ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

Ψηφιακή Λογική Σχεδίαση

Επιμέλεια:

Γεώργιος Θεοδωρίδης, Επίκουρος Καθηγητής

Ανδρέας Εμερετλής, Υποψήφιος Διδάκτορας

Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

Άδειες Χρήσης

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

- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

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

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.

Ανάπτυξη

Το παρόν εκπαιδευτικό υλικό αναπτύχθηκε στο Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών του Πανεπιστημίου Πατρών.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

ΑΣΚΗΣΗ 1

Α. Σχεδιάστε έναν καταχωρητή τεσσάρων δυαδικών ψηφίων (Q_3,Q_2,Q_1,Q_0) η λειτουργία του οποίου καθορίζεται από 2 εισόδους ελέγχου c_0, και c_1 όπως περιγράφεται στον παρακάτω πίνακα. Έχετε στη διάθεση σας D Flip-Flops (FFs), πολυπλέκτες και πύλες.

c_1	c_0	**Λειτουργία Καταχωρητή**
0	**0**	Συγκράτηση δεδομένων δηλαδή, (Q_3,Q_2,Q_1,Q_0,) = (Q_3,Q_2,Q_1,Q_0,)
0	**1**	Παράλληλη φόρτωση δεδομένων από εξωτερικές εισόδους I_0, I_1, I_2 και I_3. δηλαδή, (Q_3,Q_2,Q_1,Q_0,) = (I_3,I_2,I_1,I_0,)
1	**0**	Αντιστροφή δεδομένων δηλαδή, (Q_3,Q_2,Q_1,Q_0,) = (Q'_3,Q'_2,Q'_1,Q'_0,)
1	**1**	Αριστερή κυκλική ολίσθηση

Β. Τροποποιήστε τον παραπάνω καταχωρητή ώστε να υλοποιείται με T Flip-Flops. Δείξτε την τροποποίηση για ένα μόνο ψηφίο.

ΛΥΣΗ

Α. Το παραπάνω κύκλωμα μπορεί να υλοποιηθεί με χρήση 4 καταχωρητών (4 ψηφίων ο καθένας), όπου ο κάθε καταχωρητής θα επιτελούσε μία από τις τέσσερις λειτουργίες του πίνακα. Όμως, μια τέτοια υλοποίηση είναι δαπανηρή από άποψη επιφάνειας καθώς απαιτούνται 4x4=16 FFs. Συνεπώς, πρέπει να βρεθεί μία πιο αποδοτική λύση.

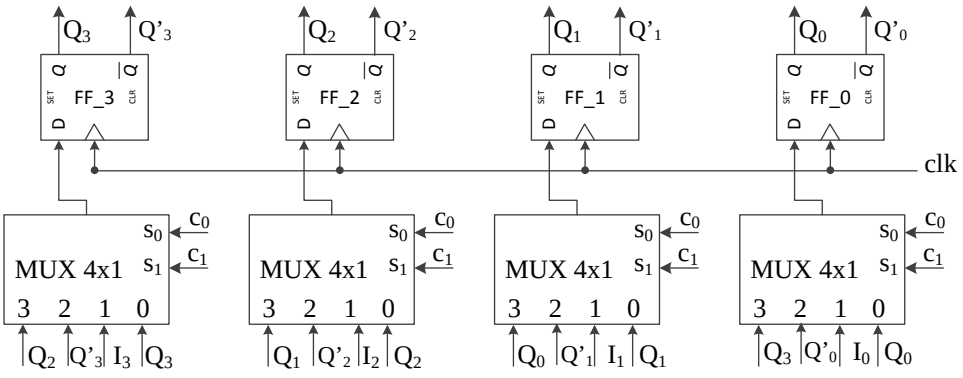
Η αποδοτική λύση συνιστάται στη χρήση ενός καταχωρητή 4 ψηφίων (άρα 4 FFs) και πολυπλεκτών κατάλληλα συνδεδεμένων ώστε να παρέχουν για κάθε λειτουργία τη σωστή είσοδο στα FFs του καταχωρητή. Έτσι, το κύκλωμα θα περιλαμβάνει έναν καταχωρητή αποτελούμενο από 4 D FFs, όπου η είσοδος, D, του κάθε FF θα παρέχεται από έναν πολυπλέκτη. Όσον αφορά τον κάθε πολυπλέκτη, όπως εξηγείται αναλυτικά στη συνέχεια, αυτός θα είναι ένας 4-σε-1 πολυπλέκτης όπου τα σήματα επιλογής του s_1 και s_0 είναι τα σήματα c_1 και c_0, αντίστοιχα δηλαδή, (s_1,s_0) = (c_1,c_0).

Μελετώντας τον πίνακα λειτουργίας παρατηρούμε τις ακόλουθες περιπτώσεις:

- Όταν (c_1,c_0) = (0,0) πρέπει να γίνεται συγκράτηση των δεδομένων του καταχωρητή δηλαδή, $(Q_3^t, Q_2^t, Q_1^t, Q_0^t) = (D_3^t, D_2^t, D_1^t, D_0^t) = (Q_3^{t+T}, Q_2^{t+T}, Q_1^{t+T}, Q_0^{t+T})$. Αυτό επιτυγχάνεται συνδέοντας την είσοδο Q_i στην είσοδο 0 του αντίστοιχου πολυπλέκτη καθώς όταν (c_1,c_0)=(s_1,s_0)=(0,0) η έξοδος του πολυπλέκτη είναι ίση με την είσοδο 0 αυτού.

- Όταν (c_1,c_0) = (0,1) πρέπει να γίνεται παράλληλη φόρτωση από μία εξωτερική είσοδο, I, τεσσάρων ψηφίων (I_0,I_1,I_2,I_3). Για να φορτωθεί ο καταχωρητής παράλληλα, θα πρέπει να συνδέσουμε την είσοδο D του κάθε FF με την αντίστοιχη είσοδο I_i (i=0,1,2,3) δηλαδή, (D_3,D_2,D_1,D_0)=(I_3,I_2,I_1,I_0). Αυτό επιτυγχάνεται συνδέοντας την είσοδο I_i στην είσοδο 1 του αντίστοιχου πολυπλέκτη καθώς όταν (c_1,c_0)=(s_1,s_0)=(0,1) η έξοδος του πολυπλέκτη είναι ίση με την είσοδο 1 αυτού.
- Όταν (c_1,c_0) = (1,0) πρέπει να γίνεται αντιστροφή των δεδομένων του καταχωρητή δηλαδή, $(\bar{Q}_3^t,\ \bar{Q}_2^t,\ \bar{Q}_1^t,\ \bar{Q}_0^t) = (D_3^t\ , D_2^t\ , D_1^t\ , D_0^t) = (Q_3^{t+T}, Q_2^{t+T}, Q_1^{t+T}, Q_0^{t+T})$. Αυτό επιτυγχάνεται συνδέοντας την είσοδο $\bar{Q}_i$ στην είσοδο 2 του αντίστοιχου πολυπλέκτη καθώς όταν (c_1,c_0)=(s_1,s_0)=(1,0) η έξοδος του πολυπλέκτη είναι ίση με την είσοδο 2 αυτού. Σημειώστε ότι δε χρειάζεται χρήση πυλών NOT για την αντιστροφή των δεδομένων του καταχωρητή καθώς η συμπληρωματική έξοδος Q'_i παρέχεται από κάθε FF.
- Όταν (c_1,c_0) = (1,1) τότε πρέπει να εκτελείται αριστερή κυκλική ολίσθηση. Αυτό σημαίνει ότι το ψηφία του καταχωρητή ολισθαίνουν αριστερά δηλαδή, ισχύει ότι $D_i^t = Q_{i-1}^t$ (i=1,2,3), και επιπλέον λόγω της κυκλικής ολίσθησης ισχύει $D_0^t = Q_{3.}^t$ Η λειτουργία αυτή εκτελείται όταν (c_1,c_0) = (1,1) που σημαίνει ότι η έξοδος του πολυπλέκτη είναι ίση με την είσοδο 3 αυτού αφού (c_1,c_0)=(s_1,s_0)=(1,1). Άρα, για του πολυπλέκτες που τροφοδοτούν τις εισόδους των FF_i (i=1,2.3) συνδέουμε στην είσοδο 3 του πολυπλέκτη την έξοδο του FF που βρίσκεται δεξιάτου του, αυτό αντιστοιχεί στην υλοποίηση της πράξης $D_i^t = Q_{i-1}^t$ (i=1,2,3). Τέλος, για την πράξη $D_0^t = Q_{3.}^t$ εκτελούμε την αντίστοιχη σύνδεση.

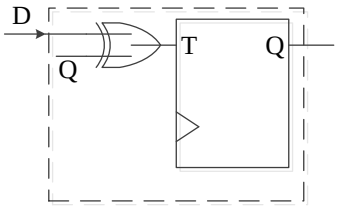
Με βάση τις παραπάνω παρατηρήσεις το αντίστοιχο κύκλωμα είναι το παρακάτω



Β. Όσον την υλοποίηση με T FF, αυτό που στην ουσία ζητείται είναι ένα κύκλωμα που να υλοποιεί τη λειτουργία του D FF με T FF και πύλες. Αν γίνει αυτό, τότε το κάθε FF του κυκλώματος του προηγούμενου ερωτήματος αντικαθίσταται από το νέο κύκλωμα. Για να υλοποιήσουμε το D FF με T FF και πύλες εργαζόμαστε σύμφωνα με τη θεωρία σχεδίασης ακολουθιακών κυκλωμάτων. Συγκεκριμένα έχουμε τον ακόλουθο πίνακ λειτουργίας

Είσοδος (D^t)	**Παρούσα Κατάσταση (Q^t)**	**Επόμενη Κατάσταση (Q^{t+T})**	**Είσοδος (T^t)**
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Δεδομένου ότι θέλουμε να υλοποιήσουμε τη λειτουργία του D FF, η επόμενη κατάσταση είναι ίση με την είσοδο D. Τέλος, με βάση τις τιμές των στηλών παρούσα και επόμενη κατάσταση προκύπτουν οι τιμές για την είσοδο T του T FF. Με βάση τον παραπάνω πίνακα προκύπτει εύκολα ότι $T^t = D^t$ XOR Q^t. Έτσι, το αντίστοιχο κύκλωμα που αντικαθιστά το κάθε FF του ερωτήματος Α είναι το ακόλουθο



ΑΣΚΗΣΗ 2

Σχεδιάστε έναν καταχωρητή τεσσάρων δυαδικών ψηφίων (Q_3,Q_2,Q_1,Q_0) η λειτουργία του οποίου καθορίζεται από 3 εισόδους ελέγχου c_0, c_1 και c_2 όπως περιγράφεται στον ακόλουθο πίνακα, όπου X σημαίνει αδιάφορη δυαδική τιμή. Θεωρείστε ότι οι αριθμοί είναι σε αναπαράσταση συμπληρώματος τους 2 και για τις λειτουργίες της πρόσθεσης και της αφαίρεσης αγνοήστε το κρατούμενο εξόδου. Έχετε στη διάθεση σας D Flip-Flops (FFs), πολυπλέκτες, κυκλώματα αθροιστών του ενός ψηφίου και πύλες.

c_2	c_1	c_0	**Λειτουργία Καταχωρητή**
X	**0**	**0**	Παράλληλη φόρτωση δεδομένων από εξωτερικές εισόδους I_0, I_1, I_2 και I_3.
X	**0**	**1**	Δεξιά κυκλική ολίσθηση
0	**1**	**0**	Εκτέλεση της πράξης Q=Q+I δηλαδή, (Q_3,Q_2,Q_1,Q_0,) = (Q_3,Q_2,Q_1,Q_0,) + (I_3,I_2,I_1,I_0,)
1	**1**	**0**	Εκτέλεση της πράξης Q=Q-I δηλαδή, (Q_3,Q_2,Q_1,Q_0,) = (Q_3,Q_2,Q_1,Q_0,) – (I_3,I_2,I_1,I_0,)
X	**1**	**1**	Σειριακή φόρτωση του αριθμού B_3,B_2,B_1,B_0 τα ψηφία του οποίου παρέχονται από μία εξωτερική είσοδο B. Θεωρείστε ότι εισέρχεται ένα ψηφίο ανά παλμό με 1ο εισερχόμενο ψηφίο το LSB δηλ., το B_0

ΛΥΣΗ

Το παραπάνω κύκλωμα μπορεί να υλοποιηθεί με χρήση 5 καταχωρητών (4 ψηφίων ο καθένας), όπου ο κάθε καταχωρητής θα επιτελούσε μία από τις πέντε λειτουργίες του πίνακα. Όμως, μια τέτοια υλοποίηση είναι δαπανηρή από άποψη επιφάνειας καθώς απαιτούνται 4x5=20 FFs. Συνεπώς, πρέπει να βρεθεί μία πιο αποδοτική λύση.

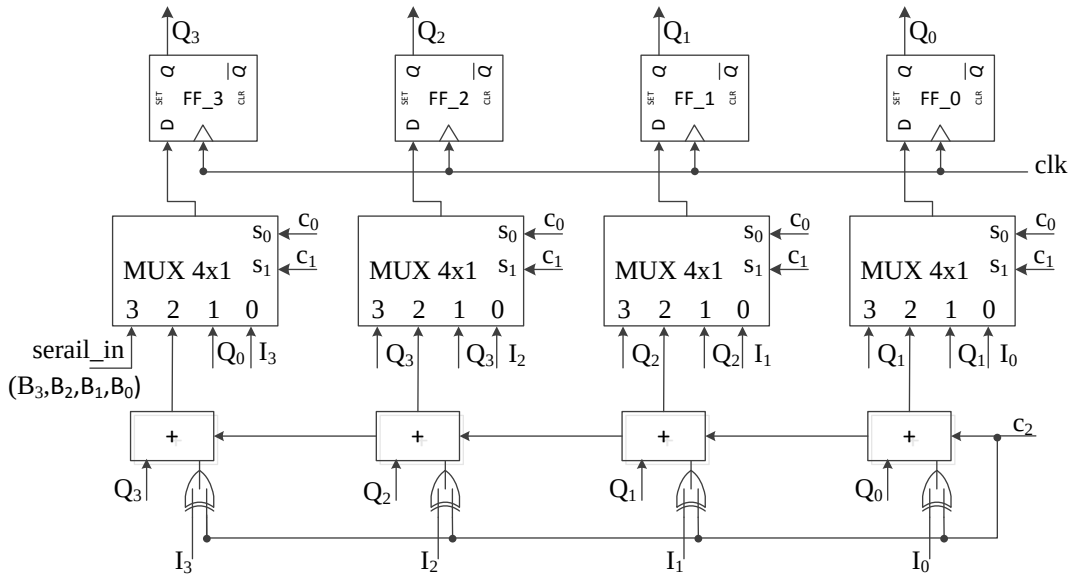
Η αποδοτική λύση συνιστάται στη χρήση ενός καταχωρητή 4 ψηφίων (άρα 4 FFs) και πολυπλεκτών κατάλληλα συνδεδεμένων ώστε να παρέχουν για κάθε λειτουργία τη σωστή είσοδο στα FFs του καταχωρητή. Έτσι, το κύκλωμα θα περιλαμβάνει έναν καταχωρητή αποτελούμενο από 4 D FFs, όπου η είσοδος, D, του κάθε FF θα παρέχεται από έναν πολυπλέκτη. Όσον αφορά τον κάθε πολυπλέκτη, όπως εξηγείται αναλυτικά στη συνέχεια, αυτός θα είναι ένας 4-σε-1 πολυπλέκτης όπου τα σήματα επιλογής του s_1 και s_0 είναι τα σήματα c_1 και c_0, αντίστοιχα δηλαδή, (s_1,s_0) = (c_1,c_0).

Μελετώντας τον πίνακα λειτουργίας παρατηρούμε τις ακόλουθες περιπτώσεις:

- Όταν (c_2,c_1,c_0) = (X,0,0) τότε πρέπει να γίνεται παράλληλη φόρτωση από μία εξωτερική είσοδο, I, τεσσάρων ψηφίων (I_0,I_1,I_2,I_3). Για να φορτωθεί ο καταχωρητής παράλληλα, θα πρέπει να συνδέσουμε την είσοδο D του κάθε FF με την αντίστοιχη είσοδο I_i (i=0,1,2,3) δηλαδή, $(D_3,D_2,D_1,D_0)=(I_3,I_2,I_1,I_0)$. Αυτό επιτυγχάνεται συνδέοντας την είσοδο I_i στην είσοδο 0 του αντίστοιχου πολυπλέκτη καθώς όταν $(c_1,c_0)=(s_1,s_0)=(0,0)$ η έξοδος του πολυπλέκτη είναι ίση με την είσοδο 0 αυτού. Παρατηρήστε ότι το σήμα c_2 μπορεί να πάρει οποιαδήποτε δυαδική τιμή (c_2 = X) και δε χρησιμοποιείται.
- Όταν (c_2,c_1,c_0) = (X,0,1) πρέπει να εκτελείται δεξιά κυκλική ολίσθηση. Αυτό σημαίνει ότι το ψηφία του καταχωρητή ολισθαίνουν δεξιά δηλαδή, $D_i = Q_{i+1}$ (i=0,1,2), και επιπλέον λόγω της κυκλικής ολίσθησης ισχύει $D_3 = Q_0$. Η λειτουργία αυτή εκτελείται όταν (c_2,c_1,c_0) = (X,0,1) που σημαίνει ότι η έξοδος του πολυπλέκτη είναι ίση με την είσοδο 1 αυτού αφού $(c_1,c_0)=(s_1,s_0)=(0,1)$. Άρα, για του πολυπλέκτες που τροφοδοτούν τις εισόδους των FF_i (i=0,1,2) συνδέουμε στην είσοδο 1 του πολυπλέκτη την έξοδο του FF που βρίσκεται αριστερά του, αυτό αντιστοιχεί στην υλοποίηση της πράξης $D_i = Q_{i+1}$ (i=0,1,2). Τέλος, για την πράξη $D_3 = Q_0$ εκτελούμε την αντίστοιχη σύνδεση. Παρατηρήστε επίσης ότι το σήμα c_2 μπορεί να πάρει οποιαδήποτε δυαδική τιμή και δε χρησιμοποιείται.
- Όταν (c_2,c_1,c_0) = (X,1,1) γίνεται σειριακή φόρτωση του καταχωρητή με 1ο εισερχόμενο ψηφίο το λιγότερο σημαντικό, δηλαδή το B_0. Δεδομένου ότι $(c_1,c_0)=(s_1,s_0)=(1,1)$, θα πρέπει να γίνει η κατάλληλη σύνδεση στην είσοδο 3 του κάθε πολυπλέκτη. Εφόσον τα δεδομένα εισέρχονται σειριακά (ένα ψηφίο σε κάθε παλμό), απαιτείται η υλοποίηση της λειτουργίας της ολίσθησης. Όμως, τίθενται δύο ζητήματα. Πρώτον ποια είναι η φορά της ολίσθησης (δεξιά ή αριστερή ολίσθηση) και δεύτερον σε ποιο FF (FF_0 ή FF_3) συνδέεται η σειριακή είσοδος. Δεδομένου, ότι το 1ο εισερχόμενο ψηφίο είναι το λιγότερο σημαντικό, τότε η σειριακή είσοδος συνδέεται στο FF_3 και η εκτελείται δεξιά ολίσθηση έτσι ώστε μετά από τέσσερις παλμούς να ισχύει $(D_3,D_2,D_1,D_0) = (B_3,B_2,B_1,B_0)$. Παρατηρήστε επίσης ότι το σήμα c_2 μπορεί να πάρει οποιαδήποτε δυαδική τιμή και δε χρησιμοποιείται.
- Όταν (c_2,c_1,c_0) = (0,1,0) εκτελείται η πράξη Q=Q+I, ενώ όταν $(c_2,c_1,c_0)=(1,1,0)$ εκτελείται η πράξη Q=Q-I. Επίσης, και για τις δύο περιπτώσεις ισχύει ότι $(c_1,c_0)=(s_1,s_0)=(1,0)$. Συνεπώς, θα πρέπει να χρησιμοποιήσουμε την είσοδο 2 του κάθε πολυπλέκτη και για τις δύο πράξεις. Μία λύση θα ήταν να σχεδιάσουμε σε ξεχωριστά κυκλώματα ένα κύκλωμα άθροισης 4 ψηφίων και ένα κύκλωμα

αφαίρεσης 4 ψηφίων και χρησιμοποιώντας πολυπλέκτες 2-σε-1 με σήμα επιλογής το c_2 να τροφοδοτήσουμε την είσοδο 2 των πολυπλεκτών 4-σε-1. Όμως, ένα τέτοιο κύκλωμα είναι δαπανηρό από άποψη επιφάνειας. Αυτό που απαιτείται είναι ό σχεδιασμός ενός μόνο κυκλώματος που θα εκτελεί πρόσθεση όταν c_2=0 ή αφαίρεση όταν c_2=1, δηλαδή ενός κυκλώματος πρόσθεσης/αφαίρεσης σε συμπλήρωμα ως προς 2. Παρατηρήστε ότι όταν c_2 = 0 εκτελείται η πράξη Q=Q+I, ενώ όταν c_2=1 εκτελείται η πράξη Q=Q-I. Για την πρόσθεση ισχύει ότι Q=Q+I=Q+I+c_{in}=Q+I+0=Q+I+c_2 αφού όταν c_2=0 εκτελεί η πράξη της πρόσθεσης. Άρα, πρέπει να υλοποιήσουμε ένα αθροιστή 4 ψηφίων με c_{in}=0. Αναλυτικότερα, για την περίπτωση της πρόσθεσης, για κάθε ψηφίο, Q_i, έχουμε $Q_i^{t+T} = Q_i^t + I_i + c_i$, όπου c_i το κρατούμενο εισόδου. Για την αφαίρεση Q=Q-I σε αναπαράσταση συμπληρώματος_ως_προς_2 ισχύει ότι Q = Q+ (συμπλήρωμα_ως_προς_2_του I) = $Q + (I' + 1) = Q + (I' + c_2)$, αφού η αφαίρεση εκτελείται όταν c_2=1. Άρα, πρέπει να υλοποιήσουμε ένα αθροιστή 4 ψηφίων με c_{in}=1. Αναλυτικότερα, για κάθε ψηφίο, Q_i, έχουμε $Q_i^{t+T} = Q_i^t + I_i' + c_i$. Με βάση τα παραπάνω παρατηρούμε ότι: α) χρειαζόμαστε έναν αθροιστή 4 ψηφίων με κρατούμενο εισόδου c_{in} = c_2 και β) για κάθε ψηφίο του αθροιστή η μία είσοδος είναι πάντα η Q_i^t ενώ η δεύτερη είσοδος είναι είτε η I_i όταν c_2=0 είτε η I_i' όταν c_2=1. Η 1η παρατήρηση υλοποιείται κάνοντας την σύνδεση c_{in} = c_2, όπου c_{in} το κρατούμενο εισόδου του αθροιστή 4 ψηφίων, ενώ η 2η υλοποιείται με μία πύλη XOR με εισόδους I_i και c_2.

Με βάση τις παραπάνω παρατηρήσεις το αντίστοιχο κύκλωμα είναι το παρακάτω



ΑΣΚΗΣΗ 3

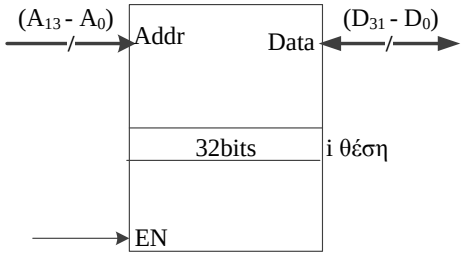
Να σχεδιαστεί σύστημα μνήμης RAM που έχει δίαυλο διευθύνσεων 14bits (A_{13}-A_0), ενώ κάθε θέση μνήμης περιλαμβάνει μία λέξη δεδομένων των 32bits (D_{31}-D_0). Έχετε στη διάθεση σας Ολοκληρωμένα Κυκλώματα (Ο.Κ.) μνήμης τύπου RAM 1Κ θέσεων (το καθένα) με οργάνωση 8bits ανά θέση, αποκωδικοποιητές, και πύλες. Πόσες θέσεις μνήμης έχει το παραπάνω σύστημα, πόσο είναι το μέγεθος του σε Bytes και ποιο είναι το πεδίο διευθύνσεων του; Πόσο είναι το μέγεθος σε Bytes του κάθε Ο.Κ. RAM και πόσα ψηφία διεύθυνσης απαιτούνται για αυτό;

ΛΥΣΗ

Εφόσον το σύστημα μνήμης έχει δίαυλο διευθύνσεων 14bits και κάθε θέση μνήμης περιλαμβάνει μία λέξη δεδομένων, τότε οι συνολικές θέσεις μνήμης είναι 2^{14}. Επιπλέον, αφού σε κάθε θέση μνήμης υπάρχει μία λέξη των 32bits δηλαδή, των 4 Bytes, το συνολικό μέγεθος του συστήματος μνήμης είναι $4 \times 2^{14} = 2^6 \times 2^{10}$ Bytes = 64KB (64 Kbytes). Επίσης, λόγω των 14 ψηφίων διεύθυνσης, το πεδίο διευθύνσεων του συστήματος είναι από $(00\ 0000\ 0000\ 0000)_2$ έως και $(11\ 1111\ 1111\ 1111)_2$ δηλαδή, 0000_{16} – $3FFF_{16}$.

Αναφορικά, με το διαθέσιμο Ο.Κ. RAM έχουμε: α) εφόσον αυτό περιλαμβάνει 1Κ θέσεις μνήμης με 8bits ανά θέση, τότε το μέγεθος του είναι 1KB (8bits = 1 Byte) και β) αφού περιλαμβάνει 1Κ θέσεις, απαιτούνται 10 ψηφία διεύθυνσης (2^{10} = 1024 = 1Κ).

Με βάση τα παραπάνω αν είχαμε διαθέσιμο ένα Ο.Κ. για το συγκεκριμένο σύστημα μνήμης, τότε αυτό θα ήταν το ακόλουθο

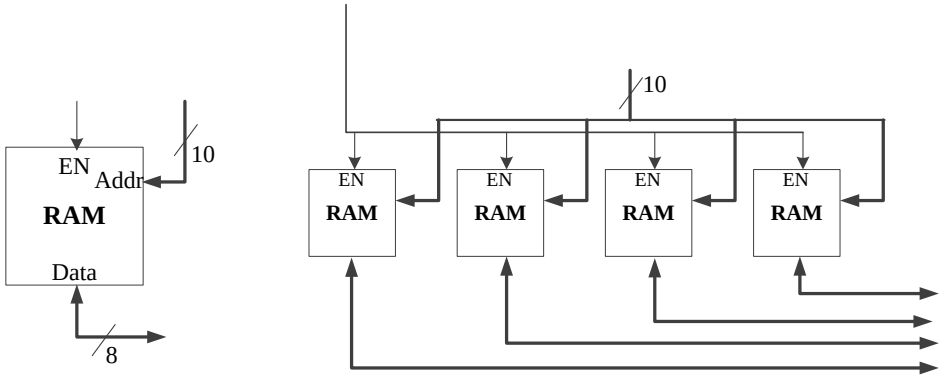


Σχ.1: Οργάνωση μνήμης υψηλού επιπέδου

Όμως, λόγω της μη διαθεσιμότητας ενός τέτοιου Ο.Κ. RAM, πρέπει να υλοποιήσουμε το σύστημα μνήμης με τα διαθέσιμα Ο.Κ. RAM.

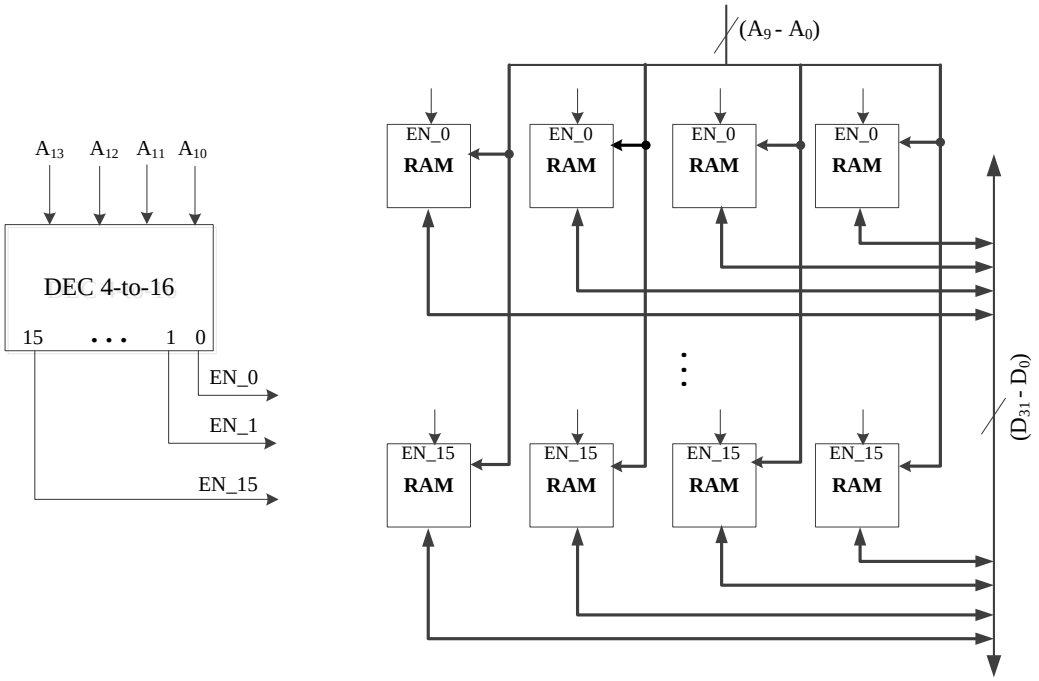
Με βάση τα παραπάνω συμπεράσματα, το κάθε Ο.Κ. RAM έχει την ακόλουθη διεπαφή (όπως φαίνεται στο Σχ.2): α) δίαυλος διευθύνσεων 10bits, β) δίαυλος δεδομένων 8bits και γ) σήμα ενεργοποίησης ολοκληρωμένου κυκλώματος, EN (Enable).

Για να δημιουργήσουμε τη λέξη των 32bits, που απαιτείται για το συνολικό σύστημα μνήμης, πρέπει να χρησιμοποιήσουμε 4 Ο.Κ. παράλληλα όπου: α) όλα έχουν το ίδιο σήμα ενεργοποίησης, EN και β) όλα δέχονται τα ίδια ψηφία διεύθυνσης, όπως δείχνεται στο επόμενο σχήμα.



Σχ.2. Οργάνωση μνήμης για την προσπέλαση μιας λέξης

Συνδέοντας, το ίδιο σήμα EN σε κάθε ένα από τα 4 Ο.Κ. RAM επιτυγχάνουμε την ταυτόχρονη ενεργοποίηση τους. Επίσης, χρησιμοποιώντας τα ίδια ψηφία διεύθυνσης, επιτυγχάνουμε την προσπέλαση της ίδιας θέσης μνήμης σε καθένα από 4 Ο.Κ. Δεδομένου, ότι κάθε μνήμης του Ο.Κ. είναι 8bits, με τον τρόπο αυτό προσπελαύνονται ταυτόχρονα 4 θέσεις με την ίδια διεύθυνση που αντιστοιχεί στην προσπέλαση μίας λέξης των 32bits (4x8=32) του συνολικού συστήματος μνήμης. Πρέπει να το τονιστεί ότι, τα 10 ψηφία διεύθυνσης του κάθε Ο.Κ. είναι τα λιγότερο σημαντικά (A_9-A_0) από τα 14 συνολικά ψηφία διεύθυνσης του όλου συστήματος. Όμως, καθώς το Ο.Κ. έχει 2^{10} θέσεις μνήμης ενώ το συνολικό σύστημα μνήμης είναι 2^{14} θέσεων, πρέπει να επαναλάβουμε την παραπάνω δομή 16 φορές ($2^{14}/2^{10} = 16$), όπως παρουσιάζεται στο Σχ.3. Συγκεκριμένα, η δομή του Σχ.2 επαναλαμβάνεται 16 φορές και κάθε οριζόντια σειρά των τεσσάρων Ο.Κ. έχει το ίδιο σήμα ενεργοποίησης (EN_0, …, EN_15).



Σχ.3: Οργάνωση συστήματος μνήμης

Όπως αναφέρθηκε, για κάθε Ο.Κ. RAM, τα 10 ψηφία διεύθυνσης είναι τα (A_9-A_0). Όμως, με αυτό τον τρόπο μπορούμε να προσπελάσουμε μόνο 2^{10} θέσεις μνήμης, ενώ οι θέσεις του συνολικού συστήματος είναι 2^{14}. Εξετάζοντας όμως αναλυτικότερα το πεδίο διευθύνσεων του όλου συστήματος, όπως παρατηρούμε ότι κάθε τμήμα μνήμης των 2^{10} θέσεων χαρακτηρίζεται μονοσήμαντα από τα 4 περισσότερο σημαντικά ψηφία διεύθυνσης, ενώ τα 12 υπόλοιπα ψηφία παίρνουν όλες τις δυνατές τιμές από 00…00 ως και 11…11.

Πίνακας 1. Πεδίο διευθύνσεων του συνολικού συστήματος

Τμήμα μνήμης (2^{10} θέσεις)	Διευθύνσεις	
	Αρχική	Τελική
0	**0000** 0000000000	**0000** 1111111111
1	**0001** 0000000000	**0001** 1111111111
…	**…**	**…**
15	**1111** 0000000000	**1111** 1111111111

Άρα, για να ενεργοποιήσουμε τα Ο.Κ. που αντιστοιχούν σε κάθε τμήμα των 2^{10} θέσεων, χρησιμοποιούμε έναν αποκωδικοποιητή 4-σε-16 με εισόδους τα 4 περισσότερο σημαντικά ψηφία διεύθυνσης (A_{13}-A_{10}). Οι έξοδοι του αποκωδικοποιητή είναι τα 16 σήματα επιλογής, EN (EN_0, …, EN_15) τα οποία συνδέονται στις αντίστοιχες εισόδους ενεργοποίησης των Ο.Κ. RAM, όπως φαίνεται στο Σχ. 3.

Έστω ότι γίνεται προσπέλαση στη θέση με διεύθυνση **0000** 0000000011 (3η θέση του συστήματος μνήμης). Τότε λόγω του ότι (A_{13},A_{12},A_{11},A_{10})= (0,0,0,0) ενεργοποιείται η έξοδος 0 του αποκωδικοποιητή δηλαδή, γίνεται ενεργό το σήμα EN_0 το οποίο με τη σειρά του ενεργοποιεί την 1η οριζόντια σειρά Ο.Κ. RAM. Επίσης, καθώς και τα 4 Ο.Κ. της 1ης σειράς δέχονται ως διεύθυνση τα ψηφία 0000000011, τότε γίνεται ταυτόχρονη προσπέλαση στην 3η θέση του κάθε Ο.Κ, άρα στην 3η θέση του συνολικού συστήματος μνήμης.

Σημειώστε ότι παρόλο που όλα τα Ο.Κ. δέχονται τις ίδιες τιμές ψηφίων διεύθυνσης, αποκρίνονται μόνο τα της σειράς που οδηγείται από το σήμα EN_0.

ΑΣΚΗΣΗ 4

Θεωρείστε υπολογιστικό σύστημα με δίαυλο δεδομένων 8 ψηφίων και δίαυλο διευθύνσεων 16 ψηφίων και μνήμη με 8 ψηφία ανά θέση. Έστω ότι από τη συνολική μνήμη χρησιμοποιούνται μόνο τρία τμήματα με διευθύνσεις **0000-0FFF (1ο τμήμα)**, **1000-1FFF (2ο τμήμα)** και **6000-7FFF (3ο τμήμα).** Το 1ο τμήμα αντιστοιχεί σε μνήμη ROM, ενώ τα υπόλοιπα σε μνήμη RAM.

Σχεδιάστε την οργάνωση του παραπάνω συστήματος μνήμης. Έχετε στη διάθεση σας τα ακόλουθα Ολοκληρωμένα Κυκλώματα (Ο.Κ.) μνήμης: α) **Ο.Κ. ROM μεγέθους 4KB με 8 ψηφία ανά θέση**, β) **Ο.Κ. RAM1 μεγέθους 4KB με 8 ψηφία ανά θέση**, γ) **Ο.Κ. RAM2 μεγέθους 4KB με 4 ψηφία ανά θέση**, καθώς και κάθε είδους αποκωδικοποιητή και λογικής πύλης.

ΛΥΣΗ

Δεδομένου ότι δεν καλούμαστε να καλύψουμε όλο το χώρο διευθύνσεων που αντιστοιχούν σε 16 ψηφία διεύθυνσης (64K θέσεις) αλλά τμήματα αυτού, η αντιμετώπιση του θέματος είναι μερικώς διαφορετική από αυτή της προηγούμενης άσκησης. Αρχικά, καταστρώνουμε το πίνακα διευθύνσεων που αντιστοιχεί στις συγκεκριμένες περιοχές μνήμης.

ΠΙΝΑΚΑΣ Ι: Διευθύνσεις Τμημάτων Μνήμης

Τμήμα μνήμης	**Χώρος διευθύνσεων**	**Δυαδική αναπαρ. διευθύνσεων**	**Θέσεις Μνήμης**	**Απαιτούμενα Ψηφία για διευθυνσ.**	**Ψηφία Διεύθ.**
1 (ROM)	0000 …. 0FFF	0000 0000 0000 0000 …. 0000 1111 1111 1111	2^{12}	12	A_{11}-A_0
2 (RAM)	1000 …. 1FFF	0001 0000 0000 0000 …. 0001 1111 1111 1111	2^{12}	12	A_{11}-A_0
3 (RAM)	6000 …. 7FFF	0110 0000 0000 0000 …. 0111 1111 1111 1111	2^{13}	13	A_{12}-A_0

Από την δυαδική αναπαράσταση των διευθύνσεων του κάθε τμήματος, όπως παρουσιάζεται στον πίνακα Ι, εξάγουμε το πλήθος των θέσεων μνήμης και τα ψηφία που απαιτούνται για τη διευθυνσιοδότηση του κάθε τμήματος μνήμης που μας απασχολεί. Για παράδειγμα, ο 1ο τμήμα έχει 2^{12} θέσεις μνήμης αφού τα 12 λιγότερο σημαντικά ψηφία (A_{11}-A_0) παίρνουν όλους του δυνατούς συνδυασμούς τιμών από

(0000 0000 0000 έως και 0000 1111 1111 1111). Εφόσον, στο τμήμα αυτό αντιστοιχούν 2^{12} θέσεις μνήμης, απαιτούνται 12 ψηφία διεύθυνσης.

Στη συνέχεια πρέπει να βρούμε μοναδικούς συνδυασμούς ψηφίων διεύθυνσης που ο καθένας τους θα ορίζει μονοσήμαντα το κάθε τμήμα μνήμης. Οι συνδυασμοί αυτοί θα χρησιμοποιηθούν για τη δημιουργία σημάτων ενεργοποίησης για το κάθε τμήμα μνήμης, όπως εξηγείται στη συνέχεια.

Μελετώντας τη δυαδική αναπαράσταση των διευθύνσεων, συμπεραίνουμε ότι τα 12 λιγότερο σημαντικά ψηφία (A_{11}-A_0) δε βοηθούν στο σκοπό αυτό καθώς παίρνουν όλους τους δυνατούς συνδυασμούς τιμών και στα 3 τμήματα. Επομένως, δε μπορούν να χρησιμοποιηθούν για να ορίσουν μονοσήμαντα το κάθε τμήμα μνήμης. Εξετάζοντας όμως τα 4 περισσότερο σημαντικά ψηφία έχουμε

ΠΙΝΑΚΑΣ ΙΙ: Τα 4 Σημαντικά Ψηφία Διευθ. των Τμημάτων Μνήμης

Τμήμα μνήμης	**Ψηφία Διεύθυνσης**				
	A_{15}	**A_{14}**	**A_{13}**	**A_{12}**	**Δεκαδική τιμή**
1 (ROM)	0	0	0	0	0
2 (RAM)	0	0	0	1	1
3 (RAM)	0 … 0	0 … 0	1 … 1	0 … 1	2 … 3

Από τον Πίνακα ΙΙ παρατηρούμε ότι:

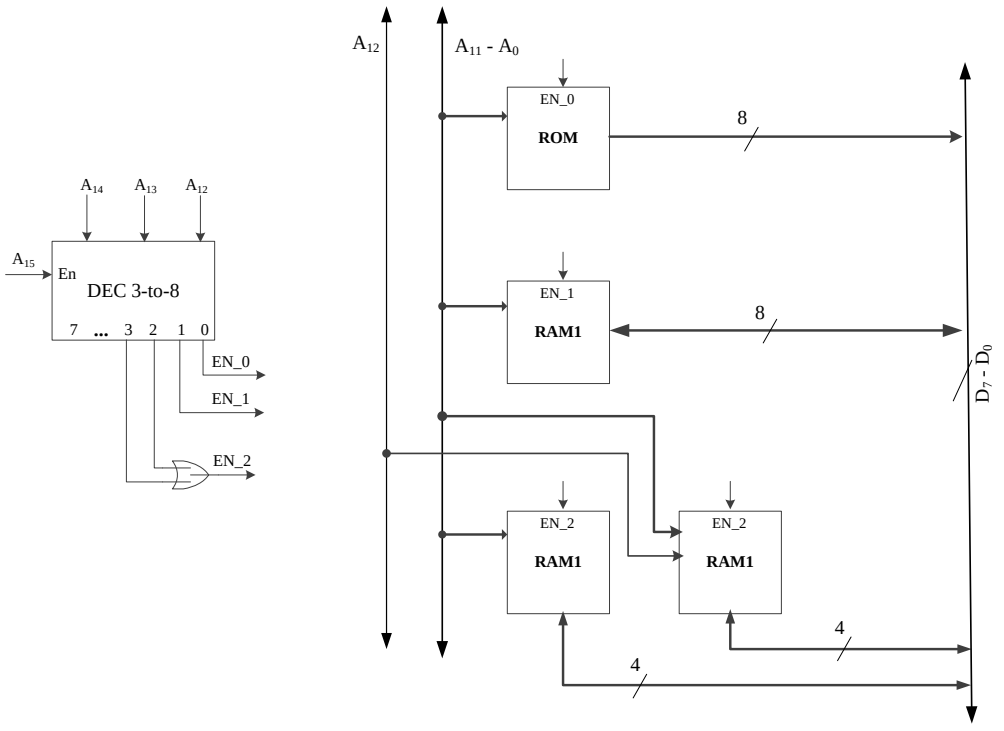
- Το ψηφίο A_{15} έχει τη τιμή 0 και στα 3 τμήματα. Επομένως, θα χρησιμοποιηθεί ως καθολικό σήμα ενεργοποίησης (Global enable) για τα 3 συγκεκριμένα τμήματα μνήμης.
- Με βάση τιμές των A_{14}, A_{13}, A_{12} έχουμε ότι: α) για το 1ο τμήμα ισχύει $(A_{14},A_{13},A_{12})=(0,0,0)=0_{10}$, β) για το 2ο τμήμα ισχύει $(A_{14},A_{13},A_{12})=(0,0,1)=1_{10}$, και γ) για το 3ο τμήμα ισχύει $(A_{14},A_{13},A_{12})=(0,1,0)=2_{10}$ και $(A_{14},A_{13},A_{12})=(0,1,1)=3_{10}$. Επομένως, μπορούμε να χρησιμοποιήσουμε έναν αποκωδικοποιητή 3-σε-8 με εισόδους τα σήματα A_{14}, A_{13}, A_{12} όπου η έξοδος 0 θα ενεργοποιεί το 1ο τμήμα, η έξοδος 1 το 2ο τμήμα και οι έξοδοι 2 ή 3 το 3ο τμήμα μνήμης.
- Εφόσον το σήμα A_{15} είναι το καθολικό σήμα ενεργοποίησης και για τα 3 τμήματα, θα χρησιμοποιηθεί ως σήμα ενεργοποίησης του αποκωδικοποιητή.

Στη συνέχεια χρειάζεται να βρεθεί η υλοποίηση του κάθε τμήματος μνήμης με βάση τα διαθέσιμα Ο.Κ. μνήμης. Με βάση τα εκφώνηση για τα διαθέσιμα Ο.Κ. μνήμης έχουμε

Ο.Κ.	**Μέγεθος (KBytes)**	**Οργάνωση (bits/θέση)**	**Θέσεις μνήμης**	**Bytes/ Θέση**	**Απαιτούμενα Ψηφία διεύθ.**
ROM	4	8 = 1 byte	$4K = 2^{12}$	1	12
RAM1	4	8 = 1 byte	$4K = 2^{12}$	1	12
RAM2	4	4 = 0,5 byte	$8K = 2^{13}$	1/2	13

Επομένως με βάση τις θέσεις μνήμης και τα ψηφία διεύθυνσης των πινάκων I και II και λαμβάνοντας υπόψη (εκφώνηση άσκησης) ότι κάθε θέση έχει 8bits, συμπεραίνουμε ότι: α) για το 1ο τμήμα θα χρησιμοποιηθεί ένα Ο.Κ. τύπου ROM, β) για το 2ο τμήμα θα χρησιμοποιηθεί ένα Ο.Κ. τύπου RAM1, και γ) για το 3ο τμήμα θα χρησιμοποιηθούν δύο Ο.Κ. τύπου RAM2 συνδεδεμένα παράλληλα καθώς το καθένα έχει 4bits/θέση.

Με βάση τα παραπάνω το η οργάνωση του συγκεκριμένου συστήματος μνήμης απεικονίζετε στο επόμενο σχήμα (Σχ.1). Παρατηρήστε ότι το σήμα επιλογής EN_2 είναι η έξοδος μία πύλης OR με εισόδους τις εξόδους 2 και 3 του αποκωδικοποιητή. Αυτό συμβαίνει γιατί το 3ο τμήμα αποκρίνεται όταν $(A_{14},A_{13},A_{12})=(0,1,0)=2_{10}$ ή $(A_{14},A_{13},A_{12})=(0,1,1)=3_{10}$. Επίσης, παρατηρήστε ότι για το 3ο τμήμα χρησιμοποιούνται δύο Ο.Κ. παράλληλα διότι το μήκος λέξης του συστήματος μνήμης είναι 8bits ενώ σε για το Ο.Κ. RAM2 ισχύει 4bits/θέση.



Σχ.1: Οργάνωση συστήματος μνήμης

ΑΣΚΗΣΗ 5

Θεωρείστε ψηφιακό κύκλωμα που δέχεται ως εισόδους δύο αριθμούς, **In_1** και **In_2** (σε αναπαράσταση συμπληρώματος ως προς 2 μήκους 16-bit ο καθένας και ένα σήμα εκκίνησης λειτουργίας (1-bit), **start**, και παράγει μία έξοδο (16-bit), **Out**. Το κύκλωμα χρησιμοποιεί τους καταχωρητές **A**, **B**, και **C** για τα σήματα In_1, In_2 και Out, αντίστοιχα και εκτελεί τις ακόλουθες λειτουργίες

- Όταν start=1 εκτελείται φόρτωση των εισόδων In_1, In_2 στους αντίστοιχους καταχωρητές δηλαδή, A= In_1, B= In_2
- Αν A<0 (δηλ. In_1<0), τότε C = B/2 (δηλ. Out = B/2) αλλιώς C = A*2 (δηλ. Out = A*2).

Για το παραπάνω κύκλωμα:

Α) Σχεδιάστε την αρχιτεκτονική του κυκλώματος σε επίπεδο καταχωρητή (Register Transfer Level – RTL) καθορίζοντας τα σήματα επικοινωνίας μεταξύ των μονάδων ελέγχου (Control Unit) και χειρισμού δεδομένων (Data Path)

Β) Σχεδιάστε το αλγοριθμικό διάγραμμα καταστάσεων (Algorithmic State Machine datapath– ASMD) που αντιστοιχεί στις λειτουργίες του κυκλώματος

Γ) Από το διάγραμμα ASMD εξάγεται το διάγραμμα καταστάσεων (State Transition Graph – STG) που αντιστοιχεί στη μονάδα ελέγχου και βρείτε τις Boolean συναρτήσεις για τα σήματα εξόδου αυτής

Δ) Δώστε την υλοποίηση της μονάδας ελέγχου και της μονάδας χειρισμού δεδομένων

ΛΥΣΗ

Α) Όπως είναι γνωστό κάθε ψηφιακό κύκλωμα αποτελείται στη γενική του μορφή από δύο επιμέρους υπό-κυκλώματα που είναι: α) η **μονάδα ελέγχου** (Control Unit) και β) η **μονάδα χειρισμού δεδομένων** (Data Path).

Η μονάδα ελέγχου είναι ένα ακολουθιακό κύκλωμα που έχει ως είσοδο σήματα από το εξωτερικό περιβάλλον (external primary inputs-commands) και σήματα κατάστασης (status signals) από τη μονάδα χειρισμού δεδομένων και παράγει σήματα ελέγχου (control signals), τα οποία καθορίζουν ποια λειτουργία θα εκτελεστεί από τη μονάδα χειρισμού δεδομένων. Επιπλέον, παράγει εξωτερικά σήματα που δηλώνουν το τέλος της εκτέλεσης των λειτουργιών και τη γενική κατάσταση του όλου κυκλώματος.

Από την άλλη, η μονάδα χειρισμού δεδομένων δέχεται από το περιβάλλον σήματα που αντιστοιχούν στα δεδομένα εισόδου (external primary inputs - data inputs) καθώς και σήματα ελέγχου (control signals) από τη μονάδα ελέγχου και εκτελεί λειτουργίες στα δεδομένα των σημάτων εισόδου. Οι έξοδοι της μονάδας αυτής είναι τα σήματα κατάστασης (status signals) που οδηγούνται και στη μονάδα ελέγχου και τα αποτελέσματα των υπολογισμών (external primary outputs). Η μονάδα αυτή είναι κατά κύριο λόγο ένα συνδυαστικό κύκλωμα που περιλαμβάνει όμως καταχωρητές για την αποθήκευση και συγχρονισμό των εξωτερικών σημάτων εισόδου και εξόδου.

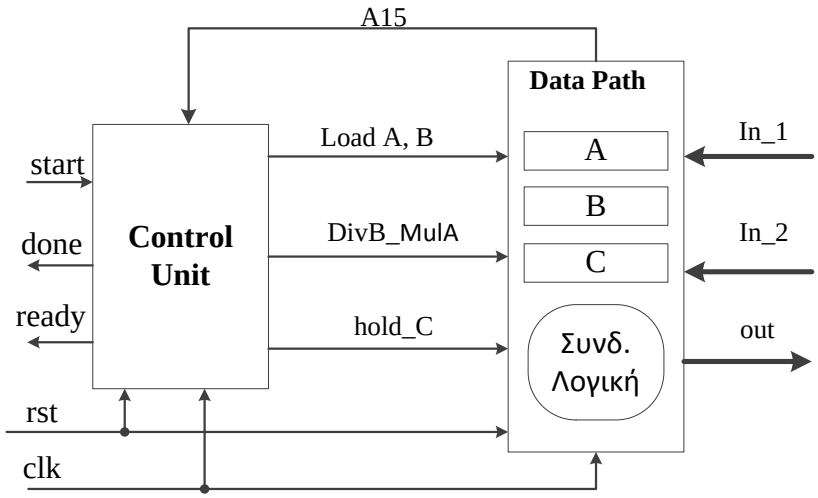
Στην περίπτωση μας, αναφορικά με τα εξωτερικά σήματα εισόδου-εξόδου της μονάδας ελέγχου έχουμε τα ακόλουθα. Η μονάδα ελέγχου δέχεται το εξωτερικό σήμα εισόδου **start** το οποίο όταν είναι ενεργό γίνεται φόρτωση των δεδομένων εισόδου, In_1 και In_2, στους καταχωρητές A και B, αντίστοιχα. Επίσης, παράγει ένα εξωτερικό σήμα, **done**, που δηλώνει το τέλος της εκτέλεσης των πράξεων. Επιπλέον, είναι χρήσιμο να παράγει ένα επιπλέον σήμα εξόδου, **ready**, που δηλώνει ότι είναι το κύκλωμα είναι έτοιμο για να δεχτεί νέα δεδομένα και εκτελέσει νέους υπολογισμούς. Επίσης, καθώς είναι ένα ακολουθιακό κύκλωμα δέχεται ως εισόδους δύο επιπλέον σήματα που είναι τα σήματα ρολογιού, **clk**, και καθαρισμού δεδομένων καταχωρητών, **rst** (reset), το οποίο προτιμάτε να είναι ασύγχρονο.

Επιπλέον, η μονάδα ελέγχου, πρέπει να παράγει σήματα ελέγχου για τη μονάδα χειρισμού δεδομένων. Συγκεκριμένα, πρέπει να παράγει ένα σήμα που να εκτελεί τη φόρτωση των καταχωρητών A και B με τα δεδομένα εισόδου δηλαδή, A= In_1, B**=** In_2. Αφού, οι καταχωρητές A, B φορτώνονται ταυτόχρονα, το σήμα αυτό είναι κοινό σήμα (1-bit) που καλείται, **Load A,B**. Συγκεκριμένα, όταν Load A,B = 1 τότε οι καταχωρητές A, B φορτώνονται με δεδομένα ενώ όταν Load A,B = 0 δε γίνεται φόρτωση δεδομένων στους A, B. Επίσης, πρέπει να εκτελεί τις πράξεις C=B/2 ή C=A*2 ανάλογα με τον αν ο αριθμός A είναι αρνητικός ή όχι. Επομένως, πρέπει να παράγει ένα σήμα ελέγχου (1-bit), που καλείται **DivB_MulA**, με την ακόλουθη λειτουργία: αν DIV_B/MUL_A=1 τότε C=B/2 ενώ αν DIV_B/MUL_A=0 τότε C=A*2. Τέλος, είναι χρήσιμο να παράγει ένα επιπλέον σήμα ελέγχου, **hold_C**, για τη διατήρηση του αποτελέσματος στον καταχωρητή εξόδου C.

Όμως, η εκτέλεση μιας εκ των παραπάνω πράξεων (C = B/2, C = A*2) εξαρτάται από το πρόσημο του αριθμού του καταχωρητή A. Για το λόγο αυτό, η μονάδα ελέγχου δέχεται ένα σήμα κατάστασης από τη μονάδα χειρισμού δεδομένων που δηλώνει το πρόσημο του αριθμού του καταχωρητή A. Δεδομένου ότι οι τα δεδομένα εισόδου είναι σε αναπαράσταση συμπληρώματος ως προς 2, το πρόσημο καθορίζεται από το

περισσότερο σημαντικό ψηφίο του κάθε αριθμού. Επομένως, η μονάδα ελέγχου δέχεται το ως σήμα κατάστασης το ψηφίο A15 το οποίο αν είναι A15=1 τότε ο αριθμός A (A15… A0) είναι αρνητικός αλλιώς ο A είναι θετικός ή μηδέν.

Όσον αφορά το κύκλωμα χειρισμού δεδομένων, έχει ως εισόδους τα εξωτερικά σήματα εισόδου, In_1 και In_2 και τα σήματα ελέγχου, που περιγράφτηκαν παραπάνω ενώ οι έξοδοι είναι το σήμα out προς το εξωτερικό περιβάλλον και το σήμα A15 προς τη μονάδα ελέγχου. Επίσης, η μονάδα αυτή περιλαμβάνει τους καταχωρητές A, B, C, συνδυαστικά κυκλώματα για την εκτέλεση των πράξεων C=B/2 και C=A*2 και συνδυαστική λογική για τη διασύνδεση των υπο-μονάδων της (steering logic – MUXes). Λαμβάνοντας υπόψη τα παραπάνω, η αρχιτεκτονική σε επίπεδο RTL είναι αυτή του Σχ.1 με τη λειτουργία των σημάτων να περιγράφεται στον πίνακα I



Σχ.1: Αρχιτεκτονική του κυκλώματος σε επίπεδο RTL

ΠΙΝΑΚΑΣ I: Λειτουργία σημάτων εισόδου-εισόδου και διεπαφής μονάδων ελέγχου και χειρισμού δεδομένων

Σήμα	Τιμή	Λειτουργία
start	1	Εκκίνηση λειτουργίας κυκλώματος
	0	Το κύκλωμα είναι αδρανές
done	1	Νέο αποτέλεσμα στον καταχωρητή C
	0	Παλιό αποτέλεσμα στον καταχωρητή C
ready	1	Το κύκλωμα μπορεί να εκτελέσει νέους υπολογισμούς
	0	Το κύκλωμα εκτελεί υπολογισμούς
A15	1	A<0 (εκτέλεση πράξης C=B/2)
	0	A≥0 (εκτέλεση πράξης C=A*2)
Load A, B	1	A=In_1, B=In_2
	0	Διατήρηση παλιών τιμών στους A, B
DivB_MulA	1	C=B/2
	0	C=A*2
hold_C	1	Ο καταχωρητής C διατηρεί την τιμή του
	0	Τα δεδομένα του C μπορούν να αλλάξουν

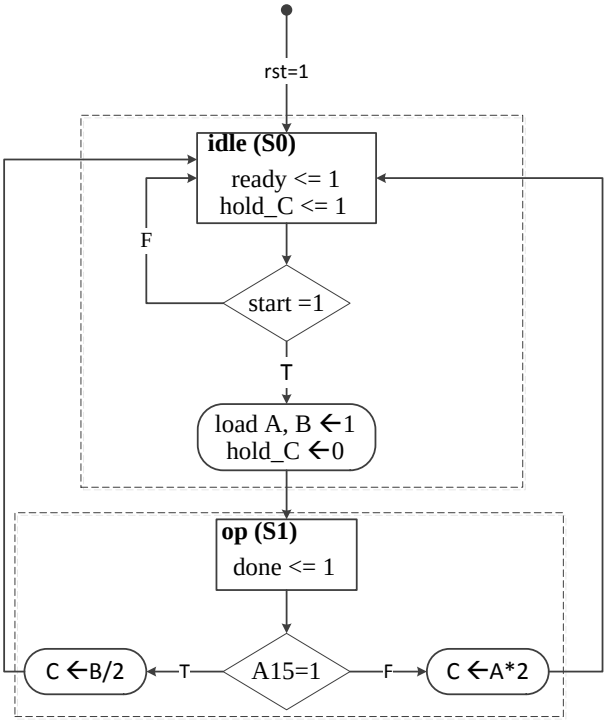
B) Η λειτουργία του κυκλώματος αποτελείται από δύο στάδια. Αρχικά το κύκλωμα έρχεται σε μία αρχική κατάσταση idle (S0) μέσω του της ασύγχρονης ενεργοποίησης του σήματος rst. Όσο βρισκόμαστε στην κατάσταση S0, η μονάδα ελέγχου παράγει τα σήματα ready<=1 και hold_C<=1. Αφού τα σήματα αυτά εξαρτώνται μόνο από την τιμή της κατάστασης είναι τύπου Moore και χρησιμοποιείται ο συμβολισμός *σήμα<=τιμή* και δηλώνονται εντός του κουτιού κατάστασης (state box) στο διάγραμμα ASMD Θυμηθείτε ότι ένα σήμα τύπου Moore μπορεί να εκτελέσει το πολύ μία μετάβαση σε μία περίοδο ρολογιού. Επίσης, καθώς τα σήματα Moore εξαρτώνται από την τιμή της κατάστασης, η οποία αλλάζει μετά την ακμή του ρολογιού, η νέα τιμή είναι διαθέσιμη στον επόμενο κύκλο ρολογιού(η ενημέρωση τους γίνεται με καθυστέρηση ενός κύκλου ρολογιού).

Όσο το κύκλωμα βρίσκεται στην κατάσταση S0, σε κάθε παλμό (ανερχομένη ακμή του ρολογιού) ελέγχεται το σήμα start. Αν start=1, τότε στην ίδια ακμή ρολογιού πρέπει να α) γίνει φόρτωση των δεδομένων εισόδου στους καταχωρητές A, B (load A,B ← 1), β) να απενεργοποιηθεί το σήμα hold_C (hold_C ← 0) και γ) να γίνει μετάβαση στην κατάσταση υπολογισμού (operation-S1). Τα σήματα load A,B και hold_C είναι σήματα ελέγχου που παράγει η μονάδα ελέγχου και δέχεται η μονάδα χειρισμού δεδομένων. Εφόσον τα σήματα load A,B και hold_C αλλάζουν κατάσταση σε συνάρτηση με την παρούσα κατάσταση (S0) και την τιμή εξωτερικών σημάτων (start), τότε αυτά είναι τύπου Mealy για τα οποία χρησιμοποιείται ο συμβολισμός *σήμα←τιμή* και δηλώνονται στα κουτιά εκτέλεσης υπό συνθήκη (decision boxes). Θυμηθείτε ότι ένα σήμα τύπου Mealy μπορεί να εκτελέσει περισσότερες από μία μεταβάσεις σε μία περίοδο ρολογιού. Συγκεκριμένα, λόγω της εξάρτησης τους και από σήματα εισόδου (start) διαφορετικά των σημάτων κατάστασης, τα οποία μπορεί να αλλάξουν οποιαδήποτε στιγμή, ένα σήμα τύπου Mealy μπορεί να εκτελέσει πολλαπλές μεταβάσεις σχεδόν ακαριαία (με καθυστέρηση της αντίστοιχης λογικής) μετά την αλλαγή των αντίστοιχων σημάτων εισόδου (start).

Όταν το κύκλωμα βρίσκεται στην κατάσταση S1, τότε στη διάρκεια της αντίστοιχη περιόδου ρολογιού πρέπει να εκτελέσει του υπολογισμούς (C=B/2, C=A*2) και να ενεργοποιήσει το σήμα done (done=1). Γι το σκοπό αυτό το σήμα DivB_mulA πρέπει να είναι ενεργό όταν έρθει η αντίστοιχη ακμή του ρολογιού. Αφού αυτό εξαρτάται από την τιμή ενός σήματος (A15) διαφορετικό από τα σήματα κατάστασης και πρέπει να είναι ενεργό στην ακμή του ρολογιού (για να μη χαθεί κύκλος), τότε το σήμα DivB_mulA είναι τύπου Mealy και δηλώνεται σε decision box. Αντίθετα, το σήμα done

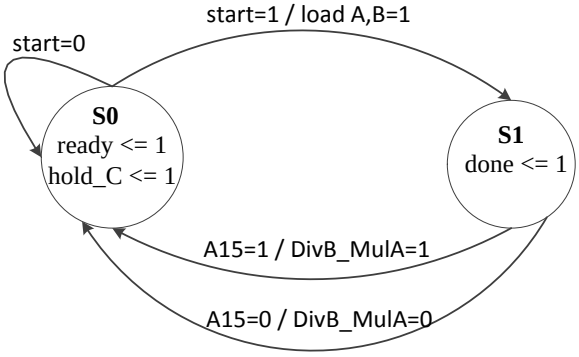
πρέπει να γίνει ενεργό στον επόμενο παλμό αφού το κύκλωμα έχει φτάσει στην S1, επομένως είναι τύπου Moore.

Έτσι, με βάση τα παραπάνω, το διάγραμμα ASMD είναι αυτό του Σχ.2, το οποίο περιλαμβάνει δύο τμήματα (ASM blocks) που δηλώνονται με διακεκομμένες γραμμές. Θυμηθείτε ότι οι λειτουργίες ενός κουτιού υπό συνθήκη εκτελούνται κατά τη μετάβαση από μία κατάσταση σε μία άλλη. Για παράδειγμα, η λειτουργία load A, B γίνεται στην ακμή του ρολογιού στην οποία γίνεται στην ακμή του ρολογιού που γίνεται η μετάβαση S0→S1. Συνεπώς, πρέπει να είναι load A, B=1 όταν έρθει η συγκεκριμένη ακμή.



Σχ.2: Διάγραμμα ASMD

Γ) Δεδομένου του διαγράμματος ASMD, η εξαγωγή του διαγράμματος καταστάσεων είναι άμεση όπως φαίνεται στο Σχ. 3.



Σχ.2: Διάγραμμα ASM

Στο διάγραμμα καταστάσεων, η κάθε κατάσταση δηλώνεται με κύκλο και οι μεταβάσεις με βέλη. Επίσης, τα σήματα τύπου Moore δηλώνονται εντός της κατάστασης. Τέλος, στα βέλη μεταβάσεων χρησιμοποιείται ο συμβολισμός *συνθήκη μετάβασης/Mealy σήμα*.

Εφόσον έχουμε δύο καταστάσεις λειτουργίας S0 και S1 χρειαζόμαστε ένα FF (Flip Flop) με είσοδο nxt_state (επόμενη κατάσταση–next_state) και έξοδο pr_state (παρούσα κατάσταση–present state). Επίσης, έστω ότι κωδικοποιούμε τις καταστάσεις ως S0=0 και S1=1. Για την εξαγωγή των λογικών συναρτήσεων των σημάτων εξόδου της μονάδας ελέγχου πρέπει να λάβουμε υπόψη το διάγραμμα καταστάσεων και τα σήματα εισόδου αυτής. Έτσι, έχουμε τον ακόλουθο πίνακα όπου X σημαίνει αδιάφορη τιμή.

Παρούσα Κατάσταση (pr_state)	**Είσοδοι**		**Επόμενη Κατάσταση (nxt_state)**	**Έξοδοι**				
	start	**A15**		**Load A, B**	**DivB_ MulA**	**hold_C**	**ready**	**done**
S0 (0)	0	X	S0 (0)	0	0	1	1	0
S0 (0)	1	X	S1(1)	1	0	1	1	0
S1 (1)	X	0	S0 (0)	0	0	0	0	1
S1 (1)	X	1	S0 (0)	0	1	0	0	1

Από τον παραπάνω πίνακα η εξαγωγή των λογικών συναρτήσεων των σημάτων εξόδου της μονάδας ελέγχου είναι άμεση. Έτσι, έχουμε:

nxt_state = **Load A,B** = S0 AND start = $\overline{pr\_state}$ AND start

DivB_MulA = S1 AND A15 = pr_state AND A15

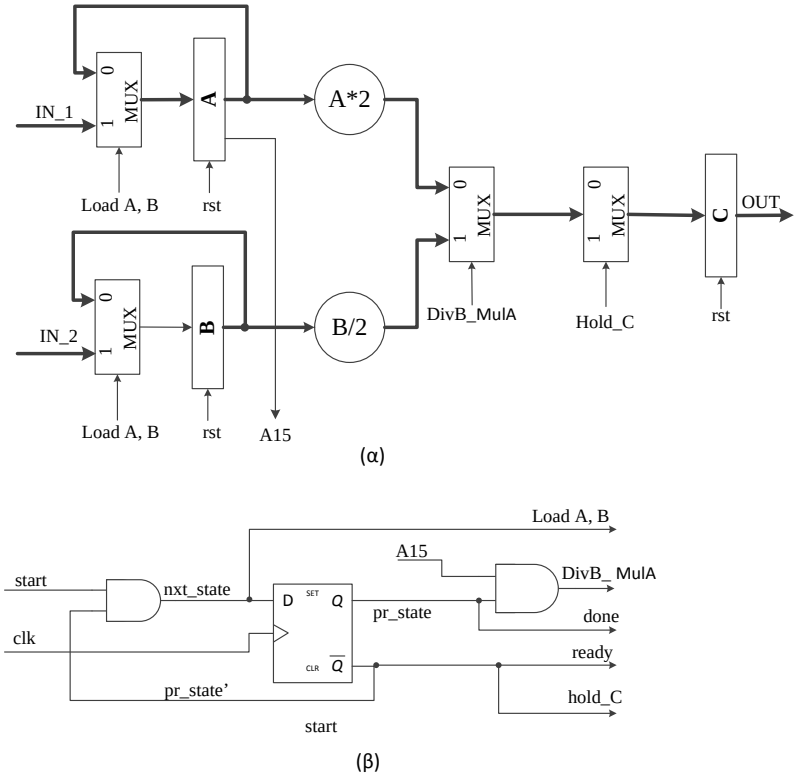
hold_C = **ready** = S0 = $\overline{pr\_state}$

done = **Load A,B** = S1 = pr_state

Δ) Έχοντας βρει τις λογικές εξισώσεις των σημάτων εξόδου της μονάδας ελέγχου, το επόμενο βήμα είναι η κυκλωματική σχεδίαση των μονάδων ελέγχου και χειρισμού δεδομένων. Για τη μονάδα ελέγχου, η κυκλωματική υλοποίηση είναι άμεση με βάση τις παραπάνω λογικές συναρτήσεις.

Για τη μονάδα χειρισμού δεδομένων η σχεδίαση είναι κάπως πιο πολύπλοκη. Αφού πρέπει να εκτελεί τις πράξεις C=B/2 και C=A*2 θα πρέπει να περιέχει κυκλώματα πολ/σμού και διαίρεσης, τα οποία είναι μεγάλα σε επιφάνεια κυκλώματα με μεγάλη καθυστέρηση. Όμως, στη συγκεκριμένη περίπτωση, ο πολλαπλασιασμός (C=A*2) και η διαίρεση (C=B/2) που εκτελούνται είναι σε δυνάμεις του 2. Επομένως, τα αντίστοιχα κυκλώματα είναι κυκλώματα ολίσθησης μίας θέσης αριστερά για τον πολλαπλασιασμό

και μίας θέσης δεξιά για τη διαίρεση. Όμως, δεδομένου ότι οι αριθμοί είναι προσημασμένοι, πρέπει να διατηρείται το πρόσημο μετά την εκτέλεση της πράξης**. Έτσι για τον πολλαπλασιασμό C=A*2 έχουμε C[C15, C14, …, C1, C0] <= [A14, A13, …, A1, 0] και για τη διαίρεση C=B/2 έχουμε C[C15, C14, …, C0] <= [B15, B15, B14, …, B1].** Τέλος, επειδή τα δεδομένα του καταχωρητή C είναι άλλοτε C=B/2 και άλλοτε C=A*2 απαιτείται οι χρήση πολυπλέκτη ώστε η έξοδος του C να παίρνει μία από τις δύο τιμές. Επίσης, καθώς οι καταχωρητές A, B, C είτε διατηρούν τις τιμές τους είτε παίρνουν καινούργιες κατά τη διάρκεια των υπολογισμών, απαιτούνται επίσης πολυπλέκτες. Έτσι, με βάση τα παραπάνω, η κυκλωματική υλοποίηση της μονάδας δείχνονται στο Σχ.3, όπου η έντονη διαγράμμιση αντιστοιχεί σε δίαυλο 16-bit.



Σχ.3: Κυκλωματικές υλοποιήσεις: (α) μονάδα χειρισμού δεδομένων, (β) μονάδα ελέγχου

Είναι σημαντικό να τονιστούν δύο παρατηρήσεις. Πρώτον, οι πολυπλέκτες στις εισόδους των καταχωρητών θα μπορούσαν να έχουν αποφευχθεί αν α FFs των καταχωρητών είχαν σήμα επίτρεψης εγγραφής, en (enable). Δεύτερον, η διαδικασία που παρουσιάστηκε ήταν άμεση και σειριακή (σχεδιασμός αρχιτεκτονικής→ASMD→STG→κυκλωματική υλοποίηση). Στην πράξη όμως αυτό δεν ισχύει αλλά υπάρχει αλληλεπίδραση μεταξύ των σταδίων. Για παράδειγμα, το στάδιο της κυκλωματικής υλοποίησης μπορεί να επιβάλλει επιπλέον σήματα ελέγχου, όποτε πρέπει να αλλάξουν κατάλληλα όλα τα προηγούμενα στάδια.

Σημείωμα Αναφοράς

«Ψηφιακή Λογική Σχεδίαση»
Έκδοση: 1.0 Πάτρα 2015
Διαθέσιμο στη διαδικτυακή διεύθυνση: https://eclass.upatras.gr/courses/EE890/

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Πατρών**» έχει χρηματοδοτήσει μόνο τη 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.