Concurrent & Sequential Statements

Introductory VHDL Methodology

Objectives

After completing this module, you will be able to…

- Write a VHDL process
- Create appropriate wait condition statements
- Determine when signals are updated
- Differentiate between signal 'transactions' and 'events'
- Determine how and when signals are read within the process
- Define the VHDL term 'delta'

Hardware Modeling

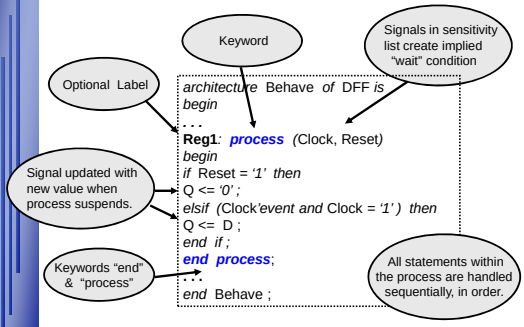
- **To effectively model a hardware environment, VHDL utilizes both concurrent and sequential statements**
- **Concurrent statements are so called because they are treated as simultaneous operations with respect to each other, although they may exist at different locations in the overall code**
- **Sequential statements are so termed because they are treated in sequence, like conventional software. All statements within a process are treated sequentially**

Language Structure

```
architecture RTL of ENTITY_1 is
. . .
begin
concurrent statements ;
. . .
process
  begin
    sequential statements ;
    . . .
end process ;
. . .
concurrent statements ;
. . .
process
  begin
    sequential statements ;
    . . .
end process ;
...
end architecture RTL ;
```

Any statement outside of a process is inherently "concurrent"

Process Basics

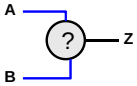


Inside Vs. Outside Process

```
architecture ...
process ( )
begin
        Z <= A;
        Z <= B;
        . . .
end process ;
end architecture ;
```

The last assignment takes effect

```
architecture . . .
      begin
        Z <= A;
        Z <= B;
. . .
end architecture ;
```



Both assignments are concurrent. A resolution function will be required on the output signal "Z"

Transactions

- The signal assignment statement "Z <= A" causes a transaction to be scheduled

```
process (. . .)
begin
        Z <= A ;
        F <= G ;
          . . .
end process ;
```

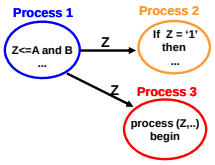
- Specifically, the current value of "A" is read (sensed) and scheduled to be driven onto "Z" when the process suspends

Events

- If the value of "Z" or "F" is actually changed as a result of the assignment (transaction), then an "event" occurs on that signal

```
process
   begin
        Z <= A;
        F <= G;
          . . .
end process ;
```

- Thus, all signal assignments cause a transaction to be scheduled, but not every transaction will result in an event on the target signal



- Only an event on a signal causes a process to trigger, if that signal is including in its sensitivity list

Suspending the Process

- Every process must have some means of triggering which is indirectly the means of suspension
- For synthesis, the signals in the sensitivity list form an implied 'wait' condition, for behavioral modeling, the explicit "wait" statement is often used
- There are four forms of the wait statement

wait on...	An event on given signal,
wait until...	A specific condition,
wait for ...	A specified time amount
wait	Indefinite suspension,

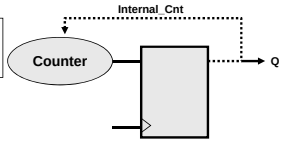
```
wait on  A, B ;
        wait until    CLK = '1' ;
              wait for 10 ns  ;
```


Treating Signals

- VHDL is fairly strict in its treatment of signals
- Basic rules for signals
  - *Signals of mode "out" may be assigned to but not be read*
  - *Signals of mode "in" may be read but not assigned to*
  - *All assigned signals must be the same type*
  - *All assigned signals must have the same size*

Treating Signals

```
entity  Count_1 is
port (Clk, D : in  bit ;
      Q : out  integer range...);
end Count_1;
```




```
architecture WRONG of  Count_1 is
begin
process (Clk)
begin
  If Clk'event and Clk ='1' then
  Q <= Q + 1;
end if ;
end process ;
```

Will produce compiler error

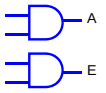
```
architecture RTL of  Count_1 is
signal  Internal_Cnt : integer range .. ;
begin
process (Clk)
 begin
  If Clk'event and Clk ='1' then
   Internal_Cnt <=  Internal_Cnt + 1 ;
 end if ;
end process ;
Q <= Internal_Cnt ;
```


Using Signals

```
process  ( B, C, D, F)
   begin
A <= B and  D ;
A <= B and  C ;
B <= F ;
E <= B and  C ;
end process;
```

The current value of the signals are read

What are final inputs to the gates ?



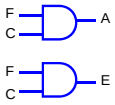
FYI: For combinatorial logic, do not update and assign a given signal in the same process!

Answer

```
process ( B, C, D, F)
   begin
A <= B and  D ;
A <= B and  C ;
B <= F ;
E <= B and  C ;
end process;
```

The current value of the signals are read

What are final inputs to the gates ?



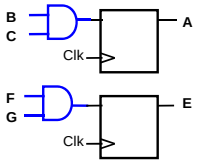
FYI: For combinatorial logic, do not update and assign a given signal in the same process

Using Signals

```
process ( Clk)
   begin
if (Clk'event and Clk = '1') then
A <= B and  C ;
E <= F and  G ;
end if;
end process;
```

The current value of the signals are read

Any signal under the statement *"if clock'event and clock = '1' then" will infer the use of a register*



Review Questions

- **When is a signal updated as the result of an assignment within a process?**
- **What causes a suspended process to resume execution?**
- **A port of mode "out" may be _____ , but not _____ within the process?**
- **A port of mode "in" may be _____ , but not _____ within the process?**
- **What happens if a given signal is assigned to from different processes ?**

Answers

- **When is a signal updated as the result of an assignment within a process?**
  - *When the process 'suspends'*
- **What causes a suspended process to resume execution?**
  - *Satisfaction of the 'wait' condition*
- **A port of mode "out" may be assigned , but not read within the process.**
- **A port of mode "in" may be read , but not assigned within the process.**
- **What happens if a given signal is assigned to from different processes?**
  - *It creates multiple drivers to that output, requiring resolution*

Summary

- **The process is the basic unit of operation in VHDL**
- **A signal assignment within a process will cause a transaction on that signal, but not necessarily an event**
- **Some form of the "wait" statement is used to control processes**
- **Signals assigned under the 'event statement will infer the use of a register for that signal**