Quick Points
- HW #3 coming out today
  - Due Tuesday, October 17 (midnight)
    - Systolic computing structures
    - Systolic mapping
    - Logic partitioning
    - FPGA synthesis

Project Proposals
- Due Sunday, 10/8 at midnight
  - Purpose – to provide a background and overview of the project
  - Goal – allow me to understand what you are intending to do
- Project topic:
  - Perform an in-depth exploration of some area of reconfigurable computing
  - Whatever topic you choose, you must include a strong experimental element in your project
  - Work in groups of 2+ (3 if very lofty proposal)

Project Proposals (cont.)
- Suggested structure [3-4 pages, IEEE conf. format]
  - Introduction – what is the context for this work? What problem are you trying to address? Why is it interesting/challenging?
  - Prior work – what is the related work? How does your work differ from these? (5-10 references)
  - Approach – how are you going to tackle the problem? What tools and methodologies do you intend on using? What experiments do you intend on running?
  - Expected results – what do you expect the outcome of your project to be? What are the deliverables? How do you intend on presenting your results?
  - Milestones – what is your expected progress schedule? Provide a weekly / bi-weekly basis

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

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
    \[
    y_i = w_1 \cdot x_i + w_2 \cdot x_{i+1} + \cdots + w_k \cdot x_{i+k-1}
    \]
    \[
    \equiv \sum_{j=1}^{k} w_j \cdot x_{i+j-1}
    \]
    for (i=1; i<=n; i++)
    for (j=1; j <=k; j++)
    \[
    y[i] += w[j] \cdot x[i+j-1];
    \]
Finite Impulse Response (cont.)

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

Example – Matrix-Vector Product

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

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

Example – Relational Database

- Relation is a collection of tuples that all have the same attributes
  - Tuple is a fixed number of objects
  - Represented in a table

<table>
<thead>
<tr>
<th>Tuple</th>
<th>Name</th>
<th>School</th>
<th>Age</th>
<th>QB Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>D. Carr</td>
<td>Fresno State</td>
<td>27</td>
<td>113.6</td>
</tr>
<tr>
<td>1</td>
<td>P. Rivers</td>
<td>NC State</td>
<td>24</td>
<td>107.4</td>
</tr>
<tr>
<td>2</td>
<td>D. McNabb</td>
<td>Syracuse</td>
<td>29</td>
<td>105.3</td>
</tr>
<tr>
<td>3</td>
<td>C. Pennington</td>
<td>Marshall</td>
<td>30</td>
<td>103.4</td>
</tr>
<tr>
<td>4</td>
<td>R. Grossman</td>
<td>Florida</td>
<td>26</td>
<td>100.9</td>
</tr>
</tbody>
</table>

Outline

- Project Proposals
- Recap
- Non-Numeric Systolic Examples
- Systolic Loop Transformations
  - Data dependencies
  - Iteration spaces
  - Example transformations
- Reading – Cellular Automata
- Reading – Bit-Serial Architectures

Database Operations

- Intersection: $A \cap B$ – all records in both relation $A$ and $B$
- Must compare all $|A| \times |B|$ tuples
- Compare via sequence compare
- Or along row or column to get inclusion bitvector
Database Operations (cont.)

- Tuple Comparison
  - Problem – tuples are long, comparison time might limit computation rate
  - Strategy – perform comparison in pipelined manner by fields
    - Stagger fields
    - Arrange to compute field $i$ on cycle after $i-1$
    - Cell:
      \[
      \text{tout} = \text{tin} \oplus \text{ain} \oplus \text{bin}
      \]

Database Intersection (cont.)

- Database Intersection
  - True $\land \text{A}_{i,1} \land \text{B}_{i,1}$
  - True $\land \text{A}_{i,2} \land \text{B}_{i,2}$
  - True $\land \text{A}_{i,3} \land \text{B}_{i,3}$
  - True $\land \text{A}_{i,4} \land \text{B}_{i,4}$
  - $\lor$ $\text{OR}$ $\text{OR}$ $\text{OR}$ $\text{OR}$
  - $\text{T}_1$ $\text{T}_2$ $\text{T}_3$

Database Operations (cont.)

- Unique: remove duplicate elements in multirelation A
  - Intersect A with A
- Union: $A \cup B$ – one copy of all tuples in A and B
  - Concatenate A and B
  - Use Unique to remove duplicates
- Projection: collapse A by removing select fields of every tuple
  - Sample fields in A'
  - Use Unique to remove duplicates

Database Summary

- Input database – $O(n)$ data
- Operations require $O(n^2)$ data
  - $O(n)$ if sorted first
  - $O(n \log(n))$ to sort
- Systolic implementation – works on $O(n)$ processing elements in $O(n)$ time
- Typical database [KunLoh80A]:
  - 1500 bit tuples
  - 10,000 records in a relation
  - ~1 4-LUT per bit-compare
  - ~1600 XC4062 FPGAs
  - ~84 XC4LX200 FPGAs
Systolic Loop Transformations

- Automatically re-structure code for
  - Parallelism
  - Locality
- Driven by dependency analysis

Defining Dependencies

- Flow Dependence \( W \rightarrow R \ \delta_f \) \ true
- Anti-Dependence \( R \rightarrow W \ \delta_a \) \ false
- Output Dependence \( W \rightarrow W \ \delta_o \)
- Input Dependence \( R \rightarrow R \ \delta_i \)

S1) \( a = 0; \)
S2) \( b = a; \)
S3) \( c = a + d + e; \)
S4) \( d = b; \)
S5) \( b = 5 + e \)

Example Dependencies

1) \( S1 \delta_f S2 \) due to \( a \)
2) \( S1 \delta_f S3 \) due to \( a \)
3) \( S2 \delta_f S4 \) due to \( b \)
4) \( S3 \delta_a S4 \) due to \( d \)
5) \( S4 \delta_a S5 \) due to \( b \)
6) \( S2 \delta_i S5 \) due to \( e \)

Data Dependencies in Loops

- Dependence can flow across iterations of the loop
- Dependence information is annotated with iteration information
- If dependence is across iterations it is loop carried otherwise loop independent

```
for (i=0; i<n; i++) {
  A[i] = B[i];
  B[i+1] = A[i];
}
```

Unroll Loop to Find Dependencies

```
for (i=0; i<n; i++) {
  A[i] = B[i];
  B[i+1] = A[i];
}
```

Thought Exercise

Consider the Laplace Transformation: \( L(f) = F(s) = \int_0^\infty e^{-st} f(t) dt \)

```
for (i=1; i<=N; i++)
  for (j=1; j<=N; j++)
    c = -4*a[i][j] + a[i-1][j] + a[i+1][j];
    c += a[i][j+1] + a[i][j+1];
    b[i][j] = c;
}
```

- In teams of two, try to determine the flow dependencies, anti dependencies, output dependencies, and input dependencies
- Use loop unrolling to find dependencies
- Most dependencies found gets a prize
Iteration Space

- Every iteration generates a point in an \( n \)-dimensional space, where \( n \) is the depth of the loop nest

\[
\begin{align*}
\text{for } (i=0; i<n; i++) & \{
\quad [4] \\
\quad \ldots
\}\text{ for } (i=0; i<n; i++) & \{
\quad [3; 2] \\
\quad \ldots
\}
\end{align*}
\]

Distance Vectors

- The difference between the target and source iterations

\[
d = I_t - I_s
\]

Exactly the distance of the dependence, i.e.,

\[
I_s + d = I_t
\]

FIR Distance Vectors

- Creates new flow dependencies

\[
D = \begin{bmatrix} Y & W & X \\ 0 & 1 & 1 \\ 1 & 0 & -1 \end{bmatrix}
\]

Re-label / Pipeline Variables

- Remove anti-dependencies and input dependencies by relabeling or pipelining variables

\[
\begin{align*}
\text{for } (i=0; i<n; i++) & \{
\quad [4] \\
\quad \ldots
\}\text{ for } (j=0; j<m; j++) & \{
\quad [3; 2] \\
\quad \ldots
\}
\end{align*}
\]

FIR Dependencies
Transforming to Time and Space

- Using data dependencies, find T
- T defines a mapping of the iteration space into a time component π, and a space component, S
- \( T = [\pi; S] \)
  - If \( \pi \cdot I_1 = \pi \cdot I_2 \), then \( I_1 \) and \( I_2 \) execute at the same time
  - \( \pi \cdot d \) – amount of time units to move data items (\( \pi \cdot d > 0 \))
  - Any S can be picked that makes T a bijection
- See [Mol83A] for more details

Calculating T for FIR

- For \( \pi = [p_1, p_2] \)
- Since \( \pi \cdot d > 0 \), we see that:
  - \( p_2 \neq 0 \) (from Y)
  - \( p_1 \neq 0 \) (from W)
  - \( p_1 > p_2 \) (from X)
- Smallest solution \( \pi = [2, 1] \)
- S can be \([0, 1], [1, 0], [1, 1] \)

An Example Transformation

\[
\begin{bmatrix}
0 & 1 & 1 \\
1 & 0 & 1
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 & 1 & 1 \\
1 & 0 & 1
\end{bmatrix}
\]

An Example Transformation (cont.)

\[
T = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}
\]

\[
T \cdot \begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} 2i + j \\ i + j \end{bmatrix}
\]

Summary

- Non-numeric (database ops) example of systolic computing
  - Multiple use of each input data item
  - Concurrency
  - Regular data and control flow
- Loop transformations
  - Data dependency analysis
  - Restructure code for parallelism, locality