

Ασκήσεις Αρχιτεκτονικής Εξηγήσεις

Στο παρακάτω έγγραφο θα δείτε τον τρόπο επίλυσης βήμα-βήμα διαφορών ασκήσεων που μπορούν να πέσουν στις εξεταστικές .

• Απλοποιήσεις με χάρτες Karnaugh (Εισαγ. Διάλεξη)

- 1) Βάσει του πλήθους μεταβλητών επιλέγουμε τον κατάλληλο χάρτη
- 2) Εκφράζουμε την F ως άθροισμα ελαχιστόρων . Δηλαδή $\Sigma(1,2,3,...)$ και τοποθετούμε τις μονάδες στις θέσεις του χάρτη που αντιστοιχούν σε αυτούς τους ελαχιστόρους
- 3) Δημιουργούμε ομάδες από άσσους οι οποίες
 - a) το πλήθος των άσσων σε κάθε ομάδα είναι δύναμη του 2 (1,2,4,8,16,32)
 - b) κάθε ομάδα πρέπει να περιέχει ΜΕΓΙΣΤΟ πλήθος άσσων δηλαδή αν έχω την δυνατότητα να πάρω μια τετράδα δεν θα πάρω δυο δυάδες
 - c) πρέπει όλοι οι άσσοι να βρεθούν σε τουλάχιστον μια ομάδα
 - d) μπορούμε έναν άσσο να το συμπεριλάβουμε σε περισσότερες από 1 ομάδες
- 4) Κάθε ομάδα δημιουργεί έναν *ΑΠΛΟΠΟΙΗΜΕΝΟ* όρο γινομένου, Αν έχουμε π.χ. 2 ομάδες θα έχουμε 2 ορους γινομένου, οι οποίοι θα αθροιστούν. Δηλαδή N ομάδες $\rightarrow N$ όρους γινομένου όπου για π.χ:

BC					
A		00	01	11	10
0	1	0	1	3	1
1	1	4	5	7	1

K2: 000 και 001
A = σταθερά 0
B = σταθερά 0
C = 0 και 1
K2 = $A'B'$

K1: όρος που προκύπτει από την τετράδα
K1 = 000 και 100 και 010 και 110
A = 0 και 1 και 0 και 1 άρα δεν είναι σταθερό
B = 0 και 0 και 1 και 1 άρα δεν είναι σταθερό
C = σταθερά 0 άρα
K1 = C'

- 5) Αθροίζουμε τα γινομενα απο το d

$$F1 = K1 + K2$$

$$F1 = C' + A'B'$$

● Pipeline 1/2 (Διάλεξη 14)

Έστω η πράξη $A_i \times B_i + C_i$, $i=1 \dots t$

1) Φορτώνουμε τα A_i, B_i

δηλαδή : $R1 \leftarrow A_i$, $R2 \leftarrow B_i$

2) Πολλαπλασιάζουμε $A_i * B_i$ και φέρνουμε από την μνήμη το C_i

δηλαδή : $R3 \leftarrow A_i * B_i$ ($R3 \leftarrow R1 * R2$)

$R4 \leftarrow C_i$

3) Προσθέτουμε το γινόμενο του βήματα 2 ($R3$) με το C_i ($R4$)

$R5 \leftarrow R3 + R4$

4) Στην συνέχεια έχουμε 3 Segments

2 καταχωρητές $R1, R2$	-2 καταχωρητές $R3, R4$ -1 κυκλώμα πολλαπλασιαστή	-1 καταχωρητή RS -1 αθροιστής
------------------------	---	------------------------------------

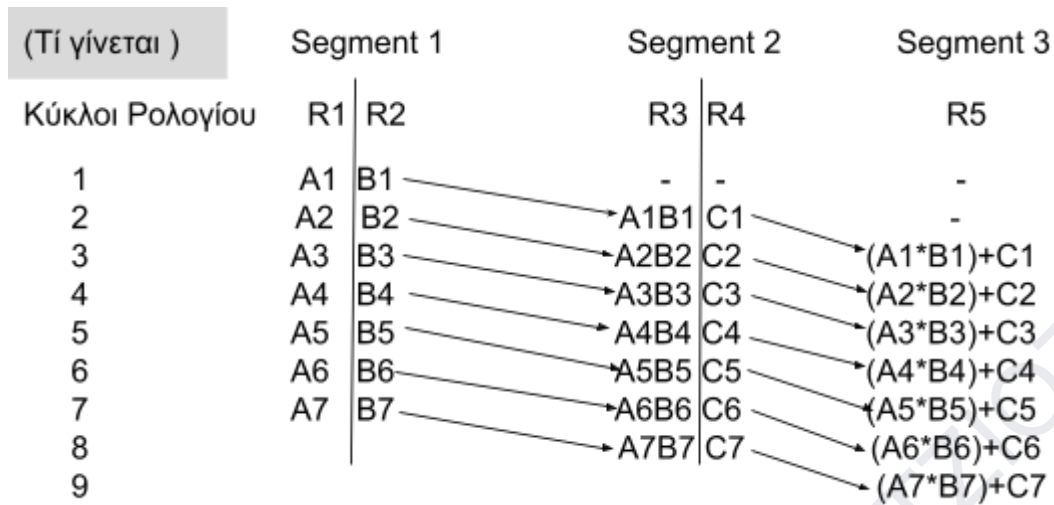
5) Εδώ ουσιαστικά βλέπουμε τα γινόμενα να πηγαίνουν από Segment σε Segment με την Μορφή που αναφέρεται στο **3)**

με το

- $R1 \leftarrow A_i$, $R2 \leftarrow B_i$, ουσιαστικά εδώ τα βάζουμε απλά στο Pipeline
- $R3 \leftarrow A_i * B_i$ ($R3 \leftarrow R1 * R2$) , $R4 \leftarrow C_i$, εδώ έχει γίνει η πρώτη πράξη και περνάει στο Segment 2
- $R5 \leftarrow R3 + R4$, τέλος μπαίνει στο Segment 3 με την τελευταία πράξη που δείξαμε στο **4)**

	Segment 1		Segment 2		Segment 3
Κύκλοι Ρολογίου	R1	R2	R3	R4	R5
1	A1	B1	-	-	-
2	A2	B2	A1B1	C1	-
3	A3	B3	A2B2	C2	(A1*B1)+C1
4	A4	B4	A3B3	C3	(A2*B2)+C2
5	A5	B5	A4B4	C4	(A3*B3)+C3
6	A6	B6	A5B5	C5	(A4*B4)+C4
7	A7	B7	A6B6	C6	(A5*B5)+C5
8			A7B7	C7	(A6*B6)+C6
9					(A7*B7)+C7

● Pipeline 2/2

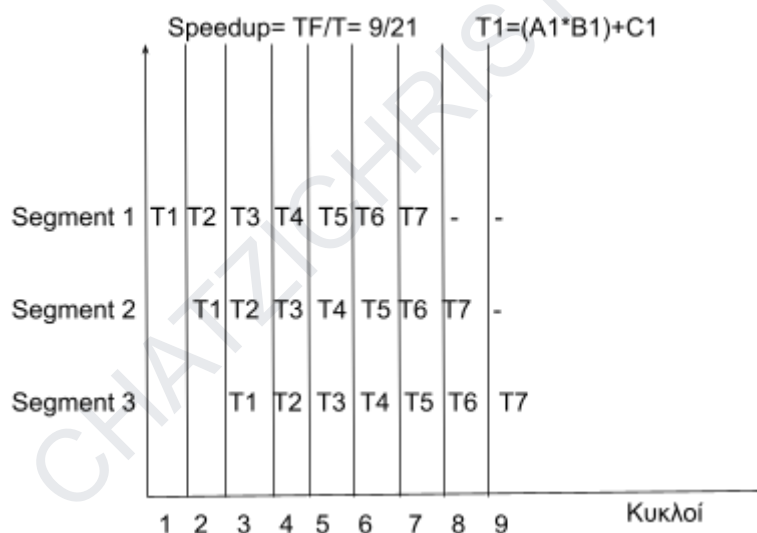


6) Εάν ζητηθεί και ο χρόνος :

$N=εργασίες=7$ επί $K=στάδια=3$ είναι ο χρόνος που απαιτείται χωρίς την ύπαρξη Pipeline (εδώ $T=7*3=21$)

Χρόνος μετά την χρήση Pipeline : $N+K-1=TF$ $7+3-1=9$

Το Speedup θα είναι $TF/T=9/21=0.42$ ή 42%



Εάν δεν υπήρχε το Pipeline τότε ο χρόνος θα ήταν $N*K$
 $7*3=21$

Οπού:

Ανάγνωση των A1,B1 από την T1 δηλαδή $T1 = (A1*B1) + C1$

μνήμη R1,R2 στο R3 δηλαδή $R3 \leftarrow R1R2$

Αποκωδικοποιητές (Διάλεξη 1)

Πιθανά ερωτήματα :

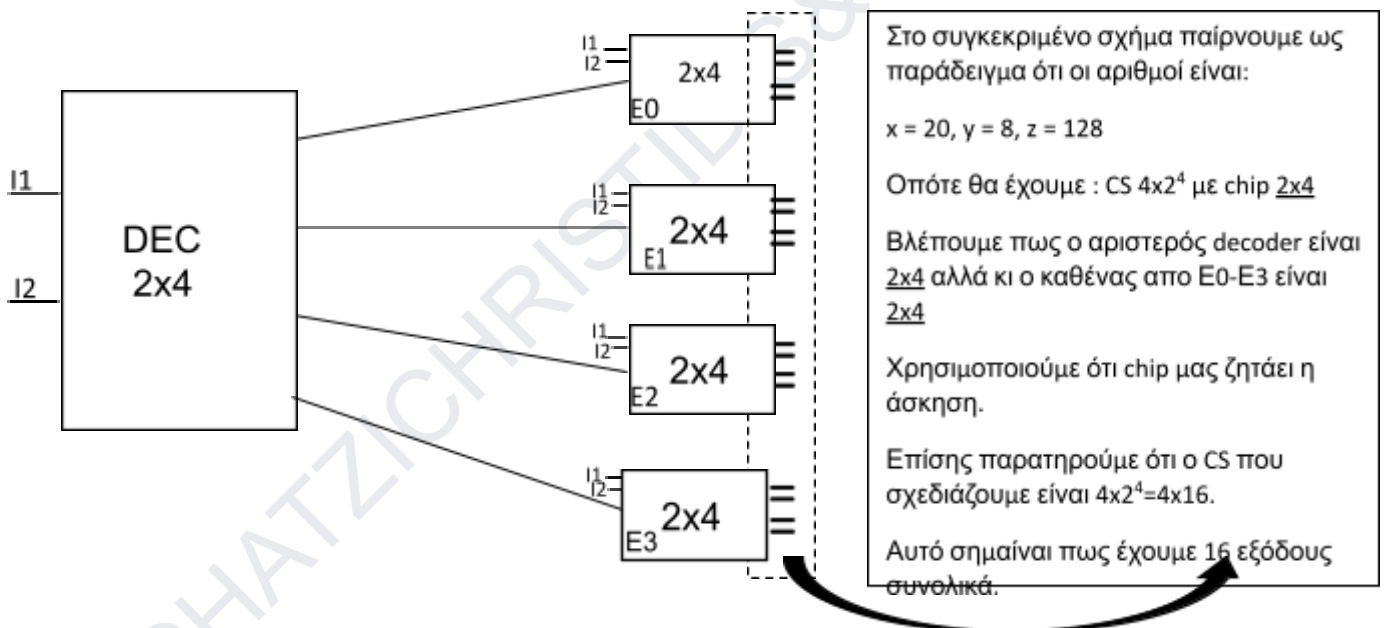
- Πόσα chips με 2^x λέξεις (μέγεθος λέξης y bytes) χρειάζονται για να κατασκευάσουμε μνήμη z bytes ; **
- Να σχεδιαστούν CS/WS με decoder $N \times M$
- Να δείξετε την αποκωδικοποίηση της λέξης με διεύθυνση

(Σημείωση: συνήθως ο ο αριθμός y , δηλαδή το μέγεθος λέξης θα είναι 8)

Λύση :

- Μετατρέπουμε τον αριθμό z σε δύναμη του 2.
- Μετατρέπουμε τον αριθμό y σε δύναμη του 2 (όπως είπαμε συνήθως 2^3).
- Μία μνήμη με z bytes , διαθέτει $z / y = a$ λέξεις των y bytes.
 - (Στην διαίρεση z / y οι αριθμοί είναι σε μορφή δύναμης του 2)
- $a / 2^x = 2^b$ chips με 2^x λέξεις των y byte απαιτούνται για αυτή τη μνήμη.
- $CS \rightarrow b \times 2^b$
- $WS \rightarrow x \times 2^x$

Σχεδίαση CS:



- Γενικά για την σχεδίαση του chip ξεκινάμε υπολογίζοντας πόσες εισόδους και πόσες εξόδους θα έχει βάσει της εκφώνησης. Συνεχίζουμε σχεδιάζοντας τον πρώτο αποκωδικοποιητή(αριστερά), έπειτα σχεδιάζουμε τις εισόδους($I1, I2, \dots, I_n$) του και τις εξόδους του($D1, D2, \dots, D_m$). Τέλος, κάνουμε το ίδιο για τα υπόλοιπα chip(δεξιά).
- Σημειώνεται πως στο παραπάνω παράδειγμα τα chip που χρησιμοποιούνται είναι 2×4 οπότε έχουν απο 2 εισόδους και 4 εξόδους το καθένα.

Σχεδίαση μονάδας ελέγχου 1/2 (Διάλεξη 9)

Έστω ότι το σήμα που θα ζητηθεί θα είναι το MARin

- 1) Ψάχνουμε σε ποιές εντολές βρίσκεται MAR. Αφού έχουμε MARin θα κοιτάξουμε μόνο τις εντολές στις οποίες είναι αριστερά από το βελάκι (\leftarrow) (μπορεί στην εκφώνηση να ζητηθούν μόνο μερικές από αυτές τις εντολές)
- 2) Βλέπουμε ότι το MARin το έχουμε στις εντολές

1)Ανάκληση

T0:MAR \leftarrow PC

F=1

2)STA

T3:MAR \leftarrow MDR[ADDRESS 1]

F=0 else G=1

3)AND

T3:MAR \leftarrow MDR[ADDRESS 1]

F=0 else G=1

4)OUT

T3:MAR \leftarrow MDR[ADDRESS 1]

F=0 else G=1

9) Κύκλο διακοπής

T1:MAR \leftarrow Z

T4:MAR \leftarrow Z

F=0 G=0

5)LDA

T3:MAR \leftarrow MDR[ADDRESS 1]

F=0 else G=1

6)ADD

T3:MAR \leftarrow MDR[ADDRESS 1]

F=0 else G=1

7)INC

T3:MAR \leftarrow MDR[ADDRESS 1]

F=0 else G=1

8)RET

T3:MAR \leftarrow SP

F=0 else G=1

10)JSR

T4:MAR \leftarrow Z(Address)

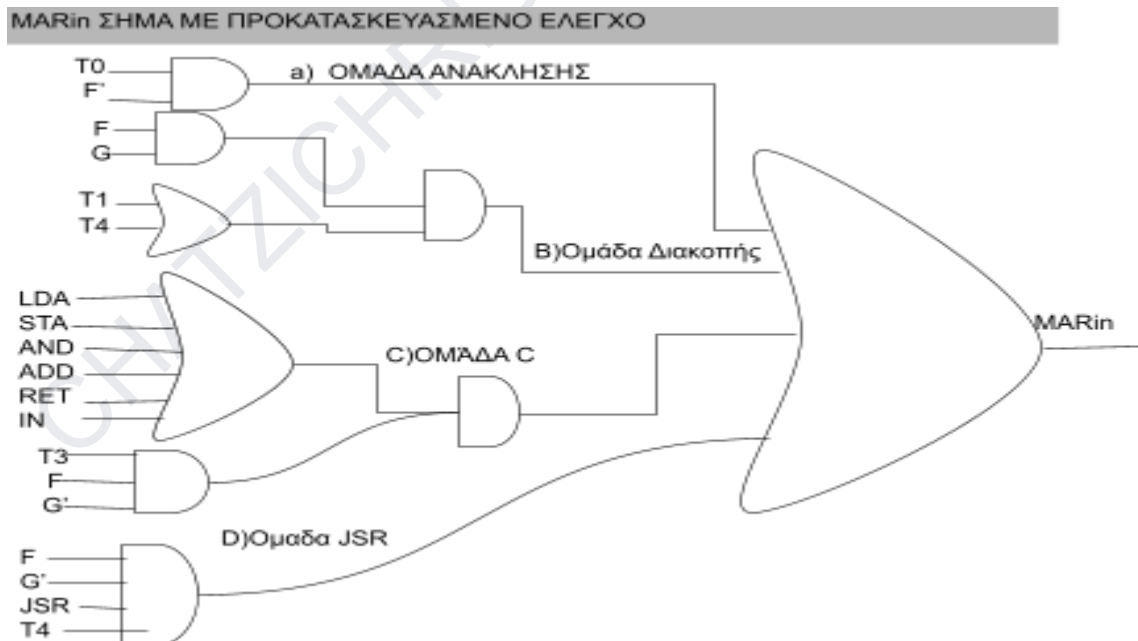
F=0 else G=1

- 3) Τα διαχωρίζουμε σε ομάδες αναλογά με τον χρόνο και τις τιμές τον F, G(κοιτάμε για τιμές ανακλ. και διακοπής τον πίνακα για τα F,G)
- 4) Οι ομάδες εδώ είναι 4
 - a) Ομάδα Ανάκλησης με T0 και F=0
 - b) Ομάδα διακοπής με T1,T4
 - c) Ομάδα των LDA,STA,AND,ADD,RET,INC με T3
F=1 και G=0
 - d) Ομάδα JSR με T4
- 5) Η εντολές έχουν F=1 και G=0 πέρα από την Ανάκληση που έχει F=0 δηλαδή F^και την διακοπή που έχει F G . Δηλαδή F=1 G=1.
Θα το καταλάβουμε καλύτερα από το πίνακάκι

Flip-flop F	Flip-flop G	Κατάσταση
0	0	Ανάκληση εντολής
0	1	Στάση (halt)
1	0	Εκτέλεση εντολής
1	1	Κύκλος διακοπής

Σχεδίαση μονάδας ελέγχου 2/2 (Διάλεξη 9)

- 6) Δημιουργούμε για κάθε ομάδα και 1 είσοδο
- 7) Οτιδήποτε είναι σταθερό στις ομάδες μπαίνει σε and αλλιώς θα μπει σε or
- 8) **A)** Στην Ομάδα ανάκλησης θα χρειαστεί μονάχα μια AND με τις 2 **σταθερές** τιμές που ξέρουμε T0 και την F^ αφού στην ανάκληση έχουμε F=0(ΚΟΙΤΑΩ ΠΙΝΑΚΑΚΙ)
- B)** Στην ομάδα διακοπής θα έχουμε μια Or με τις τιμές T1/T4 αφού **δεν είναι σταθερές** αλλά **αλλάζει** το T σε 1 και 4 και μια and με τις τιμές F=1 G=1 δηλαδή F και G αφού είναι **ετσι στο πινακάκι**
- C)** Στην ομάδα με τις πολλές εντολές βάζουμε όλες τις εντολές (**αφού αλλάζουν**) σε μια OR να ενώνονται με τις 3 σταθερές μας δηλαδή T3 και τα F=1 και G=0 ξανά απο πινακάκι αφού έχουμε εκτέλεση εντολής σε αυτην την ομάδα
- D)** Στην ομάδα JSR θα βάλουμε όλες τις σταθερές μας σε μια and και τίποτα άλλο αφού δεν έχουμε στοιχεία που αλλάζουν δηλαδή F=1 G=0 αφού είναι εκτέλεση εντολής (**ΚΟΙΤΑΩ ΠΙΝΑΚΑΚΙ**) και την T4 που είναι σταθερή (Η JSR είναι ομάδα μόνη της επειδή είναι η μοναδική εντολή εδώ που έχει T4)



ΜΕ ΤΗΝ ΊΔΙΑ ΛΟΓΙΚΉ ΛΥΝΕΙΣ ΚΑΙ ΚΑΘΕ ΑΛΛΟ ΣΗΜΑ ΕΛΕΓΧΟΥ

- Κρυφή μνήμη (Διαλεξη 11,12,13)

3 μορφές οργάνωσης:

Άμεση συσχέτιση: Κάθε Block αντιστοιχεί σε 1 γραμμή

Συσχέτιση Συνόλων: Κάθε Block αντιστοιχίζεται σε A σύνολο.

Άρα αντιστοιχίζεται σε K Γραμμές όπου K οι δρόμοι

Πλήρη συσχέτιση: Αντιστοίχιση σε όλες

Άμεση συσχέτιση

Δεδομένα :

1. Μέγεθος λέξης.
2. Μέγεθος RAM και block RAM.
3. Μέγεθος cache και γραμμής cache.

Πιθανά ζητούμενα :

1. Να αναλύσετε τη διεύθυνση της RAM.
2. Να δώσετε τις διευθύνσεις των λέξεων που σχετίζονται με την γραμμή της cache αν έχουμε τεχνική άμεσης συσχέτιση
3. Έστω η CPU ζητάει διαδοχικά τις διευθύνσεις , ,
 - a) Για κάθε διεύθυνση να βρείτε αν υπάρχει hit ή miss.
 - b) Αν κάθε προσπέλαση της RAM απαιτεί A χρονικές μονάδες και κάθε προσπέλαση της cache B χρονικές μονάδες, να βρείτε τον συνολικό χρόνο προσπέλασης των διευθύνσεων.

Λύση:

1. ΤΥΠΟΛΟΓΙΟ

- a) Μέγεθος RAM = 2^N bytes \Rightarrow Μέγεθος διεύθυνσης RAM = N bit
- b) Πλήθος block RAM = Μέγεθος RAM / Μέγεθος block
- c) Πλήθος γραμμών cache = Μέγεθος cache / Μέγεθος γραμμής
- d) Για να βρούμε το πλήθος των block :
 $\text{Μέγεθος RAM} / \text{Μέγεθος block}$
- e) Για να βρούμε το μέγεθος του tag :
 $\text{Πλήθος block} / \text{Πλήθος γραμμών cache}$
- f) Για να βρούμε το πλήθος δυαδικών ψηφίων του tag:
 $\text{Μέγεθος tag} = 2^x \Rightarrow x \text{ δυαδικά ψηφία.}$
- g) Για να βρούμε το πλήθος δυαδικών ψηφίων της γραμμής:
 $\text{Μέγεθος line} = 2^y \Rightarrow y \text{ δυαδικά ψηφία.}$
- h) Για να βρούμε το πλήθος δυαδικών ψηφίων του byte:
 $\text{Μέγεθος διεύθυνσης}_{(a)} - \text{Μέγεθος tag}_{(e)} - \text{Μέγεθος γραμμής}_{(g)} = z \text{ δυαδικά ψηφία}$
- i) Άρα η διεύθυνση θα έχει τη μορφή :

Tag	Γραμμή	Byte
x bits	y bits	z bits

2. Άμεση συσχέτιση :

Για να σχετίζονται οι διευθύνσεις των λέξεων με την γραμμή της cache, θα πρέπει το πεδίο γραμμής να δείχνει στο αντίστοιχο δυαδικό της γραμμής. Δηλαδή

Tag	Line	Byte
	

Σημείωση: όπου βλέπουμε είναι ο δοσμένος αριθμός από εκφώνηση.

Οι διευθύνσεις των ζητούμενων λέξεων θα είναι όλες όσες έχουν τον συγκεκριμένο δυαδικό αριθμό στη γραμμή, από όλα τα block της RAM.

3.

α)

1. Φτιάχνουμε ένα Tag Directory με μέγεθος = πλήθος γραμμών cache x μήκος tag.
2. Επειδή η μνήμη είναι άμεσης συσχέτισης, αν υπάρχει μέσα στη cache θα βρίσκεται μαζί με ολόκληρο το block στο οποίο ανήκει η γραμμή της διεύθυνσης.

Σημείωση: Κάθε χρονική στιγμή μπορεί να υπάρχει ένα block και αυτό καθορίζεται από την τιμή του tag της διεύθυνσης.

3. Έρχεται η πρώτη ζητούμενη διεύθυνση. Πρέπει να την αναζητήσουμε στη cache Διαβάζουμε την τιμή πεδίου Γραμμή

Πηγαίνουμε στην αντίστοιχη γραμμή του tag directory και διαβάζουμε το αποθηκευμένο tag

Αν είναι ίσο, τότε η ζητούμενη διεύθυνση μαζί με όλο το block υπάρχει στη cache => έχουμε hit. Αλλιώς έχουμε miss.

Συνεχίζουμε για όλες τις διευθύνσεις που ζητούνται ομοίως.

β)

1. $t = \text{hits} \times t_{\text{cache}} + \text{misses} \times t_{\text{RAM}}$
όπου t : ο συνολικός χρόνος προσπέλασης
 t_{cache} : χρόνος προσπέλασης cache
 t_{RAM} : χρόνος προσπέλασης RAM

Συσχέτιση συνόλων (K-way)

Ζητούμενα :

Ανάλυση διεύθυνσης.

Λύση :

Για να αναλύσουμε τη διεύθυνση εδώ πρέπει να βρούμε το πλήθος των συνόλων.
Κάθε σύνολο περιέχει K γραμμές.

Η μορφή της διεύθυνσης εδώ θα είναι :

Tag	Set	Byte

- a) Μέγεθος RAM = 2^N bytes => N bit διευθυνσιοδότηση
- b) Πλήθος block RAM = Μέγεθος RAM / Μέγεθος block
- c) Πλήθος συνόλων = Πλήθος γραμμών cache / K (Πλήθος δρόμων)
- d) Πλήθος block / Πλήθος συνόλων = Πλήθος block που αντιστοιχίζονται σε κάθε σύνολο.
- e) Κάνουμε το πλήθος block που αντιστοιχίζονται σε κάθε σύνολο δύναμη του 2.
Ο εκθέτης μας δίνει το μέγεθος του tag.
- f) Κάνουμε το πλήθος των συνόλων δύναμη του 2
Ο εκθέτης μας δίνει το μέγεθος του set.
- g) Κάνουμε το μέγεθος λέξης δύναμη του 2
Ο εκθέτης μας δίνει το μέγεθος του byte.

Πλήρης συσχέτιση 1/2

Το πλήθος bit του tag δείχνει πόσα block αντιστοιχίζονται σε κάθε γραμμή της cache

δηλ. $2^{\text{πλ bit cache}} = \text{block ανα γραμμή}$

Έχω τόσους συγκριτές όσες οι γραμμές τις cache

Πλεονέκτημα : Κάθε block πάει παντού, δε χρειάζεται συνεχείς αντικαταστάσεις

Μειονέκτημα : Αύξηση του hardware

Ζητούμενα :

1. Ανάλυση διεύθυνσης.
2. Εξήγηση του tag (τι δείχνει)
3. Hit/Miss με πολιτική αντικατάστασης LRU/FIFO

Λύση :

1. Η μορφή της διεύθυνσης εδώ θα είναι :

Tag	Byte

Το πεδίο γραμμής δε χρειάζεται, αφού κάθε block μπορεί να πάει παντού.

2. Το πλήθος bit του tag εδώ μας δείχνει πόσα block αντιστοιχίζονται σε κάθε γραμμή της cache (π.χ. 32byte cache = $2^5 \Rightarrow 5\text{bit tag} \Rightarrow 5 \text{ block μνήμης σε κάθε γραμμή}$

Byte βρίσκουμε απο τον τύπο : $2^{\text{byte}} = 2^{\text{μέγεθος λέξης} / 2 \text{ γραμμές}}$

3. Ελέγχουμε αν το tag των διευθύνσεων που μας ζητούνται υπάρχουν στο Tag Directory. Αν υπάρχει τότε λέμε ότι έχουμε X συγκρίσεις, όπου X ο αριθμός γραμμών της cache, άρα και X συγκριτές και Hit. Αν δεν υπάρχει τότε λέμε ότι έχουμε Miss.

Πλήρης συσχέτιση 2/2

ΠΑΡΑΔΕΙΓΜΑ:

RAM					Cache				
0	A	B	C	D	0	A	B	C	D
1	1	2	3	4	1	1	2	3	4
2	K	Λ	M	N	2	K	Λ	M	N
3	Z	5	7	12	3	Z	5	7	12
4	6	10	5	K	4	6	10	5	K
5	5	T	5	T	5	5	T	5	T
6	A	B	D	B	6	A	B	D	B
7	1	1	1	1	7	1	1	1	1
8	10	12	11	13					
9	X	X	X	X					
10	9	8	5	9					
.....									
.....									
31									

Σε μία χρονική στιγμή, τα Block της RAM 0-7 έχουν φορτωθεί αντίστοιχα στις γραμμές 0-7 της Cache. Για τις διευθύνσεις: 32,33,60 Νά εξετάσετε αν υπάρχει HIT η MISS .

Tag Directory(16 bit αρα 10000)

0-3	00000
4-7	00001
8-11	00010
12-15	00011
16-19	00100
20-23	00101
24-27	00110
28-31	00111

0	A	B	C	D
1	1	2	3	4
2				
3				
4				
5				
6				
7	1	1	1	1

- 1) Πλήρη συσχέτιση :τόσοι συγκριτές οσες οι γραμμές της Cache, δηλαδή 8
- 2) Επειδή $32/4=8$, η ζητούμενη διεύθυνση βρίσκεται στο Block 8
- 3) το Tag 01000 δεν υπάρχει στο Tag Directory. Αρα Miss

Βγάζουμε τη γραμμή που μπήκε πρώτη (αφού έχουμε FIFO)και τοποθετούμε το νέο Block



Το 33 όμως αφού είναι 01000 01 και το TAG Directory τώρα έχει το 01000 μέσα απο το 32 που μπήκε θα είναι hit

33=01000 01 αρα είναι hit αφού το tag directory πλέον έχει το 01000 μέσα

Το 60 όμως για παράδειγμα που αλλάζει και δεν έχει το 01000 άλλα έχει 01111 στο TAG DIRECTORY μπροστά θα είναι και αυτό ξανά miss

60=0111100 Miss αφού το Tag δεν υπάρχει στα byte
byte 60 και byte/4 = BLOCK αρα 60/4 = 15 tag directory



Για την LRU (least recently used) θα μας δίνεται λίστα για το ποιες έχουν χρησιμοποιηθεί λιγότερο από τις διευθύνσεις και θα τα πάρουμε αυτά με τη σειρά. Ο τρόπος παραμένει ο ίδιος

Flip-Flop και Clock (Διάλεξη 5)

Λειτουργία :

1. CLK: 0→1 θετική μετάβαση/ακμοπυροδότηση

Η τιμή της D έρχεται στις γραμμές εισόδου (απο MDR → διάυλο δεδομένων)

2. CLK=1(θετικό) : δεν αποθηκεύεται καμία αλλαγή
3. CLK=0(αρνητικό) : δεν γίνεται καμία αλλαγή

Αλλαγή της τιμής που αποθηκεύει η Flip-Flop μπορεί να γίνει σε έναν από τους δύο χρόνους:

A. Όταν CLK από 0 γίνεται 1 (τη στιγμή που αλλάζει)

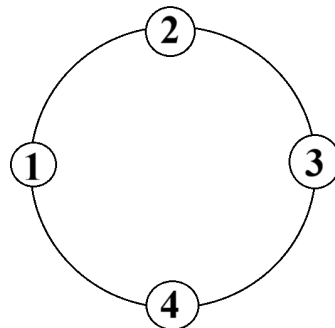
B. Όταν CLK από 1 γίνεται 0 (τη στιγμή που αλλάζει)

Η τιμή που αποθηκεύεται στη μνήμη είναι η τιμή της εξόδου Q

Τα Flip-Flop έχουν τη δυνατότητα παραγωγής και της συμπληρωματικής εξόδου Q'

Καταστάσεις CLK :

1. Παραμένει 0
2. Μεταβαίνει από 0 σε 1
3. Παραμένει 1
4. Μεταβαίνει από 1 σε 0



Το ρολόι CLK εναλλάσσεται σε τιμές 0 και 1

1. Έστω ότι αρχικά $CLK = 0$ και $Q = 0$.

$CLK = 0$ σημαίνει ότι οι πύλες 2 και 3 είναι ίσες με 1, δηλ. $S = R = 1$.

Αν $Q = 0 \Rightarrow Q' = 1$, άρα η έξοδος της πύλης 6 θα παραμείνει 1
άρα η έξοδος της πύλης 5 θα παραμείνει 0 } κύκλωμα αμετάβλητο

2. Το ρολόι μεταβαίνει από 0 σε 1.

Κατά τη στιγμή της μετάβασης ήταν $S = R = 1$.

Για να δούμε τι θα συμβεί εξετάζουμε την τιμή του D

A. Αν $D = 0$ η πύλη 4 θα έχει έξοδο 1

Ø Η (4) δίνει έξοδο 1

Ø Αυτό το 1 είναι είσοδος της (1) και μαζί με το $S = 1$ θα κάνουν την έξοδο της (1) ίση με το 0.

Ø Η (2) θα έχει μία είσοδο ίση με 0, άρα το S θα παραμείνει 1

Ø Όμως, το $CLK = 1$, $S = 1$, $(4) = 1$, άρα $(3) = 0$

Ø Άρα $Q' = 1$ και επειδή $Q' = S = 1$ θα είναι $Q = 0$

B. Αν $D = 1$, $S = R = 1$

Ø Η έξοδος της (4) θα είναι 0, γιατί $R = D = 1$.

Ø Επειδή $S = 1$ και $(4) = 0$, η (1) θα έχει έξοδο 1

Ø Άρα, οι είσοδοι της (2) είναι 1 (από την (1)) και 1 από το CLK

Ø $S = 0 \Rightarrow Q = 1$

Ø Επειδή $Q = 1$ και $R = 1 \Rightarrow Q' = 0$

Συμπέρασμα: Όταν έχουμε θετική μετάβαση ($CLK: 0 \rightarrow 1$), τότε $Q = D$.

3. Έστω $Q = 1$, $Q' = 0$ το CLK παραμένει 1, $R = 1$ και $S = 0$

Έρχεται ένα bit με τιμή 0 και τροφοδοτεί το D. Άρα, το D αλλάζει.

$D = 0 \Rightarrow (4) = 1$ Άρα, το S θα παραμείνει 0 \Rightarrow το R θα παραμείνει 1

Τελικά $S = 0$, $R = 1$ και $Q = 1$, $Q' = 0 \Rightarrow$ Το κύκλωμα μένει στην ίδια κατάσταση

Η τιμή του D δεν αποθηκεύεται όσο το $CLK = 1$

Το ρολόι κάποια στιγμή θα πάει στο 0 \Rightarrow σταθερό κύκλωμα

Το ρολόι κάποια στιγμή θα πάει στο 1 \Rightarrow η οποιαδήποτε τιμή του D θα αποθηκευτεί

