مجلة جامعة تشرين للدراسات والبحوث العلمية _ سلسلة العلوم الهندسية المجلد (29) العدد (3) 2007
Tishreen University Journal for Studies and Scientific Research- Engineering Sciences Series Vol. (29) No. (3) 2007

التنفيذ الفعال لخوارزمية Rijndael في الأجهزة القابلة لإعادة البرمجة باستخدام لغة VHDL

الدكتور محمد عيسى *
مجد الله محمد طرّاف **

(تاريخ الإيداع 19 / 7 / 2007. قُبِل للنشر في 2007/9/13)

□ الملخّص □

لقد أدى التطبيق المتزايد لخوارزميات التشفير لضمان أمن الاتصالات في شبكات تبادل المعطيات إلى ضرورة بناء تلك الخوارزميات باستخدام الدارات الصلبة (Hardware) وذلك بغية الوصول إلى أداء عالٍ في عملية التشفير وفك التشفير.

يقدم هذا البحث عرضاً لبناء خوارزمية Rijndael بالاعتماد على شرائح FPGA ولغة التصميم VHDL في بيئة التصميم MAX+PLUS II وذلك للمحاكاة وتحقيق الأمثلية في التركيب.

تم تقليد عملية التشفير باستخدام أسلوب التصميم المتكرر وذلك للوصول إلى أقل استهلاك ممكن من الشرائح، كما عَرض هذا العمل مقارنة بين أداء التصميم المقترح و الأعمال البرمجية المنفذة سابقاً. مع العلم أن الشريحة الهدف المستخدمة لبناء التصميم وتقييم الأداء هي من نوع ACEX1K.

كلمات مفتاحية: AES, VHDL ,خوارزميات التشفير ,FPGA.

* أستاذ مساعد – كلية الهندسة الالكترونية – أكاديمية الأسد للهندسة العسكرية – حلب– سورية.
** طالب دراسات عليا– كلية الهندسة الالكترونية – أكاديمية الأسد للهندسة العسكرية – حلب – سورية.

مجلة جامعة تشرين للدراسات والبحوث العلمية _ سلسلة العلوم الهندسية المجلد (29) العدد (3) 2007
Tishreen University Journal for Studies and Scientific Research- Engineering Sciences Series Vol. (29) No. (3) 2007

Efficient Implementation of the Rijndael in Reconfigurable Hardware Using VHDL Language

Dr. M. Issa[*]
M.Terraf**

(Received 19 / 7 / 2007. Accepted 13/9/2007)

□ ABSTRACT □

The increasing application of cryptographic algorithms to ensure secure communications across data communication networks has led to an ever-growing demand for high performance hardware implementations of the encryption/decryption methods.

This research investigates the Rijndael algorithm with regard to FPGA and VHDL. Altera MAX+PLUS II software is used for simulation and optimization of the synthesizable VHDL code.

The Encryption is simulated using an iterative design approach in order to minimize the hardware consumption. In this work, performance comparisons between the proposed design and previous software implementations are presented. Altera ACEX1K family devices are utilized for hardware evaluation.

Keywords : FPGA, Encryption Algorithms, AES, VHDL.

[*]**Assistant Professor, Faculty of Electronic Engineering, Al-Assad Academy for Military Engineering, Aleppo, Syria.**
****Postgraduate Student, Faculty of Electronic Engineering, Al-Assad Academy for Military Engineering, Aleppo, Syria.**

1- مقدمة:

صممت الخوارزمية من قبل Jon deamn, Finst Rijman [1]، وهي خوارزمية بلوكية، طول بلوك التشفير ومفتاح التشفير يمكن أن يكون bit(128,192,256). وتم في هذه المقالة تنفيذ الخوارزمية بطول 128-bit لكل من كتلة المعطيات ومفتاح التشفير، وقد اختيرت خوارزمية Rijndael كمعيار تشفير متقدم AES (Advanced Encryption Standard) من بين خمس خوارزميات تشفير قدمت للمعهد العالي للمعايير والتقنية في سنة 2000 لتحل محل خوارزمية DES (Data Encryption Standard) و TriplDES وذلك بسبب مزاياها المحسنة [2].

تمت عملية التشفير باستخدام عشر جولات، وفي معظم الحالات يتم البناء المادي لخوارزمية التشفير بشكل مستقل عن باقي النظام وفي هذه الحالة يفضل استخدام أجهزة منطقية قابلة للبرمجة (شرائح FPGA) Field Programmable Gate Arrays كخيار أفضل للحل [3]، وذلك لتحقيق العمل النموذجي لخوارزمية Rijndael حيث يتم وصف ومعالجة التصميم عدة مرات للوصول إلى المنتج الأفضل.

تتركز نقطة البحث على استخدام كود تشفير محسن باستخدام لغة التصميم VHDL (Very High Speed Integrated Circuit Hardware Description Language)[4] لتنفيذ الخوارزمية والوصول إلى معدل نقل معطيات من رتبة Mb/S حيث إن لعامل السرعة في التنفيذ دوراً مهماً، يتضمن البرنامج كوداً لعملية توليد مفتاح الجولة، والذي أمن التعامل السريع مع صندوق تعويض البايت، إضافة إلى كود إزاحة الأسطر، مزج الأعمدة، إضافة مفتاح الجولة.

تم الاستفادة من ميزة جداول البحث Look Up Table (LUT) [5] المميزة لشرائح FPGA لتحل مشكلة ما يسمى صندوق التعويض، وبالتالي نستطيع الحصول على قيمة بايت التعويض مباشرة بدلاً من إجراء عمليات التحويل والحسابات الرياضية بشكل متكرر في كل مرة يتم فيها تعويض البايت.

تم إجراء هذا البحث في أكاديمية الأسد للهندسة العسكرية عام 2007.

2- أهمية البحث وأهدافه:

أصبح استخدام الخوارزميات البلوكية ينتشر بشكل فعال وواسع في تطبيقات متنوعة (تطبيقات البطاقات الذكية، وأجهزة التخزين وقنوات الاتصال)، وعملية تنفيذ هذه الخوارزميات باستخدام الأجهزة المادية والبناء الفيزيائي يحقق أداء أفضل من بنائها باستخدام البرامج الحاسوبية.

يهدف التصميم من خلال استخدام الأجهزة المنطقية في بناء الخوارزمية إلى:

1- تخفيض الاستهلاك في الموارد والتكلفة.

2- المرونة وإمكانية تعديل التصميم وتغيير مفتاح التشفير باستخدام شرائح FPGA.

2- معدل التشفير العالي من خلال اعتماد أسلوب جداول البحث LUT (أزمنة تأخير قليلة من رتبة النانو ثانية).

إن عملية التنفيذ والاختبار تمت باستخدام لغة وصف المكونات المادية VHDL، على بيئة التصميم بمساعدة الحاسب MAX+PLUS II [6].

3- طريقة البحث ومواده:

تم عرض مبسط لمنهجية التصميم بمساعدة الحاسب وأسلوب برمجة شرائح FPGA في المقطع الرابع.

تم تقديم دراسة مبسطة لخوارزمية Rijandeal التي تتناولها هذه المقالة في المقطع الخامس.

يتناول المقطع السادس تقديم عملية التشفير. ويتناول المقطع السابع التنفيذ العملي باستخدام لغة التصميم VHDL. ويتضمن المقطع الثامن عرض نتائج هذا البحث ومقارنتها من خلال الجداول والأعمال السابقة التي تم الحصول عليها وفقاً لمنهجية البحث.

4- برمجة التصميم ومنهجية تصميم الـ FPGA:

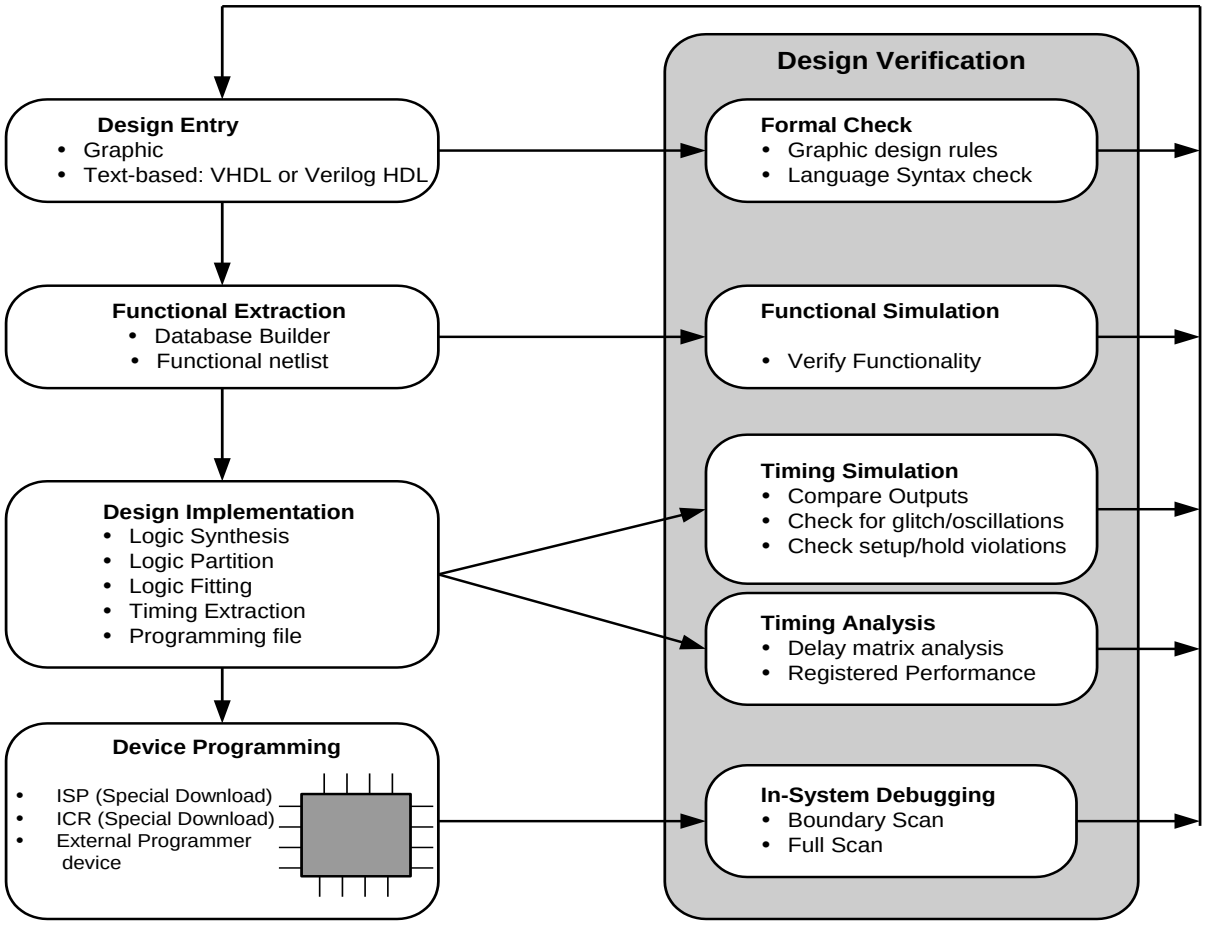
تولد أدوات بيئات التصميم بمساعدة الحاسبCAD (computer aided design) بشكل أوتوماتيكي ملف البرمجة (أو عدة ملفات برمجة) بعد انتهاء عمليتي التركيب المنطقي والتوضع وإنشاء التوصيلات الداخلية. وعند ذلك يمكن برمجة وتشكيل الشريحة الهدف وتصبح جاهزة للاستخدام على لوح الدارة المطبوع Printed circuit Board PCB. تعتمد عملية البرمجة على تكنولوجيا تصنيع الشرائح التكاملية من نوع FPGA، وبشكل عام يوجد نوعان من أنواع البرمجة المتاحة: (1) الشرائح التكاملية القابلة للبرمجة باستخدام قناع Mask-Programmable Devices، والتي تبرمج عادة من قبل المُصنع مستخدماً قناع المستخدم (تتابع الأصفار والواحدات التي تحدد عملية البرمجة) لتشكيل التوصيلات الداخلية النهائية، (2) الشرائح التكاملية الحقلية القابلة للبرمجة Field Programmable Devices، حيث يتم برمجة وتشكيل الشريحة هنا من قبل المستخدم. وتتميز برخص أسعارها بالمقارنة مع النوع الأول، وتتميز بقابلية إعادة البرمجة Reprogrammability حيث تكون نقاط البرمجة مفاتيح يمكن وضعها مفتوحة أو مغلقة.

يمكن برمجة الشريحة التكاملية خارج النظام Out-Of-System باستخدام موائم مناسب Adapter وبطاقة منطقية تركب داخل الحاسب وتعمل مع بيئة التصميم بحيث يتم نقل معلومات البرمجة عن طريق ربط كبل بين البطاقة والمبرمجة الخارجية External Programmer المركب عليها الموائم.أو أن عملية البرمجة تنفذ في النظام In-System Programming (ISP)؛ أي تنقل معلومات البرمجة المخزنة في ملف البرمجة المناسب عن طريق كبل خاص من المنفذ التسلسلي أو المنفذ التفرعي للحاسب إلى الشريحة التكاملية وهي مركبة على الـ PCB.

في الشرائح التكاملية القابلة للبرمجة من نوع FPGA والتي تعتمد في بنيتها على استخدام الـ SRAM، فإن معلومات التشكيل Configuration Data تُحمَّل إلى داخل الشريحة عند تشغيل النظام أو خلال عمل النظام الاعتيادي بعد أن تكون قد وضعت الشريحة التكاملية في مكانها المناسب على الـ PCB. عادة ما يسمى هذا النوع من أنواع البرمجة بالتشكيل في الدارة In-Circuit Reconfigurability (ICR)، والذي يمكن إجراؤه عدد غير محدود من المرات سواء عن طريق استخدام PROM خارجية مخزن عليها معطيات التشكيل مسبقاً والمتوضعة على نفس لوح الدارة المطبوع PCB المتوضع عليه الشريحة التكاملية الهدف أو من خلال عملية الربط مع الحاسب واستخدام أداة خاصة والتي هي عبارة عن كبل خاص لتحميل معلومات التشكيل والبرمجة المخزنة في ملفات البرمجة الخاصة على الشريحة الهدف.

إن تصنيف أدوات تصميم ما على شريحة تكاملية من نوع FPGA، يمكن إعادة توسيعه لتوضيح دورة تصميم الـ CAD لهذا النوع من الشرائح، كما هو موضح في الشكل (1). من الواضح أن القرار الذي يتخذه المصمم ليعمل في بيئة تصميم رسومية أو نصية أو مختلطة تعتمد على رغبته وخبرته. فمدخل التصميم التخطيطي الرسومي للتصميمات

الرقمية المعقدة يمكن أن يبين ويؤكد التنظيم والتخطيط العالي لتدفق المعطيات المرتبطة مع خوارزميات رقمية معقدة. ولكن مدخل التصميم النصي غالباً ما يكون مفضلاً لتصميم التحكم بعمل الخوارزميات الرقمية المعقدة ويسمح باستخدام مجال واسع من أنماط أو أساليب التصميم.



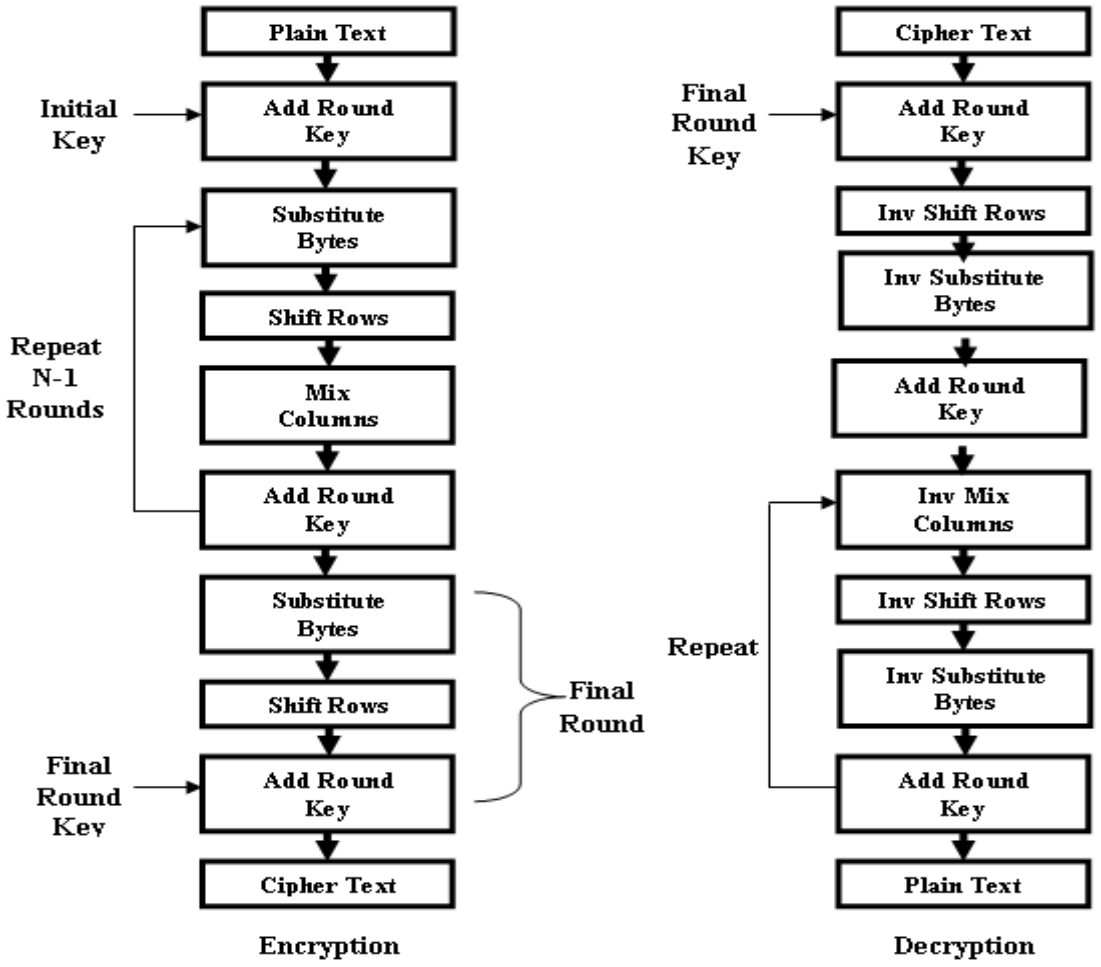
الشكل (1) دورة تصميم الـ CAD لشرائح الـ FPGA

5- خوارزمية Rijandael:

يبين الشكل(2) المخطط الصندوقي لكل من عملية التشفير وفك التشفير في الخوارزمية، كما هو ملاحظ من الشكل هناك ثلاث من التحويلات المتعاقبة المطبقة على كتلة معطيات الدخل خلال عدد محدد من الدورات التكرارية والتي تدعى الجولات، كما أن مفتاح التشفير الأساسي يتم استخدامه لتوليد عدد آخر من المفاتيح الجزئية للجولات المنفذة في الخوارزمية.

تعرف المصطلحات التالية في خوارزمية Rijandael :

- Bl :طول بلوك معطيات الدخل بالبتات. - Nb : عدد الأعمدة (كلمات ذات 32bit).
- Kl :طول مفتاح التشفير المستخدم. - Nk : عدد الكلمات ذات 32bit والتي تشكل مفتاح التعمية.
- Nr : عدد الجولات في الخوارزمية (تابع لكل من Nb و Nk).



الشكل(2) المخطط الصندوقي لعملية التشفير(a) وفك التشفير(b)

من أجل خوارزمية الـ AES فإن:

- طول بلوك المعطيات المسموح هو: $Bl = 128bit$ ، أي $Nb = 4$.
- طول المفتاح المسموح هو: $Kl = 128,192,256bit$ ، أي $Nk = 4,6,8$.
- عدد الجولات المنفذة في الخوارزمية تعتمد على طول المفتاح وبالتالي:

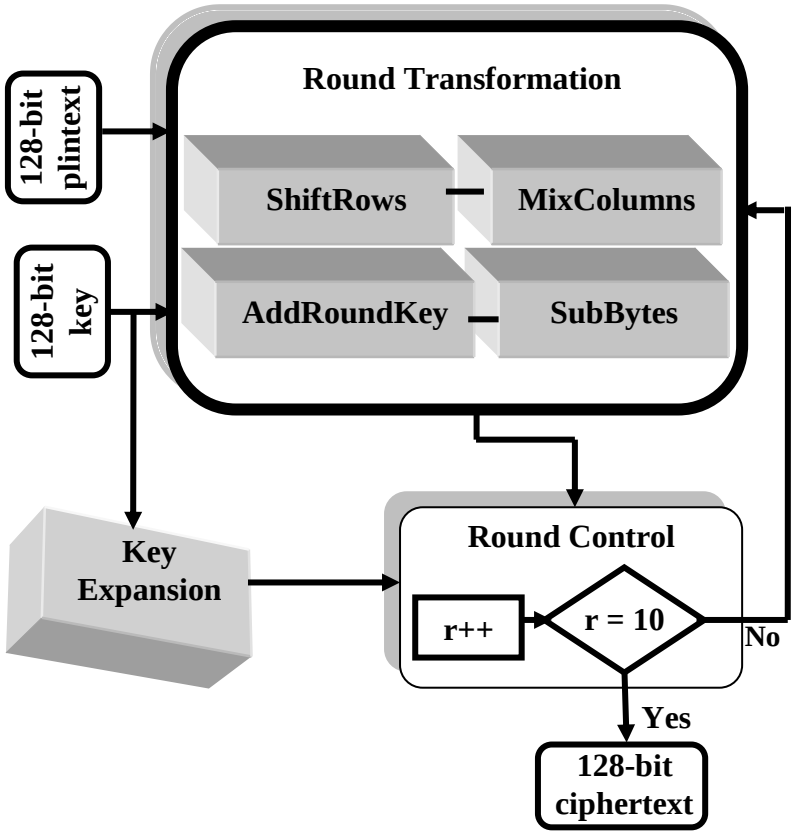
فإن $Nr = 10,12,14$ من أجل $Nk = 4,6,8$ على الترتيب.

تعمل خوارزمية Rijandeal على مصفوفة ذات بعدين من البايتات تسمى الحالة State. تتألف الحالة من أربعة صفوف من البايتات كل منها يتضمن Nb هو طول تتابع الدخل مقسوماً على 32 كما هو مبين في الشكل (4).

6- عملية التشفير:

تعتمد عملية التشفير على تنفيذ تحويلات متتالية على كل كتلة معطيات كما هو موضح في الشكل (3) وفق الخطوات التالية:

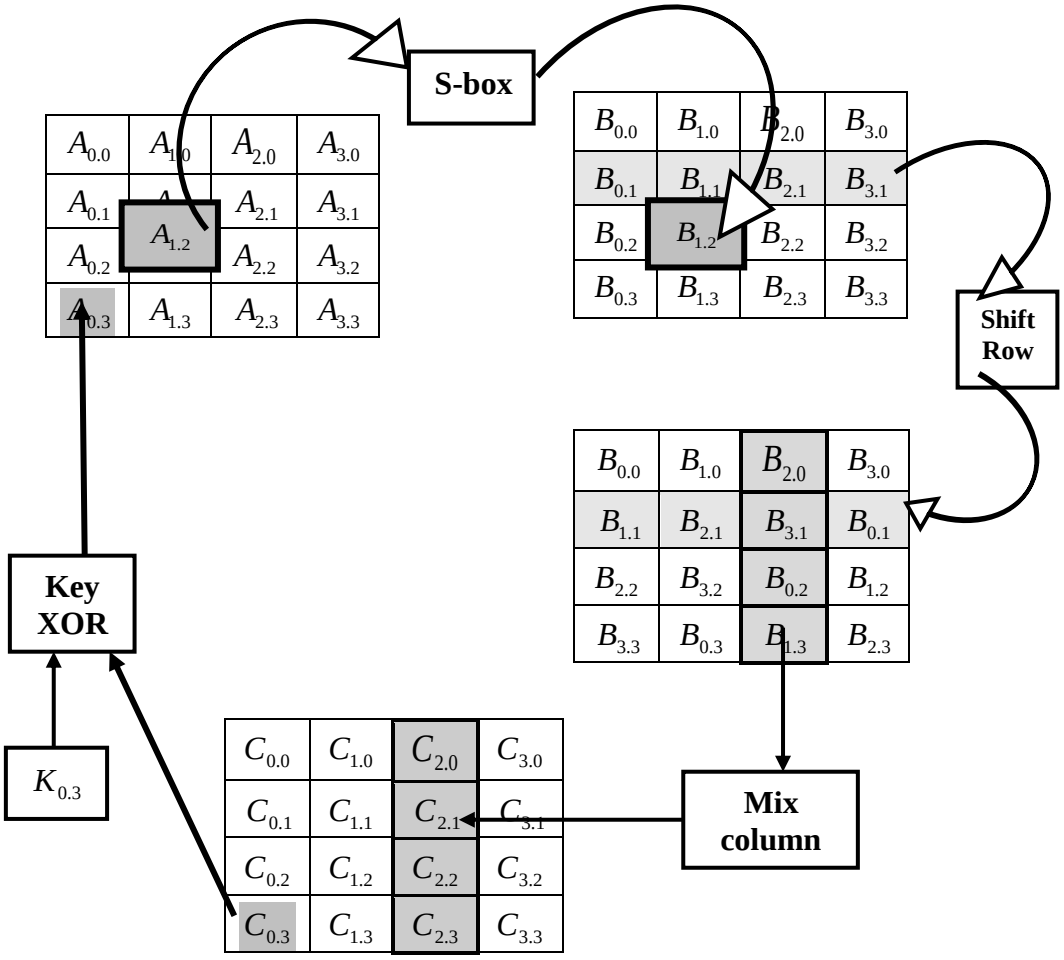
- إجراء عملية XOR لكتلة المعطيات مع المفتاح الأساسي.
- تنفيذ $Nr-1$ جولة منتظمة.
- تنفيذ الجولة الأخيرة والتي تختلف عن الجولة المنتظمة بعدم وجود تحويل مزج الأعمدة.



الشكل(3) مخطط عملية التشفير في خوارزمية Rijandeal

كل جولة منتظمة في الخطوة الثانية تتألف من أربع خطوات منفصلة ومستقلة كما يبين الشكل(4) وهي:

تعويض البايت: فيها يتم تعويض كل بايت من كتلة المعطيات ببايت أخرى وهذه العملية تمثل صندوق التعويض S-box. مع العلم أن تعويض البايت لاخطي وقابل للعكس Invertible، وينجز بأخذ المعكوس الضربي في الحقل Galois Field $GF(2^8)$ [7].



الشكل(4) مخطط الجولة المنتظمة في عملية التشفير

إزاحة الأسطر: ويتم من خلالها تدوير بايتات كتلة المعطيات، وكمية الإزاحة تعتمد على رقم السطر وكتلة المعطيات كما يبين الجدول (1).

الجدول (1) كمية الإزاحة

Nb	C_1	C_2	C_3
4	1	2	3
6	1	2	3
8	1	3	4

مزج الأعمدة: يعمل مزج الأعمدة على كل عمود من حالة المعمي Cipher، حيث تتم معاملة كل عمود ككثير حدود رباعي على الحقل $GF(2^8)$. حيث تضرب كثيرات الحدود الممثلة لكل عمود بكثير حدود ثابت $f(x)$ وتختزل نتيجة عملية الضرب قياس $M(x) = x^4 + 1$ و هذه يمكن كتابتها كعملية ضرب مصفوفة كما يلي:

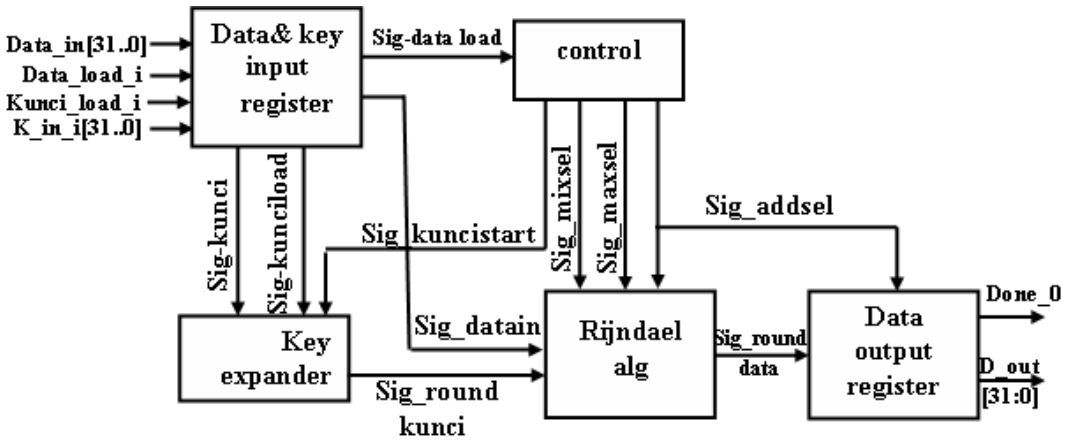
$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

إضافة مفتاح الجولة Round Key Addition: في هذه الخطوة كل من كلمات مفتاح الجولة (Nb كلمة)، والتي يتم الحصول عليها من توسيع المفتاح تضاف بإجراء عملية XOR بينها وبين أعمدة حالة المعمّي.

7- تصميم وتنفيذ عملية التشفير:

يتضمن المخطط الوظيفي للتشفير كما هو موضح بالشكل(5) خمس بلوكات أساسية وهي:

مسجل مفتاح ومعطيات الدخل، توسيع المفتاح، التحكم، خوارزمية Rijndael، مسجل معطيات الخرج.



الشكل(5) المخطط الوظيفي لخوارزمية التشفير

تنفذ خوارزمية التشفير عن طريق توليد مفتاح الجولة وحساب الجولات والتي يمكن تنفيذها بشكل متوازي،

ويتم توليد المفاتيح الجزئية للجولات من مفتاح المعمي الأساسي بواسطة عملية جدولة المفتاح، هذه العملية يمكن تنفيذها وفق المبدأ الأساسي الآتي:

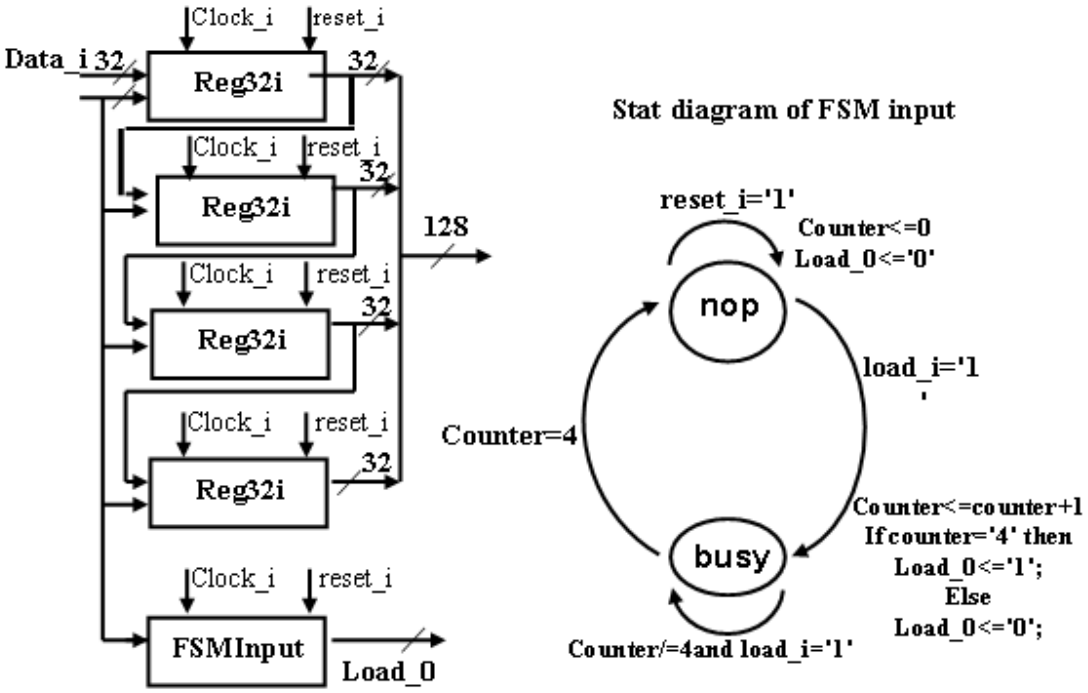
إن عدد بتات مفاتيح الجولات تساوي إلى طول كتلة المعطيات مضروباً بعدد الجولات مضافاً إليها طول مفتاح الجولة الابتدائية، وبالتالي إذا كان $Bl = 128\ bits$ و $Nr = 10$ فإن عدد البتات المفتاحية المطلوبة من جدولة المفتاح 1408-bits.

إتمام عملية توسيع المفتاح للحصول على المفاتيح الجزئية المطلوبة، وبعد ذلك يتم أخذ مفاتيح الجولات بالترتيب؛ أول Nb كلمة مفتاحية ناتجة تؤخذ كمفتاح الجولة الأولى وثاني Nb كلمة مفتاحية ناتجة تؤخذ كمفتاح الجولة الثانية وهكذا.

الميزة في هذا التصميم هو أننا لا نحتاج لتخزين مفتاح الجولة حيث إن المفتاح يتم حسابه وتوليده مباشرة مما يحقق توفير بحجم الموارد.

وفيما يلي عرض لبناء كل كتلة:

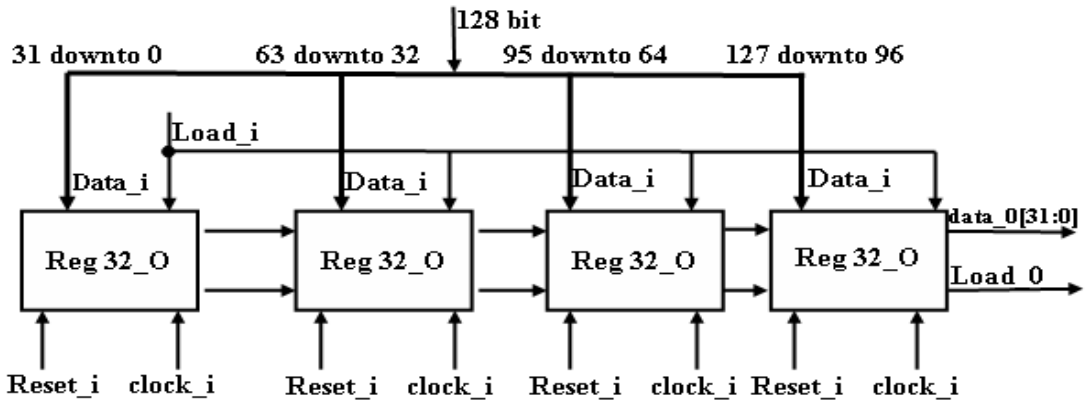
1- **مسجل المفتاح ومعطيات الدخل:** حيث يتم تحميل كل من معطيات الدخل والمفتاح الأساسي إلى أربع مسجلات دخل طول كل مسجل 32bit ليصار إلى تسجيل كامل كلمة المفتاح ومعطيات الدخل ذات الطول 128bit بعد أربع نبضات ساعة كما هو مبين بالشكل(6-a) وعولج ذلك بواسطة دارة تحكم بسيطة الموضحة بمخطط الحالة في الشكل (6-b).



(a) **(b)**

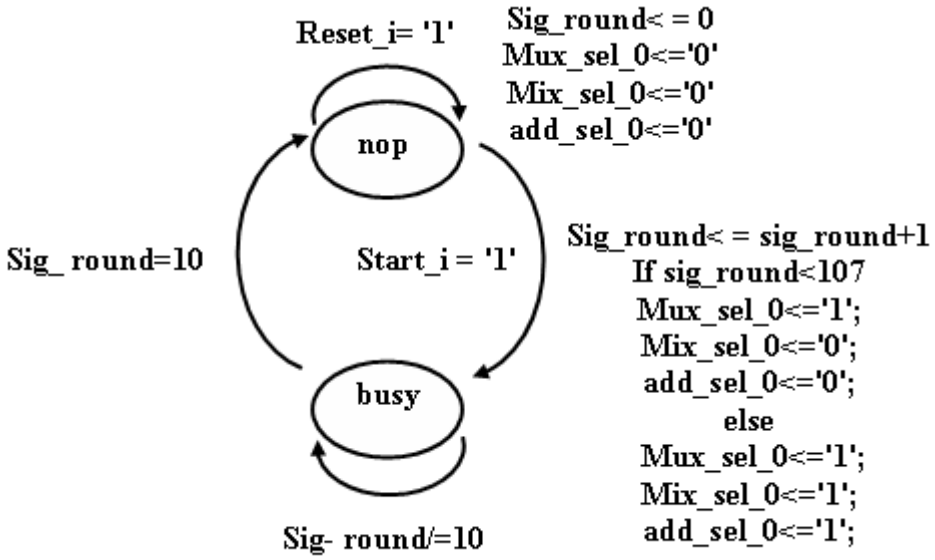
الشكل (6) مسجل الدخل (a) ومخطط الحالة لدارة التحكم (b)

2- مسجل معطيات الخرج: يعمل بطريقة معاكسة لعمل مسجل الدخل حيث إن دخله 128bit وخرجه 32bit كما يبين الشكل(7):



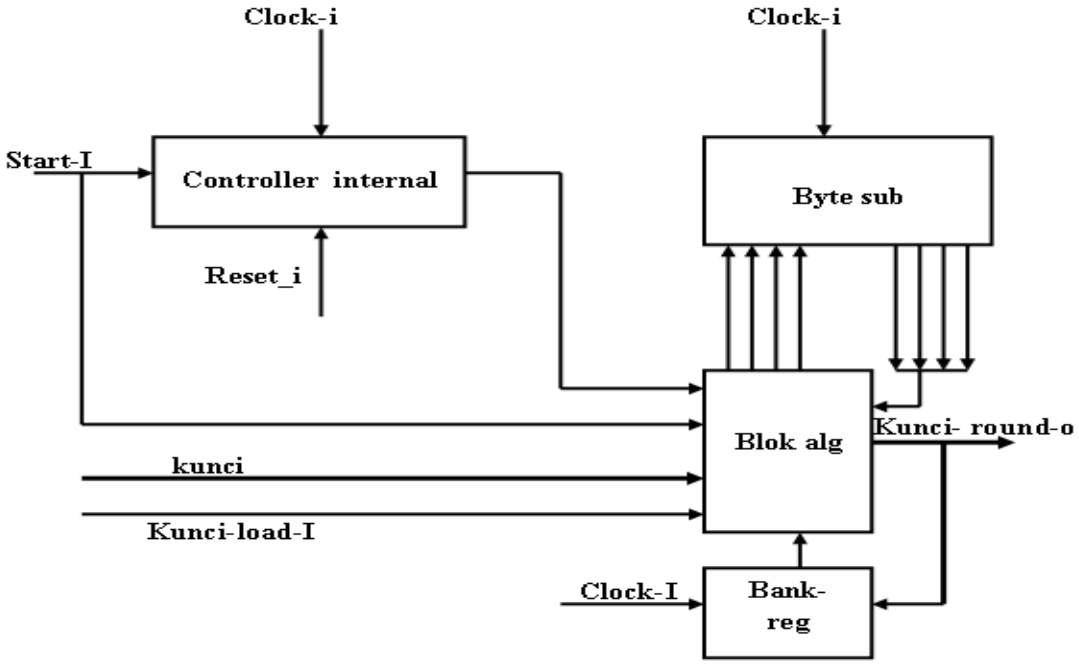
الشكل (7) مسجل معطيات الخرج

3- **التحكم** : تتحكم بكتلة الدخل والخرج وعملية توسيع المفتاح وكتلة الخوارزمية الأساسية ويمكن توضيحها باستخدام مخطط الحالة المو ضح بالشكل (8):



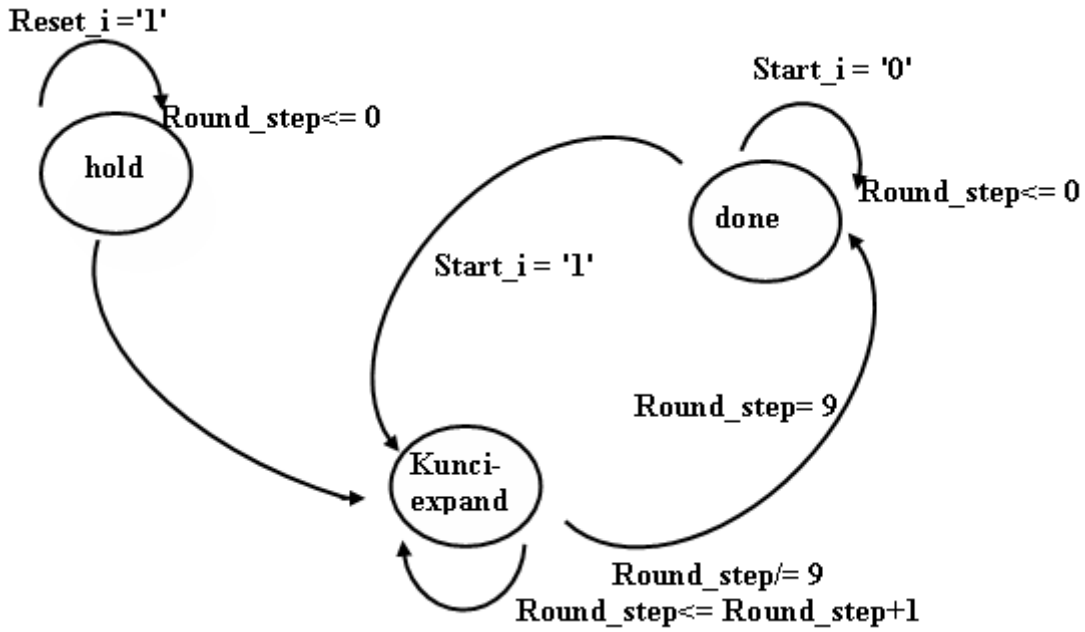
الشكل(8) مخطط الحالة لدارة التحكم

4- توسيع المفتاح: عملية توسيع المفتاح عبارة عن توليد لمفتاح جديد من أجل كل جولة في الخوارزمية و ذلك بالاعتماد على المفتاح الأساسي كما هو مبين بالشكل (9):



الشكل(9) مخطط توسيع المفتاح

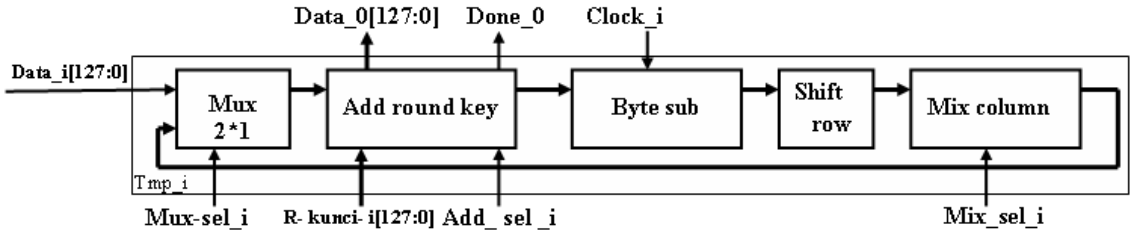
كما هو ملاحظ من الشكل يتم في البلوك الأساسي للخوارزمية إجراء عملية XOR ووظيفة تدوير البايت، كما يتم التحكم بعملية توسيع المفتاح بواسطة دارة تحكم داخلية بسيطة الموضحة في الشكل (10).



الشكل (10) مخطط الحالة لوحدة التحكم الداخلية

5- كتلة الخوارزمية الأساسية:

في هذه الكتلة يتم تنفيذ الجولة الابتدائية والجولة الأساسية المكررة والجولة النهائية، والشكل (11) يوضح المخطط الصندوقي لعمل هذه الكتلة الاساسية وهي تضم:



الشكل(11) مخطط عمل وحدة التشفير الأساسية

الناخب: يتحكم بتمرير معطيات الدخل الأساسية إلى وحدة التشفير في البداية ثم يمرر خرج الجولة السابقة إلى جولة جديدة لتنفيذ جولات التشفير المتتابعة.

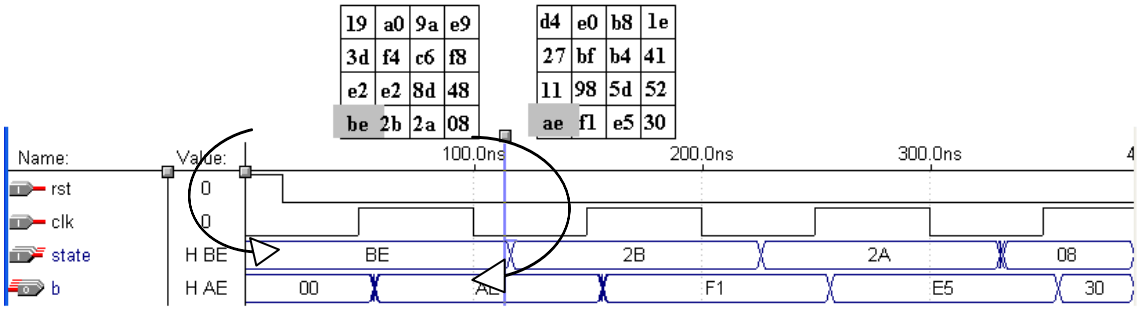
إضافة مفتاح الجولة: يتم إجراء عملية XOR بين الدخل ومفتاح الجولة المولد.

تعويض البايت: تتم عملية تعويض البايت باستخدام LUT، والكود البرمجي التالي يوضح عمل وظيفة تعويض البايت حيث نلاحظ أنه يتم طلب لوظيفة التعويض لاستبدال البايت المحدد ببايت التعويض من صندوق التعويض المحسوب مسبقاً.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use work.Rijndaeldael_package.all;
```

```
  entity byte_sub is
     port (state: in STD_LOGIC_VECTOR(7 downto 0);
           clk: in std_logic; rst: in std_logic;
           b: out STD_LOGIC_VECTOR(7 downto 0));
  end entity byte_sub;
architecture top_aes_RTL of byte_sub is
          begin
     process (clk) is
         begin
            if rst = '1' then
             b <= (others => '0');
                 elsif (clk='1' and clk'event) then
                   b <= SBOX_LOOKUP( state);
               end if;
      end process;
end architecture top_aes_RTL;
```

يبين الشكل (12) نتيجة تنفيذ عملية تعويض البايت وفق القيم المحسوبة من خلال محرر الشكل الموجي.



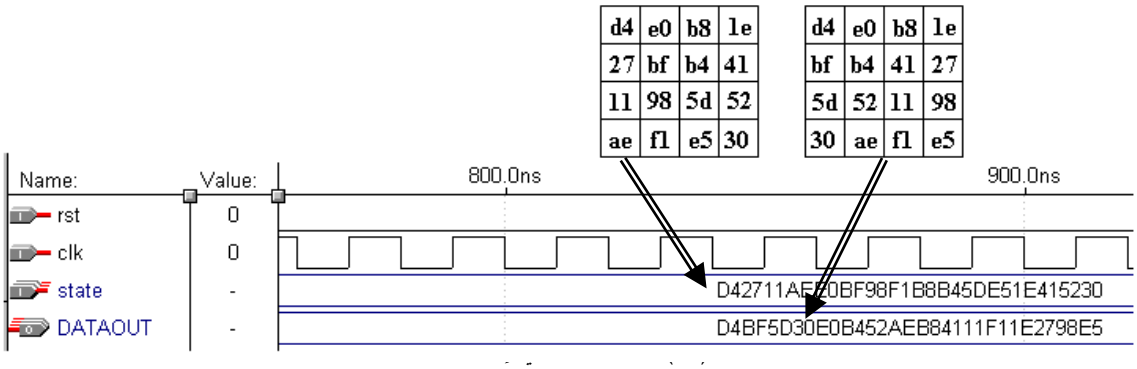
الشكل(12) المخطط الموجي لتعويض البايت

إزاحة الأسطر: يتم ذلك بشكل متتالي حيث إن مواقع البيتات تزاح بشكل متتالٍ مع كل نبضة ساعة حيث السطر الأول إزاحته صفرية والثاني يزاح بمقدار واحد والثالث باثنين والرابع ثلاث مرات، حيث تم وصف هذا العمل باستخدام الكود البرمجي التالي:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use work.Rijndaeldael_package.all;
entity shift_row is
  port (state: in STD_LOGIC_VECTOR(127 downto 0);
        clk: in std_logic; rst: in std_logic;
        DATAOUT: out STD_LOGIC_VECTOR(127 downto 0));
```

```
end entity shift_row;
architecture top_aes_RTL of shift_row is
  begin
  process (clk) is
  begin
       if rst = '1' then
                  DATAOUT <= (others => '0');
            elsif (clk='1' and clk'event) then
                  DATAOUT<= SHIFT_ROW_FUNCT(state);
       end if;
  end process;
end architecture top_aes_RTL;
```

يبين الشكل (13) نتيجة تنفيذ عملية إزاحة الأسطر من خلال استخدام محرر الشكل الموجي.

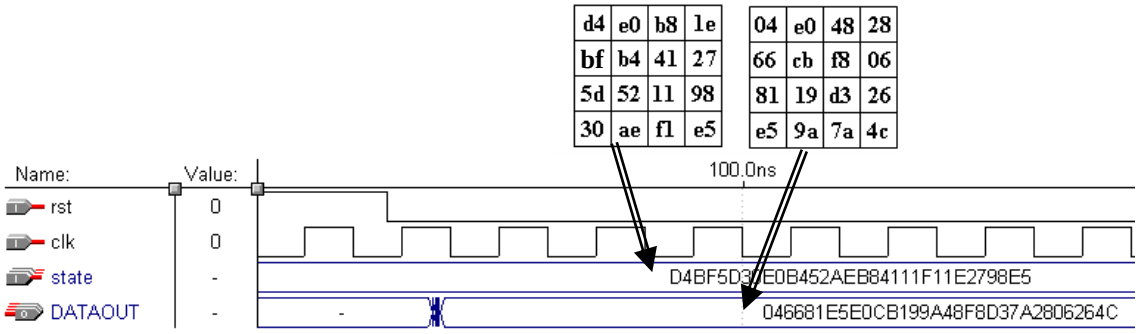


مزج الأعمدة: يتم تنفيذ هذا التحويل باستخدام عمليات الإزاحة المنطقية وXOR والكود البرمجي التالي يمثل استدعاء لوظيفة مزج الأعمدة:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use work.Rijndaeldael_package.all;
entity mix_column is
  port (state: in STD_LOGIC_VECTOR(127 downto 0);
  clk: in std_logic; rst: in std_logic;
  DATAOUT: out STD_LOGIC_VECTOR(127 downto 0));
end entity mix_column;
architecture top_aes_RTL of mix_column is
  begin
    process (clk) is
      begin
            if rst = '1' then
              DATAOUT <= (others => '0');
            elsif (clk='1' and clk'event) then
```

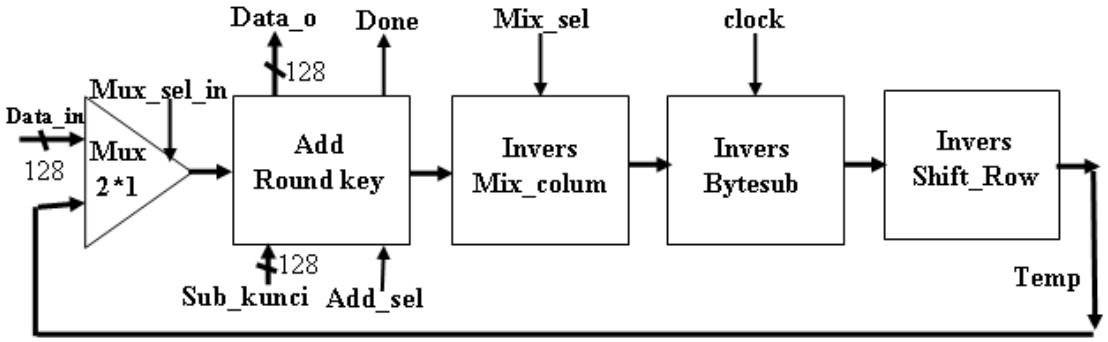
```
                DATAOUT <= MIX_COLUMN_FUNCT(state);
            end if;
    end process;
end architecture top_aes_RTL;
```

يبين الشكل (14) نتيجة تنفيذ عملية مزج الأعمدة من خلال محرر الشكل الموجي.



الشكل(14) المخطط الموجي لمزج الأعمدة

إن عملية فك التشفير هي عملية معاكسة لعملية التشفير، ويمكن الاستفادة من البنية المادية للمشفر لبناء دارة فك التشفير حيث نستبدل دارة تعويض البايت بمعكوس دارة التعويض وكذلك بالنسبة لعملية إزاحة الأسطر ومزج الأعمدة بمعكوس كل منهما وبالتالي تصبح الوحدة الأساسية لخوارزمية فك التشفير كما هو موضح بالشكل (15):



الشكل (15) المخطط الصندوقي لكتلة فك التشفير

وعبر إجراء تعديل بسيط على وحدة التحكم في دارة التشفير يمكن توسيعها بسهولة للتحكم بعملية التشفير وفك التشفير ونظراً للتشابه في عملية البناء لكلا العمليتين كما يبين الشكل(15) تم الاكتفاء بعرض تصميم عملية التشفير.

نُفذ البناء المادي للعمل الكلي بشكل هرمي متسلسل وذلك من خلال تقسيم البنية الأساسية إلى عناصر (كتل بنائية) جزئية. حيث إن الكتل الموضحة في المخطط الوظيفي للتشفير المبين في الشكل (5) والذي تم برمجة كل كتلة ضمنه للحصول على مكون جزئي منفذ باستخدام لغة VHDL وبيئة التصميم المشار اليها. بعد الانتهاء من بناء كافة المكونات الجزئية Components، فإنه يتم تنفيذ عملية تخطيط المنافذ Port Mapping فيما بينها لتشكيل التصميم النهائي الكلي للدارة العملية باستخدام VHDL أو محرر التصميم التخطيطي Schematic Editor في بيئة

التصميم MAX+PLUS II كما هو موضح في الشكل (16).

إن الوحدات البنائية الأساسية Primitive Logic Building Blocks للتصميم والموضحة في الشكل (16) تمثلت بـ (1) مسجل مفتاح ومعطيات الدخل (block_inall) ، (2) كتلة توسيع المفتاح (KEY_GEN)، (3) كتلة التحكم (KONTROL)، (4) كتلة خوارزمية Rijndael (rijndalalg)،(5) مسجل معطيات الخرج (block_out). والتي تم ربطها مع بعضها البعض وفق منافذ الدخل والخرج لكل منها حيث إن عملية الربط بين الكتل هي ربط الكتروني وفق التوصيل المبين، **ومنافذ دخل التصميم هي:**

Clock: مدخل نبضات الساعة.

Reset: مدخل التصفير.

en_d : مدخل التمكين لمعطيات الدخل.

en_k : مدخل التمكين لمفتاح التشفير.

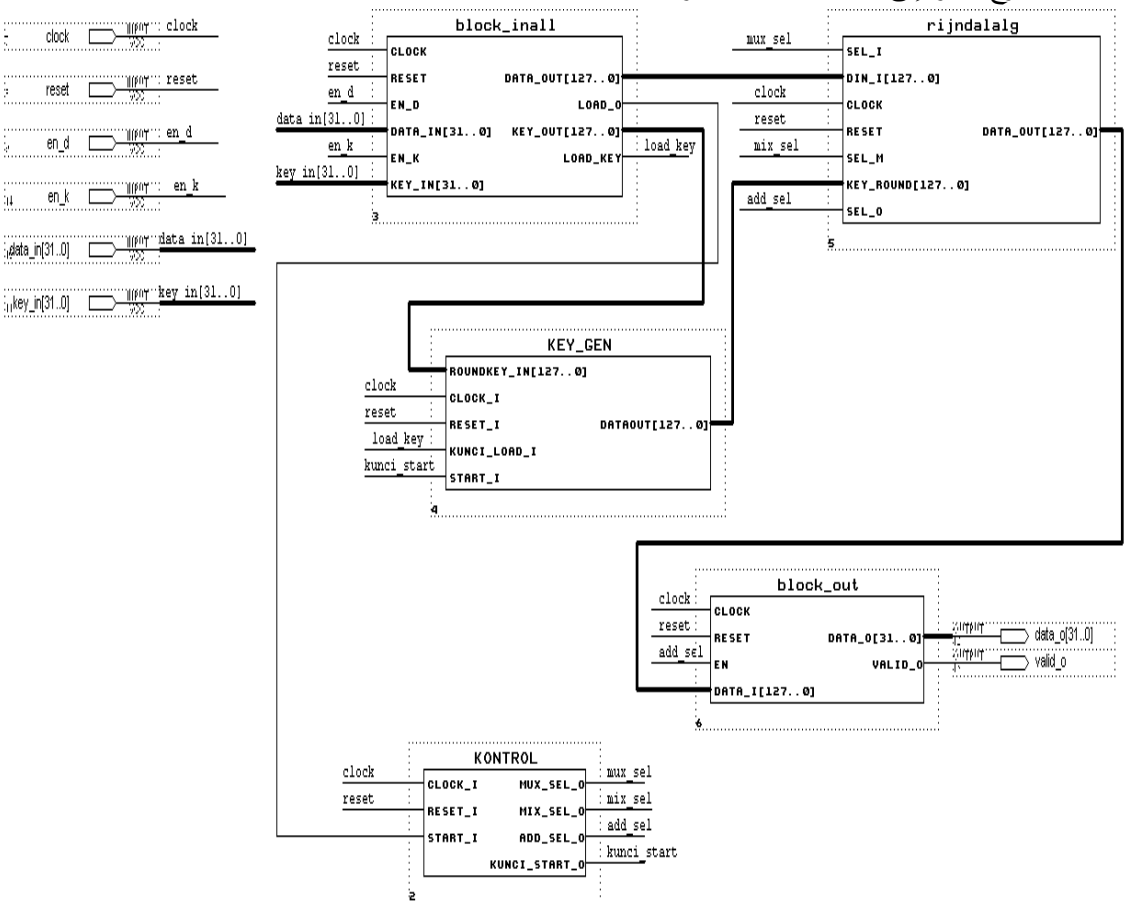
data_in : مدخل المعطيات.

key_in: مدخل مفتاح التشفير.

ومنافذ خرج التصميم هي:

data_o: خرج معطيات التشفير.

valid_o: خرج يشير إلى انتهاء عملية التشفير.



الشكل (16) المخطط العملي للدارة المنفذة للخوارزمية في بيئة التصميم

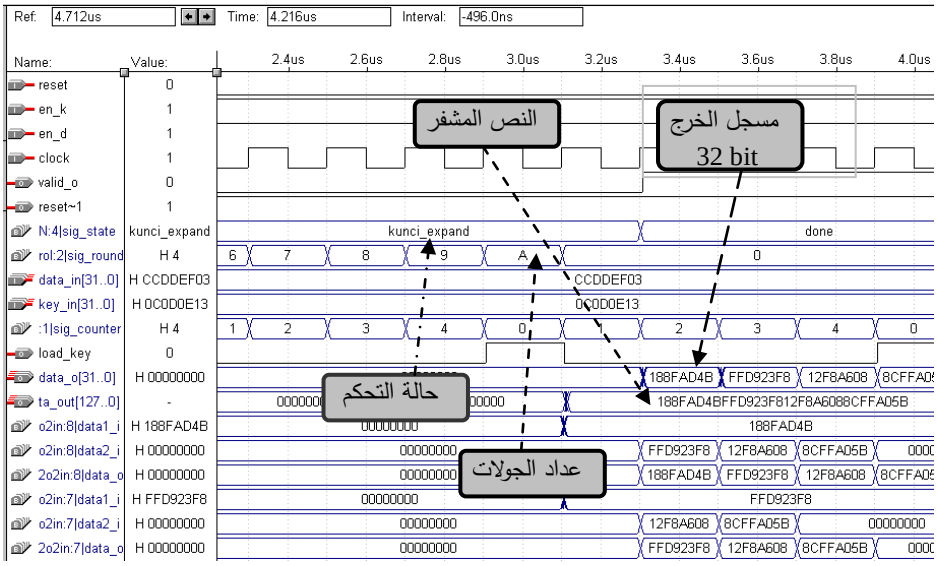
يبين الشكل (17) الرمز التخطيطي للتصميم النهائي، حيث نفذت عملية التصميم باستخدام شريحة تكاملية

FPGA نوع ACEX1K ويرقم EP1K100FC484-1، باستخدام بيئة التصميم بمساعدة الحاسب MAX+PLUS II.

الشكل (17) الرمز التخطيطي للدارة المصممة باستخدام لغة التصميم VHDL وبيئة التصميم MAX+PLUS II

8- التحليل ونتائج المحاكاة:

يبين الشكل (18) الشكل الموجي لخرج الدارة المصممة وحالة التحكم وعداد الجولات بالإضافة إلى النص المشفر حيث من خلال عرض المخطط النبضي يمكن تدقيق صحة عمل مكونات التصميم:



الشكل (18) نتائج المحاكاة لعمل الخوارزمية

يقدم الجدول (2) مقارنة بين أداء التصميم المقترح باستخدام المكونات الصلبة (شرائح FPGA) والمكونات البرمجية (Software) المنفذة في أعمال سابقة [8]، حيث يدل المردود على سرعة عملية التشفير والذي يحسب من العلاقة التالية:

$$\text{Throughput} = 128 \text{ Bits} / (\text{Cycles per Encrypted Block} * \text{Time Period})$$

أي طول كتلة المعطيات مقسوماً على عدد التكرارات لكل كتلة المشفر مضروب بالدور التكراري (أصغر دور لنبضات الساعة مسموح تطبيقه). إن الدور التكراري هو (43 ns) وعدد التكرارات من أجل كتلة التشفير (12) وبالتالي يصبح المردود:

Throughput = 128 / (12*43 ns) = 248Mb/sec

الجدول(2) مقارنة سرعة العمل للدارة المصممة والأعمال البرمجية السابقة

Implementation	Encryption Speed
Software implementation(ANSI C)	27Mb/s
Visual C++	70.5Mb/s
Hardware implementation(Altra)	248Mb/s

يبين الجدول ازدياد سرعة العمل (مردود أعلى) عند تنفيذ الخوارزمية باستخدام الدارات الصلبة، والاستفادة من ميزة LUT في شرائح FPGA.

إن عدد الخلايا المنطقية (Logic Sells) و القلابات (flipflops) المستخدمة في الشريحة الهدف هو :

Total logic cells required	Total flipflops required
18852	952

الانخفاض في استهلاك الموارد بسبب وجود دارات حساب مفتاح الجولة بدلاً من تخزين مفاتيح الجولات بشكل مسبق ولاسيما أنه يتم تغيير مفتاح التشفير من حين لآخر.

الاستنتاجات والتوصيات:

من خلال هذا البحث تم التوصل إلى:

1- سرعة تشفير عالية من رتبة النانو ثانية (دور تكراري 43ns) وهذا يجعل التصميم مفيداً لأنظمة الاتصالات اللاسلكي.

2- استهلاك موارد منخفض يساهم في تقليل حجم أجهزة التشفير المستخدمة.

3- بناء التصميم باستخدام الأجهزة القابلة لإعادة البرمجة FPGA يقدم مرونة من ناحية إمكانية تعديل التصميم لاحقا.

المراجع:

[1] DAEMEN, J.; RIJMEN,V. *A Specification for the AES Algorithm Rijndael*,V 3.7,10th, 2003.

[2] KOSARAJU,N. *A VLSI Architecture for Rijndael, the Advanced Encryption Standard* University of South Florida, November 13, 2003, 93.

[3]BEAN,M.; FICKE,C.; ROZYLOWICZ,T.; WEEKS,B. *Hardware Performance simulations of Round 2 Advanced Encryption Standard Algorithms*, 3.6.2007 available at < http://csrc.nist.gov/encryption/aes/round2/NSAAESfinalreport>.

[4]ARMSTRONG, J.; GRAY,G. *VHDL Design Representation and Synthesis*, 2nd edition, Prentice Hall, 2000,651.

[5]WOLINGER,T.; PAAR,C. *Security Aspects of FPGAs In Cryptographic Applications, New Algorithms, Architectures, and Applications For reconfigurable Computing*, Kluver, 2004, 95-135.

[6] *ALTRA, MAX+PLUS II Getting Started*, 1997-2001.

[7]RICHARD,E. *Algebraic Methods for Signal processing and Communications Coding, Springer- Verlag New York, 1992, 7-19.*

[8]GLADMAN,B. *The AES Algorithm (Rijndael) in C and C++, performance of the optimized implementation,*, 10.6.2007, < http://fp.gladman.plus.com/cryptography_technology/rijndael/index.htm>.