Outline

• Review of Previous Classes
• Operators and Data Types
• Priority of logic and relational operators
• Data-Flow VHDL
• Modeling Wires and Buses
• Combinational-Circuit Building Blocks
Review of Previous Classes - 1

- **Entity Declaration**

```vhdl
ENTITY nand_gate IS
  PORT(
    a : IN STD_LOGIC;
    b : IN STD_LOGIC;
    z : OUT STD_LOGIC
  );
END nand_gate;
```

- **Reserved words**
- **Entity name**
- **Port names**
- **Port type**
- **Semicolon**
- **No Semicolon after last port**
- **Port modes (data flow directions)**
Review of Previous Classes - 2

• Entity declaration – simplified syntax

ENTITY entity_name IS
    PORT (  
        port_name : port_mode signal_type;
        port_name : port_mode signal_type;
        ...........
        port_name : port_mode signal_type);
END entity_name;
Review of Previous Classes - 3

- Architecture example:

```vhdl
ARCHITECTURE dataflow OF nand_gate IS
BEGIN
    z <= a NAND b;
END dataflow;
```

- Simplified syntax

```vhdl
ARCHITECTURE architecture_name OF entity_name IS
    [ declarations ]
BEGIN
    code
END architecture_name;
```
Reading

Required

• P. Chu, *FPGA Prototyping by VHDL Examples*
  
  *Chapter 3, RT-level combinational circuit*
  
  *Sections 3.1, 3.2, 3.3, 3.6, 3.7.1, 3.7.3.*

Recommended

• S. Brown and Z. Vranesic, *Fundamentals of Digital Logic with VHDL Design*
  
  *Chapter 6, Combinational-Circuit Building Blocks*
  
  *Chapter 5.5, Design of Arithmetic Circuits Using CAD Tools*
Operators and Data Types
Priority of logic and relational operators
Types of VHDL Description

VHDL Descriptions

- dataflow
  - Concurrent statements

- structural
  - Components and interconnects

- behavioral
  - Sequential statements
    - Registers
    - State machines
    - Instruction decoders
  - Testbenches

Subset most suitable for synthesis
Data-Flow VHDL
Most often implied structure
Examples
Selected concurrent signal assignment

expression1 → choices_1
expression2 → choices_2
expressionN → choices_N

target_signal

choice expression
Examples

\[ b <= \\
"1000" \text{ when } a = "00" \text{ else} \\
"0100" \text{ when } a = "01" \text{ else} \\
"0010" \text{ when } a = "10" \text{ else} \\
"0001" \text{ when } a = "11"; \]
Allowed formats of \textit{choices}\_\textsubscript{k}
when-else vs. with-select-when
Modeling Wires and Buses

SIGNAL a : STD_LOGIC;

SIGNAL b : STD_LOGIC_VECTOR(7 DOWNTO 0);
Merging wires and buses

\[ d = a \parallel b \parallel c \]
Splitting buses

\[ a = d_{9..6} \]

\[ b = d_{5..1} \]

\[ c = d_0 \]
Combinational-Circuit Building Blocks

- Commonly used combinational building blocks in design of large circuits:
  - Fixed Shifters & Rotators
  - Adders
  - Multiplexers
  - Decoders and Encoders
  - Multipliers
  - Comparators
  - Buffers
  - ROM
Fixed Shifters / Rotators

- “fixed” shifters “hardwire” the shift amount into the circuit.
- Ex: X >> 2 – (right shift X by 2 places)

Fixed shift/rotator is nothing but wires!
Fixed Logical Shift Right in VHDL

SIGNAL A : STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL C : STD_LOGIC_VECTOR(3 DOWNTO 0);
Fixed Arithmetic Shift Right in VHDL
Fixed Logical Shift Left in VHDL

SIGNAL A : STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL C : STD_LOGIC_VECTOR(3 DOWNTO 0);
Fixed Rotation Left in VHDL

SIGNAL A : STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL C : STD_LOGIC_VECTOR(3 DOWNTO 0);
Multiplexers

- 2-to-1 Multiplexer

(a) Graphical symbol

(b) Truth table
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY mux2to1 IS
  PORT ( w0, w1, s : IN STD_LOGIC ;
         f : OUT STD_LOGIC ) ;
END mux2to1 ;

ARCHITECTURE dataflow OF mux2to1 IS
BEGIN

END dataflow ;
Examples: Cascade of two multiplexers
VHDL design entity implementing a cascade of two multiplexers
4-to-1 Multiplexer

(a) Graphic symbol

(b) Truth table

\[
\begin{array}{c|c|c|c}
   & s_1 & s_0 & f \\
0 0 & 0 0 & w_0 \\
0 1 & 0 1 & w_1 \\
1 0 & 1 0 & w_2 \\
1 1 & 1 1 & w_3 \\
\end{array}
\]
VHDL code for a 4-to-1 Multiplexer entity

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY mux4to1 IS
  PORT ( w0, w1, w2, w3 : IN STD_LOGIC ;
         s   : IN STD_LOGIC_VECTOR(1 DOWNTO 0) ;
         f   : OUT STD_LOGIC ) ;
END mux4to1 ;

ARCHITECTURE dataflow OF mux4to1 IS
BEGIN

END dataflow ;
Decoders

- 2-to-4 Decoder

(a) Truth table

<table>
<thead>
<tr>
<th>$En$</th>
<th>$w_1$</th>
<th>$w_0$</th>
<th>$y_3$</th>
<th>$y_2$</th>
<th>$y_1$</th>
<th>$y_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>x</td>
<td>x</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(b) Graphical symbol
VHDL code for a 2-to-4 Decoder entity

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY dec2to4 IS
    PORT ( w : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
           En : IN STD_LOGIC;
           y : OUT STD_LOGIC_VECTOR(3 DOWNTO 0) );
END dec2to4;

ARCHITECTURE dataflow OF dec2to4 IS
    SIGNAL Enw : STD_LOGIC_VECTOR(2 DOWNTO 0);
BEGIN

END dataflow;
Encoders

• Priority Encoder

\[ \begin{array}{cccc}
w_3 & w_2 & w_1 & w_0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & - \\
0 & 1 & - & - \\
1 & - & - & - \\
\end{array} \]

\[ \begin{array}{cccc}
y_1 & y_0 & z \\
d & d & 0 \\
0 & 0 & 1 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 1 \\
\end{array} \]
VHDL code for a Priority Encoder entity

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY priority IS
    PORT ( w : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
y : OUT STD_LOGIC_VECTOR(1 DOWNTO 0);
z : OUT STD_LOGIC );
END priority;

ARCHITECTURE dataflow OF priority IS
BEGIN

END dataflow;

Adders

- 16-bit Unsigned Adder
Operations on Unsigned Numbers

For operations on unsigned numbers

USE ieee.std_logic_unsigned.all
and
signals of the type
STD_LOGIC_VECTOR

OR

USE ieee.numeric_std.all
and
signals of the type
UNSIGNED
and conversion functions:
   std_logic_vector(), unsigned()
VHDL code for a 16-bit Unsigned Adder

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY adder16 IS
    PORT ( Cin : IN STD_LOGIC;
           X    : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
           Y    : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
           S    : OUT STD_LOGIC_VECTOR(15 DOWNTO 0);
           Cout : OUT STD_LOGIC );
END adder16;

ARCHITECTURE dataflow OF adder16 IS

END dataflow;
Addition of Unsigned Numbers (1)

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY adder16 IS
    PORT ( Cin : IN STD_LOGIC;
            X   : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
            Y   : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
            S   : OUT STD_LOGIC_VECTOR(15 DOWNTO 0);
            Cout : OUT STD_LOGIC ) ;
END adder16 ;
Addition of Unsigned Numbers (2)

ARCHITECTURE dataflow OF adder16 IS

END dataflow ;
Addition of Unsigned Numbers (3)

ARCHITECTURE dataflow OF adder16 IS

END dataflow ;
Operations on Signed Numbers

For operations on signed numbers

USE ieee.numeric_std.all,
signals of the type
  SIGNED,
and conversion functions:
  std_logic_vector(), signed()

OR

USE ieee.std_logic_signed.all
and signals of the type
  STD_LOGIC_VECTOR
Signed and Unsigned Types

**Behave** exactly like **STD_LOGIC_VECTOR**

plus, they determine whether a given vector should be treated as a signed or unsigned number. Require

**USE ieee.numeric_std.all;**
## Unsigned vs. Signed Multiplication

<table>
<thead>
<tr>
<th>Unsigned</th>
<th>Signed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111 x 1111 = 11100001</td>
<td>1111 x 1111 = 00000001</td>
</tr>
<tr>
<td>15 x 15 = 225</td>
<td>-1 x -1 = 1</td>
</tr>
</tbody>
</table>
8x8-bit Unsigned Multiplier
Multiplication of unsigned numbers

LIBRARY ieee;
USE ieee.std_logic_1164.all;

entity multiply is
    port(
        a : in STD_LOGIC_VECTOR(7 downto 0);
        b : in STD_LOGIC_VECTOR(7 downto 0);
        c : out STD_LOGIC_VECTOR(15 downto 0)
    );
end multiply;

architecture dataflow of multiply is
begin

end dataflow;
8x8-bit Signed Multiplier

\[
\begin{array}{c}
8 \\
\downarrow \\
a \\
* \\
c \\
\downarrow \\
16 \\
\end{array}
\quad
\begin{array}{c}
8 \\
\downarrow \\
b \\
\end{array}
\quad
\begin{array}{c}
s \\
\end{array}
\]
Multiplication of signed numbers

LIBRARY ieee;
USE ieee.std_logic_1164.all;

entity multiply is
  port(
    a : in STD_LOGIC_VECTOR(7 downto 0);
    b : in STD_LOGIC_VECTOR(7 downto 0);
    c : out STD_LOGIC_VECTOR(15 downto 0)
  );
end multiply;

architecture dataflow of multiply is begin

end dataflow;
8x8-bit Unsigned and Signed Multiplier
LIBRARY ieee;
USE ieee.std_logic_1164.all;

entity multiply is
  port(
    a : in STD_LOGIC_VECTOR(7 downto 0);
    b : in STD_LOGIC_VECTOR(7 downto 0);
    cu : out STD_LOGIC_VECTOR(15 downto 0);
    cs : out STD_LOGIC_VECTOR(15 downto 0)
  );
end multiply;

architecture dataflow of multiply is
begin

end dataflow;
4-bit Number Comparator

\[ A > B \quad \text{AgtB} \]
VHDL code for a 4-bit **Unsigned** Number Comparator entity

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;

ENTITY compare IS
    PORT ( A, B : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
            AgtB : OUT STD_LOGIC ) ;
END compare ;

ARCHITECTURE dataflow OF compare IS
BEGIN
    AgtB <= '1' WHEN A > B ELSE '0' ;
END dataflow ;
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_signed.all;

ENTITY compare IS
  PORT ( A, B : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
        AgtB : OUT STD_LOGIC ) ;
END compare ;

ARCHITECTURE dataflow OF compare IS
BEGIN
  AgtB <= '1' WHEN A > B ELSE '0' ;
END dataflow ;
Tri-state Buffer
Four types of Tri-state Buffers

(a) $x$ \hspace{1cm} $e$ \hspace{1cm} $f$

(b) $x$ \hspace{1cm} $e$ \hspace{1cm} $f$

(c) $x$ \hspace{1cm} $e$ \hspace{1cm} $f$

(d) $x$ \hspace{1cm} $e$ \hspace{1cm} $f$
Tri-state Buffer entity

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY tri_state IS
ROM (Read Only Memory) 8x16 example

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY rom IS
PORT (Addr : in STD_LOGIC_VECTOR(3 downto 0);
      Dout : out STD_LOGIC_VECTOR(15 downto 0));
END rom;
```
ARCHITECTURE dataflow OF rom IS
SIGNAL temp: INTEGER RANGE 0 TO 7;
TYPE vector_array IS ARRAY (0 to 7) OF STD_LOGIC_VECTOR(15 DOWNTO 0);
CONSTANT memory : vector_array :=
    ( X"800A",
      X"D459",
      X"A870",
      X"7853",
      X"650D",
      X"642F",
      X"F742",
      X"F548");
BEGIN
    temp <= to_integer(unsigned(Addr));
    Dout <= memory(temp);
END dataflow;
More examples can be found in the textbook section 3.9

- Multi-function barrel shifter
- Dual-priority encoder
- BCD incrementor
- Floating-point greater-than circuit
- Floating-point and signed integer conversion circuit
- Enhanced floating-point adder
Next Class

- Behavioral Design Style:
- Registers & Counters
- Constants
- Sequential Logic Synthesis

Required
- P. Chu, *FPGA Prototyping by VHDL Examples*  
  Chapter 4, *Regular Sequential Circuit*

Recommended
- S. Brown and Z. Vranesic, *Fundamentals of Digital Logic with VHDL Design*  
  Chapter 7, *Flip-Flops, Registers, Counters, and a Simple Processor*