ΗΜΥ-210: Σχεδιασμός Ψηφιακών Συστημάτων

Χειμερινό Εξάμηνο 2009

VHDL για Σχεδιασμό Συνδυαστικών Κυκλωμάτων

Διδάσκουσα: Μαρία Κ. Μιχαήλ

Πανεπιστήμιο Κύπρου
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

VHDL (*V*ery high speed integrated circuits *H*ardware *D*escription *L*anguage)

- Γλώσσα προγραμματισμού για την περιγραφή και προσομοίωση υλικού (λογικών σχεδιασμών).
- Επιχορηγήθηκε από IEEE και DoD (Department of Defense των ΗΠΑ) στις αρχές του '80.
- Βασικά χαρακτηριστικά:
  - Ιεραρχικός σχεδιασμός
  - Περιγραφή ***Διασυνδέσεων*** και ***Συμπεριφοράς*** με ακρίβεια (και ξεχωριστά)
  - Περιγραφή συμπεριφοράς: ***αλγοριθμικά*** ή με ***δομικό (structural)*** τρόπο
  - Μοντελοποίηση ***Χρονισμού*** (***Timing***) και ***Ταυτοχρονισμού*** (***Concurrency***)
    → Σχεδιασμοί μπορούν να προσομοιωθούν με ακρίβεια

Μοντελοποίηση

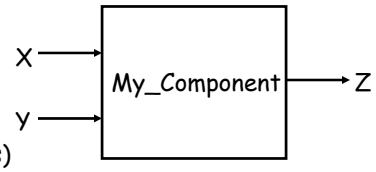
- Μια πλήρης περιγραφή ενός στοιχείου (component) με VHDL απαιτεί:
  - ***Entity (Οντότητα)***: καθορίζει τις διασυνδέσεις (interface) ενός στοιχείου (όνομα, εισόδους, εξόδους).
  - ***Architecture (Αρχιτεκτονική)***: καθορίζει την λειτουργία/συμπεριφορά (function) ενός στοιχείου.
- Σε κάθε στοιχείο αντιστοιχεί ένα μόνο ***entity*** και τουλάχιστον ένα ***architecture*** (πολλαπλά architecture είναι δυνατά).

5/11/2009 VHDL για Σχεδιασμό Συνδυαστικών Κυκλωμάτων MKM - 3

Απλό παράδειγμα: *Entity*

```
entity My_Component is -- "My_Component": όνομα
    port (X,Y: in BIT;     -- προδιαγραφές διασυνδέσεων
          Z:   out BIT);
end My_Component;
```

Εντολή **port** καθορίζει εισόδους και εξόδους



- Σχόλια (comments)
- Λέξεις κλειδιά VHDL (keywords)
- Αναγνωριστικό (identifier)
- Λειτουργία θύρας (port mode)
- Τύπος δεδομένων (data type)

5/11/2009 VHDL για Σχεδιασμό Συνδυαστικών Κυκλωμάτων MKM - 4

Απλό παράδειγμα: *Αρχιτεκτονική*

```
entity My_Component is -- "My_Component": όνομα
   Port (X,Y: in BIT;   -- προδιαγραφές διασυνδέσεων
         Z:   out BIT);
end My_Component;
```

```
Architecture My_Component_Arch of My_Component is
begin
   Z <= '1' when X='1' and Y='0' else '0';
end My_Component_Arch;
```

- Σχόλια (comments)
- Λέξεις κλειδιά VHDL (keywords)
- Αναγνωριστικό (identifier)
- Λειτουργία θύρας (port mode)
- Τύπος δεδομένων (data type)

Αντίστοιχη οντότητα (entity)

Z = X·Y'

5/11/2009 VHDL για Σχεδιασμό Συνδυαστικών Κυκλωμάτων MKM - 5

Άλλο Παράδειγμα: πύλη AND

```
entity My_AND is       -- "My_AND": όνομα
   Port (X,Y: in BIT;  -- προδιαγραφές διασυνδέσεων
         Z:   out BIT);
end My_END;
```

```
Architecture My_AND_Arch of My_AND is
begin
   Z <= '1' when X='1' and Y='1' else '0';
end My_AND_Arch;
```

- Σχόλια (comments)
- Λέξεις κλειδιά VHDL (keywords)
- Αναγνωριστικό (identifier)
- Λειτουργία θύρας (port mode)
- Τύπος δεδομένων (data type)

Z = X·Y

5/11/2009 VHDL για Σχεδιασμό Συνδυαστικών Κυκλωμάτων MKM - 6

Στοιχεία γλώσσας VHDL

- Σχόλια (Comments)
  - ξεκινούν με --, ισχύουν μέχρι το τέλος της γραμμής
- Δεσμευμένες Λέξεις (Keywords)
  - π.χ. **entity, port, is, in, out, end, architecture, begin, end, when, else, …**
- Αναγνωριστικά (Identifiers)
  - Μεταβλητές, ονόματα στοιχείων, κτλ

5/11/2009 VHDL για Σχεδιασμό Συνδυαστικών Κυκλωμάτων MKM - 7

Αναγνωριστικά (Identifiers)

- Μπορούν να περιέχουν A-Z, a-z, 0-9, _
- Πρέπει να ξεκινούν με γράμμα
- Δεν μπορούν να τελειώσουν με _
- Δεν μπορούν να περιέχουν 2 συνεχόμενες _
- Η VHDL είναι case-insensitive
  - Sel, sel και SEL αναφέρονται στο ίδιο αντικείμενο

5/11/2009 VHDL για Σχεδιασμό Συνδυαστικών Κυκλωμάτων MKM - 8

Παραδείγματα Αναγνωριστικών

- A2G
  - έγκυρο
- 8bit_counter
  - άκυρο – ξεκινά με αριθμό
- _NewValue
  - άκυρο – ξεκινά με _
- first#
  - άκυρο – περιέχει μη-αποδεκτό χαρακτήρα

5/11/2009 VHDL για Σχεδιασμό Συνδυαστικών Κυκλωμάτων MKM - 9

VHDL Αντικείμενα Δεδομένων (Data Objects)

- Σταθερές (Constants)
- Μεταβλητές (Variables)
- Σήματα (Signals)
- Αρχεία (Files*)

* Δεν υποστηρίζονται από εργαλεία σύνθεσης

5/11/2009 VHDL για Σχεδιασμό Συνδυαστικών Κυκλωμάτων MKM - 10

Χαρακτήρες και Συμβολοσειρές

- Χαρακτήρες (Characters)
  - 'A', '0', '1', '$', 'x', '*'
- Συμβολοσειρές (Strings)
  - "string of characters"
  - "00101101"
  - "0X110ZZ1"
- Δυαδικές (Bit) Συμβολοσειρές
  - B"011111010110"
  - O"3726"
  - X"7D6"

5/11/2009 VHDL για Σχεδιασμό Συνδυαστικών Κυκλωμάτων MKM - 11

VHDL Τύποι Δεδομένων (Data Types)

- Scalar
  - Integers
  - Enumerated
  - Reals (floating point)*
- Composite (σύνθετοι)
  - Arrays (πίνακες/διατάξεις)
  - Records
- Access (pointers -- δείκτες)*

* Δεν υποστηρίζονται από εργαλεία σύνθεσης

5/11/2009 VHDL για Σχεδιασμό Συνδυαστικών Κυκλωμάτων MKM - 12

Τύποι Δεδομένων Scalar: Integer

- Μικρότερο εύρος για κάθε υλοποίηση, όπως καθορίζεται από σχετικό πρότυπο:
  - 2,147,483,647 … + 2,147,483,647
- Παράδειγμα: αναθέσεις σε μεταβλητή τύπου integer :

```
ARCHITECTURE test_int OF test IS
BEGIN
   PROCESS (X)
     VARIABLE a: INTEGER;
   BEGIN
     a := 1;  -- OK
     a := -1;  -- OK
     a := 1.0;  -- άκυρο
   END PROCESS;
END test_int;
```

5/11/2009 VHDL για Σχεδιασμό Συνδυαστικών Κυκλωμάτων MKM - 13

Τύποι Δεδομένων Scalar: Integer (συν.)

- Μπορούμε επίσης να ορίσουμε integers με μικρότερο εύρος (sub-ranges)
  - Παραδείγματα:
    type CountValue **is range** 0 **to** 15;
    type Twenties **is range** 20 **to** 29;
    type Thirties **is range** 39 **downto** 30;

5/11/2009 VHDL για Σχεδιασμό Συνδυαστικών Κυκλωμάτων MKM - 14

Τύποι Δεδομένων Scalar: Enumerated

- Ο χρήστης ορίζει τη λίστα πιθανών τιμών
- Παράδειγμα:

```
TYPE binary IS ( ON, OFF );
... κάποιες εντολές ...
ARCHITECTURE test_enum OF test IS
BEGIN
    PROCESS (X)
      VARIABLE a: binary;
    BEGIN
      a := ON;  -- OK
      ... επιπρόσθετες εντολές ...
      a := OFF;  -- OK
      ... επιπρόσθετες εντολές ...
    END PROCESS;
END test_enum;
```


Τύποι Δεδομένων Scalar: Enumerated → Boolean

```
type boolean is (false, true);
... κάποιες εντολές ...
variable A,B,C: boolean;
... κάποιες εντολές ...
C := not A
C := A and B
C := A or B
C := A nand B
C := A nor B
C := A xor B
C := A xnor B
```

αντικείμενο (object) VHDL

Τελεστής ανάθεσης για μεταβλητές

Τύποι Δεδομένων Scalar: Enumerated → Bit

```
type bit is ('0', '1');
... κάποιες εντολές ...
signal x,y,z: bit;
... κάποιες εντολές ...
x <= '0';
y <= '1';
z <= x and y;
```

αντικείμενο (object) VHDL

Τελεστής ανάθεσης για σήματα

Τύποι Δεδομένων Scalar: Enumerated → Standard Logic

```
type std_logic is (   'U',   -- Uninitialized
                              (μη-αρχικοποιημένο)
                      'X'    -- Unknown
                              (άγνωστο)
                      '0'    -- Zero (μηδέν)
                      '1' ); -- One (ένα)
```

- std_logic είναι μέρος του *πακέτου* **ieee**
- ***Πακέτα (Packages):*** ήδη-μεταγλωττισμένος κώδικας VHDL που αποθηκεύεται σε βασικό κατάλογο ***(library)***

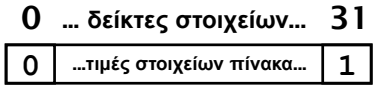
```
library IEEE;
use IEEE.std_logic_1164.all;
```

⇨ Πρέπει να περιλαμβάνεται στο κώδικά σας, πριν τη δήλωση τύπων δεδομένων ***std_logic***

Σύνθετοι Τύποι Δεδομένων (Composite Data Types)

- Array (Πίνακες/Διατάξεις):
  - Χρησιμοποιείται για ομαδοποίηση δεδομένων του ιδίου τύπου σε ένα ενιαίο αντικείμενο VHDL
  - Το εύρος μπορεί να είναι ακαθόριστο (=απεριόριστο) στη δήλωση (declaration) → καθορίζεται μόλις ο πίνακας χρησιμοποιηθεί
  - Παράδειγμα: δήλωση πίνακα μίας-διάστασης (one-dimensional array (vector))

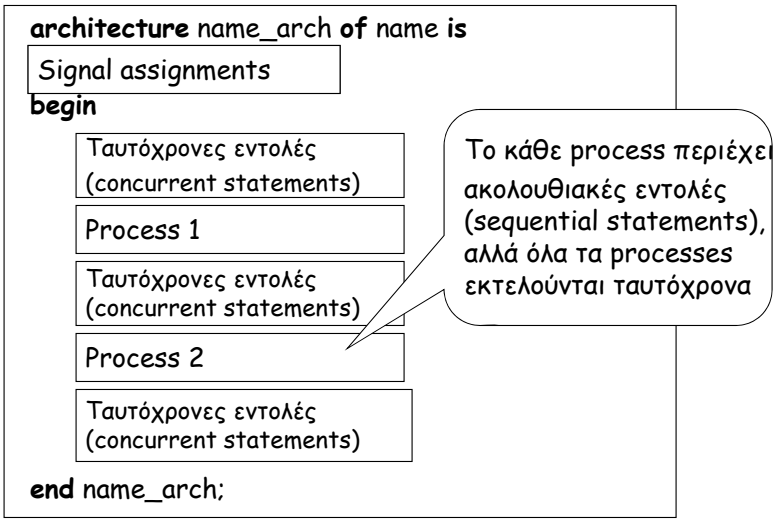
```
TYPE data_bus IS ARRAY(0 TO 31) OF BIT;
```




```
VARIABLE X : data_bus;
VARIABLE Y : BIT;

Y := X(12);   -- το Y παίρνει την τιμή του στοιχείου με δείκτη 12 του X
```


Αρχιτεκτονική VHDL



VHDL Process

```
P1: process (<sensitivity list>)
<variable declarations>
begin
    <sequential statements>
end process P1;
```

Προαιρετική σήμανση

Μέσα σε ένα process:
- Ανάθεση μεταβλητών (variables) με := και **άμεση** ενημέρωση.
- Ανάθεση σημάτων (signals) με <= και η ενημέρωση γίνετε στο **τέλος του process**.

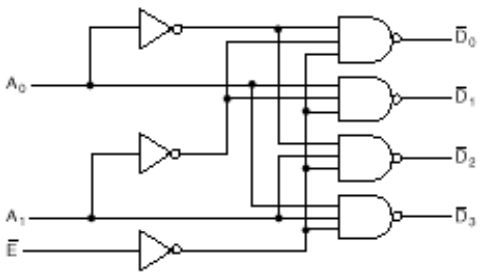
5/11/2009 VHDL για Σχεδιασμό Συνδυαστικών Κυκλωμάτων MKM - 21

Αρχιτεκτονική VHDL (συν.)

- Στο κάθε ***entity*** (μοναδικό ανά στοιχείο) αντιστοιχεί τουλάχιστον μια περιγραφή ***αρχιτεκτονικής (architecture)***
- Υπάρχουν 3 τρόποι περιγραφής της αρχιτεκτονικής:
  - ***Structural*** (Δομική): λεπτομερής περιγραφή σε επίπεδο πυλών/βασικών στοιχείων
  - ***Data Flow*** (Ροή Δεδομένων): περιγραφή βάση του τρόπου μεταφοράς δεδομένων μεταξύ των στοιχείων/σημάτων
  - ***Behavioral*** (Συμπεριφορά): αλγοριθμική περιγραφή – υψηλό επίπεδο χωρίς λεπτομέρειες
- Θα δούμε διάφορα παραδείγματα …

5/11/2009 VHDL για Σχεδιασμό Συνδυαστικών Κυκλωμάτων MKM - 22

2-to-4 DEC σε VHDL: Διάγραμμα σε επίπεδο πυλών



(a) Logic diagram

$\overline{E}$	A_1	A_0	$\overline{D}_0$	$\overline{D}_1$	$\overline{D}_2$	$\overline{D}_3$
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	X	X	1	1	1	1

(b) Truth table

$$\overline{D}_0 = \overline{E\ A_1\ A_0}$$
$$\overline{D}_1 = \overline{E\ A_1\ A_0}$$
$$\overline{D}_2 = \overline{E\ A_1\ A_0}$$
$$\overline{D}_3 = \overline{E\ A_1\ A_0}$$

(c) Logic Equations

2-to-4 DEC σε VHDL: Δήλωση Οντότητας (Entity Declaration)

```
-- 2-to-4 Line Decoder: Structural VHDL Description

library ieee, lcdf_vhdl;

use ieee.std_logic_1164.all, lcdf_vhdl.func_prims.all;

entity decoder_2_to_4 is
   port(E_n, A0, A1: in std_logic;
        D0_n, D1_n, D2_n, D3_n: out std_logic);

end decoder_2_to_4;
```

Εισαγόμενος κώδικας από βιβλιοθήκες

Είσοδοι & Έξοδοι

2-to-4 DEC σε VHDL: Αρχιτεκτονική Δομική Περιγραφή (Structural)

```
architecture structural_1 of decoder_2_to_4 is
   component NOT1
        port(in1: in std_logic;
             out1: out std_logic);
   end component;
   component NAND3
        port(in1, in2, in3: in std_logic;
             out1: out std_logic);
   end component;
```

Δήλωση απαραίτητων component (διαθέσιμα από τις βιβλιοθήκες)

5/11/2009 VHDL για Σχεδιασμό Συνδυαστικών Κυκλωμάτων MKM - 25

2-to-4 DEC σε VHDL: Αρχιτεκτονική Δομική Περιγραφή (Structural) (συν.)

```
signal E, A0_n, A1_n: std_logic;
begin
   g0: NOT1 port map (in1 => A0, out1 => A0_n);
   g1: NOT1 port map (in1 => A1, out1 => A1_n);
   g2: NOT1 port map (in1 => E_n, out1 => E);
   g3: NAND3 port map (in1 => A0_n, in2 => A1_n,
                       in3 => E, out1 => D0_n);
   g4: NAND3 port map (in1 => A0, in2 => A1_n,
                       in3 => E, out1 => D1_n);
   g5: NAND3 port map (in1 => A0_n, in2 => A1,
                       in3 => E, out1 => D2_n);
   g6: NAND3 port map (in1 => A0, in2 => A1,
                       in3 => E, out1 => D3_n);
end structural_1;
```

Τοπικά σήματα

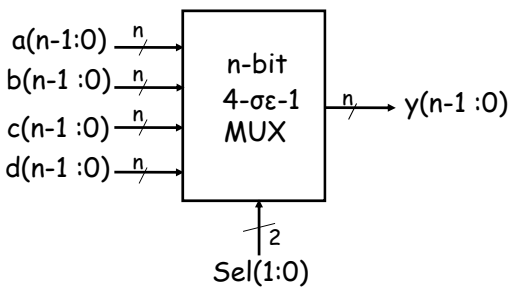
5/11/2009 VHDL για Σχεδιασμό Συνδυαστικών Κυκλωμάτων MKM - 26

2-to-4 DEC σε VHDL: Αρχιτεκτονική Περιγραφή Ροής Δεδομένων (Dataflow)

```
architecture dataflow_1 of decoder_2_to_4 is
signal E, A0_n, A1_n: std_logic;
begin
   A0_n <= not A0;
   A1_n <= not A1;
   E <= not E_n;
   D0_n <= not (A0_n and A1_n and E);
   D1_n <= not (A0 and A1_n and E);
   D2_n <= not (A0_n and A1 and E);
   D3_n <= not (A0 and A1 and E);
end dataflow_1;
```

5/11/2009 VHDL για Σχεδιασμό Συνδυαστικών Κυκλωμάτων MKM - 27

Άλλο Παράδειγμα: n-bit 4-σε-1 MUX



Sel	y
"00"	a
"01"	b
"10"	c
"11"	d

5/11/2009 VHDL για Σχεδιασμό Συνδυαστικών Κυκλωμάτων MKM - 28

n-bit 4-σε-1 MUX: Δήλωση Οντότητας (Entity declaration)

```
library IEEE;
use IEEE.std_logic_1164.all;

entity mux4g is
    generic(width:positive);
    port (
        a: in STD_LOGIC_VECTOR (width-1 downto 0);
        b: in STD_LOGIC_VECTOR (width-1 downto 0);
        c: in STD_LOGIC_VECTOR (width-1 downto 0);
        d: in STD_LOGIC_VECTOR (width-1 downto 0);
        sel: in STD_LOGIC_VECTOR (1 downto 0);
        y: out STD_LOGIC_VECTOR (width-1 downto 0)
    );
end mux4g;
```


n-bit 4-σε-1 MUX: Αρχιτεκτονική: Περιγραφή Ροής Δεδομένων με χρήση εντολής CASE

```
architecture mux4g_arch of mux4g is
begin
  process (sel, a, b, c, d)
  begin
    case sel is
      when "00"   => y <= a;
      when "01"   => y <= b;
      when "10"   => y <= c;
      when others => y <= d;
    end case;
  end process;
end mux4g_arch;
```

Sel	y
"00"	a
"01"	b
"10"	c
"11"	d

Η εντολή CASE πρέπει να περιέχει ΟΛΕΣ τις πιθανότητες τιμών.

Παράδειγμα απλής συνδυαστικής συνάρτησης: Περιγραφή Ροής Δεδομένων

```
library ieee;
use ieee.std_logic_1164.all;

entity func2 is
 port (x1,x2,x3:  in std_logic;
       f:         out std_logic );
end func2;

architecture dataflow of func2 is
begin
 f <= (not x1 and not x2 and x3) or (x1 and not x2 and not x3) or (x1
    and not x2 and x3) or (x1 and x2 and not x3);
end logicfunc;
```


Ημιαθροιστής

- Πρόβλημα: Σχεδιάστε ένα ημιαθροιστή 1-bit με κρατούμενο (carry) και σήμα ενεργοποίησης (enable).
- Προδιαγραφές
  - Είσοδοι και έξοδοι είναι 1-bit
  - Όταν το enable είναι 1, το αποτέλεσμα είναι η πρόσθεση x+y με carry
  - Έξοδοι 0 όταν το enable είναι 0

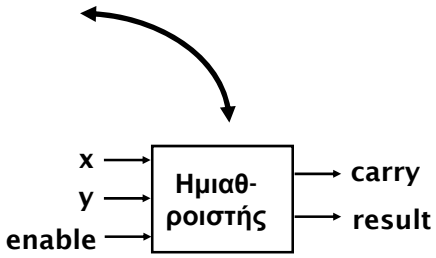
Ημιαθροιστής: Δήλωση Οντότητας

- Η οντότητα περιγράφει τις διασυνδέσεις του component -- δηλώνονται *θύρες (ports)* εισόδων και εξόδων

```
ENTITY half_adder IS

    PORT( x, y, enable: IN bit;
          carry, result: OUT bit);

END half_adder;
```



5/11/2009 VHDL για Σχεδιασμό Συνδυαστικών Κυκλωμάτων MKM - 33

Ημιαθροιστής: Αρχιτεκτονική με Περιγραφή Συμπεριφοράς (Behavioral)

- Μπορούμε να χρησιμοποιήσουμε μια περιγραφή υψηλού επιπέδου για την συνάρτηση που υλοποιεί το κύκλωμα

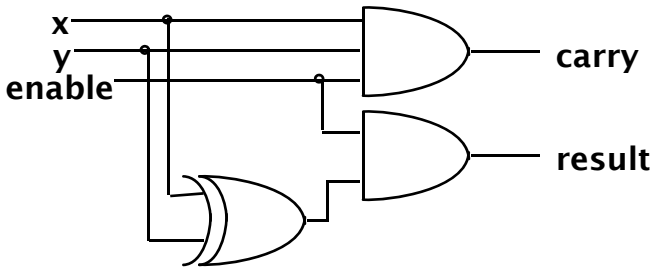
```
ARCHITECTURE half_adder_a of half_adder IS
    BEGIN
      PROCESS (x, y, enable)
        BEGIN
                IF enable = '1' THEN
                        result <= x XOR y;
                        carry <= x AND y;
                ELSE
                END IF;
        END PROCESS;
    END half_adder_a;
```

- **Αυτό το μοντέλο μπορεί να προσομοιωθεί έτσι ώστε να επαληθευτεί η σωστή λειτουργία του κυκλώματος**

5/11/2009 VHDL για Σχεδιασμό Συνδυαστικών Κυκλωμάτων MKM - 34

Ημιαθροιστής: Δομική Περιγραφή Αρχιτεκτονικής (Structural)

- Εναλλακτικά, μπορούμε να χρησιμοποιήσουμε μια δομική περιγραφή (βάση διαθέσιμων components που έχουν ήδη δηλωθεί)



- **Αυτό το μοντέλο μπορεί επίσης να προσομοιωθεί έτσι ώστε να επαληθευτεί η σωστή λειτουργία του κυκλώματος**

5/11/2009 VHDL για Σχεδιασμό Συνδυαστικών Κυκλωμάτων MKM - 35

Ημιαθροιστής: Δομική Περιγραφή Αρχιτεκτονικής (συν.)

```
ARCHITECTURE half_adder_c of half_adder_Nty IS

    COMPONENT and2
      PORT (in0, in1 : IN BIT;
            out0 : OUT BIT);
    END COMPONENT;

    COMPONENT and3
      PORT (in0, in1, in2 : IN BIT;
            out0 : OUT BIT);
    END COMPONENT;

    COMPONENT xor2
      PORT (in0, in1 : IN BIT;
            out0 : OUT BIT);
    END COMPONENT;

    FOR ALL : and2 USE ENTITY gate_lib.and2_Nty(and2_a);
    FOR ALL : and3 USE ENTITY gate_lib.and3_Nty(and3_a);
    FOR ALL : xor2 USE ENTITY gate_lib.xor2_Nty(xor2_a);

-- η περιγραφή συνεχίζεται στην επόμενη διαφάνεια
```

5/11/2(MKM - 36

Ημιαθροιστής: Δομική Περιγραφή Αρχιτεκτονικής (συν.)

```
-- συνεχιζόμενη περιγραφή half_adder_c

    SIGNAL xor_res : bit; -- εσωτερικό σήμα
    -- τα υπόλοιπα σήματα έχουν ήδη δηλωθεί στο entity

    BEGIN

      A0 : and2 PORT MAP (enable, xor_res, result);
      A1 : and3 PORT MAP (x, y, enable, carry);
      X0 : xor2 PORT MAP (x, y, xor_res);

    END half_adder_c;
```


Ημιαθροιστής: Αρχιτεκτονική με Περιγραφή Ροής Δεδομένων (Dataflow)

- Μια τρίτη μέθοδος περιγραφής της αρχιτεκτονικής ενός component χρησιμοποιεί λογικές εξισώσεις για να αναπτύξει μια περιγραφή ροής δεδομένων

```
ARCHITECTURE half_adder_b of half_adder_Nty IS
    BEGIN
      carry <= enable AND (x AND y);
      result <= enable AND (x XOR y);
    END half_adder_b;
```

- **Ξανά, το μοντέλο αυτό μπορεί να προσομοιωθεί σε αυτό το επίπεδο για να επιβεβαιωθούν οι λογικές εξισώσεις.**

Παράδειγμα Αθροιστή 4ων-bit: Δήλωση Οντότητας

```
-- Αθροιστής 4ων-bit
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity adder_4_b is
    port(B, A : in std_logic_vector(3 downto 0);
         C0 : in std_logic;
         S : out std_logic_vector(3 downto 0);
         C4 : out std_logic);
end adder_4_b;
```

5/11/2009 VHDL για Σχεδιασμό Συνδυαστικών Κυκλωμάτων ΜΚΜ - 39

Αθροιστής 4ων-bit:Αρχιτεκτονική με Περιγραφή Συμπεριφοράς (Behavioral)

```
architecture behavioral of adder_4_b is

signal sum : std_logic_vector(4 downto 0);

begin
   sum <= ('0' & A) + ('0' & B) + ("0000" & C0);
   C4 <= sum(4);
   S <= sum(3 downto 0);
end behavioral;
```

Είναι πλήρης αθροιστής;

$0A_3A_2A_1A_0$ $0B_3B_2B_1B_0$ $0000C_0$

5/11/2009 VHDL για Σχεδιασμό Συνδυαστικών Κυκλωμάτων ΜΚΜ - 40

Αθροιστής 1-bit:Αρχιτεκτονική με Περιγραφή Ροής Δεδομένων (Dataflow)

```
library ieee;
use ieee.std_logic_1164.all;

entity fulladd is
 port (Cin, x, y: in std_logic;
        s, Cout: out std_logic);
end fulladd;

architecture logicfunc of fulladd is
begin
 s <= x xor y xor Cin;
 Cout <= (x and y) or (Cin and x) or (Cin and y);
end logicfunc;
```


Αθροιστής 4ων-bit: Δήλωση Οντότητας

```
library ieee;
use ieee.std_logic_1164.all;

entity adder4 is              -- s = x+y
 port (  Cin:              in std_logic;
         x3,x2,x1,x0:      in std_logic;
          y3,y2,y1,y0:     in std_logic;
          s3,s2,s1,s0:     out std_logic;
          Cout:            out std_logic );
end adder4;
```

Αθροιστής 4ων-bit: Δομική Περιγραφή Αρχιτεκτονικής (Structural)

```
architecture structural of adder4 is
 signal c1,c2,c3: std_logic;
 component fulladd
  port (Cin,x,y: in std_logic;
        s,Cout: out std_logic);
 end component;

begin
 stage0: fulladd port map (Cin,x0,y0,s0,c1);
 stage1: fulladd port map (c1,x1,y1,s1,c2);
 stage2: fulladd port map (c2,x2,y2,s2,c3);
 stage3: fulladd port map (Cin=>c3,Cout=cout,x=>x3,y=>y3,s=>s3);
end structural;
```

Ίδια σειρά όπως στη δήλωση του entity

Προσαρμοσμένη σειρά

5/11/2009 VHDL για Σχεδιασμό Συνδυαστικών Κυκλωμάτων MKM - 43

2-σε-1 MUX

```
library ieee;
use ieee.std_logic_1164.all;

entity mux2to1 is
 port (d0,d1,s:    in std_logic;
             y:    out std_logic);
end mux2to1;

architecture behavioral of mux2to1 is
begin
 with s select
  y <=  d0 when '0',
        d1 when others;
end behavioral;
```

5/11/2009 VHDL για Σχεδιασμό Συνδυαστικών Κυκλωμάτων MKM - 44

Αποκωδικοποιητής 2-σε-4

```
library ieee;
use ieee.std_logic_1164.all;

entity dec2to4 is
 port (w:      in std_logic_vector(1 downto 0);
       e:      in std_logic;
       y:      out std_logic_vector(0 to 3));
end dec2to4;
```


Αποκωδικοποιητής 2-σε-4 (συν.)

```
architecture behavioral of dec2to4 is
signal ew: std_logic_vector(2 downto 0);
begin
 ew <= e & w;  -- concatenation!
 with ew select
  y <= "1000" when "100",
       "0100" when "101",
       "0010" when "110",
       "0001" when "111",
       "0000" when others;
end behavioral;
```