

**Σχεδιασμός και Ύλοποίηση
Επιταχυντή Ύλικου για
τον Αλγόριθμο Εκτίμησης Κίνησης
Εξαντλητικής Αναζήτησης**

Θωμάς Ι. Μαχρουνιώτης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ ΣΤΑ ΠΛΑΙΣΙΑ ΤΩΝ ΥΠΟΧΡΕΩΣΕΩΝ
ΓΙΑ ΤΗΝ ΑΠΟΝΟΜΗ ΤΟΥ ΔΙΠΛΩΜΑΤΟΣ ΜΗΧΑΝΙΚΟΥ
ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
ΤΗΣ
ΠΟΛΥΤΕΧΝΙΚΗΣ ΣΧΟΛΗΣ
ΤΟΥ
ΠΑΝΕΠΙΣΤΗΜΙΟΥ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

Επιβλεπων καθηγήτης:

ΔΡ. ΜΗΝΑΣ ΔΑΣΥΓΕΝΗΣ

Επιτροπή εξετασης:

PROFESSOR DELIGHTFUL RESEARCHER, Co-CHAIR
PROFESSOR EQUALLY D. RESEARCHER, Co-CHAIR
PROFESSOR PERSON INSIDE

ΜΑΡΤΙΟΣ 2017

©2015 – ΘΩΜΑΣ Ι. ΜΑΚΡΥΝΙΩΤΗΣ
ALL RIGHTS RESERVED.

**Σχεδιασμός και Γλοποίηση
Επιταχυντή Υλικού για
τον Αλγόριθμο Εκτίμησης Κίνησης Εξαντλητικής Αναζήτησης**

ΣΥΝΟΨΗ

Οι αλγόριθμοι εκτίμησης κίνησης (motion estimation algorithms) αποτελούν αναπόσπαστο κομμάτι όλων των σύγχρονων τεχνικών συμπίεσης/κωδικοποίησης video. Λόγω των εντατικών υπολογισμών που γίνονται για τον καθορισμό των διανυσμάτων κίνησης, κατά τη φάση αυτή απαιτείται μεγάλη επεξεργαστική ισχύς . Η συγκεκριμένη διεργασία είναι ιδιαίτερα δαπανηρή από πλευράς χρόνου και ενέργειας, ειδικά όταν γίνεται σε επίπεδο λογισμικού και ανατίθεται σε “μη-εξειδικευμένες” μονάδες όπως για παράδειγμα σ’εναν επεξεργαστή γενικής χρήσης (general purpose CPU). Η συγκεκριμένη διπλωματική εργασία, περιγράφει τον σχεδιασμό και την ανάπτυξη ενός περιφερειακού που λειτουργεί ως συνεπεξεργαστής-επιταχυντής και είναι εξειδικευμένο στην εκτέλεση του Αλγορίθμου Εκτίμησης Κίνησης Εξαντλητικής Αναζήτησης (Full-Search Motion Estimation Algorithm). Παράλληλα, περιγράφεται ο σχεδιασμός και η δημιουργία ενός πλήρους περιβάλλοντος διασύνδεσης με έναν επεξεργαστή ARM, χρησιμοποιώντας την πλατφόρμα Zynq-7000. Στόχος μας ήταν η δημιουργία ενός αξιόπιστου συστήματος που θα επιτρέπει την εκτέλεση του FSME με πολύ μεγαλύτερη ταχύτητα και πολύ μικρότερη κατανάλωση ενέργειας σε σχέση με συμβατικούς επεξεργαστές.

THIS IS THE DEDICATION.

Contents

1	ΕΙΣΑΓΩΓΗ	1
1.1	Περιγραφη του προβληματος	1
1.2	Κίνητρα και Στόχοι Γλοποίησης	3
1.3	Περιπτώσεις παρόμοιας έρευνας	4
1.4	Διάρθρωση κειμένου	4
2	ΘΕΩΡΗΤΙΚΟ ΓΠΟΒΑΘΡΟ	6
2.1	Η Κωδικοποίηση Εικόνας	6
2.2	Η Κωδικοποίηση Βίντεο	7
2.3	Το Γβριδικό Μοντέλο	9
2.4	Σχεδιασμός του Συστήματος Κωδικοποίησης	10
2.5	Εκτίμηση Κίνησης (Motion Estimation)	14
2.6	Αλγόριθμοι Εκτίμησης Κίνησης	22
3	ΣΧΕΔΙΑΣΜΟΣ ΤΟΥ ΓΛΙΚΟΥ	28
3.1	Εισαγωγή	28
3.2	Θεωρητική περιγραφή	28
3.3	Σχεδιασμός του Συστήματος	30
3.4	Εσωτερική Αρχιτεκτονική του Επιταχυντή	31
3.5	Αρχιτεκτονική του Συστήματος	42
4	ΛΟΓΙΣΜΙΚΟ	55
4.1	Το Λειτουργικό Σύστημα	55
5	CONCLUSION	57
	APPENDIX A SOME EXTRA STUFF	59
	REFERENCES	59

Ευχαριστίες

LOREM IPSUM DOLOR SIT AMET, consectetur adipiscing elit. Morbi commodo, ipsum sed pharetra gravida, orci magna rhoncus neque, id pulvinar odio lorem non turpis. Nullam sit amet enim. Suspendisse id velit vitae ligula volutpat condimentum. Aliquam erat volutpat. Sed quis velit. Nulla facilisi. Nulla libero. Vivamus pharetra posuere sapien. Nam consectetur. Sed aliquam, nunc eget euismod ullamcorper, lectus nunc ullamcorper orci, fermentum bibendum enim nibh eget ipsum. Donec porttitor ligula eu dolor. Maecenas vitae nulla consequat libero cursus venenatis. Nam magna enim, accumsan eu, blandit sed, blandit a, eros.

1

Εισαγωγή

1.1 ΠΕΡΙΓΡΑΦΗ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ

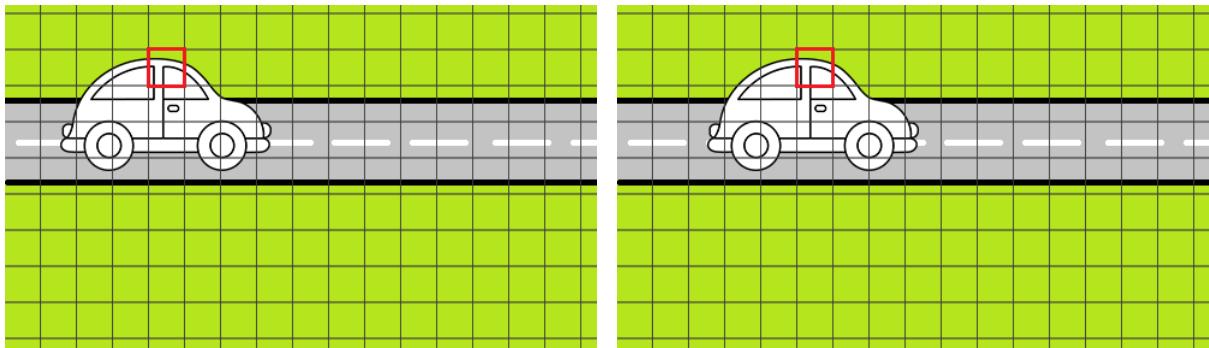
Με τη ραγδαία εξάπλωση του Internet, εξίσου ραγδαία ήταν και η εξάπλωση των πολυμέσων, μεταξύ των οποίων οι στατικές αλλά και οι κινούμενες εικόνες (βίντεο). Δυστυχώς, με την πάροδο του χρόνου, το bandwidth παρέμενε περιορισμένο, ενώ οι απαιτήσεις για ολοένα και υψηλότερη ποιότητα εικόνας αύξαναν συνεχώς. Το πρόβλημα είχε ήδη παρατηρηθεί από τα προηγούμενα χρόνια, και η αποτελεσματικότερη λύση ήταν η συμπίεση των αρχείων εικόνας/βίντεο, με σκοπό τη μείωση του μεγέθους τους και κατα συνέπεια της μνήμης και του εύρους ζώνης που απαιτούνταν [1]. Η χρήση αλγορίθμων συμπίεσης ξεκίνησε με την εφαρμογή διαφόρων τεχνικών συμπίεσης σε στατικές εικόνες, αρχικά σχετικά απλών, όπως στην περίπτωση του JPEG με τη χρήση του Διακριτού Μετασχηματισμού Συνημιτόνου (Discrete Cosine Transform - DCT), και στη συνέχεια πολυπλοκότερων όπως η συμπίεση με Fractals ή η συμπίεση πολλαπλών επιπέδων όπως στο πρότυπο PNG.

Το ίδιο μοτίβο ακολουθήθηκε και στην περίπτωση των βίντεο, τα οποία ως γνωστόν, είναι αλληλουχίες στατικών εικόνων που ονομάζονται **καρες (frames)**. Μια από τις πρώτες τεχνικές που εφαρμόστηκαν ήταν η συμπίεση κάθε καρέ με το πρότυπο JPEG κάτι που έγινε γνωστό ως MJPEG (Motion-JPEG). Φυσικά η σχέση ποιότητας/μεγέθους ήταν πολύ κακή και πλέον έπρεπε να εφευρεθούν νέες τεχνικές που θα επέτρεπαν μεγάλη συμπίεση κρατώντας όσο το δυνατόν σε υψηλότερα επίπεδα την ποιότητα.

Αυτό που παρατηρήθηκε πολύ γρήγορα ήταν το γεγονός πως, εφόσον πρόκειται για αλληλουχίες καρέ, το κάθε καρέ σε σχέση με τα γειτονικά του παρουσίαζε τεράστιες ομοιότητες ως προς την πληροφορία που έφερε. Αυτή η παρατήρηση οδήγησε στη δημιουργία

ενός νέου όρου, αυτού του χρονικού πλεονασμού (**temporal redundancy**). Ήδη ήταν γνωστή η έννοια του χωρικού πλεονασμού (**spatial redundancy**) η οποία αναφέρεται στην πληροφορία που μένει αναλλοίωτη μέσα σε μια συγκεκριμένη περιοχή της εικόνας, π.χ. οι τιμές των pixel ενός κόκκινου τετραγώνου διαστάσεων 32×32 pixel. Η έννοια του χρονικού πλεονασμού αφορά στην πληροφορία που μένει αναλλοίωτη μεταξύ των διαδοχικών καρέ. Για παράδειγμα, δύο διαδοχικά frames από ένα τμήμα βίντεο που δείχνουν την κίνηση ενός αυτοκινήτου σε ένα δρόμο έχουν εκ των πραγμάτων πολύ μικρές διαφορές μεταξύ τους (σχήμα 1.1).

Σχήμα 1.1: Δύο διαδοχικά frames

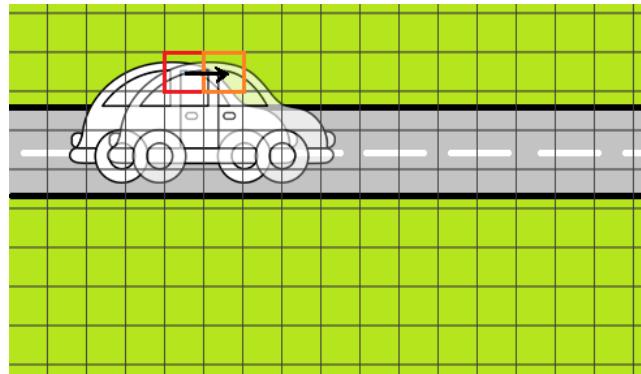


Οπως γίνεται εύκολα αντιληπτό, είναι δυνατόν να επιτευχθεί μεγάλος βαθμός συμπίεσης αν εκμεταλλευτούμε αυτά τα είδη των πλεονασμών. Πιο συγκεκριμένα, και σε ότι αφορά στον χρονικό πλεονασμό, είναι δυνατόν να συμπιέσουμε ένα καρέ χωρίζοντάς το σε μπλοκ (**blocks**), υπολογίζοντας την κίνηση τους και απεικονίζοντάς την χρησιμοποιώντας ένα διάνυσμα κίνησης (**motion vector**). Υπολογίζοντας τα διανύσματα κίνησης για όλα τα μπλοκ του καρέ έχουμε μια εκτίμηση κίνησης, και πρόκειται στην ουσία για έναν τρόπο να κωδικοποιήσουμε την πλεονάζουσα πληροφορία.

Κατα τη διαδικασία της συμπίεσης, αυτό που συμβαίνει είναι ο διαχωρισμός κάθε καρέ σε περιοχές αναζήτησης που με τη σειρά τους διαχωρίζονται σε μικρότερα μπλοκ (**macroblocks**). Κάθε macroblock της περιοχής αναζήτησης **P** ενός καρέ **n**, γίνεται προσπάθεια να αντιστοιχιθεί με κάποιο από τα macroblocks της περιοχής αναζήτησης **P** ενός καρέ **n+1** (του επόμενου ή των επόμενων ανάλογα με την τεχνική). Η καλύτερη δυνατή αντιστοίχιση, είναι και αυτή που θα δώσει τελικά το διάνυσμα κίνησης του συγκεκριμένου macroblock.

Υπάρχουν πολλοί αλγόριθμοι που υλοποιούν μια τέτοια διαδικασία αντιστοίχισης, όπως η δισδιάστατη λογαριθμική αναζήτηση (**2-D Logarithmic Search**), η αναζήτηση τριών βημάτων (**Three-Step Search**), ο αλγόριθμος **Diamond Search** και η εξαντλητική αναζήτηση. Οι αλγόριθμοι αυτοί ονομάζονται **block matching algorithms** επειδή προσπαθούν να υπολογίσουν τα διανύσματα κίνησης με βάση την αντιστοίχιση macroblocks. Όλοι τους φυσικά έχουν

Σχήμα 1.2: Το διάνυσμα κίνησης του σημειωμένου macroblock του σχήματος 1.1



τόσο πλεονεκτήματα όσο και μειονεκτήματα, όπως η ταχύτητα εκτέλεσης ή η ακρίβεια της αντιστοίχισης. Ειδικά όμως στην περίπτωση της εξαντλητικής αναζήτησης ισχύουν δύο πράγματα: παρέχει το καλύτερο δυνατό αποτέλεσμα από πλευράς ακρίβειας, δυστυχώς όμως υστερεί σε ταχύτητα και κατανάλωση πόρων, διότι πραγματοποιεί πληθώρα υπολογισμών αφού επεξεργάζεται pixel προς pixel κάθε macroblock.

Ακριβώς αυτό ήταν το πρόβλημα που έπρεπε να λύσουμε. Ο αλγόριθμος της εξαντλητικής αναζήτησης βρίσκει σημαντικές εφαρμογές στις συγχρονές τεχνικές κωδικοποίησης/συμπίεσης βίντεο όπως το πρότυπο H.265/HEVC, όμως είναι ιδιαίτερα δαπανηρός από πλευράς υπολογιστικής ισχύος και κατα συνέπεια, ενέργειας. Οι περισσότερες υλοποιήσεις του σε επίπεδο λογισμικού δεν ανταπέξερχονται ικανοποιητικά και οπωσδήποτε δεν μπορούν να χρησιμοποιηθούν για την κωδικοποίηση βίντεο σε πραγματικό χρόνο (για παράδειγμα σε φορητές κάμερες, κινητών τηλεφώνων, κλπ) αφού η εκτέλεση τους γίνεται χρησιμοποιόντας τα σε εντολών επεξεργαστών γενικής χρήσης (General-Purpose CPUs) ακόμα και όταν στους τελευταίους έχουν ενσωματωθεί ειδικά σετ εντολών για πολυμέσα.

Η λύση στο πρόβλημα μπορεί να δοθεί με το σχεδιασμό και την υλοποίηση ενός κυκλώματος ειδικού σκοπού, το οποίο λειτουργεί σε συνεργασία με την κεντρική CPU και είναι εξειδικευμένο στην εκτέλεση του αλγορίθμου εξαντλητικής αναζήτησης. Το κύκλωμα αυτό, ένας συνεπεξεργαστής επιταχυντής, μπορεί να βελτιώσει σημαντικά την απόδοση του αλγορίθμου και να τον καταστήσει μια λογική λύση ακόμα και σε περιπτώσεις όπου απαιτείται real-time κωδικοποίηση.

1.2 ΚΙΝΗΤΡΑ ΚΑΙ ΣΤΟΧΟΙ ΥΛΟΠΟΙΗΣΗΣ

Τα βασικά κίνητρα αναφέρθηκαν και στην προηγούμενη παράγραφο. Ο FSBMA (Full-Search Block Matching Algorithm) είναι μια πολύ καλύ προσέγγιση στο πρόβλημα της εκτίμησης κίνησης για εφαρμογές κωδικοποίησης βίντεο, που ωστόσο έχει τεράστιες απαιτήσεις σε επεξεργαστική ισχύ. Είναι επίσης σαφές πως αυτή η επεξεργαστική ισχύς που απαιτεί,

μεταφράζεται σε μεγάλη κατανάλωση ενέργειας από τις υπολογιστικές μονάδες, κάτι που καθιστά ασύμφορη την επιλογή του σε ενσωματωμένα συστήματα και φορητές συσκευές, που λειτουργούν κυρίως με μπαταρίες. Επιπλέον, η έλευση νέων προτύπων για την κωδικοποίηση βίντεο UHD (Ultra-High Definition) απαιτεί τη χρήση όσο το δυνατόν πιο αξιόπιστων αλγορίθμων, με ελάχιστες απώλειες στην ποιότητα.

Οι στόχοι της υλοποίησης ήταν τρεις. Πρώτον, θα έπρεπε η υλοποίηση στο hardware να είναι ταξιδιώτης μεγέθους ταχύτερη στην εκτέλεση από τις υλοποιήσεις σε software. Δεύτερον, θα έπρεπε η κατανάλωση ενέργειας να είναι όσο το δυνατόν πιο περιορισμένη, σε επίπεδα όπου θα δικαιολογείται η χρήση του κυκλώματος σε φορητές συσκευές. Τρίτος στόχος της υλοποίησης ήταν η κατασκευή ενός ολοκληρωμένου περιβάλλοντος που θα περιλαμβάνει το κύκλωμα του επιταχυντή, τον επεξεργαστή και τα κυκλώματα διασυνδεσης, αλλά και το λειτουργικό σύστημα μαζί με το λογισμικό ελέγχου (driver) του επιταχυντή. Έτσι, μπορούμε να ελέγξουμε και να διαπιστώσουμε στην πράξη την απόδοση του συστήματος.

1.3 ΠΕΡΙΠΤΩΣΕΙΣ ΠΑΡΟΜΟΙΑΣ ΕΡΕΥΝΑΣ

Παρόμοιο έργο έχει εκπονηθεί από πολλούς ερευνητές ανα τον κόσμο. Χαρακτηριστική είναι η δουλειά του Y. Ismail [2], [3], [4] αλλά και των Goel, Bayoumi [4], όπως και των Redkar και Kuwelkar [5], όπου αναφέρονται με λεπτομέρειες στις διαδικασίες σχεδιασμού και υλοποίησης ενός κυκλώματος επιτάχυνσης για τον αλγόριθμο FSBMA. Φυσικά κάθε ομάδα εστιάζει σε διαφορετικά στοιχεία, όπως για παράδειγμα η επαναχρησιμοποίηση δεδομένων έτσι όπως προτάθηκε από τον Ismail και τους συνεργάτες του.

Ο Olivares και οι συνεργάτες του [6] σχεδίασαν επίσης ένα κύκλωμα υλοποίησης του αλγορίθμου FSBMA και το συνέχριναν με παλαιότερες υλοποιήσεις των Ryszko [7], Wong [8], Roma [9] και Loukil [10], επιτυγχάνοντας σημαντική αύξηση της απόδοσης. Φυσικά, το 2006 δεν υπήρχαν εργαλεία και πλατφόρμες όπως η Zynq, που μας δίνει τη δυνατότητα να προτουποποιούμε και να δοκιμάζουμε πολύ γρήγορα τα νέα κυκλώματα σε πραγματικό περιβάλλον.

1.4 ΔΙΑΡΘΡΩΣΗ ΚΕΙΜΕΝΟΥ

Τα επόμενα κεφάλαια οργανώνονται ως εξής:

Στο δεύτερο κεφάλαιο γίνεται μια αναλυτική επεξήγηση της διαδικασίας συμπίεσης, καθώς και επιγραμματική αναφορά σε διάφορους αλγορίθμους και τεχνικές συμπίεσης εικόνας και βίντεο. Δίνεται επίσης όλο το απαραίτητο θεωρητικό υπόβαθρο για την κατανόηση του Αλγορίθμου Εκτίμησης Κίνησης Εξαντλητικής Αναζήτησης.

Το τρίτο κεφάλαιο αφορά το επίπεδο του hardware. Στο κεφάλαιο αυτό περιγράφεται αναλυτικά η αρχιτεκτονική του επιταχυντή, και οι βασικές συστατικές του μονάδες. Περιγράφονται επίσης οι διαδικασίες που ακολουθήθηκαν στο σχεδιασμό του, αλλά και οι προκλήσεις και τα

σφάλματα που έπρεπε να αντιμετωπιστούν. Αναλύεται επίσης η διασύνδεση του επιταχυντή (μονάδες DMA, διασύνδεση AXI) με το υπόλοιπο υπολογιστικό σύστημα: τον επεξεργαστή ARM και την κεντρική μνήμη.

Στο τέταρτο κεφάλαιο περιγράφεται η υλοποίηση του software. Σχολιάζονται θέματα του λειτουργικού συστήματος (Linux), και αναλύεται η διαδικασία συγγραφής του driver και του testbench για τον έλεγχο και το verification του συστήματος.

Τέλος, στο πέμπτο κεφάλαιο παρατίθενται τα αποτελέσματα, συνοδευόμενα από στοιχεία μετρικών και συμπεράσματα για τη λειτουργικότητα και αξιοπιστία του συστήματος.

Στα παραρτήματα βρίσκονται τόσο τμήματα κώδικα όσο και σχεδιαγράμματα που περιγράφουν αναλυτικότερα τη λειτουργία του επιταχυντή.

2

Θεωρητικό Υπόβαθρο

2.1 Η Κωδικοποίηση Εικόνας

Οι φηφιακές εικόνες ήταν από τις πρώτες μορφές δεδομένων που απαιτήθηκε να συμπιεστούν. Ο λόγος είναι ότι πρόκειται για πολύπλοκες μορφές δεδομένων, συχνά με πολύ μεγάλη λεπτομέρεια στην απεικόνιση και που ωστόσο, είναι αποδεκτό ορισμένες φορές να μειώσουμε τη λεπτομέρεια μιας εικόνας με σκοπό πολύ μεγάλη μείωση του μέγεθους της. Κάτι τέτοιο είναι ιδιαίτερα χρήσιμο, τόσο για την αποθήκευση όσο και τη μετάδοση εικόνων, ειδικά όταν η διαθέσιμη μνήμη ή το διαθέσιμο ύψος ζώνης είναι περιορισμένα. Η ανάγκη αυτή έγινε ακόμα μεγαλύτερη με την ταχεία εξάπλωση του διαδικτύου και των νέων εφαρμογών που βασίζονται σε αυτό. Υπάρχουν δύο μορφές συμπίεσης, η **συμπίεση με απώλειες (lossy)** και η **συμπίεση χωρίς απώλειες (lossless)**.

Στη συμπίεση με απώλειες προσπαθούμε να κωδικοποιήσουμε μέρος της πλεονάζουσας πληροφορίας με τεχνικές οι οποίες εκ των πραγμάτων αλλοιώνουν το οπτικό αποτέλεσμα. Αυτό συμβαίνει γιατί η βασική αρχή κάθε μεθόδου απωλεστικής συμπίεσης είναι οι μείωση του συσχετισμού μεταξύ των τιμών των pixel μιας εικόνας. Ο κύριος λόγος για τον οποίο είναι εφικτή η συμπίεση μιας εικόνας είναι ο πολύ μεγάλος συσχετισμός που υπάρχει ανάμεσα σ'ενα pixel και τα γειτονικά του. Αυτό πρακτικά μεταφράζεται στο ότι οι τιμές ενός pixel και των γειτονικών του είναι παρόμοιες. Αν όμως μπορέσουμε να μειώσουμε τον συσχετισμό αυτό, είναι σχετικά εύκολο να εκμεταλλευτούμε τα στατιστικά χαρακτηριστικά που παρουσιάζονται και τελικά να μειώσουμε το μέγεθος που καταλαμβάνει η εικόνα. Υπάρχουν πολλές τεχνικές που το πετυχαίνουν αυτό, όπως ο **διακριτός μετασχηματισμός συνημιτόνου (DCT)** που χρησιμοποιείται στο πρότυπο JPEG. Άλλες μέθοδοι απωλεστικής

συμπίεσης είναι η υποδειγματοληφία χρώματος (**chrominance downsampling**) και η συμπίεση με **fractals**.

Σε ότι αφορά στη μη-απωλεστική συμπίεση, οι μέθοδοι που χρησιμοποιούνται είναι λιγότερο ”επιθετικές” και πετυχαίνουν μικρότερου βαθμού συμπίεση. Παρόλα αυτά, η ποιότητα της εικόνας παραμένει αθικτη. Τέτοιες τεχνικές περιλαμβάνουν τη διαφορική παλμοκωδική διαμόρφωση (DPCM) και τον αλγόριθμο DEFLATE. Ειδικά ο τελευταίος αποτελεί το βασικό αλγόριθμο συμπίεσης του προτύπου PNG.

Όλα τα παραπάνω που αναφέρθηκαν αφορούν όπως είπαμε στην κωδικοποίηση της πλεονάζουσας πληροφορίας. Ως εκ τούτου, εισάγεται ένας νέος όρος, αυτός του **χωρικού πλεονασμού (spatial redundancy)** και ο οποίος αφορά στην επανάληψη συγκεκριμένης πληροφορίας (τιμές pixel στη συγκεκριμένη περίπτωση) μέσα σε μια δομή δεδομένων (την εικόνα).

2.2 Η Κωδικοποίηση BINTEO

Συχνά, όταν αναφερόμαστε στο θέμα του βίντεο αμελούμε το γεγονός πως στην ουσία πρόκειται για αλληλουχία στατικών εικόνων. Οι εικόνες αυτές ονομάζονται **καρέ** ή **frames** και διαθέτουν όλα τα χαρακτηριστικά μιας κοινής στατικής εικόνας όπως ανάλυση, φωτεινότητα (luminance) και χρώμα (chrominance). Είναι λοιπόν προφανές, πως όσα αναφέραμε προηγουμένως για την κωδικοποίηση/συμπίεση εικόνας, μπορούν πολύ έυκολα να εφαρμοστούν στο βίντεο. Δυστυχώς, το βίντεο εμπειρέχει ορισμένες ιδιαιτερότητες που δυσκολεύουν την αποθήκευση και την επεξεργασία του ακόμα και μετά την εφαρμογή των προαναφερθεισών τεχνικών.

Σε ότι αφορά το βίντεο ως μορφή σήματος, υπάρχουν σημαντικές διαφορές σε σχέση με τη στατική εικόνα. Η βασικότερη όμως διαφορά εντοπίζεται στο ότι η στατική εικόνα περιλαμβάνει πληροφορία η οποία είναι σταθερή και αμετάβλητη ως προς το χρόνο, σε σχέση με το βίντεο που διαθέτει επιπλέον μια χρονική διάσταση. Το σήμα του βίντεο μεταβάλλεται ως προς το χρόνο με κάποιο συγκεκριμένο ρυθμό **καρέ ανά δευτερόλεπτο (frames per second ή FPS)**. Προκειμένου να είναι ορατό στο ανθρώπινο μάτι ως ομαλή συνεχής κίνηση και όχι ως διακοπτόμενη ακόλουθία εικόνων, θα πρέπει ο ρυθμός αυτός να είναι όσο το δυνατόν μεγαλύτερος. Παρακάτω παραθέτουμε έναν πίνακας (2.1) με διάφορα framerates και την αντίληψη κίνησης όπως προκύπτει από αυτά.

Η περίπτωση των 30 FPS αποτελεί την πιο διαδεδομένη, ειδικά σε ότι αφορά το ψηφιακό βίντεο και τις υπηρεσίες που το προσφέρουν (YouTube), καθώς και τις συσκευές που το καταγράφουν/αναπαράγουν. Σε αυτήν την περίπτωση αυτό που ισχύει πρακτικά είναι ότι ανα 33ms περίπου, θα πρέπει να λαμβάνεται ένα καινούργιο frame.

Το γεγονός αυτό μας οδηγεί σε ενα πολύ σημαντικό συμπέρασμα: είναι πολύ περιορισμένη η κίνηση που μπορεί να υπάρξει και να καταγραφεί, είτε της κάμερας είτε του θέματος, μέσα σ' αυτό το παράθυρο των 33ms (ή ακόμα μικρότερο για μεγαλύτερα framerates). Επομένως,

Πίνακας 2.1: Διάφορα framerates και το αποτέλεσμά τους

Framerate (FPS)	Αντίληψη κίνησης
10-12	Το απόλυτο ελάχιστο για την αντίληψη συνεχόμενης κίνησης. Οτιδήποτε μικρότερο, γίνεται αντιληπτό ως ξεχωριστές εικόνες.
<16	Η αλληλουχία των frames είναι ορατή. Μπορεί να ενοχλήσει πολλούς θεατές.
24	Το ελάχιστο ανεκτό για την αντίληψη μιας ομαλής κίνησης. Αποτελεί το κυνηγατογραφικό πρότυπο.
30	Πολύ καλύτερο σήμα σε σχέση με τα 24 FPS. Αποτελεί την καθιερωμένη πρακτική για το πρότυπο NTSC λόγω της συχνότητας του εναλλασσόμενου ρεύματος στις ΗΠΑ (60Hz), καθώς και την πλειοψηφία των φηφιακών βίντεο.
48	Αυξημένη ποιότητα, που πλησιάζει την αντίληψη της κίνησης ως φυσική.
60	Ιδιαίτερα αυξημένη ποιότητα σε σχέση με τα προηγούμενα framerates. Οι περισσότεροι άνθρωποι αντιλαμβάνονται την κίνηση ως φυσική.

σε μεγάλο βαθμό υπάρχει πλεονάζουσα πληροφορία ανάμεσα σε γειτονικά frames. Με βάση την παρατήρηση αυτή, μπορούμε να εισάγουμε τον όρο του **χρονικού πλεονασμού (temporal redundancy)**, που υποδηλώνει τον πλεονασμό της πληροφορίας σε ένα σήμα που μεταβάλλεται ώς προς το χρόνο.

Στην περίπτωση του προτύπου Full-HD (Full High-Definition) κάθε καρέ αποτελεί μια στατική εικόνα ανάλυσης 1920×1080 pixel. Με μερικούς απλούς υπολογισμούς διαπιστώνουμε ότι:

$$1920 \cdot 1080 \cdot 32\text{bpp} \approx 8\text{MBytes/frame}$$

και τελικά:

$$8 \cdot 30 \approx 240\text{MBytes/second}$$

Δηλαδή, κάθε δευτερόλεπτο βίντεο σε μορφή Full HD με ρυθμό 30 FPS, απαιτεί 240 MBytes μνήμης εφόσον είναι πλήρως ασυμπίεστο (σε μορφή RAW). Ακόμα και με "επιθετικούς" αλγόριθμους συμπίεσης εικόνας, το μέγεθος παραμένει πολύ μεγάλο και μας περιορίζει σημαντικά στην περίπτωση που θέλουμε να μεταδώσουμε το βίντεο στο Internet ή να αυξήσουμε την ποιότητά του. Ειδικά σε ότι αφορά το τελευταίο, τα πρότυπα 4K και 8K που τείνουν να καθιερωθούν απαιτούν αντίστοιχα τετραπλάσια και δεκαεξαπλάσια μνήμη ανα δευτερόλεπτο! Είναι προφανές λοιπόν, πως είναι απαραίτητο να εφαρμοστούν νέες τεχνικές συμπίεσης εξειδικευμένες πάνω στις αλληλουχίες εικόνων, που θα εκμεταλλεύονται

τα ιδιαίτερα χαρακτηριστικά που αυτές παρουσιάζουν.

Όπως αναφέρθηκε και στην προηγούμενη ενότητα, στην περίπτωση των στατικών εικόνων εκμετάλλευμαστε το χαρακτηριστικό του χωρικού πλεονασμού προκειμένου να επιτύχουμε συμπίεση. Ένας τρόπος για να το πετύχουμε αυτό είναι να μεταφέρουμε το σήμα της εικόνας από το πεδίο του χρόνου, στο πεδίο της συχνότητας χρησιμοποιώντας είτε το διακριτό μετασχηματισμό συνημμτόνου (DCT) είτε το διακριτό μετασχηματισμό wavelet (DWT). Με τις δύο αυτές μεθόδους αυτό που πετυχαίνουμε είναι να μειώσουμε τον συσχετισμό μεταξύ των τιμών των pixel μιας εικόνας, διατηρώντας την ενέργεια στις χαμηλές συχνότητες (η ενέργεια του σήματος για τις περισσότερες εικόνες βρίσκεται στις χαμηλότερες συχνότητες) και παραλείποντας τις υψηλές, με πολύ μικρή οπτική παραμόρφωση.

Αυτό που πρέπει πλέον να φροντίσουμε είναι η αποτελεσματική εκμετάλλευση του χρονικού πλεονασμού. Στο σημείο αυτό, ένα στοιχείο που μπορεί να μας βοηθήσει σημαντικά είναι η εμπειρία μας στην καδικοποίηση του ήχου. Μια πολύ διαδεδομένη τεχνική είναι η διαφορική παλμοκωδική διαμόρφωση (DPCM) η οποία αποτελεί στην ουσία τεχνική πρόβλεψης. Με την DPCM καταφέρνουμε να προβλέψουμε την τιμή που θα έχει το σήμα μας σε μια δεδομένη χρονική στιγμή, βασιζόμενοι στις τιμές του σήματος σε προηγούμενες χρονικές στιγμές. Παρομοίως, αυτό που μπορούμε να κάνουμε στην περίπτωση του βίντεο είναι να προβλέψουμε την τιμή που έχουν τα pixel ενός frame βασιζόμενοι στις τιμές που είχαν τα γειτονικά τους frames (είτε προηγούμενα, είτε επόμενα).

Εφόσον διαθέτουμε δύο τεχνικές που η καθεμιά είναι εξειδικευμένη σε συγκεκριμένες μορφές σημάτων και εφόσον το σήμα του βίντεο μπορεί να θεωρηθεί ότι αποτελείται από δύο συνιστώσες, είναι φανερό ότι μπορούμε να δημιουργήσουμε ένα **υβριδικό σύστημα καδικοποίησης/αποκωδικοποίησης (hybrid codec)** με το οποίο μπορούμε τελικά να πετύχουμε - θεωρητικά τουλάχιστον - ικανοποιητική καδικοποίηση και συμπίεση ενός σήματος βίντεο. Εν κατακλείδι, θα έχουμε ένα σύστημα το οποίο θα χρησιμοποιεί τον μετασχηματισμό στο πεδίο των συχνοτήτων για την καδικοποίηση του χωρικού πλεονασμού και μια τεχνική πρόβλεψης (όπως η DPCM) για την καδικοποίηση του χρονικού πλεονασμού.

2.3 Το Ύβριδικο ΜΟΝΤΕΛΟ

Οι τεχνικές πρόβλεψης όπως η DPCM μπορούν να χρησιμοποιηθούν τόσο για μεταβαλλόμενα όσο και για αμετάβλητα ως προς το χρόνο σήματα. Είναι ωστόσο αποδεδειγμένο ότι ο χωρικός πλεονασμός είναι πολύ πιο εύκολο να αξιοποιηθεί με τεχνικές μεταφοράς στο πεδίο των συχνοτήτων. Από την άλλη, ο χρονικός πλεονασμός είναι πολύ πιο έυκολα αξιοποιήσιμος από τεχνικές καδικοποίησης με χρήση πρόβλεψης. Το ερώτημα που φυσικά γεννάται είναι, πως εφαρμόζεται η πρόβλεψη στα σήματα βίντεο;

Αν θεωρήσουμε πως δεχόμαστε το σήμα βίντεο ως είσοδο διαδοχικών frames, τότε αυτά τα δεχόμαστε με κάποια σειρά και αναγκαστικά θα καταφθάνουν σε χρονικές στιγμές t ,

t+1, t+2, κλπ. Αν το σύστημά μας διαθέτει μνήμη ωστε να είναι δυνατόν να αποθηκευεται ένα frame t , τότε είναι εφικτό να το χρησιμοποιήσουμε ωστε να προβλέψουμε ποιο θα είναι το επόμενο frame, $t+1$. Είναι σαφές ότι η πρόβλεψη, με όποιον μηχανισμό και ον γίνει, επειδή βασίζεται αποκλειστικά σε "παρελθοντικό" frame δεν πρόκειται να είναι απόλυτα ακριβής, επομένως θα περιέχει κάποιο σφάλμα. Αυτό το σφάλμα πρόβλεψης, όπως ονομάζεται, παρουσιάζει επίσης ένα βαθμό χωρικού πλεονασμού. Αυτό συμβαίνει γιατί όπως θα διαπιστώσουμε στη συνέχεια, σε ότι αφορά το σήμα του σφάλματος πρόβλεψης, τα διαδοχικά pixels στο χώρο παρουσιάζουν μεγάλο βαθμό ομοιότητας αφού όπως ήδη εξηγήσαμε, οι αλλαγές στο frame είναι πολύ μικρες. Αυτός ο πλεονασμός λοιπόν, είναι δυνατόν να αξιοποιηθεί με κάποια τεχνική μεταφοράς στο πεδίο των συχνοτήτων όπως η DCT.

Τελικά το υβριδικό μοντέλο που περιγράφαμε χρησιμοποιεί:

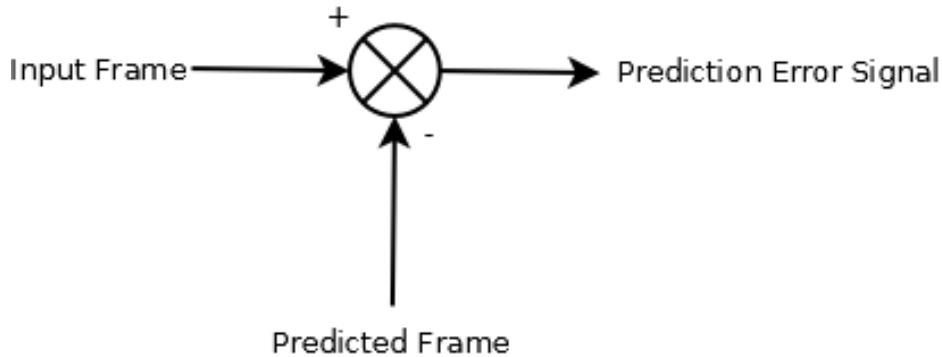
- **Κωδικοποίηση πρόβλεψης (predictive coding)**, για την αξιοποίηση του χρονικού πλεονασμού
- **Κωδικοποίηση με μεταφορά στο πεδίο συχνοτήτων (transform-domain coding)**, για την αξιοποίηση του χωρικού πλεονασμού του σφάλματος πρόβλεψης

2.4 ΣΧΕΔΙΑΣΜΟΣ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ ΚΩΔΙΚΟΠΟΙΗΣΗΣ

Σύμφωνα με τα όσα περιγράφαμε παραπάνω είναι σχετικά εύκολο να υποθέσουμε τη δομή του συστήματος που θα αποτελεί τον κωδικοποιητή του σήματος βίντεο. Τα απολύτως απαραίτητα συστατικά μέρη περιλαμβάνουν πέρα από την είσοδο/έξοδο του συστήματος, έναν αφαιρέτη, έναν μηχανισμό μεταφοράς στο πεδίο της συχνότητας, ένα μηχανισμό πρόβλεψης και μια μνήμη, στην οποία θα αποθηκευεται κάποιο ή κάποια frames. Αρχικά, η είσοδος του συστήματος μας οδηγείται στον αφαιρέτη, μαζί με το frame που έχουμε προβλέψει. Το τελευταίο αφαιρείται από το σήμα εισόδου και το αποτέλεσμα που εξάγεται αποτελεί το σφάλμα πρόβλεψης ή **υπόλοιπο**. Το σχήμα 2.1 δείχνει ακριβώς αυτή τη δομή. Στο σημείο αυτό αξίζει να σημειώσουμε πως εσκεμμένα δεν αναφερόμαστε περαιτέρω στη διαδικασία της πρόβλεψης, καθώς θα ασχοληθούμε εκτενώς με αυτήν σε επόμενη ενότητα.

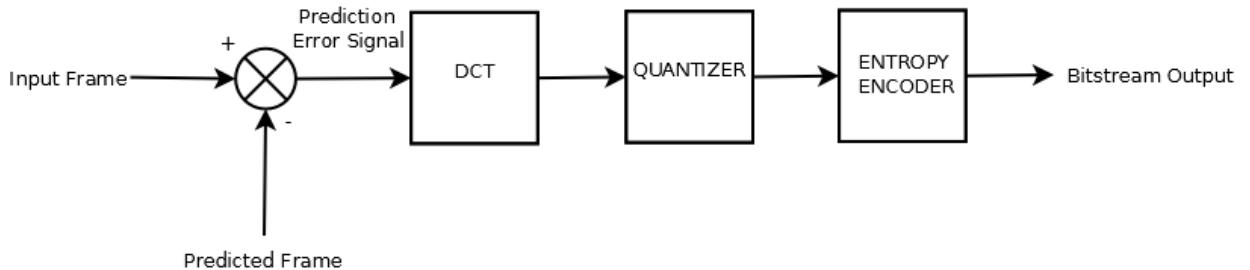
Όπως αναφέραμε και προηγουμένως, το σφάλμα πρόβλεψης είναι αυτό το οποίο θα κωδικοποιηθεί χρησιμοποιώντας τη μεταφορά στο πεδίο των συχνοτήτων. Επομένως, στο επόμενο στάδιο, θα πρέπει να τροφοδοτήσουμε το σφάλμα εισόδου σε μια μονάδα DCT (διακριτού μετασχηματισμού συνημιτόνου). Ο λόγος που χρησιμοποιούμε τον DCT και όχι κάποια άλλη τεχνική, όπως η DWT, είναι ότι είναι αρκετά διαδεδομένη σε πολλά σύγχρονα πρότυπα συμπίεσης (ως DCT-II). Αφού το σήμα μεταφερθεί στο πεδίο των συχνοτήτων θα πρέπει στη συνέχεια να κβαντιστεί και να εφαρμοστεί **κωδικοποίηση εντροπίας** έπειτα

Σχήμα 2.1: Το διάγραμμα του αφαιρέτη



από την οποία θα λάβουμε το τελικό bitstream. Το σχήμα 2.2 δείχνει το σύστημα μέχρι τώρα με τα συστατικά του στοιχεία.

Σχήμα 2.2: Το διάγραμμα του συστήματος μετά την κωδικοποίηση του σφάλματος πρόβλεψης



Είναι απαραίτητο να τονίσουμε ότι με τον τρόπο αυτό δεν κωδικοποιούμε το πραγματικό σήμα, αλλά το υπόλοιπο, το σφάλμα πρόβλεψης που είναι μόνο μια συνιστώσα του "χρήσιμου" σήματος, αυτού δηλαδή που θα εμφανίσουμε τελικά στην οθόνη. Αυτό που πρέπει να γίνει προκειμένου να πάρουμε ξανά ένα ολοκληρωμένο frame είναι να προστεθεί το σφάλμα πρόβλεψης με το predicted frame, κάτι που είναι αδύνατον σε επίπεδο αποκωδικοποιητη. Ο λόγος για τον οποίο συμβαίνει αυτό είναι ότι ο αποκωδικοποιητής δεν έχει γνώση κανενός άλλου σήματος πέραν του σφάλματος πρόβλεψης. Μπορούμε να θεωρήσουμε την είσοδο του συστήματος ως μια συνάρτηση τριών μεταβλητών:

$$s(n_1, n_2, k)$$

όπου s η ένταση (intensity) των pixel στη θέση (n_1, n_2) κατά τη χρονική στιγμή t . Είναι σημαντικό να γνωρίζουμε πως οι τρεις αυτές μεταβλητές παίρνουν ακέραιες τιμές, επομένως

η μεταβλητή t συμβολίζει τον αριθμό του frame. Αντίστοιχα μπορούμε να αναπαραστήσουμε το predicted frame ως:

$$\hat{s}(n_1, n_2, k)$$

και το σφάλμα πρόβλεψης ως:

$$e(n_1, n_2, k)$$

το οποίο μετά τον κβαντιστή αναπαριστάται ως:

$$\dot{e}(n_1, n_2, k)$$

Τελικά ο αρχικός ισχυρισμός μας για την ανακατασκευή ενός ολοκληρωμένου frame εκφράζεται ως:

$$\dot{e}(n_1, n_2, k) + \hat{s}(n_1, n_2, k) = s(n_1, n_2, k)$$

Αξίζει να σημειώσουμε ότι $\dot{s} \neq s$ όπως επίσης $\dot{e} \neq e$. Οι τελείς εκφράζουν το σήμα αφού έχει υποστεί απώλειες λόγω της δειγματοληψίας. Ως εκ τούτου, και το frame το οποίο ανακατασκευάζεται με όθροιση του κβαντισμένου σφάλματος πρόβλεψης και του predicted frame, δεν είναι το ίδιο με εκείνο που δεχτήκαμε στην είσοδο του συστήματος.

Αναφερθήκαμε ήδη στο μηχανισμό πρόβλεψης χωρις ιδιαίτερες λεπτομέρειες, πέραν του ότι θα αναπαράγει το predicted frame βασιζόμενος σε παρελθοντικά frames. Τα παρελθοντικά frames γενικά τη μορφή: $s(n_1, n_2, k - L)$, όπου $L > 0$. Αυτό που δεν αναλύσαμε ακόμη είναι το πώς χρησιμοποιούμε τα παρελθοντικά frames προκειμένου να παράξουμε τα predicted frames.

Είναι απλό κάποιος να ισχυριστεί ότι η παραγωγή του predicted frame μπορεί να γίνει σχετικά εύκολα, αρκεί να χρησιμοποιήσουμε την εξής διαδικασία:

- Έστω ότι λαμβάνουμε ένα frame $t-1$ στην είσοδο του συστήματός μας, το οποίο αποθηκεύουμε σε κάποια μνήμη.
- Στη συνέχεια λαμβάνουμε ένα νέο frame t .
- Συγκρίνουμε τα δύο frames μεταξύ τους υπολογίζοντας την μετατόπιση του κάθε block στα οποία έχουμε διαχωρίσει τα frame (εκτίμηση κίνησης).
- Τελικά με κάποιο μηχανισμό (που τον θεωρούμε ως "μαύρο κουτί" για λόγους απλότητας) κάνουμε την πρόβλεψη κίνησης και παίρνουμε το predicted frame.

Η προσέγγιση αυτή, αν και αρκετά απλή και εφικτή στα πλαίσια του κωδικοποιητή, περιέχει μια εγγενή αδυναμία. Ο μηχανισμός πρόβλεψης πρέπει να είναι ο ίδιος στον κωδικοποιητή και τον αποκωδικοποιητή. Με αυτόν τον τρόπο και μόνο, μπορούμε να είμαστε σίγουροι ότι ο αποκωδικοποιητής θα μπορέσει να αναπαράγει την ακριβή πληροφορία που του δώσαμε. Στη διαδικασία που περιγράφαμε παραπάνω έχουμε υπολογίσει λανθασμένα

το residue, αφού τελικά το predicted frame δεν περιλαμβάνει τις απώλειες του κβαντιστή. Ακριβώς λοιπόν επειδή ο αποκωδικοποιητής δεν θα έχει στη διάθεσή του άλλα σήματα, παρό μόνον το κβαντισμένο σφάλμα πρόβλεψης, δεν θα είναι σε θέση να αναπαράγει σωστά το ζητούμενο frame.

Άμεση απόρροια αυτής της παρατήρησης είναι η αλλαγή της πηγής του σήματος με βάση το οποίο θα γίνει η πρόβλεψη. Αυτό σημαίνει, ότι το παρελθοντικό frame δε θα το πάρουμε απευθείας από την είσοδο του συστήματος, αλλά θα πρέπει να το ανακατασκευάσουμε, κάνοντας χρήση του σφάλματος πρόβλεψης. Το σήμα που αφορά το σφάλμα πρόβλεψης θα προέρχεται από την έξοδο του κβαντιστή ($\hat{e}(n_1, n_2, t)$) και θα τροφοδοτείται απευθείας σε έναν απο-κβαντιστή (de-quantizer) ενώ στη συνέχεια, θα υποστεί Αντιστρόφο Διακριτό Μετασχηματισμό Συνημιτόνου (Inverse DCT - IDCT). Τελικά θα προστεθεί με το αντίστοιχο predicted frame ($\hat{s}(n_1, n_2, t)$), ώστε να μας δώσει το ανακατασκευασμένο frame $s(n_1, n_2, t)$ (Σχήμα 2.3).

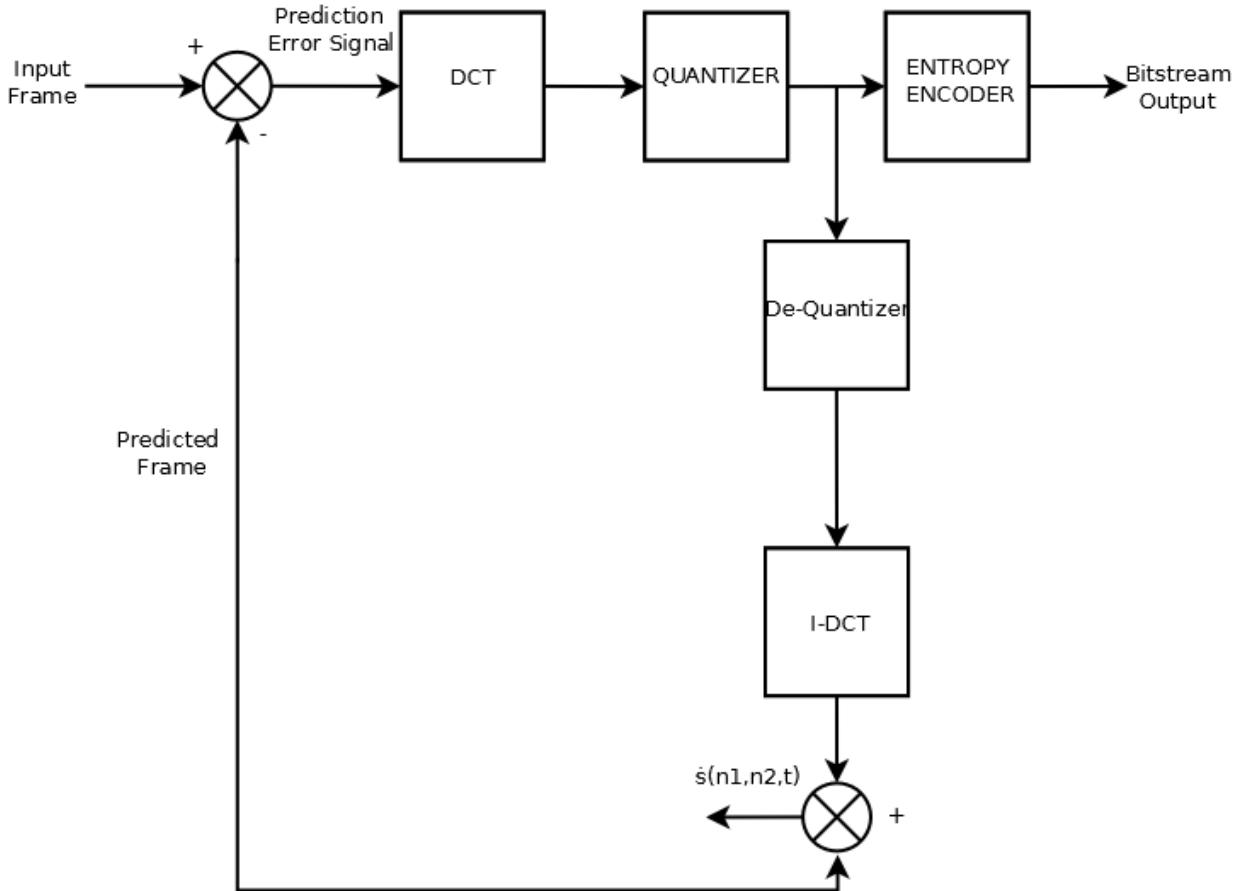
Αν το ανακατασκευασμένο frame το αποθηκεύσουμε σε μια μνήμη για μετέπειτα χρήση του, τότε αυτομάτως αυτό γίνεται παρελθοντικό και επομένως μπορούμε να το χρησιμοποιήσουμε για να προβλέψουμε ποιο θα είναι το επόμενο frame. Στην περίπτωση αυτή και για να είμαστε σύμφωνοι με τη σημειολογία που εισάγαμε προηγουμένως, θα αναφερόμαστε πλέον στο frame αυτό ως $\hat{s}(n_1, n_2, t-1)$ ενώ στο frame που μόλις δεχτήκαμε στην είσοδο ως $s(n_1, n_2, t)$.

Το τελευταίο κομμάτι του κωδικοποιητή είναι ο μηχανισμός πρόβλεψης. Είναι ίσως το σημαντικότερο κομμάτι του κωδικοποιητή αφού είναι αυτό που εκμεταλλεύεται των χρονικό πλεονασμό μεταξύ των frames προκειμένου να πετύχουμε περαιτέρω συμπίεση. Ο μηχανισμός πρόβλεψης αποτελείται από τρία επιμέρους τμήματα: τη μνήμη, η οποία περιέχει τα παρελθοντικά frames, τη μονάδα εκτίμησης κίνησης (motion estimation) και τη μονάδα πρόβλεψης (motion prediction). Για τη μνήμη έχουμε ήδη αναφέρει ορισμένα πράγματα και η ανάλυσή της μπορεί να ολοκληρωθεί εδώ αφού δεν εκτελεί κάποια περίπλοκη εργασία.

Στο σημείο αυτό μπορεί να γίνει μια αναφορά στα blocks μιας εικόνας (ή ενός frame). Πρόκειται ουσιαστικά για έναν ευρύτερο διαχωρισμό της εικόνας σε μεγαλύτερες περιοχές που περιλαμβάνουν (και ομαδοποιούν) πολλά pixels. Η λογική τους είναι παρόμοια με αυτή των οικοδομικών τετράγωνων (blocks) στις πόλεις, αφού έχουν ακριβώς την ίδια δομή πλέγματος. Συνήθως οι διαστάσεις των block (σε pixels) είναι δυνάμεις του 2 όπως, 4x4, 8x8, 8x4, 16x16, 16x8 κλπ. Στην εικόνα 2.4 φαίνεται ο διαχωρισμός μιας εικόνας σε block.

Η μόναδα εκτίμησης κίνησης έχει δύο εισόδους και μια έξοδο. Πρακτικά, η δουλειά της είναι να συγκρίνει το παρελθοντικό frame σε σχέση με το τρέχον που λάμβανουμε στην είσοδο του συστήματος και να υπολογίζει τη μετατόπιση που υπέστησαν τα blocks από το ένα frame στο επόμενο. Το αποτέλεσμα αυτής της σύγκρισης είναι ένα πλήθος διανυσμάτων που υποδεικνύουν τη μετατόπιση των blocks και ονομάζονται διανύσματα μετατόπισης (displacement vectors) ή διανύσματα κίνησης (motion vectors). Όταν τα διανύσματα

Σχήμα 2.3: Το διάγραμμα του συστήματος μετά την ανακατασκευή του frame

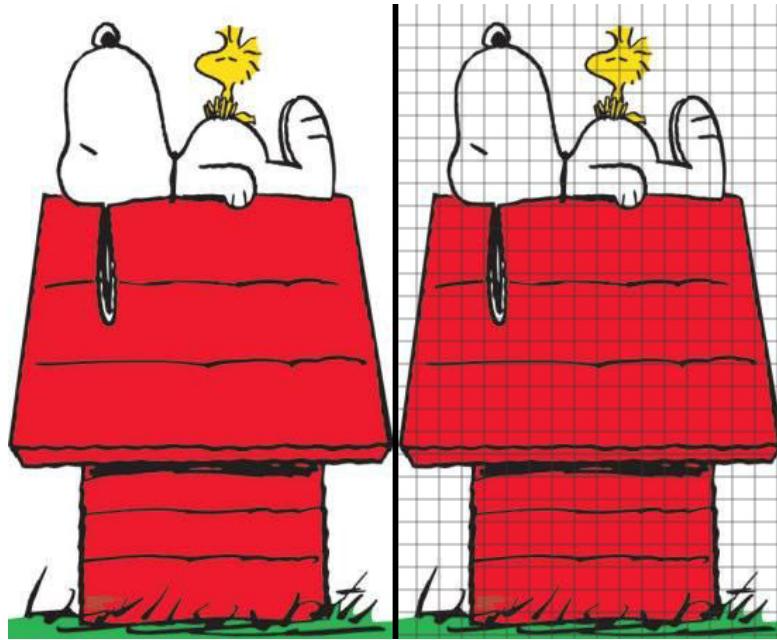


κίνησης εφαρμοστούν πάνω σ'ενα παρελθοντικό frame, τότε αυτό που προκύπτει είναι μια πρόβλεψη κίνησης που στην περίπτωσή μας είναι το predicted frame. Τα διανύσματα κίνησης θα τα δούμε αναλυτικά και θα μας απασχολήσουν στην επόμενη ενότητα αλλά και σε όλη την υπόλοιπη έκταση της εργασίας. Λάμβανοντας υπόψην τα όσα αναφέραμε περί του μηχανισμού πρόβλεψης, το τελικό σχεδιάγραμμα του κωδικοποιητή έχει ως εξής (Σχήμα 2.5):

2.5 ΕΚΤΙΜΗΣΗ ΚΙΝΗΣΗΣ (MOTION ESTIMATION)

Προηγουμένως αναφερθήκαμε στην εκτίμηση κίνησης και τη μονάδα που την υλοποιεί, σ'εναν τυπικό κωδικοποιητή βίντεο. Παραλείψαμε την αναφορά σε λεπτομέρειες και τη σε

Σχήμα 2.4: Διαχωρισμός εικόνας σε block

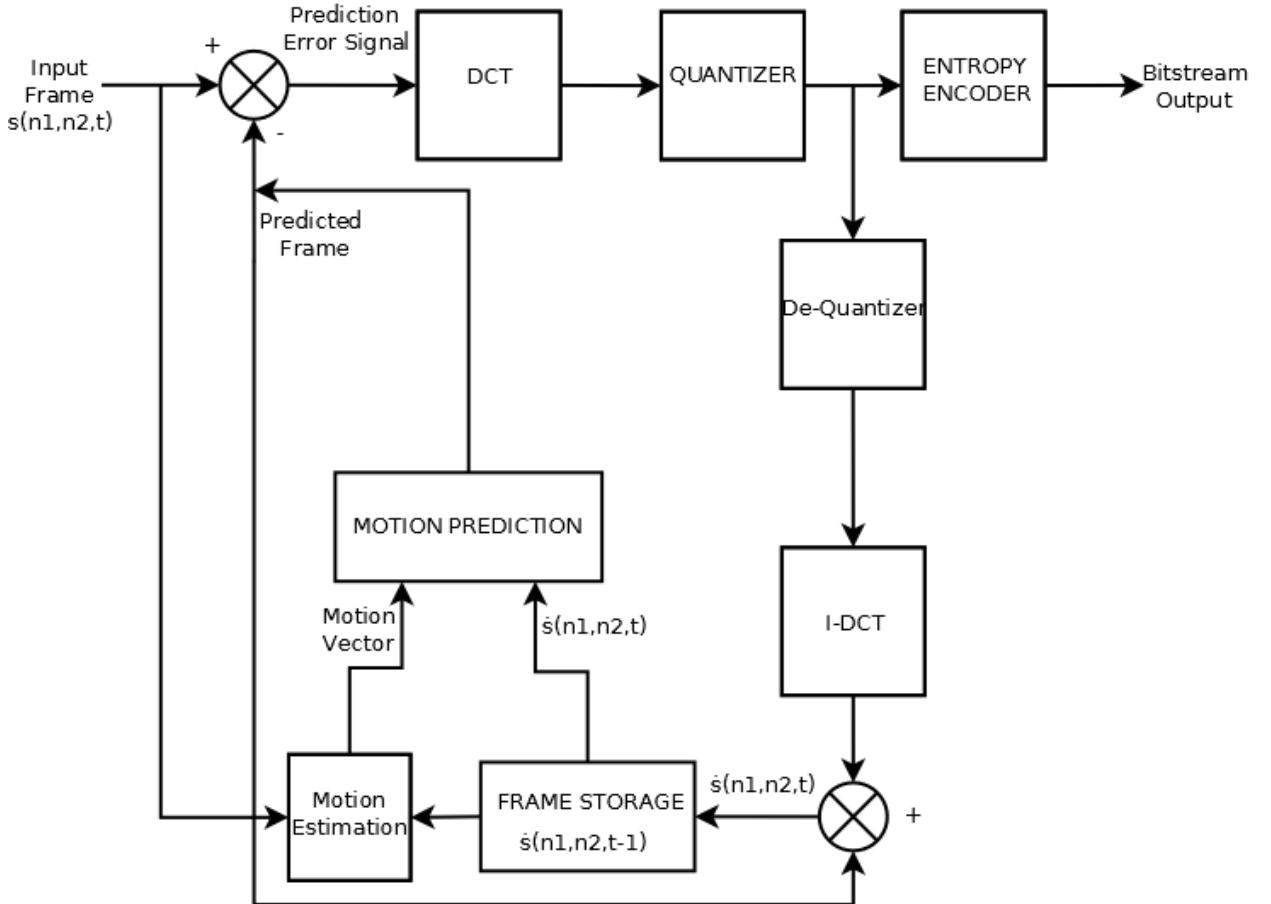


βάθος ανάλυση για δύο λόγους: πρώτον, διότι πρόκειται για ένα πολύ σημαντικό κομμάτι το οποίο είναι σκόπιμο να αναλυθεί ξεχωριστά και να καλυφθεί πλήρως, και δεύτερον, επειδή πρόκειται για τον πυρήνα της εργασίας μας, για τον οποίο μάλιστα έγινε η εισαγωγή στις βασικές αρχές της κωδικοποίησης βίντεο.

Αναφέραμε πολλές φορές ότι μεταξύ διαδοχικών frames υπάρχει πολύ μεγάλος συσχετισμός, διότι είναι εκ των πραγμάτων πολύ περιορισμένος ο αριθμός των συμβάντων που μπορούν να καταγραφούν μέσα στο χρονικό παράθυρο των 33ms (για τυπικούς ρυθμούς 30 καρέ/δευτερόλεπτο). Ακριβώς γι' αυτό το λόγο αν συγκρίνουμε δύο διαδοχικά frame $t-1$ και t θα διαπιστώσουμε ότι σε συγκεκριμένες περιοχές η ένταση (intensity) των pixel παραμένει η ίδια, ενώ σε άλλες αλλάζει. Η πρώτη περίπτωση ισχύει σε περιοχές του frame όπου δεν υπάρχει κίνηση, ή η κίνηση είναι ανεπαίσθητη, τόσο που να μην είναι ορατή από frame σε frame (για παράδειγμα ένα στατικό φόντο). Αντίθετα, στα σημεία του frame όπου εντοπίζεται κίνηση (για παράδειγμα κινήσεις χεριών) οι αλλαγές στην ένταση μπορεί να είναι μεγάλες.

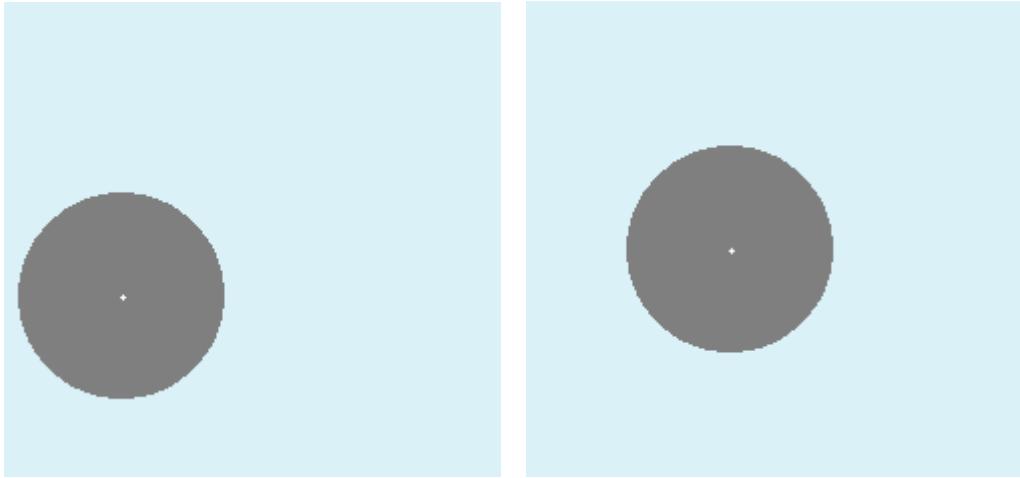
Αν υποθέσουμε ότι έχουμε δύο διαδοχικά frames στα οποία παρουσιάζεται η κίνηση ενός απλού αντικειμένου (σχήμα 2.6) η κίνηση αυτή, εφόσον είναι μεταφορική, μπορεί να περιγραφεί από δύο συνιστώσες, μία κατά τον άξονα x και μία κατά τον άξονα y . Παραμένοντας συνεπείς στην προηγούμενη σημειολογία, θεωρούμε τις δύο διαστάσεις ως n_1 και n_2 , οπότε τελικά η μεταφορική αυτή κίνηση μπορεί να οριστεί ως ένα δισδιάστατο διάνυσμα $\mathbf{d} = (d_1, d_2)^T$ όπου d_1 η μετατόπιση ως προς n_1 και d_2 η μετατόπιση ως προς n_2 .

Σχήμα 2.5: Τελικό διάγραμμα του κωδικοποιητή



Το διάνυσμα αυτό ονομάζεται **διάνυσμα μετατόπισης** (displacement vector) ή **διάνυσμα κίνησης** (motion vector). Συγχρίνοντας τα δύο frame στο σχήμα 2.6, θα παρατηρήσουμε ότι υπάρχουν περιοχές όπου η μετατόπιση είναι μηδενική και άλλες όπου είναι μη μηδενική. Πως όμως μπορεί να γίνει μια τέτοια σύγκριση; Μια συνήθης τακτική είναι η **αντιστοίχιση** (matching) του προηγούμενου frame με το επόμενο. Υπάρχουν αρκετοί τρόποι για να γίνει αυτή η αντιστοίχιση και μια απλή σκέψη θα ήταν να γίνει pixel προς pixel. Με αυτό τον τρόπο θα αντιστοιχίζαμε την τιμή κάθε ενός pixel του πρώτου frame με το δεύτερο, μέχρι να βρούμε όλες τις αλλαγές που συνέβησαν. Κάτι τέτοιο ωστόσο εκτός από εξαιρετικά χρονοβόρο δεν είναι καθόλου πρακτικό, διότι με το να συγχρίνουμε την ένταση ενός pixel με την ένταση του αντίστοιχου pixel ενός άλλου frame δε μας εξασφαλίζει ότι όντως θα υπάρχει αντιστοίχιση μεταξύ τους, αφού λόγω σφάλματος ή τύχης μπορεί να έχουν την

Σχήμα 2.6: Κίνηση ενός απλού γεωμετρικού σχήματος ανάμεσα σε δύο διαδοχικά frames



ίδια ακριβώς τιμή. Για το λόγο αυτό ελέγχουμε όχι μόνο την αντιστοίχιση μεταξύ δύο pixel αλλά και μεταξύ των γειτονικών τους. Μόνο όταν δύο τέτοιες "γειτονιές" μεταξύ δύο frame ταιριάζουν μεταξύ τους, μπορούμε να ισχυριστούμε με σιγουριά ότι έγινε σωστά η αντιστοίχιση.

Ένα σημαντικό ζήτημα είναι ο τρόπος με τον οποίο θα καθοριστούν αυτές οι "γειτονιές" με βάση τα γεωμετρικά τους χαρακτηριστικά. Μπορούμε να θεωρήσουμε επι παραδείγματι, αυθαίρετα, κυκλικες, πολύγωνες ή τετράγωνες περιοχές. Η απλούστερη εκδοχή είναι αυτή των τετράγωνων ή/και ορθογώνιων περιοχών. Μάλιστα, αφού είμαστε ήδη εξοικιωμένοι με το διαμερισμό μιας εικόνας σε μικρότερα μη-αλληλοεπικαλυπτόμενα block μπορούμε να εφαρμόσουμε ακριβώς την ίδια φιλοσοφία και εδώ, να χωρίσουμε δηλαδή την εικόνα σε διαχριτά block και τελικά να υπολογισουμε τη μετατόπιση που υπέστη κάθε block (και συνεπώς τα διανύσματα κίνησης). Η διαδικασία εύρεσης του βαθμού μετατόπισης είναι ουσιαστικά μια **τεχνική αναζήτησης** η οποία είναι περισσότερο γνωστή ως **εκτίμηση κίνησης (motion estimation)**. Πλέον είναι πιο ξεκάθαρος ο τρόπος με τον οποίο λειτουργεί η μονάδα εκτίμησης κίνησης που περιγράφαμε παραπάνω.

Από την πλευρά του αποκωδικοποιητή, τα διανύσματα κίνησης είναι απολύτως απαραίτητη πληροφορία αφού θα χρησιμοποιηθούν πάνω σε παρελθοντικά (ή μελλοντικά, ανάλογα τη μορφή κωδικοποίησης) frame προκειμένου να αναπαράγουν τα ζητούμενα frames κατά την αποκωδικοποίηση του βίντεο. Συνεπώς, είναι απαραίτητο να εφαρμόσουμε την κωδικοποίηση εντροπίας και σε αυτά ώστε να συμπεριληφθούν στο τελικό bitstream (την έξοδο του κωδικοποιητή).

Όπως είναι εύκολα αντιληπτό, η εκτίμηση κίνησης οφείλει να είναι μια γρήγορη, ακριβής και αποτελεσματική διαδικασία. Ο λόγος για τον οποίο πρέπει να έχει αυτά τα χαρακτηριστικά είναι ο πολύ μεγάλος όγκος πληροφοριών που καλείται να επεξεργαστεί, συχνά σε πραγματικό

χρόνο. Κατά την εκτίμηση κίνησης θα πρέπει να βρούμε τα διανύσματα κίνησης για κάθε ένα block του frame και δεν γνωρίζουμε εκ των προτέρων προς ποια κατεύθυνση έχει κινηθεί αυτό το συγκεκριμένο block. Η κατάσταση περιπλέκεται ακόμη περισσότερο αν εισάγουμε και άλλες μορφές κίνησης εκτός από τη μεταφορική, π.χ περιστροφική. Προκειμένου να κάνουμε την αντιστοίχιση θα πρέπει να προχωρήσουμε με έναν **αλγόριθμο αναζήτησης** και η αναζήτηση είναι εκ των πραγματών, μια υπολογιστικά πολύπλοκη διαδικασία. Στην περίπτωση ενος frame που έχει διαμεριστεί σε blocks των 8x8 pixel, για κάθε θέση αναζήτησης (block), θα πρέπει να συκρίνουμε τις τιμές 64 pixel. Φυσικά, όπως αναφέραμε ήδη πολλές φορές, η κίνηση μεταξύ γειτονικών frames είναι αρκετά περιορισμένη επομένως η αναζήτηση αυτή δεν είναι υποχρεωτικό να λάβει χώρα σε ολοκληρωτό το frame αλλά μόνο σε μια περιοχή του. Αυτή την περιοχή την ονομάζουμε **περιοχή αναζήτησης (search area)**, και αν και σημαντικά μικρότερη σε έκταση, η αναζήτηση σε αυτήν εξακολουθεί να αποτελεί υπολογιστική πρόκληση.

Κατα τη διαδικασία της αναζήτησης αυτό που προσπαθούμε να πετύχουμε είναι για κάθε ένα block του candidate frame (candidate block) να βρούμε με ποιό block του reference frame (reference block) έχει την καλύτερη αντιστοίχιση. Όπως αναφέραμε και προηγουμένως, η αναζήτηση δε γίνεται σε όλα τα blocks του frame αλλά σε μια μικρότερη περιοχή αναζήτησης. Αυτό που είναι προφανές είναι πως για να μπορέσουμε να είμαστε ακριβείς ως προς την αντιστοίχιση δύο block, πρέπει πρώτα να ορίσουμε ένα μέτρο, ένα **κριτήριο ταύτισης**. Στην εκτίμηση κίνησης βασισμένη σε block (**block-based motion estimation**) τα κριτήρια είναι τρία:

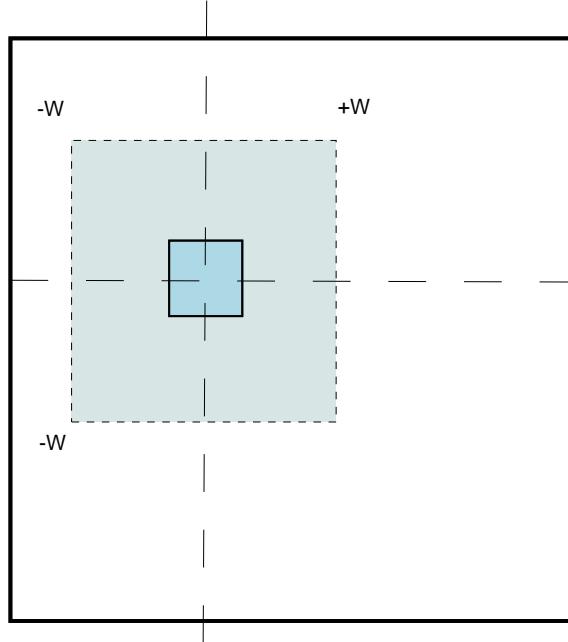
- Μέσο Τετραγωνικό Σφάλμα (Mean Square Error ή **MSE**)
- Μέση Απόλυτη Διαφορά (Mean of Absolute Difference ή **MAD**)
- Αριθμός Αντιστοιχισμένων Pixel (Matching Pixel Count ή **MPC**)

2.5.1 ΜΕΣΟ ΤΕΤΡΑΓΩΝΙΚΟ ΣΦΑΛΜΑ

Ας υποθέσουμε και πάλι ένα candidate frame με συνάρτηση έντασης $s(n_1, n_2, t)$ και ένα reference frame με συνάρτηση έντασης $s(n_1, n_2, t - l)$, $l > 0$. Το reference frame θα είναι παρελθοντικό, αφού το l είναι μεγαλύτερο του μηδενός. Ας υποθέσουμε επίσης ένα candidate block για το οποίο θέλουμε να αναζητήσουμε ποια ακριβώς ήταν η θέση του στο reference frame. Για το λόγο αυτό ορίζουμε μια περιοχή αναζήτησης γύρω από το block, με διαστάσεις $\pm w$ pixel.

Προκειμένου να καλύψουμε ολόκληρη την περιοχή αναζήτησης, θα πρέπει να φάξουμε συνολικά σε $(2w+1) \cdot (2w+1)$ θέσεις. Ο αριθμός w πρακτικά, δεν μπορεί να είναι πολύ μικρός ούτε πολύ μεγάλος. Ας μην ξεχνάμε ότι η κίνηση των φυσικών αντικειμένων είναι πολύ

Σχήμα 2.7: Το candidate frame στο οποίο φαίνεται με γαλάζιο χρώμα το candidate block και η περιοχή αναζήτησης που το περιβάλλει



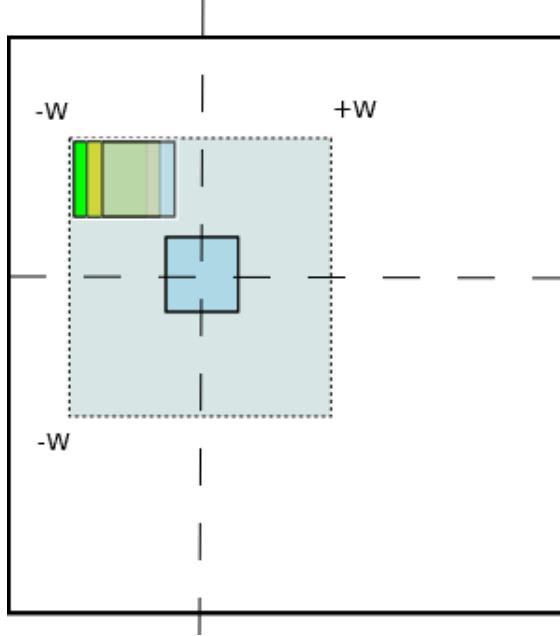
περιορισμένη στο χρονικό παράθυρο των 33ms (για βίντεο 30 FPS), επομένως μια ρεαλιστική τιμή θα ήταν $w = 7$ ή $w = 8$. Αν και εκ πρώτης όψεως μοιάζει μικρός αριθμός, ο συνολικός αριθμός των θέσεων αναζήτησης που προκύπτουν είναι: $(2w+1)^2 = (2 \cdot 7 + 1)^2 = 225$ και απαιτούνται για την εκτίμηση της κίνησης ενός μόνο block! Αν συνυπολογίσουμε το γεγονός ότι η πρόβλεψη κίνησης πρέπει να γίνεται σε πραγματικό χρόνο, τότε είναι περισσότερο από προφανής η ανάγκη για γρήγορους χρόνους επεξεργασίας.

Ο τρόπος με τον οποίο πραγματοποιούμε την αναζήτηση είναι εφαρμόζοντας μια "τεχνητή" μετατόπιση του candidate block μέσα στην περιοχή αναζήτησης και ελέγχοντας βάσει των κριτηρίων που προαναφέραμε τη συσχέτιση με την τρέχουσα θέση (το reference block), "σαρώνοντας" πρακτικά όλη την περιοχή αναζήτησης pixel-προς-pixel.

Η επεξεργαστική ισχύς καταναλώνεται σχεδόν αποκλειστικά στην εφαρμογή των κριτηρίων ταύτισης και μία περίπτωση είναι η χρήση της μεθοδολογίας του Μέσου Τετραγωνικού Σφάλματος ή MSE. Αν θεωρήσουμε ως (i,j) τη μετατόπιση του candidate block μέσα στην περιοχή αναζήτησης, ο μαθηματικός ορισμός του MSE έχει ως εξής:

$$MSE(i,j) = \frac{1}{N^2} \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} [s(n_1, n_2, k) - s(n_1 + i, n_2 + j, k - l)]^2 \quad (2.1)$$

Σχήμα 2.8: Η διαδικασία σάρωσης της περιοχής αναζήτησης. Διακρίνονται οι τρεις πρώτες θέσεις αναζήτησης.



όπου N ή διάσταση του block (στην προκειμένη περίπτωση $N \times N$ είναι οι διαστάσεις του block μας)

Κατά την αναζήτηση μέσα στην περιοχή που ορίσαμε, η θέση (i, j) μεταβάλλεται συνεχώς. Για κάθε νέα θέση κάνουμε μια σύγκριση, βρίσκοντας στην πράξη τη διαφορά μεταξύ των τιμών έντασης των pixel του candidate block με κάθε ένα reference block. Επειδή όμως αυτή η διαφορά μπορεί να είναι είτε θετική είτε αρνητική, αυτό μπορεί να οδηγήσει σε απροσδόκητα αποτελέσματα όπως για παράδειγμα να εμφανιστεί η συνολική διαφορά μεταξύ των blocks ως μηδενική, αν για παράδειγμα οι επιμέρους διαφορές των pixel έχουν τέτοιες αρνητικές και θετικές τιμές που να ευνοούν ένα τέτοιο αποτέλεσμα. Αυτός είναι και ο ρόλος που παίζει το τετράγωνο στη σχέση, αφού απαλείφει ουσιαστικά τις αρνητικές τιμές. Η παραπάνω μαθηματική σχέση δηλαδή, μεταφράζεται ως η μέση τιμή του αθροίσματος των διαφορών μεταξύ του candidate και κάθε ενός reference block υψηλόν στο τετράγωνο. Αυτό που επιδιώκουμε με τη χρήση του MSE είναι να εντοπίσουμε το reference block με τη μικρότερη δυνατή διαφορά, δηλαδή, την ελάχιστη τιμή σφάλματος. Αφού εντοπίσουμε ποιο είναι αυτό το block, κατασκευάζουμε το διάνυσμα κίνησης $\mathbf{d} = (d_1, d_2)$, με d_1 (αρχή) το κέντρο του candidate block και d_2 (πέρας) το κέντρο του reference block με το μικρότερο MSE.

Μια συνηθισμένη περίπτωση είναι να εκτελούμε την αναζήτηση με $N = 8$, επομένως τα

block θα έχουν διαστάσεις 8x8 pixel. Σε αυτήν την περίπτωση ο υπολογισμός του συνολικού αθροίσματος θα περιλαμβάνει 64 προσθέσεις (N^2) αλλά και 64 πολλαπλασιασμούς, λόγω του τετραγώνου. Ο πολλαπλασιασμός ως υπολογιστική πράξη είναι ιδιαίτερα "προβληματικός" αφού όταν γίνεται σε επίπεδο software καταναλώνει αρκετό χρόνο ενώ σε επίπεδο hardware είναι ένα σχετικά κοστοβόρο κύκλωμα. Προκειμένου να έχουμε ένα κριτήριο ταύτισης που δεν περιλαμβάνει κάποιο πολλαπλασιασμό η ίδια φιλοσοφία μπορεί να εφαρμοστεί αντικαθιστώντας τον τετραγωνισμό με την απόλυτη τιμή της διαφοράς.

2.5.2 ΑΠΟΛΥΤΗ ΤΙΜΗ ΔΙΑΦΟΡΑΣ

Η ελάχιστη απόλυτη τιμή της διαφοράς δύο block είναι το άλλο κριτήριο που μπορούμε να χρησιμοποιήσουμε προκειμένου να συγχρίνουμε το candidate block με κάθε ένα από τα reference blocks και να αποφανθούμε για τη βέλτιστη αντιστοίχιση. Το πλεονέκτημα αυτού του κριτηρίου είναι η "απουσία" του πολλαπλασιασμού, πράγμα που βελτιώνει σημαντικά την απόδοση της οποιασδήποτε υλοποίησης ενός αλγόριθμου εκτίμησης κίνησης.

$$MAD(i, j) = \frac{1}{N^2} \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} |s(n_1, n_2, k) - s(n_1 + i, n_2 + j, k - l)| \quad (2.2)$$

Η χρήση του MAD ως κριτήριο ταύτισης είναι ιδιαίτερα διαδεδομένη στους περισσότερους μηχανισμούς εκτίμησης κίνησης λόγω της σχετικά απλής υλοποίησης η οποία περιλαμβάνει μόλις 64 προσθέσεις και αφαιρέσεις.

2.5.3 ΑΡΙΘΜΟΣ ΑΝΤΙΣΤΟΙΧΙΣΜΕΝΩΝ PIXEL

Στην περίπτωση αυτού του κριτηρίου η προσέγγιση που ακολουθείται είναι διαφορετική. Η βασική μετρική που χρησιμοποιείται είναι ο αριθμός των pixel μεταξύ δύο block που θεωρούνται αντιστοιχισμένα. Πράκτικά, ορίζουμε ένα **όριο τιμής (value threshold)** ϑ , κάτω από το οποίο θεωρούμε ότι δύο pixel έχουν κοντινή συσχέτιση και άρα μπορούν να προσμετρηθούν ως αντιστοιχισμένα (να αυξήσουμε δηλαδή έναν μετρητή κατά 1). Σε αντίθετη περίπτωση το τρέχον pixel που ελέγχουμε δεν προσμετράται και άρα δεν αυξάνουμε την τιμή του μετρητή. Όπως είναι εύκολα αντιληπτό, στο συγκεκριμένο κριτήριο μας ενδιαφέρει η μέγιστη τιμή του μετρητή η οποία υποδηλώνει και τη μέγιστη ταύτιση. Η μαθηματική έκφραση του κριτηρίου συνοψίζεται στις ακόλουθες σχέσεις:

$$count(n_1, n_2) = \begin{cases} 1, & \text{αν } |s(n_1, n_2, k) - s(n_1 + i, n_2 + j, k - l)| \leq \vartheta \\ 0, & \text{οπουδήποτε άλλο} \end{cases} \quad (2.3)$$

$$MPC(i, j) = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} count(n_1, n_2) \quad (2.4)$$

Η σχέση (2.3) είναι μία δυαδική συνάρτηση η οποία παίρνει την τιμή 1 αν η διαφορά βρίσκεται εντός του ορίου τιμής που θέσαμε, ενώ σε αντίθετη περίπτωση παίρνει την τιμή 0. Η σχέση (2.4) αποτελεί το συνολικό άθροισμα των τιμών της (2.3) για κάθε pixel του block. Είναι προφανές ότι η μέγιστη τιμή καθορίζεται από τη μεταβλητή N και συνεπώς τις διαστάσεις του block. Στην περίπτωση που εξετάσαμε προηγουμένως, αν δηλαδή έχουμε block των 8×8 pixel, η μέγιστη τιμή του μετρητή θα είναι 64 ενώ η ελάχιστη 0.

2.6 ΑΛΓΟΡΙΘΜΟΙ ΕΚΤΙΜΗΣΗΣ ΚΙΝΗΣΗΣ

Η εκτίμηση κίνησης είναι ένα πρόβλημα που χαρακτηρίζεται από μεγάλη πολυπλοκότητα, ενώ η λύση του μπορεί να προσεγγιστεί με πολλές και διαφορετικές τεχνικές. Είναι επόμενο λοιπόν να έχει αναπτυχθεί ένα πλήθος τεχνικών και στρατηγικών εκτίμησης κίνησης βασισμένων σε διαφορετικά χαρακτηριστικά του προβλήματος.

2.6.1 ΕΞΑΝΤΛΗΤΙΚΗ ΑΝΑΖΗΤΗΣΗ (FULL-SEARCH BLOCK MOTION ESTIMATION ή FSBME)

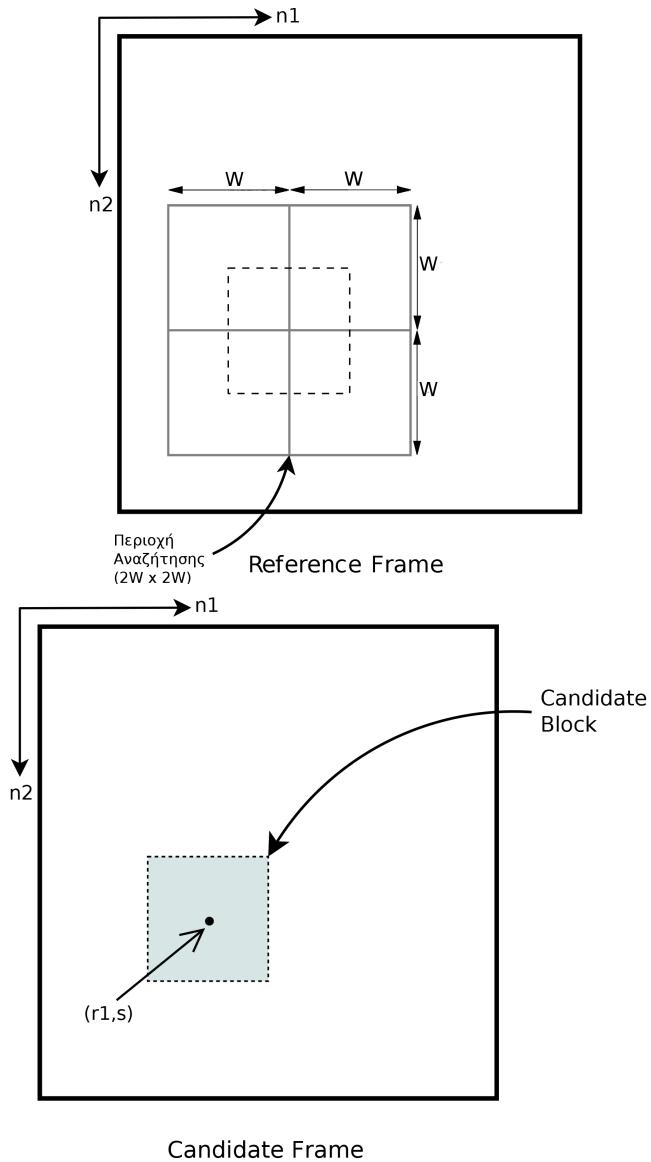
Ο αλγόριθμος ο οποίος θα μας απασχολήσει σε όλη την υπόλοιπη έκταση της εργασίας, είναι αυτός για τον οποίο ήδη έχουμε πει κάποια πράγματα, καθώς είναι ο πιο απλός και αποτελεσματικός αλγόριθμος που μπορεί να χρησιμοποιηθεί. Το βασικό του μειονέκτημα είναι ο εξαιρετικά μεγάλος χρόνος εκτέλεσης, αφού εκτελεί υπολογισμούς πάνω σε κάθε pixel του candidate block και της περιοχής αναζήτησης.

Θεωρούμε ένα block $N \times N$ pixel στο candidate frame (το candidate block) στη θέση (r_i, s) όπως φαίνεται στο σχήμα 2.9. Θεωρούμε στη συνέχεια μια περιοχή αναζήτησης στο reference frame (search window) γύρω από τη θέση του αντίστοιχου block, με διαστάσεις έντονος $\pm w$ pixel και προς τις δύο κατευθύνσεις n_1 και n_2 . Για κάθε μία από τις $(2w + 1)^2$ θέσεις αναζήτησης το candidate block συγχρίνεται με ένα block ίδιων διαστάσεων $N \times N$ pixel, σύμφωνα με τα κριτήρια που συζητήσαμε στην προηγούμενη ενότητα και το block με τη μεγαλύτερη ταύτιση, μαζί με το διάνυσμα κίνησης, καθορίζονται μόνο αφού ολοκληρωθεί η διαδικασία της εξαντλητικής αναζήτησης και ελεγχθούν όλες οι πιθανές θέσεις.

Το ιδανικό μέγεθος για μια περιοχή αναζήτησης εξαρτάται από πολλές παραμέτρους:

- Την ανάλυση της κάθε εικόνας (ένα μεγάλο παράθυρο είναι προφανώς κατάλληλο για μια πολύ μεγάλη ανάλυση).
- Τον τύπο της σκηνής (σκηνές με γρήγορη κίνηση συνήθως οφελούνται με ένα μεγάλο παράθυρο αναζήτησης σε αντίθεση με τις σκηνές που έχουν μικρές κινήσεις).
- Τους υπολογιστικούς πόρους που είναι διαθέσιμοι, καθώς ένα μεγάλο παράθυρο αναζήτησης απαιτεί περισσότερες συγκρίσεις και άρα περισσότερους υπολογισμούς.

Σχήμα 2.9: Δυο διαδοχικά frames



Ο αλγόριθμος FSBM είναι η βέλτιστη δυνατή λύση στο πρόβλημα της εκτίμησης κίνησης, κυρίως διότι, αν οριστεί σωστά η περιοχή αναζήτησης, μας εγγυάται ότι θα εντοπίσει εκείνο το block με τη μεγαλύτερη ταύτιση. Ωστόσο όπως έχουμε ήδη αναφέρει πολλές φορές είναι ιδιαίτερα δαπανηρός από πλευράς κατανάλωσης υπολογιστικής ισχύος. Για κάθε θέση αναζήτησης απαιτούνται $O(N^2)$ υπολογισμοί (προσθέσεις, αφαιρέσεις, κλπ) και εφόσον

Πίνακας 2.2: Υπολογιστική πολυπλοκότητα του αλγορίθμου FSBM

Κριτήριο ταύτισης	Απαιτούμενοι υπολογισμοί		
	Προσθέσεις/ Αφαιρέσεις	Πολλ/σμοί	Συγκρίσεις
MSE	$(2N^2 - 1)(2w + 1)^2$	$N^2(2w + 1)^2$	$(2w + 1)^2$
MAD	$(2N^2 - 1)(2w + 1)^2$	-	$(2w + 1)^2$
MPC	$(2N^2 - 1)(2w + 1)^2$	-	$N^2(2w + 1)^2$

υπάρχουν $(2w + 1)^2$ θέσεις ο αριθμός των υπολογισμών σε σχέση με το κριτήριο ταύτισης που χρησιμοποιείται απεικονίζεται στον πίνακα (2.2).

Ο μεγάλος αριθμός υπολογισμών που απαιτούνται κατά την εξαντλητική αναζήτηση οδήγησε στην ανάπτυξη νέων τεχνικών που προσεγγίζουν το πρόβλημα της εκτίμησης κίνησης πολύ γρηγορότερα. Συνήθως όμως, επειδή τα αποτελέσματα αυτών των τεχνικών δεν είναι τα βέλτιστα, ανεχόμαστε κάποιο βαθμό σφάλματος. Στις επόμενες ενότητες θα εξετάσουμε επιγραμματικά μερικές από αυτές τις τεχνικές αναζήτησης.

2.6.2 ΑΝΑΖΗΤΗΣΗ ΤΡΙΩΝ ΒΗΜΑΤΩΝ (THREE STEP SEARCH)

Στον αλόριθμο αναζήτησης τριών βημάτων[19], η πρώτη επανάληψη αξιολογεί 9 υποψήφια σημεία συμπεριλαμβανομένου του κέντρου. Το αρχικό βήμα είναι ίσο ή λίγο μεγαλύτερο από το μισό της περιοχής αναζήτησης. Με το πέρας της επαναλήψεως και τον υπολογισμό του σφάλματος για τα εννιά υποψήφια σημεία, το κέντρο αναζήτησης μετατοπίζεται στο σημείο με το μικρότερο σφάλμα και μειώνεται το βήμα στο μισό. Η ίδια διαδικασία επαναλαμβάνεται μέχρι το βήμα να γίνει ίσο με ένα εικονοστοιχείο. Στην περίπτωση αυτή, το σημείο με το μικρότερο σφάλμα θα επιλεγεί ως το βέλτιστο αποτέλεσμα και το διάνυσμα κίνησης που αντιστοιχεί σε αυτό επιλέγεται για το τρέχον μπλοκ. Ο αριθμός των σημείων που αξιολογούνται κατά τη διάρκεια της αναζήτησης είναι πολύ μικρότερος από αυτών της εξαντλητικής μεθόδου και καθορίζεται από το μέγεθος του βήματος που τίθεται στην πρώτη επανάληψη. Βασικά πλεονεκτήματα του αλγορίθμου είναι τα εξής:

- Μικρή πολυπλοκότητα
- Χαμηλός χρόνο αναζήτησης
- Αρκετά καλά αποτελέσματα και συνέπεια

Βασικά μειονεκτήματα του αλγορίθμου είναι τα εξής:

- Ο παράγοντας της πολυπλοκότητας μεγαλώνει ανάλογα με την περιοχή αναζήτησης

- Ακατάλληλος για μικρές κινήσεις
- Χρησιμοποιείται χυρίως σε εφαρμογές που απαιτούν χαμηλό ρυθμό bit όπως τηλεδιασκέψεις και τηλέφωνα με δυνατότητα βίντεοκλήσης.

2.6.3 DIAMOND SEARCH

Η λογική του συγκεριμένου αλγορίθμου[31][32] είναι αρκετά απλή και βασίζεται στην αναζήτηση με βάση την επιλογή σημείων σε μια διάταξη διαμαντιού. Πιο συγκεκριμένα, ο αλγόριθμος ξεκινάει δημιουργώντας μια διάταξη διαμαντιού επιλέγοντας 9 σημεία. Στο επόμενο βήμα, αφού βρεθεί και επιλεγεί το σημείο με το μικρότερο σφάλμα, επαναλαμβάνεται η ίδια διαδικασία με κέντρο το επιλεγμένο αυτό σημείο. Όταν το σημείο που επιλεγεί είναι το κέντρο της διάταξης, τότε ο αλγόριθμος επιλέγει 5 γειτονικά σημεία δημιουργώντας και πάλι μια διάταξη διαμαντιού, μικρότερου φυσικά από το προηγούμενο, και βρίσκει και πάλι το σημείο με το μικρότερο σφάλμα ανάμεσα σε αυτά. Αυτό, λοιπόν, το σημείο είναι που φάχνουμε και αποτελεί το βέλτιστο ταίρι. Βασικά πλεονεκτήματα του αλγορίθμου είναι τα εξής:

- Καλύτερη απόδοση και ταχύτητα από τον αλγόριθμο αναζήτησης των τριών βημάτων
- Μεγάλη μέγιστη σήματος-προς-θόρυβο αναλογία

Βασικά μειονεκτήματα του αλγορίθμου είναι τα εξής:

- Δεν υπάρχει περιορισμός στα βήματα αναζήτησης
- Δεν υπάρχει αποτελεσματική υλοποίηση του σε hardware

Στο σχήμα φαίνεται ένα παράδειγμα αναζήτησης του αλγορίθμου οπού με κύκλο, τετράγωνο, ρόμβο και τρίγωνο, συμβολίζονται σημεία που ανήκουν σε διάταξη μεγάλου διαμαντιού αντίστοιχα για κάθε σχήμα. Με τον κύκλο χρώματος γκρι, συμβολίζεται η διάταξη μικρού διαμαντιού. Βλέπουμε ότι ο αλγόριθμος χρειάζεται πέντε βήματα για να καταλήξει σε ένα διάνυσμα κίνησης με συντεταγμένες (-4, -2). Τέσσερα από τα βήματα είναι με διάταξη αναζήτησης μεγάλου διαμαντιού και ένα από αυτά, το τελευταίο, με διάταξη αναζήτησης μικρού διαμαντιού.

2.6.4 ΔΙΣΔΙΑΣΤΑΤΗ ΛΟΓΑΡΙΘΜΙΚΗ ΑΝΑΖΗΤΗΣΗ (2-D LOGARITHMIC SEARCH)

Σ' αυτήν την τεχνική που αρχικά προτάθηκε από τους Jain και Jain θεωρούμε μια αρχική περιοχή αναζήτησης με μια υποθετική βέλτιστη θέση στο κέντρο της. Η περιοχή αναζήτησης εκτείνεται προς τις τέσσερις κατευθύνσεις κατά p pixel, επομένως έχει εμβαδό $4p^2$ pixel.

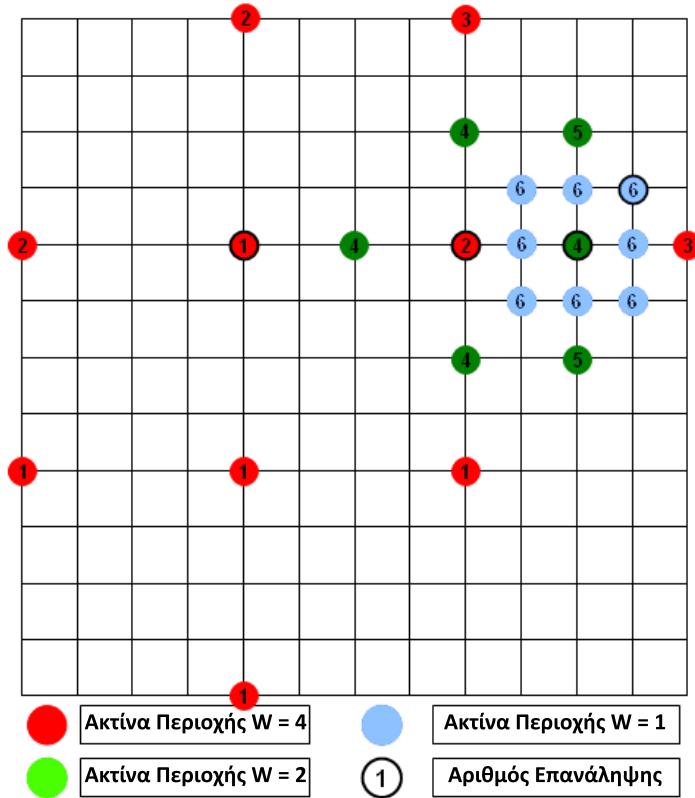
Αυτή η αρχική βέλτιστη θέση μπορεί να θεωρηθεί και ως η θέση του candidate block από τη σκοπιά του candidate frame. Αντί για την αναζήτηση σε όλες τις πιθανές θέσεις της περιοχής αναζήτησης, η αναζήτηση πραγματοποιείται μόνο σε πέντε θέσεις, την αρχική και τέσσερις γειτονικές της σε απόσταση $p/2$ από αυτήν (μία σε κάθε πλευρά του τετραγώνου), κατα τέτοιο τρόπο ώστε τα κέντρα τους να σχηματίζουν μια δομή με σχήμα σταυρού (σχήμα ??). Το $p/2$ ονομάζεται και βήμα και στο εξής μπορούμε να το συμβολίζουμε με s .

Η εκτέλεση του αλγόριθμου περιλαμβάνει τα εξής βήματα:

- Στην πρώτη επανάληψη του αλγορίθμου εξετάζουμε την πρώτη "πεντάδα" θέσεων, την αρχική υποθετική βέλτιστη, και τις τέσσερις γειτονικές της.
- Αν κάποια από τις τέσσερις γειτονικές θέσεις αποτελεί βέλτιστη λύση τότε θέτουμε αυτήν ως "κεντρική" θέση και ορίζουμε ξανά τέσσερις γειτονικές της. Μοιραία, τουλάχιστον μία από τις νέες γειτονικές θέσεις θα έχει εξεταστεί και στο παρελθόν, επομένως πλεύνεις λιγότερες θέσεις προς εξέταση. Εκτελούμε ξανά τον έλεγχο με τις νέες θέσεις.
- Αν σε κάποια επανάληψη του αλγορίθμου εντοπίσουμε ως βέλτιστη θέση αυτήν που ορίσαμε ως κεντρική και όχι κάποια γειτονική της, τότε μειώνουμε τον αριθμό s στο μισό (κατ'επέκταση υποτετραπλασιάζουμε το μέγεθος της θέσης) και εκτελούμε την επόμενη επανάληψη του αλγόριθμου. Αυτή τη φορά θέτουμε ξανά την ίδια θέση ως κεντρική όμως ορίζουμε νέες γειτονικές θέσεις σε απόσταση $s' = s/2$ από αυτήν.
- Ο αλγόριθμος τερματίζει όταν το s γίνει ίσο με 1, οπότε πλέον ελέγχουμε και τις 9 γειτονικές θέσεις που περιβάλλουν την τελευταία βέλτιστη που βρήκαμε. Από αυτή την τελευταία αναζήτηση προκύπτει η τελική βέλτιστη θέση και τελικά το διάνυσμα κίνησης.

Στο σχήμα 2.10 φαίνεται η εκτέλεση του παραπάνω αλγορίθμου. Οι αριθμοί μέσα στους κύκλους συμβολίζουν την επανάληψη κατά την οποία ορίστηκαν αυτές οι θέσεις. Οι κυκλωμένες θέσεις υποδεικνύουν βέλτιστα. Αξίζει να σημειώσουμε ότι για τις θέσεις που δεν υπάρχει αρίθμηση σύμφωνη με την τρέχουσα επανάληψη του αλγόριθμου, ισχύει ότι έχουν ήδη υπολογιστεί σε προγούμενη επανάληψη.

Σχήμα 2.10: Τα βήματα εκτέλεσης του Αλγόριθμου Λογαριθμικής Αναζήτησης



3

Σχεδιασμός του Γλικού

3.1 ΕΙΣΑΓΩΓΗ

Στο προηγούμενο κεφάλαιο αναφερθήκαμε στη διαδικασία της εκτίμησης κίνησης, το ρόλο που αυτή διαδραματίζει στη συμπίεση βίντεο και τους συνήθεις αλγορίθμους εκτίμησης κίνησης που χρησιμοποιούνται σε διάφορες εφαρμογές. Στην εργασία αυτή αναπτύξαμε ένα ενσωματωμένο σύστημα το οποίο υλοποιεί τον αλγόριθμο εξαντλητικής αναζήτησης (FSBM) μέσω ενός ειδικά σχεδιασμένου επιταχυντή/συνεπεξεργαστή. Στο κεφάλαιο αυτό λοιπόν θα υπεισέλθουμε σε λεπτομέρειες που αφορούν τόσο την αρχιτεκτονική του συστήματος όσο και τις μεθόδους και τεχνικές που ακολουθήθηκαν κατά το σχεδιασμό του.

3.2 ΘΕΩΡΗΤΙΚΗ ΠΕΡΙΓΡΑΦΗ

Οι περισσότεροι αλγόριθμοι εκτίμησης κίνησης που έχουν προταθεί ως τώρα παρόλο που αποτελούν βελτιώσεις του αλγορίθμου εξαντλητικής αναζήτησης ως προς την ταχύτητα, ακολουθούν μια μη-κανονική ροή δεδομένων προς επεξεργασία. Αυτό έχει ως αποτέλεσμα την εξαιρετικά δύσκολη - αν όχι αδύνατη - υλοποίηση τους σε επίπεδο υλικού. Αντίθετα, ο FSBM παρόλη τη χαμηλή του ταχύτητα, επεξεργάζεται μια σταθερή και κανονική ροή δεδομένων καθ'όλη τη διάρκεια της εκτέλεσης του και αυτό αποτελεί πλεονέκτημα για την υλοποίησή του σε VLSI. Αυτός είναι ένας βασικός λόγος για τον οποίο ο FSBM εξακολουθεί να χρησιμοποιείται σε πολλές εφαρμογές.

Αναφέρθηκαμε πολλές φορές στο μεγάλο πλήθος υπολογισμών που απαιτούνται κατά την εκτέλεση του FSBM, πράγμα που εξαρτάται σε μεγάλο βαθμό από το μέγεθος του frame.

Πίνακας 3.1: Framerate και μέγεθος καρέ για μερικά διάσημα πρότυπα

Format	Pixels / line	Lines / Frame	FPS
UHD 8K	7680	4320	30
UHD 4K	3840	2160	30
HDTV	1920	1080	30
SDTV	720	486	30
Τηλεδιάσκεψη (SIF)	352	240	30

Στον πίνακα 3.1 αναφέρονται επιγραμματικά οι διαστάσεις των frame και το framerate για μερικά διαδεδομένα πρότυπα.

Όπως είναι εμφανές, για πρότυπα υψηλής και υπερυψηλής ευχρίνειας το μέγεθος του καρέ μεγαλώνει εκθετικά και μαζί ο αριθμός των εν δυνάμει περιοχών αναζήτησης, με τελικό αποτέλεσμα την αύξηση του υπολογιστικού φόρτου. Εκτός από αυτό όμως υπάρχει ένα ακόμη πρόβλημα, ίσως όχι άμεσα εμφανές. Ο ίδιος ο αριθμός των pixel που πρέπει να μετακινηθούν από την εξωτερική μνήμη είναι από κάθε άποψη τεράστιος, ειδικά σε πρότυπα όπως τα UHD 4/8K όπου για κάθε frame απαιτείται μεταφορά έως και 33.2 εκατομμυρίων pixel, δηλαδή περίπου 133 MByte δεδομένων!

3.2.1 ΕΠΑΝΑΧΡΗΣΙΜΟΠΟΙΗΣΗ ΔΕΔΟΜΕΝΩΝ

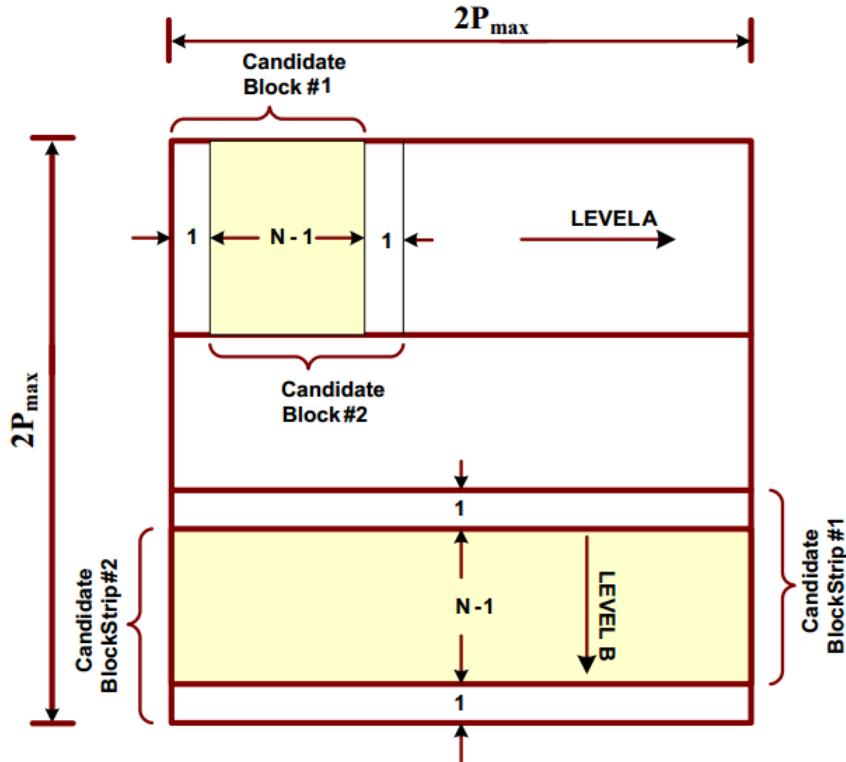
Το πρόβλημα του μεγάλου όγκου δεδομένων μπορεί να επιλυθεί χρησιμοποιώντας έξυπνες τεχνικές επαναχρησιμοποίησης δεδομένων. Με τον τρόπο αυτό μειώνουμε τις προσβάσεις στην εξωτερική μνήμη και μεταφέρουμε χρήσιμα δεδομένα μόνο όταν αυτό είναι απαραίτητο. Στο σχεδιασμό του επιταχυντή εφαρμόστηκε η επαναχρησιμοποίηση σε δύο επίπεδα, Α και Β.

Το επίπεδο Α αφορά στην επαναχρησιμοποίηση δεδομένων μεταξύ διαδοχικών θέσεων αναζήτησης σε μια κοινή ”λωρίδα” της περιοχής αναζήτησης. Αν θεωρήσουμε ένα candidate block διαστάσεων $N \times N$ pixel και μια περιοχή αναζήτησης $2P_{max} \times 2P_{max}$ τότε σε μία ”λωρίδα” της περιοχής αναζήτησης οι διαδοχικές θέσεις αναζήτησης θα αλληλοεπικαλύπτονται κατά $N-1 \times N$ pixel. Πρακτικά αυτό σημαίνει ότι κάθε επόμενο candidate block θα διαφέρει μόλις κατα μία στήλη από pixel - σε σχέση με το προηγούμενο - μεγέθους $1 \times N$ pixel. Αποτέλεσμα αυτής της παρατήρησης είναι ότι για να υπολογίσουμε κάθε νέα θέση απαιτείται η μεταφορά μίας και μόνο στήλης από την εξωτερική μνήμη, αφού το υπόλοιπο του candidate block μπορεί να επαναχρησιμοποιηθεί.

Το επίπεδο Β αφορά στην επαναχρησιμοποίηση δεδομένων μεταξύ διαδοχικών ”λωρίδων” της περιοχής αναζήτησης. Η λογική είναι ή ίδια με το επίπεδο Α αφού μεταξύ διαδοχικών λωρίδων αναζήτησης υπάρχει επικάλυψη κατά $N \times N - 1$ pixel, οπότε κατα την επεξεργασία μιας ”λωρίδας” το μεγαλύτερο μέρος της προηγούμενής της μπορεί να ξαναχρησιμοποιηθεί.

Στο σχήμα 3.1 μπορούμε να δούμε τα δύο επίπεδα επαναχρησιμοποίησης.

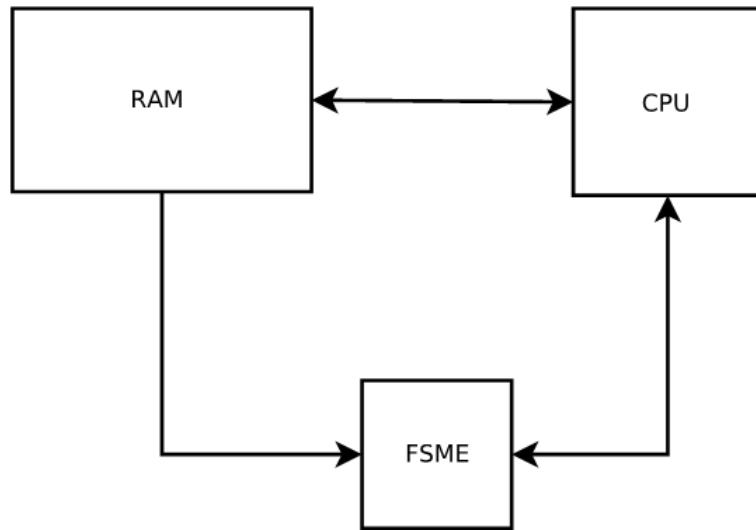
Σχήμα 3.1: Τα επίπεδα επαναχρησιμοποίησης A και B



3.3 ΣΧΕΔΙΑΣΜΟΣ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ

Το σύστημα του επιταχυντή αναπτύχθηκε σε δύο επίπεδα. Το πρώτο επίπεδο αφορά την εσωτερική λειτουργία του επιταχυντή, ενώ το δεύτερο τη διασύνδεση και την αλληλεπίδραση του με τα υπόλοιπα components. Στις επόμενες παραγράφους θα αναπτύξουμε και θα συζητήσουμε τις αρχιτεκτονικές που διέπουν τα δύο αυτά επίπεδα, καθώς και τις ιδιαιτερότητες και τα πλαινεκτήματα/μειονεκτήματά τους. Αρχικά, και προκειμένου να αποκτήσουμε μια γενική ιδέα, θα κάνουμε μια επιγραμματική αναφορά στον τρόπο με τον οποίο ο επιταχυντής αλληλεπιδρά με τα υπόλοιπα στοιχεία του συστήματος. Τα βασικά στοιχεία τα οποία πλαισιώνουν λοιπόν το περιφερειακό μας είναι ο επεξεργαστής και η μνήμη RAM. Ο επεξεργαστής παρέχει τα κατάλληλα σήματα στη RAM ώστε αυτή με τη σειρά της να τροφοδοτήσει με δεδομένα τον επιταχυντή, ενώ παράλληλα δέχεται τα αποτελέσματα της επεξεργασίας των δεδομένων και δίνει τον έλεγχο στον επιταχυντή όταν αυτό είναι απαραίτητο. Παρακάτω φαίνεται ένα εποπτικό σχήμα αυτής της δομής (σχήμα 3.2).

Σχήμα 3.2: Εποπτικό σχήμα της αρχιτεκτονικής του συστήματος

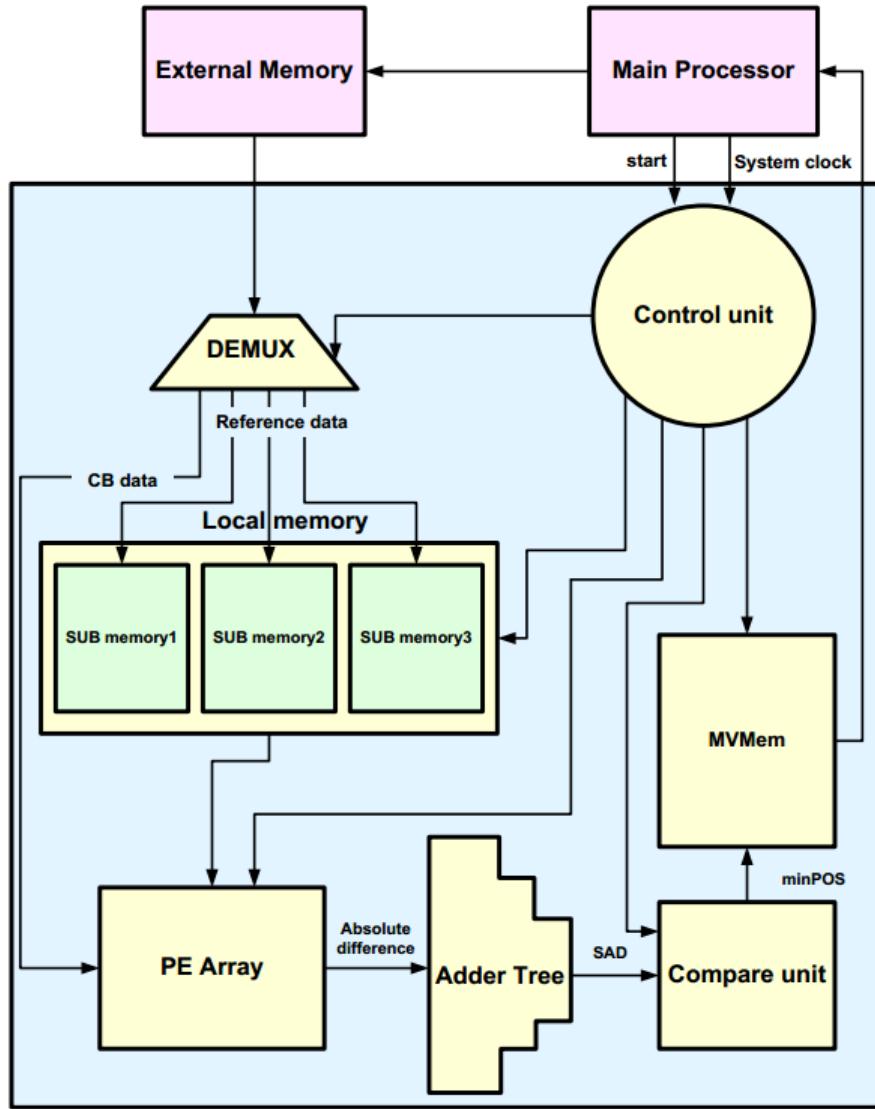


3.4 ΕΣΩΤΕΡΙΚΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΤΟΥ ΕΠΙΤΑΧΥΝΤΗ

Η αρχιτεκτονική που υλοποιήθηκε για το κύκλωμα του επιταχυντή φαίνεται στο σχήμα ?? και χρησιμοποιείται κυρίως στο πρότυπο H.264/AVC. Η περιοχή αναζήτησης που ανακτάται από την εξωτερική μνήμη έχει μέγεθος $2P_{max} \times (2P_{max} + N - 1)$ και το current block έχει διαστάσεις $N \times N$. Επιλέξαμε τόσο το N όσο και το P_{max} να έχουν την τιμή 16. Η διαδικασία της εκτίμησης κίνησης αρχίζει μόλις ο **αποπολυπλεκτης (demux)** λάβει τόσο τα pixel του current block (CB) όσο και τα pixel της περιοχής αναζήτησης από την εξωτερική μνήμη. Ο αποπολυπλέκτης δρομολογεί τα δεδομένα είτε προς την **τοπική μνήμη (local memory)** είτε απευθείας στη **μονάδα επεξεργασίας (PE Array)**. Η τοπική μνήμη αποτελείται από τρεις ”υπομνήμες”. Στη μονάδα επεξεργασίας καταλήγουν τόσο τα δεδομένα από την τοπική μνήμη (candidate blocks) όσο και το υπό εξέταση block (current block), όπου υπολογίζονται οι **απόλυτες διαφορές (absolute differences)**. Στη συνέχεια οι διαφορές αυτές θα προωθηθούν σε έναν αθροιστή δέντρου για να υπολογιστεί το άθροισμά τους και τελικά το κριτήριο **SAD (Sum of Absolute Differences)**. Το αποτέλεσμα της άθροισης τροφοδοτείται κατόπιν σε μια μονάδα σύγκρισης προκειμένου να εντοπιστεί η θέση εκείνη με το μικρότερο SAD. Μετά τη σύγκριση, η θέση του ελάχιστου τελικού SAD αποθηκεύεται με μια μνήμη διανυσμάτων κίνησης (Motion Vactor Memory), και τελικά το σύνολο των διανυσμάτων κίνησης αποστέλλεται στον επεξεργαστή στο τέλος της διαδικασίας. Η διαχείριση των σημάτων ελέγχου γίνεται από μια **μονάδα ελέγχου (control unit)** μέσα στον επιταχυντή.

Αξίζει να σημειώσουμε ότι πρόκειται για μία scalable αρχιτεκτονική η οποία μπορεί να προσαρμοστεί σε οσοδήποτε μεγάλα η μικρά frames και περιοχές αναζήτησης αρκει να

Σχήμα 3.3: Εποπτικό σχήμα της αρχιτεκτονικής του επιταχυντή



τροποποιηθεί κατάλληλα το εύρος των διαύλων, το μέγεθος των μνημών (αν απαιτείται από τις προδιαγραφές του format) και το μέγεθος της μονάδας επεξεργασίας (ώστε να έχει τη δυνατότητα παράλληλης επεξεργασίας περισσότερων δεδομένων). Είναι επομένως σχετικά εύκολο να δούμε την πρακτική της εφαρμογή σε πρότυπα όπως το H.265/HEVC, αφεί να επεκτείνουμε τη μονάδα επεξεργασίας σε 32×32 προκειμένου να εναρμονίζεται με τις απαιτήσεις του προτύπου. Στις επόμενες παραγράφους θα αναφερθούμε εκτενώς στα

επιμέρους τμήματα του επιταχυντή, την ακριβή λειτουργία τους, καθώς και τη διαδικασία σχεδιασμού τους.

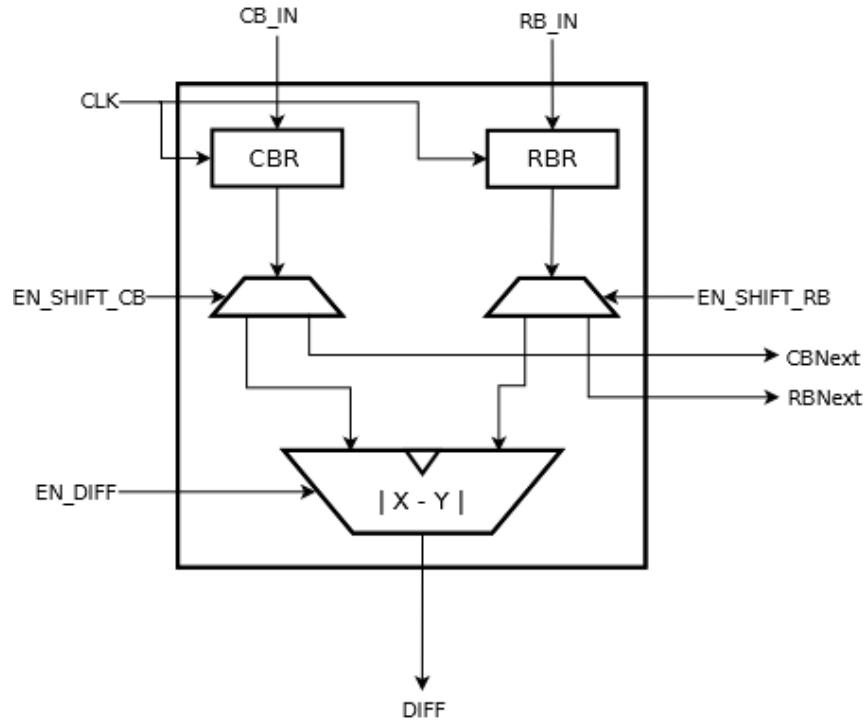
3.4.1 ΜΟΝΑΔΑ ΥΠΟΛΟΓΙΣΜΟΥ ΑΘΡΟΙΣΜΑΤΟΣ ΑΠΟΛΥΤΩΝ ΔΙΑΦΟΡΩΝ (SAD UNIT)

Η Μονάδα Υπολογισμού Αθροίσματος Απολύτων Διαφορών (στο εξής θα αναφερόμαστε σ' αυτήν ως SAD Unit για συντομία) αποτελεί την καρδιά του επιταχυντή. Πρόκειται επι της ουσίας για το μηχανισμό που εφαρμόζει το κριτήριο ταύτισης SAD και αναλαμβάνει το βάρος των απαραίτητων υπολογισμών. Το SAD Unit αποτελείται από δύο μικρότερα επιμέρους τμήματα, τη **Συστοιχία Επεξεργαστικών Στοιχείων (Processing Element Array - PE Array)** και το **Δέντρο Αθροιστών (Adder Tree)**. Κάθε ένα από αυτά τα τμήματα εξυπηρετεί διαφορετική διεργασία κατά την εφαρμογή του κριτηρίου SAD. Το PE Array είναι υπεύθυνο για τον υπολογισμό των διαφορών ενώ το Δέντρο Αθροιστών προσθέτει μεταξύ τους τις διαφορές που προκύπτουν στην έξοδο του PE Array.

Από τη στιγμή που αποφασίσαμε να χρησιμοποιήσουμε block διαστάσεων 16×16 pixel, εξηγείται εύκολα η δομή του PE Array, το οποίο αποτελείται από 16 γραμμές επεξεργασίας, κάθε μία από τις οποίες περιλαμβάνει 16 επεξεργαστικά στοιχεία (Processing Elements - PE), συνολικά δηλαδή, 256 PE. Κάθε στοιχείο επεξεργασίας λειτουργεί βρίσκοντας την απόλυτη διαφορά μεταξύ των τιμών δύο pixel, το ένα από τα οποία προέρχεται από το candidate block και το άλλο από το reference block. Ένα PE αποτελείται από δύο καταχωρητές (registers) των 8-bit, δύο πολυπλέκτες 2-σε-1 και μια μονάδα υπολογισμού απόλυτης διαφοράς. Οι δύο καταχωρητές αποθηκεύουν τις τιμές δύο pixel, ενός προερχόμενου από το candidate block και ενός από το reference block. Οι τιμές των pixel μπορεί να είναι από ο μέχρι 255, που αντιστοιχούν στις διαβαθμίσεις του γκρί (greyscale), επομένως 8 bit είναι αρκετά για την αναπαράσταση των τιμών. Τα δεδομένα του κάθε καταχωρητή σε κάθε κύκλο ρολογιού δρομολογούνται στον αντίστοιχο πολυπλέκτη. Αν βρισκόμαστε στη φάση της φόρτωσης των δεδομένων οι πολυπλέκτες θα στείλουν τα δεδομένα στο επόμενο PE, ενώ αν βρισκόμαστε στη φάση της επεξεργασίας τα δεδομένα θα δρομολογηθούν στη μονάδα αφαίρεσης όπου και θα υπολογιστεί η απόλυτη διαφορά τους. Από κάθε PE λαμβάνουμε ως τελική έξοδο αυτήν ακριβώς τη διαφορά ως τιμή των 8-bit. Στο σημείο αυτό θα πρέπει να σημειωθεί ότι τα PE δουλέυουν με μη-προσημασμένες τιμές (unsigned), επομένως το αποτέλεσμα θα είναι πάντα θετικό. Το σχήμα 3.4 περιγράφει την ακριβή δομή ενος processing element.

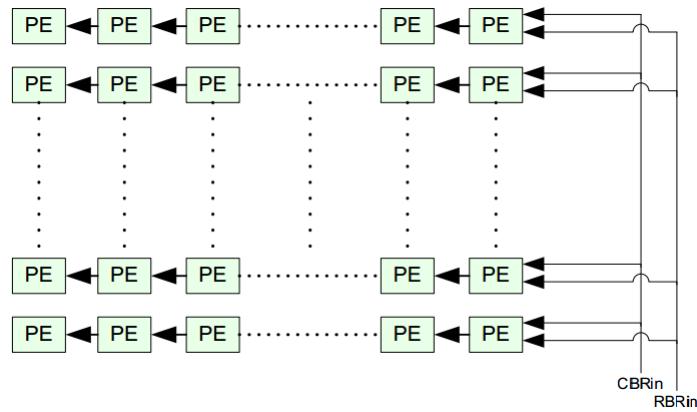
Η ροή των δεδομένων στο PE array γίνεται με παράλληλο τρόπο. Σε κάθε κύκλο ρολογιού φορτώνονται 16 pixel, από 1 σε κάθε γραμμή επεξεργασίας, πράγμα που σημαίνει ότι η φόρτωση γίνεται στήλη-προς-στήλη. Τα PE κάθε σειράς είναι συνδεδεμένα μεταξύ τους με τέτοιο τρόπο ώστε σε κάθε κύκλο ρολογιού τα δεδομένα του ενός να προωθούνται στο επόμενο σε μια λογική "οριζόντιας στοίβας". Μόνη εξαίρεση αποτελεί το τελευταίο PE κάθε σειράς αφού δεν χρειάζεται να προωθήσει τα δεδομένα σε κάποιο επόμενο. Ο υπολογισμός

Σχήμα 3.4: Ένα Στοιχείο Επεξεργασίας (Processing Element)



των διαφορών γίνεται **παράλληλα**, κάτι που σημαίνει ότι μόλις φορτωθουν τα δεδομένα των δύο block, σε έναν μόλις κύκλο ρολογιού έχουμε τη διαφορά τους.

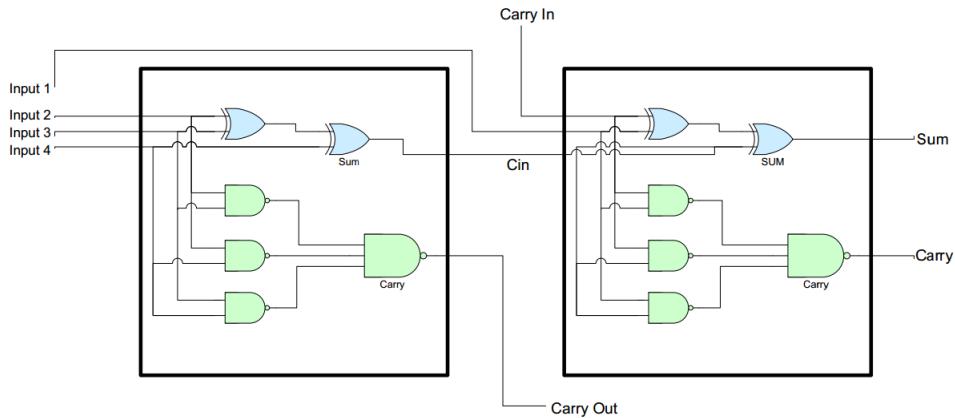
Σχήμα 3.5: Η δομή του PE Array



Κάθε PE όπως είπαμε και προηγουμένως παράγει τελικά ως αποτέλεσμα μια λέξη 8-

bit, επομένως μετά από τη σύγκριση κάθε θέσης λαμβάνουμε συνολικά 256 λέξεις των 8 bit (συνολικά 2048 bit), που κάθε μια αντιστοιχεί στη διαφορά ανάμεσα σε δύο pixel. Το επόμενο βήμα είναι η άθροιση αυτών των τιμών ωστε να βγεί το τελικό αποτέλεσμα της σύγκρισης με βάση το κριτήριο SAD. Η άθροιση μπορεί να γίνει με πολλούς τρόπους όπως για παράδειγμα με χρήση αθροιστών ριπής. Δυστυχώς το βασικό μειονέκτημα αυτής της πρακτικής είναι η πολύ μεγάλη καθυστέρηση που εισάγεται λόγω της μετάδοσης του κρατουμένου. Η καθυστέρηση αυτή θα καθιστούσε απαγορευτική τη χρήση του κυκλώματος σε εφαρμογές real-time, επομένως θα πρέπει να βρεθεί ένας τρόπος ωστε η πρόσθεση να μπορεί να γίνει γρήγορα και με ένα βαθμό παραλληλίας. Μία λύση στο πρόβλημα αυτό είναι η χρήση συμπιεστών 4:2 (4:2 compressors) σε διάταξη δέντρου.

Σχήμα 3.6: Η δομή του 4:2 compressor



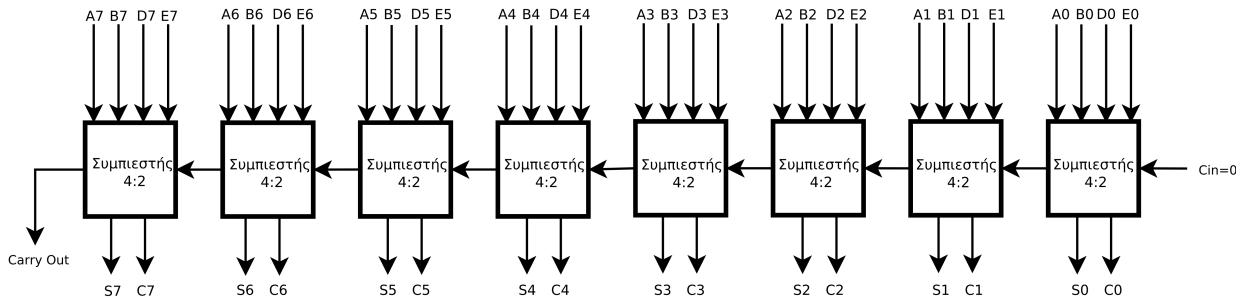
Ο συμπιεστής 4:2 αθροίζει συνολικά 4 bit τη φορά. Χρησιμοποιώντας λοιπόν 8 τέτοιους συμπιεστές είναι δυνατόν να έχουμε παράλληλη άθροιση 4 byte, με το αποτέλεσμα να είναι μια λέξη των 9 bit. Χρησιμοποιώντας αυτή τη δομή, μπορούμε να υλοποιήσουμε ένα "δέντρο" αθροιστών. Στο πρώτο στάδιο χωρίζουμε τα 256 byte που έδωσε το PE Array σε 16 ομάδες των 16 bytes η κάθε μία. Κάθε τέτοια ομάδα θα αθροιστεί από τέσσερις αθροιστές (συμπιεστές) των τεσσάρων byte ο καθένας σε συνδυασμό με έναν συμπιεστή των 10-bit. Αυτός ο δεύτερος αθροιστής, αθροίζει λέξεις των 10-bit.

Πίνακας 3.2: Πρόσθεση 2 byte με χρήση συμπιεστή

Carry Out	S_8	S_7	S_6	S_5	S_4	S_3	S_2	S_1
+								
C_8	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0

Στη συνέχεια, ακολουθείται η ίδια λογική, τροφοδοτώντας τα αποτελέσματα του πρώτου επιπέδου στο δεύτερο που αποτελείται από τέσσερις συμπιεστές των 12-bit και του δεύτερου

Σχήμα 3.7: Η δομή ενός 4-byte αθροιστή



στο τρίτο που έχει έναν μοναδικό συμπιεστή των 14-bit. Το τελικό αποτέλεσμα στην έξοδο του αθροιστή δέντρου είναι μια ποσότητα 16-bit που αποτελεί το SAD μεταξύ του candidate και του reference block. Αυτή η ποσότητα δρομολογείται στη συνέχεια στη μονάδα σύγκρισης, για την οποία θα μιλήσουμε παρακάτω.

3.4.2 ΜΟΝΑΔΑ ΤΟΠΙΚΗΣ ΜΝΗΜΗΣ

Η μονάδα τοπικής μνήμης (Local Memory Unit) είναι ο βασικός μηχανισμός με τον οποίο αφ' ενός τροφοδοτούμε με δεδομένα το PE Array και αφ' ετέρου εφαρμόζουμε την επαναχρησιμοποίηση δεδομένων με τον τρόπο που περιγράψαμε προηγουμένως. Στη μονάδα αυτή περιλαμβάνονται δύο επιμέροπυς στοιχεία, ο **αποπολυπλέκτης εισόδου (Demux)** και η **μονάδα μνήμης (Memory Unit)**, η οποία με τη σειρά της αποτελείται από τρία μικρότερα modules μνήμης, τις Submemories 1, 2 και 3.

Ο αποπολυπλέκτης εισόδου καθορίζει την αρχική ροή των δεδομένων από την εξωτερική μνήμη (RAM) στο εσωτερικό του επιταχυντή. Λειτουργεί ως διεπαφή (interface) μεταξύ της εξωτερικής και της τοπικής μνήμης και ανάλογα με την τιμή του selection signal τα δεδομένα μπορεί να δρομολογήθουν απευθείας στο SAD Unit ή σε κάποια από τις submemories. Το σήμα ελέγχου προέρχεται από τη μονάδα ελέγχου, στην οποία θα αναφερθούμε εκτενώς αργότερα.

Πίνακας 3.3

Σήμα Ελέγχου	Προορισμός Δεδομένων
00	SAD Unit
01	Submemory #1
10	Submemory #2
11	Submemory #3

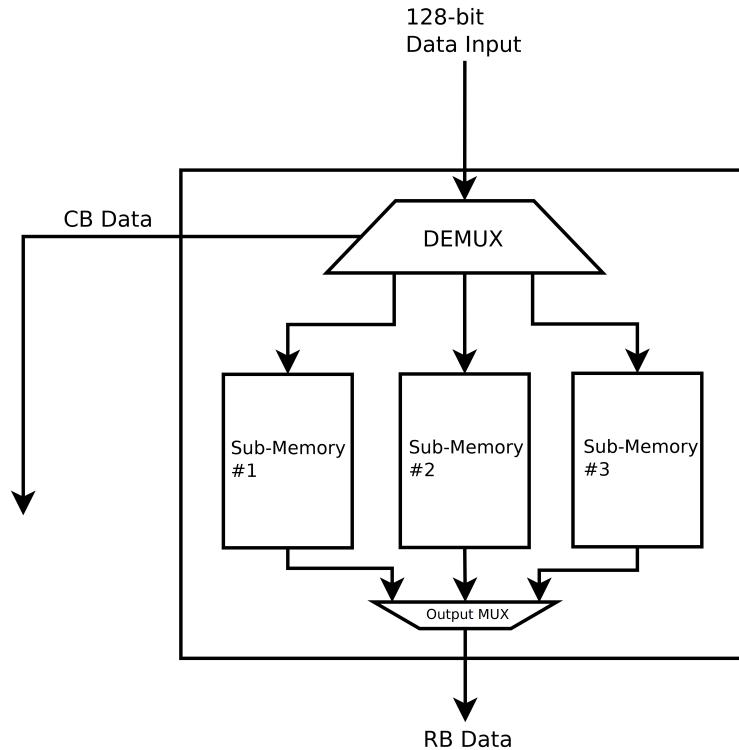
Η Μονάδα Τοπικής Μνήμης δέχεται ως είσοδο ένα δίαυλο δεδομένων των 128-bit. Με δεδομένο ότι ο επιταχυντής σχεδιάστηκε για να εφαρμόσει τεχνικές συμπίεσης σε εικόνες greyscale, το κάθε pixel της εικόνας αποτελείται από 8-bit. Ως εκ τούτου, η μονάδα μπορεί να δεχτεί κάθε φορά 16 pixel. Στο σημείο αυτό μπορούμε να αναφερθούμε στη δομή των sub-memory modules (υπο-μνήμες). Κάθε submemory λοιπόν αποτελείται από μια συστοιχία 256 καταχωρητών των 8-bit. Η συστοιχία αυτή έχει δομή 16×16 και κάθε καταχωρήτης μπορεί να αποθηκεύσει την τιμή ενός pixel. Κατά συνέπεια, μια submemory μπορεί να αποθηκεύσει ένα τμήμα εικόνας μεγέθους 16×16 pixel. Η εγγραφή των δεδομένων γίνεται γραμμή-προς-γραμμή, ενώ η ανάγνωση στήλη-προς-στήλη. Ο βασικός λόγος για τον οποίο δε χρησιμοποιήθηκε κάποια άλλη δομή μνήμης, είναι η εξαιρετική πολυπλοκότητα της διεύθυνσιοδότησης. Πέρα από το γεγονός ότι δε θα είχε κάποια πρακτική χρησιμότητα, η συστηματική διαυθυνσιοδότηση θα απαιτούσε interfaces τα οποία θα εισήγαν καθυστέρηση και περιορισμούς στη συχνότητα ρολογιού. Στη συγκεκριμένη περίπτωση τη διεύθυνσιοδότηση αναλαμβάνει ένας απλός μετρητής, ο οποίος ελέγχεται από τη μονάδα ελέγχου, και αυξάνεται σε κάθε κύκλο ρολογιού. Με τη χρήση του μετρητή μπορεί να γίνει απευθείας προσπέλαση σε μια στήλη της τοπικής μνήμης. Η αρχιτεκτονική αυτή εκτός των άλλων εξυπηρετεί φυσικά τη λογική της επαναχρησιμοποίησης δεδομένων. Το σχήμα 3.8 περιγράφει τη δομή της τοπικής μνήμης.

Σε προηγούμενη ενότητα αναφερθήκαμε στο πώς γίνεται η σύγκριση του candidate block με κάθε μία δυνητική θέση της περιοχής αναζήτησης (candidate blocks). Αυτός ο τρόπος σάρωσης της περιοχής αναζήτησης ονομάζεται **raster scan**. Ο τρόπος με τον οποίο πραγματοποιούμε αυτή τη σάρωση φαίνεται στο σχήμα 3.9, όπου απεικονίζεται η απαιτούμενη δομή δεδομένων για μια περιοχή αναζήτησης 32×32 pixel, με current block 16×16 pixel.

Η περιοχή αναζήτησης χωρίζεται σε τέσσερα μέρη, το καθένα από τα οποία έχει μέγεθος 16×16 pixel. Προκειμένου να υπολογίσουμε τα τελευταία pixel των θέσεων 2 και 4 θα χρειαστούμε ακόμα 16×15 pixel και αυτός είναι και ο ρόλος της γραμμοσκιασμένης περιοχής. Για το λόγο αυτό χρησιμοποιείται η τρίτη submemory (submemory #3) στην οποία φορτώνονται αυτά τα επιπλέον pixel. Για τις θέσεις 3 και 4 τα επιπλέον pixel (15×16) μπορούν να φορτωθούν με διαδοχικές προσβάσεις στις τρείς υπομνήμες. Πρακτικά ο ”αλγόριθμος” με τον οποίο φορτώνεται μια περιοχή αναζήτησης και γίνεται αναζήτηση ενός CB είναι η εξής:

- Αρχικά, κατά τους πρώτους 16 κύκλους ρολογιού, φορτώνεται το CB απευθείας στο PE Array, στήλη-προς-στήλη.
- Στους επόμενους 16 κύκλους φορτώνεται το πρώτο τμήμα της περιοχής αναζήτησης στην πρώτη υπο-μνήμη σειρά-προς-σειρα.
- Μόλις φορτωθεί το πρώτο τμήμα της περιοχής αναζήτησης στη μνήμη ξεκινάει αμέσως η φόρτωσή του στο PE Array, στήλη προς-στήλη. Αυτό όπως θα δούμε στη συνέχεια

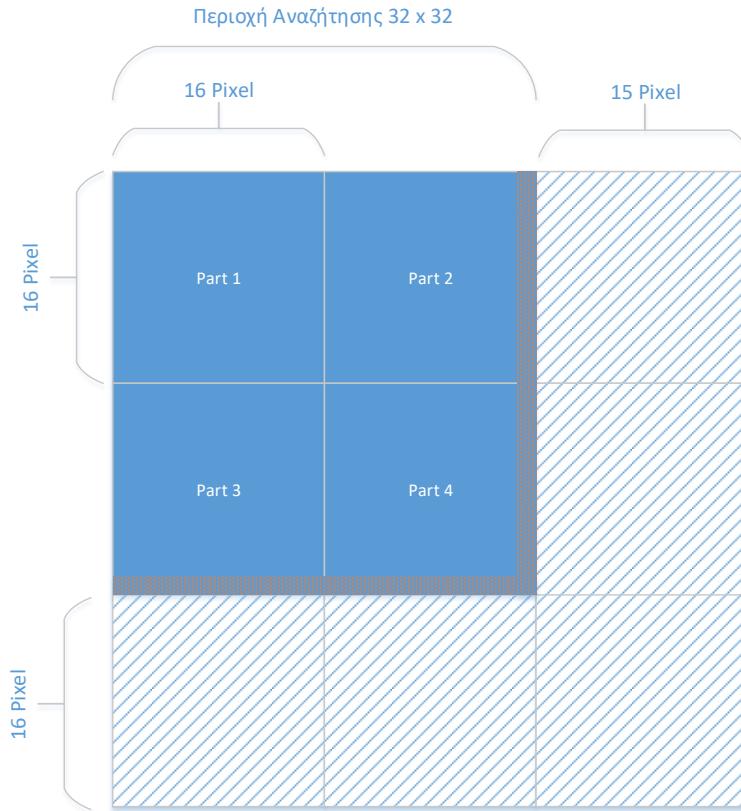
Σχήμα 3.8: Η αρχιτεκτονική της τοπικής μνήμης



εξυπηρετεί τους σκοπούς της επαναχρησιμοποίησης δεδομένων. Ταυτόχρονα, ξεκινάει η φόρτωση του δεύτερου τμήματος της περιοχής αναζήτησης στην δεύτερη υπο-μνήμη, η οποία και διαρκεί επίσης 16 κύκλους ρολογιού.

- Μόλις ολοκληρωθεί η φόρτωση του δεύτερου τμήματος της περιοχής αναζήτησης, ξεκινάει η προώθησή του στο PE Array, επίσης στήλη-προς-στήλη. Αξίζει να σημειώσουμε, ότι κάθε φορά που μια καινούργια στήλη της περιοχής αναζήτησης εισέρχεται στο PE Array, οι προηγούμενες που βρίσκονται ήδη σε αυτό ολισθαίνουν προς τα αριστερά (left-shifting). Εδώ, εμφανίζεται και το επίπεδο επαναχρησιμοποίησης A.
- Όσο προχωράει η ανάγνωση της submemory #2, η submemory #3 γεμίζει με δεδομένα. Με το τέλος των εγγραφών, μία μόνο σειρά από 16 pixel του τρίτου τμήματος της περιοχής αναζήτησης εγγράφεται στην υπο-μνήμη #1, εποιμάζοντας την ανάγνωση από την επόμενη ”λωρίδα” της περιοχής αναζήτησης. Εδώ εμφανίζεται το επίπεδο επαναχρησιμοποίησης B.
- Στους επόμενους 2 κύκλους ρολογιού θα εγραφεί μια σειρά από 16 pixel στις υπομνήμες #2 και #3, που θα αντιστοιχούν στο τέταρτο τμήμα της περιοχής αναζήτησης, και του

Σχήμα 3.9: Απεικόνιση της περιοχής αναζήτησης για ένα CB 16×16 pixel



γραμμοσκιασμένου χωρίου αντίστοιχα.

- Με το τέλος της ανάγνωσης της υπο-μνήμης #3, ενεργοποιείται εκ νέου η ανάγνωση από την υπο-μνήμη #1 και έτσι εισάγεται ξανά ένα νέο RB στο PE Array, από την επόμενη "λωρίδα" της περιοχής αναζήτησης.

Η διαδικασία αυτή επαναλαμβάνεται μέχρις ότου να εξαντληθούν όλες οι πιθανές θέσεις της περιοχής αναζήτησης.

3.4.3 ΜΟΝΑΔΑ ΣΥΓΚΡΙΣΗΣ

Η **Μονάδα Σύγκρισης (Compare Unit)** είναι ο μηχανισμός με τον οποίο αξιολογείται το υπολογισμένο SAD σχετικά με το αν αποτελεί βέλτιστο ή όχι κατα την αναζήτηση ενός CB. Η διαδικασία προκειμένου να βρούμε το μικρότερο δυνατό SAD μέσα σε μια

περιοχή αναζήτησης είναι σχετικά απλή: η μονάδα σύγκρισης περιέχει έναν καταχωρητή που αποθηκεύει το ελάχιστο SAD που έχει υπολογιστεί μέχρι εκείνη τη δεδομένη χρονική στιγμή καθώς και τη θέση του, σε ποιο σημείο δηλαδή εντοπίστηκε. Αρχικά ο καταχωρητής έχει την τιμη 65.535 (και τα 16 bit είναι μονάδες), όμως καθώς η διαδικασία της εκτίμησης κίνησης προχωράει η τιμή του καταχωρητή αντικαθίσταται κάθε φορά που εντοπίζεται μία μικρότερη. Πρακτικά δηλαδή, κάθε φορά που υπολογίζουμε ένα νέο SAD το συγκρίνουμε με το SAD που είναι αποθηκευμένο στον καταχωρητή της μονάδας σύγκρισης. Αν είναι μικρότερο, παίρνει τη θέση του παλιού, αλλιώς απορρίπτεται. Με τον τρόπο αυτό είμαστε σίγουροι ότι για κάθε CB έχουμε υπολογίσει την ελάχιστη διαφορά και κατ'επέκταση το βέλτιστο διάνυσμα κίνησης.

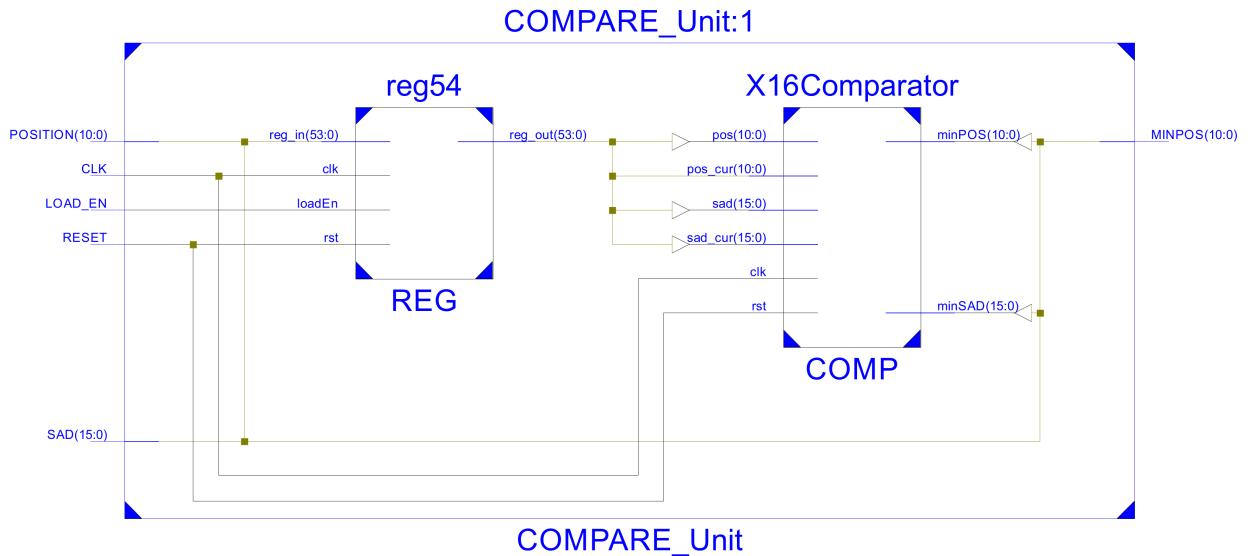
Εκτός από την τιμή του SAD ο συγκριτής διατηρεί αποθηκευμένη και την αντίστοιχη θέση στην οποία εντοπίστηκε. Η όλη διαδικασία της εκτίμησης κίνησης όπως θα δούμε και παρακάτω ελέγχεται αυστηρά από έναν μετρητή στην κεντρική μονάδα ελέγχου, ο οποίος αυξάνεται σε κάθε κύκλο ρολογιού. Εκτός των άλλων, η τιμή αυτού του μετρητή αντιπροσωπεύει παράλληλα τη θέση στην οποία βρισκόμαστε μέσα στην περιοχή αναζήτησης (και κατ'επέκταση το τρέχον RB), οπότε μπορούμε να χρησιμοποιήσουμε αυτήν την τιμή για τοποθετήσουμε χωρικά το SAD που μόλις υπολογίσαμε.

Σχεδιαστικά, τόσο το "παλιό" ζεύγος SAD-θέσης όσο και το νέο, αποθηκεύονται στον ίδιο καταχωρητή ως ενα συνενωμένο (concatenated) διάνυσμα 54-bit. Στην πορεία διαμοιράζονται τα δεδομένα, με τα SAD να τροφοδοτούνται ταυτόχρονα σ'εναν συγκριτή και έναν πολυπλέκτη, ενώ οι τιμές των θέσεων απευθείας σ'ενα δεύτερο πολυπλέκτη. Στους πολυπλέκτες διαμοιράζεται ένα κοινό σήμα ελέγχου το οποίο οδηγείται από την έξοδο του συγκριτή. Το τελικό ζεύγος SAD-θέσης συνενώνεται ξανά σ'ενα διάνυσμα 27-bit και τροφοδοτείται ξανά στον καταχωρητή της μονάδας σύγκρισης, ενώ ως τελική έξοδο λαμβάνουμε μόνο το 11-bit διάνυσμα της θέσης.

3.4.4 ΜΝΗΜΗ ΔΙΑΝΥΣΜΑΤΩΝ ΚΙΝΗΣΗΣ (Motion Vector Memory)

Η Μνήμη Διανυσμάτων Κίνησης (Motion Vector Memory) είναι η μνήμη στην οποία αποθηκεύονται τα διανύσματα κίνησης των blocks ολόκληρου του frame. Προκειται επί της ουσίας για μια ουρά FIFO αποτελούμενη από 1395 καταχωρητές των 11-bit ο καθένας (σχήμα ??). Η θέση που υπολογίζεται στο τέλος της διερεύνησης ενός CB, αποτελεί το πέρας ενος διανύσματος με αρχή την αρχική θέση του CB μέσα στο frame. Ένα frame του προτύπου SDTV (Standard Definition TeleVision) έχει διαστάσεις 720×486 pixel, επομένως με block size 16×16 pixel, κάθε frame αποτελείται από συνολικά 1395 block. Καθώς η σάρωση του frame γίνεται με τη μέθοδο raster, μπορούμε να αποθηκεύσουμε τη νέα θέση που υπολογίσαμε για κάθε block στον αντίστοιχο καταχωρητή της FIFO, αφού καθε νέα θέση που θα αποθηκεύεται θα προκαλεί ολίσθηση όλων των προηγούμενων προς τα επάνω. Τελικά το η θέση του πρώτου block θα είναι αποθηκευμένη στο τέλος της ουράς, ενώ του τελευταίου στην αρχή.

Σχήμα 3.10: Σχεδιάγραμμα του συγκριτή



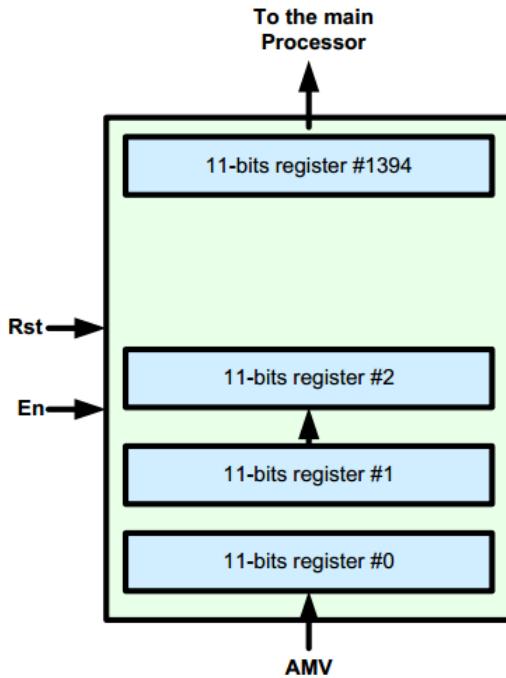
Στο τέλος της αναζήτησης κάθε frame, η μνήμη στέλνει τα δεδομένα της στον κεντρικό επεξεργαστή και κατόπιν κάνει reset τους καταχωρητές προκειμένου να είναι έτοιμοι για τα διανύσματα του επόμενου frame.

3.4.5 ΜΟΝΑΔΑ ΕΛΕΓΧΟΥ

Η Μονάδα Ελέγχου (Control Unit) είναι το σημαντικότερο και πιο περίπλοκο στοιχείο του επιταχυντή. Οδηγεί όλα τα απαραίτητα σήματα ελέγχου για τις επιμέρους μονάδες του συστήματος και αποτελεί το συνδετικό κρίκο μεταξύ του κεντρικού επεξεργαστή και του επιταχυντή. Η μονάδα ελέγχου αποτελείται από δύο τμήματα, έναν αυξοντα μετρητή και έναν controller που παράγει τα απαραίτητα σήματα. Η μονάδα ελέγχου δέχεται τρία σήματα ως είσοδο: enable, reset, και clk. Το σήμα enable ενεργοποιεί τη διαδικασία της εκτίμησης κίνησης, το reset μηδενίζει όλους τους καταχωρητές του συστήματος, ενώ το clk τροφοδοτείται από ένα σήμα ρολογιού προερχόμενο από τον επεξεργαστή. Τα σήματα εξόδου που παράγει η μονάδα ελέγχου είναι όλα όσα χρειάζονται για τον έλεγχο των επιμέρους μονάδων (Μονάδα Τοπικής Μνήμης, PE Array, κ.λπ.). Σε επόμενη ενότητα θα αναφερθούμε εκτενώς στην υλοποίηση της μονάδας ελέγχου και τα σήματα που αυτή παράγει.

Ο μετρητής της μονάδας ελέγχου αυξάνει σε κάθε κύκλο ρολογιού και χρησιμοποιείται

Σχήμα 3.11: Motion Vector Memory



για τη μέτρηση των κύκλων ρολογιού που απαιτούνται για την εκτίμηση της κίνησης κάθε block από το πρώτο (πάνω αριστερά) pixel μέχρι το τελευταίο (κάτω δεξιά). Έτσι, σε μια περιοχή αναζήτησης 32×32 pixel, οι τιμές που παίρνει ο μετρητής είναι από 000'h μέχρι 400'h. Προκειμένου να ξεχινήσει η μέτρηση θα πρέπει να ενεργοποιηθούν και τα τρία σήματα εισόδου της μονάδας ελέγχου, οπότε είναι προφανές ότι ο μετρητής μηδενίζεται με κάθε νέα αναζήτηση. Η τιμή του μετρητή αναπαριστά τη θέση του τρέχοντος CB μέσα στην περιοχή αναζήτησης και όπως είναι φυσικό, αυτή η θέση θα πρέπει στο τέλος να αντιστοιχηθεί στο συνολικό frame.

Ο controller της μονάδας ελέγχου δέχεται ως είσοδο την τιμή του μετρητή και παράγει τα ανάλογα σήματα που απαιτούνται στο συγκεκριμένο κύκλο ρολογιού. Στην προκειμένη περίπτωση ο μετρητής της μονάδας ελέγχου χρησιμεύει ως μηχανή πεπερασμένων καταστάσεων (Finite State Machine) με τα σήματα ελέγχου να αποκωδικοποιούν την κάθε κατάσταση.

3.5 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ

Ο επιταχυντής που περιγράφαμε παραπάνω δεν μπορεί να λειτουργήσει μόνος του. Στην πραγματικότητα αποτελεί μόνο ένα τμήμα ενός μεγαλύτερου και πολυπλοκότερου ενσωματωμένου συστήματος χτισμένου γύρω από την πλατφόρμα ZYNQ-7000 της Xilinx. Προκειμένου

να υπάρξει κάποια χρησιμότητα στον επιταχυντή που υλοποιήσαμε θα πρέπει αυτός να δέχεται δεδομένα από μία κεντρική μνήμη, και η λειτουργία του να ελέγχεται από έναν κεντρικό επεξεργαστή. Αυτά τα δεδομένα μας οδηγούν στην αναάπτυξη ενός πλήρους ενσωματωμένου συστήματος το οποίο περιλαμβάνει πολλά επιπλεόν υποσυστήματα όπως memory interfaces, clock generators και Direct Memory Access (DMA) Controllers. Το ενσωματωμένο σύστημα που υλοποιήσαμε περιλαμβάνει μια πληθώρα στοιχείων και βασίζεται σε τεχνολογίες και πρότυπα που θα είναι απαραίτητο να αναλυθούν προκειμένου να κατανοηθεί πλήρως ο τρόπος λειτουργίας του και οι δυνατότητες του.

3.5.1 ΒΑΣΙΚΗ ΔΟΜΗ

Η βασική δομή του συστήματος απαρτίζεται από τρία στοιχεία: την εξωτερική μνήμη (RAM), τον κεντρικό επεξεργαστή (ARM CPU) και τον επιταχυντή. Ο τελευταίος έχει το ρόλο περιφερειακού στο σύστημα, αφού δέχεται εντολές από τον επεξεργαστή και δεδομένα από την εξωτερική μνήμη. Τα δεδομένα που αποστέλλονται στον επιταχυντή είναι λέξεις των 128 bit που αντιστοιχούν στα pixel της εκάστοτε περιοχής αναζήτησης ή του εκάστοτε CB.

Είναι προφανές ότι σε καμία περίπτωση δεν πρέπει να καθυστερεί η αποστολή των δεδομένων στον επιταχυντή, γιατί σε αντίθετη περίπτωση θα έχουμε "data starvation" οπότε είτε θα πρέπει να εισάγουμε καθυστέρηση στον επιταχυντή είτε τελικά θα πάρουμε λανθασμένα αποτελέσματα. Η πρώτη σκέψη από πλευράς αρχιτεκτονικής προκειμένου να ελαχιστοποιήσουμε τον κίνδυνο του data starvation, είναι η απευθείας διασύνδεση της μνήμης RAM με το περιφερειακό. Αυτή η μέθοδος είναι γνωστή ως **Direct Memory Access (DMA)** και μας επιτρέπει να προσπελαύνουμε τη μνήμη χωρίς την προηγούμενη διαμεσολάβηση της CPU.

3.5.2 DMA

Οι δύο απλούστεροι τρόποι για τη μεταφορά δεδομένων μεταξύ της μνήμης και ενός περιφερειακού είναι το polling* και οι Διακοπές I/O (I/O Interrupts). Και οι δύο αυτές τεχνικές λειτουργούν καλύτερα με περιφερειακά που απαιτούν χαμηλό σχετικά εύρος ζώνης και δεν χρειάζονται αμεσότητα στην επικοινωνία τους με τη μνήμη. Επίσης, και οι δύο τεχνικές καθιστούν υπεύθυνο τον επεξεργαστή για τη διαχείριση της μεταφοράς των δεδομένων. Όπως είναι φυσικό χρειαζόμαστε έναν καλύτερο τρόπο για επικοινωνία με περιφερειακά που χρειάζονται μεγάλο bandwidth ή real-time δεδομένα.

Αυτός ο εναλλακτικός μηχανισμός ονομάζεται **Άμεση Προσπέλαση Μνήμης (Direct Memory Access - DMA)** και χρησιμοποιεί τις διακοπές (interrupts) μόνο για να ενημερώσει τον επεξεργαστή για το τέλος μιας μεταφοράς δεδομένων ή σε περίπτωση που υπάρξει σφάλμα.

*Polling: Ο ενεργός έλεγχος ανα τακτά χρονικά διαστήματα για το αν μια συσκευή έχει έτοιμα δεδομένα προς αποστολή.

Το DMA υλοποιείται από έναν εξειδικευμένο ελεγκτή (DMA Controller - DMAC) ο οποίος οργανώνει τη μεταφορά δεδομένων από και προς το εκάστοτε περιφερειακό και τη μνήμη, ανεξάρτητα από τον επεξεργαστή. Πρακτικά, στο μοντέλο master-slave που υλοποιείται στους περισσότερους διαυλους διασύνδεσης, ο DMAC γίνεται bus master και κατευθύνει τις αναγνώσεις/εγγραφές ανάμεσα στον ίδιο και τη μνήμη RAM. Υπάρχουν τρία βήματα σε μία μεταφορά DMA:

- Ο επεξεργαστής αρχικοποιεί το DMA παρέχοντας την ταυτότητα της συσκευής, τη λειτουργία που πρόκειται να πραγματοποιηθεί, τη διέθυνση μνήμης του προορισμού ή της πηγής των δεδομένων και των αριθμό των byte που πρόκειται να μεταφερθούν.
- Το DMA ξεκινάει τη λειτουργία στο περιφερειακό αναλαμβάνοντας το arbitration του διαύλου και μεταφέροντας τα δεδομένα όποτε αυτά είναι διαθέσιμα προς μετάδοση. Επιπλέον, το DMA παρέχει τις διευθύνσεις μνήμης από και προς τις οποίες θα γίνει η ανάγνωση/εγγραφή. Αν το request απαιτεί περισσότερες από μία διαμεταγωγές το DMA ορίζει τη νέα διεύθυνση μνήμης και ξεκινάει την επόμενη διαμεταγωγή. Χρημοποιώντας το μηχανισμό αυτό, μία συσκευή DMA μπορεί να ολοκληρώσει μία μεταφορά δεδομένων ακόμα και πολλών kBytes χωρίς να ενοχλήσει τον επεξεργαστή. Πολλοί DMA controllers περιέχουν εσωτερική μνήμη προκειμένου να διαχειριστούν τυχόν καθυστερήσεις είτε κατά τη μεταφορά, είτε κατά το διάστημα που ανέμεναν να πάρουν τον δίαυλο στον έλεγχο τους.
- Μόλις ολοκληρωθεί η μεταφορά ο ελεγκτής στέλνει ένα interrupt στον επεξεργαστή, ωστε ο τελευταίος να ελέγξει εαν και κατα πόσο ολοκληρώθηκε σωστά η διαδικασία.

Αξίζει να σημειώσουμε πως σ'ενα σύστημα μπορούν να υπάρχουν περισσότερες της μίας συσκευές DMA. Επίσης όταν υπάρχουν πολλαπλοί δίαυλοι I/O συνήθως κάθε ένας από αυτούς φέρει έναν ελεγκτή DMA που διαχειρίζεται τις μεταφορές δεδομένων από και προς τη μνήμη για το περιφερειακό που βρίσκεται συνδεδεμένο πάνω σ'αυτόν τον δίαυλο. Τέλος, ιεαραρχικά, μια συσκευή DMA έχει πάντα προτεραιότητα στην πρόσβαση στη μνήμη, ακόμα και σε σχέση με τον επεξεργαστή.

3.5.3 AXI

Στο σημείο αυτό αξίζει να αφιερώσουμε μερικές παραγράφους προκειμένου να συζητήσουμε για το πρωτόκολλο **AXI**. Στη συνέχεια αυτής της ενότητας θα αναφερθούμε επίσης σε λεπτομέρειες που διέπουν το πρωτόκολλο αυτό και θα εξηγήσουμε τη σημασία που έχει για την εσωτερική αρχιτεκτονική του συστήματός μας.

Το πρότυπο **AXI (Advanced eXtensible Interface)** αποτελεί μέλος της οικογένειας πρωτοκόλλων **AMBA (Advanced Microcontroller Bus Architecture)** που σχεδιάστηκε και υλοποιήθηκε από την ARM και περιλαμβάνει συστήματα διαύλων διασύνδεσης για μικροελεγκτές. Το

AMBA αποτελεί ένα ανοιχτό πρότυπο-αρχιτεκτονική για on-chip δίκτυα διασύνδεσης, το οποίο χρησιμοποιείται για να διασυνδέει και να διαχειρίζεται διαφορετικές λειτουργικές μονάδες σε ένα SoC. Η έκδοση 4 του προτύπου AXI που χρησιμοποιήθηκε στο σύστημα αποτελεί την τρίτη γενιά του προτύπου, που στοχεύει σε κυκλώματα υψηλών επιδόσεων και συχνοτήτων ρολογιού και εισήχθη το 2010 με την τέταρτη έκδοση του προτύπου AMBA (AMBA 4.0). Η οικογένεια διαύλων AMBA χρησιμοποιείται εκτενώς στα προϊόντα της ARM. Το σύστημα που αναπτύξαμε βασίζεται στην πλατφόρμα της Xilinx Zynq-7000 η οποία ως βασικό συστατικό της στοιχείο έχει έναν επεξεργαστή ARM και χρησιμοποιεί το πρότυπο AXI προκειμένου να πετύχει επικοινωνία μεταξύ του επεξεργαστή ARM (Processing System - PS) και της επαναπρογραμματιζόμενης λογικής (Programmable Logic - PL). Επίσης πρέπει να τονίσουμε πως η Xilinx χρησιμοποιεί αυτό το πρωτόκολλο ως την εξ'ορισμού επιλογή της για τα σύγχρονα SoC που παράγει. Ως εκ τούτου, είναι προφανές ότι οποιοδήποτε νέο περιφερειακό δημιουργούμε και σκοπεύουμε να το διασυνδέσουμε με την πλατφόρμα Zynq, θα πρέπει να επικοινωνεί πανω από το δίαυλο AXI.

Συνολικά το AXI διαθέτει τρεις τύπους interface:

- **AXI4:** Έκδοση για χρήση σε IP υψηλών επιδόσεων τα οποία χαρτογραφούνται στη μνήμη του συστήματος.
- **AXI4-Lite:** Αποτελεί υποσύνολο του AXI4. Χρησιμοποιείται για απλές, μονές διαμεταγωγές μεταξύ χαρτογραφημένων IP.
- **AXI4-Stream:** Χρησιμοποιείται σε μη χαρτογραφημένα στη μνήμη περιφερειακά, όπου απαιτείται υψηλής ταχύτητας συνεχής ροή δεδομένων.

Κάθε τύπος έχει διαφορετικές χρήσεις και πλεονεκτήματα στα οποία θα αναφερθούμε επιγραμματικά παρακάτω.

ΛΕΙΤΟΥΡΓΙΑ ΤΟΥ AXI

Οι προδιαγραφές του προτύπου ορίζουν το AXI ως διεπαφή μεταξύ ενός AXI Master και ενός AXI Slave (Μοντέλο "Αφέντη-Σκλάβου") τα οποία λογίζονται ως δύο IP τα οποία ανταλλάσσουν μεταξύ τους δεδομένα. Τα AXI4 και AXI4-Lite χρησιμοποιούν πέντε διαφορετικά κανάλια:

- Read Address Channel
- Write Address Channel
- Read Data Channel
- Write Data Channel
- Write Response Channel

Τα δεδομένα μπορούν να μεταδίδονται ταυτόχρονα και προς τις δύο κατευθύνσεις, ενώ και τα μεγέθη τους μπορούν να διαφέρουν. Αυτό υποστηρίζεται πλήρως από την ύπαρξη διαφορετικών καναλιών για διευθύνσεις και δεδομένα τόσο στις εγγραφές όσο και στις αναγνώσεις. Το όριο για το AXI4 μία ριπή μέχρι και 256 μεταδόσεων δεδομένων, ενώ το AXI4-Lite επιτρέπει μία μόνο μετάδοση ανά μεταφορά.

Σε επίπεδο υλικού, το AXI4 επιτρέπει διαφορετικά ρολόγια στον master και τον slave ενώ επιτρέπει επίσης τη χρήση καταχωρητών διασωλήνωσης (pipeline registers) για την εξομάλυνση των χρονικών αποκλίσεων. Το AXI4-Lite λειτουργεί κατα κανόνα όμοια με την πλήρη εκδοχή του προτύπου, με τη μοναδική διαφορά ότι δεν υποστηρίζει τη λειτουργία ριπής. Το AXI4-Stream ορίζει ένα μοναδικό κανάλι για τη μετάδοση ροής δεδομένων, το οποίο βασίζεται στο κανάλι εγγραφής δεδομένων του AXI4. Σε αντίθεση με το AXI4, το AXI4-Stream μπορεί να αποστείλλει ριπές απεριόριστων δεδομένων, με το μειονέκτημα όμως ότι οι μεταδόσεις του δεν είναι δυνατό να αναδιαταχθούν.

Είναι σημαντικό να τονίσουμε ότι το πρωτόκολλο AXI4 δεν αναγνωρίζει ούτε υπαγορεύει μια συγκεκριμένη μορφή δεδομένων (πχ συγκεκριμένη μορφή πακέτων). Αυτό πρακτικά σημαίνει ότι τα εκάστοτε IP είναι υπεύθυνα για τη σωστή κωδικοποίηση-αποκωδικοποίηση των δεδομένων.

IP Υποδομής (INFRASTRUCTURE IPs)

Πρόκειται για IP που χρησιμοποιούνται στη σύνθεση πολύπλοκων συστημάτων ώστε να διαμορφώσουν την υποδομή των τελευταίων. Χρησιμέουν στη μεταφορα-μετατροπή των δεδομένων εσωτερικά του συστήματος χρησιμοποιώντας διεπαφές AXI4 γενικού σκοπού. Αυτά τα IP μπορεί να περιλαμβάνουν:

- Register Slices (για pipelining)
- AXI FIFOs (buffering/μετατροπή ρολογιού)
- AXI Interconnect IP (διασύνδεση χαρτογραφημένων IP μεταξύ τους)
- AXI DMA (Direct Memory Access - μετατροπή δεδομένων από memory-mapped σε stream)

Τα παραπάνω δεν αποτελούν τερματικά δεδομένων, αλλά χρησιμοποιούνται στη διασύνδεση διαφορετικών IP σ'ενα σύστημα.

Υλοποίηση της Διασυνδεσης AXI στο Σύστημα ΜΑΣ

Όπως ήδη αναφέραμε, το πρότυπο AXI υποστηρίζει τόσο memory-mapped I/O όσο και streaming I/O (non-memory-mapped). Στην πρώτη περίπτωση η πρόσβαση στα περιφερειακά γίνεται χρησιμοποιώντας την ιδέα της προσπέλασης μιας συγκεκριμένης διεύθυνσης μνήμης.

Αυτή η μεθοδολογία έχει ορισμένα πλεονεκτήματα, όπως η ομοιογένεια του συστήματος, δηλαδή, η πρόσβαση σε κάθε περιφερειακό μπορεί να γίνει απευθείας με μια απλή προσπέλαση μιας διεύθυνσης μνήμης. Στην περίπτωσή μας ωστόσο, αυτό το μοντέλο μας είναι δύσχρηστο για δύο λόγους:

- Πρώτον, χρειαζόμαστε άμεση επικοινωνία του επιταχυντή με τη μνήμη, λόγω μεγάλων απαιτήσεων σε ταχύτητα
- Δεύτερον, δεν υπάρχει κάποια απαίτηση για διευθυνσιοδότηση, αφου το περιφερειακό μας (ο επιταχυντής) απλώς έχει ανάγκη από δεδομένα ανεξαρτήτου σειράς, μορφής ή προέλευσης. Εν ολίγοις δεν ενδιαφερόμαστε για τη φύση των δεδομένων, παρά μόνο για την κανονικότητα στη ροή τους.

Το AXI4-Stream είναι σχεδιασμένο για τέτοιου είδους εφαρμογές, οδηγούμενες από τα δεδομένα και τη ροή τους, αγνοώντας παντελώς την ιδέα της διευθυνσιοδότησης.

Φυσικά, δεν είναι από μόνο του αρκετό για το συγκεκριμένο ενσωματωμένο σύστημα που σχεδιάσαμε. Η απαίτηση πρόσβασης στον επιταχυντή από λογισμικό, καθιστά επιβεβλημένη τη χρήση ενός υβριδικού μοντέλου, το οποίο συνδυάζει τη λογική των χαρτογραφημένων περιφερειακών με τη λογική της ροής δεδομένων. Αυτό ακριβώς το μοντέλο μπορεί να υλοποιηθεί με τη χρήση της AXI DMA Engine, μιας μηχανής DMA για το πρωτόκολλο AXI η οποία παρέχεται ως IP από τη Xilinx. Δουλειά της DMA engine είναι να επικοινωνεί με τον επεξεργαστή όποτε αυτό είναι απαραίτητο, προκειμένου να διαιτητεύει τη μεταφορά των δεδομένων από και προς τη μνήμη RAM.

3.5.4 Ύλοποιηση της Αρχιτεκτονικής

Το σύστημά μας όπως αναφέραμε και προηγουμένως, χρησιμοποιεί ένα μηχανισμό DMA για τη μετακίνηση των δεδομένων, ο οποίος μάλιστα οφείλει να συμμορφώνεται με το πρωτόκολλο επικοινωνίας AXI-Stream. Το γεγονός αυτό προσθέτει ένα νέο επίπεδο πολυπλοκότητας στην υλοποίηση της αρχιτεκτονικής, καθώς, σε συνδυασμό με άλλες επιλογές που κάναμε απαιτεί ειδική προσέγγιση τόσο σε επίπεδο υλικού όσο και λογισμικού.

ΜΕΘΟΔΟΙ ΔΙΑΜΕΤΑΓΩΓΗΣ ΔΕΔΟΜΕΝΩΝ

Η μετακίνηση δεδομένων με DMA μπορεί να γίνει με διαφορετικούς τρόπους και καθορίζεται από μια πληθώρα παραμέτρων. Αυτές μπορεί να είναι το εύρος του διαύλου δεδομένων, ο αριθμός των καναλιών επικοινωνίας και ο τρόπος με τον οποίο πραγματοποιούνται οι μεταδόσεις. Σχετικά με τις δύο πρώτες παραμέτρους η επιλογή ήταν εύκολη καθώς σε μεγάλο βαθμό είχε ήδη καθοριστεί από την αρχιτεκτονική του επιταχυντή. Επομένως η μηχανή DMA επικοινωνεί με τον επιταχυντή πάνω από ένα κανάλι επικοινωνίας με δίαυλο εύρους 128 bit. Σαφώς και αυτό μπορεί να αλλάξει ανάλογα με την εφαρμογή. Για

παράδειγμα η ύπαρξη δύο καναλιών επικοινωνίας μπορεί να ωφελήσει εφαρμογές η οποίες διαχειρίζονται μεγάλο όγκο δεδομένων ή με συγκεκριμένη μορφή, όπως για παράδειγμα 2-D πίνακες. Μια τέτοια εφαρμογή θα μπορούσε να χειρίζεται δεδομένα εικόνας ή βίντεο, οπότε να χρησιμοποιεί τα επιπλέον κανάλια για να γράφει ή να διαβάζει ένα ολόκληρο frame σε κάθε μεταφορά.

Σε ότι αφορά την τρίτη παράμετρο τα πράγματα είναι λίγο πιο περίπλοκα. Ο μηχανισμός DMA μας δίνει τη δυνατότητα να μεταφέρουμε δεδομένα με δύο τρόπους: με απευθείας προσπέλαση των καταχωρητών του ελεγκτή DMA και με τη χρήση Scatter-Gather descriptors. Στην πρώτη περίπτωση για κάθε μεταφορά εφαρμόζεται η εξής διαδικασία:

- Γίνεται αίτηση στον επεξεργαστή (interrupt request) για μία μεταφορά δεδομένων
- Ο επεξεργαστής περνάει τον έλεγχο στον ελεγκτή DMA προκειμένου να εκτελέσει τη μεταφορά
- Με την ολοκλήρωση της μεταφοράς ο ελεγκτής DMA ενημερώνει τον επεξεργαστή για την ολοκλήρωση με νέο interrupt
- Η διαδικασία επαναλαμβάνεται για κάθε μεταφορά δεδομένων

Στο σημείο αυτό αξίζει να επισημάνουμε δύο πράγματα: με τον όρο "μεταφορά" εννοούμε μία μετακίνηση δεδομένων, είτε από την κεντρική μνήμη, είτε προς αυτήν. Το μέγεθος των δεδομένων είναι άσχετο και μπορεί να φτάνει τα αρκετά Mbyte. Το δεύτερο πράγμα έχει να κάνει με τον τρόπο με τον οποίο δίνεται η εντολή στον ελεγκτή DMA και αυτό συμβαίνει συνήθως με απευθείας προσπέλαση των καταχωρητών ελέγχου του τελευταίου από τον επεξεργαστή, γι' αυτό το λόγο και στη βιβλιογραφία συναντάται ως **Direct Register Access**. Αυτός ο τρόπος μεταφοράς δεδομένων είναι ιδιαίτερα χρήσιμος και εφαρμόζεται σε μικρές διαμεταγωγές όπου τα δεδομένα βρίσκονται "συγκεντρωμένα" με μία περιοχή μνήμης. Πολλές φορές ωστόσο, στην πράξη τα δεδομένα είναι καταχερματισμένα σε αρκετές περιοχές της μνήμης, χωρίς συνέχεια. Σε αυτήν την περίπτωση είμαστε υποχρεωμένοι να χρησιμοποιήσουμε πολλαπλές αιτήσεις μεταφοράς δεδομένων, κάτι το οποίο μπορεί τελικά να επηρεάσει αρνητικά την απόδοση του συστήματός μας.

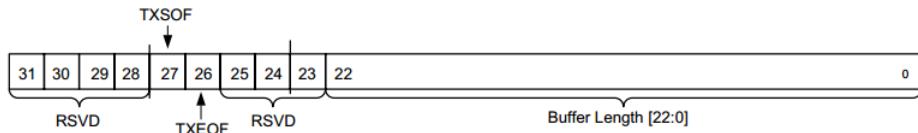
Προκειμένου να λυθεί αυτό το πρόβλημα καταφεύγουμε σε μια λύση που είναι γνωστή ως **Scatter-Gather (SG - Διασπορά-Συγκέντρωση)**. Σ' αυτή τη μέθοδο, χρησιμοποιείται η λογική των descriptors (περιγραφέων) προκειμένου να καθοδηγηθεί ο ελεγκτής DMA σχετικά με τις μεταφορές δεδομένων που πρέπει να εκτελέσει. Κάθε descriptor περιλαμβάνει οδηγίες σχετικά με τη διέύθυνση από την οποία πρέπει να αναγνωστούν (ή να εγγραφούν) τα δεδομένα, το μέγεθος της μεταφοράς που πρόκειται να γίνει, καθώς επίσης και μια διεύθυνση μνήμης η οποία δείχνει που βρίσκεται ο επόμενος descriptor. Με τον τρόπο αυτό δημιουργείται μια "αλυσίδα" από descriptors με τον καθένα να περιέχει τις οδηγίες προς το μηχανισμό DMA για μία μεταφορά δεδομένων. Ο πρώτος descriptor της αλυσίδας

ονομάζεται **head descriptor** ενώ ο τελευταίος **tail descriptor**. Οι descriptor αποθηκεύονται είτε στη μνήμη RAM είτε σε ειδική εξωτερική μνήμη που προορίζεται για αυτόν το σκοπό. Συνεπώς η διαδικασία η οποία εφαρμόζεται στη μέθοδο Scatter-Gather είναι η εξής:

- Γίνεται αίτηση στον επεξεργαστή (interrupt request) για μία μεταφορά δεδομένων
- Ο επεξεργαστής περνάει τον έλεγχο στον ελεγκτή DMA προκειμένου να εκτελέσει τη μεταφορά δίνοντας τη διεύθυνση του head descriptor
- Ο ελεγκτής DMA ενεργοποιεί τον μηχανισμό Scatter-Gather και ξεκινάει τη μεταφορά σύμφωνα με τις οδηγίες του head descriptor
- Στη συνέχεια ακολουθεί τις οδηγίες του επόμενου descriptor. Ο τελευταίος βρίσκεται στη θέση μνήμης που καθόρισε ο δείκτης του head descriptor
- Η διαδικασία εκτελείται για κάθε descriptor, μέχρι να φτάσουμε στον tail descriptor, οπότε και σημαίνει το τέλος της μετάδοσης

Όπως είναι προφανές, η χρήση του μηχανισμού Scatter-Gather δίνει έναν μεγαλύτερο βαθμό αυτονομίας στη μηχανή DMA, αφού αρχικά ο επεξεργαστής μπορεί να καθορίσει μέσω λογισμικού την αλληλουχία των μεταφορών δεδομένων που πρέπει να πραγματοποιηθούν με τους descriptors, τους οποίους χρησιμοποιεί κατόπιν η μηχανή DMA για να πραγματοποιήσει όσες μεταφορές χρειάζονται. Η γενική μορφή ενός descriptor φαίνεται στον πίνακα 3.4

Σχήμα 3.12: AXI MM2S Control Field



Οι descriptors αποτελούνται από 13 πεδία των 4-byte ή 32-bit. Μπορούν να διαχειριστούν δεδομένα που προέρχονται τόσο από διευθύνσεις μνήμης των 32-bit όσο και από διευθύνσεις μνήμης των 64-bit. Αυτός είναι και ο λόγος για τον οποίο σε ορισμένες εντολές του descriptor υπάρχουν δύο πεδία, με το ένα να καθορίζει τα 32 περισσότερο σημαντικά bit (MSB) και το άλλο τα 32 λιγότερο σημαντικά bit (LSB). Στην περίπτωση που δουλεύουμε με διευθύνσεις των 32 bit, το σκέλος που καθορίζει τα 32 MSB έχει τη δεκαεξαδική τιμή 0x00000000. Στο σημείο αυτό αξίζει να αναφερθούμε ειδικά στο control field του descriptor και τη δομή του. Όπως φαίνεται και στο σχήμα 3.12 υπάρχουν συνολικά 7 bit (#23-#25, #28-#31) τα οποία είναι κατειλημμένα (reserved) ενώ τα bit #26 και #27 σημαίνουν την αρχή και το τέλος ενός frame. Τα bit #0 - #22 χρησιμοποιούνται για να αναπαραστήσουν το μέγεθος

Πίνακας 3.4: Δομή ενός SG descriptor

Addr.	Space Offset	Όνομα	Περιγραφή
ooh		NXTDESC	Δείκτης του επόμενου descriptor - 32 LSB
o4h		NXTDESC_MSB	Δείκτης του επόμενου descriptor - 32 MSB
o8h		BUFFER_ADDRESS	Θέση μνήμης από όπου θα γίνει ανάγνωση ή εγγραφή δεδομένων - 32 LSB
oCh		BUFFER_ADDRESS_MSB	Θέση μνήμης από όπου θα γίνει ανάγνωση ή εγγραφή δεδομένων - 32 MSB
10h		RESERVED	-
14h		RESERVED	-
18h		CONTROL	Δεδομένα που θα εγγραφούν στον control register της μηχανής AXI DMA
1Ch		STATUS	Δεδομένα που θα εγγραφούν στον status register της μηχανής AXI DMA
20h		APP0	User Application Field 0
24h		APP1	User Application Field 1
28h		APP2	User Application Field 2
2Ch		APP3	User Application Field 3
30h		APP4	User Application Field 4

της μετάδοσης σε bytes. Αυτό σημαίνει ότι πρακτικά κάθε descriptor μπορεί να μεταφέρει μέχρι και 8 Mbytes δεδομένων.

Οι descriptors στο σύστημά μας αποθηκεύονται σε μία μικρή τοπική μνήμη, ξεχωριστή από την κύρια RAM, μεγέθους 8 kb. Αυτή η μνήμη είναι τύπου Block-RAM (BRAM) και αποτελεί τμήμα της επαναπρογραμματιζόμενης λογικής του Zynq. Αυτή η προσέγγιση έχει δύο βασικά πλεονεκτήματα: το πρώτο είναι η πλήρης απομόνωση των δεδομένων των descriptors από τα υπόλοιπα δεδομένα της μνήμης του συστήματος και το δεύτερο ότι στη φάση προτοτυποποίησης προσδίδει ευελιξία για πειραματισμούς με διάφορες συχνότητες ρολογιών (κατά κανόνα υψηλές) και πλήθος descriptor. Φυσικά για πιο "συμπαγή" συστήματα μπορεί να χρησιμοποιηθεί η κύρια μνήμη μειώνοντας την πολυπλοκότητα και το απαιτούμενο υλικό.

AXI STREAM

Σε προηγούμενες παραγράφους αναφερθήκαμε στο πρωτόκολλο AXI Stream, ως το βασικό πρωτόκολλο που διέπει την επικοινωνία του περιφερειακού μας με το υπόλοιπο σύστημα. Ωστόσο θα πρέπει να εξηγήσουμε τον τρόπο με τον οποίο λειτουργεί αυτή η συγκεκριμένη μορφή του πρωτύπου AXI, την αρχιτεκτονική της, τα σήματα που χρησιμοποιεί και πώς αυτά τα χαρακτηριστικά επηρεάζουν τον τρόπο με τον οποίο αλληλεπιδρά το σύστημα με τον επιταχυντή.

Το πρωτόκολλο AXI Stream χρησιμοποιείται για τη μετάδοση δεδομένων μεταξύ δύο περιφερειακών τα οποία ωστόσο δεν είναι απαραίτητο να είναι χαρτογραφημένα στο σύστημα. Αυτό σημαίνει ότι οι μεταδόσεις δεν διέπονται από κάποιας μορφής διευθυνσιοδότηση, αλλά από έναν άλλο μηχανισμό χειραφίας (handshaking) ο οποίος είναι χαρακτηριστικός του πρωτοκόλλου. Οι μεταδόσεις δεδομένων στο AXI Stream χωρίζονται σε επίπεδα:

- **Μετάδοση (Transfer):** Μία μονή μετάδοση δεδομένων μεταξύ δύο περιφερειακών. Καθορίζεται από ένα συμβόλιο χειραφίας (TVALID-TREADY handshaking)
- **Πακέτο (Packet):** Το πακέτο συμβολίζει τη μαζική μεταφορά ενός αριθμού bytes. Μπορεί να περιλαμβάνει μία ή περισσότερες μεταδόσεις. Είναι ένας εύκολος τρόπος για τα περιφερειακά υποδομή να διαχειρίζονται τη ροή των δεδομένων.
- **Πλαίσιο (Frame):** Η ανώτερη βαθμίδα ενθυλάκωσης στο πρωτόκολλο AXI Stream. Κάθε frame περιέχει ακέραιο αριθμό πακέτων και μεταφέρει μεγάλο όγκο δεδομένων.
- **Ροή (Stream):** Ως ροή αναφέρουμε τη μεταφορά δεδομένων από μία πηγή σε έναν τελικό προορισμό.

Πριν αναλύσουμε το μηχανισμό της χειραφίας (handshaking) και της μετάδοσης δεδομένων παραθέτουμε στον πίνακα 3.5 τη λίστα με τα σήματα που χρησιμοποιεί το πρωτόκολλο.

Η διαδικασία του handshaking γίνεται με την ενεργοποίηση δύο σημάτων, του TVALID και του TREADY. Τα σήματα αυτά πρέπει να ενεργοποιηθούν με την κατάλληλη σειρά και να μείνουν ενεργοποιημένα για συγκεκριμένο χρονικό διάστημα προκειμένου μια μετάδοση να είναι επιτυχής. Ένας master οδηγεί το σήμα TVALID και το ενεργοποιεί όταν έχει έτοιμα δεδομένα προς μετάδοση. Από την άλλη πλευρά, ένας slave οδηγεί το σήμα TREADY και το ενεργοποιεί μόνο όταν είναι έτοιμος να λάβει δεδομένα. Μόνο όταν ο master λάβει το TREADY και εφόσον έχει ήδη ενεργοποιημένο το TVALID μπορεί να ξεκινήσει η μετάδοση. Γενικά το handshaking διέπεται από τους εξής κανόνες:

- Μια συσκευή που παίζει το ρόλο του master **απαγορεύεται** να περιμένει το σήμα TREADY του slave για να ενεργοποιήσει το TVALID
- Μια συσκευή που παίζει το ρόλο του slave **επιτρέπεται** να περιμένει το σήμα TVALID πριν ενεργοποιήσει το TREADY

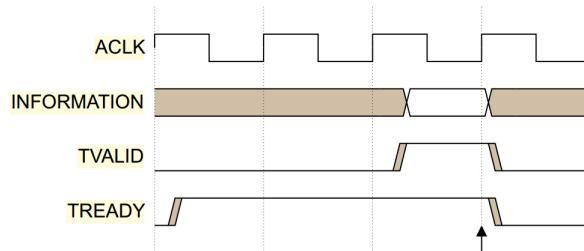
Πίνακας 3.5: Τα σήματα του AXI Stream

Σήμα	Πηγή	Περιγραφή
ACLK	Clock	Καθολικό σήμα ρολογιού. Όλα τα σήματα δειγματοληπτούνται κατά τη θετική ακμή του.
ARESETn	Reset	Καθολικό σήμα επαναφοράς. Είναι active-LOW.
TVALID	Master	Υποδεικνύει ότι υπάρχει μια έγκυρη μετάδοση σε εξέλιξη. Μια μετάδοση αρχίζει όταν ενεργοποιηθεί τόσο το TVALID όσο και το TREADY.
TREADY	Slave	Υποδεικνύει ότι ο slave είναι έτοιμος να δεχτεί δεδομένα στον ίδιο κύκλο ρολογιού.
TDATA[(8n-1):0]	Master	Πρόκειται για το σήμα που φέρει τα χρήσιμα δεδομένα της μετάδοσης (payload)
TSTRB[(n-1):0]	Master	Το σήμα χρησιμοποιείται για τη στοίχιση των δεδομένων και υποδεικνύει αν συγκεκριμένα byte θα αξιολογηθούν ως bytes δεδομένων ή bytes θέσης
TKEEP[(n-1):0]	Master	Χρησιμεύει στο να καθοριστεί ποια bytes αποτελούν χρήσιμο τμήμα της ροής δεδομένων και ποια όχι. Τα bytes που έχουν το TKEEP στο ο μπορούν να αφαιρεθούν από τη ροή των δεδομένων
TLAST	Master	Υποδεικνύει το τέλος ενός πακέτου
TID[(i-1):0]	Master	Χρησιμεύει για την ταυτοποίηση μια ροής δεδομένων, και μπορεί να υποδεικνύει διαφορετικές ροές
TDEST[(d-1):0]	Master	Περιλαμβάνει δεδομένα δρομολόγησης των δεδομένων
TUSER[(u-1):0]	Master	Επιπλέον παράλληλο σήμα το οποίο μπορεί να μεταδοθεί μαζί με την πληροφορία του TDATA και περιλαμβάνει πληροφορίες που ορίζονται από τον χρήστη

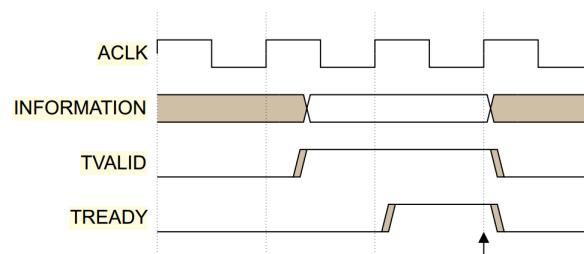
- Αν μια συσκευή που παίζει το ρόλο του slave ενεργοποιήσει το TREADY μπορεί να το απενεργοποιήσει μόνο αν δεν έχει ενεργοποιηθεί ακόμα το TVALID
- Γενικά, πρώτα ενεργοποιείται το TVALID και μετά το TREADY, διαφορετικά μπορούν να ενεργοποιηθούν ταυτόχρονα, στον ίδιο κύκλο ρολογιού

Στα σχήματα παρακάτω φαίνονται τα διαφορετικά σενάρια χειραψίας. Σε κάθε σχήμα το βέλος υποδεικνύει πότε ξεκινάει η μετάδοση των δεδομένων.

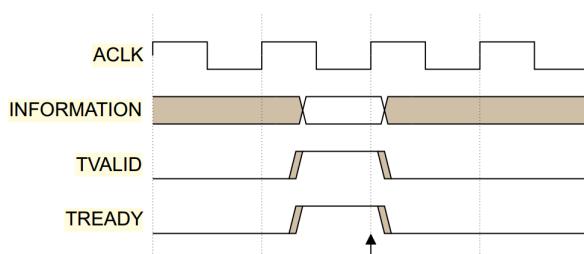
Σχήμα 3.13: Ενεργοποίηση του TREADY πριν το TVALID



Σχήμα 3.14: Ενεργοποίηση του TVALID πριν το TREADY



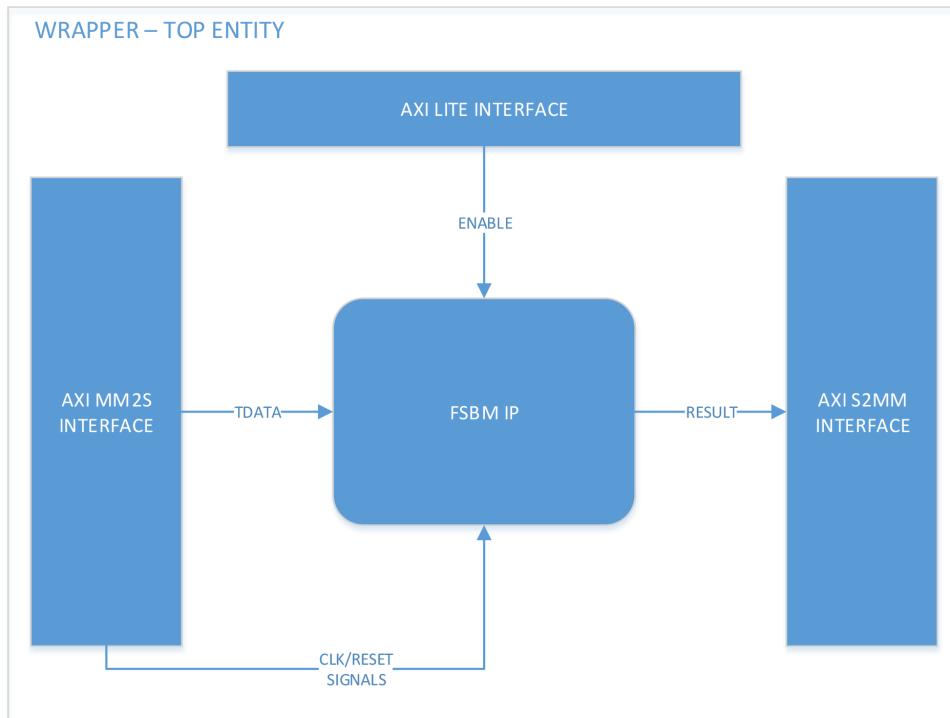
Σχήμα 3.15: Ταυτόχρονη ενεργοποίηση του TVALID και του TREADY



Προκειμένου να μπορέσει ο επιταχυντής να επικοινωνήσει πάνω από το πρωτόκολλο AXI Stream έπρεπε να τον μετατρέψουμε κατάλληλα ώστε να ενσωματώνει τα παραπάνω σήματα ελέγχου. Για να το πετύχουμε αυτό δημιουργήσαμε έναν wrapper, δηλαδή ένα πρόσθετο κύκλωμα το οποίο περιγράψαμε σε VHDL και το οποίο περιλαμβάνει τον επιταχυντή

αλλά και τρία AXI interfaces. Τα δύο από αυτά είναι AXI Stream και χρησιμοποιούνται για τη μετακίνηση δεδομένων από και προς τον επιταχυντή. Το ένα εξυπηρετεί το κανάλι **Memory-Mapped to Stream (MM2S)**, χρησιμεύει στη μεταφορά δεδομένων από την κύρια μνήμη στον επιταχυντή και παίζει το ρόλο του AXI slave. Το δεύτερο interface εξυπηρετεί το κανάλι **Stream to Memory-Mapped (S2MM)**, χρησιμεύει στη μετάδοση δεδομένων από τον επιταχυντή στην κύρια μνήμη και παίζει το ρόλο του AXI master. Το τρίτο interface είναι ένα AXI Lite interface και χρησιμεύει στην ενεργοποίηση-απενεργοποίηση του επιταχυντή μέσω ενός σήματος ελέγχου.

Σχήμα 3.16: Αρχιτεκτονική του Επιταχυντή με τα AXI Interfaces



4

Λογισμικό

Το λογισμικό αποτελεί βασικό τμήμα κάθε υπολογιστικού συστήματος. Στην περίπτωσή μας αποτελεί το συνδετικό κρίκο ανάμεσα στον χρήστη και το υλικό, αν και είναι προφανές από τη φύση του συστήματος, ότι η διαμεσολάβηση κάποιου χρήστη δεν είναι απαραίτητη, αφού ο επιταχυντής μπορεί να δουλέψει στο "παρασκήνιο" αρκεί φυσικά να έχει σταθερή ροή δεδομένων. Στην τελευταία περίπτωση το λογισμικό θα έπαιξε το ρόλο της διαιτησίας της ροής των δεδομένων. Το επίπεδο του λογισμικού χωρίζεται σε τρία επιμέρους τμήματα: το λειτουργικό σύστημα που εκτελείται στο κύριο επεξεργαστή, το πρόγραμμα-οδηγό (Driver) για το περιφερειακό που δημιουργήσαμε, και την εφαρμογή χρήστη (User Application). Θα αναφερθούμε ξεχωριστά στο καθένα από αυτά, αφού το καθένα είχε τις δικές του ιδιαιτερότητες και χρήζει ειδικής προσοχής.

4.1 Το Λειτουργικό Σύστημα

Στο ενσωματωμένο σύστημα που υλοποιήσαμε θα πρέπει να εκτελείται ένα λειτουργικό σύστημα το οποίο θα φροντίζει για την εκκίνηση του συστήματος, την κατάλληλη αναγνώριση των περιφερειακών, τη φόρτωση των κατάλληλων driver και τη παροχή μιας διεπαφής στο χρήστη ώστε να αλληλεπιδράσει με τον συνεπεξεργαστή. Όπως έχουμε ήδη αναφέρει, στο σύστημα περιλαμβάνεται ένας μικροεπεξεργαστής ARM. Αυτό πρακτικά σημαίνει ότι έχουμε μια πληθώρα λειτουργικών συστημάτων ωστε να διαλέξουμε ποιο είναι καταλληλότερο για τις ανάγκες μας. Αυτά περιλαμβάνουν Windows, Linux, FreeRTOS και πολλά άλλα. Η επιλογή που κάναμε ήταν το Linux, αφ'ενός γιατί είναι ένα δωρεάν λειτουργικό σύστημα με μεγάλη υποστήριξη από την κοινότητα των χρηστών και έτσι είναι πιο εύκολο να

αντιμετωπιστούν πιθανά προβλήματα. Ο δεύτερος λόγος είναι ότι είναι ιδιαίτερα φιλικό προς τους προγραμματιστές και τους μηχανικούς, αφού δίνει δυνατότητες όπως η παραμετροποίηση του πυρήνα, η εύκολη επικοινωνία με το hardware, και οι πολλές επεκτάσεις που διευκολύνουν την ανάπτυξη εφαρμογών.

Στην προκειμένη περίπτωση χρησιμοποιήθηκε η διανομή της Xilinx, PetaLinux η οποία χρησιμοποιείται κατα κόρον σε ενσωματωμένα συστήματα. Προκειμένου να εκτελέσουμε τη συγκεκριμένη διανομή στο σύστημά μας έπρεπε να ακολουθήσουμε μια μεγάλη διαδικασία, η οποία περιλάμβανε το compile του πυρήνα (kernel), την κατάλληλη χαρτογράφηση του hardware, και τη δημιουργία ενός boot-loader ο οποίος θα αναλάμβανε την αρχικοποίηση του συστήματος και κατόπιν θα περνούσε τον έλεγχο στον kernel.

4.1.1 LINUX KERNEL

5

Conclusion

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Morbi commodo, ipsum sed pharetra gravida, orci magna rhoncus neque, id pulvinar odio lorem non turpis. Nullam sit amet enim. Suspendisse id velit vitae ligula volutpat condimentum. Aliquam erat volutpat. Sed quis velit. Nulla facilisi. Nulla libero. Vivamus pharetra posuere sapien. Nam consectetuer. Sed aliquam, nunc eget euismod ullamcorper, lectus nunc ullamcorper orci, fermentum bibendum enim nibh eget ipsum. Donec porttitor ligula eu dolor. Maecenas vitae nulla consequat libero cursus venenatis. Nam magna enim, accumsan eu, blandit sed, blandit a, eros.

Quisque facilisis erat a dui. Nam malesuada ornare dolor. Cras gravida, diam sit amet rhoncus ornare, erat elit consectetuer erat, id egestas pede nibh eget odio. Proin tincidunt, velit vel porta elementum, magna diam molestie sapien, non aliquet massa pede eu diam. Aliquam iaculis. Fusce et ipsum et nulla tristique facilisis. Donec eget sem sit amet ligula viverra gravida. Etiam vehicula urna vel turpis. Suspendisse sagittis ante a urna. Morbi a est quis orci consequat rutrum. Nullam egestas feugiat felis. Integer adipiscing semper ligula. Nunc molestie, nisl sit amet cursus convallis, sapien lectus pretium metus, vitae pretium enim wisi id lectus. Donec vestibulum. Etiam vel nibh. Nulla facilisi. Mauris pharetra. Donec augue. Fusce ultrices, neque id dignissim ultrices, tellus mauris dictum elit, vel lacinia enim metus eu nunc.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo. Quisque sit amet est et sapien ullamcorper pharetra. Vestibulum erat wisi, condimentum sed, commodo vitae, ornare sit amet, wisi. Aenean fermentum, elit eget tincidunt condimentum,

eros ipsum rutrum orci, sagittis tempus lacus enim ac dui. Donec non enim in turpis pulvinar facilisis. Ut felis.

Cras sed ante. Phasellus in massa. Curabitur dolor eros, gravida et, hendrerit ac, cursus non, massa. Aliquam lorem. In hac habitasse platea dictumst. Cras eu mauris. Quisque lacus. Donec ipsum. Nullam vitae sem at nunc pharetra ultricies. Vivamus elit eros, ullamcorper a, adipiscing sit amet, porttitor ut, nibh. Maecenas adipiscing mollis massa. Nunc ut dui eget nulla venenatis aliquet. Sed luctus posuere justo. Cras vehicula varius turpis. Vivamus eros metus, tristique sit amet, molestie dignissim, malesuada et, urna.

Cras dictum. Maecenas ut turpis. In vitae erat ac orci dignissim eleifend. Nunc quis justo. Sed vel ipsum in purus tincidunt pharetra. Sed pulvinar, felis id consectetur malesuada, enim nisl mattis elit, a facilisis tortor nibh quis leo. Sed augue lacus, pretium vitae, molestie eget, rhoncus quis, elit. Donec in augue. Fusce orci wisi, ornare id, mollis vel, lacinia vel, massa.

 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi commodo, ipsum sed pharetra gravida, orci magna rhoncus neque, id pulvinar odio lorem non turpis. Nullam sit amet enim. Suspendisse id velit vitae ligula volutpat condimentum. Aliquam erat volutpat. Sed quis velit. Nulla facilisi. Nulla libero. Vivamus pharetra posuere sapien. Nam consectetur. Sed aliquam, nunc eget euismod ullamcorper, lectus nunc ullamcorper orci, fermentum bibendum enim nibh eget ipsum. Donec porttitor ligula eu dolor. Maecenas vitae nulla consequat libero cursus venenatis. Nam magna enim, accumsan eu, blandit sed, blandit a, eros.

 Quisque facilisis erat a dui. Nam malesuada ornare dolor. Cras gravida, diam sit amet rhoncus ornare, erat elit consectetur erat, id egestas pede nibh eget odio. Proin tincidunt, velit vel porta elementum, magna diam molestie sapien, non aliquet massa pede eu diam. Aliquam iaculis. Fusce et ipsum et nulla tristique facilisis. Donec eget sem sit amet ligula viverra gravida. Etiam vehicula urna vel turpis. Suspendisse sagittis ante a urna. Morbi a est quis orci consequat rutrum. Nullam egestas feugiat felis. Integer adipiscing semper ligula. Nunc molestie, nisl sit amet cursus convallis, sapien lectus pretium metus, vitae pretium enim wisi id lectus. Donec vestibulum. Etiam vel nibh. Nulla facilisi. Mauris pharetra. Donec augue. Fusce ultrices, neque id dignissim ultrices, tellus mauris dictum elit, vel lacinia enim metus eu nunc.

A

Some extra stuff

 Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Morbi commodo, ipsum sed pharetra gravida, orci magna rhoncus neque, id pulvinar odio lorem non turpis. Nullam sit amet enim. Suspendisse id velit vitae ligula volutpat condimentum. Aliquam erat volutpat. Sed quis velit. Nulla facilisi. Nulla libero. Vivamus pharetra posuere sapien. Nam consectetuer. Sed aliquam, nunc eget euismod ullamcorper, lectus nunc ullamcorper orci, fermentum bibendum enim nibh eget ipsum. Donec porttitor ligula eu dolor. Maecenas vitae nulla consequat libero cursus venenatis. Nam magna enim, accumsan eu, blandit sed, blandit a, eros.

 Quisque facilisis erat a dui. Nam malesuada ornare dolor. Cras gravida, diam sit amet rhoncus ornare, erat elit consectetuer erat, id egestas pede nibh eget odio. Proin tincidunt, velit vel porta elementum, magna diam molestie sapien, non aliquet massa pede eu diam. Aliquam iaculis. Fusce et ipsum et nulla tristique facilisis. Donec eget sem sit amet ligula viverra gravida. Etiam vehicula urna vel turpis. Suspendisse sagittis ante a urna. Morbi a est quis orci consequat rutrum. Nullam egestas feugiat felis. Integer adipiscing semper ligula. Nunc molestie, nisl sit amet cursus convallis, sapien lectus pretium metus, vitae pretium enim wisi id lectus. Donec vestibulum. Etiam vel nibh. Nulla facilisi. Mauris pharetra. Donec augue. Fusce ultrices, neque id dignissim ultrices, tellus mauris dictum elit, vel lacinia enim metus eu nunc.

 Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo. Quisque sit amet est et sapien ullamcorper pharetra. Vestibulum erat wisi, condimentum sed, commodo vitae, ornare sit amet, wisi. Aenean fermentum, elit eget tincidunt condimentum,

eros ipsum rutrum orci, sagittis tempus lacus enim ac dui. Donec non enim in turpis pulvinar facilisis. Ut felis.

Cras sed ante. Phasellus in massa. Curabitur dolor eros, gravida et, hendrerit ac, cursus non, massa. Aliquam lorem. In hac habitasse platea dictumst. Cras eu mauris. Quisque lacus. Donec ipsum. Nullam vitae sem at nunc pharetra ultricies. Vivamus elit eros, ullamcorper a, adipiscing sit amet, porttitor ut, nibh. Maecenas adipiscing mollis massa. Nunc ut dui eget nulla venenatis aliquet. Sed luctus posuere justo. Cras vehicula varius turpis. Vivamus eros metus, tristique sit amet, molestie dignissim, malesuada et, urna.

Cras dictum. Maecenas ut turpis. In vitae erat ac orci dignissim eleifend. Nunc quis justo. Sed vel ipsum in purus tincidunt pharetra. Sed pulvinar, felis id consectetur malesuada, enim nisl mattis elit, a facilisis tortor nibh quis leo. Sed augue lacus, pretium vitae, molestie eget, rhoncus quis, elit. Donec in augue. Fusce orci wisi, ornare id, mollis vel, lacinia vel, massa.

 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi commodo, ipsum sed pharetra gravida, orci magna rhoncus neque, id pulvinar odio lorem non turpis. Nullam sit amet enim. Suspendisse id velit vitae ligula volutpat condimentum. Aliquam erat volutpat. Sed quis velit. Nulla facilisi. Nulla libero. Vivamus pharetra posuere sapien. Nam consectetur. Sed aliquam, nunc eget euismod ullamcorper, lectus nunc ullamcorper orci, fermentum bibendum enim nibh eget ipsum. Donec porttitor ligula eu dolor. Maecenas vitae nulla consequat libero cursus venenatis. Nam magna enim, accumsan eu, blandit sed, blandit a, eros.

 Quisque facilisis erat a dui. Nam malesuada ornare dolor. Cras gravida, diam sit amet rhoncus ornare, erat elit consectetur erat, id egestas pede nibh eget odio. Proin tincidunt, velit vel porta elementum, magna diam molestie sapien, non aliquet massa pede eu diam. Aliquam iaculis. Fusce et ipsum et nulla tristique facilisis. Donec eget sem sit amet ligula viverra gravida. Etiam vehicula urna vel turpis. Suspendisse sagittis ante a urna. Morbi a est quis orci consequat rutrum. Nullam egestas feugiat felis. Integer adipiscing semper ligula. Nunc molestie, nisl sit amet cursus convallis, sapien lectus pretium metus, vitae pretium enim wisi id lectus. Donec vestibulum. Etiam vel nibh. Nulla facilisi. Mauris pharetra. Donec augue. Fusce ultrices, neque id dignissim ultrices, tellus mauris dictum elit, vel lacinia enim metus eu nunc.



THIS THESIS WAS TYPESET using L^AT_EX, originally developed by Leslie Lamport and based on Donald Knuth's T_EX. The body text is set in 11 point Egenolff-Berner Garamond, a revival of Claude Garamont's humanist typeface. The above illustration, Science Experiment 02, was created by Ben Schlitter and released under CC BY-NC-ND 3.0. A template that can be used to format a PhD dissertation with this look & feel has been released under the permissive AGPL license, and can be found online at github.com/suchow/Dissertate or from its lead author, Jordan Suchow, at suchow@post.harvard.edu.