Types of Concurrent Assignment Statements

• Simple Signal Assignments

• Selected Signal Assignments

• Conditional Signal Assignment
ARCHITECTURE my_arch OF my_entity IS
SIGNAL one_signal : STD_LOGIC;
SIGNAL A_bus : STD_LOGIC_VECTOR( 7 DOWNTO 0 );
    -- Most Significant Bit: A_bus(7), Least Significant Bit: A_bus(0)
SIGNAL D_bus : STD_LOGIC_VECTOR( 0 TO 3 );
    -- MSBit: D_bus(0), LSBit: D_bus(3)
SIGNAL Thr3_Bit_bus : STD_LOGIC_VECTOR( 3 DOWNTO 1 );
BEGIN -- Examples of Simple Concurrent Signal Assignments:
    one_signal <= '0'; --Use ‘single quotes’ for single-bit explicit assignments
    D_bus <= "1010" ; -- Use "double quotes" for multi-bit assignments
        -- D_bus(0) = __, D_bus(1) = ___, D_bus(2) = ___, D_bus(3) = ____
    A_bus <= "0000" & D_bus ; -- & used for concatenation
        -- A_bus(0) = __, A_bus(1) = ___, A_bus(2) = ___, A_bus(3) = ___
    Thr3_Bit_bus <= A_bus( 5 DOWNTO 3 );
        -- multiple-bit indexing
        -- Thr3_Bit_bus <= "0 0 1" Thr3_Bit_bus(3) = ___, Thr3_Bit_bus(1)=___
END arch_name;

ENTITY my_entity IS
    PORT (output1 : OUT STD_LOGIC);
END my_entity;
ARCHITECTURE my_arch OF my_entity IS
SIGNAL one_signal : STD_LOGIC;
BEGIN
    output1 <= '0';
    one_signal <= output1 ;
    -- ERROR: VHDL does not permit Entity OUTputs to be used on the Right
    -- Side of signal assignments as "inputs" for determining other signals
-- OK: VHDL does permit intermediate SIGNALs declared inside the
     -- ARCHITECTURE to be used on the Either Side of signal assignments

Selected Signal Assignment

• Assign one of several possible values to a signal
• Which value gets assigned depends on the present
  value of a single selection criterion

Syntax:

_________ select_criterion _____________
  signal <= expression1 WHEN criterionValue1,
            expression2 WHEN criterionValue2,
            expression3 WHEN criterionValue3,
            expression4 WHEN criterionValue4;

Defines what to do if NONE of the criterionValues are present.

Evaluated by VHDL:

• Whenever there is a value change in any signals in
  the "select_criterion" or in any of the
  "criterionValue(s)"

Cautions:

• All choices are evaluated and checked each time
• Need a “WHEN” clause _________________

_________ select_criterion _____________

if you omit the “OTHERS” criterion value

(What if: select_criterion : STD_LOGIC…??
Did you specify what to do if select_criterion = 'Z' ??, or '-' ??, or…??)
ENTITY XOR_Gate IS
PORT ( A, B : IN STD_LOGIC ;
          F : OUT STD_LOGIC ) ;
END XOR_Gate ;

ARCHITECTURE Selected_Example OF XOR_Gate IS
BEGIN

END Selected_Example ;

ENTITY XOR_Gate IS
PORT ( A, B : IN STD_LOGIC ;
          F : OUT STD_LOGIC ) ;
END XOR_Gate ;

ARCHITECTURE Selected_Example OF XOR_Gate IS
SIGNAL A_B : STD_LOGIC_VECTOR (1 DOWNTO 0);
BEGIN
A_B <= (A & B) ;  -- Combine into a 2-bit criterion
WITH (A_B) SELECT

END Selected_Example ;

ENTITY MUX_4_to_1 IS
ID : STD_LOGIC_VECTOR (3 DOWNTO 0)
END MUX_4_to_1 ;

Example: 4:1 MUX

ENTITY MUX_4_to_1 IS
ID : STD_LOGIC_VECTOR (3 DOWNTO 0)
END MUX_4_to_1 ;
ARCHITECTURE Selected_Mux OF MUX_4_TO_1 IS
BEGIN

<table>
<thead>
<tr>
<th>Sel(1)</th>
<th>Sel(0)</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>D0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>D1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>D2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>D3</td>
</tr>
</tbody>
</table>

END Selected_Mux;

Conditional Signal Assignment

• Assign one of several different values to a signal depending on which one of several possible conditions are currently true.

Syntax:

\[
\text{signal} \leq \text{expression1 WHEN condition1 ELSE expression2 WHEN condition2 ELSE expression3 WHEN condition3 ELSE expression4 ;}
\]

NOTES:

1. Each WHEN “condition” can use
   • different signals,
   • different values,…
   • can be very unique.
   • Must be a Boolean logic expression
     • Must evaluate to either ______ or ______

2. The “conditions” are evaluated in the order that they appear in the code.
   – The __________ condition is acted upon
     • The rest are __________
   – Sequence Matters (!) for the conditions
     • The Signal Assignment is still “concurrent” with other signal assignments.
Conditional Signal Assignment

NOTES:
3. The final expression (after the final ELSE) handles all the other cases that don't match the conditions specified:

```vhdl
signal <= expression1 WHEN condition1 ELSE
expression2 WHEN condition2 ELSE
expression3 WHEN condition3 ELSE
expression4;
```

Example: 4:1 MUX With Strobe Enable (Active High)

```vhdl
ENTITY MUX_4_to_1 IS
  PORT ( D0, D1, D2, D3, EN: IN STD_LOGIC;
         Sel: IN STD_LOGIC_VECTOR( 1 downto 0);
         F : OUT STD_LOGIC );
END MUX_4_to_1;
ARCHITECTURE Conditional_Example OF MUX_4_to_1 IS
BEGIN
  F <=
END Conditional_Example;
```
ENTITY MUX_4_to_1 IS
PORT ( D0, D1, D2, D3, EN: IN STD_LOGIC ;
      Sel: IN STD_LOGIC_VECTOR( 1 downto 0) ;
      F: OUT STD_LOGIC );
END MUX_4_to_1;
ARCHITECTURE Conditional_Mux OF MUX_4_TO_1 IS
BEGIN
  - Conditional Concurrent Assignment
  \begin{align*}
  &F \leftarrow \text{\textit{\textbf{- Above takes priority – Evaluated 1st}}} \\
  &\text{WHEN (Sel="00") ELSE} \\
  &\text{WHEN (Sel="01") ELSE} \\
  &\text{WHEN (Sel="10") ELSE} \\
  &\text{WHEN (Sel="11") ELSE} \\
  &'0' ;
  \end{align*}
END Conditional_Mux;

Other Applications of Conditional & Selective Assignments

- Binary Encoders / Decoders
  
  \(2^n\) binary codes \(\leftrightarrow\) “n” Channels
  “n” Channels \(\leftrightarrow\) \(2^n\) binary codes

- Code Converters
  - Change from one code type to another
    - Binary
    - Binary-Coded-Decimal (BCD)
    - Gray Codes
    - Parity / Error Detecting / Error Correcting Codes (CD’s)
    - Data Modulation Encoding (data storage / communication)
    - Display Control Codes (7-segment display)

Binary Decoders

- Converts an “n” bit Binary Code (applied to “n” inputs) to activate 1 of \(2^n\) different output lines. (“n-to-2^n”)
  - All other outputs “inactive” (0 if active high)
  - May have a strobe enable input

Binary Decoders in VHDL

Which VHDL statements for the needed operations?

- Determining \(F\) outputs for the given \(B\) inputs?
  - Simple Signal Assignments??
  - Selected Signal Assignments??
  - Conditional Assignments??

<table>
<thead>
<tr>
<th>En</th>
<th>B1</th>
<th>B0</th>
<th>F3</th>
<th>F2</th>
<th>F1</th>
<th>F0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</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>
</tbody>
</table>
**Binary Encoders**

- Outputs an “n” bit Binary Code with the Input Channel Number that is “active” (of $2^n$ different inputs. (“2^n-to-n Encoder”)
  - All other inputs “inactive” (0 if active high)

<table>
<thead>
<tr>
<th>B0</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>F0</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Priority Encoders**

- Outputs the “n” bit Binary Code with the “Highest Priority” Input Channel Number that is “active”
  - ___________ can be “active” ___________
  - Only the “___________” input gets encoded → output
  - Usually has a “strobe” output to indicate if ___________

**Gray Codes**

Main Characteristic:
- Next code in sequence changes from previous code ________________ position
  - Just like minterms/Maxterms for adjacent cells in K-Map
  - “Unit Distance Code”

2-bit Reflective Gray Code:

```
0 0
0 1
1 1
1 0
```

Creating a Reflective Gray Code:
1. Start with a 1 or 2 bit RGC
2. Make a mirror image below it
3. Add leading 0’s to upper half; add leading 1’s to lower half
4. Repeat as needed
Gray Code Encoders

“2-bit Binary-to-RGC Encoder”

<table>
<thead>
<tr>
<th>B1</th>
<th>B0</th>
<th>G1</th>
<th>G0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

ENTITY Gray_Encoder_2bit IS

PORT (Binary(0) = B0, Binary(1) = B1, Gray(0) = G0, Gray(1) = G1);

END Gray_Encoder_2bit;

Alternate VHDL – Mixed CSA Types

<table>
<thead>
<tr>
<th>Binary (1)</th>
<th>Binary (0)</th>
<th>Gray (1)</th>
<th>Gray (0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

ENTITY Gray_Encoder_2bit IS

PORT (Binary : IN STD_LOGIC_VECTOR(1 downto 0), Gray : OUT STD_LOGIC_VECTOR(1 downto 0));

END Gray_Encoder_2bit;

ARCHITECTURE Fancy_Gray_Enc OF Gray_Encoder_2bit IS

BEGIN

Gray(1) <= Binary(1); -- Simple Conc. Assign.

Gray(0) <= Binary(0) WHEN (Binary(1)='0') ELSE NOT(Binary(0));

END Fancy_Gray_Enc;

7-Segment Display Control

COMMON-CATHODE Display

1 = V_{ce}

COMMON-ANODE Display

1 = V_{ce}

LEDs (light emitting diodes)

Ground connection for common-cathode

(1 = LED ON)

1 = V_{ce}

(1 = LED ON)