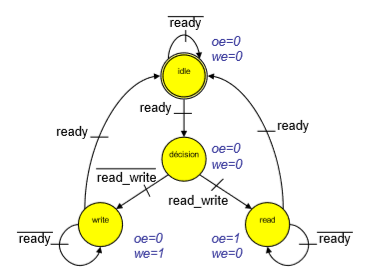
COMPTE RENDU TP 2 VHDL

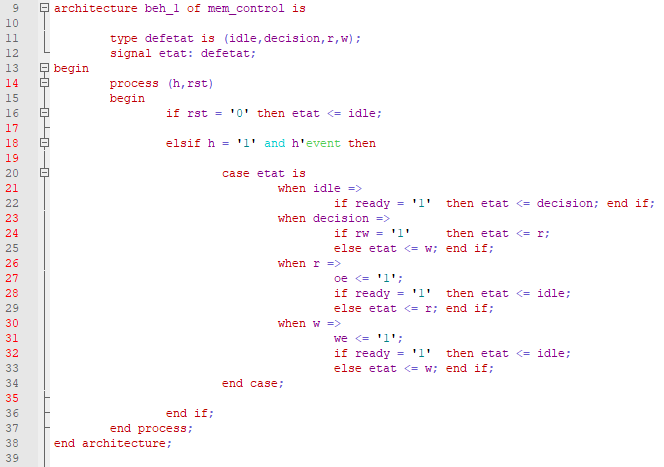
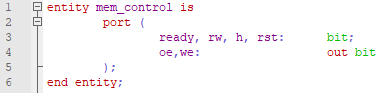
L’objectif de ce travail est d’illustrer la description comportementale en langage VHDL de machines à états finis autonomes de type *Moore* ou *Mealy*.

# CONTRÔLEUR DE MEMOIRE

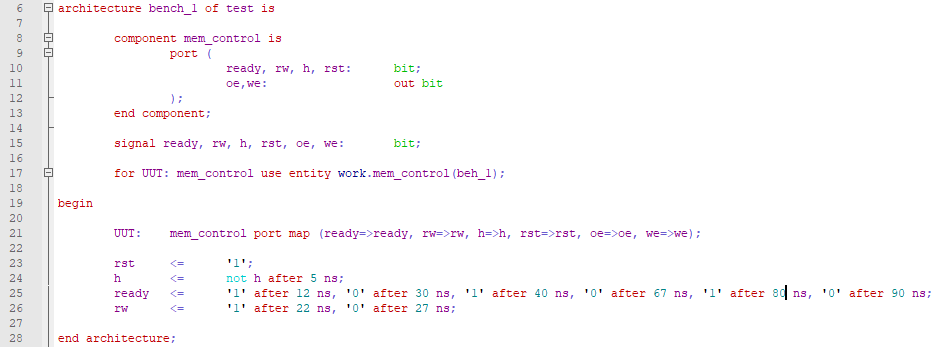
## PREPARATION

Nous réalisons en VHDL une entité et une architecture correspondant au Diagramme suivant, décrivant le fonctionnement d’un contrôleur de mémoire.

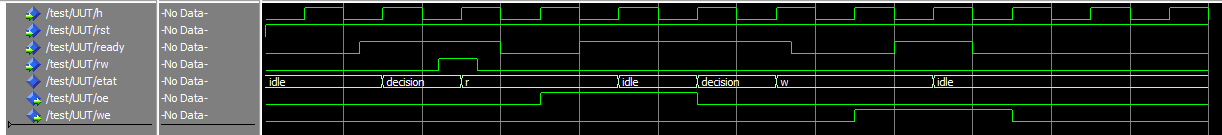




Nous définissons ensuite le *testbench* qui va nous permettre de tester cette architecture.



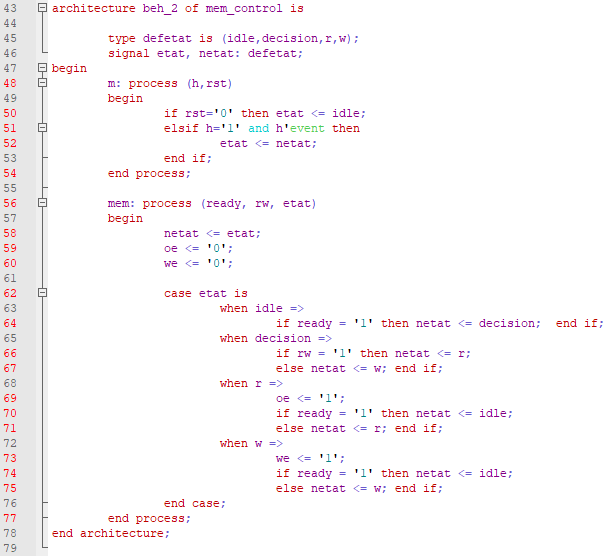
En lançant la simulation, nous obtenons le chronogramme suivant.



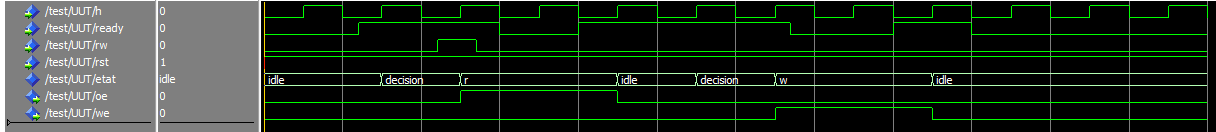
Nous observons que le chronogramme correspond bien au Diagramme des états. Seulement, on voit une latence d’un coup d’horloge entre les entrés et les sorties. Cela vient du fait qu’il faille d’abord calculer l’état interne pour ensuite, au prochain tour du process, modifier les sorties.

Ce problème devrait être réglé en créant deux process différents, un pour l’horloge et le *reset*, et un autre pour les autres signaux.

Voici la nouvelle architecture implémentant les deux process.



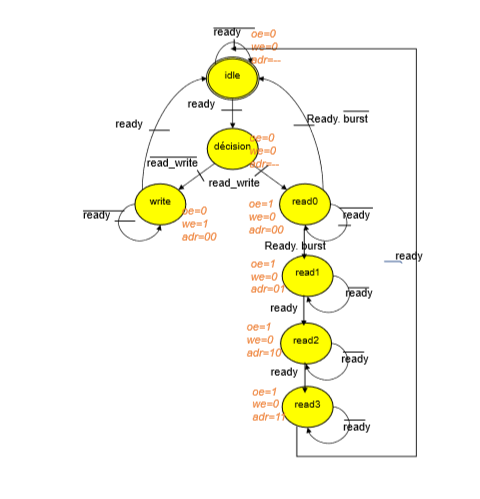
Pour la simulation, nous gardons le même *testbench* que précédemment, en utilisant l’architecture *beh\_2* à la place de *beh\_1*. Voici le chronogramme correspondant.



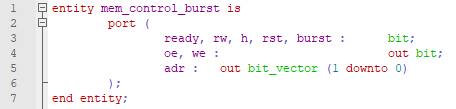
On observe bien ce qui était prévu. En effet, sur ce chronogramme on observe que les sorties sont bien synchronisées avec l’état interne.

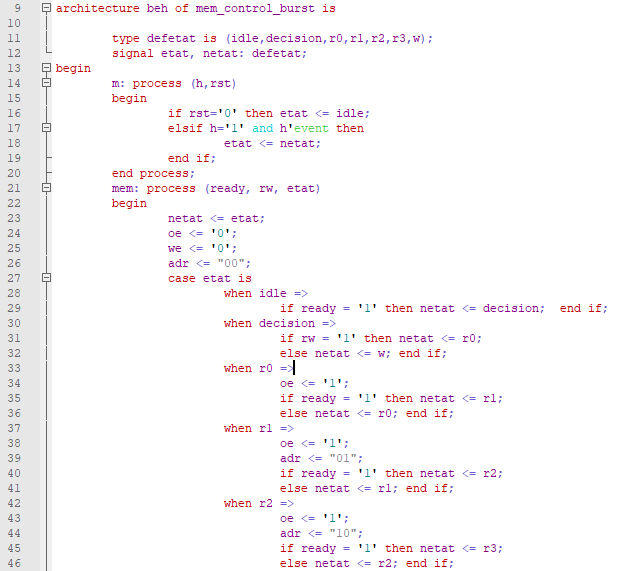
Nous concluons que cette méthode d’implémentation est plus rapide.

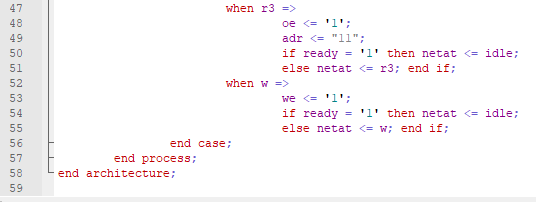
### Implémentation du *BURST*

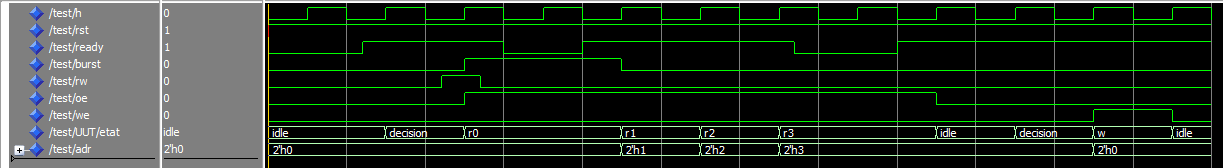
Nous ajoutons la fonctionnalité *burst* dont les caractéristiques sont décrites dans le diagramme suivant.

Voici le code VHDL correspondant.



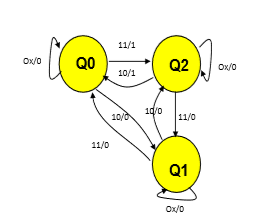




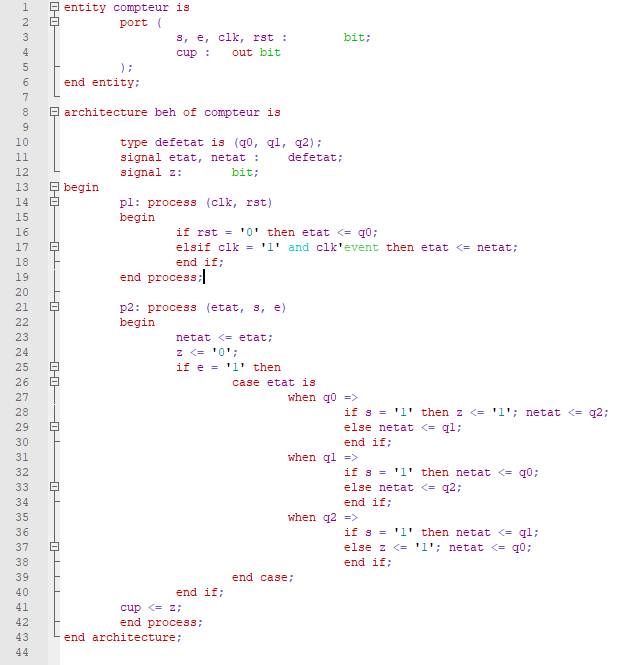


Le chronogramme correspond bien au diagramme d’états. La fonction *burst* est opérationnelle.

# COMPTEUR

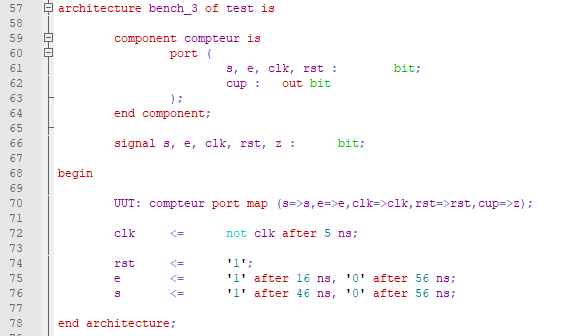
Nous voulons réaliser un compteur d’après la description suivante.

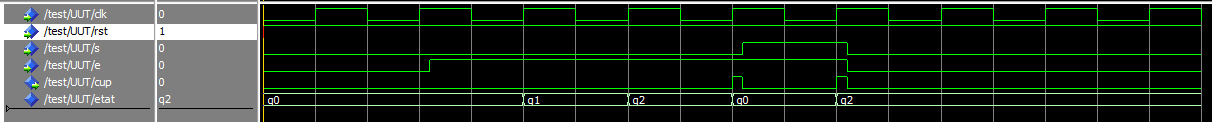
Nous commençons par décrire l’entité et l’architecture correspondante.



Nous souhaitons comparer deux scénarios différents : le premier avec les entrées synchrones et le second avec les entrées asynchrones.

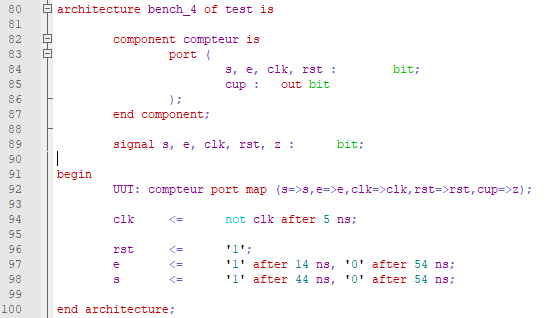
### PREMIER SCENARIO

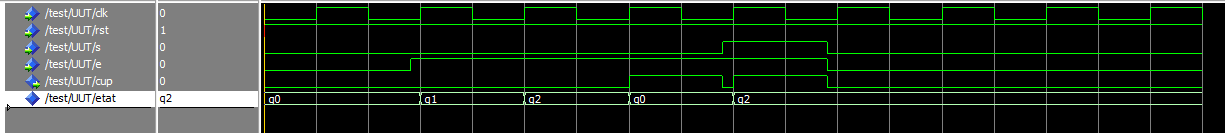




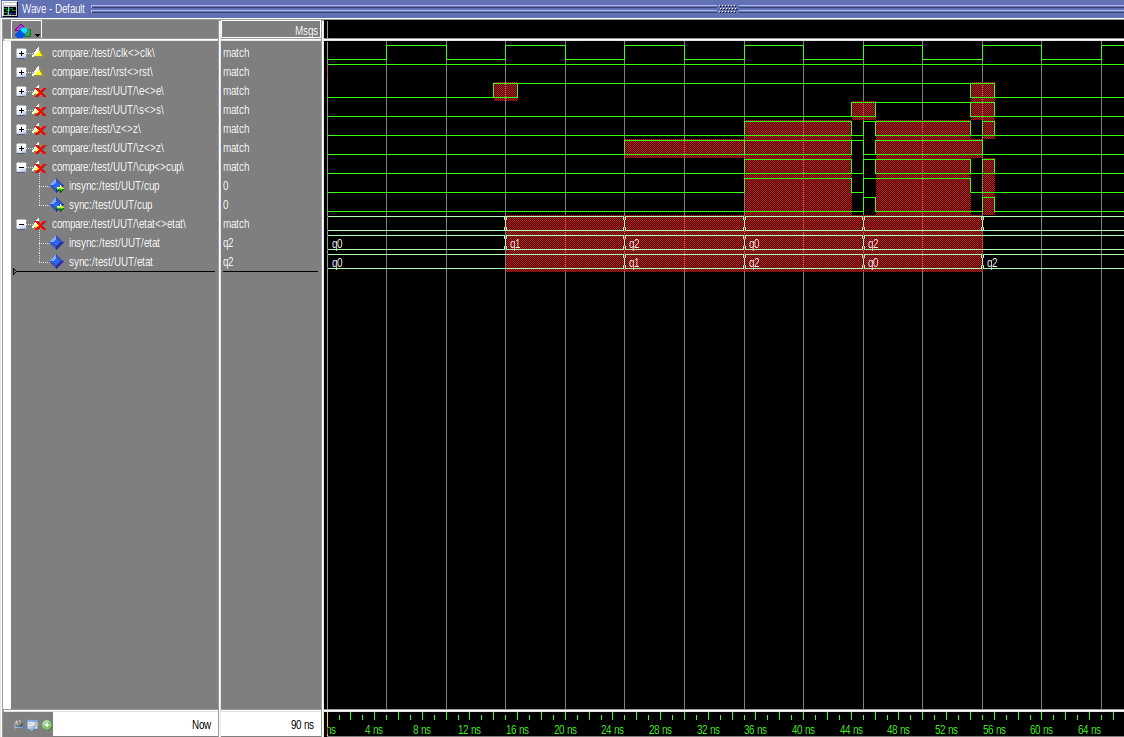
Le résultat est conforme à la description.

### SECOND SCENARIO





Le résultat est aussi conforme au fonctionnement voulu. Nous allons comparer les deux scénarios avec l’outil de comparaison de *ModelSim*.



On remarque que le scénario synchrone est en retard par rapport au scénario asynchrone. En effet, dans le cas asynchrone, les états internes ont le temps de s’actualiser avant le front d’horloge, alors que dans l’autre cas, il faut attendre le front d’horloge suivant.

# CONCLUSION

Nous avons étudié les machines de *Mealy* et *Moore* à travers les exemples d’un contrôleur de mémoire et d’un compteur. Nous avons remarqué que la description du système dans l’architecture ainsi que le timing des entrées influençaient la rapidité du composant. Il serait intéressant de simuler les mêmes éléments en prenant en compte les temps de propagations.