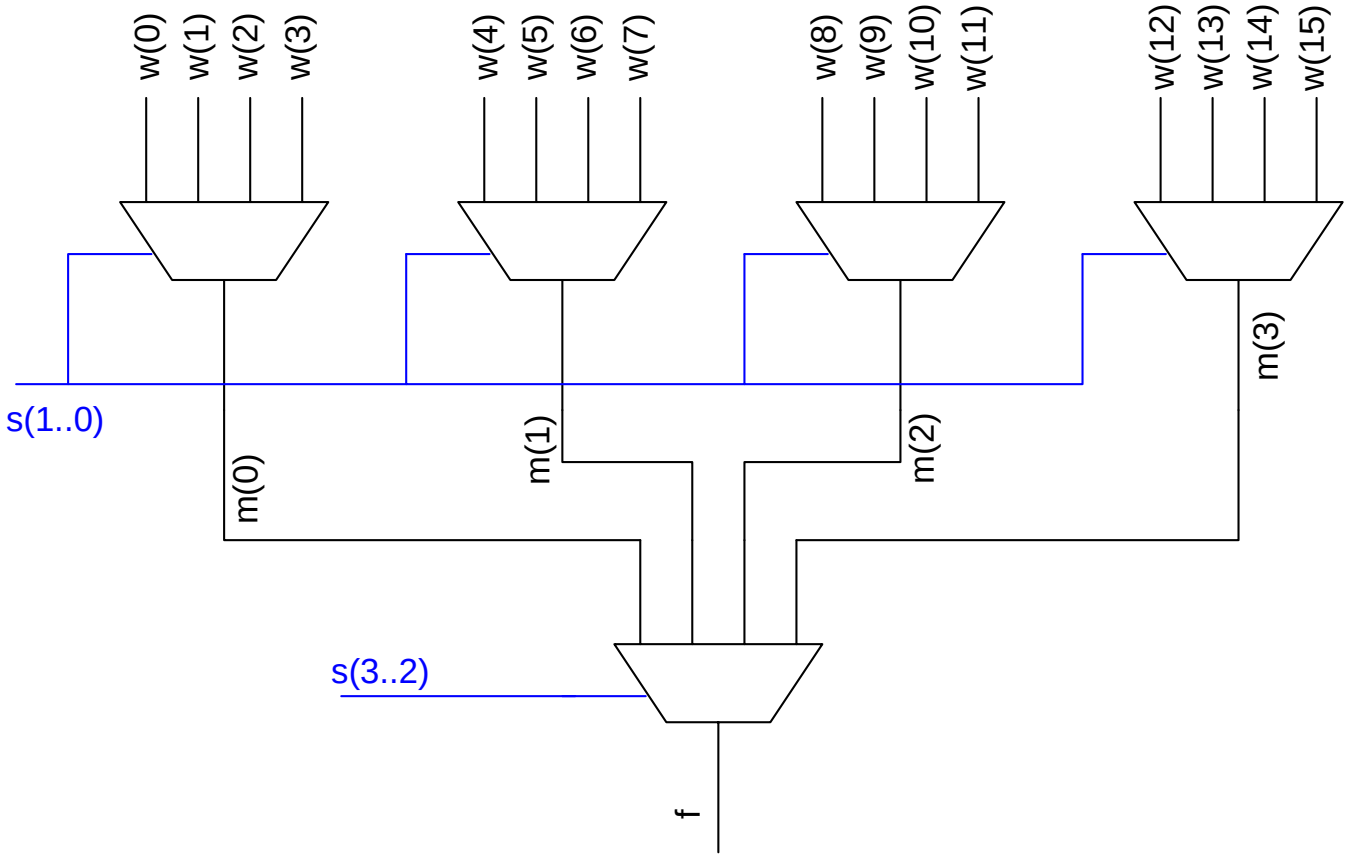
A resolution function takes all the values being driven on a certain wire and calculates what the resulting value on that wire should be.

input	output
1	1
0	0
0, 1	X
0, 0, 0, 0	0
0, 0, 0, 1	X
Z, 0	0
Z, 1, 1, 1	1
Z, 0, 1	X

When you use the std_logic_1164 package, the resolution function for std_logic is already defined

Structural Descriptions – why generate

- Suppose we want to design this circuit:



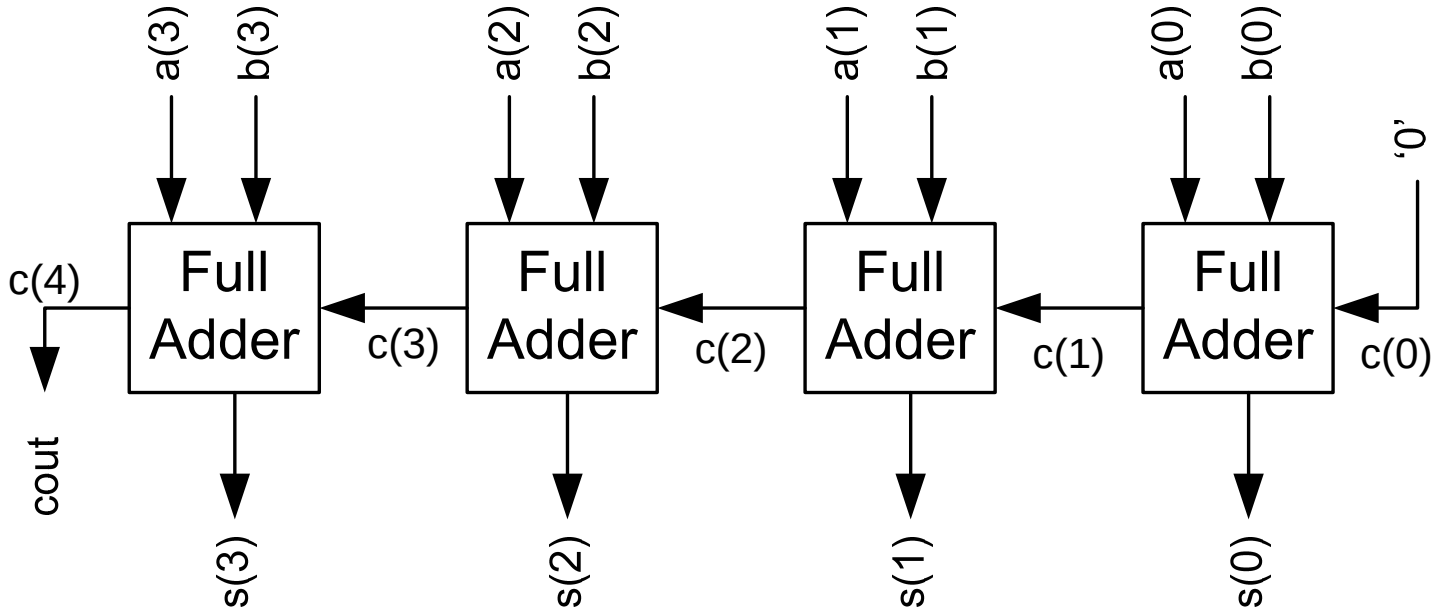
- Further, suppose we already have a 4-input 1-bit mux cell and we want to create a structural description

```
architecture structural of mux16to1 is
    signal m : std_logic_vector(0 to 3);
begin
    u1: mux4to1 port map
        ( w(0), w(1), w(2), w(3), s(1 downto 0), m(0) );
    u2: mux4to1 port map
        ( w(4), w(5), w(6), w(7), s(1 downto 0), m(1) );
    u3: mux4to1 port map
        ( w(8), w(9), w(10), w(11), s(1 downto 0), m(2) );
    u4: mux4to1 port map
        ( w(12), w(13), w(14), w(15), s(1 downto 0), m(3) );
    u5: mux4to1 port map
        ( m(0), m(1), m(2), m(3), s(3 downto 2), f );
end structural
```

Assume we've already defined this block

```
architecture structural of mux16to1 is
    signal m : std_logic_vector(0 to 3);
begin
     g1: for i in 0 to 3 generate
          muxes: mux4to1 port map
                ( w(4*i), w(4*i+1), w(4*i+2), w(4*i+3), s(1 downto 0),
    m(i) );
     end generate;
     u5: mux4to1 port map
           ( m(0), m(1), m(2), m(3), s(3 downto 2), f );
end structural
```

- Another example: 4-bit adder constructed using four full-adders:



```
architecture structural of add4 is
    signal c : std_logic_vector(4 downto 0);
begin
    c(0) <= cin;
    u0: full_adder port map(a(3), b(3), c(3), s(3), c(4));
    u1: full_adder port map(a(2), b(2), c(2), s(2), c(3));
    u2: full_adder port map(a(1), b(1), c(1), s(1), c(2));
    u3: full_adder port map(a(0), b(0), c(0), s(0), c(1));
    cout <= c(4);
end structural
```

```
architecture structural of add4 is
    signal c : std_logic_vector(4 downto 0);
Begin
    c(0) <= cin;
    g0: for i in 3 downto 0 generate
        FA: full_adder port map(a(i), b(i), c(i), s(i), c(i+1));
    end generate g0;
    cout <= c(4);
end structural
```

- Sometimes, individual modules are a little bit different depending on their position in the array

- Example: in our adder, we could argue that:
  - bit 0 is a special case because the carry-in comes from ‘0’
  - bit 3 is a special case because the carry-out goes to cout signal

- We can write “special cases” using “if… generate”

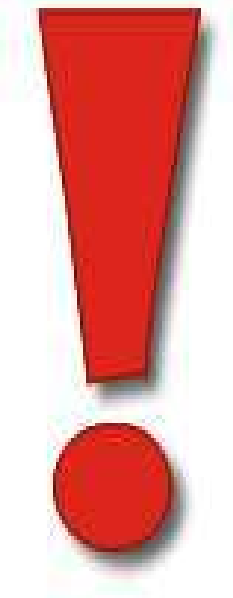
```
architecture structural of add4 is
    signal c : std_logic_vector(3 downto 1);
Begin
    g0: for i in 3 downto 0 generate
        label0: if i = 0 generate
            FA: full_adder port map(a(0), b(0), '0' , s(0), c(1));
        end generate label0;
        label2: if i >0 and i < 3 generate
            FA: full_adder port map(a(i), b(i), c(i) , s(i), c(i+1));
        end generate label2;
        label3: if i = 3 generate
            FA: full_adder port map(a(3), b(3), c(3) , s(3), cout);
        end generate label3;
    end generate g0;
end structural
```

- Warning: too many “special cases” make your code ugly. Use this with discretion!

- Another warning: Generate is *only* a tool for helping you describe structural descriptions!

- It doesn't imply any sort of "looping" in the hardware.

- This is different than a "for" statement in a process
  - A process is used to describe the behaviour of a particular module – not to describe structure
  - I'm not going to teach you how to use "for" in a process, because it almost always leads to unsynthesizable code. Once you become good at writing synthesizable code, then you can read about the "for" statement.



Summary of this slide set

Part 1:

- Wires of type std_logic can have values:
  - X: unknown value (because of fighting usually)
  - Z: high impedance (not driven)

- Really important: These values are not physical voltages
  - You can't measure a Z or X in the lab

- Part 2:
  - Generate statement helps you describe structural descriptions of array-like structures.

Questions

- What is a resolution function?

- Can two outputs drive the same wire?

Questions

- What is a tristate buffer?

- Why do we need/use tristate buffers? Give an example?

Questions

- What is the difference between ‘Z’ and ‘X’?

- What are generate statements? When should they be used?

Questions

- Can you selectively generate circuits?

- Is this like an executable for loop? Why/Why not?