EE434
ASIC & Digital Systems

VHDL

Data Types

Spring 2016

Dae Hyun Kim
daehyun@eecs.wsu.edu

Object Types

- Signal
  - Interconnection wire
- Variable
  - Local storage for temporary data
- Constant

Signal

- Format

```
SIGNAL  signal_name  :  signal_type;


SIGNAL  signal_name  :  signal_type  :=  initial_value;
```

- Example

```
ARCHITECTURE  ...
   SIGNAL  a, b : std_logic;
BEGIN
   ...
```

Variable

- Signal assignments are scheduled.
- Variable assignments occur immediately.

- Format

```
VARIABLE  var_name  :  var_type;


VARIABLE  var_name  :  var_type  :=  initial_value;
```

- Variables can be declared in PROCESS declarations.

Variable

- Example

```
ARCHITECTURE  and4arch  OF  and4  IS
BEGIN
   PROCESS (a, b, c, d)
      VARIABLE st : std_logic;
      VARIABLE delay : time;
   BEGIN
      st := a  AND  b  AND  c  AND  d;

      IF  st = '1'  THEN
         delay := 3 ns;
      ELSE
         delay := 4 ns;
      END  IF;

      z  <=  st  AFTER  delay;
   END PROCESS;
END and4arch;
```

Constant

- Format

```
CONSTANT  constant_name  :  constant_type;

CONSTANT  constant_name  :  constant_type  :=  value;
```

- Example

```
CONSTANT  PI  :  REAL  :=  3.141592;
```

Data Types

- Scalar
  - INTEGER
  - REAL
  - ENUMERATED
  - PHYSICAL

```
VARIABLE a : INTEGER;

   ...
   a := 1;
   a := -1;
   ~~a := 3.5;~~
```

```
VARIABLE a : REAL;

   ...
   a := 3.5;
   a := -2.7E12;
   ~~a := 4;~~
```

Data Type (Enumerated)

- Define a new data type.

```
TYPE  instruction  IS  (add, sub, mul, div);
VARIABLE ins : instruction;
```

- Usage

```
ARCHITECTURE  test_arch  OF  myArch  IS
BEGIN
   PROCESS  (ins, a, b)
      TYPE  instruction  IS  (add, sub, mul, div);
   BEGIN
      CASE  ins  IS
         WHEN  add  =>
            z <= a + b;
         WHEN  sub  =>
            z <= a – b;
         WHEN  mul  =>
            z <= a * b;
         WHEN  div  =>
            z <= a / b;
      END CASE;
   END PROCESS;
END  test_arch;
```

Data Type (Physical)

- Used to represent physical quantities.
  - distance
  - time
  - ...

- Pre-defined physical types
  - TIME

New Types

- Define a new type.

```
TYPE type_name IS definition
```

- Example

```
TYPE TIME IS RANGE 0 TO 1000000000

TYPE my_int IS INTEGER;

TYPE BIT IS ('0', '1');

TYPE std_logic IS ('0', '1', 'X', 'Z');
```

Array

- Composite types
  - ARRAY
    - a group of elements of the same type
  - RECORD
    - a group of elements of different types

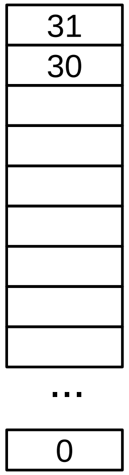
Array

- Format

```
TYPE  type_name  IS  ARRAY  (0  TO  n)  OF  data_type
```

- Example

```
TYPE  data_bus  IS  ARRAY (0 TO 31)  OF  BIT;


VARIABLE  x : data_bus;

VARIABLE  y : BIT;


y := x(0);  -- the first element

y := x(15);  -- the 16th element
```



Array

- Example

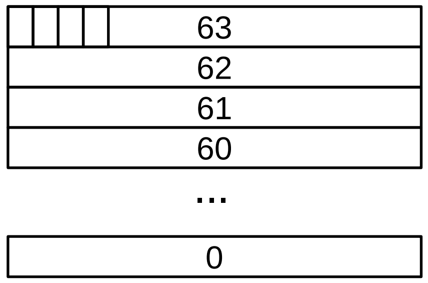
```
TYPE  data_word  IS  ARRAY (0 TO 31)  OF  BIT;
TYPE  data_byte  IS  ARRAY (0 TO 7)  OF  BIT;

VARIABLE  w : data_word;
VARIABLE  b : data_byte;
```

```
ENTITY  my_copy  IS
   PORT  ( w : IN data_word;
               s : IN  INTEGER;
               z : OUT data_byte );
END  my_copy;

ARCHITECTURE  my_copy_arch  OF  my_copy  IS
BEGIN
   PROCESS  (w, s)
   BEGIN
      FOR  i  IN  0  TO  7  LOOP
         z(i)  <=  w(i+s);
      END  LOOP;
   END  PROCESS;
END  my_copy_arch;
```

Array of Array

- Example

```
TYPE  data_byte  IS  ARRAY (0 TO 7)  OF  BIT;
TYPE  data_mem  IS  ARRAY (0 TO 63)  OF  data_byte;

ARCHITECTURE  my_mem_arch  OF  my_mem  IS
   VARIABLE  my_ram : data_mem;
   INTEGER  a;
BEGIN
   z <= my_ram (a);
   my_ram(0) = (‘0’, ‘1’, ‘1’, ‘0’, ‘0’, ‘1’, ‘0’, ‘0’);
   z(15) <= my_ram (0)(3);
END  my_mem_arch;
```



2D Array

- Example

```
TYPE  data_mem  IS  ARRAY (0 TO 63, 0 TO 7)  OF  BIT;


ARCHITECTURE  my_mem_arch  OF  my_mem  IS
   VARIABLE  my_ram : data_mem;
   VARIABLE  a, b : INTEGER;
BEGIN
   z  <=  my_ram (a, b);
END  my_mem_arch;
```