Generic Example Process:

**Process** – Collection of \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ that \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ with other concurrent statements and other processes

* Always \_\_\_\_\_\_\_\_\_ or \_\_\_\_\_\_\_\_\_
* Creates a \_\_\_\_\_\_ that listens for \_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_)
* When the \_\_\_\_\_\_\_\_\_ changes the box is told to run

\*\*\* Processes are scary!

* Add \_\_\_\_\_\_\_\_\_
* Make circuit hard to \_\_\_\_\_\_\_\_\_

🡺 If you want a \_\_\_\_\_\_\_\_ you need a \_\_\_\_\_\_\_\_

process (*signal name, signal name, …., signal name*)

*Type declarations*

*Variable declarations*

*Constant declarations*

*Functions declarations*

*Procedure definitions*

begin

*sequential statement*

*…*

*sequential statement*

end process;

Specific Example Process: We get:

process( )

begin

end process;

What if we have assignments not in a process: We get:

So we make a process: We get:

process( )

begin

end process;

**Rules for processes:**

1. Think before using variables (don’t use them if you can help it)
2. Avoid “innovative” use of language constructs
3. \*Avoid overriding a signal in a process (don’t assign a value twice)
4. \*Only use processes for sequential circuits

\*Indicates a rule for this class only

If statements:

if *boolean expression* then *sequential statement*

elsif *boolean expression* then *sequential statement* elsif executed only if boolean

… expression true and previous

elsif *boolean expression* then *sequential statement* boolean expressions were false

else *sequential statement*

end if;

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity is

Port (

);

end ;

architecture of is

begin

process( )

* How do we make a Latch:

if then

end if;

begin

if ( ) then

end if;

end process;

end ;

process( ) Result:

variable ;

begin

end process;

process( ) Result:

begin

end process;