MINES Saint-Étienne | Institut Mines-Telecom

Rapport de conception technique

Conception de Système Numérique

Modélisation SystemVerilog de l'algorithme de chiffrement ASCON

Yasmine SALMOUNI
yasmine.salmouni@etu.emse.fr

09 mai 2024

INSPIRING INNOVATION SINCE 1816

Table des matières

1 Introduction - Chiffrement authentifié avec données associées

Dans un monde de plus en plus connecté, la sécurité des communications numériques est importante. ASCON128, un algorithme de chiffrement authentifié avec données associées (AEDAD), répond à cette nécessité et protège les informations échangées sur internet. ASCON128 est un

algorithme qui permet non seulement de de chiffrer le contenu d'un message garantissant que seul le destinataire peut le lire (confidentialité), mais il vérifie aussi que le message n'a pas été altéré en cours de route grâce à un "tag" d'authenticité. Ainsi, cet algorithme sécurise les données sensibles tout en assurant que l'identité des communicateurs est vérifiée. ASCON128 a été sélectionné comme l'un des gagnants du concours CAESAR (Competition for Authentificated Encryption : Security, Applicability, and Robustness). Ce rapport présente le conception d'un

système numérique destiné à modéliser l'algorithme de chiffrement ASCON128. Dans ce projet, nous nous concentrons sur le développement d'une architecture numérique qui implémente une version simplifiée de cet algorithme, en utilisant le langage de modélisation SystemVerilog.

2 Description de l'algorithme ASCON128

2.1 Formalisme et notations

2.1.1 L'état courant S

L'algorithme agit sur un état courant (state S), qui a une taille de 320 bits. Un state S peut être vu comme un tableau de cinq registres de 64 bits chacun (Figure 12) ou 64 colonnes de 5 bits chacune (Figure 2).

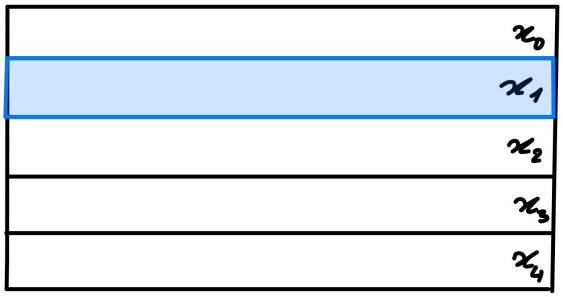

FIGURE 1 – L'état courant S représenté comme un tableau de registres x_i

FIGURE 2 – L'état courant S représenté comme un tableau de colonnes

L'état se compose de deux parties :

- Une partie externe de $r = 64$ bits, $S_r = S_{64}$.
- Une partie interne de $c = 256$ bits, $S_c = S_{256}$.

2.1.2 notations utilisées dans la version de l'algorithme ASCON128 présentée dans ce rapport

Nous utiliserons les notations suivantes pour implémenter la version simplifiée de l'algorithme ASCON présentée dans ce rapport :

- K : Clé secrète de **128 bits**.
- N : Nombre arbitraire (nonce N) de **128 bits**.
- T : Tag (nonce N) de **128 bits**.
- P : Texte clair (plaintext) de **184 bits**.
- C : Texte chiffré (ciphertext) de **184 bits**.
- A : Données associées (associated data) de **48 bits**.
- IV : Vecteur d'initialisation (initialisation vector) de **64 bits**. La valeur de IV est fixée à 0x80400C0600000000.

2.2 Les permutations p^6 et p^{12}

2.2.1 Définition d'une permutation

Une permutation est une transformation cryptographique appliquée à l'état interne S dans le but de mélanger les bits. Les permutations dans ASCON128 se déroulent sur 6 ou 12 itérations (rondes), chacune constituée de trois étapes : l'addition de constante p_C, la couche de substitution p_S et la couche de diffusion p_L.

2.2.2 L'addition de constante p_C

Chaque ronde est caractérisée par une constante appelée : constante de ronde. Au cours de cette étape, à chaque itération, la constante de ronde correspondante est ajoutée (xor) au registre x_2 de l'état S.

Ronde r de p^{12}	Ronde r de p^6	Constante c_r
0		00000000000000000f0
1		00000000000000000e1
2		00000000000000000d2
3		00000000000000000c3
4		00000000000000000b4
5		00000000000000000a5
6	0	0000000000000000096
7	1	0000000000000000087
8	2	0000000000000000078
9	3	0000000000000000069
10	4	000000000000000005a
11	5	000000000000000004b

TABLE 1 – Tableau représentant toutes les constantes de ronde de l'algorithme ASCON128

2.2.3 La couche de substitution p_S

x	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
$S(x)$	04	0B	1F	14	1A	15	09	02	1B	05	08	12	1D	03	06	1C	1E	13	07	0E	00	0D	11	18	10	0C	01	19	16	0A	0F	17

TABLE 2 – Représentation du tableau de substitution (S-box) pour l'algorithme ASCON128

En représentant l'état courant S par un tableau de 64 colonnes de 5 bits (Figure 2), la couche de substitution remplace chaque suite de cinq bits présente dans la première colonne du tableau de substitution par la suite correspondante présente dans le deuxième ligne de la même colonne.

2.2.4 La couche de diffusion linéaire p_L

Cette étape repose sur le principe de la rotation de bits vers la droite, qui est à distinguer du décalage de bits. En effet, contrairement au décalage de bits, la rotation est une opération circulaire, les bits poussés hors des limites du registre réapparaissent de l'autre côté de celui-ci.

$x_0 \ggg 28$

Rotation de 28 bits vers la droite

En représentant l'état courant S par un tableau de 5 lignes de 64 bits (Figure 12), la couche de diffusion linéaire applique la suite d'opération suivante (une diffusion) à chacune des lignes du tableau :

$$
\begin{aligned}
x_0 &\leftarrow \Sigma_0(x_0) = x_0 \oplus (x_0 \ggg 19) \oplus (x_0 \ggg 28) \\
x_1 &\leftarrow \Sigma_1(x_1) = x_1 \oplus (x_1 \ggg 61) \oplus (x_1 \ggg 39) \\
x_2 &\leftarrow \Sigma_2(x_2) = x_2 \oplus (x_2 \ggg 1) \oplus (x_2 \ggg 6) \\
x_3 &\leftarrow \Sigma_3(x_3) = x_3 \oplus (x_3 \ggg 10) \oplus (x_3 \ggg 17) \\
x_4 &\leftarrow \Sigma_4(x_4) = x_4 \oplus (x_4 \ggg 7) \oplus (x_4 \ggg 41)
\end{aligned}
$$

2.3 Détails du processus de chiffrement

Après avoir introduit les notations utilisées dans ce rapport, le concept d'une permutation et les différentes couches qui la composent, intéressons nous d'avantage au processus de chiffrement et d'authentification d'un message.

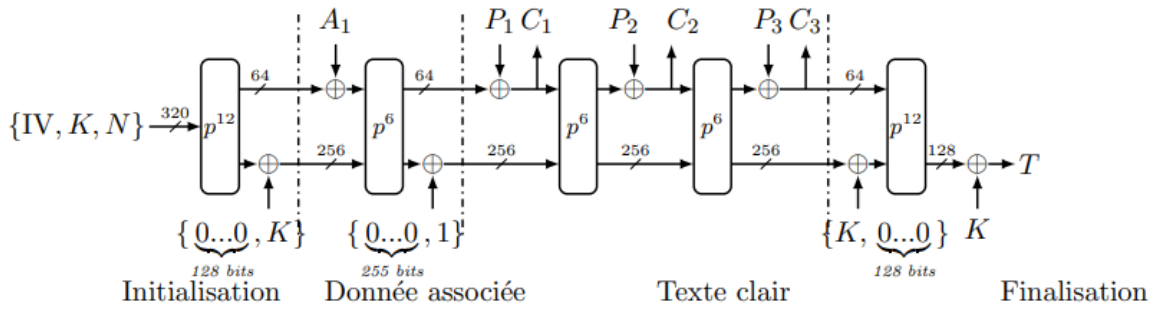

FIGURE 3 – Schéma bloc de la version simplifiée de l'algorithme de chiffrement ASCON128

Le processus de chiffrement est composé en quatre étapes :

— **Initialisation** : L'état initial est créé en combinant le vecteur d'initialisation IV, la clé secrète K, et la nonce N. Il subit ensuite un mélange complexe en utilisant la permutation p^{12}

$$\text{Initialisation} \quad \begin{cases} S \leftarrow (IV, K, N) \\ S \leftarrow p^{12}(S) \oplus \{0, \ldots, 0\}_{192\,\text{bits}}, K \end{cases}$$

— **Donnée associée** : les données associées sont fusionnées avec une partie (64 bits) du state S sortant de l'initialisation via une opération xor, au cours d'une permutation p^6.

— **Texts clair (chiffrement)** : Les parties du texte claire P_1, P_2 et P_3 sont fusionné avec l'état courant via des opérations xor, au cours de deux permutations p^{12}. Les parties du text chiffré C_1, C_2 et C_3 sont récupéré au cours de ces permutations.

— **Finalisation** : le Tag d'authentification du message chiffré est obtenu après une dernière permutation $p_1 2$.
T $\leftarrow \mathsf{S}_{128} \oplus K$

3 Conception du Système Numérique

3.1 Définition d'une machine à états finis

Une machine à états finis est un modèle utilisé pour représenter le fonctionnement d'un système qui peut se trouver dans différents états selon les situations. Elle est constituée de trois éléments :

- **Etats** : Ce sont les différentes conditions dans lequelles le système peut se trouver.
- **Evènements ou entrées** : des actions ou des signaux qui proviennent de l'extérieur du système et qui peuvent provoquer un changemant d'état.
- **Transitions** : ce sont les règles qui définissent comment un système passe d'un état à un autre. Les transitions sont souvent basées sur les évènements.

Il existe deux types de machines d'état : les machines d'état de Moore et les machine d'état de Mealy.

- **Machine à états de Moore** :Les sorties sont uniquement determinées par l'état courant du système. Les transitions entre les états dépendent uniquement des entrées.
- **Machine à états de Mealy** :Les sorties peuvent changer en réponse directe à des entrées et dépendent donc de l'état actuel et des entrées reçues.

3.2 Architecture globale du modèle

Dans l'algorithme étudié, une machine à états de Mealy est utilisée. Elle est vue comme un chef d'orchestre qui pilote les permutations et le compteur de rondes.

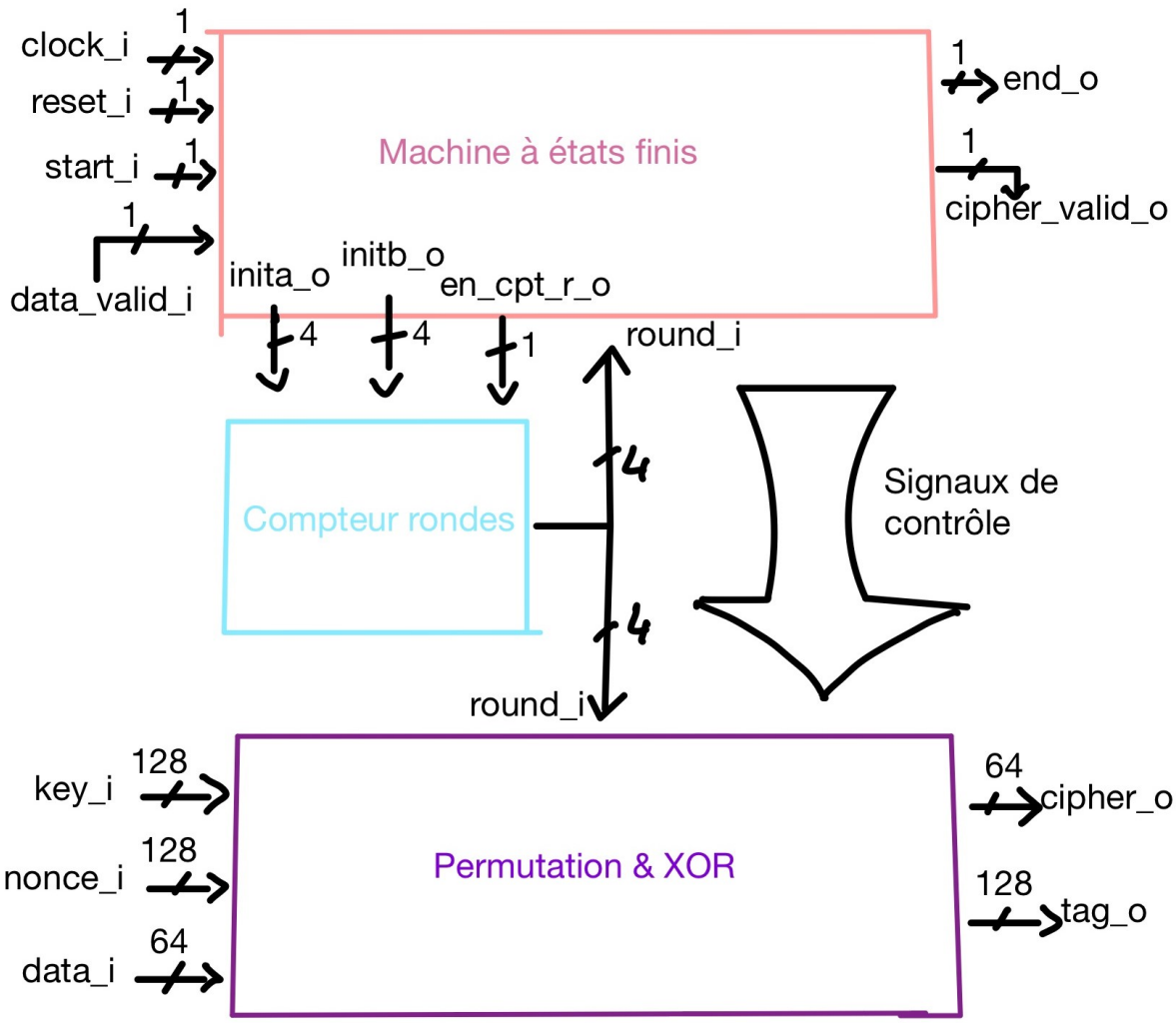

FIGURE 4 – Architecture globale du système

3.3 Permutation et XOR : Description du module de permutation

En SystemVerilog, un module est une unité de base pour la conception de circuits numériques, défini par des interfaces d'entrée et de sortie et pouvant inclure des sous-modules pour des tâches plus complexes. Le module de permutation en SystemVerilog, conçu pour le chiffrement

ASCON128, joue un rôle important dans le processus de chiffrement en manipulant et transformant l'état des données pour garantir leur sécurité. Ce module est composé de plusieurs sous-composants, chacun responsable d'une étape spécifique du processus de permutation. En effet, le module de permutation est constitué d'un multiplexeur (sous-module : *mux_state*), de deux sous-modules XOR (*xor_begin_perm* et *xor_end_perm*), et de trois sous-modules spécifiques à chaque étape de la permutation : addition (*addition_constante*), substitution (*couche_substitution*) et diffusion (*couche_diffusion*). De plus, trois registres sont instanciés à partir de sous-modules différents : *state_register_w_en* pour le REG 3, *register_tag* pour le REG 1, et *register_cipher* pour le REG 2. Ces sous-modules ont été développés séparément

et sont ensuite instanciés dans le module principal de permutation (*permutation*).

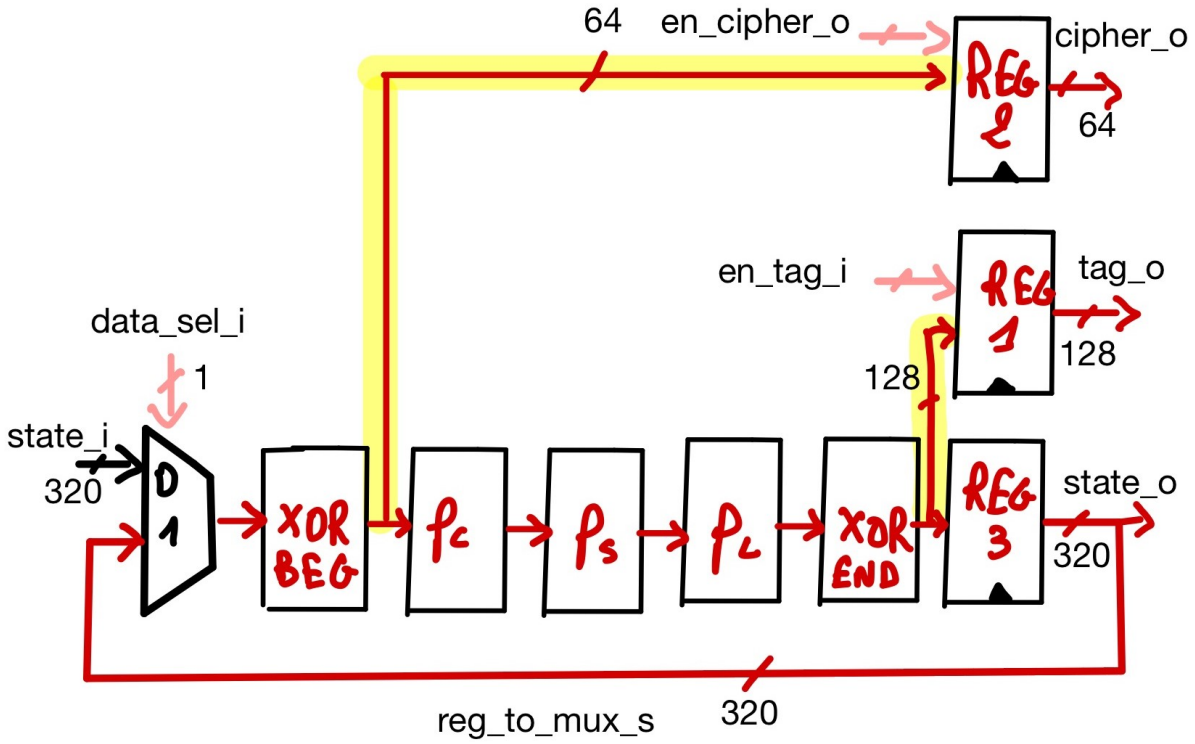

FIGURE 5 – Permutation finale

Remarque

Le signal sortant du *XOR BEG* a une taille de 64 bits, d'après la Figure 3, il s'agit du premier registre x_0 du signal sortant du *XOR BEG*, utilisé dans la phase de Text clair. Le signal sortant du *XOR END* s'étend sur 128 bits, d'après la Figure

3, il s'agit de la concaténation du quatrième et du cinquième registre du signal sortant du *XOR END* utilisés pour récupérer le Tag.

3.3.1 Simulation du testbench permutation_tb

FIGURE 6 – Chronogramme de la simulation du testbench de la permutation finale

Les résultats obtenus à la sortie de la douzième ronde sont en accord avec les valeurs attendues :

```
État ^ (0...0 & K) : 9de1e2d04b50bd86 13141c888a702ce4 55aa22c50d252d83 ef65ff04555a7c13 ac2f83ace5a6d021
```

FIGURE 7 – résultats attendus pour la sortie de la douzième permutation

3.3.2 Le sous-module *addition_constante*

Le module addition_constante permet de réaliser la première étape d'une permutation, expliquée dans le paragraphe 2.2.2.

Entrées	Sorties
state_i	
counter_i	
	state_o

TABLE 3 – Tableau des entrées et sorties du module addition_constante

Simulation du testbench addition_constante_tb

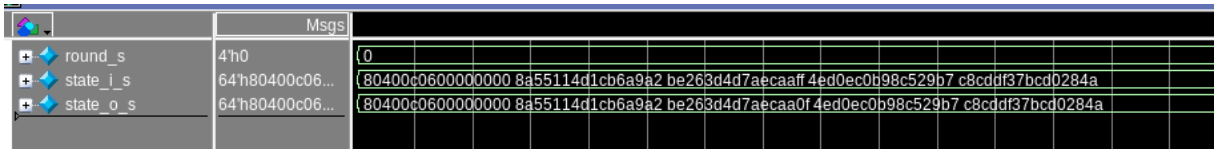

FIGURE 8 – Chronogramme de la simulation du testbench de l'addition de constante

Les résultats obtenus sont en accord avec les valeurs attendues en sortie de l'addition de constante de la première permutation.

3.3.3 Le sous-module *couche_substitution*

Le module couche_substitution permet de réaliser la deuxième étape d'une permutation, expliquée dans le paragraphe 2.2.3. Ce module nécessite de définir un tableau de substitution dans le ascon_pack et de créer un autre module *sbox* qui permet d'associer à chaque colonne de l'entrée state_i de couche_substitution, la valeur lui correspondant dans la sbox.

Entrées	Sorties
$state_i$	
	$state_o$

TABLE 4 – Tableau des entrées et sorties du module couche_substitution

Simulation du testbench couche_substitution_tb

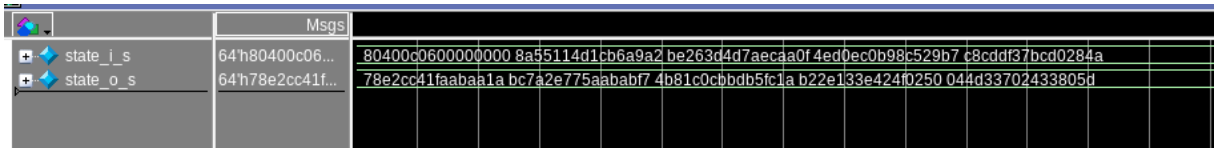

FIGURE 9 – Chronogramme de la simulation du testbench de la couche de substitution

Les résultats obtenus sont en accord avec les valeurs attendues en sortie de la couche de substitution de la première permutation.

3.3.4 Le sous-module *couche_diffusion*

Le module couche_diffusion permet de réaliser la troisième étape d'une permutation, expliquée dans le paragraphe 2.2.4.

Entrées	Sorties
$state_i$	
	state_o

TABLE 5 – Tableau des entrées et sorties du module couche_diffusion

Simulation du testbench couche_diffusion_tb

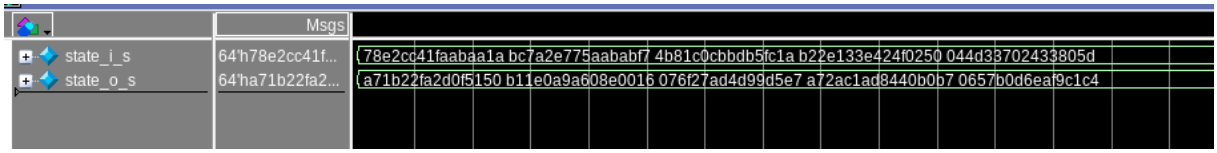

FIGURE 10 – Chronogramme de la simulation du testbench de la couche de diffusion

Les résultats obtenus sont en accord avec les valeurs attendues en sortie de la couche de diffusion de la première permutation.

3.4 La machine à état finis de l'algorithme

La machine à états finis joue un rôle important dans la gestion du processus de chiffrement en contrôlant le flux d'opérations et en garantissant que chaque étape se déroule dans l'ordre correct. Elle y parvient en envoyant des signaux de contrôle au compteur de ronde et au module de permutation à des intervales de temps déterminés par la période d'horloge choisie. Ainsi, elle coordonne les différentes phases du chiffrement : l'initialisation, le traitement des données associées et du texte clair, et la finalisation pour produire un tag d'authentification et un message chiffré.

3.4.1 Tableau des entrée/sorties de la machine à états finis

Entrées	Taille en bits	Sorties	Taille en bits des sorties
clock_i	1	end_o	1
reset_i	1	cipher_valid_o	1
start_i	1	data_sel_o	1
data_valid_i	1	en_xor_lsb	1
round_i	4	en_xor_key_end_o	1
		en_xor_data_o	1
		en_xor_key_o	1
		en_cipher_o	1
		en_tag_o	1
		en_reg_state_o	1
		inita_o	4
		initb_o	4
		en_cpt_r_o	1

TABLE 6 – Tableau des entrée/sorties de la machine à états finis

3.4.2 Diagramme d'état de la machine à états finis du système numérique conçu

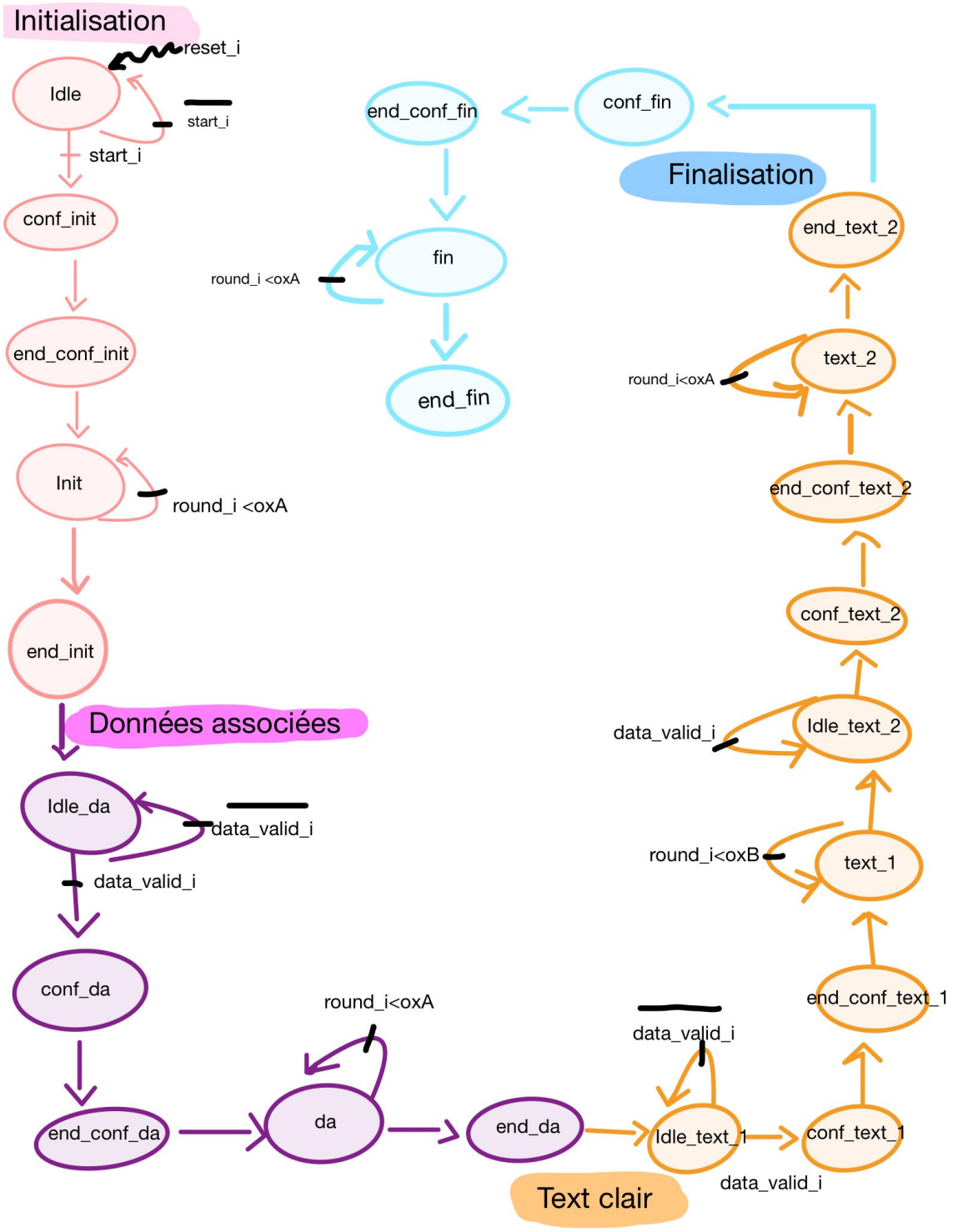

FIGURE 11 – Diagramme d'état de la machine à états finis

4 Le module final

Le module final *ascon_top* englobe la machine d'état, le compteur de rondes et la permutation finale. Il modelise l'entité ASCON128 dans sa globalité. Il a comme entrées/sorties :

- *data_i* : entrée de 64 bits
- *data_valid_i* : entrée indiquant la présence d'une donnée valide sur data_i
- *key_i* : clée de 128 bits
- *start_i* :permet de commencer le chiffrement
- *cipher_o* : sortie sur 64 bits
- *cipher_valid_o* : sortie indiquant la validité de la sortie cipher_o
- *tag_o* : sortie sur 128 bits ***end_o*** : sortie indiquant la fin du chiffrement

4.1 Simulation du testbench ascon_top_tb

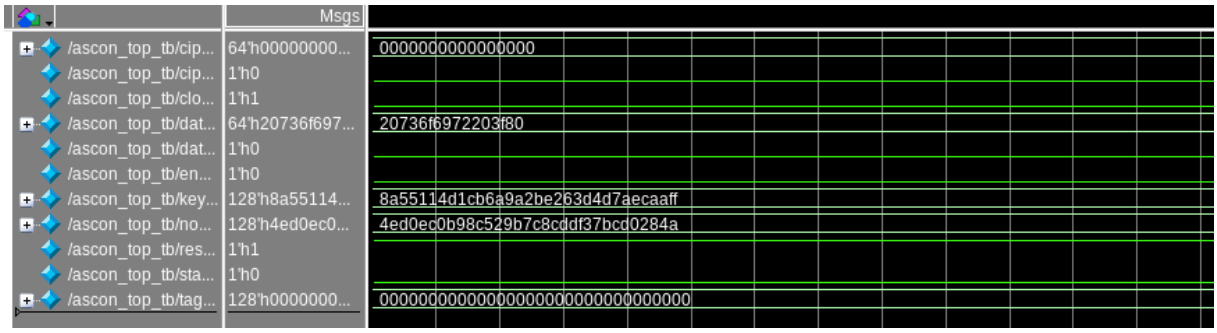

FIGURE 12 – Chronogramme de la simulation du testbench du module final

Les résultats obtenus ne concordent pas avec les valeurs attendues. En effet, nous obtenons une sortie nulle, dû à une erreur dans la programmation de la machine à états finis. Afin de résoudre ce problème, il est nécessaire de procéder par étapes, et de lancer la simulation pour chaque bloc de chiffrement, pour trouver là où l'algorithme bloque.
Commencer par vérifier la fonctionnalité de chaque bloc de chiffrement séparément peut aider à clarifier si les erreurs proviennent d'une mauvaise configuration des états, d'une logique de transition incorrecte, ou d'autres bugs dans le code Verilog. Après avoir isolé le problème, des ajustements ciblés peuvent être appliqués pour rectifier les dysfonctionnements. Cela inclut la révision des conditions de transition entre les états, l'assurance que toutes les entrées et sorties sont correctement gérées, et la confirmation que les constantes et les valeurs de contrôle sont correctement définies.
Enfin, une fois ces modifications réalisées, il est important de réexécuter les simulations pour chaque bloc, en observant les réponses du système et en ajustant progressivement la logique de la machine à états finis jusqu'à ce que les résultats escomptés soient obtenus. Cette démarche rigoureuse est indispensable pour garantir la fiabilité et la sécurité de l'algorithme de chiffrement ASCON128, assurant ainsi la confidentialité et l'authenticité des données traitées.

5 Conclusion

En conclusion, bien que nous ayons rencontré des défis qui nous ont empêchés d'arriver au bout de la modélisation et de l'implémentation de l'algorithme de chiffrement authentifié avec données associées (AEAD). Ce projet a été l'occasion d'explorer en détail des aspects tels que la gestion des états, les permutations, et la sécurisation des communications via des tags d'authentification.
Tout en travaillant dans les limites des contraintes techniques et temporelles, l'utilisation de SystemVerilog nous a aidés à simuler le comportement de l'algorithme et à tester sa validité à chaque étape du processus de chiffrement.
Bien que le projet n'ait pas atteint tous les objectifs initiaux, les leçons tirées de cette expérience sont inestimables et préparent le terrain pour de futurs projets similaires. Ces apprentissages renforcent notre compréhension des défis liés à la sécurisation des communications numériques et démontrent l'importance de l'algorithme ASCON128 dans le maintien de la confidentialité et de l'authenticité des données dans un contexte de sécurité informatique contemporain.

Table des figures