Structural Modeling

• Breaks up the implementation into a modular hierarchy of subsystems ("components")
• Describes more complex entities in terms of combinations of simpler components
• Component definitions can be created once and reused multiple times in an entity
• Component definitions can be shared by multiple entities

Structural Modeling

• Components can have any degrees of complexity (OR gate, Half Adder, 4-bit RC adder, …)
• "__________" components and then describe how they are interconnected
• Similar concept to user-defined "Functions" in high-level programming languages
  – Need to name the function (________________ name)
  – Need to tell the program what input/output parameters the function (component) needs (_________ signals)
  – Use the function by "calling" it in the code, with specific variables assigned (mapped) to each function parameter (________) for this particular usage instance. (instantiation)

Structural Modeling Example: XOR

\[
\begin{align*}
&\text{Data Flow Descriptions:} \\
&F \leftarrow (A \text{ XOR } B) ; \\
&F \leftarrow ((\text{NOT } A) \text{ AND } B) \text{ OR } (A \text{ AND } (\text{NOT } B)) ;
\end{align*}
\]

Structural Description

\[
\begin{align*}
&\text{Structural Description:} \\
&\text{components} \\
&F \leftarrow (A \text{ AND } (\text{NOT } B)) ;
\end{align*}
\]

Structural Modeling Process

1) Define the “entity” for the upper-level system.
   • Enclose in a “Black Box”
   • Signals going in/out of box belong in entity declaration

\[
\begin{align*}
&\text{ENTITY my_XOR IS} \\
&\quad \text{PORT} \ (\ A, B : \text{IN STD_LOGIC ;} \\
&\quad \quad F : \text{OUT STD_LOGIC } ) ; \\
&\quad \text{END my_XOR ;}
\end{align*}
\]
Structural Modeling Process

2) Break the upper-level system into components.
   • Define component boundaries
   • Signals going in/out of boundaries belong in component's entity statement

3) Write a complete VHDL description for each component
   • "Entity" and "Architecture"
   • Can use any style of Architecture for the component
     • Data-Flow, Behavioral, Structural

4) Begin the structural Architecture description of the upper level system.

5) Declare each lower-level component in the upper level architecture
   • 1 declaration for each component type
   • Using the same name as in the component's own VHDL entity
   • Using a format identical to the component's entity declaration, except called "component" vs "entity"
   • Provides a reference name and signal type to every input/output signal ("port") of the component
   • Same idea as a function or procedure “prototype”
Structural Modeling Process
5) Declare each lower-level component in the upper level architecture

ARCHITECTURE struct OF my_XOR IS

component and_2 is
  port (t, u: in std_logic;
      v: out std_logic);
end component;

component or_2 is
  port (x, y: in std_logic;
      z: out std_logic);
end component;

BEGIN
  \( A \) \rightarrow \text{notA} \rightarrow \text{v1out}
  \( B \) \rightarrow \text{notB} \rightarrow \text{a2out}
  \text{F} \rightarrow \text{a1out}
END struct;

IMPORTANT NOTES:
1. The COMPONENT name ________ the ENTITY name
   - How VHDL finds code for COMP.
2. The order and types of PORT signals must be ___________ in the COMPONENT & ENTITY declarations
   - They will be matched up when the COMPONENT is used.

Structural Modeling Process
6) Name and declare all the internal (intermediate) signals in the top-level architecture.
   - Signals that do not cross the top-level “entity” boundaries
   - Signals that ____________________________
Structural Modeling Process

7) Define every “instance” that each component appears in the top-level design
   • “_______________________________________”
   • Gives each component instance its own name (label)
   • Uniquely defines what internal architecture signals or external entity port signals of the top-level system are connected to each port signal of each component (port map).

Formats:  Direct Mapping:
Label : component_name port map (comp_port1 => highlevel_signal1,
      comp_port2 => highlevel_signal2);

Implied Mapping:
Label : component_name port map (highlevel_signal1,
      highlevel_signal2);

8) Add any additional behavioral / data-flow code.
ARCHITECTURE struct OF my_xor IS
      component and_2 IS
         port (t, u : in std_logic;
              v : out std_logic);
      end component;
      component or_2 IS
         port (x, y : in std_logic;
              z : out std_logic);
      end component;
      SIGNAL notA, notB, a1out, a2out : STD_LOGIC;
      BEGIN
         notA <= not( A);
         notB <= not( B);
         U1: and_2 PORT MAP (t => notA, u => B,v => a1out);
         U2: and_2 PORT MAP (A, notB, a2out);
         U3: or_2  PORT MAP (z => F, x => a1out, y => a2out );
      END struct;
ENTITY MUX_4_to_1 IS
PORT (D0, D1, D2, D3 : IN STD_LOGIC;
Sel : IN STD_LOGIC_VECTOR(1 downto 0);
F : OUT STD_LOGIC);
END MUX_4_to_1;

ARCHITECTURE Selected_Mux OF MUX_4_TO_1 IS
BEGIN
WITH Sel SELECT
F <= D0 WHEN "00",
D1 WHEN "01",
D2 WHEN "10",
D3 WHEN "11",
'0' WHEN OTHERS;
END Selected_Mux;

ARCHITECTURE struct OF MUX_15_to_1 IS
COMPONENT MUX_4_to_1 IS
PORT (D0, D1, D2, D3 : IN STD_LOGIC;
Sel : IN STD_LOGIC_VECTOR(1 downto 0);
F : OUT STD_LOGIC);
END COMPONENT;
SIGNAL F1, F2, F3, F4 : STD_LOGIC;
BEGIN
M1: MUX_4_to_1 PORT MAP (D(3 downto 0), D(3 downto 0), '0', D(3 downto 0), C3);
M2: MUX_4_to_1 PORT MAP (D(3 downto 0), D(3 downto 0), D(3 downto 0), D(3 downto 0), Sel(1 downto 0), ______);
M3: MUX_4_to_1 PORT MAP (D(3 downto 0), D(3 downto 0), D(3 downto 0), D(3 downto 0), Sel(1 downto 0), ______);
M4: MUX_4_to_1 PORT MAP (D(3 downto 0), D(3 downto 0), D(3 downto 0), D(3 downto 0), Sel(1 downto 0), ______);
M5: MUX_4_to_1 PORT MAP (________, ______, ______, ______, ______, ______);
END struct;

ENTITY MUX_15_to_1 IS
PORT (D : IN STD_LOGIC_VECTOR(14 downto 0);
Sel : IN STD_LOGIC_VECTOR(3 downto 0);
F : OUT STD_LOGIC);
END MUX_15_to_1;

ARCHITECTURE Structural MODEL OF Adder16 IS
COMPONENT Adder4 IS
PORT (A, B : IN Std_Logic_Vector(3 downto 0);
CI : IN Std_Logic;
S : OUT Std_Logic_Vector(3 downto 0);
CO : OUT Std_Logic);
END COMPONENT;
SIGNAL C3, C7, C11 : Std_Logic;
BEGIN
Low4Add: Adder4
PORT MAP (A(3 downto 0), B(3 downto 0), '0', S(3 downto 0), C3, ______);
END struct;

ENTITY MUX_15_to_1 IS
PORT (D : IN STD_LOGIC_VECTOR(14 downto 0);
Sel : IN STD_LOGIC_VECTOR(3 downto 0);
F : OUT STD_LOGIC);
END MUX_15_to_1;

ARCHITECTURE struct OF MUX_15_to_1 IS
COMPONENT MUX_4_to_1 IS
PORT (D0, D1, D2, D3 : IN STD_LOGIC;
Sel : IN STD_LOGIC_VECTOR(1 downto 0);
F : OUT STD_LOGIC);
END COMPONENT;
SIGNAL F1, F2, F3, F4 : STD_LOGIC;
BEGIN
M1: MUX_4_to_1 PORT MAP (D(3 downto 0), D(3 downto 0), D(3 downto 0), D(3 downto 0), Sel(1 downto 0), ______);
M2: MUX_4_to_1 PORT MAP (D(3 downto 0), D(3 downto 0), D(3 downto 0), D(3 downto 0), Sel(1 downto 0), ______);
M3: MUX_4_to_1 PORT MAP (D(3 downto 0), D(3 downto 0), D(3 downto 0), D(3 downto 0), Sel(1 downto 0), ______);
M4: MUX_4_to_1 PORT MAP (D(3 downto 0), D(3 downto 0), D(3 downto 0), D(3 downto 0), Sel(1 downto 0), ______);
M5: MUX_4_to_1 PORT MAP (________, ______, ______, ______, ______, ______);
END struct;

VHDL Structural Model

But...How do we get VHDL to "ignore" unused component outputs in Structural Designs?

16–Bit Adder: A15..0 + B15..0 = S15..0

ARCHITECTURE Structural OF Adder16 IS
COMPONENT Adder4 IS
PORT (A, B : IN Std_Logic_Vector(3 downto 0);
CI : IN Std_Logic;
S : OUT Std_Logic_Vector(3 downto 0);
CO : OUT Std_Logic);
END COMPONENT;
SIGNAL C3, C7, C11 : Std_Logic;
BEGIN
Low4Add: Adder4
PORT MAP (A(3 downto 0), B(3 downto 0), '0', S(3 downto 0), C3, ______);
END struct;

Structural Design