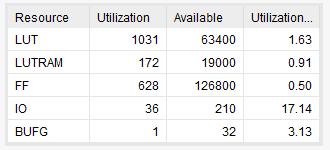
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| LUT | Bascules | Fréquence (MHz) | Latence (ns) | Débit (opérations/s) |
| 1031 | 628 | 121,728 | 3920 | 255102 |



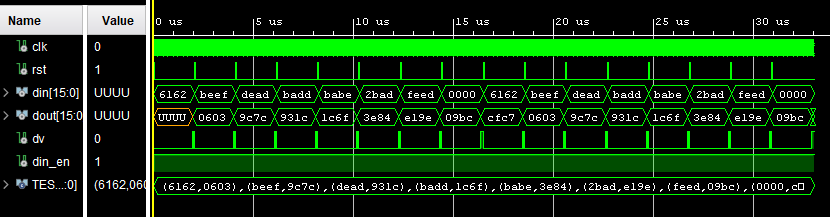


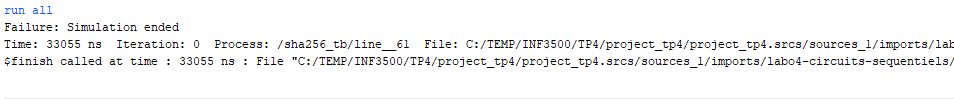
Fréquence = 1/((8.054+0.161) \* 10-9) = 121,728 MHz

Nombre de cycles : 1 + 3\*64 + 1 + 1 + 1 = 196 cycles d’horloges par opération.

Latence = 196\*20 ns = 3920 ns

Débit = 1/(3920 \* 10-9) = 255 102





Questions

1. Notre module n’aurait pas pu contenir de boucle car il faut plusieurs cycles d’horloges pour exécuter une instruction. Cela est dû à l’interdépendance des valeurs : Pour calculer a, il faut au préalable avoir mis à jour T1 qui a besoin de la dernière valeur de W qui vient d’être calculé au début de l’instruction, or cette valeur sera mise à jour au prochain front d’horloge. On se doit donc de diviser notre instruction en 3 états et de dérouler nous-même la boucle.
2. Le module debouncer une première bascule met à jour la valeur de output et une autre celle de counter. Les deux dépendent de input et de counter. Le reset remet les valeurs à 0. Le module va attendre une entrée input. Tant qu’une entrée est présente, à chaque front montant d’horloge la variable counter va s’incrémenter et la valeur output va rester à 0, jusqu’à ce que counter atteigne sa valeur maximale à savoir 2\*\*W – 1. Le counter va arreter de s’incrémenter et le output va passer à 1. S’il n’y a pas ou plus d’entrée, counter et output sont remis à 0.

Le module pulse est composée de 2 bascules : une pour mettre la valeur de state à jour et une pour la valeur de output. Le reset met les deux valeurs à 0. State prend la valeur de input au front montant de l’horloge. Output prendre la valeur de l’opération logique input and not state.