ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ
Η/Υ, ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
ΚΑΙ ΔΙΚΤΥΩΝ

Αλγόριθμοι και υλοποίηση για πρόβλεψη Macroblock

στο πρότυπο κωδικοποίησης video H.264

Μια εργασία που πραγματοποιήθηκε από
την Μαρία Κοζύρη για τις απαιτήσεις του
Διδακτορικού Διπλώματος.

Επιβλέπον Καθηγητής: Αν. Καθηγητής, Γεώργιος Σταμούλης

Βόλος, Ιούλιος 2007

ΕΥΧΑΡΙΣΤΙΕΣ

Φθάνοντας στο τέλος μιας προσπάθειας αρκετών χρόνων, νιώθω την ανάγκη να ευχαριστήσω όσους ανθρώπους με βοήθησαν στην προσπάθεια αυτή, ξεκινώντας από τον κ. Σταμούλη Γεώργιο, επιβλέποντα καθηγητή του παρόντος διδακτορικού, ο οποίος μου συμπαραστάθηκε όχι μόνο ως καθηγητής, αλλά κι ως άνθρωπος. Ιδιαίτερα σημαντική ήταν και η βοήθεια του κ. Κατσαβουνίδη Ιωάννη, ο οποίος αν και μακριά, πάντα έβρισκε τον χρόνο και τον τρόπο να με βοηθήσει και να μου δώσει τις πολύτιμες συμβουλές του. Χωρίς τη βοήθεια των δυο αυτών ανθρώπων δε θα ήταν δυνατή η ολοκλήρωση αυτής της δουλειάς.

Επίσης θα ήθελα να ευχαριστήσω τους κ.κ. Πνευματικάτο Διονύσιο και Δόλλα Απόστολο, μέλη της τριμελούς επιτροπής, για το χρόνο που αφιέρωσαν και τις σημαντικές παρατηρήσεις τους στην τελική μορφή του έργου αυτού, καθώς επίσης και τον κ. Μπέλλα Νικόλαο, ο οποίος ήταν ο άνθρωπος που έδωσε το έναυσμα για το ξεκίνημα αυτής της προσπάθειας.

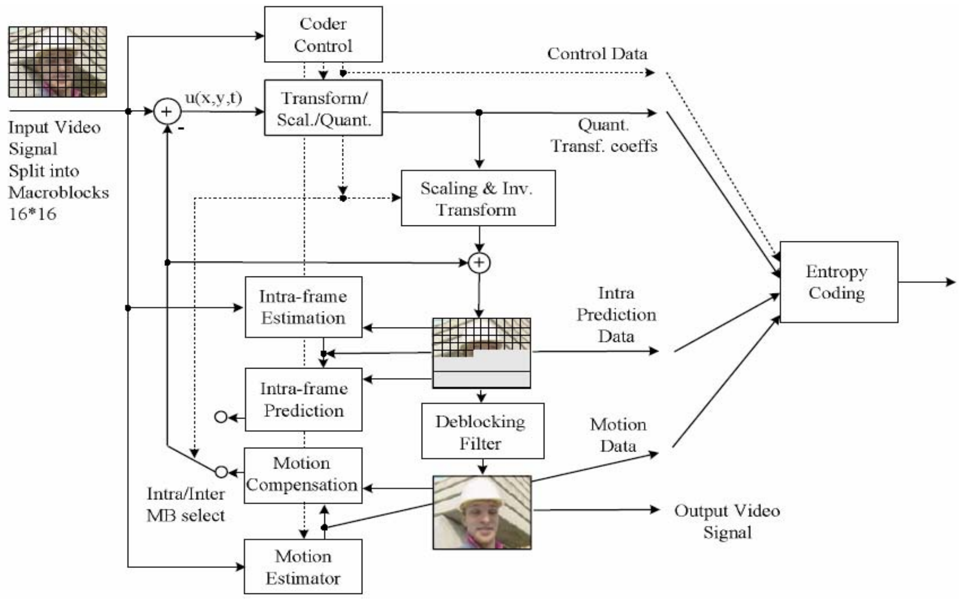
Θα ήταν παράληψη να μην αναφερθώ στα παιδιά του εργαστηριού, και ιδιαίτερα στους κ. Καραμπατζάκη Δημήτριο, κ. Δαδαλιάρη Αντώνιο, και κ. Μπουντά Δημήτριο, και να τους ευχαριστήσω για τη συνεργασία που είχαμε όλα αυτά τα χρόνια και για τις ευχάριστες ώρες που περάσαμε στο Ε5.

Τέλος, νιώθω την ανάγκη να ευχαριστήσω την οικογένεια μου, για την στήριξη τους όλα αυτά τα χρόνια, και την υπομονή που έδειξαν. Χωρίς αυτούς, το έργο αυτό δε θα είχε ολοκληρωθεί ποτέ.

1. ΕΙΣΑΓΩΓΗ

Οι τεχνικές κωδικοποίησης video είναι όλο και περισσότερο σημαντικές, εξαιτίας της ραγδαίας εμφάνισης εφαρμογών που απαιτούν ροές video χαμηλού bit – rate, όπως η video – τηλεφωνία και η video – συνεδρία. Το γεγονός αυτό κάνει απαραίτητη την ύπαρξη ενός βιομηχανικού πρότυπο για αναπαράσταση κωδικοποιημένου video με υψηλή απόδοση συμπίεσης και ενισχυμένη αντοχή σε δικτυακά περιβάλλοντα. Το πιο πρόσφατο πρότυπο στην περιοχή της κωδικοποίησης video είναι το H.264, το οποίο δημιουργήθηκε από την Joint Video Team (JVT), μια συνεργασία μεταξύ των ITU-T Video Coding Expert Group (VCEG) και ISO/IEC Moving Picture Expert Group (MPEG).

Το H.264 παρέχει παρόμοια λειτουργικότητα με τα προηγούμενα πρότυπα, επιτυγχάνοντας όμως σημαντική βελτίωση στην συμπίεση και βελτιωμένη υποστήριξη για αξιόπιστη μετάδοση. Στην Εικόνα 1 παρουσιάζεται το σχεδιάγραμμα ενός H.264 κωδικοποιητή.



Εικόνα 1 Ένας H.264 κωδικοποιητής

1.1 Αρχές κωδικοποίησης video

Ένας κωδικοποιητής video αποτελείται από 3 βασικές λειτουργικές μονάδες, το Χρονικό Μοντέλο (Temporal Model), το Χωρικό Μοντέλο (Spatial Model) και τον Κωδικοποιητής Εντροπίας (Entropy Encoder)

Στο χρονικό μοντέλο στόχος είναι να μειωθεί ο πλεονασμός μεταξύ μεταδιδόμενων καρέ. Αυτό επιτυγχάνεται με τον σχηματισμό ενός προβλεπόμενου καρέ και με αφαίρεσή του από το τωρινό καρέ. Η έξοδος του χρονικού μοντέλου είναι ένα καρέ διαφορών. Γενικότερα, όσο περισσότερο ακριβής είναι η πρόβλεψη, τόσο λιγότερη ενέργεια περιέχεται στο καρέ διαφορών.

Το χρονικό Μοντέλο περιλαμβάνει:

- **Motion Estimation** (Εκτίμηση Κίνησης): Εύρεση μιας περιοχής σε ένα καρέ αναφοράς, η οποία ταιριάζει πολύ με το τωρινό block.

- **Motion Compensation** (Συμψηφισμός Κίνησης): Αφαίρεση της καλύτερης περιοχής από το τωρινό block για τη δημιουργία ενός block διαφορών, και δημιουργία του διανύσματος κίνησης.

Στο χωρικό μοντέλο η βασική λειτουργία είναι ο επιπλέον αποσυσχετισμός της εικόνας ή της υπολειπόμενης πληροφορίας και η μετατροπή σε μια μορφή η οποία μπορεί να συμπιεστεί αποτελεσματικά (από τον κωδικοποιητή εντροπίας).

Το χωρικό μοντέλο περιλαμβάνει:

- **Χωρική Πρόβλεψη (Spatial Prediction)**: Σχηματισμός μιας πρόβλεψης ενός δείγματος εικόνας ή μιας περιοχής, από προηγουμένως μεταδιδόμενα δείγματα στο ίδιο καρέ.

- **Μετασχηματισμός (Transformation)**: Μετατροπή της εικόνας ή της υπολειπόμενης (από τον συμψηφισμό κίνησης) πληροφορίας σε ένα άλλο πεδίο.

- **Κβαντοποίηση (Quantization)**: Μείωση της ακρίβειας της μετασχηματισμένης πληροφορίας.

- **Αναδιάταξη και Κωδικοποίηση μηδενικών (Reordering and Zero Encoding)**: Ομαδοποίηση των μη μηδενικών συντελεστών και αποδοτική αναπαράσταση των μηδενικών συντελεστών.

Τέλος, έχουμε την κωδικοποίηση εντροπίας. Κατά την κωδικοποίηση αυτή πραγματοποιείται μετατροπή μιας σειρά συμβόλων που αναπαριστούν στοιχεία του video σε μια συμπιεσμένη ροή bit (bitstream), κατάλληλη για αποθήκευση ή μετάδοση. Τα σύμβολα εισόδου μπορεί να είναι κβαντισμένοι συντελεστές, διανύσματα κίνησης, επικεφαλίδες, και συμπληρωματική πληροφορία.

1.2 Χαρακτηριστικά του H.264

Στα πλαίσια αυτής της εργασίας ασχοληθήκαμε με αλγορίθμους που αφορούν την διαδικασία πρόβλεψης Macroblock, όπως αυτή ορίζεται από το πιο πρόσφατο πρότυπο κωδικοποίησης video, το H.264. Πριν προχωρήσουμε, επομένως, στην παρουσίαση των αλγορίθμων, είναι απαραίτητο να αναφερθούμε σε μερικά βασικά χαρακτηριστικά του H.264 στην διαδικασία αυτή.

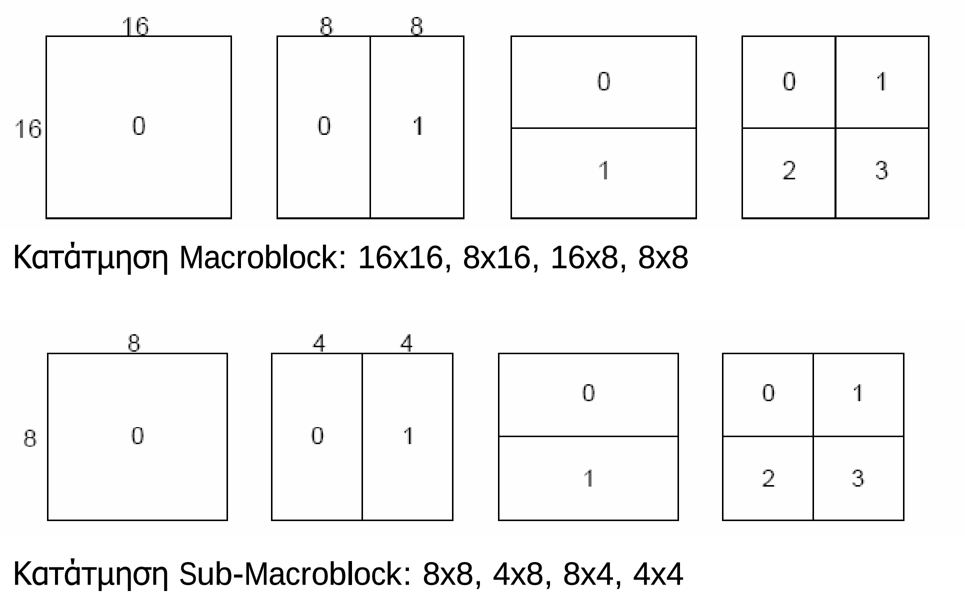
Στο H.264 υπάρχουν δυο διακριτές διαδικασίες πρόβλεψης. Η μια αφορά το χωρικό μοντέλο και ονομάζεται Intra Prediction και η δεύτερη αφορά το χρονικό και ονομάζεται Inter Prediction.

Εάν ένα block ή Macroblock κωδικοποιείται με μέθοδο Intra, τότε σχηματίζεται ένα block (ή Macroblock) πρόβλεψης, το οποίο σχηματίζεται από προηγουμένως κωδικοποιημένα και αναδομημένα blocks (ή Macroblocks) του ίδιου καρέ.

Το H.264 ορίζει 9 διαφορετικές μεθόδους πρόβλεψης για κάθε 4χ4 block και 4 για ένα 16χ16 block (Macroblock), και κάθε μέθοδος έχει τη δίκη της κατεύθυνση πρόβλεψης. Τα προβλεπόμενα δείγματα προέρχονται από έναν σταθμικό μέσο (weighted average) από προηγουμένως αποκωδικοποιημένες τιμές.

Η κωδικοποίηση Inter χρησιμοποιεί το χρονικό μοντέλο. Το προβλεπόμενο καρέ δημιουργείται από ένα ή περισσότερο, προηγούμενα ή μελλοντικά καρέ ('καρέ αναφοράς'), τα οποία έχουν ήδη κωδικοποιηθεί και μεταδοθεί. Μια πρακτική και ευρέως χρησιμοποιούμενη μέθοδος του συμψηφισμού κίνησης (motion compensation) είναι ο συμψηφισμός κίνησης ορθογώνιων τμημάτων ή 'blocks' του τωρινού καρέ (Variable Block-Size Motion Estimation (Compensation) - VBSME). Το H.264 είναι ένας block-based motion compensated hybrid transform codec. Επίσης, το H.264 υποστηρίζει μια πληθώρα μεγεθών block (τα οποία αναφέρονται και ως «μέθοδοι»), που κυμαίνονται από 16×16 μέχρι και 4×4 pixels, όπως φαίνεται στην Εικόνα 2.

.



Εικόνα 2 Μεγέθη block που υποστηρίζει το H.264

1.3 Προτεινόμενος Αλγόριθμος Απόφασης

Και στις δυο μεθόδους πρόβλεψης, Intra & Inter, δημιουργούνται προβλεπόμενα blocks, από τα οποία πρέπει να επιλεγεί το «καλύτερο». Το «καλύτερο» block θα χρησιμοποιηθεί για τη δημιουργία του block διαφορών. Ποιο είναι όμως το κριτήριο επιλογής? Δυστυχώς, δεν υπάρχει κοινώς αποδεκτό μέτρο για την ποιότητα ενός σήματος. Ένα από τα πιο ευρέως χρησιμοποιούμενα μέτρα είναι ο λόγος σήματος προς θόρυβο (signal – to – noise ratio (SNR))

$$SNR = 10 \log_{10} \frac{ενέργεια\_σήματος\_εισόδου\_κωδικοποιητή}{ενέργεια\_σήματος\_θορύβου}$$

Στην περίπτωση του video και της εικόνας, το μέτρο που χρησιμοποιείται κυρίως είναι το PSNR (peak signal-to-noise ratio) αντί του SNR. Το PSNR ορίζεται ως εξής:

- Έστω μια εικόνα ή ένα καρέ ΜχΝ, f.

- Έστω μια δεύτερη εικόνα ή καρέ ΜχΝ, F, η οποία έχει προκύψει από την αποκωδικοποίηση της κωδικοποιημένης μορφής της f.

Ορίζουμε ως μέσο τετραγωνικό σφάλμα:

$$MSE = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} [f(i,j) - F(i,j)]^2$$

Και ως ρίζα μέσου τετραγωνικού σφάλματος:

$$RMSE = \frac{1}{(MN)^{1/2}} \left(\sum_{i=0}^{M-1} \sum_{j=0}^{N-1} [f(i,j) - F(i,j)]^2 \right)^{1/2}$$

Το PSNR (db) υπολογίζεται ως:

$$PSNR = 20 \log_{10} \left(\frac{255}{RMSE} \right)$$

Το SAD ορίζεται ως εξής:

- Έστω μια εικόνα ή ένα καρέ ΜχΝ, f.

– Έστω μια δεύτερη εικόνα ή καρέ ΜχΝ, F, η οποία έχει προκύψει από την αποκωδικοποίηση της κωδικοποιημένης μορφής της f.

Τότε το SAD δίνεται από την εξίσωση:

$$SAD = \sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\left| f(i,j) - F(i,j)\right|$$

Στην επιλογή καλύτερο υποψήφιου block στην κωδικοποίηση video χρησιμοποιείται κυρίως το SAD γιατί δεν περιέχει πολλαπλασιασμούς όπως το PSNR.

Έστω ότι το κριτήριο επιλογής βασίζεται μόνο στην ποιότητα του προβλεπόμενου block, εννοώντας με τον όρο «ποιότητα» το πόσο όμοιο είναι το προβλεπόμενο block με το τωρινό. Για ένα υποψήφιο προβλεπόμενο block το SAD απαιτεί 15 προσθέσεις. Στην περίπτωση που έχουμε Intra Prediction για 4χ4 block, όπως ορίζεται από το H.264, το SAD απαιτεί 135 προσθέσεις.

Μετά τον υπολογισμό του SAD για κάθε μέθοδο πρόβλεψης, πρέπει να συγκριθούν τα αποτελέσματα για να επιλεγεί η μέθοδος που δίνει το μικρότερο SAD, επομένως η σύγκριση είναι αναπόφευκτη. Στο επίπεδο υλικού η αφαίρεση κοστίζει. Αυτό που χρειάζεται είναι η εύρεση της καλύτερης μεθόδου, και όχι πόσο καλύτερη είναι από τις υπόλοιπες. Μια ποιοτική προσέγγιση μπορεί να δώσει τα ίδια αποτελέσματα με μια ποσοτική. Επομένως, γιατί να μην συγκρίνουμε απλά, αντί να προσθέτουμε και να συγκρίνουμε; Στην ιδέα αυτή στηρίχτηκε η πρώτη προσέγγιση του προτεινόμενου αλγορίθμου, στον οποίο αντι να προσθέσουμε τις απόλυτες διαφορές κάθε μεθόδου συγκρίνουμε τις αντίστοιχες απόλυτες διαφορές δυο μεθόδων. Η σύγκριση θα καταλήξει στην μέθοδο με τις περισσότερες μικρότερες απόλυτες διαφορές. Με αυτόν τον τρόπο το στάδιο της πρόσθεσης έχει αποφευχθεί εντελώς.

Έτσι λοιπόν η πρώτη προσέγγιση του νέου αλγορίθμου έχει ως εξής:

- Πρώτα υπολογίζουμε τις απόλυτες διαφορές για κάθε μέθοδο.

$$M_{k_{ij}} = |C_{ij} - P_{k_{ij}}|$$

- Συγκρίνουμε δυο διαδοχικούς Mk πίνακες.

$$f_{k_i} = \begin{cases} 1, M_{k_i} \le M_{(k+1)_i} \\ 0, M_{k_i} > M_{(k+1)_i} \end{cases}$$

- Το διάνυσμα fk με τους περισσότερους άσους δείχνει τη μέθοδο k με τις περισσότερες μικρότερες απόλυτες διαφορές , άρα την καλύτερη μέθοδο.

Στην κωδικοποίηση video, όμως, εκτός από την ποιότητα, εξίσου σημαντικός παράγοντας είναι η συμπίεση. Το κριτήριο επιλογής θα πρέπει να περιλαμβάνει εκτός από την ποιότητα και το bit-rate κόστος. Στο H.264 το κριτήριο επιλογής του καλύτερου προβλεπόμενου block είναι το κόστος κίνησης το οποίο ορίζεται ως εξής:

$$J(\vec{m}, \lambda) = SAD(c, r(\vec{m})) + \lambda \times R(\vec{m} - \vec{p})$$

Όταν στο κριτήριο επιλογής εκτός από την ποιότητα επισέρχονται κι άλλοι παράγοντες, μια καθαρά ποιοτική δεν αρκεί. Πρέπει να βρεθεί ένας τρόπος μέτρησης της ποιότητας που θα αντικαθιστά το SAD. Το νέο αυτό μέτρο είναι το πλήθος των μεγαλύτερων απολύτων διαφορών, Sum of Greater Values – SGV. Για τον υπολογισμό του νέου αυτού μέτρου, ο αλγόριθμος που ακολουθείται είναι ο εξής:

- Πρώτα υπολογίζουμε τις απόλυτες διαφορές για κάθε μέθοδο.

$$M_{ij} = | C_{ij} - P_{ij} |$$

- Συγκρίνουμε δυο διαδοχικούς Mk πίνακες.

$$f_{k_i} = \begin{cases} 1, M_{k_i} > M_{(k+1)_i} \\ 0, M_{k_i} \le M_{(k+1)_i} \end{cases}$$

- Το SGV ισούται με:

$$F_k = \sum_{i=0}^{S} f_{k_i}$$

Σύμφωνα με τα παραπάνω, το κόστος κίνησης, με αντικατάσταση του SAD από το SGV μπορεί να γίνει:

$$J(\vec{m}_1, \vec{m}_2, \lambda) = F(c, r(\vec{m}_1), r(\vec{m}_2)) + \lambda \times R(\vec{m}_1 - \vec{p})$$

Όπου $F(c, r(\vec{m}_1), r(\vec{m}_2))$ είναι το SGV που προκύπτει από τη σύγκριση του υποψήφιου block αναφοράς με διάνυσμα κίνησης $\vec{m}_1$ και ενός άλλου υποψήφιου block αναφοράς με διάνυσμα κίνησης $\vec{m}_2$

Ο καινούργιος αλγόριθμος υλοποιήθηκε και αξιολογήθηκε τόσο σε επίπεδο λογισμικού όσο και σε επίπεδο υλικού. Τα πειραματικά αποτελέσματα έδειξαν ότι ο νέος αλγόριθμος:

- Στο επίπεδο λογισμικού:
    - Διατηρεί την ποιότητα.
    - Δεν αυξάνει σημαντικά το μέγεθος του παραγόμενου αρχείου.
- Στο επίπεδο υλικού:
    - Η αρχιτεκτονική είναι αρκετά γρήγορη ώστε να υποστηρίζει μέχρι και SHDTV.
    - Η κατανάλωση ισχύος είναι η μικρότερη σε σχέση με αυτή των υπαρχόντων εργασιών.

Στο Παράρτημα που ακολουθεί βρίσκεται λεπτομερής ανάλυση όλων όσων αναφέρθηκαν παραπάνω.

ΠΑΡΑΡΤΗΜΑ

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Table *Page Number*

1. Introduction

1.1. Aim, objectives and contribution of the thesis

Video has always been the backbone of multimedia technology. In the last two decades, the field of video coding has been revolutionized by the advent of various standards like H.261 to H.263 ([1], [3], [4]) and MPEG-1 to MPEG-4 ([2], [3], [5]), each addressing different aspects of multimedia. H.264 [6] is a new standard which adds one more step in the endeavour towards video coding excellence and provides one-stop solution for wide range of applications. The standard has been developed by the Joint Video Team (JVT) comprised of both ISO/IEC and ITU-T. The primary goal of H.264 is to achieve higher compression while preserving video quality. The motivation for compression is to compensate for the ever-present constraints of limited channel capacity.

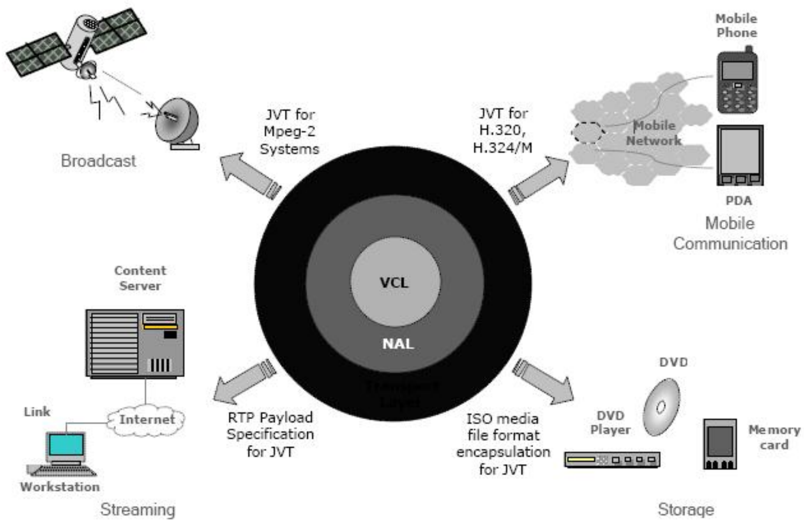


Figure 1 H.264 and its applications

This increased compression efficiency of the new ITU-T H.264/MPEG-4 Advanced Video Coding (AVC) standard will lead to new application areas or improve already existed (Figure 1). Applications concerning broadcasting over cable, satellite, cable modem, terrestrial, etc. (now using mostly using H.222.0 | MPEG-2 systems [7]), wireline and wireless real-time conversational services (e.g., using H.32x [8]-[10] or SIP [11]), Internet or LAN video streaming (using RTP/IP [12]), storage formats (e.g. digital versatile disk (DVD), digital camcorders, and personal video recorders), will benefit from the new standard. As for the field of mobile communication, H.264 will play an important role because the compression efficiency will be doubled in comparison to the coding schemes previously specified by Third-Generation Mobile (3GPP and 3GPP2) for streaming. The video coding technique used in H.264 follows a flexibility to be used in low-delay real-time applications.

The keys to high coding efficiency of H.264 are the two prediction modes (Intra & Inter) provided by the standard which adopt many new features such as variable block size searching, motion vector prediction etc. However, these result in a considerably higher encoder complexity that adversely affects speed and power, which are both significant for many of the applications targeted by the standard. Therefore, it is of high importance to design architectures that minimize the speed and power overhead of the prediction modes. In this work we present a novel matching criterion for prediction, as well as the algorithm and the architecture that implements it. The use of this criterion in a H.264 encoder can provide a power efficient hardware implementation without perceivable degradation in coding efficiency or video quality.

1.2. Video Compression

Video is a sequence of still images, consequently, video signals are just a way of transferring visual information. Therefore, many of the principles that apply in image compression, are used as they are in video compression. There are many ways of representing visual information, especially when concerning colour information. A colour space is a mathematical representation (either analogically or digitally) of a set of colours. Some of the most widely used, in image and video coding, colour spaces are RGB, YUV, YPbPr, YCbCr and others. Although there is a variety of colour spaces, all of them are related mathematically to RGB. In the latest video coding standards, among them in H.264, the colours space used is YCbCr, a digitized version of YUV. Component Y is called luma and represents the luminance, i.e. the brightness of the image, while components Cb, Cr, are called chroma and represent the colour of the image. There are several YCbCr sampling formats, such as 4:4:4, 4:2:2, 4:1:1 and 4:2:0, and although many video standards, among them the first draft versions of H.264, supported only the 4:2:0 sampling, the final version of the H.264 standard supports 4:4:4, 4:2:2 and 4:2:0 sampling [13].

Since video is a series of still images it is reasonable to think that the reproduction of video can be accomplished by displaying each still image consecutively, one after the other. This is the basic idea of the progressive video format. Apart from that, there is also the interlaced video format, where the odd-numbered lines in an image are transmitted first, followed by the even numbered lines. The odd-numbered lines comprise the top field, whereas the even-numbered lines comprise the bottom field of a video frame. The term picture will be used, either referring to a field or a frame.

As basis of many of the first video compression research, in the mid 1960s [14], [15], was the idea that because video is sequence of still images, video compression can be accomplished by compressing each image separately. This way of compression utilizes only the spatial redundancy among regions within the same picture, as done in JPEG [16], the most widespread standard for image coding. Nonetheless, one of the main characteristics of video, that distinguishes it from image, is motion. Motion is the change of the position of an object within the time. Therefore, apart from space, the other basic element of video is time, and as there is spatial redundancy because of similarities in the space, there is also temporal redundancy because of similarities in time. As a result, video compression can be improved by taking into account these temporal redundancies, as recognised even from the 1929 [17]. In the first digital video coding standard, ITU-T H.120 [18], temporal redundancy was reduced by coding only the changes in a video scene. This method was called Conditional Replenishment (CR) [19].

The modern video coding standards use two type of prediction. The first is called Intra Prediction and is based on the similarities among neighbouring areas within the same picture of video, i.e. it tries to take advantage of the large amount of spatial redundancy in the video content. The other is called Inter Prediction and is based on the fact that pictures that are timing adjacent in a video sequence tend to be very similar with each other, i.e. Inter Prediction take advantage of temporal redundancy. The video codecs which use both, Inter and Intra Prediction are called hybrid codecs (a term originated by Habibi [20] but used with a slight difference in its original meaning).

One of the basic components of modern video codecs, that was missing from the first version of H.120 and from [20] is motion-compensated prediction (MCP). MCP, which belongs to the category of temporal prediction, was introduced in the early 1970s [21] and was widely published in the form nowadays is used in [22]. In MCP, the encoder searches for a block of a previous or future frame that has already been encoded and transmitted, which is similar to the block of the new frame to be encoded. The best matching block is called predictor and the process of forming the predictor is known as Motion Compensation (MC). After the predictor is found, the encoder sends (as side information) a motion vector (MV), i.e. the spatial displacement between the predictor and the current block, telling the decoder which block of the previous or future frame it will use to predict the block of the current frame. The process of search for the best predictor and motion vector is called Motion Estimation (ME).

Improvements of MCP techniques lead to significant improvement in the compression efficiency, and they are what mostly characterize modern video coding standards when comparing them from generation to generation. As H.264 is the video coding standard studied in this thesis, we are going to outline some of its basic improvements in MCP:

1. *Fractional-sample-accurate MCP* [23]. Refers to the case when the motion vectors can have non-integer precision. Quarter or half samples are actually interpolated sub-samples caused by fractional motion vectors. Based on the vectors and full-samples, the sub-samples can be calculated by applying an one-dimensional 6-tap FIR filter horizontally and/or vertically [24]. A theoretical motivation for this can be found in [25], [26].

2. *MVs over picture boundaries* [27]. Refers to the solution of the problem for motion representation for samples in the boundary of a picture.

3. *Bi-Predictive MCP* [28] Refers to the case when the motion compensation of a frame is done with the use of the average of a previous and a future frame.

4. *Variable block size MCP* [29]. (VBSMC) is the use of Block Motion Compensation (i.e. a frame is partitioned into fixed size blocks –in the case of H.264 these are named Macroblocks and are 16x16 blocks – and each block is motion compensated separately) with the ability for the encoder to dynamically select the size of the blocks. When coding video, the use of larger blocks can reduce the number of bits needed to represent the motion vectors, while the use of smaller blocks can result in a smaller amount of prediction residual information to encode.

5. *Multi-Picture MCP* [30], [31]. Refers to the case when more than one previously encoded and transmitted frames are used for motion compensation of the current frame.

6. *Multi-hypothesis and weighted MCP* [32]-[35] Refers to the case where MCP with one prediction signal is extended to the linear, weighted superposition of several motion compensated signals. This can be accomplished in many ways, two of which are overlapped block motion compensation as in [32] and [33] (used in H.263 but not in H.264) and conventional bi-directional MCP. H.264 uses a unified generalization [34], which is the result of the combination of bi-predictive MCP, multi-picture MCP and linearly-weighted MCP.

A historical analysis of video coding can be found in [36], whilst in [37] the basic concepts of video coding design are explained, along with how these features have been integrated into the international video coding standards.

1.3. Thesis' structure

The contribution of this thesis is the presentation of a novel matching criterion for prediction in the latest video coding standard, H.264. Furthermore the algorithm that uses this matching criterion was implemented and experimental results showed that the novel matching criterion and the corresponding algorithm is competitive to those used so far in the field of video coding. Last but not least, this thesis presents the architecture of the new algorithm. The architecture is presented in detail, as some novel implementations, such as an efficient comparator, were produced. Experimental results showed that the presented architecture can achieve significant degradation in execution time and in power consumption. Concluding, the use of the criterion, presented in this thesis, in a H.264 encoder can provide a power efficient hardware implementation without perceivable degradation in coding efficiency or video quality.

The structure of the thesis is as follow:

Chapter 2 covers essential background material that is necessary for understanding both the H.264 standard and its relation to the new matching criterion. More precisely, Chapter 2 introduces the basic concepts of video coding. Furthermore, it presents the H.264 video coding standard.

Chapter 3 introduces the new matching criterion and the algorithms that implement it. This new matching criterion is intended to replace Sum of Absolute Difference (SAD) in the prediction process of H.264. Therefore, in Chapter 3, apart from the new algorithms, SAD and some basic concepts from the theory of norms are presented.

Chapter 4 deals with the software implementations of the new algorithms. The basic framework, within which all relevant research works lie for evaluating their effectiveness, is the JM reference software. Chapter 4 describes, concisely, the structure of the JM Reference Software, goes on with the description of the process followed in order to integrate the new algorithms with it and finally presents the simulation results.

Chapter 5 describes the hardware implementation of the new algorithms. Firstly, introduces the two architectures designed for the two new algorithms presented in Chapter 3. These architectures are used in Intra and Inter Prediction, respectively. So, Chapter 5 describes both the architectures of Intra and Inter Prediction, along with their hardware implementations and performance analysis. Finally, a comparison, between the presented results and those found in related research work, is performed.

Chapter 6 completes this thesis with the presentations of the conclusions and a reference to the future work that could follow the present.

2. VIDEO CODING CONCEPTS AND THE H.264 STANDARD

2.1. Introduction

Within the last two decades, multimedia has become an integral part of the way we create, communicate and consume visual information, in other words of the way we live. The "flagship" of multimedia technology has always been video, and its startling evolvement is mostly obliged to continuous advent of novel digital video compression techniques. These digital video compression techniques are mainly described by the various video coding standards, the latest of which is H.264 /AVC (Advanced Video Codign) Standard.

Video coding is the process of compressing and decompressing a digital video signal. In this chapter we present many of the basic concepts of video coding, and especially those concerning the video encoding process, as encoding is probably the largest research area in video coding.

Furthermore, this chapter deals with the H.264 video coding standard as it comprises the framework of the current research. H.264 has been developed by the Moving Picture Experts Group and the Video Coding Experts Group (MPEG and VCEG) and promises to outperform the earlier MPEG-4 and H.263 standards, providing better compression of video images.

2.2. Video Codec

Compression is the process of compacting data into a smaller number of bits. Video compression (video coding) is the process of compacting a digital video sequence into a smaller number of bits. Compression involves a complementary pair of systems, an encoder (which makes the compression) and a decoder (which makes the decompression). The encoder converts the source data into a compressed form prior to transmission or storage and the decoder converts the compressed form back into a representation of the original video data. If the representation produced by the decoder is identical to the original video sequence, then the coding process is called *lossless*, whereas, when the reconstruction of the original video sequence is imperfect, the coding process is called *lossy*. The encoder decoder pair is often described as a CODEC (enCOder/DECoder) (Figure 2)

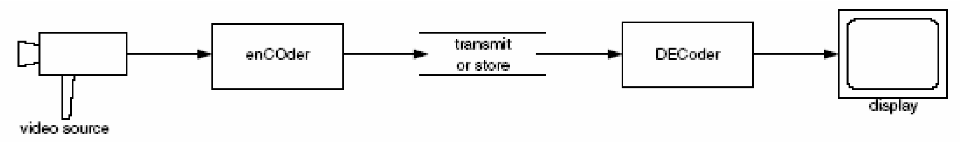


Figure 2 A video CODEC

Data compression is achieved by removing redundancy, i.e. components that are not necessary for faithful reproduction of the data. Many types of data contain statistical redundancy and can be effectively compressed using lossless compression. However, lossless compression of image and video information gives only a moderate amount of compression, therefore, lossy compression is necessary to achieve higher compression. Lossy video compression systems are based on the principle of removing subjective redundancy, elements of the image or video sequence that can be removed without significantly affecting the viewer's perception of visual quality[38].

In video coding there are two kinds of redundancy that can be exploited in order to achieve compression, temporal and spatial. Temporal redundancy refers to time. It is very likely that frames of video referring to nearby timing moments, i.e. frames captured at around the same time, are similar. Therefore, in the temporal domain, there is usually high correlation between timing adjacent frames. Spatial redundancy refers to space. It is very likely that neighbouring pixels within the same frame of a video sequence are similar, i.e. pixels that are close to each other have similar values. Therefore, in the spatial domain, there is usually high correlation between adjacent (in space) pixels.

A video encoder consists of three main functional units: a temporal model, a spatial model and an entropy encoder. The goal of the temporal model is to reduce redundancy between transmitted frames by forming a prediction frame and subtracting this from the current source frame. As already mentioned, there are two types of redundancy, temporal and spatial. Therefore, there are two ways for forming the prediction frame, one called temporal prediction and another called spatial (or image) prediction. The better the prediction is, the lower is the energy in the residual, thus fewer bits are needed to represent it. Input to the temporal model is an uncompressed (source) video frame and output is a residual frame (created by subtracting the prediction from the actual current frame) and a set of model parameters, such as motion vectors that describe how the motion was compensated. The residual frame forms the input to the spatial model, which reduces the spatial redundancy. This is accomplished by applying a transform to the residual samples and quantizing the results. Transform is used to convert spatial image pixel values to transform coefficient values. The desired effect is that most of the energy in the image will be contained in a few large transform coefficients. The coefficients are quantized to remove insignificant values, leaving a small number of significant coefficients that provide a more compact representation of the residual frame. Output of the spatial model is a set of quantized transform coefficients. Finally, the parameters of the temporal model (e.g. motion vectors) and the output of the spatial model form the input to entropy encoder, where the statistical redundancy is removed. This is achieved by, for example, representing commonly-occurring vectors and coefficients by short binary codes and rare – occurring with larger binary codes. Output of the entropy encoder is a compressed bit-stream or file that can be transmitted and/or stored.

A video decoder performs the inverse process in order to reconstruct a video sequence from a compressed bit – stream. The header information, the parameters of the temporal model and the coefficients, that are present in the compressed bit – stream, are decoded with the use of an entropy decoder. The decoded coefficients are rescaled and inverse transformed to reconstruct a representation of the residual frame. The decoder uses the information provided by the motion vector parameters, along with previously decoded frames, to create a prediction frame. The current frame is reconstructed by adding the residual frame to the prediction.

2.2.1. *Spatial prediction*

In spatial prediction the encoder creates a prediction of a region of the current (source) frame based on previously encoded samples of the same frame and subtracts this prediction from the current region to form a residual. Intra coding refers to the case where only spatial redundancies within a video picture are exploited. The resulting frame is referred to as an I-picture (or I- frame). I-pictures are typically encoded by directly applying the transform to the different blocks in a frame. As a consequence, encoded I-pictures are large in size since no temporal information is used as part of the encoding process. In order to increase the efficiency of the intra coding process, spatial correlation between adjacent macroblocks in a given frame is exploited. The idea is based on the observation that adjacent blocks tend to have similar properties [39].

Therefore, as a first step in the encoding process for a given block, one may predict the block of interest from data in neighbouring blocks as shown in Figure 3. Since, in most video codecs, blocks are coded in left to right, top to bottom order (known as raster scan order) intra prediction is performed using data in blocks above and to the left of the block being predicted. Specifically, the data used is from:

- the lower right pixel of the block above and to the left
- the lower row of pixel of the block above
- the lower row of pixel of the block above and to the right
- the right column of pixel of the block to the left

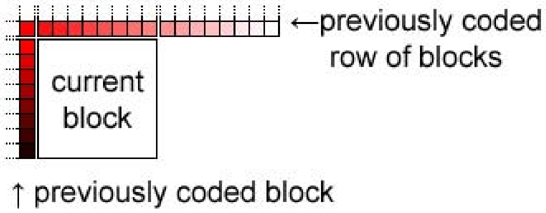


Figure 3 Data used in Intra Prediction

Every video codec defines modes of intra predicting blocks. The mode used to encode the prediction of a block is selected based on the textures and gradients in the video source data. To predict a block amid a flat field of colour, an intra prediction mode, that copies the lower right pixel of the block above and to the left into every pixel position in the predicted block, might be used, as shown in Figure 4.

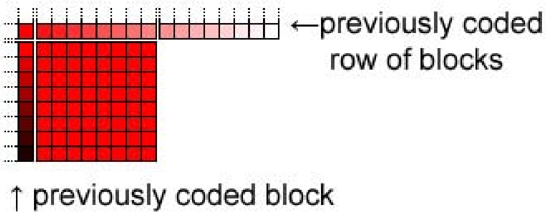


Figure 4 Intra prediction mode for flat field of colour

To predict a block amid a left to right gradient, an intra prediction mode, that copies the lower line of the block above to every line in the predicted block, might be used, as shown in Figure 5.

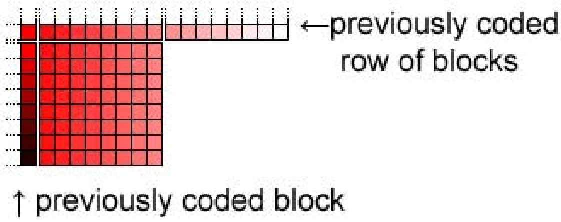


Figure 5 Intra Prediction mode for left to right gradient

To predict a block amid a top to bottom gradient, an intra prediction mode, that copies the rightmost column of the block to the left to every column in the predicted block, might be used, as shown in Figure 6.

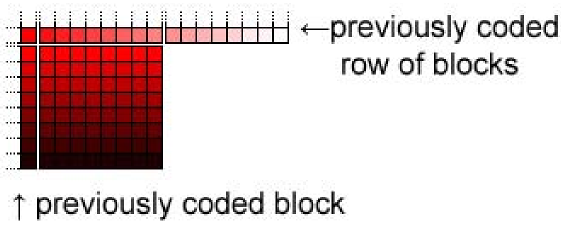


Figure 6 Intra Prediction mode for top to bottom gradient

To predict a block amid a diagonal gradient, an intra prediction mode, that copies the lower line of the block above and the block above and to the right to the diagonally down corresponding pixel positions in the predicted block, might be used, as shown in Figure 7.

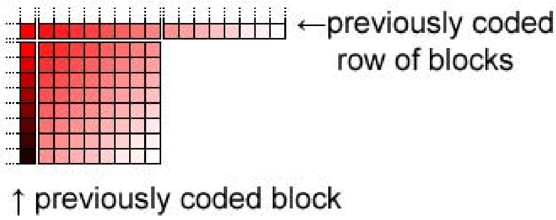


Figure 7 Intra prediction mode for diagonal gradient

These are just a few intra prediction modes. Different variations of intra prediction modes exist in different video codecs.

Aside from its value at scene changes, periodic intra predicted frames within video sequences are valuable for other important reasons. Occasionally an error might occur in a video program due to dust on a DVD disc or interference in a broadcast transmission. The error can cause one or more corrupted blocks. Inter predicted frames can copy and multiply the corrupted block from one frame of video to the next. Since an intra predicted frame does not depend on previously coded frames, it will cause a complete, independent, fresh redrawing of the video image, which will correct any errors encountered.

Periodic intra predicted frames in a video sequence also enable methods known as trick play, such as fast play (fast forward) and reverse play (rewind). In a fast playing mode, the video decoder shows only some of the coded frames at regular intervals in the frame sequence. In fast playing mode, the decoder does not have time to draw all inter predicted frames in order to determine the correct frame at the interval required. If intra predicted frames are included in the sequence then the decoder can skip P and B (inter) frames up to the next I frame that it needs to draw the next frame at the required interval.

Similarly, playing a video sequence in reverse requires the decoder to move backwards in the video sequence and draw frames in reverse order. This can only be performed if the decoder can use periodic I frames in order to determine how to draw groups of dependent P and B (inter) frames for the reversed video sequence [40].

2.2.2. *Temporal prediction*

Temporal prediction uses motion estimation and compensation, where the encoder creates a prediction of a region of the current (source) frame based on one or more, previous or future frames ('reference frames') and subtracts this prediction from the current region to form a residual. If the frame is predicted using only previous encoded frames is called P-Frame, whereas if both previous and future frames are used in the prediction the frame is called B-Frame (Figure 8).

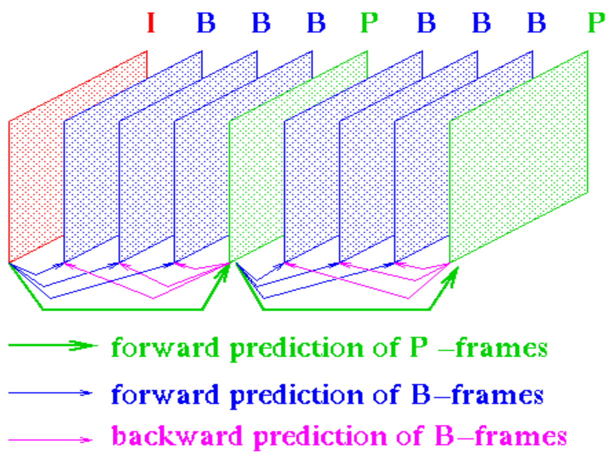


Figure 8 P and B frames

Successive video frames may contain the same objects (still or moving). Motion estimation examines the movement of objects in an image sequence to try to obtain vectors representing the estimated motion. Motion compensation uses the knowledge of object motion so obtained to achieve data compression. Figure 9 shows the (displacement or motion) vectors produced by the motion estimation process and the residual frame produced by motion compensation. In Inter coding, motion estimation and compensation have become powerful techniques to eliminate the temporal redundancy due to high correlation between consecutive frames.

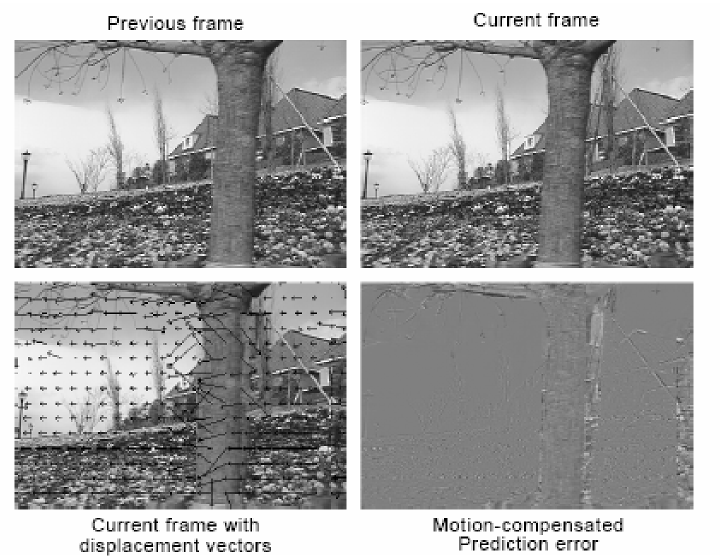


Figure 9 Motion Estimation and Compensation

In real video scenes, motion can be a complex combination of translation and rotation. Such motion is difficult to estimate and may require large amounts of processing. However, translational motion is easily estimated and has been used successfully for motion compensated coding.

There are two mainstream techniques of motion estimation: pel-recursive algorithm (PRA) and block-matching algorithm (BMA). PRAs are iterative refining of motion estimation for individual pels (pixels) by gradient methods. BMAs assume that all pixels within a block has the same motion activity. BMAs estimate motion on the basis of rectangular blocks and produce one motion vector for each block. PRAs involve more computational complexity and less regularity, so they are difficult to realize in hardware. In general, BMAs are more suitable for a simple hardware realization because of their regularity and simplicity.

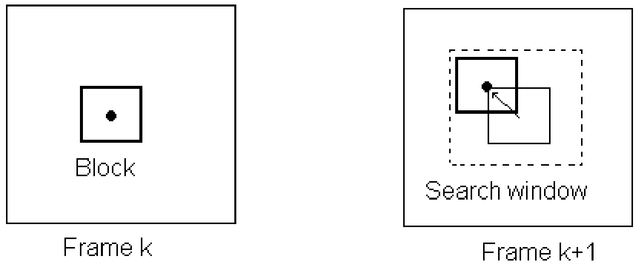


Figure 10 Block Matching Motion Estimation

Figure 10 illustrates a process of block-matching algorithm. In a typical BMA, each frame is divided into blocks, each of which consists of luminance and chrominance blocks. Usually, for coding efficiency, motion estimation is performed only on the luminance block. The following procedure is carried out for each block of M x N (luma) samples in the current frame:

1. Search an area in the reference frame(s) (past or future, previously coded and transmitted) to find a matching M x N sample region. This is carried out by comparing the M x N block in the current frame with some or all possible M x N regions in the search area (usually a region centred on the current block position) and finding the region that gives the best “match”. There are a number of criteria to evaluate the "goodness" of a match and some of them are:

    i. Cross Correlation Function

    ii. Pel Difference Classification (PDC)

    iii. Mean Absolute Difference

    iv. Mean Squared Difference

    v. Integral Projection

Some of these criteria are simple to evaluate, while others are more involved. Different kinds of algorithms use different criteria for comparison of blocks. This process of finding the best match is known as *motion estimation.*

2. The chosen candidate region becomes the predictor for the current M x N block and is subtracted from the current block to form a residual M x N block. This process is known as *motion compensation*.

3. The residual block is encoded and transmitted and the offset between the position of the current block and that of the predictor (*motion vector*) is also transmitted.

The decoder uses the received motion vector to re-create the predictor region and decodes the residual block, adds it to the predictor and reconstructs a version of the original block.

2.3. The H.264 Standard

The H.264 was developed jointly by SG16 Q.6 group of ITU-T, also known as VCEG (Video Coding Experts Group), and by ISO/IEC JTC1/SC29/WG11, also known as MPEG (Moving Picture Experts Group). The reason of the development of the H.264 standard was the increased need for higher compression of video by applications such as videoconferencing, digital storage media, television broadcasting, internet streaming and communication. Furthermore, H.264 was designed is such a way that permits the use of the coded video representation in a flexible manner for a wide variety of network environments [6].

In order to serve this wide range of applications and environments, H.264 had to consider requirements for various bit-rates, resolutions, qualities and services and to integrate them into a single syntax. Therefore the syntax provided by the standard is quite complicated and impractical (in many cases) to be implemented. Considering the practicality of implementation, H.264 specifies a number of subsets of the syntax by the means of "profiles" and "levels". Profiles and levels specify restrictions on bitstreams and hence limit on the capabilities needed to decode the bitstreams. H.264 specifies seven profiles, Baseline, Main, Extended, High, High10, High 4:2:4 and High 4:4:4. A full list of the coding functions supported by each profile is given in Table 91, in Appendix B. The levels specify a set of limits on parameters, such as coded bit-rate, resolution, quality and others. The same set of level definitions is used with all profiles, but individual implementations may support a different level for each supported profile.

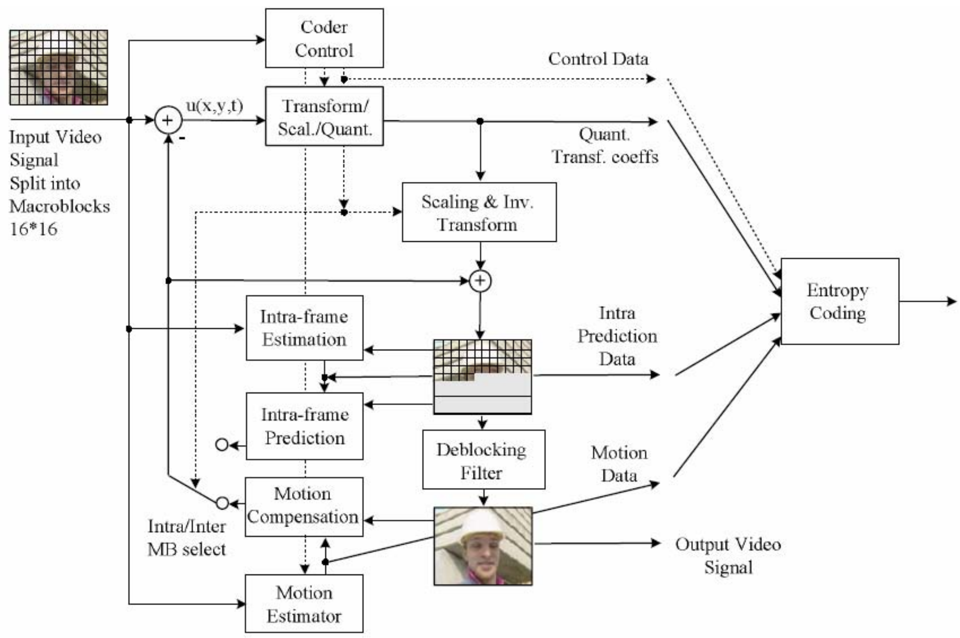


Figure 11 H.264 video encoder

The H.264 standard specifies the syntax of an encoded video bitstream together with the decoding process of this bitstream. The encoding process is not defined by the standard, therefore it provides a research framework in which many novel ideas, algorithms and implementations can be developed. Within this framework lies the present work, therefore the rest of this Section is dedicated mostly in an H.264 encoder, rather than the standard itself.

An H.264 encoder, a block diagram of which is shown in Figure 11 [19], consists of two dataflow paths, an encoding path and a reconstruction path which is identical to an H.264 decoder. The input of the encoder is a sequence of video frames, each of which is processed in units of Macroblocks (16x16 samples) and the encoding process has as follow:

1. A prediction Macroblock is formed, using either Intra or Inter prediction, as analysed in the following Sections. In the case of Inter prediction, along with the prediction block, a number of motion vectors are specified.

2. The prediction Macroblock is subtracted by the current (source) Macroblock and a residual Macroblock is created.

3. The residual Macroblock is transformed and quantized. Although previous standards used mainly the discrete cosine transform (DCT) [41], H.264 specifies an integer transform, which is based on DCT, yet with some main differences [42], [43], [44]:

    a. All operations can be carried out with integer arithmetic, without loss of accuracy.

    b. The inverse transform is fully specified in the H.264 standard and if this specification is followed correctly, mismatch between encoders and decoders should not occur.

    c. The core part of the transform is multiply-free, i.e. it only requires additions and shifts.

    d. A scaling multiplication (part of the complete transform) is integrated into the quantizer (reducing the total number of multiplications).

4. The quantized transformed coefficients are entropy encoded. H.264 supports two alternative entropy coding schemes, the Context-Adaptive Variable Length Coding (CAVLC) [45] and Context-Adaptive Binary Arithmetic Coding (CABAC) [46].

As already mentioned, the H.264 encoder has a reconstruction path, in order to reconstruct encoded frames for use as predictors in Intra or Inter Prediction. After the quantized transformed coefficients are formed, apart from being entropy encoded they are also rescaled and inverse transformed to produce a residual Macroblock. The residual Macroblock is added to the prediction Macroblock and a reconstructed Macroblock uF, which is a decoded version of the original (source) Macroblock, is formed. A filter, known as Deblocking Filter [47], is applied to the reconstructed Macroblock uF to reduce the effects of blocking distortion and produces a filtered reconstructed Macroblock F. Intra prediction makes use of the unfiltered reconstructed Macroblocks, whereas Inter Prediction uses filtered reconstructed Macroblocks.

2.3.1. Intra Prediction

In Section 2.2.1 we have seen how spatial (intra) prediction is performed, generally, in any video encoder. The H.264 standard exploits the spatial correlation between adjacent macroblocks/blocks for Intra prediction. That is, the current macroblock/block is predicted by adjacent pixels in the upper and the left macroblocks/blocks of the same frame that have been decoded earlier. H.264 offers a rich set of prediction patterns for Intra prediction, more specifically. nine prediction modes for 4x4 luma blocks, nine prediction modes for 8x8 luma blocks and four prediction modes for 16x16 luma blocks. Each mode has its own direction of prediction and the predicted samples are obtained from a weighted average of decoded values of neighbourhood macroblocks/blocks [39].

A further intra coding mode, I PCM, enables an encoder to transmit the values of the image samples directly (without prediction or transformation). In some special cases (e.g. anomalous image content and/or very low quantizer parameters), this mode may be more efficient than the ‘usual’ process of intra prediction, transformation, quantization and entropy coding. Including the I PCM option makes it possible to place an absolute limit on the number of bits that may be contained in a coded macroblock without constraining decoded image quality [38].

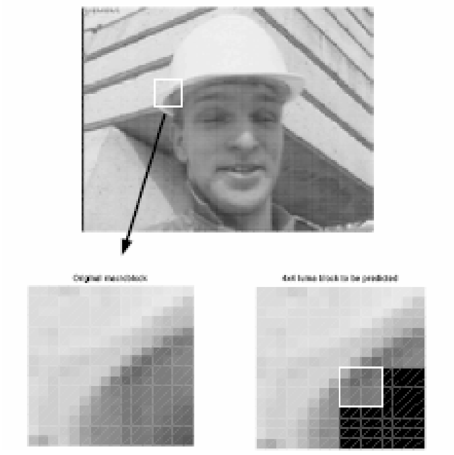


Figure 12 Original macroblock and 4x4 block to be predicted

Let's suppose we have a 4 x 4 luma block (part of the highlighted macroblock in Figure 12) that is required to be predicted. The samples above and to the left, labelled A–M in Figure 13 have previously been encoded and reconstructed and are therefore available in the encoder and decoder to form a prediction reference. The samples a, b, c, . . . , p of the prediction block P (Figure 13) are calculated based on the samples A–M as indicated by each mode. Mode 2 (DC prediction) is modified depending on which samples A–M have previously been coded; each of the other modes may only be used if all of the required prediction samples are available. Note that if samples E, F, G and H have not yet been decoded, the value of sample D is copied to these positions and they are marked as 'available'. The nine prediction modes for a 4x4 block in Intra Prediction, as they are specified in H.264 are:

– Mode 0 (Vertical) The upper samples A, B, C, D are extrapolated vertically.

– Mode 1 (Horizontal) The left samples I, J, K, L are extrapolated horizontally.

- Mode 2 (DC) All samples in P are predicted by the mean of samples A . . . D and I . . . L.
- Mode 3 (Diagonal Down-Left) The samples are interpolated at a 45° angle between lower-left and upper-right.
- Mode 4 (Diagonal Down-Right) The samples are extrapolated at a 45° angle down and to the right.
- Mode 5 (Vertical-Right) Extrapolation at an angle of approximately 26.6° to the left of vertical (width/height = 1/2).
- Mode 6 (Horizontal-Down) Extrapolation at an angle of approximately 26.6° below horizontal.
- Mode 7 (Vertical-Left) Extrapolation (or interpolation) at an angle of approximately 26.6° to the right of vertical.
- Mode 8 (Horizontal-Up) Interpolation at an angle of approximately 26.6° above horizontal.

M	A	B	C	D	E	F	G	H
I	a	b	c	d				
J	e	f	g	h				
K	i	j	k	l				
L	m	n	o	p				

Figure 13 Labelling of samples

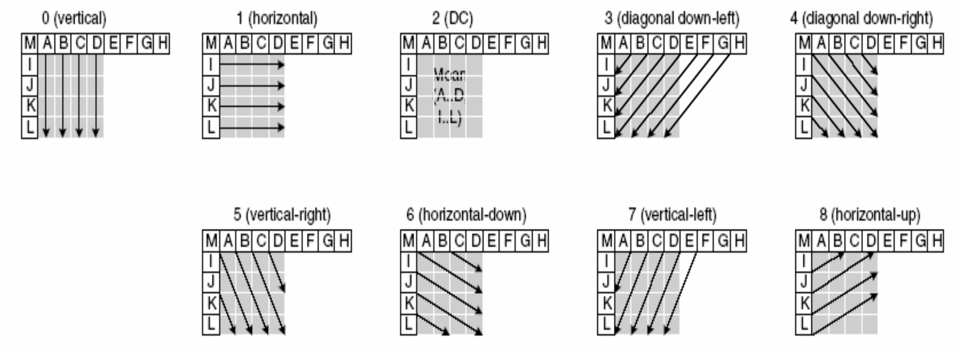


Figure 14 4x4 luma Intra prediction modes in H.264

The arrows in Figure 14 indicate the direction of prediction in each mode. For modes 3–8, the predicted samples are formed from a *weighted average* of the prediction samples A–M. For example, if mode 4 is selected, the top-right sample of P, labelled 'd', is predicted by: round(B/4+ C/2+ D/4).

Figure 15 shows the prediction block P for the 4x4 block of Figure 12, created by each of the nine prediction modes. The Sum of Absolute Differences (SAD) (or Sum of Absolute Errors - SAE) for each prediction indicates the magnitude of the prediction error. In this case, the best match to the actual current block is given by mode 7 (vertical-right) because this mode gives the smallest SAE; a visual comparison shows that the P block appears quite similar to the original 4x4 block.

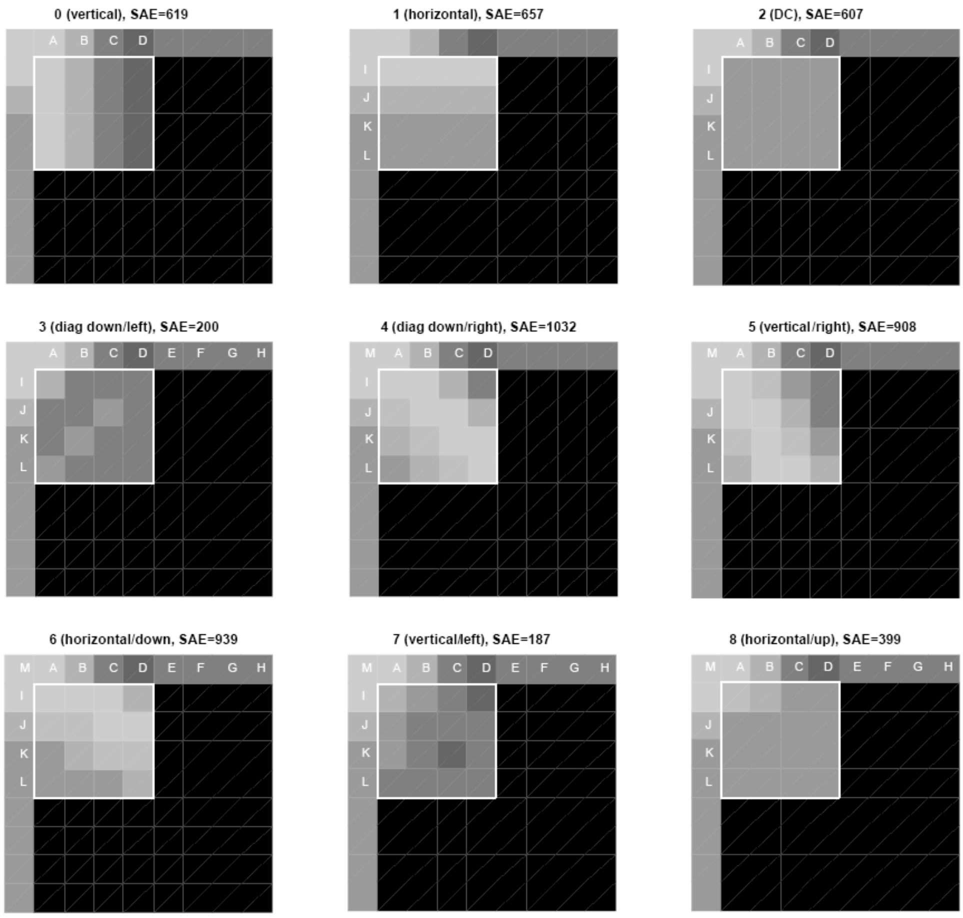


Figure 15 Prediction blocks

As an alternative to the 4x4 luma modes, H.264 specifies that the entire 16x16 luma component of a macroblock may be predicted in one operation. The choice of 16x16 Intra prediction works well in areas of smoothly-varying luminance. Four modes are available, as shown in Figure 16:

- Mode 0 (vertical) Extrapolation from upper samples (H)
- Mode 1 (horizontal) Extrapolation from left samples (V)
- Mode 2 (DC) Mean of upper and left-hand samples (H + V).

– Mode 4 (Plane) A linear ‘plane’ function is fitted to the upper and left-hand samples H and V.

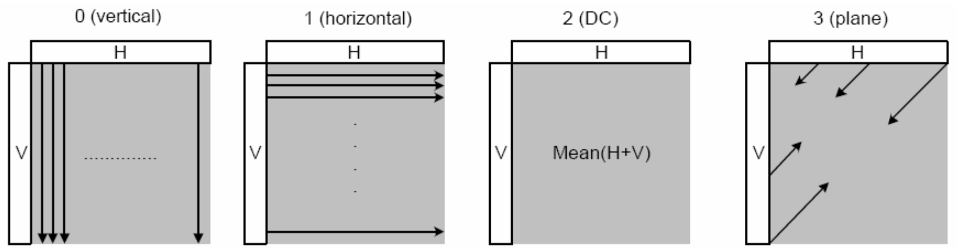


Figure 16 16x16 luma Intra prediction modes in H.264

Each 8x8 chroma component of a macroblock is predicted from chroma samples above and/or to the left that have previously been encoded and reconstructed. The four prediction modes are very similar to the 16x16 luma prediction modes described above, except that the order of mode numbers is different: DC (mode 0), horizontal (mode 1), vertical (mode 2) and plane (mode 3). The same prediction mode is always applied to both chroma blocks.

The choice of intra prediction mode for each 4x4 block must be signalled to the decoder and this could potentially require a large number of bits. However, intra modes for neighbouring 4x4 blocks are often correlated. For example, let A, B and E be the left, upper and current 4x4 blocks respectively, as shown in Figure 17. If previously-encoded 4x4 blocks A and B are predicted using mode 1, it is probable that the best mode for block E (current block) is also mode 1. To take advantage of this correlation, predictive coding is used to signal 4x4 intra modes. For each current block E, the encoder and decoder calculate the most probable prediction mode, the minimum of the prediction modes of A and B. If either of these neighbouring blocks is not available (outside the current slice or not coded in Intra4x4 mode), the corresponding value of A or B is set to 2 (DC prediction mode).

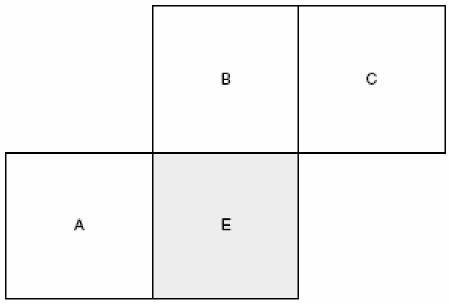


Figure 17 Current and neighbouring block

2.3.2. *Inter Prediction*

In Sections 1.2 and 2.2.1 we have analyzed temporal prediction in a video encoder and seen many aspects concerning motion compensated prediction. In this Section we are going to see how H.264 uses most of these aspects in Inter Prediction, in order to achieve higher coding efficiency.

As we have already seen, there are two kinds of frames that use Inter Prediction to form a prediction frame, P frames, where the prediction is formed by previous frame(s), already encoded and transmitted, and B frames, where the prediction is formed by previous and future frames, already encoded and transmitted. In H.264 the concept of B slices (i.e. sets of Macroblocks – from 1 to the total number of Macroblocks in the frame – that form regions of the frame that can be decoded independently) has been extended as analysed in [48], based on previous research results concerning motion compensated prediction [32], [33], [34].

Another characteristic of Inter Prediction in H.264, is that it supports multiple – picture motion compensated prediction [30], [31]. In other words, in H.264 more than one previously decoded and transmitted frames can be used as reference for the prediction of a frame, as shown in Figure 18. Furthermore, H.264 supports weighted prediction in P and B slices. Up to now, when two frames were used to predict another, the prediction was done with a simple average of the two prediction frames. An H.264 encoder can specify scaling weights and offsets to be used for each Macroblock (P and B).

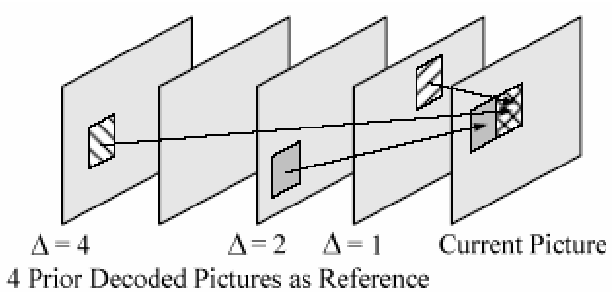


Figure 18 Multiple - picture motion compensated prediction

A practical and widely-used method of motion estimation and compensation is to estimate and compensate for movement of rectangular sections or 'blocks' of the current frame. This process is known as Variable Block Size Motion Estimation (VBSME). H.264 is a block-based motion-compensated hybrid transform codec, which supports a variety of block sizes (denoted as modes), varying from 16×16, 8×16, 16×8, 8×8, 8×4, 4×8 to 4×4 pixels as shown in Figure 19. H.264 supports nine main different modes in Inter Prediction

- Mode 0 (Copy) The Macroblock is transmitted as is, without any prediction.
- Mode 1 (16x16) The Macroblock was Inter Predicted and the motion estimation and compensation process was performed in the entire Macroblock.
- Mode 2 (16x8) The Macroblock was partitioned into two blocks of 16x8 size, each of which was Inter Predicted. The motion estimation and compensation process was performed separately for each block.

– Mode 3 (8x16) The Macroblock was partitioned into two blocks of 8x16 size, each of which was Inter Predicted. The motion estimation and compensation process was performed separately for each block.

– Mode 4 (8x8) The Macroblock was partitioned into four blocks of 8x8 size, each of which was Inter Predicted. The motion estimation and compensation process was performed separately for each block.

– Mode 5 (Intra 4x4) The Macroblock was predicted using Intra 4x4 Prediction.

– Mode 6 (Intra 8x8) The Macroblock was predicted using Intra 8x8 Prediction.

– Mode 7 (Intra 16x16) The Macroblock was predicted using Intra 16x16 Prediction.

– Mode Intra I-PCM The Macroblock is transmitted directly, without prediction and transformation.

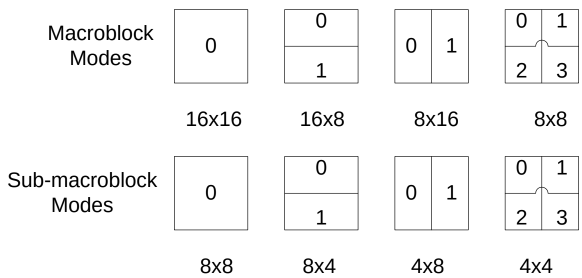


Figure 19 Macroblock and Sub-Macroblock partitions in H.264

In the case of Mode 4, when the Macroblock is divided in four 8x8 blocks (named sub-macroblocks) the inter prediction process has as follow. Each sub-macroblock can be further divided into four ways as shown in Figure 19. Motion Estimation is performed for each of the blocks in the sub-block separately. As Mode 4 is selected the block-size partitioning which provide the best match. Therefore as Mode 4 (8x8) can be chosen any of the 8x8, 8x4, 4x8 or 4x4 modes. In this case, the actual size partitioning is send, along with the best prediction(s) and the corresponding motion vector.

From the previous analysis it is clear that for each block in each block size partition of the Macroblock, motion estimation and compensation is performed and produces a predictor and a corresponding motion vector. Therefore, in H.264 we have a total of 41 MVs and best predictors for each Macroblock. The process of selecting the best among the available predictors for each mode is known an Mode Selection.

In motion estimation a search area is defined, in which the motion estimation algorithm searches to find the best predictor. There are numerous motion estimation algorithm that lie in two main categories, Full Motion Estimation and Fast Motion Estimation. In the first case, all possible candidates (one for each position in the search area) are examined. In Fast Motion Estimation, algorithms find –using various algorithms- the most probable candidates and examine them. In any case, it is necessary to decide which is the best among all the available predictors. Usually the criterion to find the matching block is the energy in the residual formed by subtracting the candidate block from the current M×N block, and the candidate region that minimizes the residual energy is chosen as the best match. However, in order to reduce the computational complexity, most real world applications, among them H.264, use the sum of absolute differences (SAD). Furthermore the H.264 reference software, when calculating the motion cost, takes also into account the bit-rate cost for the motion vector. So in H.264 the criterion for finding the matching block is the motion cost, which is given by the following equation

$$J(\vec{m}, \lambda) = SAD\,(c, r(\vec{m})) + \lambda \times R(\vec{m} - \vec{p}) \qquad (1)$$

where $\vec{m} = (mv_x, mv_y)$ is the current candidate motion vector, $SAD\,(c, r(\vec{m}))$ is the Sum of Absolute Difference between current block and the candidate reference block, $\vec{p} = (mvp_x, mvp_y)$ is the predicted motion vector, $R(\vec{m} - \vec{p})$ is the number of bits needed to code the motion vector difference, and λ is the Lagrangian multiplier.

The use of bit-rate cost in the criterion for motion estimation indicates that H.264 attempts to optimize the inter prediction to achieve the best possible tradeoff between bit - rate (compression) and distortion (quality). This optimization problem is a fertile research area, where a lot of work has been done over the past years and still goes on. Some of the work is focused only on Lagrangian optimization methods [49]-[51], others develop optimized video encoders but with little regard in the complexity of the produced encoding process [52]-[62], others take into account the complexity as well, and so on.

3. THE NEW ALGORITHM

3.1. Introduction

In the previous chapter we have seen some of the basic concepts of video coding, and especially those concerning the encoding process. Furthermore, we showed how these features are integrated in the latest video coding standard, H.264. As it is derived from the above analysis one of the major concepts in the encoding process is the Macroblock Prediction, either Intra or Inter.

Macroblock Prediction is the process where the best candidate, among all the available predictors, must be found, in order to be subtracted by the source Macroblock and create the residual Macroblock, which is the one that is transformed, quantized and entropy encoded. In all video encoders, the Macroblock Prediction process, i.e. inter and intra prediction, is the most complex and computational expensive. Therefore, the criterion of selection used in this process is of great importance, as it plays a decisive role in the complexity of the overall prediction process.

There is a great debate over which is the selection criterion in video coding. Nonetheless, most of them are a representation or variant of a norm. In this chapter we are going to see some basic concepts from the theory of norms and how norms are used as basis for some of the most popular matching criteria. We proceed with the presentation of the Sum of Absolute Differences (SAD), which is the metric of quality in H.264. Finally we introduce a new metric and the algorithm that produces it, which can replace SAD in prediction process, reducing significantly the prediction process, especially in the hardware level.

3.2. Theory of norms

In linear algebra, functional analysis and related areas of mathematics, a norm is a function which assigns a positive *length* or *size* to all vectors in a vector space, other than the zero vector. A seminorm (or pseudonorm) on the other hand is allowed to assign zero length to some non-zero vectors.

Given a vector space V over a subfield F of the complex numbers such as the complex numbers themselves or the real or rational numbers, a seminorm on V is a function $p:V\rightarrow\mathbf{R}$; $x\rightarrow p(x)$ with the following properties:

For all a in F and all **u** and **v** in V,

$p(a\,\mathbf{v}) = |a|\,p(\mathbf{v})$, (*positive homogeneity* or *positive scalability*)

$p(\mathbf{u} + \mathbf{v}) \leq p(\mathbf{u}) + p(\mathbf{v})$ (*triangle inequality* or *subadditivity*).

A simple consequence of these two axioms, positive homogeneity and the triangle inequality, is $p(\mathbf{0}) = 0$ and thus

$p(\mathbf{v}) \geq 0$ (*positivity*).

A norm is a *seminorm* with the additional property

$p(\mathbf{v}) = 0$ if and only if **v** is the zero vector (*positive definiteness*).

A norm is usually denoted $||\mathbf{v}||$, and sometimes $|\mathbf{v}|$, instead of $p(\mathbf{v})$.

Although every vector space is seminormed (e.g., with the trivial seminorm in the Examples section below), it may not be normed. Any vector space V with seminorm $p(\mathbf{v})$ can be made into a normed space by forming the quotient space V/W where W is the subspace of V consisting of all vectors **v** in V with $p(\mathbf{v}) = 0$. The induced norm on V/W is given by $||W+\mathbf{v}|| = p(\mathbf{v})$ and is clearly well-defined [63].

L_1-distance is a positive-definite metric defined over vectors in k-dimensional vector spaces by the corresponding L_1 norm of the difference vector, as follow:

Let $\underline{x}, \underline{y} \in R^k$. Then,

$$L_1(\underline{x}, \underline{y}) = \sum_{i=1}^{k} | x_i - y_i | \qquad (2)$$

It is easy to show the following properties of the L_1 norm

1. $L_1(x,y) = L_1(y,x)$ (symmetric)

2. $L_1(x,y) \geq 0$, with $L_1(x,y) = 0 \Leftrightarrow x=y$ (positive-definite)

3. $L_1(x,y) + L_1(y,z) \geq L_1(x,z)$ (Triangle inequality)

In addition to the L_1-norm, there are other metrics used in vector spaces. The well-known Euclidean distance is perhaps the most popular, also known as L_2-norm, defined as

$$L_2(\underline{x}, \underline{y}) = \left(\sum_{i=1}^{k} | x_i - y_i |^2 \right)^{1/2} \quad (3)$$

It is easy to show that L_2 has the same properties (1)-(3) as L_1.

We can consider the n^{th}-norm of two vectors as:

$$L_n(\underline{x}, \underline{y}) = \left(\sum_{i=1}^{k} | x_i - y_i |^n \right)^{1/n} \quad (4)$$

and show that it has the same properties (1)-(3) above.

The limit case ($n \to \infty$) known as L_∞ can be shown to be

$$L_\infty(\underline{x}, \underline{y}) = \lim_{n \to \infty} L_n(\underline{x}, \underline{y}) = \lim_{n \to \infty} \left(\sum_{i=1}^{k} | x_i - y_i |^n \right)^{1/n} \Rightarrow$$

$$L_\infty(\underline{x}, \underline{y}) = \max_{i \in \{1, \cdots k\}} | x_i - y_i | \quad (5)$$

3.3. The Sum of Absolute Differences (SAD)

In Section 3.2 we have seen that Ln-norms try to quantify in a single number the amount of difference between two vectors. There is great debate over which norm is the best to use in order to express error in signal processing, and especially audio, image and video [64]. Traditionally, researchers use the L_2-norm as the minimization criterion for improving signal processing and compression. That is the main reason the peak signal-to-noise ratio (PSNR) has been used throughout the signal processing literature to express signal quality.

As noted, there is no universally accepted measure for signal quality. One measure that is often cited is the signal – to – noise ratio (SNR), which can be expressed as

$$SNR = 10\log_{10}\frac{encoder\_input\_signal\_energy}{noise\_signal\_energy} \qquad (6)$$

The noise_signal_energy is defined as the energy measured for a hypothetical signal that is the difference between the encoder input signal and the decoder output signal [65]. In the cases of images or video, where the signal is an image/frame, PSNR is used instead of SNR.

Assume we are given a source image f that contains M x N pixels and a reconstructed image F where F is reconstructed by decoding the encoded version of f. Error metrics are computed on the luminance signal only so the pixel values f(i,j) range between black (0) and white (255) [66], [67].

First the mean squared error (MSE) of the reconstructed image is computed as follows

$$MSE = \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}[f(i,j) - F(i,j)]^2 \qquad (7)$$

where f(i,j) and F(i,j) are the pixels of the (i,j) position of the corresponding images. The summation is over all pixels. The root mean squared error (RMSE) is the square root of MSE, i.e.

$$RMSE = \frac{1}{(MN)^{1/2}} \left(\sum_{i=0}^{M-1} \sum_{j=0}^{N-1} [f(i,j) - F(i,j)]^2 \right)^{1/2} \quad (8)$$

PSNR in decibels (dB) is computed by using

$$PSNR = 20 \log_{10} \left(\frac{255}{RMSE} \right) \quad (9)$$

As it is clear from equation 8, the second factor of RMSE is the L_2-norm, so PSNR is a logarithmic representation of the L_2-norm.

On the other hand, Sum of Absolute Differences (SAD), which, as it will be shown, is the L_1-norm, has been used as the basic computational block to find block matches in video compression, since it does not require the additional complexity of the multiplier needed for L_2-norms. This is a necessary compromise – one of many one needs to make – in order to have a practical implementation of a video encoder.

Assume we are given a source image f that contains M x N pixels and a reconstructed image F, where F is reconstructed by decoding the encoded version of f. SAD is the sum over all the absolute differences between the corresponding pixels of source image f and those of reconstructed image F, therefore it can be thought as a metric for the similarity among the two images, as it measures how much different they are.

The Sum of Absolute Differences (SAD) of the reconstructed image is computed as follow

$$SAD = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \left| f(i,j) - F(i,j) \right| \quad (10)$$

where f(i,j) and F(i,j) are the pixels of the (i,j) position of the corresponding images.

Comparing equations 2 and 10 results that SAD is equivalent to the L_1-norm.

3.4. The proposed algorithm

3.4.1. A first approach

In this section we introduce a new algorithm for approaching the problem of selecting the best among various prediction block candidates. The base of the new algorithm is to avoid the stage of addition, which increases significantly the power and delay cost at the hardware level. In video encoding there are two criteria to decide whether a prediction block is better than an other. The first one is how similar is each candidate with the original block being encoded. The second is a combination of similarity and minimization of the bit-rate cost.

An encoder forms, for each block, several predictors and selects the best of them to encode the source block. The more similar is the predictor to the source, the better it is. So, the encoder has to find the predictor which is more similar to the source block. In video encoding, SAD has been established as the most common metric for the similarity among two blocks. The smaller the SAD is, the more similar the two blocks are. Therefore, in the case where the criterion is just similarity, the encoder computes the SAD for all prediction blocks and selects, as best predictor, the one with the smallest SAD.

Clearly, what is of most importance in the above process is to find the predictor which is most similar to the source than the other predictors, and not how much similar it is. Therefore, a qualitative approach may give the same results as a quantitative one.

Assume we have two prediction blocks for a given source block and we have to find which of the two is the most similar to the source. As already analyzed, the absolute difference of a pixel of each candidate and the corresponding pixel of the source indicates how similar the two pixels are. Thus, we compute the absolute differences of the pixels of the two candidates. If the absolute difference of pixel (i,j) of the first candidate is smaller than the absolute difference of pixel (i,j) of the second candidate, then the pixel (i,j) of the first prediction block is more similar to the pixel (i,j) of source. In a qualitative approach, the candidate which has the most similar pixels to the source is the most similar to it.

Based on the above observation, we propose a new algorithm which, after calculating the absolute differences among the predicted and the original pixels, compares these absolute differences for the available modes, instead of adding them. This comparison will conclude to the mode with the most minimum differences. This way the addition stage is completely bypassed.

The similarity criterion is mostly used in Intra Prediction, where H.264 defines that there are a total of 9 optional prediction modes for each 4x4 luma block and 4 optional modes for a 16x16 luma block. For a given 4×4 source block C, according to equation 11, a total of 16 subtractions and 15 additions are needed in order to produce the SAD for the 4x4 block P of one prediction mode.

$$SAD = \sum_{i=0}^{3}\sum_{j=0}^{3}\left|C(i,j) - P(i,j)\right| \qquad (11)$$

Therefore, for the nine modes used in the Intra Prediction Mode, we need 144 subtractions and 135 additions. After computing the SADs for the blocks of all modes, a comparison between the results is made in order to find the mode which produces the smallest SAD.

With the new algorithm, we try to avoid the 135 additions needed by JM. How this is accomplished?

We first calculate the absolute difference between the corresponding pixels for each mode. This can be written as

$$M_{k_{ij}} = | C_{ij} - P_{k_{ij}} | \qquad (12)$$

where M, C, P are 4×4 arrays and k indicates the mode (in the case of the Intra Prediction k is in the range of $0 \le k \le 8$).

Two successive M_k arrays compromise a pair and a comparison among them is done. The array with the largest number of minimum values is selected. In the next step the array selected from each pair compromises a new pair with the array selected from its successive pair and the same procedure is repeated until we end up with just one pair of arrays. The array chosen by the last comparison is the one which corresponds to the best mode.

The comparison among two arrays gives the array with the largest number of minimum values in the following way. Let's assume that we have the following function

$$f_{k_i} = \begin{cases} 1, M_{k_i} \le M_{(k+1)_i} \\ \\ 0, M_{k_i} > M_{(k+1)_i} \end{cases} \tag{13}$$

$$F_k = \sum_{i=0}^{15} f_{k_i} \tag{14}$$

where M_k is the array with the absolute differences for mode k and i (with $0 \le i \le 15$) is the number of absolute differences . According to equation 14 we chose M_k if $F_k > F_{k+1}$, otherwise we chose M_{k+1}.

One may claim that, even with the new algorithm the addition is unavoidable, as according to equation 14, we need to sum the values f_k for each of the 16 pixel of mode k. Nevertheless, this is not the case. For each pixel, f_k can take the values 1 or 0, therefore what is really needed is to count the number of ones presented in f_k. This function can be implemented in hardware by specialised circuits, without using any adders.

3.4.2. The Sum of Greater Values (SGV)

In the previous section we introduced a new algorithm that can replace SAD in the selection of the best between numerous predictors, when the selection criterion is based only on the similarity between predictors and source. Moving on, we extend this algorithm in order to cover the case when the selection criterion is based not only on the similarity but on the bit-rate cost as well.

In video encoding, two are the main goals, quality and compression. The produced video should have quality as close to the original one as possible, and its size should be as smaller as possible. This forces the encoder to take into account both parameters in the decisions it must take. In the Macroblock Prediction process, the decision is which is the best candidate. To preserve the quality, the encoded video should be as similar to the original one as possible, so for this case the criterion is similarity which was analyzed in the previous section. In order to maximize compression, the encoder must select candidates that, after the encoding process, will produce the smallest number of bits possible. This means that for each candidate, the encoder has to have a metric for the produced number of bits. This metric, usually called bit-rate cost, gives the cost in number of bits for each candidate.

There is a tradeoff between quality and compression efficiency, as, in order to produce video with high quality a large number of bits are required. Therefore, high quality leads to lower compression efficiency. The encoder has to find the golden section between quality and compression in order to produce the best result. This means that it has to find the right weights that similarity and bit – rate cost should have in its decisions. This process is called Rate – Distortion (RD) optimization, and is the one used by H.264 in motion estimation.

So, in H.264 the criterion to find the matching block is the motion cost, which is a combination of quality and compression efficiency and is given by the following equation

$$J(\vec{m}, \lambda) = SAD\,(c, r(\vec{m})) + \lambda \times R(\vec{m} - \vec{p}) \qquad (15)$$

where $\vec{m} = (mv_x, mv_y)$ is the current candidate motion vector, $SAD\ (c, r(\vec{m}))$ is the Sum of Absolute Differences between current block and the candidate reference block,

$\vec{p} = (mvp_x, mvp_y)$ is the predicted motion vector, $R(\vec{m} - \vec{p})$ is the number of bits needed to code the motion vector difference, and λ is the Lagrangian multiplier.

According to all these, in this case a pure qualitative approach would not work, as it is necessary to have a measure of how similar the available blocks are, i.e. to have a measure for the quality. Therefore the algorithm introduced in the previous section has to be modified in order to give a metric which could replace SAD in equation 15.

Assume we have two candidate prediction blocks for a given source block and we have to select the best of them. As already analyzed, the absolute difference of a pixel of each candidate and the corresponding pixel of the source indicates how similar the two pixels are. Thus, we compute the absolute differences of the pixels of the two candidates. If the absolute difference of pixel (i,j) of the first candidate is greater than the absolute difference of pixel (i,j) of the second candidate, then the pixel (i,j) of the first candidate block is less similar to the pixel (i,j) of source than the corresponding pixel of the second candidate. The candidate which has smaller number of pixels with greater absolute difference than the other is more similar to the source block therefore it is better. And to be precise the one candidate is worse than the other per N, where N is the difference between the number of pixels with greater absolute difference of the two candidates. Thus, the number of greater absolute differences is a metric for the similarity between the candidates. In the modified algorithm we introduce this metric, called Sum of Greater Values (SGV).

In order to implement the new approach, we first calculate the absolute difference between the corresponding pixels for each candidate reference block. This can be written as:

$$M_{ij} = | C_{ij} - P_{ij} | \qquad (16)$$

where M, C, P are 4×4 arrays, and i, j are the indices specifying the single pixel within the 4×4 array.

So, for each candidate reference block there is an M array. The proposed algorithm compares the values of the corresponding elements of all the available M arrays and chooses the one with the smallest number of maximum values. The metric of the new algorithm is the number of greater values.

Let's suppose we have two candidate reference blocks. That means we have two M array, let's say M_k and M_{k+1}. The value of the new metric for the first candidate is the number of elements of array M_k which are greater than the corresponding elements of M_{k+1}, whereas the for the second candidate is the number of elements of array M_{k+1} which are greater than the corresponding elements of array M_k. This can be written as:

$$f_{k_i} = \begin{cases} 1, M_{k_i} > M_{(k+1)_i} \\ 0, M_{k_i} \le M_{(k+1)_i} \end{cases} \tag{17}$$

$$F_k = \sum_{i=0}^{S} f_{k_i} \tag{18}$$

where M_k and M_{k+1} are the MxN arrays with the absolute differences of the two candidate reference blocks, S is the total number of the elements of each array, i.e. S= MxN and is always equal to the block size. According to equations 17 and 18 the new metric F is a positive number, smaller or equal to S. For example, in the case of 4x4 blocks we have $0 \le F \le 16$.

As we have seen, in H.264, the criterion to find the matching block is to minimize equation 15. With the new algorithm, equation 15 changes slightly as we replace SAD with the new metric. So, according to H.264 and in combination with the proposed algorithm the criterion for the matching block is to minimize the equation

$$J(\vec{m}_1, \vec{m}_2, \lambda) = F(c, r(\vec{m}_1), r(\vec{m}_2)) + \lambda \times R(\vec{m}_1 - \vec{p}) \tag{19}$$

where $F(c, r(\vec{m}_1), r(\vec{m}_2))$ is the new metric for the candidate reference block with corresponding motion vector $\vec{m}_1$ when compared with another candidate reference block with corresponding motion vector $\vec{m}_2$.

4. SOFTWARE IMPLEMENTATION

4.1. Introduction

In the previous chapter we presented a new algorithm that can replace SAD in the Macroblock Prediction process in a hardware video encoder. Before going on with the hardware implementation of this algorithm, it is important to evaluate its performance in the overall context of an H.264 video encoder.

In order to make a reliable evaluation we must use an encoder which is acceptable by the entire video compression community. Unfortunately such an encoder is available only as a software application. Therefore, although the new algorithm is mostly intended for hardware applications, its first evaluation ought to be done in a software context.

The encoder which is generally accepted for the evaluation of every research work in the field of video coding is the official H.264/AVC reference software, also known as JM reference software [68], which is freely available by the International Standards Organization (ISO) and the responsibility for its integration of maintenance is undertaken by HHI [69].

In this chapter we present the JM reference software and how our new algorithm was integrated in it. Furthermore, we present a software version of the proposed algorithm. Finally, we conclude with the results of the performance evaluation of the new algorithm, obtained by the use of the JM reference software as the workspace of the evaluation.

4.2. The JM Reference Software

The JM Reference Software provides both an H.264 encoder and decoder. Nevertheless, the interest of this work focuses only to the encoder. The JM H.264 encoder is a complicated project, created and maintained by a large number of authors, and consists of numerous functions. The complexity of an H.264 encoder makes the JM Reference Software an application which is hard, nonetheless very interesting, to analyse. However, as such an analysis is not the aim of this thesis, we are going to briefly explain the way the JM reference encoder works and see in some detail the process of Macroblock Prediction.

The H.264 standard provides a variety of Levels and Profiles. Each Profile supports a particular set of coding function, while Levels place limits on parameters such as sample processing rate, picture size, coded bitrate and memory requirements. Furthermore, H.264 defines numerous optional functions, also known as tools, which can be enabled by one or more Profiles in order to increase the quality and/or compression efficiency; nonetheless, their presence is not requisite. All these form a set of parameters which are necessary for the determination of the framework in which the encoder will work. JM Reference software uses a configuration file, in which these parameters are presented and set to the appropriate values, and which is read by the encoder before beginning the encoding process. Table 1 shows a list of the parameters presented in the configuration file of the JM reference software (version JM11.0) of the encoder.

Table 1 Parameters in the Configuration File of the encoder of the JM11.0 reference software

# Files	
InputFile	Input sequence
InputHeaderLength	If the inputfile has a header, state it's length in byte here
StartFrame	Start frame for encoding. (0-N)
FramesToBeEncoded	Number of frames to be coded
FrameRate	Frame Rate per second (0.1-100.0)
SourceWidth	Frame width
SourceHeight	Frame height
TraceFile	"src.txt"
ReconFile	"rec.yuv"
OutputFile	"test.264"
# Encoder Control	
ProfileIDC	Profile IDC
LevelIDC	Level IDC (e.g. 20 level 2.0)
IntraPeriod	Period of I-Frames
EnableOpenGOP	Support for open GOPs
IDRIntraEnable	Force IDR Intra (0 disable, 1enable)
QPISlice	Quant. param for I Slices (0-51)

QPPSlice	Quant. param for P Slices (0-51)
FrameSkip	Number of frames to be skipped in input (e.g 2 will code every third frame)
ChromaQPOffset	Chroma QP offset (-51..51)
UseHadamard	Hadamard transform
DisableSubpelME	Disable Subpixel Motion Estimation
SearchRange	Max search range
NumberReferenceFrames	Number of previous frames used for inter motion search (1-16)
PList0References	P slice List 0 reference override (0 disable, N < NumberReferenceFrames)
Log2MaxFNumMinus4	Sets log2_max_frame_num_minus4 (-1 : based on FramesToBeEncoded/Auto, > 0 : Log2MaxFNumMinus4)
Log2MaxPOCLsbMinus4	Sets log2_max_pic_order_cnt_lsb_minus4 (-1 : Auto, > 0 : Log2MaxPOCLsbMinus4)
GenerateMultiplePPS	Transmit multiple parameter sets. Currently parameters basically enable all WP modes
ResendPPS	Resend PPS (with pic_parameter_set_id 0) for every coded Frame/Field pair
MbLineIntraUpdate	Error robustness(extra intra macro block updates)(0 off, N: One GOB every N frames are intra coded)
RandomIntraMBRefresh	Forced intra MBs per picture
InterSearch16x16	Inter block search 16x16
InterSearch16x8	Inter block search 16x8
InterSearch8x16	Inter block search 8x16
InterSearch8x8	Inter block search 8x8
InterSearch8x4	Inter block search 8x4
InterSearch4x8	Inter block search 4x8
InterSearch4x4	Inter block search 4x4
IntraDisableInterOnly	Apply Disabling Intra conditions only to Inter Slices
Intra4x4ParDisable	Disable Vertical & Horizontal 4x4
Intra4x4DiagDisable	Disable Diagonal 45degree 4x4
Intra4x4DirDisable	Disable Other Diagonal 4x4
Intra16x16ParDisable	Disable Vertical & Horizontal 16x16
Intra16x16PlaneDisable	Disable Planar 16x16
ChromaIntraDisable	Disable Intra Chroma modes other than DC
EnableIPCM	Enable IPCM macroblock mode
DisposableP	Enable Disposable P slices in the primary layer
DispPQPOffset	Quantizer offset for disposable P slices

# B Slices	
NumberBFrames	Number of B coded frames inserted
QPBSlice	Quant. param for B slices (0-51)
BRefPicQPOffset	Quantization offset for reference B coded pictures (-51..51)
DirectModeType	Direct Mode Type
DirectInferenceFlag	Direct Inference Flag
BList0References	B slice List 0 reference override
BList1References	B slice List 1 reference override
BReferencePictures	Referenced B coded pictures
HierarchicalCoding	B hierarchical coding
HierarchyLevelQPEnable	Adjust QP based on hierarchy level (in increments of 1). Overrides BRefPicQPOffset behavior.
ExplicitHierarchyFormat	Explicit Enhancement GOP. Format is {FrameDisplay_orderReferenceQP}.
ReferenceReorder	Reorder References according to Poc distance for HierarchicalCoding
PocMemoryManagement	Memory management based on Poc Distances for HierarchicalCoding
BiPredMotionEstimation	Enable Bipredictive based Motion Estimation (0:disabled, 1:enabled)
BiPredMERefinements	Bipredictive ME extra refinements (0: single, N: N extra refinements (1 default)
BiPredMESearchRange	Bipredictive ME Search range (8 default). Note that range is halved for every extra refinement.
BiPredMESubPel	Bipredictive ME Subpixel Consideration (0: disabled, 1: single level, 2: dual level)
# SP Frames	
SPPicturePeriodicity	SP-Picture Periodicity
QPSPSlice	Quant. param of SP-Slices for Prediction Error (0-51)
QPSP2Slice	Quant. param of SP-Slices for Predicted Blocks (0-51)
SI_FRAMES	SI frame encoding flag
SP_output	Controls whether coefficients will be output to encode switching SP frames
SP_output_name	Filename for SP output coefficients
SP2_FRAMES	Switching SP frame encoding flag
SP2_input_name1	Filename for the first swithed bitstream coefficients
SP2_input_name2	Filename for the second switched bitstream coefficients
# Output Control, NALs	

SymbolMode	Symbol mode (Entropy coding method: 0 UVLC, 1 CABAC)
OutFileMode	Output file mode, 0:Annex B, 1:RTP
PartitionMode	Partition Mode, 0: no DP, 1: 3 Partitions per Slice
# CABAC context initialization	
ContextInitMethod	Context init (0: fixed, 1: adaptive)
FixedModelNumber	model number for fixed decision for inter slices (0, 1, or 2)
# Interlace Handling	
PicInterlace	Picture AFF (0: frame coding, 1: field coding, 2:adaptive frame/field coding)
MbInterlace	Macroblock AFF (0: frame coding, 1: field coding, 2:adaptive frame/field coding)
IntraBottom	Force Intra Bottom at GOP Period
# Weighted Prediction	
WeightedPrediction	P picture Weighted Prediction
WeightedBiprediction	B picture Weighted Prediciton
UseWeightedReferenceME	Use weighted reference for ME
# Picture based Multi-pass encoding	
RDPictureDecision	Perform RD optimal decision between different coded picture versions.
	If GenerateMultiplePPS is enabled then this will test different WP methods. Otherwise it will test QP +-1 (0: disabled, 1: enabled)
RDPictureIntra	Perform RD optimal decision also for intra coded pictures.
RDPSliceWeightOnly	Only consider Weighted Prediction for P slices in Picture RD decision.
RDBSliceWeightOnly	Only consider Weighted Prediction for B slices in Picture RD decision.
# Loop filter parameters	
LoopFilterParametersFlag	Configure loop filter
LoopFilterDisable	Disable loop filter in slice header
LoopFilterAlphaC0Offset	Alpha & C0 offset div. 2, {-6, -5, . 0, +1, . +6}
LoopFilterBetaOffset	Beta offset div. 2, {-6, -5, ... 0, +1, .. +6}
# Error Resilience / Slices	
SliceMode	Slice mode
SliceArgument	Slice argument

num_slice_groups_minus1	Number of Slice Groups Minus 1, 0
slice_group_map_type	
slice_group_change_direction_flag	0: box-out clockwise, raster scan or wipe right, 1: box-out counter clockwise, reverse raster scan or wipe left
slice_group_change_rate_minus1	
SliceGroupConfigFileName	Used for slice_group_map_type 0, 2, 6
UseRedundantPicture	0: not used, 1: enabled
NumRedundantHierarchy	0-4
PrimaryGOPLength	GOP length for redundant allocation (1-16)
	NumberReferenceFrames must be no less than PrimaryGOPLength when redundant slice enabled
NumRefPrimary	Actually used number of references for primary slices (1-16)
# Search Range Restriction / RD Optimization	
RestrictSearchRange	restriction for (0: blocks and ref, 1: ref, 2: no restrictions)
RDOptimization	rd-optimized mode decision
	0: RD-off (Low complexity mode)
	1: RD-on (High complexity mode)
	2: RD-on (Fast high complexity mode – not work in FREX Profiles)
	3: with losses
DisableThresholding	Disable Thresholding of Transform Coefficients
DisableBSkipRDO	Disable B Skip Mode consideration from RDO Mode decision (0:off, 1:on)
SkipIntraInInterSlices	Skips Intra mode checking in inter slices if certain mode decisions are satisfied
# Explicit Lambda Usage	
UseExplicitLambdaParams	Use explicit lambda scaling parameters
LambdaWeightIslice	scaling param for I slices. This will be used as a multiplier i.e. lambda LambdaWeightISlice * 2^((QP-12)/3)
LambdaWeightPslice	scaling param for P slices. This will be used as a multiplier i.e. lambda LambdaWeightPSlice * 2^((QP-12)/3)
LambdaWeightBslice	scaling param for B slices. This will be used as a multiplier i.e. lambda LambdaWeightBSlice * 2^((QP-12)/3)
LambdaWeightRefBslice	scaling param for Referenced B slices. This will be used as a multiplier i.e. lambda LambdaWeightRefBSlice * 2^((QP-12)/3)
LambdaWeightSPslice	scaling param for SP slices. This will be

	used as a multiplier i.e. lambda LambdaWeightSPSlice * 2^((QP-12)/3)
LambdaWeightSIslice	scaling param for SI slices. This will be used as a multiplier i.e. lambda LambdaWeightSISlice * 2^((QP-12)/3)
LossRateA	expected packet loss rate of the channel for the first partition, only valid if RDOptimization
LossRateB	expected packet loss rate of the channel for the second partition, only valid if RDOptimization
LossRateC	expected packet loss rate of the channel for the third partition, only valid if RDOptimization
NumberOfDecoders	Numbers of decoders used to simulate the channel, only valid if RDOptimization
RestrictRefFrames	Doesnt allow reference to areas that have been intra updated in a later frame.
# Additional Stuff	
UseConstrainedIntraPred	If 1, Inter pixels are not used for Intra macroblock prediction.
LastFrameNumber	Last frame number that have to be coded (0: no effect)
ChangeQPI	QP (I-slices) for second part of sequence (0-51)
ChangeQPP	QP (P-slices) for second part of sequence (0-51)
ChangeQPB	QP (B-slices) for second part of sequence (0-51)
ChangeQPBSRefOffset	QP offset (stored B-slices) for second part of sequence (-51..51)
ChangeQPStart	Frame no. for second part of sequence
NumberofLeakyBuckets	Number of Leaky Bucket values
LeakyBucketRateFile	File from which encoder derives rate values
LeakyBucketParamFile	File where encoder stores leakybucketparams
NumberFramesInEnhancementLa yerSubSequence	number of frames in the Enhanced Scalability Layer(0: no Enhanced Layer)
NumberOfFrameInSecondIGOP	Number of frames to be coded in the second IGOP
SparePictureOption	0: no spare picture info, 1: spare picture available
SparePictureDetectionThr	Threshold for spare reference pictures detection
SparePicturePercentageThr	Threshold for the spare macroblock percentage
PicOrderCntType	(0: POC mode 0, 1: POC mode 1, 2: POC mode 2)
#Rate control	

RateControlEnable	0 Disable, 1 Enable
Bitrate	Bitrate(bps)
InitialQP	Initial Quantization Parameter for the first I frame. InitialQp depends on two values: Bits Per Picture and the GOP length
BasicUnit	Number of MBs in the basic unit. Should be a fractor of the total number of MBs in a frame
ChannelType	type of channel
#Fast Mode Decision	
EarlySkipEnable	Early skip detection
SelectiveIntraEnable	Selective Intra mode decision
#FREXT stuff	
YUVFormat	YUV format
RGBInput	1 RGB input, 0 GBR or YUV input
BitDepthLuma	Bit Depth for Luminance (8...12 bits)
BitDepthChroma	Bit Depth for Chrominance (8...12 bits)
CbQPOffset	Chroma QP offset for Cb-part (-51..51)
CrQPOffset	Chroma QP offset for Cr-part (-51..51)
Transform8x8Mode	0: only 4x4 transform, 1: allow using 8x8 transform additionally, 2: only 8x8 transform
ResidueTransformFlag	0: no residue color transform 1: apply residue color transform
ReportFrameStats	0:Disable Frame Statistics 1: Enable
DisplayEncParams	0:Disable Display of Encoder Params 1: Enable
Verbose	level of display verboseness
#Q-Matrix (FREXT)	
QmatrixFile	"q_matrix.cfg"
ScalingMatrixPresentFlag	Enable Q_Matrix (0 Not present, 1 Present in SPS, 2 Present in PPS, 3 Present in both SPS & PPS)
ScalingListPresentFlag0	Intra4x4_Luma (0 Not present, 1 Present in SPS, 2 Present in PPS, 3 Present in both SPS & PPS)
ScalingListPresentFlag1	Intra4x4_ChromaU (0 Not present, 1 Present in SPS, 2 Present in PPS, 3 Present in both SPS & PPS)
ScalingListPresentFlag2	Intra4x4_chromaV (0 Not present, 1 Present in SPS, 2 Present in PPS, 3 Present in both SPS & PPS)
ScalingListPresentFlag3	Inter4x4_Luma (0 Not present,

	1 Present in SPS, 2 Present in PPS, 3 Present in both SPS & PPS)
ScalingListPresentFlag4	Inter4x4_ChromaU (0 Not present, 1 Present in SPS, 2 Present in PPS, 3 Present in both SPS & PPS)
ScalingListPresentFlag5	Inter4x4_ChromaV (0 Not present, 1 Present in SPS, 2 Present in PPS, 3 Present in both SPS & PPS)
ScalingListPresentFlag6	Intra8x8_Luma (0 Not present, 1 Present in SPS, 2 Present in PPS, 3 Present in both SPS & PPS)
ScalingListPresentFlag7	Inter8x8_Luma (0 Not present, 1 Present in SPS, 2 Present in PPS, 3 Present in both SPS & PPS)
#Rounding Offset control	
OffsetMatrixPresentFlag	Enable Explicit Offset Quantization Matrices (0: disable 1: enable)
QOffsetMatrixFile	Explicit Quantization Matrices file
AdaptiveRounding	Enable Adaptive Rounding based on JVT-N011 (0: disable, 1: enable)
AdaptRndPeriod	Period in terms of MBs for updating rounding offsets.
AdaptRndChroma	Enables coefficient rounding adaptation for chroma
AdaptRndWFactorIRef	Adaptive Rounding Weight for I/SI slices in reference pictures /4096
AdaptRndWFactorPRef	Adaptive Rounding Weight for P/SP slices in reference pictures /4096
AdaptRndWFactorBRef	Adaptive Rounding Weight for B slices in reference pictures /4096
AdaptRndWFactorINRef	Adaptive Rounding Weight for I/SI slices in non reference pictures /4096
AdaptRndWFactorPNRef	Adaptive Rounding Weight for P/SP slices in non reference pictures /4096
AdaptRndWFactorBNRef	Adaptive Rounding Weight for B slices in non reference pictures /4096
#Lossless Coding (FREXT)	
QPPrimeYZeroTransformBypass Flag	Enable lossless coding when qpprime_y is zero (0 Disabled, 1 Enabled)
#Fast Motion Estimation Control Parameters	
UseFME	Use fast motion estimation
FMEDSR	Use Search Range Prediction. Only for UMHexagonS method
FMEScale	Use Scale_factor for different image sizes.

	Only for UMHexagonS method
EPZSPattern	Select EPZS primary refinement pattern.
EPZSDualRefinement	Enables secondary refinement pattern.
EPZSFixedPredictors	Enables Window based predictors
EPZSTemporal	Enables temporal predictors
EPZSSpatialMem	Enables spatial memory predictors
EPZSMinThresScale	Scaler for EPZS minimum threshold.
EPZSMedThresScale	Scaler for EPZS median threshold.
EPZSMaxThresScale	Scaler for EPZS maximum threshold.

After the configuration file is read and the framework, in which the encoding process will take place, is set, the encoder begins to encode the input video sequence frame by frame. For each frame, the encoding process involves two main tasks, the encoding at picture level, i.e. the encoding of the frame, and the calculation of the distortion produced by the encoding process of the frame. The distortion is calculated for all three components of the picture (Y, U, V) and is expressed as the PSNR for the current frame in units of db. No need to mention that the calculation of the distortion follows the encoding process and it involves pixel-by-pixel comparison of the original frame with respect to the reconstructed (after decoding) frame. Furthermore, for the encoding of the frame to take place, some other functions are necessary to be done, such as memory allocation and others.

In Figure 20, a graphical representation of the above process is presented. This graph, as also the rest call-graphs presented in this chapter, was produced by the documentation of the JM11.0 reference software encoder. In the same way, Figure 21 shows the graph of the process adopted by the encoder in case the video picture is encoded as frame and not as field. It presents the functions involved in the Frame_Picture function, which is the one called by the encoder in order to encode a video picture as frame.

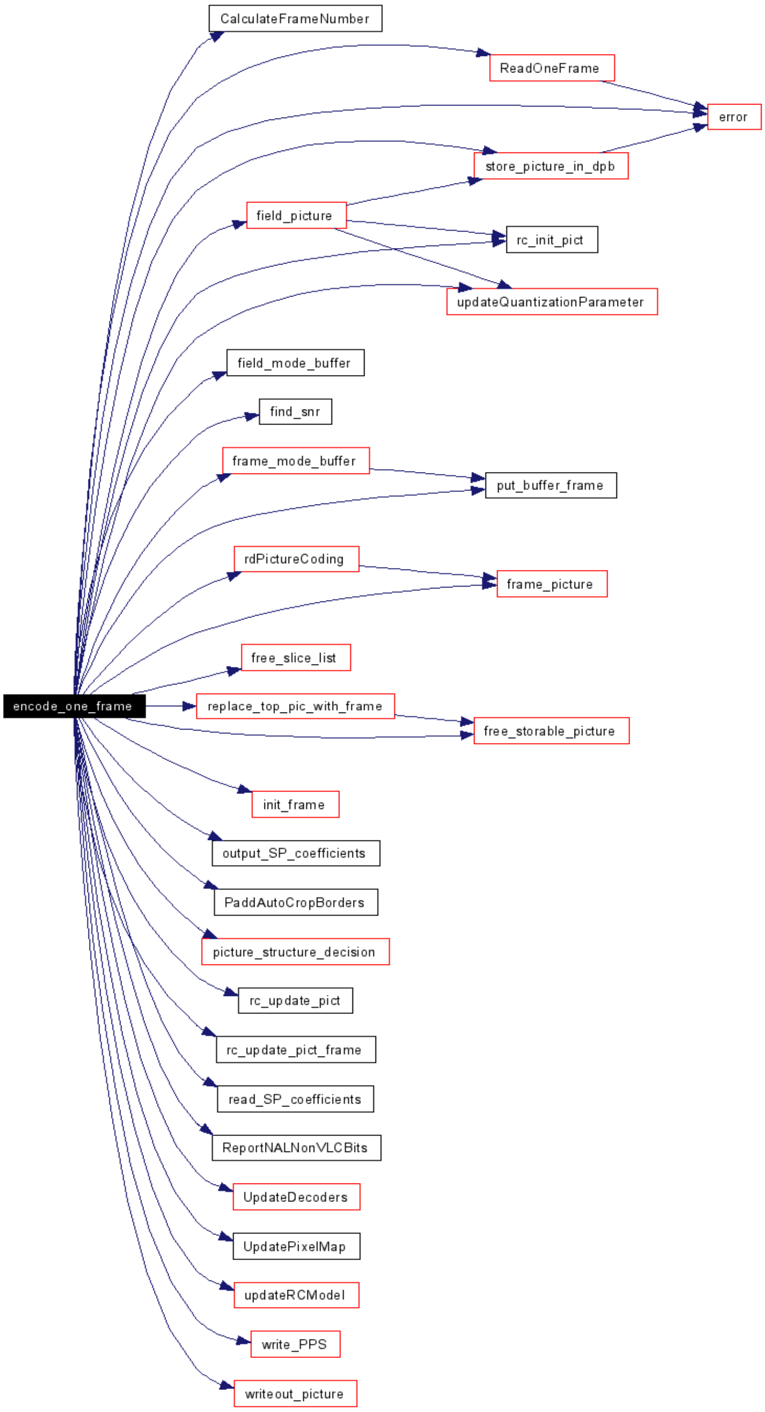


Figure 20 Call Graph of the function "encode_one_frame"

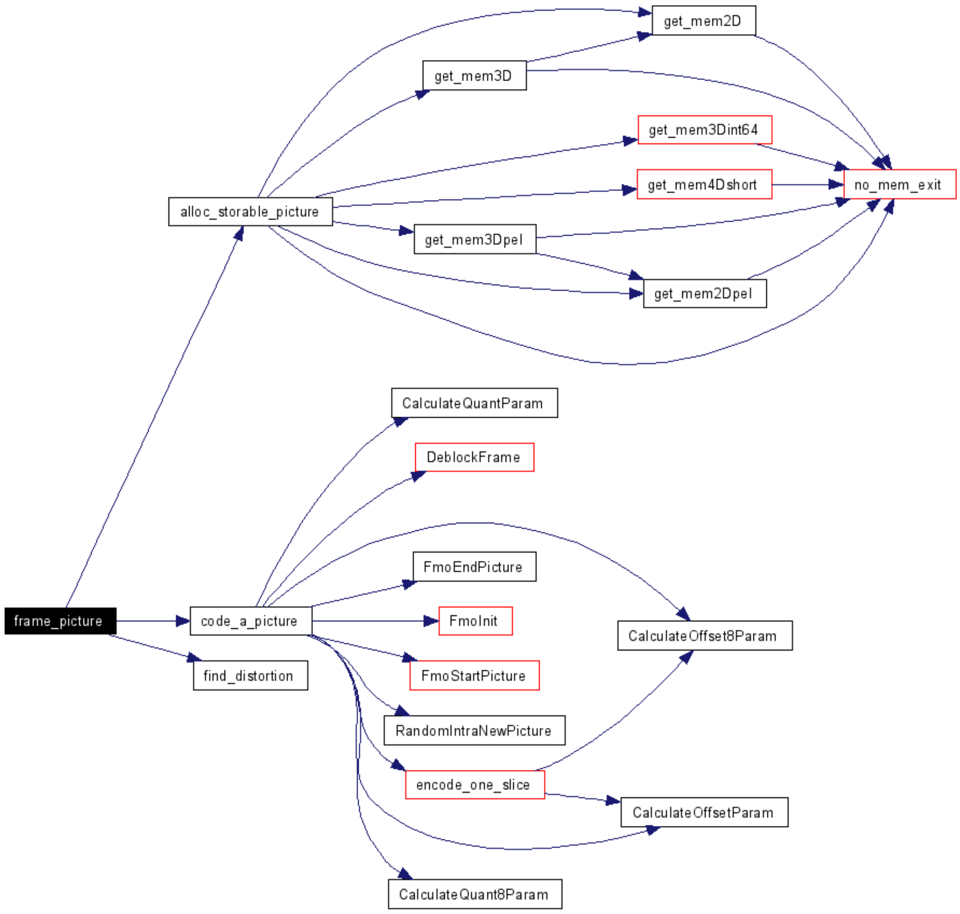


Figure 21 Call Graph of the function "frame_picture"

H.264 defines that a video picture can be coded as one or more slices, each containing an integral number of macroblocks from 1 (1 MB per slice) to the total number of macroblocks in a picture (1 slice per picture). According to this definition, JM, in order to encode a picture, encodes all the slices presented in this picture, as shown in Figure 22. After all the slices in the picture have been encoded, the picture passes through a filter which reduces the blocking artefacts at the transform block (typically 4x4) level. This process is known as deblocking filter, and is one of the innovations presented in H.264. At this stage, JM filters all the Macroblocks of the frame.

The encoding process of a slice comprises of the encoding of each Macroblock presented in the slice and the storage of each encoded Macroblock. In order to perform the above process JM uses the function shown in the call graph of the function "encode_one_slice", shown in Figure 22.

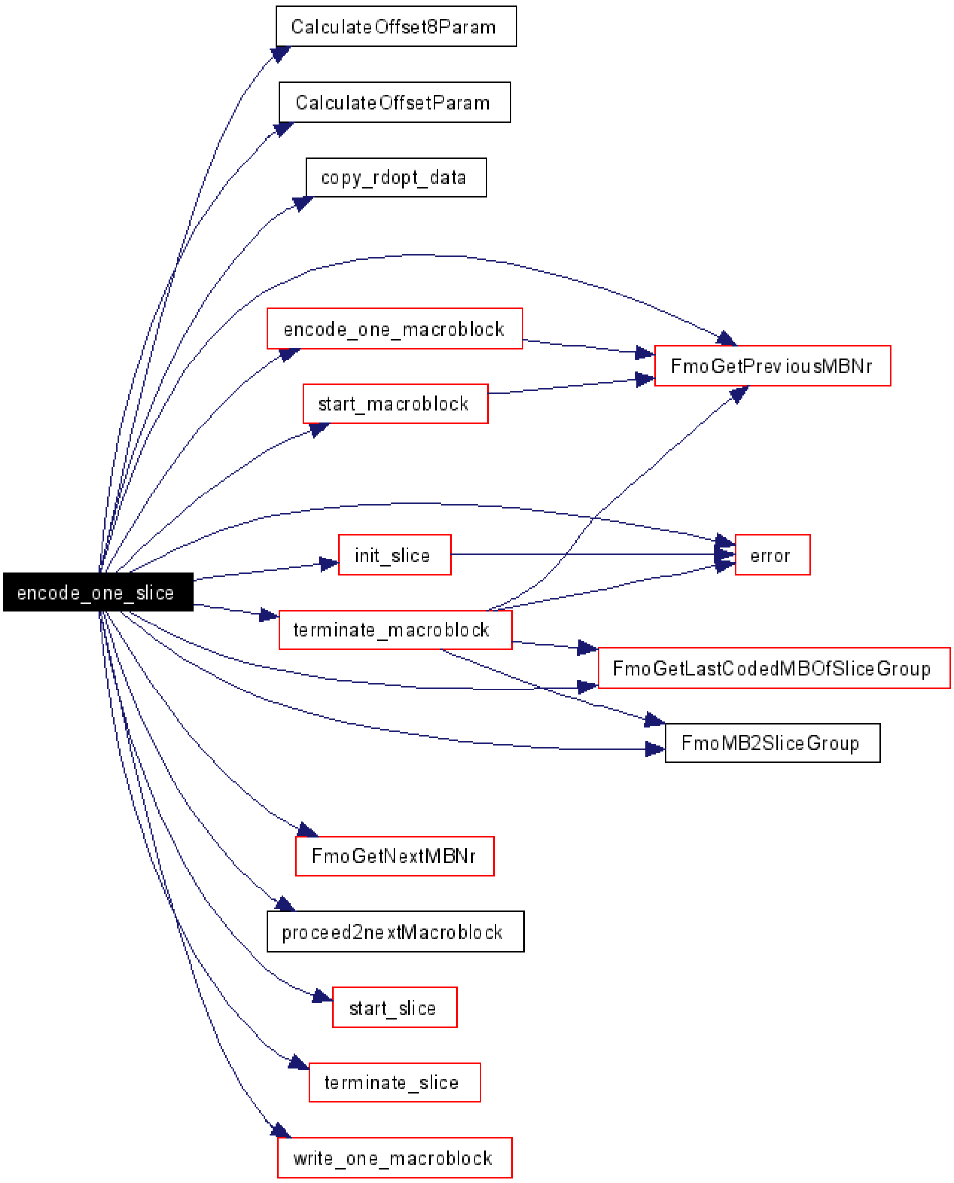


Figure 22 Call Graph of the function "encode_one_slice"

The core of the encoding process can be thought to be the encoding of one Macroblock and was analyzed in detail in Chapter 3. Nonetheless we are going to summarize it here, as this is implemented by JM Reference Software in the "encode_one_macroblock" function, which is the point where the integration of the proposed algorithm begins. This is the level where the majority of the functions defined in H.264 take place. At first, the best predictor for the macroblock being encoded is found. This means that, in general, both Intra and Inter Prediction are performed. The best candidates for each mode of prediction are compared in order to find the best mode of prediction. This involves the mode decision process, in which several aspects may be taken into account, such as the use of rate-distortion optimization, support for variable block size encoding and others. Variable Block Size Motion Estimation (VBSME) is a new coding technique, presented in H.264, and provides more accurate predictions compared to traditional fixed block size motion estimation used by previous standards. H.264 allows for each macroblock (16x16 samples) to be split in four different ways, i.e. any one of a single 16x16 macroblock partition, two 16x8 partitions, two 8x16 partitions or four 8x8 partitions. If the 8x8 mode is chosen, each of the four 8x8 sub-macroblocks within the macroblock may be split in a further 4 ways, i.e. any one of a single 8x8 sub-macroblock partition, two 8x4 sub-macroblock partitions, two 4x8 sub-macroblock partitions or four 4x4 sub-macroblock partitions. In the case of Intra Prediction the available block sizes are 4x4 and 16x16. Please note that the H.264 standard, as amended on March 2005 allows for an additional Intra Prediction mode of 8x8, but only for High-Profiles. That is outside the scope of this work, which has targeted primarily Baseline and Main profile encoding, but it should not affect the generality of our work. After the best predictor is found, the residual formed by subtracting the best predictor from the current macroblock is transformed and quantized. Finally the quantized coefficients are encoded using either CAVLC or CABAC entropy encoding. The way JM implements the macroblock encoding process can be shown by the call graph presented in Figure 23, and is the part of the reference software which we are going to analyse in more detail, as it is the one where the new algorithm will be integrated.

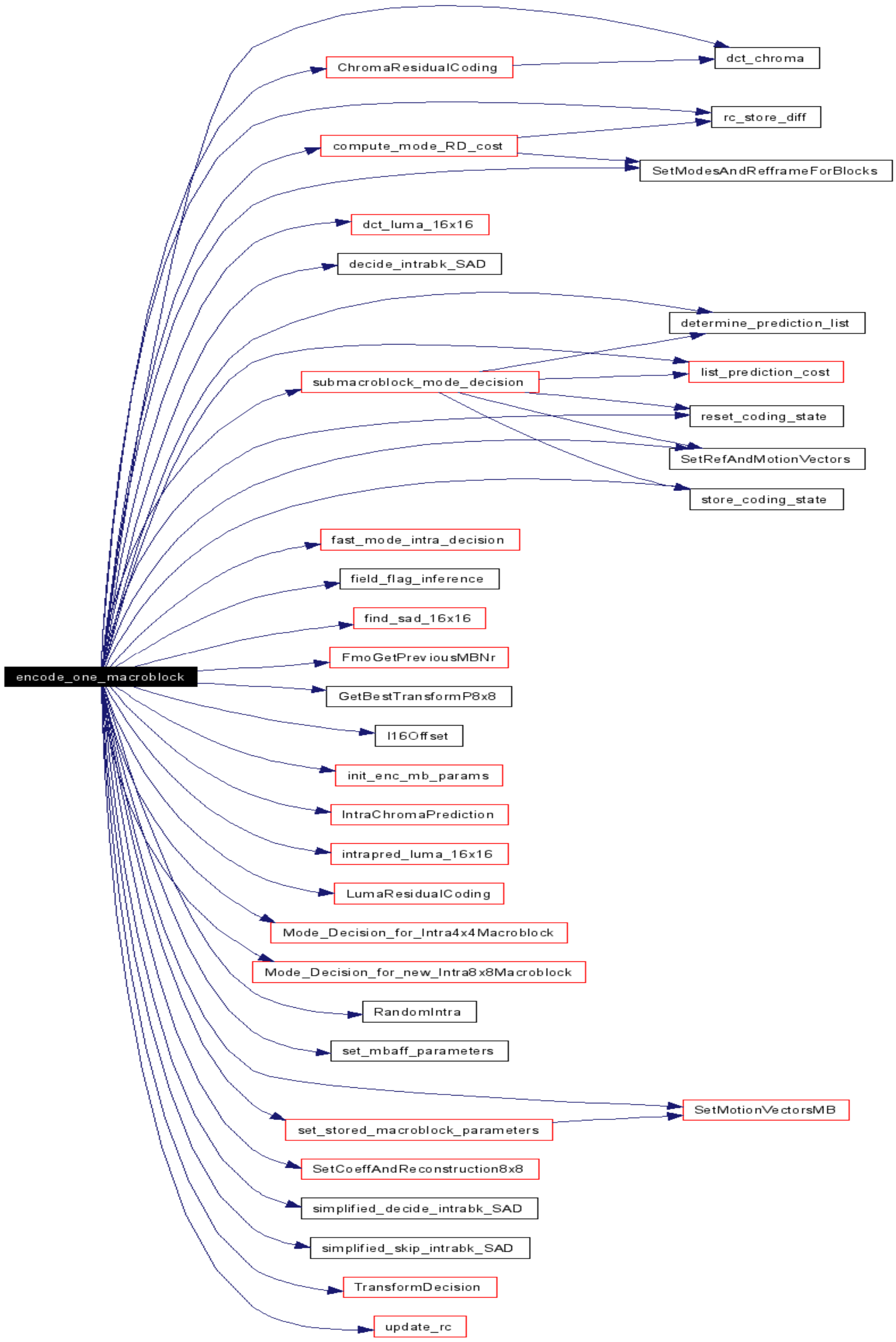


Figure 23 Call Graph of the function “encode_one_macroblock”

The basic idea of the encoding of a macroblock is to find the best predictor among the available candidates, and then perform transform and quantization. For the JM reference software the basic criterion for choosing the best candidate is to minimize a variable named "cost". The calculation of cost varies among the different prediction modes and the rate–distortion (rd) optimization conditions. In the analysis that follows we suppose that no rd–optimization is used. After initializing the macroblock encoding parameters, which determine the flow of the encoding process, JM begins to calculate the cost. If the macroblock is going to be encoded in Inter Mode, JM calculates the cost for the available block-sizes, starting with the biggest one, i.e. 16x16 and continues with the smaller ones, down to the 8x8 mode. For the modes 8x8, 8x4, 4x8 and 4x4, JM calculates the cost, in the same way as for the larger block sizes, and the best of them is the one representing the P8x8 mode. For each mode, it performs the motion estimation algorithm defined by the encoding parameters. The motion estimation process involves selecting the best position within a search range which is also defined by the encoding parameters. As best position, JM defines the position that minimizes the cost, which is a function of SAD and the motion vector cost, i.e. the bit-rate cost.

In the Full Search Motion Estimation algorithm, JM calculates the cost for each position according to the following algorithm

```
//===== loop over all search positions =====
  for (pos=0; pos<max_pos; pos++)
  {
    //--- set candidate position (absolute position in pel units) --
-
    cand_x = center_x + spiral_search_x[pos];
    cand_y = center_y + spiral_search_y[pos];

    //--- initialize motion cost (cost for motion vector) and check
---
    mcost = MV_COST (lambda_factor, 2, cand_x, cand_y, pred_x,
pred_y);
    if (check_for_00 && cand_x==pic_pix_x && cand_y==pic_pix_y)
    {
      mcost -= WEIGHTED_COST (lambda_factor, 16);
    }
    if (mcost >= min_mcost)   continue;

    //--- add residual cost to motion cost ---
    for (y=0; y<blocksize_y; y++)
    {
```

```
      ref_line  = get_ref_line (blocksize_x, ref_pic, cand_y+y,
cand_x, img_height, img_width);
      orig_line = orig_pic [y];

      for (x4=0; x4<blocksize_x4; x4++)
      {
        mcost += byte_abs[ *orig_line++ - *ref_line++ ];
        mcost += byte_abs[ *orig_line++ - *ref_line++ ];
        mcost += byte_abs[ *orig_line++ - *ref_line++ ];
        mcost += byte_abs[ *orig_line++ - *ref_line++ ];
      }

      if (mcost >= min_mcost)
      {
        break;
      }
    }

    //--- check if motion cost is less than minimum cost ---
    if (mcost < min_mcost)
    {
      best_pos  = pos;
      min_mcost = mcost;
    }
  }
```

Apart from the Full Search Motion Estimation algorithm, JM supports a variety of Fast Motion Estimation algorithms, the most interesting and novel of which is the EPZS algorithm [70],[71]. The basic idea is that there is a set of positions, for which it is expected that they are more probable to be the best position. These positions are called predictors. Using the same criterion as in Full Search, EPZS selects the best predictor. The next step of the algorithm is to check the positions nearby the selected predictor. The EPZS algorithm provides a variety of search patterns, each of which specifies which positions, around the predictor, will be searched. According to the search pattern, which is defined in the configuration file, EPZS, using the same algorithm as the one mentioned above, selects the best of the available positions, i.e. the position which minimizes the cost. The way JM selects the best predictor, in the EPZS algorithm, is shown in the following algorithm. The selection of the best position within the search pattern with center the best predictor, is implemented with the same algorithm, using different values, so it will not be repeated.

```
//! Check all predictors
    for (pos = 0; pos < prednum; pos++)
    {
      mvx = predictor->point[pos].x;
      mvy = predictor->point[pos].y;

      cand_x = pic_pix_x + mvx;
      cand_y = pic_pix_y + mvy;

      //--- set motion cost (cost for motion vector) and check ---
      mcost = MV_COST (lambda_factor, 2, cand_x, cand_y, pred_x,
pred_y);

      if (mcost >= second_mcost) continue;
      get_ref_line = CHECK_RANGE ? FastLineX : UMVLineX;

      mcost = computeSad(cur_pic, blocksize_y,blocksize_x,
        blockshape_x, mcost, second_mcost, cand_x, cand_y);

      //--- check if motion cost is less than minimum cost ---
      if (mcost < min_mcost)
      {
        tempmv_x2 = tempmv_x;
        tempmv_y2 = tempmv_y;
        second_mcost = min_mcost;
        tempmv_x = mvx;
        tempmv_y = mvy;
        min_mcost = mcost;
        checkMedian = TRUE;
      }
      //else if (mcost < second_mcost && (tempmv_x != mvx ||
tempmv_y != mvy))
      else if (mcost < second_mcost)
      {
        tempmv_x2 = mvx;
        tempmv_y2 = mvy;
        second_mcost = mcost;
        checkMedian = TRUE;
      }
       }
```

As one can notice, in both Full Search and EPZS, JM initializes the cost, noted as mcost in the codes presented, with the following value

```
  mcost = MV_COST (lambda_factor, 2, cand_x, cand_y, pred_x,
  pred_y);
```

The function MV_COST, is a mathematical function, created in JM, and is used in order to give the bit-rate cost for the motion vector, defined by the cand_x, cand_y, pred_x, pred_y, which are the coordinates of the block being encoded and of the predictor block, respectively. The definition of MV_COST in JM has as follow:

```
LAMBDA_ACCURACY_BITS = 16
LAMBDA_FACTOR(lambda)=((int)((double)(1<<LAMBDA_ACCURACY_BITS)*lambd
a+0.5))
WEIGHTED_COST(factor,bits)=(((factor)*(bits))>>LAMBDA_ACCURACY_BITS)
MV_COST(f,s,cx,cy,px,py)=(WEIGHTED_COST(f,mvbits[((cx)<<(s))-
px]+mvbits[((cy)<<(s))-py]))
```

It is quite interesting to see how JM calculates, and to be more accurate, how it gives the initial value of the bits which would be needed to encode a motion vector, denoted as mvbits in the above equation. JM, before it begins the encoding process and after it has set the parameters according to the configuration file, initializes the motion search, and among others it initializes the motion vector bits. This is done by the following code fragment:

```
int bits, i, imin, imax, k, l;

int search_range               = input->search_range;
int number_of_subpel_positions = 4 * (2*search_range+3);
int max_mv_bits                = 3 + 2 * (int)ceil
(log(number_of_subpel_positions+1) / log(2) + 1e-10);
//--- init array: motion vector bits ---
mvbits[0] = 1;
for (bits=3; bits<=max_mv_bits; bits+=2)
{
  imax = 1 << (bits >> 1);
  imin = imax >> 1;

  for (i = imin; i < imax; i++)
     mvbits[-i] = mvbits[i] = bits;
}
```

Where the "ceil" function, presented in the definition of the "max_mv_bits" variable, is defined by the H.264 standard as:

Ceil(x)	the smallest integer greater than or equal to x.

As one can see in the above algorithm, JM assumes that motion vectors with the same norm but different direction, i.e. vectors that specify the same distance but have the opposite direction, need the same number of bits to be encoded. Furthermore, the number of bits needed for each vector is proportional to the distance they specify. According to this algorithm the number of bits needed for each vector follows the scheme of the Table 2, where is presented the number of bits needed to encode the motion vectors with measure the absolute of the values presented in the second column.

Table 2 Number of bits needed to encode the motion vectors.

Number of Bits	Index of mvbits Vector
1	0
3	- 1, 1
5	-4, 4, -5, 5, -6, 6, -7, 7
7	-8, 8, -9, 9, -10, 10, -11, 11, -12, 12, -13, 13, -14, 14, -15, 15

With a more careful look, one may notice that the above scheme is equivalent to the one presented in the standard for the Exp-Golomb coding scheme, which is shown in Table 3. This means that JM, in order to calculate the bitrate cost of the motion vectors, supposes that they are encoded using Exp-Golomb, without taking into account the entropy encoding scheme the encoder will finally use, which will be either CAVLC or CABAC. It must be noted here that this approach is correct for CAVLC, since this is what is being used, but for CABAC is an approximation that is quite accurate, since both CAVLC and CABAC are trying to represent the same symbols, as close to its entropy as possible.

Table 3 Bit strings with "prefix" and "suffix" bits and assignment to codeNum ranges (informative)

Bit string form	Range of codeNum
1	0
0 1 x_0	1-2
0 0 1 x_1 x_0	3-6
0 0 0 1 x_2 x_1 x_0	7-14
0 0 0 0 1 x_3 x_2 x_1 x_0	15-30
0 0 0 0 0 1 x_4 x_3 x_2 x_1 x_0	31-62
…	…

The fact that MV_COST is presented in every motion estimation algorithm in JM reveals that in JM, even if the encoder is set not to perform Rate-Distortion Optimization, the latter is always used in the motion estimation process. This means that the parameters in the configuration file, related with R-D optimization, concern other functions of the encoder, such as the mode decision process, and not the motion estimation.

Following the above process, JM selects - for Inter Prediction - the best mode among the 16x16, 8x16 and 16x8 modes. Next, it finds the best mode among the 8x8, 8x4, 4x8 and 4x4 modes. If this best mode is better than the one selected in the first step, then the mode P8x8 is selected for the Inter Prediction case, else the best mode is the first one selected (one of 16x16, 8x16 or 16x8). When JM finishes with the selection of the best inter mode, continues to find the best intra mode. This function is performed if the encoder's parameters allow intra prediction in P frames. Moreover, if the encoder processes an I –frame, then the process of motion estimation and inter mode decision, previously presented, is skipped, and the encoder continues directly to the intra-mode selection.

In the case of intra prediction, JM first calculates the cost for the mode Intra8x8, if the encoder is set to perform 8x8 transform. If not, JM continues directly to calculate the cost of Intra4x4 and Intra16x16 modes. In Intra prediction, the encoder does not have to select between various positions, but between various prediction modes. H.264 defines nine prediction modes for the case of Intra4x4 and four for the case of Intra16x16, as analyzed in Chapter 2.

The function JM uses for the Intra4x4 mode, works as follow. After all the available out of the (maximum) nine prediction modes are computed, it compares the cost produced by each of them. In this case cost consists of the SAD and one other constant value. This value is zero for the most probable mode, and for all the other modes is a fixed number, consistent with the encoding method used for CAVLC encoding for it. The most probable mode is selected using the following algorithm.

For each current block E, the encoder calculates the most probable prediction mode, which is the minimum of the prediction modes of A and B, i.e. the prediction modes selected for the blocks left and upper to the current block, as shown in Figure 24. If either of these neighbouring blocks is not available (outside the current slice or not coded in Intra4X4 mode), the corresponding value A or B is set to 2 (DC prediction mode).

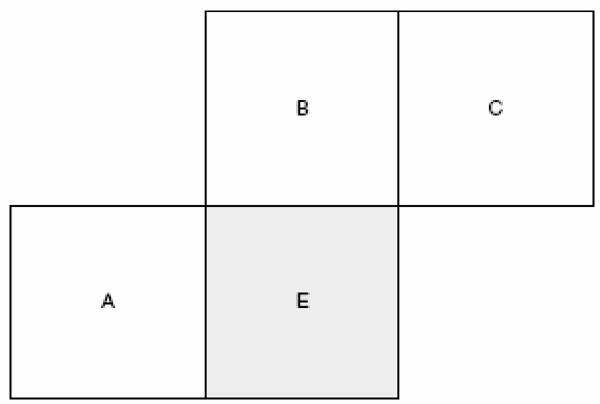


Figure 24 Current and neighbouring blocks (same size)

The same process is followed for the Intra16x16 mode, with the only difference that it is not used a most probable mode, meaning that in this case the cost is equivalent to SAD.

When the process of Intra Prediction is complete, JM has found the best of the available modes, and for that mode the best motion_vector(s) (for Inter Prediction) or the best prediction mode(s) (for Intra Prediction). JM ends up with the best mode by first choosing the best Inter mode (in the case of P-Frames). Then the cost of this mode is compared with the cost produced by the best Intra4x4 prediction mode. The best of them is compared with the cost of the best Intra16x16 prediction, and the one with the smaller cost is the final best mode. Next, JM transforms the best mode, and the coefficients produced by the transform process are quantized. Finally, if the best mode was Inter with block size 16x16, JM examines the case of the COPY mode, where no prediction is performed and the macroblock is used in its original form. This is done by comparing the cost of the COPY mode, which is calculated with a special function to take into account the fact that COPY mode saves even more bits, since there are no motion vectors to be transmitted, with the cost of the P16x16 mode. The mode with the smallest cost is the final best mode.

4.3. Integration of the New Algorithm with the JM Reference Software

In this section we are going to describe the software implementation of the new algorithm, presented and analysed in Chapter 3, and how it was integrated with the JM reference software encoder.

The basic idea of the new algorithm is that we compare the absolute differences of one candidate with the corresponding differences of the next one. For each absolute difference that is greater than the other the SGV (Sum of Grater Values) of the candidate with the greater difference is increased by one. The candidate with the smallest SGV is selected, and is compared with the next candidate. This process is repeated until all candidates are examined and ends up with the candidate with the smallest SGV of all. In the case where the SGV of the two candidates are equal, the first of the two is selected.

SGV is a metric alternative to SAD, introduced in the new algorithm. Nonetheless, as we have seen in the analysis presented in the previous section, JM does not use only SAD as the criterion for selecting the best candidate. In the case of motion estimation, JM takes, also, into account the bit-rate cost of each motion vector, whereas in the case of Intra prediction a fixed cost is added to the SAD of the prediction modes, others than the most probable mode. This fact lead to the extension of the basic idea of the algorithm, so as to include the additional cost in the criterion of selection. Therefore, the new algorithm, after it calculates the SGV of each candidate, adds up the corresponding cost and selectd the candidate with the smallest sum.

The implementation of the new algorithm, as it was finally modulated, may be more easily understood if presented like the pseudo-code that follows. The array *block* contains the absolute differences of all candidates, whereas the array *lambdacost* contains the additional cost of all candidates. The indexes *k1* and *k2,* refer to candidate k1 and candidate k2, respectively.

```
cost1 = lambdacost[k1];
cost2 = lambdacost[k2];

for (in=0; in<block_size; in++){
          m2 = blocks[k2][in];
          m1 = blocks[k1][in];
          if (m1>m2)
            counter1++;
          else
            if (m1<m2)
              counter2++;
 }

if ((cost1 + counter1)<(cost2+counter2))
          return(k1);
else
          return(k2);
```

Although the implementation of the new algorithm is quit trivial, as indicated by the code presented, its integration in the JM's environment proved to be rather complicated, as the whole structure of JM is based on SAD. SGV and SAD are two metrics that may have some similarities, but their philosophy is very different. One of the main differences of the two metrics is that SAD refers to one candidate, whereas SGV refers to one candidate with regard to another. Consequently, while SAD can be calculated for each candidate separately, at the same time when the absolute differences of this candidate are calculated, this is not the case for SGV.

This problem was confronted in the following way. First of all we calculate the absolute differences of all the available candidates. These values are kept in a 2-D array, where the first dimension refers to the candidate, i.e. the position (for the motion estimation) or the prediction mode (for the intra mode), while the second refers to the absolute differences. At the same time two more 1-D arrays are used, one for keeping the position of the current candidate, and another for keeping the additional cost (bit-rate cost or fixed).

The next thing to do is to decide how the available candidates are going to be compared with each other. There are two ways to perform the above action. The first one is to compare the first candidate with its next one. The best of them will be compared with the next, and so on. The other way is to follow a tree-structure comparison, where we take pairs of successive candidates, compare them and the best of each pair, comprise a new pair with the best candidate of the successive pair, as shown in Figure 25. This process is repeated until we end up with one pair. The best candidate of the last pair is the best among all candidates. In this work we decided to follow the tree-structure comparison in the software implementation.

Another crucial aspect that one should take into account in the integration process is the value of the additional cost. As we have seen in the previous section, JM uses the MV_COST function to calculate the bit-rate cost in motion estimation. We remind that MV_COST, has as follow:

mcost = MV_COST (lambda_factor, 2, cand_x, cand_y, pred_x, pred_y)

As one may notice, one of the arguments is the *lambda_factor*. This is a factor used by JM in order to have the appropriate proportion of the values of SAD and the actual bitrate cost. This means that value of *lambda_factor* has immediate association with SAD. So, if one decides to replace SAD should also replace *lambda_factor*.

As a first approach, we decided to use the pure bit-rate cost, as it is calculated by the use of the mv_bits, as it was described in the previous section. This concluded to the following function:

```
MV_COST_PURE(s,cx,cy,px,py)=(mvbits[((cx)<<(s))-
px]+mvbits[((cy)<<(s))-py]))
```

Finding a theoretical way to prove which should be the factor with which the bit-rate would be multiplied in each algorithm, could be an interesting field for research, but it is over the limits of this work.

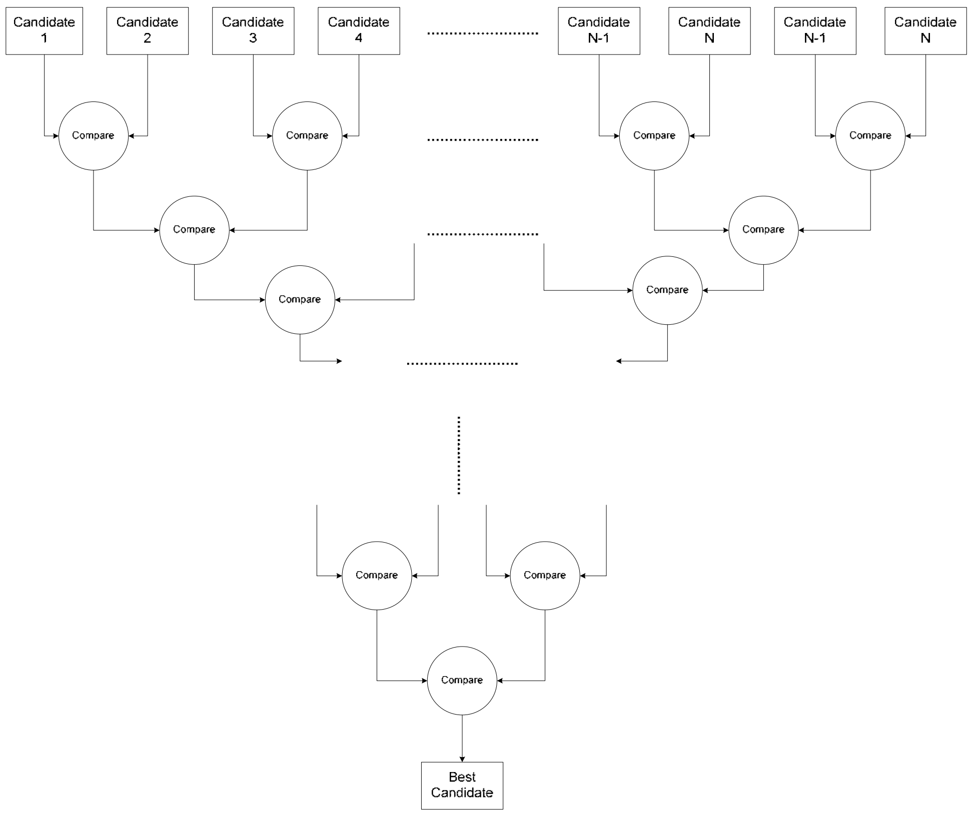


Figure 25 Tree – Structure Comparison

The integration of the new algorithm in the environment of JM required numerous changes in the original code, not only at the programming level but also in the algorithmic one. The major changes in the algorithmic level of JM, were those presented into the pages preceded. As for the programming level, the changes concern purely code writing and is not worth going down to so much detail.

4.4. Simulation Results

After the integration of the new algorithm with the JM reference software encoder has finished and exhaustively tested for its correctness, we proceeded to the evaluation of the new algorithm.

The evaluation was done in two ways. The first was to keep the original algorithms, presented in JM, and let them being the ones deciding the flow of the encoding process, while the new algorithm was working at the same time. In this way we obtained some interesting statistical information. For this method we used the "foreman" QCIF (176x144 @ 15 fps) video sequence. The reason for choosing "Foreman" is because it is one of the most representative test sequences used in research. Furthermore, we noticed that the results of the encoding of "Foreman" are close to the average of the results of the encoding numerous other video sequence.

The encoding of the "Foreman" video sequence was done using full motion estimation, CABAC as the entropy encoding scheme and QP equal to 28. The aim of this encoding was mainly to find the percentage of same decisions made by the two algorithms. This was done by comparing the motion vectors chosen by the original JM and those chosen by the new algorithm. The encoding was done using Variable Block Size Motion Estimation (VBSME), with all the available block sizes enabled. Under these conditions the two algorithms had to find a total of 41 motion vectors for each macroblock.. The precise number of motion vectors (mv) required by each block according to its size, is shown in Table 4.

Table 4 Number of motion vectors needed for each block size.

Block Size	16x16	16x8	8x16	8x8	8x4	4x8	4x4
Number of mv	1	2	2	4	8	8	16

Apart from the percentage of different motion vectors, we tried to find the impact of these different decisions of the two algorithms. The only way to do so was to calculate the SAD produced by the motion vector chosen by JM and the one produced by the motion vector chosen by the new algorithm. The difference of the two SADs in regard with the SAD of the original algorithm may give an idea about the impact of the different choice. This process was done for every macroblock in the video sequence and the average values are presented in Table 5.

Table 5 Simulation Results for "Foreman" QCIF video sequence.

Block Size	# of same mv	% similarity	Difference of SADs	Minimum SAD	% difference of sad
4x4	12,51	78	-134,31	1523,37	8,82
4x8	5,58	70	-115,52	1433,53	8,06
8x4	5,61	70	-113,45	1434,01	7,91
8x8	2,63	70	-97,93	1400,31	6,99
8x16	1,30	60	-91,49	1441,50	6,35
16x8	1,27	60	-114,41	1460,73	7,83
16x16	0,63	63	-136,78	1435,67	9,53
Average % of similarity		67	Average % of difference		7,93

The evaluation showed that in nearly the 70% of circumstances the new algorithm chooses the same motion vectors as SAD. The rest 30% produces an overhead in the SAD which does not exceed 8%.

The second way, for evaluating the new algorithm, was to implement two encoders, one with the new algorithm and the other being the original JM reference software encoder. We used the encoders for encoding a significant number of sequences, under different encoding conditions, and their results were compared in terms of bitrate and PSNR, which are the metrics for the compression efficiency and quality of an encoding process.

The new algorithm was integrated in the entire macroblock prediction process, which means that replaced SAD in Intra and Inter Prediction. Furthermore, Inter Prediction can be implemented with various motion estimation algorithms. We have chosen to use the proposed algorithm in two of them, full motion estimation and EPZS fast motion estimation. Apart from the motion estimation algorithm used, the encoding process is affected by numerous other parameters, some of which are important to the evaluation of every novel algorithm presented in the encoder. Some of them are the QP, i.e. quantization parameter, which controls the compression efficiency of the encoder, the entropy encoding used by the encoder, the selection for performing rate – distortion optimization or not, and many others.

The evaluation of the new algorithm was done separately for the case of full motion estimation and of EPZS. For each one of these evaluations, we encoded video sequences for various QPs and entropy encoding schemes. The rd- optimization parameter, which enables rate – distortion optimization in the mode selection process, was turned off. In this way no rate-distortion optimization is performed (in the mode selection process), which means that the results of the encoding process are mainly effected by the performance of the algorithm used in the macroblock prediction process.

We used 10 sequences from the Video Quality Experts Group (VQEG) [72], src13_ref__720x480_420, src14_ref__720x480_420, src15_ref__720x480_420, src16_ref__720x480_420, src17_ref__720x480_420, src18_ref__720x480_420, src19_ref__720x480_420, src20_ref__720x480_420, src21_ref__720x480_420 and src22_ref__720x480_420 all in 525 SD video format. The video sequences are 720×480 and have frame rate 30 fps. These ten sequences cover the majority of the issues that could be presented in a video, as for example vivid colours, high mobility, different motion directions within the same frame and many others. The total number of frames used in the simulation is 260 for each sequence, with an I-Frame for each 15 frames.

The results that follow concern the evaluation using as motion estimation scheme the "EPZS" scheme with the search range set to 16 and number of reference frames set to 1. The entropy encoding scheme is CAVLC and the quantization parameter (QP) was set to QP= {2, 4, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50} for I and P frames, so as to test the ten sequences from low to high bitrates. First of all, we compared, for every value of the QP parameter, the bitrate results of the two encoders. The results of the comparisons between the bitrates of the two encoders are presented in detail in Appendix A, whereas in this chapter the data selected for the rate – distortion curves and the r-d graphs are presented.

The results presented in Appendix A show that the percentage of the average difference between the filesizes of the video sequences produced by the two encoders does not exceed the 5.5%. More precisely, in the case of CAVLC entropy encoding the maximum percentage of average difference between the filesizes produced by the two encoders is met when QP is 42 and is 5.417%. Therefore, the compression efficiency of the new algorithm is competitive to that of SAD. Despite these positive results, the evaluation could not be completed without presenting the rate – distortion curves for the two encoders. For each of the ten video sequences we present the rate – distortion curves produced by each encoder. The data used for the curves is presented in the corresponding tables, where the average percentages of the difference between the filesizes and between the PSNR, for every sequence at different QPs, are also shown.

Table 6 PSNR and Bitrate for the src13 video sequence

src13	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,42	75091	56,42	74794	0,00	0,40
4	54,45	68477	54,45	68171	0,00	0,45
6	52,5	61198	52,5	60896	0,00	0,50
8	50,91	54071	50,91	53760	0,00	0,58
10	49,47	47703	49,47	47400	0,00	0,64
12	47,55	40432	47,55	40142	0,00	0,72
14	45,94	34181	45,94	33895	0,00	0,84
16	44,23	28339	44,25	27928	-0,05	1,47
18	42,42	22390	42,45	22033	-0,07	1,62
20	40,97	17995	41,01	17630	-0,10	2,07
22	39,69	14692	39,73	14374	-0,10	2,21
24	38,26	11503	38,3	11304	-0,10	1,76
26	37,02	9291	37,05	9098	-0,08	2,12
28	35,78	7511	35,82	7327	-0,11	2,51
30	34,36	6021	34,4	5864	-0,12	2,68
32	33	4732	33,04	4586	-0,12	3,18
34	31,72	3768	31,77	3643	-0,16	3,43
36	30,33	2826	30,38	2715	-0,16	4,09
38	29,02	2169	29,08	2078	-0,21	4,38
40	27,88	1682	27,94	1605	-0,21	4,80
42	26,68	1282	26,74	1224	-0,22	4,74
44	25,54	1027	25,61	982	-0,27	4,58
46	24,61	778	24,69	744	-0,32	4,57
48	23,69	600	23,79	577	-0,42	3,99
50	22,75	508	22,88	495	-0,57	2,63
	Average % difference				-0,14	2,44

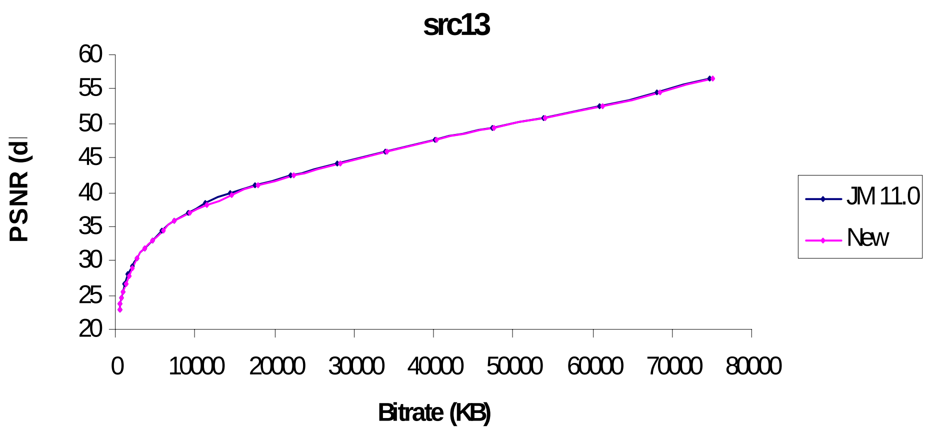


Figure 26 Rate – Distortion curves for src13

Table 7 PSNR and Bitrate for the src14 video sequence

src14	NEW		JM		% difference	
qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,52	55714	56,53	55438	-0,02	0,50
4	54,43	48747	54,43	48438	0,00	0,64
6	52,65	41854	52,66	41537	-0,02	0,76
8	51,06	34526	51,07	34263	-0,02	0,77
10	49,7	28806	49,72	28593	-0,04	0,74
12	47,99	22852	48,03	22705	-0,08	0,65
14	46,57	18462	46,61	18376	-0,09	0,47
16	45,14	14711	45,16	14640	-0,04	0,48
18	43,62	11278	43,63	11206	-0,02	0,64
20	42,29	8906	42,29	8834	0,00	0,82
22	41,05	7073	41,05	7007	0,00	0,94
24	39,59	5277	39,6	5243	-0,03	0,65
26	38,31	4003	38,32	3966	-0,03	0,93
28	37,07	3005	37,08	2968	-0,03	1,25
30	35,67	2174	35,67	2141	0,00	1,54
32	34,39	1528	34,4	1499	-0,03	1,93
34	33,2	1089	33,22	1064	-0,06	2,35
36	31,93	710	31,97	692	-0,13	2,60
38	30,81	496	30,85	478	-0,13	3,77

40	29,81	364	29,85	352	-0,13	3,41
42	28,81	281	28,88	272	-0,24	3,31
44	27,85	231	27,94	224	-0,32	3,13
46	27,02	188	27,11	183	-0,33	2,73
48	26,21	163	26,28	158	-0,27	3,16
50	25,46	149	25,55	146	-0,35	2,05
Average % difference					-0,10	1,61

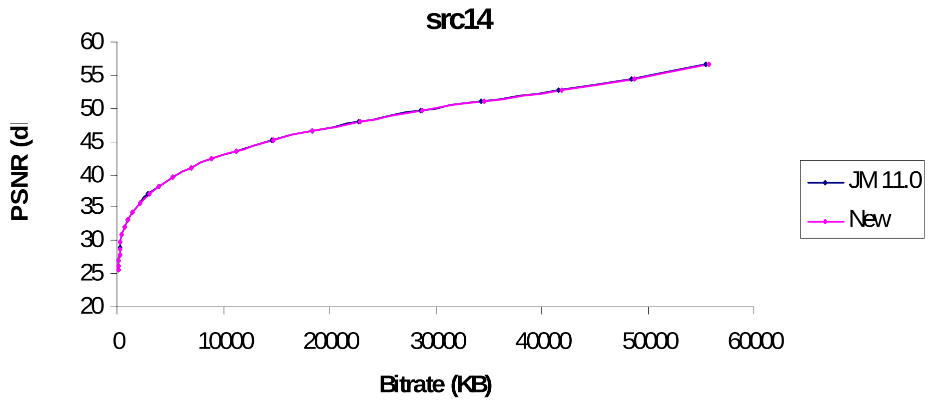


Figure 27 Rate – Distortion curves for src14

Table 8 PSNR and Bitrate for the src15 video sequence

src15	NEW		JM		% difference	
qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,28	85641	56,28	85654	0,00	-0,02
4	54,37	78517	54,37	78533	0,00	-0,02
6	52,34	70857	52,34	70871	0,00	-0,02
8	50,79	63737	50,79	63756	0,00	-0,03
10	49,32	57384	49,32	57408	0,00	-0,04
12	47,3	50294	47,3	50318	0,00	-0,05
14	45,63	44217	45,63	44248	0,00	-0,07
16	43,82	38381	43,83	38334	-0,02	0,12
18	41,78	32169	41,79	32120	-0,02	0,15
20	40,05	27128	40,05	27069	0,00	0,22

22	38,49	22974	38,5	22921	-0,03	0,23
24	36,68	18766	36,69	18726	-0,03	0,21
26	35,14	15588	35,14	15536	0,00	0,33
28	33,59	12877	33,59	12819	0,00	0,45
30	31,85	10330	31,86	10269	-0,03	0,59
32	30,26	8187	30,27	8126	-0,03	0,75
34	28,75	6414	28,76	6357	-0,03	0,90
36	27,07	4646	27,09	4588	-0,07	1,26
38	25,57	3323	25,59	3270	-0,08	1,62
40	24,21	2298	24,23	2246	-0,08	2,32
42	22,82	1467	22,85	1422	-0,13	3,16
44	21,65	986	21,7	945	-0,23	4,34
46	20,69	696	20,74	661	-0,24	5,30
48	19,68	543	19,77	510	-0,46	6,47
50	18,68	521	18,79	483	-0,59	7,87
	Average % difference				-0,08	1,44

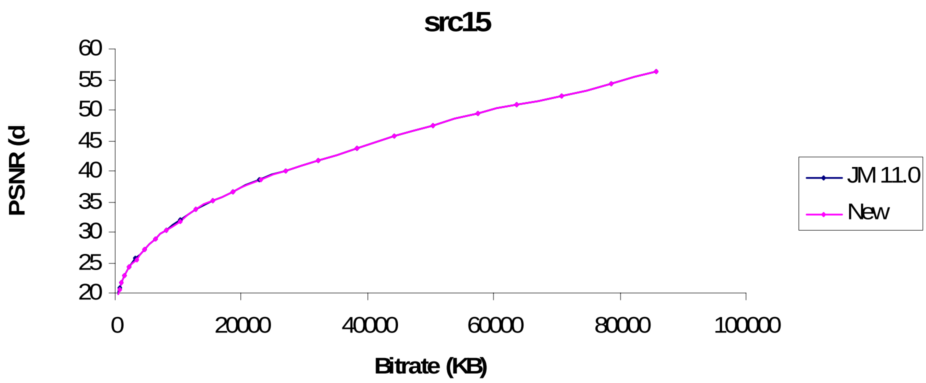


Figure 28 Rate – Distortion curves for src15

Table 9 PSNR and Bitrate for the src16 video sequence

src16	NEW		JM		% difference	
qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	58,34	18613	58,39	18429	-0,09	1,00
4	56,21	15848	56,22	15700	-0,02	0,94
6	54,38	13376	54,38	13216	0,00	1,21

8	53,11	11002	53,09	10865	0,04	1,26
10	51,91	9416	51,86	9297	0,10	1,28
12	50,64	7771	50,58	7629	0,12	1,86
14	49,37	6602	49,32	6456	0,10	2,26
16	47,99	5600	47,94	5468	0,10	2,41
18	46,47	4501	46,45	4366	0,04	3,09
20	45,12	3712	45,13	3591	-0,02	3,37
22	43,83	3094	43,87	2974	-0,09	4,03
24	42,34	2465	42,39	2372	-0,12	3,92
26	41,01	1992	41,06	1903	-0,12	4,68
28	39,7	1620	39,76	1537	-0,15	5,40
30	38,24	1302	38,31	1233	-0,18	5,60
32	36,86	1037	36,93	974	-0,19	6,47
34	35,61	842	35,68	786	-0,20	7,12
36	34,31	658	34,39	613	-0,23	7,34
38	33,08	532	33,18	494	-0,30	7,69
40	32,02	441	32,12	411	-0,31	7,30
42	30,86	368	31	340	-0,45	8,24
44	29,76	319	29,91	299	-0,50	6,69
46	28,81	270	28,91	260	-0,35	3,85
48	27,77	233	27,89	224	-0,43	4,02
50	26,72	215	26,81	211	-0,34	1,90
	Average % difference				-0,14	4,12

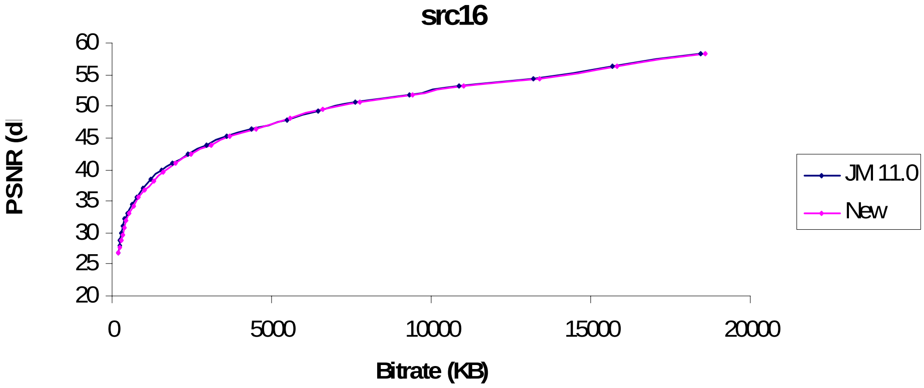


Figure 29 Rate – Distortion curves for src16

Table 10 PSNR and Bitrate for the src17 video sequence

src17	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	58,23	26126	58,26	25485	-0,05	2,52
4	56,05	23420	56,04	22826	0,02	2,60
6	54,21	21220	54,2	20635	0,02	2,83
8	52,76	18387	52,74	17834	0,04	3,10
10	51,51	16410	51,49	15884	0,04	3,31
12	50,04	14327	50,06	13790	-0,04	3,89
14	48,67	12604	48,71	12090	-0,08	4,25
16	46,97	11336	46,99	10816	-0,04	4,81
18	45,31	9688	45,37	9187	-0,13	5,45
20	43,8	8427	43,88	7938	-0,18	6,16
22	42,33	7317	42,42	6848	-0,21	6,85
24	40,54	6114	40,67	5717	-0,32	6,94
26	39,05	5195	39,16	4812	-0,28	7,96
28	37,56	4384	37,67	4384	-0,29	0,00
30	35,84	3659	35,99	3343	-0,42	9,45
32	34,31	3018	34,45	2726	-0,41	10,71
34	32,85	2505	33,03	2248	-0,54	11,43
36	31,3	1985	31,51	1767	-0,67	12,34
38	29,91	1622	30,13	1444	-0,73	12,33
40	28,68	1339	28,91	1193	-0,80	12,24
42	27,34	1098	27,59	992	-0,91	10,69
44	26,07	944	26,31	866	-0,91	9,01
46	24,94	769	25,16	711	-0,87	8,16
48	23,73	631	23,94	592	-0,88	6,59
50	22,53	552	22,7	527	-0,75	4,74
	Average % difference				-0,38	6,73

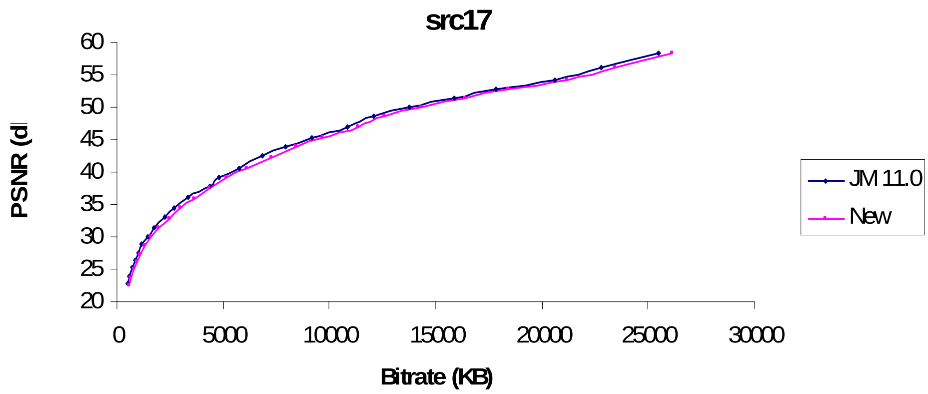


Figure 30 Rate – Distortion curves for src17

Table 11 PSNR and Bitrate for the src18 video sequence

src18	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,28	69039	56,28	69037	0,00	0,00
4	54,35	62666	54,35	62664	0,00	0,00
6	52,41	55571	52,41	55563	0,00	0,01
8	50,86	48701	50,87	48695	-0,02	0,01
10	49,43	42617	49,42	42626	0,02	-0,02
12	47,47	35538	47,47	35542	0,00	-0,01
14	45,81	29357	45,81	29357	0,00	0,00
16	44,06	23417	44,06	23383	0,00	0,15
18	42,09	17484	42,09	17453	0,00	0,18
20	40,44	13040	40,44	13003	0,00	0,28
22	38,93	9566	38,93	9526	0,00	0,42
24	37,36	6339	37,37	6314	-0,03	0,40
26	36,13	4237	36,13	4209	0,00	0,67
28	34,97	2839	34,97	2809	0,00	1,07
30	33,77	1835	33,77	1811	0,00	1,33
32	32,64	1207	32,64	1184	0,00	1,94
34	31,63	856	31,63	838	0,00	2,15
36	30,57	601	30,58	585	-0,03	2,74
38	29,5	461	29,5	447	0,00	3,13
40	28,54	368	28,55	357	-0,04	3,08
42	27,53	290	27,54	282	-0,04	2,84

44	26,62	233	26,65	227	-0,11	2,64
46	25,94	170	25,98	165	-0,15	3,03
48	25,34	131	25,42	128	-0,31	2,34
50	24,78	116	24,84	115	-0,24	0,87
	Average % difference				-0,04	1,17

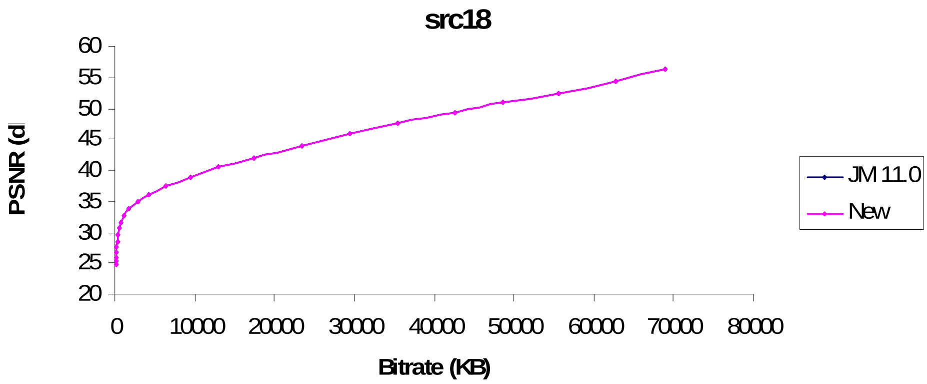


Figure 31 Rate – Distortion curves for src18

Table 12 PSNR and Bitrate for the src19 video sequence

src19	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,3	74054	56,31	73659	-0,02	0,54
4	54,37	67275	54,37	66864	0,00	0,61
6	52,44	59950	52,45	59557	-0,02	0,66
8	50,89	52895	50,89	52505	0,00	0,74
10	49,45	46601	49,45	46228	0,00	0,81
12	47,5	39360	47,51	38979	-0,02	0,98
14	45,88	33205	45,88	32825	0,00	1,16
16	44,16	27554	44,18	27085	-0,05	1,73
18	42,35	22009	42,37	21589	-0,05	1,95
20	40,88	17781	40,92	17374	-0,10	2,34
22	39,55	14523	39,6	14144	-0,13	2,68
24	38,06	11187	38,08	10963	-0,05	2,04
26	36,67	8876	36,69	8665	-0,05	2,44

28	35,36	7036	35,39	6836	-0,08	2,93
30	33,94	5475	33,96	5307	-0,06	3,17
32	32,65	4241	32,68	4085	-0,09	3,82
34	31,5	3336	31,53	3198	-0,10	4,32
36	30,26	2493	30,3	2376	-0,13	4,92
38	29,13	1884	29,16	1786	-0,10	5,49
40	28,11	1432	28,17	1354	-0,21	5,76
42	27,04	1056	27,1	994	-0,22	6,24
44	26,07	804	26,14	754	-0,27	6,63
46	25,23	599	25,32	566	-0,36	5,83
48	24,37	450	24,49	428	-0,49	5,14
50	23,63	368	23,73	355	-0,42	3,66
	Average % difference				-0,12	3,06

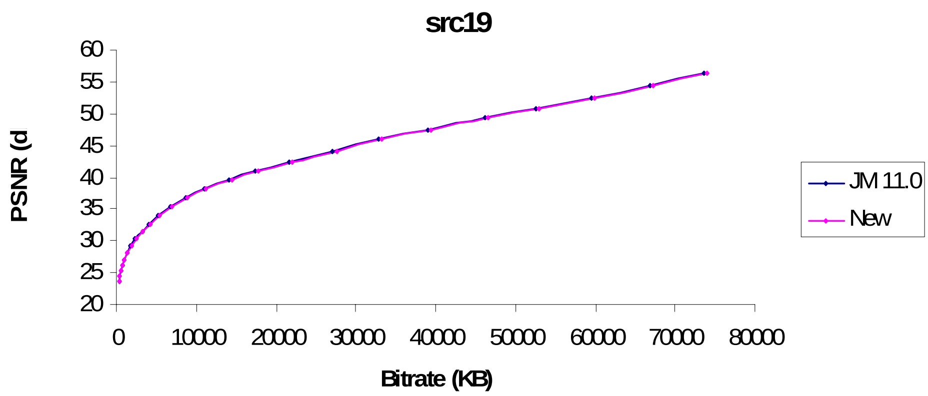


Figure 32 Rate – Distortion curves for src19

Table 13 PSNR and Bitrate for the src20 video sequence

src20	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,33	61723	56,33	61447	0,00	0,45
4	54,35	55129	54,35	54855	0,00	0,50
6	52,47	48152	52,47	47895	0,00	0,54
8	50,89	40994	50,9	40770	-0,02	0,55
10	49,47	35023	49,47	34820	0,00	0,58

12	47,58	28288	47,59	28106	-0,02	0,65
14	45,97	22366	45,99	22205	-0,04	0,73
16	44,22	16595	44,23	16474	-0,02	0,73
18	42,54	11407	42,54	11337	0,00	0,62
20	41,06	8051	41,06	7990	0,00	0,76
22	39,59	5389	39,6	5336	-0,03	0,99
24	37,99	3341	37,99	3314	0,00	0,81
26	36,65	2354	36,65	2330	0,00	1,03
28	35,35	1765	35,35	1739	0,00	1,50
30	33,92	1329	33,93	1306	-0,03	1,76
32	32,58	1024	32,59	1001	-0,03	2,30
34	31,32	810	31,33	789	-0,03	2,66
36	29,91	622	29,93	603	-0,07	3,15
38	28,6	494	28,61	475	-0,03	4,00
40	27,37	397	27,4	381	-0,11	4,20
42	26,01	315	26,04	300	-0,12	5,00
44	24,73	255	24,76	243	-0,12	4,94
46	23,58	207	23,61	197	-0,13	5,08
48	22,35	172	22,38	163	-0,13	5,52
50	21,17	147	21,21	143	-0,19	2,80
	Average % difference				-0,04	2,07

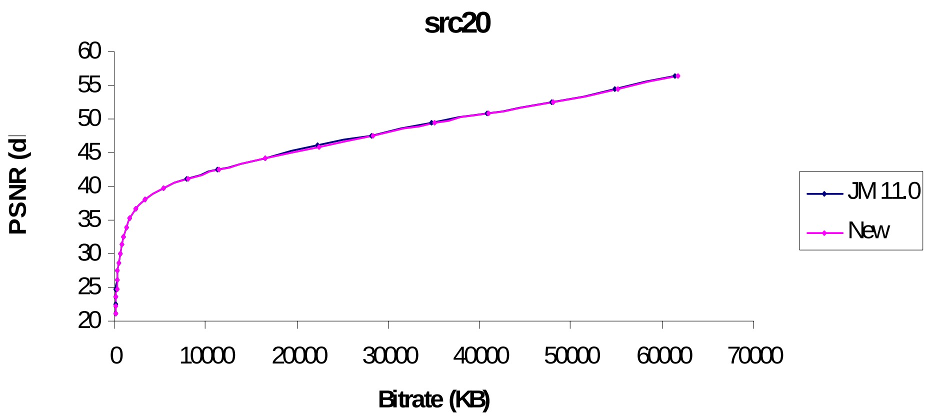


Figure 33 Rate – Distortion curves for src20

Table 14 PSNR and Bitrate for the src21 video sequence

src21	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,36	55683	56,36	55210	0,00	0,86
4	54,34	48997	54,33	48520	0,02	0,98
6	52,52	42094	52,53	41623	-0,02	1,13
8	50,93	34712	50,93	34298	0,00	1,21
10	49,51	28801	49,52	28443	-0,02	1,26
12	47,66	22624	47,67	22318	-0,02	1,37
14	46,07	17657	46,09	17392	-0,04	1,52
16	44,3	13056	44,34	12729	-0,09	2,57
18	42,78	8449	42,81	8325	-0,07	1,49
20	41,64	5857	41,65	5766	-0,02	1,58
22	40,6	4117	40,61	4037	-0,02	1,98
24	39,41	2679	39,42	2636	-0,03	1,63
26	38,39	1793	38,4	1754	-0,03	2,22
28	37,45	1237	37,46	1202	-0,03	2,91
30	36,44	844	36,47	815	-0,08	3,56
32	35,53	596	35,55	572	-0,06	4,20
34	34,66	447	34,7	426	-0,12	4,93
36	33,8	335	33,84	319	-0,12	5,02
38	32,92	261	32,97	246	-0,15	6,10
40	32,18	212	32,23	198	-0,16	7,07
42	31,34	170	31,42	160	-0,25	6,25
44	30,54	139	30,64	131	-0,33	6,11
46	29,65	121	29,76	115	-0,37	5,22
48	28,78	107	28,9	105	-0,42	1,90
50	27,89	108	28,04	108	-0,53	0,00
	Average % difference				-0,12	2,92

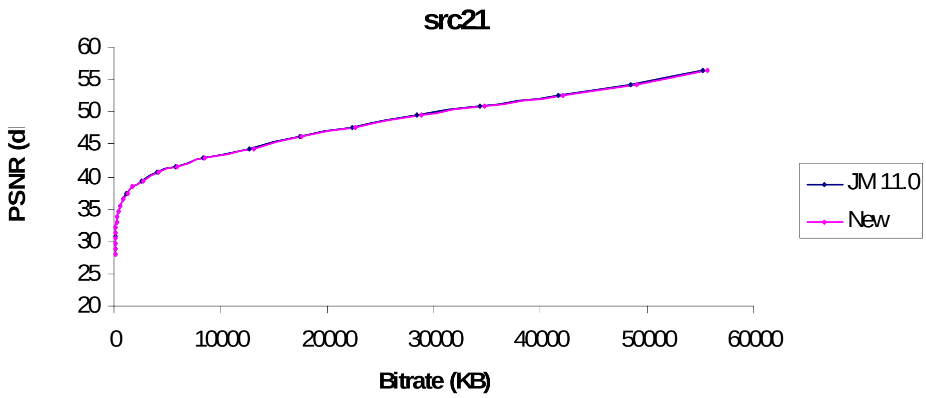


Figure 34 Rate – Distortion curves for src21

Table 15 PSNR and Bitrate for the src22 video sequence

src22	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,27	79365	56,27	79200	0,00	0,21
4	54,37	72659	54,37	72494	0,00	0,23
6	52,38	65384	52,38	65228	0,00	0,24
8	50,84	58456	50,84	58306	0,00	0,26
10	49,38	52188	49,39	52043	-0,02	0,28
12	47,38	45032	47,39	44899	-0,02	0,30
14	45,72	38760	45,72	38645	0,00	0,30
16	43,9	32799	43,91	32475	-0,02	1,00
18	41,89	26208	41,92	26002	-0,07	0,79
20	40,27	21158	40,29	21008	-0,05	0,71
22	38,84	17279	38,86	17163	-0,05	0,68
24	37,18	13481	37,19	13416	-0,03	0,48
26	35,76	10750	35,77	10685	-0,03	0,61
28	34,36	8556	34,37	8485	-0,03	0,84
30	32,79	6580	32,8	6516	-0,03	0,98
32	31,34	4982	31,35	4924	-0,03	1,18
34	29,99	3746	30	3688	-0,03	1,57
36	28,52	2592	28,53	2542	-0,04	1,97
38	27,2	1780	27,23	1737	-0,11	2,48
40	26,06	1230	26,09	1194	-0,11	3,02
42	24,87	838	24,91	808	-0,16	3,71

44	23,76	614	23,83	590	-0,29	4,07
46	22,8	464	22,85	444	-0,22	4,50
48	21,76	357	21,83	343	-0,32	4,08
50	20,81	294	20,87	284	-0,29	3,52
	Average % difference				-0,08	1,52

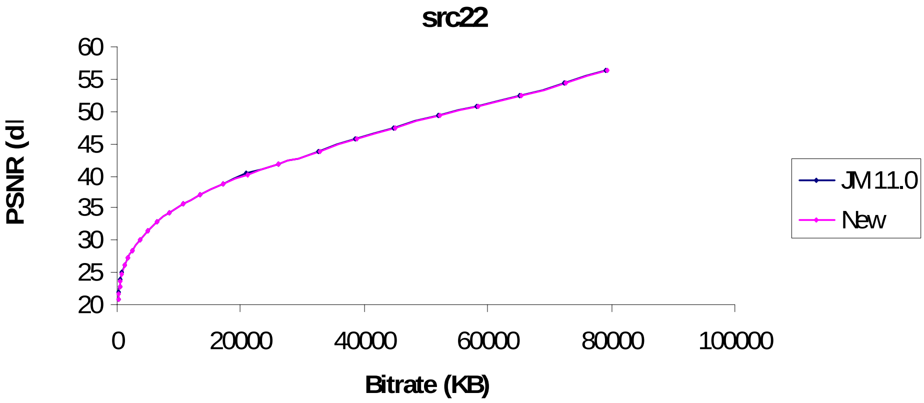


Figure 35 Rate – Distortion curves for src22

With the exception of video sequences src16 and src17, for all the other video sequences the rate – distortion curves for the two encoders are nearly identical, which means that there is no substantial difference between the performance of the two algorithms. In particular, results showed that for small QPs, i.e high bitrates the proposed architecture can achieve the almost the same performance compared to JM. For low-bitrate the performance of the proposed algorithm is reduced but remains competitive to that of JM11. The performance evaluation process also showed that the average PSNR remains practically the same, as the measurements gave us average differences that do not exceed the 0.4%. Video sequences src16 and src17 share a common property which is not met in the other video sequences. They are computer graphics. Furthermore, src17, for which the new algorithm has its worst results, has motion in opposite directions within the same frame. Nevertheless, the fact that for all the video sequences, which are produced by natural moving pictures, the new algorithm has nearly the same performance with SAD, shows its effectiveness.

The same evaluation process was repeated, changing this time the entropy encoding scheme from CAVLC to CABAC, for every value of the QP parameter. As in the case of CAVLC entropy encoding, the results of the comparisons between the bitrates of the two encoders are presented in detail in Appendix A, and show that the percentage of the difference between the filesizes of the video sequences produced by the two encoders does not exceed the 5.55%. More precisely, in the case of CABAC the maximum percentage of average difference between the filesizes produced by the two encoders is 5.541% and is met for QP = 40.

Taking a more careful look at the results presented in Appendix A, one can notice that the new algorithm has the same behaviour in both CAVLC and CABAC entropy encoding scheme, with a slight increase in the bitrate for CABAC. In particular, results showed that for small QPs (2 to 20), i.e high bitrates, the proposed architecture can achieve nearly the same performance compared to JM, as it does not increase the filesize of the produced video sequence more than 2%. The maximum average differences (above 5%) are met for QP being 38 to 46, while for QP greater than 46, the average difference is decreased. As it is clear from Figure 36 the new algorithm, in any case, does not produce an increase in the bitrate that exceeds the 6%, therefore it is save to say that the compression efficiency of the new algorithm is competitive to that of SAD.

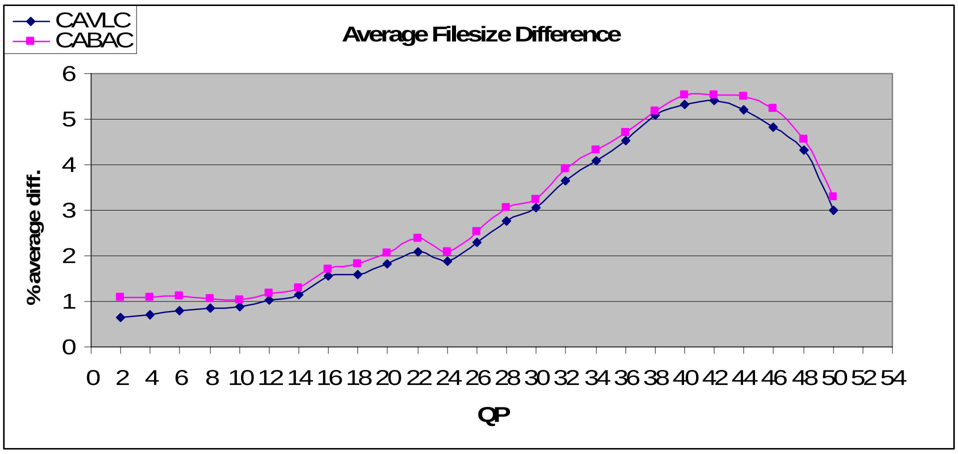


Figure 36 Percentage of the average filesize increment

To complete the evaluation of the new algorithm for the “EPZS” motion estimation scheme, we present the rate – distortion curves produced by each encoder, in the case of CABAC entropy encoding, for each of the ten video sequences. The data used for the curves is presented in the corresponding tables, where the average percentages of the difference between the filesizes and between the PSNR, for every sequence at different QPs, are also shown.

Table 16 PSNR and Bitrate for the src13 video sequence

src13	NEW		JM		% difference	
qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,42	73551	56,42	73058	0,00	0,67
4	54,45	65007	54,45	64612	0,00	0,61
6	52,5	56971	52,5	56615	0,00	0,63
8	50,91	49987	50,91	49663	0,00	0,65
10	49,47	44024	49,47	43740	0,00	0,65
12	47,55	37259	47,55	36995	0,00	0,71
14	45,94	31534	45,94	31276	0,00	0,82
16	44,23	26156	44,25	25796	-0,05	1,40
18	42,42	20591	42,45	20276	-0,07	1,55
20	40,97	16425	41,01	16103	-0,10	2,00
22	39,69	13248	39,73	12971	-0,10	2,14

24	38,26	10217	38,3	10040	-0,10	1,76
26	37,02	8127	37,05	7956	-0,08	2,15
28	35,78	6483	35,82	6319	-0,11	2,60
30	34,36	5123	34,4	4988	-0,12	2,71
32	33	3964	33,04	3836	-0,12	3,34
34	31,72	3110	31,77	3005	-0,16	3,49
36	30,33	2294	30,38	2204	-0,16	4,08
38	29,02	1735	29,08	1662	-0,21	4,39
40	27,88	1331	27,94	1269	-0,21	4,89
42	26,68	1000	26,74	955	-0,22	4,71
44	25,54	796	25,61	761	-0,27	4,60
46	24,61	598	24,69	571	-0,32	4,73
48	23,69	460	23,79	442	-0,42	4,07
50	22,75	389	22,88	379	-0,57	2,64
	Average % difference				-0,14	2,48

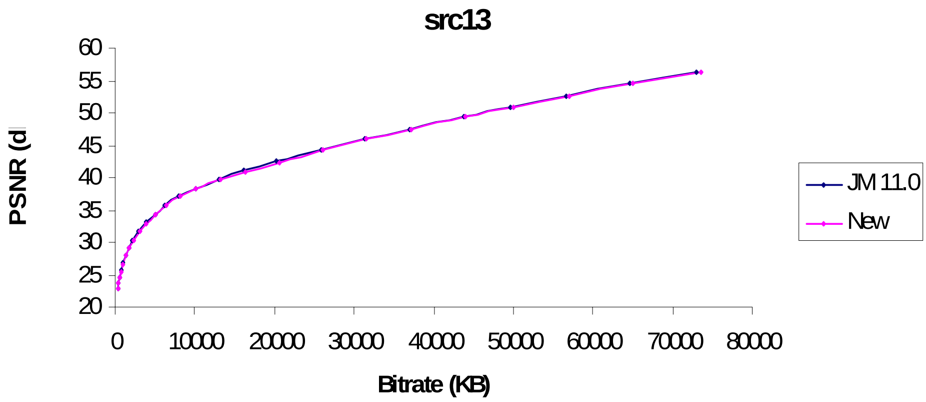


Figure 37 Rate – Distortion curves for src13

Table 17 PSNR and Bitrate for the src14 video sequence

src14	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,52	50700	56,53	50458	-0,02	0,48
4	54,43	44320	54,43	44064	0,00	0,58
6	52,65	38016	52,66	37751	-0,02	0,70

8	51,06	31598	51,07	31368	-0,02	0,73
10	49,7	26434	49,72	26244	-0,04	0,72
12	47,99	20991	48,03	20847	-0,08	0,69
14	46,57	16906	46,61	16810	-0,09	0,57
16	45,14	13344	45,16	13266	-0,04	0,59
18	43,62	10178	43,63	10100	-0,02	0,77
20	42,29	7956	42,29	7878	0,00	0,99
22	41,05	6248	41,05	6174	0,00	1,20
24	39,59	4598	39,6	4563	-0,03	0,77
26	38,31	3441	38,32	3403	-0,03	1,12
28	37,07	2553	37,08	2515	-0,03	1,51
30	35,67	1821	35,67	1790	0,00	1,73
32	34,39	1267	34,4	1238	-0,03	2,34
34	33,2	896	33,22	872	-0,06	2,75
36	31,93	582	31,97	565	-0,13	3,01
38	30,81	403	30,85	387	-0,13	4,13
40	29,81	292	29,85	281	-0,13	3,91
42	28,81	221	28,88	213	-0,24	3,76
44	27,85	179	27,94	172	-0,32	4,07
46	27,02	143	27,11	137	-0,33	4,38
48	26,21	120	26,28	116	-0,27	3,45
50	25,46	108	25,55	105	-0,35	2,86
	Average % difference				-0,10	1,91

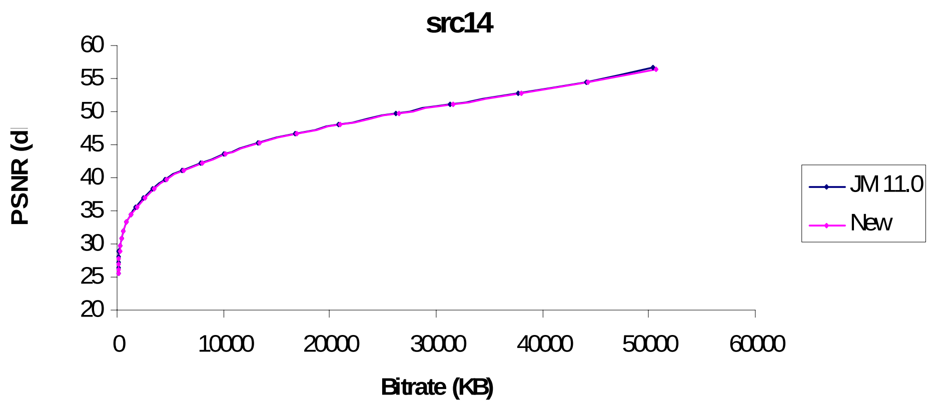


Figure 38 Rate – Distortion curves for src14

Table 18 PSNR and Bitrate for the src15 video sequence

src15	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,28	94736	56,28	94613	0,00	0,13
4	54,37	83383	54,37	83251	0,00	0,16
6	52,34	72044	52,34	71908	0,00	0,19
8	50,79	62458	50,79	62328	0,00	0,21
10	49,32	54637	49,32	54532	0,00	0,19
12	47,3	47184	47,3	47144	0,00	0,08
14	45,63	41390	45,63	41357	0,00	0,08
16	43,82	35874	43,83	35812	-0,02	0,17
18	41,78	30055	41,79	29992	-0,02	0,21
20	40,05	25240	40,05	25171	0,00	0,27
22	38,49	21188	38,5	21127	-0,03	0,29
24	36,68	17150	36,69	17095	-0,03	0,32
26	35,14	14110	35,14	14049	0,00	0,43
28	33,59	11571	33,59	11506	0,00	0,56
30	31,85	9227	31,86	9162	-0,03	0,71
32	30,26	7265	30,27	7202	-0,03	0,87
34	28,75	5672	28,76	5613	-0,03	1,05
36	27,07	4090	27,09	4034	-0,07	1,39
38	25,57	2918	25,59	2867	-0,08	1,78
40	24,21	2019	24,23	1971	-0,08	2,44
42	22,82	1287	22,85	1247	-0,13	3,21
44	21,65	861	21,7	825	-0,23	4,36
46	20,69	600	20,74	570	-0,24	5,26
48	19,68	455	19,77	427	-0,46	6,56
50	18,68	423	18,79	391	-0,59	8,18
	Average % difference				-0,08	1,56

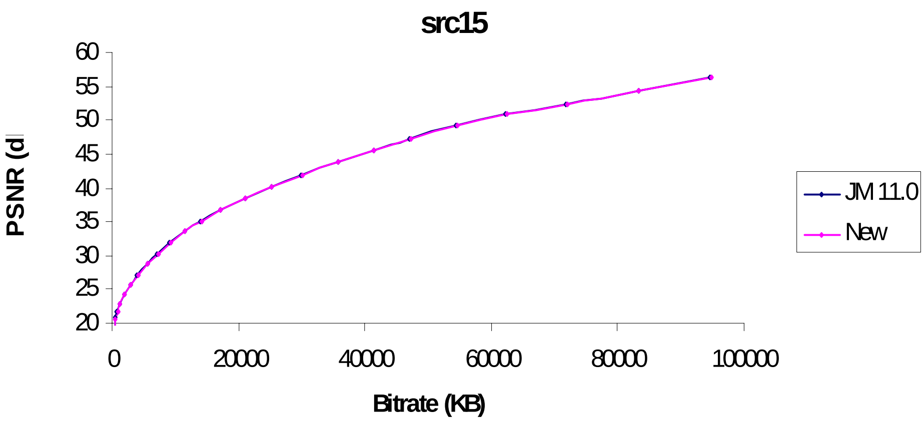


Figure 39 Rate – Distortion curves for src15

Table 19 PSNR and Bitrate for the src16 video sequence

src16	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	58,34	17825	58,39	17594	-0,09	1,31
4	56,21	15251	56,22	15060	-0,02	1,27
6	54,38	12887	54,38	12690	0,00	1,55
8	53,11	10545	53,09	10376	0,04	1,63
10	51,91	8931	51,86	8784	0,10	1,67
12	50,64	7310	50,58	7149	0,12	2,25
14	49,37	6160	49,32	5996	0,10	2,74
16	47,99	5182	47,94	5036	0,10	2,90
18	46,47	4167	46,45	4019	0,04	3,68
20	45,12	3425	45,13	3297	-0,02	3,88
22	43,83	2834	43,87	2708	-0,09	4,65
24	42,34	2241	42,39	2151	-0,12	4,18
26	41,01	1806	41,06	1721	-0,12	4,94
28	39,7	1463	39,76	1384	-0,15	5,71
30	38,24	1168	38,31	1105	-0,18	5,70
32	36,86	927	36,93	868	-0,19	6,80
34	35,61	747	35,68	696	-0,20	7,33
36	34,31	577	34,39	537	-0,23	7,45
38	33,08	459	33,18	426	-0,30	7,75
40	32,02	374	32,12	348	-0,31	7,47

42	30,86	303	31	281	-0,45	7,83
44	29,76	258	29,91	241	-0,50	7,05
46	28,81	212	28,91	203	-0,35	4,43
48	27,77	179	27,89	171	-0,43	4,68
50	26,72	161	26,81	158	-0,34	1,90
	Average % difference				-0,14	4,43

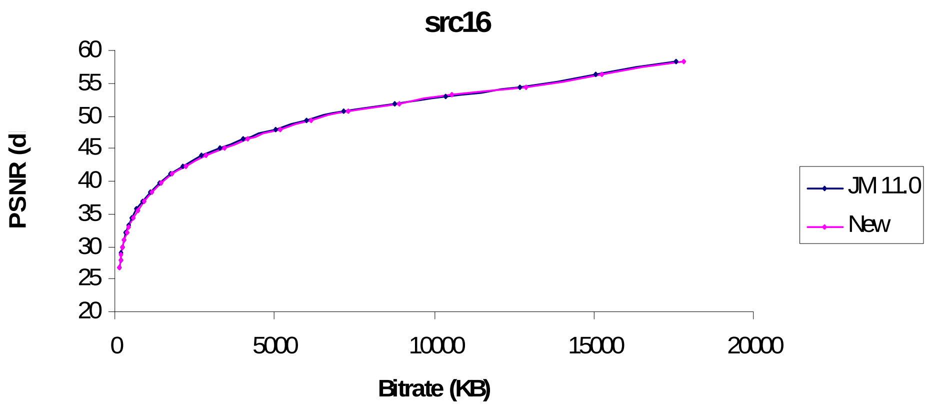


Figure 40 Rate – Distortion curves for src16

Table 20 PSNR and Bitrate for the src17 video sequence

src17	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	58,23	28269	58,26	26847	-0,05	5,30
4	56,05	24519	56,04	23275	0,02	5,34
6	54,21	21338	54,2	20319	0,02	5,02
8	52,76	18036	52,74	17283	0,04	4,36
10	51,51	15877	51,49	15258	0,04	4,06
12	50,04	13750	50,06	13151	-0,04	4,55
14	48,67	12038	48,71	11470	-0,08	4,95
16	46,97	10753	46,99	10203	-0,04	5,39
18	45,31	9153	45,37	8632	-0,13	6,04
20	43,8	7920	43,88	7428	-0,18	6,62
22	42,33	6843	42,42	6380	-0,21	7,26
24	40,54	5702	40,67	5300	-0,32	7,58

26	39,05	4816	39,16	4441	-0,28	8,44
28	37,56	4043	37,67	3701	-0,29	9,24
30	35,84	3359	35,99	3066	-0,42	9,56
32	34,31	2752	34,45	2488	-0,41	10,61
34	32,85	2270	33,03	2042	-0,54	11,17
36	31,3	1778	31,51	1591	-0,67	11,75
38	29,91	1434	30,13	1283	-0,73	11,77
40	28,68	1167	28,91	1046	-0,80	11,57
42	27,34	940	27,59	852	-0,91	10,33
44	26,07	795	26,31	730	-0,91	8,90
46	24,94	637	25,16	589	-0,87	8,15
48	23,73	515	23,94	483	-0,88	6,63
50	22,53	444	22,7	425	-0,75	4,47
	Average % difference				-0,38	7,56

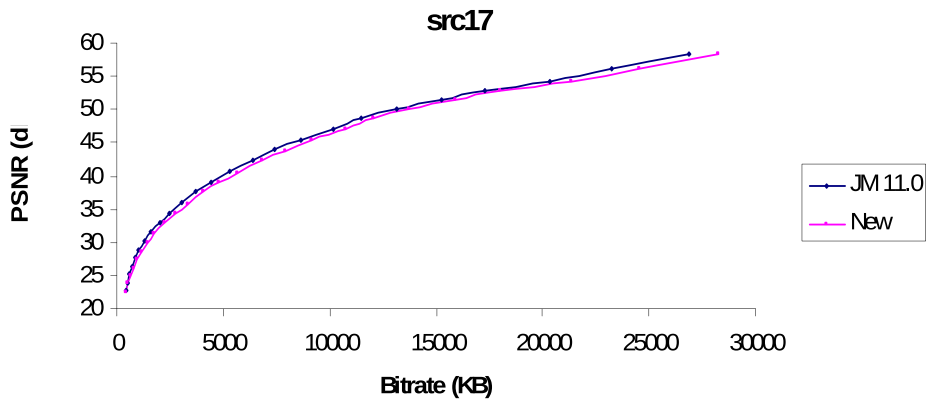


Figure 41 Rate – Distortion curves for src17

Table 21 PSNR and Bitrate for the src18 video sequence

src18	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,28	64337	56,28	64271	0,00	0,10
4	54,35	57783	54,35	57726	0,00	0,10
6	52,41	50697	52,41	50641	0,00	0,11

8	50,86	44184	50,87	44165	-0,02	0,04
10	49,43	38527	49,42	38525	0,02	0,01
12	47,47	31971	47,47	31965	0,00	0,02
14	45,81	26439	45,81	26430	0,00	0,03
16	44,06	21267	44,06	21236	0,00	0,15
18	42,09	15929	42,09	15898	0,00	0,19
20	40,44	11911	40,44	11879	0,00	0,27
22	38,93	8702	38,93	8667	0,00	0,40
24	37,36	5740	37,37	5713	-0,03	0,47
26	36,13	3801	36,13	3773	0,00	0,74
28	34,97	2522	34,97	2494	0,00	1,12
30	33,77	1614	33,77	1592	0,00	1,38
32	32,64	1055	32,64	1034	0,00	2,03
34	31,63	744	31,63	727	0,00	2,34
36	30,57	517	30,58	502	-0,03	2,99
38	29,5	391	29,5	377	0,00	3,71
40	28,54	307	28,55	294	-0,04	4,42
42	27,53	235	27,54	226	-0,04	3,98
44	26,62	184	26,65	177	-0,11	3,95
46	25,94	133	25,98	127	-0,15	4,72
48	25,34	101	25,42	98	-0,31	3,06
50	24,78	88	24,84	87	-0,24	1,15
	Average % difference				-0,04	1,50

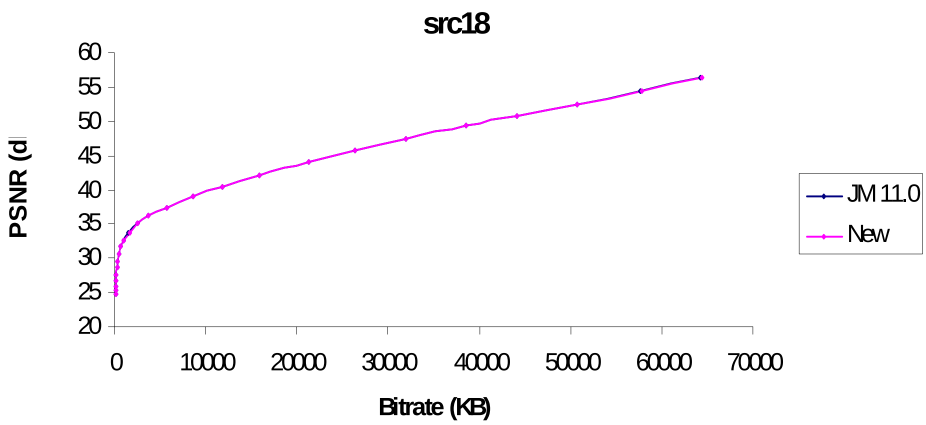


Figure 42 Rate – Distortion curves for src18

Table 22 PSNR and Bitrate for the src19 video sequence

src19	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,3	71761	56,31	71021	-0,02	1,04
4	54,37	62875	54,37	62187	0,00	1,11
6	52,44	54739	52,45	54165	-0,02	1,06
8	50,89	48017	50,89	47490	0,00	1,11
10	49,45	42167	49,45	41676	0,00	1,18
12	47,5	35500	47,51	35013	-0,02	1,39
14	45,88	29896	45,88	29460	0,00	1,48
16	44,16	24735	44,18	24173	-0,05	2,32
18	42,35	19609	42,37	19096	-0,05	2,69
20	40,88	15686	40,92	15180	-0,10	3,33
22	39,55	12668	39,6	12213	-0,13	3,73
24	38,06	9583	38,08	9351	-0,05	2,48
26	36,67	7532	36,69	7303	-0,05	3,14
28	35,36	5928	35,39	5711	-0,08	3,80
30	33,94	4551	33,96	4385	-0,06	3,79
32	32,65	3485	32,68	3329	-0,09	4,69
34	31,5	2699	31,53	2570	-0,10	5,02
36	30,26	1985	30,3	1882	-0,13	5,47
38	29,13	1473	29,16	1390	-0,10	5,97
40	28,11	1104	28,17	1039	-0,21	6,26
42	27,04	800	27,1	751	-0,22	6,52
44	26,07	599	26,14	561	-0,27	6,77
46	25,23	441	25,32	415	-0,36	6,27
48	24,37	331	24,49	315	-0,49	5,08
50	23,63	271	23,73	260	-0,42	4,23
	Average % difference				-0,12	3,60

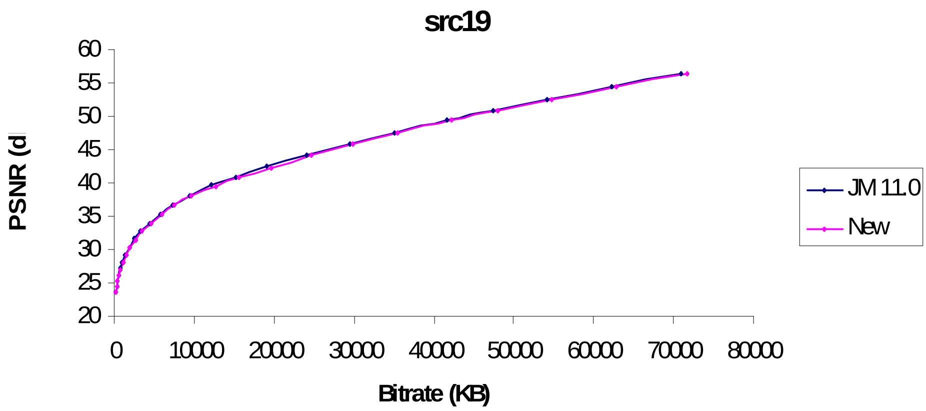


Figure 43 Rate – Distortion curves for src19

Table 23 PSNR and Bitrate for the src20 video sequence

src20	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,33	56242	56,33	56018	0,00	0,40
4	54,35	49847	54,35	49621	0,00	0,46
6	52,47	42922	52,47	42712	0,00	0,49
8	50,89	36305	50,9	36126	-0,02	0,50
10	49,47	30783	49,47	30616	0,00	0,55
12	47,58	24567	47,59	24411	-0,02	0,64
14	45,97	19395	45,99	19278	-0,04	0,61
16	44,22	14490	44,23	14387	-0,02	0,72
18	42,54	10088	42,54	10022	0,00	0,66
20	41,06	7181	41,06	7122	0,00	0,83
22	39,59	4894	39,6	4840	-0,03	1,12
24	37,99	3102	37,99	3073	0,00	0,94
26	36,65	2203	36,65	2177	0,00	1,19
28	35,35	1653	35,35	1626	0,00	1,66
30	33,92	1241	33,93	1218	-0,03	1,89
32	32,58	956	32,59	934	-0,03	2,36
34	31,32	756	31,33	736	-0,03	2,72
36	29,91	579	29,93	561	-0,07	3,21
38	28,6	457	28,61	440	-0,03	3,86
40	27,37	365	27,4	350	-0,11	4,29

42	26,01	286	26,04	273	-0,12	4,76
44	24,73	229	24,76	218	-0,12	5,05
46	23,58	182	23,61	173	-0,13	5,20
48	22,35	146	22,38	137	-0,13	6,57
50	21,17	119	21,21	115	-0,19	3,48
	Average % difference				-0,04	2,17

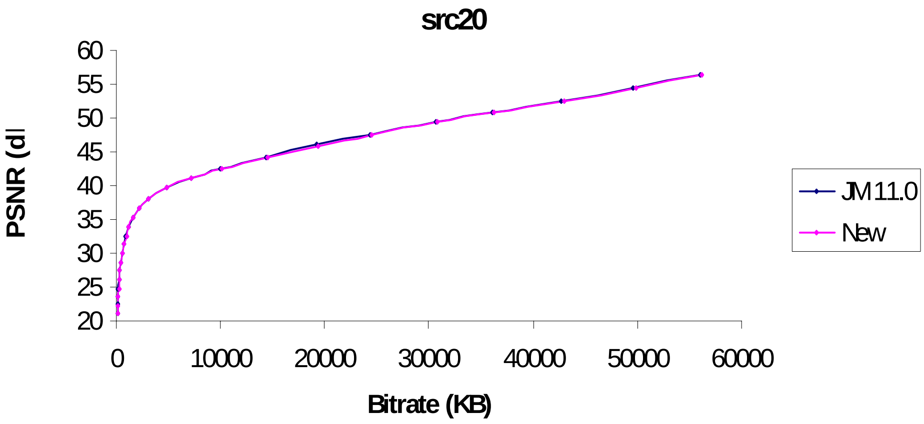


Figure 44 Rate – Distortion curves for src20

Table 24 PSNR and Bitrate for the src21 video sequence

src21	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,36	49349	56,36	49012	0,00	0,69
4	54,34	43223	54,33	42884	0,02	0,79
6	52,52	36789	52,53	36454	-0,02	0,92
8	50,93	30275	50,93	29989	0,00	0,95
10	49,51	25060	49,52	24814	-0,02	0,99
12	47,66	19625	47,67	19402	-0,02	1,15
14	46,07	15369	46,09	15173	-0,04	1,29
16	44,3	11379	44,34	11118	-0,09	2,35
18	42,78	7464	42,81	7355	-0,07	1,48
20	41,64	5182	41,65	5095	-0,02	1,71
22	40,6	3622	40,61	3544	-0,02	2,20

24	39,41	2337	39,42	2294	-0,03	1,87
26	38,39	1551	38,4	1512	-0,03	2,58
28	37,45	1062	37,46	1026	-0,03	3,51
30	36,44	712	36,47	686	-0,08	3,79
32	35,53	499	35,55	476	-0,06	4,83
34	34,66	371	34,7	351	-0,12	5,70
36	33,8	274	33,84	259	-0,12	5,79
38	32,92	209	32,97	197	-0,15	6,09
40	32,18	167	32,23	156	-0,16	7,05
42	31,34	133	31,42	125	-0,25	6,40
44	30,54	108	30,64	102	-0,33	5,88
46	29,65	93	29,76	89	-0,37	4,49
48	28,78	80	28,9	79	-0,42	1,27
50	27,89	75	28,04	75	-0,53	0,00
	Average % difference				-0,12	2,95

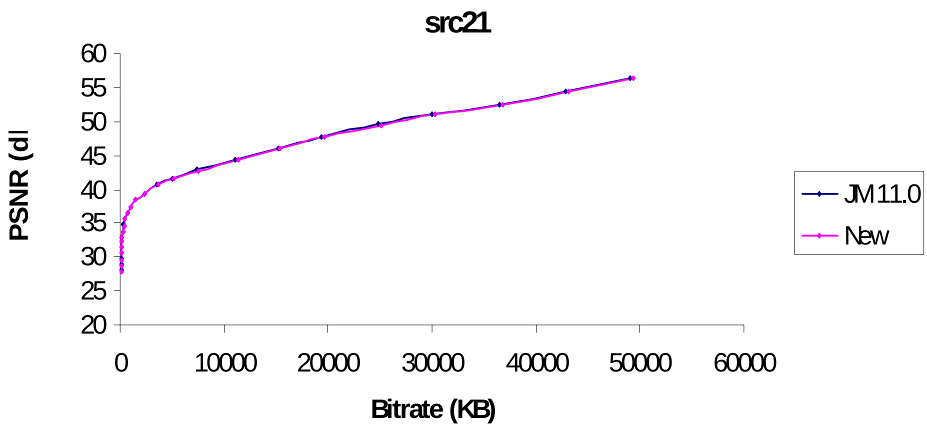


Figure 45 Rate – Distortion curves for src21

Table 25 PSNR and Bitrate for the src22 video sequence

src22	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,27	83119	56,27	82464	0,00	0,79
4	54,37	72473	54,37	72171	0,00	0,42

6	52,38	62021	52,38	61742	0,00	0,45
8	50,84	54210	50,84	54025	0,00	0,34
10	49,38	48203	49,39	48032	-0,02	0,36
12	47,38	41462	47,39	41333	-0,02	0,31
14	45,72	35787	45,72	35675	0,00	0,31
16	43,9	30414	43,91	30127	-0,02	0,95
18	41,89	24424	41,92	24225	-0,07	0,82
20	40,27	19757	40,29	19609	-0,05	0,75
22	38,84	16059	38,86	15945	-0,05	0,71
24	37,18	12465	37,19	12402	-0,03	0,51
26	35,76	9867	35,77	9804	-0,03	0,64
28	34,36	7793	34,37	7724	-0,03	0,89
30	32,79	5938	32,8	5880	-0,03	0,99
32	31,34	4451	31,35	4398	-0,03	1,21
34	29,99	3316	30	3266	-0,03	1,53
36	28,52	2271	28,53	2227	-0,04	1,98
38	27,2	1545	27,23	1508	-0,11	2,45
40	26,06	1057	26,09	1025	-0,11	3,12
42	24,87	708	24,91	682	-0,16	3,81
44	23,76	511	23,83	490	-0,29	4,29
46	22,8	380	22,85	363	-0,22	4,68
48	21,76	288	21,83	276	-0,32	4,35
50	20,81	234	20,87	225	-0,29	4,00
	Average % difference				-0,08	1,63

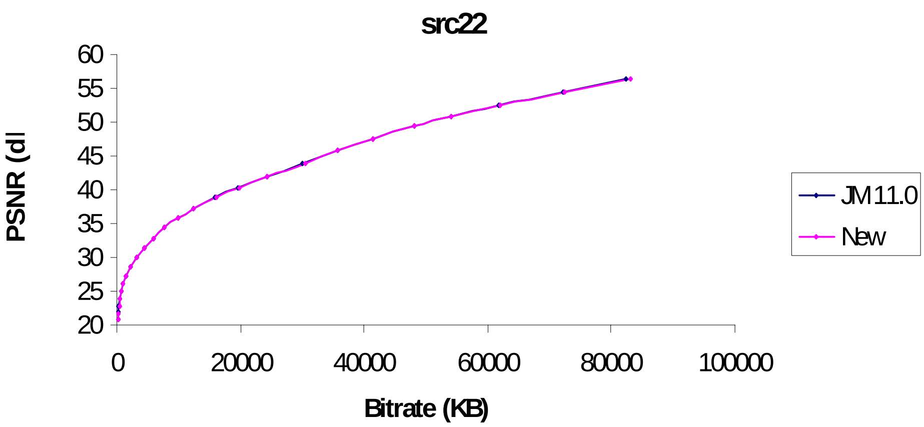


Figure 46 Rate – Distortion curves for src22

The same evaluation process was followed for the case of Full Motion Estimation and for QP being 2, 14, 28, 38 and 50. The results of the comparisons between the bit-rates of the two encoders, along with the data selected for the rate – distortion curves and the respective r – d graphs are presented in Appendix A. These results, although they show that the new algorithm has nearly the same behaviour in both EPZS and Full Motion Search, they present greater variation for low and high bit-rates. More precisely, for high bit-rates the proposed algorithm can perform even better than JM, as the average bit-rate difference for QP = 14 is -0.18 %, while for low bit-rates the performance of the new algorithm is reduced, as the average bit-rate difference for QP = 50 is -15.85 %.

This difference between the results for EPZS and Full Motion Estimation is due to the fact that the philosophy of the proposed algorithm is more close to the one of fast motion estimation algorithms rather that full motion estimation. In full search we calculate the SAD for each position in the search area (for a 16x16 search area there are 1089 positions) and we select the one with the minimum cost. In EPZS we calculate the SAD for the few (usually 4 – 8) candidate positions which will be used as the centre of the search pattern, and select the position with the smallest cost as the best predictor. Then we proceed, using the best predictor as centre, and calculate the SAD for the (limited) positions within this search pattern and selected, similarly, the best position. In the presented algorithm we calculate the SGV for two positions and select the one with the minimum cost. Then we proceed by calculating the SGV for the best position so far and the next candidate and select, similarly, the new best position.

5. HARDWARE IMPLEMENTATION

5.1. Introduction

The H.264 digital video standard promises to be an excellent video format for use with a broad spectrum of applications. Its high coding efficiency, achieved by the introduction of many new features, especially in Macroblock Prediction, makes H.264 ideal for mobile multimedia applications. Nevertheless, real-time encoding is the main requirement for the adoption of the standard by the consumer marketplace. However, the new features adopted by H.264 result in a considerably higher encoder complexity that adversely affects delay and power, which are both critical for real-time applications. Therefore, it is of high importance to design architectures that minimize the delay and power overhead for the implementation of prediction modes.

In the previous chapters we presented and validated a new algorithm that can replace the standard Sum of Absolute Differences (SAD) approach in both Intra and Inter prediction modes as they are defined in the standard. The simulation results of the software implementation showed that the proposed algorithm maintains the same quality level as SAD, without having any perceivable increase of the file size.

In this chapter we present the architecture of the new algorithm and its hardware implementation. This architecture brings significant reduction to the complexity of the encoder, which results in meeting the real-time encoding requirements of H.264. Furthermore, the architectures for Intra and Inter prediction modes using the new algorithm are presented and compared against the equivalent SAD ones.

5.2. The New Algorithm

5.2.1. The architecture of the first approach

The first approach we decided to follow for the problem of Macroblock Prediction was qualitative rather than quantitative, as described in detail in Chapter 3. By this approach we need only find the best prediction and not quantify the result compared to the other candidates. As a consequence, the first version of the algorithm compares the absolute differences of all 4x4 prediction blocks and selects the one with the largest number of minimum values. The new implementation has two 128bit numbers as inputs (each one representing an entire 4x4 block of 8bit numbers), compares them and signals the results of the comparison through a 2-bit output.

The comparison is not performed by the standard comparator [73] but by specialized bit-wise circuit developed specifically for this application. The comparison between absolute difference i of candidate j and absolute difference i of candidate k is done by the circuit, whose block diagram is shown in Figure 47 and may be more easily understood if presented like the following pseudo-code:

```
main()
{
     bit candidate1[8], candidate2[8], equal[8], comp[3], equ[3];
     bit comp1, equal1, comp2, equal2, comp3, equal3;
     bit final_comp, final_equal;

//Process of First Level
     if (candidate1== candidate2)
           equal = 1;
     else
           equal = 0;

//Process of the first 3-bit Comparator, which has inputs
//candidate2[2 downto 0] and equal[2 downto 0] and outputs comp1
//and equal1
     if (candidate2[2 downto 0]> candidate1[2 downto 0])
           comp1 = 1;
     else
           comp1 = 0;
     if (equal[2 downto 0] == 1)
           equal1 = 1;
     else
           equal1 = 0;

//Process of the second 3-bit Comparator which has inputs
//candidate2[5 downto 3] and equal[5 downto 3] and outputs comp2
//and equal2
```

```
     if (candidate2[5 downto 3]> candidate1[5 downto 3])
          comp2 = 1;
     else
          comp2 = 0;
     if (equal[5 downto 3] == 1)
          equal2 = 1;
     else
          equal2 = 0;

//Process of the 2-bit Comparator which has inputs
//candidate2[7 downto 6] and equal[7 downto 6] and outputs comp3
//and equal3
     if (candidate2[7 downto 6]> candidate1[7 downto 6])
          comp3 = 1;
     else
          comp3 = 0;
     if (equal[7 downto 6] == 1)
          equal3 = 1;
     else
          equal3 = 0;

     comp = comp3 & comp2 & comp1;
     equ = equ3 & equ2 & equ1;

//Process of the last 3-bit Comparator which has inputs
//comp and equ and outputs final_comp andfinal_equal
     if (candidate2 > candidate1)
          final_comp = 1;
     else
          final_comp = 0;
     if (equ == 1)
          final_equal = 1;
     else
          final_equal = 0;
}
```

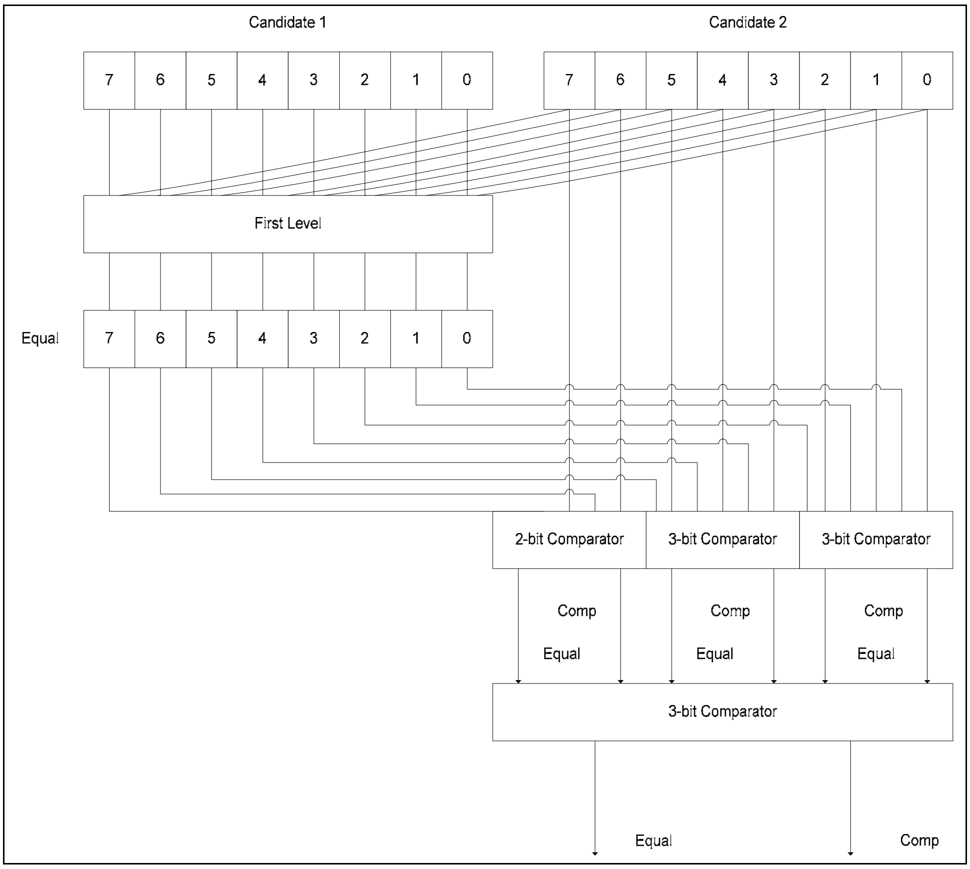


Figure 47 Block Diagram of an 8-bit Comparator

The absolute difference selection circuit comprises of 3 stages, as shown in Figure 47. In the first stage, denoted as "First Level", we determine which of the 8 bits of the two inputs are equal. The other two stages are those that determine which one of the initial input vectors represents an 8-bit number with greater magnitude.

The first stage has candidates 1 and 2 as inputs and outputs an 8-bit vector, named *equal*, which indicates the bits that are equal between the two input vectors. The "First Level" circuit comprises of 8 2-input *xnor* gates, whose one input comes from candidate 1 and the other is the equivalent bit of candidate 2.

In the second stage we have the vector *equal*, produced in the previous stage and one of the two candidates – in the present implementation the second one, as inputs. We part the 8-bit vectors in two triplets and one pair. For each one of these components we perform a comparison that determines if the respective bits of the second candidate are greater than those of the first or not. For the triplets we use a three-bit comparator and for the pair we use a two-bit one.

The three-bit comparator works as follows. If the third bit of the second candidate is set to 1 and, at the same time, the third bit of *equal* is set to 0, then the number represented by the second candidate is greater than the first one. The same applies when the third bit of *equal* and the second bit of the second candidate are set to 1 and, at the same time the second bit of *equal* is set to 0. When the third and second bit of *equal* and the first bit of the second candidate are set to 1, then whatever value may the first bit of *equal* have the second candidate will be either greater than or equal to the first one. In the same sense works the two-bit comparator. The truth table that shows the encoding of the output of the 3-bit Comparator is shown in Table 26, while the equivalent truth table for the 2-bit Comparator is shown in Table 27. The schematics of the 3-bit Comparator and the 2-bit Comparator are shown in Figures 48 and 49, respectively.

Table 26 Truth Table of the 3-bit Comparator

Cand2 (2)	Cand2 (1)	Cand2 (0)	Equal (2)	Equal (1)	Equal (0)	Comp	Equal
1	X	X	0	X	X	1	0
X	1	X	1	0	X	1	0
X	X	1	1	1	0	1	0
X	X	1	1	1	1	1	1

Table 27 Truth Table of the 2-bit Comparator

Cand2 (1)	Cand2 (0)	Equal (1)	Equal (0)	Comp	Equal
1	X	0	X	1	0
X	1	1	0	1	0
X	1	1	1	1	1

At the output of this stage we have 3 bits that represent equality and three bits that represent if the corresponding part of candidate 2 is greater than the equivalent part of candidate 1. In the last stage the output of the second stage drives a three-bit comparator that produces the bit *comp,* which is 1 if candidate 2 is greater than or equal to candidate 1 and 0 if it is smaller, and the bit *equal* which is 1 if the two candidates are equal.

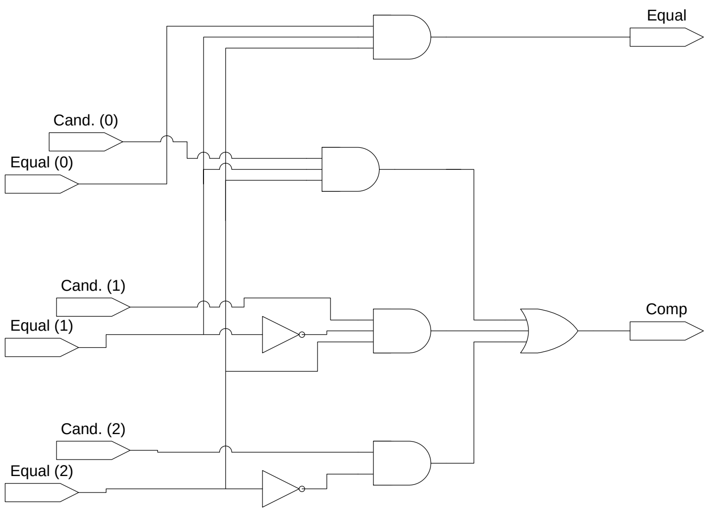


Figure 48 Schematic of a 3-bit Comparator

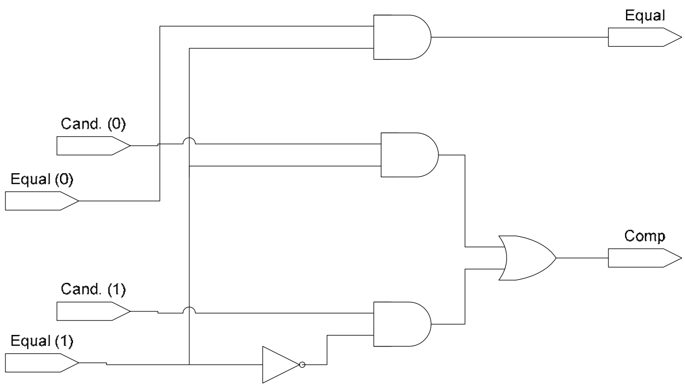


Figure 49 Schematic of a 2-bit Comparator

Next we proceed to determine which of the two candidates has the most minimum values. Each candidate has a total of 16 pixel values absolute differences i.e. 8-bit vectors. The circuit described above produces a selection bit for each of these 16 differences, or more precisely, it produces a 1 if the absolute difference of the first candidate is smaller than or equal to the absolute difference of the second candidate. This implementation takes into account the equality in favour of the first candidate. Therefore the 16-bit vector produced by the aforementioned stage has the necessary information for both candidates, as the remaining bits that are not set to 1 denote the absolute difference values of the second candidate that are smaller. This, consequently, means that if the number of ones in this bit vector is equal to or greater than 8 then the first candidate has the largest number of minimum values and, therefore, is the best candidate, otherwise it is the second one. The block diagram of the circuit implementing this function is shown in Figure 50.

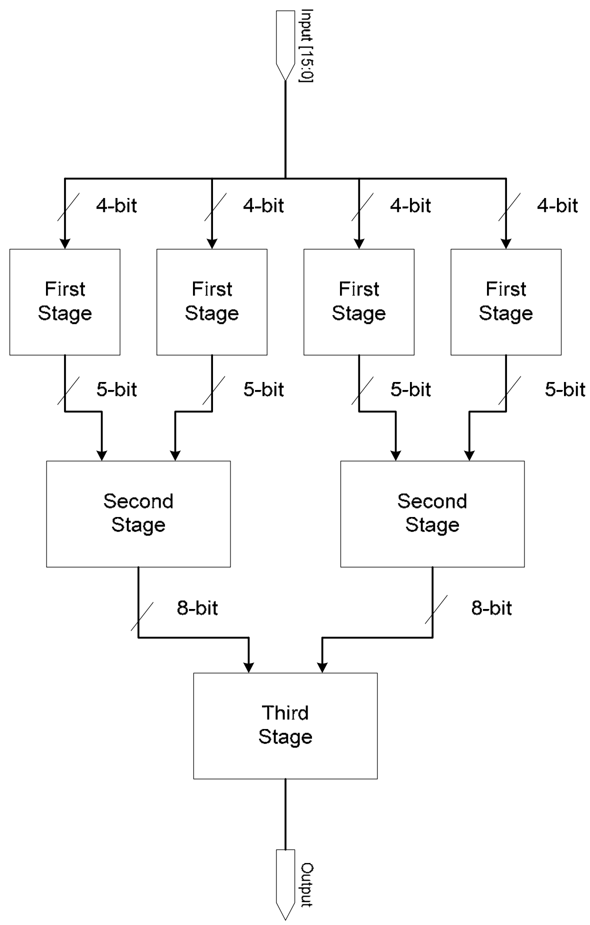


Figure 50 Block Diagram of the Selection Circuit

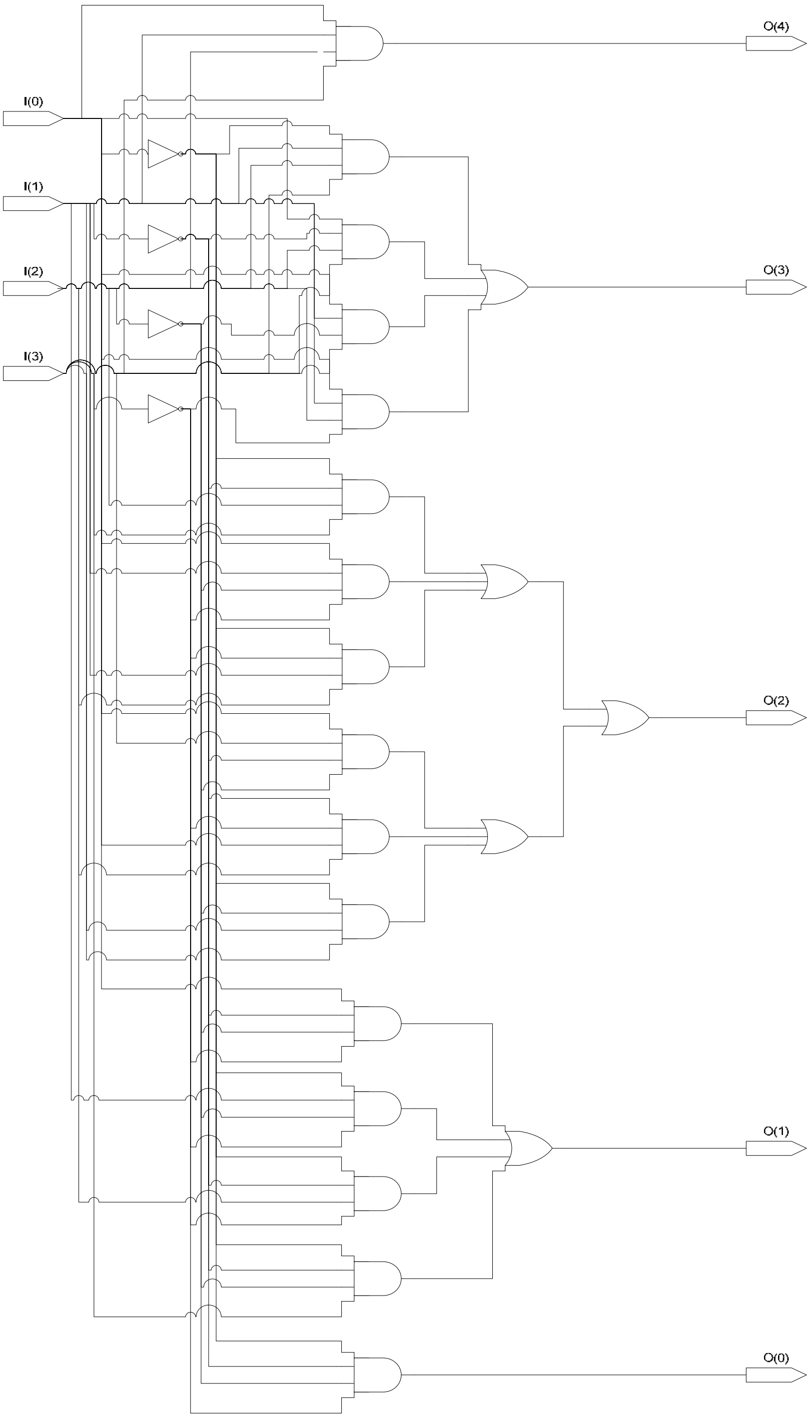


Figure 51 Schematic of the "FirstStage" Circuit

The first stage splits the 16-bit vector in four quadruplets. For each quadruplet we count the number of ones. This is done with the circuit named FirstStage in Figure 50. This circuit takes as input a 4-bit vector and has output a 5-bit vector named *sum*. The output vector has only one, the position of which indicates the number of ones found in the input vector. More precisely, if the bit *sum(4)* is set to one then the number of ones presented in the input is four, if the bit *sum(3)* is set to one then the number of ones presented in the input is three and so on. The circuit just described implements the following truth table, and its schematic is shown in Figure 51.

Table 28 Truth Table of FirstStage

I(3)	I(2)	I(1)	I(0)	Sum(4)	Sum(3)	Sum(2)	Sum(1)	Sum(0)
0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	1	0
0	0	1	0	0	0	0	1	0
0	0	1	1	0	0	1	0	0
0	1	0	0	0	0	0	1	0
0	1	0	1	0	0	1	0	0
0	1	1	0	0	0	1	0	0
0	1	1	1	0	1	0	0	0
1	0	0	0	0	0	0	1	0
1	0	0	1	0	0	1	0	0
1	0	1	0	0	0	1	0	0
1	0	1	1	0	1	0	0	0
1	1	0	0	0	0	1	0	0
1	1	0	1	0	1	0	0	0
1	1	1	0	0	1	0	0	0
1	1	1	1	1	0	0	0	0

The bit-vectors produced in the first stage are the inputs to the second stage. In this stage we have two circuits like the one in Figure 52. Each one of them takes as input two 5-bit vectors that represent a value between 4 and 0, and has as output a 9-bit vector. As in the first stage, the output has only one "1" and its position indicates a number between 0 and 8, which is the sum of ones that can be found in two quadruplets. The truth table of the circuit that implements the above function is shown in Table 29.

Table 29 Truth Table of SecondStage

I1 (4)	I1 (3)	I1 (2)	I1 (1)	I1 (0)	I2 (4)	I2 (3)	I2 (2)	I2 (1)	I2 (0)	Sum (8)	Sum (7)	Sum (6)	Sum (5)	Sum (4)	Sum (3)	Sum (2)	Sum (1)	Sum (0)
0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0
0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0
0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0

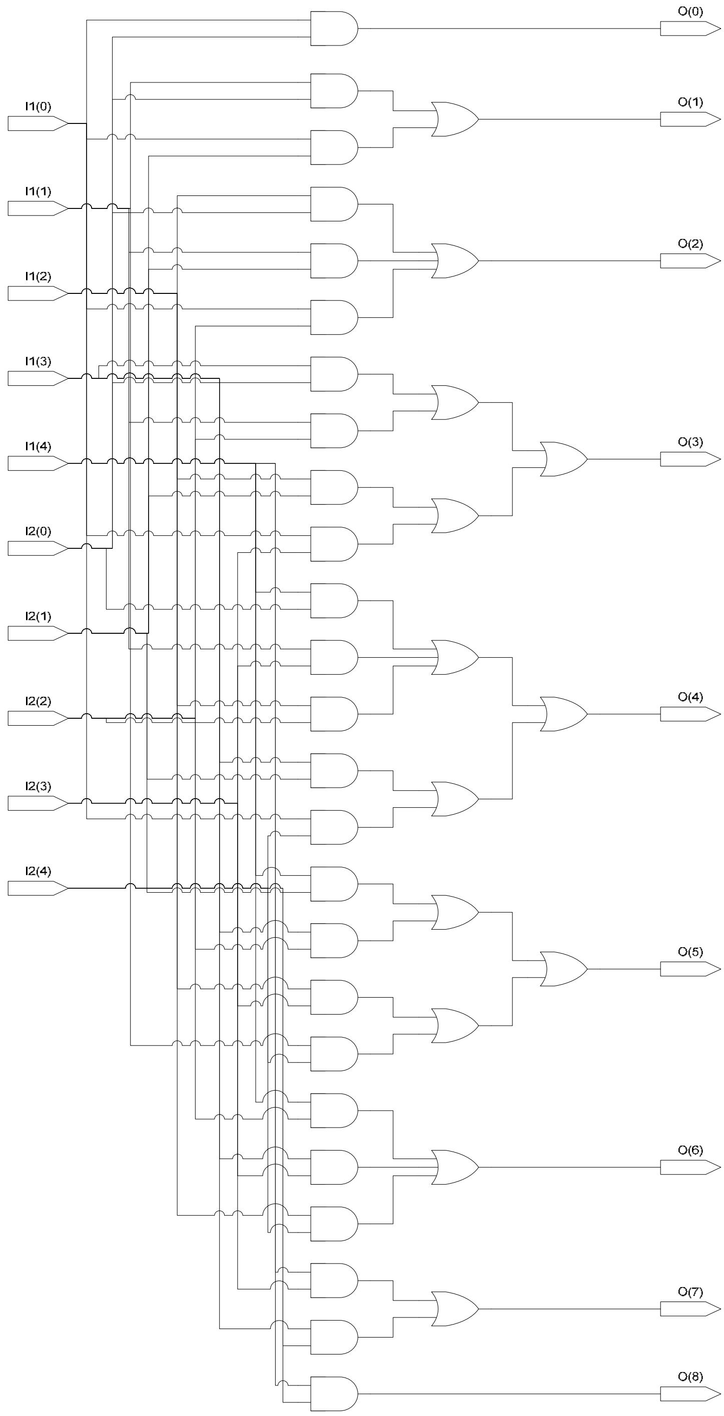


Figure 52 Schematic of the "SecondStage" Circuit

Finally the inputs to the third stage are the two 9-bit vectors produced by the previous stage. The circuit named ThirdStage has one bit as output, signalling the selected candidate. This bit is set to 1 if the total number of ones (represented by the two 9-bit vectors as described above) is greater than or equal to 8. The circuit just described implements the following Boolean equations, where *O* is the output bit and *I1* and *I2* are the two 9-bit input vectors, and its schematic is shown in Figure 53.

$$F0 = I1(8)$$

$$F1 = I1(7) \cdot \bar{I}2(0)$$

$$F2 = I1(6) \cdot \overline{(I2(0) + I2(1))}$$

$$F3 = I1(5) \cdot \overline{(I2(0) + I2(1) + I2(2))}$$

$$F4 = I1(4) \cdot \overline{(I2(0) + I2(1) + I2(2) + I2(3))}$$

$$F5 = I1(3) \cdot (I2(7) + I2(6) + I2(5))$$

$$F6 = I1(2) \cdot (I2(7) + I2(6))$$

$$F7 = I1(1) \cdot I2(7)$$

$$F8 = I2(8)$$

$$O = F0 + F1 + F2 + F3 + F4 + F5 + F6 + F7 + F8$$

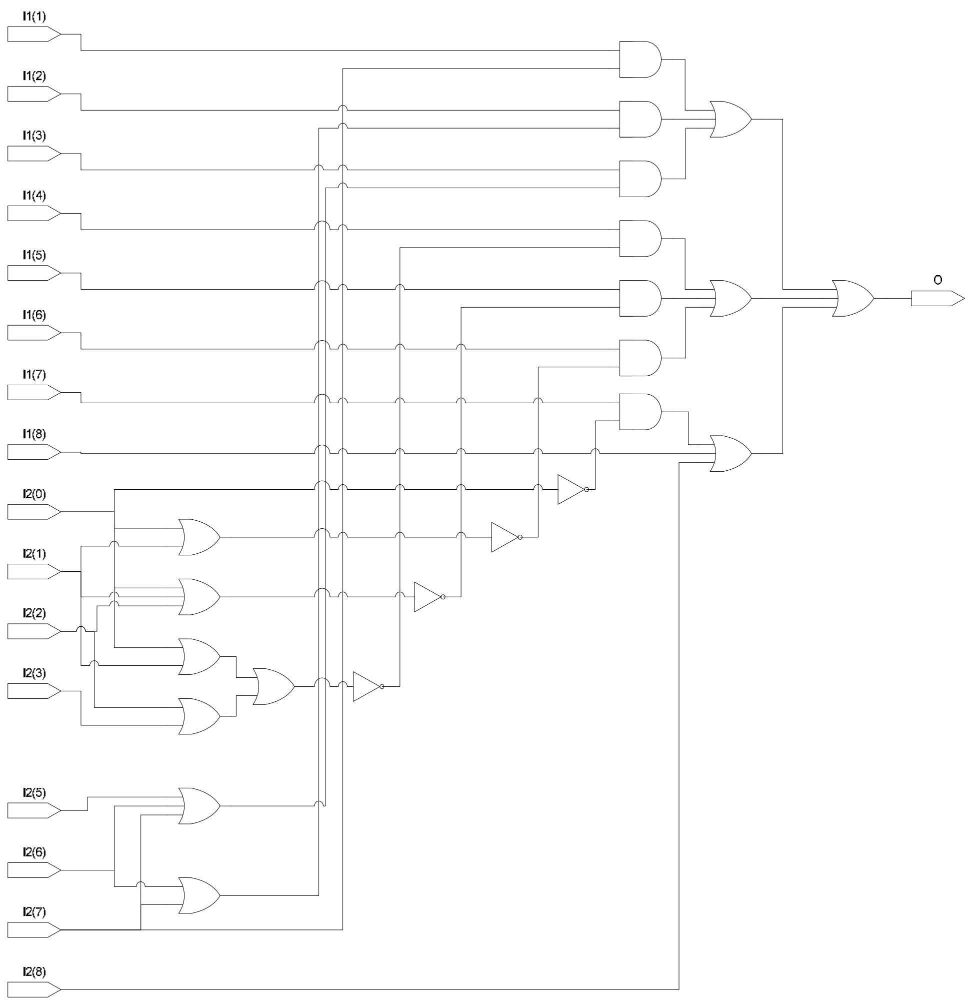


Figure 53 Schematic of the "ThirdStage" Circuit

5.2.2. *The architecture of the SGV*

The JM reference software of the H.264 encoder indicates that the criterion for the best candidate among the various macroblock predictors is not to minimize SAD but to minimize the sum of SAD and bit-rate cost. The first implementation of our approach is incompatible with this criterion, which requires a quantitative approach. In order to solve this problem a new metric was to created, based on the concept of comparisons, which replaces SAD. This new metric, the Sum of Greater Values (SGV), was described in detail in Chapter 3. Each candidate produces an SGV when compared to another candidate and the optimality criterion is to minimize the sum of SGV and bit-rate cost.

As already mentioned, the proposed algorithm in this last form is a variation of the original one. So, as described in the previous section, the algorithm compares the absolute differences of all the 4x4 prediction blocks, but in this implementation, instead of selecting the one with the largest number of minimum values, it produces the Sum of Greater Values for each candidate. Furthermore, SGV refers to values that are only greater, as opposed to greater or equal in the original approach, and does not consider equality. This means that the architecture presented in the previous section requires some modifications in order to produce the SGV for both candidates and to exclude the equality. The modified architecture has two 128-bit vectors as inputs (each for an entire 4x4 block of 8-bit numbers that are the absolute differences of the candidate and the current block), and two 5-bit numbers as outputs that represent the SGVs of each candidate.

The comparison between the absolute difference i of the candidate j and the corresponding absolute difference i of the candidate k is done by a specialized circuit, the block diagram of which is shown in Figure 54. The concept is similar to the one presented in the block diagram of Figure 47. The only difference is that the final 3-bit Comparator produces one bit for each candidate and not one for both of them.

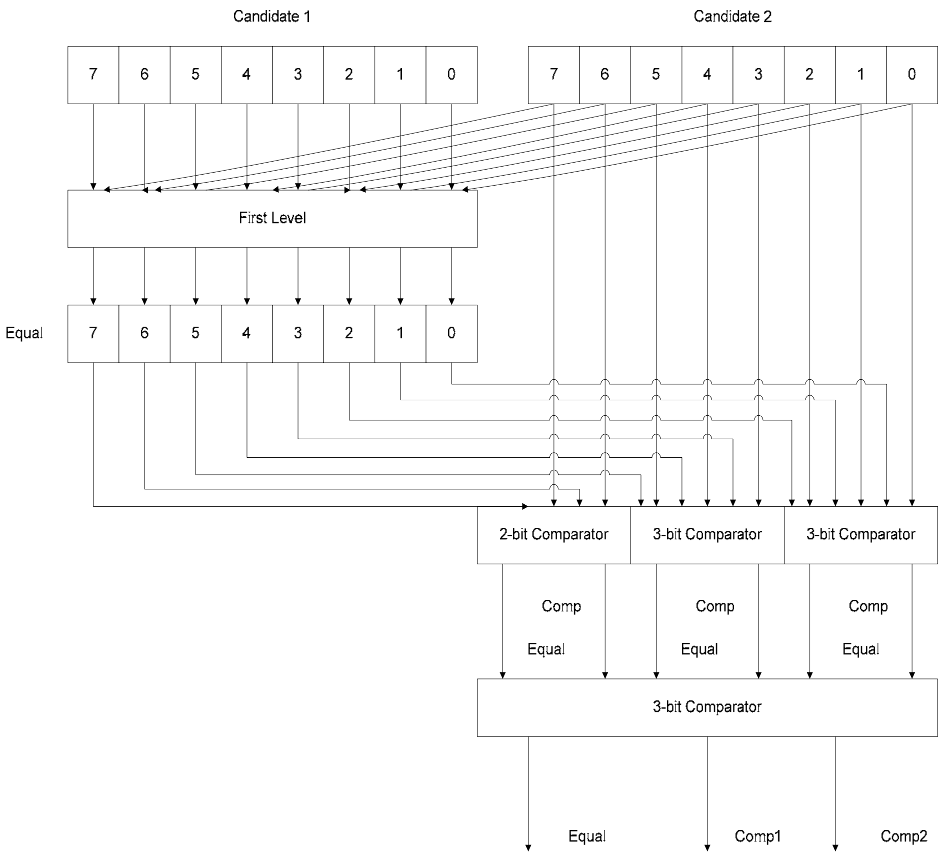


Figure 54 Block Diagram of the new 8-bit Comparator (which has an additional output compared to the one presented in Figure 47)

As already mentioned, this algorithm does not take into account the case of equality, as does its former version, where the equality case was favouring of the first candidate. That means that the comparators presented in Figure 54 are not the same with the ones presented in Figure 47, as the latter gives us 1 if the value of the second candidate was greater than or equal to value of the first one.

The three-bit comparator works similarly with the one presented in the previous section with the only difference the conditions that must apply in order to set the bit *comp*. More precisely, if the third bit of the second candidate is set to 1 and, at the same time, the third bit of *equal* is set to 0, then the number represented by the second candidate is greater than the first one. The same applies when the third bit of *equal* and the second bit of the second candidate are set to 1 and, at the same time the second bit of *equal* is set to 0. When the third and second bit of *equal* and the first bit of the second candidate are set to 1, then if the first bit of *equal* is set to 0 the second candidate will be greater than the first one, else if it is set to 1 then the two candidates are equal. In the same sense works the two-bit comparator. The truth table that shows the encoding of the output of the 3-bit Comparator is shown in Table 30, while the equivalent truth table for the 2-bit Comparator is shown in Table 31. The schematics of the 3-bit Comparator and the 2-bit Comparator are shown in Figures 55 and 56, respectively.

Table 30 Truth Table of the new 3-bit Comparator

Cand2 (2)	Cand2 (1)	Cand2 (0)	Equal (2)	Equal (1)	Equal (0)	Comp	Equal
1	X	X	0	X	X	1	0
X	1	X	1	0	X	1	0
X	X	1	1	1	0	1	0
X	X	1	1	1	1	0	1

Table 31 Truth Table of the new 2-bit Comparator

Cand2 (1)	Cand2 (0)	Equal (1)	Equal (0)	Comp	Equal
1	X	0	X	1	0
X	1	1	0	1	0
X	1	1	1	0	1

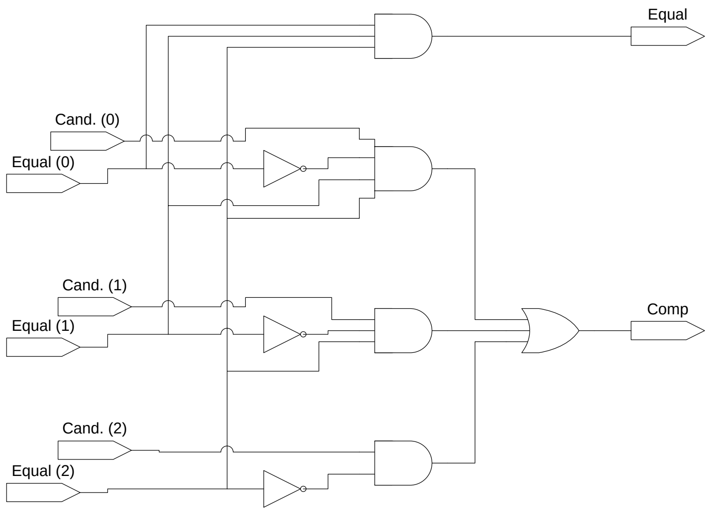


Figure 55 Schematic of the new 3-bit Comparator

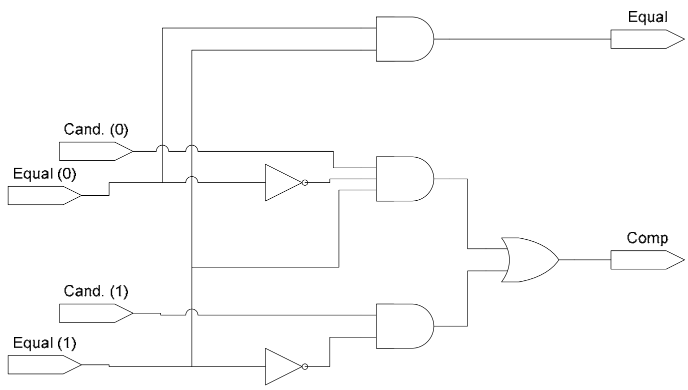


Figure 56 Schematic of the new 2-bit Comparator

As one can notice in Figure 54 the final 3-bit comparator produces 3 bits, one for each candidate and one for the equality case, and not just 2 as the comparator just described. This is done by checking the case when the first candidate is greater. That is, when the second candidate is not greater and at the same time the two candidates are not equal, i.e. when both the *equal* and the *comp2* bits are set to 0. The truth table that shows the encoding of the output of the 3-bit Comparator is shown in Table 32, and its schematic is shown in Figure 57. It must be noted that in the truth table is presented the encoding for the output *comp2* and *equal. T*he output *comp1* will be 1 in all the other cases, which are not present in the truth table.

Table 32 Truth Table of the last 3-bit Comparator

Cand2 (2)	Cand2 (1)	Cand2 (0)	Equal (2)	Equal (1)	Equal (0)	Comp2	Comp1	Equal
1	X	X	0	X	X	1	0	0
X	1	X	1	0	X	1	0	0
X	X	1	1	1	0	1	0	0
X	X	1	1	1	1	0	0	1

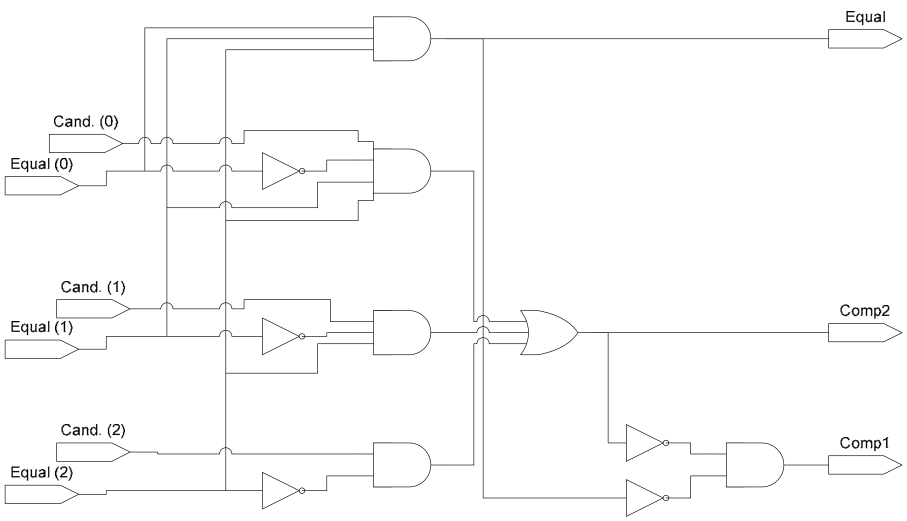


Figure 57 Schematic of the last 3-bit Comparator

The architecture described above compares two 8-bit numbers and outputs seperataly for each candidate if it is greater than the other, and if they are equal. For a 4x4 block, which is the basic size of the blocks used in Motion Prediction, we have a total of 16 8-bit numbers for each candidate. Next we calculate the number of greater values for each candidate. This sum is the value for the new metric, SGV. The circuit that counts the total number of greater values for each candidate is quite different from that presented in Section 5.2.1 and its block diagram is shown in Figure 58.

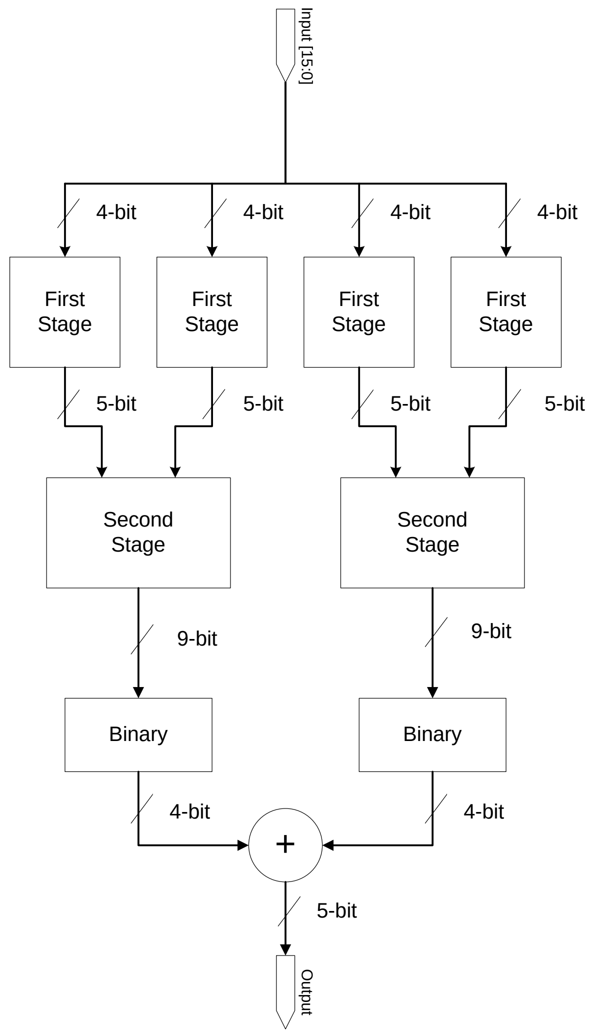


Figure 58 Block Diagram of the SGV generator circuit

One of the main differences between the two circuits is that, although the original one produced one result for both the candidates, the one presented in this section, produces one SGV for each candidate. The comparison among the 16 absolute differences of the two candidates results in two 16-bit vectors, one for each candidate, and not just one, as was the case in the circuit implemented for the first approach of the algorithm. Each one of these two 16-bit vectors is the input to the circuit that counts the number of greater values, i.e. the number of ones present in each vector. Therefore, in this version of the algorithm, we need to implement two circuits like the one presented in Figure 58.

The SGV generator circuit works as follows. It has a 16-bit vector as input, which indicates with "1" which of the 16 absolute differences of the examined candidate is greater than the corresponding absolute difference of the other candidate. Our goal is to count the ones presented in this vector. First, we divide the 16-bit vector into 4 quadruplets. For each of these quadruplets we count the number of ones, using the "FirstSatge" circuit presented in detail in the previous section. This circuit has a 5-bit vector as output, which has exactly one "1", and the total number of ones in the input circuit is indicated by the position of the "1". For example, if the first bit of the output is a "1", then the input vector had no ones, if the second bit is a "1", then the input vector had one "1", and so on.

In the next step we merge the outputs of two "FirstStage" circuits in order to count the number of ones in one octet. For each of the two octets in the 16-bit vector, we count the number of ones, using the "SecondSatge" circuit presented in detail in the previous section. This circuit has a 9-bit vector as output, which has exactly one "1", and its position indicates the total number of ones of the input circuit, as was the case of the output of the "FirstStage".

The final step is to merge the two octets, in order to count the number of ones in the 16-bit vector. We had to select between two implementations in order to achieve this goal. The first was to follow the same concept, as in Section 5.2.1, that is to merge the two outputs of the two "SecondStage" circuits, and, by using a "ThirdStage" circuit, produce a 17-bit vector, which would have exactly one "1", and indicating the total number of ones in the input circuit. After this vector has been produced, we convert the number it indicates into its binary representation, which is done by a "Binary" circuit, similar to the one presented in Figure 58.

Nevertheless, the "ThirdStage" circuit needed for this implementation is a circuit which is over two times more complicated than the "SecondStage". To be more precise this circuit should implement the following Boolean equations, where *O* is the 17-bit output vector and *I1* and *I2* are the two 9-bit input vectors. Furthermore, it would be necessary to convert the output to its binary representation, so least one "Binary" circuit would be required.

$$O(0) = I2(0) \cdot I1(0)$$

$$O(1) = (I1(1) \cdot I2(0)) + (I2(1) \cdot I1(0))$$

$$O(2) = (I1(2) \cdot I2(0)) + (I1(1) \cdot I2(1)) + (I2(2) \cdot I1(0))$$

$$O(3) = (I1(3) \cdot I2(0)) + (I1(1) \cdot I2(2)) + (I1(2) \cdot I2(1)) + (I2(3) \cdot I1(0))$$

$$O(4) = (I1(4) \cdot I2(0)) + (I1(3) \cdot I2(1)) + (I1(2) \cdot I2(2)) + (I1(1) \cdot I2(3)) + (I2(4) \cdot I1(0))$$

$$O(5) = (I1(5) \cdot I2(0)) + (I1(4) \cdot I2(1)) + (I1(3) \cdot I2(2)) + (I1(2) \cdot I2(3)) + (I2(4) \cdot I1(1) + (I2(5) \cdot I1(0))$$

$$O(6) = (I1(6) \cdot I2(0)) + (I1(5) \cdot I2(1)) + (I1(4) \cdot I2(2)) + (I1(3) \cdot I2(3)) + (I1(2) \cdot I2(4)) + (I2(5) \cdot I1(1) + (I2(6) \cdot I1(0))$$

$$O(7) = (I1(7) \cdot I2(0)) + (I1(6) \cdot I2(1)) + (I1(5) \cdot I2(2)) + (I1(4) \cdot I2(3)) + (I1(3) \cdot I2(4)) + (I1(2) \cdot I2(5)) + (I2(6) \cdot I1(1) + (I2(7) \cdot I1(0))$$

$$O(8) = (I1(8) \cdot I2(0)) + (I1(7) \cdot I2(1)) + (I1(6) \cdot I2(2)) + (I1(5) \cdot I2(3)) + (I1(4) \cdot I2(4)) + (I1(3) \cdot I2(5)) + (I1(2) \cdot I2(6)) + (I2(7) \cdot I1(1) + (I2(8) \cdot I1(0))$$

$$O(9) = (I1(8) \cdot I2(1)) + (I1(7) \cdot I2(2)) + (I1(6) \cdot I2(3)) + (I1(5) \cdot I2(4)) + (I1(4) \cdot I2(5)) + (I1(3) \cdot I2(6)) + (I2(7) \cdot I1(2) + (I2(8) \cdot I1(1))$$

$$O(10) = (I1(8) \cdot I2(2)) + (I1(7) \cdot I2(3)) + (I1(6) \cdot I2(4)) + (I1(5) \cdot I2(5)) + (I1(4) \cdot I2(6)) + (I2(7) \cdot I1(3) + (I2(8) \cdot I1(2))$$

$$O(11) = (I1(8) \cdot I2(3)) + (I1(7) \cdot I2(4)) + (I1(6) \cdot I2(5)) + (I1(5) \cdot I2(6)) + (I2(7) \cdot I1(4) + (I2(8) \cdot I1(3))$$

$$O(12) = (I1(8) \cdot I2(4)) + (I1(7) \cdot I2(5)) + (I1(6) \cdot I2(6)) + (I1(5) \cdot I2(7)) + (I2(8) \cdot I1(4))$$

$$O(13) = (I1(8) \cdot I2(5)) + (I1(7) \cdot I2(6)) + (I1(6) \cdot I2(7)) + (I2(8) \cdot I1(5))$$

$$O(14) = (I1(8) \cdot I2(6)) + (I1(7) \cdot I2(7)) + (I2(8) \cdot I1(6))$$

$$O(15) = (I1(7) \cdot I2(8)) + (I2(7) \cdot I1(8))$$

$$O(16) = I2(8) \cdot I1(8)$$

The second implementation is the one presented in Figure 58. In this implementation, we convert the 9-bit outputs of the two “SecondStage” circuits to their binary representation by the “Binary” circuit, i.e. two 4-bit vectors which will take any value from 0 to 8. In order to find the total number of ones of the 16-bit input vector we just add these two 4-bit vectors, and their sum will be the Sum of Greater Values (SGV) for the examined candidate.

This implementation is far simpler and smaller in size than the former approach, although we use two “Binary” circuits, instead of one, and one 4-bit adder. This can be clearly seen if we examine the structure of the “Binary” circuit. This circuit takes a 9-bit vector as input, which has only one “1”, the position of which indicates a number, and has a 4-bit output which is the binary representation of the number indicated by the position of the “1” in the input vector. The Boolean equations that implement the aforementioned process are the following, where *O* is the 4-bit output vector and *I* is the 9-bit input vector. The schematic of the circuit produced by these equations is shown in Figure 59.

$$O(0) = I(1) + I(3) + I(5) + I(7)$$

$$O(1) = I(2) + I(3) + I(6) + I(7)$$

$$O(2) = I(4) + I(5) + I(6) + I(7)$$

$$O(3) = I(8)$$

The “Binary” circuit requierd for the first implementation would work in the same manner as the one just presented but it would be larger and more complicated as it would have a 17-bit vector as input instead of a 9-bit one, and a 5-bit vector as output instead of a 4-bit one. So, it is now clear that the implementation of the SGV generator using the first approach with the “ThirdStage” circuit and the “Binary” circuit just mentioned would result in a much more complicated and larger in size circuit than the implementation with the 4-bit adder and two of the “Binary” circuits shown in Figure 59.

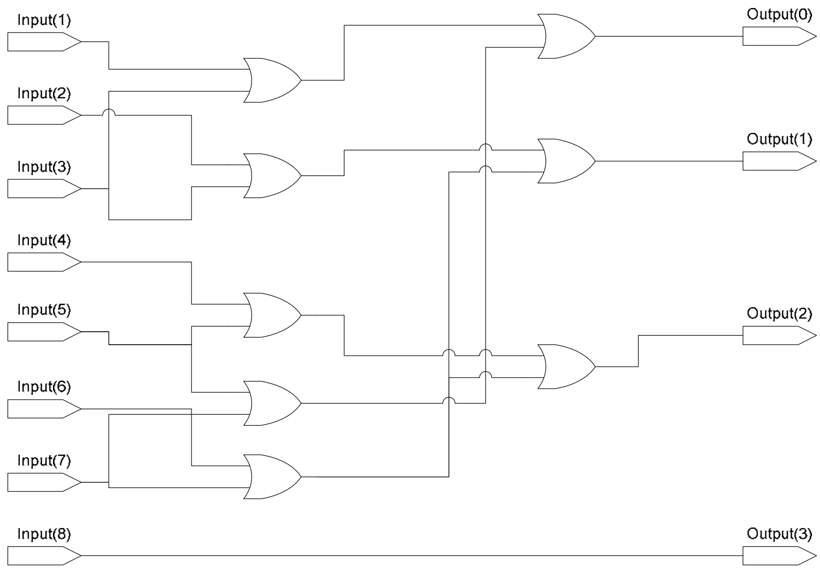


Figure 59 Schematic of the “Binary” circuit

5.3. Intra Prediction

5.3.1. The architecture for Intra Prediction

One of the main claims of this work is the replacement of SAD in the Macroblock Prediction process of video encoding according to the H.264 standard, both in Intra and Inter Prediction, each of which comprises numerous features, as described in detail in Chapter 2. At the beginning of this work it was important to find the simplest framework in which we could replace the SAD and use the new approach. This framework was the Intra Prediction of the 4x4 luma blocks, with no rate-distortion optimization aspects taken into account.

Intra coding refers to the case where only spatial redundancies within a video picture are exploited. In order to increase the efficiency of the intra coding process in H.264, spatial correlation between adjacent blocks within a given frame is also exploited. The idea is based on the observation that adjacent blocks tend to have similar properties. Therefore, as a first step in the encoding process, for a given macroblock, would be to predict the block of interest from its surrounding blocks, typically the ones located on top and to the left of the block of interest, since those blocks would have already been encoded. After a prediction block P is formed, based on previously encoded and reconstructed blocks, it is subtracted from the current block prior to encoding. The H.264 standard offers a rich set of prediction patterns for Intra prediction, among them nine prediction modes for the 4x4 luma blocks, each of which has its own direction of prediction and the predicted samples are obtained from a weighted average of decoded values of neighbourhood blocks The encoder typically selects the prediction mode, for each block, which minimises the difference between P and the block to be encoded. The selection is done by using SAD which indicates the magnitude of the absolute error.

Based on the concept that the encoder typically selects the prediction mode that minimises the difference between P and the block to be encoded, the idea that a qualitative approach may give the same results as a quantitative one was conceived. This resulted in the first implementation of the algorithm and the architecture presented in Section 5.2.1. The next step was to implement a circuit that would perform Intra

Prediction for the 4x4 luma blocks as described in the previous paragraph, using the aforementioned architecture and to compare it with an equivalent one which would use SAD.

Such a circuit would have the absolute differences of the nine prediction blocks and the block to be encoded as inputs and would select which one of these nine prediction blocks is the best. The block diagram of this circuit is shown in Figure 60, where the "mode i" blocks refer to the absolute differences for each prediction mode, and the "comparator" blocks implement the architecture of the first approach of the algorithm as it is described in Section 5.2.1.

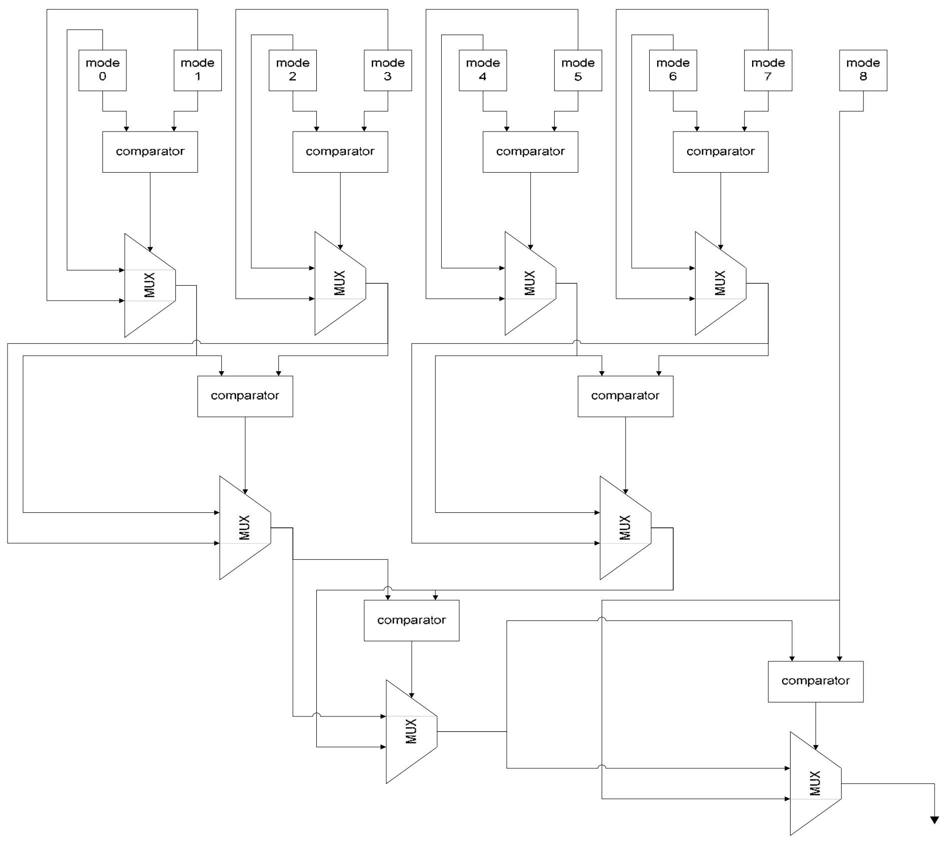


Figure 60 Block Diagram of an Intra Prediction circuit

The input word-lengths of the circuit are 128 bits (for an entire 4x4 block of 8-bit numbers), which is consistent with common video standards, and are indicated as mode i, where i is in the range $0 \leq i \leq 8$. This circuit can compare 9 4×4 blocks at a time and works as follows. At the first stage we create pairs of two successive modes and we compare the modes of each pair using the Comparator circuit, so at this stage we make 4 comparisons. The Comparator selects the mode with the largest number of minimum values. In the next, step the mode chosen from each pair is compared with the mode chosen from the next pair and the same procedure is repeated, so at this stage we make 2 comparisons. The modes selected by these two comparisons are then compared with each other and we end up with the best mode among the 8 that were initially compared. This best mode is compared with the 9th mode which has not taken part so far in the comparisons. The result of this last comparison is the best mode among the 9 available for Intra Prediction.

As we have already pointed out, the Comparator circuit, which performs the comparison among two modes, is implemented according to the architecture described in Section 5.2.1. This means that it has 16 absolute differences for each mode as inputs and for each of these 16 absolute differences, it performs a comparison using an 8-bit Comparator, the block diagram of which is shown in Figure 47. These 16 8-bit Comparators produce the 16-bit vector which is input to the Selection Circuit, the block diagram of which is shown in Figure 50, and which has as output a bit that indicates which one of the two modes compared has the maximum number of smaller absolute differences. The block diagram of the Comparator circuit is shown in Figure 61. In this block diagram the components indicated as PEi, where i is in the range $0 \leq i \leq 15$, are the 16 8-bit Comparators.

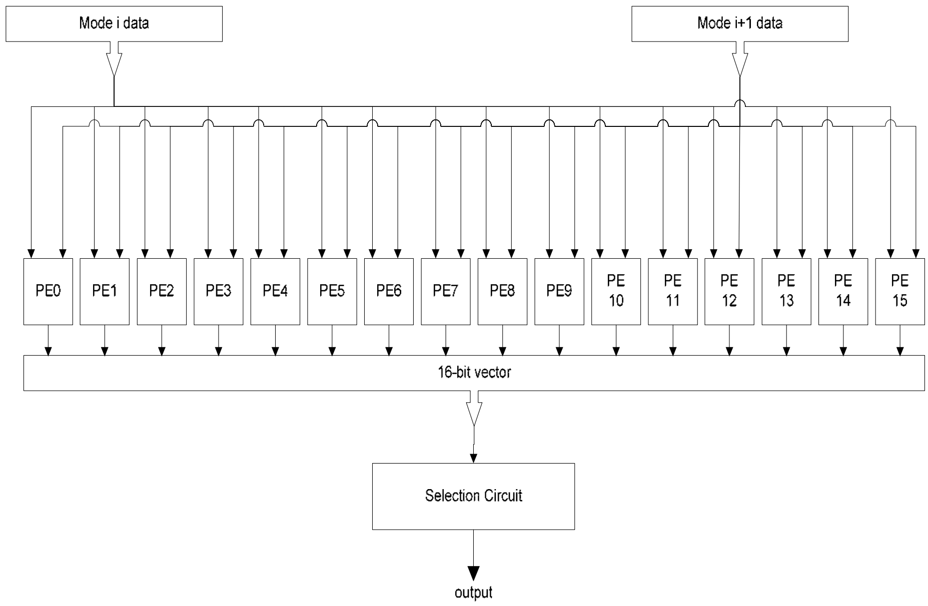


Figure 61 Block Diagram of a Comparator

Following the design of the circuit for Intra Prediction for the 4x4 luma blocks with the use of the first approach of the algorithm, we proceeded to build an equivalent circuit that would perform Intra Prediction with the use of SAD. Although there are many ways to design such a circuit, we decided to use an architecture that would be as close as possible to the one used for the circuit with the new algorithm. This decision led to a circuit with a block diagram same as the one presented in Figure 60.

The architecture of this circuit is identical to the one presented in Figure 60, except from the fact that instead of comparing the corresponding absolute differences of two successive modes we calculate their respective SAD. The circuit works as follows: In the first stage we create pairs from two successive modes and we calculate the SAD of each mode for each pair, compare them and choose the one with the smallest SAD. Thus, at this stage we calculate 8 SAD, compare them in pairs and choose the 4 best modes. The mode chosen from each pair forms a new pair with the mode chosen from the next pair and the procedure is repeated, so at this stage we calculate 4 SAD and compare in pairs and choose the two best modes. The SADs for the modes, selected by these two comparisons, are then calculated and compared to select the best mode among the 8 that were initially compared. Finally the SAD of the 9th mode is calculated and compared with the SAD of the best mode selected from the 8 previous. The result of this last comparison is the best mode among the 9 available for Intra Prediction.

Nevertheless, although the two circuits may have the same block diagram, the block named Comparator in Figure 60, has a different function, and therefore different structure, in order to implement what is needed for the modified algorithm to work. The new block diagram of the Comparator block is shown in Figure 62. It must be noted here that the comparator circuit presented in this block diagram the standard 12-bit comparator. The reason for choosing a standard comparator was to make the comparison of the two circuits more objective.

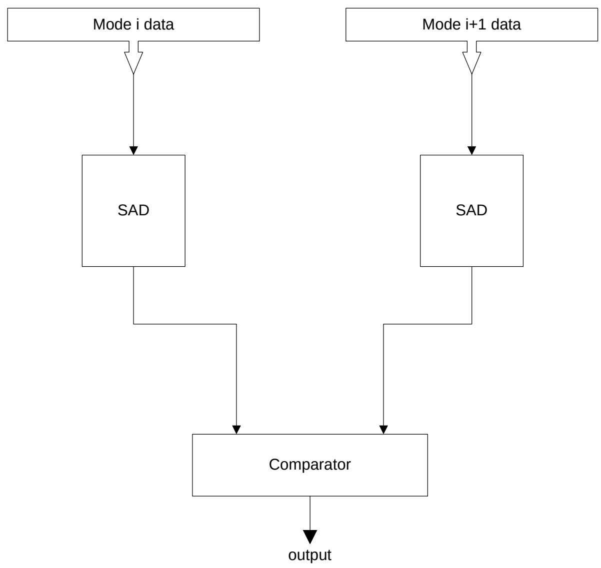


Figure 62 Block Diagram of the new Comparator circuit for SAD

The SAD block presented in Figure 62 is a circuit which calculates the SAD for a mode. It consists of an adder tree as shown in Figure 63.

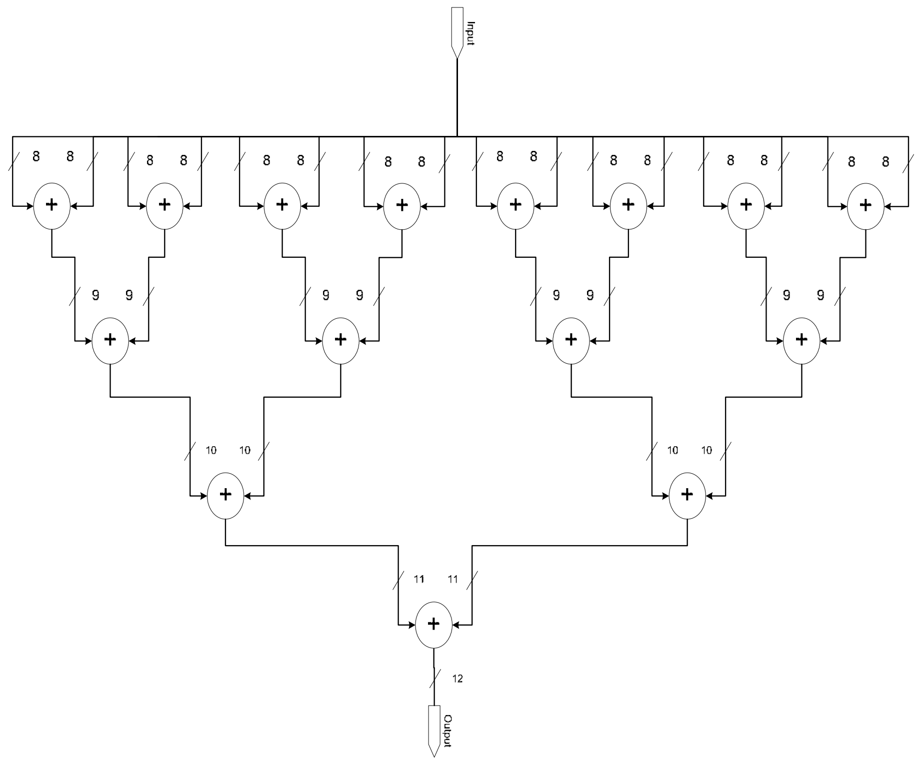


Figure 63 Block Diagram of the SAD circuit

As already noted, there are also other architectures for the Intra Prediction circuit using SAD, many of them are more efficient than the one presented in this section. For example, one could reuse the SAD of each mode selected in every step instead of recalculating it, with the assistance of registers that would keep the SAD of each of the nine modes. And that is the case for most of the architectures used to implement Intra Prediction. Nevertheless, the intent of this implementation was to build a circuit as similar as possible with the one of the new algorithm, in order to compare the two algorithms as objectively as possible.

5.3.2. Implementation and performance analysis

The implementation using SAD and the one using the new algorithm have been captured using VHDL and synthesized to allow comparisons of delay and power on a common reference. They have both been implemented on a 130nm CMOS technology (1.08V) using the UMC standard cell library. The architectures have been simulated using the Mentor Graphics ModelSim® SE 6.0 and synthesized using the Synopsys Design Compiler®, while the power estimates for the two circuits were obtained by Synopsys PrimePower®.

In order to assess the power-performance characteristics of both implementations we proceeded to form their respective power-delay curves, which are shown in Figure 84. The data points for the power-delay curves were determined as follows. At the begging, after testing various timing constraints, we found the best delay possible for each circuit. This process meant that, for every timing constraint we were testing, we synthesized each circuit with Synopsys Design Compiler®, which also provided the delay for each case. The best delay for each circuit was the minimum for which Design Compiler could synthesize the circuit without producing negative slack. After finding the best delay for each circuit we had to obtain the data points for their power – delay curves. For each data point we had to synthesize the corresponding circuit with Synopsys Design Compiler® with a new timing constraint, starting from the best delay possible, and relaxing the timing requirement by 1ns at a time. Each synthesized circuit was run through by Synopsys PrimePower® to calculate the average power dissipation. The operating frequency, needed by PrimePower® to calculate the power dissipation was set, each time, according to the maximum delay of the synthesized circuit, as this was reported by Design Compiler®.

The curves obtained by the aforementioned process are presented in Figure 64, from where it can be readily concluded that the new algorithm and its implementation outperforms the SAD approach in both power and performance, as its power-delay curve lies below and to the left of the one for SAD.

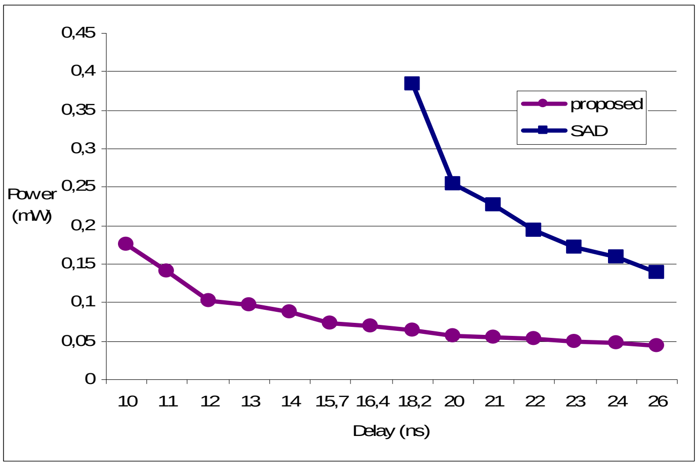


Figure 64 Power-Delay Curves for Intra Prediction with SAD and the Proposed algorithm

A careful look at the two power-delay curves intimates that the power reduction that was achieved ranges from 6X at 18.15ns to 3X at 26ns. The observed reduction is due to the reduced complexity of both the prediction algorithm and its circuit implementation. The power reduction at the best delay for both approaches is over 2X in favour of the proposed approach over SAD.

A more detailed analysis reveals that the design created contains less than 59k and can operate at frequencies of up to 100 MHz. The best delay for SAD that Design Compiler® could achieve was 18.15ns vs. 10ns for the proposed implementation, a 45% reduction, whereas the best area for SAD was 132k vs. 58k for the proposed algorithm, a 56% reduction. The above data were obtained by the synthesis results produced by Design Compiler® from the synthesis of the two circuits at their best delay, and are summarized in Table 33

.

Table 33 Performance

Architecture	Performance		
	Critical Path (ns)	***Max Clk Frequency (MHz)***	***#Cells***
SAD	18.15	55	132356
Proposed Algorithm	10	100	58100

In order to complete the performance analysis of the proposed architecture we need to find out up to which video format the presented circuit can support. For a frame with size of W x H pixels, N prediction modes and frame rate of F fps, where W is the frame's width, H is the frame's height, the processing requirements (in units of b/s) are given by the equation:

$$TpB = \frac{W}{16} \times \frac{H}{16} \times N \times F \times 16$$

According to above equation, the time needed to find the best Intra predictor for one 4x4 block for an SHDTV video sequence (1920x1080, 60 fps) and 9 prediction modes is 14,3 ns, which is more than the time needed (10 ns) for the proposed architecture to choose the prediction mode for 4x4 block.

5.4. Inter Prediction

5.4.1. The architecture for Inter Prediction

The implementation of two circuits that perform Intra Prediction, one using SAD and the other using the first approach of the new algorithm, showed clearly that working in a frame, where the comparisons among corresponding absolute differences give the basis of the criterion for choosing among various candidate predictors, leads to hardware implementations with considerably higher efficiency than implementations where the criterion for the best candidate is based on the addition of the absolute differences. This was a confirmation for the correctness of the basic idea of this work, which was no other than the replacement of SAD with a new metric based on comparisons, which leaded to the extension of the first form of the algorithm in order to be used in Inter Prediction.

Inter Prediction refers to the case where the temporal model is used in the coding process. In this case the predicted frame is created from one or more past or future frames ('reference frames'). The process of finding the best predicted frame is known as Motion Estimation and is analysed in details in Chapter 2. The accuracy of the prediction can usually be improved by compensating for motion between the reference frame(s) and the current frame (Motion Compensation). A practical and widely-used method of motion compensation is to compensate for movement of rectangular sections or 'blocks' of the current frame. H.264 is a block-based motion-compensated hybrid transform codec, which supports a variety of block sizes (denoted as modes), varying from 16×16, 8×16, 16×8, 8×8, 8×4, 4×8 to 4×4 pixels. A separate motion vector is required for each partition or sub-macroblock, so we have a total of 41 MVs for each macroblock. Usually the criterion to find the matching block is the energy in the residual formed by subtracting the candidate block from the current M×N block, and the candidate region that minimizes the residual energy is chosen as the best match. However, in order to reduce the computational complexity, most real world application, among them H.264, uses the sum of absolute differences (SAD). Furthermore the H.264 reference software, when calculating the motion cost, takes also into account the bitrate cost for the motion vector. So, in H.264 the criterion to find the matching block is the motion cost, a combination of SAD and bit-rate cost.

The presence of the bit-rate cost in the criterion for finding the best predictor required the creation of a new metric, based on the comparison model, which would behave in a similar manner to that of SAD. This new metric, named SGV, is presented in Section 3.4.2 and based on it, we propose an architecture which can support Variable Block Size Motion Estimation (VBSME).

VBSME is a new coding technique, presented in H.264, and provides more accurate predictions compared to traditional fixed block size motion estimation used by previous standards. There are many methods to support VBSME in hardware but the one used by most of the presented architectures, as for example in [74], [75], [76], [77], [78] and [79], is that of reusing the SADs of the smallest blocks, which are the blocks partitioned with the smallest block size , to derive the SADs of larger blocks. Nevertheless, the proposed architecture, because of the dependency of data occurring in each mode, cannot follow the same logic. On the contrary, it processes each mode separately, but because of the reduced execution time, and the lower power consumption of the new algorithm, the proposed architecture can process all modes in parallel. Furthermore, the proposed architecture takes into account the bit-rate cost, i.e. chooses the best candidate according to equation 19 (Chapter 3). In order to achieve the parallelism needed to implement the proposed architecture we use the modified predicted motion vectors, proposed in [78] and [79].

The core of the new algorithm is the comparison of each new candidate with the best found so far. It is clear that each mode has different selections, so it is obligatory to be processed independently. Because of this need for re-use of different data provided by the process of each mode the design consists by 7 similar blocks, one for each mode i.e. 4x4, 4x8, 8x4, 8x8, 8x16, 16x8 and 16x16, which work in parallel. In Figures 65 to 71 we present the block diagrams of these seven blocks. The proposed architecture selects the best among two candidate reference blocks for the 41 different sub-blocks of a MB in two stages. In the first stage 16 PEs are used to calculate the new metric for the two reference blocks for the 16 4x4 blocks. In the second stage the metrics of the previous stage are used to calculate the metrics for the larger blocks and the value of equation 19 (Chapter 3) for each candidate block for each one of the 41 different sub-blocks of the processing MB.

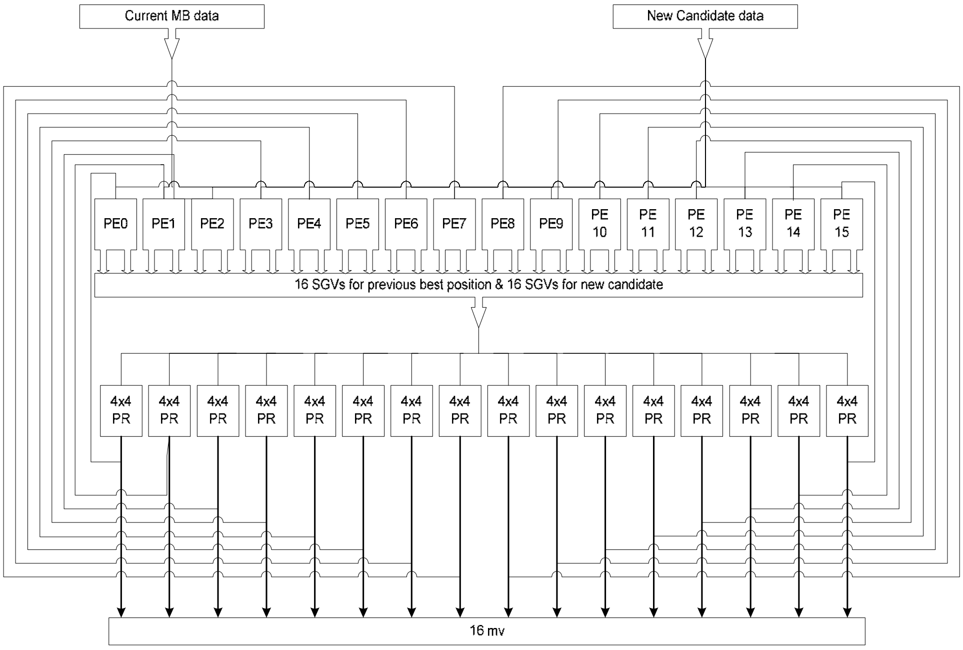


Figure 65 Block diagram for the 4x4 block size motion estimation

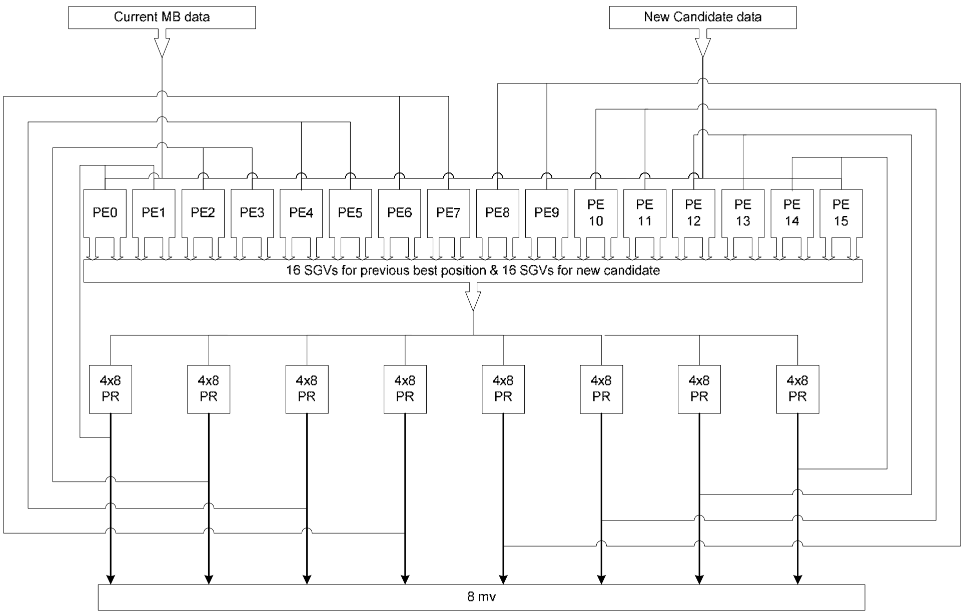


Figure 66 Block diagram for the 4x8 block size motion estimation

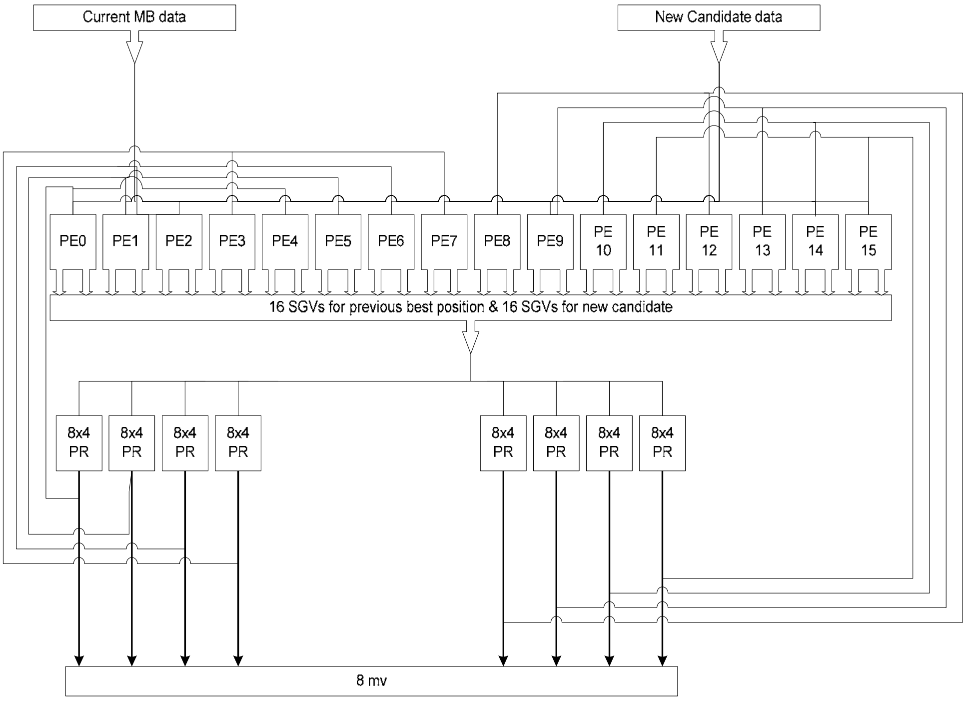


Figure 67 Block diagram for the 8x4 block size motion estimation

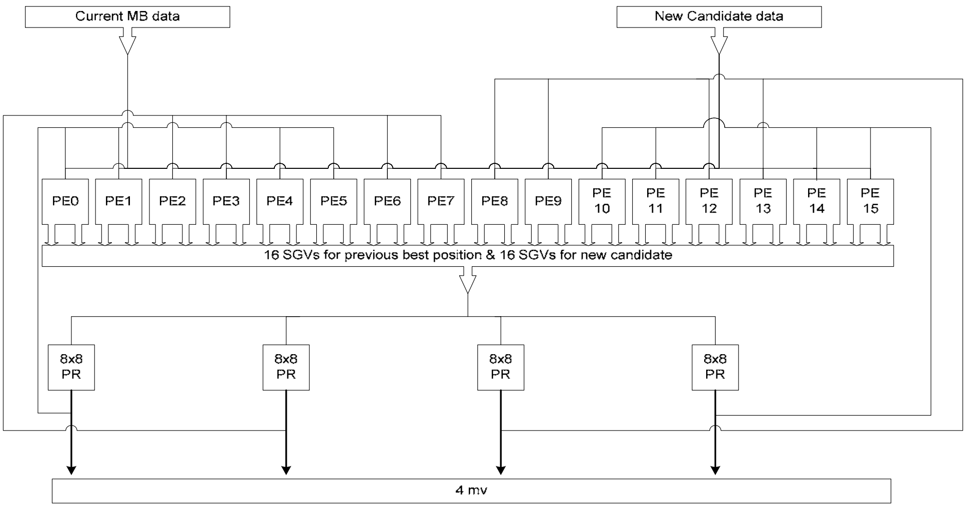


Figure 68 Block diagram for the 8x8 block size motion estimation

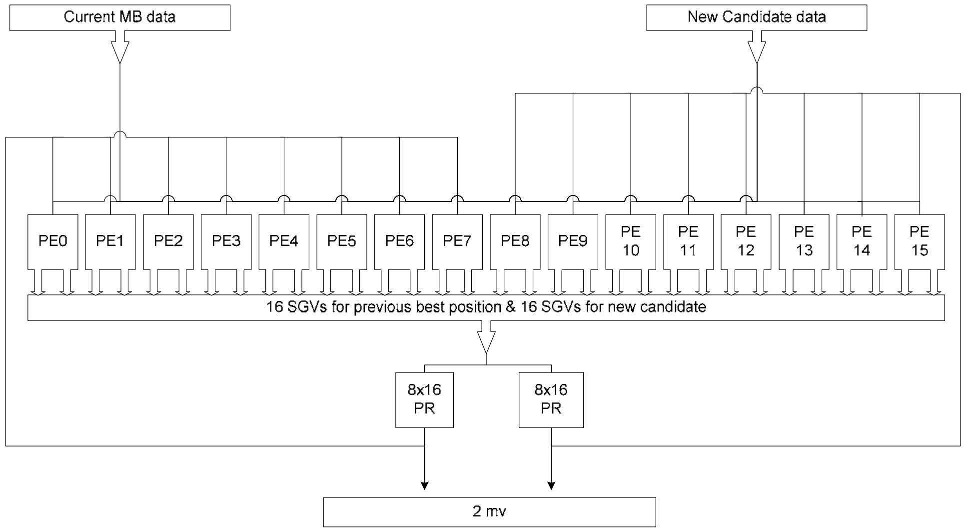


Figure 69 Block diagram for the 8x16 block size motion estimation

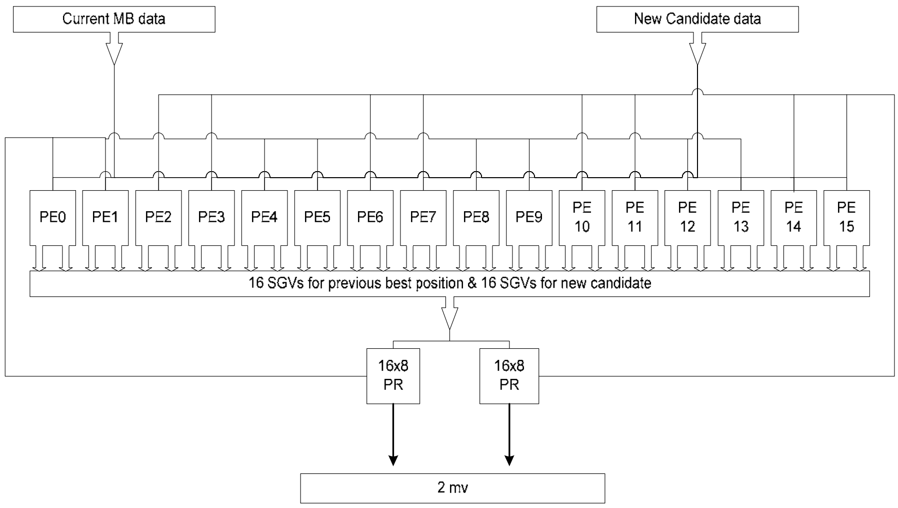


Figure 70 Block diagram for the 16x8 block size motion estimation

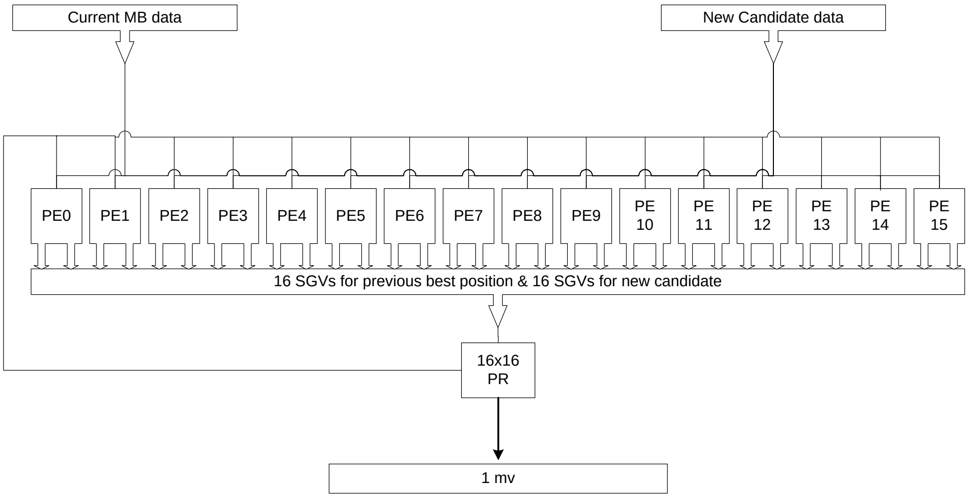


Figure 71 Block diagram for the 16x16 block size motion estimation

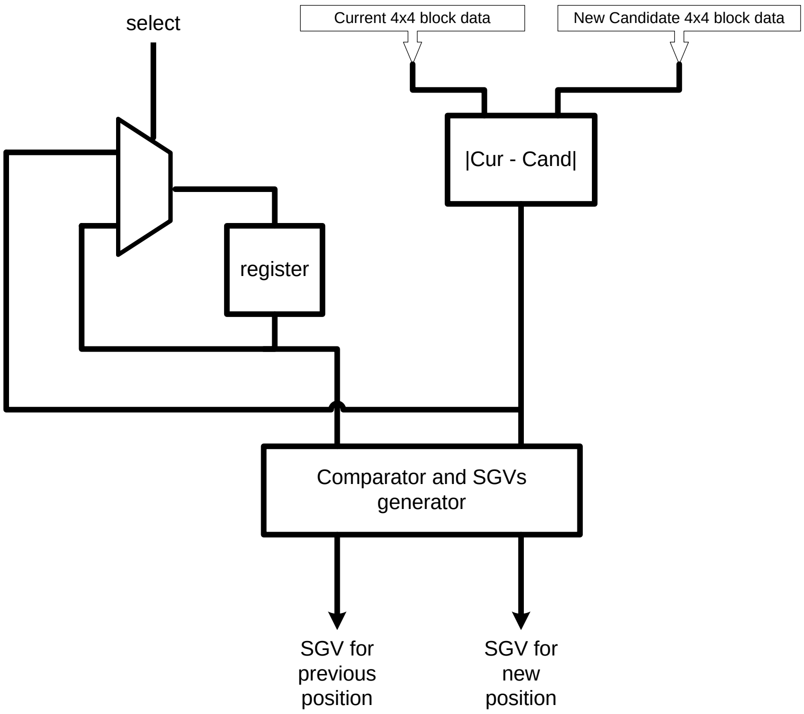


Figure 72 Block Diagram of a PE

Each PE element presented in stage 1 has as inputs 16 pixels of the current block, 16 pixels of the new candidate, and a signal which selects which of the two previous candidates was the best, as shown in Figure 72. At the beginning, the absolute differences among the pixels of current block and those of the new candidate are computed. The computation of the absolute differences is done by 16 identical circuits, the block diagram of which is shown in Figure 73.

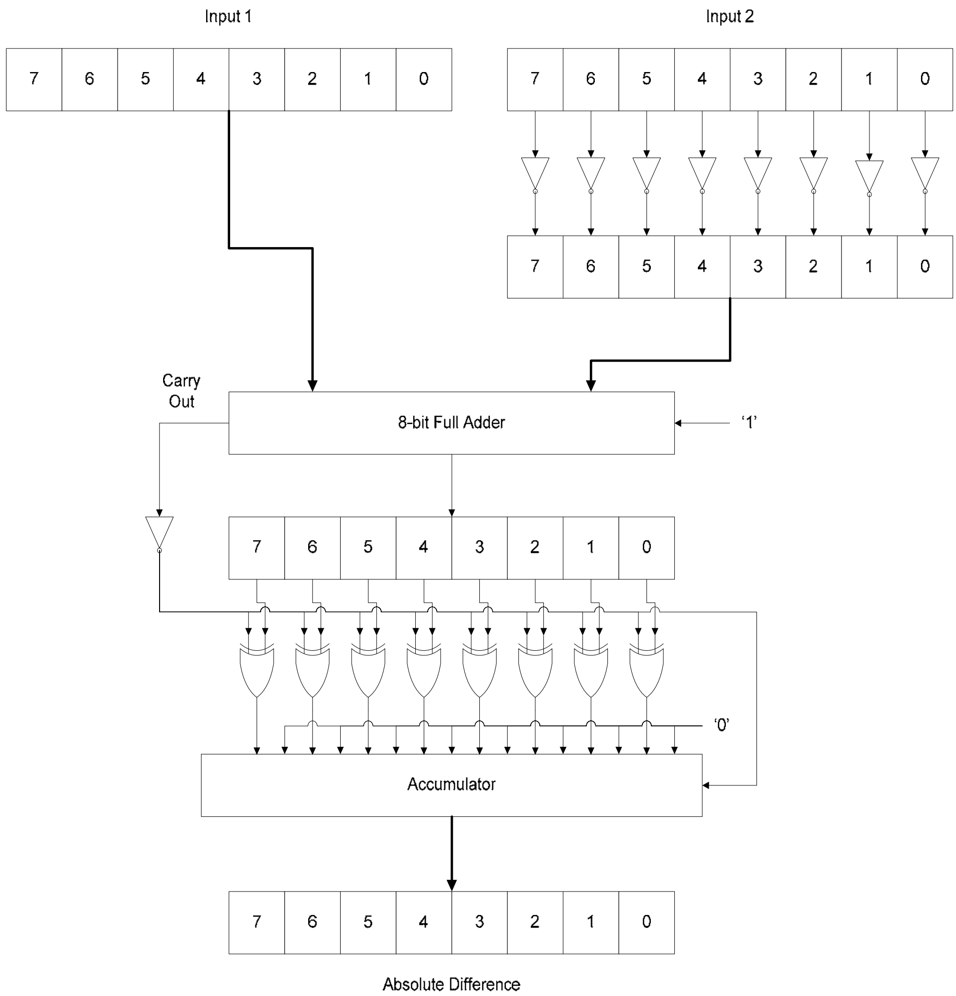


Figure 73 Block Diagram of the Absolute Difference circuit

Each absolute difference (AD), produced by the new candidate, is compared with the corresponding absolute difference produced of the previous best candidate. If the AD of candidate 1 is greater than that of candidate 2, the Sum of Greater Values (SGV) of the previous best candidate is increased by 1, else if the AD the new candidate is greater than that of candidate 1 the Sum of Greater Values (SGV) of candidate 2 is increased by 1. When all the absolute differences are compared we have the final values of the new metric (SGV) for the two candidates. The above process is done by the circuit denoted as Comparator and SGVs generator. This circuit is the implementation of the architecture of the SGV, as it is presented analytically in Section 5.2.2.

After all the 16 SGVs for each of the previous and new candidates have been calculated for all the 4x4 blocks, they are used by the modes processors in order to find the best of the two candidates for each mode. A block diagram of such a processor is shown in Figure 74. In all mode processors the inputs are the SGVs of the 4x4 blocks, as computed in stage 1 and the corresponding bit-rate cost for each candidate position. In order to calculate the SGV for the current mode, the SGVs of the appropriate 4x4 blocks are added. After the SGV for each candidate for the current mode is calculated, the corresponding bit-rate cost (shown as mv1 and mv2 in Figure 74) is added in order to have the final motion cost for each candidate. The two motion costs are compared and the one with the minimum cost is chosen.

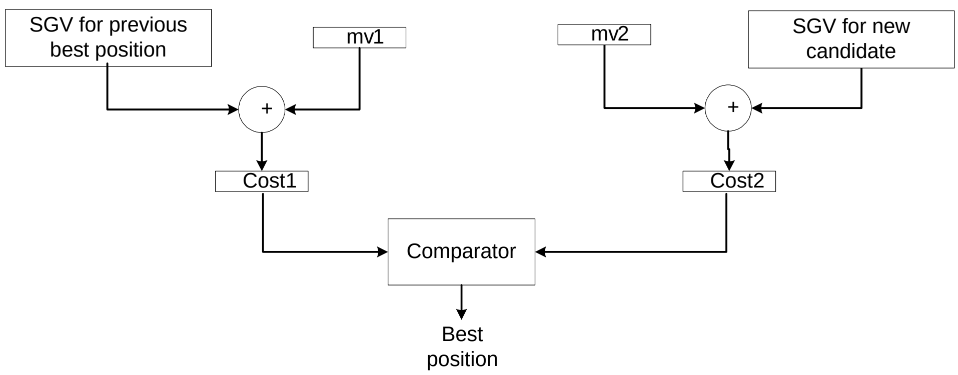


Figure 74 Block Diagram of a MxN mode processor

5.4.2. *Implementation and performance analysis*

The implementation of the architecture presented in the previous section has been captured using VHDL and synthesized with Synopsys Design Compiler® using the UMC 130nm CMOS standard cell technology (1.20V). Because the majority of the work presented in the field of architectures and implementation of Variable Block Size Motion Estimation uses standard cell technologies at their typical case, i.e. 1.20V we performed synthesis at that design corner, so, that we could fairly compare our results with other implementations.

The power estimates were obtained by Synopsys PrimePower®. As, this time the implemented circuit would not be compared with other implementations using SAD created by the author of this thesis, but by implementations presented in the bibliography, we could not proceed to the creation of power – delay curves, as the required data for the other implementations is not available. This fact changed the process of power estimation of the design under test. After the best delay for the circuit was calculated, as it is described in Section 5.3.2., we used the synthesised circuit with the best delay in order to estimate the power dissipation. Thus, this time, the power analysis concerns only a circuit synthesised at its best performance point, and not at various timing constraints, as was the former case.

Before we proceed to the presentation of the synthesis and power estimation results, we need to describe the process used by PrimePower® in order to derive the power estimation of a given circuit. PrimePower performs the following steps to accurately analyze the power of the design [80]:

1. Based on the design connectivity and wire-capacitance, PrimePower determines the transition times for all the pins within the design.

2. For average power analysis, PrimePower determines the state- and path- dependent switching for all the nodes within the design. If nets are not annotated, the propagation engine is used to determine their switching of activity.

3. PrimePower accesses the Synopsys library power tables and determines the power consumption for leaf-level cells which are summed to obtain the total design power. When performing dynamic analysis, PrimePower processes every activity in the activity file to build an accurate power profile over time and to determine the peak power consumption.

A crucial aspect for an accurate power analysis is the way the switching activity is derived. PrimePower offers a variety of methods for deriving this activity, for both dynamic and static analysis. One of them is to provide the gate-level time-based switching activity in one of the supported formats: VCD/VCD+/FSDB. These files are generated via gate-level simulation. That leads to the most accurate power analysis, supposing the VCD files contain correct data. Another method is that of the user – defined switching activity generation. Although we tried to generate the appropriate VCD files, for an accurate power analysis, that was not accomplished. Probably because of lack of timing specifications, the gate-level simulation could not provide the real timing requirements of the circuit. This meant that if we simulated the circuit in real timing constraints the circuit under test could not come to a stable state. As a consequence, the VCD files would either have false data, or would refer to a circuit working in a much lower frequency. This fact lead us to adopt the method of user – defined switching activity generation, in order to derive the power estimation results.

The performance of the proposed architecture is shown in Table 34. The proposed architecture fulfils the selection of one search position among two candidates using the VBSME in every clock cycle, i.e. in every 4.15 ns.

Table 34 The proposed VBSME chip performance

Algorithm	Variable block size full search motion estimation
Number of PE	112
Block Size	4x4, 4x8, 8x4, 8x8, 8x16, 16x8, 16x16
Technology	UMC 130nm CMOS (1.20V)
Gate Count	370k
Max Frequency	241 MHz
Power Consumption	95 mW @ 233MHz

In order to complete the performance analysis of the proposed architecture we need to identify which video format the presented circuit could support. For a frame with size of W x H pixels and search range of P_x x P_y pixels and frame rate of F fps, where W is the frame's width, H is the frame's height, P_x is the search range's width and P_y is the search range's height, the processing requirements (in units of search-positions/s) are given by the equation:

$$SP = \frac{W}{16} \times \frac{H}{16} \times ((2 \times Px) + 1) \times ((2 \times Py) + 1) \times F$$

According to this equation, the time needed to calculate one search position for an HDTV video sequence (1280x720, 60 fps) and search range 16x16 i.e. 1089 search positions, is 4,3 ns, which is more than the time needed (4.15 ns) for the proposed architecture to choose one search position.

5.5. Related work vs. proposed architecture

When this work began one could find rare information about motion estimation for H.264, and especially about variable block size motion estimation, in the bibliography. Nevertheless, as the time passed the work presented on this topic increased and at this time the related work is sufficient. In this section we discuss the performance of some of this work and compare it with the one of the proposed hardware architecture of the novel SGV algorithm.

An exact comparison between the different VBSME circuits is complicated by the fact that these have been implemented on different technologies and exhibit variations in their specifications and capabilities. A characteristic example of these variations is the correspondence between the search range and the number of search positions. The author, based on what is defined in the JM reference software of the H.264 standard and in many other sources in bibliography, assumes that if we have a search P_x x P_y then, the total number of search positions within this range is $((2 \times P_x)+1) \times ((2 \times P_y)+1)$. Nevertheless, the results presented in the majority of the related work denotes that the total number of search positions within an equivalent search range is assumed to be P_x x P_y, despite the fact that in many cases the theory presented in the work agrees with the definition of the reference software. This inconsistency, not only between different works, but even between theory and presentation of results in the same work, makes things even more complicated.

Because there is no common design goal for such applications, so as to establish a common reference for comparisons among the related work, we searched the bibliography, selected the works where a more objective and completed comparison is presented and compared our design against them.

For the comparison of different VBSME architectures L. Deng et al. [74] introduced the efficiency E, which can be expressed by the ratio of the through-put rate R and the required silicon area A of the architecture. *R* is expressed by the number of which the architecture computes the search points per second:

$$R = \frac{f}{T} \times (2P+1)^2$$

where *f* is the frequency of the architecture, *T* is the cycle number for processing one MB, and [-P, P] is the search range. To evaluate the silicon area they used the gate count *G*, thus the *E* is described as:

$$E = \frac{R}{A} = \frac{\frac{f}{T} \times (2P+1)^2}{G}$$

The unit of E is search point per second per gate. Table 35 shows the comparison between the proposed and others VBSME architectures. In all architectures, the proposed one can provide the highest computational capability.

Table 35 Comparison according to L. Deng et al

	Number of PE	Search range	Process	Block size	Frequence	Gate count	R	E
[86]	256	32x32	0.25um	4x4 to 16x16	100MHz	105k	96,215,040	916,3
[85]	256	64x64	0.18um	4x4 to 16x16	100MHz	154k	99,916,185	674,1
[84]	256	16x16	0.6um	2nx2n n>=1	72MHz	263k	76,308,480	291,2
[83]	256	48x32	0.35um	4x4 to 16x16	67MHz	105k	62,208,000	592,5
[82]	64	32x32	0.6um	8x8,16x16 32x32	60MHz	67k	15,084,748	242,1
[81]	16	32x32	0.25um	4x4 to 16x16	150MHz	71k	12,165,120	171,4
[76]	16	16x16	0.13um	4x4 to 16x16	294MHz	61k	18,247,680	299,1
[74]	256	65x65	0.18um	4x4 to 16x16	260MHz	210k	210,601,993	1002.8
[75]	256	36x35	0.18um	4x4 to 16x16	200MHz	597k	124,416,000	208,4
Proposed	112	70x70	0.13um	4x4 to 16x16	241MHz	370k	238,140,000	643,6

An other important aspect of the efficiency of an architecture for VBSME is the video format which can be supported by each implementation. C. –M. Ou [75] et al include in their work an interesting table where they present the frequency needed for their architecture and that of Yap' s [76] for various video formats. The architecture of the proposed architecture is added to this table and is shown in Table 36.

Table 36 Comparison according to C. –M. Ou et al

Frame size		QCIF (176x144)	CIF (352x288)	4CIF (704x576)	16CIF (1408x760)		SDTV (1280x720)	HDTV (1280x720)	SHDTV (1920x1080)
Frame rate (fps)		30	30	30	30	60	30	60	60
Clock rate (MHz)	[75]	0.76	3.04	12.16	48.66	97.32	27.64	55.29	123.49
	[76]	12.16	48.64	194.56	778.56	1557.12	442.24	884.64	1975.84
	Ours	0.76	3.04	12.44	32.10	64.21	27.65	55.30	124.43

As it is clear from Table 36, according to C. –M. Ou et al, the proposed architecture can process a SHDTV video sequence working at nearly half the frequency of what it can achieve. Nevertheless, C. –M. Ou assumes that the search positions, in a P_x x Py search range, are P_x x Py, which is not true.

Last but not least, we compare the power consumption of the proposed architecture with those presented in the bibliography. Although the research on this field has increased the last years, the works that present information about power consumption remain very few. We found two relevant works and compare the results presented.

Yap et al [76] presents two normalized units to determine the performance of an architecture for motion estimation. The first is the normalized power consumption, which is defined in terms of the power dissipation per macroblock (MB) per frame per second (fps). The second determines the number of frames per second (fps) that can be processed at a specific resolution. For the circuit presented in their work, these values were determined to be 0.008 mW/MB/fps and 181 fps/CIF, respectively.

Which are the corresponding values for the architecture presented in this work? Lets begin with the second normalized unit, i.e. the one that determines the number of frames per second (fps) that can be processed at a specific resolution. As presented in Table 56, the proposed architecture can process a CIF sequence at 30 fps at 3.04 MHz. This means that the clock period should be, at the most, 329ns. Nevertheless, the presented circuit can work with a clock period of 4.15ns, i.e. 79 times faster. Therefore, the proposed architecture can process 79*30 fps/CIF, i.e. 2370 fps/CIF.

To define the value for the first normalized unit we must assume, as Yap et al did, that the power dissipation is purely proportional with clock frequency, which of course is not the real case. The proposed circuit consumes 95mW at 233MHz. The frequency needed to process a QCIF sequence at 30 fps is 0.76 MHz. This means that the power consumption in this case is, approximately, 0.31 mW. Each frame in a QCIF sequence consists of 99 MB, so, the power dissipation per macroblock (MB) per frame per second (fps) is, approximately, 0.0001 mW/MB/fps.

The results of the previous analysis are presented in the Table 37.

Table 37 Comparison according to Yap et al

	Normalized dissipation (mW/MB/fps)	**Normalized speed (fps/CIF)**
Yap et al	0.008	181
Proposed	0.0001	2370

Finally, Sayed et al [77] present in their work a comparison between the architecture they propose and the architectures of Shen et al [82] and Huang et al [83]. As in their comparison they do not use any normalization, so as to have a comparison in a common base, we just add the respective data of our architecture.

Table 38 Comparison according to Sayed et al

	Shen et al.	Huang et al.	Sayed et al.	Proposed
Block size	8x8, 16x16, 32x32	4x4, 4x8, 8x4, 8x8, 8x16, 16x8, 16x16	4x4, 4x8, 8x4, 8x8, 8x16, 16x8, 16x16	4x4, 4x8, 8x4, 8x8, 8x16, 16x8, 16x16
Process	0.60μm	0.35μm	0.18μm	0.13μm
Voltage (V)	2.5 & 5	-	1.6	1.2
Clock freq. (MHz)	60	66.67	122	241
Power (mW)	423.8	737.32	283.96	95

6. Conclusions And Future Work

In this thesis we have examined the Macroblock Prediction process in video coding, beginning from basic concepts of video coding and ending up to the hardware implementation of architectures performing Macroblock Prediction according to the latest video coding standard, H.264.

After an introduction to video compression and a historical analysis, the basic concepts of video coding are presented. More emphasis is given in video encoding, and especially in spatial and temporal prediction. Next, the H.264, which is the framework of the presented work, is introduced and the Intra and Inter Prediction are analysed. When the complete background is given, the algorithms derived from this research are presented.

One of the main tasks of this work was the evaluation of the presented algorithms, which can be divided in two main part, the software and the hardware evaluation. The steps followed were:

- Presentation and analysis of the JM reference software, which is the official reference software for the H.264 coding standard and which was the framework of the software evaluation of the proposed algorithms.

- The software implementation of the new algorithms and its integration with the JM reference software.

- Simulation of various test video sequences, under different encoding conditions, for the evaluation of quality and compression efficiency of the proposed algorithms.

- Presentation and analysis of the architectures designed for the new algorithms.

- Implementation of the architecture of the first algorithm for Intra Prediction, as it is specified in H.264.
- Implementation of a corresponding architecture for Intra prediction using SAD.
- Synthesis and Power Consumption estimation of the implemented circuits, for the hardware evaluation of the first algorithm.
- Implementation of the architecture of the final algorithm for Inter Prediction, as it is specified in H.264.
- Synthesis and Power Consumption estimation of the implemented circuits, for the hardware evaluation of the final algorithm.
- Presentation of the results of previous related work and comparison with those presented in this thesis.

Simulation results of the software implementation of the proposed algorithm showed that by replacing SAD (Sum of Absolute Differences) with SGV (Sum of Greater Values) we can preserve the quality of the encoded video sequences, as the average PSNR difference of the two algorithms does not exceed the 0.4 %. The other basic measure for the effectiveness of an algorithm in video coding is compression efficiency, i.e. the bit-rate of the produced bit-stream. When using EPZS as the motion estimation algorithm the difference in the bit-rate is negligible, as it does not exceed the 5.55% (either with CABAC or with CAVLC entropy encoding). In particular, results showed that for small QPs, i.e high bit-rates the proposed algorithm can achieve nearly the same performance (with average bit-rate difference varying between 0.65 % and 2%) compared to JM. For low bit-rate the performance of the proposed algorithm is reduced but remains competitive to that of JM11. In the case of Full Motion Estimation the results present greater variation for low and high bit-rates. More precisely, for high bit-rates the proposed algorithm can perform even better than JM, as the average bit-rate difference for QP = 14 is -0.18 %, while for low bit-rates the performance of the new algorithm is reduced, as the average bit-rate difference for QP = 50 is -15.85 %. This difference between the results for EPZS and Full Motion Estimation is due to the fact that the nature of the proposed algorithm is more related to the one of fast motion estimation algorithms rather that full motion estimation.

The experimental results of the hardware implementation of the proposed algorithm indicate that the design achieves the real-time encoding requirements for processing an HDTV video sequence with search range 16x16, with a power consumption of just 95mW. Compared with other architectures the proposed achieves better throughput rate (about 13% better than the best presented so far in previous works), and power reduction of about 3X.

Concluding, the new algorithm, by reducing the complexity of the computation, can achieve significantly faster execution time and significant reduction in power consumption without having any perceivable impact on the file size and the quality of the final video sequence.

The algorithms presented in this thesis are intended for macroblock prediction in video coding and, more precisely in the latest video coding standard, H.264. Nonetheless, as a future research avenue, the algorithms could be extended for use in mode decision process in video encoding. Furthermore, in many areas, apart from video coding, measures corresponding to SAD are used. It would, also, be interesting to see if measures based on the concept of “majority vote”, as SGV presented in this thesis, could successfully replace these measures. Last but not least, some novel architectures were presented in the hardware level, such as the comparators. A future work could be to generalize these architectures so as to be used in numerous applications beyond video coding.

APPENDIX A

In this appendix we present some additional simulation results from the software evaluation process of the new algorithm. More precisely, the tables that follow show the file sizes produced by the two encoders, the one with the integration of the new algorithm and the original JM reference software encoder, for the video sequences tested. The encoding was done for various QPs and different entropy encoding scheme, firstly using the "EPZS" fast motion estimation algorithm and secondly using full motion estimation.

Tables 39 to 51 show the results for various QPs, CAVLC as the entropy encoding scheme and "EPZS" as the motion estimation algorithm.

Table 39 File sizes for the video sequences encoded with QP 2 and 4 and CAVLC, EPZS

	QP =2			QP = 4		
Video Sequence	NEW (KB)	% dif	JM (KB)	NEW (KB)	% dif	JM (KB)
src13	75091	0,397	74794	68477	0,449	68171
src14	55714	0,498	55438	48747	0,638	48438
src15	85641	-0,015	85654	78517	-0,020	78533
src16	18613	0,998	18429	15848	0,943	15700
src17	26126	2,515	25485	23420	2,602	22826
src18	69039	0,003	69037	62666	0,003	62664
src19	74054	0,536	73659	67275	0,615	66864
src20	61723	0,449	61447	55129	0,499	54855
src21	55683	0,857	55210	48997	0,983	48520
src22	79365	0,208	79200	72659	0,228	72494
Average % difference		0,645			0,694	

Table 40 File sizes for the video sequences encoded with QP 6 and 8 and CAVLC, EPZS

	QP =6			QP = 8		
Video Sequence	NEW (KB)	% dif	JM (KB)	NEW (KB)	% dif	JM (KB)
src13	61198	0,496	60896	54071	0,578	53760
src14	41854	0,763	41537	34526	0,768	34263
src15	70857	-0,020	70871	63737	-0,030	63756
src16	13376	1,211	13216	11002	1,261	10865
src17	21220	2,835	20635	18387	3,101	17834
src18	55571	0,014	55563	48701	0,012	48695
src19	59950	0,660	59557	52895	0,743	52505
src20	48152	0,537	47895	40994	0,549	40770
src21	42094	1,132	41623	34712	1,207	34298
src22	65384	0,239	65228	58456	0,257	58306
Average % difference		0,787			0,845	

Table 41 File sizes for the video sequences encoded with QP 10 and 12 and CAVLC, EPZS

	QP = 10			QP = 12		
Video Sequence	NEW (KB)	% dif	JM (KB)	NEW (KB)	% dif	JM (KB)
src13	47703	0,639	47400	40432	0,722	40142
src14	28806	0,745	28593	22852	0,647	22705
src15	57384	-0,042	57408	50294	-0,048	50318
src16	9416	1,280	9297	7771	1,861	7629
src17	16410	3,312	15884	14327	3,894	13790
src18	42617	-0,021	42626	35538	-0,011	35542
src19	46601	0,807	46228	39360	0,977	38979
src20	35023	0,583	34820	28288	0,648	28106
src21	28801	1,259	28443	22624	1,371	22318
src22	52188	0,279	52043	45032	0,296	44899
Average % difference		0,884			1,036	

APPENDIX A

Table 42 File sizes for the video sequences encoded with QP 14 and 16 and CAVLC, EPZS

	QP = 14			QP = 16		
Video Sequence	NEW (KB)	% dif	JM (KB)	NEW (KB)	% dif	JM (KB)
src13	34181	0,844	33895	28339	1,472	27928
src14	18462	0,468	18376	14711	0,485	14640
src15	44217	-0,070	44248	38381	0,123	38334
src16	6602	2,261	6456	5600	2,414	5468
src17	12604	4,251	12090	11336	4,808	10816
src18	29357	0,000	29357	23417	0,145	23383
src19	33205	1,158	32825	27554	1,732	27085
src20	22366	0,725	22205	16595	0,734	16474
src21	17657	1,524	17392	13056	2,569	12729
src22	38760	0,298	38645	32799	0,998	32475
Average % difference		1,146			1,548	

Table 43 File sizes for the video sequences encoded with QP 18 and 20 and CAVLC, EPZS

	QP = 18			QP = 20		
Video Sequence	NEW (KB)	% dif	JM (KB)	NEW (KB)	% dif	JM (KB)
src13	22390	1,620	22033	17995	2,070	17630
src14	11278	0,643	11206	8906	0,815	8834
src15	32169	0,153	32120	27128	0,218	27069
src16	4501	3,092	4366	3712	3,370	3591
src17	9688	5,453	9187	8427	6,160	7938
src18	17484	0,178	17453	13040	0,285	13003
src19	22009	1,945	21589	17781	2,343	17374
src20	11407	0,617	11337	8051	0,763	7990
src21	8449	1,489	8325	5857	1,578	5766
src22	26208	0,792	26002	21158	0,714	21008
Average % difference		1,598			1,832	

Table 44 File sizes for the video sequences encoded with QP 22 and 24 and CAVLC, EPZS

	QP = 22			QP = 24		
Video Sequence	NEW (KB)	% dif	JM (KB)	NEW (KB)	% dif	JM (KB)
src13	14692	2,212	14374	11503	1,760	11304
src14	7073	0,942	7007	5277	0,648	5243
src15	22974	0,231	22921	18766	0,214	18726
src16	3094	4,035	2974	2465	3,921	2372
src17	7317	6,849	6848	6114	6,944	5717
src18	9566	0,420	9526	6339	0,396	6314
src19	14523	2,680	14144	11187	2,043	10963
src20	5389	0,993	5336	3341	0,815	3314
src21	4117	1,982	4037	2679	1,631	2636
src22	17279	0,676	17163	13481	0,484	13416
Average % difference		2,102			1,886	

Table 45 File sizes for the video sequences encoded with QP 26 and 28 and CAVLC, EPZS

	QP = 26			QP = 28		
Video Sequence	NEW (KB)	% dif	JM (KB)	NEW (KB)	% dif	JM (KB)
src13	9291	2,121	9098	7511	2,511	7327
src14	4003	0,933	3966	3005	1,247	2968
src15	15588	0,335	15536	12877	0,452	12819
src16	1992	4,677	1903	1620	5,400	1537
src17	5195	7,959	4812	4384	8,946	4024
src18	4237	0,665	4209	2839	1,068	2809
src19	8876	2,435	8665	7036	2,926	6836
src20	2354	1,030	2330	1765	1,495	1739
src21	1793	2,223	1754	1237	2,912	1202
src22	10750	0,608	10685	8556	0,837	8485
Average % difference		2,299			2,779	

APPENDIX A

Table 46 File sizes for the video sequences encoded with QP 30 and 32 and CAVLC, EPZS

	QP = 30			QP = 32		
Video Sequence	NEW (KB)	% dif	JM (KB)	NEW (KB)	% dif	JM (KB)
src13	6021	2,677	5864	4732	3,184	4586
src14	2174	1,541	2141	1528	1,935	1499
src15	10330	0,594	10269	8187	0,751	8126
src16	1302	5,596	1233	1037	6,468	974
src17	3659	9,453	3343	3018	10,712	2726
src18	1835	1,325	1811	1207	1,943	1184
src19	5475	3,166	5307	4241	3,819	4085
src20	1329	1,761	1306	1024	2,298	1001
src21	844	3,558	815	596	4,196	572
src22	6580	0,982	6516	4982	1,178	4924
Average % difference		3,065			3,648	

Table 47 File sizes for the video sequences encoded with QP 34 and 36 and CAVLC, EPZS

	QP = 34			QP = 36		
Video Sequence	NEW (KB)	% dif	JM (KB)	NEW (KB)	% dif	JM (KB)
src13	3768	3,431	3643	2826	4,088	2715
src14	1089	2,350	1064	710	2,601	692
src15	6414	0,897	6357	4646	1,264	4588
src16	842	7,125	786	658	7,341	613
src17	2505	11,432	2248	1985	12,337	1767
src18	856	2,148	838	601	2,735	585
src19	3336	4,315	3198	2493	4,924	2376
src20	810	2,662	789	622	3,151	603
src21	447	4,930	426	335	5,016	319
src22	3746	1,573	3688	2592	1,967	2542
Average % difference		4,086			4,542	

Table 48 File sizes for the video sequences encoded with QP 38 and 40 and CAVLC, EPZS

	QP = 38			QP = 40		
Video Sequence	NEW (KB)	% dif	JM (KB)	NEW (KB)	% dif	JM (KB)
src13	2169	4,379	2078	1682	4,798	1605
src14	496	3,766	478	364	3,409	352
src15	3323	1,621	3270	2298	2,315	2246
src16	532	7,692	494	441	7,299	411
src17	1622	12,327	1444	1339	12,238	1193
src18	461	3,132	447	368	3,081	357
src19	1884	5,487	1786	1432	5,761	1354
src20	494	4,000	475	397	4,199	381
src21	261	6,098	246	212	7,071	198
src22	1780	2,476	1737	1230	3,015	1194
Average % difference		5,098			5,319	

Table 49 File sizes for the video sequences encoded with QP 42 and 44 and CAVLC, EPZS

	QP = 42			QP = 44		
Video Sequence	NEW (KB)	% dif	JM (KB)	NEW (KB)	% dif	JM (KB)
src13	1282	4,739	1224	1027	4,582	982
src14	281	3,309	272	231	3,125	224
src15	1467	3,165	1422	986	4,339	945
src16	368	8,235	340	319	6,689	299
src17	1098	10,685	992	944	9,007	866
src18	290	2,837	282	233	2,643	227
src19	1056	6,237	994	804	6,631	754
src20	315	5,000	300	255	4,938	243
src21	170	6,250	160	139	6,107	131
src22	838	3,713	808	614	4,068	590
Average % difference		5,417			5,213	

Table 50 File sizes for the video sequences encoded with QP 46 and 48 and CAVLC, EPZS

	QP = 46			QP = 48		
Video Sequence	NEW (KB)	% dif	JM (KB)	NEW (KB)	% dif	JM (KB)
src13	778	4,570	744	600	3,986	577
src14	188	2,732	183	163	3,165	158
src15	696	5,295	661	543	6,471	510
src16	270	3,846	260	233	4,018	224
src17	769	8,158	711	631	6,588	592
src18	170	3,030	165	131	2,344	128
src19	599	5,830	566	450	5,140	428
src20	207	5,076	197	172	5,521	163
src21	121	5,217	115	107	1,905	105
src22	464	4,505	444	357	4,082	343
Average % difference		4,826			4,322	

Table 51 File sizes for the video sequences encoded with QP 50 and CAVLC, EPZS

	QP =50		
Video Sequence	NEW (KB)	% dif	JM (KB)
src13	508	2,626	495
src14	149	2,055	146
src15	521	7,867	483
src16	215	1,896	211
src17	552	4,744	527
src18	116	0,870	115
src19	368	3,662	355
src20	147	2,797	143
src21	108	0,000	108
src22	294	3,521	284
Average % difference		3,004	

APPENDIX A

Tables 52 to 64 show the results for various QPs, CABAC as the entropy encoding scheme and "EPZS" as the motion estimation algorithm.

Table 52 File sizes for the video sequences encoded with QP 2 and 4 and CABAC, EPZS

	QP =2			QP = 4		
Video Sequence	NEW (KB)	% dif	JM (KB)	NEW (KB)	% dif	JM (KB)
src13	73551	0,675	73058	65007	0,611	64612
src14	50700	0,480	50458	44320	0,581	44064
src15	94736	0,130	94613	83383	0,159	83251
src16	17825	1,313	17594	15251	1,268	15060
src17	28269	5,297	26847	24519	5,345	23275
src18	64337	0,103	64271	57783	0,099	57726
src19	71761	1,042	71021	62875	1,106	62187
src20	56242	0,400	56018	49847	0,455	49621
src21	49349	0,688	49012	43223	0,791	42884
src22	83119	0,794	82464	72473	0,418	72171
Average % difference		1,092			1,083	

Table 53 File sizes for the video sequences encoded with QP 6 and 8 and CABAC, EPZS

	QP =6			QP = 8		
Video Sequence	NEW (KB)	% dif	JM (KB)	NEW (KB)	% dif	JM (KB)
src13	56971	0,629	56615	49987	0,652	49663
src14	38016	0,702	37751	31598	0,733	31368
src15	72044	0,189	71908	62458	0,209	62328
src16	12887	1,552	12690	10545	1,629	10376
src17	21338	5,015	20319	18036	4,357	17283
src18	50697	0,111	50641	44184	0,043	44165
src19	54739	1,060	54165	48017	1,110	47490
src20	42922	0,492	42712	36305	0,495	36126
src21	36789	0,919	36454	30275	0,954	29989
src22	62021	0,452	61742	54210	0,342	54025
Average % difference		1,112			1,052	

APPENDIX A

Table 54 File sizes for the video sequences encoded with QP 10 and 12 and CABAC, EPZS

	QP = 10			QP = 12		
Video Sequence	NEW (KB)	% dif	JM (KB)	NEW (KB)	% dif	JM (KB)
src13	44024	0,649	43740	37259	0,714	36995
src14	26434	0,724	26244	20991	0,691	20847
src15	54637	0,193	54532	47184	0,085	47144
src16	8931	1,673	8784	7310	2,252	7149
src17	15877	4,057	15258	13750	4,555	13151
src18	38527	0,005	38525	31971	0,019	31965
src19	42167	1,178	41676	35500	1,391	35013
src20	30783	0,545	30616	24567	0,639	24411
src21	25060	0,991	24814	19625	1,149	19402
src22	48203	0,356	48032	41462	0,312	41333
Average % difference		1,037			1,181	

Table 55 File sizes for the video sequences encoded with QP 14 and 16 and CABAC, EPZS

	QP = 14			QP = 16		
Video Sequence	NEW (KB)	% dif	JM (KB)	NEW (KB)	% dif	JM (KB)
src13	31534	0,825	31276	26156	1,396	25796
src14	16906	0,571	16810	13344	0,588	13266
src15	41390	0,080	41357	35874	0,173	35812
src16	6160	2,735	5996	5182	2,899	5036
src17	12038	4,952	11470	10753	5,391	10203
src18	26439	0,034	26430	21267	0,146	21236
src19	29896	1,480	29460	24735	2,325	24173
src20	19395	0,607	19278	14490	0,716	14387
src21	15369	1,292	15173	11379	2,348	11118
src22	35787	0,314	35675	30414	0,953	30127
Average % difference		1,289			1,693	

Table 56 File sizes for the video sequences encoded with QP 18 and 20 and CABAC, EPZS

	QP = 18			QP = 20		
Video Sequence	NEW (KB)	% dif	JM (KB)	NEW (KB)	% dif	JM (KB)
src13	20591	1,554	20276	16425	2,000	16103
src14	10178	0,772	10100	7956	0,990	7878
src15	30055	0,210	29992	25240	0,274	25171
src16	4167	3,683	4019	3425	3,882	3297
src17	9153	6,036	8632	7920	6,624	7428
src18	15929	0,195	15898	11911	0,269	11879
src19	19609	2,686	19096	15686	3,333	15180
src20	10088	0,659	10022	7181	0,828	7122
src21	7464	1,482	7355	5182	1,708	5095
src22	24424	0,821	24225	19757	0,755	19609
Average % difference		1,810			2,066	

Table 57 File sizes for the video sequences encoded with QP 22 and 24 and CABAC, EPZS

	QP = 22			QP = 24		
Video Sequence	NEW (KB)	% dif	JM (KB)	NEW (KB)	% dif	JM (KB)
src13	13248	2,136	12971	10217	1,763	10040
src14	6248	1,199	6174	4598	0,767	4563
src15	21188	0,289	21127	17150	0,322	17095
src16	2834	4,653	2708	2241	4,184	2151
src17	6843	7,257	6380	5702	7,585	5300
src18	8702	0,404	8667	5740	0,473	5713
src19	12668	3,726	12213	9583	2,481	9351
src20	4894	1,116	4840	3102	0,944	3073
src21	3622	2,201	3544	2337	1,874	2294
src22	16059	0,715	15945	12465	0,508	12402
Average % difference		2,369			2,090	

APPENDIX A

Table 58 File sizes for the video sequences encoded with QP 26 and 28 and CABAC, EPZS

	QP = 26			QP = 28		
Video Sequence	NEW (KB)	% dif	JM (KB)	NEW (KB)	% dif	JM (KB)
src13	8127	2,149	7956	6483	2,595	6319
src14	3441	1,117	3403	2553	1,511	2515
src15	14110	0,434	14049	11571	0,565	11506
src16	1806	4,939	1721	1463	5,708	1384
src17	4816	8,444	4441	4043	9,241	3701
src18	3801	0,742	3773	2522	1,123	2494
src19	7532	3,136	7303	5928	3,800	5711
src20	2203	1,194	2177	1653	1,661	1626
src21	1551	2,579	1512	1062	3,509	1026
src22	9867	0,643	9804	7793	0,893	7724
Average % difference		2,538			3,061	

Table 59 File sizes for the video sequences encoded with QP 30 and 32 and CABAC, EPZS

	QP = 30			QP = 32		
Video Sequence	NEW (KB)	% dif	JM (KB)	NEW (KB)	% dif	JM (KB)
src13	5123	2,706	4988	3964	3,337	3836
src14	1821	1,732	1790	1267	2,342	1238
src15	9227	0,709	9162	7265	0,875	7202
src16	1168	5,701	1105	927	6,797	868
src17	3359	9,556	3066	2752	10,611	2488
src18	1614	1,382	1592	1055	2,031	1034
src19	4551	3,786	4385	3485	4,686	3329
src20	1241	1,888	1218	956	2,355	934
src21	712	3,790	686	499	4,832	476
src22	5938	0,986	5880	4451	1,205	4398
Average % difference		3,224			3,907	

Table 60 File sizes for the video sequences encoded with QP 34 and 36 and CABAC, EPZS

	QP = 34			QP = 36		
Video Sequence	NEW (KB)	% dif	JM (KB)	NEW (KB)	% dif	JM (KB)
src13	3110	3,494	3005	2294	4,083	2204
src14	896	2,752	872	582	3,009	565
src15	5672	1,051	5613	4090	1,388	4034
src16	747	7,328	696	577	7,449	537
src17	2270	11,166	2042	1778	11,754	1591
src18	744	2,338	727	517	2,988	502
src19	2699	5,019	2570	1985	5,473	1882
src20	756	2,717	736	579	3,209	561
src21	371	5,698	351	274	5,792	259
src22	3316	1,531	3266	2271	1,976	2227
Average % difference		4,309			4,712	

Table 61 File sizes for the video sequences encoded with QP 38 and 40 and CABAC, EPZS

	QP = 38			QP = 40		
Video Sequence	NEW (KB)	% dif	JM (KB)	NEW (KB)	% dif	JM (KB)
src13	1735	4,392	1662	1331	4,886	1269
src14	403	4,134	387	292	3,915	281
src15	2918	1,779	2867	2019	2,435	1971
src16	459	7,746	426	374	7,471	348
src17	1434	11,769	1283	1167	11,568	1046
src18	391	3,714	377	307	4,422	294
src19	1473	5,971	1390	1104	6,256	1039
src20	457	3,864	440	365	4,286	350
src21	209	6,091	197	167	7,051	156
src22	1545	2,454	1508	1057	3,122	1025
Average % difference		5,191			5,541	

Table 62 File sizes for the video sequences encoded with QP 42 and 44 and CABAC, EPZS

	QP = 42			QP = 44		
Video Sequence	NEW (KB)	% dif	JM (KB)	NEW (KB)	% dif	JM (KB)
src13	1000	4,712	955	796	4,599	761
src14	221	3,756	213	179	4,070	172
src15	1287	3,208	1247	861	4,364	825
src16	303	7,829	281	258	7,054	241
src17	940	10,329	852	795	8,904	730
src18	235	3,982	226	184	3,955	177
src19	800	6,525	751	599	6,774	561
src20	286	4,762	273	229	5,046	218
src21	133	6,400	125	108	5,882	102
src22	708	3,812	682	511	4,286	490
Average % difference		5,531			5,493	

Table 63 File sizes for the video sequences encoded with QPs 46 and 48 and CABAC, EPZS

	QP = 46			QP = 48		
Video Sequence	NEW (KB)	% dif	JM (KB)	NEW (KB)	% dif	JM (KB)
src13	598	4,729	571	460	4,072	442
src14	143	4,380	137	120	3,448	116
src15	600	5,263	570	455	6,557	427
src16	212	4,433	203	179	4,678	171
src17	637	8,149	589	515	6,625	483
src18	133	4,724	127	101	3,061	98
src19	441	6,265	415	331	5,079	315
src20	182	5,202	173	146	6,569	137
src21	93	4,494	89	80	1,266	79
src22	380	4,683	363	288	4,348	276
Average % difference		5,232			4,571	

Table 64 File sizes for the video sequences encoded with QP 50 and CABAC, EPZS

	QP =50		
Video Sequence	NEW (KB)	% dif	JM (KB)
src13	389	2,639	379
src14	108	2,857	105
src15	423	8,184	391
src16	161	1,899	158
src17	444	4,471	425
src18	88	1,149	87
src19	271	4,231	260
src20	119	3,478	115
src21	75	0,000	75
src22	234	4,000	225
Average % difference		3,291	

Tables 65 to 67 show the results for various QPs, CAVLC as the entropy encoding scheme and "Full Search" as the motion estimation algorithm.

Table 65 File sizes for the video sequences encoded with QP 2 and 14 and CAVLC, Full Search

	QP = 2			QP = 14		
Video Sequence	NEW (KB)	% dif	JM (KB)	NEW (KB)	% dif	JM (KB)
src13	74774	-1,028	75551	33905	-2,712	34850
src14	55643	-0,253	55784	18286	-1,593	18582
src15	85303	-1,457	86564	43872	-2,798	45135
src16	18550	2,543	18090	6578	4,845	6274
src17	26271	5,846	24820	12677	7,024	11845
src18	68961	-0,807	69522	29234	-2,021	29837
src19	73886	-0,655	74373	33020	-2,003	33695
src20	61695	0,157	61598	22323	0,238	22270
src21	55712	0,286	55553	17593	0,291	17542
src22	79080	-1,406	80208	38433	-3,040	39638
Average % difference		0,323			-0,177	

APPENDIX A

Table 66 File sizes for the video sequences encoded with QP 28 and 38 and CAVLC, Full Search

	QP = 28			QP = 38		
Video Sequence	NEW (KB)	% dif	JM (KB)	NEW (KB)	% dif	JM (KB)
src13	7546	2,541	7359	2220	5,815	2098
src14	3002	2,562	2927	527	8,884	484
src15	12868	0,398	12817	3378	3,556	3262
src16	1666	10,112	1513	568	15,682	491
src17	4460	14,623	3891	1700	20,739	1408
src18	2897	2,115	2837	486	3,846	468
src19	7026	2,915	6827	1917	8,305	1770
src20	1765	1,495	1739	494	3,782	476
src21	1256	4,232	1205	285	12,205	254
src22	8581	0,586	8531	1821	3,584	1758
Average % difference		4,158			8,640	

Table 67 File sizes for the video sequences encoded with QP 50 and CAVLC, Full Search

	QP =50		
Video Sequence	NEW (KB)	% dif	JM (KB)
src13	520	4,839	496
src14	164	28,125	128
src15	634	13,620	558
src16	224	24,444	180
src17	582	14,793	507
src18	117	2,632	114
src19	385	6,648	361
src20	147	14,844	128
src21	111	38,750	80
src22	315	9,756	287
Average % difference		15,845	

APPENDIX A

Tables 68 to 70 show the results for various QPs, CABAC as the entropy encoding scheme and "Full Search" as the motion estimation algorithm.

Table 68 File sizes for the video sequences encoded with QP 2 and 14 and CABAC, Full Search

	QP = 2			QP = 14		
Video Sequence	NEW (KB)	% dif	JM (KB)	NEW (KB)	% dif	JM (KB)
src13	73347	0,064	73300	31381	-1,528	31868
src14	50675	0,180	50584	16785	-0,492	16868
src15	94767	-0,173	94931	41286	-1,627	41969
src16	17831	3,596	17212	6162	6,812	5769
src17	28799	13,066	25471	12132	9,179	11112
src18	64492	-0,102	64558	26499	-0,961	26756
src19	71677	0,633	71226	29798	-0,541	29960
src20	56255	0,242	56119	19402	0,326	19339
src21	49416	0,531	49155	15368	1,152	15193
src22	83197	0,282	82963	35715	-1,487	36254
Average % difference		1,832			1,083	

Table 69 File sizes for the video sequences encoded with QP 28 and 38 and CABAC, Full Search

	QP = 28			QP = 38		
Video Sequence	NEW (KB)	% dif	JM (KB)	NEW (KB)	% dif	JM (KB)
src13	6514	3,070	6320	1778	6,340	1672
src14	2559	3,269	2478	432	9,924	393
src15	11594	1,390	11435	2971	4,173	2852
src16	1508	11,292	1355	490	15,839	423
src17	4101	15,651	3546	1495	19,984	1246
src18	2577	2,506	2514	413	4,557	395
src19	5925	4,295	5681	1500	8,853	1378
src20	1654	1,847	1624	457	3,864	440
src21	1081	5,156	1028	228	11,765	204
src22	7827	1,478	7713	1581	3,808	1523
Average % difference		4,995			8,911	

Table 70 File sizes for the video sequences encoded with QP 50 and CABAC, Full Search

	QP =50		
Video Sequence	NEW (KB)	% dif	JM (KB)
src13	398	5,851	376
src14	121	28,723	94
src15	518	14,856	451
src16	168	19,149	141
src17	467	13,625	411
src18	89	4,706	85
src19	283	6,792	265
src20	119	12,264	106
src21	76	28,814	59
src22	250	11,111	225
Average % difference		14,589	

The results that follow are from the evaluation process for the case of Full Motion Estimation, where 10 sequences from VQEG were encoded as in the case of EPZS presented in Chapter 4.

The rate – distortion curves which follow are for the case of CAVLC entropy encoding scheme, and the data used for these graphs are presented in the corresponding tables.

Table 71 PSNR and Bitrate for the src13 video sequence (full search)

src13	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,43	74774	56,42	75551	0,02	-1,03
14	45,94	33905	45,94	34850	0,00	-2,71
28	35,79	7546	35,79	7359	0,00	2,54
38	29,04	2220	29,06	2098	-0,07	5,82
50	22,69	520	22,65	496	0,18	4,84
	Average % difference				0,03	1,89

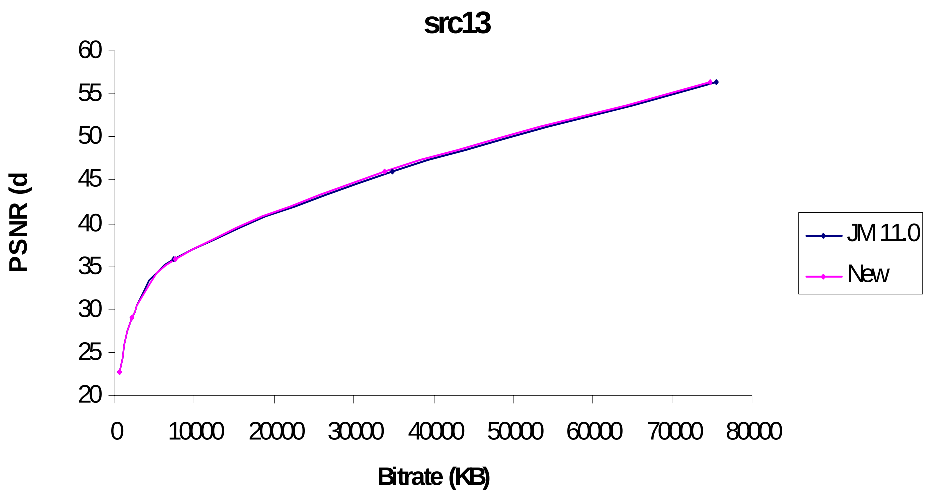


Figure 75 Rate – Distortion curves for src13 (full search)

Table 72 PSNR and Bitrate for the src14 video sequence (full search)

src14	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,53	55643	56,54	55784	-0,02	-0,25
14	46,59	18286	46,63	18582	-0,09	-1,59
28	37,09	3002	37,08	2927	0,03	2,56
38	30,79	527	30,82	484	-0,10	8,88
50	25,39	164	25,33	128	0,24	28,13
Average % difference					0,01	7,55

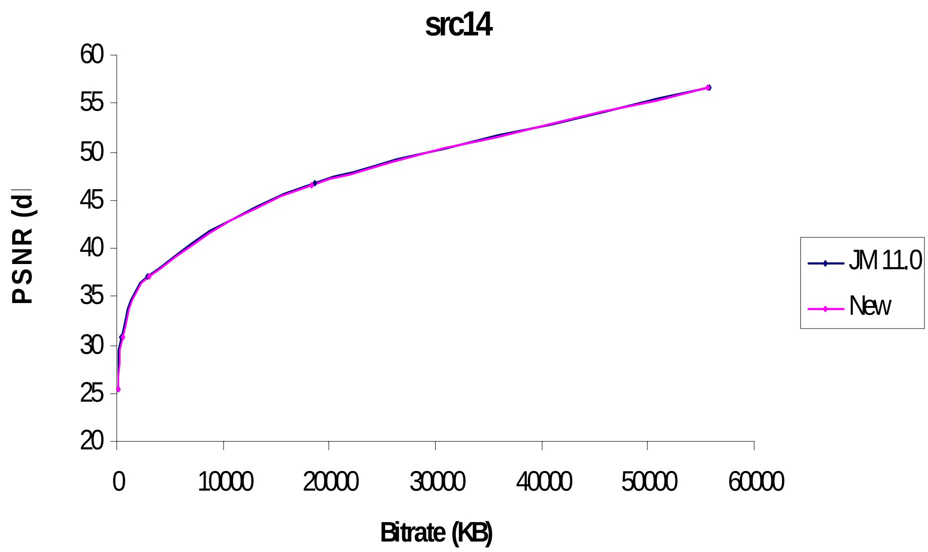


Figure 76 Rate – Distortion curves for src14 (full search)

Table 73 PSNR and Bitrate for the src15 video sequence (full search)

src15	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,28	85303	56,28	86564	0,00	-1,46
14	45,62	43872	45,64	45135	-0,04	-2,80
28	33,6	12868	33,61	12817	-0,03	0,40
38	25,27	3378	25,59	3262	-1,25	3,56
50	18,46	634	18,55	558	-0,49	13,62
Average % difference					-0,36	2,66

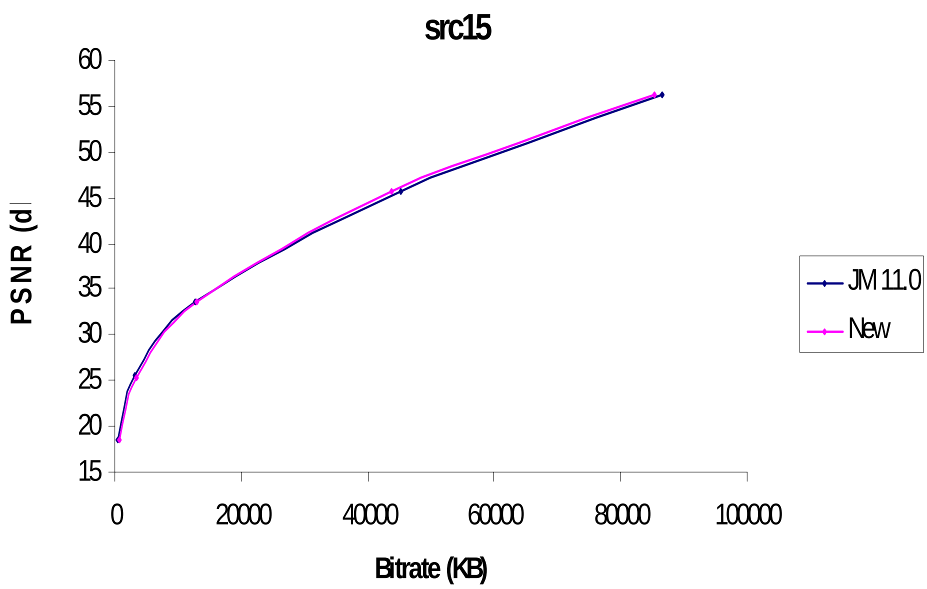


Figure 77 Rate – Distortion curves for src15 (full search)

Table 74 PSNR and Bitrate for the src16 video sequence (full search)

src16	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	58,35	18550	58,41	18090	-0,10	2,54
14	49,44	6578	49,42	6274	0,04	4,85
28	39,73	1666	39,77	1513	-0,10	10,11
38	33,04	568	33,11	491	-0,21	15,68
50	26,63	224	26,69	180	-0,22	24,44
Average % difference					-0,12	11,53

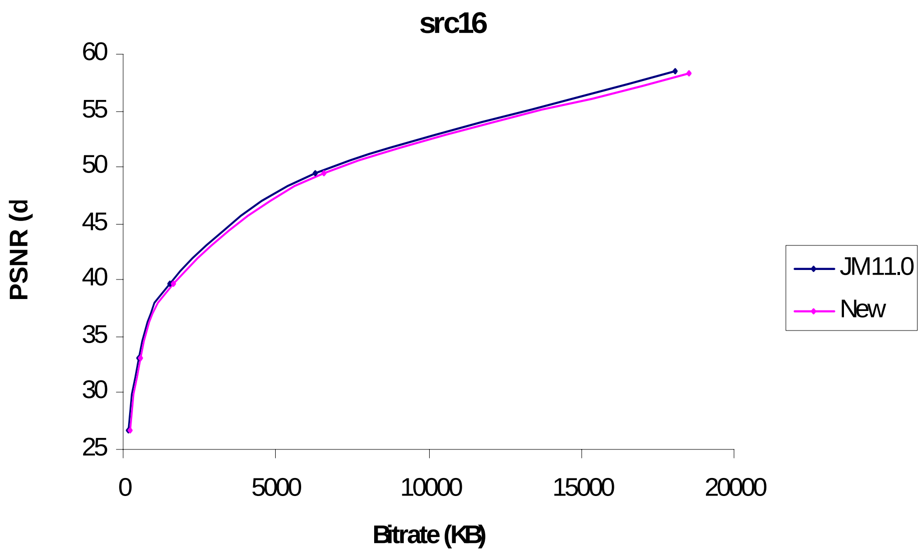


Figure 78 Rate – Distortion curves for src16 (full search)

Table 75 PSNR and Bitrate for the src17 video sequence (full search)

src17	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	58,25	26271	58,33	24820	-0,14	5,85
14	48,72	12677	48,74	11845	-0,04	7,02
28	37,58	4460	37,71	3891	-0,34	14,62
38	29,93	1700	30,15	1408	-0,73	20,74
50	22,5	582	22,67	507	-0,75	14,79
	Average % difference				-0,40	12,61

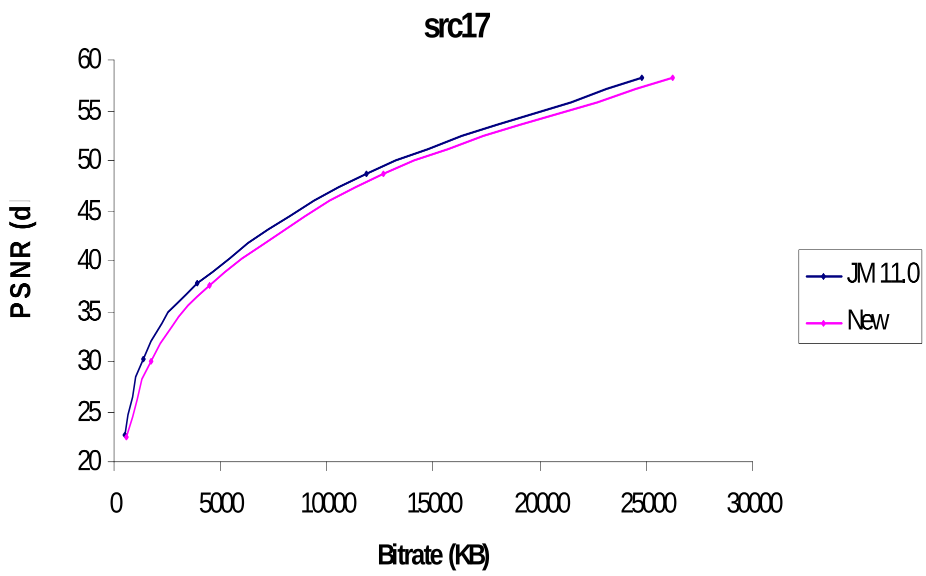


Figure 79 Rate – Distortion curves for src17 (full search)

Table 76 PSNR and Bitrate for the src18 video sequence (full search)

src18	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,28	68961	56,29	69522	-0,02	-0,81
14	45,81	29234	45,81	29837	0,00	-2,02
28	34,97	2897	34,96	2837	0,03	2,11
38	29,46	486	29,35	468	0,37	3,85
50	24,74	117	24,72	114	0,08	2,63
	Average % difference				0,09	1,15

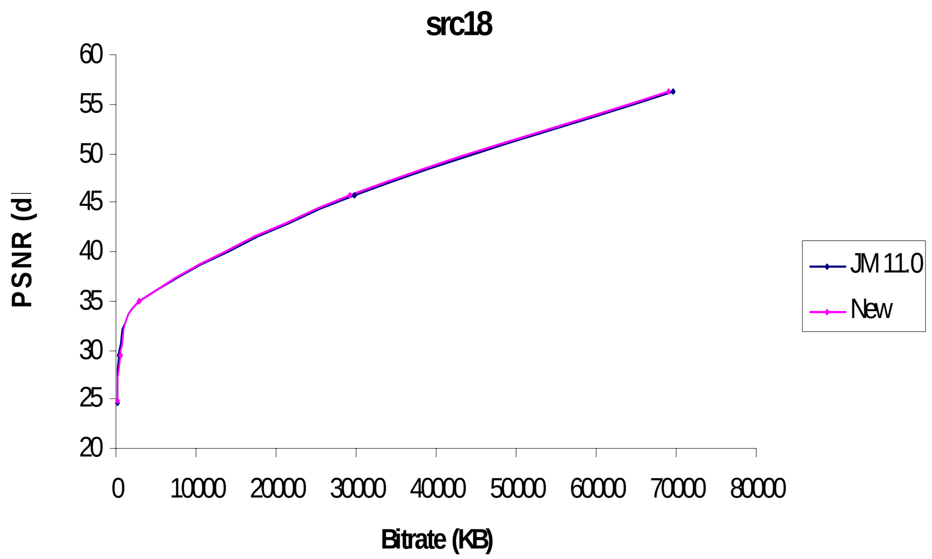


Figure 80 Rate – Distortion curves for src18 (full search)

Table 77 PSNR and Bitrate for the src19 video sequence (full search)

src19	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,31	73886	56,31	74373	0,00	-0,65
14	45,87	33020	45,88	33695	-0,02	-2,00
28	35,36	7026	35,38	6827	-0,06	2,91
38	29,12	1917	29,12	1770	0,00	8,31
50	23,61	385	23,61	361	0,00	6,65
Average % difference					-0,02	3,04

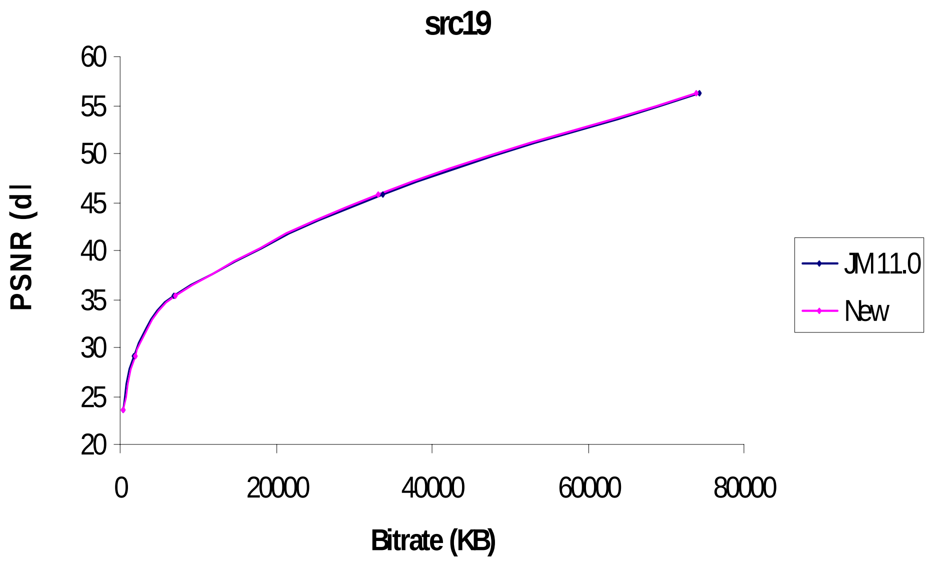


Figure 81 Rate – Distortion curves for src19 (full search)

Table 78 PSNR and Bitrate for the src20 video sequence (full search)

src20	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,33	61695	56,33	61598	0,00	0,16
14	45,96	22323	45,98	22270	-0,04	0,24
28	35,35	1765	35,35	1739	0,00	1,50
38	28,6	494	28,61	476	-0,03	3,78
50	21,17	147	21,2	128	-0,14	14,84
	Average % difference				-0,04	4,10

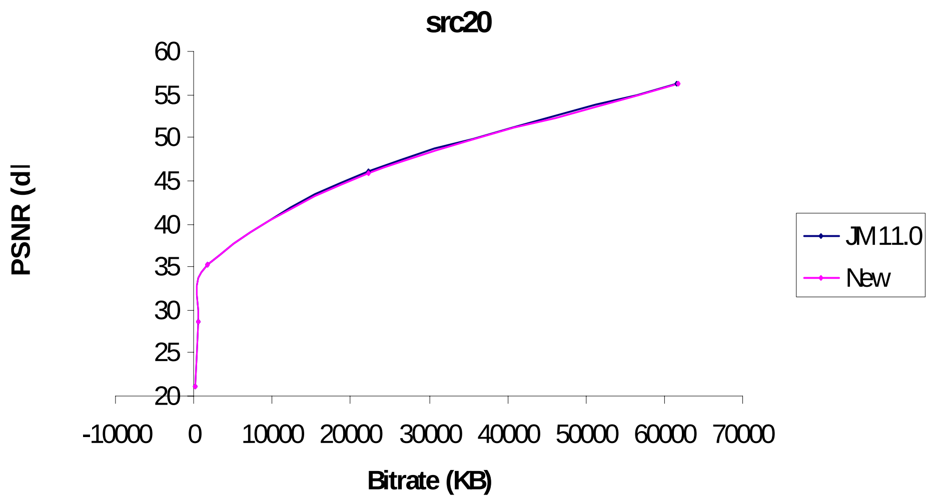


Figure 82 Rate – Distortion curves for src20 (full search)

Table 79 PSNR and Bitrate for the src21 video sequence (full search)

src21	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,36	55712	56,37	55553	-0,02	0,29
14	46,06	17593	46,08	17542	-0,04	0,29
28	37,47	1256	37,45	1205	0,05	4,23
38	32,87	285	32,85	254	0,06	12,20
50	27,8	111	27,77	80	0,11	38,75
	Average % difference				0,03	11,15

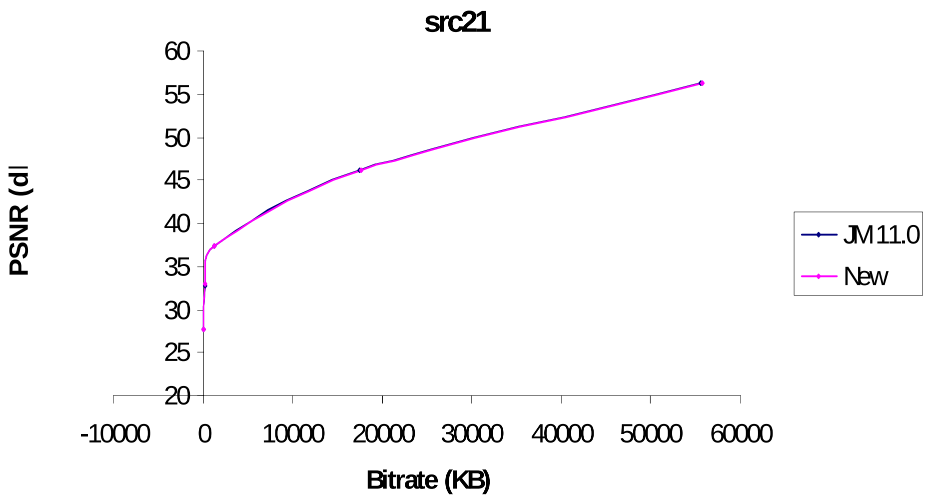


Figure 83 Rate – Distortion curves for src21 (full search)

Table 80 PSNR and Bitrate for the src22 video sequence (full search)

src22	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,27	79080	56,27	80208	0,00	-1,41
14	45,71	38433	45,73	39638	-0,04	-3,04
28	34,37	8581	34,37	8531	0,00	0,59
38	27,2	1821	27,2	1758	0,00	3,58
50	20,7	315	20,64	287	0,29	9,76
	Average % difference				0,05	1,90

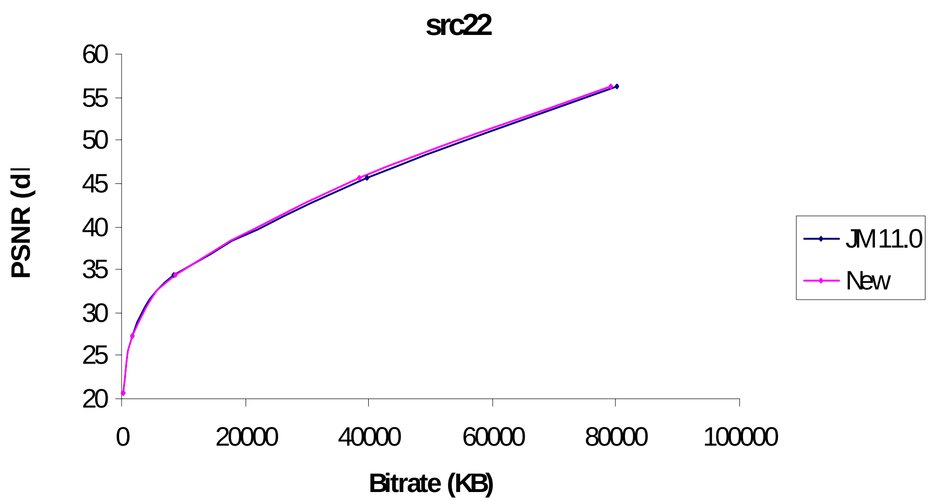


Figure 84 Rate – Distortion curves for src22 (full search)

The rate – distortion curves that follow are for the case of CABAC entropy encoding scheme, and the data used for these graphs are presented in the corresponding tables.

Table 81 PSNR and Bitrate for the src13 video sequence (full search)

src13	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,43	73347	56,42	73300	0,02	0,06
14	45,94	31381	45,94	31868	0,00	-1,53
28	35,79	6514	35,79	6320	0,00	3,07
38	29,04	1778	29,06	1672	-0,07	6,34
50	22,69	398	22,65	376	0,18	5,85
Average % difference					0,03	2,76

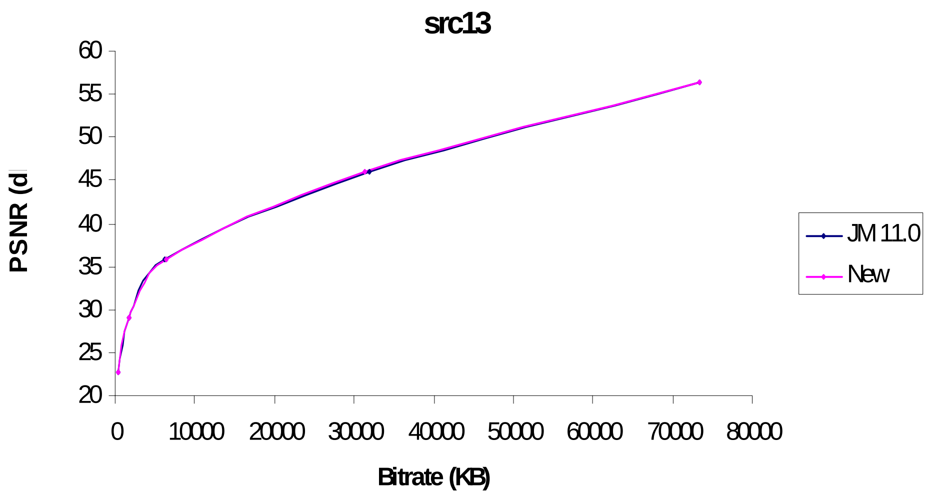


Figure 85 Rate – Distortion curves for src13 (full search)

Table 82 PSNR and Bitrate for the src14 video sequence (full search)

src14	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,53	50675	56,54	50584	-0,02	0,18
14	46,59	16785	46,63	16868	-0,09	-0,49
28	37,09	2559	37,08	2478	0,03	3,27
38	30,79	432	30,82	393	-0,10	9,92
50	25,39	121	25,33	94	0,24	28,72
Average % difference					0,01	8,32

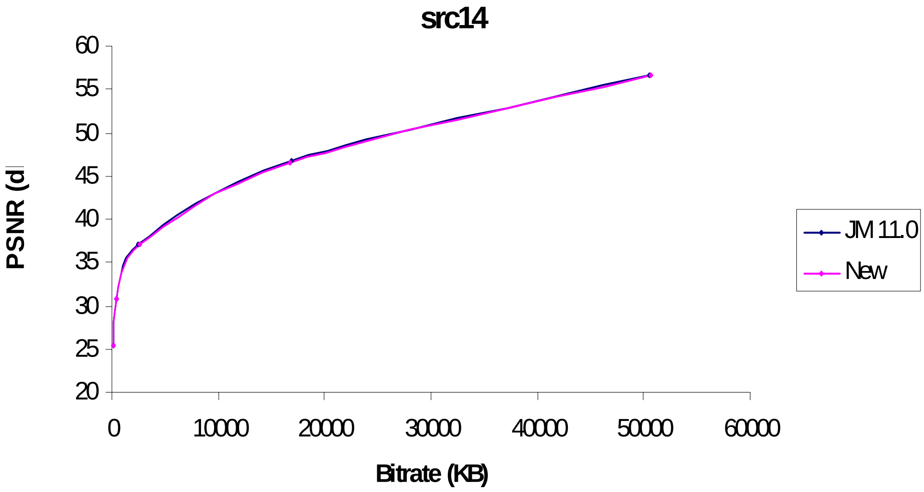


Figure 86 Rate – Distortion curves for src14 (full search)

Table 83 PSNR and Bitrate for the src15 video sequence (full search)

src15	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,28	94767	56,28	94931	0,00	-0,17
14	45,62	41286	45,64	41969	-0,04	-1,63
28	33,6	11594	33,61	11435	-0,03	1,39
38	25,27	2971	25,59	2852	-1,25	4,17
50	18,46	518	18,55	451	-0,49	14,86
	Average % difference				-0,36	3,72

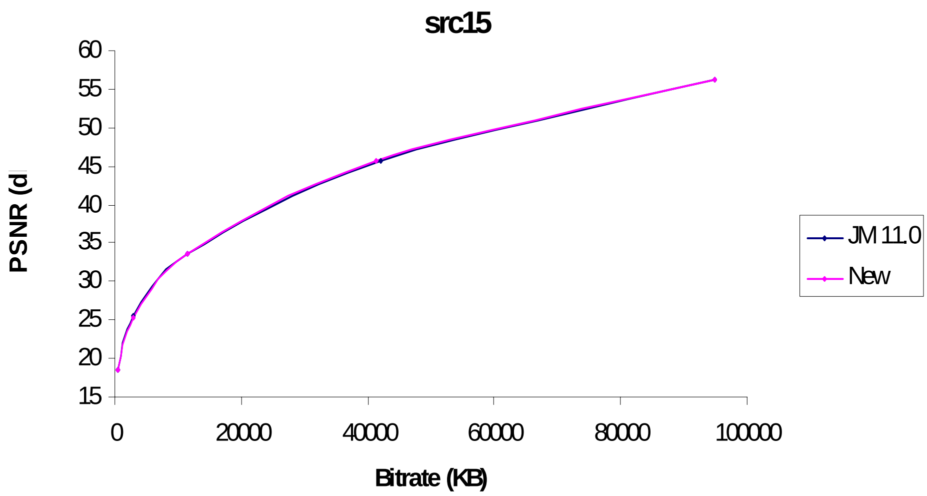


Figure 87 Rate – Distortion curves for src15 (full search)

Table 84 PSNR and Bitrate for the src16 video sequence (full search)

src16	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	58,35	17831	58,41	17212	-0,10	3,60
14	49,44	6162	49,42	5769	0,04	6,81
28	39,73	1508	39,77	1355	-0,10	11,29
38	33,04	490	33,11	423	-0,21	15,84
50	26,63	168	26,69	141	-0,22	19,15
Average % difference					-0,12	11,34

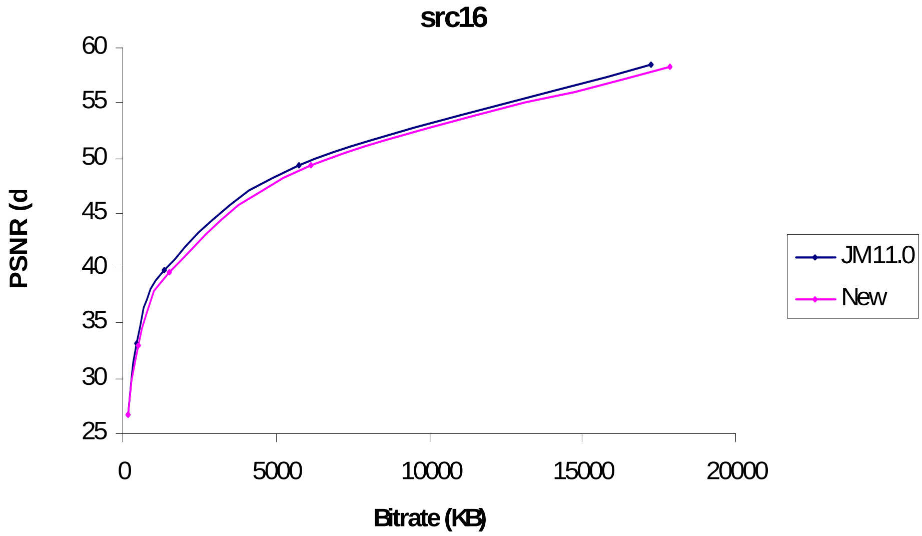


Figure 88 Rate – Distortion curves for src16 (full search)

Table 85 PSNR and Bitrate for the src17 video sequence (full search)

src17	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	58,25	28799	58,33	25471	-0,14	13,07
14	48,72	12132	48,74	11112	-0,04	9,18
28	37,58	4101	37,71	3546	-0,34	15,65
38	29,93	1495	30,15	1246	-0,73	19,98
50	22,5	467	22,67	411	-0,75	13,63
Average % difference					-0,40	14,30

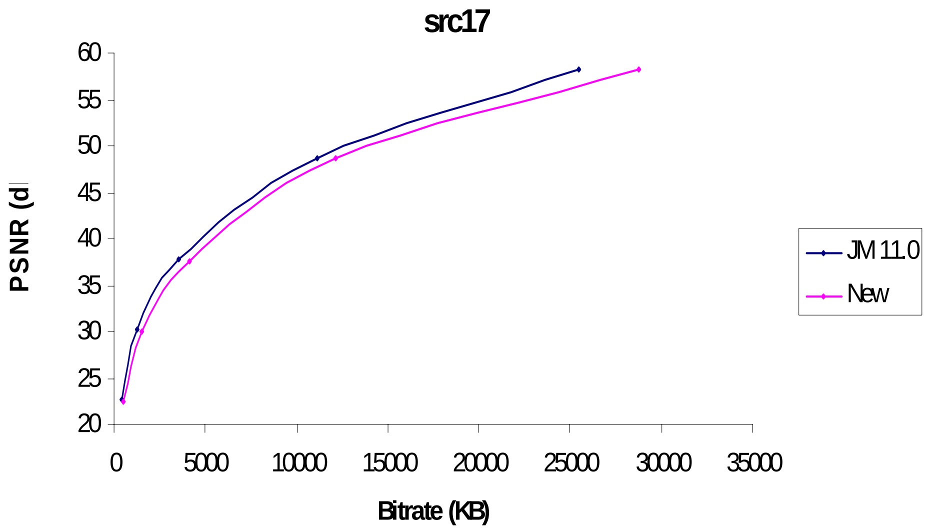


Figure 89 Rate – Distortion curves for src17 (full search)

Table 86 PSNR and Bitrate for the src18 video sequence (full search)

src18	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,28	64492	56,29	64558	-0,02	-0,10
14	45,81	26499	45,81	26756	0,00	-0,96
28	34,97	2577	34,96	2514	0,03	2,51
38	29,46	413	29,35	395	0,37	4,56
50	24,74	89	24,72	85	0,08	4,71
Average % difference					0,09	2,14

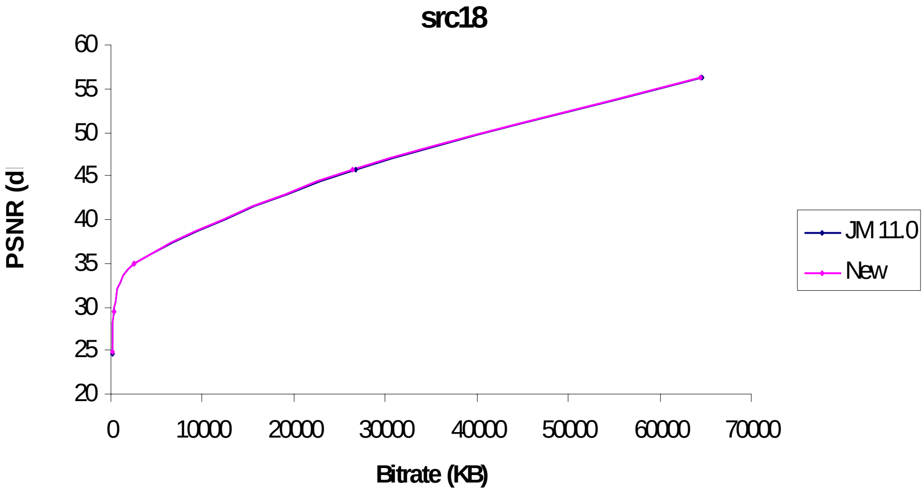


Figure 90 Rate – Distortion curves for src18 (full search)

Table 87 PSNR and Bitrate for the src19 video sequence (full search)

src19	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,31	71677	56,31	71226	0,00	0,63
14	45,87	29798	45,88	29960	-0,02	-0,54
28	35,36	5925	35,38	5681	-0,06	4,30
38	29,12	1500	29,12	1378	0,00	8,85
50	23,61	283	23,61	265	0,00	6,79
Average % difference					-0,02	4,01

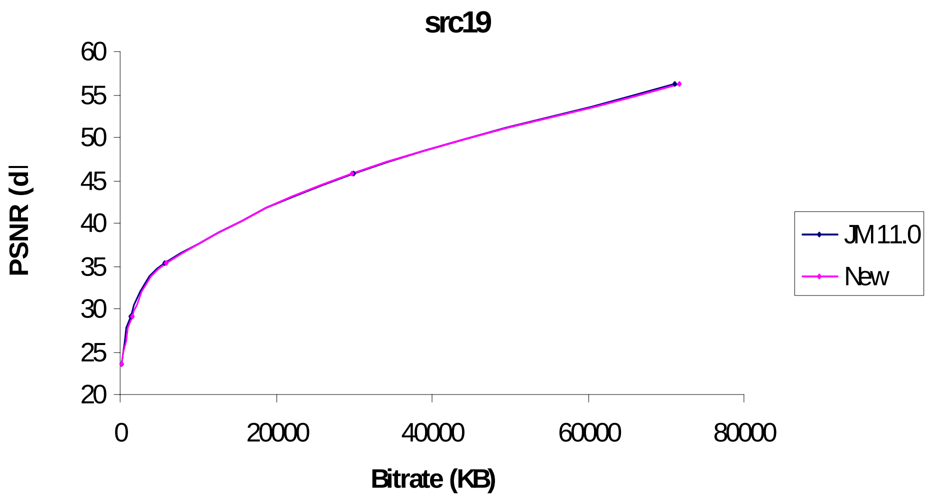


Figure 91 Rate – Distortion curves for src19 (full search)

Table 88 PSNR and Bitrate for the src20 video sequence (full search)

src20	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,33	56255	56,33	56119	0,00	0,24
14	45,96	19402	45,98	19339	-0,04	0,33
28	35,35	1654	35,35	1624	0,00	1,85
38	28,6	457	28,61	440	-0,03	3,86
50	21,17	119	21,2	106	-0,14	12,26
Average % difference					-0,04	3,71

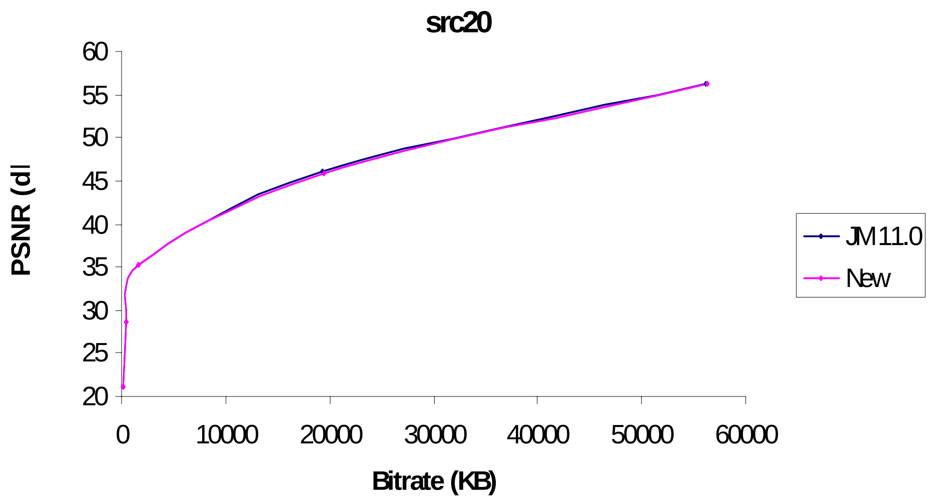


Figure 92 Rate – Distortion curves for src20 (full search)

Table 89 PSNR and Bitrate for the src21 video sequence (full search)

src21	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,36	49416	56,37	49155	-0,02	0,53
14	46,06	15368	46,08	15193	-0,04	1,15
28	37,37	1081	37,45	1028	-0,21	5,16
38	32,87	228	32,85	204	0,06	11,76
50	27,8	76	27,77	59	0,11	28,81
	Average % difference				-0,02	9,48

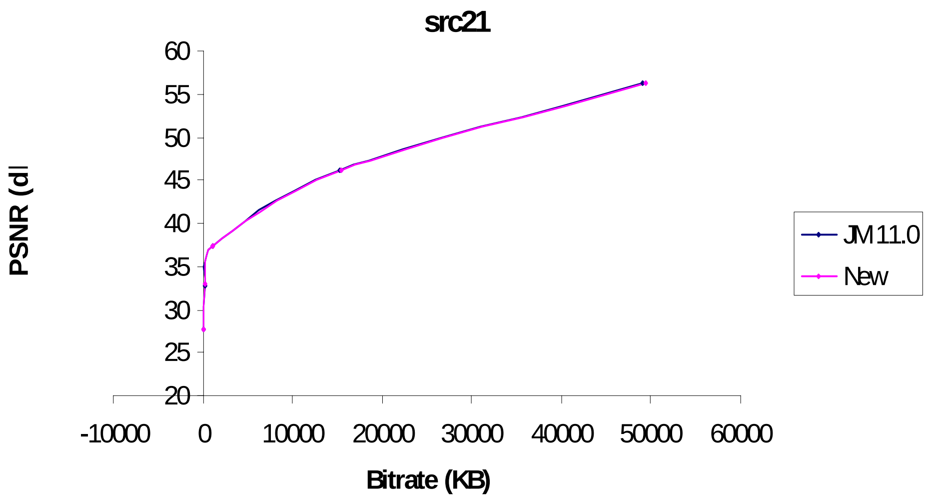


Figure 93 Rate – Distortion curves for src21 (full search)

Table 90 PSNR and Bitrate for the src22 video sequence (full search)

src22	NEW		JM		% difference	
Qp	PSNR (db)	Bitrate (kB)	PSNR (db)	Bitrate (kB)	PSNR	Bitrate
2	56,27	83197	56,27	82963	0,00	0,28
14	45,71	35715	45,73	36254	-0,04	-1,49
28	34,37	7827	34,37	7713	0,00	1,48
38	27,2	1581	27,2	1523	0,00	3,81
50	20,7	250	20,64	225	0,29	11,11
Average % difference					0,05	3,04

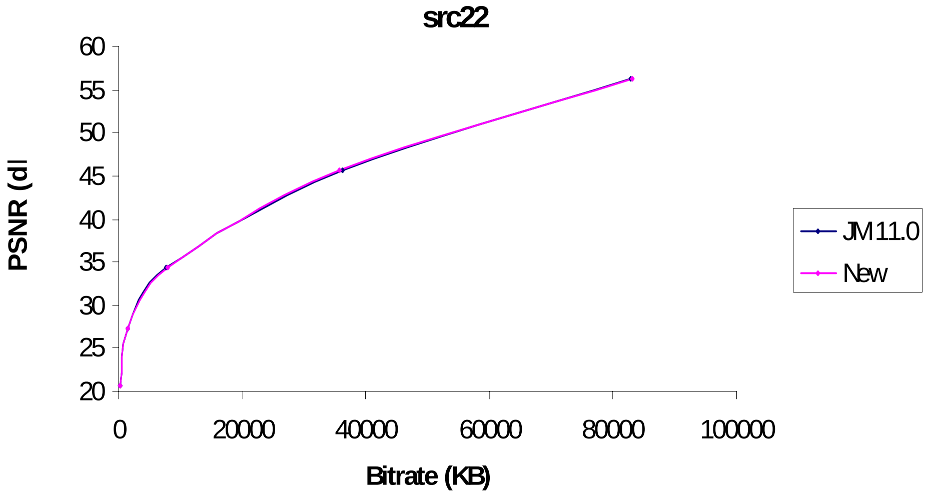


Figure 94 Rate – Distortion curves for src22 (full search)

APPENDIX A

APPENDIX B

Table 91 provides a full list of the coding functions supported by each of the seven available profiles in H.264 Standard.

Table 91 Profiles in H.264

	Baseline	Main	Extended	High	High10	High 4:2:4	High 4:4:4
I and P Slices	√	√	√	√	√	√	√
B Slices	X	√	√	√	√	√	√
SI and SP Slices	X	√	X	X	X	X	X
Multiple Reference Frames	√	√	√	√	√	√	√
In-Loop Deblocking Filter	√	√	√	√	√	√	√
CAVLC Entropy Coding	√	√	√	√	√	√	√
CABAC Entropy Coding	X	X	√	√	√	√	√

APPENDIX B

	Baseline	Main	Extended	High	High10	High 4:2:4	High 4:4:4
Flexible Macroblock Ordering (FMO)	√	√	X	X	X	X	X
Arbitrary Slice Ordering (ASO)	√	√	X	X	X	X	X
Redundant Slices (RS)	√	√	X	X	X	X	X
Data Partitioning	X	√	X	X	X	X	X
Interlaced Coding (PicAFF, MBAFF)	X	√	√	√	√	√	√
4:2:0 Chroma Format	√	√	√	√	√	√	√
Monochrome Video Format (4:0:0)	X	X	X	√	√	√	√
4:2:2 Chroma Format	X	X	X	X	X	√	√

APPENDIX B

	Baseline	Main	Extended	High	High10	High 4:2:4	High 4:4:4
4:4:4 Chroma Format	X	X	X	X	X	X	√
8 Bit Sample Depth	√	√	√	√	√	√	√
9 and 10 Bit Sample Depth	X	X	X	X	√	√	√
11 to 14 Bit Sample Depth	X	X	X	X	X	X	√
8x8 vs. 4x4 Transform Adaptivity	X	X	X	√	√	√	√
Quantization Scaling Matrices	X	X	X	√	√	√	√
Separate Cb and Cr QP control	X	X	X	√	√	√	√
Separate Color Plane Coding	X	X	X	X	X	X	√
Predictive Lossless Coding	X	X	X	X	X	X	√

BIBLIOGRAPHY

[1] ITU-T, "Video Codec for Audiovisual Services at px64 kbit/s," ITU-T Rec. H.261, Version 1: Nov. 1990; Version 2: Mar. 1993.

[2] ISO/IEC JTC 1, "Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s – Part 2: Video," ISO/IEC 11172-2 (MPEG-1), Mar. 1993.

[3] ITU-T and ISO/IEC JTC 1, "Generic coding of moving pictures and associated audio information – Part 2: Video," ITU-T Rec. H.262 – ISO/IEC 13818-2 (MPEG-2 Video), Nov. 1994.

[4] ITU-T, "Video coding for low bit rate communication," ITU-T Rec. H.263; version 1, Nov. 1995; version 2, Jan. 1998; version 3, Nov. 2000.

[5] ISO/IEC JTC 1, "Coding of audio-visual objects – Part 2: Visual," ISO/IEC 14496-2 (MPEG-4 visual version 1), April 1999; Amd. 1 (ver. 2), Feb., 2000; Amd. 2, 2001, Amd. 3, 2001, Amd. 4 (streaming video profile), 2001, Amd 1 to 2nd ed. (studio profile), 2001, Amd. 2 to 2nd ed. 2003.

[6] ITU-T and ISO/IEC JTC 1, "Advanced video coding for generic audiovisual services," ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, 2003.

[7] ITU-T and ISO/IEC JTC 1, "Generic coding of moving pictures and associated audio information – Part 1:Systems," ITU-T Rec. H.222.0 | ISO/IEC 13818-1 (MPEG-2 Systems), Nov. 1994

[8] ITU-T, "Narrow-band Visual Telephone Systems and Terminal Equipment," ITU-T Rec. H.320, 1999.

[9] ITU-T, "Packet-based multimedia communications systems," ITU-T Rec. H.323, 1998.

[10] ITU-T, "Terminal for low bit rate multimedia communication," ITU-T Rec. H.324, 1996.

[11] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session initiation protocol," IETF RFC 3261, 2002.

[12] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications," IETF RFC 1889, 1996.

[13] J. Keith, Video Demystified: a handbook for the digital engineer (3rd Ed), LLH Technology Publishing, Eagle Rock, VA24085 (2001)

[14] H. Enomoto and K. Shibata, "Features of Hadamard transformed television signal," Natl. Conf. of IECE Japan, paper 881, 1965.

[15] H. C. Andrews and W. K. Pratt, "Fourier transform coding of images," Hawaii Intl. Conf. on System Sciences, Sep. 1967.

[16] ITU-T and ISO/IEC JTC 1, "Digital compression and coding of continuous-tone still images", ITU-T Rec. T.81 | ISO/IEC 10918-1 (JPEG), Sep. 1992

[17] R. D. Kell, "Improvements relating to electric picture transmission systems," British Patent No. 341,811, 1929

[18] ITU-T, "Codec for videoconferencing using primary digital group transmission" ITU-T Rec. H.120; version 1, 1984; version 2, 1988, version 3, 1993

[19] F. W. Mounts, "A video encoding system with conditional picture element replenishment," Bell System Technical Journal, vol. 48, no. 7, pp. 2545-2554, Sep. 1969

[20] ITU-T and ISO/IEC JTC 1, "Generic coding of moving pictures and associated audio information – Part 2: Video," ITU-T Rec. H.262 – ISO/IEC 13818-2 (MPEG-2 Video), Nov. 1994

[21] Y. Taki, M. Hatori, and S. Tanaka, "Interframe coding that follows the motion," Inst. of Electronics and Commun. Eng. of Japan Annual Conv. (IECEJ), p. 1263, July 1974.

[22] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," IEEE Trans. On Commun., Vol. COM-29, No. 12, pp. 1799-1808, Dec. 1981.

[23] S. Brofferio and F. Rocca, "Interframe redundancy reduction of video signals generated by translating objects," IEEE Trans. on Commun., Vol. COM-25, pp. 448- 455, Apr 1977.

[24] T. Wedi and H. G. Musmann, "Motion- and aliasing compensated prediction for hybrid video coding," IEEE Trans. on Circuits and Syst. for Video Tech., vol. 13, no. 7, pp. 577-586, July 2003

[25] B. Girod, "The efficiency of motion-compensating prediction for hybrid coding of video sequences," IEEE Journal on Selected Areas in Commun., vol. 5, no. 7, pp. 1140–1154, Aug. 1987.

[26] B. Girod, "Motion-compensating prediction with fractional pel accuracy," IEEE Trans. on Commun., vol. 41, no. 4, pp. 604–612, April 1993.

[27] G. J. Sullivan and R. L. Baker, "Motion compensation for video compression using control grid interpolation," IEEE Intl. Conf. on Acoust., Speech, Signal Proc. (ICASSP), Toronto, Canada, pp. 2713-2716, May 1991.

[28] T. Hidaka, "Description of the proposing algorithm and its score for moving image (a part of the proposal package)," ISO/IEC JTC 1/SC 2/WG 8 MPEG 89/188, Oct 1989.

[29] F. Giorda and A. Racciu, "Bandwidth reduction of video signals via shift vector transmission," IEEE Trans. On Commun., pp. 1002-1004, Sep. 1975.

[30] T. Wiegand, X. Zhang, and B. Girod, "Long-term memory motion-compensated prediction," IEEE Trans. on Circuits and Systems for Video Tech., vol. 9, no. 1, pp. 70-84, Feb. 1999.

[31] T. Wiegand and B. Girod, "Multi-frame motion compensated prediction for video transmission," Kluwer Academic Publishers, Sep. 2001

[32] G. J. Sullivan, "Multi-hypothesis motion compensation for low bit-rate video coding," IEEE Intl. Conf. on Acoust., Speech, Signal Proc. (ICASSP), Minneapolis, MN, pp. 437– 440, Apr. 1993.

[33] M. T. Orchard and G. J. Sullivan, "Overlapped block motion compensation: an estimation-theoretic approach," IEEE Trans. on Image Proc., vol. 3, no. 5, pp. 693–699, Sep. 1994.

[34] M. Flierl, T. Wiegand, and B. Girod, "A locally optimal design algorithm for block-based multi-hypothesis motion compensated prediction," IEEE Data Compression Conference (DCC), Snowbird, UT, USA, pp. 239-248, Mar. 1998.

[35] B. Girod, "Efficiency analysis of multi-hypothesis motion compensated prediction," IEEE Trans. on Image Proc., vol. 9(2), pp. 173–183, Feb. 2000.

[36] C. Reader, "History of MPEG video compression – Ver. 4.0," Joint Video Team (JVT) doc. JVT-E066, Oct., 2002

[37] G. Sullivan and T. Wiegand, “Video compression – From concepts to the H.264/AVC standard”, IEEE Proc. Of the IEEE, Dec. 2004

[38] Richardson, I., H.264 and MPEG-4 Video Compression – Video Coding for Next-generation Multimedia, John Wiley & Sons, 2003.

[39] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, “Overview of the H.264/AVC video coding standard, ” IEEE Transactions on Circuits, System and Video Technology, vol. 7, no. 1-19, July 2003

[40] VideoBits.org, The educational forum on digital video, http://videobits.org/index.html

[41] N. Ahmed, T. Natarajan, and K. R. Rao, "On image processing and a discrete cosine transform," IEEE Trans. On Computers, vol. C-23, no. 1, pp.90-93, Jan 1974.

[42] H. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, “Low Complexity transform and quantization in H.264/AVC”, IEEE Transactions on Circuits and System for Video Technology, vol. 13, pp. 674-687, July 2003.

[43] I. Amer, W. Badawy and G. Jullien, “Hardware prototyping for the H.264 4x4 Transformation”, IEEE Intl. Conf. on Acoust. Speech, Signal Proc. ICASP 2004, pp v77-v80

[44] O. Tasdizen and I. Hamzaoglu, “A High Performance and Low-Cost Hardware Architecture for H.264 Transform and Quantization Algorithms”, 13th European Signal Processing Conference, EUSIPCO2005.

[45] O. Tasdizen and I. Hamzaoglu, “Towards MPEG-4 Part 10 System on Chip: A VLSI Prototype For Context-based Adaptive Variable Length Coding (CAVLC)”, IEEE Workshop on Signal Processing Systems, SIPS2004, pp 275-279.

[46] D. Marpe, H. Schwarz and T. Wiegand, "Context-adaptive binary arithmetic coding for H.264/AVC", IEEE Transactions on Circuits and System for Video Technology, vol. 13, no7, pp. 598-603, July 2003.

[47] P. List, A. Joch, J. Lainema, G. Bjøntegaard, M. Karczewicz, "Adaptive deblocking filter," IEEE Trans. on Circuits and Syst. for Video Tech., vol. 13, no. 7, pp. 614-619, July 2003.

[48] M. Flierl and B. Girod, "Generalized B pictures and the draft JVT/H.264 video compression standard," IEEE Trans. On Circuits and Syst. for Video Tech., vol. 13, no. 7, pp. 587- 597, July 2003

[49] H. Everett, "Generalized Lagrange multiplier method for solving problems of optimum allocation of resources," Operations Research, vol. 11, pp. 399-417, 1963.

[50] Y. Shoham and A. Gersho, "Efficient bit allocation for an arbitrary set of quantizers," IEEE Trans. on Acoust., Speech, Signal Proc., vol. 36, pp. 1445-1453, Sep. 1988.

[51] P. A. Chou, T. Lookabaugh, and R. M. Gray, "Entropy-constrained vector quantization," IEEE Trans. on Acoust., Speech, Signal Proc., vol. 37, no. 1, pp. 31--42, Jan. 1989.

[52] G. J. Sullivan and R. L. Baker, "Rate-distortion optimized motion compensation for video compression using fixed or variable size blocks," IEEE Global Telecommun. Conf. (GLOBECOM), pp. 85-90, Dec. 1991.

[53] S.-W. Wu and A. Gersho, "Enhanced video compression with standardized bitstream syntax," IEEE Intl. Conf. Acoust., Speech, Signal Proc., Minneapolis, MN, USA, vol. 1, pp. 103-106, April, 1993.

[54] K. Ramchandran, A. Ortega, and M. Vetterli, "Bit allocation for dependent quantization with applications to multi-resolution and MPEG video coders," IEEE Trans. On Image Proc., vol. 3, no. 5, pp.533-545, Sep. 1994.

[55] A. Ortega, K. Ramchandran, and M. Vetterli, "Optimal trellis-based buffered compression and fast approximations," IEEE Trans. on Image Proc., vol. 3, no. 1, pp. 26-40, Jan. 1994.

[56] T. Wiegand, M. Lightstone, D. Mukherjee, T. G. Campbell, and S. K. Mitra, "Rate-distortion optimized mode selection for very low bit rate video coding

and the emerging H.263 standard," IEEE Trans. on Circuits and Syst. for Video Tech., vol. 6, no. 2, pp. 182-190, Apr. 1996.

[57] M. C. Chen and A. N. Willson, "Design and optimization of a differentially coded variable block size motion compensation system," IEEE Intl. Conf. on Image Proc., Lausanne, Switzerland, vol. 3, pp. 259-262, Sep. 1996.

[58] G. M. Schuster and A. K. Katsaggelos, "A video compression scheme with optimal bit allocation among segmentation, motion, and residual error," IEEE Trans. On Image Proc., vol. 6, pp. 1487-1502, Nov. 1997.

[59] M. C. Chen and A. N. Willson, "Rate-distortion optimal motion estimation algorithms for motion-compensated transform video coding," IEEE Trans. on Circuits and Syst. for Video Tech., vol. 8, no. 2, pp. 147--158, Apr. 1998.

[60] A. Ortega and K. Ramchandran, "Rate-distortion methods for image and video compression: An Overview," IEEE Signal Proc. Magazine, pp. 23-50, Nov. 1998.

[61] G. J. Sullivan and T. Wiegand "Rate-distortion optimization for video compression," IEEE Signal Proc. Magazine, vol. 15, pp. 74-90, Nov. 1998.

[62] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. J. Sullivan, "Rate-constrained coder control and comparison of video coding standards," IEEE Trans. On Circuits and Syst. for Video Tech., vol. 13, no. 7, pp. 688-703, July 2003.

[63] Wikipedia.org, The free encyclopedia, http://www.wikipedia.org/

[64] T. Toivonen, J. Heikkila, "A new rate – minimizing matching criterion and a fast algorithm for block motion estimation", International Conference on Image Processing (ICIP 2003).

[65] V. Bhaskaran and K. Konstantinides, Image and Video Compression Standards, Algorithms and Architectures (2nd Ed), Kluwer Academic Publishers, Norwell, Massachusetts (1997)

[66] A.N. Netravali and B.G. Haskell, Digital Pictures: Representation, Compression, and Standards (2nd Ed), Plenum Press, New York, NY (1995).

[67] M. Rabbani and P.W. Jones, Digital Image Compression Techniques, Vol TT7, SPIE Optical Engineering Press, Bellvue, Washington (1991).

[68] Joint Video Team (JVT) Reference Software JM11.0, at http://iphome.hhi.de/suehring/tml/download/

[69] Heinrich-Hertz-Institut, The Fraunhofer-Institute for Telecommunications, at http://www.hhi.fraunhofer.de/english/index.html

[70] A. M. Tourapis, "Enhanced Predictive i!onal Search for Single and Multiple Frame Motion Estimation", in proceedings of Visual Communications and Image Processing 2002 (VCIP - 2002), pp. 1069-79, San Jose, CA, Jan'O2

[71] H. C. Tourapis and A. M. Tourapis, "Fast Motion Estimation within the H.264 Codec", IEEE International Conference on Multimedia & Expo (ICME) 2003

[72] VQEG – Video Quality Experts Group http://www.its.bldrdoc.gov/vqeg/

[73] C Wang, P. Lee, C. Wu, and H. Wu, "High Fan-In Dynamic CMOS Comparators with Low Transistor Count", IEEE Transactions on Circuits and Systems – I: Fundamental theory and applications, vol. 50, no. 9, pp. 1216-1220, September 2003.

[74] Deng, L., Gao, W., Hu, Z., and Ji, Z., "An Efficient Hardware Implementation for Motion Estimation of AVC Standard", IEEE Transactions on Consumer Electronics, vol.51, no.4, pp.1360-1366, November 2005.

[75] Ou, C., Le, C., and Hwang, W., "An Efficient VLSI Architecture for H.264 Variable Block Size Motion Estimation", IEEE Transactions on Consumer Electronics, vol.51, no.4, pp.1291-1299, November 2005.

[76] Yap, S., McCanny, J., "A VLSI Architecture for Variable Block Size Video Motion Estimation", IEEE Transactions on Circuits and Systems – II: Express Briefs, vol. 51, no. 7,pp. 384-389, July 2004.

[77] Sayed, M., Amer, I., and Badawy, W., "Towards an H.264/AVC Full Encoder on Chip: An Efficient Real-Time VBSME ASIC Chip", International Symposium on Circuits and Systems (ISCAS 2006).

[78] Song, Y., Liu, Z., Goto, S., and Ikenaga, T., "Scalable VLSI Architecture for Variable Block Size Integer Motion Estimation in H.264/AVC", IEICE Trans. Fundamentals, vol. E89-A, no. 4, pp. 979-988, April 2006.

[79] Chen, C., Chien, S., Huang, Y., Chen, T., Wang, T., and Chen, L., "Analysis and Architecture Design of Variable Block-Size Motion Estimation for

H.264/AVC", IEEE Transactions on Circuits and Systems – I: Regular Papers, vol. 53, no. 2, pp. 578-593, February 2006.

[80] PrimePower Manual, Version V-2004.06, June 2004, Synopsys, Inc.

[81] Cao Wei, Mao Zhi Gang, Lv Zhi Qiang, Zhang Yan, "VLSI architecture design for variable-size block motion estimation in MPEG-4 AVC/H.264", IEEE Asia-Pacific Conference on Circuits and Systems. Proc., 6-9 December 2004, pp.617 – 620.

[82] J. F. Shen et al., "A novel low-power full-search block-matching motion estimation design for H.263+", *IEEE Trans.* Circuits Syst. Video Technol., 2001, 7. 890–897.

[83] Yu-Wen Huang, Tu-Chih Wang, Bing-Yu Hsieh, "Hardware Architecture Design for Variable Block Size Motion Estimation in MPEG-4 AVC/JVT/ITU-T H.264", ISCAS'03, International Symp., 2003, pp. 796-799.

[84] L.deVos, M. Schobinger, "VLSI architecture for a flexible block matching processor", *IEEE Trans.* Circuits Syst.Video Technol., 1995, 5. 417–428.

[85] Minho. Kim, Ingu Hwang, Soo-Ik. Chae, "A Fast VLSI Architecture for Full-Search Variable Block Size Motion Estimation in MPEG-4 AVC/H.264", ASP-DAC, 2005, PP.631-634.

[86] Cao Wei, Mao Zhi Gang, "A Novel VLSI Architecture for VBSME in MPEG-4 AVC/H.264", ISCAS 2005. IEEE International Symp., 23-26 May 2005, pp.1794 – 1797.

BIBLIOGRAPHY