ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ Η/Υ

ΤΟΜΕΑΣ ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Υλοποίηση σε αναδιατασσόμενη λογική του αλγορίθμου συμπίεσης εικόνας WepP

Ακαδημαϊκό έτος 2013-2014

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΤΟΥ ΦΟΙΤΗΤΗ

ΚΡΟΜΜΥΔΑ ΘΕΟΔΩΡΟΥ
ΑΜ: 2005030095

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ:
ΠΑΠΑΕΥΣΤΑΘΙΟΥ ΙΩΑΝΝΗΣ
Αναπληρωτής Καθηγητής Πολυτεχνείου Κρήτης

ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ Η/Υ

ΤΟΜΕΑΣ ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Υλοποίηση σε αναδιατασσόμενη λογική του αλγορίθμου συμπίεσης εικόνας WepP

Ακαδημαϊκό έτος 2013-2014

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΚΡΟΜΜΥΔΑΣ ΘΕΟΔΩΡΟΣ
ΑΜ: 2005030095

Επιτροπή
Αναπληρωτής Καθηγητής Πολυτεχνείου Κρήτης **κ. ΠΑΠΑΕΥΣΤΑΘΙΟΥ ΙΩΑΝΝΗΣ**
Καθηγητής Πολυτεχνείου Κρήτης **κ. ΔΟΛΛΑΣ ΑΠΟΣΤΟΛΟΣ**
Καθηγητής Πολυτεχνείου Κρήτης **κ. ΖΕΡΒΑΚΗΣ ΜΙΧΑΗΛ**

Περιεχόμενα

Εισαγωγή

Στην σύγχρονη εποχή η παραδοσιακή εικόνα έχει αντικατασταθεί από ψηφιακές μεθόδους απεικόνισης, οι οποίες είναι κυρίαρχες στην μετάδοση της πληροφορίας. Στο διαδίκτυο υπάρχουν πάνω από ένα τρισεκατομμύριο ψηφιακές εικόνες. Για την αποθήκευση και μετάδοση ενός τόσο μεγάλου όγκου πληροφοριών απαραίτητη είναι η συμπίεση τους.

Η συμπίεση την ψηφιακής εικόνας αποτελεί έναν τρόπο με τον οποίο μπορούμε να την κωδικοποιήσουμε με σκοπό να επιτύχουμε μείωση στο μέγεθος της αποθήκευσής της και λόγω του μειωμένου μεγέθους να γλιτώσουμε χρόνο αλλά και πόρους κατά την μετάδοσή της. Η συμπίεση είναι επιτυχής μόνο όταν το αποτέλεσμά της είναι οπτικά συγκρίσιμο με την αρχική εικόνα. Μέσω της χρήσης της συμπίεσης μπορούμε να επιτύχουμε μεγάλη μείωση στο μέγεθος της εικόνας που πολλές φορές το τελικό μέγεθος μπορεί να είναι μέχρι και το 20% του μεγέθους της αρχικής εικόνας.

Για να συμπιέσουμε τις εικόνες υπάρχουν δύο τρόποι ανάλογα με την δυνατότητα επιστροφής της συμπιεσμένης εικόνας μέσω αποσυμπίεσης στην αρχική. Αυτές οι μέθοδοι ονομάζονται lossy και lossless. Η lossy μέθοδος είναι μια μέθοδος η οποία έχει απώλειες στο περιεχόμενο κατά την συμπίεση και η αρχική εικόνα δεν μπορεί να ανακατασκευαστεί πλήρως. Αυτές οι απώλειες πληροφορίας είναι ανεπαίσθητες στο οπτικό αποτέλεσμα της εικόνας. Αντιθέτως εικόνες που συμπιέστηκαν με lossless συμπίεση μπορούν να ανακατασκευαστούν πλήρως και να συνθέσουν την αρχική εικόνα χωρίς την παραμικρή διαφορά.

Η lossy συμπίεση χρησιμοποιείται συνήθως σε εικόνες στις οποίες μικρή απώλεια του περιεχομένου δεν δημιουργεί πρόβλημα στην οπτική απεικόνιση. Συνήθως οι lossy μέθοδοι δημιουργούν ένα μικρό θόλωμα στην εικόνα και για αυτό το λόγο είναι καταλληλότερες για φυσικές εικόνες όπως οι φωτογραφίες. Στις περισσότερες lossy μεθόδους η διαφορά δεν γίνεται ορατή με το ανθρώπινο μάτι εάν δεν έχει υποστεί η εικόνα μεγάλης συμπίεση. Αντιθέτως η lossless συμπίεση χρησιμοποιείται κυρίως σε εικόνες με έντονες ακμές όπως τα γραφικά κειμένων και τα τεχνικά σχέδια.

Οι πιο συχνές μέθοδοι lossless συμπίεσης σε εικόνες είναι οι Run-length κωδικοποίηση(RLE),η κωδικοποίηση εντροπίας και οι κωδικοποιητές λεξικού. Η RLE απλά απεικονίζει μεγάλες ακολουθίες των ίδιων δεδομένων σε συντομότερη μορφή. Για παράδειγμα μια ακολουθία της μορφής AAAAAAAAABBBBBBBBBΓΓΓΓΔΔΔ μπορεί να απεικονιστεί ως 9A9B4Γ3Δ. Η κωδικοποίηση εντροπίας αντιστοιχίζει κωδικούς σε σύμβολα έτσι ώστε τα σύμβολα που εμφανίζονται πιο συχνά να έχουν τους μικρότερους κωδικούς. Η ποιο συνήθης τεχνική κωδικοποίησης εντροπίας είναι η κωδικοποίηση Huffman. Η κωδικοποίηση λεξικού φτιάχνει ένα πίνακα με συμβολοσειρές και έπειτα αντικαθιστά τις εμφανίσεις τους με τους συντομότερους κωδικούς. Ο αλγόριθμος LZW (Lempel-Ziv-Welch) είναι πιθανότερα ο πιο συνήθης κωδικοποιητής λεξικού και χρησιμοποιείται σε τύπους συμπίεσης όπως το ZIP και το GIF.

Η lossy συμπίεση συνήθως βασίζετε σε τεχνικές οι οποίες αφαιρούν λεπτομέρειες οι οποίες δεν είναι εμφανείς στο ανθρώπινο μάτι. Οι ψηφιακές εικόνες αποτελούνται από pixel τα οποία προσδιορίζουν πληροφορίες χρώματος. Όταν ένα pixel έχει μικρή διαφορά από τα γειτονικά του μπορεί εύκολα χωρίς να επηρεαστεί το αποτέλεσμα της εικόνας να αντικατασταθεί η τιμή του από κάποια τιμή από τα γειτονικά pixel. Αυτό θα έχει ως αποτέλεσμα να χαθούν κάποιες πληροφορίες από την εικόνα οι οποίες όμως δεν είναι παρατηρήσιμες από το ανθρώπινο μάτι , εάν ο αλγόριθμος είναι καλός. Κατόπιν μπορεί να χρησιμοποιηθεί ο RLE ή η κωδικοποίηση Huffman για την συμπίεση των δεδομένων. Η πιο χρησιμοποιούμενη μέθοδος για συμπίεση τύπου lossy είναι η κωδικοποίηση μετασχηματισμού όπως ο μετασχηματισμός discrete cosine (DCT, χρησιμοποιούμενος στο JPEG) ή ο μετασχηματισμός κύματος(wavelet transform, χρησιμοποιείται στο JPEG2000).

Άλλες δημοφιλείς μέθοδοι είναι ο κβαντισμός των χρωμάτων (μειώνει την παλέτα των χρωμάτων) και η υποδειγματοληψία του χρωματισμού. Αυτές οι μέθοδοι βασίζονται στο γεγονός ότι το ανθρώπινο μάτι είναι πιο ευαίσθητο στην φωτεινότητα παρά στο χρώμα , επομένως το μέγεθος του αρχείου μπορεί να βελτιστοποιηθεί αποθηκεύοντας πιο πολλές πληροφορίες για την φωτεινότητα από το χρώμα. Ακόμα χρησιμοποιείται η κλασματική συμπίεση αλλά δεν είναι τόσο συχνή.

Υπάρχουν πολλοί τύποι αρχείων εικόνας οι οποίοι χρησιμοποιούν αυτές τις μεθόδους συμπίεσης διαφορετικά. Μερικοί χρησιμοποιούν μόνο lossless άλλοι χρησιμοποιούν και lossy και lossless και κάποιοι καθόλου συμπίεση. Τα πιο συνηθισμένα πρότυπα εικόνας είναι : BMP, PNG, JPEG,GIF και TIFF.

Το BMP είναι ένα πρότυπο εικόνας χωρίς συμπίεση, το οποίο χρησιμοποιείται στα Windows και οι εικόνες του καταλαμβάνουν μεγάλο χώρο. Το JPEG είναι ένα πρότυπο το οποίο χρησιμοποιεί lossy μεθόδους συμπίεσης όπως ο DCT και η υποδειγματοληψία του χρωματισμού. Επιπλέον χρησιμοποιεί lossless μεθόδους όπως ο RLE και η κωδικοποίηση Huffman αλλά δεν επιτρέπει την αποθήκευση σε lossless μορφή. Το JPEG είναι ιδιαίτερα καλό με φυσικές εικόνες. Το GIF χρησιμοποιεί τον lossless αλγόριθμο συμπίεσης LZW. Οπότε μειώνει το μέγεθος αρχείου χωρίς να υποβαθμίζει την ποιότητα της εικόνας. Όμως το GIF χρησιμοποιεί μόνο 256 χρώματα , γεγονός το οποίο δεν το κάνει κατάλληλο για φωτογραφίες οι οποίες αποτελούνται από εκατομμύρια χρώματα. Παρόλα αυτά , η ποιότητα μπορεί να ενισχυθεί χρησιμοποιώντας τη χρωματική πρόσμιξη. Το GIF είναι ιδιαιτέρως καλό για τεχνητές εικόνες οι οποίες περιέχουν έντονες ακμές και λίγα χρώματα. Ο αλγόριθμος συμπίεσης ο οποίος χρησιμοποιεί το GIF , έγινε πατέντα το 1985 και η αντιπαράθεση για αυτήν την πατέντα οδήγησε στην ανάπτυξη του τύπου αρχείου PNG. Το PNG χρησιμοποιεί και αυτό lossless συμπίεση η οποία ονομάζεται DEFLATE η οποία είναι συνδυασμός του κωδικοποιητή λεξικού LZ77 και της κωδικοποίησης Huffman. Το PNG προσφέρει καλύτερη συμπίεση και περισσότερες δυνατότητες από το GIF αλλά δεν υποστηρίζει κινούμενες εικόνες, πράγμα το οποίο υποστηρίζει το πρότυπο GIF. Μια ακόμα σημαντική διαφορά είναι ότι το PNG υποστηρίζει εικόνες με εκατομμύρια χρώματα.

Το πρότυπο TIFF είναι ένας ευέλικτος και προσαρμοστικός τύπος αρχείου. Μπορεί να αποθηκεύσει πολλές εικόνες σε ένα μόνο αρχείο και μπορείς να επιλέξεις ποιόν αλγόριθμο συμπίεσης θα χρησιμοποιήσεις. Το TIFF μπορεί να χρησιμοποιηθεί σαν ένα δοχείο για JPEG ή μπορείς να επιλέξεις μια lossless συμπίεση όπως η RLE και η LZW. Επειδή το TIFF υποστηρίζει πολλές εικόνες σε ένα μόνο αρχείο , έγγραφα πολλαπλών σελίδων μπορούν να αποθηκευτούν σαν ένα μοναδικό αρχείο TIFF αντί για μια σειρά αρχείων για κάθε σκαναρισμένη σελίδα.

Υπάρχει και μια νέα έκδοση του πρότυπου JPEG το οποίο ονομάζεται JPEG2000 το οποίο υποστηρίζει και lossless συμπίεση. Οι lossy αλγόριθμοι wavelet παράγουν ελαφρώς καλύτερο αποτέλεσμα από το JPEG(μέχρι 20% και παραπάνω) αλλά δεν χρησιμοποιείται ευρέως λόγω κάποιων ζητημάτων με την πατέντα. Η Microsoft ανέπτυξε ένα “νέας γενιάς” πρότυπο το οποίο ονομάζεται HD Photo (παλιότερα Windows Media Photo). Η Microsoft υποστηρίζει ότι το HD Photo προσφέρει ποιότητα εικόνας η οποία είναι αντιληπτά συγκρίσιμη με αυτή του JPEG2000 και παράγει εικόνες με διπλάσια ποιότητα από το JPEG.

Ακόμα υπάρχουν και τα γραφικά SVG τα οποία χρησιμοποιούνται για να παράγουν διανυσματικά γραφικά. Στην πραγματικότητα δεν είναι ένας αλγόριθμος συμπίεσης αλλά μία γλώσσα σήμανσης βασισμένη στο XML. Αντί να αντιστοιχεί το χρώμα σε κάθε pixel όπως στα γραφικά ψηφιδωτού (JPEG,GIF,PNG), το SVG χρησιμοποιεί μαθηματικές εκφράσεις για να προσδιορίσει συντεταγμένες σχημάτων. Οι εικόνες SVG μπορούν να είναι υπερβολικά μικρές. Η εικόνα είναι ακόμα κλιμακούμενη επειδή είναι βασισμένη σε διανύσματα , οπότε μπορείς να μεγεθύνεις την εικόνα και να έχει ακόμα πολύ καλή ποιότητα.

Τελευταίο αλλά αυτό που θα μας απασχολήσει κατά κύριο λόγο στην διπλωματική

αυτή εργασία είναι το πρότυπο συμπίεσης εικόνας WebP , το οποίο αναπτύσσεται από την Google από το 2010 , το οποίο υποστηρίζει τόσο lossy όσο και lossless συμπίεση. Το πρότυπο αυτό είναι ένα open-source πρότυπο , δηλαδή ο καθένας μπορεί να προτείνει αλλαγές, και είναι ταχέως αναπτυσσόμενο λόγω των δυνατοτήτων του, αλλά και της μεγάλης ποικιλίας που προσφέρει σε επιλογές συμπίεσης, υποστήριξης κινούμενης εικόνας αλλά και alpha (transparent) frame, αλλά και των συνδυασμών αυτών.

Κεφάλαιο 1. Η Ψηφιακή εικόνα

1.1 Εισαγωγή

Με τον όρο ψηφιακή εικόνα αναφερόμαστε στην δισδιάστατη απεικόνιση ενός πλήθος στοιχείων, τα οποία όταν συντεθούν και απεικονιστούν σε ένα μέσο προβολής μέσω της κατάλληλης επεξεργασίας, απεικονίζουν μία εικόνα. Κάθε ψηφιακή εικόνα μπορεί να αποτελείται από διανυσματικά γραφικά ή από τον συνδυασμό των τιμών που χρειάζεται για να απεικονιστεί κάθε εικονοστοιχείο.

Οι «ψηφιδωτές» εικόνες (raster images) έχουν ένα πεπερασμένο αριθμό ψηφιακών τιμών τα οποία ονομάζονται εικονοστοιχεία (picture elements , pixels). Η κάθε ψηφιακή εικόνα περιέχει ένα συγκεκριμένο αριθμό γραμμών και στηλών από pixel. Το pixel είναι το μικρότερο ατομικό στοιχείο σε μια εικόνα και περιέχει πληροφορίες για την φωτεινότητα ή το χρώμα σε ένα συγκεκριμένο κομμάτι της τελικής απεικόνισης. Όλα τα pixel μιας εικόνας μαζί δημιουργούν ουσιαστικά έναν δισδιάστατο πίνακα τιμών , ο οποίος «χαρτογραφεί» την εικόνα με βάση διάφορα χαρακτηριστικά. Οι raster εικόνες είναι ο πιο συνηθισμένος τρόπος με τον οποίο μπορεί να γίνει μια ψηφιακή απεικόνιση καθότι είναι ο πιο διαδεδομένος σε χρήση. Raster εικόνες αποθηκεύονται από ψηφιακές φωτογραφικές μηχανές, από κινητά τηλέφωνα, από web cameras καθώς και πληθώρα άλλων μέσων.

Αντιθέτως οι διανυσματικές εικόνες δεν είναι τόσο δημοφιλείς καθώς χρησιμοποιούνται για την απεικόνιση τεχνικών σχεδίων κατά βάση , ή εικόνων που βασίζονται στην γεωμετρία και στο κείμενο.

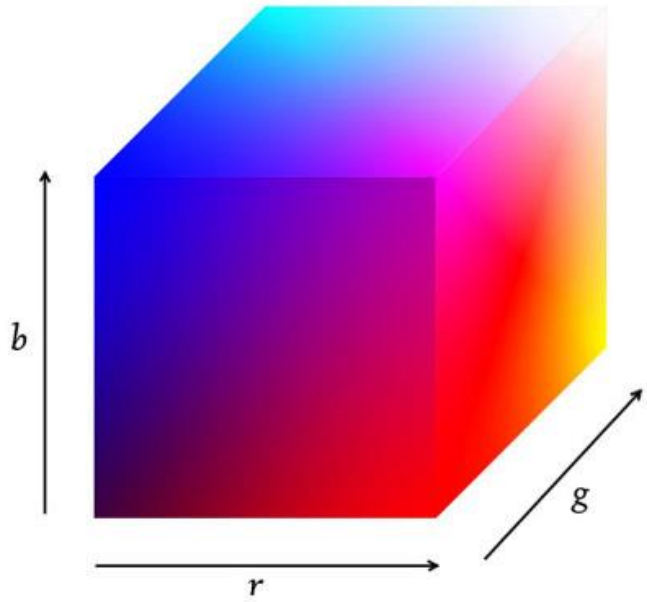
1.2 Ιστορικά

Η πρώτη ψηφιακή εικόνα, με τον τρόπο που την γνωρίζουμε σήμερα ,η οποία σαρώθηκε, αποθηκεύτηκε και μεταδόθηκε με ψηφιακά εικονοστοιχεία παρουσιάστηκε στον SEAC (Standards Eastern Automatic Computer) στο Εθνικό Ινστιτούτο Προτύπων και Τεχνολογίας (National Institute of Standards and Technology , NIST) στο Μέρυλαντ των Ηνωμένων Πολιτειών στα τέλη της δεκαετίας του 1950.

Η ψηφιακή εικόνα όμως χρησιμοποιούταν ήδη από την δεκαετία του 1920 και την υποβρύχια γραμμή τηλεομοιότυπου(fax) η οποία συνέδεε το Λονδίνο με την Νέα Υόρκη, που ως σκοπό είχε την μετάδοση ψηφιοποιημένων εικόνων εφημερίδας. Η πρώτη κωδικοποίηση έγινε σε 5 κλίμακες του γκρι οι οποίες σύντομα αυξήθηκαν σε 15.

Κατά την δεκαετία του 1960 με την τρομακτική έκρηξη της προόδου της τεχνολογίας λόγω των διαστημικών προγραμμάτων αλλά και της ιατρικής έρευνας , η ψηφιακή εικόνα εξελίχθηκε έτσι ώστε να ικανοποιήσει τις ανάγκες των δορυφορικών απεικονίσεων, της ιατρικής απεικόνισης , της τεχνολογίας του βιντεόφωνου καθώς και της αναγνώρισης χαρακτήρων. Πρωτεργάτες ήταν το MIT , το πανεπιστήμιο του Μέρυλαντ, τα Bell Labs αλλά και η NASA.

Μεγαλύτερη εξέλιξη ακόμα παρατηρήθηκε την επόμενη δεκαετία με την εισαγωγή των μικροεπεξεργαστών αλλά και την ανάπτυξη καλύτερων μεθόδων στην αποθήκευση αλλά και στην απεικόνιση της πληροφορίας. Μια από τις μεγαλύτερες και σημαντικότερες χρήσεις αναπτύχθηκε για ιατρικούς σκοπούς την περίοδο εκείνην και είναι η αξονική τομογραφία, δηλαδή η ψηφιακή απεικόνιση μέσω ακτινών x μιας τομής ενός τρισδιάστατου αντικειμένου. Τότε ξεκίνησε και η ψηφιοποίηση παραδοσιακών αναλογικών εικόνων γεγονός το οποίο βοήθησε σε



Εικόνα 1 Ο χρωματικός χώρος του χρωματικού μοντέλου RGB (http://www.codeproject.com/Articles/9207/An-HSV-RGBA-colour-picker)

πολλούς τομείς της επιστήμης όπως η πυρηνική ιατρική, η αστρονομία, η αρχαιολογία η άμυνα αλλά άρχισε και η χρήση των ψηφιακών εικόνων και σε μεγάλες βιομηχανικές μονάδες.

Στα χρόνια που ακολούθησαν η ραγδαία ανάπτυξη των υπολογιστών, αλλά και των μονάδων απεικόνισης και σύλληψης εικόνων (τηλεοράσεις , κάμερες , φωτογραφικές μηχανές κ.α.) βοήθησε στην έλευση της εποχής της πληροφορίας στην οποία η εικόνα , και κατά κύριο λόγο η ψηφιακή, παίζει κυρίαρχο ρόλο.

1.3 Η ψηφιακή απεικόνιση των χρωμάτων

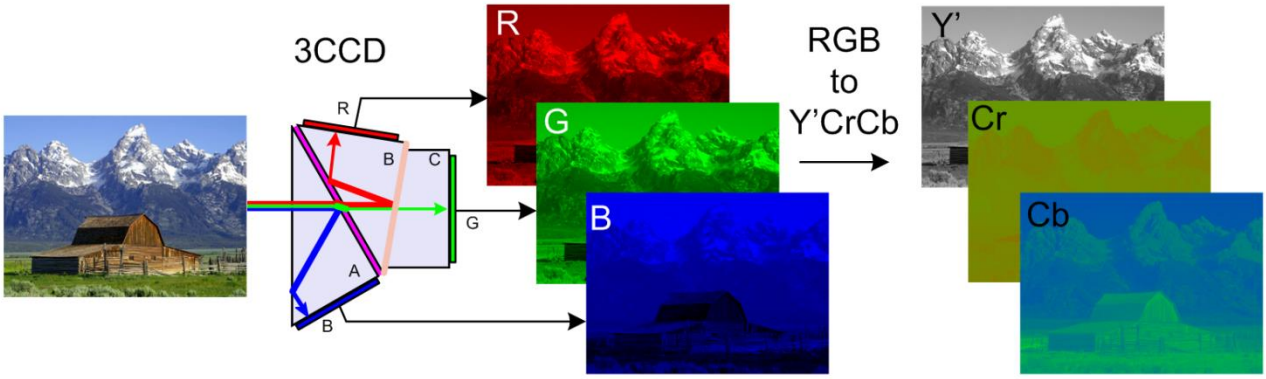
Η απεικόνιση των χρωμάτων των εικονοστοιχείων σε μια ψηφιακή εικόνα γίνεται μέσω της αντιστοίχησής τους σε τιμές μέσω ενός χρωματικού χάρτη (color map) ο οποίος είναι μέρος ενός χρωματικού μοντέλου. Κάθε μαθηματικό μοντέλο είναι ένα αφηρημένο μαθηματικό μοντέλο το οποίο περιγράφει τον τρόπο με τον οποίο μπορούν τα χρώματα να αναπαρασταθούν μέσω μιας πλειάδας αριθμών, συνήθως τριών ή τεσσάρων ή μέσω συστατικών χρωμάτων. Παρόλα αυτά ένα χρωματικό μοντέλο χωρίς μια συνάρτηση αντιστοίχησης σε έναν χρωματικό χώρο (color space) δεν είναι τίποτα άλλο παρά ένα αυθαίρετο σύστημα χρωμάτων χωρίς κάποι0ο ουσιαστικό αντίκρισμα.

Προσθέτοντας μια συγκεκριμένη συνάρτηση αντιστοίχισης μεταξύ του χρωματικού μοντέλου και ενός συγκεκριμένου χρωματικού χώρου, οδηγεί σε ένα μοναδικό «αποτύπωμα» μέσα στον χρωματικό χώρο. Αυτό το αποτύπωμα ονομάζεται και φάσμα και προσδιορίζει έναν καινούριο χρωματικό χώρο.

Τα χρωματικά μοντέλα είναι το RGB, το CMY, το YIQ, το YCbCr, το CIELAB, και τα HSI, HSV και HLS. Το μοντέλο RGB χρησιμοποιείται κυρίως για την απεικόνιση σε έγχρωμες οθόνες και κάμερες, το CMY χρησιμοποιείται σε έγχρωμους εκτυπωτές ενώ το YIQ είναι το πρότυπο για την τηλεοπτική μετάδοση. Τα υπόλοιπα μοντέλα χρησιμοποιούνται κυρίως σε εφαρμογές όπου είναι επιθυμητή η διαχείριση των χρωμάτων.

Για την απεικόνιση κάθε εικόνας σε μία οθόνη είναι απαραίτητη η μετατροπή του

από οποιοδήποτε μοντέλο έχει επιλεχθεί για την αποθήκευσή του στο μοντέλο RGB.



Εικόνα 2 Η μετατροπή από την σύλληψη της εικόνας μέχρι το μοντέλο YCbCr (http://en.wikipedia.org/wiki/YCbCr)

Σε αυτό το μοντέλο κάθε χρώμα απεικονίζεται σαν συνιστώσα τριών χρωμάτων, του κόκκινου(R), του πράσινου(G) και του μπλε(B). Στο μοντέλο CMY τα τρία χρώματα που συνδυάζονται για να παραχθεί όλος ο χρωματικός χώρος του είναι το κυανό(C), το φουξ(magenta, M), και το κίτρινο(Y). Τα μοντέλα RGB και CMY θεωρούνται συμπληρωματικά καθώς για την μετατροπή από το ένα στο άλλο χρησιμοποιείται ο τύπος:

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \qquad (1.1)$$

Για τις μετατροπές των υπόλοιπων μοντέλων η διαδικασία της μετατροπής είναι πιο πολύπλοκη. Το μοντέλο που μας απασχολεί κυρίως σε αυτή την εργασία είναι το YCbCr το οποίο μετατρέπεται σε RGB χρησιμοποιώντας τον τύπο:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.13983 \\ 1 & -0.39465 & -0.58060 \\ 1 & 2.03211 & 0 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \end{bmatrix} \qquad (1.2)$$

Στο μοντέλο YCbCr όπως και στα υπόλοιπα μοντέλα κάθε pixel αποτελείται από μια τριάδα τιμών. Οι τιμές όμως στο μοντέλο YCbCr δεν αντιστοιχούν σε χρώματα, αλλά οι τρείς μεταβλητές Y , Cb και Cr αντιστοιχούν το Y στην φωτεινότητα(luma) , και τα Cb και Cr στην χρωματικότητα(Chroma) του εικονοστοιχείου. Το μοντέλο αναπτύχθηκε κατά αυτό τον τρόπο για την ευκολότερη μετάβαση από την ασπρόμαυρη τηλεόραση και εικόνα στην έγχρωμη, καθώς στην ήδη υπάρχουσα μεταβλητή της φωτεινότητας προστέθηκαν μόνο 2 μεταβλητές για το χρώμα και δεν χρειάστηκε μεγάλες αλλαγές. Υπάρχουν τρείς κύριες εκδοχές του μοντέλου αυτού, η YUV 4:4:4 , 4:2:2 , 4:2:0 (ή 4:1:1) όπου το κάθε στοιχείο δειγματοληπτείται με βάση αυτή την αναλογία.

Στον πίνακα φαίνονται για κάθε pixel το πώς είναι δειγματοληπτημένο ανάλογα με το

κάθε μοντέλο που χρησιμοποιείται:

Pixel number	Χρωματικό μοντέλο			
	RGB	YCbCr 4:4:4	YCbCr 4:2:2	YCbCr 4:1:1
0	R0 G0 B0	Y0 Cr0 Cb0	Y0 Cr0 Cb 0	Y0 Cr0 Cb0
1	R1 G1 B1	Y1 Cr1 Cb1	Y1 Cr0 Cb 0	Y1 Cr0 Cb0
2	R2 G2 B2	Y2 Cr2 Cb2	Y2 Cr2 Cb 2	Y2 Cr0 Cb0
3	R3 G3 B3	Y3 Cr3 Cb3	Y3 Cr2 Cb 2	Y3 Cr0 Cb0
4	R4 G4 B4	Y4 Cr4 Cb4	Y4 Cr4 Cb 4	Y4 Cr1 Cb1
5	R5 G5 B5	Y5 Cr5 Cb5	Y5 Cr4 Cb 4	Y5 Cr1 Cb1
6	R6 G6 B6	Y6 Cr6 Cb6	Y6 Cr6 Cb 6	Y6 Cr1 Cb1

Πίνακας 1. Τιμές εικονοστοιχείων ανάλογα με το μοντέλο απεικόνισης

Επιπρόσθετα μπορούμε να βγάλουμε κάποια συμπεράσματα όσον αφορά και το μέγεθος αποθήκευσης αλλά και μεταφοράς κάθε ψηφιακής ασυμπίεστης εικόνας ανάλογα με το χρωματικό μοντέλο που χρησιμοποιείται. Για κάθε pixel το μοντέλο RGB (θεωρώντας κάθε τιμή ότι αντιστοιχεί σε 8 bit όπως είναι το σύνηθες) χρειάζεται 24 bit για να το απεικονίσει. Το μοντέλο YCbCr 4:4:4 χρειάζεται επίσης 24 bit για κάθε pixel. Τα άλλα μοντέλα όμως χρειάζονται μικρότερο μέγεθος απεικόνισης , το YCbCr 4:2:2 χρειάζεται κατά μέσο όρο 16 bits/pixel ενώ το 4:1:1 χρειάζεται 12 bits/pixel.

Όλα τα μοντέλα είναι αποδοτικά για τον σκοπό που χρησιμοποιούνται αλλά μπορούμε να παρατηρήσουμε ότι για μία εικόνα μεγέθους 800x600 pixel χρειάζονται για την ασυμπίεστη αποθήκευσή της 7,68 megabit ή 960 kilobyte χρησιμοποιώντας το μοντέλο YCbCr 4:1:1. Για τον λόγο αυτό κρίνεται απαραίτητη η χρήση μεθόδων συμπίεσης τόσο για την αποθήκευση όσο και την μεταφορά της ψηφιακής εικόνας.

Κεφάλαιο 2. Συμπίεση εικόνας

Η συμπίεση εικόνας είναι μια εφαρμογή της συμπίεσης δεδομένων η οποία κωδικοποιεί της αρχική εικόνα σε μερικά bit. Ο σκοπός της συμπίεσης εικόνας είναι να μειώσει τις πλεονάζουσες πληροφορίες που περιέχει η εικόνα και να αποθηκευτεί ή να μεταδοθεί σε μία πιο αποτελεσματική μορφή. Ο σκοπός ενός συστήματος συμπίεσης εικόνας είναι να μειώσει την αποθηκευμένη ποσότητα όσο το δυνατόν περισσότερο , αλλά να κρατήσει την μετέπειτα αποκωδικοποιημένη εικόνα πολύ κοντά στην αρχική κατά την απεικόνισή της στην οθόνη.

Η συμπίεση μιας εικόνας μετά την σύλληψή της από κάποιο μέσο σύλληψης(φωτογραφική μηχανή, κάμερα κ.α.) , η οποία γίνεται πάντα με βάση το μοντέλο RGB , ακολουθεί μια συγκεκριμένη ροή. Μετατρέπεται στο πρότυπο YCbCr 4:4:4 για ψηφιακή αποθήκευση και κατόπιν γίνεται υποδειγματοληψία της χρωματικότητας στα πρότυπα YCbCr 4:2:2 ή 4:2:0 έτσι ώστε να περάσει από τον κωδικοποιητή (encoder), που πραγματοποιεί την συμπίεση, με το μικρότερο δυνατό μέγεθος για να γίνει η αποθήκευσή της. Η κωδικοποίηση-συμπίεση μετατρέπει την εικόνα σε ένα bit stream με όλα τα απαραίτητα στοιχεία για την ακριβή (οπτικά) ανακατασκευή της κατά της διαδικασία της αποσυμπίεσης , αποκωδικοποίησης και προβολής της.

Η διαδικασία της συμπίεσης έχει συνήθως τρία στάδια:

- Μείωση της συσχέτισης μεταξύ των γειτονικών pixel
- Κβαντισμός της τιμής των pixel
- Κωδικοποίηση εντροπίας

Τα τρία αυτά στάδια θα παρουσιαστούν αναλυτικά παρακάτω.

2.1 Δείκτες απόδοσης

Για να ελέγξουμε την αποτελεσματικότητα της συμπίεσης υπάρχουν δείκτες οι οποίοι, είτε ποσοτικά είτε ποιοτικά, ελέγχουν την συμπίεση των δεδομένων γενικά, και ειδικότερα στην περίπτωσή μας της εικόνας.

Ο πρώτος ποσοτικός δείκτης είναι η αναλογία συμπίεσης (Compress ratio) οποία δείχνει τι ποσοστό του μεγέθους της αρχικής-ασυμπίεστης εικόνας καταλαμβάνει η τελική-συμπιεσμένη. Η αναλογία συμπίεσης δίνεται από τον τύπο :

$$C_r = \frac{n1}{n2} \qquad (2.1)$$

Όπου n1 είναι το μέγεθος της αρχικής εικόνας και n2 το μέγεθος της κωδικοποιημένης.

Όμως αυτός ο δείκτης δεν παρουσιάζει την ποιοτική σύγκριση του αποτελέσματος της κωδικοποίησης. Για το σκοπό αυτό χρησιμοποιούνται 2 άλλοι δείκτες. Οι δείκτες αυτοί είναι το μέσο τετραγωνικό σφάλμα(mean square error, MSE), το Peak Signal to Noise Ratio (PSNR). Τα περισσότερα συστήματα συμπίεσης εικόνας έχουν ως σκοπό την μείωση του MSE , μεγιστοποιώντας το PSNR. Η σύγκριση για την μέτρηση αυτών των δεικτών γίνεται μεταξύ της αρχικής εικόνας και της εικόνας αφού έχει αποκωδικοποιηθεί ξανά μετά από την συμπίεσή της βάση κάποιου συστήματος συμπίεσης. Οι δείκτες αυτοί δίνονται από τους ακόλουθους τύπους:

$$MSE = \sqrt{\frac{\sum_{x=0}^{W-1}\sum_{y=0}^{H-1}\left(f(x,y) - f'(x,y)\right)^2}{WH}} \quad (2.2)$$

$$PSNR = 20\log_{10}\frac{255}{MSE} \quad (2.3)$$

Όπου f(x,y) η τιμή του pixel στην αρχική εικόνα, f'(x,y) η τιμή του pixel στην εικόνα μετά την αποκωδικοποίηση.

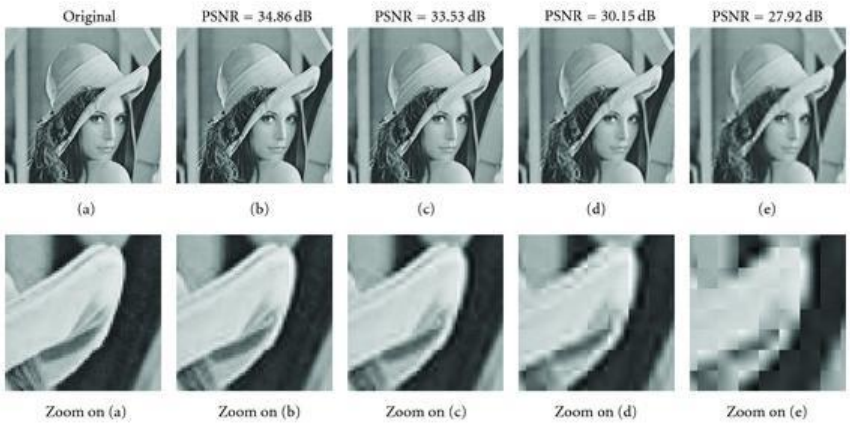
Πρόσφατα άρχισε να εφαρμόζεται και ένας ακόμα δείκτης ο οποίος βοηθάει στην καλύτερη μέτρηση της ποιότητας της συμπίεσης. Ονομάστηκε μετρικός δείκτης δομικής ομοιότητας (Structural Similarity Index Metric, SSIM). Ο δείκτης αυτός σχεδιάστηκε διότι οι παραδοσιακές μέθοδοι του MSE και του PSNR, αποδείχτηκαν ασυμβίβαστες σε σχέση με αυτό που αντιλαμβάνεται το ανθρώπινο μάτι. Σε αντίθεση με αυτές τις τεχνικές, οι οποίες υπολογίζουν τα αντιληπτά λάθη, ο SSIM υπολογίζει την υποβάθμιση της εικόνας λαμβάνοντας υπόψη τις αντιληπτές αλλαγές στις δομικές πληροφορίες, οι οποίες ορίζονται ως οι ισχυρές αλληλεξαρτήσεις μεταξύ γειτονικών pixel οι οποίες προσδιορίζουν την μορφή της εικόνας.

Ο SSIM υπολογίζεται σε ένα παράθυρο μεγέθους NxN στην αρχική εικόνα το οποίο αναπαριστάται ως x και στην τελική εικόνα το οποίο αναπαρίσταται ως y. Ο SSIM δίνεται από τον τύπο:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \quad (2.4)$$

όπου

- μ_x είναι ο μέσος όρος των pixel στο παράθυρο x,
- μ_y είναι ο μέσος όρος των pixel στο παράθυρο y,
- σ_x είναι η διασπορά των τιμών στο παράθυρο x,
- σ_y είναι η διασπορά των τιμών στο παράθυρο y,
- σ_{xy} είναι η συνδιασπορά των τιμών στα x και y,
- $c_1 = k_1 L$, $c_2 = k_2 L$ δύο μεταβλητές για την σταθεροποίηση της διαίρεσης με ασθενή παρονομαστή,
- L το δυναμικό εύρος των τιμών των pixel (συνήθως $2^{\text{αριθμός bit ανά pixel}}$ -1)
- $k_1 = 0.01$, $k_2 = 0{,}03$ εξ ορισμού.



Εικόνα 3 Λεπτομέρεια εικόνας σε σχέση με το PSNR της (http://www.hindawi.com/journals/vlsi/2012/209208/fig9/)

Για να υπολογιστεί η ποιότητα της εικόνας ο τύπος αυτός εφαρμόζεται μόνο στις τιμές του luma της εικόνας. Το αποτέλεσμα είναι μια δεκαδική τιμή μεταξύ -1 και 1, όπου η τιμή 1 είναι εφικτή μόνο όταν πρόκειται για παράθυρα με ακριβώς τις ίδιες τιμές. Συνήθως το παράθυρο έχει μέγεθος 8x8. Μπορεί να μετακινείται κατά ένα pixel την φορά αλλά συνηθίζεται να εφαρμόζεται μόνο σε ένα υποσύνολο από τα δυνατά παράθυρα για να μειωθεί η πολυπλοκότητα των υπολογισμών.

Αντίστοιχα υπάρχει και ο δείκτης δομικής ανομοιότητας DSSIM όπου δίνεται από τον τύπο:

$$DSSIM = \frac{1 - SSIM}{2} \tag{2.5}$$

2.2 Μείωση της συσχέτισης των pixel

Η συσχέτιση μεταξύ των τιμών γειτονικών pixel είναι ο κύριος λόγος που καθίσταται δυνατή η συμπίεση μίας εικόνας. Σε κάθε εικόνα τα γειτονικά pixel έχουν μεγάλη ομοιότητα στην τιμή τους καθώς σε οποιαδήποτε εικόνα υπάρχει ομαλή μετάβαση στο θέμα το οποίο παρουσιάζει. Με το που μειωθεί η συσχέτιση μεταξύ των γειτονικών τιμών, μπορούμε να επωφεληθούμε από τα στατιστικά χαρακτηριστικά και την θεωρία κωδικοποίησης μεταβλητού μήκους για να μειώσουμε το μέγεθος αποθήκευσης της εικόνας. Το κομμάτι αυτό (μείωση της συσχέτισης) είναι το πιο σημαντικό μέρος του αλγορίθμου συμπίεσης εικόνας.

Υπάρχουν πολλές γνωστοί μέθοδοι μείωσης της συσχέτισης των γειτονικών τιμών, με τις κυριότερες από αυτές να είναι:

- Κωδικοποίηση πρόβλεψης

Η κωδικοποίηση πρόβλεψης όπως η DPCM(Differential Pulse Code Modulation) είναι μια lossless μέθοδος κωδικοποίησης, το οποίο σημαίνει ότι η κωδικοποιημένη και η αρχική εικόνα έχουν την ίδια τιμή για κάθε αντίστοιχο στοιχείο. Η DPCM εφευρέθηκε το 1950 από τον C. Chapin Cutler των Bell Labs το 1950.

- Ορθογώνιος Μετασχηματισμός

Ο μετασχηματισμός Karhunen-Loeve(KLT) και ο διακριτός μετασχηματισμός συνημίτονου (Discrete Cosine Transform, DCT) είναι οι δύο πιο γνωστοί ορθογώνιοι μετασχηματισμοί. Οι μέθοδοι συμπίεσης εικόνας οι οποίες βασίζονται στον DCT , όπως το JPEG, είναι lossy μέθοδοι συμπίεσης , δηλαδή το αποτέλεσμά τους έχει μικρή απώλεια σε λεπτομέρειες και μη ανακτήσιμη παραμόρφωση. Στο πρότυπο WebP χρησιμοποιείται και ο μετασχηματισμός Walsh Hadamard ο οποίος είναι και αυτός ορθογώνιος μετασχηματισμός.

- Κωδικοποίηση Υποζώνης (Subband Coding)

Η κωδικοποίηση υποζώνης, όπως ο διακριτός μετασχηματισμός κύματος (Discrete Wavelet Transform, DWT), είναι και αυτή lossy μέθοδος κωδικοποίησης. Ο σκοπός της κωδικοποίησης υποζώνης είναι ο διαχωρισμός του φάσματος μιας εικόνας σε χαμηλοπερατά και υψιπερατά στοιχεία. Μία μέθοδος συμπίεσης η οποία χρησιμοποιεί τον DWT είναι το JPEG2000.

2.3 Κβαντισμός

Ο σκοπός του κβαντισμού μιας εικόνας είναι η μείωση της ακρίβειας έτσι ώστε να επιτευχθεί μεγαλύτερη αναλογία συμπίεσης. Για παράδειγμα, εάν η αρχική εικόνα χρησιμοποιεί 8 bit για να αποθηκεύσει ένα στοιχείο για κάθε pixel, εάν εμείς χρησιμοποιήσουμε 6 bit για να αποθηκεύσουμε την πληροφορία της εικόνας, τότε η ποσότητα που πρέπει να αποθηκευτεί μειώνεται και η εικόνα έχει ήδη συμπιεστεί. Το μειονέκτημα του κβαντισμού είναι ότι είναι μια lossy διαδικασία, και έχει ως αποτέλεσμα την απώλεια μέρους της πληροφορίας. Κάθε μέθοδος συμπίεσης έχει την δική της μέθοδο κβαντισμού ανάλογα με τα αποτελέσματα που θέλει να έχει.

Οι πιο συνηθισμένες μέθοδοι κβαντισμού είναι ο κβαντισμός του χρώματος και ο κβαντισμός συχνότητας.

- Κβαντισμός Χρώματος

Ο Κβαντισμός χρώματος με σκοπό την συμπίεση μιας εικόνας είναι μια μέθοδος η οποία μειώνεις τον αριθμό των χρωμάτων, τα οποία χρησιμοποιούνται σε μια εικόνα. Αυτό είναι πολύ χρήσιμο για απεικόνιση εικόνων σε οθόνες με περιορισμένο αριθμό χρωμάτων και για την αποτελεσματικότερη συμπίεση συγκεκριμένου τύπου εικόνων. Οι πιο συνηθισμένοι αλγόριθμοι κβαντισμού του χρώματος είναι ο αλγόριθμος κοντινότερου χρώματος, ο αλγόριθμος median cut, και ένας αλγόριθμος ο οποίος βασίζεται σε οχτάδεντρα (δεντρικές δομές δεδομένων με οχτώ παιδιά σε κάθε κόμβο).

- Κβαντισμός συχνότητας

Το ανθρώπινο μάτι έχει μεγάλη ευαισθησία στο να παρατηρεί μικρές αλλαγές φωτεινότητας σε μια μεγάλη περιοχή , αλλά δεν μπορεί να παρατηρήσει την ακριβές μέγεθος μιας γρήγορης (μεγάλης συχνότητας) αλλαγής της φωτεινότητα. Αυτό το γεγονός βοηθάει στην μείωση της πληροφορίας που χρειάζεται αγνοώντας τις υψηλές συχνότητες. Αυτό επιτυγχάνεται απλά διαιρώντας κάθε στοιχείο στο πεδίο της συχνότητας με μια σταθερά για το καθένα και κατόπιν στρογγυλοποιώντας το στον κοντινότερο ακέραιο. Σαν αποτέλεσμα αυτού οι περισσότερες υψηλές συχνότητες στρογγυλοποιούνται στο 0 και πολλές από τις υπόλοιπες γίνονται μικροί θετικοί ή αρνητικοί αριθμοί.

Επειδή το ανθρώπινο μάτι είναι πιο ευαίσθητο στην φωτεινότητα παρά στην χρωματικότητα , περεταίρω συμπίεση μπορεί να επιτευχθεί εάν εργαζόμαστε σε ένα χρωματικό χώρο ο οποίος διαχωρίζει την φωτεινότητα και την χρωματικότητα (όπως ο YCbCr) και κβαντίσουμε το κάθε κανάλι ξεχωριστά.

2.4 Κωδικοποίηση εντροπίας

Ο κύριος σκοπός της κωδικοποίησης εντροπίας μιας εικόνας είναι να επιτευχθεί μικρότερο μέσο μήκος δεδομένων για την εικόνα. Οι κωδικοποιήσεις εντροπίας αντιστοιχίζουν κωδικούς στα αντίστοιχα σύμβολα (τιμές) ανάλογα με την πιθανότητα εμφάνισης του κάθε συμβόλου. Γενικά οι κωδικοποιητές εντροπίας χρησιμοποιούνται για να συμπιέσουν τα δεδομένα αντικαθιστώντας σύμβολα τα οποία απεικονίζονται από κωδικούς ίδιου μεγέθους , με «κωδικές λέξεις» το μήκος των οποίων είναι αντιστρόφως ανάλογο με την πιθανότητά τους. Υπάρχουν διάφοροι τύποι κωδικοποιητών εντροπίας και ο καθένας χρησιμοποιεί διαφορετικό

τρόπο για την κωδικοποίηση των δεδομένων.

Οι πιο συχνά χρησιμοποιούμενοι κωδικοποιητές εντροπίας είναι με κώδικα Huffman και η αριθμητική κωδικοποίηση. Εάν τα χαρακτηριστικά της εντροπίας των δεδομένων είναι γνωστά χρήσιμες είναι και πιο απλοί στατικοί κώδικες. Τέτοιοι είναι η κωδικοποίηση γάμμα Elias, η κωδικοποίηση Fibonacci όπως και οι κώδικες Golomb.

Κεφάλαιο 3 . Το πρότυπο εικόνας WebP

Το πρότυπο εικόνας WebP είναι ένα πρότυπο εικόνας ανοιχτού κώδικα το οποίο άρχισε να αναπτύσσεται το 2010 από την Google , μετά την εξαγορά της εταιρίας On2 Technologies, από την οποία πήρε το πρότυπο συμπίεσης βίντεο VP8.

Το WebP είναι ένα επαναστατικό πρότυπο στην τεχνολογία συμπίεσης εικόνας, όπως είναι αντίστοιχα και το αδελφό του πρότυπο WebM, για την συμπίεση βίντεο. Το πρότυπο παρουσιάστηκε σαν ένα lossy πρότυπο συμπίεσης εικόνων πραγματικού χρώματος (μέγεθος pixel 24 bit), το οποίο παράγει εικόνες με μικρότερο μέγεθος σε συγκρίσιμη ποιότητα με το κυρίαρχο πρότυπο αυτή τη στιγμή, το JPEG. Το 2011 , ανακοινώθηκε από την εταιρία η υποστήριξη κίνησης, προφίλ ICC(σετ δεδομένων που χαρακτηρίζει μια συσκευή εισόδου ή αναπαραγωγής χρώματος/ χρωματικός χώρος), μεταδεδομένα XMP(πρότυπο αποθήκευσης επιπρόσθετων χαρακτηριστικών), καθώς και πλακιδιοποίηση (tiling) (αναπαράσταση πολύ μεγάλων εικόνων που αποτελούνται από 16384x16384 κομμάτια). Κατόπιν άρχισε ο πειραματισμός για lossless συμπίεση εικόνων καθώς και υποστήριξη διαφάνειας (alpha frames) τόσο σε lossy όσο και lossless συμπίεση, τα οποία εισήλθαν στην έκδοση 0.2.0 του προτύπου.

Σύμφωνα με την Google το πρότυπο επιτυγχάνει συμπίεση με 45% μικρότερο μέγεθος όταν αρχίζει με ασυμπίεστες εικόνες PNG, και 28% μείωση σε μέγεθος όταν αρχίζει με συμπιεσμένες εικόνες PNG.

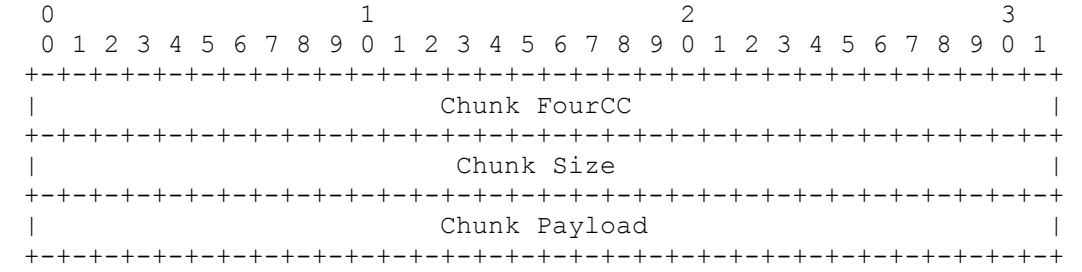
Το πρότυπο WebP προτάθηκε και για να αντικαταστήσει και το GIF όσον αφορά τις κινούμενες εικόνες, έχοντας ως πλεονεκτήματα το βάθος χρώματος (24 bit αντί για 8), η δυνατότητα συνδυασμού lossy και lossless frames, η υποστήριξη διάφανων alpha frames,την δυνατότητα αναζήτησης στα frames αλλά και το βασικότερο το μειωμένο κατά 64% μέγεθος όταν μετατρέπεται κάποιο GIF σε lossy WebP , ή 19% όταν μετατρέπεται σε lossless.

Όλες οι πράξεις γίνονται με μεταβλητές μεγέθους 16 bit, γεγονός το οποίο ενισχύει την αποτελεσματικότητα του αλγορίθμου.

Το διάγραμμα του προτύπου WebP φαίνεται στην παρακάτω εικόνα. Θα αναλύσουμε κάθε κομμάτι ξεχωριστά στην συνέχεια.

3.1 RIFF

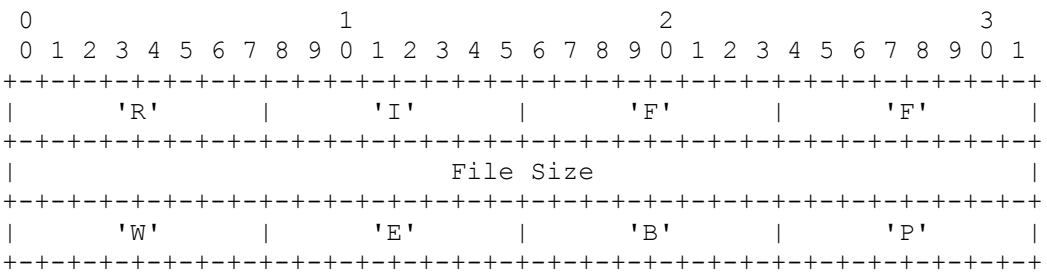
Ο περιέκτης(container) RIFF είναι ένας container γενικού τύπου ο οποίος αποθηκεύει δεδομένα σε κομμάτια με ετικέτες. Χωρίς περιεχόμενο ο περιέκτης RIFF έχει μόνο 20 byte κεφαλίδα και υποστήριξη για μεταδεδομένα. Στην γενική του μορφή έχει την ακόλουθη μορφή:



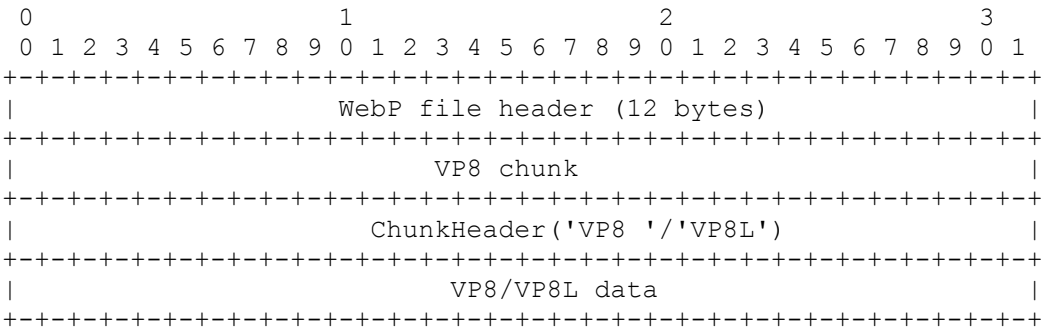
Όπου το Chunk FourCC περιέχει έναν κωδικό τεσσάρων χαρακτήρων ASCII για την αναγνώριση του κομματιού, το Chunk Size περιέχει το μέγεθος των δεδομένων

και το Chunk Payload το ωφέλιμο μέγεθος των δεδομένων.

Μετά την κεφαλίδα του file container ακολουθεί η κεφαλίδα του αρχείου WebP η οποία περιέχει τις βασικές πληροφορίες συμπίεσης του αρχείου, όπως μέγεθος, τύπος συμπίεσης και τα δεδομένα. Η κεφαλίδα του WebP είναι της μορφής:



Στα πρώτα 4 byte έχει τους χαρακτήρες *'R', 'I', 'F', 'F'* όπου προσδιορίζουν τον περιέκτη, μετά ακολουθεί το μέγεθος της εικόνας και τέλος τα 4 byte που έχουν τους χαρακτήρες *'W', 'E', 'B', 'P'* που προσδιορίζουν ότι το αρχείο είναι τύπου WebP. Τέλος ανάλογα με το αν έχουμε lossy ή lossless συμπίεση ακολουθεί άλλη μια κεφαλίδα της μορφής:



Η μόνη διαφορά μεταξύ lossy και lossless συμπίεσης βρίσκεται στα bytes 9-12 όπου εάν έχουμε lossy υπάρχουν οι χαρακτήρες *'V', 'P', '8', ' '* ενώ αν έχουμε lossless οι χαρακτήρες *V', 'P', '8', 'L'.*

3.2 Τεχνολογία WebP

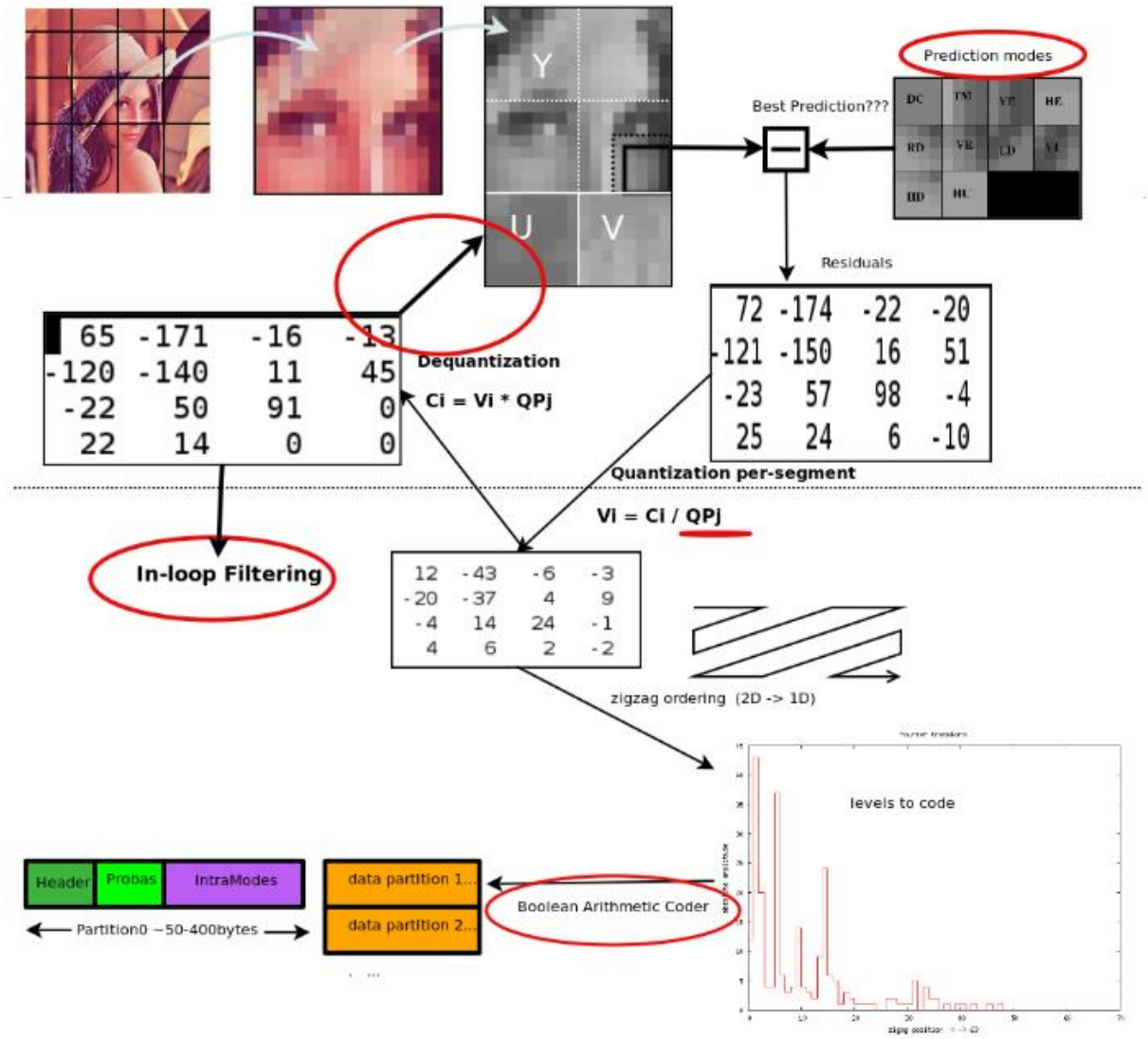
Ο lossy αλγόριθμος συμπίεσης WebP βασίζεται στην intra-frame κωδικοποίηση του κωδικοποιητή VP8 και περιέχεται στον περιέκτη αρχείου RIFF(Resource Interchange File Format). Λόγω της προέλευσής του από τον κωδικοποιητή VP8 βασίζεται σε μετασχηματισμούς των block της εικόνας με 8 bit βάθος χρώματος και μοντέλο φωτεινότητας-χρωματικότητας με υποδειγματοληψία χρωματικότητας 1:2, δηλαδή χρησιμοποιεί το μοντέλο YCbCr 4:2:0. Για να μετασχηματίσει το κάθε block εκτός από το προς επεξεργασία block χρησιμοποιεί τα τρία block που βρίσκονται από πάνω καθώς και ένα block από τα αριστερά. Υπάρχουν 4 κύριες μέθοδοι πρόβλεψης block: DC,TM, Οριζόντια και κάθετη. Κάθε block κατόπιν συμπιέζεται σε ένα 4x4 subblock μέσω DCT και μετασχηματισμού Walsh-Hadamard. Και οι δύο μετασχηματισμοί γίνονται με αριθμούς σταθερής υποδιαστολής για την αποφυγή λαθών στρογγυλοποίησης. Το μέγιστο μέγεθος εικόνας που μπορεί να συμπιεστεί έχει μήκος πλευράς 16383 pixel(14 bit).

Η lossless κωδικοποίηση βασίζεται σε εξειδικευμένους κωδικοποιητές εντροπίας για κάθε ένα ξεχωριστό κανάλι, εκμεταλλευόμενη την δισδιάστατη τοπικότητα των αναφορών και μια cache πρόσφατα χρησιμοποιούμενων χρωμάτων. Αυτά

συμπληρώνουν τις κλασσικές μεθόδους όπως η κωδικοποίηση Huffman, η κωδικοποίηση λεξικού και οι μετασχηματισμοί ευρετηρίου χρωμάτων.

Στην διπλωματική αυτή εργασία θα ασχοληθούμε μόνο με lossy συμπίεση καθώς ασχοληθήκαμε με την έκδοση 0.1.3 του προτύπου ενώ η lossless εισήχθη στο πρότυπο 0.2.0.

3.3 Διάγραμμα Lossy WebP



Εικόνα 4 Ο κωδικοποιητής του πρότυπου WebP (https://developers.google.com/speed/webp/docs/compression)

3.4 Μοντέλα Πρόβλεψης Block

Το πρότυπο συμπίεσης WebP χρησιμοποιεί μοντέλα πρόβλεψης για το κάθε block που έπεται να επεξεργαστεί. Τα μοντέλα αυτά πρόβλεψης είναι μέρος του κωδικοποιητή VP8, ο οποίος χρησιμοποιείται για βίντεο, αλλά εφαρμόζονται επιτυχώς και σε στατική εικόνα.

Υπάρχουν 4 μοντέλα πρόβλεψης για κάθε block.

- H_PRED (Horizontal Prediction): Γεμίζει κάθε στήλη με την τιμή της σειράς πάνω από το block.
- V_PRED (Vertical Prediction): Γεμίζει κάθε σειρά με την τιμή της στήλης αριστερά από το block.
- DC_PRED (DC prediction): Το επόμενο block γεμίζει με την μέση τιμή από την σειρά πάνω από το block και την στήλη αριστερά από το block.
- TM_PRED (True Motion Prediction): Επιπλέον της στήλης αριστερά και της σειράς πάνω από το block η True Motion μέθοδος πρόβλεψης χρησιμοποιεί και την τιμή του pixel C το οποίο είναι πάνω και αριστερά από το block. Υπολογίζονται οι διαφορές μεταξύ των pixel της πάνω σειράς και της στήλης αριστερά ξεκινώντας από το pixel C και έπειτα πολλαπλασιάζονται μεταξύ τους.
  Ο τρόπος πρόβλεψης True Motion είναι μοναδικός στο πρότυπο VP8.
  Στον παρακάτω πίνακα φαίνεται ο τρόπος πρόβλεψης σε ένα 4x4 block.

C	**A_0**	**A_1**	**A_2**	**A_3**
L_0	X_{00}	X_{01}	X_{02}	X_{03}
L_1	X_{10}	X_{11}	X_{12}	X_{13}
L_2	X_{20}	X_{21}	X_{22}	X_{23}
L_3	X_{30}	X_{31}	X_{32}	X_{33}

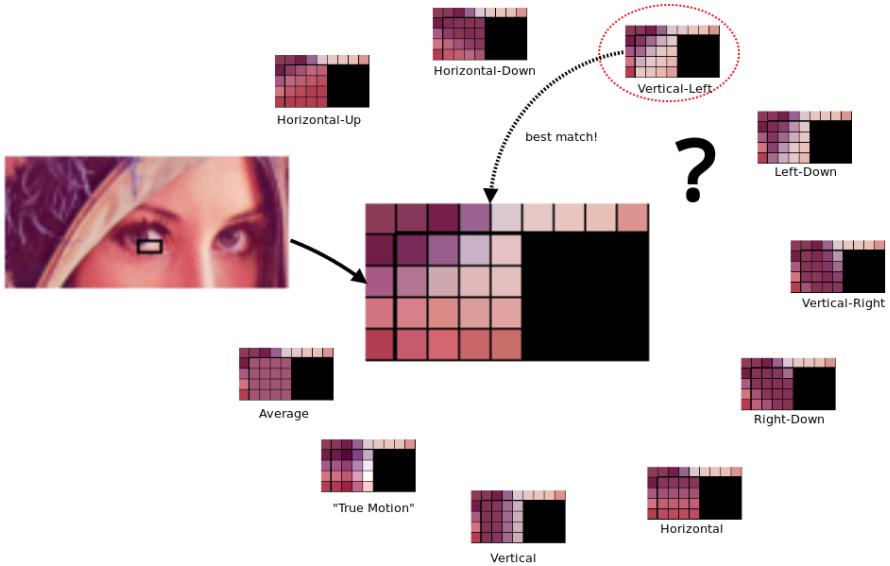
Τα C,L,A αντιπροσωπεύουν ανακατασκευασμένα pixel από προηγούμενα block , και το X_{00} έως X_{33} τις προβλέψεις για αυτό το συγκεκριμένο block. Για τον υπολογισμό των τιμών αυτών χρησιμοποιείται ο τύπος:

$$X_{ij} = L_i + A_j - C\ , (i,j = 0,1,2,3) \qquad (3.1)$$

Για block μεγαλύτερου μεγέθους (8x8, 16x16) η πρόβλεψη λειτουργεί με τον ίδιο τρόπο. Αυτός ο τρόπος πρόβλεψης είναι και ο πιο συχνά χρησιμοποιούμενος, χρησιμοποιείται περίπου 20% - 45%.

Για τα block με μέγεθος 4x4 υπάρχουν και άλλοι 6 τρόποι πρόβλεψης για να υπολογίζουν τα pixel προς άλλες κατευθύνσεις.

Όλοι αυτοί οι τρόποι μαζί συνδυάζονται ώστε να πετύχουν υψηλό βαθμό συμπίεσης για κάθε εικόνα.



Εικόνα 5 Τα μοντέλα πρόβλεψης για 4x4 block (https://developers.google.com/speed/webp/docs/compression)

3.5 DCT(Discrete Cosine Transform)

Ο διακριτός μετασχηματισμός συνημίτονου είναι ένας ορθογώνιος μετασχηματισμός ανεξάρτητος από την είσοδο του συστήματος στον οποίο εφαρμόζεται και έχει αποδειχθεί ότι είναι μόνο λίγο χειρότερος από τον μετασχηματισμό Karhunen-Loeve ο οποίος υπολογίζεται έτσι ώστε να πετύχει την βέλτιστη συμπίεση ενέργειας, δεδομένου μίας συγκεκριμένης εισόδου. Εν αντιθέσει με την πολυπλοκότητα του KLT ο μετασχηματισμός DCT έχει απλή υλοποίηση και ευκολότερη εφαρμογή σε κινούμενες εικόνες. Επιπλέον ο DCT είναι διαχωρίσιμος και έχει εύκολες υλοποιήσεις.

Το πρότυπο WebP χρησιμοποιεί έναν απλό δισδιάστατο DCT σαν βάση του μετασχηματισμού για την κωδικοποίηση και ο αποκωδικοποιητής χρησιμοποιεί έναν αντίστροφο δισδιάστατο DCT για κάθε 4x4 block.

Ο μαθηματικός ορισμός του DCT και του αντίστροφου DCT είναι:

Discrete Cosine Transform

$$F(u,v) = \frac{2}{N}C(u)C(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\cos\left(\frac{\pi(2x+1)u}{2N}\right)\cos\left(\frac{\pi(2y+1)v}{2N}\right) \quad (3.2)$$

Για *u*=0,…,N-1 και *v*=0,…,N-1, όπου N το μέγεθος του block (N=4 για 4x4) και

$$C(k) = \begin{cases} \frac{1}{\sqrt{2}}, k = 0 \\ 1, \alpha\lambda\lambda\iota\acute{\omega}\varsigma \end{cases}$$

Inverse Discrete Cosine Transform

$$f(x,y) = \frac{2}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} C(u)C(v)F(u,v)\cos\left(\frac{\pi(2x+1)u}{2N}\right)\cos\left(\frac{\pi(2y+1)v}{2N}\right) \quad (3.3)$$

Για *x*=0,…,N-1 και *y*=0,…,N-1 όπου N το μέγεθος του block.

Το *f(x,y)* είναι η τιμή του κάθε pixel στο επιλεγμένο block και το *F(u,v)* είναι η τιμή του συντελεστή DCT μετά τον μετασχηματισμό. Ο μετασχηματισμός ενός block μεγέθους NxN είναι ένα block μεγέθους και αυτό NxN.

Ο DCT είναι στενά συνδεδεμένος με τον διακριτό μετασχηματισμό Fourier καθώς και οι δύο παίρνουν ένα σετ δεδομένων από το πεδίο του χώρου και το μετατρέπουν στο πεδίο της συχνότητας. Παρόλα αυτά ο DCT είναι πιο κατάλληλος για δύο κύριους λόγους:

1. Ο DCT μπορεί να συγκεντρώσει την ενέργεια ενός μετασχηματισμένου σήματος σε χαμηλή συχνότητα, ενώ ο DFT όχι. Σύμφωνα με το θεώρημα του Parseval η ενέργεια είναι ίδια τόσο στο πεδίο του χώρου όσο και στο πεδίο της συχνότητας. Επειδή το ανθρώπινο μάτι είναι λιγότερο ευαίσθητο στις

συνιστώσες χαμηλών συχνοτήτων, μπορούμε να εστιάσουμε σε αυτές και να μειώσουμε την συνεισφορά των υψηλών αφού έχουμε κάνει τον μετασχηματισμό.
2. Για την συμπίεση εικόνας , ο DCT μπορεί να μειώσει τον τετραγωνισμό του αποτελέσματος, ενώ ο DFT όχι.

Μετά τον μετασχηματισμό, το στοιχείο στην πάνω αριστερά γωνία ανταποκρίνεται σε μηδενική συχνότητα και ονομάζεται συντελεστής DC ενώ όλα τα υπόλοιπα ονομάζονται συντελεστές AC.

3.6 Μετασχηματισμός Walsh – Hadamard

Ο μετασχηματισμός δεύτερης τάξεως, που ακολουθεί τον DCT κατά την συμπίεση μιας εικόνας στο πρότυπο WebP, είναι ο μετασχηματισμός Walsh – Hadamard(WHT), ο οποίος έχει σχεδιαστεί έτσι ώστε να εκμεταλλευτεί την συσχέτιση μεταξύ των πρωτοβάθμιων συντελεστών DC των 16 4x4 block που υπάρχουν μέσα στο κυρίαρχο macroblock, το οποίο είναι το κύριο κομμάτι επεξεργασίας. Όπως ακριβώς και με τον DCT ο αντίστροφος μετασχηματισμός ορίζεται για την αποκωδικοποίηση.

Ο WHT μπορεί να περιγραφεί από τον τύπο:

$$Y = HXH^T \tag{3.4}$$

Όπου *X* και *Y* είναι τα 4x4 block εισόδου και εξόδου αντίστοιχα. Ο πίνακας *H* ορίζεται ως:

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \tag{3.5}$$

Και ο πίνακας H^T ο ανάστροφος του *H*.

3.7 Κβαντισμός

Ο κβαντισμός των τιμών στο πρότυπο WebP είναι και αυτός πρωτότυπος και βασίζεται στο κβαντισμό που έχει το πρότυπο κωδικοποίησης VP8. Πρόκειται για ένα προσαρμόσιμο σύστημα κβαντισμού το οποίο ορίζει 128 στάθμες. Για κάθε εικόνα και κάθε block ο κβαντιστής του προτύπου επιλέγει διαφορετικά για κάθε μια από τις έξι συνιστώσες της συχνότητας: DC συνιστώσα luma πρώτης τάξεως, AC συνιστώσα luma πρώτης τάξεως, DC συνιστώσα luma δεύτερης τάξεως, AC συνιστώσα luma πρώτης τάξεως, DC συνιστώσα chroma, AC συνιστώσα chroma. Επιπλέον η εικόνα χωρίζεται σε 4 κομμάτια και το κάθε ένα έχει τις δικές του παραμέτρους κβάντισης.

3.8 Κωδικοποίηση εντροπίας

Για την κωδικοποίηση εντροπίας το πρότυπο WebP χρησιμοποιεί έναν Boolean αριθμητικό κωδικοποιητή και ένα ψευδό-Huffman δέντρο.

Ο Boolean αριθμητικός κωδικοποιητής κωδικοποιεί μία δυαδική (0/1) τιμή κάθε φορά και χρησιμοποιείται για την χωρίς απώλειες συμπίεση μιας ακολουθίας από δυαδικούς αριθμούς των οποίων οι πιθανότητες είναι αρκετά γνωστές.

Το δεύτερο σκέλος της κωδικοποίησης εντροπίας είναι ένα ψευδό-Huffman δέντρο. Σε ένα τέτοιο σχήμα δημιουργείται ένα δυαδικό δέντρο αναζήτησης για ένα σετ τιμών, όπως ακριβώς δημιουργείται και το δέντρο Huffman, και κάθε σύμβολο μπορεί να

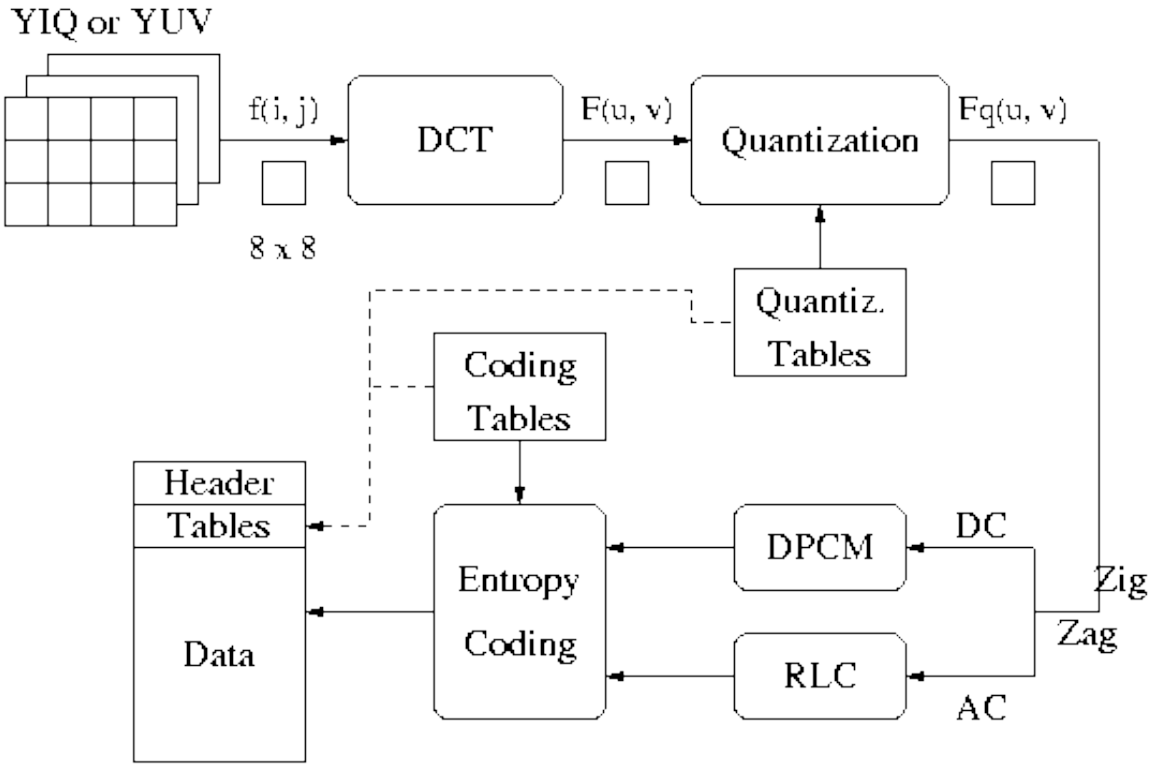
αναπαρασταθεί από μία δυαδική ακολουθία η οποία δημιουργείται διασχίζοντας το δέντρο από την ρίζα μέχρι το αντίστοιχο φύλλο. Κάθε κόμβος ο οποίος δεν είναι έχει μια πιθανότητα, σχετιζόμενη με την πιθανότητα να διασχισθεί προς τον αριστερό κλάδο.

Λόγω αυτής της δυαδικότητας και της αποθήκευσης δεδομένων σε αυτό το ψευδό-Huffman δέντρο η κωδικοποίηση είναι συνεπής σε όλες τις τιμές του τελικού αποτελέσματος, συμπεριλαμβανομένων του τρόπου κωδικοποίησης, των τιμών των pixel, κ.ο.κ. Αυτό βοηθάει στην επαναχρησιμοποίηση των κομματιών και για παράλληλες υλοποιήσεις τόσο σε software όσο και σε hardware.

Κεφάλαιο 4. Σύγκριση αλγορίθμων WebP και JPEG/GIF

Την περίοδο αυτή το κυρίαρχο πρότυπο συμπίεσης εικόνας είναι το JPEG, το οποίο προτάθηκε και αναπτύχθηκε από το Joint Photographic Experts Group το 1992 για την συμπίεση στατικών εικόνων, λόγω του γεγονότος ότι η ψηφιακή εικόνα είχε αρχίσει να αναπτύσσεται και οι δυνατότητες των αποθηκευτικών μέσων ήταν ακόμα περιορισμένες.

Ο κωδικοποιητής του προτύπου JPEG έχει την ακόλουθη μορφή:



Εικόνα 6 Ο κωδικοποιητής του προτύπου JPEG(http://www-i6.informatik.rwth-aachen.de/web/Misc/Coding/365/li/material/notes/Chap4/Chap4.2/Chap4.2.html)

Μπορούμε να παρατηρήσουμε ότι η γενικότερη ροή κωδικοποίησης-συμπίεσης είναι ίδια με αυτή του WebP αλλά ουσιαστικά το μόνο κομμάτι το οποίο μοιράζονται οι δύο αλγόριθμοι είναι ο DCT. Η διαδικασία κβάντισης αλλά και της κωδικοποίησης εντροπίας είναι διαφορετικές στα δύο πρότυπα.

Σε αυτό το κεφάλαιο θα κάνουμε σύγκριση των δύο προτύπων με βάση ποσοτικούς, αλλά και ποιοτικούς δείκτες απόδοσης.

4.1 Ποσοτική Σύγκριση

Σύμφωνα με τα δεδομένα τα οποία αναφέρει η Google , στην εισαγωγή του προτύπου για την ποσοτική απόδοσή του, προσδιορίζει ότι οι εικόνες που συμπιέζονται με το WebP έχουν κατά 30% μικρότερο μέγεθος από αυτές που συμπιέζονται με JPEGστην ίδια ποιότητα εικόνας.

Κάνοντας έρευνα μπορέσαμε να βρούμε και να παρουσιάσουμε ποσοτικά αποτελέσματα για να επιβεβαιώσουμε την σύγκριση αυτή.

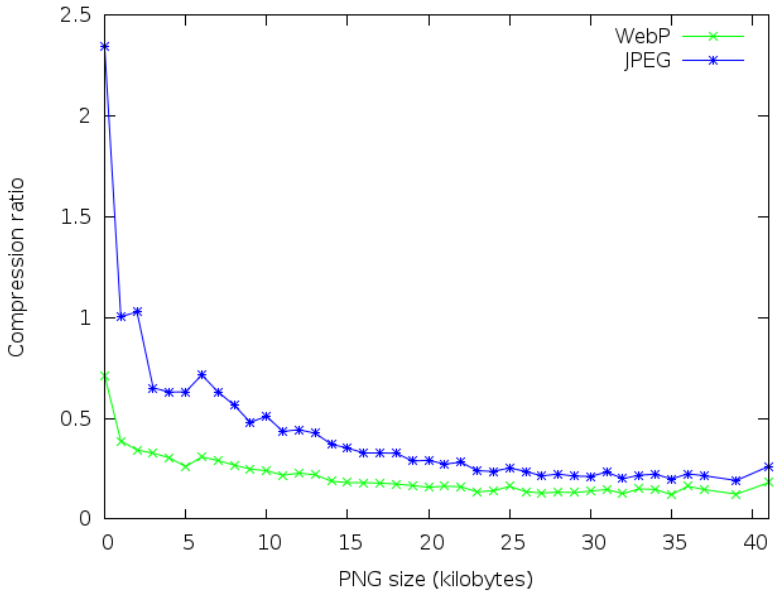
Στην συνέχεια θα παραθέσουμε οπτικά συγκρίσιμες εικόνες με τα μεγέθη αρχείου για τα δύο συγκρινόμενα πρότυπα.

Εικόνα 7 JPEG (Πηγή : https://www.andrewmunsell.com/blog/jpg-vs-webp)

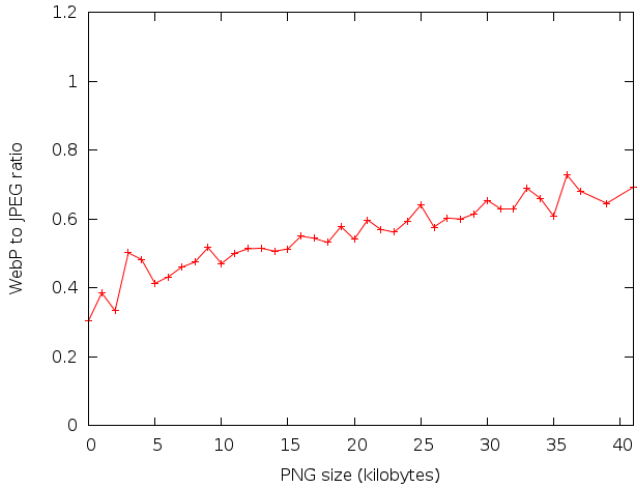
Εικόνα 8 WebP (Πηγή : https://www.andrewmunsell.com/blog/jpg-vs-webp)

Στις παραπάνω εικόνες μπορούμε να παρατηρήσουμε ότι για την ίδια οπτικά ποιότητα το WebP παρουσιάζει πολύ μικρότερο μέγεθος. Συγκεκριμένα θέτοντας την ποιότητα του JPEG στο 4 (αντιστοιχεί σε 69% ποιότητα) και του WebP στο 50% ποιότητα, οπτικά έχουμε το ίδιο αποτέλεσμα και στις δύο εικόνες με το μέγεθος του WebP να είναι περίπου στο 50%.

Με την ίδια αρχική εικόνα τα αποτελέσματα για συμπίεση ανάλογα με το μέγεθος εικόνας είναι:



Εικόνα 9 Σύγκριση Αναλογίας Συμπίεσης (https://github.com/EverythingMe/webp-test)



Εικόνα 10 Σύγκριση Αποτελεσμάτων συμπίεσης (Μέγεθος) (https://github.com/EverythingMe/webp-test)

Βλέπουμε ότι ο συντελεστής συμπίεσης του WebP είναι μικρότερος (επιτυγχάνει υψηλότερη συμπίεση). Όσον αφορά την ποιότητα βλέπουμε στον παρακάτω πίνακα ότι τα δύο πρότυπα έχουν συγκρίσιμες τιμές στο SSIM με το μέγεθος του WebP να είναι το μισό από αυτό του JPEG.

	Average ratio to original size	**Average DSSIM**
WebP	20.4%	0.09126
JPEG	40.4% (with alpha channel removed)	0.09961 (compared to source without alpha channel)

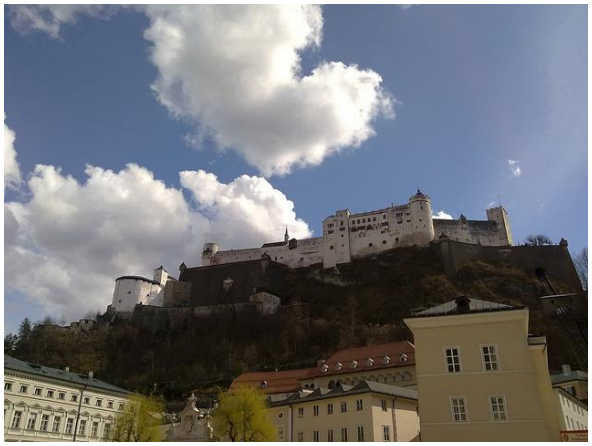
Πίνακας 2 Σύγκριση μεγέθους ως προς την αρχική εικόνα και δείκτη ανομοιότητας (http://geeks.everything.me/2013/04/24/why-we-like-webp/)

Συγκρίνοντας ακόμα τα μεγέθη των αποτελεσμάτων της συμπίεσης (Εικόνα 10) βλέπουμε ότι όντως το πρότυπο WebP παράγει εικόνες πολύ μικρότερες σε μέγεθος από αυτές του JPEG.

4.2 Ποιοτική σύγκριση

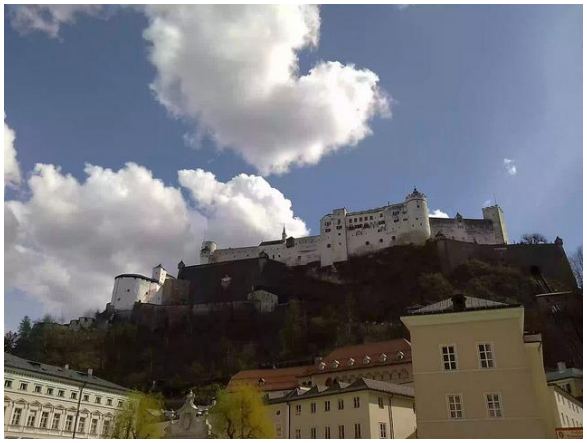
Κρατώντας ίδιο το μέγεθος της εικόνας του αποτελέσματος μπορούμε να παρατηρήσουμε χωρίς να χρησιμοποιήσουμε δείκτες, καθώς το αποτέλεσμα είναι εμφανές μέσω γυμνού οφθαλμού, ότι για το ίδιο μέγεθος εικόνας το αποτέλεσμα που παρουσιάζει το WebP είναι πραγματικά με διαφορά καλύτερο από το αποτέλεσμα του JPEG για το ίδιο μέγεθος.

Παρουσιάζονται τα οπτικά αποτελέσματα αυτής της διαφοράς στην συνέχεια.



Εικόνα 11 Αρχική Εικόνα (http://carlodaffara.conecta.it/a-small-webp-test/)

Εικόνα 12 Εικόνα συμπιεσμένη με JPEG (http://carlodaffara.conecta.it/a-small-webp-test/)



Εικόνα 13 Εικόνα συμπιεσμένη με WebP σε ίδιο μέγεθος με την παραπάνω JPEG (http://carlodaffara.conecta.it/a-small-webp-test/)

4.3 Σύγκριση με το πρότυπο GIF

Το πρότυπο WebP για κινούμενη εικόνα εισήλθε στην έκδοση 0.2.0 του πρότυπου και δεν υπάρχουν συγκριτικές μελέτες της απόδοσης. Σύμφωνα με την Google προσφέρει έως 64% μικρότερο μέγεθος με υποστήριξη συμπίεσης, alpha frames μεγέθους 8 bit (το GIF υποστηρίζει 1 bit) και αναζήτησης μέσα στα frames, καθώς και μεγαλύτερο βάθος χρώματος (24 bit ενώ το GIF υποστηρίζει 8 bit).

Στα μειονεκτήματα που αναφέρονται είναι η μεγαλύτερη χρήση επεξεργαστικής ισχύος για την αναπαραγωγή των κινούμενων εικόνων σε σύγκριση με το GIF, δηλαδή χρειάζεται περίπου 0.57 φορές παραπάνω χρόνο για την αποκωδικοποίηση και αναπαραγωγή.

4.4 Λόγος επιλογής WebP

Το πρότυπο WebP παρουσιάζει όπως είδαμε πολύ μεγάλη συμπίεση σε σχέση με την αρχική εικόνα και αυτός είναι ένας πολύ καλός λόγος για την χρησιμοποίησή του για εφαρμογές όπου χρειάζεται μικρό μέγεθος ή γρήγορη μετάδοση δεδομένων.

Το μειονέκτημά του είναι η πολυπλοκότητά του και ότι χρειάζεται μεγαλύτερη υπολογιστική δύναμη για την κωδικοποίηση και αποκωδικοποίησή του. Γεγονός που κάνει τα ήδη υπάρχοντα πρότυπα πιο ικανά σε συστήματα με μικρή επεξεργαστική ισχύ.

Κεφάλαιο 5. Υλοποίηση – Αποτελέσματα

Στην διπλωματική αυτή εργασία ζητήθηκε η υλοποίηση του αλγορίθμου συμπίεσης WebP σε αναδιατασσόμενη λογική, με την γλώσσα περιγραφής υλικού VHDL, για την ανάπτυξη του συστήματος σαν μέρος ενός ενσωματωμένου συστήματος συμπίεσης εικόνας. Η πορεία της διπλωματικής εργασίας αυτής παρουσιάζεται σε αυτό το κεφάλαιο. Υλοποίησα μέρος του προτύπου WebP και συγκεκριμένα τις πιο χρονοβόρες συναρτήσεις αλλά και ολόκληρο τον μηχανισμό πρόβλεψης της έκδοσης 0.1.3 του WebP.

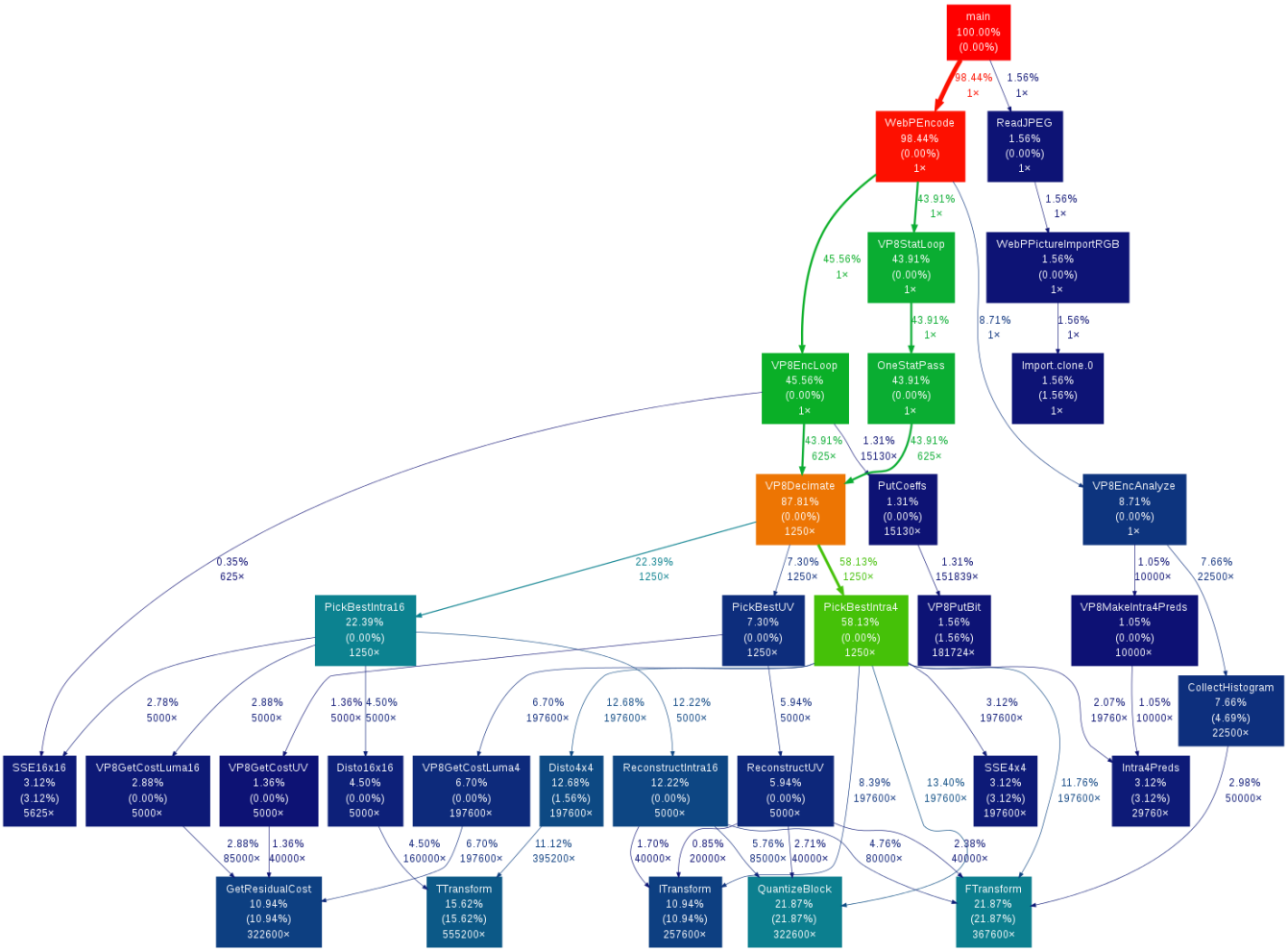
5.1 Ανάλυση του συστήματος

Η πρώτη εργασία για την σχεδίαση και ανάπτυξη του συστήματος έγινε με το profiling του αρχικού αλγορίθμου σε software, δηλαδή τον υπολογισμό της χρονικής πολυπλοκότητας της κάθε συνάρτησης. Το profiling του αρχικού κώδικα έγινε σε έναν υπολογιστή με 6GB μνήμη επεξεργαστή Intel i7 920 (4 πυρήνες με χρονισμό 2,67 GHz ο καθένας) σε λειτουργικό σύστημα Ubuntu 12.04 χρησιμοποιώντας το εργαλείο gprof. Τα αποτελέσματα του profiling (παρουσιάζοντας μόνο τις χρονοβόρες συναρτήσεις) ήταν τα εξής.

% time	cumulative seconds	self		name
		seconds	calls	
21.88	0.14	0.14	367600	FTransform
21.88	0.28	0.14	322600	QuantizeBlock
15.62	0.38	0.10	555200	TTransform
10.94	0.45	0.07	322600	GetResidualCost
10.94	0.52	0.07	257600	ITransform
4.69	0.55	0.03	22500	CollectHistogram
3.12	0.57	0.02	197600	SSE4x4
3.12	0.59	0.02	29760	Intra4Preds
3.12	0.61	0.02	5625	SSE16x16
1.56	0.62	0.01	197600	Disto4x4
1.56	0.63	0.01	181724	VP8PutBit

Στην πρώτη στήλη παρουσιάζεται το ποσοστό του χρόνου που καταναλώνει η κάθε συνάρτηση ποσοστιαία επί του συνολικού χρόνου, στην δεύτερη ο πραγματικός χρόνος που καταναλώνει σε δευτερόλεπτα προσθέτοντας τους από πάνω χρόνους, η τρίτη στήλη παρουσιάζει τα δευτερόλεπτα που καταναλώνει κάθε συνάρτηση μόνη της και η τέταρτη το πόσες κλήσεις γίνονται σε κάθε συνάρτηση.

Γραφικά μαζί με τις κλήσεις από συνάρτηση σε συνάρτηση το διάγραμμα είναι:



Εικόνα 14 Γραφική απεικόνιση των κλήσεων των συναρτήσεων του αλγόριθμου WebP

Από τον πίνακα και το διάγραμμα μπορούμε να παρατηρήσουμε ότι οι κύριες χρονοβόρες συναρτήσεις είναι οι 3 μετασχηματισμοί καθώς και ο υπολογισμός του ιστογράμματος της εικόνας. Επομένως επιλέξαμε να υλοποιήσουμε τις 4 αυτές συναρτήσεις.

5.2 Σχεδίαση του Συστήματος

Μετά την ανάλυση, έπρεπε να σχεδιάσω το σύστημα. Παρατήρησα ότι η συνάρτηση που υπολόγιζε το Ιστόγραμμα της εικόνας το έκανε αφού είχε υπολογίσει τον DCT ο οποίος υλοποιούταν από την συνάρτηση FTransform.

Άρα επιλέξαμε να υλοποιήσουμε τον DCT πρώτα από όλους. Επειδή αποτελείται μόνο από αθροιστές και έναν απλό πολλαπλασιαστή αποφασίσαμε την ασύγχρονη υλοποίηση του και την βέλτιστη χρονική πολυπλοκότητα σε βάρος της χωρικής.

Ακολούθως υλοποίησα την συνάρτηση η οποία ελέγχει το ιστόγραμμα συχνοτήτων σε ένα block και στην συνέχεια τις συναρτήσεις υλοποίησης του μετασχηματισμού Walsh – Hadamard (TTransform) , Κβαντισμού και αντίστροφου μετασχηματισμού (ITransforn).

Τέλος συνέθεσα την συνάρτηση του Ιστογράμματος με τις συναρτήσεις πρόβλεψης και υλοποίησα τον μηχανισμό πρόβλεψης.

Η σχεδίαση του συστήματος έγινε top-down , και η υλοποίηση έγινε bottom-up.

5.3 Υλοποίηση

Η υλοποίηση του συστήματος έγινε με σκοπό την βελτιστοποίηση στον χρόνο και την παραλληλοποίηση του αρχικού αλγόριθμου. Με αυτό το σκοπό αναπτύχθηκαν τα loop του αρχικού κώδικα, ώστε οι διαδικασίες να γίνονται παράλληλα και χρησιμοποιήθηκαν μέθοδοι ώστε να ελαχιστοποιήσουν την χρονική καθυστέρηση του κάθε τμήματος. Η υλοποίηση έγινε με την χρήση του εργαλείου Xilinx ISE 14.4 σε γλώσσα VHDL.

Το κάθε module υλοποιήθηκε από εμένα μέσω των αρχών των οποίων διδάχθηκα στα μαθήματα του τομέα MHL του Πολυτεχνείου Κρήτης. Έγινε μελέτη για την γρηγορότερη παραγωγή του αποτελέσματος μετά την είσοδο των κατάλληλων σημάτων και μεταβλητών.

5.3.1 Encoder Analyze

Η συνάρτηση Encoder Analyze αποτελεί το top module (κεντρικό κομμάτι) του συστήματός μου. Περιλαμβάνει τους 3 τρόπους για τις προβλέψεις που χρησιμοποιούνται για την κωδικοποίηση πρόβλεψης στο πρότυπο WebP. Αποτελεί υλοποίηση της συνάρτησης MBAnalyze η οποία βρίσκεται στο αρχείο analysis.c του αρχικού αλγορίθμου σε C.

Στη σχεδίαση περιλαμβάνει τις 3 συναρτήσεις MBAnalyzeBestIntra16Mode, MBAnalyzeBestIntra4Mode και MBAnalyzeBestUVMode. Το σύστημα έχει σαν είσοδο ένα πίνακα μεγέθους 16x24 ο οποίος έχει τις συνιστώσες luma και chroma για την απεικόνιση 256 pixel (yuv_in). Οι συνιστώσες luma αποτελούν το 16x16 πίνακα ενώ οι chroma το υπόλοιπο του πίνακα (8x16). Με το μέγεθος κάθε απεικόνισης συνιστώσας να είναι στα 8bit (εύρος τιμών 0-255) η είσοδος θα είχε μέγεθος 3072 bit.Λόγω όμως του ότι χρειαζόμαστε και επιπλέον pixel γύρω από το block χρησιμοποιούμε είσοδο μεγέθους 4096 bit.

Το block μεγέθους 16x24 αποτελείται από 4 συνεχόμενα block της μορφής :

Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8	Y9	Y10	Y11	Y12	Y13	Y14	Y15	Y16
Y17	Y18	Y19	Y20	Y21	Y22	Y23	Y24	Y25	Y26	Y27	Y28	Y29	Y30	Y31	Y32
Cb1	Cr1	Cb2	Cr2	Cb3	Cr3	Cb4	Cr4	Cb5	Cr5	Cb6	Cr6	Cb7	Cr7	Cb8	Cr8
Y33	Y34	Y35	Y36	Y37	Y38	Y39	Y40	Y41	Y42	Y43	Y44	Y45	Y46	Y47	Y48
Y49	Y50	Y51	Y52	Y53	Y54	Y55	Y56	Y57	Y58	Y59	Y60	Y61	Y62	Y63	Y64
Cb9	Cr9	Cb10	Cr10	Cb11	Cr11	Cb12	Cr12	Cb13	Cr13	Cb14	Cr14	Cb15	Cr15	Cb16	Cr16

Εικόνα 15 Δεδομένα YCbCr για 64 pixel στην μορφή εισόδου στο προς επεξεργασία block (Με ίδια χρώματα απεικονίζονται οι τιμές που ανήκουν στην ίδια τετράδα pixel)

Επιπλέον έχει σαν είσοδο την μέθοδο που θα χρησιμοποιηθεί για την πρόβλεψη, δηλαδή εάν θα χρησιμοποιηθούν και οι 4x4 προβλέψεις ή όχι. Άλλες είσοδοι είναι οι στήλες αριστερά και σειρές οι οποίες χρειάζονται για την παραγωγή των προβλέψεων των προς επεξεργασία block, ξεχωριστά για το luma (y_left, y_top) και ξεχωριστά για το chroma (u_left, uv_top). Τέλος σαν είσοδοι λαμβάνονται οι συνιστώσες x,y του γενικού loop που κάνει την επεξεργασία της εικόνας, η διεύθυνση της αποθήκευσης της εξόδου των προβλέψεων(dst), ένα σήμα ενεργοποίησης του συστήματος(enable) και το ρολόι του συστήματος (clk).

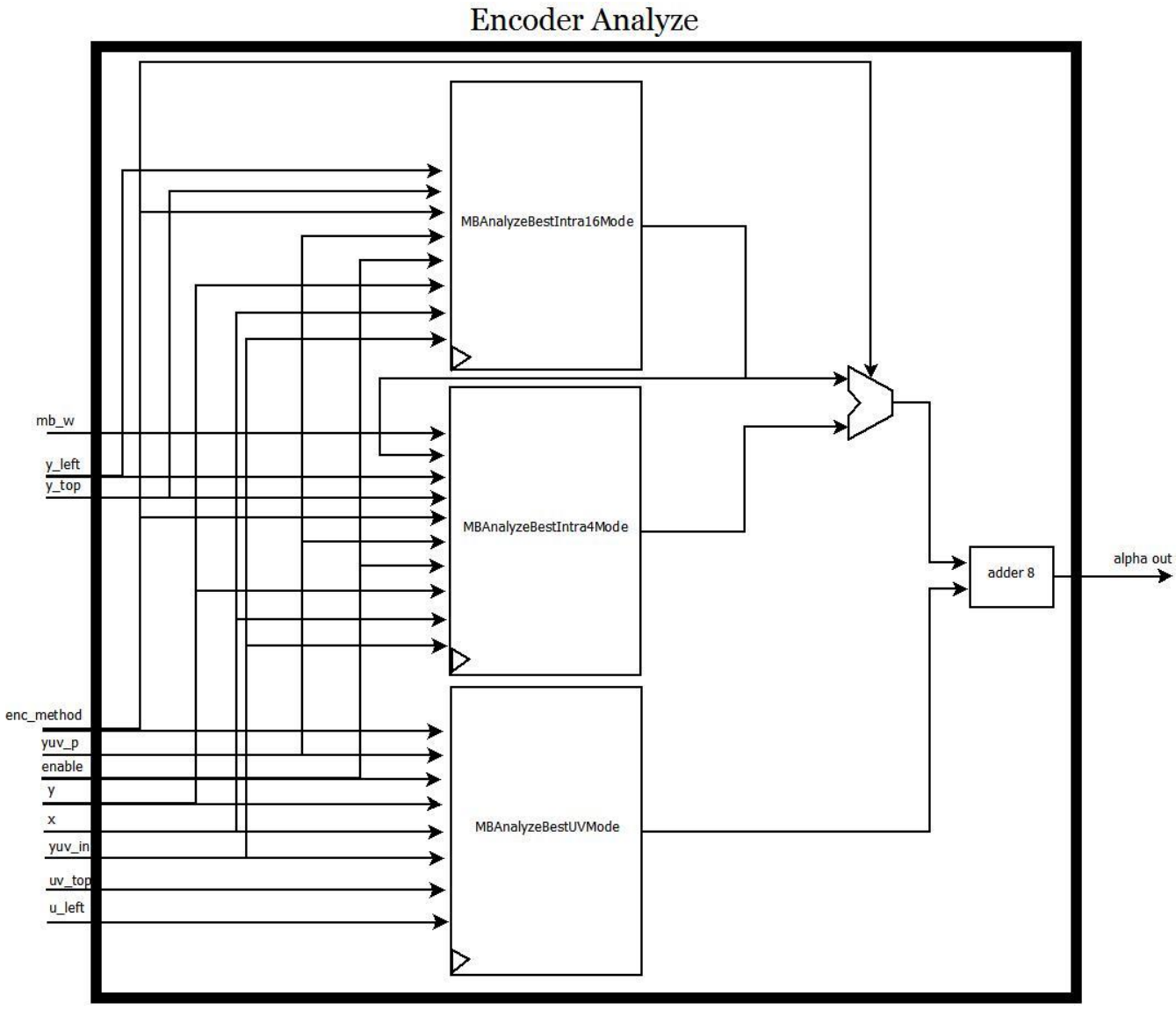
Όλες αυτές οι είσοδοι ανάλογα με την χρησιμότητά τους, για πρόβλεψη block

16x16, 4x4 ή UV, μπαίνουν στο κατάλληλο module για περεταίρω επεξεργασία. Για μέγεθος 16x16 υπεύθυνο είναι το τμήμα MBAnalyzeBestIntra16Mode , για 4x4 το MBAnalyzeBestIntra4Mode και για τις συνιστώσες χρωματικότητας το MBAnalyzeBestUVMode.

Η έξοδος αποτελείται από μια μόνο τιμή η οποία ονομάζεται alpha, η οποία εξαρτάται από τον βαθμό τον μη μηδενικών στοιχείων σε μεγάλες συχνότητες. Όσο περισσότερα στοιχεία σε μεγάλες συχνότητες τόσο καλύτερη η πρόβλεψη η οποία έχει γίνει. Βγαίνει από τον κάθε τύπο πρόβλεψης μια τιμή alpha, 2 από τις οποίες χρησιμοποιούνται για να προσδιοριστεί εάν θα χρησιμοποιηθούν οι 4x4 προβλέψεις ή οι 16x16 , ανάλογα με την ζητούμενη μέθοδο συμπίεσης, και επιλέγεται η κατάλληλη τιμή από τις 2 για να αθροιστεί με τη τιμή alpha του module που κάνει τις προβλέψεις για το chroma και να μας βγάλει την τελική έξοδο. Για αυτό το λόγο χρησιμοποιείται ο πολυπλέκτης στις εξόδους των δύο module που χρησιμοποιούνται για την πρόβλεψη luma ο οποίος ακολουθείται από έναν αθροιστή για την πρόσθεση με το alpha του chroma.

Επιπλέον έξοδοι για μελλοντική σχεδίαση μπορούν να είναι οι τιμές των προβλέψεων από το κάθε module.

Το ακόλουθο διάγραμμα δείχνει την δομή της συνάρτησης:



Εικόνα 16 Σχηματικό διάγραμμα υλοποίησης Encoder Analyze

5.3.2 MBAnalyzeBestIntra16Mode

Το τμήμα MBAnalyzeBestIntra16Mode είναι το κομμάτι το οποίο υπολογίζει την αποτελεσματικότητα κάθε μεθόδου πρόβλεψης για block 16x16. Αντιστοιχεί στην συνάρτηση MBAnalyzeBestIntra16Mode του αρχείου analysis.c του αρχικού κώδικα.

Παίρνει σαν είσοδο το 16x16 luma block μέσω της εισόδου yuv_in μεγέθους 2048 bit , την μέθοδο πρόβλεψης μέσω της εισόδου enc_method, τις τιμές των pixel της στήλης αριστερά και της γραμμής πάνω από το block μέσω των εισόδων y_top και y_left την διεύθυνση του αποτελέσματος από την είσοδο yuv_p καθώς και σήμα ενεργοποίησης και το ρολόι του συστήματος.

Σαν εξόδους βγάζει το καλύτερο alpha μέσω της εξόδου best_alpha καθώς και την αποτελεσματικότερη μέθοδο πρόβλεψης, από τις 4 που χρησιμοποιούνται, μέσω της εξόδου best_mode.

Αποτελείται από τα τμήματα Intra16Preds, το οποίο υπολογίζει τις προβλέψεις τιμών για το block, το τμήμα CollectHistogram το οποίο υπολογίζει το ιστόγραμμα του μετασχηματισμού των διαφορών των τιμών των pixel, σε σχέση με την πρόβλεψη, το τμήμα PredRegister το οποίο αποτελεί έναν καταχωρητή ο οποίος αποθηκεύει τις τιμές για κάθε τρόπο πρόβλεψης, και 3 συγκριτές(comparators) για την επιλογή του καλύτερου τρόπου πρόβλεψης για το block. Κάθε συγκριτής αφαιρεί τις 2 τιμές και στην έξοδό του βγάζει άσσο εάν η πρώτη τιμή είναι μεγαλύτερη από την δεύτερη. Ο κάθε συγκριτής στην συνέχεια τροφοδοτεί την είσοδο επιλογής ενός πολυπλέκτη για να πάρουμε στην έξοδο του πολυπλέκτη την μεγαλύτερη τιμή για το alpha.

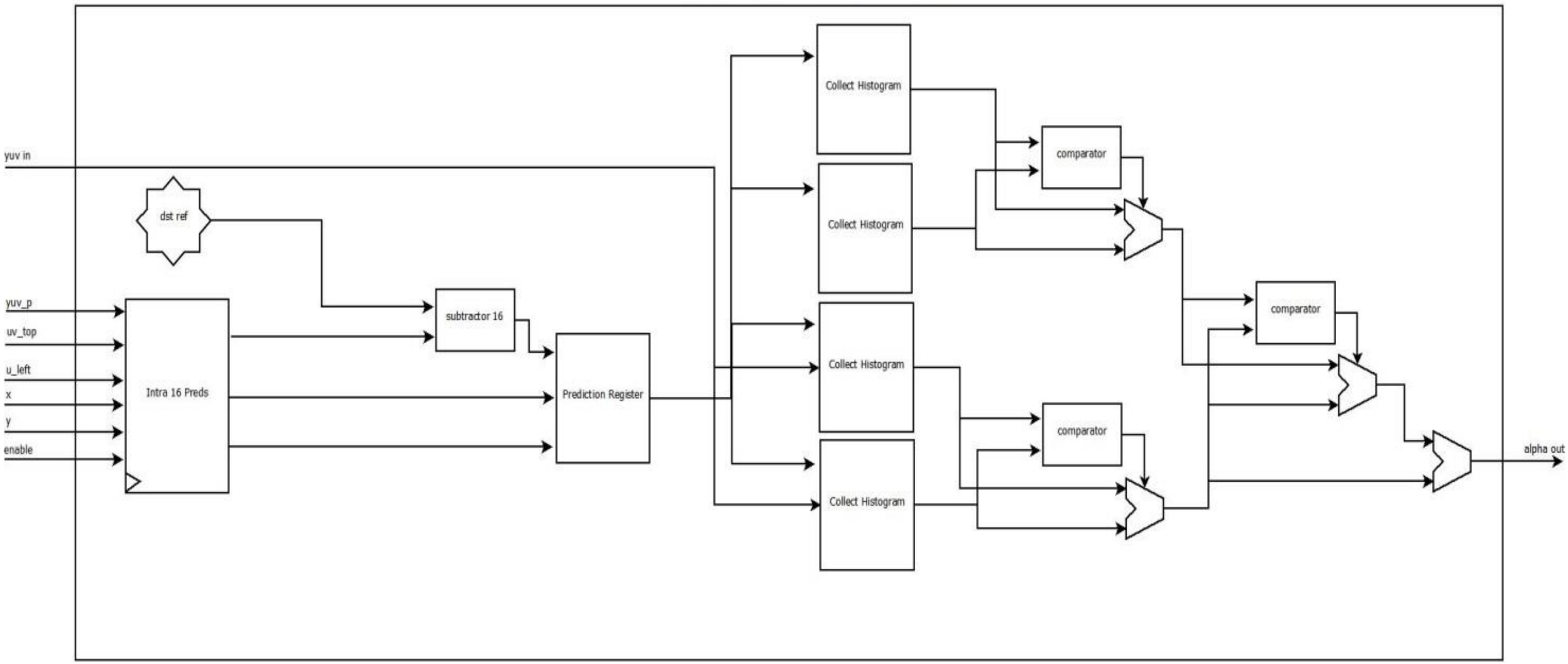
Επιπλέον έχει μία λογική (dst_ref) η οποία αποθηκεύει την αρχική διεύθυνση για να γίνει η αφαίρεση μέσω του subtractor_16(αφαιρετής μεγέθους 16 bit) και να έχουμε τις διευθύνσεις αναφοράς για την αποθήκευση των τιμών των προβλέψεων στον καταχωρητή.

Όλες οι είσοδοι εκτός από το block της αρχικής εικόνας εισέρχονται στο κομμάτι που κάνει τις προβλέψεις , και κατόπιν αυτές οι προβλέψεις αποθηκεύονται 4 ανά κύκλο στον PredRegister. Έπειτα περνάν μαζί με την αρχική εικόνα από το CollectHistogram όπου βγαίνουν σαν έξοδοι τα alpha και χρησιμοποιούνται οι συγκριτές για να δούμε την καλύτερη μέθοδο.

Χρησιμοποιήθηκε αυτή η λογική για την εκμετάλλευση της παραλληλίας στα αποτελέσματα, λόγω των τεσσάρων διαφορετικών τρόπων πρόβλεψης που θα μπορούν να λειτουργούν παράλληλα, και την γρηγορότερη χρονικά λύση , χάνοντας όμως σε χώρο υλοποίησης. Η δομή της συνάρτησης παρουσιάζεται στην εικόνα 17.

Το κομμάτι PredictionRegister είναι μια σειρά 1024 διευθυνσιοδοτημένων καταχωρητών για την αποθήκευση των τιμών των προβλέψεων. Παίρνει ως είσοδο τις διευθύνσεις και τα δεδομένα, 4 ανά φορά και τα αποθηκεύει στην κατάλληλη θέση. Η διεύθυνση κανονικοποιείται στον προηγούμενο αφαιρέτη έτσι ώστε να έχει μία τιμή από 0 έως 1023 το οποίο είναι η μεγαλύτερη δυνατή τιμή μετά την κανονικοποίηση, λόγω του ότι από το module των προβλέψεων η πρώτη από την τελευταία τιμή έχουν αυτή τη διαφορά. Οι διευθύνσεις 0-255 ανήκουν στα δεδομένα των προβλέψεων DC, 256-511 στις προβλέψεις TM, 512-767 στις κάθετες προβλέψεις και οι τιμές 768-1023 στις οριζόντιες προβλέψεις.

Η έξοδός του είναι οι 1024 τιμές των προβλέψεων, οι οποίες πάνε ανά 256 σαν είσοδοι στα τμήματα CollectHistogram αντίστοιχα. Οι τιμές για τις DC προβλέψεις στο πρώτο, για τις TM στο δευτερο, για τις κάθετες στο τρίτο και για τις οριζόντιες στο τέταρτο αντίγραφο της CollectHistogram.



Εικόνα 17 Σχηματικό διάγραμμα υλοποίησης συνάρτησης MBAnalyzeBestIntra16Mode

5.3.3 Intra16Preds

Η υλοποίηση της συνάρτησης Intra16Preds στον αρχικό κώδικα γίνεται στο αρχείο enc.c. Υλοποιεί τον μηχανισμό πρόβλεψης για block φωτεινότητας μεγέθους 16x16.

Παίρνει σαν είσοδο τα pixel στην σειρά επάνω (top) και την στήλη αριστερά(left) από το block που είναι να προβλεφθεί καθώς και την διεύθυνση του (dst). Ακόμα σαν είσοδο λαμβάνει και το σήμα ενεργοποίησης enable και το ρολόι του συστήματος clk.

Οι έξοδοί του είναι οι τιμές που προβλέφθηκαν (value_out), η διεύθυνσή τους(dst_out) και ένα σήμα ενεργοποίησης εγγραφής για τον καταχωρητή προβλέψεων(wr_en).

Η υλοποίηση του κομματιού περιλαμβάνει τους μηχανισμούς που υλοποιούν τις 4 προβλέψεις, 3 αθροιστές για την σωστή διευθυνσιοδότηση της κάθε τιμής πρόβλεψης και μια FSM.

Για την πρόβλεψη DC υπολογίζεται ο μέσος όρος των εισόδων top και left εάν υπάρχουν και οι δύο και γεμίζει όλο το block πρόβλεψης με αυτές τις τιμές. Εάν δεν υπάρχει ένα από τα δύο τότε υπολογίζεται ο μέσος όρος εκείνου που υπάρχει και τοποθετείται ενώ αν πρόκειται για το πρώτο πάνω αριστερά block της εικόνας τότε η DC πρόβλεψη είναι 128, η οποία είναι η μέση τιμή του εύρους των τιμών κάθε pixel.

Για την πρόβλεψη TM υπολογίζονται σύμφωνα με τον τύπο (3.1) οι τιμές του κάθε pixel και ορίζεται η πρόβλεψη.

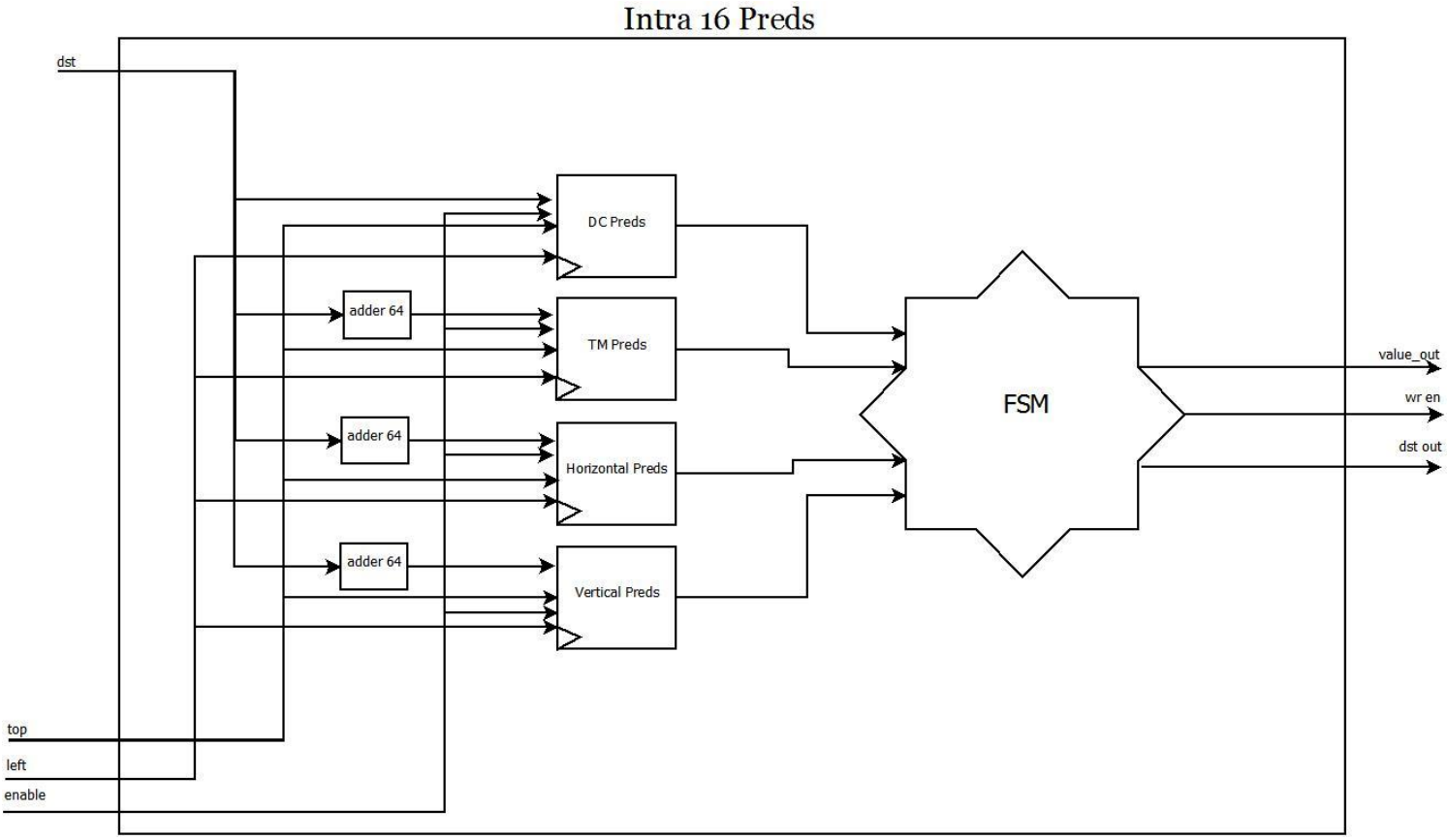
Στην οριζόντια πρόβλεψη η έξοδος κάθε σειράς είναι ίση με την τιμή της στήλη αριστερά εκτός εάν δεν υπάρχει οπότε όλα γεμίζουν με την τιμή 129 (έτσι το θέτει ο αλγόριθμος).

Στην κάθετη κάθε σειρά της εξόδου έχει τις τιμές την σειράς πάνω από το προς πρόβλεψη block. Εάν πρόκειται για block της πρώτης σειράς της εικόνας , δηλαδή δεν γίνεται να έχει από πάνω σειρά, θέτεται η έξοδος ίση με 127.

Κάθε μέθοδος πρόβλεψης έχει υλοποιηθεί χρησιμοποιώντας μία μηχανή πεπερασμένων καταστάσεων αντί για shift register, λόγω λανθασμένης εκτίμησης. Σε κάθε κύκλο οποιαδήποτε μέθοδος πρόβλεψης βγάζει στην έξοδό της μία τιμή πρόβλεψης και την αντίστοιχη διεύθυνση στην οποία θα πρέπει να αποθηκευτεί.

Η FSM η οποία υπάρχει είναι υπεύθυνη για τον ορισμό του κομματιού ως busy ή όχι καθώς και για την έξοδο σε κάθε κύκλο ρολογιού μίας τιμής από την κάθε μέθοδο πρόβλεψης

Το σχηματικό διάγραμμα της συνάρτησης Intra16Preds παρουσιάζεται την εικόνα 18.



Εικόνα 18 Διάγραμμα υλοποίησης module Intra16Preds

5.3.4 MBAnalyzeBestUVMode

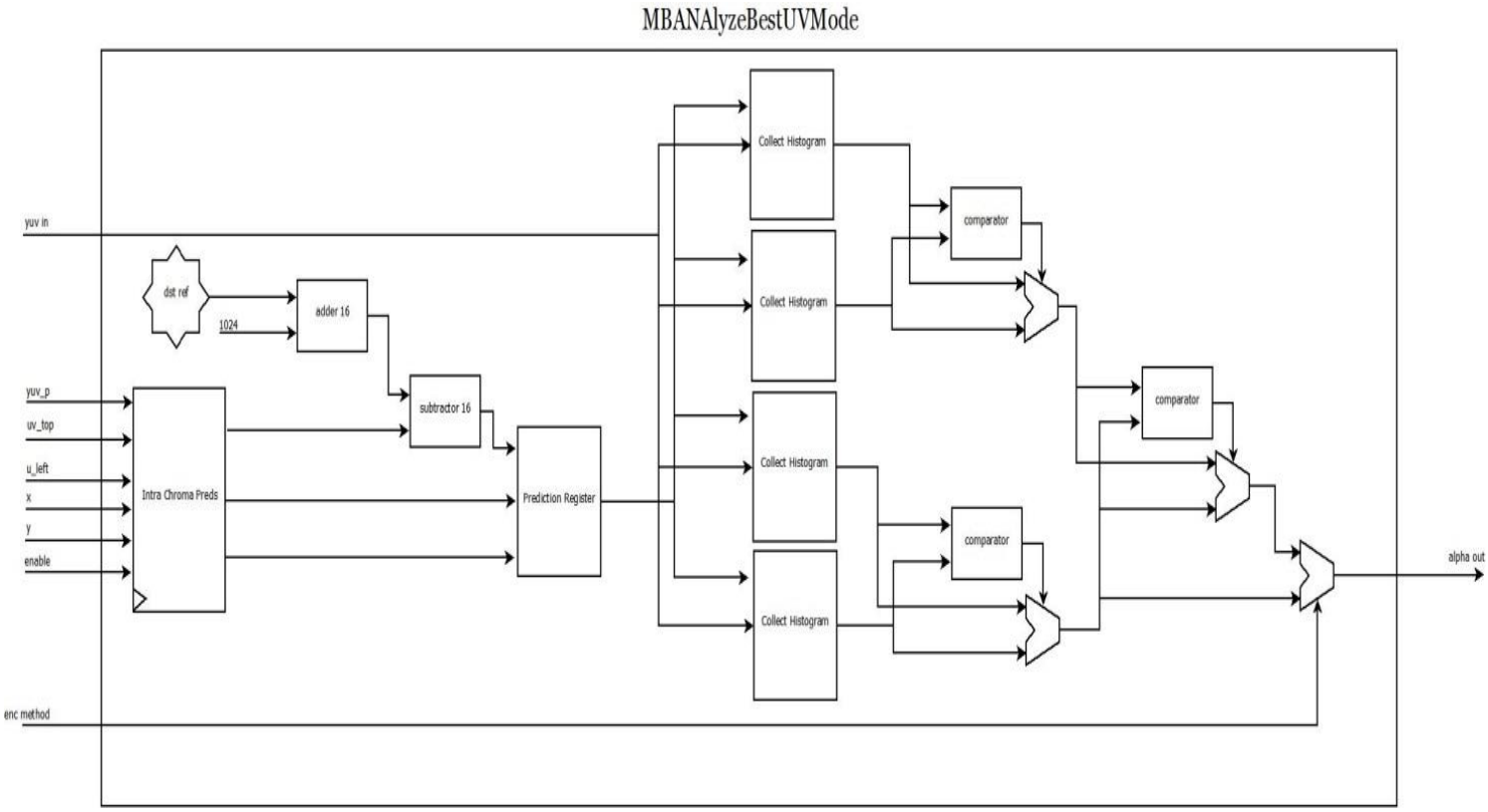
Το module MBAnalyzeBestUVMode υλοποιεί την συνάρτηση MBAnalyzeBestUVMode του αρχείου analysis.c του αρχικού κώδικα.

Αποτελεί την συνάρτηση η οποία υπολογίζει την βέλτιστη πρόβλεψη για τις συνιστώσες chroma του block που επεξεργαζόμαστε.

Έχει σαν εισόδους ολόκληρο το αρχικό block (yuv_in), την στήλη αριστερά και την σειρά πάνω από τις συνιστώσες(u_left,uv_top), τις συνιστώσες x,y του γενικού loop που κάνει την επεξεργασία της εικόνας, την διεύθυνση που θα αποθηκευτούν οι προβλέψεις(yuv_p), ένα σήμα ενεργοποίησης του module, η μέθοδος που θα χρησιμοποιηθεί για την πρόβλεψη (enc_method). Έξοδοι είναι η τιμή alpha και η καλύτερη μέθοδος πρόβλεψης.

Το κομμάτι αυτό έχει παρόμοια διαδικασία με την υλοποίηση του MBAnalyzeBestIntra16Mode, με μόνες διαφορές στο κομμάτι των προβλέψεων (κουτί IntraChromaPreds), όπου ακολουθείται διαφορετική διαδικασία η οποία θα παρουσιαστεί στη συνέχεια, και έναν αθροιστή για την διεύθυνση αναφοράς καθώς η διεύθυνση αποθήκευσης της πρώτης τιμής συνιστώσας chroma είναι στη θέση 256.

Η αρχιτεκτονική του module παρουσιάζεται στην εικόνα 19.



Εικόνα 19 Διάγραμμα υλοποίησης του module MBAnalyzeBestUVMode

5.3.5 IntraChromaPreds

Οι προβλέψεις για το chroma γίνονται ξεχωριστά για το Cb και το Cr. Για αυτό το λόγο στην υλοποίηση χρησιμοποιήσαμε 2 παράλληλες μεθόδους για τις προβλέψεις, μία για κάθε συνιστώσα του chroma.

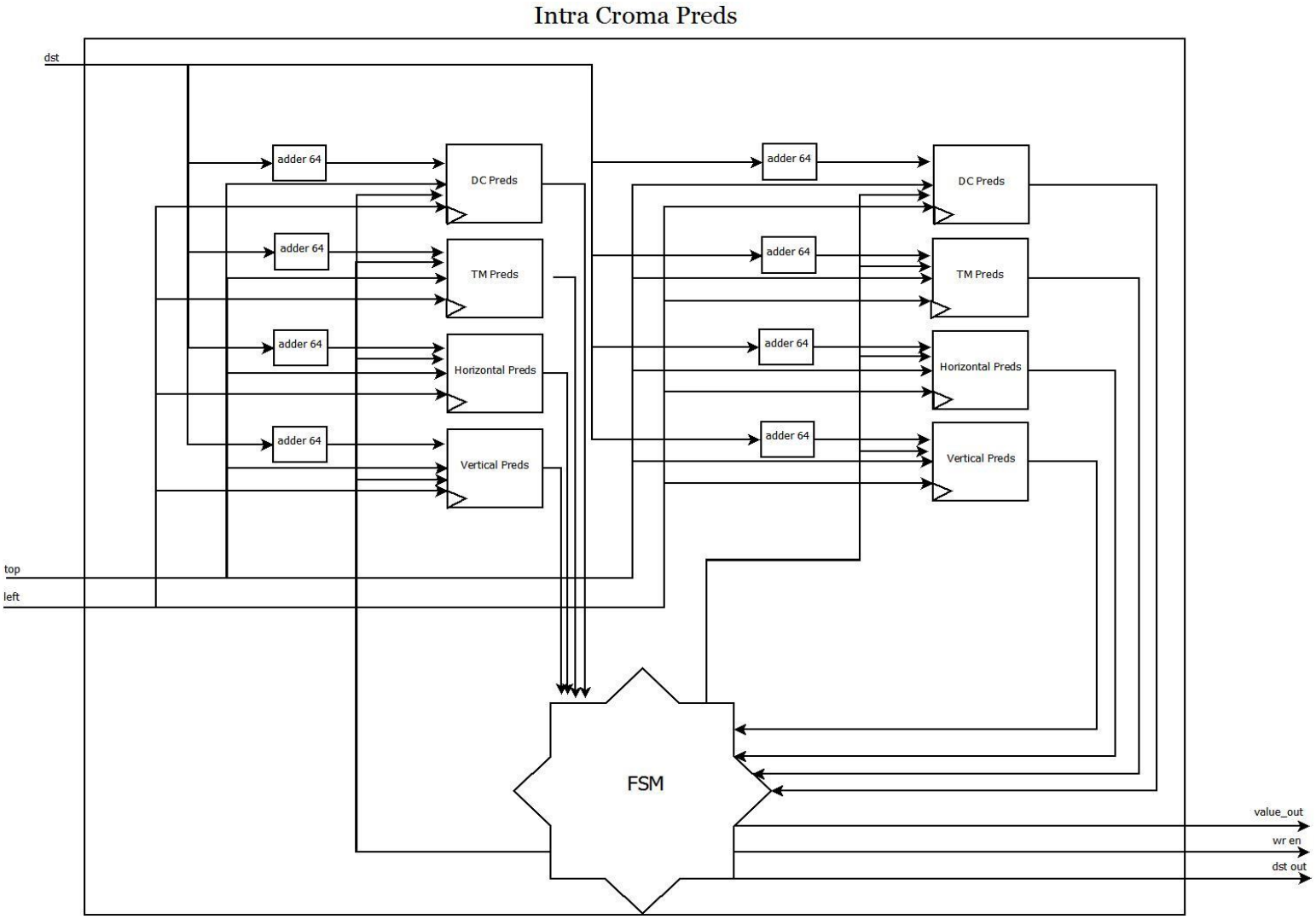
Ειδικότερα, η σχεδίασή μας περιέχει δύο αντίγραφα από κάθε μέθοδο πρόβλεψης (DC, TM, Horizontal, Vertical) με τους αντίστοιχους αθροιστές για την δημιουργία των διευθύνσεων αποθήκευσής τους. Στην μία σειρά αντιγράφων δημιουργούνται οι προβλέψεις για τις συνιστώσες Cb και στην δεύτερη οι προβλέψεις για την συνιστώσα Cr.

Σαν είσοδοι στο IntraChromaPreds είναι η σειρά πάνω και η στήλη αριστερά από το προς πρόβλεψη block (u_top, uv_left), η διεύθυνση που θα αποθηκευτεί το αποτέλεσμα(dst), το ρολόι του συστήματος(clk) και ένα σήμα ενεργοποίησης(enable). Έξοδοι είναι οι τιμές των προβλέψεων (value_out), η διεύθυνση αποθήκευσης των προβλέψεων(dst_out) καθώς και ένα σήμα ενεργοποίησης εγγραφής για τις προβλέψεις(wr_en).

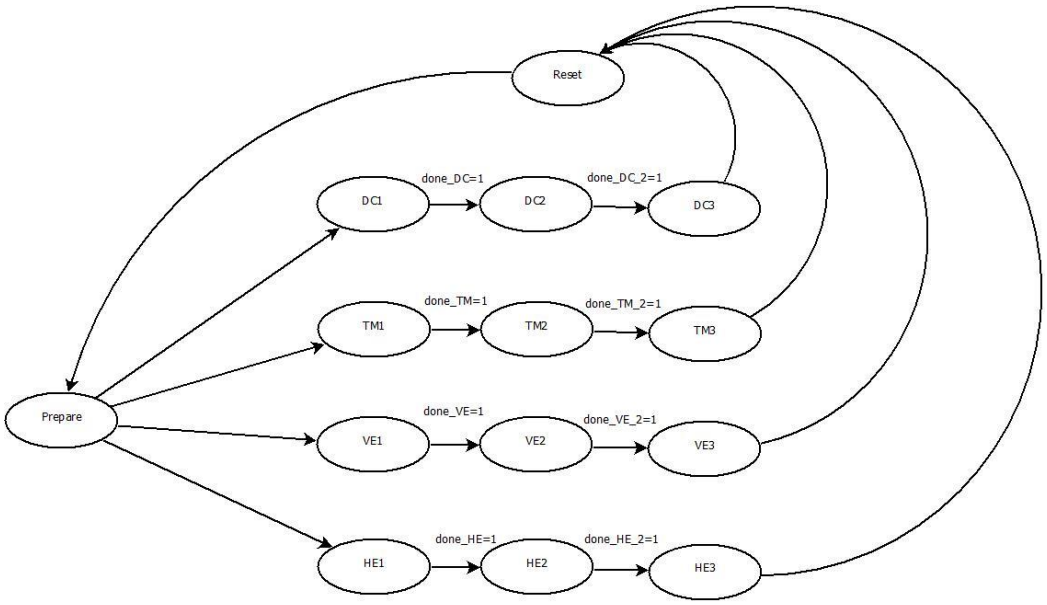
Ακόμα το περιέχεται μια FSM η οποία υλοποιεί το σύστημα ελέγχου των μηχανισμών προβλέψεων και των εξόδων του συστήματος. Περιέχει τέσσερις αυτόνομες FSM με κοινή κατάσταση reset, η κάθε μία από τις οποίες ελέγχει έναν τρόπο πρόβλεψης. Η πρώτη κατάσταση πρόβλεψης(DC_1, TM_1, VE_1, HE_1) επιτρέπει να δουλεύουν οι προβλέψεις για τις συνιστώσες Cb και μόλις λάβει σήμα ότι τελείωσε η πρόβλεψη, μεταβαίνει στην δεύτερη κατάσταση πρόβλεψης(DC_2, TM_2, VE_2, HE_2) όπου γίνονται οι προβλέψεις για τις συνιστώσες Cr. Η τρίτη κατάσταση(DC_3, TM_3, VE_3, HE_3) είναι κατάσταση αναμονής μέχρι να τελειώσουν όλες οι προβλέψεις και για τις δύο συνιστώσες και να γυρίσει στην κατάσταση reset. Υπάρχει και η κατάσταση Prepare οποία είναι μεταβατική μεταξύ

της κατάστασης reset και της πρώτης κατάστασης πρόβλεψης και χρησιμοποιείται για να έχουν οι μηχανισμοί προβλέψεων τα κατάλληλα σήματα κατά την εκκίνηση.

Στην εικόνα 20 παρουσιάζεται η σχεδίαση και στην εικόνα 21 παρουσιάζεται η FSM ελέγχου.



Εικόνα 20 Διάγραμμα υλοποίησης του module IntraChromaPreds



Εικόνα 21 Διάγραμμα ροής FSM του InrtraChromaPreds

5.3.6 MBAnalyzeBestIntra4Mode

Η συνάρτηση MBAnalyzeBestIntra4Mode αποτελεί υλοποίηση της συνάρτησης με το ίδιο όνομα του αρχείου analysis.c του αρχικού κώδικα. Η συνάρτηση αυτή παίρνει ένα block 16x16 το χωρίζει σε 16 block μεγέθους 4x4 τα οποία κατόπιν αφού γίνουν για το κάθε ένα οι προβλέψεις και με τους 10 τύπους προβλέψεων , υπολογίζεται το ιστόγραμμα του μετασχηματισμού των διαφορών τους.

Το VP8IteratorStart μαζί με τα VP8IteratorRotate δημιουργούν τα κατάλληλα σήματα τα οποία θα αποτελέσουν είσοδο στο Intra4Loop. Το Intra4Loop αποτελεί το κύριο κομμάτι επεξεργασίας στο οποίο γίνονται οι προβλέψεις και ο μετασχηματισμός των διαφορών για τις προβλέψεις των block 4x4.Το IteratorStart αρχικοποιεί τον χωρισμό σε block.Με είσοδο τα top και left δημιουργεί τον ακόλουθο πίνακα:

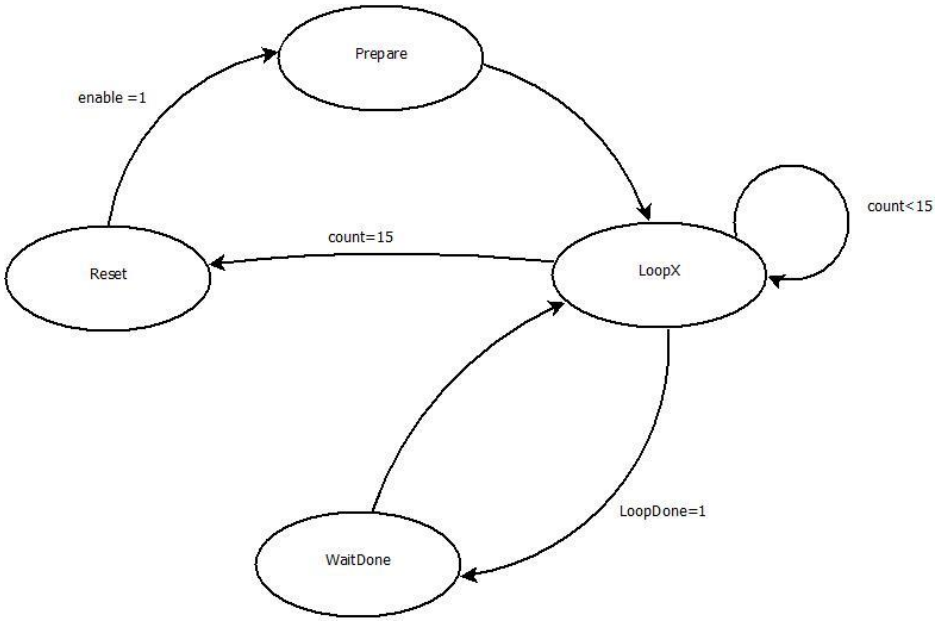
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
15				19				23				27				31				
14				18				22				26				30				
13				17				21				25				29				
12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28				
11				15				19				23				27				
10				14				18				22				26				
9				13				17				21				25				
8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24				
7				11				15				19				23				
6				10				14				18				22				
5				9				13				17				21				
4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20				
3				7				11				15				19				
2				6				10				14				18				
1				5				9				13				17				
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16				

Ο πίνακας αυτός έχεις τις ακραίες τιμές για κάθε 4x4 block. Το IteratorRotate ξεκινώντας από κάτω αριστερά «περιστρέφει» τον πίνακα έτσι ώστε μετά από κάθε IteratorRotate να έχει στην έξοδο τις ξεχωριστές τιμές (boundary values) οι οποίες χρειάζονται για τις προβλέψεις καθώς και το αντίστοιχο κομμάτι της αρχικής εικόνας. Ουσιαστικά προετοιμάζει τις τιμές τριγύρω από το προς πρόβλεψη block , όπως έχουμε δει και στην εικόνα 5.

Το σύνολο των αθροιστών οι οποίοι έχουν δενδρική δομή για να επιτύχουμε την μικρότερη χρονική πολυπλοκότητα, δημιουργούν την τιμή alpha για τις 16 προβλέψεις. Το αποτέλεσμα αυτό κανονικοποιείται μέσω ενός πολυπλέκτη στις τιμές 0-255 (0 εάν είναι αρνητικό,255 εάν είναι μεγαλύτερο από 255, αλλιώς μένει ίδιο) και κατόπιν συγκρίνεται με την εισερχόμενη τιμή από το module Intra16Preds για το alpha και επιλέγεται ποια από τις δύο μεθόδους πρόβλεψης (4x4 ή 16x16) είναι η καλύτερη.

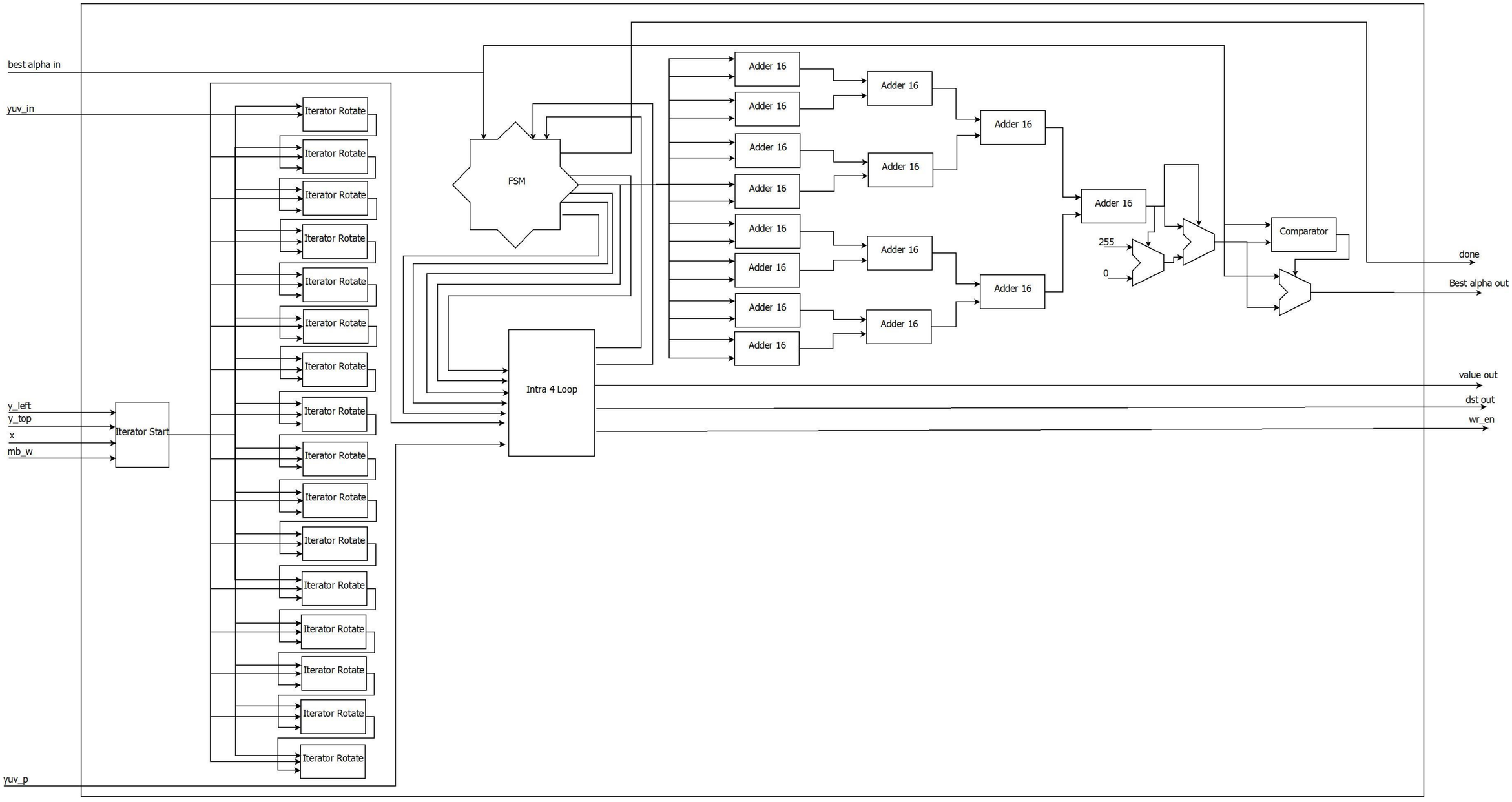
Υπάρχει ακόμα μια FSM η οποία υλοποιεί έναν counter από 0 έως 15 για τα 16 4x4 κομμάτια που περιέχει ένα 16x16 block. Η FSM έχει 4 καταστάσεις τις Reset, Prepare, LoopX, WaitDone. Η κατάσταση Reset μηδενίζει όλες τις τιμές εξόδου και όταν το σύστημα βρίσκεται σε αυτή τη κατάσταση είναι ανενεργό. Μόλις λάβει σήμα ενεργοποίησης μεταβαίνει στην κατάσταση Prepare η οποία θέτει τις κατάλληλες τιμές στα σήματα εξόδου και προετοιμάζει το σύστημα για την μετάδοση των τιμών στην έξοδο καθώς και στο module Intra4Loop. Στον επόμενο κύκλο μεταβαίνουμε στην κατάσταση LoopX που πρόκειται για την κατάσταση στην οποία έχουν τεθεί όλες οι τιμές και περιμένουμε να τελειώσει η πρόβλεψη ώστε να μεταβούμε στην κατάσταση WaitDone η οποία αυξάνει τον μετρητή κατά ένα ή στην κατάσταση Reset εάν έχουν γίνει και οι 16 προβλέψεις. Η κατάσταση WaitDone εκτός από την αύξηση του μετρητή αλλάζει και τις εισόδους για το module Intra4Loop.

Παραθέτονται σχηματικά διαγράμματα για την FSM(Εικόνα 22) καθώς και το διάγραμμα υλοποίησης του module(Εικόνα 23).



Εικόνα 22 Διάγραμμα FSM MBAnalyzeBestIntra4Mode

MBAnalyzeIntra4Mode



Εικόνα 23 Διάγραμμα υλοποίησης MBAnalyzeBestIntra4Mode

5.3.7 Intra4Loop

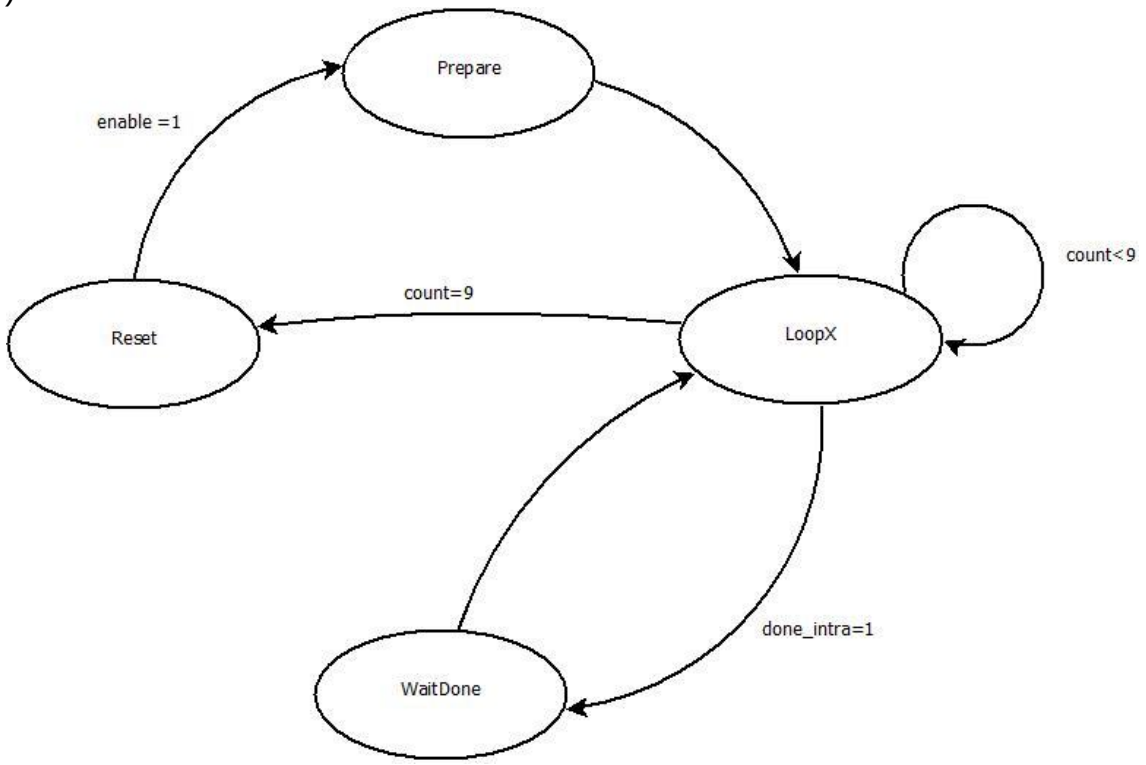
Η συνάρτηση Intra4Loop αποτελεί την υλοποίηση του μηχανισμού πρόβλεψης και μετασχηματισμού για block 4x4.

Αποτελείται από το Intra4Preds, το οποίο υλοποιεί τον μηχανισμό προβλέψεων για block 4x4 , τον PredRegister ο οποίος αποθηκεύει τις τιμές των προβλέψεων έτσι ώστε να τις διοχετεύσει στο CollectHistogram, και να υπολογιστεί ο μετασχηματισμός των διαφορών μεταξύ της αρχικής εικόνας και των προβλέψεων. Επίσης έχει μια δομή συγκριτών για την επιλογή του καλύτερου τρόπου πρόβλεψης με βάση το alpha που παράγεται. Τέλος έχει μια FSM η οποία υλοποιεί έναν counter για την διοχέτευση ξεχωριστά κάθε τρόπου πρόβλεψης στο CollectHistogram για την μείωση της χωρικής πολυπλοκότητας, χάνοντας σε παραλληλία και χρόνο, γιατί αλλιώς θα ήθελε πολύ μεγαλύτερο χώρο.

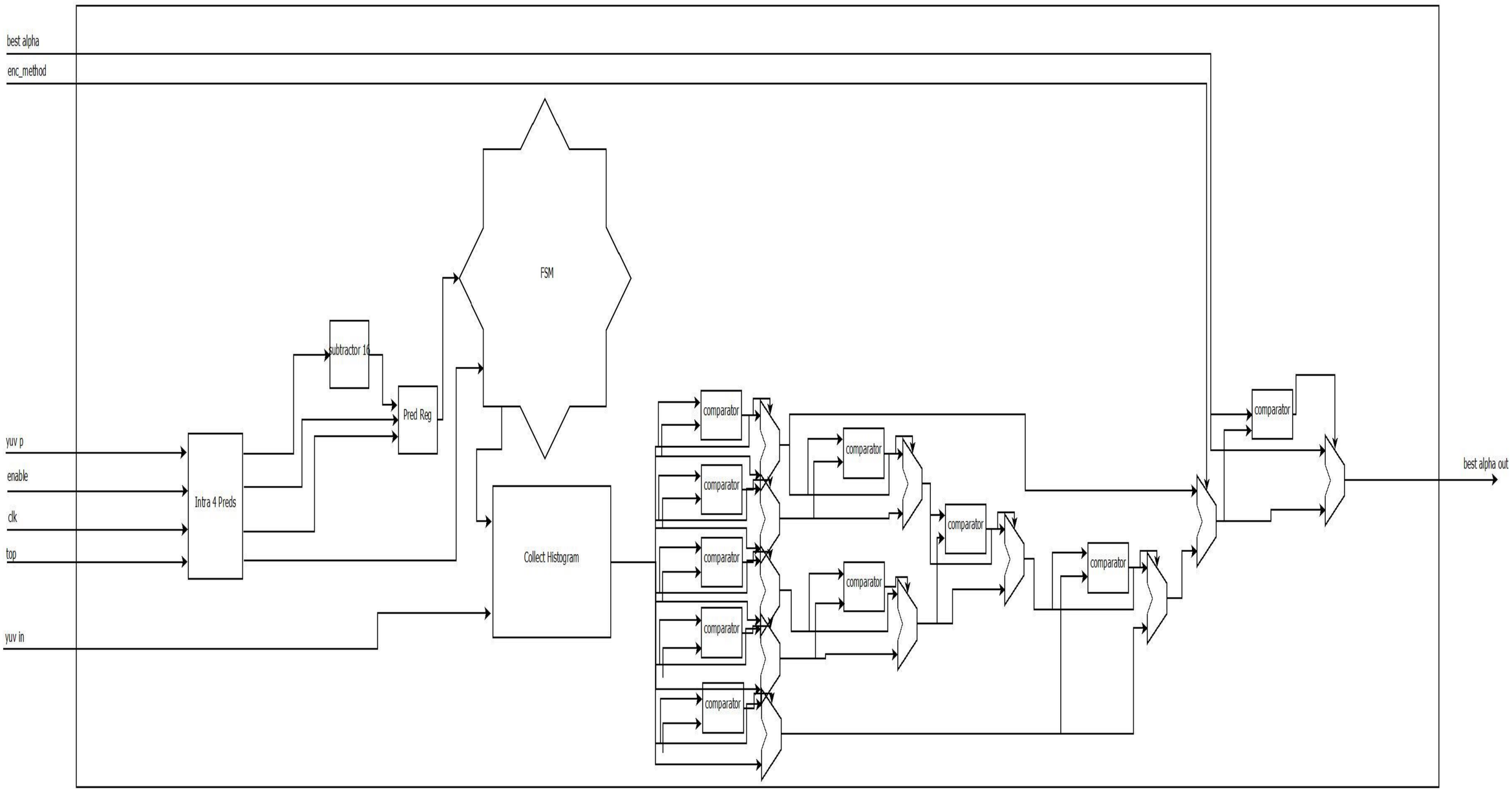
Η FSM έχει 4 καταστάσεις τις: Reset , Prepare, LoopX, WaitDone. Η κατάσταση Reset μηδενίζει όλες τις τιμές εξόδου προς το CollectHistogram, η κατάσταση Prepare θέτει τις τιμές για να ξεκινήσει η επεξεργασία, η LoopX περιμένει μέχρι να ολοκληρωθεί ο μετασχηματισμός και κατόπιν μεταβαίνει στην κατάσταση Reset αν δεν υπάρχουν ακόμα προβλέψεις που δεν έχει ολοκληρωθεί ο μετασχηματισμός τους ή στην κατάσταση WaitDone εάν υπάρχουν, η οποία αυξάνει τον μετρητή και θέτει τις κατάλληλες εισόδους για τον μετασχηματισμό.

Το σύστημα έχει σαν εισόδους το αντίστοιχο block της αρχικής εικόνας (yuv_in), την διεύθυνση αποθήκευσης των προβλέψεων (yuv_p) , την σειρά από επάνω(top), την μέθοδο κωδικοποίησης(enc_method), το alpha των μέχρις στιγμής προβλέψεων, το ρολόι του συστήματος καθώς και το σήμα ενεργοποίησης. Έξοδοι του είναι το alpha του μετασχηματισμού της καλύτερης πρόβλεψης.

Παρουσιάζονται το διάγραμμα υλοποίησης (Εικόνα 25) και η δομή της FSM(εικόνα 24).



Εικόνα 24 Διάγραμμα καταστάσεων FSM Intra4Loop



Εικόνα 25 Διάγραμμα υλοποίησης Intra4Loop

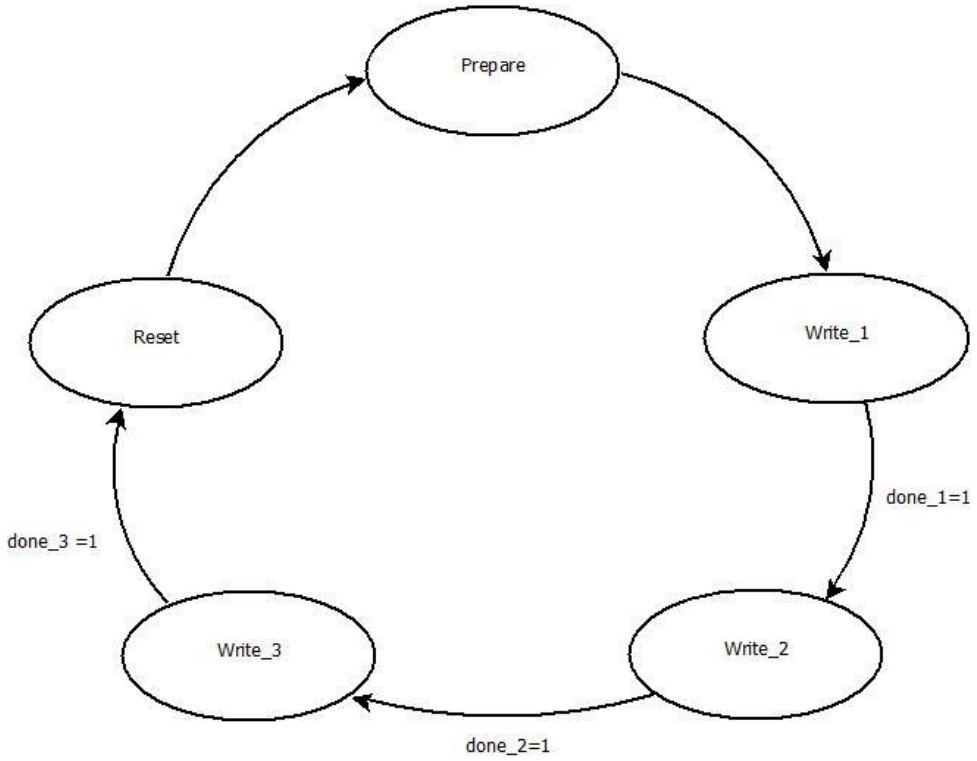
5.3.8 Intra4Preds

Η συνάρτηση Intra4Preds υλοποιεί τον μηχανισμό προβλέψεων για block 4x4. Έχει σαν εισόδους την αρχική διεύθυνση αποθήκευσης των προβλέψεων η οποία για κάθε τύπο πρόβλεψης εισέρχεται σε έναν αθροιστή για να λάβει την σωστή τιμή της για την αποθήκευση. Επιπλέον έχει σαν είσοδο την σειρά επάνω από το προς επεξεργασία block για την πρόβλεψη των αποτελεσμάτων και το σήμα ενεργοποίησης enable. Τέλος λαμβάνει και το ρολόι του συστήματος για την λειτουργία της.

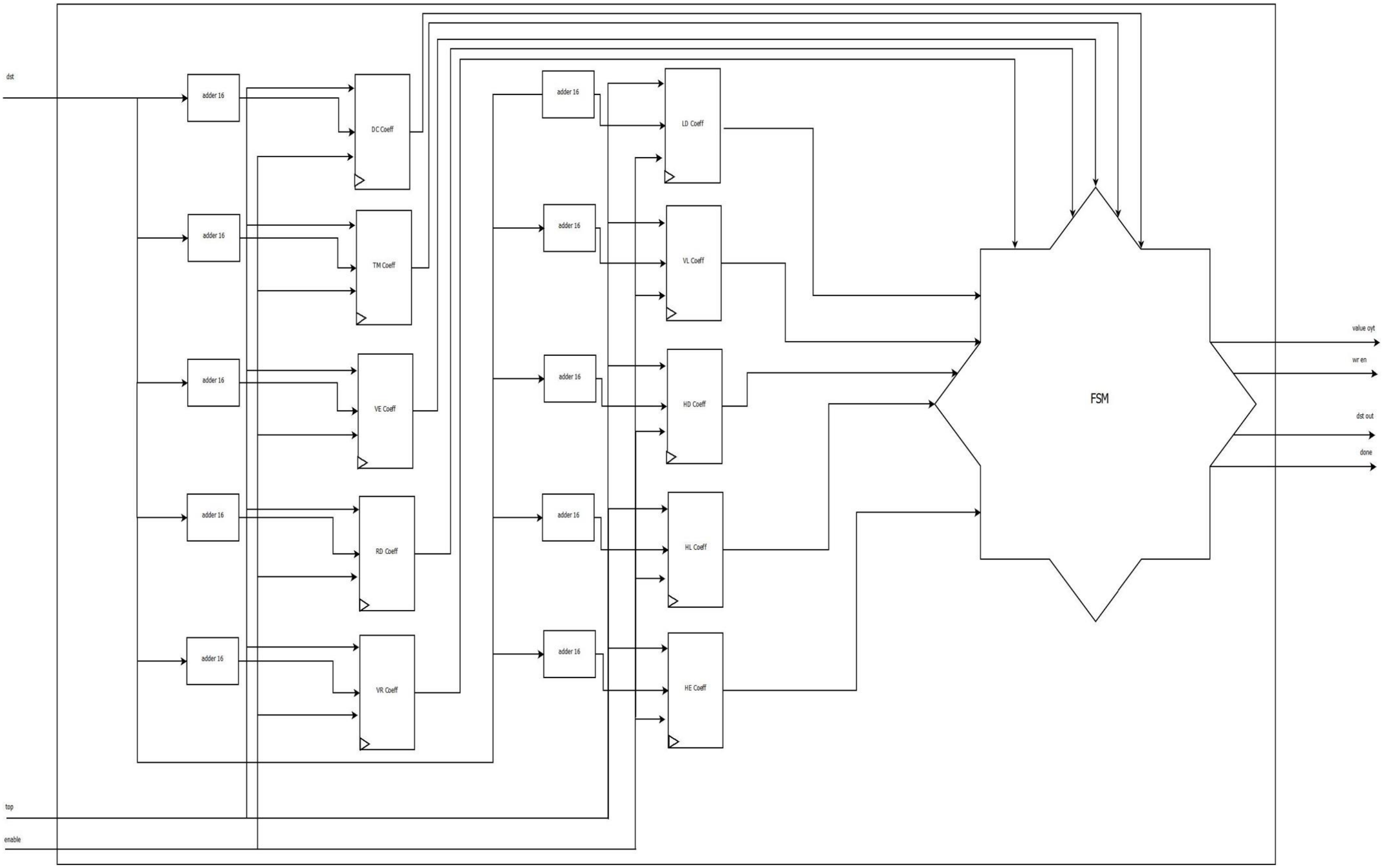
Έξοδοί της είναι το σήμα τέλους επεξεργασίας done, η τιμή των προβλέψεων(value_out), το σήμα ενεργοποίησης εγγραφής (wr_en) και την τελική διεύθυνση αποθήκευσης.

Αποτελείται από τους αθροιστές για την δημιουργία των διευθύνσεων εξόδου, τις υλοποιήσεις των 10 μοντέλων πρόβλεψης και μια FSM η οποία ελέγχει το σύστημα. Η λειτουργία της είναι να βγάζει σε κάθε κύκλο 4 τιμές πρόβλεψης. Ελέγχει και ενεργοποιεί κάθε φορά 4 μεθόδους πρόβλεψης για να μπορέσουν να αποθηκευτούν στον PredRegister ο οποίος έχει φτιαχτεί να λαμβάνει 4 τιμές ανά κύκλο. Έχει 5 καταστάσεις : Reset, Prepare, Write_1, Write_2, Write_3. Η κατάσταση Reset μηδενίζει όλα τα σήματα και μόλις λάβει σήμα ενεργοποίησης μεταβαίνει στην κατάσταση Prepare η οποία προετοιμάζει τα σήματα για να βγάλει τις σωστές εξόδους. Στην κατάσταση Write_1 βγαίνουν στην έξοδο οι τιμές των προβλέψεων DC, TM, Horizontal και Vertical οι οποίες μόλις τελειώσουν βγάζουν σήμα ολοκλήρωσης και γίνεται η μετάβαση στην κατάσταση Write_2. Σε αυτήν στην έξοδο του συστήματος εμφανίζονται οι τιμές για τις προβλέψεις VL, LD, VR και RD. όταν ολοκληρωθούν και αυτές οι προβλέψεις μέσω σήματος ολοκλήρωσης γίνεται μετάβαση στην κατάσταση Write_3 η οποία θέτει για την έξοδο τους εναπομείναντες 2 τρόπους πρόβλεψης.

Εικόνα 26 Διάγραμμα καταστάσεων FSM Intra4Preds



Intra4Preds



Εικόνα 27Διάγραμμα υλοποίησης Intra4Preds

5.3.9 CollectHistogram

Το module CollectHistogram υλοποιεί την συνάρτηση VP8CollectHistogram του αρχείου enc_sse2.c του αρχικού κώδικα.

Πρόκειται για την συνάρτηση η οποία υπολογίζει τον πίνακα των μετασχηματισμών των διαφορών της αρχικής εικόνας με τις προβλέψεις. Το module CollectHistogram αποτελεί το κύριο σκέλος της συνάρτησης και περιέχει το δέντρο αθροιστών που θα βγάλει το τελικό αποτέλεσμα αφού λάβει τις τιμές από το HistoLoop, το οποίο είναι και ο πυρήνας της συνάρτησης. Οι 8 πρώτοι αθροιστές λαμβάνουν μια σειρά από 1040 bits από το κάθε HistoLoop τα οποία αποτελούν την σειριακή αναπαράσταση του πίνακα του μετασχηματισμού των διαφορών και τα προσθέτουν στοιχείο προς στοιχείο.Τέλος έχει το block_select το οποίο επιλέγει ανάλογα με το πόσα block ζητείται να επεξεργαστούν (μέθοδος πρόβλεψης 4x4 ή 16x16) από τις εισόδους την κατάλληλη είσοδο του HistoLoop. Το VP8GetAlpha τέλος υπολογίζει από τον πίνακα των μετασχηματισμών την τιμή alpha σαν υλοποίηση του τύπου:

$$alpha = 10 * \frac{\sum_1^{65} histo[k] * k}{\sum_1^{65} k^2} - 5 \qquad (5.1)$$

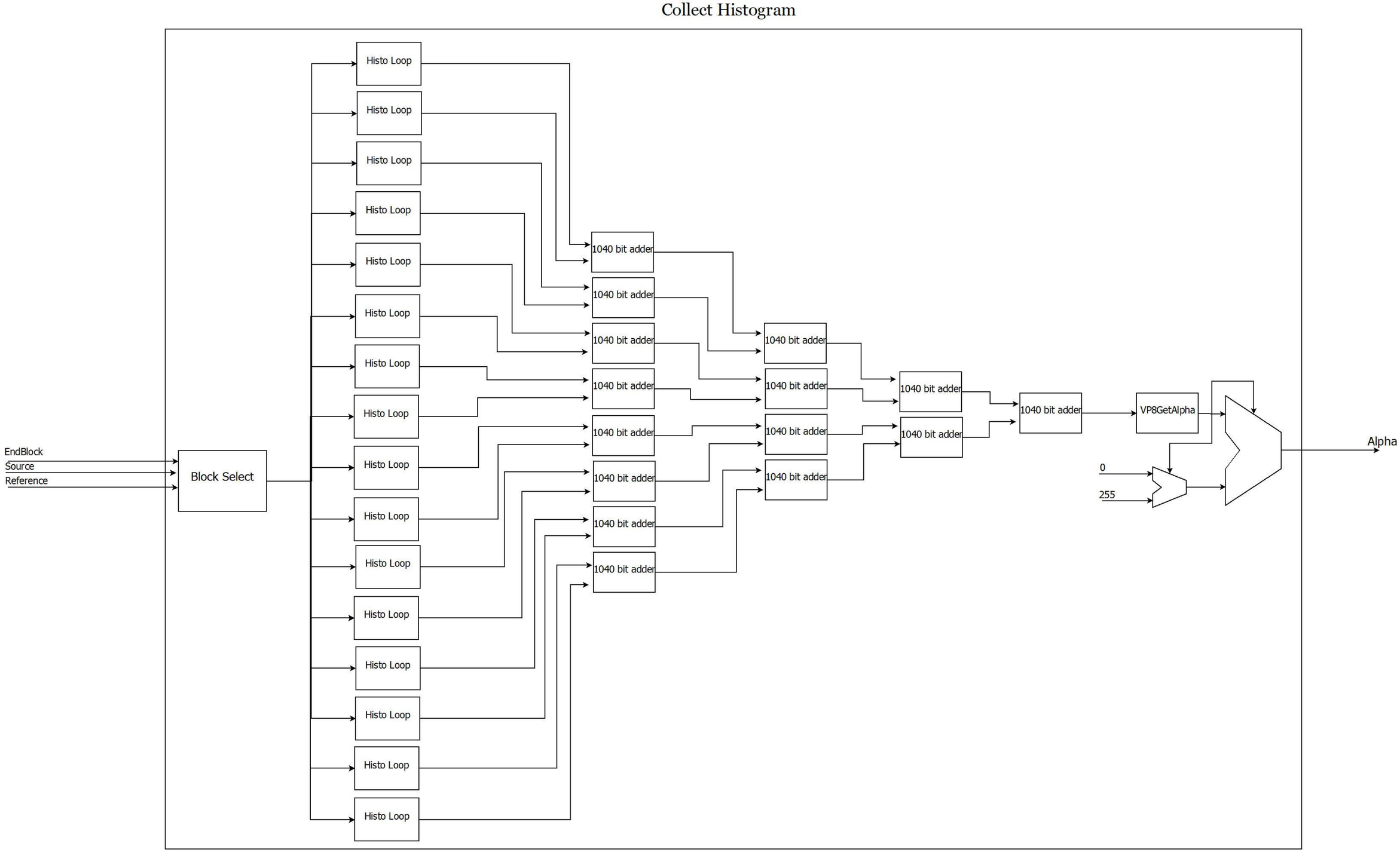
Σαν εισόδους το κύριο σύστημα λαμβάνει τον αρχικό block(Source), τις προβλέψεις(Ref), το εύρος των block που θα επεξεργαστούν (start_block,end_block) και βγάζει σαν έξοδο την βέλτιστη τιμή alpha κατόπιν κανονικοποίησης στις τιμές 0-255(0 εάν είναι αρνητικό,255 εάν είναι μεγαλύτερο από 255, η τιμή του εάν είναι ανάμεσα στο 0-255).Το διάγραμμα υλοποίησης παρουσιάζεται στην εικόνα 28.

5.3.9 HistoLoop

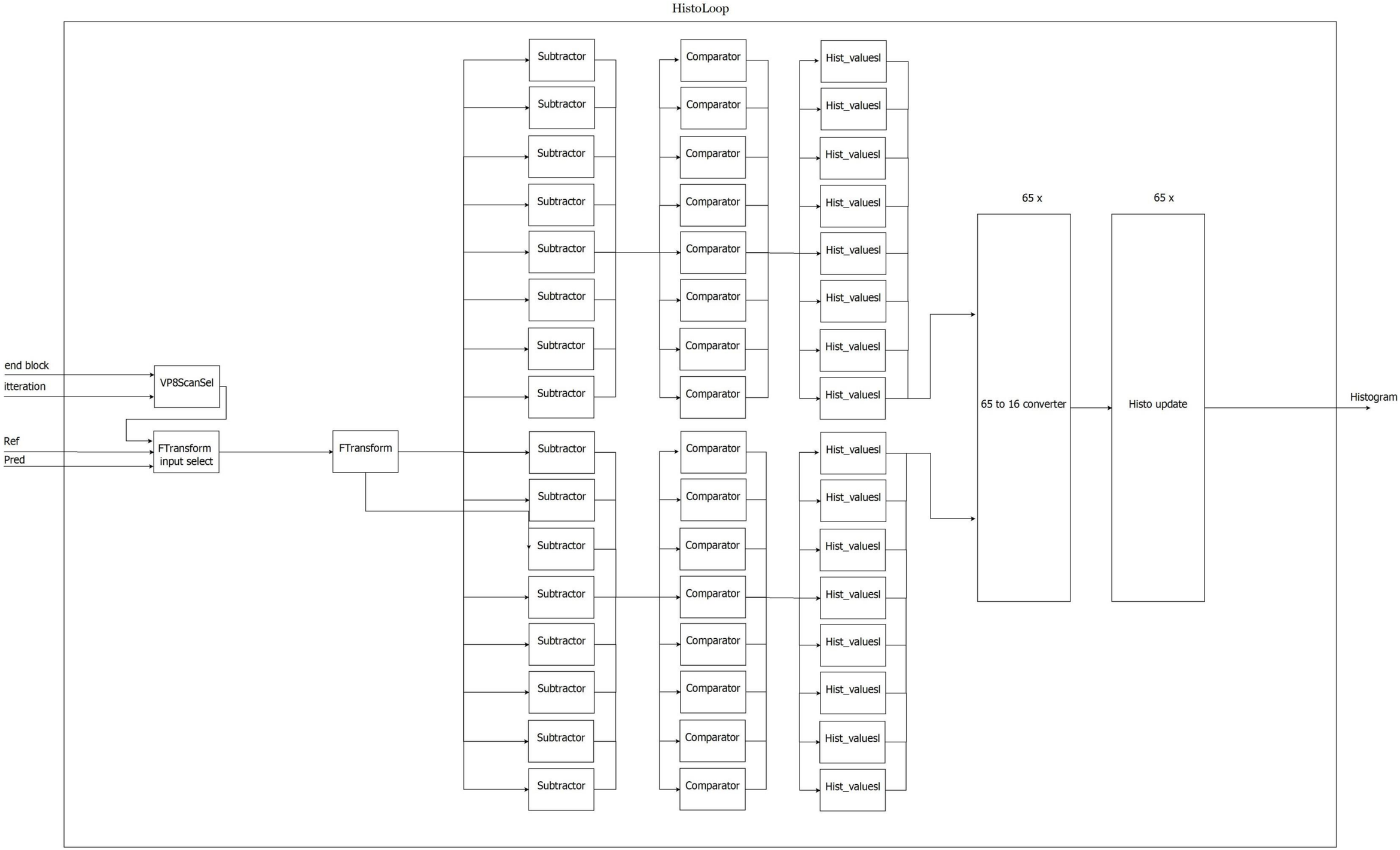
Το HistoLoop αποτελεί τον πυρήνα του μετασχηματισμού των διαφορών.

Παίρνει σαν είσοδο την τιμή του τελευταίου προς μετασχηματισμό block(end_block), καθώς και το ποια επανάληψη του αλγορίθμου είναι(iteration). Φυσικά παίρνει σαν εισόδους τις τιμές των pixel της εισόδου(block αναφοράς,Ref) και της πρόβλεψης(Pred). Στο VP8ScanSel γίνεται επιλογή της σχετικής θέσης του block που θα μετασχηματιστεί και αφού γίνει η επιλογή στο FTrans_input_sel επιλέγεται το κατάλληλο block βάση αυτής της σχετικής θέσης. Αυτό το κομμάτι της αρχικής εικόνας και της πρόβλεψης μπαίνουν σαν είσοδο στο FTransform στον οποίο υπολογίζονται οι διαφορές και γίνεται ο μετασχηματισμός τους. Η έξοδος του μετασχηματισμού μπαίνει σαν είσοδος σε μια σειρά απο αφαιρετές οι οποίοι με την κατάλληλη λογική υπολογίζουν την απόλυτη τιμή των τιμών που λαμβάνουν σαν είσοδο.Κατόπιν οι τιμές αυτές συγκρίνονται με την μέγιστη τιμή που επιτρέπεται να λάβει (στην περίπτωση μας 65) και μέσω συγκριτών επιλέγεται εάν θα επιλεγεί η τιμή της εξόδου ή η μέγιστη επιτρεπόμενη. Τα block hist_value λαμβάνουν μια τιμή μεγέθους 16 bit και έχουν σαν έξοδο μια τιμή μεγέθους 65 bit με άσσο στην θέση της τιμής εισόδου. Κατόπιν αυτές οι τιμές μεγέθους 65 bit εισέρχονται στα 65_to_16 converters τα οποία δημιουργούν από τις 16 τιμές μεγέθους 65 bit η κάθε μία, 65 τιμές μεγέθους 16 bit οι οποίες σε κάθε τιμή περιέχουν τόσους άσους όσα τα στοιχεία του μετασχηματισμού που έχουν αυτήν την τιμή. Τέλος το histo_update δημιουργεί τον σειριακό πίνακα του μετασχηματισμού που σε κάθε στοιχείο πλέον έχει την δυαδική απεικόνιση του αριθμού των στοιχείων του μετασχηματισμού που έχουν αυτή την τιμή.

Το διάγραμμα του HistLoop παρουσιάζεται στην εικόνα 29.



Εικόνα 28 Διάγραμμα υλοποίησης CollectHistogram



Εικόνα 29 Διάγραμμα υλοποίησης HistLoop

5.3.10 FTransform

Ο FTransform είναι ο μετασχηματισμός των διαφορών της αρχικής εικόνας σε σχέση με τις προβλέψεις του block προβλέψεων του αλγορίθμου. Υλοποιείται στην συνάρτηση enc_sse2.c του αρχικού κώδικα.

Ο μετασχηματισμός παίρνει σαν είσοδο τα δύο block (Ref – αρχική εικόνα , Pred– προβλέψεις) και βγάζει στην έξοδο τις τιμές του μετασχηματισμού για το block. Το πρώτο σκέλος το οποίο υλοποιείται στο ConvetAndSubtract υπολογίζει τις pixel-to-pixel διαφορές και μεγαλώνει το μέγεθος που χρειάζεται για την σωστότερη απεικόνιση θετικών και αρνητικών τιμών από 8 σε 16 bit αλλά και γιατί χρειάζεται να κάνουμε πράξεις με αριθμούς που δεν χωράνε σε 8 bit. Κατόπιν κάνει 2 μετασχηματισμούς έναν στο module pass_one και έναν στο pass_two. Ο τύπος των δύο μετασχηματισμών είναι:

Αν θέσουμε τις διαφορές ως diff όπου *diff(i)=src(i)-ref(i)* τότε

Pass One:

Για τις τιμές y=0,4,8,12:

$$tmp[y] = \sum_{i=y}^{y+3} [diff(i) \ll 3]$$

Για τις τιμές y=1,5,9,13:

$$tmp[y] = \{2217 * \left[\left(diff(y) - diff(y+1)\right) \ll 3\right] + 5352 * \left[\left(diff(y-1) - diff(y+2)\right) \ll 3\right] + 14500\} \gg 12$$

Για τις τιμές y=2,6,10,14:

$$tmp[y] = \left[\left(diff(y) - diff(y+1) - diff(y+2) + diff(y+3)\right) \ll 3\right]$$

Για τις τιμές y=3,7,11,15:

$$tmp[y] = \{2217 * \left[\left(diff(y-3) - diff(y)\right) \ll 3\right] - 5352 * \left[\left(diff(y-2) - diff(y-1)\right) \ll 3\right] + 14500\} \gg 12$$

Pass Two

Για τις τιμές y=0 ,1 ,2 ,3:

$$out[y] = [tmp(y) + tmp(y+4) + tmp(y+8) + tmp(y+12) + 7] \gg 4$$

Για τις τιμές y=4 ,5 ,6 ,7:

$$out[y] = \{2217 * (tmp(y) - tmp(y+4)) + 5352 * \left(tmp(y-4) - tmp(y+8)\right) + 12000\} \gg 16$$

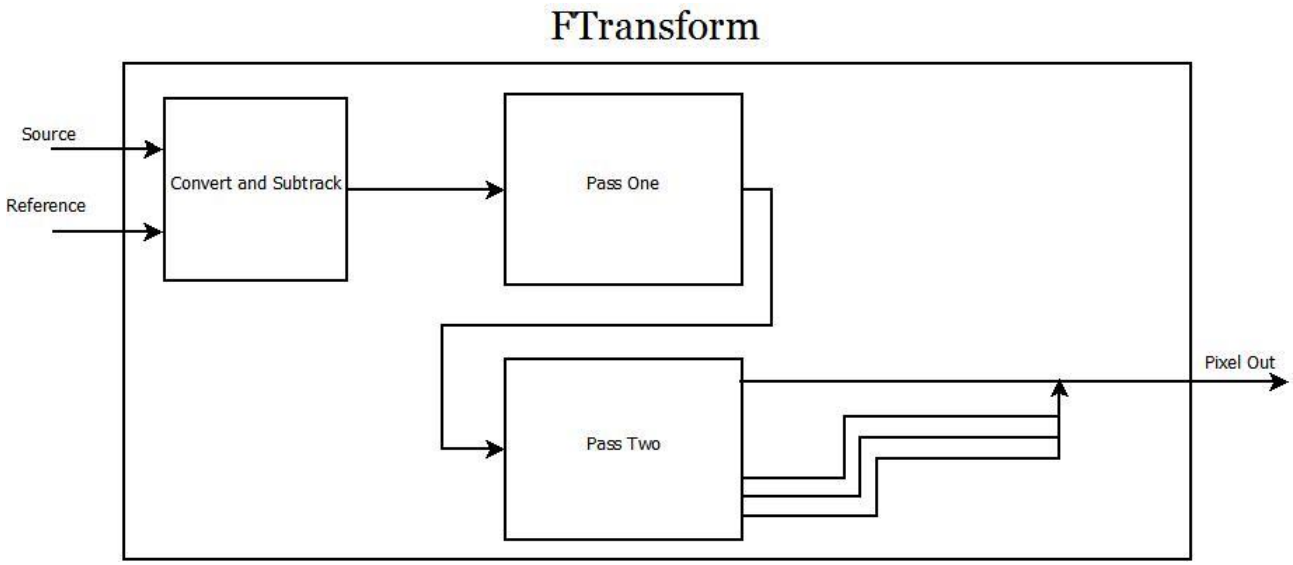
Για τις τιμές y=8 ,9 ,10 ,11:

$$out[y] = [tmp(y-8) + tmp(y+4) - tmp(y-4) + tmp(y) + 7] \gg 4$$

Για τις τιμές y=12 ,13 ,14 ,15:

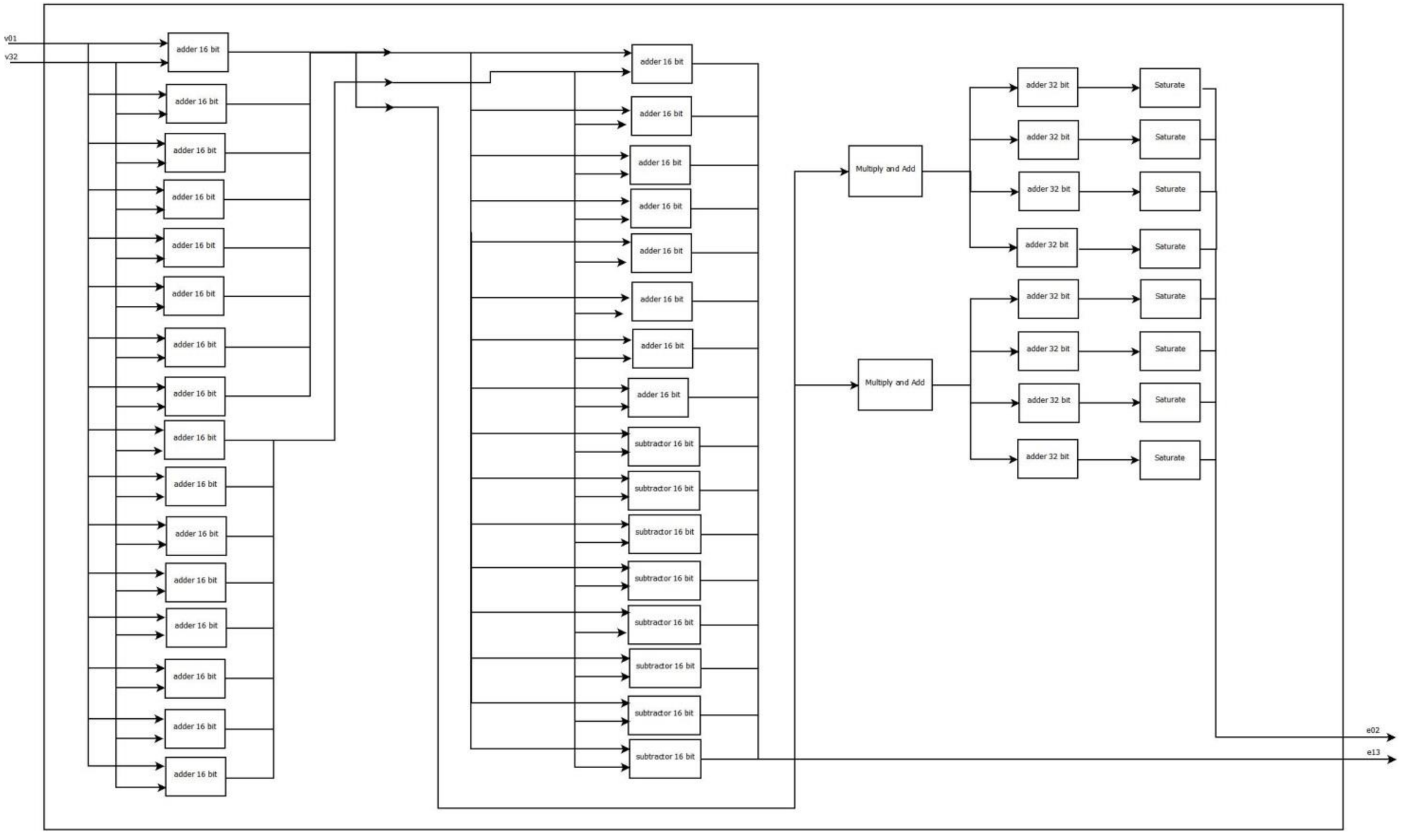
$$out[y] = \{2217 * (tmp(y-12) - tmp(y)) - 5352 * (tmp(y-8) - tmp(y-4)) + 51000\} \gg 16$$

Αυτές οι πράξεις οδηγούν στο αποτέλεσμα *out* της συνάρτησης FTransform και υλοποιούνται όπως στα ακόλουθα διαγράμματα υλοποίησης

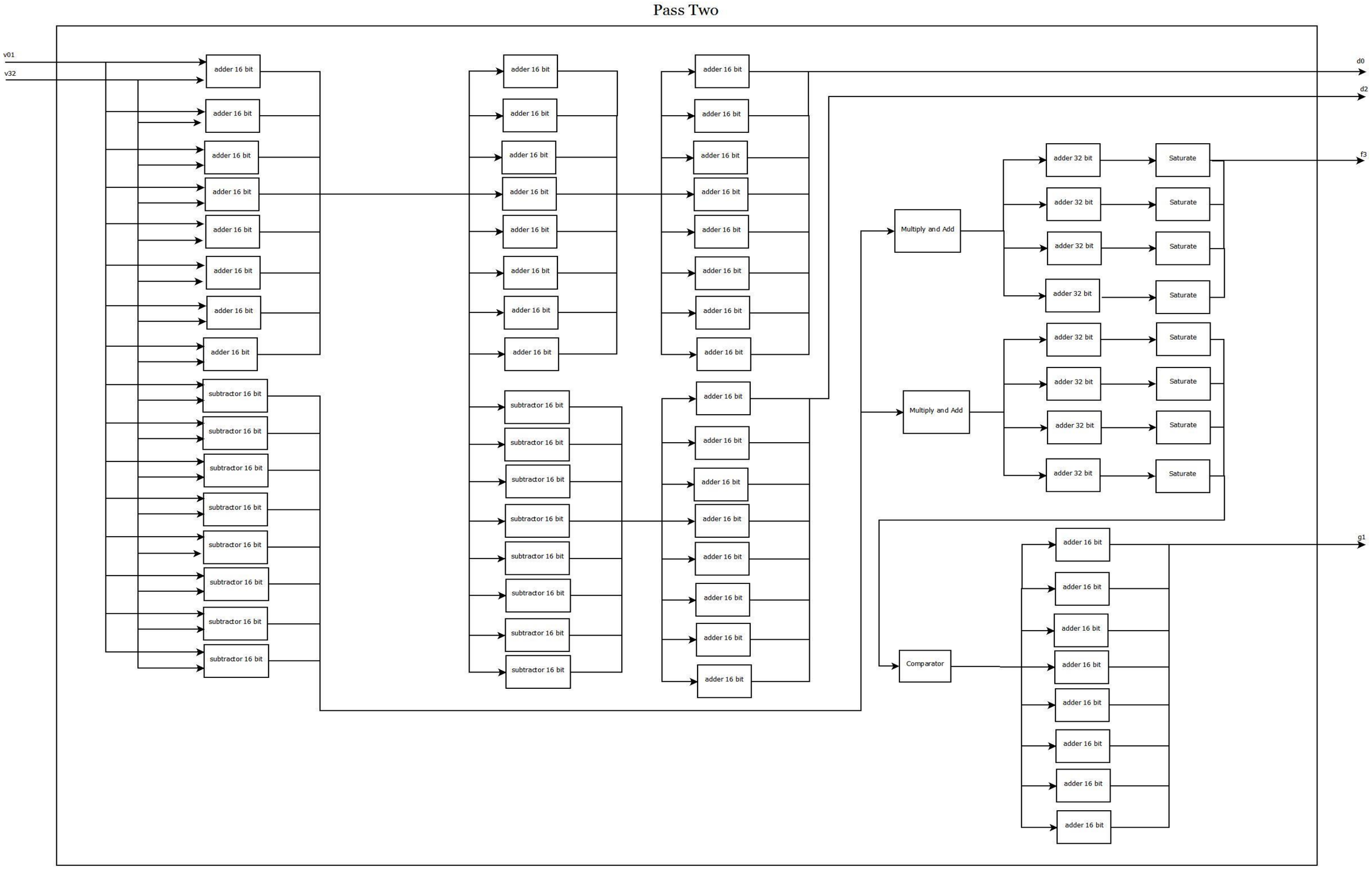


Εικόνα 30 Διάγραμμα υλοποίησης FTransform

Pass One



Εικόνα 31 Διάγραμμα υλοποίησης PassOne



Εικόνα 32 Διάγραμμα υλοποίησης PassTwo

5.4 Αποτελέσματα

Το σύστημα δεν κατέστη δυνατό να γίνει Synthesize και να δούμε μια ολοκληρωμένη προσομοίωση καθώς κατανάλωνε το πρόγραμμα υπερβολικά πολύ μνήμη και «έσκαγε». Τα ακόλουθα αποτελέσματα προέκυψαν από το άθροισμα των χρόνων του κάθε module το οποίο μπορέσαμε να δούμε προσομοίωση.

Η πιο χρονοβόρα συνάρτηση είναι η συνάρτηση υπολογισμού των Intra 4 προβλέψεων (MBAnalyzeIntra4) η οποία καταναλώνει 7680 κύκλους ρολογιού. Οι συναρτήσεις υπολογισμού των προβλέψεων Intra 16 και UV απαιτούν μόνο 16 κύκλους ρολογιού.

Κάνοντας υποθέσεις για τον χρονισμό μπορούμε να βγάλουμε τα ακόλουθα αποτελέσματα.

Με 100 MHz συχνότητα ρολογιού και υπολογίζοντας ότι η υποσυνάρτηση της MBAnalyzeIntra4 Intra4Preds έχει 29760 κλήσεις και κάθε MBAnalyzeIntra4 καλεί την Intra4Preds 16 φορές μπορούμε να υπολογίσουμε ότι χρειάζεται συνολικά 142.848 μs ή 142,8 ms για την συνάρτηση αυτή. Σύμφωνα με νέο profiling στο ίδιο σύστημα με το αρχικό, χρησιμοποιώντας όμως την τεχνολογία sse2, η οποία υποστηριζόταν από τον αρχικό αλγόριθμο και σκοπός της είναι η υλοποίηση συναρτήσεων σε ένα πυρήνα με καταχωρητές μεγέθους 128bit για μεγαλύτερη ταχύτητα και παραλληλία, αυτή η συνάρτηση έχει χρόνο 160 ms άρα μιλάμε για μία έστω και μικρή επιτάχυνση.

Ακόμα ήταν δύσκολη και χρονοβόρα η κατανόηση του αρχικού κώδικα λόγω του μεγέθους του αλλά και της έλλειψης πηγών για αυτόν.

Κεφάλαιο 6 Μελλοντική Εργασία

Η παρούσα εργασία σίγουρα επιδέχεται επιπλέον βελτιώσεων όσον αφορά τον χρόνο που καταναλώνει όσο και τον χώρο τον οποίο καταλαμβάνει.

Η εργασία είναι ημιτελής όσον αφορά τον αλγόριθμο οπότε σαν μελλοντική εργασία θα ήταν και η ολοκλήρωση του αλγορίθμου για την σωστότερη και ολοκληρωμένη σύγκριση.

Τέλος θα ήταν πολύ καλό και η υλοποίηση του αλγορίθμου με άλλες παραμέτρους και στοιχεία έτσι ώστε να μελετηθούν και άλλες περιπτώσεις συμπίεσης.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. TECHNICAL OVERVIEW OF VP8, AN OPEN SOURCE VIDEO CODEC FOR THE WEB by *Jim Bankoski, Paul Wilkins, Yaowu Xu*

2. http://search.eb.com/eb/article-9002216
3. http://www.maximumcompression.com/algoritms.php
4. http://en.wikipedia.org/wiki/Image_compression
5. http://en.wikipedia.org/wiki/Run-length_encoding
6. http://en.wikipedia.org/wiki/Entropy_coding
7. http://en.wikipedia.org/wiki/Huffman_coding
8. http://www.huffmancoding.com/david/algorithm.html
9. http://en.wikipedia.org/wiki/Dictionary_coder
10. http://en.wikipedia.org/wiki/LZW
11. http://en.wikipedia.org/wiki/Color_quantization
12. http://en.wikipedia.org/wiki/Comparison_of_graphics_file_formats
13. http://en.wikipedia.org/wiki/JPEG
14. http://www.impulseadventure.com/photo/jpeg-compression.html
15. http://en.wikipedia.org/wiki/GIF
16. http://www.w3.org/Graphics/GIF/spec-gif89a.txt
17. http://www.cis.udel.edu/~amer/CISC651/lzw.and.gif.explained.html
18. http://en.wikipedia.org/wiki/PNG
19. http://www.w3.org/TR/2003/REC-PNG-20031110/
20. http://en.wikipedia.org/wiki/Tagged_Image_File_Format
21. http://en.wikipedia.org/wiki/JPEG_2000
22. http://www.jpeg.org/faq.phtml?action=show_answer&question_id=q3d5bc0701c9b6
23. http://www.microsoft.com/whdc/xps/wmphoto.mspx
24. http://en.wikipedia.org/wiki/Digital_image
25. http://www.ptgrey.com/support/kb/index.asp?a=4&q=313
26. http://en.wikipedia.org/wiki/Color_space
27. http://en.wikipedia.org/wiki/Color_space
28. http://en.wikipedia.org/wiki/YCbCr
29. http://en.wikipedia.org/wiki/YUV
30. http://en.wikipedia.org/wiki/Resource_Interchange_File_Format
31. http://en.wikipedia.org/wiki/VP8
32. http://en.wikipedia.org/wiki/Structural_similarity
33. http://www.cnet.com/news/facebook-tries-googles-webp-image-format-users-squawk/
34. https://www.andrewmunsell.com/blog/jpg-vs-webp
35. http://en.wikipedia.org/wiki/WebP
36. http://blog.chromium.org/2011/11/lossless-and-transparency-encoding-in.html
37. https://developers.google.com/speed/webp/docs/compression
38. http://carlodaffara.conecta.it/a-small-webp-test/
39. http://www.codeproject.com/Articles/9207/An-HSV-RGBA-colour-picker
40. http://imrannazar.com/An-Introduction-to-Compression
41. http://www.hindawi.com/journals/vlsi/2012/209208/
42. https://developers.google.com/speed/webp/docs/riff_container
43. http://blog.webmproject.org/2010/07/inside-webm-technology-vp8-intra-and.html
44. https://developers.google.com/speed/webp/
45. http://www.scribd.com/doc/88355366/Realtime-VP8-2-9-2011
46. http://tools.ietf.org/rfc/rfc6386.txt
47. http://www.scribd.com/doc/78556576/Shah-EE5359Spring2011FinalPPT
48. http://blog.webmproject.org/2010/07/inside-webm-technology-vp8-intra-and.html

49. http://www.drtonygeorge.com/Video/h264/comp_architecture.htm
50. http://www.codeproject.com/Articles/9207/An-HSV-RGBA-colour-picker
51. http://www.hindawi.com/journals/vlsi/2012/209208/fig9/
52. http://www-i6.informatik.rwth-aachen.de/web/Misc/Coding/365/li/material/notes/Chap4/Chap4.2/Chap4.2.html