Εισαγωγή στη VHDL

VHDL

VHSIC Hardware Description Language

VHSIC : Very-High-Speed Integrated Circuits

Εισαγωγή στη VHDL

- Ιστορική Αναδρομή
- Λόγοι για την χρήση της VHDL
- Ανάπτυξη ηλεκτρονικού κυκλώματος
- Σύνθεση (Synthesis)
- Επίπεδα προγραμματισμού σε VHDL
- Προσομοίωση (Simulation)
- Σχεδίαση με τη λογική της Ιεραρχίας
- VHDL component
- Γλώσσες Προγραμματισμού

Ιστορική Αναδρομή

- η ανάπτυξη της VHDL ξεκίνησε στις αρχές της δεκαετίας του 1980 από το American Defense Department
- βασίστηκε στην γλώσσα προγραμματισμού ADA
- καθιερώθηκε το 1987 (VHDL-87) από την IEEE (Institute of Electrical and Electronics Engineers)
- 1993: βελτιωμένη έκδοση της γλώσσας (VHDL-93)
- σήμερα αποτελεί μία από τις πλέον σημαντικές καθιερωμένες γλώσσες προγραμματισμού για τον ορισμό, τον έλεγχο και τη σχεδίαση ηλεκτρονικών κυκλωμάτων

Λόγοι για την χρήση της VHDL

- σχεδιασμός ηλεκτρονικών components (στοιχείων)
  - ανεξάρτητος τεχνολογίας (technology-independent)
  - ανεξάρτητος τεχνολογίας για μία δεδομένη τεχνική οικογένεια

- υποστηρίζει το περιβάλλον ανάπτυξης ψηφιακών κυκλωμάτων:
  - Ιεραρχιών (Structural VHDL)
  - Παράλληλου σχεδιασμού (Concurrent VHDL)
  - Ακολουθιακού σχεδιασμού (Sequential VHDL)
  - Ccomponents τα οποία μπορούν να ξαναχρησιμοποιηθούν
  - Έλεγχος σφαλμάτων και επιβεβαίωση (error management – verification)

- προσομοίωση του κώδικα VHDL ενός component (simulator)

Λόγοι για την χρήση της VHDL

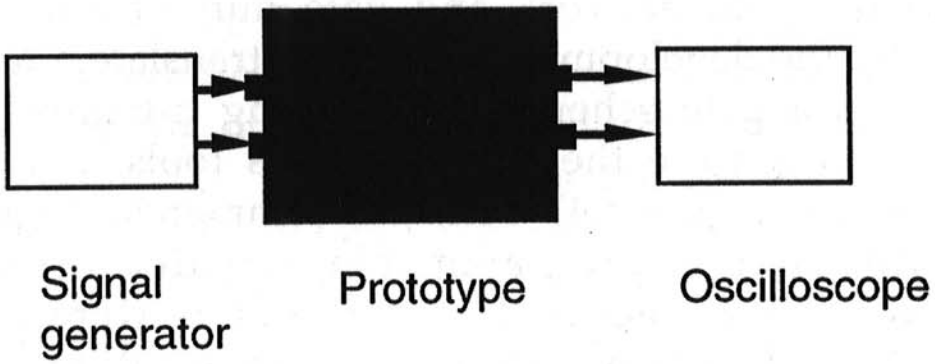
- επιτρέπει την εστίαση στη λογική του προβλήματος,
  χωρίς να μας απασχολούν προβλήματα όπως χρονισμός, μέγεθος,
  επιλογή components, οδήγηση ισχύος, σήματα εξόδου κτλ.,
  σε αντίθεση με παραδοσιακά σχηματικά σχέδια

- όντας καθιερωμένη γλώσσα προγραμματισμού,
  η VHDL επιτρέπει τη μεταφορά κώδικα
  μεταξύ διαφόρων αναπτυξιακών συστημάτων modeling, π.χ.
  - ViewLogic
  - Mentor Graphics
  - Synopsys

- δεν χρησιμοποιείται για σχεδιασμό αναλογικών ηλεκτρονικών
  Εν αναμονή της AHDL (Analog HDL) …

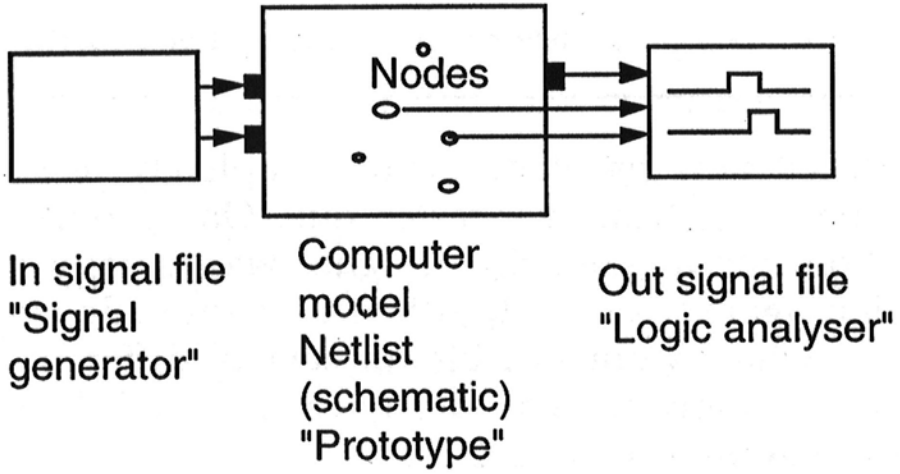
Ανάπτυξη ηλεκτρονικού κυκλώματος

- ανάπτυξη πρωτοτύπου
- έλεγχός του με μία γεννήτρια σημάτων (signal generator) και έναν παλμογράφο (oscilloscope)



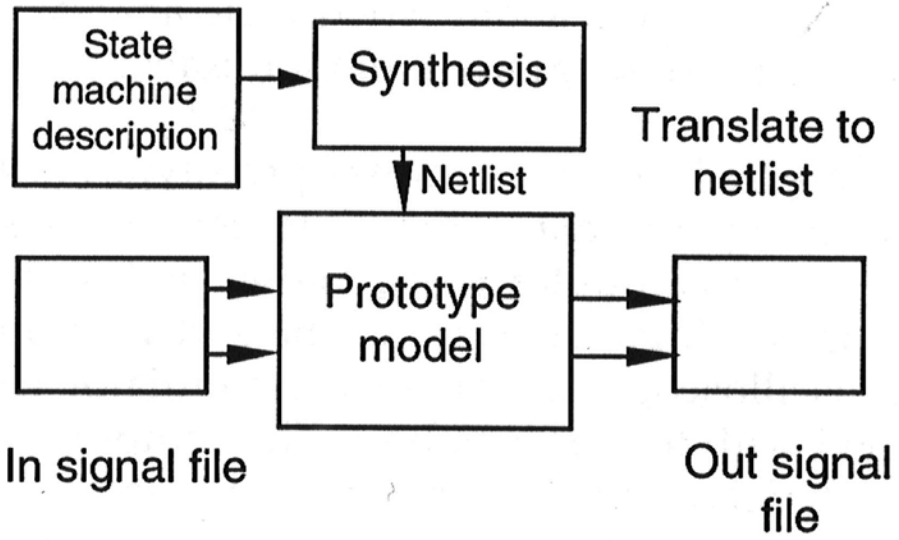
Ανάπτυξη ηλεκτρονικού κυκλώματος

- ανάπτυξη κατάλληλων εργαλείων σε υπολογιστή.
  - προϋποθέτει την ανάπτυξη σε λογισμικό (software)
    ενός μοντέλου του πρωτοτύπου (prototype),
    μίας γεννήτριας σημάτων (signal generator)
    και ενός λογικού αναλυτή (logic analyser)
- επιτρέπει την ακριβή προσομοίωση



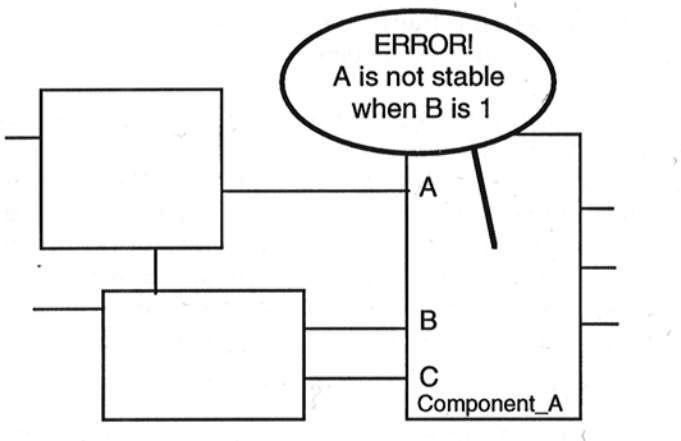
Ανάπτυξη ηλεκτρονικού κυκλώματος

- Για να ελαττωθεί ο χρόνος σχεδιασμού εξισώσεων Boole αναπτύχθηκαν νέα εργαλεία τα οποία “μεταφράζουν” τις εξισώσεις σε σχηματικό με πύλες
- Η διαδικασία αυτή ονομάστηκε Λογική Σύνθεση.



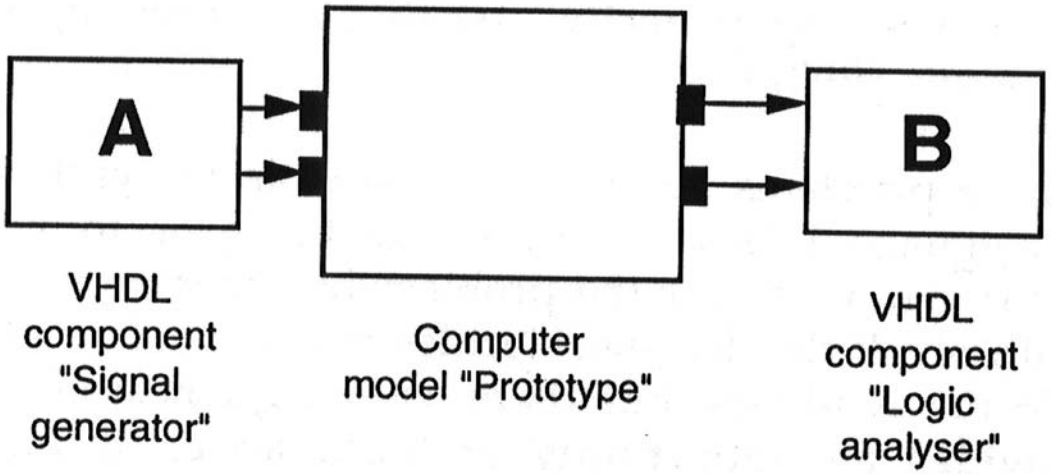
Ανάπτυξη ηλεκτρονικού κυκλώματος

- Προβλήματα
  - Κάθε εργαλείο χρησιμοποιεί την δική του γλώσσα.
  - Η εξαφάνιση μιας γενιάς ηλεκτρονικών εξαρτημάτων από την αγορά σήμαινε τον επανασχεδιασμό από την αρχή.
  - Η αύξηση της πολυπλοκότητας των σχεδίων καθιστούσε δύσκολο τον έλεγχο της λογικής του κυκλώματος.
- Η VHDL έλυσε αρκετά από τα παραπάνω προβλήματα, ενώ είναι δυνατός ο έλεγχος κάθε component (στοιχείου).



Ανάπτυξη ηλεκτρονικού κυκλώματος

- η VHDL δίνει επίσης τη δυνατότητα ανάπτυξης testbenches, όπου το πρωτότυπο μπορεί να προέρχεται από άλλο σχεδιαστικό εργαλείο



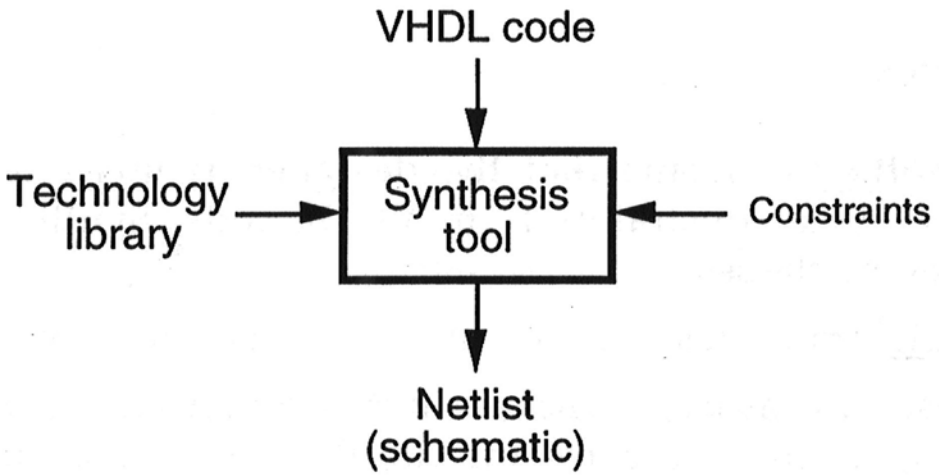
Σύνθεση (Synthesis)

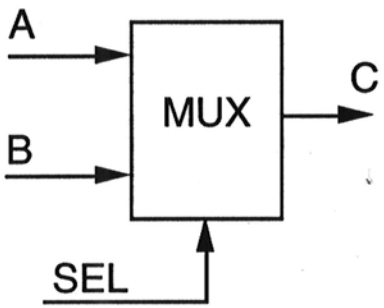
το λογισμικό της Σύνθεσης “μεταφράζει”, ελαχιστοποιεί και αναγνωρίζει τους χρονικούς περιορισμούς ενός κυκλώματος το οποίο περιγράφεται από κώδικα σε VHDL

- Logic Synthesis
  - “Μεταφράζει” και ελαχιστοποιεί συναρτήσεις Boole σε πύλες
- RTL (Register Transfer Level) Synthesis
  - το ίδιο με την Logic Synthesis και επί πλέον “μεταφράζει” την ακολουθιακή λογική σε πύλες και flip-flops (state machines)
- Behavioral Synthesis
  - Ενα component μπορεί να χρησιμοποιηθεί για πολλές παράλληλες και ακολουθιακές λογικές

Σύνθεση (Synthesis)

```
process(sel,a,b)
begin
    if sel='1' then
            c<=b;
    else
            c<=a;
    end if;
end process;
```

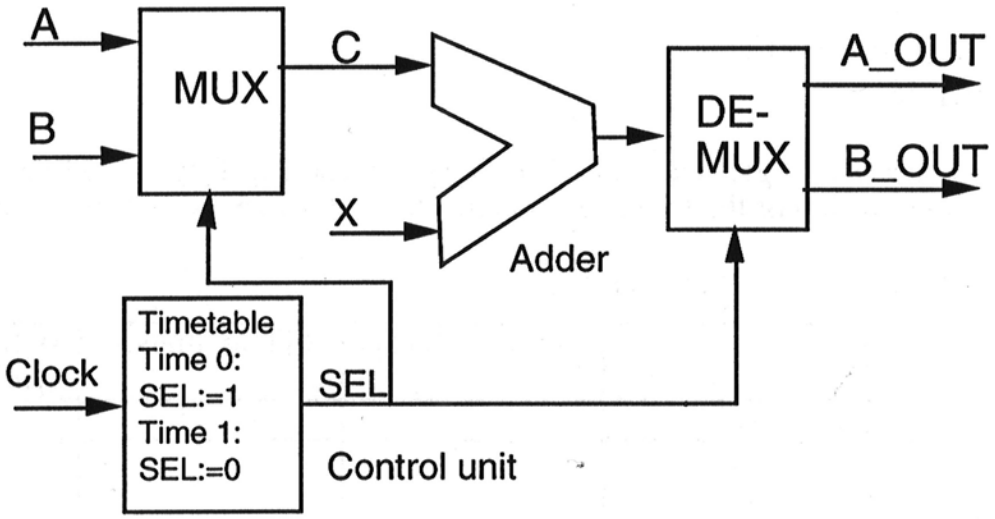




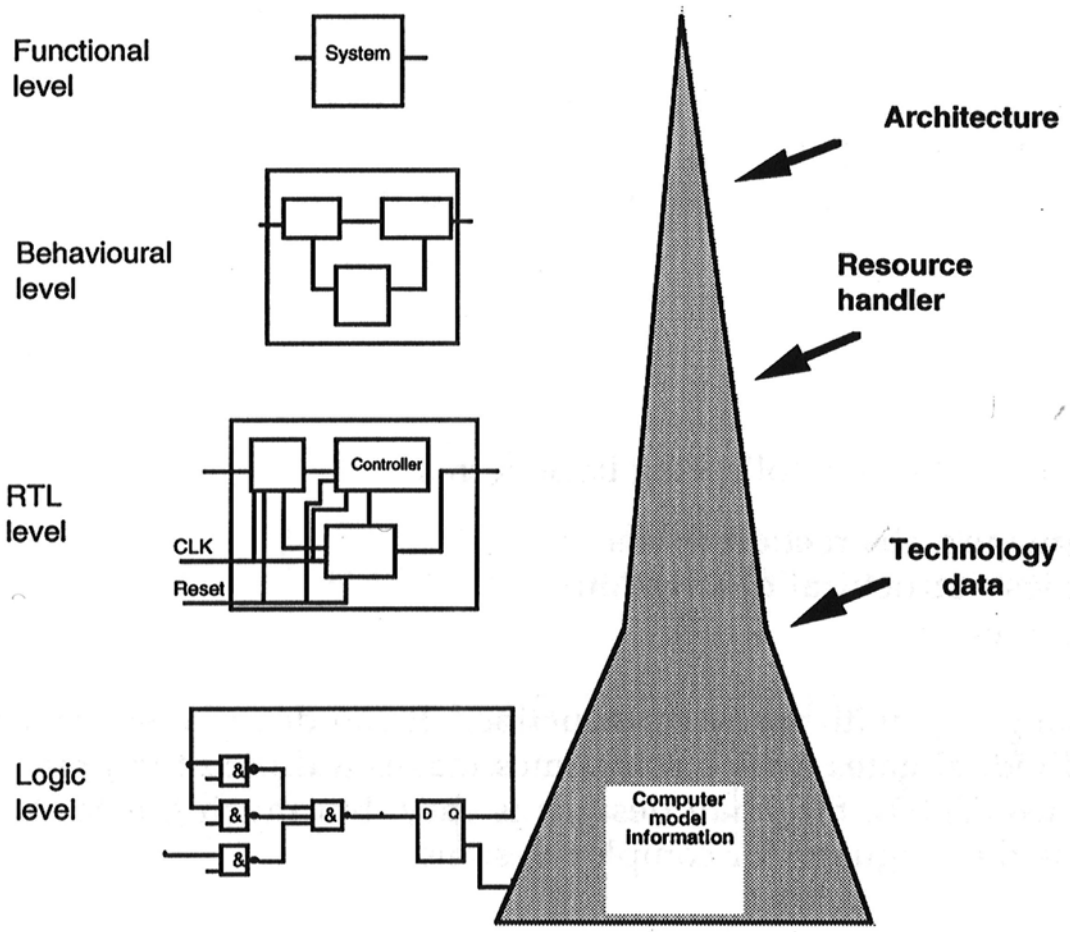
Σύνθεση (Synthesis)

- σε μεγάλα σχέδια υπάρχει η ανάγκη επανάληψης συναρτήσεων (functions) σε διάφορα μέρη
  - ενώ στο software αυτό είναι εύκολο με την χρήση των functions στο hardware είναι δυσκολότερο

- Παράδειγμα Behavioral Synthesis με χρήση πινάκων χρόνου.



Επίπεδα προγραμματισμού σε VHDL



Επίπεδα προγραμματισμού σε VHDL

- Functional Level:
  - περιγραφή και εξομοίωση ενός αλγορίθμου
  - δεν απαιτείται ο αλγόριθμος να περιέχει πληροφορία χρονισμού
- Behavioral Level:
  - περιγραφή της “συμπεριφοράς” και του χρονισμού
  - δεν απαιτείται ορισμός αρχιτεκτονικής
  - ταχύτατη ανάπτυξη και έλεγχος (simulation-verification) του μοντέλου
- RT Level (Register Transfer Level):
  - περιγραφή ασύγχρονων και σύγχρονων state machines, data paths, operators, registers, …
- Logic or Gate Level:
  - Περιγραφή με άλγεβρα Boole ή κύκλωμα σε πύλες

Προσομοίωση (Simulation)

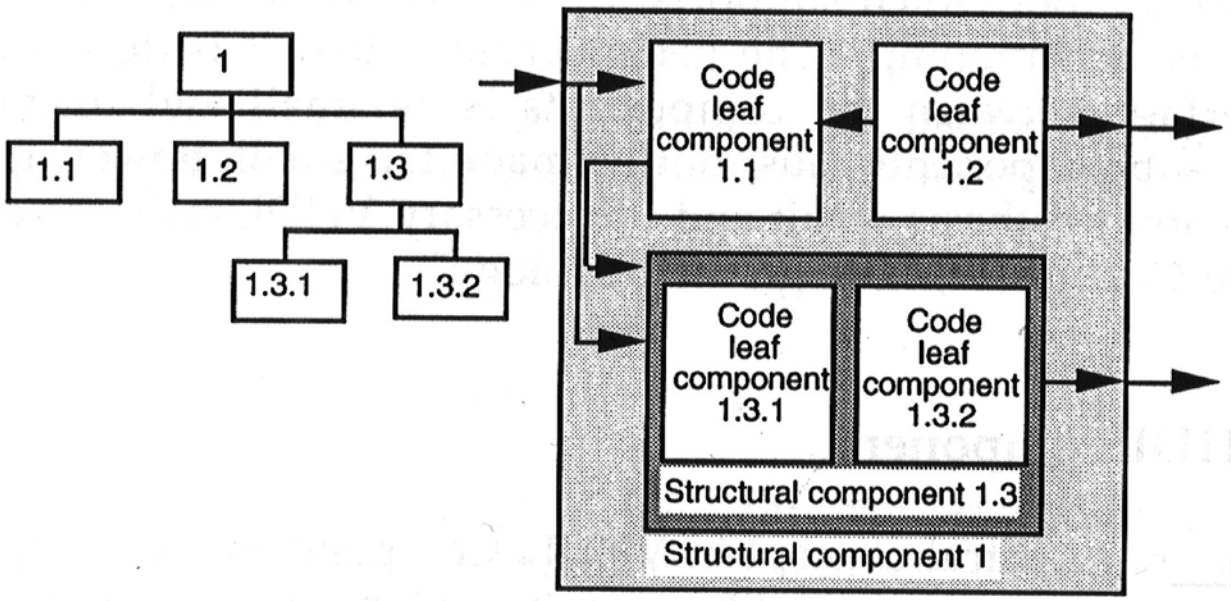
- η ακρίβεια στην προσομοίωση
  εξαρτάται από το επίπεδο προγραμματισμού

- ο χρόνος της προσομοίωσης αυξάνει
  καθώς κατεβαίνουμε σε επίπεδο προγραμματισμού

Abstraction levels	Description	Time granularity
Behavioural level	Behavioural level	microsecond
RT level	RTL language	clock cycles
Logic level	Boolean equations	nanosecond

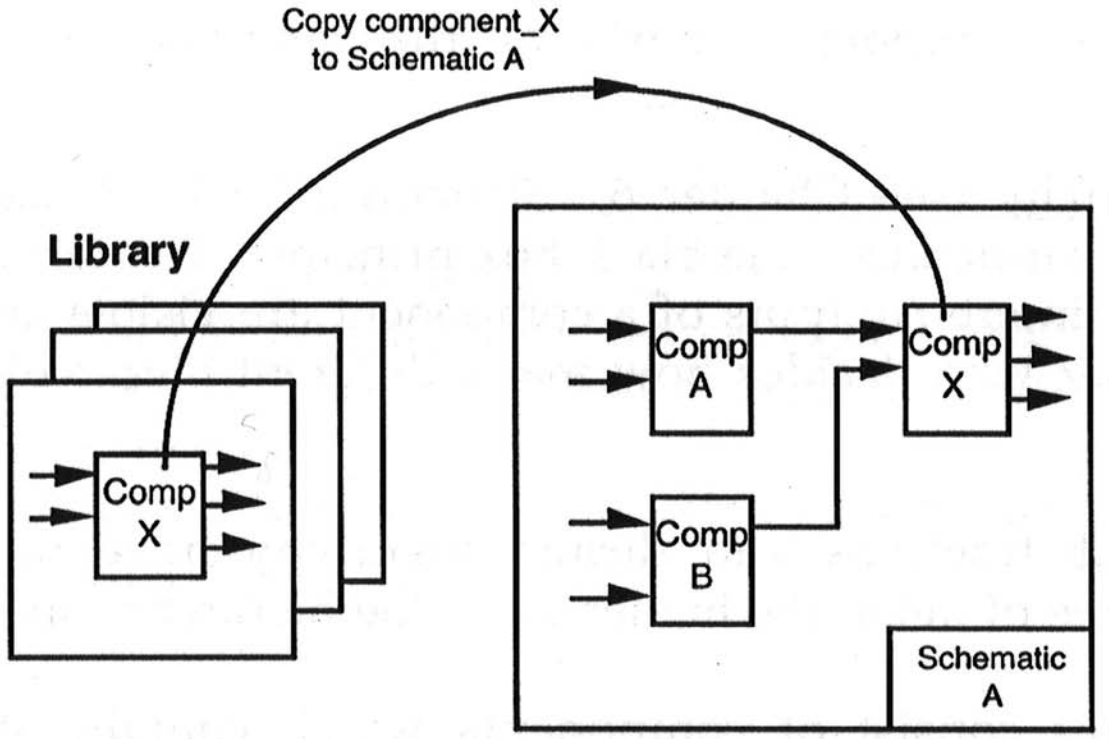
Σχεδίαση με τη λογική της Ιεραρχίας

- η κατάλληλη επιλογή επιπέδου προγραμματισμού και η χρήση λογικής Ιεραρχίας (Structural VHDL) συνήθως μειώνει την πολυπλοκότητα του προβλήματος



VHDL component

- δημιουργία Βιβλιοθηκών (Libraries) με components



Γλώσσες Προγραμματισμού

- **VHDL**
- **VERILOG** (βασισμένη στη C)

SLIDE	Structured Language for Interface Description and Evaluation (Parker and Wallace, 1981)
CONLAN	CONsensus LANguage (Piloty *et al.*, 1983)
ISPS	Instruction Set Processor Specification (Barbacci *et al.*, 1979
ADLIB	A Design Language for Indicating Behaviour [Hill *et al.*, 1979]
HILL	HIerarchical Layout Language (Lengauer and Melhorn, 1984)
OODE	Object Oriented Description Environment for computer hardware (Takeuchi, 1981)
BORIS	Block-ORiented Interacting Simulation system (Decker and Maierhofer, 1984)