

INF1500

Logique des systèmes numériques

Laboratoire 5

Soumis par:

*Hannachi, Skander - 2088988*

*Benthami, Omar* - *2114098*

*Date*

# Description du système

**DESCRIPTION DU CIRCUIT :**

Durant ce laboratoire nous avons dû concevoir un système composé de plusieurs sous-systèmes dont, CALCULATRICE, le module déjà connu 7 SEG, de 4 modules « Pulse Gen » et de 4 modules « Debounce ».

D’abord, la calculatrice comportera 4 entrées (machine de MEALEY), soit une pour une addition de 1, soustraction de 1, multiplication par 2 ainsi que la division par 2. Par conséquent, la machine aura 8 états différents. Ainsi ces entrées seront ADD, SST, MULT, DIV, RST et CLK pour nous donner une sortie sur 3 bits. Au-delà de l’intervalle de représentation possible (0 à 7), on effectuera un modulo par 8 des chiffres à représenter.

Ensuite, il y a le module 7SEG qui lui prendra comme entrée la sortie de la calculatrice. Il servira à représenter en notation décimale le résultat de la calculatrice. Il prend alors en entrée un nombre sur 3 bits, puis renvoie en sortie un nombre sur 8 bits qui servira à effectuer l’affichage sur une LED. Bien que la machine de Mealy soit asynchrone, notre circuit sera synchronisé par le biais d’une clock reliée au module CALCULATRICE et au RESET.

Par la suite, on inclut lit à la clock du circuit 4 modules nommés Pulse Gen, qui sont de machines de Moore. Ces machines serviront à synchroniser le signal d’entrée de la calculatrice avec la clock. Un autre de leur rôle est d’allonger la période de l’entrée. C’est une mesure nécessaire puisqu’une période d’appui pour modifier une entrée est plus grande qu’une période de clock. Effectivement, il se pourrait que l’horloge effectue plusieurs cycles le temps d’appuyer une fois et par conséquent effectuer des additions par 1 non désirées. Pour y remédier, la machine de Moore aura 2 états, soit S0 et S1 ayant chacun les valeurs respectives 0 et 1. Lorsque la valeur d’entrée est à 1, à l’état S0, on passe à l’état S1 pour retourner ensuite à 0 au prochin front montant de la clock.

De plus, nous utilisons dans le circuit 4 filtres anti-rebonds (Debounce) qui serviront à ne tenir compte que d’un seul signal lors de l’appui. Ils seront placés avant les Pulse Gens.

Enfin, pour activer l’affichage SEVEN\_SEG, on placera une sortie constante AN (fixée à 0xFE).

**RÉDACTION EN VHDL :**

**CALCULATRICE**

La première étape est bien sûr d’implémenter le registre state en séquentielle (issu des notes du prof). Pour la calculatrice, nous devions d’abord définir l’entité CALCLTRC qui prend comme entrée ADD, SST, MULT, DIV, CLK et RST. Elle nous donnera une sortie que nous appelons RESULT (représenté sur 3 bits). La machine Mealy sera faite à partir un case, qui définit tous les états possibles que peut prendre le module (soit de 0 à 7) selon chaque opération effectuée.

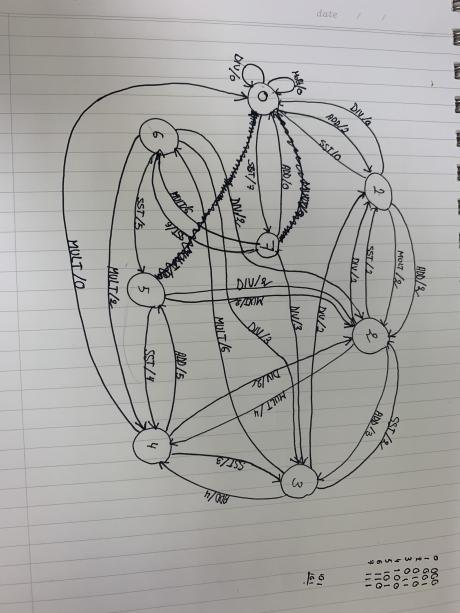
**PULSEGEN**

Pour ce module nous avons suivi le code proposé sur moodle. Son implémentation fut quasiment la même.

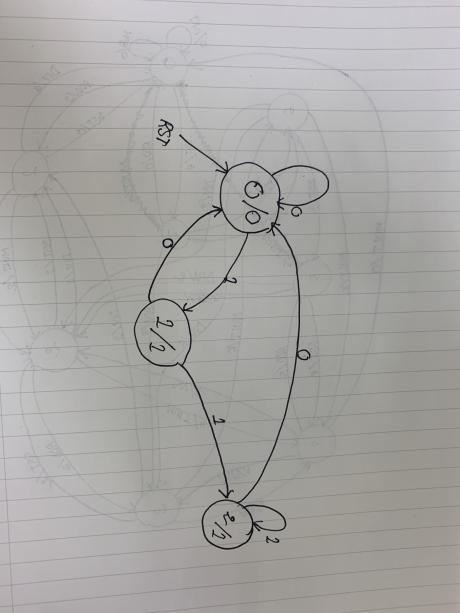
**CIRCUIT FINAL**

Enfin, l’implémentation du circuit final se fait comme suit. On définit l’entité FINALE à laquelle on y met les 6 entrées soit les opérateurs, le reset et la clock. Il faut aussi définir les 10 éléments composant le circuit. Par la suite, on définit les 9 signaux. On représente ensuit ce qui les relie soit DEBOUNCE vers PULSGEN, les PULSGEN vers la calculatrice, puis la CALCULATRICE vers le 7 SEG. Pour finir, il nous faut nommer les entrées et sorties des modules et fixer une constante de valeur FE à AN, ce qui conclut le circuit.

## Schémas des circuits



*Machine de Mealy (Calculatrice)*

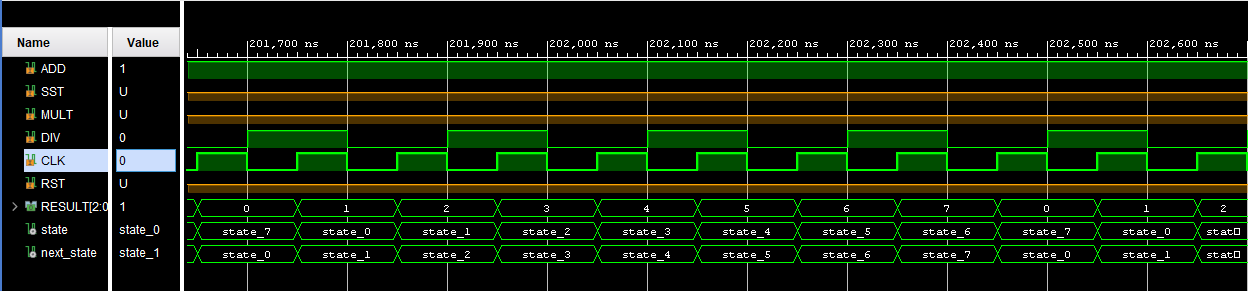


*Machine de Moore (Générateur de Pulsion)*

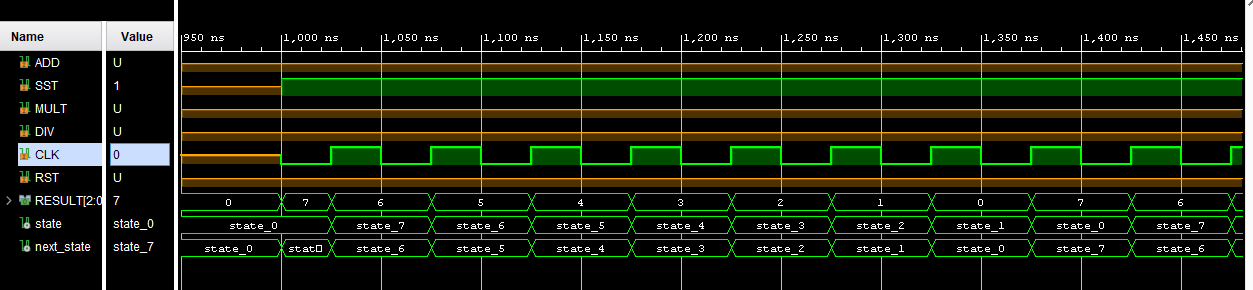
## Validation du système

Pour effectuer la validation de nos différents modules. Nous avons décidé de faire des tests non-exhaustifs sur Vivado, puisque d’effectuer toutes les opérations que propose le module calculatrice sur tous les combinaisons de chiffres binaires en entrée aurait été quasi-impossible. Nous avons donc sur Vivado forcé aux entrées des modules des constantes pour voir si le résultat obtenu correspond bien à la sortie désirée.

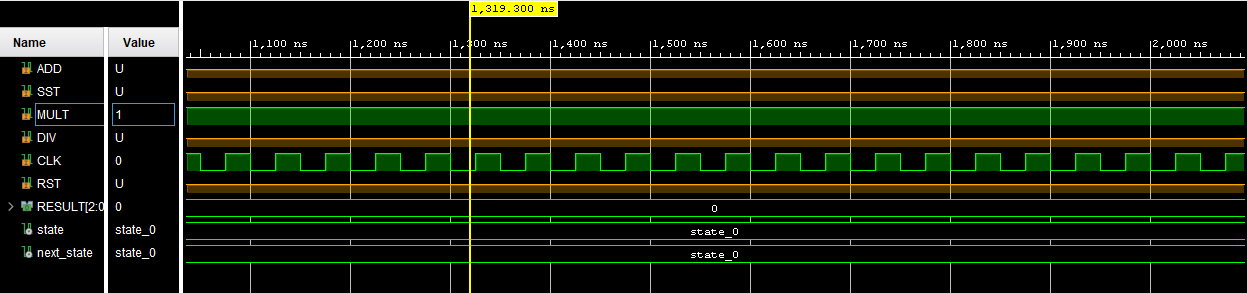
**CHRONOGRAMME D’UN TEST NON-EXHAUSTIF DU MODULE ADD**



**CHRONOGRAMME D’UN TEST NON-EXHAUSTIF DU MODULE SST**

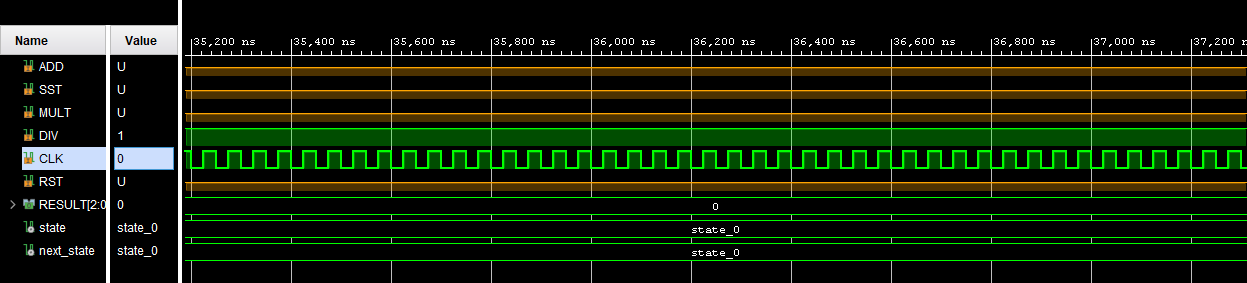


**CHRONOGRAMME D’UN TEST NON-EXHAUSTIF DU MODULE MULT**



\*Note : La simulation n’est pas supposée nous donner ce résultat. En effet, il est supposé y avoir un changement d’état pour presque toutes les multiplications. L’erreur provient de notre code VHDL.

**CHRONOGRAMME D’UN TEST NON-EXHAUSTIF DU MODULE DIV**



\*Note : La simulation n’est pas supposée nous donner ce résultat. En effet, il est supposé y avoir un changement d’état pour presque toutes les multiplications. L’erreur provient de notre code VHDL.

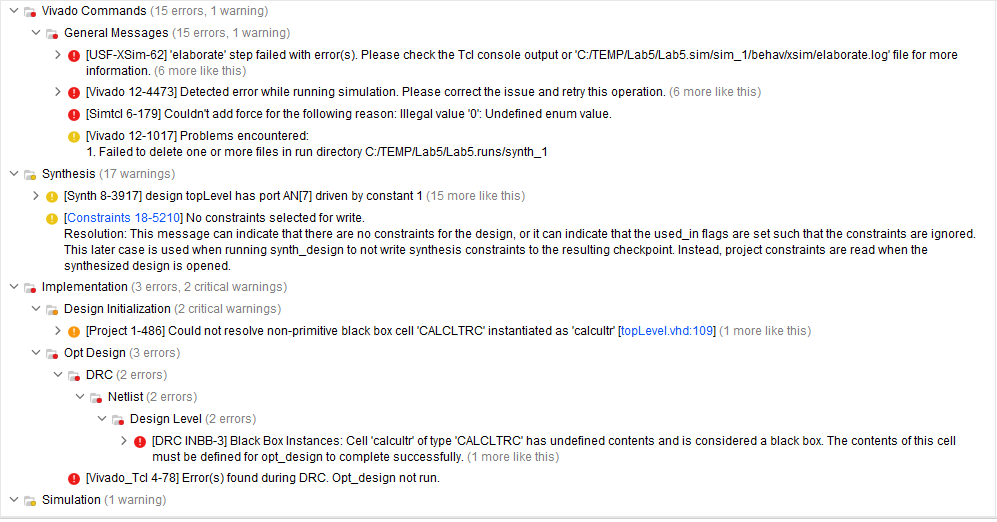
## Observations et validation

**Modules d’addition et de Soustraction – Valides**

Nous pouvons constater que les deux premiers modules semblent justes et valides. En effet, pour le module d’addition, nous observons bien une incrémentation de 1 à chaque front montant de la clock pour le chiffre fixé en entrée (0. Ceci concorde donc avec la théorie. Ensuite, nous pouvons observer que le module de soustraction nous permet bien de décrémenter le chiffre fixé en entrée (7).

**Modules de multiplication et de division – Non Valides**

Les chronogrammes des modules de division et de multiplication ne nous donnent pas les valeurs de sortie espérées. En effet, la seule sortie observée sur celui de la multiplication est une sortie de 1 et celle sur celui de la division est UNDEFINED. Les modules sont donc non-valides. Bien que nous ayons cherché à trouver l’erreur dans le code VHDL nous n’en somment pas parvenus. Or, lorsque nous voulions générer le bitstream (évidemment cela ne marchait pas) nous avons obtenue une liste d’erreur que l’on va essayer d’analyser…



On peut voir au bas de la page que Vivado nous indique que notre module CLCLTRC contient des éléments non-définis. Cette erreur empêche la génération du bitsream lors de la synthèse puisqu’en raison de ces éléments dits « manquants » , c’est tout le module qui ne peut être compilé.

## Réponses aux questions

## Questions 1, 2 et 3 :

## MODULE CLCLTRC Moore ou Mealy?

## D’abord pour notre module de calculatrice nous avons opté pour la machine de Mealy. En effet, celle-ci nous donne un résultat qui prend en considération l’état présent ainsi que des entrées ce qui est la solution optimale pour la réalisation de ce module. Puisque nous pouvons obtenir une entrée à tout moment, il est primordial que le résultat de la machine ne dépende pas des fluctuations de l’horloge (attendre au prochain front montant pour un changement de valeur). Enfin, la machine nous donnera un résultat différent selon son état. D’ailleurs, chaque opération en entrée devrait par logique nous donner un résultat différent en sortie, mais aussi il est à tenir en compte qu’un même opérateur peur nous donner une sortie différente selon l’état.

## **Preuve :**

## Comme convenu, une machine à état de Mealy dépend bien de l’entrée et de l’état. C’est le cas pour le module CALCLTRC dont la sortie va dépendre de l’opérateur sélectionné en entrée puis du l’état conséquent.

## **Séquentiel ou combinatoire?**

## Il en va de soi que ce module est un circuit séquentiel puisque sa sortie dépend des sorties précédentes ainsi que de l’opérateur en entrée.

## **MODULE PULSEGEN**

## **Moore ou Mealy?**

## Afin d’avoir un résultat indépendant des entrées, nous avons choisi la première option. En effet, le signal envoyé au départ sera synchronisé avec celui de la clock.

## **Preuve :**

## Évidemment, ce module est une machine de Moore puisque ses états sont synchronisés avec le signal de l’horloge. En effet, l’état ne changera que lors d’un front montant.

## **Séquentiel ou combinatoire?**

## Encore ici, nous avons affaire à un circuit séquentiel. En effet, l’ordre des sorties précédentes dictera la valeur de sortie de la suivante.

## **MODULE SEVEN\_SEG**

## Nous avons cette fois un circuit combinatoire. La sortie du module ne dépendra que de la valeur d’entrée.