CS137:
Electronic Design Automation

Andre' DeHon

Day 5: April 15, 2002

Systolic Algorithms

Today

- Systolic Style
- Systolic Algorithms and Structures
  - Filters
  - Matching
  - Matrix Multiply and Transitive Closure
  - Dynamic Programming
  - Priority Queue
- Decomposition

Systolic

- **Systole** a rhythmically recurrent contraction;

esp: the contraction of the heart by which the blood is forced onward and the circulation kept up --- *Merriam Webster*

- **Systolic architectures** are characterized by regular, rhythmic, data flow through numerous processing elements

Motivation

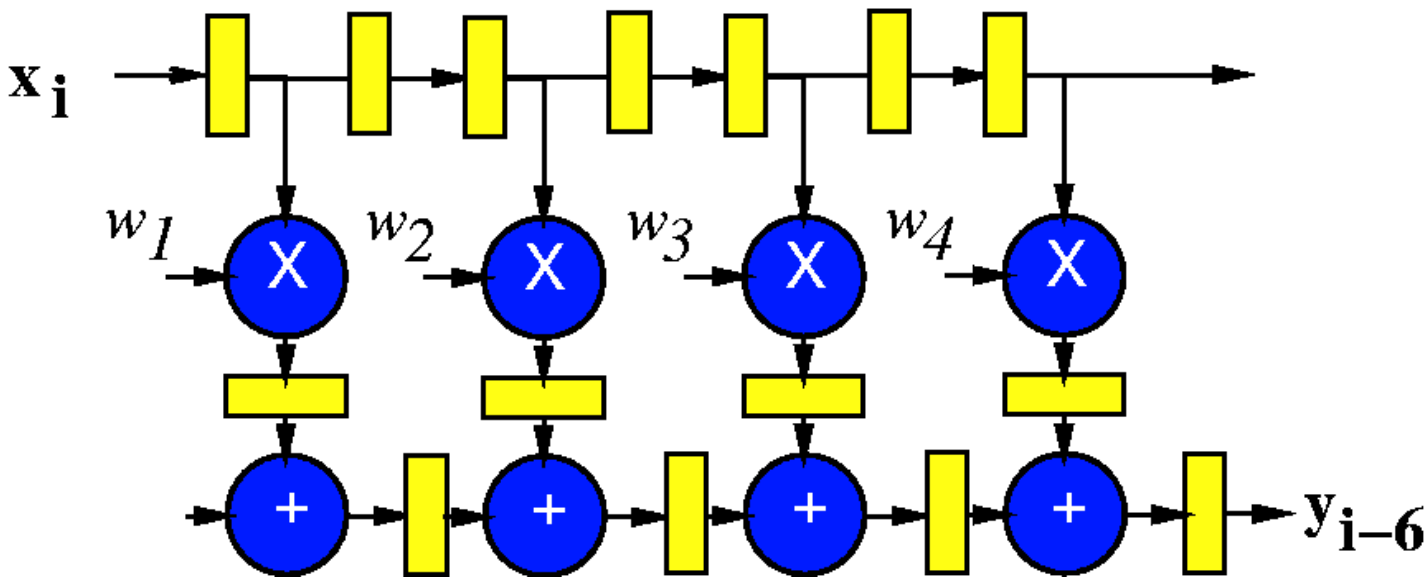
- Accelerate computationally bound problems
- For each data item read from memory
  - Use it multiple times
  - Perform many operations
  - Compute on intermediate results before discarding/writing back to memory

Philosophy

- Match VLSI Costs
  - Exploit concurrency
  - Wires take up space
  - Long wires take time
  - Minimize design effort
    - Exploit regularity
    - Scale with problem size and/or perf. requirements

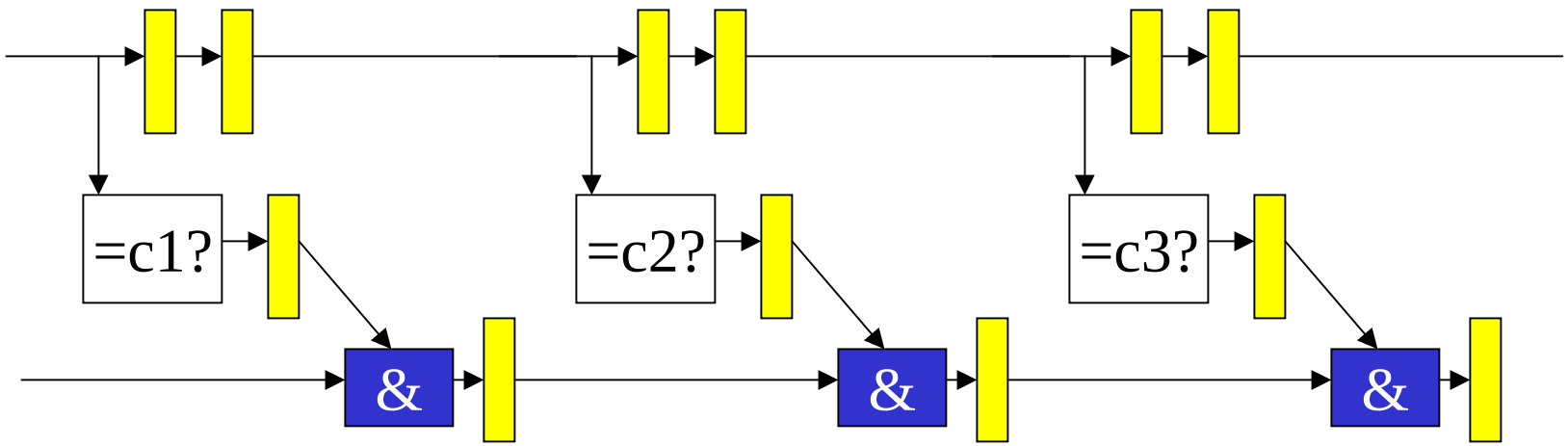
Systolic Filter

- Finite-Impulse Response Filter
  - y[n]=w1*x[n]+w2*x[n-1]+w3*x[n-3]…
  - weighted sum of k previous samples



Simple String Matching

- Stream text past set of matches



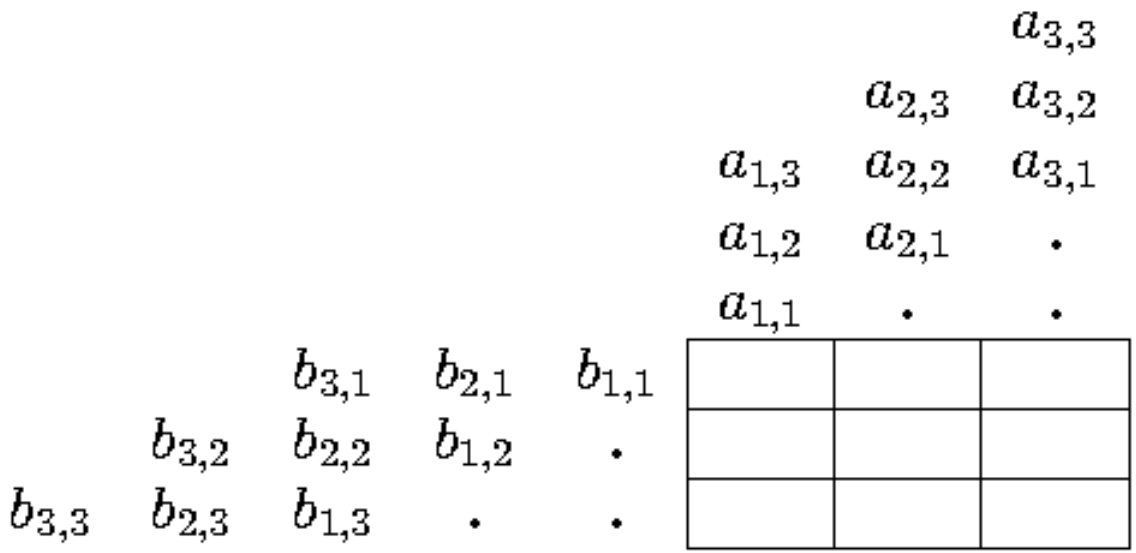
Matrix Multiply

- C = B * A
- $C[I,j]=\Sigma_k A[i,k]*B[k,j]$

- NxN matrix
- Takes N^3 multiplications

Systolic Matrix Multiply

- Time N with N^2 cells

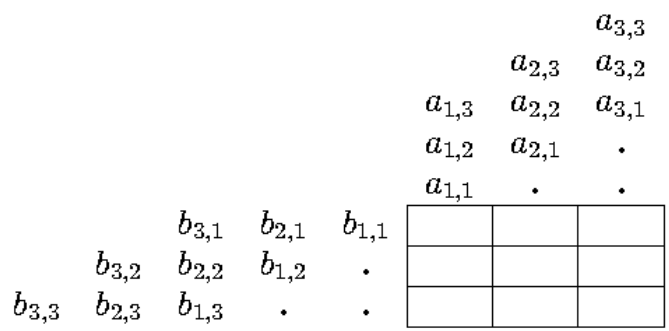


Matrix Multiply Cell

CELL behavior

- If (reset)
  - Sum=0
- Else
  - Sum+=Xin*Yin
- Xout=Xin
- Yout=Yin
- ;; plus output data

- Tag reset ahead of data row/column
- Tag end of data



Transitive Closure

- Same basic shape
- $C[I,j]=a[I,j]+OR_k\ (a[i,k]\&a[k,j])$
- Replace
  - Sum with OR
  - Multiply with AND

Dynamic Programming

- Two Examples
  - SPLASH sequence matching/edit distances
    - $O(N^2)$ operation in O(N) time with O(N) hardware
  - CMU parenthesis matching
    - $O(N^3)$ operation in O(N) time with $O(N^2)$ hardware

Sequence Matching

- Find *edit distance* between two strings
  - E.g.
    - Insert cost 1
    - Delete cost 1
    - Replace cost 2
    - Match 0

Edit Example

- SHMOO
  - Add E (cost 1)
- SHMOOE
  - Remove M (cost 1 + 1=2)
- SHOOE
  - Replace O with R (cost 2+2=4)
- SHORE

Dynamic Programming

- Build a table representing string prefixes
- Fill in costs

		S	H	O	R	E
S						
H						
M						
O						
O						

Local Move Costs

- D(0,0) = 0
- D(i,0)=D(i-1,0)+Delete(Si)
- D(0,j)=D(0,j-1)+Insert(Tj)
- D(i,j)=min
  - D(i-1,j)+Delete(Si)
  - D(i,j-1)+Insert(Tj)
  - D(i-1,j-1)+Replace(Si,Tj)

		S	H	O	R	E
S						
H						
M						
O						
O						

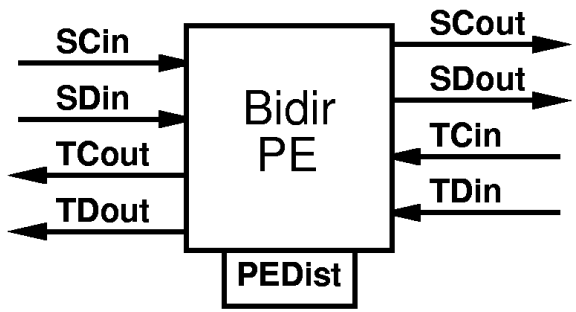
Edit Distance Table

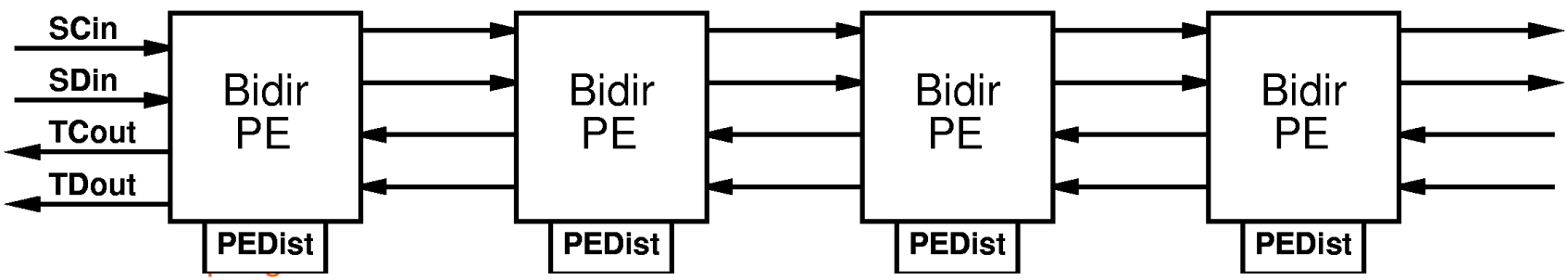
		S	H	O	R	E
	0	1	2	3	4	5
S	1	0	1	2	3	4
H	2	1	0	1	2	3
M	3	2	1	2	3	4
O	4	3	2	1	2	3
O	5	4	3	2	3	4

Systolic Array

- Feed Strings from opposite ends
- Compute along diagonals

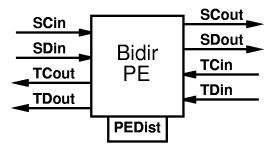
		S	H	O	R	E
S						
H						
M						
O						
O						

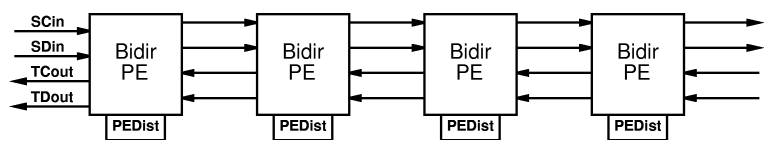




Systolic Array

- PEDist = min
  - PEDist + Cost(Substitute(TCin,SCin))
  - TDin+Cost(Delete(SCin))
  - SDin+Cost(Insert(TCin))
- SDout=TDout=PEDist
- ….plus details for edge cases





In Operation

		S	H	O	R	E
	0	1	2	3	4	5
S	1	0	1	2	3	4
H	2	1	0	1	2	3
M	3	2	1	2	3	4
O	4	3	2	1	2	3
O	5	4	3	2	3	4

TC	**O**		**O**		**M**		**H**		**S**										
TD	5		4		3		2		1										
TDout	5		4		3		2		1										
SC											**S**		**H**		**O**		**R**		**E**
SD											1		2		3		4		5
SDout											1		2		3		4		5
PEDist	5	4	4	3	3	2	2	1	1	0	1	1	2	2	3	3	4	4	5
TC		**O**		**O**		**M**		**H**		**S**									
TD		5		4		3		2		1									
TDout		5		4		3		2		0									
SC										**S**		**H**		**O**		**R**		**E**	
SD										1		2		3		4		5	
SDout										0		2		3		4		5	
PEDist	5	5	4	4	3	3	2	2	1	0	1	2	2	3	3	4	4	5	5
TC			**O**		**O**		**M**		**H**		**S**								
TD			5		4		3		2		0								
TDout			5		4		3		1		1								
SC									**S**		**H**		**O**		**R**		**E**		
SD									0		2		3		4		5		
SDout									1		1		3		4		5		
PEDist	5	5	5	4	4	3	3	2	1	0	1	2	3	3	4	4	5	5	5

In Operation (Cont.)

		S	H	O	R	E
	0	1	2	3	4	5
S	1	0	1	2	3	4
H	2	1	0	1	2	3
M	3	2	1	2	3	4
O	4	3	2	1	2	3
O	5	4	3	2	3	4

TC				**O**		**O**		**M**		**H**		**S**						
TD				5		4		3		1		1						
TDout				5		4		2		0		2						
SC								**S**		**H**		**O**		**R**		**E**		
SD								1		1		3		4		5		
SDout								2		0		2		4		5		
PEDist	5	5	5	5	4	4	3	2	1	0	1	2	3	4	4	5	5	5	5

TC					**O**		**O**		**M**		**H**		**S**					
TD					5		4		2		0		2					
TDout					5		3		1		1		3					
SC							**S**		**H**		**O**		**R**		**E**			
SD							2		0		2		4		5			
SDout							3		1		1		3		5			
PEDist	5	5	5	5	5	4	3	2	1	0	1	2	3	4	5	5	5	5	5

TC						**O**		**O**		**M**		**H**		**S**				
TD						5		3		1		1		3				
TDout						4		2		2		2		4				
SC						**S**		**H**		**O**		**R**		**E**				
SD						3		1		1		3		5				
SDout						4		2		2		2		4				
PEDist	5	5	5	5	5	4	3	2	1	2	1	2	3	4	5	5	5	5	5

In Operation (Cont.)

		S	H	O	R	E
	0	1	2	3	4	5
S	1	0	1	2	3	4
H	2	1	0	1	2	3
M	3	2	1	2	3	4
O	4	3	2	1	2	3
O	5	4	3	2	3	4

TC							**O**		**O**		**M**		**H**		**S**				
TD							4		2		2		2		4				
TDout							3		2		3		3		4				
SC					**S**		**H**		**O**		**R**		**E**						
SD					4		2		2		2		4						
SDout					4		3		2		3		3						
PEDist	5	5	5	5	4	4	3	2	2	2	3	2	3	4	4	5	5	5	5
TC								**O**		**O**		**M**		**H**		**S**			
TD								3		2		3		3		4			
TDout								2		3		4		3		4			
SC				**S**		**H**		**O**		**R**		**E**							
SD				4		3		2		3		3							
SDout				4		3		2		3		4							
PEDist	5	5	5	4	4	3	3	2	2	3	3	4	3	3	4	4	5	5	5
TC									**O**		**O**		**M**		**H**		**S**		
TD									2		3		4		3		4		
TDout									3		4		4		3		4		
SC			**S**		**H**		**O**		**R**		**E**								
SD			4		3		2		3		4								
SDout			4		3		2		3		4								
PEDist	5	5	4	4	3	3	2	2	3	3	4	4	4	3	3	4	4	5	5

[inconsistency here]

In Operation (Cont.)

		S	H	O	R	E
	0	1	2	3	4	5
S	1	0	1	2	3	4
H	2	1	0	1	2	3
M	3	2	1	2	3	4
O	4	3	2	1	2	3
O	5	4	3	2	3	4

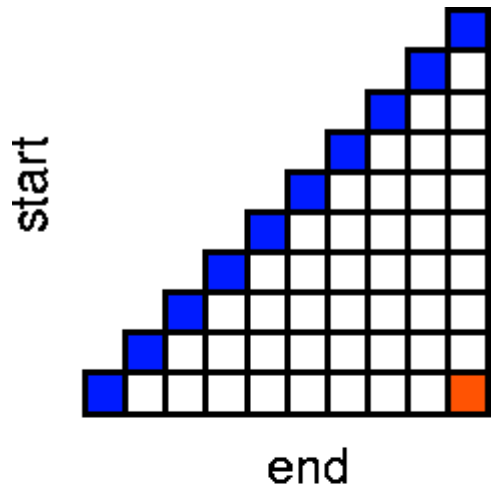
TC										**O**		**O**		**M**		**H**		**S**	
TD										3		4		4		3		4	
TDout										4		4		4		3		4	
SC		**S**		**H**		**O**		**R**		**E**									
SD		4		3		2		3		4									
SDout		4		3		2		3		4									
PEDist	5	4	4	3	3	2	2	3	3	4	4	4	4	4	3	3	4	4	5

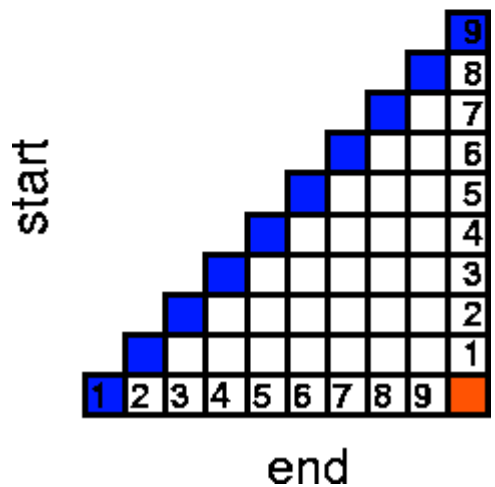
Edit Distance Table

		S	H	O	R	E
	0	1	2	3	4	5
S	1	0	1	2	3	4
H	2	1	0	1	2	3
M	3	2	1	2	3	4
O	4	3	2	1	2	3
O	5	4	3	2	3	4

Parenthesis Matching

- Similar
- But compute from all breaks across a diagonal
  - Not just nearest neighbor
- Hence extra O(N)



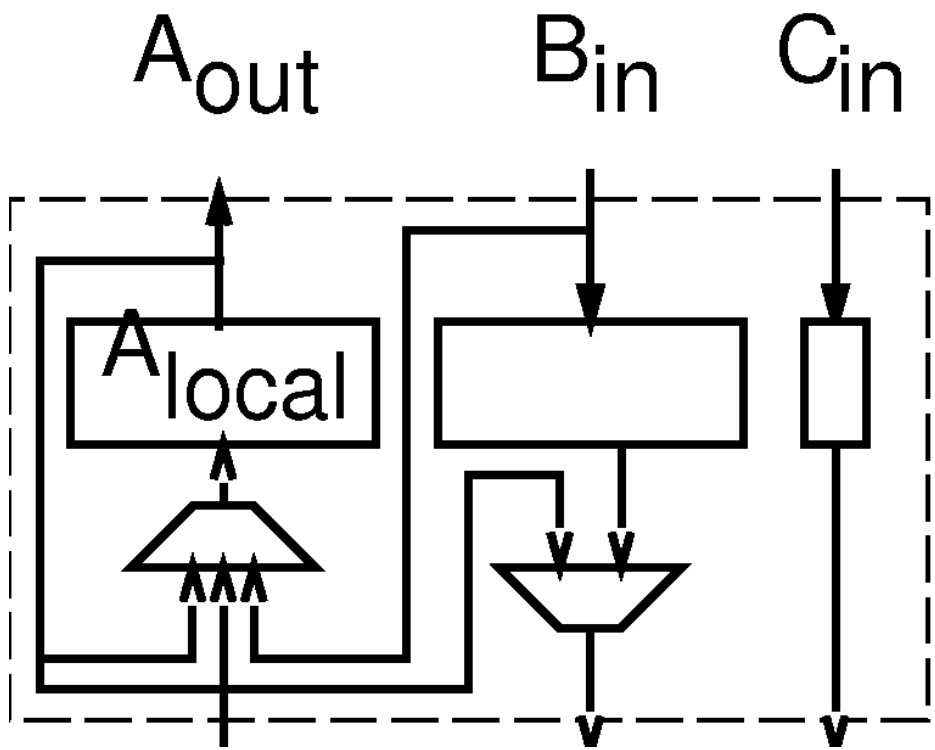


Priority Queue

- Insert top
- Extract Largest
- W/ O(N) cells
- O(1) Extract

Priority Queue Cell

- If (Cin=insert)
  Alocal←largest
  Bout←smallest
- If (Cin=extract)
  Alocal←Ain
  Bout←Bin
- Cout←Cin



Decomposition