Projet Conception d'un Système Numérique

Modélisation VHDL de l'algorithme de chiffrement AES

Table des matières

Table des figures

Liste des tableaux

1 Introduction - Chiffrement par bloc

Bob et Alice souhaite garder secrète leur conversation vis-à-vis d'Eve jalouse. Ils décident donc de s'envoyer des messages dont ils seront les seuls à comprendre (cf. exemple de Table 1). Dans ce but, ils se basent sur la théorie de la cryptographie pour chiffrer/déchiffrer ces messages. Parmi les solutions existantes, ils choisissent un algorithme de chiffrement symétrique par bloc réputé mathématiquement inviolable : AES [1]. Dans le cas de l'AES, la taille des blocs de données est fixée à 16 octets soit 128 bits. La clé peut être néanmoins de taille 128 bits, 192 bits ou 256 bits.
L'objectif de ce projet est de développer en VHDL l'algorithme de chiffrement AES avec une clé de 128 bits.

Clé	2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c
Message	Resto en ville ?
Message en ASCII (Hexadecimal)	52 65 73 74 6f 20 65 6e 20 76 69 6c 6c 65 20 3f
Message chiffré (Hexadécimal)	d6 ef a6 dc 4c e8 ef d2 47 6b 95 46 d7 6a cd f0
Message chiffré (ASCII)	Öï|ÜLèïÒGkFxjíõ

Table 1 – Message envoyé par bob et reçu chiffré par Alice

2 Description de l'AES 128 bits

L'algorithme AES se compose de 4 fonctions élémentaires :AddRoundKey, SubBytes, ShiftRows et MixColumns détaillées dans la figure 1. L'algorithme se déroule en 11 rondes : 1 ronde initiale suivie de 9 rondes exécutant ces 4 fonctions et 1 dernière ronde ne comprenant pas l'appel à la fonction *MixColumns*.

Un bloc de données sur 16 octets (i.e. 128 bits) forme le message à chiffrer en entrée de l'algorithme. Une clé ***secrète*** utilisée lors de la ronde initiale et dérivée ensuite dans les rondes suivantes constitue un paramètre de l'algorithme AES. Une fonction supplémentaire est néanmoins nécessaire pour la génération des clés de ronde. Cette fonction est appelée *Key Expansion*.

Description des entrées / sorties

Dans le cadre de ce projet, l'entité AES décrite dans la figure 2 comporte :
- Une entrée 'data_i' de 128 bits
- Une horloge 'clock_i'
- Un signal d'initialisation 'reset_i' actif à l'état haut
- Un signal 'start_i' indiquant la présence d'un message à chiffrer en entrée
- Une sortie 'data_o' sur 128 bits
- Un signal 'aes_on_o' indiquant un chiffrement en cours du message
- Une clé sur 128 bits 'key_i'

Chronogramme du fonctionnement de l'AES

Le chronogramme de l'entité est donné dans la figure 3. Il n'inclut pas les contraintes de *timing* de l'horloge comme des relations entre signaux de commande et données présentes sur les bus. Les signaux évolueront de manière synchrone avec l'horloge.

2.1 Formalisme et notation

Pour être en adéquation avec la norme, vous utiliserez les notations suivantes :
- La numérotation des indices de la Figure 4
- La représentation des octets (cf. Figure 5) avec
- Un état peut également être considéré comme un tableau de colonnes, chaque colonne comprenant 4 octets soit un mot (word) de 32 bits (cf. Figure 6)

1. Advanced Encryption Standard

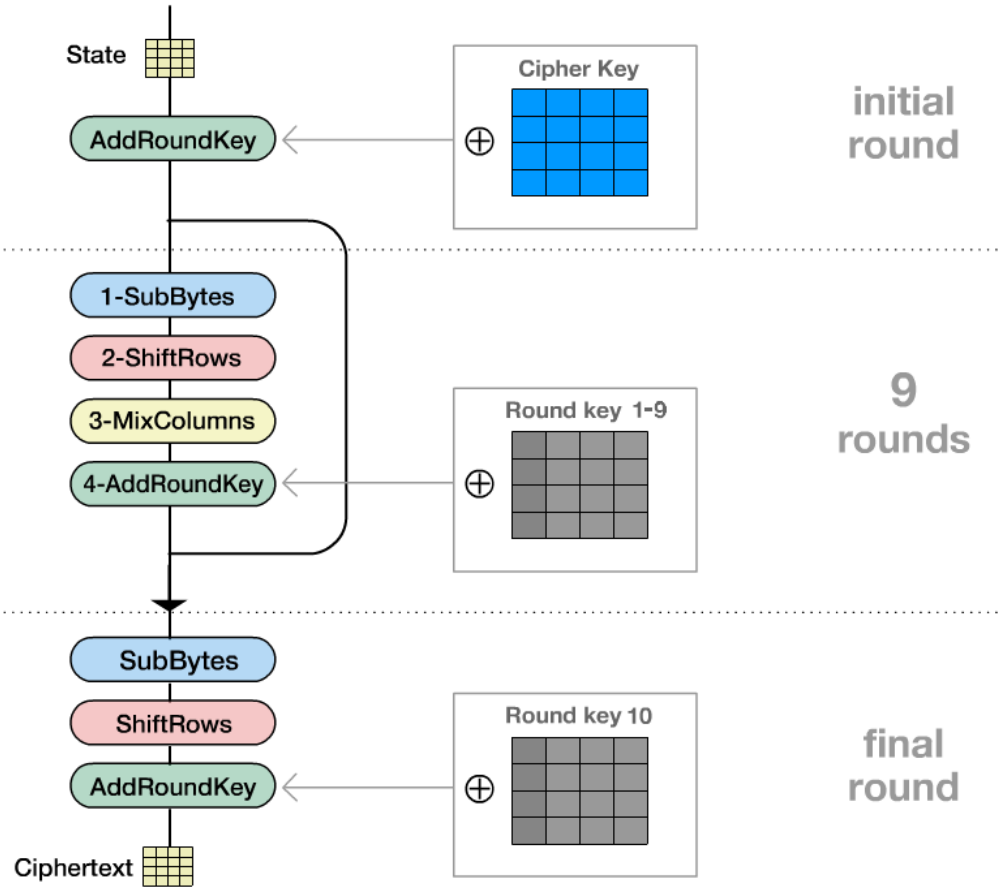

FIGURE 1 – Chiffrement AES 128bits

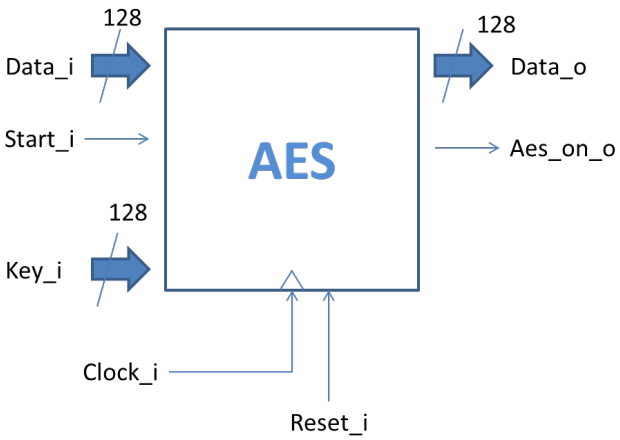

FIGURE 2 – Entrées/Sorties de l'entité AES

2.2 La fonction SubBytes

La fonction SubBytes consiste en une transformation non linéaire appliquée à tous les octets de l'état en utilisant une table de substitution appelée S-Box (cf. Figure 7). Vous utiliserez la S-Box de la figure 8 lors du développement de ce projet. Si une implantation sous la forme d'une mémoire (i.e. un tableau en VHDL) est une possibilité, une autre implantation sous la forme de fonctions booléennes est donnée dans [1].
Pour exemple, soit $S_{2,1} = \{0x53\}$, l'application de la fonction SubBytes donnera l'écriture de $S'_{2,1} = \{0xed\}$.

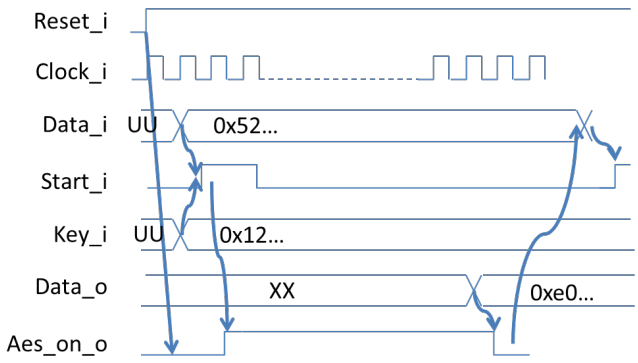

Figure 3 – Evolution des signaux de l'entité AES

Input bit sequence	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	...
Byte number	0								1								2								...
Bit numbers in byte	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	...

Figure 4 – Numérotation des vecteurs de bits

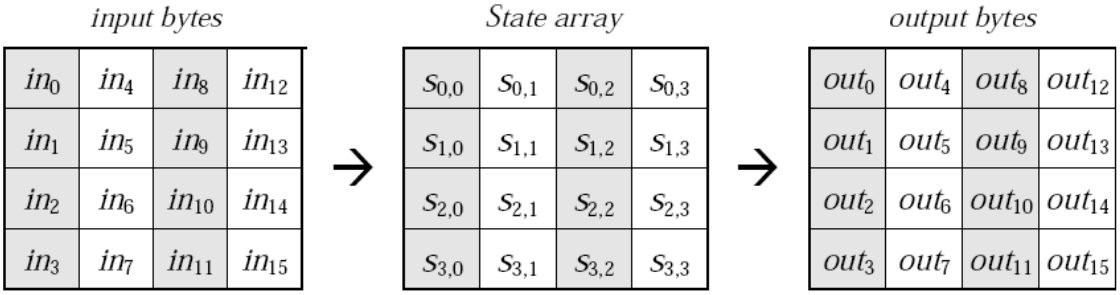

Figure 5 – Représentation de la matrice d'état de l'AES

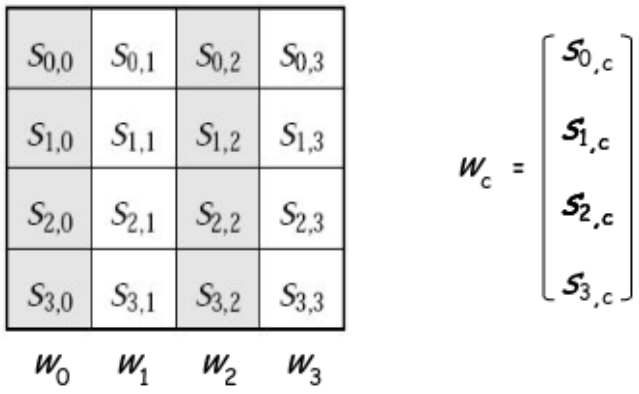

Figure 6 – Représentation d'une colonne de la matrice d'état

2.3 La fonction ShiftRows

La fonction ShiftRows effectue une permutation cyclique (i.e. rotation) des octets des lignes de l'état. Le décalage des octets correspond à l'indice de la ligne considérée ($0 \leq r < 4$). Ainsi, dans l'hypothèse ou $S_{1,x} = \{0x23, 0xeb, 0x5f, 0x99\}$, l'application de la fonction donnera $S'_{1,x} = \{0xeb, 0x5f, 0x99, 0x23\}$.

Une illustration de la fonctionnalité est détaillée dans la Figure 9.

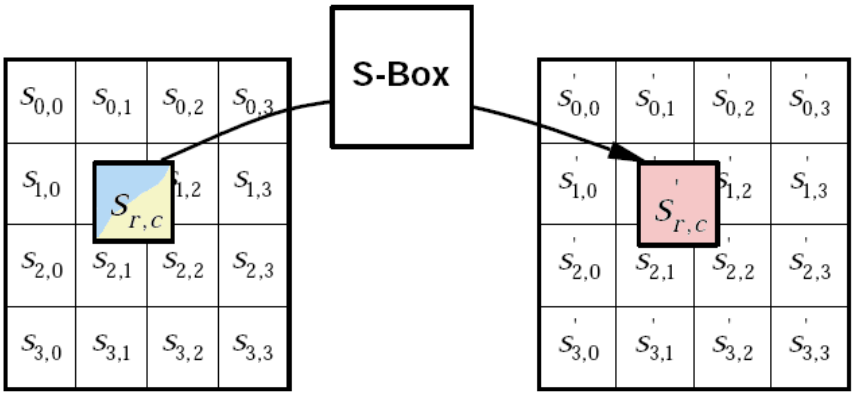

Figure 7 – Principe de la fonction SubBytes

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

Figure 8 – Table de substitution utilisé pour le projet

2.4 La fonction MixColumns

La fonction MixColumns applique une transformation colonne après colonne à une état. Cette transformation linéaire d'un produit matriciel utilisant les 4 octets d'une colonne (cf. Figure 10). Les colonnes sont traitées comme des polynômes dans GF(2^8) [2] et multipliées modulo $x^8+x^4+x^3+x+1$ (noté $\otimes$) avec les polynômes fixes de la Figure 10.
Ces opérations réalisées dans le champs de Gallois de Rijndael sont calculées pour les additions à l'aide d'une fonction XOR. Illustrons la multiplication sur 2 exemples :

— Soit $S_{0,c} = 0xD4 = b'11010100$, $\{02\} \otimes S_{0,c}$ s'écrit sous la forme polynomiale :

$$(x^7+x^6+x^4+x^2)\cdot x = x^8+x^7+x^5+x^3$$

A la lecture de *Cryptography and Network Security* [2], la multiplication polynômiale par x peut être implanté à l'aide d'un décalage à gauche suivi d'un ou-exclusif avec la valeur b'100011011 conditionné par le bit de poid fort du polynôme. Ainsi nous écrivons l'opération telle que :

$$\begin{aligned} \{02\} \otimes S_{0,c} &= (\{02\} \odot S_{0,c}) \ mod \ b'100011011 && (1)\\ &= shiftleft(b'11010100) \ xor \ b'100011011 \ (0xD4_7 = 1 \Rightarrow ou-exclusif) && (2)\\ &= b'110101000 \ xor \ b'100011011 && (3) \end{aligned}$$

2. Galois Field ou champs de Galois de 2^8 = corps commutatif fini de cardinal 256

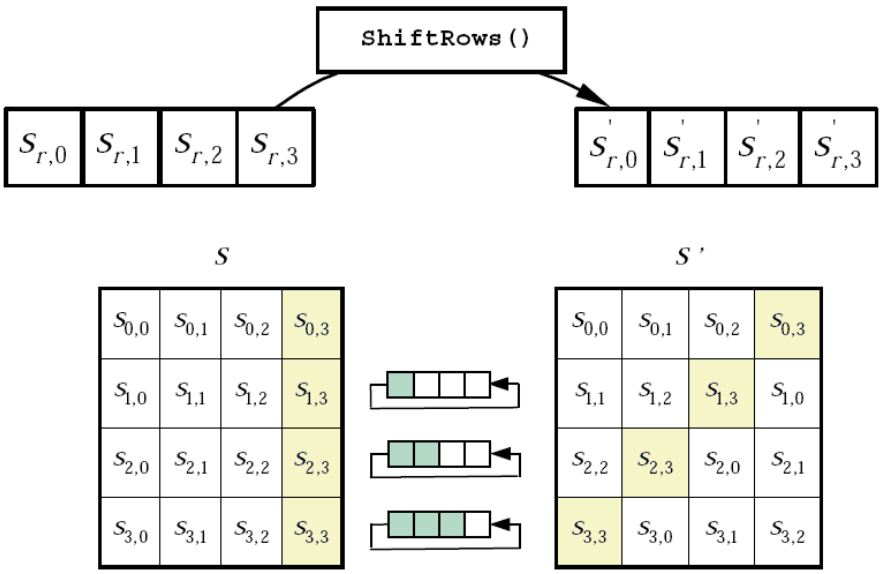

Figure 9 – Illustration de la fonction ShiftRows

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

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

Figure 10 – Produit matriciel de la fonction MixColumns

$$= \quad b'10110011 = 0xB3 \tag{4}$$

— Soit $S_{0,c} = 0x04 = b'00000100$, $\{02\} \otimes S_{0,c}$ s'écrit sous la forme polynomiale :

$$(x^2) \cdot x = x^3$$

idem, nous écrivons l'opération telle que :

$$\{02\} \otimes S_{0,c} = (\{02\} \cdot S_{0,c}) \; mod \; b'100011011 \tag{5}$$
$$= shiftleft(b'00000100) \tag{6}$$
$$= b'00001000 = 0x08 \tag{7}$$

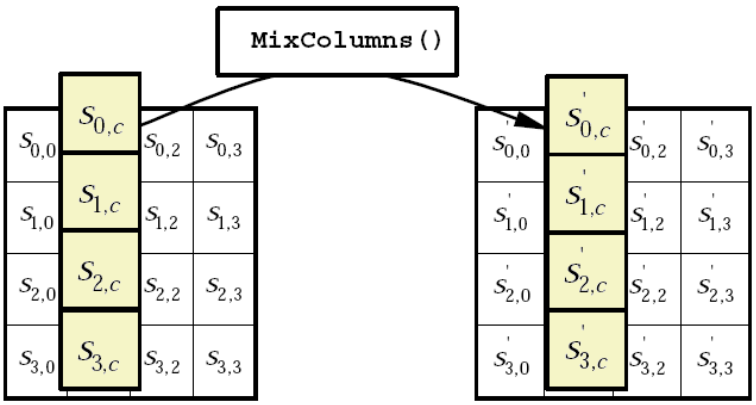

Figure 11 – Application de la fonction MixColumns sur les colonnes de l'état courant

Soit $W_2 = \{0x14, 0xE5, 0xFF, 0x00\}$, la sortie de la fonction MixColumns pour $S'_{0,2}$ sera :

$$S'_{0,2} = (\{02\} \otimes S_{0,2}) + (\{03\} \otimes S_{1,2}) + S_{2,2} + S_{3,2} \tag{8}$$

$$= (\{02\} \otimes S_{0,2}) \; xor \; (\{10 \; xor \; 01\} \otimes S_{1,2}) \; xor \; S_{2,2} \; xor \; S_{3,2} \quad (9)$$
$$= (\{02\} \otimes S0,2) \; xor \; ((\{02\} \otimes S_{1,2}) \; xor \; (\{01\} \otimes S_{1,2})) \; xor \; S_{2,2} \; xor \; S_{3,2} \quad (10)$$
$$= 0x28 \; xor \; 0x34 \; xor \; 0xFF \; xor \; 0x00 \quad (11)$$
$$= 0xE3 \quad (12)$$

Avec $\oplus$ (i.e. xor) la fonction booléenne ou-exclusif définit selon :

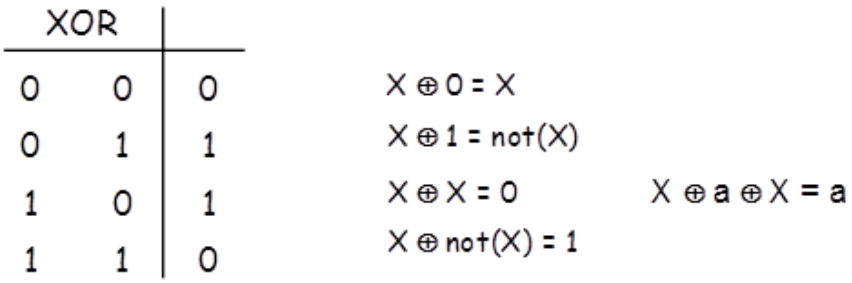

Figure 12 – Fonction Ou-exclusif

2.5 La fonction AddRoundKey

la fonction AddRoundKey ajoute la clef de ronde courante (Ronde 0 à Ronde 10) à l'état ; l'addition étant prise au sens ou-exclusif. Par conséquent, la fonction booléenne XOR est appliquée bit à bit entre les octets de l'état et les octets de la clef de ronde. La figure 13 montre l'application de la fonction.

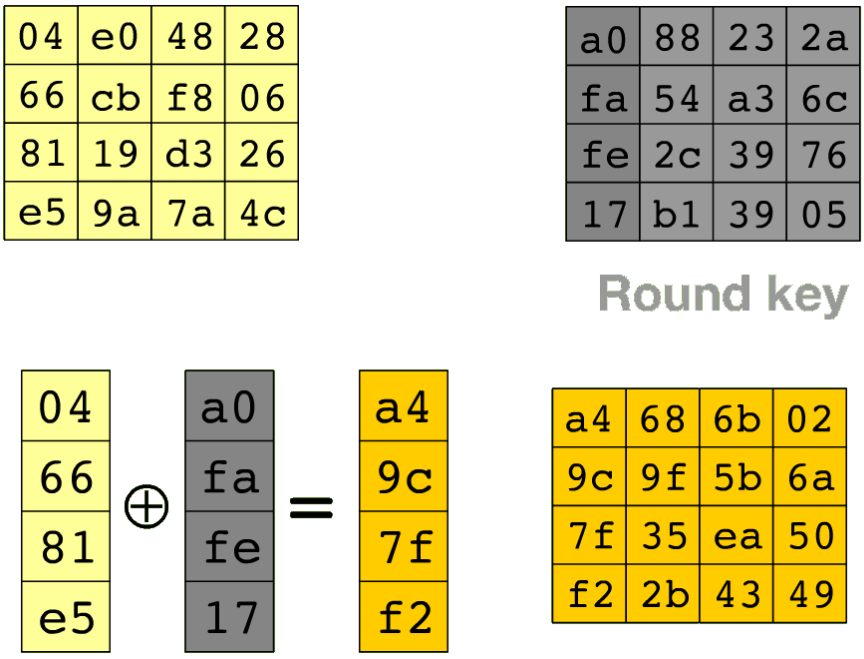

Figure 13 – Calcul de la fonction AddRoundKey

A noter que la clef de la ronde 0 est la clef de chiffrement alors que les clefs des rondes suivantes sont issues d'un processus de diversification des clefs dénommé 'Key Expansion'.

2.6 La fonction KeyExpansion

La génération des clefs de ronde s'applique à partir de la ronde 1. Les clefs sont issues d'un processus faisant appel à une fonction de rotation appliquée sur une colonne (RotWord), de l'appel à la table de substitution (SubBytes) et la fonction booléenne XOR. Il répond à l'algorithme de la figure 14.

```
KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
begin
   word  temp

   i = 0

   while (i < Nk)
      w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
      i = i+1
   end while

   i = Nk

   while (i < Nb * (Nr+1)]
      temp = w[i-1]
      if (i mod Nk = 0)
         temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
      else if (Nk > 6 and i mod Nk = 4)
         temp = SubWord(temp)
      end if
      w[i] = w[i-Nk] xor temp
      i = i + 1
   end while
end
```

Figure 14 – Algorithme de diversification des clefs avec Nk = 4, Nr = numéro de ronde et Nb = 4

Un exemple de calcul de la clef pour la première de ronde est détaillé dans la figure 15.

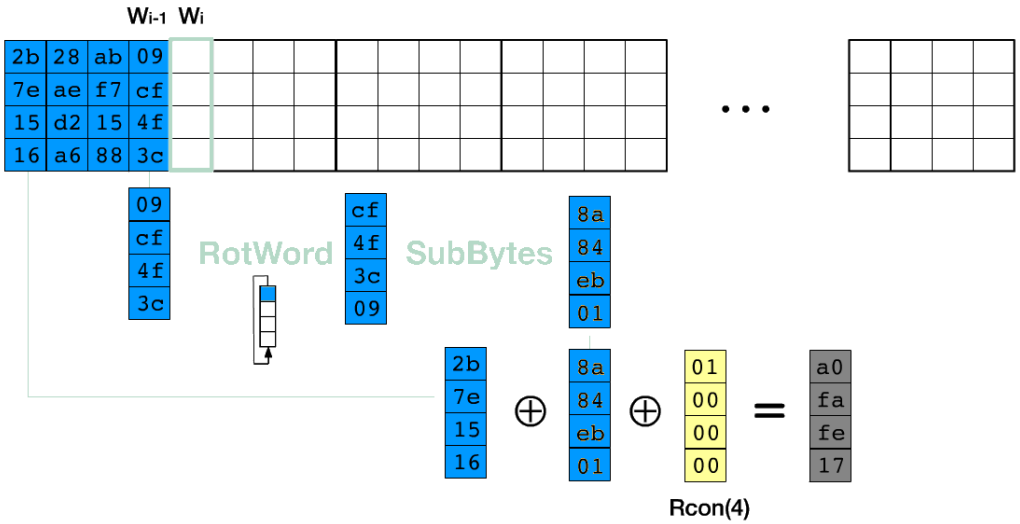

Figure 15 – Calcul de la première colonne (i mod Nk = 0) à l'aide de la matrice Rcon ci-dessous

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

Rcon

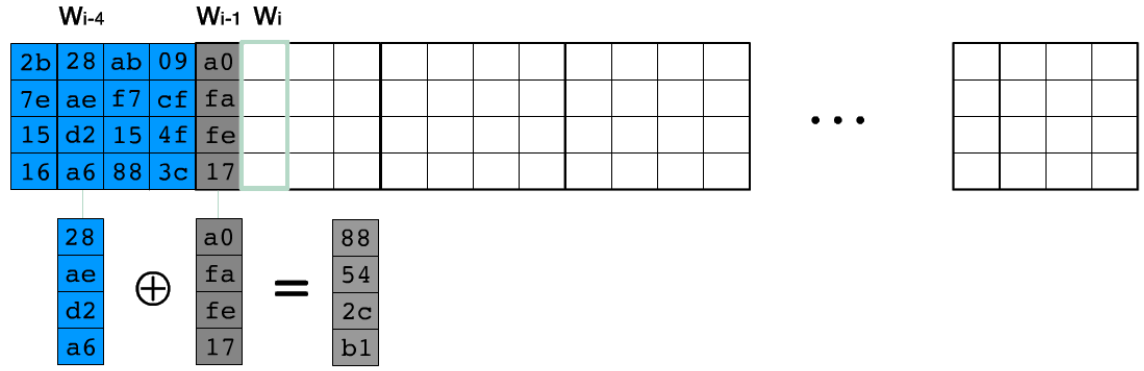

Figure 16 – Calcul de la deuxième colonne

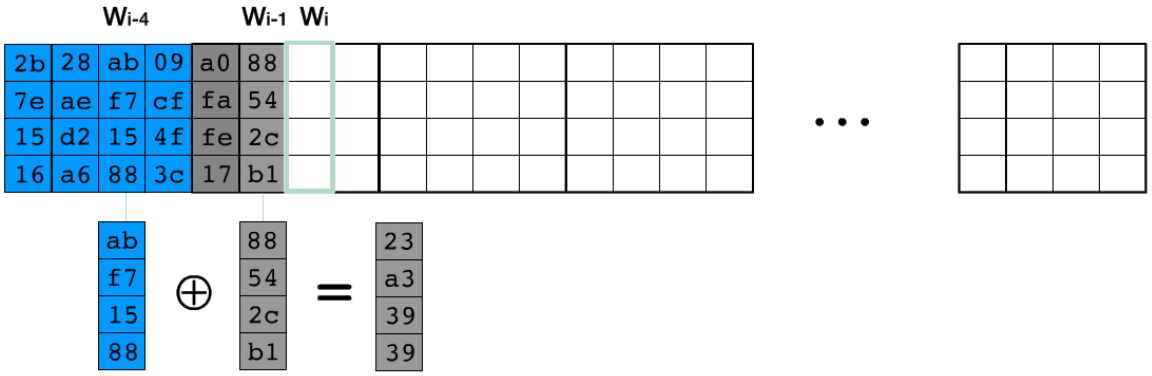

Figure 17 – Calcul de la troisième colonne

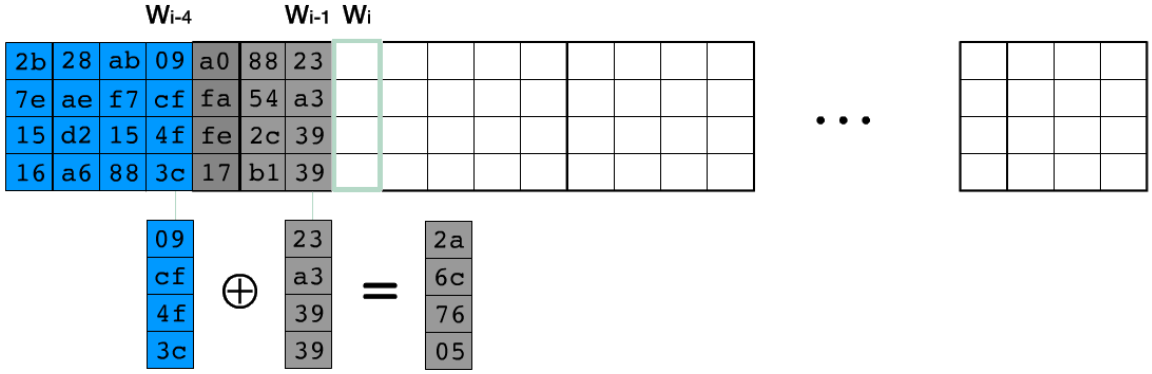

Figure 18 – Calcul de la quatrième colonne

Figure 19 – Production de toutes les clefs de ronde

2.7 Architecture globale de l'AES

L'architecture complète de l'AES est décrite dans la figure 20. Elle contient l'entité *Key Expander* ainsi que la machine d'états (ou Finite State Machine (FSM)) et un composant *Round* intégrant les fonctions de calcul de l'état courant.

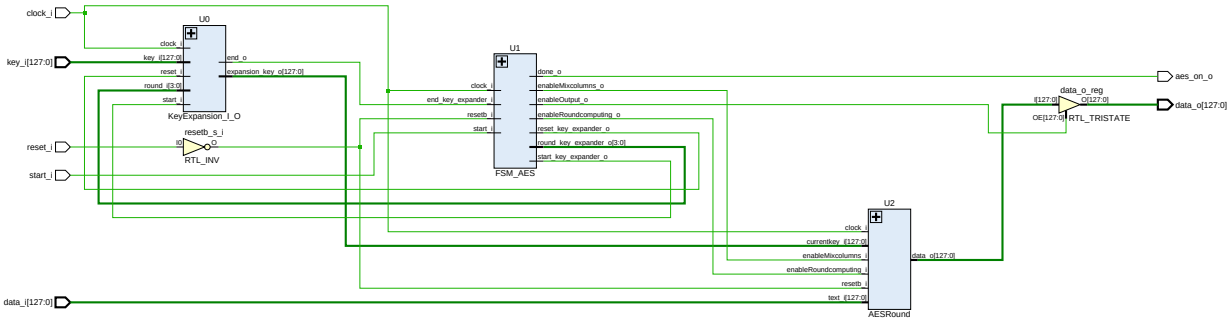

Figure 20 – Architecture globale de l'AES

L'entité *Round* instanciée dans l'entité AES peut s'organiser comme décrit dans la figure 21. Le chemin de donnée dépendra de votre représentation de l'état courant.

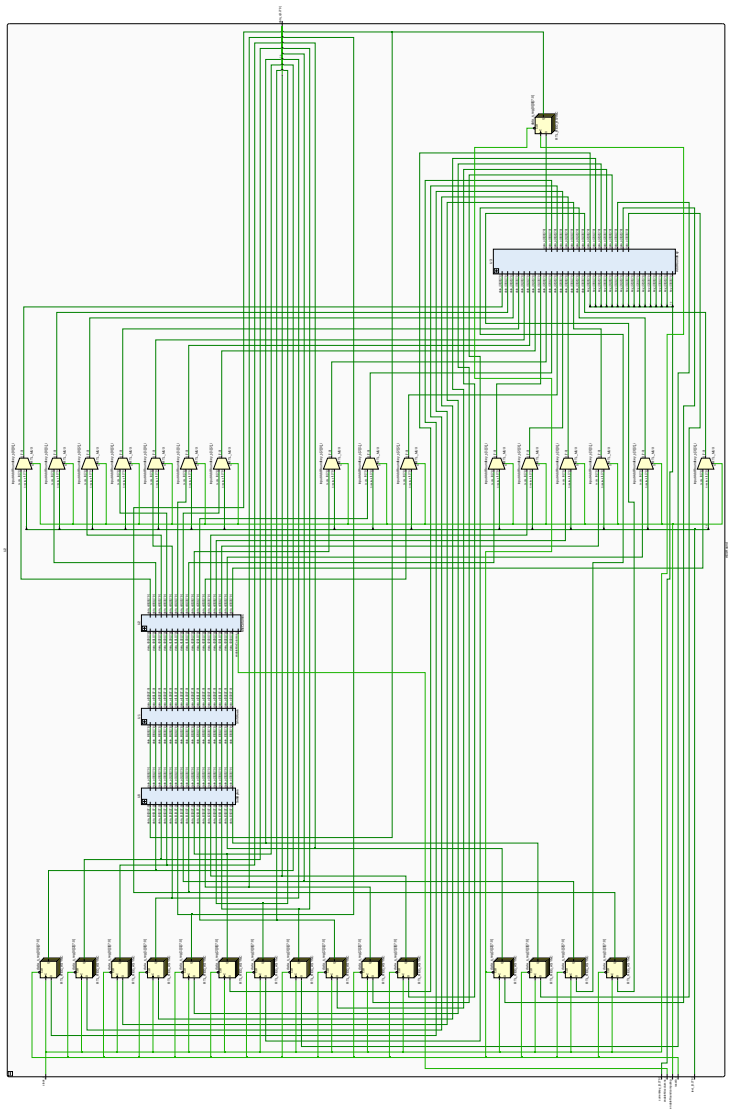

Figure 21 – La fonction RoundExe intégrant SubBytes, ShiftRows, MixColumns et AddRoundKey

3 Organisation du travail

3.1 Planning

Vous disposez de 5 séances pour modéliser l'entité AES en VHDL. Chacune des séances sera dédiée à la réalisation d'une composante de l'entité et de sa validation par son test. Il est par conséquent important de terminer cette composante avant le début de la séance suivante. Le découpage des séances est proposé comme suit :

1. Lecture et compréhension du sujet / Conception de la fonction *S-BOX* et de la fonction *SubBytes*
2. Conception des modules *ShiftRows* et *AddRoundKey*
3. Conception de la fonction *MixColumns*
4. Conception de la fonction *FSM*
5. Suite de la conception de la fonction *FSM* et intégration de tous les éléments dans l'entité AES (Top level) fournie pour validation complète. La fonction *KeyExpansion* est fournie.

Afin de coller à la réalité d'un projet industriel, le planning de la figure 22 décrit l'évolution que doit suivre votre travail. Ainsi un bilan d'avancement du projet en pourcentage sera demandé pour chacune des séances.

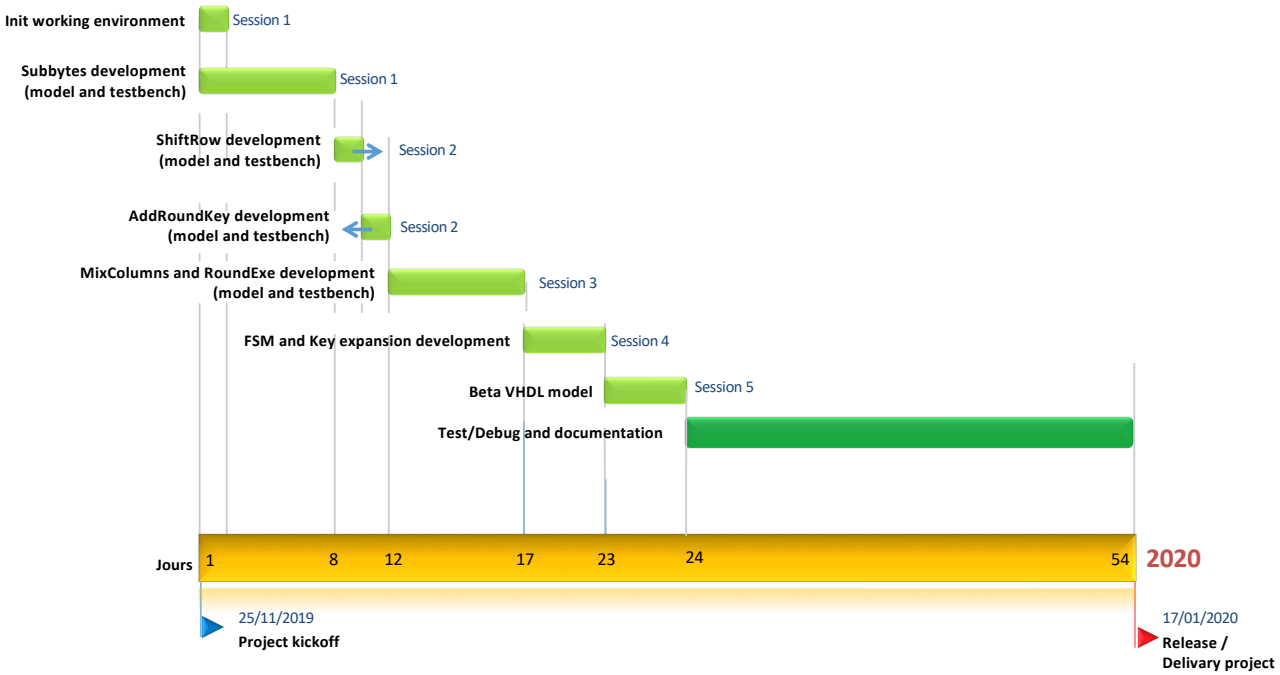

FIGURE 22 – Représentation Gantt du projet

Note : Les dates dépendent de votre groupe (cf. PROMETHEE et CAMPUS)

3.2 Contraintes de développement

Le projet sera développé à l'aide de l'outil Mentor Graphics ModelSim. Les fichiers de description VHDL (xxx.vhd) et de test (xxx_tb.vhd) seront enregistrés dans les répertoires nommés "SRC/RTL" et "SRC/BENCH" respectivement. Les bibliothèques associées seront nommées "LIB_RTL " et "LIB_BENCH" et créées dans un répertoire nommé "LIB" (donc respectivemnt dans les répertoires associés "LIB/LIB_RTL" et "LIB/LIB_BENCH"). Un script automatisant la compilation vous sera demandé.
De plus, et afin de disposer d'un code VHDL lisible, une attention toute particulière sera apportée à l'écriture du code. Ainsi certaines règles devront être respectées :

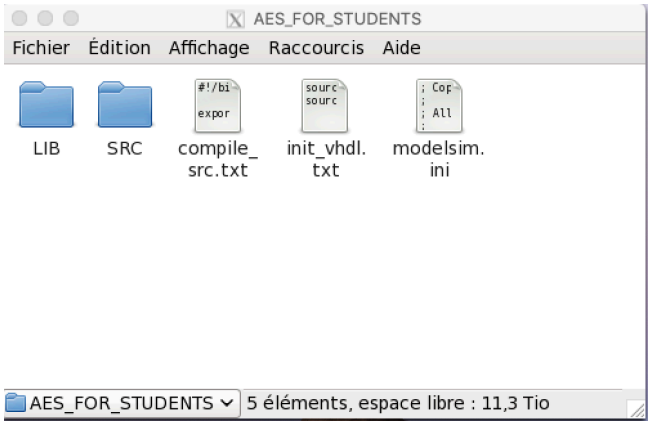

Figure 23 – Organisation des répertoires de travail

1. Une architecture associée à une entité doit se trouver dans le même fichier VHDL. Le nom du fichier correspond au nom de l'entité.
2. Le nom de l'architecture reprend le nom de l'entité complété du suffixe "_arch".
3. Pour les machines d'états précisez si le modèle d'architecture est Mealy ou Moore. Exemple : "FSM_Moore_arch".
4. Les signaux entrant d'une entité doivent être complétés par le suffixe "_i"
5. Les signaux sortant d'une entité doivent être complétés par le suffixe "_o"
6. Les signaux internes d'une architecture doivent être complétés par le suffixe "_s"
7. Les commentaires sont "obligatoires"

3.3 Validation du fonctionnement

A titre d'évaluation du bon fonctionnement de votre développement, les états intermédiaires après chaque étape de calcul de l'AES pour le chiffrement du message "Resto ce soir ?" avec la clé "2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c" vous sont listés ci-dessous.

```
Plain text : (Hex) 52 65 73 74 6f 20 65 6e 20 76 69 6c 6c 65 20 3f
(ASCII) Resto en ville ?

InitKey : 2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c
SetPlaintext : 52 65 73 74 6f 20 65 6e 20 76 69 6c 6c 65 20 3f
AddRoundKey : 79 1b 66 62 47 8e b7 c8 8b 81 7c e4 65 aa 6f 03
Round 0
SubBytes : af 44 d3 ab 16 e6 20 b1 ce 91 01 ae bc 62 06 d5
ShiftRows : af e6 01 d5 16 91 06 ab ce 62 d3 b1 bc 44 20 ae
MixColumns : a0 ae 2f bc 29 8e 6d e0 43 d5 d9 81 21 fa 51 fc
ComputeKey : 75 ec 78 56 5d 42 aa f0 f6 b5 bf 78 ff 7a f0 44
AddRoundKey : d5 42 57 ea 74 cc c7 10 b5 60 66 f9 de 80 a1 b8
Round 1
SubBytes : b5 f6 da bb ca 27 31 7c d2 90 d3 69 9c 3a f1 9a
ShiftRows : b5 27 d3 9a ca 90 f1 bb d2 3a da 7c 9c f6 31 69
MixColumns : 51 0f 9a 1f 6e 42 75 49 57 af c3 75 7a 51 b3 aa
ComputeKey : ca fb fe 2b 97 b9 54 db 61 0c eb a3 9e 76 1b e7
AddRoundKey : 9b f4 64 34 f9 fb 21 92 36 a3 28 d6 e4 27 a8 4d
Round 2
SubBytes : e8 ba 8c 28 69 63 7b 74 24 71 ee 4a ae 3d 6f 65
ShiftRows : e8 63 ee 65 69 71 6f 28 24 3d 8c 74 ae ba 7b 4a
MixColumns : e5 62 e3 64 06 12 be f5 f7 a5 86 35 a3 06 3c bc
```

```
ComputeKey : c1 bf 4e f4 56 06 1a 2f 37 0a f1 8c a9 7c ea 6b
AddRoundKey : 24 dd ad 90 50 14 a4 da c0 af 77 b9 0a 7a d6 d7
Round 3
SubBytes : a6 c9 18 96 6c 9b 1d 7a 1f 1b 02 db a3 bd 4a 0d
ShiftRows : a6 9b 02 0d 6c 1b 4a 96 1f bd 18 7a a3 c9 1d db
MixColumns : ee 80 2e 72 29 12 42 d2 80 2c 1c 70 db d6 26 87
ComputeKey : c8 04 4b 43 9e 02 51 6c a9 08 a0 e0 00 74 4a 8b
AddRoundKey : 26 84 65 31 b7 10 13 be 29 24 bc 90 db a2 6c 0c
Round 4
SubBytes : 23 4f bc 2e 20 7c 82 5a 4c a6 78 96 9f 1a b8 81
ShiftRows : 23 7c 78 81 20 a6 b8 2e 4c 1a bc 5a 9f 4f 82 96
MixColumns : 3b d2 37 78 27 8a 9f 22 50 fd db c6 e0 0a 6e 40
ComputeKey : 12 58 85 11 8c 5a d4 7d 25 52 74 9d 25 26 3e 16
AddRoundKey : 29 8a b2 69 ab d0 4b 5f 75 af af 5b c5 2c 50 56
Round 5
SubBytes : 4c cf 3e e4 0e 60 cc 84 3f 1b 1b 57 07 42 6c b9
ShiftRows : 4c 60 1b b9 0e 1b 6c e4 3f 42 3e 84 07 cf cc 57
MixColumns : 9a 18 ca c6 b9 68 fa b6 02 7d 96 2e df 9a b2 a4
ComputeKey : 11 89 7a d3 9d d3 ae ae b8 81 da 33 9d a7 e4 25
AddRoundKey : 8b 91 b0 15 24 bb 54 18 ba fc 4c 1d 42 3d 56 81
Round 6
SubBytes : ce ac fc 2f a6 fe fd 34 c0 55 5d de f6 8b b9 91
ShiftRows : ce fe 5d 91 a6 55 b9 2f c0 8b fc 34 f6 ac fd de
MixColumns : 52 5f 22 d3 3e f3 eb 43 d5 e6 f4 44 3b 77 c2 f7
ComputeKey : d8 27 b8 a6 45 f4 16 08 fd 75 cc 3b 60 d2 28 1e
AddRoundKey : 8a 78 9a 75 7b 07 fd 4b 28 93 38 7f 5b a5 ea e9
Round 7
SubBytes : cf c1 37 3f 03 38 21 cc ee 22 76 6b 57 29 bb eb
ShiftRows : cf 38 76 eb 03 22 bb 3f ee 29 37 cc 57 c1 21 6b
MixColumns : 50 ce 3d c9 e4 ae 0d e2 47 29 e6 b4 bc c6 69 cf
ComputeKey : 27 c9 51 36 62 3d 47 3e 9f 48 8b 05 ff 9a a3 1b
AddRoundKey : 77 07 6c ff 86 93 4a dc d8 61 6d b1 43 5c ca d4
Round 8
SubBytes : 02 38 b8 7d dc 22 5c 93 2d d8 b3 56 64 a7 10 19
ShiftRows : 02 22 b3 19 dc d8 10 7d 2d a7 b8 93 64 38 5c 56
MixColumns : c8 91 76 a5 bd 3a a3 4d 83 38 4f 55 8a a6 1e 64
ComputeKey : 0b b8 15 4b 69 85 52 75 f6 cd d9 70 09 57 7a 6b
AddRoundKey : c3 29 63 ee d4 bf f1 38 75 f5 96 25 83 f1 64 0f
Round 9
SubBytes : 33 4c 00 99 19 f4 2b 76 3f 77 35 c2 41 2b 8c fb
ShiftRows : 33 f4 35 fb 19 77 8c 99 3f 2b 00 76 41 4c 2b c2
ComputeKey : e7 05 10 0b 8e 80 42 7e 78 4d 9b 0e 71 1a e1 65
AddRoundKey : d4 f1 25 f0 97 f7 ce e7 47 66 9b 78 30 56 ca a7
GetCiphertext : d4 f1 25 f0 97 f7 ce e7 47 66 9b 78 30 56 ca a7
Cipher text at the end: (Hex) d4 f1 25 f0 97 f7 ce e7 47 66 9b 78 30 56 ca a7
(ASCII) ‘Ù%ó˜ŒÁGfõx0V ß
Cipher text to decipher: (Hex) d4 f1 25 f0 97 f7 ce e7 47 66 9b 78 30 56 ca a7

SetCiphertext : d4 f1 25 f0 97 f7 ce e7 47 66 9b 78 30 56 ca a7
InitKey : 2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c
KeyExpansion (round 0): 2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c
KeyExpansion (round 1): 75 ec 78 56 5d 42 aa f0 f6 b5 bf 78 ff 7a f0 44
KeyExpansion (round 2): ca fb fe 2b 97 b9 54 db 61 0c eb a3 9e 76 1b e7
```

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

```
InvMixColumns : af e6 01 d5 16 91 06 ab ce 62 d3 b1 bc 44 20 ae
Round 0
InvShiftRows : af 44 d3 ab 16 e6 20 b1 ce 91 01 ae bc 62 06 d5
InvSubBytes : 79 1b 66 62 47 8e b7 c8 8b 81 7c e4 65 aa 6f 03
AddRoundKey : 52 65 73 74 6f 20 65 6e 20 76 69 6c 6c 65 20 3f
Getplaintext : 52 65 73 74 6f 20 65 6e 20 76 69 6c 6c 65 20 3f
Obtained plain text : (Hex) 52 65 73 74 6f 20 65 6e 20 76 69 6c 6c 65 20 3f
(ASCII) Resto en ville ?
```

La simulation de l'AES pour le chiffrement du message de Bob est donnée dans la figure 24 ci-dessous.

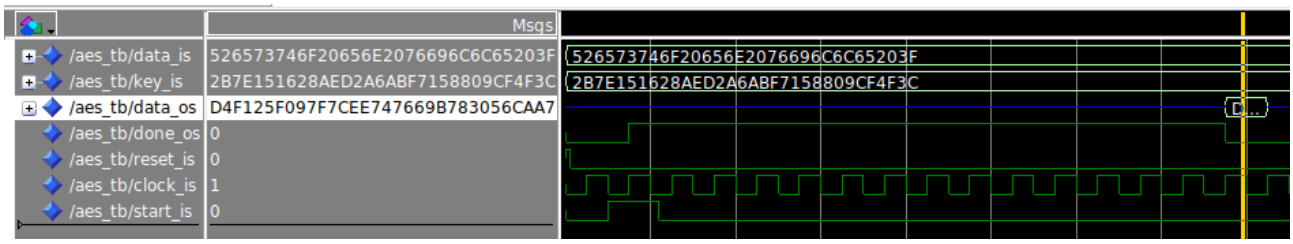

Figure 24 – Simulation du chiffrements

3.4 Livrables

La note sera composée des notes obtenues aux questionnaires et du PV de réception du projet complet. Le PV de réception sera validé après l'envoi par l'élève d'un fichier compressé (.zip ou .tgz) à la date prévue par le Gantt. Il sera constitué :

1. Un rapport au format pdf contenant la description de votre entité AES et le chronogramme résultant de sa simulation
2. La description de la machine d'états gérant le chiffrement
3. Les chronogrammes (et leur interprétation) issus de la simulation des éléments constituant l'AES
4. La description des points bloquants et les solutions envisagées
5. Les codes VHDL répartis dans les répertoires "SRC/RTL/" et "SRC/BENCH"
6. Le script de compilation de ces codes VHDL

La notation sera réalisée selon le barème suivant :

Correction PCSN (ISMIN 2018)		Modules						Règles d'écritures	Consignes VHDL		Rapport			
		SubBytes	ShiftRows	AddRoundkey	MixColumn	FSM	Top		Script	Organisation	Contenu	Orthographe		
Nom	Prénom	1 pts	1 pts	1 pts	2 pts	3 pts	3 pts	5 pts	1 pts	3 pts	7 pts	3 pts	Note / 30	Note / 20

Figure 25 – Notation du projet PCSN

Note 1 : Toute ressemblance avec un projet existant ou copié sur un autre élève sera sanctionnée d'un 0/20

Note 2 : Il n'y a pas de rattrapage pour cette UP

Références

[1] J. Boyar and R. Peralta, *A Small Depth-16 Circuit for the AES S-Box*, pp. 287–298. Berlin, Heidelberg : Springer Berlin Heidelberg, 2012.

[2] W. Stallings, *Cryptography and Network Security : Principles and Practice*. Upper Saddle River, NJ, USA : Prentice Hall Press, 5th ed., 2010.

[3] NIST, "Fips-197, announcing the advanced encryption standard (aes),"