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

Αρχιτεκτονική Η/Υ ΙΙ

**Ασκήσεις Πράξης:** Ενότητα 4

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/)

**Σκοποί ενότητας**

Σε συνέχεια των προηγούμενων ενοτήτων, και 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 ακολουθιακών κυκλωμάτων.

**Περιεχόμενα**

[1. Ακολουθιακά κυκλώματα 4](#_Toc411445539)

[1.1. Χρονιζόμενος μανδαλωτής D (*D – gate latch*) 4](#_Toc411445540)

[1.1.1. Άσκηση 1 5](#_Toc411445541)

[1.2. Καταχωρητές (*registers*) 6](#_Toc411445542)

[1.2.1. Άσκηση 2 6](#_Toc411445543)

[1.2.2. Άσκηση 3 8](#_Toc411445544)

[1.2.3. Άσκηση 4 9](#_Toc411445545)

[1.3. Μετρητές (*Counters*) 11](#_Toc411445546)

[1.3.1. Άσκηση 5 11](#_Toc411445547)

[Σημειώματα 14](#_Toc411445548)

**Πίνακες**

[*Πίνακας 1* 4](#_Toc411445549)

**Εικόνες**

[*Εικόνα 1* 4](#_Toc411445550)

[*Εικόνα 2* 6](#_Toc411445551)

[*Εικόνα 3* 7](#_Toc411445552)

[*Εικόνα 4* 10](#_Toc411445553)

[*Εικόνα 5* 12](#_Toc411445554)

[*Εικόνα 6* 12](#_Toc411445555)

[*Εικόνα 7* 13](#_Toc411445556)

# Ακολουθιακά κυκλώματα

Γνωρίζουμε ότι τα ακολουθιακά κυκλώματα (sequential circuits) είναι τα κυκλώματα στα οποία οι τιμές των εξόδων, εξαρτώνται όχι µόνο από τις τρέχουσες τιμές των σημάτων που εφαρμόζονται στις εισόδους, αλλά και από την προηγούμενη συμπεριφορά του κυκλώματος. Τέτοια κυκλώματα είναι τα στοιχεία αποθήκευσης (που αποθηκεύουν τις τιμές λογικών σημάτων), τα περιεχόμενα των οποίων αντιπροσωπεύουν την κατάσταση (state) του κυκλώματος. Όταν αλλάξουν οι τιμές των εισόδων του κυκλώματος, οι νέες τιμές μπορεί να διατηρήσουν το κύκλωμα στην υπάρχουσα κατάσταση, ή να το οδηγήσουν σε μία νέα.

Ορισμένα από τα ακολουθιακά κυκλώματα που παρουσιάζονται εδώ είναι: στοιχεία

μνήμης όπως χρονιζόμενοι μανδαλωτές (D gated latches), καταχωρητές (registers),

ολισθητής (shifter), μετρητής (counter).

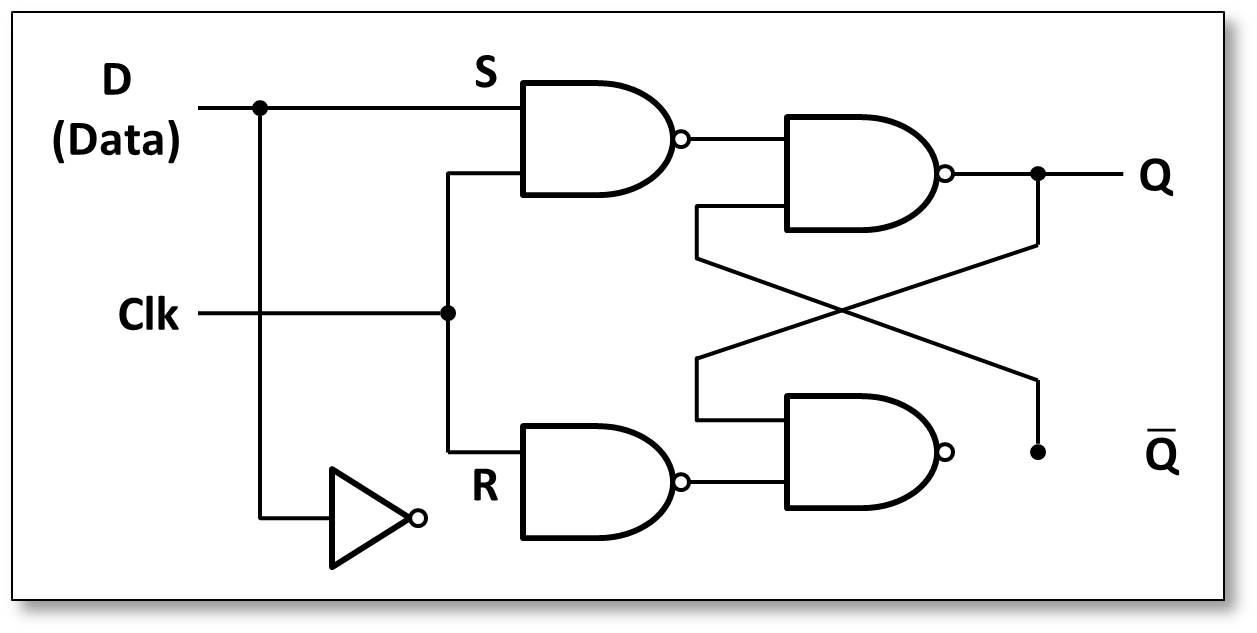
## Χρονιζόμενος μανδαλωτής D (D – gate latch)

Ο χρονιζόμενος μανδαλωτής D είναι ένα στοιχείο μνήμης με μία είσοδο δεδομένων D, ο οποίος αποθηκεύει την τιμή που υπάρχει σε αυτήν, υπό τον έλεγχο ενός ωρολογιακού σήματος. Στο χρονιζόμενο μανδαλωτή D, η έξοδος Q, απλά παρακολουθεί την τιμή της εισόδου D όταν είναι Clk=1. Εάν η τιμή Clk γίνει ίση με 0, τότε ο χρονιζόμενος μανδαλωτής D ‘παγώνει’ έως ότου το σήμα Clk γίνει ξανά 1. Επομένως, ο χρονιζόμενος μανδαλωτής D, αποθηκεύει την τιμή της εισόδου του D, τη στιγμή που ο ωρολογιακός παλμός από 1 γίνεται 0.

Ακολουθεί το λογικό κύκλωμα και ο πίνακας αληθείας του D χρονιζόμενου μανδαλωτή.

Πίνακας 1

| Clk | D | Q(t+1) |
| --- | --- | --- |
| 0 | x | Q(t) |
| 1 | 0 | 0 |
| 1 | 1 | 1 |



Εικόνα 1

Όταν η είσοδος Clk είναι ενεργοποιημένη (Clk = 1), τότε εάν είναι D=1 ο μανδαλωτής, μετέρχεται στην κατάσταση Q=1, αλλιώς εάν είναι D=0, τότε ο μανδαλωτής μετέρχεται στην κατάσταση Q=0.

### Άσκηση 1

Για το παραπάνω κύκλωμα του D χρονιζόμενου μανδαλωτή, να γράψετε τον κώδικα VHDL που περιγράφει τη λειτουργία του.

Ο κώδικας σε VHDL που περιγράφει τον παραπάνω D χρονιζόμενου μανδαλωτή είναι ο ακόλουθος:

-- κώδικας χρονιζόμενου μανδαλωτή D

**LIBRARY** ieee ;

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

**ENTITY** latch **IS**

**PORT** ( D, Clk : **IN** STD\_LOGIC ;

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

**END** latch ;

**ARCHITECTURE** Behavior **OF** latch **IS**

**BEGIN**

**PROCESS** ( D, Clk )

**BEGIN**

**IF** Clk = '1' **THEN**

Q <= D ;

**END IF** ;

**END PROCESS** ;

**END** Behavior ;

Αξίζει να σημειώσουμε ότι με μερικές αλλαγές στον παραπάνω VHDL κώδικα, καταλήγουμε σε ένα κώδικα που περιγράφει ένα **flip-flop τύπου D** (με θετική διέγερση μετώπου). Συγκεκριμένα, η διεργασία εδώ (process – βασικό στοιχείο σε behavioral περιγραφές) περιέχει μόνο το ωρολογιακό σήμα, επειδή είναι το μοναδικό που μπορεί να προκαλέσει αλλαγή στην έξοδο Q. Επίσης, η εντολή IF-THEN-ELSE, χρησιμοποιεί μία διαφορετική συνθήκη. Εδώ, η ιδιότητα EVENT, αναφέρεται σε οποιαδήποτε αλλαγή της κατάστασης του σήματος Clock.

Ο κώδικας αυτός έχει ως εξής:

-- κώδικας D flip-flop

**LIBRARY** ieee ;

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

**ENTITY** flipflop IS

**PORT** ( D, Clock : **IN** STD\_LOGIC ;

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

**END** flipflop ;

**ARCHITECTURE** Behavior **OF** flipflop **IS**

**BEGIN**

**PROCESS** ( Clock )

**BEGIN**

**IF** Clock'**EVENT** **AND** Clock = '1' **THEN**

Q <= D ;

**END IF** ;

**END PROCESS** ;

**END** Behavior ;

## Καταχωρητές (registers)

Όταν ένα σύνολο από n flip-flops χρησιμοποιούνται για να αποθηκεύουν n bits πληροφορίας (πχ ένα αριθμό μήκους n bits), τότε τα flip-flop αυτά συγκροτούν ένα καταχωρητή. Τα flip-flop ενός καταχωρητή χρησιμοποιούν ένα κοινό ωρολογιακό σήμα.

Συχνά στη σχεδίαση ενός κυκλώματος, μπορούν να χρησιμοποιηθούν και έτοιμα προκατασκευασμένα κυκλώματα, τα οποία μπορούμε να εισάγουμε από κάποια βιβλιοθήκη. Πχ, έτσι και στο περιβάλλον του λογισμικού Max+plusII, μπορούμε να χρησιμοποιήσουμε προκατασκευασμένες μονάδες καταχωρητών από μία βιβλιοθήκη παραμετροποιημένων κυκλωμάτων που ονομάζεται LPM.

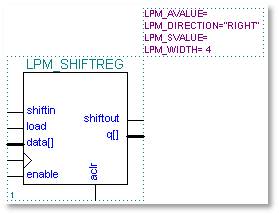
### Άσκηση 2

Εισαγωγή ενός στιγμιότυπου κυκλώματος με το όνομα lpm\_shiftreg, το οποίο είναι ένας καταχωρητής ολίσθησης μήκους n bits (βασικά ο αριθμός των flip- flops του καταχωρητή ολίσθησης ορίζεται ίσος µε 4), και ο κώδικας περιγραφής του σε VHDL.

Η εισαγωγή του παραπάνω κυκλώματος καταχωρητή μπορεί να γίνει επιλέγοντας

(SYMBOL ⇒ Enter Symbol) από τη βιβλιοθήκη **mega\_lpm** το όνομα του καταχωρητή

lpm\_shiftreg., και το εισάγουμε στο διάγραμμα εργασίας µας.



Εικόνα 2

Ο κώδικας σε VHDL που περιγράφει την παραπάνω μονάδα έχει ως εξής:

-- κώδικας που περιγράφει την προκατασκευασμένη μονάδα lpm\_shiftreg

**LIBRARY** ieee ;

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

**LIBRARY** mega\_lpm ; --χρήση της βιβλιοθήκης των παραμετροποιημένων μονάδων

**USE** mega\_lpm.lpm\_components.**all** ;

**ENTITY** shift **IS**

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

Reset : **IN** STD\_LOGIC ;

Shiftin, : **IN** STD\_LOGIC ;

R : **IN** STD\_LOGIC\_VECTOR(3 **DOWNTO** 0) ;

Q : **OUT** STD\_LOGIC\_VECTOR(3 **DOWNTO** 0) ) ;

**END** shift ;

**ARCHITECTURE** Structure **OF** shift **IS**

**BEGIN**

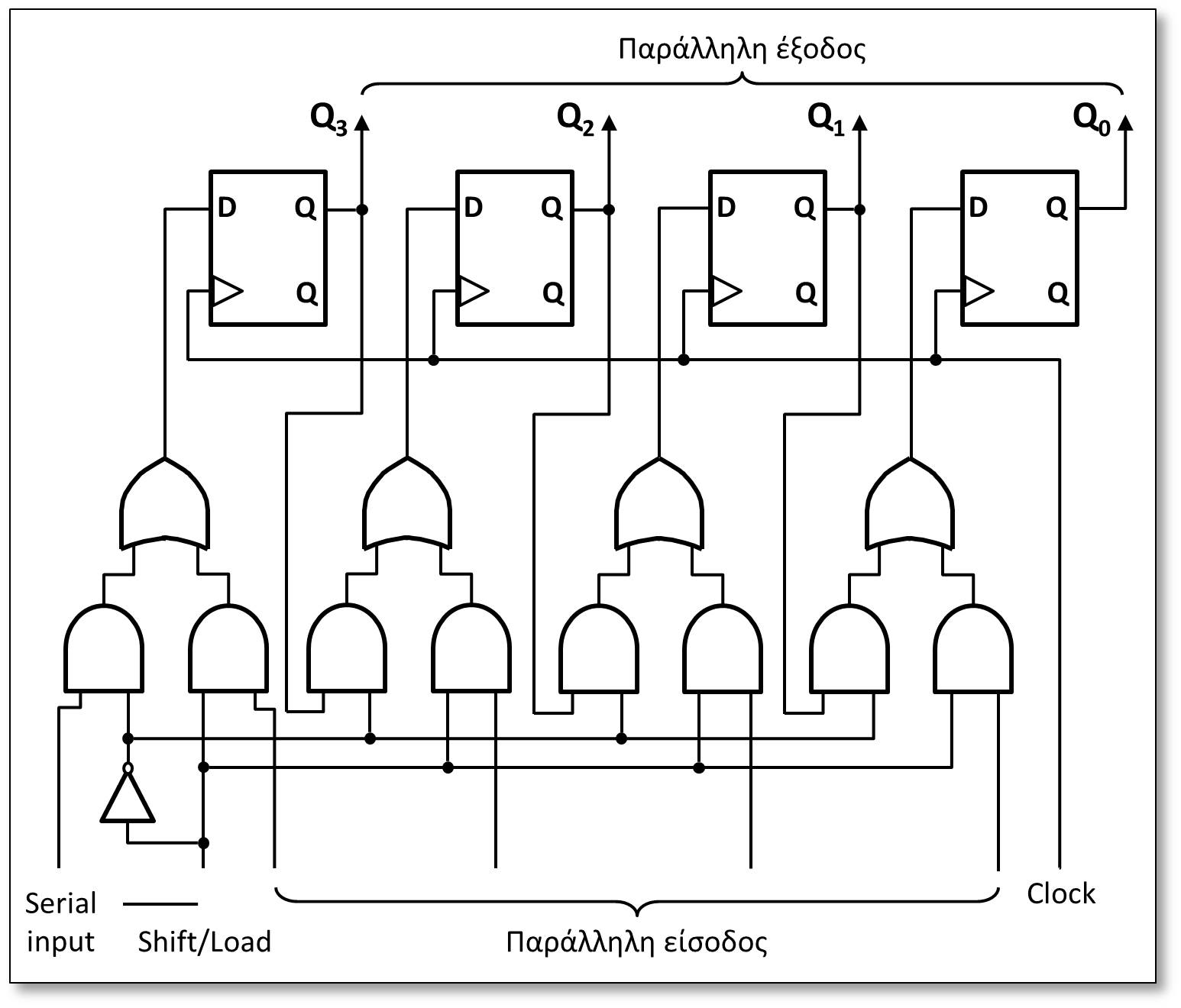
instance: lpm\_shiftreg

**GENERIC** **MAP** (LPM\_WIDTH => 4, LPM\_DIRECTION => "RIGHT")

**PORT** **MAP** (data => R, clock => Clock, aclr => Reset, load => Load, shiftin => Shiftin, q => Q ) ;

**END** Structure ;

Βασικά, η παραπάνω μονάδα όταν μεταφράζεται σε κύκλωμα, έχει την παρακάτω γνωστή δομή ενός καταχωρητή ολίσθησης παράλληλης προσπέλασης των 4bits.



Εικόνα 3

### Άσκηση 3

Να γραφεί ο κώδικας σε VHDL που περιγράφει το παραπάνω κύκλωμα του καταχωρητή ολίσθησης παράλληλης προσπέλασης των 4bits.

1. Μία πρώτη εκδοχή του κώδικα VHDL που περιγράφει το παραπάνω κύκλωμα του καταχωρητή ολίσθησης παράλληλης προσπέλασης των 4bits, έχει ως εξής:

-- ιεραρχικός κώδικας του καταχωρητή ολίσθησης 4bits (4-bits shift register)

**LIBRARY** ieee ;

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

**ENTITY** shift4 **IS**

**PORT** ( R : **IN** STD\_LOGIC\_VECTOR(3 **DOWNTO** 0) ;

L, w, Clock : **IN** STD\_LOGIC ;

Q : **BUFFER** STD\_LOGIC\_VECTOR(3 **DOWNTO** 0) ) ;

**END** shift4 ;

**ARCHITECTURE** Structure **OF** shift4 **IS**

**COMPONENT** muxdff -- το υποκύκλωμα

**PORT** (D0, D1, Sel, Clock : **IN** STD\_LOGIC ;

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

**END COMPONENT** ;

**BEGIN**

Stage3: muxdff **PORT MAP** ( w, R(3), L, Clock, Q(3) ) ;

Stage2: muxdff **PORT MAP** ( Q(3), R(2), L, Clock, Q(2) ) ;

Stage1: muxdff **PORT MAP** ( Q(2), R(1), L, Clock, Q(1) ) ;

Stage0: muxdff **PORT MAP** ( Q(1), R(0), L, Clock, Q(0) ) ;

**END** Structure ;

1. Μία δεύτερη εκδοχή του κώδικα VHDL που περιγράφει το παραπάνω κύκλωμα του καταχωρητή ολίσθησης των 4bits, δεν χρησιμοποιεί υποκυκλώματα, και έχει ως εξής (χρήση της **WAIT-UNTIL**):

**LIBRARY** ieee ;

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

**ENTITY** shift4 **IS**

**PORT** ( R : **IN** STD\_LOGIC\_VECTOR(3 **DOWNTO** 0) ;

Clock : **IN** STD\_LOGIC ;

L, w : **IN** STD\_LOGIC ;

Q : **BUFFER** STD\_LOGIC\_VECTOR(3 **DOWNTO** 0) ) ;

**END** shift4 ;

**ARCHITECTURE** Behavior **OF** shift4 **IS**

**BEGIN**

**PROCESS**

**BEGIN**

**WAIT UNTIL** Clock'**EVENT** **AND** Clock = '1' ;

**IF** L = '1' **THEN**

Q <= R ;

**ELSE**

Q(0) <= Q(1) ;

Q(1) <= Q(2);

Q(2) <= Q(3) ; Q(3) <= w ;

**END IF** ;

**END PROCESS** ;

**END** Behavior ;

1. Μία τρίτη εκδοχή σε κώδικα VHDL που περιγράφει το παραπάνω κύκλωμα του καταχωρητή ολίσθησης των 4bits, μπορεί να χρησιμοποιηθεί για να περιγράψει καταχωρητές ολίσθησης οποιουδήποτε μεγέθους (πχ µε αρχική τιμή του N ίση µε 8) , έχει ως εξής:

**LIBRARY** ieee ;

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

**ENTITY** shiftn **IS**

**GENERIC** ( N : **INTEGER** := 8 ) ;

**PORT** (R : **IN** STD\_LOGIC\_VECTOR(N-1 **DOWNTO** 0) ;

Clock : **IN** STD\_LOGIC ;

L, w : **IN** STD\_LOGIC ;

Q : **BUFFER** STD\_LOGIC\_VECTOR(N-1 **DOWNTO** 0) ) ;

**END** shiftn ;

**ARCHITECTURE** Behavior **OF** shiftn **IS**

**BEGIN**

**PROCESS**

**BEGIN**

**WAIT UNTIL** Clock'**EVENT** **AND** Clock = '1' ;

**IF** L = '1' **THEN**

Q <= R ;

**ELSE**

Genbits: **FOR** i **IN** 0 **TO** N-2 **LOOP**

Q(i) <= Q(i+1) ;

**END LOOP** ; Q(N-1) <= w ;

**END** **IF** ;

**END PROCESS** ;

**END** Behavior ;

### Άσκηση 4

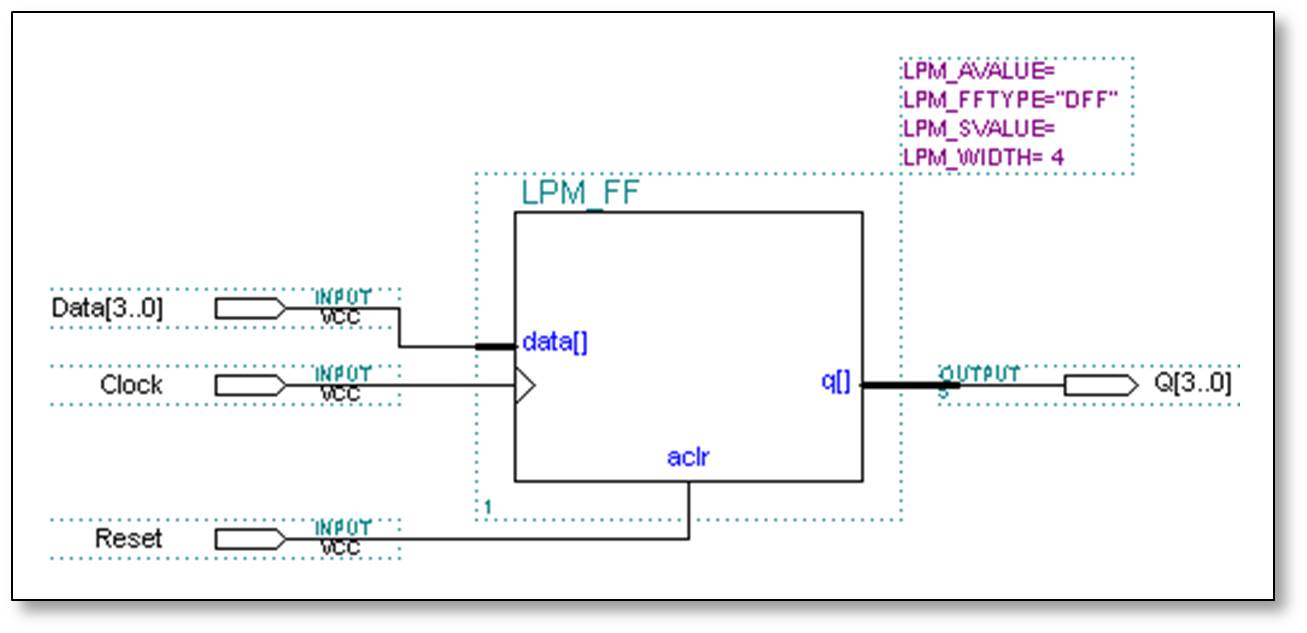
Εισαγωγή ενός κυκλώματος µε το όνομα lpm\_ff, το οποίο είναι ένας καταχωρητής με ένα ή περισσότερα flip-flops (με διέγερση μετώπου – edge triggered) τα οποία μπορεί να είναι τύπου D ή T.

Η εισαγωγή του παραπάνω κυκλώματος καταχωρητή μπορεί να γίνει επιλέγοντας

(SYMBOL ⇒ Enter Symbol) από τη βιβλιοθήκη **mega\_lpm** το όνομα του καταχωρητή

lpm\_ ff, για την εισαγωγή του στο διάγραμμα εργασίας μας. Το κύκλωμα αυτό όπως

αναφέραμε έχει παραμέτρους που επιτρέπουν την επιλογή του αριθμού και του τύπου των flip-flops. Στην περίπτωση της άσκησης επιλέγουμε 4 D flip-flops.



Εικόνα 4

Ένας άλλος τρόπος φυσικά για να σχεδιάσουμε και να περιγράψουμε ένα τέτοιο καταχωρητή, είναι να εισάγουμε την περιγραφή του σε κώδικα VHDL. Μία τέτοια περιγραφή ενός καταχωρητή μήκους 4-bits με ασύγχρονη είσοδο μηδενισμού (4-bit register with asynchronous clear), παρουσιάζεται στο πρόγραμμα που ακολουθεί.

-- κώδικας του 4-bits καταχωρητή με ασύγχρονη είσοδο μηδενισμού

**LIBRARY** ieee ;

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

**ENTITY** reg4 **IS**

**PORT** ( D : **IN** STD\_LOGIC\_VECTOR(3 **DOWNTO** 0) ;

Resetn, Clock : **IN** STD\_LOGIC ;

Q : **OUT** STD\_LOGIC\_VECTOR(3 **DOWNTO** 0) ) ;

**END** reg4 ;

**ARCHITECTURE** Behavior **OF** reg4 **IS**

**BEGIN**

**PROCESS** ( Resetn, Clock )

**BEGIN**

**IF** Resetn = '0' **THEN**

Q <= "00000000" ;

**ELSIF** Clock'**EVENT** **AND** Clock = '1' **THEN**

Q <= D ;

**END IF** ;

**END PROCESS** ;

**END** Behavior ;

Εφόσον βέβαια τα λογικά κυκλώματα απαιτούν συχνά καταχωρητές διαφορετικού μεγέθους, μπορούμε να ορίσουμε μία οντότητα καταχωρητή της οποίας ο αριθμός των flip-flops να μπορεί να αλλαχθεί εύκολα (n-bit register with asynchronous clear). Ένας τρόπος να γίνει αυτό, φαίνεται στην άσκηση που ακολουθεί, το οποίο περιλαμβάνει και μία παράμετρο (Generic N) με την οποία καθορίζεται ο αριθμός των flip-flops. Συγκεκριμένα στην άσκηση αυτή, το N του n-bit καταχωρητή έχει τεθεί ίσο µε 16.

-- κώδικας του n-bit καταχωρητή µε ασύγχρονη είσοδο μηδενισμού

**LIBRARY** ieee ;

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

**ENTITY** regn **IS**

**GENERIC** ( N : **INTEGER** := 16 ) ;

**PORT** ( D : **IN** STD\_LOGIC\_VECTOR(N-1 **DOWNTO** 0);

Resetn, Clock : **IN** STD\_LOGIC ;

Q : **OUT** STD\_LOGIC\_VECTOR(N-1 **DOWNTO** 0));

**END** regn ;

**ARCHITECTURE** Behavior **OF** regn **IS**

**BEGIN**

**PROCESS** ( Resetn, Clock )

**BEGIN**

**IF** Resetn = '0' **THEN**

Q <= (OTHERS => '0') ;

**ELSIF** Clock'**EVENT** **AND** Clock = '1' **THEN**

Q <= D ;

**END IF** ;

**END PROCESS** ;

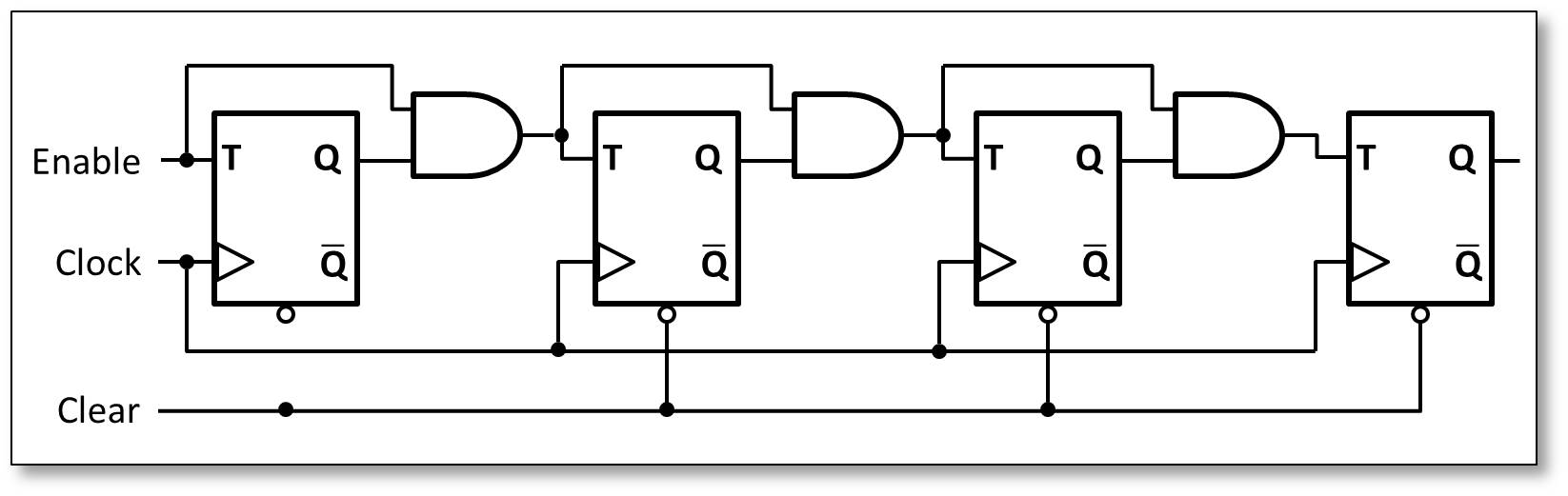
**END** Behavior ;

## Μετρητές (Counters)

Τα κυκλώματα των μετρητών, χρησιμοποιούνται στα ψηφιακά συστήματα για την απαρίθμηση ή μέτρηση της εμφάνισης κάποιων γεγονότων (αυξάνουν / μειώνουν), για να δημιουργούν χρονικά διαστήματα για τον έλεγχο διαφόρων εργασιών ενός συστήματος, για να παρακολουθούν το χρόνο που έχει μεσολαβήσει ανάμεσα σε συγκεκριμένα γεγονότα, κλπ. Τα κυκλώματα αυτά, θα μπορούσαν να σχεδιαστούν τόσο με τη βοήθεια των κυκλωμάτων αθροιστή/αφαιρέτη και των καταχωρητών, όσο και πιο απλά με τη βοήθεια T και D flip-flops.

### Άσκηση 5

Δίνεται το λογικό κύκλωμα ενός 4-bit σύγχρονου μετρητή (4-bit synchronous up- counter) από T flip-flops, το οποίο λειτουργεί ως αύξοντας μετρητής χωρητικότητας 16 (όλες οι αλλαγές λαμβάνουν χώρα με την ίδια καθυστέρηση μετά το ενεργό μέτωπο του ωρολογιακού παλμού).

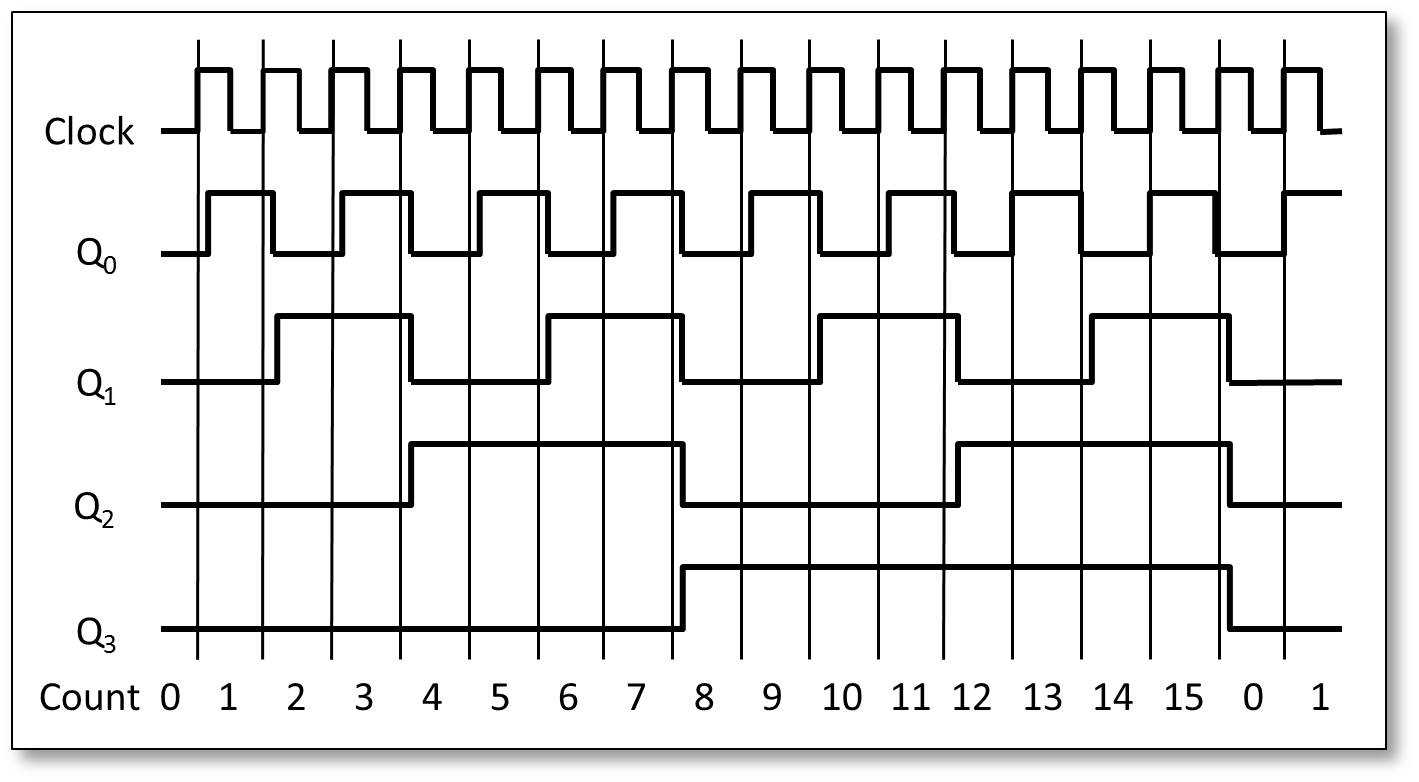


Εικόνα 5

Η είσοδος μηδενισμού (Clear) των flip-flops επιτρέπει την εκκίνηση της μέτρησης από το 0, ενώ η είσοδος ελέγχου (Enable) ενεργοποιεί το κύκλωμα με τέτοιο τρόπο, ώστε ο μετρητής να παραμένει στην τρέχουσα κατάσταση. Θέτοντας Enable =1, τότε ο μετρητής λειτουργεί ως αύξοντας μετρητής.

Να μελετήσετε το διάγραμμα χρονισμού του κυκλώματος σε αυτή την περίπτωση, και να γράψετε τον κώδικα σε VHDL που το περιγράφει.

Το διάγραμμα χρονισμού σε αυτή την περίπτωση έχει ως εξής:



Εικόνα 6

Ο κώδικας σε VHDL που περιγράφει το παραπάνω κύκλωμα του μετρητή έχει ως εξής:

-- κώδικας ενός 4-bit αύξοντα σύγχρονου μετρητή

**LIBRARY** ieee ;

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

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

**ENTITY** upcount **IS**

**PORT** ( Clock, Resetn, E: **IN** STD\_LOGIC ;

Q : **OUT** STD\_LOGIC\_VECTOR (3 **DOWNTO** 0)) ;

**END** upcount ;

**ARCHITECTURE** Behavior **OF** upcount **IS**

**SIGNAL** Count : STD\_LOGIC\_VECTOR (3 **DOWNTO** 0) ;

**BEGIN**

**PROCESS** ( Clock, Resetn )

**BEGIN**

**IF** Resetn = '0' **THEN**

Count <= "0000" ;

**ELSIF** (Clock'**EVENT** **AND** Clock = '1') **THEN**

**IF** E = '1' **THEN**

Count <= Count + 1 ;

**ELSE**

Count <= Count ;

**END IF** ;

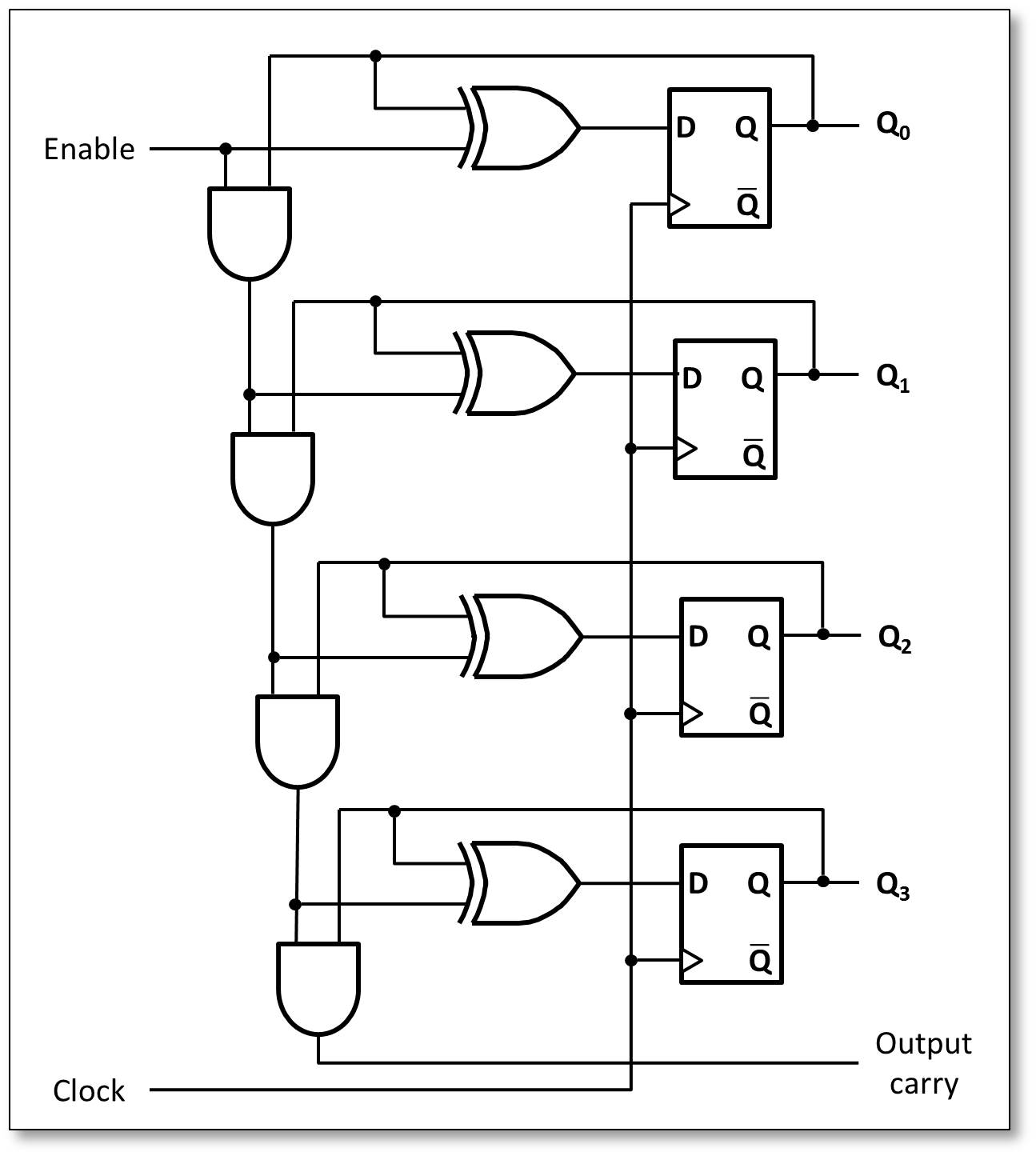
**END IF** ;

**END PROCESS** ;

Q <= Count ;

**END** Behavior ;

Πρέπει να σημειώσουμε ότι εάν ο μεταφραστής της γλώσσας VHDL επιλέξει να χρησιμοποιήσει D flip-flops (αντί για T flip-flops) τότε ο κώδικας θα αντιστοιχεί στο παρακάτω κύκλωμα (με μία επιπρόσθετη είσοδο μηδενισμού):



Εικόνα 7

**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.

**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.