Lab # 4

Basic Modeling Constructs

October, 2015
Entity declarations

An entity declaration describes the external interface of a module showing its ports names, modes and types.

```
entity_decl<=
entity id is
    [ port ( port_interface_list ) ]
    { entity_declarative_item }
End [ entity ] [ id ] ;
```

```
port_interface_list<=
    { id { ... } : [ mode ] subtype { := expr } }{ ; ... }
```

```
mode<= in / out / inout / buffer
```

entity_declarative_item may include type declarations, signal declarations and constants, but no variables.

Example:

```
entity program_rom is
    port ( address : in integer range 0 to 2**15-1 ;
    data : out bit_vector (7 downto 0) ;
    enable : in bit );
    subtype instruction_byte is bit_vector (7 downto 0);
    type program is array [0 to 2**15-1] of instruction_byte;
end program_rom;
```

Architecture bodies

The architecture body describes the internal operation of a module. The operation can be described by processes containing sequential statements, or by a collection of components representing sub-circuits.

```
arch_body<=
architecture id of entity_name is
    { block_declaration_item }
begin
    { concurrent_stmt}
End [architecture ] [ id ] ;
```

block_declaration_item may include type, constant, or signal declarations.
Example:

```vhdl
architecture program_rom of program_rom is
begin
    process
        variable rom:program;
    begin
        if(enable = '1') then
            data <= rom(address);
        else
            data <= "00000000"
        end if;
        wait;
    end process;
end program_rom;
```

**Concurrent statements**

They describe the module's operation. Examples of Concurrent statements are:
- Process statement
- Concurrent signal assignment statements
- Concurrent Assertion statements

They are so called because they can be activated and perform their actions together, i.e., concurrently. Contrast this with the sequential statements inside a process, which are executed one after another.

**Signal declarations**

They help in declaring internal signals in an architecture body.

```
nSignal_declaration<=
    signal identifier {,...} : subtype_indication [ := expression];
```

**Examples**

- Example 1

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

entity example1 is
    port(a,b : in bit_vector(1 downto 0);
y: out bit);
end example1;
```
processes `and_gate_a`, `and_gate_b`, `or_gate`, and `inv` operate independently on each other; as each of them is sensitive to different signals from those to which other processes are.
Example_2:

library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity example2 is
end example2;

--}} End of automatically maintained section

architecture example2 of example2 is
signal x,y,z : integer := 0;
begin

p1: process is
    variable a,b : integer := 0;
    begin
    a:=a+20;
b:=-b+10;
x<=a+b after 20 ns;
y<=a-b after 10 ns;
wait for 30 ns;
    end process p1;

p2: process is
    begin
    z<=x+y;
wait on x,y;
    end process p2;

end example2;

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y1</td>
<td>40 to 60</td>
</tr>
<tr>
<td>Y2</td>
<td>20 to 30</td>
</tr>
<tr>
<td>nr x</td>
<td>60</td>
</tr>
<tr>
<td>nr y</td>
<td>20</td>
</tr>
<tr>
<td>nr z</td>
<td>60</td>
</tr>
</tbody>
</table>
**Lab. Exercises**

- **L.Exercise:**
  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.

    | select | function |
    |--------|----------|
    | 00     | +        |
    | 01     | -        |
    | 10     | *        |
    | 11     | /        |

  2. Write a counter model with a clock input **clk** of type **bit**, and an output **q** of type integer. The behavioral architecture should contain a process that declares a count variable initialized to zero. The process should wait for changes on **clk**. When **clk** changes to '1', the process should increment the count and assign its value to the output port.

- **H.Exercise:**
  1. Develop a VHDL model for thermostat that has two 8-bit unsigned binary inputs representing the target temperature and the actual temperature in degrees Fahrenheit(ºF). The detector has two outputs: one to turn a heater on when the actual temperature is more than 5 ºF below target, and one to turn a cooler on when the actual temperature is more than 5 ºF above the target (Use. Conditional signal assignment statement)

    \[ Y = \begin{cases} 
    a - c & \text{when } a > c \\
    a - b & \text{when } a > b \\
    a + 3 & \text{else}
    \end{cases} \]

    Rewrite the code using one case statement, and then by using Nested if statement.

  2. Develop a model (entity and architecture) for a register file that stores 1028 words of 32 bits each. The register file has data input and output ports, each of which is a 32-bit word; read-address and write-address ports, each of which is an integer in the range 0 to 1027; and a write-enable port of type bit. The data output port reflects the content of the location whose address is given by the read-address port. When the
write-enable port is '1', the input data is written to the register file at the location whose address is given by the write-address port.