Σχεδίαση Υπολογιστικών Συστημάτων

Εισαγωγή

Μιχάλης Ψαράκης

Εισαγωγή

- Βιβλιογραφία:
  - **Βιβλίο Α: «Ψηφιακή Σχεδίαση – Ενσωματωμένα Συστήματα με VHDL», P.J. Ashenden. Εκδόσεις Νέες Τεχνολογίες**
  - Βιβλίο Β: «Σχεδιασμός Κυκλωμάτων με τη VHDL», V.A. Pedroni. Εκδόσεις Κλειδάριθμος
- Διαφάνειες
  - Μπορείτε να τις κατεβάσετε από την ιστοσελίδα του μαθήματος
- Επιπλέον βιβλιογραφία
  - Μπορείτε να βρείτε στην ιστοσελίδα του μαθήματος
  - Διατίθεται σε ηλεκτρονική μορφή μέσω της βιβλιοθήκης:
    - The designer's guide to VHDL (3rd edition), P.J. Ashenden, Morgan Kaufmann, 2008
- Ιστοσελίδα μαθήματος: http://gunet2.cs.unipi.gr/eclass/courses/TMC112/

Χρονοδιάγραμμα μαθήματος

- Τις πρώτες 2 εβδομάδες:
  - Τετάρτη 14.15-16.00: θεωρία
  - Πέμπτη 8.15-10.00: θεωρία
- Τις επόμενες εβδομάδες:
  - Τετάρτη 14.15-16.00: θεωρία
  - Μία μέρα (x2 ώρες) εργαστήριο (στον 2ο όροφο)

Βαθμολογία μαθήματος

- Μέσω εργασιών:
  - 3 ή 4 ατομικές ασκήσεις: 10% βαθμού η καθεμία
    - Προθεσμία παράδοσης 1 ή 2 βδομάδες
    - Εξέταση στο εργαστήριο
    - Μπορείτε να τις παραδώσετε μόνο κατά την διάρκεια του εξαμήνου
  - 1 ομαδική εργασία: 60-70% βαθμού
    - Μπορείτε να τις παραδώσετε στις εξεταστικές του Ιουνίου και Σεπτεμβρίου
  - Οι εργασίες δεν ισχύουν για τα επόμενα ακαδημαϊκά έτη

Περιεχόμενο μαθήματος

- Σχεδίαση ψηφιακών κυκλωμάτων με χρήση γλώσσας περιγραφής υλικού
  - VHDL
- Σχεδίαση απλών ψηφιακών κυκλωμάτων:
  - Συνδυαστικά κυκλώματα, π.χ. πολυπλέκτες, αποκωδικοποιητές, αριθμητικά κυκλώματα
  - Ακολουθιακά κυκλώματα: flip-flop, καταχωρητές, μετρητές
- Σχεδίαση πιο πολύπλοκων ψηφιακών συστημάτων:
  - Μηχανές καταστάσεων, μνήμες, επεξεργαστές
- Προσομοίωση ψηφιακών κυκλωμάτων
  - Χρήση εργαλείων προσομοίωσης (HDL simulators)

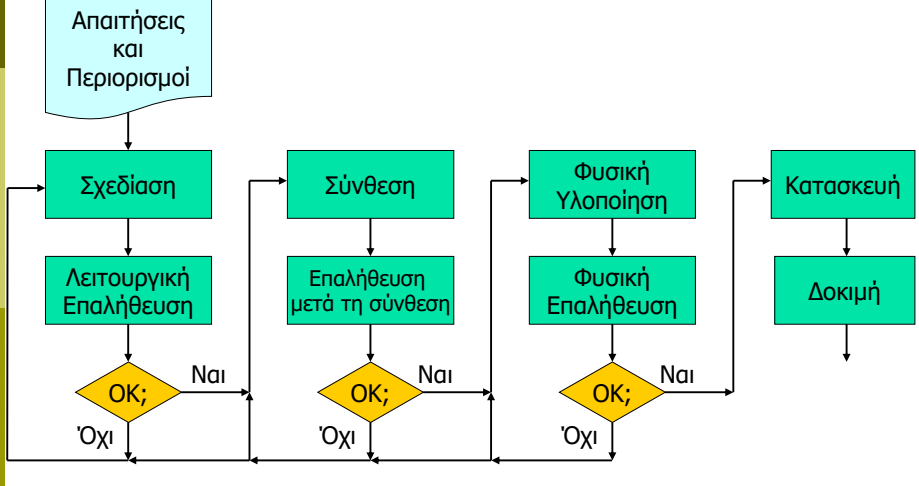
Προαπαιτούμενα

- Γνώσεις από το μάθημα λογικής σχεδίασης ψηφιακών συστημάτων:
  - Λογικές πύλες
  - Συνδυαστικά κυκλώματα:
    - πολυπλέκτες, αποκωδικοποιητές
  - Αριθμητικά κυκλώματα
    - αθροιστές, πολλαπλασιαστές
  - Ακολουθιακά κυκλώματα
    - καταχωρητές, καταχωρητές ολίσθησης, μετρητές
  - Μηχανές καταστάσεων

Εργαστήριο

- Περιβάλλον σχεδίασης, προσομοίωσης και αποσφαλμάτωσης ψηφιακών κυκλωμάτων (σε γλώσσα περιγραφής υλικού VHDL)
- Υλοποίησης των κυκλωμάτων σε εκπαιδευτικές πλατφόρμες FPGA (Field Programmable Gate Arrays)
  - Xilinx Vivado Design Suite
- Εργαστηριακές ασκήσεις:
  - Γνωριμία με το περιβάλλον
  - Γνωριμία με την πλακέτα FPGA
  - Συνδυαστικά κυκλώματα
  - Ακολουθιακά κυκλώματα
  - Μηχανές πεπερασμένων καταστάσεων
  - Μνήμες
- Μπορείτε να κατεβάσατε και να εγκαταστήσετε την student version (free) του εργαλείου: https://www.xilinx.com/products/design-tools/vivado/vivado-webpack.html

Μεθοδολογία σχεδίασης



Σύνθεση (synthesis)

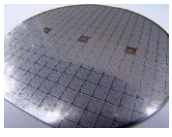
- Συνήθως, η σχεδίαση των συστημάτων γίνεται σε επίπεδο μεταφοράς καταχωρητή (register-transfer-level – RTL)
  - Υψηλότερο επίπεδο αφαίρεσης σε σχέση με τη σχεδίαση με πύλες
- Τα εργαλεία σύνθεσης μεταφράζουν τη σχεδίαση RTL σε ένα κύκλωμα με πύλες που εκτελεί την ίδια λειτουργία
- Στο εργαλείο σύνθεσης πρέπει να καθορίσουμε:
  - Την τεχνολογία υλοποίησης
  - Περιορισμούς σε χρόνο, επιφάνεια, κτλ. (αν υπάρχουν)
- Επαλήθευση μετά τη σύνθεση (post-synthesis verification):
  - Ότι το κύκλωμα που έχει προκύψει από τη σύνθεση ικανοποιεί τους περιορισμούς

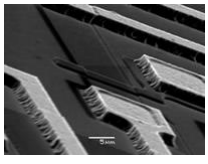
Φυσική υλοποίηση (physical implementation)

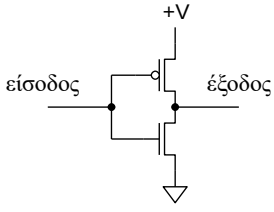
- Δομές υλοποίησης:
  - Application-specific ICs (ASICs)
  - Field-programmable gate arrays (FPGAs)
- Χωροθέτηση (floor-planning)
  - Τοποθετεί τα υποσυστήματα
- Τοποθέτηση (placement)
  - Τοποθετεί τις πύλες μέσα στα υποσυστήματα
- Δρομολόγηση (routing)
  - Συνδέει τις πύλες με αγωγούς
- Φυσική επαλήθευση (physical verification)
  - Το φυσικό κύκλωμα ικανοποιεί ακόμα τους περιορισμούς
  - Καλύτερη εκτίμηση των χρονικών προδιαγραφών

Κατασκευή (manufacturing)

- Τα ολοκληρωμένα κυκλώματα (integrated circuits) κατασκευάζονται στην επιφάνεια ενός πλακιδίου πυριτίου (silicon wafer)
  - Ελάχιστο χαρακτηριστικό μέγεθος (feature size) που μειώνεται σε κάθε τεχνολογική γενιά
  - Τώρα 25nm
  - Νόμος του Moore: αύξηση του αριθμού των τρανζίστορ
  - CMOS: συμπληρωματικά (complementary) MOSFET κυκλώματα

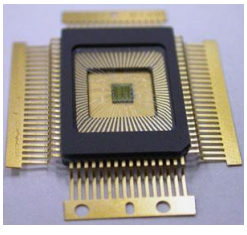


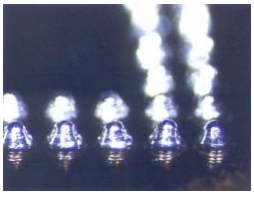




Συσκευασίες ολοκληρωμένων κυκλωμάτων

- Τα ολοκληρωμένα κυκλώματα ενθυλακώνονται σε προστατευτική συσκευασία
  - Εξωτερικοί ακροδέκτες για να συνδεθούν με την πλακέτα κυκλώματος
  - Καλώδια συγκόλλησης ή συνδέσεις flip-chip





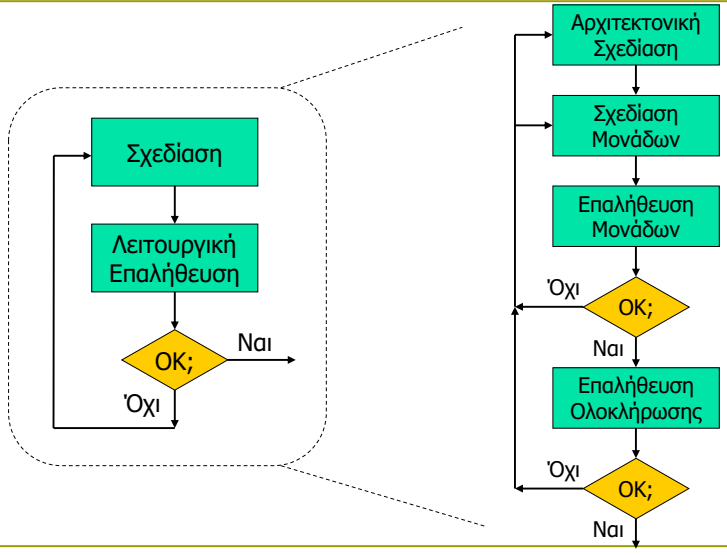
Printed Circuit Boards (PCBs)

- Πλακέτες τυπωμένου κυκλώματος
- Στρώματα μεταλλικής καλωδίωσης μεταξύ στρωμάτων μονωτικού υλικού (fiberglass)
  - Κατασκευάζεται με χρήση φωτολιθογραφίας και εγχάραξης
- Οι αγωγοί διασυνδέουν IC και άλλα στοιχεία
  - Εξωτερικές συνδέσεις σε άλλα στοιχεία του συστήματος

Ιεραρχική σχεδίαση

- Τα κυκλώματα είναι αρκετά πολύπλοκα για να σχεδιάσουμε όλες τις λεπτομέρειες με τη μία
- Σχεδιάζουμε υποσυστήματα για απλές λειτουργίες
- Συνθέτουμε υποσυστήματα για να σχηματίσουμε το σύστημα
  - Αντιμετωπίζουμε τα υποκυκλώματα ως «μαύρα κουτιά»
  - Επαληθεύουμε ανεξάρτητα, και έπειτα επαληθεύουμε την ολοκλήρωσή τους
- Σχεδίαση top-down (από πάνω προς τα κάτω) ή bottom-up (από κάτω προς τα πάνω)

Ιεραρχική σχεδίαση



Γλώσσες περιγραφής υλικού

- Hardware Description Language (HDL)
  - Μια γλώσσα για την μοντελοποίηση της συμπεριφοράς και της δομής των ψηφιακών συστημάτων
- Electronic Design Automation (EDA) using HDL - Αυτοματοποίηση ηλεκτρονικής σχεδίασης: σχεδίαση ηλεκτρονικών κυκλωμάτων με χρήση εργαλείων CAD (computer-aided design)
  - Εισαγωγή σχεδίασης (design entry)
    - κώδικας αντί για σχηματικά διαγράμματα
  - Επαλήθευση (verification)
    - προσομοίωση του κώδικα
  - Σύνθεση (synthesis)
    - αυτόματη παραγωγή των κυκλωμάτων

Πλεονεκτήματα των HDL

- Υπερτερούν από τα σχηματικά διαγράμματα:
  - Η μοντελοποίηση του συστήματος μπορεί να γίνει σε όλα τα επίπεδα (από τα υψηλότερα ως τα χαμηλότερα)
  - Η περιγραφή σε HDL είναι συνήθως (?) πιο κατανοητή από ένα σχηματικό διάγραμμα
  - Η περιγραφή σε HDL είναι ανεξάρτητη από τις βιβλιοθήκες σχεδίασης (design libraries) και τα CAD εργαλεία
- Υπερτερούν από τις γλώσσες προγραμματισμού:
  - Παρέχουν δομές που περιγράφουν καλύτερα το υλικό
  - Παράλληλη εκτέλεση εντολών αντί για ακολουθιακή
  - Παρέχουν δυνατότητα για περιγραφή χρονισμών

Γλώσσες περιγραφής υλικού: VHDL

- **VHDL**: **V**HSIC **H**ardware **D**escription **L**anguage
  - VHSIC: **V**ery **H**igh-**S**peed **I**ntegrated **C**ircuits
- Ιστορική αναδρομή:
  - Ξεκίνησε το 1981 από το Υπουργείο Άμυνας των ΗΠΑ ως γλώσσα περιγραφής ολοκληρωμένων κυκλωμάτων
  - Οι εταιρείες IBM, Texas Instruments, Intermetrics ανάπτυξαν και κυκλοφόρησαν την 1η έκδοση το 1985
- Πρότυπο από τον οργανισμό IEEE
  - IEEE Standard 1076-1987 (VHDL-87)
  - IEEE Standard 1076-1993 (VHDL-93)
  - IEEE Standard 1076a (VHDL-2000)
- Πιο διαδεδομένη στην Ευρώπη

Γλώσσες περιγραφής υλικού: Verilog

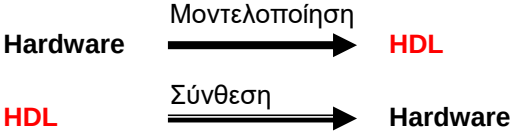
- Ιστορική αναδρομή:
  - Αναπτύχθηκε ως γλώσσα μοντελοποίησης υλικού από την εταιρεία Gateway Design Automation το 1984 για ιδιωτική χρήση
  - Η εταιρεία Cadence Design Systems αγόρασε την Gateway το 1990
  - Η εταιρεία Cadence είναι υπεύθυνη για την προώθηση της Verilog ως γλώσσα μοντελοποίησης & προσομοίωσης
  - Η εταιρεία Synopsys είναι υπεύθυνη για την προώθηση της Verilog ως γλώσσα σύνθεσης
- Πρότυπο από τον οργανισμό IEEE το 1995
- Πιο διαδεδομένη στην Αμερική

HDL: μοντελοποίηση & προσομοίωση

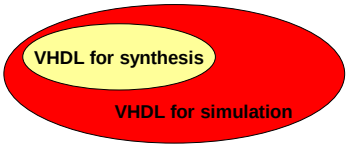
- Αρχικά οι γλώσσες περιγραφής υλικού (HDL) σχεδιάστηκαν για τη μοντελοποίηση και τη προσομοίωση των συστημάτων
  - Η ιδέα ήταν να εισάγουν δομές στην γλώσσα που να επιτρέπουν τη μοντελοποίηση και τη προσομοίωση του υλικού στα υψηλότερα επίπεδα αφαίρεσης
- Χαρακτηριστικά μοντελοποίησης των HDLs:
  - παράλληλη εκτέλεση
  - ιεραρχική σχεδίαση
  - περιγραφή χρονισμών
  - περιγραφή ακολουθίας γεγονότων
  - περιγραφή σύγχρονης/ασύγχρονης συμπεριφοράς

HDL: μοντελοποίηση & σύνθεση

- Αργότερα όμως αναπτύχθηκαν εργαλεία για σύνθεση ...
- ... τα εργαλεία σύνθεσης όμως δεν μπορούν να υποστηρίξουν όλες τις δομές των HDLs



- Ένα υποσύνολο των HDL είναι συνθέσιμο



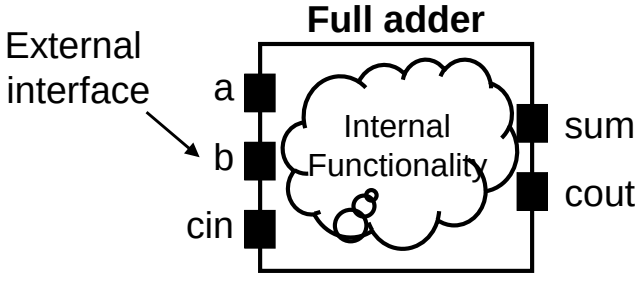
Πώς να ΜΗΝ γράφετε κώδικα VHDL …

- Επειδή οι HDLs έχουν τις ρίζες τους σε γλώσσες προγραμματισμού (η VHDL στην Ada και η Verilog στην C) είναι εύκολες στην εκμάθηση …
- … αλλά δύσκολες στη σωστή χρήση τους !
- Οι αρχάριοι σχεδιαστές τείνουν να γράφουν κώδικα VHDL που μοιάζει με τα προγράμματα υπολογιστών (…πολλές μεταβλητές και πολλούς βρόχους …)
- Για αυτό:
  - Μη γράφετε VHDL όπως θα γράφατε ένα πρόγραμμα
  - Θυμηθείτε τις δυνατότητες που σας δίνει η VHDL (π.χ. παράλληλη εκτέλεση, περιγραφή χρονισμών, περιγραφή ακολουθίας γεγονότων)
  - Να έχετε πάντα στο μυαλό σας τι κύκλωμα αντιστοιχεί στον κώδικα VHDL που γράφετε

Έννοιες μοντελοποίησης της VHDL

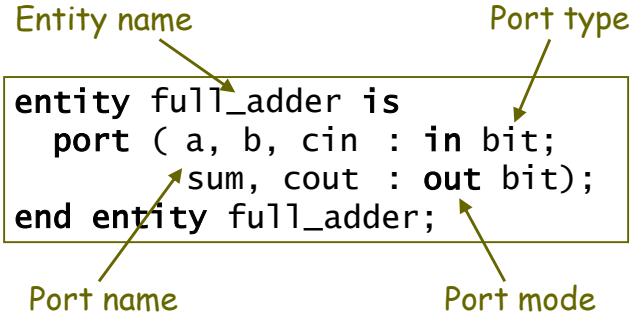
- Διασύνδεση (interface)
- Συμπεριφορά (behavior)
- Δομή (structure)
- Μοντέλα δοκιμής (test benches)

Σχεδιαστική μονάδα στη VHDL



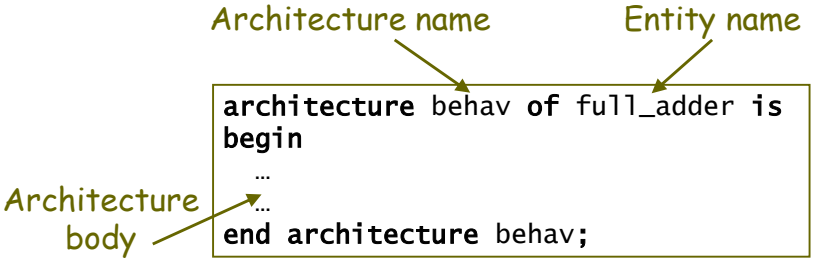
Οντότητα (entity)

- Περιγράφει την εξωτερική διασύνδεση (external interface) της σχεδιαστικής μονάδας



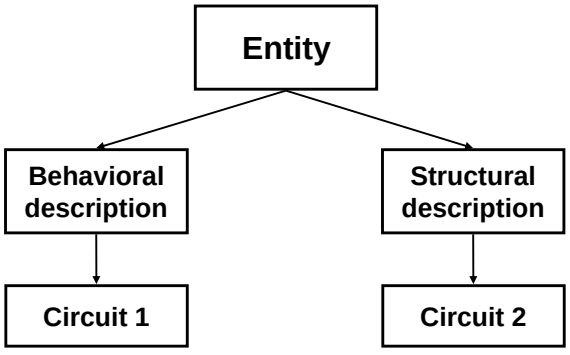
Αρχιτεκτονική (architecture)

- Περιγράφει την εσωτερική συνάρτηση (internal functionality) της σχεδιαστικής μονάδας



Οντότητα και αρχιτεκτονικές

- Υπάρχουν διαφορετικές αρχιτεκτονικές για να περιγράψουν την συνάρτηση μίας οντότητας



Περιγραφή συμπεριφοράς (behavioral description)

```
architecture behav of full_adder is
begin

p: process (a,b,cin) is
  begin
    if a = '1' then
      cout <= b or cin;
      sum <= b xnor cin;
    else
      cout <= b and cin;
      sum <= b xor cin;
  end if;
end process;

end architecture behav;
```

Περιγραφή δομής (structural description)

```
entity half_adder is
  port (a,b      : in bit;
        sum,cout : out bit);
end entity half_adder;

architecture behav of half_adder is

begin

  sum <= a xor b;
  cout <= a and b;

end architecture behav;
```


Περιγραφή δομής (structural description)

```
architecture struct of full_adder is

signal sum1,cout1,cout2: bit;

begin

  ha1: entity work.half_adder(behav)
       port map(a,b,sum1,cout1);

  ha2: entity work.half_adder(behav)
       port map(cin,sum1,sum,cout2);

  cout <= cout1 or cout2;

end architecture struct;
```

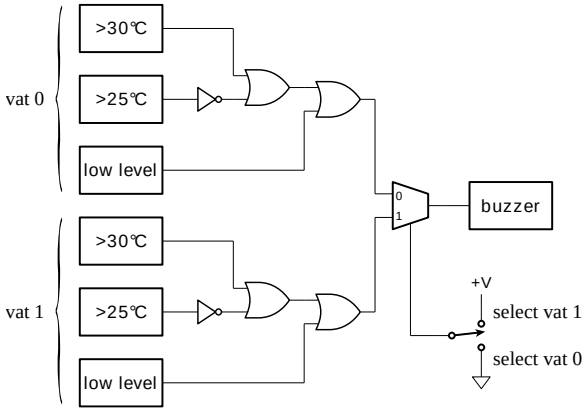
Ποια περιγραφή είναι καλύτερη;

- Πολυπλοκότητα της σχεδίασης:
  - Η περιγραφή συμπεριφοράς προτιμάται για να περιγράψει μία πολύπλοκη συνάρτηση
  - Η περιγραφή δομής προτιμάται για να περιγράψει μία ιεραρχική σχεδίαση (επαναχρησιμοποίηση μονάδων)
- Απόδοση της σχεδίασης: μέγεθος, καθυστέρηση, κατανάλωση
  - Εξαρτάται από το εργαλείο σύνθεσης
  - Εξαρτάται από την εμπειρία του σχεδιαστή
- Όλοι οι τύποι περιγραφής μπορούν να συνδυαστούν σε μία σχεδίαση

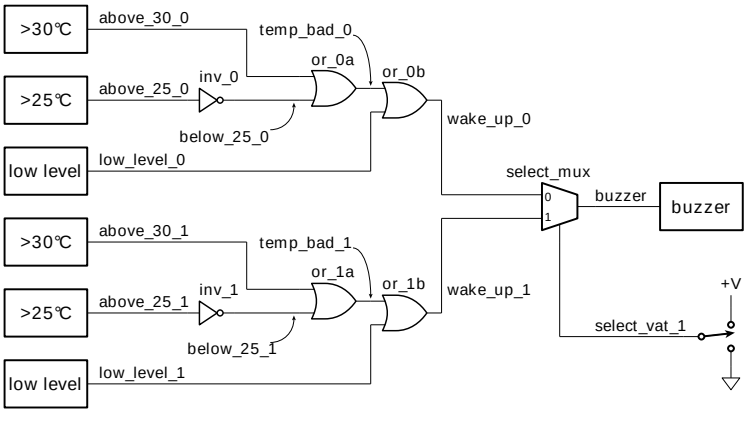
Παράδειγμα: συνδυαστικό κύκλωμα

- Δύο δοχεία επεξεργασίας υγρών:
  - Διακόπτης επιλογής δοχείου
  - Σωστή θερμοκρασία (μεταξύ 25°C και 30°C)
  - Αισθητήρες θερμοκρασίας
  - Αισθητήρες χαμηλής στάθμης
- Κύκλωμα ελέγχου που ενεργοποιεί ένα κουδούνι (συναγερμού)
  - Όταν η θερμοκρασία είναι πολύ υψηλή ή πολύ χαμηλή ή
  - Όταν η στάθμη του υγρού είναι πολύ χαμηλή

Σχεδίαση κυκλώματος



Σχεδίαση κυκλώματος



Δήλωση οντότητας

```
entity vat_buzzer is
  port ( above_25_0, above_30_0,
         low_level_0   : in bit;
         above_25_1, above_30_1,
         low_level_1   : in bit;
         select_vat_1  : in bit;
         buzzer        : out bit );
end entity vat_buzzer;
```


Αρχιτεκτονική δομής

```
library dld; use dld.gates.all;
architecture struct of vat_buzzer is
  signal below_25_0, temp_bad_0, wake_up_0 : bit;
  signal below_25_1, temp_bad_1, wake_up_1 : bit;
begin
  -- components for vat 0
  inv_0 : inv (above_25_0, below_25_0);
  or_0a : or2 (above_30_0, below_25_0, temp_bad_0);
  or_0b : or2 (temp_bad_0, low_level_0, wake_up_0);
  -- components for vat 1
  inv_1 : inv (above_25_1, below_25_1);
  or_1a : or2 (above_30_1, below_25_1, temp_bad_1);
  or_1b : or2 (temp_bad_1, low_level_1, wake_up_1);
  select_mux : mux2 (wake_up_0, wake_up_1,
select_vat_1, buzzer);
end architecture struct;
```

Αρχιτεκτονική συμπεριφοράς

```
architecture behavior of vat_buzzer is
begin
  buzzer <=
    low_level_1 or (above_30_1 or not above_25_1)
        when select_vat_1 = '1' else
    low_level_0 or (above_30_0 or not above_25_0);
end architecture behavior;
```


Test Benches

- Επαλήθευση της σχεδίασης με προσομοίωση
- Τι είναι ένα test bench;
  - Ένα VHDL μοντέλο χωρίς εισόδους/εξόδους που περιέχει ένα στιγμιότυπο της μονάδας υπό δοκιμή
- Τι κάνει ένα test bench;
  - Εφαρμόζει ακολουθίες τιμών δοκιμής στις εισόδους της μονάδας
  - Παρακολουθεί τις τιμές στις εξόδους της μονάδας

Παράδειγμα test bench

```
entity test_vat_buzzer is
end entity test_vat_buzzer;

architecture testbench of test_vat_buzzer is

   signal Above_25_0, Above_30_0, Low_level_0  : bit;
          Above_25_1, Above_30_1, Low_level_1  : bit;
          Select_vat_1, Buzzer                 : bit;

begin

   dut : entity work.vat_buzzer(behav)
      port map (Above_25_0, Above_30_0, Low_level_0,
                Above_25_1, Above_30_1, Low_level_1,
                Select_vat_1, Buzzer);

   stimulus : process is
   begin
      …
```


Παράδειγμα test bench (συν.)

```
      --Temperature between 25°C and 30°C, Levels OK
      Above_25_0 <= '1'; Above_30_0 <= '0'; Low_level_0 <= '0';
      Above_25_1 <= '1'; Above_30_1 <= '0'; Low_level_1 <= '0';
      Select_vat_1 <= '0';  wait for 20 ns;
      --Vat0: temperature above 30°C
      Above_30_0 <= '1'; wait for 20 ns;
      --Vat0: temperature below 25°C
      Above_30_0 <= '0'; Above_25_0 <= '0'; wait for 20 ns;
      --Temperature between 25°C and 30°C, Levels OK
      Above_25_0 <= '1'; wait for 20 ns;
      --Vat1: low level
      Low_level_1 <= '1';
      Select_vat_1 <= '1'; wait for 20 ns;
      …

      wait;
   end process stimulus;
end architecture testbench;
```