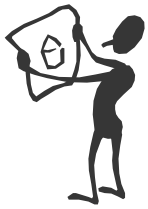
Chapter 6
Data-Flow Modeling

VHDL

Outline

- Concurrent Signal Assignment
- Conditional Signal Assignment
- Selected Signal Assignment
- Unaffected value
- Block Statement
- Concurrent Assertion Statement



Concurrent Statement

- **The Data-Flow modeling is a collections of concurrent statements.**
- **All the statements must be write only in the architecture body.**
- **There is no meaning to the order of the statements.**
- **There are 3 Data-Flow statement:**
  - Concurrent Signal Assignment
  - Conditional Signal Assignment
  - Selected Signal Assignment

Concurrent Signal Assignment

- **The syntax is:**

target-signal* <= *waveform;

  - Examples:

```
z <= a;
z <= a AFTER 10 ns;
z <= (a AND b) AFTER 20 ns;
z <= a AFTER 10 ns, '1' AFTER 20 ns, '0' AFTER 30 ns;
```


Conditional Signal Assignment

- Also called a **When-Else** Statement.
- Concurrent statement, thus all signals.
- Similar to a sequential IF-THEN-ELSE statement.
- Select one of several values to drive an output signal.
- Selection based on first condition that is TRUE.
- Syntax:

```
target_signal <=  value1 WHEN condition1 ELSE
                  value2 WHEN condition2 ELSE
                    ...
                  value9;
```


Conditional Signal Assignment

- Conditions
  - Boolean expressions
  - Set of conditions NOT necessarily mutually exclusive or all inclusive
  - First true condition determines the value assigned
  - Last value assigned if all conditions FALSE
- Values
  - Anything normally legal on right side of concurrent signal assignment
- Synthesis results
  - If conditions are mutually exclusive, it synthesizes to a simple multiplexer like the WITH-SELECT-WHEN statement
  - Otherwise, it synthesizes to a more complex priority encoder, with the first condition having highest priority

Conditional Signal Assignment

```
opcode <= add WHEN (addsub = '0') ELSE
          sub WHEN (addsub = '1' ) ELSE
          nop;
```

```
out <= "1011" WHEN (a = '0') ELSE
       "11--" WHEN (b = '0') ELSE
       x OR y WHEN (a AND b)='1' ELSE
       "0000";
```

```
-- 4-to-2 priority encoder
outcode <= "11" WHEN in3='1' ELSE
           "10" WHEN in2='1' ELSE
           "01" WHEN in1='1' ELSE
           "00" WHEN in0='1' ELSE
           "00";
```


Selected Signal Assignment

- Also called a **With-Select-When** statement.
- Concurrent statement, thus all signals.
- Similar to a sequential CASE statement.
- Select one of several values to drive an output signal.
- Selection based on **all possible values** of a selector expression.
- Syntax:

```
WITH expression SELECT
   target_signal <= value1 WHEN choice1,
                    value2 WHEN choice2,
                    ...
                    value9 WHEN choice9;
```


Selected Signal Assignment

- Selector expression
  - Signal name, or expression with signal names
- Choices
  - Match type of selector
  - Set of choices must be mutually exclusive and all inclusive
  - Number, string, expression, **choice1 | choice2**, **OTHERS**
- Values
  - Anything normally legal on right side of concurrent signal assignment
- Synthesis result
  - Statement synthesizes to an N-bit M-to-1 multiplexer
    - Values are data inputs
    - Selector and choices form select input codes
    - target_signal is the data output

Selected Signal Assignment

```
-- 2-to-1 multiplexer
WITH addsub SELECT
   opcode <= add WHEN '0',
             sub WHEN '1';
```

```
WITH(a AND b) SELECT
   out <= "1011" WHEN "00" | "01",
          "11--" WHEN "1Z",
          x OR y WHEN "11",
          "0000" WHEN OTHERS;
```

```
WITH myint SELECT
   outvec <= X"1F" WHEN 35,
             X"27" WHEN 2 TO 5,
             X"FF" WHEN OTHERS;
```


The UNAFFECTED Value

- **It is possible to assign a value of unaffected to a signal in a concurrent signal assignment statement. Such an assignment causes no change to the driver for the target signal.**
- **For example:**

```
WITH sel SELECT
   mix1 <= "1011" WHEN "00",
           "11--" WHEN "01",
           "0000" WHEN "11",
           UNAFFECTED WHEN OTHERS;

mix2 <= "1011" WHEN (a = '0') ELSE UNAFFECTED;
```


Don't Cares in Conditions

- Conditions with don't care values '-' should not be used in IF and WHEN-ELSE statements
  - They are literally compared to the '-' value in simulation
  - They always evaluate to **FALSE** for synthesis
  - Example: **`IF a = "0--" THEN …`**
- However, don't cares can be used with the **std_match** function
  - std_match available in **numeric_std** and **std_arith** packages
  - Format: **`std_match (name, bitstring)`**
  - Returns true for either 0 or 1 in place of the '-'
  - Example: **`IF std_match(a, "0--") THEN ...`**