CS294-7: Reconfigurable Computing

Day 9: Tuesday, February 18

Spatial Computing Styles

- Admin
- Sampling of Systolic Structures

Admin

- **Project 1**
  - sent feedback Thursday
- **CAD Tools for Project 1**

Systolic

General methodology for mapping computations into hardware (spatial computing) structures.

- **Composition**
  - simple compute cells
  - interconnected in regular pattern (local interconnect only)
  - pipelined communication between cells
  - I/O at boundaries
- **Candidate Problem Requirements**
  1. make multiple use of each input data item
  2. concurrency
  3. few types of simple cells
  4. simple/regular data and control flow

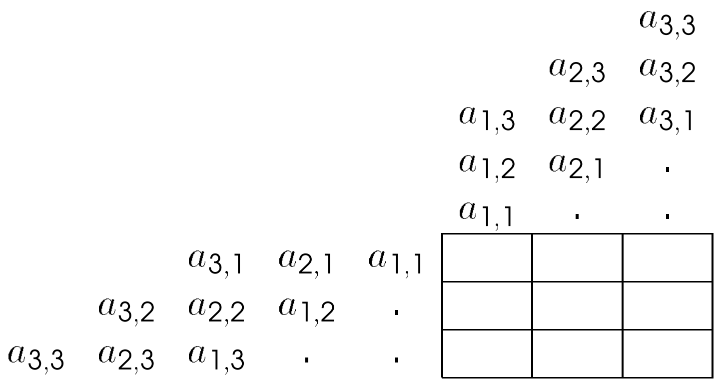
Today's Reading

- **Dynamic Programming**
- **Transitive Closure**
- **Priority Queue**
- **Database operators:**
  intersect, difference, project, join, divide

Transitive Closure

Like Matrix Multiply:

$$a_{i,j}^{(t+1)} = a_{i,j}^{(t)} + a_{i,t}^{(t)} \cdot a_{t,j}^{(t)}$$



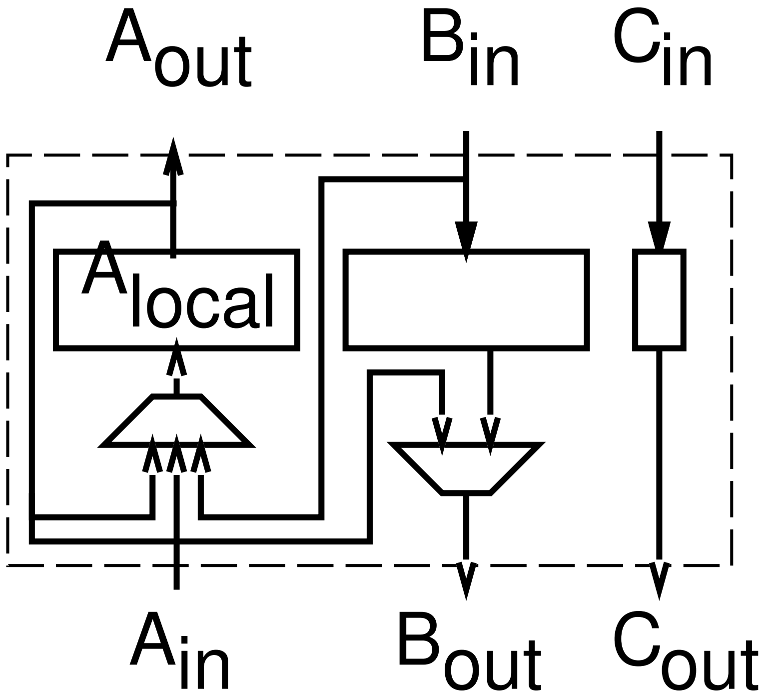
- replace $a_{i,j}$'s as cross "home" in array
- requires 3 passes to complete
- $5n$ cycles on n^2 array
- shortest path similar

Priority Queue

- Maintain ordered list of values
- Two basic operations:
  - Insert
  - Extract-min
- $O(1)$ response time
- Use:
  - move/task ordering for optimization and search
  - priority-based resource allocation
  - sorting

Priority Queue

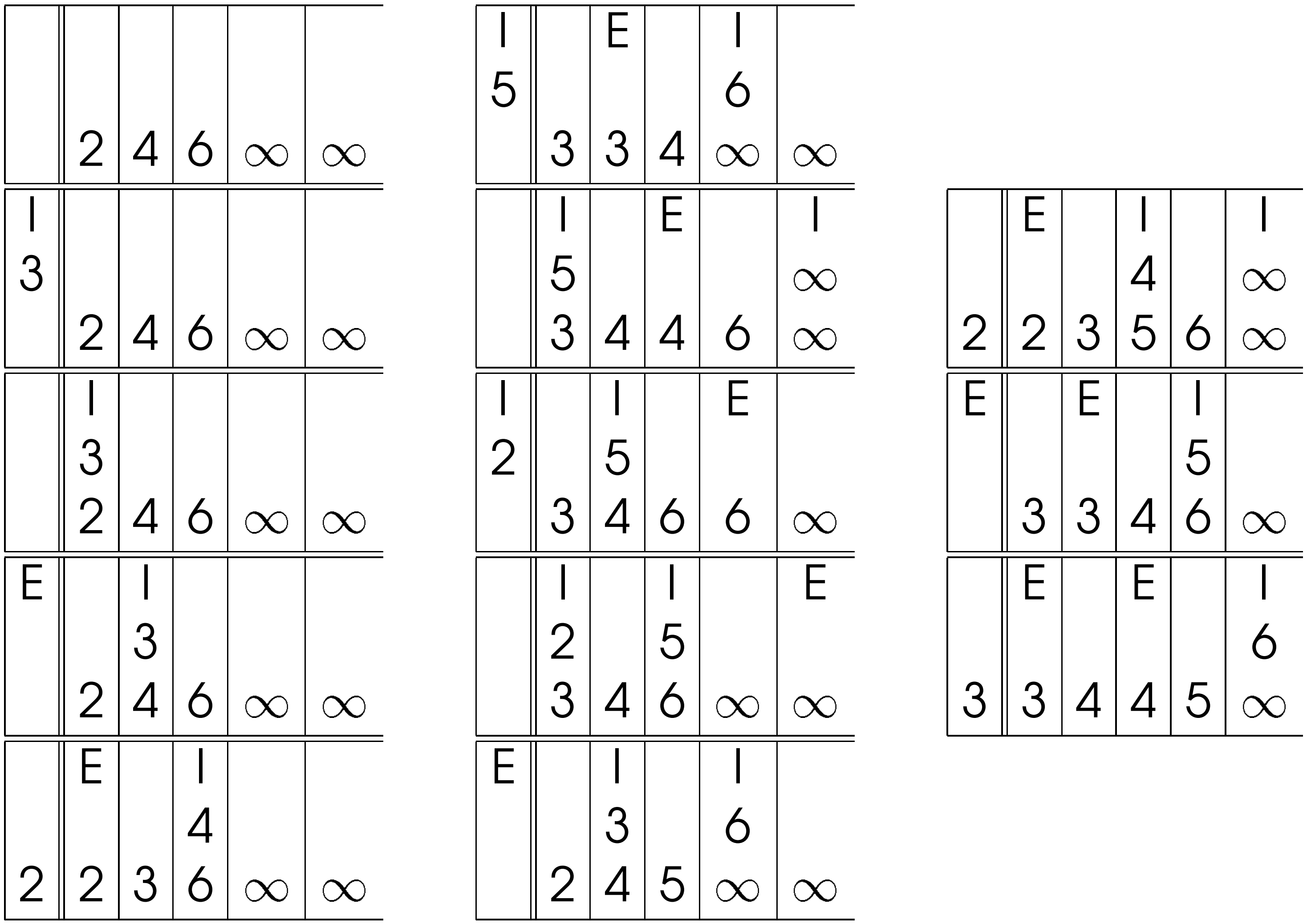
```
if (C_in =insert)
    if (A_local ≤ B_in)
        B_out = B_in
        A_local = A_local
    else    ;; (A_local > B_in)
        B_out = A_local
        A_local = B_in
else if (C_in =extract)
        A_local = A_in
        B_out = B_in
A_out = A_local
C_out = C_in
```



Priority Queue – Example

Operation	out	queue
Start		2 4 6
Insert 3		2 3 4 6
Extract	2	3 4 6
Insert 5		3 4 5 6
Insert 2		2 3 4 5 6
Extract	2	3 4 5 6
Extract	3	4 5 6

Priority Queue – Example



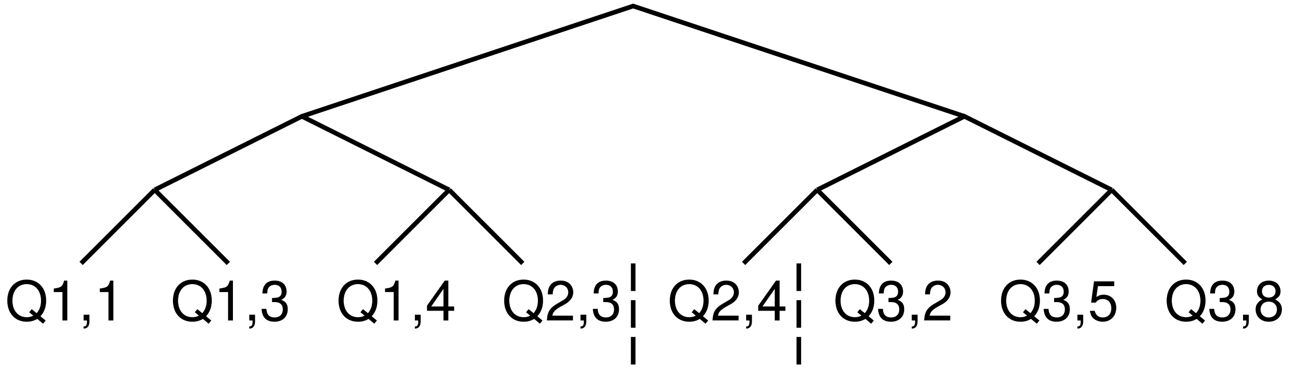
Priority Queue

- Constant response time
- Does $O(n)$ work doing insert (versus $O(\log(n))$)
- Efficient only
  - as approach full (w/out overflowing)
  - continuously doing insert/extracts

Systolic Tree Multiqueue

- Shares capacity among multiple queues
- Efficient operating on set of queues

- $O(\log(n))$ operation latency



Systolic Priority Multiqueue

Combine:

- m systolic priority queues
- systolic tree multiqueue

Gives:

- constant time response
- PE sharing among queues
  (work and storage)

Relational Database

Relation is a collection of tuples (list, array, sequence)

Tuple is a fixed number of fields (`struct`, record)

Fields

tuple #	1	2	3	4
0	Ben Bitdiddle	Intern	123456789	1.00
1	Alyssa P. Hacker	Programmer	135792468	2.50
2	Louis Reasoner	Programmer	888324681	3.00

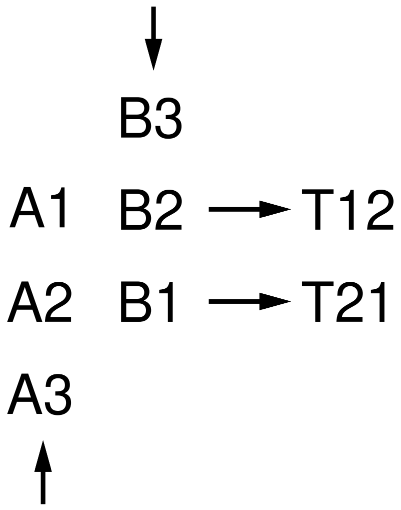
DB: Intersection

Intersection: $A \cap B$ – all records in both relation A and B.

Must compare all $|A| \times |B|$ tuples.

Compare via convolution like SPLASH sequence compare.

	B1	B2	B3
A1	T11	T12	T13
A2	T21	T22	T23
A3	T31	T32	T33



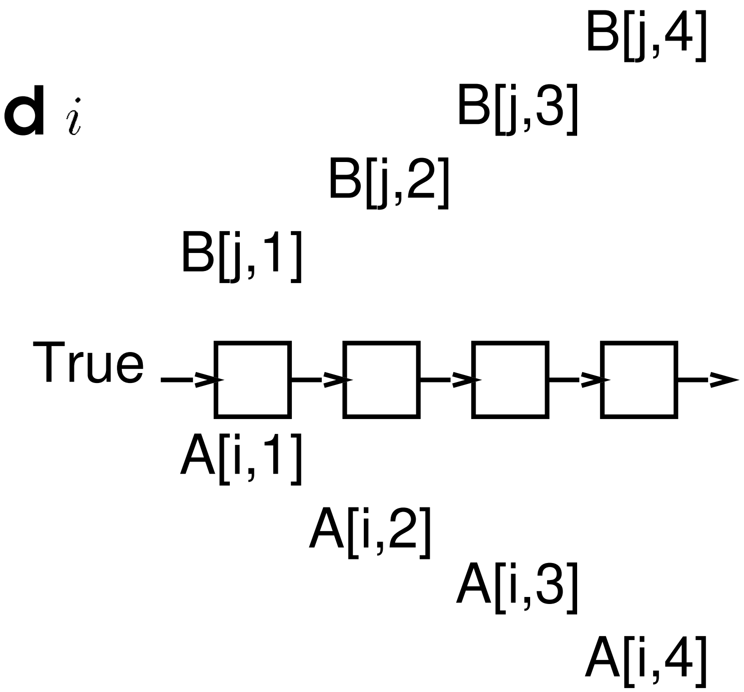
OR along row or column to get inclusion bitvector.

DB: Tuple Comparison

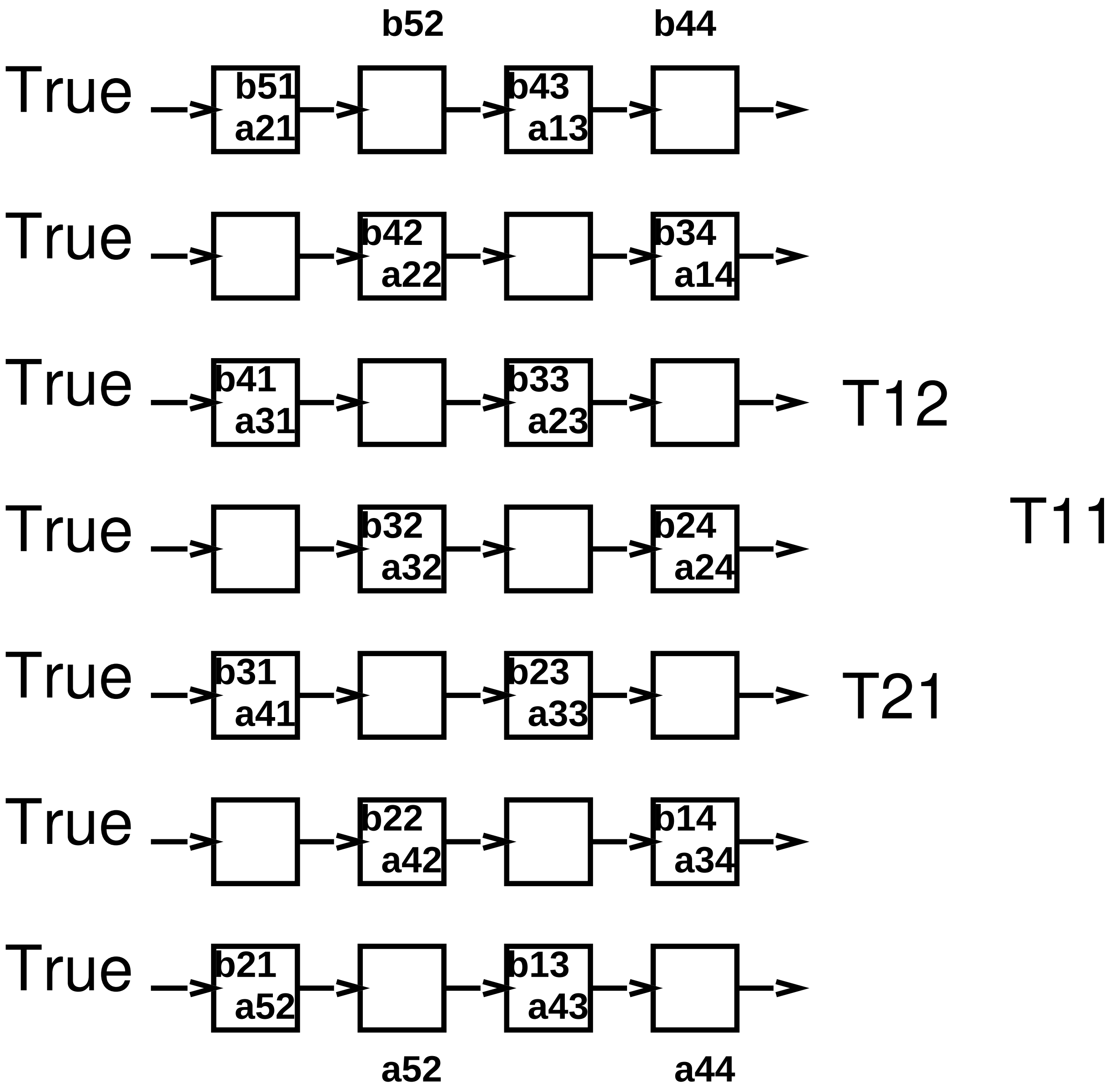
Problem: Tuples are long – do not want tuple comparison time to limit computation rate
Strategy: Perform comparison in pipelined manner by (sub) fields

- **stagger fields**
- **arrange to compute field i on cycle after $i-1$**

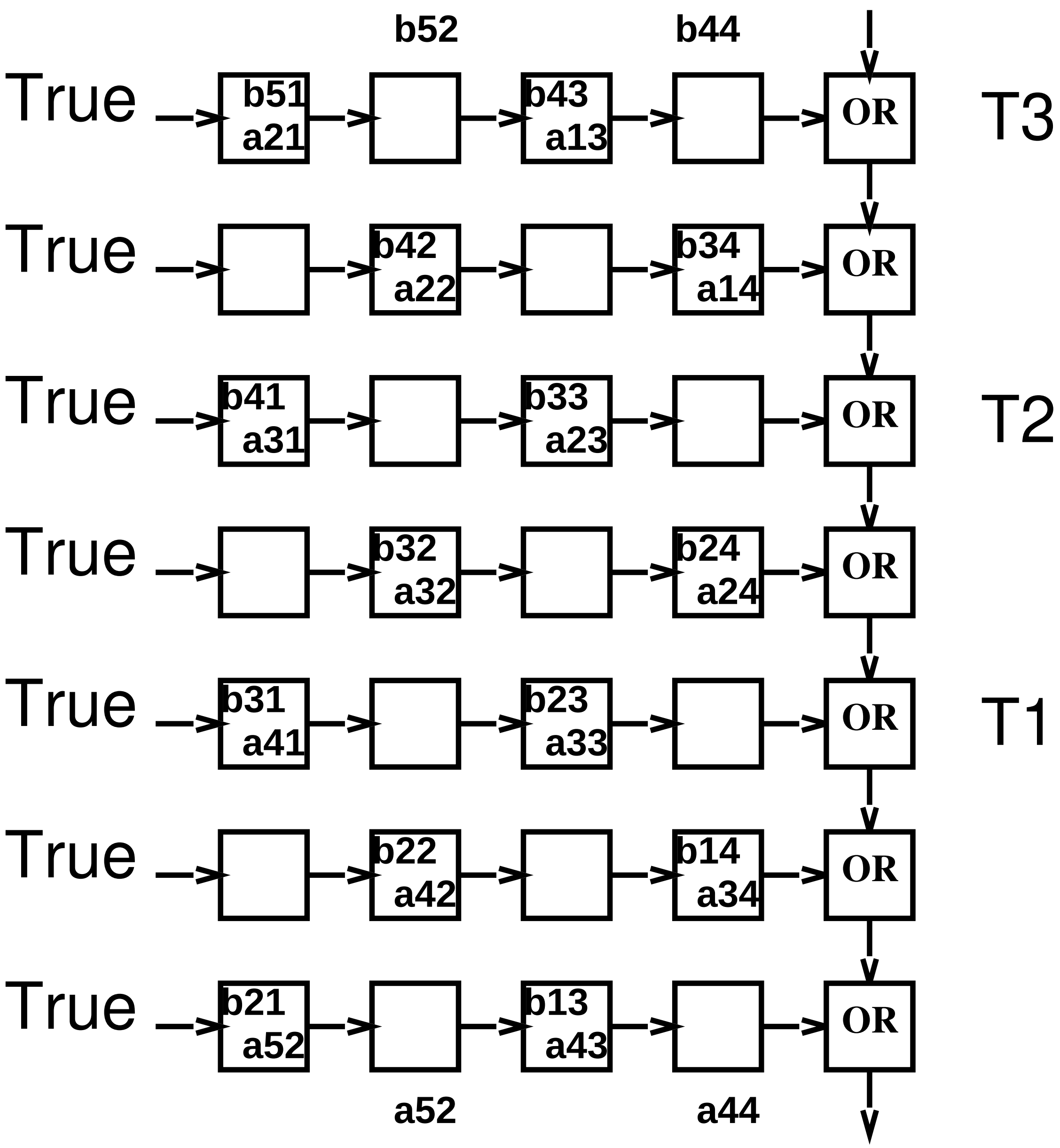
Cell: $t_{out} = t_{in} \wedge a_{in} \oplus b_{in}$



DB: Intersect



DB: Intersect



DB: Intersect$\rightarrow$ Subtract

Subtract: $B - A$ – all tuples of B not in A.

- **Same operation, keep tuples for which Ti is false.**

DB: Intersect$\rightarrow$ Unique

Unique: A – remove duplicate elements in multirelation A

- **Intersect A with A**
- **Initialize upper triangle false so keep one copy.**

	A1	A2	A3
A1	false	false	false
A2	T21	false	false
A3	T31	T32	false

DB: Unique$\rightarrow$ Union

Union: $A \cup B$ – one copy of all tuples in A and B

- **Concatenate A and B**
- **Use Unique to remove duplicates**

DB: Unique$\rightarrow$ Projection

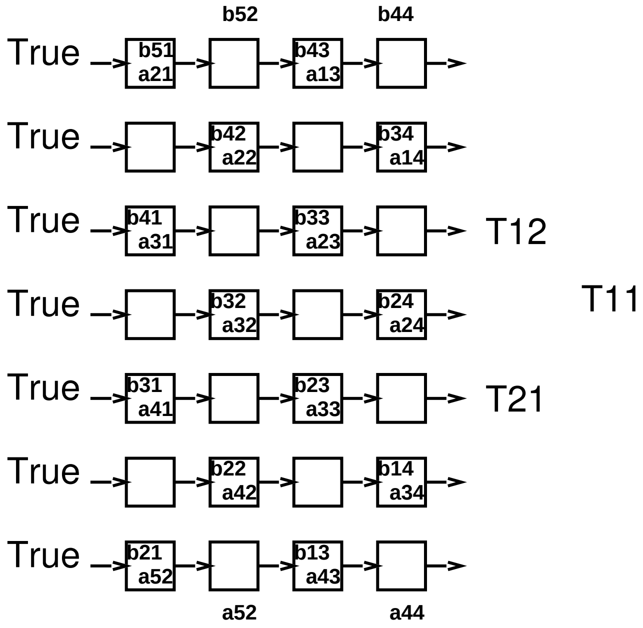
Projection: $A \leftarrow A'$ – collapse A by removing select fields of every tuple

- **Sample fields in A'**
- **Use Unique to remove duplicates**

DB: Match Tuples$\rightarrow$ Join

Join: $AJ_{C_A,C_B}B$ – where columns C_A in A intersect columns C_B in B, concatenate tuple A_i and B_j.

- **Match C_A of A with C_B of B**
- **Keep all Ti,j**
- **Filter i,j for which $Ti,j =$ true**
- **Construct join from matched pairs**



Claim: Typically, $|AJ_{C_A,C_B}B| << |A||B|$

DB Ops

- $O(n)$ data
- Operations require $O(n^2)$ computation
- Work on $O(n)$ processors

Data unsorted.

Why not sort data first?

- $O(n \log(n))$ to sort
- After sort, operations are $O(n)$

- Sort need more bandwidth?
- if $|A| << |B|$, cheaper not to sort B?

DB Ops

- Can use (SPLASH) "unidirectional" stream equivalent to increase comparison efficiency

Suggest typically:

- 1500 bit tuples
- 10,000 records in a relation

- XC4062 $\rightarrow$ 48$\times$48 CLBs
- $\approx$ 1 CLB per bit-compare
- 31 form a 48$\times$1500 band of array
- $\approx$200$\times$30 for a 1500$\times$10,000 intersection array
- $\approx$20Tλ^2

Operations Decomposable

- **All operations can be decomposed into bands operating on smaller array**
- **Matrix-like Ops**: $k \times k$ submatrix
- **Intersect**: Segment one relation into array-size pieces

	B1	B2	B3	B4
A1	T11	T12	T13	T14
A2	T21	T22	T23	T24
A3	T31	T32	T33	T34
A4	T41	T42	T43	T44

Review

Today

- $O(n^3)$ dynamic programming problem in $2n$ time on $n^2/2$ cells
- Transitive closure (shortest path) in $5n$ on n^2 cells (throughput of one every n cycles achievable)
- Priority queue with constant time insert/extract, $O(n)$ cells
- Union, Intersect, Project, Join in $O(n)$ time on $O(n)$ cells
- Decomposition to use smaller available arrays

Review

Take Away

- Stylized form $\rightarrow$ good for deriving suitable spatial computations
- Pays attention to interconnect costs
- Systolic solutions to common tasks $\rightarrow$ bag of tricks
- Rich body of theory accompanying