Περιεχόμενα

Σύντομη επισκόπηση του πρωτοκόλλου Dragon

4 Καταστάσεις/ States

1. **Exclusive-clean(E):** το μπλοκ της cache έχει μη κοινόχρηστα, μη τροποποιημένα δεδομένα.

2. **Shared-clean(Sc):** το μπλοκ της cache έχει κοινόχρηστα, μη τροποποιημένα δεδομένα.

3. **Shared-modified(Sm):** το μπλοκ της cache έχει κοινόχρηστα, τροποποιημένα δεδομένα.

4. **Modified(M):** το μπλοκ της cache έχει τροποποιημένα, μη διαμοιραζόμενα δεδομένα.

4 πιθανά Transactions του επεξεργαστή για κάθε block της cache.

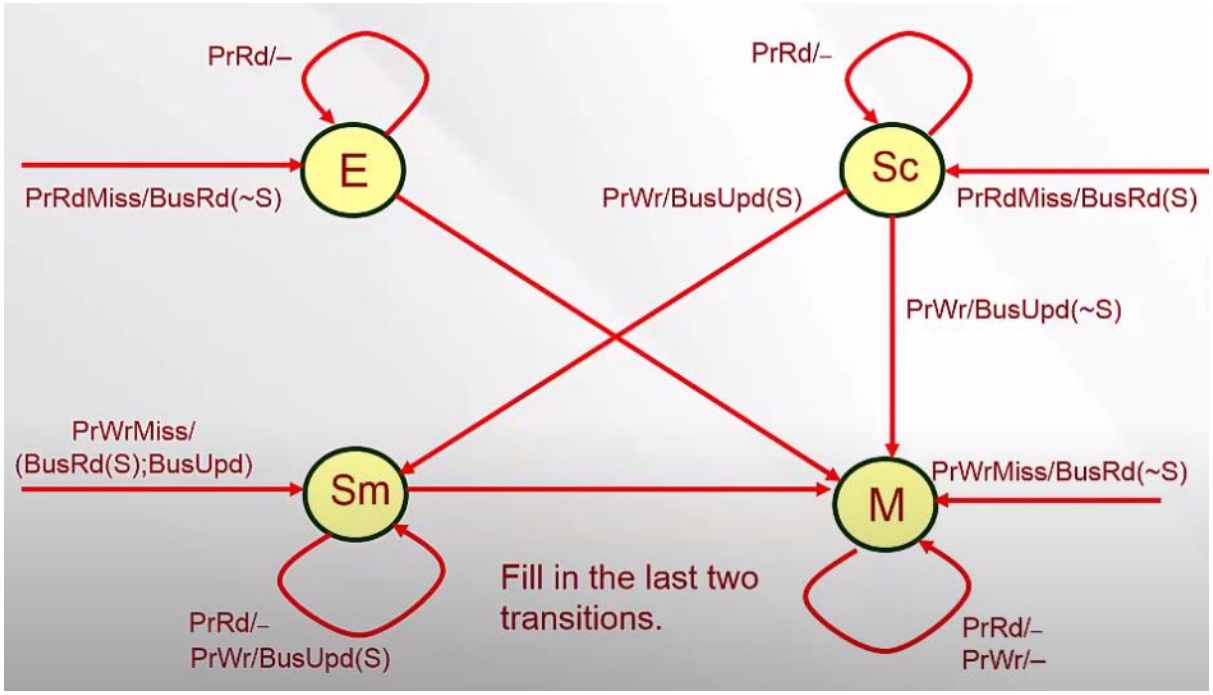
1. **Processor Read (PrRd):** ο επεξεργαστής ολοκληρώνει μια επιτυχή ανάγνωση σε ένα συγκεκριμένο μπλοκ της cache που έχει τοποθετηθεί στην cache του.
2. **Processor Write (PrWr):** Αυτό συμβαίνει όταν ο επεξεργαστής ολοκληρώσει μια επιτυχή εγγραφή σε ένα συγκεκριμένο μπλοκ της cache που έχει τοποθετηθεί στην cache του. Αυτό κάνει τον επεξεργαστή να είναι ο τελευταίος που ενημερώνει το μπλοκ της cache.
3. **Processor Read Miss (PrRdMiss):** Αυτό συμβαίνει όταν ο επεξεργαστής αποτυγχάνει να διαβάσει ένα μπλοκ cache από την κρυφή μνήμη του και πρέπει να ανακτήσει το μπλοκ είτε από τη μνήμη είτε από άλλη cache.
4. **Processor Write Miss (PrWrMiss):** Αυτό συμβαίνει όταν ο επεξεργαστής αποτυγχάνει να βρει την συγκεκριμένη διεύθυνση στη cache και γίνεται miss κατά την εγγραφή. Έτσι έχουμε αστοχία εγγραφής και επειδή η τεχνική της κρυφής μνήμης είναι write-allocate on miss πρέπει να ανακτήσει το μπλοκ πριν γίνει η εγγραφή.

BUS Transactions

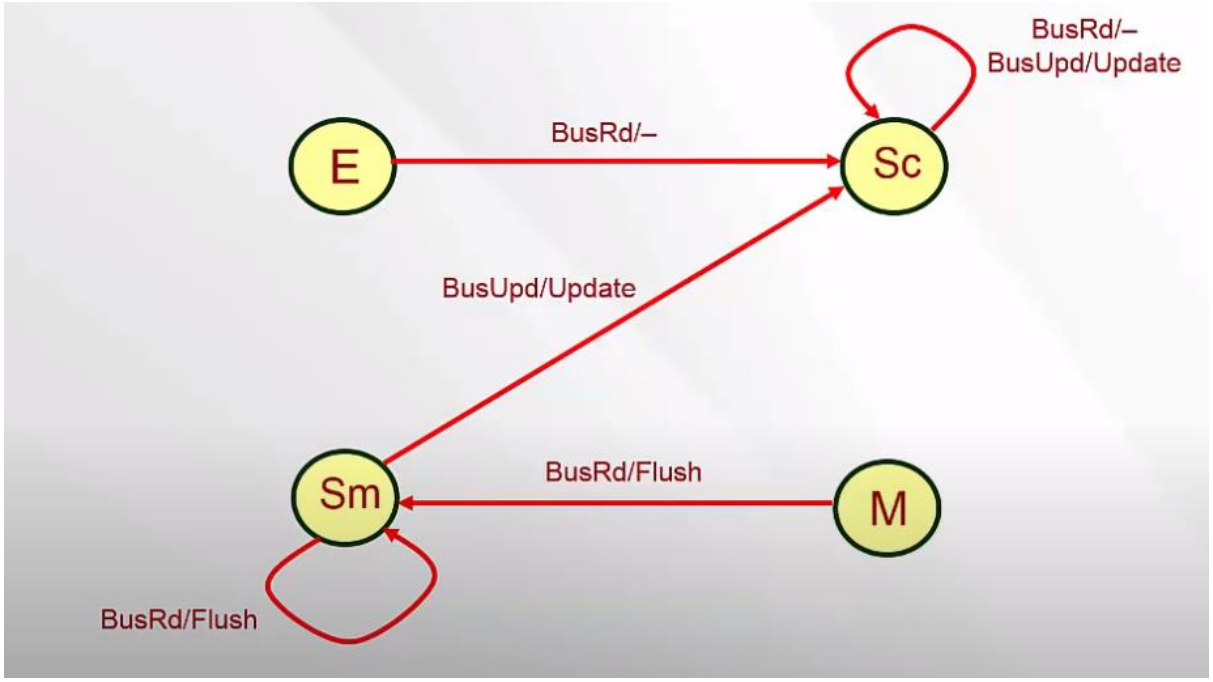
1. **Bus Read (BusRd)**: Αυτό συμβαίνει όταν ένας επεξεργαστής ζητά από το bus να ανακτήσει την τελευταία τιμή του μπλοκ που φέρει τη διεύθυνση προσπέλασης της RAM, είτε από την κύρια μνήμη είτε από την cache ενός άλλου επεξεργαστή.
2. **Flush**: Αυτό γίνεται για να αντικατοπτρίζει τις αλλαγές που γίνονται από τον επεξεργαστή στο μπλοκ που έχει γίνει cached στην κρυφή μνήμη και αφορά αντίγραφα δεδομένων που έχουν έρθει από την κύρια μνήμη.
3. **Bus Update (BusUpd)**: Αυτό συμβαίνει όταν ένας επεξεργαστής τροποποιεί ένα μπλοκ cache και άλλοι επεξεργαστές χρειάζονται ενημέρωση στα αντίστοιχα μπλοκ της cache. Αυτό είναι μοναδικό μόνο για τα πρωτόκολλα write update. Το BusUpd διαρκεί λιγότερο χρόνο σε σύγκριση με τη λειτουργία Flush, καθώς οι εγγραφές που πραγματοποιούνται στις cache είναι ταχύτερες από ό,τι στη μνήμη.

Επίσης απαιτείται μία **shared line** να υποδεικνύει εάν ένα συγκεκριμένο μπλοκ που φέρει τη διεύθυνση προσπέλασης της RAM cache είναι διαθέσιμο σε πολλαπλές cache.

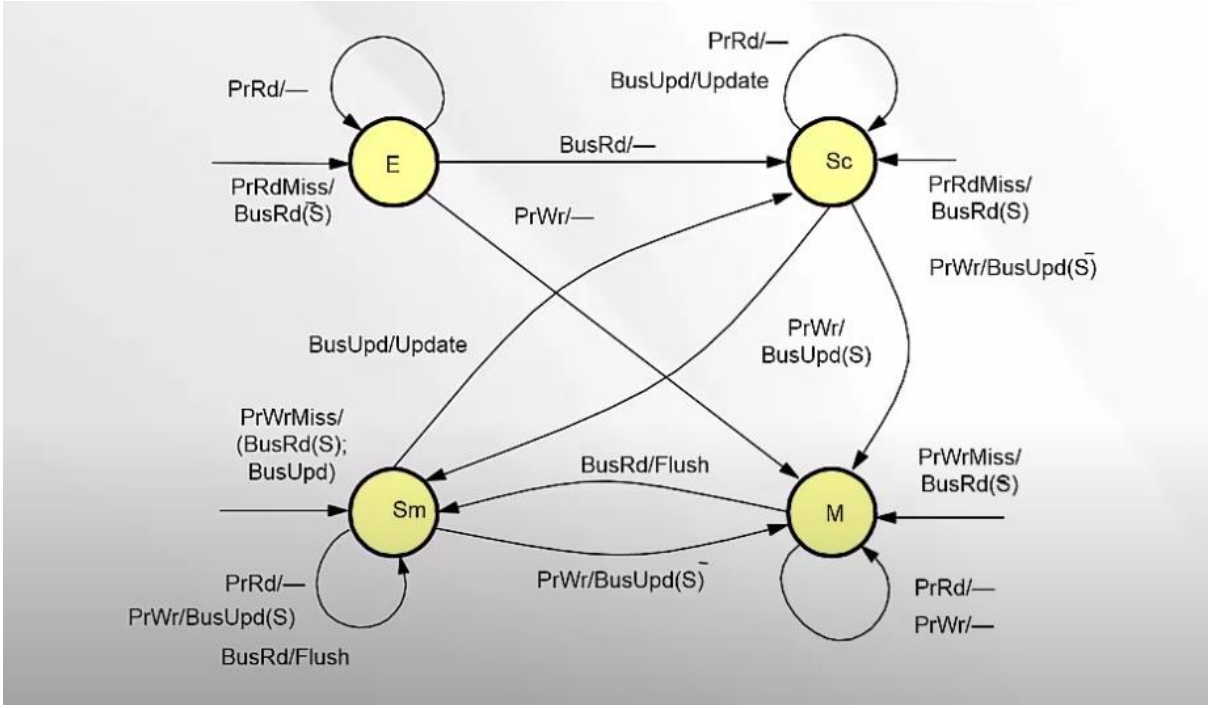
Διάγραμμα State-transition (processor initiated)



Διάγραμμα State-transition (BUS initiated)



Διάγραμμα State-transition (και Processor και BUS initiated)



Εκφώνηση Προβλήματος:

Ας θεωρήσουμε ένα σύστημα 2 πυρήνων, 2 επεξεργαστών που χρησιμοποιεί το πρωτόκολλο DRAGON (βλ. σχήμα παραπάνω), όπου κάθε επεξεργαστής χρησιμοποιεί 32 γραμμές κρυφής μνήμης(Cache line) των 8 bytes η κάθε μια. Αν έχουμε τις ακόλουθες εντολές των δύο επεξεργαστών, συμπληρώστε τον πίνακα:

	CPU 1 cache					CPU 2 Cache							
Instruction	Cache line (before execution)	Cache line (after execution)	Hit/ Miss	Old State	New State	Cache line (before execution)	Cache line (after execution)	Hit/ Miss	Old State	New State	Cache line Shared ?	Περιγραφή	BUS Signal
CPU 1 Read 0x07													
CPU 1 Write 0x03													
CPU 2 Read 0x08													
CPU 2 Write 0x0F													
CPU 2 Read 0x01													
CPU 1 Read 0x0C													
CPU 1 Write 0x05													
CPU 2 Write 0x0A													

Λύση Προβλήματος

Καθώς κάθε επεξεργαστής διαθέτει 32 cache line των 8 byte, κάθε φορά που φορτώνονται δεδομένα στην κρυφή μνήμη, φορτώνεται ένα μπλοκ 8 byte σε κάποιο cache line της cache.

Ως παράδειγμα:

Συνολικά υπάρχουν 256 bytes, από το 0 έως το 255, άρα 32 block όπου περιέχει 8 bytes το καθένα (256/8=32 blocks). Τα block είναι τα εξής:

1)0x00 ως 0x07 σε ένα block,
2)0x08 ως 0x0F σε ένα block,
3)0x10 ως 0x17 σε ένα block,
4)0x18 ως 0x1F σε ένα block,
5)0x20 ως 0x27 σε ένα block,
.
.
.
32)0x00F8 ως 0x00FF σε ένα block.

8 byte = 2^3 άρα έχουμε 3-bit offset.
32 cache lines = 2^5 άρα έχουμε 5-bit index.

Παρατηρώντας τις διευθύνσεις, μπορούμε να δούμε πως οι 0x01, 0x03, 0x05, 0x07 ανήκουν στο block 0x00 – 0x07 και οι 0x08, 0x0A, 0x0C, 0x0F στο 0x08 – 0x0F block. Παρακάτω υπάρχει αναλυτική επεξήγηση διαχωρισμού μέσω των index-offset σε γραμμές.

Για να μετατρέψουμε τις διευθύνσεις σε bytes, τις μετατρέπουμε σε δυαδικά bits και από εκεί ανά 4 δυαδικά bits τα μετατρέπουμε σε bytes.
Παρακάτω υπάρχουν οι πράξεις που έγιναν ώστε να καταλήξουμε στο παραπάνω συμπέρασμα:

1) 0x01: 0000 0001 = 0001 = (0 x 2^3) + (0 x 2^2) + (1 x 2^1) + (1 x 2^0) = $(1)_{10}$
2) 0x03: 0000 0011 = 0011 = (0 x 2^3) + (0 x 2^2) + (1 x 2^1) + (1 x 2^0) = $(3)_{10}$
3) 0x05: 0000 0101 = 0101 = (0 x 2^3) + (0 x 2^2) + (1 x 2^1) + (1 x 2^0) = $(5)_{10}$
4) 0x07: 0000 0111 = 0111 = (0 x 2^3) + (1 x 2^2) + (1 x 2^1) + (1 x 2^0) = $(7)_{10}$

   Το πρώτο block έχει διευθύνσεις από 0 έως 7 (οπότε παρατηρούμε πως αυτές οι 4 διευθύνσεις βρίσκονται στο ίδιο block 0x00 – 0x07).

5) 0x08: 0000 1000 = 1000 = (1 x 2^3) + (0 x 2^2) + (0 x 2^1) + (0 x 2^0) = $(8)_{10}$
6) 0x0A: 0000 1010 = 1010 = (1 x 2^3) + (0 x 2^2) + (1 x 2^1) + (0 x 2^0) = $(10)_{10}$
7) 0x0C: 0000 1100 = 1100 = (1 x 2^3) + (1 x 2^2) + (0 x 2^1) + (0 x 2^0) = $(12)_{10}$
8) 0x0F: 0000 1111 = 1111 = (1 x 2^3) + (1 x 2^2) + (1 x 2^1) + (1 x 2^0) = $(15)_{10}$

Το block που ανήκουν οι παραπάνω διευθύνσεις είναι το 0x08 – 0x0F, δηλαδή το 2^{ο} στην σειρά των block.

ADDRESS	**TAG**	**INDEX(5)**					**OFFSET(3)**		
0x01	0	0	0	0	0	0	0	0	1
0x03	0	0	0	0	0	0	0	1	1
0x05	0	0	0	0	0	0	1	0	1
0x07	0	0	0	0	0	0	1	1	1
0x08	0	0	0	0	0	1	0	0	0
0x0A	0	0	0	0	0	1	0	1	0
0x0C	0	0	0	0	0	1	1	0	0
0x0F	0	0	0	0	0	1	1	1	1

- Παρατηρώντας τον παραπάνω πίνακα, βλέπουμε πως οι πρώτες 4 διευθύνσεις έχουν index 00000. Αν το χωρίσουμε ανά 4 έχουμε: 0 0000, δηλαδή 0, άρα το μπλοκ 0x00-0x07 που φέρει τις διευθύνσεις των πρώτων τεσσάρων ανήκει στην γραμμή 0h.

  Επομένως, οι 4 τελευταίες διευθύνσεις του παραπάνω πίνακα εφόσον έχουν index 00001 και το χωρίσουμε ανά 4 θα έχουμε: 0 0001, δηλαδή 1, άρα το μπλοκ 0x08-0x0F που φέρει τις διευθύνσεις των τελευταίων τεσσάρων ανήκει στην γραμμή 1h.

- Κάπου εδώ αξίζει να σημειωθεί πως στις συγκεκριμένες διευθύνσεις έχουμε tag 0 όπως μπορούμε να δούμε από τον πίνακα, όμως θα μπορούσαν να έχουν και άλλες διευθύνσεις διαφορετικό tag και να άνηκαν στα συγκεκριμένα index.

Μερικοί γενικοί κανόνες σχετικά με το πρωτόκολλο Dragon που πρέπει να ακολουθήσουμε:

1. Κάθε φορά που ένα δεδομένο είναι καθαρό (clean/ αμετάβλητο), μεταβαίνει είτε στην κατάσταση E είτε στην κατάσταση Sc.

2. Κάθε φορά που ένα δεδομένο τροποποιείται (dirty), μεταβαίνει είτε στην κατάσταση M είτε στην κατάσταση Sm.

3. Ένα κοινόχρηστο δεδομένο πηγαίνει πάντα είτε στην κατάσταση Sc είτε στην κατάσταση Sm.

4. Ένα μη κοινόχρηστο δεδομένο πηγαίνει πάντα στην κατάσταση E ή M.

5. Σε οποιοδήποτε σημείο, μπορεί να υπάρχουν πολλοί επεξεργαστές στην κατάσταση Sc.

6. Αλλά σε οποιοδήποτε σημείο, το πολύ ένας επεξεργαστής μπορεί να βρίσκεται στην κατάσταση Sm.

7. Από την κατάσταση Sc, τα δεδομένα μπορούν να απορριφθούν (drop), αλλά από την κατάσταση Sm, πρέπει να γίνουν flush στην κύρια μνήμη, πριν μεταβούν σε οποιαδήποτε άλλη κατάσταση.

Δεδομένων όλων των παραπάνω προχωράμε στην επίλυση του πίνακα.

Δεδομένων των παραπάνω, λύνουμε το πρόβλημα ως εξής:

1. Αρχικά, ας θεωρήσουμε ότι και οι δύο κρυφές μνήμες είναι άδειες και χωρίς κατάσταση λόγο αποχής bit.
2. **CPU1 Read 0x07**: Η κρυφή μνήμη της CPU1 δεν έχει τα δεδομένα του block όπου ανήκει η διεύθυνση 0x07 δηλαδή το 0x00 – 0x07, οπότε ενεργοποιείται ένα bus transaction (BusRd). Τα δεδομένα αυτού του block φορτώνονται στην κρυφή μνήμη διότι έρχεται όλο το cache line και μεταβαίνει στην κατάσταση E, καθώς το μπλοκ κρυφής μνήμης δεν μοιράζεται με το μπλοκ κρυφής μνήμης άλλου επεξεργαστή.
3. **CPU1 Write 0x03**: Η CPU1 έχει τα δεδομένα του block όπου ανήκει η 0x03 διεύθυνση από την προηγούμενη εντολή. Επομένως, πρόκειται για hit(Write Hit) και, ως εκ τούτου, δεν ενεργοποιείται καμία BUS transaction. Πραγματοποιείται μόνο PrWr, δηλαδή εγγραφή στην ίδια την κρυφή μνήμη.
4. **CPU2 Read 0x08**: Η κρυφή μνήμη CPU2 δεν έχει τα δεδομένα του block όπου ανήκει η διεύθυνση 0x08 δηλαδή του 0x08 – 0x0F, οπότε ενεργοποιείται ένα bus transaction(BusRd). Τα δεδομένα αυτού το block φορτώνονται στην κρυφή μνήμη διότι έρχεται όλο το cache line και μεταβαίνει στην κατάσταση E, καθώς το μπλοκ κρυφής μνήμης δεν είναι shared με το μπλοκ κρυφής μνήμης άλλου επεξεργαστή.
5. **CPU2 Write 0x0F**: Η CPU2 έχει τα δεδομένα του block όπου ανήκει η 0x0F διεύθυνση από την προηγούμενη εντολή. Επομένως, πρόκειται για hit και, ως εκ τούτου, δεν ενεργοποιείται καμία BUS transaction. Πραγματοποιείται μόνο PrWr, δηλαδή εγγραφή στην ίδια την κρυφή μνήμη.
6. **CPU2 Read 0x01**: Τώρα, εδώ θα πρέπει να καταλάβουμε, σε αυτό το στάδιο, ότι τόσο η κρυφή μνήμη CPU1, όσο και η κρυφή μνήμη CPU2 έχουν τροποποιημένα, μη κοινόχρηστα δεδομένα. Δηλαδή η CPU1 έχει δεδομένα του block 0x00 – 0x07 σε κάποιο cache line της και η CPU2 έχει δεδομένα του block 0x08 – 0x0F αντίστοιχα. Και τώρα η CPU2 απαιτεί δεδομένα δηλαδή την διεύθυνση 0x01 όπου ανήκει στο block 0x00 – 0x07, το οποίο ανήκει σε κάποιο cache line της κρυφής μνήμης CPU1. Έτσι, η CPU2 θα κάνει flush πρώτα τα δεδομένα του block όπου βρίσκονται στο cache line της στην κύρια μνήμη μεταβαίνοντας στην κατάσταση Sm και στη συνέχεια θα μεταβεί στην κατάσταση Sc (ο λόγος είναι ότι πρέπει να διαβάσει το μπλοκ διευθύνσεων 0x01 δηλαδή το 0x00 – 0x07 και αυτό είναι κοινόχρηστο). Και η CPU1, θα μετακινηθεί στην κατάσταση Sm, καθώς έχει την αποκλειστική πρόσβαση του block 0x00 – 0x07 δεδομένων. Στη συνέχεια, η CPU2 διαβάζει την διεύθυνση 0x01 από το block 0x00 – 0x07 της κρυφής μνήμης της CPU1.
7. **CPU1 Read 0x0C**: Η CPU 1 ήταν νωρίτερα στην κατάσταση Sm, θα γράψει το μπλοκ δεδομένων 0x00 - 0x07 στην κύρια μνήμη. Και καθώς πρέπει να διαβάσει ένα μη κοινόχρηστο, καθαρό δεδομένο, μεταβαίνει στην κατάσταση E.

8. **CPU1 Write 0x05**: Τώρα, και πάλι, η CPU1 πρέπει να γράψει ένα δεδομένο, το οποίο είναι κοινόχρηστο. Έτσι, θα μεταφερθεί στην κατάσταση Sm και θα γράψει τα δεδομένα.
9. **CPU2 Write 0x0A**: Η CPU2 βρίσκεται αυτή τη στιγμή στην κατάσταση Sc και κατέχει κάποια δεδομένα. Από την κατάσταση Sc, θα κάνει drop τα δεδομένα και θα μεταβεί στην κατάσταση M αφού τροποποιήσει τα δεδομένα.

Επομένως ο πίνακας θα γίνει όπως φαίνεται στο συνοδευόμενο excel

Αναφορές

1. https://www.youtube.com/watch?v=KAsqghrrlpM&t=506s
2. https://stackoverflow.com/questions/21126034/msi-mesi-how-can-we-get-read-miss-in-shared-state
3. https://en.wikipedia.org/wiki/Dragon_protocol
4. https://handwiki.org/wiki/Dragon_protocol