**INF1500 – logique des systèmes numériques**

**Laboratoire 3:**

**Simulation et implémentation d’un circuit en logique combinatoire**

**Groupe : 02**

**Présenté par :**

**Nathan Ramsay-Vejlens (1989944)**

**Louis Dutheil (1994257)**

**Octobre 2019**

**Département de génie informatique et de génie logiciel**

**École Polytechnique de Montréal**

**Introduction**

Dans le cadre de ce TP on nous demande de créer un circuit qui alterne entre un codage en code gray et un code secret, on doit ensuite implémenter ce code à la carte FPGA.

**Table de vérité du Code Gray**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Code décimal | Code hexadécimal |  | Entrée (code binaire) | | | |  | Sortie (code Gray) | | | |
|  |  | E3 | E2 | E1 | E0 |  | S3 | S2 | S1 | S0 |
| 0 | 0 | 0 | 0 | 0 | 0 |  | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |  | 0 | 0 | 0 | 1 |
| 2 | 2 | 0 | 0 | 1 | 0 |  | 0 | 0 | 1 | 1 |
| 3 | 3 | 0 | 0 | 1 | 1 |  | 0 | 0 | 1 | 0 |
| 4 | 4 | 0 | 1 | 0 | 0 |  | 0 | 1 | 1 | 0 |
| 5 | 5 | 0 | 1 | 0 | 1 |  | 0 | 1 | 1 | 1 |
| 6 | 6 | 0 | 1 | 1 | 0 |  | 0 | 1 | 0 | 1 |
| 7 | 7 | 0 | 1 | 1 | 1 |  | 0 | 1 | 0 | 0 |
| 8 | 8 | 1 | 0 | 0 | 0 |  | 1 | 1 | 0 | 0 |
| 9 | 9 | 1 | 0 | 0 | 1 |  | 1 | 1 | 0 | 1 |
| 10 | A | 1 | 0 | 1 | 0 |  | 1 | 1 | 1 | 1 |
| 11 | B | 1 | 0 | 1 | 1 |  | 1 | 1 | 1 | 0 |
| 12 | C | 1 | 1 | 0 | 0 |  | 1 | 0 | 1 | 0 |
| 13 | D | 1 | 1 | 0 | 1 |  | 1 | 0 | 1 | 1 |
| 14 | E | 1 | 1 | 1 | 0 |  | 1 | 0 | 0 | 1 |
| 15 | F | 1 | 1 | 1 | 1 |  | 1 | 0 | 0 | 0 |

**Table de Karnaugh du Code Gray**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S0** | | E1 E0 | | | |
| **00** | **01** | **11** | **10** |
| E3 E2 | **00** | 0 | 1 | 0 | 1 |
| **01** | 0 | 1 | 0 | 1 |
| **11** | 0 | 1 | 0 | 1 |
| **10** | 0 | 1 | 0 | 1 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S1** | | E1 E0 | | | |
| **00** | **01** | **11** | **10** |
| E3 E2 | **00** | 0 | 0 | 1 | 1 |
| **01** | 1 | 1 | 0 | 0 |
| **11** | 1 | 1 | 0 | 0 |
| **10** | 0 | 0 | 1 | 1 |

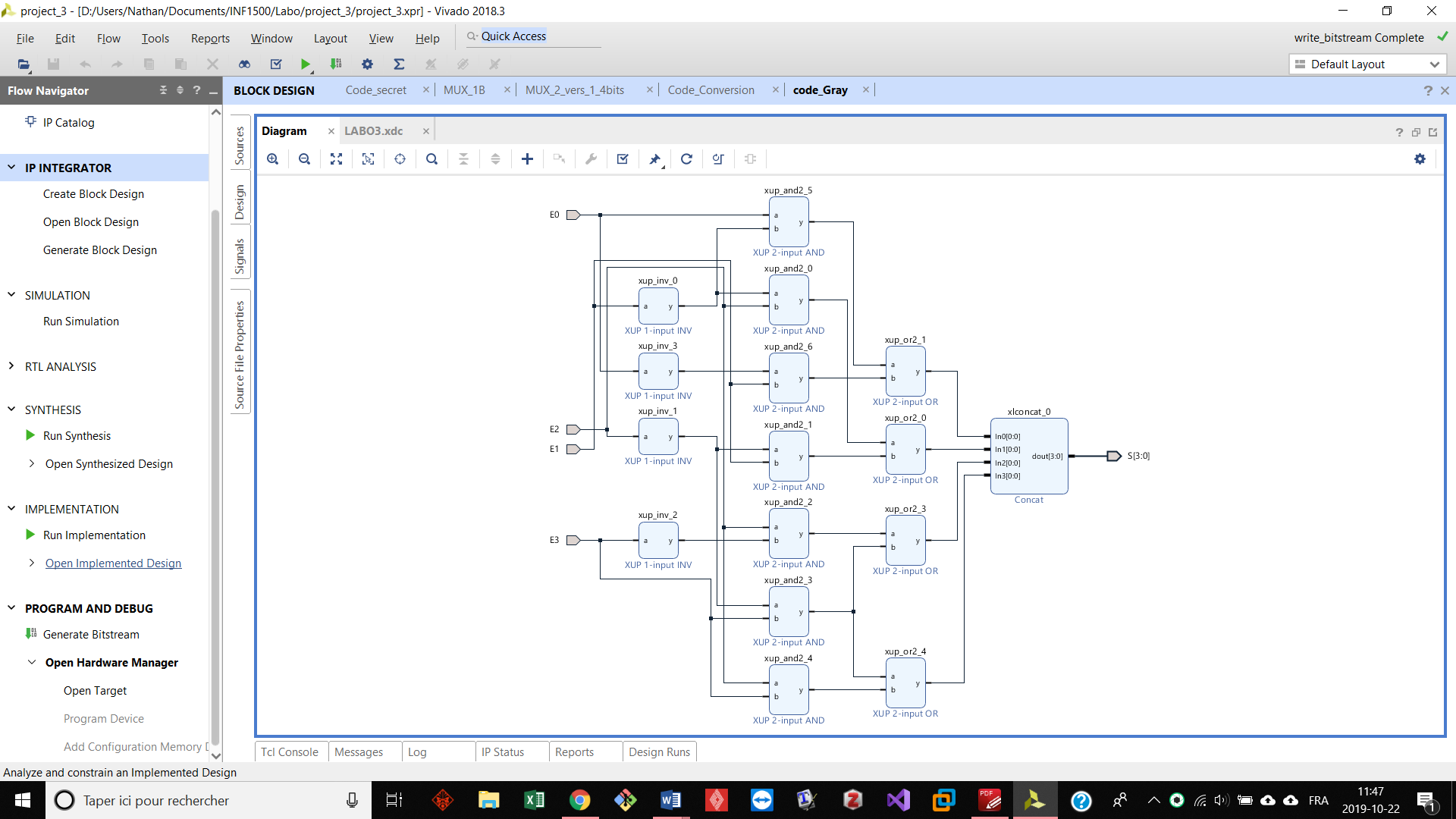
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S2** | | E1 E0 | | | |
| **00** | **01** | **11** | **10** |
| E3 E2 | **00** | 0 | 0 | 0 | 0 |
| **01** | 1 | 1 | 1 | 1 |
| **11** | 0 | 0 | 0 | 0 |
| **10** | 1 | 1 | 1 | 1 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S3** | | E1 E0 | | | |
| **00** | **01** | **11** | **10** |
| E3 E2 | **00** | 0 | 0 | 0 | 0 |
| **01** | 0 | 0 | 0 | 0 |
| **11** | 1 | 1 | 1 | 1 |
| **10** | 1 | 1 | 1 | 1 |

F0=E0+E1 F1=E2\*E1’+E1\*E2’

F2=E3’\*E2+E3\*E2’ F3=E3\*E2+E3\*E2’

**Design du convertisseur en code Gray**

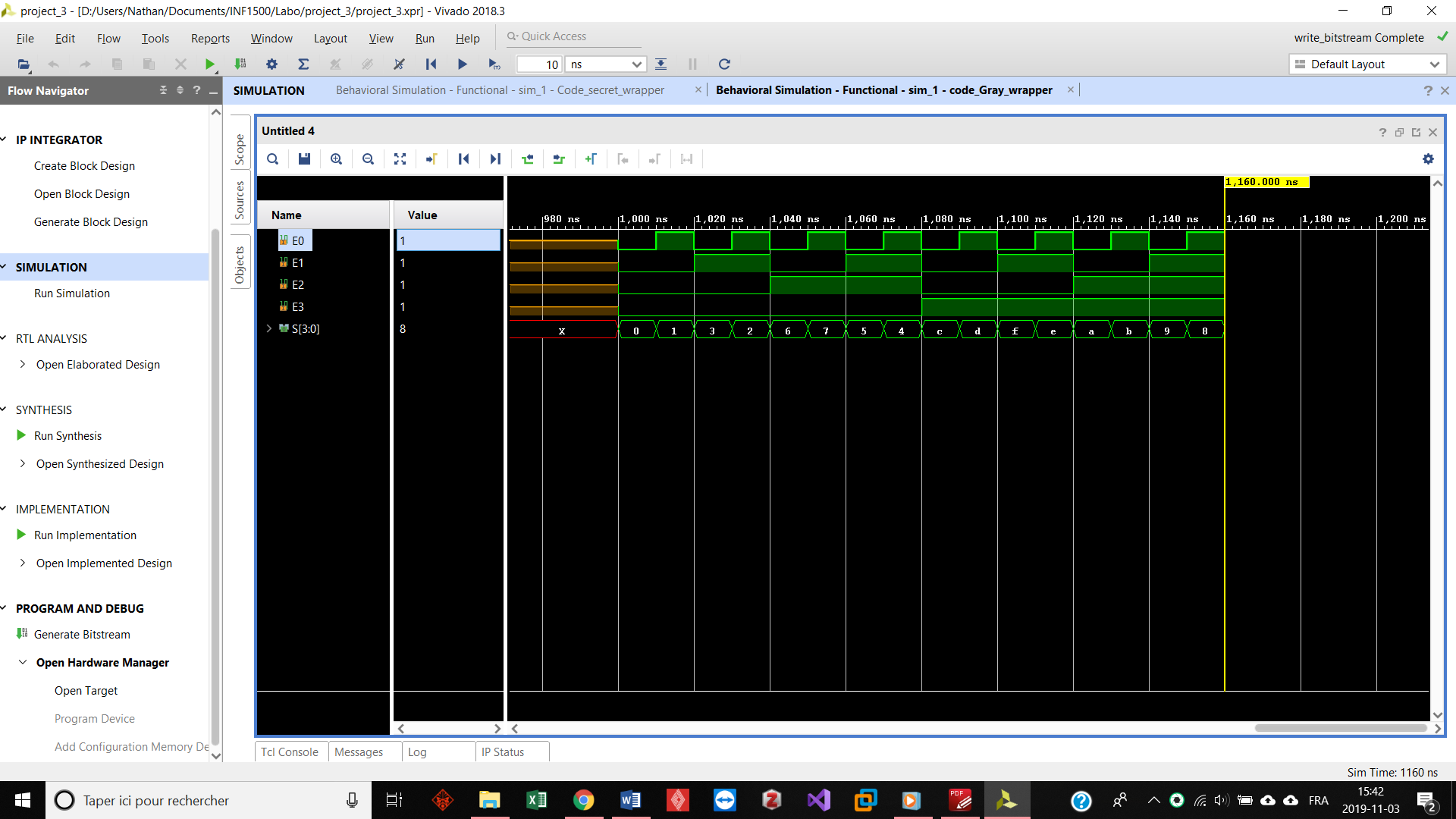


**Description Générale :**

Pour la conversion de binaire au code gray, on va créer le circuit à partir des portes logiques de base OU, ET et INV, on va juste utiliser un concat pour regrouper tous les signaux en un seul signal composé de 4 bits. L’idée est d’inverser le bit le plus à droite (le moins significatif) permettant d’obtenir un nombre nouveau. Nous avons fait en sorte que les équations décrites plus haut soient respectées.

**Simulation :**

Ce n’était pas long de tester toutes les valeurs alors nous avons procédé de cette manière.



**Table du code secret :**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| A2 | A1 | A0 |  | S3 | S2 | S1 | S0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | X |
| 0 | 1 | 1 | 0 | 1 | X | X |
| 1 | 0 | 0 | 1 | X | X | X |
| X | X | X | 0 | 0 | 0 | 0 |

Pour S0, nous avons 1 terme obligatoire (A0A1’A2’) et 3 termes optionnels qui sont :

S0 = A0\*A1\*A2’ S0 = A0’\*A1\*A2’ et S0 = A0’\*A1’\*A2.

Voici la simplification :

S0 = A0\*A1’\*A2’+A0\*A1\*A2’(optionnel)+A0’\*A1\*A2’(optionnel)+A0’\*A1’\*A2(optionnel)

Par la loi d’absorption logique :

S0 = A0\*(A1\*A2’+A1’\*A2’) + A0’\*(A1\*A2’+A1’\*A2)

Supprimons le deuxième terme puisqu’il est optionnel.

Par la loi d’adjacence logique :

S0 = A0\*A2’

Pour S1, nous avons 1 terme obligatoire (A0’\*A1\*A2’) et 2 termes optionnels qui sont :

S1 = A0’\*A1\*A2’ et S1 = A0’\*A1’\*A2.

Simplification :

S1 = A0’\*A1\*A2’ + A0\*A1\*A2’(optionnel) + A0’\*A1’\*A2(optionnel)

S1 = A1\*(A0’\*A2’+A0\*A2’) + A0’\*A1’\*A2’(optionnel)

S1 = A1\*A2’+A0’\*A1’\*A2’(optionnel)

Supprimons le deuxième terme puisqu’il est optionnel

S1=A1\*A2’

Pour S2, nous avons 1 terme obligatoire (A0\*A1\*A2’) et 1 terme optionnel qui est A0’\*A1’\*A2.

Simplification :

S2=A0\*A1\*A2’+A0’\*A1’\*A2(optionnel)

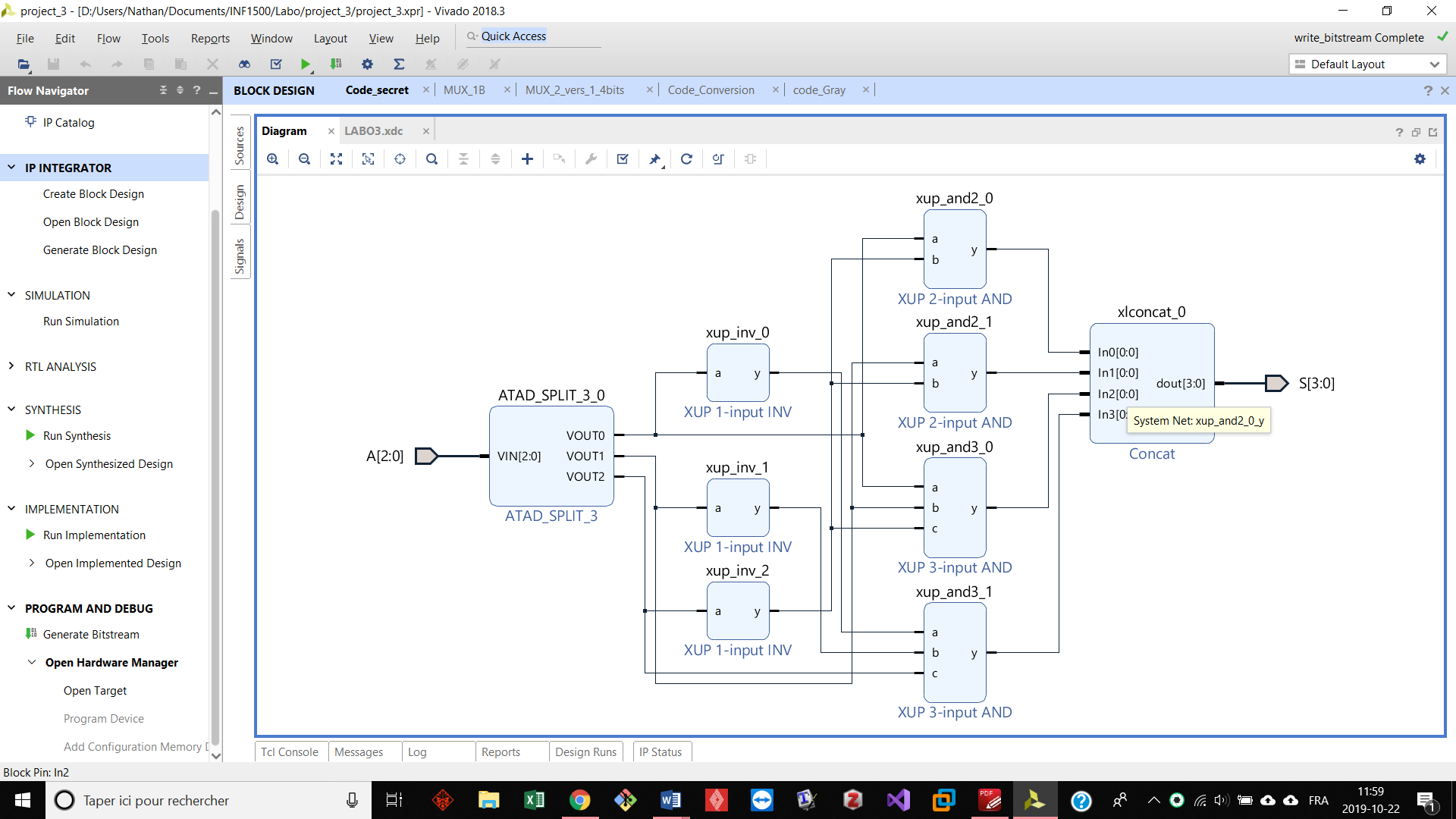
Supprimons le deuxième terme:

S2=A0\*A1\*A2’

Il n’y a pas de simplification possible pour S3.

S3=A0’\*A1’\*A2

**Design du code secret:**

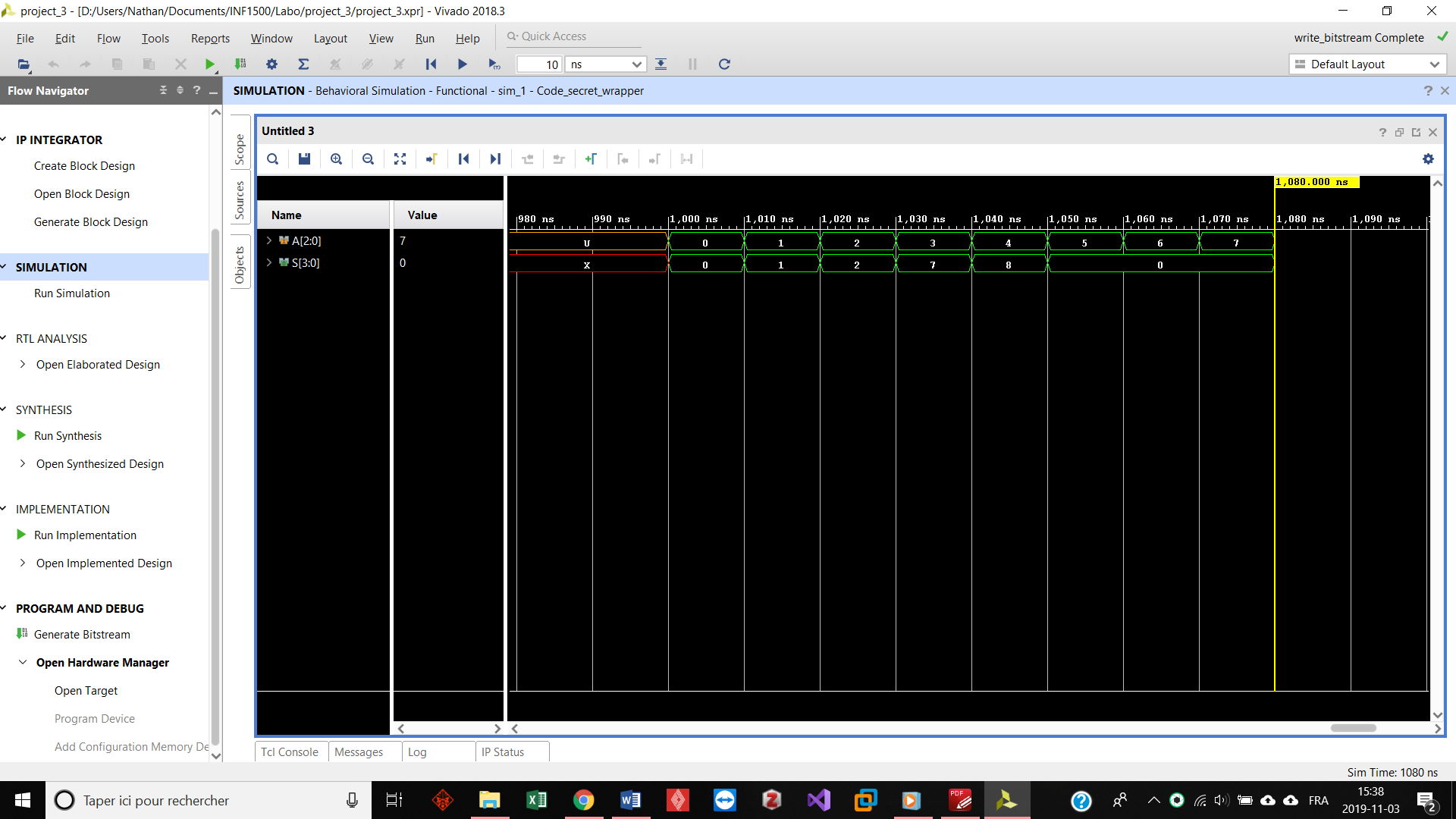


**Description Générale :**

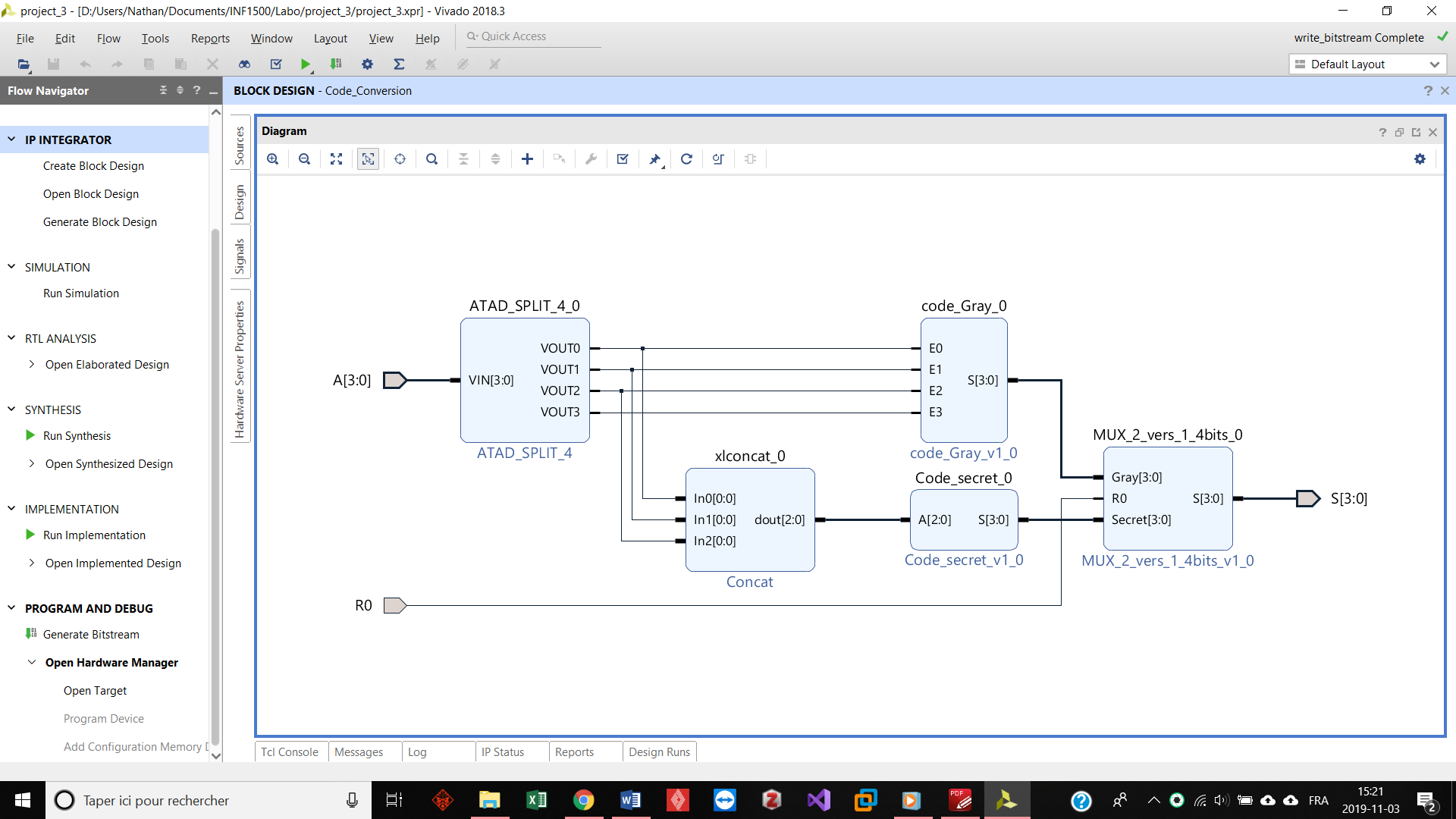
Pour le code secret, nous avons commencé par SPLIT le vecteur et ensuite, nous avons réalisé les opérations nécessaires afin de correspondre aux équations trouvées de S0, S1, S2 et S3. C’est-à-dire que S0 est fait d’une porte ET prenant en entrée la valeur A[0] et la valeur inverse de A[2]. S1 est fait d’une porte ET prenant en entrée la valeur A[1] et la valeur inverse de A[2]. S2 est fait d’une porte ET prenant en entrée la valeur A[0], la valeur A[2] et la valeur inverse de A[2]. S3 est fait d’une porte ET prenant la valeur A[0] inverse, A[1] inverse et A[2]. Finalement, nous avons concaténé ces résultats afin de former un vecteur de 4 bits.

**Simulation :**

Ce n’était pas long de tester toutes les valeurs alors nous avons procédé de cette manière.



**Design final :**

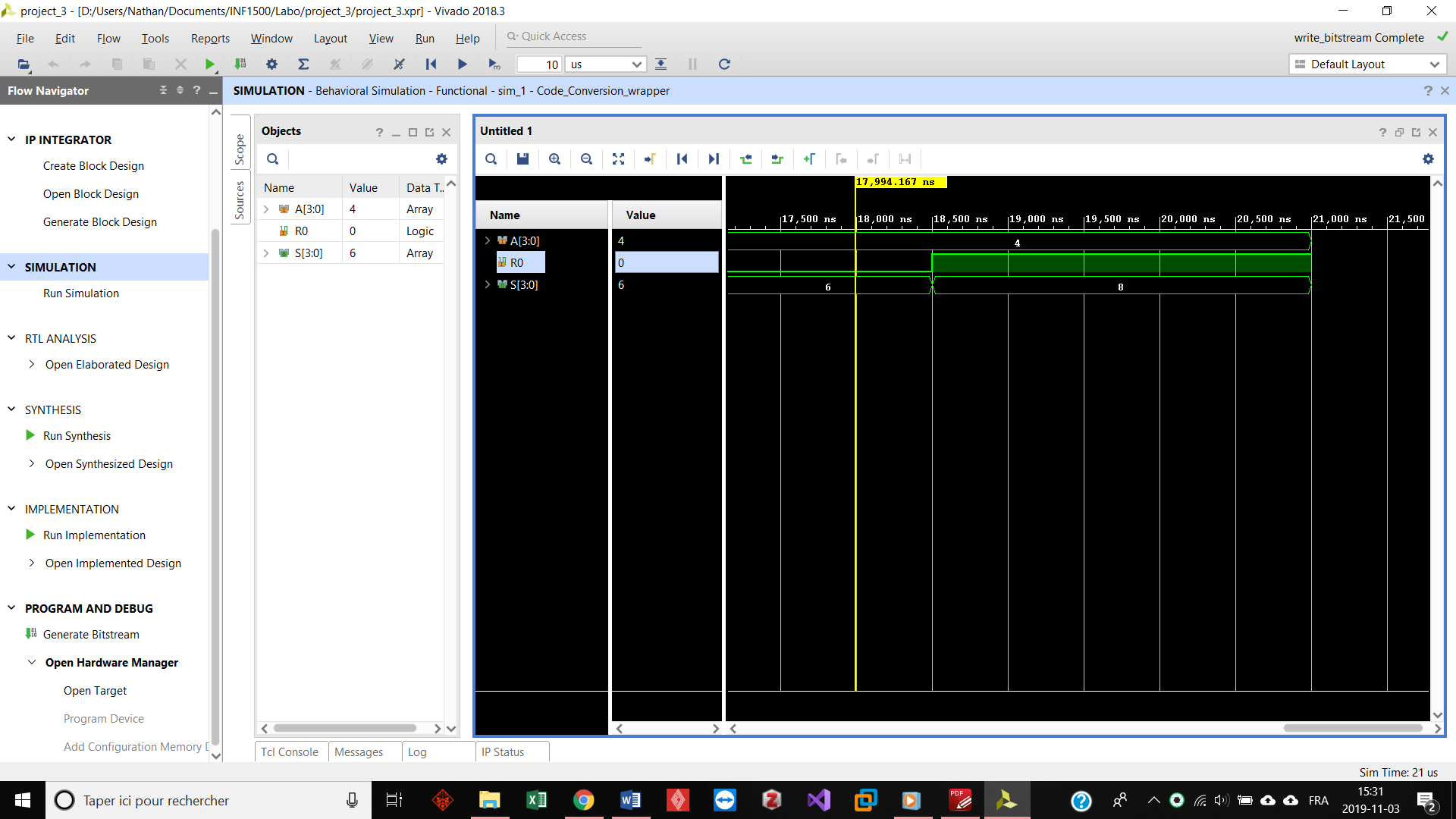


**Description générale :**

Pour le design final, nous avons simplement ajouter les deux modules précédant avec un split pour permettre de garder seulement 3 bits pour le code secret et 4 bits pour le code gray. Ensuite, nous avons ajouter notre MUX afin de choisir quelle conversion nous voulions.

**Simulation :**

Pour la simulation, nous avions déjà tester chaque module séparément alors nous avons simplement procéder aux mêmes tests en ne faisant que varier le sélecteur du MUX. Pour R0=0, nous avons le code Gray et R0=1 donne le code secret.



**Conclusion :**

En conclusion, nous avons créé un système permettant de transformer une entrée en code gray ou en code secret qui permet de faciliter grandement ce genre de calcul en l’automatisant.