dscal
DIGITAL SYSTEMS & COMPUTER ARCHITECTURE LABORATORY

Εργαστήριο Λογικής Σχεδίασης

2η Διάλεξη

Εισαγωγή στη VHDL και το εργαλείο Vivado

Βασιλόπουλος Διονύσης

Ε.ΔΙ.Π Τμήματος Πληροφορικής & Τηλεπικοινωνιών

ΤΜΗΜΑ
ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

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
- Φυσική επαλήθευση (physical verification)
  - Το φυσικό κύκλωμα συνεχίζει να ικανοποιεί τους περιορισμούς
  - Χρησιμοποιεί καλύτερες εκτιμήσεις των καθυστερήσεων

Ψηφιακά Συστήματα

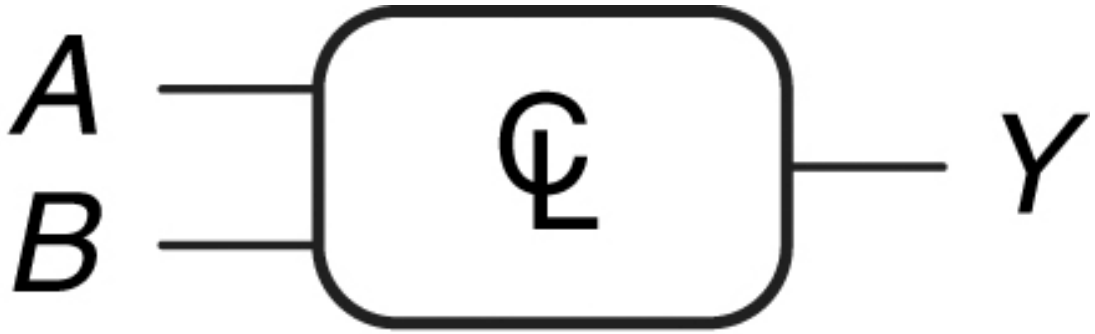
Ψηφιακό κύκλωμα – Φυσική υλοποίηση

- 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

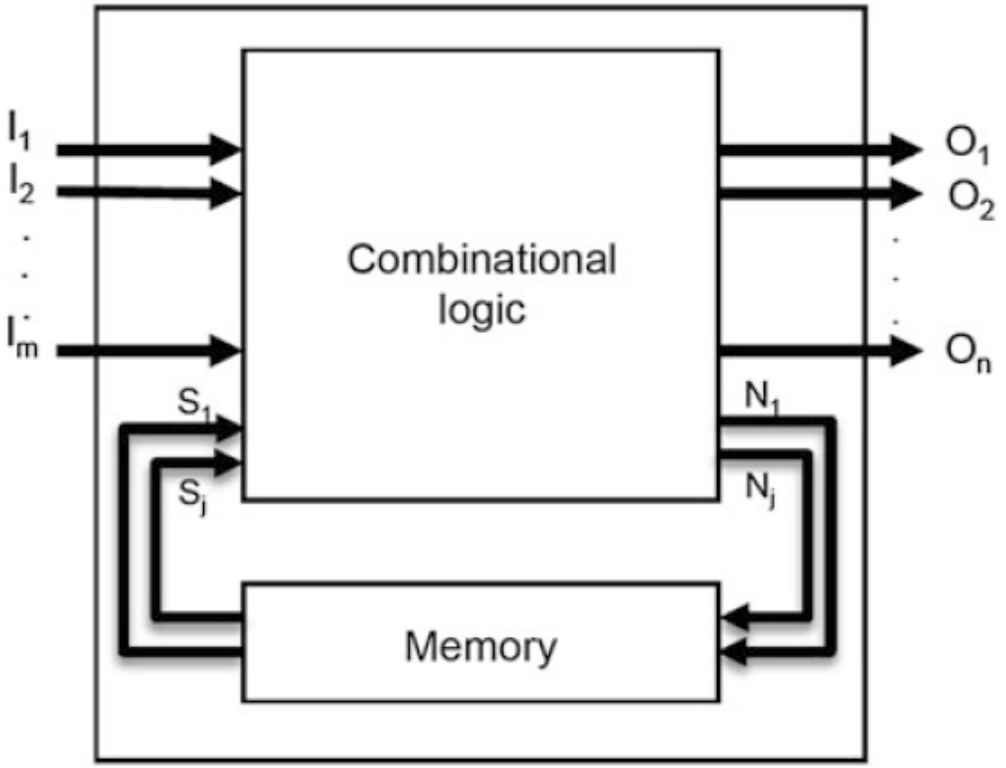


$$Y = F(A, B)$$

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).
  - **<u>Σύγχρονα</u>**: Η συμπεριφορά τους ορίζεται από τις τιμές των εξόδων σε διακριτές στιγμές του χρόνου. Υπάρχει σήμα συγχρονισμού (ρολόι/clock-CLK)
  - **<u>Ασύγχρονα</u>**: Οι τιμές των εξόδων αλλάζουν ανά πάσα στιγμή.



HDL

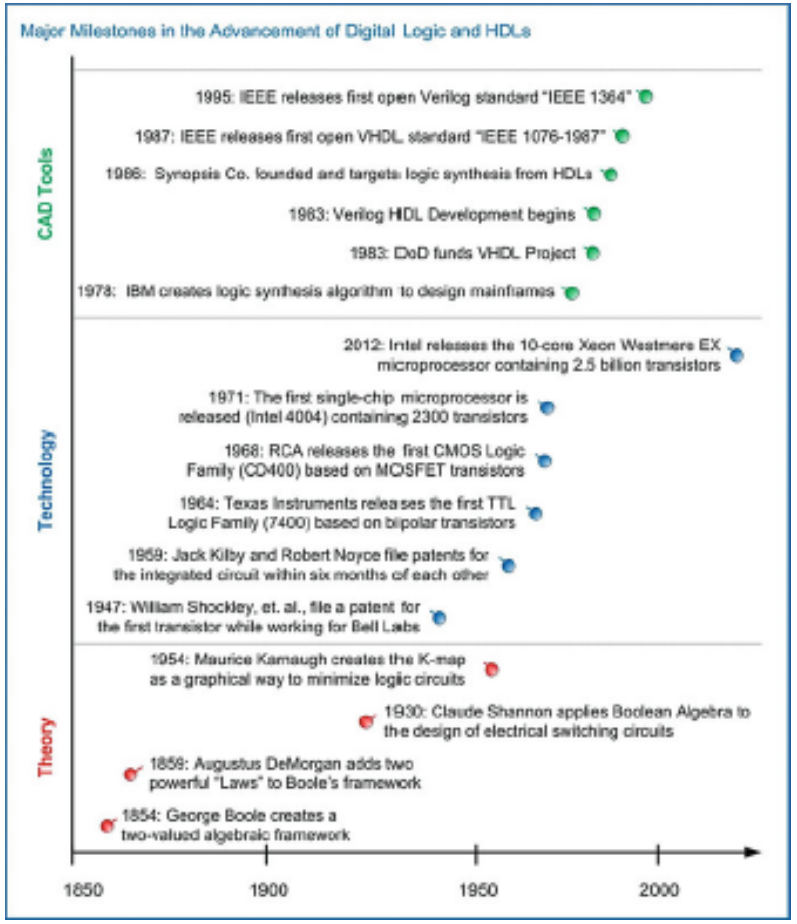
- Hardware Description Language (HDL)
  - Μια γλώσσα για την μοντελοποίηση της συμπεριφοράς και της δομής των ψηφιακών συστημάτων
- Electronic Design Automation (EDA) using HDL
  - Σχεδίαση ηλεκτρονικών κυκλωμάτων με χρήση εργαλείων CAD (computer-aided design)
  - Εισαγωγή σχεδίασης (design entry)
    - 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 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 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.

HDL-VHDL



Εξέλιξη γλωσσών HDL και Λογικής Σχεδίασης

HDL

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

HDL

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 1000 Στήλη των 100 Στήλη των 10 Στήλη των 1

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

Έξι Χιλιάδες — Πέντε Εκατοντάδες — Εννέα Δεκάδες — Οκτώ Μονάδες

- Δυαδικό σύστημα αναπαράστασης αριθμών (0-1)

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

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

Μια Οκτάδα — Μια Τετράδα — Μηδέν Δυάδες — Μια 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(δεκαδικό) (A)

Εφαρμόζουμε αριστερή ολίσθηση και γεμίζουμε δεξιά με το 0

10**0** =4(δεκαδικό) (B) = 2*(A)

10**00** =8(δεκαδικό) (C) = 2*(B)=4*(A)

1000 =8(δεκαδικό) (A)

Εφαρμόζουμε δεξιά ολίσθηση διαγράφοντας το πιο δεξί ψηφίο

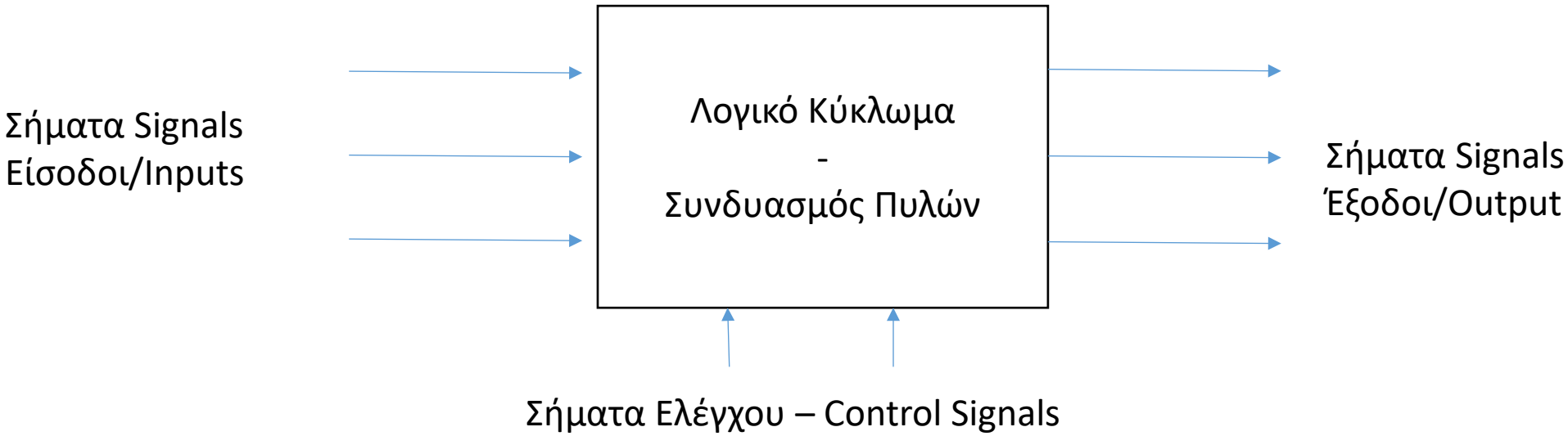
100 =4(δεκαδικό) (B) = (A)/2

10 =2(δεκαδικό) (C)= (B)/2=(A)/4

Λογικές Πύλες

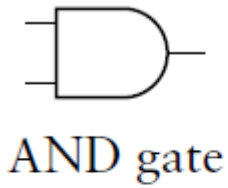
Όλοι οι είσοδοι και έξοδοι είναι σήματα με πιθανές τιμές 0 ή 1

Λογικές Πύλες

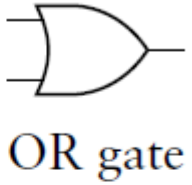


Λογικές Πύλες – Πίνακας Αληθείας (1/2)

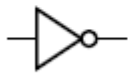
- Σε κάθε Πύλη ή Κύκλωμα αντιστοιχεί ένας Πίνακας Αληθείας (Truth Table).
- Ο Πίνακας Αληθείας καθορίζει την τιμή εξόδου για κάθε συνδυασμό εισόδων στην Πύλη ή το Κύκλωμα



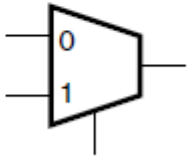
AND gate



OR gate



inverter



multiplexer

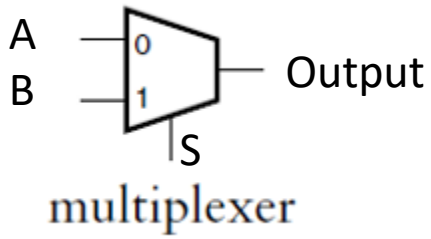
x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

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



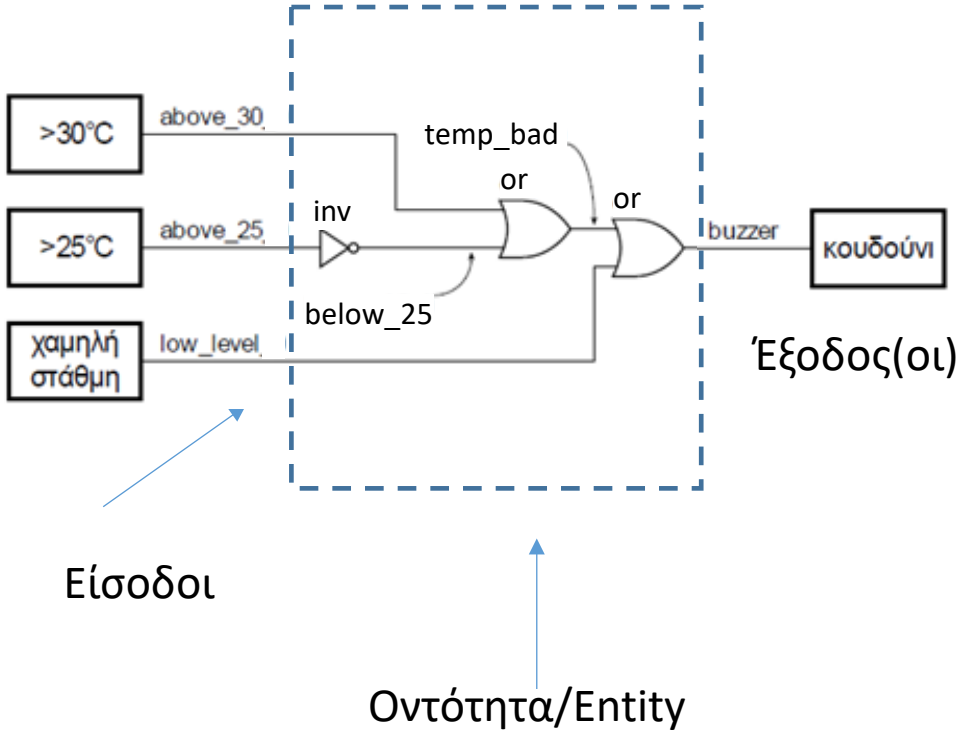
S	Output
0	A
1	B

Αν A=0, B=1, S=1, Output= ;
Αν A=1, B=0, S=1, Output= ;

VHDL - Vivado

Ψηφιακό κύκλωμα - Παράδειγμα

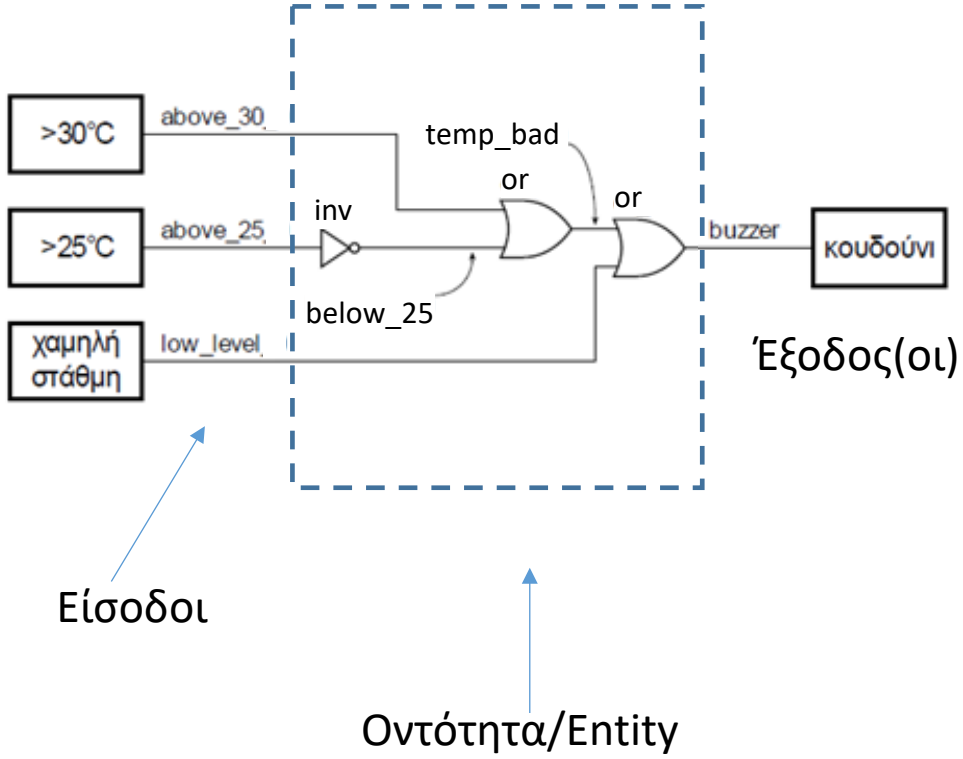
Παράδειγμα: Εργοστάσιο έχει δοχείο επεξεργασίας υγρών. Το δοχείο πρέπει α)να έχει θερμοκρασία μεταξύ 25 και 30 βαθμών και β) η στάθμη του πρέπει να είναι πάνω από ένα επίπεδο. Σε περίπτωση που το α) ή το β) δεν ικανοποιούνται πρέπει να ενεργοποιηθεί ένα κουδούνι. Στη διάθεσή μας έχουμε αισθητήρες (θερμόμετρα) 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:
Α)Η θερμοκρασία ΕΙΝΑΙ πάνω από 30
ή
Β)Η θερμοκρασία ΕΙΝΑΙ κάτω από 25 =>
Η θερμοκρασία ΔΕΝ ΕΊΝΑΙ πάνω από 25.
ή
Γ)Η στάθμη είναι χαμηλή



VHDL - Vivado

Ψηφιακό κύκλωμα - Παράδειγμα

Το κουδούνι χτυπάει όταν:
Α)Η θερμοκρασία ΕΙΝΑΙ πάνω από 30
ή
Β)Η θερμοκρασία ΕΙΝΑΙ κάτω από 25 =>
Η θερμοκρασία ΔΕΝ ΕΊΝΑΙ 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

VHDL - Vivado

Ψηφιακό κύκλωμα - Παράδειγμα

Το κουδούνι χτυπάει όταν:
Α)Η θερμοκρασία ΕΙΝΑΙ πάνω από 30
ή
Β)Η θερμοκρασία ΕΙΝΑΙ κάτω από 25 =>
Η θερμοκρασία ΔΕΝ ΕΊΝΑΙ 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

VHDL - Vivado

Ψηφιακό κύκλωμα - Παράδειγμα

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

Ισοδύναμο σε Άλγεβρα Boole

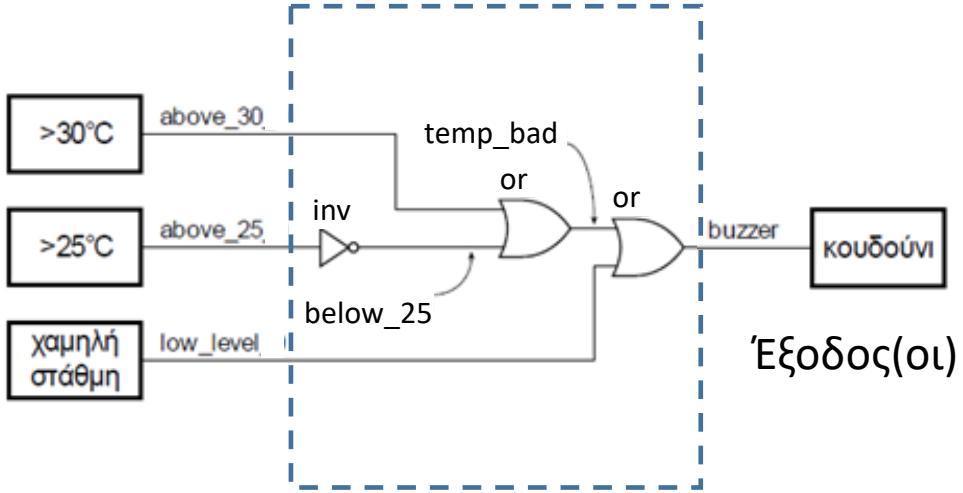
Buzzer=Above_30 or not Above_25 or Low_level

Ισοδύναμο σε VHDL

```
Buzzer<=Above_30 or not Above_25 or Low_level;
```

VHDL - Vivado

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

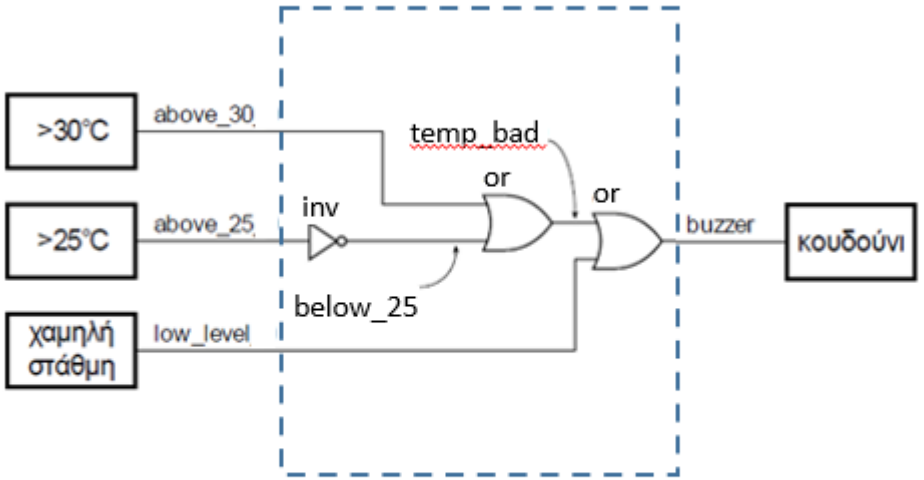



```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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)

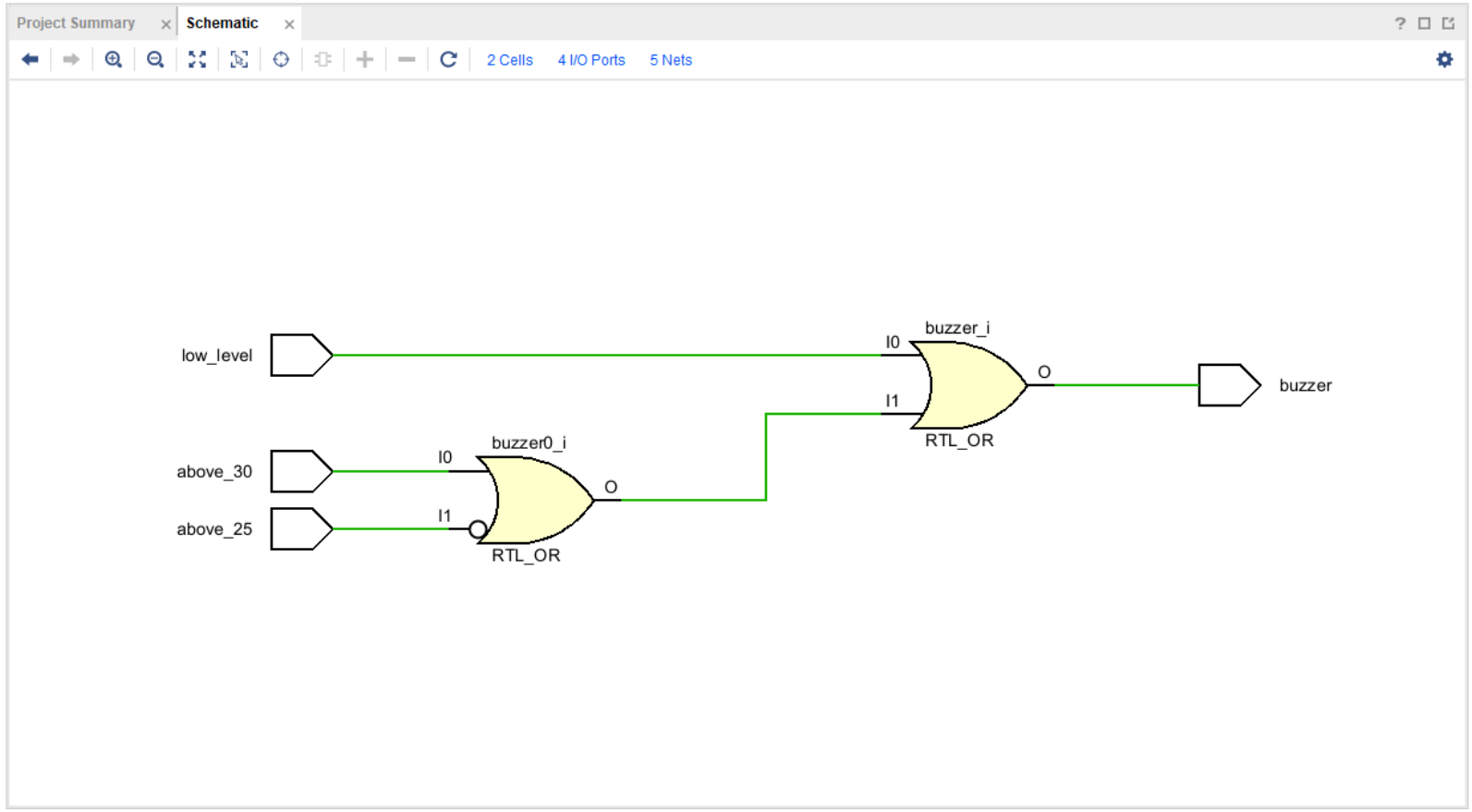


Αρχιτεκτονική
Περιγραφή της λειτουργικότητας που προσφέρει το λογικό κύκλωμα

```
architecture Behavioral of buzzer is

begin

buzzer<= above_30 or not above_25 or low_level;

end architecture Behavioral;
```

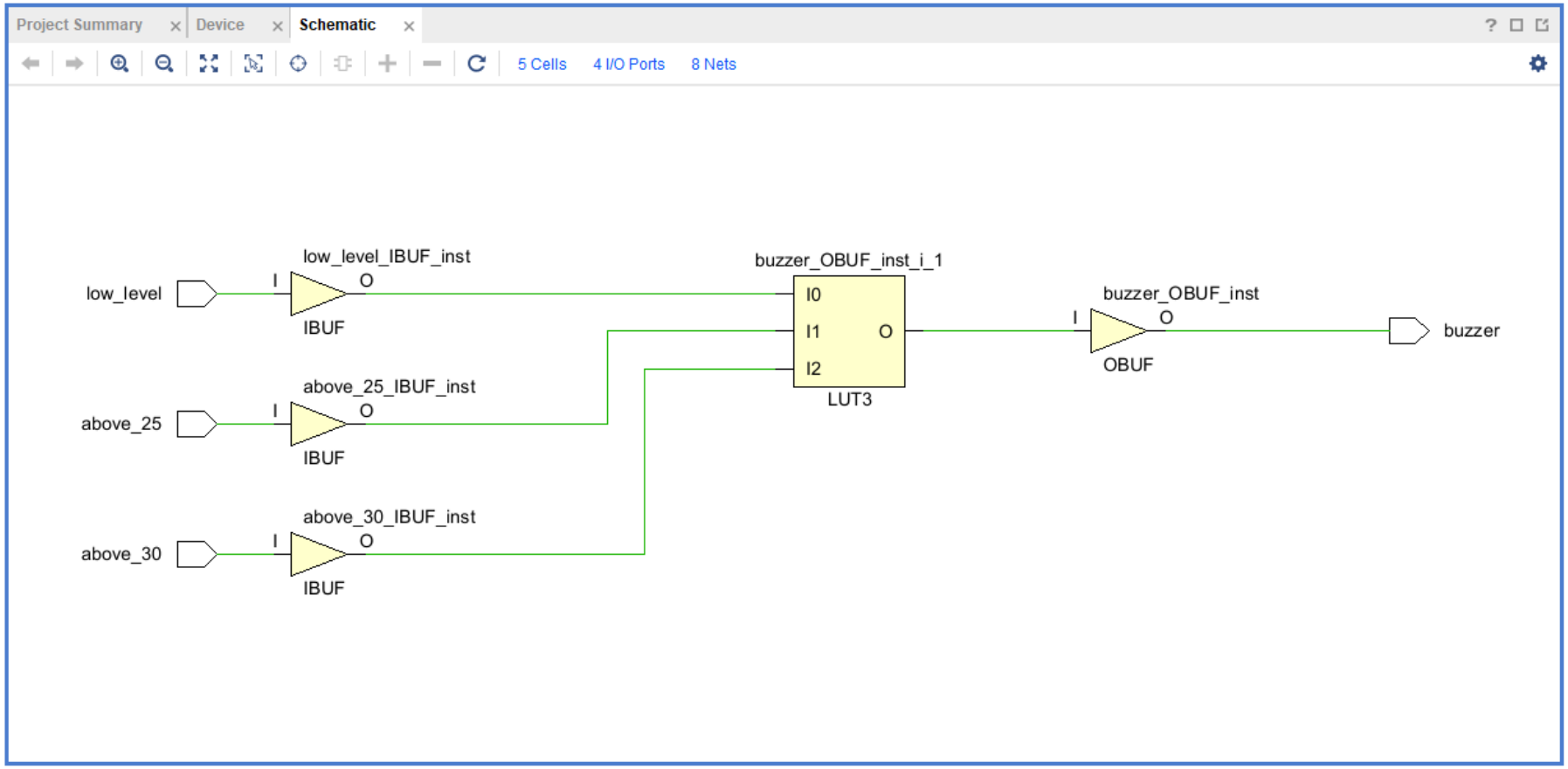
VHDL - Vivado

Ψηφιακό κύκλωμα – VHDL: RTL Αναπαράσταση



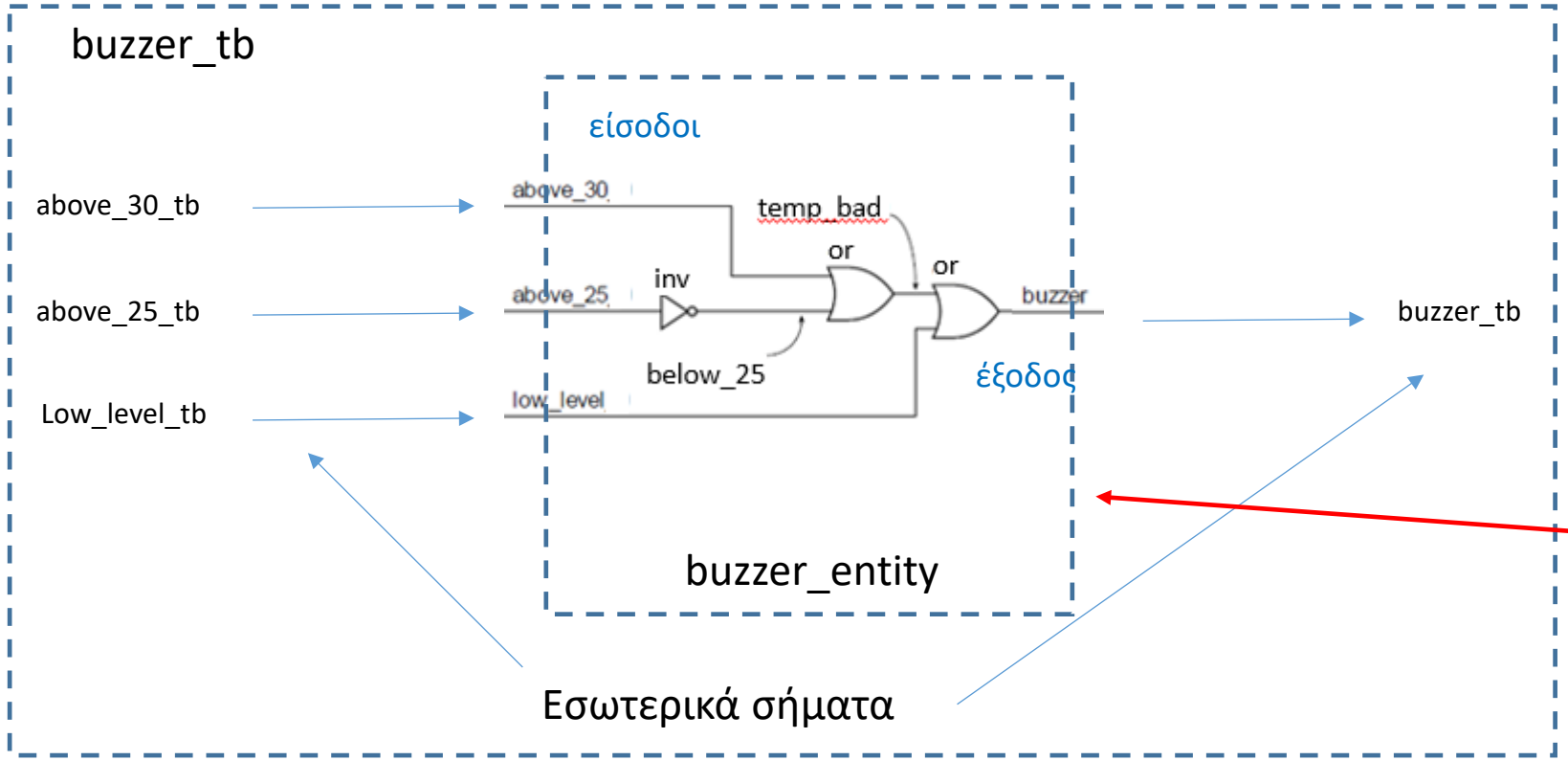
VHDL - Vivado

Ψηφιακό κύκλωμα – VHDL: Σύνθεση



VHDL - Vivado

Ψηφιακό κύκλωμα – VHDL: Προσομοίωση



Component=Μια οντότητα
Που την έχουμε ήδη ορίσει
στο πρόγραμμά μας

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

VHDL - Vivado

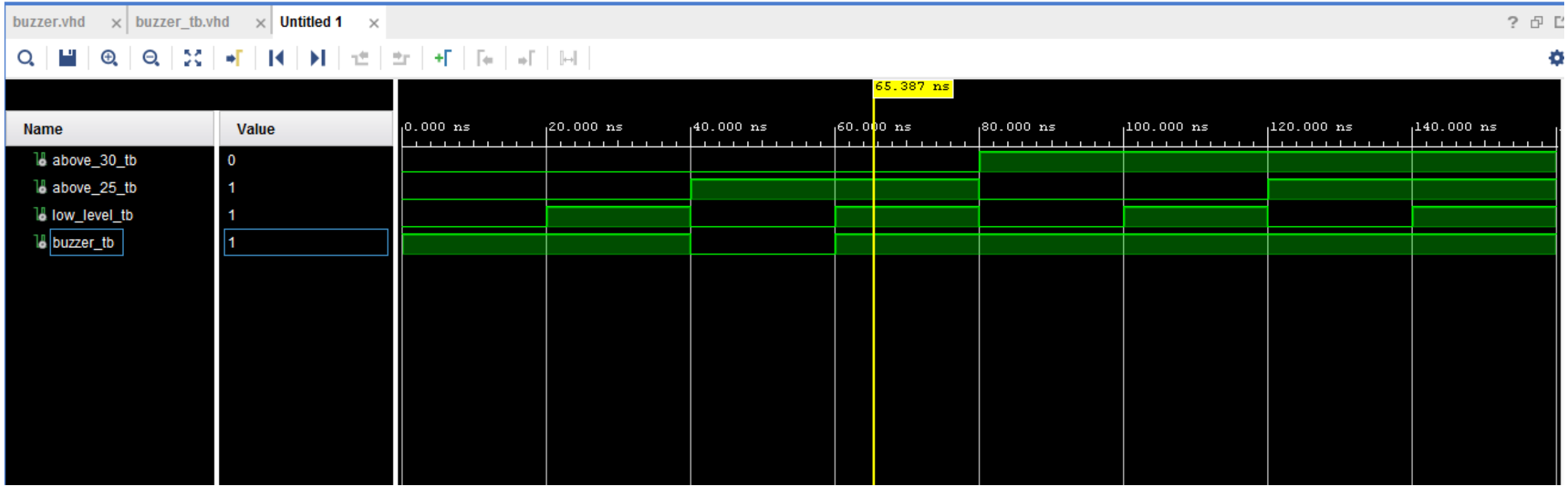
Ψηφιακό κύκλωμα – VHDL: Προσομοίωση - Testbench

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

entity buzzer_tb is
end buzzer_tb ;

architecture Beh_tb 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);

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 Beh_tb;
```

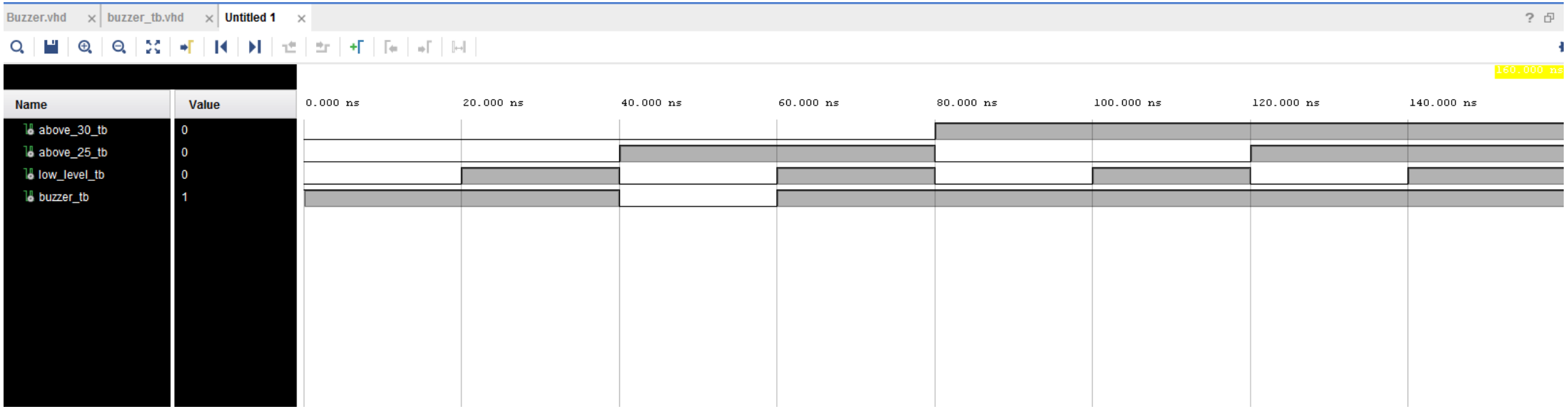
VHDL - Vivado

Ψηφιακό κύκλωμα – VHDL: Προσομοίωση - Χρονοσειρά



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 από το βιβλίο των Harris.