ΥΠΟΥΡΓΕΙΟ ΕΘΝΙΚΗΣ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ ΕΠΕΑΕΚ
ΕΥΡΩΠΑΪΚΗ ΕΝΩΣΗ
ΣΥΓΧΡΗΜΑΤΟΔΟΤΗΣΗ
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Η ΠΑΙΔΕΙΑ ΣΤΗΝ ΚΟΡΥΦΗ
Επιχειρησιακό Πρόγραμμα
Εκπαίδευσης και Αρχικής
Επαγγελματικής Κατάρτισης
ΕΛΛΑΔΑ 2008
Ανάπτυξη παντού. Ανάπτυξη για όλους.

Σύνθετοι τύποι και λειτουργίες

(Peter Ashenden, The Students Guide to VHDL)

Πίνακες

Πίνακες: Αποτελούνται από στοιχεία του ίδιου τύπου.

Μονοδιάστατοι Πίνακες

type table1 **is array** (0 to 7) **of** std_logic; --Ascending Order

type table2 **is array** (7 downto 0) **of** std_logic; --Descending Order

Ο δείκτης μπορεί να είναι τύπου απαρίθμησης

type table3 **is array** (std_ulogic) **of** std_logic;*

*Προσπέλαση: Table('u'), Table('x') κλπ.

type controller_state **is** (initial, idle, active, error);

type state_counts **is array** (idle to error) of integer;

Πίνακες

Η δήλωση αντικειμένων γίνεται με τον γνωστό τρόπο:

variable word1: table1;

signal word2: table2:=B"10010110";

➢Τα στοιχεία του πίνακα απαριθμούνται με την σειρά των δεικτών από αριστερά προς τα δεξιά.

➢Μπορούμε να προσπελάσουμε ένα υποσύνολο του πίνακα όπως table2 [4 downto 2].

Παράδειγμα

(Μνήμη 64 πραγματικών αριθμών)

```
subtype coeff_ram_address is integer range 0 to 63;
entity coeff_ram is
  port ( rd, wr : in bit;  addr : in coeff_ram_address;
        d_in : in real;  d_out : out real );
end entity coeff_ram;
```

Πίνακες

```
architecture abstract of coeff_ram is
begin
  memory : process is
  type coeff_array is array (coeff_ram_address) of real;
  variable coeff : coeff_array;
  begin
            for index in coeff_ram_address loop
                  coeff(index) := 0.0;
            end loop;
            loop
                  wait on rd, wr, addr, d_in;
                  if rd = '1' then d_out <= coeff(addr);
                  end if;
                  if wr = '1' then coeff(addr) := d_in;
                  end if;
            end loop;
  end process memory;
end architecture abstract;
```

Πίνακες

Αρχικοποίηση πινάκων

- Μπορεί να γίνει με υπονοούμενη αντιστοίχιση στοιχείων και θέσεων

type Matrix **is array** (1 to 5) **of** integer;

variable ArrayMatrix: Matrix := (10, 20, 30, 40, 50);

Οπότε Matrix(1)=10, Matrix(2)=20, Matrix(3)=30 …

- Η αρχικοποίηση μπορεί να γίνει με ρητή ανάθεση

variable ArrayMatrix: Matrix := (1=>10, 2=>20, 3=>30, 4=>40, 5=>50);

variable ArrayMatrix: Matrix := (1 | 3=>10, 4=>20, **others**=>50);

Οί λίστες (10, 20, 30, 40, 50), (1=>10, 2=>20, 3=>30, 4=>40, 5=>50) κλπ ονομάζονται συναθροιστές

Πίνακες

Πολυδιάστατοι Πίνακες:

type Matrix **is array** (1 to 2, 1 to 3) **of** integer;

variable ArrayMatrix: Matrix := ((0,0,1), (0,1,2))

0	0	1
0	1	2

- Η προσπέλαση των στοιχείων γίνεται με δύο δείκτες πχ ArrayMatrix(1,1).
- Μπορούμε να μην ορίζουμε την διάσταση ενός πίνακα κατά την δήλωσή του τύπου : **type** ONOMA **is array** (τύπος **range** <>) **of** στοιχεία;

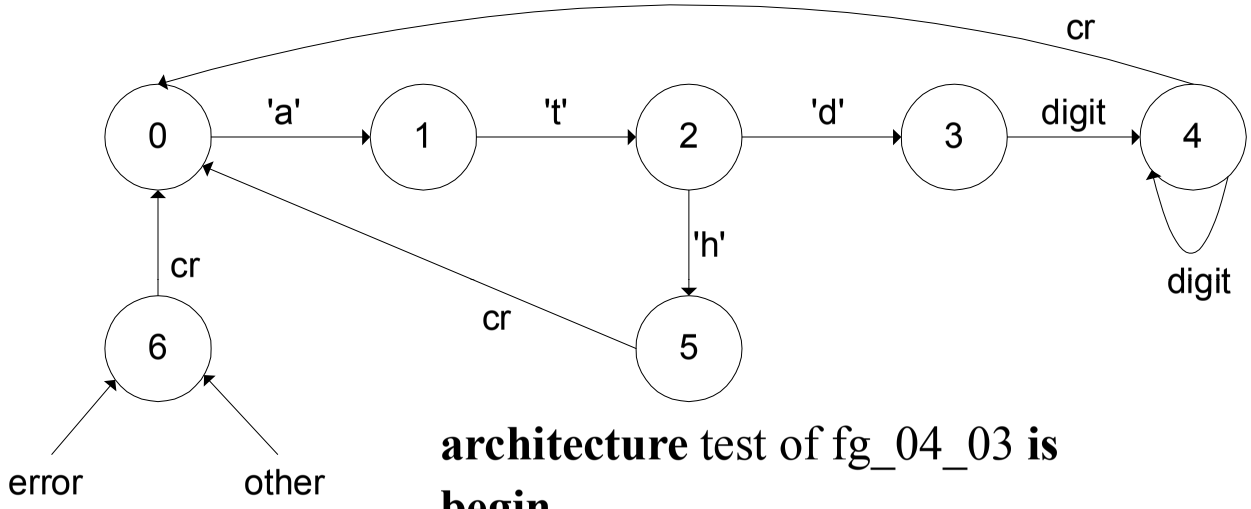
  type Matrix **is array** (integer **range** <>, integer **range**<>) **of** integer;

  variable Matrix8: Matrix(2 **downto** –8, 1 to 10);

Η δήλωση του μεγέθους κατά την δήλωση του αντικειμένου είναι <u>υποχρεωτική</u>

Πίνακες

Παράδειγμα: Χρήση δισδιάστατου πίνακα για την αναπαράσταση μήτρας μετάβασης FSM: η εντολή αποτελείται από το string ‘atd’ ακολουθούμενο από έναν αριθμό και cr ή το string ‘ath’ ακολουθούμενο από cr.




```
architecture test of fg_04_03 is
begin
Modem_controller : process is
   type symbol is ('a', 't', 'd', 'h', digit, cr, other);
   type symbol_string is array (1 to 20) of symbol;
   type state is range 0 to 6;
   type transition_matrix is array (state, symbol) of state;
```

Πίνακες

```
constant next_state : transition_matrix :=
  ( 0 =>('a' => 1,others => 6), 1 =>('t' => 2,others =>6),2 =>('d' => 3, 'h' =>5,others => 6),
    3 => (digit => 4, others => 6), 4 => (digit => 4, cr => 0, others => 6),
    5 => (cr => 0, others => 6), 6 => (cr => 0, others => 6) );
 variable command : symbol_string;
 variable current_state : state := 0;
begin
  for index in 1 to 20 loop
     current_state := next_state( current_state, command(index) );
     case current_state is
       -- . . .
       when 0 => exit;
       when others => null;
       -- …
     end case;
   end loop;
end process modem_controller;
```

Ιδιότητες Πινάκων

Attribute	Returns
MATRIX'**left**(N)	left-most element index
MATRIX'**right**(N)	right-most index
MATRIX'**high**(N)	upper bound
MATRIX'**low**(N)	lower bound
MATRIX'**length**(N)	the number of elements
MATRIX'**range**(N)	range
MATRIX'**reverse_range**(N)	reverse range
MATRIX'**ascending**(N)	a Boolean value TRUE if index is an ascending range, otherwise FALSE

Type A **is array** (1 **to** 4, 31 **downto** 0) **of** boolean;

A'left(1)=1 A'low(1)=1 A'length(1)=4

A'right(2)=0 A'high(2)=31 A'length(2)=32

A'range(1) is 1 to 4 A'reverse_range(2) is 0 to 31 A'ascending(1)=true

Ιδιότητες Πινάκων

- Προσπέλαση Πίνακα χωρίς γνώση της διάστασης του:

```
count:=0;
for index in free_map'range loop
    if free_map(index)='1' then
        count:=count+1;
    end if;
end loop;
```

Παρέχει προγραμματιστική ευελιξία και δυνατότητα παραμετροποίησης.

- Η ρητή δήλωση του μεγέθους μπορεί να παραληφθεί με αρχικοποίηση

```
type sample is array (natural range <>) of integer;
constant beep is sample := (127, 63, 0, -63, -127);      -- assumes range 0 to 4
```

Ειδικοί Πίνακες

String: Είναι ένας προκαθορισμένος τύπος πίνακα:

type string **is array** (positive **range** <>) **of** character;

constant message : **string** := “Ready”;

Bit Vector: Είναι ένας προκαθορισμένος τύπος πίνακα:

type bit_vector **is array** (natural **range** <>) **of** bit

Standard Logic (package std_logic_1164):

type std_ulogic_vector **is array** (natural **range** <>) **of** std_ulogic

constant hex_vector : std_ulogic_vector(15 downto 0) := X “05FA”

Μη-περιορισμένες θύρες

Ένας πίνακας σε μία θύρα μπορεί να μην έχει διάσταση μέχρι να γίνει δέσμευση της οντότητας του:

```
entity and_multiple is  port ( a : in bit_vector;  y : out bit ); end entity and_multiple;

architecture behavioral of and_multiple is
begin
  and_reducer : process ( a ) is
    variable result : bit;
  begin
    result := '1';
    for i in a'range loop
      result := result and a(i);
    end loop;
    y <= result;
  end process and_reducer;
end architecture behavioral;
```

```
signal input: bit_vector(7 downto 0)
signal output: bit;
tc_gate: and_multiple(input, output)
```

Τελεστές σε Πίνακες

Τελεστές με εφαρμογή σε ολόκληρους πίνακες:

- Οι τελεστές and, or, nand, nor, xor, xnor μπορούν να εφαρμοστούν σε δύο μονοδιάστατους πίνακες ίδιου τύπου (bit – boolean) και διάστασης.
- Ο τελεστής not μπορεί να εφαρμοστεί σε έναν μονοδιάστατο πίνακα.
- Οι τελεστές sll, srl, sla, sra, rol, ror εφαρμόζονται σε μονοδιάστατους πίνακες (bit – boolean) με δεξιό τελούμενο έναν ακέραιο.
- Οι τελεστές σύγκρισης μπορούν να εφαρμοστούν σε μονοδιάστατους πίνακες οποιουδήποτε αλλά κοινού τύπου.
- Ο τελεστής συγχώνευσης μπορεί να εφαρμοστεί σε δύο μονοδιάστατους πίνακες ίδιου τύπου “abc” & ’d’ = “abcd”

Τελεστές σε Πίνακες

Φέτες Πινάκων: Μπορούν να χρησιμοποιηθούν επιλεγμένες φέτες πινάκων

```
entity byte_swap is
  port (input : in bit_vector(0 to 15);  output : out bit_vector(0 to 15));
end entity byte_swap;

architecture behavior of byte_swap is
begin
  swap : process (input)
  begin
    output(8 to 15) <= input(0 to 7);
    output(0 to 7) <= input(8 to 15);
  end process swap;
end architecture behavior;
```

Εγγραφές

Τύπος εγγραφής: είναι μία σύνθετη δομή από στοιχεία διαφορετικών τύπων

```
type ONOMA is
record
        identifier: subtype_indication;
        …
        identifier: subtype_indication;
end record;
```

Παράδειγμα

```
type RecordExample is
record
  WIDTH: integer;
  BUS: bit_vector (3 downto 0);
  ACK: bit;
end record
signal A, B: RecordExample
```

- Η προσπέλαση ενός στοιχείου της εγγραφής γίνεται με χρήση της τελείας '.' Πχ. A.WIDTH<=5;
- Η αρχικοποίηση μπορεί να γίνει όπως στους πίνακες:

A:=(5, "0010", '1')

A:=(WIDTH=>5, BUS=>"0010", ACK=>'1')