Γλώσσες Περιγραφής Υλικού

Εισαγωγή στην VHDL

Νικόλαος Καββαδίας
nkavv@physics.auth.gr

17 Μαρτίου 2009

Αντικείμενο του μαθήματος CST256: Γλώσσες Περιγραφής Υλικού

- Επιμέρους στόχοι του μαθήματος
  - Σχεδιασμός ψηφιακών κυκλωμάτων με τη γλώσσα περιγραφής υλικού VHDL
  - Παρουσίαση χαρακτηριστικών συνθέσιμων κυκλωμάτων που συναντώνται στη μοντέρνα ψηφιακή σχεδίαση
  - Εργαστηριακή εξάσκηση στην περιγραφή και προσομοίωση ψηφιακών κυκλωμάτων
- Τρόπος εξέτασης του μαθήματος
  - Γραπτές εξετάσεις στο τέλος του εξαμήνου: 60% του τελικού βαθμού
  - Εργασία: 40% του τελικού βαθμού
- Ενημέρωση για ανακοινώσεις, διαλέξεις, ύλη, εργασίες από τον ιστότοπο του μαθήματος:
  `http://eclass.uop.gr/courses/CST256/index.php`

Περίγραμμα μαθήματος (1)

- Εισαγωγή στην VHDL
  - Ιστορικό της ανάπτυξης της γλώσσας
  - Τα επίπεδα μοντελοποίησης για την περιγραφή ενός ψηφιακού κυκλώματος σε VHDL
  - Τυποποιήσεις που διέπουν τη χρήση της VHDL
- Βασικά δομικά στοιχεία της VHDL
  - Οντότητες
  - Περιγραφές αρχιτεκτονικών
  - Βαθμωτοί και σύνθετοι τύποι δεδομένων
  - Διεργασίες
  - Συντρέχων και ακολουθιακός κώδικας
  - Συστατικά και δομικές περιγραφές
  - Διαδικασίες και συναρτήσεις
  - Πακέτα και βιβλιοθήκες
  - Ιδιότητες
  - Διαμορφώσεις

Περίγραμμα μαθήματος (2)

- Σχεδιασμός κυκλωμάτων με την VHDL
  - Βασικά συνδυαστικά κυκλώματα
  - Βασικά ακολουθιακά κυκλώματα
  - Γενικές σταθερές
  - Μοντελοποίηση παραμετρικών κυκλωμάτων
  - Γεννήτορες δομών
  - Μηχανές πεπερασμένων καταστάσεων
- VHDL για προχωρημένους
  - Μοντελοποίηση κυκλωμάτων για λογική σύνθεση
  - Λειτουργικός έλεγχος της ορθής λειτουργίας των κυκλωμάτων (τεχνικές συγγραφής testbench)
  - Κυκλώματα επεξεργασίας δεδομένων - χειριστές δεδομένων
  - Μοντελοποίηση απλών επεξεργαστών
  - Παρουσίαση υποδειγματικής εργασίας

Οργάνωση των παραδόσεων

Ενδεικτική κατανομή των διαλέξεων

1. Εισαγωγή στην VHDL
2. Δομές ακολουθιακού και συντρέχοντος κώδικα
3. Προχωρημένα στοιχεία της VHDL
4. Σύνταξη παραμετρικών περιγραφών
5. Σύνταξη κώδικα για λογική σύνθεση
6. Δομές ελέγχου/επαλήθευσης λειτουργίας των κυκλωμάτων
7. Μηχανές πεπερασμένων καταστάσεων
8. Υποδειγματική εργασία
9. Κυκλώματα επεξεργασίας δεδομένων - χειριστές δεδομένων
10. Μοντελοποίηση απλών επεξεργαστών
11. Τεχνικές περιγραφής και κυκλώματα για προχωρημένους

Εισαγωγικά

- Η VHDL αποτελεί μια γλώσσα **καταγραφής δομής** και **περιγραφής λειτουργικής συμπεριφοράς**
  - **Δεν** θα την αντιμετωπίσουμε ως άλλη μία γλώσσα διαδικαστικού προγραμματισμού
- Επιτρέπει τη μοντελοποίηση υλικού
  - Περιγραφή ψηφιακών κυκλωμάτων από το επίπεδο πύλης μέχρι το αλγοριθμικό επίπεδο
- VHDL είναι η συντομευμένη εκδοχή του αρκτικόλεξου "VHSIC HDL": Very High Speed Integrated Circuit Hardware Description Language
- Αναπτύχθηκε με βάση τη γλώσσα Ada ύστερα από αίτημα του αμερικανικού Department of Defense κατά τα μέσα της δεκαετίας του ΄80

Δυνατότητες και χαρακτηριστικά της VHDL

- Επιτρέπει τη χρήση διαφορετικών μεθοδολογιών σχεδιασμού και δεν προδιαθέτει ούτε επιβάλλει κάποια συγκεκριμένη
- Προσφέρει ανεξαρτησία από την εκάστοτε τεχνολογία υλοποίησης (standard cell VLSI, FPGA)
- Διευκολύνει την επικοινωνία σχεδίων μεταξύ συνεργαζόμενων ομάδων σχεδιασμού
- Βοηθά στην καλύτερη διαχείριση του έργου του σχεδιασμού
- Στην VHDL μπορεί να περιγραφεί ένα μεγάλο εύρος ψηφιακών κυκλωμάτων
- Χαρακτηριστικά της είναι:
  - Αυστηρή τυποποίηση η οποία βοηθά στον περιορισμό των σφαλμάτων
  - Βασίζεται στη χρήση εξωτερικών βιβλιοθηκών για την πρόσβαση σε τύπους δεδομένων και συχνά χρησιμοποιούμενες ρουτίνες

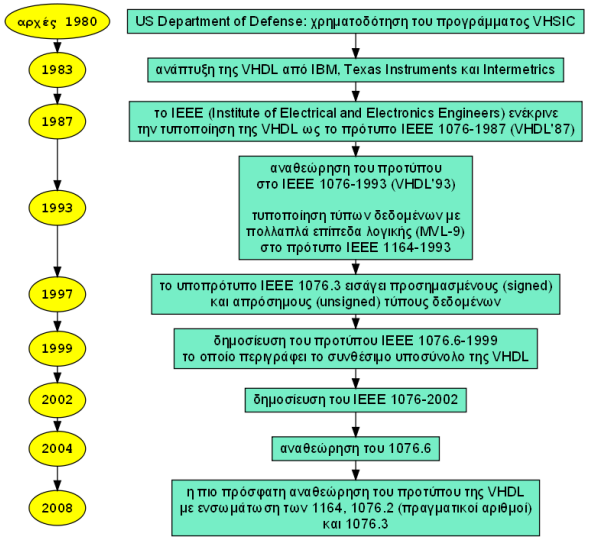
Λογική σύνθεση

- Σκοπός της περιγραφής ψηφιακών κυκλωμάτων είναι η υλοποίησή τους σε ολοκληρωμένο
- Η λογική σύνθεση είναι είδος μεταγλώττισης από το υψηλό επίπεδο μιας HDL (δομική, RTL ή μικτή) περιγραφής στο χαμηλό επίπεδο της λίστας κόμβων (netlist) με τα στοιχειώδη κυκλωματικά στοιχεία της τεχνολογίας
- Δημοφιλείς τεχνολογίες: διεργασίες τυποποιημένου κελιού (standard cell VLSI), FPGA
- Περιορισμοί στον τρόπο σχεδιασμού με μια HDL ώστε η τελική περιγραφή/κώδικας να είναι συνθέσιμη
- Κατάλληλες τεχνικές στην ανάπτυξη του κώδικα οδηγούν στην επίτευξη καλύτερων επιδόσεων (ως προς ταχύτητα επεξεργασίας, επιφάνεια ολοκληρωμένου, κατανάλωση ισχύος/ενέργειας)
- Για τη σύνθεση χρησιμοποιούνται εμπορικά εργαλεία (ISE Webpack, LeonardoSpectrum, Synopsys DC) ή εργαλεία ανοικτού κώδικα (ABC, Alliance, OCEAN, Signs, VPR) με ισχυρούς αυτοματισμούς (βελτιστοποίηση άκυκλων γράφων, παραγωγή διανυσμάτων ελέγχου, κ.α.)

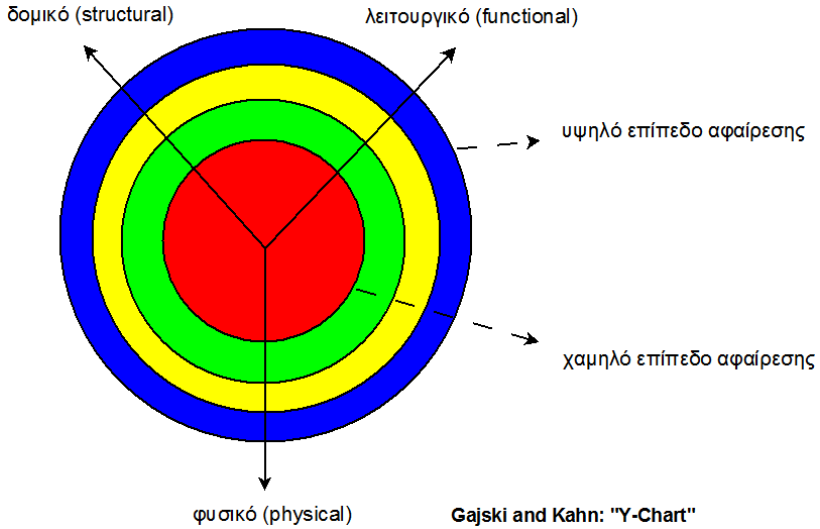
Επιλογή της κατάλληλης γλώσσας περιγραφής υλικού: VHDL και Verilog HDL

- Η VHDL αρχικά σχεδιάστηκε με σκοπό την **αυτοτεκμηρίωση (self-documentation) ψηφιακών συστημάτων**
- Με την ανάπτυξη κατάλληλων εργαλείων λογισμικού χρησιμοποιήθηκε για την **προσομοίωση** και τη **λογική σύνθεση** κυκλωμάτων
- Εν γένει, οποιοδήποτε κύκλωμα μπορεί να μοντελοποιηθεί σε VHDL μπορεί να μοντελοποιηθεί και στην Verilog HDL και το αντίστροφο
- Βασικά κριτήρια στην επιλογή γλώσσας περιγραφής υλικού (HDL) στο ψηφιακό σχεδιασμό είναι:
  - Διαθεσιμότητα εργαλείων ανάπτυξης
  - Δυνατότητα επαναχρησιμοποίησης κώδικα
  - Υποκειμενικά κριτήρια όπως οικειότητα με τις συντακτικές δομές της γλώσσας

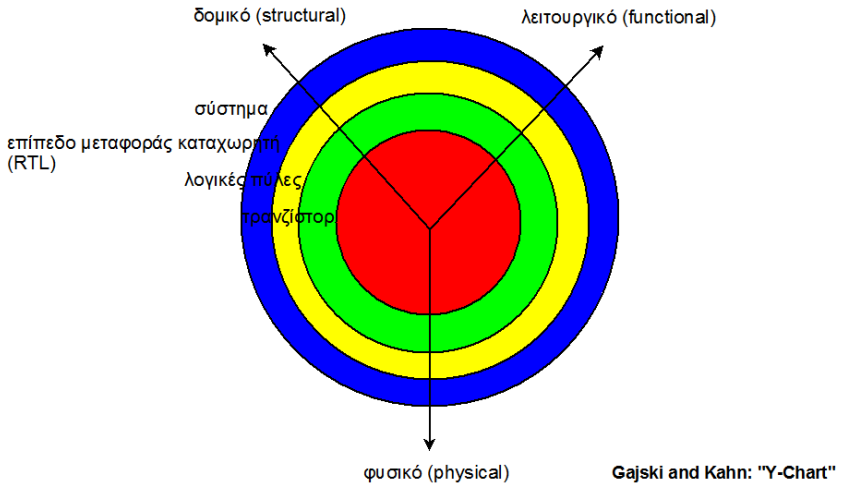
Ιστορική αναδρομή της ανάπτυξης της VHDL



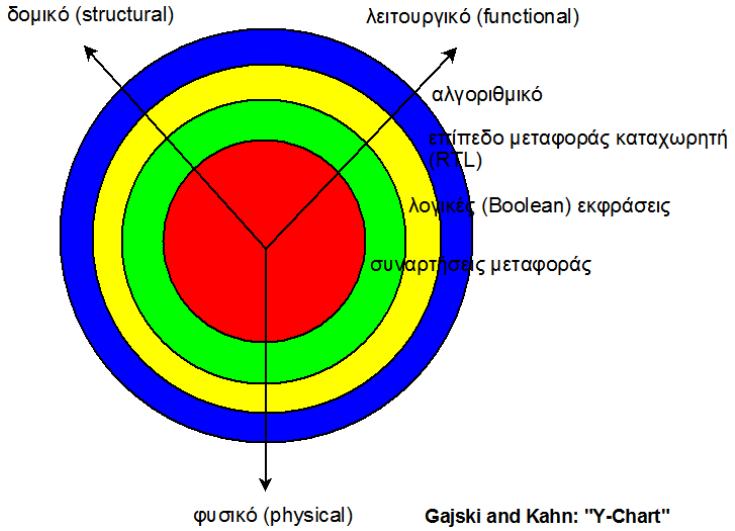
Πεδία και επίπεδα μοντελοποίησης (1)



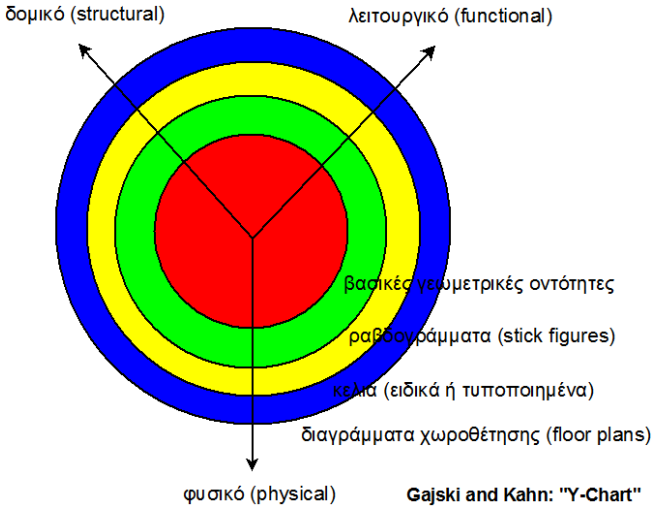
Πεδία και επίπεδα μοντελοποίησης (2)



Πεδία και επίπεδα μοντελοποίησης (3)



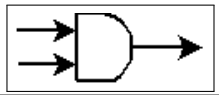
Πεδία και επίπεδα μοντελοποίησης (4)



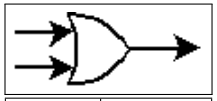
Πρακτικές που θα εξασκηθούν για τον ψηφιακό σχεδιασμό στο μάθημα

- Θα δοθεί βάρος στο σχεδιασμό κυκλωμάτων σε επίπεδο λογικών πυλών (gate level design) και σε επίπεδο μεταφοράς καταχωρητή (για συντομία: RTL)
- Οι περιγραφές κυκλωμάτων από το αλγοριθμικό επίπεδο (επίπεδο 'συμπεριφοράς') δεν είναι πάντα συνθέσιμες
  - Ακόμη και όταν είναι συνθέσιμες, τα φυσικά χαρακτηριστικά του παραγώμενου κυκλώματος είναι δύσκολο να εκτιμηθούν (μέγιστη επιτρεπτή συχνότητα ρολογιού, απαιτήσεις σε επιφάνεια υλικού)
  - Τα εργαλεία σύνθεσης υψηλού επιπέδου (High-Level Synthesis: HLS) στοχεύουν ακριβώς στη λογική σύνθεση από το αλγοριθμικό επίπεδο
- Στην πράξη, ο σχεδιασμός ενός ψηφιακού συστήματος ακολουθεί ιεραρχική δομή
  - Σχεδιασμός υπομονάδων: RTL
  - Διασύνδεση υπομονάδων για τη δημιουργία του συνολικού κυκλώματος: δομική περιγραφή

Προκαθορισμένες λογικές πύλες στη VHDL (1)



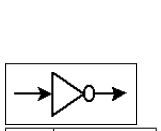
a	b	a and b
0	0	0
0	1	0
1	0	0
1	1	1



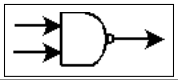
a	b	a or b
0	0	0
0	1	1
1	0	1
1	1	1

a	b	a xor b
0	0	0
0	1	1
1	0	1
1	1	0

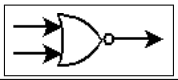
Προκαθορισμένες λογικές πύλες στη VHDL (2)



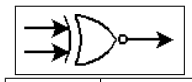
a	not a
0	1
1	0



a	b	a nand b
0	0	1
0	1	1
1	0	1
1	1	0

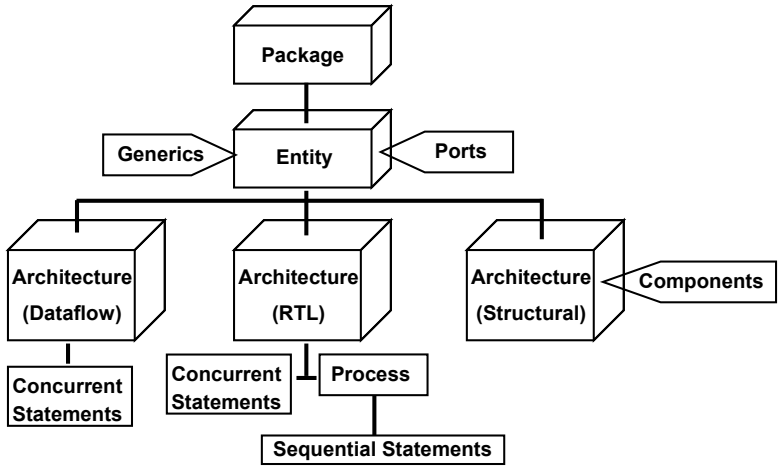


a	b	a nor b
0	0	1
0	1	0
1	0	0
1	1	0

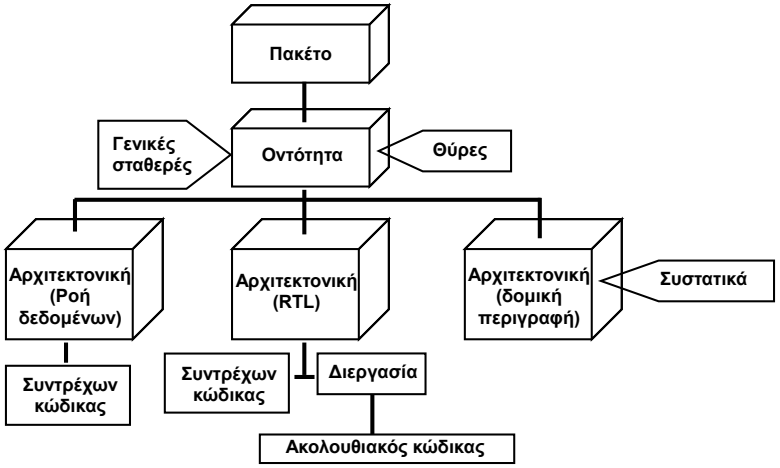


a	b	a xnor b
0	0	1
0	1	0
1	0	0
1	1	1

Ιεραρχικός σχεδιασμός στην VHDL (Αγγλικά)



Ιεραρχικός σχεδιασμός στην VHDL (Ελληνικά)



Ένα πολύ απλό παράδειγμα

- Στο παρακάτω μοντέλο η αρχιτεκτονική `rtl` της οντότητας `andgate` υλοποιεί μια πύλη AND

```
-- this is the entity
entity andgate is
  port (
    in1 : in BIT;
    in2 : in BIT;
    out1: out BIT
  );
end andgate;

architecture rtl of andgate is
begin
  out1 <= in1 and in2;
end rtl;
```

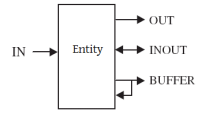
Θεμελιώδεις γνώσεις για τη συγγραφή κώδικα VHDL

- Οι βασικές δομές της VHDL είναι η **ΟΝΤΟΤΗΤΑ** (ENTITY) και η **ΑΡΧΙΤΕΚΤΟΝΙΚΗ** (ARCHITECTURE) της περιγραφής ενός κυκλώματος
  - ENTITY: Η διεπαφή του κυκλώματος (θύρες εισόδου και εξόδου) με το περιβάλλον
  - ARCHITECTURE: Καταγράφει τους απαιτούμενους μηχανισμούς λειτουργίας για την υλοποίηση του κυκλώματος
- Δεν υφίσταται ευαισθησία πεζών-κεφαλαίων (case insensitivity). Κεφαλαίοι και πεζοί χαρακτήρες χρησιμοποιούνται ελεύθερα για τη σύνταξη λέξεων-κλειδιών, τελεστών (τερματικά) και αναγνωριστικών (identifiers) που είναι μη τερματικά (non-terminals)
- Η γραμμή σχολίου δηλώνεται με δύο διαδοχικές παύλες: `--`
- Ο τύπος δεδομένων BIT είναι προκαθορισμένος και μπορεί να πάρει τις τιμές 0 ή 1

ΟΝΤΟΤΗΤΑ (1)

- Περιγράφει τον τρόπο διασύνδεσης του κυκλώματος
- Δήλωση των θυρών εισόδου/εξόδου προς και από το κύκλωμα
- Δήλωση *γενικών σταθερών* με εμβέλεια την οντότητα και τις αρχιτεκτονικές που υπάγονται σε αυτή

- Για μια θύρα δηλώνονται: όνομα, κατευθυντικότητα, τύπος δεδομένων
- Τύποι θυρών: *IN*, *OUT*, *INOUT*, *BUFFER*
  - IN: Είσοδος
  - OUT: Έξοδος (δεν διαβάζεται εσωτερικά)
  - INOUT: Είσοδος και έξοδος
  - BUFFER: Έξοδος με δυνατότητα εσωτερικής ανάγνωσης



ΟΝΤΟΤΗΤΑ (2)

☞ Καλό είναι να αποφεύγεται η χρήση του τύπου θύρας **buffer**. Όταν χρησιμοποιούνται ιεραρχικά, οι θύρες buffer μπορούν να συνδεθούν μόνο με άλλες θύρες buffer κάτι που υποχρεώνει το ιεραρχικό κύκλωμα να παρουσιάζει διεπαφή τύπου buffer. Αυτό περιορίζει την διασύνδεσιμότητα του κυκλώματος με άλλες μονάδες.

Σύνταξη μιας οντότητας:

```
entity name-of-entity is
  generic (
    generic_list with possible initializations
  );
  port (
    port_list
  );
end [entity] name-of-entity;
```


ΑΡΧΙΤΕΚΤΟΝΙΚΗ (1)

- Αποδίδει τη λειτουργικότητα (εσωτερικοί μηχανισμοί) του σχεδιαζόμενου κυκλώματος
- Καταγράφει δηλώσεις υποπρογραμμάτων, σταθερών, συστατικών και σημάτων στην περιοχή δηλώσεων
- Στο σώμα της αρχιτεκτονικής (architecture body) δίνεται ο τρόπος λειτουργίας του κυκλώματος
- Μπορεί να περιλαμβάνει συντρέχοντα ή/και ακολουθιακό κώδικα
  - Στο παράδειγμα της πύλης and χρησιμοποιείται μια συντρέχουσα ανάθεση (concurrent assignment)
- Κάθε κύκλωμα περιγράφεται από μία μόνο οντότητα, αλλά επιτρέπονται περισσότερες της μιας αρχιτεκτονικές υλοποιήσεις

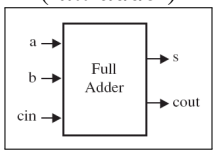
ΑΡΧΙΤΕΚΤΟΝΙΚΗ (2)

Σύνταξη μιας αρχιτεκτονικής:

```
architecture architecture-name is
  [architecture declarations]
begin
  [concurrent statements]
  [sequential statements]
  [structural code]
end [architecture] architecture-name;
```


Ο πλήρης αθροιστής δυαδικού ψηφίου: Προδιαγραφές

Η διεπαφή του πλήρους αθροιστή (full-adder)

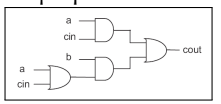


Ο πίνακας αληθείας του full-adder

a	b	cin	s	cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Υπολογισμός του ψηφίου s

Υπολογισμός του ψηφίου cout



Ο πλήρης αθροιστής δυαδικού ψηφίου: Υλοποίηση σε VHDL

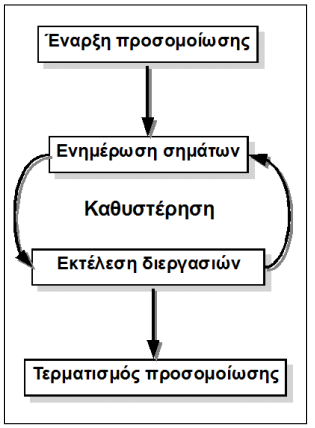
```
library IEEE;
use IEEE.std_logic_1164.all;

entity full_adder is
  port (
    a    : in  STD_LOGIC;
    b    : in  STD_LOGIC;
    cin  : in  STD_LOGIC;
    s    : out STD_LOGIC;
    cout : out STD_LOGIC
  );
end full_adder;

architecture structural of full_adder is
begin
  s    <= a xor b xor cin;
  cout <= (a and b) or (a and cin) or (b and cin);
end structural;
```


Ο κύκλος της προσομοίωσης κυκλωματικών περιγραφών σε VHDL

Η VHDL χρησιμοποιεί τον παρακάτω κύκλο προσομοίωσης για την μοντελοποίηση της διέγερσης (stimulus) και της απόκρισης (response) των ψηφιακών κυκλωμάτων

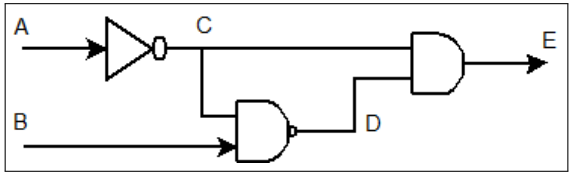


Η έννοια της χρονικής καθυστέρησης (delay) στην VHDL (1)

- Οι αναθέσεις σημάτων παρουσιάζουν μια χρονική καθυστέρηση για την ενημέρωση της εξόδου ως προς τις εφαρμοζόμενες τιμές εισόδου
- Τύποι (χρονικής) καθυστέρησης:
  - Αδρανειακή (inertial) καθυστέρηση: καθυστέρηση διάδοσης λόγω εσωτερικής αδράνειας απόκρισης. Παράδειγμα:
    `output <= not (input) after 10 ns;`
  - Καθυστέρηση βήματος 'δέλτα' (delta delay): η μονάδα χρόνου *δέλτα* προσδιορίζει το χρόνο που χρειάζεται για την ανανέωση των σημάτων στο κύκλωμα (εσωτερικών ή εξόδων)
- Υπολογισμοί των σημάτων σε βήματα delta μέχρις ότου δεν υπάρχει δραστηριότητα (activity) στο κύκλωμα

Η έννοια της χρονικής καθυστέρησης (delay) στην VHDL (2)

Παράδειγμα:



Δέλτα	Ενέργειες
1	A: 1→0, B=1 υπολ. NAND,NOT
2	C: 0→1 υπολ. NAND,AND
3	D: 1→0, E:0→1 υπολ. AND
4	E: 1→0

Οι τύποι BIT, STD_ULOGIC, STD_LOGIC και οι αντίστοιχοι διανυσματικοί τύποι (1)

- Τα σήματα αντιστοιχούν σε φυσικό επίπεδο με 'καλώδια' (wires) τα οποία διασυνδέουν τις διάφορες υπομονάδες
- Σαν σήματα αντιμετωπίζονται και οι θύρες μιας οντότητας
- Προκαθορισμένοι τύποι στην VHDL είναι:
  - BIT: '0', '1'
  - Διάνυσμα (μονοδιάστατος πίνακας) από BIT (BIT_VECTOR): π.χ. στο myvector(0 to 3) αναθέτουμε την τιμή '0011'.
  - ☞ Οι εκφράσεις <LSB> `to` <MSB> και <MSB> `downto` <LSB> δηλώνουν περιοχή τιμών ή διευθύνσεων
- Το πρότυπο IEEE 1164-1993 καθορίζει τους εξής τύπους σημάτων: std_ulogic και std_logic με τους διανυσματικούς τύπους std_ulogic_vector και std_logic_vector

Οι τύποι BIT, STD_ULOGIC, STD_LOGIC και οι αντίστοιχοι διανυσματικοί τύποι (2)

- Ο τύπος std_ulogic υλοποιεί λογική MVL9 (με 9 επίπεδα λογικής)
- Στον std_ulogic δύο ή περισσότερα συγκρουόμενα (conflicting) λογικά επίπεδα (π.χ. κατά την οδήγηση μιας τιμής από διαφορετικές πηγές) δεν μπορούν να αναλυθούν. Ο τύπος std_logic (MVL8: MVL9 χωρίς το 'U') προσφέρει αυτόματη επίλυση

Επίπεδο	Ερμηνεία
U	uninitialized
X	forcing unknown
0	forcing 0
1	forcing 1
Z	high impedance
W	weak unknown
L	weak 0
H	weak 1
–	don't care

	X	0	1	Z	W	L	H	-
X	X	X	X	X	X	X	X	X
0	X	0	X	0	0	0	0	X
1	X	X	1	1	1	1	1	X
Z	X	0	1	Z	W	L	H	X
W	X	0	1	W	W	W	W	X
L	X	0	1	L	W	L	W	X
H	X	0	1	H	W	W	H	X
-	X	X	X	X	X	X	X	X

Λέξεις-κλειδιά (keywords) της VHDL

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

Μπλε: Χρήση σε συνθέσιμο κώδικα Κυανό: Εξάρτηση από συγκεκριμένες βιβλιοθήκες Κόκκινο: Όχι για σύνθεση
Μωβ: Δεν συστήνεται η χρήση του Πράσινο: Σε κώδικα επαλήθευσης (testbench)

Τύποι δεδομένων για την VHDL

Τύποι δεδομένων, προκαθορισμένες σταθερές και ένα σύνολο από βοηθητικές υπορουτίνες (συναρτήσεις, διαδικασίες) είναι δηλωμένα στα *ΠΑΚΕΤΑ* (PACKAGES) των *ΒΙΒΛΙΟΘΗΚΩΝ* IEEE και STD:

- *standard* της std: Ορίζει τους τύπους δεδομένων BIT, BOOLEAN, INTEGER, REAL
- *std_logic_1164* της IEEE: STD_ULOGIC, STD_LOGIC
- *numeric_std* της IEEE: SIGNED, UNSIGNED καθώς και αριθμητικούς, λογικούς και τελεστές σύγκρισης για αυτούς
- *std_logic_unsigned, std_logic_signed* στην IEEE βιβλιοθήκη γραμμένα από την Synopsys: Αριθμητικοί και τελεστές σύγκρισης για STD_LOGIC_VECTOR
- *std_logic_arith* της Synopsys: SIGNED, UNSIGNED καθώς και αριθμητικούς και τελεστές σύγκρισης για αυτούς

Συχνά χρησιμοποιούμενοι τύποι της VHDL

ΤΥΠΟΣ	ΤΙΜΗ	ΠΡΟΕΛΕΥΣΗ
std_ulogic	'U', 'X', '0', '1', 'Z', 'W', 'L', 'H', '-'	std_logic_1164
std_ulogic_vector	array of std_ulogic	std_logic_1164
std_logic	resolved std_ulogic	std_logic_1164
std_logic_vector	array of std_logic	std_logic_1164
unsigned	array of std_logic	numeric_std, std_logic_arith
signed	array of std_logic	numeric_std, std_logic_arith
boolean	true, false	standard
character	191 / 256 characters	standard
string	array of character	standard
integer	$-(2^{31}-1)$ to $2^{31}-1$	standard
real	$-1.0E38$ to $1.0E38$	standard
time	1 fs to 1 hr	standard

Τύποι δεδομένων στο package STANDARD

- BIT: `0, 1`
- BIT_VECTOR: `"001100", X"00FF"` στο δεκαεξαδικό
- BOOLEAN: `true, TRUE, TruE` για το αληθές και `False, false, FALsE` για το ψευδές
- CHARACTER: `'A', 'a', '@', '''`. Ένας πίνακας από `CHARACTER` αποτελεί συμβολοσειρά (string): `"hold time out of range", "i'll be back", "0$#1324"`
- REAL: `-1.0, +2.35, 36.6, -1.0E+38`
- INTEGER με εύρος τιμών {-2,147,483,647, +2,147,483,647}: $+1$, 862, -257, $+15$
- TIME: `10 ns, 100 us, 6.3 ns`

Οι τύποι SIGNED και UNSIGNED

Περιοχή τιμών:
`unsigned:` `0` to 2^N - 1
`signed:` -2^{N-1} to 2^{N-1}-1 (για αναπαράσταση συμπλήρωμα ως-προς-2)
Χρήση παραπλήσια με τον τύπο std_logic_vector:

```
signal A_unsigned : unsigned(3 downto 0);
signal B_signed : signed (3 downto 0);
signal C_slv : std_logic_vector (3 downto 0);
--
--
A_unsigned <= "1111"; -- 15 decimal
B_signed <= "1111"; -- -1 decimal
C_slv <= "1111"; -- 15 decimal only if using std_logic_unsigned
```


ΒΙΒΛΙΟΘΗΚΗ (LIBRARY)

Μια LIBRARY αποτελεί συλλογή από κοινώς χρησιμοποιούμενα τμήματα κώδικα. Απαρτίζεται από ΠΑΚΕΤΑ (PACKAGES) τα οποία περιλαμβάνουν δηλώσεις COMPONENTS και δηλώσεις/υλοποιήσεις FUNCTIONS και PROCEDURES

```
LIBRARY library_name;
USE library_name.package_name.package_parts;
```

Συχνά χρησιμοποιούμε τις βιβλιοθήκες IEEE, STD και WORK. Οι STD και WORK δεν χρειάζεται να δηλωθούν. Στην WORK μεταγλωττίζονται τα αρχεία πηγαίου κώδικα του χρήστη, κατά σύμβαση:

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.numeric_std.all;
USE STD.standard.all;
USE work.all;
```

Με `all` ζητείται το συνολικό περιεχόμενο ενός package

Ονοματοδοσία αναγνωριστικών

- Ένα όνομα αναγνωριστικού ξεκινά με αλφαβητικό χαρακτήρα (`a--z`) και ακολουθείται από αλφαριθμητικό χαρακτήρα ή υπογράμμιση (underscore)
- Η VHDL είναι case-insensitive και έτσι το `x` δεν διαφέρει από το `X`
- Οι λέξεις-κλειδιά της VHDL δεν είναι έγκυρα αναγνωριστικά
  - Έγκυρα αναγνωριστικά: `xyz, red, marker, Nexus6`
  - Μη αποδεκτά: `in, out, signal, port`
- Επιτρέπονται ιεραρχικά αναγνωριστικά με τα επίπεδα της ιεραρχίας διακρινόμενα με ΄.΄: `library_name.item_name, my_defs.unit_delay`

Αντικείμενα στην VHDL: ΣΤΑΘΕΡΑ (CONSTANT), ΣΗΜΑ (SIGNAL) και ΜΕΤΑΒΛΗΤΗ(VARIABLE)

- Αντικείμενο στην VHDL:
  - Σταθερής (CONSTANT) ή μεταβαλλόμενης τιμής (SIGNAL, VARIABLE)
  - Η δήλωση ενός αντικειμένου περιλαμβάνει ΥΠΟΧΡΕΩΤΙΚΑ όνομα (αναγνωριστικό) και τύπο
  - Ο τύπος ενός αντικειμένου δεν μπορεί να μεταβληθεί
  - Βαθμωτά (scalar) ή διανυσματικά (πίνακας: array) αντικείμενα
  - Για αντικείμενα τύπου array
    - Αν `S` το όνομα ενός array τύπου std_logic_vector
    - `S(3)` είναι ένα στοιχείο του και αποτελεί βαθμωτό αντικείμενο τύπου std_logic
    - `S(4 downto 1)` είναι ένα πεδίο (φέτα: slice) του πίνακα

Δήλωση ΣΤΑΘΕΡΑΣ (CONSTANT)

- Σε μια ΣΤΑΘΕΡΑ ανατίθεται μία τιμή η οποία δεν μπορεί στη συνέχεια να μεταβληθεί
- Οι σταθερές αντικαθιστούν συχνά εμφανιζόμενες αριθμητικές τιμές ή συμβολοσειρές
- Σταθερές οι οποίες είναι δηλωμένες σε ένα PACKAGE έχουν καθολική εμβέλεια
- Σύνταξη μιας CONSTANT:
  constant identifier : type-indication [:=expression];
- Παραδείγματα δηλώσεων για CONSTANTS:

```
constant PI : REAL := 3.147592;
constant CYCLE : TIME := 100 ns;
constant FIVE : INTEGER := 3;
constant FIVE : BIT_VECTOR := "0101";
```

- Δηλώνονται σε πακέτα, στην περιοχή δηλώσεων μιας οντότητας, στην περιοχή δηλώσεων μιας αρχιτεκτονικής, σε υποπρογράμματα

Δήλωση ΜΕΤΑΒΛΗΤΗΣ (VARIABLE)

- Μια ΜΕΤΑΒΛΗΤΗ αποτελεί ένα αντικείμενο στο οποίο κάποια στιγμή ανατίθεται μία τιμή η οποία μπορεί να μεταβληθεί εντός μιας διεργασίας (PROCESS)
- Χρησιμοποιείται στη σύνταξη ακολουθιακού κώδικα
- Μια VARIABLE μπορεί να δηλωθεί με περιοχή (range) τιμών
- Η ανάθεση μιας νέας τιμής στην VARIABLE συμβαίνει ακαριαία
- Σύνταξη της δήλωσης μιας VARIABLE:
  variable identifier : type-indication [constraint] [:=expression];
- Παραδείγματα δηλώσεων VARIABLES:

```
variable index: INTEGER range 1 to 50 := 50;
variable x, y : INTEGER;
variable cycle_time : TIME range 10 ns to 50 ns := 15 ns;
variable memory : STD_LOGIC_VECTOR(0 to 7);
```

- Μπορούν να δηλωθούν σε διεργασίες και υποπρογράμματα

Δήλωση ΣΗΜΑΤΟΣ (SIGNAL)

- Τα ΣΗΜΑΤΑ χρησιμοποιούνται για την υλοποίηση διασυνδέσεων εντός ενός κυκλώματος αλλά και για την εξωτερική διασύνδεση διαφορετικών μονάδων σχεδιασμού
- Ένα SIGNAL μπορεί να δηλωθεί όπου και μία CONSTANT
- Χρησιμοποιείται εντός διεργασιών σε ακολουθιακό κώδικα και εκτός διεργασιών σε συντρέχοντα κώδικα
- ☞ Η ενημέρωση ενός SIGNAL με νέα τιμή σε μια διεργασία ΔΕΝ είναι ακαριαία αλλά πραγματοποιείται σε χρόνο προσομοίωσης
- Σύνταξη της δήλωσης ενός SIGNAL:
  signal identifier : type-indication [constraint] [:=expression];
- Παραδείγματα δηλώσεων SIGNALS:

```
signal control: BIT := '0';
signal count: INTEGER range 0 to 100;
signal y: STD_LOGIC_VECTOR(7 downto 1);
```


Τελεστές της VHDL (VHDL operators)

Συνοπτικός πίνακας των τελεστών

λογικοί	and	or	nand	nor	xor	xnor	not
αριθμητικοί	+	-	*	/	mod	rem	
σύγκρισης	=	/=	<	<=	>	>=	
ολίσθησης	sll	srl	sla	sra	rol	ror	
μοναδιαίοι	+	-					
άλλοι	**	abs	&				

- Οι τελεστές αναγωγής σε δύναμη "**", απόλυτης τιμής "abs", και υπολογισμού ακέραιου υπολοίπου ("mod", "rem") δεν είναι συνθέσιμοι
- Οι τελεστές πολλαπλασιασμού και διαίρεσης υποστηρίζονται από ορισμένα εργαλεία λογικής σύνθεσης υπό προϋποθέσεις
- Η διαφορά των mod και rem είναι ότι: το `A rem B` παίρνει το πρόσημο του A ενώ το `A mod B` το πρόσημο του B

Προτεραιότητα τελεστών