dscal
DIGITAL SYSTEMS & COMPUTER ARCHITECTURE LABORATORY

Εργαστήριο Λογικής Σχεδίασης

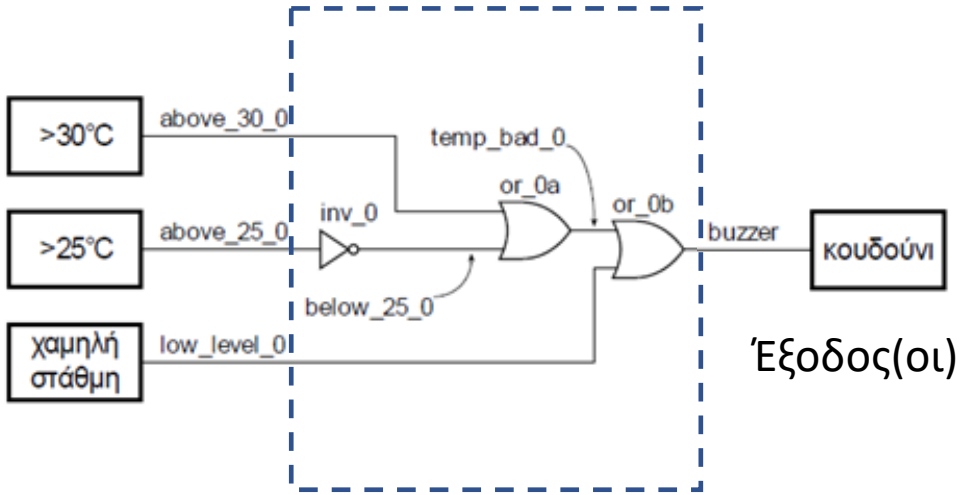
Εισαγωγή στη VHDL και το εργαλείο Vivado

Βασιλόπουλος Διονύσης

Ε.ΔΙ.Π Τμήματος Πληροφορικής & Τηλεπικοινωνιών

VHDL

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



Είσοδοι

Έξοδος(οι)

Οντότητα/Entity

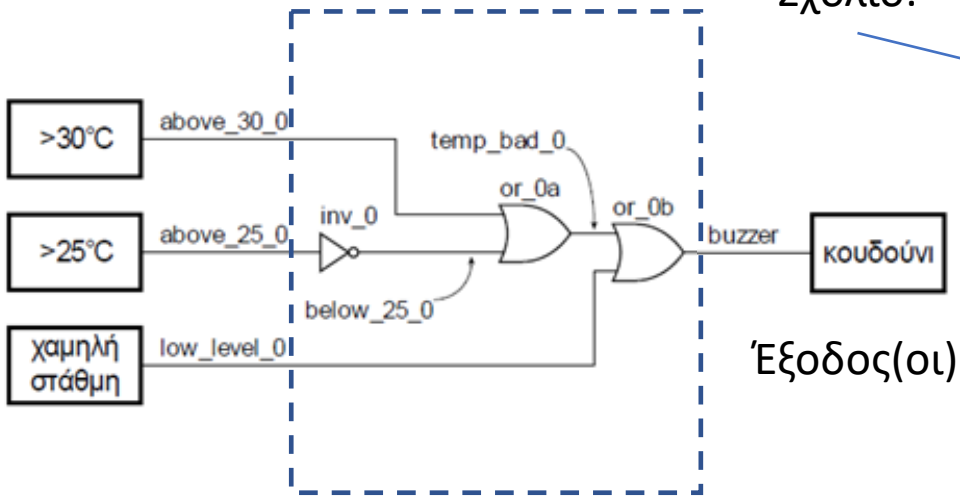
Υποχρεωτικές Βιβλιοθήκες

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

entity buzzer is

port (
above_25_0: in std_logic;
above_30_0: in std_logic;
low_level_0: in std_logic;
buzzer      : out std_logic );

end entity buzzer;
```

Περιγραφή Οντότητας

VHDL

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



Σχόλιο: --

Χωρίς τις Βιβλιοθήκες

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

entity buzzer is

port (
above_25_0: in bit;
above_30_0: in bit;
low_level_0: in bit;
buzzer      : out bit);

end entity buzzer;
```

Περιγραφή Οντότητας

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

VHDL

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;
```

VHDL

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)

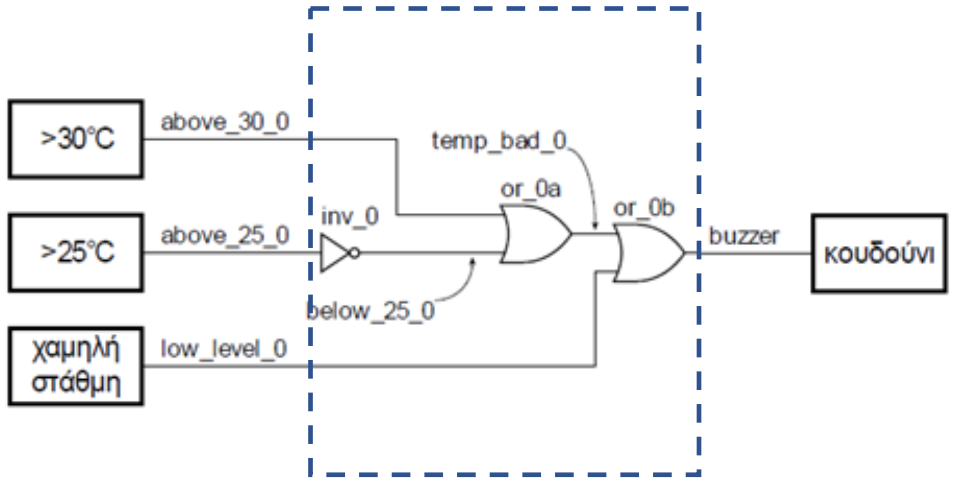
VHDL

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

Ψηφιακό κύκλωμα – VHDL: Architecture (Dataflow)

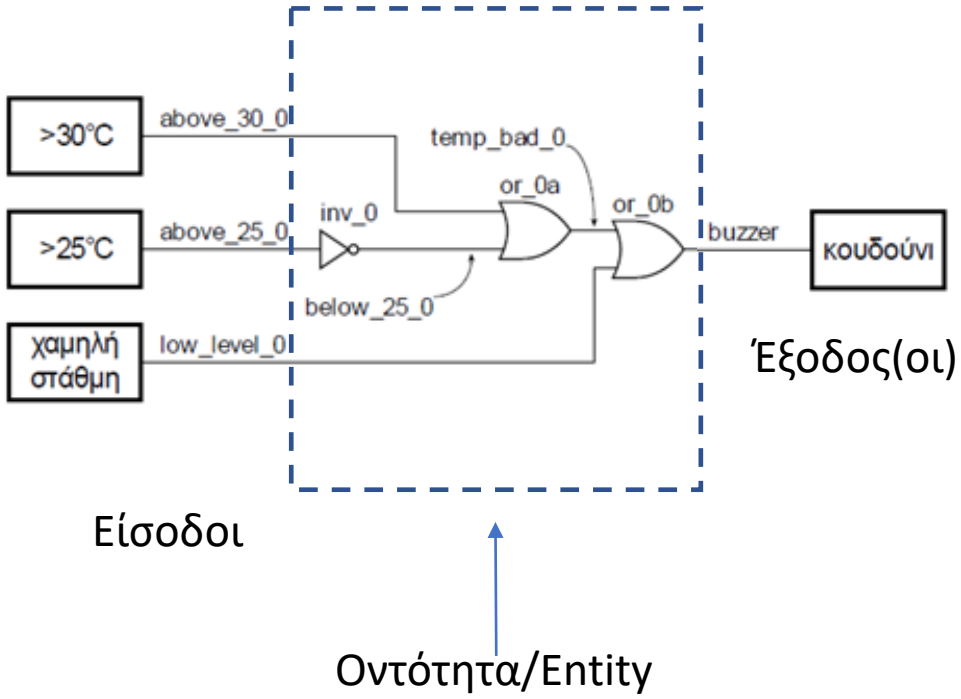



```
architecture Dataflow of buzzer is

begin

buzzer<= low_level_0 or (above_30_0 or not above_25_0);

end architecture buzzer;
```

Αρχιτεκτονική
Περιγραφή της λειτουργικότητας που προσφέρει το λογικό κύκλωμα

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

Ψηφιακό κύκλωμα – VHDL: Entity (Εσωτερικά σήματα)



Εσωτερικά σήματα

temp_bad_0

-

below_25_0

VHDL

Ψηφιακό κύκλωμα – Αναπαράσταση σε VHDL – Architecture

```
architecture arch_name of entity_name is
    signal signal_name: signal_type;
    component comp_name
        port (
            signal_name: mode signal_type;
            ...
            signal_name: mode signal_type);
    end component;
    ...
begin
    concurrent_component_statement;
    ...
    concurrent_component_statement;
end architecture arch_name;
```

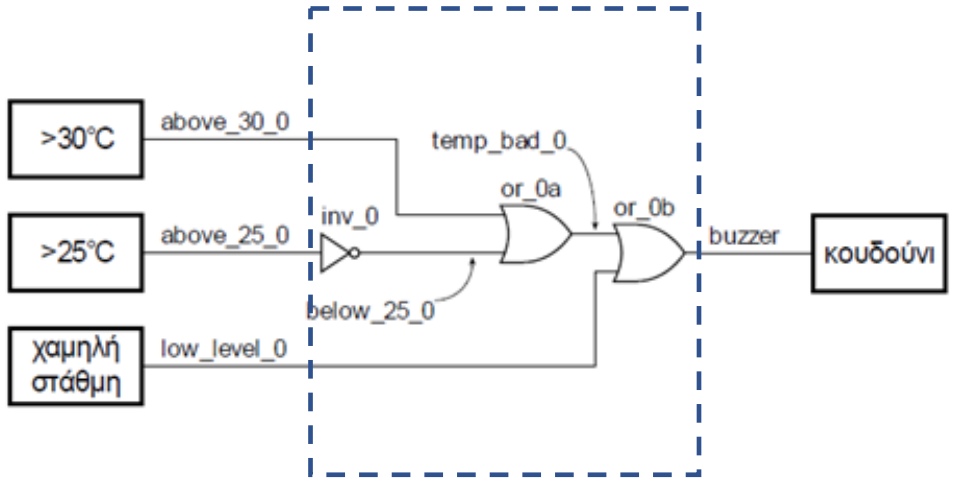
VHDL

Ψηφιακό κύκλωμα – Αναπαράσταση σε VHDL – Architecture

- **arch_name**: το όνομα της αρχιτεκτονικής
- **entity_name**: το όνομα της οντότητας
- **comp_name**: το όνομα του **στοιχείου (component)** που χρησιμοποιείται στην αρχιτεκτονική της οντότητας.
  - Το στοιχείο είναι μία ήδη προκαθορισμένη οντότητα.
- **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)

VHDL

Ψηφιακό κύκλωμα – VHDL: Architecture (Dataflow)



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

```
architecture Dataflow of buzzer is
signal temp_bad_0, below_25_0: std_logic;
begin

below_25_0<=not above_25_0;
temp_bad_0<=above_30_0 or below_25_0;
buzzer<= temp_bad_0 or low_level_0;

end architecture buzzer;
```

concurrent statements

Τα εσωτερικά σήματα μπορούν να είναι και αριστερά και δεξιά

VHDL

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

VHDL

Ψηφιακό κύκλωμα – Αναπαράσταση σε VHDL – Architecture

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

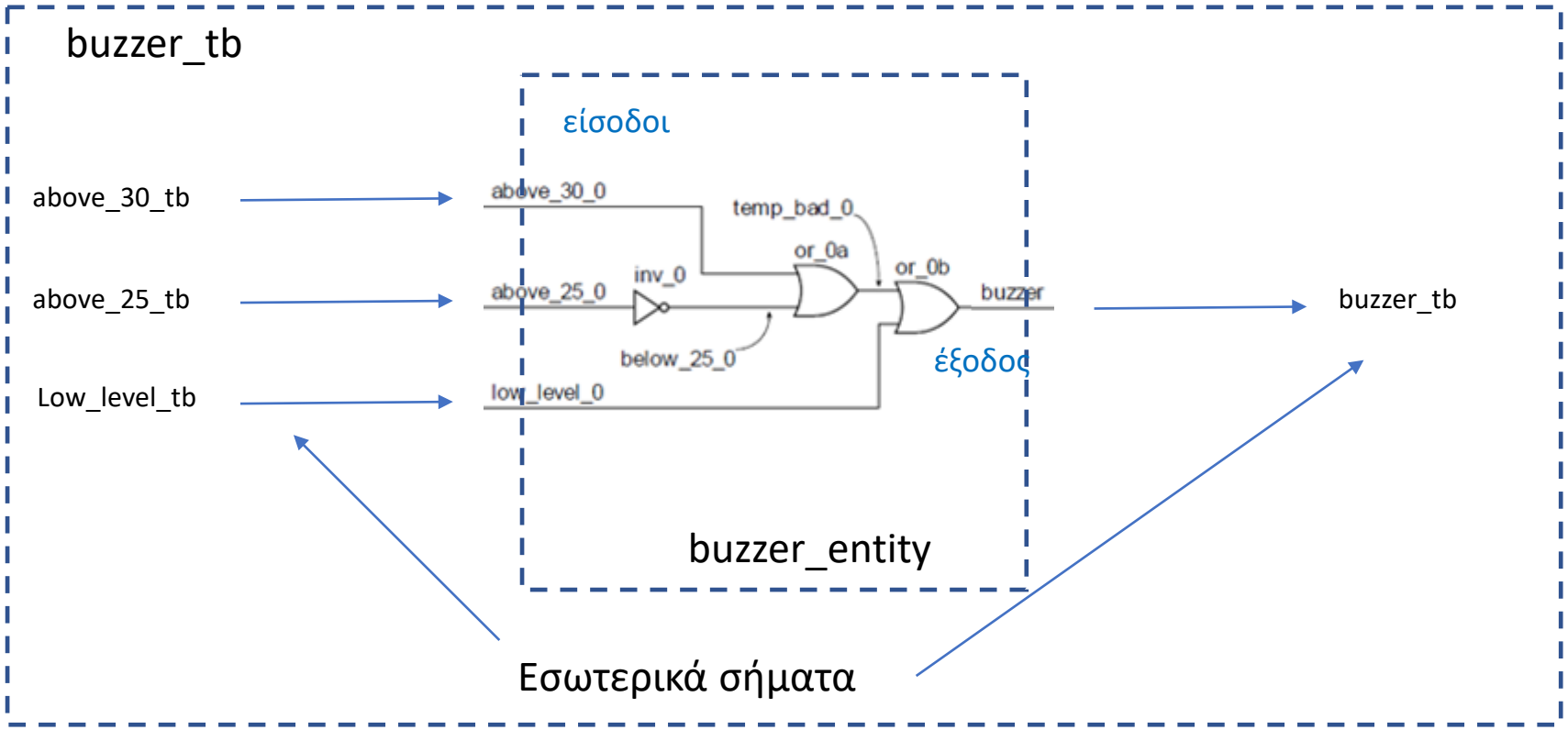
```
signal_name <= expression;
```

  - 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

Ψηφιακό κύκλωμα – VHDL: Προσομοίωση



VHDL

Ψηφιακό κύκλωμα – VHDL: Προσομοίωση – Πίνακας Αληθείας

Πίνακας Αληθείας της συνάρτησης που εκφράζει το σήμα buzzer

above_30_0	above_25_0	low_level_0	buzzer_0
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_0: in std_logic;
above_25_0: in std_logic;
low_level_0: 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_0 => above_25_tb,
above_30_0 => above_30_tb,
low_level_0 => low_level_tb,
buzzer => buzzer_tb);
```

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

```
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 - 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_0: in std_logic;
above_25_0: in std_logic;
low_level_0: 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_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  Beh_tb;
```

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

VHDL

Ψηφιακό κύκλωμα – Αναπαράσταση σε VHDL – Αρχιτεκτονική – Components

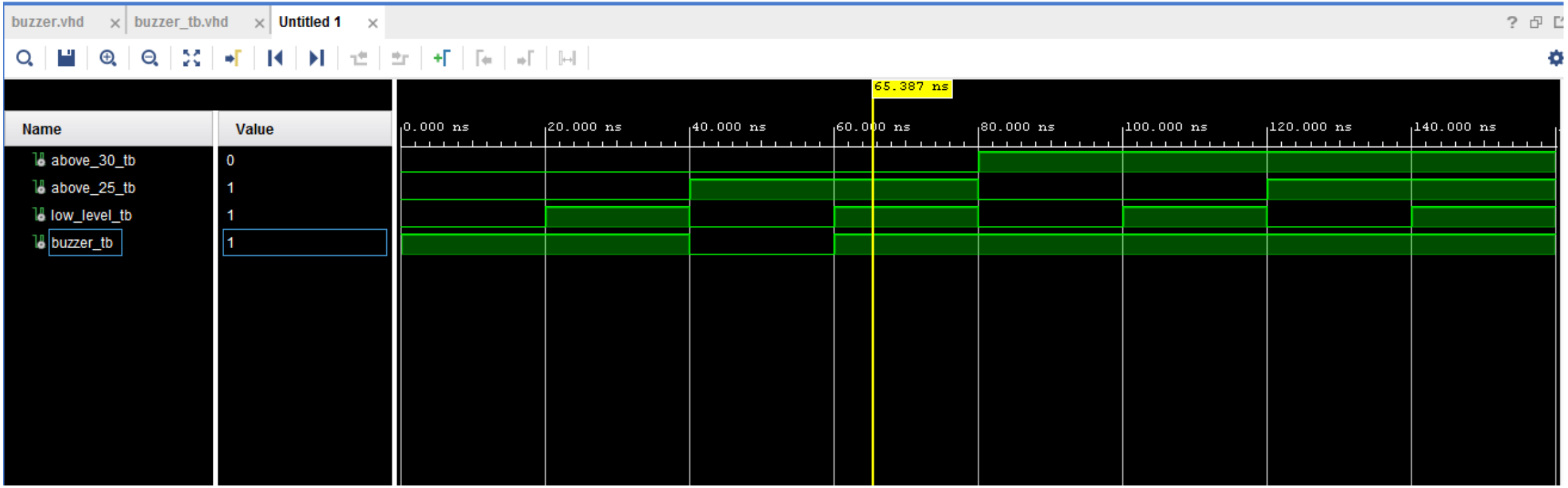
- **Ταυτόχρονες εντολές στοιχείων** (concurrent_component_statements)

```
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 το οποίο συνδέουμε με στο σήμα port_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 διαύλων του στοιχείου (**θέλει προσοχή η σειρά των σημάτων**)

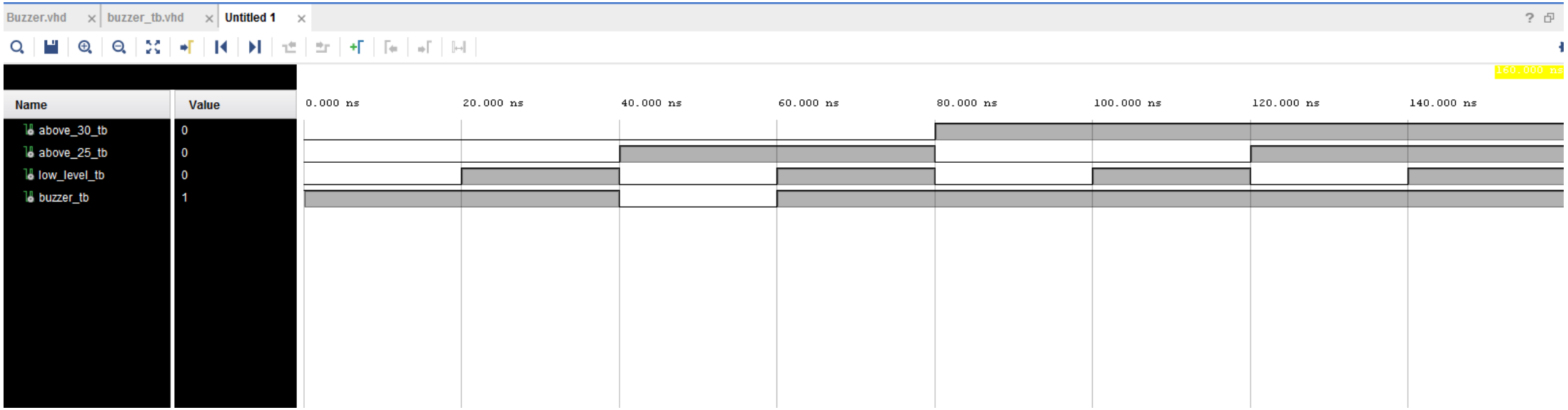
VHDL

Ψηφιακό κύκλωμα – VHDL: Προσομοίωση - Χρονοσειρά



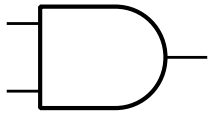
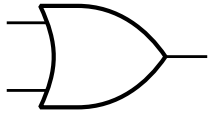
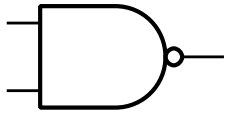
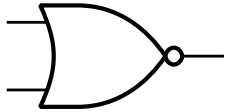
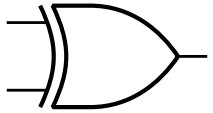
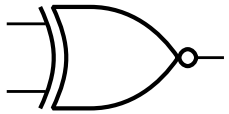
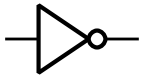
VHDL - Vivado

Ψηφιακό κύκλωμα – VHDL: Προσομοίωση - Χρονοσειρά



VHDL

Λογικοί Τελεστές

a **and** b	$a \cdot b$	
a **or** b	$a + b$	
a **nand** b	$\overline{a \cdot b}$	
a **nor** b	$\overline{a + b}$	
a **xor** b	$a \oplus b$	
a **xnor** b	$\overline{a \oplus b}$	
not a	$\overline{a}$	

- 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'

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_in_0: in std_logic;
signal_in_1: in std_logic;
signal_in_2: in std_logic;
signal_in_3: in std_logic;
```

Εναλλακτικά

```
signal_in : in std_logic_vector (3 downto 0);
```

```
signal_in <="0101";
```

```
signal_in(0) <='1'
signal_in(1) <='0'
signal_in(2) <='1'
signal_in(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 τύπου **natural**
- 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:
  - V = "00000000" για σύγκριση **ολόκληρου** του διανύσματος
  - V(3 downto 0) = "0000" για σύγκριση **μέρους** του διανύσματος
  - Προσοχή. **Μη επιτρεπτή σύγκριση**: V = "----0000"
    - το '-' δεν εκλαμβάνεται σαν don't care κατά τη σύγκριση

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

Εγγραφές σε Τμήματα Εργαστηρίων

- Στο διάστημα 23/10 – 27/10 θα επιλέξετε ομάδες για την παρακολούθηση του αντίστοιχου εργαστηρίου (δίπλα στο αναγνωστήριο).
- Την εβδομάδα 30/10 – 3/11 θα γίνει το πρώτο ΕΡΓΑΣΤΗΡΙΟ (και **ΔΕΝ** θα γίνει διάλεξη στην Α2).

Περίληψη

- Παράδειγμα ανάπτυξης εφαρμογής σε VHDL
- Δηλώσεις Οντότητας (entity), Αρχιτεκτονικής (architecture).
- Ports και Εσωτερικά σήματα
- Ταυτόχρονες εντολές
- Components
- Προτεραιότητες πράξεων
- Τύποι σημάτων (std_logic, std_logic_vector)
- Διαβάζετε τις παραγράφους 3.2 από Ashenden και 2.1, 2.6, 4.1-4.6 (ΟΧΙ το κομμάτι της VERILOG) από το βιβλίο των Harris.