ΗΥ-120: Ψηφιακή Σχεδίαση
Φθινόπωρο 2021

Τμ. Επ. Υπολογιστών
© Πανεπιστήμιο Κρήτης

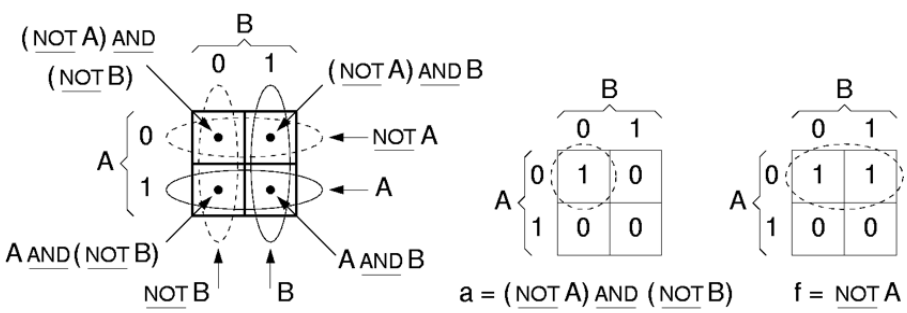
Εργαστήριο 4:
Χάρτες Karnaugh, Άλγεβρα Boole, Μνήμη RAM

1 - 13 Νοεμβρίου 2021

[Βιβλία: *προαιρετικά* μπορείτε να διαβάσετε: Dally: § **6.1 - 6.8** (σελ. 114-130), § **3.1 - 3.5** (σελ. 47-55)· Mano (5η έκδ.): § **3.1 - 3.3** (σελ. 73-84), § **3.5** (σελ. 88-90), § **3.8** (σελ. 103-108), § **2.3 - 2.5** (σελ. 40-50)· Wakerly (3η έκδ.): § **4.1** έως και **4.3** (σελ. 231 - 279)].

4.1 Διαγράμματα Venn Δύο Μεταβλητών, Χάρτες Karnaugh

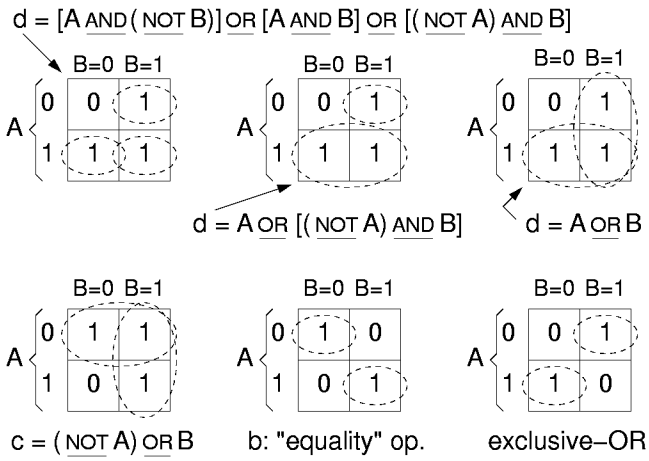
Οι λογικές συναρτήσεις 2 εισόδων καθορίζονται πλήρως από τις 4 τιμές που αυτές παίρνουν στον καθένα από τους 4 ($=2^2$) συνδυασμούς των 2 εισόδων τους (γι' αυτό και υπάρχουν ακριβώς 16 ($=2^4$) διαφορετικές τέτοιες λογικές συναρτήσεις 2 εισόδων). Άρα, ο πίνακας αληθείας τους περιέχει 4 δυαδικές τιμές. Γιά εύκολη αναγνώριση "με το μάτι" των λογικών αυτών συναρτήσεων μας βολεύει να διατάξουμε τις 4 αυτές τιμές σ' ένα διδιάστατο πίνακα 2x2, όπου οι γραμμές αντιστοιχούν στην τιμή της μιάς μεταβλητής εισόδου και οι στήλες αντιστοιχούν στην άλλη μεταβλητή εισόδου, όπως φαίνεται στο σχήμα. Η παράσταση αυτή, ανάλογα πώς την βλέπει και πώς την χρησιμοποιεί κανείς, λέγεται είτε **Χάρτης Karnaugh** (Καρνώ) είτε **Διάγραμμα Venn**.



Το διάγραμμα Venn έχει το εξής νόημα: θεωρούμε ότι η μεταβλητή εισόδου A προσδιορίζει κατά πόσον είμαστε *μέσα* στην "περιοχή A" (κάτω γραμμή) ή έξω από αυτήν (1 = μέσα, 0 = έξω). Ομοίως, η είσοδος B μας λέει αν είμαστε μέσα (1) στην περιοχή B (δεξιά στήλη) ή έξω (0) από αυτήν. Όταν μας δίνουν μιά λογική συνάρτηση (δηλαδή άσσους και μηδενικά στα 4 τετραγωνάκια), εμείς κοιτάζουμε σε ποιά περιοχή η συναρτηση αυτή γίνεται αληθής (τιμή = 1), και περιγράφουμε αυτή την περιοχή σαν συνάρτηση των τιμών των εισόδων A και B. Αν η συνάρτηση γίνεται αληθής μόνο στο κάτω δεξιό τετράγωνο, τότε γίνεται αληθής όταν είμαστε "μέσα" στο A (A=1) **και** "μέσα" στο B (B=1), δηλαδή όταν είναι αληθές το A **και** αληθές το B, δηλαδή όταν "A **ΚΑΙ** B" ("A **AND** B"). Κατ' αναλογία, όταν μιά λογική συνάρτηση 2 μεταβλητών γίνεται αληθής μόνο στο κάτω αριστερό τετράγωνο, τότε αυτή γίνεται αληθής όταν είμαστε μέσα στο A (A=1) **και** "έξω" από το B (B=0), δηλαδή όταν είναι αληθές το A **και** ψευδές (**όχι** αληθές) το B, δηλαδή όταν "A **ΚΑΙ** (**ΟΧΙ** B)" ("A **AND** (**NOT** B)"). Ομοίως, η λογική συνάρτηση που γίνεται αληθής μόνο στο πάνω δεξιό τετράγωνο είναι η "(**NOT** A) **AND** B", ενώ η συνάρτηση που γίνεται αληθής μόνο στο πάνω αριστερό τετράγωνο είναι η "(**NOT** A) **AND** (**NOT** B)". Στην περίπτωση του παραδείγματος της παραγράφου 3.11 με την ένδειξη 7 τμημάτων, η έξοδος a τυχαίνει να είναι ακριβώς αυτή η τελευταία συνάρτηση, όπως διαπιστώνουμε εύκολα αν σχεδιάσουμε τον πίνακα αληθείας της στη μορφή που φαίνεται στο σχήμα.

Όταν μιά λογική συνάρτηση γίνεται αληθής (τιμή = 1) σε δύο διπλανά τετράγωνα, τότε αυτή μπορεί να περιγραφεί σαν "μέσα" ή "έξω" από την περιοχή της μιάς από τις δύο μεταβλητές εισόδου. Έτσι, στο παραπάνω παράδειγμα, η έξοδος f, που γίνεται 1 στα δύο τετράγωνα της επάνω γραμμής, μπορεί να περιγραφεί σαν "έξω" από το A, δηλαδή **ΟΧΙ** A, όπως φαίνεται στο παραπάνω σχήμα. Αντίστοιχα, μιά συνάρτηση που θα γίνονταν 1 στην κάτω γραμμή θα ήταν ίση με A, αν γίνονταν 1 στη δεξιά στήλη θα ήταν ίση με B, και αν γίνονταν 1 στην αριστερή στήλη θα ήταν ίση με **NOT** B.

Μιά λογική συνάρτηση που γίνεται αληθής σε τρία τετράγωνα μπορεί να περιγραφεί όπως στο επόμενο σχήμα: επάνω φαίνονται τρείς εναλλακτικές περιγραφές γιά τη συνάρτηση που οδηγεί το τμήμα d της ένδειξης 7 τμημάτων (§3.11). Στην πρώτη περιγραφή, η περιοχή αληθείας της d ορίζεται σαν η ένωση τριών περιοχών μεγέθους ενός τετραγώνου η καθεμία. Καθώς παραπάνω η τομή περιοχών αντιστοιχούσε στο λογικό **και**, η ένωση περιοχών, εδώ, αντιστοιχεί στο λογικό **ή**, αφού η συνάρτηση είναι αληθής όποτε είναι αληθής ή ο ένας όρος, ή ο δεύτερος, ή ο τρίτος (ή περισσότεροι ταυτόχρονα). Έτσι, οι τρείς περιοχές του πρώτου σχήματος μας δίνουν την περιγραφή της συνάρτησης που φαίνεται από πάνω, η οποία είναι το λογικό **ή** τριών όρων που ο καθένας τους είναι ένα λογικό **και**.



Η δεύτερη περιγραφή της λογικής συνάρτησης d οδηγεί σε απλούστερη έκφραση, διότι χρησιμοποιεί λιγότερες και μεγαλύτερες βασικές περιοχές. Όσο μεγαλύτερη είναι μιά βασική περιοχή **γειτονικών τετραγώνων**, τόσο λιγότερους όρους "**και**" έχει η αντίστοιχη λογική έκφραση· και όσο λιγότερες περιοχές χρειάζεται να ενώσουμε γιά να καλύψουμε την περιοχή αληθείας της συνάρτησής μας, τόσο λιγότερα κομμάτια θα ενώνουν οι πράξεις **ή**. Τελικά, η οικονομικότερη περιγραφή της λογικής συνάρτησης d είναι η τρίτη, δεξιά, διότι χρησιμοποιεί τις μεγαλύτερες δυνατές βασικές περιοχές, δηλαδή τους απλούστερους όρους· η εν μέρει επικάλυψη των περιοχών δεν έχει καμία σημασία, αφού η λογική πράξη **ή** είναι αληθής όταν είναι αληθείς είτε μία είτε περισσότερες από τις εισόδους της. Κατ' ανάλογο τρόπο, η εξοδος c που ζητάμε γιά το κύκλωμα του παραδείγματός μας είναι η λογική συνάρτηση "(**NOT** A) **OR** B", όπως φαίνεται στο παραπάνω σχήμα, κάτω αριστερά. Η μέθοδος αυτή της απλοποίησης λογικών συναρτήσεων μέσω συνένωσης γειτονικών τετραγώνων λέγεται "μέθοδος του **Χάρτη Καρνώ**" (Karnaugh Map).

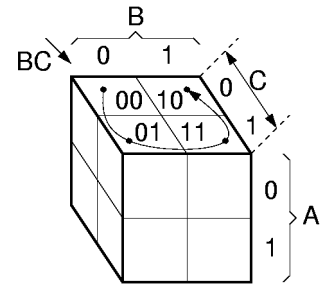
Η έξοδος b του κυκλώματος της §3.11 πρέπει να ανάβει στα δύο τετράγωνα που φαίνονται στο παραπάνω σχήμα (κάτω μέση), τα οποία όμως, δυστυχώς, **δεν** είναι γειτονικά. Γιά το λόγο αυτό, τα δύο αυτά τετράγωνα δεν μπορούν να συνενωθούν όπως παραπάνω, και δεν μπορεί να γίνει καμιά ιδιαίτερη απλοποίηση της σχετικής συνάρτησης --αυτή παραμένει αναγκαστικά η ένωση δύο ανεξάρτητων τετραγώνων: b = [(**NOT** A) **AND** (**NOT** B)] **OR** [A **AND** B]. Πρόκειται γιά τη συνάρτηση ισότητας που είδαμε στην §1.4. Η άλλη λογική συνάρτηση 2 μεταβλητών που δεν απλοποιείται είναι η συνάρτηση αποκλειστικού Ή (exclusive OR, ή "XOR"), που επίσης είδαμε στην ίδια παράγραφο, και που φαίνεται στο παραπάνω σχήμα, κάτω δεξιά.

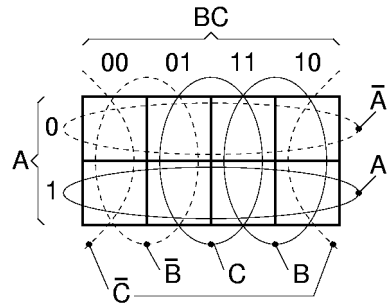
4.2 Διαγράμματα Venn & Χάρτες Karnaugh Τριών - Τεσσάρων Μεταβλητών

Η επιτυχία των διαγραμμάτων Venn δύο μεταβλητών στο να μας βοηθούν να απλοποιούμε λογικές συναρτήσεις οφείλεται στη διάταξη των τιμών της συνάρτησης στις δύο διαστάσεις με τρόπο τέτοιο που η κάθε μεταβλητή να αντιστοιχεί στη μία διάσταση. Γιά να πετύχουμε το ίδιο αποτέλεσμα με τρείς μεταβλητές εισόδου, φανταζόμαστε ότι διατάσουμε τα περιεχόμενα του πίνακα αληθείας στις 3 διαστάσεις, σ' ένα σχήμα κύβου, όπως στο εδώ σχήμα. Στη συνέχεια, επειδή είναι άβολο να σχεδιάζουμε τριδιάστατα σχήματα στο χαρτί, φανταζόμαστε ότι **ξεδιπλώνουμε** τον κύβο, κόβοντάς τον πίσω στη μέση, και φέρνοντας τις δύο πίσω στήλες στην αριστερή και στη δεξιά άκρη αντίστοιχα, όπως φαίνεται στο σχήμα, δεξιά.

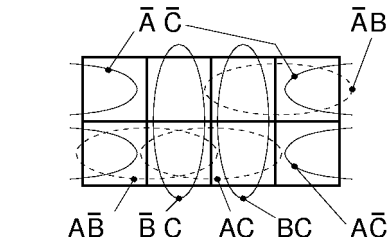
Η συνέπεια είναι ότι οι στήλες του πίνακα αντιστοιχούν στις δύο μεταβλητές εισόδου B και C με τη σειρά που φαίνεται στο σχήμα: 00, 01, 11, 10 --η σειρά αυτή (που λέγεται και "κώδικας Gray") διαφέρει από τη συνηθισμένη σειρά αρίθμησης (μέτρησης) στο δυαδικό (00, 01, 10, 11). Με τη σειρά αυτή πετυχαίνουμε η περιοχή αληθείας της μεταβλητής B να

αποτελείται από 4 **γειτονικά** τετράγωνα --τα 4 δεξιά τετράγωνα-- και ταυτόχρονα η περιοχή αληθείας της μεταβλητής C να αποτελείται επίσης από 4 γειτονικά τετράγωνα --τα 4 μεσαία τετράγωνα. Η περιοχή αληθείας του C' (δηλ. του "NOT C") είναι επίσης 4 "γειτονικά" τετράγωνα --δύο στη άκρη αριστερά και δύο στην άκρη δεξιά-- αλλά γιά να καταλάβουμε ότι αυτά είναι γειτονικά πρέπει να φανταστούμε τον πίνακα σαν ένα ξετυλιγμένο βαρέλι: τα τετράγωνα αυτά ήταν γειτονικά πριν το ξετύλιγμα. Γενικά, στο χάρτη 3 μεταβλητών, οιαδήποτε 4 "γειτονικά" τετράγωνα --δηλαδή 4 τετράγωνα σε σχήμα 2x2 ή 4x1-- αντιστοιχούν σε μία μεταβλητή εισόδου ή στην άρνησή της. Οι δύο οριζόντιες τετράδες αντιστοιχούν στο A και στο A' (NOT A).

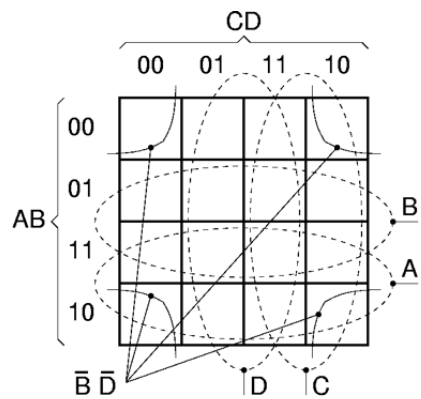




Περιοχές δύο γειτονικών τετραγώνων αντιστοιχούν στο λογικό ΚΑΙ δύο εκ των τριών μεταβλητών εισόδου (ή των αρνήσεών τους). Μερικά τέτοια ζευγάρια φαίνονται στο κάτω μέρος του σχήματος. Παρατηρήστε ότι τα ζευγάρια που περιλαμβάνουν το C' μοιάζουν "κομένα" --ένα τετράγωνο στην άκρη αριστερά και ένα στην άκρη δεξιά-- όμως αποτελούνται και αυτά από τετράγωνα που ήταν γειτονικά πριν ξετυλίξουμε το χάρτη από τη μορφή βαρελιού που λέγαμε πιό πάνω. Τέλος, φυσικά, όπως και στους χάρτες 2 μεταβλητών, μεμονωμένα τετράγωνα αντιστοιχούν στο λογικό ΚΑΙ όλων των μεταβλητών εισόδου ή των αρνήσεών τους, και περιοχές που προκύπτουν από την ένωση ομάδων τετραγώνων αντιστοιχούν στο λογικό Ή των σχετικών όρων. Όταν καλύπτουμε περιοχές με τέτοιες ενώσεις, επιδιώκουμε η κάθε ομάδα γειτονικών τετραγώνων να είναι όσο μεγαλύτερη γίνεται (πάντα βέβαια μεγέθους δύναμης του 2, διατεταγμένη σε σχήμα ορθογωνίου)· επικαλύψεις περιοχών δεν μας ενοχλούν --αντίθετα βοηθούν στη μεγιστοποίηση της έκτασης της κάθε μεμονωμένης περιοχής.

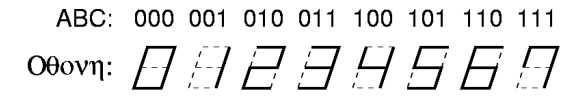


Ο χάρτης Karnaugh τεσσάρων μεταβλητών σχεδιάζεται όπως φαίνεται στο επόμενο σχήμα. Πρέπει να φανταστούμε ότι αυτός ο πίνακας 4x4 προέρχεται από **διπλό ξετύλιγμα** μιάς παράξενης σφαιροειδούς επιφάνειας και οριζόντια και κατακόρυφα. Η αριστερή και η δεξιά στήλη ήταν γειτονικές πριν το ξετύλιγμα, σαν να προέρχονται από ένα όρθιο βαρέλι, και αντιστοιχούν στη μεταβλητή D' (NOT D). Ταυτόχρονα, η επάνω και η κάτω γραμμή ήταν κι αυτές γειτονικές, πριν το ξετύλιγμα από ένα πλαγιαστό βαρέλι, και αντιστοιχούν στη μεταβλητή B' (NOT B). Τετράδες γειτονικών τετραγώνων αντιστοιχούν στο λογικό ΚΑΙ δύο εκ των τεσσάρων μεταβλητών εισόδου, ή των αρνήσεών τους. Μία τέτοια τετράδα --η πιό πολύ κομμένη απ' όλες-- φαίνεται στο σχήμα στις 4 γωνίες: πρόκειται γιά την περιοχή B'D', και τα τετράγωνά της ήταν όλα γειτονικά πριν το διπλό ξετύλιγμα. Άλλες κομμένες τετράδες έχουν 2 τετράγωνα αριστερά και 2 δεξιά, ή 2 τετράγωνα επάνω και 2 κάτω. Ζευγάρια γειτονικών τετραγώνων αντιστοιχούν στο λογικό ΚΑΙ τριών εκ των τεσσάρων μεταβλητών εισόδου, ή των αρνήσεών τους. Μεμονωμένα τετράγωνα αντιστοιχούν στο λογικό ΚΑΙ όλων των μεταβλητών εισόδου (ή των αρνήσεών τους).



Μπορούν να οριστούν και χάρτες Karnaugh πέντε ή περισσοτέρων μεταβλητών, αλλά δεν είναι πρακτικοί. Εξ' άλλου, ας μην ξεχνάμε ότι η απλοποίηση λογικών συναρτήσεων "με το μάτι" και "με το χέρι" ανήκει στο παρελθόν: σήμερα υπάρχουν αποδοτικοί αλγόριθμοι και αντίστοιχα προγράμματα που κάνουν αυτές τις απλοποιήσεις και πολλές άλλες αυτόματα. Το πιό γνωστό τέτοιο πακέτο αυτόματης σύνθεσης υλικού, σήμερα, είναι το "Synopsys".

Άσκηση 4.3: Οι Αριθμοί 0-7 στην Οθόνη 7 Τμημάτων



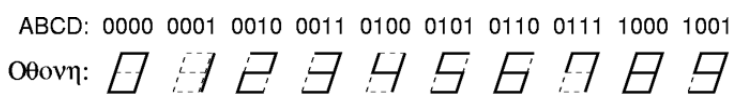
Πρίν φτάσετε στο εργαστήριο κάντε αυτή την άσκηση με μολύβι και χαρτί· η άσκηση αυτή δεν έχει τίποτα να φτιάξτε στο εργαστήριο, διότι τα κυκλώματα που προκύπτουν είναι πολύ μεγάλα γιά να υλοποιηθούν στο χρόνο ενός εργαστηρίου μας --απλώς θα παραδώσετε τη λύση σας μέσα στην αναφορά του εργαστηρίου σας. Ζητείται να βρεθούν οι 7 λογικές συναρτήσεις οδήγησης των 7 τμημάτων του γνωστού μας ενδείκτη, προκειμένου αυτός να εμφανίζει τις 8 ενδείξεις που φαίνονται στο σχήμα, ανάλογα με τον εκάστοτε συνδυασμό τιμών των τριών bits εισόδου A, B, και C. Φτιάξτε τους 7 πίνακες αληθείας σε μορφή 7 χαρτών Karnaugh τριών μεταβλητών, βρείτε τις ομάδες γειτονικών τετραγώνων που χρειάζονται γιά να καλύψουν τις περιοχές ανάματος της κάθε LED, γράψτε τις αντίστοιχες λογικές συναρτήσεις γιά τις 7 εξόδους, και σχεδιάστε το αντίστοιχο κύκλωμα με πύλες.

4.4 Συνθήκες Αδιαφορίας (Don't Care Conditions)

Σε όλες τις παραπάνω περιπτώσεις, οι πίνακες αληθείας των επιθυμητών εξόδων ήταν πλήρως προδιαγεγραμμένοι, δηλαδή μας ενδιέφερε η κάθε έξοδος να έχει μιά συγκεκριμένη, προκαθορισμένη τιμή γιά τον κάθε δυνατό συνδυασμό τιμών των εισόδων. Υπάρχουν όμως και περιπτώσεις "μερικώς προδιαγεγραμμένων" συστημάτων. Γιά παράδειγμα, στην παρακάτω άσκηση, μας ζητείται να οδηγήσουμε τη γνωστή μας οθόνη 7 τμημάτων σε τρόπον ώστε να εμφανίζονται σε αυτήν τα δέκα ψηφία από το 0 ως το 9. Γιά να πετύχουμε δέκα διαφορετικές εξόδους δεν αρκούν προφανώς 3 bits εισόδου --χρειάζονται τέσσερα. Όμως, τα 4 bits έχουν 16 δυνατούς συνδυασμούς· διαλέγουμε δέκα από αυτούς γιά να γεννάνε στην οθόνη τα δέκα επιθυμητά ψηφία, ενώ **δεν μας ενδιαφέρει** τι θα κάνει η οθόνη όταν στις εισόδους εμφανίζεται ένας από τους υπόλοιπους έξι συνδυασμούς --π.χ. το κύκλωμα που μας τροφοδοτεί με τα 4 bits εισόδου ποτέ δεν θα μας δίνει έναν από τους υπόλοιπους 6 συνδυασμούς, ή αν μας δώσει, δεν μας ενδιαφέρει τι θα δείξει η οθόνη σε αυτή την περίπτωση.

Όταν ένας πίνακας αληθείας δεν προκαθορίζει την τιμή εξόδου γιά ορισμένο συνδυασμό τιμών εισόδου, λέμε ότι εκεί έχουμε μία **συνθήκη αδιαφορίας** (don't care condition), και συχνά βάζουμε στον στη θέση εκείνη του πίνακα και του χάρτη Karnaugh σα σύμβολο ένα "x". Στο χάρτη Karnaugh, όταν αναζητούμε την ελάχιστη δυνατή ένωση των μέγιστων δυνατών περιοχών γειτονικών τετραγώνων προκειμένου να καλύψουμε τους άσσους του χάρτη, θεωρούμε ότι το κάθε "x" είναι **ό,τι μας βολεύει**. Αν βολεύει να το θεωρήσουμε σαν άσσο, προκειμένου να πετύχουμε μεγαλύτερη περιοχή γειτονικών τετραγώνων, το θεωρούμε σαν άσσο. Αν βολεύει να το θεωρήσουμε σαν μηδενικό, προκειμένου να μην χρειαστούμε μιά επιπλέον περιοχή γιά να το καλύψουμε, το θεωρούμε σαν μηδενικό. Η συμπεριφορά του τελικού κυκλώματος γιά κάθε αδιάφορη τιμή εισόδων θα καθοριστεί προφανώς από το τι μας βόλεψε και θεωρήσαμε το αντίστοιχο "x" στο χάρτη.

Άσκηση 4.5: Οι Αριθμοί 0-9 στην Οθόνη 7 Τμημάτων



Πρίν το εργαστήριο κάντε αυτή την άσκηση με μολύβι και χαρτί· εδώ, δεν έχετε τίποτα να φτιάξτε στο εργαστήριο, διότι τα κυκλώματα που προκύπτουν είναι πολύ μεγάλα γιά να υλοποιηθούν στο χρόνο ενός εργαστηρίου μας --απλώς θα παραδώσετε τη λύση σας στην αναφορά του εργαστηρίου σας. Ζητείται να βρεθούν οι 7 λογικές συναρτήσεις οδήγησης των 7 τμημάτων του γνωστού μας ενδείκτη, προκειμένου αυτός να εμφανίζει τις 10 ενδείξεις που φαίνονται στο σχήμα, ανάλογα με τον εκάστοτε συνδυασμό τιμών των 4 bits εισόδου A, B, C, και D· δεν μας ενδιαφέρει τι σχήμα θα εμφανίζεται στον ενδείκτη όταν οι 4 είσοδοι βρίσκονται σε έναν από τους υπόλοιπους 6 συνδυασμούς τους. Φτιάξτε τους 7 πίνακες αληθείας σε μορφή 7 χαρτών Karnaugh τεσσάρων μεταβλητών που περιέχουν και από 6 όρους αδιαφορίας καθένας· βρείτε και σημειώστε τις ομάδες γειτονικών τετραγώνων που βολεύουν και

χρειάζονται γιά να καλύψουν τις περιοχές ανάματος της κάθε LED, και γράψτε τις αντίστοιχες λογικές συναρτήσεις γιά τις 7 εξόδους. Δεν χρειάζεται να σχεδιάσετε το κύκλωμα με πύλες, εδώ.

4.6 Άλγεβρα Boole

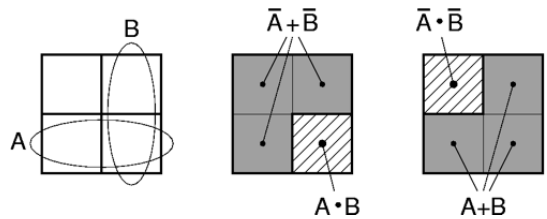
Οι λογικές πράξεις ΚΑΙ, Ή, ΌΧΙ πάνω σε δυαδικές ψηφιακές μεταβλητές ορίζουν μιάν Άλγεβρα, η οποία ονομάστηκε "**Άλγεβρα Boole**" (Boolean Algebra), προς τιμήν του Μαθηματικού George Boole ο οποίος δημοσίευσε τις πρώτες σχετικές ιδέες το 1849, καθώς εργάζονταν πάνω στη μαθηματική διατύπωση των κανόνων της λογικής νόησης. Περίπου 90 χρόνια αργότερα, το 1938, ο Claude Shannon έδειξε ότι η Άλγεβρα Boole, όπως είχε εν τω μεταξύ εξελιχθεί, περιγράφει και τη λειτουργία των κυκλωμάτων διακοπτών, όπως κάναμε κι εμείς στην αρχή του μαθήματός μας.

Η Άλγεβρα Boole μπορεί να δομηθεί ξεκινώντας από τον ("αξιωματικό") ορισμό των τριών πράξεων, ΚΑΙ, Ή, ΌΧΙ, βάσει του πίνακα αληθείας τους που είδαμε στην §1.1, δηλαδή από τον ορισμό τους μέσω της εξαντλητικής απαρίθμησης του αποτελέσματός τους γιά τον κάθε δυνατό συνδυασμό εισόδων. Γιά σκοπούς συντομογραφίας, από δω και πέρα, θα συμβολίζουμε τις λογικές αυτές πράξεις με AB [ή και με τελεία στη μέση] (A και B), A+B (A ή B), και A' [ή και με παύλα από πάνω] (όχι A), όπως είπαμε στην §3.5. Τις μεταβλητές της Άλγεβρας Boole, δηλαδή τις δυαδικές ψηφιακές μεταβλητές, τις λέμε και "*Μεταβλητές Boole*" (Boolean variables). Όπως έχουμε πεί, οι δύο τιμές μιάς μεταβλητής Boole μπορεί να συμβολίζουν πολλά και διαφορετικά πράγματα, π.χ. αναμένο-σβηστό, ζεστό-κρύο, πάνω-κάτω, μπρός-πίσω, αριστερά-δεξιά, πατημένος-ελεύθερος (διακόπτης), ψηλή-χαμηλή (ηλεκτρική τάση), περνάει - δεν περνάει (ρεύμα), αληθές-ψευδές, ναι-όχι, 1-0, κλπ. Γιά σκοπούς συντομογραφίας, και πάλι, συνήθως θα χρησιμοποιούμε τα σύμβολα **1** (αληθές, αναμένο, κλπ), και **0** (ψευδές, σβηστό, κλπ).

Ξεκινώντας από τον ορισμό των τριών πράξεων Boole, μπορούμε να αποδείξουμε πολλά θεωρήματα της Άλγεβρας Boole, τα οποία συχνά αντιστοιχούν και σε συνηθισμένες διατυπώσεις της καθημερινής μας λογικής σκέψης. Η απόδειξη μπορεί να γίνει με εξαντλητική επαλήθευση όλων των περιπτώσεων (πίνακας αληθείας), ή με διαγράμματα Venn, ή με τη χρήση άλλων θεωρημάτων. Εκθέτουμε εδώ κάμποσα τέτοια θεωρήματα, κατά σειρά σημαντικότητας.

Δύο Αρνήσεις κάνουν Μία Κατάφαση:
Όπως ξέρουμε πολύ καλά και από την καθημερινή μας ζωή, και όπως αποδεικνύεται άμεσα και από τον πίνακα αληθείας, το ΌΧΙ (ΌΧΙ A) είναι το ίδιο με το A, δηλαδή: (A')' = A.

Θεώρημα DeMorgan
(Άρνηση και Δυϊσμός):
Όπως παρατηρήσαμε ήδη κάμποσες φορές, από την §0.10 και μετά, η άρνηση του ΚΑΙ ισοδυναμεί με το Ή των αρνήσεων, ενώ η άρνηση του Ή ισοδυναμεί με το ΚΑΙ των αρνήσεων. Η ιδιότητα αυτή, γνωστή σαν Αρχή του Δυϊσμού (Duality Principle) ή "Θεώρημα DeMorgan", διατυπώνεται επίσημα ως εξής:




```
(AB)'  =  A'+B'
(A+B)' =  A'B'
```

Γιά να το αποδείξουμε, αρκεί να κατασκευάσουμε τους πίνακες αληθείας των παραπάνω συναρτήσεων και να τους συγκρίνουμε. Αν τους κατασκευάσουμε σε μορφή διαγράμματος Venn / χάρτη Karnaugh, και σημειώσουμε τα τετράγωνα όπου η κάθε συνάρτηση γίνεται 1, θα προκύψει το σχήμα που φαίνεται. Όπως βλέπουμε, η συνάρτηση AB γίνεται 1 στο κάτω δεξιά τετράγωνο, άρα το συμπλήρωμά της (η άρνησή της), δηλ. η συνάρτηση (AB)', θα γίνεται 1 στις υπόλοιπες περιπτώσεις (αριστερά και πάνω "Γ"). Αυτή η τελευταία περιοχή είναι ίδια με την ένωση (λογικό Ή) της περιοχής A' (A bar --δύο επάνω τετράγωνα) με την περιοχή B' (B bar --δύο αριστερά τετράγωνα), αποδεικνύοντας έτσι την πρώτη από τις παραπάνω σχέσεις. Αντίστοιχα μπορεί να αποδειχτεί και η δεύτερη σχέση, όπως φαίνεται

στο δεξί μέρος του σχήματος.

Εναλλακτικά, η δεύτερη σχέση μπορεί να προκύψει από την πρώτη και από την ιδιότητα ότι δύο αρνήσεις κάνουν μία κατάφαση. Ξεκινάμε από το δεξί μέλος της ισότητας που θέλουμε να αποδείξουμε, και το μετασχηματίζουμε με δύο αρνήσεις: A'B' = [(A'B')']'. Μέσα στις αγκύλες υπάρχει η άρνηση ενός λογικού ΚΑΙ, άρα μπορούμε να εφαρμόσουμε σε αυτήν το πρώτο θεώρημα DeMorgan, και να την μετατρέψουμε στο λογικό Ή των αρνήσεων, οι οποίες στη συνέχεια μπορούν να απλοποιηθούν: (A'B')' = (A')' + (B')' = A+B. Βάσει αυτού, η προηγούμενη ισότητα μας δίνει: A'B' = [(A'B')']' = [A+B]' = (A+B)', πράγμα που είναι ακριβώς το δεύτερο θεώρημα DeMorgan. Ο τρόπος αυτός απόδειξης μας οδηγεί σε μιά δεύτερη διατύπωση της αρχής του δυϊσμού: εάν σε μιάν ισότητα της άλγεβρας Boole αλλάξουμε όλα τα ΚΑΙ με Ή, και όλα τα Ή με ΚΑΙ, τότε προκύπτει μιά άλλη, επίσης αληθής ισότητα, η "δυϊκή" της πρώτης (όπως θα δούμε πιό κάτω, αν η ισότητα περιέχει και άσσους ή μηδενικά, τότε πρέπει και αυτά να τα αλλάξουμε από 0 σε 1 και από 1 σε 0).

Επιμεριστική Ιδιότητα (Distributive Property):
Κατ' ανάλογο τρόπο, μέσω του πίνακα αληθείας / διαγράμματος Venn, μπορεί κανείς να διαπιστώσει ότι:

```
A(B+C) = AB + AC
A+(BC) = (A+B)(A+C)
```

Ειπωμένο με λόγια, αν ισχύει το A και επίσης ισχύει το B ή το C, τότε θα πρέπει να ισχύει το A και το B ή να ισχύει το A και το C. Αντίστοιχα, η δεύτερη σχέση λέει ότι αν ισχύει το A ή ισχύει το B και το C, τότε θα ισχύει το A ή το B, καθώς επίσης θα ισχύει το A ή το C. Παρατηρήστε ότι όταν χρησιμοποιούμε τα παραπάνω σύμβολα του ΚΑΙ που μοιάζει με το σύμβολο του πολλαπλασιασμού και του Ή που μοιάζει με το σύμβολο της πρόσθεσης, τότε η πρώτη από τις παραπάνω σχέσεις μοιάζει οικεία, αλλά η δεύτερη καθόλου (αφού, φυσικά, δεν μιλάμε γιά πρόσθεση και πολλαπλασιασμό).

Όπως και με τις δύο μορφές του θεωρήματος DeMorgan, οι δύο παραπάνω σχέσεις είναι *δυϊκές* μεταξύ τους: αν αντικαταστήσουμε τα ΚΑΙ με Ή και τα Ή με ΚΑΙ, τότε προκύπτει η μία από την άλλη. Ο λόγος είναι ότι η δεύτερη μπορεί να προκύψει από την πρώτη, εφαρμόζοντας τα θεωρήματα DeMorgan (δηλαδή την αρχή του δυϊσμού), και το ότι δύο αρνήσεις κάνουν μία κατάφαση. Ξεκινώντας με το αριστερό μέλος της δεύτερης σχέσης, το μετασχηματίζουμε ώς εξής μέχρι να προκύψει το δεξί: A+(BC) = { [A+(BC)]'}' (δύο αρνήσεις) = { A'(BC)' }' (από DeMorgan) = { A'(B'+C') }' (από DeMorgan) = { A'B' + A'C' }' (από την πρώτη επιμεριστική ιδιότητα) = { (A+B)' + (A+C)' }' (από DeMorgan) = { [(A+B)(A+C)]' }' (από DeMorgan) = (A+B)(A+C) (δύο αρνήσεις).

Αντιμεταθετική και Προσεταιριστική Ιδιότητα (Commutative and Associative Property):
Όπως ξέρουμε, η σειρά των μεταβλητών δεν παίζει ρόλο στις πράξεις ΚΑΙ και Ή (αντιμεταθετική ιδιότητα), όπως επίσης στα πολλαπλά ΚΑΙ η σειρά των πράξεων δεν παίζει ρόλο, και το ίδιο και στα πολλαπλά Ή (προσεταιριστική ιδιότητα --γι' αυτό και συνήθως τα γράφουμε χωρίς παρενθέσεις). Παρατηρήστε και πάλι τα ζευγάρια δυϊκών σχέσεων:

```
 AB = BA       A+B = B+A                              (αντιμεταθετική)
 A(BC)  = (AB)C    [συνήθως γράφεται:  ABC   ]  (προσεταιριστική)
A+(B+C) = (A+B)+C  [συνήθως γράφεται:  A+B+C ]
```

Άλλα Θεωρήματα της Άλγεβρας Boole:
Μπορούν εύκολα να αποδειχτούν τα παρακάτω επίσης θεωρήματα. Τα δύο θεωρήματα σε κάθε γραμμή --αριστερό και δεξί-- είναι *δυϊκά* μεταξύ τους: το ένα προκύπτει από το άλλο ανταλλάζοντας τα ΚΑΙ με τα Ή, και τα 1 με τα 0.

```
A·0 = 0          A+1 = 1
A·1 = A          A+0 = A
A·A = A          A+A = A
A·A'= 0          A+A'= 1
A(A+B) = A       A+AB = A
A(A'+B)= AB      A+A'B= A+B
```

Άσκηση 4.7: Αποδείξεις Θεωρημάτων

*[Προκειμένου να αποφευχθεί ο κίνδυνος υπερφόρτωσής σας με ασκήσεις αυτή τη βδομάδα, την άσκηση αυτή, μαζί με την επόμενη, θα την παραδώσετε (σε χαρτί) μαζί με την αναφορά σας του **5ου** εργαστηρίου, την επόμενη βδομάδα. Όμως, κάντε την από τώρα, προκειμένου να μην υπερφορτωθεί στη συνέχεια η επόμενη βδομάδα σας...].*

(α) Αποδείξτε όλα τα παραπάνω θεωρήματα μέσω εξαντλητικού ελέγχου της ταυτότητας των δύο μελών τους σε όλες τις περιπτώσεις συνδυασμού τιμών των μεταβλητών τους. Με άλλα λόγια, φτιάξτε τους πίνακες αληθείας των δύο μελών κάθε θεωρήματος, και διαπιστώστε ότι είναι ίδιοι. Γράψτε τον πίνακα αληθείας της κάθε εμπλεκόμενης συνάρτησης σε μορφή στήλης, κατακόρυφα (όχι χάρτη Karnaugh).
(α1) Γιά τα θεωρήματα που εμπλέκουν μία μόνο μεταβλητή, ο πίνακας αληθείας θα έχει 2 γραμμές· φτιάξτε χωριστές στήλες γιά το A, το A', το (A')', το A·A, A+A, A·A', A+A', A·0, A·1, A+0, και A+1. Δείξτε με βέλη ποιές στήλες είναι ίσες με ποιές, και σε ποιό θεώρημα αντιστοιχεί κάθε τέτοιο ζευγάρι ίσων στηλών.
(α2) Γιά τα θεωρήματα που εμπλέκουν δύο μεταβλητές, A και B, ο πίνακας αληθείας θα έχει 4 γραμμές· φτιάξτε χωριστές στήλες γιά τα A', B', A'+B', A'B', AB, (AB)', A+B, (A+B)'. Αν έχετε χρόνο και διάθεση, φτιάξτε στήλες και γιά τα AB, A+AB, A+B, A(A+B), A', A'B, A+A'B, A'+B, A(A'+B). Δείξτε πάλι τις ίσες στήλες και τα θεωρήματα στα οποία αυτές αντιστοιχούν.
(α3) Γιά τα θεωρήματα που εμπλέκουν τρείς μεταβλητές, A, B, και C, ο πίνακας αληθείας θα έχει 8 γραμμές· φτιάξτε χωριστές στήλες γιά τα B+C, A(B+C), AB, AC, AB+AC, BC, A+(BC), A+B, A+C, (A+B)(A+C). Πάλι, δείξτε τις ίσες στήλες. Αν έχετε χρόνο και διάθεση, φτιάξτε στήλες και γιά τα AB, (AB)C, BC, A(BC), A+B, (A+B)+C, B+C, A+(B+C).

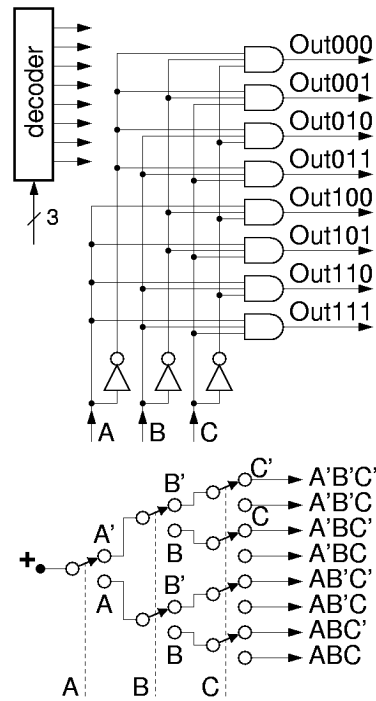
(β) Αποδείξτε με διαγράμματα Venn την επιμεριστική (και την προσεταιριστική;) ιδιότητα, στις δύο δυϊκές τους μορφές την κάθε μία, καθώς και τα θεωρήματα A(A+B)=A, A+AB=A, A(A'+B)=AB, και A+A'B=A+B. Χρησιμοποιήστε 2 ή 3 τεμνόμενες ελλείψεις, που παριστάνουν τα σύνολα A, B, και C. Σημειώστε με κατάλληλο χρώμα ή διαγράμμιση τις περιοχές του επιπέδου που αντιστοιχούν στα B+C, A(B+C), AB, AC, AB+AC, BC, A+(BC), A+B, A+C, (A+B)(A+C)· ποιές περιοχές είναι ίδιες με ποιές; Αν έχετε χρόνο και διάθεση, κάντε το ίδιο γιά τις περιοχές AB, (AB)C, BC, A(BC), A+B, (A+B)+C, B+C, A+(B+C), και τέλος γιά τις AB, A+AB, A+B, A(A+B), A'B, A+A'B, A'+B, A(A'+B).

Άσκηση 4.8: Αποκλειστικό-Ή και Αποκλειστικό-ΟΥΤΕ (Ισότητα)

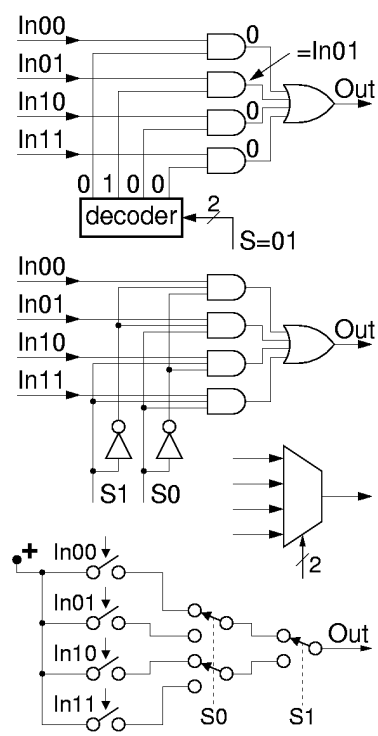
*[Προκειμένου να αποφευχθεί ο κίνδυνος υπερφόρτωσής σας με ασκήσεις αυτή τη βδομάδα, την άσκηση αυτή, μαζί με την προηγούμενη, θα την παραδώσετε (σε χαρτί) μαζί με την αναφορά σας του **5ου** εργαστηρίου, την επόμενη βδομάδα. Όμως, κάντε την από τώρα, προκειμένου να μην υπερφορτωθεί στη συνέχεια η επόμενη βδομάδα σας...].*

Είδαμε στο πείραμα 1.4 ότι το *αποκλειστικό-Ή* ("exclusive-OR" ή "**XOR**") δύο μεταβλητών είναι αληθές τότε και μόνο τότε όταν μία και μόνο μία από τις δυό τους είναι αληθής. Επίσης, η συνάρτηση ισότητας είναι αληθής όποτε και οι δύο μεταβλητές έχουν την ίδια τιμή.
(α) Φτιάξτε τους πίνακες αληθείας των δύο αυτών συναρτήσεων, και αποδείξτε μέσω αυτών ότι η συνάρτηση ισότητας είναι η άρνηση (το "συμπλήρωμα") της συνάρτησης αποκλειστικού-Ή. Γιά το λόγο αυτό, η συνάρτηση ισότητας ονομάζεται και "**αποκλειστικό-ΟΥΤΕ**" ("exclusive-NOR" ή "**XNOR**").
(β) Από τον χάρτη Karnaugh του αποκλειστικού-Ή έχουμε δεί ότι αυτό είναι: A XOR B = AB'+A'B. Αποδείξτε μέσω αλγεβρικών μετασχηματισμών ότι ισχύει επίσης: A XOR B = (A+B)(A'+B'). Ξεκινήστε από αυτή τη δεύτερη έκφραση, και εφαρμόστε πάνω της δύο φορές την επιμεριστική ιδιότητα του ΚΑΙ πάνω στο Ή, A(B+C)=AB+AC· στη συνέχεια, απλοποιήστε τους τέσσερεις όρους που προκύπτουν, μέχρι να φτάσετε στην πρώτη έκφραση.
(γ) Αφού η συνάρτηση ισότητας είναι η άρνηση του αποκλειστικού-Ή, θα ισχύει: A XNOR B = [A XOR B]' = [AB'+A'B]' = [(A+B)(A'+B')]'. Απλοποιήστε αυτές τις δύο τελευταίες εκφράσεις, εφαρμόζοντας το θεώρημα DeMorgan, μέχρι να φτάσετε να αποδείξετε ότι: A XNOR B = AB+A'B' = (A+B')(A'+B).

4.9 Αποκωδικοποιητής και Πολυπλέκτης με Πύλες



Γιά τον αποκωδικοποιητή μιλήσαμε ήδη στις παραγράφους 2.7 και 3.10: είναι ένα ψηφιακό κύκλωμα που έχει μία έξοδο γιά κάθε ένα, χωριστό συνδυασμό τιμών των εισόδων του. Η κάθε έξοδος ανάβει (γίνεται 1) μόνον όταν οι τιμές των εισόδων βρίσκονται στον αντίστοιχο συνδυασμό. Επομένως, κάθε φορά, μία και μόνο μία έξοδος είναι αναμένη --αυτή η οποία αντιστοιχεί στον παρόντα συνδυασμό τιμών των εισόδων. Από τα παραπάνω προκύπτει ότι ο πίνακας αληθείας της κάθε εξόδου, σε μορφή χάρτη Karnaugh, θα είναι παντού 0 εκτός ενός και μόνου τετραγώνου όπου θα είναι 1. Άρα, η συνάρτηση της κάθε εξόδου αντιστοιχεί στο λογικό ΚΑΙ όλων των μεταβλητών εισόδου --είτε αυτών καθεαυτών, είτε των συμπληρωμάτων τους· κάθε έξοδος θα έχει διαφορετικό συνδυασμό αληθών/συμπληρωμάτων γιά τις μεταβλητές εισόδου. Έτσι, η υλοποίηση του αποκωδικοποιητή με λογικές πύλες είναι αυτή που φαίνεται στο σχήμα --εδώ γιά την περίπτωση αποκωδικοποιητή 3-σε-8. Πάνω αριστερά στο σχήμα υπάρχει ένα συνηθισμένο, απλό σύμβολο γιά τον αποκωδικοποιητή· το σύρμα εισόδου με την πλάγια γραμμούλα και το "3" σημαίνει "τρία (αδελφά) σήματα (bits)", που και τα 3 μαζί ερμηνεύονται σαν μία *λέξη* των τριών bits. Στο κάτω μέρος του σχήματος φαίνεται και η παλαιά υλοποίηση του εργαστηρίου 1, με διακόπτες. Σε αυτήν βλέπουμε, π.χ., ότι ρεύμα θα φτάσει στην επάνω έξοδο όταν και μόνον όταν οι διακόπτες A, B, και C είναι όλοι "όχι πατημένοι", δηλαδή όταν (A')ΚΑΙ(B')ΚΑΙ(C'). Ομοίως, στην δεύτερη έξοδο θα φτάσει ρεύμα όταν και μόνον όταν A και B είναι "όχι πατημένοι" και C είναι πατημένος, δηλαδή όταν (A')ΚΑΙ(B')ΚΑΙ(C), κ.ο.κ. Παρατηρούμε ότι οι λογικές αυτές συναρτήσεις είναι ίδιες με τις αντίστοιχες του κυκλώματος με πύλες.



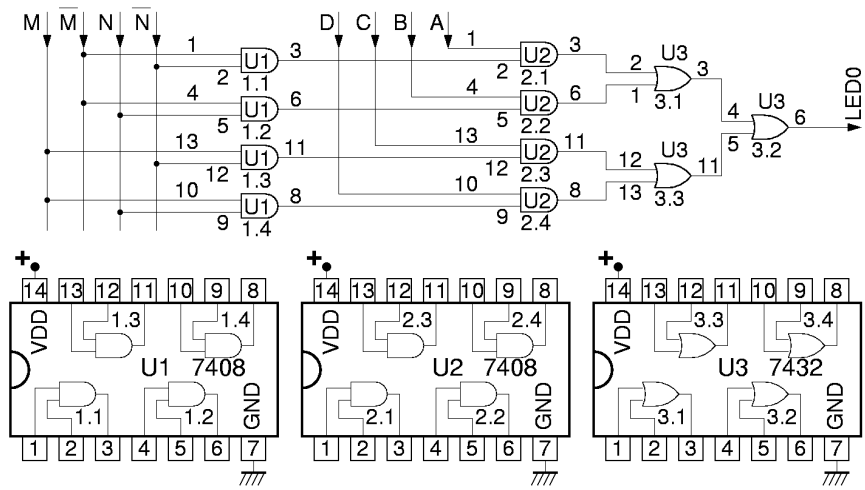
Γιά πολυπλέκτες μιλήσαμε αρχικά στην §1.5 (παραλλαγή με "αποκωδικοποιημένα" σήματα επιλογής, ένα ανά είσοδο δεδομένων), και μετά στην §3.1 γιά την παραλλαγή με μιάν είσοδο επιλογής (με όσα bits χρειάζεται) η οποία με την τιμή της επιλέγει μιά από τις εισόδους δεδομένων, και στη συνέχεια η τιμή αυτής της επιλεγμένης εισόδου δεδομένων οδηγείται και δίδεται στην έξοδο· αλλάζοντας την τιμή των bits επιλογής αλλάζει και το ποιά είσοδος δεδομένων οδηγείται στην έξοδο. Επίσης, στο πείραμα 3.1 παρατηρήσαμε ότι ο πολυπλέκτης έχει μιά συγγένεια με τον αποκωδικοποιητή, επειδή τα bits επιλογής πρέπει να αποκωδικοποιηθούν, ούτως ώστε γιά καθε διαφορετικό συνδυασμό τους να προκληθεί διαφορετική ροή πληροφοριών. Αυτό φαίνεται στο επάνω μέρος του σχήματος: σε κάθε συνδυασμό τιμών των bits επιλογής, δηλαδή σε κάθε έξοδο του αποκωδικοποιητή, αντιστοιχεί κι από μία είσοδος δεδομένων, In00 έως In11, καθώς και από μιά πύλη ΚΑΙ. Στο παράδειγμα που δείχνει το σχήμα (πολυπλέκτης 4-σε-1), τα (δύο) bits επιλογής έχουν τιμή 01, και γι' αυτό η δεύτερη έξοδος του αποκωδικοποιητή είναι αναμένη (1), και φυσικά όλες οι άλλες σβηστές (0). Επειδή `A·0 = 0` γιά οιοδήποτε A, όλες οι πύλες ΚΑΙ εκτός από την μία "επιλεγμένη" πύλη δίνουν έξοδο 0. Επειδή `A·1 = A` γιά οιοδήποτε A, η επιλεγμένη πύλη δίνει στην έξοδό της τιμή ίση με αυτήν της εισόδου δεδομένων της --δηλαδή της In01 στο εδώ παράδειγμα με S=01. Στη συνέχεια, οι έξοδοι όλων των πυλών ΚΑΙ οδηγούνται στις

εισόδους μάς μεγάλης πύλης Ή. Επειδή `A+0 = A` γιά οιοδήποτε A, οι είσοδοι 0 της πύλης Ή --που αντιστοιχούν σε όλες τις πύλες ΚΑΙ πλην της επιλεγμένης-- δεν επηρεάζουν την τιμή εξόδου της Ή· έτσι, η τελική αυτή έξοδος παίρνει την τιμή της μίας εισόδου της πύλης Ή που δεν είναι αναγκαστικά 0, δηλαδή της εξόδου της επιλεγμένης πύλης ΚΑΙ, η οποία όπως είδαμε ισούται με την τιμή της επιλεγμένης εισόδου δεδομένων (της In01 εδώ, γιά S=01).

Επειδή ο αποκωδικοποιητής αποτελείται και αυτός από πύλες ΚΑΙ, και επειδή (AB)C = A(BC) = ABC, οι πύλες ΚΑΙ του αποκωδικοποιητή μπορούν να συνενωθούν με τις πύλες ΚΑΙ που αυτές οδηγούν, δίνοντας το κύκλωμα που φαίνεται στο μέσον του σχήματος. Στο κάτω μέρος του σχήματος υπάρχει το παλαιό κύκλωμα του πολυπλέκτη με διακόπτες, από το πείραμα 2.6· παρατηρούμε ότι ουσιαστικά πρόκειται γιά την ίδια λογική συνάρτηση: ρεύμα μπορεί να περάσει από την θετική τροφοδοσία προς την έξοδο Out όταν βρεί διέξοδο μέσα από έναν από τέσσερεις εναλλακτικούς δρόμους (4 παράλληλοι δρόμοι αντιστοιχούν στο λογικό Ή 4 εισόδων). Ο πρώτος δρόμος άγει όταν In00 πατημένος και S0 και S1 όχι πατημένοι, δηλαδή όταν (In00)(S0')(S1'), που αντιστοιχεί στην πρώτη πύλη ΚΑΙ του νέου κυκλώματος, κ.ο.κ. γιά τους άλλους τρείς εναλλακτικούς δρόμους.

<u>Πείραμα 4.10</u>: Πολυπλέκτης 4-σε-1 με Πύλες

Κατασκευάστε και ελέγξτε έναν πολυπλέκτη 4-σε-1 με πύλες, όπως το πρώτο κύκλωμα του προηγουμένου σχήματος. Δεν χρησιμοποιούμε τη δεύτερη παραλλαγή του κυκλώματος επειδή δεν έχουμε πύλες AND 3 εισόδων· ομοίως, επειδή δεν έχουμε πύλες OR 4 εισόδων, χρησιμοποιούε ένα δέντρο τέτοιων πυλών των 2 εισόδων. Η πρώτη βαθμίδα του κυκλώματος είναι ένας αποκωδικοποιητής 2-σε-4 ίδιος με εκείνον του πειράματος 3.10· εκμεταλλευόμαστε και πάλι το γεγονός ότι η πλακέτα εισόδων/εξόδων μας δίνει τόσο τη θετική όσο και την αρνητική πολικότητα των διακοπτών M και N.



Στο σχήμα που δίδεται εδώ, οι πύλες είναι τοποθετημένες σε σημεία που βοηθούν στην ανθρώπινη κατανόηση της λειτουργίας του κυκλώματος, και όχι στις θέσεις που αυτές έχουν μέσα στα chips της υλοποίησης. Γιά να μπορούμε όμως να βρίσκουμε αμέσως σε ποιό σύρμα του κυκλώματος αντιστοιχεί το κάθε σημείο του σχήματος, η κάθε πύλη έχει το όνομα του chip που την υλοποιεί (U1, U2, U3,...), και ο κάθε ακροδέκτης πύλης έχει τον αριθμό του ακροδέκτη του αντίστοιχου chip όπου αυτός βρίσκεται στο κύκλωμα. Συνδέστε την έξοδο σε μία ενδεικτική λυχνία και ελέγξτε το κύκλωμά σας: η LED δείχνει πάντα την τιμή της σωστής "επιλεγμένης" εισόδου δεδομένων A, B, C, ή D; Όταν τελειώσετε **μην** χαλάσετε το κύκλωμά σας, διότι θα το χρειαστείτε στο επόμενο πείραμα.

4.11 Μεθοδολογία Αποσφαλμάτωσης (Debugging) Κυκλωμάτων

Καθώς αυξάνει η πολυπλοκότητα των κυκλωμάτων που έχετε να υλοποιήσετε, αυξάνουν και οι κίνδυνοι σφαλμάτων κατά την κατασκευή που κάνετε στο εργαστήριο. Πρώτο σας μέλημα πρέπει πάντα να είναι να φτιάχνετε κυκλώματα **σωστά εκ κατασκευής**: σ' ένα πολύπλοκο σύστημα είναι ευκολότερο να προλάβετε τα λάθη από το να προσπαθείτε εκ των υστέρων να βρείτε σε τι οφείλονται λανθασμένες συμπεριφορές του συστήματος σε σπάνιες και παράξενες περιπτώσεις εισόδων του (και το ίδιο ισχύει και γιά τα προγράμματα που γράφετε σε άλλα μαθήματα και κυρίως στην επαγγελματική σας σταδιοδρομία). Επειδή όμως τα λάθη είναι ανθρώπινα, μερικές φορές θα σας τύχει να κάνετε μερικά, οπότε πρέπει

να έχετε μια μεθοδολογία **ελέγχου** (test) του αποτελέσματος (είναι το κύκλωμα σωστό;), και αν ο έλεγχος αυτός διαπιστώσει λάθη χρειάζεστε μιά μεθοδολογία εύρεσής τους (**αποσφαλμάτωση** - debugging). Επιπλέον, υπάρχουν και άλλοι αστάθμητοι παράγοντες, όπως π.χ. ότι ενδέχεται μερικά chips ή μερικοί ακροδέκτες από chips να είναι καμένα/καμένοι, ή μερικοί ακροδέκτες ή σύρματα να μην κάνουν καλή επαφή στην πλακέτα. Τις δυσκολίες αυτές δεν έχετε να τις αντιμετωπίσετε μόνο στο εργαστηριο του μαθήματός μας, αλλά και με τα πραγματικά κυκλώματα, μελλοντικά στην επαγγελματική σας σταδιοδρομία· οπλιστείτε λοιπόν με υπομονή και μέθοδο, και μάθετε από τώρα να τις αντιμετωπίζετε.

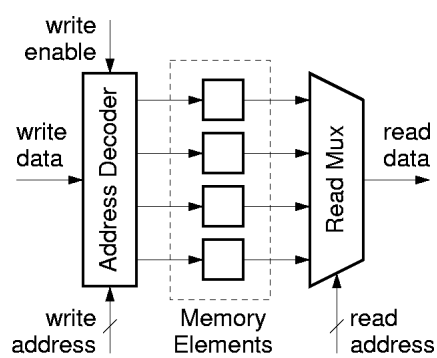
Ξεκινάτε πάντα την κατασκευή σας με ένα καθαρό, **κατανοητό** σχεδιάγραμμα του κυκλώματός σας όπως το σχήμα του παραπάνω πειράματος 4.10. Κάνετε τις συνδέσεις σας με την τροφοδοσία κλειστή. Μόλις ανάψετε την τροφοδοσία, ακουμπήστε κάθε chip με το δάκτυλο σας να δείτε αν **ζεσταίνεται** υπερβολικά. Αν υποπτευθείτε ότι κάποιο chip ζεσταίνεται απότομα, σβήστε αμέσως την τροφοδοσία: ίσως και να το προλάβετε πρίν καεί! Ελέγξτε αν οι τάσεις τροφοδοσίας είναι συνδεδεμένες στους σωστούς ακροδέκτες. Γιά το ύποπτο chip, ελέγξτε τις εξόδους του: ίσως κάποια από αυτές είναι βραχυκυκλωμένη με τάση τροφοδοσίας ή με άλλη έξοδο του ίδιου ή άλλου chip.

Όταν ελέγχετε την τιμή (0 ή 1) ενός ακροδέκτη, προτιμάτε να την ελέγχετε **πάνω στον ίδιο** τον ακροδέκτη του chip, ει δυνατόν, αντί πάνω σε κάποιο σύρμα που (υποτίθεται ότι) είναι συνδεδεμένο στον ακροδέκτη: εάν η σύνδεση είναι κακή, ο ακροδέκτης θα σας δείξει τι βλέπει ή τι βγάζει το ίδιο το chip, ενώ το σύρμα μπορεί και να μην κάνει καλή επαφή. Όταν έχετε συνδέσει, στο κύκλωμά σας, τους ακροδέκτες δύο chips μεταξύ τους, ελέγξτε πρώτα την τιμή πάνω στον έναν ακροδέκτη, και στη συνέχει πάνω στον άλλον ακροδέκτη· αν τις βρείτε διαφορετικές, σημαίνει ότι η σύνδεση δεν είναι καλή. Γιά να ελέγξτε την τιμή (τάση) ενός ακροδέκτη, χρησιμοποιήστε ένα σύρμα συνδεδεμένο σε μιά ενδεικτική λυχνία (LED).

Εάν το κύκλωμά σας δεν συμπεριφέρεται όπως πρέπει, **ιχνηλατήστε** (trace) το σφάλμα κινούμενοι είτε προς τα πίσω, από τη λανθασμένη έξοδο προς τις εισόδους που την επηρεάζουν, είτε προς τα εμπρός, από τις εισόδους προς τις εξόδους των πυλών. Ας πούμε ότι προχωρούμε από τις εισόδους προς τις εξόδους. Οι είσοδοι που εσείς δίνετε από τους διακόπτες, φτάνουν σωστές στα ποδαράκια του chip όπου φτάνουν; Αν όχι, φταίει κάποιο σύρμα ή σύνδεση. Αν όλες οι είσοδοι μιάς πύλης ενός chip έχουν τις σωστές τιμές (πάνω στα ποδαράκια του chip), η έξοδος αυτής της πύλης (πάνω στα ποδαράκια του chip) έχει τη σωστή τιμή; Αν όχι (και οι τροφοδοσίες είναι σωστές), υποπτευόμαστε είτε ότι η έξοδος είναι βραχυκυκλωμένη με τάση τροφοδοσίας ή με άλλη έξοδο του ίδιου ή άλλου chip, είτε ότι το chip μπορεί να είναι καμένο. Μετά τον έλεγχο των πρώτων πυλών που τροφοδοτούνται από τις εξωτερικές εισόδους, προχωρούμε στον έλεγχο των επομένων πυλών, που τροφοδοτούνται από τις εξόδους των πρώτων, κ.ο.κ.

Με ανάλογη μεθοδολογία προχωρούμε και από τις εξόδους πίσω προς τις εισόδους. Ποιά έξοδος δεν έχει τη σωστή τιμή; Ποιά πύλη τροφοδοτεί αυτή την έξοδο; Η πύλη αυτή, πάνω στα ποδαράκια του chip, έχει τη σωστή ή λάθος τιμή; Αν η έξοδος της πύλης (πάνω στα ποδαράκια του chip) είναι λάθος, τότε οι είσοδοί της (πάνω στα ποδαράκια του chip) τι τιμή έχουν; Η τιμή των εισόδων δικαιολογεί την τιμή της εξόδου; Αν το λάθος της εξόδου δεν δικαιολογείται από τις τιμές των εισόδων, μήπως η έξοδος είναι βραχυκυκλωμένη με τάση τροφοδοσίας ή με άλλη έξοδο του ίδιου ή άλλου chip; Αν το λάθος της εξόδου οφείλεται σε λανθασμένες τιμές των εισόδων, ποιός φταίει γι' αυτές; Έτσι προχωρούμε προς τα πίσω στο κύκλωμα, μέχρι να βρούμε τον αρχικό φταίχτη....

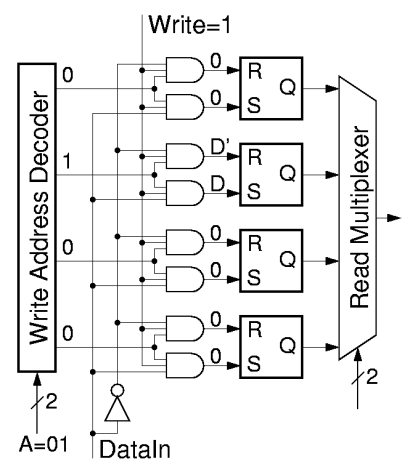
4.12 Μνήμη Τυχαίας Προσπέλασης (RAM)



Η *Μνήμη Τυχαίας Προσπέλασης* (Random Access Memory - **RAM**) αποτελεί τη μιά από τις τρείς βασικές συνιστώσες των ψηφιακών συστημάτων --αν θεωρήσουμε ότι οι επεξεργαστές και τα δίκτυα επικοινωνίας είναι οι άλλες δύο. Τη βασική οργάνωση μιάς μνήμης σε λέξεις αποτελούμενες από bits την είδαμε στην § 1.8, και στη συνέχεια, με αποκωδικοποιητή διευθύνσεων, στην §2.3. Επίσης,

στο πείραμα 3.4 είχαμε δεί τη βασική ιδέα γιά την κατασκευή ενός στοιχείου μνήμης που μπορεί να "θυμάται" 1 bit πληροφορίας. Οι σύγχρονες μνήμες RAM ημιαγωγών (ιδιαίτερα οι "στατικές" RAM - SRAM) χρησιμοποιούν ένα κύκλωμα CMOS βασισμένο στην ίδια ιδέα της θετικής ανάδρασης γιά να αποθηκεύουν το κάθε ένα bit πληροφορίας, και ως γνωστόν περιλαμβάνουν χιλιάδες ή (συνήθως) εκατομμύρια τέτοια κυκλώματα μέσα σε κάθε ένα chip, προκειμένου να χωρούν χιλιάδες ή εκατομμύρια bits πληροφορίας. Προφανώς όμως, τα chips αυτά δεν μπορούν να έχουν χιλιάδες ή εκατομμύρια σύρματα προκειμένου εμείς απ' έξω να διαβάζουμε ή να γράφουμε όλα αυτά τα bits. Η "προσπέλαση" της μνήμης, λοιπόν, γίνεται επιλεκτικά κάθε φορά, γιά την ανάγνωση από ή την εγγραφή σε ένα υποσύνολο μόνο των στοιχείων μνήμης --μία "λέξη" όπως έχουμε πεί. Το κύκλωμα έχει τη δυνατότητα να επιλέγει αυθαίρετα --"τυχαία"-- κάθε φορά το οιοδήποτε στοιχείο μνήμης επιθυμεί να προσπελάσει ο χρήστης, δίνοντας έτσι το όνομα σε αυτόν τον τύπο μνήμης. Το όνομα "τυχαίας προσπέλασης" αντιδιαστέλει αυτή τη μνήμη από άλλες μνήμες, π.χ. σειριακής προσπέλασης, όπως είναι π.χ. οι μαγνητικές ταινίες, ή --εν μέρει-- οι μαγνητικοί δίσκοι.

Γιά να επιτευχθεί η τυχαία προσπέλαση, η RAM χρησιμοποιεί έναν αποκωδικοποιητή γιά να επιλέξει το επιθυμητό υποσύνολο των στοιχείων μνήμης που ο χρήστης θέλει να προσπελάσει. Μπορούμε να φανταστούμε ότι ο αποκωδικοποιητής χρησιμοποιείται γιά τις εγγραφές (write) στη μνήμη, όπως δείχνει το σχήμα, αν και στην πραγματικότητα, γιά λόγους οικονομίας, χρησιμοποιείται και γιά τις αναγνώσεις. Η βασική είσοδος του αποκωδικοποιητή είναι η **διεύθυνση** (address), δηλαδή μιά ψηφιακή πληροφορία με τόσα bits όσα χρειάζονται γιά να επιλεγεί μονοσήμαντα μία λέξη. Οι άλλες είσοδοι που χρειάζονται γιά την εγραφή είναι μιά είσοδος ελέγχου (write enable), που να λέει πότε θέλουμε να γράψουμε και πότε όχι, και η είσοδος δεδομένων (write data), που να λέει τι θέλουμε να γράψουμε στην επιλεγείσα λέξη. Γιά την τυχαία προσπέλαση κατά την ανάγνωση, η RAM χρησιμοποιεί έναν πολυπλέκτη γιά να οδηγήσει στην έξοδο το περιεχόμενο που είναι αποθηκευμένο στην επιθυμητή λέξη. Φυσικά, ο πολυπλέκτης εμπεριέχει κι έναν αποκωδικοποιητή, και γιά να λειτουργήσει χρειάζεται κι αυτός τη *διεύθυνση* της λέξης. Έτσι, γιά καθε δοσμένη διεύθυνση ανάγνωσης (read address) θα εμφανίζονται στην έξοδο δεδομένων ανάγνωσης (read data) εκείνες οι πληροφορίες που είχαν γραφτεί την τελευταία φορά στην ίδια διεύθυνση εγγραφής, ενώ τα περιεχόμενα (πληροφορίες) των υπολοίπων θέσεων (διευθύνσεων) της μνήμης είναι αυθαίρετα, και δεν αλληλεπιδρούν με τα περιεχόμενα της θέσης που εμείς κοιτάμε αυτή τη στιγμή.

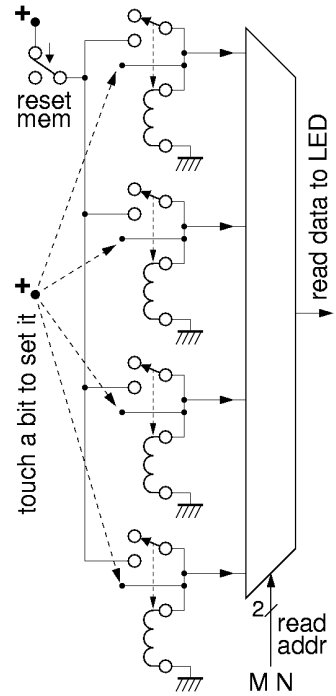


Γιά να συγκεκριμενοποιήσουμε την παραπάνω γενική εικόνα, ας υποθέσουμε ότι η μνήμη μας έχει λέξεις του 1 bit καθεμία (οι συνηθισμένες μνήμες έχουν λέξεις π.χ. των 8 ή 16 ή 32 ή 64 bits). Επίσης, γιά να μας χωράει το σχήμα, ας υποθέσουμε ότι η μνήμη μας χωράει μόνο 4 bits, συνολικά (αντί των χιλιάδων ή εκατομμυρίων bits των πραγματικών μνημών). Επίσης, κατ' αναλογία προς τις εισόδους Reset και Set του πειράματος 3.4, ας υποθέσουμε ότι κάθε στοιχείο μνήμης έχει δύο εισόδους, R και S, γιά τον έλεγχο εγγραφής: όταν R=S=0 δεν γίνεται καμία εγγραφή, και το στοιχείο διατηρεί (θυμάται) την προϋπάρχουσα κατάστασή του· όταν R=1 (καθώς S=0), το στοιχείο μηδενίζεται (εγγραφή πληροφορίας 0)· όταν S=1 (καθώς R=0), στο στοιχείο εγγράφεται η πληροφορία 1· και τέλος, θέλουμε ποτέ να μην είναι ταυτόχρονα αναμένο και το R και το S. Q είναι η έξοδος του στοιχείου μνήμης, δηλαδή το σύρμα που έχει πάντα πάνω του την τιμή του αποθηκευμένου bit. Τότε, η μνήμη αυτή, μεγέθους "4x1" (4 "λέξεις", μεγέθους 1 bit καθεμία), θα είναι όπως φαίνεται στο σχήμα. Η ανάγνωση γίνεται με τον πολυπλέκτη 4-σε-1, δηλ. το κύκλωμα της §4.9.

Η εγγραφή χρησιμοποιεί τον αποκωδικοποιητή διευθύνσεων, που είναι παρόμοιος με το κύκλωμα της §4.9, παραπάνω. Οι πύλες ΚΑΙ που ακολουθούν τον αποκωδικοποιητή εξασφαλίζουν τα εξής. Γιά όλα τα στοιχεία μνήμης εκτός του "επιλεγμένου" από τη διεύθυνση εγγραφής (εδώ, το δεύτερο στοιχείο, επειδή A=01), R=S=0 επειδή η αντίστοιχη έξοδος του αποκωδικοποιητή είναι 0, άρα δεν γίνεται καμία εγγραφή εκεί. Γιά το επιλεγμένο στοιχείο μνήμης, όταν Write=0, δηλαδή όταν δεν θέλουμε να κάνουμε καμία εγγραφή, τότε

πάλι R=S=0, επομένως ούτε εκεί γίνεται εγγραφή. Εάν τώρα θέλουμε να γίνει εγγραφή, δηλαδή εάν Write=1 όπως στο σχήμα, τότε το επιλεγμένο στοιχείο μνήμης βλέπει R=DataIn' και S=DataIn (επειδή οι άλλες δύο είσοδοι των πυλών ΚΑΙ είναι 1). Αυτό σημαίνει ότι εάν DataIn=0 τότε R=1 και S=0, άρα το στοιχείο μνήμης μηδενίζεται (δηλ. εγγράφεται 0=DataIn), ενώ εάν DataIn=1 τότε R=0 και S=1, άρα στο στοιχείο μνήμης εγγράφεται 1 (=DataIn). Η περίπτωση R=S=1 αποκλείεται, διότι τα DataIn' και DataIn (το ένα συμπλήρωμα του άλλου) δεν είναι ποτέ 1 και τα δύο ταυτοχρόνως.

Πείραμα 4.13: Μνήμη Ηλεκτρονόμων



Κατασκευάστε και ελέγξτε την απλοϊκή μνήμη τυχαίας προσπέλασης (RAM) μεγέθους 4x1 με ηλεκτρονόμους που φαίνεται στο σχήμα, κατ' αναλογία της μνήμης που είδαμε στην προηγουμένη παράγραφο. Γιά λόγους απλοποίησης, παραλείπουμε εντελώς το κύκλωμα εγγραφών, και το αντικαθιστούμε με τις πρόχειρες συνδέσεις που φαίνονται στο σχήμα (κατά συνέπεια, η μνήμη αυτή είναι RAM μόνον όσον αφορά τις αναγνώσεις, και όχι όσον αφορά τις εγγραφές). Τα 4 στοιχεία μνήμης είναι κατασκευασμένα με 4 ηλεκτρονόμους, κατ' αναλογία του πειράματος 3.4. Οι αναγνώσεις γίνονται μέσω του πολυπλέκτη 4-σε-1 με πύλες που έχετε έτοιμο από το προηγούμενο πείραμα 4.10. Η σύνδεση ηλεκτρονόμων και πυλών ημιαγωγών, εδώ, επιτρέπεται, διότι τα σημεία που τροφοδοτούν τις πύλες δεν είναι ποτέ ανοικτοκυκλωμένα, και η τάση τους είναι πάντα μεταξύ 0 και 5 Volt: όταν ο ηλεκτρονόμος είναι σβηστός, το σημείο που τροφοδοτεί την πύλη ΚΑΙ έχει 0 Volt, δεδομένου ότι η πεπερασμένη αντίσταση του πηνίου το συνδέει με τη γή, ενώ όταν ο ηλεκτρονόμος είναι αναμένος, το σημείο εκείνο συνδέεται με την θετική τροφοδοσία, άρα έχει 5 Volt.

Οι εγγραφές στη μνήμη μας γίνονται με ένα τρόπο καθόλου βολικό, λόγω των απλοποιήσεων που αναγκαστήκαμε να κάνουμε. Στην αρχή, μηδενίζουμε τα περιεχόμενα ολόκληρης της μνήμης, φέρνοντας τον πάνω αριστερά διακόπτη στη θέση "reset memory", δηλαδή διακόπτοντας την τροφοδοσία σε όλα τα στοιχεία μνήμης. Εν συνεχεία, επαναφέρουμε την τροφοδοσία (διακόπτης σε θέση ηρεμίας), και γράφουμε άσσους (1) επιλεκτικά, σε όποια στοιχεία μνήμης θέλουμε, ακουμπώντας ένα σύρμα του οποίου η μιά του άκρη είναι στη θετική τροφοδοσία στον ακροδέκτη ενεργοποίησης του πηνίου που θέλουμε να ανάψει· μόλις το πηνίο ανάψει μπορούμε να απομακρύνουμε το σύρμα, αφού το στοιχείο μνήμης "θυμάται" μόνο του ότι το ανάψαμε. Γιά να αλλάξουμε μερικούς ή όλους τους άσσους σε μηδενικά, ο μόνος τρόπος είναι να ξαναμηδενίσουμε όλη τη μνήμη μέσω του διακόπτη "reset memory" και να γράψουμε άσσους από την αρχή σε όλα τα μέρη όπου τους θέλουμε.

Ελέγξτε τη σωστή λειτουργία της μνήμης ως εξής. Γράψτε σ' ένα χαρτί 4 bits (π.χ. 1, 0, 0, 1) που θέλετε να αποθηκεύσετε στη μνήμη, με τη σειρά που θέλετε να τα αποθηκεύσετε. Μετά, γράψτε αυτά τα bits στη μνήμη, με τον χειροκίνητο τρόπο που είπαμε παραπάνω. Στη συνέχεια, δώστε διάφορες διευθύνσεις M και N (00, 01, 10, 11), εναλλάξ, επανειλημμένα, και ανακατωμένα. Γιά κάθε διεύθυνση, η ενδεικτική λυχνία στην έξοδο της μνήμης (του πολυπλέκτη) δίνει το σωστό περιεχόμενο σύμφωνα με το τι είχατε γράψει εκεί; Επαναλάβετε από την αρχή, με διαφορετικές πληροφορίες εγγραφής στη μνήμη (π.χ. 0, 1, 1, 1). Υπάρχουν 16 διαφορετικά σύνολα πληροφοριών που μπορείτε να αποθηκεύσετε σε αυτή τη μνήμη --αν έχετε χρόνο, ελέγξτε τη σωστή λειτουργία της μνήμης με όσα περισσότερα από αυτά γίνεται (φυσικά, δεν θα ελέγχατε με τέτοιον εξαντλητικό τρόπο μιά μνήμη π.χ. του 1 Mbit, διότι δεν θα τελειώνατε ποτέ (2 εις την ένα εκατομμύριο συνδυασμοί...) --πρέπει όμως να ελέγξτε ότι δεν είναι καμένο κανένα από τα στοιχεία μνήμης και καμία από τις πύλες του πολυπλέκτη).

Up to the Home Page of CS-120

last updated: 1 Nov. 2021, by M. Katevenis.