Προηγμένα Θέματα Αρχιτεκτονικής Υπολογιστών
2η άσκηση

Νικόλαος Παγώνας, el18175

1 Εισαγωγή

Στα πλαίσια αυτής της άσκησης θα μελετήσουμε διαφορετικά συστήματα πρόβλεψης εντολών άλματος (branch predictors) με χρήση του εργαλείου `PIN`.

2 PINTOOL

Κεντρικά σε αυτή την άσκηση είναι τα αρχεία:

- `cslab_branch_stats.cpp` για την εξαγωγή στατιστικών σχετικά με τις εντολές άλματος που εκτελούνται από την εκάστοτε εφαρμογή,
- `cslab_branch.cpp` για την αξιολόγηση τεχνικών πρόβλεψης άλματος,
- `branch_predictor.h`, στο οποίο ορίζονται οι διαφορετικοί branch predictors που θα χρησιμοποιηθούν. Σε αυτό το αρχείο μπορούμε να προσθέσουμε δικούς μας branch predictors, που θα κληρονομούν από την `BranchPredictor`, και ως εκ τούτου θα υλοποιούν τις μεθόδους:
  - `predict()`, που δέχεται ως ορίσματα το `PC` (Program Counter) της εντολής και τη διεύθυνση προορισμού, και προβλέπει αν το άλμα θα εκτελεστεί (Taken) ή όχι (Not Taken),
  - `update()`, που αποθηκεύει τις πληροφορίες που απαιτούνται για μελλοντικές προβλέψεις,
  - `getName()`, για την εκτύπωση των αποτελεσμάτων στο αρχείο εξόδου του pintool.

3 Μετροπρογράμματα

Σε αυτή την άσκηση θα χρησιμοποιήσουμε τα `SPEC_CPU2006` benchmarks. Συγκεκριμένα θα χρησιμοποιηθούν τα:

1. 403.gcc
2. 429.mcf
3. 434.zeusmp
4. 436.cactusADM

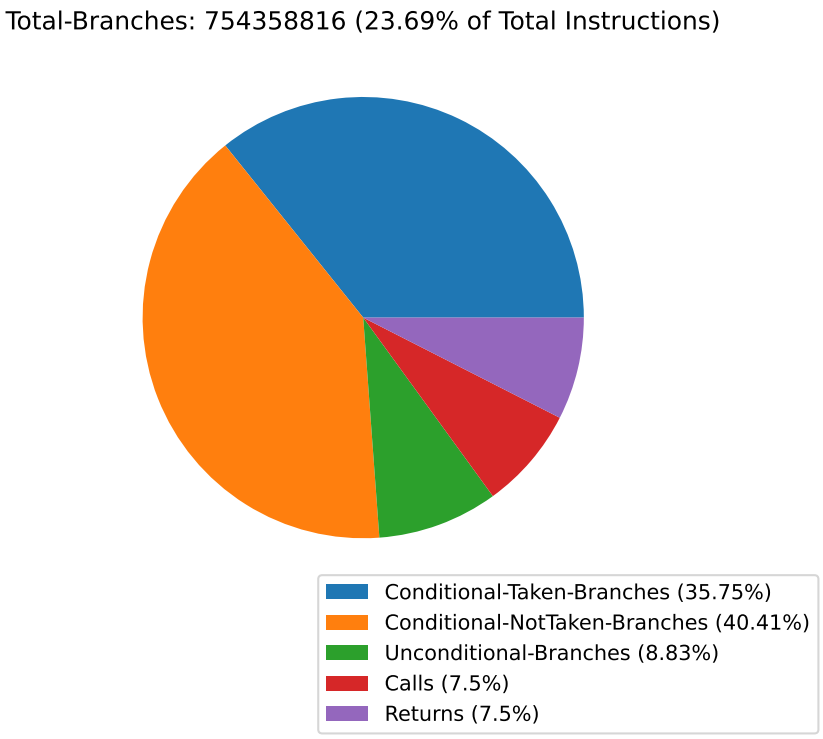
5. 445.gobmk
6. 450.soplex
7. 456.hmmer
8. 458.sjeng
9. 459.GemsFDTD
10. 462.libquantum
11. 470.lbm
12. 471.omnetpp
13. 473.astar
14. 483.xalancbmk

4 Πειραματική αξιολόγηση

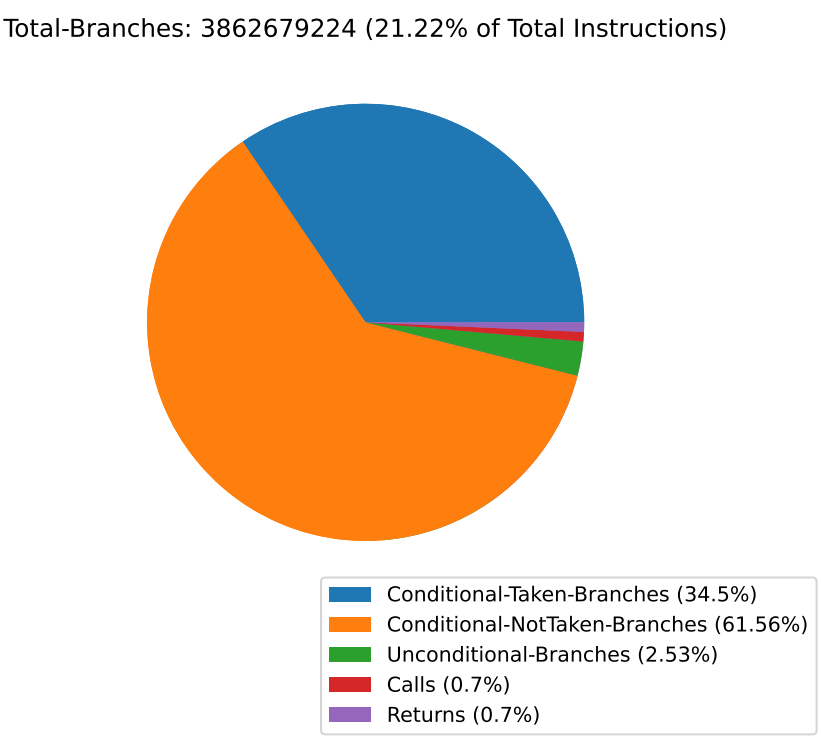
4.1 Μελέτη εντολών άλματος

Εδώ ο στόχος είναι η συλλογή στατιστικών για τις εντολές άλματος που εκτελούνται από τα benchmarks, οπότε χρησιμοποιείται το `cslab_branch_stats.cpp`. Για κάθε benchmark παρουσιάζουμε τον αριθμό των εντολών άλματος που εκτελέστηκαν, καθώς και το ποσοστό ανά κατηγορία:

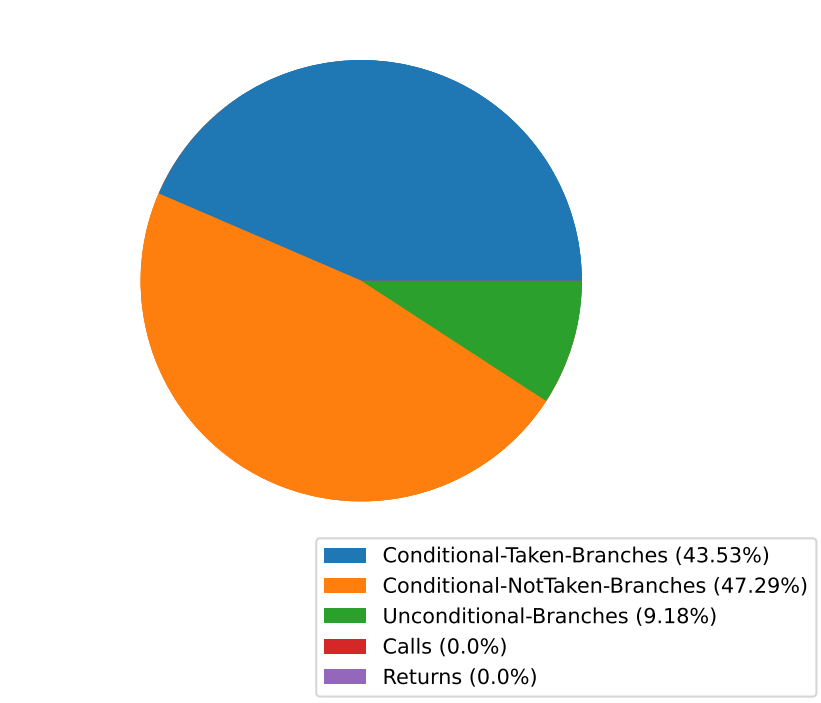
4.1.1 403.gcc



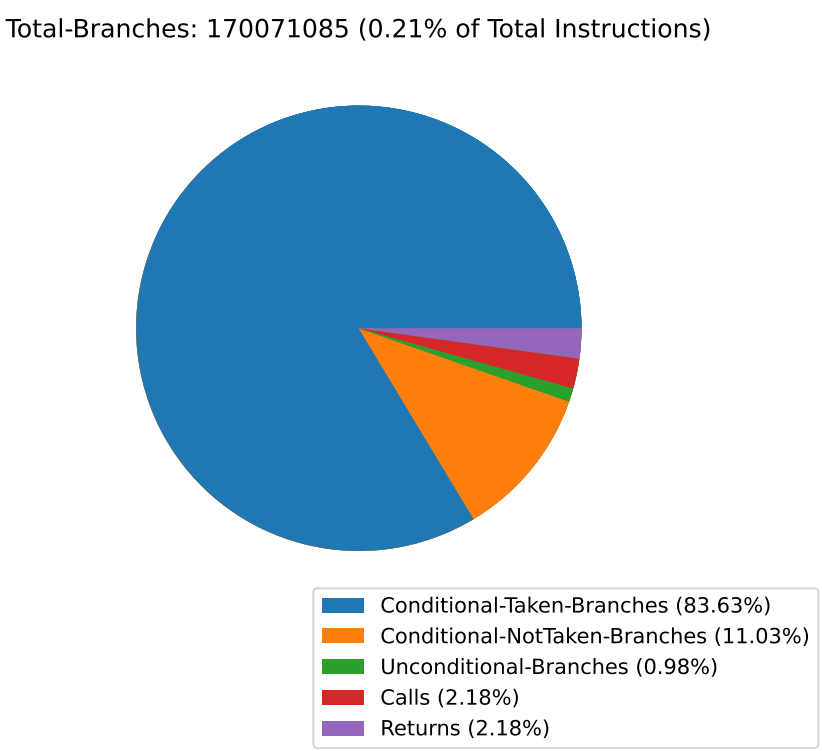
4.1.2 429.mcf



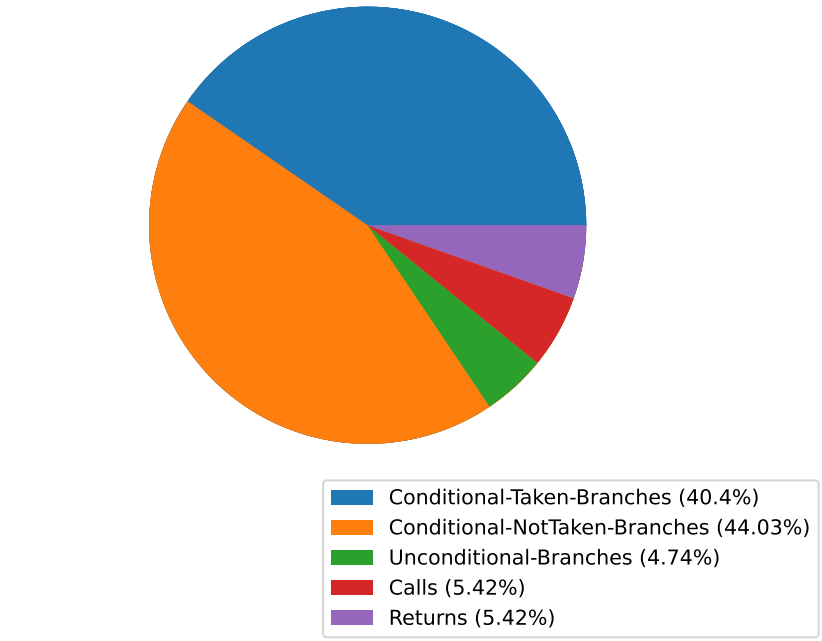
4.1.3 434.zeusmp



4.1.4 436.cactusADM

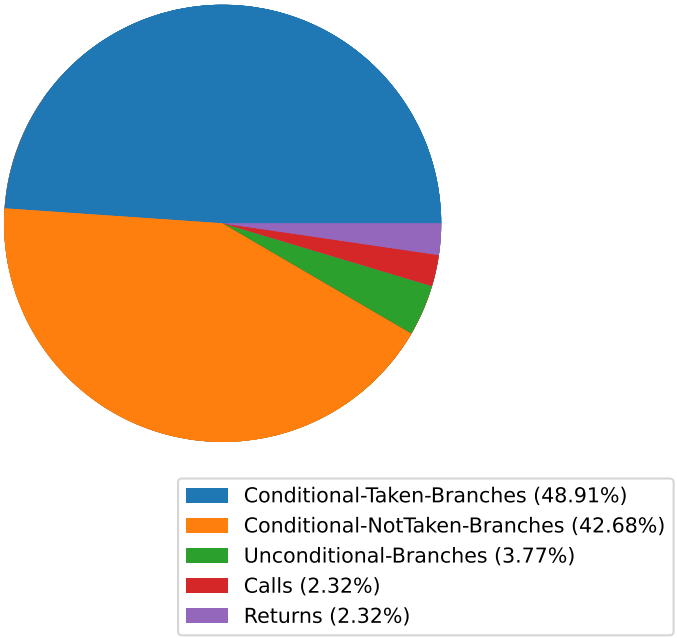


4.1.5 445.gobmk



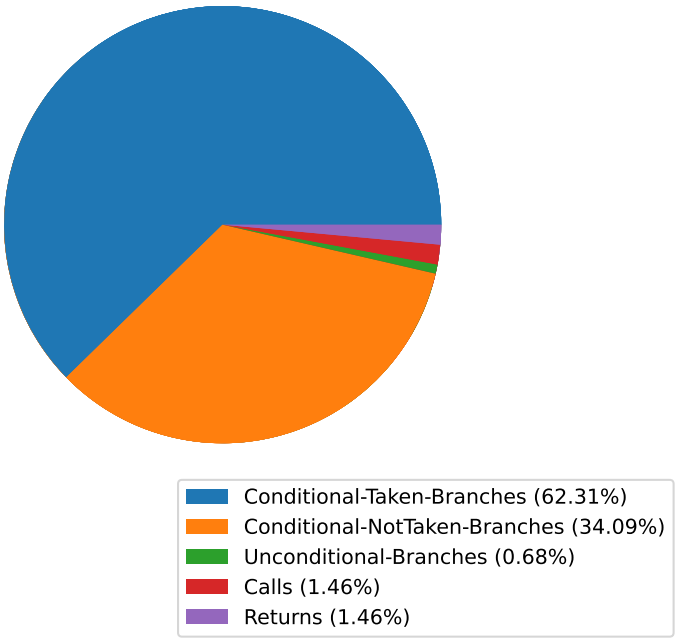
4.1.6 450.soplex

Total-Branches: 1824173459 (20.14% of Total Instructions)

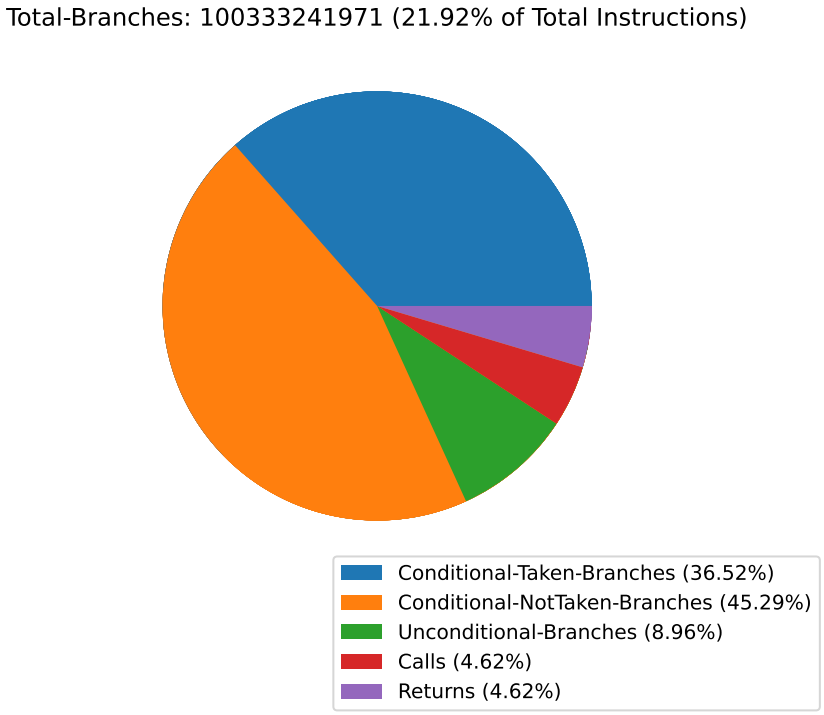


4.1.7 456.hmmer

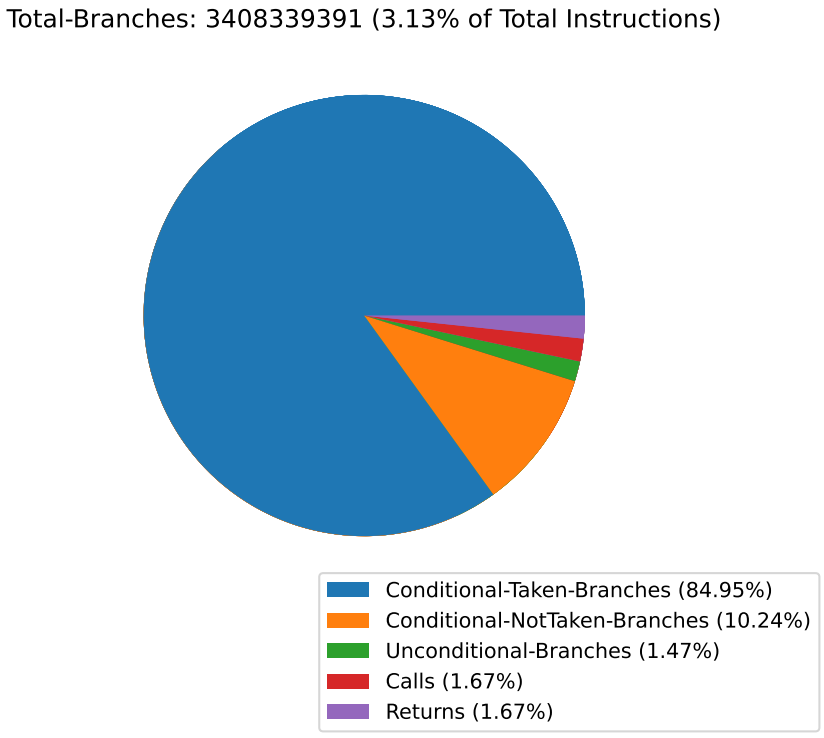
Total-Branches: 13914143675 (5.18% of Total Instructions)



4.1.8 458.sjeng

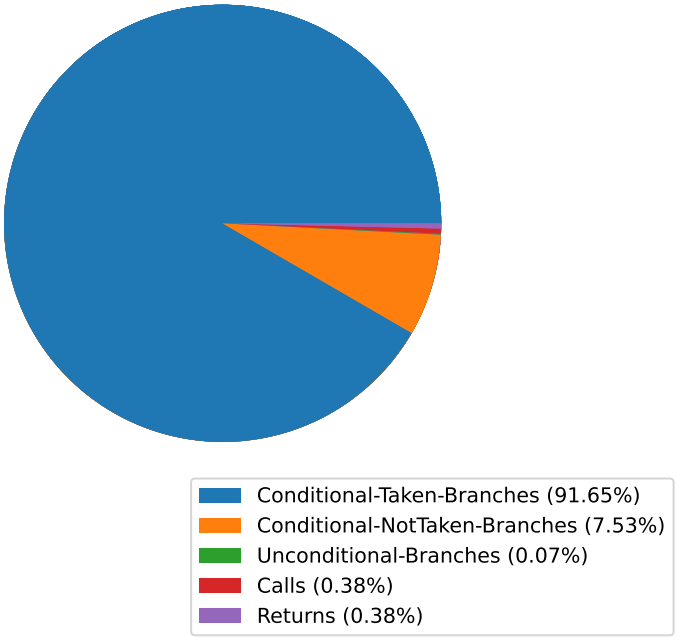


4.1.9 459.GemsFDTD



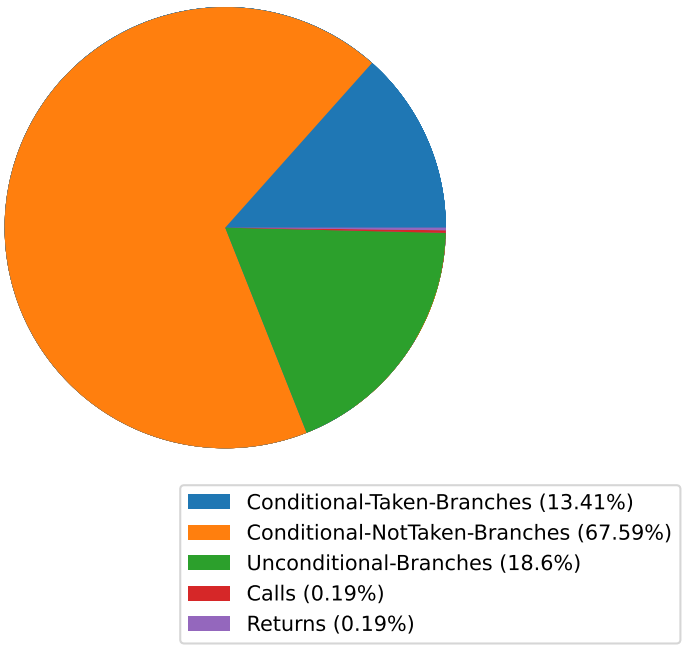
4.1.10 462.libquantum

Total-Branches: 2158568097 (23.99% of Total Instructions)

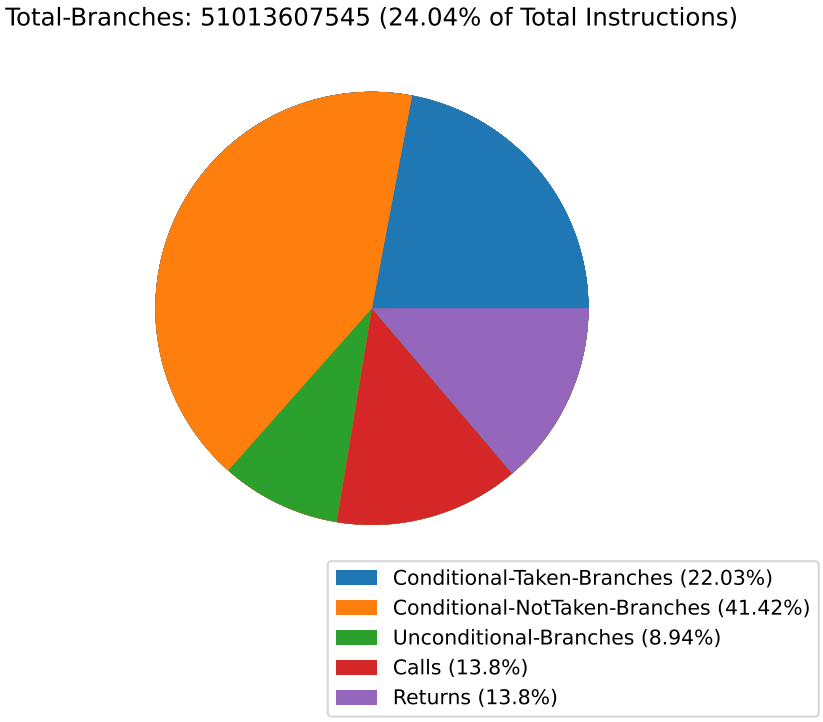


4.1.11 470.lbm

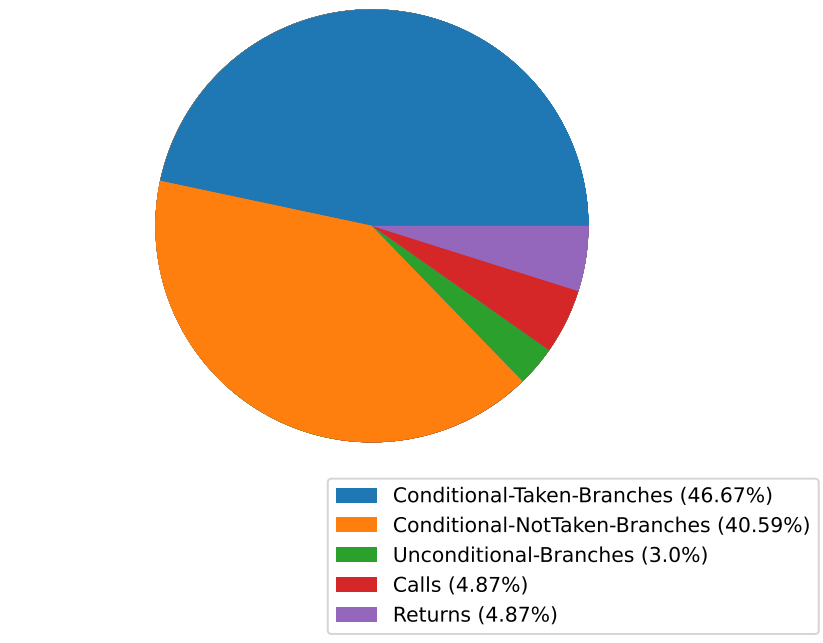
Total-Branches: 1354362060 (1.47% of Total Instructions)



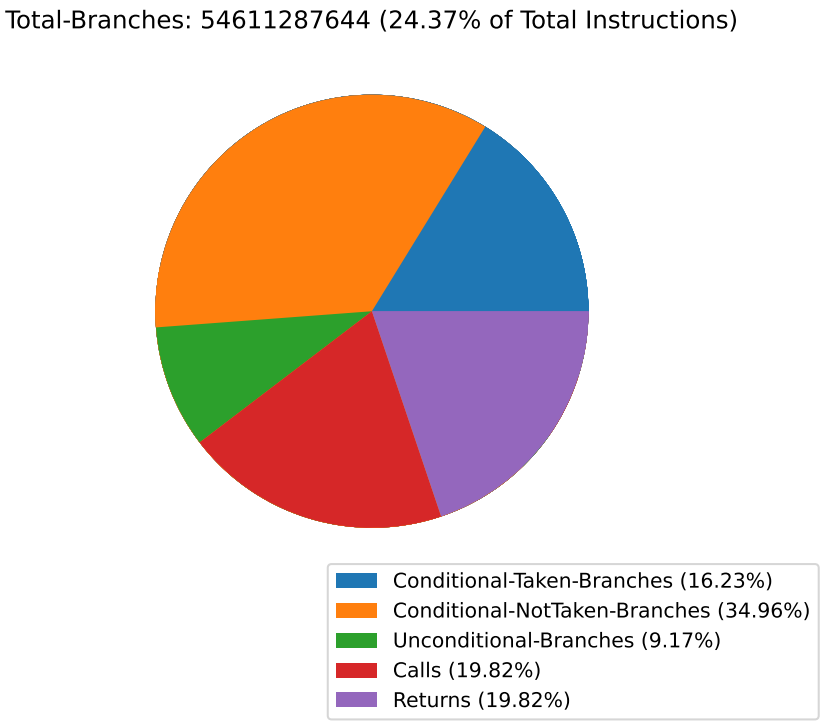
4.1.12 471.omnetpp



4.1.13 473.astar



4.1.14 483.xalancbmk



4.1.15 Παρατηρήσεις

Παρατηρούμε ότι οι εντολές άλματος αποτελούν ένα υπολογίσιμο ποσοστό των συνολικών εντολών, ειδικά σε benchmarks όπως:

`403.gcc`
`429.mcf`
`445.gobmk`
`450.soplex`
`458.sjeng`
`462.libquantum`
`471.omnetpp`
`483.xalancbmk`

όπου το ποσοστό ανέρχεται σε 20%-30%. Φυσικά υπάρχουν και εξαιρέσεις, όπως τα:

`436.cactusADM`
`459.GemsFDTD`
`470.lbm`

όπου οι εντολές άλματος απαντώνται σε ποσοστά 0%-3%.

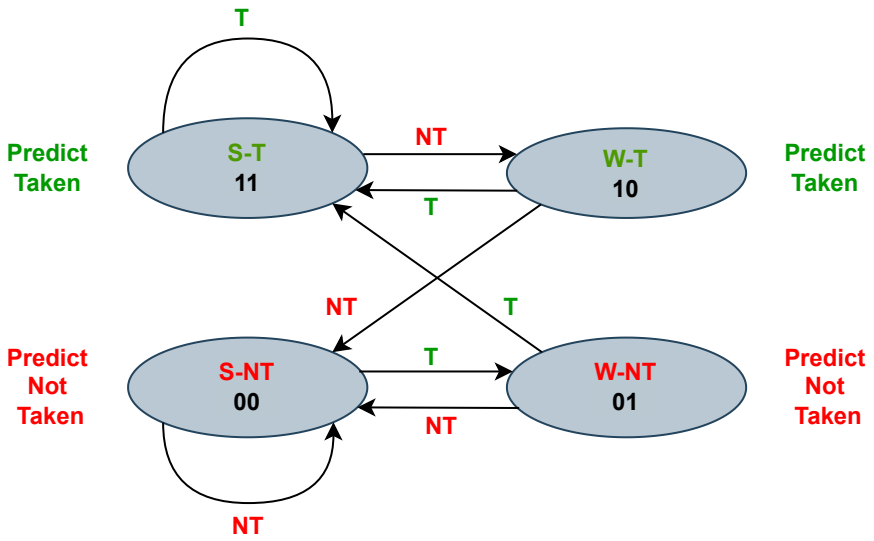
Ως επί το πλείστον πρόκειται για Conditional-Taken-Branches/Conditional-NotTaken-Branches. Τα υπόλοιπα branches εμφανίζονται σε ποσοστό ≈5-20%. Να σημειωθεί ότι ορισμένα benchmarks έχουν σημαντικά μεγαλύτερο αριθμό εντολών, όπως τα `sjeng`, `hmmer`, `omnet` και `xalanc`, κάτι που φάνηκε άμεσα από τον σαφώς μεγαλύτερο χρόνο που χρειάστηκαν για να εκτελεστούν.

4.2 Μελέτη των N-bit predictors

Εδώ μελετάμε την απόδοση των N-bit predictors, χρησιμοποιώντας την υλοποίησή τους στο `cslab_branch.cpp`.

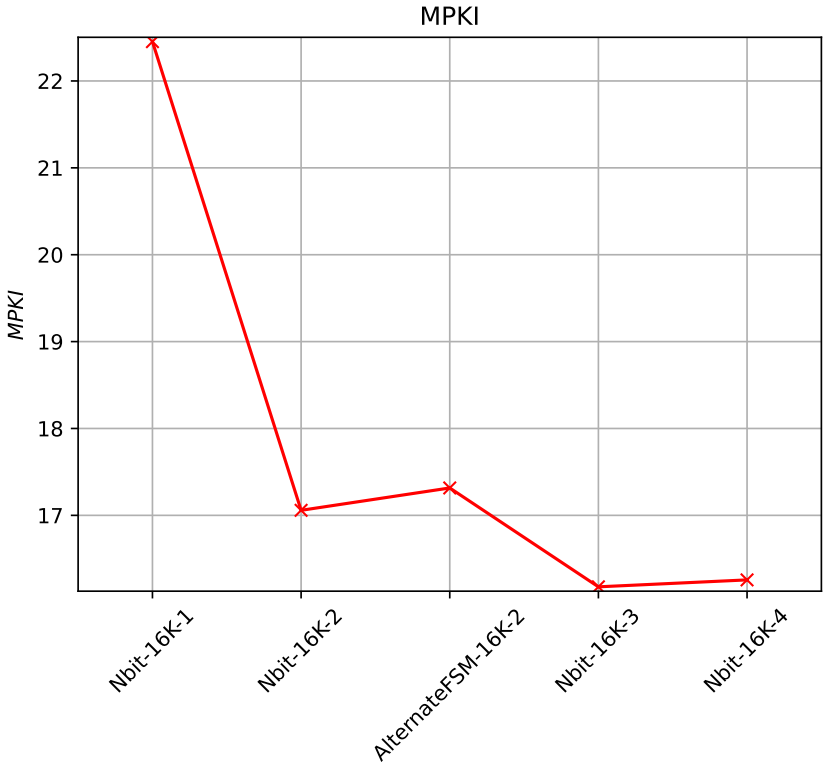
(i)

Διατηρούμε τον αριθμό των BHT entries σταθερό και ίσο με $16\text{K} = 2^{14}$, ενώ το N παίρνει τιμές $\{1, 2, 3, 4\}$. Για $N = 2$ αναλαμβάνουμε να υλοποιήσουμε επιπλέον και το παρακάτω FSM (στο οποίο αναφερόμαστε ως 2β):

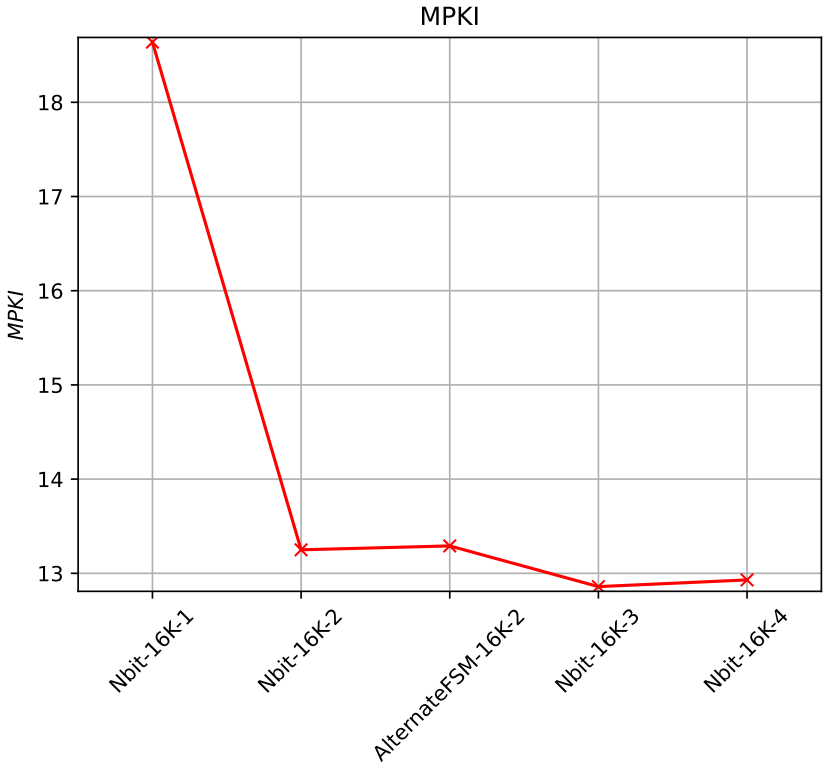


Ως μετρική χρησιμοποιούμε το direction Mispredictions Per Thousand Instructions (direction MPKI), στο οποίο θα αναφερόμαστε ως MPKI από εδώ και στο εξής. Έτσι έχουμε:

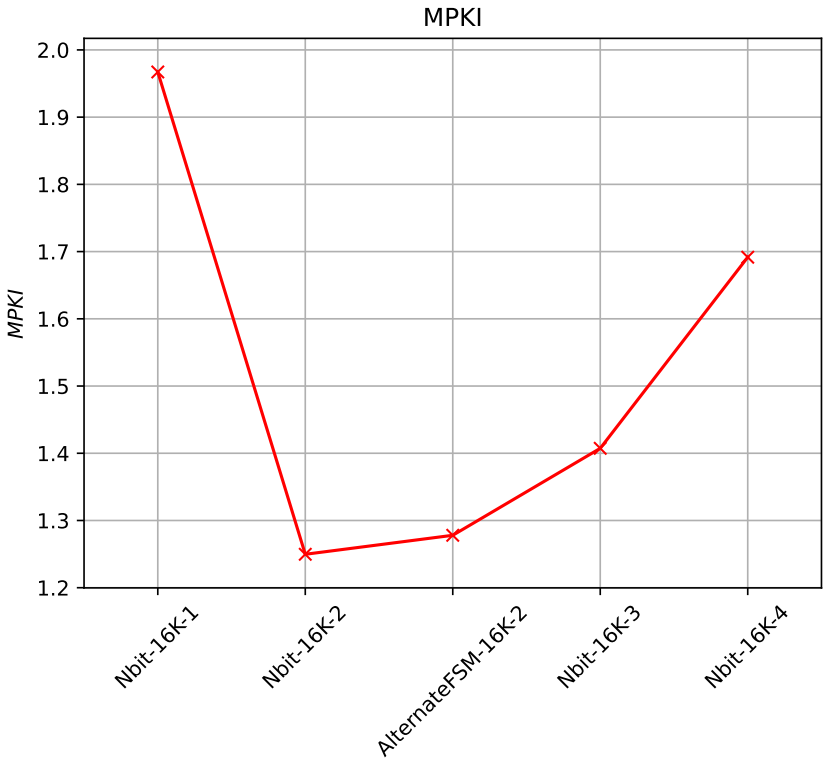
4.2.1 403.gcc



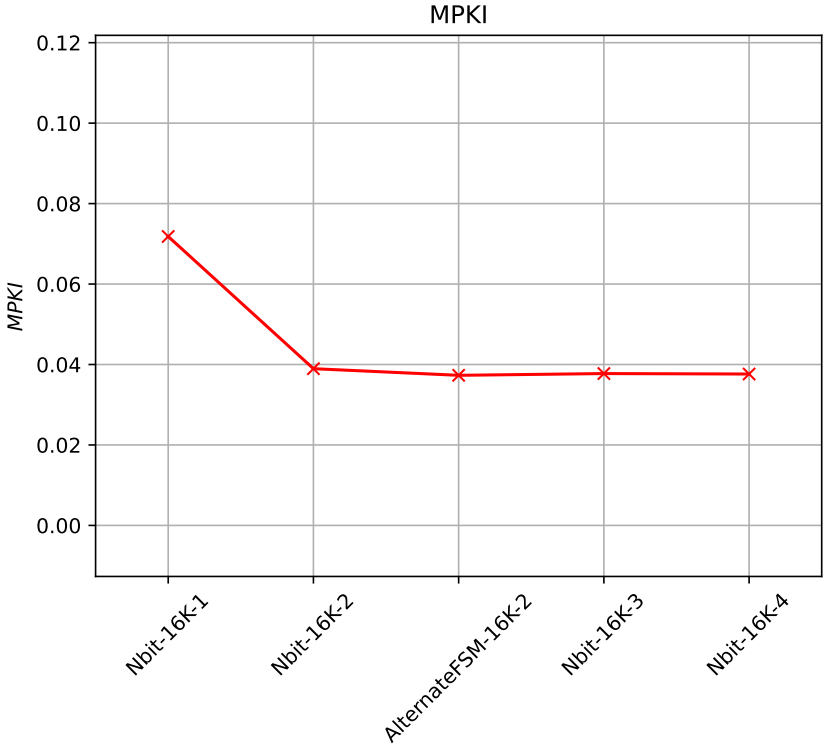
4.2.2 429.mcf



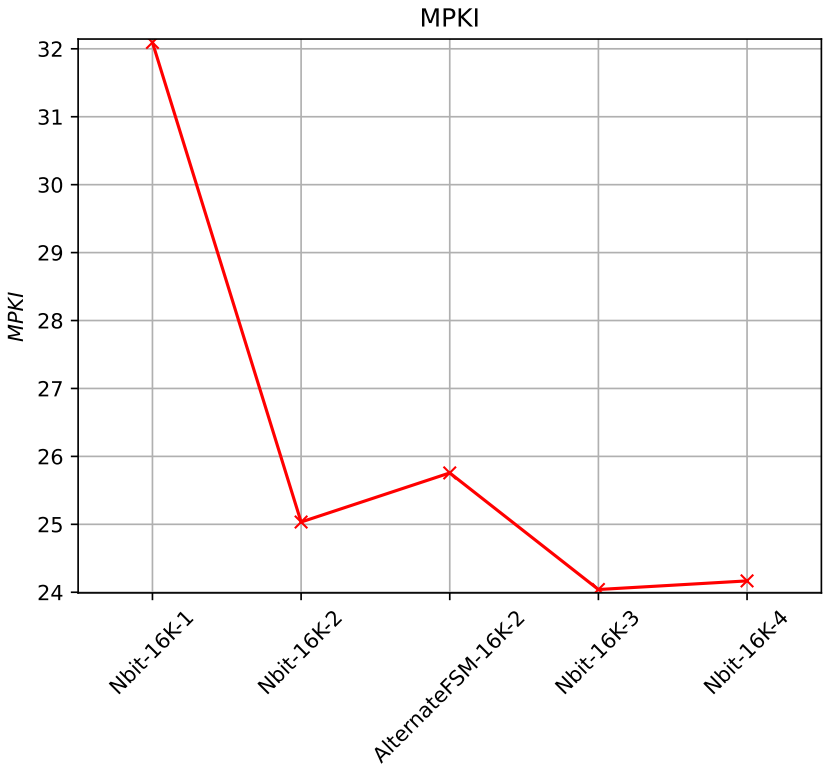
4.2.3 434.zeusmp



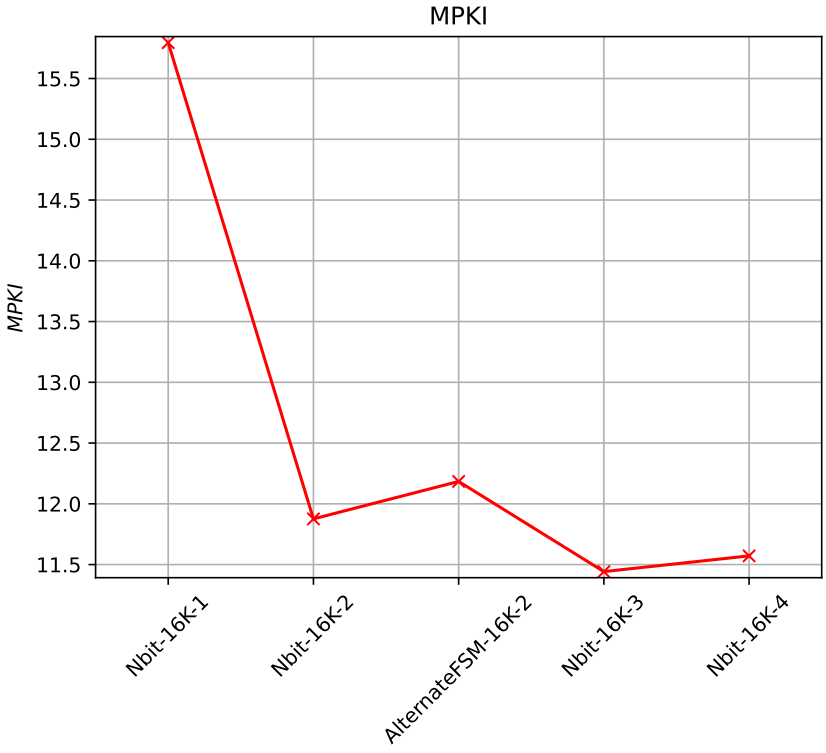
4.2.4 436.cactusADM



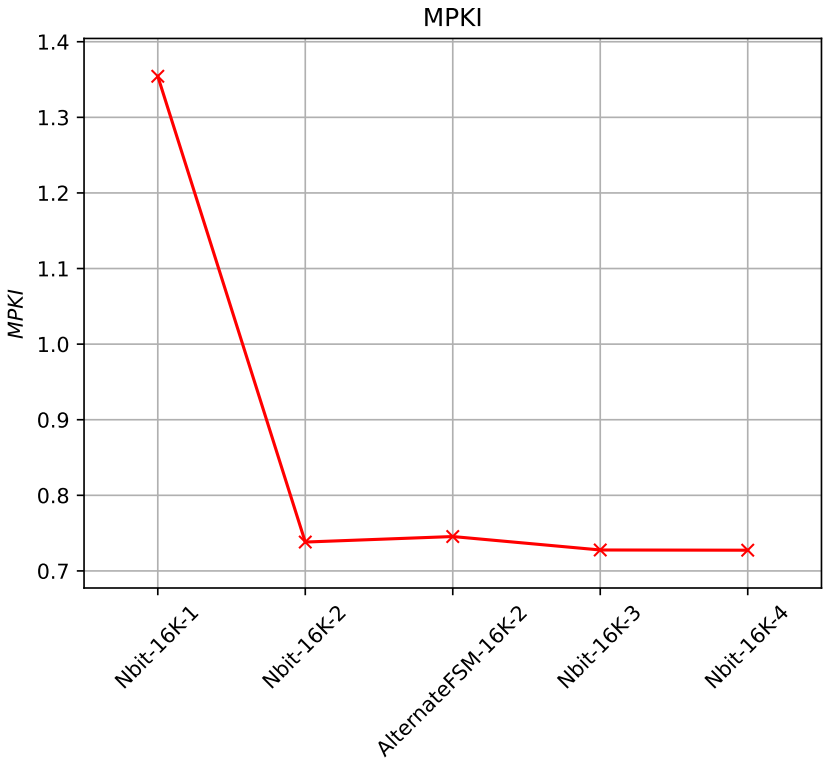
4.2.5 445.gobmk



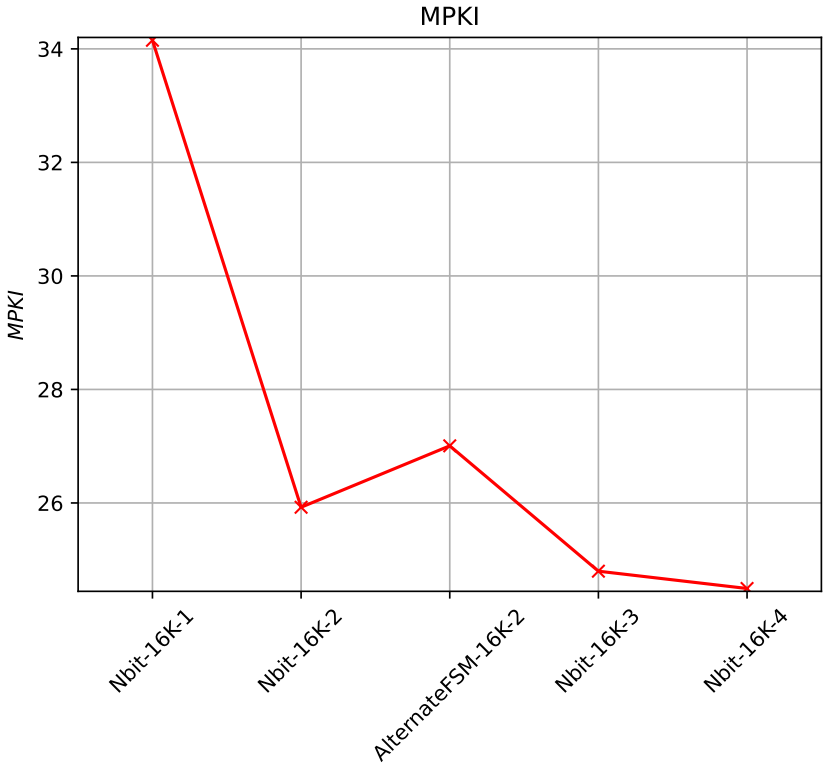
4.2.6 450.soplex



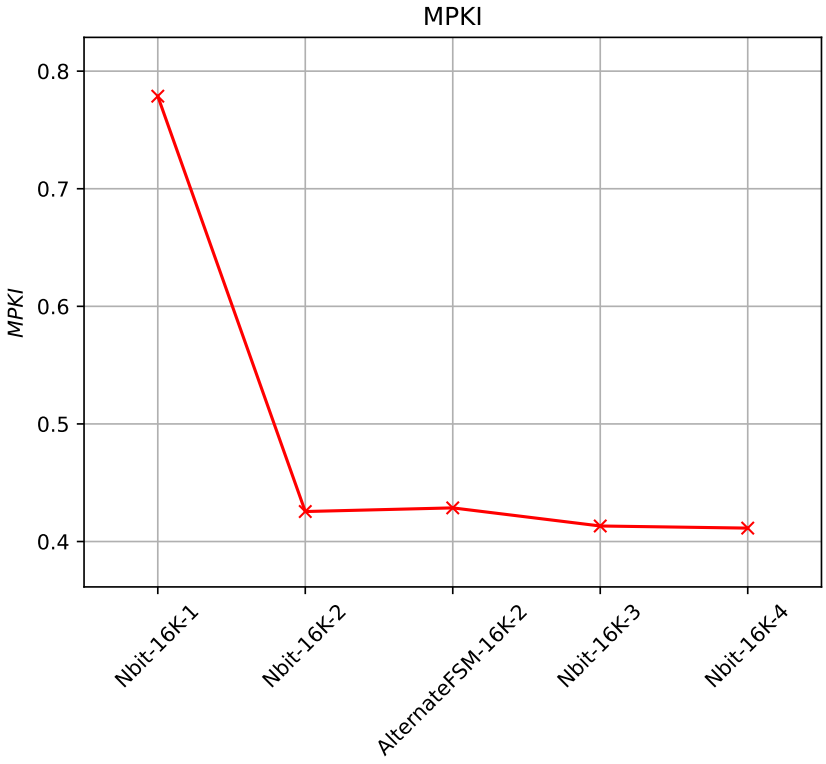
4.2.7 456.hmmer



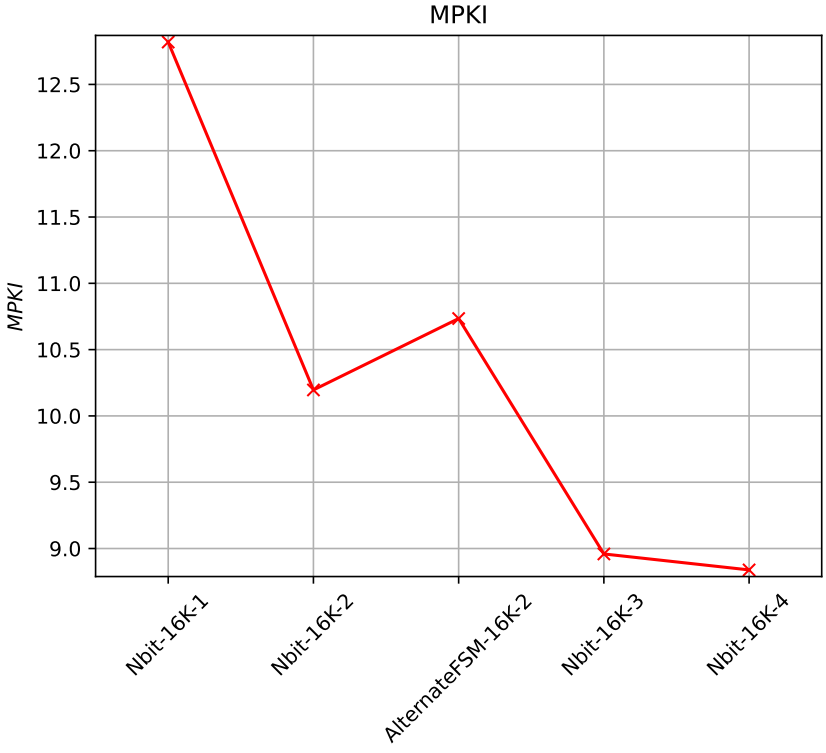
4.2.8 458.sjeng



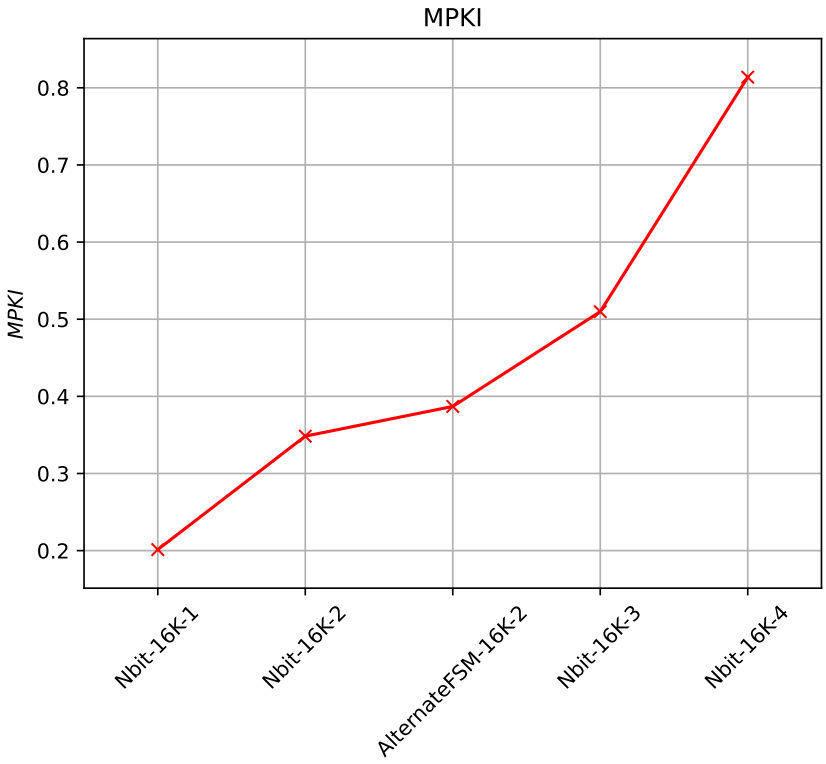
4.2.9 459.GemsFDTD



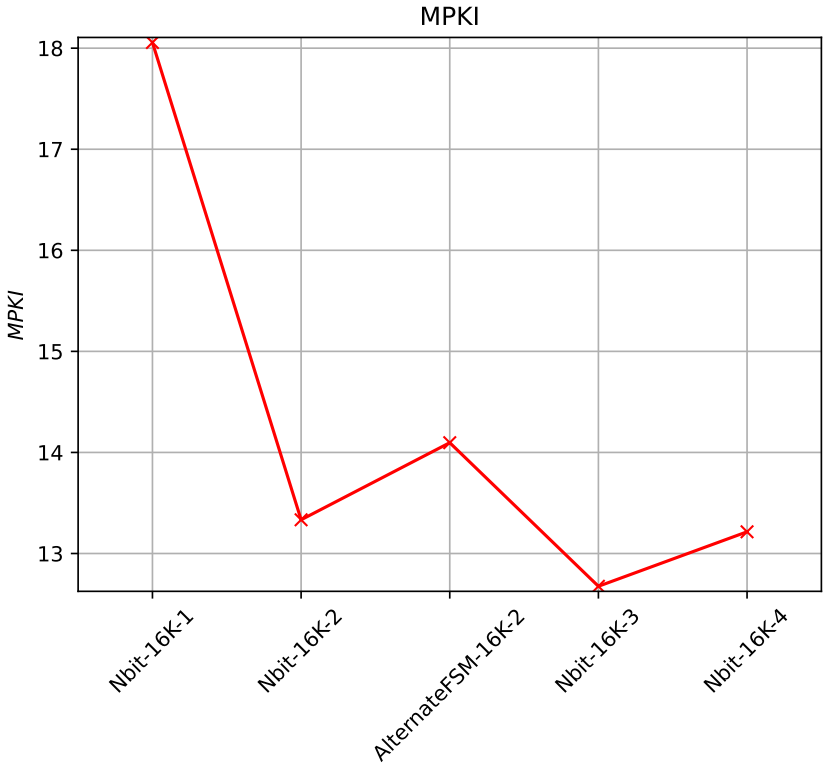
4.2.10 462.libquantum



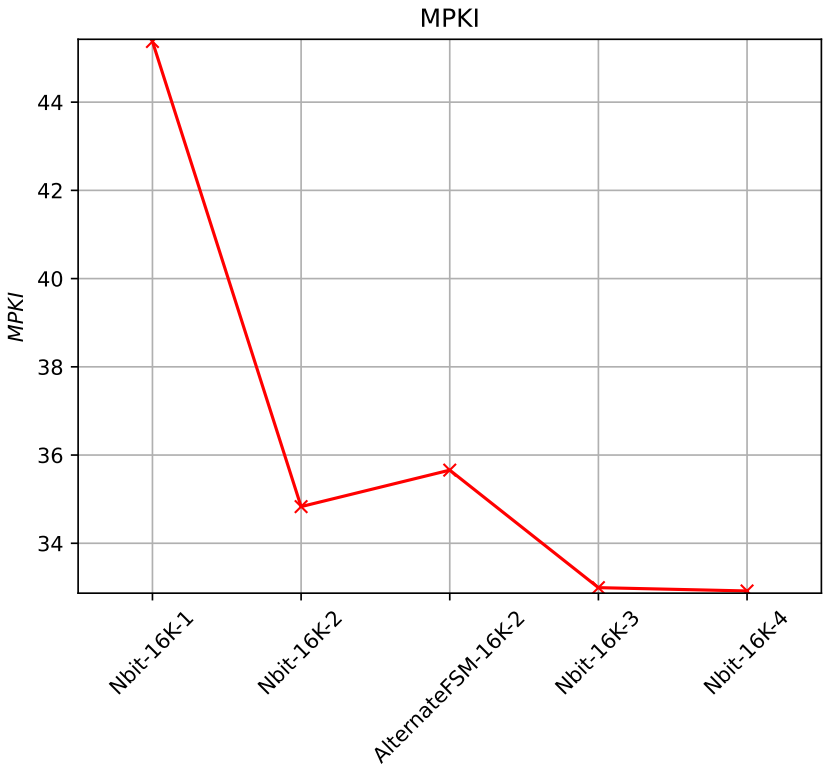
4.2.11 470.lbm



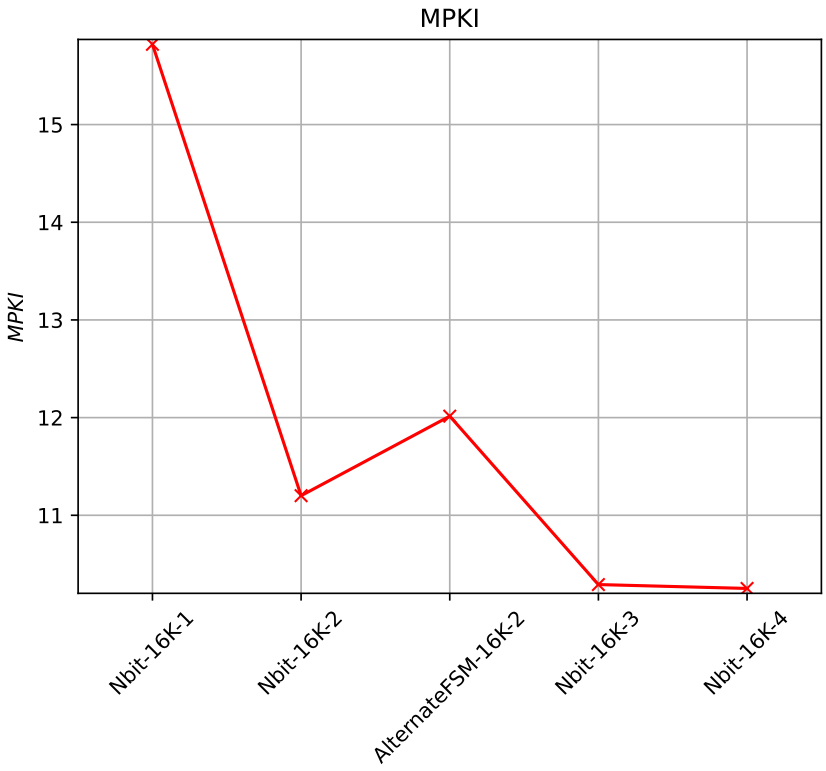
4.2.12 471.omnetpp



4.2.13 473.astar

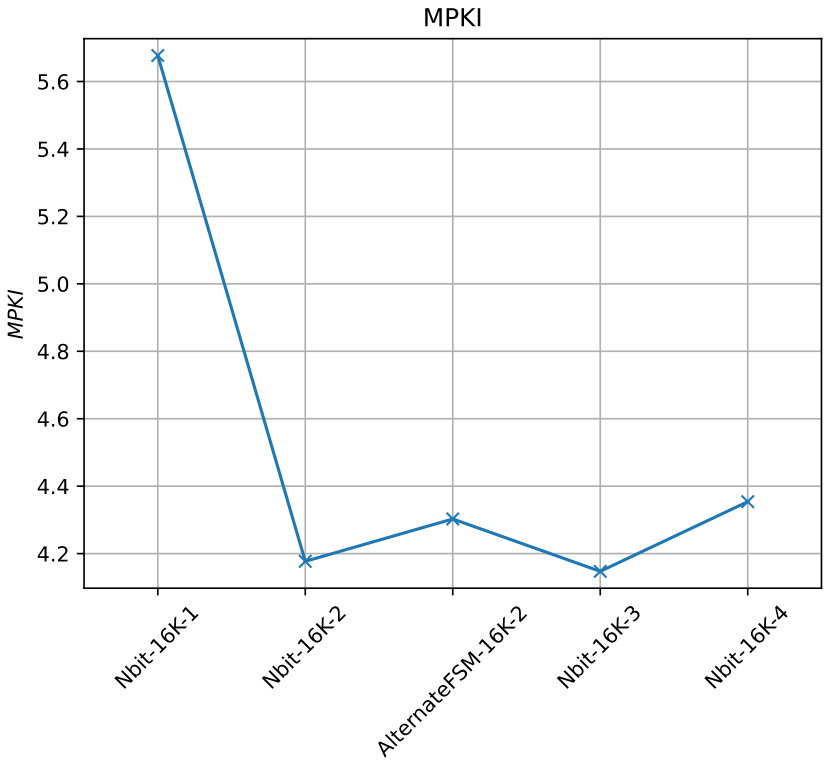


4.2.14 483.xalancbmk



4.2.15 Γεωμετρικός μέσος των benchmarks

Για λόγους συνολικής εποπτείας, παρουσιάζουμε και τον γεωμετρικό μέσο των benchmarks:



4.2.16 Παρατηρήσεις

Παρατηρούμε ότι σε όλα τα benchmarks εκτός από τα `434.zeusmp` και `470.lbm`, η αύξηση των bits του predictor βελτιώνει τις επιδόσεις (μειώνεται το MPKI). Εξαίρεση αποτελεί η αύξηση από 3 bits σε 4 bits, η οποία δεν είναι πάντα βελτιωτική. Αξίζει να παρατηρηθεί βέβαια ότι στα benchmarks `434.zeusmp` και `470.lbm`, το MPKI είναι της τάξης του 0-2, οπότε οι διακυμάνσεις στις τιμές για διαφορετικές τιμές του N είναι σχεδόν αμελητέες.

Όσον αφορά το εναλλακτικό FSM που υλοποιήθηκε, αυτό δεν αποδίδει ποτέ καλύτερα από τον αντίστοιχο 2-bit predictor (saturating up-down counter) που υπήρχε στον βοηθητικό κώδικα που μας δόθηκε.

Σύμφωνα με τον γεωμετρικό μέσο όρο των benchmarks, ο οποίος συμφωνεί με τα παραπάνω συμπεράσματα, ο καλύτερος predictor είναι ο **Nbit-16K-3bit**.

(ii)

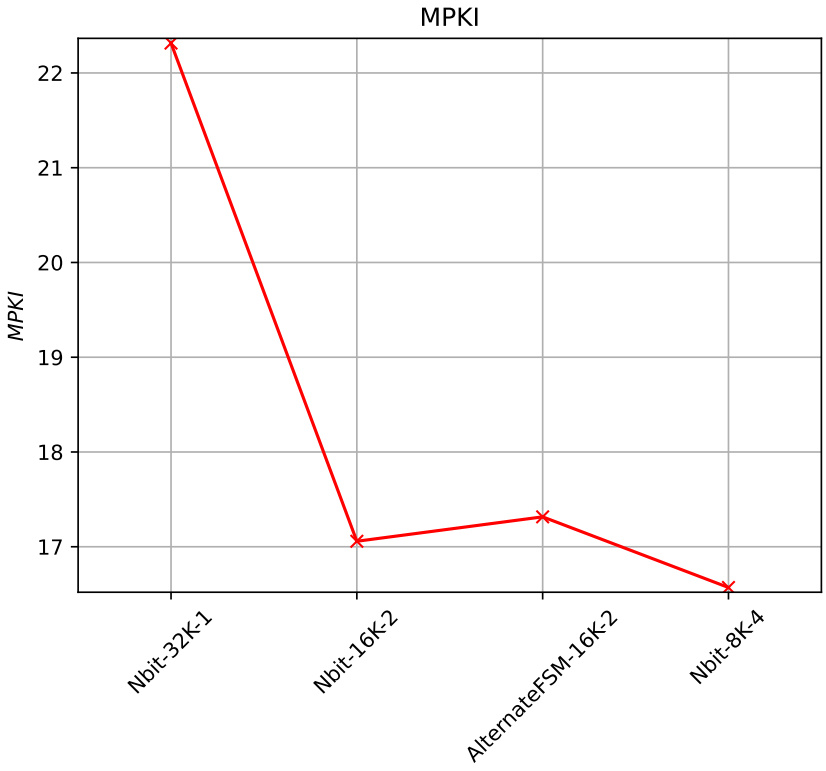
Αυτή τη φορά διατηρούμε σταθερό το hardware (ίσο με $32\text{K} = 2^{15}$ bits), ενώ το N παίρνει πάλι τιμές $\{1, 2, 3, 4\}$. Επομένως θα έχουμε:

- BHT entries $=$ 32K και $N = 1$
- BHT entries $=$ 16K και $N = 2$

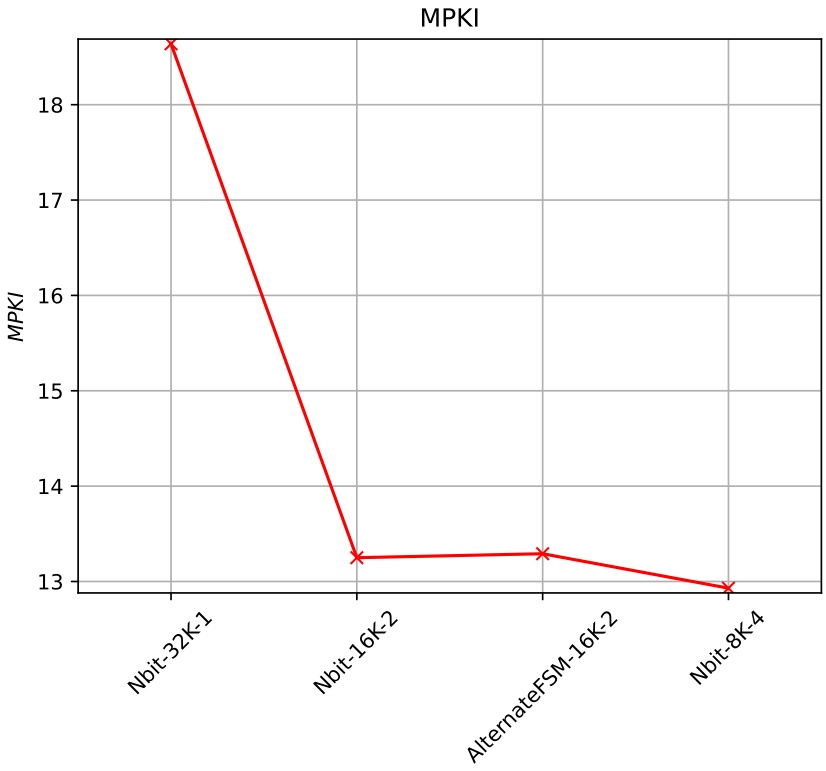
- BHT entries $=$ 16K και $N = 2\beta$ (εναλλακτικό FSM)
- BHT entries $=$ 8K και $\mathrm{N} = 4$

Έχουμε:

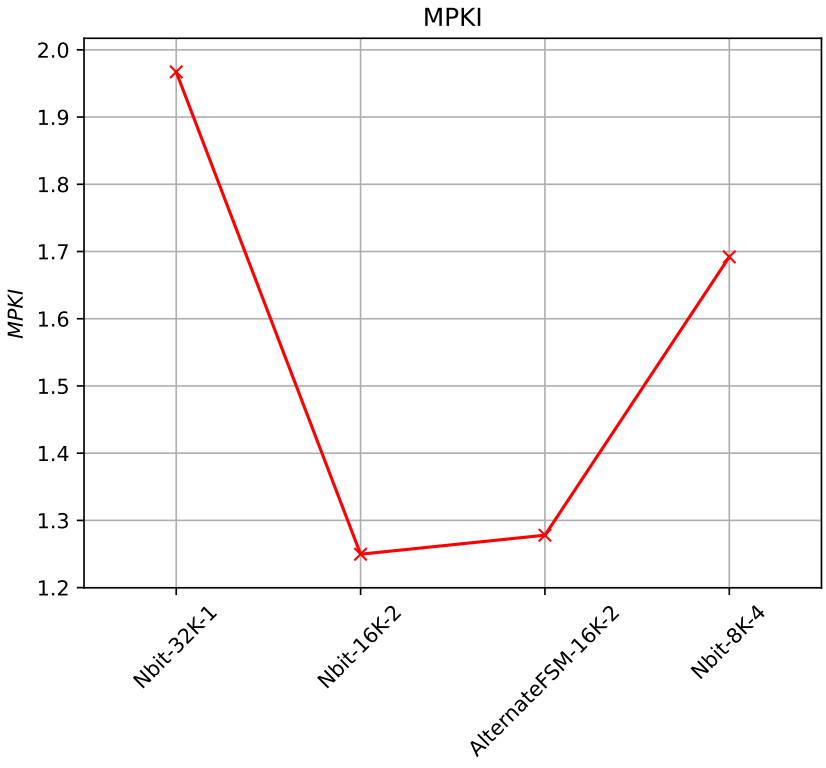
4.2.17 403.gcc



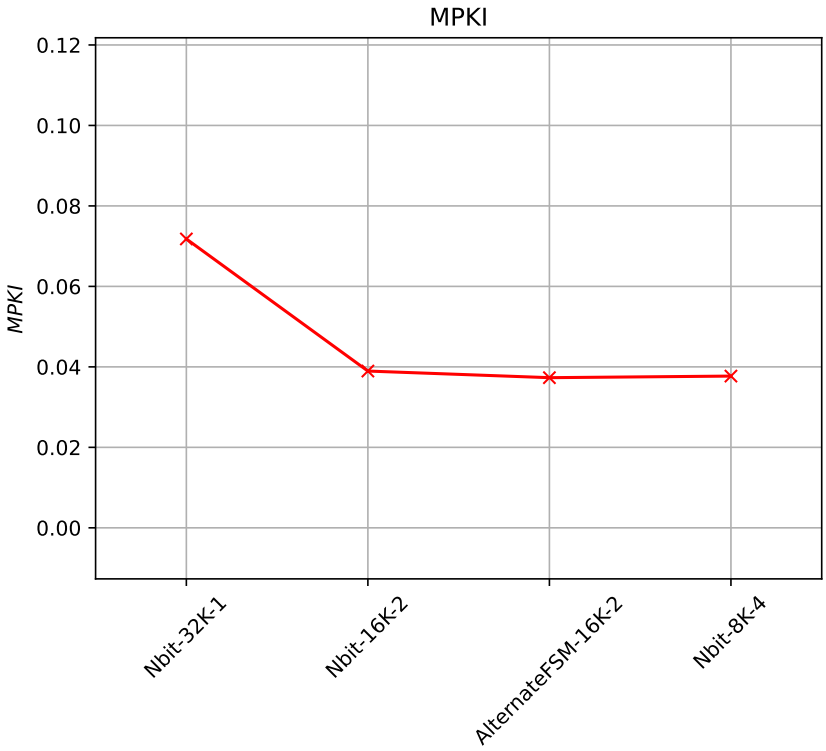
4.2.18 429.mcf



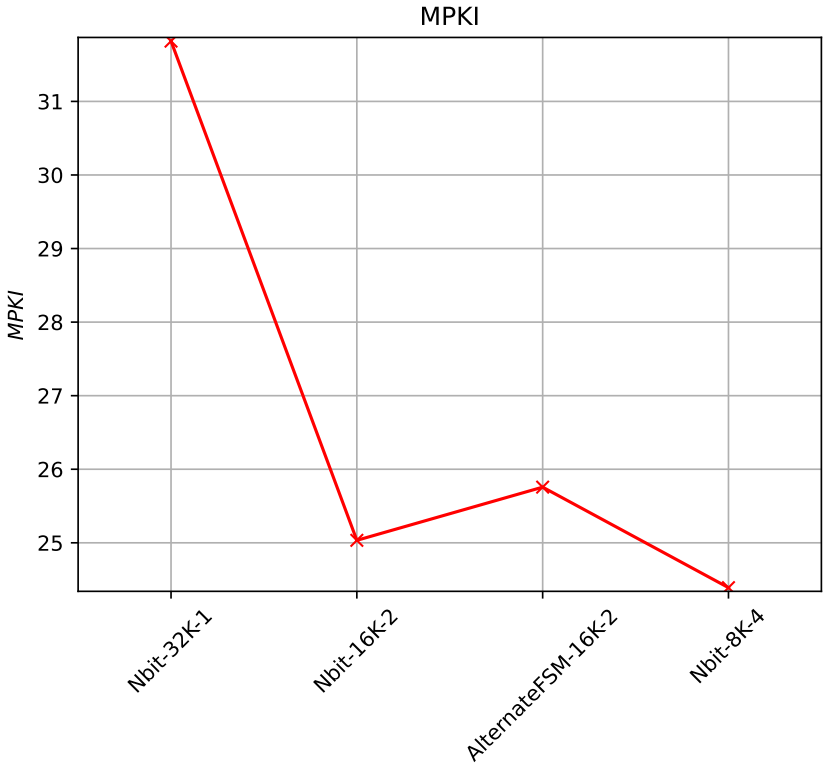
4.2.19 434.zeusmp



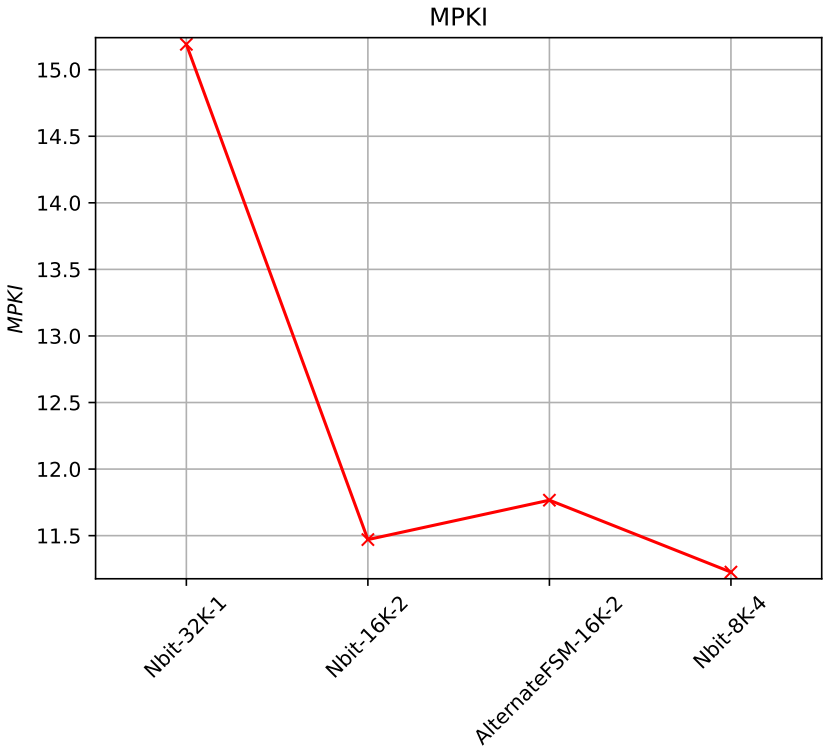
4.2.20 436.cactusADM



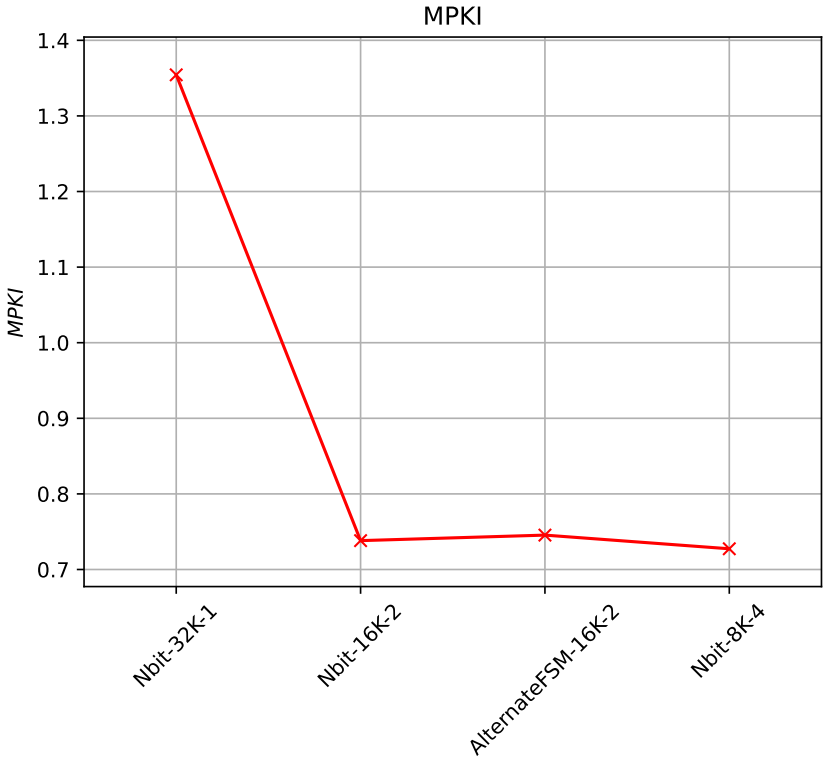
4.2.21 445.gobmk



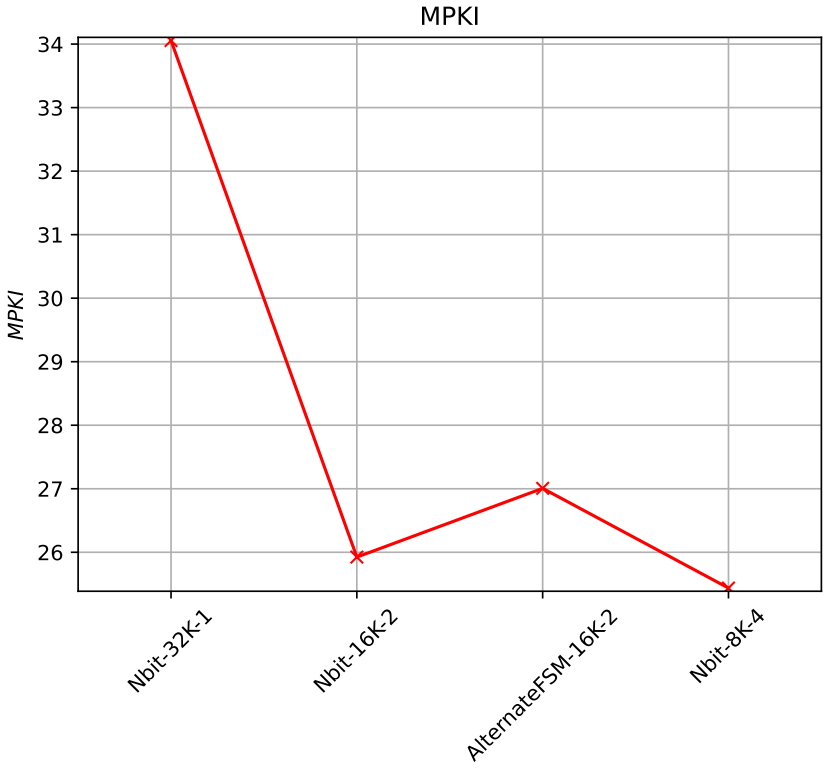
4.2.22 450.soplex



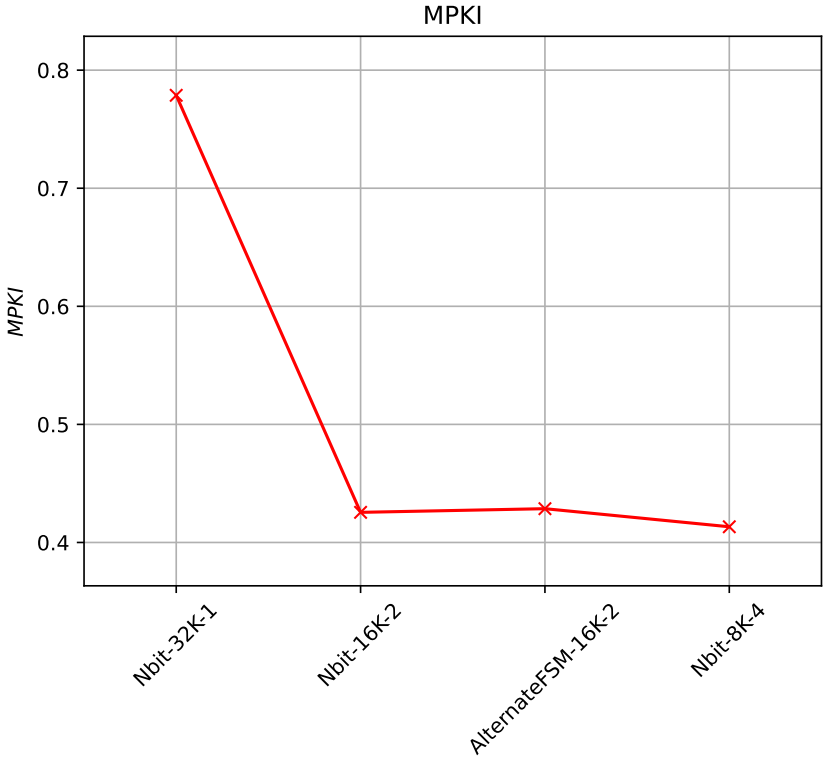
4.2.23 456.hmmer



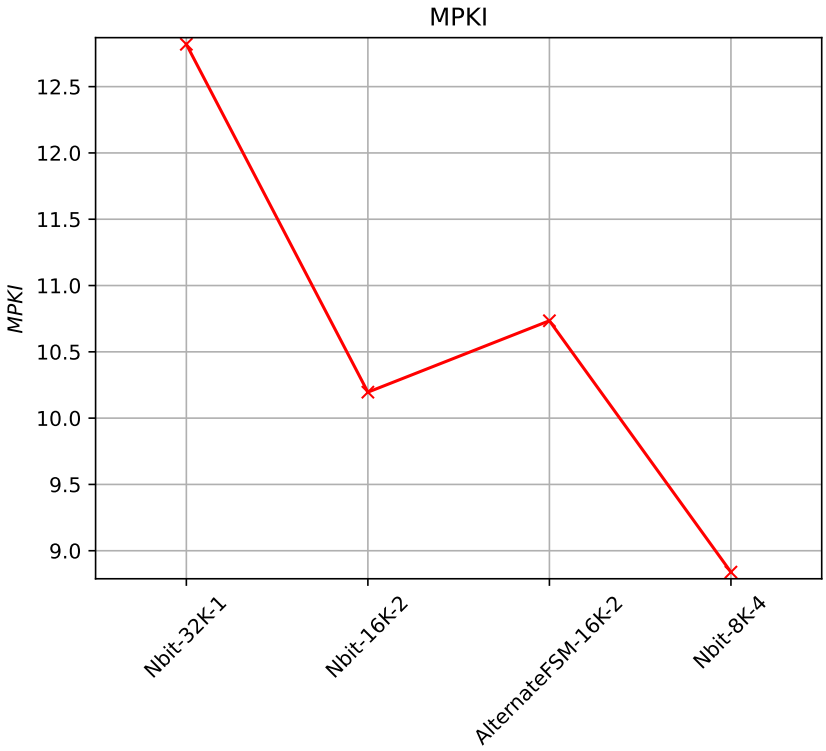
4.2.24 458.sjeng



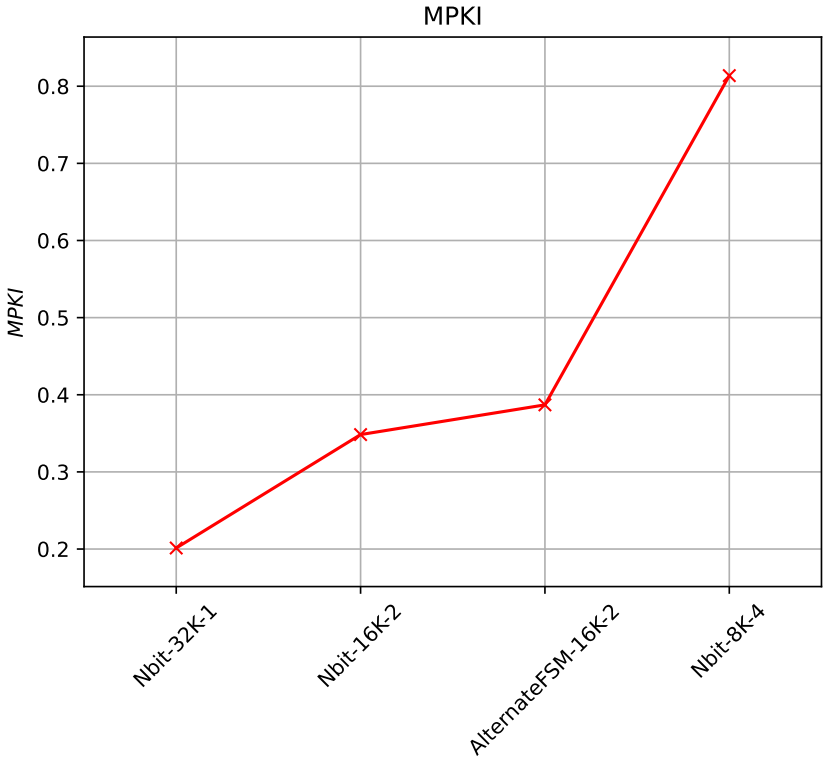
4.2.25 459.GemsFDTD



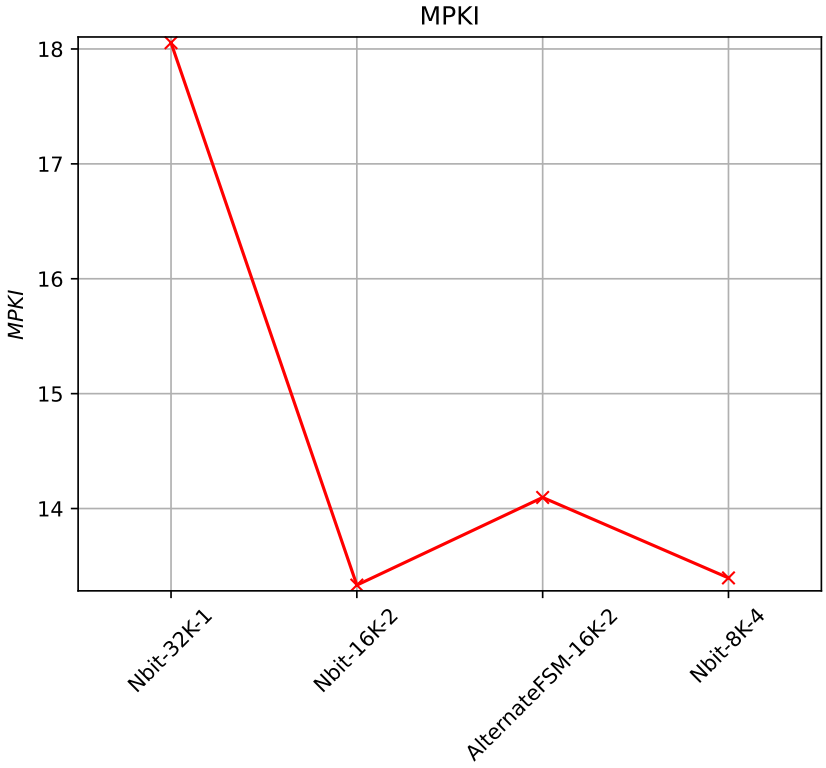
4.2.26 462.libquantum



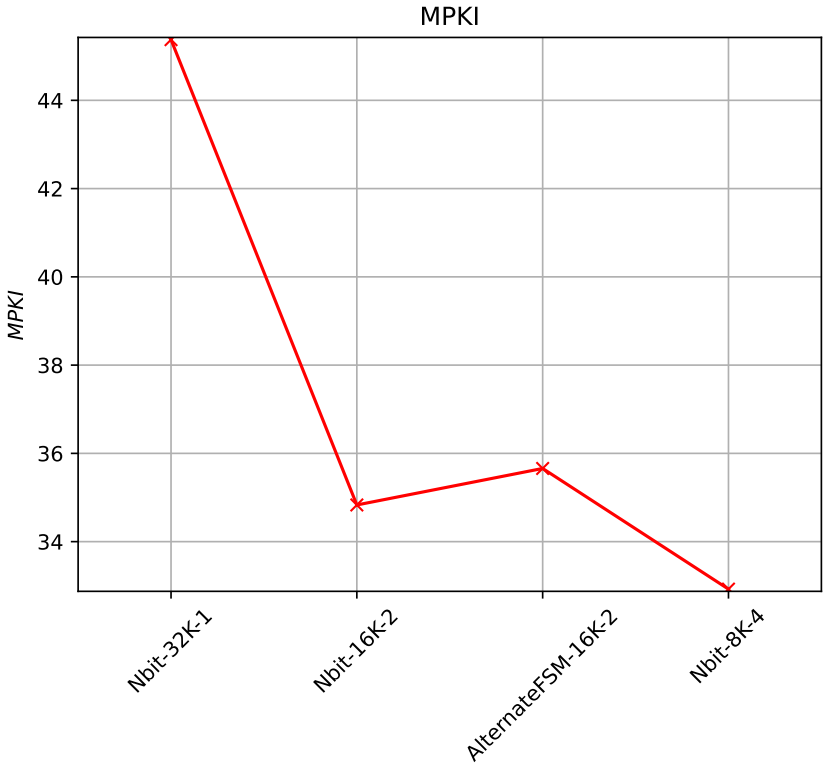
4.2.27 470.lbm



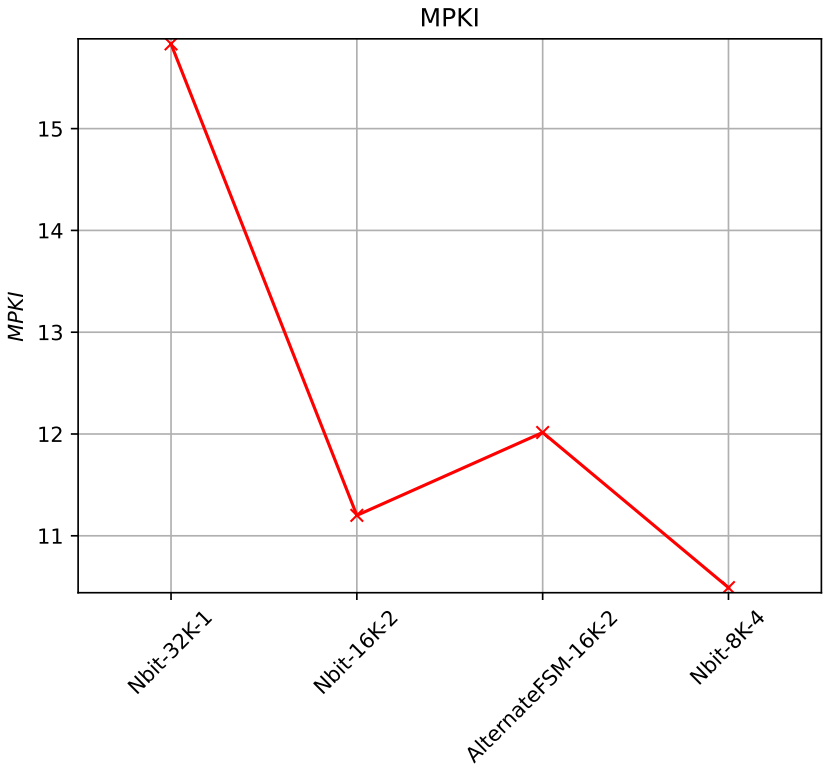
4.2.28 471.omnetpp



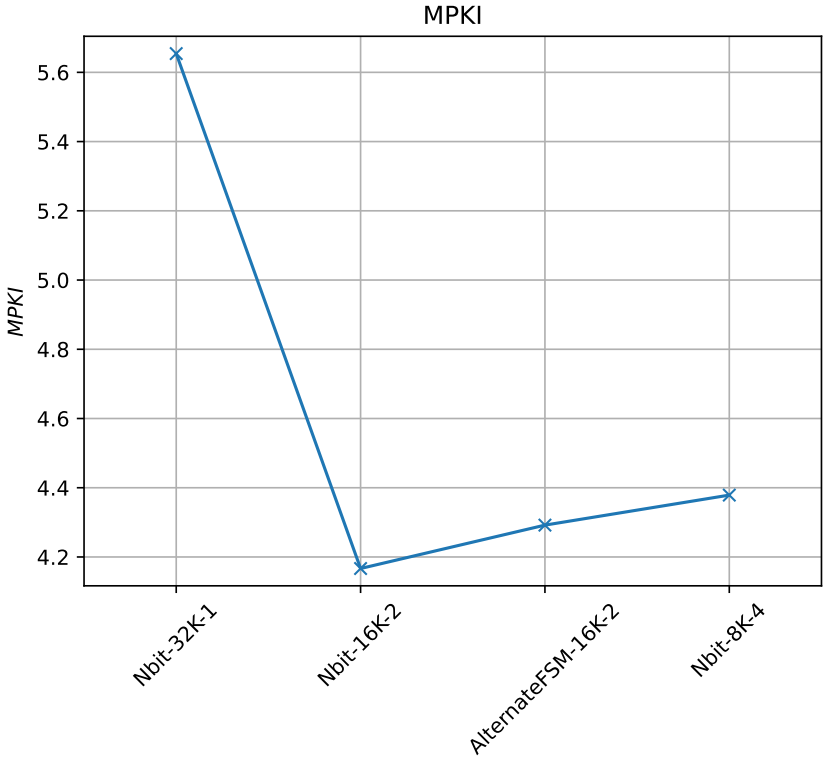
4.2.29 473.astar



4.2.30 483.xalancbmk



4.2.31 Γεωμετρικός μέσος των benchmarks



4.2.32 Παρατηρήσεις

Τα αποτελέσματα είναι ανάλογα με αυτά του προηγούμενου ερωτήματος. Εδώ με βάση τα διαγράμματα για κάθε benchmark αλλά και τον γεωμετρικό μέσο των αποτελεσμάτων, θα επιλέγαμε τον **Nbit-16K-2bit** predictor.

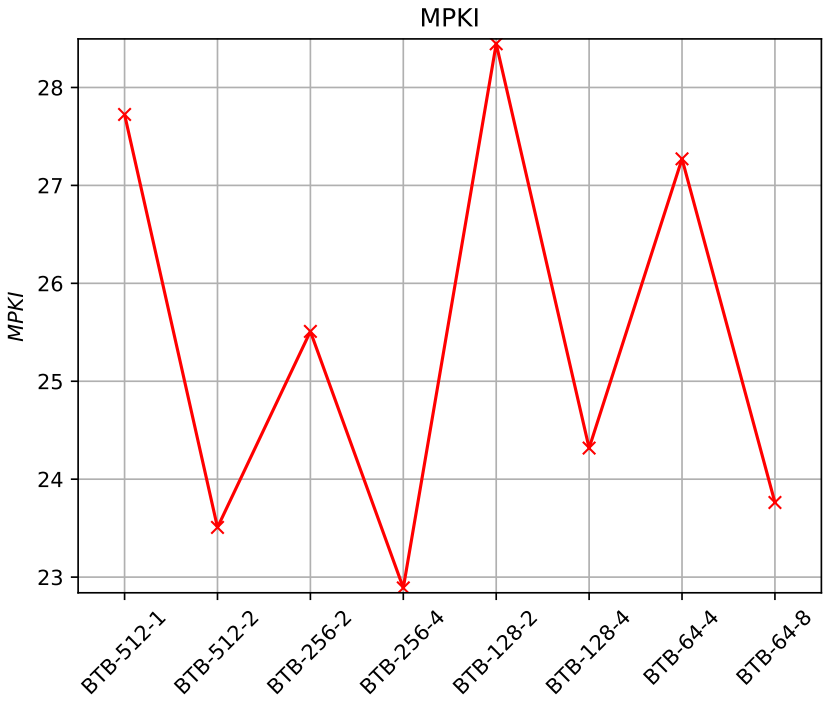
4.3 Μελέτη του BTB

Υλοποιούμε έναν BTB και μελετούμε την ακρίβεια πρόβλεψής του για τις περιπτώσεις:

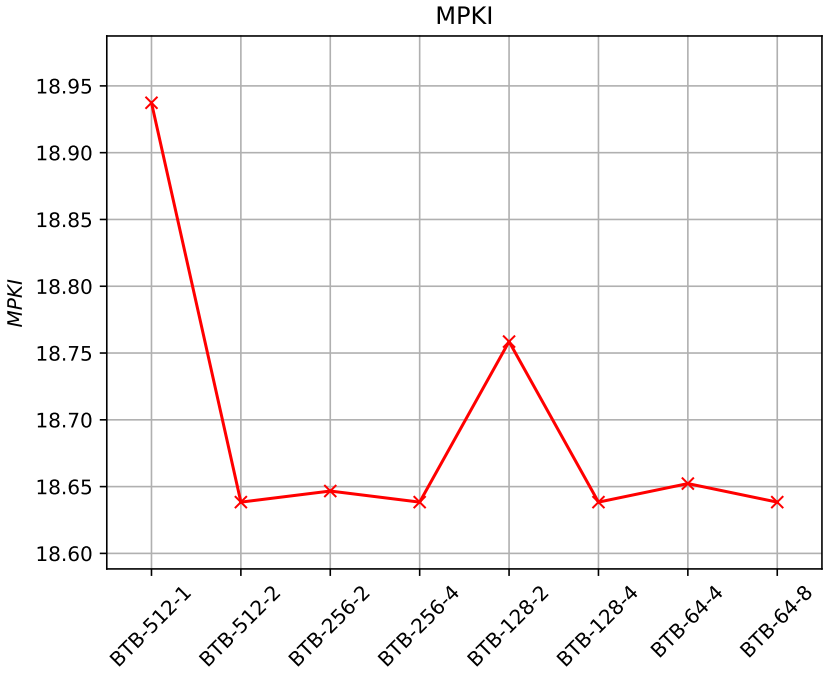
BTB entries	BTB associativity
512	1
512	2
256	2
256	4
128	2
128	4
64	4
64	8

Έχουμε:

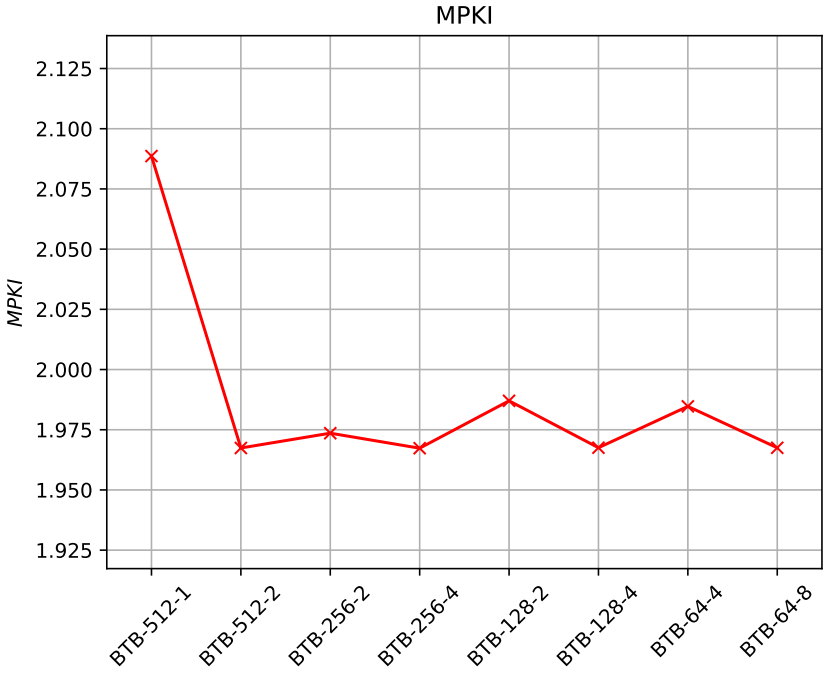
4.3.1 403.gcc



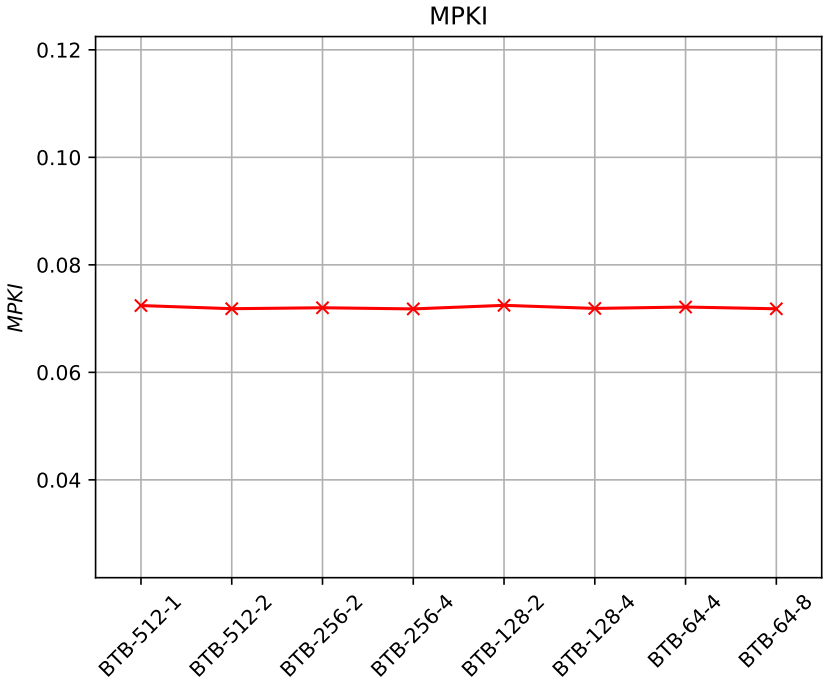
4.3.2 429.mcf



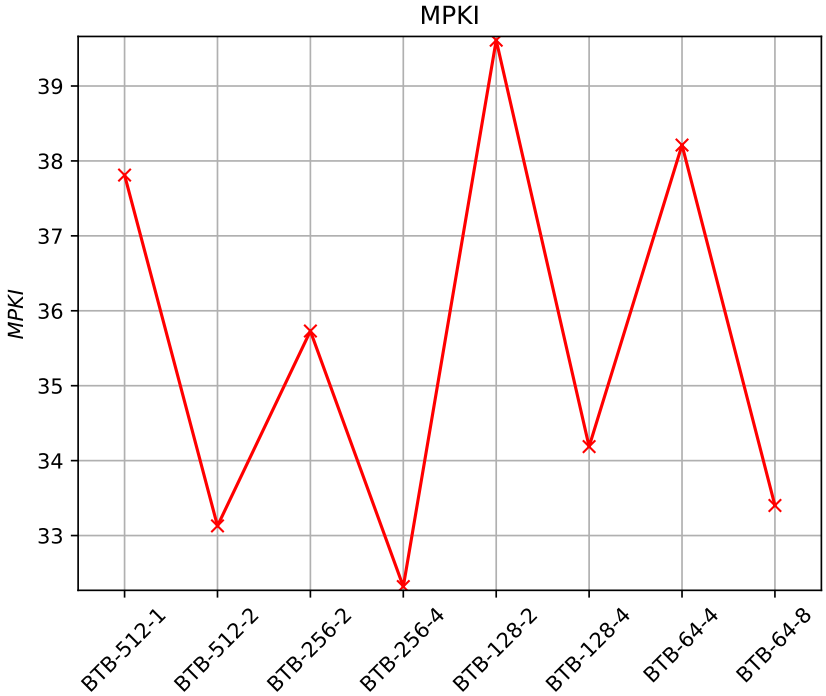
4.3.3 434.zeusmp



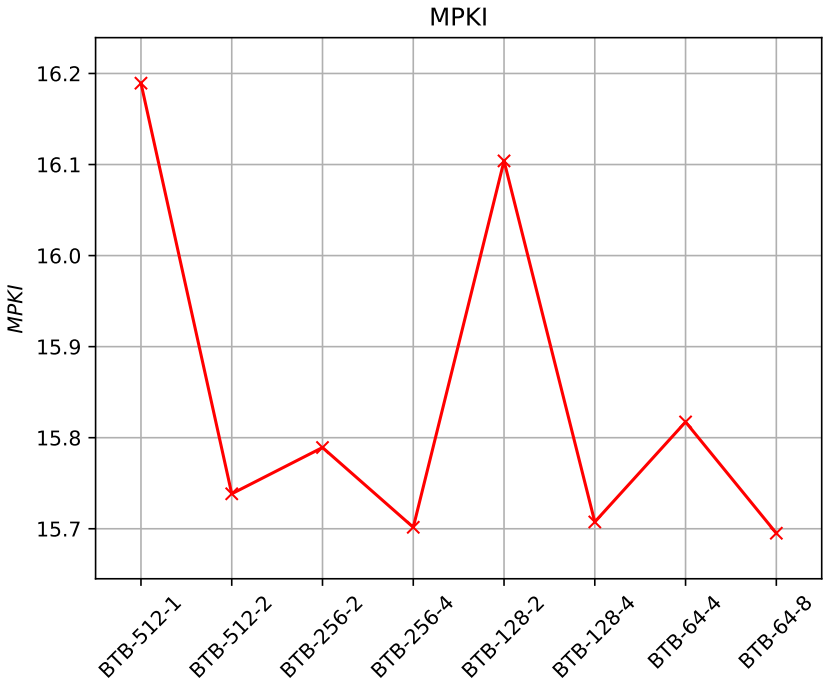
4.3.4 436.cactusADM



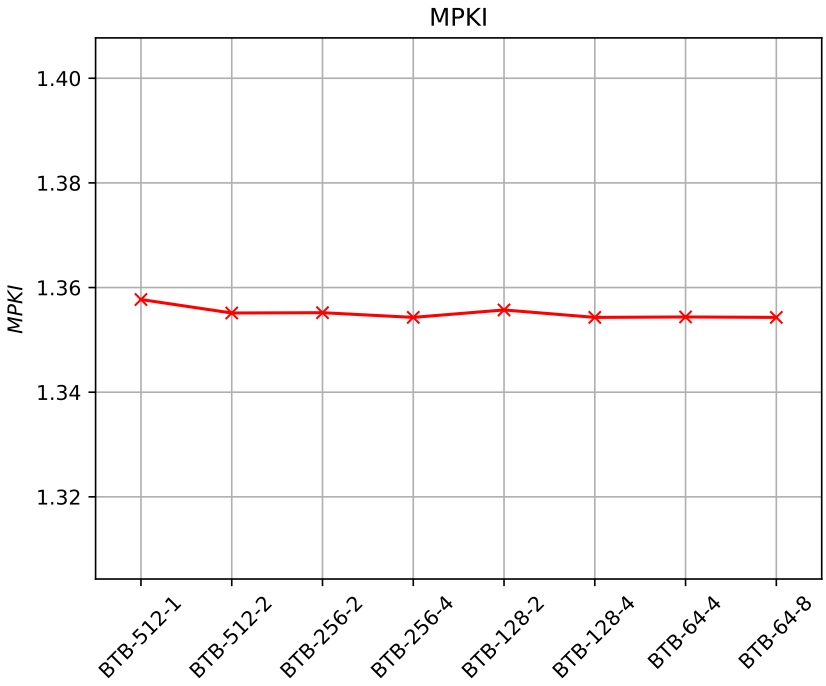
4.3.5 445.gobmk



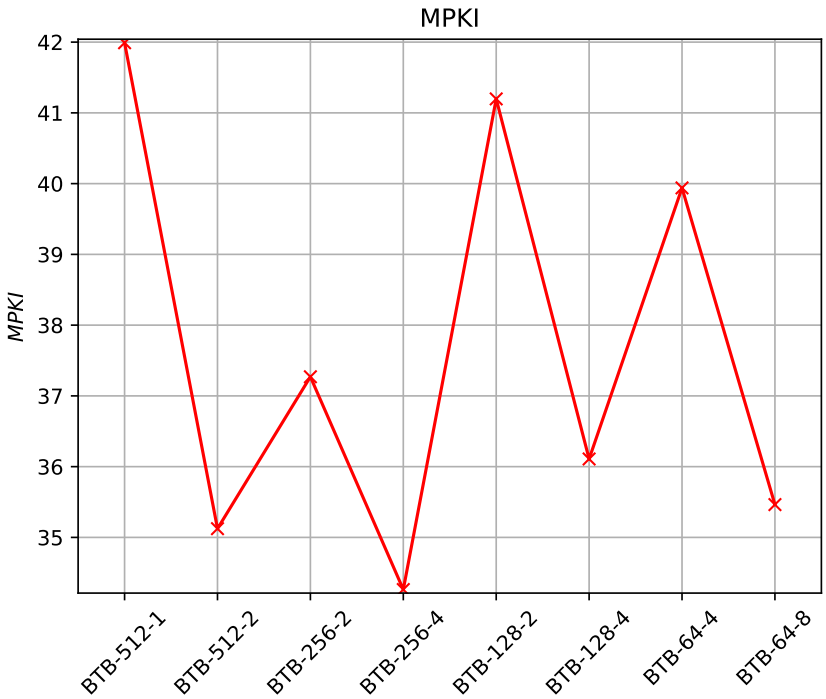
4.3.6 450.soplex



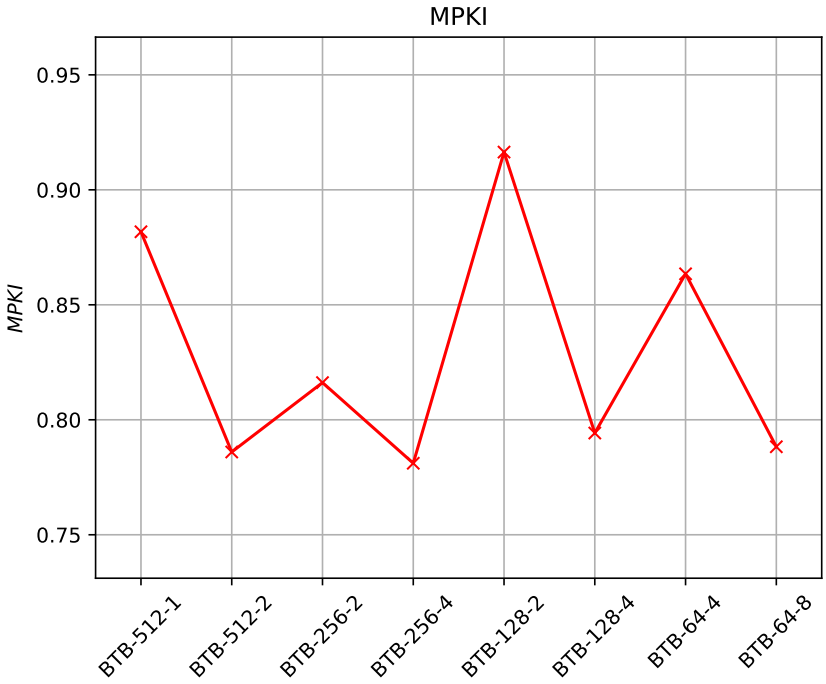
4.3.7 456.hmmer



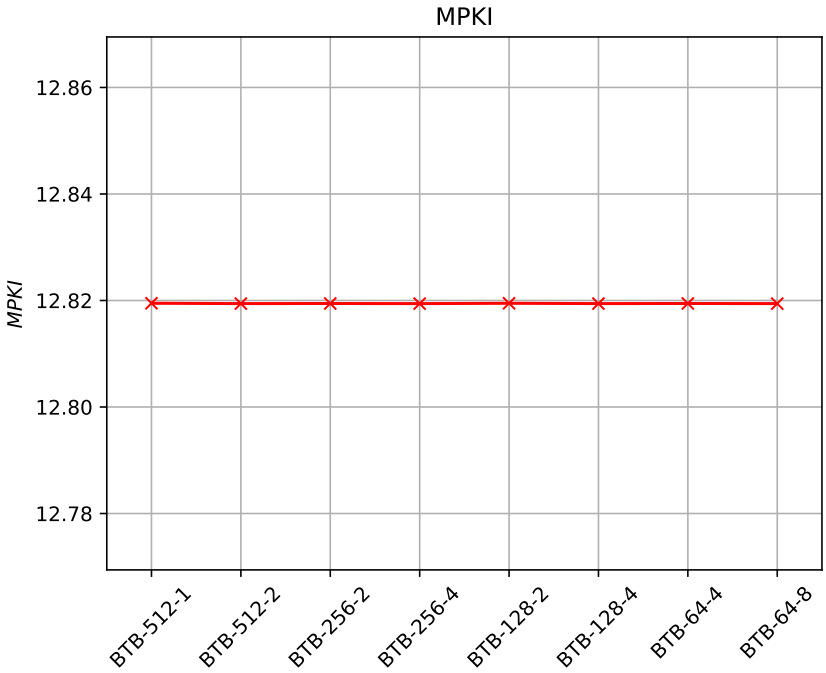
4.3.8 458.sjeng



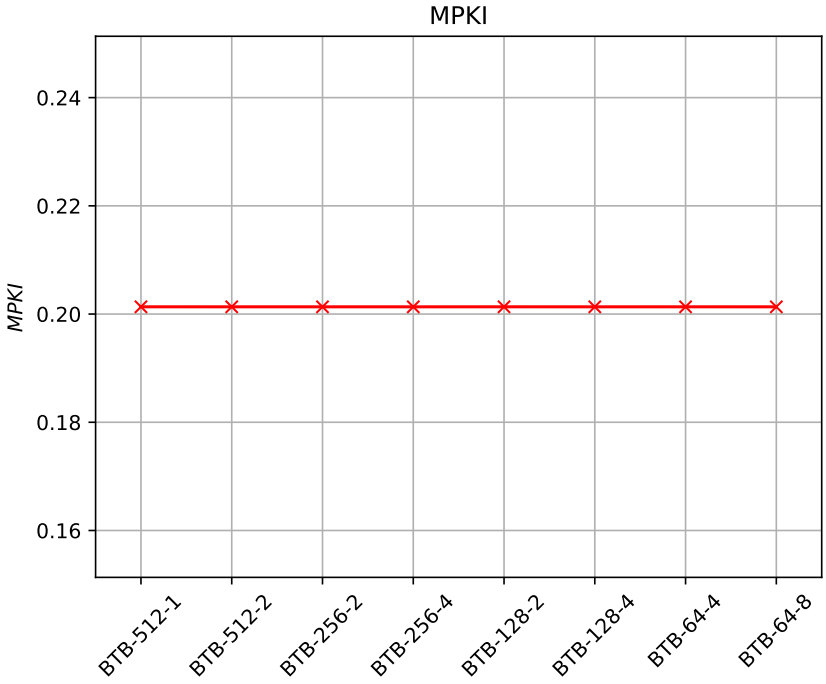
4.3.9 459.GemsFDTD



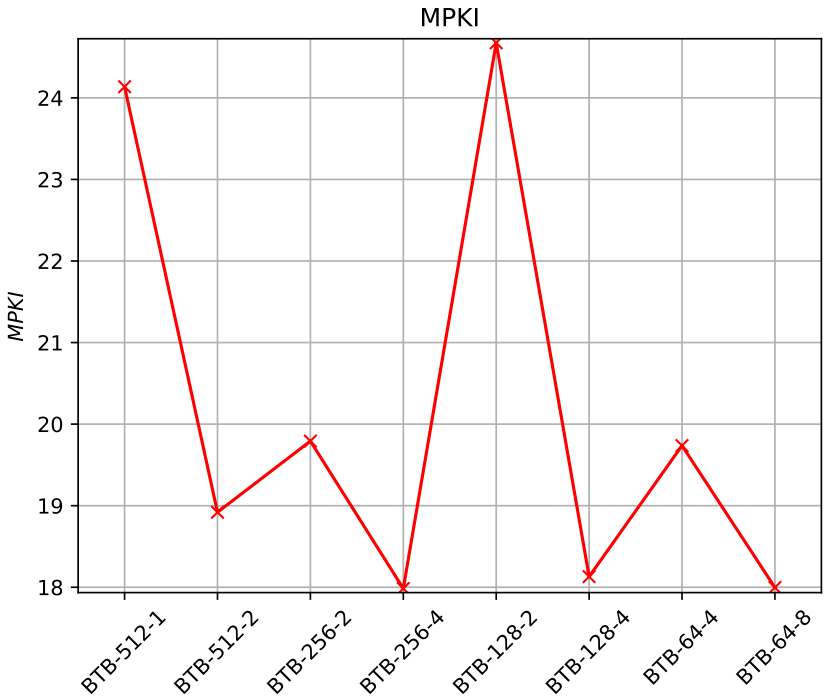
4.3.10 462.libquantum



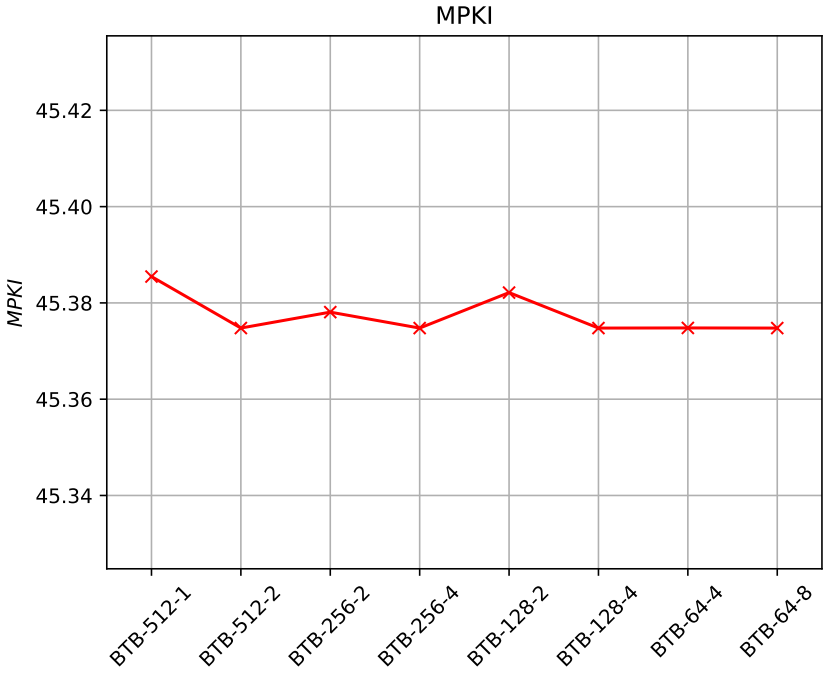
4.3.11 470.lbm



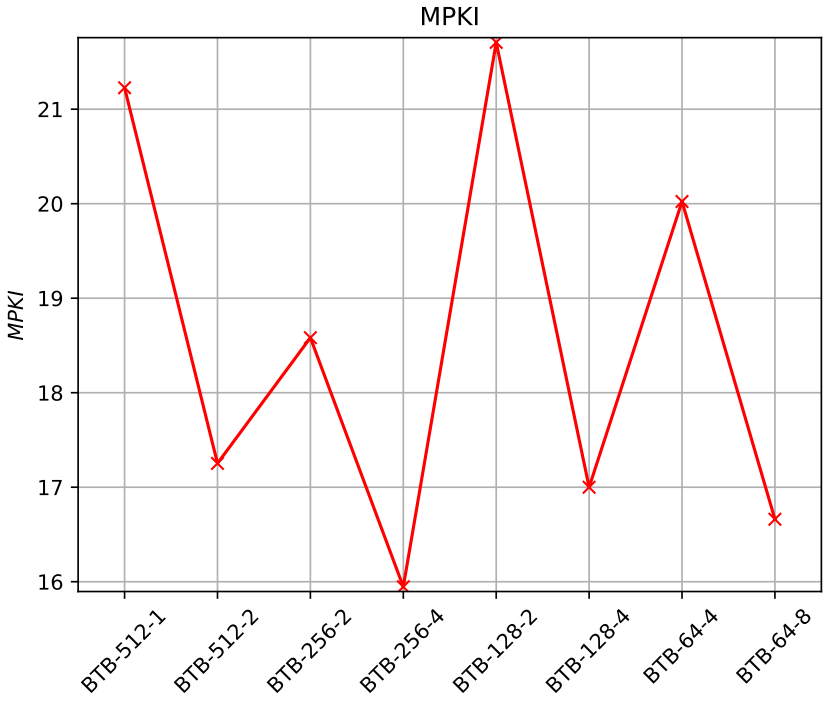
4.3.12 471.omnetpp



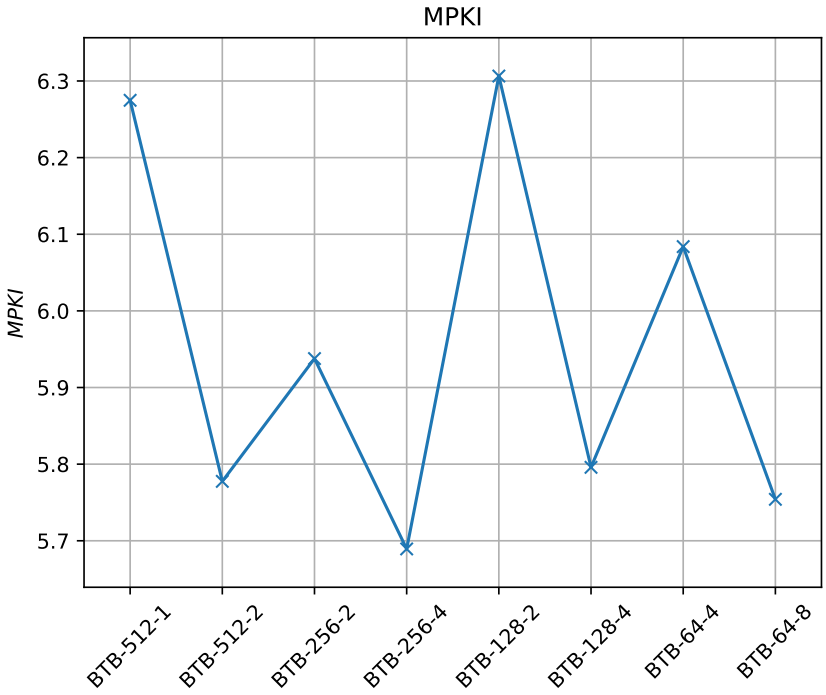
4.3.13 473.astar



4.3.14 483.xalancbmk



4.3.15 Γεωμετρικός μέσος των benchmarks



4.3.16 Παρατηρήσεις

Σημειώνεται ότι επειδή έχουμε να κάνουμε με BTB, υπάρχουν δύο περιπτώσεις misses, τα direction misprediction misses και τα target misprediction misses.

Από τα διαγράμματα παρατηρούμε ότι για σταθερό πλήθος entries (γινόμενο table lines $\times$ associativity σταθερό), το associativity δρα βελτιωτικά. Συγκεκριμένα, για γινόμενο 512, ο BTB-64-8 έχει το μικρότερο MPKI, ενώ για γινόμενο 256, ο BTB-64-4 έχει τις καλύτερες επιδόσεις. Συνολικά, ο BTB-256-4 έχει συνολικά την καλύτερη επίδοση. Να σημειωθεί ότι ενώ σε κάποια benchmarks το MPKI φαίνεται να είναι σχεδόν 0, αυτό μπορεί να οφείλεται στο ότι τα συγκεκριμένα benchmarks δεν έχουν πολλές εντολές άλματος εξαρχής. Με βάση το διάγραμμα των γεωμετρικών μέσων, καλύτερη επίδοση φαίνεται να έχει ο **BTB-256-2** predictor.

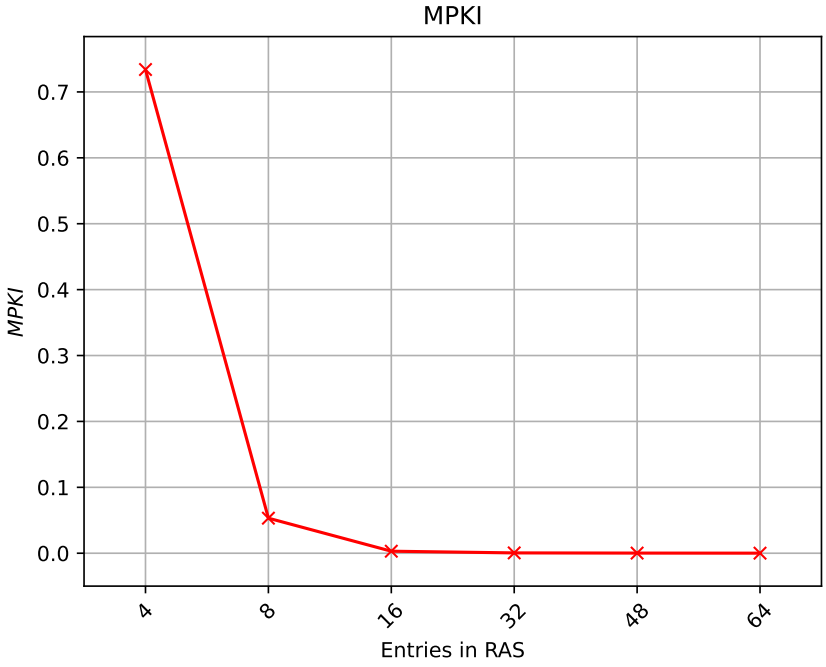
4.4 Μελέτη του RAS

Σε αυτό το σημείο χρησιμοποιούμε το αρχείο `ras.h`, το οποίο περιέχει μια υλοποίηση της RAS. Μελετούμε το ποσοστό αστοχίας για τις ακόλουθες περιπτώσεις:

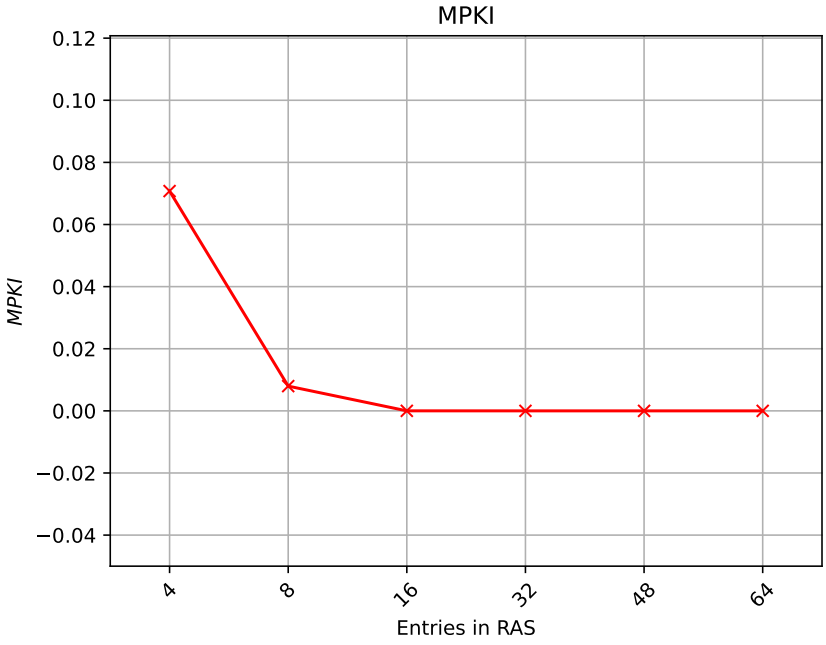
Αριθμός εγγραφών στη RAS
4
8
16
32
48
64

Έχουμε:

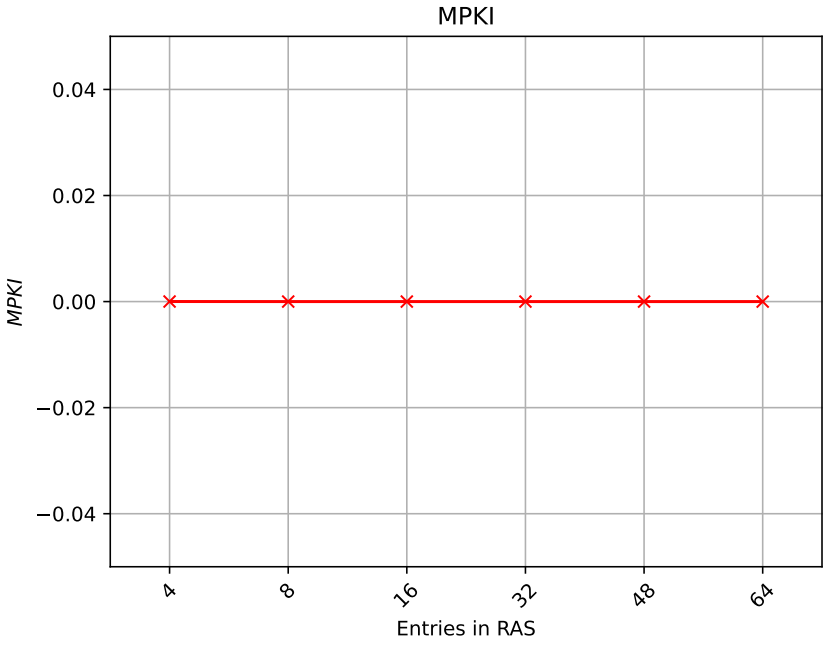
4.4.1 403.gcc



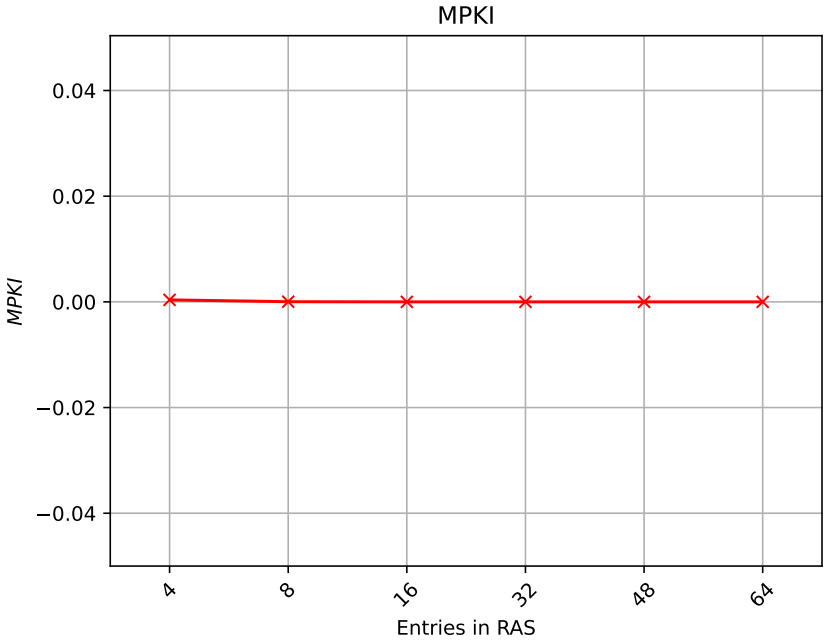
4.4.2 429.mcf



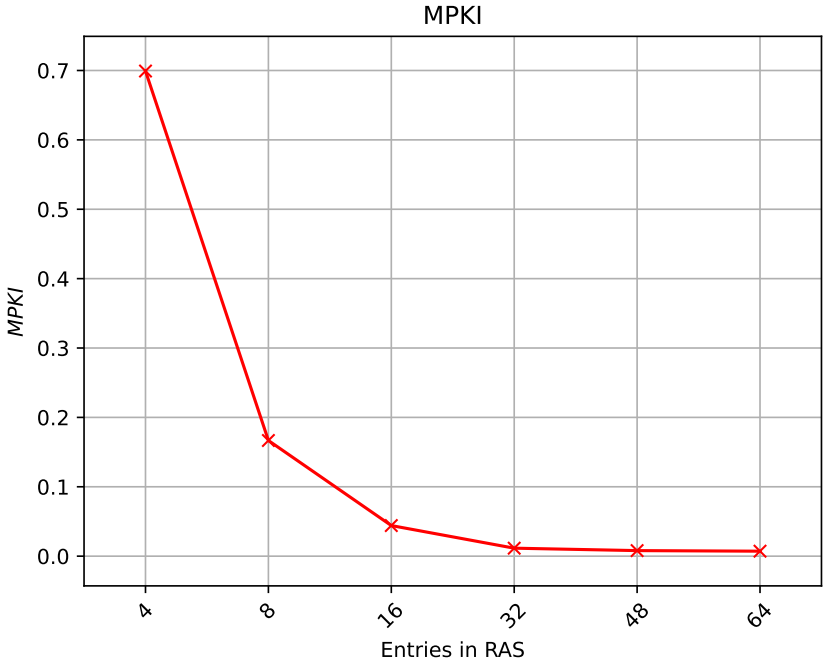
4.4.3 434.zeusmp



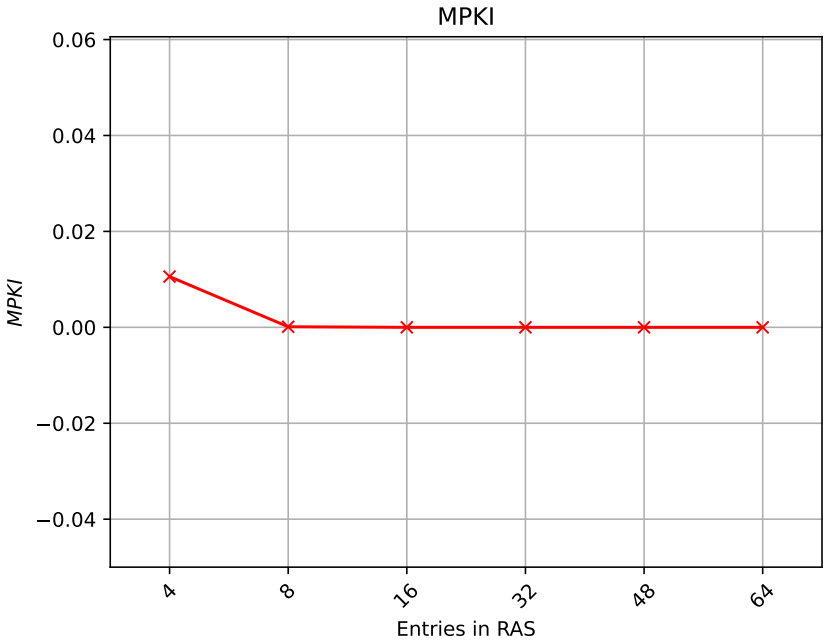
4.4.4 436.cactusADM



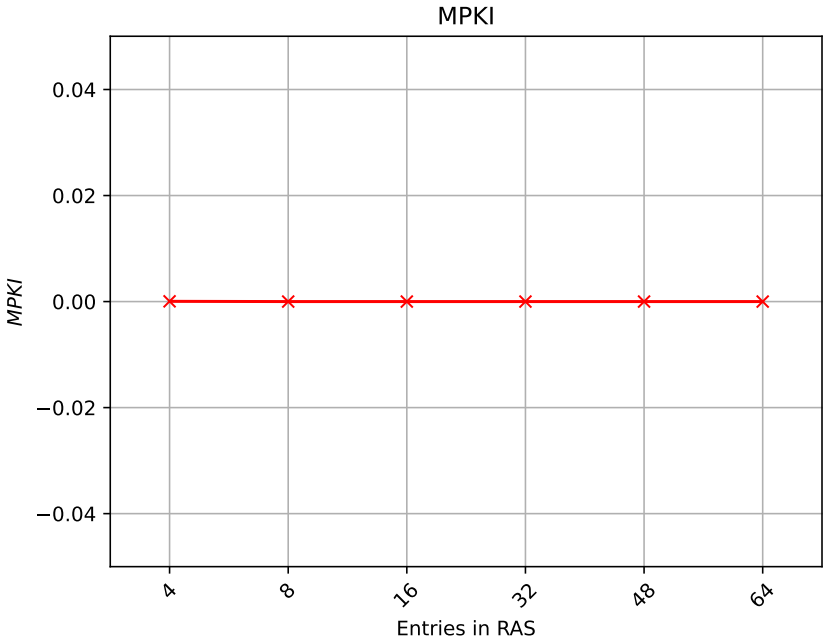
4.4.5 445.gobmk



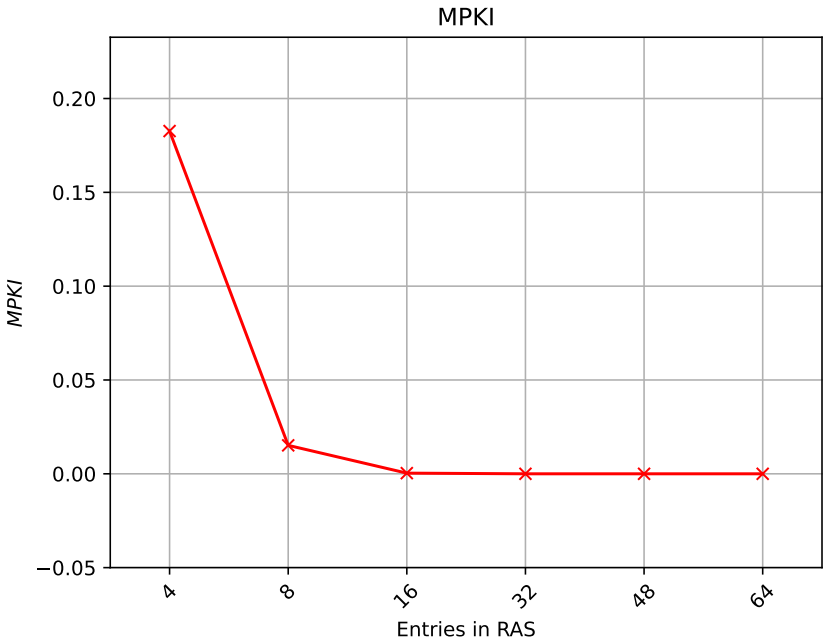
4.4.6 450.soplex



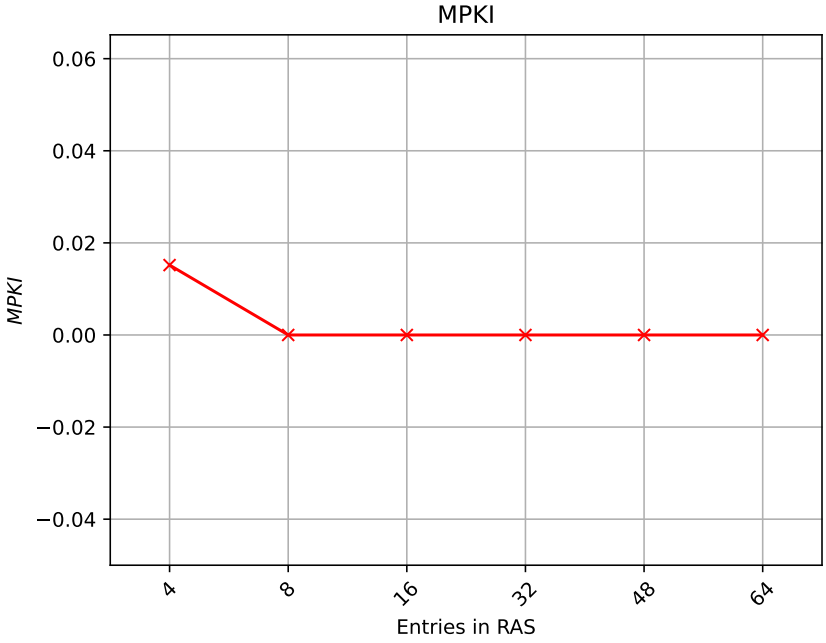
4.4.7 456.hmmer



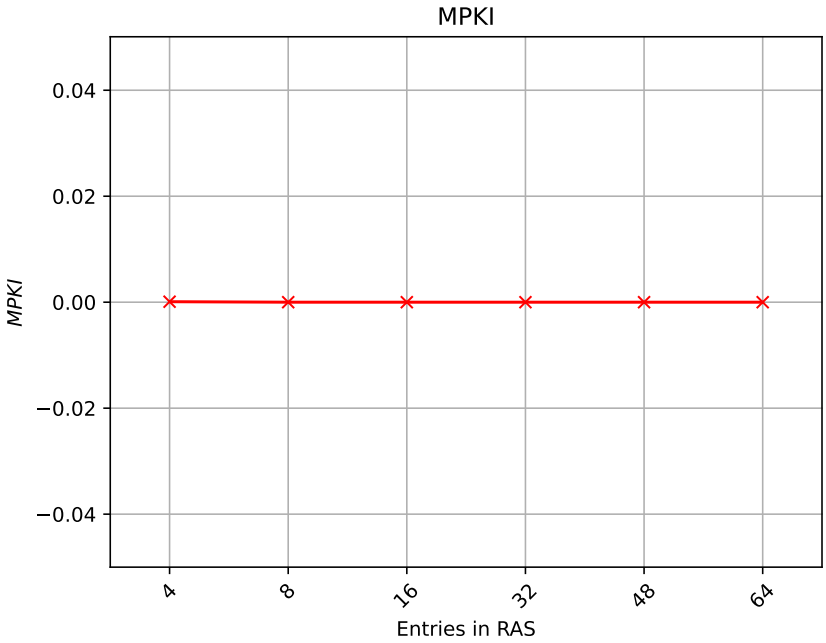
4.4.8 458.sjeng



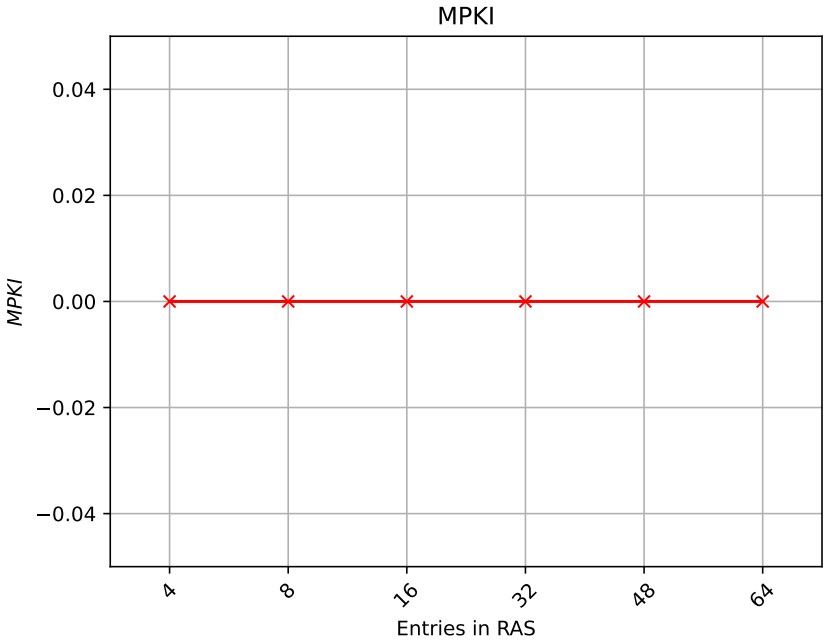
4.4.9 459.GemsFDTD



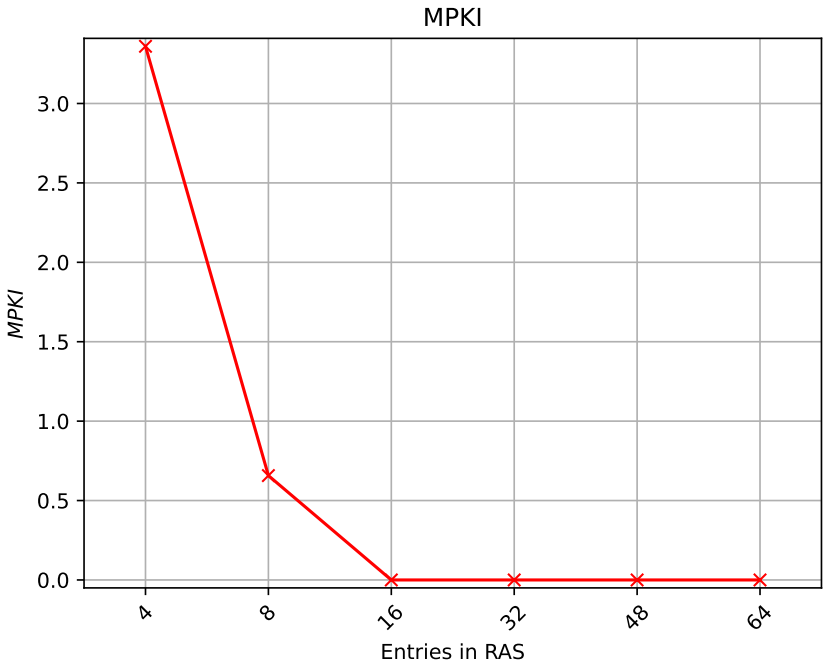
4.4.10 462.libquantum



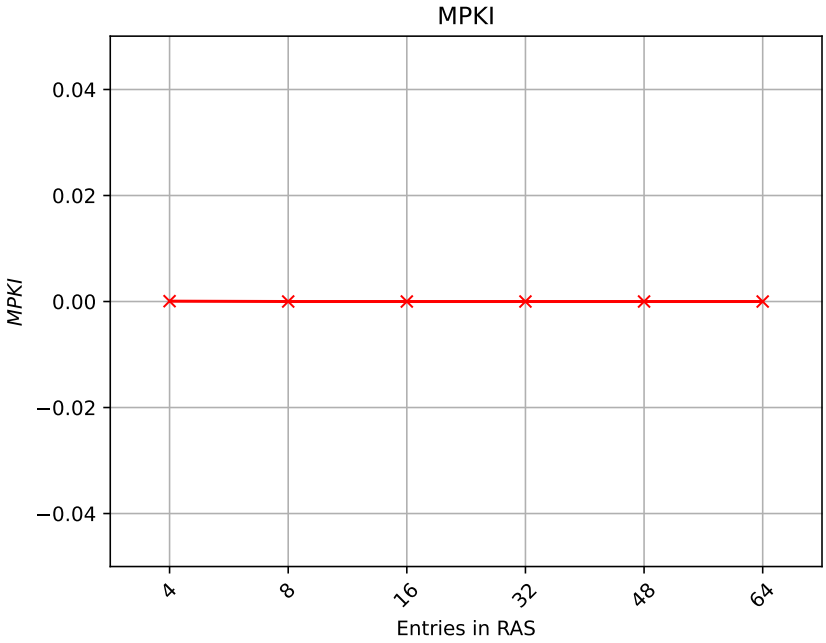
4.4.11 470.lbm



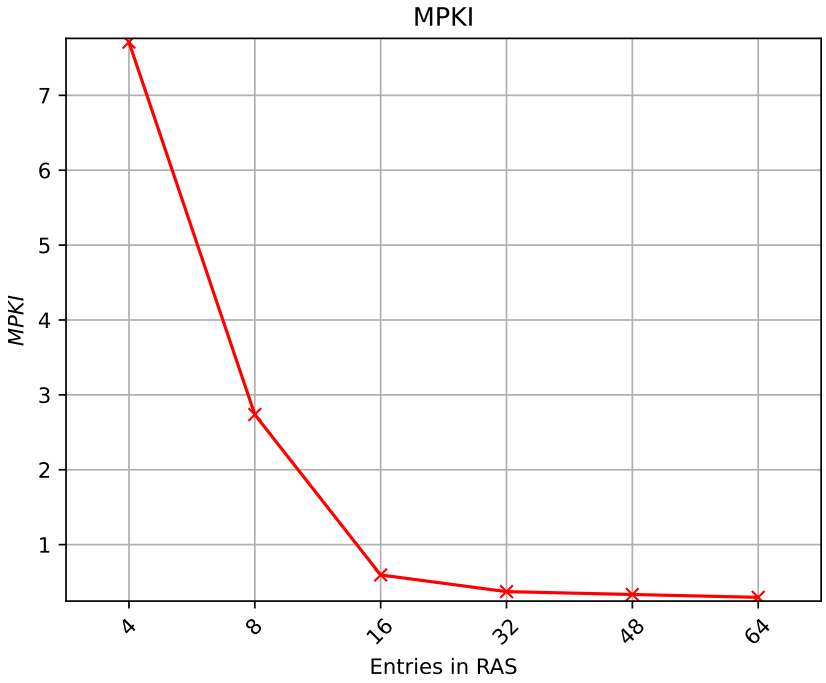
4.4.12 471.omnetpp



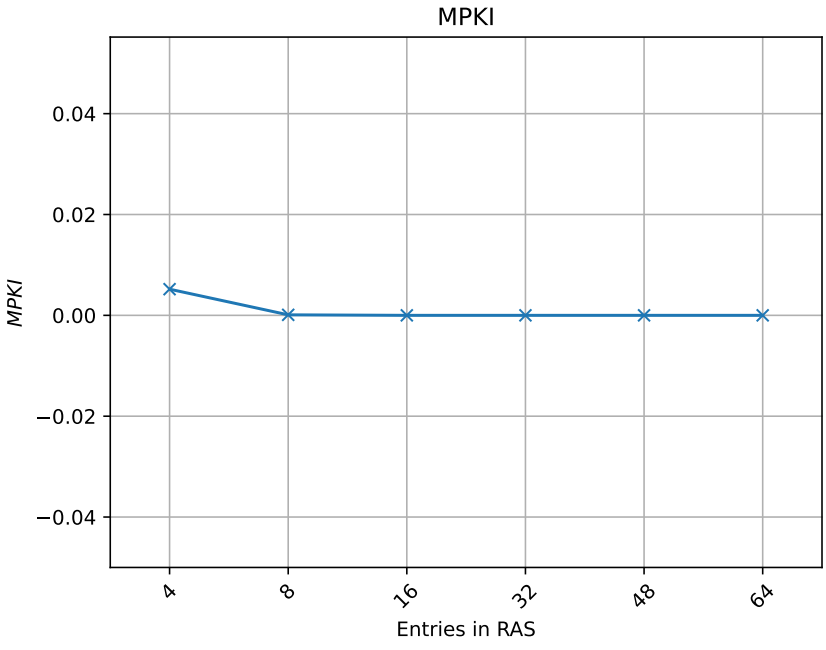
4.4.13 473.astar



4.4.14 483.xalancbmk



4.4.15 Γεωμετρικός μέσος των benchmarks



4.4.16 Παρατηρήσεις

Σε όλα τα benchmarks, για 4 entries παρατηρείται πολύ μεγαλύτερο MPKI σε σχέση με τις μετέτειτα τιμές. Με αύξηση των entries πάντα το MPKI μειώνεται. Στις περισσότερες περιπτώσεις,

μετά από 8 entries, δεν υπάρχει ουσιαστική αλλαγή στις επιδόσεις, αφού το MPKI έχει ουσιαστικά εκμηδενιστεί. Το γεγονός αυτό επηρεάζει και την επιλογή του βέλτιστου αριθμού entries, αφού θέλουμε να έχουμε τις καλύτερες επιδόσεις, με το ελάχιστο δυνατό υλικό. Επομένως, αφού από 16 entries και έπειτα έχει γίνει ήδη πολύ καλή βελτίωση της επίδοσης, επιλέγουμε αριθμό entries ίσο με **16**, και δεν κάνουμε overengineering στην RAS.

4.5 Σύγκριση διαφορετικών predictors

Σε αυτό το κομμάτι θα συγκρίνουμε τους παρακάτω predictors (όσοι είναι με **bold** δεν δίνονται και πρέπει να υλοποιηθούν)

1. **Static AlwaysTaken**
2. **Static BTFNT (BackwardTaken-ForwardNotTaken)**
3. Ο N-bit predictor που επιλέξαμε στο 4.2 (ii).
   Σημείωση: Λόγω λάθους κατά την εκτέλεση του ερωτήματος 4.2 (ii), αρχικά φάνηκε ότι ο Nbit-8K-4bit predictor απέδιδε καλύτερα, οπότε επιλέξαμε αυτόν για το παρόν ερώτημα. Τελικά δεν έφτασε ο χρόνος για να ξαναεκτελεστούν τα benchmarks και να διορθωθεί το λάθος, οπότε στα παρακάτω διαγράμματα θα φαίνεται ο **Nbit-8K-4bit**.
4. Pentium-M predictor (hardware overhead: $\approx$30K)
5. **Local-History two-level predictor #1** με τα εξής χαρακτηριστικά:
   - PHT entries = 8K
   - PHT N-bit counter length = 2
   - BHT entries = 4K
   - BHT entry length = 4 (ώστε το απαιτούμενο hardware να είναι ίσο με 32K)
6. **Local-History two-level predictor #2** με τα εξής χαρακτηριστικά:
   - PHT entries = 8K
   - PHT N-bit counter length = 2
   - BHT entries = 8K
   - BHT entry length = 2 (ώστε το απαιτούμενο hardware να είναι ίσο με 32K)
7. **Global History two-level predictor #1** με τα εξής χαρακτηριστικά:
   - PHT entries = 8K (ώστε το απαιτούμενο hardware να είναι ίσο με 32K)
   - PHT N-bit counter length = 2
   - BHR length = 4
8. **Global History two-level predictor #2** με τα εξής χαρακτηριστικά:
   - PHT entries = 16K (ώστε το απαιτούμενο hardware να είναι ίσο με 32K)
   - PHT N-bit counter length = 2
   - BHR length = 8

9. **Global History two-level predictor #3** με τα εξής χαρακτηριστικά:
    - PHT entries = 8K (ώστε το απαιτούμενο hardware να είναι ίσο με 32K)
    - PHT N-bit counter length = 4
    - BHR length = 4
10. **Global History two-level predictor #4** με τα εξής χαρακτηριστικά:
    - PHT entries = 8K (ώστε το απαιτούμενο hardware να είναι ίσο με 32K)
    - PHT N-bit counter length = 4
    - BHR length = 8
11. **Alpha 21264 predictor** (hardware overhead: $\approx$29K)
12. **Tournament Hybrid predictor #1** με τα εξής χαρακτηριστικά:
    - Ο meta-predictor M είναι ένας 2-bit predictor με 1024 entries
    - Ο P_0 είναι ένας **N-bit predictor** με τα εξής χαρακτηριστικά:
        - BHT entries = 8K
        - N = 2
    - Ο P_1 είναι επίσης ένας **N-bit predictor** με τα εξής χαρακτηριστικά:
        - BHT entries = 4K
        - N = 4
    - Οι P_0, P_1 έχουν overhead 16K ο καθένας
13. **Tournament Hybrid predictor #2** με τα εξής χαρακτηριστικά:
    - Ο meta-predictor M είναι ένας 2-bit predictor με 2048 entries
    - Ο P_0 είναι ένας **local-history predictor** με τα εξής χαρακτηριστικά:
        - PHT entries = 4K
        - PHT N-bit counter length = 2 bit
        - BHT entries = 4K
        - BHT entry length = 2 bit
    - Ο P_1 είναι ένας **N-bit predictor** με τα εξής χαρακτηριστικά:
        - BHT entries = 8K
        - N = 2
    - Οι P_0, P_1 έχουν overhead 16K ο καθένας
14. **Tournament Hybrid predictor #3** με τα εξής χαρακτηριστικά:
    - Ο meta-predictor M είναι ένας 2-bit predictor με 1024 entries
    - Ο P_0 είναι ένας **global-history predictor** με τα εξής χαρακτηριστικά:
        - PHT entries = 8K
        - PHT N-bit counter length = 2
        - BHR length = 8

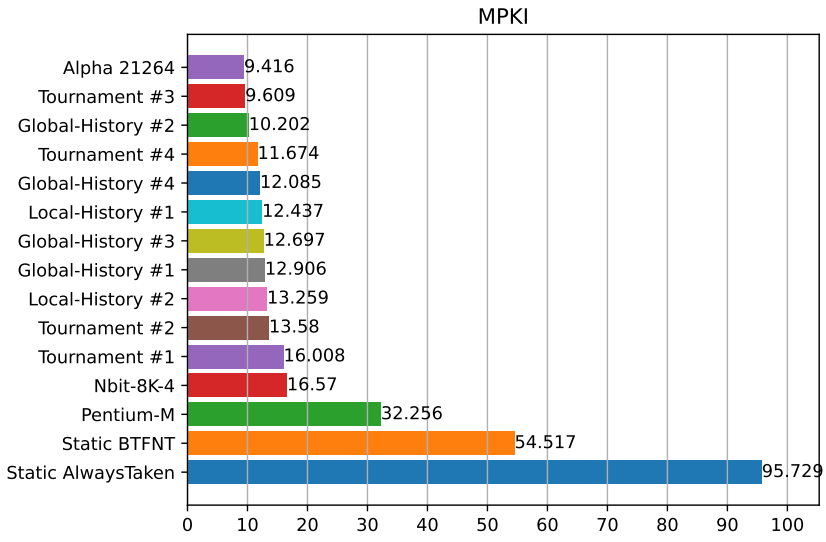
- Ο P_1 είναι ένας **N-bit predictor** με τα εξής χαρακτηριστικά:
  - BHT entries = 4K
  - N = 4
- Οι P_0, P_1 έχουν overhead 16K ο καθένας

15. **Tournament Hybrid predictor #4** με τα εξής χαρακτηριστικά:

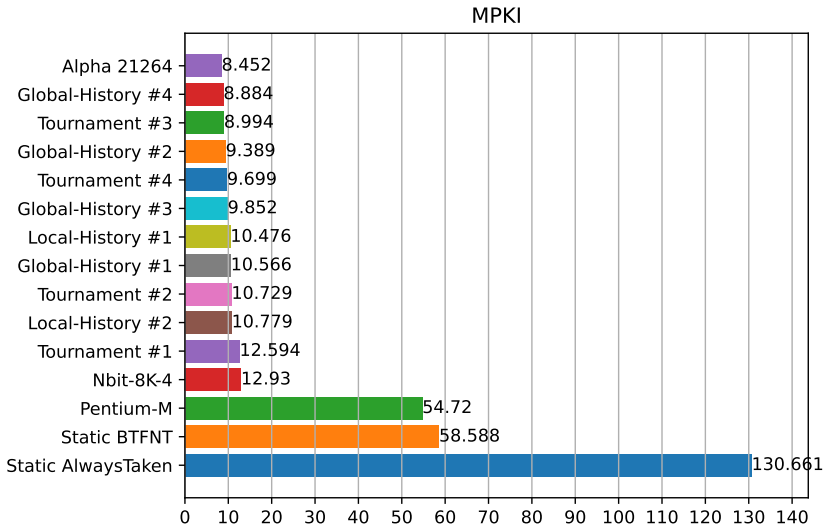
- Ο meta-predictor M είναι ένας 2-bit predictor με 2048 entries
- Ο P_0 είναι ένας **local-history predictor** με τα εξής χαρακτηριστικά:
  - PHT entries = 8K
  - PHT N-bit counter length = 1
  - BHT entries = 4K
  - BHT entry length = 2
- Ο P_1 είναι ένας **global-history predictor** με τα εξής χαρακτηριστικά:
  - PHT entries = 4K
  - PHT N-bit counter length = 4
  - BHR length = 4
- Οι P_0, P_1 έχουν overhead 16K ο καθένας

Έχουμε:

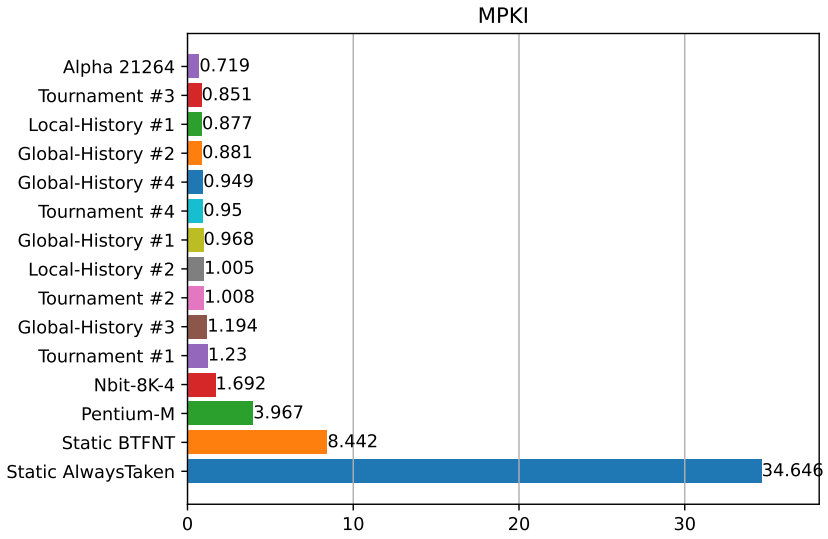
4.5.1 403.gcc



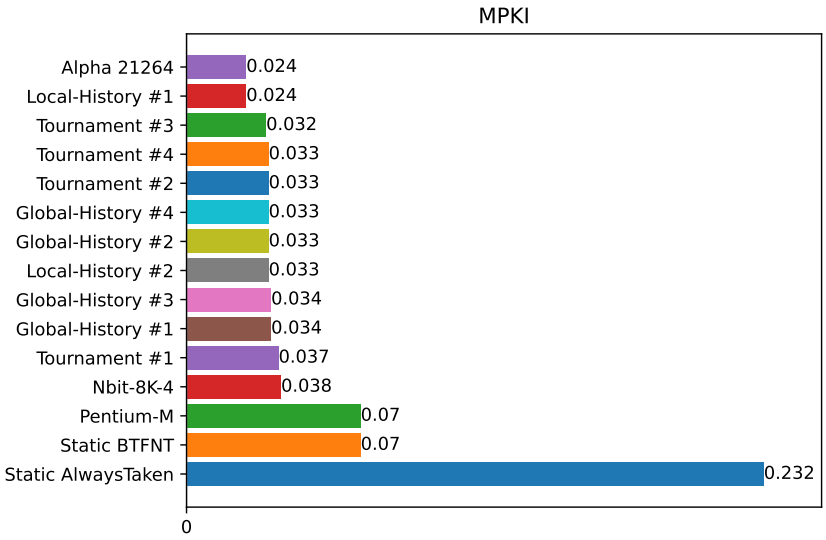
4.5.2 429.mcf



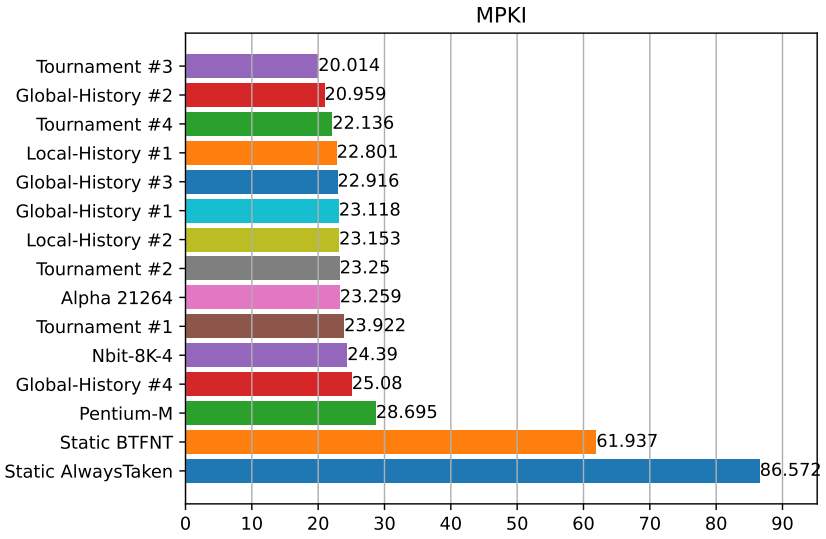
4.5.3 434.zeusmp



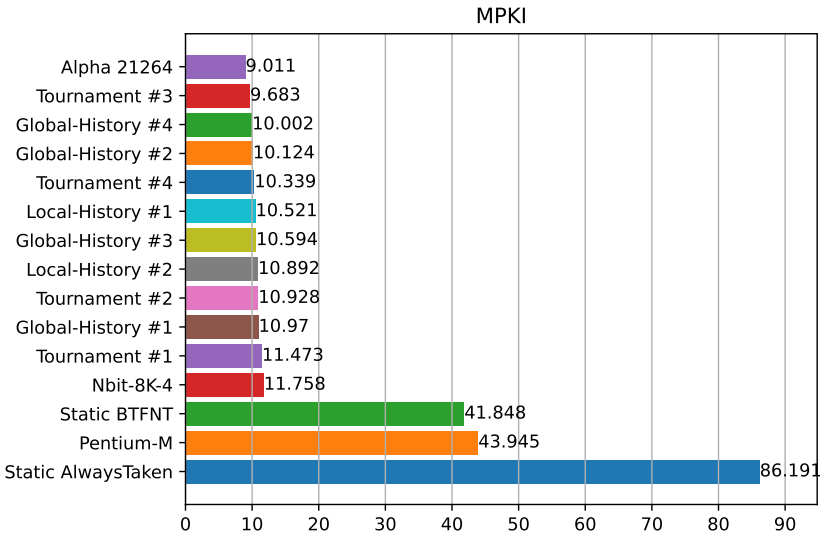
4.5.4 436.cactusADM



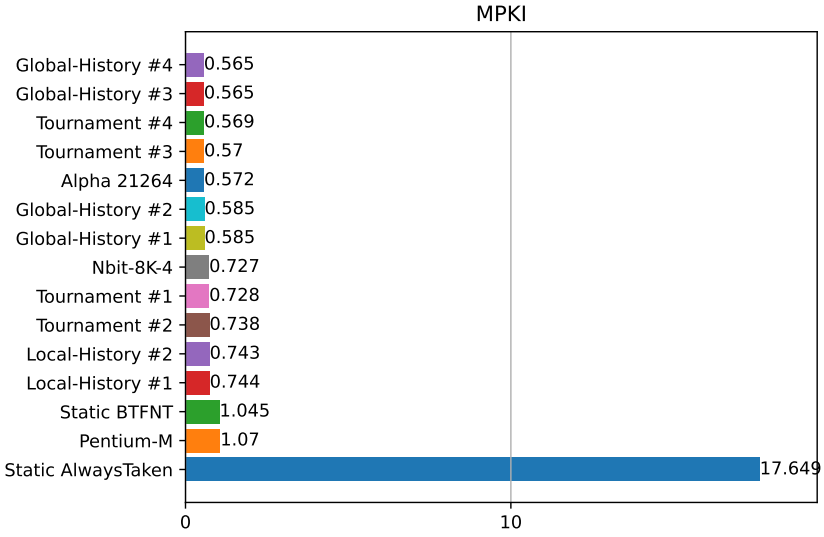
4.5.5 445.gobmk



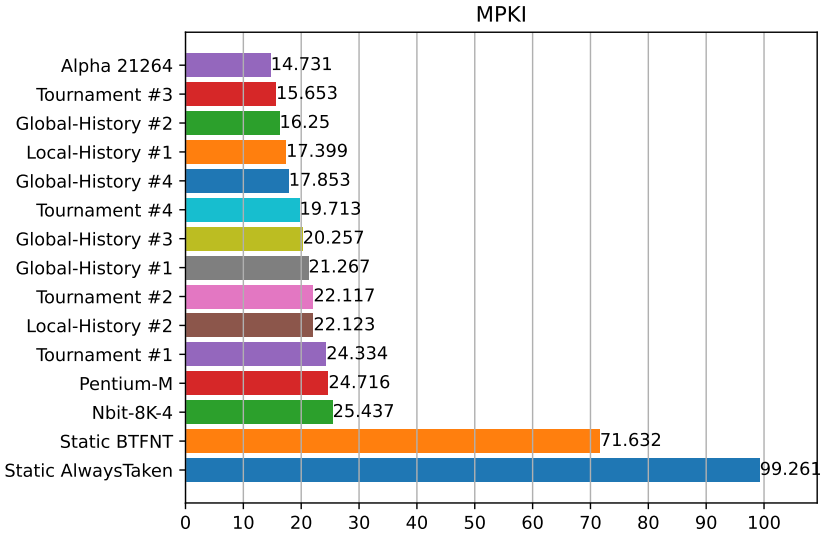
4.5.6 450.soplex



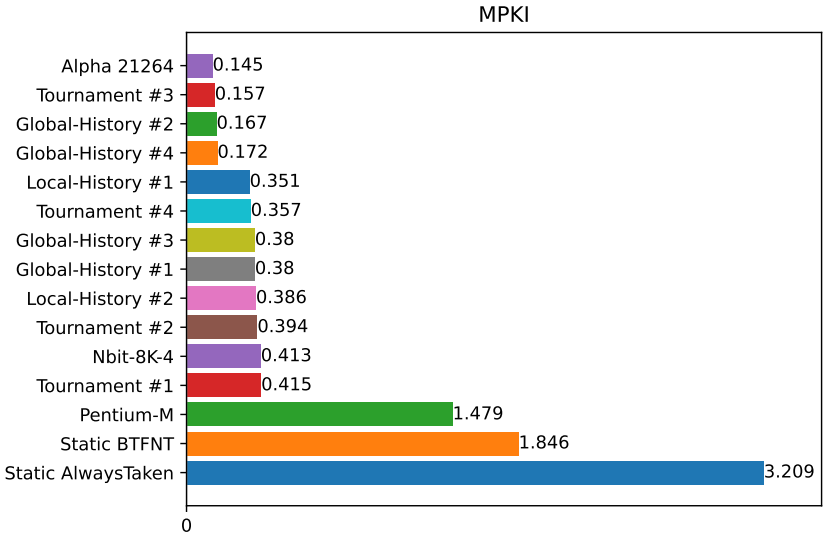
4.5.7 456.hmmer



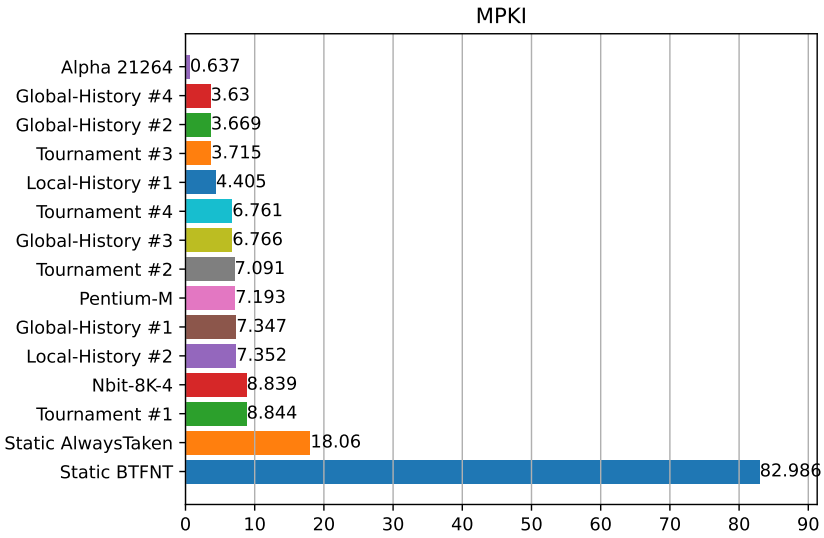
4.5.8 458.sjeng



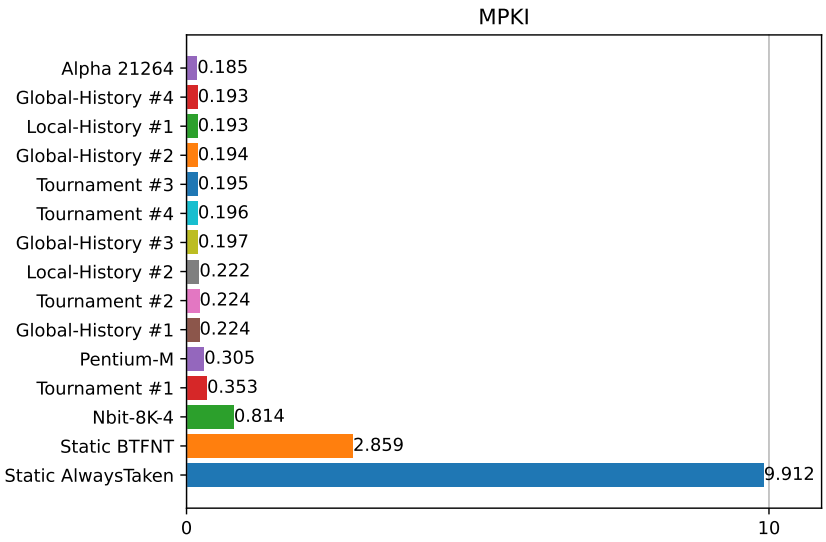
4.5.9 459.GemsFDTD



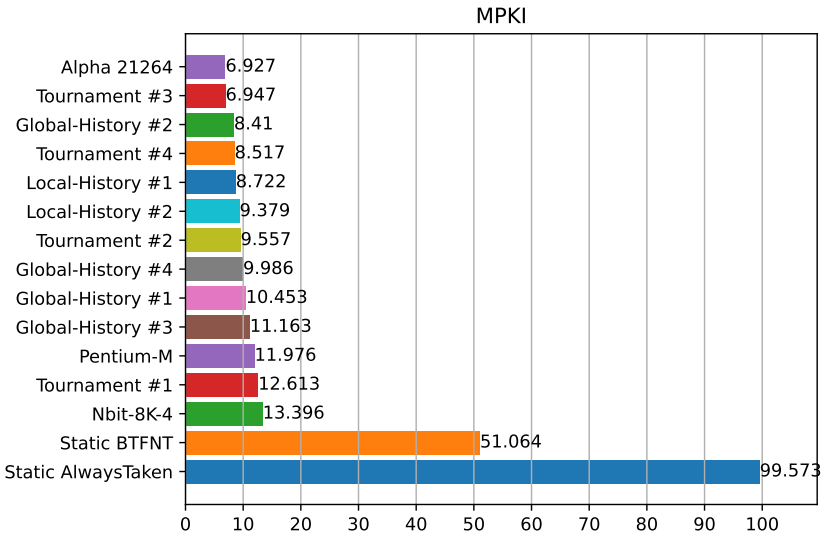
4.5.10 462.libquantum



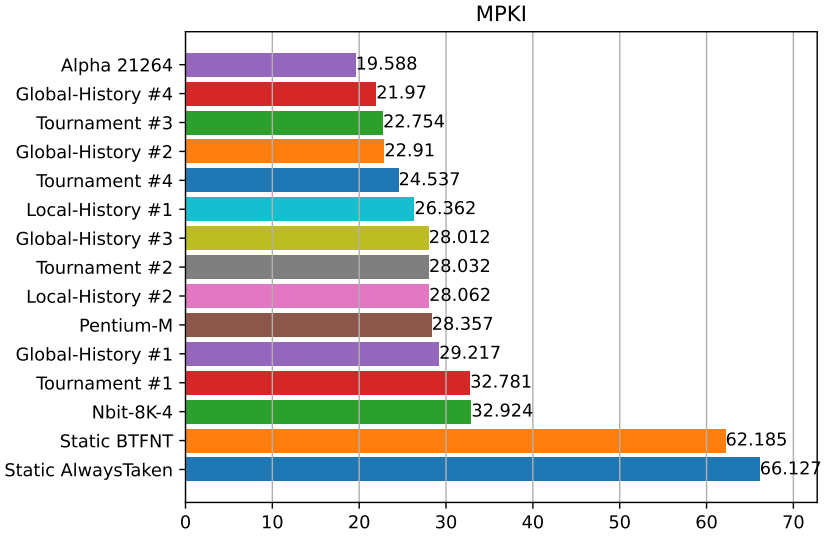
4.5.11 470.lbm



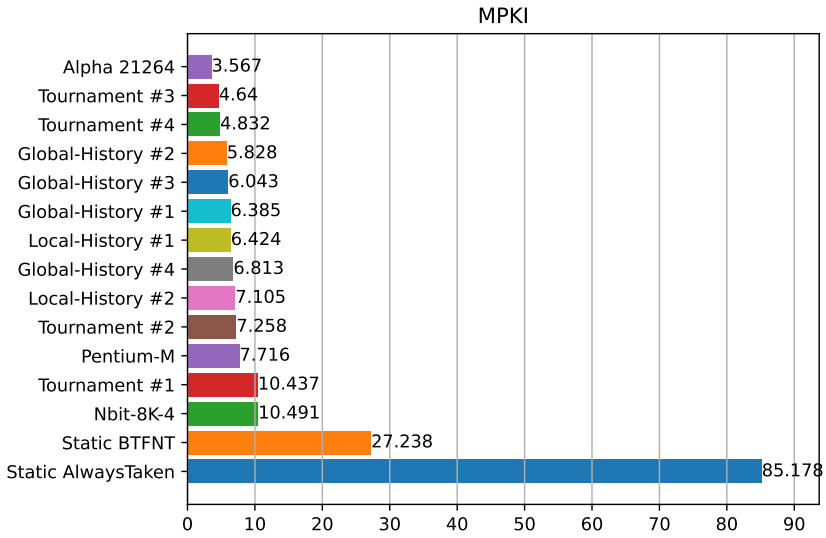
4.5.12 471.omnetpp



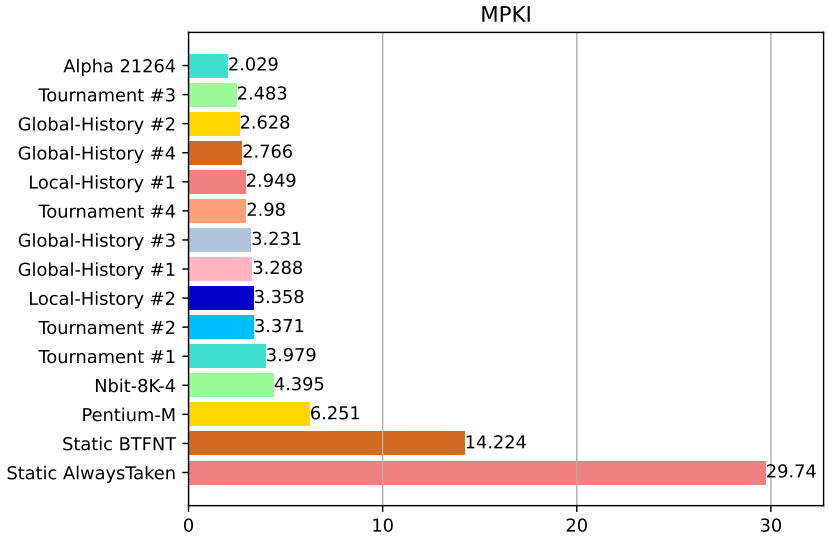
4.5.13 473.astar



4.5.14 483.xalancbmk



4.5.15 Γεωμετρικός μέσος των benchmarks



4.5.16 Παρατηρήσεις

Όπως είναι λογικό, ο Static AlwaysTaken predictor είναι ο χειρότερος από όλους, ενώ ακολουθεί ο Static BTFNT, καθώς αυτοί οι δύο predictors δεν διαθέτουν καθόλου "ευφυΐα". Ύστερα ακολουθεί ο Pentium-M predictor, ενώ οι υπόλοιποι predictors βρίσκονται περίπου στην ίδια τάξη επίδοσης (πχ 9-14% στο `403.gcc`). Σε όλα τα benchmarks η επίδοση του Alpha predictor είναι η καλύτερη, με εξαίρεση το `445.gobmk`, όπου καλύτερος είναι ο Tournament predictor #3 (όπως ορίστηκε παραπάνω). Αυτό έχει άμεση συνέπεια ο Alpha predictor να είναι ο καλύτερος και όσον αφορά τον γεωμετρικό μέσο όρο. Αυτό σημαίνει ότι πρόκειται και για την καλύτερη επιλογή, εκτός κι αν πραγματικά χρειαζόμαστε τις βέλτιστες επιδόσεις για μία πολύ συγκεκριμένη λειτουργία, η οποία μοιάζει πολύ με το `445.gobmk`. Σε αυτή την περίπτωση, αξίζει να ελέγξουμε και τις

επιλογές των Tournament #3 και Global-History #2.

Τελικά, κατά μέσο όρο, η κατάταξη των predictors έχει ως εξής:

1. Alpha 21264
2. Tournament #3
3. Global-History #2
4. Global-History #4
5. Local-History #1
6. Tournament #4
7. Global-History #3
8. Global-History #1
9. Local-History #2
10. Tournament #2
11. Tournament #1
12. Nbit-8K-4
13. Pentium-M
14. Static BTFNT
15. Static AlwaysTaken