Chapter 3
Concurrent Statements

- Sequential constructs
  - Loops:
    - For/while
  - Case
  - If/then/elsif/else
- Process execution
- Sensitivity lists vs. wait statements
- Compound waits:
  - wait on X,Y until Z='0' for 100 ns;

Concurrent Statements
- Statements which occur within the architecture block
- "Concurrent" because all are evaluated for execution at the same time
- The order in which concurrent statements appear is not important (to the simulator)

Process Statement Review
- The PROCESS block is, as a whole, a concurrent statement
- As with all statements in an architecture block, all PROCESSes within an ARCHITECTURE are considered to execute in parallel.
- Processes contain only sequential statements.
- All concurrent statements are equivalent to some process statement.

Architecture Basics
architecture MINE of MY_ENTITY is
begin
LABEL_1: process ([signal_list])
  [constant_variable_declarations]
  begin
  [sequential_statements]
  end process;
[other_concurrent_statements]
end MINE;
The following are equivalent:

- process (S1, S2) begin
  X1 <= S1 or S2;
  end process;

- process begin
  X1 <= S1 or S2;
  wait on S1, S2;
  end process;

- X1 <= S1 or S2;

When are these statements executed?

In general:

```
[Label!:] signal_name <= expression [after some_time],
expression [after sometime], ...
```

Order doesn't matter:

```
a <= b + c;
c <= d + 1;
c <= d + 1;
a <= b + c;
```

A concurrent signal assignment statement is evaluated when a signal on the right-hand side changes value.
Concurrent Signal Assignment Statements

S1 <= '1' after 10 ns, '0' after 20 ns, 'Z' after 30 ns;

Time values must be ascending in time.

Concurrent Conditional Signal Assignment

In general:
[label:] signal_name <= wave1 when cond1 else wave2 when cond2 else
wave3 when cond3 else
wave4 when cond4 else

Order matters:
The first true conditional determines the output’s value

Concurrent Conditional Assignment

Example:
Fred: S1 <= A or B after 10 ns, 'Z' after 30 ns when MODE='1'
else '1' after 10 ns, 'Z' after 30 ns when CAR=Red
else 'Z' after 30 ns;

Fred: process (A, B, MODE, CAR) begin
if MODE='1' then S1 <= A or B after 10 ns, 'Z' after 30 ns;
elsif CAR=Red then S1 <= '1' after 10 ns, 'Z' after 30 ns;
else S1 <= 'Z' after 30 ns;
end if; end process;

Conditional Concurrent Assignment

- Example:
  Fred: S1 <= A or B after 10 ns, C after 20 ns, 'Z' after 30 ns
  when MODE='1'
  else '1' after 10 ns, '0' after 20 ns, 'Z' after 30 ns
  when CAR=Red
  else 'Z' after 30 ns;

  DBUS <= REG1 when REG1SEL
  else REG2 when REG2SEL
  else REG3 when REG3SEL
  else REG4;

Can A, B, Mode, be variables?

When are these statements executed?

Equivalent Process Statement

Example:
with SEL select
DBUS <= INP0 when "00",
INP1 when "01",
INP2 when "10",
INP3 when others;

Selected Concurrent Signal Assignment

- In general:
  [Label :] with expression select
  signal_name <= wave1 when choice1,
  wave2 when choice2,
  o o o
  waven-1 when choicen-1,
  waven when others;

Choices must be mutually exclusive and must cover all possible values of expression.

Selected Concurrent Signal Assignment

- Example:
  with SEL select
  DBUS <= INP0 when "00",
  INP1 when "01",
  INP2 when "10",
  INP3 when others;
**Equivalent Process Statement**

```vhdl
with SEL select
  DMUX <= INP0 when "00", INP1 when "01",
          INP2 when "10", INP3 when others;
```

```vhdl
process (SEL, INP0, INP1, INP2, INP3) begin
  case SEL is
    when "00" => DMUX <= INP0;
    when "01" => DMUX <= INP1;
    when "10" => DMUX <= INP2;
    when others => DMUX <= INP3;
  end case; end process;
```

**Concurrent behavior**

```vhdl
ENTITY mux4_1bit IS
  PORT ( i0, i1, i2, i3, a, b : IN std_logic; z: OUT std_logic);
END mux4_1bit;
```

```vhdl
ARCHITECTURE conc OF mux4_1bit IS
  SIGNAL sel: INTEGER;
BEGIN
  WITH sel SELECT
    Z <= i0 AFTER 10 ns WHEN 0,
        i1 AFTER 10 ns WHEN 1,
        i2 AFTER 10 ns WHEN 2,
        i3 AFTER 10 ns WHEN 3,
        'X' AFTER 10 ns WHEN OTHERS;
  sel <= 0 WHEN a = '0' AND b = '0' ELSE
        1 WHEN a = '1' AND b = '0' ELSE
        2 WHEN a = '0' AND b = '1' ELSE
        3 WHEN a = '1' AND b = '1' ELSE
        4;
END conc;
```

**Assert Statement**

- Method of producing interactive feedback during simulation
- Diagnostic messages:
  - "Error: Bus contention – put fire out"
  - "Warning: reset occurred when P0 is high"
- Assert statement is both a sequential statement and a concurrent statement.

**Severity Level**

- Optional SEVERITY clause:
  - `Severity_level` can be one of (NOTE, WARNING, ERROR, or FAILURE) (Default is ERROR).
  - Action on different severity.level is implementation dependent.

**Assert**

- Example:
  ```vhdl
  TESTPOINT1 <= RESET and ABORT;
  assert not ( TESTPOINT1 = '1')
  report "watch out: reset and abort active at the same time" severity note;
  ```

- When is the assert statement executed?
- When is the report printed?
Example:

-- build SR flip/flop:
U1: NAND2 port map ( SET_B, QB, Q );
U2: NAND2 port map ( CLR_B, Q, QB );
assert not ( SET_B = '0' and CLR_B = '0' )
report "input violation: SET_B & value(SET_B) &
  CLR_B = & value(CLR_B);"
-- NOTE: function "value" is a VHDL93 extension

Summary

• Concurrent statements
  • Order not important
  • A process block is a concurrent statement
• Selected/conditional signal assignments
• Assert

• Next time: Data types and attributes, pp. 51-72.