Multi-Valued Logic

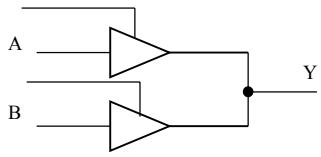
- VHDL allows users to extend the language by defining their own data types
- Early on, users recognized that the BIT data type was insufficient for digital simulation
  - BIT can only have values of '1' or '0'
  - Digital systems require other conditions such as 'Z' (tri-state), 'X' (unknown), weak/strong drivers, etc.
- Companies began writing VHDL models that used proprietary data types that added support for these logic values
  - Each company defined their own data types
  - Models were not interoperable because they used these custom data types

IEEE Standard Multivalue Logic System (1164)

- An IEEE standards body created the *std_logic_1164* standard to support multivalued logic in VHDL
- A *std_logic* data type has the following values:
  - 'U' uninitialized (leftmost literal, default initial value)
  - 'X' forcing unknown
  - '0' forcing 0
  - '1' forcing 1
  - 'Z' high impedance
  - 'W' weak unknown
  - 'L' weak 0 (pulldown)
  - 'H' weak 1 (pullup)
  - '-' don't care (used for synthesis only)

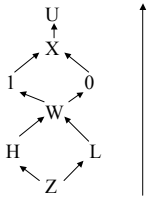
Resolved Data Types

- *std_logic* is known as a *resolved* data type in VHDL
- A *resolved* data type allows more than one driver for a signal
- Necessary for modeling things like tri-state drivers, pullup/pulldown networks.



The Resolution Function

- The resolution function determines the final value of a signal in the case of multiple drivers
- The diagram below is a graphical representation of the resolution function for *std_logic* .



Increasing strength

'Z' is weakest, 'U' is strongest.

Unresolved Types

Cannot have multiple drivers for types that do not have a resolution function. A BIT type is an unresolved type.

```
signal ta,tb : bit;

begin
 ta <= transport '1'  after 2 ns; -- 'ta' only has one driver
 tb <= transport '1' after 3 ns; --  'tb' has multiple drivers

 p1:process
    begin
      tb <= transport '0' after 5 ns;
      wait;
    end process p1;
```

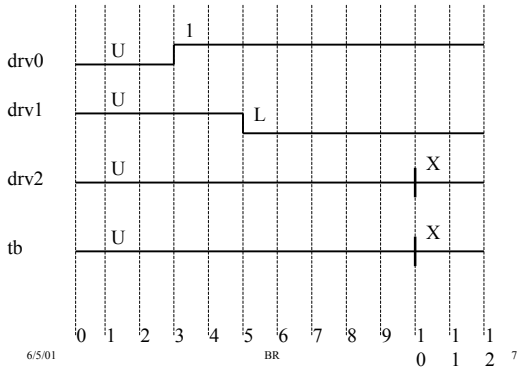
```
-- The following compilation error is generated
--###### exam2/driverbad.vhd(27):       tb <= transport
--ERROR: exam2/driverbad.vhd(27): Nonresolved signal tb
 line 24).
--###### exam2/driverbad.vhd(32):  end;
--ERROR: exam2/driverbad.vhd(32): VHDL Compiler exiting
 end;
```


Driver Resolution: Example #1

```
signal tb : std_logic;

begin

  --  'tb' has multiple drivers

  tb <= transport '1' after 3 ns;    -- DRV0

  p1:process
    begin
      tb <= transport 'L' after 5 ns;  -- DRV1
      wait;
    end process p1;

   tb <=  transport 'X' after 10 ns;  -- DRV 2
   end;
```

What does the waveform of signal *tb* look like?

Driver Resolution: Example #1



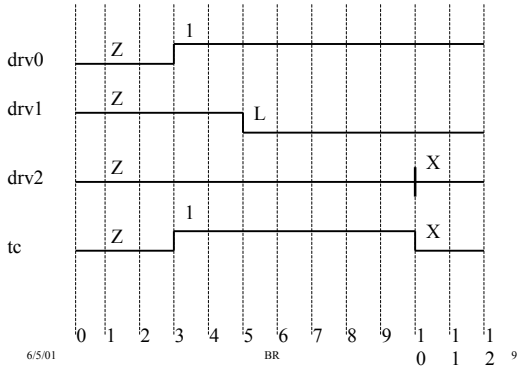
Driver Resolution: Example #2

```
signal tc : std_logic := 'Z';
begin

 --  'tc' has multiple drivers
 tc <= transport '1' after 3 ns;    -- DRV0

 p1:process
   begin
     tc <= transport 'L' after 5 ns;  -- DRV1
     wait;
   end process p1;

  tc <=  transport 'X' after 10 ns;  -- DRV 2
  end;
```

Only difference from previous example is *tc* has an initial value of 'Z'

What does the waveform of signal *tc* look like?

Driver Resolution: Example #2

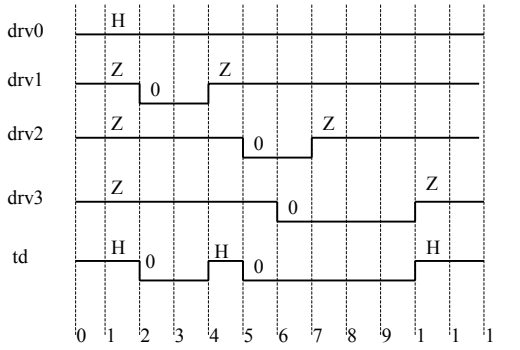


Driver Resolution: Example #3

```
signal td : std_logic := 'Z';
begin

  -- emulate a pullup resistor using 'H' drive
  td <= 'H';    -- DRV0
  -- DRV1
  td <= transport '0' after 2 ns, 'Z' after 4 ns;
  -- DRV2
  td <= transport '0' after 5 ns, 'Z' after 7 ns;
  -- DRV3
  td <= transport '0' after 6 ns, 'Z' after 10 ns;
end;
```

What does the waveform of signal *td* look like?

Driver Resolution: Example #3



Details on 1164

- Look at the *std_logic_1164.vhd* file attached to the class WWW page
- *std_ulogic* is the base type defined in the 1164 standard
  - This is an unresolved type
  - signals of type *std_ulogic* cannot have multiple drivers
- *std_logic* is the resolved subtype of *std_ulogic*
  - a resolved type is always a subtype of another unresolved type

```
type std_ulogic is ('U', 'X', '0', '1', 'Z',
'W','L', 'H', '-');
subtype std_logic is resolved std_ulogic;
```

std_logic vs *std_ulogic*

- *std_logic* is subtype of *std_ulogic*
- Subtypes can be used in place of the type from which it was derived, without needing an explicit conversion function
- *std_ulogic* signals can be assigned to *std_logic* signals, and vice versa
- *std_ulogic_vector* and *std_logic_vector* are the array types for *std_ulogic* and *std_logic* respectively
  - *std_logic_vector* is not a subtype of *std_logic_ulogic* and a type conversion function is needed for assignments between signals of these types

std_logic vs *std_ulogic* (cont.)

- You should use *std_ulogic* and *std_ulogic_vector* for signals that only require one driver
  - accidental connections between signals that should only have one driver can be detected by the compiler

1164 Resolution Table

```
TYPE stdlogic_table IS ARRAY(std_ulogic, std_ulogic) OF std_ulogic;
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', 'W', 'W', 'X' ), -- | W |
   ( 'U', 'X', '0', '1', 'L', 'W', 'L', 'W', 'X' ), -- | L |
   ( 'U', 'X', '0', '1', 'H', 'W', 'W', 'H', 'X' ), -- | H |
   ( 'U', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X' )  -- | - |
     );
```

The resolution function uses the lookup table to resolve types.

1164 Resolution Function

```
FUNCTION resolved ( s : std_ulogic_vector ) RETURN std_ulogic IS
VARIABLE result : std_ulogic := 'Z';  -- weakest state default
BEGIN
  -- the test for a single driver is essential otherwise the
  -- loop would return 'X' for a single driver of '-' and that
  -- would conflict with the value of a single driver unresolved
  -- signal.
IF    (s'LENGTH = 1) THEN    RETURN s(s'LOW);
       ELSE
           FOR i IN s'RANGE LOOP
               result := resolution_table(result, s(i));
           END LOOP;
       END IF;
      RETURN result;
END resolved;
```

'S' is a list of all driver values for the signal to be resolved.

What Else is in IEEE 1164?

- Boolean functions (AND, OR, etc)
- Subtypes with restricted members (X01, X01Z, UX01, UX01Z)
  - conversion functions to/from vector types to these types
  - useful in models where you do not want to deal with the full range of types
- Conversion functions to/from BIT, BIT_VECTOR to std_ulogic, std_ulogic_vector
- Misc functions
  - rising_edge, falling_edge, is_X