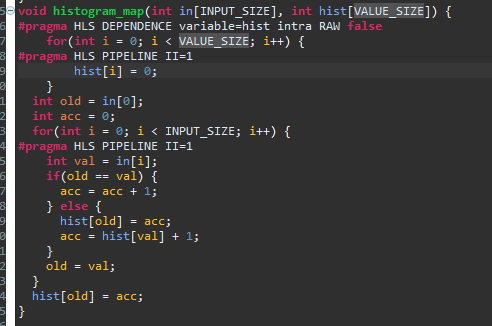
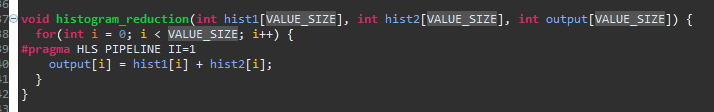
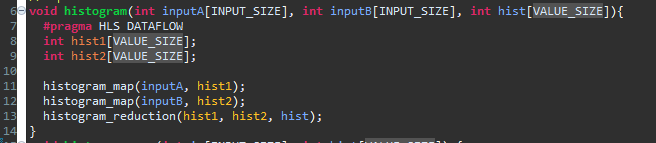
1. Code解說:



首先這個histogram\_map函式，因為我想要更優化直方圖計算，所以我將其分為兩階段來達到更多並行性，而這個 histogram\_map函數的輸入矩陣就是hist陣列中的一個分區數據，將輸入數據分成兩個分區，分別存儲在inputA和inputB陣列中。 它使用histogram\_map函數計算每個分區的直方圖，然後將其存儲在hist1和hist2陣列中。 這兩個陣列被輸入到histogram\_reduce函數中。 該函數將它們合併後並將結果存儲在hist陣列中，其中合併的hist陣列是頂層直方圖函數的最終輸出。首先這個DEPENDENCE PRAGMA是因為每次反覆運算迴圈中，我們讀取另一個hist陣列中另一個位置x1中陣列。 x0和x1都取決於輸入值，並可以取任何值。 因此我們考慮到綜合成電路時最壞的情況，如果x0 = x1，則在前一個寫入完成前，位置x1處的讀取無法開始。 因此，我們必須在讀寫之間進行切換。再來使用PIPELINE PRAGMA在展開內部迴圈時能夠得到更少的時間，再來下面的迴圈可以看到流程中的'loop'包含存儲out[]陣列的輸出存儲部分，流程中的迴圈部分僅包含一個存儲累加值的寄存器和只寫的輸出記憶體。



Histogram\_reduce 函數實現了模式中的「還原」部分。 它將分區數據的直方圖作為輸入，並通過將每個直方圖的計數相加，將它們組合成完整的直方圖。因為我們只有兩個處理物件，因此將兩個輸入陣列hist1和hist2合併。 這可以很容易的擴展以處理更多的元素。



在histogram函數中使用dataflow指令來達到流水線設計。 這邊有三個模組：兩個histogram\_map函數和一個histogram\_reduce函數。因為兩個histogram\_map處理的數據相互獨立，所以可以同時執行。 Histogram\_reduce 函數處理過程必須在histogram\_map處理完成後才開始。 因此，dataflow 指令本質上是創建了一個兩階段的管道。 第一階段執行histogram\_map函數，而第二階段執行histogram\_reduce函數。

1. 執行結果:

