

1^ο Εργαστήριο Λογικής Σχεδίασης

1^η Διάλεξη

Εισαγωγή

Βασιλόπουλος Διονύσης

Ε.ΔΙ.Π. Τμήματος Πληροφορικής & Τηλεπικοινωνιών

Αντικείμενο Μαθήματος

Εργαστήριο Λογικής Σχεδίασης – Τι είναι

- Εφαρμοσμένο μέρος του μαθήματος της Λογικής Σχεδίασης (ΛΣ)
 - Συνοπτική επανάληψη θεωρίας ΛΣ
- Προγραμματισμός στη γλώσσα VHDL
- Περιγράφουμε (σε VHDL) ένα Σύστημα Λογικού Κυκλώματος/Ψηφιακό Κύκλωμα
- Τι κάνει το Λογικό Κύκλωμα; Δίνει λύση σε μία πραγματική ανάγκη
- Με τι προγραμματίζουμε; Με το εργαλείο Vivado
- Τι προγραμματίζουμε; Hardware (κάρτες FPGA)

Αντικείμενο Μαθήματος

Πραγματικές Ανάγκες

Ψηφιακά Συστήματα



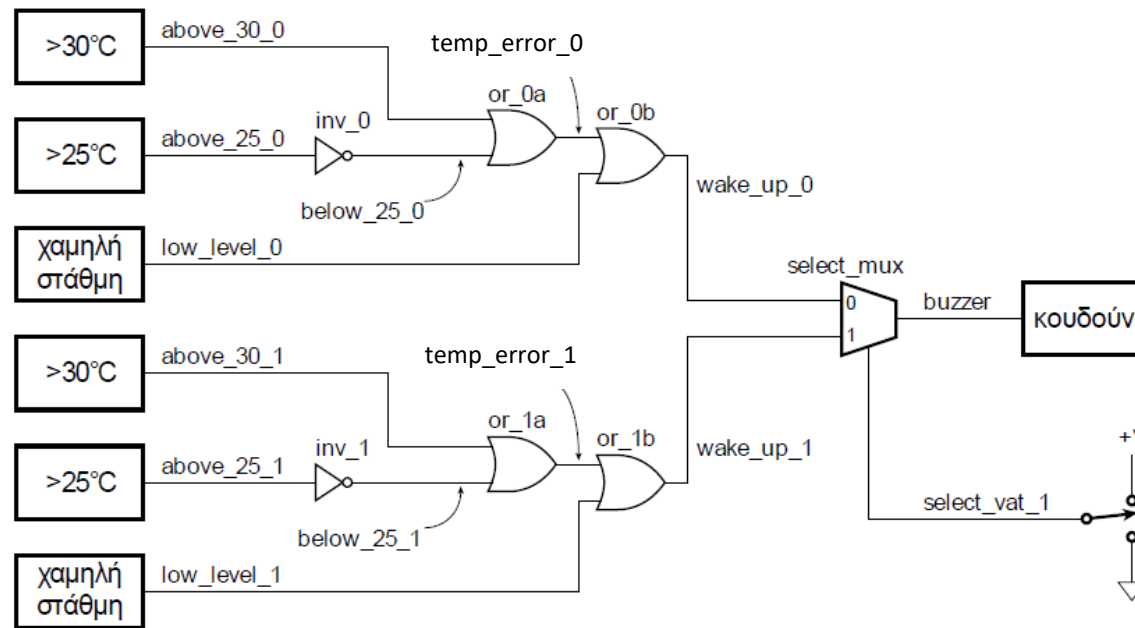
Αντικείμενο Μαθήματος

Εργαστήριο Λογικής Σχεδίασης – Περιεχόμενο μαθήματος

- Βασικές Αρχές της **Γλώσσας Προγραμματισμού VHDL**
- **Ανάπτυξη απλών εφαρμογών** για κατανόηση του μοντέλου της VHDL
- **Χρήση του εργαλείου VIVADO** για την ανάπτυξη εφαρμογών
- **Βασικά στοιχεία της θεωρίας Λογικής Σχεδίασης** (δυαδικό σύστημα αρίθμησης, λογικές πύλες, συνδυαστικά κυκλώματα, ακολουθιακά κυκλώματα, μνήμες)
 - ΔΕΝ θα γίνει ανάπτυξη στα θέματα της θεωρίας της Λογικής Σχεδίασης, παρά μόνο συνοπτικά και ΜΟΝΟ όσα χρειάζονται για την ανάπτυξη των εφαρμογών σε VHDL.
- **Προγραμματισμός κάρτας FPGA** (μέσω VHDL+VIVADO) σε πραγματικές συνθήκες (εργαστήριο)

Αντικείμενο Μαθήματος

Λογικά κυκλώματα



Αντικείμενο Μαθήματος

Προγραμματισμός Υλικού σε VHDL

```
library ieee; use ieee.std_logic_1164.all;
entity vat_buzzer is
  port ( above_25_0, above_30_0,
         low_level_0  : in std_logic;
         above_25_1, above_30_1,
         low_level_1  : in std_logic;
         select_vat_1 : in std_logic;
         buzzer       : out std_logic );
end entity vat_buzzer;
```

Χρήση πρότυπης βιβλιοθήκης

package

Entity name

Port list

Port type

Port mode

Port name

Αντικείμενο Μαθήματος

Εργαστήριο Λογικής Σχεδίασης – Οργάνωση μαθήματος 1/11

- Διδάσκων: Ε.ΔΙ.Π. Διονύσης Βασιλόπουλος (denis@di.uoa.gr) – Γραφείο: Α33
- Ώρες γραφείου: Τρίτη 15:00-16:00 και Πέμπτη 13:00 – 15:00, κατόπιν επικοινωνίας
- 3 διαλέξεις εισαγωγικές και 5 σετ Θεωρίας/εργαστηρίου (θεωρία/παράδοση-χρήση εργαλείου Vivado, επίλυση άσκησης-προγραμματισμός κάρτας).
- Σύνολο 8 διαλέξεις στο Αμφιθέατρο Α2 και 5 μαθήματα στο Εργαστήριο Ψηφιακής Σχεδίασης & ΗΥ Υψηλών Επιδόσεων (Αναγνωστήριο)

Αντικείμενο Μαθήματος

Εργαστήριο Λογικής Σχεδίασης – Οργάνωση μαθήματος 2/11

Βαθμολογία Μαθήματος

- Εξάμηνα φοίτησης: 1, 2, 3, 4 (Α)

- ✓ Τρεις (3) εργασίες: 6 μονάδες του τελικού βαθμού. Βάρη εργασιών 20%, 40%, 40%.

Θα πρέπει να βαθμολογηθείτε με τουλάχιστον 50/100 στην 3^η εργασία.

- ✓ Γραπτή εξέταση: 3 μονάδες του τελικού βαθμού.

Αν η γραπτή εξέταση έχει απόκλιση πάνω από 30 μονάδες (στις 100) από τη βαθμολογία των ασκήσεων, τότε ως βαθμολογία των ασκήσεων θεωρείται η βαθμολογία της γραπτής εξέτασης.

- ✓ Παρουσίες: 1 μονάδα του τελικού βαθμού.

Θα πρέπει να έχετε παρουσία σε τουλάχιστον 4 διαλέξεις και 3 εργαστήρια για να βαθμολογηθείτε στο μάθημα.

Αντικείμενο Μαθήματος

Εργαστήριο Λογικής Σχεδίασης – Οργάνωση μαθήματος 3/11

Βαθμολογία Μαθήματος

- Εξάμηνα φοίτησης: 1, 2, 3, 4 (A)
Παράδειγμα A

Βαθμός σε Εργασία A 80%, σε Εργασία B 90%, σε Εργασία Γ 100%, Παρουσία σε 4 διαλέξεις και 4 εργαστήρια, Βαθμός Γραπτής εξέτασης 90%.

Βαθμός από Εργασίες: $20\% \cdot 80\% + 40\% \cdot 90\% + 40\% \cdot 100\% = 16\% + 36\% + 40\% = 92\%$ (A)

Βαθμός από Γραπτή Εξέταση: 90% (B)

Βαθμός από Παρουσίες: $(50\% \cdot 4/8 = 25\%$ από διαλέξεις, $50\% \cdot 4/5 = 40\%$ από εργαστήρια) 75% (Γ)

Βαθμός τελικός: $(A) \cdot 6 + (B) \cdot 3 + (\Gamma) \cdot 1 = 92\% \cdot 6 + 90\% \cdot 3 + 75\% \cdot 1 = 5.52 + 2.7 + 0.75 \Rightarrow 8.97 \Rightarrow 9$

Αντικείμενο Μαθήματος

Εργαστήριο Λογικής Σχεδίασης – Οργάνωση μαθήματος 4/11

Βαθμολογία Μαθήματος

- Εξάμηνα φοίτησης: 1, 2, 3, 4 (Α)
Παράδειγμα Β

Βαθμός σε Εργασία Α 80%, σε Εργασία Β 90%, σε Εργασία Γ 100%, Παρουσία σε 4 διαλέξεις και 4 εργαστήρια, Βαθμός Γραπτής εξέτασης 50%.

Βαθμός από Εργασίες: $20\% \cdot 80\% + 40\% \cdot 90\% + 40\% \cdot 100\% = 16\% + 36\% + 40\% = 92\%$ (Α). Επειδή η Γραπτή Εξέταση έχει διαφορά από 30 μονάδες σε σχέση με τις εργασίες, ο βαθμός των Εργασιών γίνεται ίσος με αυτόν της Γραπτής εξέτασης. Άρα **50% (Α)**

Βαθμός από Γραπτή Εξέταση: **50% (Β)**

Βαθμός από Παρουσίες: $(50\% \cdot 4/8 = 25\%$ από διαλέξεις, $50\% \cdot 4/5 = 40\%$ από εργαστήρια) **75% (Γ)**

Βαθμός τελικός: $(Α) \cdot 6 + (Β) \cdot 3 + (Γ) \cdot 1 = 50\% \cdot 6 + 50\% \cdot 3 + 75\% \cdot 1 = 3 + 1,5 + 0,75 \Rightarrow 5,25 \Rightarrow 5$

Αντικείμενο Μαθήματος

Εργαστήριο Λογικής Σχεδίασης – Οργάνωση μαθήματος 5/11

Βαθμολογία Μαθήματος

- Εξάμηνα φοίτησης: 1, 2, 3, 4 (Α)
Παράδειγμα Γ

Βαθμός σε Εργασία Α 80%, σε Εργασία Β 90%, σε **Εργασία Γ 40%**, Παρουσία σε 4 διαλέξεις και 4 εργαστήρια, Βαθμός Γραπτής εξέτασης 50%.

Ο φοιτητής δεν περνάει το μάθημα γιατί έχει στην Εργασία Γ 40% (ελάχιστο 50%)

Αντικείμενο Μαθήματος

Εργαστήριο Λογικής Σχεδίασης – Οργάνωση μαθήματος 6/11

Βαθμολογία Μαθήματος

- **Εξάμηνα φοίτησης: 5, 6, 7, 8 (B)**
 - ✓ Τρεις (3) **εργασίες**: 6 μονάδες του τελικού βαθμού. Βάρη εργασιών 20%, 40%, 40%.
Θα πρέπει να βαθμολογηθείτε με τουλάχιστον 50/100 στην 3^η εργασία.
 - ✓ **Γραπτή εξέταση**: 4 μονάδες του τελικού βαθμού.
Αν η γραπτή εξέταση έχει απόκλιση πάνω από 30 μονάδες (στις 100) από τη βαθμολογία των ασκήσεων, τότε ως βαθμολογία των ασκήσεων θεωρείται η βαθμολογία της γραπτής εξέτασης.

ή

Να βαθμολογηθούν όπως οι φοιτητές των 2 πρώτων ετών (περίπτωση Α)

(όποια τους συμφέρει βαθμολογικά)

Αντικείμενο Μαθήματος

Εργαστήριο Λογικής Σχεδίασης – Οργάνωση μαθήματος 7/11

Βαθμολογία Μαθήματος

- Εξάμηνα φοίτησης: Επί πτυχίω (εξάμηνο φοίτησης ≥ 9) (Γ)

✓ Βιβλιογραφική Εργασία

ή

Να βαθμολογηθούν όπως οι περιπτώσεις Α ή Β (όποια τους συμφέρει βαθμολογικά)

Αντικείμενο Μαθήματος

Εργαστήριο Λογικής Σχεδίασης – Οργάνωση μαθήματος 8/11

Διαδικαστικά Βαθμολογίας

- Η βιβλιογραφική εργασία για τους επί πτυχίω φοιτητές μπορεί να δοθεί και στην επαναληπτική εξεταστική Σεπτεμβρίου.
- Η γραπτή εξέταση γίνεται και στην επαναληπτική εξεταστική Σεπτεμβρίου.
- Οι επί πτυχίω φοιτητές θα πρέπει να δηλώσουν την πρόθεσή τους για βιβλιογραφική εργασία μέχρι τις 31/10/26 επιλέγοντας εργασία μέσω του eclass.
- Η γραπτή (online) **εξέταση VHDL** θα είναι διάρκειας 1 ώρας και 30 λεπτών μετά από τις διακοπές των Χριστουγέννων (11/1/2026).
- Σε κάθε περίπτωση **αν κριθεί απαραίτητο** μπορεί να υπάρχουν μεμονωμένες προφορικές εξετάσεις για επιβεβαίωση, ή όχι, του τελικού βαθμού.
- Απαγορεύεται η χρήση εργαλείων AI για την επίλυση των εργασιών.
- **ΔΕΝ βαθμολογείται μόνο η κατανόηση μιας λύσης αλλά κυρίως να μπορείτε να καταλήξετε εσείς στη λύση.**

Αντικείμενο Μαθήματος

Εργαστήριο Λογικής Σχεδίασης – Οργάνωση μαθήματος 9/11

Κρίσιμες ημερομηνίες (δεν θα δοθεί παράταση στις εργασίες)

- **9/11/2026 (Δευτέρα):** Τελική ημερομηνία υποβολής 1^{ης} εργασίας
 - **1/12/2026 (Τρίτη):** Τελική ημερομηνία υποβολής 2^{ης} εργασίας
 - **7/1/2027 (Πέμπτη):** Τελική ημερομηνία υποβολής 3^{ης} εργασίας (για Χειμερινή Εξεταστική) ή
 - **29/8/2027 (Κυριακή):** Τελική ημερομηνία υποβολής 3^{ης} εργασίας (για Εξεταστική Σεπτεμβρίου)
-
- **11/1/2027 (Δευτέρα):** Η εξέταση **VHDL** διάρκειας 1 ώρας και 30 λεπτών (για Χειμερινή Εξεταστική) ή
 - **1/10/2027 (Δευτέρα):** Η εξέταση **VHDL** διάρκειας 1 ώρας και 30 λεπτών (για Εξεταστική Σεπτεμβρίου)

Αντικείμενο Μαθήματος

Εργαστήριο Λογικής Σχεδίασης – Οργάνωση μαθήματος 10/11

<https://eclass.uoa.gr/> (eclass)

- Ανέβασμα διαφανειών/υλικού και ασκήσεων
- Επικοινωνία μέσω eclass
- Παράδοση εργασιών
- Παρουσίες (**eclass: Ερωτηματολόγιο ή email προς denis@di.uoa.gr**)

Εργαλείο λογισμικού για το μάθημα: VIVADO

- Στο eclass θα ανέβει αρχείο που περιγράφει τον τρόπο με τον οποίο θα εγκαταστήσετε το εργαλείο στον υπολογιστή σας (απαραίτητο).
- Υπάρχει έκδοση για Windows και Linux (όχι για Mac)

<https://uniflix.uoa.gr/start/>

- Απευθείας μετάδοση και Καταγραφή μαθημάτων



Αντικείμενο Μαθήματος

Εργαστήριο Λογικής Σχεδίασης – Οργάνωση μαθήματος 11/11

Υλικό για μελέτη

Α.Παρουσιάσεις (**ΒΑΣΙΚΟ ΕΓΧΕΙΡΙΔΙΟ**), κώδικας και άλλα κείμενα που ανεβαίνουν στο eclass

Β.Προτεινόμενη Βιβλιογραφία

- 1. Ψηφιακή Σχεδίαση. Ενσωματωμένα Συστήματα με VHDL,**
Peter J. Ashenden, Επιστημονική Επιμέλεια – Μετάφραση:
Μ. Παράκης, Ν. Κρανίτης, Δ. Γκιζόπουλος,
Εκδόσεις Νέων Τεχνολογιών, 2010
- 2. Ψηφιακή Σχεδίαση και Αρχιτεκτονική Υπολογιστών,** S.L.Harris, D.M.Harris,
Έκδοση Risc-V, 2025
- 3. Σχεδίαση Ψηφιακών Συστημάτων με τη Γλώσσα VHDL,** S. Brown, Z.Vranesic,
Εκδόσεις Τζιόλα, 2024

Ψηφιακά Συστήματα

Ψηφιακή Σχεδίαση

- **Ψηφιακή (Digital):** κυκλώματα που χρησιμοποιούν δύο επίπεδα τάσης (π.χ. 0V, +5V) για αναπαράσταση της πληροφορίας
 - Λογική (Logic): χρήση τιμών αληθείας (0/1, true/false) και κανόνων λογικής (άλγεβρα boole) για την ανάλυση των κυκλωμάτων
- **Σχεδίαση (Design):** ανταποκρίνονται σε λειτουργικές απαιτήσεις ενώ ταυτόχρονα ικανοποιούν περιορισμούς
 - Περιορισμοί: απόδοση, μέγεθος (κόστος), ισχύς, κλπ.

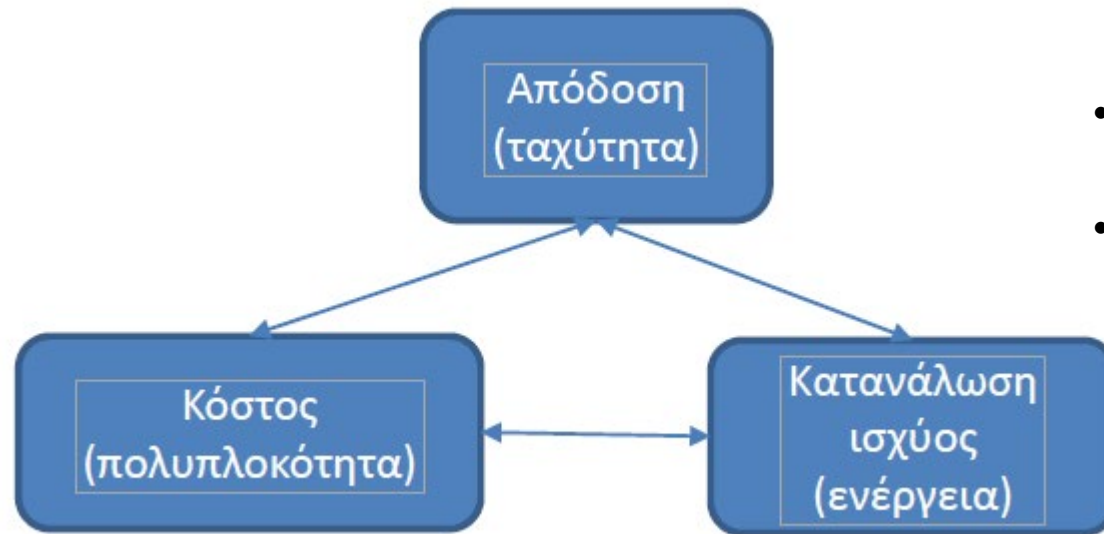
Ψηφιακά Συστήματα

Ψηφιακή Σχεδίαση

- **Δυαδικό Σύστημα Αρίθμησης (0,1)**
- **Λογικοί κανόνες.** Άλγεβρα Boole: ΝΑΙ/ΟΧΙ, ΙΣΧΥΕΙ/ΔΕΝ ΙΣΧΥΕΙ
- **Πύλες.** Στοιχειώδη κυκλώματα, υλοποιούν την άλγεβρα Boole σε επίπεδο υλικού, κάνουμε στοιχειώδεις πράξεις στο δυαδικό σύστημα. Στη βάση τους υπάρχει το Περνάει Ρεύμα/Δεν περνάει ρεύμα ή Υπάρχει Τάση/Γείωση.
- **Συνδυαστικά κυκλώματα** (π.χ. ένα κύκλωμα που προσθέτει δύο δυαδικούς αριθμούς). Έχει εισόδους που παράγουν κάποιες τιμές εξόδων.
- **Ακολουθιακά κυκλώματα.** Συστήματα με μνήμη. Η τιμή της εξόδου μπορεί να εξαρτάται όχι μόνο από τις εισόδους αλλά και από την προηγούμενη τιμή της εξόδου. Συνήθως υπάρχει συγχρονισμός στο πότε αλλάζουν οι τιμές στην είσοδο και στην έξοδο μέσω ενός ρολογιού (Clock) (π.χ. χρονόμετρο).

Ψηφιακά Συστήματα

Ψηφιακή Σχεδίαση



Συμβιβασμοί

- Μπορούμε να βελτιώσουμε το ένα εις βάρος ενός άλλου ή και των δύο άλλων
- Αυτοί οι συμβιβασμοί υπάρχουν σε κάθε επίπεδο στη σχεδίαση του συστήματος

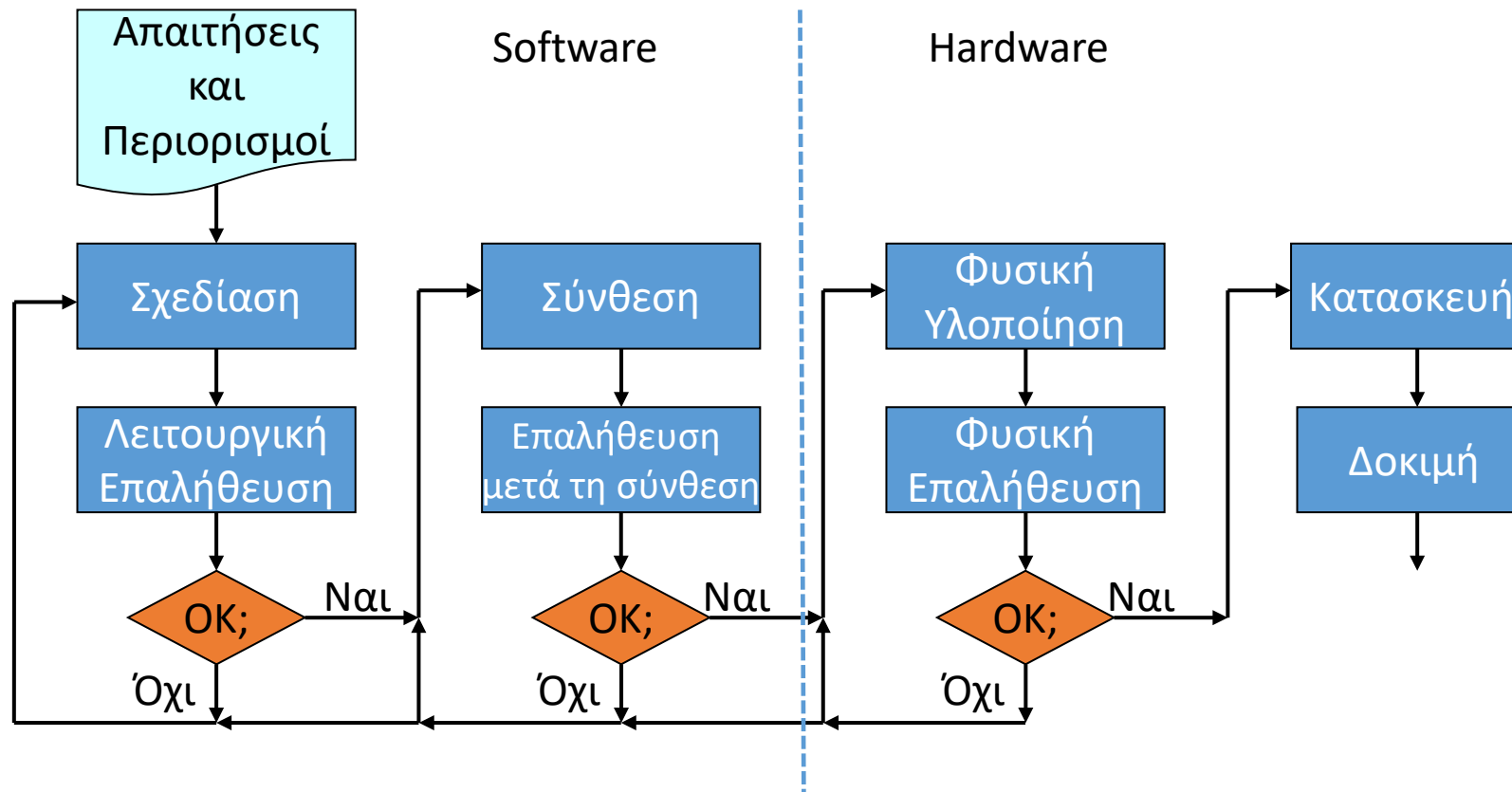
Ψηφιακά Συστήματα

Ψηφιακή Σχεδίαση – Μεθοδολογία Σχεδίασης

- Τα απλά συστήματα μπορούν να σχεδιαστούν από ένα άτομο χρησιμοποιώντας ειδικές μεθόδους
- Τα πραγματικά συστήματα σχεδιάζονται από ομάδες
 - Απαιτούν μια συστηματική μεθοδολογία σχεδίασης
- Καθορίζει
 - Τις εργασίες που αναλαμβάνουμε
 - Την πληροφορία που απαιτείται και παράγεται
 - Τις σχέσεις μεταξύ των εργασιών
 - εξαρτήσεις, καθορισμός των ακολουθιών
 - Τα εργαλεία EDA (Electronic design automation) που χρησιμοποιούνται (αλλιώς και eCAD – electronic Computer Aided Design)

Ψηφιακά Συστήματα

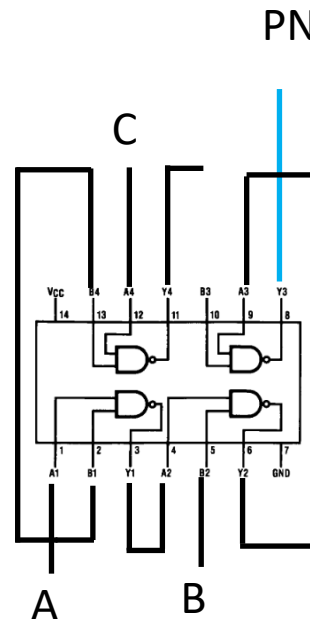
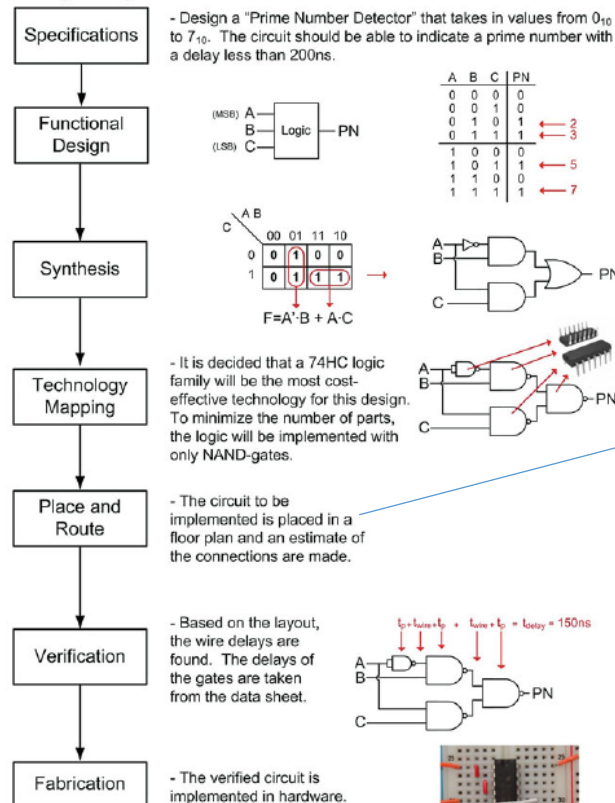
Ψηφιακή Σχεδίαση – Μια απλή μεθοδολογία ανάπτυξης



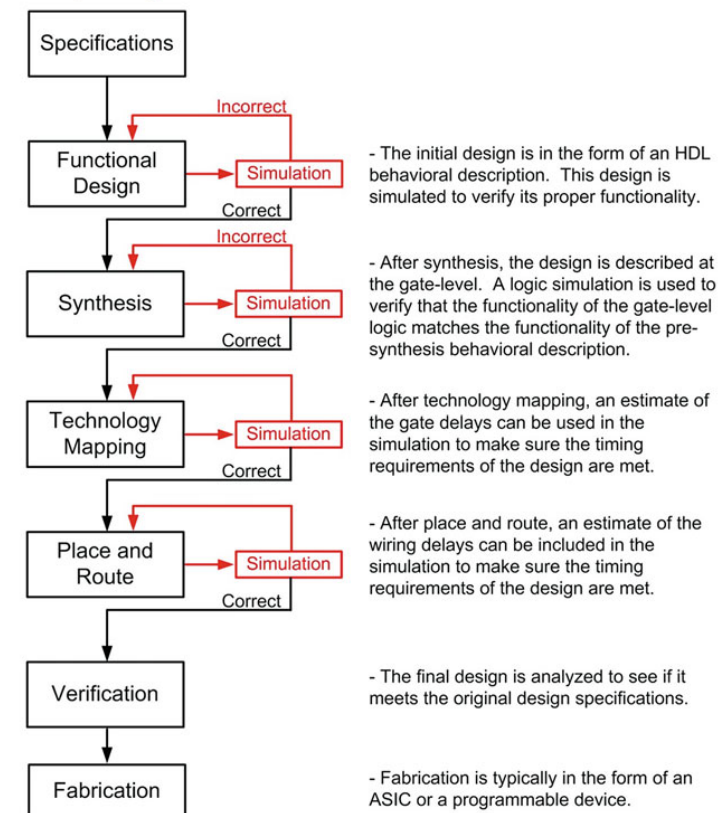
Ψηφιακά Συστήματα

Ψηφιακή Σχεδίαση – Μεθοδολογία ανάπτυξης

Classical Digital Design Flow



Modern Digital Design Flow



<https://www.futurlec.com/Datasheet/74HC/74HC00.pdf>

Εργαστήριο Λογικής Σχεδίασης 2026-27 Δ.Βασιλόπουλος

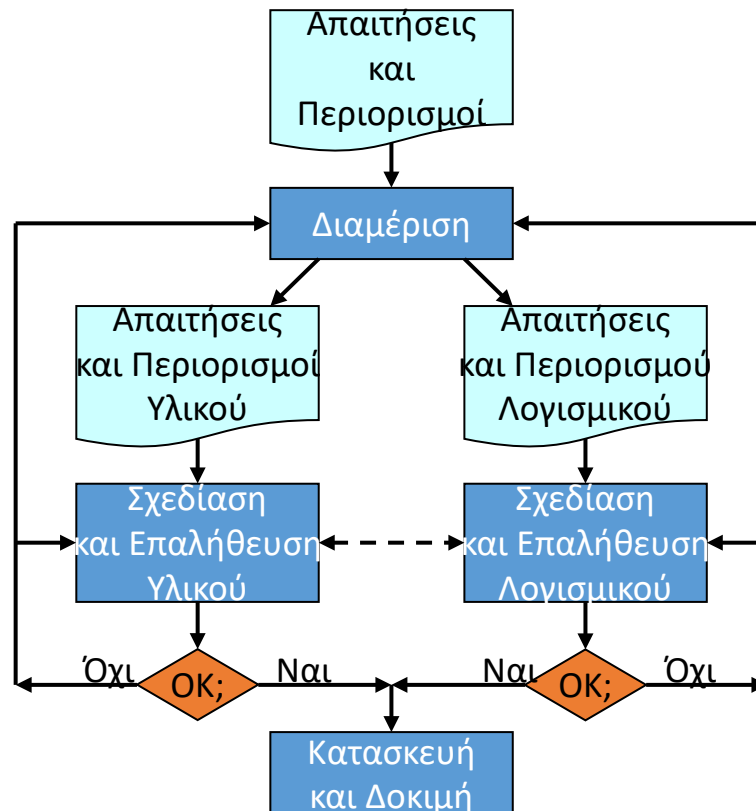
Ψηφιακά Συστήματα

Ψηφιακή Σχεδίαση – Ιεραρχική σχεδίαση

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

Ψηφιακά Συστήματα

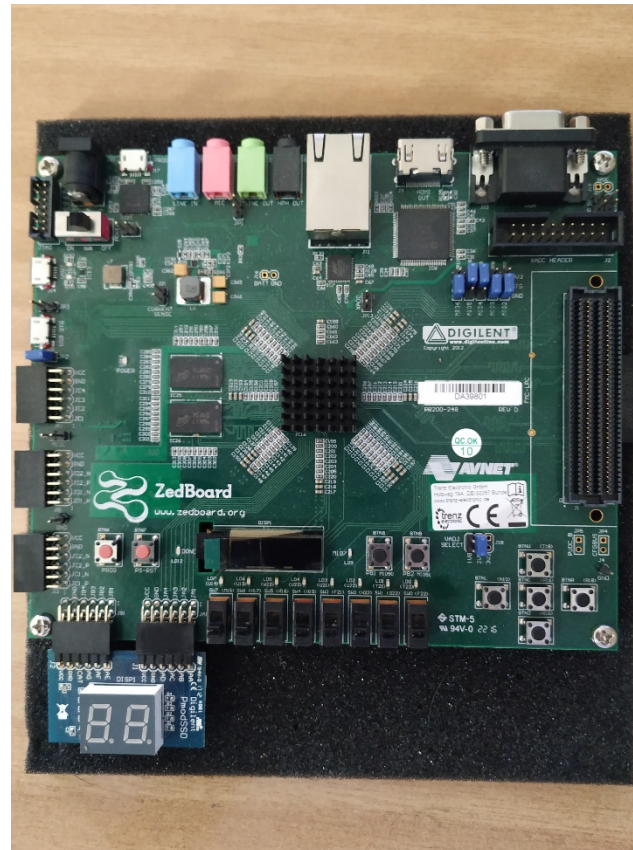
Ψηφιακή Σχεδίαση – Μεθοδολογία Συσχεδίασης



Κάθε ομάδα υλοποιεί ένα ξεχωριστό τμήμα (module) του συστήματος

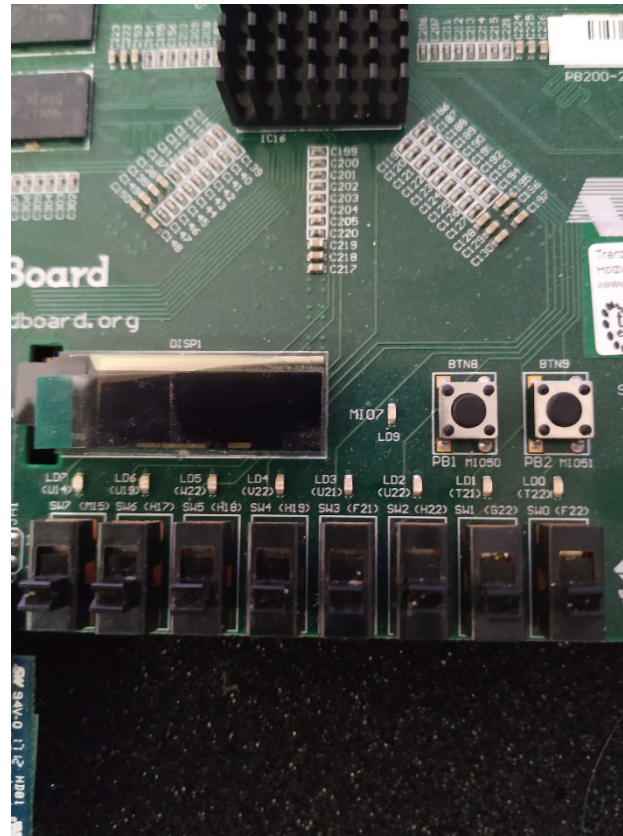
VHDL - VIVADO

Ψηφιακό κύκλωμα – Αναπαράσταση VHDL – FPGA



VHDL - VIVADO

Ψηφιακό κύκλωμα – Αναπαράσταση VHDL – FPGA



Πρωτοετείς

- Εγγραφή στο eclass.uoa.gr (όσοι δεν έχετε κωδικούς από το τμήμα για οποιοδήποτε λόγο, μπορείτε να δείτε το μάθημα στο link <https://eclass.uoa.gr/courses/DI375/>).
- Χρήση του e-mail σας (sdixx00xxx@di.uoa.gr)
- [my-uni.uoa.gr](https://my.uni.uoa.gr) (δηλώσεις/βαθμοί)
- Κωδικοί πρόσβασης στο εργαστήριο linux (σας έχουν δοθεί από τη γραμματεία)
- Στο εργαστήριο Windows ΔΕΝ χρειάζεστε ατομικούς κωδικούς

Πρωτοετείς

- <https://delos365.grnet.gr/>
 - Microsoft office (Word, Excel, Powerpoint,...) σε online και σε desktop έκδοση. Έχετε χώρο και στο onedrive (150GB). Σύνδεση απλά με τα ακαδημαϊκά σας credentials.
- <http://google.com>
 - ΠΡΟΣΟΧΗ: σύνδεση ως sdixxxxxxxxx@uoa.gr και μετά θα σας ζητηθούν τα ακαδημαϊκά σας credentials
 - Εφαρμογές google + 200GB (google drive – Αλλάζει κάθε χρονιά, ανάλογα με τη συνδρομή του ΕΚΠΑ)

Περίληψη

- Εισαγωγή στο μάθημα «Εργαστήριο Λογικής Σχεδίασης»
- Μεθοδολογία σχεδίασης
- Διαβάζετε το κεφάλαιο 1 (εκτός του 1.3) από το βιβλίο του Ashenden.
- Διαβάζετε τις παραγράφους 1.1, 1.2, 1.3, 1.4.1 – 1.4.5. από το βιβλίο των Harris.
- Διαβάζετε τις παραγράφους 1.1, 1.2, 1.3, 2.9 από το βιβλίο των Brown - Vranesic

2^ο Εργαστήριο Λογικής Σχεδίασης

2^η Διάλεξη

Εισαγωγή στη VHDL και το εργαλείο Vivado

Βασιλόπουλος Διονύσης

Ε.ΔΙ.Π. Τμήματος Πληροφορικής & Τηλεπικοινωνιών

Ψηφιακά Συστήματα

Ψηφιακό κύκλωμα – Φυσική υλοποίηση

- Τεχνολογίες υλοποίησης
 - Application-specific ICs (ASICs): Ολοκληρωμένα κυκλώματα εξειδικευμένα για εφαρμογές (δεν προγραμματίζονται)
 - Field-programmable gate arrays (FPGAs): Επιτόπου προγραμματιζόμενοι πίνακες πυλών
 - <https://www.maven-silicon.com/blog/key-differences-between-asic-and-fpga/>
 - <https://numato.com/blog/differences-between-fpga-and-asics/>
- Αντιστοίχιση(mapping): καθορίζει τους πόρους για κάθε υποσύστημα
- Τοποθέτηση (placement): διευθετεί τις πύλες μέσα στα υποσυστήματα
- Δρομολόγηση (routing): ενώνει τις πύλες με αγωγούς (σύρματα/nets)
- Φυσική επαλήθευση (physical verification)
 - Το φυσικό κύκλωμα συνεχίζει να ικανοποιεί τους περιορισμούς
 - Υπολογίζει με μεγαλύτερη ακρίβεια τις εκτιμήσεις των καθυστερήσεων του κυκλώματος

Ψηφιακά Συστήματα

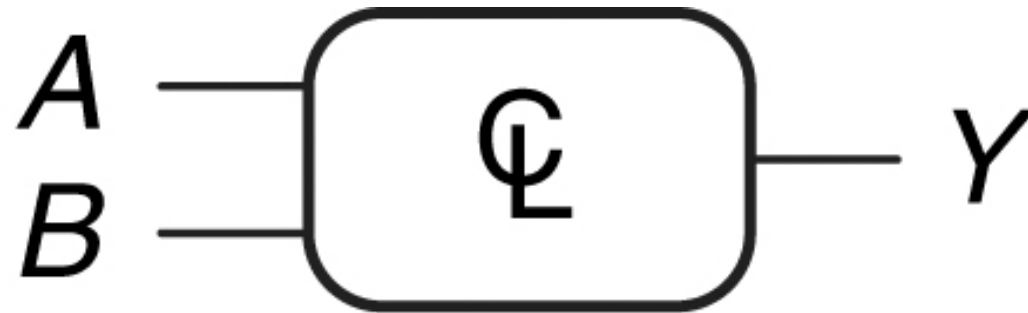
Ενσωματωμένοι επεξεργαστές

- Ένας ενσωματωμένος επεξεργαστής είναι ένας επεξεργαστής κρυμμένος σε μία συσκευή, μαζί και με άλλα ηλεκτρονικά ή ηλεκτρομηχανικά μέρη, σχεδιασμένος για να κάνει μια ή και περισσότερες λειτουργίες πραγματικού χρόνου. Προγραμματίζεται σε ορισμένες λειτουργίες.
- Οι ενσωματωμένοι επεξεργαστές εφαρμόζονται σήμερα εκτενώς σε διάφορες συσκευές ήχου, εικόνας, σε κάμερες, σε ηλεκτρονικά παιχνίδια, σε PDAs, σε υπολογιστές τσέπης, σε περιφερειακά συστήματα γενικής χρήσης όπως modem, κάρτες video, στα τηλέφωνα και στα δίκτυα.
- Κάθε μικροεπεξεργαστής τέτοιου είδους έχει ένα σχετικά μικρό σύνολο εντολών, που εκτελούν λειτουργίες όπως αριθμητικές πράξεις και μεταφορά δεδομένων σε καταχωρητές και μνήμη και συναντώνται σε ενσωματωμένα συστήματα

Ψηφιακά Συστήματα

Κατηγορίες

- **Συνδυαστικά** κυκλώματα: Οι τιμές των εξόδων εξαρτώνται μόνο από τις τιμές των εισόδων του συστήματος

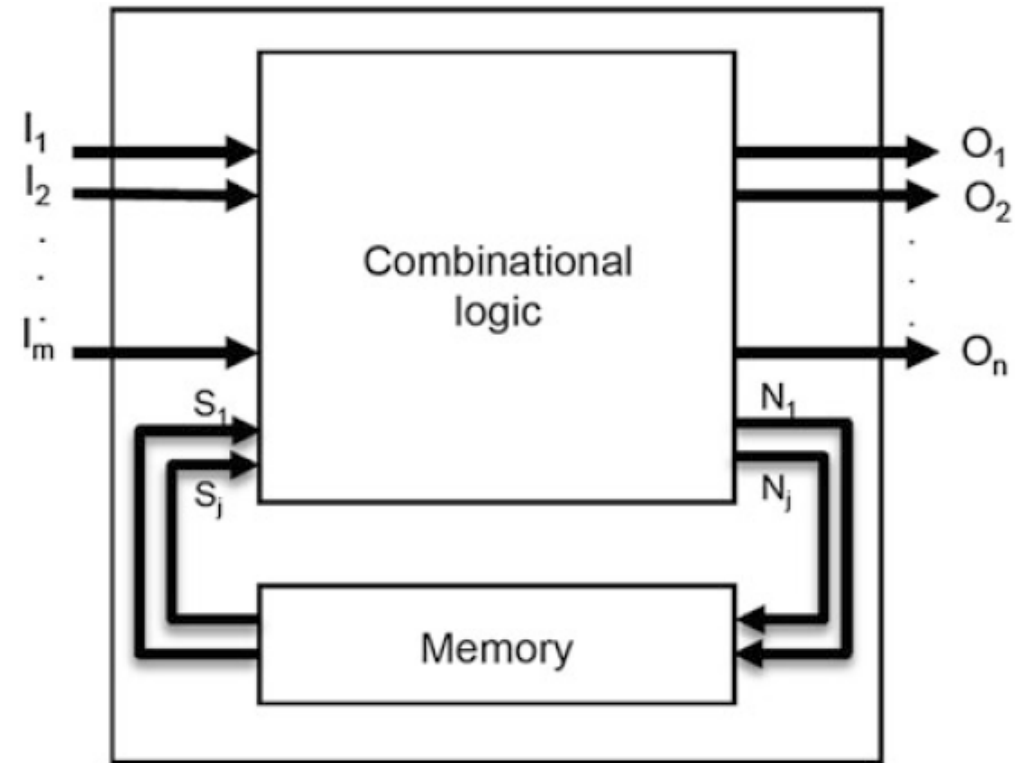


$$Y = F(A, B)$$

Ψηφιακά Συστήματα

Κατηγορίες

- **Ακολουθιακά** κυκλώματα: Οι τιμές των εξόδων εξαρτώνται τόσο από τις τιμές των εισόδων του συστήματος όσο και από τις προηγούμενες τιμές των εξόδων (έχουμε ανάδραση).
 - **Σύγχρονα**: Η συμπεριφορά τους ορίζεται από τις τιμές των εξόδων σε διακριτές στιγμές του χρόνου. Υπάρχει σήμα συγχρονισμού (ρολόι/clock-CLK)
 - **Ασύγχρονα**: Οι τιμές των εξόδων αλλάζουν ανά πάσα στιγμή.



Programming Languages

Έτη δημιουργίας διαφόρων γλωσσών προγραμματισμού

- Fortran, 1957
- Lisp, 1958
- Cobol, 1960
- Simula, 1962
- Logo, 1967
- Pascal, 1970
- C, 1972
- Prolog, 1972
- SQL, 1978
- C++, 1980
- **VHDL**, 1983
- **Verilog**, 1984
- Matlab, 1984
- Perl, 1987
- Python, 1991
- Java, 1995
- Javascript, 1995
- Php, 1995

https://en.wikipedia.org/wiki/History_of_programming_languages

<https://en.wikipedia.org/wiki/VHDL>

- **Hardware Description Language (HDL)**
 - Μια γλώσσα για την μοντελοποίηση της συμπεριφοράς και της δομής των ψηφιακών συστημάτων
- **Electronic Design Automation (EDA) using HDL**
 - Σχεδίαση ηλεκτρονικών κυκλωμάτων με χρήση εργαλείων CAD (computer-aided design)
 - Εισαγωγή σχεδίασης (design entry)
 - Χρήση κώδικα αντί για σχηματικά διαγράμματα
 - Επαλήθευση (verification)
 - Προσομοίωση (simulation) του κώδικα
 - Σύνθεση (synthesis)
 - Αυτόματη παραγωγή των κυκλωμάτων
 - Φυσική υλοποίηση (implementation)
 - Υλοποίηση του κυκλώματος στην τεχνολογία επιλογής

HDL-VHDL

- VHSIC Hardware Description Language (Γλώσσα Περιγραφής Υλικού)
 - VHSIC: Very High-Speed Integrated Circuits
- Ιστορική αναδρομή:
 - Ξεκίνησε το 1981 από το Υπουργείο Άμυνας των ΗΠΑ ως γλώσσα περιγραφής ολοκληρωμένων κυκλωμάτων
 - Οι εταιρείες IBM, Texas Instruments, Intermetrics ανέπτυξαν 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 1076-2000 and 1076 2002 (VHDL-2000, VHDL-2002)
 - IEEE Standard 1076-2008 (VHDL-2008)
- Πιο διαδεδομένη στην Ευρώπη
 - Στην Αμερική η Verilog είναι πιο διαδεδομένη
- Χρησιμοποιείται για την σχεδίαση συστημάτων για το διάστημα
 - Από την NASA και την ESA

Εμείς προγραμματίζουμε μόνο με VHDL-93 για μέγιστη συμβατότητα των προγραμμάτων μας.

Πλεονεκτήματα των HDLs

- 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 εργαλεία CAD
- Υπερτερούν από τις γλώσσες προγραμματισμού
 - Παρέχουν δομές που περιγράφουν καλύτερα το υλικό
 - Παράλληλη εκτέλεση εντολών αντί για ακολουθιακή/σειριακή
 - Παρέχουν δυνατότητα για περιγραφή χρονισμών

Μοντελοποίηση και προσομοίωση

- Αρχικά οι HDLs σχεδιάστηκαν για τη μοντελοποίηση και τη προσομοίωση των συστημάτων υλικού στα υψηλότερα επίπεδα αφαίρεσης
- Χαρακτηριστικά μοντελοποίησης των HDLs:
 - παράλληλη εκτέλεση
 - ιεραρχική σχεδίαση
 - περιγραφή χρονισμών
 - περιγραφή ακολουθίας γεγονότων
 - περιγραφή σύγχρονης/ασύγχρονης συμπεριφοράς

VHDL - VIVADO

Ψηφιακό κύκλωμα – Αναπαράσταση VHDL – Σύνθεση

- Συνήθως σχεδιάζουμε χρησιμοποιώντας HDL επιπέδου μεταφοράς καταχωρητή (Register Transfer Level - RTL)
 - Καταχωρητής (Register): Ψηφιακό στοιχείο που μπορεί να διατηρεί την σταθερή την τιμή του για κάποιο χρονικό διάστημα (κύκλους ρολογιού)
 - υψηλότερο επίπεδο αφαίρεσης από τις πύλες
- Το εργαλείο σύνθεσης μεταφράζει το μοντέλο σε ένα κύκλωμα από πύλες που εκτελεί την ίδια λειτουργία
- Προσδιορίζουμε στο εργαλείο
 - την τεχνολογία υλοποίησης που στοχεύουμε
 - περιορισμούς στο χρονισμό, στην επιφάνεια, κλπ.
- Επαλήθευση μετά τη σύνθεση
 - το κύκλωμα που προέκυψε από τη σύνθεση ικανοποιεί τους περιορισμούς

Δυαδικό Σύστημα

Δυαδική Αναπαράσταση

- Δεκαδικό σύστημα αναπαράστασης αριθμών (0-9)

Στήλη των 1
Στήλη των 10
Στήλη των 100
Στήλη των 1000

$$6598_{10} = 6 \times 10^3 + 5 \times 10^2 + 9 \times 10^1 + 8 \times 10^0$$

Έξι Πέντε Εννέα Οκτώ
Χιλιάδες Εκατοντάδες Δεκάδες Μονάδες

- Δυαδικό σύστημα αναπαράστασης αριθμών (0-1)

Στήλη των 1
Στήλη των 2
Στήλη των 4
Στήλη των 8

$$1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13_{10}$$

Μια Μια Μηδέν Μια
Οκτάδα Τετράδα Δυάδες Μονάδα

Δυαδικό Σύστημα

Αριθμοί χωρίς πρόσημο (Unsigned – Μόνο θετικοί)

Πρόσθεση

0	0	1	1
+0	+1	+0	+1
---	---	---	---
0	1	1	10
Κρατούμενο/(C)arry)			
0	0	0	1

00	01	11	11
01	+01	+01	+11
----	----	-----	-----
01	10	100	110
Κρατούμενο/(C)arry)			
0	0	1	1

Δυαδικό Σύστημα

Αριθμοί χωρίς πρόσημο (Unsigned – Μόνο θετικοί)

Πολλαπλασιασμός/Διαίρεση με πολλαπλάσια του δύο

$$10 = 2(\text{δεκαδικό}) \quad (A)$$

Εφαρμόζουμε αριστερή ολίσθηση και γεμίζουμε δεξιά με το 0

$$10\mathbf{0} = 4(\text{δεκαδικό}) \quad (B) = 2 * (A)$$

$$10\mathbf{00} = 8(\text{δεκαδικό}) \quad (C) = 2 * (B) = 4 * (A)$$

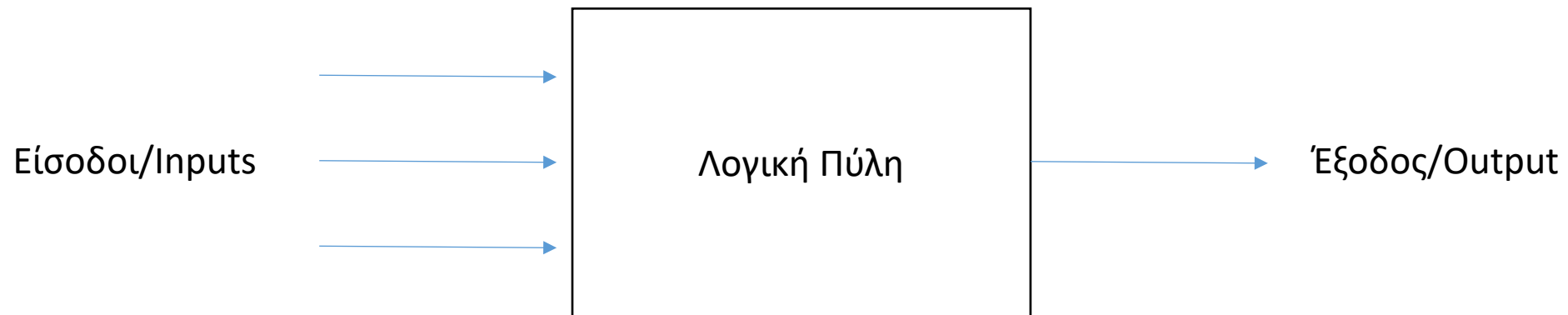
$$1000 = 8(\text{δεκαδικό}) \quad (A)$$

Εφαρμόζουμε δεξιά ολίσθηση διαγράφοντας το πιο δεξί ψηφίο

$$100 = 4(\text{δεκαδικό}) \quad (B) = (A)/2$$

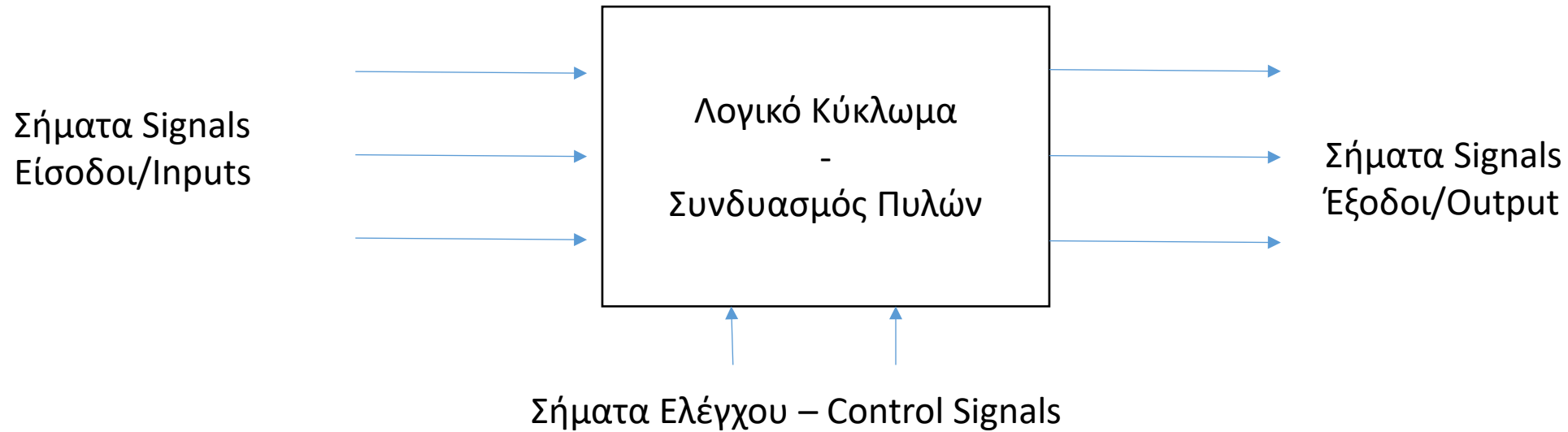
$$10 = 2(\text{δεκαδικό}) \quad (C) = (B)/2 = (A)/4$$

Λογικές Πύλες



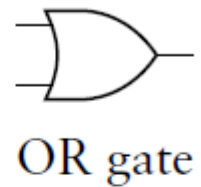
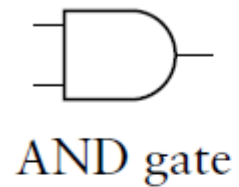
Όλοι οι είσοδοι και έξοδοι είναι σήματα με πιθανές τιμές 0 ή 1

Λογικό κύκλωμα

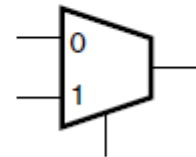


Λογικές Πύλες – Πίνακας Αληθείας (1/2)

- Σε κάθε Πύλη ή Κύκλωμα αντιστοιχεί ένας Πίνακας Αληθείας (Truth Table).
- Ο Πίνακας Αληθείας καθορίζει την τιμή εξόδου για κάθε συνδυασμό εισόδων στην Πύλη ή το Κύκλωμα



inverter



multiplexer

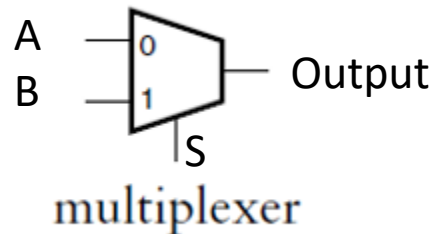
x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

x	$\bar{x}$
0	1
1	0

Λογικές Πύλες – Πίνακας Αληθείας (2/2)

- Σε κάθε Πύλη ή Κύκλωμα αντιστοιχεί ένας Πίνακας Αληθείας (Truth Table).
- Ο Πίνακας Αληθείας καθορίζει την τιμή εξόδου για κάθε συνδυασμό εισόδων στην Πύλη ή το Κύκλωμα



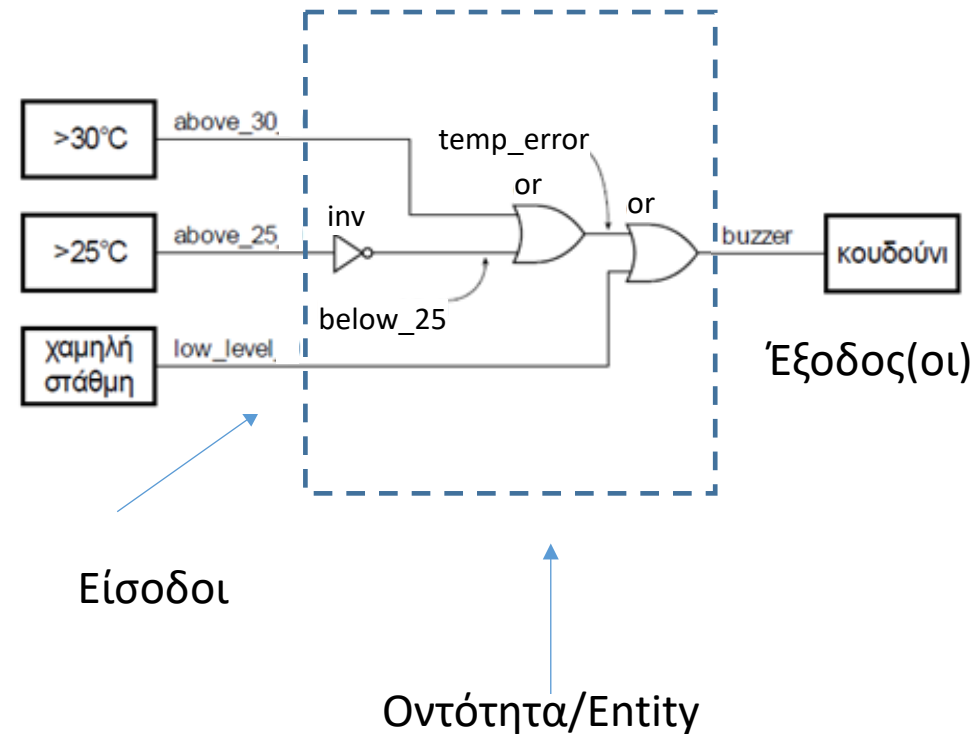
S	Output
0	A
1	B

Αν $A=0$, $B=1$, $S=1$, Output= ;

Αν $A=1$, $B=0$, $S=1$, Output= ;

Ψηφιακό κύκλωμα - Παράδειγμα

Παράδειγμα: Εργοστάσιο έχει δοχείο επεξεργασίας υγρών. Το δοχείο πρέπει α) να έχει θερμοκρασία μεταξύ 25 και 30 βαθμών και β) η στάθμη του πρέπει να είναι πάνω από ένα επίπεδο. Σε περίπτωση που το α) ή το β) δεν ικανοποιούνται πρέπει να ενεργοποιηθεί ένα κουδούνι. Στη διάθεσή μας έχουμε αισθητήρες (θερμόμετρα) που δείχνουν αν η θερμοκρασία ξεπερνά ένα όριο (το οποίο ορίζεται από εμάς για κάθε αισθητήρα) και αισθητήρα που μας ενημερώνει εάν η στάθμη του δοχείου είναι κάτω από ένα επίπεδο ασφαλείας. Περιγράψτε το σύστημα.



VHDL - Vivado

Ψηφιακό κύκλωμα - Παράδειγμα

Το κουδούνι χτυπάει όταν:

A) Η θερμοκρασία ΕΙΝΑΙ πάνω από 30

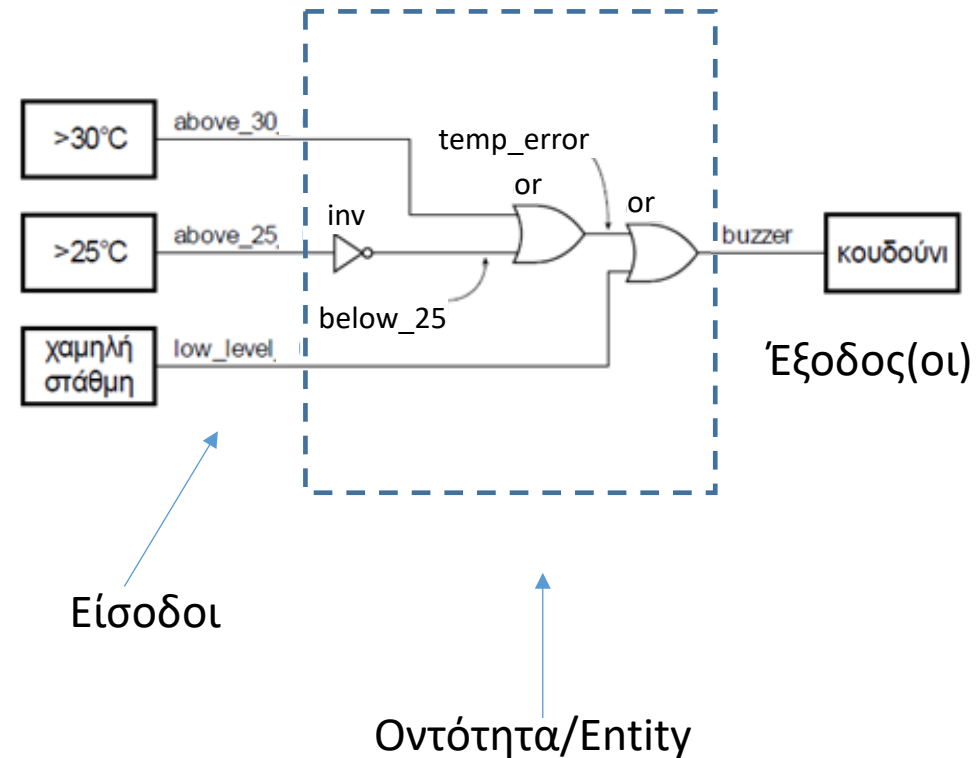
ή

B) Η θερμοκρασία ΕΙΝΑΙ κάτω από 25 =>

Η θερμοκρασία ΔΕΝ ΕΙΝΑΙ πάνω από 25.

ή

Γ) Η στάθμη είναι χαμηλή



Ψηφιακό κύκλωμα - Παράδειγμα

Το κουδούνι χτυπάει όταν:

A) Η θερμοκρασία ΕΙΝΑΙ πάνω από 30

ή

B) Η θερμοκρασία ΕΙΝΑΙ κάτω από 25 =>

Η θερμοκρασία ΔΕΝ ΕΙΝΑΙ πάνω από 25.

ή

Γ) Η στάθμη είναι χαμηλή

Συνθήκη	Τιμή σήματος
Θερμοκρασία πάνω από 30	Above_30 =1/TRUE
Θερμοκρασία κάτω από 30	Above_30 =0/FALSE
Θερμοκρασία πάνω από 25	Above_25 =1/TRUE
Θερμοκρασία κάτω από 25	Above_25 =0/FALSE
Χαμηλή Στάθμη ΝΑΙ	Low_level=1/TRUE
Χαμηλή Στάθμη ΟΧΙ	Low_level=0/FALSE
Κουδούνι χτυπάει	Buzzer=1/TRUE
Κουδούνι ΔΕΝ χτυπάει	Buzzer=0/FALSE

**Ψάχνουμε πότε χτυπάει το κουδούνι, δηλαδή πότε
θέλουμε το σήμα Buzzer να έχει την τιμή 1/TRUE**

Ψηφιακό κύκλωμα - Παράδειγμα

Το κουδούνι χτυπάει όταν:

A) Η θερμοκρασία ΕΙΝΑΙ πάνω από 30

ή

B) Η θερμοκρασία ΕΙΝΑΙ κάτω από 25 =>

Η θερμοκρασία ΔΕΝ ΕΙΝΑΙ πάνω από 25.

ή

Γ) Η στάθμη είναι χαμηλή

Συνθήκη	Τιμή σήματος
Θερμοκρασία πάνω από 30	Above_30 =1/TRUE
Θερμοκρασία κάτω από 30	Above_30 =0/FALSE
Θερμοκρασία πάνω από 25	Above_25 =1/TRUE
Θερμοκρασία κάτω από 25	Above_25 =0/FALSE
Χαμηλή Στάθμη ΝΑΙ	Low_level=1/TRUE
Χαμηλή Στάθμη ΟΧΙ	Low_level=0/FALSE
Κουδούνι χτυπάει	Buzzer=1/TRUE
Κουδούνι ΔΕΝ χτυπάει	Buzzer=0/FALSE

Buzzer=1 όταν Above_30=1 ή Above_25=0 ή Low_level=1

Ψηφιακό κύκλωμα - Παράδειγμα

Buzzer=1 όταν Above_30=1 ή Above_25=0 ή Low_level=1

Ισοδύναμο σε Άλγεβρα Boole

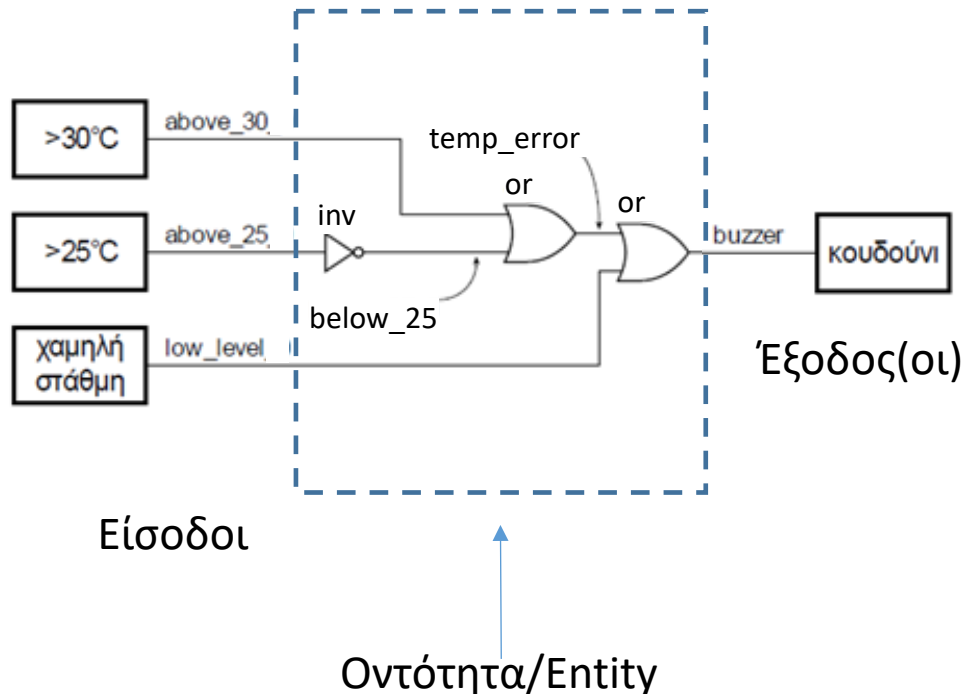
Buzzer=Above_30 or (not Above_25) or Low_level

Ισοδύναμο σε VHDL

Buzzer<=Above_30 or (not Above_25) or Low_level;

VHDL - Vivado

Ψηφιακό κύκλωμα – VHDL: Entity (Input/Output PORTS)



```
library IEEE;  
use IEEE.STD_LOGIC_1164.ALL;
```

entity buzzer is

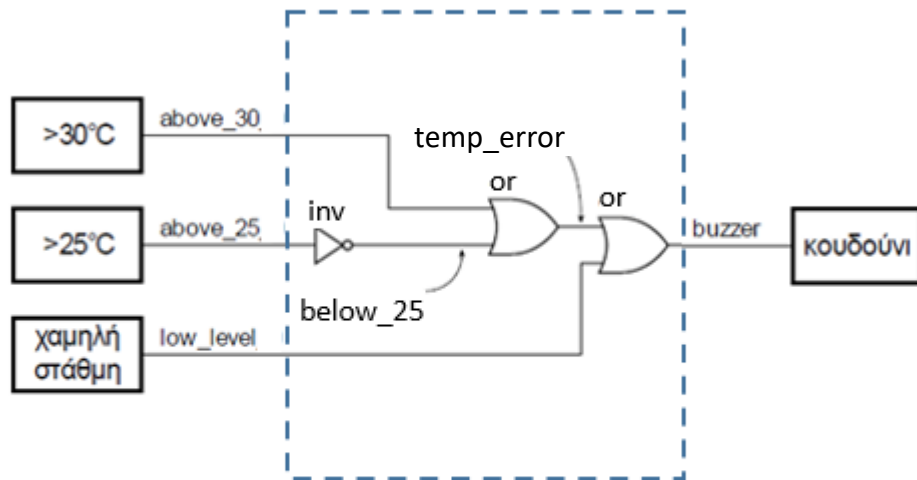
```
port (  
  above_25 : in std_logic;  
  above_30 : in std_logic;  
  low_level : in std_logic;  
  buzzer   : out std_logic );
```

end entity buzzer;

Περιγραφή Οντότητας

VHDL - Vivado

Ψηφιακό κύκλωμα – VHDL: Architecture (Behavioral)



Αρχιτεκτονική
Περιγραφή της λειτουργικότητας
που προσφέρει το λογικό
κύκλωμα

architecture Dataflow of buzzer is

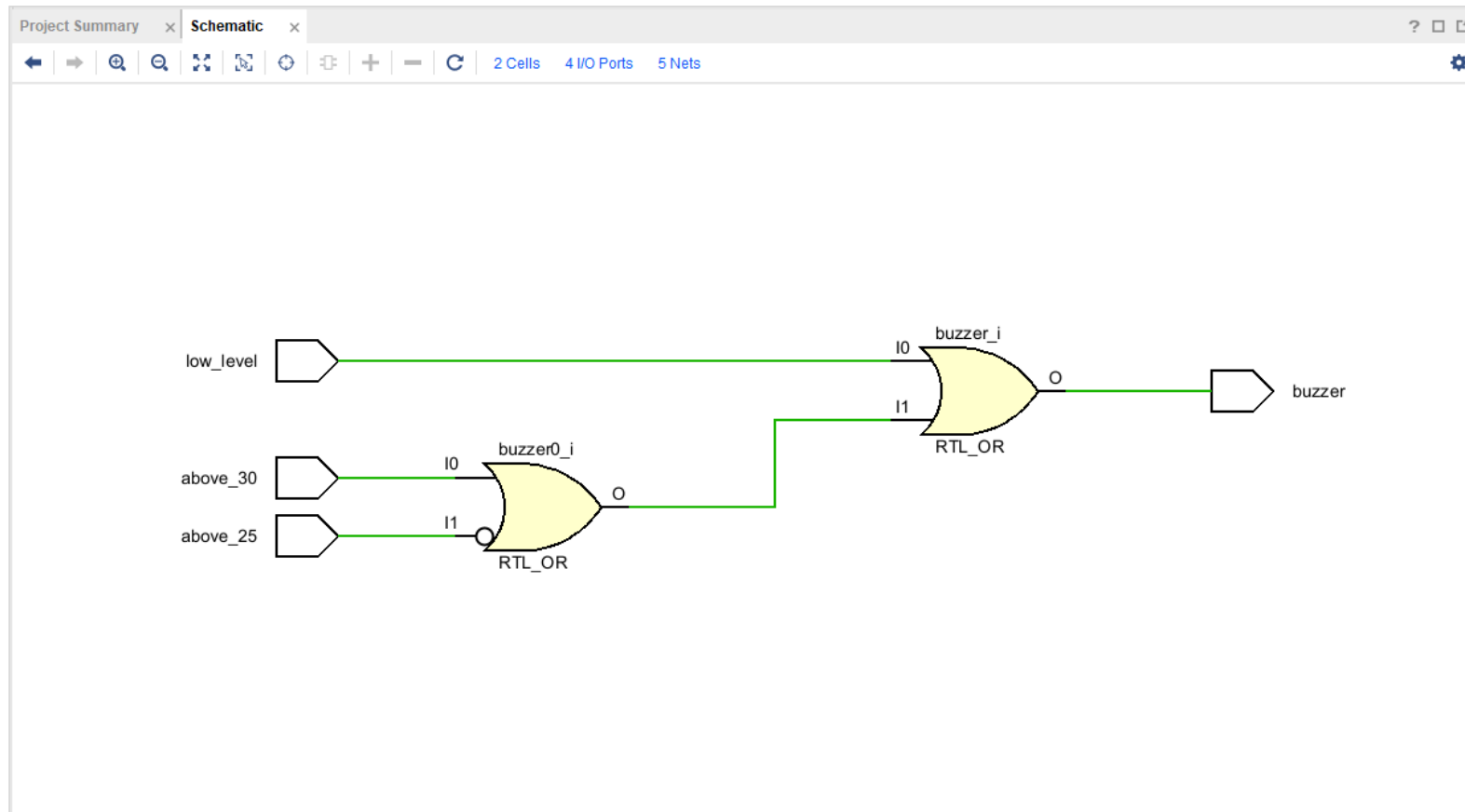
begin

buzzer<= above_30 or not above_25 or low_level;

end architecture Dataflow;

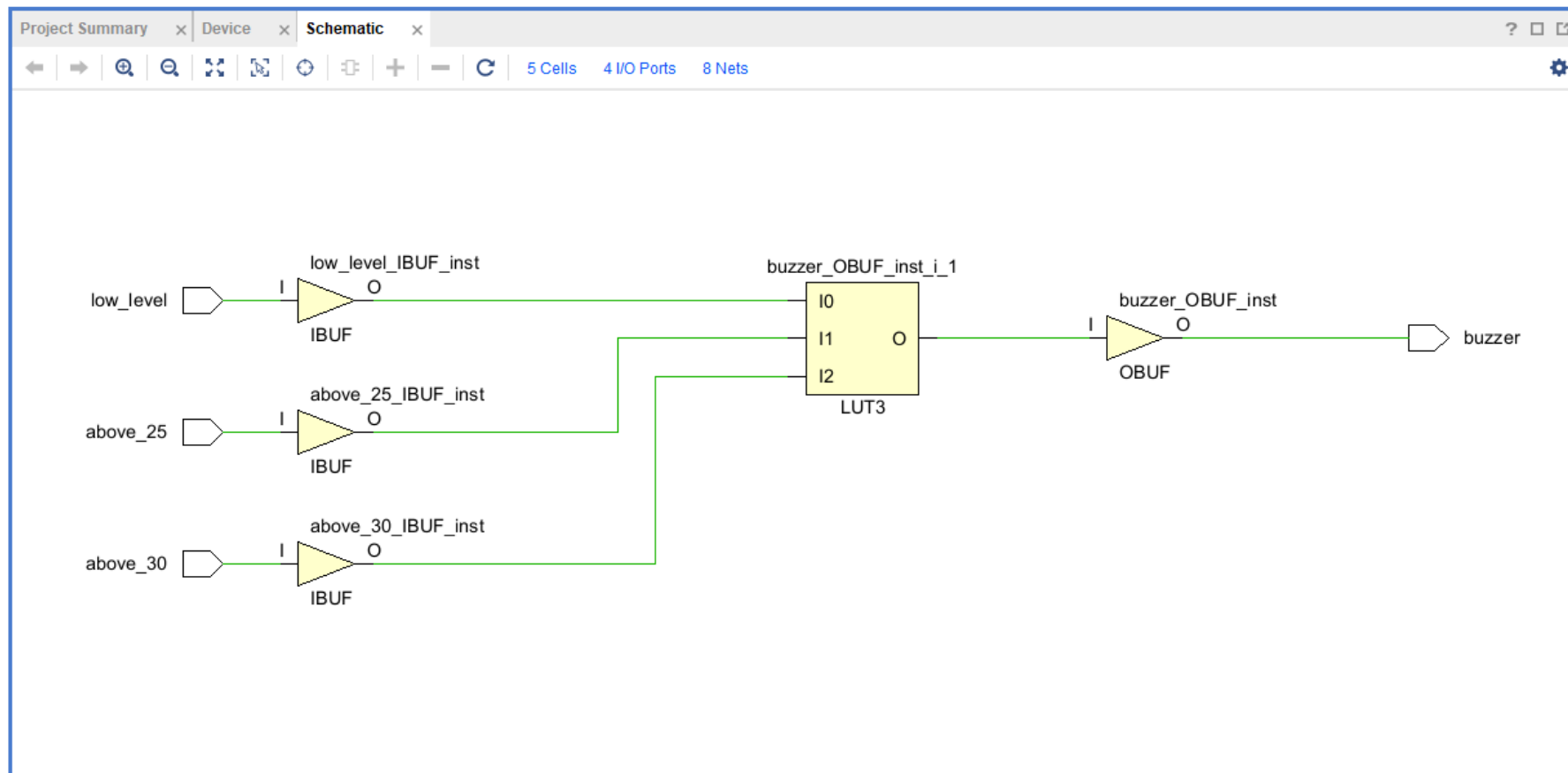
VHDL - Vivado

Ψηφιακό κύκλωμα – VHDL: RTL Αναπαράσταση

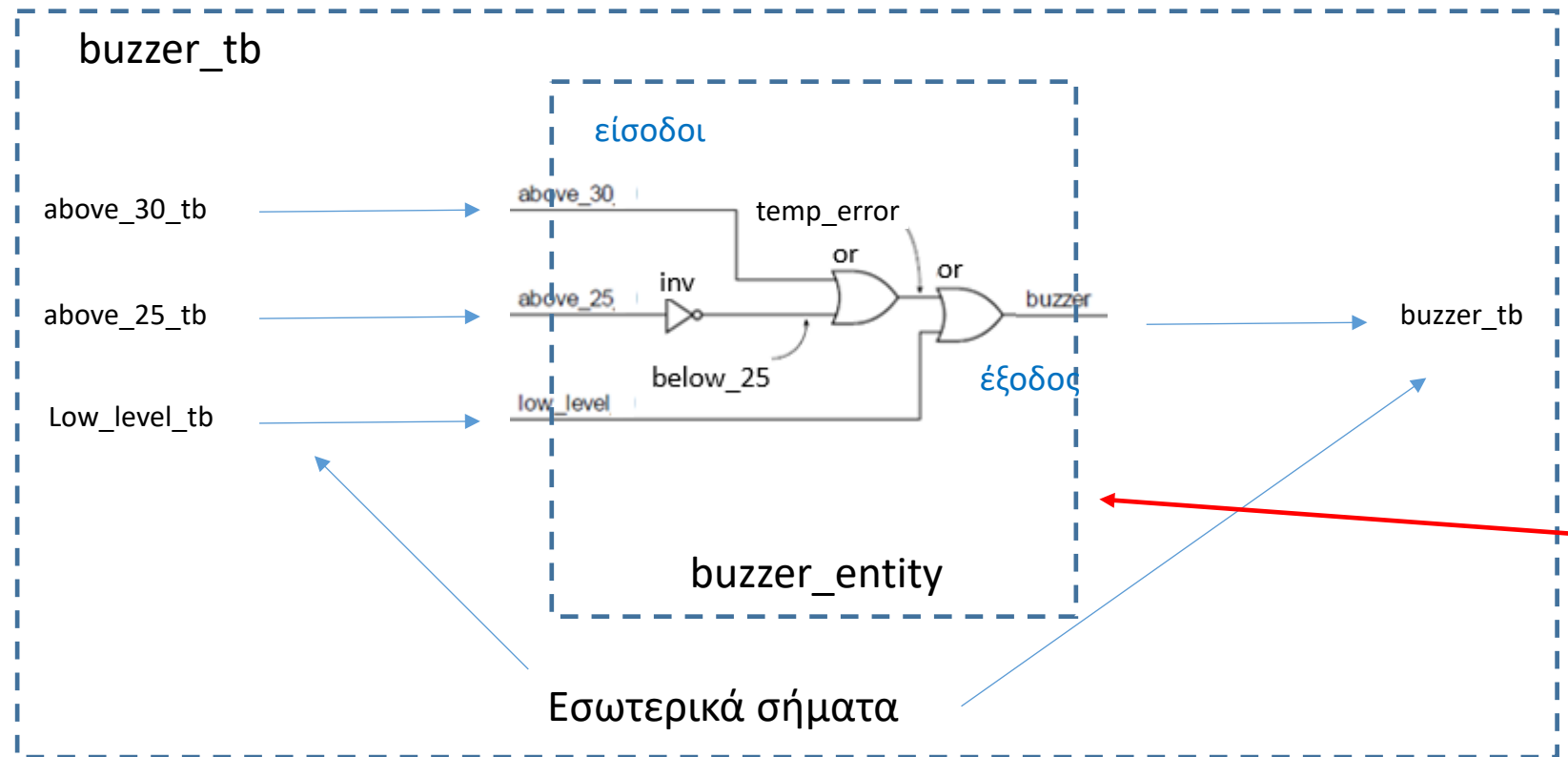


VHDL - Vivado

Ψηφιακό κύκλωμα – VHDL: Σύνθεση/Υλοποίηση



Ψηφιακό κύκλωμα – VHDL: Προσομοίωση



VHDL - Vivado

Ψηφιακό κύκλωμα – VHDL: Προσομοίωση – Πίνακας Αληθείας

Πίνακας Αληθείας της συνάρτησης που εκφράζει το σήμα buzzer

above_30	above_25	low_level	buzzer
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Ψηφιακό κύκλωμα – VHDL: Προσομοίωση - Testbench

```
library IEEE; use IEEE.STD_LOGIC_1164.ALL;
```

```
entity buzzer_tb is  
end buzzer_tb ;
```

```
architecture Behavioral of buzzer_tb is
```

```
component buzzer port(  
  above_30: in std_logic;  
  above_25: in std_logic;  
  low_level: in std_logic;  
  buzzer: out std_logic);  
end component buzzer;
```

```
signal  above_25_tb, above_30_tb, low_level_tb  : std_logic;  
signal  buzzer_tb                                : std_logic;
```

```
begin  
  uut: buzzer port map (above_25 => above_25_tb,  
    above_30 => above_30_tb, low_level => low_level_tb,  
    buzzer => buzzer_tb);
```

```
  apply_test_cases: process is  
  begin
```

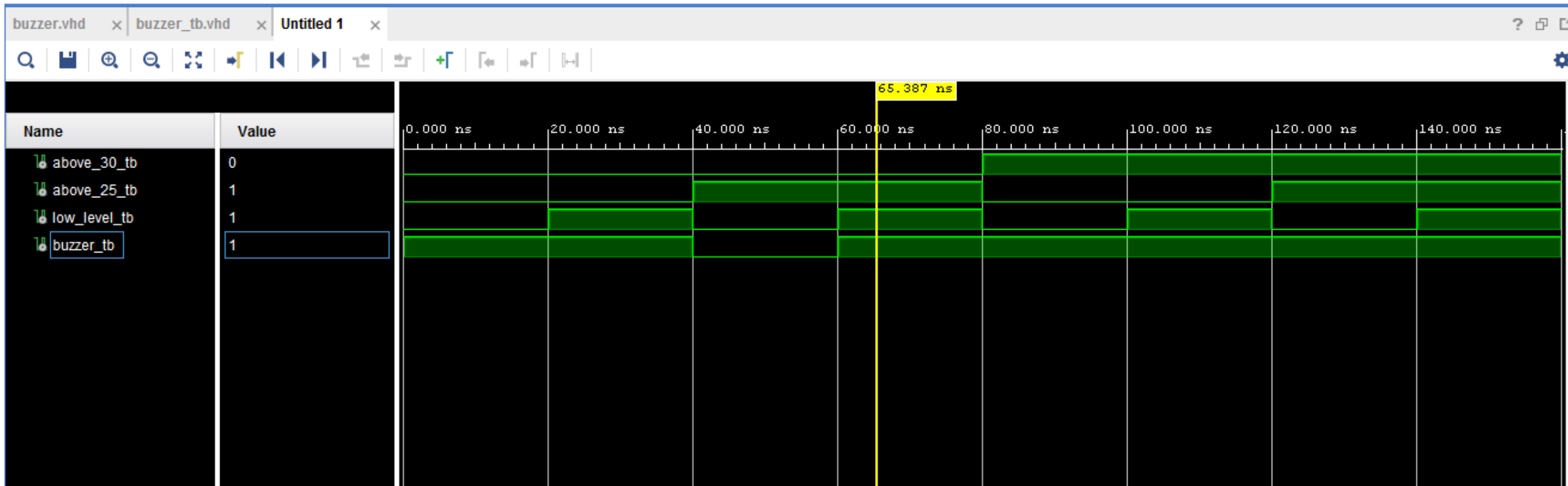
```
    above_30_tb <='0'; above_25_tb <='0'; low_level_tb <='0'; wait for 20 ns;  
    above_30_tb <='0'; above_25_tb <='0'; low_level_tb <='1'; wait for 20 ns;  
    above_30_tb <='0'; above_25_tb <='1'; low_level_tb <='0'; wait for 20 ns;  
    above_30_tb <='0'; above_25_tb <='1'; low_level_tb <='1'; wait for 20 ns;  
    above_30_tb <='1'; above_25_tb <='0'; low_level_tb <='0'; wait for 20 ns;  
    above_30_tb <='1'; above_25_tb <='0'; low_level_tb <='1'; wait for 20 ns;  
    above_30_tb <='1'; above_25_tb <='1'; low_level_tb <='0'; wait for 20 ns;  
    above_30_tb <='1'; above_25_tb <='1'; low_level_tb <='1'; wait for 20 ns;
```

```
  end process apply_test_cases;
```

```
end Behavioral;
```

VHDL - Vivado

Ψηφιακό κύκλωμα – VHDL: Προσομοίωση - Χρονοσειρά



Περίληψη

- Εισαγωγή στη θεωρία γλωσσών HDL
- Εισαγωγή στο δυαδικό σύστημα αρίθμησης και στις λογικές πύλες
- Παράδειγμα ανάπτυξης εφαρμογής σε VHDL
- Διαβάζετε τις παραγράφους 2.1 και 3.1 από το βιβλίο του Ashenden.
- Διαβάζετε τις παραγράφους 1.5.1, 1.5.2, 1.5.3, 1.5.4, 4.1 από το βιβλίο των Harris.
- Διαβάζετε τις παραγράφους 1.5, 1.5.1, 1.5.2, 2.9, 2.10, 2.10.1 από το βιβλίο των Brown-Vranesic.
- Κατεβάζετε και κάνετε εγκατάσταση το Vivado. [Οδηγίες υπάρχουν στο eclass.](#)

Εργαστήριο Λογικής Σχεδίασης

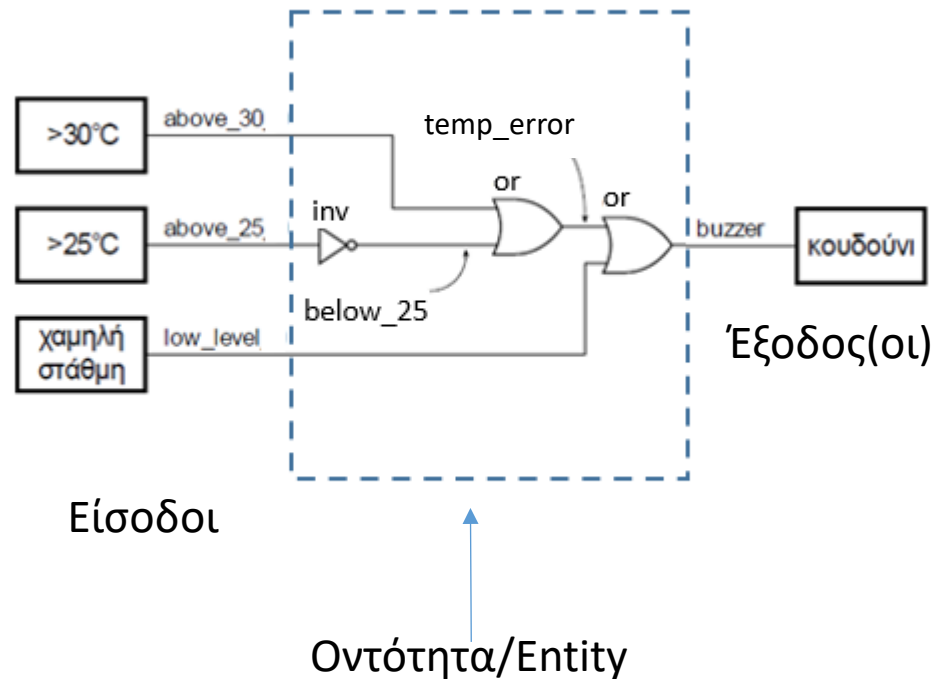
3^η Διάλεξη

**Ανάπτυξη προγράμματος στη VHDL
(Αρχιτεκτονική Dataflow)**

Βασιλόπουλος Διονύσης

Ε.ΔΙ.Π. Τμήματος Πληροφορικής & Τηλεπικοινωνιών

Ψηφιακό κύκλωμα – VHDL: Entity (Input/Output PORTS)



library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity buzzer **is**

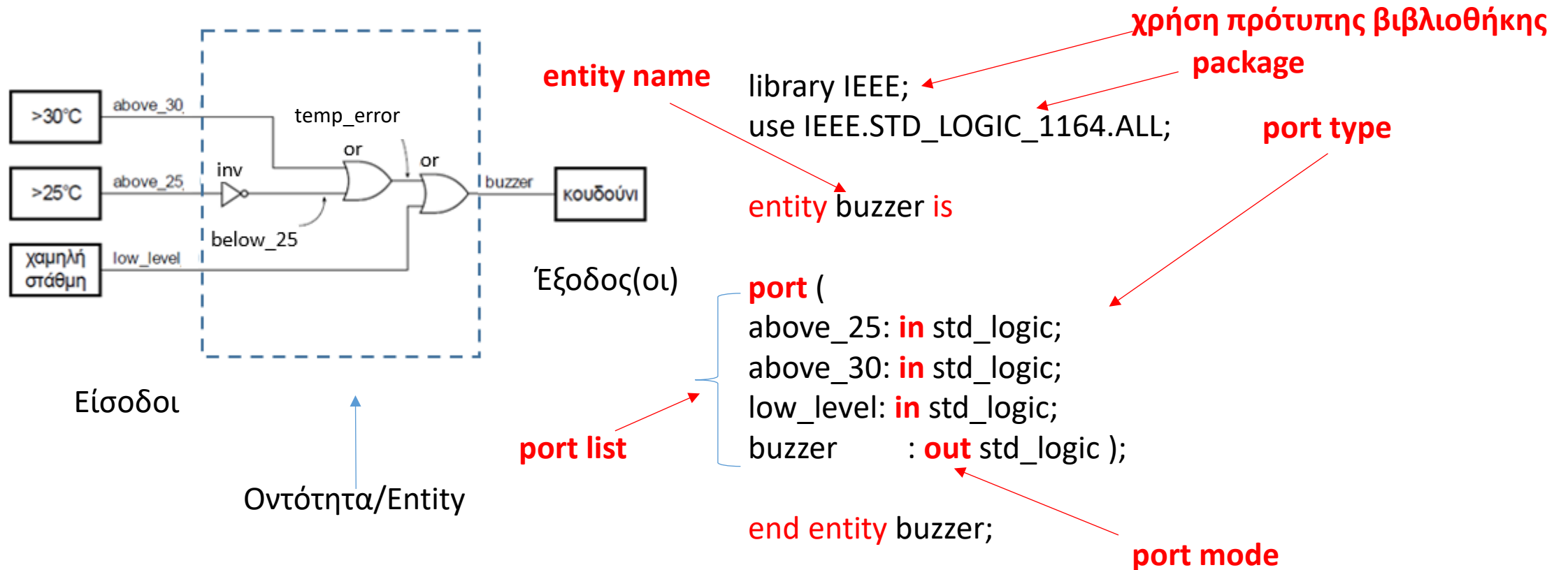
port (
above_25: **in** std_logic;
above_30: **in** std_logic;
low_level: **in** std_logic;
buzzer : **out** std_logic);

end entity buzzer;

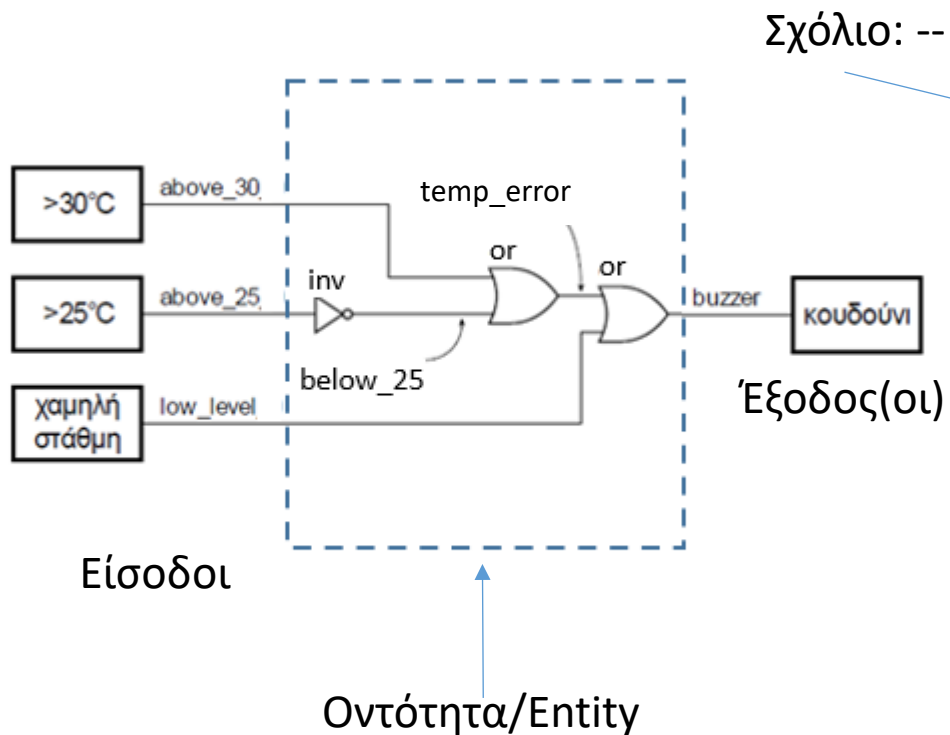
Υποχρεωτικές Βιβλιοθήκες

Περιγραφή Οντότητας

Ψηφιακό κύκλωμα – VHDL: Entity (Input/Output PORTS)



Ψηφιακό κύκλωμα – VHDL: Entity (Input/Output PORTS)



Χωρίς τις Βιβλιοθήκες

```
--library IEEE;  
--use IEEE.STD_LOGIC_1164.ALL;
```

Περιγραφή Οντότητας

```
entity buzzer is  
  
  port (  
    above_25: in bit;  
    above_30: in bit;  
    low_level: in bit;  
    buzzer   : out bit);  
  
end entity buzzer;
```

Ο τύπος **bit**, (αλλά και **bit_vector**, ...) υποστηρίζεται χωρίς την ανάγκη κλήσης βιβλιοθηκών

Ψηφιακό κύκλωμα – Αναπαράσταση σε VHDL – Entity: Input/Output

- Περιγράφει τη διασύνδεση μίας λογικής μονάδας, χωρίς να προσδιορίζει τη συμπεριφορά της (μαύρο κουτί - black box)
- Η διασύνδεση της μονάδας περιγράφεται με μία δήλωση των **διαύλων/θυρών επικοινωνίας (ports - signals)**

```
entity entity_name is -- σχόλια
    port (
        signal_name: mode
    signal_type;
        signal_name: mode
    signal_type;
        ...
        signal_name: mode
    signal_type);
end entity entity_name;
```

Ψηφιακό κύκλωμα – Αναπαράσταση σε VHDL – Entity: Input/Output

- **entity_name**: το όνομα της οντότητας
- **signal_name**: το όνομα του σήματος
(εάν είναι πολλά σήματα χωρίζονται με κόμμα)
- **mode**: η κατεύθυνση του σήματος
 - **in**: είσοδος της οντότητας
 - **out**: έξοδος της οντότητας
 - **inout**: είσοδος ή έξοδος της οντότητας (bidirectional), (ΔΕΝ θα μας απασχολήσει στο μάθημα)
- **signal_type**: ο τύπος του σήματος (STD_LOGIC ή άλλος)

Ψηφιακό κύκλωμα – Αναπαράσταση σε VHDL – Ονόματα & Ετικέτες

- Είναι **μοναδικά** μέσα σε μία συγκεκριμένη οντότητα (και αρχιτεκτονική)
- Τα σχόλια σε μία γραμμή έπονται του --
- Χρησιμοποιούνται οι χαρακτήρες: **a-z, A-Z, 0-9, _**
- Δεν χρησιμοποιούνται οι χαρακτήρες, όπως: **+, -, !, &**
- Δεν χρησιμοποιούνται ούτε **σημεία στίξης** στα ονόματα και τις ετικέτες, ούτε διπλό **"_"**, δηλαδή **__**
- Δεν διαχωρίζονται κεφαλαία γράμματα από μικρά
- Ο πρώτος χαρακτήρας είναι **αλφαβητικός**
- **Προσοχή στις δεσμευμένες λέξεις**

VHDL – Τύποι σημάτων

Std_logic

Τιμή	Modeling for simulation	Synthesis
U	Uninitialized	Uninitialized
X	Strong driven unknown	Don't care
0	Strong driven 0	0
1	Strong driven 1	1
Z	High impedance	High impedance
W	Weakly driven unknown	Don't care
L	Weakly driven 0	0
H	Weakly driven 1	1
-	Don't care	Don't care

VHDL – Τύποι σημάτων

std_logic_vector

```
signal_0: out std_logic;  
signal_1: out std_logic;  
signal_2: out std_logic;  
signal_3: out std_logic;
```

Εναλλακτικά

```
signal_all : out std_logic_vector (3 downto 0);
```

```
signal_all <= "0101";
```

```
signal_all(0) <= '1';  
signal_all(1) <= '0';  
signal_all(2) <= '1';  
signal_all(3) <= '0';
```

Προσοχή σε " και '

VHDL – Τύποι σημάτων

std_logic_vector

- Ο τύπος του λογικού διανύσματος (μονοδιάστατου array) **STD_LOGIC_VECTOR** είναι μέρος του πακέτου **IEEE.std_logic_1164** της βιβλιοθήκης **IEEE**
- Προσδιορίζει ένα διατεταγμένο σύνολο από σήματα (μεταβλητές) τύπου **STD_LOGIC**.
- Η διάταξη μπορεί να είναι αύξουσα
STD_LOGIC_VECTOR (0 to 7)
ή φθίνουσα
STD_LOGIC_VECTOR (7 downto 0)
- Οι δείκτες των στοιχείων του array είναι **θετικοί ακέραιοι ή μηδέν**
- Προσοχή, δεν είναι **ακέραιος δυαδικός αριθμός**

VHDL – Τύποι σημάτων

std_logic_vector

- Δήλωση τιμών για το 8-ψήφιο λογικό διάνυσμα V
 - `V <= "11110000"`
 - `V <= (others => '0')` -- όλα '0'
- Συγκρίσεις:
 - If `V = "00000000"` για σύγκριση **ολόκληρου** του διανύσματος
 - If `V(3 downto 0) = "0000"` για σύγκριση **μέρους** του διανύσματος

Προσοχή. Σε όλα τα προγράμματα τα PORT στον ορισμό της Οντότητας θα είναι **MONO STD_LOGIC ή STD_LOGIC_VECTOR**

VHDL – Packages

Package: IEEE.std_logic_1164.all

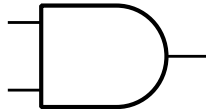
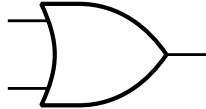
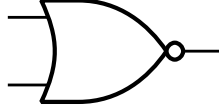
Logical Operators

and, nand, or, nor, xor, xnor, not

Edge Detection

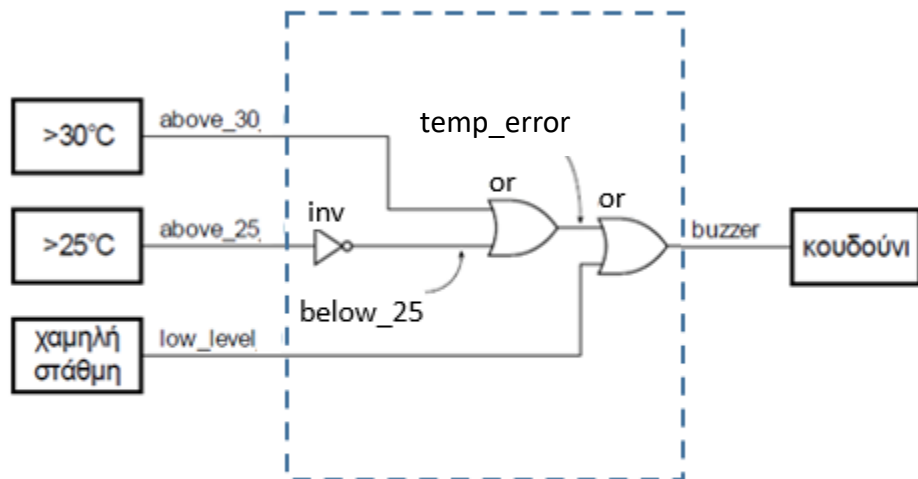
rising_edge() (πότε ένα σήμα γίνεται από '0'-'>'1'),
falling_edge() (πότε ένα σήμα γίνεται από '1'-'>'0')

Package: IEEE.std_logic_1164.all

<code>a and b</code>	$a \cdot b$	
<code>a or b</code>	$a + b$	
<code>a nand b</code>	$\overline{a \cdot b}$	
<code>a nor b</code>	$\overline{a + b}$	
<code>a xor b</code>	$a \oplus b$	
<code>a xnor b</code>	$\overline{a \oplus b}$	
<code>not a</code>	$\overline{a}$	

- Προτεραιότητα
 - το **not** έχει την υψηλότερη
 - οι υπόλοιποι τελεστές έχουν ίση προτεραιότητα
 - από αριστερά προς τα δεξιά
 - χρησιμοποιούμε παρενθέσεις για να διακρίνουμε τη σειρά υπολογισμού
- Τιμές bit στην VHDL
 - '0' και '1'

Ψηφιακό κύκλωμα – VHDL: Architecture (Dataflow)



architecture Dataflow of buzzer is

begin

```
buzzer<= low_level or (above_30 or not above_25);
```

end architecture Dataflow;

Αρχιτεκτονική

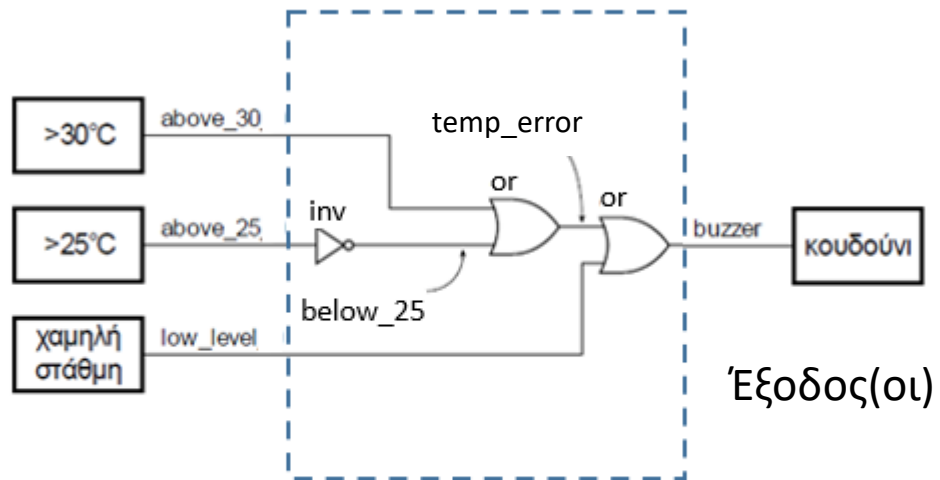
Περιγραφή της λειτουργικότητας που προσφέρει το λογικό κύκλωμα

Ανάθεση τιμής

Με το **<=** δίνουμε τιμή σε ένα σήμα

Τα σήματα εξόδου (out) ΠΑΝΤΑ αριστερά, τα σήματα εισόδου (in) ΠΑΝΤΑ δεξιά.

Ψηφιακό κύκλωμα – VHDL: Entity (Εσωτερικά σήματα)



Είσοδοι

Οντότητα/Entity

Εσωτερικά σήματα

temp_error

-

below_25

Ψηφιακό κύκλωμα – Αναπαράσταση σε VHDL – Architecture

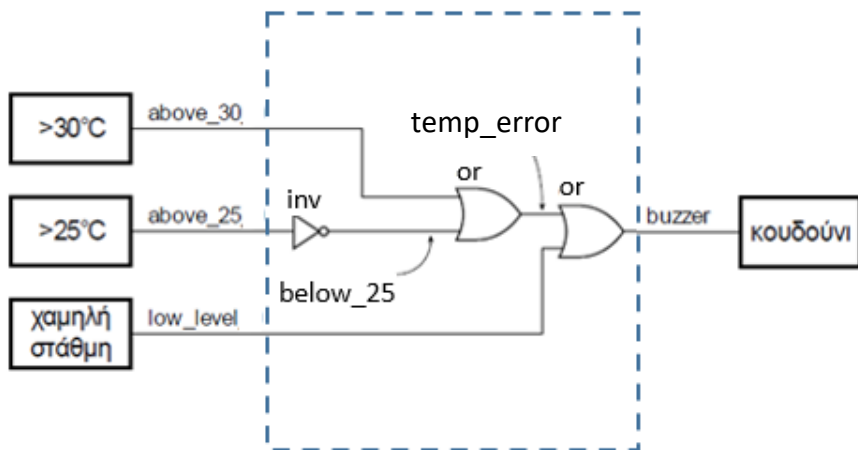
Αρχιτεκτονική Dataflow

```
architecture arch_name of entity_name is  
    signal signal_name: signal_type;  
    ...  
begin  
    concurrent_component_statement;  
    ...  
    concurrent_component_statement;  
end architecture arch_name;
```

Ψηφιακό κύκλωμα – Αναπαράσταση σε VHDL – Architecture

- **arch_name**: το όνομα της αρχιτεκτονικής
- **entity_name**: το όνομα της οντότητας
- **signal_name**: το όνομα του σήματος (εάν είναι πολλά σήματα χωρίζονται με κόμμα)
 - στις δηλώσεις σημάτων (μετά το **is**) το σήμα είναι μία **εσωτερική διασύνδεση** της αρχιτεκτονική της οντότητας
 - στις δηλώσεις των διαύλων του στοιχείου (component) το σήμα είναι είσοδος, έξοδος του στοιχείου, όπως ακριβώς προκύπτει από τη δήλωση των διαύλων της οντότητας του συγκεκριμένου στοιχείου
- **signal_type**: ο τύπος του σήματος (STD_LOGIC ή άλλος)

Ψηφιακό κύκλωμα – VHDL: Architecture (Dataflow)



architecture Dataflow of buzzer is
signal temp_error, below_25: std_logic;
begin

below_25 <= not above_25;
temp_error <= above_30 or **below_25**;
buzzer <= **temp_error** or low_level;

end architecture Dataflow;

Περιγραφή της λειτουργικότητας που προσφέρει το λογικό κύκλωμα με **χρήση εσωτερικών σημάτων**

ταυτόχρονες (concurrent) statements

Τα εσωτερικά σήματα μπορούν να είναι και αριστερά και δεξιά

Ψηφιακό κύκλωμα – Αναπαράσταση σε VHDL – Architecture

- Ταυτόχρονες εντολές ανάθεσης σήματος (concurrent_signal_assignment_statements)

```
signal_name <= expression;
```

- **expression**: έκφραση με σήματα και τελεστές
- **signal_name**: το όνομα του σήματος
 - στις ταυτόχρονες εντολές ανάθεσης σήματος :
 - στην **έκφραση (expression)** προσδιορίζονται σήματα που είναι **είσοδοι** στην οντότητα και δηλώνονται κατά τη δήλωση των διαύλων της οντότητας, και **εσωτερικές διασυνδέσεις (εσωτερικά σήματα)** που δηλώνονται κατά τη δήλωση σημάτων
 - στο αριστερό μέρος της εντολής προσδιορίζεται σήμα που είναι **έξοδος** της οντότητας και δηλώνεται κατά τη δήλωση των διαύλων της οντότητας, ή **εσωτερική διασύνδεση (εσωτερικό σήμα)** που δηλώνεται κατά τη δήλωση σημάτων

Ψηφιακό κύκλωμα – Αναπαράσταση σε VHDL – Architecture

- **Εκτέλεση** ταυτόχρονων εντολών ανάθεσης σήματος (concurrent_signal_assignment_statements)

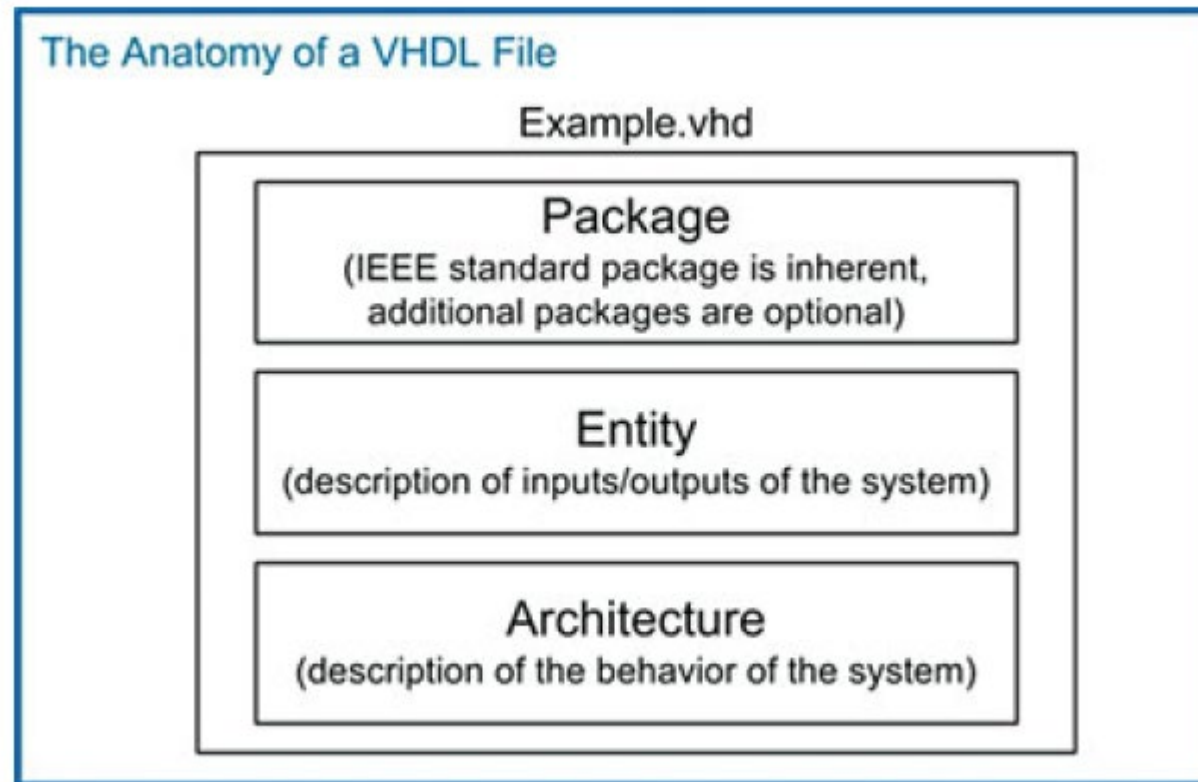
```
signal_name <= expression;
```

- Οι ταυτόχρονες εντολές ανάθεσης σήματος εκτελούνται μόνο, όταν υπάρξει αλλαγή τιμής στις εισόδους (στα σήματα της δεξιάς πλευράς της ταυτόχρονης εντολής ανάθεσης σήματος).
- Δεν προσδιορίζεται καθυστέρηση διάδοσης άλλη, εκτός από μία απειροελάχιστη καθυστέρηση διάδοσης, την **καθυστέρηση δέλτα δ_{delay}** , που δεν επηρεάζει τον χρονισμό του κυκλώματος
- Η πραγματική καθυστέρηση διάδοσης θα προσδιορισθεί με την υλοποίηση σε μία συγκεκριμένη τεχνολογία

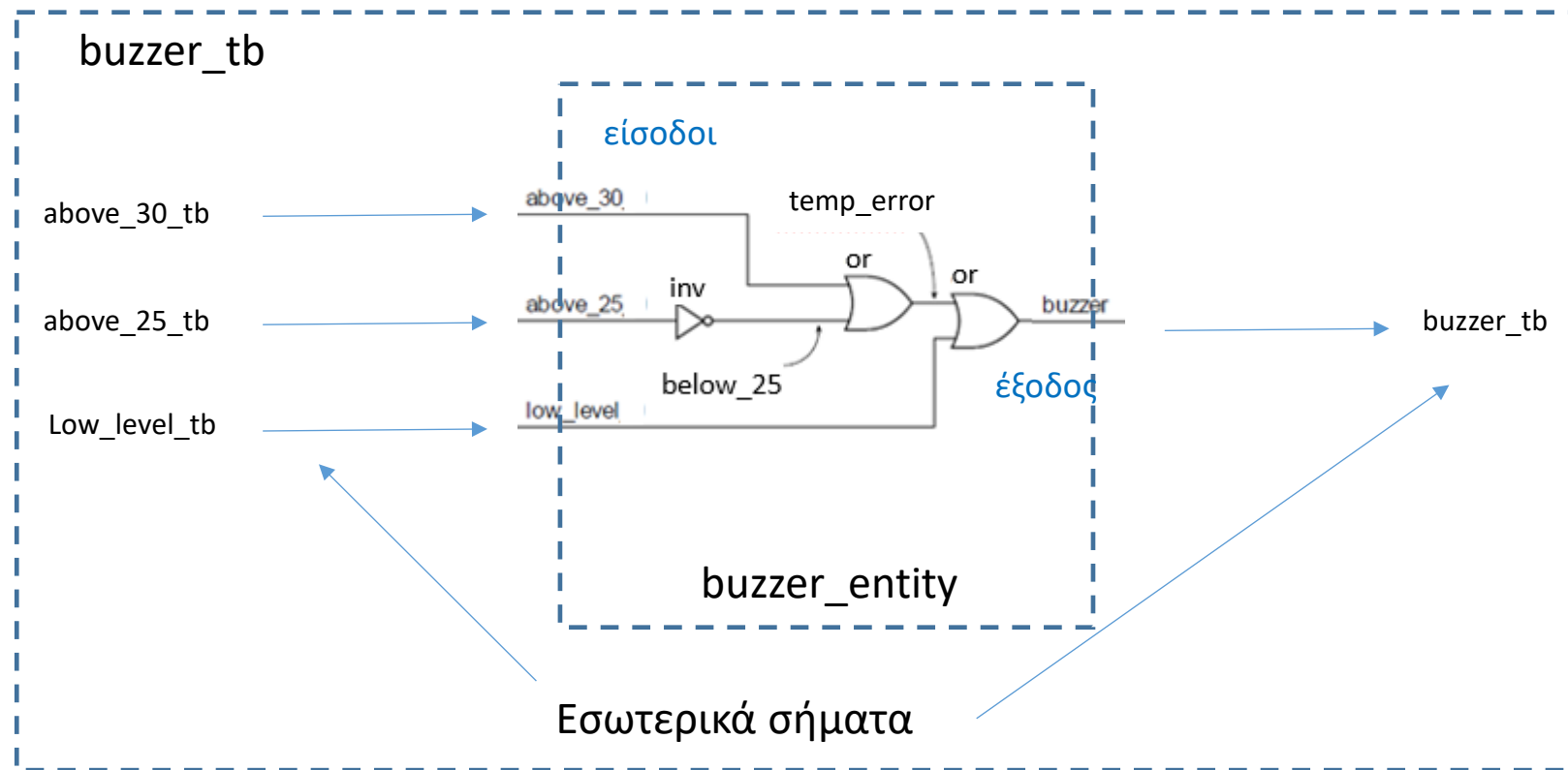
Η **καθυστέρηση δέλτα δ_{delay}** δεν είναι πραγματική καθυστέρηση που επηρεάζει την προσομοίωση, αλλά απλώς ιεραρχεί τις μεταβάσεις που συμβαίνουν στα σήματα την ίδια χρονική στιγμή.

VHDL - Vivado

Ψηφιακό κύκλωμα – VHDL: Δομή Αρχείου



Ψηφιακό κύκλωμα – VHDL: Προσομοίωση



Ψηφιακό κύκλωμα – VHDL: Προσομοίωση – Πίνακας Αληθείας

Πίνακας Αληθείας της συνάρτησης που εκφράζει το σήμα buzzer

above_30	above_25	low_level	buzzer
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Ψηφιακό κύκλωμα – VHDL: Προσομοίωση - Testbench

```
library IEEE; use IEEE.STD_LOGIC_1164.ALL;
```

```
entity buzzer_tb is  
end buzzer_tb ;
```

```
architecture Behavioral of buzzer_tb is
```

```
component buzzer is  
port(  
  above_30: in std_logic;  
  above_25: in std_logic;  
  low_level: in std_logic;  
  buzzer: out std_logic);  
end component buzzer;
```

```
signal  above_25_tb, above_30_tb, low_level_tb  : std_logic;  
signal  buzzer_tb                               : std_logic;
```

```
begin
```

```
 uut: buzzer port map (  
  above_25 => above_25_tb,  
  above_30 => above_30_tb,  
  low_level => low_level_tb,  
  buzzer => buzzer_tb);
```

```
apply_test_cases: process is  
begin
```

```
  above_30_tb <='0'; above_25_tb <='0'; low_level_tb <='0'; wait for 20 ns;  
  above_30_tb <='0'; above_25_tb <='0'; low_level_tb <='1'; wait for 20 ns;  
  above_30_tb <='0'; above_25_tb <='1'; low_level_tb <='0'; wait for 20 ns;  
  above_30_tb <='0'; above_25_tb <='1'; low_level_tb <='1'; wait for 20 ns;  
  above_30_tb <='1'; above_25_tb <='0'; low_level_tb <='0'; wait for 20 ns;  
  above_30_tb <='1'; above_25_tb <='0'; low_level_tb <='1'; wait for 20 ns;  
  above_30_tb <='1'; above_25_tb <='1'; low_level_tb <='0'; wait for 20 ns;  
  above_30_tb <='1'; above_25_tb <='1'; low_level_tb <='1'; wait for 20 ns;  
end process apply_test_cases;
```

```
end Behavioral;
```

Όταν ορίζουμε άμεσα την αντιστοίχιση των εσωτερικών σημάτων με τα port, η σειρά με την οποία το κάνουμε είναι αδιάφορη

Ψηφιακό κύκλωμα – VHDL: Προσομοίωση - Testbench

```
library IEEE; use IEEE.STD_LOGIC_1164.ALL;
```

```
entity buzzer_tb is  
end buzzer_tb ;
```

```
architecture Behavioral of buzzer_tb is
```

```
component buzzer port(  
  above_30: in std_logic;  
  above_25: in std_logic;  
  low_level: in std_logic;  
  buzzer: out std_logic);  
end component buzzer;
```

```
signal  above_25_tb, above_30_tb, low_level_tb  : std_logic;  
signal  buzzer_tb                                : std_logic;
```

```
begin  
  uut: buzzer port map (above_30_tb, above_25_tb, low_level_tb,  
    buzzer_tb);
```

```
  apply_test_cases: process is  
  begin
```

```
    above_30_tb <='0'; above_25_tb <='0'; low_level_tb <='0'; wait for 20 ns;  
    above_30_tb <='0'; above_25_tb <='0'; low_level_tb <='1'; wait for 20 ns;  
    above_30_tb <='0'; above_25_tb <='1'; low_level_tb <='0'; wait for 20 ns;  
    above_30_tb <='0'; above_25_tb <='1'; low_level_tb <='1'; wait for 20 ns;  
    above_30_tb <='1'; above_25_tb <='0'; low_level_tb <='0'; wait for 20 ns;  
    above_30_tb <='1'; above_25_tb <='0'; low_level_tb <='1'; wait for 20 ns;  
    above_30_tb <='1'; above_25_tb <='1'; low_level_tb <='0'; wait for 20 ns;  
    above_30_tb <='1'; above_25_tb <='1'; low_level_tb <='1'; wait for 20 ns;  
  end process apply_test_cases;
```

```
end architecture Behavioral;
```

Όταν γράφουμε μόνο τα εσωτερικά σήματα, η σειρά τους καθορίζει έμμεσα την αντιστοίχιση με τα port του component

Ψηφιακό κύκλωμα – Αναπαράσταση σε VHDL – Αρχιτεκτονική – Components

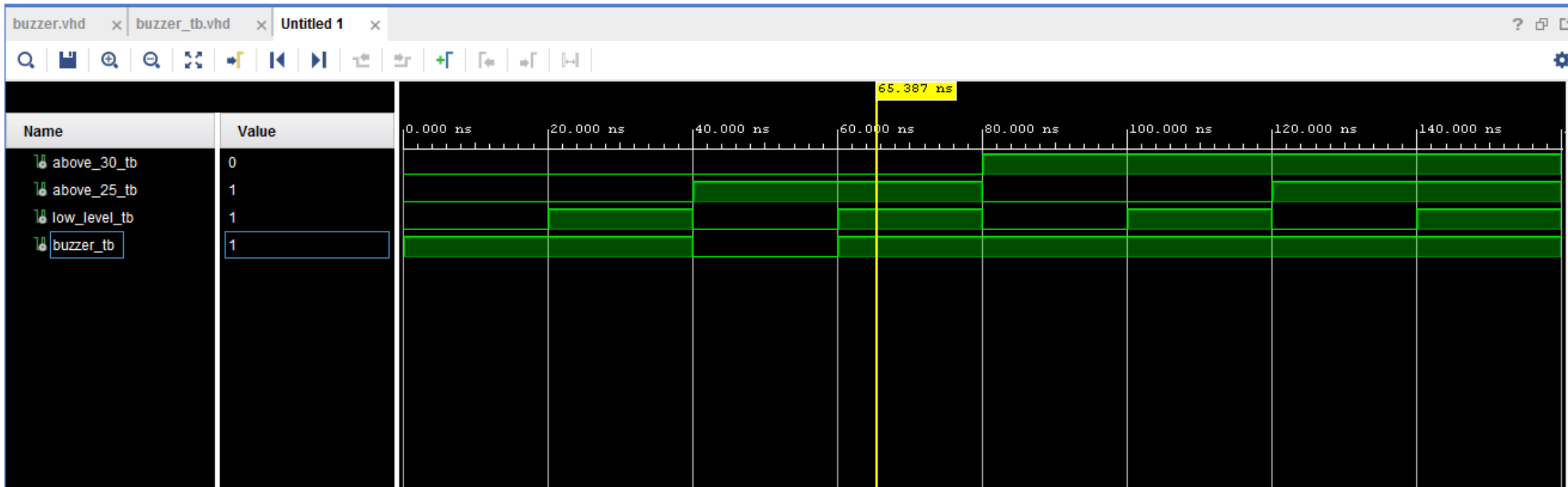
- Ταυτόχρονες εντολές στοιχείων (concurrent_component_statements)

```
label: comp_name port map (signal_name, ..);
```

- **label**: οι μοναδικές ετικέτες των στοιχείων
- **comp_name**: το όνομα του στοιχείου (υποκύκλωμα) που χρησιμοποιείται στην αρχιτεκτονική της οντότητας
- **port_name**: το όνομα του port της οντότητας (component) το οποίο καλούμε.
- **signal_name**: το όνομα του σήματος που συνδέεται στο υποκύκλωμα (comp_name) (εάν είναι πολλά σήματα χωρίζονται με κόμμα)
 - το σήμα είναι μία διασύνδεση που αφορά τη συγκεκριμένη αρχιτεκτονική της οντότητας που χρησιμοποιεί το στοιχείο
 - αντιστοιχεί αμφιμονοσήμαντα στο αντίστοιχο σήμα της δήλωσης των port του υποκυκλώματος (comp_name) (**θέλει προσοχή η σειρά των σημάτων**)

VHDL

Ψηφιακό κύκλωμα – VHDL: Προσομοίωση - Χρονοσειρά



Εγγραφές σε Τμήματα Εργαστηρίων

- Στο διάστημα 21/10 – 30/10 θα γίνει το 1ο ΕΡΓΑΣΤΗΡΙΟ (και **ΔΕΝ** θα γίνει διάλεξη στις 27/10 στην Α2).
- Στο διάστημα 13/10 – 20/10 θα επιλέξετε το εργαστηριακό τμήμα που επιθυμείτε να παρακολουθήσετε
- Το εργαστήριο είναι στο αναγνωστήριο.
- Μπορείτε να **αλλάξετε τμήμα** εάν υπάρχουν θέσεις (απεγγραφή από παλιό, εγγραφή στο νέο). Αυτό μπορεί να γίνει έως και 20/10.
- Σε περίπτωση που χρειαστεί θα ανοίξουν και επιπλέον τμήματα. Θα υπάρξουν θέσεις για όλους.
- Όσοι φοιτητές δεν έχουν κάνει ακόμα εγγραφή στο τμήμα ή απλά δεν έχουν κωδικούς για το eclass, θα μου στείλουν email δηλώνοντας **τουλάχιστον δύο τμήματα** με διαθέσιμες θέσεις.
- Οι μέρες και ώρες που επιλέγετε για το 1ο εργαστήριο ΔΕΝ σας δεσμεύουν για το 2ο εργαστήριο. Μπορείτε, τότε, να επιλέξετε διαφορετική μέρα και ώρα.
- Κατεβάζετε και κάνετε εγκατάσταση το Vivado. **Οδηγίες υπάρχουν στο eclass.** Επίσης, στον κατάλογο **Χρήσιμα**, θα βρείτε έναν οδηγό για να τρέξετε το πρώτο σας πρόγραμμα στο Vivado σε VHDL

Περίληψη

- Παράδειγμα ανάπτυξης εφαρμογής σε VHDL
- Δηλώσεις Οντότητας (entity), Αρχιτεκτονικής (architecture).
- Ports και Εσωτερικά σήματα
- Ταυτόχρονες εντολές
- Προτεραιότητες πράξεων
- Τύποι σημάτων (std_logic, std_logic_vector)
- Προσομοίωση
- Διαβάζετε τις παραγράφους 2.1, 2.6, 4.2.2, 4.2.5, 4.2.8, 4.9 (ΟΧΙ το κομμάτι της VERILOG) από το βιβλίο των Harris.
- Διαβάζετε τις παραγράφους 12.1, 12.2.2, 12.2.3, 12.2.4, 12.4.1, 12.4.2, 12.5, 12.6, 12.7, 12.7.1 από το βιβλίο των Brown-Vranesic.

4^ο Εργαστήριο Λογικής Σχεδίασης

4^η Διάλεξη

Εισαγωγή στο εργαλείο Vivado

Βασιλόπουλος Διονύσης

Ε.ΔΙ.Π Τμήματος Πληροφορικής & Τηλεπικοινωνιών

VHDL - Παράδειγμα

Πραγματικό πρόβλημα

Σε ένα Computer Room υπάρχουν δύο (2) αισθητήρες θερμοκρασίας (**Sensor_1** και **Sensor_2**) και δύο (2) κλιματιστικά (**AirCond_1** και **AirCond_2**). Το πρώτο κλιματιστικό (AirCond_1) λειτουργεί εάν τουλάχιστον ένας από τους δύο αισθητήρες ανιχνεύσουν θερμοκρασία άνω των 35 βαθμών στο computer room. Το δεύτερο κλιματιστικό (AirCond_2) λειτουργεί εάν και οι δύο αισθητήρες ανιχνεύσουν θερμοκρασία άνω των 35 βαθμών στο computer room. Θεωρείστε ότι κάθε αισθητήρας δίνει σήμα ('1') μόνο όταν η θερμοκρασία που ανιχνεύει γίνει μεγαλύτερη των 35 βαθμών (>35). Σε άλλη περίπτωση ο αισθητήρας στέλνει την τιμή '0'. Σχεδιάστε και υλοποιήστε το λογικό κύκλωμα που περιγράφει το ανωτέρω πρόβλημα. Το όνομα του Vivado Project θα είναι Lab_1, της οντότητας θα είναι CR_AC ενώ το όνομα της αρχιτεκτονικής Dataflow.

Σχεδιάστε και υλοποιήστε το λογικό κύκλωμα που περιγράφει το ανωτέρω πρόβλημα.

VHDL - Παράδειγμα

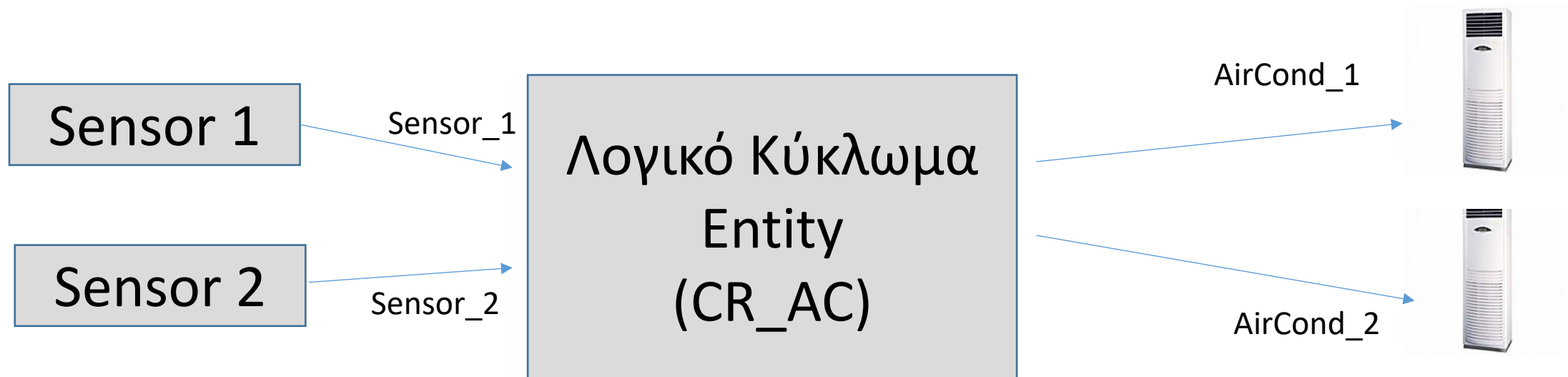
Πραγματικό πρόβλημα – Βήματα επίλυσης

1. Δημιουργία νέου project
2. Δημιουργία Entity – Εντοπισμός Input/Output του συστήματος
3. Δημιουργία Architecture – Θα έχετε (τουλάχιστον) τόσες εντολές όσες είναι και οι έξοδοι του συστήματος. Κάθε μία εντολή αντιστοιχεί σε μία έξοδο.
4. Δημιουργία RTL αναπαράστασης
5. Προσομοίωση

(Αναλυτικές οδηγίες στον Οδηγό Vivado στο eclass)

VHDL - Παράδειγμα

Απλοποιημένη μορφή κυκλώματος



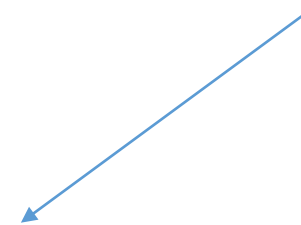
VHDL - Παράδειγμα

Περιγραφή Οντότητας (entity)

```
library IEEE;  
use IEEE.STD_LOGIC_1164.all;
```

```
entity CR_AC is  
port (  
    Sensor_1      :in STD_LOGIC;  
    Sensor_2      :in STD_LOGIC;  
    AirCond_1     :out STD_LOGIC;  
    AirCond_2     :out STD_LOGIC  
);  
end entity CR_AC;
```

Περιγραφή Οντότητας



VHDL - Παράδειγμα

Λογική κατανόηση της άσκησης

AirCond_1: Ανάβει όταν δίνει σήμα τουλάχιστον ένας αισθητήρας =>
 Ανάβει όταν δίνει σήμα οποιοσδήποτε αισθητήρας

AirCond_2: Ανάβει όταν δίνουν σήμα και οι δύο αισθητήρες

VHDL - Παράδειγμα

Πίνακας Αληθείας

Είσοδοι (Inputs)		Έξοδοι (outputs)	
Sensor_1	Sensor_2	AirCond_1	AirCond_2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

VHDL - Παράδειγμα

Πίνακας Αληθείας - Ελαχιστόροι

Είσοδοι (Inputs)		Έξοδοι (outputs)	
Sensor_1	Sensor_2	AirCond_1	AirCond_2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

AirCond_1= **Sensor_1'*****Sensor_2**+Sensor1*Sensor_2'+
Sensor_1* **Sensor_2**=
Sensor_2+Sensor1*Sensor_2'=
Sensor_2+**Sensor1**

AirCond_2= **Sensor1*****Sensor_2**

***=AND**

+=OR

VHDL - Παράδειγμα

Υλοποίηση Αρχιτεκτονικής (Dataflow)

architecture Dataflow of CR_AC is
begin

```
AirCond_1<=Sensor_1 or Sensor_2;  
AirCond_2<=Sensor_1 and Sensor_2;
```

end architecture Dataflow;

Αρχιτεκτονική
Περιγραφή της λειτουργικότητας
που προσφέρει το λογικό
κύκλωμα

Εκτελούνται ταυτόχρονα

VHDL - Παράδειγμα

Τελική έκδοση προγράμματος

```
library IEEE;  
use IEEE.STD_LOGIC_1164.ALL;
```

```
Entity CR_AC is  
Port (  
  Sensor_1   :in std_logic;  
  Sensor_2   :in std_logic;  
  AirCond_1  :out std_logic;  
  AirCond_2  :out std_logic  
);
```

```
end entity CR_AC;
```

```
architecture Dataflow of CR_AC is
```

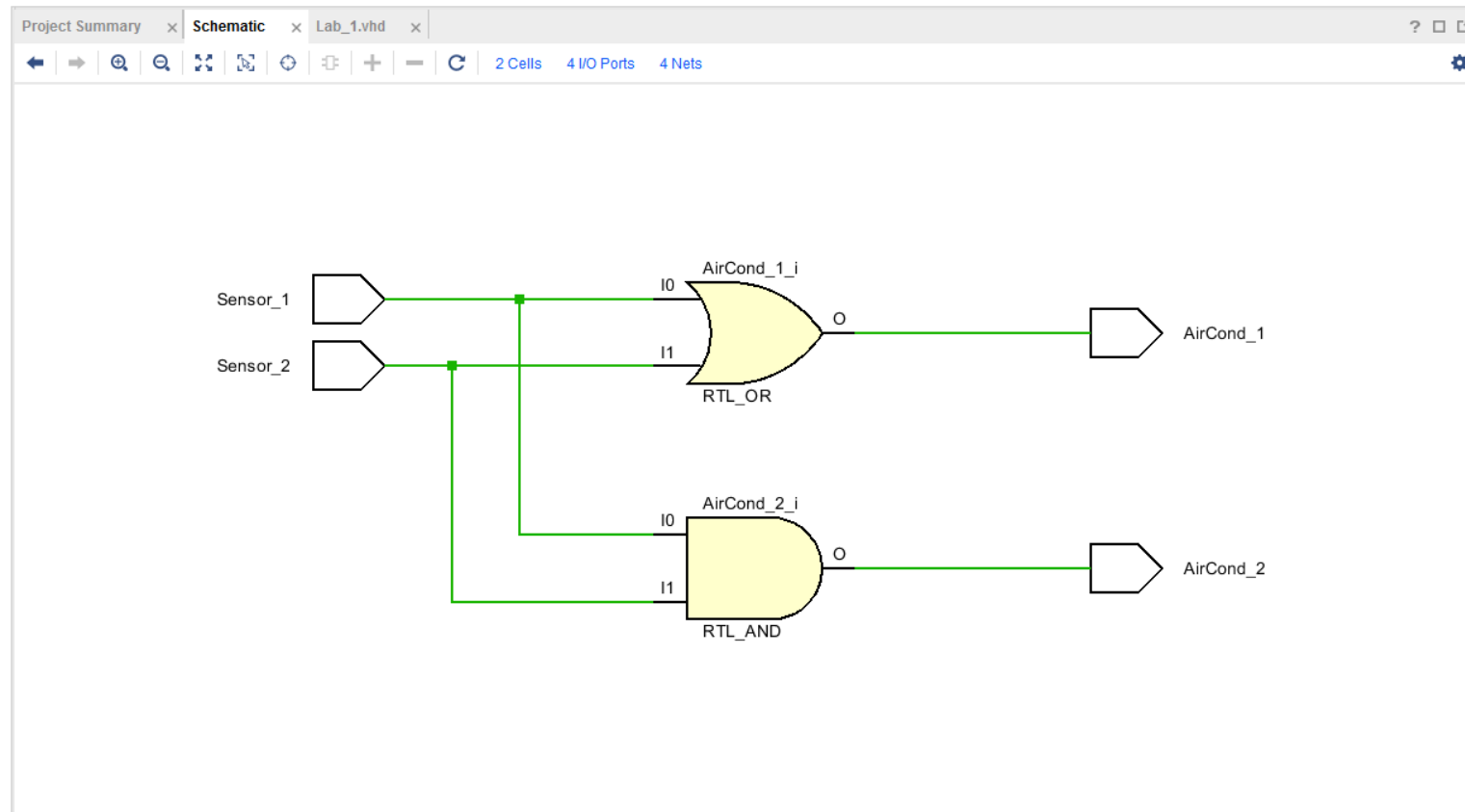
```
begin
```

```
AirCond_1<=Sensor_1 or Sensor_2;  
AirCond_2<=Sensor_1 and Sensor_2;
```

```
end architecture Dataflow;
```

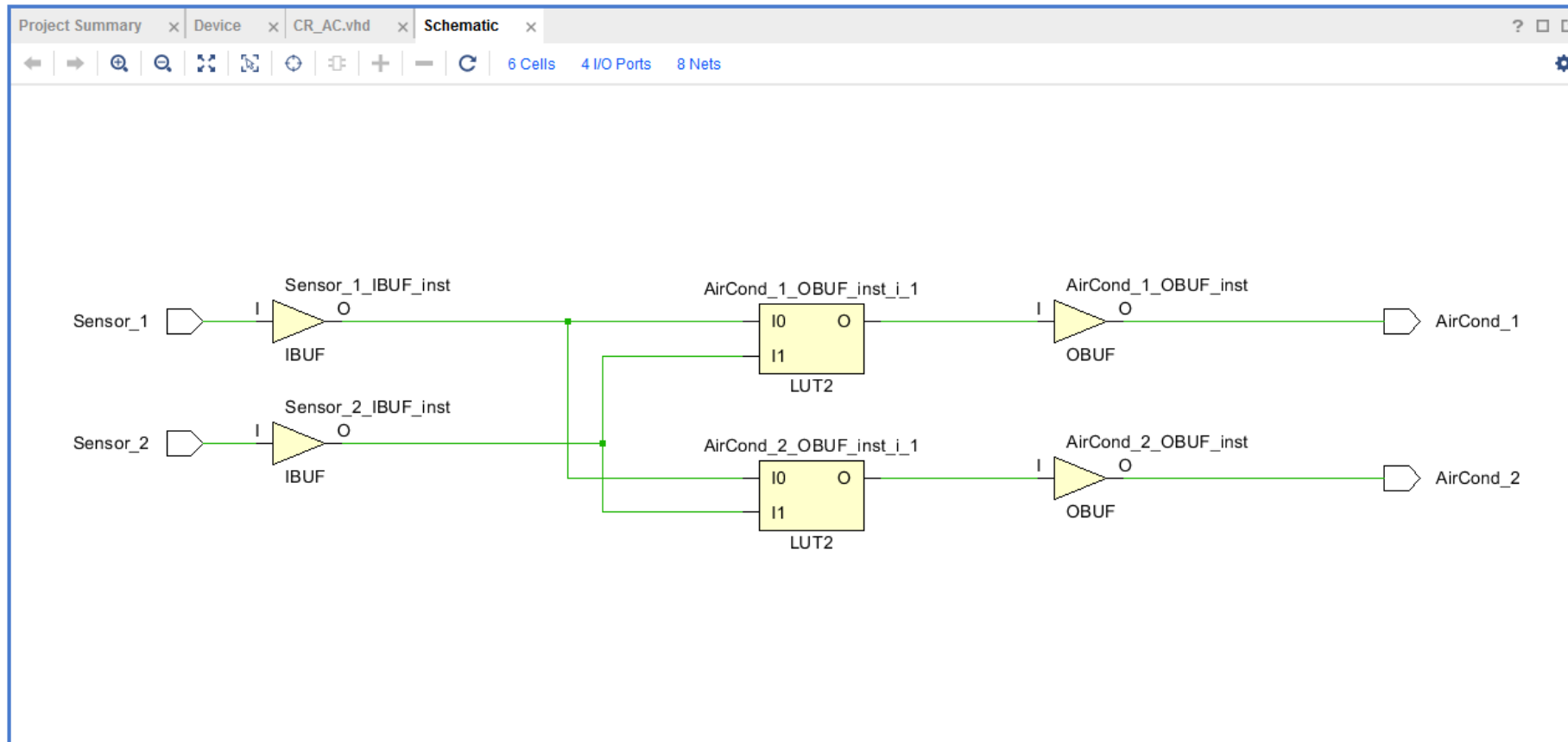
VHDL - Παράδειγμα

Λογικό κύκλωμα: Αναπαράσταση RTL



VHDL - Παράδειγμα

Λογικό κύκλωμα: Φάση Σύνθεσης – LUT(LookUp Table), υλοποιούν Πίνακες Αληθείας (προγραμματιζόμενα μέρη της κάρτας FPGA)

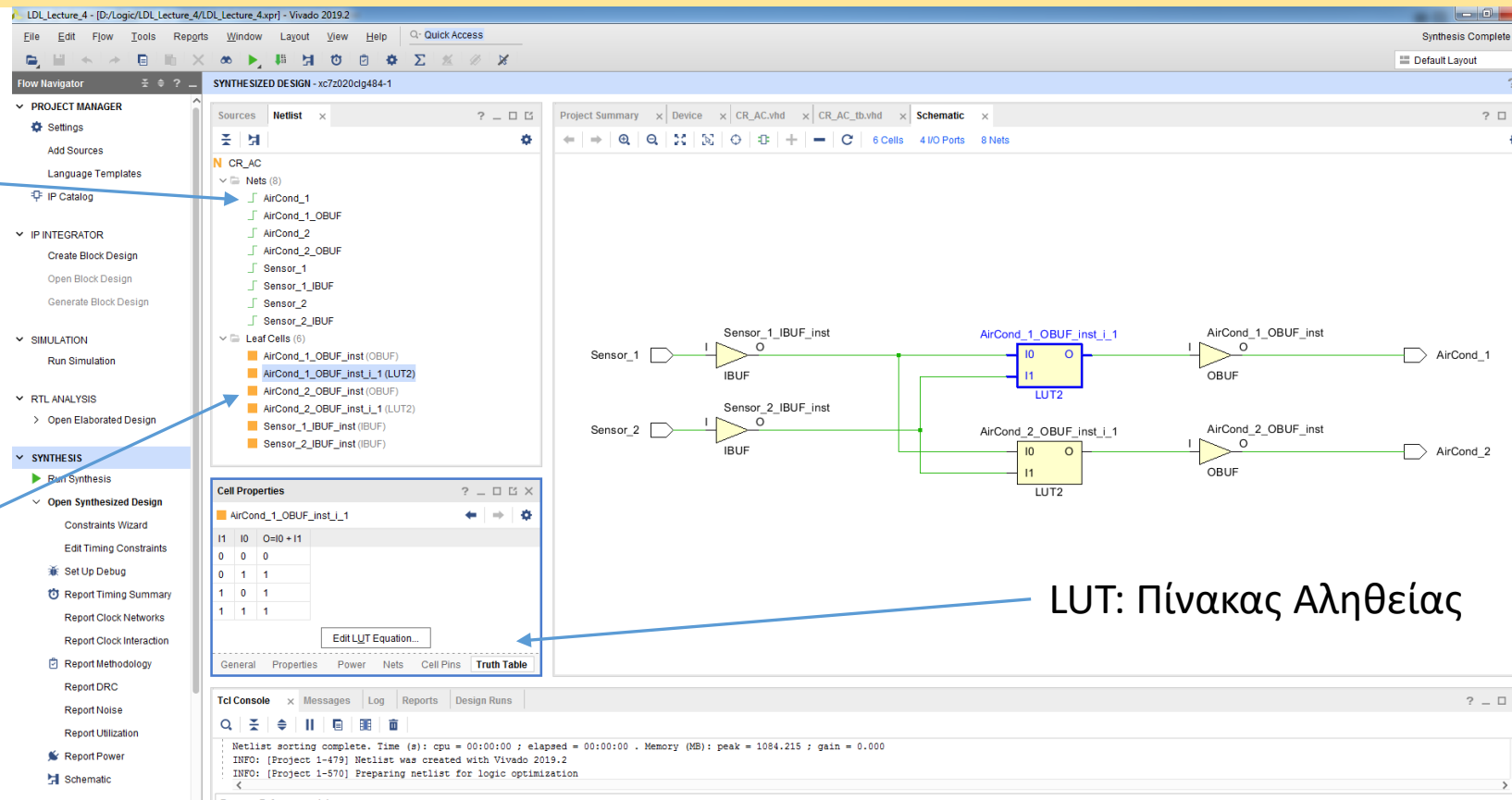


VHDL - Παράδειγμα

Λογικό κύκλωμα: Φάση Σύνθεσης – LUT(LookUp Table), υλοποιούν Πίνακες Αληθείας (προγραμματιζόμενα μέρη της κάρτας FPGA)

Nets->Σύρματα

Cells->Ψηφιακά
στοιχεία

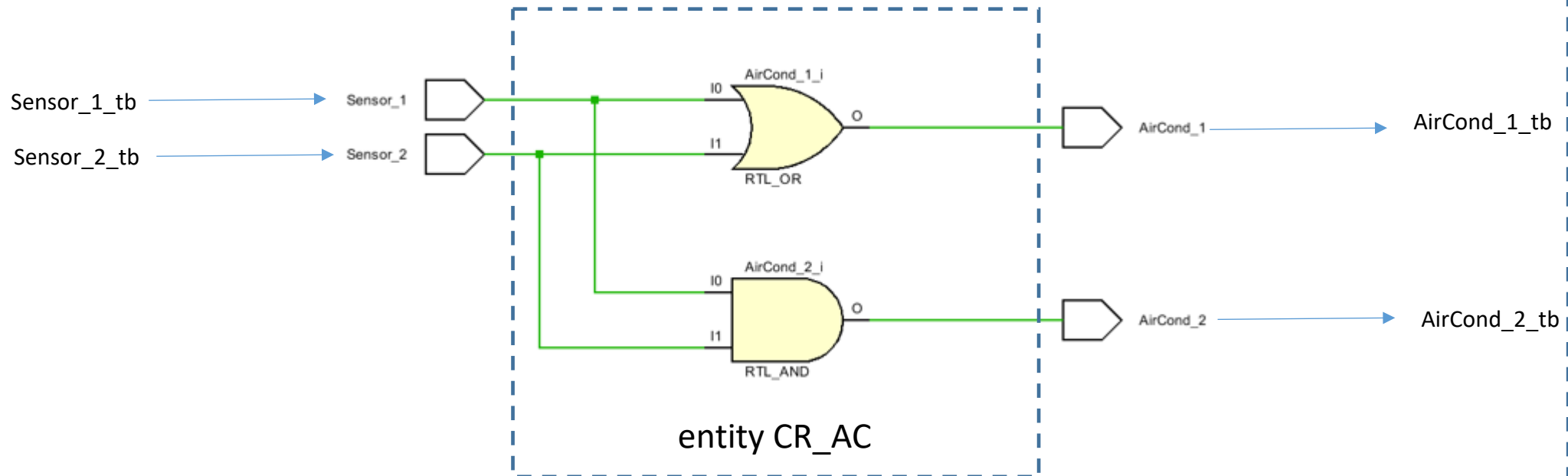


LUT: Πίνακας Αληθείας

VHDL – Παράδειγμα σε Vivado

Ψηφιακό κύκλωμα - VHDL: Προσομοίωση

entity CR_AC_tb (οντότητα προσομοίωσης)



VHDL - Παράδειγμα

Προσομοίωση (Πρόγραμμα testbench)

```
library IEEE; use IEEE.STD_LOGIC_1164.ALL;
```

```
entity CR_AC_tb is  
end CR_AC_tb;
```

```
architecture Behavioral of CR_AC_tb is
```

```
component CR_AC is  
Port (  
Sensor_1 :in std_logic;  
Sensor_2 :in std_logic;  
AirCond_1 :out std_logic;  
AirCond_2 :out std_logic);  
end component CR_AC;
```

```
-- Internal Signals: One for each port  
signal Sensor_1_tb :std_logic; signal Sensor_2_tb :std_logic;  
signal AirCond_1_tb :std_logic; signal AirCond_2_tb :std_logic;
```

```
begin
```

```
-- create entity into testbench entity  
test_entity: CR_AC port map (Sensor_1=>Sensor_1_tb,  
Sensor_2=>Sensor_2_tb, AirCond_1=>AirCond_1_tb,  
AirCond_2=>AirCond_2_tb);
```

```
enter_test_values: process is  
begin
```

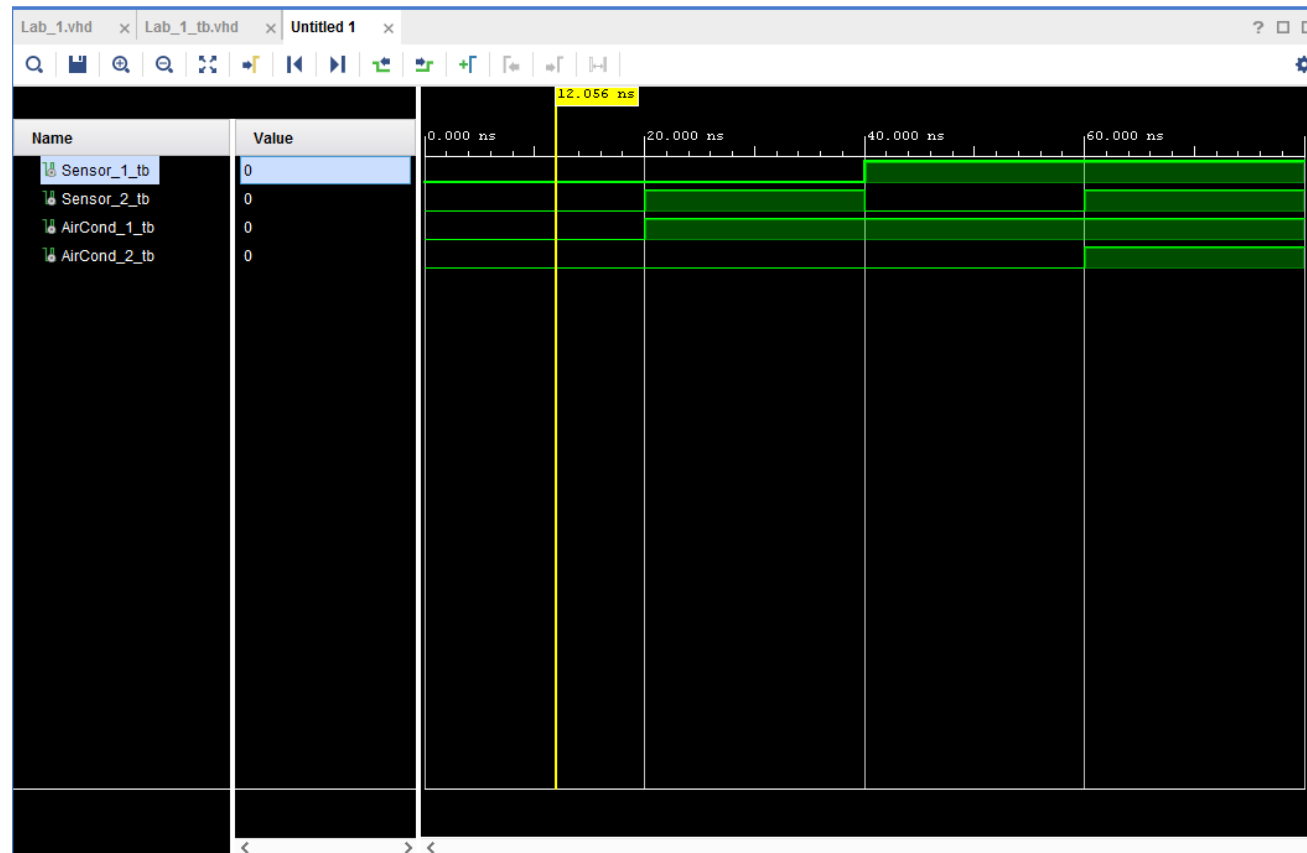
```
Sensor_1_tb<='0';Sensor_2_tb<='0';wait for 20 ns;  
Sensor_1_tb<='0';Sensor_2_tb<='1';wait for 20 ns;  
Sensor_1_tb<='1';Sensor_2_tb<='0';wait for 20 ns;  
Sensor_1_tb<='1';Sensor_2_tb<='1';wait for 20 ns;
```

```
end process enter_test_values;
```

```
end architecture Behavioral;
```

VHDL - Παράδειγμα

Προσομοίωση Behavioral (Χρονοσειρά)



Περίληψη

- Ανάπτυξη βήμα-βήμα μιας απλής εφαρμογής στο Vivado
- RTL->Synthesis
- Προσομοίωση
- LUT
- Διαβάζετε τον οδηγό Vivado