



الجمهورية العربية السورية
جامعة دمشق
كلية الهندسة الميكانيكية والكهربائية
قسم هندسة الحواسيب والأتمتة

**ضغط و تشفير الفيديو باستخدام تقانة مصفوفات البوابات
القابلة للبرمجة حقلياً (FPGA)**
**Compression and Encryption of Videos Using Field
Programmable Gate Arrays (FPGA) Technique**

رسالة أعدت لنيل درجة الماجستير في هندسة الحواسيب وشبكاتها

إعداد
المهندسة أمل عطا الله

المشرف
الدكتور المهندس كمال قمر

العام الدراسي
2014-2013

لجنة الحكم

قرار مجلس البحث العلمي والدراسات العليا رقم /2335/ المتخذ بالجلسة رقم /15/

تاريخ 2014/5/19

- د. محمد خالد شاهين الأستاذ المساعد في قسم هندسة الالكترونيات والاتصالات عضواً
- د. كمال قمر المدرس في قسم هندسة الحواسيب والأتمتة عضواً مشرفاً
- د. محمد صالحاني المدرس في قسم هندسة الحواسيب والأتمتة عضواً

شكر وتقدير

في البداية أشكر الله تعالى الذي منّ علي بالنجاح والتوفيق في إتمام هذا العمل.

وانطلاقاً من العرفان بالجميل، فإنه ليسرني أن أتقدم بالشكر والامتنان إلى أستاذي، ومشرفي الدكتور **كمال قمر** الذي أخذ بيدي خطوة بخطوة و لم يبخل عليّ يوماً بجهدته أو وقته حتى تمكنت من إنجاز هذا البحث.

كما أتقدم بجزيل الشكر إلى أساتذتي أعضاء لجنة الحكم الموقرين الدكتور **محمد خالد شاهين** والدكتور **محمد صالحاني** على ما تكبدوه من عناء في قراءة رسالتي المتواضعة وإغنائها بمقترحاتهم القيمة.

وأتقدم كذلك بجزيل الشكر إلى كل أساتذة كلية الهندسة الميكانيكية والكهربائية وأخص بالذكر العميد الدكتور **حسين تينة** ورئيس قسم هندسة الحواسيب والأتمتة الدكتور **مأمون يونس** لكل ما قدموه لي من مساعدة ومساندة مكننتني من المضي في مسيرتي العلمية.

وفي النهاية يسرني أن أتقدم بجزيل الشكر إلى كل من مد لي يد العون في إنجاز بحثي هذا وأخص بالذكر الدكتور في البحوث العلمية **داود قره كولا** لما قدمه لي من مساعدة.

الإهداء

إلى من علمني النجاح والصبر أبي الرحيم.
إلى من تحت قدمها تكمن الجنة أُمي الحنون.
إلى من ساندني وآزرني في دربي زوجي الغالي.
إلى سندي وقوتي وملأني بعد الله إخوتي وفقهم الله.
إلى من قاسموني الأفراح والأحزان أقاربي الأعزاء.
إلى كل من علمني حرفاً أصبح سناً بركة يضيء الطريق أمامي.

فهرس المحتويات

١	فهرس المحتويات.....
٤	فهرس الجداول
٥	فهرس الأشكال.....
٨	فهرس المصطلحات.....
١٠	قائمة الاختصارات.....
١١	ملخص الأطروحة.....
١٢	مقدمة عامة.....
١٤	الفصل الأول: أساسيات في ضغط الفيديو.....
١٦	١-١-١- الفيديو الرقمي.....
١٦	١-١-١-١- الميز وأخذ العينات Resolution & Sampling.....
١٧	١-١-٢- فضاءات اللون Color Sppaces.....
١٩	٢-١- المخطط الصندوقي العام لضغط الفيديو.....
٢٠	١-٢-١- وحدة التنبؤ الداخلي Intra Prediction Unit.....
٢١	٢-٢-١- وحدة التنبؤ البيني Inter Prediction Unit.....
٢٢	٣-٢-١- وحدة التحويل Transform Unit.....
٢٣	٤-٢-١- وحدة التكمية Quantization Unit.....
٢٤	٥-٢-١- وحدة الترميز الانتروبي Entropy Coding Unit.....
٢٤	٣-١- معايير ضغط الفيديو.....
٢٧	الفصل الثاني: المعيار H.264.....
٢٨	١-٢- نظام الضغط الفيديوي في المعيار H.264.....
٢٨	١-٢-١- الأشكال الجانبية Profiles.....
٢٩	١-٢-٢- بنية الصورة Picture Structure.....
٢٩	٢-٢-١- طبقة موائمة الشبكة NAL.....
٣٠	٢-٢-١-٢- وحدة الضاغط (المرمّز) H.264 Encoder.....
٣٢	٢-٢-١-٢- وحدة مفكك الضغط H.264 Decoder.....
٣٢	٢-٢-٢- مكونات وحدة الضغط H.264.....
٣٣	١-٢-٢- التنبؤ الداخلي Intra Prediction.....
٣٥	٢-٢-٢- التنبؤ البيني Inter Prediction.....

٤١	 Transform/Quantitation التحويل والتكمية ٣-٢-٢
٤٣	 Re-ordering/Scanning إعادة الترتيب ٤-٢-٢
٤٤	 Entropy Coding الترميز الانتروبي ٥-٢-٢
٤٥	 De-blocking filter مرشح التنعيم ٦-٢-٢
٤٧	 الفصل الثالث: تصميم مفكك الضغط
٤٨	 ١-٣ دراسة مرجعية حول التصميم العتادي لمفككات ضغط الفيديو وفق المعيار H.264
٤٨	 ١-١-٣ إضافة التوازي Parallelism والعالجة التواردية Pipeline
٥٠	 ٢-١-٣ استخدام بنية متعددة النواة Multi-Core Architecture
٥٢	 ٣-١-٣ أمثلة الذاكرة Memory Optimization
٥٥	 ٢-٣ المخطط الصندوقي لمفكك الضغط H.264 decoder
٥٥	 ١-٢-٣ وحدة الذاكرة Memory Block
٥٦	 ٢-٢-٣ وحدة الإظهار Video Display
٥٧	 ٣-٢-٣ وحدة تفكيك ضغط الفيديو Video Decoder
٥٧	 ٤-٢-٣ إدارة عمل وحدات النظام
٥٨	 ٣-٣ وحدة التحكم بقراءة مقطع الفيديو المضغوط
٥٩	 ١-٣-٣ وحدة التحكم Bitstream Controller FSM
٦١	 ٢-٣-٣ كاشف البت الواحد Heading-one Detector
٦١	 ٣-٣-٣ مفكك الترميز الانتروبي من نوع CAVLC
٦٥	 ٤-٣ وحدة إعادة بناء الفيديو reconstruction datapath
٦٦	 ١-٤-٣ وحدة التكمية والتحويل العكسيين IQIT
٦٩	 ٢-٤-٣ وحدة التنبؤ الداخلي Intra Pred
٧١	 ٣-٤-٣ وحدة التنبؤ البيئي Inter Pred
٧٤	 ٤-٤-٣ وحدة مرشح التنعيم
٧٦	 الفصل الرابع: دراسة خوارزمية التعمية المتقدمة AES وتصميمها
٧٧	 ١-٤ مقدمة
٧٩	 ٢-٤ الخوارزمية Rijndael أو AES
٧٩	 ١-٢-٤ المصطلحات و الاختصارات
٨٠	 ٢-٢-٤ الرموز والتوابع
٨١	 ٣-٢-٤ بعض الخواص الرياضية المستخدمة
٨٢	 ٣-٤ مراحل الخوارزمية
٨٢	 ١-٣-٤ الوحدة aes-demo

٨٣	aes-init الوحدة ٢-٣-٤
٨٧	 cipher الوحدة ٣-٣-٤
٨٨	 inv_cipher الوحدة ٤-٣-٤
٨٩	AES تصميم الخوارزمية ٤-٤-٤
٩٢	الفصل الخامس: التنفيذ العملي
٩٣	١-٥ مقدمة
٩٣	٢-٥ المنهجية العامة للتصميم باستخدام تقانة FPGA
٩٥	٣-٥ بطاقة التطوير DE2_115
٩٩	٤-٥ الكاميرا الرقمية TRDB-D5M
١٠٠	٥-٥ تصميم النظام
١٠٤	٦-٥ اختبار النظام
١٠٤	١-٦-٥ التطبيق الأول: تعمية مقطع فيديو مخزن ضمن ذاكرة وفك تعميته
١١٠	٢-٦-٥ التطبيق الثاني: تعمية مقطع فيديو مقروء من كاميرا وفك تعميته
١١٣	٣-٦-٥ التطبيق الثالث: فك ضغط مقطع فيديو مضغوط ومخزن ضمن ذاكرة
١١٩	٤-٦-٥ التطبيق الرابع: تعمية وفك تعمية وفك ضغط مقطع فيديو مضغوط ومخزن ضمن ذاكرة
١٢١	٧-٥ مقارنة النتائج
١٢٤	الخاتمة والآفاق المستقبلية
١٢٦	المراجع References
١٣٣	الملحق A: جداول مفك الترميز الانتروبي CAVLC
١٣٧	الملحق B: وحدة التنبؤ الداخلي بلغة VHDL
١٤٨	الملحق C: وحدة التنبؤ البيئي بلغة VHDL
١٥٧	الملحق D: وحدة خلط الأعمدة في AES بلغة VHDL
١٦٠	Abstract

فهرس الجداول

١٧	الجدول (1.1): بعض الدقات المستخدمة في معيار CIF
٤٤	الجدول (1.2): الترميز Exp-Golomb
٤٥	الجدول (2.2): ترميز المعاملات غير الصفرية
٤٥	الجدول (3.2): ترميز المعامل Total_zeros
١٠٦	الجدول (1.3): عناصر الكتلة الماكروية MB
١٠٨	الجدول (2.3): اختيار جدول ترميز المعامل Total_coeff تبعاً لقيمة العدد nC
٦٤	الجدول (3.3): ترميز عدد الاصفار التي تسبق كل معامل غير صفري
٦٥	الجدول (4.3): مراحل فك الترميز CAVLC
١٤٢	الجدول (1.5): المواصفات الزمنية لإشارة التزامن الأفقي
٩٧	الجدول (2.5): المواصفات الزمنية لإشارة التزامن العمودي
١١٠	الجدول (3.5): عدد العناصر المستهلكة في التطبيق الأول
١١٠	الجدول (4.5): حساب زمن وسرعة عملية التعمية وفك التعمية
١١٣	الجدول (5.5): عدد العناصر المستهلكة في التطبيق الثاني
١٦٤	الجدول (6.5): عدد العناصر المستهلكة في التطبيق الثالث
١١٩	الجدول (7.5): مقارنة زمن فك الضغط مع زمن المسح
١٢٠	الجدول (8.5): عدد العناصر المستهلكة في التطبيق الرابع
١٢١	الجدول (9.5): عدد العناصر المستهلكة لمفكك الضغط من شركة AsicSoft
١٢٢	الجدول (10.5): عدد العناصر المستهلكة لمفكك الضغط من جامعة Turku
١٢٣	الجدول (11.5): مقارنة أزمنة فك ضغط الفيديو
١٣٣	الجدول (1.A): جداول ترميز الزوج (Total_coeff, T1)
١٣٤	الجدول (2.A): جدول ترميز total_zeros
١٨٠	الجدول (3.A): جدول ترميز run_before
١٣٦	الجدول (4.A): جداول ترميز العناصر غير الصفرية

فهرس الأشكال

١٥	الشكل (1.1): معايير ضغط الفيديو
١٦	الشكل (2.1): الصورة الرقمية
١٨	الشكل (3.1): نمط أخذ العينات
١٨	الشكل (4.1): ترتيب قراءة العينات بحسب نمط أخذ العينات
٢٠	الشكل (5.1): المخطط الصندوقي العام لضغط الفيديو
٢٠	الشكل (6.1): مثال عن عملية التنبؤ الداخلي
٢١	الشكل (7.1): مثال عن التنبؤ البيئي
٢٢	الشكل (8.1): ترتيب الصور في مقطع الفيديو
٢٢	الشكل (9.1): تحويل التجيب المتقطع
٢٣	الشكل (10.1): عملية التكمية والتكمية العكسية
٢٤	الشكل (11.1): إعادة ترتيب عينات الكتلة 8×8 بشكل تعرجي zig-zag
٢٩	الشكل (1.2): بنية الصورة
٢٩	الشكل (2.2): بنية طبقة نقل الفيديو
٣٠	الشكل (3.2): مثال عن محتوى طبقة نقل الفيديو
٣٠	الشكل (4.2): المخطط الصندوقي لوحدة الضاغط H.264
٣٢	الشكل (5.2): مخطط وحدة مفكك الضغط H.264
٣٣	الشكل (6.2): عينات الإضاءة المجاورة لكتلة إضاءة ماكروية 4×4
٣٣	الشكل (7.2): اتجاهات أنماط التنبؤ الداخلي
٣٤	الشكل (8.2): أنماط التنبؤ التسعة من أجل كتلة إضاءة ماكروية بقياس 4×4
٣٥	الشكل (9.2): أنماط التنبؤ الأربعة من أجل كتلة ألوان ماكروية بقياس 4×4
٣٥	الشكل (10.2): مثال أول عن التنبؤ البيئي
٣٦	الشكل (11.2): مثال ثاني عن التنبؤ البيئي
٣٧	الشكل (12.2): ترتيب الصور ضمن مجموعات GOP
٣٧	الشكل (13.2): تقسيم الكتل الماكروية إلى كتل جزئية
٣٨	الشكل (14.2): مثال عن تقسيم الكتل الماكروية
٣	الشكل (15.2): مثال عن أشعة الحركة
٣٩	الشكل (16.2): حساب البيكسلات الجزئية
٤١	الشكل (17.2): مثال عن تطبيق التحويل DCT
٤٢	الشكل (18.2): إعادة الترتيب بطريقة تعرجية zig-zag
٤٣	الشكل (19.2): إرسال المعاملات المستمرة
٤٦	الشكل (20.2): تأثير ترشيح الحواف على تحسين الصورة
٤٩	الشكل (1.3): المخطط الصندوقي لمفكك الضغط H.264 [31]
٥١	الشكل (2.3): البنية متعددة النواة المستخدمة في [30]
٥١	الشكل (3.3): طرق توزيع العمل [30]
٥٢	الشكل (4.3): ارتباط المعطيات بين الكتل الماكروية [30]
٥٢	الشكل (5.3): تجزئة إطار المعطيات [30]
٥٣	الشكل (6.3): دمج طريقتي تجزئة المعالجة وتجزئة الإطار [30]
٥٤	الشكل (7.3): تقنية ضغط وفك ضغط الإطارات المرجعية [35]
٥٤	الشكل (8.3): تطبيق تقنية المعالجة الأنبوبية على مستوى الكتل الماكروية
٥٥	الشكل (9.3): المخطط الصندوقي لمفكك الضغط

٥٨	الشكل (10.3): ترتيب عمل وحدة الإظهار ومفكك الضغط
٥٨	الشكل (11.3): المخطط الصندوقي لوحدة التحكم بقراءة مقطع الفيديو المضغوط
٥٩	الشكل (12.3): مخطط آلة الحالة للتحكم بقراءة مقطع الفيديو المضغوط
٦٠	الشكل (13.3): تقسيم معطيات الفيديو
٦١	الشكل (14.3): كاشف البت الواحد
٦٢	الشكل (15.3): مخطط مفكك الترميز الانتروبي
٦٤	الشكل (16.3): القيم التي يأخذها العد <code>Run_before</code> في المثال المذكور
٦٦	الشكل (17.3): المخطط الصندوقي لوحدة إعادة بناء الفيديو
٦٧	الشكل (18.3): تصميم الكيان الصلب لتحويل 4×4 Hadamard العكسي بطريقة الفراشة
٦٧	الشكل (19.3): تصميم الكيان الصلب لتحويل 2×2 Hadamard العكسي بطريقة الفراشة
٦٨	الشكل (20.3): تصميم الكيان الصلب لتحويل 4×4 IDCT بطريقة الفراشة
٦٩	الشكل (21.3): إرسال كتل المتبقي
٦٩	الشكل (22.3): خوارزمية IQIT
٧٠	الشكل (23.3): المخطط الصندوقي لوحدة التنبؤ الداخلي
٧١	الشكل (24.3): الأنماط الفرعية للتنبؤ الداخلي
٧٢	الشكل (25.3): استقراء مركبات الإضاءة بدقة $1/2$ بيكسل
٧٢	الشكل (26.3): استقراء المركبات اللونية بدقة $1/8$ بيكسل
٧٣	الشكل (27.3): تقدير شعاع الحركة
٧٤	الشكل (28.3): المخطط الصندوقي لوحدة التنبؤ البيئي
٧٥	الشكل (29.3): تطبيق مرشح التنعيم على الحواف
٧٧	الشكل (1.4): نظام ضغط الفيديو و إرساله وفك ضغطه
٧٩	الشكل (2.4): نظام ضغط الفيديو و تعميته
٨٣	الشكل (3.4): المخطط الصندوقي لعملية محاكاة الخوارزمية AES ضمن بيئة Matlab
٨٣	الشكل (4.4): المخطط الصندوقي لعمل وحدة الـ <code>aes_init</code>
٨٤	الشكل (5.4): مخطط توليد جداول التبديل
٨٤	الشكل (6.4): جدول التبديل المباشر <code>s_box</code>
٨٥	الشكل (7.4): جدول التبديل العكسي <code>inv_s_box</code>
٨٥	الشكل (8.4): مصفوفة الدوران الثابتة <code>rcon</code>
٨٦	الشكل (9.4): مخطط عمل تابع توسيع المفتاح
٨٦	الشكل (10.4): تحويل مفتاح التعمية من شعاع إلى مصفوفة 4×4
٨٧	الشكل (11.4): توليد جدول المفاتيح <code>w</code>
٨٧	الشكل (12.4): المصفوفتان <code>poly_mat</code> , <code>inv_poly_mat</code>
٨٨	الشكل (13.4): المخطط الصندوقي لعملية التعمية
٨٨	الشكل (14.4): مصفوفة الحالة
٨٩	الشكل (15.4): إزاحة أسطر مصفوفة الحالة باستخدام تابع الـ <code>shift_rows</code>
٩٠	الشكل (16.4): المخطط الصندوقي لعملية فك التعمية
٩٣	الشكل (1.5): العناصر التي تتألف منها شريحة FPGA
٩٤	الشكل (2.5): العنصر المنطقي LE
٩٤	الشكل (3.5): منهجية التصميم باستخدام تقانة FPGA
٩٥	الشكل (4.5): بطاقة التطوير DE2_115
٩٦	الشكل (5.5): الوصلات بين شريحة FPGA و VGA
٩٦	الشكل (6.5): المخطط الصندوقي للبطاقة DE2_115
٩٧	الشكل (7.5): إشارة التزامن الأفقي VGA_HS
٩٨	الشكل (8.5): الشريحة EP4CE115F29C8N

٩٩	الشكل (9.5): ربط الكاميرا TRDB_D5M بالبطاقة DE2_115
٩٩	الشكل (10.5): توضع بيكسلات الألوان ضمن الأسطر
١٠٠	الشكل (11.5): طريقة قراءة الصورة زمنياً
١٠٠	الشكل (12.5): استيفاء اللون الأخضر
١٠٢	الشكل (13.5): المخطط الصندوقي للتطبيق النهائي على شريحة FPGA
١٠٣	الشكل (14.5): منهجية العمل المتبعة
١٠٤	الشكل (15.5): المخطط الصندوقي للتطبيق الأول
١٠٤	الشكل (16.5): المخطط الصندوقي للتطبيق الأول على شريحة FPGA
١٠٥	الشكل (17.5): مداخل و مخارج المُعَمِّي
١٠٥	الشكل (18.5): مداخل ومخارج مفكك المُعَمِّي
١٠٧	الشكل (19.5): المخطط الزمني للمُعَمِّي
١٠٨	الشكل (20.5): المخطط الزمني لمفكك التعمية
١٠٩	الشكل (21.5): خرج التطبيق الأول عند استخدام نفس المفتاح للتعمية وفك التعمية
١٠٩	الشكل (22.5): خرج التطبيق الأول عند عدم استخدام نفس المفتاح للتعمية وفك التعمية
١١١	الشكل (23.5): المخطط الصندوقي للتطبيق الثاني
١١٢	الشكل (24.5): متحكم الكاميرا Camera Controller
١١٢	الشكل (25.5): المخطط الصندوقي للتطبيق الثاني على شريحة FPGA
١١٣	الشكل (26.5): خرج التطبيق الثاني
١١٤	الشكل (27.5): المخطط الصندوقي للتطبيق الثالث
١١٤	الشكل (28.5): المخطط الصندوقي للتطبيق الثالث على شريحة FPGA
١١٥	الشكل (29.5): مداخل ومخارج مفكك الضغط
١١٦	الشكل (30.5): مخطط خوارزمية التطبيق الثالث
١١٧	الشكل (31.5): المخطط الزمني لمفكك الضغط
١١٨	الشكل (32.5): خرج التطبيق الثالث
١٢٠	الشكل (33.5): المخطط الصندوقي للتطبيق الرابع
١٢٠	الشكل (34.5): خرج التطبيق الرابع
١٢١	الشكل (35.5): المخطط الصندوقي لمفكك الضغط من شركة AsicSoft
١٢٢	الشكل (36.5): المخطط الصندوقي لمفكك الضغط من شركة Cast
١٣٧	الشكل (1.B): أنماط التنبؤ الداخلي لكثافة إضاءة بقياس 4×4
١٣٨	الشكل (2.B): وحدة المعالجة في التنبؤ الداخلي
١٣٩	الشكل (3.B): خرج وحدة التنبؤ الداخلي
١٤٣	الشكل (4.B): مستويات وحدة المعالجة الأولى في الحلقة الأولى
١٤٤	الشكل (5.B): مستويات وحدة المعالجة الأولى في الحلقة الثانية
١٤٤	الشكل (6.B): مستويات وحدة المعالجة الثانية في الحلقة الأولى
١٤٨	الشكل (1.C): استقرار مركبات الإضاءة بميزر 1/2 و 1/4 بيكسل
١٤٩	الشكل (2.C): بنية المرشح FIR
١٥١	الشكل (3.C): مراحل الحصول على خرج التنبؤ البيني

فهرس المصطلحات

Advanced Encryption Standard	معيار التعمية المتقدمة
Cipher Key	مفتاح التعمية
Cipher Text	النص المُعمى
Coding	الترميز
Compression	الضغط
Deblocking Filter	مرشح التنعيم
Decoding	فك الترميز
Decompression	فك الضغط
Decryption	فك التعمية
Encryption	التعمية
Field Programmable Gate Arrays	مصفوفات البوابات القابلة للبرمجة حقلياً
Forward Path	المسار المباشر
Frame	إطار
Inter Prediction	التنبؤ البيني
Intra Prediction	التنبؤ الداخلي
Macro Block	الكتلة الماكروية
Motion Compensation	تعويض الحركة
Motion Estimation	التنبؤ بالحركة
Motion Vector	شعاع الحركة
Network Adaption Layer	طبقة موائمة الشبكة
Plain Text	النص الواضح
Quantization	التكمية
Real Time	الزمن الحقيقي
Residual	المتبقي
Resolution	الميز
Reverse Path	المسار العكسي
Scanning	المسح

State	الحالة
Transform	التحويل
Video Coding Layer	طبقة ترميز الفيديو

قائمة الاختصارات

AES	Advanced Encryption Standard
ATM	Automated Teller Machine
AVC	Advanced Video Coding
CABAC	Context-based Adaptive Binary Arithmetic Coding
CAVLC	Context-based Adaptive Variable Length Coding
CIF	Common Intermediate Format
DC	Direct Confessions
DCT	Discrete Cosine Transform
DES	Data Encryption Standard
FPGA	Field programmable gate arrays
GF	Galois Field
GOB	Group Of Blocks
HD	High Definition
HDTV	High Definition Television
ISDN	Integrated Services Digital Networks
ISO	International Organization for Standardization
ITU-T	International Telecommunications Union, Telecommunication Standardization Sector
JM	Joint Model (H.264/AVC reference test model codec)
JPEG	Joint Photographic Experts Group
JVT	Joint Video Team
LUT	Look-up table
MB	Macroblock
ME	Motion Estimation
MPEG	Motion Picture Experts Group
MSE	Mean Squared Error
MV	Motion Vector
MVD	Motion Vector Difference
MVP	Motion Vector Predictor
NAL	Network Abstraction Layer
PPS	Picture Parameter Set
QCIF	Quarter Common Intermediate Format
RBSP	Raw Byte Sequence Payload
SAD	Sum of Absolute Differences
SPS	Sequence Parameter Set
VHDL	Very high speed integrated circuits Hardware Description Language

ملخص الأطروحة

تهدف هذه الرسالة إلى بناء كيان صلب يعمل بالزمن الحقيقي لمُعَمّي ومفكك تعمية و مفكك ضغط فيديو باستخدام تقنية مصفوفات البوابات القابلة للبرمجة حقيقياً (FPGA) .Field Programmable Gate Arrays

يقوم هذا النظام بتعمية فك تعمية مقطع فيديو مضغوط وفق المعيار H.264 باستخدام معيار التعمية المتقدمة (AES) Advanced Encryption Standard ثم فك ضغطه. يتألف مقطع الفيديو من 300 إطار وبمیز 176×144 بيكسل (pixel).

قمنا في المرحلة الأولى بدراسة معيار الضغط H.264، ومكونات نظام الضغط وفك الضغط. ثم عملنا على تصميم مفكك الضغط انطلاقاً من بعض التصاميم الجاهزة بلغة Verilog لمكونات نظام الضغط واستكمال الأجزاء الناقصة بلغة VHDL.

درسنا في المرحلة الثانية معيار التعمية AES، وقمنا بتصميم وحدتي التعمية وفك التعمية بلغة VHDL. وانطلاقاً من المرحلتين السابقتين حصلنا على نظام يقوم بقراءة مقطع فيديو مضغوط ومُعَمّي ومخزن ضمن ذاكرة، ثم يقوم النظام بفك تعميته وفك ضغطه ومن بعدها يعرضه على الشاشة.

صممنا وحدات النظام باللغتين VHDL و Verilog. وقمنا بمحاكاة النظام ضمن بيئة ModelSim، وبرمجته على شريحة FPGA من نوع Cyclone vi e تتسع لما يقارب 115000 عنصر منطقي وتوجد على بطاقة التطوير DE2_115.

استهلك مفكك الضغط 50% من العناصر المنطقية الموجودة في شريحة FPGA و 40% من ذاكرتها الداخلية، بينما استهلك مفكك التعمية 6% فقط من العناصر المنطقية و 2% من الذاكرة الداخلية. يحتاج مفكك الضغط إلى حجم كبير من الذاكرة لتخزين آخر إطاريّ فيديو جرت معالجتهما. كما يستهلك مفكك الضغط عدداً مهماً من العناصر المنطقية لأنه يقوم بعمليات حسابية معقدة كالنتبؤ البيني والترميز الانتروبي. يحتاج النظام زمناً وقدره تقريباً 2.2 ms لفك ضغط إطار واحد. لذلك فهو يعمل بالزمن الحقيقي لأن زمن معالجة إطار واحد أقل من زمن مسح الصورة والذي يعادل 4.6 ms من أجل تردد مسح يساوي 25 MHz.

مقدمة عامة :General Introduction

يشهد عالمنا المعاصر ثورة في تقانة الاتصالات الرقمية وتعدّ تطبيقات الفيديو من أهم التطبيقات المستخدمة في هذا المجال، حيث تستخدم في أغلب هذه التطبيقات تقنيات تسمح بضغط الفيديو من أجل نقله أو حفظه بطريقة فعالة. وبالرغم من سرعات النقل العالية المتاحة عبر الانترنت أو غيرها من الوسائط فهي تبقى غير كافية لنقل الفيديو غير المضغوط بالزمن الحقيقي. وبدون تطبيق تقنية الضغط، لا يمكن للقرص الليزري أن يخزن إلا بضع ثوانٍ من فيديو ذو جودة عالية. لذلك يُعد ضغط الفيديو أمراً أساسياً في تقانة الفيديو الرقمي.

يعتمد مبدأ ضغط الفيديو على إزالة تكرار المعلومات سواءً في الحيز البيني أو الداخلي أو الترددي. توجد هذه التكرارات نتيجة وجود مناطق متجانسة في الصورة الواحدة أو وجود خلفية ثابتة مع حركة جسم، إضافةً إلى أن العين البشرية تتحسس أكثر للترددات المنخفضة. جرى تعريف عدة معايير لضغط الفيديو من أجل توحيد العمل والسماح لتطبيقات مختلفة من شركات مختلفة بالتوافق فيما بينها، ومن أهم هذه المعايير: سلسلة (MPEG) Motion Picture Experts Group و H.26x.

أسهمت تقنيات ضغط الفيديو في نقل الفيديو بالزمن الحقيقي عبر شبكات الاتصالات وقنواتها. وأصبحت حماية هذه المعطيات حاجةً ضرورية عند نقلها في بيئات غير آمنة. ومن هنا جاءت أهمية استعمال تقنيات التعمية (التشفير) بغرض حماية المعطيات. تتصف خوارزميات ضغط الفيديو وتعميته بزمن معالجة طويل نسبياً في حال تحقيقها برمجياً، مما يؤدي لمحدودية استخدامها في تطبيقات الزمن الحقيقي. لذا كان من الضروري تقليل زمن المعالجة بواسطة تنفيذ هذه الخوارزميات باستخدام الكيان الصلب. وتكمن صعوبة ذلك في ظل القيود التي يفرضها الكيان الصلب ومنها:

- محدودية سرعة النقل وضرورة العمل بالزمن الحقيقي.
- محدودية الذاكرة وعدد العناصر المنطقية المتاحة.
- محدودية دقة التمثيل وصعوبة تنفيذ العمليات الحسابية المعقدة.
- تسمح تقنية FPGA بتخفيف هذه القيود لما تتمتع به من إمكانيات، ومنها:
 - وجود عدد كبير من العناصر المنطقية المتاحة ضمن شريحة FPGA.
 - مرونة التصميم.
 - إمكانية توصيف أي نظام رقمي بواسطة لغات عالية المستوى.
 - إمكانية إدخال وتطبيق المعالجة الفرعية على كثير من الخوارزميات.
- تتألف هذه الأطروحة من عدة فصول.

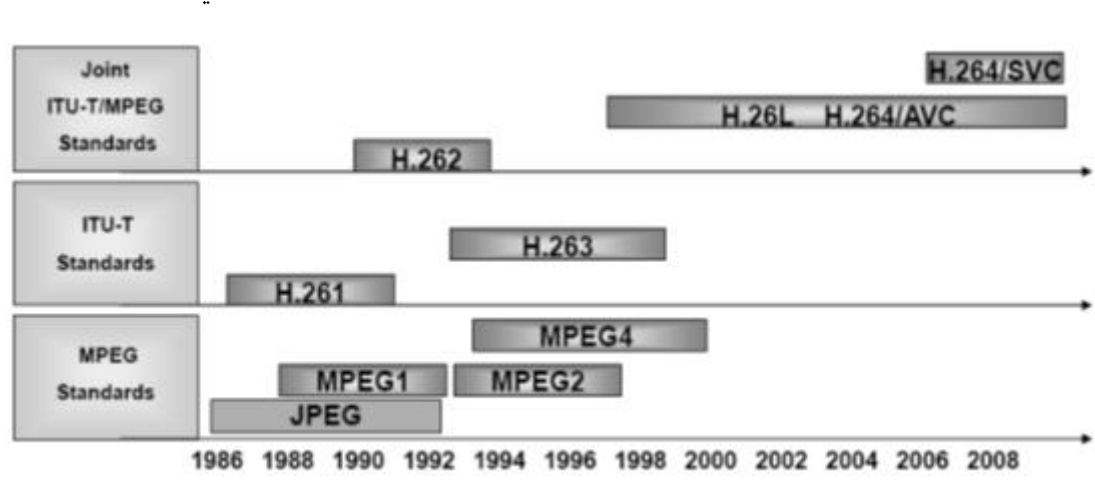
يقدم الفصل الأول مبادئ وأساسيات ضغط الفيديو إضافةً إلى المخطط الصندوقي العام لضغط الفيديو مع شرح مكوناته الأساسية، كما يقدم هذا الفصل لمحة عن المعايير الأساسية لضغط الفيديو. ويعرض الفصل الثاني دراسة للخوارزمية المعيارية H.264، ومواصفات هذا المعيار وميزاته إضافةً إلى العناصر المكوّنة له. أما الفصل الثالث فقد خُصص للدراسة المرجعية عن تصاميم الكيان الصلب لمفكك الضغط H.264، ومن ثمّ تصميم وحدات مفكك الضغط H.264. ويقدم الفصل الرابع دراسة خوارزمية التعمية المتقدمة AES، والتتابع المكونة لعملية التعمية، والخواص الرياضية المستخدمة في تصميمها. ويستعرض الفصل الخامس والأخير البطاقة DE2_115 والكاميرا المستخدمتين في هذه الأطروحة، ثم يقدم منهجية التصميم باستخدام تقانة FPGA، إضافةً إلى التنفيذ العملي للمُعْمي ومفكك تعمية ومفكك ضغط فيديو من أجل عدة تطبيقات ونتائج هذه التطبيقات.

الفصل الأول

أساسيات في ضغط الفيديو

The Basics Of Video Compression

يعتمد مبدأ ضغط الفيديو على إزالة تكرار المعلومات سواءً في الحيز البيني أو الداخلي أو الترددي. من أجل الوصول لهذه الغاية، يمكن الاستفادة من وجود خلفية ثابتة مع حركة جسماً ووجود مناطق متجانسة في الصورة، أو تحسس العين البشرية للترددات المنخفضة أكثر من الترددات العالية. ثمة معايير لضغط الفيديو من أهمها: سلسلة MPEG و H.26x، كما هو مبين في الشكل (1.1).



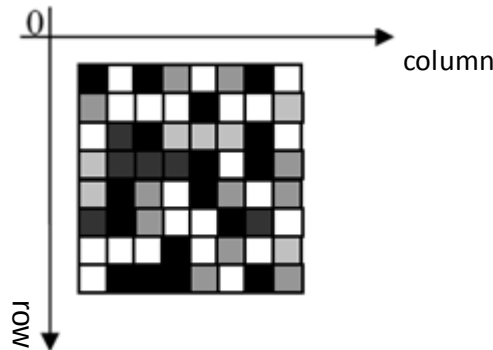
وحقق المعيار H.263 نقلة نوعية في العام 1993 لنجاحه بنقل الفيديو عبر الهواتف النقالة وفي المؤتمرات من بعد وبجودة تفوق ما سبق من معايير. إذ تصل مثلاً إلى ضعف ما هي عليه في المعيار MPEG2/H.262 عند سرعات النقل المنخفضة [13].

في العام 1995، قدم المعيار MPEG-4 باقة واسعة من الخيارات تسمح بترجيح جودة الضغط على سرعة النقل أو بالعكس.

ونجح المعيار H.264 في العام 2003 بضغط الفيديو بنسبة تعادل ضعف ما هو عليه في المعايير السابقة مع المحافظة على الجودة نفسها [14]. فتصنّف بذلك القائمة وجرى اعتماده في كثير من تطبيقات وأجهزة الفيديو مثل: iPod و Playstation وفي البث التلفزيوني DVB-H والأقراص عالية الجودة HD-DVD.

١-١- الفيديو الرقمي

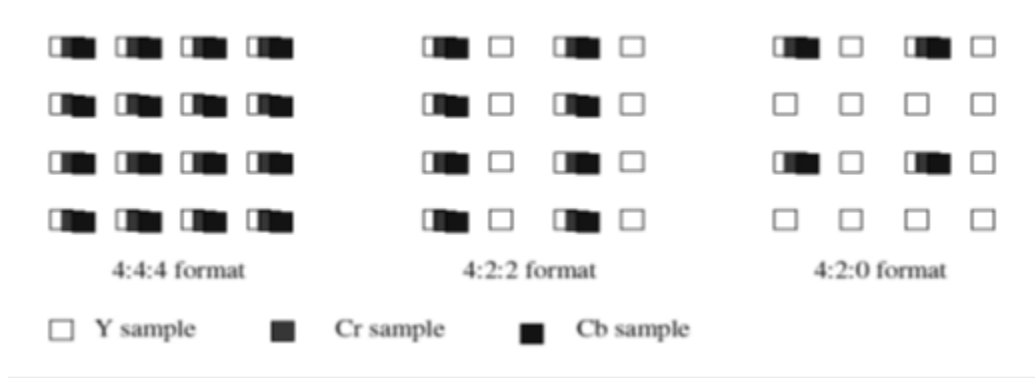
يتألف مقطع الفيديو من مجموعة من الصور يجري عرضها على التتالي بمعدل يناسب العين يطلق عليه اسم معدل الإطار frame rate. ويمثل البيكسل pixel أصغر نقطة في الصورة الرقمية. يُحدّد موقع كل بيكسل وفقاً لثنائية (سطر row، عمود column) كما هو موضح في الشكل (2.1). يعبر كل بيكسل عن لون معين ويجري تمثيله باستخدام عدد محدد من البتات.



الشكل (2.1): مثال على الصورة الرقمية [1]

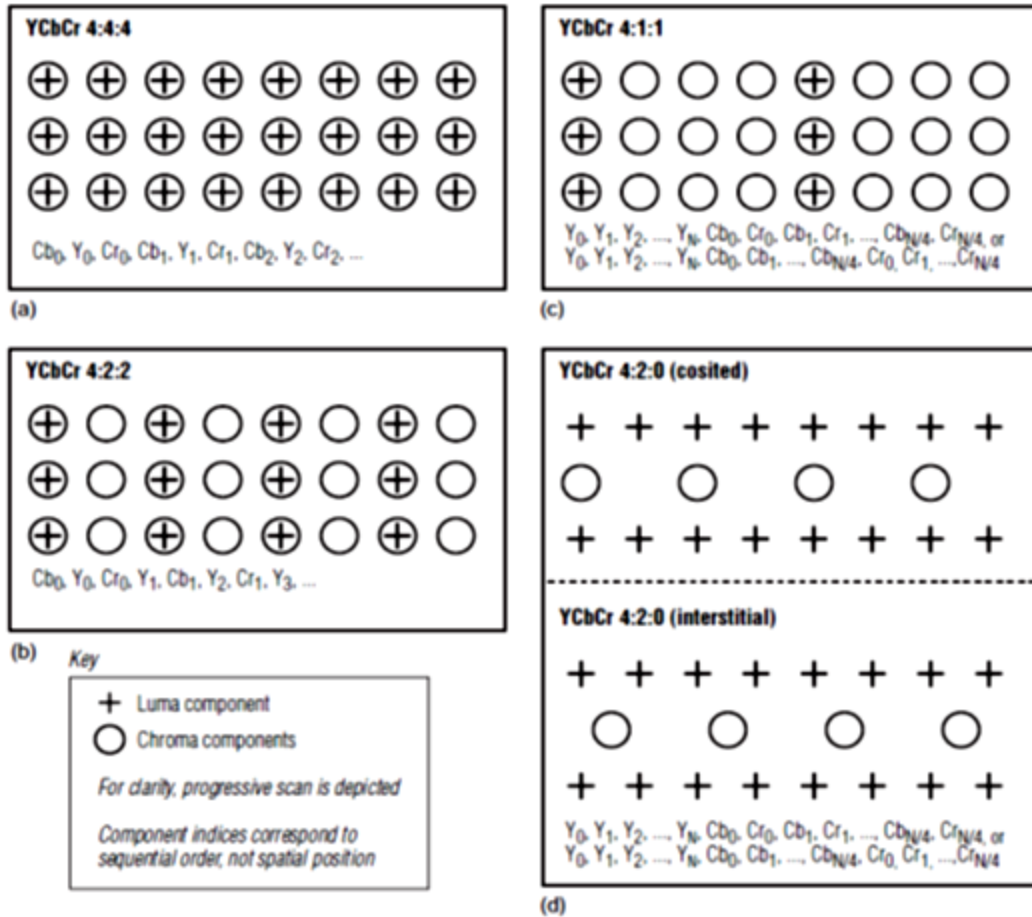
١-١-١- الميز وأخذ العينات Resolution & Sampling

يُعبّر ميز الصورة الأفقية عن عدد البيكسلات الموجودة في السطر الواحد من الصورة، بينما تعبر الميز العمودية عن عدد الأسطر في الصورة. يجب أن يتوافق تردد أخذ العينات (البيكسلات) مع معدل الإطار والذي يتراوح بين 25 إلى 30 إطار في الثانية. بينما تختلف الميز المستخدمة بحسب تطبيق الفيديو. يظهر الجدول (1.1) بعض أنواع الدقة المستخدمة على نطاق واسع اعتماداً على معيار (CIF) [1]. وُضِعَ هذا المعيار من أجل توحيد الدقات المستخدمة في تطبيقات بث الفيديو والتي تسمح بتجزئة الصورة إلى عدد صحيح من الكتل الماكروية المربعة 16×16 بُغية تسهيل



الشكل (3.1): نمط أخذ العينات [2]

تؤخذ في النمط 4:4:4 المركبات Y/Cr/Cb لكل بيكسل. وفي النمط 4:2:2، يجري أخذ نصف المركبات Cr/Cb بالاتجاه الأفقي. أما في النمط 4:2:0، فيجري أخذ نصف المركبات Cr/Cb بالاتجاهين الأفقي والعمودي. يسمح النمط الأخير بتخزين عدد أقل من العينات. يظهر في الشكل (4.1) ترتيب قراءة العينات بحسب نمط أخذ العينات [3]. فمثلاً يجري في النمط 4:2:0 تخزين كل المركبات Y ومن ثم المركبات Cr/Cb.



الشكل (4.1): ترتيب قراءة العينات بحسب نمط أخذ العينات [3]

١-٢- المخطط الصندوقي العام لضغط الفيديو

تحتاج عملية إرسال الفيديو إلى قنوات اتصال عالية السرعة، بينما تحتاج عملية تخزين الفيديو إلى ذواكر أحجامها كبيرة. إذ يتطلب إرسال فيديو بميز 720×480 ، وبتمثيل 24-bit لكل بيكسل وبمعدل 30 صورة في الثانية إلى معدل إرسال بقيمة 248 Mb/s وهو معدل يصعب تحقيقه، حيث تعاني هذه العمليات من محدودية قنوات الاتصال والذواكر. ومن هنا جاءت أهمية ضغط (ترميز) الفيديو قبل إرساله أو تخزينه.

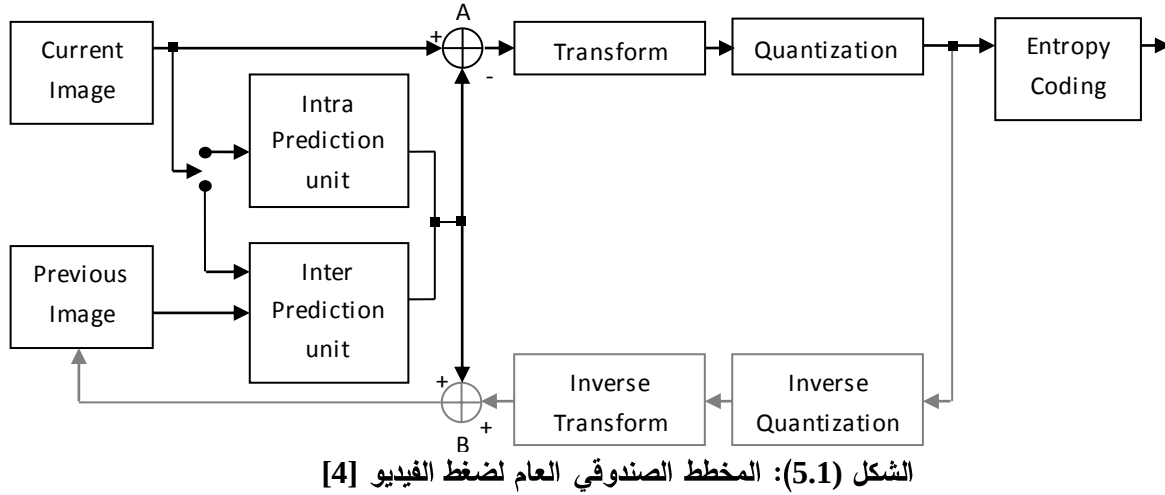
يقوم ضغط الفيديو على مبدأ أساسي وهو الاستفادة من تكرار المعلومات في الفيديو. ويمكن أن نجد هذا التكرار على ثلاثة مستويات [4]:

- مستوى داخلي ناتج عن ترابط بين بيكسلات الصورة كوجود ألوان موحدة في الصورة.
- مستوى بيني ناتج عن ترابط بين الصور المتتالية كحركة جسم واحد ضمن خلفية معينة.
- مستوى بصري ناتج بوساطة تلقي العين البشرية للمعلومات البصرية كتحسسها لشدة الإضاءة أكثر من اختلاف الألوان.

على مستوى التنفيذ العملي، نلاحظ وجود تشابه بين طرائق الضغط والمعايير المستخدمة. حيث يعتمد ضغط الفيديو عموماً على تقسيم الصورة إلى كتل ماكروية مربعة (16×16 بيكسل مثلاً). ويقسم المخطط الصندوقي العام إلى مسار مباشر وآخر عكسي [4]، كما هو مبين في الشكل (5.1). يمثل المسار المباشر (بالأسود) عملية ضغط الكتل الماكروية، بينما يمثل المسار العكسي (بالرمادي) عملية فك ضغط (إعادة بناء) الكتل الماكروية التي جرى ضغطها ضمن المسار المباشر.

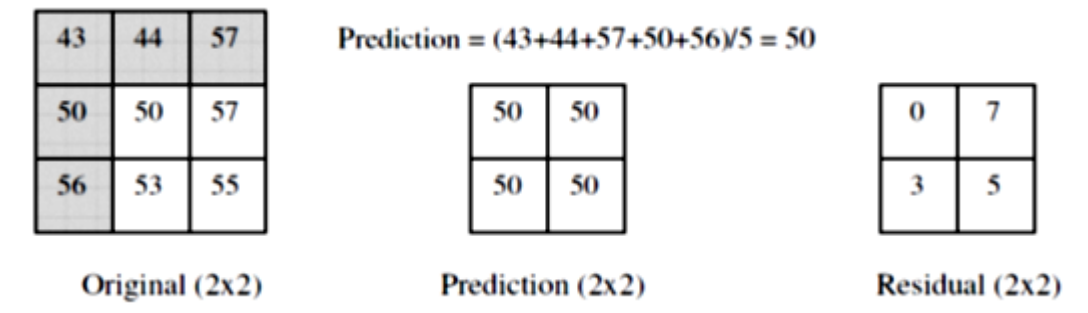
يتألف المخطط الصندوقي (الشكل 5.1) لضغط الفيديو من الوحدات:

- وحدة التنبؤ الداخلي Intra Prediction Unit.
- وحدة التنبؤ البيني Inter Prediction Unit.
- وحدة التحويل Transform Unit.
- وحدة التكمية Quantization Unit.
- وحدة الترميز الانتروبي Entropy Coding Unit.



١-٣-١ وحدة التنبؤ الداخلي Intra Prediction Unit

تعتمد عملية ضغط الفيديو على التنبؤ كأداة لإلغاء المعلومات المتكررة [5]، بحيث يجري طرح الكتلة الماكروية (Macro Block (MB) الناتجة عن عملية التنبؤ من الكتلة الماكروية الحالية (العقدة A في الشكل 5.1). تشكل هذه النتيجة الكتلة المتبقية residual التي يجري ضغطها وإرسالها بدلاً من الكتلة الماكروية الحالية لغاية ضغط النتيجة. يستفيد مخطط المسار العكسي أيضاً من نتيجة التنبؤ وذلك بإضافتها إلى الكتلة الماكروية الناتجة عنه (العقدة B في الشكل 5.1). يجري التنبؤ الداخلي intra prediction باستخدام العينات المجاورة (التي جرى ضغطها وفك ضغطها) من نفس الصورة. يكون التنبؤ الداخلي مفيداً عند ضغط المناطق المتقاربة من الصورة. يُظهر الشكل (6.1) مثلاً على عملية التنبؤ الداخلي لكتلة 2×2 باستخدام متوسط العينات المجاورة (باللون الرمادي).



الشكل (6.1): مثال على عملية التنبؤ الداخلي [4]

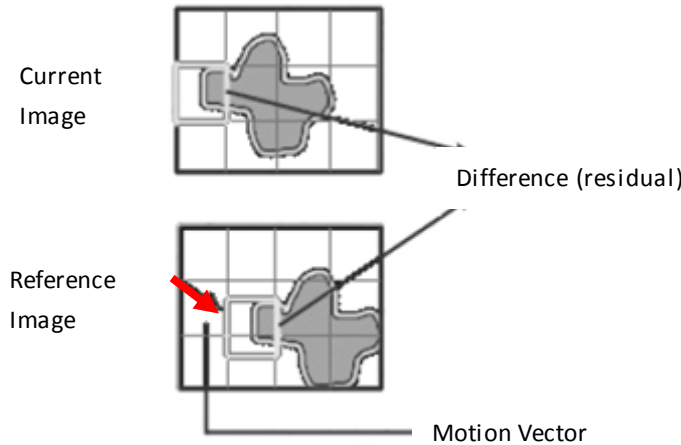
نلاحظ من المثال في الشكل (6.1) فائدة استخدام التنبؤ الداخلي. فبدون تطبيق عملية التنبؤ، نحتاج إلى تمثيل كل عينة من الكتلة 2×2 باستخدام 6-bit. أما عند تطبيق عملية التنبؤ مع الاستفادة من التقارب بين العينات فنحتاج فقط إلى 3-bit لتمثيل كل عينة من كتلة المتبقي residual.

١-٤-١ وحدة التنبؤ البيئي Inter Prediction Unit

إضافةً إلى التقارب المحتمل بين عينات الصورة نفسها، فعادةً ما يكون التقارب موجوداً بين الصور المتتالية للفيديو ويعود سبب الاختلاف إلى حركة الأجسام ضمن مشهد الفيديو. تهدف عملية التنبؤ البيئي إلى إلغاء التكرار الموجود بين صور الفيديو المتتالية. فمثلاً إذا كانت الكتلة الماكروية الحالية جزءاً من جسم متحرك فإننا سوف نجد ترابطاً قوياً بينها وبين كتلة ماكروية أخرى من المشهد السابق. يمكن إيجاد الترابط بواسطة البحث عن التشابه في منطقة ما من الصورة. تعتمد هذه العملية على مفاهيم أساسية ومنها [6]:

- الصورة المرجعية reference image: هي الصورة الناتجة عن إعادة بناء الصورة السابقة، وتستخدم للتنبؤ بالكتل الماكروية من الصورة الحالية.
- التنبؤ بالحركة motion estimation: تهدف هذه العملية إلى البحث عن أقوى ترابط بين الكتل الماكروية من الصورة المرجعية والصورة الحالية.
- تعويض الحركة motion compensation: تهدف هذه العملية إلى حساب المتبقي residual بين نتيجة التنبؤ والكتلة المرجعية.
- شعاع الحركة motion vector: و يمثل شعاع الحركة بين الكتلة المرجعية والكتلة المترابطة معها.

بعد حساب كتلة المتبقي residual، يجري ضغط هذه الكتلة وإرسالها مع شعاع الحركة المناسب إلى مفكك الضغط. يستخدم هذا الشعاع لإيجاد الموقع الصحيح للكتلة المتنبأ بها، ويضاف الفرق للحصول على الكتلة الأصلية الشكل (7.1).

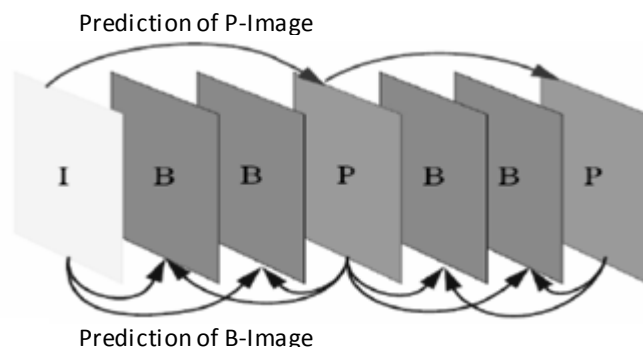


الشكل (7.1): مثال على التنبؤ البيئي [10]

يمكن تصنيف الصور إلى ثلاثة أنواع، كما هو مبين في الشكل (8.1) وهي:

- I-Pictures: يجري ضغط كل الكتل الماكروية في هذه الصورة وفق التنبؤ الداخلي. تستخدم هذه الصورة في بداية كل تتالي من الصور.

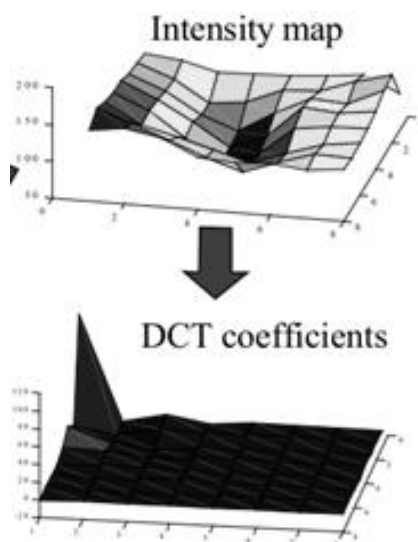
- P-Pictures: يجري ضغط كل الكتل الماكروية في هذه الصورة وفق التنبؤ البيني. وتستخدم الصورة السابقة كصورة مرجعية لهذه الصورة.
- B-Pictures: يجري ضغط كل الكتل الماكروية في هذه الصورة وفق التنبؤ البيني وبالاتجاهين. وتحتاج هذه الصورة إلى صورة مرجعية سابقة وأخرى لاحقة.



الشكل (8.1): ترتيب الصور في مقطع الفيديو [4]

١-٥-١ وحدة التحويل Transform Unit

تحول هذه الوحدة كتل المتبقي residuals من فضاء الصورة إلى فضاء التردد. هذا التحويل لا يضغط الصورة، لكنه يهدف إلى تركيز الطاقة في عدد قليل من المعاملات coefficients. التحويل الأكثر استخداماً في ضغط الصورة والفيديو هو تحويل التجيب المتقطع (DCT). يؤدي تطبيق هذا التحويل على صورة ثنائية البعد، كما هو مبين في الشكل (9.1)، إلى تجمع المعاملات ذات الأوزان المهمة في زاوية واحدة بينما تقترب أوزان بقية المعاملات من الصفر.



الشكل (9.1): تحويل التجيب المتقطع [4]

من أجل صورة قياسها 8×8 بيكسل، يعطى التحويل DCT بالعلاقة (4.1) [7]:

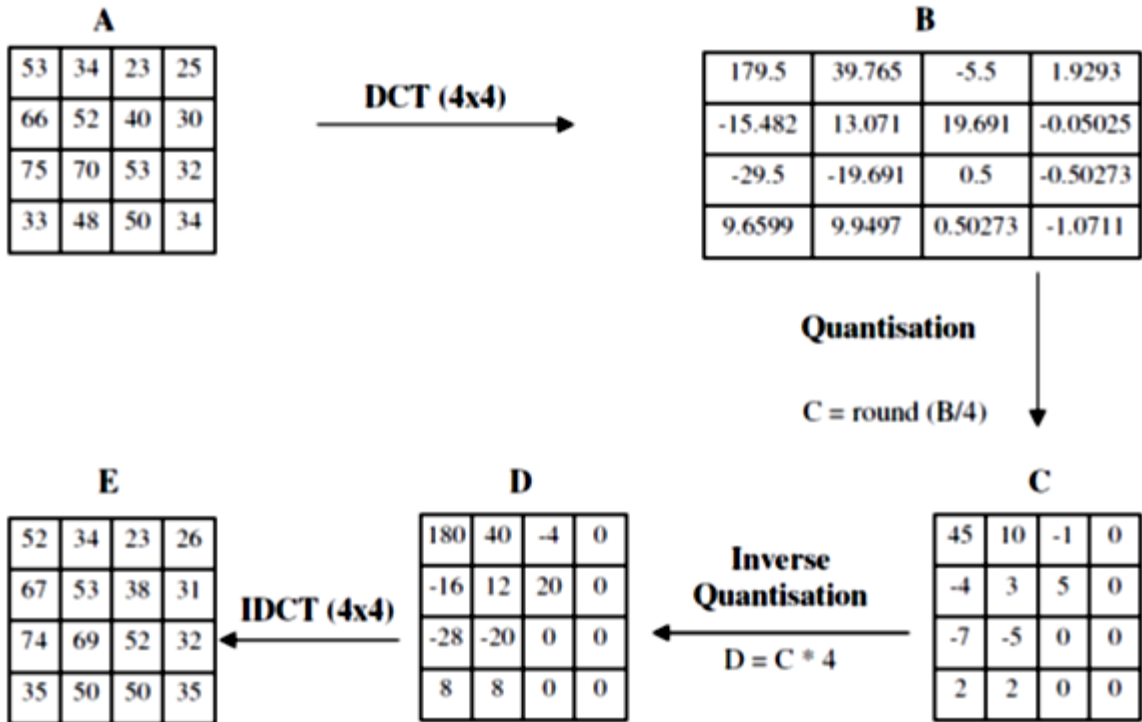
$$F(u, v) = \frac{C_u}{2} \frac{C_v}{2} \sum_{y=0}^7 \sum_{x=0}^7 f(x, y) \cos \left[\frac{(2x+1)u\pi}{16} \right] \cos \left[\frac{(2y+1)v\pi}{16} \right] \quad (4.1)$$

$$C_u = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } u = 0, \\ 1 & \text{if } u > 0 \end{cases}; C_v = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } v = 0, \\ 1 & \text{if } v > 0 \end{cases} \quad (5.1)$$

يمثل العدد $f(x, y)$ إحدى مركبات اللون عند البيكسل الموافق للموقع $[x, y]$. ويمثل الموقع $[0, 0]$ الزاوية اليسارية العليا.

٦-١-١ وحدة التكمية Quantization Unit

تهدف عملية التكمية إلى تمثيل الكميات المستمرة باستخدام عدد محدد من البتات. تقوم هذه العملية على تقسيم معاملات التحويل على خطوة التكمية ثم التقريب إلى أقرب عدد صحيح [4]. تطبق العملية العكسية لحساب المسار العكسي. تؤدي عملية التكمية إلى ضغط المعطيات لكن مع حدوث بعض الضياعات. يُبين الشكل (10.1) العمليات المباشرة والعكسية المطبقة على كتلة 4×4 بيكسل.

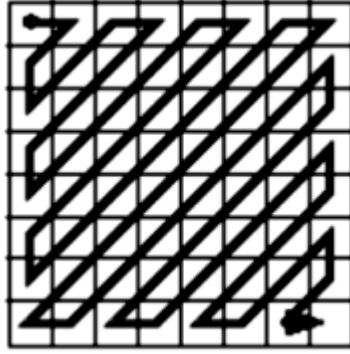


الشكل (10.1): عملية التكمية والتكمية العكسية [4]

نلاحظ بعد تطبيق التحويل DCT، أن معظم الطاقة في الكتلة B تتركز باتجاه الأعلى وإلى اليسار. تنتج الكتلة C بعد التكمية، أي بعد التقسيم على عدد طبيعي وإجراء عملية التقريب. نقوم في هذا

المثال بالتقسيم على العدد 4 ومن ثم بالتقريب إلى أقرب عدد صحيح. يجري إرسال الكتلة C إلى مفكك الضغط.

نلاحظ أن كمية المعطيات المرسلة في الكتلة C أقل مقارنةً مع الكتلة الأولية A. بعد استقبال الكتلة C نقوم بجراء العملية العكسية لعملية التكمية وفي هذا المثال نقوم بضرب الكتلة بالعدد 4 فنحصل على الكتلة D. بسبب الضياعات الناتجة عن التكمية، نلاحظ أن $D \neq B$. ومع ذلك فإن الاختلاف بسيط بين الكتلة الأولية A والكتلة النهائية E التي نحصل عليها بإجراء التحويل العكسي IDCT. يجري إرسال واستقبال الكتلة المربعة C وفق ترتيب تعرجي zigzag بدءاً من الزاوية العليا إلى اليسار وانتهاءً بالزاوية المقابلة وذلك لإرسال المعاملات وفق أوزانها الشكل (11.1).



الشكل (11.1): إعادة ترتيب عينات الكتلة 8×8 بشكل تعرجي zig-zag [4]

١-٧- وحدة الترميز الانتروبي Entropy Coding Unit

تقوم وحدة الترميز الانتروبي بضغط المعلومات قبل إرسالها بوساطة معرفة تكرارها الإحصائي. حيث يجري ترميز المعطيات الأكثر تكراراً بأقل عدد من البتات والأقل تكراراً بأكبر عدد من البتات. وتضم المعلومات المرسلة إضافةً إلى كتل المتبقي المرمزة encoded residuals، أشعة الحركة، ومعاملات التكمية ومعلومات أخرى. ومن أنواع الترميز الانتروبي الشائعة: ترميز هوفمان Huffman والترميز الحسابي Arithmetic Coding [8].

١-٣- معايير ضغط الفيديو

تساعد عملية وضع معايير لضغط الفيديو على توحيد المواصفات بين المصنّعين [9]. ثمة هيئتان تعنيان بقضايا التقييس :

- الاتحاد الدولي للاتصالات - فرع تقييس الاتصالات
Telecommunication Standardization Sector (ITU-T)
- منظمة التقييس الدولية - هيئة التقنيات الكهربائية
International Electrotechnical Commission (ISO/IEC)

المعايير التي جرى إصدارها من قبل ITU-T سميت بسلسلة H26x، وأصدرت الهيئة ISO/IEC سلسلة معايير (MPEG) Motion Picture Experts Group. توجّهت سلسلة MPEG نحو تخزين معطيات الوسائط المتعددة وتوزيعها، في حين توجّهت سلسلة H26x نحو تطبيقات اتصالات الفيديو في الزمن الحقيقي. ومن أهم المعايير الشائعة:

المعيار MPEG1 :

جرى إصدار مسودة هذا المعيار في العام 1993 [54] وعلى الرغم من أنه معيار عام أي غير موجه نحو تطبيق معين لكنه مصمم أساساً من أجل التخزين على أجهزة الوسائط المتعددة الرقمية مثل الأقراص المدمجة CD-ROM. يسمح هذا المعيار بمعدلات نقل أكثر من 1.5 Mbps. يطبق هذا المعيار خوارزمية ضغط كتلي (تحويل وتنبيؤ) تدعم ثلاث أنواع من الصور هي I,P,B وذلك لتأمين فعالية ضغط جيدة.

المعيار MPEG2 :

صدر هذا المعيار في العام 1995 [55] واستخدم من أجل طيف واسع من التطبيقات مثل التخزين على أجهزة الوسائط المتعددة والبث التلفزيوني الفضائي. جرى تطوير هذا المعيار باستخدام خوارزمية MPEG1 مع إضافة أدوات جديدة من أجل الحصول على فعالية وجودة أفضل مثل ضغط الفيديو المتداخل من أجل بعض التطبيقات مثل التلفزيون الرقمي عالي التعريف (HDTV) High Definition Television.

المعيار MPEG4:

صدر هذا المعيار في العام 1998 [12] وهو يدعم تطبيقات واسعة ومختلفة بما فيها عرض الفيديو عبر الإنترنت والبث التلفزيوني الرقمي، إضافةً إلى تطبيقات دمج مشاهد الفيديو الحقيقية مع الرسوم المولدة بالحاسوب. يمكن لهذا المعيار دعم معدلات نقل أقل من معدلات MPEG2/MPEG1، وهو يدعم ضغط فيديو غرضي بحيث يجري تقسيم مشهد الفيديو إلى عدة أغراض يجري ضغط كل منها باستقلالية عن الأغراض الأخرى.

المعيار H261:

صدر في العام 1993 [1] واستخدم على نطاق واسع من أجل تطبيقات الهاتف الفيديو والمؤتمرات الفيديوية عبر قنوات الشبكة الرقمية للخدمات المتكاملة ISDN الفرعية والأساسية وهو يدعم معدلات نقل تصل حتى 1920 kbps. يستعمل هذا المعيار خوارزمية تشفير هجينة مشابهة

لتلك المستخدمة في MPEG1 من أجل ترميز فعال بمعدلات نقل منخفضة معتمداً على خوارزمية حسابية بسيطة.

المعيار H263:

استخدم هذا المعيار [13] من أجل اتصالات الفيديو بمعدلات نقل منخفضة، حيث تعتمد خوارزمية هذا المعيار على المعيار السابق H261. يدعم هذا المعيار أنماط فيديو واسعة وأدوات ضغط متطورة مثل تعويض الحركة بميز نصف بكسل وأدوات ضغط مرنة كما في التنبؤ حيث يجري تقسيم الكتلة الماكروية 16×16 ، وهي الوحدة الأساسية في الضغط، إلى 4 كتل ويجري تعويض الحركة في كل واحدة منها باستخدام أشعة حركة مستقلة عن الأخرى مما يقود إلى درجة أعلى من الدقة والمرونة لعملية الضغط.

المعيار H264\MPEG4 :

جرى تطوير هذا المعيار [5] بالتعاون بين MPEG وITU-T وهو قادر على الوصول إلى كفاءة ومرونة ضغط محسنة جداً مقارنة مع معايير ضغط الفيديو السابقة. ينتج تحسين الأداء عن أدوات وخيارات عديدة متاحة في هذا المعيار تزيد التعقيد الحسابي لدرجة عالية. قمنا في هذا البحث بالاعتماد المعيار الأخير بسبب جودة أدائه، ويعرض الفصل الثاني مواصفات هذا المعيار وميزاته إضافةً إلى العناصر المكونة له.

الفصل الثاني

H.264 المعيار

H.264 Standard

٢-١- نظام الضغط الفيديوي في المعيار H.264

تعدّ عملية ضغط الفيديو من التقنيات الأساسية المستخدمة في تطبيقات الفيديو ومنها [14][15]:

- بث الفيديو الرقمي (Digital Video Broadcast (DVB).
 - عقد المؤتمرات المرئية video conferencing.
 - نقل الفيديو عبر الانترنت internet video streaming.
 - عرض التلفزيون عبر الهاتف النقال mobile TV.
 - تخزين الفيديو باستخدام أقراص Digital Video Disks (DVD).
- يعدّ المعيار H.264 من أهم المعايير العالمية لضغط الفيديو (أو ترميز الفيديو) [5]. نُشرَ هذا المعيار في العام 2003 وجرى تطويره اعتماداً على معايير أخرى كالمعيار MPEG-2 والمعيار MPEG-4 لتحسين عملية الضغط وزيادة المرونة في نقل الفيديو وتخزينه. يقسم نظام ضغط الفيديو حسب هذا المعيار إلى طبقتين [6]:

- طبقة ضغط الفيديو (Video Coding Layer (VCL: تجري في هذه الطبقة عملية ضغط الفيديو فعلياً.
- طبقة موائمة الشبكة (Network Adaption Layer (NAL: تجري في هذه الطبقة عملية نقل معطيات الفيديو الموجودة في الطبقة VCL إضافةً إلى معلومات أخرى كمعاملات النظام.
- يسمح فصل عمليتي ضغط ونقل الفيديو بتحقيق التوافق المطلوب مع أنظمة ضغط مختلفة.

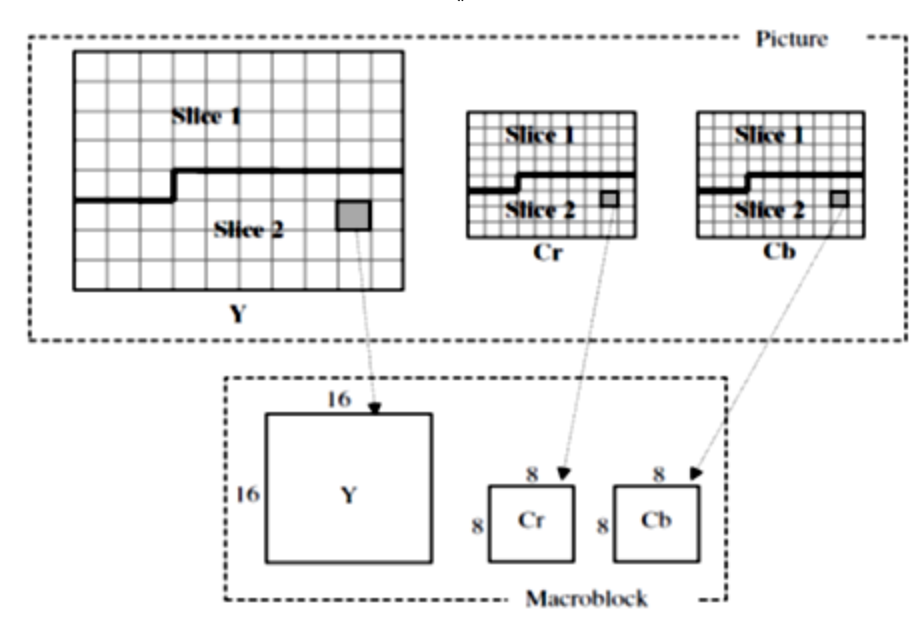
٢-١-١- الأشكال الجانبية Profiles

يدعم المعيار H.264 تطبيقات تختلف في تعقيدها ومعدلات الضغط وسرعة عملها. لذا يضم هذا المعيار عدة أشكال جانبية profiles تتدرج بحسب أدائها من ناحيتي السرعة ومعدلات الضغط وهي:

- الشكل المُبسّط Baseline: يدعم هذا الشكل التطبيقات التي تعمل بمعدلات سرعة وضغط منخفضة كتطبيق عقد المؤتمرات المرئية.
- الشكل الأساسي Main: يدعم هذا الشكل التطبيقات التي تتطلب معدلات ضغط عالية كتطبيقات تخزين وبث الفيديو.
- الشكل المُوسّع Extended: هو الشكل المُحسّن من الشكل المُبسّط baseline ويستخدم في تطبيقات نقل الفيديو بواسطة الإنترنت.

٢-١-٢- بنية الصورة Picture Structure

تتألف كل صورة من شريحة واحدة slice أو أكثر، كما هو موضح في الشكل (1.2). وتتألف كل شريحة من عدد صحيح من الكتل الماكروية MacroBlock (MB). تمثل الكتلة الماكروية أصغر وحدة ضغط في الصورة وتحتوي على 16×16 مركبة Y إضافة إلى ما يرافقها من المركبات Cr/Cb. يجري هنا أخذ العينات وفق التشكيلة (4:2:0) وهي من التشكيلات الأكثر انتشاراً.

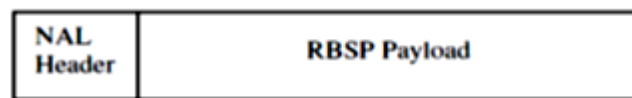


الشكل (1.2): بنية الصورة [4]

٢-١-٣- طبقة موائمة الشبكة NAL

يجري تضمين معطيات الفيديو المرمزة إضافة إلى معلومات أخرى كمعاملات النظام في طبقة نقل الفيديو (VCL) على شكل وحدات NAL Units. وتتألف كل وحدة من قسمين (الشكل 2.2) هما:

- الحمل المفيد لمتتالية البايت الخام Raw Byte Sequence Payload (RBSP): يحتوي هذا القسم على معطيات الفيديو المرمزة أو موسطات (بارامترات) النظام.



الشكل (2.2): بنية طبقة نقل الفيديو [4]

- الترويسة NAL header: يسمح هذا القسم بتحديد نوع الوحدة NAL Unit، ويوجد نوعان من الوحدات:
 - VCL-NAL: تحتوي هذه الوحدة على معطيات الفيديو المرمزة (VCL).
 - Non-VCL-NAL: تحتوي هذه الوحدة على معاملات النظام وتُقسّم بدورها أيضاً إلى نوعين:

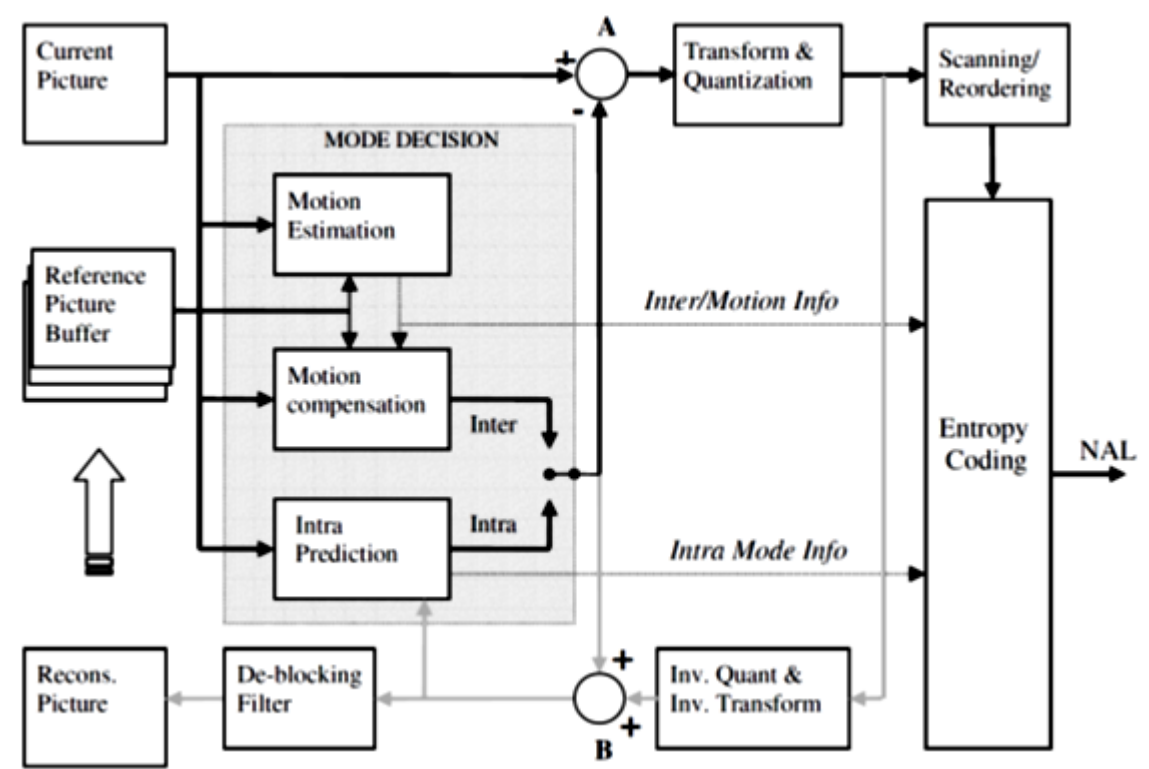
- مجموعة موسطات الفيديو (Sequence Parameter Sets (SPS): تحتوي هذه الوحدة على موسطات تخص إطارات الفيديو. من أهم هذه الموسطات: عدد الإطارات المرجعية، وحجم الإطار مقدراً بعدد الكتل الماكروية.
 - مجموعة موسطات الصورة (Picture Parameter Sets (PPS: تحتوي هذه الوحدة على معلومات تخص الصور الفردية. من أهم هذه الموسطات: معامل التكمية، ونوع الشريحة (I/P/B...)، ونمط الترميز الانتروبي.
- يوضح الشكل (3.2) مثلاً عن محتوى طبقة نقل الفيديو.

NAL Header	SPS	NAL Header	PPS	NAL Header	I-Slice	NAL Header	P-Slice
------------	-----	------------	-----	------------	---------	------------	---------

الشكل (3.2): مثال عن محتوى طبقة نقل الفيديو [4]

٢-١-٤ - وحدة الضاغط (المرمّز) H.264 Encoder

يجري في البداية تقسيم مقطع الفيديو إلى كتل ماكروية Macro Blocks بحجم 16×16 بيكسل. يتألف مخطط وحدة الضاغط (الشكل 4.2) من وحدة مسار مباشر Forward path (بالخط الأسود) مهمتها ضغط معطيات الفيديو ووحدة مسار عكسي Reverse path (بالخط الرمادي) لفك الضغط وإعادة بناء الصورة.



الشكل (4.2): المخطط الصندوقي لوحدة الضاغط H.264 [4]

تطبق عملية تنبؤ prediction على كل وحدة بإحدى طريقتين: النمط الداخلي intra mode أو النمط البيئي inter mode. يعتمد التنبؤ الداخلي على العينات المجاورة من الشريحة نفسها والتي جرى ضغطها وفك ضغطها وإعادة بنائها مسبقاً. بينما يعتمد التنبؤ inter على التنبؤ بالحركة motion estimation وتعويض الحركة motion compensation باستخدام صور مرجعية مخزنة في ذاكرة العازل buffer.

يجري طرح الكتلة الماكروية المتنبأ بها من الكتلة الماكروية الأصلية في العقدة A للحصول على كتلة المتبقي residual كما هو مبين في الشكل (4.2).

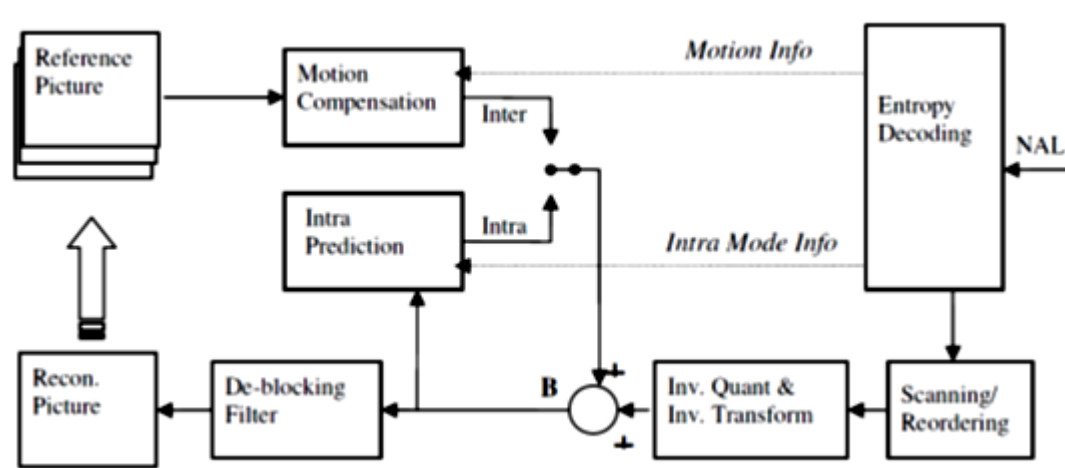
يجري تحويل هذه الكتلة وتكميتها وإعادة ترتيبها قبل ترميزها بواسطة الترميز الانتروبي entropy. يستعمل هذا الترميز لتقليص حجم المعطيات التي تشمل معطيات الفيديو إضافة إلى معطيات أخرى كتلك المتعلقة بموسطات التنبؤ بنوعيه الداخلي والبيئي intra/inter. تقوم المرحلة الأخيرة على تضمين المعطيات المرمزة في طبقة NAL.

يهدف مخطط المسار العكسي إلى إعادة بناء الصورة تلاماً كما يفعل مفكك الضغط. فبعد إعادة بناء الكتل الماكروية، تُستخدم المتجاورة منها والتي تنتمي إلى الشريحة ذاتها لإجراء عملية التنبؤ الداخلي. كما تُستخدم الصور المُعاد بناؤها لإجراء عملية التنبؤ البيئي للصور المستقبلية. يجري في العقدة B جمع الكتلة الماكروية المتنبأ بها خلال المسار المباشر مع كتلة المتبقي الناتجة عن المسار العكسي لإعادة بناء الكتلة الماكروية. ومن ثم يُطبق على الكتلة الأخيرة مرشح التنعيم deblocking filter من أجل تخفيف مشاكل الحواف.

٢-١-٥ - وحدة مفكك الضغط H.264 Decoder

يُظهر الشكل (5.2) مخطط وحدة مفكك الضغط. يجري في البداية تفكيك الترميز الانتروبي لوحدة طبقة NAL من أجل الحصول على موسطات التكمية ومعاملات التنبؤ إضافة إلى موسطات أخرى. تعقب هذه المرحلة، مرحلة إعادة ترتيب المعطيات ومرحلتي التكمية العكسية والتحويل العكسي.

تجري عملية التنبؤ البيئي بواسطة تعويض الحركة باستخدام المعطيات المفككة كالصور المرجعية وأشعة الحركة. وتجري عملية التنبؤ الداخلي أيضاً باستخدام المعطيات المفكك ضغطها كنمط التنبؤ والكتل الماكروية المتجاورة. تجري إعادة بناء الكتل بجمع نتيجة التنبؤ مع كتلة المتبقي عند العقدة B. ومن ثم يطبق مرشح التنعيم deblocking filter للحصول على الصورة النهائية.



الشكل (5.2): مخطط وحدة مفكك الضغط H.264 [4]

٢-٢ - مكونات وحدة الضغط H.264

يتألف الضاغط H.264 من عدة مكونات أساسية. وفي حال عدم ذكر نوع الكتلة الماكروية (Y أو Cr/Cb) فالمقصود ضمناً كتلة الإضاءة الماكروية Y. وما ينطبق على هذه الكتلة ينطبق أيضاً على كتلة الألوان Cr/Cb ولكن بقياس مختلف. سنعتمد في دراستنا على التشكيلة YCrCb (4:2:0). وفقاً لهذه التشكيلة فإن قياس كتلة الألوان Cr أو Cb يساوي ربع قياس كتلة الإضاءة Y. فمثلاً كتلة الإضاءة ذات القياس 16×16 تتوافق مع كتلة ألوان بقياس 8×8 .

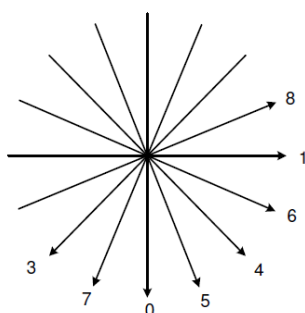
٢-٢-١ - التنبؤ الداخلي Intra Prediction

يعتمد التنبؤ الداخلي على الكتل الماكروية المجاورة والتي أعيد بناؤها مسبقاً. إذ يستفيد هذا التنبؤ من الترابط الفراغي للبيكسلات spatial correlation. يقدم التنبؤ الداخلي 9 خيارات (أنماط) للتنبؤ بكل كتلة إضاءة ماكروية بقياس 4×4 و 4 أنماط لكل كتلة إضاءة ماكروية (Y block) بقياس 16×16 و 4 أنماط أيضاً لكل كتلة ألوان ماكروية (Cr/Cb block) [16]. على الضاغط اختيار التنبؤ الذي يحقق أقل فرق بين الكتلة المختارة للتنبؤ والكتلة الأصلية. ثمة حاجة إلى 13 عينة إضاءة مجاورة من أجل التنبؤ بكتلة إضاءة ماكروية 4×4 ، كما هو موضح في الشكل (6.2) وهي A, B, C, ..., G, H, M, I, J, K, L.

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

الشكل (6.2): عيّنات الإضاءة المجاورة لكتلة إضاءة ماكروية 4×4 [16]

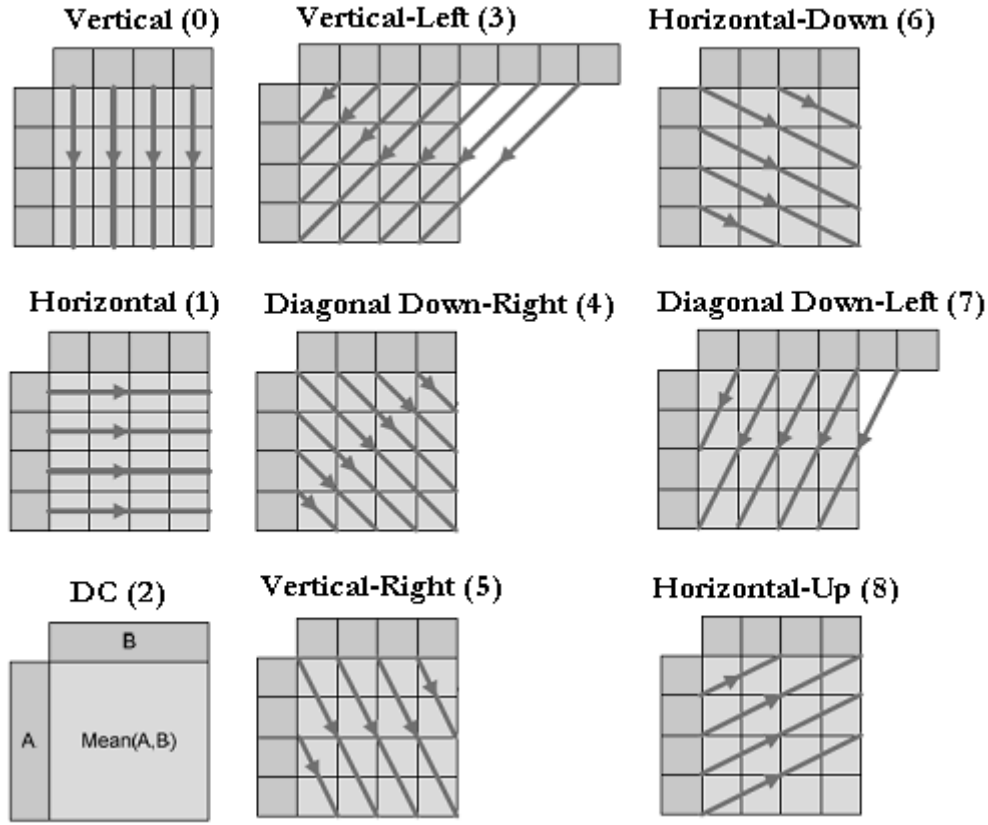
ويعتمد كل نمط من أنماط التنبؤ التسع على استخدام التوسيط mean أو التكرار أو استخدام مرشح ذو استجابة محدودة Finite Response Filter (FIR) يعتمد على ضرب المعاملات بالمركبات ($\frac{1}{4}, \frac{1}{2}, \frac{1}{4}$). يظهر الشكل (7.2) الاتجاهات التي يجري وفقها كل نمط من أنماط التنبؤ.



الشكل (7.2): اتجاهات أنماط التنبؤ الداخلي [16]

تعدّ الأنماط الثلاثة الأولى من الأنماط الأكثر استخداماً. ينتج عن عملية التنبؤ الداخلي معلوماتان: النمط الذي جرى اعتماده (بين الأنماط التسعة) إضافة إلى الكتلة المتبقية residual block. يظهر الشكل (8.2) أنماط التنبؤ التسعة من أجل كتلة إضاءة ماكروية بقياس 4×4. يجري التنبؤ العمودي لكتلة الإضاءة بتكرار عينات كل عمود بالتساوي مع العينة العليا المجاورة له. ويجري بطريقة مشابهة التنبؤ الأفقي لكتلة الإضاءة بتكرار عينات كل سطر بالتساوي مع العينة اليسرى المجاورة له.

أما التنبؤ المستمر DC فيعدّ أن كل عينات الكتلة متساوية وتساوي متوسط العينات الثمانية المجاورة في الأعلى وعلى اليسار. ويعدّ التنبؤ المائل عينات كل شعاع قطري متساوية. يجري حساب عينة كل قطر بتطبيق مرشح (FIR) مركباته ($\frac{1}{4}, \frac{1}{2}, \frac{1}{4}$) على ثلاث عينات مجاورة لمبدأ الشعاع القطري في الأعلى أو على اليسار.



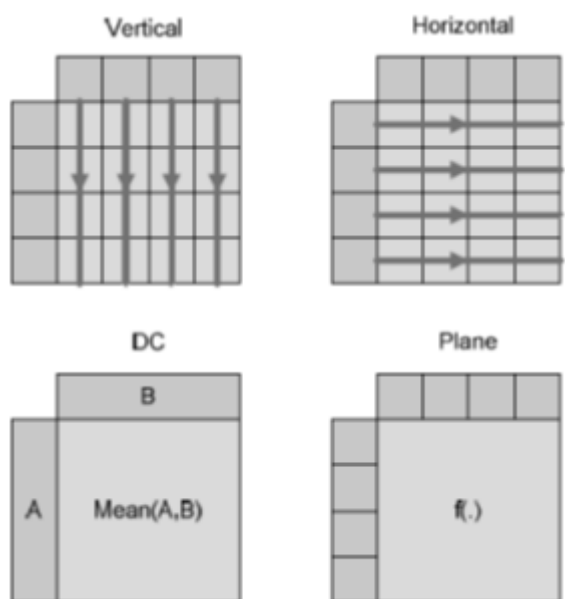
الشكل (8.2): أنماط التنبؤ التسعة من أجل كتلة إضاءة ماكروية بقياس 4×4 [16]

تقوم وحدة التنبؤ الداخلي بحساب مجموع القيم المطلقة للفروقات (SAD) Sum of Absolute Differences بين عينات الكتلة الأصلية $B(k,l)$ وعينات الكتلة $P(m,n)$ الناتجة عن أحد أنماط التنبؤ التسعة، ويقع الاختيار على النمط الذي يحقق أقل فرق. يُعطى مجموع القيم المطلقة للفروقات بالعلاقة (1.2) .

$$SAD(k,l;m,n) = \sum_{i=0}^N \sum_{j=0}^N |B(k+i,l+j) - P(m+i,n+j)| \quad (1.2)$$

كما يقدم التنبؤ الداخلي 4 أنماط لكل كتلة إضاءة ماكروية 16×16 . وهي: التنبؤ الأفقي والتنبؤ العمودي والتنبؤ المستمر إضافةً إلى التنبؤ المعتمد على حساب معادلة خطية تضم كل العينات المجاورة [16].

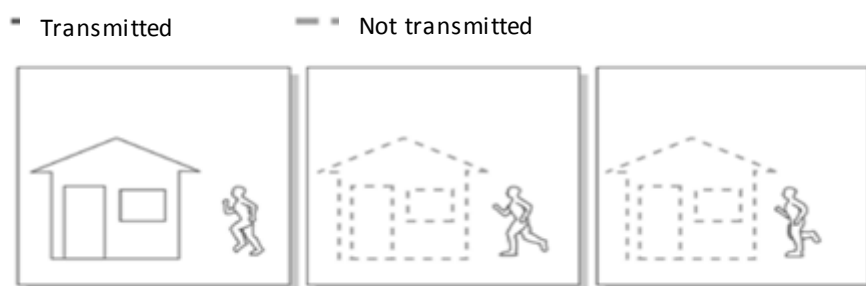
ويقدم التنبؤ الداخلي أيضاً، 4 أنماط لكل كتلة ألوان ماكروية (الشكل 9.2) بأخذ المركبات Cr/Cb أكثر تجانساً من المركبات Y. تعدّ هذه الأنماط مشابهة لأنماط التنبؤ من أجل كتلة الإضاءة الماكروية 16×16 .



الشكل (9.2): أنماط التنبؤ الأربعة من أجل كتلة ألوان ماكروية بقياس 4×4 [16]

٢-٢-٢ - التنبؤ البيني Inter Prediction

يسمح التنبؤ البيني بضغط معطيات الفيديو بواسطة ضغط الاختلافات (residuals (differences بين الصور المتتالية بدلاً من ضغط الصور نفسها، حيث تجري مقارنة كل صورة مع صورة مرجعية ويجري ضغط وإرسال العينات المختلفة فيما بينهما [6].



الشكل (10.2): مثال أول عن التنبؤ البيني [11]

يبين الشكل (10.2) عملية ضغط الصورة الأولى بكاملها (I-Picture)، بينما يجري ضغط الأجزاء المتحركة فقط (الرجل الذي يركض) في الصورتين التاليتين (P-Picture) ويمكن ضغط كمية أقل من المعلومات إذا استطعنا التنبؤ بالحركة وهذا ما يقوم به التنبؤ البيني في الضغط H.264. يقدم الشكلان (11.2 a و b)، إطارين متتاليين من مقطع فيديو ويظهر في الشكل (11.2 c) الفرق بين الإطارين. وتظهر أشعة الحركة باللون الأبيض في الشكل (11.2 d). ويعبر الشكل (11.2 e) عن تعويض الحركة تبعاً لأشعة الحركة. نلاحظ أن الشكل (11.2 c) يقتصر على حساب الفرق بين الإطارين دون الاستفادة من تكرار المعلومات، بينما تظهر الصورة في الشكل (11.2 e) مضغوطة أكثر بسبب تطبيق عملية التنبؤ بالحركة.



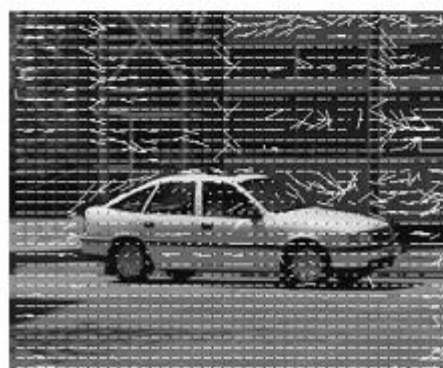
(a) Frame 1



(b) Frame 2



(c) Residual (difference) frame



(d) Motion Vector

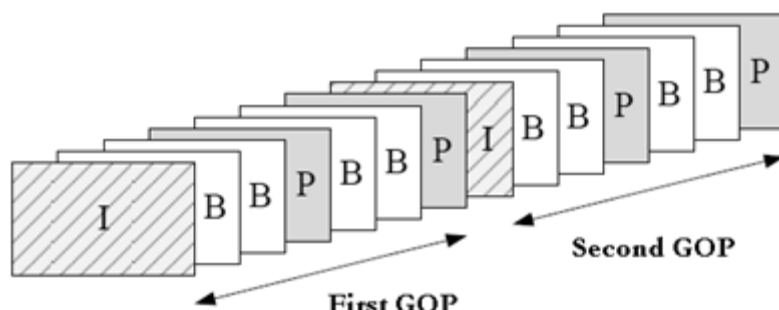


(e) Motion compensated residual frame

الشكل (11.2): مثال ثاني عن التنبؤ البيني [11]

ينتج عن التنبؤ البيني نوعان من الصور: P-Picture و B-Picture (الشكل 12.2). تُبنى الصورة من النوع P-Picture اعتماداً على صورة سابقة من نوع I-Picture أو P-Picture، بينما تُبنى الصورة من النوع B-Picture اعتماداً على صورتين: سابقة ولاحقة من نوع I-Picture أو P-Picture.

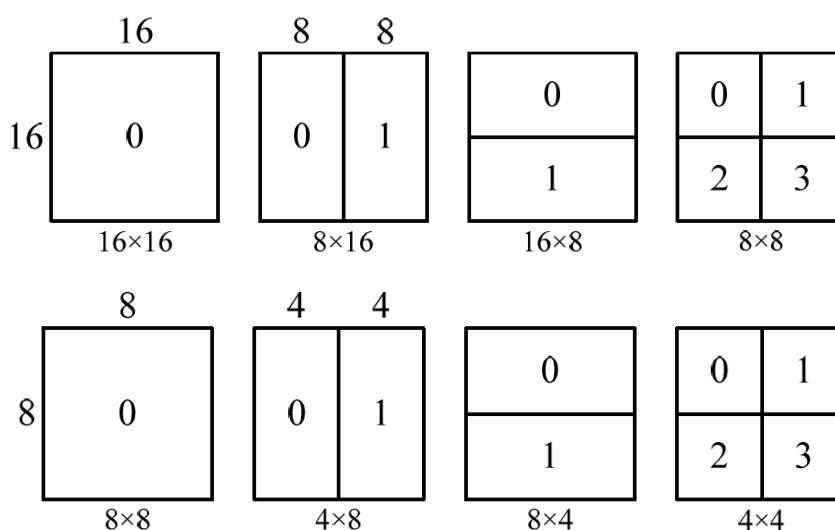
يجري ترتيب الصور بحسب أنواعها (I/P/B-Picture) ضمن مجموعات من الصور (Group Of Pictur (GOP). من أشهر هذه المجموعات: GOP=IBBPBBP (الشكل (12.2).



الشكل (12.2): ترتيب الصور ضمن مجموعات GOP [11]

٢-٢-١ - التنبؤ بالحركة وتعويضها باستخدام الكتل المتغيرة الحجم

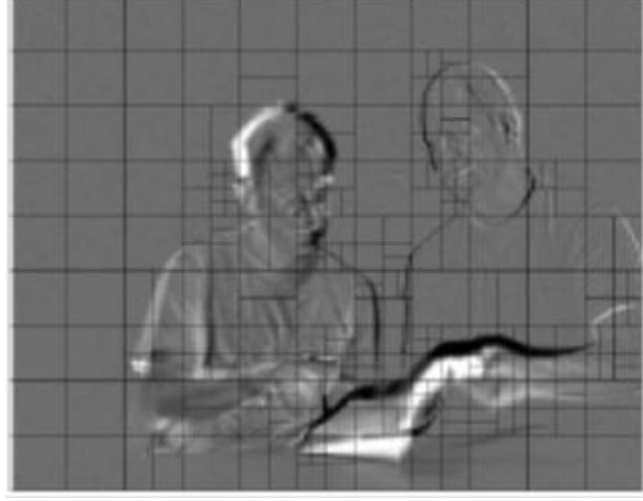
تستخدم عمليتا التنبؤ بالحركة وتعويضها من أجل تحقيق التنبؤ البيني [6][16]. حيث تُقسَّم الكتل الماكروية إلى كتل جزئية يتغير قياسها بحسب تفاصيل الحركة، كما هو موضح في الشكل (13.2).



الشكل (13.2): تقسيم الكتل الماكروية إلى كتل جزئية [16]

إذ يمكن تقسيم كتلة الإضاءة الماكروية 16×16 بعدة طرائق: 16×16 (بدون تقسيم)، كتلتين 8×16 أو 16×8 ، أربع كتل 8×8 . يمكن كذلك تقسيم الكتل الجزئية 8×8 بعدة طرائق: 8×8 (بدون تقسيم)، كتلتين 8×4 أو 4×8 ، أربع كتل 4×4 . يؤدي إجراء التنبؤ بالحركة على كتل ماكروية بقياسات مختلفة إلى تحسين أداء الضغط. وتجرى عملية التنبؤ على مرحلتين، تُستخدم في المرحلة الأولى الكتل $[16 \times 16 - 16 \times 8 - 8 \times 16 - 8 \times 8]$ ، فإذا وقع الاختيار على الكتلة 8×8 تبدأ المرحلة الثانية بتنعيم عملية تقدير الحركة باستخدام الكتل $[8 \times 8 - 4 \times 8 - 8 \times 4 - 4 \times 4]$ ويقع

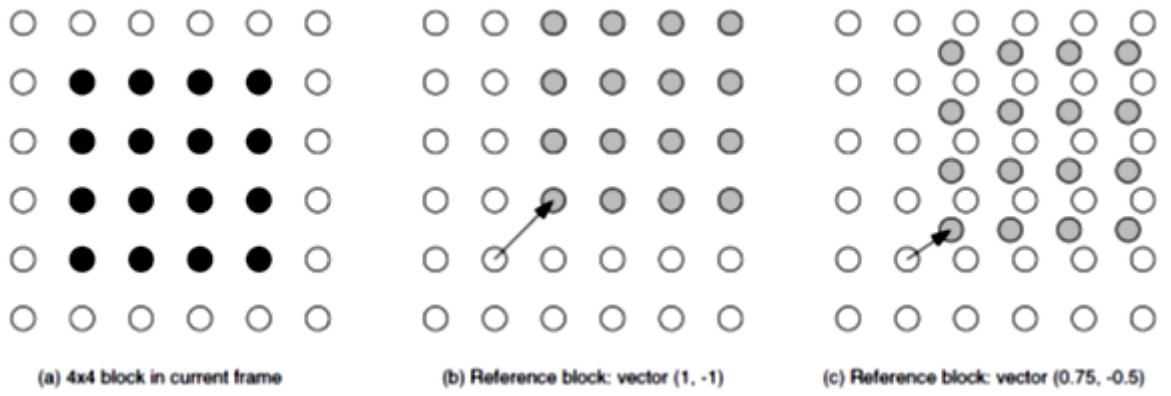
الاختيار في النهاية على الكتلة التي تحقق أقل قيمة للفرق SAD بينها وبين الكتلة الأصلية من أجل تحقيق أفضل أداء. يُظهر الشكل (14.2) طريقة تقسيم الكتل الماكروية، وجرى اختيار الكتل الماكروية 16×16 (بدون تقسيم) في المناطق التي تتسم بحركة بسيطة، بينما جرى اختيار كتل جزئية أصغر وذلك تبعاً لتفاصيل الحركة.



الشكل (14.2): مثال على تقسيم الكتل الماكروية [6]

٢-٢-٢-٢ أشعة الحركة بميز ربع بيكسل

تدل أشعة الحركة على الموقع النسبي للكتلة الحالية في الصورة المرجعية [16]. تتمتع أشعة الحركة في المعيار H.264 بميز ربع بيكسل. يبين الشكل (15.2) مثالاً على ذلك. يجري في هذا المثال التنبؤ بحركة كتلة جزئية بقياس 4×4 في الصورة الحالية انطلاقاً من الصورة المرجعية كما في الشكل (a15.2).



الشكل (15.2): مثال عن أشعة الحركة [6]

يمكن لشعاع الحركة أن يكون بميز بيكسل واحد (الشكل 15.2 b) أو بميز ربع بيكسل (الشكل 15.2 c). يجري حساب البيكسلات الجزئية (بميز $\frac{1}{4}$ بيكسل) على مرحلتين. تحسب البيكسلات الجزئية، في المرحلة الأولى، بميز $\frac{1}{2}$ بيكسل بتطبيق مرشح أفقي وعمودي كما هو

موضح في الشكل (16.2) [16]. تظهر البيكسلات الأساسية بالرمادي والبيكسلات الجزئية بالأبيض. ويجري حساب البيكسلات الجزئية b و h (بميزر $\frac{1}{2}$ بيكسل) انطلاقاً من بيكسلات الصورة المرجعية كما هو مبين في العلاقات (a.2.2) إلى (d.2.2).

$$b1 = (E - 5F + 20G + 20H - 5I + J) \quad (a.2.2)$$

$$h1 = (A - 5C + 20G + 20M - 5R + T) \quad (b.2.2)$$

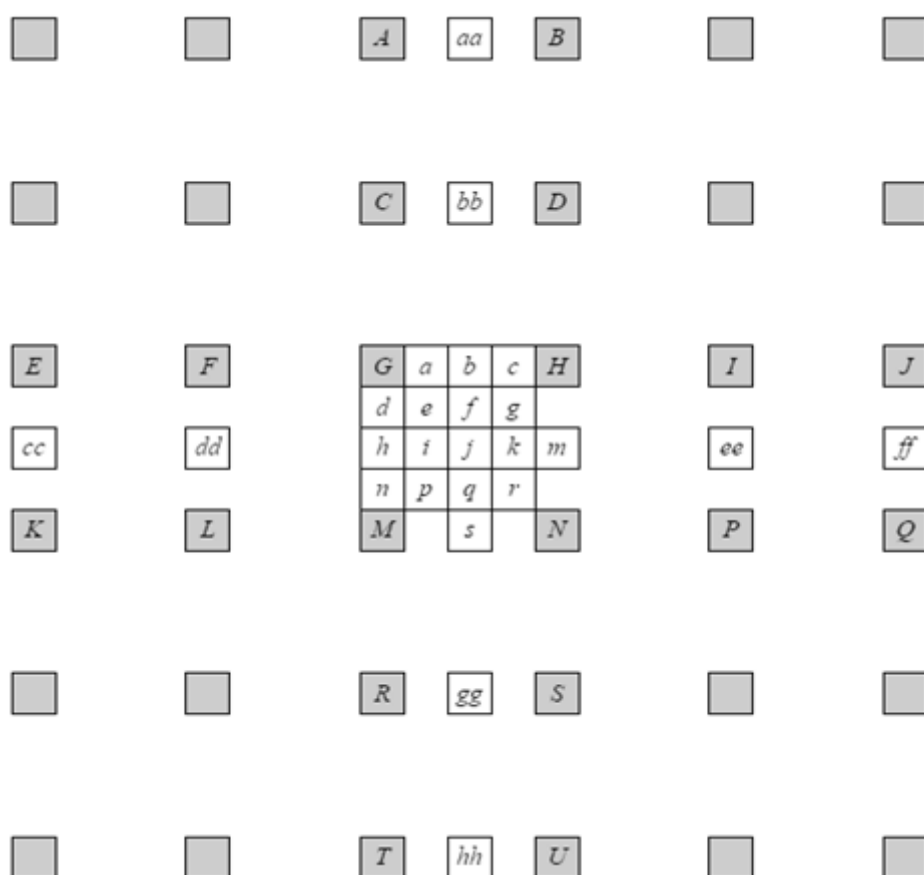
$$b = (b1 + 16) \gg 5 \quad (c.2.2)$$

$$h = (h1 + 16) \gg 5 \quad (d.2.2)$$

وتحسب البيكسلات الجزئية (بميزر $\frac{1}{4}$ بيكسل) كمتوسط حسابي للبيكسلات الجزئية (بميزر $\frac{1}{2}$ بيكسل) كما هو مَعطى بالعلاقين (a.3.2) و (b.3.2).

$$a = (G + b + 1) \gg 1 \quad (a.3.2)$$

$$c = (b + H + 1) \gg 1 \quad (b.3.2)$$



الشكل (16.2): مثال على حساب البيكسلات الجزئية [6]

٢-٣-٢ - التحويل والتكمية Transform/Quantization

تلي عملية التنبؤ الداخلي أو البيني، عملية التحويل والتكمية. إذ يجري أولاً حساب الكتلة المتبقية بواسطة طرح الكتلة الناتجة عن التنبؤ من الكتلة الأصلية. ويطبق على هذه الكتلة عملية تحويل ثنائية البعد هي تحويل التجيب المتقطع (Discrete Cosine Transform (DCT) [7]. ويعطى هذا التحويل وعكسه من أجل كتلة $N \times N$ ، بالعلاقين (a.4.2) و (b.4.2) [6]:

$$Y_{xy} = C_x C_y \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} X_{ij} \cos \frac{(2j+1)y\pi}{2N} \cos \frac{(2i+1)x\pi}{2N} \quad (a.4.2)$$

$$X_{ij} = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} C_x C_y Y_{xy} \cos \frac{(2j+1)y\pi}{2N} \cos \frac{(2i+1)x\pi}{2N} \quad (b.4.2)$$

و تُعطى فيها المعاملات c_i بالعلاقة (5.2):

$$C_i = \sqrt{\frac{1}{N}} \quad (i = 0), \quad C_i = \sqrt{\frac{2}{N}} \quad (i > 0) \quad (5.2)$$

تمثل X مصفوفة عينات صورة بقياس $N \times N$ ، وتمثل Y مصفوفة معاملات تحويل التجيب المتقطع DCT بالقياس نفسه.

يمكننا إعادة كتابة تحويل التجيب المتقطع $N \times N$ DCT وعكسه IDCT على شكل تحويل في فضاء المصفوفات اعتماداً على مصفوفة A بقياس $N \times N$ ، كما هو مبين في العلاقتين (a.6.2) و (b.6.2) [6].

$$\text{DCT:} \quad Y = A X A^T \quad (a.6.2)$$

$$\text{IDCT:} \quad X = A^T Y A \quad (b.6.2)$$

حيث يمكن التعبير عن عناصر المصفوفة A ، كما هو مُعطى بالعلاقة (7.2).

$$A_{ij} = C_i \cos \frac{(2j+1)i\pi}{2N} \quad (7.2)$$

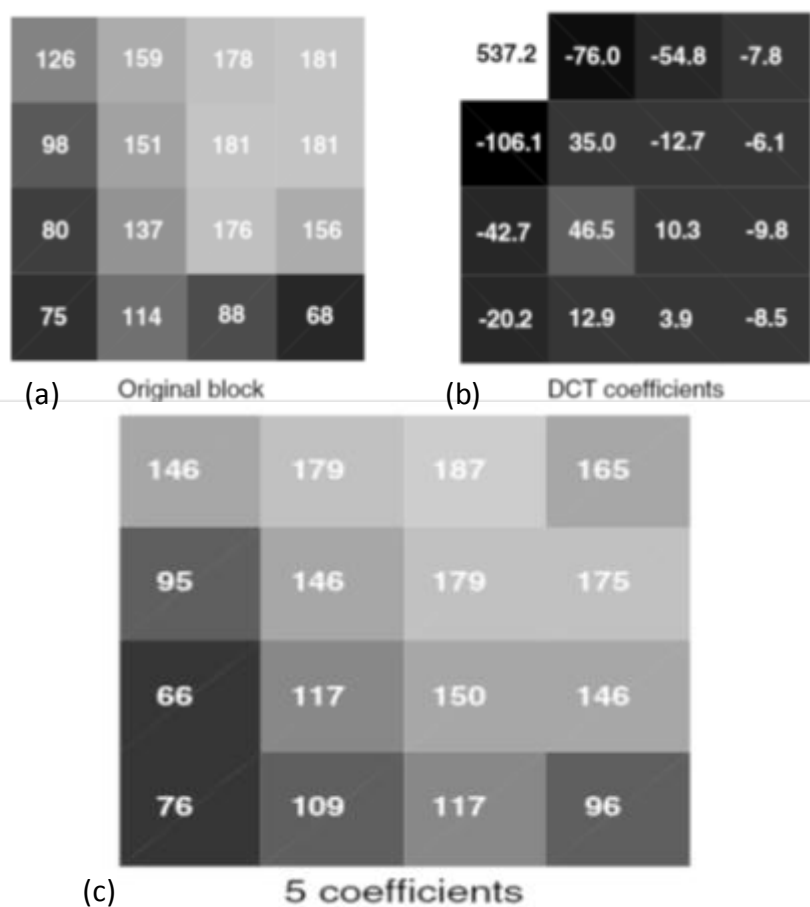
يجري في المعيار H.264 تطبيق التحويل DCT 4×4 على كتل ماكروية X بقياس 4×4 . وتُعطى المصفوفة A من أجل هذا التحويل، بالعلاقة (8.2).

$$A = \begin{bmatrix} \frac{1}{2} \cos(0) & \frac{1}{2} \cos(0) & \frac{1}{2} \cos(0) & \frac{1}{2} \cos(0) \\ \sqrt{\frac{1}{2}} \cos\left(\frac{\pi}{8}\right) & \sqrt{\frac{1}{2}} \cos\left(\frac{3\pi}{8}\right) & \sqrt{\frac{1}{2}} \cos\left(\frac{5\pi}{8}\right) & \sqrt{\frac{1}{2}} \cos\left(\frac{7\pi}{8}\right) \\ \sqrt{\frac{1}{2}} \cos\left(\frac{2\pi}{8}\right) & \sqrt{\frac{1}{2}} \cos\left(\frac{6\pi}{8}\right) & \sqrt{\frac{1}{2}} \cos\left(\frac{10\pi}{8}\right) & \sqrt{\frac{1}{2}} \cos\left(\frac{14\pi}{8}\right) \\ \sqrt{\frac{1}{2}} \cos\left(\frac{3\pi}{8}\right) & \sqrt{\frac{1}{2}} \cos\left(\frac{9\pi}{8}\right) & \sqrt{\frac{1}{2}} \cos\left(\frac{15\pi}{8}\right) & \sqrt{\frac{1}{2}} \cos\left(\frac{21\pi}{8}\right) \end{bmatrix} \quad (8.2)$$

بعد التعويض بقيم التجيب، تصبح المصفوفة مساوية للمصفوفة المُعطاة بالعلاقة (9.2).

$$A = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.653 & 0.271 & 0.271 & -0.653 \\ 0.5 & -0.5 & -0.5 & 0.5 \\ 0.271 & -0.653 & -0.653 & 0.271 \end{bmatrix} \quad (9.2)$$

يسمح استخدام المصفوفة A بإجراء التحويل DCT بواسطة عمليات حسابية بسيطة. يبين المثال المُوضح بالشكل (17.2) [6] فائدة تطبيق التحويل DCT.



الشكل (17.2): مثال عن تطبيق التحويل DCT [6]

يُلاحظ في (الشكل 17.2) المعاملات الناتجة عن عملية التحويل وذات الأوزان المهمة تتركز في الزاوية اليسارية العليا. يمكن ضغط هذه المعاملات بتصفير المعاملات ذات الأوزان الصغيرة. فمثلاً بعد تصفير 11 معامل والإبقاء على 5 معاملات مهمة فقط يمكن تطبيق التحويل العكسي IDCT للحصول على الكتلة المبينة في الشكل (17.2 c). يُلاحظ أن النتيجة مقاربة للكتلة الأصلية الشكل (17.2 a).

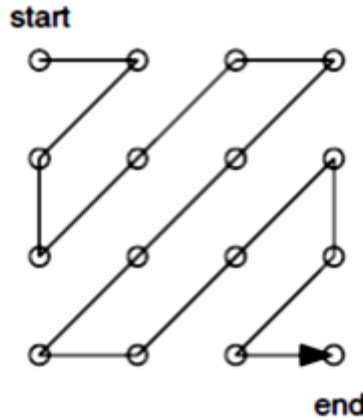
تقوم التكمية على التقسيم على عدد صحيح، وعلى إجراء التقريب إلى أقرب عدد صحيح، المعروف بالعلاقة (10.2) [6].

$$Z_{ij} = \text{round}(Y_{ij} / Q_{\text{step}}) \quad (10.2)$$

وفيها تمثل Y_{ij} معاملات التحويل السابق، وتمثل Q_{step} خطوة التكمية. يجري تحديد Q_{step} بواسطة معامل يدعى معامل التكمية QP. يتغير هذا المعامل من 0 إلى 51 مع تمثيل العينات على 8-bit. تتضاعف خطوة التكمية كلما زاد معامل التكمية بقيمة 6. ويسمح معامل التكمية بإجراء تسوية بين أداء عملية الضغط وبين سرعة إنجازها.

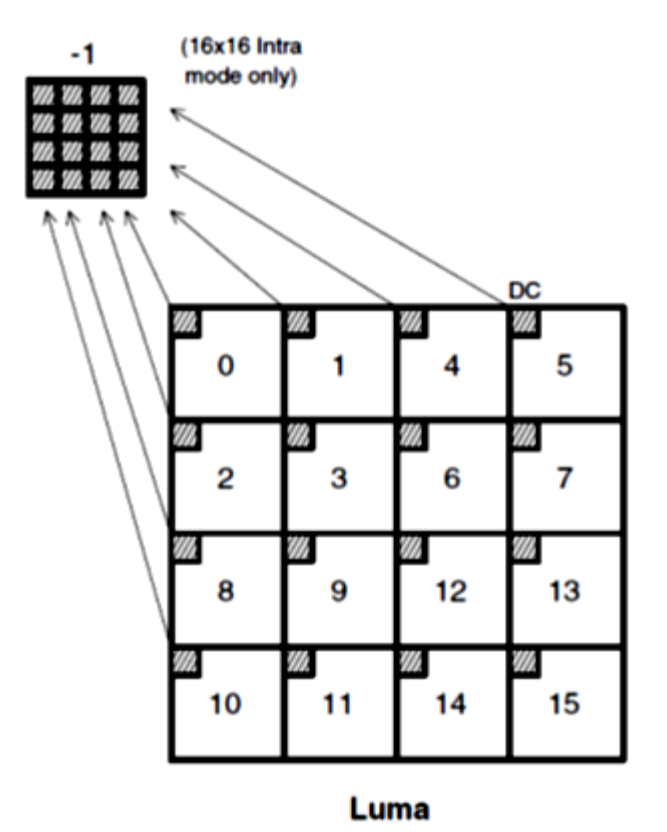
٢-٢-٤ - المسح وإعادة الترتيب Re-ordering/Scanning

بعد تطبيق عملية التحويل والتكمية يجري إرسال عينات كل كتلة قياسها 4×4 بيكسل بطريقة تعرجية zig-zag، كما يظهر في الشكل (18.2). و تسمح هذه الطريقة بفصل المعاملات المهمة عن المعاملات الصفرية.



الشكل (18.2): إعادة الترتيب بطريقة تعرجية zig-zag [6]

أما في حال ضغط الكتل بنمط داخلي 16×16 intra، فيجري إرسال المعاملات المستمرة DC، بشكل منفصل، ضمن كتلة بقياس 4×4 لكل كتلة بقياس 16×16 (الشكل 19.2).



الشكل (19.2): إرسال المعاملات المستمرة [6]

٢-٥- الترميز الانتروبي Entropy Coding

يهدف الترميز الانتروبي إلى ضغط المعلومات بالاستفادة من تكرارها إحصائياً حيث تُمثّل المعطيات الأكثر تكراراً بأقل عدد من البتات بينما تُمثّل الأقل تكراراً بعدد أكبر من البتات. تشمل المعلومات المرمّزة انتروبياً مجموعة متوسطات الصورة والفيديو sequence/picture parameter set، وترويسة الشريحة slice header، معطيات الفيديو كالكثل المتبقية residuals وأشعة الحركة وأنماط الكتل الماكروية [6].

تستخدم الترميزات ثابتة الطول fixed length codes من أجل ترميز المعلومات المختلفة عن معطيات الفيديو. أما معطيات الفيديو فيجري ترميزها باستخدام ترميز يتغير طوله بطريقة تفاعلية كالترميز متغير الطول المتكيف مع السياق (CAVLC) Context Adaptive Variable Length Coding أو الترميز الحسابي الثنائي المتكيف مع السياق Context Adaptive Binary Arithmetic Coding (CABAC). تُرمز كتل المتبقي residuals عموماً باستخدام الترميز CAVLC وبقيّة المعلومات المتغيرة الطول باستخدام الترميز Exp-Golomb.

٢-٢-٥-١ - الترميز Exp-Golomb

يملك هذا الترميز بنية بسيطة ومنتظمة (الجدول 1.2). تتألف كل كلمة ترميز من M صفراً، إضافةً إلى 1 في المنتصف، يليها M بت معلومة INFO. يصبح طول كل كلمة ترميز $2M+1$. يتشكل جزء المعلومة INFO كما هو مُعطى بالعلاقة (11.2).

$$\text{INFO} = \text{code_num} + 1 - 2^M \quad (11.2)$$

code_num	codeword
0	1
1	010
2	011
3	00100
4	00101
5	00110
6	00111
7	0001000
8	0001001
...	...

[M zeros][1][INFO]

الجدول (1.2): الترميز Exp-Golomb [6]

٢-٢-٥-٢ - الترميز متغير الطول المتكيف مع السياق CAVLC

يستخدم هذا الترميز بسبب أدائه الجيد لترميز مركبات كتل المتبقي residuals بعد إجراء التحويل والتكمية وإعادة الترتيب. تتصف هذه الطريقة بالخواص الآتية:

- تحتوي الكتل الناتجة عن إجراء التحويل والتكمية وإعادة الترتيب على كثير من الأصفار.
- تحتوي الكتل الناتجة على سلاسل من الواحدات الهابطة (Trailing Ones) ± 1 .
- يوجد تقارب بين عدد العناصر غير الصفورية في الكتل المتجاورة. فيجري ترميز هذا العدد باستخدام جداول خاصة بحيث يُعتمد اختيار الجدول المناسب بحسب عدد العناصر غير الصفورية.
- تزداد قيم العناصر غير الصفورية باتجاه بداية كل كتلة (بعد إعادة ترتيبها بطريقة zigzag). ويمكن الاستفادة من هذه الخاصية من أجل اختيار تمثيل مناسب لهذه العناصر بحسب قيمها.

يجري ترميز كل كتلة وفق ترتيب معين. ففي البداية يرسل عدد العناصر غير الصفريّة coeff_token ثم سلسلة القيم الواحدة الهابطة ± 1 trailing_ones ثم قيم العناصر غير الصفريّة level-prefix وهكذا. وتستخدم جداول خاصة لكل نوع من أنواع المعطيات كما يظهر في الجدول (2.2) وفي الجدول (3.2). سنأتي في الفصل الثالث على شرح هذا الترميز بالتفصيل بوساطة مثال.

Level_prefix	Bit string
0	1
1	01
2	001
3	0001
4	0000 1
5	0000 01
6	0000 001
7	0000 0001
.....	

الجدول (2.2): ترميز المعاملات غير الصفريّة [6]

Total_zeros	TotalCoeff (coeff_token)		
	1	2	3
0	1	111	0101
1	011	110	111
2	010	101	110
3	0011	100	101
4	0010	011	0100
5	0001 1	0101	0011
6	0001 0	0100	100
7	0000 11	0011	011
.....			

الجدول (3.2): ترميز المعامل Total_zeros [6]

٢-٢-٦ - مرشح التنعيم De-blocking filter

إن تولّد كتل مرئية مشوّهة عند الحواف هي إحدى مشاكل الضغط الكتلّي، لذا يستخدم المعيار H.264 مرشح تنعيم حلقي in loop deblocking filter. يهدف هذا الترشيح إلى تنعيم حواف الكتل لتحسين الصورة حيث يجري تطبيقه على الحواف الأفقية والعمودية للكتل بقياس 4×4 [6]. ترتبط فعالية الترشيح بمعامل التكمية QP كما هو مبين في الشكلين (a.20.2) و (b.20.2) والذان يوضحان كيفية تحسين الصورة بعد تطبيق المرشح.



الشكل (a.20.2) Qp=32 (no filter)



الشكل (b.20.2) Qp=32 (with filter)

الشكل (20.2): تأثير ترشيح الحواف

على تحسين الصورة [6]

الفصل الثالث

تصميم مفك الضغط

Decompression Unit Design

تشتمل مراحل عملية تصميم وتنفيذ نظام تعمية وفك تعمية وفك ضغط مقطع فيديو مُعمى وفق المعيار AES ومضغوط وفق المعيار H.264 على:

- تخزين مقطع فيديو مضغوط بالصيغة (264). نفي ملف نصي.
 - تحميل هذا الملف ضمن ذاكرة ROM.
 - تعمية وفك تعمية وفك ضغط مقطع الفيديو بوساطة الكيان الصلب إضافةً إلى إظهار النتيجة على شاشة عرض.
- حيث أن الجزء الأكثر تعقيداً هو تصميم الكيان الصلب لمفكك ضغط الفيديو سنقدم في هذا الفصل أجزاء ومراحل تصميم مفكك الضغط كنظام متكامل يقبل على دخله مقطع الفيديو المضغوط ويظهر على خرجه الفيديو بعد فك ضغطه.

٣-١ - دراسة مرجعية حول التصميم العتادي لمفككات ضغط الفيديو وفق

المعيار H.264

تسعى معظم الأبحاث المتعلقة بالتصميم العتادي لأنظمة ضغط أو فك ضغط الفيديو إلى واحد أو أكثر من الأهداف الآتية:

- تخفيض استهلاك الطاقة.
 - تخفيض مساحة الذاكرة المستخدمة.
 - تسريع عمليات الضغط وفك الضغط.
- و تستخدم عدة تقنيات من أجل الوصول إلى هذه الغاية ومنها:
- إضافة التوازي Parallelism وتقنية المعالجة التواردية Pipeline.
 - استخدام بنيان متعددة النواة Multi-Core Architecture.
 - أمثلة الذاكرة Memory Optimization.

٣-١-١ - إضافة التوازي Parallelism والمعالجة التواردية Pipeline

تسمح المعالجة التواردية بتقسيم المهمة الأساسية إلى مهام فرعية وتوزيع هذه المهام على عدد من الوحدات بحيث تقوم كل وحدة بإجراء جزء من المهمة. ومن ثمّ يمكن أن تعمل كل الوحدات بشكل متواصل وعلى التوازي. فمثلاً عندما تبدأ الوحدة الثانية بتنفيذ الجزء الثاني من المهمة، تكون الوحدة الأولى قد بدأت بتنفيذ الجزء الأول من المهمة التالية وهكذا دواليك.

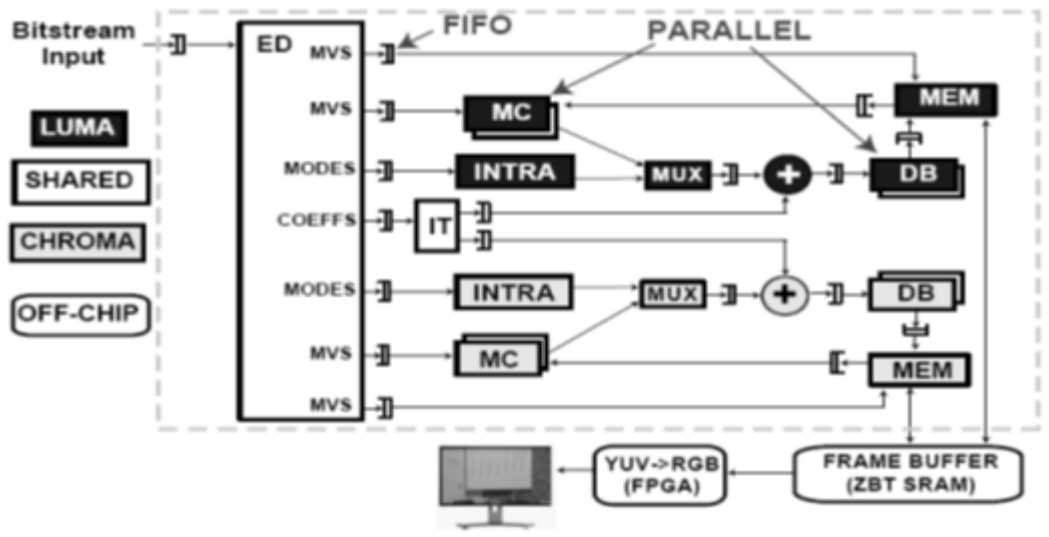
جرى تطبيق المعالجة التواردية في [21][23] على مختلف وحدات مفكك الضغط مع إضافة ذواكر FIFO بأطوال مختلفة بين هذه الوحدات. كما جرى تطبيق المعالجة التواردية في [24] بعرض أربع بيكسلات مما أدى إلى تسريع المعالجة، إضافةً إلى إدخال التوازي على مرشح

تنعيم الحواف Deblocking Filter بواسطة تشغيل مرشحين معاً، واحد عمودي وآخر أفقي قياس كل منهما 4×4 . أما في [23]، فقد جرى تطبيق المعالجة التواردية بطريقة هجينة hybrid بحيث تطبق في معظم وحدات مفكك الضغط على كتل قياسها 4×4 ، بينما تطبق في مرشح الحواف Deblocking Filter على كتل قياسها 16×16 .

يقوم [25] بإدخال التوازي على وحدة التحويل العكسي IDCT (ثنائية البعد) وذلك بتطبيق التحويل العكسي أحادي البعد 1-D IDCT، على التوازي، أفقياً وعمودياً. كما تضاف المعالجة التواردية لوحدة التنبؤ البيني في [26]، بحيث يمكنها معالجة كل كتلة 4×4 خلال 9-4 أدوار ساعة. ويستطيع مرشح الحواف في [21][23] معالجة أربع كتل 4×4 خلال دورة ساعة واحدة.

تُستخدم في [27] مجموعة هرمية من جداول التقابل hierarchical Look Up Table (LUT) من أجل فك الترميز الانتروبي CAVLD مما يسمح بقراءة عدة جداول على التوازي و تسريع المعالجة. بينما يقوم [28] بتسريع فك الترميز CAVLD بواسطة معالجة أكثر من معامل في دورة واحدة. يمكن إضافة التوازي على مستوى الإطارات frame parallelism باستخدام عدد من مفككات الضغط [36].

نقدم كمثال لتطبيق تقنية التوارد pipeline ما ورد في المرجع [31]. حيث جرى تطبيق المعالجة التواردية على مختلف وحدات مفكك الضغط مع إضافة نواكر FIFO بأطوال مختلفة بين هذه الوحدات، كما هو مبين في الشكل (1.3).



الشكل (1.3): المخطط الصندوقي لمفكك الضغط H.264 [31]

ED: Entropy Decoder, IT: Inverse Transform
MC: Motion Compensation, INTRA: Spatial Prediction
DB: Deblocking Filter, MEM: Memory Controller
FB: Frame Buffer

تسمح تقنية التوارد pipeline بمعالجة عدة كتل 4×4 على التوازي، لكن يختلف عدد الدورات (cycles) اللازمة لمعالجة كل كتلة تبعاً لوحدة المعالجة. فمثلاً يتعلق عدد الدورات اللازمة لوحدة تفكيك الترميز الانتروبي ED بعدد المعاملات غير الصفرية وبأشعة الحركة وغيرها من العناصر وقد يصل عدد الدورات إلى 33 cycles، بينما قد يصل عدد الدورات اللازمة لوحدة تعويض الحركة MC بأشعة الحركة (بوجود أو عدم وجود جزء كسري لشعاع الحركة) إلى 9 cycles. ومن هنا جاءت ضرورة إضافة ذواكر FIFOs بين الوحدات المختلفة لتعويض الفروق بعدد الدورات بحيث تحتاج كل وحدة إلى الانتظار أقل ما يمكن. وتظهر النتائج تحسّن الأداء بنسبة 45% عند اختيار الذواكر FIFOs بعمق 4 خانات. ويمكن تحسين هذه النتيجة عند اختيار أطوال مختلفة ومناسبة لكل وحدة. فمثلاً يجب أن يكون طول FIFO بين الوحدة ED والوحدة IT أكبر ما يمكن.

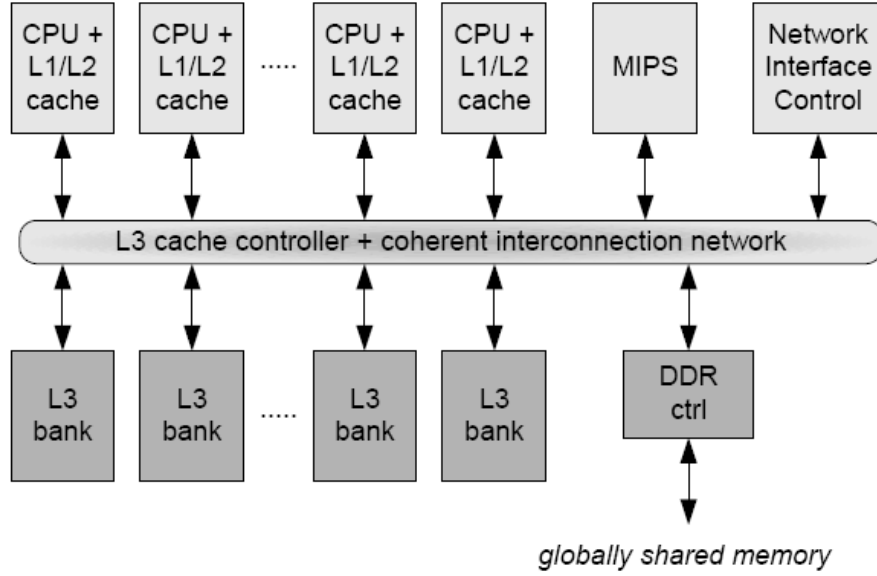
٣-١-٢ - استخدام بنية متعددة النواة Multi-Core Architecture

تظهر الدراسة في [19] أن عقدة التصميم في مفكك الضغط تكمن في وحدة فك الترميز الانتروبي، وذلك لصعوبة إدخال التوازي على الخوارزمية الخاصة به وهي CAVLD. نجد ذلك أيضاً في [29][30] حيث جرى تكرار جميع وحدات مفكك الضغط عدة وحدات فك الترميز الانتروبي. تتمثل إحدى الطرق لتسريع وحدة فك الترميز الانتروبي في تشغيل هذه الوحدة بتردد أعلى [22][31]، لكن تبقى هذه الطريقة محدودة لوجود سقف أعلى للتردد. فيجرى اللجوء إلى طرائق أخرى، كما هو الحال في [32][33]، وذلك بتقسيم الفيديو إلى شرائح slices يمكن معالجة كل منها بواسطة نواة عتادية منفردة. يسمح ذلك بتسريع عمل مفكك الضغط. ويمكن الوصول لنفس الغاية باستخدام بنية متعددة المعالج [29][30]، حيث يجري معالجة الكتل الماكروية بشكل قطري وذلك بسبب وجود تبعية في المعطيات بين الكتل الماكروية.

نقدم كمثال لتطبيق بنية متعددة النواة ما ورد في [30]. يظهر الشكل (2.3) البنية متعددة النواة المستخدمة لتطبيق مفكك الضغط. وتتألف من 8 معالجات صغيرة من نوع TriMedia VLIW تعمل بتردد 600MHz، مع وجود هرمية بأربع مستويات للذواكر: مستويين L1/L2-cache لكل معالج حيث L2-cache 128-kbyte ومستوى L3-cache 12-MByte مشترك بين المعالجات ومستوى آخر DDR SDRAM خارجية.

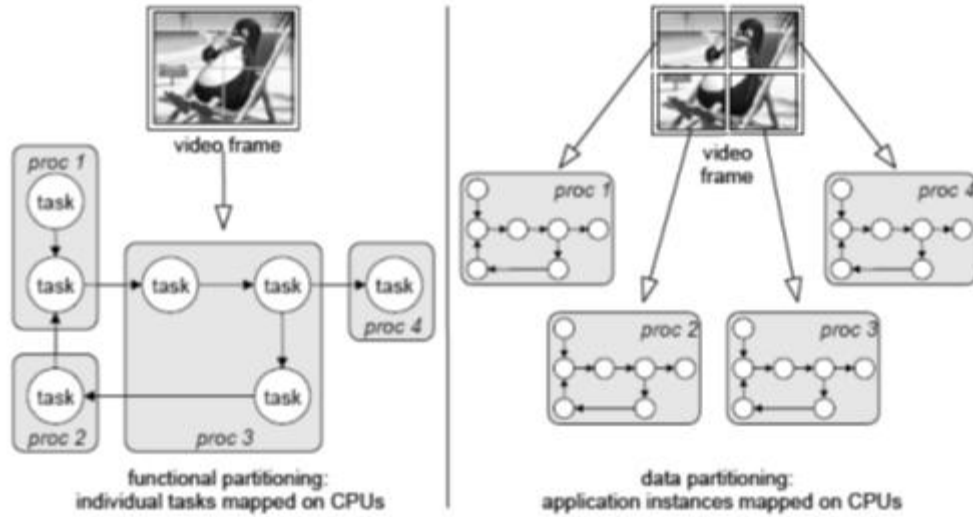
يظهر العمل في [30] استخدام بنية متعددة المعالج multiprocessor architecture لتنفيذ مفكك الضغط H.264 وذلك بطريقتين: إما بواسطة تجزئة إطار المعطيات، حيث يقوم كل

معالج بتفكيك ضغط جزء من الإطار، أو بواسطة تجزئة المعالجة، حيث يقوم كل معالج بإجراء جزء من خوارزمية التفكيك، كما يظهر في الشكل (3.3).



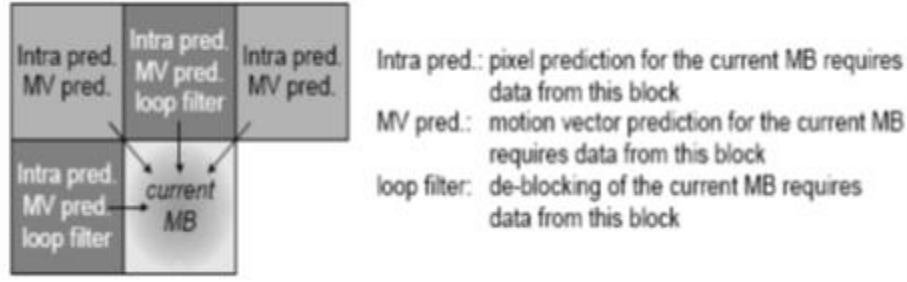
الشكل (2.3): البنية متعددة النواة المستخدمة في [30]

تبدو طريقة تجزئة المعالجة مناسبة أكثر للمعالجة التسلسلية وهي تسمية أخرى لتقنية التوارد pipeline. وتكون طريقة تجزئة إطار المعطيات فعالة أكثر عند اختيار أجزاء مستقلة أكثر ما يمكن.



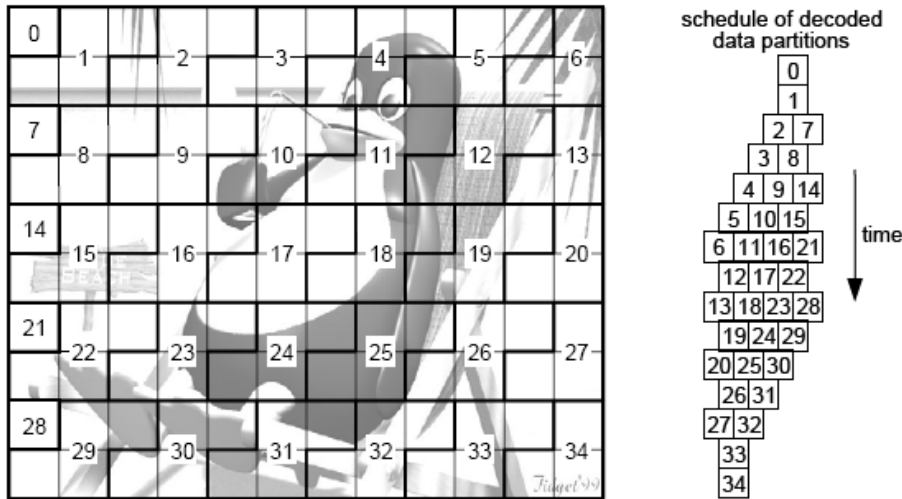
الشكل (3.3): طرق توزيع العمل [30]

نعلم أن ارتباط المعطيات بين الكتل الماكروية ينتج إما عن التنبؤ الداخلي أو البيني أو مرشح الحواف، كما يظهر في الشكل (4.3).



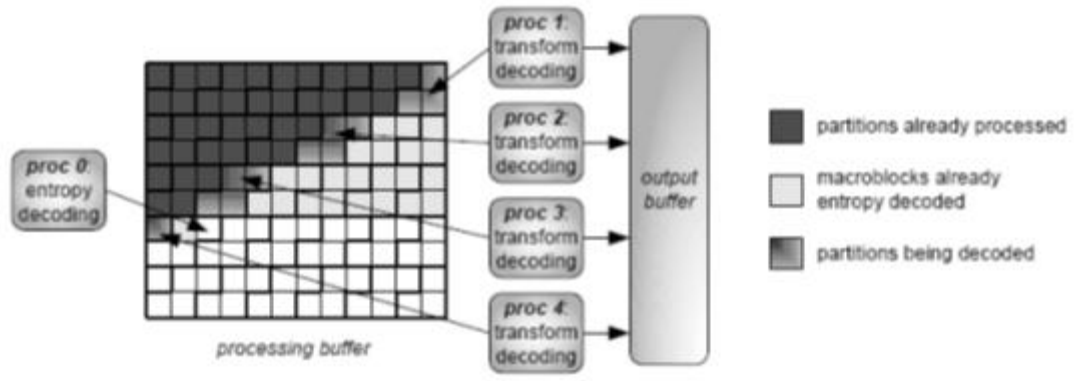
الشكل (4.3): ارتباط المعطيات بين الكتل الماكروية [30]

وبسبب هذا الارتباط جرى اختيار تجزئة الإطار كما هو مبين في الشكل (5.3)، حيث يتألف كل جزء من أربعة كتل ماكروية، ويظهر على يمين الشكل ترتيب معالجة هذه الأجزاء بسبب ارتباط المعطيات. ونحتاج على الأكثر إلى أربع معالجات تعمل على التوازي لمعالجة الأجزاء 6, 11, 16, 21 مثلاً.



الشكل (5.3): تجزئة إطار المعطيات [30]

أما بالنسبة لتفكيك الترميز الانتروبي وبسبب طبيعة المعالجة التسلسلية الخاصة به، فمن الأفضل استخدام طريقة تجزئة المعالجة في هذه الحالة وذلك بتخصيص معالج خاص يقوم بهذه المهمة ومن بعدها يجري تطبيق تجزئة الإطار كما ورد سابقاً. يظهر الشكل (6.3) كيفية دمج الطريقتين من أجل الحصول على أفضل أداء.



الشكل (6.3): دمج طريقتي تجزئة المعالجة وتجزئة الإطار [30]

3-1-3 - أمثلة الذاكرة Memory Optimization

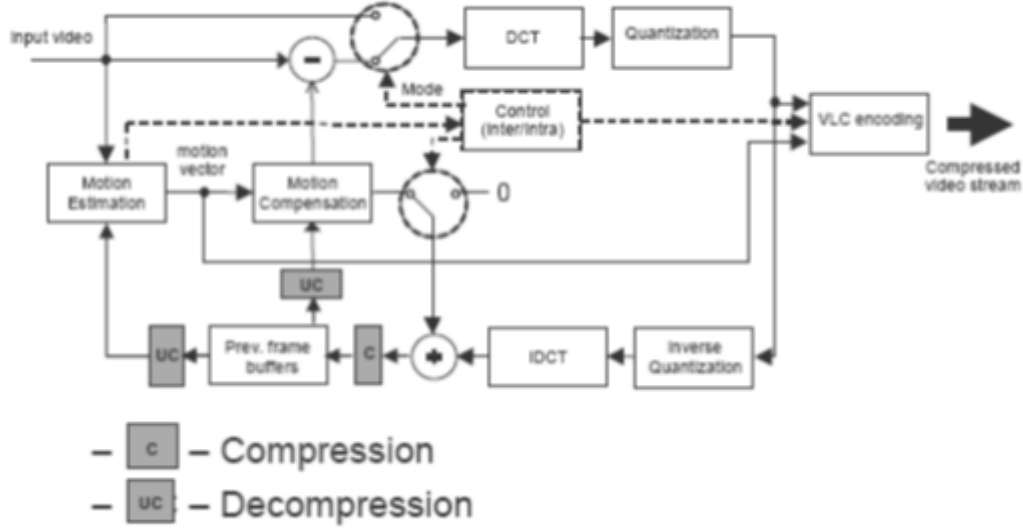
تقوم بعض الدراسات على إضافة ذواكر من نوع last-line cache من أجل تقليل استدعاء الذاكرة الأساسية [21][23]. فيسمح استخدام ذاكرتين cache بحجم 1kByte، واحدة للمركبة الضوئية والأخرى للمركبة اللونية بتخفيض أكثر من 30% من قراءة وحدة التنبؤ البيئي للذاكرة الأساسية. ومن أجل تخفيف الضغط على الذاكرة الأساسية، تضاف في [20][34] وحدات تخزين محلية local buffers لتخزين المعطيات المتداخلة بين الكتل المتجاورة بقياس 4×4 في وحدة التنبؤ البيئي.

إضافةً إلى تقنية on-chip caching تستخدم تقنية أخرى وهي ضغط الإطارات المرجعية [35]، من أجل تقليل استدعاء الذاكرة الأساسية. ويمكن تخفيض 25% من حجم الذاكرة عند ضغط البيكسلات من 8-bits إلى 6-bits وذلك على حساب تخفيض سرعة المعالجة بسبب إضافة عمليات الضغط وفك الضغط.

يظهر العمل في [30] كيفية تخفيض استدعاء الذاكرة الأساسية باستخدام بنيان متعددة المعالجات multiprocessor architecture وذلك بطريقتين: أما بوساطة تجزئة إطار المعطيات، حيث يقوم كل معالج بتفكيك ضغط جزء من الإطار أو بوساطة تجزئة المعالجة، فيقوم كل معالج بإجراء جزء من خوارزمية التفكيك. تظهر النتائج أن الطريقة الأولى تتفوق بنسبة 65%.

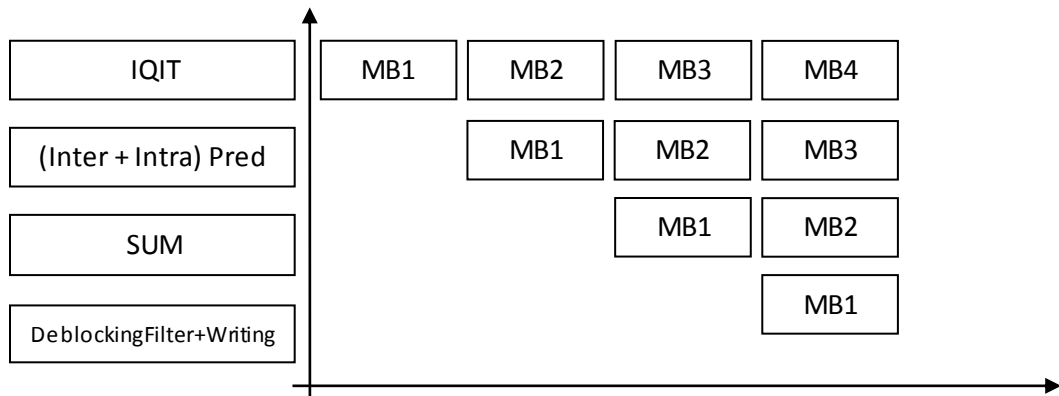
نقدم كمثال لتطبيق أمثلة الذاكرة ما ورد في [35]. يظهر الشكل (7.3) الفكرة الأساسية من تطبيق عملية ضغط وفك ضغط الإطارات المرجعية. حيث يجرى ضغطها قبل الكتابة في الذاكرة مما يقلل من حجم الذاكرة وعدد عمليات الكتابة، بينما يجرى فك ضغط الإطارات بعد القراءة من الذاكرة مما يقلل من عدد عمليات القراءة. ومن ثمَّ ينجم عن تخفيض حجم الذاكرة تخفيض السعر والحجم الفيزيائي والوزن. إما تقليل عدد عمليات الكتابة والقراءة من الذاكرة

فيقود إلى تخفيض استهلاك الطاقة. وقد أظهرت النتائج أن استخدام ضغط الإطارات المرجعية، أدى إلى تخفيض كل من حجم واستهلاك الذاكرة بنسبة 25%.



الشكل (7.3): تقنية ضغط وفك ضغط الإطارات المرجعية [35]

أما في بحثنا، فقد قمنا باستخدام ذاكرتين مزدوجتين واحدة من أجل المركبة الضوئية والأخرى من أجل المركبة اللونية، وذلك من أجل المحافظة على مسار المعطيات ومن أجل معالجة المركبتين الضوئية واللونية على التوازي. كما أننا استخدمنا تقنية المعالجة التواردية على مستوى الكتل الماكروية، كما هو مبين في الشكل (8.3)، حيث يجري تقسيم وحدة معالجة الكتل الماكروية إلى عدة خطوط معالجة وبأطوال متجانسة حتى يمكن لها العمل على التوازي وبسرعات متقاربة. يمكننا في هذه الحالة تسريع المعالجة الكلية بما يتناسب مع عدد خطوط المعالجة التواردية.



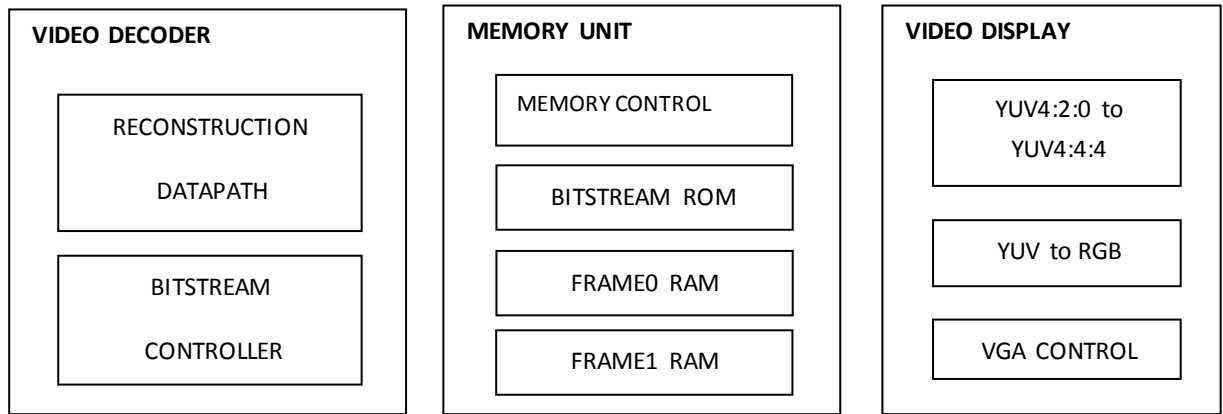
الشكل (8.3): تطبيق تقنية المعالجة التواردية على مستوى الكتل الماكروية

٣-٢ - المخطط الصندوقي لمفك الضغط H.264 decoder

يتألف مفك الضغط H.264 decoder الذي قمنا بتصميمه وبناءه في هذا البحث من ثلاثة أجزاء رئيسية (الشكل 9.3):

- وحدة الذاكرة Memory Block.
- وحدة الإظهار Video Display.
- مفك الضغط Video Decoder.

سنأتي على شرح كل من هذه الأجزاء تباعاً.



الشكل (9.3): المخطط الصندوقي لمفك الضغط

٣-٢-١ - وحدة الذاكرة Memory Block

تحتوي هذه الوحدة على ثلاثة ذواكر: ذاكرة لتخزين ملف الدخل وذاكرة مزدوجة لتخزين فيديو الخرج. تسمح ذاكرة الخرج المزدوجة بالحفاظ على مسار المعطيات، بحيث يجري على إحداهما تخزين إطار الفيديو الفردي وعلى الأخرى تخزين إطار الفيديو الزوجي. حيث تقوم وحدة مفك الضغط بكتابة الإطار الزوجي في الذاكرة الأولى، في حين يجري على التوازي قراءة الإطار الفردي من الذاكرة الثانية، ثم يجري تبادل الأدوار. إذاً تتألف هذه الوحدة من الأجزاء الآتية:

- ذاكرة BitStream_Rom بحجم $32K \times 16$ bits لتخزين مقطع الفيديو المضغوط. وجرى اختيار هذه الذاكرة بحجم يكفي لتخزين فيديو مضغوط وبمميز 176×144 بيكسل ومؤلف من 300 إطار.
- ذاكرة Frame0_Ram بحجم 16384×32 bits لتخزين إطار زوجي من الفيديو بعد فك ضغطه.
- ذاكرة Frame1_Ram بحجم 16384×32 bits لتخزين إطار فردي من الفيديو بعد فك ضغطه.

▪ متحكم Memory Controller لقراءة الإطارات الزوجية والفردية على التتابع من أجل إظهارها على شاشة العرض.

يجري تمثيل كل عينة من المركبات الضوئية Y/Cr/Cb على 8-bits. يتكون كل إطار من 176×144 بيكسل. يجري هنا أخذ العينات وفق التشكيلة YCrCb (4:2:0). من أجل كل بيكسل يجري تخزين مركبته Y، أي 25344 عينة من نوع Y. بينما يجري أخذ ربع عدد المركبات Cr/Cb، أي $6336 = 25344/4$ عينة لكل من Cr/Cb. ويخزن كل إطار في الذاكرة وفق الترتيب الآتي:

$$Y_0 Y_1 Y_2 Y_3 \dots Y_{25343} Cr_0 Cr_1 \dots Cr_{6335} Cb_0 Cb_1 \dots Cb_{6335}$$

يجري استخدام ذواكر بعرض 32-bits لتخزين الإطارات: Frame0_RAM/Frame1_RAM، و من ثمَّ يجري وضع كل أربع عينات في خانة واحدة. ومن ثمَّ نحتاج إلى $6336 = 25344/4$ خانة لتخزين عينة Y. كما نحتاج إلى $3168 = (6336 + 6336)/4$ خانة لتخزين المركبات Cr/Cb.

ويكون بذلك عدد الخانات المطلوبة لتخزين إطار واحد $9504 = (6336 + 3168)$ خانة بعرض 32-bits لكل خانة. لذلك كانت ذواكر الإطارات بحجم $16384 \times 32\text{bits}$.

٣-٢-٢ - وحدة الإظهار Video Display

تعمل هذه الوحدة على عرض كل إطار فيديو بعد فك ضغطه. وتتألف وحدة الإظهار من الأجزاء الآتية:

- وحدة تحويل العينات من النمط YCrCb(4:2:0) إلى النمط YCrCb(4:4:4). يجري أخذ العينات وفق النمط YCrCb(4:2:0) ومن ثمَّ يجري تخزين المركبات اللونية فقط من أجل الأسطر والأعمدة الزوجية. وعند إعادة تشكيل الصورة بأكملها وفقاً للنمط YCrCb(4:4:4)، تأخذ المركبات اللونية للأعمدة والأسطر الفردية نفس قيم المركبات اللونية للأعمدة والأسطر الزوجية المجاورة.
- وحدة تحويل معطيات الفيديو من الفضاء YCrCb إلى الفضاء RGB. يعتمد هذا التحويل على المعادلات الخطية (1.1) و (2.1) و (3.1).
- وحدة التحكم بعرض الفيديو VGA_Ctrl. تقوم هذه الوحدة بتزويد شاشة العرض بإشارات التزامن الأفقي والعمودي إضافةً إلى إشارات الألوان RGB. ويجري عرض كل إطار بميز 176×144 بيكسل.

٣-٢-٣ - وحدة تفكيك ضغط الفيديو Video Decoder

يشكل مفكك الضغط الوحدة الرئيسية في النظام. ويتألف من الأجزاء الآتية:

▪ وحدة التحكم بقراءة مقطع الفيديو المضغوط bitstream controller:

تحلل هذه الوحدة المعلومات المقروءة من المقطع المضغوط وتحدد نوعها ونوع الترميز الانتروبي الموافق لها. حيث يجري في طرف المرسل ترميز معطيات الفيديو كالكتل المتبقية residuals باستخدام الترميز CAVLC، بينما يُستخدم الترميز Exp-Golomb لبقية المعطيات متغيرة الطول. ويُستخدم الترميز الثابت الطول fixed length code لترميز المعلومات المختلفة عن معطيات الفيديو. إذا تفكك هذه الوحدة المعلومات المرمزة انتروبياً بكافة أنواعها وترسل النتائج إضافة إلى إشارات التحكم المناسبة إلى وحدة إعادة بناء الفيديو.

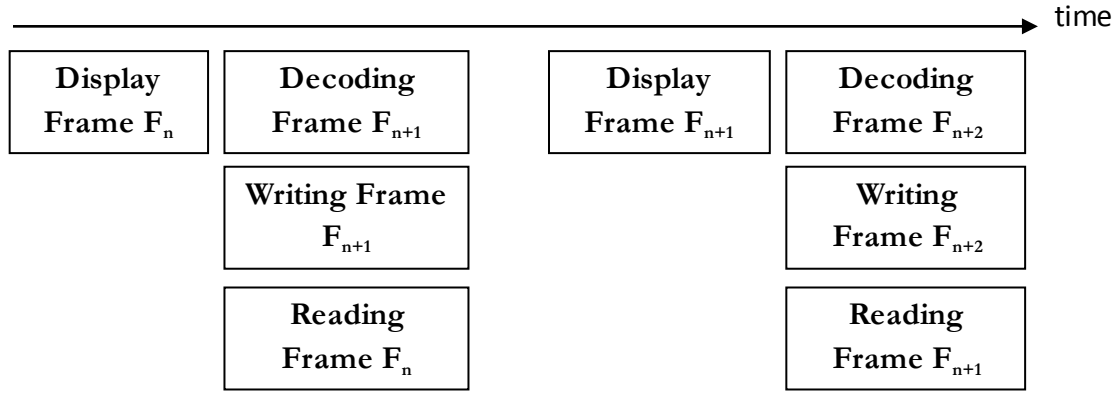
▪ وحدة إعادة بناء الفيديو reconstruction datapath:

تستقبل هذه الوحدة معطيات الفيديو إضافةً إلى موسطات النظام من وحدة التحكم السابقة بعد إتمام عملية فك الترميز الانتروبي. تطبق هذه الوحدة التكمية والتحويل العكسيين للحصول على كتل المتبقي residuals ثم تضيفها إلى كتل التنبؤ لإعادة بناء الصورة مع تطبيق مرشح التنعيم DF.

٣-٢-٤ - إدارة عمل وحدات النظام

تهدف هذه العملية إلى إدارة عمل وحدات النظام من قيادة وترتيب للمهام الموكلة إلى كل وحدة، مما يسمح بتسريع عمل النظام بحيث يجري تنفيذ ما يمكن من المهام على التفرع. تقاد هذه العملية من قبل متحكم الذاكرة Memory Controller الموضح في الشكل (9.3). تعمل وحدة الإظهار ومفكك الضغط بالتناوب. فعند عرض إطار معين على الشاشة (بميز 144×176) يتوقف مفكك الضغط عن العمل، وعند الانتهاء من العرض تجري إعادة تفعيل مفكك الضغط خلال الزمن المتبقي من العرض على الشاشة.

فبعد الانتهاء من عرض الإطار F_n ، تبدأ مرحلة فك ضغط الإطار F_{n+1} وكتابته في الذاكرة (الشكل 10.3). وتحتاج عملية فك الضغط إلى قراءة الإطار السابق F_n (الصورة المرجعية). يمكن إجراء العمليات الثلاث الأخيرة على التوازي بحيث تنتهي عملية فك ضغط الإطار F_{n+1} قبل عرضه على الشاشة. ومن ثمَّ تجري عملية فك الضغط بالزمن الحقيقي.

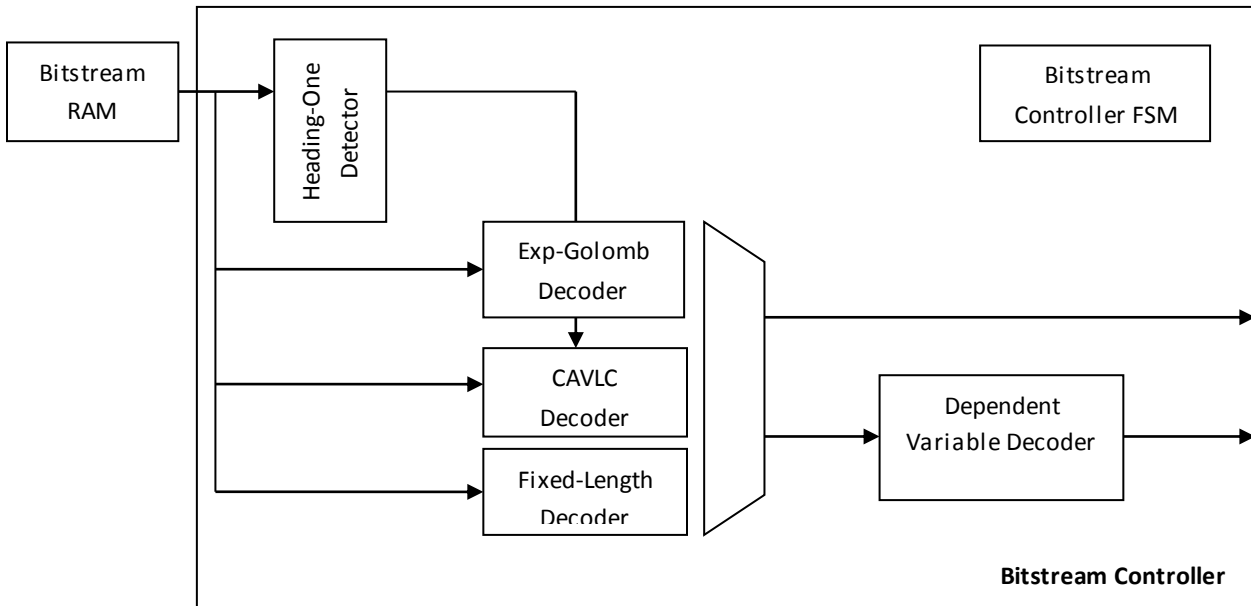


الشكل (10.3): ترتيب عمل وحدة الإظهار ومفك الضغط

يُعدُّ فك ضغط الفيديو العملية الأساسية والأكثر تعقيداً في النظام، كما ذكرنا سابقاً. لذلك سنشرح المكونين الأساسيين لوحدة تفكيك ضغط الفيديو وهما: وحدة التحكم بقراءة مقطع الفيديو المضغوط ووحدة إعادة بناء مقطع الفيديو.

٣-٣ - وحدة التحكم بقراءة مقطع الفيديو المضغوط

تقوم هذه الوحدة بقراءة مقطع الفيديو المضغوط وتحليل مكوناته. حيث يتكون المقطع المضغوط من طبقتين: طبقة مواعمة الشبكة NAL وطبقة ضغط الفيديو VLC. وتنقسم معطيات طبقة مواعمة الشبكة إلى نوعين من الوحدات: وحدات VCL-NAL تحمل معطيات الفيديو ووحدات non-VCL-NAL تحمل معلومات إضافية كموسطات النظام. يظهر الشكل (11.3) المخطط الصندوقي لهذه الوحدة.



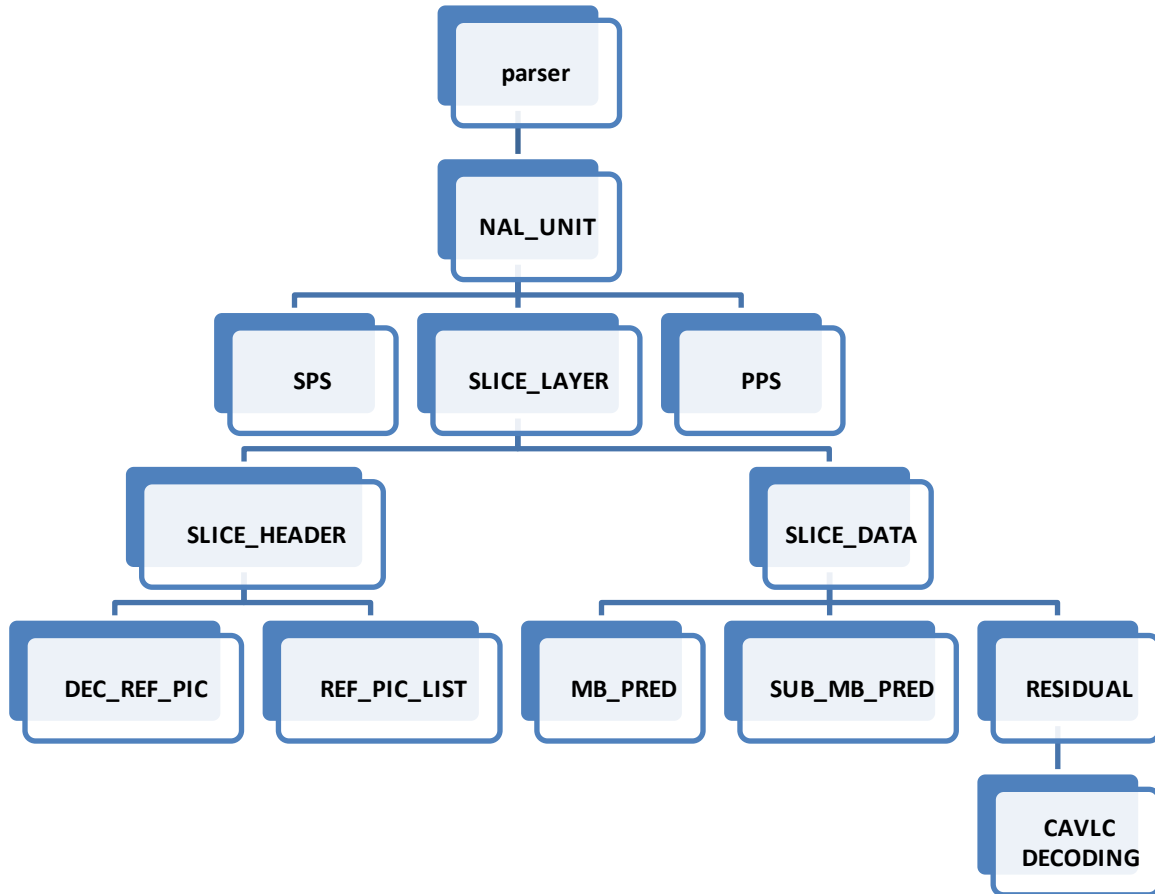
الشكل (11.3): المخطط الصندوقي لوحدة التحكم بقراءة مقطع الفيديو المضغوط [6]

وتحتوي هذه الوحدة على المكونات الأساسية الآتية:

- وحدة التحكم بدفق البتات Bitstream Controller FSM.
- كاشف البت الواحد Heading-One Detector.
- مفكك الترميز الانتروبي من نوع CAVLC.
- مفكك الترميز الانتروبي بطول ثابت Fixed-Length Decoder.
- مفكك الترميز الانتروبي من نوع Exp-Golomb.
- مفكك الترميز الانتروبي المتغير الطول Dependent Variable Decoder.

٣-١-٣ وحدة التحكم بدفق البتات Bitstream Controller FSM

تعمل وحدة التحكم هذه وفق مخطط آلة الحالة Finite State Machine (FSM) الموضح في الشكل (12.3).

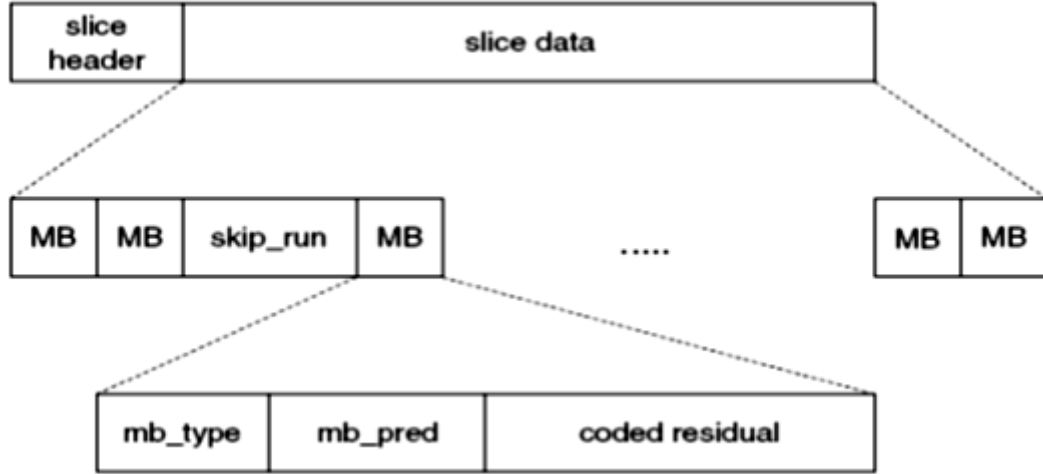


الشكل (12.3): مخطط آلة الحالة للتحكم بقراءة مقطع الفيديو المضغوط [6]

تقوم هذه الوحدة في المرحلة الأولى بتحديد بداية كل وحدة NAL_UNIT ومن ثم تحديد نوعها:

- طبقة الشريحة Slice Layer: تحتوي على معطيات الفيديو.

- مجموعة موسطات الصورة (PPS) Picture Parameter Set.
- مجموعة موسطات الفيديو (SPS) Sequence Parameter Set.
- وتقسم معطيات الفيديو بدورها إلى (الشكل 13.3):
- ترويسة الشريحة Slice Header: تحمل هذه الترويسة معلومات عن الصور المرجعية.
- معطيات الفيديو Slice Data والتي تنقسم بدورها إلى العناصر الآتية (الجدول 1.3):
 - كتلة المتبقي Residuals.
 - عنصر التنبؤ MB_Pred.
 - عنصر التنبؤ الجزئي SUB_MB_Pred.



الشكل (13.3): تقسيم معطيات الفيديو [6]

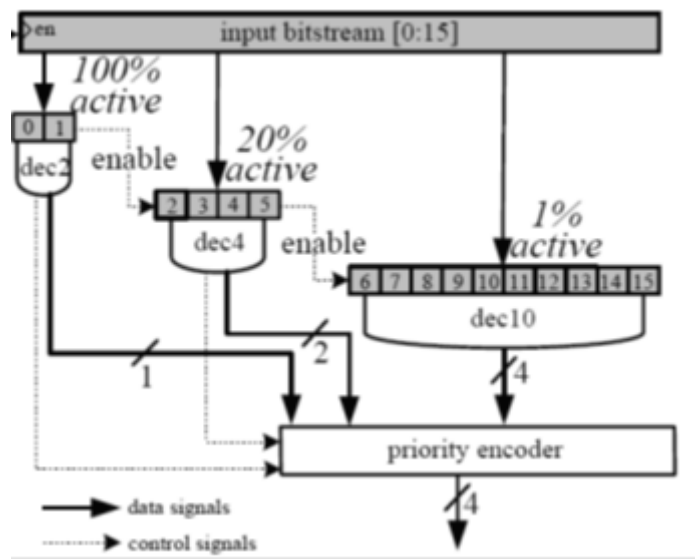
عناصر الكتلة الماكروية	عملها
mb_type	يحدد هذا العنصر طريقة ضغط الكتل الماكروية أي بطريقة inter أو intra. كما يحدد طريقة تقسيم الكتلة الماكروية (الشكل 13.2).
mb_pred	يحدد هذا العنصر نمط التنبؤ الداخلي (الشكل 8.2 و 9.2). كما يحدد أشعة الحركة التي جرى ترميزها تفاضلياً في جميع حالات التنبؤ البيئي عدا حالة تقسيم الكتلة الماكروية إلى أربع كتل بقياس 8×8.
sub_mb_pred	يحدد طريقة التقسيم الجزئي لكل كتلة بقياس 8×8 (الشكل 13.2)، إضافةً إلى أشعة الحركة الجزئية التي جرى ترميزها تفاضلياً.
Residual	يحتوي هذا العنصر على كتلة المتبقي التي جرى ضغطها (ترميزها).

الجدول (1.3): عناصر الكتلة الماكروية MB [6]

تعمل وحدة التحكم على قراءة المعلومات المضغوطة وتصنيفها وتوليد إشارات التحكم المناسبة لبقية الأطراف.

٣-٣-٢ - كاشف البت الواحد Heading-one Detector

يعتمد كاشف البت الواحد على كشف موقع أول بت واحد في كل كلمة مرمّزة. ويعتمد تفكيك الترميزين: CAVLC و Exp-Golomb على هذه العملية. يظهر في الشكل (14.3) تصميم كاشف البت الواحد. ينقسم الكاشف إلى ثلاثة أقسام. يعالج كل منها جزء من الكلمة المرمّزة، والتي يجري تمثيلها على 16-bits. يقوم القسم الأول Dec2 بالكشف عن وجود بت واحد في الجزء b_0b_1 من الكلمة المرمّزة.



الشكل (14.3): كاشف البت الواحد [6]

في حال عدم وجود بت واحد في هذا الجزء يجري تفعيل القسم الثاني Dec4 للكشف عن وجود بت واحد في الجزء $b_2b_3b_4b_5$ من الكلمة المرمّزة. وفي حال عدم وجود بت واحد في الجزء $b_2b_3b_4b_5$ يجري تفعيل القسم الأخير Dec10 للكشف عن وجود بت واحد في الجزء الأخير من الكلمة المرمّزة. يعتمد هذه التصميم على أن الاحتمال الأكبر لظهور بت واحد يتركز في البتين الأوليين من الكلمة (احتمال 80%). لذلك يجري فحص هذين البتين في البداية (الجزء الأول من الكلمة)، ثم فحص الأجزاء المتبقية في حال عدم ظهور بت واحد في الجزء الأول.

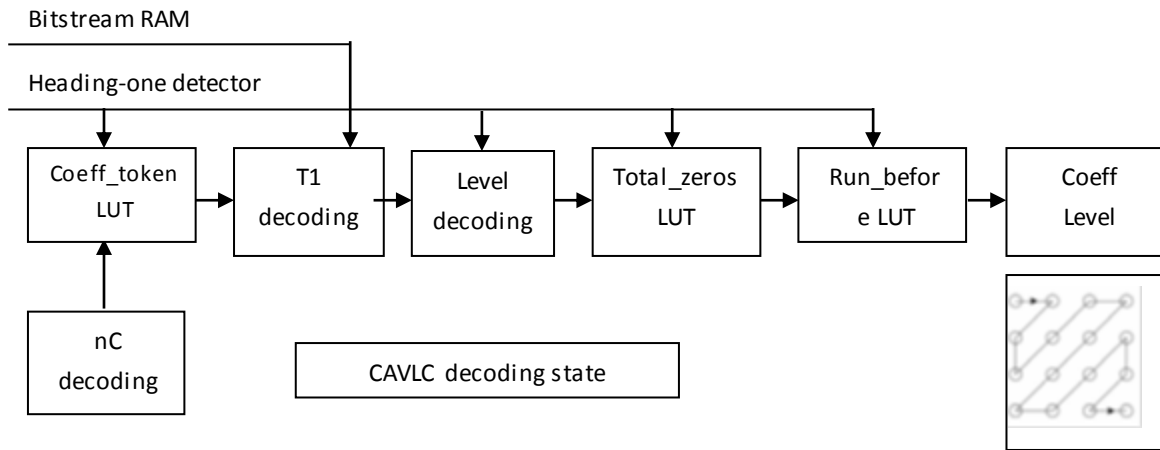
٣-٣-٣ - مفكك الترميز الانتروبي من نوع CAVLC

يؤدي إجراء عمليات التحويل والتكمية وإعادة الترتيب على كتل المتبقي بقياس 4×4 إلى إنتاج سلاسل بطول 16 bits تحتوي على كثير من الأصفار والواحد، وتتركز المعاملات المهمة

في بدايتها وتتشابه السلاسل مع مجاوراتها. يُستخدم الترميز CAVLC للاستفادة من هذه الخصائص بغاية ضغط المعلومات [6].

يظهر في الشكل (15.3) مخطط مفكك الترميز الانتروبي CAVLC. ويعمل وفق التسلسل الآتي:

1. تفكيك ترميز عدد المعاملات غير الصفريّة Total_coeff وعدد المعاملات الواحديّة T1. يشير T1 إلى عدد المعاملات التي تأخذ القيم ± 1 . يمكن أن يأخذ المعامل Total_coeff أي قيمة بين 0-16، ويمكن أن يأخذ المعامل T1 أي قيمة بين 0-3. في حال وجود أكثر من ثلاث معاملات بقيم ± 1 أو -1، تؤخذ بالحسبان فقط آخر ثلاث قيم ضمن سلسلة الواحدات، وتُعدّ المعاملات الواحديّة الأخرى كمعاملات غير صفريّة.
2. تفكيك ترميز الإشارات ($\pm$) لكل معامل من المعاملات الواحديّة.
3. تفكيك ترميز قيم المعاملات غير الصفريّة وترتيبها عكسياً بطريقة zig-zag. تتألف كل قيمة من قسمين: لاحقة prefix، وبادئة suffix.
4. تفكيك ترميز العدد الكلي للأصفار Total_zeros الواقعة قبل آخر معامل غير صفري. يُستخدم من أجل تفكيك الترميز عدة جداول LUT. يجري اختيار أحد هذه الجداول اعتماداً على نتائج السلاسل المجاورة.
5. تفكيك ترميز عدد الأصفار Run_before التي تسبق كل معامل غير صفري



الشكل(15.3): مخطط مفكك الترميز الانتروبي [6]

تقوم الوحدة nC_decoding (الشكل 15.3) بالتنبؤ بعدد المعاملات غير الصفريّة nC اعتماداً على الكتل المجاورة. حيث يُعرّف العدد nC بأنه متوسط عدد المعاملات غير

الصفرية في الكتلتين الواقعتين على يسار وإلى أعلى (nA و nB على الترتيب) الكتلة المعنية. من العلاقة (1.3):

$$nC = \text{round}((nA+nB)/2) \quad (1.3)$$

يؤثر العدد nC على طريقة ترميز المعاملات Total_coeff و T1 بواسطة اختيار الجداول المناسبة: Table1, ..., Table4 (الجدول 2.3). يستخدم الجدول Table1 عندما يكون عدد المعاملات غير الصفرية Total_coeff صغيراً، وكلما زاد هذا العدد جرى الانتقال إلى الجدول الذي يليه.

nC	
0,1	Table 1
2,3	Table 2
4,5,6,7	Table 3
8 <=	Table 4

الجدول (2.3): اختيار جدول ترميز المعامل Total_coeff تبعاً لقيمة العدد nC [6]

يُبين المثال الآتي [6] طريقة الترميز وفك الترميز CAVLC، لتكن الكتلة 4×4 المراد ترميزها.

0	3	-1	0
0	-1	1	0
1	0	0	0
0	0	0	0

يمكن كتابة عناصر هذه الكتلة على شكل السلسلة بعد إعادة الترتيب (zig-zag):

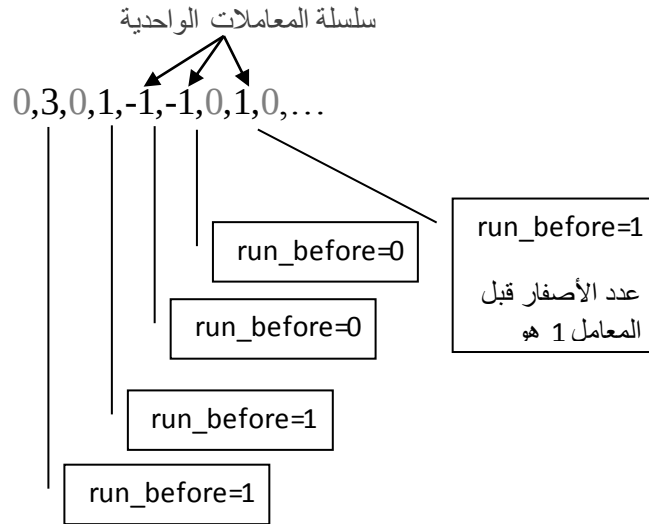
$$0,3,0,1,-1,-1,0,1,0,\dots$$

ويكون بذلك عدد المعاملات غير الصفرية Total_coeff=5 والعدد الكلي للأصفار الواقعة قبل آخر معامل غير صفري Total_zeros=3 وعدد المعاملات الواحدة T1=3 (عدد المعاملات الواحدة فعلياً يساوي 4، لكن الرمز T1 لا يمكن أن يتجاوز العدد 3). يُعد عدد المعاملات غير الصفرية في هذه الحالة صغيراً، لذلك يستخدم الجدول Table1 للترميز (الجدول 1.3).

يجري الترميز وفق الخطوات الآتية:

- ترميز الزوج (Total_coeff, T1) بحسب الجدول المناسب، وهو في هذه الحالة اعتماداً على الجدول (A.1) الرمز 0000100.

- ترميز إشارات المعاملات الواحدية: 0 للإشارة الموجبة، 1 للإشارة السالبة. فيجري الحصول في حالتنا على: 011.
- ترميز المعاملات غير الصفرية بالترتيب العكسي +1 ثم 3. يجري أولاً ترميز المعامل الأخير +1 باستخدام الجدول Level VLC 0 (الجدول 4.A) ثم ننقل إلى ترميز المعامل السابق 3 باستخدام الجدول Level VLC 1، فيجري الحصول على: 10010.
- ترميز العدد Total_zeros الذي يمثل العدد الكلي للأصفار الواقعة قبل آخر معامل غير صفري فيجري الحصول في حالتنا على 111 (الجدول 2.A).
- أخيراً، يجري ترميز العدد Run_before أي عدد الأصفار التي تسبق كل معامل غير صفري بالترتيب العكسي. يُبين الشكل (16.3) قيم العدد Run_before.



الشكل (16.3): مثال على القيم التي يأخذها العدد Run_before في المثال المذكور [6]

يجري ترميز هذا العدد بمرافقة عدد آخر هو Zero_left (الجدول 3.A) الذي يعبر عن عدد الأصفار المتبقية قبل كل معامل غير صفري كما هو موضح في الجدول (3.3)، وتنتج سلسلة البتات: 0000100 011 10010 111 101101.

الترميز	Zero_left, Run_before
10	Zero_left=3, Run_before=1
1	Zero_left=2, Run_before=0
1	Zero_left=2, Run_before=0
01	Zero_left=2, Run_before=1
-	Zero_left=1, Run_before=1

الجدول (3.3): ترميز عدد الأصفار التي تسبق كل معامل غير صفري

يُبين الجدول (4.3) مراحل فك الترميز. ثمة خط تحت القيمة المضافة في كل مرحلة من مراحل فك الترميز. في النهاية يصبح عدد الأصفار المضافة مساوياً 2 وبما أن عدد الأصفار هو $Total_zeros=3$ ، يضاف صفر إضافي لتصبح النتيجة: 0, 3, 0, 1, -1, -1, 0, 1.

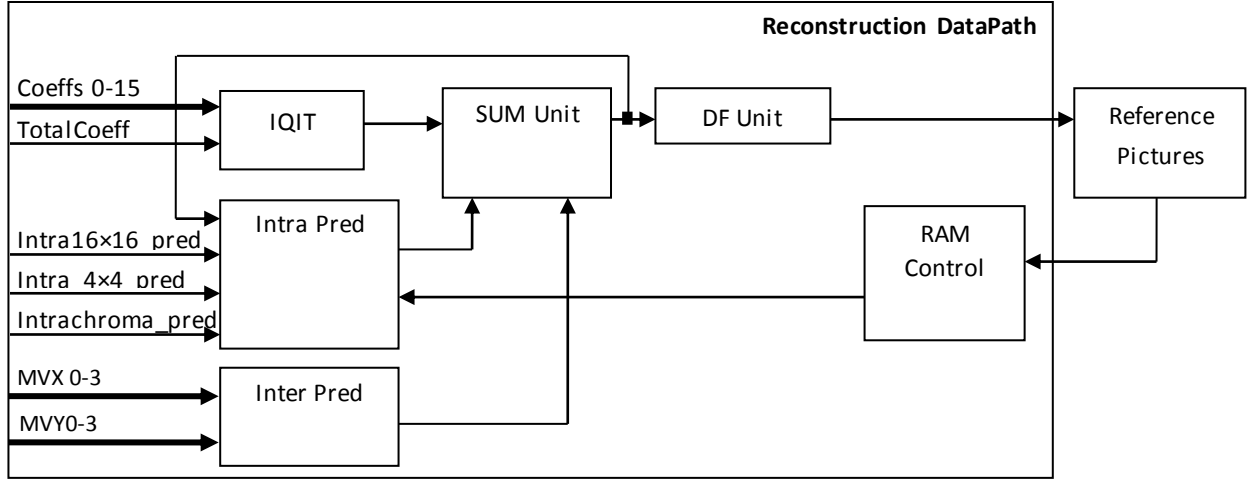
Code	Element	Value	Output array
0000100	coeff_token	TotalCoeffs = 5, T1 = 3	Empty
0	T1 sign	+	<u>1</u>
1	T1 sign	-	<u>-1</u> , 1
1	T1 sign	-	<u>-1</u> , -1, 1
1	Level	+1	<u>1</u> , -1, -1, 1
0010	Level	+3	<u>3</u> , 1, -1, -1, 0, 1
111	total_zeros	3	<u>3</u> , 1, -1, -1, 1
10	run_before	1	<u>3</u> , 1, -1, -1, <u>0</u> , 1
1	run_before	0	<u>3</u> , 1, -1, -1, 0, <u>1</u>
1	run_before	0	<u>3</u> , 1, -1, -1, 0, 1
01	run_before	1	<u>3</u> , <u>0</u> , 1, -1, -1, 0, 1

الجدول (4.3): مراحل فك الترميز CAVLC [6]

٣-٤ - وحدة إعادة بناء الفيديو reconstruction datapath

تقوم هذه الوحدة باستقبال معطيات الفيديو وإشارات التحكم من وحدة التحكم بعد تفكيك ترميزها الانتروبي. فيجري فيها تطبيق التكمية والتحويل العكسيين والاستفادة من التنبؤ الداخلي أو البيني من أجل توليد الفيديو النهائي. تضم معطيات الفيديو معاملات السلاسل $Coeff0, Coeff1, \dots, Coeff15$ الناتجة عن مفكك الترميز CAVLC. وتضم إشارات التحكم المعلومات الخاصة بأنماط التنبؤ الداخلي وأشعة الحركة من أجل التنبؤ البيني. وتتألف هذه الوحدة، كما هو موضح في الشكل (17.3) من الأجزاء الأساسية الآتية:

- وحدة التكمية والتحويل العكسيين IQIT.
- وحدة التنبؤ الداخلي Intra Pred.
- وحدة التنبؤ البيني Inter Pred.
- وحدة الجمع Sum Unit.
- وحدة مرشح التنعيم DF Unit.
- وحدة التحكم بالقراءة والكتابة على ذواكر الاطارات RAM Control.



الشكل (17.3): المخطط الصندوقي لوحدة إعادة بناء الفيديو [16]

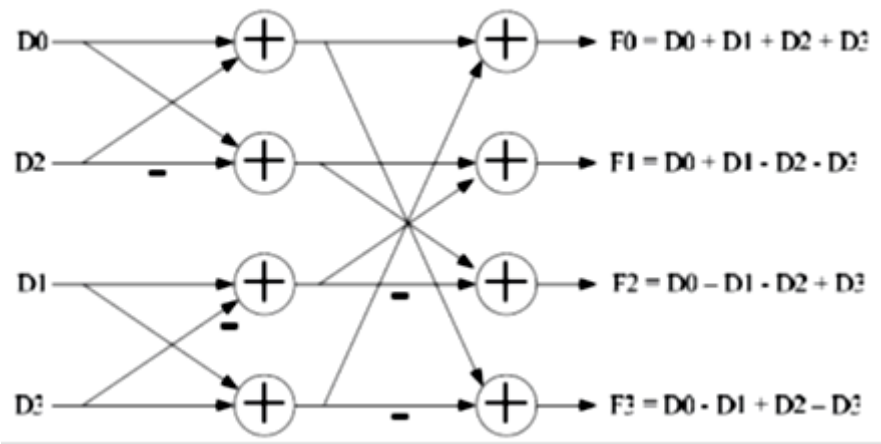
٣-٤-١ - وحدة التكمية والتحويل العكسيين IQIT

تعمل هذه الوحدة على إجراء التطبيق العكسي لعمليتي التكمية والتحويل. حيث يوجد ثلاثة أنواع من التحويلات:

- تحويل هادمارد 4×4 Hadamard العكسي ويطبق على كتلة بقياس 4×4 تضم 16 قيمة c_{ij} من المعاملات المستمرة للمركبات Y والناجمة عن التنبؤ الداخلي بحسب النمط $Intra_16 \times 16$. ويعطى بالعلاقة (2.3):

$$f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} c_{00} & c_{01} & c_{02} & c_{03} \\ c_{10} & c_{11} & c_{12} & c_{13} \\ c_{20} & c_{21} & c_{22} & c_{23} \\ c_{40} & c_{41} & c_{42} & c_{44} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \quad (2.3)$$

يظهر الشكل (18.3) تصميم الكيان الصلب لهذا التحويل العكسي بطريقة الفراشة Butterfly [17]. تُعدّ هذه الطريقة سريعة التنفيذ حيث تحتاج إلى مرحلتين للحصول على النتائج، إذ يجرى في المرحلة الأولى تمرير أعمدة المصفوفة C على بنية الفراشة ثم تمرير أسطر مصفوفة الخرج وإعادة إدخالها على بنية الفراشة.

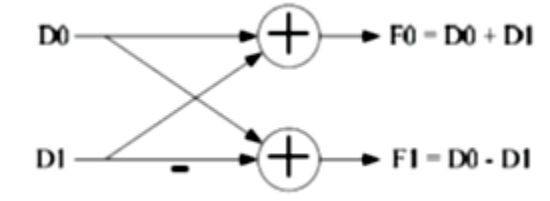


الشكل (18.3): تصميم الكيان الصلب لتحويل 4x4 Hadamard العكسي بطريقة الفراشة [17]

- تحويل هادامارد 2x2 Hadamard العكسي، ويطبق على كتلة بقياس 2x2 تضم 4 قيم c_{ij} من المعاملات المستمرة للمركبات Cr/Cb، ويعطى بالعلاقة (3.3):

$$f = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (3.3)$$

يُبين الشكل (19.3) تصميم الكيان الصلب لهذا التحويل العكسي بطريقة الفراشة.

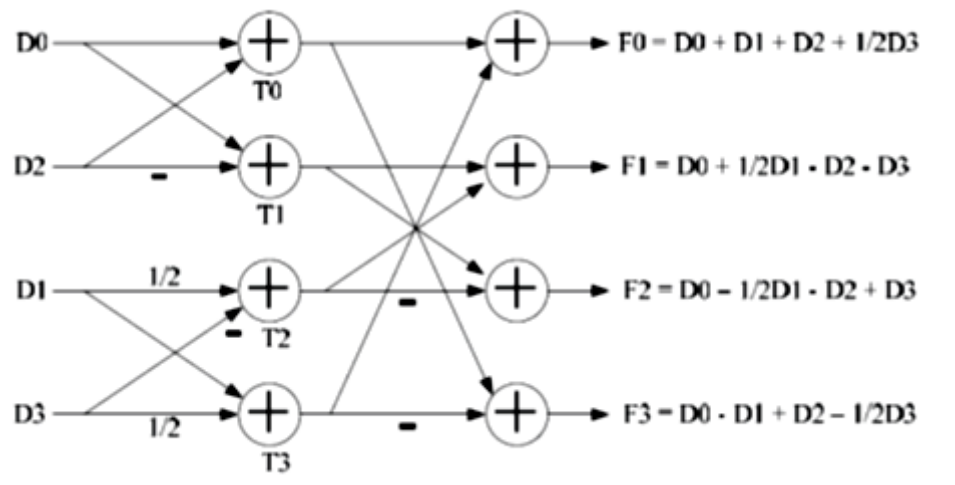


الشكل (19.3): تصميم الكيان الصلب لتحويل 2x2 Hadamard العكسي بطريقة الفراشة [17]

- تحويل DCT العكسي. ويطبق هذا التحويل على جميع الكتل المتبقية بقياس 4x4. ويعطى بالعلاقة (4.3):

$$f = \begin{bmatrix} 1 & 1 & 1 & 1/2 \\ 1 & 1/2 & -1 & -1 \\ 1 & -1/2 & -1 & 1 \\ 1 & -1 & 1 & -1/2 \end{bmatrix} \begin{bmatrix} c_{00} & c_{01} & c_{02} & c_{03} \\ c_{10} & c_{11} & c_{12} & c_{13} \\ c_{20} & c_{21} & c_{22} & c_{23} \\ c_{40} & c_{41} & c_{42} & c_{44} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1/2 & -1/2 & -1 \\ 1 & -1 & -1 & 1 \\ 1/2 & -1 & 1 & -1/2 \end{bmatrix} \quad (4.3)$$

يجري أيضاً تنفيذ هذا التحويل العكسي بطريقة الفراشة كما هو موضح بالشكل (20.3).

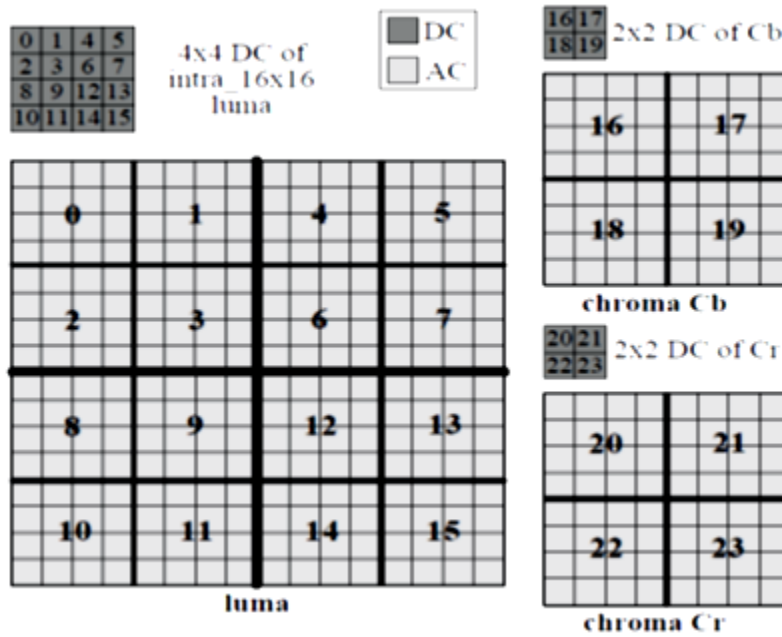


الشكل (20.3): تصميم الكيان الصلب لتحويل IDCT 4×4 بطريقة الفراشة [17]

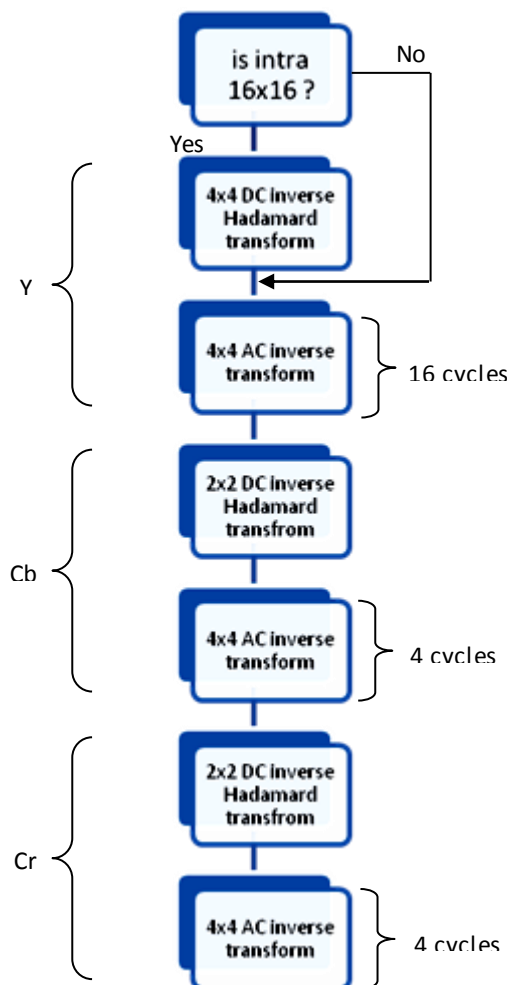
ويجري إرسال كتل المتبقي residuals وفق ترتيب معين كما هو مبين في الشكل (21.3):

- إرسال كتلة 4×4 تضم المعاملات المستمرة للكتلة 16×16 Intra.
- إرسال الكتل الجزئية 4×4 والمرفقة من 0 إلى 15 من الكتلة 16×16 Y.
- إرسال كتلة 2×2 تضم المعاملات المستمرة للكتلة 8×8 Cb.
- إرسال كتلة 2×2 تضم المعاملات المستمرة للكتلة 8×8 Cr.
- إرسال الكتل الجزئية 4×4 والمرفقة من 16 إلى 19 من الكتلة 8×8 Cb.
- إرسال الكتلة الجزئية 4×4 والمرفقة من 20 إلى 23 من الكتلة 8×8 Cr.

يظهر الشكل (22.3) خوارزمية IQIT وفقاً لترتيب إرسال كتل المتبقي residuals. نبدأً بفحص نمط التنبؤ، فإذا لم يكن النمط intra 16×16 يجري تطبيق IQIT على 16 كتل جزئية 4×4 من الكتلة 16×16 Y. وبطريقة مشابهة يجري تطبيق الخوارزمية على الكتل Cr/Cb. تجري عملية التكمية العكسي عادةً بعد عملية التحويل العكسي، وتقوم على إجراء عملية ضرب عناصر كل كتلة 4×4 بالمعامل Q_{step} الذي يجري استنتاجه من معامل التكمية QP.



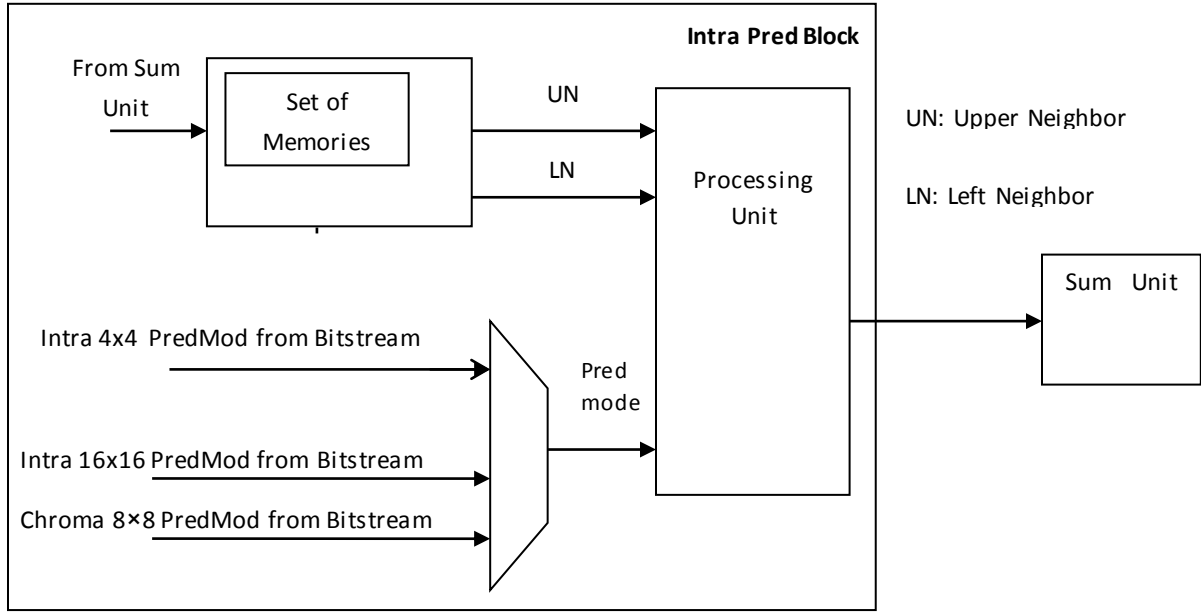
الشكل (21.3): إرسال كتل المتبقي [17]



الشكل (22.3): خوارزمية IQIT [17]

٣-٤-٢- وحدة التنبؤ الداخلي Intra Pred

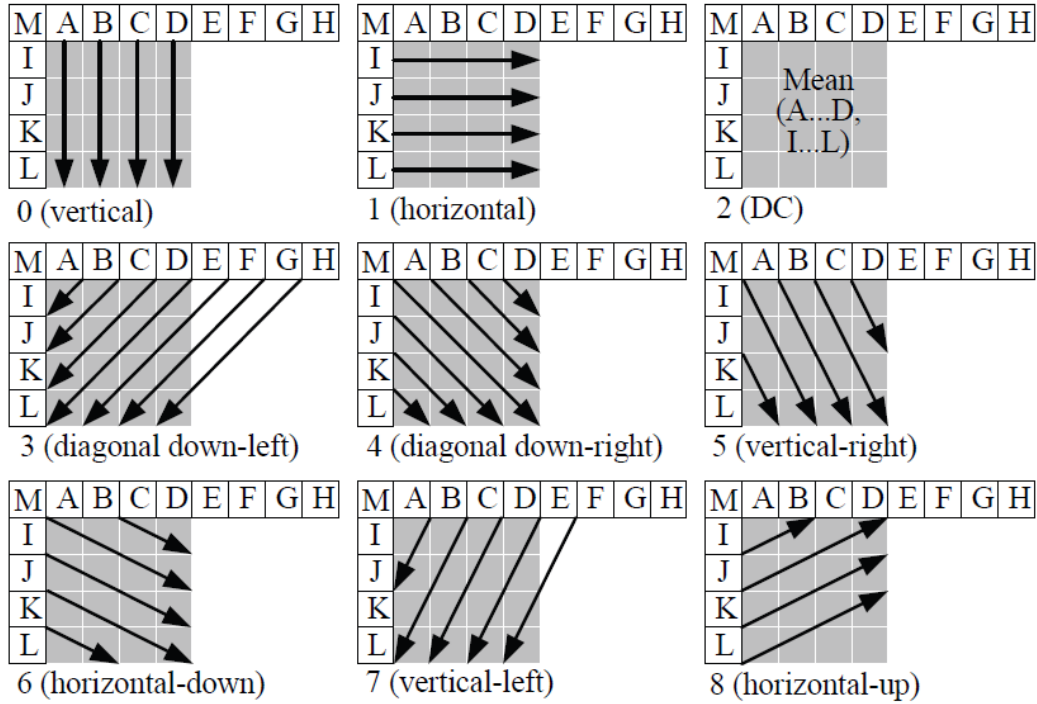
يجري التنبؤ الداخلي بكل كتلة ماكروية وفق عدة أنماط رئيسية: (Intra 16×16، Intra 4×4، Chroma 8×8). يضم النمط Intra 4×4 9 أنماط فرعية، بينما يضم كل من النمطين: (Intra 16×16 و Chroma 8×8) 4 أنماط فرعية. يقوم المرمز بإرسال النمط الذي جرى اعتماده لكل كتلة إلى مفك الضغط. يبين الشكل (23.3) المخطط الصندوقي لهذه الوحدة.



الشكل (23.3): المخطط الصندوقي لوحدة التنبؤ الداخلي [16]

يجري إرسال أنماط التنبؤ الداخلي (Intra 16×16, Chroma 8×8) من قبل المرسل. بينما تقوم وحدة مفك الترميز Intra 4×4 PredMode Decoder في حال اعتماد نمط التنبؤ الداخلي Intra 4×4 باستنتاج النمط الفرعي المعتمد بين الأنماط الفرعية التسعة كما هو موضح في الشكل (24.3).

وفي الشكل (24.3) تطابق أسطر الكتلة الماكروية في النمط 0 (vertical) البيكسلات العلوية المجاورة (UN) Upper Neighbour. وتطابق أعمدة الكتلة الماكروية في النمط 1 (horizontal) البيكسلات اليسارية المجاورة (LN) Left Neighbour. يجري اعتماد النمط الذي يحقق أفضل تقارب مع البيكسلات المجاورة. وتقوم وحدة المعالجة processing unit وفقاً للنمط المعتمد بحساب كتل التنبؤ والتي يجري إضافتها إلى كتل المتبقي residuals في وحدة الجمع Sum Unit.



الشكل (24.3): الأنماط الفرعية للتنبؤ الداخلي [16]

٣-٤-٣ - وحدة التنبؤ البيني Inter Pred

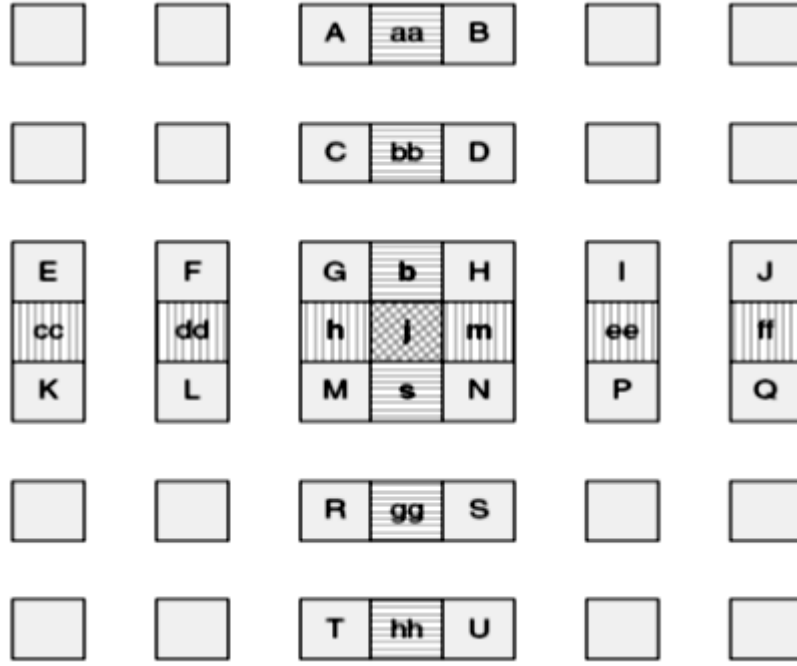
تعتمد عملية التنبؤ البيني على حساب أشعة الحركة بدقات مختلفة: 1 بيكسل أو 1/2 بيكسل أو 1/4 بيكسل.

عندما يكون شعاع الحركة بميز 1 بيكسل، فيكون بالإمكان القيام بالتنبؤ البيني لكنتلة معينة انطلاقاً من معرفة موقعها الأصلي في الإطار المرجعي. أما عندما يكون شعاع الحركة بميز 1/2 بيكسل أو 1/4 بيكسل، لا يكون بالإمكان إيجاد الموقع الأصلي للكنتلة في الإطار المرجعي لأن إحداثياته في هذه الحالة غير صحيحة، ويجب القيام باستقراء رياضي للحصول على الكنتلة في الموقع المطلوب.

من أجل الحصول على كنتلة إضاءة بقياس 4×4 وفقاً للميز 1/2 بيكسل، يجري في مرحلة أولى تطبيق 9 مرشحات أفقية 6-tap على كنتلة بقياس 9×9 للحصول على كنتلة بقياس 4×9. ثم يجري تطبيق 4 مرشحات عمودية 6-tap على الكنتلة الأخيرة للحصول على الكنتلة المطلوبة بقياس 4×4.

كل مرشح 6-tap هو عبارة عن مرشح ذو استجابة محدودة FIR معاملاته هي: (1,-5,20,20,-5,1). فمثلاً، للحصول على العينة b بميز 1/2 بيكسل، يُطبق المرشح على البيكسلات الصحيحة E, F, G, H, I, J (الشكل 25.3).

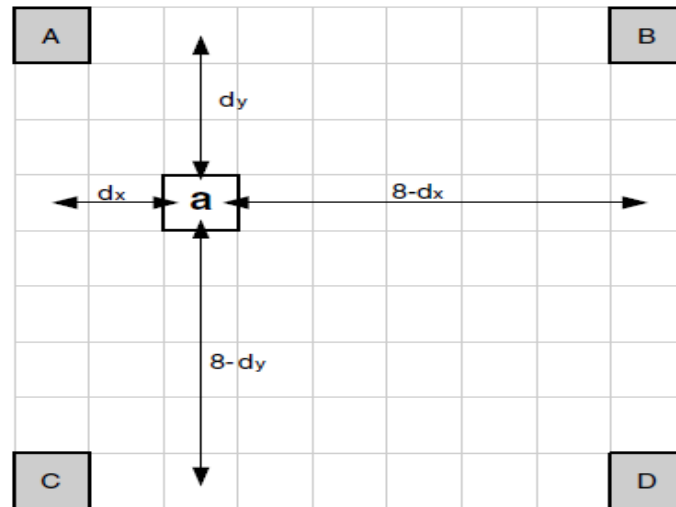
$$b = \text{round}((E - 5F + 20G + 20H - 5I + J) / 32)$$



الشكل(25.3): استقراء مركبات الإضاءة بميز 1/2 بيكسل [6]

ويمكن تطبيق مرشحات ثنائية الخطية bilinear من أجل الحصول على المركبات الضوئية بميز 1/4 بيكسل. عند استخدام نمط أخذ العينات 4:2:0، فإن استقراء المركبات الضوئية بميز 1/4 بيكسل يقتضي استقراء المركبات اللونية بميز 1/8 بيكسل. لذلك يجري استقراء المركبات اللونية Cr/Cb بميز 1/8 بيكسل بواسطة تطبيق مرشح على كتلة لونية بقياس 2×2، كما هو مبين في الشكل (26.3).

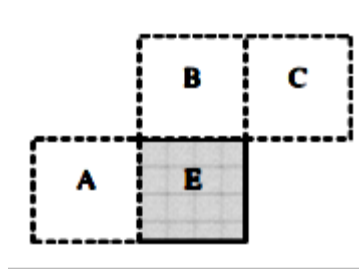
$$a = \text{round}([(8 - d_x) \cdot (8 - d_y)A + d_x \cdot (8 - d_y)B + (8 - d_x) \cdot d_y C + d_x \cdot d_y D]/64)$$



الشكل(26.3): استقراء المركبات اللونية بميز 1/8 بيكسل [6]

من أجل ضغط المعطيات يجري إرسال أشعة فرق الحركة mvd بدلاً من أشعة الحركة نفسها mv. فتقوم الوحدة Motion Vector Pred بتقدير شعاع الحركة mvpE للكتلة E اعتماداً على أشعة الحركة المرتبطة بالكتل المجاورة (A, B, C) كما هو مبين في الشكل (27.3). يجري تقدير شعاع الحركة عموماً كوسيط لأشعة الحركة الأخرى وفق العلاقة (5.3). يبين الشكل (28.3) المخطط الصندوقي لوحدة التنبؤ البيني. يجري استقراء المركبات الضوئية Y ضمن الكتلة Y Processing، بينما يجري استقراء المركبات اللونية ضمن الكتلة Cr/Cb Processing.

$$mvpE = \text{median}(mvpA, mvpB, mvpC) \quad (5.3)$$

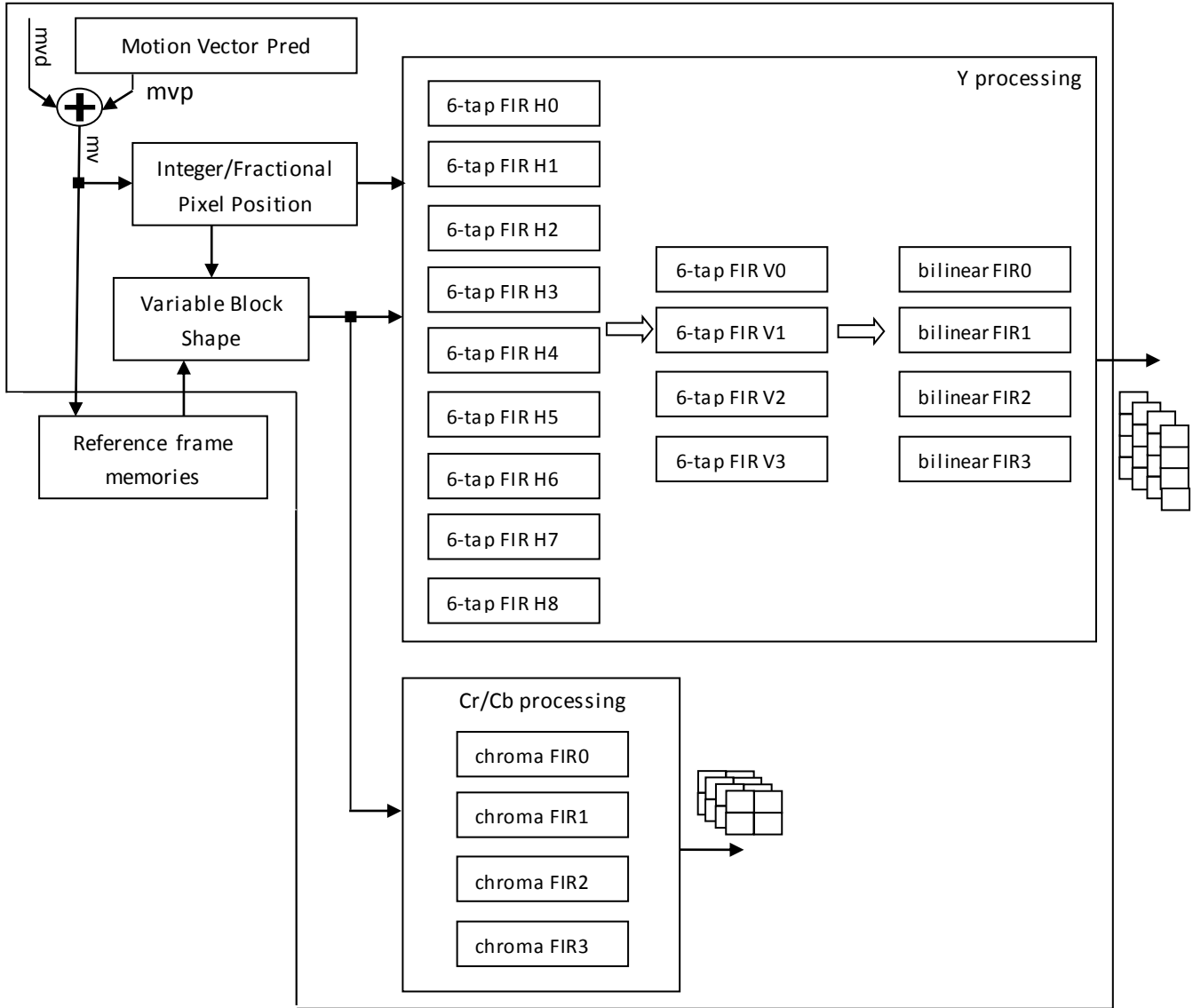


الشكل (27.3): تقدير شعاع الحركة [6]

ثم يجري حساب شعاع الحركة الحالي mv كمجموع لشعاع الحركة المقدر mvp وشعاع فرق الحركة mvd الذي جرى استقباله اعتماداً على العلاقة (6.3):

$$mv_E = mvp_E + mvd_E \quad (6.3)$$

ويجري اعتماداً على شعاع الحركة تحديد نوع الميز صحيح/كسري (integer/fractional) وقراءة الكتل الجزئية المناسبة من نواكر الاطارات وتخزينها في الوحدة Variable Block Shape.

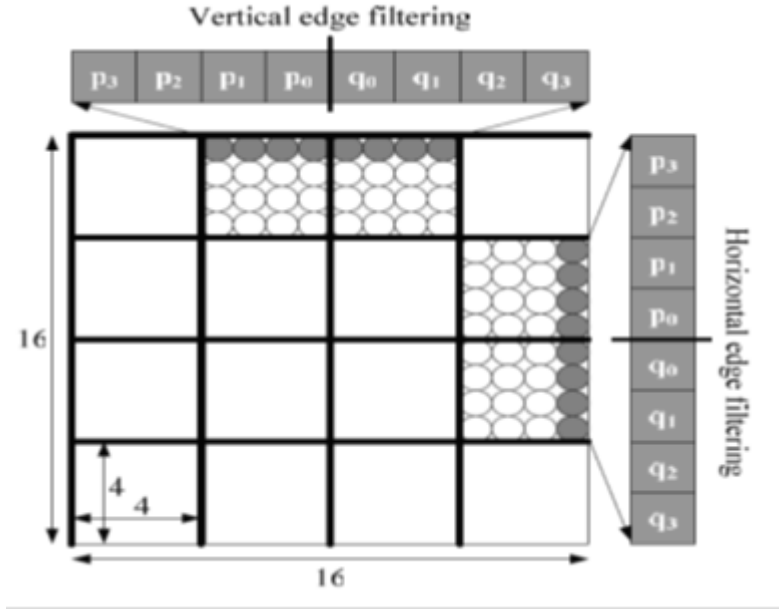


الشكل (28.3): المخطط الصندوقي لوحدة التنبؤ البيني [6]

٣-٤-٤ - وحدة مرشح التنعيم

يُستخدم مرشح التنعيم (DF) Deblocking filter من آثار الأخطاء الناجمة عن مرحلة التحويل DCT.

يعمل المرشح DF على مستوى الكتل الضوئية الماكروية 16×16 أو الكتل اللونية 8×8 . ويطبّق على الحواف العمودية ثم الحواف الأفقية. وتعتمد قوة الترشيح على معامل Bs يأخذ قيمة بين 0 و 4. القيمة 0 تعني عدم تطبيق المرشح والقيمة 4 تعني أعلى مستوى للترشيح. يحتاج الترشيح العمودي إلى أربع بيكسلات p0-p3 على يسار الحافة العمودية وأربع بيكسلات q0-q3 على يمينها كما هو مبين في الشكل (29.3). يجرى بنفس الطريقة تحديد البيكسلات المستخدمة في الترشيح الأفقي.



الشكل (29.3): تطبيق مرشح التنعيم على الحواف [17]

يظهر الجدول (5.3) مثالاً على كيفية حساب البيكسلات عند $B_s=4$. يأخذ هذا المعامل القيمة 4 عند تحقق الشرطين $|p_0-q_0|<\alpha$ و $|p_2-p_0|<\beta$. حيث تحدد المعاملات α و β من جداول خاصة.

$B_s=4$	Y	$p_0' = (p_2+2p_1+2p_0+2q_0+q_1+4)/8$ $p_1' = (p_2+p_1+p_0+q_0+2)/4$ $p_2' = (2p_3+3p_2+p_1+p_0+q_0+4)/8$	الشرط $ p_0-q_0 <\alpha$ $ p_2-p_0 <\beta$
		$q_0' = (q_2+2q_1+2q_0+2p_0+p_1+4)/8$ $q_1' = (q_2+q_1+q_0+p_0+2)/4$ $q_2' = (2q_3+3q_2+q_1+q_0+p_0+4)/8$	

الجدول (5.3): معاملات المرشح DF عند $B_s=4$ [17]

الفصل الرابع

دراسة خوارزمية التعمية

المتقدمة AES وتصميمها

AES Study And Design

٤-١ - مقدمة

تسمح عملية ضغط الفيديو بتقليص حجمه فيسهل تخزينه وإرساله. يبين الشكل (1.4) النظام الأساسي لضغط الفيديو وفك ضغطه. تقوم الكاميرا بالنقاط مقطع فيديو وإرساله إلى الضاغط الذي يقوم بضغط هذه الإشارة ثم إرسالها عبر إحدى قنوات الاتصال. في الطرف المقابل، يستقبل مفك الضغط الإشارة المضغوطة وفك ضغطها ليظهرها على شاشة عرض.



الشكل(1.4): نظام ضغط الفيديو و إرساله وفك ضغطه [38]

عندما يحتوي الفيديو على معلومات حساسة يصبح من الضروري الحفاظ على سلامتها وتأمينها من العبث والسرقة. يُطلق اسم التعمية على عملية حماية المعلومات القيّمة (المستندات، الصور، الفيديو،...) لمنع المتطفلين والمخربين واللصوص من الاطلاع عليها أو تغييرها أو سرقتها.

تُعرف التعمية Encryption بأنها عملية تحويل المعلومات إلى شيفرات غير مفهومة لمنع الأشخاص غير المرخص لهم من الاطلاع على المعلومات أو فهمها. وقد استخدم الإنسان التعمية منذ نحو ألفي عام قبل الميلاد لحماية رسائله السرية، وبلغ هذا الاستخدام ذروته في فترات الحروب؛ خوفاً من وقوع الرسائل الحساسة في أيدي العدو. وقام يوليوس قيصر بتطوير خوارزميته المعيارية المعروفة باسم شيفرة قيصر Caesar Cipher التي كانت نصاً معمى Cipher text لتأمين اتصالاته ومراسلاته مع قادة جيوشه.

وشكل الحاسوب في بدايات ظهوره وسيلةً جديدةً للاتصالات الآمنة، وفك تعمية رسائل العدو. ومنذ ذلك الحين، أخذت العديد من الشركات والمؤسسات تقوم بتطوير أنظمة تعمية جديدة، مما أبرز الحاجة إلى وجود معيار لعمليات التعمية.

ومن أبرز المؤسسات التي أسهمت في هذا المجال، المعهد الوطني للمعايير والتقانة (NIST) National Institute of Standards and Technology الذي قام في العام ١٩٧٣ بتطوير معيار تعمية المعطيات Data Encryption Standard (DES). يستند هذا المعيار إلى خوارزمية لوسيفر Lucifer وتشتت أن يكون لكل من المرسل والمستقبل المفتاح السري ذاته (بطول 64-bits). وقد اعتمدته البنوك لتشغيل آلات الصراف الآلي

Automated Teller Machine (ATM). ثم جاء المعيار AES كنسخة مطورة عن المعيار DES بزيادة طول الكلمة المشفرة وطول المفتاح (128-bits). وبعد عام واحد من تطبيق معيار تسمية المعطيات (DES)، طُوِّر ثلاثة أساتذة جامعيين نظامَ تسمية آخر أطلقوا عليه اسم (RSA)، ويستخدم هذا النظام زوجاً من المفاتيح (مفتاح عام ومفتاح خاص) عوضاً عن استخدام مفتاح واحد فقط. ومع أن هذا النظام كان ملائماً جداً لأجهزة الحاسوب المعقّدة، إلا إنه قد جرى اختراقه فيما بعد. وبقيت الحال على ذلك حتى قام فيل زيمرمان في العام 1986 بتطوير برنامج تسمية يعتمد نظام (RSA)، ولكنه يتميز باستخدام مفتاح بطول 128-bits ويُدعى برنامج الخصوصية المتفوّقة (Pretty Good Privacy (PGP). ويتوفر من هذا البرنامج نسخة تجارية ونسخة مجانية، وهو من أكثر برامج التسمية انتشاراً في وقتنا الحالي.

تقوم عملية التسمية (التشفير) بتحويل نص واضح plain text باستخدام مفتاح سري secret key إلى نص مُعَمَّى cipher text. ويوجد نوعان مختلفان للتسمية [٣٧]:

■ التسمية المتناظرة (وتسمى أيضاً بالمفتاح السري):

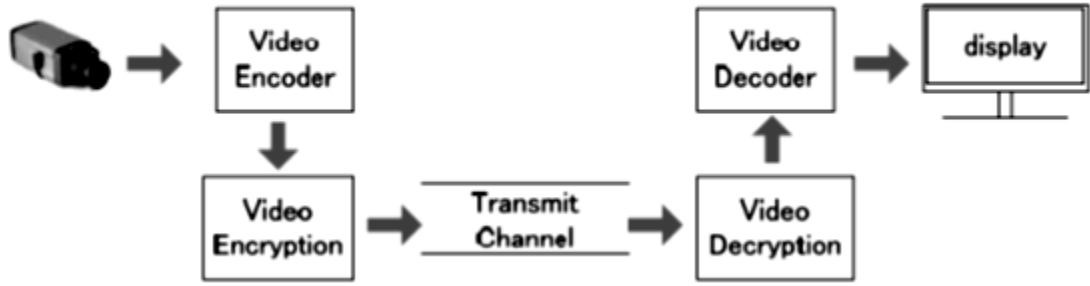
يستخدم في هذا النوع كل من المرسل والمستقبل المفتاح السري ذاته لتسمية الرسالة وفك تسميتها. فيتفق الطرفان على المفتاح السري نفسه وهنا يجب أن يجري تبادل المفتاح السري في بيئة آمنة. تتميز التسمية المتناظرة بصغر حجم المفتاح وسرعة التسمية وفك التسمية. ومن أهم خوارزمياتها: AES, DES, Triple DES.

■ التسمية غير المتناظرة (أو تسمية المفتاح العام):

جاءت التسمية غير المتناظرة حلاً لمشكلة التبادل غير الآمن للمفاتيح في الوصلات غير الآمنة. يُستخدم في هذا النوع من التسمية مفتاح عام public key ومفتاح خاص private key. يُستخدم الأول لتسمية الرسالة ويستخدم الثاني لفك تسمية الرسالة. يجري تبادل المفتاح العام ولا يجري تبادل المفتاح الخاص. تتميز التسمية غير المتناظرة بتبادل المفتاح في بيئة آمنة لكنها أبطأ من التسمية المتناظرة. ومن أهم خوارزمياتها: RSA, Diffie-Hellman, ElGamal, ECC.

تُستخدم التسمية غير المتناظرة عموماً لتسمية المفاتيح وكلمات السر والتوقيعات الرقمية، بينما تستخدم التسمية المتناظرة لتسمية الفيديو [٣٨]. وتعد خوارزمية AES من أشهر خوارزميات التسمية المتناظرة المستخدمة.

يبين الشكل (2.4) النظام الأساسي لتسمية الفيديو وفك تسميته. ويمكن تسمية أجزاء من إشارة الفيديو ونحدث هنا عن التسمية الانتقائية selective encryption أو تسمية الفيديو بكامله ويسمى التسمية الكاملة full encryption.



الشكل (2.4): نظام ضغط الفيديو وتعميته [38]

اعتمدنا في هذا البحث استخدام الخوارزمية المعيارية AES لتعمية الفيديو لما تتمتع به من درجة أمان عالية وسرعة وسهولة في التطبيق. كما اتبعنا التعمية الكاملة لمقطع الفيديو المضغوط وذلك لسهولة تطبيقها. سنأتي في هذا الفصل على شرح هذه الخوارزمية وكيفية تصميمها.

٤-٢- الخوارزمية Rijndael أو AES

خوارزمية Rijndael المعيارية standard هي خوارزمية كتلية block متناظرة symmetric، وطول مفتاحها السري 128-bits، 192-bits أو 256-bits. ويجري التعامل مع كتل بايتات عوضاً عن بتات، مما يُسرّع التعمية وفك التعمية [41][39][40]. يُطلق عادةً على خوارزمية Rijndael اسم "AES algorithm"، ويُشار عادةً إلى تغير طول مفتاح خوارزمية Rijndael بالرموز: "AES-128"، "AES-192" و "AES-256". سنقدّم فيما يأتي دراسة عن هذه الخوارزمية، تتضمن تعريفاً بأهم المصطلحات، والاختصارات، والتوابع المستخدمة فيها.

٤-٢-١ - المصطلحات والاختصارات

تُستخدم المصطلحات الآتية ضمن هذه الخوارزمية المعيارية:

- **AES**: معيار التعمية المتقدمة
- **Affine Transformation**: التحويل الأفيني هو تحويل يعتمد على ضرب بمصفوفة ثم جمع مع شعاع.
- **Block**: الكتلة هي تسلسل من البتات الثنائية لتشكيل الدخل والخرج.
- **Word**: الكلمة هي مجموعة من 32-bits وتُعامل كعنصر واحد أو كمصفوفة من أربع بايتات.

- **Cipher Key**: مفتاح التعمية هو كتلة من البايتات، يمكن تمثيلها بمصفوفة بايتات (bytes) مستطيلة بأربع أسطر، و Nk عمود.
- **Plaintext**: النص الواضح هو معطيات الدخل المراد تعميته.
- **Ciphertext**: النص المُعمى وهو خرج عملية التعمية، وهو بلا معنى إذا لم يجر فك تعميته باستخدام مفتاح التعمية (تعمية متناظرة).
- **Cipher**: التعمية هي سلسلة من التحويلات التي تطبق على النص الواضح فتحوله إلى نص مُعمى باستخدام مفتاح التعمية.
- **Inverse Cipher**: فك التعمية هي سلسلة من التحويلات التي تطبق على النص المُعمى فتحوله إلى نص واضح باستخدام مفتاح التعمية (في حالة التعمية المتناظرة).
- **Key Expansion**: توسيع المفتاح هو التابع الذي يولد سلسلة من مفاتيح الدورات Round Keys من مفتاح التعمية.
- **Round Keys**: مفاتيح الدورات هي كتل blocks مُشتقة من مفتاح التعمية باستخدام تابع توسيع المفتاح Key Expansion.
- **State**: الحالة هي نتيجة وسيطة للتعمية، يمكن تمثيلها بمصفوفة بايتات مستطيلة بأربع أسطر، و Nk عمود.
- **S-box**: هو مصفوفة تبديل غير خطية، تُستخدم في تابع توسيع المفتاح لاستبدال قيمة البايتات وهي تقابلية (واحد لواحد).

٤-٢-٢ - الرموز والتوابع

فيما يلي الرموز والتوابع المُستخدمة في خوارزمية معيار التعمية المتقدمة AES:

- **AddRoundKey()**: هو التابع الذي تجري فيه إضافة الحالة State إلى مفتاح الدورة Round Key باستخدام عملية XOR، ويُستخدم في التعمية وفك التعمية.
- **SubBytes()**: يقوم هذا التابع بالاستعاضة عن قيمة كل بايت من بايتات الحالة (State) بقيم موافقة في مصفوفة التبديل S-box.
- **InvSubBytes()**: يُعكس عمل هذا التابع عمل التابع SubBytes.
- **ShiftRows()**: يقوم هذا التابع بعملية إزاحة دورانية على بايتات كل سطر من أسطر الحالة State، بقيم إزاحة مختلفة لكل من هذه الأسطر.
- **InvShiftRows()**: يُعكس عمل هذا التابع عمل التابع ShiftRows.
- **MixColumns()**: يقوم هذا التابع بخلط بيانات كل عمود من أعمدة الحالة (State) بشكل مستقل عن بقية الأعمدة، لينتج أعمدة جديدة.

- **InvMixColumns()**: يُعكس عمل هذا التابع عمل التابع **MixColumns**.
- **Rcon []**: مصفوفة الكلمات الثابتة (كل كلمة مؤلفة من أربع بايتات).
- **K**: مفتاح التعمية وفك التعمية.
- **Nb**: هو عدد الكلمات المؤلفة من أربع بايتات 32-bits والموجودة في الحالة State، وقيمتها 4.
- **Nk**: هو عدد الكلمات المؤلفة من أربع بايتات 32-bits والموجودة في مفتاح التعمية، ويمكن أن يأخذ القيم 8 أو 6 أو 4.
- **Nr**: هو عدد الدورات التي ننفذها ضمن الخوارزمية AES، ويمكن أن يأخذ القيم 14 أو 12 أو 10 وهي توافق قيم Nk على الترتيب.

٤-٢-٣ - بعض الخواص الرياضية المستخدمة

تنتمي قيم كل البايتات bytes في الخوارزمية AES إلى حقل منتهٍ. حيث يمكن أن تطبق عليها عمليات جمع أو جداء بايتين، ولكن بشكل مختلف عن العمليات المألوفة. حيث يجري تمثيل كل بايت بكثير حدود، من الدرجة السابعة وأمثلة كثير الحدود هي 0 أو 1.

- عملية الجمع

يتحقق جمع عددين في حقل منتهٍ بإضافة أمثلة القوى المتوافقة في كثير الحدود (polynomials) الموافقين لهذين العددين. هنا نتعامل مع حقل غالوا Galois المنتهٍ $GF(2^8)$ والذي يضم 256 عنصراً، وكل عنصر ممثّل على 8-bits. وعملية الجمع المستخدمة هي العملية XOR (أي modulo 2)، حيث يرمز لهذه العملية بالرمز $\oplus$. أي لجمع بايتين $\{a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7\}$ و $\{b_0 b_1 b_2 b_3 b_4 b_5 b_6 b_7\}$ في حقل منتهٍ، يكون الناتج هو البايت $\{c_0 c_1 c_2 c_3 c_4 c_5 c_6 c_7\}$ ، حيث: $c_i = a_i \text{ xor } b_i$. يبين الجدول (1.4) ثلاث عبارات متكافئة في الحقل $GF(2^8)$.

$\{D6\} \text{ xor } \{6F\} = \{B9\}$	hexadecimal notation
$\{11010110\} \text{ xor } \{01101111\} = \{10111001\}$	binary notation
$(x^7+x^6+x^4+x^2+x) \text{ xor } (x^6+x^5+x^3+x^2+x+1) = (x^7+x^5+x^4+x^3+1)$	polynomial notation

الجدول (1.4) عبارات متكافئة في الحقل $GF(2^8)$ [43]

- عملية الضرب

يمثل جداء قيمة عددين في الحقل $GF(2^8)$ باقي قسمة جداء كثيري الحدود الموافقين على كثير حدود معياري غير قابل للاختزال (أي ليس له قواسم إلا نفسه والواحد) درجته 8، ويُعطى بالعلاقة (1.4).

$$m(x) = x^8 + x^4 + x^3 + x + 1 \quad (1.4)$$

ويمكن تمثيل كثير الحدود $m(x)$ على بايتين وفق العلاقة (2.4):

$$M = \{01\ 1B\}_{16} = \{0000\ 0001\ 0001\ 1011\}_2 \quad (2.4)$$

مثال: إن جداء الباييتين $\{A4\}_{16} \cdot \{45\}_{16}$ يساوي $\{C9\}_{16}$ إذ أن الجداء معطى بالعلاقة (3.4).

$$(x^6 + x^2 + 1) \cdot (x^7 + x^5 + x^2) = x^{13} + x^{11} + x^9 + x^8 + x^7 + x^5 + x^4 + x^2 \quad (3.4)$$

و يُعطى باقي قسمته على كثير الحدود المُعتمد بالعلاقة (4.4).

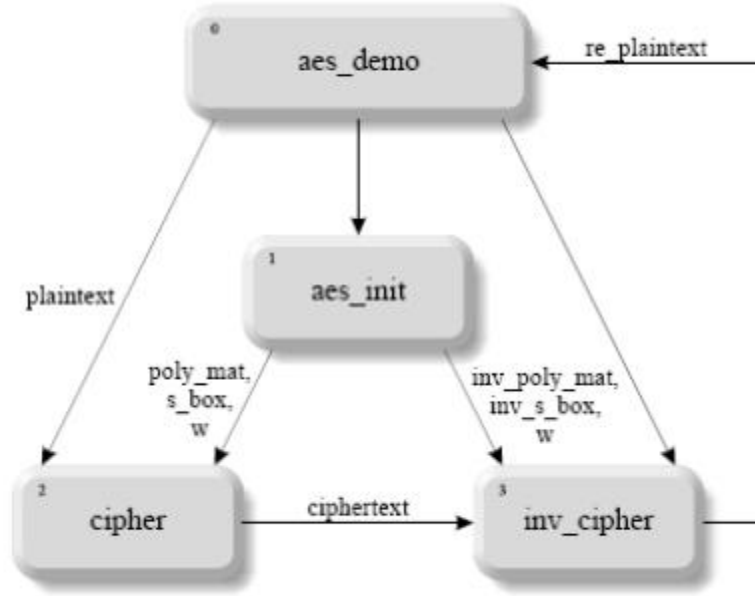
$$(x^{13} + x^{11} + x^9 + x^8 + x^7 + x^5 + x^4 + x^2) \text{ modulo } (x^8 + x^4 + x^3 + x + 1) \\ = x^7 + x^6 + x^3 + 1 = \{1100\ 1001\}_2 = \{C9\}_{16} \quad (4.4)$$

٤-٣- مراحل الخوارزمية

جرى تنفيذ عملية المحاكاة باستخدام برمجية ماتلاب MATLAB وفق المخطط الصندوقي المبين في الشكل (3.4) [٤٢]. ويتألف مخطط عملية المحاكاة، من الوحدات الآتية: `aes_demo`, `aes_init`, `cipher`, `inv_cipher`.

٤-٣-١- الوحدة `aes_demo`

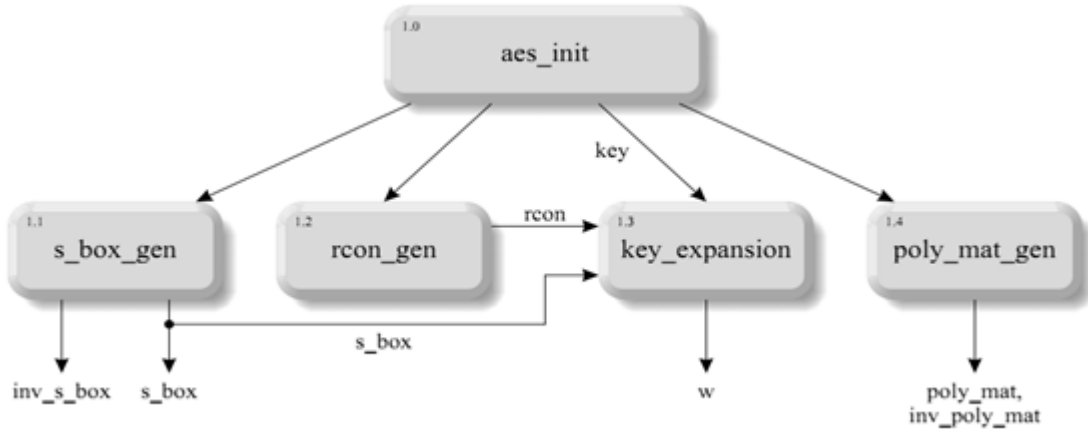
هي الوحدة الأساسية، وتحوي كل الكتل والتوابع التي نحتاجها في تنفيذ المحاكاة. وفيها تجرى قراءة النص الواضح `plaintext` ثم تعميته لنحصل على النص المُعمى `ciphertext` ثم فك تعميته لنحصل على النص `re_plaintext` الذي تجري مقارنته بالنص الأولي.



الشكل (3.4): المخطط الصندوقي لعملية محاكاة الخوارزمية AES ضمن بيئة Matlab [42]

٤-٣-٢ - الوحدة aes_init

قبل تنفيذ أي عملية تعمية أو فك تعمية، لابد من مرحلة ابتدائية، يجري بواسطتها تهيئة كل ما يلزم لعملية التعمية وفك التعمية، من ثوابت ومصفوفات. وتقوم الوحدة aes_init بهذه المرحلة الابتدائية. وتتألف هذه الوحدة من عدة توابع، موضحة في الشكل (4.4).

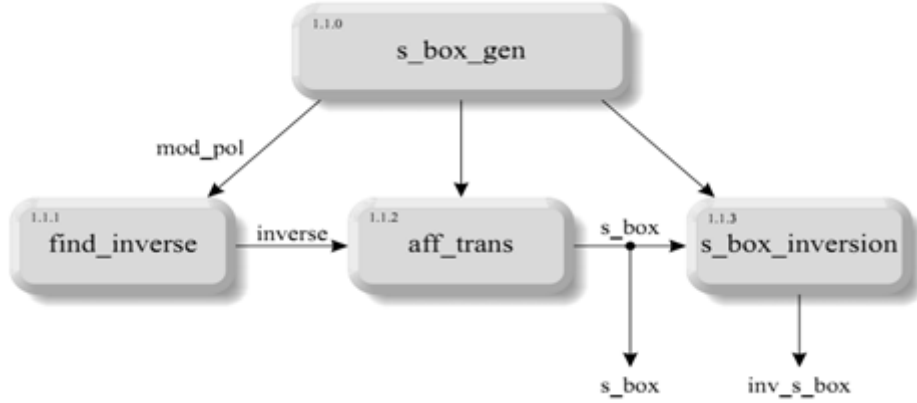


الشكل (4.4): المخطط الصندوقي لعمل وحدة aes_init [42]

- التابع s_box_gen

يولد هذا التابع جدول التبدل الثابت (s_box)، المستخدم في عملية التعمية، وكذلك جدول التبدل العكسي (inv_s_box)، المستخدم في عملية فك التعمية.

في التطبيقات العملية، تكون جداول التبدل الثابتة مُخزنة في أماكن ثابتة من الذاكرة، وذلك من أجل السرعة في طلبها، ولكن في هذه المحاكاة سيجري توليدها برمجياً وفق المخطط الموجود في الشكل (5.4).



الشكل (5.4) : مخطط توليد جداول التبدل [42]

أولاً نقوم بإدخال كثير الحدود modular reduction polynomial (mod_pol) المعروف في الخوارزمية AES. وهو كثير حدود مُعرّف في الحقل $GF(2^8)$ ، بالعلاقة (5.4):

$$m(x) = x^8 + x^4 + x^3 + x + 1 \quad (5.4)$$

ثم يطبق تحويل أفيني Affin transformations للحصول على جدول التبدل s_box. ثم يجري توليد جدول التبدل العكسي inv_s_box باستخدام جدول التبدل المباشر s_box. يظهر في الشكلين (6.4) و (7.4) جداول التبدل المباشر والعكسي.

```

s_box : 63 7c 77 7b f2 6b 6f c5 30 01 67 2b fe d7 ab 76
        ca 82 c9 7d fa 59 47 f0 ad d4 a2 af 9c a4 72 c0
        b7 fd 93 26 36 3f f7 cc 34 a5 e5 f1 71 d8 31 15
        04 c7 23 c3 18 96 05 9a 07 12 80 e2 eb 27 b2 75
        09 83 2c 1a 1b 6e 5a a0 52 3b d6 b3 29 e3 2f 84
        53 d1 00 ed 20 fc b1 5b 6a cb be 39 4a 4c 58 cf
        d0 ef aa fb 43 4d 33 85 45 f9 02 7f 50 3c 9f a8
        51 a3 40 8f 92 9d 38 f5 bc b6 da 21 10 ff f3 d2
        cd 0c 13 ec 5f 97 44 17 c4 a7 7e 3d 64 5d 19 73
        60 81 4f dc 22 2a 90 88 46 ee b8 14 de 5e 0b db
        e0 32 3a 0a 49 06 24 5c c2 d3 ac 62 91 95 e4 79
        e7 c8 37 6d 8d d5 4e a9 6c 56 f4 ea 65 7a ae 08
        ba 78 25 2e 1c a6 b4 c6 e8 dd 74 1f 4b bd 8b 8a
        70 3e b5 66 48 03 f6 0e 61 35 57 b9 86 c1 1d 9e
        e1 f8 98 11 69 d9 8e 94 9b 1e 87 e9 ce 55 28 df
        8c a1 89 0d bf e6 42 68 41 99 2d 0f b0 54 bb 16
    
```

الشكل (6.4): جدول التبدل المباشر s_box [42]

```
inv_s_box : 52 09 6a d5 30 36 a5 38 bf 40 a3 9e 81 f3 d7 fb
             7c e3 39 82 9b 2f ff 87 34 8e 43 44 c4 de e9 cb
             54 7b 94 32 a6 c2 23 3d ee 4c 95 0b 42 fa c3 4e
             08 2e a1 66 28 d9 24 b2 76 5b a2 49 6d 8b d1 25
             72 f8 f6 64 86 68 98 16 d4 a4 5c cc 5d 65 b6 92
             6c 70 48 50 fd ed b9 da 5e 15 46 57 a7 8d 9d 84
             90 d8 ab 00 8c bc d3 0a f7 e4 58 05 b8 b3 45 06
             d0 2c 1e 8f ca 3f 0f 02 c1 af bd 03 01 13 8a 6b
             3a 91 11 41 4f 67 dc ea 97 f2 cf ce f0 b4 e6 73
             96 ac 74 22 e7 ad 35 85 e2 f9 37 e8 1c 75 df 6e
             47 f1 1a 71 1d 29 c5 89 6f b7 62 0e aa 18 be 1b
             fc 56 3e 4b c6 d2 79 20 9a db c0 fe 78 cd 5a f4
             1f dd a8 33 88 07 c7 31 b1 12 10 59 27 80 ec 5f
             60 51 7f a9 19 b5 4a 0d 2d e5 7a 9f 93 c9 9c ef
             a0 e0 3b 4d ae 2a f5 b0 c8 eb bb 3c 83 53 99 61
             17 2b 04 7e ba 77 d6 26 e1 69 14 63 55 21 0c 7d
```

الشكل (7.4): جدول التبدل العكسي inv_s_box [42]

- التابع rcon_gen

يقوم هذا التابع بتوليد مصفوفة الدوران الثابتة (rcon)، المُستخدمة في عملية توليد المفاتيح الجزئية. وهي مصفوفة أبعادها 10×4 ، كلها أصفار ما عدا العمود الأول، الذي يتكون من قوى الأساس ٢.

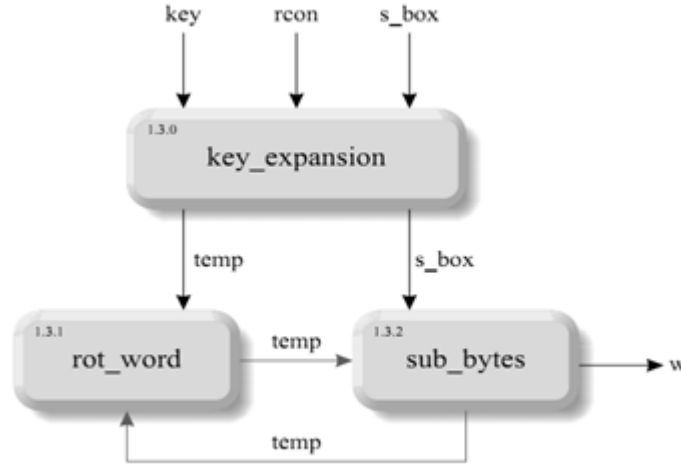
وكل قيمة تنتج عن القيمة السابقة، بضرب كثير الحدود الموافق لها بكثير الحدود الموافق للقيمة ٢، أي كثير الحدود (x). ونستخدم كثير الحدود modular reduction polynomial (mod_pol) المُعرّف في الخوارزمية AES، لحساب النتيجة. ويبين الشكل (8.4) مصفوفة الدوران الناتجة.

```
rcon : 01 00 00 00
        02 00 00 00
        04 00 00 00
        08 00 00 00
        10 00 00 00
        20 00 00 00
        40 00 00 00
        80 00 00 00
        1b 00 00 00
        36 00 00 00
```

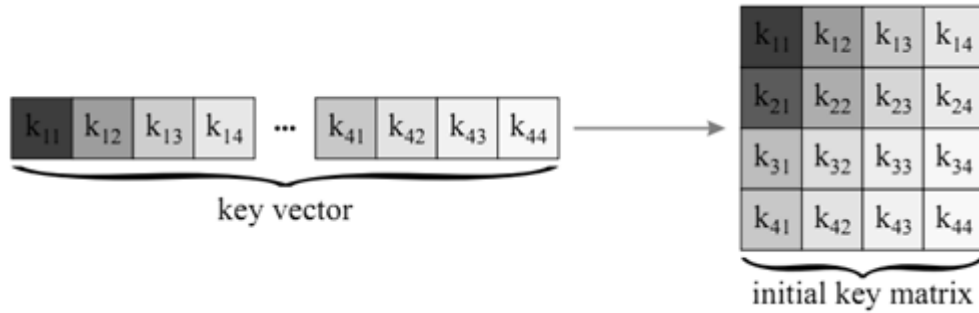
الشكل (8.4): مصفوفة الدوران الثابتة rcon [42]

- تابع توسيع المفتاح key_expansion

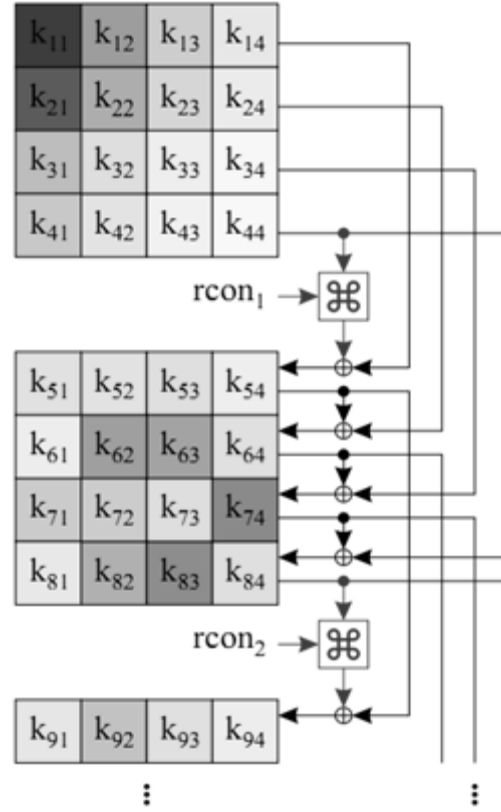
يستخدم هذا التابع المفتاح الذي يُدخله المستخدم، ومصفوفة الدوران الثابتة (rcon)، وجدول التبديل (s_box)، لتوليد جدول مفاتيح طويل w بطول 176-bytes. ويُستخدم في التعمية وفك التعمية. ويعمل تابع توسيع المفتاح وفق المخطط الموضح في الشكل (9.4). إذ يجري قبل ذلك تحويل مفتاح التعمية من شعاع إلى مصفوفة 4×4 ، كما هو موضح في الشكل (10.4).



الشكل (9.4): مخطط عمل تابع توسيع المفتاح [42]



الشكل (10.4): تحويل مفتاح التعمية من شعاع إلى مصفوفة 4×4 [42]



الشكل (11.4): توليد جدول المفاتيح w [42]

يُولد بعد ذلك جدول المفاتيح الطويل w ، كما هو مبين في الشكل (11.4). حيث ينتج السطر ذو الرقم i من إضافة السطر السابق $i-1$ مع السطر ذو الرقم $i-4$. ولكن في الأسطر ذات الأرقام $i=5,9,13,\dots$ أي الأسطر ذات الأرقام التي تحقق $i=4n+1$ ، نقوم بتدوير السطر السابق باستخدام التابع rot_word ، ثم استبدال قيم بايتاته باستخدام التابع sub_bytes ، ثم نضيف له السطر ذو الرقم i من مصفوفة $rcon$ ، ثم نضيفه للسطر ذو الرقم $i-4$.

- التابع $poly_mat_gen$

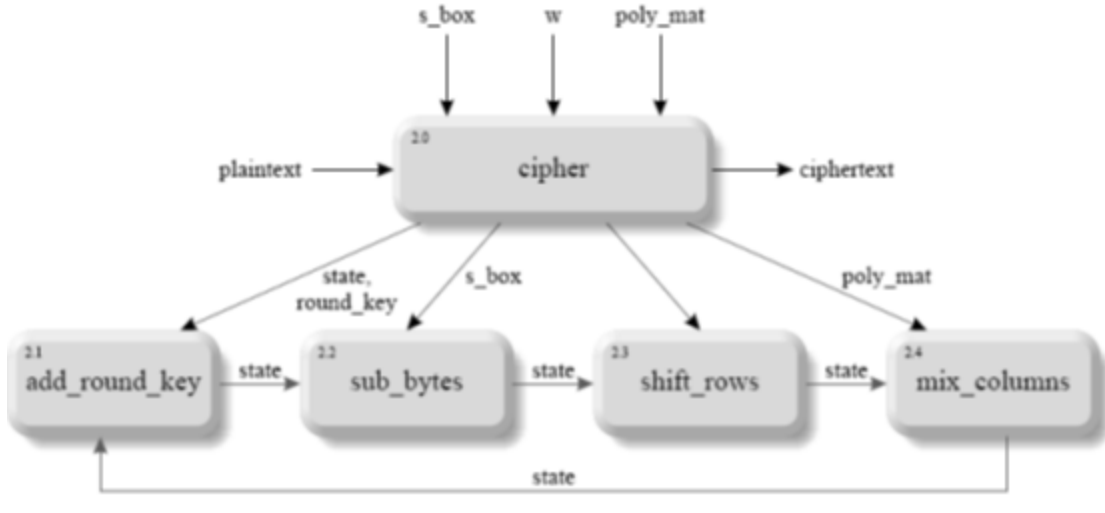
وهو تابع بدون دخل يُولد مصفوفتين ثابتتين، نسميهما $poly_mat$ و inv_poly_mat ، بُعد كل منهما 4×4 ، تُستخدمان في توابع مزج الأعمدة في عمليتي التعمية وفك التعمية بالترتيب. وتجرى قراءة مفتاح التعمية من الملف. ويُظهر الشكل (12.4) هاتين المصفوفتين.

```
poly_mat : 02 03 01 01      inv_poly_mat : 0e 0b 0d 09
           01 02 03 01      09 0e 0b 0d
           01 01 02 03      0d 09 0e 0b
           03 01 01 02      0b 0d 09 0e
```

الشكل (12.4): المصفوفتان $poly_mat$, inv_poly_mat [42]

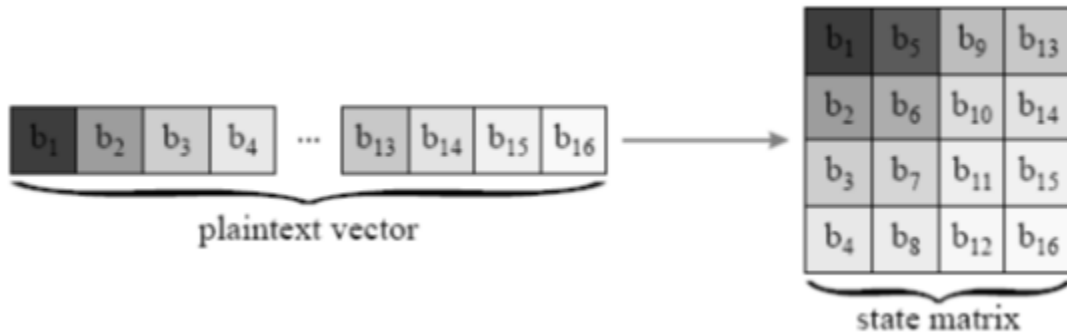
٤-٣-٣- الوحدة cipher

هذه الوحدة هي التابع الفعلي الذي يقوم بتعمية النص الواضح (16 bytes)، وتحويله إلى نص مُعمّى. دخل هذه الوحدة هو النص الواضح إضافةً إلى المتحولات والثوابت التي جرى إنشائها في الوحدة aes_init، وهي جدول التبدل s_box، جدول المفاتيح الطويل w، ومصفوفة كثير الحدود poly_mat. ولها خرج وحيد هو شعاع النص المعمّى، الذي له نفس طول النص الواضح أي 16-bytes. والشكل (13.4) يُظهر المخطط الصندوقي لعملية التعمية.



الشكل (13.4): المخطط الصندوقي لعملية التعمية [42]

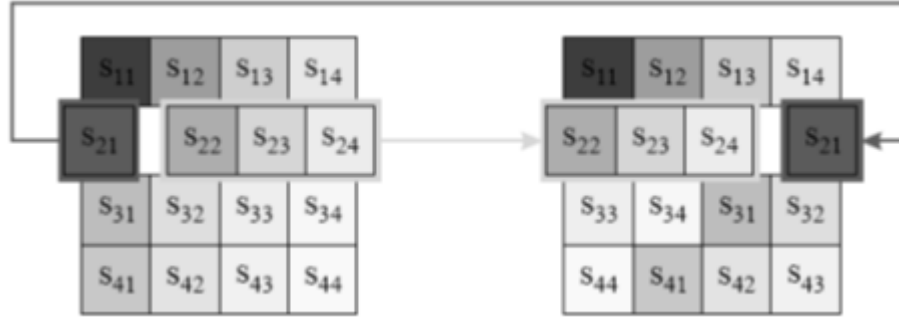
وتجرى عملية التعمية وفق المخطط الموضّح في الشكل (13.4)، وذلك وفق الخطوات الآتية: نرتب النص الواضح 16-bytes، في مصفوفة مربعة 4×4 ، ونسميها مصفوفة الحالة state (الشكل 14.4).



الشكل (14.4): مصفوفة الحالة [42]

نطبق أربع عمليات متتالية على مصفوفة الحالة بشكل حلقة كما هو مبين في الشكل (13.4)، وتكرر هذه الحلقة تسع مرات. هذه العمليات هي:

- **add_round_key**: جمع مصفوفة الحالة إلى مصفوفة المفتاح الجزئي round key، وهو عبارة عن أربع أسطر من مصفوفة المفاتيح الطويلة w .
- **sub_bytes**: تبديل قيم البايتات في مصفوفة الحالة، باستخدام جدول التبديل s_box .
- **shift_rows**: إزاحة أسطر مصفوفة الحالة دورانياً نحو اليسار، (الشكل 15.4).

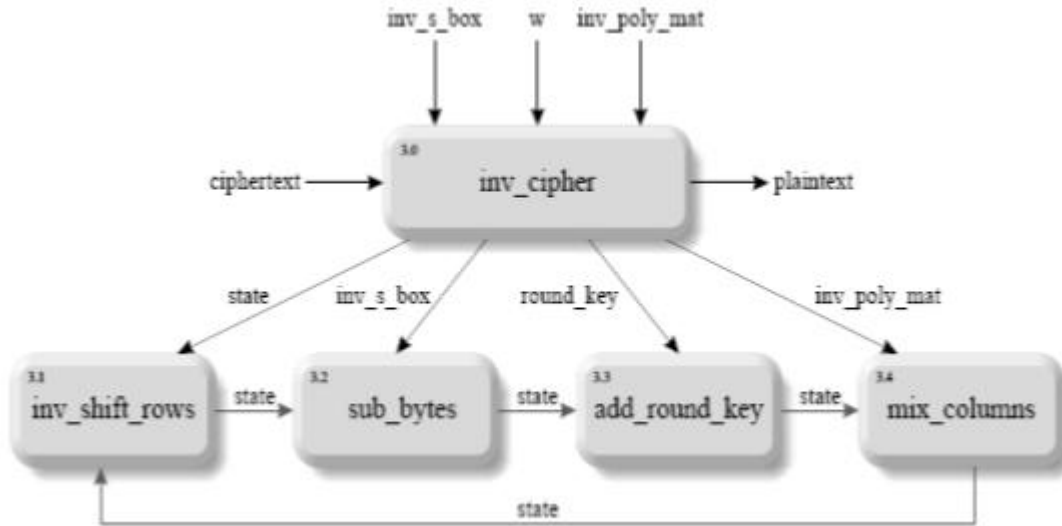


الشكل (15.4): إزاحة أسطر مصفوفة الحالة باستخدام تابع الـ **shift_rows** [42]

- **mix_columns**: تنتج مصفوفة الحالة الجديدة، من جداء مصفوفة كثير الحدود P الذي جرى تعريفه بوساطة التابع **poly_mat_gen** بمصفوفة الحالة الحالية. ويكون الخرج النهائي هو النص المعمّى.

٤-٣-٤ - الوحدة **inv_cipher**

وحدة فك التعمية **inv_cipher** هي عملية معاكسة تماماً لوحدة التعمية **cipher**. تقوم هذه الوحدة بفك تعمية النص المعمّى 16-bytes، وتحويله إلى نص واضح. دخل هذه الوحدة هو النص المعمّى إضافة إلى متحولات وثوابت جرى إنشائها في الوحدة **aes_init**، وهي جدول التبديل العكسي **s_box**، جدول المفاتيح الطويل w ، ومصفوفة كثير الحدود **inv_poly_mat**. ولها خرج وحيد هو شعاع النص الواضح، الذي له نفس طول النص الواضح أي 16-bytes. يُظهر الشكل (15.4) المخطط الصندوقي لعملية فك التعمية. وتجري عملية فك التعمية وفق طريقة معاكسة لعملية التعمية وتكون مصفوفة الحالة النهائية هي النص الواضح المطلوب استخراجه.



الشكل (15.4): المخطط الصندوقي لعملية فك التعمية [42]

٤-٤ - تصميم الخوارزمية AES

قمنا باستخدام لغة توصيف الكيان الصلب (VHDL) Very high speed integrated circuits Hardware Description Language، لبناء خوارزمية التعمية المعيارية AES أي لبناء وحدتي التعمية وفك التعمية. واجهتنا في هذه المرحلة بعض المشاكل التصميمية وقمنا بتقديم الحلول المناسبة لها. جرى استخدام الذاكر ROM لتخزين الثوابت كجداول التبدل المباشرة والعكسية ومصفوفة الدوران rcon ومصفوفتي المزج poly_mat و inv_poly_mat.

أما بالنسبة للمفاتيح الجزئية فيجب تنفيذ مرحلة توسيع المفتاح لتوليدها قبل البدء بتنفيذ التعمية أو فك التعمية، بحيث يجري استهلاكها في كل مرحلة من مراحل التعمية أو فك التعمية. تقوم وحدة التعمية باستخدام المفاتيح الجزئية بالترتيب المباشر، بينما تقوم وحدة فك التعمية باستخدامها بالترتيب العكسي. ومن ثمَّ لا تحتاج وحدة التعمية إلى تخزين المفاتيح الجزئية لأنها تستخدم كل مفتاح جزئي فور توليده. أما بالنسبة لوحدة فك التعمية فعليها الانتظار حتى توليد جميع المفاتيح الجزئية لأنها تستخدمها على نحو عكسي. في حال تخزين المفاتيح الجزئية في ذاكرة، تنتظر وحدة فك التعمية مرة واحدة لتوليد جميع المفاتيح الجزئية ثم تجري قراءتها من الذاكرة في المرات الأخرى (عند كل عملية فك تعمية). يجب الانتظار عند كل استخدام لمفتاح سري جديد.

تقتصر العمليات الحسابية في الخوارزمية على عمليات الجمع والضرب في الحقل $GF(2^8)$. تبني عملية الجمع باستخدام العملية xor. تستخدم عملية الضرب عند تطبيق مزج الأعمدة،

حيث تجري عملية ضرب مصفوفة الحالة 4×4 بإحدى المصفوفتين الثابتتين poly_mat و inv_poly_mat المبينتين بالعلاقة (6.4).

$$\begin{aligned} \text{poly_mat} : & \begin{matrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{matrix} \\ \text{inv_poly_mat} : & \begin{matrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{matrix} \end{aligned} \quad (6.4)$$

إن جداء كل عنصر من المصفوفة الأولى بعنصر من الثانية هو جداء كثيري حدود. تنتمي كثيرات الحدود في المصفوفتين الثابتتين إلى المجموعة: {01, 02, 03, 09, 0B, 0D, 0E}. و من ثَمَّ يمكن تبسيط عملية الضرب إلى الضرب بكثير حدود ثابت. يمكن تحقيق هذه العملية بعدة طرائق [٤٣]:

- استخدام جداول LUT لكل عنصر من المجموعة السابقة. حيث تحتفظ هذه الجداول بجميع احتمالات الضرب الممكنة. تتمثل نقطة ضعف هذه الطريقة في أنها تحتاج إلى ذواكر كبيرة الحجم.
- حساب الجداء رياضياً لكل عنصر من المجموعة السابقة وفق العلاقات (7.4) إلى (12.4):

$$\{0D\} * M(x) = (m0 + m5 + m6) + (m1 + m5 + m7) x + (m0 + m2 + m6) x^2 + (m0 + m1 + m3 + m5 + m6 + m7) x^3 + (m1 + m2 + m4 + m5 + m7) x^4 + (m2 + m3 + m5 + m6) x^5 + (m3 + m4 + m6 + m7) x^6 + (m4 + m5 + m7) x^7. \quad (7.4)$$

$$\{0E\} * M(x) = (m5 + m6 + m7) + (m0 + m5) x + (m0 + m1 + m6) x^2 + (m0 + m1 + m2 + m5 + m6) x^3 + (m1 + m2 + m3 + m5) x^4 + (m2 + m3 + m4 + m6) x^5 + (m3 + m4 + m5 + m7) x^6 + (m4 + m5 + m6) x^7. \quad (8.4)$$

$$\{0B\} * M(x) = (m0 + m5 + m7) + (m0 + m1 + m5 + m6 + m7) x + (m1 + m2 + m6 + m7) x^2 + (m0 + m2 + m3 + m5) x^3 + (m1 + m3 + m4 + m5 + m6 + m7) x^4 + (m2 + m4 + m5 + m6 + m7) x^5 + (m3 + m5 + m6 + m7) x^6 + (m4 + m6 + m7) x^7. \quad (9.4)$$

$$\{09\} * M(x) = (m0 + m5) + (m1 + m5 + m6) x + (m2 + m6 + m7) x^2 + (m0 + m3 + m5 + m7) x^3 + (m1 + m4 + m5 + m6) x^4 + (m2 + m5 + m6 + m7) x^5 + (m3 + m6 + m7) x^6 + (m4 + m7) x^7. \quad (10.4)$$

$$\{03\} * M(x) = (m0 + m7) + (m0 + m1 + m7) x + (m1 + m2) x^2 + (m2 + m3 + m7) x^3 + (m3 + m4 + m7) x^4 + (m4 + m5) x^5 + (m5 + m6) x^6 + (m6 + m7) x^7. \quad (11.4)$$

$$\{01\} * M(x) = M(x). \quad (12.4)$$

لا تحتاج هذه الطريقة إلى ذواكر لكن يمكن تبسيطها بالاعتماد على خواص الحقل $GF(2^8)$. تعتمد الطريقة الأخيرة على أن الضرب بكثير الحدود $\{02\}_{16}$ في الحقل $GF(2^8)$ يكافئ إزاحة الضارب مرة واحدة إلى اليسار مع إضافة كثير الحدود $\{1B\}_{16}$ [44] كما هو مبين بالعلاقة (13.4).

$$\{02\} * M(x) = (m6.x^7 + m5.x^6 + m4.x^5 + m3.x^4 + m2.x^3 + m1.x^2 + m0.x) \text{ xor } \{1B\}_{16} \quad (13.4)$$

ويتحول الجداء بكثير الحدود $\{03\}_{16} = \{02\}_{16} + \{01\}_{16}$ إلى عملية جمع إضافة إلى إزاحة لمرة واحدة. كما توضحه الأمثلة الثلاث (14.4) و (15.4) و (16.4):

$$\{03\}_{16} * M(x) = (\{02\}_{16} + \{01\}_{16}) * M(x) = \{02\} * M(x) \text{ xor } M(x) \quad (14.4)$$

$$\{09\}_{16} * M(x) = (\{08\}_{16} + \{01\}_{16}) * M(x) = \{08\} * M(x) \text{ xor } M(x) \quad (15.4)$$

$$\{0B\}_{16} * M(x) = (\{08\}_{16} + \{02\}_{16} + \{01\}_{16}) * M(x) = \{08\} * M(x) \text{ xor } \{02\} * M(x) \text{ xor } M(x) \quad (16.4)$$

وقد اتبعنا الطريقة الأخيرة في بناء عملية الضرب لسهولة تنفيذها. لقد عرضنا في هذه الفقرة مشاكل تصميم الكيان الصلب لهذه الخوارزمية والحلول المقدمة على نحو موجز.

الفصل الخامس:

التففيذ العملي

The Practical Execution

٥-١- مقدمة

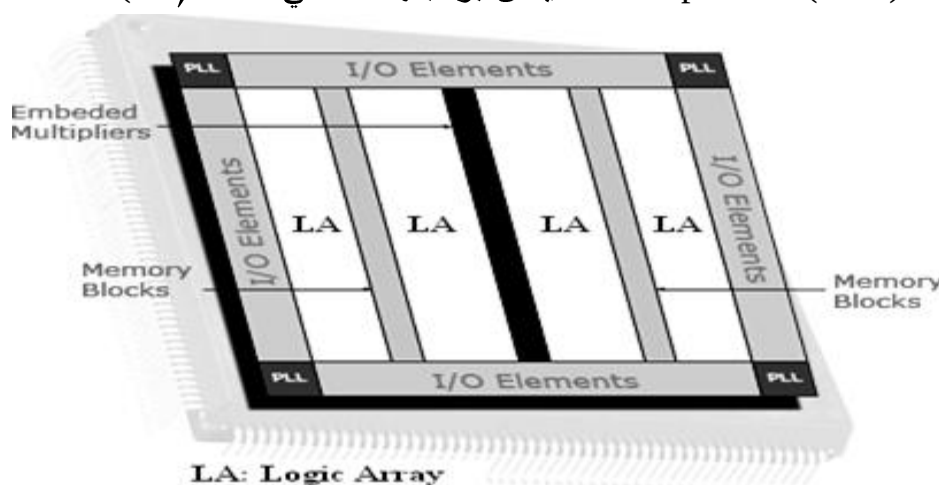
يهدف هذا البحث إلى بناء الكيان الصلب لنظام تعمية، فك تعمية وفك ضغط فيديو حسب معياري AES و H.264 على الترتيب. يعرض هذا الفصل التنفيذ العملي لهذا المشروع، و أدواته ومراحله. يهدف التنفيذ العملي إلى تطبيق الدراسة النظرية والتأكد من صحة عمل الخوارزميات إضافةً إلى حساب تعقيد هذه الخوارزميات وسرعة عملها. تُعدّ مرحلة التنفيذ العملي من المراحل الأساسية والصعبة للوصول إلى المنتج النهائي. عادةً ما تقتصر المشاريع على مرحلة الدراسة النظرية وتطوير الخوارزميات دون الوصول إلى منتج النهائي.

أصبح بالإمكان اليوم الوصول إلى هذه المرحلة بفضل توفر بطاقات التطوير بإمكانياتها العالية وميزاتها المتعددة كبطاقات تطوير FPGA Development Kits. قمنا في مشروعنا باستخدام بطاقة DE2_115 التي تحتوي على شريحة FPGA من نوع CYCLONE IV من شركة ALTERA.

يبدأ هذا الفصل بشرح المنهجية العامة للتصميم باستخدام تقانة FPGA ثم يقدم البطاقة DE2_115 والكاميرا المستخدمتان في هذا المشروع. وينتهي بتقديم المراحل والخطوات المتبعة لتنفيذ المشروع على هذه البطاقة بوساطة أربع تطبيقات.

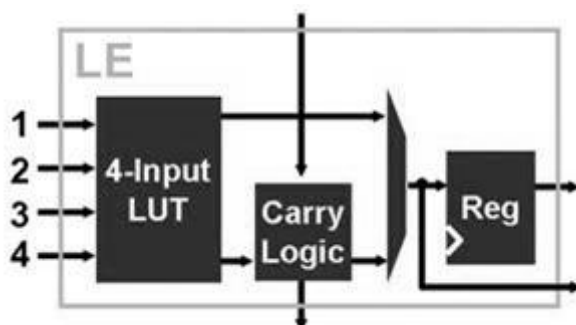
٥-٢- المنهجية العامة للتصميم باستخدام تقانة FPGA

تتألف شريحة FPGA من مصفوفة كبيرة من العناصر القابلة للبرمجة والتوصيل. يمكن لهذه العناصر أن تكون عناصر منطقية، عناصر حسابية، ذواكر أو حتى معالجات صغيرة بأكملها (الشكل 1.5). فمثلاً تستخدم شركة ALTERA العنصر المنطقي (Logic Element (LE) كعنصر أساسي لبناء مصفوفة العناصر [45]. يحتوي كل عنصر منطقي على سجل register وجدول حقيقة (Look Up Table (LUT يمكن برمجتهما كما في الشكل (2.5).



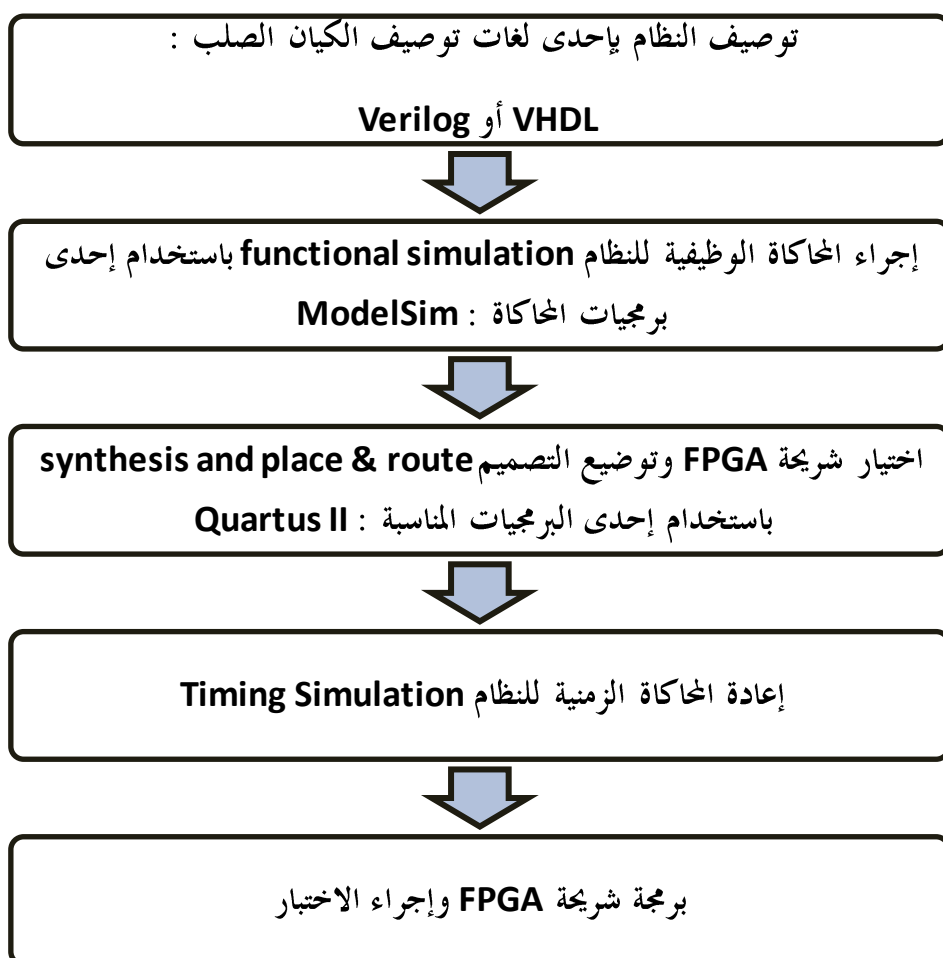
الشكل (1.5): العناصر التي تتألف منها شريحة FPGA [45]

تستخدم جداول الحقيقة LUT عادةً في حالات بناء دارات المنطق التركيبي Combinatorial Logic، كما يمكن استخدامها كنواكر أو سجلات إزاحة.



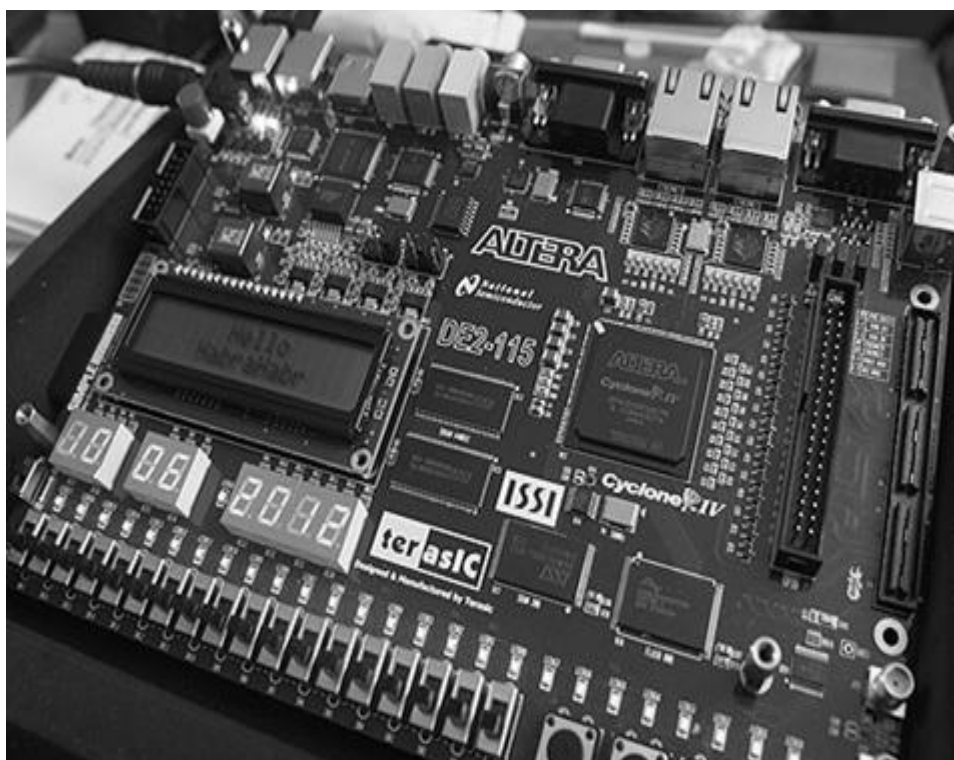
الشكل (2.5): العنصر المنطقي LE [45]

تتألف منهجية التصميم باستخدام تقانة FPGA من عدة مراحل كما هو مبين في الشكل (3.5). تسمح اللغتان VHDL أو Verilog بتوصيف أي نظام رقمي بواسطة كتابة code [46]. يمكن من بعدها محاكاة هذا التوصيف وظيفياً باستخدام إحدى برمجيات المحاكاة. تُعدّ المحاكاة ضرورة حتمية للتأكد من عمل النظام ولا يمكن الاستغناء عنها، خاصةً عندما يزداد النظام تعقيداً. وتعدّ الأداة ModelSim من أهم برمجيات المحاكاة.



الشكل (3.5): منهجية التصميم باستخدام تقانة FPGA [46]

بعد محاكاة النظام يمكن تحويله من هيئة شيفرة code إلى لائحة بوابات منطقية Netlist [46]. يطلق على هذه العملية تسمية: التركيب synthesis. بعد اختيار شريحة FPGA يمكن إجراء توزيع التصميم على هذه الشريحة place & route. ويمكن إتمام عمليات التركيب والتوزيع باستخدام برمجيات خاصة بكل شركة FPGA. فمثلاً برمجيات شركة Xilinx هي ISE وبرمجيات شركة Altera هي Quartus II [47]. يمكن إعادة المحاكاة بإضافة الأزمنة الحقيقية والمتناسبة مع شريحة FPGA التي جرى اختيارها. في النهاية تجري برمجة الشريحة ويجري اختبار عمل النظام.



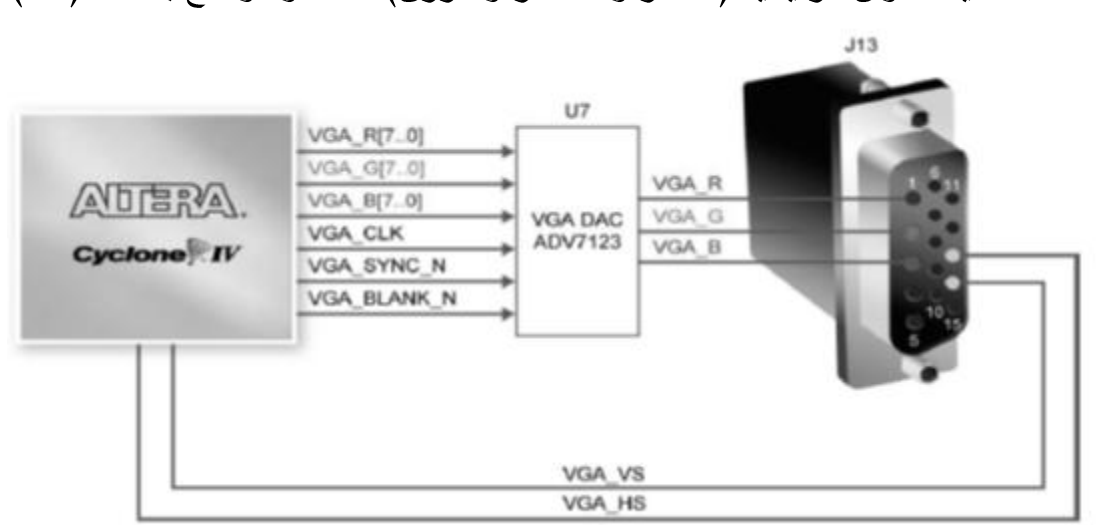
الشكل (4.5): بطاقة التطوير DE2_115 [48]

نفذَ هذا البحث باستخدام بطاقة التطوير DE2_115. يوضح الشكل (4.5) الشكل العام لهذه البطاقة. تسمح هذه البطاقة بتنفيذ تطبيقات الفيديو المعقدة كونها تحتوي على شريحة FPGA من شركة Altera عالية التعقيد وعلى ذواكر كبيرة الحجم وعلى طرفيات خاصة لإدخال وإظهار الفيديو. جرى اعتماد برمجيات Quartus II من شركة Altera لتوصيف النظام المقترح للحل وإنشاؤه وبرمجته. أما المحاكاة فجرت ضمن بيئة ModelSim.

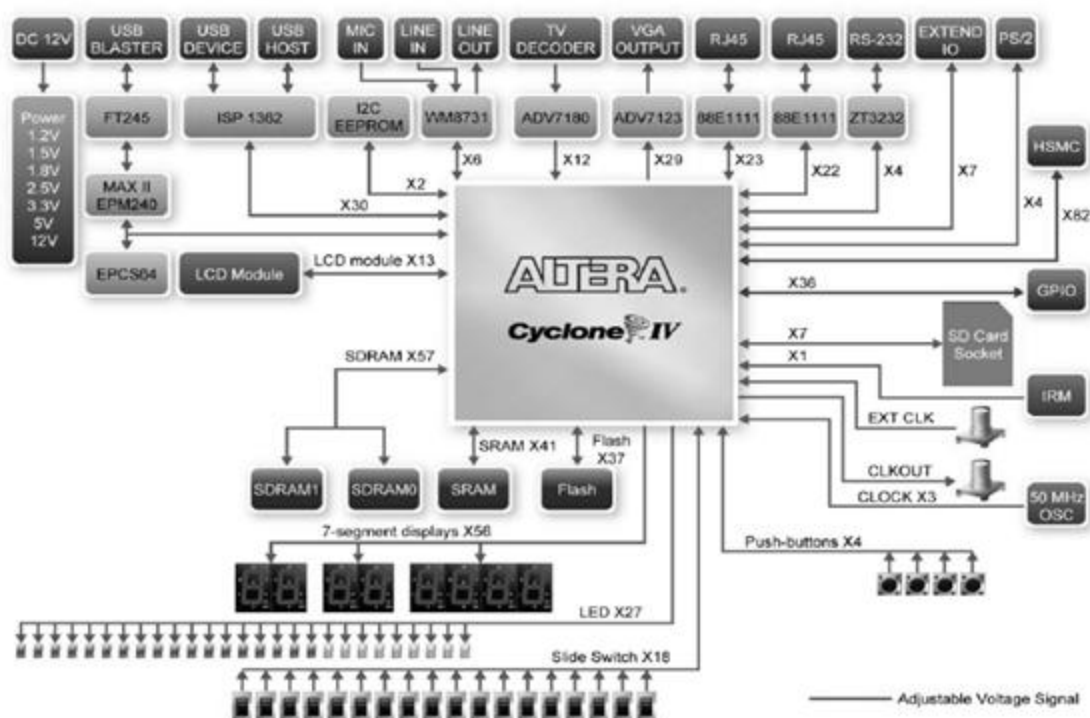
٥-٣ - بطاقة التطوير DE2_115

جرى اعتماد البطاقة DE2_115 [48] لما تتمتع به من مواصفات. يوضح الشكل (6.5) المخطط الصندوقي لهذه البطاقة. تحتوي هذه البطاقة على العناصر الآتية:

- عدد كبير من العناصر الطرفية وتشمل:
 - عدد كبير من الثنائيات الضوئية: 18 Red LEDs ,8 Green LEDs
 - عدد من وحدات الإظهار الرقمية: 7-segment displays
 - عدد من المفاتيح والأزرار: 4 Push-button ,18 Switches
 - شاشة إظهار 16×2 LCD
 - مبدل رقمي تماثلي: VGA DAC (ADV7123) يسمح هذا المبدل بتوليد الإشارات التماثلية للألوان الرئيسية (الأحمر والأخضر والأزرق)، كما هو موضح بالشكل (5.5).



الشكل (5.5): الوصلات بين شريحة FPGA و VGA [48]



الشكل (6.5): المخطط الصندوقي للبطاقة DE2_115 [48]

يجري توليد إشارات التزامن الأفقي والعمودي VGA_HS/VGA_VS من قبل شريحة FPGA. تعطى المواصفات الزمنية لهاتين الإشارتين في الجدولين (1.5) و(2.5).

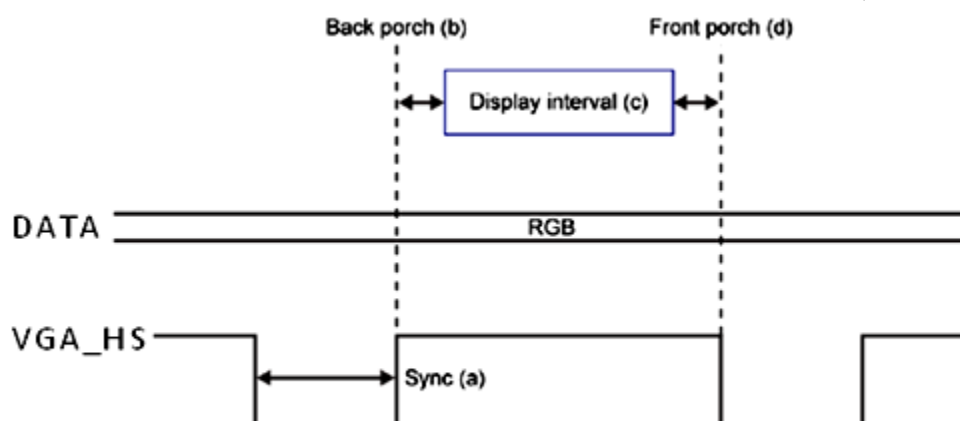
VGA mode		Horizontal Timing Spec				
Configuration	Resolution(HxV)	a(us)	b(us)	c(us)	d(us)	Pixel clock(MHz)
VGA(60Hz)	640x480	3.8	1.9	25.4	0.6	25
VGA(85Hz)	640x480	1.6	2.2	17.8	1.6	36
SVGA(60Hz)	800x600	3.2	2.2	20	1	40
SVGA(75Hz)	800x600	1.6	3.2	16.2	0.3	49
SVGA(85Hz)	800x600	1.1	2.7	14.2	0.6	56

الجدول (1.5): المواصفات الزمنية لإشارة التزامن الأفقي [48]

VGA mode		Vertical Timing Spec				
Configuration	Resolution(HxV)	a(lines)	b(lines)	c(lines)	d(lines)	Pixel clock(MHz)
VGA(60Hz)	640x480	2	33	480	10	25
VGA(85Hz)	640x480	3	25	480	1	36
SVGA(60Hz)	800x600	4	23	600	1	40
SVGA(75Hz)	800x600	3	21	600	1	49
SVGA(85Hz)	800x600	3	27	600	1	56

الجدول (2.5): المواصفات الزمنية لإشارة التزامن العمودي [48]

يظهر الشكل (7.5) إشارة التزامن الأفقي VGA_HS. تبدأ هذه الإشارة بنبضة صفرية عرضها (a) وتمثل نهاية سطر من الصورة وبداية سطر جديد. يجري إظهار البيكسلات على الشاشة لفترة زمنية (c) تتعلق بعرض الصورة. تمتد هذه الفترة بعد زمن (b) من بداية سطر إلى ما قبل نهاية سطر بزمن (d). تتشكل إشارة التزامن العمودي بطريقة مشابهة لإشارة التزامن الأفقي حيث تشير النبضة الصفرية للإشارة VGA_VS إلى نهاية إطار صورة وبداية آخر.



الشكل (7.5): إشارة التزامن الأفقي VGA_HS [48]

■ ذواكر خارجية كبيرة الحجم:

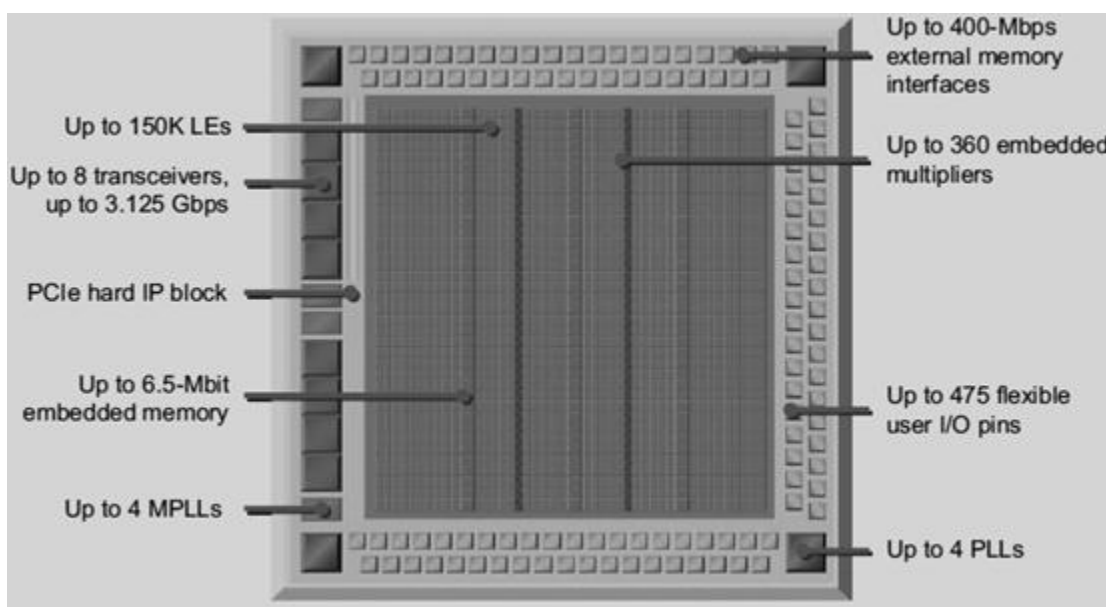
- ذاكرة من نوع SRAM بحجم 2MB وعرض 16-bits. وهي ذاكرة مؤقتة، سهلة الاستخدام وتعمل بسرعات عالية: 125 MHz.

- ذاكرتان من نوع SDRAM حجم كل منهما 64MB بعرض 16-bits. الذاكرة SDRAM هي أيضاً ذاكرة مؤقتة، تتميز بأحجامها الكبيرة وتحتاج إلى بناء متحكم للتعامل معها.
- ذاكرة من نوع FLASH بحجم 8MB وعرض 8-bits. عادةً ما تستخدم هذه الذاكر لتخزين الصور والفيديو كونها ذواكر دائمة تحتفظ بمحتواها بعد إيقاف التشغيل.

■ شريحة FPGA:

هذه الشريحة من نوع EP4CE115F29C8N من عائلة Cyclone IV E (الشكل 8.5). وتضم:

- 114480 عنصر منطقي Logic Elements
- ذاكرة داخلية بحجم 3888 Kbits
- ضواريب مضمنة 18×18 بعدد 266
- حلقة إقفال الطور PLL عدد 4
- 528 user I/O



الشكل (8.5): الشريحة EP4CE115F29C8N [48]

■ يمكن ربط البطاقة بكاميرا رقمية

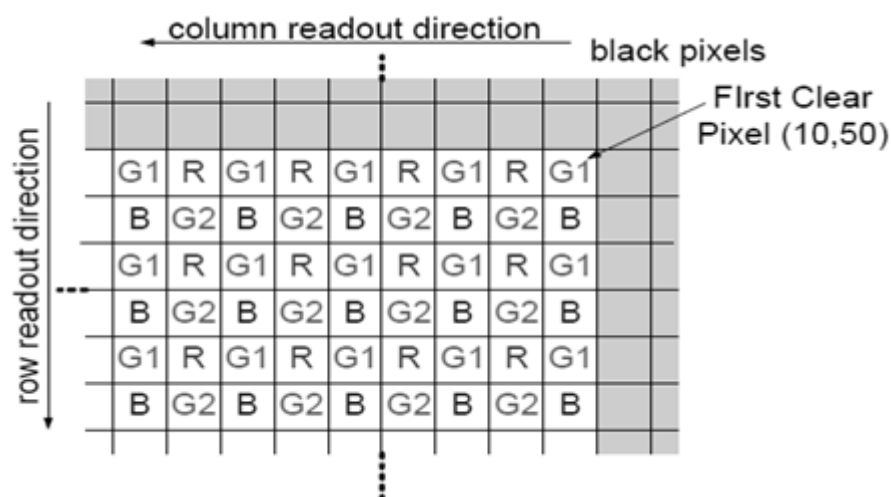
تجري عملية الربط إما بوساطة مدخل الفيديو التماثلي الموصول بمفكك الضغط الفيديوي أو بوساطة وصلة خارجية كما هو وضع الكاميرا المستخدمة في مشروعنا. هذه الكاميرا هي من نوع TRDB_D5M من شركة Terasic (الشكل 9.5).



الشكل (9.5): ربط الكاميرا TRDB_D5M بالبطاقة DE2_115 [49]

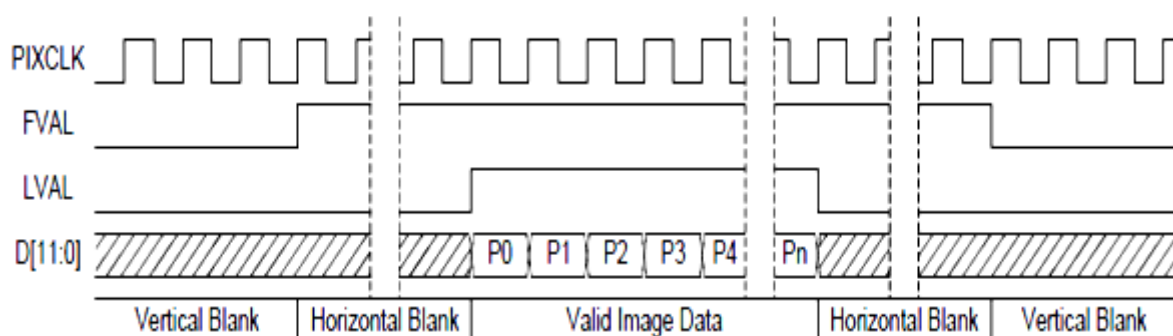
٥-٤- الكاميرا الرقمية TRDB-D5M

تتألف الكاميرا من مصفوفة من البيكسلات: 2752 عمود و 2004 سطر تعنون بواسطة رقم العمود والسطر، والعنوان (column 0, row 0) الزاوية العليا اليمنى للمصفوفة. يرتبط كل بيكسل بحساس ضوئي أحمر أو أخضر أو أزرق [49][50]. تتوضع حساسات الألوان على شكل مصفوفة باير "Bayer Matrix"، حيث يتألف السطر الأول من تناوب بيكسلات R, G1 (الشكل 10.5) ويتألف السطر الثاني من تناوب بيكسلات B, G2 وهكذا.... يمتلك اللونان G1, G2 نفس الحساس ولكن يعاملان كلونين مستقلين.



الشكل (10.5): توزيع بيكسلات الألوان ضمن الأسطر [50]

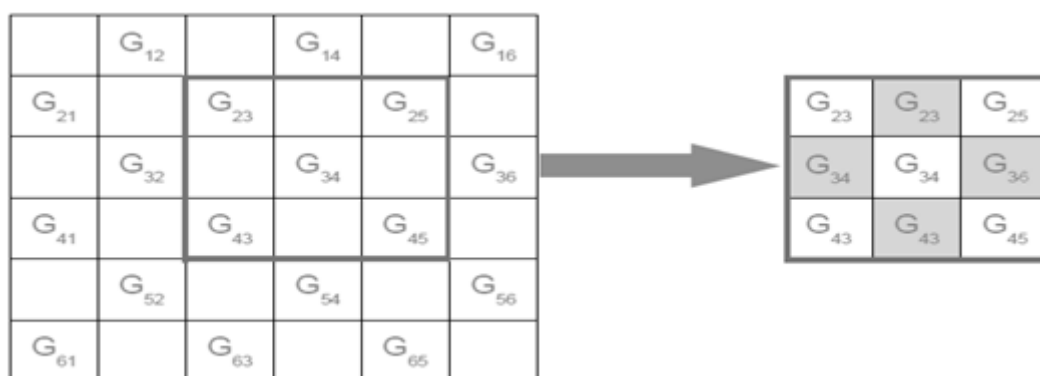
تقسم صورة الخرج إلى أطر ويُقسّم كل إطار إلى أسطر. وتحدد الإشارتان LINE_VAL و FRAME_VAL حدود الأسطر والأطر على الترتيب.



الشكل (11.5): طريقة قراءة الصورة زمنياً [50]

تجري قراءة البيكسل P_i (الشكل 11.5) من الصورة عند كل دورة (Cycle) من الساعة $PIXCLK$ عندما تكون $FRAME_VAL=1$ و $LINE_VAL=1$. ويجري تمثيل البيكسل على 12-bits.

يتألف كل بيكسل ضمن الصورة نمط RGB من ثلاثة عينات لونية (R, G, B). أما ضمن مصفوفة باير "Bayer Matrix" فكل بيكسل هو عبارة عن لون واحد فقط من الألوان (R, B, G1, G2) و من ثم كل بيكسل ينقصه عينتان ويجب أن نقوم باستيفاء هاتين العينتين كي نحصل على الصورة RGB. تستخدم لهذا الغرض طريقة تكرار الجار الأقرب. يوضح الشكل (12.5) مثلاً عن استيفاء اللون الأخضر الناقص في أربع بيكسلات، وتطبق الطريقة ذاتها على باقي البيكسلات ومن أجل اللونين الأحمر والأزرق.



الشكل (12.5): استيفاء اللون الأخضر [50]

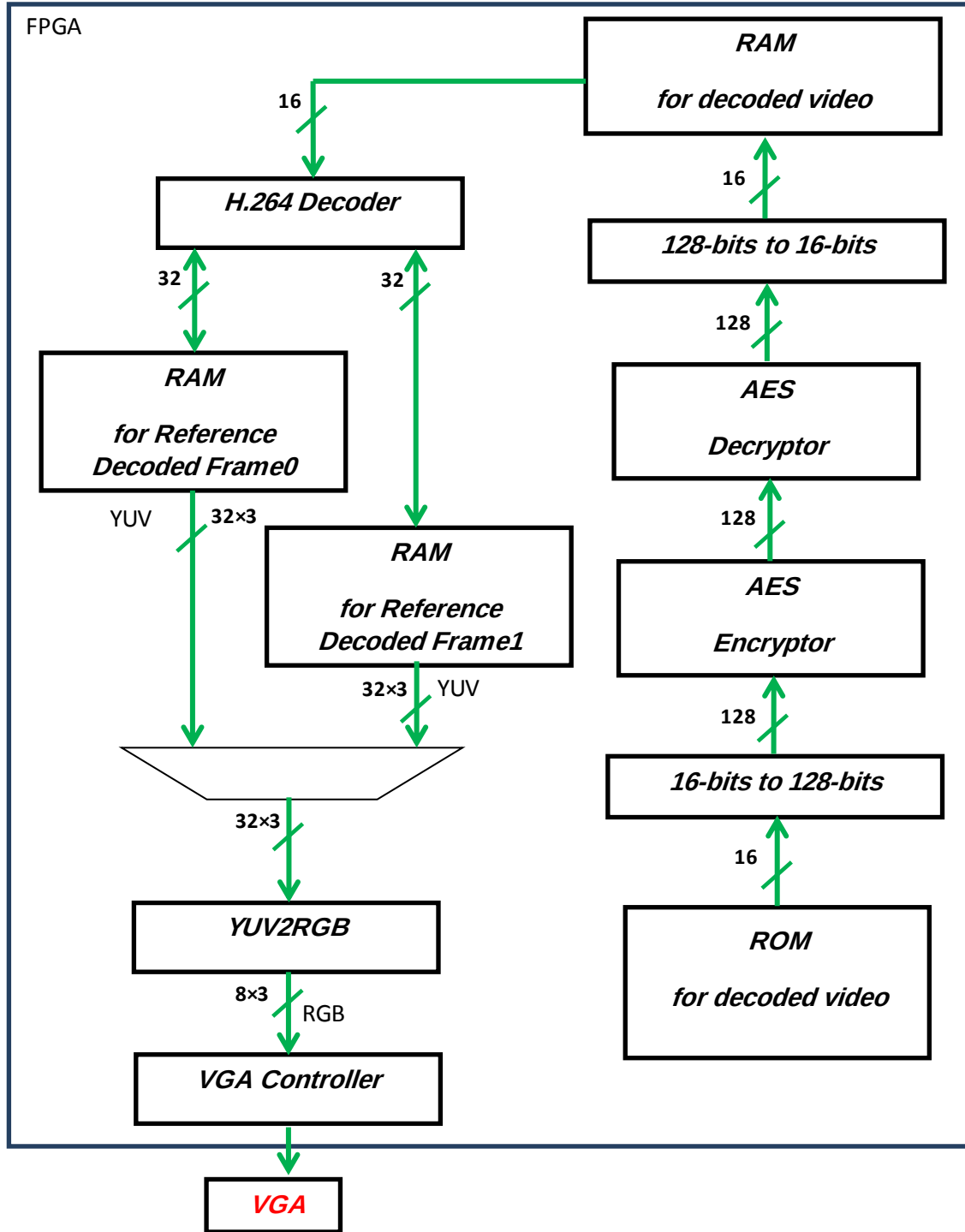
٥-٥ - تصميم النظام

يبين الشكل (13.5) المخطط الصندوقي للنظام. ويتألف من العناصر الآتية:

- ذاكرة ROM لتخزين مقطع الفيديو المضغوط.
- الوحدة AES Encryptor وتقوم بتعمية مقطع الفيديو كلمة كلمة بعرض 128-bits.
- الوحدة AES Decryptor وتقوم بفك تعمية الكلمات المُعمّاة.
- ذاكرة RAM لاستعادة مقطع الفيديو المضغوط.

- الوحدة H.264 Decoder وتقوم بفك ضغط الفيديو المضغوط.
- ذاكرتان لتخزين الإطار الحالي والإطار السابق على الترتيب بعد فك ضغطهما.
- الوحدة YUV2RGB وتقوم بتحويل معطيات الفيديو من الفضاء YUV إلى الفضاء RGB.
- الوحدة VGA Controller وتقوم بتوليد إشارات التزامن اللازمة للشاشة إضافة إلى إرسال إشارات الألوان.

صُممت وحدات النظام باللغتين VHDL و Verilog. جرت محاكاة النظام ضمن بيئة ModelSim. وجرى برمجة النظام على شريحة FPGA من نوع Cyclone vi e تتسع لما يقارب 115000 عنصر منطقي وتوجد على بطاقة التطوير DE2_115. يظهر الشكل (14.5) منهجية العمل المتبعة للوصول إلى النظام النهائي، عملنا على تصميم مفك الضغط انطلاقاً من بعض التصاميم الجاهزة بلغة Verilog لمكونات الضاغط [57][58]، فقد قمت بإعادة كتابة وحدات التنبؤ الداخلي والبيني، ووحدة التكمية والتحويل العكسيين بلغة VHDL، كما جرى إعادة صياغة وحدات التحكم الرئيسية والفرعية، ووحدات توليد الساعات في وحدة مفك الضغط بلغة Verilog. بينما جرى تصميم كامل وحدات التعمية وفك التعمية، ووحدات الذواكر، ووحدات الإظهار بلغة VHDL، إضافة إلى التصميم العام لمختلف التطبيقات. جرى التحقق من عمل مفك الضغط باستخدام مقطع فيديو مضغوط بصيغة (264). كالمقطع akiyo.264 [56]. وانطلاقاً من المراحل السابقة جرى الحصول على نظام يقوم بقراءة مقطع فيديو مضغوط ومخزن ضمن ذاكرة، ثم يعمل على تعميته وفك تعميته وفك ضغطه ومن بعدها يعرضه على الشاشة.



الشكل (13.5): المخطط الصندوقي للتطبيق النهائي على شريحة FPGA



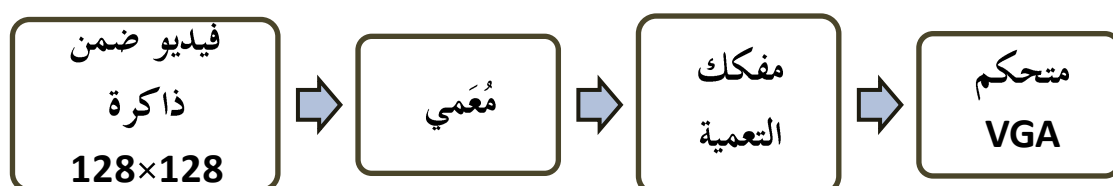
الشكل (14.5): منهجية العمل المتبعة

٥-٦- اختبار النظام

جرى اختبار الخوارزميات المستخدمة في هذا المشروع على بطاقة التطوير DE2_115 من خلال أربع تطبيقات، وسوف نأتي على شرحها تباعاً.

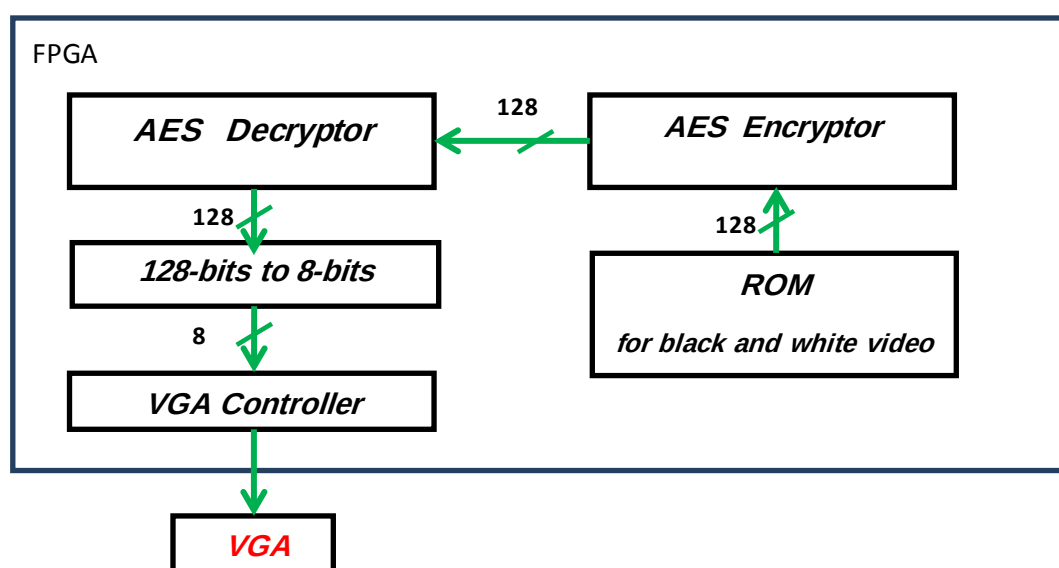
٥-٦-١- التطبيق الأول: تعمية مقطع فيديو مخزن ضمن ذاكرة وفك تعميته

يقوم التطبيق الأول على بناء المُعَمِّي ومفكك التعمية وفق المعيار AES (الشكل 15.5)، ويطبق ذلك على مقطع فيديو قصير مؤلف من مجموعة صور ثنائية اللونين الأبيض والأسود وبقياس 128×128 بيكسل.



الشكل (15.5): المخطط الصندوقي للتطبيق الأول

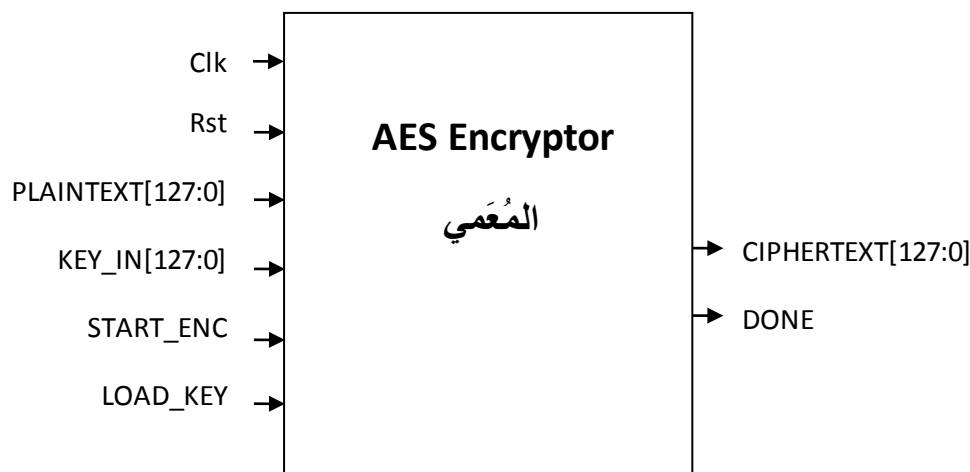
يظهر الشكل (16.5) المخطط الصندوقي للتطبيق الأول. يجري تخزين مقطع الفيديو في ذاكرة ROM بعرض 128-bits بحيث يحتوي كل سطر من الذاكرة على سطر من الصورة فكل بيكسل يمثل على 1-bit (أبيض أو أسود). جرى اختيار عرض الصورة وعرض الذاكرة بما يتناسب مع دخل المُعَمِّي، حيث يتلقى المُعَمِّي كلمة بعرض 128-bits ليقوم بتعميتها بنفس العرض. تطبق من بعدها عملية فك التعمية وترسل النتيجة إلى المتحكم VGA لإظهار الفيديو الناتج. عند استخدام نفس المفتاح من قبل المُعَمِّي ومفكك التعمية، يظهر الفيديو المخزن على الشاشة وإلا لن يتمكن مفكك التعمية من استرجاع الفيديو الأصلي فتظهر على الشاشة صورة مغرلة.



الشكل (16.5): المخطط الصندوقي للتطبيق الأول على شريحة FPGA

- توصيف المُعَمِّي وفق المعيار AES

يظهر الشكل (17.5) مداخل ومخارج المُعَمِّي وفق المعيار AES. يجري في المرحلة الأولى شحن المُعَمِّي بمفتاح التعمية KEY_IN عند تفعيل نبضة واحدة من الإشارة LOAD_KEY.

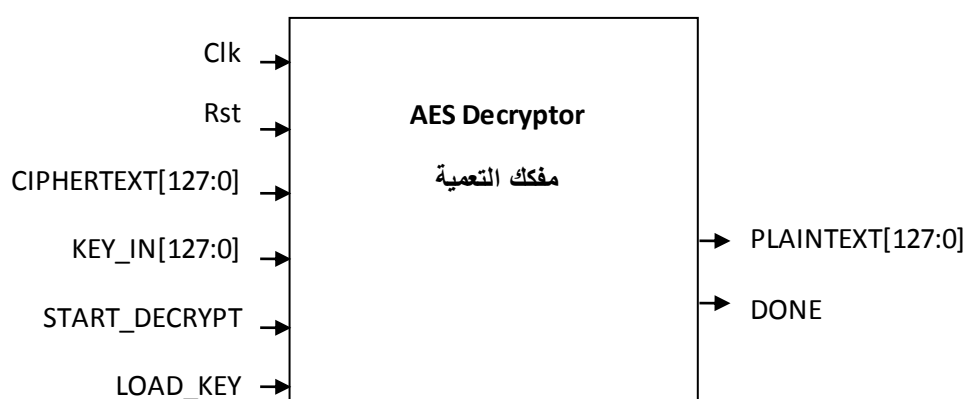


الشكل (17.5): مداخل ومخارج المُعَمِّي

تبدأ التعمية عند تفعيل الإشارة START_ENC=1. النص الواضح PLAINTEXT بعرض 128-bits هو الكلمة المطلوب تعميته والكلمة الناتجة هي النص المعمي CIPHERTEXT بنفس العرض. تحتاج عملية التعمية إلى 48 نبضة ساعة وعند نهاية العملية يجري تفعيل نبضة من الإشارة DONE.

- توصيف مفكك التعمية وفق المعيار AES

يظهر الشكل (18.5) المداخل والمخارج لمفكك التعمية وفق المعيار AES.



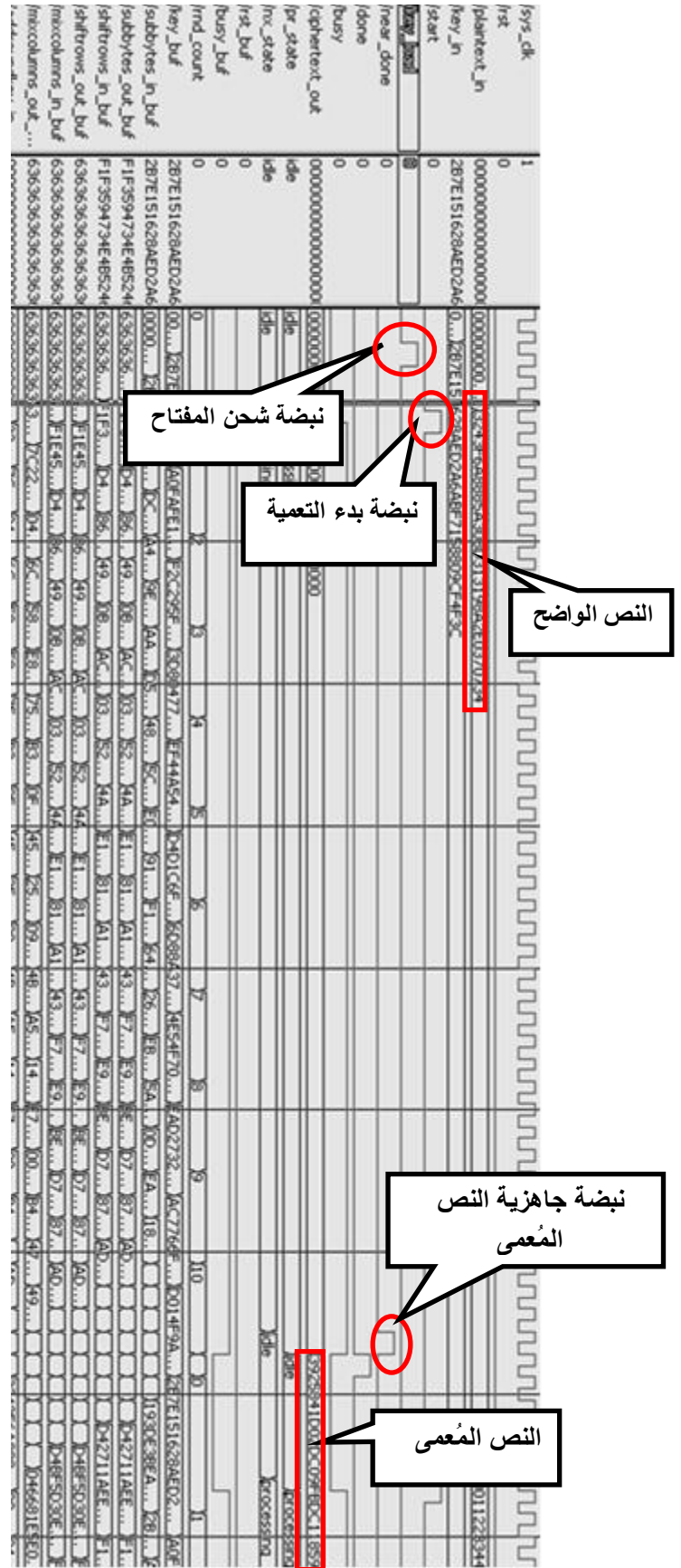
الشكل (18.5): مداخل ومخارج مفكك التعمية

وبطريقة مشابهة للمُعَمِّي يجري في المرحلة الأولى شحن مفكك التعمية بنفس مفتاح التعمية KEY_IN عند إعطاء نبضة واحدة للإشارة LOAD_KEY. يبدأ تفكيك التعمية عند تفعيل

الإشارة START_DECRYPT=1. كلمة الدخل CIPHERTEXT بعرض 128-bits هي النص المُعمى والكلمة الناتجة عن تفكيك التعمية هي النص الواضح PLAINTEXT بنفس العرض. تحتاج عملية فك التعمية أيضاً إلى 48 نبضة ساعة وعند نهاية هذه العملية يجرى تفعيل نبضة من الإشارة DONE.

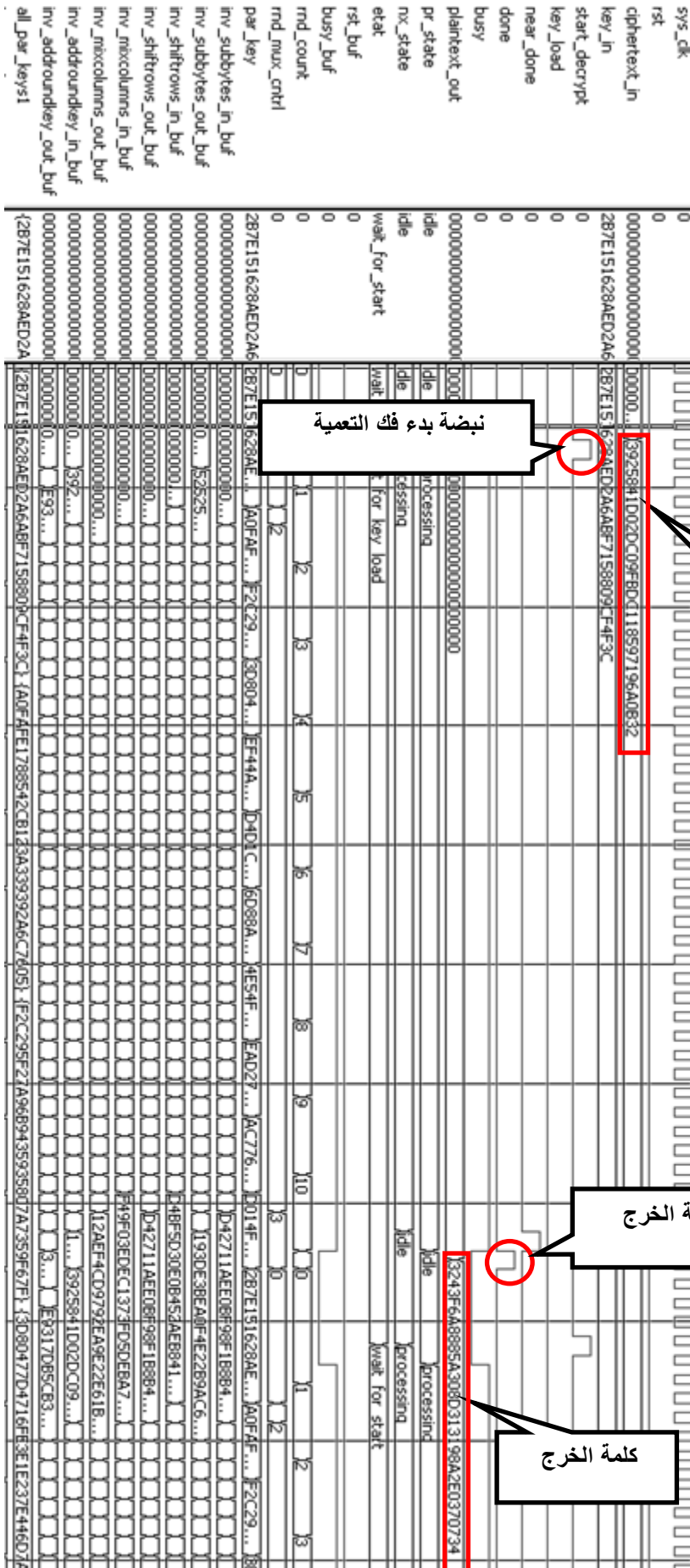
- محاكاة التطبيق الأول ضمن بيئة Modelsim

يظهر في الشكل (19.5) المخطط الزمني للمُعَمَّى. فمثلاً عند إدخال النص الواضح PLAINTEXT بقيمة وقدرها: 3243F6A8...370734 بصيغة Hexadecimal، تكون قيمة النص المُعمى CIPHERTEXT: 3925841D...A0B32 وذلك من أجل قيمة للمفتاح KEY_IN قدرها 2B7E1516...CF4F3C.



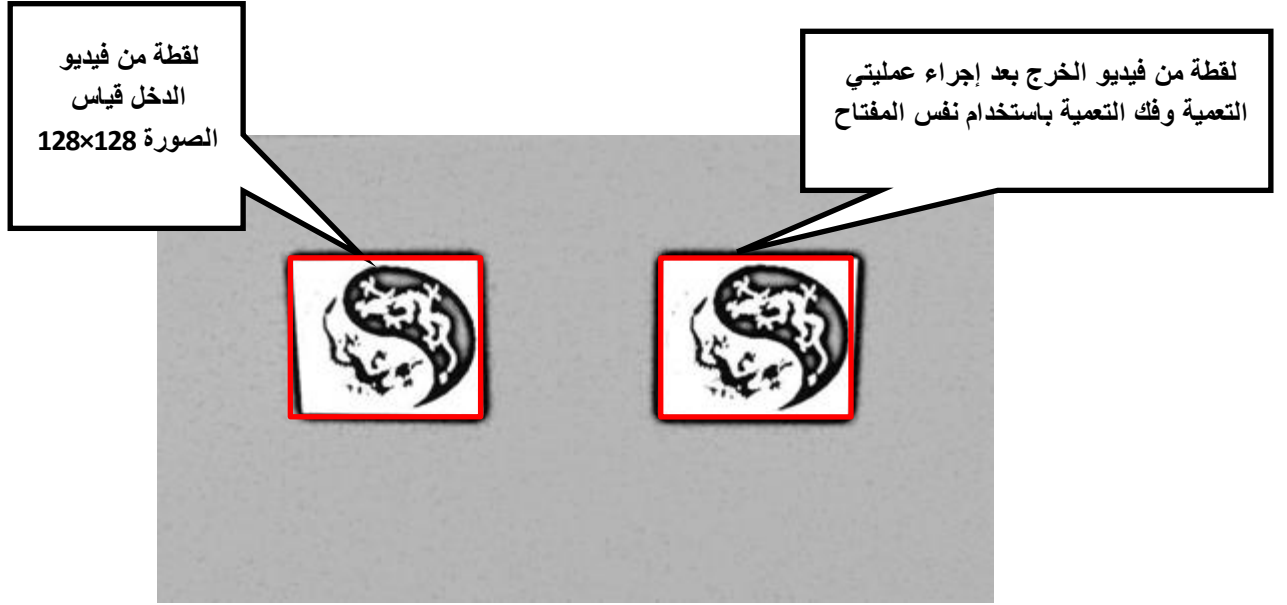
الشكل (19.5): المخطط الزمني للمعمي

كما يظهر في الشكل (20.5) المخطط الزمني لمفك التعمية. نجد أنه عند إدخال النص المُعَمَّى نفسه CIPHERTEXT الناتج على خرج المُعَمَّى نحصل على النص الواضح PLAINTEXT على خرج مفك التعمية.

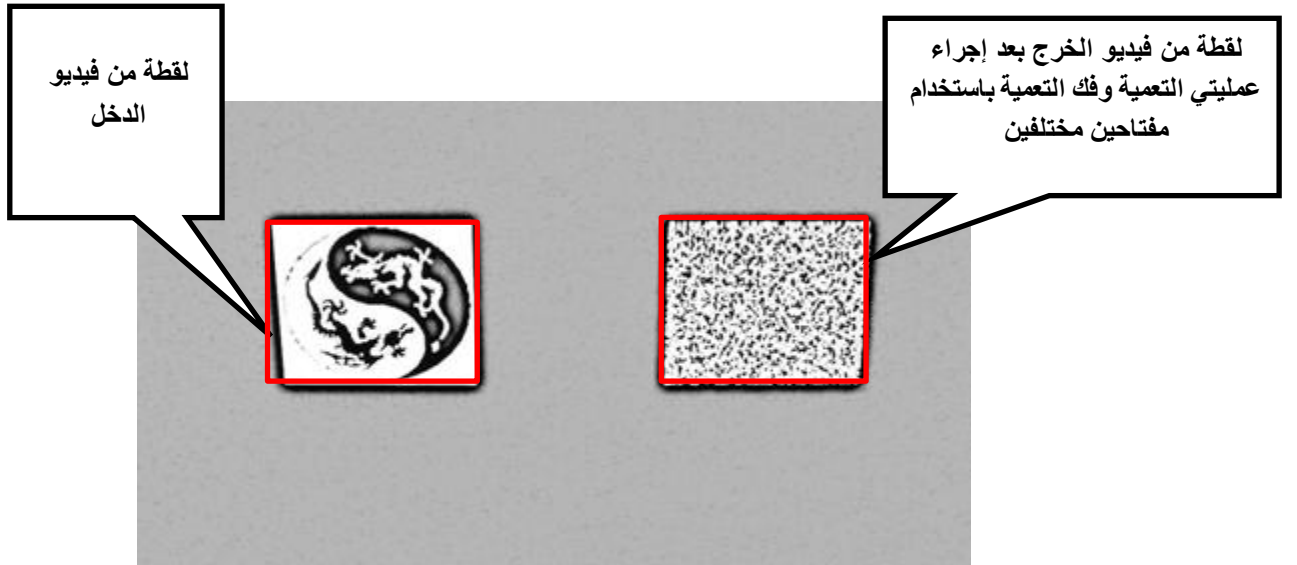


الشكل (20.5): المخطط الزمني لمفك التعمية

- تركيب وتوضيع وبرمجة التطبيق الأول على بطاقة DE2_115 والتأكد من عمله يظهر في الشكل (21.5) خرج المتحكم VGA على شاشة العرض، حيث يظهر على اليسار لقطة من فيديو الدخل وعلى اليمين لقطة من فيديو الخرج بعد إجراء عمليتي التعمية وفك التعمية باستخدام نفس المفتاح. ويظهر في الشكل (22.5) فيديو الخرج مع غريلة عند استخدام مفاتيحين مختلفين من قبل المُعمي ومفكك التعمية، حيث لا يتمكن مفكك التعمية من أداء عمله.



الشكل (21.5): خرج التطبيق الأول عند استخدام نفس المفتاح للتعمية وفك التعمية



الشكل (22.5): خرج التطبيق الأول عند عدم استخدام نفس المفتاح للتعمية وفك التعمية

يظهر الجدول (3.5) عدد العناصر المستهلكة في هذا التطبيق من حيث عدد العناصر المنطقية وحجم الذواكر الداخلية وعدد الضواريب، كما يظهر تردد العمل الأعظمي. يستهلك المُعمي ومفك التعمية عدد قليل ومتقارب من العناصر المنطقية ولا تستهلك الخوارزمية AES أياً من الضواريب. نحتاج إلى حلقة إقفال الطور من أجل توليد ساعة النظام وهي نفسها ساعة مسح البيكسلات بتردد 25 MHz.

Cyclone iv e: ep4ce115f29	المُعمي	مفك التعمية	
العناصر المنطقية LEs	5042	7202	12244 (11%)
الذاكرة Mem(bits)	-	-	98304 (2%)
الضواريب Multipliers	-	-	-
حلقة إقفال الطور PLLs	-	-	1
تردد العمل الأعظمي MHz			150 MHz

الجدول (3.5): عدد العناصر المستهلكة في التطبيق الأول

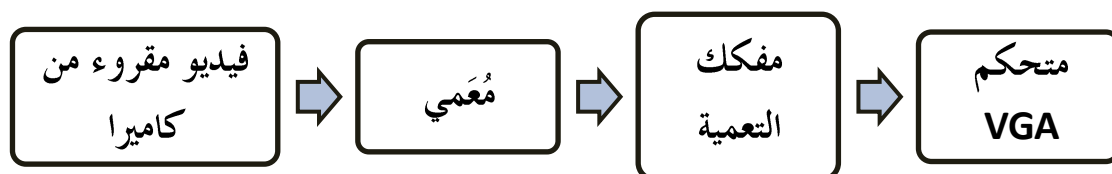
كما يبيّن الجدول (4.5) حساب سرعة التعمية والزمن الكلي اللازم لتعمية ملف الفيديو المضغوط. وجرى استنتاج أن سرعة التعمية تزداد بزيادة تردد التعمية أو بإنقاص طول مفتاح التعمية.

طول المفتاح	عدد الدورات	تردد عمل المُعمي أو مفك التعمية	مدة تعمية أو فك تعمية كلمة واحدة	سرعة التعمية أو فك التعمية	مدة تعمية أو فك تعمية ملف الفيديو المضغوط بحجم 639 Kbyte
128 bits	10 rounds	25 MHz	2 μ s	$\frac{128}{2\mu s} = 64 Mbps$	80 ms
128 bits	10 rounds	50 MHz	1 μ s	$\frac{128}{1\mu s} = 128 Mbps$	40 ms
192 bits	12 rounds	25 MHz	2.2 μ s	$\frac{128}{2.2\mu s} = 58 Mbps$	88 ms
256 bits	14 rounds	25 MHz	2.6 μ s	$\frac{128}{2.6\mu s} = 49 Mbps$	104 ms

الجدول (4.5): حساب زمن وسرعة عملية التعمية و فك التعمية

٥-٦-٢ - التطبيق الثاني: تعمية مقطع فيديو مقروء من كاميرا وفك تعميته

يقوم التطبيق الثاني على قراءة فيديو من الكاميرا TRDB_D5M ومن ثم تعميته وتفكيك تعميته وإظهار النتيجة على الشاشة (الشكل 23.5).



الشكل (23.5): المخطط الصندوقي للتطبيق الثاني

يجري تحصيل صورة أبعادها 640×480 بيكسل من كاميرا ومن ثم تخزينها في ذاكرة خارجية SDRAM. يجري استخدام الذاكرة الخارجية SDRAM إضافةً إلى ذواكر داخلية من نوع FIFO من أجل الحفاظ على مسار المعطيات والعمل بالزمن الحقيقي بحيث يمكن على التوازي تخزين الإطار الحالي وقراءة الإطار السابق. جرى توصيف عملية قراءة الفيديو من الكاميرا وتخزينها في ذاكرة SDRAM، بلغة Verilog من قبل شركة Terasic [49][50].

يقوم متحكم الكاميرا Camera Controller بقراءة الصورة من الذاكرة SDRAM وإرسالها إلى المتحكم VGA. يتألف متحكم الكاميرا من الأجزاء الرئيسية الآتية (الشكل 24.5):

- وحدة التقاط الصورة CCD_CAPTURE. تقوم هذه الوحدة بتحصيل عينات الكاميرا. تنتج كل عينة عن حساس لوني واحد (أحمر أو أخضر أو أزرق) من حساسات الكاميرا الموزعة بطريقة Bayer.

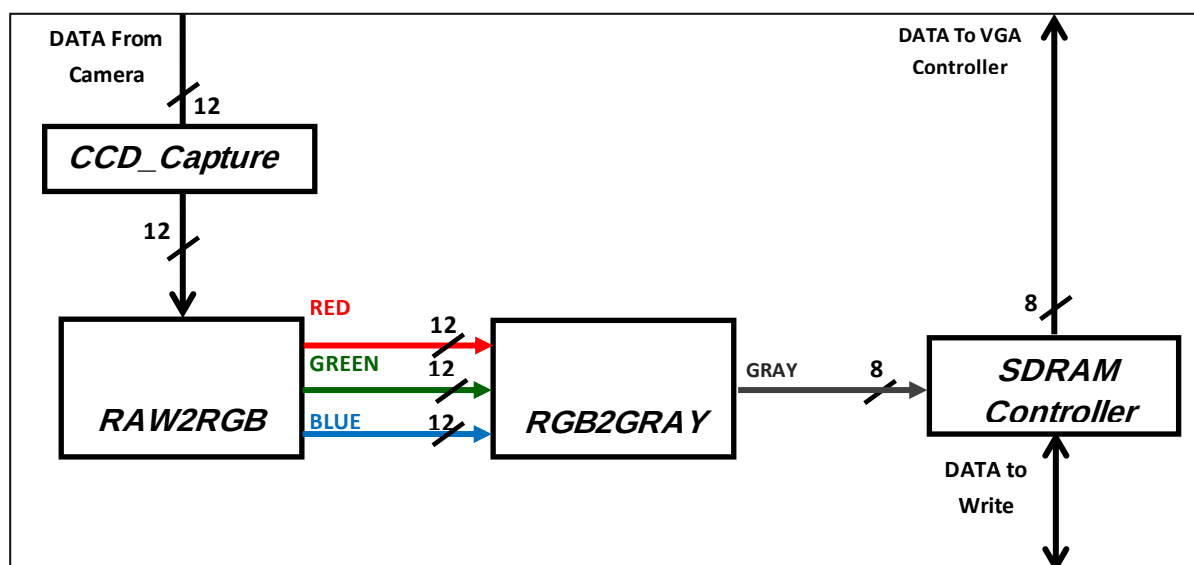
- وحدة التحويل RAW2RGB. تقوم هذه الوحدة باستيفاء المعلومات اللونية الناقصة عن كل حساس لوني بالاستفادة من الحساسات المجاورة لتعطي على خرجها المركبات اللونية الثلاثة لكل عينة من عينات الصورة.

- وحدة التحويل RGB2GRAY. تقوم هذه الوحدة بتشكيل صورة ذات تدرج رمادي.

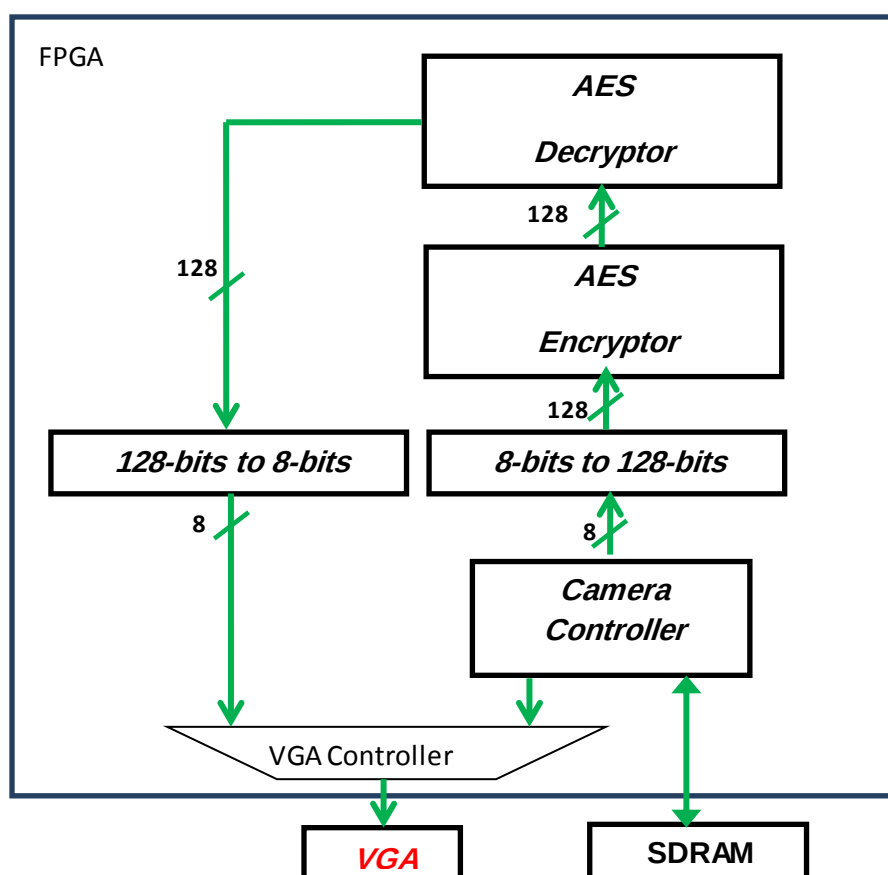
- وحدة المتحكم SDRAM Controller. يقوم هذا المتحكم بقراءة وكتابة عينات الصورة من وإلى الذاكرة SDRAM. ولا يمكن إجراء هذه العمليات إلا على دقات. للمحافظة على مسار المعطيات يقوم المتحكم باستخدام ذاكرة كتابة من نوع FIFO تكتب فيها عينات الصورة بسرعة المسح 25MHz ويقوم بنقلها وإدخالها على دقات إلى الذاكرة SDRAM بسرعة 100MHz. كما يقوم من بعدها بقراءة الذاكرة SDRAM على دقات وتعبئة ذاكرة قراءة من نوع FIFO نقوم بقراءتها من أجل المعالجة والعرض.

نأخذ صورة أبعادها 640×480 بيكسل ذات تدرج رمادي. أي يجري تمثيل كل بيكسل على 8-bits ومن ثمَّ يجب تجميع كل 16 بيكسل من أجل تشكيل كلمة بعرض 128-bits على دخل

المُعَمِّي (الشكل 25.5). تجري من بعدها عمليتي التعمية وفك التعمية على كل كلمة بعرض 128-bits. في النهاية تحول كل كلمة 128-bits إلى بايتات (بيكسلات) من أجل عرضها على الشاشة.



الشكل (24.5): متحكم الكاميرا Camera Controller



الشكل (25.5): المخطط الصندوقي للتطبيق الثاني على شريحة FPGA

- توضيح وبرمجة التطبيق الثاني على بطاقة DE2_115 والتأكد من عمله

يعرض الشكل (26.5) خرج المتحكم VGA على شاشة العرض بعد إجراء عمليتي التعمية وفك التعمية على الفيديو المقروء من الكاميرا. ويظهر الجدول (5.5) تعقيد هذا التطبيق من خلال عدد العناصر المنطقية وحجم الذواكر الداخلية وعدد الضواري. نلاحظ أن التطبيق الثاني أكثر تعقيداً من التطبيق الأول بسبب استخدام ذاكرة خارجية SDRAM لتخزين إطارين من إطارات الفيديو. نحتاج إلى حلقة إقفال الطور من أجل توليد الترددات الآتية: ساعة النظام وهي نفسها ساعة مسح البيكسلات وساعة الكاميرا بتردد 25MHz إضافةً إلى ساعة الذاكرة الخارجية SDRAM بتردد 100MHz.

14443 (13%)	العناصر المنطقية Les
101328 (3%)	الذاكرة Memory
-	الضواري
1	PLLs

الجدول (5.5): عدد العناصر المستهلكة في التطبيق الثاني



الشكل (26.5): خرج التطبيق الثاني

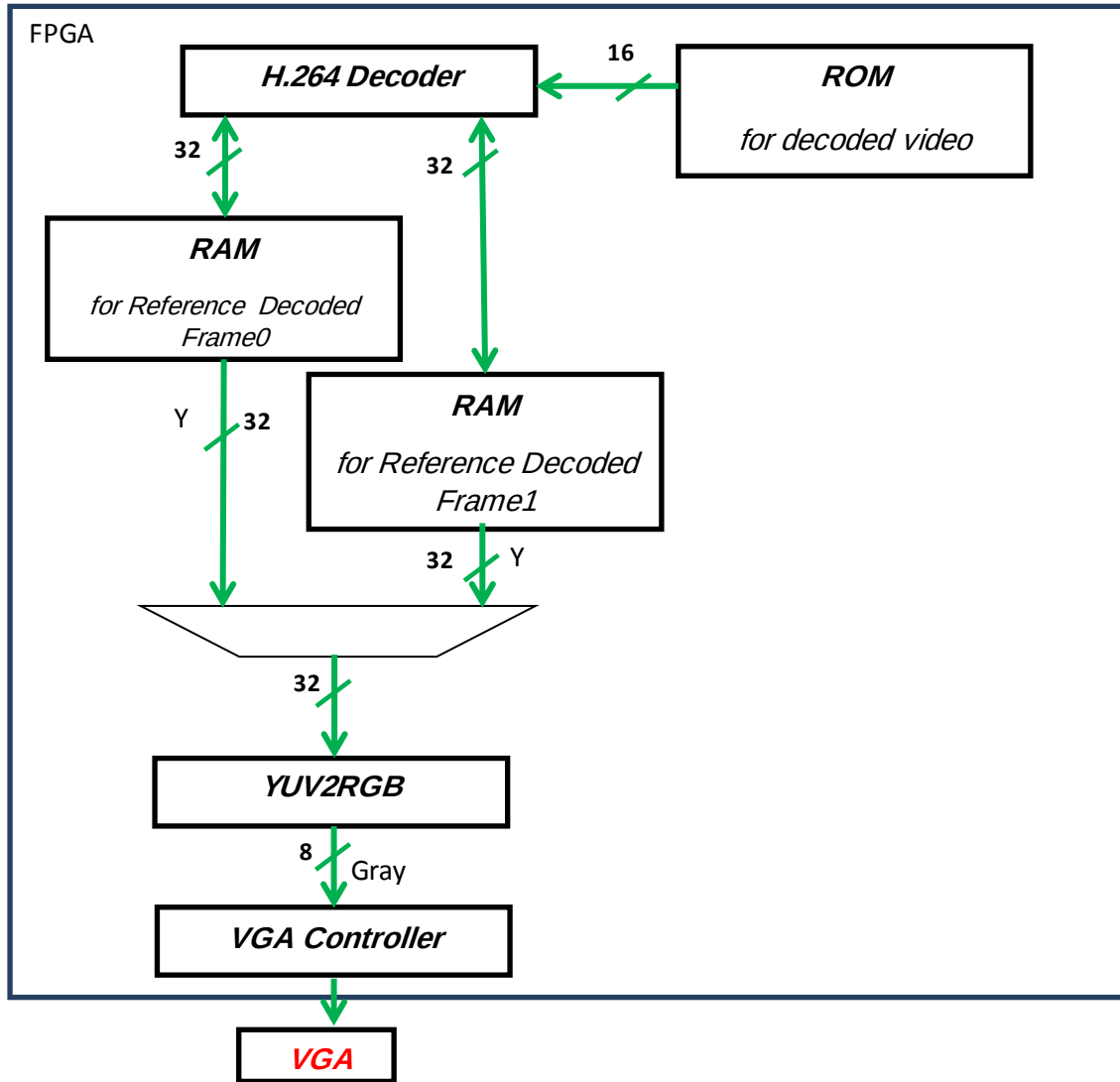
٥-٦-٣ - التطبيق الثالث: فك ضغط مقطع فيديو مضغوط ومخزن ضمن ذاكرة

نقوم في التطبيق الثالث باستخدام مقطع فيديو مضغوط بصيغة H.264 [56] وتوليد ذاكرة ROM ضمن شريحة FPGA لتخزين ذلك المقطع. تجري ضمن شريحة FPGA عملية قراءة مقطع الفيديو المضغوط من الذاكرة ROM ومن ثم فك ضغطه وإظهاره على الشاشة (الشكل 27.5).



الشكل (27.5): المخطط الصندوقي للتطبيق الثالث

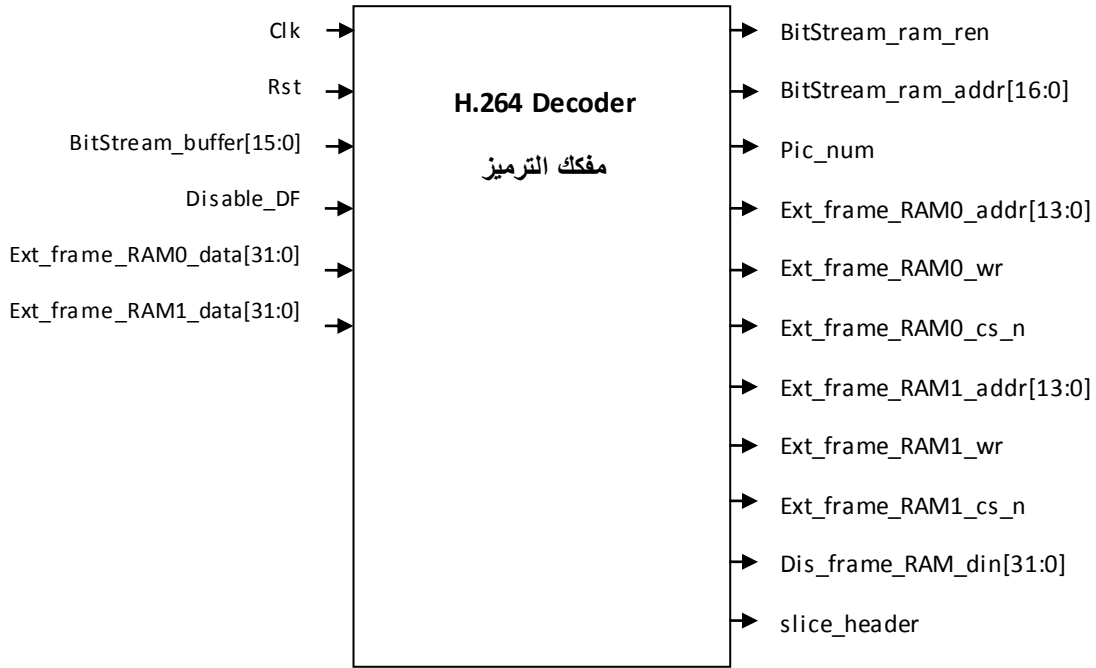
يظهر الشكل (28.5) المخطط الصندوقي لمفكك الضغط. تقوم الوحدة H.264 Decoder بقراءة مقطع الفيديو المضغوط من الذاكرة ROM وبفك ضغطه. يجري تخزين إطارات الفيديو ذات الأرقام الزوجية في الذاكرة RAM_Frame0 وذات الأرقام الفردية في الذاكرة RAM_Frame1. يقوم المتحكم VGA بقراءة الذاكرتين على التناوب بعد تحويل صيغة إطارات الفيديو من YUV إلى RGB، علماً أن مقطع الفيديو في هذا التطبيق يضم مركبة واحدة هي مركبة الإضاءة Y.



الشكل (28.5): المخطط الصندوقي للتطبيق الثالث على شريحة FPGA

- توصيف مفكك الضغط

يعرض الشكل (29.5) مداخل ومخارج مفكك الضغط. يبدأ مفكك الضغط بقراءة المقطع المضغوط من ذاكرة ROM ومن ثم يقوم بتفكيك الضغط. يستخدم مفكك الضغط الذاكرتين RAM_Frame0 و RAM_Frame1. فيقوم على التوازي بكتابة إطار بعد فك تعميته في إحدى الذاكرتين وبقراءة الإطار الذي سبق فك تعميته من الذاكرة الأخرى ليستخدمه كإطار مرجعي. تعمل هاتان الذاكرتان على التناوب. تمثل الإشارة Dis_frame_RAM_din خرج مفكك الضغط. وتعطي الإشارة Pic_num رقم الإطار الذي تجري معالجته.

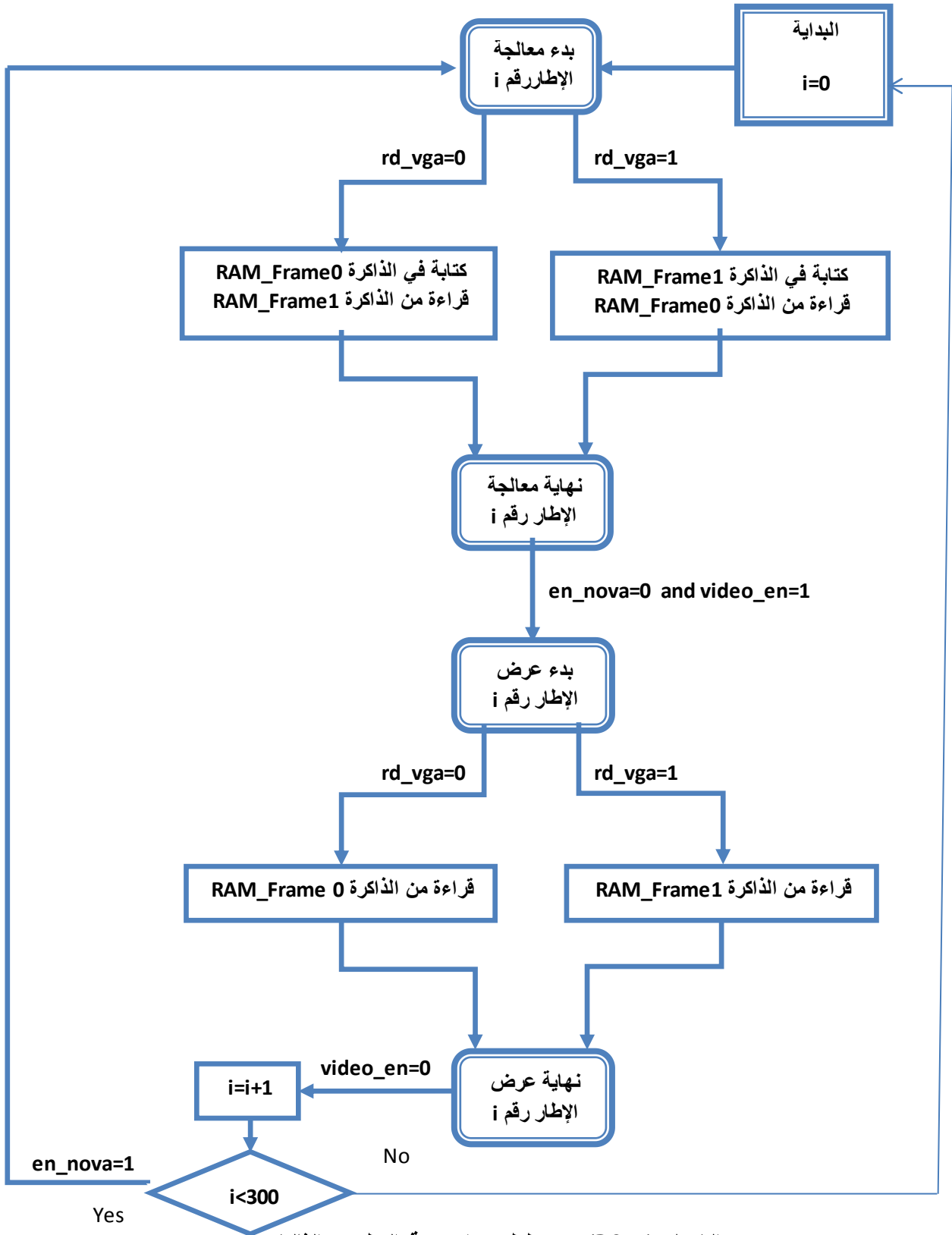


الشكل (29.5): مداخل ومخارج مفكك الضغط

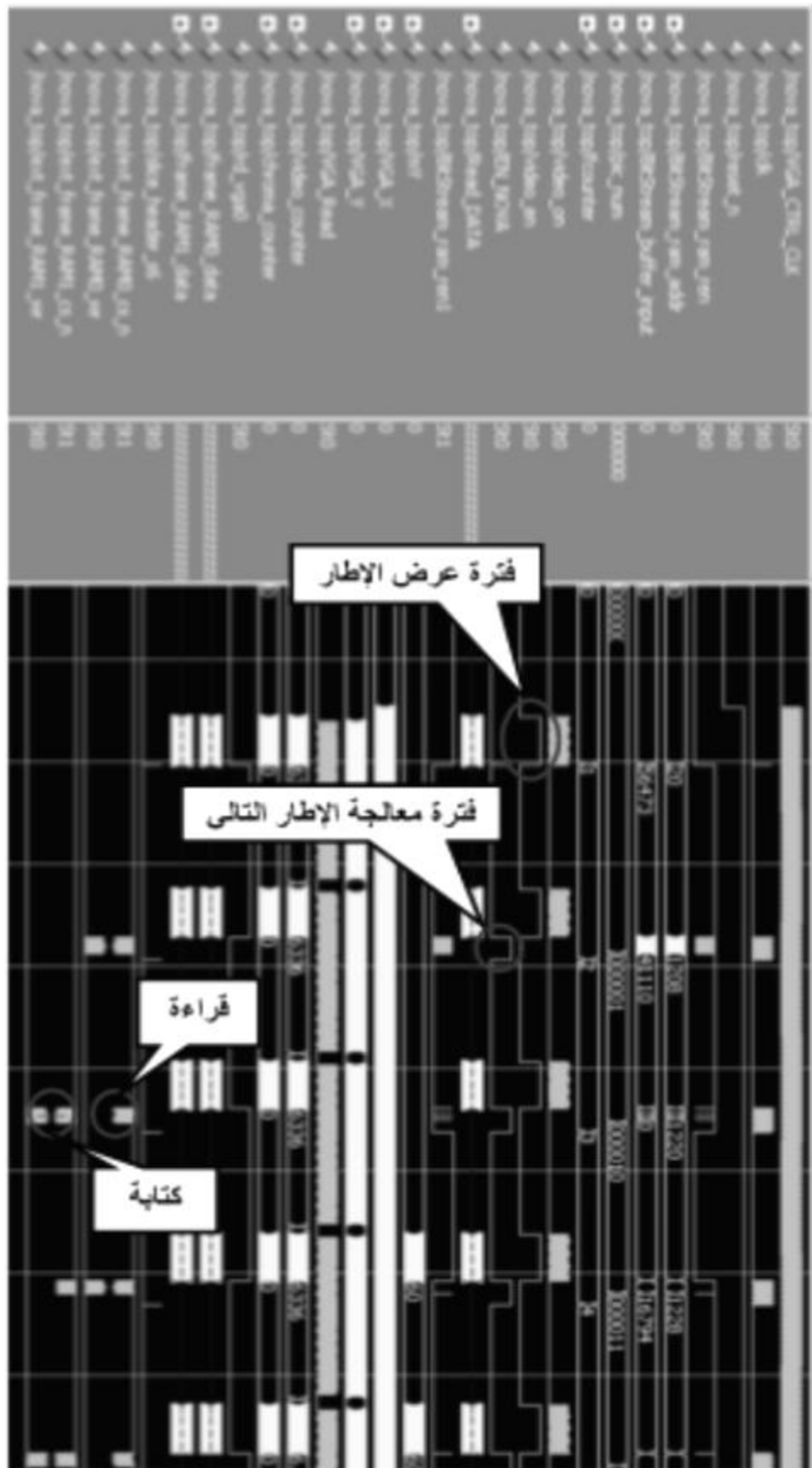
تعطي الإشارة slice_header نبضة عند نهاية معالجة كل إطار. تسمح الإشارة Disable_DF عند تفعيلها بإلغاء عمل مرشح التنعيم deblocking filter.

- محاكاة التطبيق الثالث ضمن بيئة Modelsim

يبين الشكل (30.5) مخطط خوارزمية التطبيق الثالث. تجري معالجة الإطار خلال المدة التي تكون فيها الإشارة EN_NOVA=1 ويجري بعدها عرض الإطار الذي تمت معالجته خلال المدة التي تكون فيها الإشارة video_en=1، أي خلال عرض نافذة بالأبعاد 176×144. تصبح الإشارة EN_NOVA=0 عند الانتهاء من معالجة الإطار، أي عند ورود النبضة slice_header. تسمح الإشارة rd_vga باستخدام الذاكرتين RAM_Frame0 و RAM_Frame1 بالتناوب. يتألف مقطع الفيديو من 300 إطار، و يعرض الشكل (31.5) المخطط الزمني لمفكك الضغط الذي جرت محاكاته ضمن بيئة ModelSim.

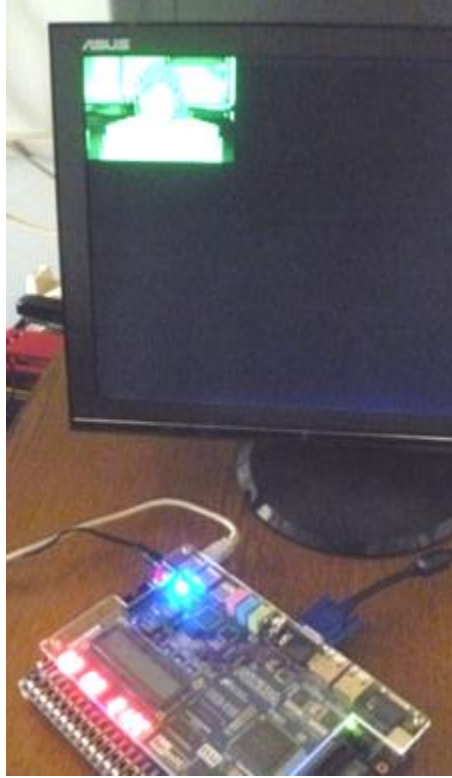


الشكل (30.5): مخطط خوارزمية التطبيق الثالث



الشكل (31.5): المخطط الزمني لمفكك الضغط

- تركيب وتوضيع وبرمجة التطبيق الثالث على بطاقة DE2_115 والتأكد من عمله يعرض الشكل (32.5) نتيجة إظهار الفيديو بعد فك ضغطه. يتألف الفيديو المستخدم من 300 إطار، حيث يظهر هذا الرقم بصيغة Hexadecimal على لوحة 7-segments في البطاقة. كما تظهر الصورة بلون مائل إلى الأخضر وهذا بسبب إظهار مركبة وحيدة وهي المركبة Y وكان الغرض من ذلك تصغير حجم الذاكر وتبسيط التطبيق كمرحلة أولية.



الشكل (32.5): خرج التطبيق الثالث

يظهر الجدول (6.5) تعقيد مفكك الضغط بمفرده دون إدخال ذاكر الإطارات ووحدة الإظهار. نلاحظ أن وحدتي الجمع SUM ووحدة التكمية والتحويل العكسيين IQIT تستهلكان العدد الأقل من العناصر المنطقية، إضافةً إلى استهلاك الوحدة IQIT لعدد من الضواريب لإتمام عملية التحويل. تستهلك وحدة التنبؤ البيئي عدداً من العناصر المنطقية أكثر من وحدة التنبؤ الداخلي وذلك لأن التنبؤ البيئي يحتاج إلى عمليات حسابية أكثر تعقيداً كحساب أشعة الحركة بدقات مختلفة تصل إلى مئز الربع بيكسل. أما بالنسبة لوحدة التحكم bitstream controller والتي تضم مفكك الترميز الانتروبي فتستهلك فعلياً 3229 عنصراً منطقياً بينما تُستخدم العناصر المتبقية 6936 كسجلات لتخزين الجداول الخاصة بالترميز (الملحق A). ويحتاج المرشح الكتلي DF Filter ذاكرة بحجم 6144-bits لتطبيق المرشح على الحواف الأفقية والعمودية.

مفكك الضغط	Filter DF	Intra pred.	Inter pred.	SUM	IQIT	Bitstream Control.	
44328 (51%)	4303	8107	18596	770	2387	10165	Les
6144 (0%)	6144	-	-	-	-	-	Mem. Bits
4	-	-	-	-	4	-	Mult.

الجدول (6.5): عدد العناصر المستهلكة في التطبيق الثالث

يبين الجدول (7.5) أن زمن معالجة (فك ضغط) إطار واحد يساوي تقريباً نصف زمن مسح الإطار عند استخدام نفس الساعة من أجل المسح وفك الضغط (في حالتنا 25MHz). مما يتيح لنا معالجة كل إطار قبل إظهاره، أي إمكانية العمل بالزمن الحقيقي. وجرى الاستنتاج أنه يمكن تسريع المعالجة باستخدام تردد أكبر للمعالجة من تردد المسح.

الميز	تردد المسح	تردد عمل مفكك الضغط	زمن المسح	زمن المعالجة (فك الضغط)
176×144	25 MHz	25 MHz	4.6 ms/frame	2.2 ms/frame
176×144	25 MHz	50 MHz	4.6 ms/frame	1.1 ms/frame
352×288	25 MHz	25 MHz	18.4 ms/frame	8.8 ms/frame
704×576	25 MHz	25 MHz	73.6 ms/frame	35.2 ms/frame

الجدول (7.5) مقارنة زمن فك الضغط مع زمن المسح

٥-٦-٤ - التطبيق الرابع: ترميز وفك ضغط مقطع فيديو مضغوط

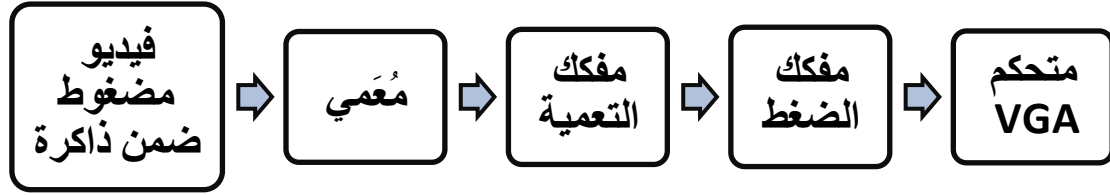
ومخزن ضمن ذاكرة

يضيف التطبيق الرابع الترميز وفك الترميز إلى التطبيق الثالث كما هو موضح بالشكل (32.5) والشكل (13.5).

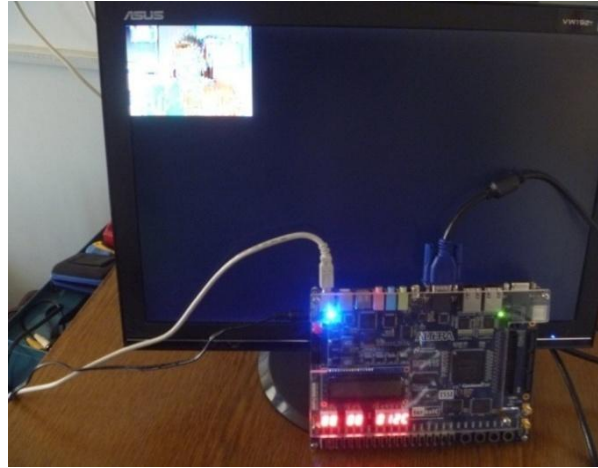
- توضيح وبرمجة التطبيق الرابع على بطاقة DE2_115 والتأكد من عمله

يعرض الشكل (33.5) نتيجة إظهار الفيديو بعد ترميزه وفك ترميزه وفك ضغطه. تظهر الصورة ملونة وهذا بسبب إظهار المركبات الثلاثة YUV. تظهر الصورة مع بعض التشويه بسبب عدم التزامن بين المركبات الثلاثة، ويرجع السبب إلى مشاكل التوصيل والتوضيح

والمسارات الحرجة كون التطبيق يستهلك معظم الذاكر والعاصر المنطقية في شريحة FPGA وقد يكمن الحل باستخدام ذاكر خارجية لتسهيل عمليات توصيل وتوضيع العناصر.



الشكل (32.5): المخطط الصندوقي للتطبيق الرابع



الشكل (33.5): خرج التطبيق الرابع

يظهر الجدول (8.5) تعقيد هذا التطبيق من خلال عدد العناصر المنطقية وحجم الذاكر الداخلية وعدد الضواري المستخدمة، كما يظهر تردد العمل الأعظمي.

72010 (62%)	العناصر المنطقية Les
1316864 (40%)	الذاكرة Memory
4	الضواري Multipliers
1	حلقة إقفال الطور PLLs
250 MHz	تردد العمل الأعظمي

الجدول (8.5): عدد العناصر المستهلكة في التطبيق الرابع

يدخل في حساب التعقيد جميع الوحدات المكونة للنظام (الشكل 13.5): المُعْمِي، ومفكك التعمية، ومفكك الضغط إضافةً إلى الذاكر اللازمة لتخزين آخر إطار فيديو جرت معالجتهما. يمكن لهذا النظام أن يعمل بتردد أقصاه 250 MHz، علماً أنه يعمل الآن بتردد 25MHz. يحتاج النظام زمناً وقدره 80ms لفك تعمية فيديو بحجم 639Kbytes مكون من 300 إطار بحجم 176×144 بيكسل على نحو كامل ثم يبدأ فك ضغط إطار تلو الإطار وذلك بزمان 2.2ms، لذلك فهو يعمل بالزمن الحقيقي لأن زمن معالجة إطار واحد أقل من زمن

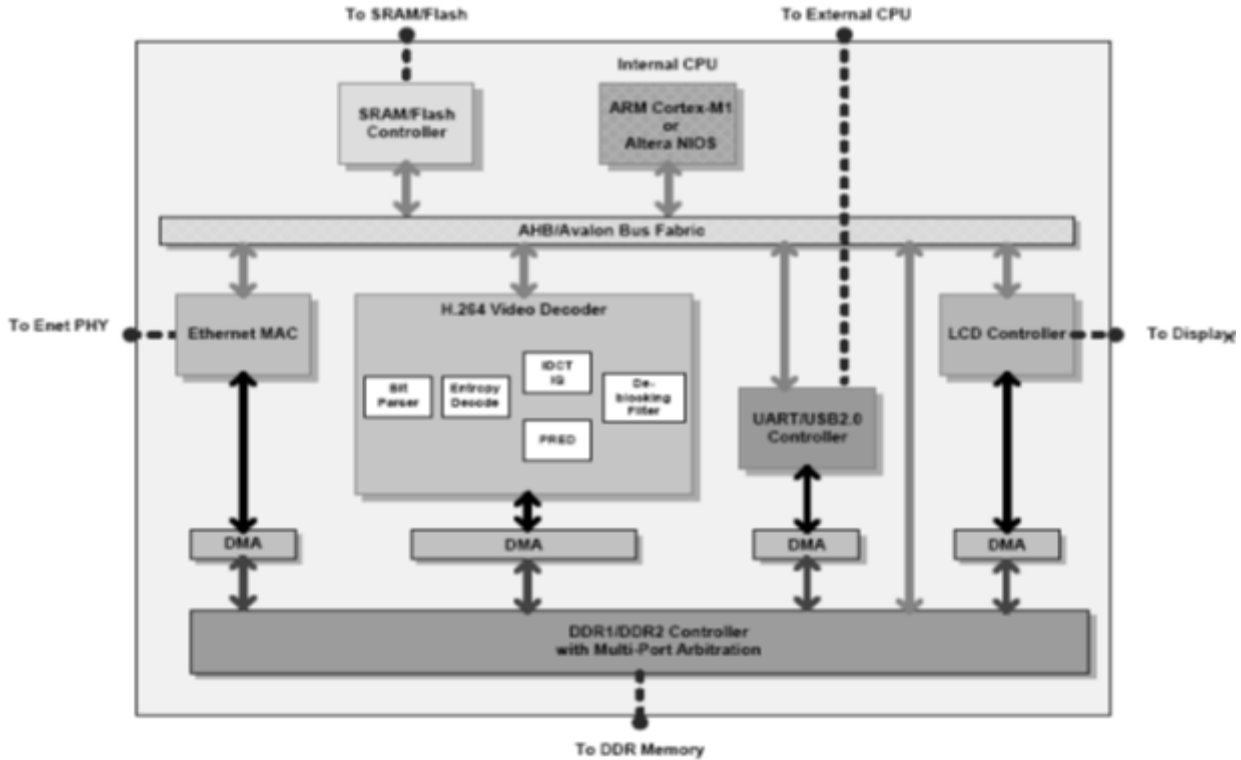
مسح الإطار والذي يعادل 4.6ms من أجل تردد مسح يساوي 25MHz ومن أجل صورة بميز 176×144 بيكسل.

٥-٧- مقارنة النتائج

يمثل مفكك الضغط الجزء الأكثر تعقيداً في هذه الأطروحة، لذلك اعتمدنا في مقارنة نتائجنا على مقارنة تعقيد مفكك الضغط مع ما أمكننا الحصول عليه سواءً في الحقل الأكاديمي أو الحقل التجاري.

تقدم بعض شركات FPGA بنى جاهزة (IP COREs) يمكن شراؤها وتحميلها وإضافتها إلى التصميم كعنصر جاهز أو كمجموعة من ملفات VHDL.

تقدم شركة AsicSoft بنية لمفكك الضغط H.264 [51]، كما هو وارد في الشكل (33.5). وقد جرى توضيحه على شريحة FPGA من نوع Cyclone III من شركة Altera.



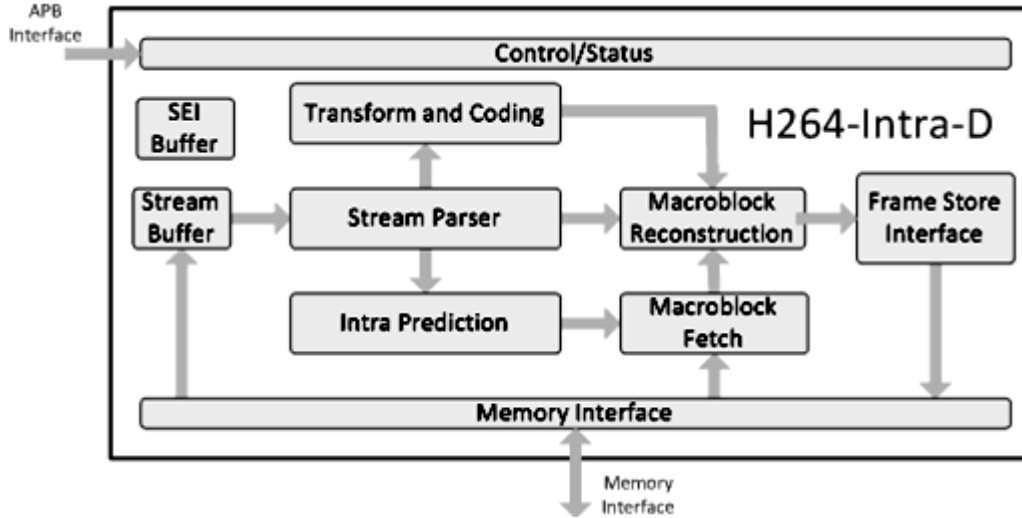
الشكل (33.5): المخطط الصندوقي لمفكك الضغط من شركة AsicSoft [51]

ويظهر الجدول (9.5) تعقيد هذه البنية.

DEVICE	AREA	ON-CHIP RAM	PERFORMANCE
Cyclone III EP3C120	90K Logic Cells	60 Kbits	Fmax = 120 MHz

الجدول (9.5): عدد العناصر المستهلكة لمفكك الضغط من شركة AsicSoft [51]

وتقدم شركة Cast بنية IP Core لمفكك الضغط H264-Intra-D يقوم فقط بتطبيق التنبؤ الداخلي [52]. يظهر الشكل (34.5) المخطط الصندوقي لهذه البنية. تستهلك هذه البنية 50000 عنصر منطقي.



الشكل (34.5): المخطط الصندوقي لمفكك الضغط من شركة Cast [52]

كما وجدنا في الحقل الأكاديمي عدة أمثلة من أهمها ما قام به فريق من جامعة Turku بفنلندا [53] لتصميم مفكك ضغط H.264 ويظهر الجدول (10.5) تعقيد هذا التصميم بأجزائه وبمجمله.

FPGA Block	Logic elements			Memory bits
	Stratix	Stratix II	Cyclone II	
Entropy decode	16	9	14	2600
Reorder	1	1	1	180
Inverse quantization	1199	1151	1163	1150
Inverse transform	4664	4152	4672	500
Intra prediction	58560	37440	56640	4000
Inter prediction	69504	62464	67776	9900
Filter	46848	48832	45056	46000
Total	180792	154049	175322	64330

الجدول (10.5): عدد العناصر المستهلكة لمفكك الضغط من جامعة Turku [53]

نجد بالمقارنة من ناحية عدد العناصر المنطقية أن تعقيد مفكك الضغط في حالتنا (الجدول 6.5) أقل من قرينه في جامعة Turku الفنلندية أو من شركة AsicSoft.

تؤكد الأبحاث في [59] أن خوارزميات ضغط الفيديو تتطلب أزمنة معالجة طويلة نسبياً في حال تحقيقها برمجياً. جرت برمجة خوارزميتي ضغط وفك ضغط الفيديو وفقاً للمعيار H.264 بالاعتماد على مكتبتين مختلفتين لمعالجة الفيديو: JM, Intel IPP. يقدم الجدول (11.5) مقارنة لأزمنة فك ضغط الفيديو من أجل إطار من مقطع فيديو بالمواصفات الآتية: الميزر 176×144 بيكسل وعدد الإطارات = 300. تؤكد هذه المقارنة أهمية استخدام تقانة الكيان الصلب لتنفيذ عملية فك ضغط الفيديو عوضاً عن تنفيذها برمجياً.

Decoding time	
66 ms	Jm software
100 ms	Intel IPP software
2.2 ms	Our design

الجدول (11.5): مقارنة أزمنة فك ضغط الفيديو

الخاتمة والآفاق المستقبلية

تمثل الهدف من هذا البحث في دراسة معيار ضغط الفيديو H.264 ومعيار التعمية AES وتنفيذ الكيان الصلب لنظام يقوم بتعمية وفك تعمية وفك ضغط مقطع فيديو مضغوط ومخزن ضمن ذاكرة باستخدام شريحة FPGA.

جرى لا الاعتماد على طريقة منهجية للوصول إلى النتيجة المطلوبة في هذا المشروع. فقد نُفذت واختُبرت خوارزميات النظام المختلفة على عدة مراحل، وقد جرت برمجتها والتأكد من عملها. جرى بناء الكيان الصلب لمفكك التعمية ومفكك الضغط على شريحة FPGA من نوع ep4ce115 الموجودة على بطاقة التطوير DE2_115. وقمنا باختيار مقطع فيديو مضغوط من المكتبة [56] التي تضم أهم المقاطع المستخدمة لاختبار أنظمة فك الضغط الفيديوي. تم اختيار مقطع مؤلف من 300 إطار وبمميز 176×144 بيكسل، وتخزينه في ذاكرة ROM داخل شريحة FPGA. ثم تعمية المقطع عتادياً بإضافة وحدة تعمية إلى الشريحة نفسها.

وتبيّن أن تعقيد النظام ناتج أساساً عن مفكك ضغط الفيديو وذلك بسبب تعقيد خوارزميته إضافة إلى استهلاكه كمية كبيرة من الذاكر اللازمة لتخزين إطارات الفيديو. علماً أنه بالإمكان استخدام ذواكر خارج شريحة FPGA للتقليل من تعقيد مفكك الضغط غير أن هذا الاستخدام يقلل من أداء مفكك الضغط بسبب زمن الوصول إلى الذاكرة الخارجية. احتاج النظام في هذا التطبيق زمناً وقدره 2.2 ms لفك ضغط إطار واحد. لذلك فهو يعمل بالزمن الحقيقي لأن زمن معالجة إطار واحد أقل من زمن مسح الصورة والذي يبلغ 4.6 ms من أجل تردد مسح يساوي 25 MHz.

تقوم شريحة FPGA بقراءة مقطع فيديو مضغوط ومسجل ضمن ذاكرة ثم تعميته ومن بعدها تفك تعميته وتفك ضغطه لتعرضه على الشاشة. يمكن في المستقبل بناء الكيان الصلب لضغط الفيديو، فيصبح لدينا نظام متكامل مؤلف من:

- مرسل يضغط ويُعمي مقطع فيديوي.
- مستقبل يفك التعمية ويفك الضغط لاستعادة الفيديو.

كما يمكن، في حال توفر بطاقتي تطوير، توضيع المرسل على شريحة FPGA والمستقبل على شريحة أخرى. يجب في هذه الحالة ولغرض الاختبار فقط تعريف طريقة الاتصال بين البطاقتين: سلكية أم لاسلكية. يمكن في البداية تعريف اتصال سلكي بوساطة وصلة إيثرنت Ethernet سلكية ثم الانتقال إلى اتصال لاسلكي بوساطة وصلة إيثرنت Ethernet فضائية. لذلك نقترح من أجل الآفاق المستقبلية ما يأتي:

- التقليل من تعقيد الضاغط ومفكك الضغط باستخدام ذواكر خارجية من أجل تخزين آخر إطارٍ فيديو.
 - بناء الكيان الصلب لضغط الفيديو وفق المعيار H.264.
 - تنفيذ المرسل (ضاغط ومُعمي فيديو) على بطاقة تطوير.
 - تنفيذ المستقبل (مفكك ترميز ومفكك ضغط) على بطاقة تطوير أخرى.
 - تعريف وصلة Ethernet من أجل إنشاء الاتصال بين البطاقتين.
- كما يمكن تزويد بطاقة المرسل بكاميرا رقمية لتقوم بتصوير عدد من المشاهد ثم ترسلها إلى بطاقة المستقبل على مسافة محددة من المرسل.

المراجع References

- [1] <http://pdf.datasheetcatalog.com/datasheets2/12/1264151.pdf>
ITU-T Recommendation H.261, "Video codec for audiovisual services at px64 kb/s", 1993.(retrieved on 10/5/2011)
- [2] I. E. G. Richardson, "Color Spaces in Video Codec Design Developing Image and Video Compression Systems", John Wiley & Sons, 2002.
- [3] https://www.eet-china.com/Embedded_Video_ENG.pdf
D. Katz and R. Gentile. "Fundamentals of Embedded Video Processing", Analog Devices, Inc.(retrieved on 15/2/2012)
- [4] https://www.rgu.ac.uk/files/Sampath_thesis.PDF
C. S. Kannangara, "Complexity Management of H.264/AVC Video Compression", PhD Thesis, The Robert Gordon University, 2006. (retrieved on 10/4/2014)
- [5] <https://www.itu.int/rec/T-REC-H.264>
ITU-T Recommendation H.264, "Advanced video coding for generic audiovisual services", May 2003 .(retrieved on 2/4/2014)
- [6] I. E. G. Richardson, "H.264 and MPEG-4 video compression", John Wiley & Sons, 2003.
- [7] K. R. Rao and P. Yip, "Discrete Cosine Transform: Algorithms, advantages, applications". San Diego, CA: Academic Press, 1990.
- [8] R. C. Gonzalez and R. E. Woods, "Error Free Compression in Digital Image Processing", Prentice Hall, 2002.
- [9] R. Schafer and T. Sikora, "Digital video coding standards and their role in video communications", *Proceedings of the IEEE*, June 1995 Vol 83, No. 6 pp. 907–924,.

[10] Daniele Sannino, "Design of a FPGA based HDTV post-processor", Master's Thesis, February 2010

[11] SamiaSharminShimu, "Performance Analysis of H.264 Encoder for High-definition Video Transmission over Ultra-wideband Communication Link", University of Saskatchewan, 2010

[12] http://en.wikipedia.org/wiki/MPEG-4_Part_2
Standard MPEG-4: ISO/IEC 14496-2, Information Technology – Coding of Audio-Visual Objects.

[13] <http://www.itu.int/rec/T-REC-H.263/en>
ITU-T Recommendation H.263, "Video coding for low bit rate communications", 1998.(retrieved on 2/4/2014)

[14] http://ip.hhi.de/imagecom_G1/assets/h264avc_nutshell.pdf
G. J. Sullivan, "The H.264/MPEG-4 AVC Video Coding Standard and its Deployment Status", SPIE/VCIP 2005, Beijing, China, July 2005.

[15] http://images.apple.com/quicktime/pdf/H264_Technology_Brief.pdf.
Technology Brief: QuickTime and MPEG-4 .(retrieved on 10/4/2014)

[16]http://books.google.com/books/about/Inter_and_Intra_Compensation_Architectur.html
Y.H. Ting, "*Inter and Intra Compensation Architecture Designs for H.264 Video Decoders*", PhD Thesis, National Cheng Kung University, Department of Electrical Engineering, 2005.

[17] <http://books.google.com/books?isbn=054977372X>
Xu, K. "*Power-efficient design methodology for video decoding*", PhD Dissertation, The Chinese University of Hong Kong, 2007.

[18] C.-C. Lin, J.-W Chen, H.-C.Chang, Y.-C.Yang, Y.-H. O. Yang, M.C. Tsai, J.I. Guo, and J.S. Wang, "A 160K gates/4.5 KB SRAM H.264 Video Decoder for HDTV Applications",*IEEE Journal of Solid-State Circuits*, Vol. 42, No. 1, pp. 170-182, 2007.

[19] D. F. Finchelstein, V. Sze, M. E. Sinangil, Y. Koken and A. P. Chandrakasan, "A Low-Power 0.7-V H.264 720p Video Decoder",*IEEE Asian Solid State Circuits Conference*, pp. 173-176, November 2008.

- [20] S. Na, W. Hwangbo, J. Kim, S. Lee, and C. M. Kyung, "1.8mW, Hybrid-Pipeline H.264/AVC Decoder For Mobile Devices", *IEEE Asian Solid State Circuits Conference*, November 2007 Vol.54, No.4.
- [21] E. Fleming, C. C. Lin, N. Dave, Arvindm, G. Raghavan, and J. Hicks, "H.264 Decoder: A Case Study in Multiple Design Points," *Proceedings of Formal Methods and Models for Co-Design*, pp. 165-174, June 2008.
- [22] K. Iwata, S. Mochizuki, T. Shibayama, F. Izuhara, H. Ueda, K. Hosogil, H. Nakatal, M. Ehama, T. Kengaku, T. Nakazawa, and H. Watanabe, "A 256mW Full-HD H.264 High-Profile CODEC Featuring Dual Macroblock-Pipeline Architecture in 65nm CMOS," *Symp. VLSI Circuits Dig. Tech. Papers*, 2008. Vol.20, NO. 9. Page(s): 1495 – 1498.
- [23] <http://dspace.mit.edu/bitstream/handle/1721/219733601.pdf>
C. C. Lin, "Implementation of H.264 Decoder in Bluespec System Verilog," *Master's Thesis, Massachusetts Institute of Technology*, 2007. (retrieved on 11/4/2014)
- [24] V. Chandra, A. Xu, H. Schmit, and L. Pileggi, "An Interconnect Channel Design Methodology for High Performance Integrated Circuits," in *Design, Automation and Test In Europe Conference*, 2004, pp.21138-21143.
- [25] T. C. Wang, Y. W. Huang, H. C. Fang, and L. G. Chen, "Parallel 44 2D transform and inverse transform architecture for MPEG-4 AVC/H.264," *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2003.
- [26] S. Z. Wang, T. A. Lin, T. M. Liu, and C. Y. Lee, "A New Motion Compensation Design for H.264/AVC Decoder," *IEEE International Symposium on Circuits and Systems*, Vol. 5, No. 2, pp. 4558-4561, May 2005.
- [27] H. C. Chang, C. C. Lin, and J. I. Guo, "A Novel Low-Cost High-Performance VLSI Architecture for MPEG-4 AVC/H.264 CAVLC Decoding," *IEEE Symposium on Circuits and Systems*, Vol 6, pp. 6110-6113, May 2005.
- [28] T. H. Tsa, D. L. Fang, and Y. N. Pan, "A Hybrid CAVLD Architecture Design with Low Complexity and Low Power Considerations," *IEEE International Conference on Multimedia and Expo*, pp. 1910-1913, July 2007.
- [29] S. Nomura, "A 9.7mW AAC-Decoding, 620mW H.264 720p 60fps Decoding, 8-Core Media Processor with Embedded Forward-Body-Biasing and Power Gating Circuit in 65nm CMOS Technology ", *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, February 2008.
- [30] E. van der Tol, E. Jaspers, and R. Gelderblom, "Mapping of H.264 Decoding on a Multiprocessor Architecture", *Image and Video Communications and Processing*, 2003. Vol. 16, No. 10, pp. 1321–1334.

[31] V. Sze, D. F. Finchelstein, M. E. Sinangil, and A. P. Chandrakassan, "A 1.8-mW 0.7V H.264 720p Video Decoder" ,*IEEE Journal of Solid-State Circuits*, November 2009.Vol. 8, No.1.

[32] http://www.learningace.com/doc/finchelstein_phd_2008

L. Philipps, S. V. Naimpally, R. Meyer, and S. INoue, "Parallel Architecture for a High Definition Television Video Decoder Having Multiple Independent frame Memories", U.S. Patent No. 5510842, April 2008.

[33] J. Zhao and A. Segall, " VCEG-AI32: New Results using Entropy Slices for Parallel Decoding," ITU-T Study Group, Video Coding Experts Group, 35th Meeting, Germany, July 2008.

[34] C. D. Chien, C. et al. ,"A 252kgate/71mW Multi-Standard Multi-Channel Video Decoder for High Definition Video Applications", *IEEE Int. Solid-State Circuits Conf. Dig. Tech*, 2007, San Francisco,pp 542-621.

[35] M. Budagavi and M. Zhou, "Video Coding using Compressed Reference Frames, " *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2008, Las Vegas. 1165 -1168.

[36] <http://www-mtl.mit.edu/researchgroups/icsystems/theses.html>

D. F. Finchelstein, "Low Power Techniques for Video Decoding", PhD Thesis, MIT, 2009 .(retrieved on 20/1/2014)

[37] B. Schneier, "Applied cryptography: protocols, algorithms, and source code in C". *John Wiley & Sons*, 1996.

[38] http://biblioteca.universia.net/html_bura/id/19904.html

Y. Fan, "Algorithm and Hardware Design of Encryption Scheme for H.264/AVC", Waseda University, 2009.

[39] <http://csrc.nist.gov/archive/aes/round1/conf1/j41ce-rob.pdf>

"U.S. DEPARTMENT OF COMMERCE/National Institute of Standards and Technology. Announcing request for candidate algorithm Nominations for the Advanced Encryption Standard (AES) ", Sep 1997.

[40] <http://csrc.nist.gov/encryption/aes>.

"Advanced Encryption Standard Home Page", 2001 .(retrieved on 20/1/2011)

- [41] <http://www.nist.gov/aes>. Daemen, Joan, and Vincent Rijmen. *The Rijndael Block Cipher.AES Proposal*.
- [42] <http://buchholz.hs-bremen.de>
J. J. Buchholz, Advanced Encryption Standard - AES using MATLAB.
- [43] <http://arxiv.org/pdf/1209.3061>
SounakSamanta., "FPGA Implementation of AES Encryption and Decryption", B.E. III Yr, Electronics & Communication Engg, Sardar Vallabhbhai National Institute of Technology, Surat, 2007.
- [44] S. Arrag, A. Hamdoun, A. Tragha and S. Khamlich, "Design and Implementation A different Architectures of mixcolumn in FPGA", UNIVERSITE HASSAN II MOHAMMEDIA, Casablanca, Morocco.
- [45] <http://www.altera.com> .(retrieved on 15/4/2014)
- [46] F. Vahid, "Digital Design with RTL Design, VHDL, and Verilog", University of California, Riverside, *John Wiley and Sons*, 2011.
- [47] www.altera.com
Altera Corporation. "Quartus II Handbook (Version 9)", November 2009.
- [48] <http://www.terasic.com>
Terasic Technologies Inc, "DE2-115 User Manual", Terasic, 2010.
- [49] <http://www.terasic.com.tw>
Terasic. TRDB_D5M 5 Mega Pixel Digital Camera Development Kit, version 1.0 ed., March 2008.
- [50] <http://www.terasic.com.tw>
Terasic, TRDB-D5M Hardware Specification, version 0.2 ed., June 2009.
- [51] www.asicsoft.com
ASICSoft, Inc. *Atria Logic - Marketing Partner*.
- [52] <http://www.cast-inc.com/ip-cores/video/h264-intra-d/>
CAST Inc, H264_Intra_D,
- [53] T. Lindroth, N. Avessta, J. Teuhola and T. Seceleanu "Complexity Analysis of H.264 Decoder for FPGA Design", University of Turku, Finland, 2006.

- [54] <http://www.di.univr.it/documenti/matdid279434.pdf>
Standard MPEG-1: ISO/IEC 11172-2, "Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to About 1,5 Mb/s".
- [55] http://en.wikipedia.org/wiki/ISO/IEC_JTC_1/SC_29
Standard MPEG-2: ISO/IEC 13818-2, "Information Technology – Generic Coding of Moving Pictures and Associated Audio Information".
- [56] www2.tkn.tu-berlin.de/research/evalvid/cif.html
YUV CIF reference videos (lossless H.264 encoded),
- [57] <http://hardh264.sourceforge.net/H264-encoder-manual.html>
Zexia H.264 Hardware Encoder.
- [58] <http://opencores.org>
- [59] Swaroop, K.V.S. and Rao, KR, "Performance Analysis and Comparison of JM 15.1 and Intel IPP H.264 Encoder and Decoder", IEEE 2010 ,42nd Southeastern Symposium on System Theory (SSST), pp. 371-375.

الملاحق

Appendixes

الملحق A

جداول مفكك الترميز الانتروبي من نوع CAVLC

تُستخدم الجداول الآتية في تفكيك ترميز عدد المعاملات غير الصفرية Total_coeff و عدد المعاملات الواحدة T1، وتفكيك ترميز قيم المعاملات غير الصفرية، إضافة إلى تفكيك ترميز العدد الكلي للأصفار Total_zeros الواقعة قبل آخر معامل غير صفري، و تفكيك ترميز عدد الأصفار Run_before التي تسبق كل معامل غير صفري.

coeff_token mapping to TotalCoeff(coeff_token) and TrailingOnes(coeff_token)

TrailingOnes (coeff_token)	TotalCoeff (coeff_token)	$0 \leq nC < 2$	$2 \leq nC < 4$	$4 \leq nC < 8$	$8 \leq nC$	$nC == -1$	$nC == -2$
0	0	1	11	1111	0000 11	01	1
0	1	0001 01	0010 11	0011 11	0000 00	0001 11	0001 111
1	1	01	10	1110	0000 01	1	01
0	2	0000 0111	0001 11	0010 11	0001 00	0001 00	0001 110
1	2	0001 00	0011 1	0111 1	0001 01	0001 10	0001 101
2	2	001	011	1101	0001 10	001	001
0	3	0000 0011 1	0000 111	0010 00	0010 00	0000 11	0000 0011 1
1	3	0000 0110	0010 10	0110 0	0010 01	0000 011	0001 100
2	3	0000 101	0010 01	0111 0	0010 10	0000 010	0001 011
3	3	0001 1	0101	1100	0010 11	0001 01	0000 1
0	4	0000 0001 11	0000 0111	0001 111	0011 00	0000 10	0000 0011 0
1	4	0000 0011 0	0001 10	0101 0	0011 01	0000 0011	0000 0010 1
2	4	0000 0101	0001 01	0101 1	0011 10	0000 0010	0001 010
3	4	0000 11	0100	1011	0011 11	0000 000	0000 01
0	5	0000 0000 111	0000 0100	0001 011	0100 00	-	0000 0001 11
1	5	0000 0001 10	0000 110	0100 0	0100 01	-	0000 0001 10
2	5	0000 0010 1	0000 101	0100 1	0100 10	-	0000 0010 0
3	5	0000 100	0011 0	1010	0100 11	-	0001 001
0	6	0000 0000 0111 1	0000 0011 1	0001 001	0101 00	-	0000 0000 111
1	6	0000 0000 110	0000 0110	0011 10	0101 01	-	0000 0000 110

جدول (1.A): ترميز الزوج (Total_coeff, T1) [6]

ملاحظة : يظهر في المربع المحدد قيمة الترميز لعدد المعاملات غير الصفرية Total_coeff وعدد المعاملات الواحدية T1 الموافقة للمثال الوارد في الفقرة ٣-٣-٣.

total_zeros tables for 4x4 blocks with TotalCoeff(coeff_token) 1 to 7

total_zeros	TotalCoeff(coeff_token)						
	1	2	3	4	5	6	7
0	1	111	0101	0001 1	0101	0000 01	0000 01
1	011	110	111	111	0100	0000 1	0000 1
2	010	101	110	0101	0011	111	101
3	0011	100	101	0100	111	110	100
4	0010	011	0100	110	110	101	011
5	0001 1	0101	0011	101	101	100	11
6	0001 0	0100	100	100	100	011	010
7	0000 11	0011	011	0011	011	010	0001
8	0000 10	0010	0010	011	0010	0001	001
9	0000 011	0001 1	0001 1	0010	0000 1	001	0000 00
10	0000 010	0001 0	0001 0	0001 0	0001	0000 00	
11	0000 0011	0000 11	0000 01	0000 1	0000 0		
12	0000 0010	0000 10	0000 1	0000 0			
13	0000 0001 1	0000 01	0000 00				
14	0000 0001 0	0000 00					
15	0000 0000 1						

جدول (2.A): ترميز total_zeros [6]

ملاحظة : يظهر في المربع المحدد قيمة الترميز لقيم المعاملات غير الصفرية الموافقة للمثال الوارد في الفقرة ٣-٣-٣.

Tables for run_before

run_before	zerosLeft						
	1	2	3	4	5	6	>6
0	1	1	11	11	11	11	111
1	0	01	10	10	10	000	110
2	-	00	01	01	011	001	101
3	-	-	00	001	010	011	100
4	-	-	-	000	001	010	011
5	-	-	-	-	000	101	010
6	-	-	-	-	-	100	001
7	-	-	-	-	-	-	0001
8	-	-	-	-	-	-	00001
9	-	-	-	-	-	-	000001
10	-	-	-	-	-	-	0000001
11	-	-	-	-	-	-	00000001
12	-	-	-	-	-	-	000000001
13	-	-	-	-	-	-	0000000001
14	-	-	-	-	-	-	00000000001

جدول (3.A): ترميز run_before [6]

ملاحظة: يظهر في المربعات المُحددة قيم الترميز للعدد الكلي للأصفار Total_zeros الواقعة قبل آخر معامل غير صفري الموافقة للمثال الوارد في الفقرة 3-3-3.

■ Level VLC 0
(Unary code)

Level	Code
1	1
-1	01
2	001
-2	0001
3	00001
-3	000001
..	..
-7	000000000000001
±8 to ±15	000000000000001xxxx
±16 ->	000000000000001xxxxxx xxxxxx

Level VLC 1
(Golomb-Rice(2) Code)

Level	Code
1	10
-1	11
2	010
-2	011
3	0010
-3	0011
..	..
14	0000000000000010
-14	0000000000000011
±15 to ±22	000000000000001x xxx
±23 ->	000000000000001x xxxxxxxxxxx

جدول (4.A): ترميز العناصر غير الصفريّة [6]

ملاحظة : يظهر في المربعات المُحددة قيم الترميز لعدد الأصفار Run_before التي تسبق كل معامل غير صفري الموافقة للمثال الوارد في الفقرة ٣-٣-٣.

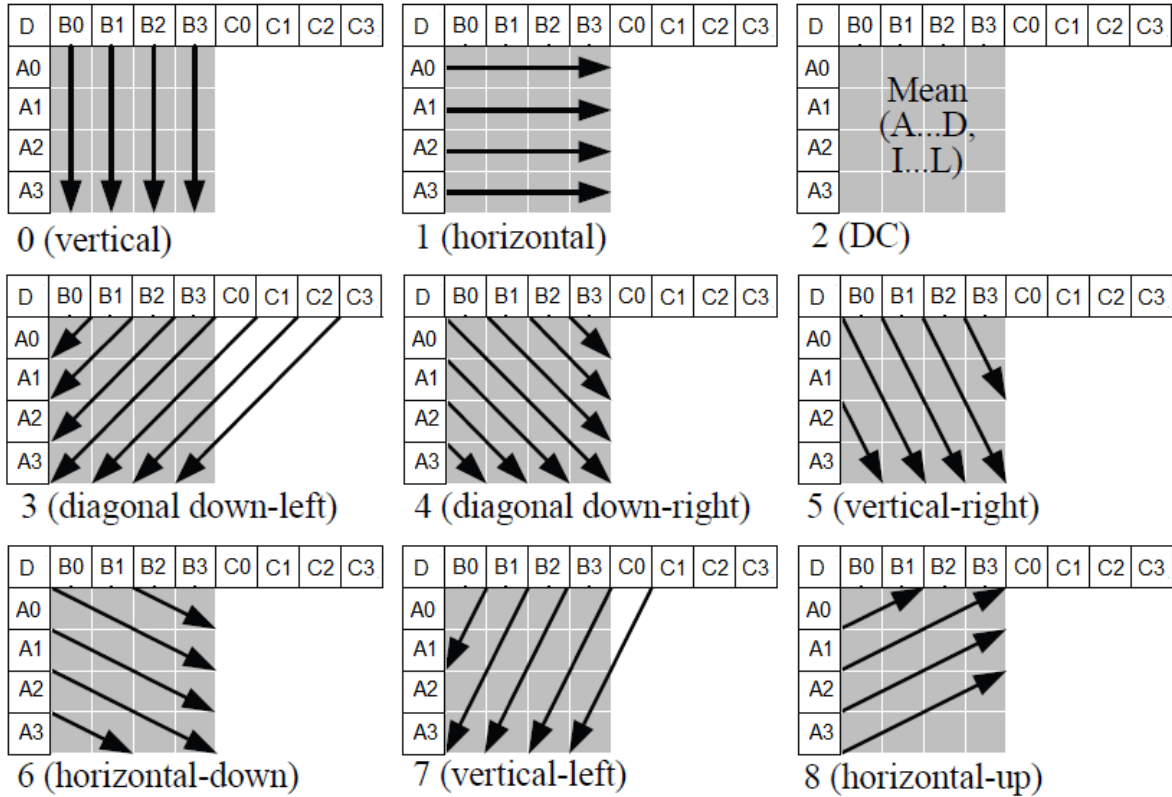
الملحق B

وحدة التنبؤ الداخلي Intra Prediction unit بلغة VHDL

تقوم هذه الوحدة بالتنبؤ اعتماداً على المركبات المجاورة:

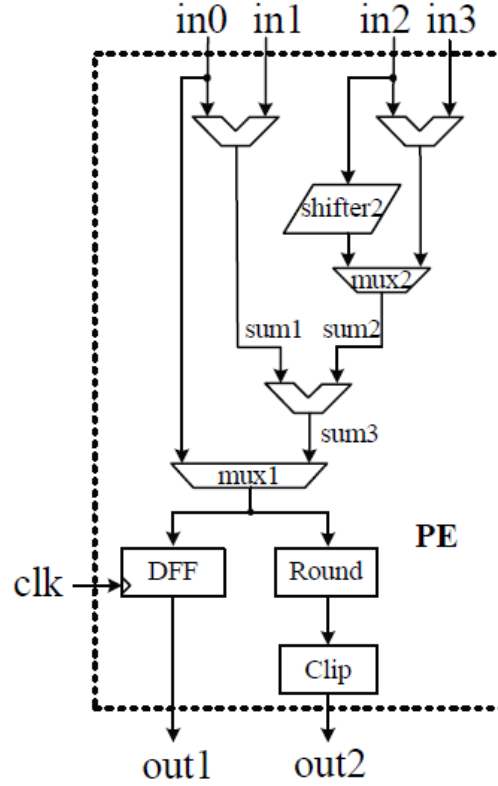
A0, A1, ..A3, B0, B1, .. B3, C0, C1, ..C3, D

بكتلة إضاءة بقياس 4×4 وفقاً لتسعة أنماط كما هو مبين في الشكل (1.B).



الشكل (1.B): أنماط التنبؤ الداخلي لكثلة إضاءة بقياس 4×4 [16]

يعتمد التنبؤ على تطبيق مرشح من نوع FIR له 3 معاملات: $(\frac{1}{4}, \frac{1}{2}, \frac{1}{4})$ أو على التكرار أو على التوسيط. يوضح الشكل (2.B) بنية وحدة المعالجة التي تسمح بتطبيق الترشيح أو التكرار أو التوسيط.



الشكل (2.B): بنية وحدة المعالجة في التنبؤ الداخلي [16]

تنفيذ وحدة المعالجة في التنبؤ الداخلي بلغة VHDL

```

component PE is port(
  clk,reset_n: in std_logic;
  in0,in1,in2,in3: in std_logic_vector(15 downto 0);
  IsShift,IsStore,IsClip,full_bypass: in std_logic;
  round_value: in std_logic_vector(4 downto 0);
  shift_len: in std_logic_vector(2 downto 0);
  PE_out_reg: out std_logic_vector(15 downto 0);
  PE_out: out std_logic_vector(7 downto 0);
  sum_out:outstd_logic_vector(15 downto 0))
end component;
    
```

تسمح لنا إشارات الدخل:

`IsShift, IsStore, IsClip, full_bypass, round_value, shift_len`

بإعادة ضبط وحدة المعالجة بحسب نمط التنبؤ.

`IsShift`: تسمح هذه الإشارة عند تفعيلها بتطبيق إزاحة بطول `shift_len` على المدخل `in2`.

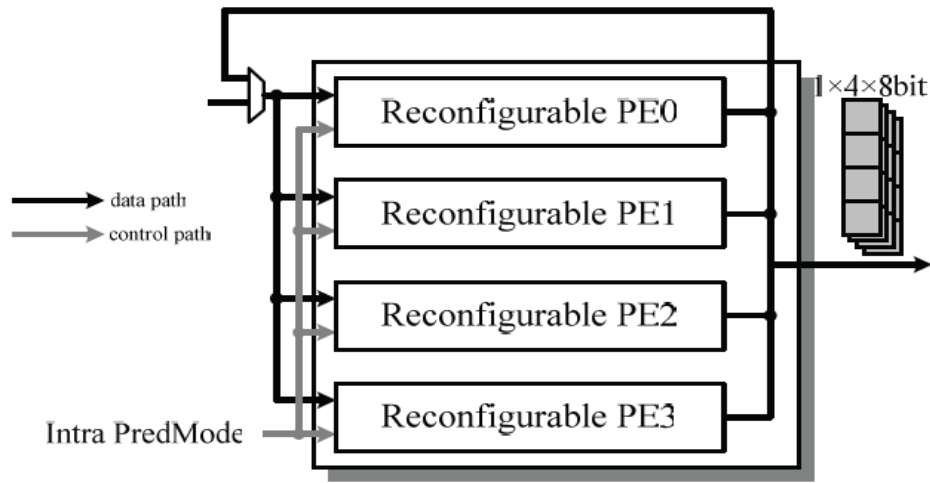
`IsStore`: تسمح هذه الإشارة عند تفعيلها بتخزين نتيجة خرج الناخب `mux1`.

`IsClip`: تسمح هذه الإشارة عند تفعيلها بتطبيق التقريب على خرج الناخب `mux1` بقيمة `round_value`.

`full_bypass`: تسمح هذه الإشارة عند تفعيلها باختيار المدخل `in0` على خرج الناخب `mux1`.

جرى استخدام أربع وحدات معالجة من أجل تسريع المعالجة، فيكون خرج وحدة التنبؤ على شكل شعاع مؤلف من أربع عناصر وهي:

`PE0_out, PE1_out, PE2_out, PE3_out`



الشكل (3.B): خرج وحدة التنبؤ الداخلي [16]

يجري حساب الكتلة بقياس 4×4 بواسطة أربع حلقات (4 cycles) حيث يجري إخراج عمود بقياس 1×4 في كل حلقة.

فمثلاً، يجري في حالة التنبؤ العمودي Vertical ومن أجل جميع وحدات المعالجة:

- تكرار القيمة المجاورة العلوية B0 في الحلقة الأولى
(PE0_out=PE0_in=B0, PE1_out=PE1_in=B0, ...).
- تكرار القيمة المجاورة العلوية B1 في الحلقة الثانية:
(PE0_out=PE0_in=B1, PE1_out=PE1_in=B1, ...).
- وهكذا

```
when Intra4x4_Vertical =>
case (blk4x4_intra_calculate_counter) is
when "100" =>
    PE0_in0 <= B0;
    PE1_in0 <= B0;
    PE2_in0 <= B0;
    PE3_in0 <= B0;
when "011" =>
    PE0_in0 <= B1;
    PE1_in0 <= B1;
    PE2_in0 <= B1;
    PE3_in0 <= B1;
when "010" =>
    PE0_in0 <= B2;
    PE1_in0 <= B2;
    PE2_in0 <= B2;
    PE3_in0 <= B2;
when "001" =>
    PE0_in0 <= B3;
    PE1_in0 <= B3;
    PE2_in0 <= B3;
    PE3_in0 <= B3;
```

```

when others =>

    PE0_in0 <= (others => '0');

    PE1_in0 <= (others => '0');

    PE2_in0 <= (others => '0');

    PE3_in0 <= (others => '0');

end case;

PE0_in1 <= (others => '0');      PE0_in2 <= (others => '0');

PE0_in3 <= (others => '0');

PE0_IsShift <= '0';      PE0_IsStore <= '0';      PE0_IsClip <= '0';

PE0_full_bypass <= '1';  PE0_round_value <= (others => '0');

PE0_shift_len <= (others => '0');

PE1_in1 <= (others => '0'); PE1_in2 <= (others => '0');

PE1_in3 <= (others => '0');

PE1_IsShift <= '0'; PE1_IsStore <= '0';      PE1_IsClip <= '0';

PE1_full_bypass <= '1';  PE1_round_value <= (others => '0');

PE1_shift_len <= (others => '0');

PE2_in1 <= (others => '0');      PE2_in2 <= (others => '0');

PE2_in3 <= (others => '0');

PE2_IsShift <= '0'; PE2_IsStore <= '0';      PE2_IsClip <= '0';

PE2_full_bypass <= '1';  PE2_round_value <= (others => '0');

PE2_shift_len <= (others => '0');

PE3_in1 <= (others => '0');      PE3_in2 <= (others => '0');

PE3_in3 <= (others => '0');

PE3_IsShift <= '0';      PE3_IsStore <= '0'; PE3_IsClip <= '0';

PE3_full_bypass <= '1';  PE3_round_value <= (others => '0');

PE3_shift_len <= (others => '0');

```

أما في حالة التنبؤ الأفقي Horizontal، فيجري في جميع الحلقات:

- تكرار القيمة المجاورة اليسارية A0 من أجل وحدة المعالجة PE0:
 (PE0_full_bypass=1, PE0_in=PE0_out=A0).

- تكرار القيمة المجاورة اليسارية A1 من أجل وحدة المعالجة PE1:
(PE1_full_bypass=1, PE1_in=PE1_out=A1).
- وهكذا.

```

when Intra4x4_Horizontal =>
if (blk4x4_intra_calculate_counter = "000") then
    PE0_in0 <= (others => '0');
    PE1_in0 <= (others => '0');
    PE2_in0 <= (others => '0');
    PE3_in0 <= (others => '0');
else
    PE0_in0 <= A0;
    PE1_in0 <= A1;
    PE2_in0 <= A2;
    PE3_in0 <= A3;
end if;
PE0_in1 <= (others => '0');      PE0_in2 <= (others => '0');
PE0_in3 <= (others => '0');
PE0_IsShift <= '0';      PE0_IsStore<= '0';      PE0_IsClip <= '0';
PE0_full_bypass <= '1';  PE0_round_value <= (others => '0');
PE0_shift_len <= (others => '0');
PE1_in1 <= (others => '0');      PE1_in2 <= (others => '0');
PE1_in3 <= (others => '0');
PE1_IsShift <= '0';      PE1_IsStore <= '0';      PE1_IsClip <= '0';
PE1_full_bypass <= '1';  PE1_round_value <= (others => '0');
PE1_shift_len <= (others => '0');
PE2_in1 <= (others => '0');      PE2_in2 <= (others => '0');
PE2_in3 <= (others => '0');
PE2_IsShift <= '0';      PE2_IsStore <= '0'; PE2_IsClip <= '0';

```

```

PE2_full_bypass <= '1';    PE2_round_value <= (others => '0');

PE2_shift_len <= (others => '0');

PE3_in1 <= (others => '0');    PE3_in2 <= (others => '0');

PE3_in3 <= (others => '0');

PE3_IsShift <= '0';    PE3_IsStore <= '0';    PE3_IsClip <= '0';

PE3_full_bypass <= '1';    PE3_round_value <= (others => '0');

PE3_shift_len <= (others => '0');

```

وفي حالة التنبؤ من نمط diagonal down-right:

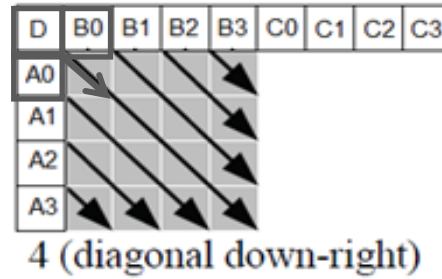
- تكون في الحلقة الأولى موسطات وحدة المعالجة الأولى كما يلي:

PE0_full_bypass=0,

PE0_IsShift=1 (ضرب الدخل PE0_in2 بالقيمة 2),

PE0_shift_len=2 (تقسيم الخرج على 4)

⇒ PE0_out=(A0 + 2D + B0)/4 (خرج وحدة المعالجة الأولى)



الشكل (4.B): موسطات وحدة المعالجة الأولى في الحلقة الأولى

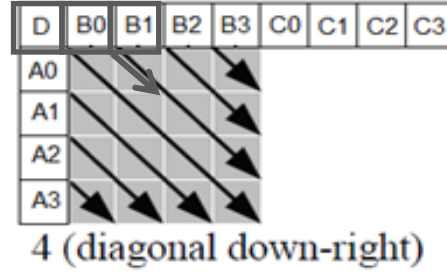
بنفس الطريقة يكون في الحلقة الأولى:

خرج وحدة المعالجة الثانية: PE1_out=(A1 + 2A0 + D)/4

- خرج وحدة المعالجة الثالثة: PE2_out=(A2 + 2A1 + A0)/4

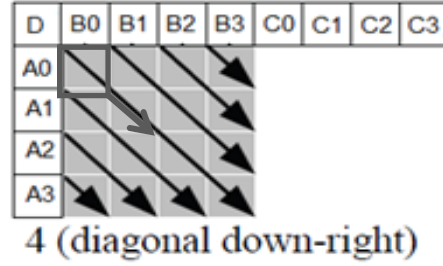
- خرج وحدة المعالجة الرابعة: PE3_out=(A3 + 2A2 + A1)/4

أما في الحلقة الثانية فيكون خرج وحدة المعالجة الأولى: PE0_out=(B1 + 2B0 + D)/4



الشكل (5.B): موسطات وحدة المعالجة الأولى في الحلقة الثانية

ويكون خرج الوحدات الأخرى في الحلقة الثانية بتكرار القيم التي سبق حسابها. فمثلاً خرج وحدة المعالجة الثانية هو: $PE1_in = blk4 \times 4_0 = PE1_out$ ($PE1_full_bypass = 1$). حيث تمثل $blk4 \times 4_0$ قيمة مركبة الإضاءة المحسوبة في الحلقة الأولى من الكتلة 4×4 في الموقع $(0,0)$.



الشكل (6.B): موسطات وحدة المعالجة الثانية في الحلقة الأولى

when Intra4x4_Diagonal_Down_Right =>

case (blk4x4_intra_calculate_counter) is

when "100" =>

PE0_in0 <= B0; PE0_in1 <= A0; PE0_in2 <= D;

PE1_in0 <= D; PE1_in1 <= A1; PE1_in2 <= A0;

PE2_in0 <= A0; PE2_in1 <= A2; PE2_in2 <= A1;

PE3_in0 <= A1; PE3_in1 <= A3; PE3_in2 <= A2;

PE1_full_bypass <= '0';

PE2_full_bypass <= '0';

PE3_full_bypass <= '0';

when "011" =>

PE0_in0 <= D; PE0_in1 <= B1; PE0_in2 <= B0;

```

PE1_in0 <= blk4x4_0;

PE2_in0 <= blk4x4_4;

PE3_in0 <= blk4x4_8;

PE1_full_bypass <= '1';

PE2_full_bypass <= '1';

PE3_full_bypass <= '1';

when "010" =>

    PE0_in0 <= B0; PE0_in1 <= B2; PE0_in2 <= B1;

    PE1_in0 <= blk4x4_1;

    PE2_in0 <= blk4x4_0;

    PE3_in0 <= blk4x4_4;

    PE1_full_bypass <= '1';

    PE2_full_bypass <= '1';

    PE3_full_bypass <= '1';

when "001" =>

    PE0_in0 <= B1; PE0_in1 <= B3; PE0_in2 <= B2;

    PE1_in0 <= blk4x4_2;

    PE2_in0 <= blk4x4_1;

    PE3_in0 <= blk4x4_0;

    PE1_full_bypass <= '1';

    PE2_full_bypass <= '1';

    PE3_full_bypass <= '1';

when others =>

    PE0_in0 <= (others => '0'); PE0_in1 <= (others => '0');

    PE0_in2 <= (others => '0');

    PE1_in0 <= (others => '0');

    PE2_in0 <= (others => '0');

    PE3_in0 <= (others => '0'); PE1_full_bypass <= '1';

    PE2_full_bypass <= '1'; PE3_full_bypass <= '1'; end case;

```

```
if (blk4x4_intra_calculate_counter="000") then
```

```
    PE0_IsShift <= '0';
    PE0_round_value <= "00000";
    PE0_shift_len    <= "000";
    PE1_IsShift <= '0';
    PE1_round_value <= "00000";
    PE1_shift_len    <= "000";
    PE2_IsShift <= '0';
    PE2_round_value <= "00000";
    PE2_shift_len    <= "000";
    PE3_IsShift <= '0';
    PE3_round_value <= "00000";
    PE3_shift_len    <= "000";
```

```
else
```

```
    PE0_IsShift <= '1';
    PE0_round_value <= "00010"; -- +2
    PE0_shift_len    <= "010"; -->>2

    PE1_IsShift <= '1';
    PE1_round_value <= "00010";
    PE1_shift_len    <= "010";

    PE2_IsShift <= '1';
    PE2_round_value <= "00010";
    PE2_shift_len    <= "010";

    PE3_IsShift <= '1';
    PE3_round_value <= "00010";
    PE3_shift_len    <= "010";
```

```
end if;
```

```
PE0_in3 <= (others => '0');  
PE0_IsStore <= '0'; PE0_IsClip <= '0';      PE0_full_bypass <= '0';  
PE1_in3 <= (others => '0');  
PE1_IsStore <= '0'; PE1_IsClip <= '0';  
PE2_in3 <= (others => '0');  
PE2_IsStore <= '0'; PE2_IsClip <= '0';  
PE3_in3 <= (others => '0');  
PE3_IsStore <= '0'; PE3_IsClip <= '0';
```

الملحق C

وحدة التنبؤ البيني للمركبات الضوئية Luma Inter Prediction unit بلغة VHDL

تقوم وحدة التنبؤ البيني، كما هو موضح بالشكل (1.C)، بإجراء استقراء رياضي انطلاقاً من البيكسلات الصحيحة مثل:

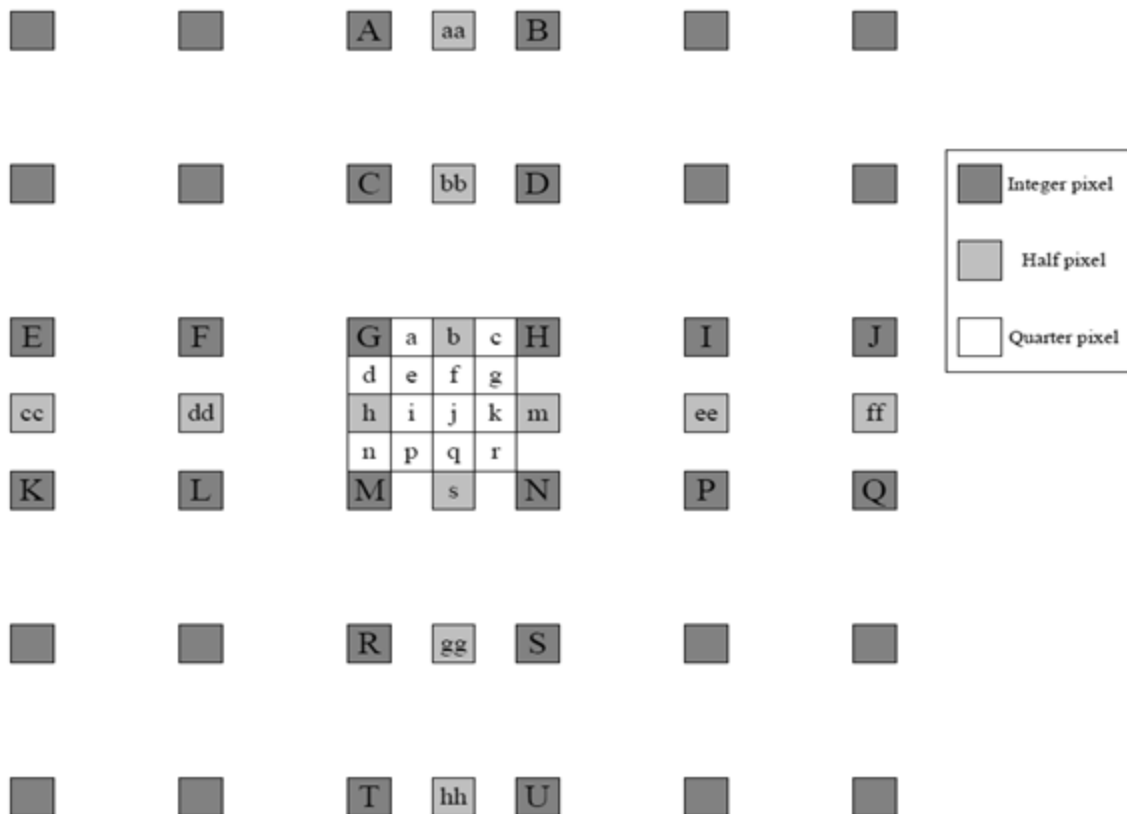
A, B, C, D, E, F, G, ..., T, U

للحصول على البيكسلات بميز 1/2 بيكسل مثل:

aa, bb, cc, dd, ee, ..., b, h, s, m

والبيكسلات بميز 1/4 بيكسل مثل:

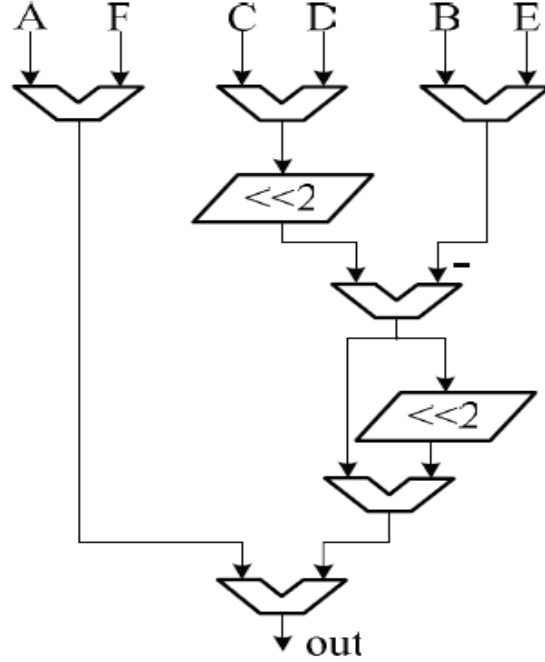
.a, c, d, e, ..., p, q, r



الشكل(1.C): استقراء مركبات الإضاءة بميز 1/2 و 1/4 بيكسل [6]

يعتمد الاستقراء الرياضي على تطبيق مرشح ذو استجابة محدودة FIR له 6 معاملات:

1, -5, 20, 20, -5, 1. يوضح الشكل (2.C) بنية هذا المرشح.



الشكل(2.C): بنية المرشح FIR [6]

يمكن إعادة صياغة معادلة المرشح للاستغناء عن استخدام الضواري كما يلي:

$$\text{out} = (A+F) - 5 \times (B+E) + 20 \times (C+D) = (A+F) - 4 \times (B+E) + 16 \times (C+D) - (B+E) + 4 \times (C+D)$$

وبالتالي يمكن تشكيل المعادلة اعتماداً على عمليات الجمع والطرح والإزاحة.

نقوم باستخدام عدد من المرشحات من أجل تسريع عملية التنبؤ، فيكون خرج وحدة التنبؤ على شكل شعاع بطول أربع عناصر: LPE0_out, LPE1_out, LPE2_out, LPE3_out

إذا كان العنصر المطلوب هو مركبة بميزر 1/2 بيكسل بين سطرين مثل المركبتين cc, dd (الشكل 1.C)، فيكفي تطبيق المرشح FIR بشكل عمودي على المداخل:

Inter_V_0, Inter_V_1, ..., Inter_V_5, ...

أما إذا كان العنصر المطلوب هو مركبة بميزر 1/2 بيكسل بين عامودين مثل المركبتين aa, bb (الشكل 1.C)، فيكفي تطبيق المرشح FIR بشكل عمودي على المداخل:

Inter_H_0_0,Inter_H_1_0,Inter_H_2_0,Inter_H_3_0,Inter_H_4_0,Inter_H_5_0,..

أما إذا كان العنصر المطلوب هو مركبة بين عامودين وسطرين مثل المركبة z (الشكل 1.C)، فيجب تطبيق مرحلتي ترشيح أفقية وعمودية:

يجري في المرحلة الأولى تطبيق 9 مرشحات أفقية وكل مرشح له ٦ مداخل وخرج وحيد:
مدخل المرشح الأفقي الأول:

Inter_H_0_0,Inter_H_1_0,Inter_H_2_0,Inter_H_3_0,Inter_H_4_0,Inter_H_5_0

وخرجه:

H_6tapfilter0_raw_out

مدخل المرشح الأفقي الثاني:

Inter_H_0_1,Inter_H_1_1,Inter_H_2_1,Inter_H_3_1,Inter_H_4_1,Inter_H_5_1

وخرجه:

H_6tapfilter1_raw_out

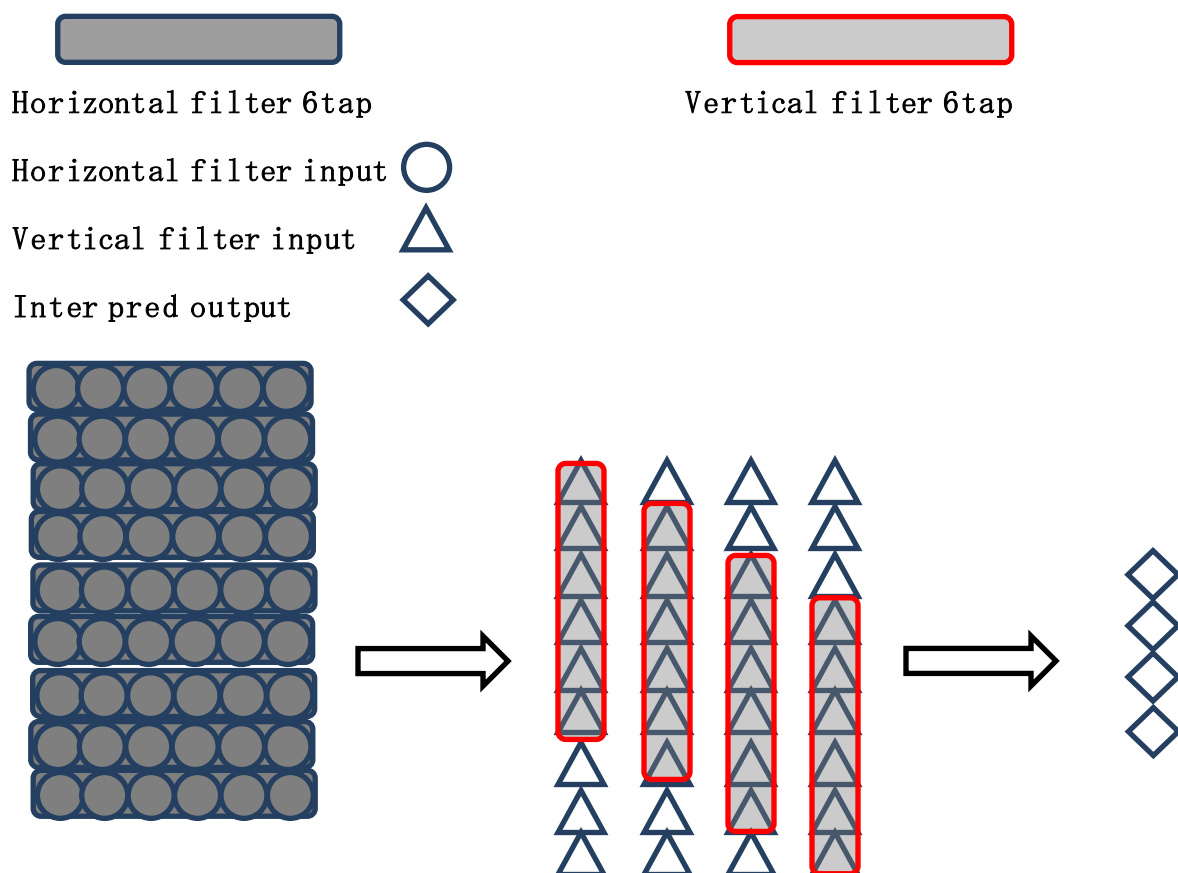
الخ. فنحصل على 9 مخارج وهي:

H_6tapfilter0_raw_out, H_6tapfilter1_raw_out,...,
H_6tapfilter8_raw_out

نقوم بإدخالها في المرحلة الثانية على أربعة مرشحات عمودية كما يلي:

نقوم بإدخال أول 6 عناصر من مخارج المرحلة الأولى إلى المرشح العمودي الأول، ثم نقوم بإدخال ثاني 6 عناصر من مخارج المرحلة الأولى إلى المرشح العمودي الثاني وهكذا. فنحصل في النهاية على شعاع الخرج المؤلف من أربعة عناصر.

أما للحصول على المركبات بميزر $\frac{1}{4}$ بيكسل فلما أن نطبق مرشح توسيطي bilinear بين مركبة صحيحة ومركبة بميزر $\frac{1}{2}$ بيكسل فنحصل على مركبة مثل المركبة a (الشكل 1.C) أو بين مركبتين بميزر $\frac{1}{2}$ بيكسل فنحصل على مركبة مثل المركبة e (الشكل 1.C).



الشكل (3.C): مراحل الحصول على خرج التنبؤ البيني

تنفيذ وحدة التنبؤ البيني بلغة VHDL

```
entity Inter_pred_L is
port
(clk, reset_n, IsInterLuma: in std_logic;
pos_FracL, blk4x4_inter_calculate_counter: in std_logic_vector(3 downto 0);
--block 9x6
Inter_H_0_0, Inter_H_1_0, Inter_H_2_0, Inter_H_3_0, Inter_H_4_0, Inter_H_5_0,
Inter_H_0_1, Inter_H_1_1, Inter_H_2_1, Inter_H_3_1, Inter_H_4_1, Inter_H_5_1,
Inter_H_0_2, Inter_H_1_2, Inter_H_2_2, Inter_H_3_2, Inter_H_4_2, Inter_H_5_2,
Inter_H_0_3, Inter_H_1_3, Inter_H_2_3, Inter_H_3_3, Inter_H_4_3, Inter_H_5_3,
Inter_H_0_4, Inter_H_1_4, Inter_H_2_4, Inter_H_3_4, Inter_H_4_4, Inter_H_5_4,
```



```

Inter_H_0_5,Inter_H_1_5,Inter_H_2_5,Inter_H_3_5,Inter_H_4_5,Inter_H_5_5,
Inter_H_0_6,Inter_H_1_6,Inter_H_2_6,Inter_H_3_6,Inter_H_4_6,Inter_H_5_6,
Inter_H_0_7,Inter_H_1_7,Inter_H_2_7,Inter_H_3_7,Inter_H_4_7,Inter_H_5_7,
Inter_H_0_8,Inter_H_1_8,Inter_H_2_8,Inter_H_3_8,Inter_H_4_8,Inter_H_5_8,
Inter_V_0,Inter_V_1,Inter_V_2,Inter_V_3,Inter_V_4,
Inter_V_5,Inter_V_6,Inter_V_7,Inter_V_8,
Inter_bi_0,Inter_bi_1,Inter_bi_2,Inter_bi_3: in std_logic_vector(7 downto 0);
LPE0_out,LPE1_out,LPE2_out,LPE3_out: out std_logic_vector(7 downto 0)
);end entity;

architecturearchi of Inter_pred_L is
begin
    H_6tapfilter0: filterH_6tap
    port map(
        A => Inter_H_0_0,B => Inter_H_1_0,C => Inter_H_2_0,D => Inter_H_3_0,E => Inter_H_4_0,
        F => Inter_H_5_0,H_need_round => '0',raw_out => H_6tapfilter0_raw_out,
        round_out => H_6tapfilter0_round_out);

    H_6tapfilter1: filterH_6tap
    port map(
        A => Inter_H_0_1,B => Inter_H_1_1,C => Inter_H_2_1,D => Inter_H_3_1,E => Inter_H_4_1,
        F => Inter_H_5_1,H_need_round => '0',raw_out => H_6tapfilter1_raw_out,
        round_out => H_6tapfilter1_round_out);

    .....

    H_6tapfilter8: filterH_6tap
    port map(
        A => Inter_H_0_8,B => Inter_H_1_8,C => Inter_H_2_8,D => Inter_H_3_8,E => Inter_H_4_8,
        F => Inter_H_5_8,H_need_round => '0',raw_out => H_6tapfilter8_raw_out,
        round_out => H_6tapfilter8_round_out);

    process (clk, reset_n)begin
        if (reset_n = '0') then

```

```

b0_raw_reg <= (others => '0'); b1_raw_reg <= (others => '0');
...
b8_raw_reg <= (others => '0');

elsif (rising_edge(clk)) then
if (b_raw_reg_ena='1') then
b0_raw_reg <= H_6tapfilter0_raw_out;b1_raw_reg <= H_6tapfilter1_raw_out;
b2_raw_reg <= H_6tapfilter2_raw_out;b3_raw_reg <= H_6tapfilter3_raw_out;
b4_raw_reg <= H_6tapfilter4_raw_out;b5_raw_reg <= H_6tapfilter5_raw_out;
b6_raw_reg <= H_6tapfilter6_raw_out;b7_raw_reg <= H_6tapfilter7_raw_out;
b8_raw_reg <= H_6tapfilter8_raw_out;
end if;end if;end process;

filterV_6tap0: filterV_6tap
port map(
A => V_6tapfilter0_A,B => V_6tapfilter0_B,C => V_6tapfilter0_C,D => V_6tapfilter0_D,
E => V_6tapfilter0_E,F => V_6tapfilter0_F,Is_jfqik =>Is_V_jfqik,
round_out=> V_6tapfilter0_round_out);
.....
filterV_6tap3: filterV_6tap
port map(
A => V_6tapfilter3_A,B => V_6tapfilter3_B,C => V_6tapfilter3_C,D => V_6tapfilter3_D,
E => V_6tapfilter3_E,F => V_6tapfilter3_F,Is_jfqik =>Is_V_jfqik,
round_out=> V_6tapfilter3_round_out);

V_6tapfilter0_A <= b0_raw_reg when (Is_V_jfqik='1') else
"0000000"&Inter_V_0;
V_6tapfilter0_B <= b1_raw_reg when (Is_V_jfqik='1') else
"0000000"&Inter_V_1;
...
V_6tapfilter0_F <= b5_raw_reg when (Is_V_jfqik='1') else
"0000000"&Inter_V_5;
V_6tapfilter1_A <= b1_raw_reg when (Is_V_jfqik='1') else

```

```

"0000000"&Inter_V_1;

V_6tapfilter1_B <= b2_raw_reg when (Is_V_jfqik='1') else

"0000000"&Inter_V_2;

...

V_6tapfilter1_F <= b6_raw_reg when (Is_V_jfqik='1') else

"0000000"&Inter_V_6;

V_6tapfilter2_A <= b2_raw_reg when (Is_V_jfqik='1') else

"0000000"&Inter_V_2;

V_6tapfilter2_B <= b3_raw_reg when (Is_V_jfqik='1') else

"0000000"&Inter_V_3;

...

V_6tapfilter2_F <= b7_raw_reg when (Is_V_jfqik='1') else

"0000000"&Inter_V_7;

V_6tapfilter3_A <= b3_raw_reg when (Is_V_jfqik='1') else

"0000000"&Inter_V_3;

V_6tapfilter3_B <= b4_raw_reg when (Is_V_jfqik='1') else

"0000000"&Inter_V_4;

...

V_6tapfilter3_F <= b8_raw_reg when (Is_V_jfqik='1') else

"0000000"&Inter_V_8;

bilinear0: bilinear
port map(
A => bilinear0_A,B => bilinear0_B,bilinear_out => bilinear0_out);

bilinear1: bilinear
port map(
A => bilinear1_A,B => bilinear1_B,bilinear_out => bilinear1_out);\

bilinear2: bilinear
port map(
A => bilinear2_A,B => bilinear2_B,bilinear_out => bilinear2_out);

bilinear3: bilinear

```

```

port map(A => bilinear3_A,B => bilinear3_B,bilinear_out => bilinear3_out);

process (IsInterLuma, pos_FracL, blk4x4_inter_calculate_counter, V_6tapfilter0_round_out,
V_6tapfilter1_round_out, V_6tapfilter2_round_out, V_6tapfilter3_round_out, H_6tapfilter2_round_out,
H_6tapfilter3_round_out, H_6tapfilter4_round_out, H_6tapfilter5_round_out, bilinear0_out,bilinear1_out
,bilinear2_out, bilinear3_out)

begin

if (IsInterLuma='1') then

case (pos_FracL) is

whenpos_b=>

if (blk4x4_inter_calculate_counter /= "0000") then

LPE0_out <= H_6tapfilter2_round_out; LPE1_out <= H_6tapfilter3_round_out;

LPE2_out <= H_6tapfilter4_round_out; LPE3_out <= H_6tapfilter5_round_out;

else

LPE0_out <= (others => '0'); LPE1_out <= (others => '0');

LPE2_out <= (others => '0'); LPE3_out <= (others => '0');

end if;

whenpos_h=>

if (blk4x4_inter_calculate_counter /= "0000") then

LPE0_out <= V_6tapfilter0_round_out; LPE1_out <= V_6tapfilter1_round_out;

LPE2_out <= V_6tapfilter2_round_out; LPE3_out <= V_6tapfilter3_round_out;

else

LPE0_out <= (others => '0'); LPE1_out <= (others => '0');

LPE2_out <= (others => '0'); LPE3_out <= (others => '0');

end if;

whenpos_j=>

if (blk4x4_inter_calculate_counter /= "0101" and blk4x4_inter_calculate_counter /= "0000") then

LPE0_out <= V_6tapfilter0_round_out; LPE1_out <= V_6tapfilter1_round_out;

LPE2_out <= V_6tapfilter2_round_out; LPE3_out <= V_6tapfilter3_round_out;

else

LPE0_out <= (others => '0'); LPE1_out <= (others => '0');

```

```

LPE2_out <= (others => '0'); LPE3_out <= (others => '0');

end if;

when pos_a | pos_c | pos_d | pos_e | pos_g | pos_n | pos_p | pos_r | pos_f | pos_q =>

if (blk4x4_inter_calculate_counter = "0100" or blk4x4_inter_calculate_counter = "0011" or
blk4x4_inter_calculate_counter = "0010" or blk4x4_inter_calculate_counter = "0001") then

LPE0_out <= bilinear0_out; LPE1_out <= bilinear1_out;

LPE2_out <= bilinear2_out; LPE3_out <= bilinear3_out;

else

LPE0_out <= (others => '0'); LPE1_out <= (others => '0');

LPE2_out <= (others => '0'); LPE3_out <= (others => '0');

end if;

when pos_i | pos_k =>

if (blk4x4_inter_calculate_counter = "0111" or blk4x4_inter_calculate_counter = "0101" or
blk4x4_inter_calculate_counter = "0011" or blk4x4_inter_calculate_counter = "0001") then

LPE0_out <= bilinear0_out; LPE1_out <= bilinear1_out;

LPE2_out <= bilinear2_out; LPE3_out <= bilinear3_out;

else

LPE0_out <= (others => '0'); LPE1_out <= (others => '0');

LPE2_out <= (others => '0'); LPE3_out <= (others => '0');

end if;

when others =>

LPE0_out <= (others => '0'); LPE1_out <= (others => '0');

LPE2_out <= (others => '0'); LPE3_out <= (others => '0');

end case;

else

LPE0_out <= (others => '0'); LPE1_out <= (others => '0');

LPE2_out <= (others => '0'); LPE3_out <= (others => '0');

end if;

end process;

```

الملحق D

وحدة خلط الأعمدة MixColumns في معيار التعمية AES بلغة VHDL

تعتمد هذه الوحدة على الضرب بالمصفوفة poly_mat و يمكن تبسيط الجداء بالاعتماد على خواص الحقل $GF(2^8)$. حيث تعتمد هذه الطريقة على أن الضرب بكثير الحدود $\{02\}_{16}$ في الحقل $GF(2^8)$ يكافئ إزاحة الضارب مرة واحدة إلى اليسار مع إضافة كثير الحدود $\{1B\}_{16}$ كما هو مبين بالعلاقة (13.4).

$$\{02\} * M(x) = (m6.x^7 + m5.x^6 + m4.x^5 + m3.x^4 + m2.x^3 + m1.x^2 + m0.x) \text{ xor } \{1B\}_{16} \quad (13.4)$$

ويتحول الجداء بكثير الحدود $\{03\}_{16} = \{02\}_{16} + \{01\}_{16}$ إلى عملية جمع إضافة إلى إزاحة لمرة واحدة. كما توضحه العلاقة (14.4) :

$$\{03\}_{16} * M(x) = (\{02\}_{16} + \{01\}_{16}) * M(x) = \{02\} * M(x) \text{ xor } M(x) \quad (14.4)$$

$$\begin{array}{c|c} & \text{Poly_mat} \\ \hline \begin{array}{c} a'_0 \\ a'_1 \\ a'_2 \\ a'_3 \end{array} & = \begin{array}{c|c} \begin{array}{cccc} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{array} & \begin{array}{c} a_0 \\ a_1 \\ a_2 \\ a_3 \end{array} \end{array} \end{array}$$

تنفيذ وحدة خلط الأعمدة بلغة VHDL

```
entity MixColumns is
  Port ( SYS_CLK, RST : in STD_LOGIC;
        DATA_IN : in STD_LOGIC_VECTOR (127 downto 0);
        DATA_OUT : out STD_LOGIC_VECTOR (127 downto 0));
end MixColumns;

architecture Behavioral of MixColumns is
  COMPONENT Column_Matrix_Mul
    PORT(
      SYS_CLK : IN std_logic;
      RST : IN std_logic;
      COLUMN_IN : IN std_logic_vector(31 downto 0);
      COLUMN_OUT : OUT std_logic_vector(31 downto 0));
  END COMPONENT;
begin
```

```

Column_Matrix_Mul_0: Column_Matrix_Mul PORT MAP(
    SYS_CLK => SYS_CLK,
    RST => RST,
    COLUMN_IN => DATA_IN(127 downto 96),
    COLUMN_OUT => DATA_OUT(127 downto 96)
);

Column_Matrix_Mul_1 : Column_Matrix_Mul PORT MAP(
    SYS_CLK => SYS_CLK,
    RST => RST,
    COLUMN_IN => DATA_IN(95 downto 64),
    COLUMN_OUT => DATA_OUT(95 downto 64)
);

Column_Matrix_Mul_2: Column_Matrix_Mul PORT MAP(
    SYS_CLK => SYS_CLK,
    RST => RST,
    COLUMN_IN => DATA_IN(63 downto 32),
    COLUMN_OUT => DATA_OUT(63 downto 32)
);

Column_Matrix_Mul_3: Column_Matrix_Mul PORT MAP(
    SYS_CLK => SYS_CLK,
    RST => RST,
    COLUMN_IN => DATA_IN(31 downto 0),
    COLUMN_OUT => DATA_OUT(31 downto 0)
);

end Behavioral;

entity Column_Matrix_Mul is
    Port ( SYS_CLK,RST : in STD_LOGIC;
          COLUMN_IN : in STD_LOGIC_VECTOR (31 downto 0);
          COLUMN_OUT : out STD_LOGIC_VECTOR (31 downto 0));
end Column_Matrix_Mul;

architecture Behavioral of Column_Matrix_Mul is
    SIGNAL a0      : STD_LOGIC_VECTOR(7 downto 0) := (OTHERS => '0');
    SIGNAL a0x2    : STD_LOGIC_VECTOR(7 downto 0) := (OTHERS => '0');
    SIGNAL a0x3    : STD_LOGIC_VECTOR(7 downto 0) := (OTHERS => '0');
    SIGNAL a1      : STD_LOGIC_VECTOR(7 downto 0) := (OTHERS => '0');
    SIGNAL a1x2    : STD_LOGIC_VECTOR(7 downto 0) := (OTHERS => '0');
    SIGNAL a1x3    : STD_LOGIC_VECTOR(7 downto 0) := (OTHERS => '0');
    SIGNAL a2      : STD_LOGIC_VECTOR(7 downto 0) := (OTHERS => '0');
    SIGNAL a2x2    : STD_LOGIC_VECTOR(7 downto 0) := (OTHERS => '0');
    SIGNAL a2x3    : STD_LOGIC_VECTOR(7 downto 0) := (OTHERS => '0');
    SIGNAL a3      : STD_LOGIC_VECTOR(7 downto 0) := (OTHERS => '0');
    SIGNAL a3x2    : STD_LOGIC_VECTOR(7 downto 0) := (OTHERS => '0');
    SIGNAL a3x3    : STD_LOGIC_VECTOR(7 downto 0) := (OTHERS => '0');

    SIGNAL r0      : STD_LOGIC_VECTOR(7 downto 0) := (OTHERS => '0');
    SIGNAL r1      : STD_LOGIC_VECTOR(7 downto 0) := (OTHERS => '0');
    SIGNAL r2      : STD_LOGIC_VECTOR(7 downto 0) := (OTHERS => '0');
    SIGNAL r3      : STD_LOGIC_VECTOR(7 downto 0) := (OTHERS => '0');

begin
    --Input Column Breakdown
    a0 <= COLUMN_IN(31 downto 24);
    a1 <= COLUMN_IN(23 downto 16);
    a2 <= COLUMN_IN(15 downto 8);
    a3 <= COLUMN_IN(7 downto 0);

```

--Galois Field Multiplications

GaloisXTwo : PROCESS (a0,a1,a2,a3) ---Multiply by 2 in the Galois Field is done with a left shift and a conditional XOR with X"1B" if the MSB is 1

begin

IF (a0(7) = '1') then ---Each byte is it's own condition
a0x2 <=(a0(6 downto 0) & '0') XOR X"1B";

ELSE

a0x2 <= a0(6 downto 0) & '0';

END IF;

IF (a1(7) = '1') then

a1x2 <=(a1(6 downto 0) & '0') XOR X"1B";

ELSE

a1x2 <= a1(6 downto 0) & '0';

END IF;

IF (a2(7) = '1') then

a2x2 <=(a2(6 downto 0) & '0') XOR X"1B";

ELSE

a2x2 <= a2(6 downto 0) & '0';

END IF;

IF (a3(7) = '1') then

a3x2 <=(a3(6 downto 0) & '0') XOR X"1B";

ELSE

a3x2 <= a3(6 downto 0) & '0';

END IF;

END PROCESS;

--Multiply by 3: Multiply by 3 is done by first doing the Multiply by 2 and then XOR with (equivalent to adding) the original value

a0x3 <= a0x2 XOR a0;

a1x3 <= a1x2 XOR a1;

a2x3 <= a2x2 XOR a2;

a3x3 <= a3x2 XOR a3;

--MixColumns Function

MixColumns : process (SYS_CLK)

begin

IF RST = '1' then

r0 <= X"00";

r1 <= X"00";

r2 <= X"00";

r3 <= X"00";

ELSIF (SYS_CLK'event and SYS_CLK = '1') then

r0 <= a0x2 XOR a1x3 XOR a2 XOR a3; ----Final Sum of products for the matrix

operation

r1 <= a1x2 XOR a2x3 XOR a3 XOR a0;

r2 <= a2x2 XOR a3x3 XOR a0 XOR a1;

r3 <= a3x2 XOR a0x3 XOR a1 XOR a2;

END IF;

END PROCESS;

----Assemble Output Column

COLUMN_OUT <= r0 & r1 & r2 & r3;

end Behavioral;

Abstract

This Master thesis presents an FPGA-based implementation of a real-time video encryption, decryption and decompression system.

H.264 is the most advanced standard for video compression while the confidentiality of the content encoded by H.264 is not well implemented in the coding standard. In this study, we propose to use the Advanced Encryption Standard (AES) algorithm for encrypting encoded video frames.

A test video sequence of 300 frames with QCIF (176×144) resolution is used for demonstration.

This system is implemented on the Cyclone EP4CE115 device equipped on the DE2-115 development kit. The decompression unit consumes 50% of logic elements and 40% of memory elements, whereas the decryption unit consumes only 6% of logic elements and 2% of memory elements. This is due to the complexity of the video decompression and the large amounts of memory required for frame buffering.

The implemented system is capable of real-time processing, because the processing time (2.2 ms per frame) is less than one frame time (4.6 ms per frame = 25 fps).

**Damascus University
Faculty of Mechanical
and Electrical Engineering
Computer and Automation
Engineering Department**



Compression and Encryption of Videos Using Field Programmable Gate Arrays (FPGA) Technique

**A thesis submitted in partial fulfillment of the requirements for the
degree of Master in Computer and Networks Engineering**

**By
Eng. Amal Atallah**

**Supervised by
Dr. Kamal Kamar**

**Academic Year
2013-2014**