[](http://www.teilar.gr/)

Αρχιτεκτονική Η/Υ ΙΙ

**Ασκήσεις Πράξης:** Ενότητα 3

Πετρέλλης Νικόλαος

Σχολή Τεχνολογικών Εφαρμογών

Τμήμα Μηχανικών Πληροφορικής Τ.Ε.

**Χρηματοδότηση**

* Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
* Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Τ.Ε.Ι. Θεσσαλίας**» έχει χρηματοδοτήσει μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.
* Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

[](http://www.edulll.gr/)

«Οι παρούσες ασκήσεις βασίζονται στις εργαστηριακές

ασκήσεις που είχε προετοιμάσει για το μάθημα της Αρχιτεκτονικής ΗΥ ΙΙ ο

καθηγητής Γεώργιος Αδάμ.»

**Σκοποί ενότητας**

Ο σκοπός αυτής της ενότητας είναι να γνωρίσουμε τη μεθοδολογία σχεδίασης και προσομοίωσης μονάδων του υλικού ενός υπολογιστικού συστήματος, με τη βοήθεια λογισμικού εισαγωγής και προσομοίωσης (πχ Max+PlusII) με τη γλώσσα προγραμματισμού VHDL. Συγκεκριμένα, σε αυτήν την ενότητα, γίνεται η μελέτη των συνδυαστικών κυκλωμάτων.

**Περιεχόμενα**

[1. Συνδυαστικά κυκλώματα 5](#_Toc411445107)

[1.1. Ημι-αθροιστής (*Half adder*) 5](#_Toc411445108)

[1.1.1. Άσκηση 1 5](#_Toc411445109)

[1.2. Πλήρης αθροιστής (*full adder*) 8](#_Toc411445110)

[1.2.1. Άσκηση 2 8](#_Toc411445111)

[1.2.2. Άσκηση 3 2](#_Toc411445112)

[1.3. Πολυπλέκτης δύο προς ένα (*two-to-one multiplexer*) 3](#_Toc411445113)

[1.3.1. Άσκηση 4 3](#_Toc411445114)

[1.4. Αποκωδικοποιητής 2 προς 4 (*2-to-4 decoder*) 4](#_Toc411445115)

[1.4.1. Άσκηση 5 5](#_Toc411445116)

[1.5. Συγκριτής 4bits (*4-bits comparator*) 5](#_Toc411445117)

[1.5.1. Άσκηση 6 6](#_Toc411445118)

[Σημειώματα 7](#_Toc411445119)

**Πίνακες**

[*Πίνακας 1* 5](#_Toc411445223)

[*Πίνακας 2* 8](#_Toc411445224)

[*Πίνακας 3* 3](#_Toc411445225)

[*Πίνακας 4* 5](#_Toc411445226)

**Εικόνες**

[*Εικόνα 1* 5](#_Toc411445227)

[*Εικόνα 2* 6](#_Toc411445228)

[*Εικόνα 3* 6](#_Toc411445229)

[*Εικόνα 4* 7](#_Toc411445230)

[*Εικόνα 5* 7](#_Toc411445231)

[*Εικόνα 6* 8](#_Toc411445232)

[*Εικόνα 7* 8](#_Toc411445233)

[*Εικόνα 8* 1](#_Toc411445234)

[*Εικόνα 9* 1](#_Toc411445235)

[*Εικόνα 10* 3](#_Toc411445236)

[*Εικόνα 11* 5](#_Toc411445237)

[*Εικόνα 12* 6](#_Toc411445238)

# Συνδυαστικά κυκλώματα

Γνωρίζουμε ότι τα συνδυαστικά κυκλώματα (combinational circuits) είναι τα κυκλώματα στα οποία η τιμή της κάθε εξόδου, εξαρτάται αποκλειστικά από τις τιμές των σημάτων που εφαρμόζονται στις εισόδους.

Ορισμένα από τα συνδυαστικά κυκλώματα που παρουσιάζονται εδώ είναι:

ημι-αθροιστής, πλήρης αθροιστής, πολυπλέκτης, αποκωδικοποιητής, συγκριτής.

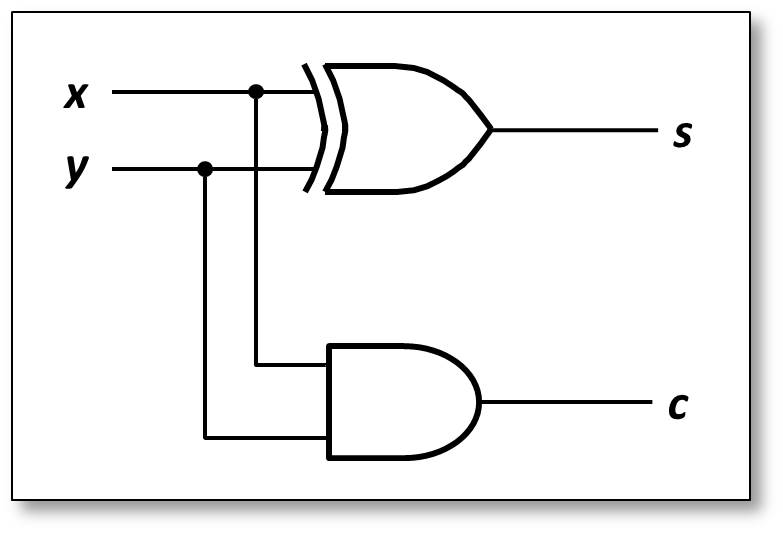
## Ημι-αθροιστής (Half adder)

Η δυαδική πρόσθεση εκτελείται με τον ίδιο τρόπο με τη δεκαδική πρόσθεση, με τη διαφορά ότι η τιμή που μπορεί να έχει το κάθε ψηφίο είναι 0 ή 1.

Ακολουθεί το λογικό κύκλωμα και ο πίνακας αληθείας του ημι-αθροιστή (half-adder).

Πίνακας 1

|  |  | Carry | Sum |
| --- | --- | --- | --- |
| x | y | c | s |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |



Εικόνα 1

### Άσκηση 1

Για το παραπάνω κύκλωμα του ημι-αθροιστή να εκτελέσετε τα εξής:

1. Να γράψετε τον κώδικα VHDL που περιγράφει τη λειτουργία του.
2. Να εισάγετε το σχηματικό του διάγραμμα στο λογισμικό Max+plusII, και να το μελετήσετε με προσομοίωση.

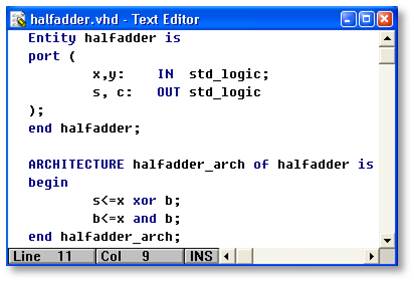
Από το παραπάνω κύκλωμα και τον πίνακα αληθείας του, παρατηρούμε ότι η συμπεριφορά του θα μπορούσε να περιγραφεί σε VHDL, λαμβάνοντας υπόψη την παρακάτω λογική σχέση:

Carry <= X **AND** Y;

Sum <= X **XOR** Y;

Εισαγωγή του κώδικα VHDL που περιγράφει τη λειτουργία του κυκλώματος.

Το παράθυρο εισαγωγής (text editor) του κώδικα σε VHDL (halfadder.vhd) στο λογισμικό Max+plusII είναι:

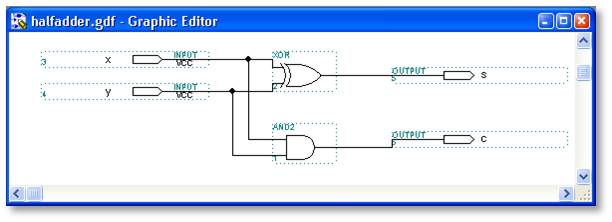


Εικόνα 2

**Σημείωση**: Φυσικά για την εισαγωγή του κώδικα θα μπορούσε να χρησιμοποιηθεί και οποιοσδήποτε άλλος editor (πχ notepad).

1. Εισαγωγή του σχηματικού διαγράμματος του κυκλώματος και μελέτη με προσομοίωση.

Το παράθυρο εισαγωγής (graphics editor) του σχηματικού διαγράμματος (halfadder.gdf) του κυκλώματος στο λογισμικό Max+plusII είναι:



Εικόνα 3

Σε αυτό το σημείο, πρέπει να σημειώσουμε ότι μπορούμε να παράγουμε από ένα σχηματικό διάγραμμα κυκλώματος, όπως το παραπάνω, τον ισοδύναμο κώδικα που του αντιστοιχεί σε VHDL (σε σχέση με κάποια συγκεκριμένη δομή προγραμματιζόμενης συσκευής - structural VHDL) και σε AHDL (Altera HDL) με τους εξής τρόπους αντίστοιχα:

1. Δημιουργία αρχείου ισοδύναμου κώδικα σε VHDL.

Εάν κατά τη μεταγλώττιση του παραπάνω κυκλώματος δεν επιλέξουμε:

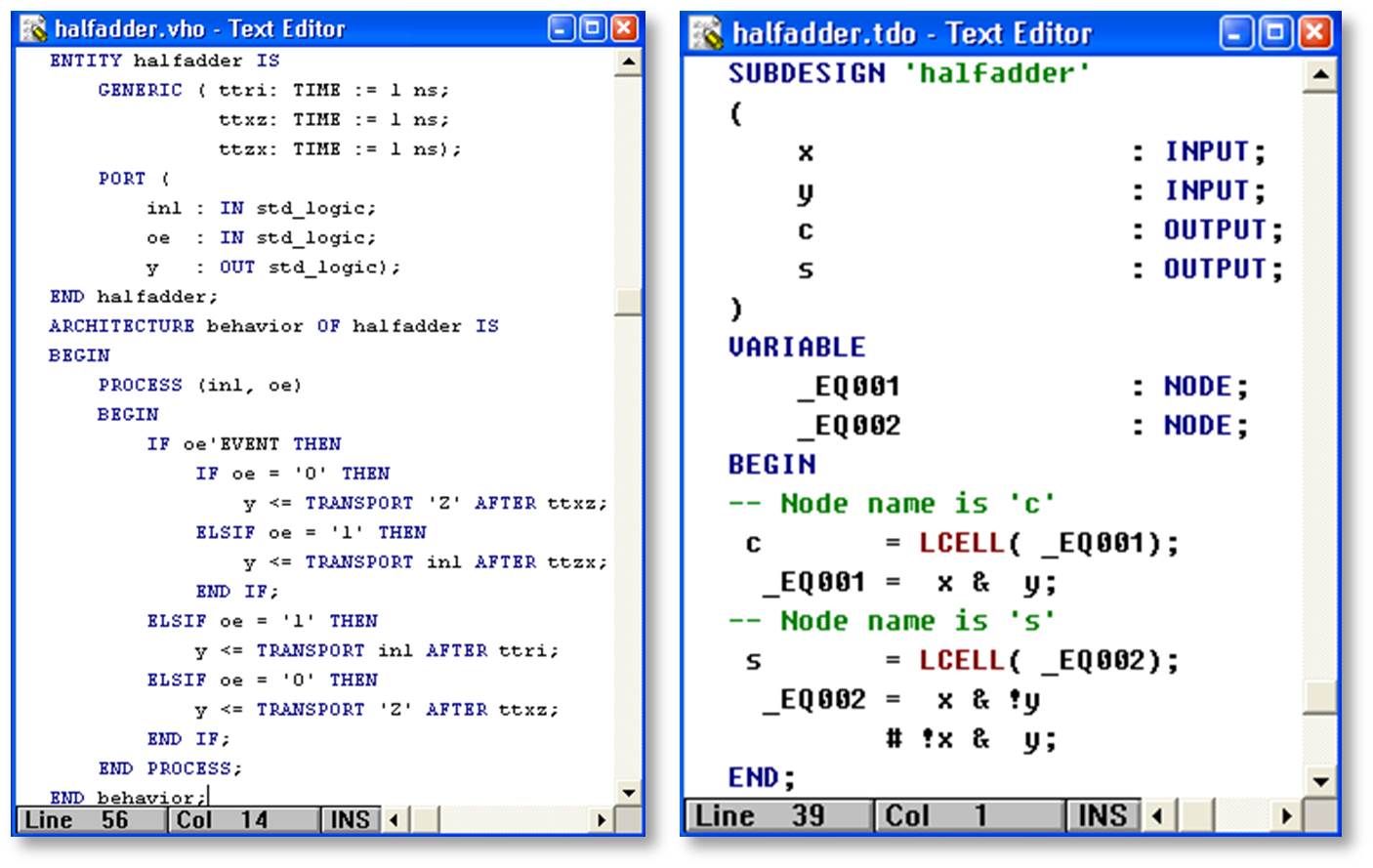
Processing ⇒ Functional SNF Extractor (αλλά αφήσουμε την εξορισμού επιλογή **Timing SNF Extractor**), και επιλέξουμε από το μενού Interfaces ⇒ **VHDL Netlist Writer**, τότε το αρχείο που θα παραχθεί από τη μεταγλώττιση με την κατάληξη **.vho** περιέχει το ζητούμενο VHDL κώδικα (σε σχέση με μία συσκευή EP610ILC- 10).

1. Δημιουργία αρχείου ισοδύναμου κώδικα σε AHDL (Altera HDL).

Εάν κατά τη μεταγλώττιση του παραπάνω κυκλώματος δεν επιλέξουμε:

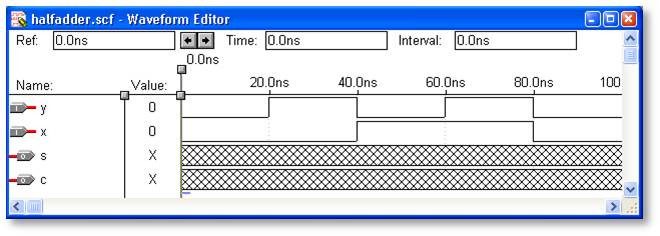
Processing ⇒ Functional SNF Extractor (αλλά αφήσουμε την εξορισμού επιλογή **Timing SNF Extractor**), και επιπλέον επιλέξουμε και την επιλογή Generate AHDL TDO File, τότε το αρχείο που θα παραχθεί από τη μεταγλώττιση με την κατάληξη **.tdo**, περιέχει το ζητούμενο AHDL κώδικα.

Παραδείγματα του ισοδύναμου κώδικα σε VHDL (halfadder.vho) και AHDL (halfadder.tdo) αντίστοιχα, του παραπάνω κυκλώματος, δίνεται στα παράθυρα που ακολουθούν:



Εικόνα 4

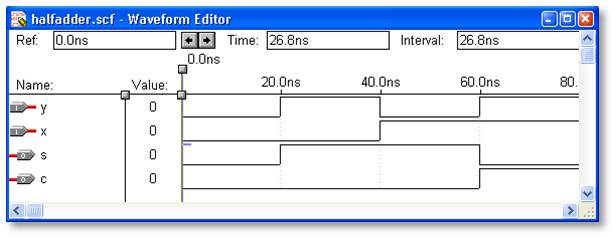
Το παράθυρο του επεξεργαστή κυματομορφών (waveform editor) για την εισαγωγή των διανυσμάτων δοκιμής (halfadder.scf), με τα οποία παριστάνουμε τα σήματα εισόδου του κυκλώματος (για την προσομοίωσή του) στο λογισμικό Max+plusII είναι:



Εικόνα 5

Εδώ θα πρέπει να σημειώσουμε, ότι καθορίσαμε τον χρόνο προσομοίωσης (File ⇒ End Time) σε 100ns, και το πλάτος του πλέγματος (Options ⇒ Grid Size) σε 20ns.

Τέλος, μετά και την εκτέλεση της προσομοίωσης, τα αποτελέσματα που προκύπτουν για τη συμπεριφορά του κυκλώματος μπορούμε να τα παρατηρήσουμε στο παράθυρο του επεξεργαστή κυματομορφών (Open SCF) που ακολουθεί:



Εικόνα 6

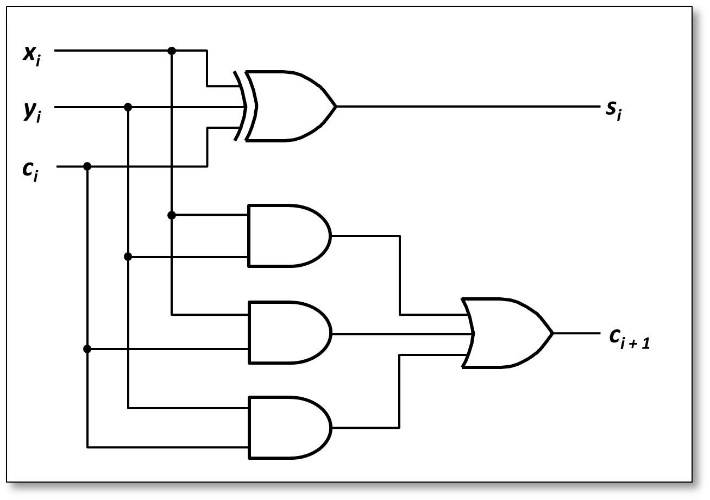
Ελέγχοντας τα παραπάνω αποτελέσματα της προσομοίωσης στο παράθυρο του επεξεργαστή κυματομορφών, παρατηρούμε ότι οι τιμές των εξόδων που παίρνουμε ότι είναι οι αναμενόμενες. Θα μπορούσαμε στη συνέχεια να πειραματιστούμε αλλάζοντας τα στοιχεία με νέα δεδομένα και τιμές.

## Πλήρης αθροιστής (full adder)

Το λογικό κύκλωμα, η συνάρτηση, καθώς και ο πίνακας αληθείας του πλήρη αθροιστή, είναι επίσης γνωστά.

Πίνακας 2

| ci | xi | yi | ci+1 | si |
| --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |



Εικόνα 7

### Άσκηση 2

Για το παραπάνω κύκλωμα του πλήρη αθροιστή, να κάνετε τα εξής:

1. Να γράψετε τον κώδικα VHDL που περιγράφει τη λειτουργία του.
2. Να εισάγετε το σχηματικό του διάγραμμα στο λογισμικό Max+plusII και να το μελετήσετε με προσομοίωση.
3. Κώδικας VHDL που περιγράφει τη λειτουργία του κυκλώματος.

-- VHDL κώδικας για τον πλήρη αθροιστή

**LIBRARY** ieee ; -- βιβλιοθήκη συστήματος (περιέχει το std\_logic\_1164,

και άλλα)

**USE** ieee.std\_logic\_1164.**all** ;

**ENTITY** fulladder **IS**

**PORT** ( Cin, x, : **IN** STD\_LOGIC ;

s, Cout : **OUT** STD\_LOGIC ) ;

**END** fulladder ;

**ARCHITECTURE** LogicFunc **OF** fulladder **IS**

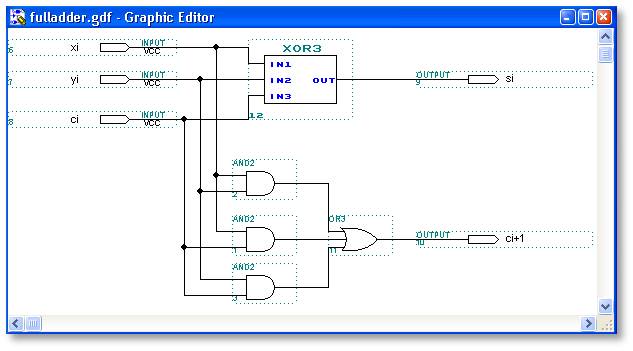
**BEGIN**

s <= x **XOR** y **XOR** Cin ;

Cout <= (x **AND** y) **OR** (Cin **AND** x) **OR** (Cin **AND** y) ;

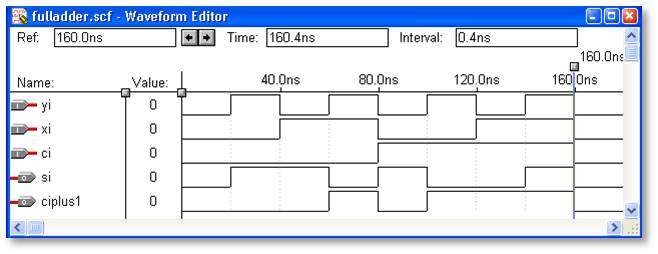
**END** LogicFunc ;

1. Σχηματικό διάγραμμα του κυκλώματος.



Εικόνα 8

Αποτελέσματα της προσομοίωσης.



Εικόνα 9

### Άσκηση 3

Με βάση την παραπάνω άσκηση του πλήρη αθροιστή, να δημιουργηθεί σε κώδικα VHDL ένας αθροιστής 4 bits, ο οποίος θα αποτελείται από επιμέρους στοιχεία (components) της παραπάνω άσκησης της μορφής full adder.

Ένας τρόπος για να γίνει αυτό, είναι μέσα στο νέο κώδικα να δηλώσουμε (πχ στο τμήμα της αρχιτεκτονικής ή ως πακέτο) το παραπάνω υποκύκλωμα (subcircuit) ως συνιστώσα (component), δηλαδή στοιχείο του νέου μας κυκλώματος.

Αυτό μπορεί να γίνει με την εντολή **Component** ως εξής:

**COMPONENT** fulladder

**PORT** ( Cin, x, y : **IN** STD\_LOGIC ;

s, Cout : **OUT** STD\_LOGIC ) ;

**END COMPONENT** ;

Εφόσον γίνει αυτή η δήλωση μπορεί να δημιουργηθεί το στιγμιότυπο (instantiate) της συνιστώσας, ως υποκύκλωμα του νέου μας σχεδίου με την παρακάτω εντολή ως εξής:

stage0: fulladder **PORT** **MAP** ( Cin, x0, y0, s0, c1 ) ;

stage1: fulladder **PORT** **MAP** ( c1, x1, y1, s1, c2 ) ;

……

Ο κώδικας σε VHDL που περιγράφει τη λειτουργία του 4-bits αθροιστή είναι ο ακόλουθος:

-- κώδικας του 4-bits αθροιστή

**LIBRARY** ieee ;

**USE** ieee.std\_logic\_1164.**all** ;

**ENTITY** adder4 **IS**

**PORT** (Cin : **IN** STD\_LOGIC ;

x3, x2, x1, x0 : **IN** STD\_LOGIC ;

y3, y2, y1, y0 : **IN** STD\_LOGIC ;

s3, s2, s1, s0 : **OUT** STD\_LOGIC ;

Cout : **OUT** STD\_LOGIC ) ;

**END** adder4 ;

**ARCHITECTURE** Structure **OF** adder4 **IS**

**SIGNAL** c1, c2, c3 : STD\_LOGIC ;

**COMPONENT** fulladder

**PORT(** Cin, x, y : IN STD\_LOGIC ;

s, Cout : OUT STD\_LOGIC ) ;

**END COMPONENT** ;

**BEGIN**

stage0: fulladder **PORT** **MAP** ( Cin, x0, y0, s0, c1 ) ;

stage1: fulladder **PORT** **MAP** ( c1, x1, y1, s1, c2 ) ;

stage2: fulladder **PORT** **MAP** ( c2, x2, y2, s2, c3 ) ;

stage3: fulladder **PORT** **MAP** (

Cin => c3, Cout => Cout, x => x3, y => y3, s => s3 ) ;

**END** Structure ;

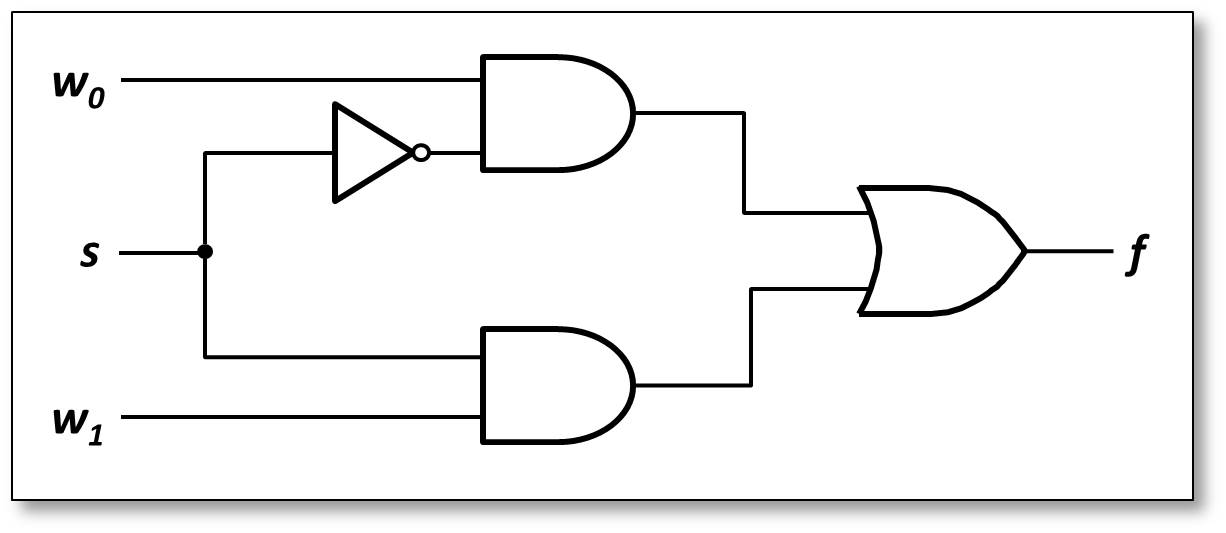
## Πολυπλέκτης δύο προς ένα (two-to-one multiplexer)

Το κύκλωμα ενός πολυπλέκτη όπως είναι γνωστό, διαβιβάζει την τιμή του σήματος κάποιας εισόδου δεδομένων στην έξοδο.

Ακολουθεί το λογικό κύκλωμα και ο πίνακας αληθείας του πολυπλέκτη 2 προς 1.

Πίνακας 3

| s | f |
| --- | --- |
| 0 | W0 |
| 1 | W1 |



Εικόνα 10

### Άσκηση 4

Μία πρώτη εκδοχή του κώδικα VHDL που περιγράφει τον παραπάνω πολυπλέκτη, είναι με χρήση της **WITH** – **SELECT**.

-- κώδικας του πολυπλέκτη 2 προς 1

**LIBRARY** ieee ;

**USE** ieee.std\_logic\_1164.**all** ;

**ENTITY** mux2to1 **IS**

**PORT** ( w0, w1, : **IN** STD\_LOGIC ;

f : **OUT** STD\_LOGIC ) ;

**END** mux2to1;

**ARCHITECTURE** Behavior **OF** mux2to1 **IS**

**BEGIN**

**WITH** s **SELECT**

f <= w0 **WHEN** '0',

w1 **WHEN** OTHERS ;

**END** Behavior ;

Μία δεύτερη εκδοχή του κώδικα VHDL του παραπάνω κυκλώματος, είναι με χρήση της εντολής **IF**-**THEN**-**ELSE**.

**LIBRARY** ieee ;

**USE** ieee.std\_logic\_1164.**all** ;

**ENTITY** mux2to1 **IS**

**PORT** ( w0, w1, s : IN STD\_LOGIC ;

f : OUT STD\_LOGIC ) ;

**END** mux2to1 ;

**ARCHITECTURE** Behavior **OF** mux2to1 **IS**

**BEGIN**

**PROCESS** ( w0, w1, s )

**BEGIN**

**IF** s = '0' **THEN**

f <= w0 ;

**ELSE**

f <= w1 ;

**END IF** ;

**END PROCESS** ;

**END** Behavior ;

Μία τρίτη εκδοχή του κώδικα VHDL του παραπάνω κυκλώματος, είναι με χρήση της **CASE**.

**LIBRARY** ieee ;

**USE** ieee.std\_logic\_1164.**all** ;

**ENTITY** mux2to1 **IS**

**PORT** ( w0, w1, s : **IN** STD\_LOGIC ;

f : **OUT** STD\_LOGIC ) ;

**END** mux2to1;

**ARCHITECTURE** Behavior **OF** mux2to1 **IS**

**BEGIN**

**PROCESS** ( w0, w1, s )

**BEGIN**

**CASE** s **IS**

**WHEN** '0' =>

f <= w0 ;

**WHEN** OTHERS =>

f <= w1 ;

**END CASE** ;

**END PROCESS** ;

**END** Behavior ;

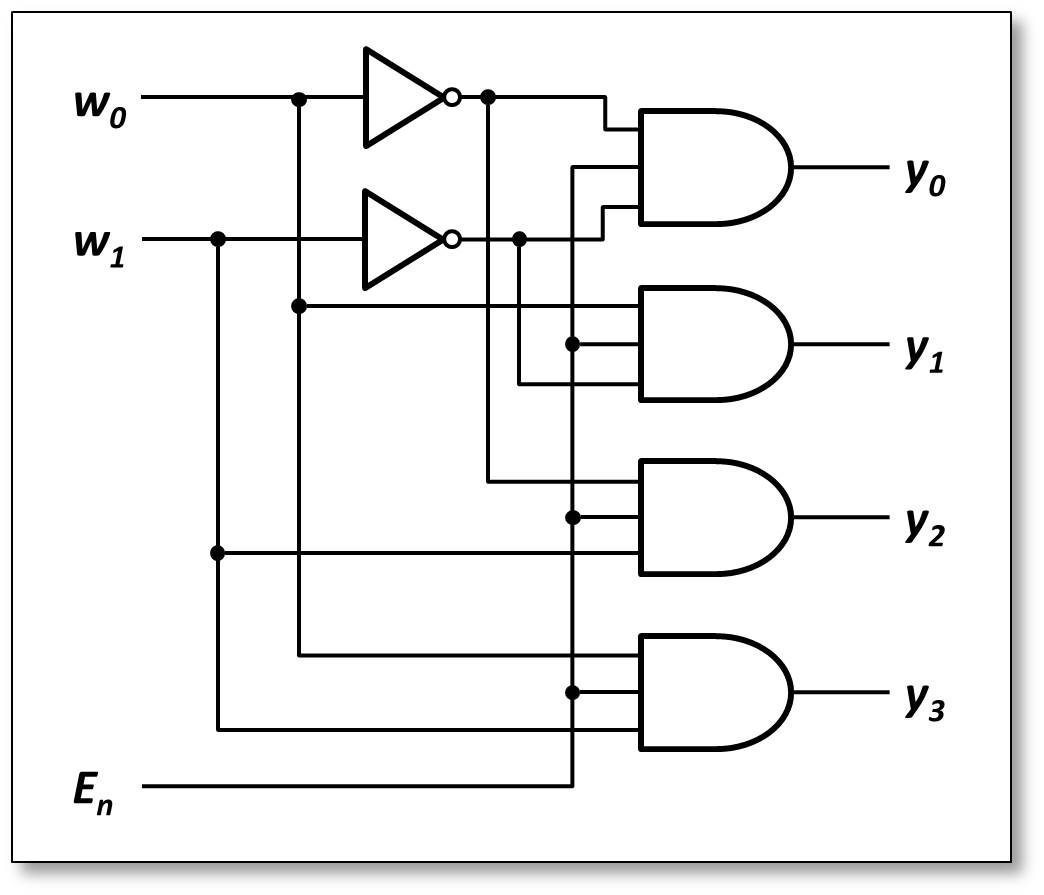
## Αποκωδικοποιητής 2 προς 4 (2-to-4 decoder)

Το κύκλωμα ενός αποκωδικοποιητή χρησιμοποιείται για την αποκωδικοποίηση πληροφοριών. Ένας δυαδικός αποκωδικοποιητής είναι ένα λογικό κύκλωμα με n εισόδους και 2n εξόδους.

Ακολουθεί το λογικό κύκλωμα και πίνακας αληθείας του αποκωδικοποιητή 2 προς 4.

Πίνακας 4

| En | W1 | W0 | Y0 | Y1 | Y2 | Y3 |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | x | x | 0 | 0 | 0 | 0 |



Εικόνα 11

### Άσκηση 5

Για το παρακάτω κύκλωμα του αποκωδικοποιητή 2 προς 4 να γράψετε τον κώδικα VHDL που περιγράφει τη λειτουργία του.

Ο κώδικας σε VHDL που περιγράφει τον παραπάνω αποκωδικοποιητή 2 προς 4 (µε χρήση της **WITH** – **SELECT**) είναι:

-- κώδικας αποκωδικοποιητή 2 προς 4

**LIBRARY** ieee ;

**USE** ieee.std\_logic\_1164.**all** ;

**ENTITY** dec2to4 **IS**

**PORT** ( w : **IN** STD\_LOGIC\_VECTOR(1 DOWNTO 0) ;

En : **IN** STD\_LOGIC ;

y : **OUT** STD\_LOGIC\_VECTOR(0 TO 3) ) ;

**END** dec2to4 ;

**ARCHITECTURE** Behavior **OF** dec2to4 **IS**

**SIGNAL** Enw : STD\_LOGIC\_VECTOR(2 **DOWNTO** 0) ;

**BEGIN**

Enw <= En & w ;

**WITH** Enw SELE**C**T

y <= "1000" **WHEN** "100",

"0100" **WHEN** "101",

"0010" **WHEN** "110",

"0001" **WHEN** "111",

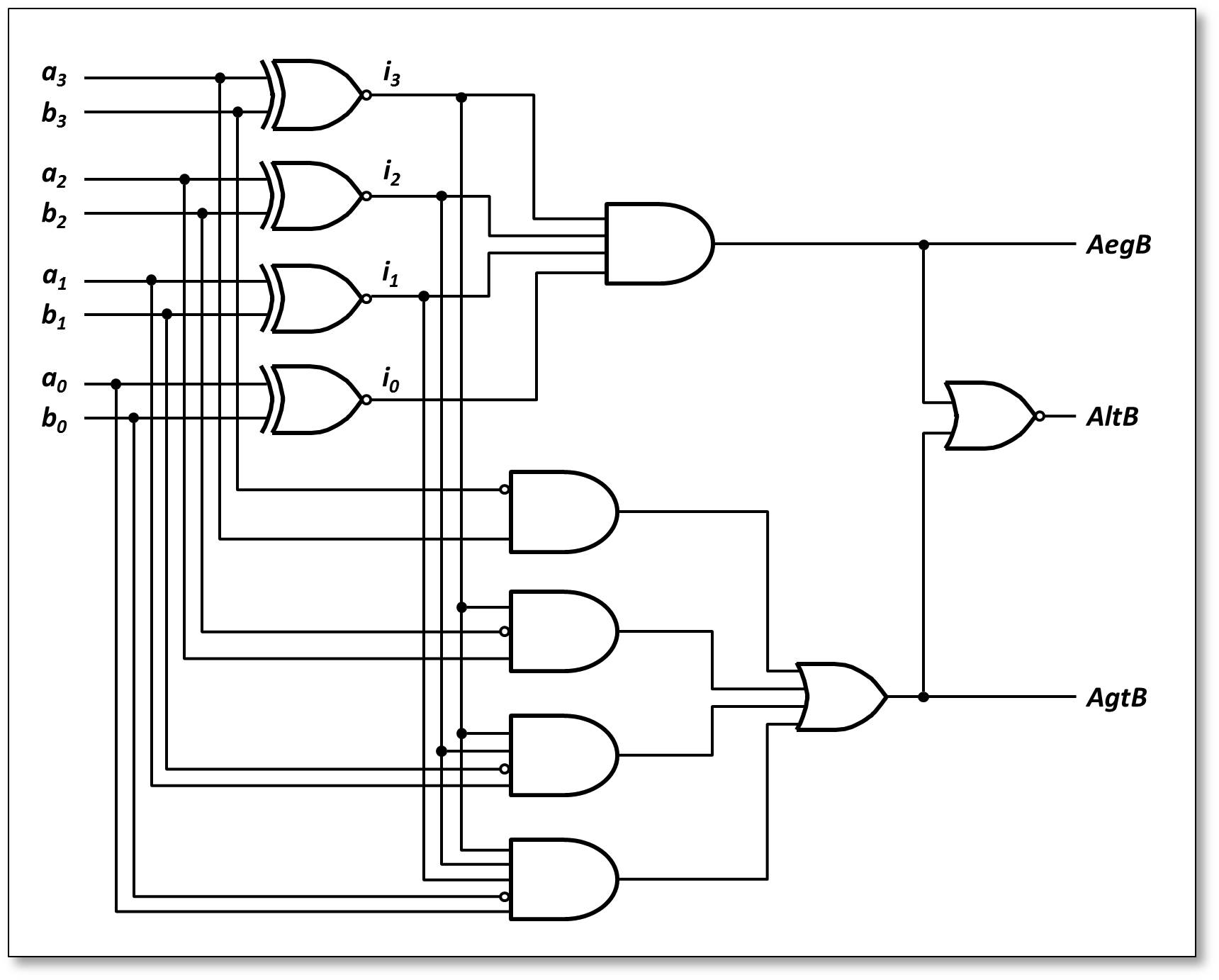
"0000" **WHEN** OTHERS ;

**END** Behavior ;

## Συγκριτής 4bits (4-bits comparator)

Το κύκλωμα ενός συγκριτή (comparator) είναι χρήσιμο στη σύγκριση μεγεθών δυαδικών αριθμών.

Ακολουθεί το λογικό κύκλωμα ενός 4bit συγκριτή με δύο εισόδους (A, B) μεγέθους 4bits, και τρεις εξόδους (για τον μεγαλύτερο A>B, μικρότερο A<B, και ίσα A=B).



Εικόνα 12

### Άσκηση 6

Για το παραπάνω κύκλωμα του 4bit συγκριτή, να γράψετε τον κώδικα VHDL που περιγράφει τη λειτουργία του.

Ο κώδικας σε VHDL που περιγράφει τον παραπάνω 4bit συγκριτή είναι:

-- κώδικας του 4bit συγκριτή

**LIBRARY** ieee ;

**USE** ieee.std\_logic\_1164.**all** ;

**USE** ieee.std\_logic\_unsigned.**all** ; -- εφόσον τα σήματα A και B μπορούν να θεωρηθούν ως μη προσημασμένοι δυαδικοί αριθμοί.

**ENTITY** compare **IS**

**PORT** ( A, B : **IN** STD\_LOGIC\_VECTOR(3 **DOWNTO** 0) ;

AeqB, AgtB, AltB: **OUT** STD\_LOGIC ) ;

**END** compare ;

**ARCHITECTURE** Behavior **OF** compare **IS**

**BEGIN**

AeqB <= '1' **WHEN** A = B ELSE '0' ;

AgtB <= '1' **WHEN** A > B ELSE '0' ;

AltB <= '1' **WHEN** A < B ELSE '0' ;

**END** Behavior ;

**Τέλος Ενότητας**

Επεξεργασία: Σοφιανίδου Γεωργία

[](http://creativecommons.org/licenses/by-nc-sa/4.0/deed.el)[](http://www.edulll.gr/)

# Σημειώματα

**Σημείωμα Ιστορικού Εκδόσεων Έργου**

Το παρόν έργο αποτελεί την έκδοση 1.01.

**Σημείωμα Αναφοράς**

Copyright Τεχνολογικό Εκπαιδευτικό Ίδρυμα Θεσσαλίας, Πετρέλλης Χ. Νικόλαος, 2015. Πετρέλλης Χ. Νικόλαος. «Αρχιτεκτονική Η/Υ ΙΙ». Έκδοση: 1.0. Λάρισα 01/03/2015. Διαθέσιμο από τη δικτυακή διεύθυνση: <http://cdev.teilar.gr/courses/TMA112/index.php>.

**Σημείωμα Αδειοδότησης**

Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά Δημιουργού - Μη Εμπορική Χρήση - Παρόμοια Διανομή 4.0 [1] ή μεταγενέστερη, Διεθνής Έκδοση. Εξαιρούνται τα αυτοτελή έργα τρίτων π.χ. φωτογραφίες, διαγράμματα κ.λπ., τα οποία εμπεριέχονται σε αυτό και τα οποία αναφέρονται μαζί με τους όρους χρήσης τους στο «Σημείωμα Χρήσης Έργων Τρίτων».

[](http://creativecommons.org/licenses/by-nc-sa/4.0/deed.el)

[1] [http://creativecommons.org/licenses/by-nc-sa/4.0/](http://creativecommons.org/licenses/by-nc-sa/4.0/deed.el)

Ως **Μη Εμπορική** ορίζεται η χρήση:

που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο,

που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο,

που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο.

Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.

**Σημείωμα Χρήσης Έργων Τρίτων**

Το Έργο αυτό κάνει χρήση των ακόλουθων έργων:

Οι παρούσες ασκήσεις βασίζονται στις εργαστηριακές

ασκήσεις που είχε προετοιμάσει για το μάθημα της Αρχιτεκτονικής ΗΥ ΙΙ ο

καθηγητής Γεώργιος Αδάμ.

**Διατήρηση Σημειωμάτων**

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

* το Σημείωμα Αναφοράς,
* το Σημείωμα Αδειοδότησης,
* τη Δήλωση Διατήρησης Σημειωμάτων,
* το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει).

μαζί με τους συνοδευόμενους υπερσυνδέσμους.