

Prefix Sum and Histogram

Speaker: Jia-Ming Lin

Outline

- Prefix Sum and its Applications
- Histogram
- Histogram Optimization

Prefix Sum

- Cumulative sum of sequence of numbers

Given $\{in_0, in_1, \dots, in_n\}$

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

Output $\{out_0, out_1, \dots, out_n\}$

Prefix Sum

- Example

Sum of Sequence:

input

6	4	16	10	16	14	2	8
---	---	----	----	----	----	---	---

output

76

Prefix-Sum of Sequence:

input

6	4	16	10	16	14	2	8
---	---	----	----	----	----	---	---

output

6	10	26	36	52	66	68	76
---	----	----	----	----	----	----	----

Recurrence equation

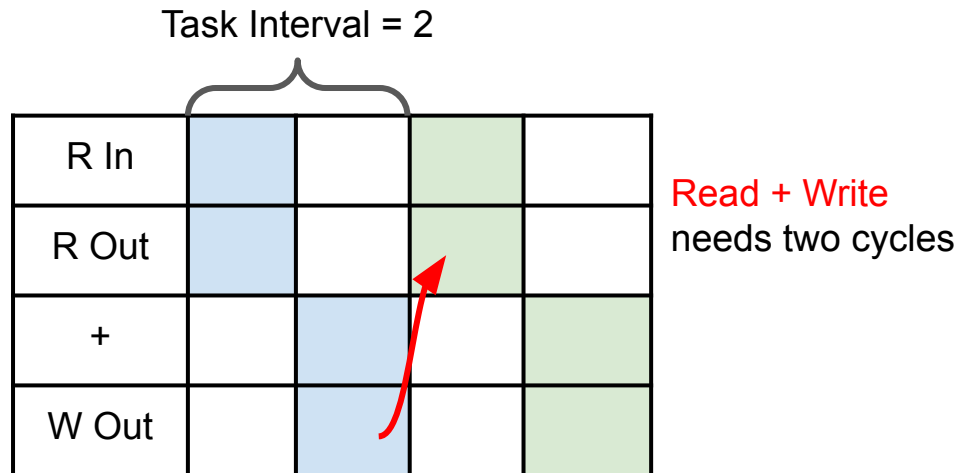
- **Without recompute** sum of all of previous inputs

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

↑
Stored in memory

- HLS implementation

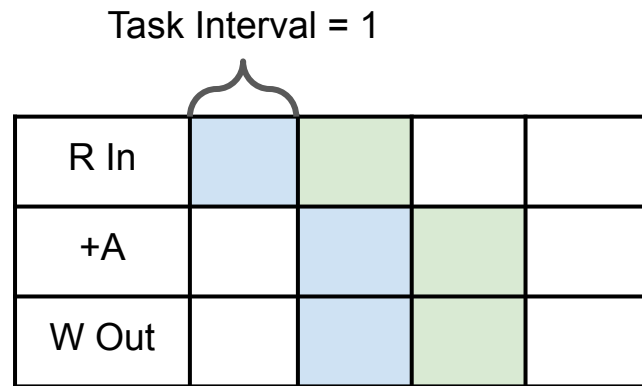
```
#define SIZE 128
void prefixsum(int in[SIZE], int out[SIZE]) {
    out[0]=in[0];
    for(int i = 1; i < SIZE; i++) {
        #pragma HLS PIPELINE
        out[i] = out[i-1] + in[i];
    }
}
```



Recurrence equation

- To achieve Pipeline II=1,
 - Access to BRAM takes too long to achieve.
 - One solution is **reducing the # of accesses to BRAM**
- HLS implementation,

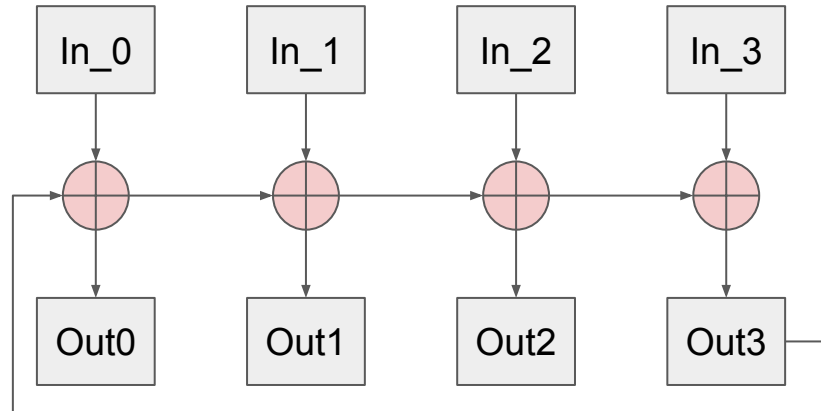
```
#define SIZE 128
void prefixsum(int in[SIZE], int out[SIZE]) {
    int A = in[0];
    for(int i=0; i < SIZE; i++) {
        #pragma HLS PIPELINE
        A = A + in[i];
        out[i] = A;
    }
}
```



One read on out[] is removed

Optimization: Pipeline and Unroll

```
#define SIZE 128
void prefixsum(int in[SIZE], int out[SIZE]) {
    #pragma HLS ARRAY_PARTITION variable=out cyclic factor=4 dim=1
    #pragma HLS ARRAY_PARTITION variable=in cyclic factor=4 dim=1
    int A = in[0];
    for(int i=0; i < SIZE; i++) {
        #pragma HLS UNROLL factor=4
        #pragma HLS PIPELINE
        out[i] = out[i-1] + in[i];
    }
}
```

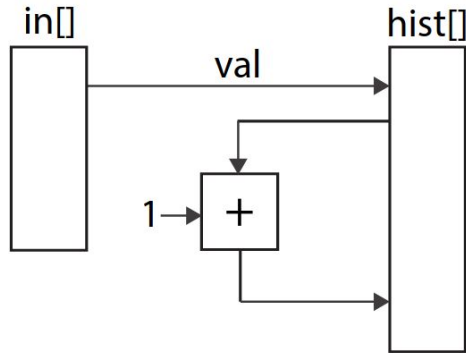


See Timeline Analysis
from Vivado HLS

Histogram

Baseline

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

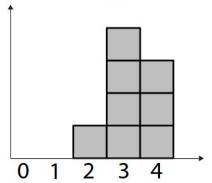


- Given data

4	2	4	4	3	3	3	4
---	---	---	---	---	---	---	---

- Output histogram values

Bins	2	3	4
Counts	1	4	3

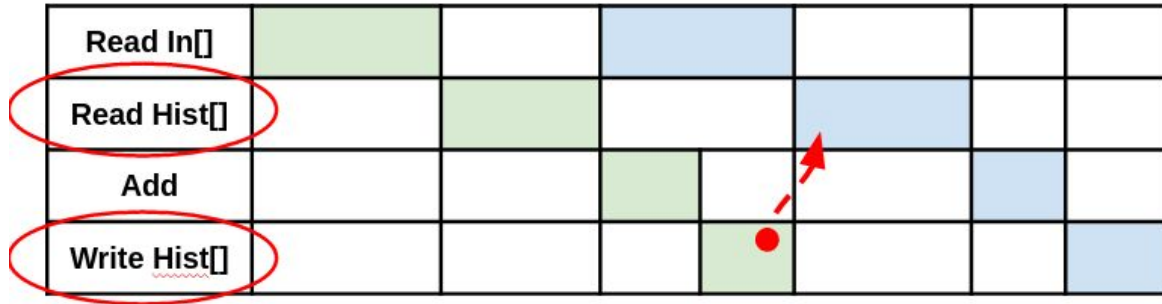


- Sequentially processing on the input data

Read In[]						
Read Hist[]						
Add						
Write Hist[]						

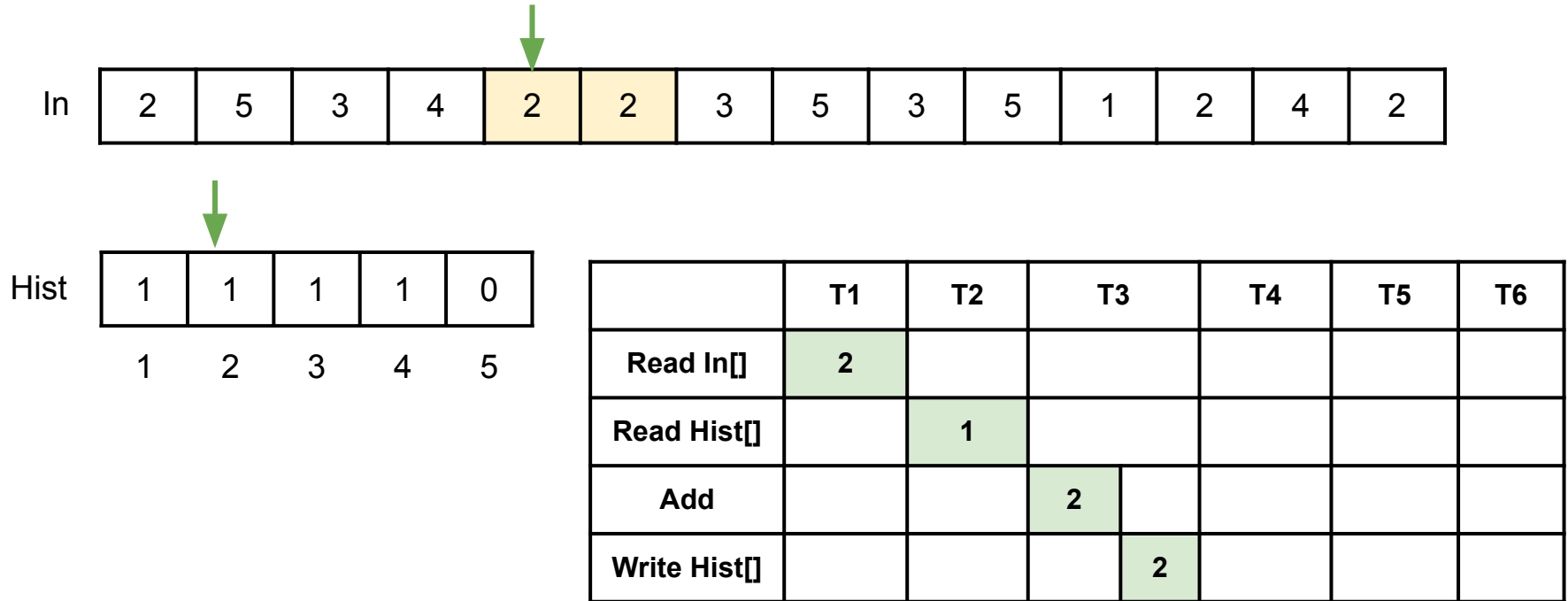
Optimization: Pipeline and Dependencies

- Issue while **using Pipeline**
- **Read Hist[]** dependence on **Write Host[]**
 - They **can not be executed simultaneously.**
 - So, **Task Interval ≥ 2** in this case.
 - We call this **Read after Write(RAW)** Dependency



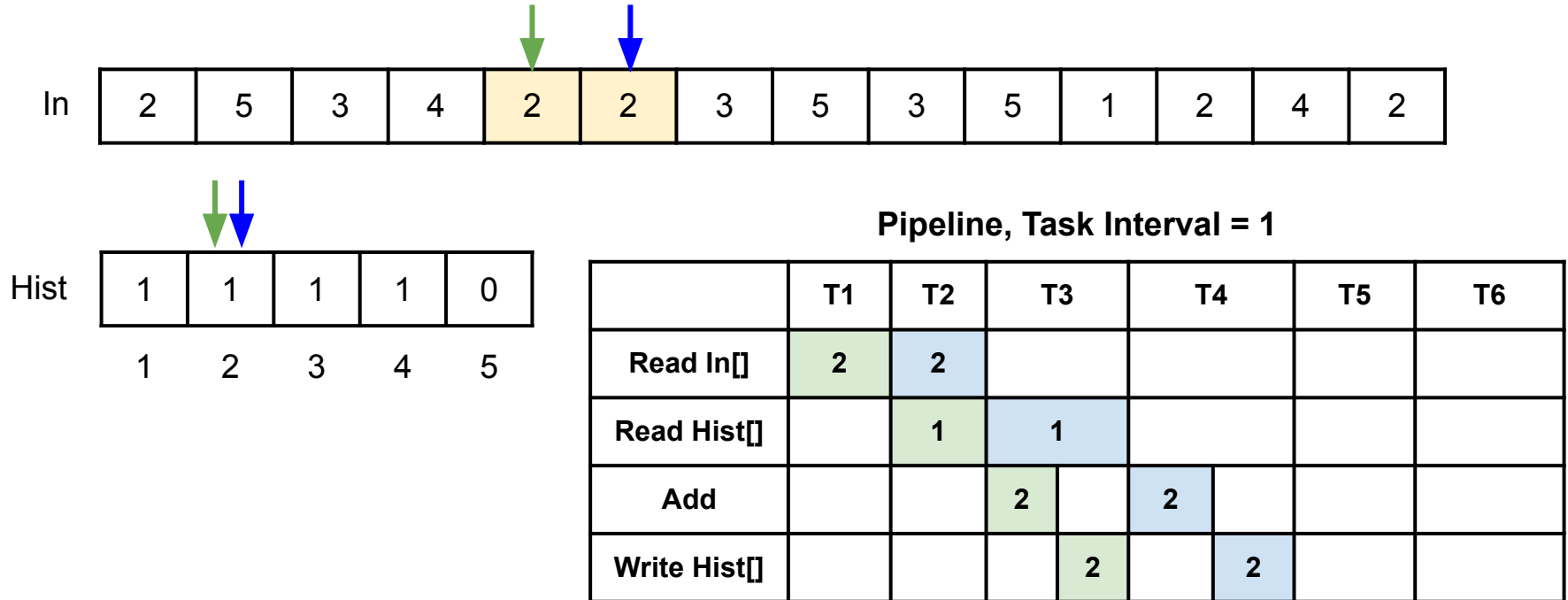
Optimization: Pipeline and Dependencies

- Example: RAW dependency



Optimization: Pipeline and Dependencies

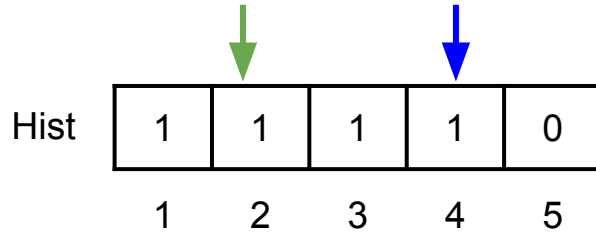
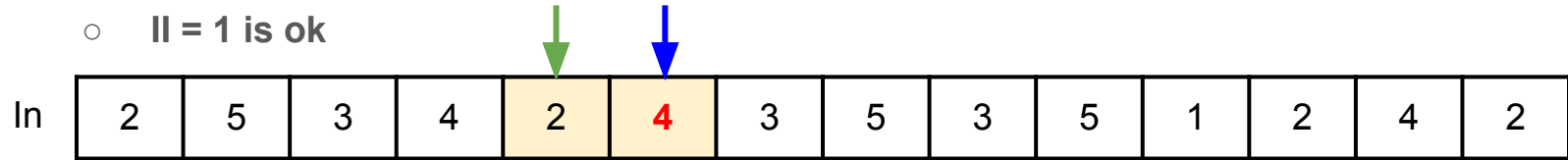
- Example: RAW dependency



Optimization: Pipeline and Dependencies

- Observation: When no two consecutive data are with same value

- $II = 1$ is ok



Pipeline, Task Interval = 1

	T1	T2	T3	T4	T5	T6
Read In[]	2	4				
Read Hist[]		1	1			
Add			2	2		
Write Hist[]				2	2	

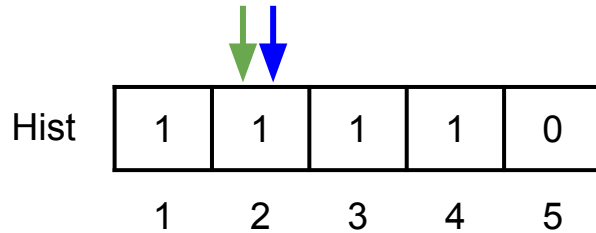
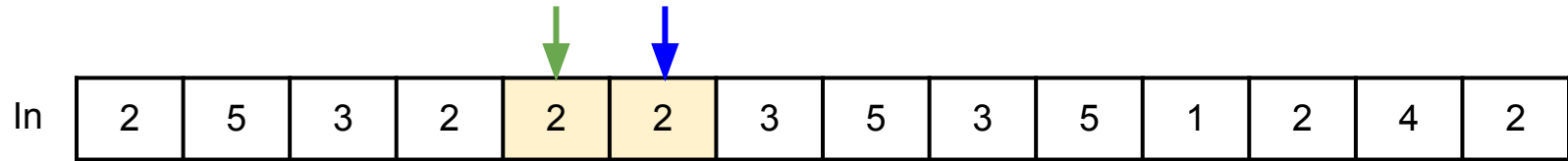
Optimization: Pipeline and Dependencies

- Observation: When no two consecutive data are with same value
 - `II = 1` is ok
 - In HLS, using `pragma` to specify `dependency is false` for “hist[]”

```
void histogram(int in[INPUT_SIZE], int hist[VALUE_SIZE]) {  
    // if no two consecutive data are with same value  
    #pragma HLS DEPENDENCE variable=hist inter false  
  
    int val;  
    for(int i = 0; i < INPUT_SIZE; i++) {  
        #pragma HLS PIPELINE  
        val = in[i];  
        hist[val] = hist[val] + 1;  
    }  
}
```

Optimization: Pipeline and Dependencies

- What about general case? Is Pipeline II=1 possible?
 - Yes, by using **register** to **reduce the # of accesses to "hist[]"**

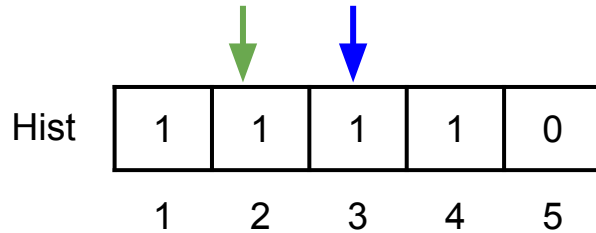
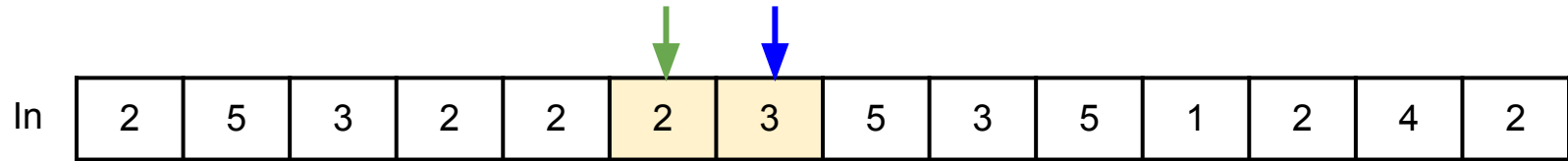


Pipeline, Task Interval = 1

	T1	T2	T3	T4	T5	T6
Read In[]	2	2				
Compare		True	True			
reg+=1			3	4		

Optimization: Pipeline and Dependencies

- What about general case? Is Pipeline II=1 possible?
 - For consecutive data with **different values**, waiting on write hist[] is not needed.



Pipeline, Task Interval = 1

	T1	T2	T3	T4	T5	T6
Read In[]	2	3				
Compare		True	False			
Read hist[]				hist[3]		
Write hist[]				hist[2]=4		

Intra-Iteration, dependency = false

Optimization: Pipeline and Dependencies

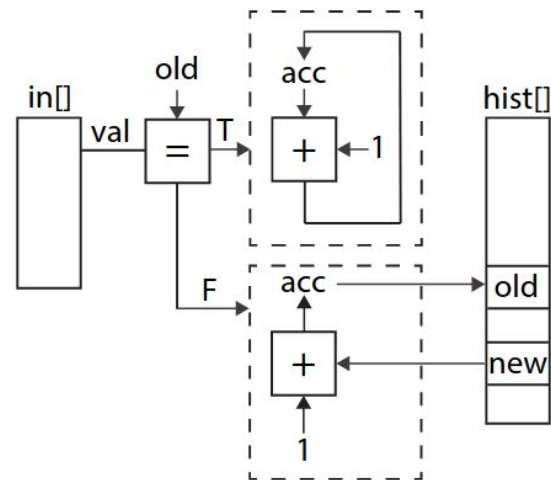
- What about general case? Is Pipeline II=1 possible?
 - Yes, using register to **reduce the # of BRAM accesses**
 - For consecutive values, adding count on register
 - Otherwise, refer to non-consecutive case

```
void histogram(int in[INPUT_SIZE], int hist[VALUE_SIZE]) {
    int acc = 0;
    int i, val;
    int old = in[0];
    #pragma HLS DEPENDENCE variable=hist intra RAW false

    for(i = 0; i < INPUT_SIZE; i++) {
        #pragma HLS PIPELINE II=1
        val = in[i];
        if(old == val) { // consecutive case
            acc = acc + 1;
        } else { // non-consecutive case,
            hist[old] = acc;
            acc = hist[val] + 1;
        }
        old = val;
    }
    hist[old] = acc;
}
```

Note that

- This optimization only works for ZYNQ Ultrascale+ FPGA
- Timing constraint missed for ZYNQ 7000



Optimization: More Parallelism by using MapReduce

- Partitioning the input and parallelly processing on each part
- Then merge the results from each partition.

