dscal
DIGITAL SYSTEMS & COMPUTER ARCHITECTURE LABORATORY

4ο Εργαστήριο Σχεδίασης Ψηφιακών Συστημάτων

Ορισμός Οντότητας – Ορισμός/Είδη Αρχιτεκτονικής

Βασιλόπουλος Διονύσης

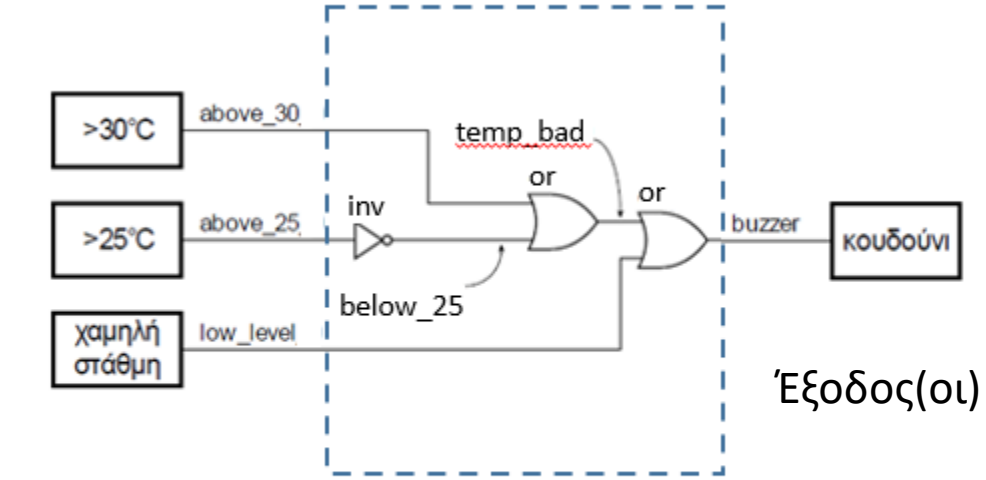
Ε.ΔΙ.Π Τμήματος Πληροφορικής & Τηλεπικοινωνιών - ΕΚΠΑ

ΤΜΗΜΑ
ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

VHDL

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

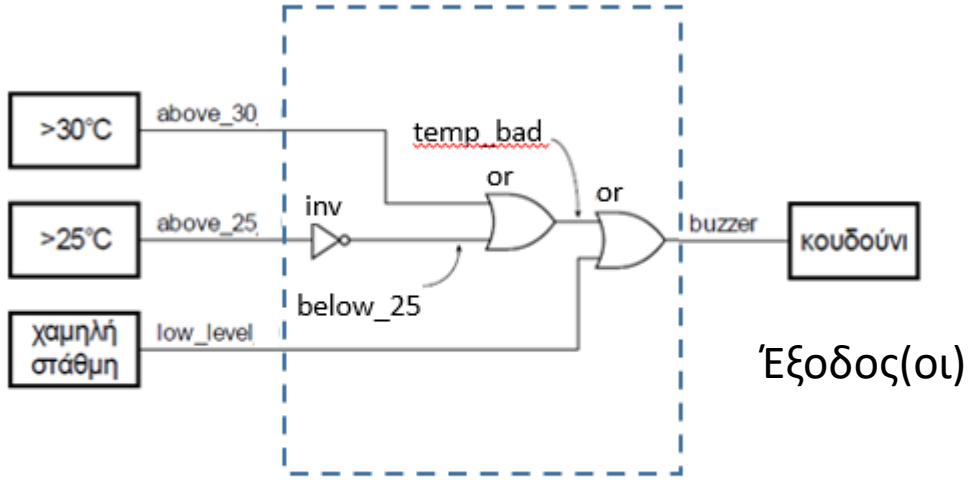
Ψηφιακό κύκλωμα – Αναπαράσταση σε VHDL – Entity: Input/Output

- 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 – VHDL: Entity (Input/Output PORTS)



Είσοδοι

Έξοδος(οι)

Οντότητα/Entity

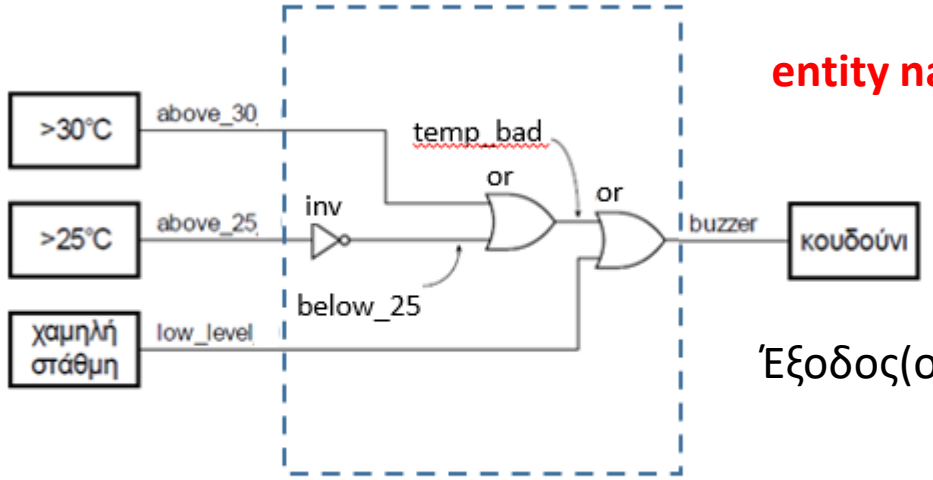
Υποχρεωτικές Βιβλιοθήκες

Περιγραφή Οντότητας

```
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;
```

VHDL

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



Είσοδοι

Οντότητα/Entity

Έξοδος(οι)

entity name

χρήση πρότυπης βιβλιοθήκης

package

port type

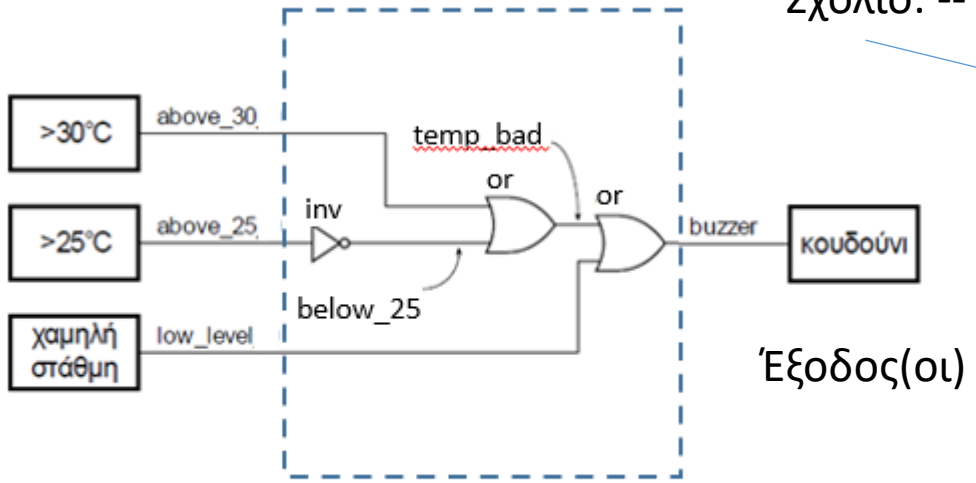
port list

port mode

```
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;
```

VHDL

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



Είσοδοι

Έξοδος(οι)

Οντότητα/Entity

Σχόλιο: --

Χωρίς τις Βιβλιοθήκες

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

entity buzzer is

port (
above_25: in bit;
above_30: in bit;
low_level: 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

- **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 – Τύποι σημάτων

Std_logic

Τιμή	Modeling for simulation	Synthesis
U	Uninitialized	Uninitialized
X	Strong driven unknown	Don't care/Multiple Values
0	Strong driven 0	0
1	Strong driven 1	1
Z	High impedance	High impedance/Δεν περνάει ρεύμα
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

U	X	0	1	Z	W	L	H	-	
U	X	X	1	1	1	1	1	X	**1**
U	X	0	X	0	0	0	0	X	**0**
U	U	U	U	U	U	U	U	U	**U**
U	X	X	X	X	X	X	X	X	**X**
U	X	0	1	Z	W	L	H	X	**Z**
U	X	0	1	W	W	W	W	X	**W**
U	X	0	1	L	W	L	W	X	**L**
U	X	0	1	H	W	W	H	X	**H**
U	X	X	X	X	X	X	X	X	**-**

resolution table για σήμα std_logic με δύο drivers
https://vhdlwhiz.com/std_logic/

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**

VHDL – Τύποι σημάτων

Std_logic_vector

- Μπορούμε να έχουμε και δήλωση σήματος ως std_logic_vector χωρίς index range.
- Σε αυτή την περίπτωση υπονοείται από άλλη οντότητα
- Μπορείτε να βρείτε παράδειγμα :

https://electronics.stackexchange.com/questions/622590/does-vhdl-allow-a-std-logic-vector-port-with-no-bounds

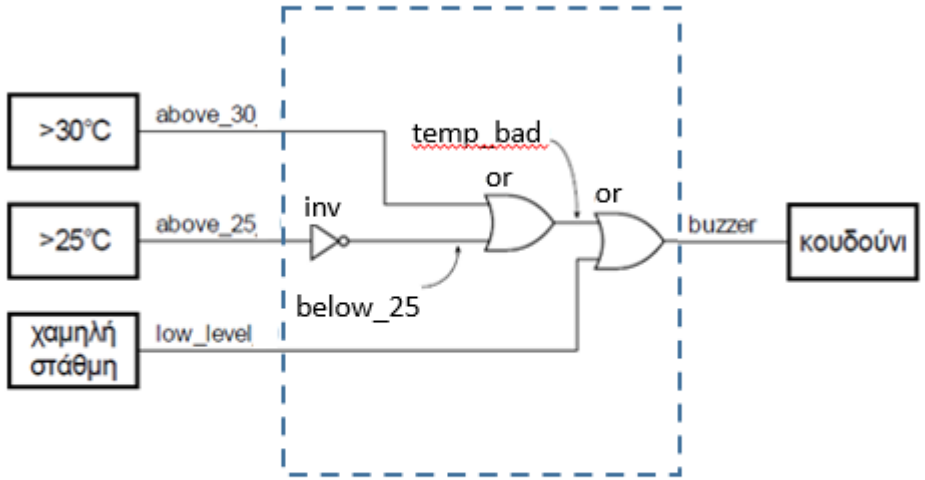
VHDL

Ψηφιακό κύκλωμα – VHDL: Architecture (Dataflow)

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

VHDL

Ψηφιακό κύκλωμα – VHDL: Architecture (Dataflow)

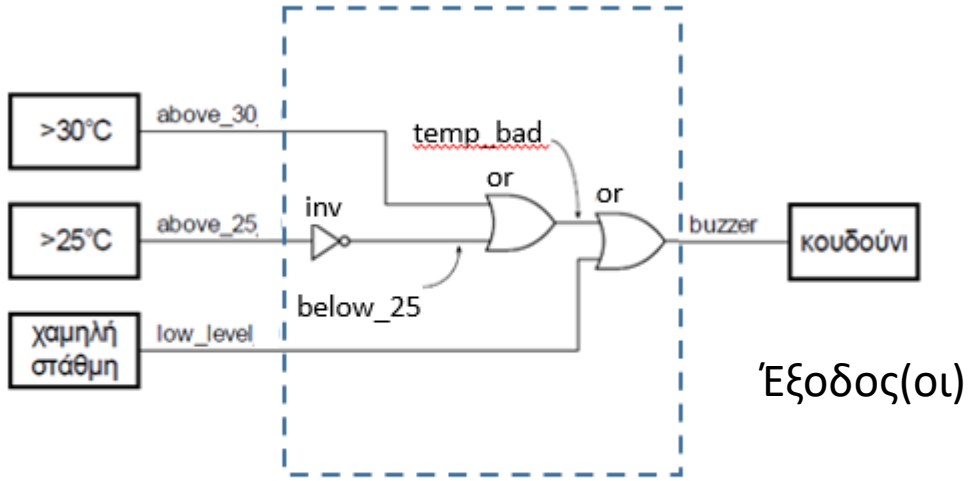



```
architecture Dataflow of buzzer is

begin

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

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)



Είσοδοι

Έξοδος(οι)

Οντότητα/Entity

Εσωτερικά σήματα

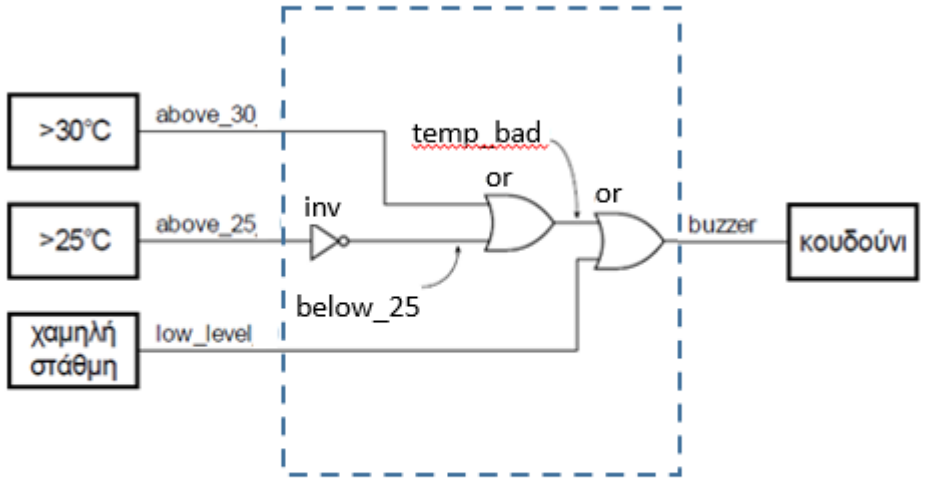
temp_bad

-

below_25

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, below_25: std_logic;
begin

below_25<=not above_25;
temp_bad<=above_30 or below_25;
buzzer<= temp_bad or low_level;

end architecture buzzer;
```

concurrent statements

Τα εσωτερικά σήματα μπορούν να είναι και αριστερά και δεξιά

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 - Παράδειγμα

Υλοποίηση Αρχιτεκτονικής (Dataflow)

1. Αποτελείται από απλές εντολές ανάθεσης τιμών σε σήματα
2. Κάθε εντολή εκτελείται όταν μεταβληθεί η τιμή ενός σήματος στο αριστερό μέρος.
3. Όλες οι εντολές ανάθεσης εκτελούνται ταυτόχρονα (παράλληλα)
4. Οι εντολές ανάθεσης αντιστοιχούν σε λογικές πράξεις της άλγεβρας Boole.
5. Το RTL διάγραμμα που προκύπτει είναι μία απεικόνιση της άλγεβρας Boole που εκφράζουν οι εντολές ανάθεσης σε στοιχειώδεις λογικές πύλες (AND, OR, NOT) και πολυπλέκτες.

VHDL – Process

Υλοποίηση Αρχιτεκτονικής (Behavioral)

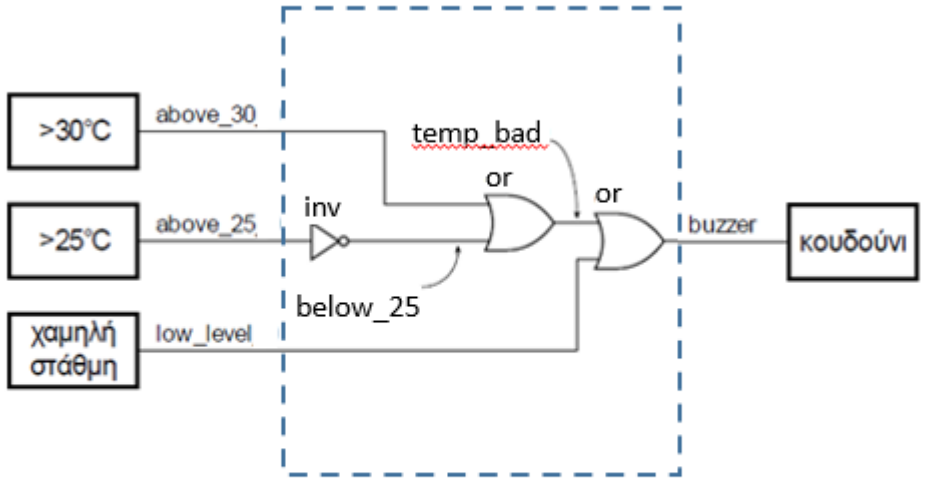
- **process**: η διεργασία είναι μία ομάδα από εντολές που εκτελούνται ακολουθιακά

Sensitivity list

```
architecture arch_name of entity_name is
begin
  label: process (signal_name, ..., signal_name)
    variable variable_name: variable_type;
  begin
    sequential_statement;
    ...
    sequential_statement;
  end process;
end arch_name;
```

VHDL

Ψηφιακό κύκλωμα – VHDL: Architecture (Behavioral)




```
architecture behavior of vat_buzzer is

begin

test process (above_30, above_25, low_level ) is
begin
buzzer <= low_level or (above_30 or not above_25);
end process test;
end architecture behavior;
```

VHDL

Ψηφιακό κύκλωμα – VHDL: Architecture (Structural)

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

end architecture arch_name;
```

VHDL

Ψηφιακό κύκλωμα – VHDL: Architecture (Structural)

- **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**. Ξέρουμε τη λειτουργικότητα που προσφέρει. Όταν επαναχρησιμοποιούμε μια οντότητα (entity) την ονομάζουμε **component**.
- **signal_name**: το όνομα του σήματος (εάν είναι πολλά σήματα χωρίζονται με κόμμα)
  - στις δηλώσεις σημάτων (μετά το **is** και πριν το **begin**) 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)
- **Ταυτόχρονες εντολές ανάθεσης σήματος** (Όπως σε αρχιτεκτονική Dataflow)

VHDL

Ψηφιακό κύκλωμα – Αναπαράσταση σε VHDL – Αρχιτεκτονική – Components (1/2)

- **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
- **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 - Παράδειγμα

Υλοποίηση Αρχιτεκτονικής (Structural – Δήλωση Οντοτήτων)

```
entity OR_gate is
port(A  : in  std_logic;
     B  : in  std_logic;
     O  : out std_logic);
end entity OR_gate;

architecture Dataflow of OR_gate is
begin
   O<=A or B;
end Dataflow;
```

VHDL - Παράδειγμα

Υλοποίηση Αρχιτεκτονικής (Structural – Δήλωση Οντοτήτων)

```
entity NOT_gate is
port(A  : in  std_logic;
       O  : out std_logic);
end entity NOT_gate;

architecture Dataflow of NOT_gate is
begin
   O<=not A;
end Dataflow;
```

VHDL - Παράδειγμα

Υλοποίηση Αρχιτεκτονικής (Structural – Δήλωση Αρχιτεκτονικής Κύριας Οντότητας)

```
architecture Structural of buzzer is

component OR_gate is
port(A  : in  std_logic;
     B  : in  std_logic;
     O  : out std_logic);
end component OR_gate;

component NOT_gate is
port(A  : in  std_logic;
     O  : out std_logic);
end component NOT_gate;

signal below_25: std_logic;
signal temp_bad: std_logic;

begin

comp_not: NOT_gate port map(A=>above_25, O=>below_25);
comp_or1: OR_gate port map (A=>above_30, B=>below_25, O=>temp_bad);
comp_or2: OR_gate port map (A=>low_level, B=>temp_bad, O=>buzzer);

end architecture Structural;
```

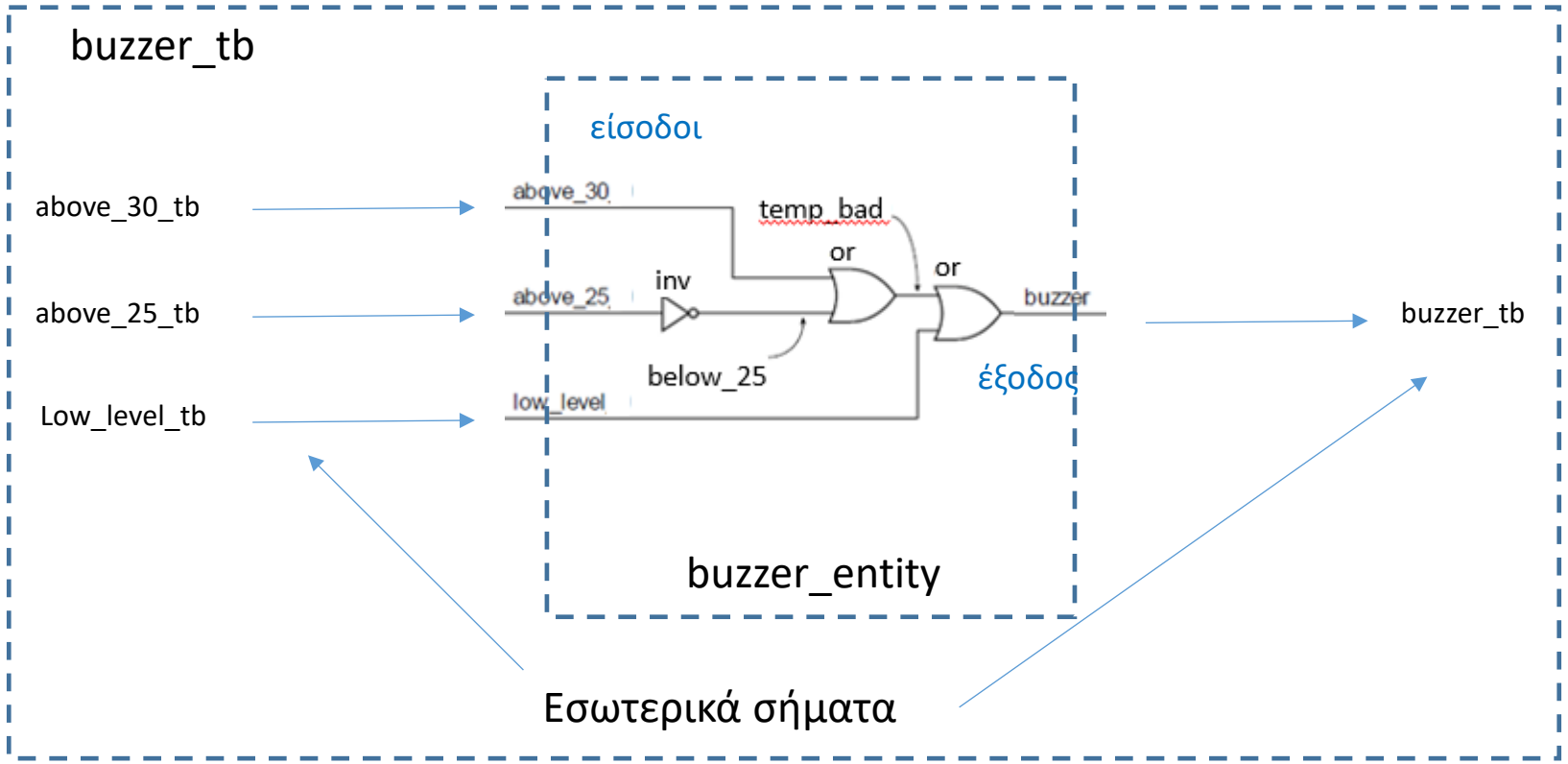
VHDL

Υλοποίηση Αρχιτεκτονικής (Structural)

1. Για την υλοποίηση της αρχιτεκτονικής χρησιμοποιούμε τη λειτουργικότητα άλλων Οντοτήτων που έχουμε ήδη υλοποιήσει.
2. Τις οντότητες που θα χρησιμοποιήσουμε τις δηλώνουμε (όνομα και port) ανάμεσα στο is και το begin της αρχιτεκτονικής. Όμως πλέον έχουν την έννοια της συνιστώσας (component).
3. Η αρχιτεκτονική πλέον αποτελείτε και από εντολές που καλούν(δημιουργούν) τα components, τα οποία μπορεί και να επικοινωνούν μεταξύ τους (συνηθέστερη περίπτωση).
4. Άρα μια οντότητα μπορεί να χρησιμοποιεί άλλες οντότητες (με τη μορφή component), οι οποίες μπορεί να είναι υλοποιημένες με οποιοδήποτε από τις 3 αρχιτεκτονικές. Στη περίπτωση της structural δομής βλέπουμε ότι σχηματίζεται ένα δέντρο, τα φύλλα του οποίου είναι οντότητες. Τα φύλλα που είναι τερματικά έχουν δομή Dataflow ή Behavioral.
5. ΔΕΝ αλλάζει τίποτα στην προσομοίωση

Μπορείτε να δείτε και το: https://buzztech.in/vhdl-modelling-styles-behavioral-dataflow-structural/

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

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 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_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 - 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 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
```

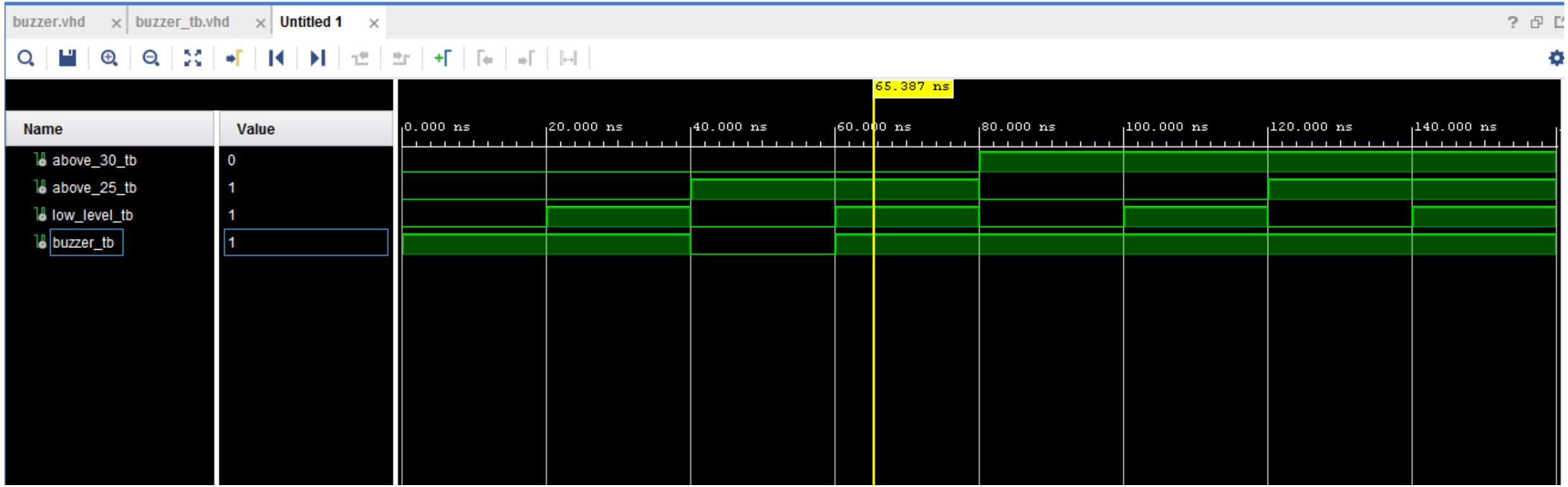
```
uut: buzzer  port map (
above_25_0 => above_25_tb,
above_30_0 => above_30_tb,
low_level_0 => low_level_tb,
buzzer => buzzer_tb);
```

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 architecture  Beh_tb;
```

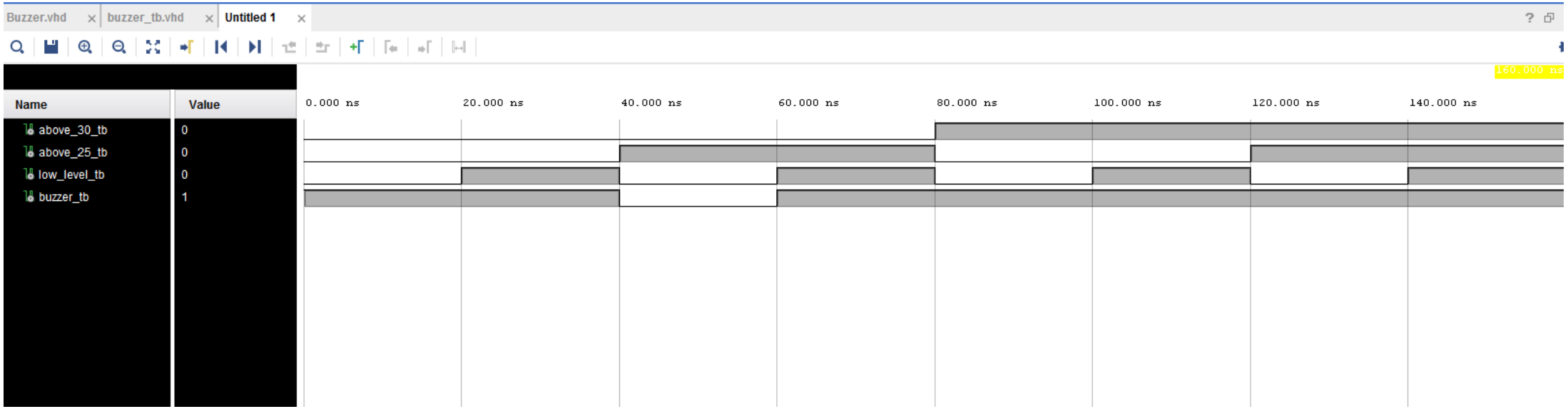
VHDL

Ψηφιακό κύκλωμα – VHDL: Προσομοίωση - Χρονοσειρά



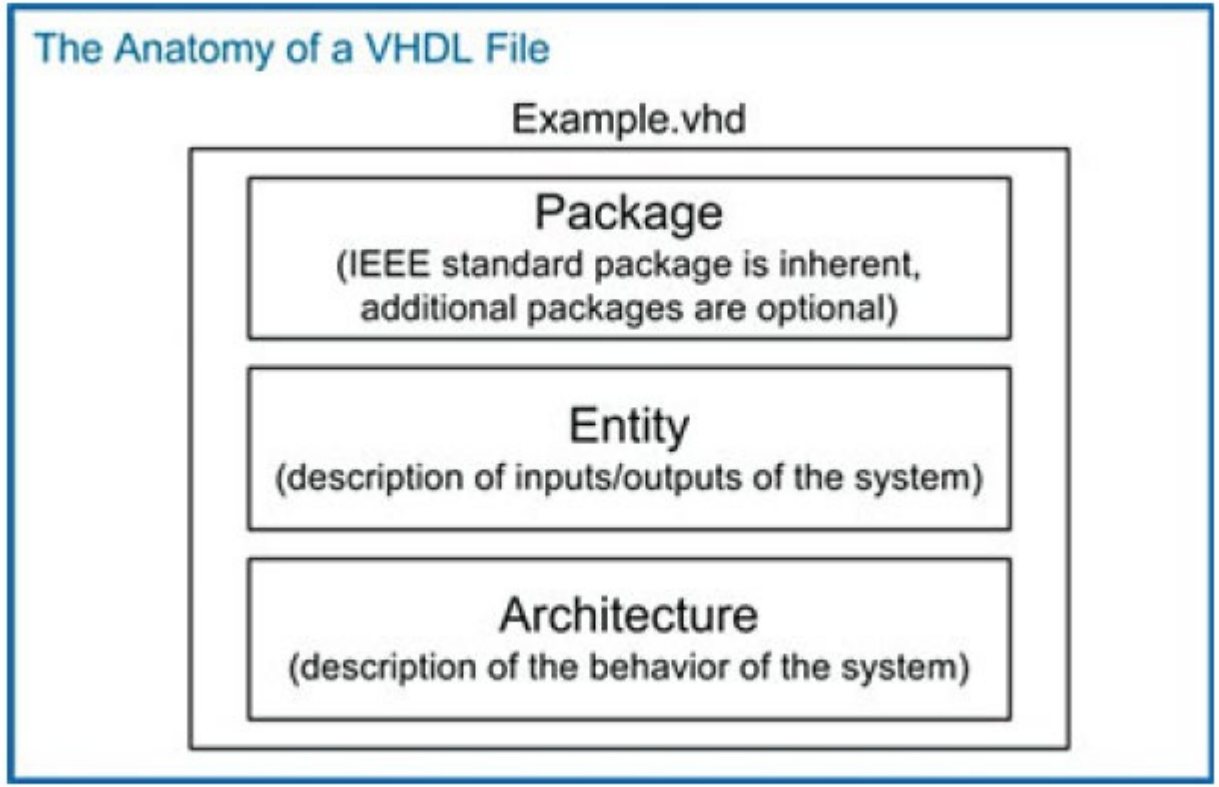
VHDL - Vivado

Ψηφιακό κύκλωμα – VHDL: Προσομοίωση - Χρονοσειρά



VHDL - Vivado

Ψηφιακό κύκλωμα – VHDL: Δομή Αρχείου



Boards

- Boards που μπορείτε να ερευνήσετε για home use:

Without ARM
https://www.digikey.gr/en/products/detail/digilent-inc/410-183/4970275

With ARM (Zynq7000)
https://www.digikey.gr/en/products/detail/digilent-inc/410-370/9445909
https://www.digikey.gr/en/products/detail/digilent-inc/6003-410-017/9839382
https://www.digikey.gr/en/products/detail/digilent-inc/410-351-10/7652757

Περίληψη

- Παράδειγμα ανάπτυξης εφαρμογής σε VHDL
- Δηλώσεις Οντότητας (entity), Αρχιτεκτονικής (architecture).
- Ports και Εσωτερικά σήματα
- Ταυτόχρονες εντολές
- Components
- Παράδειγμα των 3 ειδών αρχιτεκτονικής (Dataflow, Behavioral, Structural)
- Τύποι σημάτων (std_logic, std_logic_vector)
- Διαβάζετε τις παραγράφους 2.3, 2.4 από Ashenden και 2.1 - 2.6, 4.1, 4.2, 4.3 (ΟΧΙ το κομμάτι της VERILOG) από το βιβλίο των Harris.
- Συνοπτικός οδηγός VHDL: https://redirect.cs.umbc.edu/portal/help/VHDL/summary_one.html