CprE / ComS 583 Reconfigurable Computing

Prof. Joseph Zambreno
Department of Electrical and Computer Engineering
Iowa State University

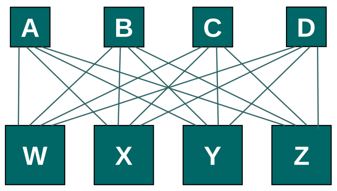
Lecture #10 – Systolic Computing

Quick Points

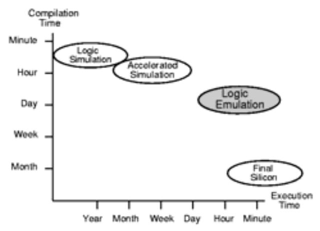
- HW #2 due Tuesday at 12:00pm
  - Any questions?
- HW #3 (9/28 – 10/17), project proposal (9/26 – 10/5), midterm exam (10/12) coming soon

Recap – Multi-FPGA Systems

- Crossbar topology:
  - Devices A-D are routing only
  - Gives predictable performance
  - Potential waste of resources for near-neighbor connections

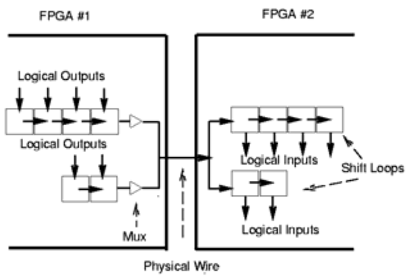


Recap – Logic Emulation



- Emulation takes a sizable amount of resources
- Compilation time can be large due to FPGA compiles

Recap – Virtual Wires



- Overcome pin limitations by multiplexing pins and signals
- Schedule when communication will take place

Outline

- Recap
- Introduction and Motivation
- Common Systolic Structures
- Algorithmic Mapping
- Mapping Examples
  - Finite impulse response
  - Matrix-vector product
  - Banded matrix-vector product
  - Banded matrix multiplication

Systolic Computing

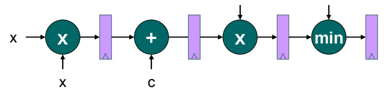
sys·to·le (sĭs'tə-lē) *n.* – the rhythmic contraction of the heart, especially of the ventricles, by which blood is driven through the aorta and pulmonary artery after each dilation or diastole

[Greek *systolē,* from *systellein* to contract, from *syn-* + *stellein* to send]
– **sys·tol·ic** (sĭs-tŏl'ĭk) *adj.*

Data flows from memory in a rhythmic fashion, passing through many processing elements before it returns to memory.
[Kung, 1982]

Systolic Architectures

- Goal – general methodology for mapping computations into hardware (spatial computing) structures
- Composition:
  - Simple compute cells (e.g. add, sub, max, min)
  - Regular interconnect pattern
  - Pipelined communication between cells
  - I/O at boundaries



Motivation

- Effectively utilize VLSI
- Reduce “Von Neumann Bottleneck”
- Target compute-intensive applications
- Reduce design cost
  - Simplicity
  - Regularity
- Exploit concurrency
- Local communication
  - Short wires (small delay, less area)
  - Scalable

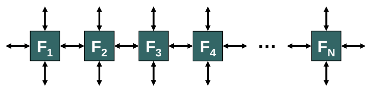
Why Study?

- Original motivation – specialized accelerator for an application
- Model/goals is a close match to reconfigurable computing
- Target algorithms match
- Well-developed theory, techniques, and solutions

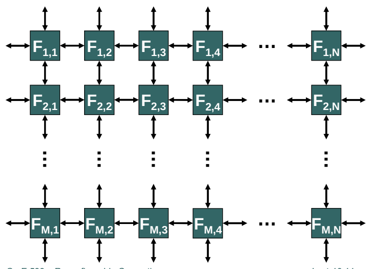
- One big difference – Kung’s approach targeted custom silicon (not a reconfigurable fabric)
  - Compute elements needed to be more general

Common Systolic Structures

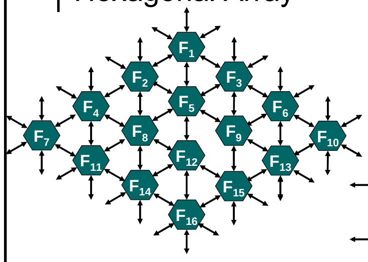
- One-dimensional linear array



- Two-dimensional mesh

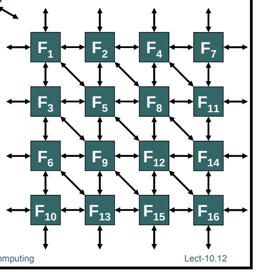


Hexagonal Array

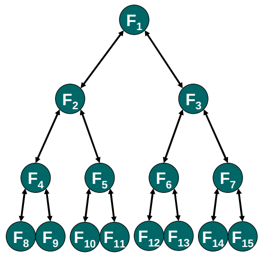


Communicates with six nearest neighbors

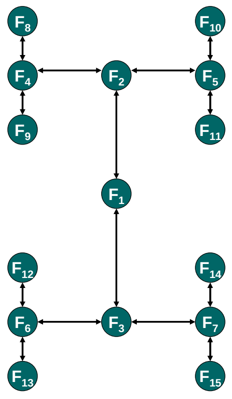
Squared-up representation



Binary Tree



H-Tree Representation



Mapping Approach

- Allocate PEs
- Schedule computation
  - Schedule PEs
  - Schedule data flow
- Optimize
- Available Transformations:
  - Preload repeated values
  - Replace feedback loops with registers
  - Internalize data flow
  - Broadcast common input

Example – Finite Impulse Response

- A Finite Impulse Response (FIR) filter is a type of digital filter
  - Finite – response to an impulse eventually settles to zero
  - Requires no feedback

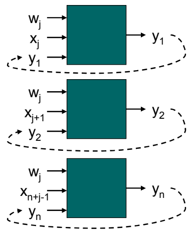
$$y_i = w_1 \cdot x_i + w_2 \cdot x_{i+1} + \cdots + w_k \cdot x_{i+k-1}$$
$$= \sum_{j=1}^{k} w_i \cdot x_{i+j-1}$$

```
for (i=1; i<=n; i++)
  for (j=1; j <=k; j++)
    y[i] += w[j] * x[i+j-1];
```

FIR Attempt #1

- Parallelize the outer loop

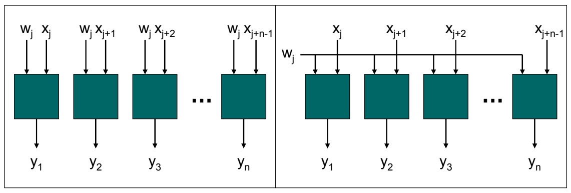
```
for (i=1; i<=n; i++)     in parallel
  for (j=1; j <=k; j++) sequential
    y[i] += w[j] * x[i+j-1];
```



FIR Atttempt #1 (cont.)

- Broadcast common inputs

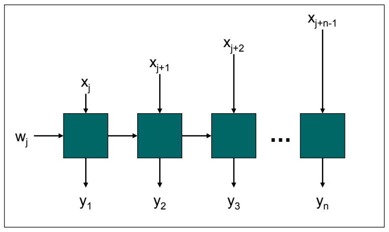
```
for (i=1; i<=n; i++)     in parallel
  for (j=1; j <=k; j++) sequential
    y[i] += w[j] * x[i+j-1];
```



FIR Attempt #1 (cont.)

- Retime to eliminate broadcast

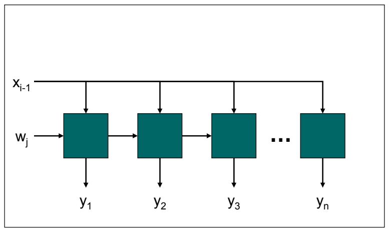
```
for (i=1; i<=n; i++)     in parallel
  for (j=1; j <=k; j++) sequential
    y[i] += w[j] * x[i+j-1];
```



FIR Attempt #1 (cont.)

- Broadcast common values

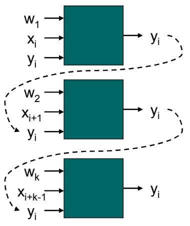
```
for (i=1; i<=n; i++)    in parallel
  for (j=1; j <=k; j++) sequential
    y[i] += w[j] * x[i+j-1];
```



FIR Attempt #2

- Parallelize the inner loop

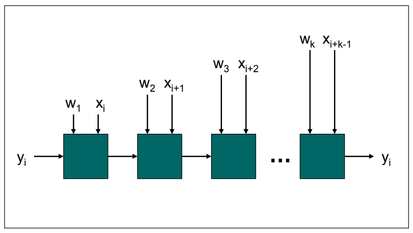
```
for (i=1; i<=n; i++)    sequential
  for (j=1; j <=k; j++) in parallel
    y[i] += w[j] * x[i+j-1];
```



FIR Attempt #2 (cont.)

- Internalize data flow

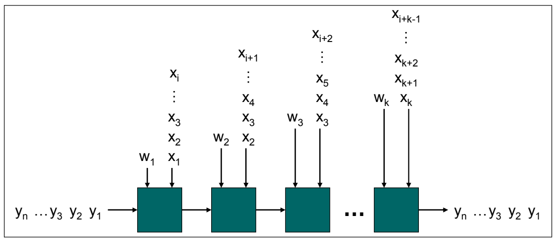
```
for (i=1; i<=n; i++)    sequential
  for (j=1; j <=k; j++) in parallel
    y[i] += w[j] * x[i+j-1];
```



FIR Attempt #2 (cont.)

- Allocation schedule

```
for (i=1; i<=n; i++)    sequential
  for (j=1; j <=k; j++) in parallel
    y[i] += w[j] * x[i+j-1];
```

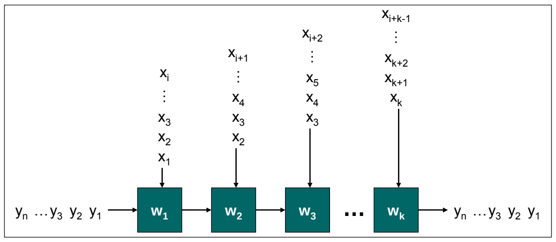


x_i ⋮ x_3 x_2 x_1
x_{i+1} ⋮ x_4 x_3 x_2
x_{i+2} ⋮ x_5 x_4 x_3
x_{i+k-1} ⋮ x_{k+2} x_{k+1} x_k
w_1
w_2
w_3
w_k
$y_n \cdots y_3\ y_2\ y_1$
…
$y_n \cdots y_3\ y_2\ y_1$

FIR Attempt #2 (cont.)

- Preload repeated values

```
for (i=1; i<=n; i++)    sequential
  for (j=1; j <=k; j++) in parallel
    y[i] += w[j] * x[i+j-1];
```

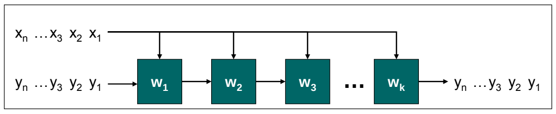


x_i ⋮ x_3 x_2 x_1
x_{i+1} ⋮ x_4 x_3 x_2
x_{i+2} ⋮ x_5 x_4 x_3
x_{i+k-1} ⋮ x_{k+2} x_{k+1} x_k
$y_n \cdots y_3\ y_2\ y_1$
w_1
w_2
w_3
…
w_k
$y_n \cdots y_3\ y_2\ y_1$

FIR Attempt #2 (cont.)

- Broadcast common values

```
for (i=1; i<=n; i++)    sequential
  for (j=1; j <=k; j++) in parallel
    y[i] += w[j] * x[i+j-1];
```

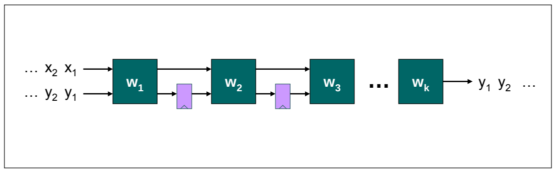


$x_n \cdots x_3\ x_2\ x_1$
$y_n \cdots y_3\ y_2\ y_1$
w_1
w_2
w_3
…
w_k
$y_n \cdots y_3\ y_2\ y_1$

FIR Attempt #2 (cont.)

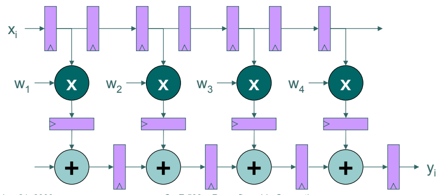
- Retime to eliminate broadcast

```
for (i=1; i<=n; i++)     sequential
  for (j=1; j <=k; j++)  in parallel
    y[i] += w[j] * x[i+j-1];
```



FIR Summary

- Sequential
  - Memory bandwidth per output – 2*k*+1
  - *O*(*k*) cycles per output
  - *O*(1) hardware
- Systolic
  - Memory bandwidth per output – 2
  - *O*(1) cycles per output
  - *O*(*k*) hardware



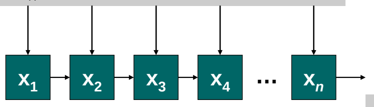
Example – Matrix-Vector Product

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

```
for (i=1; i<=n; i++)
  for (j=1; j<=n; j++)
    y[i] += a[i][j] * x[j];
```

Matrix-Vector Product (cont.)

t = 4	a_{41}	a_{23}	a_{23}	a_{14}	–
t = 3	a_{31}	a_{22}	a_{13}	–	–
t = 2	a_{21}	a_{12}	–	–	–
t = 1	a_{11}	–	–	–	–



y_1	*t* = *n*
y_2	*t* = *n*+1
y_3	*t* = *n*+2
y_4	*t* = *n*+3

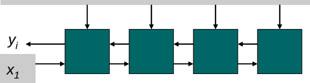
Banded Matrix-Vector Product

$$p\left\{\begin{bmatrix} \overbrace{a_{11} \quad a_{12}}^{q} & & & & 0 \\ a_{21} \quad a_{22} & a_{23} & & & \\ a_{31} \quad a_{32} & a_{33} & a_{34} & & \\ \quad\;\; a_{42} & a_{43} & a_{44} & a_{45} & \\ & & & \ddots & \\ 0 & & & & \end{bmatrix}\right. \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ \vdots \\ y_n \end{bmatrix}$$

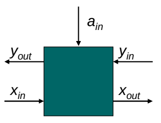
```
for (i=1; i<=n; i++)
  for (j=1; j<=p+q-1; j++)
    y[i] += a[i][j-q-i] * x[j];
```

Banded Matrix-Vector Product (cont.)

t = 5	a_{23}	–	a_{32}	–
t = 4	–	a_{22}	–	a_{31}
t = 3	a_{12}	–	a_{21}	–
t = 2	–	a_{11}	–	–
t = 1	–	–	–	–



t = 1	x_1
t = 2	–
t = 3	x_2
t = 4	–
t = 5	x_3

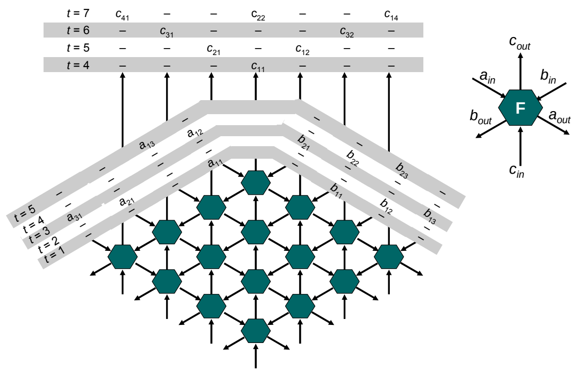


Banded Matrix Multiplication

$$\begin{bmatrix} a_{11} & a_{12} & & & & 0 \\ a_{21} & a_{22} & a_{23} & & & \\ a_{31} & a_{32} & a_{33} & a_{34} & & \\ & a_{42} & a_{43} & a_{44} & a_{45} & \\ & & & & \ddots & \\ 0 & & & & & \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & & & & 0 \\ b_{21} & b_{22} & b_{23} & & & \\ b_{31} & b_{32} & b_{33} & b_{34} & & \\ & b_{42} & b_{43} & b_{44} & b_{45} & \\ & & & & \ddots & \\ 0 & & & & & \end{bmatrix}$$

$$= \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} & & 0 \\ c_{21} & c_{22} & c_{23} & c_{24} & c_{25} & \\ c_{31} & c_{32} & c_{33} & c_{34} & c_{35} & \\ c_{41} & c_{42} & c_{43} & c_{44} & c_{45} & \\ & c_{52} & c_{53} & c_{54} & c_{55} & \ddots \\ 0 & & & & & \end{bmatrix}$$

Banded Matrix Multiplication (cont.)



Summary

- Systolic structures are good for computation-bound problems
- Models costs in VLSI systems
  - Minimize number of memory accesses
  - Emphasize local interconnections (long wires are bad)
- Candidate algorithms
  - Makes multiple use of input data (ex: *n* inputs, $O(n^3)$ computations
  - Concurrency
  - Simple control flow, simple processing elements