EE434
ASIC & Digital Systems

VHDL
Sequential Processing

Spring 2016
Dae Hyun Kim
daehyun@eecs.wsu.edu

Sequential Statements

- Sequential statements are executed sequentially.

- Format

```
ARCHITECTURE  architecture_name  OF  entity_name  IS
BEGIN
   PROCESS  ( signal names )  <----  Sensitivity list
   BEGIN                             (should include all input signals
      -- Sequential statements       used inside the process)
   END  PROCESS;
END  architecture_name;
```

Example

- 2:1 MUX

```
ARCHITECTURE  vmux2_arch  OF  vmux2  IS
BEGIN
   PROCESS  (a, b, s)
   BEGIN
      IF s = ‘0’  THEN
         z <= a;
      ELSE
         z <= b;
      END  IF;
   END PROCESS;
END  vmux2_arch;
```

Example

- 2:1 MUX

```
ARCHITECTURE  vmux2_arch  OF  vmux2  IS
BEGIN
   PROCESS  (a, b, s)
   BEGIN
      IF s = ‘0’  THEN
         z <= a;
      ELSIF  s = ‘1’  THEN
         z <= b;
      ELSE
         z <= ‘X’;
      END  IF;
   END PROCESS;
END  vmux2_arch;
```

Example

- If multiple assignment statements are executed for a signal, only the last one will be active.
- Actual assignments happen at the end of the process.

```
ARCHITECTURE  vmux2_arch  OF  vmux2  IS
BEGIN
   PROCESS  (a, b, s)
   BEGIN
      z <= a;

      IF s = ‘1’  THEN
         z <= b;
      END  IF;
   END PROCESS;
END  vmux2_arch;
```

Sequential Statements

- IF
- CASE
- LOOP
  - FOR
  - WHILE
- EXIT
- WAIT

IF

- Format

```
IF  condition  THEN
   sequential_statements;
ELSIF  condition  THEN
   sequential_statements;
ELSIF  condition  THEN
   sequential_statements;
ELSE
   sequential_statements;
END  IF;
```

Example

- Example
  - 2:1 MUX

```
ARCHITECTURE  vmux2_arch  OF  vmux2  IS
BEGIN
   PROCESS  (a, b, s)
   BEGIN
      IF s = ‘0’  THEN
         z <= a;
      ELSIF  s = ‘1’  THEN
         z <= b;
      ELSE
         z <= ‘X’;
      END  IF;
   END PROCESS;
END  vmux2_arch;
```

IF

- Example (sequential + concurrent)
  - FA

```
ARCHITECTURE  vFA_arch  OF  vFA  IS
BEGIN
   PROCESS  (a, b, ci)
   BEGIN
      IF (a = ‘1’) AND (b = ‘1’)  THEN
         co <= ‘1’;
      ELSIF  (b = ‘1’) AND (ci = ‘1’)  THEN
         co <= ‘1’;
      ELSIF  (a= ‘1’) AND (ci = ‘1’)  THEN
         co <= ‘1’;
      ELSE
         co <= ‘0’;
      END  IF;
   END PROCESS;

   s <= a XOR b XOR ci;
END  vFA_arch;
```

CASE

- Format

```
CASE expression IS
   WHEN choices =>
      sequential_statements;
   WHEN choices =>
      sequential_statements;
   ...
   WHEN OTHERS =>
      sequential_statements;
END CASE;
```

CASE

- Example
  - 2:1 MUX

```
ARCHITECTURE  vmux2_arch  OF  vmux2  IS
BEGIN
   PROCESS  (a, b, s)
   BEGIN
      CASE  s  IS
         WHEN  ‘0’  =>
            z <= a;
         WHEN  OTHERS  =>
            z <= b;
      END  CASE;
   END PROCESS;
END  vmux2_arch;
```

CASE

- Example
  - 2:1 MUX

```
ARCHITECTURE  vmux2_arch  OF  vmux2  IS
BEGIN
   PROCESS  (a, b, s)
   BEGIN
      CASE  s  IS
         WHEN  ‘0’  =>
            z <= a;
         WHEN  ‘1’  =>
            z <= b;
         WHEN  OTHERS  =>
            z <= ‘X’;
      END  CASE;
   END PROCESS;
END  vmux2_arch;
```

CASE

- Example
  - 4:1 MUX (s : std_logic_vector (1 downto 0))

```
ARCHITECTURE  vmux4_arch  OF  vmux4  IS
BEGIN
   PROCESS  (a, b, c, d, s)
   BEGIN
      CASE  s  IS
         WHEN  “00”  =>
            z <= a;
         WHEN  “01”  =>
            z <= b;
         WHEN  “10”  =>
            z <= c;
         WHEN  “11”  =>
            z <= d;
         WHEN  OTHERS  =>
            z <= ‘X’;
      END  CASE;
   END PROCESS;
END  vmux4_arch;
```

FOR Loop

- Format

```
FOR  range  LOOP
    sequential_statements;
END  LOOP;
```

- Example

```
FOR  i  IN  1  TO  10  LOOP
    a := i + 3;
    b := i * 2;
    d := a + b;
END  LOOP;
```

```
FOR  i  IN  10  DOWNTO  1  LOOP
    a := i + 3;
    b := i * 2;
    d := a + b;
END  LOOP;
```

- Note: Loop variables (i in the above example) do not need to be explicitly declared outside the loop.

FOR Loop

- Example
  - count # ‘1’s
    - input: x std_logic_vector (2 DOWNTO 0)
    - output : z std_logic_vector (1 DOWNTO 0)

FOR Loop

```
ARCHITECTURE  vc_arch  OF  vc  IS
BEGIN
   PROCESS  (x)
      VARIABLE  sum : INTEGER;
   BEGIN
      sum := 0;  -- initialization
      FOR  i  IN  2  DOWNTO  0  LOOP
         IF x(i) = ‘1’  THEN
            sum := sum + 1;
         END  IF;
      END  LOOP;
      IF ( sum = 0 )  THEN
         z <= “00”;
      ELSIF  ( sum = 1 )  THEN
         z <= “01”;
      ELSIF  ( sum = 2 )  THEN
         z <= “10”;
      ELSE
         z <= “11”;
      END  IF;
   END PROCESS;
END  vc_arch;
```

WHILE Loop

- Format

```
WHILE  condition  LOOP
   sequential_statements;
END  LOOP;
```

- Example

```
WHILE  ( a <= b )  LOOP
   a := a + 3;
   b := b – 2;
   c := a + b;
END  LOOP;
```

WHILE Loop

- Count # ‘1’s

```
ARCHITECTURE  vc_arch  OF  vc  IS
BEGIN
   PROCESS  (x)
      VARIABLE  tmp, sum : INTEGER;
   BEGIN
      tmp := 0;  -- initialization
      sum := 0;
      WHILE  ( tmp < 3 )  LOOP
         IF x(tmp) = ‘1’  THEN
            sum := sum + 1;
         END  IF;
         tmp := tmp + 1;
      END  LOOP;
      IF ( sum = 0 )  THEN
         z <= “00”;
      ELSIF  ( sum = 1 )  THEN
         z <= “01”;
      ELSIF  ( sum = 2 )  THEN
         z <= “10”;
      ELSE
         z <= “11”;
      END  IF;
   END PROCESS;
END  vc_arch;
```

Loop (How to Skip)

- If “Next;” is used, all the statements below the “next” statement will be skipped and the for loop go back to the first line of the loop.

- Example

```
FOR i IN 1 TO 10 LOOP
   IF ( i /= 8 ) THEN
      a := i + 3;
      b := i * 2;
      d := a + b;
   END IF;
END LOOP;
```

```
FOR i IN 1 TO 10 LOOP
   IF ( i = 8 ) THEN
      NEXT;
   END IF;

   a := i + 3;
   b := i * 2;
   d := a + b;
END LOOP;
```

Loop (How to Exit)

- If “Exit;” is used, we immediately exit from the loop.

- Example

```
FOR i IN 1 TO 10 LOOP
   IF ( i = 8 ) THEN
      NEXT;
   ELSIF ( ( i + a ) = 9 ) THEN
      EXIT;
   END IF;

   a := i + 3;
   b := i * 2;
   d := a + b;
END LOOP;
```

PROCESS

- If a process has a sensitivity list, the process is held at the beginning of the process until any of the signals in the sensitivity list changes.

- If a process does not have a sensitivity list, the process is executed infinitely.

```
PROCESS  (a, b, s)
BEGIN
   IF s = ‘0’  THEN
      z <= a;
   ELSIF  s = ‘1’  THEN
      z <= b;
   ELSE
      z <= ‘X’;
   END  IF;
END PROCESS;
```

WAIT

- WAIT FOR time_expression
  - The process is held at the WAIT statement for the time specified by the expression.

- Example

```
PROCESS
BEGIN
   ...
   WAIT  FOR  time_expression;
   ...
END  PROCESS;
```

```
PROCESS
BEGIN
   ...
   WAIT  FOR  10 NS;
   ...
   WAIT  FOR  ( a + b );
   ...
END  PROCESS;
```

WAIT

- WAIT UNTIL condition
  - The process is held at the WAIT statement until the condition is satisfied.

- Example

```
PROCESS
BEGIN
   ...
   WAIT  UNTIL  condition;
   ...
END  PROCESS;
```

```
PROCESS
BEGIN
   ...
   WAIT  UNTIL  rising_edge (clock);
   ...
END  PROCESS;
```

WAIT

- WAIT ON signals
  - The process is held at the WAIT statement until one of the signals changes.

- Example

```
PROCESS
BEGIN
   ...
   WAIT  ON  clk, reset;
   ...
END  PROCESS;
```

```
PROCESS
BEGIN
   IF  reset = ‘1’  THEN
      Q <= ‘0’;
   ELSIF  clk = ‘1’  THEN
      Q <= D;
   END  IF;
   WAIT  ON  clk, reset;
END  PROCESS;
```

Sensitivity List vs. WAIT

```
PROCESS  ( clock )
BEGIN
  …
END  PROCESS;
```

```
PROCESS
BEGIN
  …
 WAIT  ON  clock;
END  PROCESS;
```

- Why should we put the “WAIT ON” statement at the end of the process?
  - All processes are executed at least once.
  - To guarantee that the process having a wait statement is exactly the same as the sensitivity-list-based process, we need a wait statement at the end of the process.

Signals vs. Variables

- Signal assignment

```
t1  <=  a  AND  b;
```

  - Scheduled as an event for signal t1.

- Variable assignment

```
v1  :=  a  AND  b;
```

  - Happens immediately (no delay).
  - Variables can be used only in a PROCEESS or a SUBPROGRAM.

Example

```
ARCHITECTURE  and2beh  OF  and2  IS
 SIGNAL  t1 : BIT;
BEGIN
 PROCESS  ( a , b )
 BEGIN
  t1  <=  a  AND  b;
  IF  (t1 = ‘1’)  THEN
   z  <=  t1  AFTER  5 NS;
  ELSIF  (t1 = ‘0’)  THEN
   z  <=  t1  AFTER  4 NS;
  ELSE
   z  <=  t1  AFTER  6 NS;
  END  IF;
 END  PROCESS;
END and2beh;
```

This is wrong.

```
ARCHITECTURE  and2beh  OF  and2  IS
BEGIN
 PROCESS  ( a , b )
  VARIABLE  t1 : BIT;
 BEGIN
  t1  :=  a  AND  b;
  IF  (t1 = ‘1’)  THEN
   z  <=  t1  AFTER  5 NS;
  ELSIF  (t1 = ‘0’)  THEN
   z  <=  t1  AFTER  4 NS;
  ELSE
   z  <=  t1  AFTER  6 NS;
  END  IF;
 END  PROCESS;
END and2beh;
```

This is correct.