INGÉNIEUR SPÉCIALISTE
MICRO-ÉLECTRONIQUE
INFORMATIQUE ET NOUVELLE
TECHNOLOGIE
ISMIN-2A

PROJET PCSN
SEMESTRE 7

Modélisation VHDL de l'algorithme de chiffrement AES

Réalisé par :
M. Ismaiyl ZERMOUM

Encadré par :
M. Olivier POTIN

Le 1er janvier 2021
Année universitaire 2020/2021

Le succès vient de la curiosité, de la concentration, de la persévérance et de l'autocritique.
Albert Einstein

Table des matières

Table des figures

1 Introduction

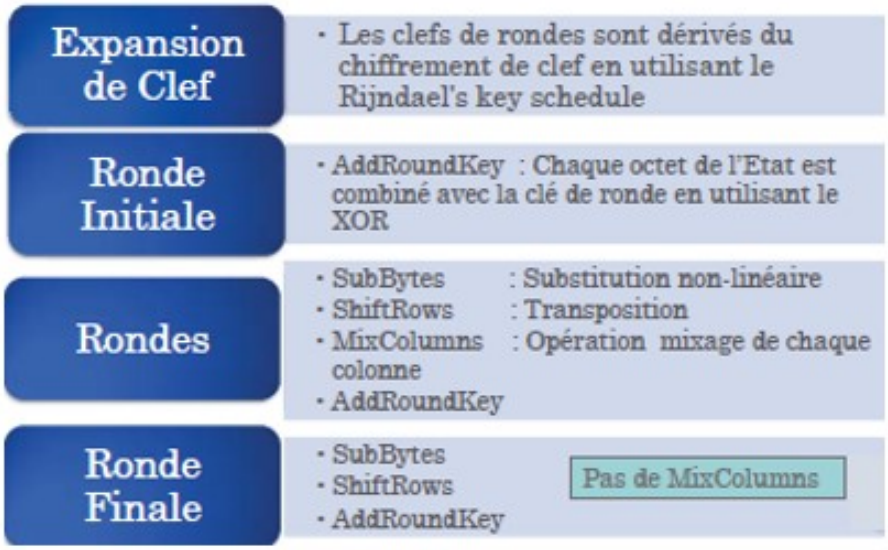
Dans le cadre de développement technologie des nombreuses communications ce fait en utilisant des supports électronique. La protection des données est ultra essentiel dans n'importe quelle communication, donc on aura besoin de protéger les données contre les attaques malveillantes.

La cryptographie est une science de secret permis de protéger tell canal de communication contre les espionnés, on transforme le message claire a un message chiffré on utilisant l'une des algorithmes le plus fiable dans la cryptographie. Dans ce projet on s'intéresse à la conception de l'algorithme de chiffrement AES-128 à l'aide de langage de description matériel VHDL.

AES est conçu par Rijmen-Daemen en Belgique, contient 128/192/256 bits de clef. 128 bits de données, c'est un chiffrement itératif plutôt que Feistel, il traite les données sous forme de bloc données entier à chaque ronde.

AES conçu pour être :
— Résistant aux attaques connues ;
— Rapide et de code compacte sur différents processeurs ;
— De conception simple.

1.1 Description de AES



2 Description de l'AES 128 bits

2.1 Fonction SBox

Sbox c'est un tableau de substitution de taille 16*16 byte il est utilisé dans l'algorithme de cryptographie AES (Advanced Encryption Standard). Pour l'implémentation de ce tableau dans le VHDL nous avons utilisé l'approche comportementale basé sur un processus et une structure conditionnelle, multiplexeur que nous permis d'avoir à la sortie la valeur corresponde à chaque paire « xy ».
La boit Sbox il a une entrée Sbox_i de taille un octet et une sortie Sbox_o de même taille comme illustrée la figure suivante :

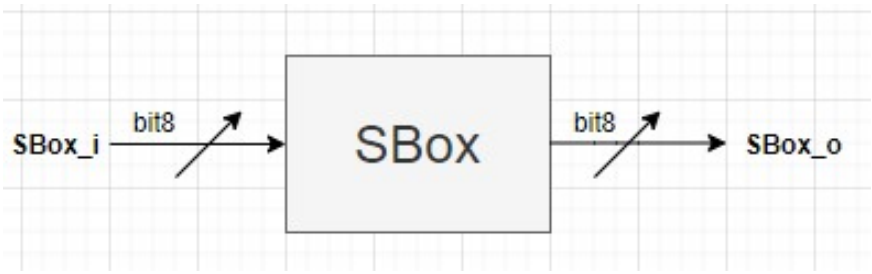


FIGURE 1 – Entité SBox

Après la conception de model Sbox nous avons fait un testbench pour simulé le modèle et vérifier son fonctionnement. Pour cela on a utilisé tous les données du tableau avec un délai entre deux valeurs successives 10 ns (utilisation de fonction wait for 10 ns), les résultats obtenus après la simulation sont illustrée dans le schéma suivant :

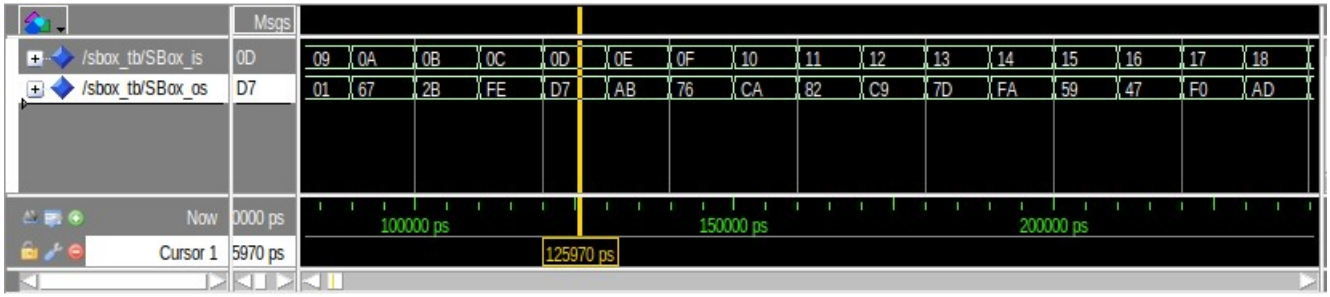


FIGURE 2 – Simulation de SBox

2.2 Fonction SubBytes

SubBytes c'est une fonction de substitution no linéaire que reçoit en entre un table de 4*4 chaque bloc est de taille 8 bits, elle fait la Sbox de

ce tableau pour nous donne un nouveau tableau de même taille.

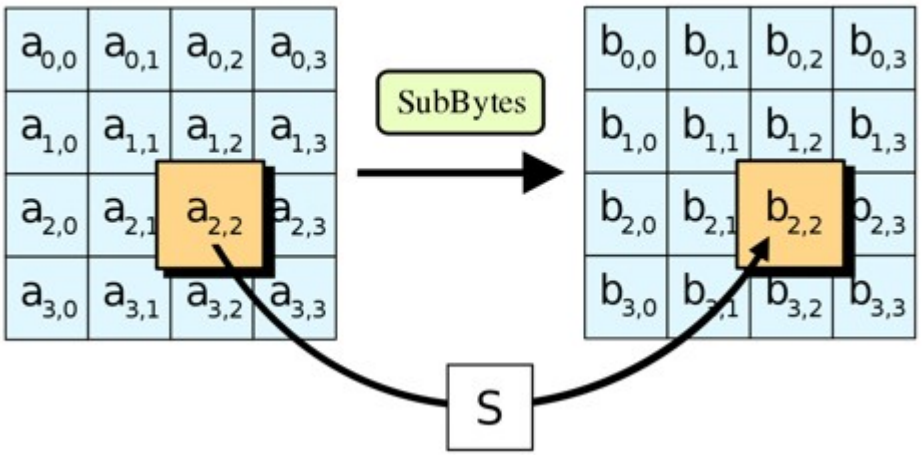


FIGURE 3 – Fonction SubBytes

On peut faire la conception de cette fonction a l'aide de deux boucles for

```
-- code vhdl après l'instanciation de SBox
G1: for i in 0 to 3 generate
   G2: for j in 0 to 3 generate
    Sb:    SBox port map (
      SubBytes_i(i)(j), SubBytes_o(i)(j));
   end generate G2;
end generate G1;
```

Pour la partie simulation on a fait un testbench avec un tableau (4x4) des données voici les résultats obtenu :

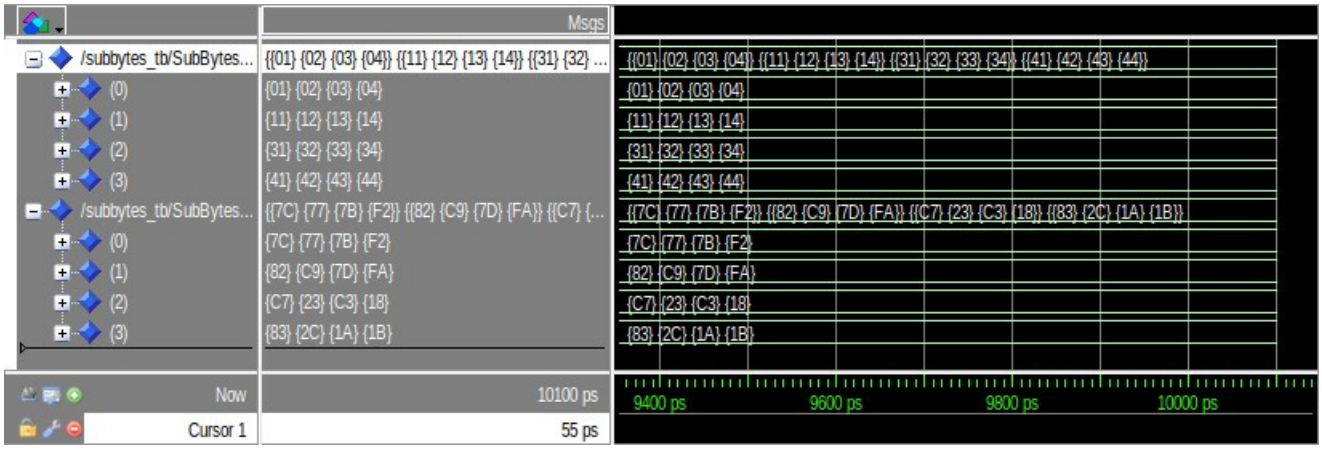


FIGURE 4 – Simulation de SubBytes

2.3 Fonction ShiftRow

Le composant shiftRow permis de faire un décalage circulaire gauche des octets des 4 lignes respectivement 0, 1, 2 et 3 octets. La première ligne n'est donc pas décalée. Comme illustré la figure suivante :

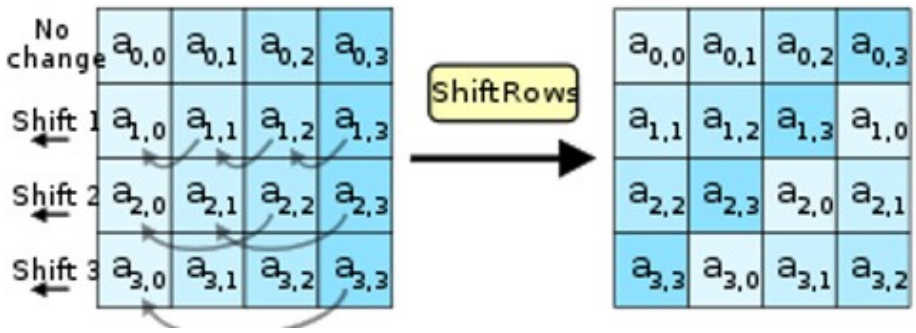


FIGURE 5 – Fonction ShiftRow

Nous pouvons écrire que a'(r,c) = a(r,(c+shift(r,4)) mod 4 pour r et c varie de 0 à 4. Cette fonction en peut l'écrire en VHDL comme suit :

```vhdl
-- code vhdl de fonction shiftRow
P: process (shiftrows_i)
begin
  B0: for i in 0 to 3 loop
       B1: for j in 0 to 3 loop
        shiftrows_o(i)(j) <= shiftrows_i((i+j) mod 4)(j);
        end loop B1;
  end loop B0;
end process P;
```

Après la conception de ShiftRow on a fait une testbench pour le testé, le résultats de simulation sont illustré dans la figure suivante :

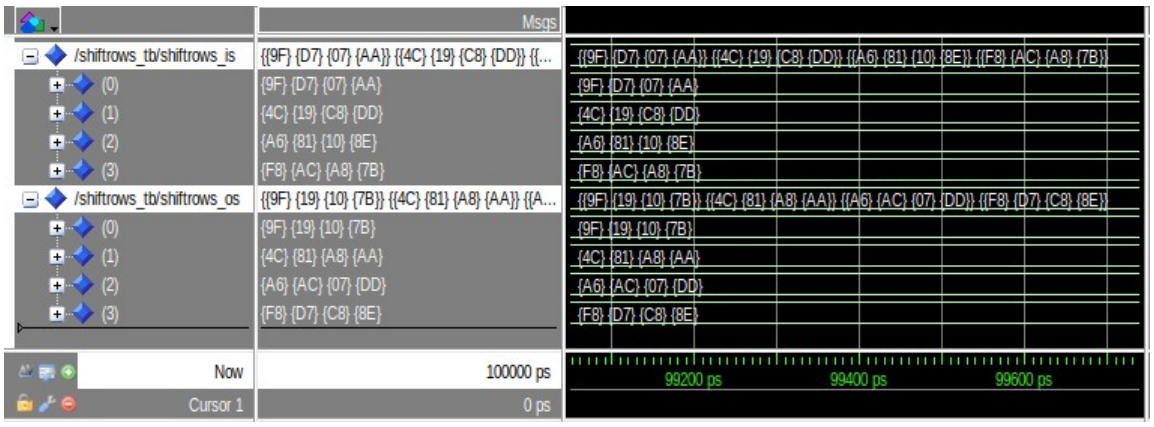


FIGURE 6 – Simulation de shiftRow

2.4 La fonction MixColumns

C'est une fonction qui permis de transforme chaque octet d'entrée en une combinaison linéaire d'octets d'entrée (produit matriciel sur CG(2^8)).

Dans la description VHDL pour le produit nous avons fait des décalages droit et pour la somme (connu dans les produit matricielle) on a fait des Xor.

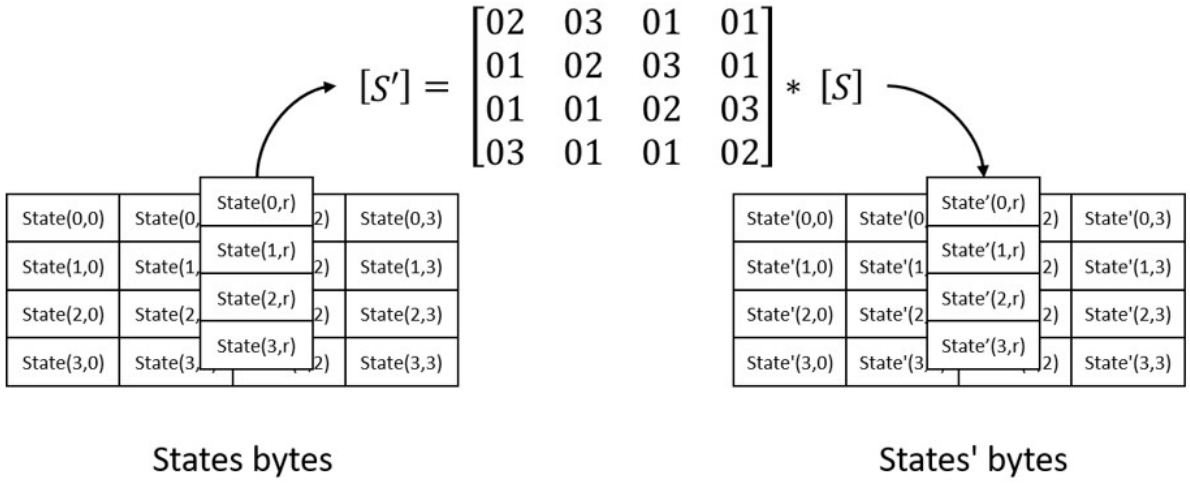


FIGURE 7 – Fonction MixColumns

voici les résultats obtenu de multiplication des deux matrices :

$$S'_{0,c} = (\{02\}\ S_{0,c}) \oplus (\{03\} \bullet S_{1,c}) \oplus S_{2,c} \oplus S_{3,c}$$
$$S'_{1,c} = S_{0,c} \oplus (\{02\} \bullet S_{1,c}) \oplus (\{03\} \bullet S_{2,c}) \oplus S_{3,c}$$
$$S'_{2,c} = S_{0,c} \oplus S_{1,c} \oplus (\{02\} \bullet S_{2,c}) \oplus (\{03\} \bullet S_{2,c})$$
$$S'_{3,c} = (\{03\} \bullet S_{0,c}) \oplus S_{1,c} \oplus S_{2,c} \oplus (\{02\} \bullet S_{3,c})$$

la multiplication polynomiale (.) est définie sur le champ fini GF(2^8) pour la conception de MixColumns nous avons fait la conception de composant qui fait le produit matricielle dans l'espace CG puis on a fait l'instanciation de composant pour la conception de MixColumns.
La figure suivante montre les résultats de simulation du composant :

FIGURE 8 – Simulation de MixColumns

2.5 Fonction AddRoundKey

La fonction AddRoundKey consiste à addition deux matrices. Xor entre les la matrice d'état et celle de la clé de round.

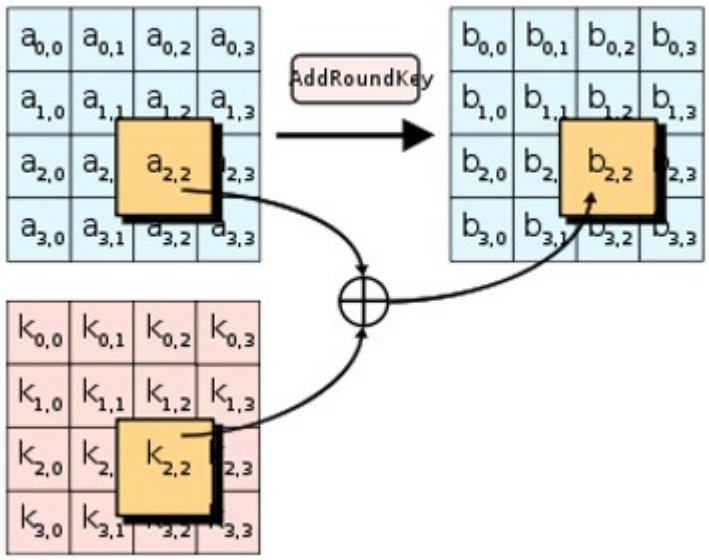


FIGURE 9 – Fonction AddRoundKey

Après la conception de AddRoundKey on a assurè la bonne fonctionnalité de ce dernier a l'aide du simulation comme illustré la figure suivante :

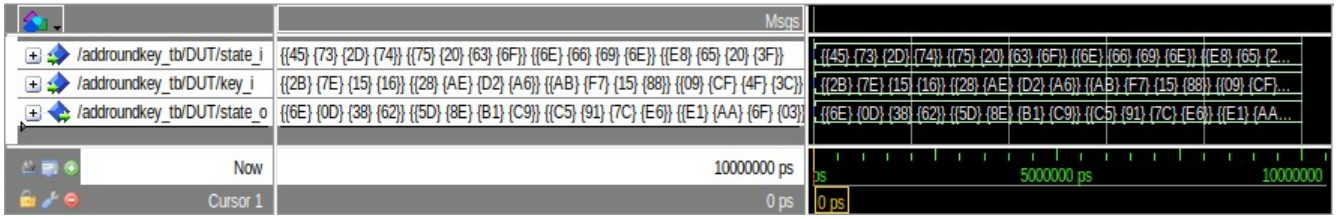


FIGURE 10 – Simulation de AddRoundKey

2.6 Fonction AESRound

Le but de AESRound c'est de traiter les rounds de 0 à 10. l'AESRound contient un ensemble des composant (Figure 11) pour faire le chiffrement de chaque round nous avons utilisé les signaux enableMixcolumns_i et enableRounComputing_i pour traiter Round1 et Rouind 10, et un multiplexeur pour les autres Rounds. à la sortie on a un registre pour la mémorisation de l'état de Round.

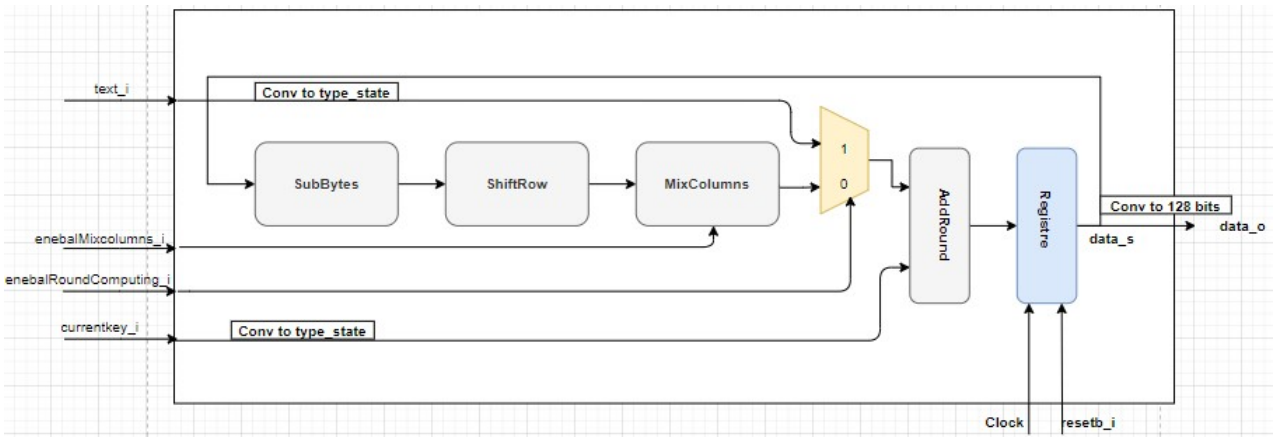


FIGURE 11 – Description de AESRound

Nous avons réalise un testbench pour assuré la fonctionnement de AESRound, la figure suivante illustre les résultats trouvé après la simulation :

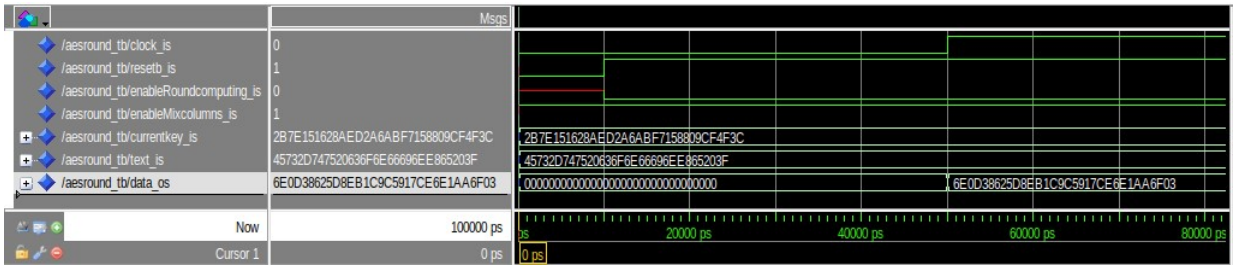


FIGURE 12 – Simulation de AESRound

2.7 Fonction KeyExpander

keyExpansion permis de génère les clé à partir de round 1, on prend une clef de 128 bits (16 octets) et la développe en un tableau de 44 mots de 32 bites chacun.
Puis on fait une rotation_word on effectuant un décalage circulaire à gauche d'un octet sur un mot.et après on applique SubByte_Word sur les résultats de rotation_word et on effectuant une substitution d'octets sur chaque octet du mot d'entrée en utilisant la Sbox. Et finalement on fait le XOR avec Rcon (constante du round) comme illustré la figure suivante :

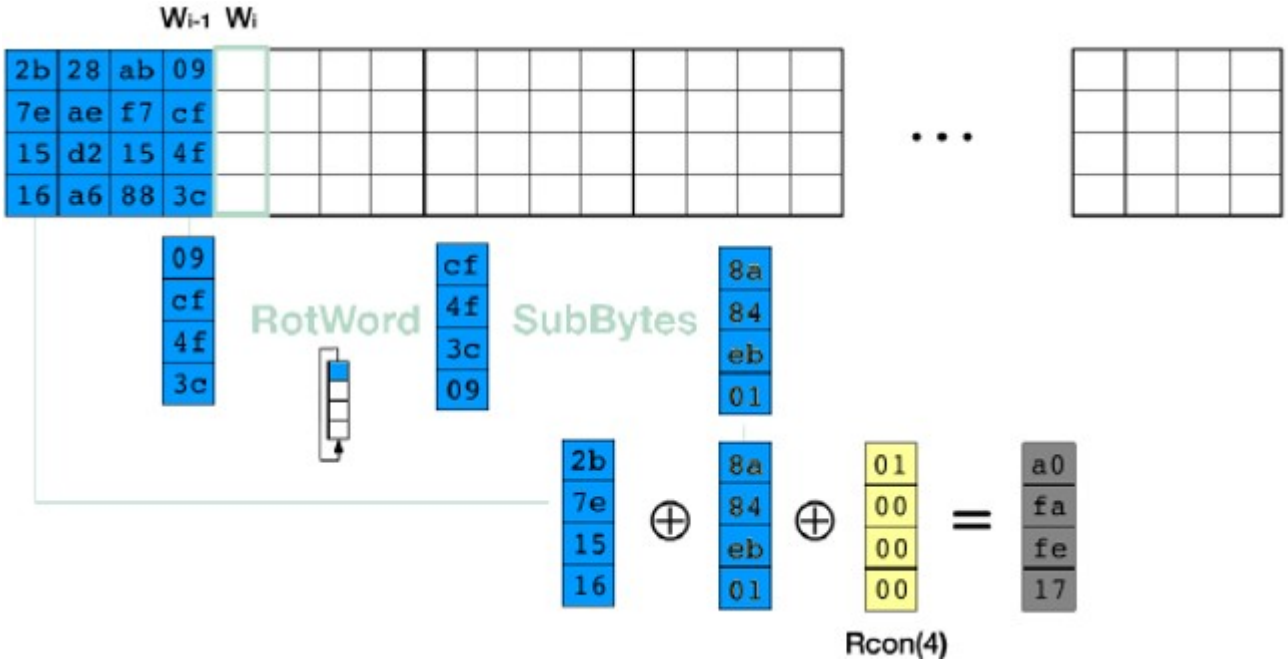


FIGURE 13 – Fonctionnement de Key_Expansion

Après avoir fini la conception de ce composant on a fait un testbesnch pour assurer la bonne fonctionnalité de ce dernier,la figure suivante montre les résultats obtenir.

2.8 Machin à état : KeyExpander_FSM_Moor

KeyExpander_FSM_Moor c'une machine de Moor qui permis de mettre à jours l'état présent, l'état futur et les sorties associées à l'état présent lors des fronts montants de l'horloge. L'état Init correspond au Round 0, l'état Count aux rounds 1 à 9 et l'état done au Round 10 comme le montre la figure suivante :

FIGURE 14 – Simulation de Key_Expander

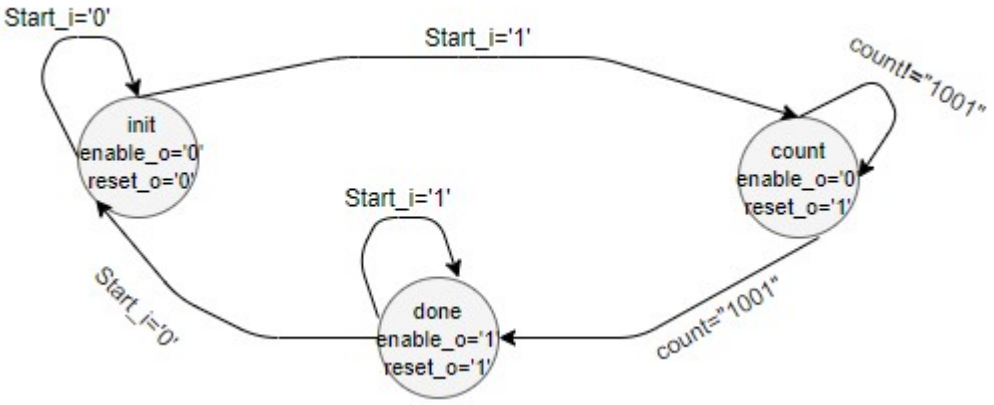


FIGURE 15 – KeyExpander_FSM_Moor

2.9 KeyExpansion_I_O

KeyExpansion_I_O c'est un composant qui instancié un compteur qui nous permit de parcoure le tableau Rcon et une machine a état de Moor pour mettre à jours le compteur et un registre pour la mémorisation de clé de round, la figure suivante illustre les différent composants de KeyExpansion_I_O.

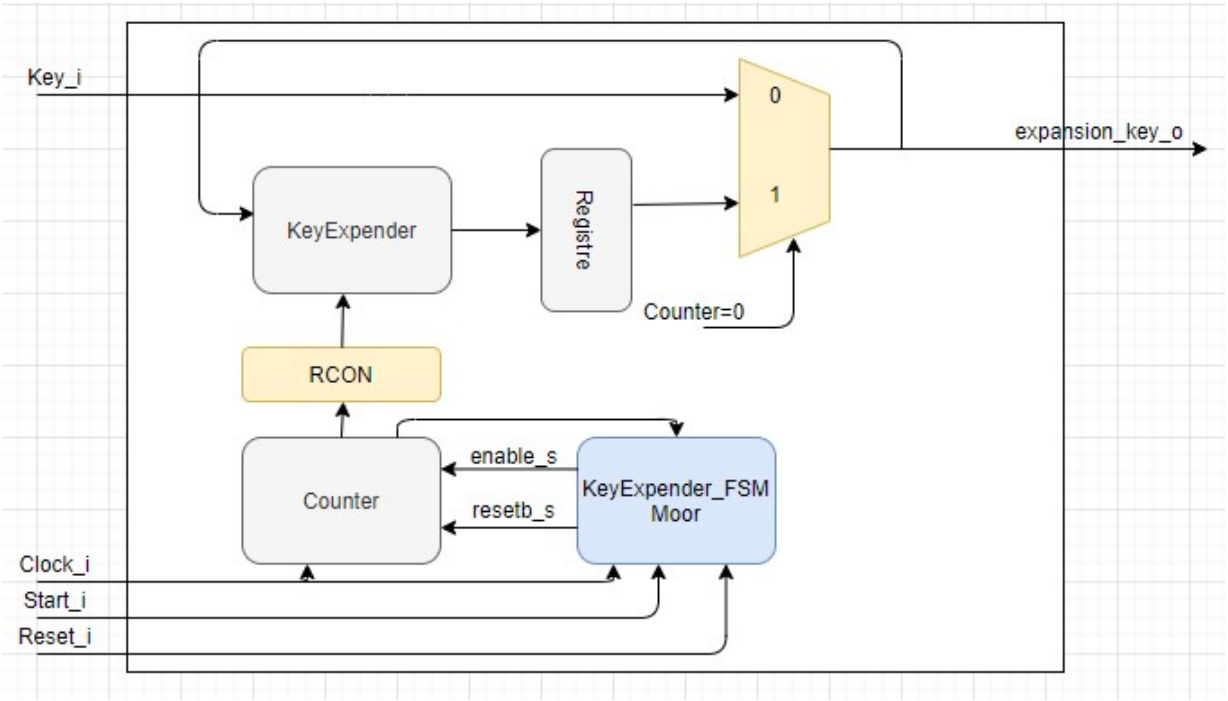


FIGURE 16 – Fonction de KeyExpansion_I_O

Après la conception nous avons réalisé une testbench pour test la fonctionnalité et on a obtenir des bons résultats comme illustré dans la figure suivant :

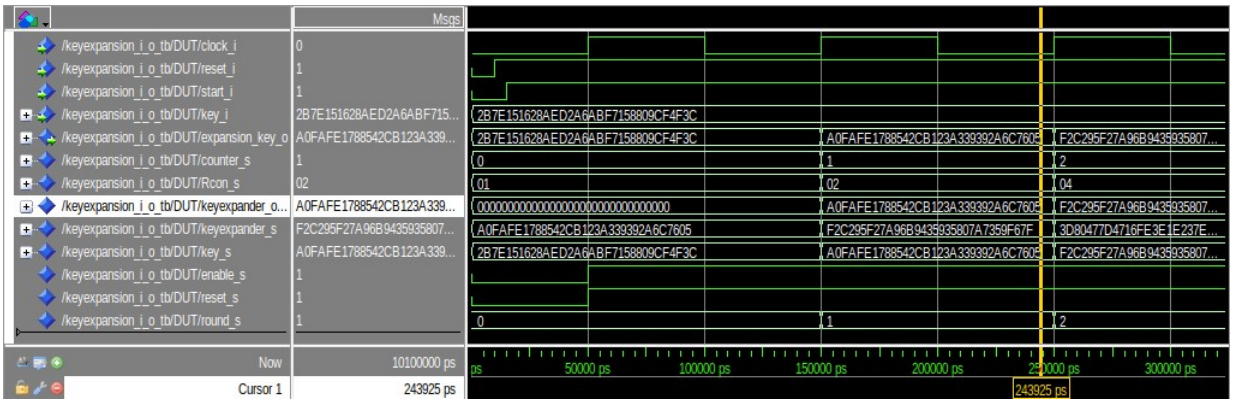


FIGURE 17 – Simulation de KeyExpansion_I_O

2.10 Architecture globale de l'AES

Nous avons préparé tous les composants nécessaire pour l'AES, maintenant on va instancié toute les composants on ajoutant un multiplexeur a la sortie pour ne pas sortir les résultats de chiffrement que a la fin de round 10 comme le montre la figure suivante : On a préparé un testbench pour

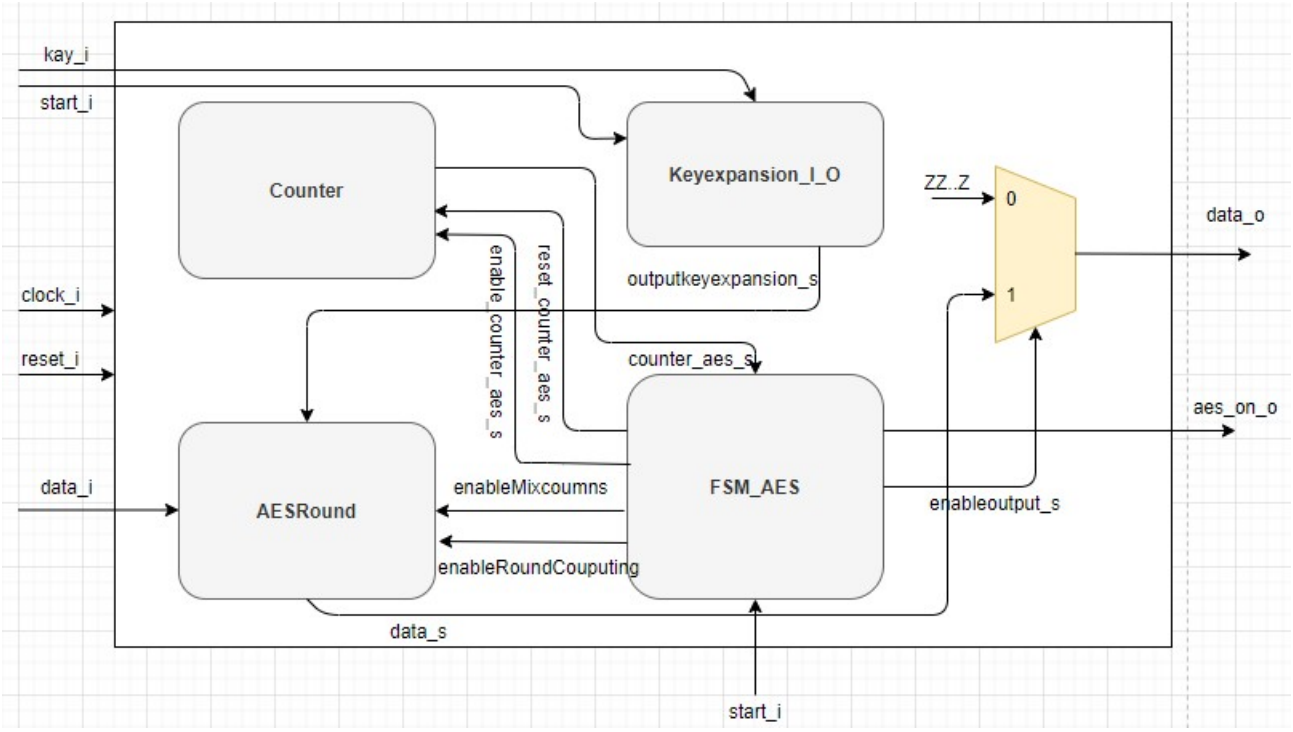


FIGURE 18 – AES

simulé notre composant AES et les résultats montre la bon fonctionnalité de l'algorithme après la comparaison avec les résultats de cours :

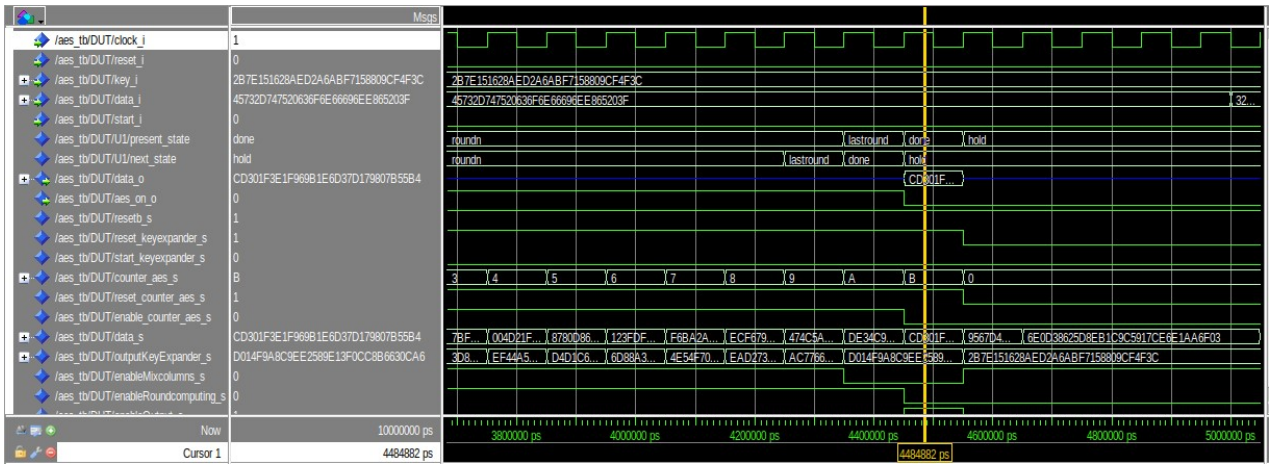


FIGURE 19 – Simulation du chiffrement pour le message de BOB

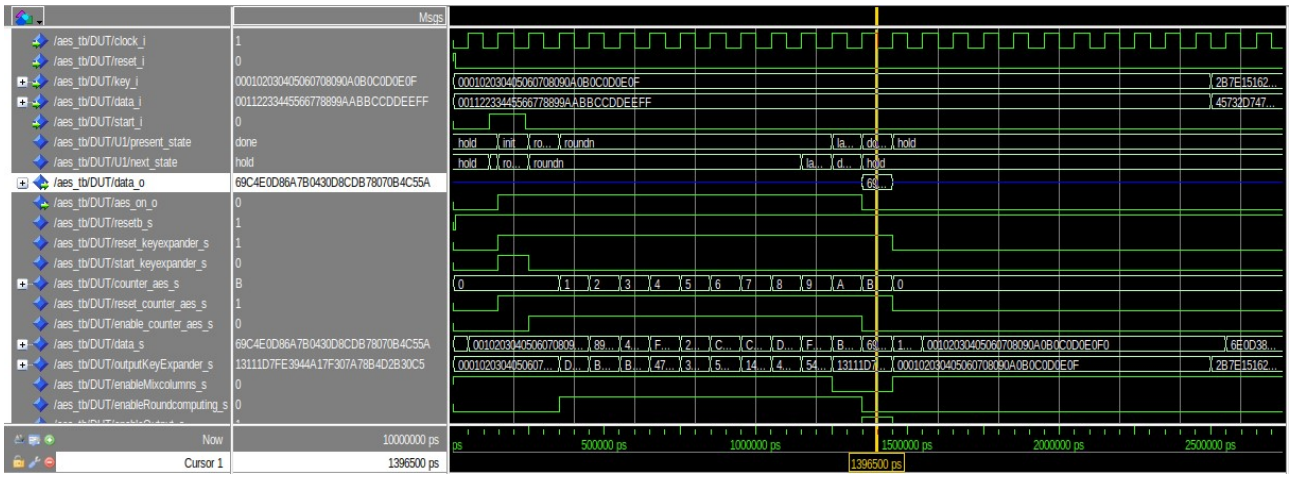


FIGURE 20 – Simulation du chiffrement : un exemple du standard

3 Conclusion

J'ai trouvé le projet de conception AES très intéressant, il me permet de bien assimilé le cours de VHDL et toutes les fonctions qui nous avons vu dans le cours, en plus ce projet c'est une application pratique de cours de sécurité des systèmes embarqués, il me permet aussi d'acquérir une méthodologie pratique de réalisation d'un projet avec langage de description matériel on respecte toutes les exigences du client.

Pour les difficultés qui j'ai rencontré durant le projet il y a le respect des règles d'écriture pour adopter une composante à une composante déjà faite, et le produit matriciel dans l'espace GF pour le MicColumns.

Références

[1] http://aescryptography.blogspot.com/2012/04/shiftrows-step.html. Accessed : 2020-12-15.

[2] https://en.wikipedia.org/wiki/Advanced_Encryption_Standard. Accessed : 2020-12-18.

[3] CNAM LIMOGES. Introduction aux techniques de chiffrement et de securite. 2003-2004.