Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Πανεπιστήμιο Θεσσαλίας

Διπλωματική Εργασία

“ Τεχνικές παραλληλοποίησης συμπίεσης HEVC video ”
-
“ Parallelizing techniques for video coding in HEVC ”

Οικονόμου Φωτεινή

Επιβλέπων:
Σταμούλης Γεώργιος
Συνεπιβλέπων:
Λουκόπουλος Αθανάσιος

Βόλος , Ιούλιος 2017

Στην οικογένεια και τους φίλους μου

Περίληψη

Το πρότυπο High Efficiency Video Coding (HEVC) είναι ένα έργο που πραγματοποιήθηκε από κοινού από τους οργανισμούς τυποποίησης ITU-T Video Coding Experts Group (VCEG) και τη ISO/IEC Moving Picture Experts Group (MPEG), και εργάζονται μαζί στην εταιρία Joint Collaborative Team on Video Coding (JCT-VC). Τα πρότυπα κωδικοποίησης βίντεο έχουν εξελιχθεί κυρίως μέσω της ανάπτυξης των γνωστών προτύπων ITU-T και ISO/IEC. Η ITU-T παρήγαγε το H.261 και H.263 και η ISO/IEC παρήγαγε MPEG-1 και MPEG-4 Visual, και από κοινού οι δυο οργανώσεις παρήγαγαν τα πρότυπα H.262/MPEG-2 Video και H.264/MPEG-4 Advanced Video Coding (AVC).

Το HEVC έχει σχεδιαστεί για να αντιμετωπίσει ουσιαστικά όλες τις υπάρχουσες εφαρμογές του H.264/MPEG-4 AVC και εστιάζει ιδιαίτερα σε δυο βασικά ζητήματα: στην αυξημένη ανάλυση του βίντεο και στην αύξηση της χρήσης της αρχιτεκτονικής παράλληλης επεξεργασίας.

Επικεντρωνόμαστε στον πηγαίο κώδικα που παρέχεται για πειραματικές δοκιμές αναλύοντας τη δομή και τη σύνταξη του bitstream. Το λογισμικό αναφοράς περιλαμβάνει τις λειτουργίες κωδικοποίησης και αποκωδικοποίησης και είναι γραμμένος σε C++.

Παρέχεται μια περιγραφή της διαδικασίας κωδικοποίησης του HEVC, χρησιμοποιώντας για οδηγό το λογισμικό αναφοράς Test Model HM 16. Μέσω της κατανόησης των αλγορίθμων και των μεθόδων κωδικοποίησης που υποστηρίζονται στο λογισμικό HM εκτιμάται η υπολογιστική τους πολυπλοκότητα σε χρόνο με σκοπό να βελτιωθεί ο χρόνος εκτέλεσης κατά τη διαδικασία κωδικοποίησης των διάφορων ακολουθιών βίντεο. Αφού εντοπιστούν οι χρονοβόρες διαδικασίες της κωδικοποίησης προτείνονται τροποποιήσεις αυτών και ακολουθούν πειράματα.

Abstract

The High Efficiency Video Coding (HEVC) is a video project of the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG) standardization organizations, working together as the Joint Collaborative Team on Video Coding (JCT-VC).Video coding standards have evolved primarily through the development of the well-known ITU-T and ISO/IEC standards. H.261 and H.263 ware produced by the ITU-T , MPEG-1 and MPEG-4 were produced by Visual ISO/IEC. The two organizations together produced the H.262/MPEG-2 Video and H.264/MPEG-4 Advanced Video Coding (AVC) standards.

HEVC has been designed to address applications of H.264/MPEG-4 AVC and to focus on two key issues: increased video resolution and increased use of parallel processing architectures.

We focus on the reference code that is being provided for experimental testing, by analysing the structure and syntax of the bitstream. The reference code ,which is developed in C++,includes coding and decoding functions.

A description of the decoding procedure of the HEVC is included with the use of the reference code Test Model HM16 as a guide. Through the understanding of the algorithms and the coding methods of the HM code, their calculating complexity in terms of time is being evaluated, so that the time needed for the coding of the video sequences will be optimized. After detecting the time-consuming coding procedures, some modifications are suggested and then the experimental procedures follow.

Περιεχόμενα

Κατάλογος Σχημάτων

1. Εισαγωγή

Το πρότυπο HEVC σχεδιάστηκε για να πετύχει πολλαπλούς στόχους συμπεριλαμβάνοντας την αποδοτικότερη κωδικοποίηση, την ευκολία μεταφοράς του ως ένα ολοκληρωμένο σύστημα και την ανθεκτικότητα του στην απώλεια των δεδομένων, καθώς και την δυνατότητα εκτέλεσης χρησιμοποιώντας την παράλληλη αρχιτεκτονική επεξεργασίας.

Το πρότυπο HEVC έχει βρει εφαρμογή σε ένα ευρύ φάσμα προϊόντων, τα οποία είναι διαδεδομένα στην καθημερινότητα μας. Σε όλη αυτή την εξέλιξη, συνεχίστηκαν οι προσπάθειες για να μεγιστοποιηθεί η συμπίεση και να βελτιωθούν τα χαρακτηριστικά ,όπως η αξιοπιστία της απώλειας δεδομένων συνυπολογίζοντας κάθε φορά τους υπολογιστικούς πόρους που πρακτικά μπορούσαν να χρησιμοποιηθούν στα προϊόντα κατά την διάρκεια ανάπτυξης του κάθε προτύπου.

Το πρότυπο που προηγείται του έργου HEVC ήταν το H.264/MPEG-4 AVC, το οποίο αρχικά αναπτύχθηκε το 1999-2003. Το πρότυπα αυτά χρησιμοποιούνται ευρέως για πολλές εφαρμογές, όπως είναι η μετάδοση υψηλής ανάλυσης (HD), βιντεοκάμερες, εφαρμογές ασφαλείας, βίντεο μέσω Internet, δίσκους Blu-ray, εφαρμογές άμεσης συνομιλίας όπως το chat, και βίντεο κλήσεις. Ωστόσο, η αυξανόμενη ποικιλομορφία των υπηρεσιών, η αυξανόμενη δημοτικότητα του HD βίντεο και η εμφάνιση και άλλων μορφών ανάλυσης HD (όπως 4k×2k ή 8k×4k ανάλυση) δημιουργούν ακόμα μεγαλύτερη ανάγκη για απόδοση κωδικοποίησης ανώτερη των δυνατοτήτων της H.264/MPEG-4 AVC.

Το HEVC χρησιμοποιεί την ίδια υβριδική προσέγγιση (inter-/intrapicture πρόβλεψη) που χρησιμοποιείται σε όλα τα πρότυπα συμπίεσης βίντεο από το H.261. Ένας αλγόριθμος κωδικοποίησης που παράγει το HEVC bitstream περιγράφεται ως εξής: κάθε εικόνα είναι χωρισμένη σε περιοχές από σχηματισμένα μπλοκ, και κάθε μπλοκ ακριβώς μεταφέρεται στον αποκωδικοποιητή. Η πρώτη εικόνα μίας ακολουθίας βίντεο (και η πρώτη εικόνα σε μια τυχαία στιγμή μιας ακολουθίας βίντεο) κωδικοποιείται χρησιμοποιώντας μόνο intrapicture πρόβλεψη (αυτή χρησιμοποιεί μια πρόβλεψη δεδομένων χωρικά από περιοχή σε περιοχή μέσα στην ίδια εικόνα αλλά δεν έχει εξάρτηση από άλλες εικόνες) . Για όλες τις υπόλοιπες εικόνες (pictures) της ακολουθίας ή μεταξύ τυχαίων σημείων της προσπέλασης, η interpicture χρονική λειτουργία πρόβλεψης χρησιμοποιείται συνήθως για τα περισσότερα μπλοκ. Η διαδικασία κωδικοποίησης για την interpicture πρόβλεψη αποτελείται από την επιλογή δεδομένων κίνησης που περιλαμβάνει την επιλεγμένη εικόνα αναφοράς και το διάνυσμα κίνησης (MV) που εφαρμόζεται για την πρόβλεψη των δειγμάτων του κάθε μπλοκ. Ο κωδικοποιητής και ο αποκωδικοποιητής παράγουν όμοια interpicture πρόβλεψη εφαρμόζοντας αντιστάθμιση κίνησης (motion compensation) (MC) χρησιμοποιώντας το MV.

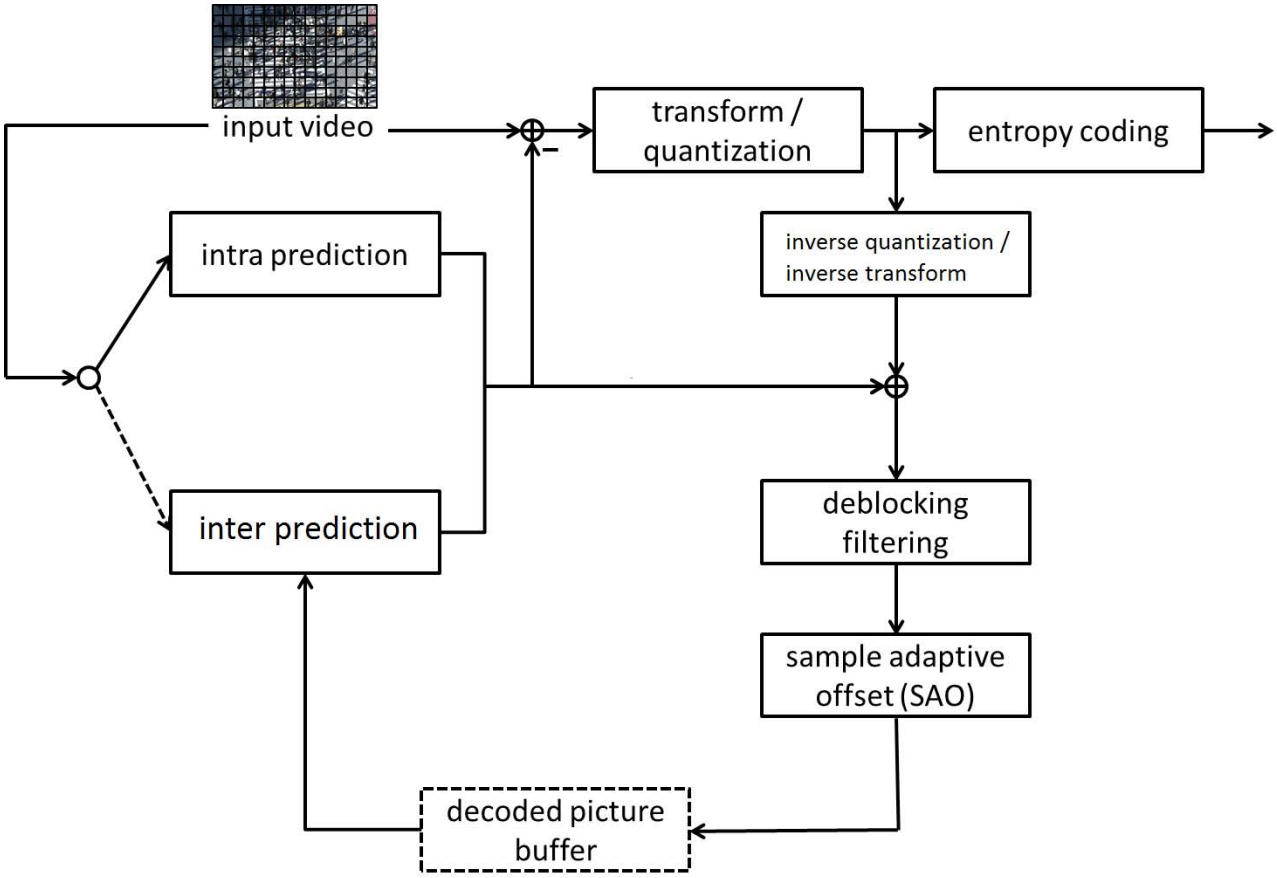
Το λογισμικό αναφοράς που χρησιμοποιείται είναι το HEVC test Model. Ο σκοπός έγκειται στο να βοηθά τους χρήστες ενός προτύπου κωδικοποίησης βίντεο επιτρέποντας την δημιουργία, τη δοκιμή, καθώς επίσης, αναλαμβάνει την εκπαίδευσή των χρηστών επιδεικνύοντας τις δυνατότητές του. Ένας από τους κύριους στόχους του είναι να δώσει μια βάση πάνω στην οποία θα διεξάγονται πειράματα προκειμένου να προσδιοριστεί ποια εργαλεία κωδικοποίησης παρέχουν την επιθυμητή απόδοση κωδικοποίησης.

Στο κεφάλαιο 2 θα αναλύσουμε τα στοιχεία σύνταξης του προτύπου HEVC, τις δομές δεδομένων και την ανάλυση του bitstream. Στο κεφάλαιο 3 θα αναφερθούν οι τεχνικές που εφαρμόζονται στην κωδικοποίηση και αποκωδικοποίηση των ακολουθιών βίντεο. Τέλος στο κεφάλαιο 4 θα αναφερθούμε στις συναρτήσεις κόστους, στον τρόπο βελτίωσης τους και την υλοποίηση των πειραμάτων.

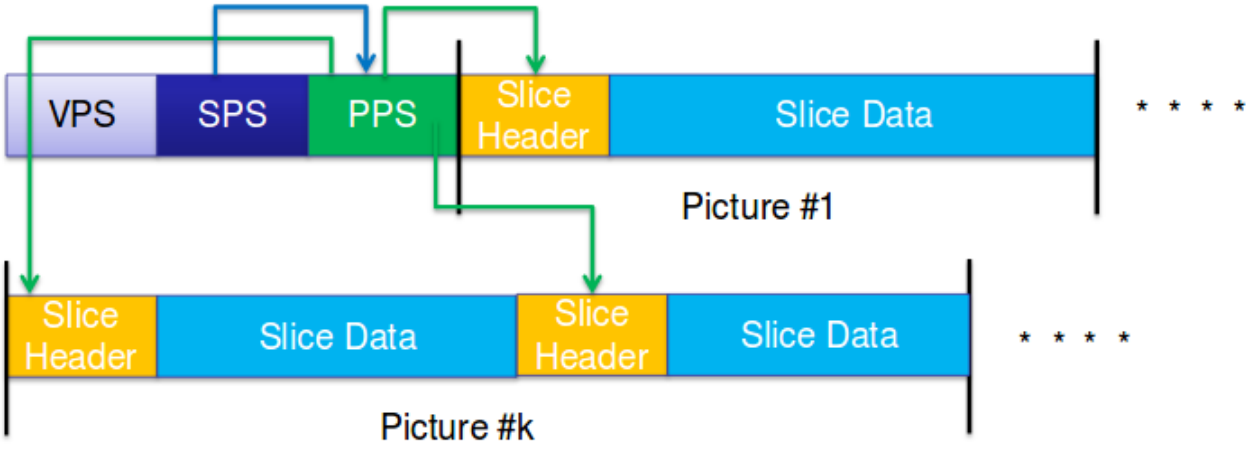
2. Στοιχεία συντακτικού HEVC

2.1 Εισαγωγή

Το HEVC είναι μια αρχιτεκτονική υβριδικής κωδικοποίησης βασισμένη σε μπλοκ, που συνδυάζει την inter και την intra πρόβλεψη και την κωδικοποίηση μετασχηματισμού με κωδικοποίηση εντροπίας υψηλής απόδοσης. Σε αντίθεση με τα προηγούμενα πρότυπα κωδικοποίησης βίντεο, το HEVC χρησιμοποιεί μια δομή διαχωρισμού μπλοκ κωδικοποίησης δέντρου τετραγώνων (quad-tree coding), η οποία επιτρέπει την ευέλικτη χρήση μεγάλων και μικρών μπλοκ κωδικοποίησης, πρόγνωσης και μετασχηματισμού. Το HEVC επιτρέπει, επίσης, βελτιωμένη intra πρόβλεψη και κωδικοποίηση, την πρόβλεψη και κωδικοποίηση παραμέτρων προσαρμοσμένης κίνησης (adaptive motion parameter), ένα νέο φίλτρο βρόχου και μία βελτιωμένη έκδοση κωδικοποίησης εντροπίας, την κωδικοποίηση δυαδικής αριθμητικής προσαρμοσμένου πλαισίου (CABAC). Επίσης, σχεδιάστηκαν νέες δομές υψηλού επιπέδου για να βοηθήσουν την παράλληλη επεξεργασία.



Σχήμα 1: Διάγραμμα HM encoder



Σχήμα 2: Δομή Bitsream

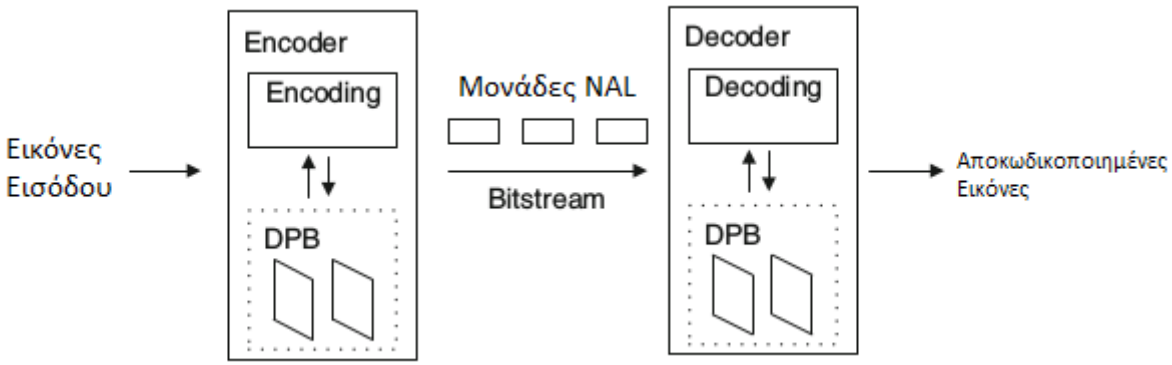
2.2 Δομή συνόλου παραμέτρων (Parameter set structure)

Τα σύνολα παραμέτρων περιέχουν πληροφορίες που σχετίζονται με την αποκωδικοποίηση των διάφορων περιοχών του βίντεο. Η δομή των συνόλων παραμέτρων παρέχει ένα ισχυρό μηχανισμό για τη μεταφορά δεδομένων που είναι απαραίτητες για την διαδικασία αποκωδικοποίησης. Υπάρχει το σύνολο παραμέτρων ακολουθίας (sequence parameter set) SPS, το σύνολο εικόνας (picture parameter set) PPS και μια νέα δομή συνόλου παραμέτρων βίντεο το (video parameter set) VPS, το οποίο δεν υπήρχε στο H.264/MPEG-4 AVC.

2.3 Η δομή σύνταξης της μονάδας NAL

Κάθε δομή σύνταξης (syntax structure) τοποθετείται σε ένα λογικό πακέτο δεδομένων που ονομάζεται μονάδα αφηρημένου στρώματος δικτύου (network abstraction layer) NAL.

Η σύνταξη υψηλού επιπέδου του HEVC περιέχει πολυάριθμα στοιχεία τα οποία έχει κληρονομήσει από το NAL του H.264/MPEG-4 AVC. Το NAL παρέχει τη δυνατότητα να σχεδιαστεί το στρώμα κωδικοποίησης βίντεο (video coding layer) VCL, και παρέχει ένα πλαίσιο (framework) για την ανθεκτικότητα στην απώλεια δεδομένων. Οι μονάδες NAL ταξινομούνται σε VCL και non-VCL μονάδες NAL ανάλογα με το αν περιέχουν κωδικοποιημένες εικόνες ή άλλα συσχετιζόμενα δεδομένα αντίστοιχα. Στο HEVC πρότυπο αρκετοί τύποι μονάδων VCL NAL περιλαμβάνονται προσδιορίζοντας κατηγορίες εικόνων για την προετοιμασία του αποκωδικοποιητή ή για την τυχαία πρόσβαση.



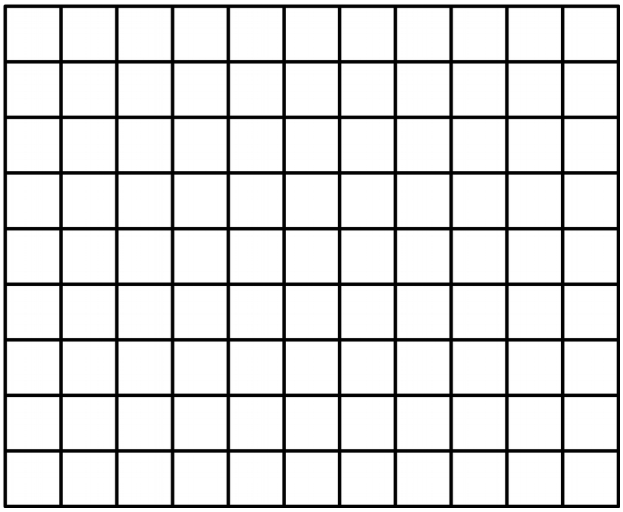
Σχήμα 3: Μονάδες NAL

2.4 Σύνταξη Εικόνας (Picture Syntax)

Η δομή διαχωρισμού της εικόνας διαιρεί το βίντεο εισόδου σε μπλοκ, που ονομάζονται μονάδες κωδικοποίησης δέντρων (coding tree units, CTUs). Αυτά τα CTUs έχουν τον ανάλογο ρόλο με εκείνο των macroblocks σε προηγούμενα πρότυπα. Ένα CTU χωρίζεται χρησιμοποιώντας την δομή δέντρου quad-tree σε μονάδες κωδικοποίησης (coding units, CUs) όπου κάθε CU ορίζει μια περιοχή που μοιράζεται τον ίδιο τρόπο πρόβλεψης (intra, inter ή skip). Το CU ορίζει το σχήμα των μονάδων πρόβλεψης (prediction units, PUs), όπου κάθε PU περιγράφει λεπτομερώς τις πληροφορίες πρόβλεψης που χρησιμοποιούνται για την αντίστοιχη περιοχή εικόνας. Τα CUs ορίζουν επίσης ένα άλλο quad-tree , το quad-tree διαφορών (residual-quad-tree, RQT) που περιέχει τις μονάδες μετασχηματισμού (transform units, TUs), που ορίζουν μια περιοχή που μοιράζεται την ίδια διαδικασία μετασχηματισμού και κβαντισμού. Ο όρος μονάδα ορίζει μια περιοχή μιας εικόνας που καλύπτει όλα τα συστατικά. Ο όρος μπλοκ χρησιμοποιείται για τον ορισμό μιας περιοχής που καλύπτει ένα συγκεκριμένο στοιχείο (π.χ luma).

2.4.1 Coding tree units (CTU)

Οι εικόνες διαιρούνται σε μία ακολουθία μονάδων κωδικοποίησης CTUs, όλες έχουν το ίδιο μέγεθος και κάθε μία καλύπτει μία τετραγωνική περιοχή pixel της εικόνας.



Σχήμα 4: Διαχωρισμός εικόνας σε CTUs

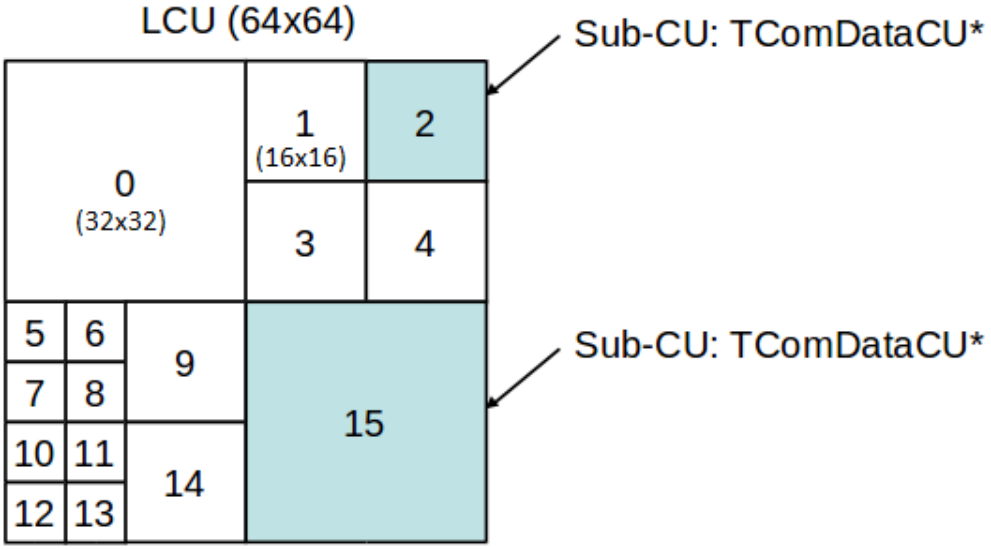
Ο πυρήνας του στρώματος κωδικοποίησης των προηγούμενων προτύπων ήταν τα macroblocks, που περιείχαν ένα 16x16 μπλοκ από luma δείγματα (luma=lumination) και στη συνήθη περίπτωση της 4:2:0 δειγματοληψίας, δύο αντιστοιχίσεις 8x8 μπλοκ των chroma δειγμάτων, ενώ η ανάλογη δομή στο HEVC είναι η μονάδα coding tree unit (CTU), η οποία έχει μέγεθος που επιλέγεται από τον κωδικοποιητή και μπορεί να είναι μεγαλύτερη από το παραδοσιακό macroblock. Η CTU αποτελείται από ένα luma CTB, τα αντίστοιχα chroma CTBs και στοιχεία σύνταξης (syntax elements). Το μέγεθος από ένα luma CTB μπορεί να είναι 16x16, 32x32 ή 64x64 δειγμάτων, με τα μεγαλύτερα μεγέθη να επιτρέπουν τυπικά καλύτερη συμπίεση. Το HEVC υποστηρίζει ένα διαχωρισμό των CTBs σε μικρότερα μπλοκ χρησιμοποιώντας τη δομή quadtree.

2.4.2 Μονάδες και μπλοκ κωδικοποίησης, Coding units (CUs) και coding blocks (CBs)

Η σύνταξη του quadtree των μονάδων CTU καθορίζει το μέγεθος και τις θέσεις των luma και chroma μπλοκ CBs. Η ρίζα (root) του quadtree συνδέεται με το CTU. Ως εκ τούτου, το μέγεθος του luma CTB μπλοκ είναι το μεγαλύτερο υποστηριζόμενο μέγεθος για ένα luma μπλοκ CB. Η διάσπαση της μονάδας CTU σε luma και chroma μπλοκ CBs σηματοδοτείται από κοινού. Ένα luma CB μπλοκ και συνήθως δύο chroma CBs μαζί με την σχετική σύνταξη, σχηματίζουν μία μονάδα κωδικοποίησης CU. Ένα CTB μπορεί να περιέχει μόνο μία CU ή μπορεί να χωριστεί για να σχηματιστούν πολλαπλά CU, και κάθε CU διαμερίζεται σε μονάδες προβλέψεις (prediction units PUs) και ένα δέντρο μετασχηματισμού μονάδων (tree of transform units TUs).

2.4.3 Διαίρεση της εικόνας σε Coding Tree Units

Οι μονάδες κωδικοποίησης δέντρου (CTUs), περιέχουν luma CTBs και chroma CTBs. Ένα luma CTB καλύπτει μία περιοχή ορθογώνιας εικόνας από δείγματα LxL με την πληροφορία luma και τα αντίστοιχα chroma CTBs καλύπτουν δείγματα L/2×L/2 από πληροφορίες chroma. Η τιμή του L που είναι ίση με 16, 32 ή 64 καθορίζεται στο SPS. Σε σύγκριση με το παραδοσιακό macroblock που χρησιμοποιεί σταθερό μέγεθος συστοιχιών δειγμάτων 16x16 luma, όπως και τα προηγούμενα πρότυπα ITU-T και ISO/IEC JTC 1 από το H.261 (που τυποποιήθηκε το 1990), το HEVC υποστηρίζει κυμαινόμενου μεγέθους CTBs που επιλέγονται ανάλογα με τις ανάγκες των κωδικοποιητών, δηλαδή ανάλογα της μνήμης και υπολογιστικών πόρων. Η υποστήριξη μεγαλύτερων CTBs σε σχέση με τα προηγούμενα πρότυπα είναι ιδιαίτερα ευεργετική όταν κωδικοποιείται βίντεο υψηλής ανάλυσης. Το luma CTB και τα δύο chroma CTBs μαζί με τη σχετική σύνταξη αποτελούν ένα CTU. Το CTU είναι η βασική μονάδα επεξεργασίας που χρησιμοποιείται για να καθορίσει την διαδικασία αποκωδικοποίησης.



Σχήμα 5: Διαίρεση των Cus

Τα μπλοκ που ορίζονται ως luma και chroma CTBs μπορούν να χρησιμοποιηθούν απευθείας ως CBs ή μπορούν να χωριστούν περαιτέρω σε πολλαπλά CBs. Η κατανομή γίνεται με δομές δέντρων. Η κατανομή δέντρων σε HEVC εφαρμόζεται ταυτόχρονα και στα δύο luma και chroma, ωστόσο ισχύουν εξαιρέσεις όταν επιτυγχάνονται σίγουρα ελάχιστα μεγέθη για το chroma.
Το CTU επιτρέπει την διάσπαση των CBs σε ένα επιλεγμένο κατάλληλο μέγεθος που βασίζεται στα χαρακτηριστικά σήματος της περιοχής, η οποία καλύπτεται από το CTB. Η διαδικασία διαχωρισμού quadtree μπορεί να επαναληφθεί μέχρις ότου το μέγεθος για ένα Luma CB φτάσει σε ένα ελάχιστο επιτρεπόμενο μέγεθος που επιλέγεται από τον κωδικοποιητή και είναι πάντοτε 8x8 ή μεγαλύτερο.
Τα όρια της εικόνας ορίζονται σε μονάδες του ελάχιστου επιτρεπόμενου μεγέθους Luma CB. Ως αποτέλεσμα στα δεξιά και κάτω άκρα της εικόνας, μερικά CTUs μπορεί να καλύπτουν περιοχές που είναι εν μέρει εκτός των ορίων της εικόνας. Αυτή η συνθήκη ανιχνεύεται από τον αποκωδικοποιητή και το CTU quadtree είναι χωρισμένο όσο χρείαζεται για να μειωθεί το μέγεθος του CB στο σημείο όπου ολόκληρο το CB θα χωρέσει στην εικόνα.

2.4.4 Δομή Slice και Tile

2.4.4.1 Slices

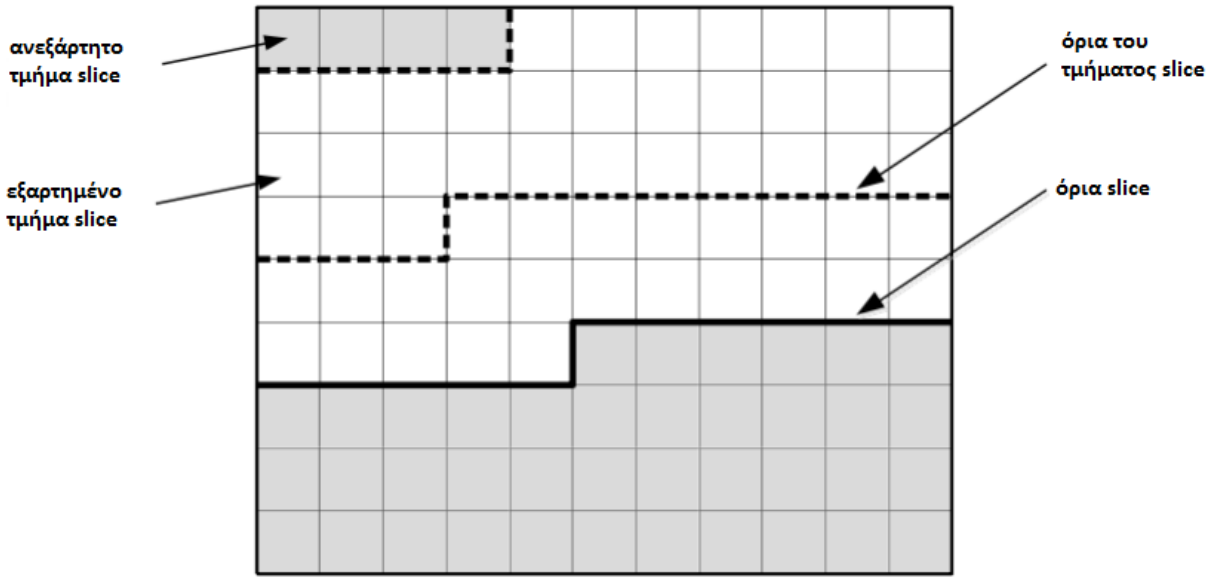
Ένα slice είναι μία δομή δεδομένων που μπορεί να αποκωδικοποιηθεί ανεξάρτητα από άλλα slices της ίδια εικόνας, δηλαδή να πραγματοποιηθεί η κωδικοποίηση εντροπίας (entropy coding), η πρόβλεψη σήματος (signal prediction) και η ανακατασκευή σήματος διαφορών (residual signal reconstruction). Ένα slice μπορεί να είναι είτε μία ολόκληρη εικόνα είτε μία περιοχή εικόνας, η οποία δεν είναι απαραίτητα ορθογώνια. Ένας από τους κύριους σκοπούς των slices είναι ο επανασυγχονισμός στην περίπτωση που χαθούν δεδομένα. Στην περίπτωση της μετάδοσης πακέτων, ο μέγιστος αριθμός των bit του payload μέσα σε ένα slice είναι περιορισμένος και ο αριθμός των μονάδων CTUs στο slice συχνά μεταβάλλεται για να ελαχιστοποιήσει την επιβάρυνση δημιουργίας πακέτων δεδομένων (packetization overhead) διατηρώντας παράλληλα το μέγεθος κάθε πακέτου μέσα σε αυτό το όριο. Ένα slice αποτελείται από μία ακολουθία από ένα ή περισσότερα τμήματα slices ξεκινώντας με ένα ανεξάρτητο τμήμα slice και περιέχει όλες τις υπο-ακολουθίες των τμημάτων slice (εάν υπάρχουν) που προηγούνται του επόμενου ανεξάρτητου τμήματος slice μέσα στην ίδια μονάδα πρόσβασης.

Όπως στο AVC, μία ακολουθία από CTBs ονομάζεται slice. Μία εικόνα μπορεί να χωριστεί σε οποιονδήποτε αριθμό slices ή ολόκληρη η εικόνα μπορεί να είναι ένα slice. Με τη σειρά του κάθε slice χωρίζεται σε ένα ή περισσότερα τμήματα slices (slice segments), το καθένα από τα οποία ανήκει σε μια μονάδα NAL. Μόνο το πρώτο τμήμα slice από ένα slice περιέχει την πλήρη κεφαλίδα (slice header), και τα υπόλοιπα τμήματα αναφέρονται ως εξαρτημένα τμήματα slice. Ένα εξαρτημένο τμήμα slice δεν αποκωδικοποιείται από μόνο του, ο αποκωδικοποιητής πρέπει να έχει πρόσβαση στο πρώτο τμήμα slice του slice. Αυτός ο διαχωρισμός των slices υπάρχει για να επιτρέψει την μετάδοση χαμηλής καθυστέρησης των εικόνων χωρίς να μειώνει την απόδοση κωδικοποίησης χρησιμοποιώντας πολλά πλήρεις slices.

2.4.4.2 Εξαρτημένα τμήματα slice (Dependent slice segments)

Μία δομή που ονομάζεται dependent slice segment επιτρέπει δεδομένα να σχετίζονται με ένα ειδικό σημείο εισόδου wavefront entry point ή ένα tile που θα μεταφερθεί σε μία ξεχωριστή μονάδα NAL, και έτσι δυνητικά καθιστά διαθέσιμα αυτά τα δεδομένα σε ένα σύστημα για κατακερματισμό πακετοποίησης με χαμηλότερο latency απ’ ό,τι αν ήταν κωδικοποιημένα μαζί σε ένα slice. Ένα

εξαρτημενό τμήμα slice στο σημείο εισόδου wavefront μπορεί να αποκωδικοποιηθεί μόνο μετά από ένα μέρος της διαδικασίας αποκωδικοποίησης του άλλου τμήματος slice που εκτελείται. Εξαρτημένα τμήματα slice είναι κυρίως χρήσιμα σε low-delay κωδικοποιήση, όπου άλλα παράλληλα εργαλεία μπορούν αντισταθμίσουν την μείωση της απόδοσης συμπίεσης.



Σχήμα 6: Παράδειγμα όπου ένα slice περιέχει δύο εξαρτημένα τμήματα slices, και ένα slice που δεν περιέχει εξαρτημένα τμήματα slices.

Τα Slices επεξεργάζονται με τη σειρά μιας raster σάρωσης -raster scan-. Τα slices είναι αυτοτελή, υπό την έννοια ότι, δεδομένης της διαθεσιμότητας των ενεργών ακολουθιών και των συνόλων των παραμέτρων της εικόνας, τα στοιχεία σύνταξής τους μπορούν να αναλυθούν από το bitstream.Οι τιμές των δειγμάτων στην περιοχή της εικόνας, που αντιπροσωπεύει το slice, μπορούν να αποκωδικοποιηθούν σωστά (εκτός από τις επιδράσεις του βρόχου φιλτραρίσματος κοντά στις άκρες του slice) χωρίς την χρήση οποιωνδήποτε δεδομένων από άλλα slices στην ίδια εικόνα. Αυτό σημαίνει ότι η πρόβλεψη εντός εικόνας (π.χ πρόβλεψη intrapicture ή πρόβλεψη διανυσμάτων κίνησης) δεν εκτελείται κατά μήκος των ορίων του slice. Ωστόσο, ορισμένες πληροφορίες από άλλα slices μπορούν να απαιτηθούν για την εφαρμογή του βρόχου φιλτραρίσματος κατά μήκος των ορίων του slice. Κάθε slice μπορεί να κωδικοποιηθεί χρησιμοποιώντας διαφορετικούς τύπους κωδικοποίησης ως εξής:

1)I slice: Ενα slice στο οποίο όλα τα CUs του slice κωδικοποιούνται χρησιμοποιώντας μόνο intrapicture πρόβλεψη.

2)P slice: Εκτός από τους τύπους κωδικοποίησης ενός I slice, ορισμένα CUs ενός P slice μπορούν επίσης να κωδικοποιηθούν χρησιμοποιώντας interpicture πρόβλεψη με το πολύ ένα σήμα πρόβλεψης αντιστάθμισης κίνησης – motion-compensated - ανά PB (δηλαδή uniprediction). P slices χρησιμοποιούν μόνο τη λίστα εικόνων αναφοράς 0 -list 0-.

3)B slice: Εκτός από τους τύπους κωδικοποίησης που είναι διαθέσιμες σε ένα P slice, ορισμένα CUs του B slice μπορούν να κωδικοποιηθούν χρησιμοποιώντας interpicture πρόβλεψη με το πολύ δύο σήματα πρόβλεψης ανά PB (δηλαδή biprediction). Τα B slices χρησιμοποιούν τόσο τη λίστα αναφοράς εικόνων 0 όσο και την 1 (list 0 and list 1).

Επιπλέον, τα slices συχνά περιορίζονται να χρησιμοποιούν ένα μέγιστο αριθμό bits, π.χ για την μετάδοση των πακέτων. Επομένως, τα slices μπορεί συχνά να να περιέχουν μια μεγάλη ποικιλία στον αριθμό των CTUs ανά slice κατά τρόπο που εξαρτάται από την δραστηριότητα στις σκηνές του video.

2.4.4.3 Wavefront parallel processing

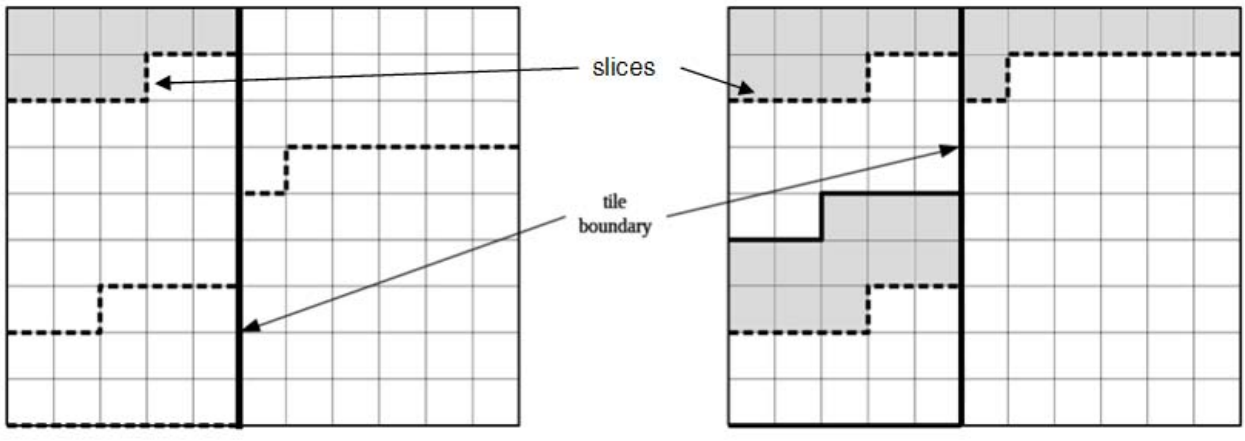
Όταν είναι ενεργοποιημένη η παράλληλη επεξεργασία wavefront parallel processing (WPP), ένα slice διαιρείται σε σειρές απο μονάδες CTUs.
Η αποκωδικοποίηση της κάθε σειράς μπορεί να ξεκινήσει μόλις ληφθούν λίγες αποφάσεις που απαιτούνται για την πρόβλεψη και την προσαρμογή του κωδικοποιητή εντροπίας στην προηγούμενη σειρά. Αυτό υποστηρίζει την παράλληλη επεξεργασία των σειρών CTU χρησιμοποιώντας αρκετά threads στον κωδικοποιητή ή τον αποκωδικοποιητή (ή και στα δύο). Για την απλότητα σχεδιασμού, το WPP δεν επιτρέπεται να χρησιμοποιείται σε συνδιασμό με tiles.

Η πρώτη σειρά επεξεργάζεται με το συνηθισμένο τρόπο, η δεύτερη σειρά μπορεί να αρχίσει να επεξεργάζεται αφού έχουν υποστεί επεξεργασία τα δύο CTUs της πρώτης σειράς, η τρίτη σειρά μπορεί να αρχίσει να επεξεργάζεται αφού έχουν υποστεί επεξεργασία τα δύο CTUs της δεύτερης σειράς και ούτω καθεξής. Το πλαίσιο μοντέλων του κωδικοποιητή εντροπίας σε κάθε σειρά έχει αναχθεί από εκείνες της προηγούμενης σειράς με χρονική καθυστέρηση επεξεργασία δυο CTUs. Το WPP παρέχει μία μορφή παράλληλης επεξεργασίας μέσα στο slice. Το WPP μπορεί συχνά να παρέχει καλύτερη απόδοση συμπίεσης από τα tiles.

2.4.4.4 Tiles

Η επιλογή του διαχωρισμού της εικόνας σε ορθόγωνιες περιοχές καθορίζεται από τα ονομαζόμενα tiles. Ο κύριος σκοπός των tiles είναι η αύξηση της παράλληλης επεξεργασίας και όχι τόσο στο να προβάλλουν ανθεκτικότητα λάθους. Τα tiles είναι ανεξάρτητες αποκωδικοποιημένες περιοχές της εικόνας. Τα tiles μπορούν επιπλέον να χρησιμοποιηθούν για το σκοπό της τυχαίας χωρικής πρόσβασης σε τοπικές περιοχές του κωδικοποιημένου βίντεο. Μία τυπική διαμόρφωση ενός tile μίας εικόνας αποτελείται από την τμηματοποίηση της εικόνας σε ορθογώνιες περιοχές με περίπου ίσο αριθμό από μονάδες CTUs σε κάθε tile. Τα tiles παρέχουν παράλληλη επεξεργασία.

Τα πολλαπλά tiles μπορεί να μοιράζονται τις πληροφορίες που υπάρχουν στο header με το να περιέχονται στο ίδο slice. Εναλλακτικά, ένα μόνο tile μπορεί μπορεί να περιέχει πολλαπλά slices. Ένα tile αποτελείται από μία ορθογώνια διατεταγμένη ομάδα CTUs (τυπικά, αλλά όχι απαραίτητα, με όλα αυτά να περιέχουν τον ίδιο αριθμό από CTUs).



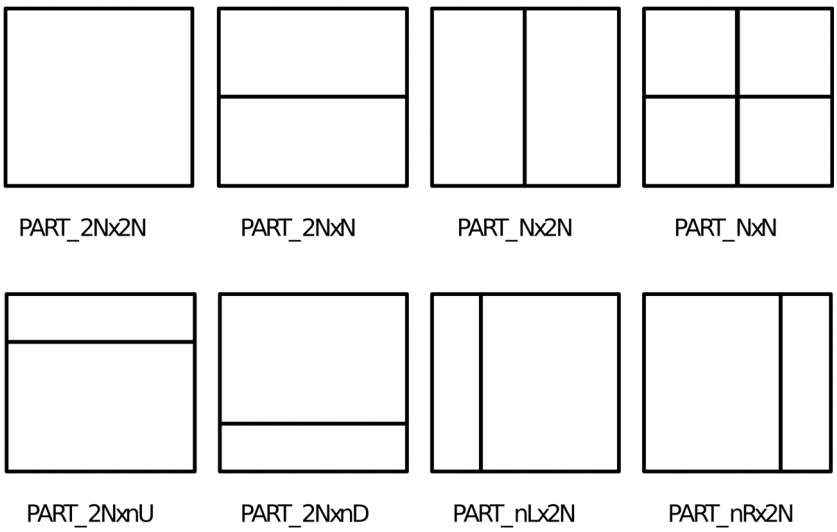
Σχήμα 7: Δύο διαφορετικές περιπτώσεις όπου το tile δε χωρίζει ένα slice (αριστερά) και περίπτωση όπου χωρίζει ένα slice στη μέση (δεξιά)

2.4.5 Μονάδες πρόβλεψης (PU) και μπλοκ πρόβλεψης (PBs)

Η απόφαση για το αν θα κωδικοποιηθεί μία περιοχή εικόνας χρησιμοποιώντας interpicture (temporal) ή intrapicture (spatial) πρόβλεψη γίνεται στο επίπεδο CU. Η διχοτόμηση PU έχει τη ρίζα της στο CU επίπεδο. Ανάλογα με την απόφαση του βασικού τύπου πρόβλεψης, τα luma και chroma CBs μπορεί στη συνέχεια να διασπαστούν σε μέγεθος όπου δημιουργούνται τα luma και chroma προβλεπόμενα μπλοκ (prediction blocks) PBs. Το HEVC υποστηρίζει μεταβλητά μεγέθη PB από 64x64 μέχρι 4x4 δείγματα.

Όταν ο τρόπος πρόβλεψης σηματοδοτείται ως intra, το μέγεθος του PB είναι το ίδιο με το μέγεθος του CB για όλα τα μεγέθη μπλοκ εκτός από το μικρότερο σε μέγεθος CB. Για την τελευταία περίπτωση, υπάρχει ένα flag που δείχνει εάν το CB είναι χωρισμένο σε τέσσερα PB quadrants που το καθένα έχει τη δικιά του λειτουργία πρόβλεψης. Ο λόγος για τον οποίο επιτρέπεται αυτή η διάσπαση είναι για να επιτρέπονται ξεχωριστές επιλογές της intrapicture πρόβλεψης για μπλοκ μεγέθους 4x4. Όταν η Luma intrapicture πρόβλεψη λειτουργεί σε μπλοκ 4x4 η chroma intrapicture πρόβλεψη χρησιμοποιεί επίσης 4x4 μπλοκ (το καθένα καλύπτει την ίδια περιοχή εικόνας με τέσσερα 4x4 block luma). Το πραγματικό μέγεθος της περιοχής στο οποίο λειτουργεί η intrapicture πρόβλεψη (το οποίο διακρίνεται από το μέγεθος PB, από Intrapicture πρόβλεψη) εξαρτάται από το διαχωρισμό διαφορών κωδικοποίησης.

Όταν η λειτουργία πρόβλεψης σηματοδοτείται ως inter, διευκρινίζεται αν τα luma και chroma CBs χωρίζονται σε ένα, δύο ή τέσσερα PBs. Η διάσπαση σε τέσσερα PBs επιτρέπεται μόνο όταν το μέγεθος CB είναι ίσο με το ελάχιστο επιτρεπόμενο μέγεθος CB, χρησιμοποιώντας ένα ισοδύναμο τύπο διάσπασης που διαφορετικά θα μπορούσε να εκτελεστεί σε επίπεδο CB του σχεδιασμού και όχι σε επίπεδο PB. Όταν ένα CB είναι χωρισμένο σε τέσσερα PBs, κάθε PB καλύπτει ένα quadrant του CB. Όταν ένα CB χωριστεί σε δύο PBs είναι δυνατοί έξι τύποι αυτού του διαχωρισμού.



Σχήμα 8: Περιπτώσεις διαχωρισμού του μπλοκ πρόβλεψης PB

Από την εικόνα φαίνονται οι περιπτώσεις όπου το CB δεν χωρίζεται και έχει μέγεθος 2Nx2N, το διαχωρισμό του CB σε δύο PBs μεγέθους 2N×N ή N×2N ή τη διάσπαση σε τέσσερα PBs μεγέθους N×N. Οι κάτω τέσσερεις τύποι διαχωρισμού αναφέρονται ως ασύμμετρος διαχωρισμός κίνησης, (asymmetric motion partitioning, AMP), και επιτρέπονται μόνο όταν το N είναι 16 ή μεγαλύτερο για το luma. Κάθε interpicture προβλεπόμενο PB ανατίθεται ένα ή δύο διανύσματα κίνησης και δείκτες αναφοράς εικόνας. Για να ελαχιστοποιηθεί η χειρότερη περίπτωση σχετικά με το έυρος της μνήμης (memory bandwidth), τα PBs luma μεγέθους 4x4 δεν επιτρέπονται για Interpicture

πρόβλεψη, και luma PBs μεγέθους 4x8 και 8x4 περιορίζονται στην unipredictive κωδικοποίηση. Η διαδικασία πρόβλεψης Interpicture περιγράφεται παρακάτω.
Τα Luma και chroma PBs, μαζί με τη σχετική σύνταξη πρόβλεψης, διαμορφώνουν το PU.

2.4.6 Μονάδες Μετασχηματισμού (TU)

Η πρόβλεψη διαφορών κωδικοποιείται χρησιμοποιώντας τα μπλοκ μεταφοράς (transform blocks). Μία μονάδα δέντρου TU έχει τη ρίζα της στο CU επίπεδο. Το luma μπλοκ CB διαφορών μπορεί να είναι ταυτόσημο με το luma μπλοκ μετασχηματισμού (TB) ή μπορεί να είναι περαιτέρω χωρισμένο σε μικρότερα luma TBs. Το ίδιο ισχύει και για τα chroma TBs. Συναρτήσεις βασιζόμενες σε ακεραίους παρόμοιες με εκείνες του διακριτού μετασχηματισμού συνημιτόνου (DCT) ορίζονται για να εξισώσουν τα μεγέθη TB 4×4, 8×8, 16×16, και 32×32. Για το 4x4 μετασχηματισμό διαφορών της Luma intrapicture πρόβλεψης εναλλακτικά προσδιορίζεται ένας ακέραιος μετασχηματισμός που προέρχεται από μία μορφη διακριτών ημιτόνου μετασχηματισμών (DST).

2.4.6.1 Δομή Δέντρου για το διαχωρισμό σε μονάδες και μπλοκ μετασχηματισμού

Για την κωδικοποίηση διαφορών (residual coding), ένα CB μπορεί να κατανεμηθεί αναδρομικά σε μπλοκ μετασχηματισμού (TBs). Ο διαχωρισμός σηματοδοτείται από ένα residual quadtree. Καθορίζεται μόνο διαχωρισμός τετραγωνικών CB και TB, όπου ένα μπλόκ μπορεί να χωριστεί αναδρομικά. Για ένα δεδομένο luma CB μεγέθους MxM, ένα flag σηματοδοτεί αν χωρίζεται σε τέσσερα μπλοκ μεγέθους M/2xM/2. Εάν είναι δυνατή η περαιτέρω διάσπαση, ανάλογα το μέγιστο μέγεθος του quadtree διαφορών που υποδεικνύεται στο SPS, κάθε quadrant έχει οριστεί ένα flag που υποδεικνύει αν είναι χωρισμένη σε τέσσερα quadrants. Ο κωδικοποιητής υποδεικνύει τα μέγιστα και τα ελάχιστα μεγέθη luma TB που θα χρησιμοποιήσει. Ο διαχωρισμός υπονοείται όταν το μέγεθος του CB είναι μεγαλύτερο από το μέγιστο μέγεθος του TB. Ο μη διαχωρισμός υπονοείται όταν η διάσπαση θα είχε ως αποτέλεσμα ένα μέγεθος luma TB μικρότερο από το ελάχιστο που υποδεικνύεται. Το μέγεθος του chroma TB είναι το μισό του Luma TB σε κάθε διάσταση, εκτός από την περίπτωση που το μέγεθος του luma TB είναι 4x4, στην οποία περίπτωση ένα μόνο 4x4 chroma TB χρησιμοποιείται για την περιοχή που καλύπτεται από τέσσερα 4x4 luma TBs. Στην περίπτωση των intrapicture προβλεπόμενων CUs, τα αποκωδικοποιημένα δείγματα των πληρέστερα γειτονικών TBs (μέσα ή έξω από το CB) χρησιμοποιούνται ως δεδομένα αναφοράς για την Intrapicture πρόβλεψη.
Σε αντίθεση με τα προηγούμενα πρότυπα, ο σχεδιασμός του HEVC επιτρέπει σε ένα TB να εκτείνεται σε πολλαπλά PBs για την interpicture πρόβλεψη των CUs έτσι ώστε να μεγιστοποιούν τα οφέλη της αποδοτικής κωδικοποίησης του διαχωρισμού του δέντρου TB.

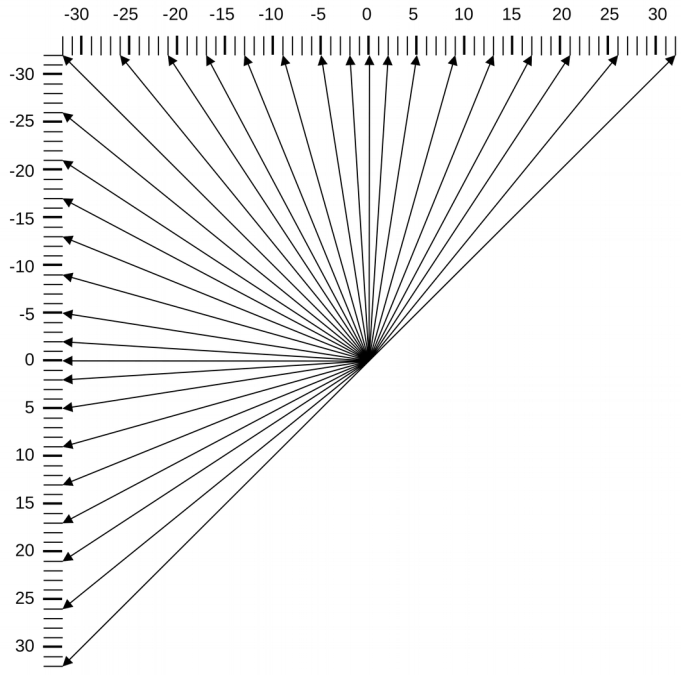
3. Τεχνικές Κωδικοποίησης του HEVC

3.1 Εισαγωγή

Όπως συμβαίνει και στα προηγούμενα πρότυπα κωδικοποίησης βίντεο ITU-T και ISO/IEC JTC 1, από το H.261, ο σχεδιασμός του H.265 ακολουθεί την κλασσική μέθοδο κωδικοποίησης υβριδικού βίντεο που βασίζεται στο μπλοκ (block-based hybrid video). Ο βασικός αλγόριθμος κωδικοποίησης είναι ένα υβριδικό μοντέλο interpicture πρόβλεψης για την εκμετάλλευση των χρονικών εξαρτήσεων και intrapicture πρόβλεψης για την εκμετάλλευση των χωρικών εξαρτήσεων. Δεν υπάρχει ενιαίο στοιχείο κωδικοποίησης στο σχεδιασμό του HEVC που να παρέχει στο μεγαλύτερο μέρος του σχεδιασμού του σημαντική βελτίωση στην απόδοση συμπίεσης σε σχέση με τα προγενέστερα πρότυπα κωδικοποίησης βίντεο. Πρόκειται για πολλές μικρότερες βελτιώσεις που αν προστεθούν αυξάνουν σημαντικά το κέρδος.

3.2 Πρόβλεψη κωδικοποίησης Intrapicture

Η intrapicure πρόβλεψη λειτουργεί σύμφωνα με το μέγεθος του TB, και χρησιμοποιούνται τα προηγούμενα αποκωδικοποιημένα συνοριακά δείγματα από χωρικά γειτονικά TBs για να σχηματίσουν το σήμα πρόβλεψης. Για την πρόβλεψη της κατεύθυνσης υπάρχουν 33 διαφορετικοί προσανατολισμοί που ορίζονται για (τετράγωνα) μεγέθη TB από 4x4 έως και 32x32.



Σχήμα 9: Πιθανές κατευθύνσεις πρόβλεψης

Εναλλακτικά μπορεί να χρησιμοποιηθεί επίσης η επίπεδη πρόβλεψη, η planar πρόβλεψη (υποθέτοντας μια επιφάνεια πλάτους με οριζόντια και κάθετη κλίση που προέρχεται από τα όρια) και η DC πρόβλεψη (επίπεδη επιφάνεια με τιμή που αποτελεί τη μέση τιμή των συνοριακών δειγμάτων). Για το chroma, οι καταστάσεις οριζόντιας, κάθετης, planar και DC πρόβλεψης μπορούν να σηματοδοτηθούν ακριβώς, ή ο τρόπος chroma πρόβλεψης μπορεί να υποδειχθεί ότι είναι ο ίδιος με τον τρόπο luma πρόβλεψης (και ως ειδική περίπτωση για να αποφευχθεί η περιττή σηματοδότηση, όταν μια από τις τέσσερις πρώτες επιλογές υποδεικνύεται και είναι ίδια με το τρόπο της luma πρόβλεψης, εφαρμόζεται η λειτουργία Intra −Angular).

Κάθε CB μπορεί να κωδικοποιηθεί από έναν από τους διάφορους τύπους κωδικοποίησης, ανάλογα με τον τύπο του slice. Παρόμοια με το H.264/MPEG-4 AVC, η κωδικοποίηση πρόβλεψης intrapicture υποστηρίζεται από όλους τους τύπους slices. Το HEVC υποστηρίζει διάφορους intrapicture μεθόδους κωδικοποίησης που αναφέρονται ως Intra−Angular, Intra−Planar, και Intra−DC.

3.2.1 Διαχωρισμός των block πρόβλεψης PB

Ένα μέγεθος intrapicture προβλεπόμενου CB MxM μπορεί να έχει έναν από τους δύο τύπους διαχωρισμού PB που αναφέρονται ως PART−2N×2N και PART−N×N, ο πρώτος από τους οποίους υποδεικνύει ότι το CB δε χωρίζεται και το δεύτερο υποδεικνύει ότι το CB είναι χωρισμένο σε τέσσερα ισάριθμα PBs. Ωστόσο, είναι δυνατόν να αναπαρασταθούν οι ίδιες περιοχές που θα καθορίζονταν από τα τέσσερα PBs χρησιμοποιώντας τέσσερα μικρότερα CBs όταν το μέγεθος του τρέχοντος CB είναι μεγαλύτερο από το ελάχιστο μέγεθος CU. Έτσι, ο σχεδιασμός του HEVC επιτρέπει μόνο τον τύπο διαχωρισμού PART−N×N να χρησιμοποιείται όταν το τρέχον μέγεθος CB είναι ίσο με το ελάχιστο μέγεθος CU. Αυτό σημαίνει ότι το μέγεθος PB είναι πάντα ίσο με το μέγεθος CB, όταν το CB κωδικοποιείται χρησιμοποιώντας Intrapicture πρόβλεψη. Παρόλο που ο τρόπος πρόβλεψης intrapicture καθορίζεται σε επίπεδο PB, η διαδικασία πρόβλεψης λειτουργεί ξεχωριστά για κάθε TB.

3.2.2 Πρόβλεψη Intra−Angular

Η intrapicture πρόβλεψη του HEVC λειτουργεί παρόμοια με αυτή στο H.264/MPEG-4 AVC, αλλά επεκτείνεται σημαντικά, κυρίως λόγω του αυξημένου μεγέθους TB και ενός αυξημένου αριθμού επιλεγόμενων κατευθύνσεων πρόβλεψης. Σε σύγκριση με τις 8 κατευθύνσεις πρόβλεψης του H.264/MPEG-4 AVC, ο HEVC υποστηρίζει συνολικά 33 κατευθύνσεις πρόβλεψης που σημειώνονται ως Intra−Angular[k], όπου το k είναι ένας αριθμός κατάστασης από το 2 μέχρι το 34. Οι γωνίες σχεδιάζονται σκόπιμα για να παρέχουν πυκνότερη κάλυψη για τις σχεδόν οριζόντιες και τις σχεδόν κατακόρυφες γωνίες και μεγαλύτερη κάλυψη για τις σχεδόν διαγώνιες γωνίες ώστε να είναι αποτελεσματικότερη η επεξεργασία πρόβλεψης των σημάτων.
Όταν χρησιμοποιείται Intra−Angular λειτουργία, κάθε TB προβλέπεται από χωρικά γειτονικά δείγματα που έχουν ανακατασκευαστεί (αλλά δεν έχουν ακόμα φιλτραριστεί από in-loop φίλτρα) πριν χρησιμοποιηθούν γι αυτή την πρόβλεψη. Για ένα TB μεγέθους NxN, συνολικά μπορούν να χρησιμοποιηθούν τα 4N+1 χωρικά γειτονικά δείγματα για την πρόβλεψη. Στο HEVC όταν είναι διαθέσιμα από προηγούμενες διαδικασίες αποκωδικοποίησης, μπορούν να χρησιμοποιηθούν για πρόβλεψη δείγματα από τα κάτω αριστερά TBs, επιπλέον μπορούν να χρησιμοποιηθούν δείγματα από TBs αριστερά, πάνω και πάνω δεξιά του τρέχοντος TB.

Προκειμένου να βελτιωθεί η ακρίβεια της intrapicture πρόβλεψης, η προβλεπόμενη θέση δείγματος αναφοράς υπολογίζεται με ακρίβεια δείγματος 1/32. Χρησιμοποιείται γραμμική παρεμβολή για να ληφθεί η τιμή του προβλεπόμενου δείγματος αναφοράς χρησιμοποιώντας τα δύο κοντινότερα δείγματα αναφοράς που βρίσκονται σε ακέραιες θέσεις.
Η διαδικασία πρόβλεψης των λειτουργιών Intra−Angular είναι σταθερές σε όλα τα μεγέθη μπλοκ και στις κατευθύνσεις πρόβλεψης, ενώ το H.264/MPEG-4 AVC χρησιμοποιεί διαφορετικές μεθόδους για τα υποστηριζόμενα μεγέθη μπλοκ των 4x4, 8x8 και 16x16. Αυτή η σταθερότητα σχεδιασμού είναι ιδιαίτερα επιθυμητή, καθώς το HEVC υποστηρίζει μεγαλύτερη ποικιλία μεγεθών TB και σημαντικά αυξημένο αριθμό κατευθύνσεων πρόβλεψης σε σύγκριση με το H.264/MPEG-4 AVC.

3.2.3 Πρόβλεψη Intra−Planar και Intra−DC

Εκτός από την Intra−Angular, το HEVC υποστηρίζει δυο εναλλακτικές μεθόδους πρόβλεψης την Intra−Planar και την Intra−DC με παρόμοιες λειτουργίες να έχουν καθοριστεί στο H.264/MPEG-4 AVC. Ενώ η πρόβλεψη Intra−DC χρησιμοποιεί μία μέση τιμή των δειγμάτων αναφοράς για την πρόβλεψη, στην Intra−Planar πρόβλεψη χρησιμοποιείται η μέση τιμή δύο γραμμικών προβλέψεων με την χρήση τεσσέρων γωνιακών δειγμάτων αναφοράς για την πρόβλεψη των ασυνεχειών κατά μήκος των ορίων του μπλοκ. Η Intra−Planar λειτουργία πρόβλεψης υποστηρίζεται σε όλα τα μεγέθη μπλοκ στο HEVC, ενώ στο H.264/MPEG-4 AVC υποστηρίζει την πρόβλεψη επιπέδων μόνο όταν το μέγεθος luma PB είναι 16x16, και αυτή η πρόβλεψη επιπέδου λειτουργεί διαφορετικά από την επίπεδη πρόβλεψη στο HEVC.

3.2.4 Δείγματα αναφοράς

Στο HEVC τα δείγματα αναφοράς που χρησιμοποιούνται για την intrapicture πρόβλεψη μερικές φορές φιλτράρονται από ένα φίλτρο εξομάλυνσης με τρόπο παρόμοιο με αυτόν που χρησιμοποιήθηκε για την 8x8 intrapicture πρόβλεψη στο H.264/MPEG-4 AVC. Το HEVC εφαρμόζει λειτουργίες εξομάλυνσης ανάλογα με την κατεύθυνση, το μέγεθος της ασυνέχειας που ανιχνεύθηκε, και το μέγεθος του μπλοκ. Όπως στο H.264/MPEG-4 AVC, το φίλτρο εξομάλυνσης δεν εφαρμόζεται για 4x4 μπλοκ. Για 8x8 μπλοκ, μόνο οι διαγώνιες κατευθύνσεις, Intra−Angular[k] με k= 2, 18, or 34, χρησιμοποιούν το φίλτρο. Για 16x16 μπλοκ τα δείγματα αναφοράς φιλτράρονται για τις περισσότερες κατευθύνσεις εκτός από τις σχεδόν οριζόντιες και τις σχεδόν κάθετες κατευθύνσεις με το k στην περιοχή απο 9-11 και 25-27. Για 32x32 μπλοκ όλες οι κατευθύνσεις ,εκτός από τις ακριβώς οριζόντιες k=10 και ακριβώς κάθετες k=26, χρησιμοποιούν το φίλτρο εξομάλυνσης και όταν το μέγεθος της ασυνέχειας που ανιχνεύθηκε υπερβαίνει ένα κατώφλι, εφαρμόζεται γραμμική παρεμβολή.
Η Intra−Planar λειτουργία χρησιμοποιεί ,επίσης, το φίλτρο εξομάλυνσης όταν το μπλοκ είναι μεγαλύτερο ή ίσο του 8x8 μπλοκ ,ενώ στην περίπτωση της Intra−DC δεν χρησιμοποιείται εξομάλυνση.

3.2.5 Οι τιμές στα όρια δειγμάτων

Για να αφαιρεθούν οι ασυνέχειες κατά μήκος των ορίων των μπλοκ, στις τρεις λειτουργίες, Intra−DC (mode 1) και Intra−Angular[k] με k= 10 ή 26 (ακριβώς οριζόντια και κάθετη) όταν το μέγεθος του TB είναι μικρότερο από 32x32 τότε τα συνοριακά δείγματα εντός του TB αντικαθίσταται από φιλτραρισμένες τιμές.
Για την λειτουργία Intra−DC, τόσο η πρώτη σειρά όσο και η στήλη των δειγμάτων στο TB αντικαθίσταται από την έξοδο ενός φίλτρου που τροφοδοτείται από την αρχική τους τιμή και το γειτονικό δείγμα αναφοράς. Στην οριζόντια πρόβλεψη (Intra−Angular[10]) τα όρια δειγμάτων της πρώτης στήλης του TB τροποποιούνται έτσι ώστε να είναι το μισό της διαφοράς μεταξύ του γειτονικού δείγματος αναφοράς και του πάνω αριστερού δείγματος αναφοράς. Αυτό κάνει το σήμα πρόβλεψης περισσότερο ομαλό. Στην κατακόρυφη πρόβλεψη (Intra−Angular[26]) εφαρμόζεται το ίδιο στην πρώτη σειρά δειγμάτων.

3.2.6 Αντικατάσταση δειγμάτων αναφοράς

Τα γειτονικά δείγματα αναφοράς δεν είναι διαθέσιμα στα όρια των slices ή tiles. Επιπλέον, όταν ενεργοποιείται το χαρακτηριστικό για την ανθεκτικότητα των απωλειών, τα γειτονικά δείγματα αναφοράς εντός οποιουδήποτε interpicture προβλεπόμενου PB θεωρούνται επίσης μη διαθέσιμα. Με το τρόπο αυτό αποφεύγεται η πιθανότητα αλλοίωσης των πιθανώς κατεστραμμένων

προηγουμένων αποκωδικοποιημένων δεδομένων εικόνας και δε μεταδίδονται σφάλματα στο σήμα πρόβλεψης. Αν και στο H.264/MPEG-4 AVC επιτρεπόταν μόνο η πρόβλεψη Intra−DC για αυτές τις περιπτώσεις, το HEVC επιτρέπει τη χρήση άλλων λειτουργιών Intrapicture πρόβλεψης αφού αντικαθίστανται οι μη διαθέσιμες τιμές δείγματος αναφοράς με τις γειτονικές διαθέσιμες τιμές δείγματος αναφοράς.

3.2.7 Περιγραφή των τρόπων πρόβλεψης

Το HEVC υποστηρίζει συνολικά 33 Intra−Angular τρόπους πρόβλεψης, καθώς και τους τρόπους πρόβλεψης Intra−Planar και Intra−DC για όλα τα μεγέθη μπλοκ. Λόγω του αυξημένου αριθμού των κατευθύνσεων, το HEVC θεωρεί τρεις πιο πιθανούς τρόπους πρόβλεψης (most probable modes, MPMs) όταν κωδικοποιεί πρόβλεψη τύπου intrapicture luma, σε σχέση με τον ένα πιο πιθανό τρόπο πρόβλεψης που χρησιμοποιεί το H.264/MPEG-4 AVC.
Μεταξύ των τριών πιο πιθανών τρόπων πρόβλεψης, οι δύο πρώτοι αρχικοποιούνται από τον τρόπο πρόβλεψης του luma intrapicture των παραπάνω και αριστερών PBs, εάν αυτά τα PBs είναι διαθέσιμα και κωδικοποιούνται χρησιμοποιώντας Intrapicure πρόβλεψη. Οποιοσδήποτε μη διαθέσιμος τρόπος πρόβλεψης θεωρείται Intra −DC. Το PB πάνω από το luma CTB θεωρείται πάντα ότι είναι μη διαθέσιμο για να αποφευχθεί η ανάγκη αποθήκευσης buffer γραμμών των γειτονικών τρόπων luma πρόβλεψης. Όταν οι δύο πρώτοι πιο πιθανοί τρόποι δεν είναι ίσοι, ο τρίτος πιο πιθανός τρόπος ορίζεται ίσος με Intra−Planar, Intra−DC, ή Intra−Angular[26] (κάθετος). Όταν οι δύο πρώτοι πιο πιθανοί τρόποι πρόβλεψης είναι ίδιοι και ο πρώτος τρόπος έχει την τιμή Intra−Planar ή Intra−DC, ο δεύτερος και ο τρίτος πιο πιθανός τρόπος ορίζεται ως Intra−Planar, Intra−DC, ή Intra−Angular[26]. Όταν οι δύο πρώτοι πιο πιθανοί τρόποι είναι ίδιοι και ο πρώτος τρόπος έχει μια τιμή Intra−Angular, τότε ο δεύτερος και ο τρίτος πιο πιθανός τρόπος επιλέγονται με βάσει την γωνία, δηλαδή την πληρέστερα στη γωνία (τιμή του k) του πρώτου. Στην περίπτωση που ο τρέχων τρόπος luma πρόβλεψης είναι ένας από τους τρεις MPMs, μόνο ο δείκτης του MPM μεταδίδεται στον αποκωδικοποιητή. Διαφορετικά, ο δείκτης του τρέχοντος τρόπου luma πρόβλεψης, εξαιρουμένων των τριών MPMs θα μεταδοθεί στον αποκωδικοποιητή.
Για την chroma intrapicture πρόβλεψη, ο HEVC επιτρέπει στον αποκωδικοποιητή να επιλέξει έναν από του πέντε τρόπους Intra−Planar, Intra−Angular[26] (vertical), Intra−Angular[10] (horizontal), Intra−DC, και Intra−Derived. Ο Intra−Derived τρόπος δηλώνει ότι η chroma πρόβλεψη χρησιμοποιεί την ίδια γωνιακή κατεύθυνση όπως η Luma πρόβλεψη. Με αυτό το σχήμα, όλοι οι γωνιακοί τρόποι που καθορίζονται για το Luma στο HEVC μπορούν θεωρητικά να χρησιμοποιηθούν στην chroma πρόβλεψη, και με αυτόν τον τρόπο επιτυγχάνεται μία καλή αντιστοίχιση της ακρίβειας πρόβλεψης. Ο επιλεγμένος τρόπος chroma πρόβλεψης κωδικοποιείται απευθείας (χωρίς την χρήση μηχανισμού πρόβλεψης MPM).

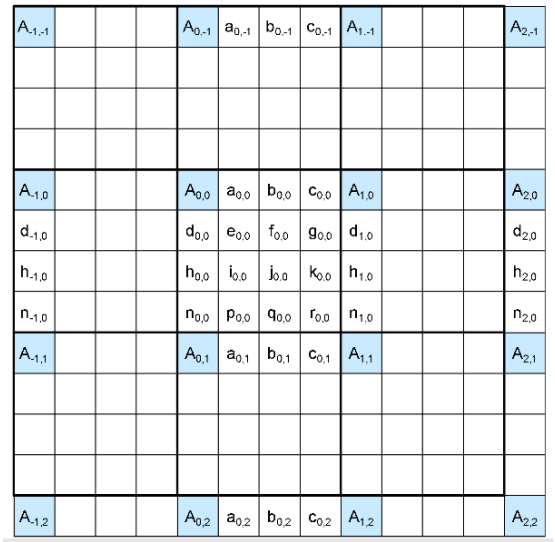
3.3 Πρόβλεψη κωδικοποίησης Interpicture

3.3.1 Διαχωρισμός των block πρόβλεψης PB

Σε σύγκριση με τα intrapicture προβλεπόμενα CBs, το HEVC υποστηρίζει περισσότερα σχήματα διαμέρισης PB για interpicture πρόβλεψη. Οι τρόποι διαμερισμού PART−2N×2N, PART−2N×N, και PART−N×2N υποδεικνύουν τις περιπτώσεις όταν το CB δεν χωρίζεται, χωρίζεται σε δύο ίσου μεγέθους PBs οριζόντια, και σε δύο ίσου μεγέθους PBs κάθετα, αντίστοιχα. Το PART−N×N καθορίζει ότι το CB χωρίζεται σε τέσσερα ίσου μεγέθους PBs, αλλά αυτός ο τρόπος υποστηρίζεται μόνο όταν μέγεθος CB είναι το μικρότερο επιτρεπόμενο μέγεθος CB. Επιπροσθέτως, υπάρχουν τέσσερις τύποι διαχωρισμού που υποστηρίζουν τη διάσπαση του CB σε δυο PBs με διαφορετικά μεγέθη και είναι γνωστοί ως ασύμμετρα χωρίσματα κίνησης.

3.3.2 Κλασματική παρεμβολή δείγματος

Τα δείγματα του PB ή ενός Intrapicture προβλεπόμενου CB λαμβάνονται από αυτά μια αντίστοιχης περιοχής μπλοκ στην εικόνα αναφοράς και ταυτοποιούνται από ένα δείκτη εικόνας αναφοράς, ο οποίος είναι σε μια θέση που μετατοπίζεται από τις οριζόντιες και κάθετες συνιστώσες των διανυσμάτων κίνησης. Εκτός από την περίπτωση που το διάνυσμα κίνησης έχει μια ακέραια τιμή, χρησιμοποιείται κλασματική παρεμβολή δείγματος για την δημιουργία των δειγμάτων πρόβλεψης για τις μη ακέραιες θέσεις δειγματοληψίας. Όπως στο H.264/MPEG-4 AVC, το HEVC υποστηρίζει διανύσματα κίνησης με μονάδες ενός τετάρτου της απόστασης μεταξύ των luma δειγμάτων. Για τα δείγματα chroma, η ακρίβεια του διανύσματος κίνησης προσδιορίζεται σύμφωνα με τη μορφοποίηση της δειγματοληψίας chroma, η οποία για δειγματοληψία 4:2:0 δημιουργεί μονάδες του ενός ογδόου της απόστασης μεταξύ chroma δειγμάτων.
Η παρεμβολή κλασματικού δείγματος για luma δείγματα στο HEVC χρησιμοποιεί ξεχωριστή εφαρμογή ενός 8-tap φίλτρου για τις θέσεις μισού δείγματος και ένα 7-tap φίλτρου για τις θέσεις ενός τετάρτου δείγματος. Αυτό είναι σε αντίθεση με την διαδικασία που χρησιμοποιείται στο H.264/MPEG-4 AVC. Τότε η εφαρμογή της διεργασίας δύο σταδίων παρεμβολής δημιουργεί πρώτα τις τιμές ενός ή δυο γειτονικών δειγμάτων σε θέσεις μισού δείγματος χρησιμοποιώντας 6-tap φίλτρο, στρογγυλοποιεί μέσω του φίλτρου τα ενδιάμεσα αποτελέσματα .Βρίσκοντας, έπειτα, τον μέσο όρο σε ακέραιες ή μισού δείγματος θέσεις. Το HEVC αντ’ αυτού χρησιμοποιεί μία διεργασία παρεμβολής δημιουργώντας όλες τις κλασματικές θέσεις χωρίς ενδιάμεσες πράξεις στρογγυλοποίησης, γεγονός που βελτιώνει την ακρίβεια και απλοποιεί την αρχιτεκτονική της κλασματικής παρεμβολής δείγματος. Η ακρίβεια παρεμβολής βελτιώνεται επίσης στο HEVC με τη χρήση μεγαλύτερων φίλτρων, δηλαδή 7-tap ή 8-tap φίλτρα σε σχέση με το 6-tap φίλτρο που χρησιμοποιείται στο H.264/MPEG-4 AVC.



Σχήμα 10: Θέσεις δειγμάτων luma

Οι θέσεις που σημειώνονται με κεφαλαία γράμματα, $A_{i,j}$, αντιπροσωπεύουν τα διαθέσιμα δείγματα luma σε ακέραιες θέσεις δείγματος, ενώ οι άλλες θέσεις που σημειώνονται με μικρά γράμματα αντιπροσωπεύουν δείγματα σε μη ακέραιες θέσεις δείγματος, τα οποία πρέπει να δημιουργούνται με παρεμβολή. Τα δείγματα $a_{0,j}$, $b_{0,j}$, $c_{0,j}$, $d_{0,0}$, $h_{0,0}$, και $n_{0,0}$ προέρχονται από τα δείγματα $A_{i,j}$ με εφαρμογή του 8-tap φίλτρου για θέσεις μισού δείγματος και 7-tap για τις θέσεις τετάρτου δείγματος. Τα δείγματα $e_{0,0}$, $f_{0,0}$, $g_{0,0}$, $i_{0,0}$, $j_{0,0}$, $k_{0,0}$, $p_{0,0}$, $q_{0,0}$,και $r_{0,0}$ μπορούν να προέλθουν εφαρμόζοντας τα αντίστοιχα φίλτρα σε δείγματα που βρίσκονται σε κατακόρυφη γειτονική θέση $a_{0,j}$, $b_{0,j}$ και $c_{0,j}$.

Το φιλτράρισμα παρεμβολής διαχωρίζεται με τέτοιον τρόπο ώστε οι ίδιες τιμές να μπορούν να υπολογιστούν στην περίπτωση που εφαρμόζεται κατακόρυφο φιλτράρισμα πριν από το οριζόντιο φιλτράρισμα.
Σε αυτό το σημείο της διαδικασίας εφαρμόζεται η σταθμισμένη πρόβλεψη (weighted prediction) που επιλέγεται από τον κωδικοποιητή. Ενώ το H.264/MPEG-4 AVC υποστηρίζει τόσο την χρονική όσο και την σταθμισμένη πρόβλεψη, στο HEVC εφαρμόζεται μόνο η σταθμισμένη πρόβλεψη, με κλιμάκωση και αντιστάθμιση (scaling and offsetting) της πρόβλεψης με τιμές που αποστέλλονται απο τον κωδικοποιητή. Στην περίπτωση της unipredictιon, η παρεμβαλλόμενη (και πιθανώς σταθμισμένη) τιμή πρόβλεψης είναι στρογγυλεμένη, μετατοπισμένη δεξιά, και έχει περικοπεί ώστε να έχει το αρχικό αριθμό από bits. Στην περίπτωση της biprediction, η παρεμβαλλόμενη (και πιθανώς σταθμισμένη) τιμή πρόβλεψης από τα δύο PBs προστίθεται πρώτα, και στη συνέχεια στρογγυλοποιείται, μετατοπίζεται δεξιά, και περικόπτεται.

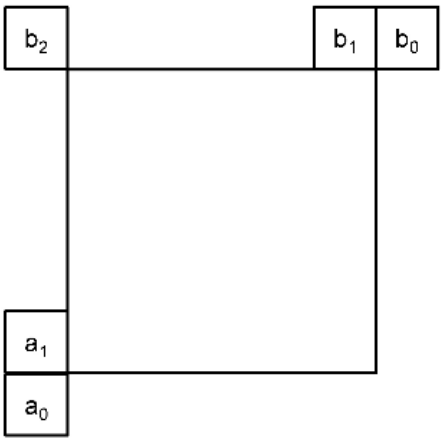
Στο H.264/MPEG-4 AV, απαιτούνται έως και τρία στάδια εργασιών στρογγυλοποίησης για τη λήψη κάθε δείγματος πρόβλεψης (για δείγματα που βρίσκονται σε θέσεις του 1/4 δείγματος). Εάν χρησιμοποιείται η biprediction, ο συνολικός αριθμός των εργασιών στρογγυλοποίησης είναι επτά στη χειρότερη περίπτωση. Στο HEVC, χρειάζονται το πολύ δυο εργασίες στρογγυλοποίησης για την απόκτηση κάθε δείγματος που βρίσκεται στις θέσεις 1/4 δείγματος, επομένως πέντε λειτουργίες στρογγυλοποίησης είναι επαρκείς στην χειρότερη περίπτωση όταν χρησιμοποιείται biprediction. Λόγω του μικρότερου αριθμού εργασιών στρογγυλοποίησης το συσσωρευμένο σφάλμα στρογγυλοποίησης μειώνεται και επιτρέπεται μεγαλύτερη ευελιξία όσον αφορά τον τρόπο εκτέλεσης των απαραίτητων λειτουργιών στον αποκωδικοποιητή.
Η διαδικασία παρεμβολής κλασματικού δείγματος για τα chroma συστατικά είναι παρόμοια με αυτή για το luma στοιχείο. Το HEVC ορίζει ένα σύνολο 4-tap φίλτρων για θέσεις 1/8 δείγματος (ενώ το H.264/MPEG-4 AVC χρησιμοποιούσε 2-tap φίλτρο).

Οι τιμές συντελεστών φίλτρου δηλώνονται ως filter1[i], filter2[i], filter3[i], και filter4[i] με i=−1,. . . , 2 χρησιμοποιούνται για την παρεμβολή των κλασματικών θέσεων 1/8, 2/8, 3/8 και 4/8 για τα δείγματα chroma αντίστοιχα. Στις θέσεις 5/8η, 6/8η και 7/8η, χρησιμοποιούνται οι συμμετρικές τιμές των filter3[1−i], filter2[1−i], και filter1[1−i] με i=−1, ...,2 , αντίστοιχα.

3.3.3 Λειτουργία Συγχώνευσης

Οι πληροφορίες κίνησης προέρχονται από τις τιμές οριζόντιας και κάθετης μετατόπισης του διανύσματος κίνησης, ένα ή δύο δείκτες αναφοράς εικόνας και στην περίπτωση των περιοχών πρόβλεψης για B slices, ένα αναγνωριστικό από το ποια λίστα αναφοράς εικόνων συνδέεται με τον κάθε δείκτη. Το HEVC περιλαμβάνει μία λειτουργία συγχώνευσης για την εξαγωγή των πληροφοριών κίνησης από χωρικά ή χρονικά γειτονικά μπλοκ. Δηλώνεται ως λειτουργία συγχώνευσης αφού αποτελεί συγχωνευμένη περιοχή που μοιράζεται όλες τις πληροφορίες κίνησης.

Η λειτουργία συγχώνευσης είναι παρόμοια με αυτή στο H.264/MPEG-4 AVC. Ωστόσο, υπάρχουν δύο σημαντικές διαφορές. Πρώτον, μεταδίδει πληροφορίες δεικτών για να διαλέξει ένα από τους διαφορετικούς υποψηφίους με έναν τρόπο που μερικές φορές αναφέρεται ως σχέδιο motion vector competition. Δεύτερον, προσδιορίζει την λίστα εικόνων αναφοράς και τον δείκτη εικόνας αναφοράς, ενώ η άμεση λειτουργία υποθέτει ότι αυτές έχουν ορισμένες προκαθορισμένες τιμές.
Το σύνολο των πιθανών υποψηφίων στον τρόπο συγχώνευσης αποτελείται από χωρικούς γειτονικούς υποψηφίους, ένα προσωρινό υποψήφιο και παραγόμενους υποψήφιους.



Σχήμα 11: Θέσεις των χωρικών υποψηφίων

Στο σχήμα φαίνονται οι θέσεις των πέντε χωρικών υποψηφίων. Για κάθε υποψήφια θέση, η διαθεσιμότητα ελέγχεται με την ακόλουθη σειρά $\{a_1, b_1, b_0, a_0, b_2\}$. Αν το μπλοκ βρίσκεται σε θέση που είναι για intrapicture πρόβλεψη ή σε θέση που βρίσκεται εκτός του τρέχοντος slice ή tile, θεωρείται ως μη διαθέσιμη.

Μετά την επικύρωση των χωρικών υποψηφίων καταργούνται δύο είδη πλεονασμού. Αν η υποψήφια θέση για το τρέχον PU αναφέρεται στο πρώτο PU εντός του ίδιου CU, η θέση αποκλείεται καθώς η ίδια η συγχώνευση θα μπορούσε να επιτευχθεί από τη CU χωρίς να χωριστεί σε τμήματα πρόβλεψης. Επιπλέον, αποκλείονται τυχόν περιττές καταχωρήσεις όπου οι υποψήφιοι έχουν ακριβώς τις ίδιες πληροφορίες κίνησης.

Για το χρονικό υποψήφιο χρησιμοποιείται η δεξιά κατώτατη θέση λίγο έξω από το παρατιθέμενο PU της εικόνας αναφοράς, εάν είναι διαθέσιμη. Διαφορετικά χρησιμοποιείται η κεντρική θέση. Ο τρόπος επιλογής του παρατιθέμενου PU είναι παρόμοιος με αυτό των προηγούμενων προτύπων, αλλά στο HEVC επιτρέπει μεγαλύτερη ευελιξία μεταδίδοντας ένα δείκτη για να καθορίσει ποια λίστα αναφοράς εικόνας χρησιμοποιείται για την παρατιθέμενη εικόνα αναφοράς.

Ένα ζήτημα που σχετίζεται με την χρήση του χρονικού υποψηφίου είναι το μέγεθος της μνήμης για την αποθήκευση των πληροφοριών κίνησης της εικόνας αναφοράς. Αυτό αντιμετωπίζεται περιορίζοντας την λεπτομέρεια για την αποθήκευση της χρονικής κίνησης των υποψήφιων μόνο για την επίλυση ενός πλέγματος 16x16, ακόμα και όταν χρησιμοποιούνται μικρότερες δομές PB στην αντίστοιχη θέση εικόνας αναφοράς. Επιπλέον, ένα flag , που ορίζεται, στο PPS, επιτρέπει στον αποκωδικοποιητή να απενεργοποιήσει τη χρήση χρονικού υποψηφίου, κάτι που είναι χρήσιμο για εφαρμογές που είναι επιρρεπείς στα σφάλματα.

Ο μέγιστος αριθμός των υποψήφιων συγχώνευσης C, καθορίζεται στο header του slice. Εάν ο αριθμός των υποψήφιων συγχώνευσης (συμπεριλαμβανομένου του χρονικού) που βρέθηκε είναι μεγαλύτερος από το C, μόνο οι πρώτοι C-1 χωρικοί υποψήφιοι και ο χρονικός θα διατηρηθούν. Διαφορετικά, αν είναι μικρότερος από C, δημιουργούνται πρόσθετοι υποψήφιοι μέχρι ο αριθμός να έιναι ίσος με C. Αυτό απλοποιεί την ανάλυση και την καθιστά πιο ισχυρή, καθώς είναι η δυνατότητα ανάλυσης κωδικοποιημένων δεδομένων δεν εξαρτάται από την διαθεσιμότητα των υποψήφιων συγχώνευσης.

Για τα B slices, δημιουργούνται πρόσθετοι υποψήφιοι συγχώνευσης επιλέγοντας δύο υφιστάμενους υποψήφιους σύμφωνα με μια προκαθορισμένη σειρά από την λίστα αναφοράς εικόνων 0 και 1. Για παράδειγμα, ο πρώτος υποψήφιος που δημιουργήθηκε χρησιμοποιεί τον πρώτο υποψήφιο συγχώνευσης για την λίστα 0 και τον δεύτερο υποψήφιο συγχώνευσης για την λίστα 1. Το HEVC προσδιορίζει συνολικά προκαθορισμένα ζεύγη των δύο με την ακόλουθη σειρά στην ήδη κατασκευασμένη λίστα υποψηφίων συγχώνευσης ως (0, 1), (1, 0), (0, 2), (2, 0), (1, 2), (2, 1), (0, 3), (3, 0), (1, 3), (3, 1), (2, 3), και (3, 2). Μεταξύ αυτών, μπορούν να συμπεριληφθούν έως και πέντε υποψήφιοι μετά την κατάργηση των περιττών καταχωρήσεων.

Στο HEVC, η λειτουργία skip, αντιμετωπίζεται ως ειδική περίπτωση της λειτουργίας συγχώνευσης, όλα τα flags των κωδικοποιημένων μπλοκ είναι ίσα με το μηδέν. Σε αυτή την συγκεκριμένη περίπτωση, μόνο ένα skip flag και ο αντίστοιχος δείκτης συγχώνευσης μεταδίδονται στον αποκωδικοποιητή. Η λειτουργία B-direct στο H.264/MPEG-4 AVC αντικαθίσταται, επίσης, από την λειτουργία συγχώνευσης, δεδομένου ότι η λειτουργία συγχώνευσης επιτρέπει την εξαγωγή όλων των πληροφοριών κίνησης από τις πληροφορίες χωρικής και χρονικής κίνησης των γειτονικών μπλοκ με κωδικοποίηση διαφορών.

3.3.4 Πρόβλεψη διανύσματος κίνησης

Όταν ένα interpicture προβλεπόμενο CB δεν έχει κωδικοποιηθεί στις λειτουργίες παράκαμψης ή συγχώνευσης (skip ή merge), το διάνυσμα κίνησης κωδικοποιείται με διαφορετικούς τρόπους πρόβλεψης διανύσματος κίνησης. Όμοια με την λειτουργία συγχώνευσης, το HEVC επιτρέπει στον κωδικοποιητή να επιλέξει το πρότυπο διάνυσμα κίνησης μεταξύ των πολλαπλών υποψήφιων μέσων πρόβλεψης. Η διαφορά μεταξύ του τρόπου πρόβλεψης και του πραγματικού διανύσματος, καθώς και του δείκτη του υποψήφιου μεταδίδονται στον αποκωδικοποιητή.
Μόνο δύο χωρικοί υποψήφιοι κίνησης επιλέγονται ανάλογα με την διαθεσιμότητα μεταξύ των πέντε υποψηφίων. Ο πρώτος χωρικός υποψήφιος κίνησης επιλέγεται από το σύνολο των αριστερών θέσεων (a_0,a_1) και ο δεύτερος από το σύνολο των παραπάνω θέσεων (b_0,b_1.b_2) , ανάλογα με την διαθεσιμότητά τους, διατηρώντας ταυτόχρονα τη σειρά αναζήτησης, όπως υποδεικνύεται στα δύο σύνολα.
Το HEVC επιτρέπει ένα μικρότερο αριθμό υποψηφίων που θα χρησιμοποιηθούν στη διαδικασία πρόβλεψης του διανύσματος κίνησης, δεδομένου ότι ο κωδικοποιητής μπορεί να στείλει μια κωδικοποιημένη διαφορά για να αλλάξει το διάνυσμα κίνησης. Πρόσθετα, ο κωδικοποιητής πρέπει να εκτελεί την εκτίμηση κίνησης, η οποία είναι μία από τις πιο ακριβές υπολογιστικά διεργασίες στον κωδικοποιητή, και η πολυπλοκότητα μειώνεται με το να επιτρέπει έναν μικρό αριθμό υποψηφίων.
Όταν ο δείκτης αναφοράς του γειτονικού PU δεν είναι ίσος με αυτόν του τρέχοντος PU, χρησιμοποιείται μια κλιμακωτή έκδοση του διανύσματος κίνησης. Το γειτονικό διάνυσμα κίνησης κλιμακώνεται σύμφωνα με τις χρονικές αποστάσεις μεταξύ της τρέχουσας εικόνας και των εικόνων αναφοράς που επισημαίνονται από τους δείκτες αναφοράς του γειτονικού PU και του τρέχοντος PU, αντίστοιχα. Όταν δύο χωρικοί υποψήφιοι έχουν τα ίδια στοιχεία διανύσματος κίνησης, εξαιρείται ένας χωρικός υποψήφιος.
Όταν ο αριθμός των προβλεπόμενων διανυσμάτων κίνησης δεν είναι ίσος με δύο και η χρήση της χρονικής πρόβλεψης MV δεν είναι απενεργοποιημένη, συμπεριλαμβάνεται ο υποψήφιος χρονικός MV. Αυτό σημαίνει ότι ο χρονικός υποψήφιος δεν χρησιμοποιείται καθόλου όταν δύο χωρικοί υποψήφιοι είναι διαθέσιμοι. Τέλος, ένα μηδενικό διάνυσμα κίνησης συμπεριλαμβάνεται επανειλημμένα έως ότου ο αριθμός των υποψηφίων διανυσμάτων κίνησης είναι ίσος με δύο, γεγονός που εγγυάται ότι ο αριθμός των προβλεπόμενων διανυσμάτων κίνησης είναι δύο. Επομένως, χρησιμοποιείται ένα flag για να προσδιοριστεί ποια είναι η πρόβλεψη διανύσματος.

3.4 Μεταφορά και κβαντοποίηση

Το HEVC χρησιμοποιεί κωδικοποίηση μετασχηματισμού με παρόμοιο τρόπο όπως σε προηγούμενα πρότυπα. Το μπλοκ διαφορών χωρίζεται σε πολλαπλά τετράγωνα TBs , όπως περιγράφηκε πιο πάνω. Τα υποστηριζόμενα μεγέθη μπλοκ μετασχηματισμού είναι 4×4,8×8,16×16, και 32×32.

3.4.1 Μετασχηματισμός

Οι δισδιάστατοι μετασχηματισμοί υπολογίζονται εφαρμόζοντας μετασχηματισμούς 1-D στις οριζόντιες και κάθετες κατευθύνσεις. Τα στοιχεία των πινάκων για τον μετασχηματισμό προέκυψαν από την προσέγγιση της κλιμάκωσης των συναρτήσεων βάσης DCT, κάτω από θεωρήσεις όπως η μεγιστοποίηση της ακρίβειας και η εγγύτητα της ορθογωνιότητας όταν οι καταχωρήσεις πίνακα καθορίζονται από ακέραιες τιμές. Για λόγους απλούστευσης, καθορίζεται μόνο ένας ακέραιος πίνακας για το μήκος 32 σημείων.
Αν και το πρότυπο καθορίζει τον μετασχηματισμό απλώς ως προς την τιμή του πίνακα, οι τιμές των καταχωρήσεων του πίνακα που επιλέχθηκαν ώστε να έχουν βασικές ιδιότητες συμμετρίας που επιτρέπουν ενσωματωμένες υλοποιήσεις με πολύ λιγότερες μαθηματικές πράξεις από ένα κανονικό πολλαπλασιασμό πινάκων , και οι μεγαλύτεροι μετασχηματισμοί μπορούν να κατασκευαστούν με τη χρήση των μικρότερων μετασχηματισμών ως κατασκευασμένα μπλοκ.
Λόγω του αυξημένου μεγέθους των υποστηριζόμενων μετασχηματισμών, είναι πολύ σημαντικός ο περιορισμός του δυναμικού εύρους των ενδιάμεσων αποτελεσμάτων από το πρώτο στάδιο του μετασχηματισμού. Το HEVC εισάγει μια δεξιά μετατόπιση 7-b και πράξη αποκοπής 16-b, μετά το πρώτο στάδιο μετατροπής της αντιστροφής 1-D του μετασχηματισμού (στάδιο κάθετου αντίστροφου μετασχηματισμού) για να διασφαλίσει ότι όλες οι ενδιάμεσες τιμές μπορούν να αποθηκευτούν στη μνήμη 16-b .

3.4.2 Εναλλακτικός 4×4 μετασχηματισμός

Για μετασχηματισμό μπλοκ μεγέθους 4x4, ένας εναλλακτικός ακέραιος μετασχηματισμός προερχόμενος από το DST εφαρμόζεται στα μπλοκ διαφορών luma για τις λειτουργίες πρόβλεψης intrapicure.
Οι βασικές συναρτήσεις του DST ταιριάζουν καλύτερα στη στατιστική ιδιότητα, που τα πλάτη διαφορών τείνουν να αυξάνουν, καθώς η απόσταση από τα συνοριακά δείγματα που χρησιμοποιούνται για την πρόβλεψη γίνονται μεγαλύτερα. Όσον αφορά την πολυπλοκότητα, ο 4x4 DST τρόπος μετασχηματισμού δεν είναι πιο απαιτητικός υπολογιστικά από τον 4x4 DCT τρόπο μετασχηματισμού και παρέχει περίπου 1% μείωση του bit-rate στη intrapicture πρόβλεψη κωδικοποίησης.
Η χρήση του τύπου μετασχηματισμού περιορίζεται μόνο σε 4x4 luma μπλοκ μετασχηματισμού, επειδή για τις άλλες περιπτώσεις είναι οριακή η απόδοση βελτίωσης της κωδικοποίησης.

3.4.3 Κβαντοποίηση

Για την κβαντοποίηση το HEVC χρησιμοποιεί ουσιαστικά το ίδιο σχήμα URQ ελεγχόμενο από μία παράμετρο κβαντοποίησης (QP), όπως στο H.264/MPEG-4 AVC. Το εύρος των τιμών QP ορίζεται από 0 έως 51 και μία αύξηση κατά 6 διπλασιάζει το βήμα κβαντοποίησης έτσι ώστε η χαρτογράφηση των τιμών QP στο μέγεθος των βημάτων να είναι περίπου λογαριθμική. Επίσης, υποστηρίζονται πίνακες κλιμάκωσης κβαντοποίησης. Για τη μείωση της μνήμης που απαιτείται για την αποθήκευση ειδικών για την συχνότητα τιμών κλιμάκωσης, χρησιμοποιούνται μόνο πίνακες κβαντοποίησης των μεγεθών 4x4 και 8x8. Για τους μεγαλύτερους μετασχηματισμούς των 16x16 και 32x32, ένας 8x8 πίνακας κλιμάκωσης στέλνεται και εφαρμόζεται με την κοινή χρήση μέσα στις

2x2 και 4x4 ομάδες συντελεστών σε υποσύστημα συχνοτήτων, εκτός από τις τιμές σε θέσεις DC (μηδενική συχνότητα), για τις οποίες αποστέλλονται και εφαρμόζονται ξεχωριστές τιμές.

3.5 Κωδικοποίηση Εντροπίας

Το HEVC καθορίζει μόνο μία μέθοδο κωδικοποίησης εντροπίας, την CABAC και όχι δύο όπως το H.264/MPEG-4 AVC. Ο πυρήνας του αλγόριθμου CABAC παραμένει αμετάβλητος, και οι ακόλουθες υποενότητες παρουσιάζουν διάφορες πτυχές του τρόπου με τον οποίο χρησιμοποιείται στο σχεδιασμό του HEVC.

3.5.1 Μοντελοποίηση Περιεχομένου

Η μοντελοποίηση περιβάλλοντος αποτελεί βασικό παράγοντα για τη βελτίωση της αποτελεσματικότητας της κωδικοποίησης CABAC. Στο HEVC, το βάθος διάσπασης του δέντρου κωδικοποίησης ή του δέντρου μετασχηματισμού αξιοποιείται για να εξάγει τους δείκτες των διάφορων στοιχείων σύνταξης. Για παράδειγμα, το στοιχείο σύνταξης skip_flag που καθορίζει αν το CB κωδικοποιείται ως intrapicture που έχει προγνωστικά παραλειφθεί και το στοιχείο σύνταξης split_coding_unit_flag που καθορίζει εαν το CB είναι περαιτέρω χωρισμένο κωδικοποιείται χρησιμοποιώντας μοντέλα περιβάλλοντος που βασίζονται στις πληροφορίες των χωρικών γειτόνων. Το στοιχείο σύνταξης split_transform_flag προσδιορίζει εαν το TB είναι περαιτέρω χωρισμένο και τα τρία στοιχεία σύνταξης που καθορίζουν μη μηδενικούς συντελεστές μετασχηματισμού για κάθε συνιστώσα χρώματος, cbf_luma, cbf_cb, cbf_cr, κωδικοποιούνται βάσει του βάθους διάσπασης του δέντρου μετασχηματισμού. Αν και ο αριθμός των πλασίων που χρησιμοποιούνται στο HEVC είναι ουσιαστικά μικρότερος από H.264/MPEG-4 AVC, ο σχεδιασμός κωδικοποίησης εντροπίας προσφέρει στην πράξη καλύτερη συμπίεση απ ότι μια απλή επέκταση του συστήματος του H.264/MPEG-4 AVC. Επιπλέον γίνεται πιο εκτεταμένη χρήση στο HEVC του τρόπου παράκαμψης της λειτουργίας CABAC για να αυξηθεί το throughput μειώνοντας την ποσότητα των δεδομένων που πρέπει να κωδικοποιηθούν χρησιμοποιώντας τα πλαίσια CABAC. Οι εξαρτήσεις μεταξύ των κωδικοποιημένων δεδομένων εξετάζονται επίσης προσεκτικά ώστε να καταστεί δυνατό το μεγαλύτερο throughput.

3.5.2 Σάρωση Συντελεστών

Η σάρωση συντελεστών πραγματοποιείται στο 4x4 υπο-μπλοκ για όλα τα μεγέθη TB (χρησιμοποιώντας μόνο μια περιοχή συντελεστών για το 4x4 TB μέγεθος, και χρησιμοποιώντας πολλαπλά 4x4 συντελεστές στις περιοχές μεγαλύτερου μετασχηματισμού μπλοκ). Τρεις μέθοδοι σάρωσης συντελεστών, διαγώνιες άνω δεξιά , οριζόντιες και κάθετες σαρώσεις επιλέγονται για την κωδικοποίηση των συντελεστών μετασχηματισμού των 4x4 και 8x8 μεγάθη TB στις προβλεπόμενες intrapicture περιοχές. Η επιλογή της σειράς σάρωσης συντελεστών εξαρτάται από τις κατευθύνσεις της Intrapicture πρόβλεψης. Η κάθετη σάρωση χρησιμοποιείται όταν η κατεύθυνση πρόβλεψης είναι κοντά στην οριζόντια, και η οριζόντια σάρωση χρησιμοποιείται όταν η κατεύθυνση πρόβλεψης είναι κοντά στην κάθετη. Για τις άλλες κατευθύνσεις πρόβλεψης χρησιμοποιείται η διαγώνια επάνω δεξιά σάρωση.
Για τους συντελεστές μετασχηματισμού στην Interpicture λειτουργία πρόβλεψης για όλα τα μεγέθη μπλοκ και για τους συντελεστές μετασχηματισμού των 16x16 ή 32x32 intrapicure πρόβλεψης , η 4x4 διαγώνια πάνω δεξιά σάρωση εφαρμόζεται στα υπο-μπλοκ των συντελεστών μετασχηματισμού.

3.5.3 Κωδικοποίηση Συντελεστών

Παρόμοια με το H.264/MPEG-4 AVC, το HEVC μεταδίδει τη θέση του τελευταίου μη μηδενικού συντελεστή μετασχηματισμού, ένα χάρτη με τα σημαντικά σημεία, σημειώνοντας τα bit και το επίπεδο των συντελεστών μετασχηματισμού. Ωστόσο έγιναν διάφορες αλλαγές για κάθε μέρος, ειδικά για την καλύτερη διαχείριση του αυξημένου μεγέθους των TBs. Πρώτον, οι οριζόντιες και οι κάθετες θέσεις των συντεταγμένων του τελευταίου μη μηδενικού συντελεστή κωδικοποιούνται για το TB πριν την αποστολή των χαρτών σημαντικών σημείων των 4x4 υπο-μπλοκ που υποδεικνύουν ποιοι άλλοι συντελεστές μετασχηματισμού έχουν μη μηδενικές τιμές, αντί να στέλνουν μια σειρά σημάτων ταυτοποίησης τελευταίου συντελεστή που παρεμβάλλονται με το χάρτη σημαντικών σημείων όπως γίνεται στο H.264/MPEG-4 AVC.
Ο χάρτης σημαντικών σημείων παράγεται για ομάδες σημαντικών σημείων που σχετίζονται με το σταθερό μέγεθος 4x4 των υπο-μπλοκ. Για όλες τις ομάδες που έχουν τουλάχιστον έναν συντελεστή που προηγείται της τελευταίας θέσης συντελεστών, μεταδίδεται ένα flag για την ομάδα σημαντικών σημείων που προσδιορίζει μια ομάδα μη μηδενικού συντελεστή , ακολουθούμενη από coefficient significance flags για κάθε συντελεστή πριν από τη υποδεικνυόμενη θέση τελευταίου του τελευταίου significant coefficient. Τα μοντέλα περιεχομένου για τα flags των significant coefficient εξαρτώνται τη θέση συντελεστή καθώς και από τις τιμές των δεξιά και κάτω flags των significant group.
Για την περαιτέρω βελτίωση της συμπίεσης χρησιμοποιείται μια μέθοδος που είναι γνωστή sign data hiding (απόκρυψη δεδομένων σήματος). Τα bits σημάτων κωδικοποιούνται υπό όρους με βάση τον αριθμό και τις θέσεις των κωδικοποιημένων συντελεστών. Όταν χρησιμοποιείται η sign data hiding και υπάρχουν τουλάχιστον δυο μη μηδενικοί συντελεστές σε ένα 4x4 υπομπλοκ και η διαφορά μεταξύ των θέσεων σάρωσης του πρώτου και του τελευταίου μη μηδενικού συντελεστή είναι μεγαλύτερη του τρία, το bit σήματος του πρώτου μη μηδενικού συντελεστή συνάγεται από την ισοτιμία του αθροίσματος των συντελεστών πλάτους. Διαφορετικά, το bit σήματος κωδικοποιείται κανονικά. Στην πλευρά του κωδικοποιητή, αυτό μπορεί να υλοποιηθεί επιλέγοντας έναν συντελεστή με ένα πλάτος κοντά στο όριο του διαστήματος της κβαντοποίησης που θα αναγκαστεί να χρησιμοποιήσει το γειτονικό διάστημα κβαντοποίησης, σε περιπτώσεις όπου η ισοτιμία δεν θα έδειχνε αλλιώς το σωστό σήμα του πρώτου συντελεστή. Αυτό επιτρέπει το bit σήματος να κωδικοποιείται με χαμηλό κόστος απ’ ότι εάν κωδικοποιήθηκε ξεχωριστά, δίνοντας στον κωδικοποιητή την ελευθερία να επιλέξει σε ποιο εύρος συντελεστή κωδικοποίησης μπορεί να μεταβληθεί με το χαμηλότερο κόστος παραμόρφωσης.
Για κάθε θέση όπου το αντίστοιχο flag significant coefficient είναι ίσο με μηδέν, κωδικοποιούνται δύο flags που καθορίζουν αν η τιμή επιπέδου είναι μεγαλύτερη από μια ή δυο και τότε η τιμή επιπέδου κωδικοποιείται ανάλογα με αυτές τις δυο τιμές.

3.6 Φίλτρα (In-Loop Filters)

Στο HEVC, εφαρμόζονται στην ανακατασκευή δειγμάτων δυο βήματα επεξεργασίας, ένα φίλτρο deblocking (DBF) ακολουθούμενο από ένα φίλτρο SAO, προτού εγγραφούν στο buffer αποκωδικοποιημένων εικόνων στο βρόχο του αποκωδικοποιητή. Το DBF είναι παρόμοιο με αυτό του προτύπου H.264/MPEG-4 AVC, ενώ το SAO εισάγεται στο HEVC. Ενώ το DBF εφαρμόζεται μόνο στα δείγματα που βρίσκονται στα όρια των μπλοκ, το φίλτρο SAO εφαρμόζεται προσαρμοστικά σε όλα τα δείγματα που ικανοποιούν ορισμένες συνθήκες, π.χ με βάση της κλήση.

3.6.1 Φίλτρο Deblocking

Το deblocking φίλτρο εφαρμόζεται σε όλα τα δείγματα που είναι γειτονικά με ένα PU ή TU όριο εκτός από την περίπτωση που ένα όριο είναι επίσης ένα όριο εικόνας ή όταν το deblocking

απενεργοποιείται στα όρια slice ή tile (η οποία είναι μια επιλογή που μπορεί να σηματοδοτηθεί από τον κωδικοποιητή). Θα πρέπει να σημειωθεί ότι τόσο τα όρια PU και TU πρέπει να ληφθούν υπόψη αφού τα όρια PU δεν είναι πάντα ευθυγραμμισμένα με τα όρια TU σε μερικές περιπτώσεις interpicture πρόβλεψης CBs. Τα στοιχεία σύνταξης στο SPS και τα headers των slice ελέγχουν αν το φίλτρο deblocking εφαρμόζεται σε όλα τα όρια του slice και του tile.
Σε αντίθεση με το H.264/MPEG-4 AVC, όπου το deblocking φίλτρο εφαρμόζεται σε πλέγμα δειγμάτων 4x4, το HEVC εφαρμόζει το φίλτρο μόνο στις άκρες που είναι ευθυγραμμισμένες σε ένα 8x8 πλέγμα δείγματος, για τα δείγματα luma και chroma. Αυτός ο περιορισμός μειώνει τη χειρότερη περίπτωση υπολογιστικής πολυπλοκότητας χωρίς αξιοσημείωτη υποβάθμιση της οπτικής ποιότητας. Επίσης βελτιώνει την λειτουργία παράλληλης επεξεργασίας, αποτρέποντας την αλληλεπίδραση μεταξύ των πράξεων φιλτραρίσματος.
Η ισχύς του deblocking φίλτρου ελέγχεται από τις τιμές αρκετών στοιχείων σύνταξης παρόμοια με το H.264/MPEG-4 AVC, αλλά χρησιμοποιούνται μόνο τρεις δυνάμεις αντί για πέντε. Δεδομένου ότι τα P και Q είναι δυο γειτονικά μπλοκ με ένα κοινό όριο 8x8 πλέγματος, η δύναμη του φίλτρου 2 αποδίδεται όταν ένα από τα μπλοκ προβλέφθηκε ως intrapicture. Διαφορετικά η δύναμη του φίλτρου 1 αντιστοιχεί σε περίπτωση που πληρούνται οποιασδήποτε από τις ακόλουθες συνθήκες.

1. Το P ή το Q έχει τουλάχιστον ένα μη μηδενικό συντελεστή μετασχηματισμού
2. Ο δείκτης αναφοράς του P και Q δεν είναι ίσοι.
3. Τα διανύσματα κίνησης του P και Q δεν είναι ίσα
4. Η διαφορά μεταξύ ενός διανύσματος κίνησης του P και Q είναι μεγαλύτερο ή ίσο από ένα ακέραιο δείγμα.

Εάν δεν πληρούνται καμία από τις παραπάνω συνθήκες, αποδίδεται η δύναμη φίλτρου 0, πράγμα που σημαίνει ότι ότι δεν εφαρμόζεται η διαδικασία deblocking. Σύμφωνα με τη δύναμη του φίλτρου και τη μέση παράμετρο κβαντοποίησης των P και Q, δυο κατώφλια, tC και β, προσδιορίζονται από προκαθορισμένους πίνακες. Για τα luma δείγματα, μια από τις τρεις περιπτώσεις, χωρίς φιλτράρισμα, ισχυρό φιλτράρισμα, ασθενές φιλτράρισμα, επιλέγεται με βάση το β. Σημειώστε ότι αυτή η απόφαση μοιράζεται σε τέσσερεις σειρές ή στήλες luma, χρησιμοποιώντας τις πρώτες και τις τελευταίες σειρές ή στήλες για να μειωθεί η υπολογιστική πολυπλοκότητα. Υπάρχουν μόνο δυο περιπτώσεις, χωρίς φιλτράρισμα και κανονικό φιλτράρισμα για δείγματα chroma. Το κανονικό φιλτράρισμα εφαρμόζεται μόνο όταν το η δύναμη του φίλτρου είναι μεγαλύτερη από ένα. Η διαδικασία φιλτραρίσματος διεξάγεται χρησιμοποιώντας τις μεταβλητές ελέγχου tC και β.
Στο HEVC, η σειρά επεξεργασίας του deblocking φίλτρου ορίζεται ως οριζόντιο φιλτράρισμα για τις κατακόρυφες ακμές για ολόκληρη την εικόνα, ακολουθούμενη από κάθετο φιλτράρισμα για οριζόντιες ακμές. Αυτή η συγκεκριμένη σειρά δίνει τη δυνατότητα είτε πολλαπλών διαδικασιών οριζόντιου φιλτραρίσματος είτε κάθετου φιλτραρίσματος να εφαρμόζονται σε παράλληλα νήματα, ή ακόμα να μπορούν να εφαρμοστούν σε βάση CTB-by-CTB με μικρή καθυστέρηση επεξεργασίας (latency).

3.6.2 SAO

Το SAO είναι μια διαδικασία που τροποποιεί τα αποκωδικοποιημένα δείγματα προσθέτοντας υπό όρους μια τιμή μετατόπισης (offset) σε κάθε δείγμα μετά την εφαρμογή του deblocking φίλτρου με βάση τις τιμές των πινάκων αναζήτησης από τον κωδικοποιητή. Το φιλτράρισμα SAO εκτελείται σε μια βάση περιοχής, βάσει ενός τύπου φιλτραρίσματος που επιλέγεται ανά CTB από ένα στοιχείο σύνταξης sao_type_idx. Μια τιμή 0 για το sao_type_idx υποδεικνύει ότι το φίλτρο SAO δεν εφαρμόζεται στο CTB, και οι τιμές 1 και 2 σηματοδοτούν τη χρήση των τύπων φιλτραρίσματος μετατόπισης ζώνης (band offset) και μετατόπισης άκμης (edge offset), αντίστοιχα.
Στη κατάσταση μετατόπισης ζώνης που καθορίζεται από το sao_type_idx ίσο με 1, η επιλεγμένη τιμή μετατόπισης εξαρτάται άμεσα από το πλάτος δείγματος. Σε αυτή τη κατάσταση, το πλήρες

εύρος πλάτους δείγματος διαιρείται ομοιόμορφα σε 32 ομοιόμορφα τμήματα που ονομάζονται ζώνες (bands), και οι τιμές των δειγμάτων που ανήκουν στις τέσσερεις από αυτές τις ζώνες (οι οποίες είναι διαδοχικές εντός των 32 ζωνών) τροποποιούνται με την προσθήκη τιμών που μεταδίδονται και υποδηλώνονται ως μετατοπίσεις ζώνης, οι οποίες μπορεί να είναι θετικές ή αρνητικές. Ο κύριος λόγος για τη χρήση τεσσάρων διαδοχικών ζωνών είναι ότι στις ομαλές περιοχές όπου μπορούν να εμφανιστούν αντικείμενα συνένωσης, τα πλάτη δείγματος σε ένα CTB τείνουν να συγκεντρώνονται σε λίγες μόνο από τις ζώνες. Επιπλέον η επιλογή σχεδιασμού με τη χρήση τεσσάρων μετατοπίσεων ενεργοποιείται με τον τρόπο λειτουργίας μετατόπισης ακμής που χρησιμοποιεί επίσης τέσσερεις τιμές μετατόπισης.
Στην κατάσταση μετατόπισης ακμής που καθορίζεται από sao_type_idx ίσο με το 2, το στοιχείο σύνταξης sao_eo_class με τιμές από από 0 έως 3 σηματοδοτεί κατά πόσο μια οριζόντια, κάθετη ή μια από τις δύο διαγώνιες κατευθύνσεις χρησιμοποιείται για την ταξινόμηση της μετατόπισης ακμής στο CTB. Κάθε δείγμα στο CTB κατατάσσεται σε μια από τις πέντε κατηγορίες EdgeIdx συγκρίνοντας την τιμή δείγματος p που βρίσκεται σε κάποια θέση με τις τιμές n0 και n1 των δύο δειγμάτων που βρίσκονται σε γειτονικές θέσεις (πίνακας το δείχνει). Αυτή η ταξινόμηση γίνεται για κάθε δείγμα βασισμένο στις τιμές αποκωδικοποιημένων δειγμάτων, έτσι δεν απαιτείται επιπλέον σηματοδότηση για την ταξινόμηση EdgeIdx. Ανάλογα με την κατηγορία EdgeIdx στη θέση δείγματος, για τις κατηγορίες EdgeIdx από 1 μέχρι 4 προστίθεται στην τιμή δείγματος η τιμή μετατόπισης από ένα μεταδιδόμενο πίνακα αναζήτησης. Οι τιμές μετατόπισης είναι πάντα θετικές για τις κατηγορίες 1 και 2 και αρνητική για τις κατηγορίες 3 και 4 . Έτσι γενικά το φίλτρο έχει ως αποτέλεσμα την εξομάλυνση στη λειτουργία της μετατόπισης ακμής.
Έτσι, για τους τύπους SAO 1 και 2, μεταφέρονται συνολικά τέσσερεις τιμές μετατόπισης πλάτους στον αποκωδικοποιητή για κάθε CTB. Για τον τύπο 1, το σήμα είναι επίσης κωδικοποιημένο. Οι τιμές μετατόπισης και τα συσχετιζόμενα στοιχεία σύνταξης όπως sao_type_idx και sao_eo_class καθορίζονται από τον κωδικοποιητή – τυπικά χρησιμοποιώντας κριτήρια που βελτιστοποιούν την την απόδοση του ρυθμού παραμόρφωσης. Οι παράμετροι SAO μπορεί να υποδεικνύονται για να κληρονομηθούν από τον αριστερό ή πάνω CTB χρησιμοποιώντας ένα flag συγχώνευσης για να καταστήσει αποτελεσματική την σηματοδότηση. Συνοπτικά, το SAO είναι μια μη γραμμική λειτουργία φιλτραρίσματος η οποία επιτρέπει την περαιτέρω βελτίωση του ανακατασκευασμένου σήματος και μπορεί να ενισχύσει την αναπαράσταση σήματος τόσο στις ομαλές περιοχές όσο και στις άκρες.

4. Βελτιστοποίηση των Συναρτήσεων Κόστους του Κωδικοποιητή

4.1 Συναρτήσεις Κόστους

Διάφορες συναρτήσεις κόστους χρησιμοποιούνται στον κωδικοποιητή λογισμικού HM για να προσδιορίσουν το κόστος που χρειάζεται για την λήψη των αποφάσεων του κωδικοποιητή. Παρακάτω αναφέρονται οι συναρτήσεις κόστους που χρησιμοποιούνται στη διαδικασία κωδικοποίησης του λογισμικού HM.

4.1.1 Άθροισμα τετραγώνου σφάλματος (SSE)

Η διαφορά μεταξύ δύο μπλοκ με το ίδιο μέγεθος μπλοκ παράγεται χρησιμοποιώντας

Diff(i,j) = BlockA(i,j) - BlockB(i,j)

Η SSE υπολογίζεται από την ακόλουθη εξίσωση

$$SSE = \sum_{i,j} Diff(i,j)^2$$

4.1.2 Άθροισμα απόλυτων διαφορών (SAD)

Το SAD υπολογίζεται χρησιμοποιώντας την ακόλουθη εξίσωση:

$$SAD = \sum_{i,j} |Diff(i,j)|$$

Περιπτώσεις όπου χρησιμοποιείται το άθροισμα απόλυτων διαφορών είναι οι παρακάτω :

Πρόβλεψη Διανύσματος Κίνησης (Motion Vector Prediction):

Για κάθε PU, το καλύτερο διάνυσμα πρόβλεψης κίνησης υπολογίζεται με τη διαδικασία που καθορίζεται ως εξής. Πρώτον, ένα σύνολο υποψήφιων προγνωστικών διανυσμάτων κίνησης, αναφέρει τις παραμέτρους κίνησης των γειτονικών PU.

Στη συνέχεια, το καλύτερο από το σύνολο των υποψήφιων καθορίζεται από ένα κριτήριο που επιλέγει έναν υποψήφιο προγνωστικού διανύσματος κίνησης που ελαχιστοποιεί το κόστος $J_{pred,\,SAD}$.

Εκτίμηση Κίνησης (Motion estimation):

Για να ληφθεί διάνυσμα κίνησης για κάθε PU, ο αλγόριθμος αντιστοίχισης μπλοκ (BMA) πραγματοποιείται στον κωδικοποιητή. Η ακρίβεια του διανύσματος κίνησης που υποστηρίζεται στο HEVC είναι quarter-pixel. Για τη δημιουργία δειγμάτων ακρίβειας half-pixel και quarter-pixel, διεξάγεται φιλτράρισμα παρεμβολής στα δείγματα εικόνων αναφοράς.

Αντί να αναζητήσουμε όλες τις θέσεις για την κίνηση quarter-pixel, αρχικά εφαρμόζεται ακρίβεια integer-pixel στα διανύσματα κίνησης . Για την αναζήτηση half-pixel, αναζητούνται μόνο 8

δείγματα γύρω από το διάνυσμα κίνησης που έχει το ελάχιστο κόστος. Παρομοίως, για την αναζήτηση quarter-pixel, εξετάζονται 8 δείγματα γύρω από την κίνηση που έχει το ελάχιστο κόστος μέχρι στιγμής. Το διάνυσμα κίνησης που έχει το ελάχιστο κόστος επιλέγεται ως διάνυσμα κίνησης της PU. Για να υπολογίσουμε το κόστος, χρησιμοποιείται το SAD για αναζήτηση κίνησης integer-pixel και η SA(T)D χρησιμοποιείται για αναζήτηση κίνησης half-pixel και quarter-pixel.

Πρόβλεψη διαφορών στη περίπτωση transquant bypass και transform skip

Όταν χρησιμοποιείται κωδικοποίηση χωρίς απώλειες (δηλ. Όταν Cu_transquant_bypass_flag είναι ίσο με ένα) και implicit_rdpcm_enabled_flag είναι ίσο με ένα, οι διαφορές που λαμβάνονται από την intra πρόβλεψη, προβλέπονται περαιτέρω χρησιμοποιώντας DPCM. Το σήμα της διαφοράς DPCM εφαρμόζεται μόνο όταν η κατεύθυνση της intra πρόβλεψης είναι είτε οριζόντια είτε κάθετη. Για τις διαφορές που λαμβάνονται από την inter πρόβλεψη, το RDPCM εφαρμόζεται εάν το explicit_rdpcm_enabled_flag είναι ίσο με ένα. Σε αυτή την περίπτωση ο κωδικοποιητής επιλέγει εάν θα εφαρμόσει το RDPCM στις διαφορές και εάν εφαρμοστεί, αν θα εκτελέσει το RDPCM κατά μήκος της οριζόντιας ή κατακόρυφης κατεύθυνσης. Η απόφαση αυτή βασίζεται στο άθροισμα της απόλυτης διαφοράς (SAD). Η απόφαση που ελαχιστοποιεί το SAD επιλέγεται ως η καλύτερη και υποδεικνύεται στον αποκωδικοποιητή χρησιμοποιώντας δύο δυαδικά flags: ένα για να δηλώνει εάν εφαρμόζεται το RDPCM και ένα για να σηματοδοτήσει την κατεύθυνση στην οποία εφαρμόζεται το RDPCM.

4.1.3 Hadamard transformed SAD (SATD)

Δεδομένου ότι οι μετασχηματισμένοι συντελεστές κωδικοποιούνται, μπορεί να επιτευχθεί μια βελτιωμένη εκτίμηση του κόστους κάθε τρόπου με την εκτίμηση του DCT με το μετασχηματισμό Hadamard.

Το SATD υπολογίζεται χρησιμοποιώντας:

$$SATD = (\sum_{i,j} |DiffT(i,j)|) / 2$$

Υπάρχει ένα flag μετασχηματισμού Hadamard έτσι ώστε να ενεργοποιείται ή να απενεργοποιείται ο μετασχηματισμός αυτός. SA(T)D αναφέρεται είτε σε SAD είτε σε SATD ανάλογα με την τιμή του flag μετασχηματισμού Hadamard.
Το SAD χρησιμοποιείται όταν υπολογίζεται η εκτίμηση κίνησης full-pixel ενώ η SA(T)D χρησιμοποιείται για την εκτίμηση κίνησης sub-pixel.

Περιπτώσεις όπου χρησιμοποιείται το SATD:

Κωδικοποίηση CU με MODE_INTER:

Όταν ένα CU κωδικοποιείται με MODE_INTER, η απόφαση για την παράμετρο κίνησης του κάθε PU εκτελείται πρώτα με βάση το κόστος $J_{pred,\ SATD}$.

Για την περίπτωση λειτουργίας συγχώνευσης, η απόφαση για την παράμετρο κίνησης αρχίζει με τον έλεγχο των διαθέσιμων όλων των γειτονικών PU για να σχηματίσει υποψήφιους συγχώνευσης. Εάν δεν υπάρχει διαθέσιμος υποψήφιος συγχώνευσης, ο κωδικοποιητής HM απλά παραλείπει τον

υπολογισμό του κόστους για τη λειτουργία συγχώνευσης και δεν επιλέγει τη λειτουργία συγχώνευσης για το τρέχων PU. Διαφορετικά (εάν υπάρχει τουλάχιστον ένας υποψήφιος συγχώνευσης), υπολογίζεται το κόστος $J_{pred,SATD}$ για όλα τα πιθανά PUs ως υποψήφιους για συγχώνευση και το καλύτερο επιλέγεται ως η καλύτερη παράμετρος κίνησης για το προβλεπόμενο PU. Το SATD μεταξύ δειγμάτων προέλευσης και πρόβλεψης χρησιμοποιείται ως συντελεστής παραμόρφωσης.
Για την περίπτωση Inter λειτουργίας, προκύπτουν οι καλύτερες παράμετροι κίνησης εφαρμόζοντας τη διαδικασία εκτίμησης κίνησης. Κατά τη διάρκεια της διαδικασίας εκτίμησης κίνησης επιτυγχάνονται οι καλύτεροι παράμετροι κίνησης με βάση τη συνάρτηση κόστους $J_{pred,SATD}$. Το SATD μεταξύ δειγμάτων προέλευσης και πρόβλεψης χρησιμοποιείται ως συντελεστής παραμόρφωσης. Αφού ληφθούν οι καλύτεροι παραμέτροι κίνησης, τότε προσδιορίζονται συγκρίνοντάς τες και παίρνοντας την καλύτερη που έχει ως αποτέλεσμα χαμηλότερο κόστος.

4.2 Συναρτήσεις RD (Rate–Distortion)

4.2.1. Συνάρτηση κόστους βασισμένη στο SAD για την απόφαση παραμέτρου πρόβλεψης

Το κόστος για την απόφαση της παράμετρου πρόβλεψης $J_{pred,\ SAD}$ καθορίζεται από τον ακόλουθο τύπο:

$$J_{pred,SAD} = SAD + \lambda_{pred} * B_{pred}$$

4.2.2 Συνάρτηση κόστους SATD για την απόφαση της παραμέτρου πρόβλεψης

Το κόστος για την απόφαση της παραμέτρου κίνησης $J_{pred,\ SATD}$ καθορίζεται από τον ακόλουθο τύπο:

$$J_{pred,SATD} = SATD + \lambda_{pred} * B_{pred}$$

4.2.3 Συνάρτηση κόστους για την απόφαση λειτουργίας

Το κόστος για τη λήψη απόφασης Jmode καθορίζεται από τον ακόλουθο τύπο.

$$J_{mode} = (SSE_{luma} + w_{chroma} * SSE_{chroma}) + \lambda_{mode} * B_{mode}$$

όπου το B_{pred} καθορίζει το κόστος bit που πρέπει να ληφθεί για τη λήψη απόφασης, το οποίο εξαρτάται ανάλογα με την κάθε περίπτωση απόφασης
και με $\lambda_{pred} = = \sqrt{\lambda_{mode}}$

όπου $\lambda_{mode} = \alpha * W_k * 2^{((QP-12)/3.0)}$

Περίπτωση που συναντάμε το υπολογισμό του κόστους για την λήψη απόφασης J_{mode} είναι:

Intra prediction καταστάσεις και παράμετροι

Όταν ο κωδικοποιητής επιλέγει μια λειτουργία Intra prediction για ένα PU το κάνει ως εξής:

1. Ο υπολογισμός του υποψήφιου τρόπου πρόβλεψης εξετάζει όλους τους πιθανούς τρόπους πρόβλεψης για το luma PB με ένα κατά προσέγγιση κόστος πρόβλεψης $J_{pred,\ SATD}$ που περιγράψαμε

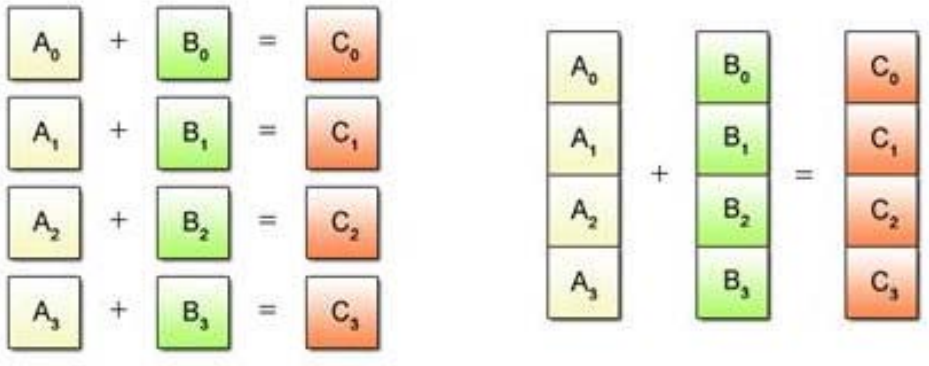
πιο πάνω. Ένας προκαθορισμένος αριθμός ενδιάμεσων υποψηφίων βρίσκεται για κάθε μέγεθος PU (8 για τα 4x4 και 8x8 PUs, 3 για άλλα μεγέθη PU). Σε αυτό το βήμα, ο αριθμός των κωδικοποιημένων bits για ένα Intra prediction τρόπο ορίζεται ίσο με Bpred.

2. Η βελτιστοποίησης RD, όπου χρησιμοποιείται το κόστος κωδικοποίησης J_{mode} που περιγράψαμε πιο πάνω, εφαρμόζεται στους προηγούμενους προσδιορισμένους υποψήφιους τρόπους. Κατά τη διάρκεια αυτού του βήματος, οι παράμετροι πρόβλεψης και οι συντελεστές για το στοιχείο luma της PU συγκεντώνονται στο Bmode. Όσον αφορά την απόφαση του τρόπου chroma PB, αξιολογούνται όλοι οι τρόποι πρόβλεψης intra chroma μέσω της διαδικασίας απόφασης RD.

4.3 Χρήση των εντολών SIMD

4.3.1 Εισαγωγή

Οι εντολές SIMD (Single instruction, multiple data), επιτρέπουν την επεξεργασία πολλαπλών στοιχείων σε μία μόνο εντολή. Αντίθετα, η συμβατική ακολουθιακή προσέγγιση χρησιμοποιεί μια εντολή για τη διεκπεραίωση κάθε μεμονωμένου στοιχείου.
Οι SIMD εντολές βρίσκουν μεγάλη απήχηση σε εφαρμογές γραφικών, ψηφιακής επεξεργασίας εικόνας και βίντεο, που απαιτούν απλούς, επαναλαμβανόμενους υπολογισμούς τεράστιων ποσοτήτων δεδομένων.



Σχήμα 12: Πρόσθεση δεδομένων ανά εντολή (αριστερά) και παράλληλη επεξεργασία με SIMD εντολές (δεξιά)

Πρόκειται για συναρτήσεις γραμμένες σε assembly που επιτρέπεται να καλούνται ως συναρτήσεις και μεταβλητές από γλώσσες προγραμματισμού υψηλού επιπέδου όπως η C και η C++. Θα χρησιμοποιηθούν οι εντολές Streaming SIMD Extensions (SSE), SSE2, οι εντολές Advanced Vector Extensions (AVX) καθως και AVX2. Πρόκειται για επεκτάσεις της αρχιτεκτονικής x86 για τους επεξεργαστές τις Intel και της AMD. Οι AVX παρέχουν νέα χαρακτηριστικά και νέες εντολές από τις SSE και SSE2 καθώς χρησιμοποιούνται οι μεγαλύτεροι σε μέγεθος καταχωρητές των 256 bits. Οι AVX2 που είναι μεταγενέστερες των υπολοίπων αποτελούν επέκταση αυτών με τις εντολές επεξεργασίας συναρτήσεων ακεραίων.

Σε αυτή την ενότητα περιγράφεται ο βελτιστοποιημένος κωδικοποιητής HEVC που χρησιμοποιεί ως βασική γραμμή για τη βελτιστοποίηση τις εντολές SIMD. Χρησιμοποιείται ως λογισμικό αναφοράς στα πειράματα ο HM16.7 που δεν χρησιμοποιεί SIMD εντολές στις συναρτήσεις κόστους. Επίσης λαμβάνεται υπ’ όψην και η τελευταία έκδοση του λογισμικού αναφοράς ο HM16.15 που χρησιμοποιεί εντολές SIMD, οι οποίες έχουν επεκταθεί με τις εντολές AVX και AVX2.

4.3.2 Αλλαγή των συναρτήσεων xGetSAD

Αλλαγή των συναρτήσεων SAD τροποποιώντας τις πράξεις υπολογισμού με χρήση των εντολών SIMD. Αξιοποιώντας τους μεγαλύτερους καταχωρητές που προσφέρονται από τους επεξεργαστές της Intel επιδιώκεται η επιτάχυνση των πράξεων μεταξύ των pixels.

Το HM απαριθμεί δέκα συναρτήσεις υπολογισμού των απόλυτων διαφορών SAD που αναφέρονται ως TcomRdCost::xGetSAD. Οι συναρτήσεις αυτές παίρνουν σαν παραμέτρους τα pixels για να υπολογίσουν τη μεταξύ τους διαφορά και να επιστρέψουν έπειτα το απόλυτο άθροισμα των διαφορών. Επίσης ορίζεται μια παράμετρο που υποδεικνύει πόσα από τα pixels χρειάζεται να επιλεχθούν αντίστοιχα σε κάθε συνάρτηση. Συγκεκριμένα:

Εισαγωγή στην xGetSAD4

Στην xGetSAD4 θα χρειαστούν 4 pixels όπου το κάθε ένα έχει μέγεθος 16 bits με σκοπό να υπολογίσει συνολικά τέσσερεις διαφορές SAD. Επομένως οι καταχωρητές που μπορούν να χωρέσουν και τα 4 pixels επιταχύνοντας τη διαδικασία είναι οι _m128i της κατηγορίας SSE και SSE2.
Παρατηρούμε ότι:

4 pixels x 16 bits το κάθε pixel = 64 bits,

που χωράει στους _m128 καταχωρητές παρέχοντας ταυτόχρονα όλες εκείνες τις συναρτήσεις που απαιτούνται. Όπως είναι:

- _mm_loadl_epi64 που θα φορτώσει από την μνήμη τα συνολικά 64 bits που χρειάζονται.
- _mm_subs_epi16 που υπολογίζει την διαφορά διαβάζοντας ανά 16 bits δεδομένα μέσα από τον καταχωρητή.
- _mm_adds_epu16 την ανάγνωση ανά 16 bits δεδομένων προσθέτει τα τέσσερα pixels.

Έτσι εκτελούνται σε λίγοτερους κύκλους μηχανής οι πράξεις πρόσθεσης, αφαίρεσης, η εύρεση ελαχίστου και μεγίστου καθώς και η απόλυτη τιμή.

Εισαγωγή στην xGetSAD8

Με τον ίδιο τρόπο υπολογισμού, είναι η περίπτωση της xGetSAD8 που χρειάζονται 8 pixels για την εύρεση του SAD. Οι SIMD εντολές της κατηγορίας SSE και SSE2 μπορούν να δώσουν λύση βελτιστοποιώντας την συνάρτηση.

Παρατηρήσουμε ότι:

8 pixels x 16 bits το κάθε pixel = 128 bits,

όσο και το μέγεθος των καταχωρητών _m128 προσφέροντας τις αντίστοιχες συναρτήσεις. Εδώ επιλέγουμε την :

_mm_loadu_si128 φορτώνοντας τα 128 bits που απαιτούνται και χρησιμοποιώντας παράλληλα και άλλες συναρτήσεις όπως αυτές που αναφέραμαι πιο πάνω.

Εισαγωγή στις xGetSAD16, xGetSAD16n,

Στις συναρτήσεις xGetSAD16 και xGetSAD16n χρειάζονται 16 pixels αντίστοιχα. Παρατηρούμε ότι οι μεγαλύτεροι καταχωτητές τις Intel, οι AVX και AVX2 είναι αυτοί που θα χρησιμοποιηθούν σε αυτή την περίπτωση και θα βελτιστοποιήσουν την απόδοση του λογισμικού. Είναι καταχωρητές μεγαλύτεροι σε μέγεθος που μπορούν να ικανοποιήσουν τις αυξημένες απαιτήσεις σε όγκο δεδομένων. Συγκεκριμένα:

16 pixels x 16 bits το κάθε pixel = 256 bits,

δηλαδή όσο είναι σε μέγεθος και οι καταχωρητές _m256. Μερικές από τις συναρτήσεις που προστέθηκαν είναι:

- _mm256_loadu_si256: φορτώνονται τα 256 bits δεδομένων που απαιτούνται
- _mm256_subs_epi16: αφαιρούνται τα pixels διαβάζοντάς τα ανα 16 bits που είναι και το μέγεθος τους
- _mm256_adds_epu16: προστίθενται τα pixels διαβάζοντάς τα ανα 16 bits που είναι και το μέγεθος τους

Αρχικός κώδικας, που χρησιμοποιεί τον συμβατικό τρόπο για τις πράξεις μεταξύ των στοιχείων

```
466
467 Distortion TComRdCost::xGetSAD16( DistParam* pcDtParam )
468 {
469   if ( pcDtParam->bApplyWeight )
470   {
471     return TComRdCostWeightPrediction::xGetSADw( pcDtParam );
472   }
473   const Pel* piOrg   = pcDtParam->pOrg;
474   const Pel* piCur   = pcDtParam->pCur;
475   Int  iRows   = pcDtParam->iRows;
476   Int  iSubShift  = pcDtParam->iSubShift;
477   Int  iSubStep   = ( 1 << iSubShift );
478   Int  iStrideCur = pcDtParam->iStrideCur*iSubStep;
479   Int  iStrideOrg = pcDtParam->iStrideOrg*iSubStep;
480
481   Distortion uiSum = 0;
482
483   for( ; iRows != 0; iRows-=iSubStep )
484   {
485     uiSum += abs( piOrg[0] - piCur[0] );
486     uiSum += abs( piOrg[1] - piCur[1] );
487     uiSum += abs( piOrg[2] - piCur[2] );
488     uiSum += abs( piOrg[3] - piCur[3] );
489     uiSum += abs( piOrg[4] - piCur[4] );
490     uiSum += abs( piOrg[5] - piCur[5] );
491     uiSum += abs( piOrg[6] - piCur[6] );
492     uiSum += abs( piOrg[7] - piCur[7] );
493     uiSum += abs( piOrg[8] - piCur[8] );
494     uiSum += abs( piOrg[9] - piCur[9] );
495     uiSum += abs( piOrg[10] - piCur[10] );
496     uiSum += abs( piOrg[11] - piCur[11] );
497     uiSum += abs( piOrg[12] - piCur[12] );
498     uiSum += abs( piOrg[13] - piCur[13] );
499     uiSum += abs( piOrg[14] - piCur[14] );
500     uiSum += abs( piOrg[15] - piCur[15] );
501
502     piOrg += iStrideOrg;
503     piCur += iStrideCur;
504   }
505
506   uiSum <<= iSubShift;
507   return ( uiSum >> DISTORTION_PRECISION_ADJUSTMENT(pcDtParam->bitDepth-8) );
508 }
509
```

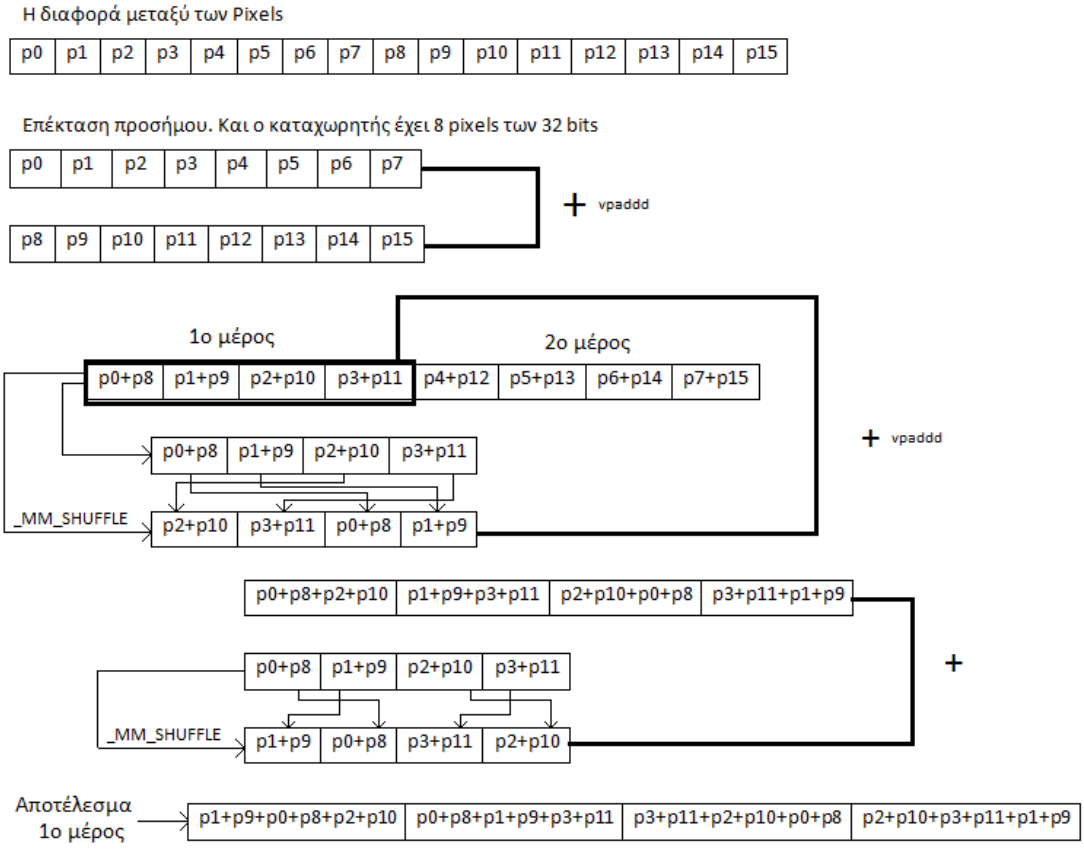
Η νέα συνάρτηση που καλείται είναι η simdSAD16n16b :

```
Distortion TComRdCost::xGetSAD16( DistParam* pcDtParam )
{
  if ( pcDtParam->bApplyWeight )
  {
    return TComRdCostWeightPrediction::xGetSADw( pcDtParam );
  }
  const Pel* piOrg   = pcDtParam->pOrg;
  const Pel* piCur   = pcDtParam->pCur;
  Int  iRows   = pcDtParam->iRows;
  Int  iSubShift  = pcDtParam->iSubShift;
  Int  iSubStep   = ( 1 << iSubShift );
  Int  iStrideCur = pcDtParam->iStrideCur*iSubStep;
  Int  iStrideOrg = pcDtParam->iStrideOrg*iSubStep;

  Distortion uiSum = 0;

  if( pcDtParam->bitDepth <= 10 )
  {
    for( ; iRows != 0; iRows-=iSubStep )
    {
      uiSum += simdSADLine16n16b( piOrg , piCur , 16 );
      piOrg += iStrideOrg;
      piCur += iStrideCur;
    }
  }

  uiSum <<= iSubShift;
  return ( uiSum >> DISTORTION_PRECISION_ADJUSTMENT(pcDtParam->bitDepth-8) );
}
```

Υλοποίηση της συνάρτησης :

```
inline Int simdSADLine16n16b( const Pel * piOrg , const Pel * piCur , Int nWidth )
{
  // internal bit-depth must be 12-bit or lower
  assert( !( nWidth & 0x0F ) );
  __m256i org , cur , abs , sum;
  sum = _mm256_setzero_si256();
  for( Int n = 0 ; n < nWidth ; n += 15 )
  {
    org = _mm256_loadu_si256( ( __m256i* )( piOrg + n ) );
    cur = _mm256_loadu_si256( ( __m256i* )( piCur + n ) );
    abs = _mm256_subs_epi16( _mm256_max_epi16( org , cur )  , _mm256_min_epi16( org , cur ) );
    sum = _mm256_adds_epu16( abs , sum );
  }
  __m256i zero =  _mm256_setzero_si256();
  __m256i hi = _mm256_unpackhi_epi16( sum , zero );
  __m256i lo = _mm256_unpacklo_epi16( sum , zero );
  sum = _mm256_add_epi32( lo , hi );

  __m128i v0h_sum = _mm256_extractf128_si256(sum, 0);
  __m128i v0l_sum = _mm256_extractf128_si256(sum, 1);

  v0h_sum = _mm_add_epi32( v0h_sum , _mm_shuffle_epi32( v0h_sum , _MM_SHUFFLE( 2 , 3 , 0 , 1 ) ) );
  v0h_sum = _mm_add_epi32( v0h_sum , _mm_shuffle_epi32( v0h_sum , _MM_SHUFFLE( 1 , 0 , 3 , 2 ) ) );
  v0l_sum = _mm_add_epi32( v0l_sum , _mm_shuffle_epi32( v0l_sum , _MM_SHUFFLE( 2 , 3 , 0 , 1 ) ) );
  v0l_sum = _mm_add_epi32( v0l_sum , _mm_shuffle_epi32( v0l_sum , _MM_SHUFFLE( 1 , 0 , 3 , 2 ) ) );

  v0h_sum = _mm_add_epi32( v0h_sum , v0l_sum );

  return( _mm_cvtsi128_si32(v0h_sum ) );
}
```

Ακολουθεί ένα διάγραμμα που φαίνεται η διαδικασία των πράξεων. Περιγράφεται το πρώτο μέρος της πρόσθεσης, δηλαδή το p0+p8, p1+p9, p2+p10, p3+p11. Η ίδια διαδικασία εφαρμόζεται και για το δεύτερο μέρος p4+p12, p5+p13, p6+p14, p7+p15. Στο τέλος προστίθενται και οι τιμές αυτών μεταξύ τους και παίρνουμε το τελικό αποτέλεσμα.



Σχήμα 13: Πρόσθεση των εντολών στην xGetSAD16

Εισαγωγή στις xGetSAD32, xGetSAD48 και xGetSAD64

Οι συναρτήσεις xGetSAD32, xGetSAD48 και xGetSAD64 απαιτούν για τον υπολογισμό τους 32, 48 και 64 pixels αντίστοιχα. Παρατηρούμε ότι όλοι οι παραπάνω αριθμοί είναι πολλαπλάσια του 16 και επομένως αν επαναλάβουμε την διαδικασία που περιγράψαμε στη πιο πάνω περίπτωση της xGetSAD16 και xGetSAD16n θα έχουμε το επιθυμητό αποτέλεσμα. Συγκεκριμένα:

(16 pixels x 16 bits το κάθε pixel = 256 bits) x 2 φορές, περίπτωση xGetSAD32

(16 pixels x 16 bits το κάθε pixel = 256 bits) x 3 φορές, περίπτωση xGetSAD48

(16 pixels x 16 bits το κάθε pixel = 256 bits) x 4 φορές, περίπτωση xGetSAD64

Εισαγωγή στις xGetSAD24

Τέλος, η συνάρτηση xGetSAD24 απαιτεί 24 pixel για τον υπολογισμό των διαφορών SAD. Στην συγκεκριμένη περίπτωση παρατηρούμε ότι το 24 είναι πολλαπλάσιο του 8. Επομένως αν χρησιμοποιήσουμε επαναληπτικά την διαδικασία υπολογισμού της xGetSAD8 θα το επιθυμητό αποτέλεσμα. Συγκεκριμένα:

(8 pixels x 16 bits το κάθε pixel = 128 bits) x 2 φορές.

4.3.3 Αλλαγή της συνάρτησης xCalcHADs8x8

Η επόμενη συνάρτηση που μελετήθηκε είναι η xCalcHADs8x8 που υπολογίζει τις διαφορές SAD με μετασχηματισμό Hadamard, SATD. Πρόκειται για ένα πίνακα 8x8 στον οποίο αρχικά πραγματοποιούνται οι διαφορές των pixels που προέρχονται από την κάθε σειρά του πίνακα. Έπειτα γίνονται οριζόντια και κάθετα οι προσθέσεις και οι διαφορές των στοιχείων. Τέλος υπολογίζεται η απόλυτη τιμή και παίρνουμε το τελικό αποτέλεσμα.

Αρχικός κώδικας έχει ως εξής:

```
Distortion TComRdCost::xCalcHADs8x8( const Pel *piOrg, const Pel *piCur, Int iStrideOrg, Int iStrideCur, Int iStep )
{
  Int k, i, j, jj;
  Distortion sad = 0;
  TCoeff diff[64], m1[8][8], m2[8][8], m3[8][8];
  assert( iStep == 1 );
  for( k = 0; k < 64; k += 8 )
  {
    diff[k+0] = piOrg[0] - piCur[0];
    diff[k+1] = piOrg[1] - piCur[1];
    diff[k+2] = piOrg[2] - piCur[2];
    diff[k+3] = piOrg[3] - piCur[3];
    diff[k+4] = piOrg[4] - piCur[4];
    diff[k+5] = piOrg[5] - piCur[5];
    diff[k+6] = piOrg[6] - piCur[6];
    diff[k+7] = piOrg[7] - piCur[7];

    piCur += iStrideCur;
    piOrg += iStrideOrg;
  }

  //horizontal
  for (j=0; j < 8; j++)
  {
    jj = j << 3;
    m2[j][0] = diff[jj  ] + diff[jj+4];
    m2[j][1] = diff[jj+1] + diff[jj+5];
    m2[j][2] = diff[jj+2] + diff[jj+6];
    m2[j][3] = diff[jj+3] + diff[jj+7];
    m2[j][4] = diff[jj  ] - diff[jj+4];
    m2[j][5] = diff[jj+1] - diff[jj+5];
    m2[j][6] = diff[jj+2] - diff[jj+6];
    m2[j][7] = diff[jj+3] - diff[jj+7];

    m1[j][0] = m2[j][0] + m2[j][2];
    m1[j][1] = m2[j][1] + m2[j][3];
    m1[j][2] = m2[j][0] - m2[j][2];
    m1[j][3] = m2[j][1] - m2[j][3];
    m1[j][4] = m2[j][4] + m2[j][6];
    m1[j][5] = m2[j][5] + m2[j][7];
    m1[j][6] = m2[j][4] - m2[j][6];
    m1[j][7] = m2[j][5] - m2[j][7];

    m2[j][0] = m1[j][0] + m1[j][1];
    m2[j][1] = m1[j][0] - m1[j][1];
    m2[j][2] = m1[j][2] + m1[j][3];
    m2[j][3] = m1[j][2] - m1[j][3];
    m2[j][4] = m1[j][4] + m1[j][5];
    m2[j][5] = m1[j][4] - m1[j][5];
    m2[j][6] = m1[j][6] + m1[j][7];
    m2[j][7] = m1[j][6] - m1[j][7];
  }

  for (i=0; i < 8; i++)
  {
    m3[0][i] = m2[0][i] + m2[4][i];
    m3[1][i] = m2[1][i] + m2[5][i];
    m3[2][i] = m2[2][i] + m2[6][i];
    m3[3][i] = m2[3][i] + m2[7][i];
    m3[4][i] = m2[0][i] - m2[4][i];
    m3[5][i] = m2[1][i] - m2[5][i];
    m3[6][i] = m2[2][i] - m2[6][i];
    m3[7][i] = m2[3][i] - m2[7][i];

    m1[0][i] = m3[0][i] + m3[2][i];
    m1[1][i] = m3[1][i] + m3[3][i];
    m1[2][i] = m3[0][i] - m3[2][i];
    m1[3][i] = m3[1][i] - m3[3][i];
    m1[4][i] = m3[4][i] + m3[6][i];
    m1[5][i] = m3[5][i] + m3[7][i];
    m1[6][i] = m3[4][i] - m3[6][i];
    m1[7][i] = m3[5][i] - m3[7][i];

    m2[0][i] = m1[0][i] + m1[1][i];
    m2[1][i] = m1[0][i] - m1[1][i];
    m2[2][i] = m1[2][i] + m1[3][i];
    m2[3][i] = m1[2][i] - m1[3][i];
    m2[4][i] = m1[4][i] + m1[5][i];
    m2[5][i] = m1[4][i] - m1[5][i];
    m2[6][i] = m1[6][i] + m1[7][i];
    m2[7][i] = m1[6][i] - m1[7][i];
  }

  for (i = 0; i < 8; i++)
  {
    for (j = 0; j < 8; j++)
    {
      sad += abs(m2[i][j]);
    }
  }

  sad=((sad+2)>>2);

  return sad;
}
```

Η συνάρτηση τροποποιείται φορτώνοντας στην πρώτη επανάληψη δυο σειρές pixels από τον πίνακα ώστε να υπολογιστούν οι διαφορές. Διαπιστώνουμε ότι:

8 pixels $1^{η}$ σειρά x 8 pixels $2^{η}$ σειρά x 16 bits το κάθε pixel = 256 bits,

τα δεδομένα χωράνε στους καταχωρητές Intel της κατηγορίας AVX και AVX2.

Μέσω των συναρτήσεων που προσφέρονται είναι εφικτό μέσα σε τέσσερεις επαναλήψεις να υπολογιστούν οι διαφορές των στοιχείων. Συναρτήσεις όπως οι παρακάτω απαιτήθηκαν για τον υπολογισμό:

- _mm256_sub_epi16: Διαφορά μεταξύ των pixels κάνοντας ανάγνωση ανά 16 bits μέσα από τον καταχωρητή _m256.
- _mm256_cmpgt_epi16: Συγκρίνει το κάθε pixel με το μηδέν επιστρέφοντας ένα ή μηδέν, δίνοντας δηλαδή θετική ή αρνητική απάντηση. Τα pixels θα αποθηκευτούν σε μεγαλύτερο μέγεθος ίσο με 32 bits, όπου τα 16 νέα bits θα ισούνται με άσσους ή μηδενικά. Πράξη που αποτελεί την επέκταση προσήμου.

Στη συνέχεια του αλγορίθμου παρατηρείται ότι αν υπολογιστεί ο ανάστροφος πίνακας τότε οι οριζόντιες και οι κάθετες πράξεις μπορούν να υπολογιστούν με τον ίδιο τρόπο.

Από το reference code του HM16.15 παρατηρούμε ότι υλοποιείται η Transpose ,η οποία (μετά τον υπολογισμό των διαφορών) βρίσκει τον ανάστροφο του πίνακα 8x8. Στην κατηγορία των SSE εντολών υπάρχει υλοποιημένη η παρακάτω συνάρτηση:

- _MM_TRANSPOSE4_PS η οποία βρίσκει τον ανάστροφο ενός 4x4 πίνακα.

Για το λόγο αυτό ο αρχικός 8x8 πίνακας αναλύεται σε τέσσερεις μικρότερους πίνακες 4x4. Στο σημείο αυτό υπολογίζουμε ότι για τους 4x4 πίνακες θα χρειαστούν:

4 pixels x 32 bits το κάθε pixel (λόγω επέκτασης προσήμου) = 128 bits,

και γι αυτό το λόγο χρησιμοποιήθηκαν οι εντολές και οι καταχωρητές της κατηγορίας SSE και SSE2.

Τέλος τροποποιούμε τον κώδικα χρησιμοποιώντας τις συναρτήσεις της κατηγορίας AVX2 για τον υπολογισμό της απόλυτης τιμής, ώστε να επιταχύνουμε τη διαδικασία. Όπως είναι:

- _mm256_abs_epi32 η οποία υπολογίζει την απόλυτη τιμή των 32 bits

Η simd8x8HAD1D32b και η simdTranspose32b είναι όμοιες με αυτές που δίνονται στο reference code HM16.15 . Τροποποιούνται οι πράξεις της αφαίρεσης μέσα στα for, καθώς και η εύρεση της απόλυτης τιμής για τον υπολογισμό του αποτελέσματος, εφόσον διαβάζονται περισσότερα στοιχεία κάθε φορά μέσα από τους καταχωρητές _m256.

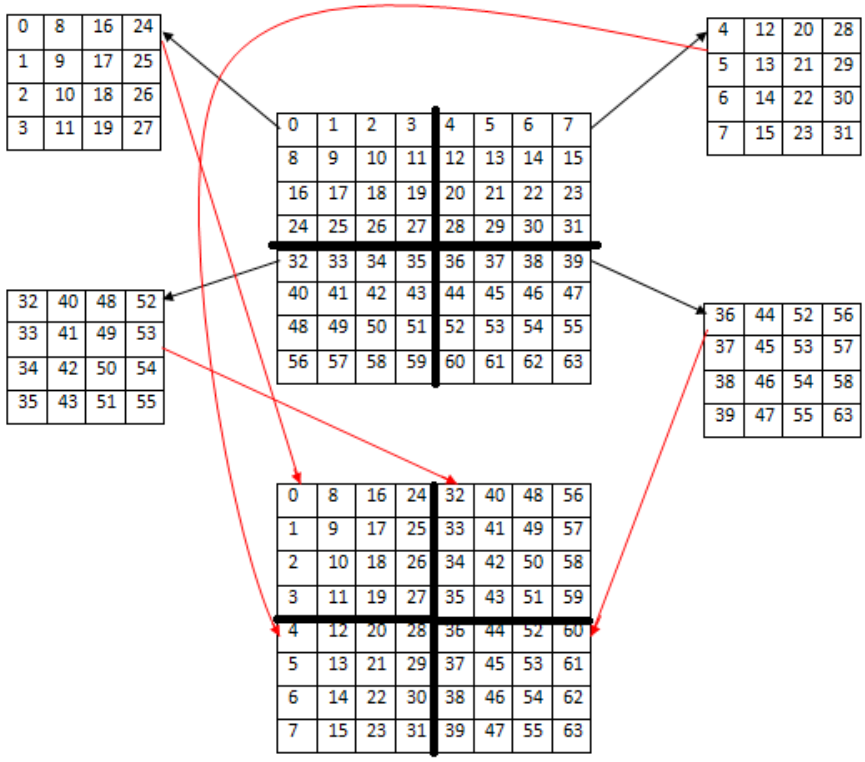
Ο κώδικας έχει ως εξής :

```
UInt simdHADs8x8( const Pel * piOrg, const Pel * piCur, Int iStrideOrg, Int iStrideCur )
{

  __m256i mmDiff[4][2];
  __m256i mmZero = _mm256_setzero_si256();

  __m128i tmp1;
  __m128i tmp2;
  __m128i tmp3;
  __m128i tmp4;


  for( Int n = 0 ; n < 4 ; n++ , piOrg += iStrideOrg , piCur += iStrideCur )
  {
      tmp1 = _mm_loadu_si128( ( __m128i* )piOrg );
      tmp2 = _mm_loadu_si128( ( __m128i* )piCur );

      piOrg += iStrideOrg;
      piCur += iStrideCur;

      tmp3 = _mm_loadu_si128( ( __m128i* )piOrg );
      tmp4 = _mm_loadu_si128( ( __m128i* )piCur );

      __m256i diff1 = _mm256_set_m128i (tmp3, tmp1);
      __m256i diff2 = _mm256_set_m128i (tmp4, tmp2);


      //Subtract packed 16-bit integers. Load 8 pixels

      __m256i diff = _mm256_sub_epi16 (diff1, diff2);

      // sign extension
      __m256i mask = _mm256_cmpgt_epi16 (mmZero , diff); //dst[i+15:i] := ( a[i+15:i] > b[i+15:i] ) ? 0xFFFF : 0

      //Unpack and interleave 16-bit integers from the low half of each 128-bit lane in a and b
      mmDiff[n][0] = _mm256_unpacklo_epi16( diff , mask );
      mmDiff[n][1] = _mm256_unpackhi_epi16( diff , mask );
  }



  // transpose
  simd8x8Transpose32b( &mmDiff[0][0] );

  // horizontal
  simd8x8HAD1D32b( &mmDiff[0][0] , &mmDiff[0][0] );

  // transpose
  simd8x8Transpose32b( &mmDiff[0][0] );

  // vertical
  simd8x8HAD1D32b( &mmDiff[0][0] , &mmDiff[0][0] );

 __m256i mmSum = _mm256_setzero_si256();

  for( Int n = 0 ; n < 4 ; n++ )
  {
    __m256i temp1 = _mm256_abs_epi32 (mmDiff[n][0]);
    __m256i temp2 = _mm256_abs_epi32 (mmDiff[n][1]);
    mmSum = _mm256_add_epi32 ( mmSum , temp1 );
    mmSum = _mm256_add_epi32 ( mmSum , temp2 );

  }

  __m128i v0h_sum = _mm256_extractf128_si256(mmSum, 0);
  __m128i v0l_sum = _mm256_extractf128_si256(mmSum, 1);

  v0h_sum = _mm_add_epi32( v0h_sum , _mm_shuffle_epi32( v0h_sum , _MM_SHUFFLE( 2 , 3 , 0 , 1 ) ) );
  v0h_sum = _mm_add_epi32( v0h_sum , _mm_shuffle_epi32( v0h_sum , _MM_SHUFFLE( 1 , 0 , 3 , 2 ) ) );
  v0l_sum = _mm_add_epi32( v0l_sum , _mm_shuffle_epi32( v0l_sum , _MM_SHUFFLE( 2 , 3 , 0 , 1 ) ) );
  v0l_sum = _mm_add_epi32( v0l_sum , _mm_shuffle_epi32( v0l_sum , _MM_SHUFFLE( 1 , 0 , 3 , 2 ) ) );

  v0h_sum = _mm_add_epi32( v0h_sum , v0l_sum );

  UInt sad = _mm_cvtsi128_si32( v0h_sum );
  sad = ( sad + 2 ) >> 2;

  return( sad );
}
```

Θα αναφέρουμε τον υπολογισμό της transpose, και στην συνέχεια θα την τροποποιήσουμε. Από HM16.15 έχουμε :

```
_MM_TRANSPOSE4_PS( tmp[0] , tmp[2] , tmp[4] , tmp[6] );
_MM_TRANSPOSE4_PS( tmp[1] , tmp[3] , tmp[5] , tmp[7] );
_MM_TRANSPOSE4_PS( tmp[8] , tmp[10] , tmp[12] , tmp[14] );
_MM_TRANSPOSE4_PS( tmp[9] , tmp[11] , tmp[13] , tmp[15] );

for( Int n = 0 ; n < 8 ; n += 2 )
{
  pBuffer[n] = _mm_castps_si128( tmp[n] );
  pBuffer[n+1]  = _mm_castps_si128( tmp[n+8] );
  pBuffer[n+8] = _mm_castps_si128( tmp[n+1] );
  pBuffer[n+9]  = _mm_castps_si128( tmp[n+9] );
}
```

Ο αλγόριθμος που υπολογίζει την transpose, χρησιμοποιώντας την συνάρτηση _MM_TRANSPOSE4_PS που βρίσκει κατευθείαν το ανάστροφο ενός 4x4 πίνακα περιγράφεται από το παρακάτω διάγραμμα



Σχήμα 14: Υπολογισμός της transpose με χρήση της _MM_TRANSPOSE4_PS

4.3.3 Αλλαγή της συνάρτησης Transpose

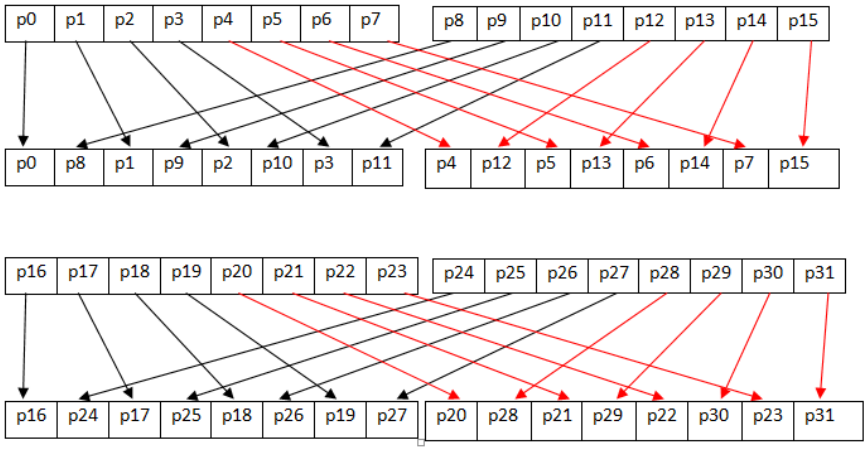
Εισαγωγή των εντολών AVX/AVX2 στη συνάρτηση transpose. Αλλάζουμε τον αλγόριθμο υπολογισμού του ανάστροφου πίνακα με τη δημιουργία μίας συνάρτησης που διαβάζει τον πίνακα ανά σειρά και αντιμεταθέτει τα στοιχεία των σειρών συνεχώς μέχρι να πάρει το επιθυμητό αποτέλεσμα. Ακολουθεί ένα παράδειγμα υπολογισμού της πρώτης σειράς του ανάστροφου πίνακα.

Κώδικας για την transpose

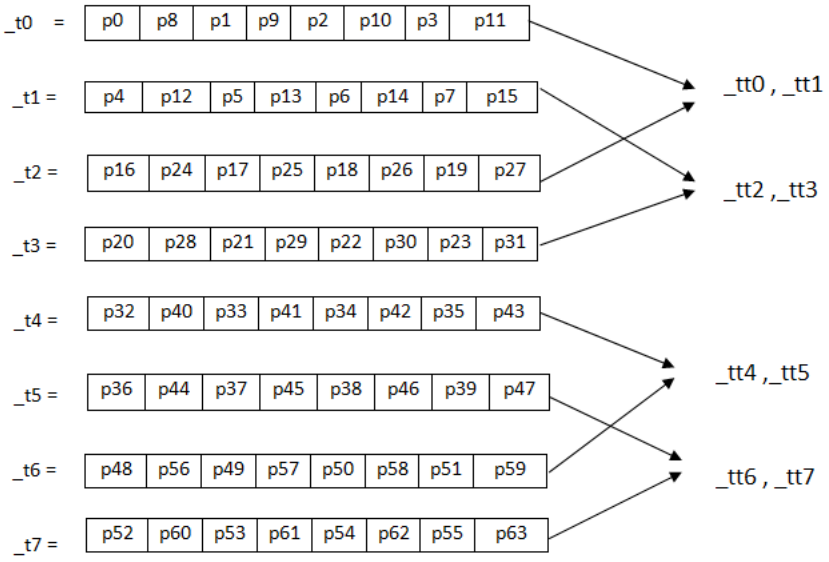
```
inline Void simd8x8Transpose32b(__m256i * pBuffer) {

  __m256 __t0, __t1, __t2, __t3, __t4, __t5, __t6, __t7;
  __m256 __tt0, __tt1, __tt2, __tt3, __tt4, __tt5, __tt6, __tt7;
  __m256 mat[8];


  //type cast to reinterpret four 32-bit integers
  for( Int n = 0 ; n < 8 ; n++ )
  {
    mat[n] = _mm256_castsi256_ps ( pBuffer[n] );
  }

  //Unpack and interleave single-precision (32-bit) floating-point elements from the low half of each 128-bit lane in a and b
  __t0 = _mm256_unpacklo_ps(mat[0], mat[1]);
  __t1 = _mm256_unpackhi_ps(mat[0], mat[1]);
  __t2 = _mm256_unpacklo_ps(mat[2], mat[3]);
  __t3 = _mm256_unpackhi_ps(mat[2], mat[3]);
  __t4 = _mm256_unpacklo_ps(mat[4], mat[5]);
  __t5 = _mm256_unpackhi_ps(mat[4], mat[5]);
  __t6 = _mm256_unpacklo_ps(mat[6], mat[7]);
  __t7 = _mm256_unpackhi_ps(mat[6], mat[7]);

  //Shuffle single-precision (32-bit) floating-point elements
  __tt0 = _mm256_shuffle_ps(__t0,__t2,_MM_SHUFFLE(1,0,1,0));
  __tt1 = _mm256_shuffle_ps(__t0,__t2,_MM_SHUFFLE(3,2,3,2));
  __tt2 = _mm256_shuffle_ps(__t1,__t3,_MM_SHUFFLE(1,0,1,0));
  __tt3 = _mm256_shuffle_ps(__t1,__t3,_MM_SHUFFLE(3,2,3,2));
  __tt4 = _mm256_shuffle_ps(__t4,__t6,_MM_SHUFFLE(1,0,1,0));
  __tt5 = _mm256_shuffle_ps(__t4,__t6,_MM_SHUFFLE(3,2,3,2));
  __tt6 = _mm256_shuffle_ps(__t5,__t7,_MM_SHUFFLE(1,0,1,0));
  __tt7 = _mm256_shuffle_ps(__t5,__t7,_MM_SHUFFLE(3,2,3,2));

  mat[0] = _mm256_permute2f128_ps(__tt0, __tt4, 0x20);
  mat[1] = _mm256_permute2f128_ps(__tt1, __tt5, 0x20);
  mat[2] = _mm256_permute2f128_ps(__tt2, __tt6, 0x20);
  mat[3] = _mm256_permute2f128_ps(__tt3, __tt7, 0x20);
  mat[4] = _mm256_permute2f128_ps(__tt0, __tt4, 0x31);
  mat[5] = _mm256_permute2f128_ps(__tt1, __tt5, 0x31);
  mat[6] = _mm256_permute2f128_ps(__tt2, __tt6, 0x31);
  mat[7] = _mm256_permute2f128_ps(__tt3, __tt7, 0x31);

}
```

Περιγραφή του αλγορίθμου:

Βήμα 1: _mm256_unpacklo_ps, _mm256_unpackhi_ps

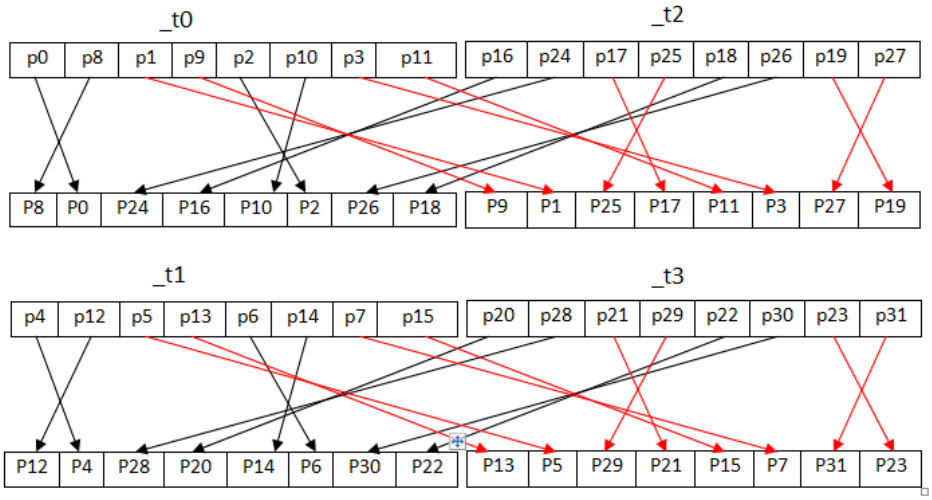


Βήμα 2: Τα νέα ζευγάρια υπολογισμού θα έχουν ως εξής

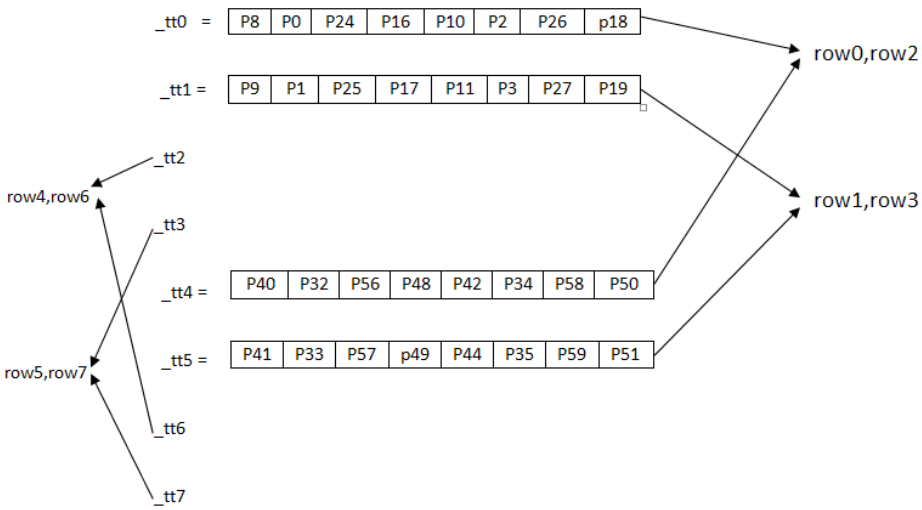


.

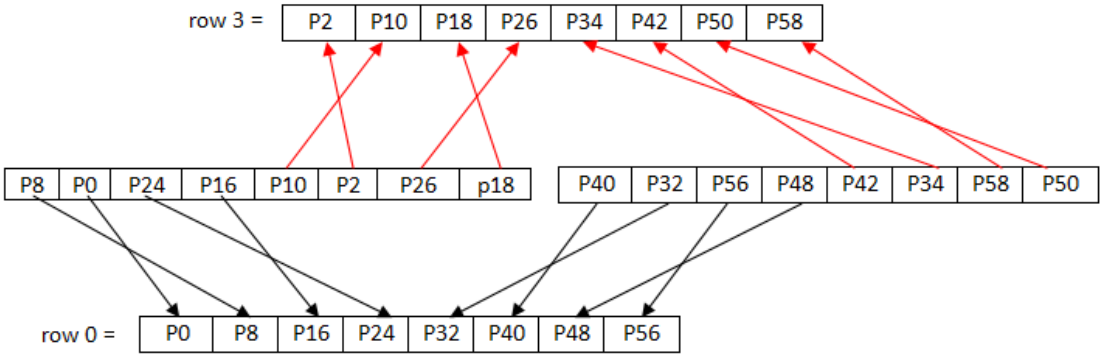
Βήμα 3: _MM_SHUFFLE(1,0,1,0), _MM_SHUFFLE(3,2,3,2)



Βήμα 4: Τα νέα ζευγάρια υπολογισμού



Βήμα 5: Υπολογισμός της 1ης σειράς



4.4 Πειράματα

4.4.1 Εισαγωγή

Χρησιμοποιώντας τον Intel® VTune™ Amplifier 2017 κάνουμε profiling τον κώδικα και δείχνουμε τις πιο χρονοβόρες διαδικασίες του HM reference software. Αρχικά χρησιμοποιώντας την ακολουθία Kimono_1920x1080 με το QP να ισούται με 32 δείχνουμε πως βελτιώνεται διαδοχικά ο κώδικας μέσω των νέων συναρτήσεων που έχουμε εισάγει. Έπειτα επαναλαμβάνουμε τα πειράματα και για την ακολουθία Traffic_2560x1600 για διάφορες τιμές QP, δείχνοντας συνολικά τη διαφορά χρόνου μεταξύ αρχικού και του βελτιστοποιημένου κώδικα.

Όλα τα πειράματα έγιναν σε ηλεκτρονικό υπολογιστή με τα ακόλουθα χαρακτηριστικά:

Operating System	Ubuntu 16.04 LTS
επεξεργαστής	Intel® Core™ i7-4790 CPU @ 3.60GHz × 8
RAM	15,6 GiB
Αρχιτεκτονική	64-bit

Το λογισμικό αναφοράς είναι το HM16.7 και πραγματοποιήθηκαν συνολικά τα ακόλουθα πειράματα :

- Η ακολουθία βίντεο Kimono_1920x1080_24.yuv με QP = 32
- Η ακολουθία βίντεο Kimono_1920x1080_24.yuv με QP = 27
- Η ακολουθία βίντεο Traffic_2560x1600_30_crop.yuv με QP = 32
- Η ακολουθία βίντεο Traffic_2560x1600_30_crop.yuv με QP = 27

- Το configuration file για το Kimono_1920x1080_24.yuv ρυθμίστηκε ως:

FrameRate	24 # Frame Rate per second
FrameSkip	0 # Number of frames to be skipped in input
SourceWidth	1920 # Input frame width
SourceHeight	1080 # Input frame height
FramesToBeEncoded	240 # Number of frames to be coded
Profile	main
QP	32/27* # Quantization parameter(0-51)
Number of Slices	1

*Με την τιμή του QP να είναι στην πρώτη εκτέλεση 32 και στη δεύτερη 27.

- Το configuration file για το Traffic_2560x1600_30_crop.yuv ρυθμίστηκε ως:

FrameRate	30 # Frame Rate per second
FrameSkip	0 # Number of frames to be skipped in input
SourceWidth	2560 # Input frame width
SourceHeight	1600 # Input frame height
FramesToBeEncoded	150 # Number of frames to be coded
Profile	main
QP	32/27 * # Quantization parameter(0-51)
Number of Slices	1

*Με την τιμή του QP να είναι στην πρώτη εκτέλεση 32 και στη δεύτερη 27.

4.4.2 Εκτέλεση κώδικα στην αρχική του μορφή - Kimono_1920x1080 - QP=32

Χρησιμοποιήθηκε η ακολουθία Kimono_1920x1080 με QP=32. Ο συνολικός χρόνος εκτέλεσης του προγράμματος χωρίς χρήση SIMD εντολών είναι 7911.139 δευτερόλεπτα. Και οι συναρτήσεις TcomRdCost:xGetSAD καθώς και TcomRdCost:xCalcHADs8x8 που τροποποιήσαμε είναι από τις πιο χρονοβόρες.

Η συνολική διάρκεια :

Οι συναρτήσεις που πήραν τον περισσότερο χρόνο να εκτελεστούν είναι :

Top Hotspots

This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance.

Function	Module	CPU Time
TComRdCost::xCalcHADs8x8	hevc_parallel	1136.073s
TComTrQuant::xRateDistOptQuant	hevc_parallel	857.504s
TComInterpolationFilter::filter<(int)8, (bool)1, (bool)0, (bool)1>	hevc_parallel	777.970s
TComRdCost::xGetSAD8	hevc_parallel	590.680s
TComInterpolationFilter::filterHor	hevc_parallel	570.495s
[Others]		3958.238s

Παρακάτω δίνουμε μια καλύτερη εικόνα των συναρτήσεων της TcomRdCost και του χρόνου διεκπεραίωσής τους :

Class / Function / Call Stack	CPU Time ▼ (Effective Time by Utilization: Idle, Poor, Ok, Ideal, Over)
TComRdCost	3049.873s
TComRdCost::xCalcHADs8x8	1136.073s
TComRdCost::xGetSAD8	590.680s
TComRdCost::xGetSAD32	246.322s
TComRdCost::xGetSAD16	241.828s
TComRdCost::xGetSAD64	193.620s
TComRdCost::xCalcHADs4x4	139.347s
TComRdCost::xGetSAD4	82.453s
TComRdCost::xGetExpGolombNumberOfBits	77.280s
TComRdCost::xGetSSE16	63.442s
TComRdCost::xGetSSE8	52.151s
TComRdCost::xGetSSE32	49.719s
TComRdCost::xGetHADs	46.225s
TComRdCost::setDistParam	45.633s
TComRdCost::xGetSSE4	28.991s
TComRdCost::getDistPart	16.974s
TComRdCost::xGetSSE64	12.938s
TComRdCost::xGetSAD24	11.878s
TComRdCost::calcRdCost	6.462s
TComRdCost::xGetSAD12	3.748s
TComRdCost::setDistParam	1.730s
TComRdCost::setDistParam	1.504s
TComRdCost::xGetSAD	0.876s
TComInterpolationFilter	1569.465s
TComTrQuant	1048.669s
[Not part of any known object class]	890.359s
TEncSearch	547.220s
TEncSbac	234.436s
TComDataCU	156.506s
TComPrediction	106.665s
TComYuv	79.868s
TEncBinCABACCounter	47.478s
TEncBinCABAC	32.008s
TEncEntropy	28.796s
TComTU	23.089s
TComCUMvField	21.312s
TComTURecurse	17.986s
TEncCu	10.254s
OpParam	8.566s

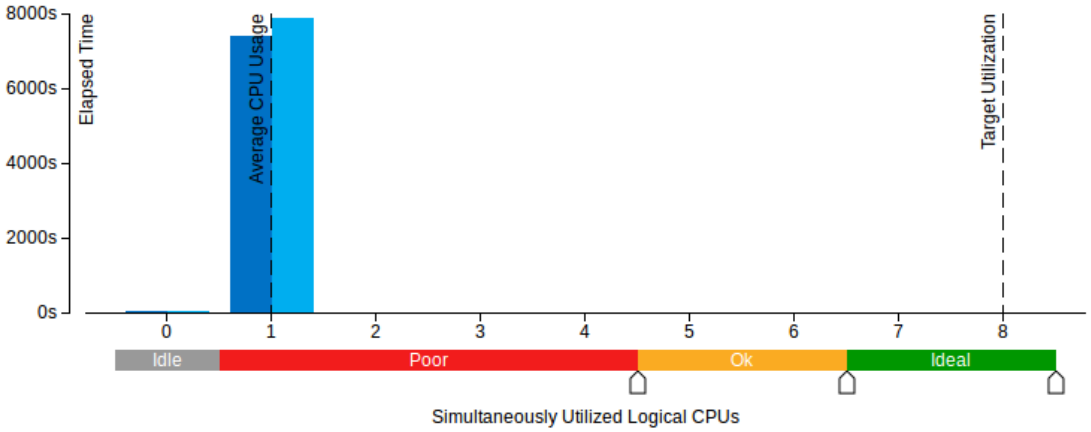
4.4.3 Αλλαγή στις xGetSAD - Kimono_1920x1080 - QP=32

Αφού έχουμε εισάγει τις νέες συναρτήσεις SAD, συγκρίνουμε τα αποτελέσματα με το αρχικό κώδικά και δείχνουμε την βελτίωση σε χρόνο. Χρησιμοποιήθηκε η ακολουθία Kimono_1920x1080 με QP=32.

Ο νέος χρόνος εκτέλεσης του λογισμικού είναι 7403.789 δευτερόλεπτα.

Η διαφορά χρόνου είναι: Αρχικός κώδικας εκτέλεσης - χρόνος εκτέλεσης με την αλλαγή στις xGetSAD :

Διάγραμμα σύγκρισης :



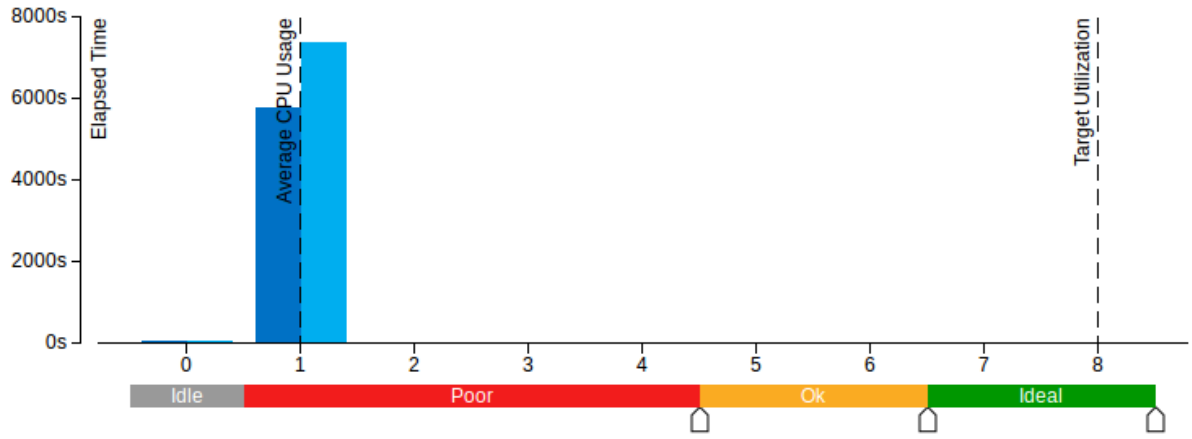
4.4.4 Αλλαγή στην xCalcHADs8x8 και την Transpose - Kimono_1920x1080 - QP=32

Έπειτα βλέπουμε την βελτίωση από τις αλλαγές που κάναμε στην xCalcHADs8x8 συνάρτηση και την εισαγωγή της transpose. Έπειτα συγκρίνουμε το νέο κώδικα σε σύγκριση με το προηγούμενο βήμα, δηλαδή με εκείνο που είχαμε εισάγει τις νέες συναρτήσεις SAD.

Χρόνος εκτέλεσης του κώδικα είναι 5804,784 δευτερόλεπτα :

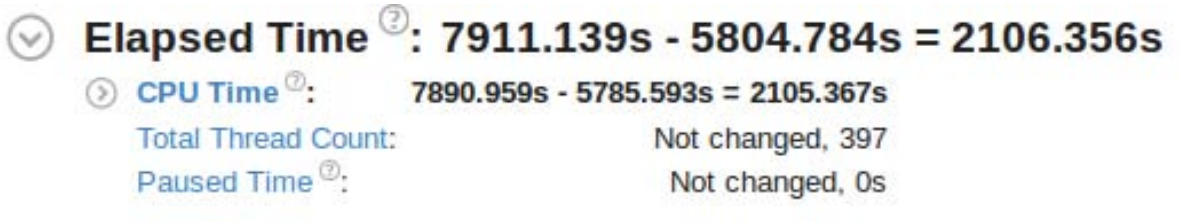
Η διαφορά χρόνου με τον προηγούμενο κώδικα που περιείχε τις νέες xGetSAD συναρτήσεις είναι:

Το διάγραμμα σύγκρισης :

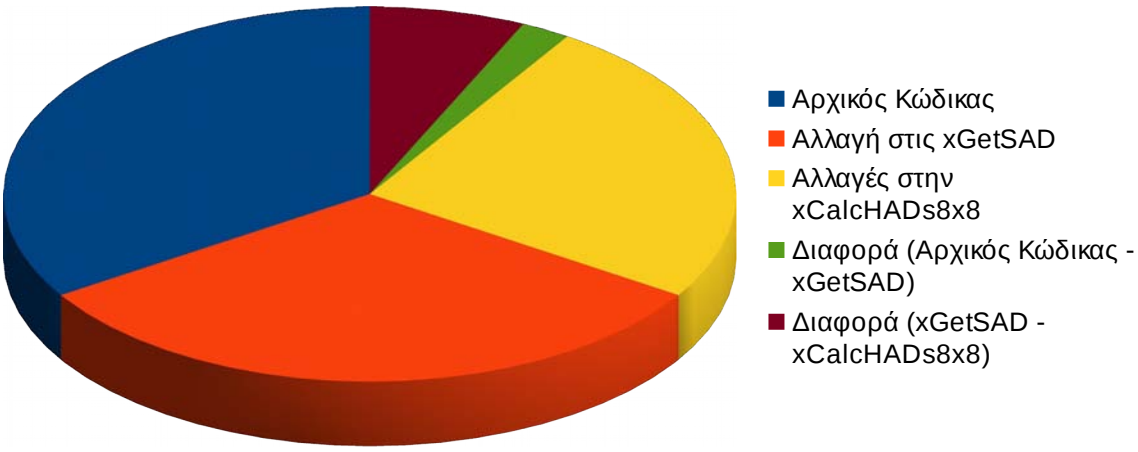


4.4.5 Αποτελέσματα των διαδοχικών εκτελέσεων του βελτιστοποιημένου κώδικα - Kimono_1920x1080 – QP=32

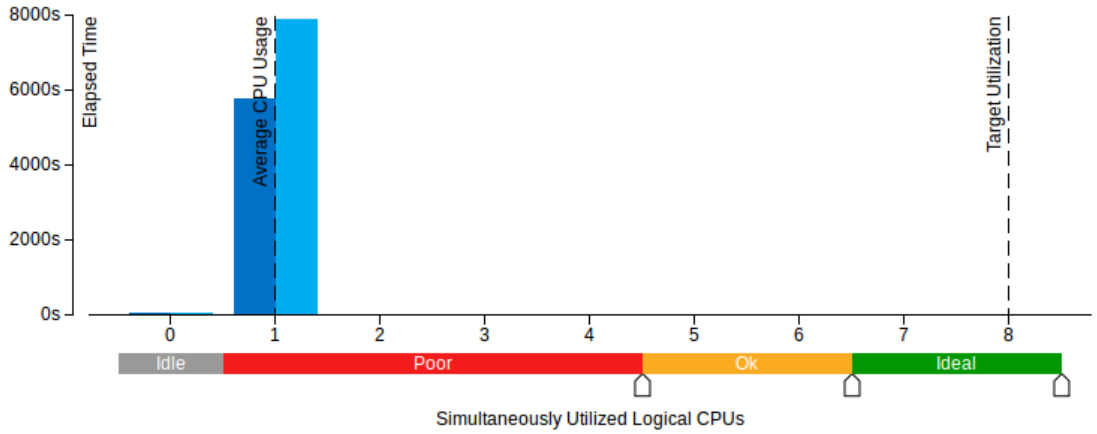
Για την ακολουθία Kimono_1920x1080 με QP=32, έχουμε ότι:
Συνολικός χρόνος είναι: Αρχικός χρόνος εκτέλεσης – Συνολικός βελτιστοποιημένος χρόνος εκτέλεσης :



	Kimono_1920x1080 με QP=32
Αρχικός Κώδικας	7911,139
Αλλαγή στις xGetSAD	7403,789
Αλλαγές στην xCalcHADs8x8	5804,784
Διαφορά (Αρχικός Κώδικας - xGetSAD)	507,35
Διαφορά (xGetSAD - xCalcHADs8x8)	1599,005
Ποσοστό βελτίωσης με εισαγωγή xGetSAD	6,41%
Ποσοστό βελτίωσης με εισαγωγή xCalcHADs8x8	27,55%
Συνολικό ποσοστό βελτίωσης	26,63%



Διάγραμμα σύγκρισης : Αρχικός χρόνος εκτέλεσης – Συνολικός βελτιστοποιημένος χρόνος εκτέλεσης :



Παρουσιάζουμε και την βελτίωση των συναρτήσεων :

Top Hotspots by Difference

This section displays the performance difference between two selected results for the most active functions in your application.

Function	Module	CPU Time, sorted by abs. difference
TComRdCost::xCalcHADs8x8	hevc_parallel	1136.073s - 15.585s = 1120.488s
TComRdCost::xGetSAD8	hevc_parallel	590.680s - 134.167s = 456.513s
TComRdCost::xGetSAD32	hevc_parallel	246.322s - 84.003s = 162.320s
TComRdCost::xGetSAD64	hevc_parallel	193.620s - 35.170s = 158.450s
TComRdCost::xGetSAD16	hevc_parallel	241.828s - 187.441s = 54.387s
[Others]		5482.436s - 5329.227s = 153.209s

4.4.6 Αποτελέσματα - Kimono_1920x1080 – QP=27

Για την ακολουθία Kimono_1920x1080 με QP=27, έχουμε ότι:
Συνολικός χρόνος είναι: Αρχικός χρόνος εκτέλεσης – Συνολικός βελτιστοποιημένος χρόνος εκτέλεσης :

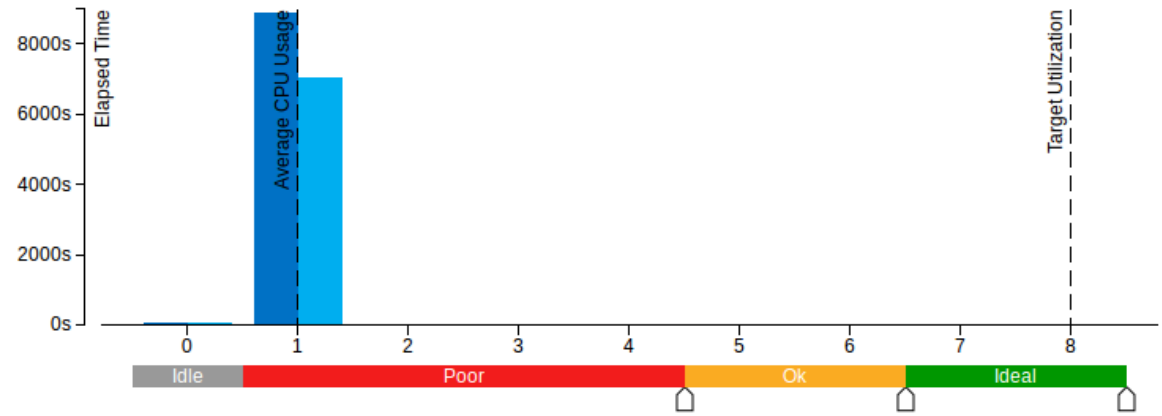
Elapsed Time: 8919.273s - 7055.537s = 1863.736s

CPU Time: 8891.645s - 7036.708s = 1854.936s

Total Thread Count: Not changed, 397

Paused Time: Not changed, 0s

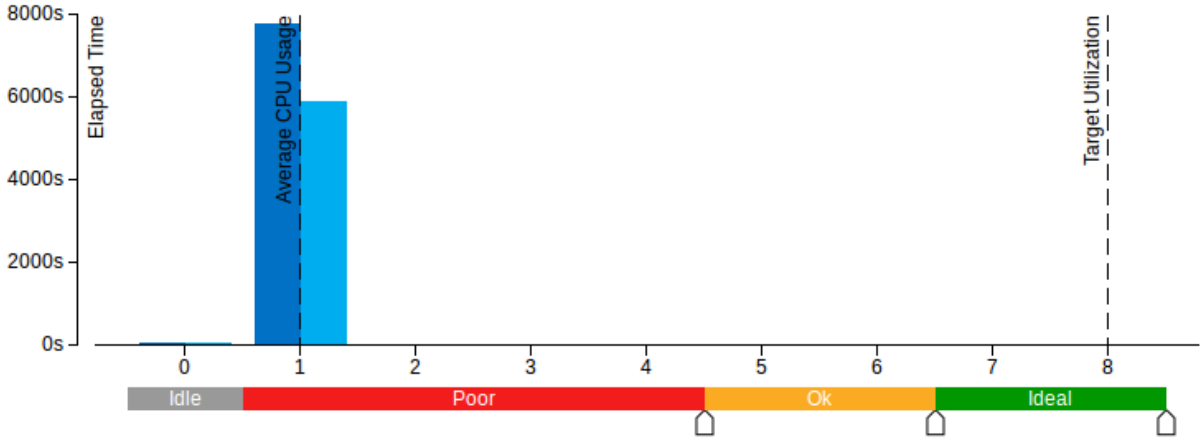
Διάγραμμα σύγκρισης :



4.4.7 Αποτελέσματα - Traffic_2560x1600 – QP=32

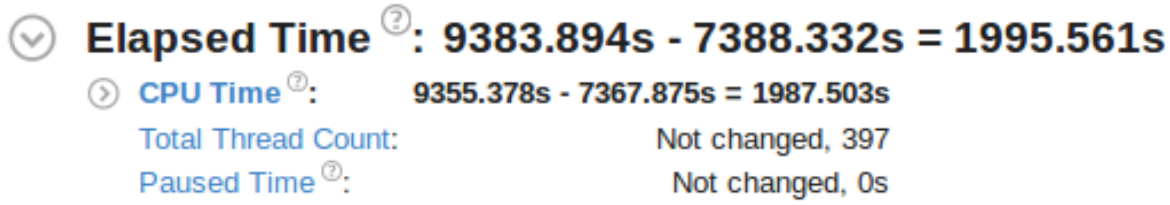
Για την ακολουθία Traffic_2560x1600 με QP=32, έχουμε ότι:
Συνολικός χρόνος είναι: Αρχικός χρόνος εκτέλεσης – Συνολικός βελτιστοποιημένος χρόνος εκτέλεσης :

Διάγραμμα σύγκρισης :

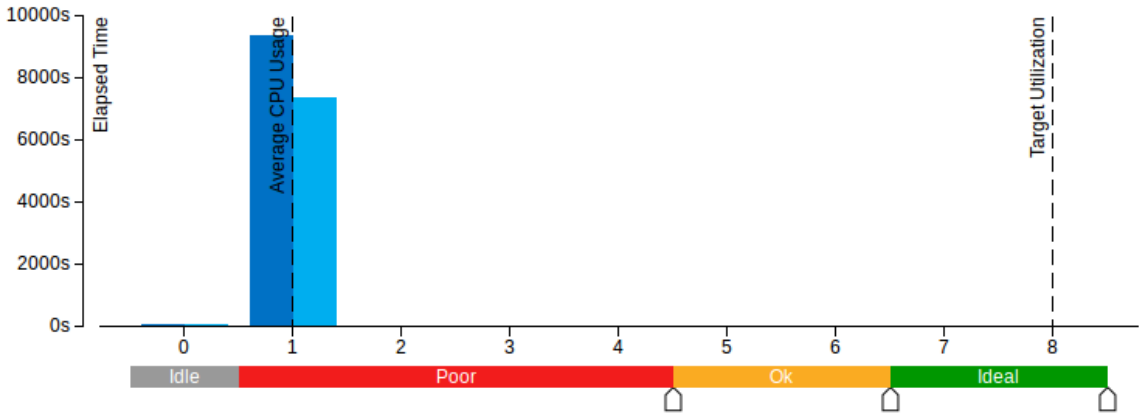


4.4.8 Αποτελέσματα Χρόνου - Traffic_2560x1600 – QP=27

Για την ακολουθία Traffic_2560x1600 με QP=32, έχουμε ότι:
Συνολικός χρόνος είναι: Αρχικός χρόνος εκτέλεσης – Συνολικός βελτιστοποιημένος χρόνος εκτέλεσης :



Διάγραμμα σύγκρισης :



4.4.9 Σύνοψη Αποτελεσμάτων

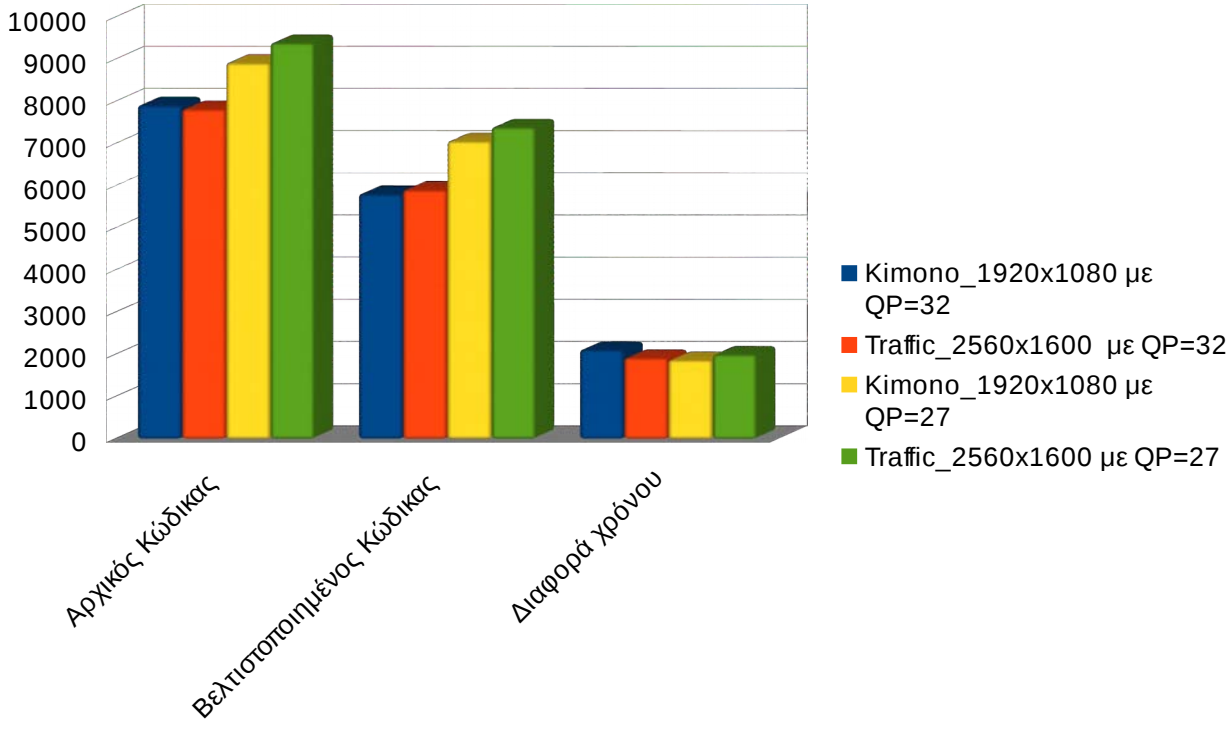
	Kimono_1920x1080 με QP=32
Αρχικός Κώδικας	7911,139
Βελτιστοποιημένος Κώδικας	5804,784
Διαφορά χρόνου	2106,355
Ποσοστό	26,63%

	Traffic_2560x1600 με QP=32
Αρχικός Κώδικας	7818,432
Βελτιστοποιημένος Κώδικας	5907,27
Διαφορά χρόνου	1911,162
Ποσοστό	24,44%

	Kimono_1920x1080 με QP=27
Αρχικός Κώδικας	8919,273
Βελτιστοποιημένος Κώδικας	7055,537
Διαφορά χρόνου	1863,736
Ποσοστό	20,90%

	Traffic_2560x1600 με QP=27
Αρχικός Κώδικας	9383,894
Βελτιστοποιημένος Κώδικας	7388,332
Διαφορά χρόνου	1995,562
Ποσοστό	21,27%

Συγκεντρωτικό διάγραμμα των αποτελεσμάτων των ακολουθιών που χρησιμοποιήθηκαν :



5. Συμπεράσματα

Το HEVC ή αλλιώς H.256 είναι ένας codec βίντεο ,που έχει ως στόχο να αντικαταστήσει το ευρέως διαδεδομένο πρότυπο AVC (H.264). Η καθημερινή ψυχαγωγία εκατομμυρίων ανθρώπων σχετίζεται με δραστηριότητες όπως η παρακολούθηση τηλεοπτικών προγραμμάτων τόσο από την επίγεια όσο και την δορυφορική ψηφιακή τηλεόραση, η παρακολούθηση σειρών και ταινιών μέσω streaming, η εγγραφή βίντεο από κινητά τηλέφωνα και κάμερες, και γενικότερα η αναπαραγωγή οποιασδήποτε ροής βίντεο μέσα από το διαδίκτυο. Για την ανάγκη όλων των παραπάνω περιπτώσεων απαιτείται η μεταφορά, η μετάδοση, η αναπαραγωγή και η εγγραφή βίντεο που επιτυγχάνεται με την συμπίεση του αρχικού σήματος βίντεο. Έτσι δημιουργείται ένα μικρότερο πακέτο πληροφορίας που μπορεί να αποθηκευτεί και να μεταδοθεί ευκολότερα μέσω διαδικτύου ή της τηλεόρασης.

Σε σύγκριση με το AVC, το HEVC προσφέρει σχεδόν τη διπλάσια συμπίεση των δεδομένων βίντεο, διατηρώντας την ίδια ποιότητα ή βελτιώνει σημαντικά την ποιότητα του βίντεο στον ίδιο ρυθμό μετάδοσης (bit rate). Υποστηρίζει αναλύσεις εικόνας έως 8192x4320, συμπεριλαμβάνοντας την 8K UHD.

Ο αλγόριθμος συμπίεσης εκμεταλλεύεται την πλεονάζουσα πληροφορία και αφαιρεί λεπτομέρειες μεταξύ των pixel πετυχαίνοντας έτσι υψηλότερη συμπίεση. Οι δύο βασικοί τύποι που χρησιμοποιούνται για την εκμετάλλευση του πλεονασμού είναι ο χωρικός και ο χρονικός. Ο χωρικός πλεονασμός βρίσκεται σε ομογενείς περιοχές, όπου τα pixel έχουν την ίδια τιμή. Ο χρονικός πλεονασμός εφαρμόζεται ανάμεσα σε δύο διπλανά καρέ, εκεί όπου υπάρχουν περιοχές μέσα στην εικόνας που μένουν στατικές. Χρησιμοποιώντας τις τιμές των pixels από τις ίδιες περιοχές των προηγούμενων καρέ, μπορεί να μειωθεί η μετάδοση της συνολικής πληροφορίας.

Μέσω των λογισμικών αναφοράς που είναι διαθέσιμοι επιτρέπεται η μελέτη και η βελτίωση ενός σύνθετου λογισμικού που συνδυάζει γνώσεις σημάτων και συστημάτων, μελέτη της αρχιτεκτονικής υπολογιστών καθώς και εξοικείωση με τον προγραμματισμό υψηλού επιπέδου.

Παρουσιάστηκε μια λεπτομερής ανάλυση των αλγορίθμων που τροποποιήθηκαν και το αντίκτυπο που είχε η βελτιστοποίηση αυτών με χρήση εντολών SIMD. Η τεχνική βελτιστοποίησης, με παραλληλισμό των δεδομένων πρέπει να είναι μέθοδος επιλογής σε τέτοιας φύσης λογισμικά, που περιέχουν μεγάλο όγκο υπολογιστικών δεδομένων. Έτσι οι αλγόριθμοι επωφελούνται προκαλώντας μείωση του χρόνου εκτέλεσης.

Αναφορές

[1] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, “Overview of the High Efficiency Video Coding (HEVC) Standard,” IEEE Trans. Circuits Syst. Video Technol., vol. 22, no. 12, pp. 1649-1668, Dec. 2012

[2] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, “Overview of the H.264/AVC video coding standard,” IEEE Trans. Circuits Syst.Video Technol., vol. 13, no. 7, pp. 560–576, Jul. 2003.

[3] F. Bossen, B. Bross, K. Sühring, and D. Flynn, “HEVC Complexity and Implementation Analysis,” IEEE Trans. Circuits Syst. Video Techn. vol. 22(12), pp. 1685-1696, 2012.

[4] Jens-Rainer Ohm,Thiow Keng Tan “Comparison of the Coding Efficiency of Video Coding Standards—Including High Efficiency Video Coding (HEVC)”, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, VOL. 22, NO. 12, DECEMBER 2012

[5] HM 16.7 reference software. http://hevc.hhi.fraunhofer.de

[6] Y.-J. Ahn, T.-J. Hwang, D.-G. Sim, and W.-J. Han, “Implementation of fast HEVC encoder based on SIMD and data-level parallelism,” EURASIP J. Image and Video Processing, vol. 16, 2014.

[7] https://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-optimization-manual.pdf

[8] Intel® C++ Intrinsic Reference Document Number: 312482-003US. Intel Corporation

[9] Intel® 64 and IA-32 Architectures Software Developer’s Manual Volume 2 (2A, 2B, 2C & 2D): Instruction Set Reference, A-Z. Intel Corporation, September 2016

[10] Yang Lu (Intel) Real-time End-to-End H.265/HEVC Solution for Intel® Architecture-based Platforms, Ferbruary 2014

[11] Keji Chen, Yizhou Duan, Leju Yan, Jun Sun and Zongming Guo “Efficient SIMD Optimization of HEVC Encoder over X86 Processors”

[12] Intel Corp., Intel® 64 and IA-32 Architectures Software Developers Manual

[13] N14970, High Efficiency Video Coding (HEVC) Test Model 16 (HM16) Improved Encoder Description

[14] Iain E. G. Richardson “H.264 and MPEG-4 VideoCompression”

[15] Vivienne Sze, Madhukar Budagavi, Gary J. Sullivan High Efficiency Video Coding (HEVC) Algorithms and Architectures

[16] Vivienne Sze, Madhukar Budagavi “Design and Implementation of Next Generation Video Coding Systems (H.265/HEVC Tutorial)”, ISCAS Tutorial 2014

[17] Intel Intrinsics Guide