Accelerating 3D Convolution using Stream Computing on FPGAs

Haohuan Fu (Van)
haohuan@gmail.com
Postdoctoral Fellow, Stanford Center for Computational Earth & Environment Science (CEES)

SEP annual meeting, SEP-138, pp. 281-290, May, 2009

Outline

- Stream Computing on FPGAs
- Stream Computing:
  Advantages and Constraints
- A Streaming 3D Convolution Design
- Design Exploration and Performance Results

Outline

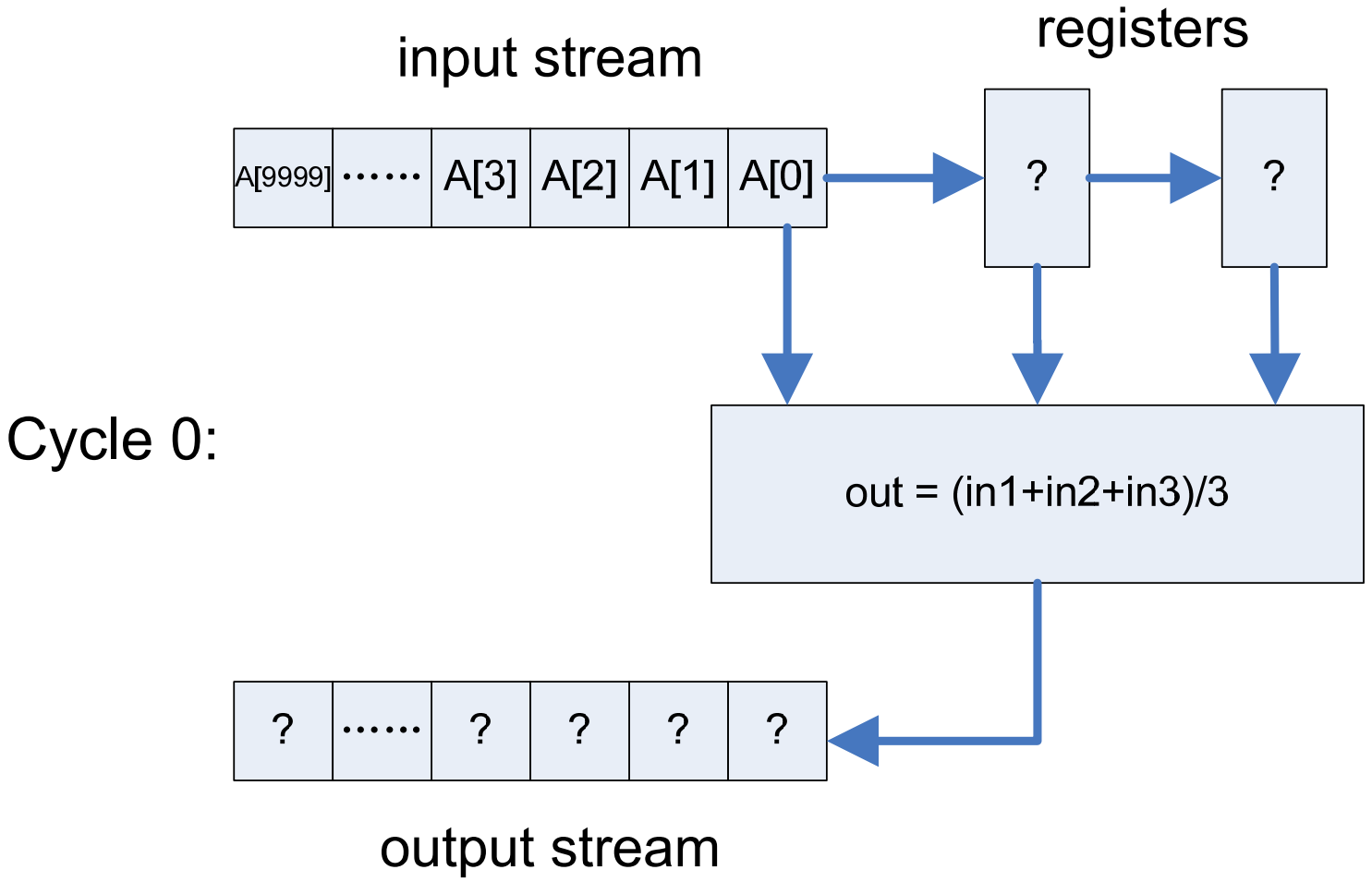
- **Stream Computing on FPGAs**
- Stream Computing:
  Advantages and Constraints
- A Streaming 3D Convolution Design
- Design Exploration and Performance Results

A Simple Example: 3-Point Convolution on 1D Array

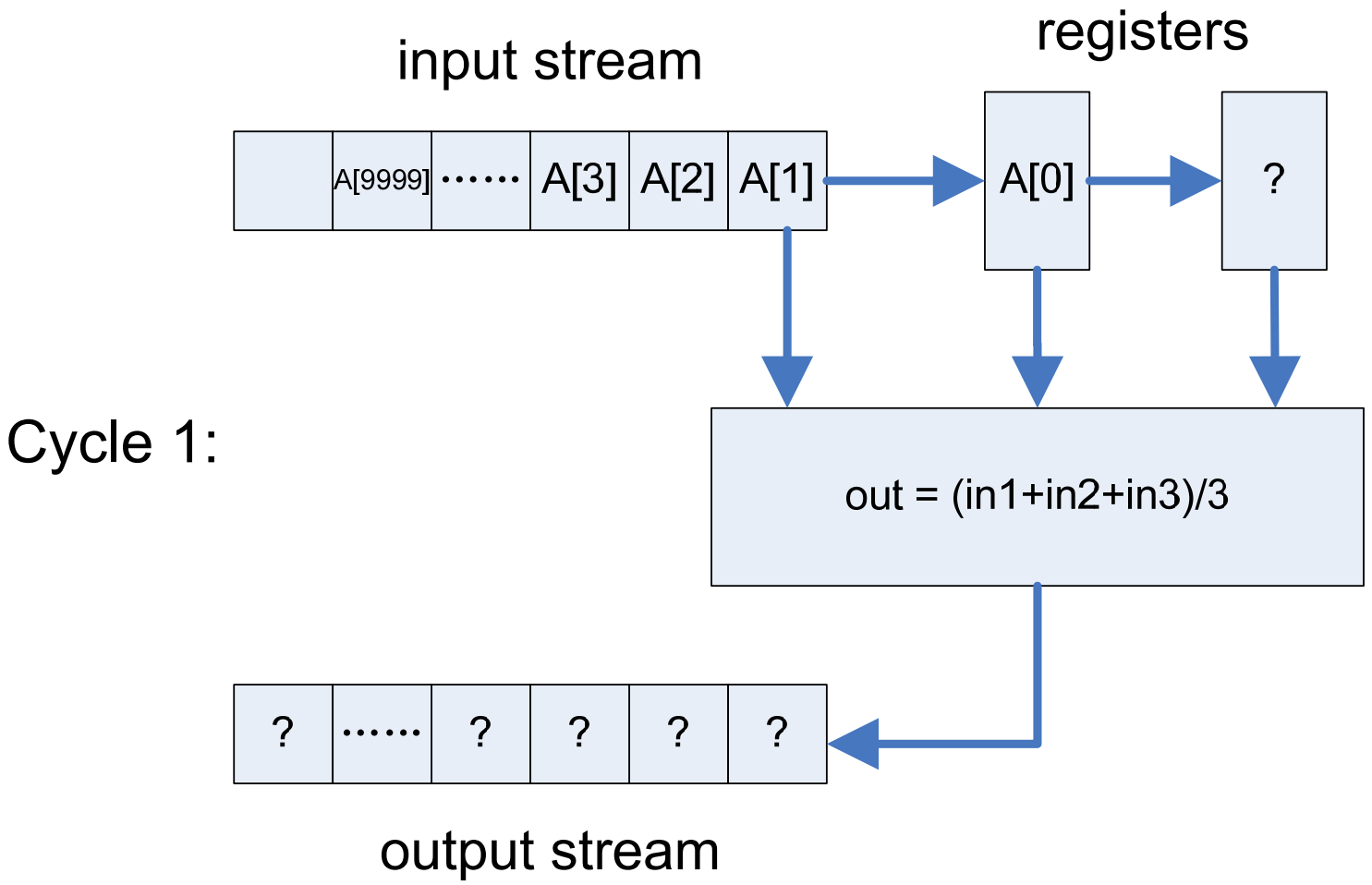
- input data: a[10000]
- output data: b[10000]
- computation:

```
for (int i = 1; i < 9999; i++){
    b[i] = (a[i-1]+a[i]+a[i+1])/3;
}
```

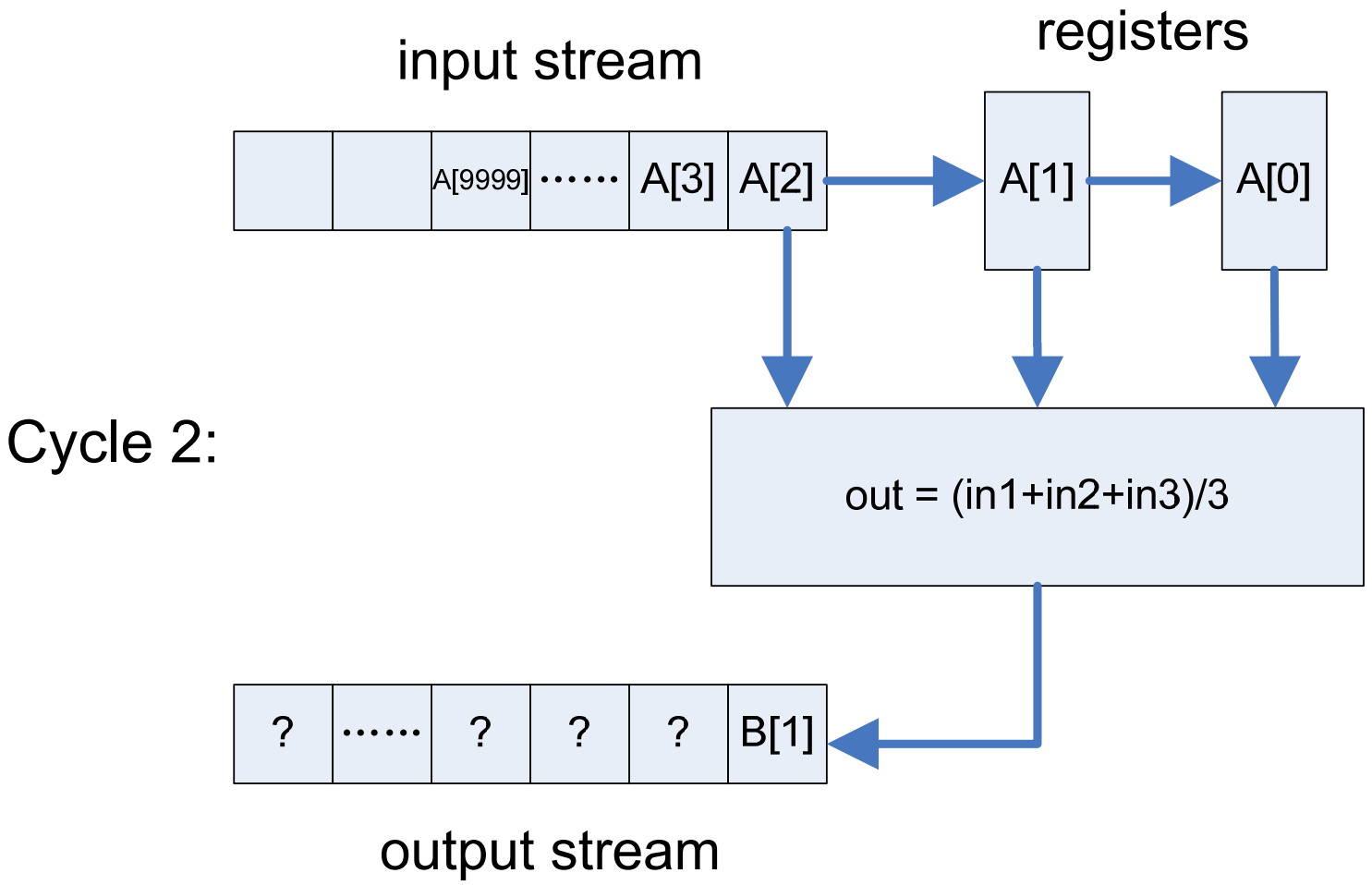
Compute in a Streaming Approach



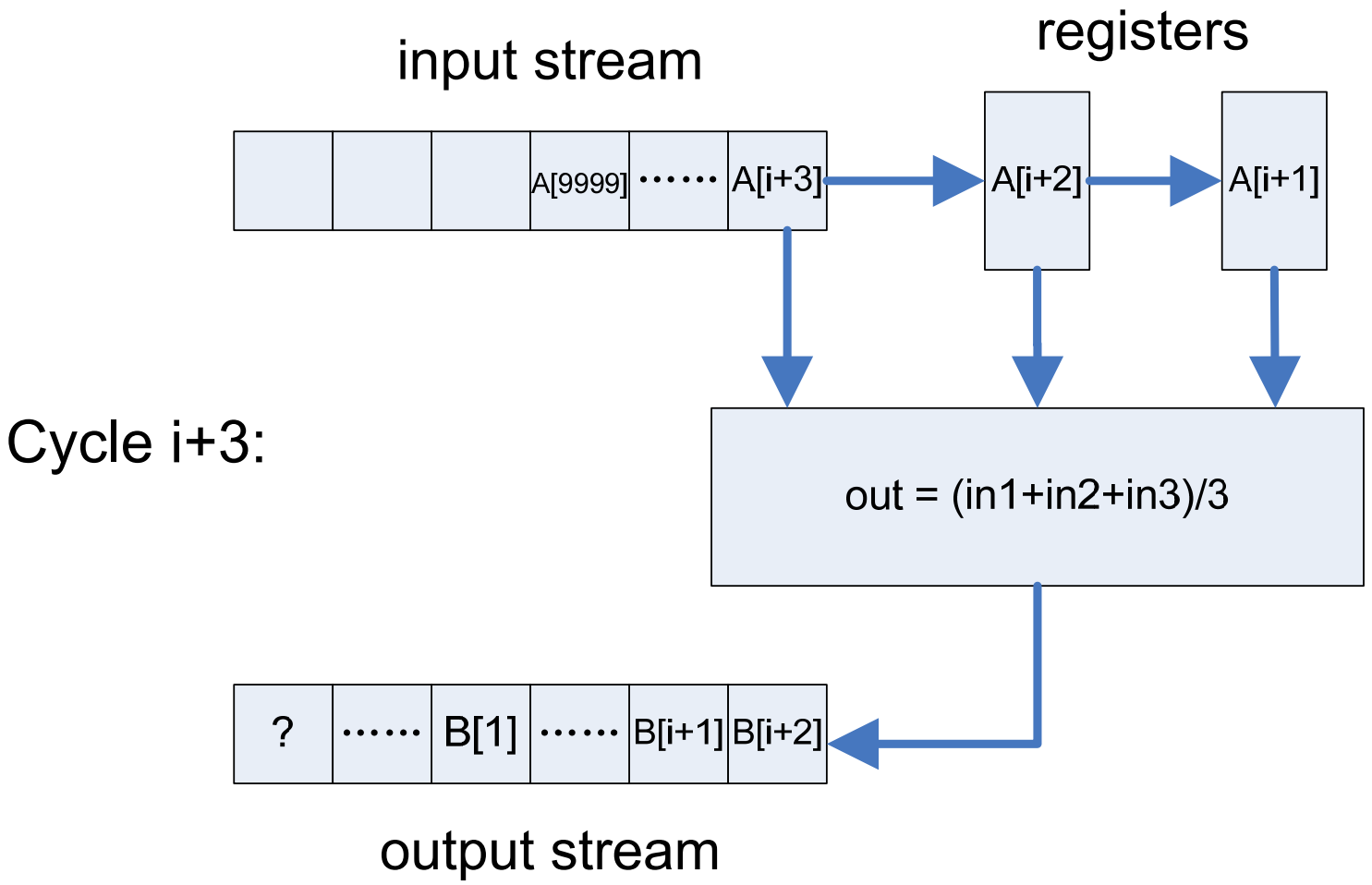
Compute in a Streaming Approach



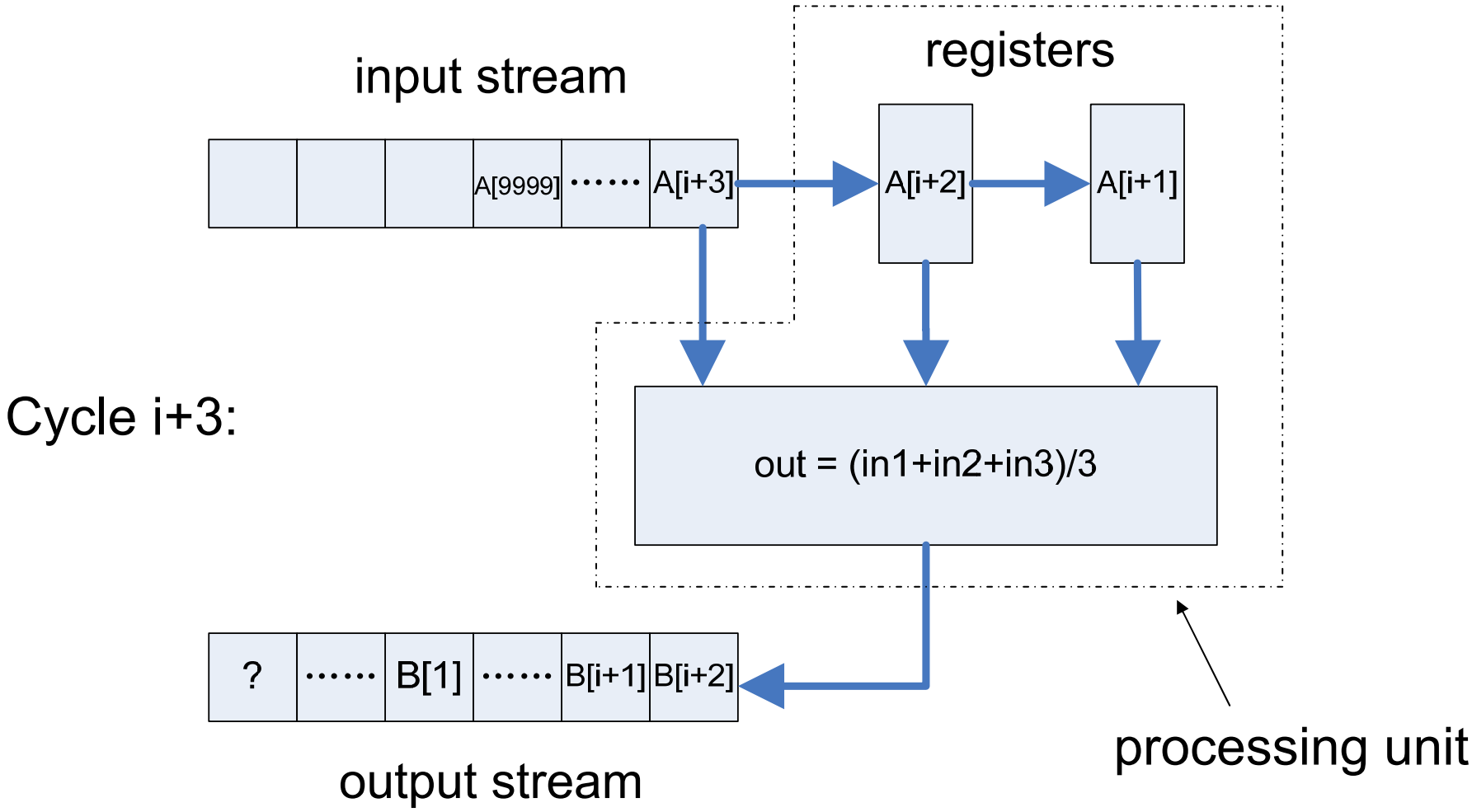
Compute in a Streaming Approach



Compute in a Streaming Approach

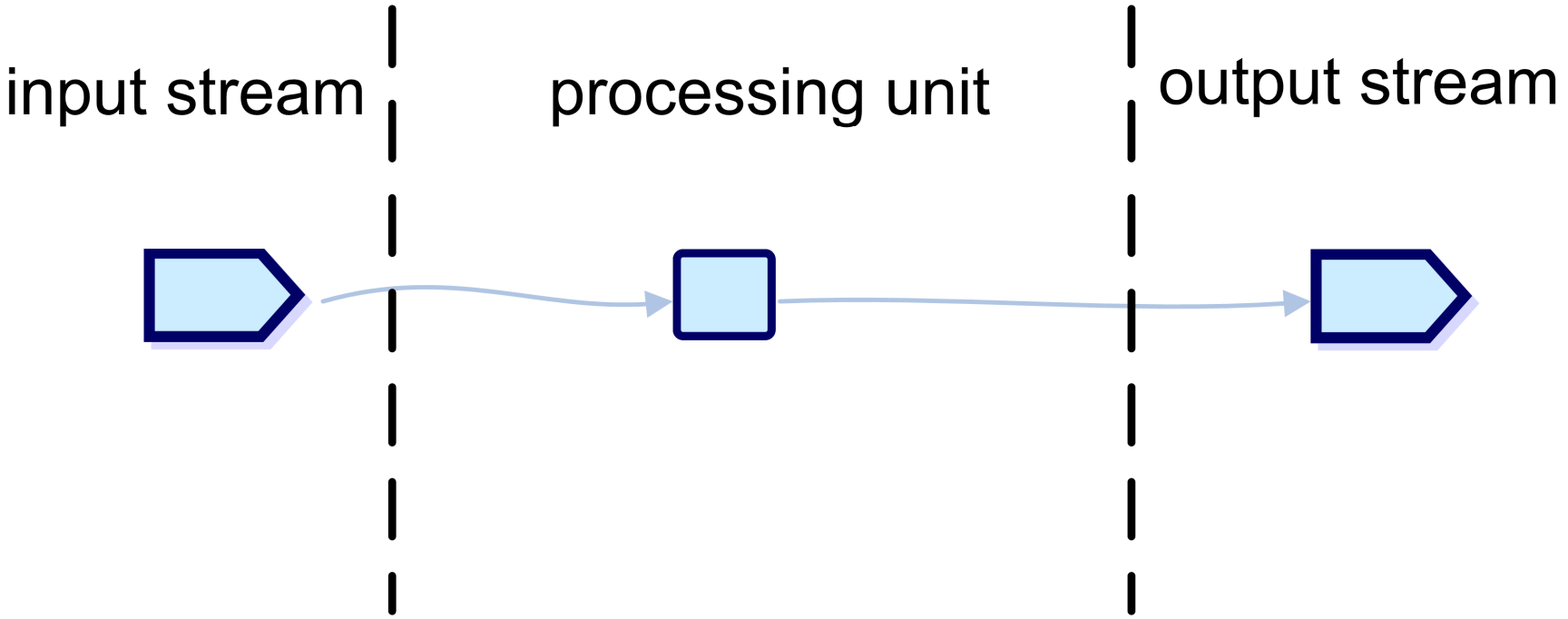


Compute in a Streaming Approach

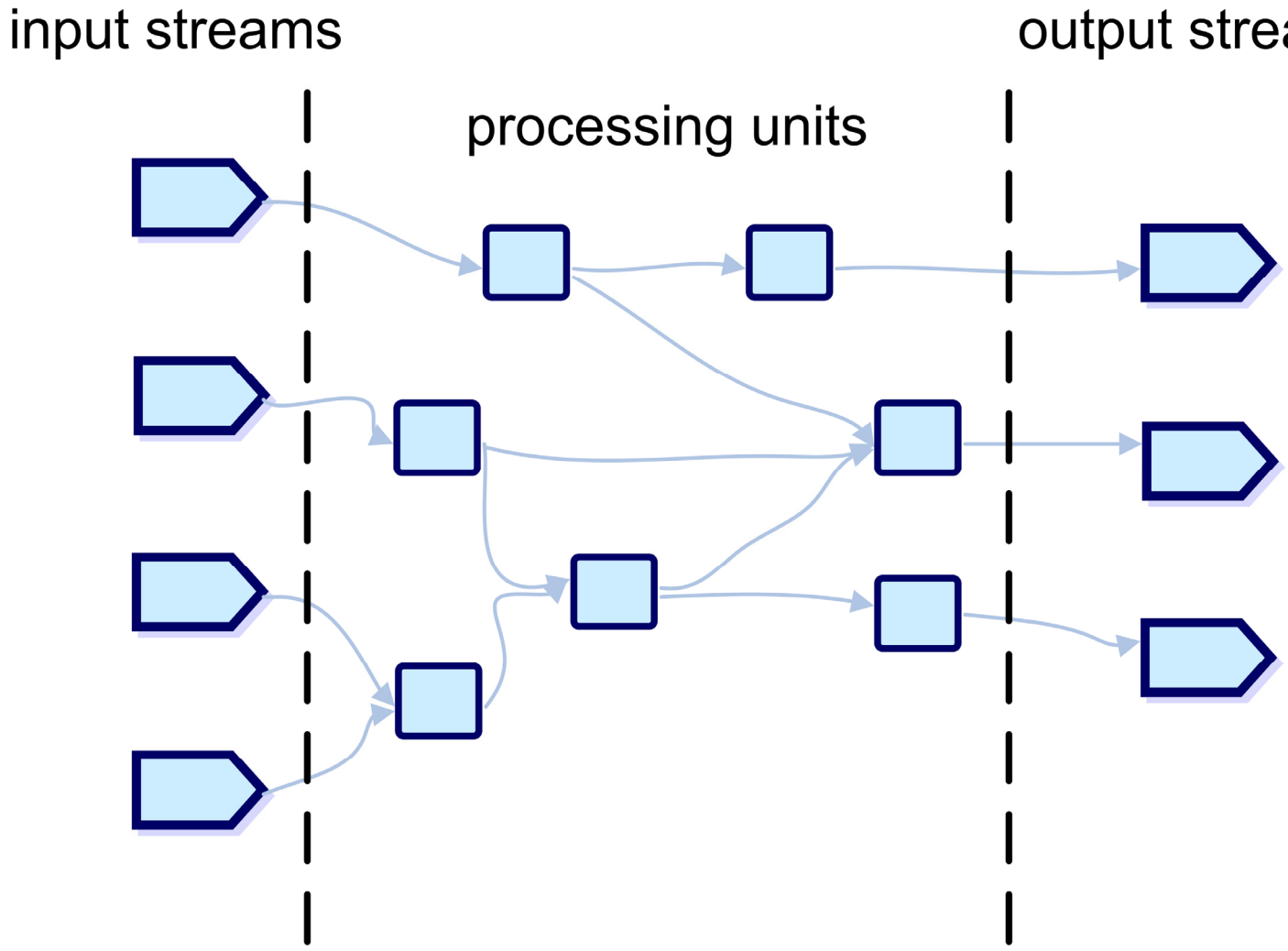


Stream Computing: A Simple Case

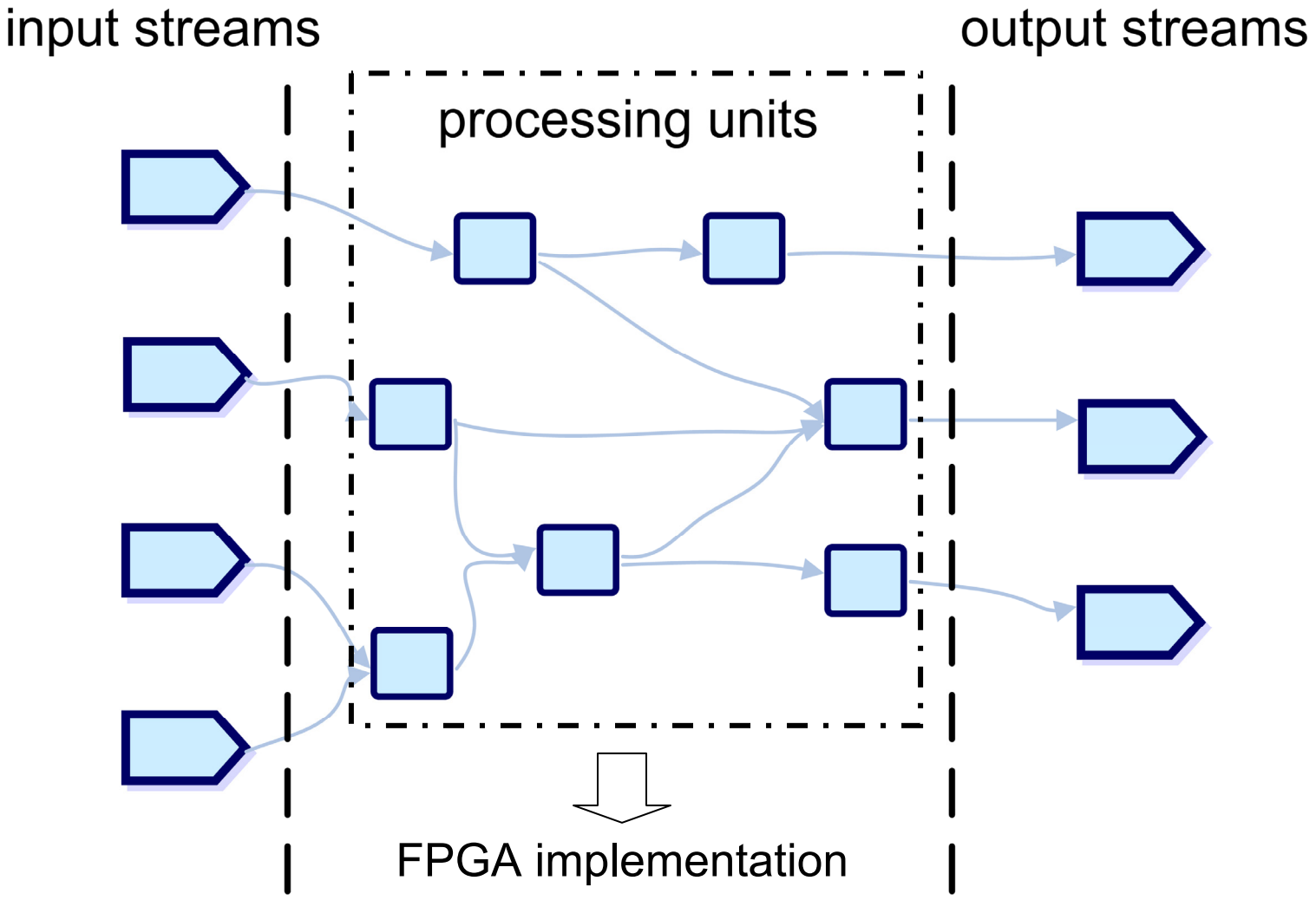
- The computation is performed as the data streams flow through the corresponding units



Stream Computing: A General Picture



Stream Computing on FPGAs



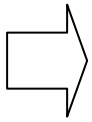
Field Programmable Gate Arrays (FPGAs)

- CMOS technology based hardware chips
- Contain reconfigurable computation units, memory units, and interconnections
- Provide application-specific acceleration solutions

Computation Capacity of FPGAs

- Resource summary of FPGAs

FPGAs	#LUTs	#FFs	#DSP48Es	#BRAMs
LX330T	207,360	207,360	196	324
SX475T	287,600	595,200	2,016	1,064



A large FPGA:

- 1000 multipliers
- 500 adders

- Resource cost for single-precision floating-point arithmetic units

Operations	#LUTs	#FFs	#DSP48Es	#BRAMs
+/−	425	557	0	0
×	122	173	2	0

Memory Bandwidth of FPGAs

- Maxeler FPGA acceleration card[1]
  - two FPGAs on one PCI Express card
  - up to 24 GB onboard memory
- 'External' memory bandwidth
  - 4 GB/s PCI Express between host and FPGA
  - 13 GB/s memory bandwidth between FPGA and the onboard memory
- 'Internal' memory bandwidth
  - arrays of Block RAMs with multiple I/O ports
  - around 4 MB storage with up to 1 TB/s bandwidth
  - current multi-core CPUs: each core has 64 KB L1 cache, around 96 GB/s bandwidth

[1] http://www.maxeler.com

Outline

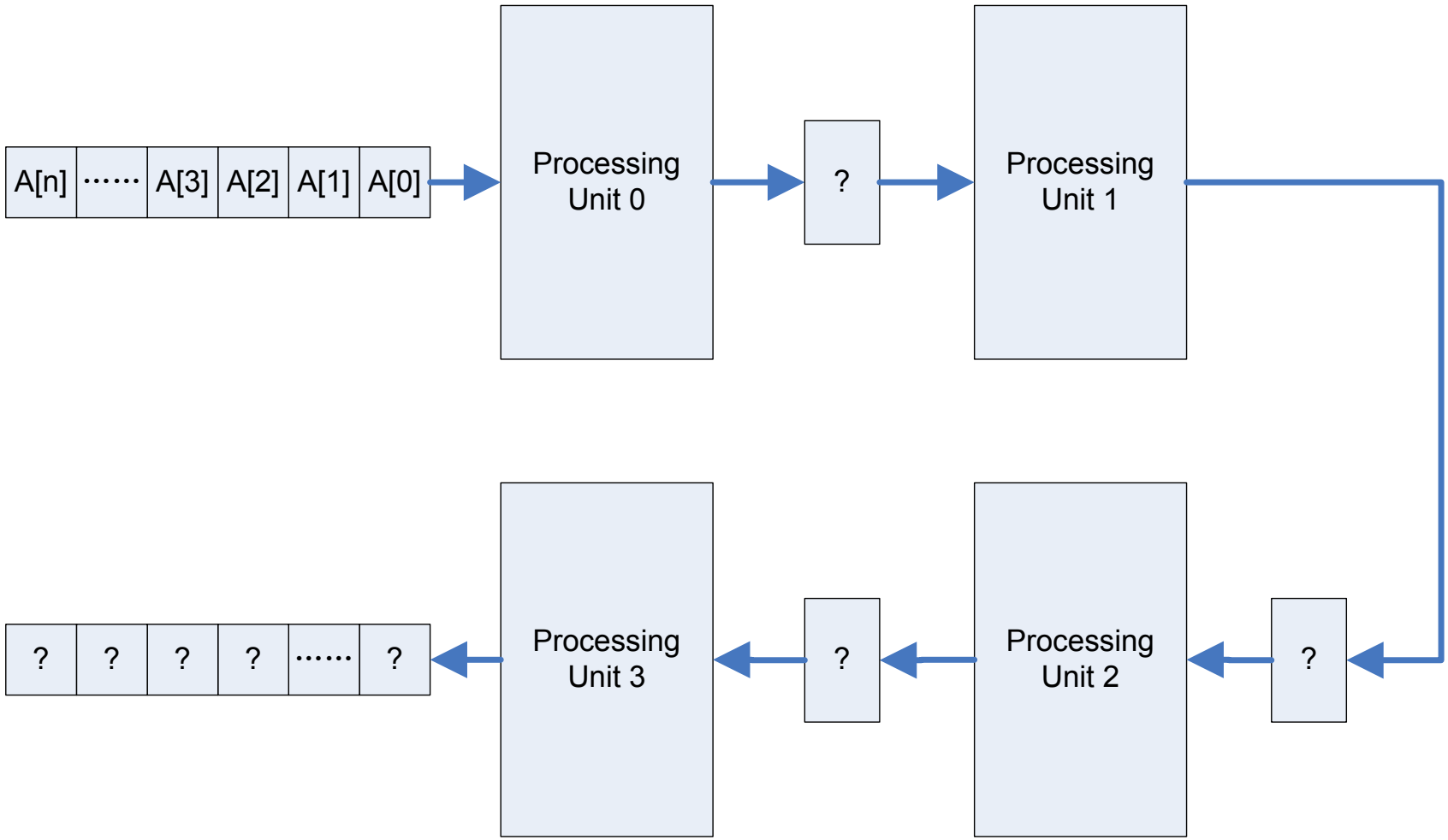
- Stream Computing on FPGAs
- **Stream Computing: Advantages and Constraints**
- A Streaming 3D Convolution Design
- Design Exploration and Performance Results

Stream Computing on FPGAs: Advantages

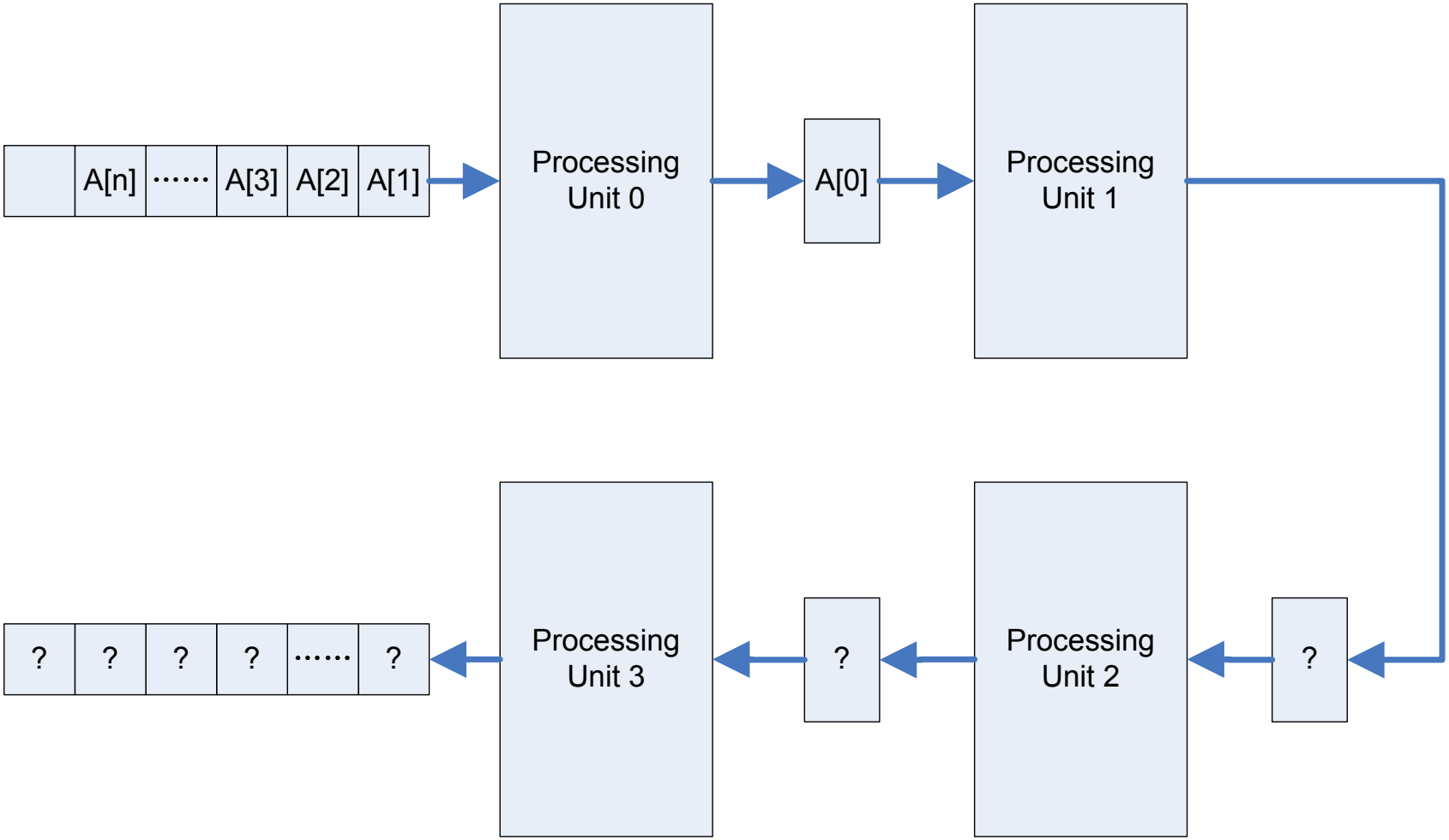
- Compared to the standard Single Instruction Multiple Data (SIMD) parallel computing, stream computing provides Multiple Instruction Multiple Data (MIMD)

- Each processing unit performs different operations on different data items concurrently

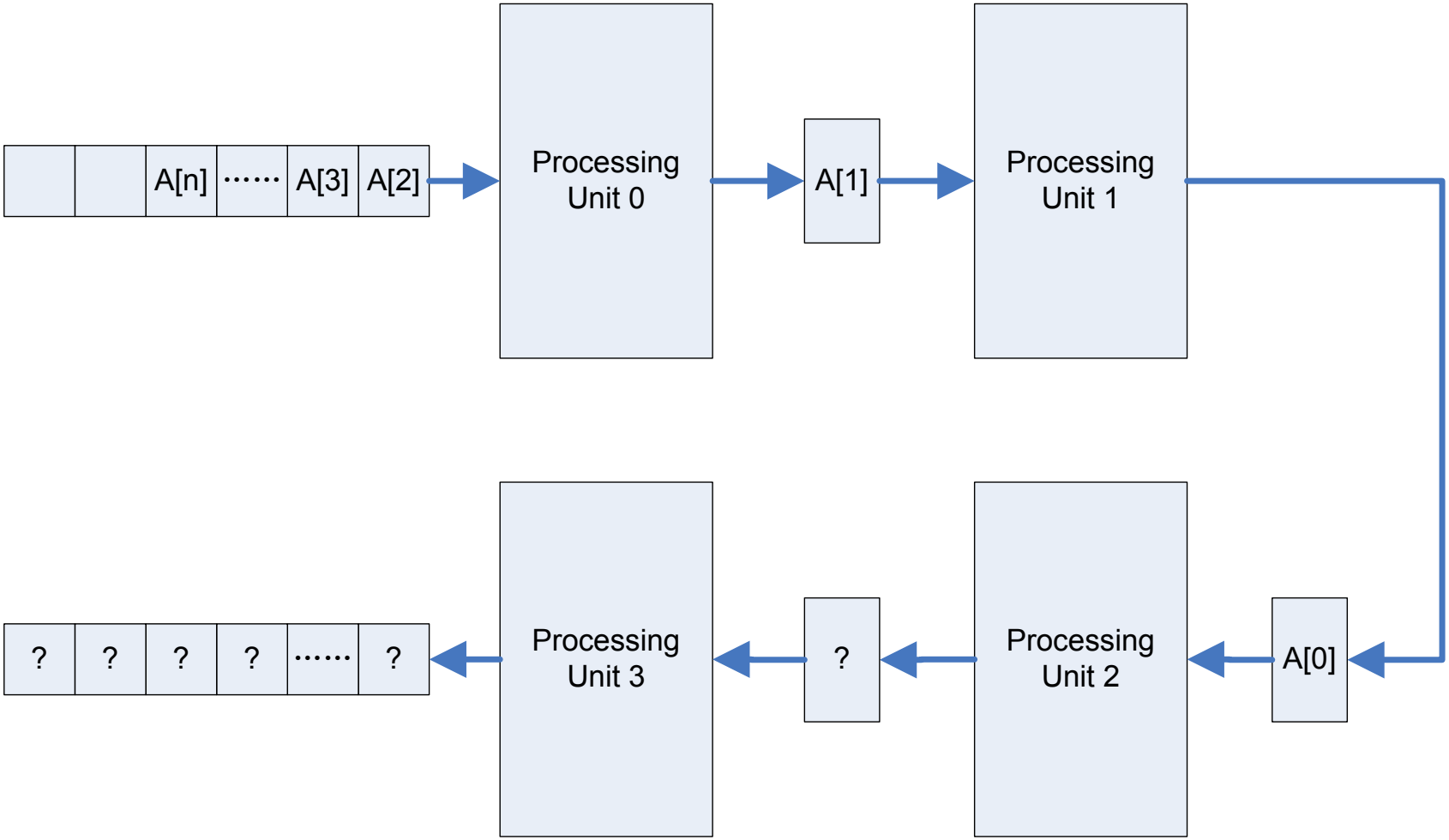
Stream Computing: Pipelining: Cycle 0



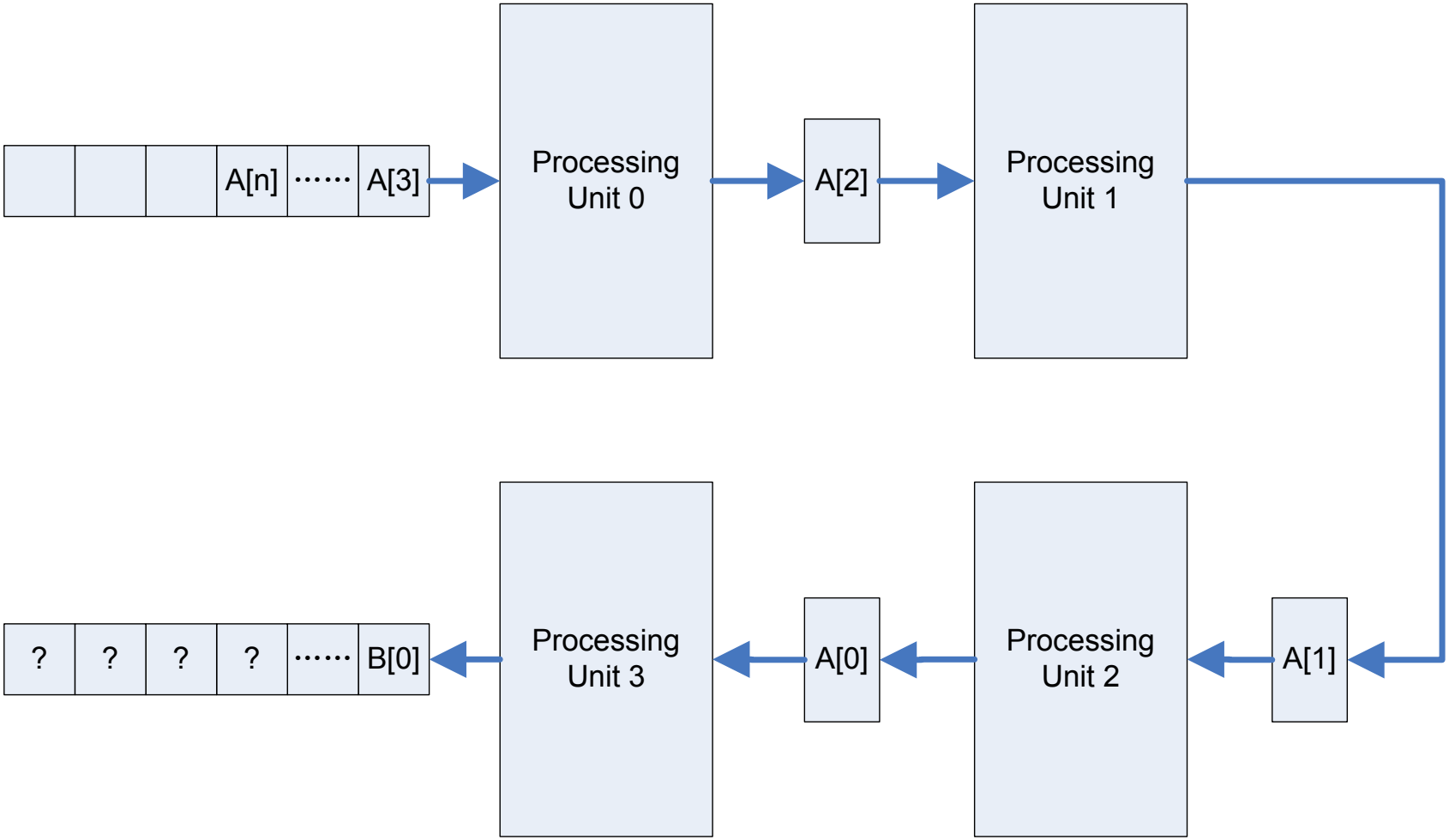
Stream Computing: Pipelining: Cycle 1



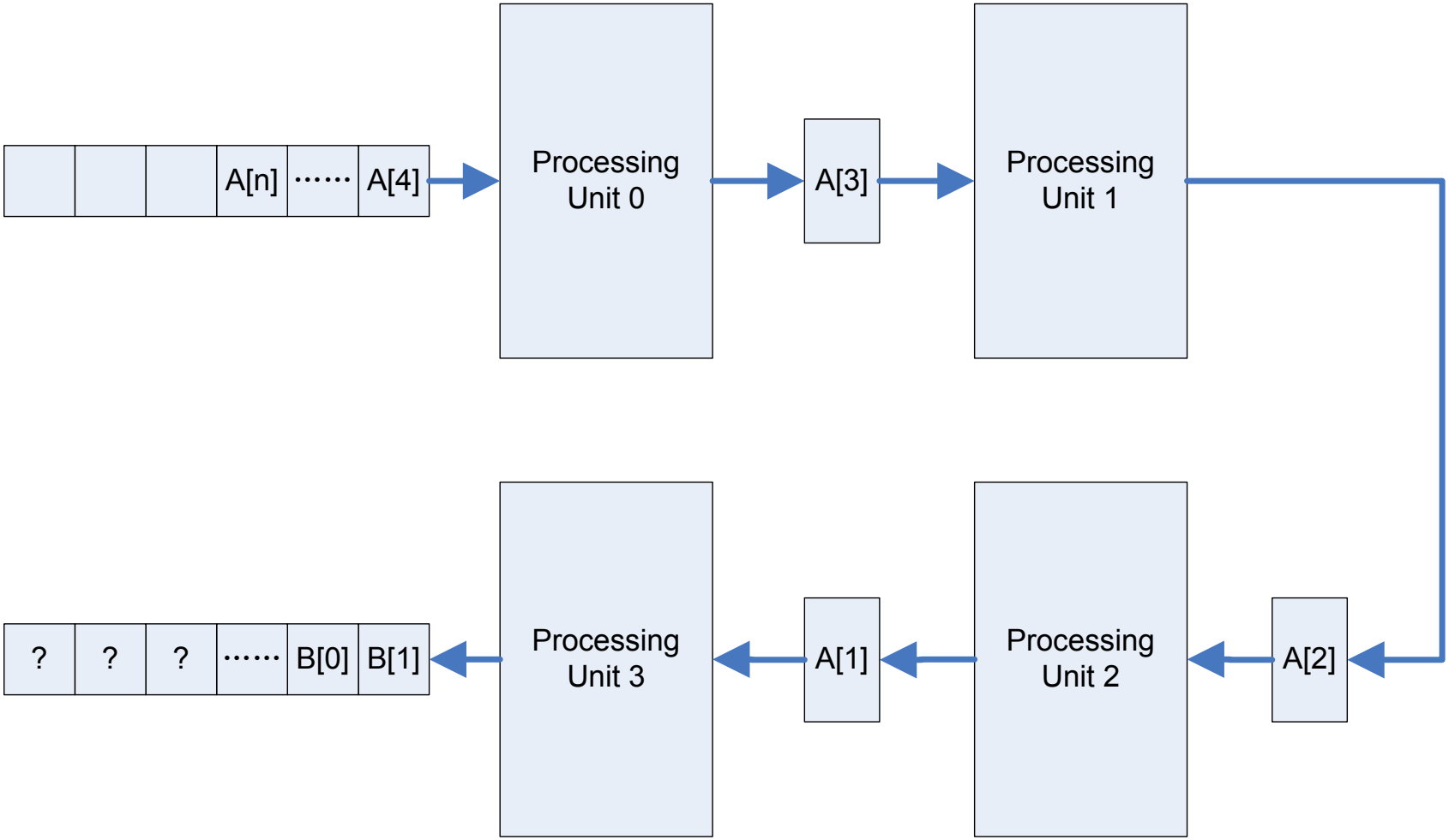
Stream Computing: Pipelining: Cycle 2



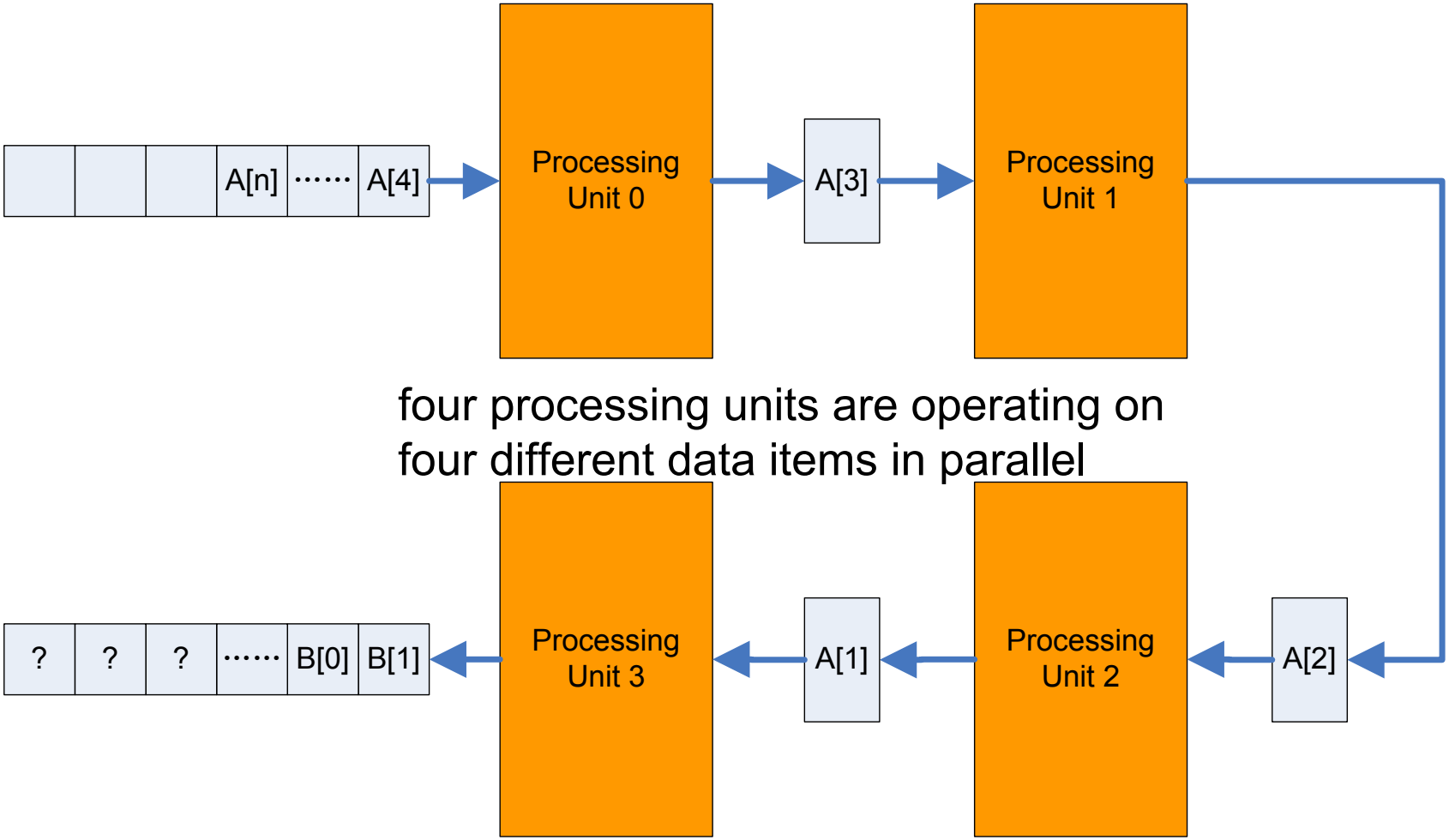
Stream Computing: Pipelining: Cycle 3



Stream Computing: Pipelining: Cycle 4



Stream Computing: Pipelining: Cycle 4



Stream Computing on FPGAs: Advantages

- Scalability of the Computation Power
  - adding more processing units between input and output only consumes more hardware resources
  - the latency from input to output is increased, but with a large amount of data streaming through, the throughput of the circuit remains the same
  - the computation power scales almost linearly with the hardware resource capacity of the system

Stream Computing on FPGAs: Advantages

- Data locality
  - at each cycle, the input item always sits next to the corresponding processing unit
  - each processing unit takes the output of the previous processing unit as input, which minimizes the memory bandwidth requirement

Stream Computing on FPGAs: Constraints

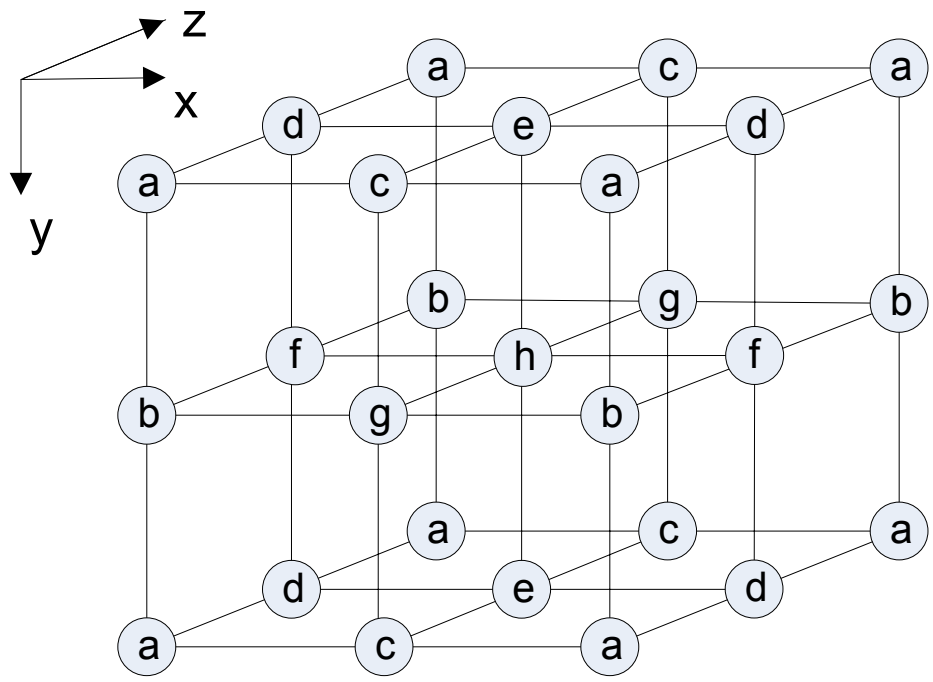
- Less flexible than the standard CPU computing:
  - same or similar computation on all the data items
  - a sequential access pattern of the data items
  - very suitable for feed-forward streaming computations, not suitable for feedbacks
- Much more difficult to program

Outline

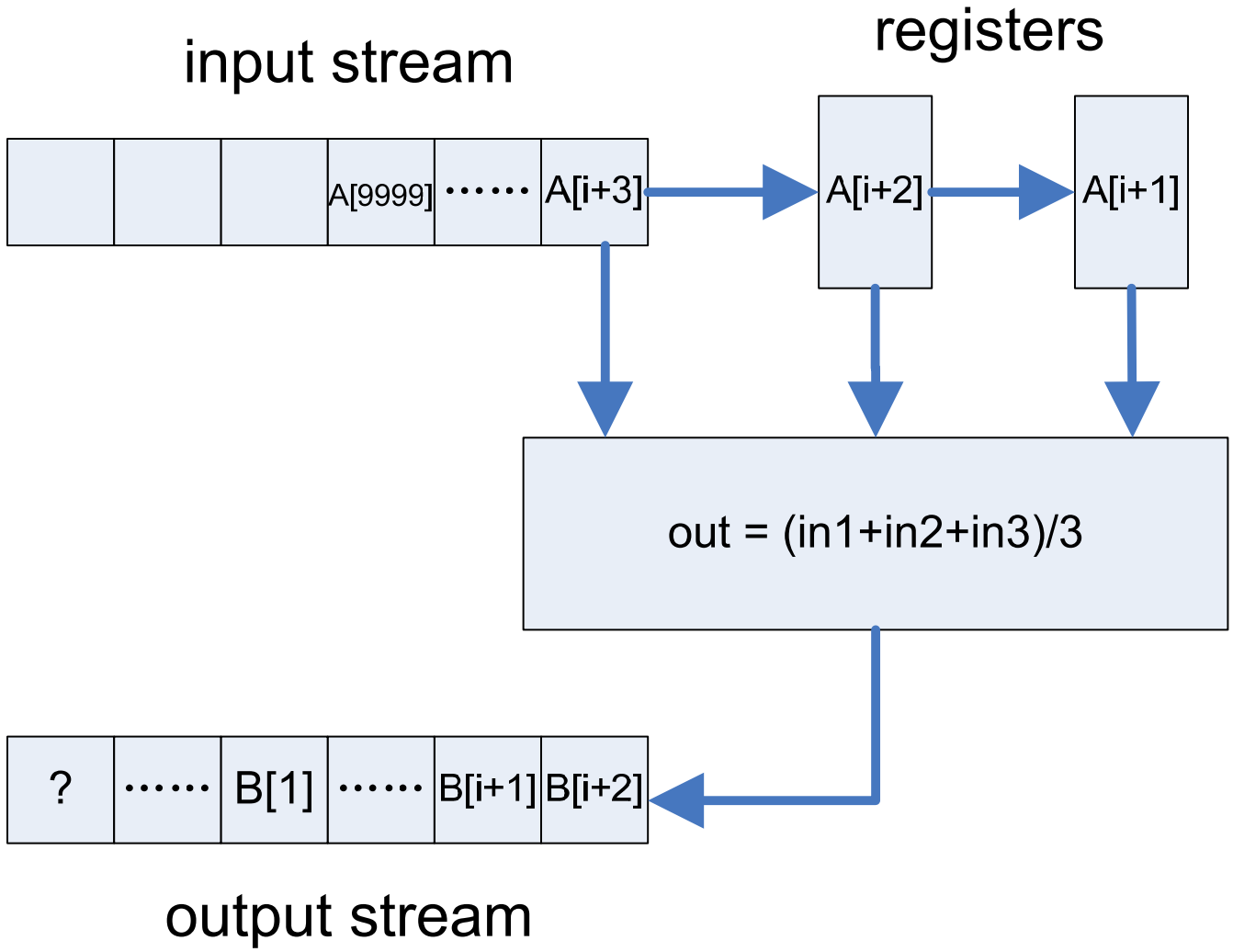
- Stream Computing on FPGAs
- Stream Computing:
  Advantages and Constraints
- **A Streaming 3D Convolution Design**
- Design Exploration and Performance Results

3D Convolution using a Cube Filter

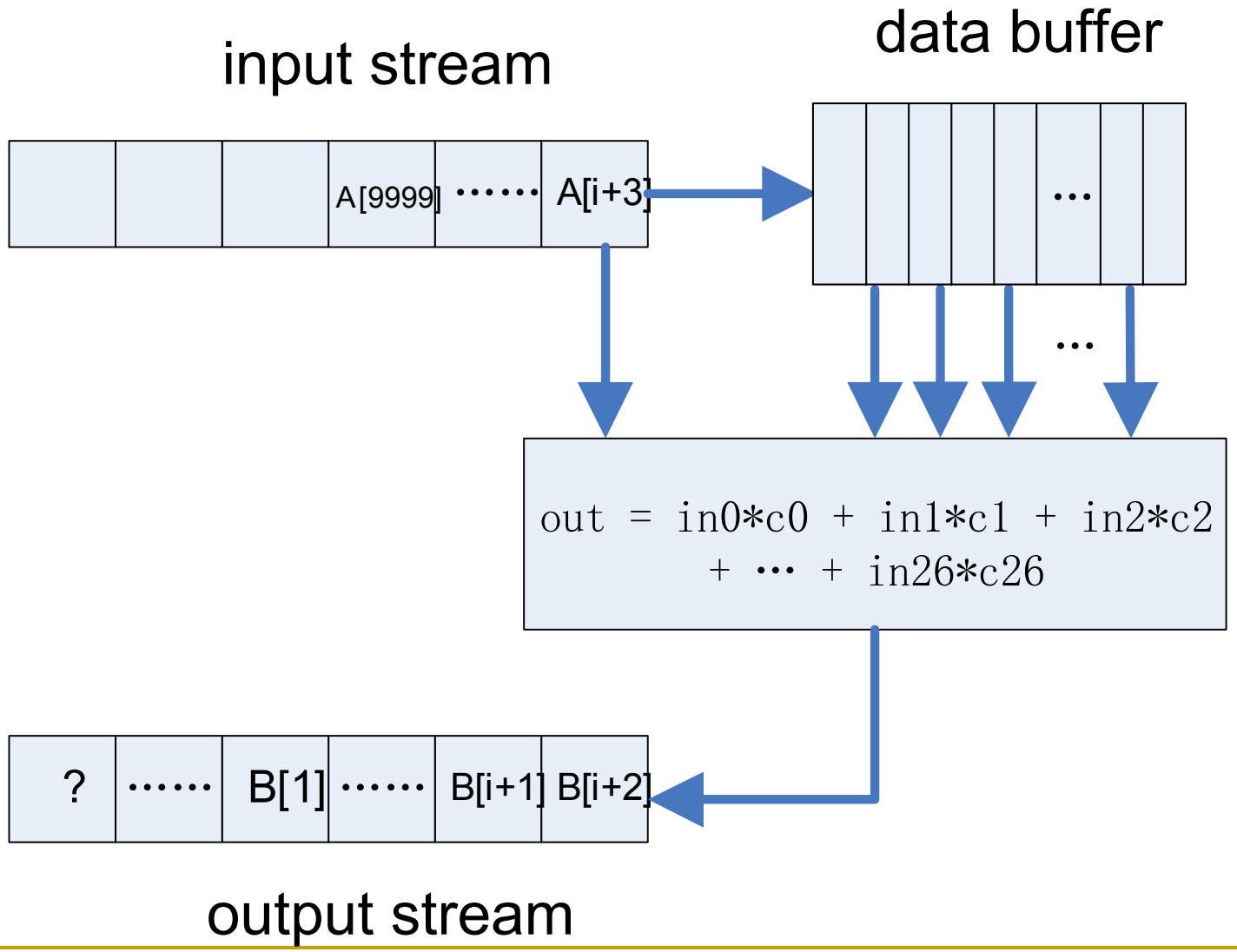
- 512x512x512 3D convolution,
  using a symmetric stencil (6th order in space)
  - 8 multiplications
  - 26 additions



3-Point 1D Convolution



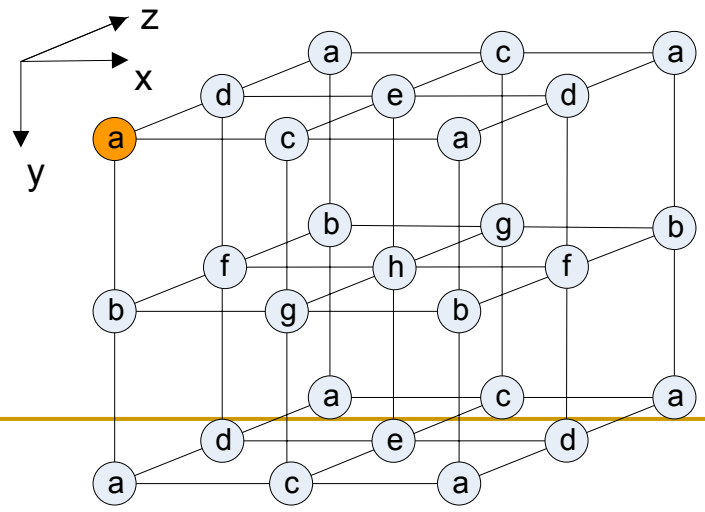
27-Point 3D Convolution



Buffering of Input Stream

- The convolution operator requires a wide range of data items in the input stream

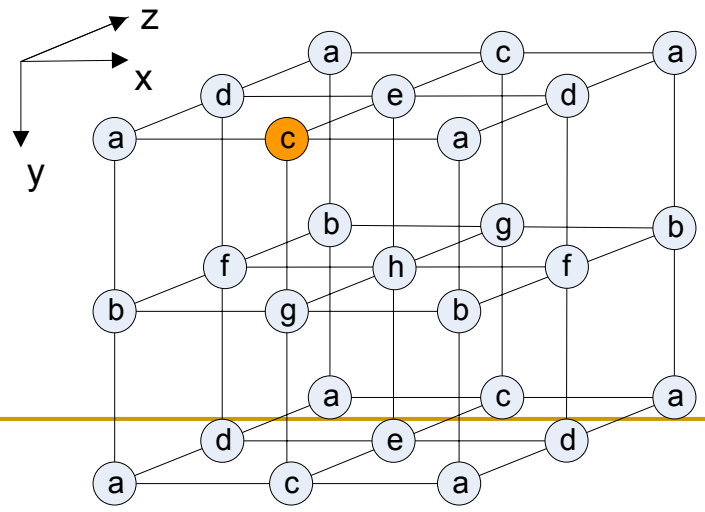
A[i]	A[i+1]	A[i+2]	.....	A[i+512]	.....	A[i+512*512]	.....



Buffering of Input Stream

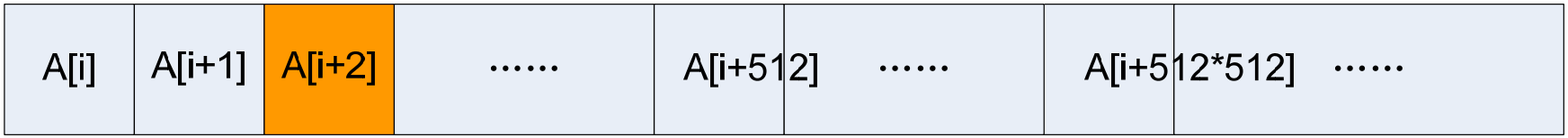
- The convolution operator requires a wide range of data items in the input stream

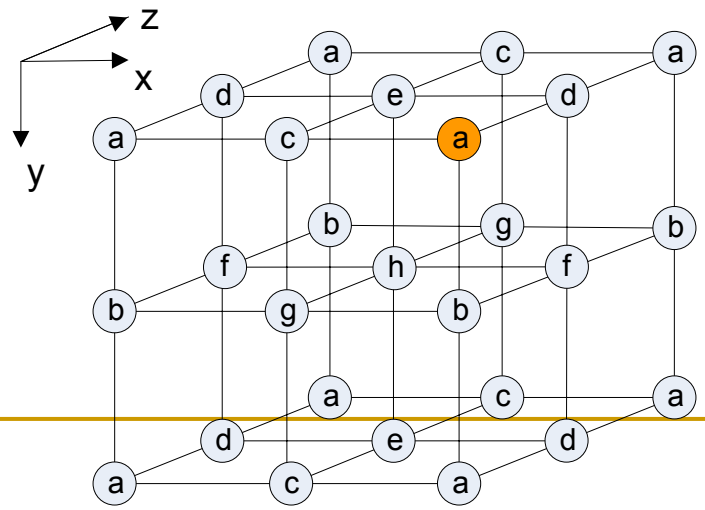
A[i]	A[i+1]	A[i+2]	.....	A[i+512]	.....	A[i+512*512]	.....



Buffering of Input Stream

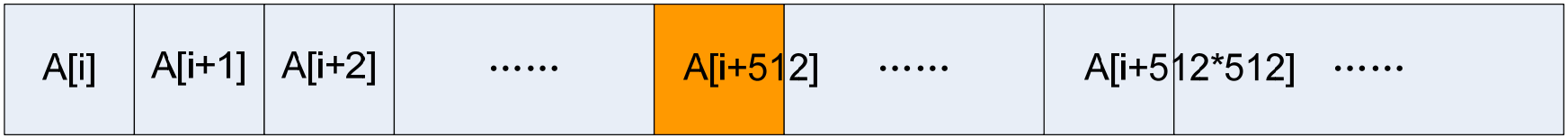
- The convolution operator requires a wide range of data items in the input stream

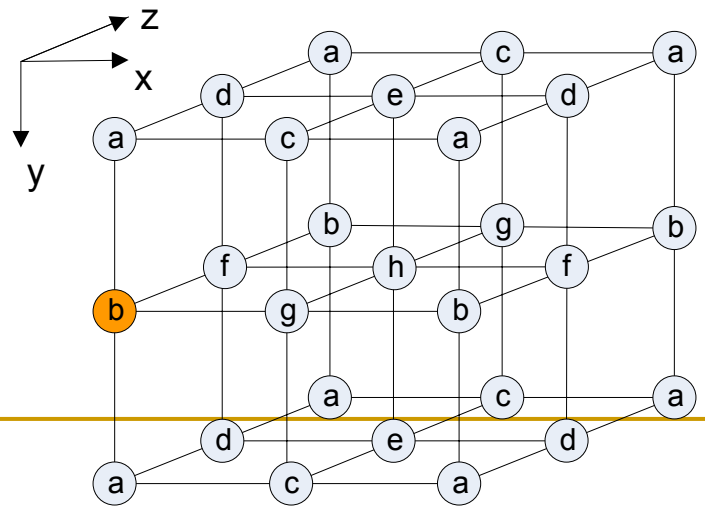




Buffering of Input Stream

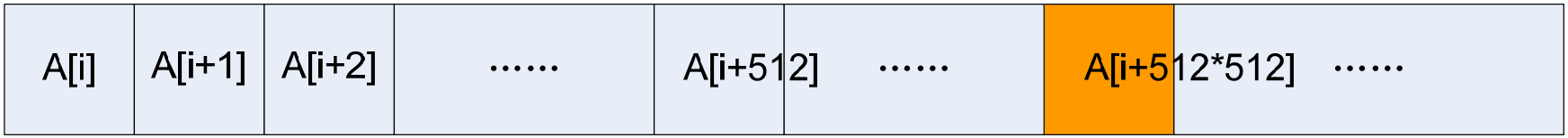
- The convolution operator requires a wide range of data items in the input stream

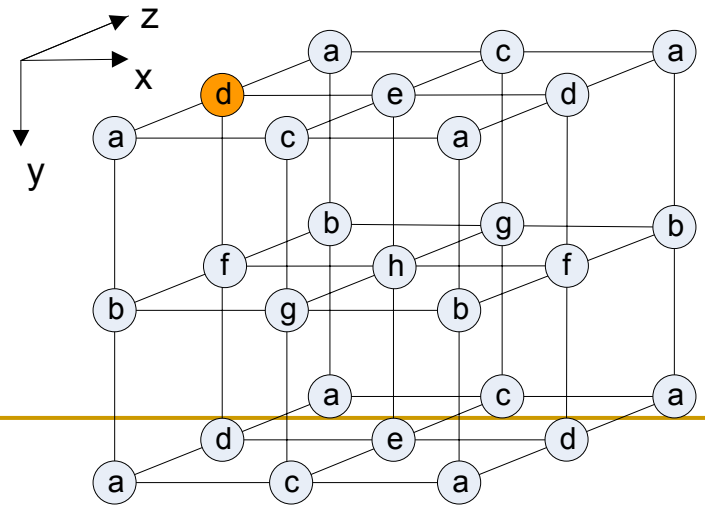




Buffering of Input Stream

- The convolution operator requires a wide range of data items in the input stream

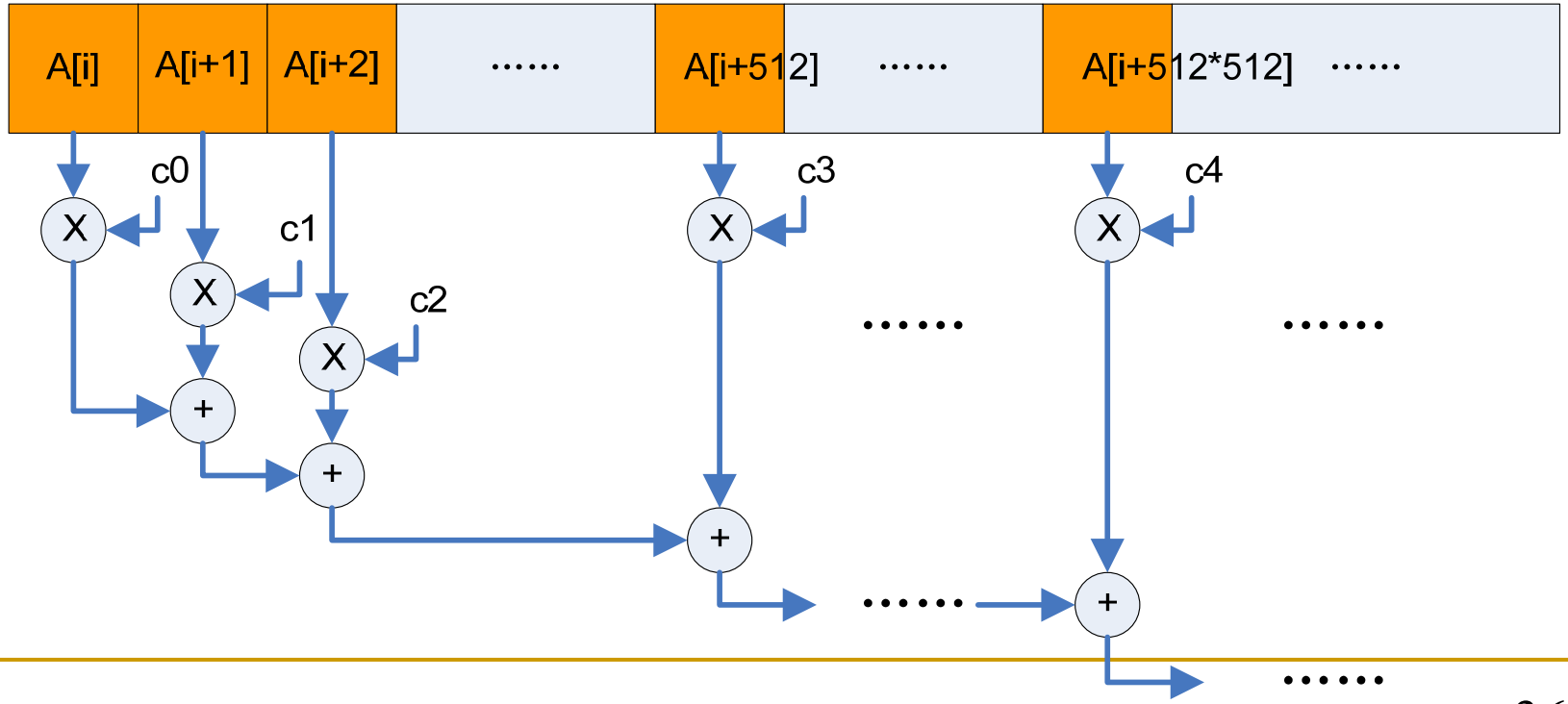




buffer size:
around 2*nx*ny

Buffering of Input Stream

- By buffering the entire range of all the points, we can read all the values concurrently and perform the computation in parallel



Buffering of Input Stream

- Therefore, to apply a 3x3x3 cube operator on a 512x512x512 data array, we need to buffer 2*512*512+2*512+3 = 525315 data items
- Using single-precision floating point, the required buffer size (wave field and velocity) can be over 4 MB, which is beyond the capacity of most current FPGAs
- To fit the problem into FPGA, we do a 2D blocking of the 3D array

Cost of 2D Blocking

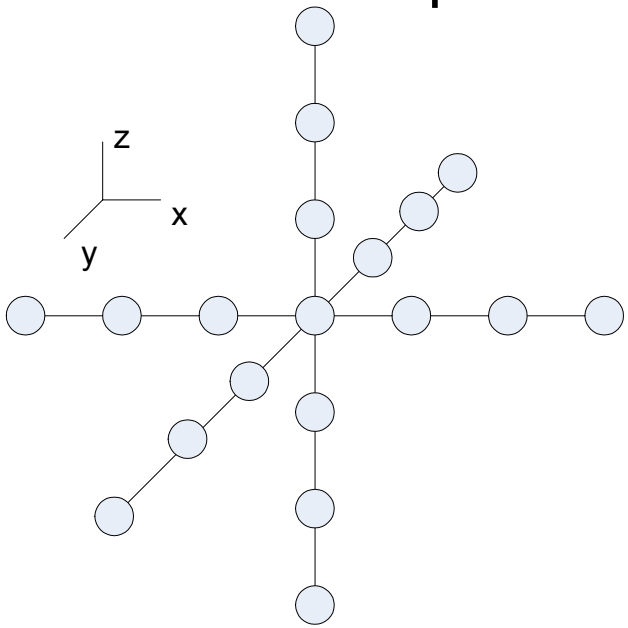
- Redundant processing of overlapping boundaries
  - for a stencil with *ns* non-zero lags in each direction, we compute a *(nx+ns)*(ny+ns)*nz* block to get a *nx*ny*nz* result block
  - extra cost brought by 2D blocking:
    (nx+ns)(ny+ns) / nx*ny – 1*
  - *E.g.: (100+5)*(100+5)/100*100 – 1 = 10.3%*

Outline

- Stream computing on FPGAs
- Stream computing:
  advantages and constraints
- A Streaming 3D Convolution Design
- Design Exploration and Performance Results

Stencil Shape: ‘star’ OR ‘cube’

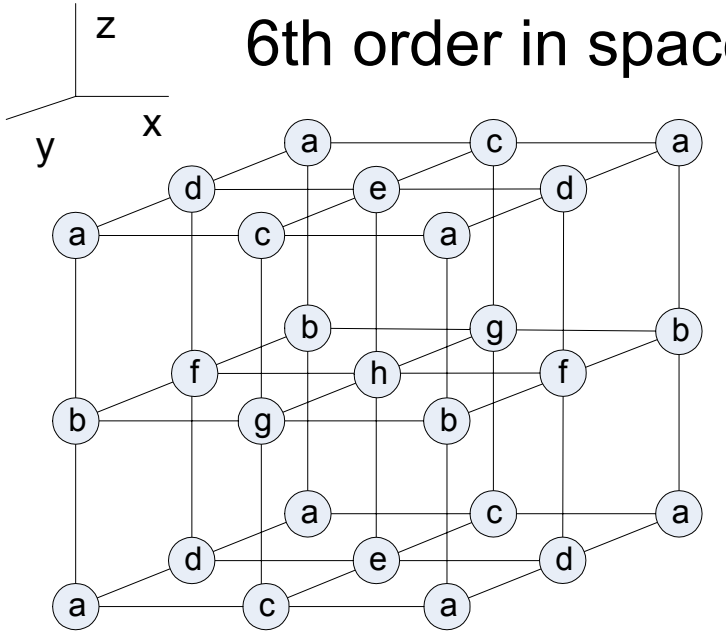
8th order in space



Computation: 9 MUL, 18 ADD

Stream buffer: 6 slices

6th order in space



Computation: 8 MUL, 26 ADD

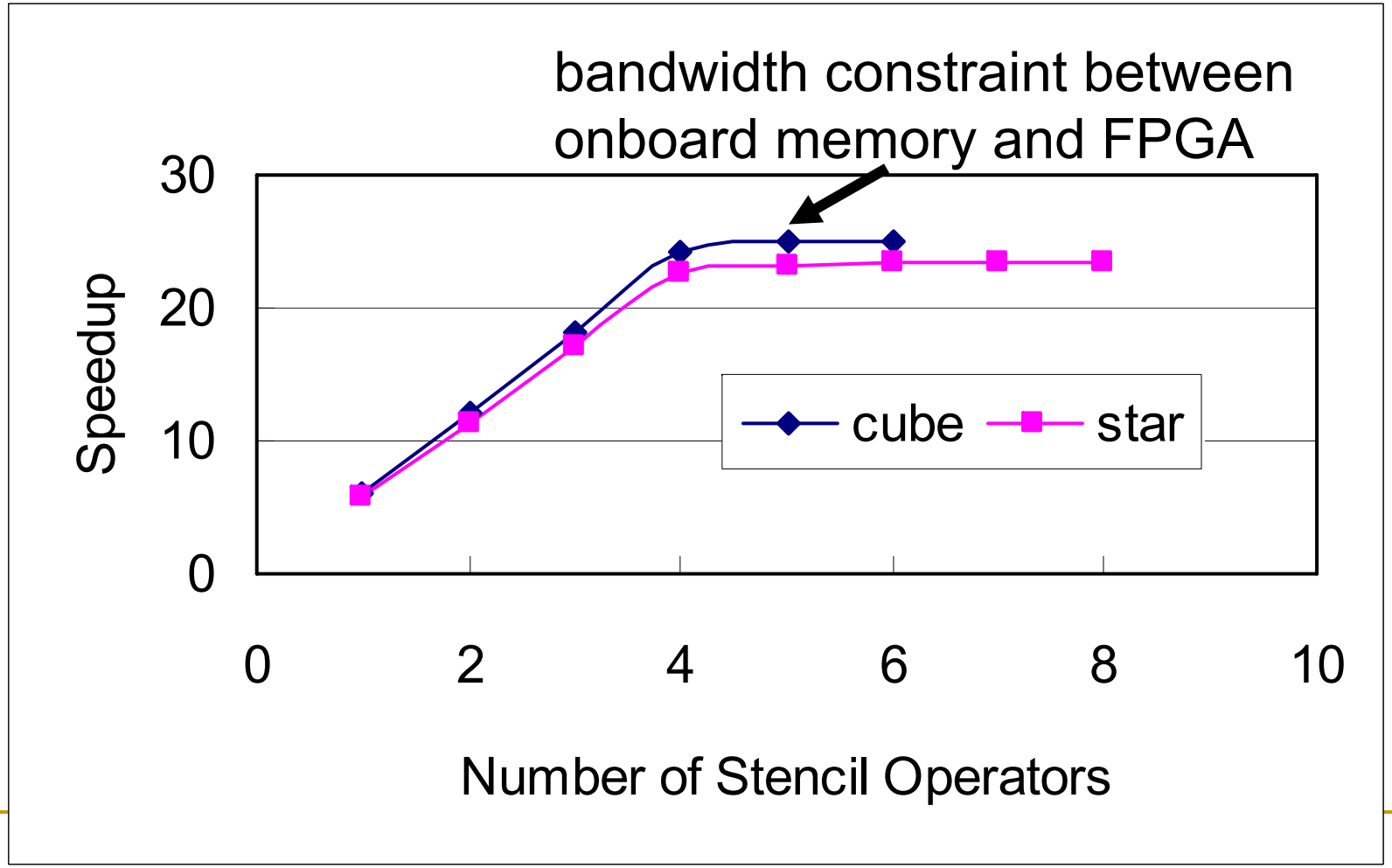
Stream buffer: 2 slices

Different Options of Parallel Computing

- Putting multiple stencil operators
  - b[1] = conv(a[1]); b[2] = conv(a[2]);
    b[3] = conv(a[3]); b[4] = conv(a[4]);

- Processing multiple time steps in one pass
  - b = conv_step1(a); c = conv_step2(b);
    d = conv_step3(c); e = conv_step4(d);

Multiple Stencil Operator: Speedup over a Single-Core Implementation

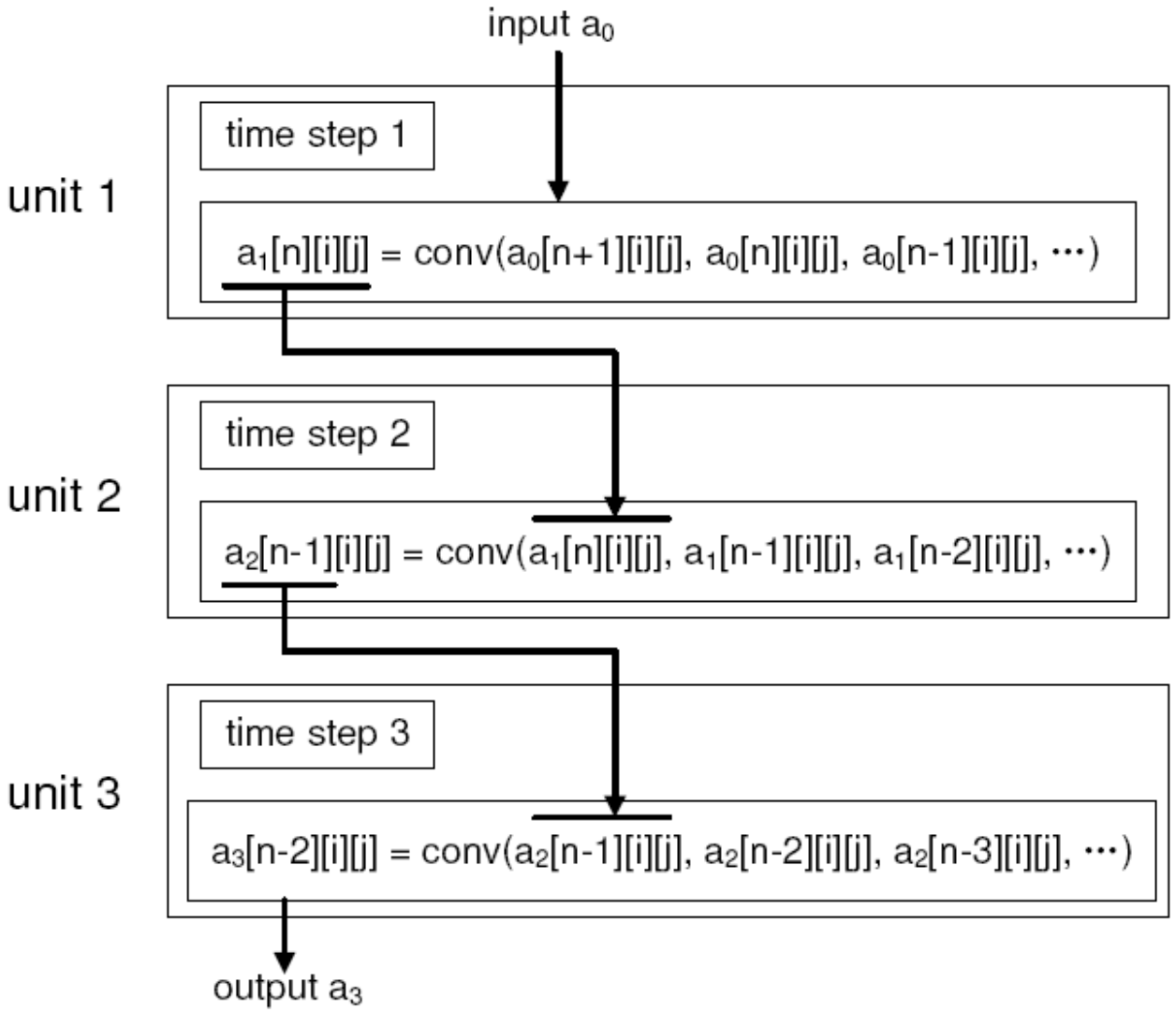


Different Options of Parallel Computing

- Putting multiple stencil operators
  - b[1] = conv(a[1]); b[2] = conv(a[2]);
    b[3] = conv(a[3]); b[4] = conv(a[4]);

- Processing multiple time steps in one pass
  - b = conv_step1(a); c = conv_step2(b);
    d = conv_step3(c); e = conv_step4(d);

Processing Multiple Time Steps in One Pass

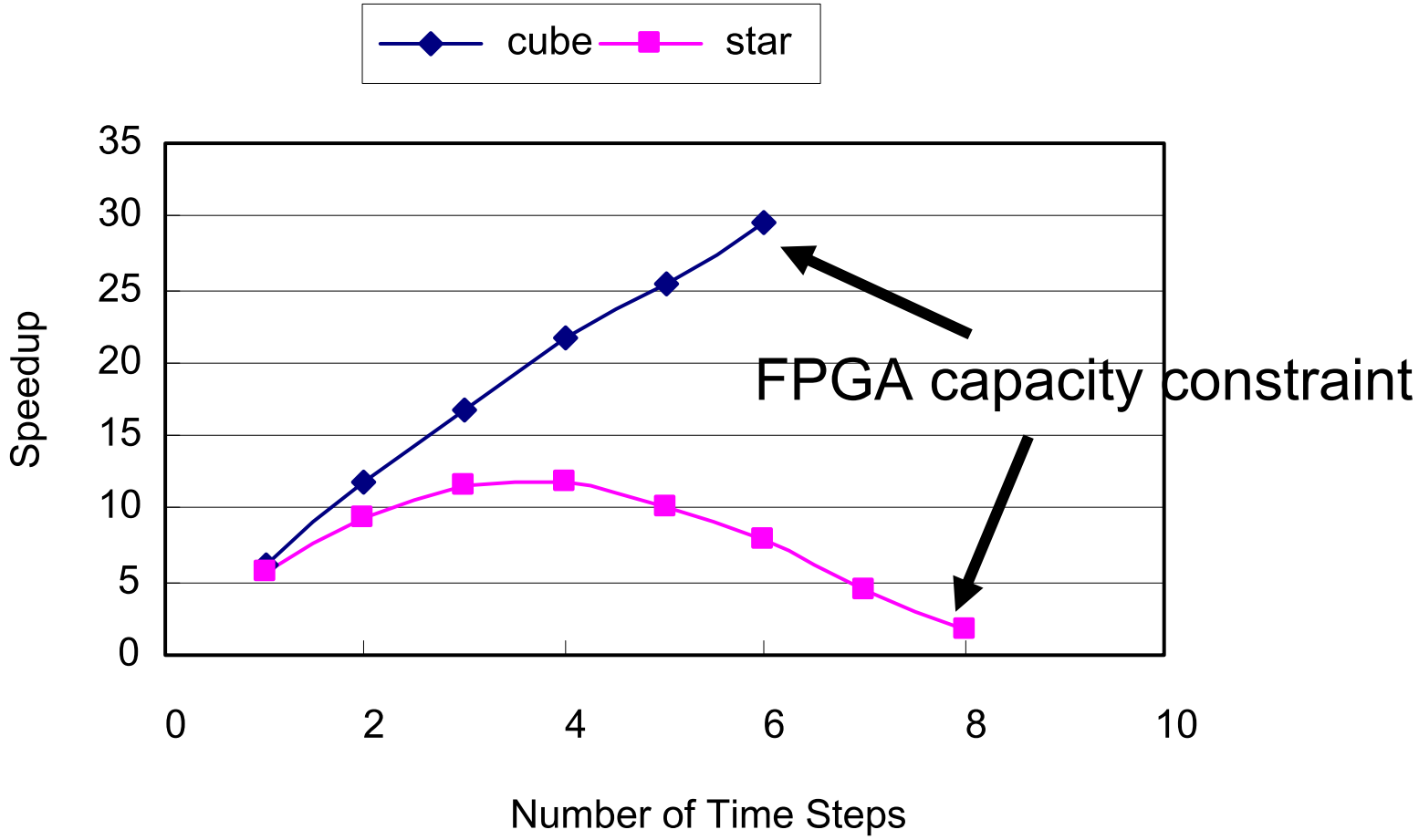


Cost of Processing Multiple Time Steps

- Redundant processing of an increased volume of overlapping boundaries
  - for a stencil with *ns* non-zero lags in each direction, we compute a *(nx+ns)*(ny+ns)*nz* block to get a *nx*ny*nz* result block for processing one time step
  - we compute a *(nx+ns*m)*(ny+ns*m)*nz* block to get a *nx*ny*nz* result block for processing *m* time step
  - star: *(100+6*4)*(100+6*4) /100*100 – 1 = 84.5%*

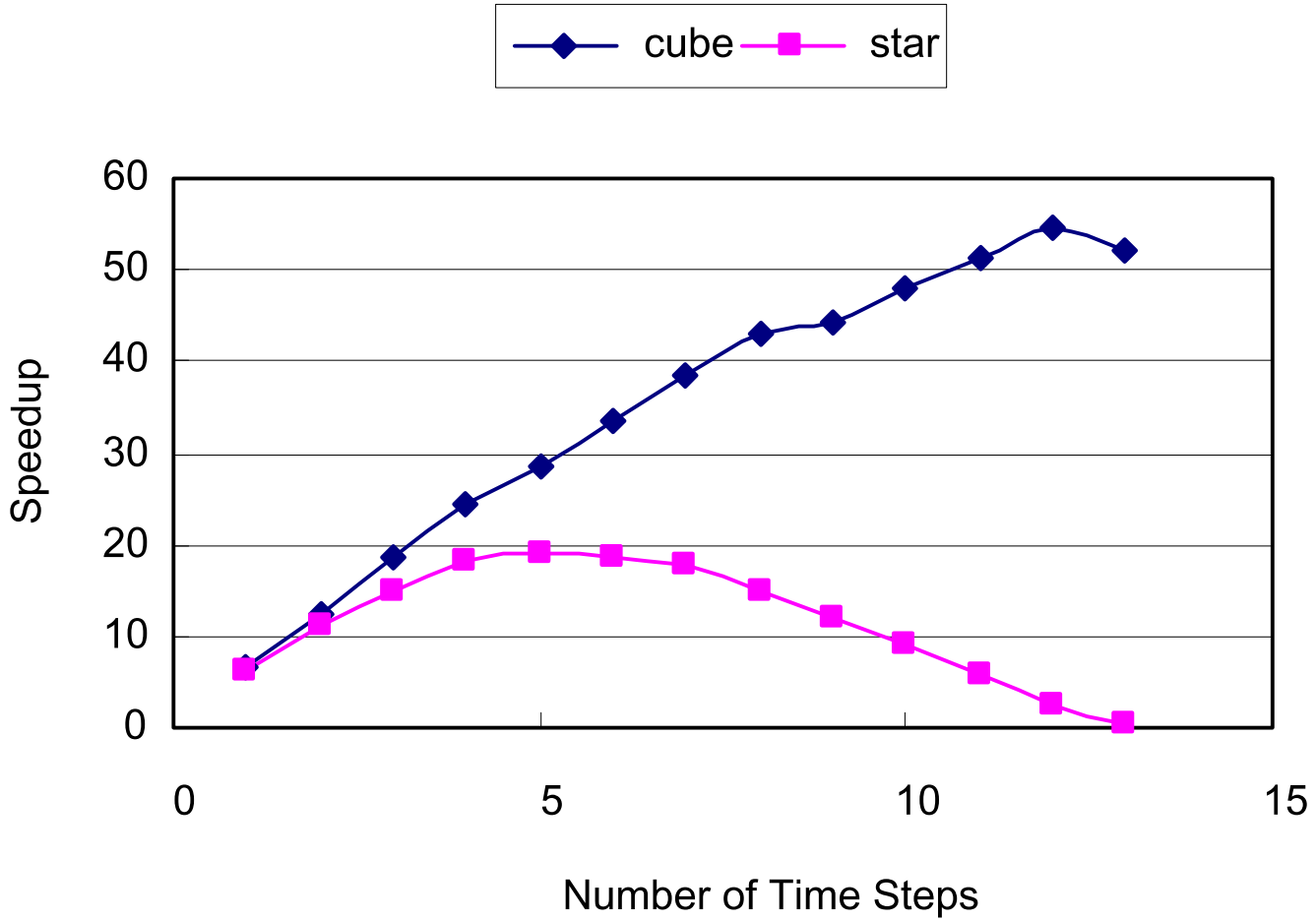
    cube: *(100+2*4)*(100+2*4)/100*100 – 1 = 16.6%*

Multiple Time Steps: Speedup over a Single-Core Implementation



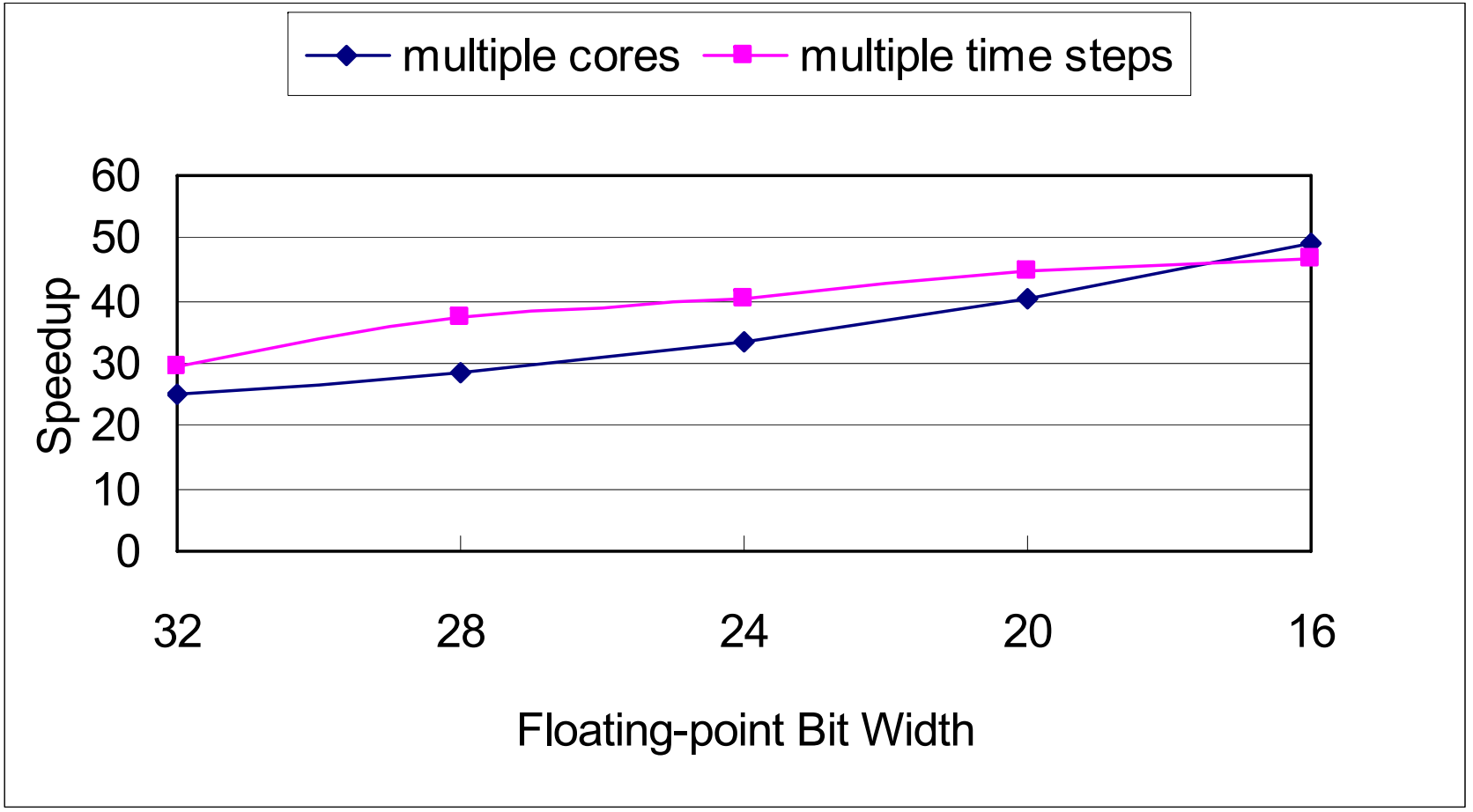
Using Virtex 5 LX330 FPGA

Multiple Time Steps: Speedup over a Single-Core Implementation

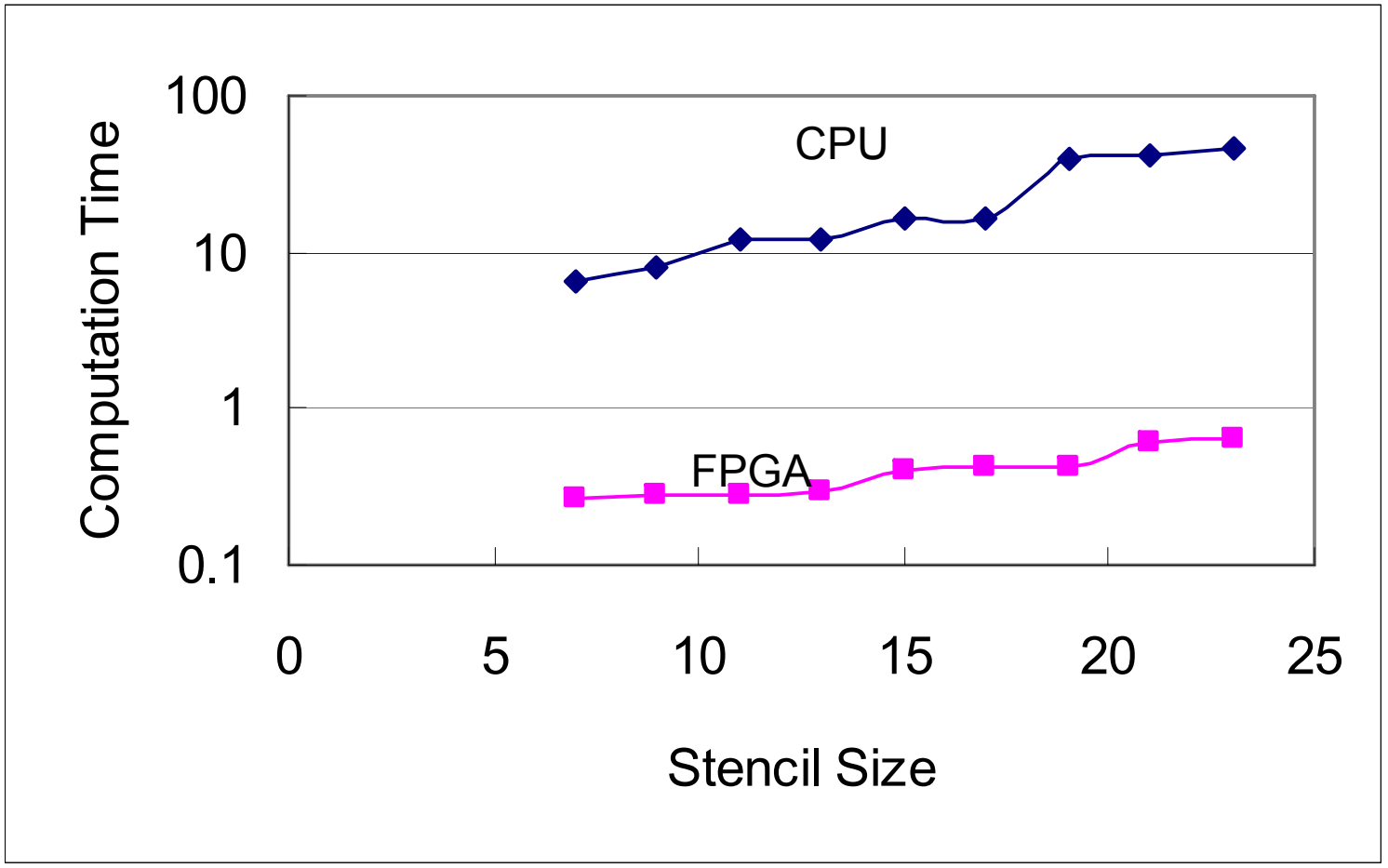


Using Virtex 6 SX475 FPGA

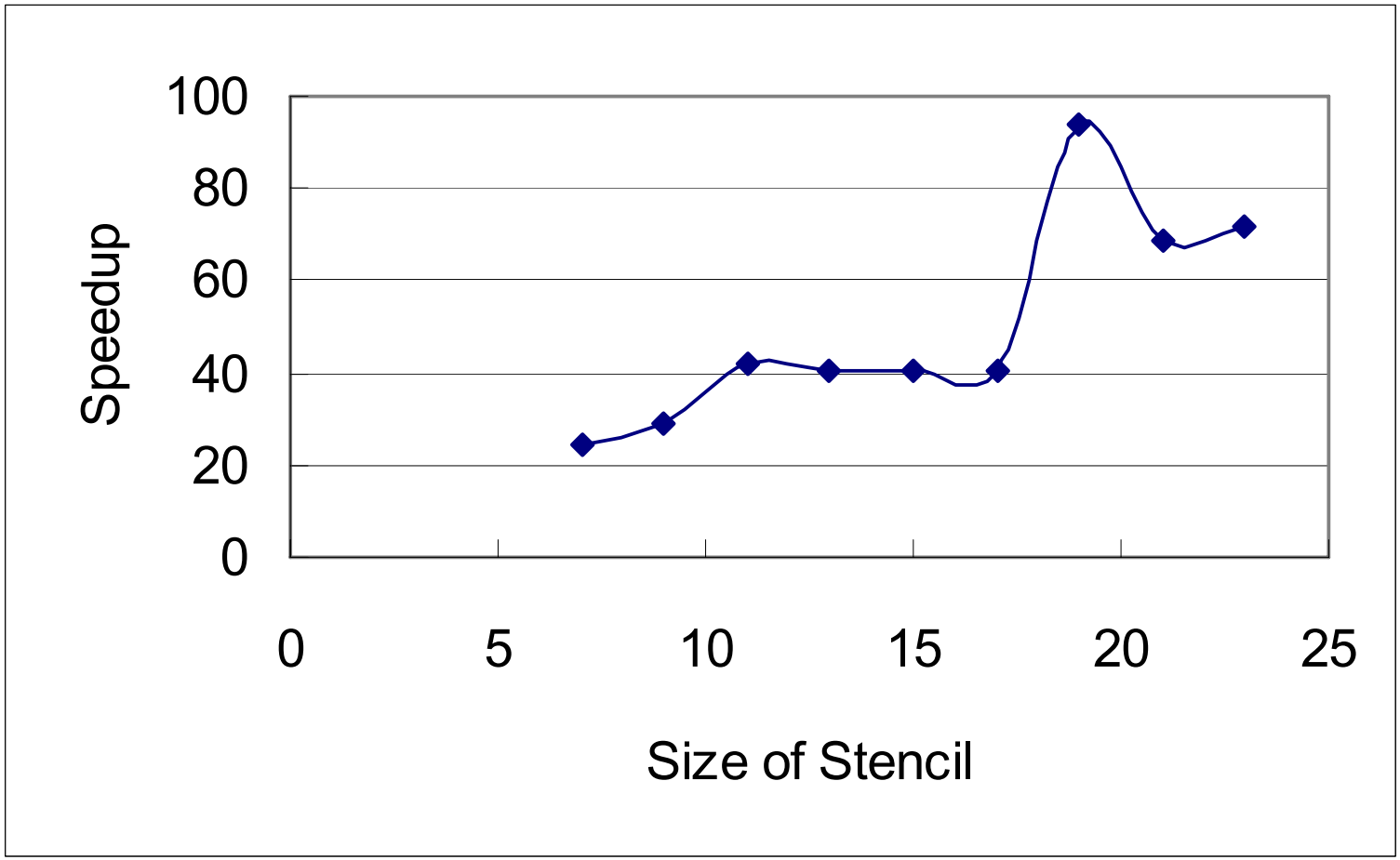
Reduced Precision: Speedup over a Single-Core Implementation



Different ‘Star’ Stencil Sizes



Different 'Star' Stencil Sizes: Speedup over a Single-Core Implementation



Conclusions

- Stream computing on FPGAs achieves great performance for 3D convolution computations
  - up to two orders of magnitude speedup
  - better scalability over different stencil sizes
- The ‘cube’ stencil shows much better performance than the ‘star’ stencil for stream computing on FPGAs

Thanks for your attention.
Any Questions?

haohuan@gmail.com