Διπλωματική Εργασία

του Καραπατή Αθανάσιου

Υλοποίηση αποκωδικοποιητή video AVS με χρήση επαναδιατασσόμενης λογικής(FPGA): Υλοποίηση σε επίπεδο υλικού

Επιβλέποντες

Μπέλλας Νικόλαος, **αναπληρωτής καθηγητής**

Αντωνόπουλος Δ. Χρήστος, **επίκουρος καθηγητής**

Τμήμα Μηχανικών Η/Υ, Τηλεπικοινωνιών και Δικτύων

Πανεπιστήμιο Θεσσαλίας 2011

Στα αδέλφια μου
0100 0100 και 0100 0011

Ευχαριστίες

Φτάνοντας στο τέλος των προπτυχιακών μου σπουδών, παρά των οποιωνδήποτε δυσκολιών, θα ήθελα να ευχαριστήσω όλους όσους στάθηκαν δίπλα μου και με στήριξαν τόσο σε ακαδημαϊκό επίπεδο όσο και σε προσωπικό.

Πάνω όλους θέλω να ευχαριστήσω την οικογένεια μου που είτε μακριά είτε κοντά με στήριξαν σε όλη τη διάρκεια των σπουδών μου.

Στη συνέχεια θα ήθελα να ευχαριστήσω τους επιβλέποντες καθηγητές μου, κ. Νικόλαο Μπέλλα και Χρήστο Αντωνόπουλο, οι οποίοι με καθοδήγησαν και βοήθησαν στην περάτωση αυτής της διπλωματικής, αλλά και για τις πολύτιμες συμβουλές τους και τις γνώσεις που μου έδωσαν στην ακαδημαϊκή μου πορεία.

Επίσης θέλω να ευχαριστήσω όλους τους φίλους μου και την κοπέλα μου. Οι οποίοι με βοήθησαν έχοντας εμπιστοσύνη στις ικανότητες μου, στηρίζοντας με στις δύσκολες στιγμές και για όλο το κουράγιο που μου έδιναν. Χωρίς αυτούς δεν θα είχα καταφέρει ούτε τα μισά.

Καραπατής Αθανάσιος
Βόλος 2011

Περιεχόμενα

1. Εισαγωγή

Η ραγδαία αύξηση της χρήσης πολυμέσων σε διάφορους τομείς κάνουν επιτακτική την ανάγκη της βέλτιστης και αποτελεσματικής αποθήκευσης αυτών. Ένα είδος πολυμέσων είναι και το ψηφιακό βίντεο το οποίο αποτελείται από μία διαδοχική αλληλουχία στατικών εικόνων σε ψηφιακή μορφή. Με τον όρο «συμπίεση βίντεο» αναφερόμαστε στην διαδικασία μείωσης της ποσότητας των δεδομένων που απαιτούνται για την αποθήκευση. Τα κριτήρια της συμπίεσης είναι η ροή των δεδομένων να μην ξεπερνά κάποια όρια συναρτήσει της ποιότητας του βίντεο και της πολυπλοκότητας του υπολογισμού για την εφαρμογή. Η διαδικασία της συμπίεσης γίνεται στους εξής τομείς, χωρικό (εντός των δεδομένων μιας στατικής εικόνας) , χρονικό (μεταξύ των δεδομένων διαδοχικών εικόνων) και στο πεδίο της συχνότητας (αφαίρεση υψηλών συχνοτήτων οι οποίες δεν είναι ορατές από το ανθρώπινο μάτι).

Υπάρχουν δύο είδη συμπίεσης : lossy και lossless. Στη lossy συμπίεση έχουμε αφαίρεση δεδομένων κατά τη διαδικασία συμπίεσης τα οποία δεν είναι απαραίτητα για μία καλή αντίληψη του βίντεο από τον ανθρώπινο παράγοντα. Ο βαθμός της συμπίεσης σε αυτήν την περίπτωση είναι ανάλογο της ποιότητας της εικόνας και αντιστρόφως ανάλογος της ποσότητας των δεδομένων. Στη lossless συμπίεση δεν έχουμε καμία αφαίρεση δεδομένων απλά η αποθήκευση των δεδομένων γίνεται με βέλτιστο τρόπο και η ποιότητα της εικόνας δεν επηρεάζεται καθόλου. Η lossless συμπίεση χρησιμοποιείται σπάνια μιας και στη lossy συμπίεση επιτυγχάνεται πολύ καλύτερη αναλογία του βαθμού συμπίεσης προς την ποιότητα της εικόνας.

Έτσι λοιπόν με την ανάπτυξη διαφόρων υπηρεσιών όπως η ψηφιακή τηλεόραση, το internet video streaming (Youtube, dailymotion, κλπ), εμπορικά προϊόντα, όπως το DVD video , και νεότερες ανακαλύψεις όπως η τεχνολογία Blu-ray, βιντεοκλήση (πχ Skype, ooVoo) και πλέον η τρισδιάστατη τηλεόραση απαιτούν τη χρήση αποδοτικών τρόπων συμπίεσης του βίντεο. Μπορεί πλέον το κόστος αποθήκευσης ανά GB να είναι χαμηλό και το εύρος μετάδοσης να έχουν αυξηθεί παρόλα αυτά η μετάδοση και η αποθήκευση ασυμπίεστου βίντεο δεν αποτελεί αποδοτική λύση. Το μέχρι τώρα πιο διαδεδομένο standard συμπίεσης, το «γερασμένο» πια MPEG-2, καλείται σιγά σιγά να αποσυρθεί και να δώσει τη θέση του σε νεότερα, πιο αποδοτικά και τεχνολογικά καινοτόμα standards. Τα δύο επικρατέστερα, με όμοια καταγωγή, αλλά εντελώς διαφορετική στόχευση, είναι το MPEG-4 και το H.264. Το πρώτο εξ αυτών, επιχειρεί να δει τη συμπίεση βίντεο από μια εντελώς νέα οπτική σε σχέση με τα παλαιότερα standard, ενώ το H.264, ακολουθεί την πεπατημένη οδό, με μεγαλύτερη όμως επιτυχία από τους προκατόχους του στον τομέα των επιδόσεων. Κάπου ανάμεσα, το AVS video

standard, το οποίο θα μας απασχολήσει στην παρούσα εργασία, προσπαθεί να εδραιωθεί ως ο αντικαταστάτης του MPEG-2 στον αγώνα διαδοχής.

Σε αυτά, λοιπόν, τα πλαίσια, καθίσταται φανερή η ανάγκη για τη δυνατότητα γρήγορου encoding, και -πολύ περισσότερο- γρήγορου decoding συμπιεσμένου βίντεο. Η ολοένα αυξανόμενη ευκρίνεια του βίντεο μάλιστα (όπως High Definition Video και Ultra High Definition Video), κάνει την ανάγκη αυτή επιτακτική. Στα πλαίσια αυτά, επιχειρείται τόσο στον τομέα της ακαδημαϊκής έρευνας, όσο και στη βιομηχανία η βελτιστοποίηση των μεθόδων κωδικοποίησης/αποκωδικοποίησης βίντεο, όσο και η βελτιστοποίηση των ήδη υπαρχουσών, στο πλαίσιο της παροχής καλύτερου/πιο αποδοτικού hardware γενικής χρήσης, ή και embedded λύσεων ειδικού σκοπού.

Μια τέτοια προσπάθεια είναι και η παρούσα, η οποία προσπαθεί να εκμεταλλευτεί την υπολογιστική δύναμη και ευελιξία που μπορεί να παράσχει η λύση της επαναδιατασσόμενης λογικής.

2. Σκοπός εργασίας

Η παρούσα πτυχιακή έχει σα σκοπό την περιγραφή διαφόρων hardware accelerators του AVS video decoder σε υλικό(hardware) και η υλοποίηση τους σε συσκευή επαναδιατασσόμενης λογικής. Η περιγραφή του υλικού θα γίνει με τη χρήση της Verilog η οποία είναι μια γλώσσα περιγραφής υλικού (HDL).Πιο συγκεκριμένα τα hardware accelerator που θα έχουν σχεδιαστεί και παρουσιάζονται είναι το Deblocking Filter.

Συνοπτικά τα βήματα που ακολουθήθηκαν για την υλοποίηση των kernel είναι τα εξής :

1. Μελέτη των αλγορίθμων που πρόκειται να υλοποιηθούν και καθορισμός των δεδομένων που απαιτούνται για την εκτέλεση τους.

2. Σχεδιασμός της αρχιτεκτονικής του αλγορίθμου για την υλοποίηση του αλγορίθμου

3. Περιγραφή της αρχιτεκτονικής με τη γλώσσα περιγραφής υλικού Verilog και έλεγχος σε επίπεδο προσομοίωσης ως προς την ορθότητα της υλοποίησης

4. Χρησιμοποιώντας μια συσκευή επαναδιατασσόμενης λογικής για συσκευή προορισμού εξάγουμε χαρακτηριστικά αξιοποίησης λογικής και συχνότητα λειτουργίας.

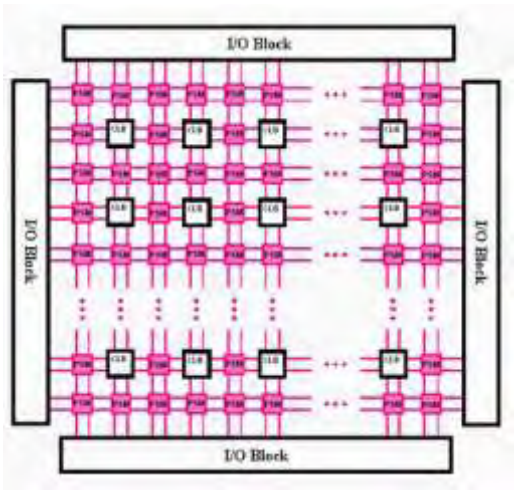
5. Στόχος της υλοποίησης ήταν να φτάσουμε τη συχνότητα λειτουργίας τουλάχιστον τα 160MHz .Στην περίπτωση που δεν ικανοποιούνταν η συγκεκριμένη συνθήκη γινόταν εντοπισμός του κρίσιμου μονοπατιού και χωρίζαμε τη λογική σε επιπλέον στάδια

Το σύστημα που χρησιμοποιηθεί για την εξαγωγή αποτελεσμάτων και μετρήσεων είναι μια FPGA **Xilinx®** της οικογένειας **Virtex® 5**, πιο συγκεκριμένα θα χρησιμοποιηθεί το μοντέλο **XC5VFX70T** και το εργαλείο της Xilinx ,**ISE** version 12.4. Χρησιμοποιώντας τη συγκεκριμένη FPGA σα συσκευή προορισμού θα εξάγουμε αποτελέσματα όπως ποσοστό λογικής που καταλαμβάνετε, μέγιστης συχνότητας λειτουργίας του κυκλώματος και άλλα.

Επιπλέον για την πλήρη κατανόηση των παραπάνω ζητημάτων και την επίτευξη των στόχων απαιτείται μελέτη διαφόρων ζητημάτων , όπως γενικά την φιλοσοφία και λειτουργία της FPGA , της θεωρίας Video decoding κ.α.

3. FPGA

3.1 Εισαγωγικά



Η (Field programmable gate array) είναι ένα ολοκληρωμένο κύκλωμα επαναδιατασσόμενης το οποίο έχει σχεδιαστεί να διαμορφώνεται επιτόπια από τον πελάτη ή το σχεδιαστή μετά την κατασκευή του. Αυτός είναι και ο λόγος που περιέχει το field programmable στην ονομασία της. Η διαμόρφωση της λογικής της FPGA στη γενική περίπτωση καθορίζεται χρησιμοποιώντας γλώσσα περιγραφής υλικού (hardware description language HDL). Με κυριότερα παραδείγματα τέτοιων γλωσσών να είναι η VHDL και Verilog .Χρησιμοποιώντας FPGA μπορεί να υλοποιηθεί οποιαδήποτε λογική συνάρτηση .Επομένως η δυνατότητα να αλλάζουν λειτουργικότητα, ακόμα και μερικώς εν’ ώρα χρήσης τους , και το χαμηλό κόστος αλλαγής συγκριτικά με ένα ASIC (ολοκληρωμένο κύκλωμα συγκεκριμένης λειτουργίας) , προσφέρει πολλά πλεονεκτήματα για χρήση σε πολλές εφαρμογές.

Οι FPGAs παρέχουν στοιχεία προγραμματιζόμενης λογικής τα οποία ονομάζονται "logic blocks"" ή fabric και μια ιεραρχία επαναρυθμιζόμενων διασυνδέσεων το οποίο επιτρέπει τα "logic blocks" να συνδέονται μεταξύ τους, έτσι ώστε οι διάφορες πύλες που μπορούν να αλλάξουν με πολλούς τρόπους να μπορούν να διασυνδέονται μεταξύ με ποικίλους τρόπους. Τα "logic blocks" μπορούν να ρυθμιστούν να υλοποιούν πολύπλοκες συνδυαστικές λογικές συναρτήσεις, ή απλές λογικές πύλες όπως AND και XOR. Στις περισσότερες FPGAs τα "logic blocks" περιέχουν και στοιχεία μνήμης, τα οποία μπορεί να είναι απλά flip-flop μέχρι και πλήρης block μνήμης.

Πέρα από "logic blocks" που παρέχονται στις FPGA υπάρχει μια τάση να παρέχονται στο chip και επιπλέον hard-ip. Δηλαδή συνδυάζονται τα logic blocks και τις διασυνδέσεις υλικό το οποίο κάνεις συγκεκριμένες πράξεις. Τέτοια hard-ip μπορεί να είναι κάποιος μικροεπεξεργαστής, γρήγορες μονάδες DSP και άλλα περιφερειακά συνιστώντας ένα πλήρες «σύστημα μέσα σε ένα προγραμματιζόμενο κύκλωμα». Παραδείγματα τέτοιας υβριδικής τεχνολογίας είναι και Xilinx Virtex 4 και 5 οι οποίες περιλαμβάνουν ένα PowerPC embedded processor , και αρκετές μονάδες DSP διάσπαρτες στο τσιπ. Πέρα της χρήσης του hard ip παρέχονται και βελτιστοποιημένες λύσεις soft-ip για κάθε οικογένεια , που αφορούν υλοποιήσεις

κάποιων κυκλωμάτων χρησιμοποιώντας logic blocks, κάτι το οποίο μπορεί να δώσει και καλύτερη απόδοση[1].Για παράδειγμα ένας soft processor cores απλούστερης αρχιτεκτονικής όπως ο Microblaze® από τη Xilinx. Επιπλέον είναι δυνατός και ο συνδυασμός των παραπάνω λύσεων, χρησιμοποιώντας δηλαδή και τους δύο μικροεπεξεργαστές.

3.2 Αρχιτεκτονική FPGA

Η πιο συνηθισμένη FPGA αρχιτεκτονική αποτελείται από ένα πίνακα από logic blocks (τα οποία ονομάζονται Configurable Logic Block, CLB, or Logic Array Block, LAB, εξαρτάται από την εκάστοτε εταιρία), I/O επαφές και κανάλια διασύνδεσης. Γενικά, όλα τα κανάλια διασύνδεσης έχουν το ίδιο μήκος (αριθμό από γραμμές). Πολλαπλές επαφές I/O είναι πιθανό να χωράνε στο ύψος μιας γραμμής ή στο πλάτος μιας στήλης του πίνακα.

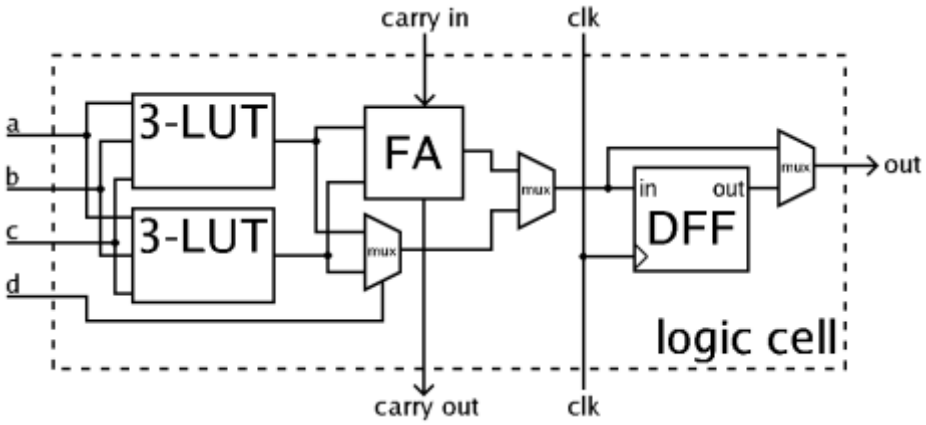
Το κύκλωμα για μία εφαρμογή πρέπει να τοποθετηθεί σε μια FPGA με επαρκείς πόρους. Ενώ ο αριθμός των CLB/LAB και I/O που απαιτούνται καθορίζονται εύκολα σε μία σχεδίαση, ο αριθμός των γραμμών διασύνδεσης τα οποία απαιτούνται μπορεί να διαφέρει σημαντικά ακόμα και σε αρχιτεκτονικές με τον ίδιο ποσοστό λογικής. Αφού μη χρησιμοποιημένες γραμμές διασύνδεσης αυξάνουν το κόστος (και μειώνουν την απόδοση) χωρίς να προσδίδουν κάποιο επιπλέον όφελος, οι κατασκευαστές FPGA επιδιώκουν να παρέχουν τόσες γραμμές έτσι ώστε οι περισσότερες αρχιτεκτονικές οι οποίες χωράνε σε νούμερα LUT και I/O να μπορούν να διασυνδεθούν . Κάτι το οποίο καθορίζεται από εκτιμήσεις οι οποίες βγαίνουν από διάφορους κανόνες ή από πειράματα σε υπάρχουσες αρχιτεκτονικές.

Configurable Logic Block

Στη γενική περίπτωση ένα logic block (CLB/ LAB) αποτελείται από μερικά λογικά κελιά (που ονομάζονται ALM, LE, Slice κ.ά.). Ένα τυπικό κελί αποτελείται από (βλέπε και εικόνα 3-1) :

- 4-εισόδων (πλέον έως και 6) LUTs, τα οποία είναι στοιχεία που μπορούν να υλοποιήσουν οποιαδήποτε λογική συνάρτηση τεσσάρων εισόδων
- Πλήρη αθροιστή
- D flip flop
- Πολυπλέκτες επιλογής μονοπατιών

[1] Περίπτωση όπου η τοποθέτηση των στοιχείων αποτελεί bottleneck επομένως η χρήση logic blocks να δίνει καλύτερα αποτελέσματα



Εικόνα 3-1 Απλοποιημένο παράδειγμα ενός λογικού κελιού

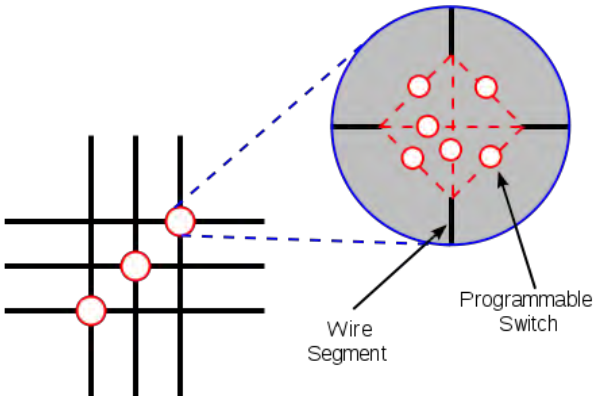
Programmable wiring

Όσον αφορά τη διασύνδεση και τη δρομολόγηση αυτών στην FPGA, αυτή είναι συνήθως unsegmented. Αναλυτικότερα, αυτό σημαίνει ότι κάθε γραμμή διασύνδεσης εκτείνεται όσο ένα logic block μέχρι να καταλήξει σε ένα switch box. Ανοίγοντας μερικά από τις προγραμματιζόμενες πύλες μέσα σε ένα switch box μπορούν να δημιουργηθούν μεγαλύτερα μονοπάτια. Μερικές αρχιτεκτονικές FPGA για να πετύχουν υψηλές ταχύτητας διασυνδέσεις χρησιμοποιούν γραμμές διασύνδεσης που εκτείνονται σε πολλαπλά logic blocks.

Όταν ένα κάθετο και ένα οριζόντιο κανάλι διασταυρώνονται, τότε υπάρχει ένα switch box. Σε αυτή την αρχιτεκτονική όταν μια γραμμή εισέρχεται σε ένα switch box υπάρχουν τρεις προγραμματιζόμενες πύλες το οποίο του δίνει τη δυνατότητα να συνδέεται με 3 άλλες γραμμές σε παρακείμενα κανάλια (βλέπε εικόνα 3-2) .

3.3 FPGA design και προγραμματισμός

Όσον αφορά τη διαδικασία υλοποίησης ενός FPGA design, ο σχεδιαστής περιγράφει τη σχεδίαση με μια γλώσσα περιγραφής υλικού, όπως αναφέρθηκαν και παραπάνω, ή χρησιμοποιώντας κάποιο schematic design . Σε μεγάλες σχεδιάσεις είναι



Εικόνα 3-2: Τοπολογία switch box

προτιμότερο η χρήση της γλώσσας περιγραφής υλικού επειδή ευκολότερο να τις αριθμήσεις από το να σχεδιάσεις με το χέρι καθεμία. Παρόλα αυτά, το schematic προσφέρει καλύτερη οπτική μιας σχεδίασης.

Σε επόμενη φάση, χρησιμοποιώντας κάποιο εργαλείο EDA (electronic design automation), παράγεται ένα σύνολο πυλών (netlist) για εκάστοτε τεχνολογία. Το netlist μπορεί να «αντιστοιχιστεί» σε μία αρχιτεκτονική FPGA εφαρμόζοντας τη διαδικασία που ονομάζεται place –and-route (τοποθέτηση και δρομολόγηση), συνήθως γίνεται με ιδιόκτητο λογισμικό της εταιρίας της FPGA. Ο σχεδιαστής επαληθεύει τα αποτελέσματα της αντιστοίχησης (mapping) και του place and route μέσω timing analysis, προσομοίωσης, και άλλων μεθόδων επαλήθευσης . Όταν η σχεδίαση και διαδικασία ελέγχου ολοκληρωθούν, παράγεται το αρχείο που θα χρησιμοποιηθεί για να προγραμματίσει την FPGA. Το αρχείο ονομάζεται bitstream και μεταφέρεται στην FPGA μέσω μιας σειριακής διεπαφής (JTAG – serial interface) ή κάποιας εξωτερικής μνήμης, π.χ. μια Flash Memory.

Για να απλοποιηθεί η σχεδίαση πολύπλοκων συστημάτων στις FPGAs, υπάρχουν διάφορες βιβλιοθήκες προκαθορισμένων πολύπλοκων συναρτήσεων και κυκλωμάτων τα οποία έχουν ελεγχθεί και βελτιστοποιηθεί για να επιταχύνουν τη διαδικασία της σχεδίασης. Αυτές τα προκαθορισμένα κυκλώματα ονομάζονται συνήθως IP cores και είναι διαθέσιμα από τους κατασκευαστές των FPGA είτε από άλλους προμηθευτές (σπανίως δωρεάν , και συνήθως παρέχονται υπό συγκεκριμένες άδειες χρήσης). Πέρα όμως από τις προηγούμενες λύσεις, υπάρχουν και κοινότητες σχεδιαστών που παρέχουν έτοιμα κυκλώματα, ένα από αυτά είναι το OpenCores (στο οποίο εκδίδονται λύσεις με ελεύθερο ή ανοιχτού κώδικα άδειες χρήσεις όπως οι GPL και BSD).

3.4 Virtex 5 XC5VFX70T

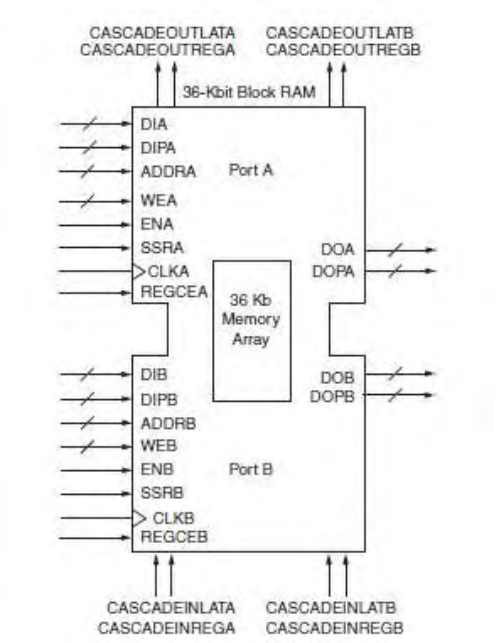
Η FPGA που χρησιμοποιήθηκε σαν συσκευή προορισμού για τις σχεδιάσεις που θα υλοποιηθούν έτσι ώστε να εξάγουμε συμπεράσματα όπως συχνότητα λειτουργίας, ποσοστό κάλυψης της FPGA κ.ά. Η οικογένεια Virtex 5 κατασκευάζεται στην τεχνολογία των 65nm.Επιπλέον κάποια από τα χαρακτηριστικά της οικογένειας Virtex 5 είναι :

- 6 εισόδων lookup tables (LUT)
- Αρχιτεκτονική διαγώνιας διασύνδεση μεταξύ των CLB
- 64-bit distributed RAM
- Γρήγορες και μεγάλες DSP λειτουργίες (DSP48E)
- BRAMS

Πιο συγκεκριμένα για τη σειρά FXT (XC5VFX70T) είναι βελτιστοποιημένες για εφαρμογές ενσωματωμένων συστημάτων, DSP processing και υψηλών απαιτήσεων σε μνήμη εφαρμογές. Επιπλέον είναι η σειρά που παρέχει έναν ή δυο(και στην συγκεκριμένη ένα) PowerPC 440 embedded block, ο οποίος είναι 32bit και αρχιτεκτονικής RISC.

BRAMS

Οι BRAMS (block RAMs) είναι προγραμματιζόμενα στοιχεία μνήμης, τα οποία μπορούν να χρησιμοποιηθούν με διάφορους τρόπους. Είναι μοιρασμένα στο fabric[2] και μπορούν να χρησιμοποιηθούν ανάλογα. Είναι hard IP, το οποίο σημαίνει ότι δεν χρησιμοποιεί επαναδιατασσόμενη λογική για να υλοποιηθεί αλλά είναι αυτούσιο κύκλωμα. Μερικά από τα χαρακτηριστικά των BRAM που προσφέρονται στη Virtex 5 είναι τα εξής:



Εικόνα 3-3: Απεικόνιση κυκλώματος BRAM

1. Είναι Dual-port, το οποίο σημαίνει ότι έχει 2-εισόδους εγγραφής όπως και 2 εξόδους
2. Το μέγιστο μέγεθος είναι 36K bits
3. Το μέγιστο μέγεθος εγγραφής είναι 72bit, στην περίπτωση που οι 2 είσοδοι ρυθμιστούν κατάλληλα.
4. Μπορούν να ρυθμιστούν σε κατάσταση ανάγνωσης byte
5. Η ανάγνωση απαιτεί 1 κύκλο ρολογιού
6. Η εγγραφή απαιτεί ένα κύκλο ρολογιού
7. Η FPGA που χρησιμοποιήθηκε περιέχει 5.328K bit σε BRAM

[2] Fabric: Με τον όρο αυτό εννοούμε το υλικό επαναδιατασσόμενης λογικής

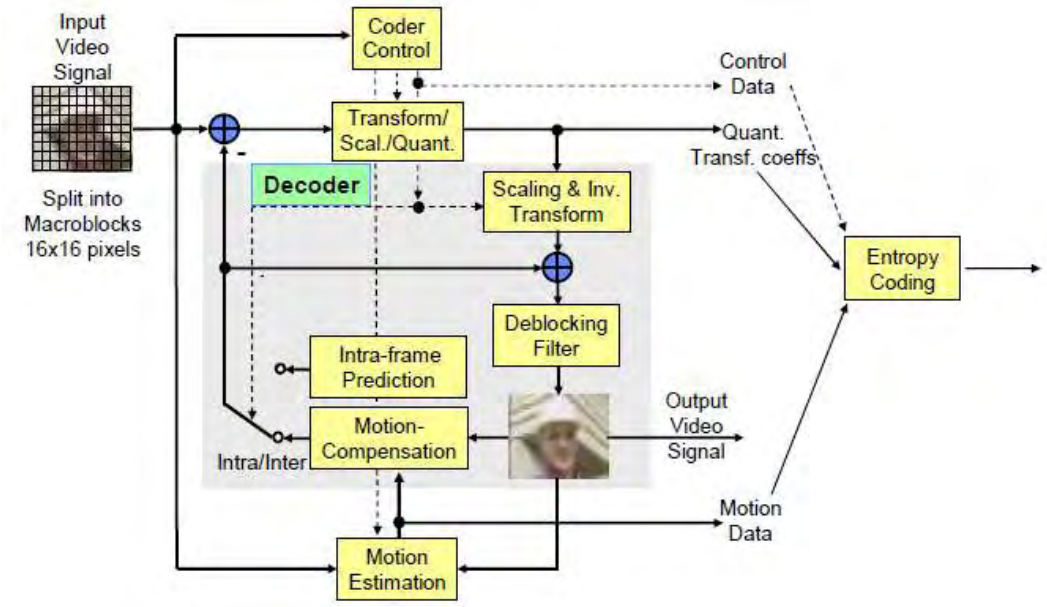
4. ΘΕΩΡΙΑ VIDEO CODEC AVS

4.1 Εισαγωγικά

Το AVS (audio video standard) είναι ένα σχετικά νέο standard συμπίεσης ήχου και βίντεο και μιμείται σε ένα βαθμό τα MPEG-4 και H.264 τα οποία σχεδιάστηκαν να αντικαταστήσουν το MPEG-2. Η ανάπτυξη του εκίνησε με πρωτοβουλία της κυβέρνησης της Κίνας, σε μια προσπάθεια να ελαττώσει τα royalty fees τα οποία καλούνταν να πληρώσουν οι κινέζοι σε ξένες εταιρίες, όπως και επίσης για να κερδίσει η κινέζικη βιομηχανία αναγνώριση στο τομέα της τεχνολογίας από τη δύση, μιας και θεωρούνταν μόνο σα χώρα μαζική παραγωγής με περιορισμένες δυνατότητες ενδογενούς ανάπτυξης νέων τεχνολογιών.

Τα κυρίαρχα πρότυπα ήχου και βίντεο , MPEG-4 και H.264, είναι ευρέως χρησιμοποιούμενα σε καταναλωτικές συσκευές αναπαραγωγής πολυμέσων , όπως για παράδειγμα το DVD player. Η χρησιμοποίηση τους από κινέζους κατασκευαστές audio/video συσκευών απαιτούσε την πληρωμή υπέρογκων ποσών για royalty fees σε ξένες εταιρίες για IP (intellectual property) που χρησιμοποιούν στις συσκευές. Ενδεικτικά, το κόστος μιας άδεια χρήσης κυμαίνεται από 2.5$ έως 4$ ανά συσκευή το οποίο ανέρχεται στο 10% του κόστους κατασκευής ενός DVD player.

Όσον αφορά στο τεχνολογικό τομέα, η αποκωδικοποίηση βίντεο, όπως ορίζεται στο AVS standard (με πολλές ομοιότητες με αντίστοιχα video standards), εκτελείται για κάθε frame και το βασικό στοιχείο αποτελεί ένα macroblock (μια υποπεριοχή



Εικόνα 4-1: Βασική δομή κωδικοποίησης

βίντεο μεγέθους 16x16 pixels). Αντίστοιχα με παλαιότερα και με τα υπόλοιπα σύγχρονα video standards, βασίζεται στις ίδιες αρχές του block-based intra και inter prediction, καθώς και του ίδιου transform-based coding framework. Το intra prediction υπολογίζεται βάσει γειτονικών pixels στο από πάνω, πάνω αριστερά, πάνω δεξιά και αριστερό macroblock. Το AVS standard καθορίζει και υποστηρίζει τέσσερα μεγέθη block (8x8, 8x16, 16x8, 16x16) για inter prediction, σε αντίθεση με το H.264, το οποίο υποστηρίζει μεγέθη block ακόμη και μεγέθους 4x4. Αυτή ήταν μια συνειδητή επιλογή των σχεδιαστών του AVS standard, που είχε ως σκοπό να μειώσει την υπολογιστική πολυπλοκότητα που θα υπεισερχόταν με τη χρήση block 4x4, αν και αυτό θα είχε ως αντιστάθμισμα επιδείνωση της ποιότητας του αποκωδικοποιημένου βίντεο. Η ακρίβεια των motion vectors, από την άλλη, είναι ¼ pixel. Τέλος, ο AVS χρησιμοποιεί φίλτρο deblocking στο αποκωδικοποιημένο frame, ώστε να ελαττωθούν τα blocking artifacts στις ακμές των blocks, ειδικά σε περιοχές χαμηλής χωρικής συχνότητας. Το φίλτρο αυτό επανακαθορίζεται αυτόματα ανάλογα με το περιεχόμενο του βίντεο και την τιμή Qp του κβαντιστή κάθε block, τα οποία και έχουν επίπτωση στο ποσοστό δημιουργίας blocking artifacts.Αυτό είναι και με το οποίο θα ασχοληθεί η παρούσα εργασία.

4.2 Βασικές Έννοιες βίντεο

YCbCr

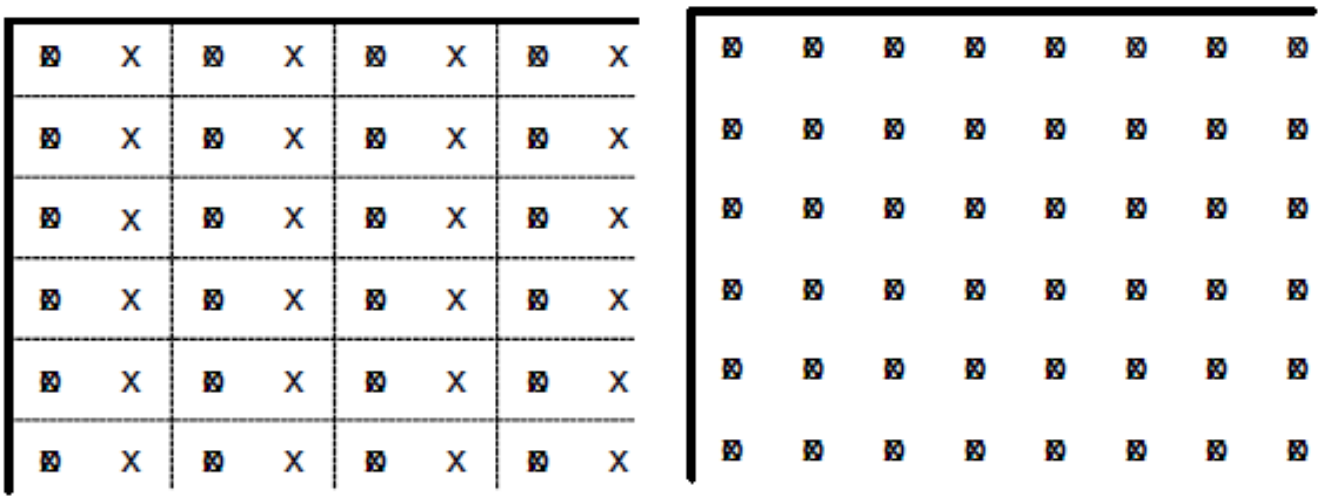
Το YCbCr είναι μια οικογένεια χρωματικών χώρων, το οποίο χρησιμοποιείται ευρεία στο βίντεο και στις φωτογραφίες. Το Y είναι το luma component και το Cb και Cr είναι το blue-difference και red-difference chroma components. Το luma αναπαριστά τη φωτεινότητα μιας εικόνας, μας δίνει μια αχρωματική εικόνα. Ενώ το chroma μας δίνει την πληροφορία του χρώματος της εικόνας και μαζί με το Y μας δίνουν όλη την εικόνα. Ο λόγος που γίνεται χρήση ενός τέτοιου σχήματος είναι γιατί το ανθρώπινο μάτι επηρεάζεται περισσότερο από τη φωτεινότητα παρά το χρώμα, μειώνοντας έτσι το ποσό δεδομένων που χρειάζεται για αποθήκευση.

YCbCr Sampling Formats

Τα sampling formats που καθορίζονται από το AVS standard είναι τρία, και απεικονίζονται στις εικόνες που έπονται. Κάθε εικόνα βίντεο περιέχει τρεις συνιστώσες: μία για το luma[3] και δύο για το chroma . Κάθε εικόνα, όπως είναι επίσης γνωστό, αποτελείται από pixels. Όταν λοιπόν οι τρεις προαναφερθείσες συνιστώσες έχουν την ίδια ανάλυση, και επομένως κάθε μια εκ των τριών συνιστωσών εμφανίζεται σε κάθε pixel, τότε μιλάμε για 4:4:4 sampling format. Όταν το chroma (Cb και Cr) περιορίζονται στο μισό στην οριζόντια διάσταση, τότε μιλάμε

[3] Η αλλιώς luminance και chrominance αντίστοιχα

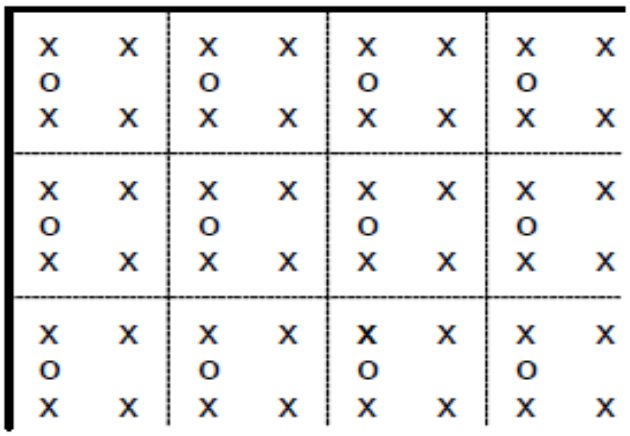
για το 4:2:2 sampling. Οι αριθμοί μπορούν να ερμηνευτούν ως η σχετική συχνότητα του κάθε στοιχείου ως προς τα υπόλοιπα. Για παράδειγμα στο 4:2:2 για κάθε τέσσερα luminance samples, υπάρχουν από δύο Cb και Cr, αντίστοιχα. Οι δυο αυτές περιπτώσεις απεικονίζονται στην Εικόνα 4-2.



Εικόνα 4-2: 4:2:2 sampling και 4:4:4 sampling (όπου 'x' το luma sample και 'o' τα chroma samples Cb,Cr)

Από τα δυο προαναφερθέντα formats, στο πρώτο διατηρείται η πλήρης πιστότητα των χρωμάτων, ενώ το δεύτερο αν και υπολείπεται του πρώτου, θεωρείται format υψηλής πιστότητας χρωμάτων. Αυτό ισχύει, δεδομένου του τρόπου με τον οποίο το ανθρώπινο μάτι αντιλαμβάνεται τα χρώματα, όπου τη μεγαλύτερη επίδραση έχει η φωτεινότητα (luminance) και όχι οι συνιστώσες του chrominance.

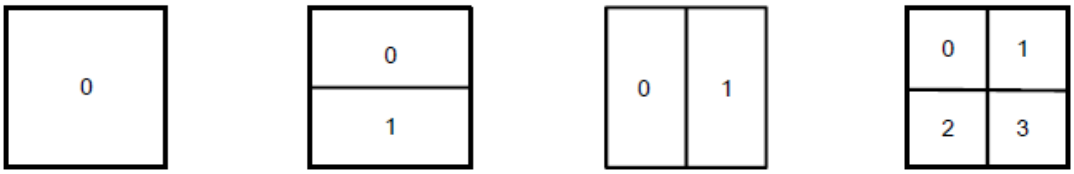
Η τρίτη και πιο συνηθισμένη περίπτωση είναι το 4:2:0 sampling format, όπου η σχετική συχνότητα εμφάνισης των Cb και Cr είναι η μισή, τόσο στην οριζόντια, όσο και στην κάθετη κατεύθυνση. Στην περίπτωση αυτού του format, η ονομασία του δεν ακολουθεί το πρότυπο που αναφέραμε πιο πάνω, αλλά έχει επικρατήσει παραδοσιακά και δε θα πρέπει να συγχέεται. Είναι αυτό το format το οποίο χρησιμοποιείται κατά κανόνα σε εφαρμογές videoconference, στην ψηφιακή τηλεόραση, καθώς και στο DVD video. Μια σχηματική απεικόνιση μπορούμε να δούμε στην εικόνα 4-3.



Εικόνα 4-3 : *4:2:0* sampling (όπου 'x' το luma sample και 'o' τα chroma samples Cb,Cr)

Macroblock

Κάθε εικόνα ενός video sequence χωρίζεται σε ένα αριθμό από macroblocks, τα οποία αντιπροσωπεύουν ορθογώνιες περιοχές εντός της εικόνας. Για τη χρήση στο motion compensation, κάθε macroblock δύναται να χωριστεί σε μικρότερα blocks, τα οποία μπορεί να έχουν μέγεθος 16x16, 16x8, 8x16 και 8x8 pixels. Σχηματικά, οι τρόποι διαχωρισμού εμφαίνονται στην παρακάτω εικόνα:

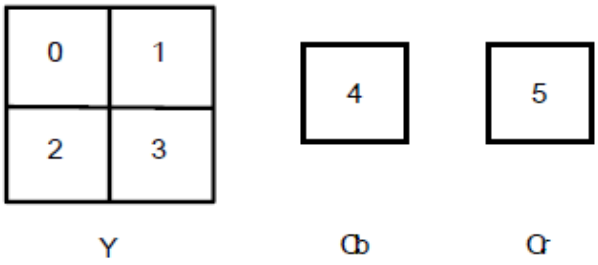


Εικόνα 4-4

8x8 Block

Το block 8x8 είναι το βασικό στοιχείο του μετασχηματισμού και του block scan, που μπορεί να παριστάνει δεδομένα της αρχικής εικόνας, δεδομένα της reconstructed εικόνας, ή 8x8 δεδομένα του ακέραιου μετασχηματισμού.

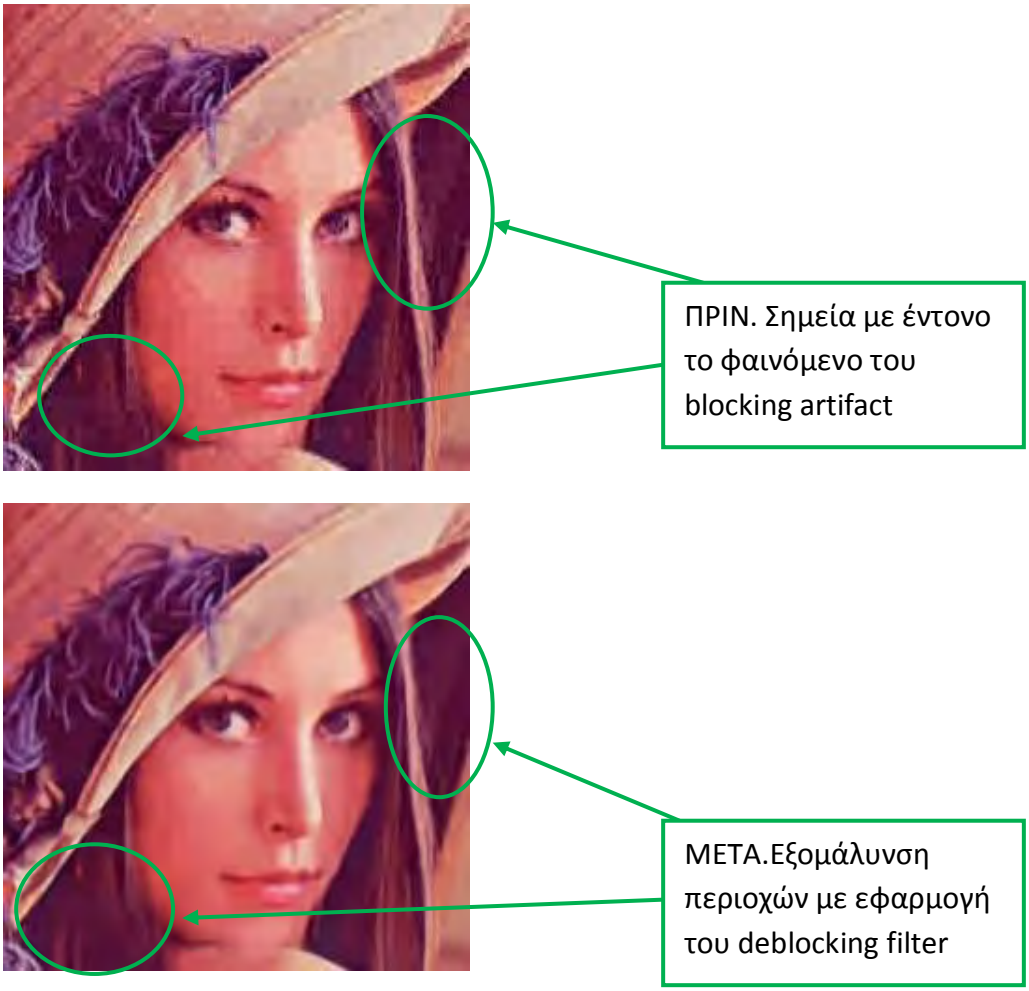
Στο 4:2:0 format που υποστηρίζει μέχρι στιγμής το profile του AVS codec, το macroblock περιλαμβάνει τέσσερα 8x8 luminance blocks (Y) και δυο chrominance blocks (Cb και Cr), όπως φαίνεται στο παρακάτω σχήμα:



Εικόνα 4-5

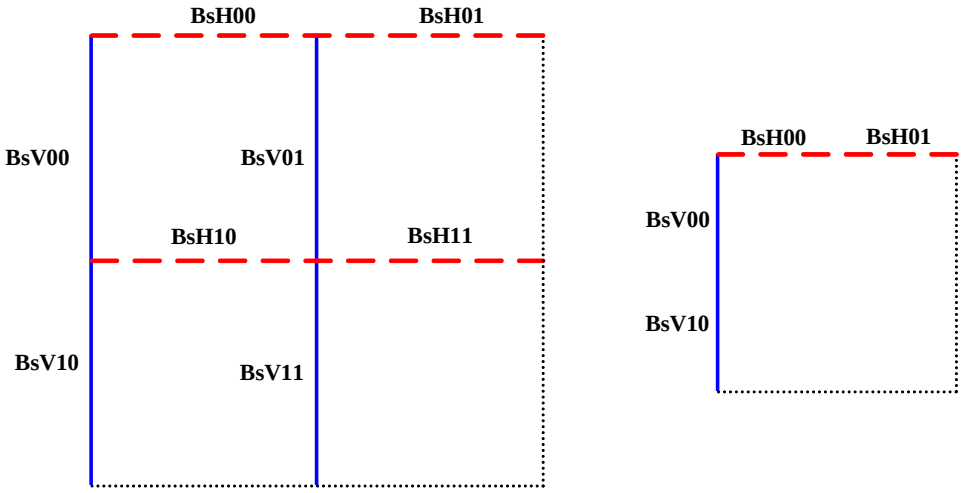
4.3 Αλγόριθμος Deblocking Filter

Το **deblocking filter** εφαρμόζεται σε block αποκωδικοποιημένο βίντεο και έχει ως σκοπό να βελτιώσει τη ποιότητας της εικόνας με την εξομάλυνση των έντονων ακμών οι οποίες δημιουργούνται μεταξύ macroblock όταν έχουν χρησιμοποιηθεί τεχνικές κωδικοποίησης block.



Εικόνα 4-6: Παραδείγματα blocking artifact που εξαλείφονται με το deblocking filter

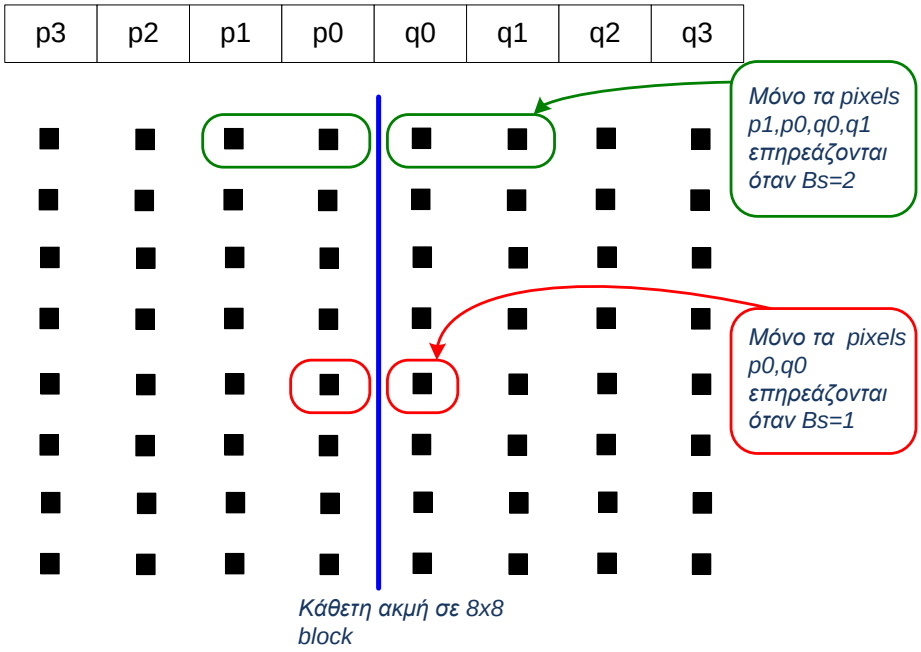
Πιο ειδικά το deblocking filter είναι ένα βαθυπερατό φίλτρο που εφαρμόζεται στα όρια ενός 8x8 block ως το τελευταίο βήμα της φάσης αποκωδικοποίησης. Εφαρμόζεται πριν το αποκωδικοποιημένο block από pixel αποθηκευτεί στη κύρια μνήμη. Χρησιμοποιείται για να εξομαλύνει τις ακμές του



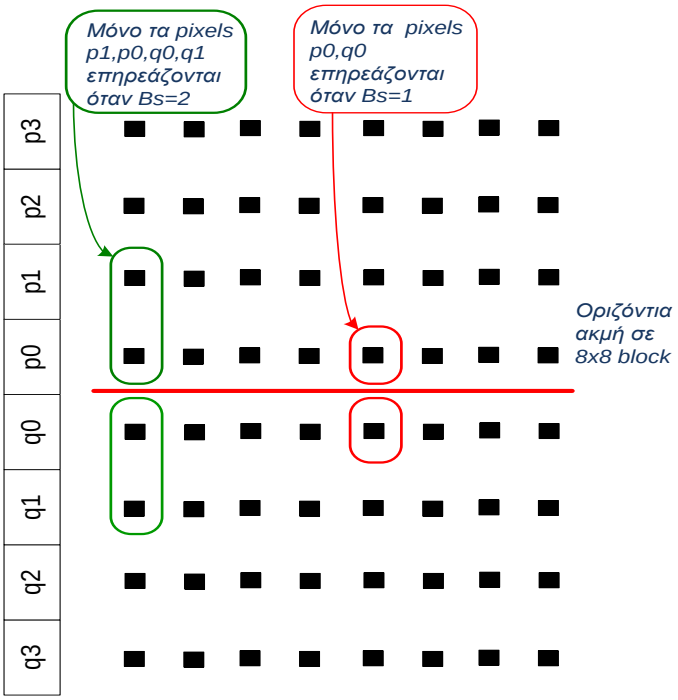
Εικόνα 4-7: Ακμές εφαρμογής του deblocking filter , αριστερά είναι ένα luma macroblock και δεξιά ένα chroma block

block και να βελτιώσει την εμφάνιση του αποκωδικοποιημένου καρέ (frame) σε περιοχές της εικόνας με χαμηλή *χωρική συχνότητα*[4]. Το φιλτράρισμα γίνεται σε 2 στάδια, πρώτα γίνεται κατά μήκος των κάθετων ακμών, δηλαδή με τη σειρά V00-V10-V01-V11, και μετά κατά μήκος των οριζόντιων ακμών, δηλαδή με τη σειρά H00-H01-H10-H11, κάθε block 8x8. Στην εικόνα βλέπουμε τις περιοχές όπου εφαρμόζεται το φίλτρο για luma macroblock 16x16(Εικόνα 4-7 αριστερά) και chroma block 8x8(Εικόνα 4-7 δεξιά).Πιο αναλυτικά η Εικόνα 4-8 δείχνει την πάνω γραμμή του τα τρέχοντος 8x8 luma block B(c, r)[5] και τις κάτω γραμμές του luma block B(c, r-1) τα οποία επηρεάζονται από το deblocking filter ανάλογα με την τιμή της παραμέτρου *Bs* , boundary strength (βαθμός φιλτραρίσματος). Αυτή είναι μια παράμετρος η οποία εξαρτάται από τη διαφορά των τρόπων κωδικοποίησης, τα διανύσματα κίνησης (motion vectors) ή τις παραμέτρους κβαντοποίησης μεταξύ των δύο blocks στις δύο πλευρές της μεταξύ τους ακμής. Η παράμετρος *Bs* καθορίζει το βαθμό του φιλτραρίσματος πρέπει να γίνει. Οι τιμές που μπορεί να πάρει είναι 0 (καθόλου φιλτράρισμα), 1 (μεσαίο φιλτράρισμα) και 2 (πλήρες φιλτράρισμα) όταν υπάρχει τουλάχιστον ένα Intra block .

[4] Περιοχές στο καρέ μιας εικόνας όπου έχουμε μικρή μεταβολή του περιεχομένου
[5] B(c, r) = Block (στήλη , γραμμή)



Εικόνα 4-8: Εφαρμογή deblocking filter σε κάθετη ακμή ενός 8x8 block



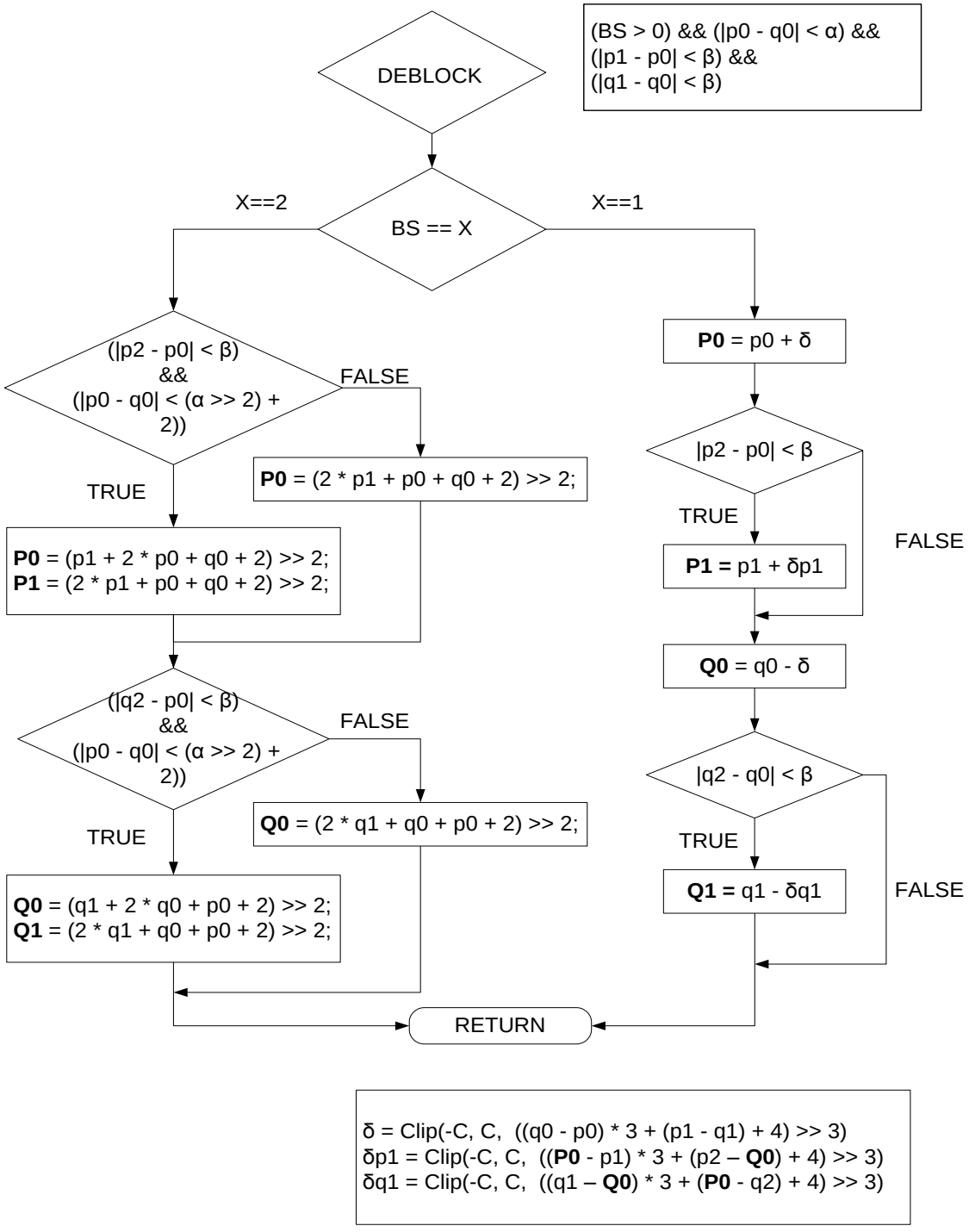
Εικόνα 4-9: Εφαρμογή deblocking filter σε οριζόντια ακμή ενός 8x8 block

Στις εικόνες 4-10 και 4-11 έχουμε το διάγραμμα ροής του αλγορίθμου που παράγει τα pixel εξόδου για το luma και το chroma. Ο αλγόριθμος πρέπει να καλεστεί 8 φορές για να υπολογίσει τα pixel εξόδου μιας ακμής ενός 8x8 block, η 16 φορές για υπολογίσει τα pixel εξόδου μιας ακμής ενός 16x16 macroblock.

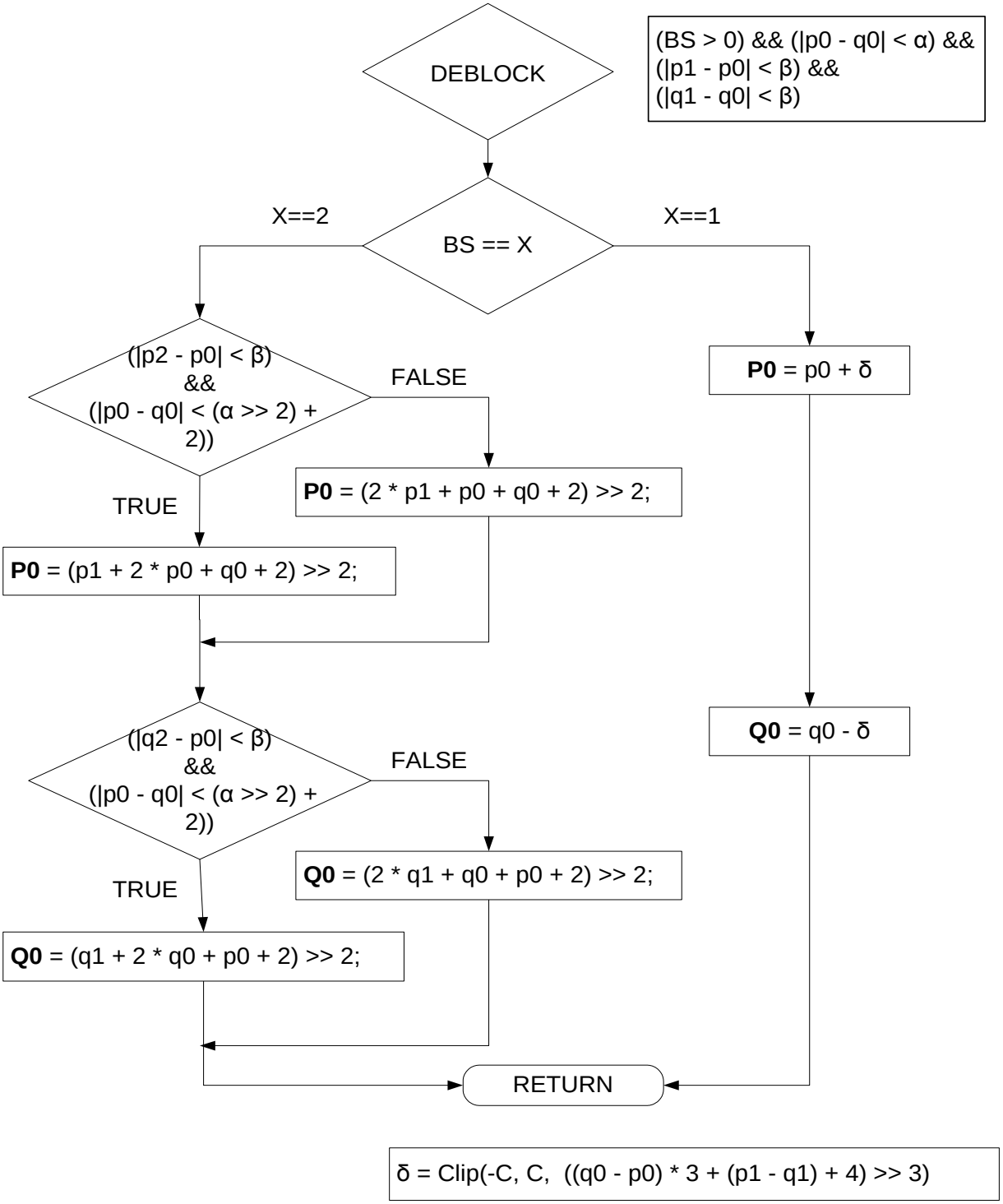
Ο αλγόριθμος που παρουσιάζεται παρακάτω στα διαγράμματα ροής χρειάζεται τις ακόλουθες εισόδους:

- Τα έξι pixel *p0, p1, p2,* και *q0, q1, q2* γύρω από την ακμή όπου εφαρμόζεται το deblocking filter, όπως αυτά φαίνονται και στις εικόνες 4-2 και 4-3 για κάθε περίπτωση.
- Το *BS* (Boundary Strength) για κάθε ακμή του block με τις δυνατές τιμές που αναφέρθηκαν παραπάνω. Σε ένα luma macroblock έχουμε 4 τιμές BS για τις κάθετες ακμές του και 4 τιμές BS για τις οριζόντιες ακμές του. Αντίστοιχα για ένα chroma block έχουμε 2 τιμές BS, 1 για την κάθετη ακμή και μία για την οριζόντια.
- Τις παραμέτρους α και β οι οποίες υπολογίζονται βάσει των παραμέτρων στην επικεφαλίδας της εικόνας *FilterOffsetA* και *FilterOffsetB* και της μέσης τιμής των παραμέτρων κβαντοποίησης των block γύρω από την ακμή. Κάθε macroblock μπορεί να έχει στο σύνολο 4 διαφορετικές τιμές για κάθε παράμετρο α και β. Συγκεκριμένα από την εικόνα 4-7 το ζευγάρι V00 και V10 , έχουν τις ίδιες τιμές για το α και β, *το* ίδιο και τα V01 και V11, H00 και H01, και τέλος το H10 και H11. Αυτό οφείλεται επειδή οι εξωτερικές τα συγκεκριμένα ζευγάρια έχουν τις ίδιες παραμέτρους. Οι δυο παράμετροι α και β λειτουργούν ως κατώφλια για τις διάφορες επιλογές του deblocking αλογορίθμου.
- Την παράμετρο αποκοπής C η οποία εξαρτάται από το *FilterOffsetA* και την μέση τιμή των τιμών κβαντοποίησης των blocks κατά μήκος μιας ακμής άρα ένα macroblock έχει 4 διαφορετικές τιμές. Η τιμή του C, η οποία δίνεται από ένα πίνακα, χρησιμοποιείται για να αποκόψει το δέλτα offset το οποίο εφαρμόζεται στα pixel γύρω από τις ακμές όταν BS=1.

ΔΙΑΓΡΑΜΜΑΤΑ ΡΟΗΣ ΑΛΓΟΡΙΘΜΟΥ



Εικόνα 4-10: Διάγραμμα ροής του deblocking filter για το luma

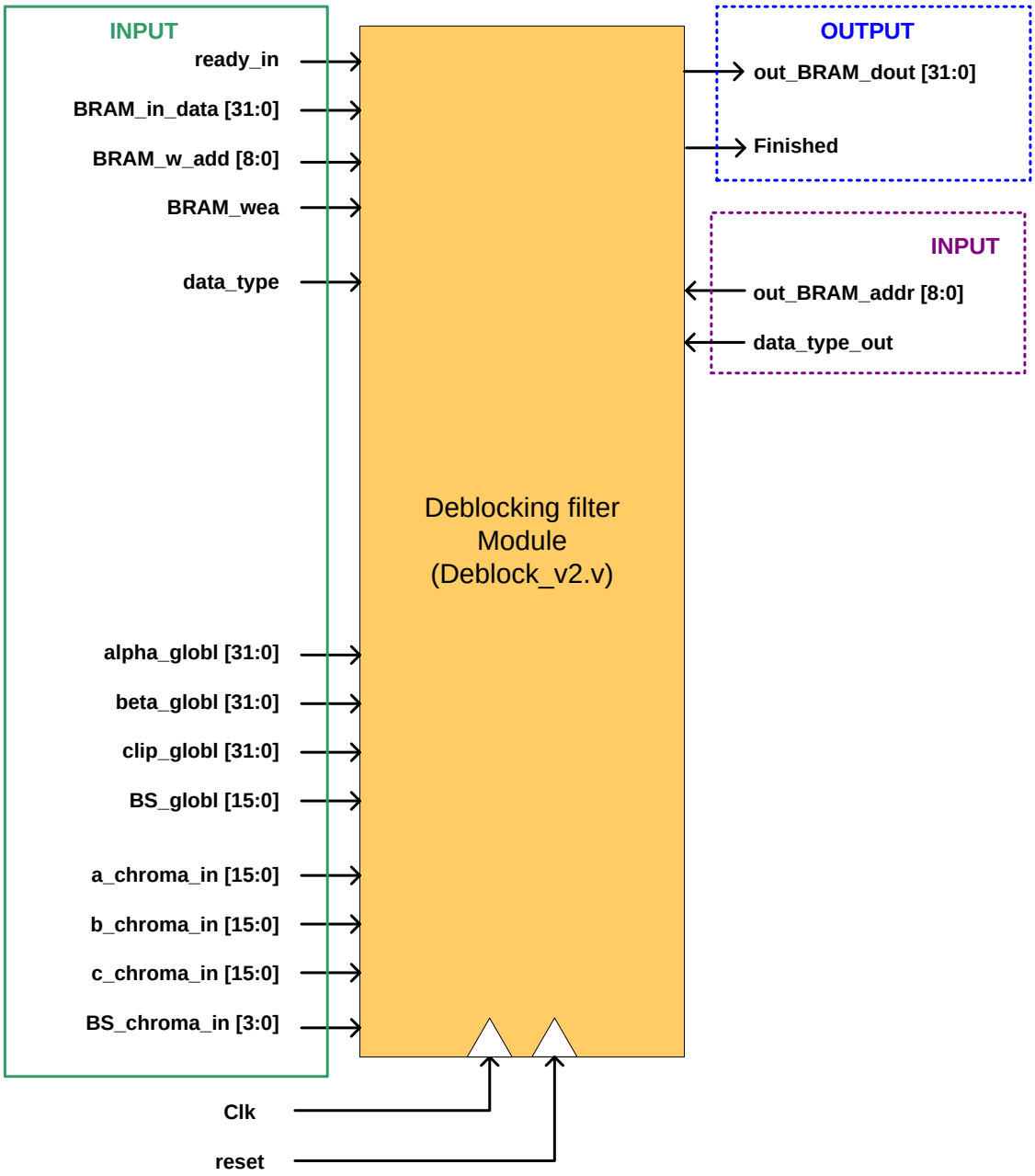


Εικόνα 4-11: Διάγραμμα ροής του deblocking filter για το chroma

5. ΥΛΟΠΟΙΗΣΗ HARDWARE MODULE - DEBLOCKING FILTER

5.1 Προδιαγραφές hardware module

Επισκόπηση



Εικόνα 5-1: Σχηματική αναπαράσταση Deblocking filter Module

Η εξωτερική αναπαράσταση του κυκλώματος που απεικονίζεται παραπάνω(Εικόνα 5-1) που υλοποιήθηκε στα πλαίσια της παρούσας εργασίας.Η συγκεκριμένη μονάδα δέχεται σαν είσοδο ένα luma(Y) macroblock μαζί με τα απαραίτητα γύρω δεδομένα και τα 2 block των chroma(Cb , Cr) μαζί με τις απαραίτητες γειτονικές τιμές για να εφαρμοστεί το φίλτρο σε καθένα από αυτά. Επίσης παίρνει σαν είσοδο τις παραμέτρους που είναι απαραίτητες για την εκτέλεση. Όταν ολοκληρωθεί η είσοδος όλων των απαραίτητων δεδομένων η είσοδος ready_in γίνεται 1. Αντίστοιχα όταν τελειώσει η εφαρμογή του Deblocking filter τότε ενεργοποιείται το σήμα εξόδου finished και δίνοντας τιμές στις διευθύνσεις τις εσωτερικής μνήμης παίρνουμε τις τιμές με τα φιλτραρισμένα στοιχεία.

Συγκεντρωτικά:

Όνομα σήματος	Εισόδου(IN) / Εξόδου (OUT)	Εύρος	Περιγραφή
ready_in	IN	1	Σήμα εισόδου που όταν ενεργοποιείται σημαίνει ότι όλα τα απαραίτητα στοιχεία έχουν εισαχθεί στην εσωτερική μνήμη του και μπορεί να ξεκινήσει η εκτέλεση του Deblocking Filter
BRAM_in_data	IN	32	Είσοδος που παρέχει τιμές στις εσωτερικές μνήμες και περιέχουν τιμές pixel των δεδομένων όπου θα εφαρμοστεί το Deblocking Filter.
BRAM_w_addr	IN	9	Διεύθυνση που αντιστοιχεί σε εσωτερικές διευθύνσεις του module και αφορά σχετικές θέσεις των pixel τα οποία τα έχουμε χωρίσει σε ομάδες των 4.
BRAM_wea	IN	1	Σήμα που ενεργοποιεί την εγγραφή στις εσωτερικές μνήμες
out_BRAM_dout	OUT	32	Έξοδος που μας δίνει τις νέες τιμές των pixel σε ομάδες των 4. Τα ονόματα είναι από ποια εσωτερική μνήμη μας έρχονται και άρα σε ποιο κομμάτι ανήκουν.
out_BRAM_addr	IN	9	Διεύθυνση που δίνεται για να διαβάσουμε τα περιεχόμενα από τις εσωτερικές μνήμες

Όνομα σήματος	Εισόδου(IN) / Εξόδου (OUT)	Εύρος	Περιγραφή
alpha_globl	IN	32	Είσοδο που περιέχει συνολικά όλες τις παραμέτρους α που απαιτούνται σε luma macroblock
beta_globl	IN	32	Είσοδο που περιέχει συνολικά όλες τις παραμέτρους β που απαιτούνται σε luma macroblock
clip_globl	IN	32	Είσοδο που περιέχει συνολικά όλες τις παραμέτρους C που απαιτούνται σε luma macroblock
BS_globl	IN	16	Είσοδο που περιέχει συνολικά όλες τις τιμές Boundary Strength που απαιτούνται σε luma macroblock
finished	IN	1	Σήμα που όταν γίνει 1, σηματοδοτεί την ολοκλήρωση του υπολογισμού.
a_chroma_in	IN	16	Είσοδο που περιέχει συνολικά όλες τις παραμέτρους α που απαιτούνται σε chroma block
b_chroma_in	IN	16	Είσοδο που περιέχει συνολικά όλες τις παραμέτρους β που απαιτούνται σε chroma block
c_chroma_in	IN	16	Είσοδο που περιέχει συνολικά όλες τις παραμέτρους C που απαιτούνται σε chroma block
BS_chroma_in	IN	4	Είσοδο που περιέχει συνολικά όλες τις τιμές Boundary Strength που απαιτούνται σε chroma block
clk	IN	1	Είσοδο παλμού ρολογιού
reset	IN	1	Είσοδος επαναφοράς στην αρχική κατάσταση

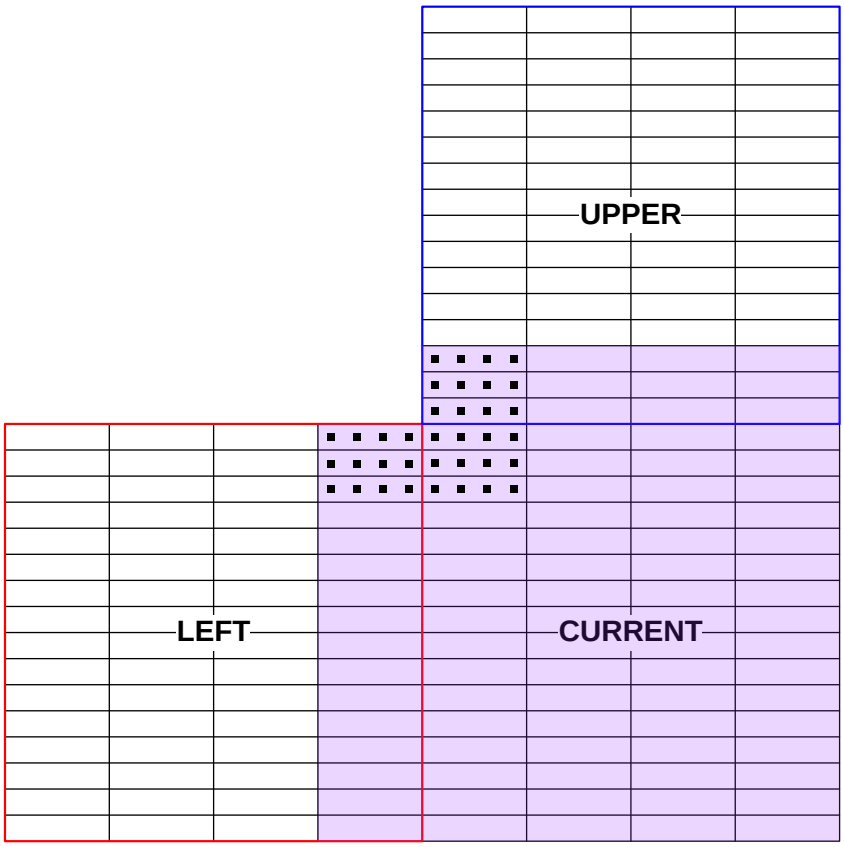
Δεδομένα Εισόδου

Τα δεδομένα εισόδου πρέπει να χωριστούν κατάλληλα για να μπορέσει το module (κύκλωμα) να τα χειριστεί κατάλληλα. Κάθε pixel έχει μέγεθος ένα byte, το μέγιστο πλάτος εισόδου των εσωτερικών μνημών BRAM του module είναι 4 bytes.Οπότε επιδιώκουμε να φέρνουμε ομάδες των 4 pixel. Επιπλέον τα δεδομένα είναι αποθηκευμένα σε κάποια εξωτερική μνήμη row-wise, το οποίο σημαίνει ότι στα δεδομένα ενός πίνακα τα στοιχεία αποθηκεύονται κατά τις σειρές[6], οπότε μας βολεύει να ομαδοποιήσουμε τις γραμμές σε ομάδες των τεσσάρων. Παρακάτω θα δούμε αναλυτικά τις περιπτώσεις για luma και chroma. Για την λειτουργία του κυκλώματος απαιτείται στην είσοδο ένα luma macroblock (Y) και 2 chroma block (Cb, Cr).

LUMA macroblock

Δεδομένου ότι για εκτελεστεί το deblocking filter θέλουμε το 3 pixel από το αριστερά/πάνω ενός ορίου και 3 από τα δεξιά/κάτω χρειαζόμαστε έτσι δεδομένα από το αριστερό macroblock (LEFT) και από το πάνω (UPPER) . Επειδή θέλουμε τον ελάχιστο αριθμό δεδομένων που απαιτούνται παίρνουμε μια ομάδα 4 pixel σε κάθε γραμμή από το αριστερό macroblock για την κάθετη περίπτωση και 3 ομάδες 4 pixel για κάθε στήλη από το άνωθεν macroblock(Βλέπε Εικόνα 5-2)

[6] Βλέπε παράρτημα Β

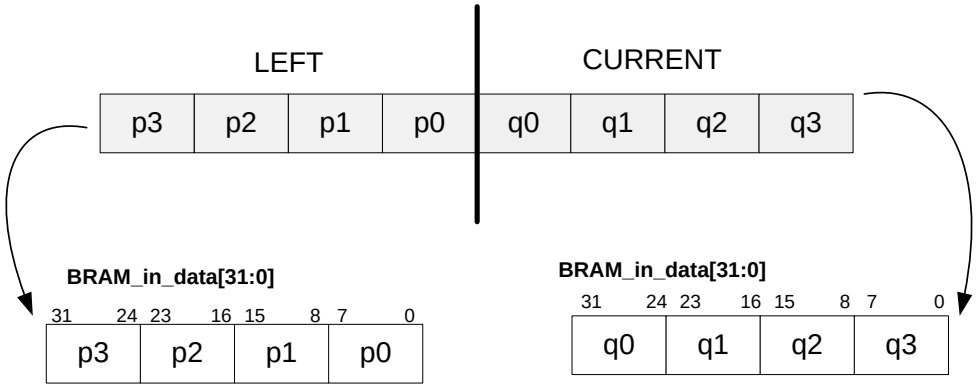


Εικόνα 5-2: Διαχωρισμός των luma macroblock και απαραίτητα δεδομένα

Οπότε λοιπόν φροντίζουμε να διοχετεύσουμε στο module τα δεδομένα στις κατάλληλες διευθύνσεις των εσωτερικών μνήμων. Συγκεκριμένα το module περιέχει 3 εσωτερικές μνήμες (block RAM) . Έχουμε μία BRAM που αποθηκεύει τα δεδομένα από τα γύρω macroblock (left και upper) , την οποία ονομάζουμε BRAM_P, έχουμε 2 BRAM για τα δεδομένα του τρέχοντος macroblock , τα οποία τα ονομάζουμε BRAM_Q03 και BRAM_Q12(βλέπε Εικόνα 5-3). Έχουμε σύνολο 3 BRAM σ τις οποίες αποθηκεύουμε με βέλτιστο τρόπο τα δεδομένα έτσι ώστε σε κάθε κύκλο ρολογιού να μπορούμε να κάνουμε ανάγνωση από 2 διαφορετικές BRAM. Αυτό το σχήμα μας εξυπηρετεί στο γεγονός ότι δεν χρειάζεται να περιμένουμε επιπλέον κύκλους για να φορτώσουμε όλα τα δεδομένα που απαιτούνται για να εκτελεστεί το deblocking filter, κάτι το οποίο θα συνέβαινε στην περίπτωση που τα αποθηκεύαμε όλα τα δεδομένα σε μια BRAM. Επομένως στη περίπτωση που πραγματοποιείται deblocking στον κάθετο άξονα μπορούμε να έχουμε μετά από μια ανάγνωση και μία εκτέλεση deblocking, ενώ στον οριζόντιο άξονα για 3 αναγνώσεις μπορούμε να έχουμε 3 εκτελέσεις deblocking.

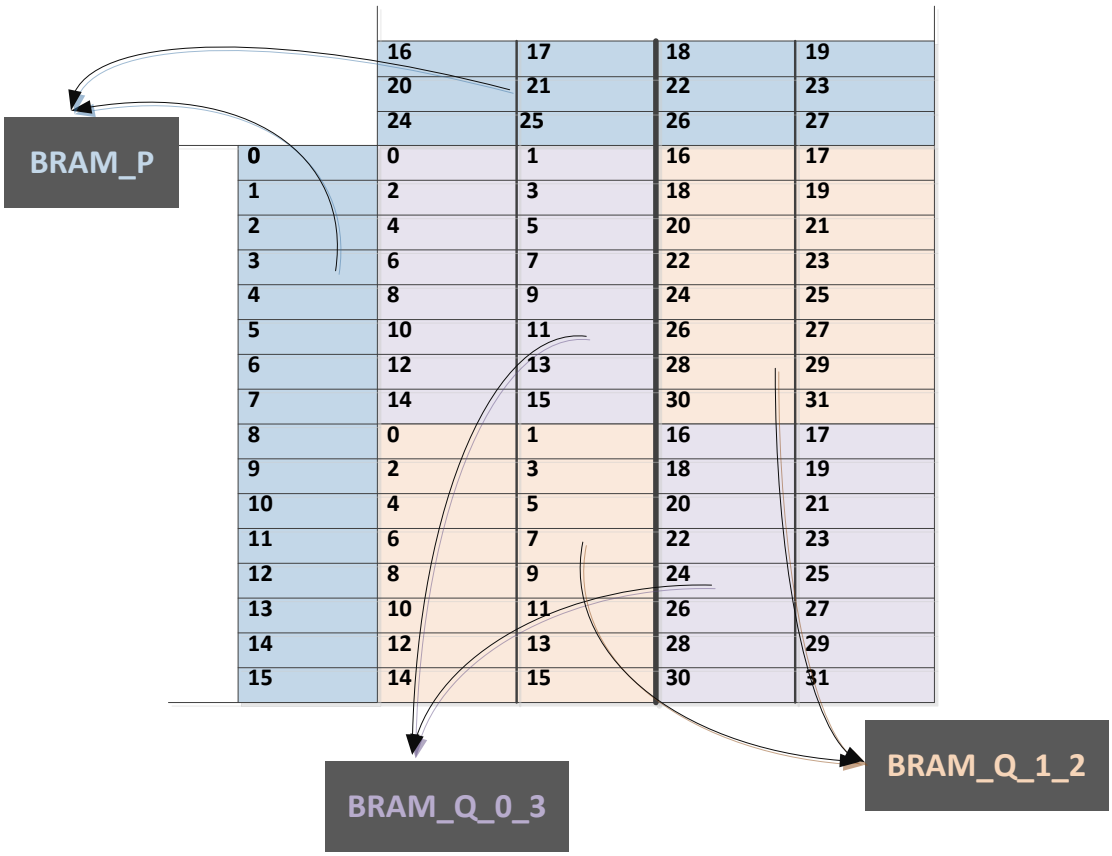
Επομένως αυτό που τροφοδοτούμε στην είσοδο του module είναι τα 4 byte δεδομένων και τη διεύθυνση που τα τοποθετούμε. Ανάλογα με τη σχετική θέση(σε ποιο macroblock ανήκουν βλέπε Εικόνα 5-3) των δεδομένων τα εισάγουμε στην είσοδο που αντιστοιχεί στην εκάστοτε BRAM θέτοντας τιμές στο data_type.

- 0 για δεδομένα στη BRAM_P
- 1 για δεδομένα στη BRAM_ Q03
- 2 για δεδομένα στη BRAM_ Q12



Εικόνα 5-3: Παράδειγμα αντιστοίχισης διαδοχικών byte στα σήματα εισόδου(περίπτωση κάθετης ακμής)

Σχηματικά έχουμε την παρακάτω κατανομή των δεδομένων στις BRAM μαζί με την αντίστοιχη διεύθυνση της εσωτερική μνήμη όπου θα τοποθετηθούν:



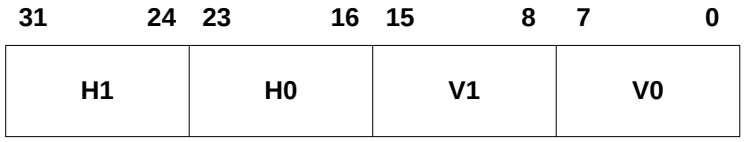
Εικόνα 5-4: Καταμερισμός δεδομένων στις εσωτερικές BRAM

Επιπλέον για κάθε deblocking ενός macroblock περνάμε μία φορά της παραμέτρους α, β, C και το Boundary Strength.

Αναλυτικά:

α, β ,C : Επειδή για κάθε macroblock όπως αναφέραμε στην παράγραφο 4.4 έχουμε το πολύ 4 διαφορετικές τιμές για το καθένα και οι τιμές που παίρνει το α είναι από 0 μέχρι 64 οπότε μας αρκούν 7 bit για την αναπαράσταση του στο δυαδικό ,το β από 0 μέχρι 27 οπότε αρκούν 5 bit για την αναπαράσταση του στο δυαδικό και το C από 0 μέχρι 9 οπότε αρκούν 4 bit για την αναπαράσταση του στο δυαδικό. Αναπαριστούμε το α σε εύρος 8 bit για να μπορέσουμε να καλύψουμε τις περιπτώσεις συγκρίσεων με αρνητικούς. Για λόγους συμμετρίας και για να είναι δύναμη του 2 θεωρούμε το ίδιο εύρος[7] και για το β. Επομένως έχουμε 3 εισόδους εύρους 32 bit οι οποίες περιέχουν όλες τις παραμέτρους για ένα macroblock.Οι οποίες τιμές αποθηκεύονται σε εσωτερικούς καταχωρητές του κυκλώματος.

Οπότε έχουμε την παρακάτω κατανομή για τις 32bit εισόδους ***alpha_globl, beta_globl*** και ***clip_globl***

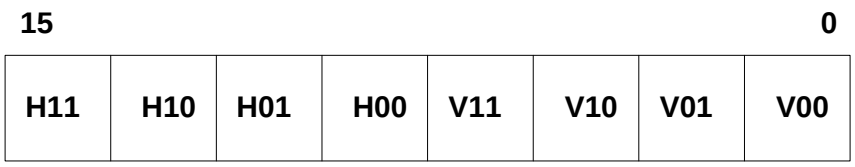


Όπου :

- V0 : Τιμή παραμέτρου στην **εξωτερική κάθετη** ακμή του macroblock
- V1 : Τιμή παραμέτρου στην **εσωτερική κάθετη** ακμή του macroblock.
- H0 : Τιμή παραμέτρου στην **εξωτερική οριζόντια** ακμή του macroblock
- H1 : Τιμή παραμέτρου στην **εσωτερική οριζόντια** ακμή του macroblock

Boundary Strength : Για κάθε macroblock έχουμε 8 τιμές boundary strength και επειδή οι δυνατές τιμές είναι 0,1 ή 2 μας αρκούν 2 bit για κάθε τιμή, άρα ένα σύνολο 16bit για όλο το macroblock

Η κατανομή της εισόδου ***BS_globl*** είναι η εξής

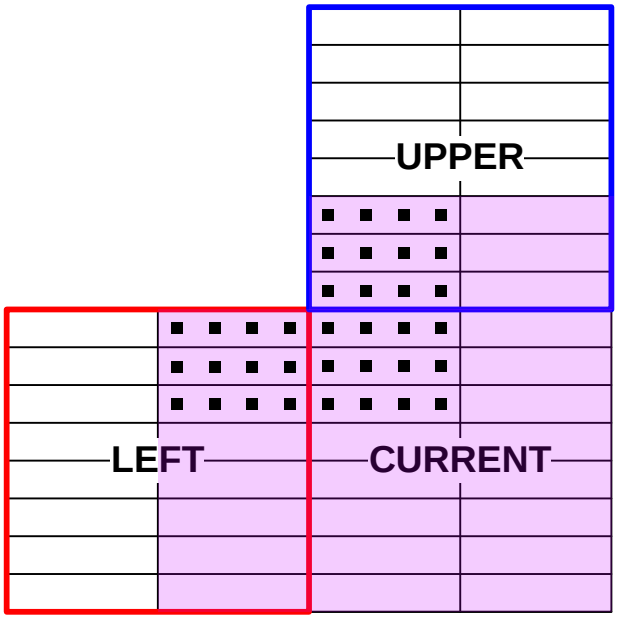


Όπου κάθε τιμή αντιστοιχεί στη θέση του macroblock όπως αυτή καθορίζεται στην Εικόνα 4-6 της παραγράφου 4.4.

[7] Σε περίπτωση αντιστοίχισης της εισόδου σε κάποια διεύθυνση μνήμης βολεύει να είναι δύναμη του 2

CHROMA block

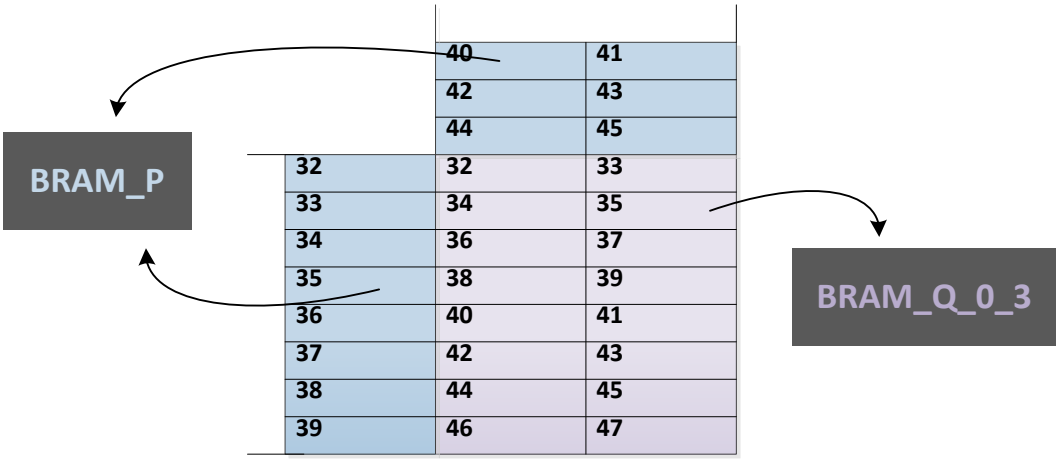
Αντίστοιχο με το διαχωρισμό στο των δεδομένων στο macroblock έχουμε και στο block.Αν και στο αλγόριθμο για το deblocking για το chroma χρειαζόμαστε μόνο 2 pixel από μεριά στα όρια της ακμής διατηρούμε το διαχωρισμό των pixel σε ομάδες των 4 μιας και οι εσωτερικές μνήμες διαβάζουν και αποθηκεύουν δεδομένα 4byte. Σχηματικά:



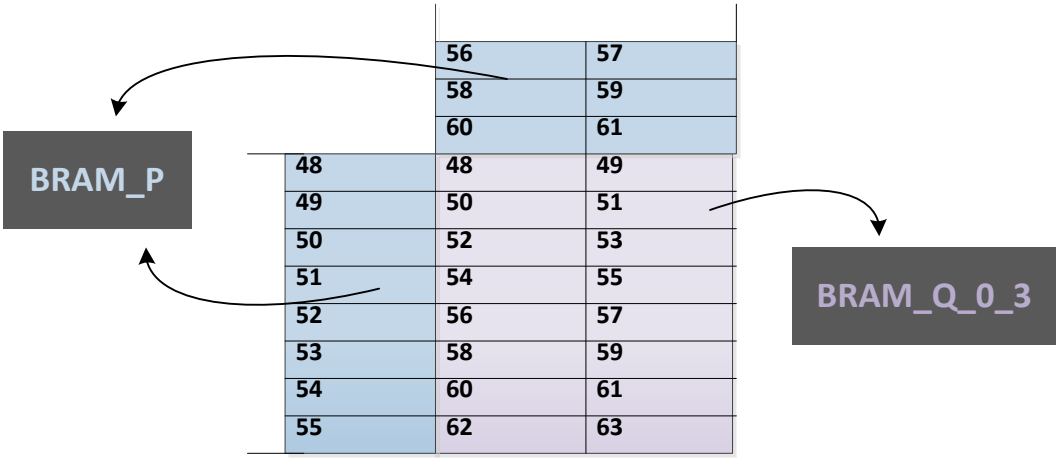
Εικόνα 5-5: Διαχωρισμός chroma block

Με παρόμοιο τρόπο χωρίζουμε τα δεδομένα στις εσωτερικές μνήμες, οι οποίες είναι οι ίδιες με παραπάνω απλά αποθηκεύουμε τα δεδομένα στις υπόλοιπες ελεύθερες θέσεις. Επειδή τα δεδομένα είναι λιγότερα αρκούν 2 BRAM για να μπορέσουμε με 1 κύκλο ανάγνωσης δεδομένων τα δεδομένα για να εκτελεστεί το deblocking στον επόμενο κύκλο (αντίστοιχα στην οριζόντια περίπτωση 3 κύκλους για 4 εκτελέσεις) .

Σχηματικά έχουμε τις παρακάτω κατανομές των δεδομένων ,ενός Cb chroma block και ενός Cr chroma block, στις BRAM μαζί με την αντίστοιχη διεύθυνση της εσωτερική μνήμη όπου θα τοποθετηθούν



Εικόνα 5-6: Κατανομή δεδομένων για Cb chroma



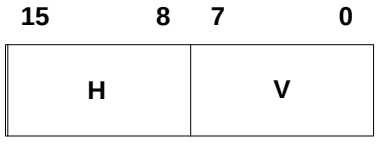
Εικόνα 5-7: Κατανομή δεδομένων για Cr chroma

Επιπλέον για κάθε deblocking ενός Cb και Cr Chroma περνάμε μία φορά της παραμέτρους α,β, C και το Boundary Strength, τα οποία είναι ίδια και για τα δύο.

Αναλυτικά:

α, β ,C :Τα χαρακτηριστικά των παραμέτρων είναι ίδια με αυτά του luma απλά είναι λιγότερα, μιας και απαιτούνται μόνο 2 (μια για την κάθετη ακμή και μια για τον οριζόντια ακμή). Επομένως έχουμε 3 εισόδους εύρους 16 bit οι οποίες περιέχουν όλες τις παραμέτρους για ένα macroblock.

Οπότε έχουμε την παρακάτω κατανομή για τις 16bit εισόδους ***a_chroma_in, b_chroma_in*** και ***c_chroma_in***

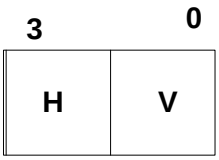


Όπου :

- V : Τιμή παραμέτρου στην **κάθετη** ακμή του block
- H : Τιμή παραμέτρου στην **οριζόντια** ακμή του block

Boundary Strength : Για τα 2 chroma block έχουμε 2 τιμές boundary strength και επειδή οι δυνατές τιμές είναι 0,1 ή 2 μας αρκούν 2 bit για κάθε τιμή, άρα ένα σύνολο 4bit για όλο το block

Η κατανομή της εισόδου ***BS_chroma_in*** είναι η εξής



Συγκεντρωτικά τα περιεχόμενα των BRAM

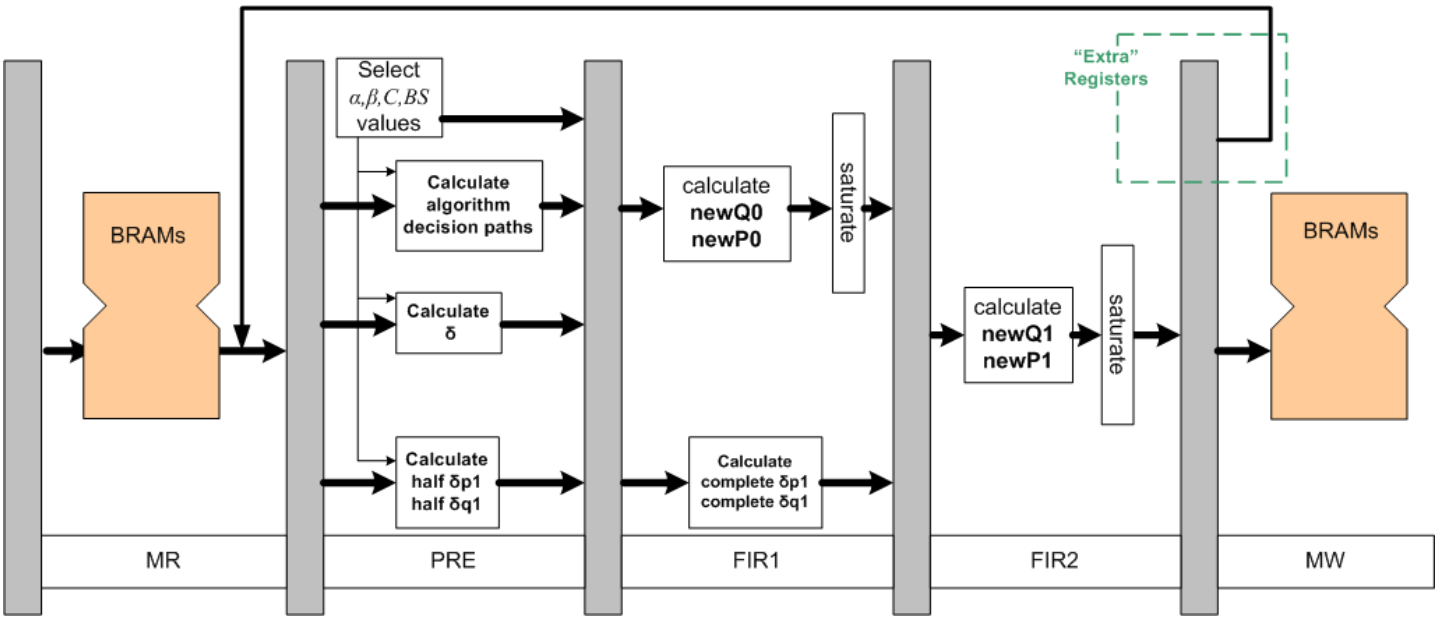
Τύπος	BRAM_P	BRAM_Q03	BRAM_Q12
Luma Y	0-27	0-31	0-31
Chroma Cb	32-45	32-47	-
Chroma Cr	48-61	48-63	-

Δεδομένα Εξόδου

Όταν τελειώσει η εφαρμογή του deblocking filter και ενεργοποιείται το σήμα **finished** και μπορεί να ξεκινήσει η ανάγνωση από τις εσωτερικές μνήμες εξόδου. Η δομή των BRAM εξόδου είναι ακριβώς οι ίδιες με αυτές τις εισόδου. Για να διαβαστούν οι τιμές των μνημών αυτών απαιτείται να δοθούν οι τιμές των διευθύνσεων. Οι οποίες διευθύνσεις είναι αντίστοιχες αυτών που καθορίζονται στην προηγούμενη παράγραφο και πρέπει να δίνοντας κατάλληλη τιμή στο data_type_out διαβάζονται τα περιεχόμενα τους.

5.2 Αρχιτεκτονική

Γενικά



Εικόνα 5-8: Αρχιτεκτονική 5 σταδίων παροχέτευσης

Η αρχιτεκτονική του κυκλώματος που φαίνεται στην εικόνα 5-8 αποτελείται είναι μία απλοποιημένη απεικόνιση του κυκλώματος το οποίο αποτελείται από 5-στάδια παροχέτευσης πλήρως επικαλυπτόμενα. Επομένως χρειάζονται 5 κύκλοι για να γίνει η εφαρμογή του deblocking filter στα όρια μιας ακμής για κάθε γραμμή στην κάθετη περίπτωση και σε κάθε επόμενο κύκλο διοχετεύουμε την επόμενη ομάδα pixel προς φιλτράρισμα. Στην οριζόντια περίπτωση χρειαζόμαστε 9 κύκλους για να εφαρμόσουμε το deblocking filter σε ομάδα 4 στηλών μιας και χρειαζόμαστε επιπλέον 2 αναγνώσεις και 2 αποθηκεύσεις για τα απαραίτητα δεδομένα, οπότε κάθε 3 κύκλους διοχετεύουμε την επόμενη ομάδα προς φιλτράρισμα. Οπότε τα στάδια είναι τα εξής

- **MR:** (Memory Read) Στάδιο όπου διαβάζονται τα απαραίτητα δεδομένα από τις BRAM και αποθηκεύονται σε ενδιάμεσους καταχωρητές για να χρησιμοποιηθούν από το επόμενο στάδιο. Στην κάθετη περίπτωση το φιλτράρισμα γίνεται σε 1 κύκλο. Στην οριζόντια περίπτωση η ανάγνωση γίνεται σε 3 κύκλους.
- **PRE:** (Precalculation) Σε αυτό το στάδιο έχουμε τον υπολογισμό των μονοπατιών στην εκτέλεση του αλγορίθμου (βλέπε Εικόνα 4-9 και 4-10), επιλογή των κατάλληλων παραμέτρων για τα στοιχεία όπου θα

εφαρμόσουμε το deblocking filter και υπολογισμός τιμών για το φιλτράρισμα όπως είναι το δ και μερικώς τα δp1 και δq1.

- **FIR1:** (Filtering 1) Πρώτο στάδιο εφαρμογής του deblocking filter όπου υπολογίζουμε τα νέα P0 και Q0 (πρώτα pixel στα όρια της ακμής, βλέπε εικόνα 4-7 και 4-8)
- **FIR2:** (Filtering 2) Δεύτερο στάδιο εφαρμογής του deblocking filter όπου υπολογίζουμε τα νέα P1 και Q1 (δεύτερα pixel στα όρια της ακμής, βλέπε εικόνα 4-7 και 4-8)
- **MW:** (Memory Write) Τελευταίο στάδιο όπου αποθηκεύονται οι νέες τιμές των pixel στις BRAM εξόδου. Στην κάθετη περίπτωση το φιλτράρισμα γίνεται σε 1 κύκλο. Στην οριζόντια περίπτωση η ανάγνωση γίνεται σε 3 κύκλους.

Τα παραπάνω στάδια είναι τα ίδια και για την περίπτωση του luma και των 2 chroma. Στην περίπτωση των chroma με κατάλληλα σήματα δεν γίνεται καμία αλλαγή στα P1 και Q1 και απλά οι αρχικές τιμές αντιγράφονται στην έξοδο.

Ο λόγος για τον οποίο επιλέχθηκε η παραπάνω αρχιτεκτονική είναι για να μπορέσουμε να πετύχουμε την απαιτούμενη συχνότητα λειτουργίας των 160MHz κάνοντας τους μέγιστους δυνατούς υπολογισμούς, χωρίς να επεκταθούμε πολύ στην ποσοστό χρήσης των πόρων.

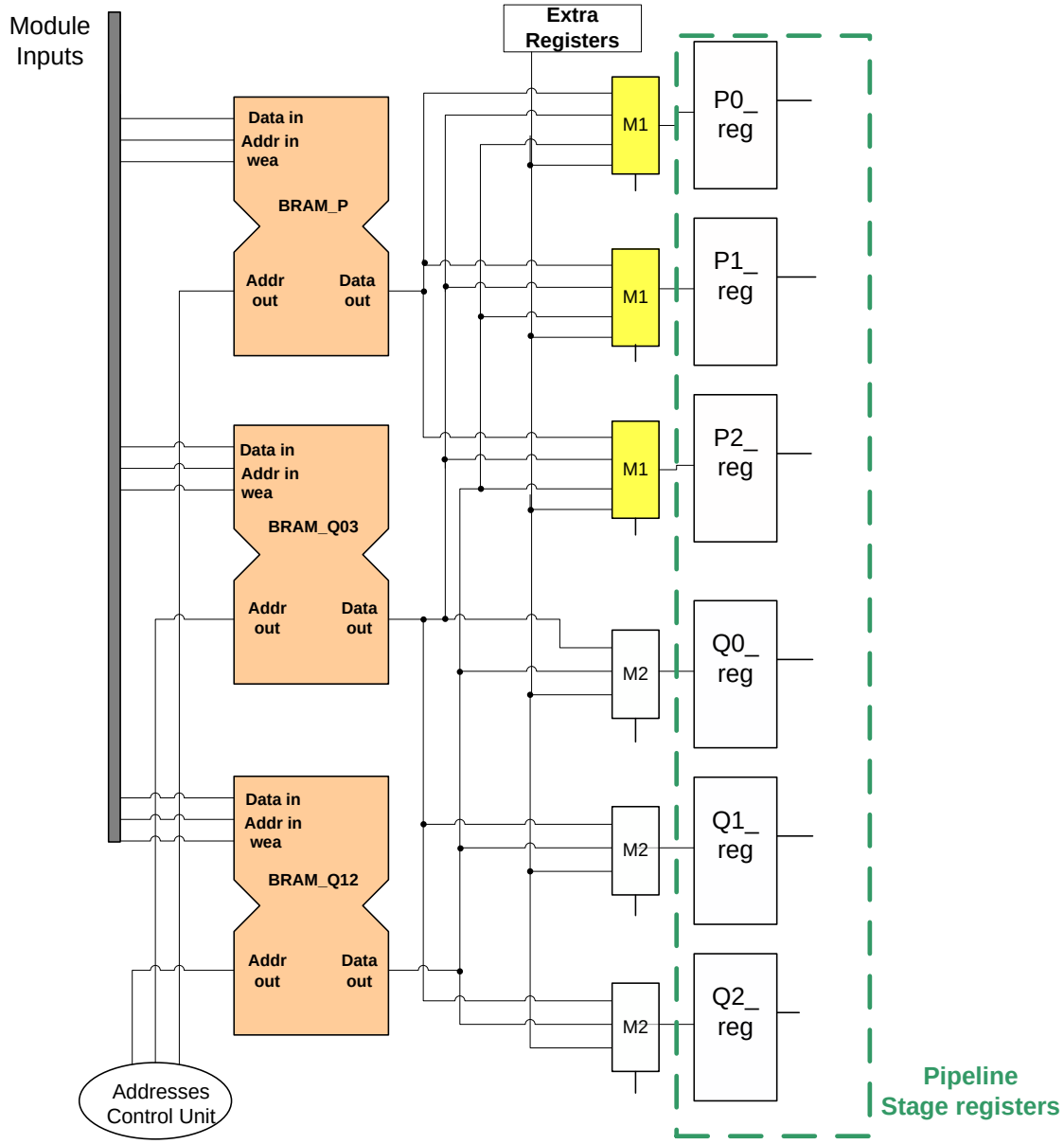
Επιπλέον για λόγους πληρότητας και συμμετρίας αποθηκευμένων δεδομένων αντιγράφονται απευθείας τα pixel στα οποία δεν γίνεται καμία αλλαγή. Άρα τα δεδομένα αντιγράφονται από τις BRAM εισόδου στις BRAM εξόδου χωρίς καμία αλλαγή και παρακάμπτουν τη συμβατική παροχέτευση.

Τα δεδομένα είναι:

BRAM	Luma	Chroma Cb	Chroma Cr
BRAM_P	-	-	-
BRAM_Q03	23,25,27,29,31	39,41,43,45,47	55,57,59,61,63
BRAM_Q12	23,25	-	-

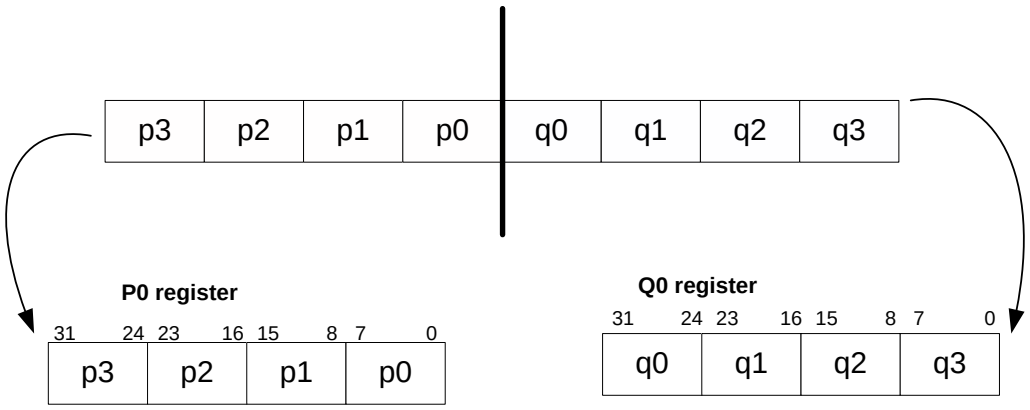
Παρακάτω αναλύονται καθένα από τα στάδια. Τα στάδια (PRE, FIR1, FIR2) αναφέρονται σαν ένα ως Execution stage.

Memory Read (MR)

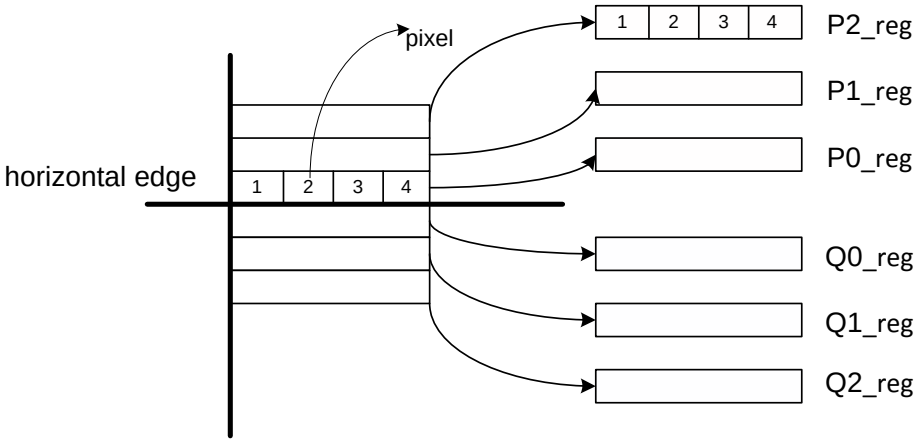


Εικόνα 5-9: Σχηματική αναπαράσταση του σταδίου MR

Στο στάδιο Memory Read (MR) γίνεται ανάγνωση των δεδομένων από τις BRAM στα οποία θα εφαρμοστεί το deblocking filter.Οι διευθύνσεις από τις οποίες θα διαβαστούν τα δεδομένα δίνονται από τη μονάδα ελέγχου (Control Unit) . Οι καταχωρητές P0_reg,P1_reg,P2_reg και Q0_reg,Q1_reg,Q2_reg είναι εύρους 32bit και περιέχουν τις τιμές στις οποίες θα εφαρμόσουμε το deblocking filter. Συγκεκριμένα στην περίπτωση των κάθετων ακμών χρησιμοποιούνται μόνο P0 και Q0 και οι τιμές τους είναι οι εξής:



Ενώ στην οριζόντια περίπτωση φορτώνουμε 4 στήλες στα P0,P1,P2 και Q0,Q1,Q2 ως εξής:



Η φόρτωση των καταχωρητών στην οριζόντια περίπτωση γίνεται σε διαδοχικούς κύκλους μίας και σε ένα κύκλο μπορούμε να διαβάσουμε το πολύ από 2 BRAM.Για να βεβαιωθούμε ότι οι άλλοι καταχωρητές διατηρούν την τιμή τους , διαθέτουν σήμα(hold) που τους επιτρέπει να διατηρούν την τιμή τους σε διαδοχικούς κύκλους. Αυτό το σήμα ενεργοποιείται κατάλληλα από τη μονάδα ελέγχου.Η επιλογή των εισόδων γίνεται από τους πολυπλέκτες M1 και M2 (βλέπε Εικόνα 5-10) επιλέγουν κατάλληλες εισόδους για τους ενδιάμεσους καταχωρητές. Συγκεκριμένα :

- **M1:** Πολυπλέκτες που επιλέγουν τιμές για τους καταχωρητές P[x]_reg. Οι συγκεκριμένοι πολυπλέκτες επιλέγουν από θα φορτωθούν τα δεδομένα για τα 2 μισά του καταχωρητή [15:0] και [31:16].Από τις παρακάτω δυνατές τιμές

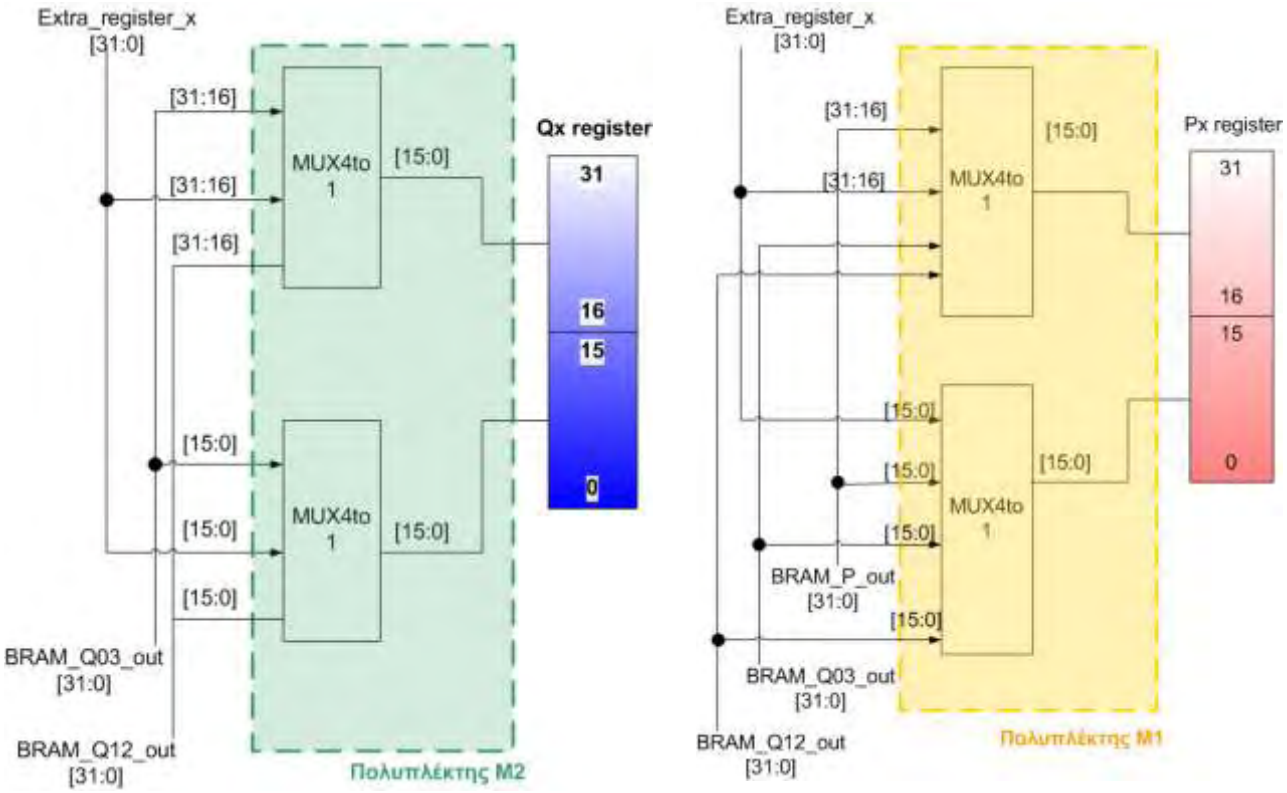
P[31:16]	P[15:0]
BRAM_P[31:16]	BRAM_P[15:0]
BRAM_Q03[31:16]	BRAM_Q03[15:0]
BRAM_Q12 [31:16]	BRAM_Q12 [15:0]

Extra_Registers[31:16]	Extra_Registers[15:0]

- **M2:** Πολυπλέκτες που επιλέγουν τιμές για τους καταχωρητές Q[x]_reg. Οι συγκεκριμένοι πολυπλέκτες επιλέγουν από θα φορτωθούν τα δεδομένα για τα 2 μισά του καταχωρητή [15:0] και [31:16].Από τις παρακάτω δυνατές τιμές

Q[31:16]	Q[15:0]
BRAM_Q03[31:16]	BRAM_Q03[15:0]
BRAM_Q12 [31:16]	BRAM_Q12 [15:0]
Extra_Registers[31:16]	Extra_Registers[15:0]

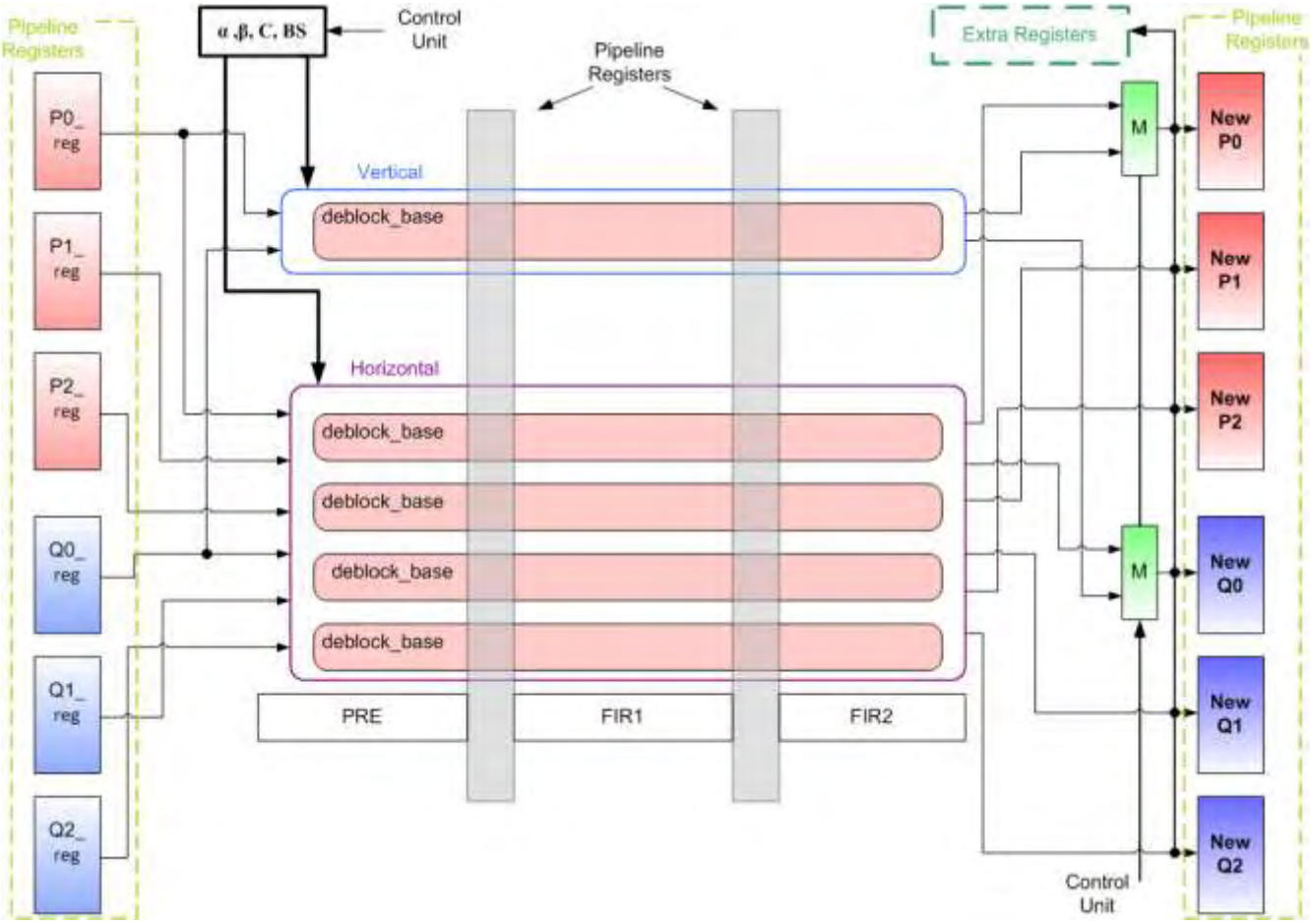
Η σχηματική αναπαράσταση των πολυπλεκτών φαίνεται παρακάτω



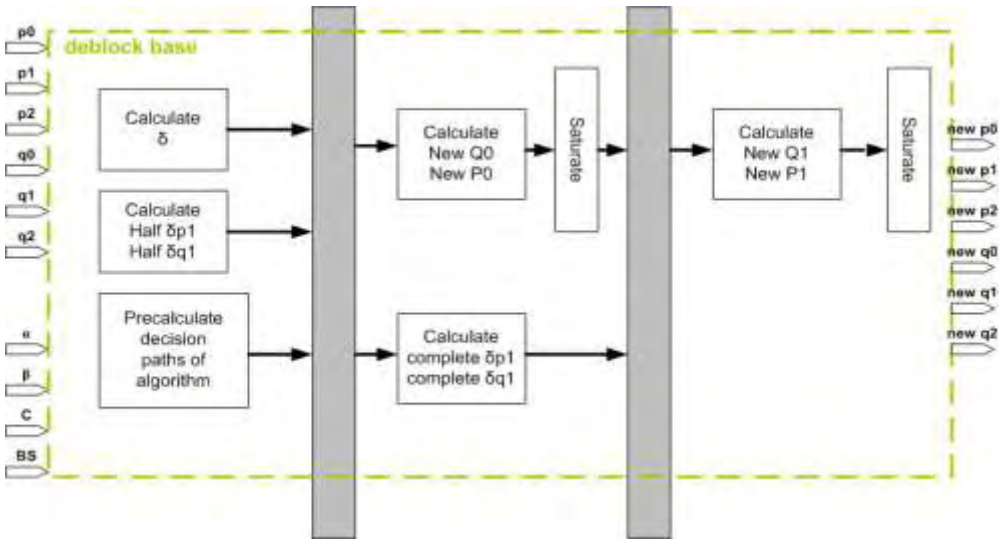
Εικόνα 5-10: Σχηματική αναπαράσταση πολυπλεκτών M1 και M2

Οι δομή και το περιεχόμενο των Extra_Registers θα αναλυθεί παρακάτω.

Execution stage (PRE, FIR1, FIR2)



Εικόνα 5-11: Αναλυτική σχηματική αναπαράσταση του Execution Stage



Εικόνα 5-12: Σχηματική αναπαράσταση του deblock_base

Στο στάδιο αυτό έχουμε την υλοποίηση του deblocking filter.Στην Εικόνα 5-11 βλέπουμε το Execution Stage χωρίζεται σε 3. Επίσης έχουμε ξεχωριστό μονοπάτι για την κάθετη περίπτωση όπου και οι είσοδοι του είναι μόνο οι καταχωρητές P0_reg και Q0_reg. Ενώ στην οριζόντια περίπτωση οι είσοδοι είναι όλοι οι ενδιάμεσοι καταχωρητές (P0_reg,P1_reg,P2_reg,Q0_reg,Q1_reg,Q2_reg).

Η κυρίως υλοποίηση του αλγορίθμου deblocking filter γίνεται από το module deblock_base (Εικόνα 5-12) έτσι στην κάθετη περίπτωση (Vertical) έχουμε χρήση ενός deblock_base επειδή εκτελούμε τον αλγόριθμο μία τη φορά. Στην οριζόντια περίπτωση επειδή φορτώνουμε 4 ομάδες δεδομένων (4 στήλες) μπορούμε να εκτελέσουμε τον αλγόριθμο 4 φορές, άρα κάνουμε instantiate[8] το module deblock_base 4 φορές. Και στις 2 περιπτώσεις έχουμε εισαγωγή παραμέτρων ***α, β, C, BS*** των οποίων οι κατάλληλες τιμές επιλέγονται από εσωτερικούς καταχωρητές ολόκληρου του κυκλώματος με σήμα από τη μονάδα ελέγχου (Control Unit).

Κατά την ολοκλήρωση του αλγορίθμου οι καινούργιες τιμές αποθηκεύονται σε καταχωρητές για να αποθηκευτούν σε επόμενο στάδιο στις BRAM. Οι καταχωρητές αυτοί είναι παρόμοιοι με αυτούς στο στάδιο Memory Read. Ανάλογα με τη θέση των δεδομένων μερικά αποθηκεύονται επιπλέον και στους Extra _Registers για να μπορέσουν να επαναχρησιμοποιηθούν σε επόμενη φάση όπως θα δούμε σε επόμενη παράγραφο. Ανάλογα με την περίπτωση οι πολυπλέκτες M επιλέγουν την είσοδο των καταχωρητών, δηλαδή είτε από το horizontal vertical τμήμα του Execution Stage.Η επιλογή καθορίζεται με σήμα από τη Μονάδα Ελέγχου.

Deblock base

Το Deblock base αποτελείται από 3 στάδια. Παίρνει σαν είσοδο τις τιμές των pixel γύρω από το όριο (P0,P1,P2,Q0,Q1,Q2) και τις παραμέτρους (α, β, C, BS) και σαν έφοδο παράγει τα newP0, newP1, newQ0, newQ1 και απλώς αντιγράφει στην έξοδο τις τιμές P2 και Q2 οι οποίες δεν επηρεάζονται από τον αλγόριθμο .Παρακάτω περιγράφουμε αναλυτικά τα στάδια, δείχνοντας ποιοι υπολογισμοί πραγματοποιούνται και ποιες τιμές μεταφέρονται στο επόμενο. Στο Παράρτημα Α παρουσιάζεται ο διαμοιρασμός σε στάδια του διαγράμματος της Εικόνας 4-9

[8]Instantiate: Χρήση του module

Σταδιο 1 – Precalculation (PRE)

Υπολογίζονται:

$$\delta = Clip(-C, C, ((q0 - p0) * 3 + (p1 - q1) + 4) >> 3)$$
$$\delta p1' = (- p1) * 3 + p2 + 4)$$
$$\delta q1' = (q1)* 3 - q2) + 4)$$

Καθορίζονται τα μονοπάτια επιλογής

- **BS ==2**

```
if (|q2 - p0| < β) && (|p0 - q0| < (α >> 2) + 2))
      q_way = TRUE
else
      q_way = FALSE


 if (|p2 - p0| < β) && (|p0 - q0| < (α >> 2) + 2))
    p_way = TRUE
else
    p_way = FALSE
```

- **BS ==1**

```
if (|q2 - p0| < β)
   q_way = TRUE
else
   q_way = FALSE

if (|p2 - p0|)
    p_way = TRUE
else
    p_way = FALSE
```

- **BS ==0**

```
q_way = FALSE
p_way = FALSE
```

Στο επόμενο στάδιο διοχετεύονται οι τιμές των

1. p0,p1,p2
2. q0,q1,q2
3. BS
4. p_way, q_way
5. δ
6. δp1', δq1'

Σταδιο 2 – Filtering 1 (FIR1)

Ανάλογα με το BS υπολογίζονται τα εξής

- **BS ==2**

```
if (q_way)
  newQ0 = (q1 + 2 * q0 + p0 + 2) >> 2;
else
  newQ0 = (2 * q1 + q0 + p0 + 2) >> 2;

if (p_way)
  newP0 = (p1 + 2 * p0 + q0 + 2) >> 2;
else
  newP0 = (2 * p1 + p0 + q0 + 2) >> 2;
```

- **BS ==1**

newP0 = p0 + δ;
newQ0 = q0 – δ;
δp1 = Clip(-C, C, ((**newP0** * 3 – **newQ0**) + δp1') >> 3)
δq1 = Clip(-C, C, ((–**newQ0**) * 3 + (**newP0**) + δq1') >> 3)

- **BS ==0**

newP0 = p0;
newQ0 = q0;

Στο επόμενο στάδιο διοχετεύονται οι τιμές των

1. p0,p1,p2
2. q0,q1,q2
3. newQ0,newP0
4. BS
5. p_way, q_way
6. δp1, δq1

Σταδιο 3 – Filtering 2 (FIR2)

Ανάλογα με το BS υπολογίζονται τα εξής

- **BS ==2**

```
if (q_way)
  newQ1 = (2 * q1 + q0 + p0 + 2) >> 2;
else
  newQ1 = q1;

if (p_way)
  newP1 = (2 * p1 + p0 + q0 + 2) >> 2;
else
  newP1 = p1;
```

- **BS ==1**

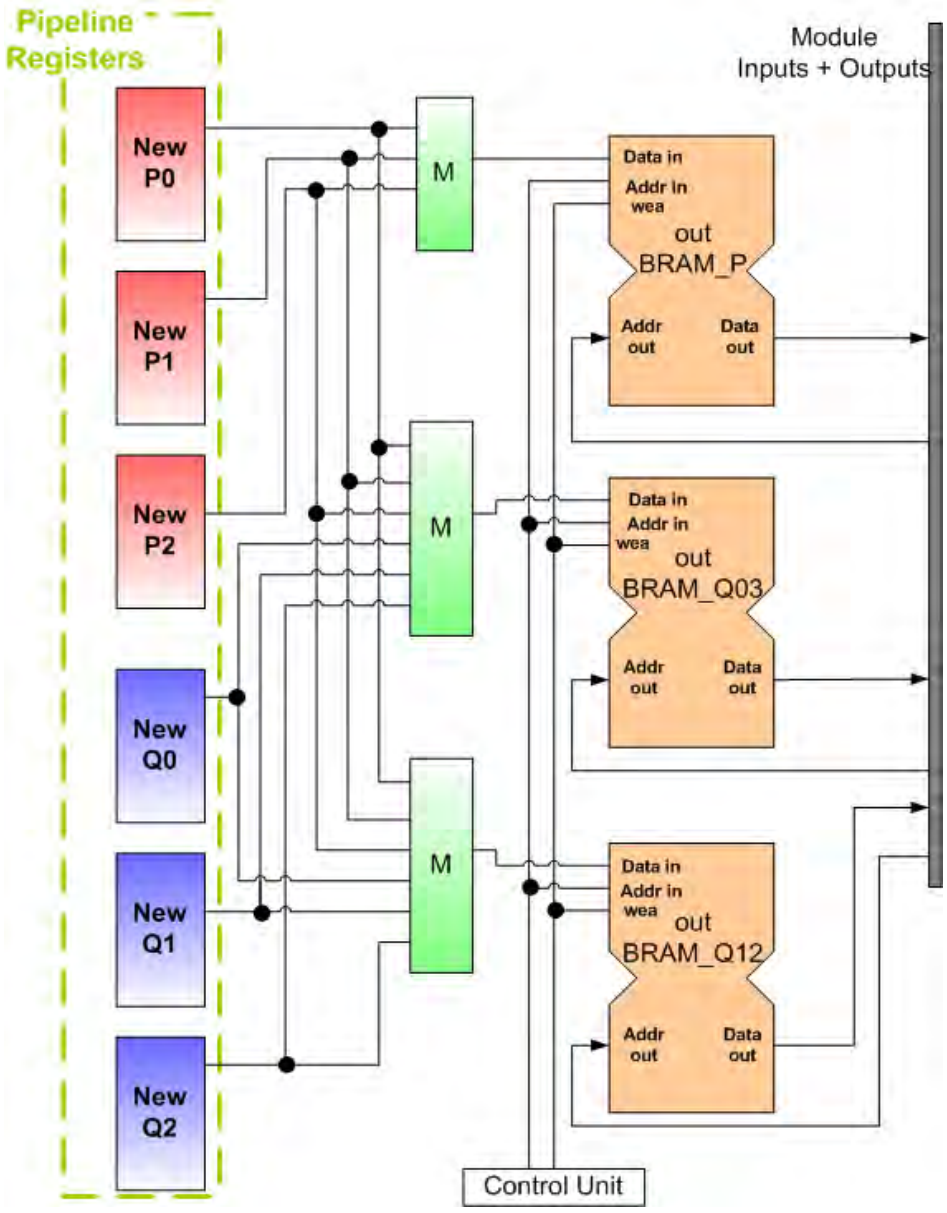
```
if (q_way)
  newQ1 = q1 - δq1;
else
  newQ1 = q1;

if (p_way)
  newP1 = p1 + δp1;
else
  newP1 = p1;
```

- **BS ==0**

newP1 = p1;
newQ1 = q1;

Memory write (MW)



Εικόνα 5-13: Σχηματική αναπαράσταση του MW σταδίου

Στο στάδιο αυτό έχουμε την αποθήκευση των νέων υπολογισμένων pixel στις BRAM εξόδου. Στην περίπτωση που εφαρμόζουμε τον αλγόριθμο στις ακμές του κάθετου άξονα τότε η εγγραφή γίνεται σε ένα κύκλο και οι πολυπλέκτες M στις BRAM παίρνουν σαν είσοδο τα περιεχόμενα των καταχωρητών P0 και Q0. Στην περίπτωση που εφαρμόζουμε τον αλγόριθμο σε ακμές του οριζόντιου άξονα τότε η εγγραφή γίνεται σε 3 κύκλους. Επειδή μπορούμε να κάνουμε το πολύ 2 εγγραφές σε κάθε κύκλο ενεργοποιούνται από τη Μονάδα Ελέγχου σήματα που ενεργοποιούνε την λειτουργία διατήρησης τιμής των καταχωρητών. Επιπλέον οι καταχωρητές επιλέγουν τις τιμές που είναι προς εγγραφή επιλέγοντας διαδοχικά τους P0,P1,P2

και Q0,Q1,Q2 τοποθετώντας τα στις κατάλληλες BRAM. Οι διευθύνσεις και οι τιμές για τις επιλογές των πολυπλεκτών δίνονται από την Μονάδα Ελέγχου.

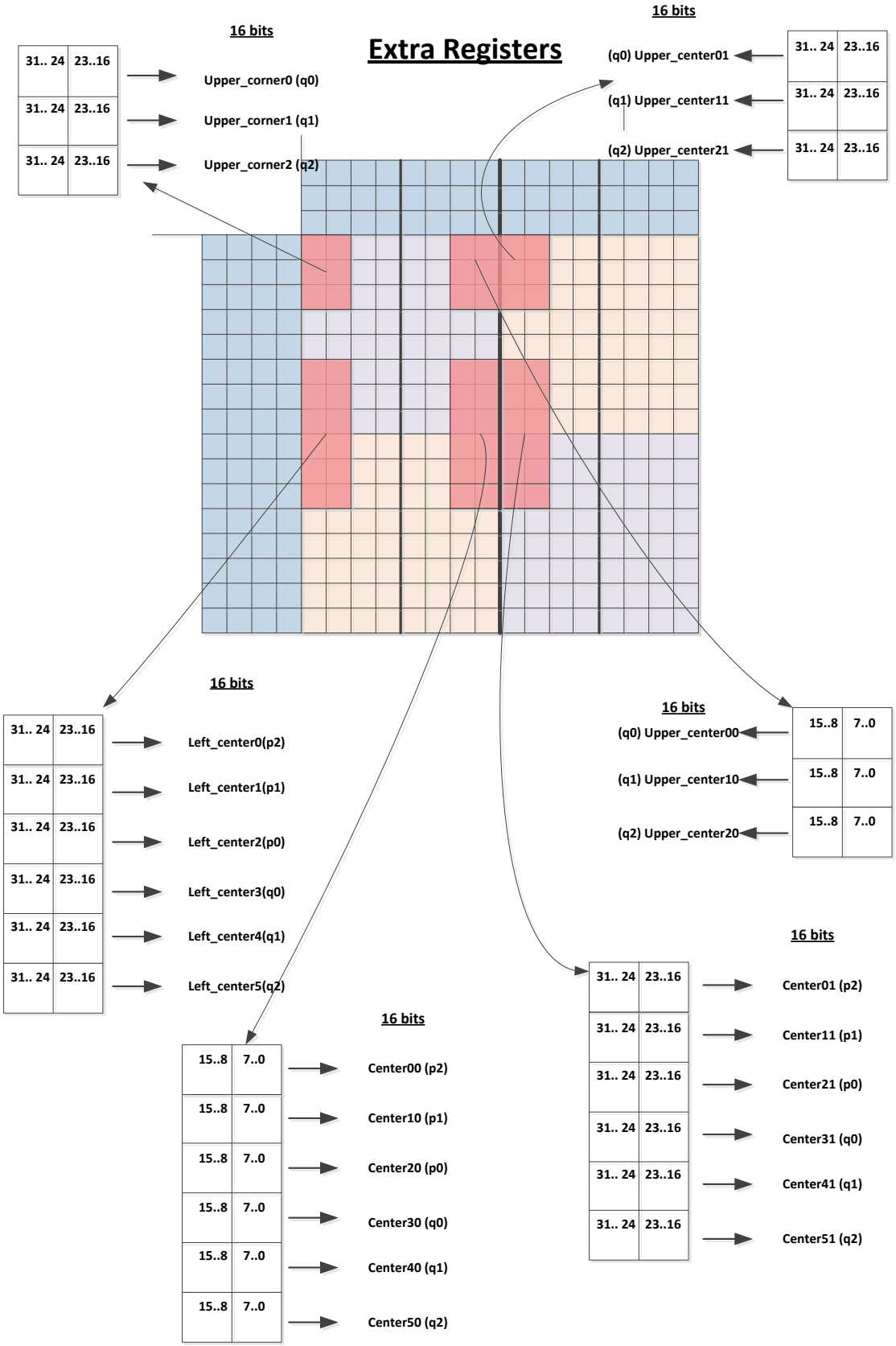
Για να μπορέσουμε να διαβάσουμε τα περιεχόμενα των εσωτερικών BRAM θα πρέπει να δώσουμε διαδοχικά τις διευθύνσεις των τιμών που θέλουμε να διαβάσουμε(Βλέπε «Δεδομένα Εξόδου» Παρ. 5.1).

Extra Registers

Οι Extra Registers είναι καταχωρητές οι οποίοι διατηρούν τις τιμές pixel στα οποία έχει τρέξει ο αλγόριθμος του deblocking μια φορά και πρέπει να ξαναχρησιμοποιηθούν.

Επειδή ο αλγόριθμος ξεκινάει να εφαρμόζει τον αλγόριθμο πρώτα στις ακμές του κάθετου άξονα, δηλαδή με τη σειρά στις ακμές V00 V10 V01 V11 , και μετά στις ακμές του οριζόντιου άξονα, δηλαδή με τη σειρά στις ακμές H00 H01 H10 H11, κάποια pixel προσπελαύνονται $2^{η}$ φορά έχοντας αλλάξει ήδη τιμές (Βλέπε εικόνα 5-14) . Για να μπορέσουμε να υλοποιήσουμε σωστά τον αλγόριθμο και να έχουμε συνέπεια με τα αποτελέσματα του κώδικα αναφοράς πρέπει να κρατάμε τιμές και να τις συνδέσουμε με τις εισόδους των καταχωρητών που χρησιμοποιούνται για τον υπολογισμό. Άρα μαζί με την αποθήκευση στους ενδιάμεσους καταχωρητές πριν από την αποθήκευση στις BRAM αποθηκεύονται στους extra_registers οι τιμές που απαιτούνται να ξαναχρησιμοποιηθούν. Για να διατηρήσουν τις τιμές τους οι καταχωρητές ενεργοποιείται κατάλληλο σήμα από την Μονάδα Ελέγχου.

Στο σχήμα παρακάτω παρουσιάζεται η περίπτωση του luma macroblock, με κόκκινο τονίζονται οι τιμές που πρέπει να επαναχρησιμοποιηθούν και σε ποιους καταχωρητές αποθηκεύονται. Για την περίπτωση των chroma block ισχύει η υποπερίπτωση του παρακάτω σχήματος και χρειάζονται μόνο οι 3 καταχωρητές στην πάνω αριστερή γωνία

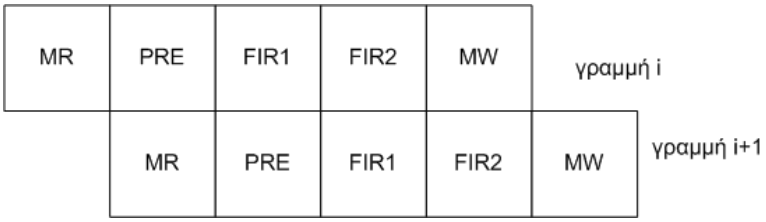


Εικόνα 5-13: Σημεία ενός macroblock που επαναχρησιμοποιούνται

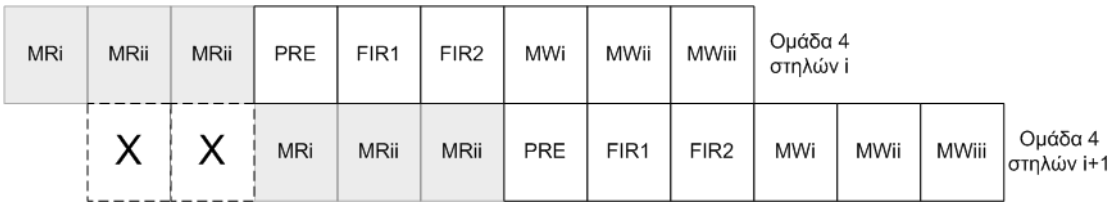
5.3 Διαδοχικές εκτελέσεις

Τα παρακάτω παραδείγματα αφορούν είτε τον αλγόριθμο για το luma είτε για το chroma.

Παράδειγμα εκτέλεσης 2 διαδοχικών εκτελέσεων του αλγορίθμου στα όρια μιας κάθετης ακμής:



Παράδειγμα εκτέλεσης 2 διαδοχικών εκτελέσεων του αλγορίθμου στα όρια μιας οριζόντια ακμής:



5.4 Ανάλυση FSM – Μονάδα ελέγχου

Η Μονάδα Ελέγχου είναι αυτή που ενεργοποιεί όλα τα σήματα ελέγχου της μονάδας. Όταν ενεργοποιηθεί το σήμα εισόδου **ready_in** του κυκλώματος τότε ξεκινάει την εκτέλεση η μηχανή πεπερασμένων καταστάσεων (FSM) από την οποία ενεργοποιούνται όλα τα σήματα.

Η μηχανή πεπερασμένων καταστάσεων αποτελείται από 118 καταστάσεις. Στις καταστάσεις αυτές γίνεται ενεργοποίηση των κατάλληλων σημάτων για εφαρμοστεί το deblocking filter σε ένα luma component και 2 chroma components.

Παρακάτω θα παρουσιάσουμε τα σήματα που ενεργοποιούνται σε κάθε φάση κατά την εκτέλεση του αλγορίθμου για τα στοιχεία μιας κάθετης ακμής και τα στοιχεία μιας οριζόντιας ακμής .Αναδρομικά αυτοί οι πίνακες ισχύουν για όλες τις ακμές του ίδιου τύπου.

Vertical Edge (Κάθετη ακμή)

Phase – Φάση	Περιγραφή	Σήματα
0	• Δίνονται οι κατάλληλες διευθύνσεις στις BRAM από τις οποίες θα διαβαστούν τα περιεχόμενα για την εκτέλεση του deblocking filter	• Διευθύνσεις στις εισόδους των BRAM
1 - MW	• Ανάγνωση τιμών από τις BRAM • Αποθήκευση στους κατάλληλους ενδιάμεσους καταχωρητές (P0,Q0)	• Σήματα ελέγχου πολυπλεκτών στις εισόδους των P0,Q0
2 –PRE	• Επιλογή κατάλληλων τιμών παραμέτρων α, β, C, BS από τους εσωτερικούς καταχωρητές που περιέχονται στο κύκλωμα • Εκκίνηση αλγορίθμου deblocking ($1^{ο}$ στάδιο)	• Σήματα πολυπλεκτών που επιλέγουν τις τιμές των παραμέτρων
3 – FIR1	• Αλγόριθμος deblocking ($2^{ο}$ στάδιο)	• -
4 – FIR2	• Αλγόριθμος deblocking ($3^{ο}$ στάδιο) • Επιλογή κατάλληλων εισόδων για τους ενδιάμεσους καταχωρητές newP0 και newQ0	• Σήματα επιλογής εισόδων newP0 και newQ0
5 - MW	• Δίνονται οι διευθύνσεις όπου θα αποθηκευτούν οι νέες τιμές • Ανάλογα με τα στοιχεία οι κατάλληλοι Extra Registers διατηρούν την τιμή τους	• Ενεργοποίηση σήματος διατήρησης στους απαραίτητους Extra Registers • Ενεργοποίηση εγγραφής στις κατάλληλες BRAM εξόδου • Διευθύνσεις προορισμού δεδομένων στις BRAM

Horizontal Edge (Οριζόντια ακμή)

Phase – Φάση	Περιγραφή	Σήματα
0	• Δίνονται οι κατάλληλες διευθύνσεις στις BRAM από τις οποίες θα διαβαστούν τα περιεχόμενα P0,Q0 για την εκτέλεση του deblocking filter	• Διευθύνσεις των P0,Q0 στις εισόδους των BRAM
1 - MWi	• Διευθύνσεις τιμών P1,Q1 • Ανάγνωση τιμών P0,Q0 από τις BRAM • Αποθήκευση στους κατάλληλους ενδιάμεσους καταχωρητές (P0,Q0)	• Διευθύνσεις των P1,Q1 στις εισόδους των BRAM • Σήματα ελέγχου πολυπλεκτών στις εισόδους των P0,Q0
2 - MWii	• Διευθύνσεις τιμών P2,Q2 • Ανάγνωση τιμών P1,Q1 από τις BRAM • Αποθήκευση στους κατάλληλους ενδιάμεσους καταχωρητές (P1,Q1) • Διατήρηση τιμών P0,Q0	• Διευθύνσεις των P2,Q2 στις εισόδους των BRAM • Σήματα ελέγχου πολυπλεκτών στις εισόδους των P1,Q1 • Ενεργοποίηση σήματος διατήρησης τιμών για P0,Q0
3 - MWiii	• Ανάγνωση τιμών P2,Q2 από τις BRAM • Αποθήκευση στους κατάλληλους ενδιάμεσους καταχωρητές (P2,Q2) • Διατήρηση τιμών P1,Q1	• Σήματα ελέγχου πολυπλεκτών στις εισόδους των P2,Q2 • Ενεργοποίηση σήματος διατήρησης τιμών για P1,Q1
4 –PRE	• Επιλογή κατάλληλων τιμών παραμέτρων α, β, C, BS από τους εσωτερικούς καταχωρητές που περιέχονται στο κύκλωμα • Εκκίνηση αλγορίθμου deblocking ($1^{ο}$ στάδιο)	• Σήματα πολυπλεκτών που επιλέγουν τις τιμές των παραμέτρων
5 – FIR1	• Αλγόριθμος deblocking ($2^{ο}$ στάδιο)	• Απενεργοποίηση σημάτων διατήρησης τιμών για P1,Q1, P0,Q0
6 – FIR2	• Αλγόριθμος deblocking ($3^{ο}$ στάδιο) • Επιλογή κατάλληλων εισόδων για τους ενδιάμεσους καταχωρητές newP0 και newQ0	• Σήματα επιλογής εισόδων newP0 και newQ0 • Απενεργοποίηση οποιονδήποτε σημάτων που διατηρούν τις τιμές στους newP1 ,newQ1 , newP2, newQ2
7 - MWi	• Δίνονται οι διευθύνσεις όπου θα αποθηκευτούν οι νέες τιμές newP0 ,newQ0 • Οι καταχωρητές newP1 ,newQ1 , newP2, newQ2 πρέπει να διατηρούν τις τιμές τους • Ενεργοποίηση εγγραφής στις κατάλληλες BRAM	• Ενεργοποίηση σημάτων που διατηρούν τις τιμές στους newP1 ,newQ1 , newP2, newQ2 • Ενεργοποίηση εγγραφής στις κατάλληλες BRAM εξόδου • Διευθύνσεις προορισμού δεδομένων στις BRAM
8 - MWii	• Δίνονται οι διευθύνσεις όπου θα αποθηκευτούν οι νέες τιμές newP1 ,newQ1 • Ενεργοποίηση εγγραφής στις κατάλληλες BRAM	• Ενεργοποίηση εγγραφής στις κατάλληλες BRAM εξόδου • Διευθύνσεις προορισμού δεδομένων στις BRAM
9 - MWiii	• Δίνονται οι διευθύνσεις όπου θα αποθηκευτούν οι νέες τιμές newP2,newQ2 • Ενεργοποίηση εγγραφής στις κατάλληλες BRAM	• Ενεργοποίηση εγγραφής στις κατάλληλες BRAM εξόδου • Διευθύνσεις προορισμού δεδομένων στις BRAM

Στην παράγραφο 8 υπάρχει συγκεντρωτικός πίνακας με όλα τα σήματα από τη Μονάδα Ελέγχου

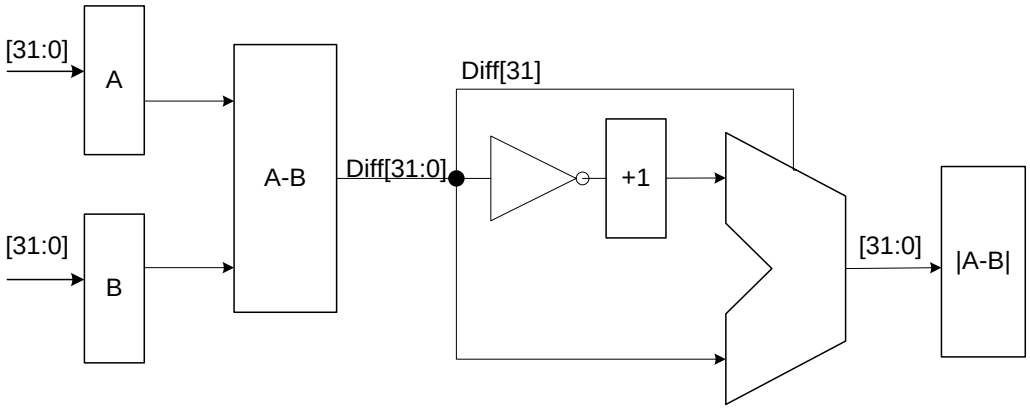
5.5 Βελτιστοποιήσεις Virtex 5

Σύμφωνα με μελέτες[11] όπως και από tutorial της Xilinx[12] έγιναν χρήση των παρακάτω βελτιστοποιήσεων για την καλύτερη απόδοση του κυκλώματος

Υπολογισμός απόλυτης διαφοράς

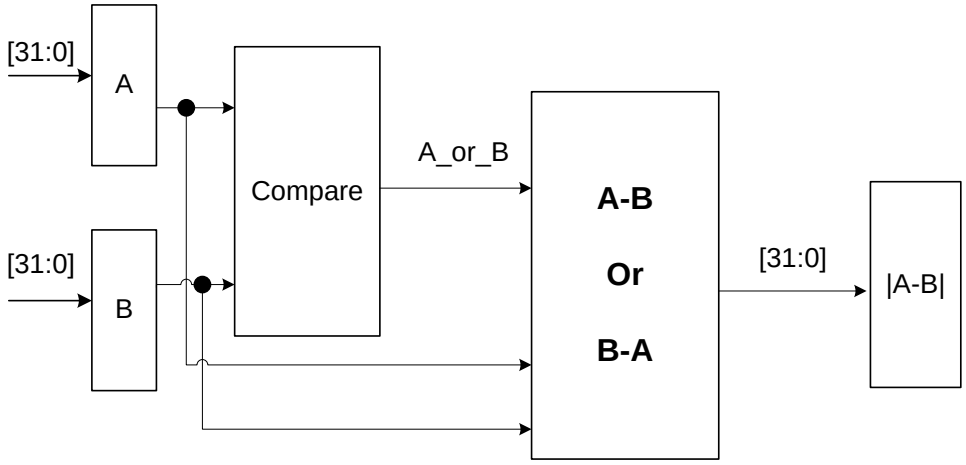
Στην μελέτη αναφέρεται ότι η αυτόματη σύνθεση του κυκλώματος υπολογισμού απόλυτης διαφοράς από το εργαλείο σύνθεσης δεν είναι βέλτιστο από άποψη συχνότητας λειτουργίας και καταλαμβανόμενων πόρων.

Το κύκλωμα που παράγετε είναι:



Το οποίο κύκλωμα υπολογίζει τη διαφορά A-B -> Diff. Έπειτα υπολογίζει το συμπλήρωμα ως προς 2 του A-B και με βάσει το πρόσημο της προηγούμενης πράξης επιλέγει το κατάλληλο νούμερο μέσω ενός πολυπλέκτη, που έχει σα σήμα ελέγχου το σημαντικότερο bit της πράξης της διαφοράς.

Το προτεινόμενο κύκλωμα είναι



Το οποίο κύκλωμα κάνει τη σύγκριση μεταξύ του του A και B και με βάσει του αποτελέσματος επιλέγει να κάνει είτε την πράξη A-B είτε B-A.

Επομένως ο κώδικας υλοποίησης της παραπάνω βελτιστοποίησης στη υλοποίηση του κυκλώματος είναι ο εξής

```
function [10:0] abs_diff;
      input [10:0] a;
      input [10:0] b;
      reg temp;
      begin

            if(a>b)
                  temp = 1;
            else
                  temp = 0;

            if(temp ==1)
                  abs_diff = a + ~b + 1;
            else
                  abs_diff = b + ~a + 1;

      end
endfunction
```

Επιπλέον βελτιστοποίησεις

Για να πετύχουμε την καλύτερη απόδοση του κυκλώματος από τις προτάσεις των σεμιναρίων της Xilinx κάναμε τις παρακάτω βελτιστοποιήσεις

1. Αφαίρεση reset σήματος όπου στην πραγματικότητα δεν χρειαζόταν κάποιος μηδενισμός. Διατηρήθηκε λοιπόν μόνο στην FSM (πεπερασμένη μηχανή καταστάσεων) για να φέρνει το κύκλωμα στην αρχική κατάσταση
2. Το reset το κάναμε σύγχρονο, το οποίο σημαίνει ότι το λαμβάνουμε υπ' όψιν μόνο όταν έχει αλλαγή στο παλμό του ρολογιού και συγκεκριμένα μόνο στο θετικό παλμό
3. Το reset το κάναμε να επαναφέρει το κύκλωμα όταν είναι active high, δηλαδή όταν έχει την τιμή 1

5.6 Χαρακτηριστικά αρχιτεκτονικής στη Virtex 5

Για τις μετρήσεις και τις υλοποιήσεις του κυκλώματος στη Virtex 5 χρησιμοποιήσαμε το εργαλείο της **Xilinx** το **ISE** έκδοση **12.4**.

Έχοντας κάνει σύνθεση (synthesis) και υλοποίηση(translate,map και place-and-route) του κυκλώματος. Πήραμε τα ακόλουθα αποτελέσματα για το

Device Utilization Summary – Πίνακας Χρήσης Πόρων			
Slice Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	2,634	44,800	5%
Number used as Flip Flops	2,449		
Number used as Latches	184		
Number used as Latch-thrus	1		
Number of Slice LUTs	4,173	44,800	9%
Number used as logic	4,097	44,800	9%
Number using O6 output only	3,500		
Number using O5 output only	74		
Number using O5 and O6	523		
Number used as Memory	66	13,120	1%
Number used as Shift Register	66		
Number using O6 output only	66		
Number used as exclusive route-thru	10		
Number of route-thrus	103		
Number using O6 output only	64		
Number using O5 output only	39		
Number of occupied Slices	1,430	11,200	12%
Number of LUT Flip Flop pairs used	4,866		
Number with an unused Flip Flop	2,232	4,866	45%
Number with an unused LUT	693	4,866	14%
Number of fully used LUT-FF pairs	1,941	4,866	39%
Number of unique control sets	71		
Number of slice register sites lost to control set restrictions	117	44,800	1%
Number of bonded IOBs	417	640	65%
IOB Latches	1		
Number of BlockRAM/FIFO	5	148	3%
Number using BlockRAM only	5		
Number of 18k BlockRAM used	6		
Total Memory used (KB)	108	5,328	2%
Number of BUFG/BUFGCTRLs	1	32	3%
Number used as BUFGs	1		
Average Fanout of Non-Clock Nets	3.72		

Παλμός Ρολογιού

Ελάχιστη περίοδος	Μέγιστη Συχνότητα
6,230 ns	160.514MHz

Στην περίπτωση που δεν εφαρμόσουμε τη βελτιστοποίηση τα συγκριτικά αποτελέσματα που παίρνουμε είναι

Τιμές	Non-Optimized	Optimized	Improvement (%)	Available
Number of Slice Registers	2651	2634	0,64	44800
Number of Slice LUTs	4306	4173	3,09	44800
Number of occupied Slices	1659	1430	13,80	11200
Number of LUT Flip Flop pairs used	5065	4866	3,93	-
Max Frequency(MHz)	151	160,514	6,23	-

Συγκεντρωτικά αποτελέσματα

Άρα αφού χρειαζόμαστε 118 κύκλους για την ολοκλήρωση του υπολογισμού 1 luma component και 2 chroma components παρουσιάζουμε τους εκτιμώμενος χρόνους εκτέλεσης μόνο[9] για τον υπολογισμό ενός frame.

Type	Resolution	# MB	Χρόνος εκτέλεσης	Frames per sec
VGA	640 x 480	1200	7,476 μs	133761,36
720p	1280 x 720	3600	22,428 μs	44587,1
1080p	1920 x 1080	8100	50,463 μs	19816,5

[9] Χωρίς το χρόνο μεταφοράς δεδομένων

6. ΕΠΙΛΟΓΟΣ

6.1 Συμπεράσματα

Μέσα από την παρούσα εργασία γνωρίσαμε την αρχιτεκτονική μιας συσκευής επαναδιατασσόμενης λογικής και τις δυνατότητές της. Αναλύσαμε τον αλγόριθμο του deblocking filter για να μπορέσουμε να τον εκφράσουμε με γλώσσα περιγραφής υλικού. Η προσπάθεια μας εντάθηκε στη υλοποίηση του αλγορίθμου με το βέλτιστο τρόπο χωρίζοντας σε επιμέρους στάδια. Έτσι λοιπόν ανακαλύψαμε δυνατότητες που μας προσέφερε η συσκευή με την οποία δουλέψαμε και κυρίως εκμεταλλευτήκαμε τον παραλληλισμό που μας προσφέρει εκ φύσεως το υλικό. Τα οφέλη μιας τέτοιας εργασίας είναι η πιθανή δημιουργία ενός επιταχυντή που θα μπορούσε να βοηθήσει στην επιτάχυνση του αλγορίθμου αποκωδικοποίησης σε ενσωματωμένες συσκευές.

6.2 Προοπτικές Επέκτασης/Εξέλιξης

Οι πιθανές προοπτικές της παρούσας διπλωματικής είναι η υλοποίηση οδηγού έτσι ώστε να μπορέσει να χρησιμοποιηθεί σαν επιταχυντής κατά την εκτέλεση του αλγορίθμου αποκωδικοποίησης είτε στον embedded processor της FPGA ή ακόμα και μέσω της χρήσης της PCI-express σε κάποιον x86 επεξεργαστή. Και με την ολοκλήρωση αυτού να εξαχθούν μετρικές απόδοσης της κάθε υλοποίησης.

Επιπλέον μπορούν να υλοποιηθούν σε επίπεδο υλικού επιπλέον κομμάτια του αλγορίθμου αποκωδικοποίησης όπως είναι το inverse transform και μαζί με το motion compensation (το οποίο υπάρχει ήδη υλοποιημένο) να δημιουργήσουν ένα πλήρες κύκλωμα που υλοποιεί τον αλγόριθμο αποκωδικοποίησης του AVS

Τέλος είναι δυνατή η επισκόπηση κατανάλωσης ενέργειας του κυκλώματος συγκρίνοντας το με άλλες υλοποιήσεις και διερεύνηση πιθανών ωφελειών.

7. ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] *H.264 and MPEG-4 Video Compression*, Iain E. G. Richardson, Wiley

[2] *Image and Video Compression for Multimedia Engineering: Fundamentals - Algorithms - and Standards – 2nd Edition*, Yun Q. Shi , Huifang Sun , CRC Press

[3] *AVS Standard Part2 Video*

[4] *Video Compression Wikipedia article*: http://en.wikipedia.org/wiki/Video_compression

[5] *Image and Video Compression Standards: Algorithms and Architectures Second Edition*, Vasudev Bhaskaran ,Konstantinos Konstantinides , Kluwer Academic Publishers

[6] *AVS Wikipedia article*: http://en.wikipedia.org/wiki/Audio_Video_Standard

[7] *Υλοποίηση του Αποκωδικοποιητή video AVS σε επαναδιατασσόμενη λογική,* Κωνσταντίνος Κρομμύδας Πτυχιακή 2010 Πανεπιστήμιο Θεσσαλίας

[8] *Deblocking filter (video) Wikipedia article*: http://en.wikipedia.org/wiki/Deblocking_filter_(video)

[9] ***A**udio **V**ideo Coding **S**tandard (AVS) – Video Techniques, Presentation* by Prof. YU, Lu Oct 20, 2004

[10] *FPGA Wikipedia article*: http://en.wikipedia.org/wiki/FPGA

[11] *Efficient absolute difference circuits in Virtex-5 FPGAs*, Perri, S.; Zicari, P.; orsonello, P.; Dept. of Electornics, Comput. Sci. & Syst., Univ. of Calabria, Calabria, Italy

[12] *Xilinx training videos:* http://www.xilinx.com/training/free-video-courses.htm

[13] *IP Processor Block RAM (BRAM) Block (v1.00a)* http://www.xilinx.com/support/documentation/ip_documentation/bram_block.pdf

[14] *Virtex-5 FPGA User Guide UG190 (v5.3) May 17, 2010* http://www.xilinx.com/support/documentation/user_guides/ug190.pdf

[15] *A Five-Stage Pipeline, 204 Cycles/MB, Single-Port SRAM-Based Deblocking Filter for H.264/AVC,* Ke Xu; Chiu-Sing Choy; Chinese Univ. of Hong Kong, Hong Kong

8. Πίνακας σημάτων Μονάδας Ελέγχου

Σήμα	Περιγραφή
address_BRAM_P	Σήμα που μας δίνει τις διευθύνσεις για να διαβάσουμε από τη BRAM_P εισόδου
address_BRAM_Q03	Σήμα που μας δίνει τις διευθύνσεις για να διαβάσουμε από τη BRAM_Q03 εισόδου
address_BRAM_Q12	Σήμα που μας δίνει τις διευθύνσεις για να διαβάσουμε από τη BRAM_ Q12 εισόδου
out_BRAM_P_w_addr	Σήμα που μας δίνει τις διευθύνσεις για να γράψουμε στη BRAM_P εξόδου
out_BRAM_P_wea	Σήμα που ενεργοποιεί την εγγραφή στη BRAM_P εξόδου
out_BRAM_Q03_w_addr	Σήμα που μας δίνει τις διευθύνσεις για να γράψουμε στη BRAM_ Q03 εξόδου
out_BRAM_Q03_wea	Σήμα που ενεργοποιεί την εγγραφή στη BRAM_ Q03 εξόδου
out_BRAM_Q12_w_addr	Σήμα που μας δίνει τις διευθύνσεις για να γράψουμε στη BRAM_ Q12 εξόδου
out_BRAM_Q12_wea	Σήμα που ενεργοποιεί την εγγραφή στη BRAM_ Q12 εξόδου
a_sel	Σήμα που ελέγχει τον πολυπλέκτη ο οποίος επιλέγει τις κατάλληλες τιμές της παραμέτρου α
b_sel	Σήμα που ελέγχει τον πολυπλέκτη ο οποίος επιλέγει τις κατάλληλες τιμές της παραμέτρου β
c_sel	Σήμα που ελέγχει τον πολυπλέκτη ο οποίος επιλέγει τις κατάλληλες τιμές της παραμέτρου C
BS_sel	Σήμα που ελέγχει τον πολυπλέκτη ο οποίος επιλέγει τις κατάλληλες τιμές της παραμέτρου BS
finished	Σήμα που ενεργοποιείται όταν έχει τελειώσει το deblocking για όλα τα δεδομένα εισόδου (1 macroblock και 2 chroma block)
UCor0_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του UCor0 (Extra Register)
UCor1_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του UCor1 (Extra Register)
UCor2_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του UCor2 (Extra Register)
LCen0_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του LCen0 (Extra Register)
LCen1_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του LCen1 (Extra Register)
LCen2_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του LCen2 (Extra Register)
LCen3_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής

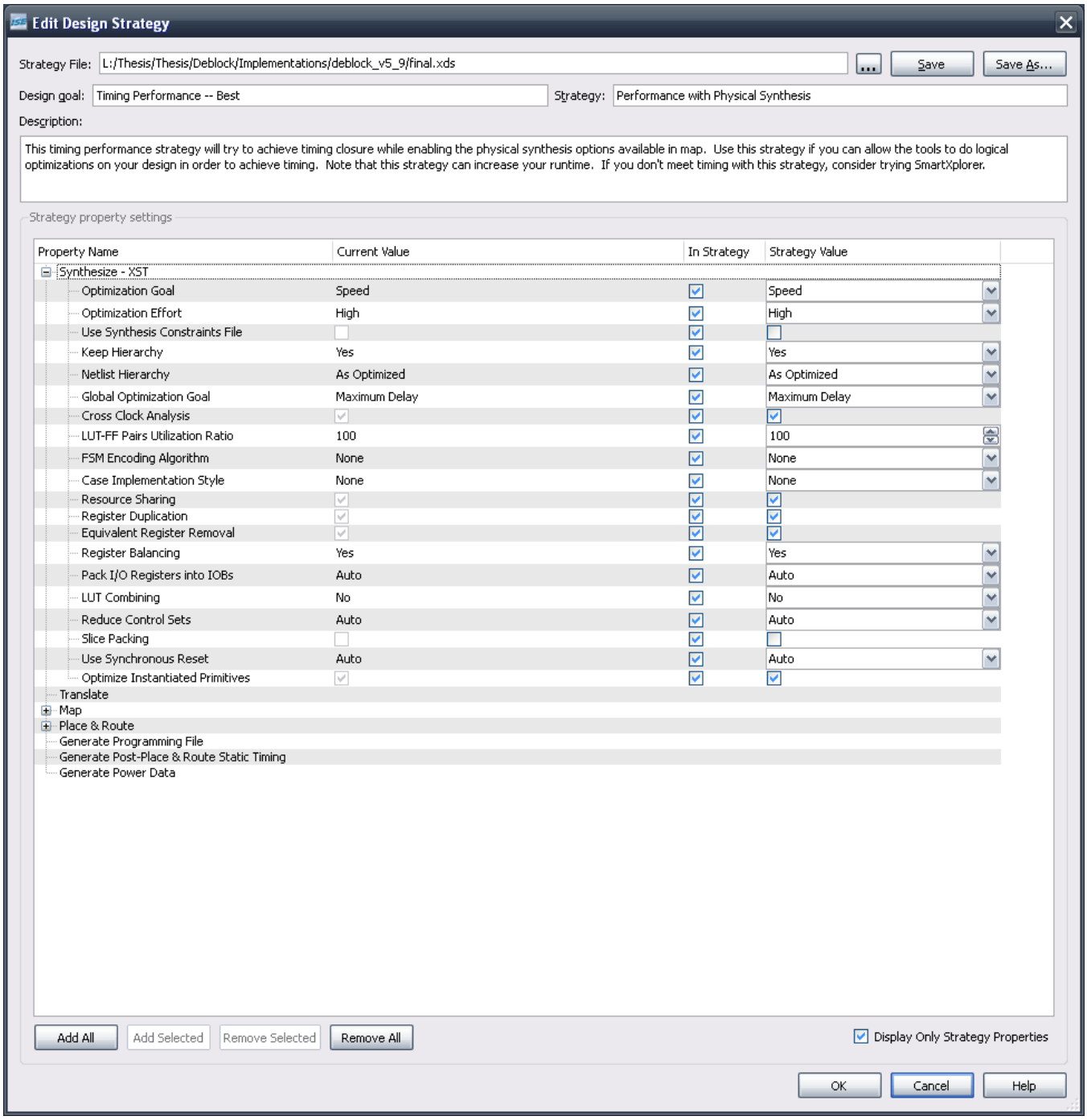
	του LCen3(Extra Register)
LCen4_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του LCen4(Extra Register)
LCen5_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του LCen5 (Extra Register)
UCen00_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του UCen00 (Extra Register)
UCen01_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του UCen01 (Extra Register)
UCen10_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του UCen10 (Extra Register)
UCen11_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του UCen11 (Extra Register)
UCen20_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του UCen20 (Extra Register)
UCen21_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του UCen21 (Extra Register)
Cen00_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του Cen00 (Extra Register)
Cen01_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του Cen01 (Extra Register)
Cen10_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του Cen10 (Extra Register)
Cen11_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του Cen11 (Extra Register)
Cen20_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του Cen20 (Extra Register)
Cen21_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του Cen21 (Extra Register)
Cen30_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του Cen30 (Extra Register)
Cen31_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του Cen31 (Extra Register)
Cen40_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του Cen40 (Extra Register)
Cen41_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του Cen41 (Extra Register)
Cen50_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του Cen50 (Extra Register)
Cen51_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του Cen51 (Extra Register)
sel_p2_31_16	Σήμα που επιλέγει την είσοδο στον καταχωρητή p2 για το μισό [31:16]
sel_p2_15_0	Σήμα που επιλέγει την είσοδο στον καταχωρητή p2 για το μισό [15:0]
sel_p1_31_16	Σήμα που επιλέγει την είσοδο στον καταχωρητή p1 για το μισό [31:16]
sel_p1_15_0	Σήμα που επιλέγει την είσοδο στον καταχωρητή p1 για το μισό [15:0]

sel_p0_31_16	Σήμα που επιλέγει την είσοδο στον καταχωρητή p0 για το μισό [31:16]
sel_p0_15_0	Σήμα που επιλέγει την είσοδο στον καταχωρητή p0 για το μισό 15:0]
sel_q0_31_16	Σήμα που επιλέγει την είσοδο στον καταχωρητή q0 για το μισό [31:16]
sel_q0_15_0	Σήμα που επιλέγει την είσοδο στον καταχωρητή q0 για το μισό [15:0]
sel_q1_31_16	Σήμα που επιλέγει την είσοδο στον καταχωρητή q1για το μισό [31:16]
sel_q1_15_0	Σήμα που επιλέγει την είσοδο στον καταχωρητή q1 για το μισό [15:0]
sel_q2_31_16	Σήμα που επιλέγει την είσοδο στον καταχωρητή q2 για το μισό [31:16]
sel_q2_15_0	Σήμα που επιλέγει την είσοδο στον καταχωρητή q2για το μισό [15:0]
sel_newP0_in	Σήμα που επιλέγει την είσοδο στον newP0
sel_newQ0_in	Σήμα που επιλέγει την είσοδο στον newQ0
sel_out_BRAM_P_in	Σήμα που επιλέγει την είσοδο στην BRAM_P
sel_out_BRAM_Q03_in	Σήμα που επιλέγει την είσοδο στην BRAM_Q03
sel_out_BRAM_Q12_in	Σήμα που επιλέγει την είσοδο στην BRAM_Q12
p1_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του p1
p0_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του p0
q0_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του q0
q1_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του q1
newP2_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του newP2
newP1_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του newP1
newP0_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του newP0
newQ0_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του newQ0
newQ1_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του newQ1
newQ2_hold	Σήμα που ενεργοποιεί τη δυνατότητα διατήρησης τιμής του newQ2
compType	Σήμα που καθορίζει να έχουμε luma ή chroma

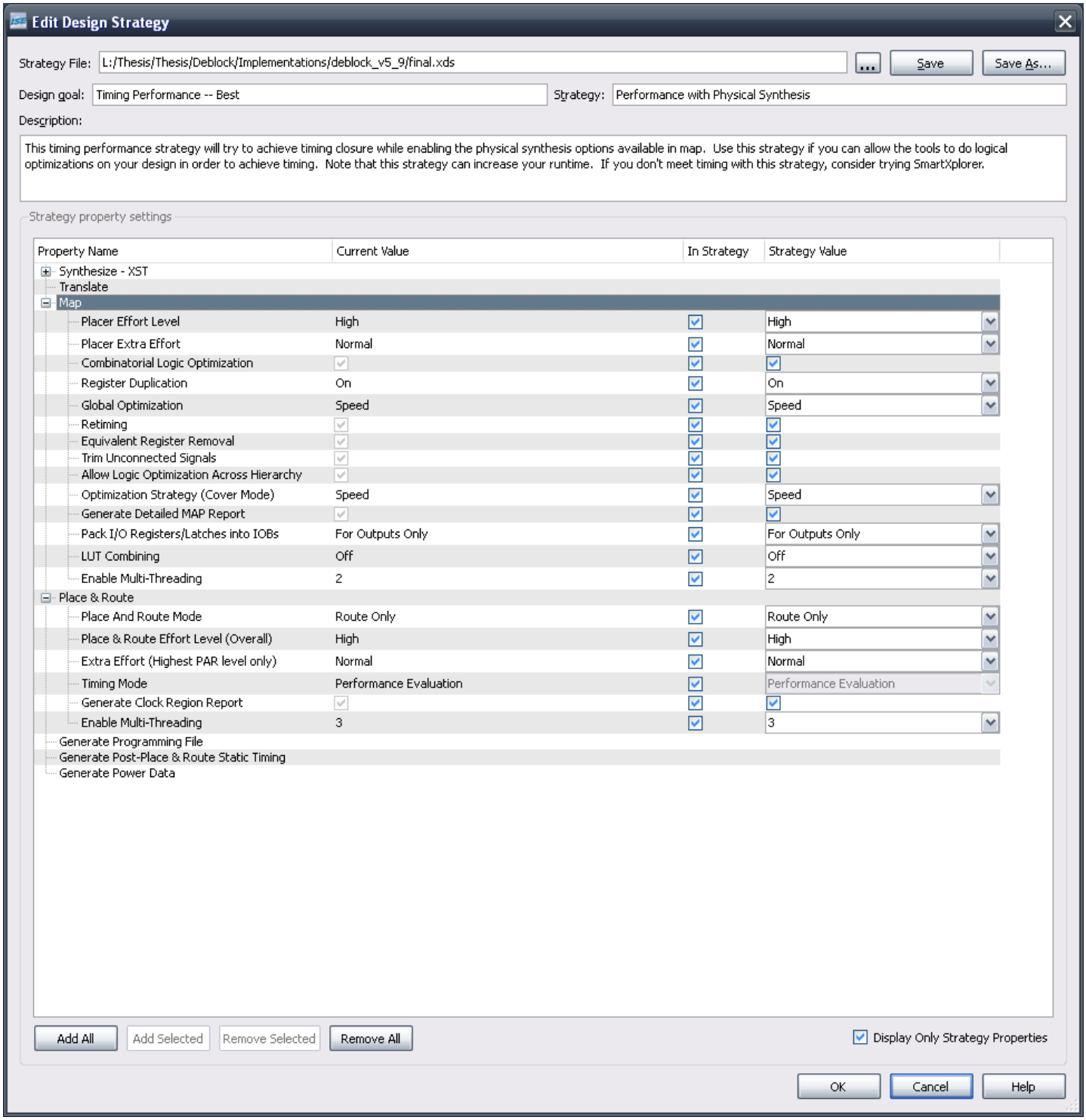
9. ΠΑΡΑΡΤΗΜΑ Α

9.1 Επιλογές Σύνθεσης και Υλοποίησης

Παρακάτω παρουσιάζονται οι επιλογές που επιλέχτηκαν για να γίνει η σύνθεση και υλοποίησης με το καλύτερο δυνατό και βέλτιστο τρόπο



Εικόνα 9-1Επιλογές synthesis



Εικόνα 9-2: Επιλογές υλοποίησης (map, place-and-route)

9.2 Διάγραμμα ροής σε στάδια