

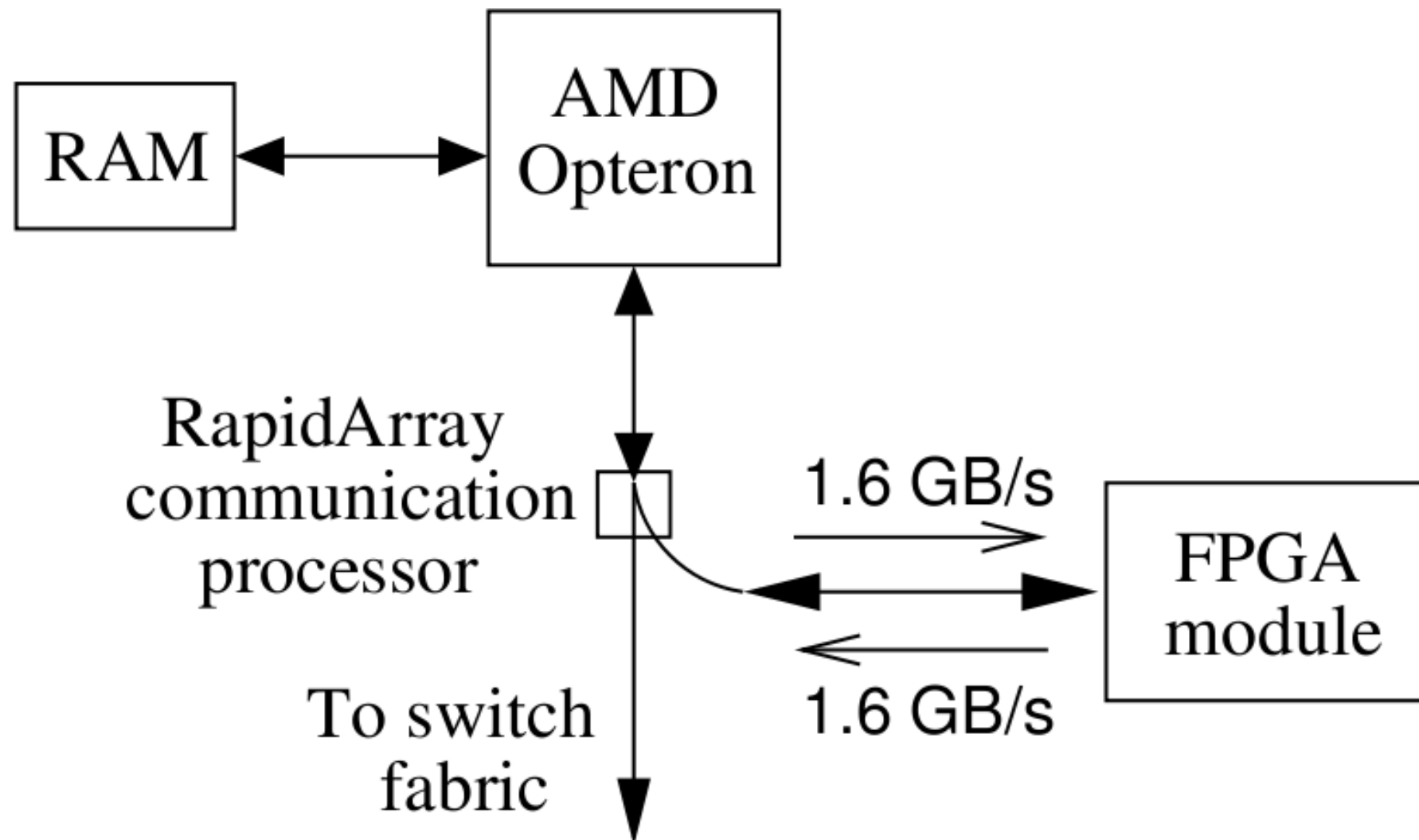
Automatic mapping of nested loops to FPGAs

By Bondhugula et al
Presented by Eric Christiansen

Goal of the paper?
First, some explanation

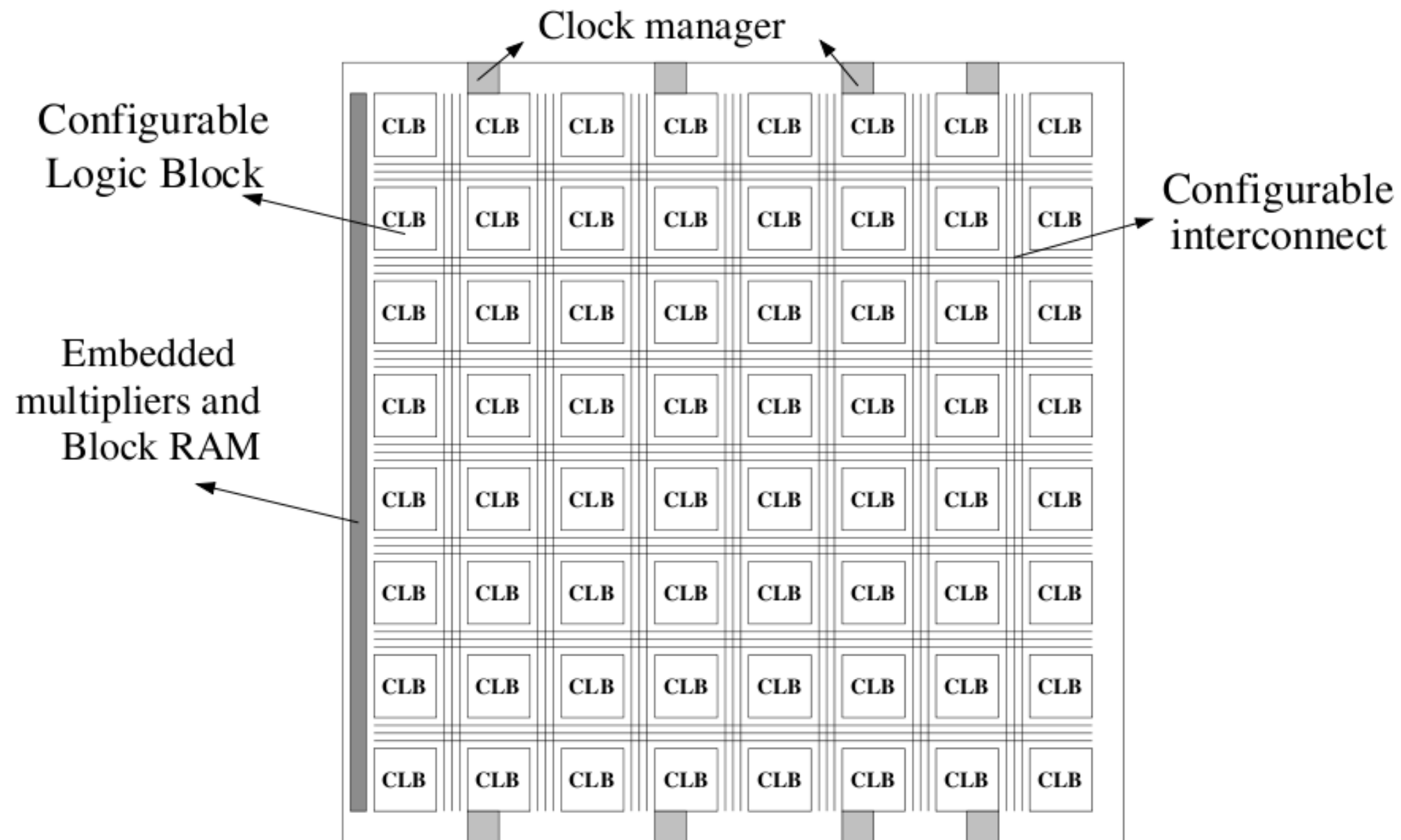
FPGA

- field programmable gate array
- like GPU, is coprocessor that connects to CPU via high-bandwidth bus
- internal computation model unlike GPU



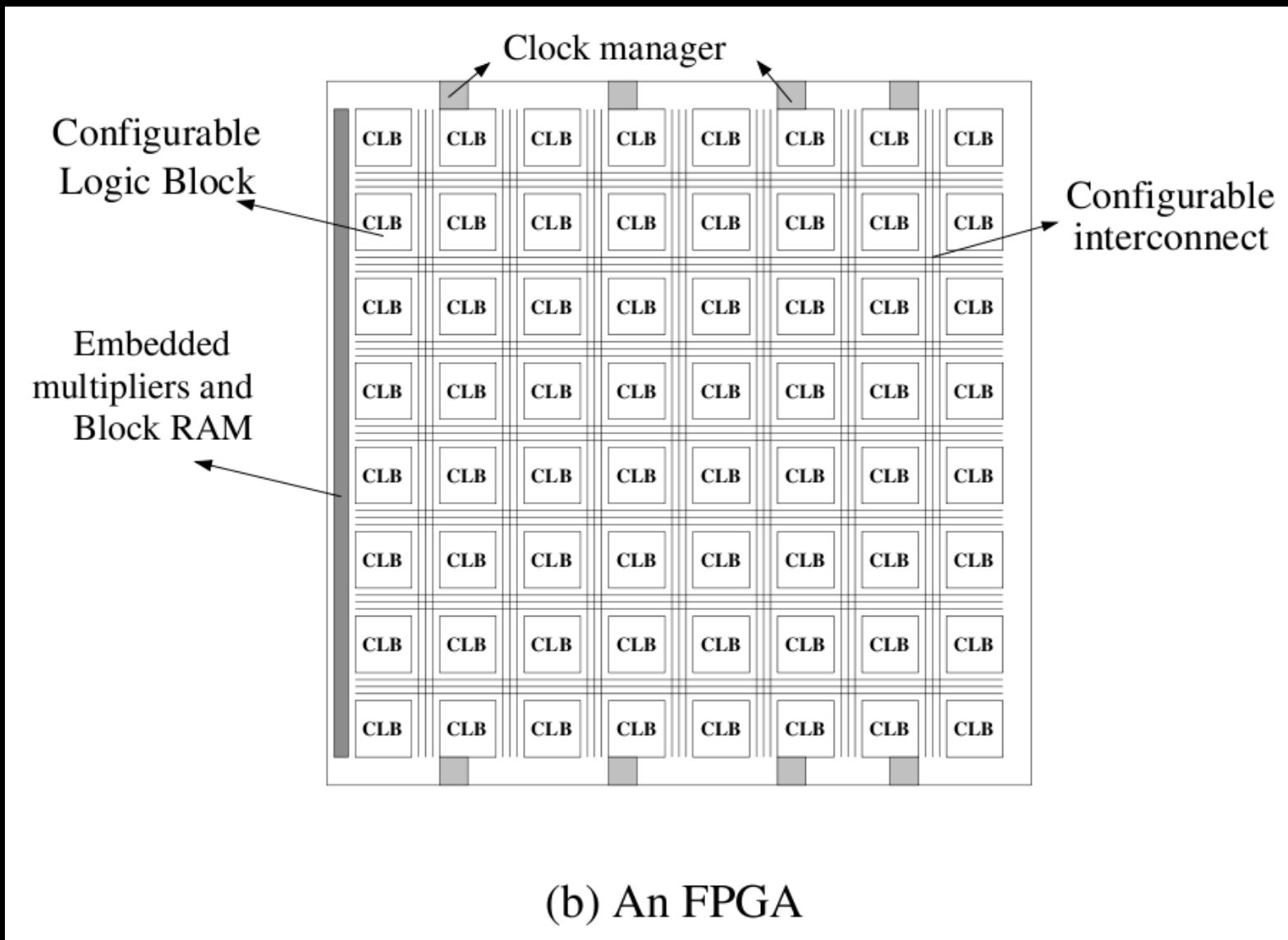
(a) The Cray XD1 system

- like GPU, is coprocessor that connects to CPU via high-bandwidth bus



(b) An FPGA

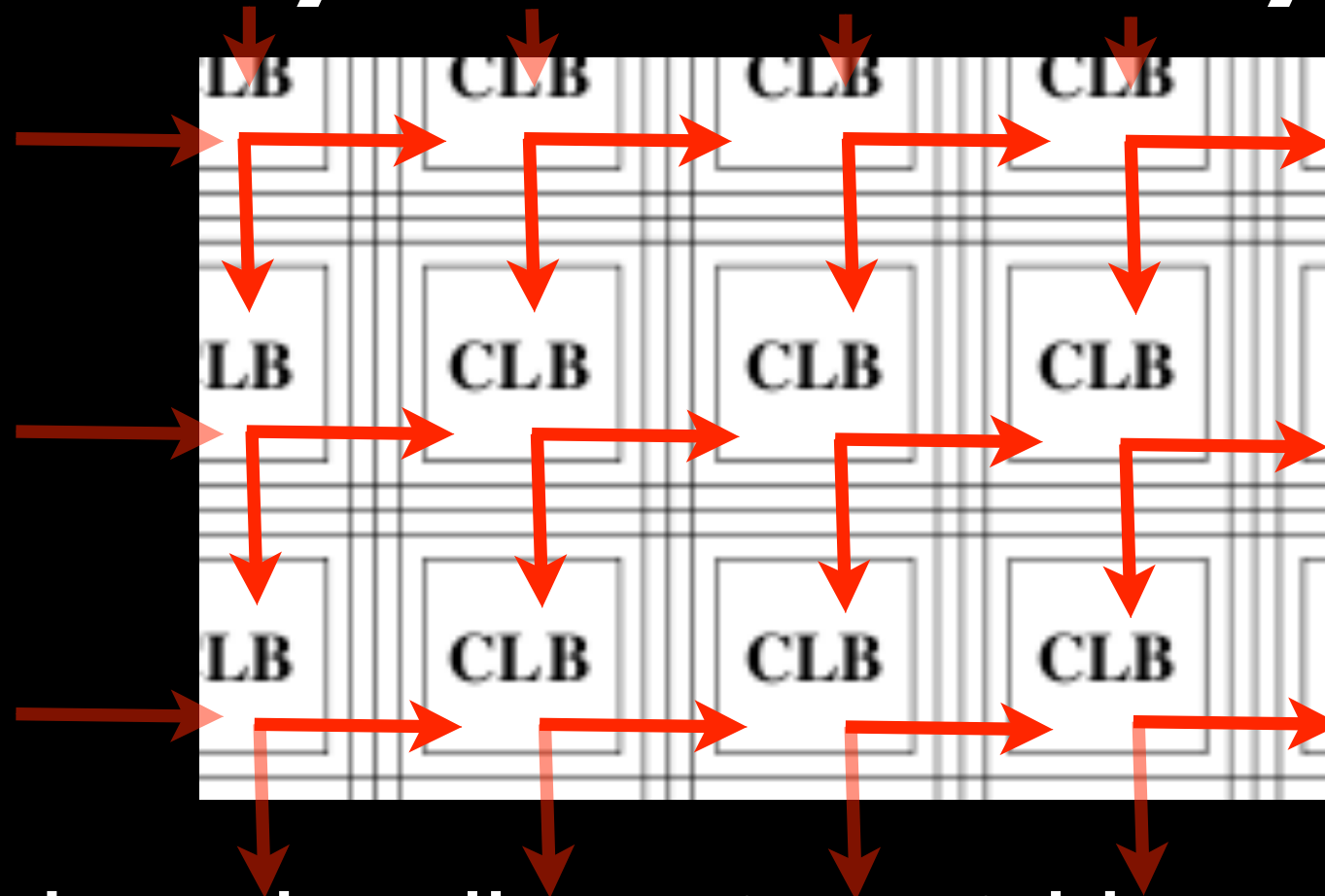
- internal computation model unlike GPU



- configurable logic block (CLB)
- configurable interconnect
- “island style” hierarchical routing fabric
- no global memory

Computation model:

Systolic array



- each node talks to its neighbors
 - this paper: DAG
- synchronous or asynchronous

Okay, that's an FPGA
What's a nested loop?

(Perfectly) nested loop

```
do  $i_1 = L_1, U_1$   
  ...  
    do  $i_n = L_n, U_n$   
       $\mathcal{S}(i_1, \dots, i_n)$ 
```

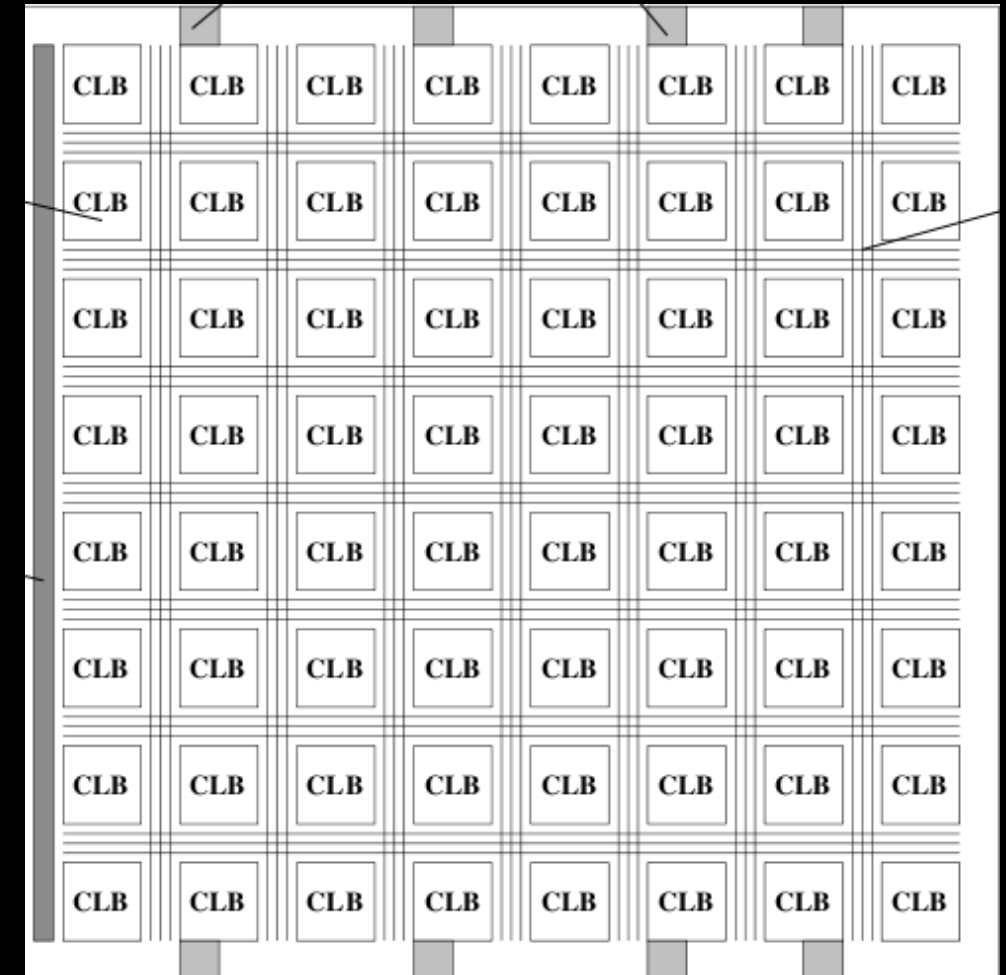
- only operation is innermost
- loop ranges *can* depend on each other

Goal of the paper

do $i_1 = L_1, U_1$

...

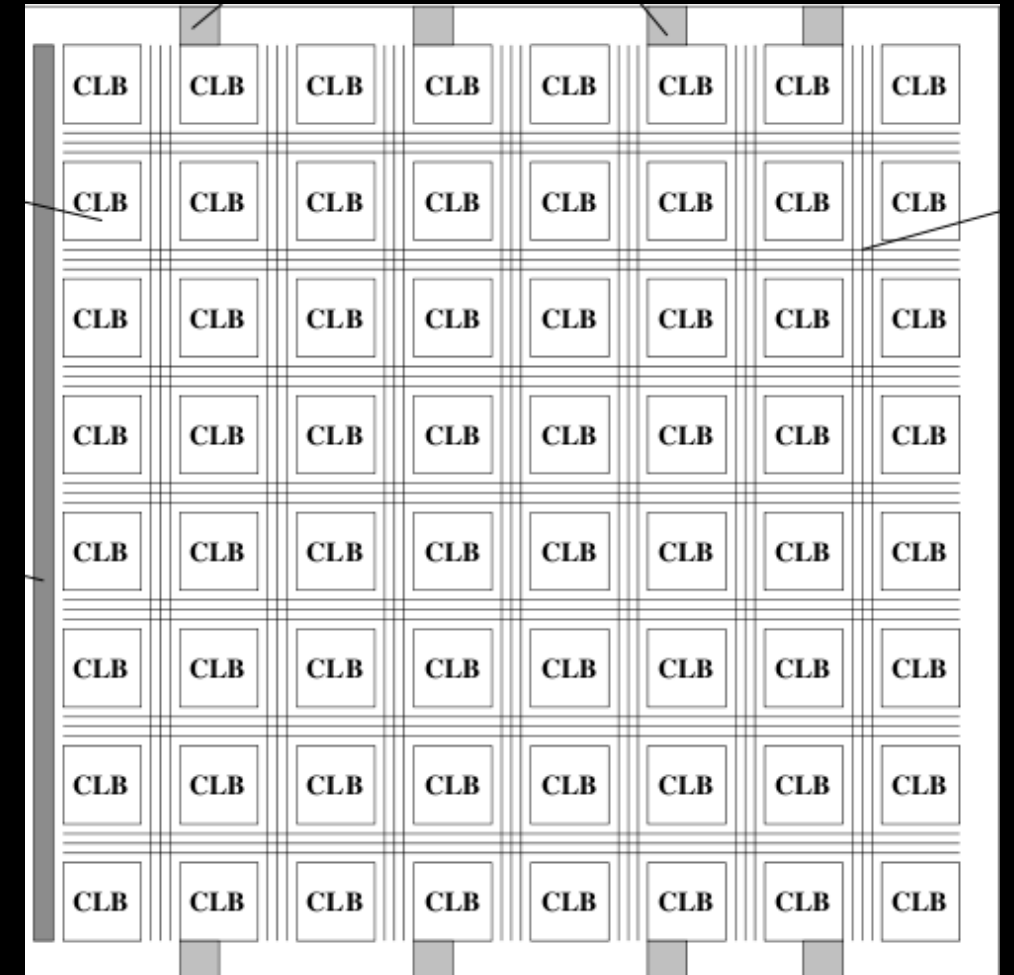
do $i_n = L_n, U_n$
 $\mathcal{S}(i_1, \dots, i_n)$



Goal of the paper

```
do  $i_1 = L_1, U_1$   
...
```

```
do  $i_n = L_n, U_n$   
   $S(i_1, \dots, i_n)$ 
```

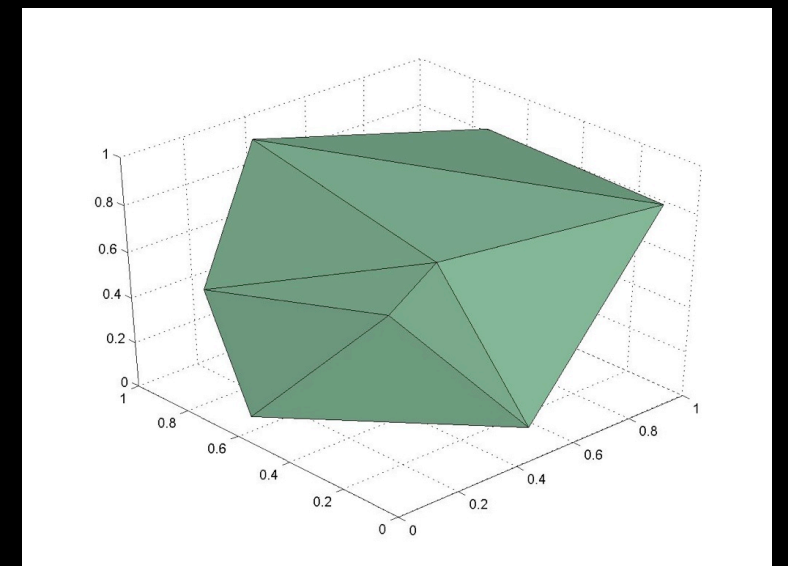
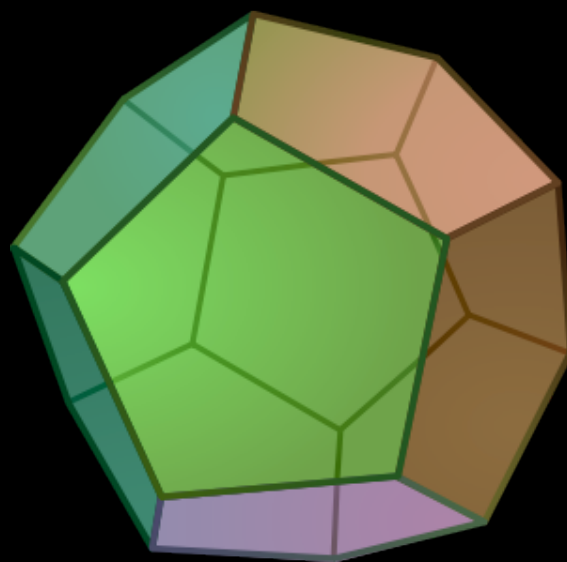
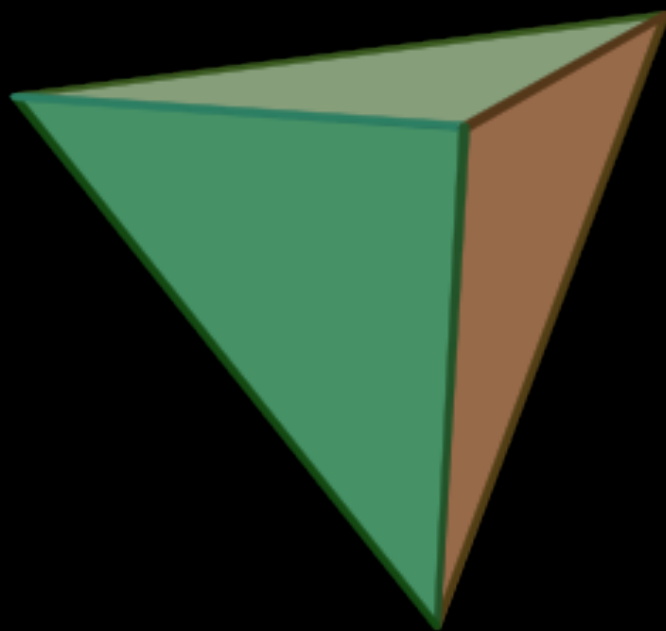


- locally parallel globally sequential (LPGS)

Now we understand
the goal

Let's talk about how
the authors think about
the problem

Iteration-space polytope



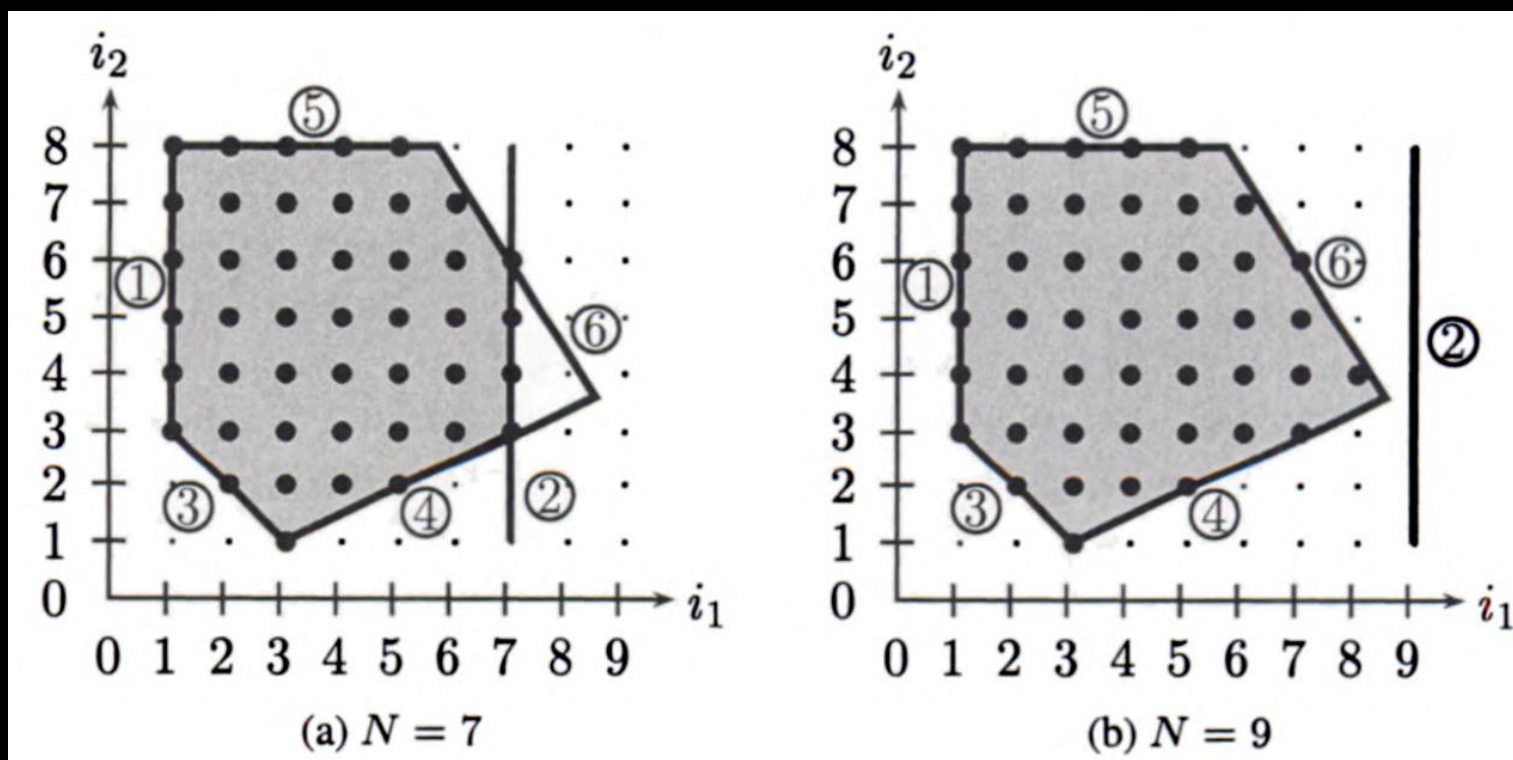
- polytope - a convex set with flat edges, e.g. a triangle or a cube

Iteration-space polytope

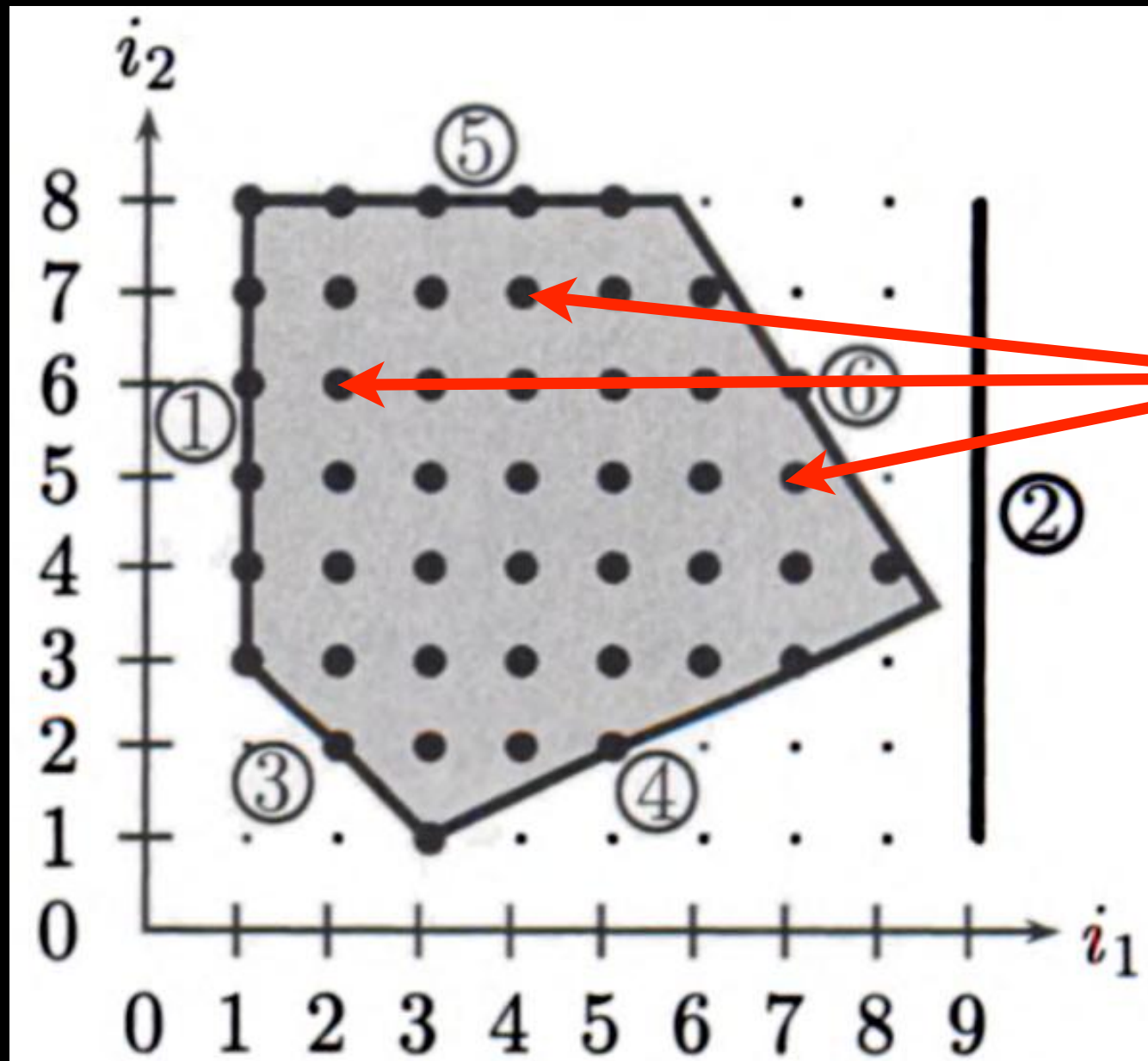
do $i_1 = 1, N$

do $i_2 = \max(-i_1 + 4, \lceil \frac{i_1 - 1}{2} \rceil), \min(8, \lfloor \frac{-3i_1 + 33}{2} \rfloor)$

$A(i_1, i_2) = A(i_1 - 1, i_2) + A(3i_1, i_2 + 1)$

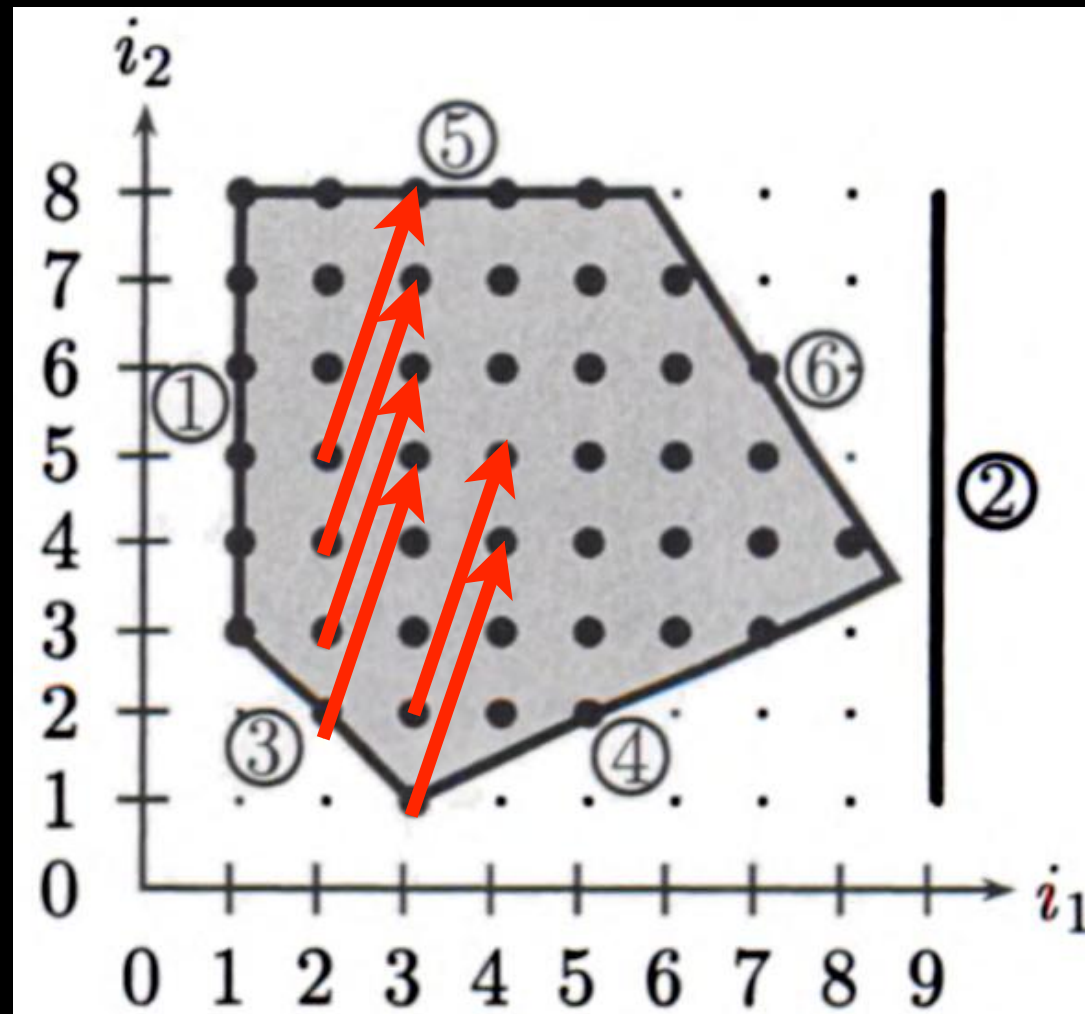


Computations are points in a polytope

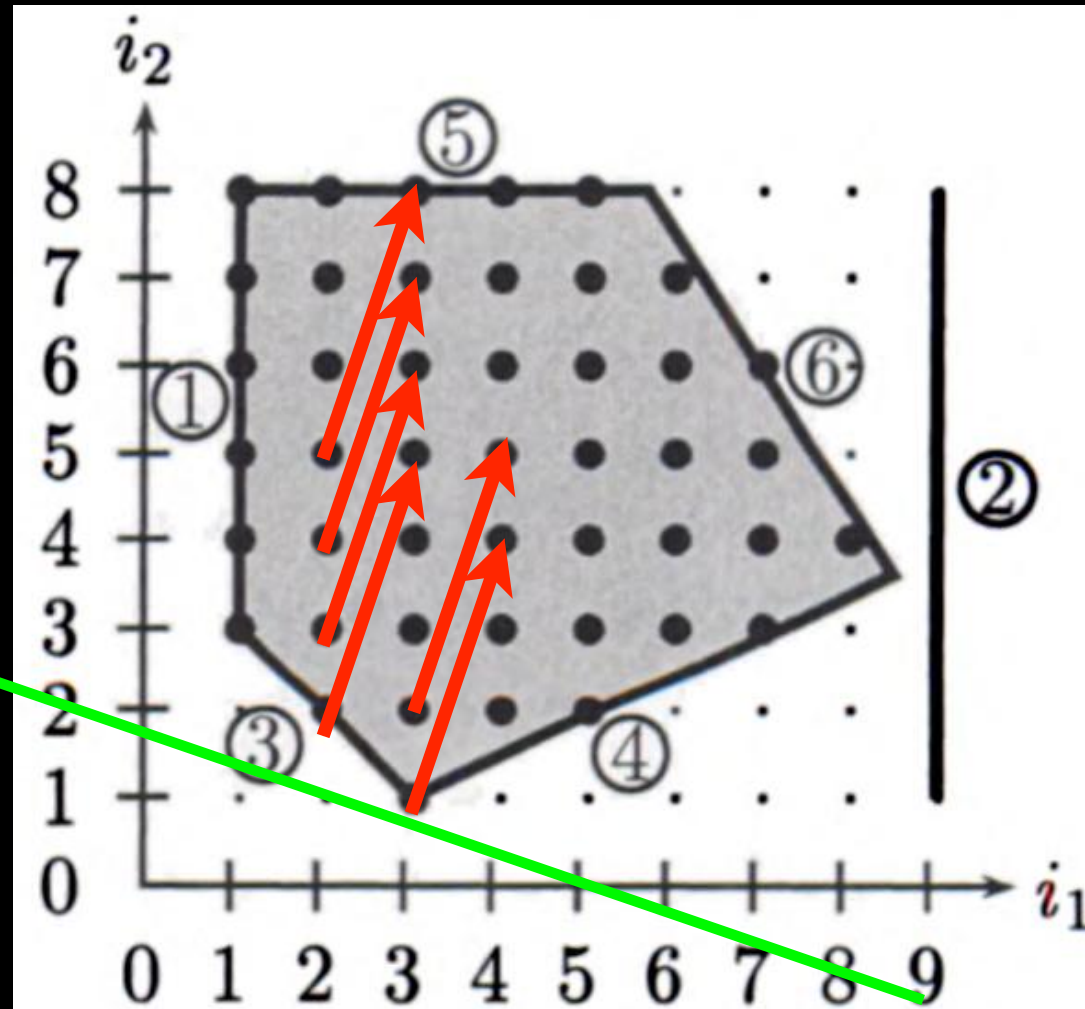


- each point is a computation performed in a perfectly nested loop

Dependencies are collections of vectors

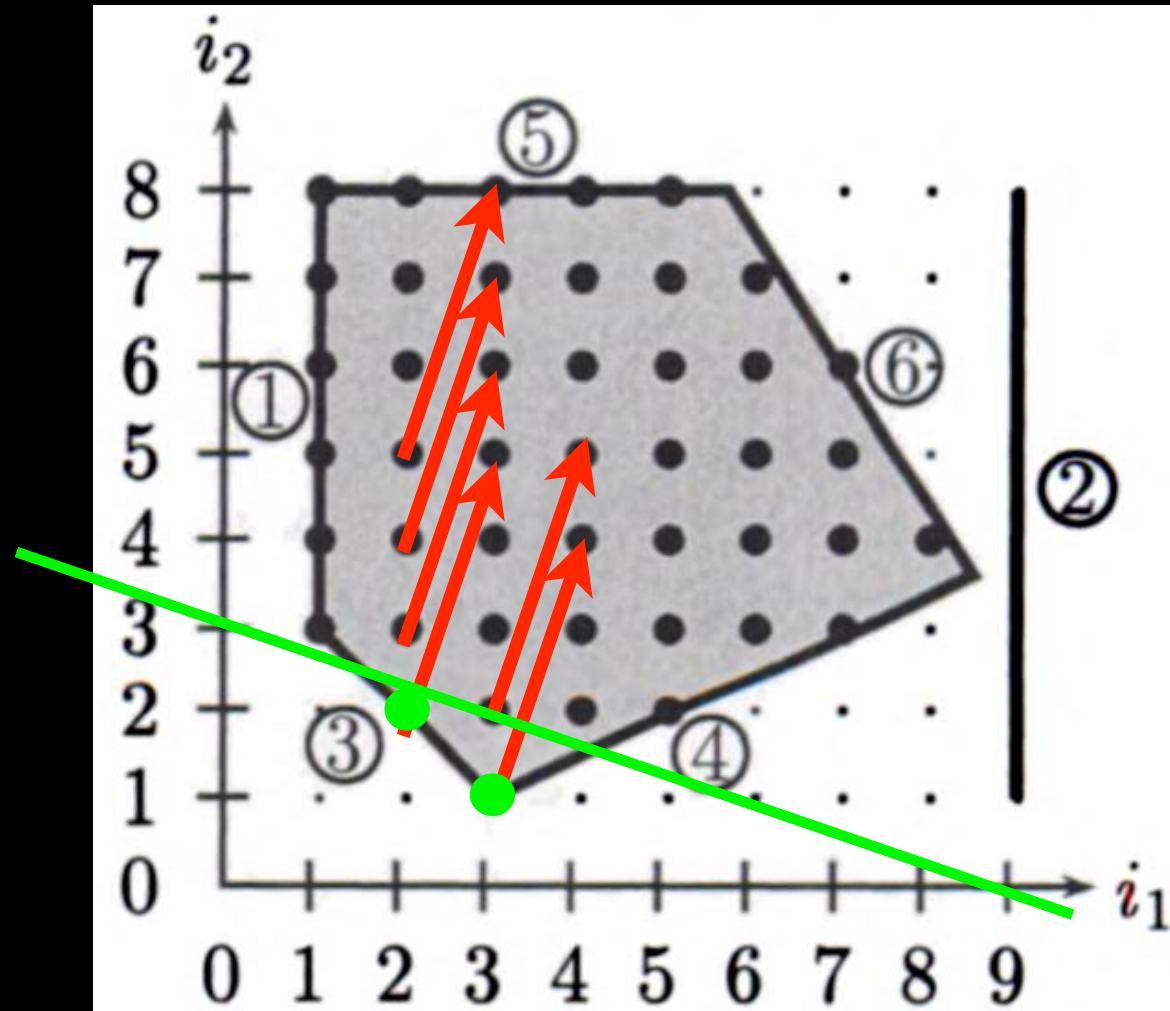


Computation example



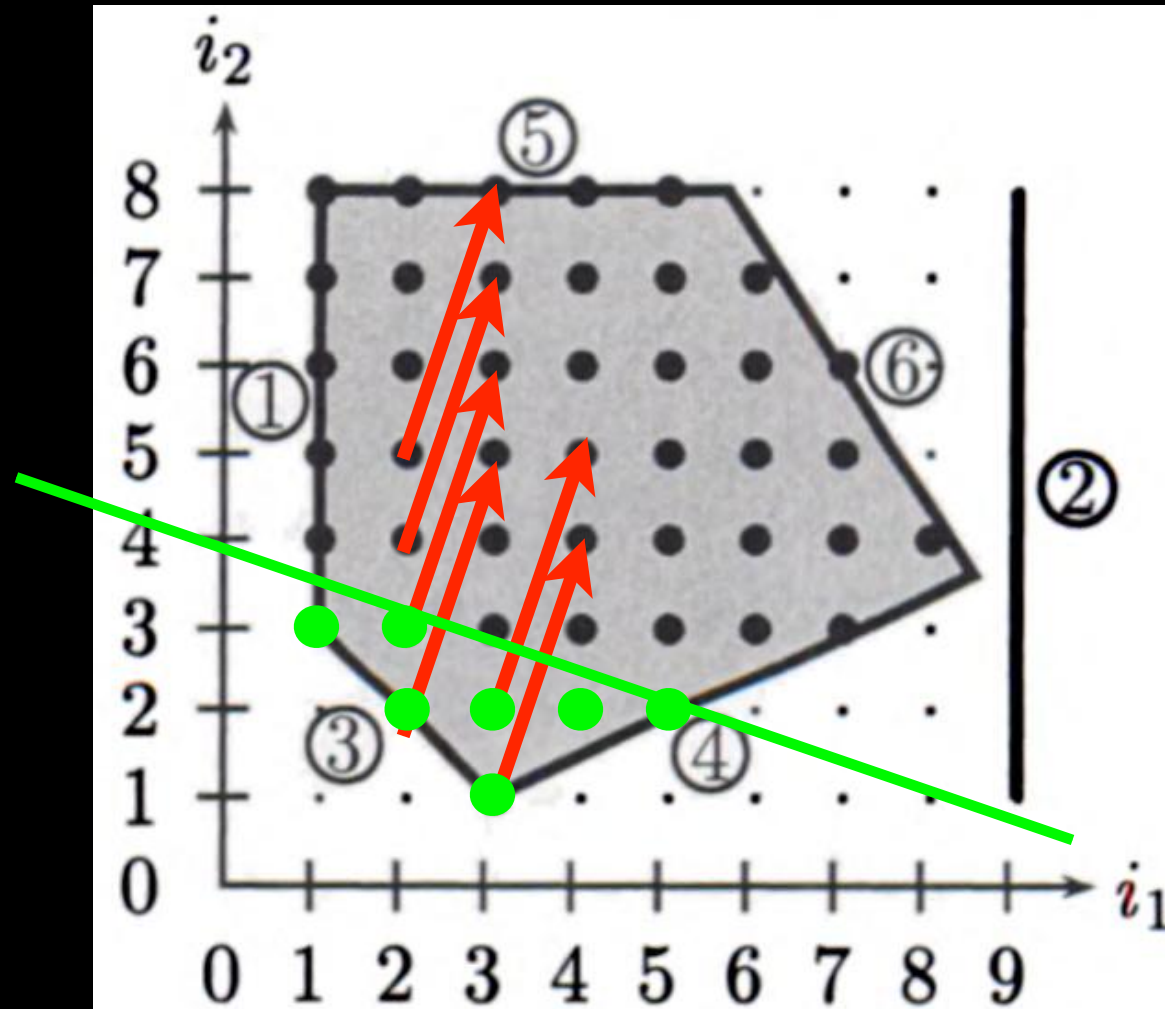
Iteration 0

Computation example



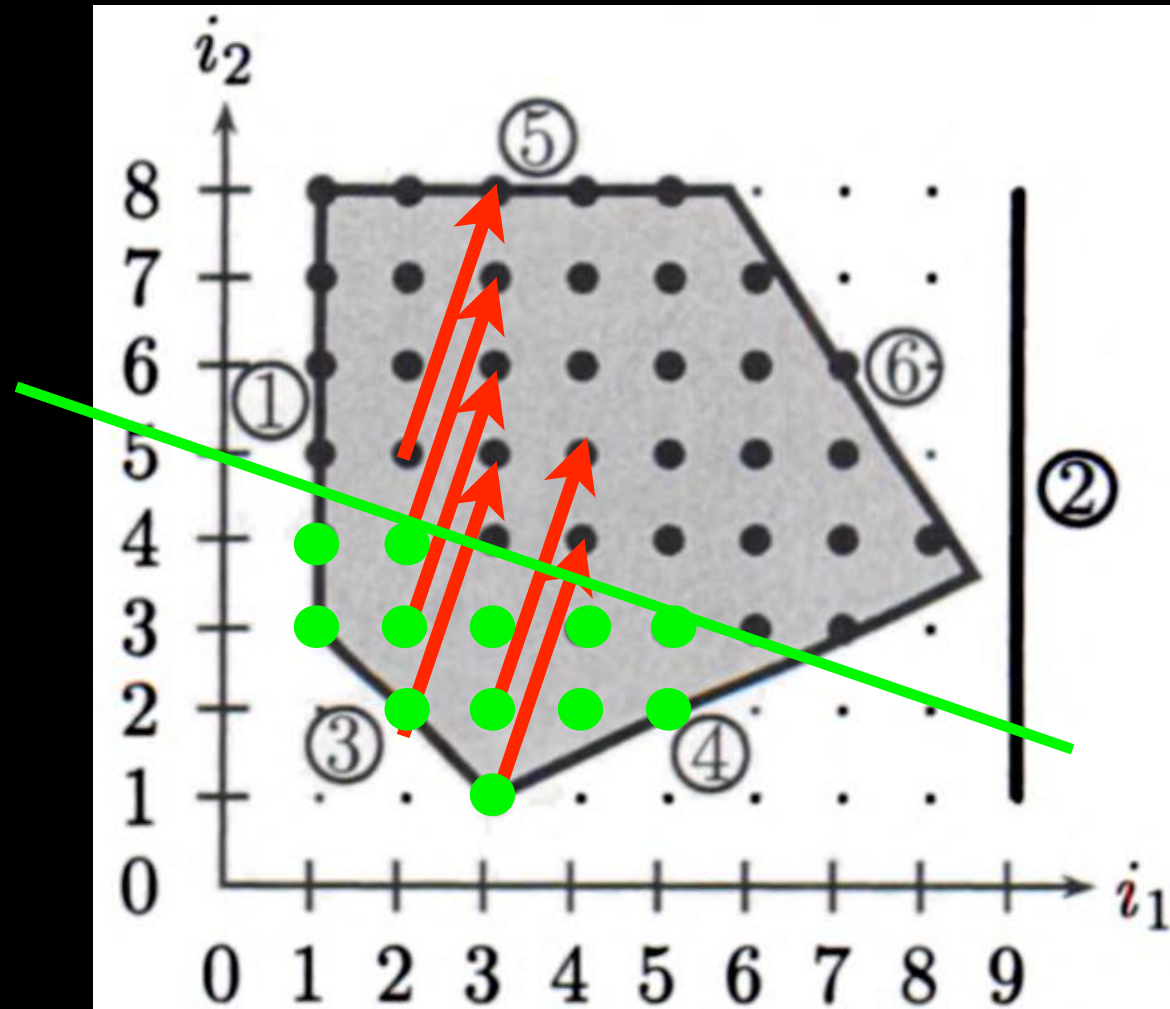
Iteration 1

Computation example



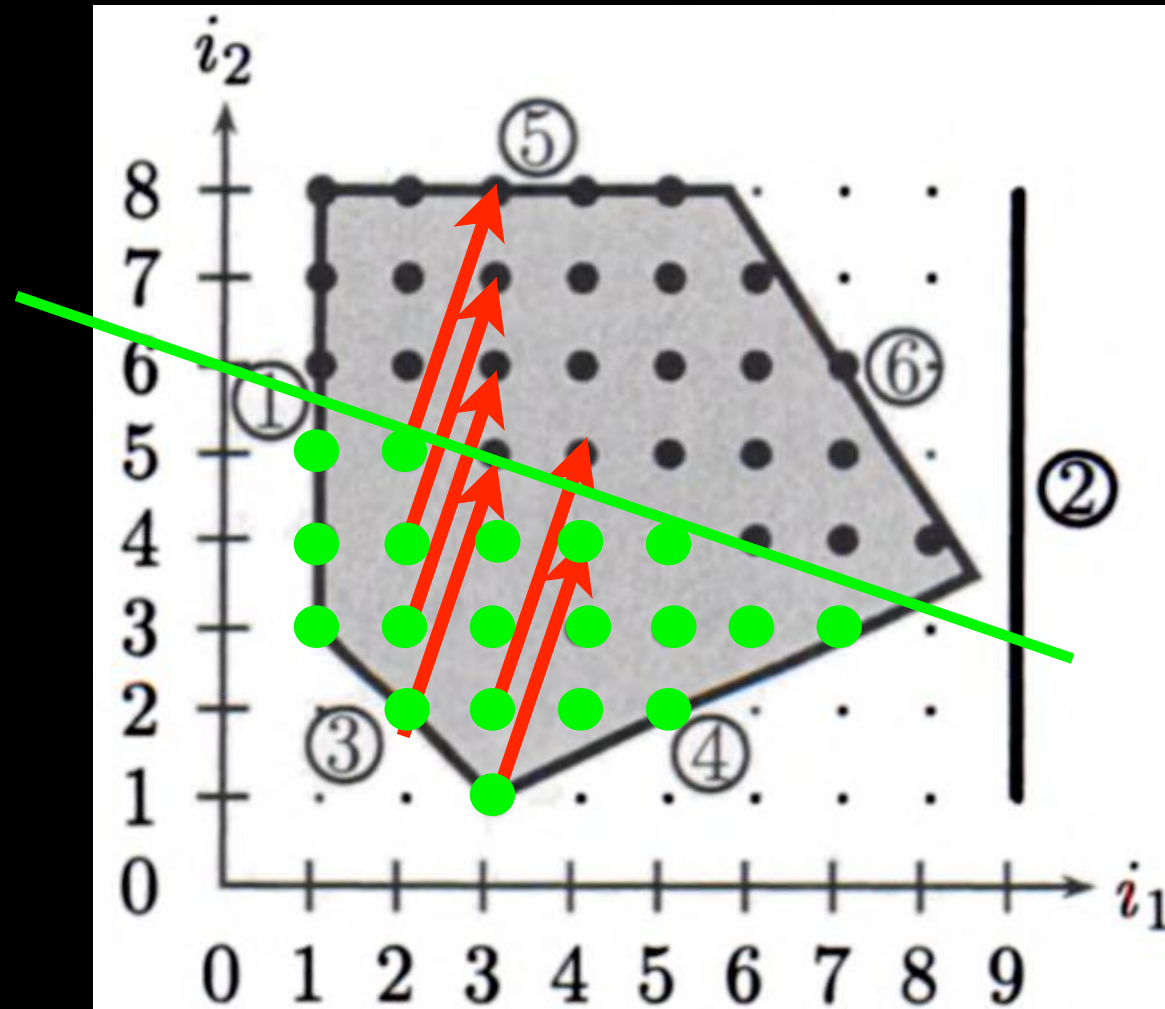
Iteration 2

Computation example



Iteration 3

Computation example



Iteration 4

It comes down to
finding a direction
through the polytope (a
single vector)*

*complicated by consideration of various
types of parallelism

Best direction through
the data is parallel to
the constraints

Q: But what if there is
no one parallel
direction?

Q: But what if there is
no one parallel
direction?

A: Just need
 $\text{rank}(\text{dependencies}) <$
 $\text{rank}(\text{polytope})$

Pipelined parallelism

- communication parallelism
 - $\text{rank}(\text{polytope}) > \text{rank}(\text{constraints})$
- pipelined parallelism
 - $\text{rank}(\text{polytope}) > \text{rank}(\text{constraints}) + 1$

Final algorithm

1. find communication-free parallelism
2. find pipelined parallelism
3. find time dimensions (not discussed here)

Find communication-free parallelism

$$\begin{aligned} D' &\leftarrow D \setminus R \\ r_1 \cdot \vec{d}_j &= 0 \text{ for each } \vec{d}_j \in D' \\ \text{minimize } &\sum_{i=1}^n c_{1i} \end{aligned}$$

- r_1 is the vector being learned
- D' are the constraints
- c_{1i} appear to be regularizers

Find pipelined parallelism

$$\begin{aligned} r_2 D &\geq \vec{0}^T \\ \sum_{\vec{d}_j \in D''} r_2 \cdot \vec{d}_j &> 0 \\ \text{minimize } \sum_{\vec{d}_j \in D''} r_2 \cdot \vec{d}_j \end{aligned}$$

- r_2 is the vector being learned
- D are the constraints
- D'' includes the direction r_1

Skipping how to turn
this into FPGA
instructions

fin

figures taken from:

- the original paper
- “Loop tiling for parallelism” by Jingling Xue
- Google image search (for polytopes)