CPE 462
VHDL: Simulation and Synthesis

Topic #05 - b) GENERATE statements

WESTERN NEW ENGLAND UNIVERSITY | WNE

GENERATE is a combinational loop

- GENERATE is another concurrent statement (along with operators and WHEN).
- It is equivalent to a 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.

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

- GENERATE must be labeled.

(partial) GENERATE example

You need to add a label! ... we haven't talked about them!

Example:

```
SIGNAL x: BIT_VECTOR (7 DOWNTO 0);
SIGNAL y: BIT_VECTOR (15 DOWNTO 0);
SIGNAL z: BIT_VECTOR (7 DOWNTO 0);

...
G1: FOR i IN x'RANGE GENERATE
    z(i) <= x(i) AND y(i+8);
END GENERATE;
```

- The range of GENERATE range must be static.
- STATIC means... value need to be defined at compilation time.

Another (partial) GENERATE example

signal "choice" is not static ...

```
NotOK: FOR i IN 0 TO choice GENERATE
   (concurrent statements)
END GENERATE;
```

- The range of GENERATE range must be static.
- STATIC means... value need to be defined at compilation time.
- If the signal "choice" is **NOT STATIC**, for example the output of another signal, then this example will **NOT** work.

Another GENERATE example

We want: the output vector must be a shifted version of the input vector, with twice its width and an amount of shift specified by another input.

Example: if the input bus has width 4, and the present value is "1111", then the output should be one of the lines of the following matrix:

row(0): 0 0 0 0 1 1 1 1
row(1): 0 0 0 1 1 1 1 0
row(2): 0 0 1 1 1 1 0 0
row(3): 0 1 1 1 1 0 0 0
row(4): 1 1 1 1 0 0 0 0

The first row corresponds to the input itself, with no shift and the most significant bits filled with '0's. Each successive row is equal to the previous row shifted one position to the left.

Complete code

```
1  ------------------------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ------------------------------------------------
5  ENTITY shifter IS
6     PORT ( inp: IN STD_LOGIC_VECTOR (3 DOWNTO 0);
7            sel: IN INTEGER RANGE 0 TO 4;
8            outp: OUT STD_LOGIC_VECTOR (7 DOWNTO 0));
9  END shifter;
10 ------------------------------------------------
11 ARCHITECTURE shifter OF shifter IS
12    SUBTYPE vector IS STD_LOGIC_VECTOR (7 DOWNTO 0);
13    TYPE matrix IS ARRAY (4 DOWNTO 0) OF vector;
14    SIGNAL row: matrix;
15 BEGIN
16    row(0) <= "0000" & inp;
17    G1: FOR i IN 1 TO 4 GENERATE
18       row(i) <= row(i-1)(6 DOWNTO 0) & '0';
19    END GENERATE;
20    outp <= row(sel);
21 END shifter;
22 ------------------------------------------------
```

Input "inp" is "1111"

Output is:

row(0): 0 0 0 0 1 1 1 1

row(1): 0 0 0 1 1 1 1 0

row(2): 0 0 1 1 1 1 0 0

row(3): 0 1 1 1 1 0 0 0

row(4): 1 1 1 1 0 0 0 0

GENERATE with WHEN

You can also use WHEN statements inside your GENERATE loops.

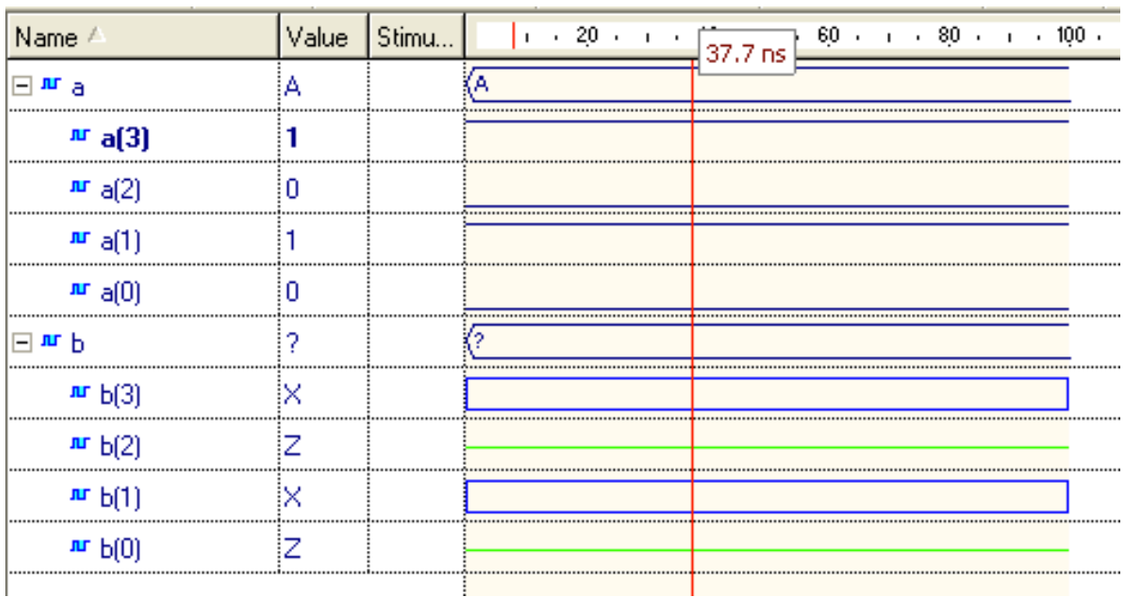
For example:

```
OK: FOR i IN 0 TO 7 GENERATE
   output(i)<='1' WHEN (a(i) AND b(i))='1' ELSE '0';
END GENERATE;
```

- The counter ‘i’ will go from 0 to 7
- At each element of output(i) a small comparison will be made

Another GENERATE example

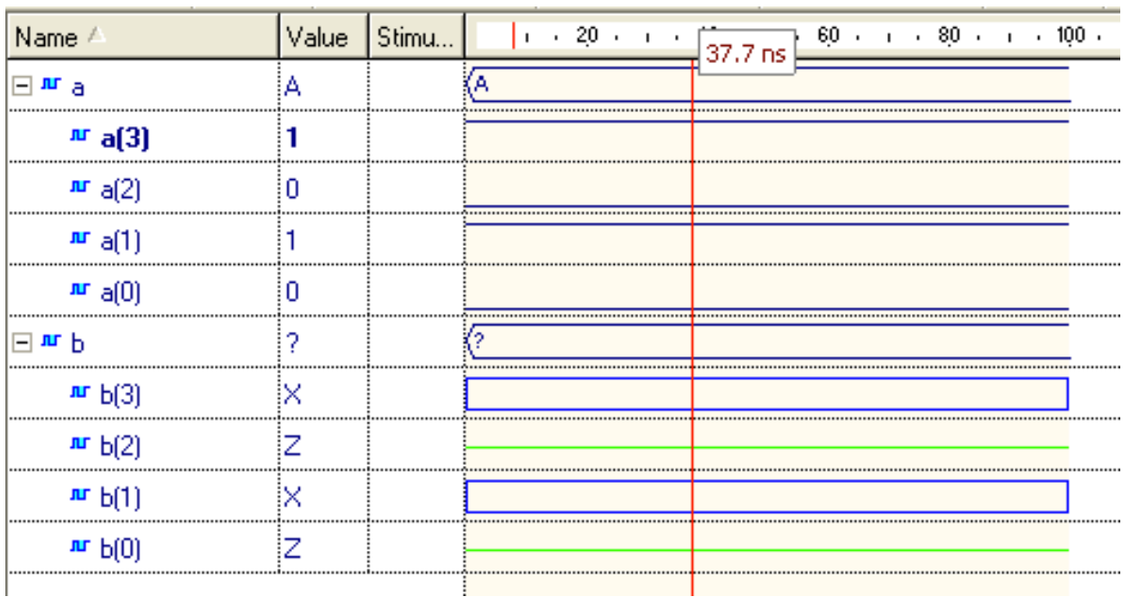
- When a bit of the input is '1', the equivalent output bit should be 'X'
- When a bit of the input is '0', the equivalent output bit should be 'Z'




```
library ieee;
use ieee.std_logic_1164.all;

entity test is
end entity;

architecture myarch of test is
 signal a : std_logic_vector(3 downto 0):="1010";
 signal b : std_logic_vector(3 downto 0);
begin

    loop1 : for i in 0 to 3 generate
        b(i) <= 'X' when a(i)='1' else 'Z';
    end generate;

end architecture;
```

You can use data attributes!

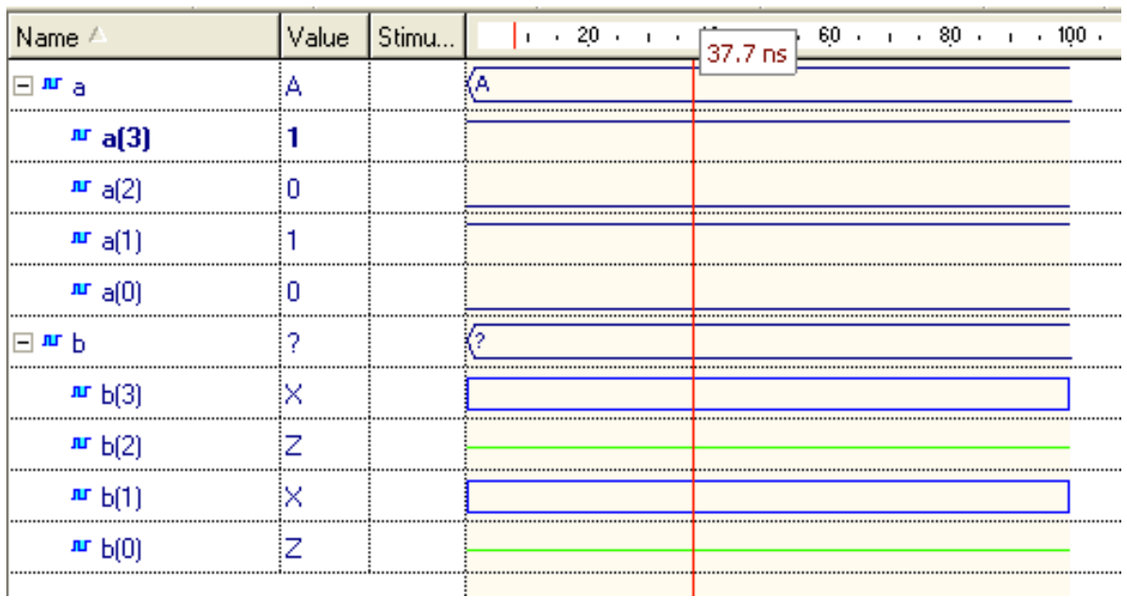
- Using the a'range attribute




```
library ieee;
use ieee.std_logic_1164.all;

entity test is
end entity;

architecture myarch of test is
 signal a : std_logic_vector(3 downto 0):="1010";
 signal b : std_logic_vector(3 downto 0);
begin

    loop1 : for i in a'range generate
        b(i) <= 'X' when a(i)='1' else 'Z';
    end generate;

end architecture;
```

You can use ALL data attributes!

- Using the a'low and a'high attributes




```
library ieee;
use ieee.std_logic_1164.all;

entity test is
end entity;

architecture myarch of test is
 signal a : std_logic_vector(3 downto 0):="1010";
 signal b : std_logic_vector(3 downto 0);
begin

    loop1 : for i in a'low to a'high generate
        b(i) <= 'X' when a(i)='1' else 'Z';
    end generate;

end architecture;
```