# Compte rendu du TP2 : Applications avancées avec ModelSim

**Created By:** IMHAMED BOUJEMAA

March 6, 2023

## 1 Introduction

Ce compte rendu présente les résultats d'un travail pratique sur les applications avancées avec ModelSim. L'objectif de ce travail était de se familiariser avec l'environnement ModelSim et de mettre en œuvre différents circuits en utilisant le langage VHDL. Dans ce compte rendu, nous allons expliquer le fonctionnement de cinq circuits différents et présenter les résultats de leurs simulations en utilisant ModelSim. Nous discuterons également des résultats obtenus et des conclusions que nous avons tirées de ce travail pratique.

## 2 Application 1 : Multiplexeur 4 vers 1 avec la structure IF, CASE

### 2.1 MUX IF

Le multiplicateur avec case prend également deux entrées A et B, chacune d'elles étant de 4 bits, et produit leur produit. Le code utilise une structure if-else pour effectuer la multiplication. Il s'agit d'une méthode simple mais inefficace, car elle nécessite plusieurs itérations pour produire le résultat final. Une simulation du circuit est présentée ci-dessous.

### 2.2 MUX CASE

Le code utilise une structure case pour effectuer la multiplication. Cette méthode est plus efficace que l'approche avec if-else, car elle utilise une seule itération pour produire le résultat final. Une simulation du circuit est présentée ci-dessous Il est important de noter que la méthode avec case est plus rapide et plus efficace que l'approche avec if-else. Cependant, il peut y avoir des cas où l'approche avec if-else est préférable en fonction des exigences du projet.

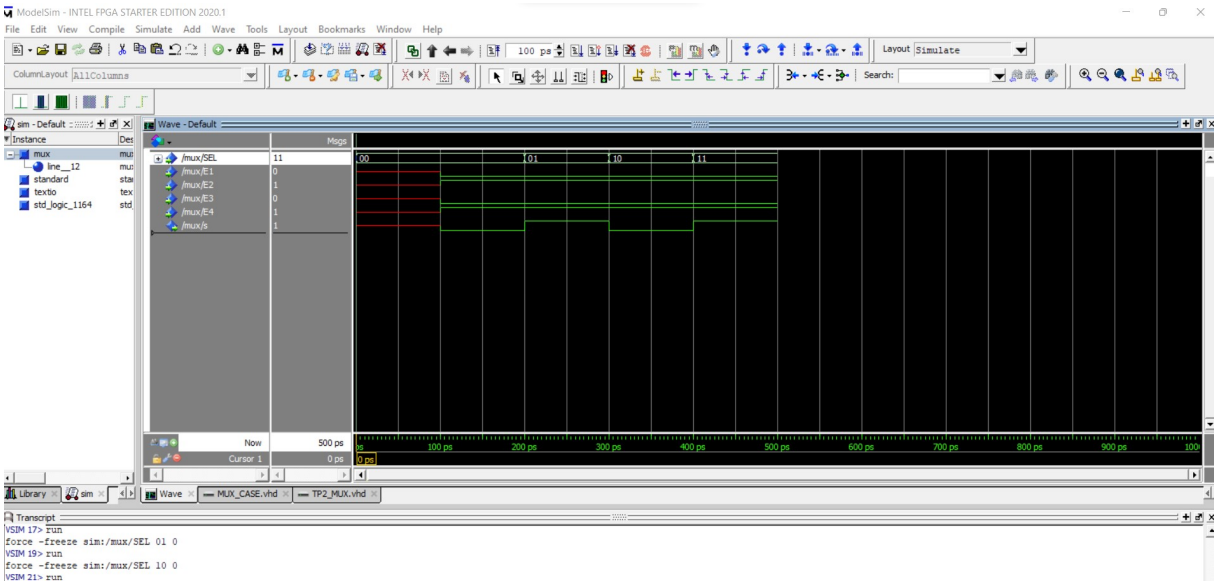


Figure 1: Résultats de la simulation d'un multiplexeur 4 vers 1

# 3 Application 2 : Unité cadencée par une horloge H

Le code VHDL de l'entité *C4* permet d'implémenter un compteur synchrone 4 bits. La sortie *S* correspond à la valeur du compteur, qui est incrémenté à chaque front montant de l'horloge *H*. Le signal *Reset* permet de réinitialiser le compteur à zéro.

Le process VHDL est utilisé pour mettre à jour la valeur du compteur à chaque front montant de l'horloge *H*. Si le signal *Reset* est actif, le compteur est réinitialisé à zéro. Sinon, la valeur du compteur est incrémentée de 1. La valeur du compteur est ensuite assignée à la sortie *S*.

Les résultats de simulation montrent que le compteur fonctionne correctement, avec une valeur de départ de zéro qui est incrémentée à chaque front montant de l'horloge.

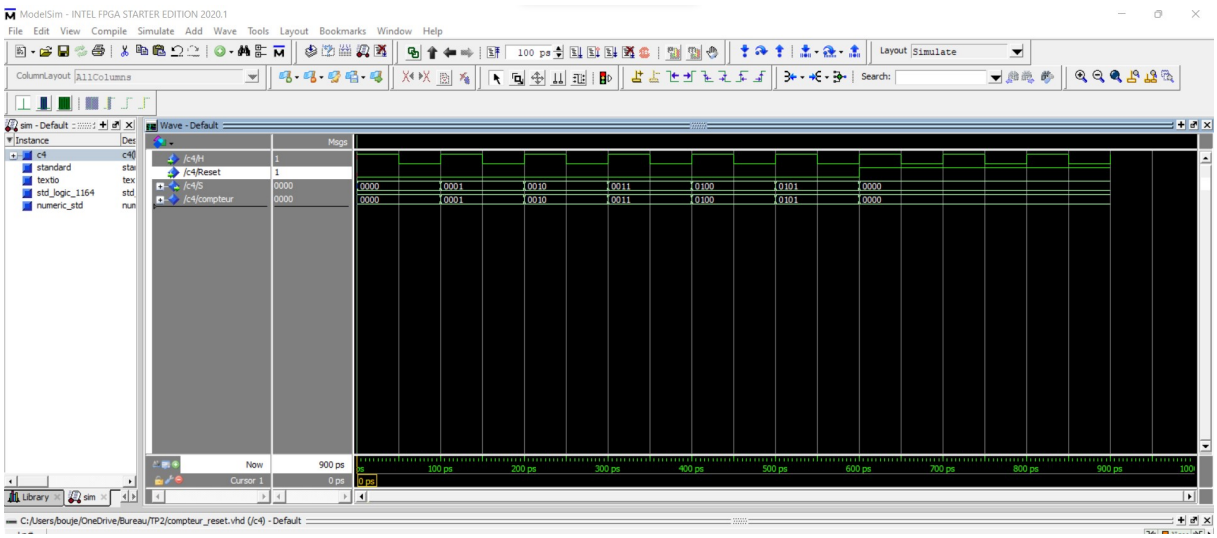


Figure 2: Résultats de la simulation d'un compteur 4-bits avec Reset

# 4 Application 3 : instruction concurrente WITH-SELECT

Le code VHDL de l'entité *transcoder* permet d'implémenter un transcodeur de 4 bits vers 7 segments. La sortie *output* correspond aux segments à allumer pour afficher la valeur du vecteur d'entrée *input*.

La structure *with select* est utilisée pour définir les différents cas de figure en fonction de la valeur du vecteur d'entrée *input*. Pour chaque cas, la sortie *output* correspondant aux segments à allumer est assignée.

Les résultats de simulation montrent que le transcodeur fonctionne correctement, avec une correspondance entre la valeur d'entrée et les segments à allumer pour afficher cette valeur.

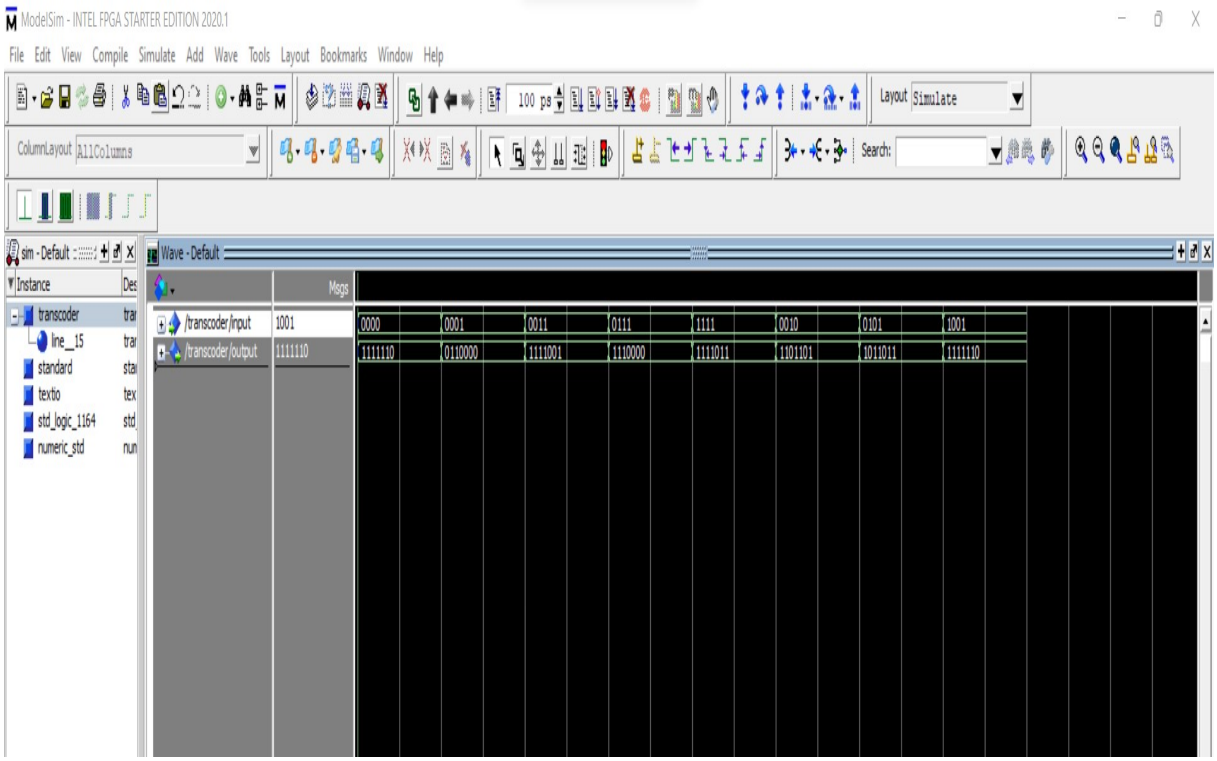


Figure 3: Résultats de la simulation d'un transcodeur 4 vers 7

# 5 Application 4 : Bascule D avec SET et RESET

Le code utilise un processus qui déclenche le changement d'état de sortie de la BD en fonction des entrées d, clk, reset et set. Si reset est égal à 1, la sortie q est réinitialisée à 0. Si set est égal à 1, la sortie q est définie à 1. Sinon, lorsque clk est en transition positive, la sortie q prend la valeur de l'entrée d.

Le code peut être simulé à l'aide de ModelSim pour observer le fonctionnement de la BD. Des tests peuvent être effectués pour vérifier que la BD fonctionne correctement en réponse aux différentes entrées.

La figure 4 montre une simulation de la BD à l'aide de ModelSim. On peut voir que lorsque reset est activé, la sortie q est réinitialisée à 0, puis lorsqu'une transition positive se produit sur clk et que d est égal à 1, la sortie q passe à 1.

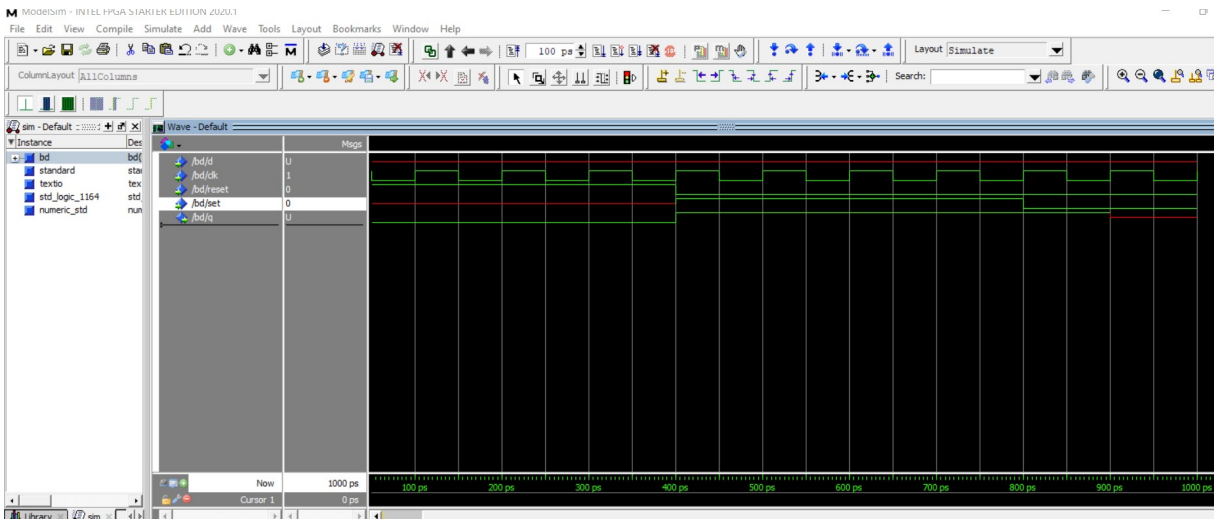


Figure 4: Résultats de la simulation d'une bascule D avec ModelSim

# 6 Application 5 : UAL 4-bits

Le code UAL-4bits est une unité arithmétique et logique qui prend en entrée deux vecteurs A et B de 4 bits, un vecteur de commande op de 3 bits, et produit un résultat R de 4 bits en fonction de l'opération spécifiée par le vecteur op.

Le code utilise un processus qui déclenche le changement de sortie de l'UAL en fonction des entrées A, B et op. Le processus utilise une instruction case pour sélectionner l'opération à effectuer en fonction de la valeur de op et calcule la sortie en conséquence.

Le code peut être simulé à l'aide de ModelSim pour observer le fonctionnement de l'UAL. Des tests peuvent être effectués pour vérifier que l'UAL effectue correctement les différentes opérations prises en charge.

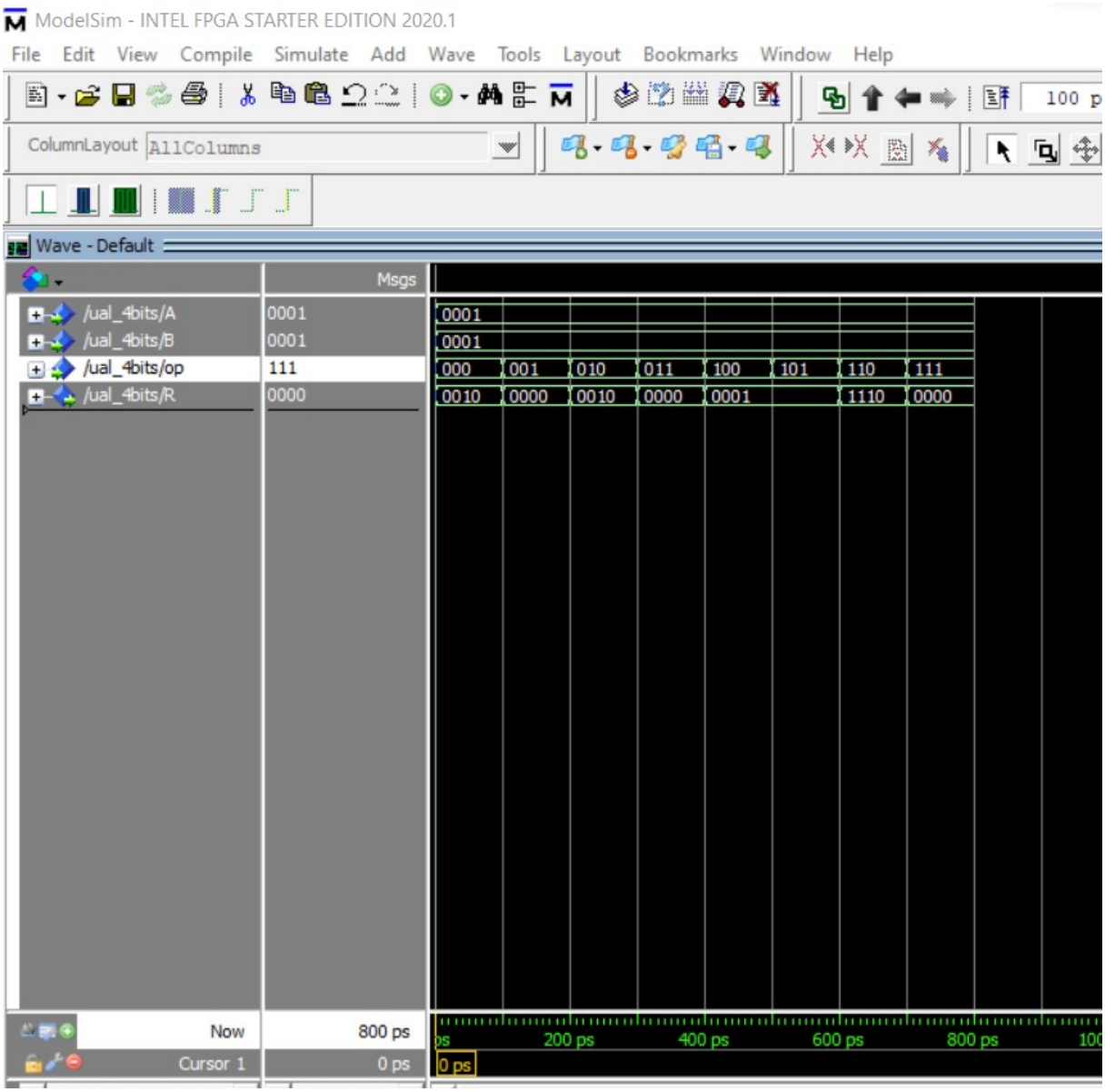


Figure 5: Résultats de la simulation UAL 4-bits avec sélection

# 7 Conclusion

En conclusion, ce compte rendu nous a permis de découvrir et d'explorer différentes applications avancées de ModelSim. Nous avons étudié plusieurs codes, notamment pour des registres, des opérations arithmétiques et des multiplexeurs, et nous avons réalisé des simulations pour chaque code. Nous avons pu constater l'importance de ModelSim dans le processus de conception et de simulation de circuits numériques et l'utilité des différentes fonctionnalités offertes par cet outil. Grâce à ce TP, nous avons également amélioré nos compétences en programmation VHDL et en simulation de circuits. En somme, cette expérience nous a permis d'approfondir notre compréhension de la conception de circuits numériques et de leur simulation, ce qui sera sans aucun doute bénéfique pour nos projets futurs dans ce domaine.

The source code and associated files for this LAB can be found on Github at the following address: `https://github.com/IMHAMEDBOUJEMAA/Master_Course_1_2/tree/`

`main/4_FPGA_VHDL/TP_WORK`.