Γλώσσα VHDL

ASIC και FPGA

ASIC: Application Specific Integrated Circuit
Ολοκληρωμένο Κύκλωμα Ειδικών Εφαρμογών σχεδιασμένο για μια συγκεκριμένη εφαρμογή.

FPGA: Field Programmable Gate Arrays
Ολοκληρωμένο επαναπρογραμματιζόμενο κύκλωμα.

VHDL Γλώσσα περιγραφής υλικού (1)

VHDL: **V**ery high speed integrated circuit **H**ardware **D**esription **L**anguage

- Η γλώσσα περιγραφής υλικού VHDL χρησιμοποιείται για να περιγράψει την εσωτερική δομή του συστήματος που θα υλοποιηθεί μέσα στο ολοκληρωμένο κύκλωμα του FPGA ή του ASIC.
- Η γλώσσα VHDL καλύπτει και την ακολουθιακή (δημιουργία ιεραρχικών κυκλωμάτων με τη βοήθεια του δομημένου προγραμματισμού) και την παράλληλη επεξεργασία ψηφιακών σημάτων (υλοποίηση παράλληλων διεργασιών).

VHDL Γλώσσα περιγραφής υλικού (2)

Το κύριο χαρακτηριστικό της VHDL είναι ότι είναι σχεδιασμένη να εκτελεί ταυτόχρονα τις λειτουργίες που περιγράφονται. Όλα τα αποτελέσματα των εξόδων παράγονται ταυτόχρονα τη στιγμή που εφαρμόζονται οι τιμές εισόδου.

Αυτό πρακτικά μεταφράζεται στο ότι δεν έχει σημασία η σειρά με την οποία γράφονται οι διάφορες λειτουργίες. Κάθε κομμάτι κώδικα παράγει ένα αποτέλεσμα παράλληλα με ένα άλλο κομμάτι κώδικα, ανεξάρτητα από το αν είναι γραμμένο πριν ή μετά από αυτό.

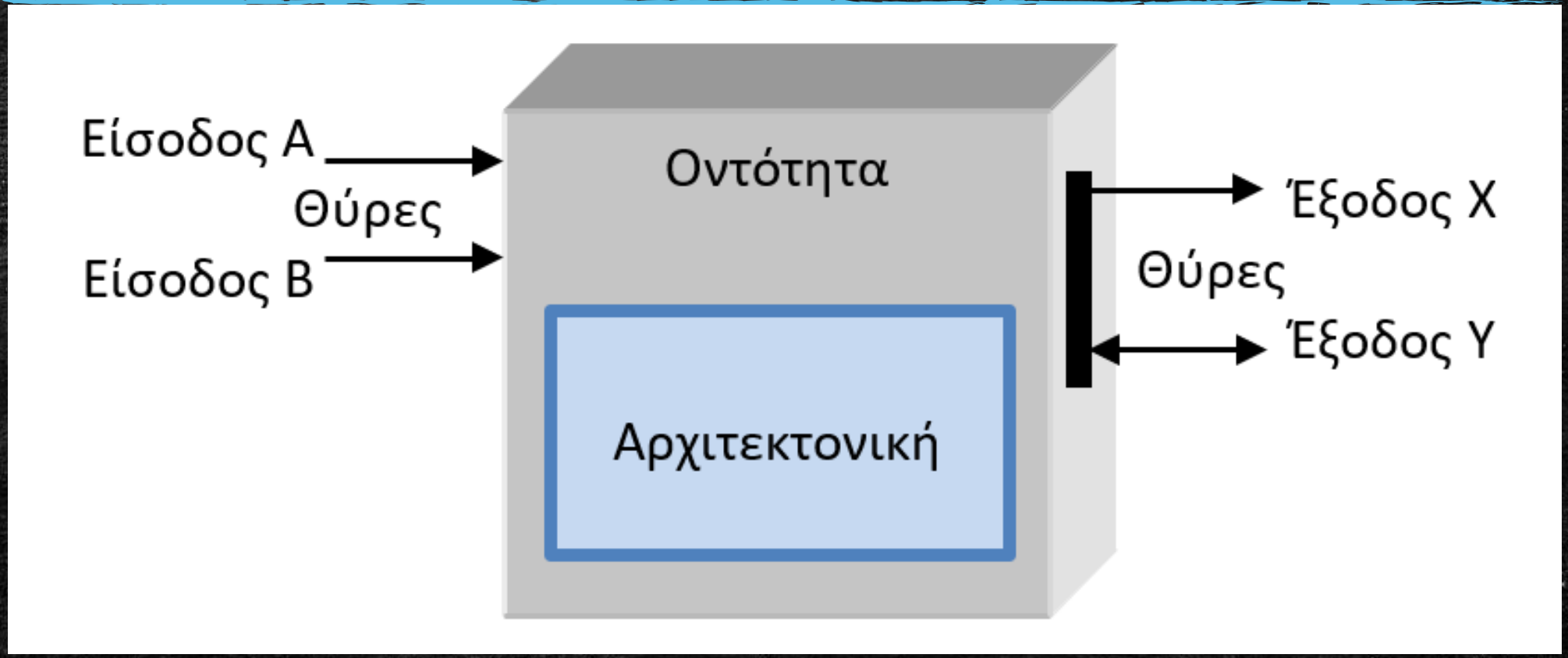
VHDL Γλώσσα περιγραφής υλικού (3)

Ωστόσο η VHDL μπορεί να υποστηρίξει εκτός από σύγχρονες και ακολουθιακές λογικές λειτουργίες. Δηλαδή λειτουργίες τα αποτελέσματα των οποίων παράγονται όταν θα λάβει χώρα ένα συμβάν (π.χ. ο επόμενος παλμός του ρολογιού).

Συνεπώς ο κώδικας ενός προγράμματος σε VHDL μπορεί να περιλαμβάνει σύγχρονες δομές (concurrent code) αλλά και ακολουθιακές δομές (sequential code).

Δομή προγράμματος σε VHDL

Οντότητα και αρχιτεκτονική (1)



Οντότητα και αρχιτεκτονική (2)

Ένα πρόγραμμα σε VHDL αποτελείται από δύο κύρια τμήματα κώδικα:

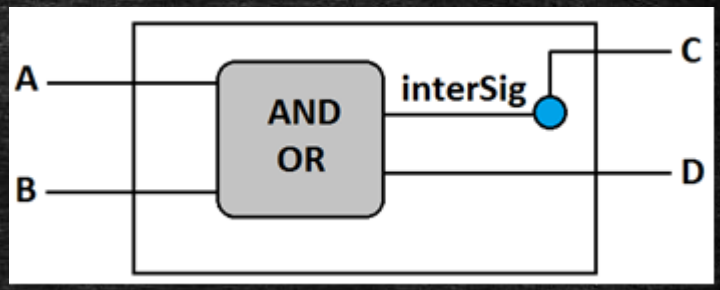
Οντότητα: *όπου περιγράφονται οι είσοδοι και οι έξοδοι της λογικής μονάδας*

Αρχιτεκτονική: *όπου περιγράφεται η δομή και η λειτουργία της λογικής μονάδας*

Όταν μια οντότητα εμπεριέχεται μέσα σε μια άλλη, τότε η εσωτερική οντότητα ονομάζεται ***στοιχείο (component)***

Παράδειγμα κώδικα σε VHDL

```
library ieee;
use ieee.std_logic_1164.all;
ENTITY My_system IS
PORT (A, B : in std_logic;
        C, D : out std_logic);
END My_system;
ARCHITECTURE Behavioral OF my_system IS
        SIGNAL interSig : std_logic;
BEGIN
        PROCESS(A, B)
        BEGIN
                interSig <= A and B;
                D <= A or B;
        END PROCESS;
        C <= interSig;
END Behavioral;
```



Σήματα

Τα σήματα είναι οι φυσικές συνδέσεις ενός ηλεκτρονικού κυκλώματος. Αποτελούν τα καλώδια καθώς και τους διαύλους επικοινωνίας μεταξύ των μονάδων.

Η δήλωση ενός σήματος γίνεται με την παρακάτω σύνταξη:

signal *όνομα σήματος: είδος σήματος τύπος σήματος*

Είδη σημάτων

Τα είδη των σημάτων που υποστηρίζονται από την VHDL είναι:

- **in:** σήμα εισόδου
- **out:** σήμα εξόδου
- **inout:** σήμα εισόδου/εξόδου
- **buffer: σ**ήμα εξόδου το οποίο μπορεί να αποτελεί και σήμα αναφοράς ενός νέου εσωτερικού σήματος μέσα στην ίδια οντότητα

Τύποι σημάτων (1)

Τύπος σήματος	Περιγραφή
BIT :	Το σήμα λαμβάνει την τιμή 0 και 1.
BIT_VECTOR :	Το σήμα λαμβάνει ακολουθία τιμών 0 και 1 π.χ. signal xxx : bit_vector (0 to 4);
STD_LOGIC :	Το σήμα λαμβάνει την τιμή: 0 = μηδέν 1 = ένα Z = κατάσταση με υψηλή σύνθετη αντίσταση - = αδιάφορη τιμή L = ασθενής τιμή 0 H = ασθενής τιμή 1

Τύποι σημάτων (2)

STD_LOGIC : (συνέχεια)	X = άγνωστη τιμή W = άγνωστη ασθενή τιμή U = μη μηδενική τιμή
STD_LOGIC_VECTOR:	Το σήμα λαμβάνει μια ακολουθία τιμών πχ. signal xxx : out std_logic_vector (7 downto 0);
SIGNED/UNSIGNED:	Καθορίζει εάν χρησιμοποιούμε προσημασμένους ή μη-προσημασμένους αριθμούς.

Τύποι σημάτων (3)

INTEGER:	Καθορίζει έναν αριθμό με μήκος 32 bits, δίνοντας τη δυνατότητα τροποποίησης του εύρους με χρήση της εντολής RANGE π.χ. signal x: integer range -60 to 60
BOOLEAN:	Καθορίζεται για το σήμα η τιμή TRUE ή FALSE.
ENUMERATION:	Καθορίζει ένα σύνολο τιμών. Χρησιμοποιείται σε δήλωση καταστάσεων μηχανής

Παραδείγματα σημάτων

```
signal A: in std_logic;
signal X: out std_logic;
signal Y: out std_logic_vector(7 downto 0);
```

Αρχικοποίηση σημάτων

SIGNAL όνομα σήματος: τύπος σήματος := {τιμή };

- **signal** A: **in** std_logic:='0';

  Ένα εισαγωγικό σημαίνει bit

- **signal A**: **in** std_logic_vector(7 downto 0):="0";

  Διπλό εισαγωγικό σημαίνει διάνυσμα

- **signal** A: **out** std_logic_vector(7 downto 0):="11001100";

Μεταβλητές – Σταθερές

CONSTANT όνομα σταθεράς: Τύπος:= τιμής σταθεράς;

VARIABLE όνομα μεταβλητής: τύπος μεταβλητής;

Οι μεταβλητές δεν μεταφέρουν πληροφορίες εξωτερικά του στοιχείου όπου έχει οριστεί. Χρησιμοποιούνται για την προσωρινή αποθήκευση τιμών στο κομμάτι του κώδικα που περιγράφει μια «Διεργασία» (Process), δηλαδή ένα τμήμα «ακολουθιακών» εντολών.

Τα ονόματα των σταθερών και των μεταβλητών:

- Μπορεί να περιέχουν κεφαλαίους και πεζούς χαρακτήρες χωρίς να υπάρχει διάκριση μεταξύ τους (το A θεωρείται ίδιο με το a)
- Πρέπει πάντα να αρχίζουν με συμβολοσειρά και όχι με αριθμό
- Δεν επιτρέπεται να αρχίζουν και να τελειώνουν με κάτω παύλα (_)
- Δεν πρέπει να ταυτίζονται με δεσμευμένες λέξεις

Δεσμευμένες λέξεις

	disconnect	is	out	sli
access	downto	label	package	sra
after	else	library	port	srl
alias	elsif	linkage	postponed	subtype
all	end	literal	procedure	then
and	entity	loop	process	to
architecture	exit	map	pure	transport
array	file	mod	range	type
assert	for	nand	record	unaffected
attribute	function	new	register	units
begin	generate	next	reject	until
block	generic	nor	return	use
body	group	not	rol	variable
buffer	guarded	null	ror	wait
bus	if	of	select	when
case	impure	on	severity	while
component	in	open	signal	with
configuration	inertial	or	shared	xnor
constant	inout	others	sla	xor

Τύποι δεδομένων

Τύπος	Περιγραφή
integer	Ακέραιος αριθμός
real	Πραγματικός αριθμός
string	Πίνακας χαρακτήρων
character	Χαρακτήρας 7-bit ASCII
time	Μονάδα χρόνου σε hr, min, sec, ms, us, ns, ps, fs

Τελεστές

+	Άθροισμα
-	Αφαίρεση
&	Concatenation
<=	Εκχώρηση σε σήμα
:=	Εκχώρηση σε μεταβλητή
and	Λογική πράξη and
nand	Λογική πράξη Nand
or	Λογική πράξη or
nor	Λογική πράξη nor
xor	Λογική πράξη xor
xnor	Λογική πράξη xnor
not	Λογική πράξη not
*	Πολλαπλασιασμός
/	Διαίρεση
mod	Πηλίκο (5 mod 3 = 2)
rem	Υπόλοιπο (7 rem 2 =1)

=	Ισότητα
/=	Διαφορά
<	Μικρότερο
<=	Μικρότερο ή ίσο
>	Μεγαλύτερο
>=	Μεγαλύτερο ή ίσο
rol	Περιστροφή αριστερά
ror	Περιστροφή δεξιά
sla	Αριθμητική ολίσθηση αριστερά
sra	Αριθμητική ολίσθηση δεξιά
sla	Λογική Ολίσθηση αριστερά
sla	Λογική Ολίσθηση δεξιά
**	Ύψωση σε δύναμη
abs	Απόλυτη τιμή

Αρχιτεκτονικές προγραμματισμού σε VHDL (1)

Στη γλώσσα VHDL η λογική συνάρτηση που περιγράφει τη δομή μιας οντότητας μπορεί να οριστεί με τρεις διαφορετικές δομές αρχιτεκτονικής:

- Περιγραφή ροής δεδομένων (dataflow description): η αρχιτεκτονική μιας οντότητας ορίζεται μέσω της περιγραφής της δρομολόγησης των δεδομένων μεταξύ των θυρών εισόδου – εξόδου
- Περιγραφή δομής (structural description): η αρχιτεκτονική μιας οντότητας βασίζεται στον καθορισμό των σημάτων διασύνδεσης των εξαρτημάτων με χρήση περιγραφής υλικού
- Περιγραφή συμπεριφοράς (behavioral description): η αρχιτεκτονική μιας οντότητας υλοποιείται με χρήση αλγορίθμων

Αρχιτεκτονικές προγραμματισμού σε VHDL (2)

Να γραφεί κώδικας στη γλώσσα VHDL για την υλοποίηση της λογικής συνάρτησης $F = XZ + YZ'$ με:

- Περιγραφή ροής δεδομένων
- Περιγραφή δομής
- Περιγραφή συμπεριφοράς

Περιγραφή Ροής Δεδομένων

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY example1 IS
    PORT (X,Y,Z: IN STD_LOGIC;
                F : OUT STD_LOGIC);
END example1;

ARCHITECTURE dataflow OF example1 IS
BEGIN
        F <=(X AND Z) OR ((NOT Z) AND Y);
END dataflow;
```

Περιγραφή Δομής (1)

```
LIBRARY IEEE;
LIBRARY work;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY NOT_Gate IS
    PORT (A : in bit;
          B : out bit);
END NOT_Gate;

ARCHITECTURE NOT_Arch OF NOT_Gate IS
BEGIN
        B <= not A;
 END NOT_Arch;
```

Περιγραφή Δομής (2)

```
LIBRARY IEEE;
LIBRARY work;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY AND_Gate IS
    PORT (A,B : in bit;
                C : out bit);
END AND_Gate;

ARCHITECTURE AND_Arch OF AND_Gate IS
BEGIN
          C <= A and B;
 END AND_Arch;
```

Περιγραφή Δομής (3)

```
LIBRARY IEEE;
LIBRARY work;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY OR_Gate IS
    PORT (A,B : in bit;
                C : out bit);
END OR_Gate;

ARCHITECTURE OR_Arch OF OR_Gate IS
BEGIN
          C <= A or B;
 END OR_Arch;
```

Περιγραφή Δομής (4)

```
LIBRARY IEEE;
LIBRARY work;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY example2 IS
   PORT (X,Y,Z : in STD_LOGIC;
                 F : out STD_LOGIC);
END example2;

ARCHITECTURE structural OF example2 IS
   COMPONENT NOT_Gate
      PORT (A: in STD_LOGIC;
            B: out  STD_LOGIC);
   END COMPONENT;
```

Περιγραφή Δομής (5)

```
    COMPONENT AND_Gate
      PORT (A,B: in STD_LOGIC;
                 C: out  STD_LOGIC);
    END COMPONENT;
    COMPONENT OR_Gate
      PORT (A,B: in STD_LOGIC;
                 C: out  STD_LOGIC);
    END COMPONENT;

    SIGNAL x1,x2,x3: STD_LOGIC;
  BEGIN
    AND1: AND_Gate PORT MAP(X,Z,x1);
    NOT1: NOT_Gate PORT MAP(Z,x2);
    AND2: AND_Gate PORT MAP(Y,x2,x3);
    OR1: OR_Gate PORT MAP(x1,x3,F);
  END structural;
```

Περιγραφή Συμπεριφοράς (1)

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY example3 IS
   PORT (X, Y, Z : IN STD_LOGIC;
                 F : OUT STD_LOGIC);
END example3;

ARCHITECTURE Behavior OF example3 IS
   BEGIN
   PROCESS(X,Y,Z)
      BEGIN
         IF(X='0' AND Y='1' AND Z='0') THEN
                  F<='1';
         ELSIF(X='1' AND Y='0' AND Z='1') THEN
                  F<='1';
```

Περιγραφή Συμπεριφοράς (2)

```
        ELSIF(X='1' AND Y='1' AND Z='0') THEN
                F<='1';
        ELSIF(X='1' AND Y='1' AND Z='1') THEN
                F<='1';
        ELSE
                F<='0';
        END IF;
    END PROCESS;
END Behavior;
```

Καλή συνέχεια…