摘要：

本次 Lab 共包含 Prefix Sum 與 histogram 兩種不同運算的優化。

Prefix Sum：

運算說明：

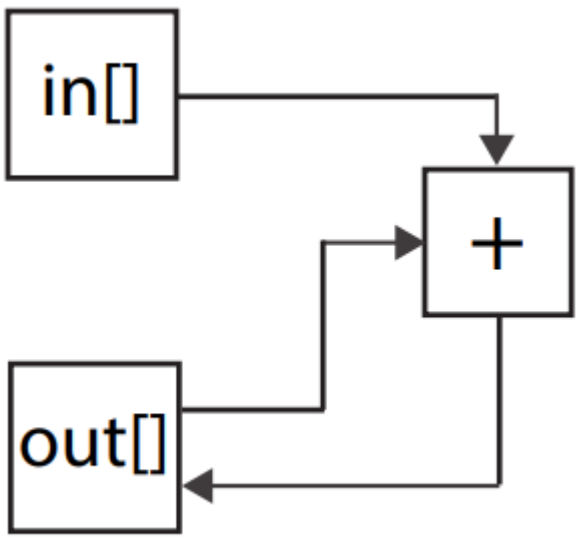
Prefix sum 計算的是一個輸入序列的累加值，具體公式如下：

$$out_0 = in_0$$
$$out_1 = in_0 + in_1$$
$$out_2 = in_0 + in_1 + in_2$$
$$out_3 = in_0 + in_1 + in_2 + in_3$$
$$\cdots$$
$$out_n = out_{n-1} + in_n$$

原始設計：

Prefix sum 為 top function，in 與 out 皆以 maxi 介面傳輸。 (SIZE = 128)

```
void perfixsum(int in[SIZE], int out[SIZE]){
    out[0] = in[0];
    for(int i=1; i<SIZE; i++){
        #pragma HLS PIPELINE
        out[i] = out[i-1] + in[i];
    }
}
```



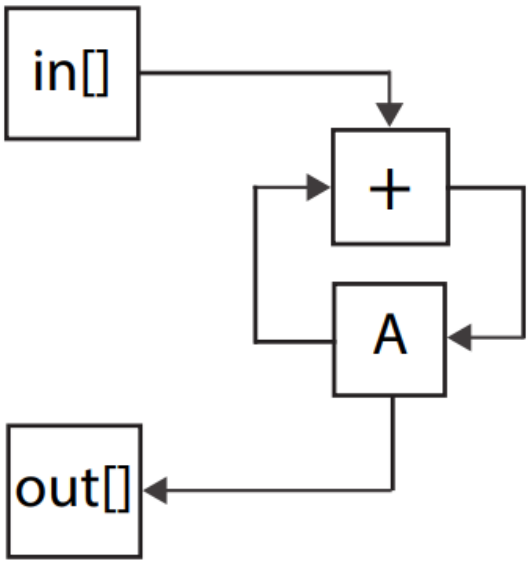
Modules & Loops	Issue Type	Slack	Latency (cycles)	Latency (ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
+ perfixsum	Timing	-0.00	159	1.590e+03	-	160	-	no	4 (1%)	-	1122 (1%)	1704 (3%)	-
+ perfixsum_Pipeline_VITIS_LOOP_5_1	Timing	-0.00	131	1.310e+03	-	131	-	no	-	-	113 (~0%)	136 (~0%)	-
o VITIS_LOOP_5_1	-	7.30	129	1.290e+03	4	1	127	yes	-	-	-	-	-

耗時：159 個 cycles

問題點：out[i-1] 與 out[i] 之間存在 RAW 的 memory dependency 關係，從 memory 讀回 output array 的值需要時間，而這個讀回的動作可以用一個 local 變數代替。

使用 local 變數優化：

```
void perfixsum(int in[SIZE], int out[SIZE]){
    int A = 0;
    for(int i=0; i<SIZE; i++){
        #pragma HLS PIPELINE rewind
        A += in[i];
        out[i] = A;
    }
}
```



Modules & Loops	Issue Type	Slack	Latency (cycles)	Latency (ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
+ perfixsum	Timing	-0.00	146	1.460e+03	-	147	-	no	4 (1%)	-	948 (~0%)	1384 (2%)	-
+ perfixsum_Pipeline_VITIS_LOOP_8_1	Timing	-0.00	132	1.320e+03	-	132	-	no	-	-	113 (~0%)	136 (~0%)	-
o VITIS_LOOP_8_1	-	7.30	130	1.300e+03	4	1	128	yes	-	-	-	-	-

耗時：147 個 cycles

總執行時間雖然縮短，但奇怪的是兩者 II 皆為 1，而主要時間差似乎在 loop 外面?

使用 loop unroll 優化：

由於 loop unroll 需要搭配 array partition，而 in 與 out 屬於 maxi 介面，因此需要先複製到 local array，進行 prefix sum 運算後再一次傳輸回去。

```
void perfixsum(int in[SIZE], int out[SIZE]){
    int local_in[SIZE], local_out[SIZE];
    #pragma HLS ARRAY_PARTITION variable=out cyclic factor=4 dim=1
    #pragma HLS ARRAY_PARTITION variable=in cyclic factor=4 dim=1
    for(int i=1; i<SIZE; i++){
        local_in[i] = in[i];
    }
    local_out[0] = local_in[0];
    for(int i=1; i<SIZE; i++){
        #pragma HLS UNROLL factor=4
        #pragma HLS PIPELINE
        local_out[i] = local_out[i-1] + local_in[i];
    }
    for(int i=1; i<SIZE; i++){
        out[i] = local_out[i];
    }
}
```

PS: '+' for module; 'o' for loop; '*' for dataflow

Modules & Loops	Issue Type	Slack	Latency (cycles)	Latency (ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
+ perfixsum	Timing	-0.00	316	3.160e+03	-	317	-	no	12 (4%)	-	1368 (1%)	2209 (4%)	-
+ perfixsum_Pipeline_VITIS_LOOP_11_1	Timing	-0.00	130	1.300e+03	-	130	-	no	-	-	60 (~0%)	77 (~0%)	-
o VITIS_LOOP_11_1	-	7.30	128	1.280e+03	3	1	127	yes	-	-	-	-	-
+ perfixsum_Pipeline_VITIS_LOOP_15_2	-	0.61	36	360.000	-	36	-	no	-	-	386 (~0%)	342 (~0%)	-
o VITIS_LOOP_15_2	-	7.30	34	340.000	5	1	31	yes	-	-	-	-	-
+ perfixsum_Pipeline_VITIS_LOOP_20_3	Timing	-0.00	130	1.300e+03	-	130	-	no	-	-	47 (~0%)	107 (~0%)	-
o VITIS_LOOP_20_3	-	7.30	128	1.280e+03	3	1	127	yes	-	-	-	-	-

耗時：316 個 cycles

其中 260 個 cycle 是 host 與 local array 的傳輸，而真正 prefix sum 的運算則降低至 34 個 cycles (相較前者使用 local 變數優化需要 132 個 cycles)。

如果 prefix sum 不是 top function，則可以採用此方案 (因為不需要 maxi 的傳輸時間) 。

在 FPGA 板子上的執行結果：

```
Entry: /usr/lib/python3/dist-packages/ipykernel_launcher.py
System argument(s): 3
Start of "/usr/lib/python3/dist-packages/ipykernel_launcher.py"
Kernel start!

Kernel execution time: 0.00010156631469726562 s
Test passed!
============================
Exit process
```

Histogram：

運算說明：

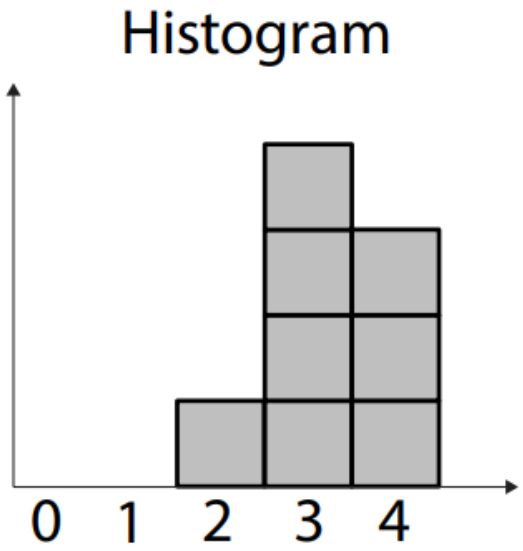
Histogram 計算的是一個輸入序列中每個值出現的次數分布，具體範例如下：

Data

4	2	4	4	3	3	3	4

Histogram Values

Bins	2	3	4
Counts	1	4	3



原始設計：

Histogram 為 top function，in 與 hist 皆以 maxi 介面傳輸。 (INPUT_SIZE = 1024, VALUE_SIZE = 128)

```
void histogram(int *in, int *hist){
    int val;
    for(int i = 0; i < INPUT_SIZE; i++) {
        #pragma HLS PIPELINE
        val = in[i];
        hist[val] += 1;
    }
}
```

```
PS: '+' for module; 'o' for loop; '*' for dataflow
+--------------------+--------+-------+---------+-----------+----------+---------+------+----------+--------+----+-----------+-----------+-----+
|       Modules      |  Issue |       | Latency |  Latency  | Iteration|         | Trip |          |        |    |           |           |     |
|       & Loops      |  Type  | Slack | (cycles)|    (ns)   |  Latency | Interval| Count| Pipelined|  BRAM  | DSP|     FF    |    LUT    | URAM|
+--------------------+--------+-------+---------+-----------+----------+---------+------+----------+--------+----+-----------+-----------+-----+
|+ histogram         |  Timing|  -0.00|    15371|  1.537e+05|         -|    15372|     -|        no|  4 (1%)|   -|  1077 (1%)|  1266 (2%)|    -|
| o VITIS_LOOP_41_1  |      II|   7.30|    15369|  1.537e+05|        25|       15|  1024|       yes|       -|   -|          -|          -|    -|
+--------------------+--------+-------+---------+-----------+----------+---------+------+----------+--------+----+-----------+-----------+-----+
```

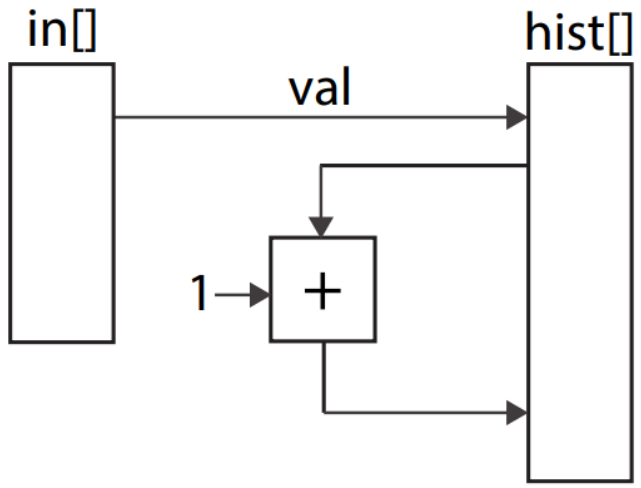
耗時：15371 個 cycles

問題：由於 hist 的讀寫順序不規則，maxi 在此情況下讀寫效率極低，因此 loop interval 高達 15 個 cycles。

使用 local array 優化：

因此為了提高效率先將 hist 複製到 local array 再進行運算。

```
void histogram(int *in, int *hist){
    int val, local_hist[VALUE_SIZE];
    for(int i = 0; i < VALUE_SIZE; i++){
        local_hist[i] = hist[i];
    }
    for(int i = 0; i < INPUT_SIZE; i++) {
        #pragma HLS PIPELINE
        val = in[i];
        local_hist[val] += 1;
    }
    for(int i = 0; i < VALUE_SIZE; i++){
        hist[i] = local_hist[i];
    }
}
```



PS: '+' for module; 'o' for loop; '*' for dataflow

Modules & Loops	Issue Type	Slack	Latency (cycles)	Latency (ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
+ histogram	Timing	-0.00	2340	2.340e+04	-	2341	-	no	6 (2%)	-	1208 (1%)	1849 (3%)	-
+ histogram_Pipeline_VITIS_LOOP_9_1	Timing	-0.00	131	1.310e+03	-	131	-	no	-	-	62 (~0%)	77 (~0%)	-
o VITIS_LOOP_9_1	-	7.30	129	1.290e+03	3	1	128	yes	-	-	-	-	-
+ histogram_Pipeline_VITIS_LOOP_30_2	Timing	-0.00	2052	2.052e+04	-	2052	-	no	-	-	187 (~0%)	234 (~0%)	-
o VITIS_LOOP_30_2	II	7.30	2050	2.050e+04	5	2	1024	yes	-	-	-	-	-
+ histogram_Pipeline_VITIS_LOOP_36_3	Timing	-0.00	131	1.310e+03	-	131	-	no	-	-	46 (~0%)	75 (~0%)	-
o VITIS_LOOP_36_3	-	7.30	129	1.290e+03	3	1	128	yes	-	-	-	-	-

耗時：2340 個 cycles

II 從先前的 15 降低至 2，而 II 無法降到 1 的原因出在 hist 存在 RAW 的 memory dependency！

消除 RAW Memory Dependency：

將次數寫回的時間點延後到下一個迴圈，若連續次讀到相同值則先累加，直到讀到不同值再寫回。避免了 RAW 也降低了 array 的存取次數。

```
void histogram(int *in, int *hist){
    int val, local_hist[VALUE_SIZE];
    for(int i = 0; i < VALUE_SIZE; i++){
        local_hist[i] = hist[i];
    }
    int old, acc;
    val = in[0];
    acc = local_hist[val] + 1;
    #pragma HLS DEPENDENCE variable=local_hist intra RAW false
    for(int i = 1; i < INPUT_SIZE; i++) {
        #pragma HLS PIPELINE II=1
        old = val;
        val = in[i];
        if(old == val){
            acc += 1;
        }else{
            local_hist[old] = acc;
            acc = local_hist[val] + 1;
        }
    }
    local_hist[val] = acc;
    for(int i = 0; i < VALUE_SIZE; i++){
        hist[i] = local_hist[i];
    }
}
```

PS: '+' for module; 'o' for loop; '*' for dataflow

Modules & Loops	Issue Type	Slack	Latency (cycles)	Latency (ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
+ histogram	Timing	-0.00	1318	1.318e+04	-	1319	-	no	6 (2%)	-	1308 (1%)	1928 (3%)	-
+ histogram_Pipeline_VITIS_LOOP_7_1	Timing	-0.00	131	1.310e+03	-	131	-	no	-	-	62 (~0%)	77 (~0%)	-
o VITIS_LOOP_7_1	-	7.30	129	1.290e+03	3	1	128	yes	-	-	-	-	-
+ histogram_Pipeline_VITIS_LOOP_15_2	Timing	-0.00	1027	1.027e+04	-	1027	-	no	-	-	117 (~0%)	174 (~0%)	-
o VITIS_LOOP_15_2	-	7.30	1025	1.025e+04	4	1	1023	yes	-	-	-	-	-
+ histogram_Pipeline_VITIS_LOOP_34_3	Timing	-0.00	131	1.310e+03	-	131	-	no	-	-	46 (~0%)	75 (~0%)	-
o VITIS_LOOP_34_3	-	7.30	129	1.290e+03	3	1	128	yes	-	-	-	-	-

耗時：1318 個 cycles

消除 RAW 的 memory dependency 後 II 從 2 進一步降低至 1。