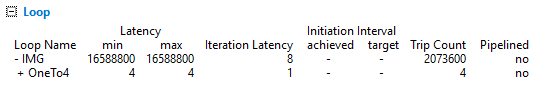
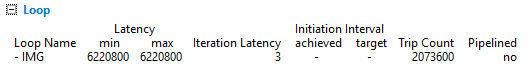
Partie 1

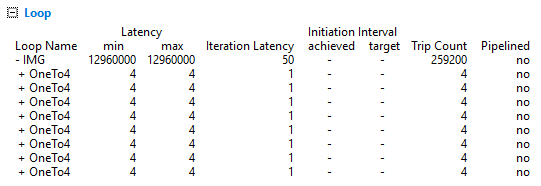
Note de la latence, etc. pour « questions initiales »



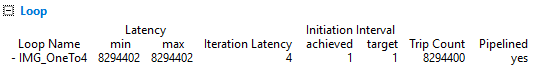
Avec #pragma HLS unroll dans OneTo4 :



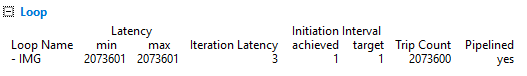
Avec #pragma HLS unroll factor=8 dans IMG :



Avec #pragma HLS pipeline dans OneTo4 :



Avec #pragma HLS pipeline dans IMG :



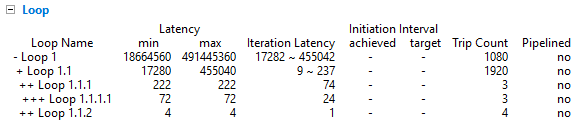
Question 1 :

Il faudrait dérouler les deux boucles. Il est utile de dérouler une petite boucle imbriquée de ce genre, qui ne fait que rendre 2D une certaine opération (qu’on aurait aussi bien pu indexer en 1D), car celle-ci a avantage à être parallélisée. Il n’y a d’ailleurs pas d’interdépendance entre les différents calculs, puisque chaque itération de la boucle affecte seulement un unique résultat dans la matrice 3x3. On peut observer l’accélération qu’amène cette méthode dans les résultats du *#pragma HLS unroll* sur OneTo4, cela rendant possible un déroulement efficace des boucles imbriquées. Il ne serait pas bénéfique de plutôt pipeliner l’opération, puisque l’on s’attend à un mode d’opération irrégulier. On se retrouverait donc souvent avec un pipeline vide, ce qui résulte en un avantage minime par rapport à une implémentation sans déroulement ni pipeline (on peut l’observer, supposant un pipeline plein toutefois, dans les résultats du *#pragma HLS pipeline* sur OneTo4).

Question 2 :

Il faudrait pipeliner la boucle principale. En effet, pipeliner une opération régulière de grande envergure (comme le passage sur chaque pixel d’une image de grande taille) permet de réduire le délai de traitement entre chaque pixel à un seul cycle, une fois le premier traité. Cela amène une grande accélération, comme on peut le voir dans les résultats du *#pragma HLS pipeline* dans IMG (qui pipeline implicitement la boucle interne en plus de la boucle principale). En revanche, dérouler la boucle principale aurait un impact minime sur l’accélération du système, puisque la boucle interne s’en retrouverait simplement multipliée en de nombreuses instances. On peut observer ce comportement dans les résultats du *#pragma HLS unroll factor=8* sur IMG.

Résultats pour l’implémentation de première partie (logicielle) :



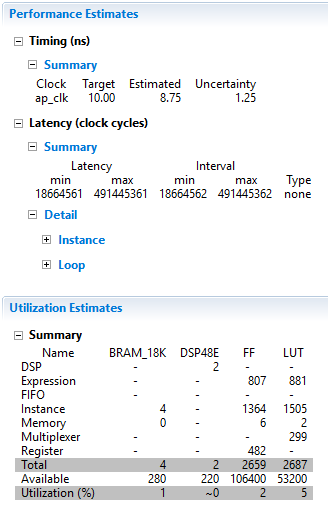
Performance : ~ 3.1292 secondes/frame 🡪 ~ 0.32 FPS

Résultats pour première exécution matérielle :

Temps : ~ 17.1455 secondes/frame

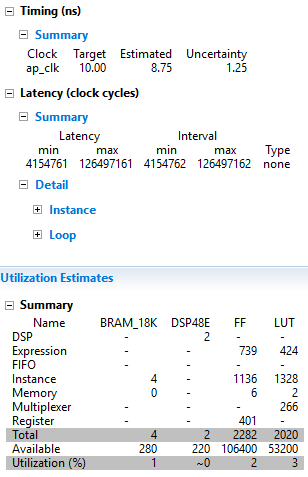
Selon Vivado HLS estimation : 4.91445362 secondes / frame

DSP = 2 (~0%), BRAM = 4 (1%), FF=2659 (2%), LUT=2687 (5%)



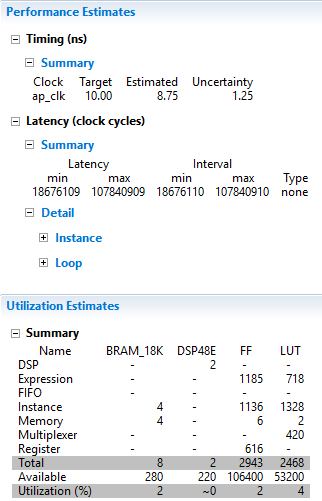
Après utilisation de la 3e signature de fonction et enlevé l’union (utilisation de shifts)

Performance : 3.1698 secondes / frame = 0.315477 FPS



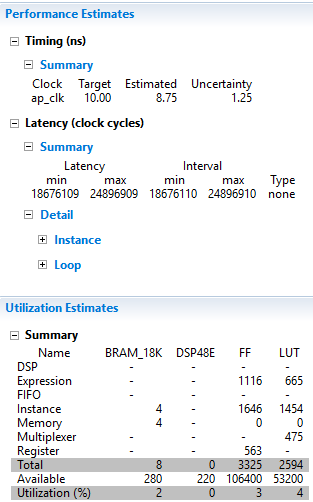
Avec cache à 4 lignes, de base :

Performance : ~ 1.7355 secondes / frame = 0.57617 FPS



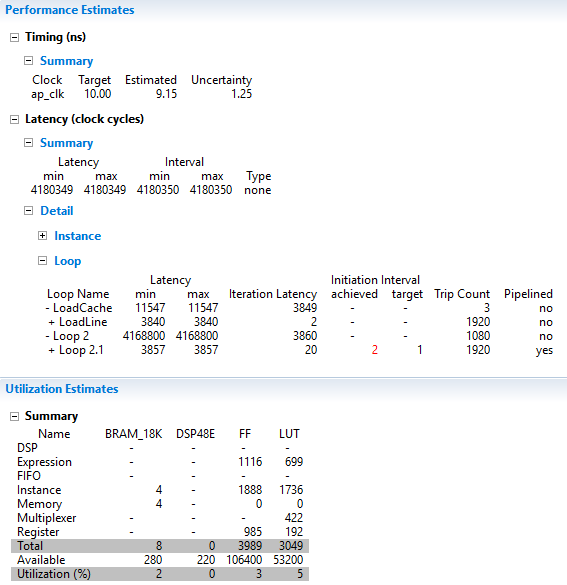
Avec unroll dans sobel\_operator et ARRAY\_PARTITION (complete dim=1) de la cache :

Performance : 0.9029 seconde / frame = 1.1075 FPS



Avec pipeline dans la boucle passant sur la ligne de cache :

Performance : ~ 0.22761 seconde/frame = 4.39348 FPS



Avec optimisation des transactions mémoire pour permettre des burst en lecture/écriture :

Performance : ~ 0.042339 seconde / frame = 23.6189 FPS ☺

