Systolic Computing
February 2, 1998

- Relevant Talks
- Lab2
- Reading For Wednesday
- Systolic Architectures
- Implementations

Talks

- Today: Maya Gokhale
- 2/4 noon D-level: Managing Pipelined reconfigurable FPGAs
- 2/16: Saman Amarasinghe on RAW

Lab 2

- Handout delayed til Wednesday
- Implement various kernels on PipeRench and X4000
  - CVHASM
  - dp
  - Verilog/VHDL

PipeRench Tutorial

- Please submit an extended review
- Write and Run a simple function:
  Running Average of past 4 samples

Systolic Computing

- Definition

sys·to·le (sĭs′tə-lē) *noun*
The rhythmic contraction of the heart, especially of the ventricles, by which blood is driven through the aorta and pulmonary artery after each dilation or diastole.

[Greek *sustolē*, contraction, from *sustellein*, to contract. See systaltic.]
— sys·tol′ic (sĭ-stŏl′ĭk) *adjective*

American Heritage Dictionary

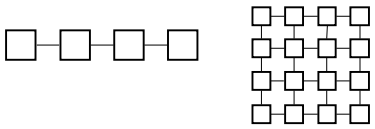
Data flows from memory in a rhythmic fashion, passing through many processing elements before it returns to memory.

H.T.Kung

Systolic Computing

- Definition

A set of simple processing elements with regular and local connections which takes external inputs and processes them in a predetermined manner in a pipelined fashion.

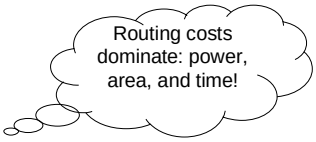


Motivation

- Effectively utilize VLSI
- Reduce “Von Neumann Bottleneck”
- Target compute-intensive applications
- Reduce design cost:
  - Simple
  - Regular
- Exploit Concurrency

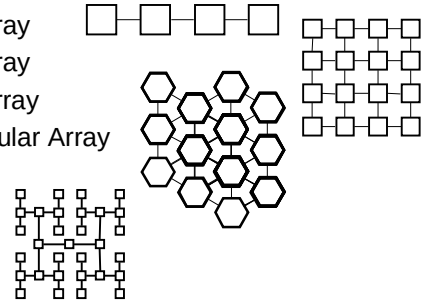
Using VLSI Effectively

- Replicate simple cells
- Local Communication ⇒
  - Short wires
    - small delay
    - low clock skew
    - small drivers
    - less area
  - Scalable
- Small number of I/Os



Regular Interconnect

- 1-d Array
- 2-d Array
- Hex Array
- Triangular Array
- Tree



Eliminating the VN Bottleneck

- Process each input multiple times.
- Keep partial results in the PEs.

- Does this still present a win today?
  - Large $
  - Many registers

Balancing I/O and Compute

- Can’t go faster than the data arrives
- Reduce bandwidth requirements
- Choose applications well!
- Choose algorithms correctly!

Exploiting Concurrency

- Large number of simple PEs
- Manage without instruction store
- Methods:
  - Pipelining
  - SIMD/MIMD
  - Vector
- Limits application space. Severely?

Systolic Architectural Model

- Simple PEs
- Regular and local interconnect
- Pipeline between PEs
- I/O at boundary

Characteristics of best RC Designs.

Systolic Is Good RC Starting Point.

- Original motivation: Custom silicon for an application.
- Model Architecture is efficient for RC
- Target Algorithms match
- Well developed theory
- Compilation technology well studied

Was Systolic Computing ahead of its time?

One Big Difference

- Kung etals approach fabricated silicon.
- RC compiles to fabric.

Kung's PEs must be "general purpose."

Mapping Approach

- Allocate PEs
- Schedule computation
  - schedule PEs *structure*
  - schedule data flow *driver*
- Optimize

Various Possible Implementations

- Broadcast
  - Y moves
  - W moves
- Fan-In
- Systolic
  - Bidirectional
    - Y in place
    - X in place
  - Unidirectional
    - Y in place
    - W in place

Running Example: Convolution

```
for (i=1; i ≤ n; i++)
  for (j=1; j ≤ k; j++)
    y_i += w_j * x_{i+j-1}
```

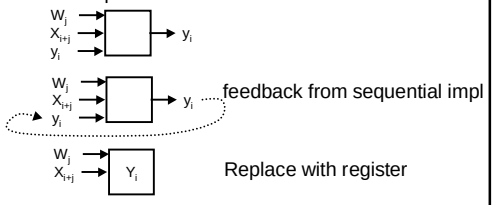

Bag of Tricks

- Preload-repeated-value
- Replace-feedback-with-register
- Internalize-data-flow
- Broadcast-common-input
- Propagate-common-input
- Retime-to-eliminate-broadcasting

Bogus Attempt at Systolic FIR

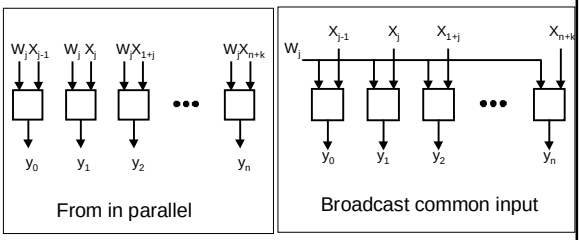
```
for i=1 to n in parallel
   for j=1 to k in place
      y_i += w_j * x_i+j-1
```

Inner Loop:



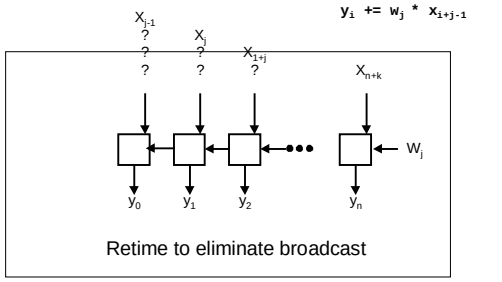
Bogus Attempt: Outer Loop

```
for i=1 to n in parallel
   for j=1 to k in place
      y_i += w_j * x_i+j-1
```



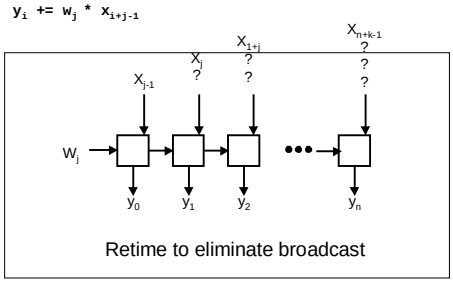
Bogus Attempt: Outer Loop-2

```
for i=1 to n in parallel
   for j=1 to k in place
      y_i += w_j * x_i+j-1
```



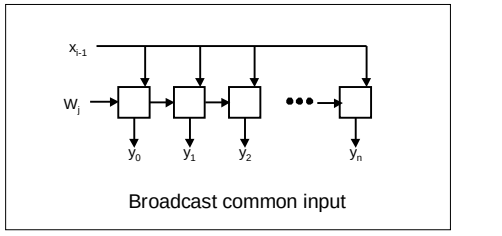
Bogus Attempt: Outer Loop-2a

```
for i=1 to n in parallel
   for j=1 to k in place
      y_i += w_j * x_i+j-1
```



Bogus Attempt: Outer Loop-3

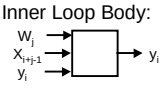
```
for i=1 to n in parallel
   for j=1 to k in place
      y_i += w_j * x_i+j-1
```



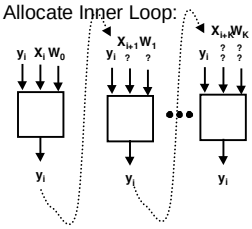
Attempt at Systolic FIR

```
for i=1 to n in place
   for j=1 to k in parallel
      y_i += w_j * x_i+j-1
```

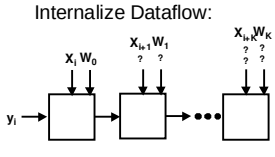
Allocate Inner Loop Body:



Allocate Inner Loop:



Internalize Dataflow:

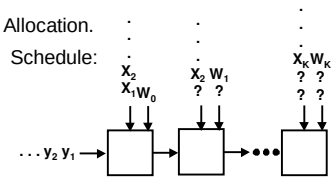


Outer Loop

```
for i=1 to n in place
   for j=1 to k in parallel
      y_i += w_j * x_{i+j-1}
```

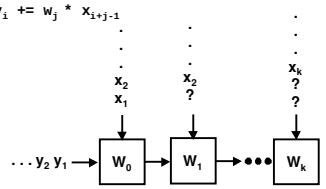
Allocation.

Schedule:



Optimize Outer Loop
Preload-repeated Value

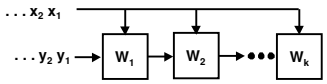
```
for i=1 to n in place
   for j=0 to k in parallel
      y_i += w_j * x_{i+j-1}
```



Optimize Outer Loop
Broadcast Common Value

```
for i=1 to n in place
   for j=1 to k in parallel
      y_i += w_j * x_{i+j-1}
```

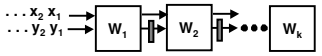
B1 From [Kung82]



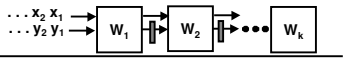
Optimize Outer Loop
Retime to Eliminate Broadcast

```
for i=1 to n in place
   for j=1 to k in parallel
      y_i += w_j * x_{i+j-1}
```

Similar to W2 From [Kung82]

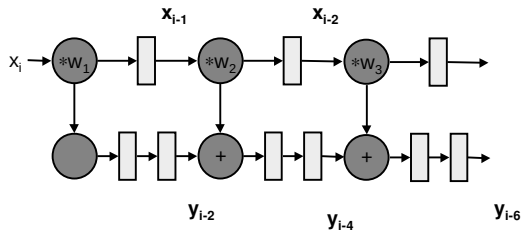


How It Works

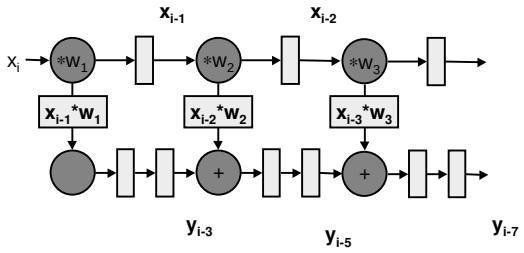


y_1			
x_1			
y_2	y_1		
x_2	x_1		
y_3	y_2	y_1	
x_3	x_2		x_1
y_4	y_3	y_2	y_1
x_4	x_3		x_2
y_5	y_4	y_3	y_2	y_1
x_5	x_4			x_3

FIR details



FIR details



FIR Summary

- Sequential:
  - Memory Bandwidth per output: 3k+1 each x k times. k w's. plus one y.
  - 1 result o(k) cycles
  - o(1) hardware
- Systolic:
  - Memory Bandwidth per output: 2 1 x + 1 y.
  - 1 result every cycle - o(1)
  - o(k) hardware