1η Σειρά Ασκήσεων
Συστήματα Μικροϋπολογιστών

Βερέμης Αλέξανδρος 03115063
Σπυρίδων Κρητικός 03116133

1η Άσκηση:

Με βάση τους πίνακες που βρήκαμε στο αρχείο mLab , οι απαραίτητες εντολές είναι οι εξής :

MVI κ,byte←→0E (Μετακινεί τον αριθμό στον καταχωρητή κ={A,B,C,D,E,H,L})

LDA adr←→3A (Φορτώνει στον καταχωρητή A το περιεχόμενο που βρίσκεται στην διεύθυνση adr= $(2000)_{16}$)

RAL←→17 (Περιστρέφει τα bits του καταχωρητή A και εκχωρεί στο CY(bit του καταχωρητή της σημαίας) το A0)

JC label←→DA (Αν CY=1 μεταβαίνει στο label =$(080D)_{16}$ που δίνεται σε μορφή διεύθυνσης 2 byte)

DCR κ←→0D (Αφαιρεί 1 από το περιεχόμενο του καταχωρητή κ={A,B,C,D,E,H,L})

JNZ label ←→C2(αν Z≠0 (το z=1 δίνεται από την C2 σημαία αν η προηγούμενη εντολή είχε ως αποτέλεσμα 0) τότε μεταβαίνει στο label=$(0805)_{16}$)

MOV κ1,κ2←79 (Αντιγράφει το περιεχόμενο του κ2 στον κ1)

CMA←2F (Συμπληρώνει ως προς 1 το περιεχόμενο του καταχωρητή A)

STA adr ←32(Αποθηκεύει το περιεχόμενο του καταχωρητή A στην διεύθυνση adr = $(3000)_{16}$)

RST 1←CF (Διακόπτει τον κώδικα και πηγαίνει τον PC στην διεύθυνση $(0800)_{16}$)

Παρακάτω καταγράφουμε τον ζητούμενο κώδικα , χρησιμοποιώντας τις παραπάνω εντολές :

```
START:

MVI C,08H

LDA 2000H

L2:

RAL

JC L1

DCR C

JNZ L2

L1:

MOV A,C

CMA
```

STA 3000H

RST 1

END

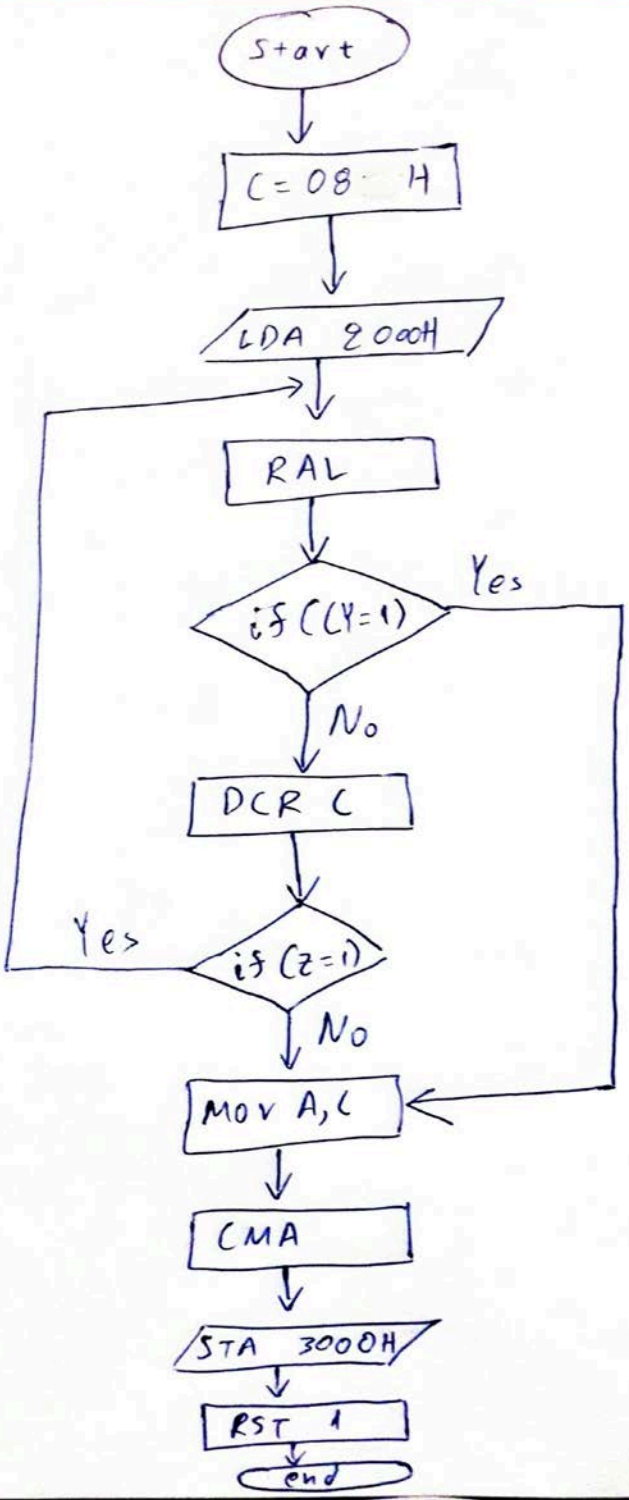
Στη συνέχεια, ο κώδικας κατ’ υπόδειξη προσομοιώθηκε στον προσομοιωτή TSIK.

Έτσι, βρέθηκε ότι αυτό που κάνει είναι ότι, ανάλογα με το ποιο (σε σειρά μετρώντας δεκαδικά από τα δεξιά προς τα αριστερά) διακοπτάκι είναι ανοιχτό, ανοίγουν τα αντίστοιχα λεντάκια σε δυαδική αναπαράσταση.

Δηλαδή, αν ανοίξω το 4ο διακοπτάκι (από δεξιά προς τα αριστερά 4ο) θα ανάψει το 3ο λεντάκι (από δεξιά προς τα αριστερά) το οποίο είναι η δυαδική αναπαράσταση του 4 = 2^ 2 .

Αν τώρα θέλω να επαναλαμβάνεται για πάντα ο κώδικας, το μόνο που χρειάζεται είναι να αντικαταστήσω την εντολή RST 1 με την εντολή JMP START .

Παρακάτω βρίσκεται το ζητούμενο διάγραμμα ροής:


Start
C = 08 H
LDA 2000H
RAL
if (CY=1)
Yes
No
DCR C
if (Z=1)
Yes
No
MOV A, C
CMA
STA 3000H
RST 1
end

2η Άσκηση

Με βάση τις οδηγίες και χρησιμοποιώντας τον προσομοιωτή , γράψαμε τον παρακάτω κώδικα, όπου παρατίθεται συνοπτική περιγραφή της λειτουργίας του με τη μορφή σχολίων.

START:

IN 10H

LXI B,0244H //Εκχωρώ στους καταχωρητές B ←$(44)_{16}$ C← $(02)_{16}$ (τους χρησιμοποιώ

//αργότερα για να καλέσω την υπορουτίνα DELB ώστε να πραγματοποιήσω

//καθυστέρηση $(0244)_{16}$ =$(500)_{10}$ ms).

MVI E,01H // Εκχωρώ στον καταχωρητή E το $(01)_{16}$ με το οποίο ορίζω και ποιό λεντάκι θα
// ανάψει αρχικά.

MAIN:

LDA 2000H // Εκχωρώ την είσοδο που δίνω από τον διακόπτη στον A.

MOV D,A // Χρησιμοποιώ τον D καταχωρητή για να διατηρηθούν προσωρινά τα
// δεδομένα του A.

RAR // Ολισθαίνω μια θέση δεξιά τα bits του A ώστε να αποθηκευτεί στο CY το LSB.

JC MAIN //Αν το LSB είναι 1, τότε μεταβαίνει στην MAIN δηλαδή ξαναπαίρνει είσοδο (δεν

// αλλάζει κίνηση).

MOV A,D

CALL DELB // Καθυστερεί για 0.5 secs .

RAL // Περιστρέφει τα bits του A αριστερά και αποθηκεύει στο CY το MSB.

JNC LOOP_R // Αν το MSB είναι 0, τότε εκτελεί την δεξιά περιστροφή των leds.

LOOP_L: // Αλλιώς την αριστερή.

MOV A,E // Καταχωρείται στον A ο αριθμός του led που θα ανάψει.

CMA // Συμπληρώνω το A ως προς 1 καθώς τα leds είναι αντίστροφης λογικής.

STA 3000H // Οδηγώ στην έξοδο το A για να ανάψει το κατάλληλο led.

CMA // Ξανασυμπληρώνω για να πάρω πάλι το αρχικό περιεχόμενο του A.

RLC // Ολίσθηση μια θέση αριστερά για να οριστεί το επόμενο led που θα ανάψει.

MOV E,A

```
JMP MAIN // Πηγαίνω ξανά στην main για να ελέγξω αν ενημερώθηκε η είσοδος.

LOOP_R: // Όμοια με πρίν.

MOV A,E

CMA

STA 3000H

CMA

RRC // Ολίσθηση μία θέση δεξιά για να οριστεί το επόμενο led που θα ανάψει.

MOV E,A

JMP MAIN

END
```

3η Άσκηση

Με βάση το παράδειγμα και χρησιμοποιώντας ως βοηθητικό εργαλείο τον προσομοιωτή, γράψαμε τον παρακάτω κώδικα :

```
LXI B,0244H ; Αποθηκεύω στο ζεύγος καταχωρητών B-C τα 500ms

START:

LDA 2000H ; Φορτώνω την είσοδο στον A

CPI 63H ;Συγκρίνω αν ο A είναι μεγαλύτερος από το (63)16=9910

JNC LABEL ; Αν είναι μεταβαίνει στην ετικέτα LABEL αλλιώς συνεχίζει κανονικά

MVI D,FFH ; Αποθηκεύω στον D το συμπλήρωμα ως προς 1 του -1.

DECA:

INR D ;Αυξάνω κατά 1 το περιεχόμενο του D

SUI 0AH ;Μειώνω κατά 10 το περιεχόμενο του A

JNC DECA ;Αν δεν είναι μικρότερο του 0 συνεχίζει να μειώνεται κατα 10 (πηγαίνει στο DECA)

ADI 0AH ;Αφού γίνει μικρότερο του 0 αυξάνει κατά 10
```

MOV E,A ; Μετακινεί το περιεχόμενο του A στον E προσωρινά

MOV A,D ; Μετακινεί το περιεχόμενο του D (τις δεκάδες δηλαδή) στον A

RLC ; 4 φορές ολίσθηση ώστε οι δεκάδες να αποθηκευτούν στα πρώτα 4 bits του A

RLC

RLC

RLC

ADD E ; Προσθέτω στον A τον E ώστε οι μονάδες να αποθηκευτούν στα τελευταία 4 bits

CMA ;Συμπληρώνω τον A γιατί οι έξοδοι έχουν αρνητική λογική

STA 3000H; Αποθήκευση-εμφάνιση μέσα από τα leds , βρίσκονται σε αυτήν την
; διεύθυνση(3000 H)

JMP START ;Επαναλαμβάνει από την αρχή

LABEL: ;Άμα είναι μεγαλύτερο του 100

MVI A,F0H ;Βάζει στο A τον αριθμό $(1111\ 0000)_2$

STA 3000H ;Αποθήκευση- Εμφάνιση

CALL DELB ; Καθυστερεί 0.5 sec

CMA ;Συμπληρώνει ώστε να έχει $(0000\ 1111)_2$

STA 3000H ;Αποθήκευση-Εμφάνιση

CALL DELB ; Καθυστέρηση

JMP START ; Ξαναπηγαίνει στην αρχή για να δεί αν άλλαξε η είσοδος

END

4η Άσκηση

Παρακάτω παρουσιάζονται οι 4 δοθείσες τεχνολογίες και παρατίθενται οι καμπύλες κόστους ανά τεμάχιο (ανά 2 τεχνολογίες), ώστε να διευκολυνθεί στη συνέχεια η σύγκρισή τους.

1η τεχνολογία

Χρήση διακριτών στοιχείων και I.C. όπως μικροελεγκτών, περιφερειακών, μνημών κλπ. Τοποθετημένα σε μια σχετικά μεγάλη πλακέτα. Το αρχικό κόστος σχεδίασης υποθέτουμε ότι είναι 20.000€. Το κόστος των I.C. ανά τεμάχιο υποθέτουμε ότι είναι 10€ και η κατασκευή της πλακέτας με την συναρμολόγησή της επίσης 10€.

Η συνάρτηση του κόστους ανά τεμάχιο είναι η εξής:

Κόστος =20.000+20x , όπου x: το πλήθος των τεμαχίων

2η τεχνολογία

Χρήση FPGAs και μικρού αριθμού περιφερειακών τοποθετημένα σε μια πλακέτα. Αρχικό κόστος σχεδίασης: 10.000€, κόστος ανά τεμάχιο των I.C.: 30€, κόστος πλακέτας ανά τεμάχιο και συναρμολόγησης: 10€.

Η συνάρτηση του κόστους ανά τεμάχιο:

Κόστος =10.000+40x , όπου x: το πλήθος των τεμαχίων

3η τεχνολογία

Σχεδίαση ειδικού SoC-1 με μια μικρή πλακέτα. Αρχικό κόστος σχεδίασης: 100.000€, κόστος ανά τεμάχιο των I.C.: 2€, κόστος πλακέτας και συναρμολόγησης ανά τεμάχιο: 2€.

Η αντίστοιχη συνάρτηση του κόστους ανά τεμάχιο:

Κόστος =100.000+4x , όπου x: το πλήθος των τεμαχίων

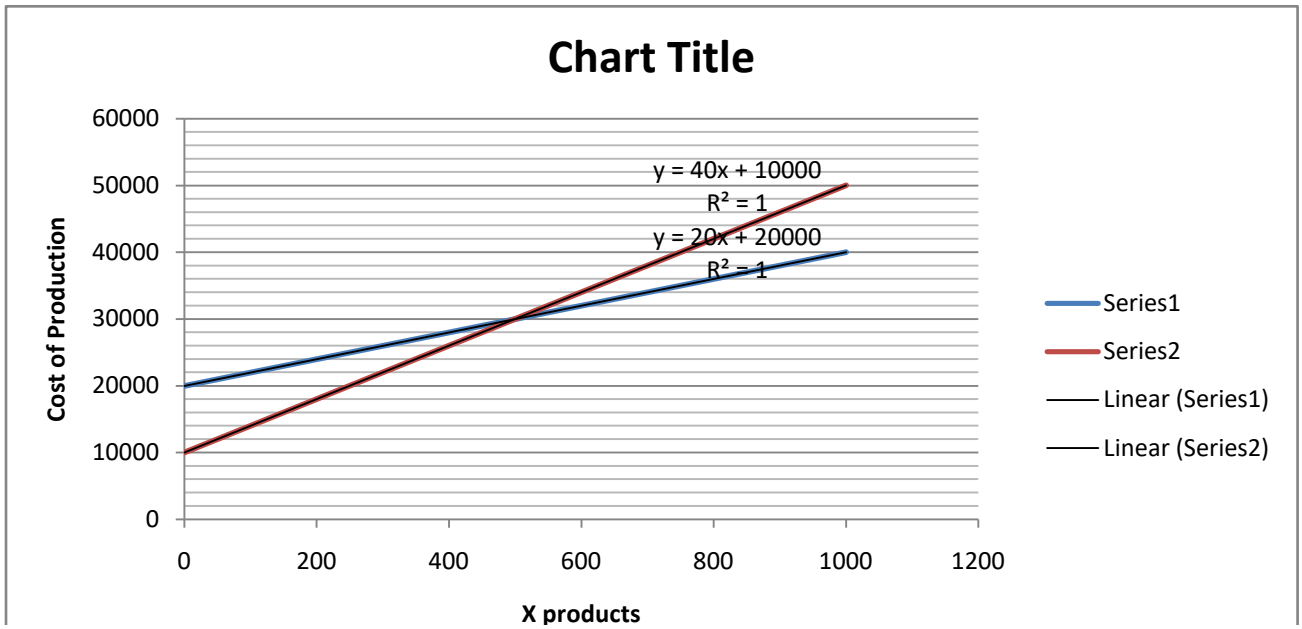
4 η τεχνολογία

Σχεδίαση ειδικού SoC-2 με μια πολύ μικρή πλακέτα. Αρχικό κόστος σχεδίασης: 200.000€, κόστος ανά τεμάχιο των I.C.: 1€, κόστος πλακέτας και συναρμολόγησης ανά τεμάχιο: 1€.

Όπου η συνάρτηση του κόστους ανά τεμάχιο:

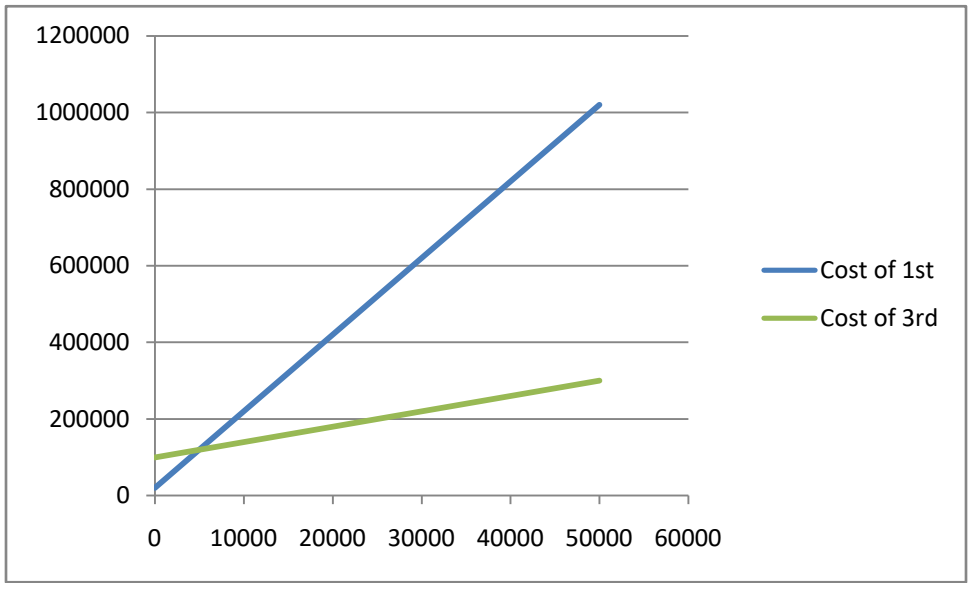
Κόστος =200.000+2x , όπου x: το πλήθος των τεμαχίων

Οι καμπύλες κόστους ανά τεμάχιο των τεχνολογιών 1 και 2:

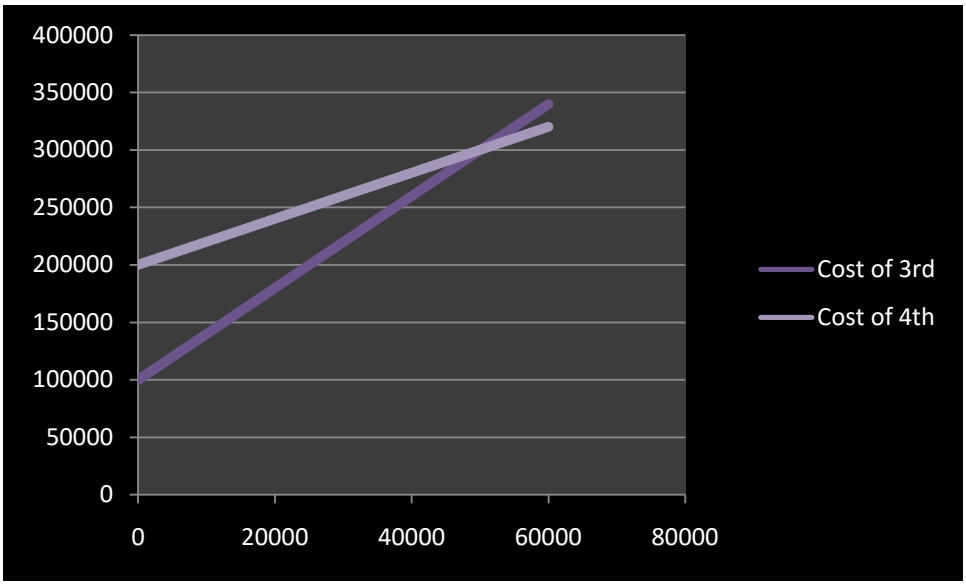


Με μπλε απεικονίζεται η τεχνολογία 1 και με μπορντό η τεχνολογία 2.

Οι καμπύλες κόστους ανά τεμάχιο των τεχνολογιών 1 και 3 :



Οι καμπύλες κόστους ανά τεμάχιο των τεχνολογιών 3 και 4 :



Οι πίνακες του Excel απο όπου προέκυψαν τα διαγράμματα:

Number ofproducts	Cost of 1st	Cost of 2nd	Cost of 3rd	Cost of 4th
0	20000	10000	100000	200000
250	25000	20000	101000	200500
500	30000	30000	102000	201000
750	35000	40000	103000	201500
1000	40000	50000	104000	202000
5000	120000	210000	120000	210000
50000	1020000	2010000	300000	300000
60000			340000	320000

Στη συνέχεια, υπολογίζονται οι περιοχές του πλήθους των τεμαχίων για τις οποίες κάθε τεχνολογία θεωρείται καταλληλότερη. Παρατηρούμε ότι από τη μία τεχνολογία στην επόμενη, όσο αυξάνεται το αρχικό κόστος της σχεδίασης, τόσο προτιμότερη είναι η τεχνολογία για μεγαλύτερο πλήθος τεμαχίων.

➔Οι τεχνολογίες 1 και 2 καταλήγουν να εμφανίζουν ίδιο κόστος για πλήθος τεμαχίων x1, όπου:

20.000+20x1=10.000+40x1 =>

=> **x1=500**

Κατά συνέπεια, δεδομένου ότι η καμπύλη που αντιστοιχεί στην 2η τεχνολογία εμφανίζει μεγαλύτερη κλίση και μικρότερο αρχικό κόστος, καταλήγουμε στο συμπέρασμα ότι για πλήθος τεμαχίων <500, συμφέρει περισσότερο η 2η τεχνολογία, ενώ για πλήθος τεμαχίων>500 η 1η τεχνολογία θεωρείται καταλληλότερη,αφού το κόστος της $2^{ης}$ ειναι αυξανόμενο.

~ Οι τεχνολογίες 1 (γιατι μας ενδιαφέρει για προιόντα >500 μονάδων)και 3 καταλήγουν να εμφανίζουν ίδιο κόστος για πλήθος τεμαχίων x2, όπου:

20.000+20x2=100.000+4x2=>

=> **x2=5.000**

Με την ίδια λογική , οδηγούμαστε στο συμπέρασμα ότι η 1η τεχνολογία συμφέρει περισσότερο για πλήθος τεμαχίων <5.000, ενώ η 3η τεχνολογία προτιμάται για πλήθος τεμαχίων>5.000 καθώς το μεγαλύτερο αρχικό κόστος σχεδίασης από αυτό το σημείο και έπειτα, αντισταθμίζεται από την πολύ μικρή της κλίση.

~ Τέλος οι τεχνολογίες 3 (αφού κοιτάζουμε προιόντα άνω των 5000 μονάδων) και 4 καταλήγουν να εμφανίζουν ίδιο κόστος για πλήθος τεμαχίων x3, όπου:

100.000+4x3=200.000+2x3=>

=> **x3=50.000**

Συνεπώς, ομοίως με πριν, συμπεραίνουμε ότι η 3η τεχνολογία συμφέρει περισσότερο για πλήθος τεμαχίων <50.000, ενώ η 4η τεχνολογία προτιμάται για πλήθος τεμαχίων>50.000.

Συνολικά λοιπόν, οι περιοχές στις οποίες κάθε τεχνολογία προτιμάται είναι οι εξής:

♣ *1η τεχνολογία*

500< πλήθος τεμαχίων< 5.000

♣ *2η τεχνολογία*

0<πλήθος τεμαχίων<500

♣ *3η τεχνολογία*

5.000<πλήθος τεμαχίων<50.000

♣ *4η τεχνολογία*

Πλήθος τεμαχίων>50.000

Στη συνέχεια διερευνάται για ποια τιμή αρχικού κόστους των SoC-1 (αλλαγή στις 100.000€) θα μπορούσε να εξαφανιστεί η επιλογή της 1ης τεχνολογίας.

~ Η ιδέα είναι η εξής:

Η 1η τεχνολογία θα εξαφανιστεί τελείως μόνο αν θεωρηθεί ότι δεν συμφέρει ακόμη και για ένα τεμάχιο, για το πρώτο ζητούμενο.

Για ένα τεμάχιο, το κόστος της 1ης τεχνολογίας ανέρχεται στα 20.020 ευρώ. Έστω ότι το νέο αρχικό κόστος σχεδίασης των SoC-1 είναι y1. Τότε, για ένα τεμάχιο, το κόστος της 3ης τεχνολογίας ανέρχεται στα y1+4 ευρώ (όπου τα 4 ευρώ αφορούν το κόστος της πλακέτας και της συναρμολόγησης ανά τεμάχιο).

Η 1η τεχνολογία θα εξαφανιστεί τελείως εφόσον αυτά τα κόστη εξισωθούν:

y1+4=20.020

Άρα : **Κόστος κατασκευής=20.016 ευρώ**

~ Τώρα, θα διερευνηθεί ποια θα πρέπει να είναι η τιμή ανά τεμάχιο στην τεχνολογία των FPGAs (συμβολίζεται ως κ), ούτως ώστε η 1η τεχνολογία να εξαφανιστεί.

Έστω ότι η 1η τεχνολογία είναι πάντα καλύτερη από την 2η για οποιονδήποτε αριθμό τεμαχίων x.Αφού η $2^{η}$ τεχνολογία συναγωνίζεται μόνο με την $1^{η}$ και αυτό μεχρι τα 500 τεμάχια όπως είδαμε. Για αυτό συγκρίνουμε $2^{η}$ με $1^{η}$.

Τότε θα ισχύει:

$20.000+20x<10.000+(κ+10)x$

$\Rightarrow 10.000< (κ-10)x$

$\Rightarrow 0 < 10.000/x < (\kappa-10)$

Δηλαδή πρέπει $\kappa-10>0 \Rightarrow \kappa>10$

Συνεπώς, προκειμένου να είναι η 2η τεχνολογία προτιμότερη από την πρώτη, θα πρέπει $\kappa \leq 10$. Έστω, λοιπόν, $\kappa=10$. Τότε για κάθε πλήθος τεμαχίων η 2η τεχνολογία υπερισχύει της πρώτης αφού $10.000 + 20x < 20.000 + 20x$. Αφού με αυτήν την τιμή του κ, θα έχουμε σταθερά την $2^{η}$ τεχνολογία χαμηλότερα σε κόστος σε σχέση με την $1^{η}$ για κάθε μονάδα προϊόντος!

Οπότε επιλέγεται η τιμή ανά τεμάχιο=10 ευρώ.

5η Άσκηση | i)

3.20)α)

```
module schema3_20_a(F, A, B, C, C_tone, D);
output F;
input A, B, C, C_tone, D;
wire w1, w2, w3, w4;
and G1(w1, C, D);
or G2(w2, w1, B);
and G3(w3, w2, A);
and G4(w4, B, C_tone);
or G5(F, w3, w4);
endmodule
```

3.21)β)|.

```
module schema3_21_b(F, A, B, D, At, Bt, Ct);
output F;
input A, B, D, At, Bt, Ct;
wire w1, w2, w3, w4, w5;
nand G1(w1, A, Bt);
nand G2(w2, At, B);
nand G3(w4, Ct, D);
nand G4(w3, w1, w2);
nand G5(w5, w3, w4);
not G6(F, w5);
endmodule
```

3.24)

```
module schema3_21 (A, B, C, D, E_not, F);
output F;
input A, B, C, D, E_not;
wire w1, w2;
nor G1(w1, A, B);
nor G2(w2, C, D);
nor G3(F, w1, w2, E_not); // Λόγω DeMorgan
endmodule
```

3.25)

```
module schema3_25 (A, B, C, A_not, B_not, D_not, F);
output F;
input A, B, C, A_not, B_not, D_not;
wire w1, w2, w3, w4;
nor G1(w1, A_not, B); // Λόγω DeMorgan
nor G2(w2, A, B_not);
nor G3(w4, C, D_not);
nor G4(w3, w1, w2);
nor G5(F, w3, w4);
endmodule
```

ii) Με εντολές διαρκούς ανάθεσης συντάξτε σε Verilog τα ακόλουθα οχήματα δηλαδή με την εντολή assign.

3.20b)

```
module schema3_20_b(A,B,C,D,C_not,B_not,F);
input A, B, C, D;
output F;
assign F=(((((C&D)|(~B))&A)|(B&~C));
endmodule
```

3.21 a)

```
module schema3_21_a(A,B,C,D,F);
input A,B,C,D;
output F;
assign F=(((A &~B)|(~A&B))&(C|~D));
endmodule
```

3.24)

```
module schema3_24(A,B,C,D,E,F);
input A,B,C,D,E;
output F;
assign F= ((A|B)&(C|D)&E);
endmodule
```

3.25)

```
module schema3_25(A,B,C,D,F);
input A,B,C,D;
output F;
assign F= ((A &~B)|(~A &B))&(C|~D);
endmodule
```

6η ΑΣΚΗΣΗ. i) 4.36 (σε επίπεδο πυλών)

```
module schema4_23 (D1,D2,D3,D0, x,y, V);
input   D0,D1,D2,D3;
output    x,y,V;
wire      w1, w2;
not       G1(w1,D2);
and       G2(w2,w1,D1);
 or       G3(y,D3,w2);
 or       G4(x,D3,D2);
 or       G5(V,x,D1,D0);
endmodule
```

ii) 4.45 (περιγραφή συμπεριφοράς).

```
module  schema4_45( D, x,y,V);
 input    [3:0] D;
output      x,y,V;
always @ (D)
   if ( D[2] or D[3]) x = 1; //αγοί x=D2+D3
   if (D[0] or D[1] or D[2] or D[3]) V=1; //αγοί V=...
   if ( D[3]) y = 1;
   else
        if ( D[1] && D[2]==0) y=1;
        // αγοί  y = D3 + D1D2'
endmodule
```