HLS LabB

Prefix Sum and Histogram

摘要

本次Lab共包含Prefix Sum與histogram兩種不同運算的優化

Prefix Sum

$$out_0 = in_0$$

$$out_1 = in_0 + in_1$$

$$out_2 = in_0 + in_1 + in_2$$

$$out_3 = in_0 + in_1 + in_2 + in_3$$

$$\dots$$

$$out_n = out_{n-1} + in_n$$

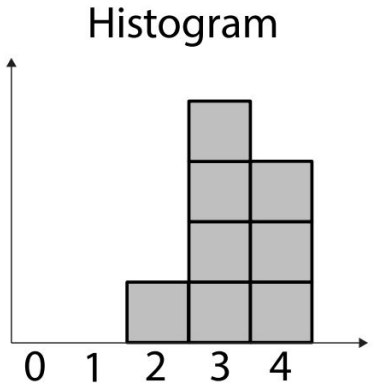
Histogram

Data

3	2	4	4	3	3	3	4

Histogram Values

Bins	2	3	4
Counts	1	4	3



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:

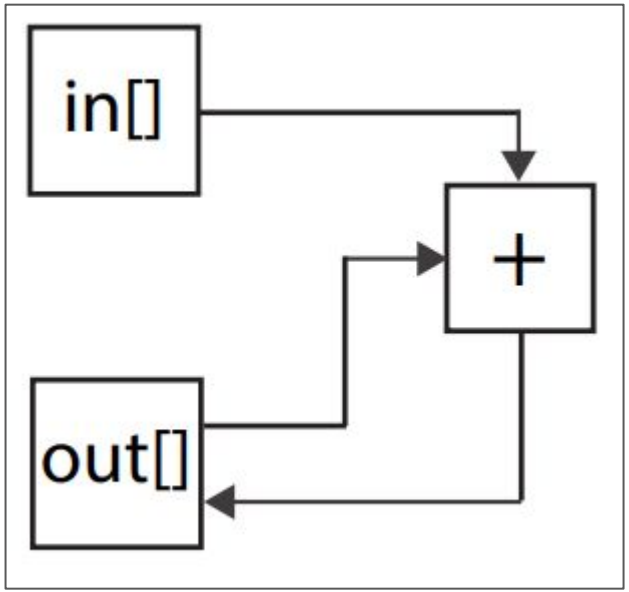
$$
\begin{aligned}
out_0 &= in_0 \\
out_1 &= in_0 + in_1 \\
out_2 &= in_0 + in_1 + in_2 \\
out_3 &= in_0 + in_1 + in_2 + in_3 \\
&\dots \\
out_n &= out_{n-1} + in_n
\end{aligned}
$$

Prefix Sum

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];
    }
}
```



Prefix Sum

原始設計:

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

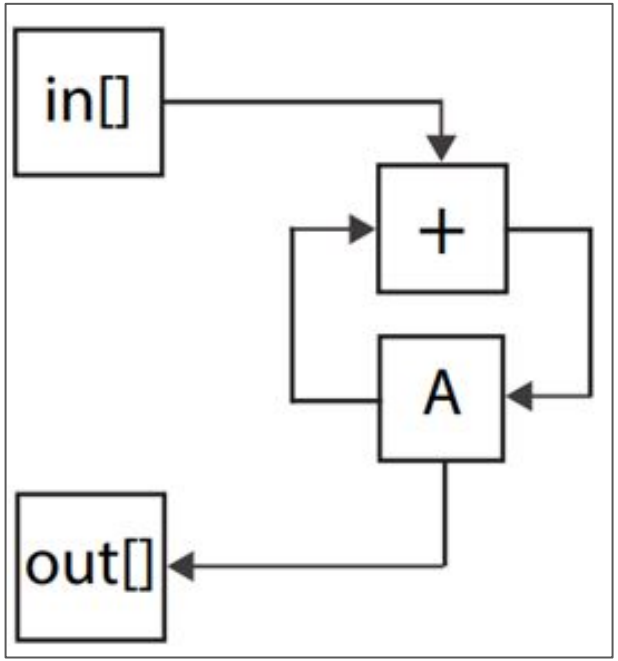
問題點: out[i-1] 與out[i] 之間存在RAW的memory dependency關係, 從memory讀回output array的值需要時間, 而這個讀回的動作可以用一個local變數代替, 直接減少一次memory存取。

Prefix Sum

使用local變數優化:

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



Prefix Sum

使用local變數優化:

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: 146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?

Prefix Sum

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

Prefix Sum

使用loop unroll優化:

Modules & Loops	Issue Type	Slack	Latency (cycles)	Latency (ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
+ perfixsum	Timing	-0.00	315	3.150e+03	-	316	-	no	12 (4%)	-	1227 (1%)	2152 (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_16_2	-	0.61	37	370.000	-	37	-	no	-	-	278 (~0%)	259 (~0%)	-
o VITIS_LOOP_16_2	-	7.30	35	350.000	5	1	32	yes	-	-	-	-	-
+ perfixsum_Pipeline_VITIS_LOOP_30_3	Timing	-0.00	130	1.300e+03	-	130	-	no	-	-	48 (~0%)	95 (~0%)	-
o VITIS_LOOP_30_3	-	7.30	128	1.280e+03	3	1	127	yes	-	-	-	-	-

耗時:315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) 。

Prefix Sum

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:

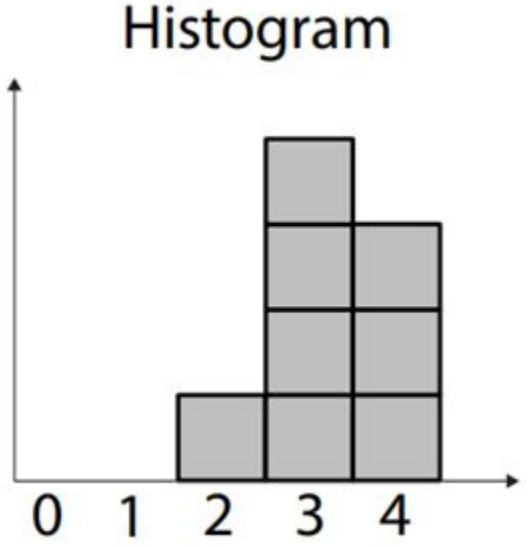
Histogram計算的是一個輸入序列中每個值出現的次數分布，具體範例如下:

Data

3	2	4	4	3	3	3	4

Histogram Values

Bins	2	3	4
Counts	1	4	3

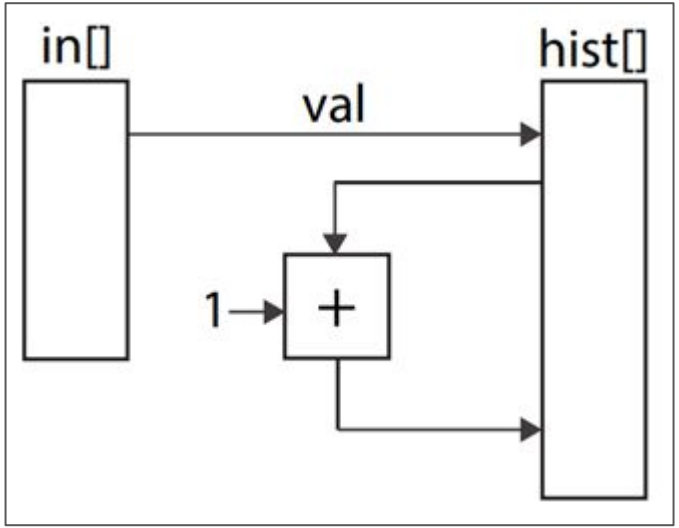


Histogram

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;
    }
}
```



Histogram

原始設計:

```
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

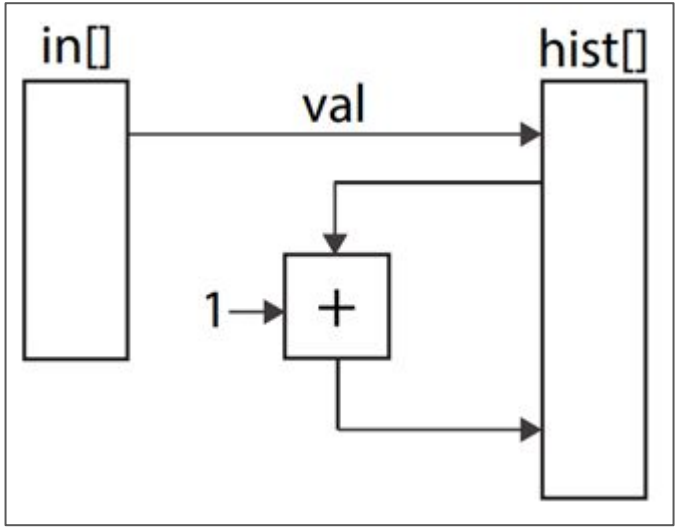
問題:由於hist的讀寫順序不規則, maxi在此情況下讀寫效率極低, 因此loop interval高達15個cycles。

Histogram

使用local array優化:

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];
    }
}
```



Histogram

使用local array優化:

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！

Histogram

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];
    }
}
```

Histogram

消除RAW Memory Dependency：

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。