Combinational Building Blocks
The std_logic VHDL Type

- We’ve been blindly using std_logic as the type for all of the inputs, outputs and signals that we’ve been using so far.
- There are others that could be used as well:
  - Bit
  - Integer
  - Std_ulogic
- For most of your work std_logic is fine.
- But, what is std_logic?
A Std_logic signal can actually take on 9 different values:

- ‘U’ – uninitialized
- ‘X’ – strong unknown
- ‘0’ – Forcing 0
- ‘1’ – Forcing 1
- ‘Z’ – high impedance
- ‘W’ – weak unknown
- ‘L’ – weak 0
- ‘H’ – weak 1
- ‘-’ – don’t care

What this means, generically, is that 0 and 1 are not the only states a value can have. This is important to note!
Resolution Functions

- Since we have 9 different levels that a bit may have, how do we determine the output of an operation like A and B when A = ‘W’ and ‘B’ = ‘L’?
- This is part of the standard library which contains a set of rules for resolving the answer.
Resolving AND in std_logic

- From the IEEE 1164 standard we have:

```vhdl
constant resolution_table : stdlogic_table := (  
  -- ---------------------------------------------------------  
  -- |  U    X    0    1    Z    W    L    H    - |   |  
  -- ---------------------------------------------------------  
  ('U', 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U'), -- | U |  
  ('U', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X'), -- | X |  
  ('U', 'X', '0', 'X', '0', '0', '0', '0', 'X'), -- | 0 |  
  ('U', 'X', 'X', '1', '1', '1', '1', '1', 'X'), -- | 1 |  
  ('U', 'X', '0', '1', 'Z', 'W', 'L', 'H', 'X'), -- | Z |  
  ('U', 'X', '0', '1', 'W', 'W', 'L', 'W', 'X'), -- | W |  
  ('U', 'X', '0', '1', 'H', 'W', 'W', 'H', 'X'), -- | L |  
  ('U', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X'), -- | H |  
  ('U', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X')   -- | - |  
);```
- Consider a device such as a tri-state buffer or inverter.
- These devices are often used when more than one device must drive a common bus (as in a memory device).
Here are the models for a non-inverting and inverting tri-state buffers.

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Tristate_Buffer is
  Port ( A : in std_logic;
         OE : in std_logic;
         Q : out std_logic);
end Tristate_Buffer;

architecture Ex1 of Tristate_Buffer is
begin
  Q <= A when OE = '1' else 'Z';
end Ex1;
```

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Tristate_Buffer is
  Port ( A : in std_logic;
         OE : in std_logic;
         Q : out std_logic);
end Tristate_Buffer;

architecture Ex1 of Tristate_Buffer is
begin
  Q <= not A when OE = '0' else 'Z';
end Ex1;
```
A Look at the ISE Output

Let’s see what the compiler thinks.

HDL Synthesis

Synthesizing Unit <tristate_buffer>.
Related source file is "C:/Tristate_Buffer.vhd".
Found 1-bit tristate buffer for signal <Q>.
Summary:
   inferred    1 Tristate(s).
Unit <tristate_buffer> synthesized.

Advanced HDL Synthesis

Advanced RAM inference ...
Advanced multiplier inference ...
Advanced Registered AddSub inference ...
Dynamic shift register inference ...

HDL Synthesis Report
Macro Statistics
# Tristates : 1
1-bit tristate buffer : 1
3-Bit Parity Encoder - Revisited

Consider the following model:

```vhdl
entity Parity_3bit is
    Port ( A : in std_logic_vector(2 downto 0);
          Q : out std_logic);
end Parity_3bit;

architecture Behavioral of Parity_3bit is
begin
    Q <= '0' when A = "000" else
         '1' when A = "001" else
         '1' when A = "010" else
         '0' when A = "011" else
         '1' when A = "100" else
         '0' when A = "101" else
         '0' when A = "110" else
         '1' when A = "111" else
         'X';
end Behavioral;
```
But, What Got Synthesized?

- Let’s look at the output again.

```
*                 HDL Synthesis                     *
```

Synthesizing Unit <parity_3bit>.
Related source file is "C:/Parity_3bit.vhd".
Found 1-bit 8-to-1 multiplexer for signal <Q>.
Summary:
  inferred 1 Multiplexer(s).
Unit <parity_3bit> synthesized.

- Does this make sense?
Here is a multiplexer with two different architecture specifications.

```vhdl
entity Mux4_1 is
  Port ( S : in std_logic_vector(1 downto 0);
          A : in std_logic_vector(3 downto 0);
          Y : out std_logic);
end Mux4_1;

architecture Model1 of Mux4_1 is
begin
  with S select
  Y <= A(0) when "00",
       A(1) when "01",
       A(2) when "10",
       A(3) when "11",
       'X' when others;
end Model1;

architecture Model2 of Mux4_1 is
begin
  multiplexer: process (S) is
  begin
    case S is
      when "00" =>
        Y <= A(0);
      when "01" =>
        Y <= A(1);
      when "10" =>
        Y <= A(2);
      when "11" =>
        Y <= A(3);
      when others =>
        Y <= 'X';
    end case;
  end process multiplexer;
end Model2;
```
Synthesis Results for Multiplexer

- Both multiplexer models synthesize exactly the same logic!
- Looking at the RTL schematic, we get the following for both models: