Introduction

VHDL code is inherently concurrent (parallel). Only statements placed inside a process, function, or procedure are sequential. Concurrent code is also called dataflow code. As an example, let us consider a code with three concurrent statements (stat1, stat2, stat3). Then any of the alternatives below will render the same physical circuit:

stat1   stat3   stat2
stat2   stat2   stat3
stat3   stat1   stat1

It is clear that, since the order does not matter, purely concurrent code cannot be used to implement synchronous circuits (except guarded block). In other words, in general we can only build combinational logic circuits with concurrent code. To obtain sequential logic circuit, sequential code must be employed.

In this lab, we will discuss concurrent statements. They are the WHEN statements and GENERATE. Besides them, assignments using only operators (logical, arithmetic, etc) can be obviously also be used to create combinational circuits. In summary, in concurrent code the following can be used:
1. Operators.
2. The WHEN statements (WHEN/ELSE or WITH/SELECT/WHEN)
3. The GENERATE statements

Using operators

This is the most basic way of creating concurrent statements code. Operators (AND, OR, +, -, *, sll, sra, etc) can be used to implement any combinational circuit.

**Example: Multiplexer#1**

![Multiplexer Diagram]

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity mux is
  port (A,B,C,D :in bit;
          s0,s1:in bit;
          y: out bit);
end mux;

architecture mux of mux is
begin
  y<= (A and not s1 and not s0)or
       (B and not s1 and  s0)or
       (C and  s1 and not s0)or
       (D and  s1 and  s0);
end mux;
```
WHEN (simple and selected)

As mentioned above, WHEN is one of the fundamental concurrent statement. It appears in two forms WHEN/ELSE (simple form) or WITH/SELECT/WHEN (selected form).

----------------------
WHEN/ELSE
----------------------
Assignment WHEN condition ELSE 
Assignment WHEN condition ELSE  
...;
----------------------
WITH/SELECT/WHEN
----------------------
WITH identifier SELECT  
Assignment WHEN value,  
Assignment WHEN value  
...;

**NOTE:** Whenever WITH/SELECT/WHEN is used, all permutation must be tested, so the keyword OTHERS is often useful. Another important keyword is **UNAFFECTED**, which should be used when no action is to take place.

----------------------
WHEN/ELSE
----------------------
Outp <= "000" WHEN (inp='0' OR reset='1')ELSE "001" WHEN ctr='1' ELSE "010";

----------------------
WITH/SELECT/WHEN
----------------------
WITH control SELECT  
Outp <= "000" WHEN reset,  
"111" WHEN set,  
UNAFFECTED WHEN OTHERS;

Another important aspect related to the WHEN statement is that the "WHEN value" shown in the syntax above can be indeed take up three forms

WHEN value -- single value
WHEN value1 to value2 -- range, for enumerated data types only
WHEN value1 | value2 | ... -- values1 or values2 or ...

**Example: Multiplexer#2**

Same as example 1 WHEN is employed instead of logical operators.
Two solutions are provided:

---WHEN/ELSE---

**VHDL Code**

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity mux1 is
  port (A,B,C,D :in bit;
        sel :in std_logic_vector (1 downto 0);
        y: out std_logic);
end mux1;

architecture mux of mux1 is
begin
  y<= A when sel="00" else
     B when sel="01" else
     C when sel="10" else
     D when sel="11" else
        'Z';
end mux;
```

---Example---

**VHDL Code**

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity mux1 is
  port (A,B,C,D :in bit;
        sel :in std_logic_vector (1 downto 0);
        y: out std_logic);
end mux1;

architecture mux of mux1 is
begin
  y<= A when sel="00" else
     B when sel="01" else
     C when sel="10" else
     D when sel="11" else
        'Z';
end mux;
```
architecture mux of mux1 is
begin
    with sel select
        y <= A when "00",
            B when "01",
            C when "10",
            D when "11",
            'Z' when others;
end mux;

GENERATE

GENERATE is another concurrent statement. It is equivalent to the sequential statement LOOP in the sense that it allows a section of code to be repeated a number of times, thus creating several instances of the same assignments. Its regular form is the FOR/GENERATE construct, with the syntax shown below.

Note: GENERATE statement must be labeled.

FOR/GENERATE

Label : FOR identifier IN range GENERATE
(concurrent assignments)
END GENERATE;

Note: range must be static.
Homework Exercises

1. Write a model that represents a simple ALU with two integer inputs and one output, and a function select input of type bit_vector (2 bits) acting as declared in the table.

<table>
<thead>
<tr>
<th>select</th>
<th>function</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>+</td>
</tr>
<tr>
<td>01</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>*</td>
</tr>
<tr>
<td>11</td>
<td>/</td>
</tr>
</tbody>
</table>

a – Using When/else statement  
b – Using With/select/when  
c – Include a test bench.

2. a- Design a 2X1 multiplexer using one of the statements mentioned above.  
b- Using components form part a : construct a 8x1 multiplexer.  
c- Include a test bench.

3. a- Using full adder from lab1, design an n-carry ripple adder that is capable of summing 2 std_logic_vectors of size n using concurrent coding (and concurrent coding only!!)